A Multivariate Bayesian Approach to Modeling Vulnerability Discovery in the Software Security Lifecycle

by Reuben Aaron Johnston

B.S. in Electrical Engineering, June 2000, Lamar University
B.S. in Computer Science, June 2000, Lamar University
M.S. in Systems Engineering, June 2011, The George Washington University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

August 31, 2018

Dissertation directed by

Shahryar Sarkani
Professor of Engineering Management and Systems Engineering

Thomas Holzer
Professor of Engineering Management and Systems Engineering

Timothy Eveleigh
Professor of Engineering Management and Systems Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Reuben Aaron Johnston has passed the Final Examination for the degree of Doctor of Philosophy as of May 31st, 2018.

This is the final and approved form of the dissertation.

A Multivariate Bayesian Approach to Modeling Vulnerability Discovery in the Software Security Lifecycle

Reuben Aaron Johnston

Dissertation Research Committee:

Shahryar Sarkani, Adjunct Professor of Engineering Management, Dissertation Director

Thomas Holzer, Professorial Lecturer in Engineering Management and Systems Engineering, Committee Member

Thomas Mazzuchi, Professor of Engineering Management and Systems Engineering and of Decision Sciences, Committee Member

Shahram Sarkani, Professor of Engineering Management and Systems Engineering, Committee Member

Ronald Iammartino, Professorial Lecturer of Engineering Management and Systems Engineering, Committee Member

© Copyright 2018 by Reuben Aaron Johnston
All rights reserved

Dedication

To my most special Bayesian posteriors, Clara Aida and Vivien Leigh: Go forth and *make* your academic dreams come true. *All* of your priors believe in you!

In memory of my father-in-law, George Williams, and my grandparents, Jessie Johnston, Archie Johnston, and Alec Dyer

Acknowledgments

To my wife, Meghan: During this journey, we experienced the births of our daughters, the loss of a father, the loss of grandparents, the loss of friends, the loss and addition of pets, career changes, panic attacks, house renovations and moving, and so much more. Throughout it all you were there and running the household while working full-time. My achievement was not possible at this point in our lives without your sacrifices. Thank you for everything you have done.

To my parents, Rick and Cathy Johnston, and my grandmother, Aida Dyer: Thank you for our family's culture of academic excellence that inspired me while I was young and for all the encouragement you provided during my graduate studies, my research, and the writing of this dissertation.

To my mother-in-law, Diana Williams: Thank you for your encouragement and for helping with Clara and Vivien on the many weekends I needed to work on this research.

To all my academic advisors: Thank you for your guidance, for understanding the quality of the research, and for trusting in me to be successful—long past the scheduled milestones.

To my Professor, Thomas Mazzuchi; and my associate, Professor Refik Soyer: Thank you for the useful suggestions and enjoyable discussions that led to realizing key elements in the Bayesian analysis.

To the Workshop Participants: Thank you for supporting my research. Its success would, truly, not have been possible without your efforts.

To my associate, Professor Roger Cooke, and my Professor and colleague, Stacy Hill: Thank you for your helpful reviews of the methodology and preliminary results.

To my former colleague, Steve Lipner: Thank you for the useful and enjoyable discussions on the SSL and for your related reviews of the material.

To the editors and reviewers of Elsevier's Reliability Engineering and System Safety journal: Thank you for the suggestions that improved the presentation of the research.

To my brother-in-law, William Peterson; editors, Claire Harrison and Jim Young; all my JHU-APL colleagues who assisted; and my JHU-ISI students: Thank you for the useful suggestions and advice.

To my associate, Professor J.P. Blackford, and the George Washington University Institutional Review Board: Thank you for your assistance in the Institutional Review Board approval process.

To my colleague, Daniel Anna: Thank you for helping me get through the approval process for eliciting volunteers.

To the Security Research Group at University of Trento-Italy: Thank you for sharing the empirical datasets with me.

To my employer, JHU-APL: Thank you for supporting my goal to achieve this Ph.D. degree.

Abstract

A Multivariate Bayesian Approach to Modeling Vulnerability Discovery in the Software Security Lifecycle

Software vulnerabilities that enable well-known exploit techniques for committing computer crimes are *preventable*, but they continue to be present in releases. When Blackhats (i.e., malicious researchers) discover these vulnerabilities they oftentimes release corresponding exploit software and malware. If vulnerabilities—or discoveries of them—are not prevented, mitigated, or addressed, customer confidence could be reduced. In addressing the issue, software-makers must choose which mitigation alternatives will provide maximal impact and use vulnerability discovery modeling (VDM) techniques to support their decision-making process. In the literature, applications of these techniques have used traditional approaches to analysis and, despite the dearth of data, have not included information from experts and do not include influential variables describing the software release (SR) (e.g., code size and complexity characteristics) and security assessment profile (SAP) (e.g., security team size or skill). Consequently, they have been limited to modeling discoveries over time for SR and SAP scenarios of unique products, whose results are not readily comparable without making assumptions that equate all SR and SAP combinations under study. This research takes an alternative approach, applying Bayesian methods to modeling the vulnerability-discovery phenomenon. Relevant data were obtained from expert judgment (i.e., information elicited from security experts in structured workshops) and from public databases. The open-source framework, MCMCBayes, was developed to perform Bayesian model averaging (BMA). It combines predictions of interval-grouped discoveries by performance-weighting results from six variants of the non-homogeneous Poisson process, two regression models, and two growth-curve models. Utilizing expert judgment also enables forecasting expected discoveries over time for arbitrary SR and SAP combinations, thus helping software-makers to better understand the effects of influential variables they control on the phenomenon. This requires defining variables that describe arbitrary SR and SAP combinations as well as constructing VDM extensions that parametrically scale results from a defined baseline SR and SAP to the arbitrary SR and SAP of interest. Scaling parameters were estimated using elicited multivariate data gathered with a novel paired comparison approach. MCMCBayes uses the multivariate data with the BMA model for the baseline to perform predictions for desired SR and SAP combinations and to demonstrate how multivariate VDM techniques could be used. The research is applicable to software-makers and persons interested in applications of expert-judgment elicitation or those using Bayesian analysis techniques with phenomena having non-decreasing counts over time.

Table of Contents

Dedication iv

Acknowledgments v

Abstract vii

Table of Contents viii

List of Figures xi

List of Tables xiii

List of Acronyms xvi

List of Symbols xviii

Glossary of Terms xxiv

Chapter 1. Introduction and Background 1

1.1 The security problem 1

1.2 Software security lifecycle (SSL) 3

1.2.1 Product lifecycle 3

1.2.2 Software vulnerability states 5

1.2.3 SSL model 8

1.3 Strategies to reduce risk 18

Chapter 2. Literature Review 20

2.1 Security modeling origins 20

2.2 Vulnerability discovery modeling (VDM) 21

2.3 Improving VDM techniques 23

Chapter 3. Research conceptual framework 25

3.1 Questions 25

3.2 Hypotheses 25

3.3 Purpose and usefulness 26

3.4 Assumptions 26

3.5 Scope and perspective 28

3.6 Conceptual model 29

3.7 Variables 31

Chapter 4. Methodology 37

4.1 Introduction 37

4.2 Approach fundamentals 38

4.2.1 Cooke's method 40

4.2.2 Bayesian analysis 42

4.3 - Data gathering 45
4.3.1 Expert validation and calibration 45
4.3.2 Phase I ("black-box") 50
4.3.3 Phase II ("clear-box") 53
4.4 Bayesian analysis of VDM techniques 55
4.4.1 Phase I ("black-box") 56
4.4.2 Phase II ("clear-box") 63
4.5 MCMCBayes modeling software 65
Chapter 5. Results and discussion 68
5.1 Cleansing 68
5.2 Calibration 70
5.3 Phase I ("black-box") 71
5.3.1 Elicitation and data aggregation 71
5.3.2 Bayesian analysis highlights 74
5.4 Phase II ("clear-box") 77
5.4.1 Elicitation and data aggregation 77
5.4.2 Bayesian analysis highlights 78
Chapter 6. Limitations, usage recommendations, and future work 83
Chapter 7. Conclusion 87
References 88
Appendix A. Probability distributions 101
A.1 Exponential 101
A.2 Gamma 101
A.3 Normal 101
A.4 Uniform 102
A.5 Piecewise linear 102
A.6 Chi-squared 102
Appendix B. Arrival processes 103
B.1 Arrival process overview 103
B.2 Nonhomogeneous Poisson process overview 104
B.3 Nonstationary GAMMA process overview 105
Appendix C. Scoring in Cooke's method 107
Appendix D. Traditional analysis techniques 111
D.1 Maximum likelihood estimation (MLE) 111
D.2 Method of least squares (LS) 111

Appendix E. Simulation techniques 112
E.1 Inverse transform method, discrete form 112
E.2 Simulating interval counts in nonstationary Gamma processes 113
E.3 Rejection and squeezed rejection sampling 114
E.4 Markov chain Monte Carlo (MCMC) 115
E.4.1 MCMC overview 115
E.4.2 MCMC chain construction 116
E.4.3 MCMC summary statistics 138
E.4.4 MCMC diagnostics 139
Appendix F. Model choice techniques 141
F.1 Power-posterior approach 141
F.2 Kullback-Leibler distance 143
Appendix G. Elicitation results supplement 144
Appendix H. Calibration and aggregation results supplement 147
Appendix I. Bayesian analysis results supplement 166

List of Figures

Figure 1-1: Vulnerability discoveries per year, 1996-2016 2
Figure 1-2: Externalities with reputation presented by van Eeten and Bauer 3
Figure 1-3: ANSI/EIA-724 Product Lifecycle Model 4
Figure 1-4: Vulnerability lifecycle states in the SSL 6
Figure 1-5: Vulnerability lifecycle state sequences in the SSL 7
Figure 1-6: SSL iterations within each product lifecycle model stage 9
Figure 1-7: Software security lifecycle from the software-maker perspective 9
Figure 1-8: Iterative and incremental development model for a major release cycle 11
Figure 1-9: Iterative and incremental development model for an update release cycle 11
Figure 1-10: Iterative and incremental software development process 12
Figure 1-11: Iterative software security assessment process (ISSAP) 15
Figure 1-12: ISO/IEC 29147, ISO/IEC 30111 processes 16
Figure 1-13: Vulnerability resolution and remediation process 18
Figure 1-14: Mitigation alternatives and application areas 19
Figure 3-1: Security researcher roles (expert population) 29
Figure 3-2: Proposed conceptual model and relevant factors 30
Figure 4-1: Methodology Overview 39
Figure 4-2: Expert judgment steps E1-E4 40
Figure 4-3: Bayesian analysis steps AN1-AN10 44
Figure 4-4: CarRace 47
Figure 4-5: ChatClient 47
Figure 4-6: Calibration module answer format example 50
Figure 4-7: Baseline elicitation module answer format example 52
Figure 4-8: Discovery rates supported by the individual models 57
Figure 5-1: Example expert-response of probability distribution bins 72
Figure 5-2: Elicited cumulative discoveries over time, example 1 73
Figure 5-3: Example prior-based predictions for cumulative discoveries over time 75
Figure 5-4: Example posterior-based predictions for cumulative discoveries over time 76
Figure 5-5: Posterior-based averaged predictions for cumulative discoveries over time, example 1 77
Figure 5-6: Linearly-scaled predictions for cumulative discoveries over time, example 1 80
Figure 5-7: Linearly-scaled predictions for cumulative discoveries over time, example 2 81

Figure 5-8: Linearly-scaled predictions for cumulative discoveries over time, example 3 82
Figure 6-1: Total discoveries in the first 50 weeks following a release vs. release number 84
Figure B-1: Arrival process example 103
Figure C-1: Comparison of expert distribution with background measure 109
Figure E-1: (Log-scale) ARS envelope and squeezing around initial points (P1, P2, and P3) 122
Figure E-2: (Natural-scale) Exponentiated form of Figure E-1 123
Figure E-3: (Log-scale) ARS envelope and squeezing around initial points and a new one, P4 124
Figure E-4: (Natural-scale) Exponentiated form of Figure E-3 125
Figure H-1: Elicited cumulative discoveries over time, example 2 153
Figure H-2: Elicited cumulative discoveries over time, example 3 154
Figure H-3: Elicited cumulative discoveries over time, example 4 154
Figure H-4: Elicited cumulative discoveries over time, example 5 155
Figure I-1: Posterior-based averaged predictions for cumulative discoveries over time, example 2 173
Figure I-2: Posterior-based averaged predictions for cumulative discoveries over time, example 3 174
Figure I-3: Posterior-based averaged predictions for cumulative discoveries over time, example 4 174
Figure I-4: Posterior-based averaged predictions for cumulative discoveries over time, example 5 175

List of Tables

Table 1-1: Notional summaries for stages in the ANSI-EIA-724 Product Lifecycle Model 4

Table 1-2: SSL stage activities from the software-maker perspective 10

Table 1-3: IID activity descriptions 13

Table 1-4: IID phase descriptions 13

Table 1-5: ISSAP steps 15

Table 2-1: VDM technique cross reference 21

Table 3-1: Subset of hypotheses associated with RQ2 25

Table 3-2: Key assumptions 27

Table 3-3: Research variable overview 31

Table 3-4: Select variables that are pertinent to security assessment 33

Table 4-1: Experience questions 46

Table 4-2: Practical assessment exercise questions 48

Table 4-3: Calibration questions 48

Table 4-4: Calibration question answer format 49

Table 4-5: Expert instructions for the calibration exercises 49

Table 4-6: Calibration answer format explanation 50

Table 4-7: Data elicitation answer format for a single SR and SAP, with example data entries 51

Table 4-8: Expert instructions for baseline SR and SAP elicitation 51

Table 4-9: Baseline elicitation module answer format explanation 52

Table 4-10: Empirical datasets for cumulative discoveries over time 53

Table 4-11: An example question appearing in one of the scenario sets 54

Table 4-12: Covariates of interest in $\boldsymbol{x}j1, \boldsymbol{x}j2$ and their values used 54

Table 4-13: Pairwise comparison elicitation instructions 55

Table 4-14: Vulnerability discovery models 57

Table 5-1: Expert experience 68

Table 5-2: Practical assessment results 69

Table 5-3: Expert scores ranked using Cooke's method 70

Table 5-4: Data results from simulating aggregated expert distributions 73

Table 5-5: Example pairwise responses by individual experts 78

Table 5-6: Parameter estimates for the linear scaling term 78

Table B-1: NHPP axioms 104

Table E-1: Pseudo-code for simulating grouped-interval data from a NHPP via inverse transform 112
Table E-2: Pseudo-code for sampling from a GAMMA process in JAGS/BUGS 113
Table E-3: Pseudo-code for Rejection and SqueezedRejection sampling 114
Table E-4: General MCMC application for a single parameter 115
Table E-5: Pseudo-code for Metropolis-Hastings sampling 117
Table E-6: Pseudo-code for single-component MH sampling 118
Table E-7: Pseudo-code for standard Gibbs sampling 119
Table E-8: Pseudo-code for ARS within Gibbs sampling 121
Table E-9: Pseudo-code for ARMS within Gibbs sampling 126
Table E-10: Pseudo-code for SliceSampler (univariate) 127
Table E-11: Pseudo-code updates for slice sampling within Gibbs 128
Table E-12: Pseudo-code for Hamiltonian Monte Carlo sampling 130
Table E-13: Pseudo-code for Leapfrog function 131
Table E-14: Pseudo-code for FindReasonableEpsilon heuristic function 132
Table E-15: Pseudo-code for Hamiltonian Monte Carlo sampler with dual-averaging 132
Table E-16: Pseudo-code for efficient No-U-Turn Sampler with dual averaging 134
Table E-17: Pseudo-code for recursive BuildTree function 136
Table E-18: Pseudo-code for highest posterior density estimation 138
Table F-1: Pseudo-code for power-posterior sequence 142
Table G-1: $\boldsymbol{xj}1$ scenarios 144
Table G-2: $\boldsymbol{xj}2$ scenarios 145
Table H-1: Excalibur "cleansed.dtt" project file containing the expert data 147
Table H-2: Excalibur "cleansed.rls" file containing seed question realizations 148
Table H-3: CCM range graph (items) 149
Table H-4: Data results from simulating individual expert distributions 153
Table H-5: Expert 1 $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 155
Table H-6: Expert 2 $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 156
Table H-7: Expert 3 $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 158
Table H-8: Expert 4 $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 159
Table H-9: Expert 5 $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 161
Table H-10: Expert 6 $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 162
Table H-11: Aggregated $\boldsymbol{D}1/\boldsymbol{D}2$ dataset 163
Table I-1: Parameter estimates for $\gamma = 148$ and release 19.0 in Mozilla Firefox 166
Table I-2: Parameter estimates for $\gamma = 55$ and release 3.0 in Firefox 167

Table I-3: Parameter estimates for $\gamma = 20$ and release 7.0 in Internet Explorer 169
Table I-4: Parameter estimates for $\gamma = 7$ and release 6.0 in Internet Explorer 170
Table I-5: Parameter estimates for $\gamma = 3$ and release 1.0 in Safari 172
Table I-6: Model choice for $\gamma = 148$ and release 19.0 in Mozilla Firefox 175
Table I-7: Model choice for $\gamma = 55$ and release 3.0 in Firefox 176
Table I-8: Model choice for $\gamma = 20$ and release 7.0 in Internet Explorer 177
Table I-9: Model choice for $\gamma = 7$ and release 6.0 in Internet Explorer 177
Table I-10: Model choice for $\gamma = 3$ and release 1.0 in Safari 178
Table I-11: Parameter estimates for the exponential scaling term 179
Table I-12: Example MATLAB session 179

List of Acronyms

1.	ACF	Auto-correlation function
2.	ALOC	Assembly language lines of code
3.	ANSI	American National Standards Institute
4.	ARMS	Adaptive rejection Metropolis sampling
5.	ARS	Adaptive rejection sampling
6.	BF	Bayes factor
7.	BMA	Bayesian model averaging
8.	BUGS	Bayesian inference using Gibbs sampling
9.	CC	Cyclomatic complexity
10.	CCM	Cooke classical model
11.	cdf	Cumulative distribution function
12.	CI	Credible interval
13.	CMMI	Capability maturity model integrated
14.	CPU	Central processing unit
15.	CVE	Common Vulnerabilities and Exposures
16.	CWE	Common Weakness Enumeration
17.	DM	Decision-maker
18.	EIA	Electronic Industries Association
19.	FTE	Full-time equivalent
20.	IID	Iterative and incremental software development
21.	HMC	Hamiltonian Monte Carlo
22.	HPD	Highest posterior density
23.	HPP	Homogeneous Poisson process
24.	IF	Information flow
25.	i.i.d.	Independent and identically distributed
26.	ISSAP	Iterative software security assessment process
27.	JAGS	Just another Gibbs sampler
28.	KL	Kullback-Leibler
29.	KLOC	Kilo-lines of code
30.	KM	Knowledge management
31.	LS	Least squares
32.	Max	Maximum
33.	MCE	Monte Carlo error
34.	MCMC	Markov chain Monte Carlo
35.	MH	Metropolis-Hastings

36.	Min	Minimum
37.	MLE	Maximum likelihood estimation
38.	MSFE	Mean-square forecasting error
39.	MVF	Mean-value function
40.	NA	Not applicable
41.	NHPP	Nonhomogeneous Poisson process
42.	NUTS	No U-Turn sampler
43.	NVD	National Vulnerability Database
44.	OP	Opaque predicates; operational profile
45.	OR	Operations research
46.	OS	Operating system
47.	PC	Personal computer
48.	pdf	Probability density function
49.	pmf	Probability mass function
50.	RE	Reverse engineering
51.	ROI	Return on investment
52.	r.v. (r.v.s)	Random variable (Random variables)
53.	SA	Security assessment
54.	SAP	Security assessment profile
55.	SD	Standard deviation
56.	SDL	Security development lifecycle
57.	SQL	Structured query language
58.	SSL	Software security lifecycle
59.	SR	Software release
60.	SR and SAP	Software release and security assessment profile
61.	US	United States
62.	VDM	Vulnerability discovery modeling

List of Symbols

1. AreaX — Numbered categories (e.g., Area1 and Area2; see Figure 1-14) for methods that managers use to control the phenomenon
2. ANX — Analysis steps AN1-10
3. ASX — Numbered research assumptions (e.g., AS1-10; see Table 3-2)
4. DG — Empirical data gathering process step
5. EJX — Expert judgment process steps EJ1-4
6. GX — Numbered conceptual groups in the conceptual model (e.g., G1-8; see Figure 3-2)
7. RHX — Numbered research hypotheses (e.g., RH1, RH2.1-10, RH3, and RH4)
8. RQX — Numbered research questions (e.g., RQ1, RQ2, RQ3, and RQ4)
9. VQX — Numbered expert validation questions used in the elicitation (e.g., VQ1-8; see Table 4-1)
10. t — Time
11. τ — Assessment time
12. $[t_{i-1}, t_i)$ — i^{th} time interval for t
13. $[\tau_{i-1}, \tau_i)$ — i^{th} time interval for τ
14. $\boldsymbol{x}_0$ — Vector of variables that defines the $nx1$ baseline SR and SAP combination $(x_{0_1} \quad x_{0_2} \quad \cdots \quad x_{0_n})^T$
15. x_{0_X} — Individual variables in the baseline SR and SAP (e.g., $x_{0_1}, x_{0_2}, \cdots, x_{0_n}$)
16. $\boldsymbol{x}_i$ — Vector of variables that defines the $nx1$ i^{th} SR and SAP combination $(x_{i_1} \quad x_{i_2} \quad \cdots \quad x_{i_n})^T$
17. x_{i_X} — Individual variables in the ith SR and SAP (e.g., $x_{i_1}, x_{i_2}, \cdots, x_{i_n}$)
18. n — Number of variables in $\boldsymbol{x}$
19. γ — Assumed number of vulnerabilities in a SR ($\gamma = 148{,}55{,}20{,}7{,}3$)
20. $\{\boldsymbol{x}_0, \gamma\}$ — Baseline SR and SAP with assumed SR quality
21. $\boldsymbol{x}_j^1$ — First SR and SAP in the j^{th} paired comparison
22. $\boldsymbol{x}_j^2$ — Second SR and SAP in the j^{th} paired comparison
23. $\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\}$ — j^{th} SR and SAP paired comparison that associates with D_j^1 / D_j^2
24. $\tilde{\boldsymbol{x}}_j$ — Baseline normalized, j^{th} SR and SAP
25. $\tilde{x}_{j_k}$ — Baseline normalized, k^{th} covariate in the j^{th} SR and SAP
26. $\boldsymbol{X}$ — $n \times n$ scenario matrix containing all defined SR and SAP combinations
27. $N(t)$ — Cumulative discoveries over time
28. n_j — Cumulative discoveries at the respective time instance t_j (i.e., $N(t_j)$)
29. r_i — Grouped discoveries in the time interval $[t_{i-1}, t_i)$ (i.e., $N(t_i) - N(t_{i-1})$)
30. $N(t|\boldsymbol{x}_0)$ — Cumulative discoveries over time given the baseline SR and SAP
31. $\boldsymbol{D}_0$ — Set of m data points in the baseline SR and SAP grouped time interval dataset that estimates $E[N(t|\boldsymbol{x}_0)]$; includes expert judgment and empirical data
32. $\mathcal{I}$ — Number of intervals in $\boldsymbol{D}_0$

33. m Number of data points in $\boldsymbol{D}_0$; assuming $N(t=0|\boldsymbol{x}_0)=0$, this is a multiple of $\mathcal{I}+1$
34. $\frac{N(\boldsymbol{x}_j^1)}{N(\boldsymbol{x}_j^2)}$ Ratio of cumulative discoveries over time given the j^{th} paired set of SR and SAP combinations
35. $\boldsymbol{D}^1/\boldsymbol{D}^2$ Set of n answers D_j^1/D_j^2 to the elements in the paired questionnaire
36. n Number of paired questions (data points for $\boldsymbol{D}^1/\boldsymbol{D}^2$)
37. D_j^1/D_j^2 $E[N(\boldsymbol{x}_j^1)]/E[N(\boldsymbol{x}_j^2)]$ answer to the j^{th} paired comparison $\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\}$
38. y_j D_j^1/D_j^2
39. z_j $\ln(y_j)$
40. $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ Subset of $\boldsymbol{D}_0$ that stemmed from expert opinion
41. $\boldsymbol{D}_{w_\alpha}$ $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ derived using $w_\alpha(e)$ performance weights (Cooke's method)
42. $\boldsymbol{D}_{w_{eq}}$ $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ derived using $w_{eq}(e)$ equal weights
43. $f_0(t)$ Baseline function for discoveries over time in the baseline SR and SAP (i.e., $f(t|\boldsymbol{x}_0)$)
44. $f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)$ Function for discoveries over time in the i^{th} SR and SAP
45. $s(\boldsymbol{x}_i)$ Modulation function for the i^{th} SR and SAP that is used to scale the baseline function
46. Δ Some observable (e.g., prediction) for a phenomenon
47. Δ_0 Phenomenon observables in the baseline SR and SAP
48. Δ_i Phenomenon observables in the i^{th} SR and SAP
49. Δ_S Scaling term observables
50. $Pr(\Delta|\boldsymbol{D})$ Posterior probability of an observable given the data
51. $F(\boldsymbol{\theta})$ Candidate function for the analysis that is within $Pr(\Delta|\boldsymbol{D})$
52. $\boldsymbol{\theta}$ Set of variables (or parameters) in the candidate function
53. $\pi(\cdot)$ $Pr(\cdot)$ for functions of $\boldsymbol{\theta}$
54. $\pi(\boldsymbol{\theta})$ Joint, prior distribution for the variables in the candidate function
55. $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})$ Likelihood function of the parameters given the data
56. $\log\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})$ Log-Likelihood function of the parameters given the data
57. $\pi(\boldsymbol{\theta}|\boldsymbol{D})$ Joint, posterior distribution for the variables in the candidate function given the data
58. $\pi(\theta_j|\boldsymbol{\theta}_{-j}, \boldsymbol{D})$ Full-conditional posterior distribution of the j^{th} variable that conditions on all the other variables and the data
59. $\mathrm{q}_{\mathrm{X}\%}$ Thresholds for a set of quantiles that contribute to defining a distribution using expert judgment (e.g., $\{\mathrm{q}_{0\%}, \mathrm{q}_{5\%}, \mathrm{q}_{50\%}, \mathrm{q}_{95\%}, \mathrm{q}_{100\%}\}$) (Cooke's method)
60. bin_x Probability quantiles, bin_1, bin_2, bin_3, bin_4, that are bounded using $\{\mathrm{q}_{0\%}, \mathrm{q}_{5\%}, \mathrm{q}_{50\%}, \mathrm{q}_{95\%}, \mathrm{q}_{100\%}\}$ (Cooke's method)
61. $\mathcal{S}$ Number of seed (or calibration) questions (Cooke's method)
62. E Number of experts (Cooke's method)
63. e Expert identifier (Cooke's method)
64. Ψ Quantitative answer to a seed question (Cooke's method)

65. Ψ_i Quantitative answer to the i^{th} seed question (Cooke's method)
66. i Seed item identifier (Cooke's method)
67. q_L Maximum lower bound for Ψ (Cooke's method)
68. q_U Maximum upper bound for Ψ (Cooke's method)
69. k $k\%$ intrinsic range for q_L and q_U (Cooke's method)
70. $C(e)$ Calibration score for expert e (Cooke's method)
71. ρ Calibration power (Cooke's method)
72. $I(e)$ Information score for expert e (Cooke's method)
73. $I(e,i)$ Individual expert e's information score for seed-item i (Cooke's method)
74. $h(e,i)$ Expert e's density function for seed-item i (Cooke's method)
75. A Quality level cutoff for experts that defines a factor multiplied against each weight (i.e., $1_{\mathrm{A}}(\mathrm{x}) = 0$ if $\mathrm{x} < \mathrm{A}$ and $1_{\mathrm{A}}(\mathrm{x}) = 1$ otherwise) (Cooke's method)
76. $w_{\mathrm{A}}(e)$ Global weight for the expert e (Cooke's method)
77. $w_{\mathrm{A}}(e,i)$ Item weight for the expert e (Cooke's method)
78. $w_{eq}(e)$ Equal weight for the expert e
79. DM_{A} Global weight decision maker (Cooke's method)
80. IDM_{A} Item weight decision maker (Cooke's method)
81. DM_{eq} Equal weight decision maker (Cooke's method)
82. $F_0(t;\boldsymbol{\theta})$ General model form (for $f_0(t)$) that is a function with a set of model variables $\boldsymbol{\theta}$ that represents the expected discoveries over time given $\boldsymbol{x}_0$
83. $H_0(t;\boldsymbol{\theta})$ Temporal parametric model form of F_0 with a set of parameters $\boldsymbol{\theta}$
84. u, ζ, κ Parameters in parametric NHPP candidate models
85. $\beta_0, \beta_1, \beta_2, r$ Parameters in regression and growth curve candidate models; r is also used for momentum variables in NUTS
86. a, b, c, d Hyperparameters in distributions for parametric variables
87. g, h, p, q Hyperparameters in distributions for parametric variables
88. ϵ Zero-mean error term for some of the parametric models
89. $G_0(t;\Theta(t))$ Temporal non-parametric form of F_0 with a stochastic process $\Theta(t)$
90. $\Theta(t)$ Variable in non-parametric NHPP candidate models for expected discoveries over time
91. r_i Variable in non-parametric NHPP candidate models for expected grouped discoveries in interval i
92. $\Lambda^*(t), c$ Hyperparameters in distributions for non-parametric NHPP candidate model variables
93. $S(\boldsymbol{x}_i;\boldsymbol{c})$ Parametric model form of $s(\boldsymbol{x}_i)$ with the $n \times 1$ parameter vector $\boldsymbol{c}$
94. $\boldsymbol{c}$ $n \times 1$ parameter vector for the scaling functions
95. $Pr(\Delta|M_k, \boldsymbol{D})$ Posterior probability of an observable given the k^{th} model and data
96. M_k k^{th} model amongst a group of model candidates
97. k Candidate model identifier
98. K Number of candidate models

99. $\mathcal{L}(\boldsymbol{\theta}_{\hbar}|M_{\hbar},\boldsymbol{D})$ Likelihood function for variables in the $\hbar^{\text{th}}$ model
100. $\boldsymbol{\theta}_{\hbar}$ Set of parameters for the $\hbar^{\text{th}}$ model, M_k
101. $\pi(\boldsymbol{\theta}_{\hbar}|M_{\hbar})$ Joint, prior distribution for the variables in the $\hbar^{\text{th}}$ model
102. $\pi(\boldsymbol{\theta}_{\hbar}|M_{\hbar},\boldsymbol{D})$ Joint, posterior distribution for the variables in the $\hbar^{\text{th}}$ model
103. $Pr(M_{\hbar}|\boldsymbol{D})$ Posterior probability for the $\hbar^{\text{th}}$ model given the data
104. $Pr(M_k)$ Prior probability for the $\hbar^{\text{th}}$ model
105. $Pr(\boldsymbol{D}|M_{\hbar})$ Marginal likelihood of the data given the $\hbar^{\text{th}}$ model
106. $f_X(x)$ Probability density function for X (i.e., $Pr(x < X < x + dx) = f(x) \cdot dx$, for infinitely small dx)
107. $F_X(x)$ Cumulative distribution function (cdf) for X (i.e., $Pr(X \leq x)$)
108. $F_U(x)$ Uniform distribution's cdf
109. $E[\cdot]$ Expected value for a probability distribution or sample set
110. $HPD[\cdot]$ Range of values in $\pi(\theta|\boldsymbol{D})$ with total probability $1-\alpha$ (i.e., $\left(\theta^{\left(\frac{\alpha}{2}\right)}, \theta^{\left(1-\frac{\alpha}{2}\right)}\right)$) for asymmetric and symmetric distributions (a.k.a. $CI[\cdot]$ for symmetric distributions)
111. α Value used to specify total probability thresholds for HPD and CI estimation; additionally refers to the acceptance probability threshold for Metropolis updates
112. $SD[\cdot]$ Standard deviation for a probability distribution or sample set
113. $Var[\cdot]$ Variance for a probability distribution or sample set
114. $Cov[\cdot]$ Covariance between random variables
115. $Corr[\cdot]$ Coefficient of correlation between random variables
116. $ACorr[\cdot]$ Autocorrelation measure between consecutive sample values that is a function of a lag L difference measure
117. $XCorr[\cdot]$ Cross-correlation measure between consecutive sample values of different variables that is a function of a lag L difference measure
118. $MCE[\cdot]$ Monte Carlo error measure of the quality of a set of samples (i.e., variance of batch means, $\bar{\Theta}_b$, and the sample mean $\bar{\Theta}$ across a set of $\mathcal{K}$ sample batches)
119. $\mathcal{K}$ Number of batches for batch-mean estimator (MCE)
120. $\bar{\Theta}_b$ Sample mean for b^{th} batch (MCE)
121. $\theta^{(\ell)}$ ℓ^{th} sample of the parameter θ
122. B Burn-in threshold sample number
123. T Number of samples to gather
124. $\hat{\theta}$ Point estimate for the parameter θ (e.g, $E[\pi(\theta)]$ or $E[\pi(\theta|\boldsymbol{D})]$)
125. $\bar{\theta}$ Sample mean for the parameter θ; also represents current best in some MCMC algorithms
126. $\acute{\theta}$ Proposal candidates for θ
127. $\{\theta^{-}, r^{-}\}$ Leftmost state leaf in the NUTS binary map
128. $\{\theta^{+}, r^{+}\}$ Rightmost state leaf in the NUTS binary map
129. $\propto$ *"Proportional to"* operator

130. $\sim$ "*Distributed according to*" operator; in the pseudo-code examples, it instead means to sample from a distribution of that type

131. $T(a, b)$ Truncation operator that only retains samples between a and b

132. $\|\cdot\|$ Vector norm, $\|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}$ for the $p \times 1$ vector $\boldsymbol{x}$

133. $\nabla_z(\cdot)$ Gradient with respect to variable z

134. $K(\cdot)$ MCMC transition kernel function

135. $q(\cdot)$ MH proposal distribution

136. $g(x)$ Distribution that is easy to sample from (rejection sampling, squeezed rejection sampling, ARS, and ARMS)

137. $e(x)$ Envelope function (rejection sampling and squeezed rejection sampling)

138. $\exp\big(h(x)\big)$ Envelope function (ARS and ARMS)

139. $s(x)$ Squeezing function (squeezed rejection sampling)

140. $\exp\big(\hbar(x)\big)$ Squeezing function (ARS)

141. S_n Current set of ascendingly ordered abscissa $\{x_i; i = 0, \dots, n+1\}$ (ARS, ARMS)

142. y_n Set of function evaluations, $\{ln(f(x_i)); i = 0, \dots, n+1\}$, that correspond with S_n (ARS, ARMS)

143. $L_{ij}(x; S_n)$ Straight lines through adjacent points $[x_i, ln(f(x_i))]$ and $[x_j, ln(f(x_j))]$, where $x_i \leq x < x_j$ (ARS, ARMS)

144. w Estimate of the typical size of a slice (Slice sampler)

145. $\mathcal{M}$ Integer limiting the size of a slice to $\mathcal{M} \cdot w$ (Slice sampler)

146. ε Size of fictitious time steps for the leapfrog algorithm (HMC, NUTS)

147. L Number of steps for the leapfrog algorithm (HMC, NUTS)

148. $\log\mathcal{L}(\theta)$ Log-density function (HMC, NUTS)

149. λ Target simulation length ($\lambda \approx \varepsilon L$) (HMC)

150. δ Target mean acceptance probability (HMC, NUTS)

151. M^{adapt} Number of iterations after which to stop the dual-averaging adaptation (HMC, NUTS)

152. $\mathbb{I}[\cdot]$ Returns 1 if the expression in brackets is true and 0 if it is false (NUTS)

153. ϕ Temperature (power-posterior)

154. ϕ_i Temperature at the i^{th} discretization step, $\phi_i = \left(\frac{i}{n_\phi}\right)^{c_\phi}$ (power-posterior)

155. n_ϕ Discretization steps (power-posterior)

156. c_ϕ Parameter in the temperature schedule function (power-posterior)

157. $\pi_\phi(\Theta|\boldsymbol{D})$ Power-posterior density function at temperature step ϕ, $\mathcal{L}(\Theta|\boldsymbol{D})^\phi \cdot \pi(\Theta)$ (power-posterior)

158. $\Theta|\boldsymbol{D}, \phi_i$ Used in the notation for $E_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, $\mathrm{Var}_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, and $\mathrm{MCE}_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$ that are all measured using samples from $\pi_{\phi_i}(\Theta|\boldsymbol{D})$ (power-posterior)

159. s_i $\mathrm{MCE}_{\Theta|\boldsymbol{D},\phi_i}\left[E_{\Theta|\boldsymbol{D},\phi_i}\left[\log\big(\mathcal{L}(\Theta|\boldsymbol{D})\big)\right]\right]$ (power-posterior)

160. $KL(a, b)$ Kullback-Leibler distance measure between a (i.e., prior distribution) and b (i.e., posterior distribution)

Glossary of Terms

Attack surface: "The sum of all code and functionality accessible to users and potential attackers" (Howard and Lipner 2006)

Blackhats: Software security researchers having malicious intent

Cleansing: The act of removing all corresponding symbol or debug information from code

Decompile: Converting assembly or portable code language (e.g., bytecode artifacts for virtual machines) back into higher level programming languages (e.g., C or JAVA)

Disassembly: Converting a software binary from machine code form (i.e., object code, or raw bytes of the program) into human readable assembly language

Dynamic access: Ability to control and monitor execution of software either on target or on similar virtual computing devices

Dynamic analysis: Software inspections that run either on target or on similar virtual computing devices and allow for controlling and monitoring execution

Exploit: A software threat that takes advantage of an existing vulnerability to support malicious software needs

Fuzz testing: Functional boundary testing for security quality of software

Information security: "Preservation of confidentiality, integrity, availability, authenticity, accountability, non-repudiation, and reliability of information" (International Organization for Standardization 2009)

Malware: Malicious software that uses exploits and generally performs something undesirable to system or user assets (e.g., computer viruses, computer worms, Trojan horses, ransomware, and adware)

Obfuscation: The act of transforming code to inhibit reverse engineering performance

Reverse engineering (RE): External inspection of a system (often without the aid of original design information) with the goal of attaining sufficient system design comprehension (E. J. Chikofsky and J. H. Cross 1990, 13-17; M. G. Rekoff 1985, 244-252) to enable the security assessment

Security assessment: Artifact RE and security analysis activities directed to the evaluation of the security quality of a software release (i.e., the inspection of artifacts to discover vulnerabilities)

Security assessment profile (SAP): Set of variables detailing the security inspection of a particular software release (e.g., security assessment team size or skill); SAP is akin to using operational profile (OP) in the security assessment context

Security quality: "Characteristics of a product or service that bear on its ability to satisfy stated or implied" (Nelsen and Daniels 2007, 39-59) security needs for all stakeholders

Security sensitive: Indicates dependencies on one or more of the security tenets ; considered to be the attack surface for malicious users

Software release (SR): Set of variables describing a particular software version (e.g., code size or complexity)

Static analysis: Software inspections that manually review raw binary, assembly, and higher-level language artifacts when they are not running (Ball 1999, 216-234)

Threat models: A "method for uncovering design flaws in a software component before the component is built" (Howard and Lipner 2006) that uses the attacker's perspective to expose any methods that undermine the security of system assets

Vulnerability: "Instance of a mistake in the specification, development, or configuration of software such that its execution can violate the explicit or implicit security policy" (Ozment 2007, 6-11; Krsul 1998).

Vulnerability discovery modeling (VDM): Forecasts security fault discovery events over time and relies on patterns in historical discoveries over time (Alhazmi and Malaiya 2008, 14-22)

Whitehats: Software security researchers having benevolent intent (e.g., security analysts performing assessments) or third-party entities supporting threat mitigation (e.g., vendors providing security products or entities having vested interests in managing security risk)

Chapter 1. Introduction and Background

This chapter provides an introduction to the security problem of interest (i.e., vulnerability discovery post-release), details the software security lifecycle (SSL) and when discoveries occur within, and introduces strategies software-makers use for risk reduction.

1.1 The security problem

The rise of electronic crime, the proliferation of networked computing devices and their extensive customer usage, as well as the increasing interaction of device software with various forms of sensitive customer information, pose significant information security risks (underlined words are defined in the Glossary of Terms) to both consumers (e.g., financial losses incurred from malware and corresponding exploit techniques for a vulnerability) and software-makers alike (e.g., loss of revenue due to poor security quality) (van Eeten and Bauer 2008, 1-69). Because of the high cost of quality[1] and other factors (e.g., compressed-release schedules or the emergence of new security risk categories), vulnerabilities exist, and external researchers discover them post-release when performing security assessments (see Section 1.2.3.2). Public disclosures of post-release vulnerabilities increased significantly between 1996 and 2018 (The MITRE Corporation 2018a; National Institute of Standards and Technology 2018; IBM 2013; IBM 2017) (e.g., see Figure 1-1), eroding the reputation of software vendors and reducing customer confidence in security quality. Addressing all of these problems is crucial for companies that develop software and computer hardware (AMD 2018; Apple 2018; ARM 2018; Google 2018; Intel 2018; Microsoft 2018), because maintaining customer satisfaction in product security is essential to their financial success (van Eeten and Bauer 2008, 1-69).

[1] The three fundamental tradeoffs in software system development are cost, schedule, and quality. In general, quality correlates positively with cost and schedule. Equally important, cost negatively correlates with schedule (Goel and Yang 1997, 197-267).

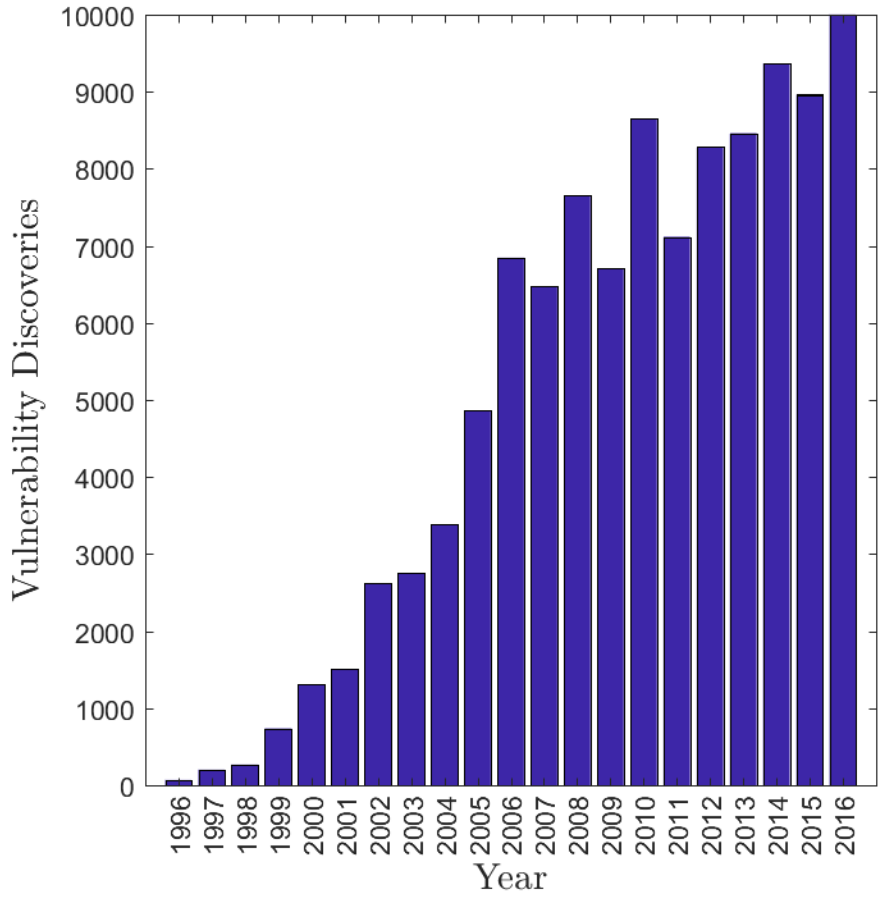


Figure 1-1: Vulnerability discoveries per year, 1996-2016

Consider the following example describing the security-relevant financial dependencies, depicted in Figure 1-2, between the i[th] vendor and j[th] customer (van Eeten and Bauer 2008, 1-69). Reduced S_i security investments from the i[th] vendor lead to increases in the j[th] customer's costs (C_{ji}) associated with product exploitation (negative correlation). These externalities (e.g., customer losses from identity theft and fraud) negatively influence vendor reputation (R_i). Of course, there are costs incurred to the vendor for investments in product security (negative correlation) and vendor standing with the customer directly influences future revenues (π_i). These relationships imply positive correlation with security investments and security quality reputation with customers. Decision-makers managing software products must therefore efficiently balance security investments throughout a product's lifecycle, as mitigation to these external customer security risks; this endeavor is certainly not trivial.

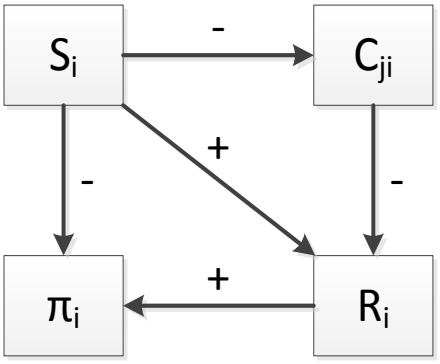


Figure 1-2: Externalities with reputation presented by van Eeten and Bauer

1.2 Software security lifecycle (SSL)

A holistic understanding of software security necessitates comprehension of the SSL, software vulnerability states in the lifecycle, and how they both relate to the software development process and product lifecycle. This description of the SSL first outlines a common model for the product lifecycle, ANSI/EIA-724, with some tailoring to encompass software security in the phase descriptions. An overview of the software vulnerability states then follows. Lastly, it provides presentation of the new model for the software security lifecycle, outlines several fundamental components of the lifecycle (software development, the trustworthy computing security development lifecycle, the process for software security assessment, and vulnerability disclosure/handling best practices).

1.2.1 Product lifecycle

ANSI/EIA-724 defines the set of phases, depicted in Figure 1-3, as the product lifecycle model (Electronic Industries Association 1997). The original application for this model was for describing the lifecycle of products in the electronics industry (Solomon, Sandborn, and Pecht 2000, 707-717). However, because modern electronics almost certainly include some form of a computer processor matched with software (e.g., smartphones, tablets, and notebook computers), with some extensions, it is relevant for this security-centric application. In chronological order, the lifecycle stages are “introduction”, “growth”, “maturity”, “saturation”, “decline”, and “phase-out”.

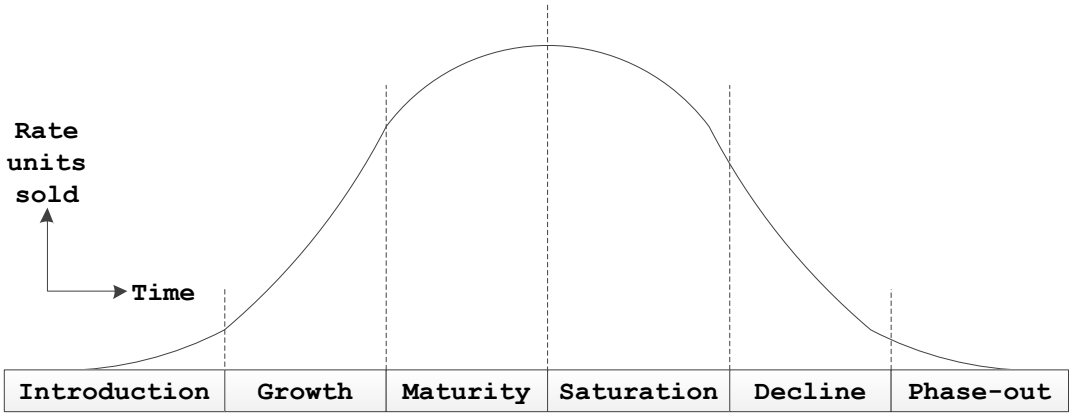


Figure 1-3: ANSI/EIA-724 Product Lifecycle Model

Table 1-1 describes by example[2] the ANSI/EIA-724 product lifecycle stages tailored to encompass security aspects relevant to this research. It assumes the software-maker follows a process similar to the trustworthy computing security development lifecycle (SDL) described in Section 1.2.3.1. For each stage, it lists notional sales, usage, feature growth, security assessment related activities performed by groups external to the software-maker, state of the security processes at the software-maker, and support levels provided by the software-maker. The table's sales and usage rows describe the changes over time in the product's customer base. The values for changes in the product's customer base should correlate positively with level changes in the frequency that external groups perform security assessment activities. The row listing feature growth in the product reflects a positive correlation with size and complexity of the released software artifacts. The row listing the software-maker's security process state describes how, over time, the implementation state of the SDL processes matures. The final row presents the status of scheduled major and as-needed security update releases.

Table 1-1: Notional summaries for stages in the ANSI-EIA-724 Product Lifecycle Model

	Introduction	**Growth**	**Maturity**	**Saturation**	**Decline**	**Phase-out**
Sales	Low	Medium→ High	High (peaks)	High (declines from peak)	High→ Medium	Low
Usage	Low	Low→ Medium	High	High	Medium	Low
Feature growth	Medium	High	Medium	Medium	Low	None

[2] In a fashion similar to how Solomon et al. (Solomon, Sandborn, and Pecht 2000, 707-717) described the lifecycle for electronic parts.

	Introduction	Growth	Maturity	Saturation	Decline	Phase-out
External security assessment activity	Low	Medium	High	Medium	Low	None
Software-maker's security process state	New	Stable	Mature	Mature	Mature	Mature
Support level	Major + security updates	Major + security updates	Major + security updates	Major + security updates	Security updates	All updates discontinued

1.2.2 Software vulnerability states

Software vulnerabilities are simply a mistake performed in the design, implementation, or configuration of software that results in information security risk. Many types of vulnerabilities are presented in the literature. Some of the notable examples include poor memory management, un-validated user input, the presence of race conditions that expose some critical asset, improper access control, inadequate initialization, and execution of software functions with unnecessary privileges (Howard 2009, 68-71).

The scope of this research does not include technical specifics of software security faults or their classification. For a discussion covering popular software security fault classifications (including taxonomy and ontology developments), interested readers may refer to Meunier (Meunier 2008, 1-18) and the Common Weakness Enumeration (CWE) initiative (The MITRE Corporation 2018b). See Erlingsson, Younan, Piessens (Erlingsson, Younan, and Piessens 2010, 633-658) and Daswani et al. (Daswani, Kern, and Kesavan 2007), for an introduction to common vulnerability examples (e.g., buffer overflows, SQL injection, and unvalidated input).

Figure 1-4 and Figure 1-5 extend the vulnerability lifecycle state model by Arbaugh et al. (Arbaugh, Fithen, and McHugh 2000, 52-59), that originally included phases for "birth", "discovery", "disclosure", "correction", "publicity", "scripting", and "death", by: renaming "birth" to "introduced"; extending "discovery" to include intent and temporal details; extending "disclosure" to include temporal details; replacing "publicity" and "scripting" with "exploited" and "malware that uses exploit"; expanding "correction" to include prerelease and post-release alternatives; and replacing "death" with "removed" or "obsolete." They illustrate the set of possible states for a vulnerability: (1) introduced; (2) discovered prerelease, benevolent; (3) vulnerability disclosed prerelease to software-maker; (4) removed

prerelease; (5) discovered post-release, benevolent; (6) vulnerability disclosed post-release to software-maker; (7) security update deployed; (8) obsolete; (9) discovered prerelease, malicious; (10) discovered post-release, malicious; (11) exploit released; and (12) malware using exploit released. "Introduction" refers to vulnerability creation and this happens within a product release cycle (i.e., prior to product release). Addressing the vulnerability takes three forms: removing prerelease; removing via security update; and having the exploitation risk go away due to obsolescence. The two elimination alternatives involve removal of the vulnerability from the software and deployment of the corresponding release to customers. The obsolete alternative involves a vulnerability that remains in a product that has been phased out and is hopefully no longer relevant.

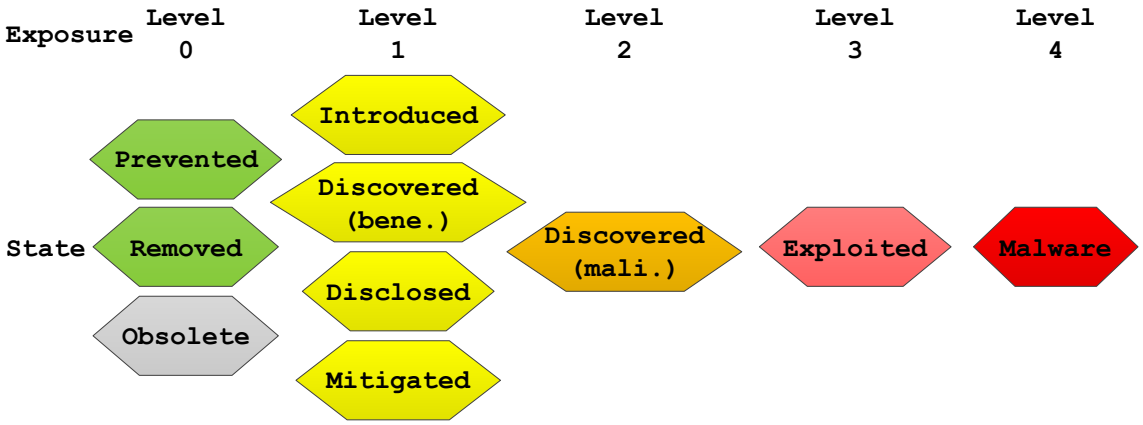


Figure 1-4: Vulnerability lifecycle states in the SSL

Vulnerability states are increasingly ranked by customer risk exposure, from left to right. The abbreviations are: "bene." for benevolent; and "mali." for malicious.

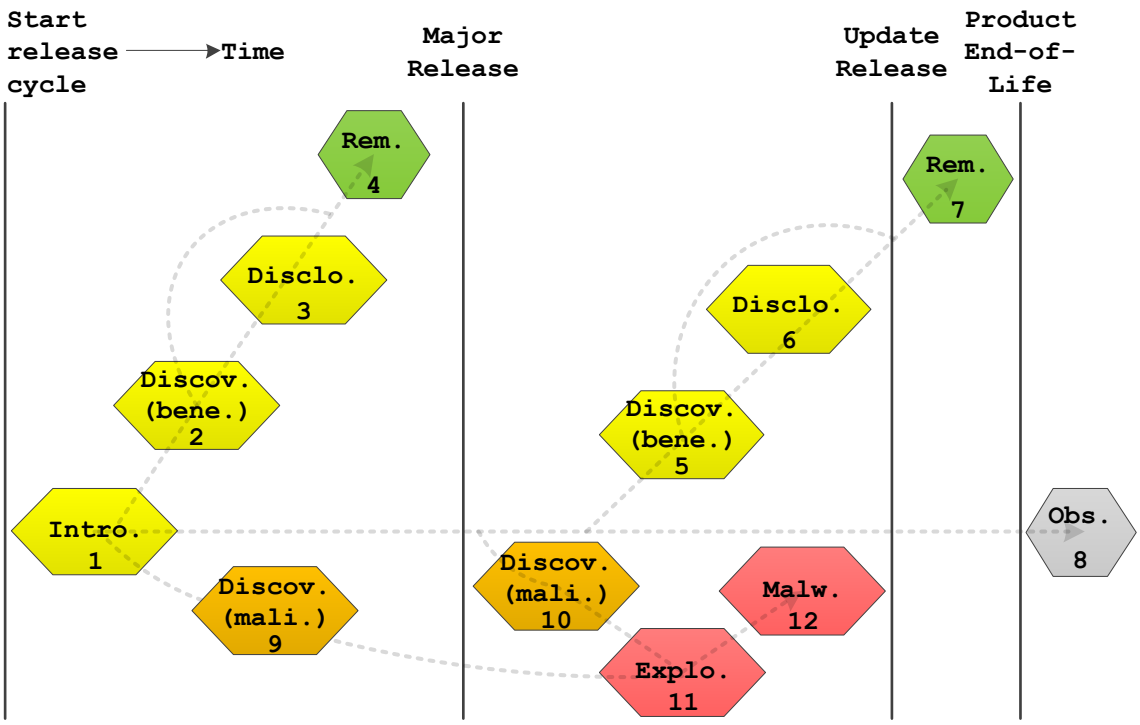


Figure 1-5: Vulnerability lifecycle state sequences in the SSL

The abbreviations are: "bene." for benevolent; "mali." for malicious; "discov." for discovered; "disclo." for disclosed; "rem." for removed; "explo." for exploited; "malw." for malware; and "obs." for obsolete.

For introduced vulnerabilities[3], the model presents seven state sequences plus several sequential combinations of them. It is important to note that security assessment activities performed by each entity type ("malicious" and "benevolent") typically happen concurrently. Consequently, dual states are possible (e.g., states "5" and "10") and the state sequences for either entity type might coexist and progress independently. The "prerelease benevolent discovery" and "elimination" state sequences (i.e., "1-2-4" and "1-2-3-4") reflect the preferred outcome for addressing introduced vulnerabilities (i.e., no customer exposure). The "post-release benevolent discovery" and "deployment of security update release" (i.e., "1-5-6-7" and "1-5-7") represent the next best result. The "introduction" to "phase-out" sequence (i.e., "1-8") describes a fortunate result that results in no customer exposure to security risk.

Sequences beginning with either "prerelease malicious discovery" or "post-release malicious discovery" are undesirable yet realistic. The former defines the infamous "zero-day" vulnerability situation and when exploited is the

[3] Obviously, preventing the creation of the fault altogether is ideal.

sequence “1-9-11-12”. The latter is the sequence “1-10-11-12”. Either of the post-release benevolent discovery sequences should follow the exploit or malware release .

Admittedly, it is possible for exploited vulnerabilities to fail benevolent discovery or lack sufficient resources to attain elimination through security update release. As these situations have been omitted from Figure 1-5, a complete diagram would reflect these changes using sequence extensions defined by connecting the states “12-8”, “5-8”, and “6-8”.

1.2.3 SSL model

It is important to first orient the security problem (i.e., vulnerability discoveries) within the context of the software security lifecycle (SSL). In this research, the SSL is presented as a circular model that: 1) fits within a product’s lifecycle (see Figure 1-6); 2) incorporates iterative and incremental development (IID) for producing secure software; 3) demonstrates IID’s relations to the problem of post-release vulnerability discovery; and 4) identifies interactions that influence customer satisfaction in security quality. Also, many sources provided inspiration in developing the SSL model (Arbaugh, Fithen, and McHugh 2000, 52-59; Lipner 2004, 2-13; Howard and Lipner 2006; Lipner 2016; Larman 2004; Electronic Industries Association 1997; Cusumano and Yoffie 1999, 60-69; Stankosky 2002; Bourque and Fairley 2014; International Organization for Standardization 2013; International Organization for Standardization 2014).

Figure 1-7 illustrates the SSL from the software-maker’s perspective and, in an overlay (cf. Figure 1-4), identifies points where the vulnerability state makes transitions[4]. The SSL starts at the top left of Figure 1-7 and proceeds clockwise, progressing chronologically with stages for product planning, major- or patch-release IID cycles (that include pre-release security assessment cycles), and post-release security assessment cycles. The chronological path then diverges into Flow1 and Flow2; from here, all ensuing traversals depend on the discovering entity types. Flow1, the outer arc, initiates when external Blackhats discover a vulnerability and includes subsequent release cycles for exploit software and malware. Flow2, the inner arc, begins when external Whitehats, or software-maker analysts, realize a vulnerability discovery post-release. For external Whitehat discoveries, Flow2 can also be diverted to associated third-party entities involved in facilitating mitigations to security threats (e.g., companies that make anti-malware software). Eventually, the SSL comes full circle when Flow1 and Flow2 paths converge at the point in which software-makers

[4] For clarity, some flow paths are omitted (e.g., the flow paths from each Blackhat release cycle rectangle to the security assessment cycle triangle).

handle vulnerabilities.

Throughout all these stages and across SSL iterations, the software-maker uses knowledge management (KM) to support continual improvements to security quality and provides customers with interim strategies for reducing risk. Moreover, the SSL explains customer perception of security quality through software-maker or third-party information, software products (i.e., new releases or updates), and external software threats.

Figure 1-6: SSL iterations within each product lifecycle model stage

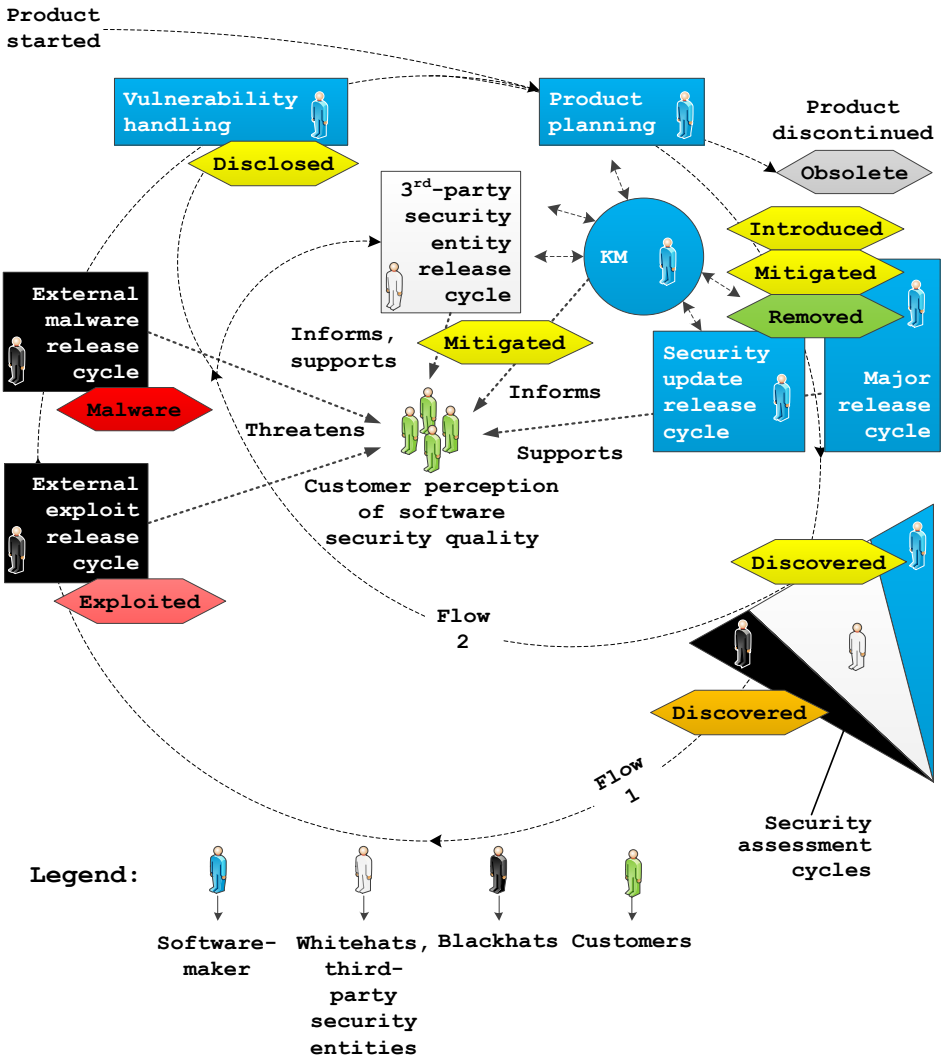


Figure 1-7: Software security lifecycle from the software-maker perspective

Table 1-2 describes SSL stage activities in further depth and indicates that vulnerability discovery occurs within the security assessment stage, which is discussed in Section 1.2.3.2.

Table 1-2: SSL stage activities from the software-maker perspective

Stage	Description of activities
Product planning	Decision-makers assign critical resources and define or refine overarching product security requirements, policy, processes, budgets, and schedule milestones. A decision to retire a product occurs at this stage, effectively rendering existing vulnerabilities obsolete.
Major/patch release cycles	Software-makers create new software using a security-focused IID model that includes steps for requirements, design, implementation, verification, and release. Occurring at this stage are fault creation (i.e., origin of the mistake), prevention (e.g., prerelease discovery with subsequent removal), removal (e.g., post-release discovery with subsequent removal), and mitigation (e.g., interim updates to reduce customer exposure).
Security assessment cycles	See Section 1.2.3.2 for details. Vulnerability discovery occurs at this stage.
Exploit/malware release cycles	External Blackhat software-makers create and release exploit software and malware; customer risk exposure escalates.
Third-party security entity release cycles	External Whitehat entities create and release security support software (e.g., anti-malware or static analysis tools for locating vulnerabilities); these external products support risk mitigations for customers.
Vulnerability handling	Security teams perform inspections to confirm discoveries, and when necessary subsequently perform root-causal analyses and risk assessments, and prepare upcoming major or security update release requirements.

1.2.3.1 Security development lifecycle (SDL)

The SDL is a series of phases that augment the software development process presented next, comprise security-focused activities, and produce additional deliverables supporting trustworthy computing goals (Howard and Lipner 2006; Lipner 2004, 2-13).

Amongst the many models for software development, this research chooses the iterative and incremental development model for demonstrating release cycles in the software security lifecycle. The iterative and incremental development (IID) model for software consists of a series of sequential, smaller release iterations, each of which builds on the previous (Larman 2004). Major release cycles for large software typically include numerous internal iterations. Conversely, update cycles for security releases usually consist of no more than one or two internal iterations.

In the software lifecycle, there are common definitions that describe maturity levels for major and security update releases. Figure 1-8 identifies the five typical versions for major releases in large software development as

"preliminary", "alpha", "beta", "candidate", and "major"[5]. Each release version could involve one or more IID sequences. Preliminary releases demonstrate a limited subset of features and are internal to the software-maker. Alpha releases are also internal but demonstrate a more complete subset of the overall feature requirements. Beta releases support most feature requirements; typically, the software-maker presents this version internally and to a small group of external users. Candidate releases support all feature requirements and software-makers present these versions internally and externally. Major releases freeze development (i.e., during this iteration software-makers only allow modifications supporting correction for critical issues) and software-makers distribute these as the final public release (Cusumano and Yoffie 1999, 60-69). In a slightly similar fashion, versions for security update releases typically have one internal candidate release prior to the final public security update release (see Figure 1-9).

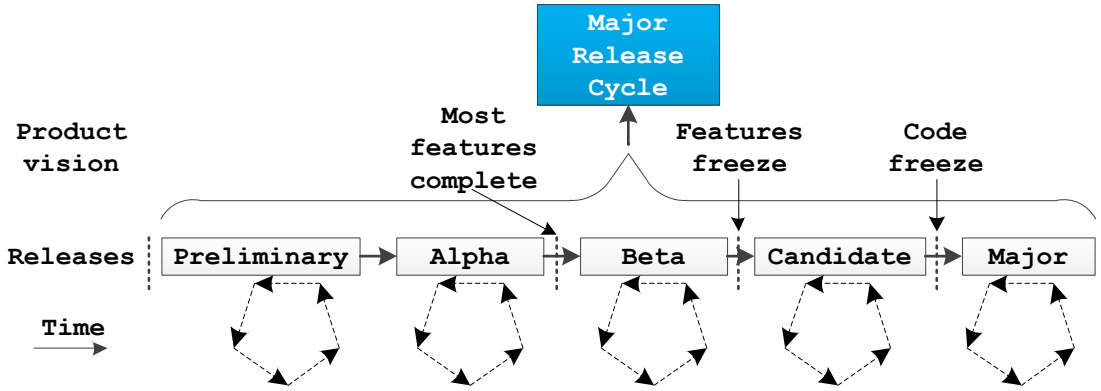


Figure 1-8: Iterative and incremental development model for a major release cycle

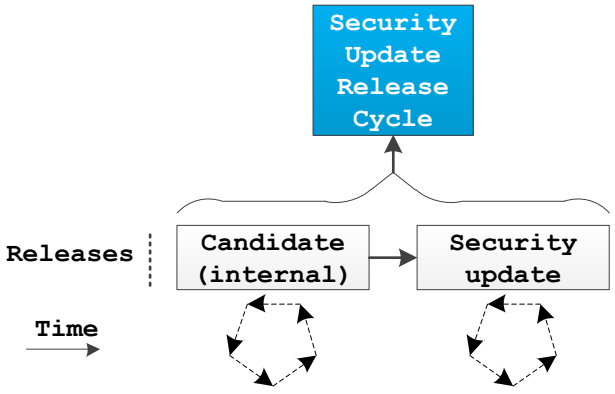


Figure 1-9: Iterative and incremental development model for an update release cycle

[5] This set of software releases is based on those provided by (Cusumano and Yoffie 1999, 60-69).

Iterations in the incremental and iterative development (IID) process for software typically execute the activities and phases listed in Figure 1-10 and in IID, the scope for each of the self-contained phases varies over time (Larman 2004). To explain, the effort proportions for each of the IID phases depend on the maturity of the corresponding iteration. For example, earlier iterations typically emphasize the requirements and design phases, while later ones have an emphasis on implementation and verification. Table 1-3 describes IID preparation and continuous support activities, and Table 1-4 provides brief descriptions for the IID phases. Additionally, it incorporates the trustworthy computing security development lifecycle (SDL) into this IID process. Last, it is worthwhile to point out the step similarities with the traditional waterfall development model (Royce 1970, 1-9) and the strong connections between a product (i.e., the system) and software.

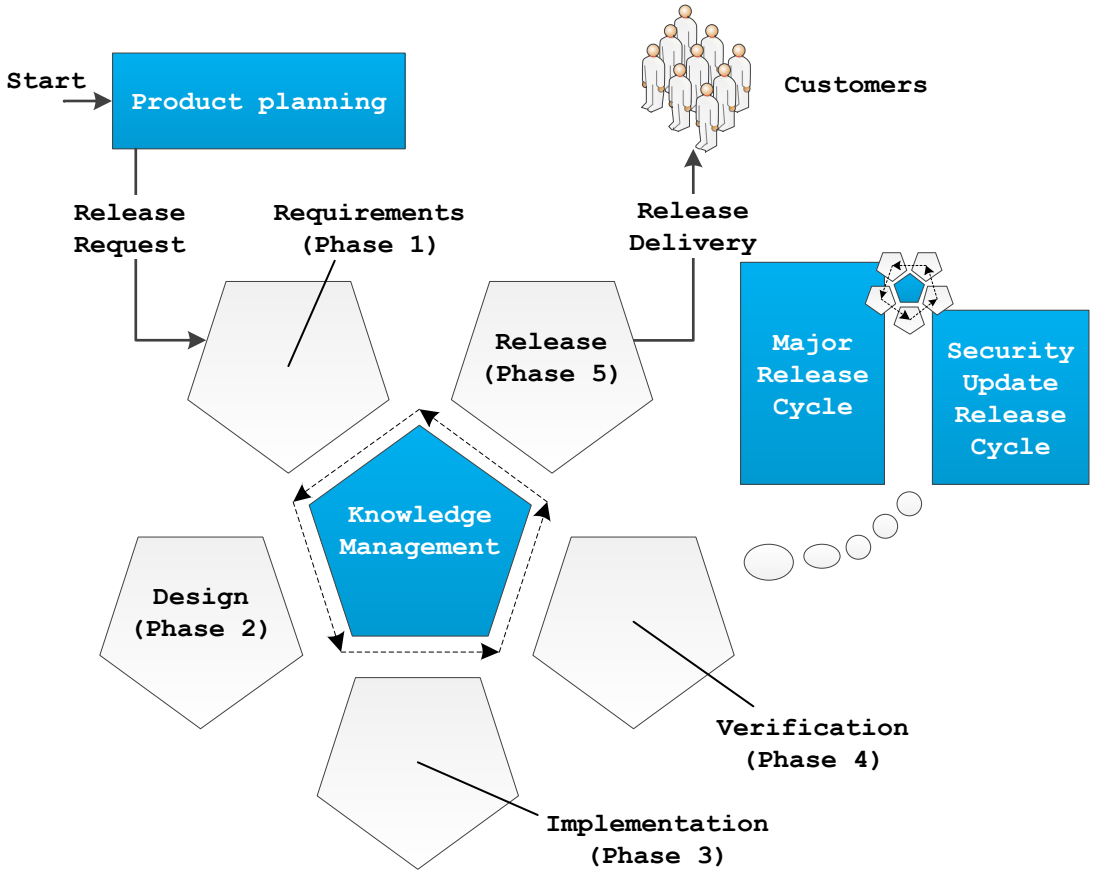


Figure 1-10: Iterative and incremental software development process

Table 1-3: IID activity descriptions

Activity name (timing)	Description
Product planning (preparation)	Product planning activities set the stage for the cycle. Decision-makers at the software-maker determine product needs/goals/objectives, define some of the critical product and process requirements, allocate resources to initiate the first few phases, and identify major schedule milestones for the initial lifecycle phases.
Knowledge management (continuous)	Knowledge management (KM) "leverages relevant intellectual assets to improve organizational performance" (Stankosky 2002). KM is a multi-disciplinary approach to retaining and growing organizational knowledge that supports all phases and iterations in the IID process. Among other things, the KM solution within an organization provides technology to archive and locate specific information or personnel, training, lessons learned from previous products, and technical information from present and past software release cycles.

Table 1-4: IID phase descriptions

Phase number: Name	Description
Phase one: Requirements	Activities that include "elicitation, analysis, specification, validation, and management" (Bourque and Fairley 2014) of software requirements (e.g., necessary software features, functions, capabilities, performance, interfaces, constraints, and so on)
Phase two: Design	Activities supporting translation of the requirements into specifications that describe the software structure and provide a foundation for constructing the product (Bourque and Fairley 2014)
Phase three: Implementation	"Coding, verification, unit testing, integration testing, and debugging" related activities supporting software creation (Bourque and Fairley 2014)
Phase four: Verification	Activities supporting evaluation of the quality of a software release (e.g., static and dynamic inspection) and confirmation that the product satisfies all requirements (Bourque and Fairley 2014)
Phase five: Release	Activities supporting generation and distribution of the various staged versions of a product (i.e., preliminary, alpha, beta, release candidate, and major release)

1.2.3.2 Vulnerability discoveries in the SSL

The problem of information security (i.e., vulnerability discovery external to software-makers) is grounded within the context of the SSL and its security assessment cycles. The security assessment cycles within the SSL implement the agile process for vulnerability discovery[6]. To describe these cycles, this research presents an iterative

[6] Software engineering processes support controlling complexity, managing scope, meeting schedule and cost constraints, and improving team performance. Agile processes enable some of the same concepts but support dealing with uncertainty and provide maneuverability (e.g., managers can monitor status and readjust goals at each iteration or sprint) (Larman 2004).

software security assessment process (ISSAP), which is a scientific method for reviewing software security quality that applies to *all* entities performing security assessments. The ISSAP, shown in Figure 1-11, starts with assessment planning (at the upper-left corner) and proceeds clockwise. As part of planning, analysts set overall security assessment goals and exit conditions, construct prioritized task backlogs, and assign key personnel to the top priority tasks. Then, analysts gather and prepare software release artifacts for assessment, perform artifact reverse engineering (RE), formulate vulnerability hypotheses and relevant functional tests, implement and verify tests on artifacts, and assess security quality of artifacts. Benevolent researchers document and report results for their final step; alternately, malicious researchers develop and release corresponding exploits and malware that use the vulnerabilities discovered in their assessment cycle. Additionally, knowledge-management (KM) activities enhance assessment performance throughout ISSAP iterations. As time and resources allow, analysts perform subsequent ISSAP cycles, each time reviewing the results from the previous cycle and then continuing with another iteration. Table 1-5 provides further details about ISSAP steps.

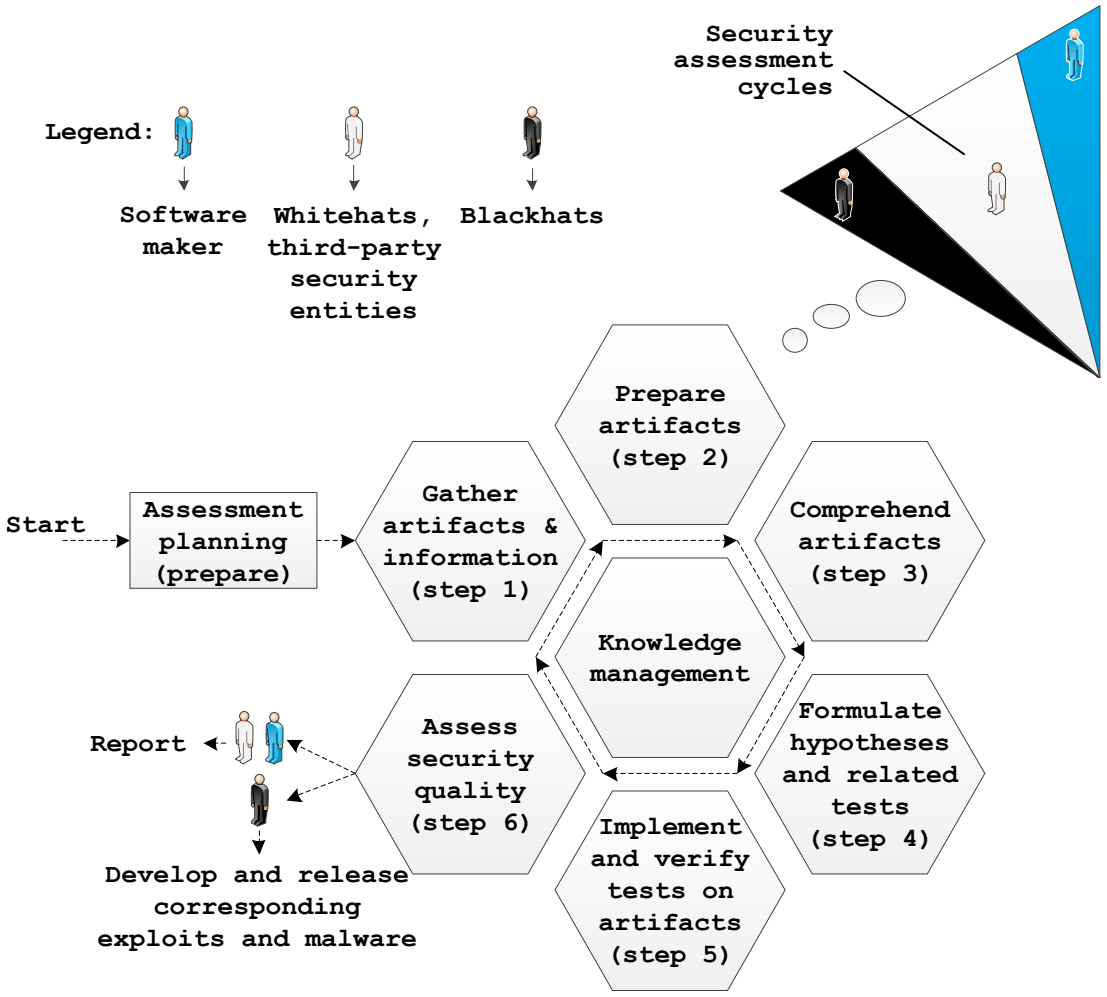


Figure 1-11: Iterative software security assessment process (ISSAP)

Table 1-5: ISSAP steps

Step	Description of activities
Gather artifacts and information	Analysts gather relevant software artifacts and information from the software release (e.g., design information, source code, binary artifacts, etc.).
Prepare artifacts	Analysts perform preparation activities, including inspection and reformatting of binary artifacts, binary disassembly into assembly language, and assembly-language decompilation into higher-level programming languages.
Comprehend artifacts	Analysts work towards attaining artifact comprehension, which involves software RE and includes performing static analysis and dynamic analysis on available software artifacts.
Formulate hypotheses and related tests	Analysts formulate or use pre-existing threat models to identify potential attack surfaces and then use these threat models to formulate hypotheses for potential vulnerabilities that could have higher impact; analysts also consider tests or analysis procedures for evaluating these hypotheses.

Step	Description of activities
Implement and verify tests on artifacts	Analysts test functional boundaries (a.k.a., fuzz testing) or vulnerability hypotheses on the target product or similar devices providing target emulation.
Assess security quality	Analysts confirm suspected vulnerability locations in the artifacts. Analysts then prepare the security assessment document and report results.

1.2.3.3 Vulnerability disclosure and handling

Two recent standards provide processes for vulnerability disclosure to software-makers (ISO/IEC 29147) and vulnerability handling by software-makers (ISO/IEC 30111). Figure 1-12 outlines these two processes and their relationships with each other. The first standard, ISO/IEC 29147, outlines recommendations for external entities performing security assessments to disclose vulnerabilities they discover to the software-maker and additionally sets expectations for the software-maker's response (International Organization for Standardization 2014). The second standard, ISO/IEC 30111, outlines recommendations for software-makers in responding to post-release vulnerability discoveries (International Organization for Standardization 2013).

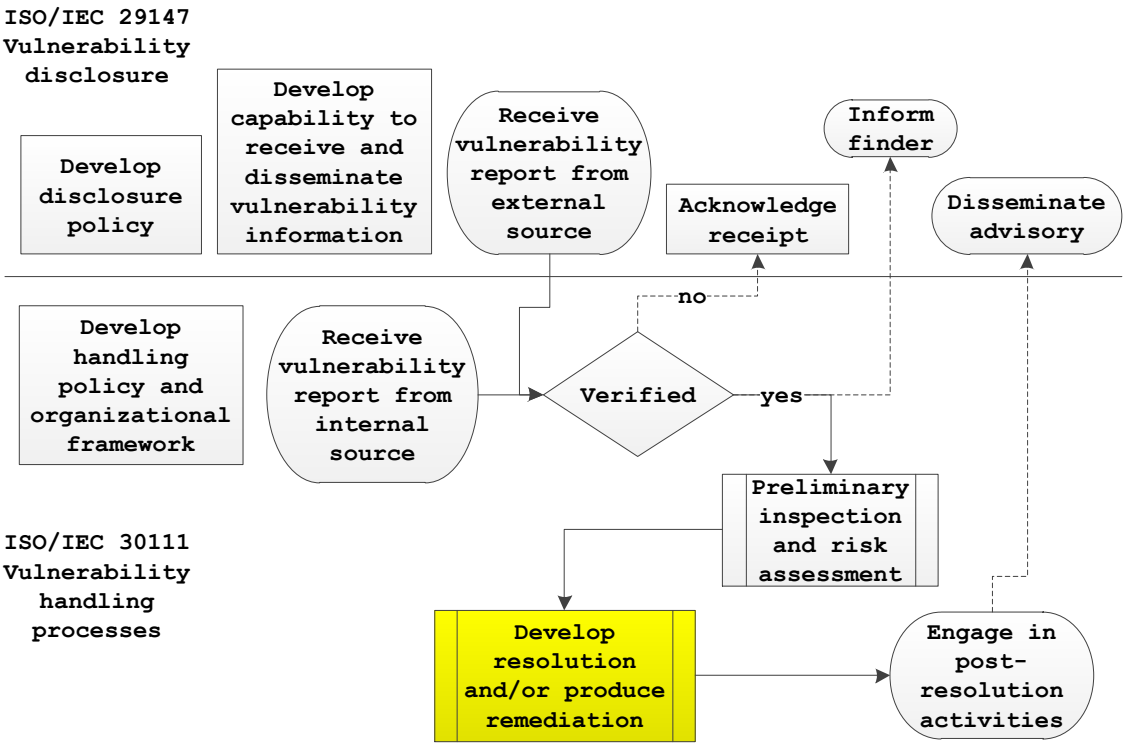


Figure 1-12: ISO/IEC 29147, ISO/IEC 30111 processes

Respectively, these are the vulnerability disclosure and vulnerability handling processes.

The ISO/IEC 30111 related sub-process for vulnerability resolution and remediation (see Figure 1-13) presented by this research stems from industry best practices (Lipner 2016). To minimize resolution delivery time for security updates addressing those vulnerabilities designated as critical, inspection and preparation of security patch requirements begins straightaway and the output subsequently supports an immediate out-of-band security patch update. For all vulnerabilities, there are inspections that result in identification of systemic problems. For those identified as systemic, the software-maker must make a decision for design and development of an automated discovery tool. All vulnerability discoveries resulting from manual and automated tool identification have detailed inspections and security patch requirements prepared. For vulnerabilities classified with negligible user impact, deferment of security update releases to the next scheduled release is common industry practice.

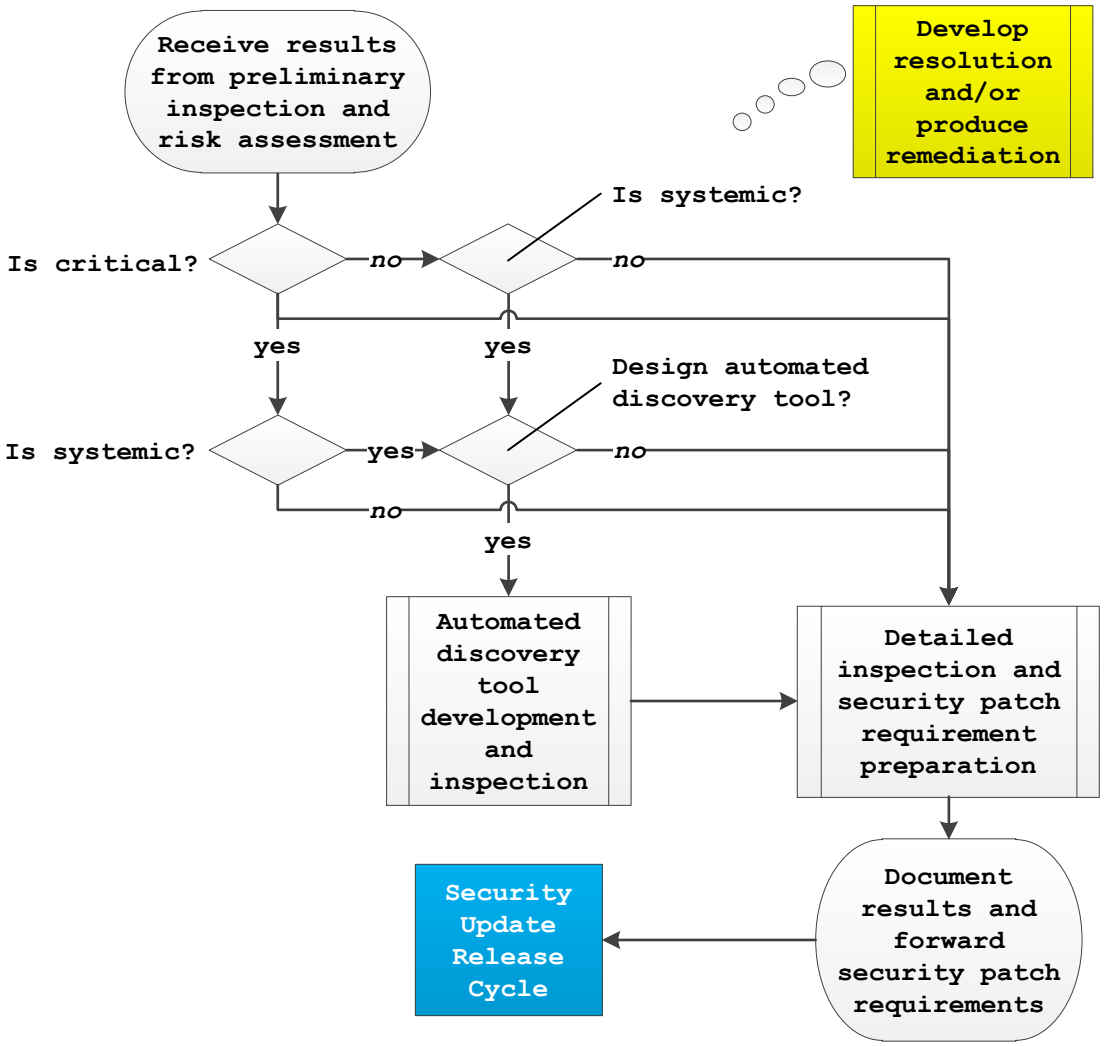


Figure 1-13: Vulnerability resolution and remediation process

1.3 Strategies to reduce risk

Fortunately, strategies do exist to reduce risk and ensure customer satisfaction in security quality throughout the software security lifecycle (SSL). Software-makers can refine processes and policies, reallocate critical resources, and alter release-cycle requirements or constraints (e.g., feature requirements or schedule and budget limitations). These adjustments have two foci of application for reducing post-release discovery risk and minimizing its impact: Area1 and Area2 (see Figure 1-14). Activities in Area1 (red) aim to improve security quality prerelease (Williams, Gegick, and

Vouk 2008), while Area2 (green) activities seek to control customer perception of security quality post-release[7] (Petter, DeLone, and McLean 2013, 7-62). Area2 is divided into two sections: Area2-A, in which the goal is to reduce customers' threat of exposure (e.g., by decreasing service-response times for discoveries or including fault tolerance by design methods); and Area2-B, in which the goal is to inhibit the discovery of vulnerabilities by entities with malicious intent (Collberg and Thomborson 2002, 735-746). Unfortunately, due to the high costs of achieving quality, managers must often decide which alternatives will provide maximal impact—a decision that is aided by the security modeling techniques discussed next.

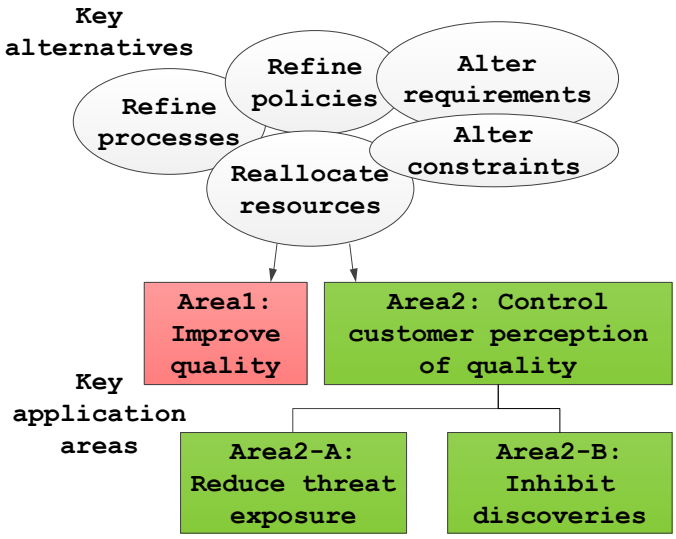


Figure 1-14: Mitigation alternatives and application areas

[7] The customer perception of security quality can be influenced by software-maker information-sharing (e.g., public transparency or prompt security risk notification) and service quality (e.g., security patch response time) (Petter, DeLone, and McLean 2013, 7-62), as well as through design information security (Parker 1997, 572-582) and the inclusion of techniques inhibiting public discovery of vulnerabilities (Collberg and Thomborson 2002, 735-746).

Chapter 2. Literature Review

This chapter presents the origins for security modeling, formally introduces vulnerability discovery modeling (VDM) techniques, and outlines areas for their improvement.

2.1 Security modeling origins

Security quality modeling requires historical information and this warrants a brief synopsis of the public vulnerability database origins.

Archives from the early 1980s through the turn of the century contain the digital footprints for the initial security assessment movement. Notable sources of information were the *alt.security* and *comp.security.unix* USENET newsgroups (Bishop 1995), the *2600*[8] and *Phrack*[9] nonstandard publications, the ACM RISKS digest mailing list (Neumann 1985, 1), and the *BugTraq* mailing list (SecurityFocus 2006). Amongst these and other sources, an increase occurred in publications that detailed common vulnerabilities and their corresponding exploitation techniques. Notable examples included: "Smashing the Stack for Fun and Profit" (i.e., stack overflows) (aleph1 1996); "Return into libc" (i.e., return-oriented programming) (solar designer 1997); "NT Web Technology Vulnerabilities"(i.e., SQL injection) (rain.forest.puppy 1998); "Once Upon a Free" (i.e., heap overflows) (anonymous 2001); "Exploiting Format String Vulnerabilities" (scut/team teso 2001); and "Basic Integer Overflows" (blexim 2002).

In the 1990s, the growth in vulnerability and exploit specific knowledge and the increasing frequency in security failures significantly raised public awareness of the existence of security faults and of the phenomena supporting their discovery. In 1999, creation of the common vulnerabilities and exposures (CVE) database answered the multiple calls in the literature for a public historical vulnerability database (Baker et al. 1999; Aslam, Krsul, and Spafford 1996).

The combination of discovery event information from vulnerability databases, software characteristic data from code repositories, and recorded engineering process execution metrics all provide valuable data for exploratory studies. For this problem phenomenon, research areas of interest focused on factors possibly influencing the discovery of software

[8] First appearing in 1984 (Goldstein 2009).

[9] First appearing in 1993 (SecurityFocus 2006).

vulnerabilities. Scholars applied software quality modeling techniques using information from these sources for better understanding of the discovery phenomenon.

2.2 Vulnerability discovery modeling (VDM)

One important software security modeling technique, vulnerability discovery modeling (VDM), helps managers make decisions on how best to reduce risk[10]. Simply put, VDM techniques forecast security fault discoveries over time (Alhazmi and Malaiya 2008, 14-22).

Vulnerability discovery models are an application of software reliability models that uses patterns found in historical discovery events following a software release (SR) to make predictions of discovery-event counts over time. They enable managers to allocate resources for subsequent release cycles that help ensure post-release vulnerability handling quality and response times remain satisfactory (see Area2 in Figure 1-14). VDM methods, demonstrating varying levels of success, include: linear and polynomial regression models (Alhazmi, Malaiya, and Ray 2007, 219-228; Alhazmi and Malaiya 2008, 14-22; F. Massacci and V. H. Nguyen 2014, 1147-1162; Ruohonen, Hyrynsalmi, and Leppänen 2015, 1-20; Rescorla 2005, 14-19); growth-curve models (Alhazmi, Malaiya, and Ray 2007, 219-228; Alhazmi and Malaiya 2008, 14-22; Woo et al. 2011, 50-62; Joh and Malaiya 2014, 1445-1459; F. Massacci and V. H. Nguyen 2014, 1147-1162; Ruohonen, Hyrynsalmi, and Leppänen 2015, 1-20); models based on the nonhomogeneous Poisson process (NHPP, see Appendix B.2) (Rescorla 2005, 14-19; Alhazmi and Malaiya 2008, 14-22; Okamura, Tokuzane, and Dohi 2013, 15-23; F. Massacci and V. H. Nguyen 2014, 1147-1162; V Nagaraju, L Fiondella, and T Wandji 2017, 31-50; Rahimi and Zargham 2013, 395-407); effort-based models (Kimura 2003, 279-287; Kimura 2006, 256-261; Alhazmi and Malaiya 2005, 615-620; Woo et al. 2011, 50-62; Ozment 2006, 25-36); time-series models (Roumani, Nwankpa, and Roumani 2015, 32-40); and various specialty models (Anderson 2002; Rahimi and Zargham 2013, 395-407). Table 2-1 lists select VDMs demonstrated in the literature and, where applicable, their originating software reliability counterparts.

Table 2-1: VDM technique cross reference

VDM	VDM Source	Software Reliability Model Source
Thermodynamic entropy model	(Anderson 2002)	(Brady, Anderson, and Ball 1999)
NHPP, Software vulnerability assessment model	(Kimura 2006, 256-261)	(Yamada and Fujiwari 2001, 205-218)

[10] Other modeling methods (e.g., release schedule modeling (Cavusoglu, Cavusoglu, and Raghunathan 2007, 171-185) and vulnerability prediction modeling (Williams, Gegick, and Vouk 2008)) exist, but are not useful here.

VDM	VDM Source	Software Reliability Model Source
Quadratic (Second order polynomial) model	(Rescorla 2005, 14-19)	(Shooman 1976, 268-280)
NHPP, exponential model	(Rescorla 2005, 14-19)	(Goel and Okumoto 1979, 206-211)
Logistic model	(Alhazmi and Malaiya 2008, 14-22)	(Yamada and Osaki 1985, 1431-1437)
Gompertz model	(Ruohonen, Hyrynsalmi, and Leppänen 2015, 1-20)	(Yamada and Osaki 1985, 1431-1437)
Effort-based model	(Alhazmi and Malaiya 2005, 615-620)	(J. D. Musa 1975, 312-327)
Littlewood-Verrall Bayesian model	(Ozment 2006, 25-36)	(Littlewood and Verrall 1973, 77)
NHPP, Musa-Okumoto logarithmic Poisson model	(Ozment 2006, 25-36)	(J. Musa and Okumoto 1984, 230-238)
NHPP, Moranda Geometric Poisson model	(Ozment 2006, 25-36)	(Moranda 1975, 327-332)
Weibull model	(Joh, Kim, and Malaiya 2008, 299-300)	(Schick and Wolverton 1978, 104-120)
NHPP, generalized Gamma model (includes exponential NHPP and logarithmic extreme-value at min, aka Weibull)	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Hiroyuki, Mitsuaki, and Tadashi 2007, 81-90; Goel 1985, 1411-1423; Goel and Okumoto 1979, 206-211)
NHPP, Pareto model	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Littlewood 1984, 157-159)
NHPP, truncated normal model	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Okamura, Dohi, and Osaki 2013, 135-141)
NHPP, log-normal model	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Achcar, Dey, and Niverthi 1998)
NHPP, *truncated* logistic model	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Okamura, Dohi, and Osaki 2004, 14-22; Ohba 1984, 144-162)
NHPP, log-logistic model	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Gokhale and Trivedi 1998, 34-41)
NHPP, truncated extreme-value min/max (Gompertz)	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Ohishi, Okamura, and Dohi 2009, 535-543; Yamada 1992, 964-969)
NHPP, logarithmic extreme-value at max (Frechet)	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Hirata, Okamura, and Dohi 2009, 225-236)
NHPP, hyper-Erlang	(Okamura, Tokuzane, and Dohi 2013, 15-23)	(Okamura and Dohi 2008, 232-239)
Scrying method	(Rahimi and Zargham 2013, 395-407)	NA

2.3 Improving VDM techniques

To date, acceptance of discovery models is very limited because several related and significant challenges remain to be addressed. First, no general pattern in web-browser discoveries exists (see NVD data issues discussion in Chapter 6); as well, there is the potential that one could make predictions prior to a general change in discovery trends (Woo et al. 2011, 50-62). Clearly, factors not accounted for would explain the lack of patterns in the phenomenon over time (e.g., differences in skill and numbers of discoverers, return on investment for discovery of vulnerabilities, levels of design information security, amount of code-reuse, etc.). Second, results from one application are not comparable to others (Ozment 2007, 6-11), as across applications there will be many differences in SR and SAP variables and these each influence the phenomenon differently. Thus, the modeling results from a web-browser are not comparable to those from an operating system or even those from different web-browsers. Third, the models do not provide a means to adjust SR and SAP variables, for exploring their influence on discovery. Fourth, as pointed out by some (Ozment 2007, 6-11; F. Massacci and V. H. Nguyen 2014, 1147-1162), many applications of discovery models violate static code assumptions (e.g., see (Alhazmi and Malaiya 2005, 615-620; Alhazmi, Malaiya, and Ray 2007, 219-228; Alhazmi and Malaiya 2008, 14-22; Woo et al. 2011, 50-62; Joh and Malaiya 2014, 1445-1459; V Nagaraju, L Fiondella, and T Wandji 2017, 31-50)).

Because software-makers can refine processes and policies, reallocate critical resources, and alter release-cycle requirements or constraints (e.g., feature requirements or schedule and budget limitations), it would be very useful to model the influence from variables in these areas to support making related decisions. Having a "clear-box" discovery model that incorporates using a well-defined set of variables for describing SR and SAP combinations would support decisions for strategies to reduce risk and ensure customer satisfaction.

Unfortunately, only four VDM techniques include any SR and SAP input variables; the remaining techniques treat the problem as an uncontrollable "black-box." Rahimi and Zargham (2013, 395-407) proposed the Scrying method, which is the sole VDM alternative using SR variables; however, this method requires access to source code and incorporates, out of numerous factors, only code-complexity and compliance rules. An application of the mean-value function (MVF) from Musa's basic execution time model (Alhazmi and Malaiya 2005, 615-620; J. D. Musa 1975, 312-327), plus altered forms of the Musa-Okumoto logarithmic NHPP (Ozment 2006, 25-36; J. Musa and Okumoto 1984, 230-238) and Yamada-Fujiwara testing-domain models (Kimura 2003, 279-287; Kimura 2006, 256-261; Yamada and Fujiwari 2001, 205-218) are the only VDM alternatives using SAP variables. Unfortunately, their quality suffers from

SAP data limitations that correspond with vulnerability discoveries; as well, some violate the static code assumption for software reliability modeling (Ozment 2007, 6-11; F. Massacci and V. H. Nguyen 2014, 1147-1162) and they all incorporate only one SAP variable (i.e., assessment effort) via proxy measures[11].

Naturally, the dearth of both SR and SAP information and corresponding data for historical vulnerability discoveries has inhibited the introduction of multivariate VDM techniques. Despite this, one can still make use of them through the Bayesian approach to analysis[12] (Samaniego 2010), as it supports multiple types of data – including expert opinion information. What's more, forecasting performance of VDM methods can also be improved by using Bayesian model averaging[13] (Madigan and Raftery 1994, 1535-1546; Sarishvili and Hanselmann 2013, 1-8) and this is also enabled through an approach that includes expert judgment methods.

[11] E.g., Alhazmi and Malaiya define this proxy as $E = \sum(U_i \cdot P_i)$, where U_i is the total number of users over time and P_i is the percentage of users operating the software over time (Woo et al. 2011, 50-62).

[12] Bayesian methods are more suitable when the data are scarce as they enable the use of numerous and diverse information types—including expert judgment (Samaniego 2010).

[13] Averaging over all the models results in better average predictive ability when compared to any single best model (Madigan and Raftery 1994, 1535-1546; Sarishvili and Hanselmann 2013, 1-8).

Chapter 3. Research conceptual framework

This chapter presents the overarching conceptual framework for this research in seven parts: Section 3.1 lists the research questions; Section 3.2 identifies the research hypotheses; Section 3.3 introduces the purpose and usefulness of the research; Section 3.4 notes the fundamental assumptions for the research; Section 3.5 discusses several items pertaining to research scope and perspective; Section 3.6 provides the conceptual model; and Section 3.7 presents the research variables.

3.1 Questions

Investigation of this problem revealed four key research questions that associate with the following two areas: (1) "How do discoveries behave over time?", and (2) "What VDM techniques were ideal?" The first research question, identified using RQ1, falls under question area (1) and was, "How does the baseline SR and SAP combination influence post-release vulnerability discovery over time?" The second one, identified using RQ2, also falls under question area (1) and was, "How do the covariates influence post-release vulnerability discovery over time?" The third, numbered RQ3, associates with question area (2) and was, "What is the best 'black-box' VDM technique?" The fourth and final research question, numbered RQ4, also associates with question area (2) and was, "What is the best 'clear-box' VDM technique?"

3.2 Hypotheses

Deliberation on the four research questions induced four corresponding hypotheses. The first hypothesis, identified using RH1, associates with RQ1 and is, "baseline SR and SAP combinations have increasing then decreasing discovery rates over time." The second is a set that associates with RQ2 and is identified using RH2.1-10. These are listed below in Table 3-1 and each associates with one of the 10 covariates that are explored in depth (see Section 4.3.3 for introduction of covariates associated with these hypotheses and Section 4.4.2 for the details concerning how these covariates are used in the multivariate models). The third is identified using RH3, associates with RQ3 and is, "Bayesian model averaging is the best 'black-box' VDM technique." The fourth and last hypothesis is identified using RH4, associates with RQ4 and is, "Linearly-scaled BMA is the best 'clear-box' VDM technique."

Table 3-1: Subset of hypotheses associated with RQ2

ID	Hypothesis
RH2.1	Software size is *negatively* correlated with vulnerability discoveries over time.

ID	Hypothesis
RH2.2	Product price is *negatively* correlated with vulnerability discoveries over time.
RH2.3	Assessment tool quality is *positively* correlated with vulnerability discoveries over time.
RH2.4	Assessment personnel effort is *positively* correlated with vulnerability discoveries over time.
RH2.5	Assessment personnel quality is *positively* correlated with vulnerability discoveries over time.
RH2.6	Level of dynamic access to software is *positively* correlated with vulnerability discoveries over time.
RH2.7	Amount of reused software is *positively* correlated with vulnerability discoveries over time.
RH2.8	Amount of available design information is *positively* correlated with vulnerability discoveries over time.
RH2.9	Amount of obfuscated software is *negatively* correlated with vulnerability discoveries over time.
RH2.10	Amount of cleansed software is *negatively* correlated with vulnerability discoveries over time.

3.3 Purpose and usefulness

The purpose of this research is fivefold and includes: 1) presentation of the Kuo-Ghosh NHPP as a VDM technique; 2) presentation of BMA as a VDM technique; 3) presentation of a multivariate, time-dependent discovery model for software with its corresponding variable set; 4) demonstration of approaches for gathering multivariate data and for performing subsequent analysis of the multivariate model using this data; and 5) evaluation of the hypotheses listed in the previous section.

This research is useful for the following reasons: 1) it can incorporate expert-judgment recommendations that bolster baseline analysis results (see NVD data issues discussion in Chapter 6); 2) it enables comparison of results from differing applications; 3) it supports trade studies for controllable variables (e.g., see forecasting examples in Section 5.4.2); 4) it does not violate modeling assumptions for a static code base; and 5) its Bayesian approach alleviates the data sparseness issue, provides informative prior distributions, and naturally characterizes uncertainty in the model parameters.

3.4 Assumptions

This research depends on several key assumptions that facilitate its methods (see Table 3-2). First, the research accounts for factors outside of the experts' perspectives[14] by making assumptions about various levels of release quality

[14] Post-release security fault discovery depends on prerelease fault creation and removal.

(Assumptions AS1-2). Accordingly, this research investigates only the variables influencing post-release discovery[15], thereby limiting the applicability of its results to Area2 (see Figure 1-14). The following assumptions (i.e., AS1*, AS3-5) come from a category of traditional software-reliability models based on the non-homogeneous Poisson process (NHPP, see Appendix B.2). Assumptions AS1* (i.e., the alternate to AS1) and AS3-5, taken from Yamada et al. (Yamada, Ohba, and Osaki 1983, 475-484), are modified for modeling the discovery of software vulnerabilities in the baseline SR and SAP. Assumptions AS6-7 are essential to modeling arbitrary SR and SAP combinations using the multivariate methods introduced by this research. Assumption AS8 is historically used in software reliability modeling[16]. The final two assumptions relate to data elicitation. Assumption AS9 eliminates an area of uncertainty in SR and SAP used to gain information about vulnerability discovery, and Assumption AS10 is necessary for the expert-judgment elicitation method.

Table 3-2: Key assumptions

ID	Description
AS1	A fixed number (i.e., $\gamma = 148, 55, 20, 7, 3$) of vulnerabilities is released for the respective elicitation scenarios
AS1*	The initial number of vulnerabilities within the software release is random (Yamada, Ohba, and Osaki 1983, 475-484)
AS2	The types of vulnerabilities within the software release are random
AS3	A software release undergoing security assessment is subject to discovery of vulnerabilities at random times caused by security faults present in the release (Yamada, Ohba, and Osaki 1983, 475-484)
AS4	The time between discoveries (k-1) and k depends on the time to discovery (k-1) (Yamada, Ohba, and Osaki 1983, 475-484)
AS5	Each time a discovery occurs, the vulnerability discovered is immediately removed, and no other vulnerabilities are introduced (Yamada, Ohba, and Osaki 1983, 475-484)
AS6	SR and SAP covariates are static throughout the security assessment of a particular software release (Cox 1972, 55-66)
AS7	Vulnerability discoveries over time, for arbitrary SR and SAP combinations, are realized by modulating (i.e., scaling) discoveries over time from the baseline SR and SAP
AS8	The elicited data, cumulative discoveries over time, are realizations of independent, identically distributed random variables
AS9	The software binaries (described by SR variables, see Section 3.7) are available unencrypted

[15] In a situation in which the number of existing faults is unknown, Assumption AS1* would be substituted for Assumption AS1.

[16] There has been some debate over Assumption AS8 in the literature (e.g., there is the special case for the emergence of new vulnerability types, which typically results in numerous related discoveries following closely thereafter) (Ozment 2007, 6-11).

ID	Description
AS10	The expert-provided distributions (see Section 4.3.2.1) are independent (i.e., Cooke classical model independence)

3.5 Scope and perspective

The research scope and perspective as well as external factors affect this study. First, as structured elicitation requires working directly with experts, practical considerations (i.e., for the amount of work involved) limit the research. Accordingly, this research investigates factors that influence the discovery of vulnerabilities by *academic* security researchers performing security assessments[17]. Figure 3-1 presents the typical roles; the red areas indicate those groups considered outside the scope for this research. Second, this research does not include vulnerability type into the models because it would make the already complex elicitation process less manageable (see Assumption AS2). Finally, two confounding factors require special attention regarding methodology: return-on-investment (ROI) variables that can significantly influence certain SAP factors[18] and software-maker decisions that can affect the quality of the artifacts released[19]. In terms of controlling ROI variables, this research uses elicitation scenario construction (introduced in Sections 4.3.2.1 and 4.3.3) while Assumption AS1 (see Table 3-2) addresses the issue of confounding factors for release-quality[20].

[17] In defense, it is reasonable to speculate that the influence from most of the proposed non-ROI SR and SAP factors on vulnerability discovery should be generalizable outside the population of experts participating in this study.

[18] For example, the levels and quality of resources applied to security assessment should correlate positively with the return on investment.

[19] For example, skill of the developers or the resources applied to prerelease security assessment should influence release quality.

[20] For example, release details concerning participating software-makers (e.g., developer and tester skills, security quality assurance processes, and design specification complexity), would not be explicitly known by most external security analysts.

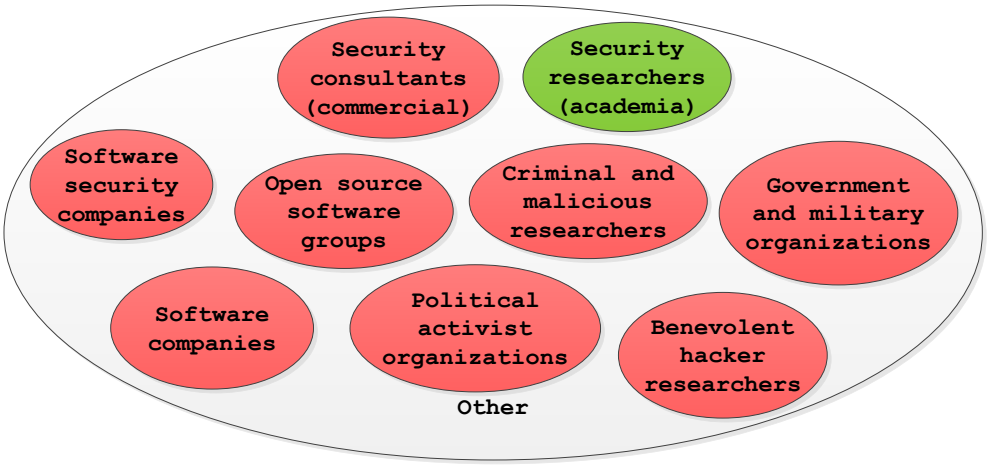


Figure 3-1: Security researcher roles (expert population)

3.6 Conceptual model

The conceptual model describes research variables within factor groups (by listing them adjacent to each group) and illustrates their interactions (see Figure 3-2). The groups include discovery return on investment (G1), the software-maker business (G2), the software release cycle (G3), software release (G4), security assessment profile (G5), security reputation (G6), exploit or malware software release (G7), and security reputation propagation (G8). In Figure 3-2, the green areas indicate the factor groups and those activities that fall within the scope of this research. Notably, this model places vulnerability discovery within the security reputation factor group and contains factor groups for SR and SAP variables, illustrating these variables' direct influence on security reputation.

Figure 3-2: Proposed conceptual model and relevant factors

3.7 Variables

Most security assessment tasks for external security analysts involve RE, which is analogous to performing product development backwards. Application of the definition for a project to endeavors with vulnerability discovery-related activities is a natural progression because persons routinely performing RE and security assessments can estimate the effort involved in a well-defined assignment. As with a project for a development effort, security assessment tasks with a security assessment profile (SAP) have variables (e.g., those defining resource quality and availability) that influence schedule, cost, and technical performance. Likewise, SR characteristics (e.g., software size, availability of design information, or anti-tampering features) would also influence the rate of vulnerability discoveries over time.

Numerous variables influence vulnerability discovery, and choosing the appropriate set of variables enables the description of *any* SR and SAP combination from an external perspective. Thus, for example, this research specifies an elicitation scenario as having a set of variable values that defines a SR and SAP for the post-release security assessment of a notional software system similar to a web-browser. When experts have an explicitly defined SR and SAP combination included in the elicitation scenario, they can better estimate how the variables influence vulnerability discovery over time. These variables, introduced briefly via hypothetical examples in Table 3-3, are listed by type (i.e., age, cost, complexity, SR information security, size/scope, security mechanisms, process, ROI, resource availability, and resource quality) and provided with the corresponding reasons that support their selection.

Table 3-3: Research variable overview

Type	**Hypothetical examples supporting selection—with reasoning**	**References**
Age, Size/scope	Levels of code-reuse between release cycles or the number of new features in a release—both influence the amount of RE necessary	---
Cost	Limited budget for personnel, tools or test units—each influences performance	---
Information security	Availability of source code, design information, or binary artifacts—all influence the amount of RE necessary	(Eilam 2011; Collberg and Thomborson 2002, 735-746; Samuelson and Scotchmer 2002, 1575-1663)
	Binary artifact cleansing and obfuscation—each influences RE performance	
Process	Use of an established process—influences performance	(Gibson, Goldenson, and Kost 2006)

Type	Hypothetical examples supporting selection—with reasoning	References
Security	Dynamic access (a.k.a. debug) levels or the presence of anti-tampering features on products running the software—each influences performance	(M. N. Gagnon, S. Taylor, and A. K. Ghosh 2007, 82-84; Collberg and Thomborson 2002, 735-746)
Return on investment (ROI)	Potential annual unit sales or customer exposure—each influences resources applied	(Bambauer and Day 2010, 1051-1107; van Eeten and Bauer 2008, 1-69; Anderson 2001, 358-365)
	Processor architecture market share—influences resources applied	
	Monetary compensation for vulnerability disclosures—influences resources applied	
	Monetary compensation from the exploitation of users—influences resources applied	
	Criminal penalties—influence resources applied	
Complexity	System or software complexity—influences performance	(E. J. Chikofsky and J. H. Cross 1990, 13-17)
Coding Language	Type of development language—influences RE performance	(Eilam 2011)
Size/scope	Software size (as measured from disassembled or decompiled binaries, or original source code)—influences performance	(Howard and Lipner 2006; Anderson 2001, 358-365)
	Proportion of software size that is security sensitive—influences performance	
Resource availability	Availability of analysts—influences performance	(Ozment 2007, 6-11)
	Levels of shared security assessment CPU resources available—influences performance	
	Availability of equipment capable of natively executing the software—influences performance	
Resource quality	Analyst experience or skill level—influences performance	(Eilam 2011; Ozment 2007, 6-11; Weiser 1982, 446-452)
	Analysis tool quality—influences performance	

To enable the description of any SR and SAP combination, the numerous variables influencing vulnerability discovery are now described from a perspective that is external to software-makers. First, their notation defines SR and SAP combination $\boldsymbol{x}_0 = (x_{0_1} \quad x_{0_2} \quad \cdots \quad x_{0_n})^T$ and $\boldsymbol{x}_i = (x_{i_1} \quad x_{i_2} \quad \cdots \quad x_{i_n})^T$, that each respectively represent covariates in baseline and arbitrary scenarios describing the post-release security assessment of notional software systems. For each variable x_j in $\boldsymbol{x}$, where $j = 1,2,\cdots,n$, Table 3-4 includes pertinent details such as identifier, description, type, baseline value(s), and its corresponding metric unit. Most of these variables are reasonably self-explanatory and, in the interest of brevity, elaboration is provided using footnotes only when necessary. Generally, the

baseline values were chosen to either ensure non-zero discoveries over time or to control the effects from confounding variables.

Table 3-4: Select variables that are pertinent to security assessment

ID	Description	Type	Baseline value(s)	Metric unit
x_1	% (percent) of functions reused from previous release	Age	25	%
x_2	Software product lifecycle phase (see Figure 1-6)	Age	*Growth*	Categorical
x_3	Product unit retail price	Cost	10,000	US $
x_4	Software unit retail price	Cost	5,000	US $
x_{5-8}	% of functions in structural complexity levels 1-4, [21]	Complexity	10, 20, 40, 30	%
x_{9-12}	% of functions in information flow complexity levels 1-4, [22]	Complexity	10, 20, 40, 30	%
x_{13}	Central processing unit (CPU) complexity level 1-5, [23]	Complexity	2	1-5 rating[24]
x_{14}	Software system virtualization complexity rating 1-4, [25]	Complexity	3	1-4 rating
x_{15}	Software system concurrency complexity rating 1-5, [26]	Complexity	2	1-5 rating
x_{16}	% of functions with source code available	Information security	10	%
x_{17}	% of functions that are cleansed	Information security	25	%
x_{18}	Publicly available % of design information[27]	Information security	25	%

[21] Function structural complexity as defined by the following: 1-"untestable," cyclomatic complexity (CC) score >50; 2-"complex," CC score 21-50; 3-"more complex," CC score 11-20; 4-"simple program," CC score 1-10 (McCabe 1976, 308-320; Software Engineering Institute 1997)

[22] Function information flow (IF) complexity as defined by the following: 1-IF4 score, $> 10^6$; 2-IF4 score, $10^5 - 10^6$; 3-IF4 score, $10^3 - 10^5$; 4-IF4 score $10^1 - 10^3$ (Henry and Kafura 1981, 510-518; Shepperd 1990, 3-10)

[23] CPU complexity as defined by the following: 1-greater than 64-bit multi-core; 2-64-bit multi-core; 3-32-bit single or multi-core; 4-16-bit; 5-8-bit

[24] All variables were constructed with ratings such that the highest rating would result in the most discoveries. This is a methodology constraint necessary for the second phase of the research (Szwed et al. 2006, 157-177).

[25] Software virtualization complexity as defined by the following: 1-multiple guest OS's in a host OS; 2-guest OS in a host OS; 3-host OS; 4-no OS;

[26] Software concurrency complexity as defined by the following: 1-(>150) processes; 2-(50-150) processes; 3-(11-50) processes; 4-(2-10) processes; Level 5-no concurrency (i.e., 1 process)

[27] An open source product defines 100%; practitioners must estimate everything else relative to this condition.

ID	Description	Type	Baseline value(s)	Metric unit
x_{19}	Publicly available % of design related binary artifacts[28]	Information security	25	%
x_{20}	% of functions that are obfuscated	Information security	25	%
x_{21}	% of functions developed in assembly language	Development language	5	%
x_{22}	Security assessment process[29]	Process	*Incomplete*	Capability level rating
x_{23}	Security assessment personnel effort[30]	Resource availability	10	FTEs[31] (40 hour / week per FTE)
x_{24}	Available shared security assessment CPU resources (separate from individual team member PCs)	Resource availability	100	Total CPU cores (available 24*7 hour / week)
x_{25}	Total product budget available for the security assessment	Resource availability	30,000	US $
x_{26}	Available equipment capable of natively executing software	Resource availability	11, [32]	Total units
x_{27}	Average assessment team experience with product technology	Resource quality	12	Months
x_{28}	Security assessment tool quality[33]	Resource quality	3	1-5 rating

[28] For example, software-maker binaries (i.e., used in production or maintenance) that are not included with the product release

[29] The following security assessment process area capability levels are defined using the capability maturity model integrated (CMMI) for development (continuous representation): 1-"incomplete," no process or partial implementation of an assessment process; 2-"performed," accomplishes security assessments and assessment goals are satisfied; 3-"managed," security assessments are planned and executed in accordance with general policy; 4-"defined," general policy for security assessments is tailored, using guidelines, specific for each assessment project (Software Engineering Institute 1997).

[30] In constructing the elicitation scenarios, the baseline level for this covariate was intentionally elevated to ensure discoveries would occur (modulation is only possible when expected baseline discoveries over time are greater than zero).

[31] Full-time equivalent number of personnel

[32] For the baseline, the personal computers for the 10 FTEs (x_{23}) are capable of natively running the software and there is additionally one product unit available.

[33] Tool quality is defined by the available RE tool value, with respect to average disassembly and decompile task effectiveness (% correct), completeness (% complete), and completion speed (completion time). The effectiveness and completion areas use the following ranges: 1-"very poor" (0-20%); 2-"poor" (20-40%); 3-"good" (40-60%); 4-"very good" (60-80%); and 5-"ideal" (80-100%). Completion time uses the following scale: 1-"very slow" (hours); 2-"slow" (tens of minutes); 3-"normal" (several minutes); 4-"fast" (tens of seconds); and 5-"very fast" (several seconds or less).

ID	Description	Type	Baseline value(s)	Metric unit
x_{29}	Security assessment average personnel quality[34]	Resource quality	3	1-5 rating
x_{30}	Potential annual unit sales	ROI	1,000,000	Units sold per year
x_{31}	Processor architecture market share	ROI	96	%
x_{32}	Potential software-maker rewards	ROI	20,000	US $
x_{33}	Potential black-market rewards	ROI	0	US $
x_{34}	Potential malware profits per install	ROI	0	US $
x_{35}	Potential criminal penalties	ROI	0	Prison years
x_{36}	Potential legal cost liabilities	ROI	0	US $
x_{37}	Average number of users interfacing with a product instance each year	ROI	1,000	Users per unit per year
x_{38}	Average number of user security sensitive transactions each year	ROI	52	User transactions per year
x_{39}	Level of dynamic access to artifacts[35]	Security	4	1-5 rating
x_{40}	Design anti-tamper security rating 1-5, [36]	Security	4	1-5 rating
x_{41}	Total number of functions	Size, scope	1,000	Functions
x_{42}	% of functions that are security sensitive	Size, scope	40	%
x_{43-46}	% of functions in size range 1-4, [37]	Size, scope	10, 20, 40, 30	%

Cumulative vulnerability discoveries, $N(t)$ or $N(\tau)$, will be interval-grouped by time, t, and assessment time, τ. The former is defined by 10-calendar week time intervals, $[t_{i-1}, t_i)$ for $i = 1,2, \dots, \mathcal{I} = 5$ (i.e., [0,10), [10, 20), [20, 30), [30, 40), [40, 50) weeks). The latter is defined by equivalent security assessment time, τ, $[\tau_{i-1}, \tau_i)$ for $i =$

[34] Security assessment average personnel quality is described as the team average personnel skill level (rated per the Dreyfus scale), which are: 1-"Novice," 2-"Advanced beginner," 3-"Competent," 4-"Proficient," and 5-"Expert" (Dreyfus 2004, 177-181).

[35] The level of dynamic access to artifacts is defined as the level of software control when executing on target (or target-similar) hardware, and the levels are categorized as follows: 1-none; 2-partial debug control on target-similar device; 3-limited privileges (i.e., user mode) on target, software debugger available; 4-full privileges (i.e., administrative user mode) on target, software debugger available; and 5-full privileges, external hardware control for debugging on target.

[36] Design security effects in response to tampering detection are rated as follows: 1-failed, un-repairable (i.e., it's a "brick," or useful only as a paperweight); 2-impaired, un-repairable; 3-failed, repairable with significant associated costs/delay; 4-impaired, repairable with some associated costs/delay; and 5-negligible effect or repairable with negligible associated costs/delay.

[37] Rated per the following function size ranges in assembly language lines of code (ALOC): 1->1000, with median=1250 ALOC; 2-250-1000, with median=750 ALOC; 3-50-250, with median=200 ALOC; and 4-1-50, with median=40 ALOC.

$1,2,\ldots,J=5$ (i.e., $x_{23}(FTE)\cdot 40\left(\frac{\frac{SA\ hours}{week}}{FTE}\right)\cdot t(week)$, or [0,4000), [4000, 8000), [8000, 12000), [12000, 16000), [16000, 20000) SA hours, where x_{23} is defined later in this Sub-section).

Chapter 4. Methodology

This chapter first introduces key concepts for the methodology and then provides the details in four subsequent parts. The approach fundamentals for eliciting expert judgment and performing Bayesian analysis are provided in Section 4.2, data-gathering details are presented in Section 4.3, Bayesian analyses of VDM techniques are outlined in Section 4.4, and their implementations within MCMCBayes are introduced in Section 4.5.

4.1 Introduction

Historically, the counts of post-release vulnerability discoveries have varied over time, and previous studies have used "black-box" techniques to describe this phenomenon for a single SR and SAP (i.e., one software product and its associated post-release environment). However, vulnerability discovery for differing SR and SAP combinations (i.e., all the possible types of software and their post-release surroundings) depends on many factors. As well, the application of time-dependent, "clear-box" multivariate models enables prediction for arbitrary SR and SAP combinations of interest – to those using VDM techniques. The selected class of multivariate models are now introduced.

These models perform scaling perturbations on a *time-dependent*, baseline function (see Figure 4-1, upper-left side) using parametric modulation (see the same, lower-left side). Model justification is supported by augmenting a performance-weighted average model for discoveries in a single SR and SAP with the inclusion of a new capability that parametrically scales a reference output – which stems from modeling the baseline SR and SAP – per an input set of variables describing the arbitrary SR and SAP of interest.

The general equation for modeling the mean-value function (MVF) in the i^{th} SR and SAP combination, $\boldsymbol{x}_i$, that represents the corresponding average discoveries over time in $\boldsymbol{x}_i$ (i.e., $E[N(t, \boldsymbol{x}_i)]$), is

$$f_i(t, \boldsymbol{x}_i | \boldsymbol{x}_0) = f_0(t) \cdot s(\boldsymbol{x}_i). \tag{4.1}$$

The two terms in the equation are $f_0(t)$, which represent: 1) the baseline function for assumed non-zero average discoveries over time in $\boldsymbol{x}_0$ (i.e., $E[N(t|\boldsymbol{x}_0)]$; and 2) a modulation term, $s(\boldsymbol{x}_i)$. Equation (4.1) provides an estimate for

$E[N(t, \boldsymbol{x}_i)]$ in $\boldsymbol{x}_i$ by perturbing the baseline function, $f_0(t)$, with the $s(\boldsymbol{x}_i)$ scaling function[38]. Notably, $s(\boldsymbol{x}_i)$ uses the $\boldsymbol{x}_i$ vector of variables to describe the SR and SAP combination of interest and scales $f_0(t)$ so that it becomes $f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)$.

I also note that using Equation (4.1) to generate the ratio of discoveries between SR and SAP combinations $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$, while constraining $s(\boldsymbol{x}_j) > 0$ in the ratio set construction, simplifies to

$$\frac{f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)}{f_j(t, \boldsymbol{x}_j|\boldsymbol{x}_0)} = \frac{s(\boldsymbol{x}_i)}{s(\boldsymbol{x}_j)}, \tag{4.2}$$

(Soyer and Tarimcilar 2008, 266-278). It is important to realize that in these ratios, the baseline terms cancel.

Consequently, there is no dependence in Equation (4.2) on $\boldsymbol{x}_0$ or the time t, and this facilitates a two-phased methodology approach. Phase I elicits the baseline SR and SAP dataset, $\boldsymbol{D}_0$ (introduced in Section 4.3.2), and performs "black-box" modeling of the phenomenon in a baseline SR and SAP (i.e., it models $f_0(t)$). Phase II uses results from Phase I, elicits the multivariate dataset, $\boldsymbol{D}^1/\boldsymbol{D}^2$ (introduced in Section 4.3.3), and supports "clear-box" modeling of the phenomenon for arbitrary SR and SAP combinations (i.e., it models $f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)$, defined by Equation (4.1)).

4.2 Approach fundamentals

When predicting *rare* events, it is appropriate to choose the subjectivist, or evidential, view of probability (Samaniego 2010) and to perform data analysis using the Bayesian approach.

At the highest level, this two-phased methodology approach performs a general sequence of elicitation and analysis steps for each phase. These include elicitation preparation (E1), elicitation execution (E2), expert calibration and elicited data-cleansing (E3), elicited data aggregation (E4), empirical data-gathering (DG), and Bayesian analysis (AN1-10) using all the data. The center of Figure 4-1 depicts this methodology sequence for both phases; its right side details the analysis steps.

[38] This abstraction loosely generalizes the modulation notion from (Cox 1972, 55-66; Soyer and Tarimcilar 2008, 266-278) to all the baseline functions.

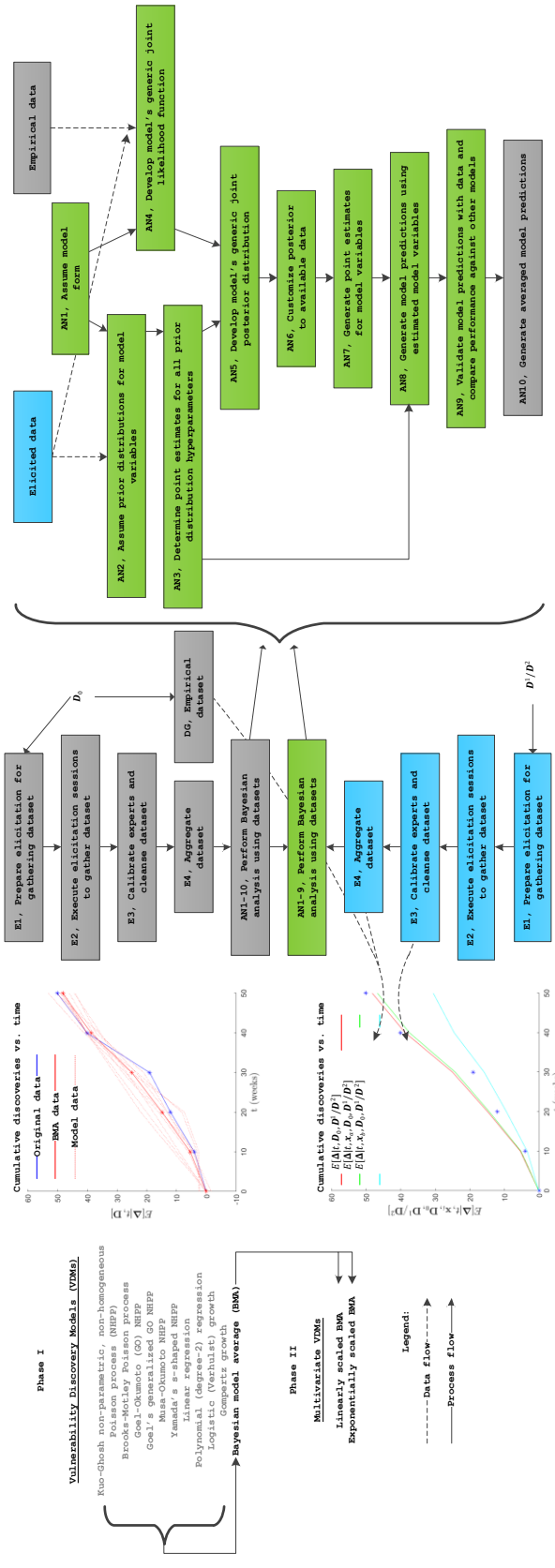


Figure 4-1: Methodology Overview

4.2.1 Cooke's method

Because the data for discoveries over time are scarce, structured expert judgment was elicited, based on the Cooke classical model (CCM) approach (Cooke 1991), to provide a dataset that enables estimating an informative Bayesian prior (Phase I) and using the multivariate model (Phase II). Cooke's method provides rational consensus in the exploration of problems involving decision processes with "substantial scientific uncertainty" (Cooke and Goossens, L L H J 2008, 657-674). The structured, expert-judgment elicitation process associated with Cooke's method (Cooke 1991) for gathering data comprises elicitation preparation (E1), elicitation execution (E2), expert calibration and elicited data-cleansing (E3), and elicited data aggregation (E4) (see Figure 4-2).

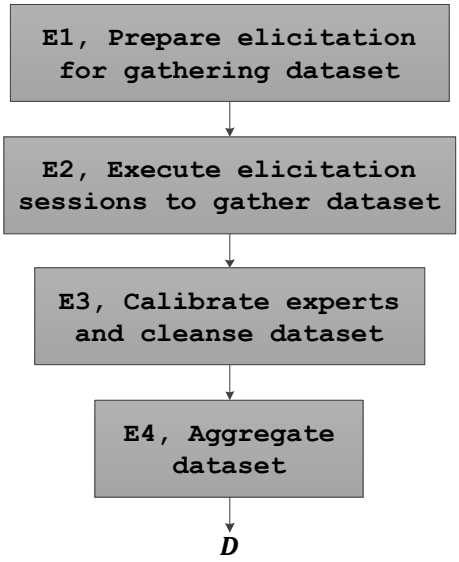


Figure 4-2: Expert judgment steps E1-E4

E1, elicitation preparation, consists of the detailed design, development, and verification of the elicitation methodology, including necessary support modules (e.g., scenario details, elicitation questionnaires, assessment exercises, etc.). A pilot elicitation session is typically conducted to provide an early evaluation of this elicitation material, which may result in some of the materials requiring subsequent refinement.

E2, elicitation execution, is the expert-judgment workshop that uses E1 materials to gather relevant, expert knowledge in a structured manner. For every workshop activity, experts receive instructions and subsequently perform elicitation tasks independently. In executing these tasks, they provide information with regard to data of interest, data-cleansing, and data aggregation. For Cooke's method, all data is elicited in a point-estimate format that specifies a

$\{q_{5\%}, q_{50\%}, q_{95\%}\}$ set of quantiles that contribute to defining a distribution (i.e., percentiles that divide the probability space into four bins, see Figure C-1 in Appendix C). The corresponding distribution extremes, $q_{0\%}$ and $q_{100\%}$, derive from the complete expert data-set (i.e., $\{q_{5\%}, q_{50\%}, q_{95\%}\}$ from all the experts) (Cooke 1991). Experts specify point-estimates for the specified phenomenon (e.g., expected vulnerability discoveries in a time interval), for each of $\{q_{5\%}, q_{50\%}, q_{95\%}\}$. To enable E3, experts also answer a series of seed-item (a.k.a., calibration) questions by providing another set of corresponding point-estimates. Under CCM, these seed questions, each having known answers (Ψ_i, for the questions $i = 1$ to $\mathcal{S}$) in quantitative form, support expert-score calibration (i.e., generation of calibration scores) and elicited data cleansing (Cooke 1991).

E3, expert calibration and elicited data-cleansing, weighs experts' opinions and cleanses their data based on how well each specifies the distribution point-estimates for the calibration-questions posed to them in E2 (Cooke 1991). The CCM sequence assesses individual, expert-estimation performance against the $\Psi_1, \Psi_2, \cdots, \Psi_{\mathcal{S}}$ answers and generates a normalized aggregation weight for each expert, thereby ensuring a data combination that emphasizes the results from the experts who performed well in the calibration exercises. The computation of weights results from two scoring measures and a quality-level cutoff. The first of these, the calibration score $C(e)$ (see Appendix C), for the expert e, measures how well the expert specifies answers to the seed questions. The second measure, an information score, $I(e)$ (see Appendix C), assesses the same expert's certainty in the point-estimates for the phenomenon of interest and is derived by averaging all of the expert's individual, seed-item, information scores, $I(e, i)$ (see Appendix C) (Cooke 1991). The quality-level cutoff, A, is determined numerically (Cooke 2008, 775-777) and defines a factor multiplied against each weight (i.e., $1_{\mathrm{A}}(\mathrm{x}) = 0$ if $\mathrm{x} < \mathrm{A}$ and $1_{\mathrm{A}}(\mathrm{x}) = 1$ otherwise) that enables the removal of all data from those experts whose information does not meet minimum-quality thresholds (Cooke and Goossens, L L H J 2008, 657-674). The global weight for the expert, e, is then

$$w_{\mathrm{A}}(e) = 1_{\mathrm{A}} \cdot C(e) \cdot I(e), \tag{4.3}$$

(Cooke 1991). For performance comparison, two additional types of aggregation weights typically accompany $w_{\mathrm{A}}(e)$. Although the item weight for the expert, e, is like $w_{\mathrm{A}}(e)$, it rests instead on the individual, seed-item scores or on

$$w_{\mathrm{A}}(e, i) = 1_{\mathrm{A}} \cdot C(e) \cdot I(e, i), \tag{4.4}$$

(Cooke 1991). Simple averaging generates the equal weight for the expert, e (out of E experts), or

$$w_{eq}(e) = \frac{1}{E}, \tag{4.5}$$

(Cooke and Goossens, L L H J 2008, 657-674).

In E4, the final step—elicited data aggregation, the data-aggregation weights are computed and then used to combine the cleansed data of interest into one aggregate probability-distribution for each of the elicited quantities of interest. Because the aggregate data then exists in probability distribution form, a final simulation step is taken to transform the data into point estimates[39] for the elicited quantities of interest[40].

4.2.2 Bayesian analysis

Bayesian analysis for a model generally performs the following nine steps (see Figure 4-3). In AN1, "Assume model form," the stochastic model form for $f(\cdot)$, $Pr(\Delta|\boldsymbol{D})$, that includes some function, $F(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the set of model variables, is identified as the candidate for the analysis. In AN2, "Assume prior distributions for model-variables," initial knowledge of the phenomenon is used to specify the joint, prior distribution for the variables in the model, $\pi(\boldsymbol{\theta})$, where $\pi(\cdot)$ generally denotes $Pr(\cdot)$ for functions of $\boldsymbol{\theta}$. In AN3, "Determine point estimates for all prior distribution hyper-parameters," point estimates are derived for hyper-parameters in the prior distributions, using *a priori* knowledge about the phenomenon (e.g., $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ elicited from expert opinion that is a subset of $\boldsymbol{D}$). In AN4, "Develop model's generic joint likelihood function," the likelihood function is developed, $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) = Pr(\boldsymbol{D}|\boldsymbol{\theta})$, which is the Bayesian notational equivalent (changing from a function of $\boldsymbol{D}$ to one of $\boldsymbol{\theta}$) for the probability of the data given the model variables. In AN5, "Develop model's generic joint posterior distribution," the likelihood function is combined with the prior distribution to construct the generic, joint, posterior distribution using Bayes' theorem

[39] The MATLAB "PiecewiseLinear" distribution supports simulation of values from these expert distributions. To use this function, the distribution point estimates require slight manipulation (i.e., adding of minute values) to ensure non-zero values for each of the specified quantiles. Also, see Appendix A.5.

[40] I chose this approach to facilitate easier model construction because some choices for prior distributions make model construction more difficult. For example, if one were to use the linear regression model, the typical choice for a prior distribution would be MVN-Gamma—which yields the conjugate posterior that is also MVN-Gamma. This symmetry obviously breaks when choosing alternate prior distributions with this model, such as the piecewise linear distribution that is generated directly from the CCM approach. Furthermore, the piecewise linear distribution isn't directly supported by the MCMC samplers used.

$$\pi(\boldsymbol{\theta}|\boldsymbol{D}) = \frac{\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) \cdot \pi(\boldsymbol{\theta})}{\int \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) \cdot \pi(\boldsymbol{\theta}) \cdot d\boldsymbol{\theta}} \tag{4.6}$$

$$\propto \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) \cdot \pi(\boldsymbol{\theta}), \tag{4.7}$$

(Robert 2010)[41]. In AN6, "Customize posterior to available data," the data (here, the prior subset of the data, $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$, is excluded from $\boldsymbol{D}$ as it is represented by $\pi(\boldsymbol{\theta})$) is used to tailor (i.e., perform term expansion) the generic posterior. AN7, "Generate point estimates for model variables," point estimates for $\boldsymbol{\theta}$, denoted $\widehat{\boldsymbol{\theta}}$, are realized either through analytical derivations or by generalizing samples from $\pi(*)$ (e.g., using either the prior-based mean, $\widehat{\boldsymbol{\theta}} = E[\pi(\boldsymbol{\theta})]$, or the posterior-based mean, $\widehat{\boldsymbol{\theta}} = E[\pi(\boldsymbol{\theta}|\boldsymbol{D})]$). In AN8, "Generate model predictions using estimated model variables," results are achieved by inserting $\widehat{\boldsymbol{\theta}}$ into the stochastic model and then generating prior- or posterior-based predictions, or samples, from the stochastic model. Finally, in AN9, "Validate model predictions with data and compare performance with other models," prediction evaluation is undertaken to identify criteria, assess model prediction performance and validate model output using the criteria. Furthermore, once additional information is available, sequential analysis becomes possible because the posterior results provide the prior distribution for a subsequent round of modeling using the newly arrived data.

[41] Where $\propto$ means "*proportional to.*"

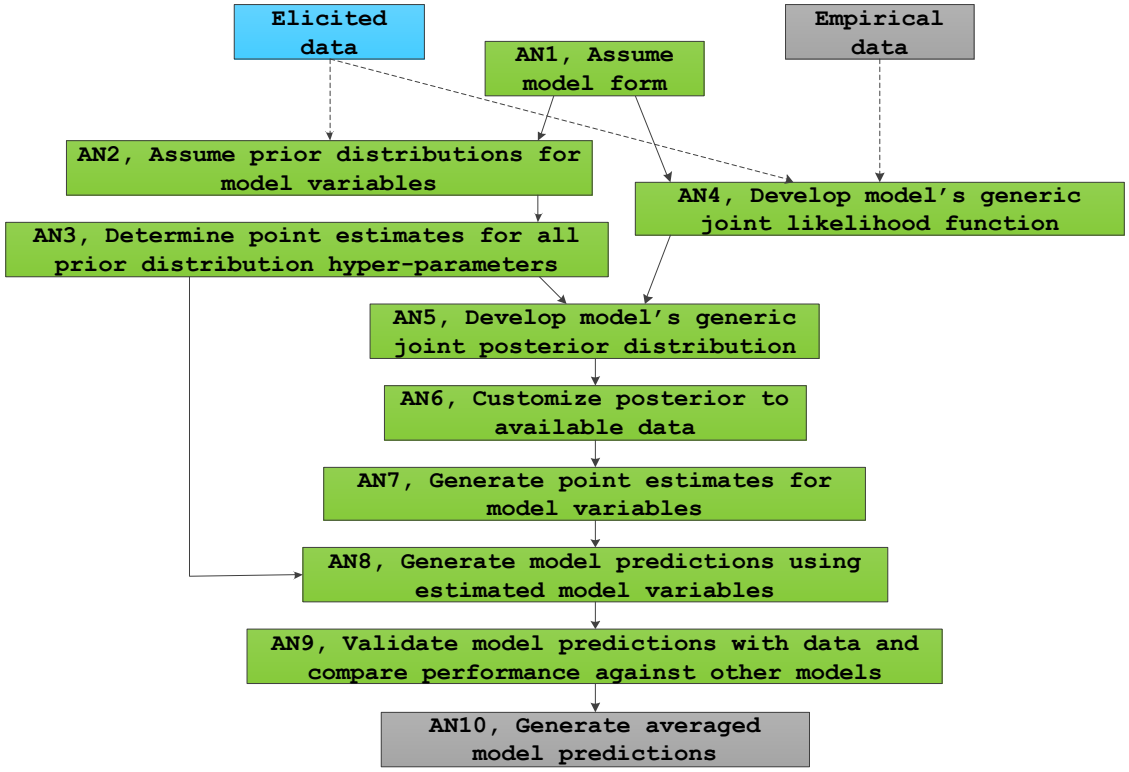


Figure 4-3: Bayesian analysis steps AN1-AN10

Another advantage to using the Bayesian approach for analysis is that results from multiple models may be combined to produce averaged model predictions[42]. In AN10, "Generate averaged model predictions" (see bottom of Figure 4-3), posterior-based predictions are generated by averaging the performance-weighted predictions from individual models. Bayesian model averaging for this application generally uses

$$\Pr(\Delta|\boldsymbol{D}) = \sum_{k=1}^{K} \Pr(\Delta|M_k, \boldsymbol{D}) \cdot \Pr(M_k|\boldsymbol{D}), \tag{4.8}$$

to predict the averaged, posterior-based observables, Δ, over models $k = 1,2, \ldots, K$, by performance-weighting the same from model M_k using $Pr(\Delta|M_k, \boldsymbol{D})$ with its corresponding posterior model probability,

$$Pr(M_k|\boldsymbol{D}) = \frac{\Pr(\boldsymbol{D}|M_k) \cdot \Pr(M_k)}{\sum_{l=1}^{K} \Pr(\boldsymbol{D}|M_l) \cdot Pr(M_l)}, \tag{4.9}$$

[42] BMA has also been used for modeling software reliability (Sarishvili and Hanselmann 2013, 1-8); however, my application was developed independently.

(Hoeting et al. 1999, 382-417). It is typically assumed that $Pr(M_k) \sim \frac{1}{K}$ (Fragoso, Bertoli, and Louzada 2017, n/a), and this results in $Pr(M_k|\boldsymbol{D}) = \frac{Pr(\boldsymbol{D}|M_k)}{\sum_{l=1}^{K} Pr(\boldsymbol{D}|M_l)}$. Thus, the posterior probability for each model given the data can be approximated as its normalized marginal likelihood.

The marginal likelihood of the data given model $M_{\hbar}$,

$$Pr(\boldsymbol{D}|M_{\hbar}) = \int_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\hbar}|M_{\hbar}, \boldsymbol{D}) \cdot \pi(\boldsymbol{\theta}_{\hbar}|M_{\hbar}) \cdot d\boldsymbol{\theta}_{\hbar}, \tag{4.10}$$

where $\boldsymbol{\theta}_{\hbar}$ is the set of parameters for model $M_{\hbar}$, may be estimated using the power-posterior approach (Friel and Pettitt 2008, 589-607) (see Appendix F.1).

Depending on the assumed form of $Pr(\Delta|\boldsymbol{D})$ and the prior distribution $\pi(\boldsymbol{\theta})$ for the model variables, the posterior distribution $\pi(\boldsymbol{\theta}\,|\,\boldsymbol{D})$, or its full-conditional distributions $\pi(\theta_j|\boldsymbol{\theta}_{-j}, \boldsymbol{D})$ for variables $j = 1,2, \dots, J$ (Gilks 1996, 75-88), may not be analytically tractable; this is one of the main drawbacks to Bayesian analysis. Fortunately, Markov chain Monte Carlo (MCMC) techniques (see Appendix E.4) provide a simulation-based alternative to analytical solutions. The key element of MCMC is the use of a specially constructed Markov chain that retains a stationary distribution equivalent to the desired posterior distribution (Robert and Casella 2004). These methods generally provide samples from the full-conditional posterior distributions, which are proportional to their true distributions, and the subsequent sample moments provide point-estimates for their corresponding variables' distributions.

4.3- Data gathering

The sources used for data-gathering are discussed in three parts: Sub-section 4.3.1 describes the expert calibration; Sub-section 4.3.2 explains the elicitation of expert judgment in the baseline scenarios and describes the empirical data obtained from the NVD; and Sub-section 4.3.3 explains the approach used for eliciting multivariate data.

4.3.1 Expert validation and calibration

Expert validation is necessary for data cleansing of expert opinion data, and expert calibration supports data aggregation under the Cooke classical model. This sub-section respectively outline the methodology details for validation and calibration.

The qualitative assessment consists of a single questionnaire that queries formal education, relevant work experience, and a participant self-assessment. Table 4-1 provides the questions supporting the assessment. Each expert's

experience level is assessed using the answers to validation questions VQ1-8. The researcher determines if the participant's response to VQ1-8 warrants their designation as an expert. Those participants not attaining this designation or providing less than competent self-assessment scores in VQ9 will have their data removed from the dataset.

Table 4-1: Experience questions

ID	Question
VQ1	What formal higher education have you completed (include degree(s), with associated major and minor)?
VQ2	How many hours of professional training have you completed which is directly related to software security, software reverse engineering (RE), or vulnerability assessment?
VQ3	How many hours of professional training have you completed that is directly related to software security, software RE, or vulnerability assessment?
VQ4	What processor architectures and assembly languages are within your technical expertise?
VQ5	What higher-level software languages are within your technical expertise?
VQ6	How often do you currently perform software RE or vulnerability assessment on assembly language artifacts? (daily, several times in a week/month/year)
VQ7	How often do you currently perform software RE or vulnerability assessment on higher-level language artifacts (e.g., C or Java source, or decompiled code)? (daily, several times in a week/month/year)
VQ8	Of the Dreyfus model of skill acquisition categories, which ones describe personnel you have worked with (including yourself) in software RE or vulnerability assessment? (Novice, Advanced beginner, Competent, Proficient, Expert)
VQ9	Classify your own personal skills at software vulnerability analysis and RE, according to the Dreyfus model. (Novice, Advanced beginner, Competent, Proficient, Expert)

The supporting material includes two network-based client-server model architecture applications, "CarRace"[43] and "ChatClient"[44] (Ceccato et al. 2014, 1040-1074). The first, "CarRace" (see Figure 4-4), is a simple game in which players compete by racing cars. The JAVA language implements the client application that is obfuscated using opaque predicate methods; it consists of 14 classes and has 3.783 KLOC (thousands of lines of code) when decompiled (Ceccato et al. 2014, 1040-1074). The second, "ChatClient" (see Figure 4-5), is a simple design providing the functionality for multiple users to have shared text based conversations. The JAVA language also implements the client application; it consists of 13 classes and has two binary versions. "ChatClient" is obfuscated using identifier-renaming methods (version contains 1.030 KLOC when decompiled) and separately using opaque predicates methods (version contains 3.642 decompiled KLOC) (Ceccato et al. 2014, 1040-1074).

[43] Ceccato et al. developed "CarRace" for a case study presented in (Ceccato et al. 2007, 27-36).

[44] https://sourceforge.net/projects/jchat/

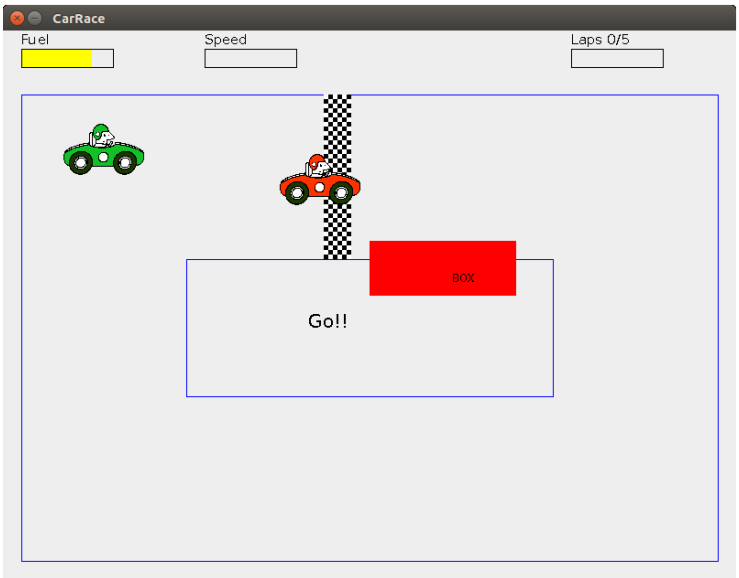


Figure 4-4: CarRace

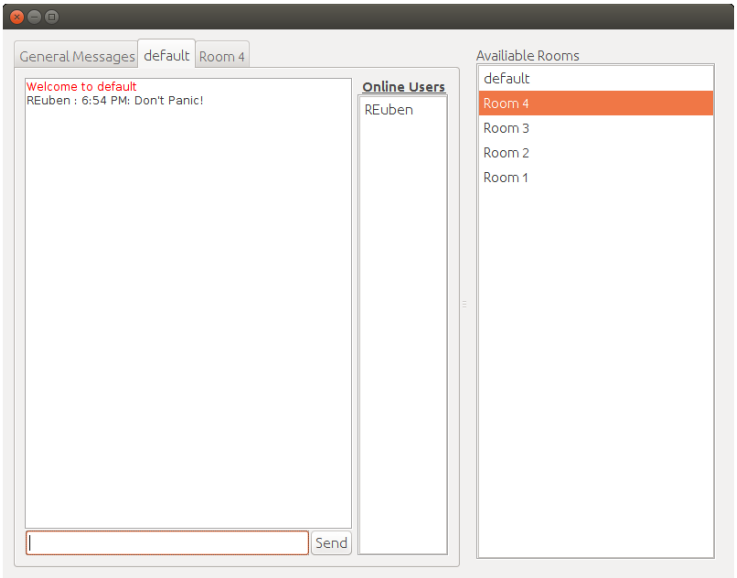


Figure 4-5: ChatClient

The practical assessment consists of two tasks (see Table 4-2) specific to the "ChatClient" application obfuscated using opaque predicates. In the first task, experts perform RE to answer a question about the "ChatClient" application's design. The second task requires the additional steps necessary to demonstrate operation of a newly requested design feature. All experts must answer correctly, and the practitioners record completion times. An incorrect answer or a completion time greater than 60-minutes results in practical assessment disqualification for the participant and their data being removed from the dataset.

Table 4-2: Practical assessment exercise questions

ID	Question
T2	"When a new user joins, the list of the displayed "Online users" is updated. Identify the section of code that updates the list of users when a new user joins (report the class name/s and line number/s with respect to the source)" (Ceccato et al. 2014, 1040-1074)
T4	"Messages are sent and displayed with the timestamp that marks when they have been sent. Modify the application such that the user sends messages with a constant timestamp = 3:00 PM." (Ceccato et al. 2014, 1040-1074)

Assessment execution proceeds with the practitioner providing a computer to participants with the Eclipse JAVA development environment, decompiled source code for the client application, and the server binary preinstalled. As not all persons are familiar with every tool, the participants receive brief instructions explaining how to perform common tasks within the Eclipse JAVA development environment, such as starting the JAVA server, performing text searches, and traversing the client application in the debugger.

The practitioner calibrates workshop participants' project estimation skills for security assessments indirectly through measuring their ability to estimate RE task performance of others. Recent research by Ceccato et al. (Ceccato et al. 2014, 1040-1074), investigating RE performance on the "CarRace" and "ChatClient" JAVA applications, provides information to generate 12 calibration questions. Calibration activities[45] require each participant to estimate the RE times for the tasks listed in Table 4-3.

Table 4-3: Calibration questions

ID	Question	Actual time for correct responses (ψ)		$\frac{No.\ correctly\ completing}{No.\ of\ participants}$
Car, T1, OP	"In order to refuel the car has to enter the box. A red rectangle delimits the box area. What is the width of the box entrance (in pixels)?" (Ceccato et al. 2014, 1040-1074)	Min	2	$\frac{16}{21}$
		Median	6.5	
		Max	60	
Car, T4, OP	"The fuel constantly decreases. Modify the application such that the fuel never decreases." (Ceccato et al. 2014, 1040-1074)	Min	3	$\frac{13}{17}$
		Median	8	
		Max	28	
Chat, T1, IR	"Messages going from the client to the server use an integer as header to distinguish the	Min	5	$\frac{14}{29}$
		Median	20	

[45] Participants complete the calibration activities prior to the practical assessment activities and separate binaries (obfuscated using a different algorithm) form the basis of the calibration activities.

ID	Question	Actual time for correct responses (ψ)		$\frac{No.\ correctly\ completing}{No.\ of\ participants}$
	type of the message. What is the value of the header for an outgoing public message sent by the client?" (Ceccato et al. 2014, 1040-1074)	Max	50	
Chat, T3, IR	"Messages are sent to a given room, if the user is registered in the room and if the message is typed in the corresponding tab. Modify the application such that all the messages from the user go to 'Room 1' without the user entering the room." (Ceccato et al. 2014, 1040-1074)	Min	4	$\frac{9}{19}$
		Median	11	
		Max	20	

Several necessary assumptions, that generalize the previous experiment, set up the question scenarios. These scenarios describe experiments where computer science graduate students performed RE exercises on obfuscated decompiled source code. Participants are instructed to assume all subjects in the experiment had: academic JAVA programming experience, no applied RE experience, and a software security background limited to academic courses. Participants are also provided with the code used in the experiment (to assist in assessing the level of effort) and the ratio of students correctly completing the task out of the total number (to assist in skill estimation, see Table 4-3).

The researcher then elicits *correct* activity completion time estimates for the fastest student, slowest student, and group average. Participants respond using the CCM's probability distribution point estimate format in Table 4-4. Table 4-5 and Table 4-6 contain the corresponding instructions and Figure 4-6 provides an illustrative example.

Table 4-4: Calibration question answer format

Fastest individual time	Median group time	Slowest individual time
5%=_____	5%=_____	5%=_____
50%=_____	50%=_____	50%=_____
95%=_____	95%=_____	95%=_____

Table 4-5: Expert instructions for the calibration exercises

For those completing the exercise successfully, estimate the fastest individual time, slowest individual time, and median group time for each task. Provide 5%, 50%, and 95% threshold estimations for each answer (in the probability distribution form explained below). Ensure that your estimates account for all *possible* time values.

Table 4-6: Calibration answer format explanation

Imagine that subjects perform 100 independent, but similar, analysis experiments. Each of these 100 experiments has values for the fastest, mean, and slowest times. Grouping these values results in sets for the fastest, mean, and slowest times (with 100 values in each set). Separately for each set, assume subjects rank order values and then assign these times into one of the four bins above. Then, bin_1 contains the 5 fastest values, bin_2 contains the lower middle 5%-50% values, bin_3 contains the upper middle 50%-95% values, and bin_4 contains the 5 slowest values. Consequently, the 50% value (which divides bin_2 and bin_3) defines the mean threshold, the 5% value specifies the upper threshold of bin_1, and the 95% value describes the lower threshold of bin_4.

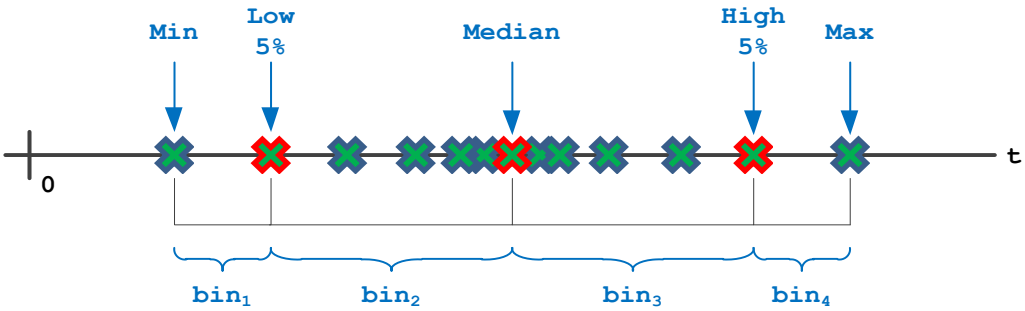


Figure 4-6: Calibration module answer format example

4.3.2 Phase I ("black-box")

Details for the Phase I data-gathering (see the grey E1-E4 and DG boxes in Figure 4-1, center) – that supports a "black-box" model for simulating cumulative discoveries in the baseline SR and SAP environment – are now detailed in two parts. The elicited data is explained by Sub-section 4.3.2.1, and Sub-section 4.3.2.2 describes the empirical data-gathering from the National Vulnerability Database (NVD).

4.3.2.1 Elicited data

The overview of analysis preparation details for the baseline SR and SAP's discovery model and the steps used to gather $\boldsymbol{D}_0$ (i.e., E1-4 and DG), which contains the corresponding expected discoveries (dependent variable) over time (independent variable), given $\boldsymbol{x}_0$ and γ (defined below), or $E[N(t|\boldsymbol{x}_0, \gamma)]$, is as follows. In E1, scenarios are constructed that describe five, assumed baseline SR and SAP combinations for software comparable to web-browsers (i.e., $\{\boldsymbol{x}_0, \gamma\}$, for which the specified total vulnerabilities present in each scenario's SR are assumed to be $\gamma = 148,55,20,7,3$ respectively). Table 4-8 and Table 4-9 contain the corresponding instructions and Figure 4-7 provides an illustrative example. For E2, the elicitation sessions for expert calibration and baseline SR and SAP data are performed and the experts provide answers to the calibration questions using the probability distribution format in Table 4-7, and for $E[N(t|\boldsymbol{x}_0, \gamma)]$ in the 10-calendar week time intervals defined by $[t_{i-1}, t_i)$ for $i = 1,2, \dots, \mathcal{I} = 5$ in each scenario. In E3,

the researcher must specify CCM inputs that define the intrinsic range, k, and the calibration power, ρ, [46]. Then, the baseline SR and SAP data are cleansed and the experts are calibrated using Cooke's method with the CCM settings and the calibration data. In E4, performance-based weights (computed using Cooke's method (Cooke 1991)) are used to aggregate the baseline SR and SAP data from all the remaining experts into one dataset. Then, data point estimates for $E[N(t|\boldsymbol{x}_0, \gamma)]$ are generated (using the aggregated distributions) for each time interval defined by the baseline scenarios. For each interval, the researcher uses these point estimates with piecewise linear distributions to extract 50,000 samples from the aggregated elicited baseline SR and SAP data distributions. This provides five point estimates specifying $E[N(t|\boldsymbol{x}_0, \gamma)]$ for each $\{\boldsymbol{x}_0, \gamma\}$ pair.

Table 4-7: Data elicitation answer format for a single SR and SAP, with example data entries

Interval	**5% (least)**	**Median**	**95% (most)**
$E[N_0(t_1\|\gamma) - N_0(t_0\|\gamma)]$	0	1	3
$E[N_0(t_2\|\gamma) - N_0(t_1\|\gamma)]$	1	3	5
$E[N_0(t_3\|\gamma) - N_0(t_2\|\gamma)]$	2	4	7
$E[N_0(t_4\|\gamma) - N_0(t_3\|\gamma)]$	2	4	3
$E[N_0(t_5\|\gamma) - N_0(t_4\|\gamma)]$	4	3	1

Table 4-8: Expert instructions for baseline SR and SAP elicitation

Assume you have an academic software security team performing vulnerability assessment and reverse engineering in the baseline environment defined below. Assume there are γ vulnerabilities present in the software release. Assume the academic security team performing the work has one licensed copy of the software and states that additional product units may be purchased using x_{25} funds (without acquisition delay). Assume that all software in the product is part of the release, to assume that it executes natively on the available equipment (x_{26}), and to assume that each team staff member has individually licensed RE software along with a dedicated personal computer (PC). Assume that all personnel and non-product material resources defined in the baseline SR&SAP are operationally available throughout the specified duration and that the product hardware failures are only dependent on the triggering of any specified anti-tampering features (x_{40}). Assume a random mixture of appropriate vulnerability types amongst the assumed faults present in each scenario (i.e., γ). Assume each discovery requires an additional three hours of static or dynamic analysis to verify the fault and one hour to document.

What would be the discoveries in the following intervals: [0,10), [10, 20), [20, 30), [30, 40), [40, 50) weeks? Provide 5%, 50%, and 95% threshold estimations for each answer (in the probability distribution form explained below). Ensure that your estimates account for all *possible* discoveries and that you follow the additional rules listed below.

Response rules:

(1) The cumulative sum of interval counts for each point estimate column (5%, 50%, and 95%) must be less than or equal to γ.

[46] This was determined manually through adjustment of the calibration power input until achieving maximal virtual DM performance (Cooke 2008, 775-777).

(2) The cumulative sum at each sequential interval for the 5%-point estimate column must be less than or equal to the 50%-point estimate column counterparts. Likewise, the cumulative sum at each sequential interval for the 50%-point estimate column must be less than or equal to the 95%-point estimate column counterparts.

Table 4-9: Baseline elicitation module answer format explanation

Imagine that subjects perform 100 independent, but similar, analysis experiments. Each of these 100 experiments has values for lowest, mean, and highest number of discoveries per interval. Grouping these values results in sets for the lowest, mean, and highest number of discoveries per interval (with 100 values in each interval set). Separately for each set, assume subjects rank order values and then assign these counts into one of the four bins above. Then bin_1 contains the 5 lowest values, bin_2 contains the lower middle 5%-50% values, bin_3 contains the upper middle 50%-95% values, and bin_4 contains the 5 highest values. Consequently, the 50%-value (which divides bin_2 and bin_3) defines the mean threshold, the 5%-value specifies the upper threshold of bin_1, and the 95%-value describes the lower threshold of the bin_4.

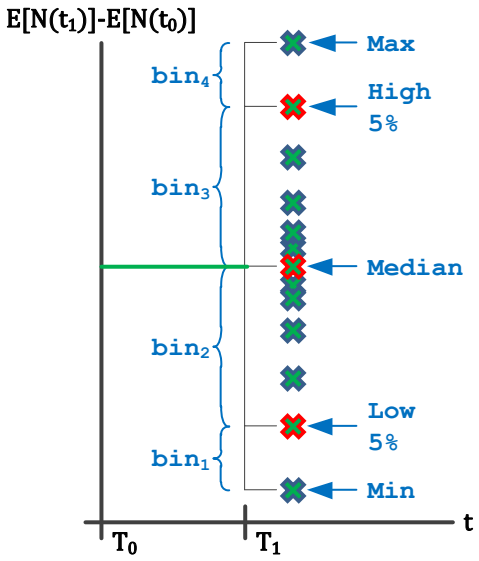


Figure 4-7: Baseline elicitation module answer format example

4.3.2.2 Empirical data

Bayesian analyses can also include empirical data-gathering (step DG) for estimation of the posterior distributions. Unfortunately, public databases for historical vulnerability discoveries over time do not capture *discovery* times. Consequently, the NVD *reporting* times for vulnerabilities found in four popular web-browsers, gathered by Nguyen, Massacci (2014, 1147-1162), were used as a proxy for discovery time[47]. Additionally, most of the 46 variables

[47] The suitability of reporting time as a proxy for discovery time is discussed by Massacci and Nguyen (2014, 1147-1162).

that define the SR and SAP combinations corresponding to the reporting times are not publicly available. Therefore, it is assumed that the SR and SAP combinations for the five, selected, NVD dataset examples are similar to $\{\boldsymbol{x}_0, \gamma\}$.

Several interesting releases in the NVD dataset merited preparing their data for study. The six examples chosen for analysis demonstration included releases 3.0 and 19.0 for Mozilla Firefox, 6.0 and 7.0 for Microsoft Internet Explorer, 1.0 for Google Chrome, and 1.0 for Apple Safari (F. Massacci and V. H. Nguyen 2014, 1147-1162; National Institute of Standards and Technology 2018) (see Table 4-10). In preparing $\boldsymbol{D}$ from each of these examples, the discoveries for each release were grouped in 10-week intervals (identical to those defined in Section 3.7) over the first 50 weeks of each release. These values were then altered such that their resulting $\boldsymbol{D}$ represented the cumulative discoveries (dependent variable) over time (independent variable) at each interval.

Table 4-10: Empirical datasets for cumulative discoveries over time

Product version	Cumulative discoveries
Safari 1.0	[0, 0, 0, 2, 2, 3]
Internet Explorer 6.0	[0, 2, 4, 5, 6, 7]
Internet Explorer 7.0	[0, 4, 6, 11, 15, 17]
Chrome 1.0	[0, 4, 6, 16, 23, 30]
Firefox 3.0	[0, 4, 12, 19, 40, 50]
Firefox 19.0	[0, 20, 76, 93, 134, 134]

(Massacci, Neuhaus, and Nguyen 2011, 195-208; Nguyen and Massacci 2012, 6-7; Nguyen and Massacci 2013, 493-498; F. Massacci and V. H. Nguyen 2014, 1147-1162; National Institute of Standards and Technology 2018)

4.3.3 Phase II ("clear-box")

Details for the Phase II data-gathering that support a "clear-box" model for simulating cumulative discoveries in arbitrary SR and SAP environments are now presented.

The blue E1-E4 boxes in Figure 4-1 (center) provide an overview of the steps used to gather $\boldsymbol{D}^1/\boldsymbol{D}^2$ (a ratio of two, distinct $n \times 1$ vectors of data defined below). In E1, a set of $j = 1, 2, \ldots, n = 50$ pairwise SR and SAP scenarios[48], in which each pairwise element defines a $\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\}$ pair (that associates with D_j^1/D_j^2), are constructed. By construction, these scenarios are relative to the baseline SR and SAP, as well as each other; and across the set, the SR and SAP

[48] This idea was conceptualized while studying the elicitation approach that Szwed et al. (2006, 157-177) used to enable their proportional probabilities model analysis (Note: their approach stems from Bradley & Terry (Bradley and Terry 1952, 324-345) and Pulkkinen (Pulkkinen 1994, 1-16)).

covariates are varied, and two approaches are used to avoid expert fatigue. First, the study is restricted only to the subset of variables listed in Table 4-12, that I believe are the most influential to discovery events. Second, the set of possible values for each of these variables is limited to five unique values per variable (also listed in Table 4-12). In E2, these questions consisting of pairwise sets of SR and SAP scenarios are presented to each of the experts, who provide $D_j^1/D_j^2 = E\left[N\left(\boldsymbol{x}_j^1\right)\right]/E\left[N\left(\boldsymbol{x}_j^2\right)\right]$, [49] or the expected discoveries (dependent variable) in $\boldsymbol{x}_j^1$ (independent variable) over the expected discoveries (dependent variable) in $\boldsymbol{x}_j^2$ (independent variable), for every $\left\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\right\}$ at $t = 50$ weeks. Table 4-11 provides an example question and the instructions are in Table 4-13 (Table G-1 and Table G-2 in Appendix G provide the complete set of $\left\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\right\}$, for $j = 1, 2, \ldots, n = 50$). E3-4 proceed similarly to what was already described in Phase I, to generate the performance-weighted aggregate $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset. However, in this case, the data are already provided in point estimate form (i.e., not in probability distribution form as was the case for the responses to the baseline scenarios).

Table 4-11: An example question appearing in one of the scenario sets

$\boldsymbol{x}_0$	$\boldsymbol{x}_j^1$	**Covariate Descriptions**	$\boldsymbol{x}_j^2$	D_j^1	D_j^2
1000.00	10000.00	Total number of functions (x_{41})	100000.00		
10000.00	5000.00	Product unit price (x_3)	1000.00		
3.00	--------	SA tool quality (x_{28})	--------		
10.00	--------	SA personnel effort (x_{23})	--------		
3.00	--------	SA average personnel quality (x_{29})	--------		
4.00	3.00	Level of dynamic access (x_{39})	--------		
0.25	--------	% software reused (x_1)	--------		
0.25	--------	% available design information (x_{18})	--------		
0.50	--------	% obfuscated software functions (x_{20})	--------		
0.50	--------	% cleansed software functions (x_{17})	--------		

Table 4-12: Covariates of interest in $\left\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\right\}$ and their values used

ID	**Description**	**Values (<u>Baseline</u>)**
x_{41}	Total number of functions	50, 250, <u>1000</u>, 10000, 100000
x_3	Product unit price (US $)	50, <u>500</u>, 1000, 5000, 100000
x_{28}	Assessment tool quality	1, 2, <u>3</u>, 4, 5
x_{23}	Assessment personnel effort (full-time equivalent, or FTE)	3, 5, <u>10</u>, 20, 30
x_{29}	Assessment average personnel quality	1, 2, <u>3</u>, 4, 5

[49] Recall from Equation (4.2) that there is no dependence on t or $\boldsymbol{x}_0$, when eliciting $\boldsymbol{D}^1/\boldsymbol{D}^2$.

ID	Description	Values (Baseline)
x_{39}	Level of dynamic access to artifacts	1, 2, 3, 4, 5
x_1	Percent of functions reused from previous release (0-100%)	0, 10, 25, 50, 75
x_{18}	Percent of available design information	0, 15, 25, 50, 100
x_{20}	Percent of obfuscated software functions (0-100%)	0, 25, 50, 75, 100
x_{17}	Percent of cleansed software functions (0-100%)	0, 25, 50, 75, 100

Table 4-13: Pairwise comparison elicitation instructions

Assume you have an academic software security team performing vulnerability assessment and reverse engineering in the defined pairs of environments. Assume there are $\gamma = 55$ vulnerabilities present in the software release. What would be the discoveries in the interval [0, 50 weeks] for each environment?

In applications, covariate inputs should be normalized to the baseline values and rescaled to their range of values in the questionnaire. Thus, one has the following baseline normalization equation for each covariate $k = 1,2, \dots, n$ in $\boldsymbol{x}_j$,

$$\tilde{x}_{j_k} = \frac{x_{j_k} - x_{0_k}}{\max(X_{1k}, X_{2k}, \cdots, X_{nk}) - \min(X_{1k}, X_{2k}, \cdots, X_{nk})}, \tag{4.11}$$

where the ~ accent denotes covariate baseline normalization, x_{0_k} is the k^{th} covariate in the baseline environment, and $\boldsymbol{X}$ is a $n \times n$ scenario matrix containing all defined SR and SAP combinations. Then by construction, this results in the normalized, baseline covariate inputs being $\tilde{\boldsymbol{x}}_0 = [0, 0, \dots, 0]$ and elements in the j^{th}, arbitrary SR and SAP $\tilde{\boldsymbol{x}}_j$ being constrained to the range $[-1, 1]$.

4.4 Bayesian analysis of VDM techniques

Here, the Bayesian analysis paradigm transforms the $f(t|\boldsymbol{x}_0)$, or $f_0(t)$, and $f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)$ of each candidate VDM technique into a stochastic functional form, from which analysts assume that the data observations originate. In other words, all the model functions, presented later in this section, modify a general form that represents the expected discoveries over time, or $F(t; \boldsymbol{\theta})$, into some stochastic distribution or process, $Pr(\Delta|t, \boldsymbol{D})$, where $\boldsymbol{D}$ is historical data over time from the phenomenon of interest, and Δ are the future observables.

This section has three parts: "black-box" VDM techniques; "clear-box" VDM techniques; and MCMCBayes modeling software.

4.4.1 Phase I ("black-box")

The "black-box" VDM technique performs Bayesian model averaging using a non-parametric NHPP model[50], five popular parametric forms of the NHPP, two common regression models, and two well-known growth-curve models. These models collectively support all the discovery rates illustrated in Figure 4-8. The general form of the candidate baseline models is $F(t;\boldsymbol{\Theta}|\mathbf{x}_0)$, or $F_0(t;\boldsymbol{\Theta})$. For further distinction, $H_0(t;\boldsymbol{\theta})$ can denote the same for the temporal parametric models, having the vector of parameters, $\boldsymbol{\theta}$, while $G_0(t;\Theta(t))$, with the temporal stochastic process $\Theta(t)$, can represent non-parametric models.

For each model, Table 4-14 lists the mean-value function (MVF), its variables and their descriptions, the forms for their prior distributions, and descriptions of hyper-parameters (see Appendix I for the associated prior distribution hyper-parameter values).

[50] Where it is applicable, as the Kuo-Ghosh NHPP model requires datasets having non-zero grouped interval data.

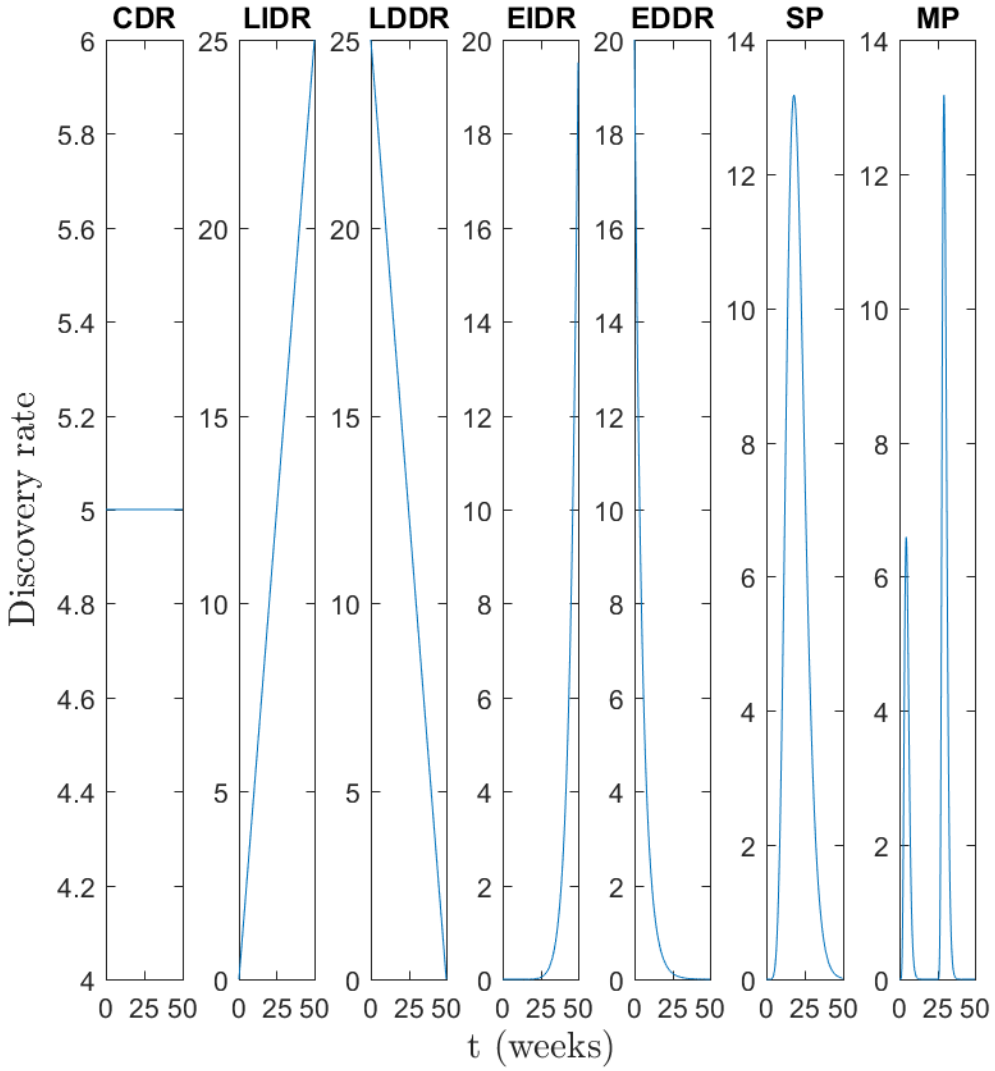


Figure 4-8: Discovery rates supported by the individual models

The abbreviations are: constant discovery rate (CFR), linearly increasing discovery rate (LIDR), linearly decreasing discovery rate (LDDR), exponentially increasing discovery rate (EIDR), exponentially decreasing discovery rate (EDDR), single pulse (SP), multiple pulse (MP)

Table 4-14: Vulnerability discovery models

Model name (M_k)	**Mean-value function**	**Variables**	**Variable descriptions**
		Prior distributions	**Hyper-parameter descriptions**
Kuo-Ghosh NHPP (M_1) (1997)	$G_0(t;\theta(t)) = \theta(t)$ (4.12)	$\theta(t) = \sum_{i=1}^{5} r_i$	r_i - variable for expected grouped discoveries in interval i, $r_i > 0$

Model name (M_k)	**Mean-value function**	**Variables**	**Variable descriptions**
		Prior distributions	**Hyper-parameter descriptions**
		$\pi(\Theta(t))\sim$ $\boldsymbol{GAMMA}(\Lambda^*(t), c)$, [51]	$\Lambda^*(t)$ - nondecreasing best guess set of point estimates over time for expected interval grouped discoveries, $\Lambda^*(0) = 0$, $\Lambda^*(t > 0) > 0$, $t \geq 0$ c - confidence in best guess set, $c > 0$
Brooks-Motley homogeneous Poisson process (M_2) (1980; Ozment 2006, 25-36)	$H_0(t;\boldsymbol{\theta}) = \zeta t$ (4.13)	$\boldsymbol{\theta} = \{\zeta\}$	ζ - expected maximum number of discoveries times a proportionality factor between expected discoveries and SAP performance, $\zeta > 0$
		$\pi(\zeta)\sim Gamma(a, b)$, [52]	a - shape, $a > 0$ b - rate, $b > 0$
Goel-Okumoto NHPP (M_3) (Goel and Okumoto 1979, 206-211; Rescorla 2005, 14-19)	$H_0(t;\boldsymbol{\theta}) = u\left(1 - e^{-\zeta t}\right)$ (4.14)	$\boldsymbol{\theta} = \{u, \zeta\}$	u - expected maximum number of discoveries, $u > 0$ ζ - proportionality factor between u and SAP performance, $\zeta > 0$
		$\pi(u)\sim Gamma(a, b)$ $\pi(\zeta)\sim Gamma(c, d)$	a, c - shape, $a > 0$, $c > 0$ b, d - rate, $b > 0$, $d > 0$
Goel's generalized Goel-Okumoto NHPP (M_4) (1985, 1411-1423; Okamura, Tokuzane, and Dohi 2013, 15-23)	$H_0(t;\boldsymbol{\theta}) = u\left(1 - e^{-\zeta t^{\kappa}}\right)$ (4.15)	$\boldsymbol{\theta} = \{u, \zeta, \kappa\}$	u - expected maximum number of discoveries, $u > 0$ ζ - proportionality factor between u and SAP performance, $\zeta > 0$ κ - power factor that scales the effect of ζ, $\kappa > 0$
		$\pi(u)\sim Gamma(a, b)$ $\pi(\zeta)\sim Gamma(c, d)$ $\pi(\kappa)\sim Gamma(g, h)$	a, c, g - shape, $a > 0$, $c > 0$, $g > 0$ b, d, h - rate, $b > 0$, $d > 0$, $h > 0$

[51] $\boldsymbol{GAMMA}(\Lambda^*(t), c)$ represents the Gamma process (see Appendix B.3); $\sim$ means "*distributed according to.*"

[52] $Gamma(a, b)$ represents the Gamma distribution (see Appendix A.2).

Model name (M_k)	Mean-value function	Variables	Variable descriptions
		Prior distributions	**Hyper-parameter descriptions**
Musa-Okumoto NHPP[53] (M_5) (1984, 230-238; Ozment 2006, 25-36)	$H_0(\tau;\boldsymbol{\theta}) = \frac{1}{\kappa}\cdot\ln(1+\zeta\kappa\tau)$ (4.16)	$\boldsymbol{\theta} = \{\zeta,\kappa\}$	ζ - expected maximum number of discoveries times a proportionality factor between expected discoveries and *initial* SAP performance, $\zeta > 0$ κ - SAP performance degradation rate of reduction per discovery, $\kappa > 0$
		$\pi(\zeta)\sim Gamma(a,b)$ $\pi(\kappa)\sim Gamma(c,d)$	a,c - shape, $a>0$, $c>0$ b,d - rate, $b>0$, $d>0$
Yamada's s-shaped NHPP (M_6) (1983, 475-484; Okamura, Tokuzane, and Dohi 2013, 15-23)	$H_0(t;\boldsymbol{\theta}) = u\left(1-(1+\zeta t)e^{-\zeta t}\right)$ (4.17)	$\boldsymbol{\theta} = \{u,\zeta\}$	u - expected maximum number of discoveries, $u>0$ ζ - steady state SAP performance, $\zeta>0$
		$\pi(u)\sim Gamma(a,b)$ $\pi(\zeta)\sim Gamma(c,d)$	a,c - shape, $a>0$, $c>0$ b,d - rate, $b>0$, $d>0$
Linear regression for a single independent variable (M_7) (DeGroot 2005; Alhazmi, Malaiya, and Ray 2007, 219-228)	$H_0(t;\boldsymbol{\theta}) = \beta_0+\beta_1 t$ (4.18)	$\boldsymbol{\theta} = \{\beta_0,\beta_1\}$, r	β_0 - discoveries at $t=0$, $-\infty<\beta_0<\infty$ β_1 - SAP performance rate, $-\infty<\beta_1<\infty$ r - precision for zero mean error term, ϵ, $r>0$
		$\pi(\beta_0)\sim Normal(a,b)$,[54] $\pi(\beta_1)\sim Normal(c,d)$ $\pi(r)\sim Gamma(g,h)$	a,c - mean, $-\infty<a<\infty$, $-\infty<c<\infty$ b,d - precision, $b>0$, $d>0$ g - shape, $g>0$ h - rate, $h>0$
	$H_0(t;\boldsymbol{\theta}) = \beta_0+\beta_1 t+\beta_2 t^2$ (4.19)	$\boldsymbol{\theta} = \{\beta_0,\beta_1,\beta_2\}$, r	β_0 - discoveries at $t=0$, $-\infty<\beta_0<\infty$ β_1 - SAP performance rate, $-\infty<\beta_1<\infty$ β_2 - SAP performance rate second degree factor, $-\infty<\beta_2<\infty$ r - precision for zero mean error term, ϵ, $r>0$

[53] For the Musa-Okumoto NHPP, security assessment time, τ (i.e., SA hours), is used in lieu of calendar time.

[54] $Normal(a,b)$ represents the Normal distribution (see Appendix A.3).

Model name (M_k)	**Mean-value function**	**Variables**	**Variable descriptions**
		Prior distributions	**Hyper-parameter descriptions**
Polynomial degree-2 regression for a single independent variable (M_8) (DeGroot 2005; Rescorla 2005, 14-19)		$\pi(\beta_0) \sim Normal(a, b)$ $\pi(\beta_1) \sim Normal(c, d)$ $\pi(\beta_2) \sim Normal(g, h)$ $\pi(r) \sim Gamma(p, q)$	a, c, g - mean, $-\infty < a < \infty$, $-\infty < c < \infty$, $-\infty < g < \infty$ b, d, h - precision, $b > 0$, $d > 0$, $h > 0$ p - shape, $p > 0$ q - rate, $q > 0$
Logistic (Verhulst) growth[55] (M_9) (Yamada, Ohba, and Osaki 1983, 475-484; Alhazmi and Malaiya 2008, 14-22; Winsor 1932, 1-8)	$H_0(t; \boldsymbol{\theta}) = \frac{\beta_1}{1 + e^{\beta_0 - \beta_2 \cdot t}}$ (4.20)	$\boldsymbol{\theta} = \{\beta_0, \beta_1, \beta_2\}$, r	β_0 - growth rate parameter that supports determining the point of inflection $t = \frac{\beta_0}{\beta_2}$, $N = \frac{\beta_1}{2}$, $\beta_0 > 0$ β_1 - carrying capacity (maximum population size, or discoveries), $\beta_1 > 0$ β_2 - population growth rate (i.e., SAP performance rate) that supports determining the point of inflection (see the description of β_0 above), $\beta_2 > 0$ r - precision for zero mean error term, ϵ, $r > 0$
		$\pi(\beta_0) \sim Normal(a, b) \cdot T(0,)$,[56] $\pi(\beta_1) \sim Normal(c, d) \cdot T(0,)$ $\pi(\beta_2) \sim Normal(g, h) \cdot T(0,)$ $\pi(r) \sim Gamma(p, q)$	a, c, g - mean, $-\infty < a < \infty$, $-\infty < c < \infty$, $-\infty < g < \infty$ b, d, h - precision, $b > 0$, $d > 0$, $h > 0$ p - shape, $p > 0$ q - rate, $q > 0$

[55] Logistic growth is symmetrical about the inflection point.

[56] $T(0,)$ denotes "*truncation below zero*."

Model name (M_k)	Mean-value function	Variables	Variable descriptions
		Prior distributions	**Hyper-parameter descriptions**
Gompertz growth[57] (M_{10}) (Yamada, Ohba, and Osaki 1983, 475-484; Okamura, Tokuzane, and Dohi 2013, 15-23; Winsor 1932, 1-8)	$H_0(t;\boldsymbol{\theta}) = \beta_1 \cdot e^{-e^{\beta_0-\beta_2 \cdot t}}$ (4.21)	$\boldsymbol{\theta} = \{\beta_0, \beta_1, \beta_2\}$, r	β_0 - growth rate parameter that supports determining the point of inflection $t = \frac{\beta_0}{\beta_2}$, $N = \frac{\beta_1}{e}$, $\beta_0 > 0$ β_1 - carrying capacity (maximum population size, or discoveries), $\beta_1 > 0$ β_2 - retardation growth rate parameter that slows the approach to β_1 and supports determining the point of inflection (see the description of β_0 above), $\beta_2 > 0$ r - precision for zero mean error term, ϵ, $r > 0$
		$\pi(\beta_0) \sim Normal(a,b) \cdot T(0,)$ $\pi(\beta_1) \sim Normal(c,d) \cdot T(0,)$ $\pi(\beta_2) \sim Normal(g,h) \cdot T(0,)$ $\pi(r) \sim Gamma(p,q)$	a, c, g - mean, $-\infty < a < \infty$, $-\infty < c < \infty$, $-\infty < g < \infty$ b, d, h - precision, $b > 0$, $d > 0$, $h > 0$ p - shape, $p > 0$ q - rate, $q > 0$

Bayesian analysis details for the individual NHPP models are as follows. Let $N(t)$ denote the number of cumulative discoveries at time t. This analysis of the discrete, counting-process form of the NHPP[58] requires grouped interval counts for the discovery data, $N(0) = 0$ and $N(t_i) - N(t_{i-1}) = n_i - n_{i-1}$ for $i = 1, \ldots, \mathcal{I}$ intervals, or $\boldsymbol{D} = \left\{N(t_{0j}) = 0, N(t_{ij}) = n_{ij};\ i = 1{:}\mathcal{I}, j = 1{:}\frac{m}{\mathcal{I}+1}\right\}$ (where the number of datapoints, m, is a multiple of $\mathcal{I}+1$). Assuming that $\boldsymbol{D}$ originates from a NHPP having MVF $F_0(t;\boldsymbol{\theta})$ (e.g., see M_1-M_6 in Table 4-14), the likelihood of $\boldsymbol{\theta}$ given $\boldsymbol{D}$, that stems from the NHPP's probability mass function (pmf) (Cinlar 1975), is

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) = \frac{\prod_{i=1}^{\mathcal{I}} \prod_{j=1}^{\frac{m}{\mathcal{I}+1}} \left(F_0(t_{ij};\boldsymbol{\theta}) - F_0\left((t_{(i-1)j};\boldsymbol{\theta})\right)\right)^{(n_{ij}-n_{(i-1)j})}}{\prod_{i=1}^{\mathcal{I}} \prod_{j=1}^{\frac{m}{\mathcal{I}+1}} (n_{ij} - n_{(i-1)j})!} e^{-\sum_{i=1}^{\mathcal{I}} \sum_{j=1}^{\frac{m}{\mathcal{I}+1}} \left(F_0(t_{ij};\boldsymbol{\theta}) - F_0\left(t_{(i-1)j};\boldsymbol{\theta}\right)\right)}. \quad (4.22)$$

[57] Gompertz growth is asymmetrical about the inflection point.

[58] The continuous, arrival-time form of the NHPP can also be used to model discovery events.

Then, the posterior-distribution equation,

$$\pi(\boldsymbol{\theta}|\boldsymbol{D}) \propto \frac{\prod_{i=1}^{J}\prod_{j=1}^{\overline{J+1}^{m}}\left(F_0\left(t_{ij};\boldsymbol{\theta}\right)-F_0\left(\left(t_{(i-1)j};\boldsymbol{\theta}\right)\right)\right)^{\left(n_{ij}-n_{(i-1)j}\right)}}{\prod_{i=1}^{J}\prod_{j=1}^{\overline{J+1}^{m}}\left(n_{ij}-n_{(i-1)j}\right)!}\, e^{-\sum_{i=1}^{J}\sum_{j=1}^{\overline{J+1}^{m}}\left(F_0\left(t_{ij};\boldsymbol{\theta}\right)-F_0\left(t_{(i-1)j};\boldsymbol{\theta}\right)\right)}\cdot\pi(\boldsymbol{\theta}), \quad (4.23)$$

results from using Equation (4.7) to combine the assumed form of the prior distribution, $\pi(\boldsymbol{\theta})$, with Equation (4.22). Simulation of future observables, $Pr(\Delta|t,\boldsymbol{D})$, proceeds by generating samples from a NHPP having MVF $F_0(t;\boldsymbol{\theta})$ and using the MCMC derived $\widehat{\boldsymbol{\theta}} = E[\pi(\boldsymbol{\theta}|\boldsymbol{D})]$ within.

Bayesian analysis details for the individual regression and growth-curve models are as follows. Let n_j denote the cumulative discoveries at the respective time instance t_j, for $j = 1$ to m data points. These models alternately assume that the j^{th} observation, D_j, for each of $j = 1$ to m, originates from the random variable $Y_j = H_0\left(t_j;\boldsymbol{\theta}\right) + \epsilon_j$, that includes an independent and identically distributed (i.i.d.) zero-mean error term from a Normal distribution,

$$\left\{\bigcup_{j=1}^{m}\epsilon_j\right\} \sim i.i.d.\, Normal(0, r^{-1}), \quad (4.24)$$

with unknown precision r (Hoff 2009). To condense, this essentially assumes that $\boldsymbol{D}$ stems from the Normal distribution

$$Normal(H_0(t;\boldsymbol{\theta}), r^{-1}), \quad (4.25)$$

having MVF $H_0(t;\boldsymbol{\theta})$ (e.g., see M_7-M_{10} Table 4-14) and precision r. Consequently, the likelihood of parameters $\{\boldsymbol{\theta}, r\}$ given $\boldsymbol{D}$ is

$$\mathcal{L}(\boldsymbol{\theta}, r|\boldsymbol{D}) = \left(\frac{r}{2\pi}\right)^{\frac{m}{2}} e^{-\frac{r}{2}\sum_{j=1}^{m}\left(n_j - H_0(t_j,\boldsymbol{\theta})\right)^2} \quad (4.26)$$

The corresponding posterior-distribution equation,

$$\pi(\boldsymbol{\theta}, r|\boldsymbol{D}) \propto \left(\frac{r}{2\pi}\right)^{\frac{m}{2}} e^{-\frac{r}{2}\sum_{j=1}^{m}\left(n_j - H_0(t_j,\boldsymbol{\theta})\right)^2} \cdot \pi(\boldsymbol{\theta}) \cdot \pi(r), \quad (4.27)$$

results from using Equation (4.7) to combine the assumed forms of the prior distributions, $\pi(\boldsymbol{\theta})$ and $\pi(r)$, with Equation (4.26). Generation of future observables, $Pr(\boldsymbol{\Delta}|t,\boldsymbol{D})$, proceeds by sampling from Equation (4.25) and using the MCMC derived $\hat{\boldsymbol{\theta}} = E[\pi(\boldsymbol{\theta}|\boldsymbol{D}, r)]$ and $\hat{r} = E[\pi(r|\boldsymbol{D}, \boldsymbol{\theta})]$ within.

An overview of the BMA model, that combines all $M_{\mathscr{k}}$, for $\mathscr{k} = 1, \cdots, K = 10$, models in Table 4-14, for the baseline SR and SAP and an outline of its Bayesian analysis steps AN1-7 is as follows. Here I denote the stochastic form for these diverse models generally using $Pr(\Delta_0|M_{\mathscr{k}}, t, \boldsymbol{D}_0)$, that is explained as the probability of the observables, or cumulative discoveries in the baseline over time, Δ_0, given model $M_{\mathscr{k}}$, t, and $\boldsymbol{D}_0$. The Markov-chain Monte Carlo (MCMC) based Bayesian analysis approach for AN1-7 uses $\boldsymbol{D}_0$ with each individual model and follows the averaged approach for estimating $Pr(\Delta_0|t, \boldsymbol{D}_0)$ outlined in Section 4.2.2.

Model choice is then performed as follows. First, the best individual model is determined using Bayes factors,

$$BF_{10} = \frac{Pr(\boldsymbol{D}|H_1)}{Pr(\boldsymbol{D}|H_0)}, \tag{4.28}$$

where H_0 is the null hypothesis model, and where H_1 is the alternate hypothesis model (Jeffreys 1961; Kass and Raftery 1995, 773-795). Interpretation of the BF score follows the guidance provided by Kass and Raftery (1995, 773-795): BF=1-3.2, "not worth more than a bare mention"; BF=3.2-10, "positive"; BF=10-100, "strong"; and BF>100, "very strong." Then, the BMA performance is evaluated against predictions from the best individual model using mean-square forecasting error (MSFE) (Clements and Hendry 1993, 617-637). For the $m \times 1$ original data vector $\boldsymbol{D}$, and $T \times m$ observables matrix Δ (generated for T parameter samples at every data point), it is generally described as the average of the sum of squares of prediction error, as measured using the observables against the original data, or

$$MSFE = \frac{1}{T}\sum_{\ell=1}^{T}\sum_{j=1}^{m}\left(D_j - \Delta_{\ell j}\right)^2. \tag{4.29}$$

Models having lower MSFE errors are interpreted as performing better.

4.4.2 Phase II ("clear-box")

The structure of Equation (4.1) supports a two-phased approach to Bayesian analysis of the "clear-box" discovery model $f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)$, that analyzes the first term in (4.1), $f_0(t)$, separately from the second term, $s(\boldsymbol{x}_i)$. This section presents two models for $s(\boldsymbol{x}_i)$, details Bayesian analysis of $s(\boldsymbol{x}_i)$, and then outlines how to use these analysis results with those from $f_0(t)$ (recall Sub-section 4.4.1).

This research introduces two parametric forms of the scaling term $s(\boldsymbol{x}_i)$ in Equation (4.1) and these are represented using the general form, $S(\boldsymbol{x}_i; \boldsymbol{c})$. Respectively, they each provide linear and exponential modulation of the

baseline model and both forms use a 46×1 regression parameter vector $\boldsymbol{c}$ to relate the influence of each entry in $\boldsymbol{x}_i$ to the scaling term. The linear form scales the baseline MVF using

$$S(\boldsymbol{x}_i; \boldsymbol{c}) = 1 + \boldsymbol{c}^T \boldsymbol{x}_i, \tag{4.30}$$

and the exponential form[59] (Cox 1972, 55-66) alternately modulates by using

$$S(\boldsymbol{x}_i; \boldsymbol{c}) = e^{\boldsymbol{c}^T \boldsymbol{x}_i} \tag{4.31}$$

Performing the AN1-7 Bayesian analysis steps with either form above by means of MCMC is straightforward, uses the dataset $\boldsymbol{D}^1/\boldsymbol{D}^2$, and assumes the expert responses follow Gaussian distributions (as in Szwed et al. (2006, 157-177)). For the linear form, I use Equation (4.30), let $y_j = D_j^1/D_j^2$ denote the data for $j = 1, 2, \ldots, n = 50$, and assume that the expert's response to the j^{th} scenario pair, $\{\boldsymbol{x}_j^1, \boldsymbol{x}_j^2\}$, is uncertain. Thus, I define the random variable (r.v.),

$$Y_j = \frac{1 + \boldsymbol{c}^T \boldsymbol{x}_j^1}{1 + \boldsymbol{c}^T \boldsymbol{x}_j^2} + \epsilon_j, \tag{4.32}$$

having the independent and identically distributed (i.i.d.) zero-mean error term from a Normal distribution,

$$\left\{ \bigcup_{j=1}^{n} \epsilon_j \right\} \sim i.i.d. Normal(0, r^{-1}), \tag{4.33}$$

For the exponential form, I similarly use Equation (4.31), alternately let $z_j = \ln(y_j)$, and define the r.v.,

$$Z_j = \boldsymbol{c}^T \cdot \left(\boldsymbol{x}_j^1 - \boldsymbol{x}_j^2\right) + \epsilon_j. \tag{4.34}$$

In both instances, I define flat priors on the elements in the model parameter vector $\boldsymbol{c}$, and a truncated flat prior (such that $r > 0$) on r, which is the precision for the zero-mean error term.

For AN8, forecasting of discoveries for arbitrary SR and SAP scenarios proceeds from using future observables from the baseline BMA model's Δ_0 (also see Sub-section 4.4.1) and the scaling model's Δ_S (described next). That is, I

[59] Note: modulating the NHPP baseline models using Equation (4.31) results in the well-known modulated Poisson process model .

combine them using Equation (4.1) and can then generate future observables from the i^{th} SR and SAP, Δ_i, using the resulting product distribution[60],

$$Pr(\Delta_i|t, \boldsymbol{x}_i, \boldsymbol{D}_0, \boldsymbol{D}^1/\boldsymbol{D}^2) = Pr(\Delta_0|t, \boldsymbol{D}_0) \cdot Pr(\Delta_S|\boldsymbol{x}_i, \boldsymbol{D}^1/\boldsymbol{D}^2). \tag{4.35}$$

So, to generate a sample from Equation (4.35), I separately obtain individual samples from the baseline and scaling functions[61] and then calculate their product. I can generate future scaling term observables, Δ_S, from

$$Pr(\Delta_S|\boldsymbol{x}_i, \boldsymbol{D}^1/\boldsymbol{D}^2) \sim Normal(S(\boldsymbol{x}_i; \boldsymbol{c}), r^{-1}) \tag{4.36}$$

by setting its MVF, $S(\boldsymbol{x}_i; \boldsymbol{c})$, to either Equation (4.30) or (4.31) (respectively, the one corresponding to the sampling approach used for $\pi(\boldsymbol{c}, r|\, \boldsymbol{D}^1/\boldsymbol{D}^2)$) and using $\tilde{\boldsymbol{x}}_i$, $\hat{\boldsymbol{c}}$, and $\hat{r}$ within.

Steps AN9-10 for the "clear-box" model proceed as follows. For AN9, the BMA alternative constructed for $f_0(t)$ is inherently the best model and selection of the ideal scaling function, $S(\boldsymbol{x}_i; \boldsymbol{c})$, is performed using MSFE. For completeness, I mention that step AN10 does not apply to analysis of $f_i(t, \boldsymbol{x}_i|\boldsymbol{x}_0)$ here.

4.5 MCMCBayes modeling software

Per modeling instance, the power-posterior approach approximates the integral on the right side of (4) using ideas from path sampling (Gelman and Meng 1998, 163-185) that increase the necessary MCMC sequences from one to n_ϕ (having typical values between $20 - 100$); each requires unique initialization, execution, and post-processing.

To reduce the manual effort involved in performing the power-posterior approach within sampling sequence analyses for multiple models, the MCMCBayes[62] framework was developed to *automate* MCMC sampling, model prediction, and model validation sequences[63]. MCMCBayes is a readily extensible, open-source, *object-oriented*,

---

[60] The expected value for a product distribution of the independent r.v.s x and y is $E[x \cdot y] = E[\mathrm{x}] \cdot E[\mathrm{y}]$ (Bohrnstedt and Goldberger 1969, 1439-1442).

[61] Using either the inverse transform method (see Appendix E.1) or standard sampling techniques (e.g., samples from $Normal$ r.v.s, see Appendix A.3).

[62] https://gitlab.com/reubenajohnston/MCMCBayes

[63] See Appendix E for general descriptions of relevant simulation techniques used by MCMCBayes and its dependencies.

software library that supports MCMC-based Bayesian analysis using MATLAB[64], with OpenBUGS[65] (Lunn et al. 2000, 325-337; Lunn et al. 2009, 3049-3067), JAGS[66] (Plummer 2003, 125), or Stan[67] (Carpenter et al. 2017).

In addition to the data for the phenomenon, MCMCBayes requires several forms of input for performing Bayesian analyses. One must supply general input values for controlling MCMC sampling, MCMC diagnostics, and summary statistics generation.

MCMCBayes provides output estimates for prior distributions, posterior distributions, and MCMC convergence diagnostics[68]. The distribution estimates include graphical histograms and numerical summary statistics such as the sampling mean, standard deviation, minimums, maximums, and highest posterior density (HPD) intervals. MCMC convergence diagnostic outputs include graphical output results for trace (i.e., raw sample), ergodic mean, and auto-correlation function (ACF) plots, as well as the Monte Carlo error (MCE) estimates (see Appendix E.4.4).

MCMCBayes forecasts using either the averaged or individual models and evaluates their prediction performance using BF, mean-square forecasting error (MSFE)[69], and visually. Bayes factors use the marginal likelihood values that are approximated here using the power-posterior's serial MCMC approach (see Appendix F.1) outlined by (Friel and Pettitt 2008, 589-607; Friel, Hurn, and Wyse 2014, 709-723), with $n_\phi = 30$ discretization steps and a temperature parameter $c_\phi = 5$. In addition to BF and MSFE, MCMCBayes supports visual evaluation of individual and averaged models by plotting their predictions alongside the original data.

MCMCBayes implements the AN1-AN9 steps in the Bayesian analysis sequence for each of the individual models and then AN10 for the averaged model. Most step instances are well-described by their general descriptions. However, AN3, "Determine point estimates for all prior distribution hyper-parameters," merits additional details concerning its tailored implementation.

---

[64] http://www.mathworks.com

[65] http://www.openbugs.net

[66] http://mcmc-jags.sourceforge.net

[67] http://mc-stan.org/

[68] In general, practitioners cannot validate achievement of convergence; however, in some cases, MCMC diagnostics (see Appendix E.4.4) imply non-convergence (Hoff 2009).

[69] For convenience, MCMCBayes computes MSFE using point-estimates for the model's posterior distributions (i.e., not using raw samples from model distributions).

To support construction of informative Bayesian priors for each of the models, AN3 is augmented. First, best judgment by the researcher should be used to specify one of the hyper-parameters in each prior distribution. Next, these would be provided to MCMCBayes, and for each $\{\boldsymbol{x}_0, \gamma\}$ scenario, it then uses maximum likelihood estimation (MLE, see Appendix D.1) or least squares (LS, see Appendix D.2) techniques for model fitment, using the derived $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ to generate a set of point estimates for each model variable. Then, because the chosen prior distribution for each model variable has known equations for its mean, MCMCBayes algebraically solves for the hyper-parameter corresponding with the specified one by assuming that the expected value for each variable is the resulting estimate from its MLE- or LS-model fitment.

Chapter 5. Results and discussion

In this chapter, outcomes are described and reviewed in four parts: Section 5.1 discusses the data cleansing; Section 5.2 highlights calibration results from using Cooke's method; Section 5.3 highlights the results from the "black-box" model data elicitation and aggregation, and analysis that includes baseline forecasting demonstrations from both individual and averaged models; and Section 5.4 highlights the results from the "clear-box" model data elicitation and aggregation, and analysis that includes several scaled forecasting demonstrations for arbitrary SR and SAP combinations.

5.1 Cleansing

In the expert judgment workshops, I gathered necessary data to support expert validation and afterwards performed data cleansing. The resulting data included self-assessment questionnaire responses and practical assessment results. Three persons self-assessed their skill level below the designated threshold (i.e., less than Dreyfus skill level 3, "Competent"). Additionally, the six persons who self-assessed as either "Competent" or "Proficient" successfully completed the practical assessment and listed satisfactory levels of training and experience. Therefore, removal of the information provided by the former three persons cleansed the elicited data set. See Table 5-1 and The abbreviations are: Electrical Engineering (E.E.), Computer Engineering (Comp. E.), Computer Science (C.S.), Systems Engineering (S.E.), Computer and Network Security (Comp. & Network Sec.), Information Assurance (Info. Assur.), Intel x86 architecture (x86), IBM PowerPC architecture (PPC), ARM architecture (ARM), Atmel AVR architecture (AVR), Microchip PIC architecture (PIC), Sun Microsystems SPARC architecture (SPARC), Digital Equipment Corporation Alpha architecture (Alpha), MIPS Technologies MIPS architecture (MIPS), Motorola HC11/6805 (Mot. HC11, MOT. 6805), Texas Instruments MSP 430 architecture (MSP 430).

Table 5-2 respectively for resulting individual experience and practical assessment data.

Table 5-1: Expert experience

Expert #	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
1	B.S. C.S., Math minor	12	36	ARM, x86, AVR, PIC	C, C++	Month	Week	Yes Yes Yes No No	Competent

Expert #	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
2	B.S. C.S., E.E. minor	10	24	x86, ARM	C, C++, JAVA	Month	Month	Yes Yes Yes Yes Yes	Proficient
REuben	B.S. E.E., B.S. C.S., M.S. S.E., PhD. S.E. (in progress)	80	36	x86, ARM, Mot. HC12, SPARC, PIC, AVR, PowerPC	Ada, C, C++, JAVA, C#, Visual Basic, Python	Week	Week	Yes Yes Yes Yes Yes	Proficient
5	B.S. Comp.E., PhD. C.S. (in progress)	20	6	x86, PPC, SPARC, Alpha, MIPS, Mot. HC11, ARM	C, C++, Python, JAVA, OCAML	Year	Week	Yes Yes Yes Yes Yes	Competent
6	B.A. C.S., Math minor, M.S. C.S.	10	36	ARM, x86, MSP 430, Mot. 6805	C, Python, Shell code	Week	Month	No No Yes Yes Yes	Proficient
7	B.S. Comp. & Network Sec., C.S. minor, M.S. Info. Assur.	40	20	x86	C, C++, Python, .NET, JAVA	Every 2 months	2-3 Week	Yes Yes Yes Yes Yes	Competent

The abbreviations are: Electrical Engineering (E.E.), Computer Engineering (Comp. E.), Computer Science (C.S.), Systems Engineering (S.E.), Computer and Network Security (Comp. & Network Sec.), Information Assurance (Info. Assur.), Intel x86 architecture (x86), IBM PowerPC architecture (PPC), ARM architecture (ARM), Atmel AVR architecture (AVR), Microchip PIC architecture (PIC), Sun Microsystems SPARC architecture (SPARC), Digital Equipment Corporation Alpha architecture (Alpha), MIPS Technologies MIPS architecture (MIPS), Motorola HC11/6805 (Mot. HC11, MOT. 6805), Texas Instruments MSP 430 architecture (MSP 430).

Table 5-2: Practical assessment results

Expert	Q1 (correct)	Q1 (time)	Q2 (correct)	Q2 (time)
1	Yes	12min	Yes	5min,25sec
2	Close enough[70]	15min	Yes	3min,24sec
REuben	Yes	21min	Yes	7min
5	Yes	13min	Yes	19min
6	Yes	50min	Yes	19min
7	Yes	27min	Yes	8min

[70] Fortunately, approximate results are acceptable when tossing horseshoes, throwing hand grenades (i.e., in the game "GoldenEye 007"), and performing reverse engineering.

5.2 Calibration

The outcomes from using Cooke's method are now discussed.

With the expert-provided, seed-question response data as input, a spreadsheet tool and Excalibur[71] were used to perform the expert calibration computations. For each of the experts, $e = 1,2,\ldots,E$, the spreadsheet tool preprocessed the information (e.g., estimated the individual, expert, distribution extremes), and Excalibur derived their calibration and information scores (i.e., $C(e)$ and $I(e)$) as well as their performance-based and equal aggregation weights (i.e., $w_{\mathrm{A}}(e)$, and $w_{eq}(e)$).

It is common practice to compare the performance of the CCM virtual decision-maker (DM) across the alternatives of global weight (DM_{A}), item weight (IDM_{A}), and equal weight (DM_{eq}) (Cooke and Goossens, L L H J 2008, 657-674). For this dataset, analysis identified the preferences as global, item, and equal weight DMs; all three DMs outperformed the calibration score of the best expert (not a good outcome). For each expert, Table 5-3 lists the numbers of seed questions assessed out of the total, calibration scores, information scores, performance weights, and equal weights.

Table 5-3: Expert scores ranked using Cooke's method

Id	**Seeds assessed out of** $\mathcal{S}$	$C(e)$	$I(e)$	$w_{\mathrm{A}}(e)$, **no DM**	$w_{eq}(e)$, **no DM**
DM_{A}	8/12	0.2472	1.001	--	--
IDM_{A}	8/12	0.1105	1.012	--	--
DM_{eq}	8/12	0.2472	0.3883	--	--
Expert 4	5/12	0.05281	1.283	0.7563	0.1667
Expert 6	4/12	0.01264	1.594	0.225	0.1667
Expert 2	2/12	0.0009623	1.74	0.01868	0.1667
Expert 3	1/12	0.0005433	2.053	0	0.1667
Expert 5	3/12	0.0005842	1.029	0	0.1667
Expert 1	2/12	0.0000397	1.028	0	0.1667

Expert performance from the calibration exercise, as measured by the CCM, was as follows. Only Experts 4, 6, and 2 received non-zero, performance-based weights[72], each having the respective normalized values of 75.6%,

[71] http://www.lighttwist.net/wp/excalibur

[72] It is common for the CCM to assign $w_{\mathrm{A}}(e) = 0$ to many experts (Ryan et al. 2012, 774-784).

22.5%, and 1.9%. The corresponding, and very low, significance level threshold, $A = 0.0009623$, was for Expert 2. Experts 3 and 5 received calibration scores in the same order of magnitude as 2. The score for Expert 1 was 10^{-1} lower in range.

Two significant factors contributed to the very low calibration scores. The first was lack of a CCM-approach training session—thus this type of instruction is strongly recommended for future studies. The second factor involved the data used to create the calibration questions. Participants should have been calibrated based on their ability to assess software security performance. As no quantitative data were available for generating directly relevant seed questions, it was necessary to use a proxy dataset. Unfortunately, even this proxy had several limitations. The trivial RE problems performed in this experiment were significantly reduced in complexity from the RE necessary for performing product security assessments; in addition, some of the relevant variables were not available (e.g., student RE skill levels).

5.3 Phase I ("black-box")

In this section, outcomes from the Phase I analysis are described and reviewed in two parts: Sub-section 5.3.1 discusses highlights of the elicitation and data aggregation; and Sub-section 5.3.2 highlights the results from the "black-box" analysis and includes forecasting demonstrations for several baseline SR and SAP examples.

5.3.1 Elicitation and data aggregation

Results from the $\boldsymbol{D}_0$ portions of elicitation and aggregation were as follows.

I executed data-gathering via a series of expert-judgment workshops. Four persons participated in the pilot (including myself) and five persons completed the main elicitation sessions. Several methodology adjustments were necessary during execution of the workshop. For example, during the pilot session, I modified some of the baseline variable values and, at the participants' request, translated values for variables described using the x_{39} metric (total number of functions) to approximate lines of code. Following the main workshop, I augmented the list of variables. This included refinement of the variables describing development languages, refinement of the variables describing available design information, insertion of an anti-tampering variable, and addition of variables defining the complexity due to virtualization and parallel processing. Consequently, in a follow-up elicitation session, participants adjusted their probability distributions per their beliefs on the updated baseline scenarios.

Figure 5-1 shows a sample response set using colored rectangles that represent the bins[73] for the probability distribution in each 10-week interval (where each is specified by a $q_{0\%}$, $q_{5\%}$, $q_{50\%}$, $q_{95\%}$, and $q_{100\%}$ set of thresholds that define a piecewise-linear probability distribution, see Appendix A.5) for cumulative discoveries in the $\{\boldsymbol{x}_0, \gamma = 148\}$ scenario.

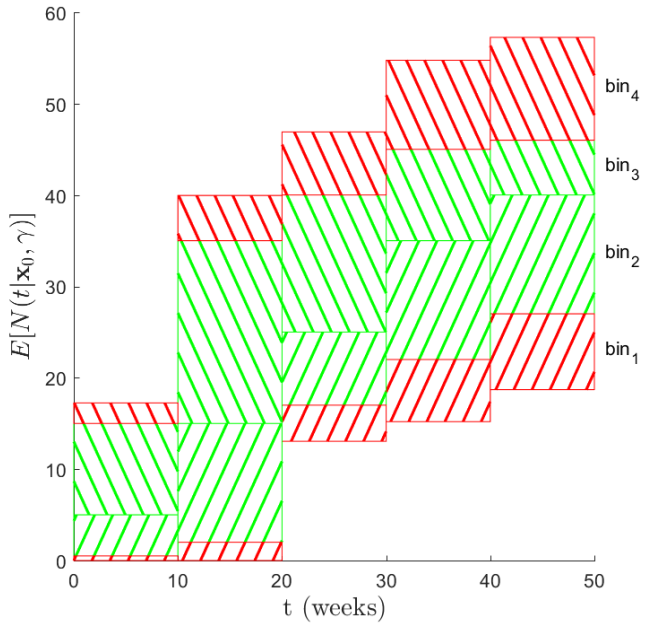


Figure 5-1: Example expert-response of probability distribution bins

Experts estimated the expected cumulative discoveries, $E[N(t|x_0, \gamma)]$), over time, t.

The final data aggregation sequence proceeded as follows. First, the individual expert data was initially combined using the computed CCM performance weights and then separately using the equal weights. Second, the aggregated data was rounded, which resulted in two sets of complete probability distributions for the expected cumulative discoveries at every time interval. Third, the individual and two aggregate distributions were used to separately derive corresponding sets of interval-means numerically via simulation. Fourth and finally, these interval-mean sets were each cumulatively summed for the baseline SR and SAP combinations of interest.

For each individual and aggregate probability distribution set, the computations above resulted in five data point estimates for $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ at $t = 10,20,30,40,50$ weeks. Assuming zero-discoveries starting at $t = 0$, Figure 5-2 depicts

[73] I.e., bin_1 contains the lowest 5% of the possible realizations from the specified distribution, bin_2 contains the lower-middle 5%-50%, bin_3 contains the upper-middle 50%-95%, and bin_4 contains the highest 5%.

a comparison of the cumulative individual and aggregate results for scenario $\{\boldsymbol{x}_0, \gamma = 55\}$: green indicates $w_{\mathrm{A}}(e)$ derived aggregate distribution, $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_{\mathrm{A}}}$; black indicates $w_{eq}(e)$ derived aggregate distribution, $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_{eq}}$; and blue indicates those derived from the individual distributions. Table 5-4 lists the aggregated results and Appendix H provides additional details and illustrations.

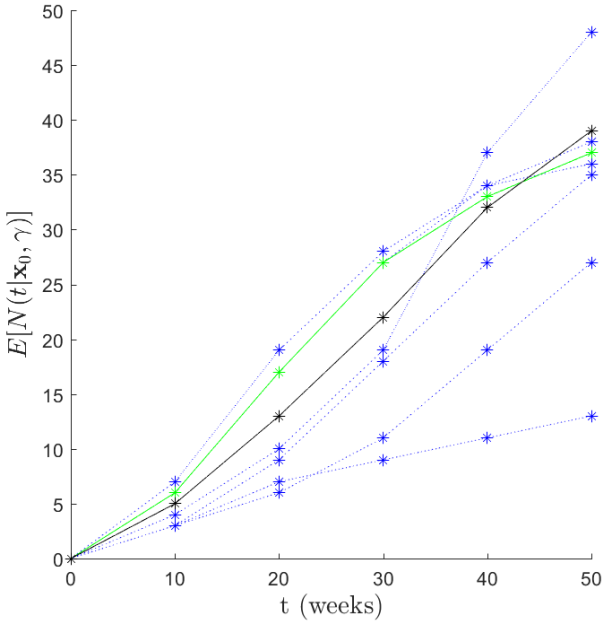


Figure 5-2: Elicited cumulative discoveries over time, example 1

Baseline SR and SAP with $\gamma = 55$

Table 5-4: Data results from simulating aggregated expert distributions

γ	**Equal weighting**	**Performance-based weighting**
148	[0, 17, 38, 62, 83, 96]	[0, 20, 41, 65, 79, 88]
55	[0, 5, 13, 22, 33, 39]	[0, 6, 17, 27, 33, 37]
20	[0, 2, 5, 9, 12, 14]	[0, 2, 6, 11, 12, 13]
7	[0, 0, 1, 2, 3, 4]	[0, 0, 2, 3, 3, 4]
3	[0, 0, 0, 1, 1, 1]	[0, 0, 0, 1, 1, 1]

The data aggregation results supported the following observations. First, given the specified baseline SR and SAP combinations, interval counts of discoveries over time provided by the experts were lower than expected. An obvious interpretation is that the experts believed that the level of difficulty for vulnerability discovery was high. Another, less obvious, rationale is that the experts assumed a population of vulnerability types that required increased

inspection effort to discover because, by design, the constructed elicitation scenarios associated with the SR and SAP, $\boldsymbol{x}_0$, instructed the experts to assume *typical* mixtures of vulnerability types in γ. Second, as the assumed quality increased (i.e., decreasing γ), the resulting differences in expected cumulative discoveries over time (i.e., between prior data $\boldsymbol{D}_{w_A}$ and $\boldsymbol{D}_{w_{eq}}$, respectively, the $w_A(e)$ and $w_{eq}(e)$ derived datasets) decreased. Third, the aggregated data indicated that discovery-event rates for the baseline SR and SAP combinations varied over time. The expert data supported RH1, as the performance-based aggregate for vulnerability discovery rates over time increased and then decreased for the specified baseline SR and SAP, having *constant* $\{\boldsymbol{x}_0, \gamma\}$ throughout the 50-week duration.

5.3.2 Bayesian analysis highlights

The outcomes from the Phase I Bayesian analysis of the "black-box" models are discussed in two parts: Sub-section 5.3.2.1 notes highlights of the individual model prediction performance; and Sub-section 5.3.2.2 presents predictions from the averaged model.

5.3.2.1 Individual models

Bayesian analysis provides two options for generating model predictions that are supported by MCMCBayes: using point estimates from either prior distributions or posterior distributions for the model variables to predict discoveries over time. Thus for each dataset, MCMCBayes was used to query point estimates from the prior distributions (MLE- or LS-derived) or to generate point estimates by averaging samples from the posterior distributions (MCMC-derived). For each model, M_k, these corresponding, variable point estimates, $\widehat{\boldsymbol{\theta}}_k$, were generated (see Appendix I for results) and then used to perform predictions. In generating each model's predictions, MCMCBayes performs interval set simulations (with $\widehat{\boldsymbol{\theta}}_k$) to generate a specified number of samples[74] for the already mentioned $i = 1$ to 5 time intervals, using each of the stochastic models. Subsequently, MCMCBayes computes interval predictions and these results are then cumulatively added together, over increasing i, to generate the corresponding predictions. Figure 5-3 depicts an example from $\boldsymbol{D}_{w_A}$ over time derived from scenario $\{\boldsymbol{x}_0, \gamma = 55\}$ with corresponding prior-based forecasting results from the Kuo-Ghosh NHPP model M_1. Here, the cyan dashed lines bound the inner two quartiles that contain 90% of the range of sampled values; the green * are the original data; and the cyan solid lines are the predictions. Figure 5-4 similarly provides a

[74] MCMCBayes uses the inverse transform method (see Appendix E.1) to simulate predictions from the NHPP models and standard sampling techniques (e.g., samples from $Normal$ random variables) for the other models.

posterior-based example (distinguished using red for the predictions and blue * for the NVD data) that additionally uses $\boldsymbol{D}$ from release 3.0 in Firefox to generate results.

MCMCBayes implements the corresponding model choice for the posterior-based, individual-model predictions as follows. It automatically identifies the model having the highest, marginal log-likelihood score and sets it as the alternate hypothesis H_1 while the other models represent each of the null hypotheses, H_0, to compare against H_1. Consequently, the resulting range for each pair of the model's BF score, using the already detailed guidance, signifies the strength of the hypothesis test. For example, when using $\boldsymbol{D}$ with $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_A}$ for $\gamma = 55$ and release 3.0 in Firefox, MCMCBayes identifies the Kuo-Ghosh NHPP model as H_1, having the highest marginal log-likelihood of -18.0662. Given that any H_0 having corresponding BF scores > 3 should be rejected, only Yamada's s-shaped NHPP models is additionally recommended for this example (i.e., all the other model alternatives are rejected). See Appendix I for a complete listing of the forecasting performance of the individual models.

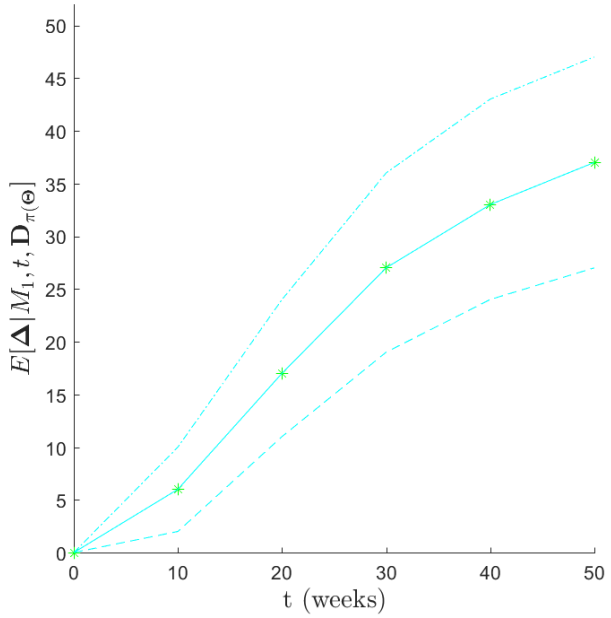


Figure 5-3: Example prior-based predictions for cumulative discoveries over time

The prior-based predictions for the Kuo-Ghosh NHPP model (M_1) used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_A}$ for $\gamma = 55$ to estimate $E[\Delta|M_1, t, \boldsymbol{D}_{\pi(\boldsymbol{\theta})}]$.

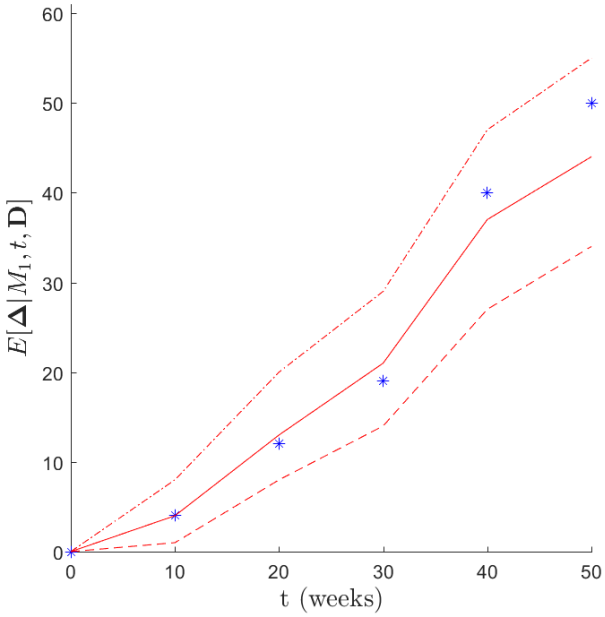


Figure 5-4: Example posterior-based predictions for cumulative discoveries over time

The posterior-based predictions for the Kuo-Ghosh NHPP model (M_1) used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_\alpha}$ for $\gamma = 55$ and release 3.0 in Firefox to estimate $E[\Delta|M_1, t, \boldsymbol{D}]$.

5.3.2.2 Averaged model

Through using the BMA, prediction quality can be further improved. Figure 5-5 plots the posterior-based averaged predictions compared with the corresponding predictions from the 10 individual models, for the same example in Sub-section 5.3.2.1. In the figure, blue represents the NVD data, red dotted lines represent the individual model predictions, and red indicates the averaged model predictions. The BMA results have as good or better prediction performance when compared to any of the single models and this is confirmed here by the MSFE score of 15.130673 when compared against the Kuo-Ghosh NHPP model's score of 28.240987 (where the Kuo-Ghosh NHPP is the individual best model as measured by its marginal log-likelihood score of -18.0662). This was also generally the case[75] for the other examples chosen for $\gamma = 148, 55, 20, 3$ (see additional illustrations and Tables in Appendix I for a complete listing of the forecasting performance) and is supportive of RH3.

[75] For $\gamma = 3$, the MSFE scores for the Verhulst growth-curve and BMA model were essentially identical.

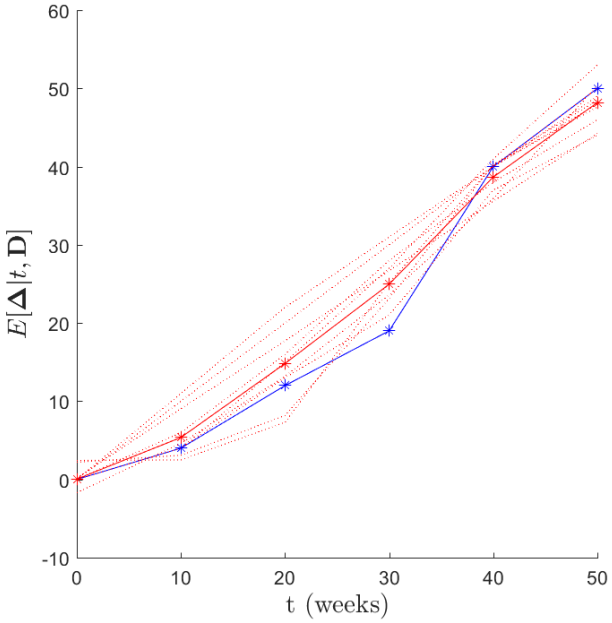


Figure 5-5: Posterior-based averaged predictions for cumulative discoveries over time, example 1

The posterior-based predictions for the BMA model used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_A}$ for $\gamma = 55$ and release 3.0 in Firefox to estimate $E[\Delta|t, \boldsymbol{D}]$.

5.4 Phase II ("clear-box")

In this section, outcomes from the Phase II analysis are described and reviewed in two parts: Sub-section 5.4.1 discusses highlights of the multivariate elicitation and data aggregation; and Sub-section 5.4.2 highlights the results from the multivariate analysis and includes forecasting demonstrations for several arbitrary SR and SAP examples.

5.4.1 Elicitation and data aggregation

Results from the $\boldsymbol{D}^1/\boldsymbol{D}^2$ portions of elicitation and aggregation were as follows. First, on many of the questions (specifically in 23 out of 50), the ratio responses from all the experts were generally in agreement to the pairwise SR and SAP set's influence on the phenomenon (e.g., see $j = 1,3,12,32$ responses for $E\left[N\left(\boldsymbol{x}_j^1\right)\right]/E\left[N\left(\boldsymbol{x}_j^2\right)\right]$ by *individual* experts in Table 5-5). Of the 27 questions having results not in general agreement, 21 had only one expert disagreeing with the consensus (e.g., see $j = 18,48$ in the same). In general, experts 1 and 2 did this more often; I avoid speculating as to what caused the former and attribute the latter to an experimental issue that leads to the next point. I did not properly baseline the experts and if I could repeat this Phase II experiment, I would instead specify the number of discoveries in $\boldsymbol{x}_0$, rather than use their individual $E[N(t|\boldsymbol{x}_0, \gamma)]$ numbers from Phase I (as was performed here). Lastly, I mention that

the problems associated with the previous point were overcome by using Cooke's method and carrying out the analysis using the performance aggregate dataset $\boldsymbol{D}^1/\boldsymbol{D}^2$.

Table 5-5: Example pairwise responses by individual experts

j	Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6
1	3	1	4	1.470588	1.2	2
3	0.860465	1	0.95	0.833333	0.875	0.8
12	1.255814	1	4	2.2	1.4	5.2
18	8.6	0.333333	0.763636	0.666667	0.829268	0.583333
32	0.813953	0.25	0.363636	0.575	0.825	0.588235
31	0.666667	20	5.5	3.846154	1.206897	3.9

5.4.2 Bayesian analysis highlights

This sub-section extends the discussion from Sub-section 5.3.2 with the related Phase II analysis of the second term, $s(\boldsymbol{x}_i)$.

MCMCBayes was used on the datasets to perform the AN1-AN9 steps in the Bayesian analysis sequence for both the linear and exponential alternatives. The analysis was straightforward and the results (e.g., mean, standard deviation, or SD, highest posterior density intervals (Box and Tiao 1992), or HPD[76], Monte Carlo error (Ntzoufras 2009), or MCE, and MSFE) for the linear form, are provided in Table 5-6 below (see Table I-11 in Appendix I for the results associated with the exponential form).

Table 5-6: Parameter estimates for the linear scaling term

Parameter	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
c_{41}	-0.38	0.08	{-0.46, -0.28}	0.0009
c_3	0.16	0.22	{-0.01, 0.45}	0.003
c_{28}	1.00	0.03	{0.95, 1.05}	0.0001
c_{23}	0.95	0.14	{0.69, 1.17}	0.0005
c_{29}	0.51	0.30	{-0.03, 0.96}	0.001
c_{39}	1.10	0.01	{1.07, 1.12}	0.00005
c_1	0.10	0.26	{-0.35, 0.54}	0.001
c_{18}	0.50	0.23	{0.13, 0.89}	0.0008
c_{20}	-0.12	0.19	{-0.43, 0.23}	0.0007

[76] For both HPD computations, the interval containing $100 \cdot (1 - \alpha)$ percent of the samples, where $\alpha = 0.05$, was measured.

Parameter	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
c_{17}	-0.46	0.16	{-0.71, -0.17}	0.0007
r	3.28	0.72	{2.17, 4.56}	0.004

Interpretation of the scaling term's parameters is as follows. First, I discuss the parameter vector $\boldsymbol{c}$, whose resulting estimates for each of its elements reveal the influence of that parameter on the phenomenon. Positive values have an additive effect to the phenomenon (i.e., more discoveries occur) and negative values have an opposing, subtractive effect. Also, the size of the number directly translates to the scaling effect associated with the corresponding covariate. In other words, the larger the value of the element in the scaling parameter, the more influential is the corresponding covariate (be it additive or subtractive) to the overall discoveries (as is relative to discoveries in the baseline). Second, I mention that the parameter r only affects the uncertainty of the resulting predictions.

The results from the Bayesian analysis appear to be very promising, and I briefly note the following points. First, the fact that nine out of ten of the elements in the parameter vector $\boldsymbol{c}$ supported their corresponding RH2 element (i.e., RH2.1 and RH2.3-10 all had expected positive or negative signs) and this was very encouraging. The only element that differed in sign was associated with variable x_3 (i.e., product unit price); thus, RH2.2 was not supported. I would have expected to see a negative influence on higher prices due to the fixed budget when anti-tampering measures, specified by x_{40}, were high (e.g., expensive smartphones turning into useless "bricks" or "paperweights" from anti-tampering measures should be a performance inhibitor in some scenarios). However, as I did not vary x_{40}, the results can be attributed to experimental setup. Second, I was also surprised by the higher influences associated with x_{28}, assessment tool quality, and x_{39}, level of dynamic access to artifacts[77]. For example, I would have expected that the influence from x_{23}, assessment personnel effort, or x_{29}, assessment average personnel quality, would have outweighed either. Last, I quickly mention that the linear scaling approach performed slightly better (or lower) via the MSFE measure (i.e., 0.321 vs 0.570) and this supports RH4.

Next, I briefly demonstrate forecasting of discoveries in arbitrary SR and SAP combinations, by extending the analysis of three web-browser examples using the multivariate methods described in Chapter 4. Linking these back to Figure 1-14, the examples do the following: the first demonstrates the influence from the SR variable x_{41} on Area2 (in

[77] The latter demonstrates how some security policies inhibit public discoveries using security mechanisms (e.g., those having restrictive clauses limiting the general user's level-of-access to software).

post-release assessments); the second does the same to *control* Area2 (specifically, when used to model the influence x_{39} has on post-release assessments); and the third demonstrates influence from a mechanism used to *control* Area1 (specifically, when used to model the influence x_{23} has on prerelease security testing[78]).

In the first example, I explore the resulting influence on discoveries caused by varying the size of the code being assessed. Figure 5-6 provides a graphical illustration that forecasts the expected cumulative discoveries over time, $E[N(t, \boldsymbol{x}_i)]$, using the BMA for $f_0(t)$ that was analyzed with the Mozilla Firefox v3.0 example and the linear form for $s(\boldsymbol{x}_i)$. Recall that in the baseline, x_{41}, the number of functions was specified to be 1,000. With the multivariate modeling extensions, one can alternately forecast discoveries for SR and SAP environments that similarly alter the baseline, and here, I specify several instances of $\boldsymbol{x}_i$ as equal to the baseline but additionally vary each of their x_{41} covariates (i.e., for all covariates except x_{41}, they are identical to those in $\boldsymbol{x}_0$). In Figure 5-6, I also point out the corresponding, negatively correlated effects to cumulative discoveries over time caused by the adjustments: $x_{41} = 50$, $x_{41} = 250$, $x_{41} = 10{,}000$, $x_{41} = 100{,}000$.

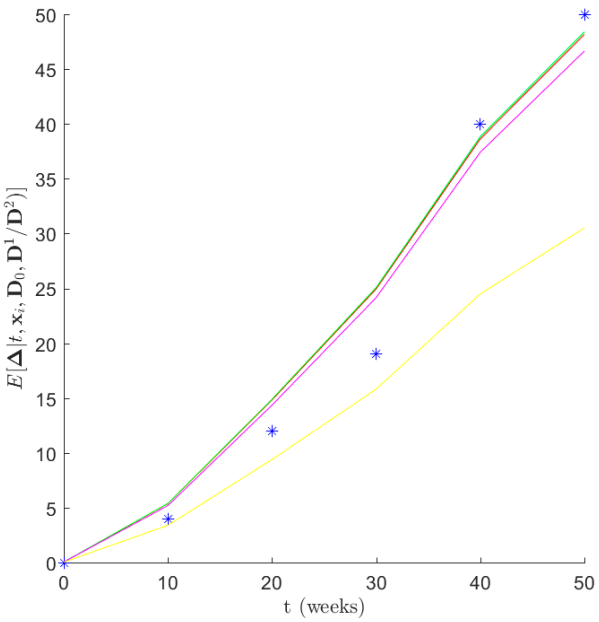


Figure 5-6: Linearly-scaled predictions for cumulative discoveries over time, example 1

[78] Albeit, as this dataset's timespan encompassed 50 weeks, the demonstration is purely academic.

Predictions are compared to Firefox's data from the assumed baseline (blue *) and highlight influence from varying x_{41}, where $x_{41} = 50$ (cyan[79]), $x_{41} = 250$ (green), $x_{41} = 1{,}000$ (red, baseline), $x_{41} = 10{,}000$ (magenta), $x_{41} = 100{,}000$ (yellow).

In the second example, I focus on what was alluded to earlier, regarding the influence from security policies that restrict the general user's level-of-access to software. Figure 5-7 forecasts $E[N(t, \boldsymbol{x}_i)]$ using the BMA for $f_0(t)$ that was analyzed with the Microsoft Internet Explorer v7.0 example and the linear form for $s(\boldsymbol{x}_i)$. Recall that in the baseline, x_{39}, the level of dynamic access to artifacts was set to Level 4 (full privileges, or administrative user mode, on target). With the multivariate modeling extensions, one can again forecast discoveries for SR and SAP environments that similarly alter the baseline, and here, I specify several instances of $\boldsymbol{x}_i$ that instead vary x_{39}. In Figure 5-7, I note the resulting and positively correlated effects to $E[N(t, \boldsymbol{x}_i)]$ caused by the alterations: $x_{39} = 1$, $x_{39} = 3$, and $x_{39} = 5$.

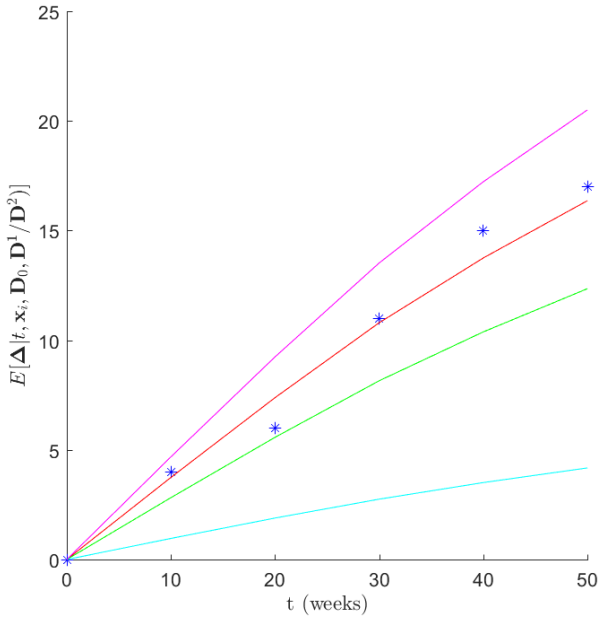


Figure 5-7: Linearly-scaled predictions for cumulative discoveries over time, example 2

Predictions are compared to Internet Explorer's data from the assumed baseline (blue *) and highlight influence from varying x_{39}, where $x_{39} = 1$ (cyan), $x_{39} = 3$ (green), $x_{39} = 4$ (red, baseline), and $x_{39} = 5$ (magenta)

In the final example, I explore the resulting influence on discoveries caused by varying the number of full-time equivalents performing an assessment. Figure 5-8 forecasts $E[N(t, \boldsymbol{x}_i)]$ using the BMA for $f_0(t)$ that was analyzed with

[79] Plots for $x_{41} = 50$ (cyan) and $x_{41} = 250$ (green) were almost identical.

the Apple Safari v1.0 example and the linear form for $s(\boldsymbol{x}_i)$. Recall that in the baseline, x_{23}, the personnel effort for the assessment, was specified to be 10 FTEs. With the multivariate modeling extensions, one can again forecast discoveries for differing SR and SAP environments; here, I specify several instances of $\boldsymbol{x}_i$ that vary x_{23}. In Figure 5-8, I note the corresponding, positively correlated, amplifying, and attenuating effects to $E[N(t, \boldsymbol{x}_i)]$ caused by the adjustments: $x_{23} = 1$, $x_{23} = 9$, $x_{23} = 11$, and $x_{23} = 25$. I also point out that increased resources to assessments for SR and SAP combinations with high quality (i.e., lower γ) have a lesser result, as there is a diminishing returns effect to discoveries.

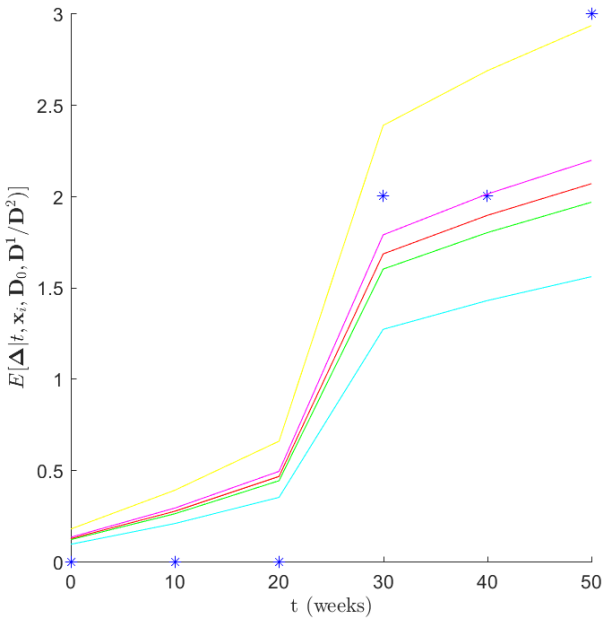


Figure 5-8: Linearly-scaled predictions for cumulative discoveries over time, example 3

Predictions are compared to Safari's data from the assumed baseline (blue *) and highlight influence from varying x_{23}, where $x_{23} = 1$ (cyan), $x_{23} = 9$ (green), $x_{23} = 10$ (red, baseline), $x_{23} = 11$ (magenta), and $x_{23} = 25$ (yellow)

Chapter 6. Limitations, usage recommendations, and future work

This chapter starts with a discussion on the limitations of the methodology, provides usage recommendations, and then lists ideas for future work.

This research deliberately omitted variables describing two controversial areas of ROI: political and military entities; and peer recognition. However, by using academic experts as the data source and defining assessment resources in the elicitation scenarios, I ensured that these omissions would not affect results.

An issue with the workshop results is the possibility of expert-response herding[80]. The workshop's plenary format meant that individual interviews were not possible, and I recommend that future researchers consider using them. Also, as a result of performing only one study with experts, I risked compromising the independence assumption[80]. This risk could be mitigated in replication studies that use different experts and calibration questions.

This methodology also avoids the much-debated topic of predicting the number of released vulnerabilities. Rather, it focuses on exploring vulnerability discovery on specified levels of release quality (recall Assumption AS1). Others might consider tailoring this methodology to model the influence of business decisions on both security quality and general quality. With the appropriate set of experts and an alternative set of variables that describe the release cycle and SAP or prerelease testing profile, researchers could explore the influence of product and prerelease process decisions on fault creation and prerelease discovery.

The power-posterior approach for estimation of the marginal likelihoods from the individual models, $Pr(\boldsymbol{D}|M_k)$, is sensitive to the choice of hyper-parameters (Friel and Pettitt 2008, 589-607). I try to mitigate this risk by developing the hyper-parameters using $\boldsymbol{D}_{\pi(\boldsymbol{\theta})}$ elicited via Cooke's method.

While performing the data analysis on the examples (see Sub-section 5.3.2.1), a major issue with the NVD data was observed that revealed related concerns and emphasized the importance of using expert-judgment-enhanced analysis. The problem was that, when comparing the available data for the four web-browser software products, no discernable

[80] Roger Cooke provided a brief review of the results (Cooke 2015).

patterns were found in the cumulative discoveries across the data—even though these products are functionally very similar. For example, there were neither consistent trends[81], in particular an overall decrease in cumulative discoveries, nor cyclic patterns having high discoveries following major releases and reduced discoveries afterwards. From top to bottom, Figure 6-1 depicts cumulative vulnerabilities, discovered over the first 50 weeks per release, for Mozilla Firefox, Google Chrome, Microsoft Internet Explorer, and Apple Safari. Clearly, factors not accounted for would explain the seemingly random patterns found (e.g., differences in release schedules, increasing skill and numbers of discoverers, significant new feature requirements, lack of discovery reporting, and so forth). However, state-of-the-art VDM techniques only model this phenomenon as a "black-box" (i.e., they do not account for variables such as those defined by $\boldsymbol{x}_0$). Therefore, as this research demonstrates, it is imperative to include expert judgment from software-makers and security professionals in the analysis.

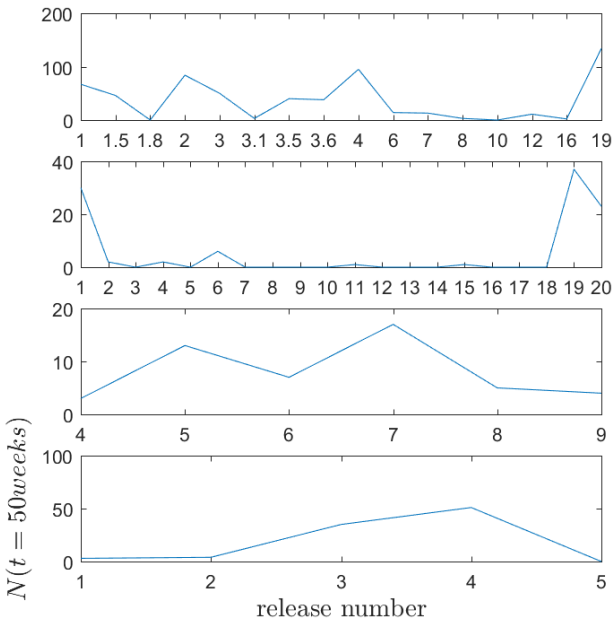


Figure 6-1: Total discoveries in the first 50 weeks following a release vs. release number

Images in top-to-bottom order respectively represent data from: Mozilla Firefox, Google Chrome, Microsoft Internet Explorer, and Apple Safari.

[81] Woo et al. (Woo et al. 2011, 50-62) mention the predictive performance limitations that VDM techniques have when trends change.

The two-phase approach results in a couple of limitations worth mentioning. The first limitation is the compounding of the uncertainty from both the baseline and scaling function terms and their product[82] (recall the structure of Equation (4.35)). The second limitation, which only applies to modeling $f_0(t)$ when using some of the individual models (i.e., not when using the BMA), is that the expected discoveries for arbitrary environments could become overly large for the model. For example, when using the Verhulst logistic growth model for a baseline function (Alhazmi and Malaiya 2008, 14-22; Yamada, Ohba, and Osaki 1983, 475-484), the model's carrying capacity limit could be easily exceeded due to excessive scaling. The former limitation could be addressed once multivariate empirical data becomes available (e.g., see the analysis of a scaled NHPP model in (Soyer and Tarimcilar 2008, 266-278)) and the effects from the latter could be constrained by limiting the scaling term (e.g., define the valid range for $s(\boldsymbol{x}_i)$ to be $[-2,2]$).

The following assumption is problematic: vulnerability discoveries over time for arbitrary SR and SAP combinations are a modulation of the same from the baseline SR and SAP[83]. Discovery rates for some SR and SAP combinations could differ significantly with respect to time as compared to the baseline. For example, researchers might expect discovery rates for a SR and SAP with high levels of assessment personnel effort, x_{23}, to differ over time when compared to an almost identical one with low levels. Therefore, they could develop a baseline model incorporating the time-effects of the variables. The use of effort-dependent NHPP models (Huang, Kuo, and Lyu 2007, 198-211; Kimura 2006, 256-261) as a baseline, cumulative-intensity function has the potential to address the problem for a subset of variables that detail expended, security-assessment resources. Additionally, one could augment the approach presented here to instead generate the modulation parameter vector in Equations (4.30) and (4.31), $\boldsymbol{c}_i$, for each $[t_{i-1}, t_i)$ interval (defined in Sub-section 4.3.2.1) for $i = 1,2,\cdots,5$.

It is important to point out that by using the assumption that equates each example NVD dataset to a baseline SR and SAP combination $\{\boldsymbol{x}_0, \gamma\}$, the corresponding $E[\Delta_0|t, \boldsymbol{D}_0]$ results from, and by extension the scaled $E[\Delta_i|t, \boldsymbol{x}_i, \boldsymbol{D}_0, \boldsymbol{D}^1/\boldsymbol{D}^2]$ predictions presented here, are only *demonstrations* of the methodology. This is because without the baseline covariates, I cannot determine the difference between the baseline and the desired SR and SAP. Thus, to

[82] The variance for a product distribution of the independent r.v.s x and y is $var[x \cdot y] = var[\mathrm{x}] \cdot var[\mathrm{y}] + var[x] \cdot var[\mathrm{y}]^2 + var[y] \cdot var[\mathrm{x}]^2$ (Goodman 1960, 708-713).

[83] The various modulations retain the shape of the baseline function, $f_0(t)$.

clarify, this limitation only applies to the future observables from the i^{th} SR and SAP and not the estimates for $E[\Delta_S|\boldsymbol{x}_i, \boldsymbol{D}^1/\boldsymbol{D}^2]$, as the latter is independent of $E[\Delta_0|t, \boldsymbol{D}_0]$.

Next, I briefly provide usage recommendations.

Given the issues with "black-box" modeling using the NVD data, it is clear that expert judgment is essential to bolstering the results. For web-browser software, the use of prior distributions stemming from the closest γ to the cumulative reporting events per release at $t = 50$ weeks is also recommended. In order to predict discoveries for a future release similar to those described here, researchers should perform simulations from all five of the examples and then work with experts to decide which γ is most realistic for the assumed vulnerabilities. Thus on one hand, a release with many new features and a compressed release schedule might require that γ would be higher. On the other hand, few new features and adequate time for prerelease testing might indicate that γ should be lower.

Regardless, for other combinations that deviate significantly from the single SR and SAP combination explored here for web-browsers, it will be necessary to perform additional elicitation with experts, and even better to consider using expert-judgment-enhanced multivariate approaches.

Finally, I briefly provide recommendations for future work.

The subjectivist approach used by the methodology, combined with Bayesian analysis techniques and expert-judgment elicitation, is supported by the following elements: 1) the "one-off nature of software" (Littlewood 1979, 103-110); 2) the rapid advance of techniques that threaten software security (Ryan et al. 2012, 774-784); and 3) inadequate public data on the rare phenomenon of vulnerability discovery (Ozment 2007, 6-11). Additional support would be produced from future research in the following areas: 1) gathering multivariate empirical data to perform posterior-based analysis; 2) gathering additional empirical data to perform posterior predictive validity tests; 3) gathering data to better generate calibration questions; 4) carrying out replication studies; and 5) extending the current research. For the latter, future studies could: 1) investigate the various influences of SR and SAP combinations on the discovery of specific vulnerability types over time; 2) gather data from experts outside academia and perform corresponding analyses; and 3) incorporate influences ROI confounding factors.

Chapter 7. Conclusion

Software vulnerabilities that enable well-known exploit techniques for committing computer crimes are *preventable*, but they continue to be present in software releases. In general, software security modeling techniques help software-makers make decisions on how best to reduce risk; in particular, this research introduces a paradigm-shifting improvement to these techniques: a multivariate form for vulnerability discovery modeling. Hence I hope that this research motivates the security community at large to instead use *multivariate* VDM techniques and to work towards improving their corresponding data-gathering.

This research has presented new techniques that apply structured expert-judgment data-gathering methods and introduced multivariate methods to modeling vulnerability discoveries in web-browser software[84]. Its capability to forecast expected discoveries over time for arbitrary SR and SAP combinations is groundbreaking. I also emphasize that its primary application is to provide software-makers with a means to explore influences on the discovery phenomenon, by variables they can control (e.g., through resource allocation and alteration of security policy). Obviously, the techniques presented here are entirely dependent on the quality of the multivariate data and also require the collection of significantly more information. Thus, I urge the software security community to foster an environment of open data by encouraging: 1) software-makers to make supporting metrics for SR available; and 2) security assessors with discoveries to make supporting metrics for SAP accessible.

[84] I additionally state that these techniques could easily be generalized and applied to modeling *any* complex phenomenon having non-decreasing counts over time.

References

Achcar, J. A., D. K. Dey, and M. Niverthi. 1998. *A Bayesian Approach using Nonhomogeneous Poisson Processes for Software Reliability Models*. Series on Quality, Reliability and Engineering Statistics. New Jersey: World Scientific. http://doi.org/10.1142/9789812816580_0001.

Alder, Berni J. and T E. Wainwright. 1959. "Studies in Molecular Dynamics. I. General Method." *The Journal of Chemical Physics* 31 (2): 459-466. http://doi.org/10.1063/1.1730376.

aleph1. "Smashing the Stack for Fun and Profit.", accessed 02/13, 2015, http://phrack.org/issues/49/14.html#article.

Alhazmi, O. H. and Y. K. Malaiya. 2008. "Application of Vulnerability Discovery Models to Major Operating Systems." *IEEE Transactions on Reliability* 57 (1): 14-22. http://dx.doi.org/10.1109/TR.2008.916872.

———. 2005. "Quantitative Vulnerability Assessment of Systems Software." Alexandria, VA, USA, IEEE, January 24-27. http://dx.doi.org/10.1109/rams.2005.1408432.

Alhazmi, O. H., Y. K. Malaiya, and I. Ray. 2007. "Measuring, Analyzing and Predicting Security Vulnerabilities in Software Systems." *Computers & Security* 26 (3): 219-228. http://doi.org/10.1016/j.cose.2006.10.002.

AMD. "Software Techniques for Managing Speculation on AMD Processors Whitepaper.", last modified Jan 24, accessed Jan 25, 2018, https://www.amd.com/en/corporate/speculative-execution.

Anderson, R. 2001. "Why Information Security is Hard-an Economic Perspective." New Orleans, LA, USA, IEEE, 10-14 Dec. http://doi.org/10.1109/ACSAC.2001.991552.

Anderson, Ross. 2002. "Security in Open Versus Closed Systems-the Dance of Boltzmann, Coase and Moore." Toulouse, France, June 20-21. https://www.tse-fr.eu/publications/security-open-versus-closed-systems-dance-boltzmann-coase-and-moore.

anonymous. "Once upon a Free.", accessed 02/13, 2015, http://phrack.org/issues/57/9.html#article.

Apple. "About Speculative Execution Vulnerabilities in ARM-Based and Intel CPUs.", last modified Jan 9, accessed Jan 25, 2018, https://support.apple.com/en-us/HT208394.

Arbaugh, W. A., W. L. Fithen, and J. McHugh. 2000. "Windows of Vulnerability: A Case Study Analysis." *Computer* 33 (12): 52-59. http://dx.doi.org/10.1109/2.889093.

ARM. "Vulnerability of Speculative Processors to Cache Timing Side-Channel Mechanism.", last modified Jan 19, accessed Jan 25, 2018, https://developer.arm.com/support/security-update.

Aslam, Taimur, Ivan Krsul, and Eugene H. Spafford. 1996. "Use of a Taxonomy of Security Faults." Baltimore, MD, USA, NIST, October 22-25. http://csrc.nist.gov/nissc/1996/papers/NISSC96/paper057/PAPER.PDF.

Baker, David W., Steven M. Christey, William H. Hill, and David E. Mann. 1999. "The Development of a Common Enumeration of Vulnerabilities and Exposures." West Lafayette, Indiana, USA, September 7-9. https://cve.mitre.org/docs/docs-2001/Development_of_CVE.html.

Ball, T. 1999. "The Concept of Dynamic Analysis." Toulouse, France, September 6-10. http://doi.org/10.1145/318774.318944.

Bambauer, D. E. and O. Day. 2010. "The Hacker's Aegis." *Emory Law Journal* 60: 1051-1107. http://dx.doi.org/10.2139/ssrn.1561845.

Bishop, Matt. 1995. *A Taxonomy of UNIX System and Network Vulnerabilities*. Davis, California, USA: Department of Computer Science, University of California at Davis. http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/ucd-ecs-95-10.pdf.

blexim. "Basic Integer Overflows.", accessed 02/13, 2015, http://phrack.org/issues/60/10.html#article.

Bohrnstedt, George W. and Arthur S. Goldberger. 1969. "On the Exact Covariance of Products of Random Variables." *Journal of the American Statistical Association* 64 (328): 1439-1442. http://dx.doi.org/10.2307/2286081.

Bourque, P. and R. Fairley, eds. 2014. *Guide to the Software Engineering Body of Knowledge*. 3rd ed. Piscataway, NJ: IEEE Computer Society. http://www.swebok.org.

Box, G. E. and G. C. Tiao. 1992. *Bayesian Inference in Statistical Analysis*. Reading, MA, USA: John Wiley & Sons. http://dx.doi.org/10.1002/9781118033197.

Bradley, Ralph Allan and Milton E. Terry. 1952. "Rank Analysis of Incomplete Block Designs: I. the Method of Paired Comparisons." *Biometrika* 39 (3): 324-345. http://dx.doi.org/10.2307/2334029.

Brady, Robert M., Ross J. Anderson, and Robin C. Ball. 1999. *Murphy's Law, the Fitness of Evolving Species, and the Limits of Software Reliability*. Cambridge, United Kingdom: University of Cambridge, Computer Laboratory. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-471.pdf.

Brooks, W. and R. Motley. 1980. *Analysis of Discrete Software Reliability Models*. Griffiss Air Force Base, New York: Rome Air Development Center. http://www.dtic.mil/dtic/tr/fulltext/u2/a086334.pdf.

Burgin, T. A. 1975. "The Gamma Distribution and Inventory Control." *Operational Research Quarterly* 26 (3): 507-525. http://dx.doi.org/10.2307/3008211.

Carpenter, B., A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, and A. Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1). http://dx.doi.org/10.18637/jss.v076.i01.

Cavusoglu, H., H. Cavusoglu, and S. Raghunathan. 2007. "Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge." *Software Engineering, IEEE Transactions On* 33: 171-185. http://dx.doi.org/10.1109/TSE.2007.26.

Ceccato, M., Miliano Di Penta, P. Falcarin, F. Ricca, M. Torchiano, and P. Tonella. 2014. "A Family of Experiments to Assess the Effectiveness and Efficiency of Source Code Obfuscation Techniques." *Empirical Software Engineering* 19 (4): 1040-1074. http://dx.doi.org/10.1007/s10664-013-9248-x.

Ceccato, M., M. D. Preda, J. Nagra, C. Collberg, and P. Tonella. 2007. "Barrier Slicing for Remote Software Trusting." IEEE, Sept. 30-Oct. 1. http://dx.doi.org/10.1109/SCAM.2007.27.

Chen, M. H. and Q. M. Shao. 1999. "Monte Carlo Estimation of Bayesian Credible and HPD Intervals." *Journal of Computational and Graphical Statistics* 8 (1): 69-92. http://dx.doi.org/10.2307/1390921.

Chen, Ming-Hui, Qi-Man Shao, and Joseph G. Ibrahim. 2000. "Computing Bayesian Credible and HPD Intervals." In *Monte Carlo Methods in Bayesian Computation*, 213-235: Springer. http://dx.doi.org/10.1007/978-1-4612-1276-8.

Cinlar, E. 1975. *Introduction to Stochastic Processes*. Englewood Cliffs, NJ, USA: Dover Publications, Inc.

Clements, Michael and David Hendry. 1993. "On the Limitations of Comparing Mean Square Forecast Errors." *Journal of Forecasting* 12 (8): 617-637. http://dx.doi.org/10.1002/for.3980120802.

Collberg, C. S. and C. Thomborson. 2002. "Watermarking, Tamper-Proofing, and Obfuscation: Tools for Software Protection." *IEEE Transactions on Software Engineering* 28 (8): 735-746. http://dx.doi.org/10.1109/TSE.2002.1027797.

Cooke, R. M. 1991. *Experts in Uncertainty: Opinion and Subjective Probability in Science*. New York, NY, USA: Oxford University Press.

Cooke, R. M. and Goossens, L L H J. 2008. "TU Delft Expert Judgment Data Base." *Reliability Engineering & System Safety; Special Issue: Expert Judgment* 93 (5): 657-674. http://dx.doi.org/10.1016/j.ress.2007.03.005.

Cooke, Roger M. 2015. *Personal Correspondence with Roger Cooke.*

———. 2008. "Response to Discussants." *Reliability Engineering & System Safety; Special Issue: Expert Judgment* 93 (5): 775-777. http://dx.doi.org/10.1016/j.ress.2008.02.006.

Cox, David R. 1972. "The Statistical Analysis of Dependencies in Point Processes." In *Stochastic Point Processes: Statistical Analysis, Theory, and Applications*, edited by Peter A. W. Lewis, 55-66. New York, NY, USA: Wiley-Interscience.

Cusumano, M. and D. Yoffie. 1999. "Software Development on Internet Time." *Computer* 32 (10): 60-69. http://dx.doi.org/10.1109/2.796110.

Daswani, Neal, Christoph Kern, and Anita Kesavan. 2007. *Foundations of Security: What Every Programmer Needs to Know* Apress. http://dx.doi.org/10.1007/978-1-4302-0377-3.

DeGroot, M. H. 2005. *Optimal Statistical Decisions*. Vol. 82 Wiley-Interscience. http://dx.doi.org/10.1002/0471729000.

Doksum, Kjell. 1974. "Tailfree and Neutral Random Probabilities and their Posterior Distributions." *The Annals of Probability* 2 (2): 183-201. http://www.jstor.org/stable/2959219.

Draper, N. R. and H. Smith. 2014. *Applied Regression Analysis*. 3rd ed. New York, NY, USA: Wiley. http://dx.doi.org/10.1002/9781118625590.

Dreyfus, S. E. 2004. "The Five-Stage Model of Adult Skill Acquisition." *Bulletin of Science, Technology & Society* 24 (3): 177-181. http://dx.doi.org/10.1177/0270467604264992.

Duane, Simon, Anthony D. Kennedy, Brian J. Pendleton, and Duncan Roweth. 1987. "Hybrid Monte Carlo." *Physics Letters B* 195 (2): 216-222. http://doi.org/10.1016/0370-2693(87)91197-X.

E. J. Chikofsky and J. H. Cross. 1990. "Reverse Engineering and Design Recovery: A Taxonomy." *IEEE Software* 7 (1): 13-17. http://dx.doi.org/10.1109/52.43044.

Eilam, E. 2011. *Reversing: Secrets of Reverse Engineering*. Indianapolis, IN: Wiley Publishing, Inc.

Electronic Industries Association. 1997. *Product Life Cycle Data Model*. Arlington, VA: Electronic Industries Association.

Erlingsson, Úlfar, Yves Younan, and Frank Piessens. 2010. "Low-Level Software Security by Example." In *Handbook of Information and Communication Security*, edited by Peter Stavroulakis and Mark Stamp, 633-658: Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-04117-4.

F. Massacci and V. H. Nguyen. 2014. "An Empirical Methodology to Evaluate Vulnerability Discovery Models." *IEEE Transactions on Software Engineering* 40 (12): 1147-1162. http://dx.doi.org/10.1109/TSE.2014.2354037.

Fragoso, Tiago M., Wesley Bertoli, and Francisco Louzada. 2017. "Bayesian Model Averaging: A Systematic Review and Conceptual Classification." *International Statistical Review*: n/a. http://dx.doi.org/10.1111/insr.12243.

Friel, N. and A. N. Pettitt. 2008. "Marginal Likelihood Estimation Via Power Posteriors." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70 (3): 589-607. http://dx.doi.org/10.1111/j.1467-9868.2007.00650.x.

Friel, Nial, Merrilee Hurn, and Jason Wyse. 2014. "Improving Power Posterior Estimation of Statistical Evidence." *Statistics and Computing* 24 (5): 709-723. http://dx.doi.org/10.1007/s11222-013-9397-1.

Gelman, Andrew and Xiao-Li Meng. 1998. "Simulating Normalizing Constants: From Importance Sampling to Bridge Sampling to Path Sampling." *Statistical Science* 13 (2): 163-185. http://dx.doi.org/10.1214/ss/1028905934.

Geman, S. and D. Geman. 1984. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6 (6): 721-741. http://dx.doi.org/10.1109/tpami.1984.4767596.

Gibson, D. L., D. R. Goldenson, and K. Kost. 2006. *Performance Results of CMMI-Based Process Improvement*. Pittsburgh, PA, USA: Carnegie Mellon University, Software Engineering Institute. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=8065.

Gilks, W. R. 1992. "Derivative-Free Adaptive Rejection Sampling for Gibbs Sampling." In *Bayesian Statistics 4: Proceedings of the Fourth Valencia International Meeting, April 15-20, 1991*, edited by J. M. Bernardo, J. O. Berger and M. H. DeGroot, 641-649: Clarendon Press.

———. 1996. "Full Conditional Distributions." In *Markov Chain Monte Carlo in Practice*, edited by W. R. Gilks, S. Richardson and D. Spiegelhalter, 75-88: Chapman & Hall, CRC Press.

Gilks, W. R., N. G. Best, and K. K. C. Tan. 1995. "Adaptive Rejection Metropolis Sampling within Gibbs Sampling." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 44: 455-472. http://dx.doi.org/10.2307/2986138.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter. 1996. "Introducing Markov Chain Monte Carlo." In *Markov Chain Monte Carlo in Practice*, edited by W. R. Gilks, S. Richardson and D. Spiegelhalter, 1-20: Chapman & Hall, CRC Press.

Gilks, W. R. and P. Wild. 1992. "Adaptive Rejection Sampling for Gibbs Sampling." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 41 (2): 337-348. http://dx.doi.org/10.2307/2347565.

Givens, G. H. and J. A. Hoeting. 2012. *Computational Statistics*. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc.

Goel, A. L. 1985. "Software Reliability Models: Assumptions, Limitations, and Applicability." *IEEE Transactions on Software Engineering* SE-11 (12): 1411-1423. http://dx.doi.org/10.1109/tse.1985.232177.

Goel, A. L. and K. Okumoto. 1979. "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures." *IEEE Transactions on Reliability* R-28 (3): 206-211. http://dx.doi.org/10.1109/tr.1979.5220566.

Goel, A. L. and K. Z. Yang. 1997. "Software Reliability and Readiness Assessment Based on the Non-Homogeneous Poisson Process." In *Emphasizing Parallel Programming Techniques*, edited by Marvin V. Zelkowitz. Vol. 45, 197-267. San Diego, CA, USA: Academic Press. http://dx.doi.org/10.1016/S0065-2458(08)60709-3.

Gokhale, S. S. and K. S. Trivedi. 1998. "Log-Logistic Software Reliability Growth Model." November 13-14. http://dx.doi.org/10.1109/HASE.1998.731593.

Goldstein, Emmanuel. 2009. *The Best of 2600, Collector's Edition: A Hacker Odyssey*. Indianapolis, IN, USA: Wiley Publishing.

Goodman, Leo A. 1960. "On the Exact Variance of Products." *Journal of the American Statistical Association* 55 (292): 708-713. http://dx.doi.org/10.2307/2281592.

Google. "Google's Mitigations Against CPU Speculative Execution Attack Methods.", last modified Jan 18, accessed Jan 25, 2018, https://support.google.com/faqs/answer/7622138.

Hastie, Trevor. 1987. "A Closer Look at the Deviance." *The American Statistician* 41 (1): 16-20. http://dx.doi.org/10.2307/2684312.

Hastings, W. K. 1970. "Monte Carlo Sampling Methods using Markov Chains and their Applications." *Biometrika* 57 (1): 97-109. http://dx.doi.org/10.2307/2334940.

Henry, S. and D. Kafura. 1981. "Software Structure Metrics Based on Information Flow." *IEEE Transactions on Software Engineering* SE-7 (5): 510-518. http://dx.doi.org/10.1109/tse.1981.231113.

Hirata, Takumi, Hiroyuki Okamura, and Tadashi Dohi. 2009. "A Bayesian Inference Tool for NHPP-Based Software Reliability Assessment." Springer, Dec. 10-12. http://dx.doi.org/10.1007/978-3-642-10509-8_26.

Hiroyuki, Okamura, Ando Mitsuaki, and Dohi Tadashi. 2007. "A Generalized Gamma Software Reliability Model." *Systems and Computers in Japan* 38 (2): 81-90. http://dx.doi.org/10.1002/scj.20350.

Hoeting, J. A., D. Madigan, A. E. Raftery, and C. T. Volinsky. 1999. "Bayesian Model Averaging: A Tutorial." *Statist. Sci.* 14 (4): 382-417. http://dx.doi.org/10.1214/ss/1009212519.

Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer. http://dx.doi.org/10.1007/978-0-387-92407-6.

Hoffman, M. D. and A. Gelman. 2014. "The no-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.* 15 (1): 1593-1623. http://jmlr.org/papers/v15/hoffman14a.html.

Howard, M. 2009. "Improving Software Security by Eliminating the CWE Top 25 Vulnerabilities." *Security & Privacy, IEEE* 7: 68-71. http://dx.doi.org/10.1109/msp.2009.69.

Howard, M. and S. Lipner. 2006. *The Security Development Lifecycle*. Redmond, WA, USA: Microsoft Press.

Huang, Chen-Yu, Sy-Yen Kuo, and Michael R. Lyu. 2007. "An Assessment of Testing-Effort Dependent Software Reliability Growth Models." *IEEE Transactions on Reliability* 56 (2): 198-211. http://dx.doi.org/10.1109/tr.2007.895301.

IBM. 2013. *IBM X-Force 2013 Mid-Year Trend and Risk Report*. Somers, NY: IBM Security Systems.

———. . 2017. *IBM X-Force Threat Intelligence Index 2017*. Somers, NY: IBM Security Systems.

Intel. "Speculative Execution and Indirect Branch Prediction Side Channel Analysis Method.", last modified Jan 22, accessed Jan 25, 2018, https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00088&languageid=en-fr.

International Organization for Standardization. 2009. *Information Technology -- Security Techniques -- Information Security Management Systems -- Overview and Vocabulary*: International Organization for Standardization.

———. . 2014. *Information Technology-Security Techniques-Vulnerability Disclosure*. Switzerland: International Organization for Standardization.

———. . 2013. *Information Technology-Security Techniques-Vulnerability Handling Processes*. Switzerland: International Organization for Standardization.

Jeffreys, H. 1961. *Theory of Probability*. 3rd ed. New York, NY, USA: Oxford University Press.

Joh, HyunChul, Jinyoo Kim, and Y. K. Malaiya. 2008. "Vulnerability Discovery Modeling using Weibull Distribution." Seattle, WA, USA, IEEE, Nov. 10-14. http://dx.doi.org/10.1109/ISSRE.2008.32.

Joh, HyunChul and Yashwant K. Malaiya. 2014. "Modeling Skewness in Vulnerability Discovery." *Quality and Reliability Engineering International* 30 (8): 1445-1459. http://dx.doi.org/10.1002/qre.1567.

Johnson, N., S. Kotz, and N. Balakrishnan. 1994. *Continuous Univariate Distributions*. 2nd ed. Vol. 1 John Wiley & Sons, Inc.

Kass, R. E. and A. E. Raftery. 1995. "Bayes Factors." *Journal of the American Statistical Association* 90 (430): 773-795. http://dx.doi.org/10.1080/01621459.1995.10476572.

Kimura, M. 2006. "Software Vulnerability: Definition, Modelling, and Practical Evaluation for E-Mail Transfer Software." *International Journal of Pressure Vessels and Piping* 83 (4): 256-261. http://doi.org/10.1016/j.ijpvp.2006.02.003.

———. 2003. "A Study on Software Vulnerability Assessment Modeling and its Application to E-Mail Distribution Software System." *The Journal of Reliability Engineering Association of Japan* 25 (3): 279-287. http://ci.nii.ac.jp/naid/110003994427/en/.

Krsul, I. V. 1998. "Software Vulnerability Analysis." PhD., Purdue University.

Kullback, Solomon, Subir Ghosh, Kenneth P. Burnham, Nico F. Laubscher, Gerard E. Dallal, Leland Wilkinson, Donald F. Morrison, et al. 1987. "Letters to the Editor." *The American Statistician* 41 (4): 338-341. http://www.jstor.org/stable/2684769.

Kuo, L. and S. K. Ghosh. 1997. *Bayesian Nonparametric Inference for Nonhomogeneous Poisson Processes*. Storrs, CT, USA: Department of Statistics, University of Connecticut.

Larman, C. 2004. *Agile and Iterative Development: A Manager's Guide*, edited by Alistair Cockburn, Jim Highsmith. 1Ed ed. Boston, MA: Pearson Education, Inc.

Larson, Ron and Bruce Edwards. 2014. *Calculus*. 10th ed. Boston, MA, USA: Brooks/Cole, Cengage Learning.

Lartillot, Nicolas and Hervé Philippe. 2004. "A Bayesian Mixture Model for Across-Site Heterogeneities in the Amino-Acid Replacement Process." *Molecular Biology and Evolution* 21 (6): 1095-1109. http://dx.doi.org/10.1093/molbev/msh112.

———. 2006. "Computing Bayes Factors using Thermodynamic Integration." *Systematic Biology* 55 (2): 195-207. http://dx.doi.org/10.1080/10635150500433722.

Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis*. 3rd ed. New York, NY, USA: McGraw-Hill.

Levenberg, K. 1944. "A Method for the Solution of Certain Nonlinear Problems in Least Squares." *Quarterly of Applied Mathematics* 2: 164-168. http://dx.doi.org/10.1090/qam/10666.

Lipner, S. 2016. *Personal Conversations with Steve Lipner.*

———. 2004. "The Trustworthy Computing Security Development Lifecycle." http://dx.doi.org/10.1109/csac.2004.41.

Littlewood, B. and J. L. Verrall. 1973. "A Bayesian Reliability Growth Model for Computer Software." IEEE Computer Society, April 30-May 2. http://dx.doi.org/10.2307/2346781.

Littlewood, Bev. 1979. "How to Measure Software Reliability and how Not To." *IEEE Transactions on Reliability* R-28 (2): 103-110. http://dx.doi.org/10.1109/tr.1979.5220510.

———. 1984. "Rationale for a Modified Duane Model." *IEEE Transactions on Reliability* R-33 (2): 157-159. http://dx.doi.org/10.1109/TR.1984.5221762.

Lunn, D., D. Spiegelhalter, A. Thomas, and N. Best. 2009. "The BUGS Project: Evolution, Critique and Future Directions." *Statistics in Medicine* 28 (25): 3049-3067. http://dx.doi.org/10.1002/sim.3680.

Lunn, D., A. Thomas, N. Best, and D. Spiegelhalter. 2000. "WinBUGS-a Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing* 10 (4): 325-337. https://doi.org/10.1023/A:1008929526011.

M. G. Rekoff. 1985. "On Reverse Engineering." *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15 (2): 244-252. http://dx.doi.org/10.1109/TSMC.1985.6313354.

M. N. Gagnon, S. Taylor, and A. K. Ghosh. 2007. "Software Protection through Anti-Debugging." *IEEE Security & Privacy* 5 (3): 82-84. http://dx.doi.org/10.1109/MSP.2007.71.

Madigan, D. and A. E. Raftery. 1994. "Model Selection and Accounting for Model Uncertainty in Graphical Models using Occam's Window." *Journal of the American Statistical Association* 89 (428): 1535-1546. http://dx.doi.org/10.2307/2291017.

Marquardt, D. 1963. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *Journal of the Society for Industrial and Applied Mathematics* 11 (2): 431-441. http://dx.doi.org/10.1137/0111030.

Marsaglia, George and Wai Wan Tsang. 1984. "A Fast, Easily Implemented Method for Sampling from Decreasing Or Symmetric Unimodal Density Functions." *SIAM Journal on Scientific and Statistical Computing* 5 (2): 349-359. http://dx.doi.org/10.1137/0905026.

———. 2000. "A Simple Method for Generating Gamma Variables." *ACM Transactions on Mathematical Software (TOMS)* 26 (3): 363-372. http://dx.doi.org/10.1145/358407.358414.

Martinez, Wendy L. and Angel R. Martinez. 2008. *Computational Statistics Handbook with MATLAB, Second Edition*. Boca Raton, FL, USA: Chapman & Hall/CRC.

Massacci, Fabio, Stephan Neuhaus, and Viet Hung Nguyen. 2011. "After-Life Vulnerabilities: A Study on Firefox Evolution, its Vulnerabilities, and Fixes.". http://dx.doi.org/10.1007/978-3-642-19125-1_15.

Matsumoto, Makoto and Takuji Nishimura. 1998. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8 (1): 3-30. http://dx.doi.org/10.1145/272991.272995.

McCabe, T. J. 1976. "A Complexity Measure." *IEEE Transactions on Software Engineering* 2 (4): 308-320. http://dx.doi.org/10.1109/TSE.1976.233837.

McLaughlin, Michael P. 2016. *A Compendium of Common Probability Distributions* Michael P. McLaughlin. http://causascientia.org/math_stat/Dists/Compendium.pdf.

Merrick, J. R. and R. Soyer. 2007. *Semiparametric Bayesian Decision Models for Optimal Replacement*. Washington D.C., USA: The Institute for Integrating Statistics in Decision Sciences, George Washington University.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. "Equation of State Calculations by Fast Computing Machines." *The Journal of Chemical Physics* 21 (6): 1087-1092. http://dx.doi.org/10.1063/1.1699114.

Meunier, Pascal. 2008. "Classes of Vulnerabilities and Attacks." Chap. 4, In *Wiley Handbook of Science and Technology for Homeland Security*, edited by John G. Voeller. Vol. 2, 1-18: John Wiley & Sons, Inc. http://dx.doi.org/10.1002/9780470087923.hhs421.

Microsoft. "Protect Your Windows Devices Against Spectre and Meltdown.", last modified Jan 23, accessed Jan 25, 2018, https://support.microsoft.com/en-us/help/4073757/protect-your-windows-devices-against-spectre-meltdown.

Moranda, P. B. 1975. "Prediction of Software Reliability during Debugging." Washington D.C., USA, IEEE, January 28-30.

More, Jorge J. 1977. "The Levenberg-Marquardt Algorithm: Implementation and Theory." Dundee, Springer Berlin Heidelberg, June 28--July 1. http://dx.doi.org/10.1007/BFb0067700.

Musa, J. D. 1975. "A Theory of Software Reliability and its Application." *Software Engineering, IEEE Transactions On* SE-1: 312-327. http://doi.org/10.1109/TSE.1975.6312856.

Musa, J. and K. Okumoto. 1984. "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement." Orlando, FL, USA, IEEE Press, March 26-29.

National Institute of Standards and Technology. "National Vulnerability Database.", accessed 01/23/, 2018, https://nvd.nist.gov/.

Neal, R. M. 2011. "MCMC using Hamiltonian Dynamics." Chap. 5, In *Handbook of Markov Chain Monte Carlo*, edited by Steve Brooks, Andrew Gelman, Galin Jones and Xiao-Li Meng. Vol. 2, 113-174. New York, NY: CRC Press.

———. 2003. "Slice Sampling." *The Annals of Statistics* 31 (3): 705-741. http://dx.doi.org/10.1214/aos/1056562461.

Nelder, J. A. and R. Mead. 1965. "A Simplex Method for Function Minimization." *The Computer Journal* 7 (4): 308-313. http://dx.doi.org/10.1093/comjnl/7.4.308.

Nelsen, D. and S. E. Daniels. 2007. "Quality Glossary." *Quality Progress* 40 (6): 39-59. https://search.proquest.com/docview/214767618.

Nesterov, Yurii. 2009. "Primal-Dual Subgradient Methods for Convex Problems." *Mathematical Programming* 120 (1): 221-259. http://dx.doi.org/10.1007/s10107-007-0149-x.

Neumann, Peter G. 1985. "Welcome to Risks@sri-Csl." *The Risks Digest*, 1.

Nguyen, Viet Hung and Fabio Massacci. 2013. "The (Un)Reliability of NVD Vulnerable Versions Data." ACM, . http://dx.doi.org/10.1145/2484313.2484377.

———. 2012. "An Independent Validation of Vulnerability Discovery Models." Seoul, Korea, ACM, . http://dx.doi.org/10.1145/2414456.2414459.

Ntzoufras, I. 2009. *Bayesian Modeling using WinBUGS*. Hoboken, NJ, USA: Wiley.

Ogata, Yosihiko. 1989. "A Monte Carlo Method for High Dimensional Integration." *Numerische Mathematik* 55 (2): 137-157. http://dx.doi.org/10.1007/BF01406511.

Ohba, Mitsuru. 1984. "Inflection S-Shaped Software Reliability Growth Model." Nagoya, Japan, Springer Berlin Heidelberg, April 23–24. http://dx.doi.org/10.1007/978-3-642-45587-2_10.

Ohishi, Koji, Hiroyuki Okamura, and Tadashi Dohi. 2009. *Gompertz Software Reliability Model: Estimation Algorithm and Empirical Validation*. Vol. 82. http://dx.doi.org/10.1016/j.jss.2008.11.840.

Okamura, H. and T. Dohi. 2008. "Hyper-Erlang Software Reliability Model.". http://dx.doi.org/10.1109/PRDC.2008.20.

Okamura, H., T. Dohi, and S. Osaki. 2004. "EM Algorithms for Logistic Software Reliability Models." Innsbruck, Austria, ACTA Press, 17-19 February.

Okamura, H., M. Tokuzane, and T. Dohi. 2013. "Quantitative Security Evaluation for Software System from Vulnerability Database." *Journal of Software Engineering and Applications; Special Issue: Software Dependability* 6 (4A): 15-23. http://dx.doi.org/10.4236/jsea.2013.64A003.

Okamura, Hiroyuki, Tadashi Dohi, and Shunji Osaki. 2013. *Software Reliability Growth Models with Normal Failure Time Distributions*. Vol. 116. http://dx.doi.org/10.1016/j.ress.2012.02.002.

Ozment, A. 2007. "Improving Vulnerability Discovery Models." Alexandria, VA, USA, ACM, October 29. http://dx.doi.org/10.1145/1314257.1314261.

———. 2006. "Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models." In *Advances in Information Security: Security Measurements and Metrics*, edited by Dieter Gollmann, Fabio Massacci and Artsiom Yautsiukhin. Vol. 23, 25-36. New York, NY, USA: Springer.

Parker, D. B. 1997. "The Strategic Values of Information Security in Business." *Computers & Security* 16 (7): 572-582. http://dx.doi.org/10.1016/S0167-4048(97)80793-6.

Petter, S., W. DeLone, and E. R. McLean. 2013. "Information Systems Success: The Quest for the Independent Variables." *Journal of Management Information Systems* 29 (4): 7-62. http://dx.doi.org/10.2753/MIS0742-1222290401.

Pettitt, Anthony N. and Candice M. Hincksman. 2012. "Introduction to MCMC." In *Case Studies in Bayesian Statistical Modelling and Analysis*, edited by Clair L. Alston, Kerrie L. Mengersen and Anthony N. Pettitt, 17-29. West Sussex, U.K.: John Wiley & Sons.

Plummer, M. 2003. "JAGS: A Program for Analysis of Bayesian Graphical Models using Gibbs Sampling." Vienna, Austria, Technische Universit at Wien, Mar 20-22. https://www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf.

Pulkkinen, U. 1994. "Bayesian Analysis of Consistent Paired Comparisons." 43: 1-16. http://dx.doi.org/10.1016/0951-8320(94)90091-4.

Rahimi, S. and M. Zargham. 2013. "Vulnerability Scrying Method for Software Vulnerability Discovery Prediction without a Vulnerability Database." *IEEE Transactions on Reliability* 62 (2): 395-407. http://dx.doi.org/10.1109/TR.2013.2257052.

rain.forest.puppy. "NT Web Technology Vulnerabilities.", accessed 02/13, 2015, http://phrack.org/issues/54/8.html#article.

Rescorla, E. 2005. "Is Finding Security Holes a Good Idea?" *IEEE Security & Privacy* 3 (1): 14-19. http://dx.doi.org/10.1109/msp.2005.17.

Robert, C. P. 2010. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. 2nd ed. New York, NY, USA: Springer-Verlag.

Robert, C. P. and G. Casella. 2004. *Monte Carlo Statistical Methods*. 2nd ed. New York, NY, USA: Springer-Verlag.

Roberts, G. O. 1996. "Markov Chain Concepts Related to Sampling Algorithms." In *Markov Chain Monte Carlo in Practice*, edited by W. R. Gilks, S. Richardson and D. Spiegelhalter, 45-58: Chapman & Hall, CRC Press.

Ross, Sheldon. 2013. *Simulation*. Boston: Academic Press.

Ross, Sheldon M. 2007. *Introduction to Probability Models*. 9th ed. Boston: Academic Press.

Roumani, Y., J. K. Nwankpa, and Y. F. Roumani. 2015. "Time Series Modeling of Vulnerabilities." *Computers & Security* 51: 32-40. http://dx.doi.org/10.1016/j.cose.2015.03.003.

Royce, Winston W. 1970. "Managing the Development of Large Software Systems." Los Angeles, CA, IEEE, August 25-28.

Ruohonen, J., S. Hyrynsalmi, and V. Leppänen. 2015. "The Sigmoidal Growth of Operating System Security Vulnerabilities: An Empirical Revisit." *Computers & Security* 55: 1-20. http://dx.doi.org/10.1016/j.cose.2015.07.001.

Ryan, J., T. A. Mazzuchi, D. J. Ryan, de la Cruz, J L, and R. M. Cooke. 2012. "Quantifying Information Security Risks using Expert Judgment Elicitation." *Computers & Operations Research; Special Issue: Operational Research in Risk Management* 39 (4): 774-784. http://dx.doi.org/10.1016/j.cor.2010.11.013.

Samaniego, F. J. 2010. *A Comparison of the Bayesian and Frequentist Approaches to Estimation*. 1st ed. New York, NY, USA: Springer-Verlag.

Samuelson, P. and S. Scotchmer. 2002. "The Law and Economics of Reverse Engineering." *The Yale Law Journal* 111 (7): 1575-1663. http://dx.doi.org/10.2307/797533.

Sarishvili, Alex and Gerrit Hanselmann. 2013. "Software Reliability Prediction Via Two Different Implementations of Bayesian Model Averaging." Prague, 23–27 September. http://www.ecmlpkdd2013.org/wp-content/uploads/2013/04/copem2013_Sharishvili.pdf.

Schick, George and Ray Wolverton. 1978. *An Analysis of Competing Software Reliability Models*. Vol. SE-4.

scut/team teso. "Exploiting Format String Vulnerabilities." https://crypto.stanford.edu/cs155/papers/formatstring-1.2.pdf.

SecurityFocus. "BugTraq Charter.", accessed 01/23, 2015, http://www.securityfocus.com/archive/1/description.

Shepperd, M. 1990. "Design Metrics: An Empirical Analysis." *Software Engineering Journal* 5 (1): 3-10. http://dx.doi.org/10.1049/sej.1990.0002.

Shooman, Martin L. 1976. "Structural Models for Software Reliability Prediction." San Francisco, California, USA, IEEE Computer Society Press, . http://dl.acm.org/citation.cfm?id=800253.807687.

Software Engineering Institute. 1997. *C4 Software Technology Reference Guide—A Prototype*. Pittsburgh, PA, USA: Carnegie Mellon University.

solar designer. "Return into Libc.", accessed 02/13, 2015, http://seclists.org/bugtraq/1997/Aug/63.

Solomon, R., P. A. Sandborn, and M. G. Pecht. 2000. "Electronic Part Life Cycle Concepts and Obsolescence Forecasting." *Components and Packaging Technologies, IEEE Transactions On* 23: 707-717. http://dx.doi.org/10.1109/6144.888857.

Soyer, Refik and M. M. Tarimcilar. 2008. "Modeling and Analysis of Call Center Arrival Data: A Bayesian Approach." *Management Science* 54 (2): 266-278. http://dx.doi.org/10.1287/mnsc.1070.0776.

Stankosky, M. 2002. "The System Dynamics of Knowledge Management." Palermo, Italy, July 28 - August 1.

Szwed, P., J. R. V. Dorp, J. R. W. Merrick, T. A. Mazzuchi, and A. Singh. 2006. "A Bayesian Paired Comparison Approach for Relative Accident Probability Assessment with Covariate Information." *European Journal of Operational Research* 169 (1): 157-177. http://dx.doi.org/10.1016/j.ejor.2004.04.047.

The MITRE Corporation. "Common Vulnerabilities and Exposures, the Standard for Information Security Vulnerability Names.", last modified Jan 19, accessed Jan 25, 2018, https://cve.mitre.org/.

———. "Common Weakness Enumeration.", last modified January 31, accessed February 23, 2018, https://cwe.mitre.org/.

V Nagaraju, L Fiondella, and T Wandji. 2017. "An Open-Source Tool to Support the Quantitative Assessment of Cyber Security for Software Intensive System Acquisition." *Journal of Information Warfare* 16 (3): 31-50.

van Eeten, M. J. and J. M. Bauer. 2008. "Economics of Malware: Security Decisions, Incentives and Externalities." *OECD Science, Technology and Industry Working Papers* 2008/01: 1-69. http://dx.doi.org/10.1787/241440230621.

van Noortwijk, J. M. 2009. "A Survey of the Application of Gamma Processes in Maintenance." *Reliability Engineering & System Safety* 94 (1): 2-21. http://dx.doi.org/10.1016/j.ress.2007.03.019.

Weiser, M. 1982. "Programmers use Slices when Debugging." *Commun. ACM* 25 (7): 446-452. http://dx.doi.org/10.1145/358557.358577.

Williams, L., M. Gegick, and M. Vouk. 2008. *Predictive Models for Identifying Software Components Prone to Failure during Security Attacks*. Pittsburgh, PA, USA: Software Engineering Institute, Carnegie Mellon University.

Winsor, C. P. 1932. "The Gompertz Curve as a Growth Curve." *Proceedings of the National Academy of Sciences of the United States of America* 18 (1): 1-8. http://www.jstor.org/stable/86156.

Woo, S. W., H. Joh, O. H. Alhazmi, and Y. K. Malaiya. 2011. "Modeling Vulnerability Discovery Process in Apache and IIS HTTP Servers." *Computers & Security* 30: 50-62. http://doi.org/10.1016/j.cose.2010.10.007.

Yamada, S., M. Ohba, and S. Osaki. 1983. "S-Shaped Reliability Growth Modeling for Software Error Detection." *IEEE Transactions on Reliability* R-32 (5): 475-484. http://dx.doi.org/10.1109/tr.1983.5221735.

Yamada, Shigeru. 1992. "A Stochastic Software Reliability Growth Model with Gompertz Curve ." *Transactions of Information Processing Society of Japan* 33 (7): 964-969.

Yamada, Shigeru and Takaji Fujiwari. 2001. "Testing-Domain Dependent Software Reliability Growth Models and their Comparisons of Goodness-of-Fit." *International Journal of Reliability, Quality and Safety Engineering* 08 (03): 205-218. http://doi.org/10.1142/S0218539301000475.

Yamada, Shigeru and S. Osaki. 1985. "Software Reliability Growth Modeling: Models and Applications." *Software Engineering, IEEE Transactions On* SE-11: 1431-1437. http://dx.doi.org/10.1109/tse.1985.232179.

Zbigniew A. Kotulski and Wojciech Szczepinski. 2010. *Error Analysis with Applications in Engineering*. Solid Mechanics and its Applications, Volume 169. http://doi.org/10.1007/978-90-481-3570-7.

Appendix A. Probability distributions

This section provides basic details for five types of random variables: $Exponential$; $Gamma$; $Normal$; $Uniform$; piecewise linear; and χ^2.

A.1 Exponential

The probability density function (pdf) for a variable x distributed according to the $Exponential$ distribution, $f_X(x) \sim Exponential(a)$ is $f_X(x) = a \cdot e^{-a \cdot x}$, where a is the rate hyper-parameter, $a > 0$ and $x \geq 0$ (McLaughlin 2016). This pdf is proportional to $f_X(x) \propto e^{-a \cdot x}$. The cumulative distribution function (cdf) is $F_X(x) = 1 - e^{-a \cdot x}$. For the $Exponential$ distribution, $E[X] = \frac{1}{a}$ is the random variable's expected value and $SD[X] = (a)^{-\frac{1}{2}}$ is the corresponding standard deviation (McLaughlin 2016). The continuous form of the inverse transform method generates r.v.s from the $Exponential$, where $F^{-1}(X) = -\frac{1}{a} \cdot \ln(1 - U)$ (and where U is a random sample from $Uniform(0,1)$), [85] (see MATLAB's *exprnd* function).

A.2 Gamma

The pdf for a variable x distributed according to the $Gamma$ distribution, $f_X(x) \sim Gamma(a, b)$ is $f_X(x) = \frac{b^a x^{a-1} e^{-bx}}{\Gamma(a)}$, where a is the shape hyper-parameter, b is the rate hyper-parameter, $\Gamma(a) = \int_0^\infty u^{a-1} e^{-u} du$ is the Gamma function, $a > 0$, and $b > 0$, and $x > 0$ (van Noortwijk 2009, 2-21). This pdf is proportional to $f_X(x) \propto x^{a-1} e^{-bx}$. The cdf is $F_X(x) = \frac{1}{\Gamma(a)} \cdot \gamma(\mathrm{a}, \mathrm{b} \cdot \mathrm{x})$, where the lower incomplete gamma function is $\gamma(\mathrm{a}, \mathrm{b} \cdot \mathrm{x}) = \int_0^{b \cdot x} t^{a-1} \cdot e^{-t} \cdot dt$. The mean for a variable x distributed per the $Gamma$ distribution is $E[X] = \frac{a}{b}$, while the corresponding standard deviation is $SD[X] = \sqrt{\frac{a}{b^2}}$ (Burgin 1975, 507-525). The Marsaglia and Tsang (Marsaglia and Tsang 2000, 363-372) technique generates r.v.s from the $Gamma$ distribution (see MATLAB's *gamrnd* function).

A.3 Normal

The pdf for a variable x distributed according to the $Normal$ distribution, $f_X(x) \sim Normal\left(\mu, \left(\frac{1}{r}\right)^{-1}\right)$ is $f_X(x) = \left(\frac{r}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2} r (x-\mu)^2}$, where μ is the mean, the precision r is the inverse of the variance (σ^2), $-\infty < \mu < \infty$, $\frac{1}{r} > 0$, and $-\infty < x < \infty$ (Hoff 2009). This pdf is proportional to $f_X(x) \propto r^{\frac{1}{2}} e^{-\frac{1}{2} r (x-\mu)^2}$. For the $Normal$ distribution, $E[X] = \mu$ is the random variable's expected value and $SD[X] = (r)^{-\frac{1}{2}}$ is the corresponding standard deviation (Hoff

[85] Compute $F_X^{-1}(u)$: $u = 1 - e^{-a \cdot x}$, so $x = -\frac{1}{a} \cdot \ln(1 - u)$.

2009). The Ziggurat method of Marsaglia and Tsang (Marsaglia and Tsang 1984, 349-359) generates r.v.s from the $Normal$ distribution (see MATLAB's *normrnd* function).

A.4 Uniform

The pdf for a variable x distributed according to the $Uniform$ distribution, $f_X(x) \sim Uniform(a,b)$ is $f_X(x) = \frac{1}{b-a}$, where a is the lower boundary, b is the upper boundary, $-\infty < a < b < \infty$, and $x \in [a,b]$ (McLaughlin 2016). For the $Uniform$ distribution, $E[X] = \frac{1}{2}(a+b)$ is the random variable's expected value and $SD[X] = \left(\frac{1}{12}(b-a)^2\right)^{\frac{1}{2}}$ is the corresponding standard deviation (McLaughlin 2016). The "Mersenne Twister" technique described by Matsumoto and Nishimura (Matsumoto and Nishimura 1998, 3-30) generates r.v.s from the Uniform (see MATLAB's *rand* function).

A.5 Piecewise linear

Define a piecewise linear function

$$g(x) = \begin{cases} a_j \cdot x + b_j \; for \; x \in [l_j, r_j), j = 1,2,\cdots,N \\ 0 \qquad\qquad otherwise \end{cases}, \tag{A.1}$$

where $l_1 < r_1 \leq l_2 < r_2 \leq l_3 < \cdots < r_{N-1} \leq l_N < r_N$ (Zbigniew A. Kotulski and Wojciech Szczepinski 2010). Then the normalizing constant used to transform $g(x)$ into a pdf $f_X(x)$ is

$$C = \int_{-\infty}^{\infty} g(x) \cdot dx = \sum_{j=1}^{N} \frac{1}{2} a_j \left(r_j^2 - l_j^2\right) + \sum_{j=1}^{N} b_j \left(r_j - l_j\right), \tag{A.2}$$

(Zbigniew A. Kotulski and Wojciech Szczepinski 2010). For the piecewise linear distribution,

$$E[X] = m = \frac{1}{3C} \sum_{j=1}^{N} a_j \left(r_j^3 - l_j^3\right) + \frac{1}{2C} \sum_{j=1}^{N} b_j \left(r_j^2 - l_j^2\right), \tag{A.3}$$

$$E[X^2] = \frac{1}{4C} \sum_{j=1}^{N} a_j \left(r_j^4 - l_j^4\right) + \frac{1}{3C} \sum_{j=1}^{N} b_j \left(r_j^3 - l_j^3\right), \tag{A.4}$$

and $SD[X] = (E[X^2] - m^2)^{\frac{1}{2}}$ (Zbigniew A. Kotulski and Wojciech Szczepinski 2010). A *PiecewiseLinear* distribution object can be created in MATLAB and r.v.s can be generated using it (see MATLAB's *makedist* and *random* functions).

A.6 Chi-squared

The pdf for a variable x distributed according to the χ^2 distribution, $f_X(x) \sim \chi^2(\nu)$ is $f_X(x) = \frac{1}{2^{\nu/2} \cdot \Gamma(\nu/2)} \cdot e^{-\frac{x}{2}} \cdot x^{(\nu/2)-1}$, where ν is the degrees of freedom, $\nu \in \mathbb{N}_{>0}$, and $x \geq 0$ (Johnson, Kotz, and Balakrishnan 1994). For the χ^2 distribution, $E[X] = \nu$ is the random variable's expected value and $SD[X] = (2 \cdot \nu)^{\frac{1}{2}}$ is the corresponding standard deviation (Johnson, Kotz, and Balakrishnan 1994). As $\chi^2(\nu) = Gamma(\nu/2, 1/2)$, random number generation techniques are noted in Appendix A.2.

Appendix B. Arrival processes

This section provides an overview of arrival processes and two sub-types used in this research. These are the nonhomogeneous Poisson process (NHPP) and the ***GAMMA*** process.

B.1 Arrival process overview

The stochastic process $N = \{N_t(\omega); t \geq 0\}$, defined on the sample space Ω, is called an arrival process when for any realization $\omega \in \Omega$ the mapping $t \rightarrow N_t(\omega)$ is: non-decreasing; increases by jumps only; is right continuous; and $N_0(\omega) = 0$ (Cinlar 1975). In these stochastic processes, $N_t(\omega)$ represents the number of arrivals in $[0, t]$ for the realization ω (Cinlar 1975). An example arrival process depicting five random event arrivals is illustrated in Figure B-1 (Cinlar 1975).

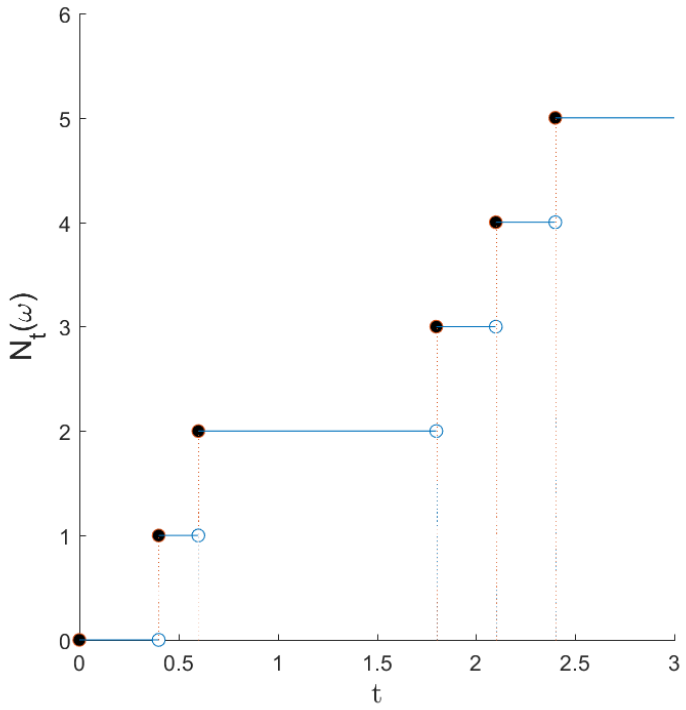


Figure B-1: Arrival process example

B.2 Nonhomogeneous Poisson process overview

Arrival processes are Poisson processes when the axioms in Table B-1 are upheld[86] and the X_1, X_2, $\cdots$ interarrivals are independent and identically distributed (i.i.d.) $Exponential$ r.v.s (see Appendix A.1)[87]. The related nonhomogeneous Poisson process (NHPP) form additionally assumes that the expected number of arrivals $E[N_t]$, is described by some mean value function, or cumulative intensity function $\Lambda(t)$, that changes over time. The rate of occurrence of arrivals at time t, $\lambda(t) = \frac{d}{dt}\Lambda(t)$, also known as the intensity function, is the derivative of $\Lambda(t)$. For these functions to uphold axiom A, the following mathematical relationships are necessary (Cinlar 1975):

$$\lim_{\Delta t \to 0} \frac{Pr(\Lambda(t + \Delta t) - \Delta(t) = 1)}{\Delta t} = \lambda(t) \qquad (B.1)$$

and

$$\lim_{\Delta t \to 0} \frac{Pr(\Lambda(t + \Delta t) - \Delta(t) \geq 2)}{\Delta t} = 0\,. \qquad (B.2)$$

Table B-1: NHPP axioms

Axiom	Name	Description
A	Single arrivals	"for almost all ω, each jump of $t \to N_t(\omega)$ is of unit magnitude" (Cinlar 1975)
B	Independent increments	"for any interval $t, s \geq 0$, the random variable $N_{t+s} - N_t$ is independent of the past history $\{N_u; u \leq t\}$ until t" (Cinlar 1975)

Several key properties are associated with the NHPP. Assume that $\Lambda(t)$ can be represented by a differentiable parametric function $H(t; \boldsymbol{\theta})$, where $t \geq 0$ and $\boldsymbol{\theta}$ is a vector of unknown parameters. The corresponding intensity function is denoted $h(t; \boldsymbol{\theta})$. Given some $H(t; \boldsymbol{\theta})$, for the time interval $[0, t]$, the mean or expected number of occurrences in $[0, t]$ denoted $H(t; \theta)$ is given by

$$H(t; \theta) = E[N(t)|h(t; \boldsymbol{\theta})] = \int_0^t h(u|\boldsymbol{\theta})du\,, \qquad (B.3)$$

(Ryan et al. 2012, 774-784). If $N(t)$ is a NHPP, then the probability mass function (pmf) for the number of occurrences in $[0, t]$ is given by

$$Pr\big(N(t) = n\big|H(t; \boldsymbol{\theta})\big) = \left[\frac{\big(H(t; \boldsymbol{\theta})\big)^n}{n!}\right] e^{-H(t;\boldsymbol{\theta})}, \qquad (B.4)$$

[86] A special case of the NHPP, the homogeneous Poisson process (HPP), upholds Axioms A, B, and "has a third 'stationarity' axiom, for any $t, s \geq 0$, the distribution of $N_{t+s} - N_t$ is independent of t" (Cinlar 1975).

[87] Ross (S. Ross 2013) notes that for any given integer $n \geq 1$ and time $t > 0$, the n^{th} arrival epoch, $S_n = \sum_{i=1}^{n} X_i$, and the $N(t)$ cumulative counts over time for the Poisson process are related by $Pr(S_n \leq t) = Pr(N(t) \geq n)$ and complementing both sides results in $Pr(S_n > t) = Pr(N(t) < n)$.

(Ryan et al. 2012, 774-784). Similarly, and given some $h(t;\boldsymbol{\theta})$, for the time interval $[s,t]$, with $s<t$, the mean or expected number of occurrences in $[s,t]$ is $H(t;\boldsymbol{\theta})-H(s;\boldsymbol{\theta})$ (Ryan et al. 2012, 774-784). The pmf for the number of occurrences in $[s,t]$ is given by

$$Pr\big(N(t)-N(s)=n\big|H(t;\boldsymbol{\theta})\big)=\left[\frac{\big(H(t;\boldsymbol{\theta})-H(s;\boldsymbol{\theta})\big)^{n}}{n!}\right]e^{-[H(t;\boldsymbol{\theta})-H(s;\boldsymbol{\theta})]}, \qquad (B.5)$$

(Ryan et al. 2012, 774-784). Given $h(t;\theta)$, $s<t$, and using the independent increments axiom (i.e., $N(s)$ and $N(t)-N(s)$ are independent random variables), the joint pmf for $N(t)=n, N(s)=k$ is then

$$Pr\big(N(t)=n,N(s)=k\big|H(t;\theta)\big)=\Pr\big(N(s)=k|H(t;\theta)\big)\cdot Pr\big(N(t)-N(s)=n-k|H(t;\theta)\big), \qquad (B.6)$$

(Ryan et al. 2012, 774-784). Additionally, the discrete form of the inverse transform method (see Appendix E.1) may be used to generate grouped interval samples from the NHPP.

B.3 Nonstationary GAMMA process overview

Arrival processes are considered ***GAMMA*** processes when the axioms in Table B-1 are upheld and the $X_1, X_2, \cdots$, interarrivals are i.i.d. $Gamma$ r.v.s (see Appendix A.2) (van Noortwijk 2009, 2-21). Relatedly, a nonstationary ***GAMMA*** process changes the distribution for the $Gamma$ r.v.s over time. To set up its description, assume that the time interval $[0,\infty)$ is partitioned into a subset of $\mathcal{I}$ intervals, $[t_0,t_1)$, $[t_0,t_1)$, $\cdots$, $[t_{\mathcal{I}-1},t_{\mathcal{I}}=\infty)$. Next, let these intervals be described by the non-decreasing mean value function, $\Lambda(t)$, where $\Lambda(0)=0$ and define $r_i=\Lambda(t_i)-\Lambda(t_{i-1})$. This indicates

$$\Lambda(t_i)=\sum_{i=1}^{\mathcal{I}} r_j\,, \qquad (B.7)$$

for $i=1,\cdots,\mathcal{I}$ (Merrick and Soyer 2007). Doksum (Doksum 1974, 183-201) shows that a set of positively increasing functions, $\{\Lambda(t)\}$, can have a $\mathcal{I}$-dimensional probability distribution specified on its space, using $Gamma$ distributions for $r_1,\cdots,r_{\mathcal{I}}$ that correspond with the time intervals above. So, in the i^{th} interval, r_i is distributed as

$$r_i\sim Gamma\big(c\cdot\big(\Lambda(t_i)-\Lambda(t_{i-1})\big),c\big), \qquad (B.8)$$

where c is a constant and $E[r_i]=\Lambda(t_i)-\Lambda(t_{i-1})$.

Unfortunately, the pmf for the ***GAMMA*** process is not analytically tractable; however, it can be estimated numerically with interval data (see Appendix E.2). Two supporting relationships now follow. They both stem from the arrival process property that describes the time range between the n^{th} and $(n+1)^{\text{th}}$ arrival epochs in the i^{th} interval where the interval count is n, or

$$r_i=n \text{ for } S_{n,i}\le t<S_{n+1,i}. \qquad (B.9)$$

First, a general relationship between the n^{th} epoch in the i^{th} interval, $S_{n,i}$ and the corresponding r_i interval count over time is

$$Pr\big(S_{n,i}\le t\big)=Pr(r_i\ge n)\,. \qquad (B.10)$$

Then, its complement provides a second relationship that is

$$Pr\left(S_{n,i} > t\right) = Pr(r_i < n)\,. \qquad (B.11)$$

Appendix C. Scoring in Cooke's method

Calculation of calibration and information measures requires practitioners to first estimate the response distribution extremes for all questions. The procedure differs only slightly per question type (i.e., calibration or phenomena of interest). First, one determines the maximum upper and lower bounds, q_U and q_L respectively, for the seed questions.

$$q_L = min\{\psi_i, q_5(1), \dots, q_5(e)\}, \tag{C.1}$$

$$q_U = max\{\psi_i, q_{95}(1), \dots, q_{95}(E)\}, \tag{C.2}$$

(Cooke and Goossens, L L H J 2008, 657-674). Specifically, ψ_i in Equations $(C.1)$-$(C.2)$ denotes the known answers to the i^{th} seed question, the $q_{\%}(e)$ represent the corresponding quantile percentage edges for the expert e; and E specifies the number of experts (Cooke and Goossens, L L H J 2008, 657-674). Second, one selects a $k\%$ intrinsic range for q_L and q_U (i.e., $q_L - k\%$ and $q_U + k\%$ respectively) and then uses Equations $(C.3)$-$(C.4)$ to generate $q_0(e,i)$ and $q_{100}(e,i)$ point estimates for all expert response distributions.

$$q_0(e,i) = q_5(e,i) - \frac{k}{100}(q_U - q_L), \tag{C.3}$$

and

$$q_{100}(e,i) = q_{95}(e,i) + \tfrac{k}{100}(q_U - q_L), \tag{C.4}$$

(Cooke and Goossens, L L H J 2008, 657-674). The procedure for the questions dealing with the phenomena of interest relates to the above steps. In this case, q_L and q_U respectively use a modified form of Equations $(C.1)$-$(C.2)$ with removed ψ_i. Likewise, for q_0 and q_{100}, use Equations $(C.3)$-$(C.4)$ with all i terms removed, as the difference is only in notation.

Having complete expert distributions for all seed questions enables practitioner computation of the expert calibration scores. For every expert, the practitioners measure their seed answer location relative frequencies, s_{1-4}, within bin_{1-4}. The e^{th} expert's responses (i.e., $[q_0(e,i), q_5(e,i), q_{50}(e,i), q_{95}(e,i), q_{100}(e,i)]$) specifies the bin_{1-4} boundaries for the i^{th} calibration question. One constructs s_{1-4} by locating each seed question's known answer ψ within the expert's corresponding defined probability bin_{1-4} for that question, summing the individual expert per bin results from all calibration questions, and dividing each of these by S. To illustrate, in the case of $\mathcal{S} = 12$ seed questions, a well-calibrated expert e scores three assignments per bin, resulting in the histogram $\left\{s_1(e) = \frac{3}{12}, s_2(e) = \frac{3}{12}, s_3(e) = \frac{3}{12}, s_4(e) = \frac{3}{12}\right\}$. The next step generates the relative information measure and the following equation derives this for expert e

$$I(s(e)|p) = \sum_{j=1}^{4} s_j(e) \cdot \ln\left(\frac{s_j(e)}{p_j}\right), \tag{C.5}$$

(Cooke and Goossens, L L H J 2008, 657-674). Typical p_{1-4} interval sizes for bin_{1-4} are respectively 0.05, 0.45, 0.45, and 0.05 (Cooke and Goossens, L L H J 2008, 657-674). Let $\chi^2_{(d)}$ denote the cdf for a chi square variable with d degrees of freedom (see Appendix A.6). According to Cooke and Goossens (Cooke and Goossens, L L H J 2008, 657-674), for reasonably large $\mathcal{S}$ values, $2\mathcal{S} \cdot I(s(e), p) \sim \chi^2_{(3)}$; therefore, the cdf of $\chi^2_{(3)}$ estimates $Pr\{\chi^2_{(3)} > 2\mathcal{S} \cdot I(s(e), p)\}$. The last step computes the calibration score for the expert e using

$$C(e) = \begin{cases} 1 - \chi^2_{(3)}[2\mathcal{S} \cdot I(s(e)|p)], \text{if } 1 - \chi^2_{(3)}[2\mathcal{S} \cdot I(s(e)|p)] > \text{A} \\ 0 \text{ otherwise} \end{cases}, \quad (C.6)$$

(Cooke and Goossens, L L H J 2008, 657-674). The calibration measure inherently provides a threshold for elimination of all expert data not attaining a specified A, [88]. In other words, the methodology eliminates data from those persons performing poorly in the calibration session, providing an "*element of empirical control*" (Cooke 1991). Calibration power augments the non-zero portion of Equation $(C.6)$ as

$$C(e) = 1 - \chi^2_{(3)}[2\mathcal{S} \cdot I(s(e)|p) \cdot \rho], \quad (C.7)$$

where the calibration power, $\rho \in [0.1,1.0]$, [89].

According to Cooke and Goossens (Cooke and Goossens, L L H J 2008, 657-674), the information score effectively measures the distance between the expert's distribution and some meaningful background. For example, one could consider the uniform distribution as a background measure (see Figure C-1) (Ryan et al. 2012, 774-784). The information score is computed for each expert using the following steps. Let $F_U(x)$ denote the cdf for the $Uniform$ distribution (see Appendix A.4). The first step calculates an expert's density for item i, $h_{1-4}(e,i)$, using the expert e's specified distributions for the i^{th} seed variable question and

$$h_1(e,i) = F_U\big(q_5(e,i)\big) - F_U\big(q_0(e,i)\big) = \frac{q_5(e,i) - q_0(e,i)}{q_{100}(e,i) - q_0(e,i)}, \quad (C.8)$$

$$h_2(e,i) = F_U\big(q_{50}(e,i)\big) - F_U\big(q_5(e,i)\big) = \frac{q_{50}(e,i) - q_5(e,i)}{q_{100}(e,i) - q_0(e,i)}, \quad (C.9)$$

$$h_3(e,i) = F_U\big(q_{95}(e,i)\big) - F_U\big(q_{50}(e,i)\big) = \frac{q_{95}(e,i) - q_{50}(e,i)}{q_{100}(e,i) - q_0(e,i)}, \quad (C.10)$$

and

$$h_4(e,i) = F_U\big(q_{100}(e,i)\big) - F_U\big(q_{95}(e,i)\big) = \frac{q_{100}(e,i) - q_{95}(e,i)}{q_{100}(e,i) - q_0(e,i)}, \quad (C.11)$$

[88] A is selected by identifying the optimal calibration threshold when a fictitious expert is added to the pool. The decision-maker (DM) assessments are determined using various combinations of expert assessments (discussed later in this section) and should receive the highest possible weight (Ryan et al. 2012, 774-784).

[89] See the documentation for Excalibur Pro.

for all seed items (Cooke and Goossens, L L H J 2008, 657-674). The second step generates the information score for expert e's answer to the i^{th} seed item using

$$I(e,i) = I(h(e,i)|p) = \sum_{j=1}^{4} p_j \ln\left(\frac{p_j}{h_j(e,i)}\right), \qquad (C.12)$$

for all seed questions (Cooke and Goossens, L L H J 2008, 657-674). The third and final step calculates the e^{th} expert's average information score, $I(e)$, across that expert's entire individual seed item set of scores.

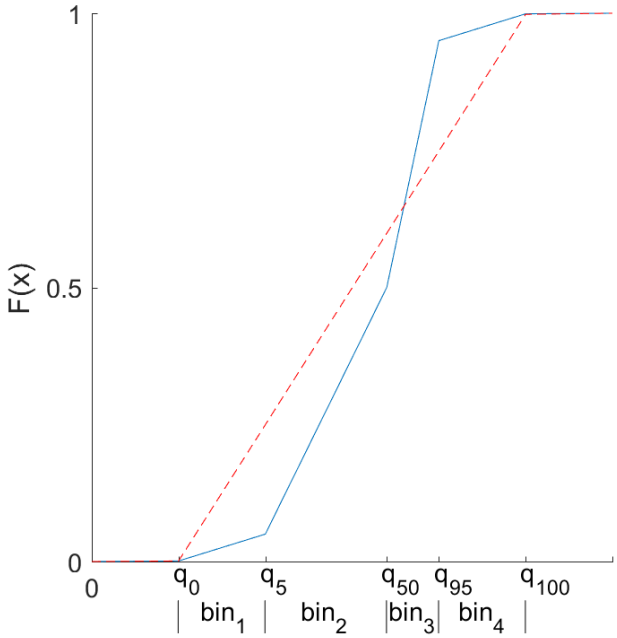


Figure C-1: Comparison of expert distribution with background measure

The dashed red denotes a $Uniform$ background measure and solid blue denotes a notional expert distribution.

Decision-makers in the CCM represent linear pooling and refer to "combinations of expert assessments" (Cooke and Goossens, L L H J 2008, 657-674). The first of the three types is the global weight DM_{A}; this is the "result of linear pooling for seed item i" using weights proportional to Equation (4.3) and defined by

$$DM_{\text{A}}(i) = \sum_{e=1}^{E} \frac{w_{\text{A}}(e) \cdot h(e,i)}{\sum_{e=1}^{E} w_{\text{A}}(e)}, \qquad (C.13)$$

(Cooke and Goossens, L L H J 2008, 657-674). The A* that maximizes $C(DM_{\text{A}}) \cdot I(DM_{\text{A}})$ identifies the value for DM_{A} (Cooke and Goossens, L L H J 2008, 657-674). The global qualifier is present because the practitioner calculates Equation (4.3) using the average $I(e)$ across all seed items (Cooke and Goossens, L L H J 2008, 657-674). The next type is the item weight DM, IDM_{A}; this is a DM_{A} variation that instead uses the individual seed item scores, $I(e,i)$ (Cooke and Goossens, L L H J 2008, 657-674). The item weight IDM_{A} uses weights proportional to Equation (4.4) and is

$$IDM_{\text{A}}(i) = \sum_{e=1}^{E} \frac{w_{\text{A}}(e,i) \cdot h(e,i)}{\sum_{e=1}^{E} w_{\text{A}}(e,i)}. \qquad (C.14)$$

The A^* that maximizes $C(IDM_A) \cdot I(IDM_A)$ identifies the value for IDM_A (Cooke and Goossens, L L H J 2008, 657-674). The last of the three types is the equal weight DM_{eq} and is the combination of participants that results from $w(e)$ and Equation (4.5) (Cooke and Goossens, L L H J 2008, 657-674).

Appendix D. Traditional analysis techniques

This section provides a brief overview of the traditional techniques called maximum likelihood estimation (MLE) and the method of least squares (LS).

D.1 Maximum likelihood estimation (MLE)

MLE techniques numerically determine point estimates for the model variables that maximize the log-likelihood of the data given the model (Givens and Hoeting 2012). Many MLE supporting maximization algorithms are available for locating the optimum parameter values. For this research application, the *direct search* sub-class of maximization algorithms retains an implementation advantage, as they require no information about the derivative from the log-likelihood function (Givens and Hoeting 2012). Nelder and Mead (Nelder and Mead 1965, 308-313) provide one of the more popular direct search algorithms for function maximization[90]. Givens and Hoeting (Givens and Hoeting 2012) present an in-depth description for this algorithm.

D.2 Method of least squares (LS)

LS techniques numerically determine model variable point estimates that minimize the sum of squares deviation from the model predictions (Draper and Smith 2014). For explanatory purposes, assume the case of a parametric model observable $\hat{y} = f(x;\theta)$, with a single independent variable x, and the parameter θ. Additionally, assume that data are available defining a $n \times 1$ dependent variable vector $\boldsymbol{y}$ and corresponding independent variable vector $\boldsymbol{x}$. The i[th] residual $f_i(x_i) = y_i - \hat{y}_i$, is defined as the difference between the i[th] data point y_i and the model prediction for $f(x_i;\theta)$. As the name suggests, the sum of squares deviation equation is then simply the sum of the squared residuals, or $\frac{1}{2}\sum_{i=1}^{n} f_i^2(x_i) = \frac{1}{2}\|F(\boldsymbol{x})\|^2$, [91] (More 1977, 105-116). More (More 1977, 105-116) presents a popular implementation of the algorithm[92] proposed by Levenberg (Levenberg 1944, 164-168) and Marquardt (Marquardt 1963, 431-441).

---

[90] MATLAB's MLE routine utilizes this algorithm by default.

[91] The notation $\|\cdot\|$ represents the LS vector norm, $\|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}$ for the $p \times 1$ vector $\boldsymbol{x}$ (More 1977, 105-116).

[92] MATLAB's *lsqcurvefit* routine optionally uses this algorithm.

Appendix E. Simulation techniques

This section provides an overview for simulation techniques used in the research. These include the inverse transform method (discrete), a method to generate values from Gamma processes, rejection and squeezed rejection sampling, and popular forms of Markov chain Monte Carlo (MCMC).

E.1 Inverse transform method, discrete form

An overview of the inverse transform method's discrete form (Law and Kelton 2000) for the NHPP, where each $[t_{i-1}, t_i)$ interval's pmf (i.e., $Pr(N(t) = n | t_{i-1} < t < t_i)$) becomes increasingly less-likely as $n \to \infty$, is as follows. First, it computes interval pmf estimates for $n = 0,1,\cdots,\mathcal{N}$, where the pmf's mass at each of these n has probability greater than some minimum value (e.g., 1.0×10^{-10}), [93]. Second, it approximates the discrete cdf by performing cumulative sums using the pmf probabilities for each realization n in that interval (i.e., $Pr(N(t) \leq n | t_{i-1} < t < t_i) = F_r(\omega | t_{i-1} < t < t_i)$, where $F_r(\omega | t_{i-1} < t < t_i) \to \sum_{\omega=0}^{n_{max}} Pr(N(t) = \omega | t_{i-1} < t < t_i)$). Third, the method generates a uniform random variable $U \sim Uniform(0,1)$ for each interval. Fourth, the algorithm sets $\hat{\mathrm{N}}(0) = 0$ and then determines each interval's output by locating $\hat{\mathrm{N}}(\mathrm{t}) = max\left(j = n : U \leq F_r(j)\right)$. See Table E-1 for the pseudo-code to use the inverse transform method for sampling from a set of time intervals in a NHPP.

Table E-1: Pseudo-code for simulating grouped-interval data from a NHPP via inverse transform

Inputs:	
n	Set of possible realizations ($\mathrm{n} = [0,1,2,\ldots,\mathcal{N}]$)
t	Vector of times to simulate for (e.g., $\mathrm{t} = [0,10,20,30,40,50]$)
$\hat{\Theta}$	Parameter point estimates
$\Lambda(\mathrm{t};\Theta)$	Cumulative intensity function handle
Outputs:	
$\hat{\mathrm{N}}(\mathrm{t})$	Estimated values from NHPP using $\hat{\Theta}$

Line	**Pseudo-code statement**	
1	$\mathcal{I} = \mathrm{numel}(t) - 1$	
2	for $\mathrm{i} = 1$ to $\mathcal{I}$ {	*//for each interval*
3	for $\mathrm{j} = 1$ to $\mathrm{numel}(\mathrm{n})$ {	*//for every possible $n(j)$*
4	$p_r(i,\mathrm{j}) = \left[\frac{(\Lambda(\mathrm{t(i+1)};\hat{\Theta}) - \Lambda(\mathrm{t(i)};\hat{\Theta}))^{\mathrm{n(j)}}}{\mathrm{n(j)!}}\right] \cdot e^{-[\Lambda(\mathrm{t(i+1)}) - \Lambda(\mathrm{t(i)})]}$	*//compute the interval's pmf (p_r)*
5	$F_r(i,\mathrm{j}) = \sum_{\mathrm{k}=1}^{\mathrm{j}} p_r(\mathrm{j},\mathrm{k})$	*//estimate the intervals cdf*
6	}	
7	}	
8	$\hat{\mathrm{N}}(0) = 0$	//NHPP is zero at t=0

[93] This implies that for values of x greater than some y, $Pr(X = x) \cong 0$.

9	for i = 1 to $\mathcal{I}$ {	*//for each interval $r(i)$*
10	bool_found=false	*//reset the exit condition variable*
11	U~Uniform(0,1)	*//generate a uniform r.v.*
12	j=1	*//reset the location counter*
13	while bool_found==false {	*//loop until exit condition*
14	if $U > F_r(i, \mathrm{j})$	*//test for exit condition*
15	bool_found=true	*//found! Set exit condition*
16	else	*//otherwise*
17	j=j+1	*//not found, increment location*
18	}	
19	$\hat{\mathrm{N}}$(t(i + 1)) = j	*//save the interval's simulated count (location)*
20	}	

E.2 Simulating interval counts in nonstationary Gamma processes

The nonstationary ***GAMMA*** process, $\pi\big(\Theta(t)\big)$~$\boldsymbol{GAMMA}(c \cdot \sum r_i(t)\,, c)$, where $r_i(t) = \Theta(t_i) - \Theta(t_{i-1})$, is a process whose $r_i(t)$ intervals are composed of independent arrivals distributed as Gamma r.v.s (see Appendix B.3) and whose interval pmf's may be approximated. These interval pmf approximations are constructed using their count data $r_i^*(t) = \Lambda^*(t_i) - \Lambda^*(t_{i-1})$, an assumption for c, and a structure where $r_i(t)$ is distributed as $r_i(t)$~$Gamma(c \cdot a_i \cdot r_i^*(t), c)$. Then, one can numerically determine $a_i > 0$ such that $E[r_i(t)] - r_i^*(t)$ is minimized across the possible values in the pmf of $r_i(t)$ (i.e., $E[r_i(t)] = \sum_{j=0}^{\mathcal{N}} n_j \cdot Pr(r_i(t) = j)$). The discrete probabilities $Pr\big(r_i(t) = n_j\big)$ are estimated by using the cdf from $Gamma\big(c \cdot a_i \cdot n_j, c\big)$ and Equations $(B.10)$-$(B.11)$ in:

$$Pr(r_i(t) = 0) = Pr(r_i(t) < 1) = Pr(r_i(t) \geq 0) - Pr(r_i(t) \geq 1) = 1 - Pr\big(S_{1,i} \leq t\big)\,, \qquad (E.1)$$

$$Pr(r_i(t) = 1) = Pr(1 \leq r_i(t) < 2) = Pr(r_i(t) \geq 1) - Pr(r_i(t) \geq 2) = Pr\big(S_{1,i} \leq t\big) - Pr\big(S_{2,i} \leq t\big)\,, \qquad (E.2)$$

and

$$Pr(r_i(t) = n) = Pr(n \leq r_i(t) < n + 1) = Pr(r_i(t) \geq n) - Pr(r_i(t) \geq n + 1) = Pr\big(S_{n,i} \leq t\big) - Pr\big(S_{n+1,i} \leq t\big)\,. \qquad (E.3)$$

Then, with the list of possible values for n and the i^{th} pmf, or $\mathcal{p}_i = \{Pr(r_i(t) = j) \text{ for } j = 0,1,\cdots,\mathcal{N}\}$, for all the intervals, one can indirectly sample from each in JAGS/BUGS as follows. Generate categorical r.v.s for each interval using *dcat*($\mathcal{p}_i$) and convert the r.v.s returned into r_i samples by using them as indices into the list of possible values for n. A final cumulative summation converts the random samples from $r_i(t)$ into $\Theta(t)$ A basic pseudo-code sequence is below in Table E-2.

Table E-2: Pseudo-code for sampling from a ***GAMMA*** process in JAGS/BUGS

Inputs:	
$\mathcal{p}_r$	Set of pmfs for all the intervals in t
n	Set of possible realizations (n = $[0,1,2,\ldots,\mathcal{N}]$)

t	Vector of times to simulate for (e.g., t = [0,10,20,30,40,50])
Outputs: $\hat{\Theta}(t)$	Samples for the intervals in t

Line	**Pseudo-code statement**	
1	$\mathcal{I} = \text{numel}(t) - 1$	
2	$\hat{\Theta}[0] = 0$	// $\boldsymbol{GAMMA}(c \cdot \Theta(0), c) = 0$
3	for i =1 to $\mathcal{I}$ {	// Begin interval loop
4	$rtemp[i] \sim dcat(p_r[i,*])$	// Get the index for the interval
5	$r[i] = n[rtemp[i]]$	// Translate index into count
6	$\hat{\Theta}[t(i+1)] = \hat{\Theta}[t(i)] + r[i]$	// Translate interval count into cumulative sum
7	}	

E.3 Rejection and squeezed rejection sampling

Rejection and squeezed rejection sampling both generate samples from a target distribution, $f(x)$, by transforming those from a sampling distribution, $g(x)$ — that is easy to sample from (e.g., piecewise linear) — via bounded, random rejection of some of its candidates (Givens and Hoeting 2012). The upper bound for accepted samples is called the envelope function and is defined as $e(x) = \frac{g(x)}{v} \geq f(x), \forall\, x \in D$ (where the domain D is an interval of the real line), and $f(x) > 0$ for a given constant $v \leq 1$ (Givens and Hoeting 2012). The lower bound for accepted samples is only used in squeezed rejection sampling, is called the squeezing function, and is defined as $s(x) \leq f(x), \forall\, x \in D$ (Givens and Hoeting 2012). Table E-3 lists the pseudo-code for rejection and squeezed rejection sampling of one candidate value X from $f(x)$ (Givens and Hoeting 2012); for the former, lines 5-7 (squeezing step) would not be performed.

Table E-3: Pseudo-code for *Rejection* and *SqueezedRejection* sampling

Inputs: $f(x)$	Target function handle
$g(x)$	Sampling distribution handle
$e(x)$	Envelope function handle
$s(x)$	Squeezing function (only for squeezed rejection sampling) handle
Outputs: X	Approximated sample from $f(x)$

Line	**Pseudo-code statement**	
1	success=false	
2	while (success==false) {	// Loop until exit condition is set
3	$X \sim g(x)$	// Sample X from $g(x)$
4	$U \sim Uniform(0,1)$	// Sample U
5	if $U \leq \frac{s(X)}{e(X)}$ {	// Test1 for accept or reject (do this only for squeezed rejection sampling)

6	success=true	// Test1 accept condition, set exit condition
7	}	
8	if $U \leq \frac{f(X)}{e(X)}$ then {	// Test2 for accept or reject (do this for rejection and squeezed rejection sampling)
9	success=true	// Test2 accept condition, set exit condition
10	}	
11	}	

E.4 Markov chain Monte Carlo (MCMC)

This section first provides an overview of Markov chain Monte Carlo (MCMC) and then details popular methods for constructing MCMC chains.

E.4.1 MCMC overview

MCMC is a random variate sampling technique that uses specially constructed Markov chains. "Markov chains are sequences of random variables $\{\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \ldots\}$ such that for any step $\ell \geq 0$, the next state $\theta^{(\ell+1)}$ is sampled from a distribution that depends only on the current state of the chain, $\theta^{(\ell)}$" (Gilks, Richardson, and Spiegelhalter 1996, 1-20). Table E-4 illustrates the general MCMC procedure for computing a single variable θ's posterior given data (Ntzoufras 2009), $\pi(\theta|\boldsymbol{D})$, without providing chain construction algorithm specifics (these are covered in Appendix E.4.2). The technique includes preparing a model for sampling, postulating initial values for model parameters, and drawing posterior distribution samples of the parameters; Monte Carlo techniques are used to generalize the sampling results. MCMC-based sampling allows for estimation of moments, quantiles, credible interval (CI) regions, or highest posterior density (HPD) sections[94] (S. M. Ross 2007) from the joint posterior and any of the corresponding marginal distributions (Gilks, Richardson, and Spiegelhalter 1996, 1-20).

Table E-4: General MCMC application for a single parameter

Step	Description
1	Select an initial value for $\theta^{(0)}$
2	Generate values from the posterior $\pi(\theta|\boldsymbol{D})$ while monitoring via convergence diagnostics (listed below in Step 7)
3	Select the step to designate as the "burn-in" threshold (defined in Appendix E.4.3) $\theta^{(B)}$, and remove all samples prior to this threshold (i.e., $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \ldots, \theta^{(B)}$)
4	Samples from the desired posterior distribution are extracted from the $\theta^{(B+1)}, \theta^{(B+2)}, \ldots, \theta^{(B+T)}$
5	Plot the posterior distribution density function and trace plot (raw samples)
6	Generate summary statistics for the posterior distribution (e.g., moments, quantiles, HPDs, etc.)
7	Generate convergence diagnostics (e.g., batch means, acceptance rate, MCMC error)

[94] In general, credible intervals (CIs) and HPDs describe region of values in the posterior distribution having a certain probability of containing θ (Box and Tiao 1992). When the posterior distribution is symmetric (e.g., normal distribution) either CIs or HPDs are appropriate; in the case of asymmetric posteriors (e.g., *Gamma* and *Beta* distributions), HPDs are necessary (M. H. Chen and Shao 1999, 69-92).

Step	Description
8	Analyze results and, if necessary, perform adjustments (e.g., "*tuning*" of the proposal distribution) and repeat the sequence above

The key element behind Markov chain Monte Carlo is the use of a specially constructed Markov chain retaining a stationary distribution equivalent to the desired posterior distribution (Robert and Casella 2004). A Markov chain stationary probability distribution is further described as a "distribution $\pi(\cdot)$ such that if $\theta^{(\ell)} \sim \pi(\cdot)$, then $\theta^{(\ell+1)} \sim \pi(\cdot)$, if the transition kernel allows for free moves all over the state space" (Robert and Casella 2004). The transition kernel, or $K(\cdot)$, describes the density of going from one point in the sequence, $\theta^{(\ell)}$, to another point $\theta^{(\ell+1)}$ (i.e., $\theta^{(\ell+1)} \sim K\left(\theta^{(\ell)}, \theta^{(\ell+1)}\right)$) (Roberts 1996, 45-58).

In Bayesian analyses, it is oftentimes necessary to compute the marginal distributions (i.e., distributions for each single variable that respectively marginalize or integrate out all the other variables). MCMC supports generation of the i^{th} marginal distribution, θ_i, where $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \cdots \theta_m]$, through kernel density estimation using

$$\pi(\theta_i) \approx \frac{1}{T-B}\sum_{\ell=B+1}^{T} K\left(\theta_i \middle| \boldsymbol{\theta}^{(\ell)}\right), \qquad (E.4)$$

where the full conditional distribution is commonly substituted for $K(\theta_i|\boldsymbol{\theta}^{(\ell)})$ (Gilks, Richardson, and Spiegelhalter 1996, 1-20). The full conditional distribution for the i^{th} component of $\boldsymbol{\theta}$ (designated using θ_i), $\pi(\theta_i|\boldsymbol{\theta}_{.-i})$, is the distribution conditioned on all the remaining components of $\boldsymbol{\theta}$ (designated using $\boldsymbol{\theta}_{.-i}$) (e.g., for $i = 2$, $\pi(\theta_2|\boldsymbol{\theta}_{.-2}) = \pi(\theta_2|\theta_1\ \theta_3\ \cdots \theta_m)$).

E.4.2 MCMC chain construction

Metropolis-Hastings (1970, 97-109), Gibbs sampling (Geman and Geman 1984, 721-741), and Hamiltonian Monte Carlo (Duane et al. 1987, 216-222) methods for Markov chain construction all rest on the algorithm originally introduced by Metropolis et al. (1953, 1087-1092). Chain construction for MCMC, an iterative optimization process, specifically seeks a Markov chain producing the desired stationary distribution and will be covered in three general areas. The first, Metropolis-Hastings (1970, 97-109) (Appendix E.4.2.1), includes the Metropolis algorithm and several other ones as special cases of its general form. One of these, the very popular Gibbs sampling (Geman and Geman 1984, 721-741) (Appendix E.4.2.2), is the second general area and it has many derivatives; notable ones include adaptive rejection sampling (ARS) within Gibbs sampling (Gilks and Wild 1992, 337-348; Gilks 1992, 641-649) (Appendix E.4.2.2.1), adaptive rejection Metropolis sampling (ARMS) within Gibbs sampling (Gilks, Best, and Tan 1995, 455-472) (Appendix E.4.2.2.2), and slice-within-Gibbs sampling (Neal 2003, 705-741) (Appendix E.4.2.2.3). The more recent algorithm Hamiltonian Monte Carlo (Duane et al. 1987, 216-222), and its extension, the No-U-Turn (NUTS) sampler, both use concepts from Hamiltonian dynamics (Alder and Wainwright 1959, 459-466) and are covered by the third general area (Appendix E.4.2.3).

E.4.2.1 Metropolis-Hastings (MH) algorithms

The Metropolis-Hastings (MH) algorithm (1970, 97-109) augmented the original Metropolis algorithm (1953, 1087-1092) and is considered to be the general approach for most MCMC methods (Ntzoufras 2009). It approximates samples θ from a target distribution (e.g., $\pi(\theta|\boldsymbol{D})$) by using samples that were generated from a proposal distribution, $q(\theta'|\theta)$ (that depend only on the most recent sample of θ), and samples are accepted according to some relative frequency α (Hoff 2009) (see its pseudo-code sequence in Table E-5 (Ntzoufras 2009)). In the Bayesian context, the latter is

$$\alpha = \min\left(1, \frac{\pi(\theta'|\boldsymbol{D})}{\pi(\theta|\boldsymbol{D})} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}\right), \tag{E.5}$$

and due to cancellation of the marginal likelihood terms in the posterior distributions, this further simplifies to

$$\alpha = \min\left(1, \frac{\mathcal{L}(\theta'|\boldsymbol{D}) \cdot \pi(\theta')}{\mathcal{L}(\theta|\boldsymbol{D}) \cdot \pi(\theta)} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}\right), \tag{E.6}$$

(Hoff 2009). Example proposal distributions include $q(\theta'|\theta) \sim Uniform(\theta - \delta, \theta + \delta)$ and $q(\theta'|\theta) \sim Normal\left(\theta, \left(\frac{1}{\delta^2}\right)^{-1}\right)$, where the parameter δ is chosen for algorithm efficiency (i.e., it can reduce the correlation in the Markov chain but is not large enough to result in too many proposal rejections) (Hoff 2009). The term on the right side of Equation $(E.6)$, $q(\theta|\theta')/q(\theta'|\theta)$, is considered a correction factor, that in some special cases is one (Hoff 2009). These include the Metropolis (in the case of symmetrical proposal distributions, $q(\theta'|\theta) = q(\theta|\theta')$), and random-walk Metropolis (in the case of asymmetrical distributions where $q(\theta'|\theta) = q(|\theta' - \theta|)$) algorithms (Ntzoufras 2009).

Table E-5: Pseudo-code for Metropolis-Hastings sampling

Inputs: $\boldsymbol{\theta}^{(0)}$	Initial value for the chain for each of m elements in the parameter vector
$\pi(\boldsymbol{\theta}|\boldsymbol{D})$	Posterior distribution (i.e., $\pi(\boldsymbol{\theta}|\boldsymbol{D}) \propto \mathcal{L}(\boldsymbol{\theta}|D) \cdot \pi(\boldsymbol{\theta})$) function handle
$q(\boldsymbol{\theta}'|\boldsymbol{\theta})$	Proposal distribution function handle
T	Number of samples to generate
Outputs: $\boldsymbol{\theta}^{(1:T)}$	$1{:}T$ output samples for each of m elements in the parameter vector

Line	**Pseudo-code statement**	
1	for $\ell = 1\ to\ T$ {	
2	$\boldsymbol{\theta} = \boldsymbol{\theta}^{(\ell-1)}$	// Use the previous θ when the candidate is rejected
3	$\acute{\boldsymbol{\theta}} = q\left(\boldsymbol{\theta}^{(\ell)}|\boldsymbol{\theta}\right)$	// Generate candidate value by sampling from proposal distribution
4	$\alpha = \min\left(1, \frac{\mathcal{L}(\boldsymbol{\theta}'|\boldsymbol{D}) \cdot \pi(\boldsymbol{\theta}')}{\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D}) \cdot \pi(\boldsymbol{\theta})} \cdot \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}')}{q(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right)$	// Acceptance probability depends only on the posterior distributions and correction factors
5	$U \sim Uniform(0,1)$	// Generate a uniform r.v.
6	$if\ (U < \alpha)$	
7	$\boldsymbol{\theta} = \acute{\boldsymbol{\theta}}$	// Probability α to accept the candidate

8	$\boldsymbol{\theta}^{(\ell)} = \boldsymbol{\theta}$
9	}

Componentwise MH performs MCMC sampling for multiple elements (e.g., a vector of parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_m]$) by performing the same sampling (described above) individually per element, and using their results to sequentially update each element's respective latest value. In other words, a MH step is performed for an element and the resulting sample becomes the latest value to use for that element—when performing the steps for the remaining ones. Table E-6 contains its pseudo-code sequence (Ntzoufras 2009).

One challenge with MH techniques is its need for tuning proposal distributions to achieve acceptable performance (Ntzoufras 2009). Typically there is a trade between the number of samples necessary to achieve convergence and how many iterations are performed by the algorithm before a sample is accepted (Ntzoufras 2009). For example, consider a case where the proposal distribution is a multivariate normal. Its covariance matrix must be tuned such that it achieves the optimal balance between the number of samples needed for convergence and the proposal acceptance rate per sample (Ntzoufras 2009).

Table E-6: Pseudo-code for single-component MH sampling

Inputs: $\boldsymbol{\theta}^{(0)}$	Initial value for the chain for each of m elements in the parameter vector
$\pi(\boldsymbol{\theta}\|\boldsymbol{D})$	Posterior distribution (i.e., $\pi(\boldsymbol{\theta}\|\boldsymbol{D}) \propto \mathcal{L}(\boldsymbol{\theta}\|D) \cdot \pi(\boldsymbol{\theta})$) function handle
$q(\boldsymbol{\theta}'\|\boldsymbol{\theta})$	Proposal distribution function handle
T	Number of samples to generate
Outputs: $\boldsymbol{\theta}^{(1:T)}$	$1{:}T$ output samples for each of m elements in the parameter vector

Line	**Pseudo-code statement**	
1	for $\ell = 1\ to\ T$ {	
2	$\boldsymbol{\theta} = \left[\theta_1^{(\ell-1)}, \theta_2^{(\ell-1)}, \cdots, \theta_m^{(\ell-1)}\right]$	// Use the previous θ_j when the candidate is rejected
3	for $j = 1\ to\ m$ {	// Perform the componentwise updates sequentially
4	$\acute{\theta}_j \sim q\left(\theta_j^{(\ell)} \middle\| \boldsymbol{\theta}\right)$	// Generate candidate value by sampling from proposal distribution
5	$\alpha = \min\left(1, \frac{\mathcal{L}(\theta_j', \boldsymbol{\theta}_{-j}\|\boldsymbol{D})\cdot\pi(\theta_j', \boldsymbol{\theta}_{-j})}{\mathcal{L}(\theta_j, \boldsymbol{\theta}_{-j}\|\boldsymbol{D})\cdot\pi(\theta_j, \boldsymbol{\theta}_{-j})} \cdot \frac{q(\theta_j\|\theta_j', \boldsymbol{\theta}_{-j})}{q(\theta_j'\|\theta_j, \boldsymbol{\theta}_{-j})}\right)$	// Acceptance probability depends only on the posterior distributions and correction factors
6	$U \sim Uniform(0,1)$	// Generate a uniform r.v.
7	$if\ (U < \alpha)$	
8	$\theta_j = \acute{\theta}_j$	// Probability α to accept the candidate
9	}	
10	$\boldsymbol{\theta}^{(\ell)} = \boldsymbol{\theta}$	// Update the set of samples
11	}	

E.4.2.2 Gibbs sampling algorithms

Gibbs sampling is a special case of componentwise MH that uses the target's full-conditional posteriors as its proposal distributions (Ntzoufras 2009). For the standard Gibbs sampling algorithm, this results in every proposed move being accepted for all elements (because $\alpha = 1$) (Ntzoufras 2009). Standard Gibbs sampling[95] is valid only when the full-conditional distributions comprise common distributions (e.g., *Normal*, *Gamma*, *Uniform*, and *Exponential*) (Ntzoufras 2009). Table E-7 contains its pseudo-code (Ntzoufras 2009) to generate T samples from a target posterior distribution, $\pi(\theta_1, \theta_2, \cdots, \theta_m|\boldsymbol{D})$.

When compared against MH techniques, standard Gibbs sampling algorithms have advantages and disadvantages. On one hand, standard Gibbs sampling easily proposes updates for one variable at a time (i.e., implementation is easier). On the other hand, standard Gibbs sampling cannot independently update variables that are related.

Table E-7: Pseudo-code for standard Gibbs sampling

Inputs: $\boldsymbol{\theta}^{(0)}$	Initial value for the chain for each of m elements in the parameter vector
$\pi(\theta_j\|\boldsymbol{\theta}_{-j}, \boldsymbol{D})$	Full-conditional distributions from target posterior function handles
T	Number of samples to generate
Outputs: $\boldsymbol{\theta}^{(1:T)}$	$1{:}T$ output samples for each of m elements in the parameter vector

Line	**Pseudo-code statement**	
1	for $\ell = 1$ *to* T {	
2	$\boldsymbol{\theta} = [\theta_1^{(\ell-1)}, \theta_2^{(\ell-1)}, \cdots, \theta_m^{(\ell-1)}]$	// Use the previous θ_j before its update step
3	for $j = 1$ *to* m {	// Perform the componentwise updates sequentially
4	$\acute{\theta}_j \sim \pi(\theta_j^{(\ell)}\|\boldsymbol{\theta}_{-j}, \boldsymbol{D})$	// Generate the candidate value by sampling from full-conditionals
5	$\theta_j = \acute{\theta}_j$	// Always accept the candidate, update the jth component of $\boldsymbol{\theta}$ to use in remaining steps
6	}	
7	$\boldsymbol{\theta}^{(\ell)} = \boldsymbol{\theta}$	// Update the set of samples
8	}	

E.4.2.2.1 Adaptive rejection sampling (ARS) within Gibbs

Adaptive rejection sampling (ARS) within Gibbs is used when the full-conditional posterior distributions do not have direct sampling techniques and are log-concave (Gilks and Wild 1992, 337-348; Gilks 1992, 641-649). It uses an augmented form of squeezed rejection sampling (see Appendix E.3), that iteratively reduces the number of function evaluations required per sample by "improving the sampling density function $g(x)$ after each rejection (Gilks, Best, and

[95] Standard Gibbs sampling does not require tuning (Neal 2011, 113-174).

Tan 1995, 455-472)." This is accomplished by utilizing information gleaned about $f(x)$ from all its previous evaluations (Gilks, Best, and Tan 1995, 455-472).

Derivative-free ARS provides a methodology for defining envelope and squeezing functions by projecting chords through previous evaluations of $ln\big(f(x)\big)$ (where $f(x)$ is $\pi\big(\theta_j\big|\boldsymbol{\theta}_{-j}, \boldsymbol{D}\big)$) and using them with squeezed rejection sampling; an overview of its algorithm now follows. Let the current set of ascendingly ordered abscissa be $S_n = \{x_i; i = 0, \dots, n+1\}$, where x_i denotes the i^{th} abscissa (Gilks, Best, and Tan 1995, 455-472) and let the matching set of function evaluations be $y_n = \{ln(f(x_i)); i = 0, \dots, n+1\}$. For $0 \le i \le j \le n$, denote the straight lines through adjacent points $[x_i, ln(f(x_i))]$ and $[x_j, ln(f(x_j))]$ as $L_{ij}(x; S_n)$; these, along with vertical lines at x_0 and x_n, define the squeezing-boundary piecewise linear function[96] (Gilks 1992, 641-649) that is

$$\hbar_n(x; S_n) = L_{i,i+1}(x; S_n), \tag{E.7}$$

for $x_i \le x < x_{i+1}$. Then, the envelope-boundary piecewise linear function is constructed by further projecting these lines until they intersect, and using their uppermost segments across the range of x (Gilks 1992, 641-649). This piecewise linear function is

$$h_n(x; S_n) = \min\big[L_{i-1,i}(x; S_n), L_{i+1,i+2}(x; S_n)\big], \tag{E.8}$$

for $x_i \le x < x_{i+1}$, [97] (Gilks, Best, and Tan 1995, 455-472). Converting these log-domain piecewise linear functions back to base 10 using

$$g_n(x; S_n) = \frac{1}{m_n} e^{h_n(x; S_n)}, \tag{E.9}$$

and

$$\mathcal{g}_n(x; S_n) = \frac{1}{m_n} e^{\hbar_n(x; S_n)}, \tag{E.10}$$

where

$$m_n = \int e^{h_n(x; S_n)} \cdot dx, \tag{E.11}$$

(Gilks, Best, and Tan 1995, 455-472), yields piecewise exponential equivalents (Gilks 1992, 641-649).

Implementation of derivative-free ARS within Gibbs is then described as a consolidation of the squeezed rejection sampling (see Appendix E.3) with ordinary Gibbs sampling (see Appendix E.4.2.2). The pseudo-code to generate T samples from a target posterior distribution, $\pi(\theta_1, \theta_2, \cdots, \theta_m|\boldsymbol{D})$, using ARS within Gibbs is outlined in Table E-8 (Gilks, Best, and Tan 1995, 455-472).

---

[96] A piecewise linear r.v. is described in Appendix A.5.

[97] If b is undefined, $\min(a, b) = \min(b, a) = a$ (Gilks, Best, and Tan 1995, 455-472).

Table E-8: Pseudo-code for ARS within Gibbs sampling

Inputs: $\boldsymbol{\theta}^{(0)}$	Initial value for the chain for each of m elements in the parameter vector
$[S_{n,1}^{(0)}, S_{n,2}^{(0)}, \cdots, S_{n,m}^{(0)}]$	Initial sets of points for each θ_j with at least one abscissa on each side of the mode of $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$
$[y_{n,1}^{(0)}, y_{n,2}^{(0)}, \cdots, y_{n,m}^{(0)}]$	Paired sets of $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$ evaluations at points in $[S_{n,1}^{(0)}, S_{n,2}^{(0)}, \cdots, S_{n,m}^{(0)}]$
$\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$	Full-conditional distributions from target posterior function handle
T	Number of samples to generate
Outputs: $\boldsymbol{\theta}^{(1:T)}$	$1{:}\,T$ output samples for each of m elements in the parameter vector

Line	Pseudo-code statement	
1	for $\ell = 1\ to\ T$ {	// MCMC sampling interactions
2	$\boldsymbol{\theta} = [\theta_1^{(\ell-1)}, \theta_2^{(\ell-1)}, \cdots, \theta_m^{(\ell-1)}]$	// Use the previous θ_j before its update step
3	for $j = 1\ to\ m$ {	// Perform the componentwise updates sequentially
4	$\acute{\theta}_j \sim$ SqueezedRejection($\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$, $g_{n,j}(\theta_j; S_{n,j}, y_{n,j})$, $\exp\left(h_{n,j}(x; S_{n,j}, y_{n,j})\right)$, $\exp\left(\hbar_{n,j}(x; S_{n,j}, y_{n,j})\right)$)	// *SqueezedRejection* function (Table E-3) generates the candidate values $\acute{\theta}_j$ using squeezed rejection sampling to approximate $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$; additionally pass in previous realizations of $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$ using $y_{n,j}$that are used by $g_{n,j}(\cdot)$, $h_{n,j}(\cdot)$,$\hbar_{n,j}(\cdot)$ and to save computing time (respectively, these are function handles for sampling, envelope, and squeezing functions that use S_n and corresponding y_n)
5	$S_{n,j} = \{S_{n,j}, \acute{\theta}_j\}$	// Update our set of abscissae
6	$y_{n,j} = \{y_{n,j}, \pi(\acute{\theta}_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})\}$	// Update our set of function evaluations
7	$\theta_j = \acute{\theta}_j$	// In Gibbs, always accept the candidate, update the j[th] component of $\boldsymbol{\theta}$ to use in remaining steps
8	}	
9	$\boldsymbol{\theta}^{(\ell)} = \boldsymbol{\theta}$	// Update the set of samples
10	}	

To help with comprehension, Figure E-1, Figure E-2, Figure E-3, and Figure E-4 illustrate sequential construction of the piecewise functions, where $\pi(\theta|\boldsymbol{D}) \sim Normal(5,1)$. In Figure E-1, piece-wise linear envelope boundary (blue line) and squeezing boundary (red line) are constructed around log density, $ln(f(\theta))$ (green line), by extending chords through initial points P1-P2 and P2-P3, [98] (Gilks 1992, 641-649). Figure E-2 similarly illustrates the exponentiated form of Figure E-1 (Gilks 1992, 641-649). Figure E-3 expands Figure E-1 by adding a new point P4 and

[98] At the extreme points (where $\theta = 4.3$ and $\theta = 5.8$), vertical lines are constructed between $y = f(\theta)$ and the upper boundary line .

additional chords that improve the envelope and squeezing boundaries (Gilks 1992, 641-649). Figure E-4 likewise shows the exponentiated form of Figure E-4 (Gilks 1992, 641-649).

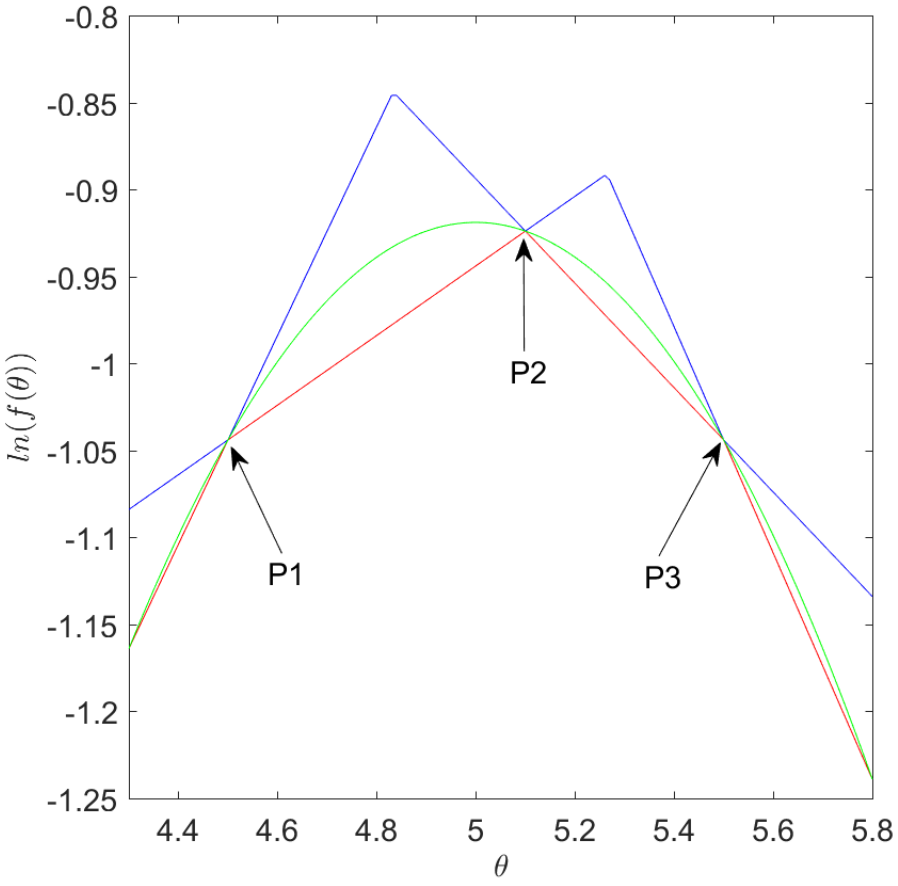


Figure E-1: (Log-scale) ARS envelope and squeezing around initial points (P1, P2, and P3)

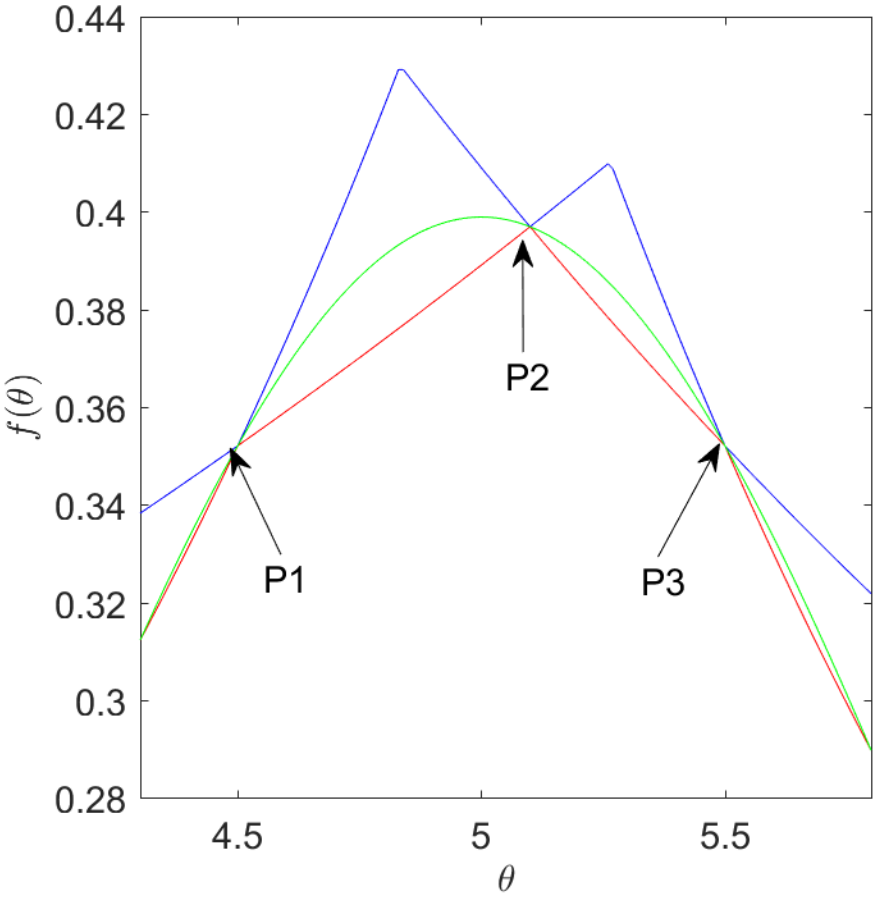


Figure E-2: (Natural-scale) Exponentiated form of Figure E-1

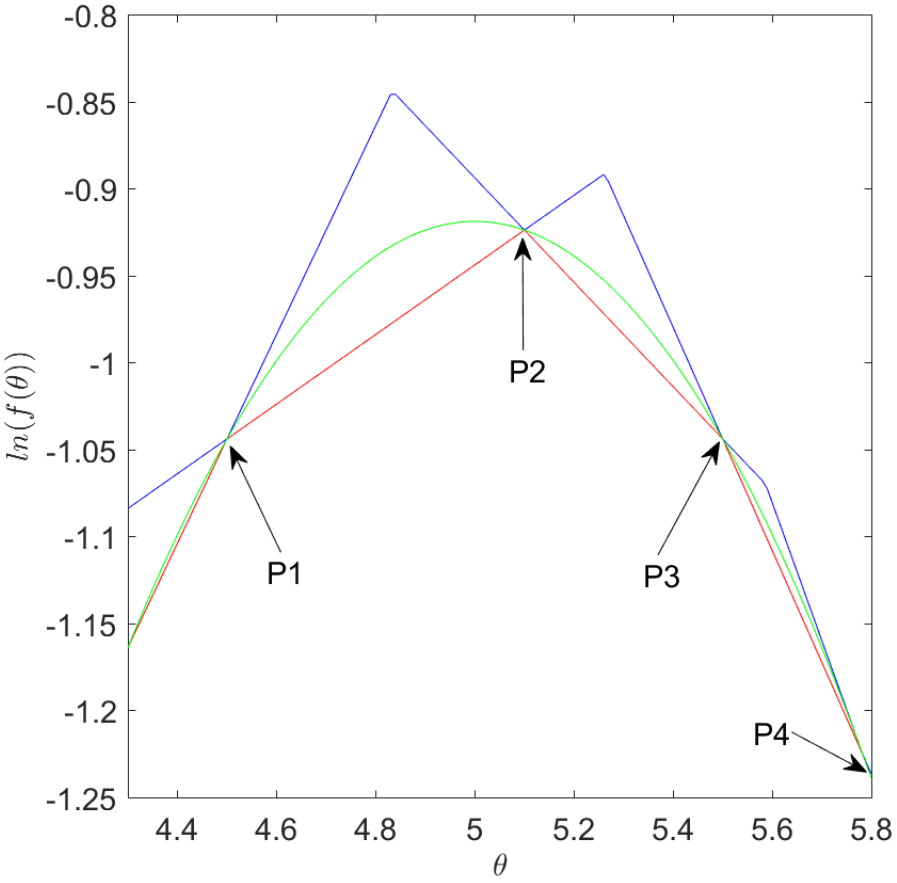


Figure E-3: (Log-scale) ARS envelope and squeezing around initial points and a new one, P4

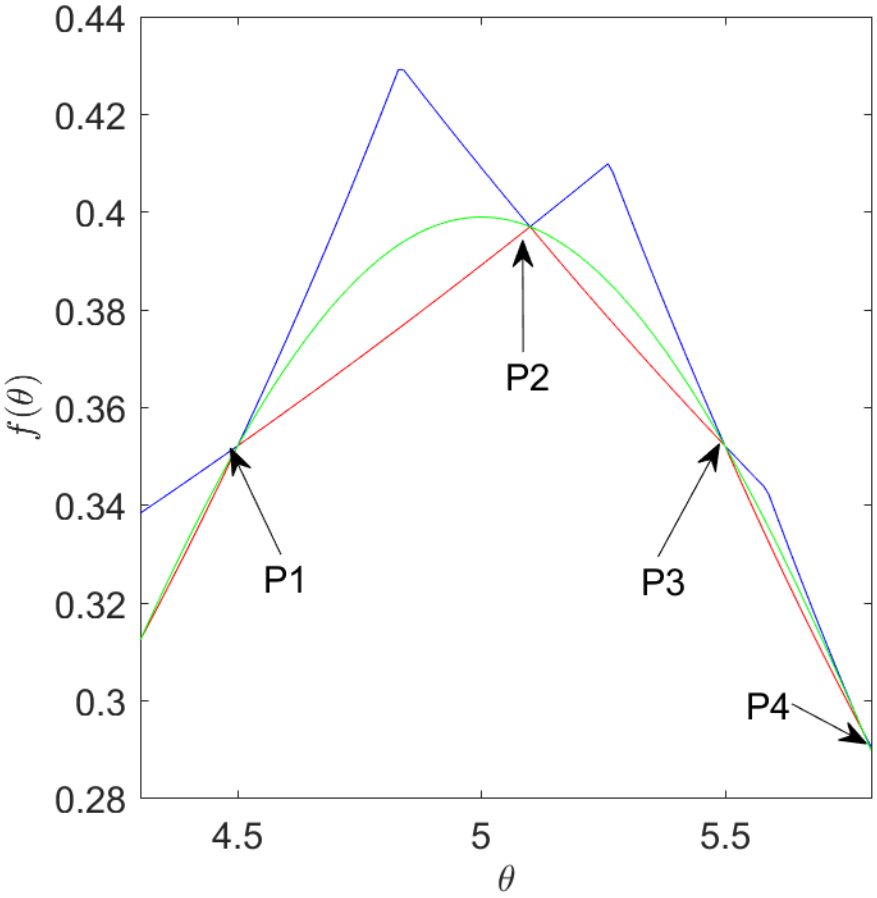


Figure E-4: (Natural-scale) Exponentiated form of Figure E-3

E.4.2.2.2 Adaptive rejection Metropolis sampling (ARMS) within Gibbs

When the full-conditional posterior distributions do not have direct sampling techniques and are not log-concave, adaptive rejection Metropolis sampling (ARMS) within Gibbs may be used. ARMS within Gibbs extends ARS by instead only performing rejection sampling (i.e., not squeezed rejection sampling) with a modified envelope function (that is not a true envelope function anymore) and an additional Metropolis rejection step (Gilks, Best, and Tan 1995, 455-472). This function is defined by the piecewise linear function,

$$h_n(x; S_n) = \max\left[L_{i,i+1}(x; S_n), \min\left\{L_{i-1,i}(x; S_n), L_{i+1,i+2}(x; S_n)\right\}\right], \qquad (E.12)$$

for $x_i \leq x < x_{i+1}$,[99] (Gilks, Best, and Tan 1995, 455-472), and it can again be converted to $g_n(\theta; S_n)$ using Equation $(E.9)$. The pseudo-code to generate T samples from a target posterior distribution, $\pi(\theta_1, \theta_2, \cdots, \theta_m|\boldsymbol{D})$, using ARMS within Gibbs is outlined in Table E-9.

[99] If b is undefined, $\min(a, b) = \min(b, a) = \max(a, b) = \max(b, a) = a$ (Gilks, Best, and Tan 1995, 455-472).

Table E-9: Pseudo-code for ARMS within Gibbs sampling

Inputs: $\boldsymbol{\theta}^{(0)}$	Initial value for the chain for each of m elements in the parameter vector
$\left[S_{n,1}^{(0)}, S_{n,2}^{(0)}, \cdots, S_{n,m}^{(0)}\right]$	Initial sets of points for each θ_j with at least one abscissa on each side of the mode of $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$
$\left[y_{n,1}^{(0)}, y_{n,2}^{(0)}, \cdots, y_{n,m}^{(0)}\right]$	Paired sets of $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$ evaluations at points in $\left[S_{n,1}^{(0)}, S_{n,2}^{(0)}, \cdots, S_{n,m}^{(0)}\right]$
$\pi\left(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right)$	Full-conditional distributions from target posterior function handle
T	Number of samples to generate
Outputs: $\boldsymbol{\theta}^{(1:T)}$	$1{:}T$ output samples for each of m elements in the parameter vector

Line	**Pseudo-code statement**	
1	for $\ell = 1$ *to* T {	// MCMC sampling iterations
2	$\boldsymbol{\theta} = \left[\theta_1^{(\ell-1)}, \theta_2^{(\ell-1)}, \cdots, \theta_m^{(\ell-1)}\right]$	// Use the previous θ_j before its update step
3	for $j = 1$ *to* m {	// Perform the componentwise updates sequentially
4	$\acute{\theta}_j \sim$ *Rejection*($\pi\left(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right)$, $g_{n,j}\left(\theta_j; S_{n,j}, y_{n,j}\right)$, $\exp\left(h_{n,j}\left(x; S_{n,j}, y_{n,j}\right)\right)$)	// *Rejection* function (Table E-3) generates the candidate values $\acute{\theta}_j$ using squeezed rejection sampling to approximate $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$; additionally pass in previous realizations of $\pi(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})$ using $y_{n,j}$that are used by $g_{n,j}(\cdot)$ and $h_{n,j}(\cdot)$to save computing time (respectively, these are function handles for sampling and envelope functions that use S_n and corresponding y_n) to save computing time
5	$U \sim Uniform(0,1)$	// Generate a uniform r.v.
6	$\alpha = \min\left(1, \frac{\pi\left(\acute{\theta}_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right)}{\pi\left(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right)} \cdot \frac{\min\left\{\pi\left(\theta_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right), g_{n,j}\left(\theta_j; S_n \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right)\right\}}{\min\left\{\pi\left(\acute{\theta}_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right), g_{n,j}\left(\acute{\theta}_j; S_n \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D}\right)\right\}}\right)$	
7	$if\ (U < \alpha)$ {	// Probability α to accept the candidate
8	$S_{n,j} = \{S_{n,j}, \acute{\theta}_j\}$	// Update our set of abscissae
9	$y_{n,j} = \{y_{n,j}, \pi(\acute{\theta}_j \vert \boldsymbol{\theta}_{.-j}, \boldsymbol{D})\}$	// Update our set of function evaluations
10	$\theta_j = \acute{\theta}_j$	
11	}	
12	}	
13	$\boldsymbol{\theta}^{(\ell)} = \boldsymbol{\theta}$	// Update the set of samples
14	}	

E.4.2.2.3 Slice sampling within Gibbs

In contrast with ARS and ARMS, slice sampling is easy to implement and tune (Neal 2003, 705-741) and is described as follows. It starts with a point, x_0, to evaluate in the desired density function, $f(x)$, and works by uniformly

sampling a value, y, from the vertical line $\left(0, f(x_0)\right)$ underneath $f(x)$, and then similarly sampling the next point, x_1, from the region $S = \{x: y < f(x)\}$ that defines the horizontal "slice" intersecting the vertical one. Neal (Neal 2003, 705-741) shows that repeated sampling in this manner, using the previous x_1 as the next x_0, converges to a Markov chain where the distribution defined by $f(x)$ is stationary (i.e., invariant). Table E-10 contains the pseudo-code for slice sampling from $f(x)$ using the "stepping out" procedure (Neal 2003, 705-741) for finding intervals within S to sample using the "shrinkage" procedure (Neal 2003, 705-741). It describes how to sample the next x_1 and uses the latest x_0 as input. A function handle for evaluating $f(x)$, and its corresponding constants w and $\mathcal{M}$, that respectively define the typical size of the slice and an integer that bounds the slice length to $\mathcal{M} \cdot w$, are also inputs.

Table E-10: Pseudo-code for *SliceSampler* (univariate)

Inputs: $\theta^{(0)}$	Previous sample
$f(\theta)$	Function proportional to the desired target distribution
w	Estimate of the typical size of a slice
$\mathcal{M}$	Integer limiting the size of a slice to $\mathcal{M} \cdot w$
Outputs: $\theta^{(1)}$	Output sample

Line	**Pseudo-code statement**	
1	$t = f\left(\theta^{(0)}\right)$	// Evaluate the function at $\theta^{(0)}$
2	$y \sim Uniform(0, f(t))$	// Sample y from the current vertical slice at $\theta^{(0)}$
3	$U \sim Uniform(0,1)$	// Start of stepping-out procedure to locate interval $I = (L, R)$ around $\theta^{(0)}$ in the horizontal slice $S = \{\theta: y < f(\theta)\}$; sample a uniform r.v., U
4	$L = \theta^{(0)} - w \cdot U$	// Initialize the left-bound for horizontal slice
5	$R = L + w$	// Initialize the right-bound for horizontal slice
6	$V \sim Uniform(0,1)$	// Sample a uniform r.v., V
7	$J = floor(\mathcal{M} \cdot V)$	// Define an upper size bound for the horizontal slice
8	while $J > 0$ and $y < f(L)$ {	// Loop until one finds a point that bounds the slice on left
9	$L = L - w$	// Step out to the left
10	$J = J - 1$	// Decrement a left-limit counter
11	}	
12	while $K > 0$ and $y < f(R)$ {	// Loop until one finds a point that bounds the slice on right
13	$R = R + w$	// Step out to the right
14	$K = K - 1$	// Decrement a left-limit counter
15	}	// Define a lower size bound
16	$\bar{L} = L$	// Set the current best left-bound
17	$\bar{R} = R$	// Set the current best right-bound
18	while (True) {	// Shrinkage loop for minimum $I = (L, R)$
19	$U \sim Uniform(0,1)$	// Sample uniform r.v., U
20	$\theta^{(1)} = \bar{L} + U \cdot (\bar{R} - \bar{L})$	// Generate a new candidate for $\theta^{(1)}$

21	if $y < f(\theta^{(1)})$ and *Accept*$(\theta^{(1)})$	// If $\theta^{(1)}$ is valid, exit; *Accept*$(\theta^{(1)})$ defines a procedure that determines validity of $\theta^{(1)}$ (i.e., it lies within $S \cap I$)
22	break	// Exit the loop
23	if $\theta^{(1)} < \theta^{(0)}$	// If $\theta^{(1)}$ is lower than $\theta^{(0)}$, update current best left-bound
24	$\bar{L} = \theta^{(1)}$	
25	else	// Else $\theta^{(1)}$ is greater than $\theta^{(0)}$, update current best right-bound
26	$\bar{R} = \theta^{(1)}$	
27	}	

Slice sampling within Gibbs uses the slice sampler within a Gibbs sampler loop to sequentially draw values from each of the full-conditional distributions. It has proved to be a popular choice for MCMC[100] due to its ease of implementation and general applicability (e.g., it supports sampling from univariate distributions that are single or multimodal and log-concave or not) (Neal 2003, 705-741). The pseudo-code to generate T samples from a target posterior distribution, $\pi(\theta_1, \theta_2, \cdots, \theta_m|\boldsymbol{D})$, using slice sampling within Gibbs is similar to the sequence outlined in Table E-7, but replaces line 4 with the contents in Table E-11 below.

Table E-11: Pseudo-code updates for slice sampling within Gibbs

Line	Pseudo-code statement	
4	$\acute{\theta}_j \sim$ *SliceSampler*$(\theta_j^{(\ell-1)}, \pi_j(\theta_j\|\boldsymbol{\theta}_{-j}, \boldsymbol{D}), w_j, \mathcal{M}_j)$	// *SliceSampler* function (Table E-10) generates the candidate values $\acute{\theta}_j$ using slice sampling to approximate $\pi_j(\theta_j\|\boldsymbol{\theta}_{-j}, \boldsymbol{D})$

E.4.2.3 Hamiltonian Monte Carlo (HMC) algorithms

For continuous distributions, Hamiltonian Monte Carlo (HMC) techniques[101] offer improved sampling performance in those frequent cases where MH and Gibbs sampling suffer from slow convergence (Neal 2011, 113-174). They do this by using constructs from Hamiltonian dynamics to generate high-acceptance proposals that are further away from current samples than ones provided by simple random-walk techniques (Neal 2011, 113-174). The premise behind the techniques involves redefining Hamilton's equations (introduced below) in terms of the desired distribution, in which the *position* variables become those of interest (i.e., from the posterior) and the *momentum* variables (artificial in this context) facilitate proposals for Metropolis updates. Hamiltonian dynamics based proposals are generated by means of the leapfrog method and these enable the sampler's exploration of corresponding path trajectories in both forwards and backwards directions (Neal 2011, 113-174).

[100] JAGS and OpenBUGS both use it extensively.

[101] Stan relies exclusively on HMC.

The Hamiltonian function, $H(\theta, r)$, and Hamilton's equations of motion (i.e., partial derivatives of the function) control how the d-dimensional *position* vector, θ, and d-dimensional *momentum* vector, r, change over a fictitious time, t (Neal 2011, 113-174)[102]. These equations facilitate Hamiltonian dynamic-based traversals across trajectories (e.g., from state $\{\theta^{(t)}, r^{(t)}\}$ at fictitious time t to state $\{\theta^{(t+s)}, r^{(t+s)}\}$ at a later time $t + s$) and are:

$$\frac{d\theta_i}{d_t} = \frac{\partial H}{\partial r_i}, \tag{E.13}$$

and

$$\frac{dr_i}{d_t} = \frac{\partial H}{\partial \theta_i}, \tag{E.14}$$

for $i = 1, \cdots, d$ (Neal 2011, 113-174). An alternate form combines θ and r into a vector $z = (\theta, r)$ and is

$$\frac{dz}{dt} = J\nabla H(z), \tag{E.15}$$

where the gradient, with respect to variable z, of Hamilton's $H(z)$ is

$$\nabla_z H(z) = \frac{\partial H}{\partial z}, \tag{E.16}$$

and

$$J = \begin{bmatrix} 0_{d\times d} & I_{d\times d} \\ -I_{d\times d} & 0_{d\times d} \end{bmatrix}, \tag{E.17}$$

where $0_{d\times d}$ is a $d \times d$ zero matrix and $I_{d\times d}$ is a $d \times d$ identity matrix (Neal 2011, 113-174).

In HMC, $H(\theta, r)$, is generally represented using the form

$$H(\theta, r) = U(\theta) + K(r), \tag{E.18}$$

that includes a *potential energy* function of position θ, $U(\theta)$, and a *kinetic energy* function of momentum r, $K(r)$ (Neal 2011, 113-174). The former is defined as a convenient constant plus the negative log-density of the desired distribution from which to sample, and the latter is

$$K(r) = \frac{r^T M^{-1} r}{2}, \tag{E.19}$$

where the symmetric, positive-definite "mass-matrix" M is usually a scalar multiple of I (Neal 2011, 113-174)[103]. This allows rewriting Equations $(E.13)$-$(E.14)$ to

[102] Neal (Neal 2011, 113-174) uses q and p respectively for the position and momentum variables. I alternately use names that match the pseudo-code from Hoffman (Hoffman and Gelman 2014, 1593-1623) that is presented later.

[103] Also, see the discussion of canonical distributions by Neal (Neal 2011, 113-174).

$$\frac{d\theta_i}{dt} = [M^{-1}r]_i \qquad (E.20)$$

and

$$\frac{dr_i}{dt} = -\frac{\partial U}{\partial \theta_i} \qquad (E.21)$$

(Neal 2011, 113-174). When the initial momentum variables are drawn from a standard normal, the Hamiltonian system is represented by the negative energy of the particle, or

$$Pr(\theta, r) \propto \exp\left(log\mathcal{L}(\theta) - \frac{1}{2}r \cdot r\right), \qquad (E.22)$$

where $\log\mathcal{L}(\theta)$ is the desired log-density for variables θ (i.e., "position-dependent negative potential energy function") and $\frac{1}{2}r \cdot r$, a simple form for Equation $(E.19)$, is the particle's kinetic energy (Hoffman and Gelman 2014, 1593-1623).

Iterations to obtain the ℓ^{th} sample using HMC include the following high-level steps: 1) initial proposals for r' are obtained by sampling from a Gaussian distribution; and 2) Hamiltonian dynamics are then used to obtain proposal states $\{\theta', r'\}$, which are either accepted or rejected by a Metropolis update (Neal 2011, 113-174). Proposal state solutions for the Hamiltonian set of linear equations are approximated by discretizing time and using the leapfrog method to compute iteratively over fictitious time steps, ε, 2ε, 3ε, etc., for some number L (Neal 2011, 113-174)[104]. The resulting proposed states are then accepted with probability

$$\alpha = \min\left\{1, \frac{\exp\left\{\log\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r'\right\}}{\exp\left\{\log\mathcal{L}(\theta^{(\ell-1)}) - \frac{1}{2}r \cdot r\right\}}\right\} \qquad (E.23)$$

(Hoffman and Gelman 2014, 1593-1623). Table E-12 contains the pseudo-code for the HMC method using the leapfrog method listed in Table E-13 (Hoffman and Gelman 2014, 1593-1623).

Table E-12: Pseudo-code for Hamiltonian Monte Carlo sampling

Inputs: $\theta^{(0)}$	Initial values for the distribution variables
ε	Size of fictitious time steps for the leapfrog algorithm
L	Number of steps for the leapfrog algorithm
$\log\mathcal{L}(\theta)$	Log-density function handle
T	Desired number of samples
Outputs: $\theta^{(1:T)}$	Distribution variables output samples
Line	Pseudo-code statement

[104] Both ε and L require tuning for optimal performance (Neal 2011, 113-174).

1	for $\ell = 1\ to\ T$ {	// Generate T samples
2	$r \sim \mathcal{N}(0, I)$	// Draw auxiliary momentum variables from zero-mean multivariate normal
3	$\theta = \left[\theta_1^{(\ell-1)}, \theta_2^{(\ell-1)}, \cdots, \theta_m^{(\ell-1)}\right]$	// Set θ to previous sample
4	$\theta' = \theta$	// Initial proposal is previous sample
5	$r' = r$	// Initial proposal is $r^{(0)}$
6	for $i = 1\ to\ L$ {	// Run L Leapfrog iterations
7	$\theta', r' = Leapfrog(\log\mathcal{L}(\theta), \theta', r', \varepsilon)$	// *Leapfrog* updates the proposal each step
8	}	
9	$r' = -r'$	// Negate momentum at end of trajectory to make the proposal symmetric
10	$U \sim Uniform(0,1)$	// Generate a uniform r.v.
11	$\alpha = \min\left\{1, \frac{\exp\left\{\log\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r'\right\}}{\exp\left\{\log\mathcal{L}(\theta^{(\ell-1)}) - \frac{1}{2}r \cdot r\right\}}\right\}$	// Compute the acceptance probability
12	$if\ (U < \alpha)$	// Probability α to accept the candidate
13	$\theta = \theta'$	// Set to proposal
14	$\theta^{(\ell)} = \theta$	// Set the output variables to the results
15	}	

Table E-13: Pseudo-code for *Leapfrog* function

Inputs: $\log\mathcal{L}(\theta)$	Log-density function handle
θ	Initial *Position* vector
r	Initial *Momentum* vector
ε	Size of fictitious time steps
Outputs: θ'	*Position* vector output proposal
r'	*Momentum* vector output proposal

Line	**Pseudo-code statement**	
1	$r' = r + (\varepsilon/2)\nabla_\theta \log\mathcal{L}(\theta)$	// Perform a half step for the momentum variables
2	$\theta' = \theta + \varepsilon r'$	// Perform a full step for the position variables using the new momentum variables
3	$r' = r' + (\varepsilon/2)\nabla_\theta \log\mathcal{L}(\theta')$	// Perform another half step for the momentum variables using the new variables for momentum and position

The dual-averaging scheme (Nesterov 2009, 221-259) extends HMC through elimination of the need for hand-tuning the leapfrog step size, ε, by alternately solving for ε using numerical optimization (Hoffman and Gelman 2014, 1593-1623). The HMC with dual-averaging algorithm converges faster when using ideal initial values, and these can be determined using the heuristic function in Table E-14 (Hoffman and Gelman 2014, 1593-1623). Table E-15 contains

the pseudo-code for the HMC algorithm with the dual-averaging scheme included (Hoffman and Gelman 2014, 1593-1623).

Table E-14: Pseudo-code for *FindReasonableEpsilon* heuristic function

Inputs: θ	Current position variables
$\log\mathcal{L}(\theta)$	Log-density function handle
Outputs: ε	Optimal value for the initial step size

Line	**Pseudo-code statement**	
1	$\varepsilon = 1$	// Start with one step
2	$r \sim \mathcal{N}(0, \mathrm{I})$	// Draw auxiliary momentum variables from zero-mean multivariate normal
3	$\theta', r' =$ *Leapfrog*$(\log\mathcal{L}(\theta), \theta, r, \varepsilon)$	// *Leapfrog* updates the proposal one step here
4	$a = 2\mathbb{I}\left[\frac{p(\theta', r')}{p(\theta, r)} > 0.5\right] - 1$	
5	while $\left(\frac{p(\theta', r')}{p(\theta, r)}\right)^a > 2^{-a}$ {	// This loop performs leapfrog updates (that double or half the step-size calculation) until acceptance probability crosses 0.5
6	$\varepsilon = 2^a\varepsilon$	// Update the step estimate
7	$\theta', r' =$ *Leapfrog*$(\log\mathcal{L}(\theta), \theta, r, \varepsilon)$	// Run a *Leapfrog* update
8	}	

The notation $\mathbb{I}[\cdot]$ is an operation that returns 1 if the expression in brackets is true or 0 if the expression in brackets is false.

Table E-15: Pseudo-code for Hamiltonian Monte Carlo sampler with dual-averaging

Inputs: $\theta^{(0)}$	Initial values for the distribution variables
δ	Target mean acceptance probability
λ	Target simulation length ($\lambda \approx \varepsilon L$)
$\log\mathcal{L}(\theta)$	Log-density function handle
T	Desired number of samples
M^{adapt}	Number of iterations after which to stop the adaptation
Outputs: $\theta^{(1:T)}$	Distribution variables output samples

Line	**Pseudo-code statement**	
1	$\varepsilon^{(0)} =$ *FindReasonableEpsilon*$(\theta, \log\mathcal{L}(\theta))$	// Call the *FindReasonableEpsilon* heuristic function to get the starting point
2	$\mu = \log(10\varepsilon^{(0)})$	// μ is a freely chosen point that the iterates $\varepsilon^{(\ell)}$ are shrunk towards (during adaptation)
3	$\bar{\varepsilon}^{(0)} = 1$	// starting point for expected value of the step size

4	$\bar{H}^{(0)} = 0$	// Initialize the test statistic for the dual averaging algorithm
5	$\gamma = 0.05$	// γ (where $\gamma > 0$) is a free parameter that controls the amount of shrinkage towards μ
6	$t_0 = 10$	// t_0 (where $t_0 \geq 0$) is a free parameter that stabilizes the initial iterations of the dual averaging adaptation algorithm
7	$\kappa = 0.75$	// κ (where $\kappa \in (0.5,1]$) is a parameter that controls the step size schedule $\eta^{(\ell)} \equiv \ell^{-\kappa}$ for the dual averaging
8	for ℓ=1 to T {	// Generate T samples
9	$r \sim \mathcal{N}(0, I)$	// Draw auxiliary momentum variables from zero-mean multivariate normal
10	$\theta = \left[\theta_1^{(\ell-1)}, \theta_2^{(\ell-1)}, \cdots, \theta_m^{(\ell-1)}\right]$	// Set θ to previous sample
11	$\theta' = \theta$	// Initial proposal is previous sample
12	$r' = r$	// Initial proposal is $r^{(0)}$
13	$L^{(\ell)} = \max\{1, Round(\lambda/\varepsilon^{(\ell)})\}$	// Compute $L^{(\ell)}$
14	for i=1 to $L^{(\ell)}$ {	// Run $L^{(\ell)}$ Leapfrog iterations
15	$\theta', r' =$ *Leapfrog*$(\log\mathcal{L}(\theta), \theta', r', \varepsilon^{(\ell-1)})$	// *Leapfrog* updates the proposal each step
16	}	
17	$r' = -r'$	// Negate momentum at end of trajectory to make the proposal symmetric
18	$U \sim Uniform(0,1)$	// Generate a uniform r.v.
19	$\alpha = \min\left\{1, \frac{\exp\{\log\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r'\}}{\exp\{\log\mathcal{L}(\theta^{(\ell-1)}) - \frac{1}{2}r \cdot r\}}\right\}$	// Compute the acceptance probability
20	$if\ (U < \alpha)$ {	// Probability α to accept the candidate
21	$\theta = \theta'$	// Set to proposal
23	}	
24	$\theta^{(\ell)} = \theta$	// Set the output variables to the results
26	if $\ell \leq M^{adapt}$ then {	// Dual-averaging adaptation section
27	$\bar{H}^{(\ell)} = \left(1 - \frac{1}{\ell + t_0}\right)\bar{H}^{(\ell-1)} + \frac{1}{\ell + t_0}(\delta - \alpha)$	
28	$\log(\varepsilon^{(\ell)}) = \mu - \frac{\sqrt{\ell}}{\gamma}\bar{H}^{(\ell)}$	// Set $\varepsilon^{(\ell)}$ (shown in log space)
29	$\log(\bar{\varepsilon}^{(\ell)}) = \ell^{-\kappa}\log(\varepsilon^{(\ell)}) + (1 - \ell^{-\kappa})\log(\bar{\varepsilon}^{(\ell-1)})$	// Update $\varepsilon^{(\ell)}$ (shown in log space)
30	}	
31	else	// Adaptation has been stopped
32	$\varepsilon^{(\ell)} = \bar{\varepsilon}^{(M^{adapt})}$	// Set $\varepsilon^{(\ell)}$ to final value from adaptation
33	}	

The No-U-Turn sampler (NUTS) (Hoffman and Gelman 2014, 1593-1623) with dual-averaging (Nesterov 2009, 221-259) extends HMC with dual-averaging by additionally eliminating the need for hand-tuning L (Hoffman and Gelman 2014, 1593-1623). Similar to HMC with dual-averaging, NUTS with dual-averaging uses the leapfrog method to incrementally identify Hamiltonian paths to be explored and solves for ε using numerical optimization. It differs by additionally exploring a range of values for L, via resampling of new slice and direction variables each iteration, and by mapping traversed paths using a balanced binary tree with paired leaf nodes that capture matching-proposal state pairs (Hoffman and Gelman 2014, 1593-1623). In each iteration, traversal stops (i.e., its sample is realized) when (a) the tree or any of its subtrees encounter a trajectory between extreme nodes that begins to change its direction towards a previous position (i.e., makes a "U-turn"), or (b) the states approach those that have unrealistic probabilities (Hoffman and Gelman 2014, 1593-1623). NUTS implements the former (a) when one of the subtrees has states $\{\theta^-, r^-\}$ (leftmost leaf) and $\{\theta^+, r^+\}$ (rightmost leaf) that satisfy either

$$(\theta^+ - \theta^-) \cdot r^- < 0 \qquad (E.24)$$

or

$$(\theta^+ - \theta^-) \cdot r^+ < 0 \qquad (E.25)$$

and the latter (b) when a leaf node exists that satisfies

$$\mathcal{L}(\theta) - \frac{1}{2} r \cdot r - \log u < -\Delta_{max}, \qquad (E.26)$$

where Δ_{max} is nonnegative (e.g., 1000) (Hoffman and Gelman 2014, 1593-1623). The pseudo-code for NUTS with dual-averaging is provided in Table E-16 and Table E-17 contains its related *BuildTree* function (Hoffman and Gelman 2014, 1593-1623).

Table E-16: Pseudo-code for efficient No-U-Turn Sampler with dual averaging

Inputs: $\theta^{(0)}$	Initial values for the distribution variables
δ	Target mean acceptance probability
$\log\mathcal{L}(\theta)$	Desired log-density for variables θ
T	Desired number of samples
M^{adapt}	Number of iterations after which to stop the adaptation
Outputs: $\theta^{(1:T)}$	Distribution variables output samples

Line	**Pseudo-code statement**	
	$\varepsilon^{(0)}$ = *FindReasonableEpsilon*$(\theta, \log\mathcal{L}(\theta))$	// Call the *FindReasonableEpsilon* heuristic function to get the starting point
	$\mu = \log(10\varepsilon^{(0)})$	// μ is a freely chosen point that the iterates $\varepsilon^{(\ell)}$ are shrunk towards (during adaptation)
	$\bar{\varepsilon}^{(0)} = 1$	// starting point for expected value of the step size

$\bar{H}^{(0)} = 0$ // Initialize the test statistic for the dual averaging algorithm

$\gamma = 0.05$ // γ (where $\gamma > 0$) is a free parameter that controls the amount of shrinkage towards μ

$t_0 = 10$ // t_0 (where $t_0 \geq 0$) is a free parameter that stabilizes the initial iterations of the dual averaging adaptation algorithm

$\kappa = 0.75$ // κ (where $\kappa \in (0.5,1]$) is a parameter that controls the step size schedule $\eta^{(\ell)} \equiv \ell^{-\kappa}$ for the dual averaging

for ℓ=1 to T { // Generate T samples

$r \sim \mathcal{N}(0, I)$ // Draw auxiliary momentum variables from zero-mean multivariate normal

$u \sim Uniform\left(\left[0, \exp\left\{\log\mathcal{L}\left(\theta^{(\ell-1)}\right) - \frac{1}{2} r \cdot r\right\}\right]\right)$ // Resample $u|\theta, r$

$\theta^- = \theta^{(\ell-1)}; \theta^+ = \theta^{(\ell-1)}$ // Set starting points

$r^- = r; r^+ = r$

$j = 0; n = 1; s = 1$

$\theta^{(\ell)} = \theta^{(\ell-1)}$

$L^{(\ell)} = \max\left\{1, Round\left(\lambda/\varepsilon^{(\ell)}\right)\right\}$ // Compute $L^{(\ell)}$

while $s = 1$ { // Loop until U-turn detected (i.e., subtrajectory from the leftmost to the rightmost nodes of any balanced subtree of the overall binary tree starts to double back on itself

$v_j \sim Uniform(\{-1,1\})$ // Sample from a discrete uniform to choose direction (i.e., -1 or 1)

if $v_j == -1$

θ^-, r^-,t1,t2,θ',n',s',α,n_α = *BuildTree*($\log\mathcal{L}(\theta), \theta^-, r^-, u, v_j, j, \varepsilon^{(\ell-1)}, \theta^{(\ell-1)}, r$) // t1 and t2 are dummy variables; n_α is only used by *BuildTree* (when it calls itself recursively)

else

t1,t2,$\theta^+, r^+, \theta', n', s', \alpha, n_\alpha$ = *BuildTree*($\log\mathcal{L}(\theta), \theta^+, r^+, u, v_j, j, \varepsilon^{(\ell-1)}, \theta^{(\ell-1)}, r$) // t1 and t2 are dummy variables; n_α is only used by *BuildTree* (when it calls itself recursively)

if $s' == 1$ {

$U \sim Uniform(0,1)$ // Generate a uniform r.v.

$\alpha = \min\left\{1, \frac{n'}{n}\right\}$ // Compute the acceptance probability

if $(U < \alpha)$ // Probability α to accept the candidate

$\theta^{(\ell)} = \theta'$ // Set to proposal

}

$n \leftarrow n + n'$

	$s = s'\mathbb{I}[(\theta^+ - \theta^-)\cdot r^- \geq 0]\cdot$ $\mathbb{I}[(\theta^+ - \theta^-)\cdot r^+ \geq 0]$	// Check for U-turn condition
	$j \leftarrow j + 1$	
	}	
	if $\ell \leq M^{adapt}$ then {	// Dual-averaging adaptation section
	$\bar{H}^{(\ell)} = \left(1 - \frac{1}{\ell + t_0}\right)\bar{H}^{(\ell-1)} + \frac{1}{\ell + t_0}\left(\delta - \frac{\alpha}{n_\alpha}\right)$	
	$\log\left(\varepsilon^{(\ell)}\right) = \mu - \frac{\sqrt{\ell}}{\gamma}\bar{H}^{(\ell)}$	// Set $\varepsilon^{(\ell)}$ (shown in log space)
	$\log\left(\bar{\varepsilon}^{(\ell)}\right) = \ell^{-\kappa}\log\left(\varepsilon^{(\ell)}\right) +$ $(1 - \ell^{-\kappa})\log\left(\bar{\varepsilon}^{(\ell-1)}\right)$	// Update $\varepsilon^{(\ell)}$ (shown in log space)
	}	
	else	// Adaptation has been stopped
	$\varepsilon^{(\ell)} = \bar{\varepsilon}^{(M^{adapt})}$	// Set $\varepsilon^{(\ell)}$ to final value from adaptation
	}	

Table E-17: Pseudo-code for recursive *BuildTree* function

Inputs:	
$\log\mathcal{L}(\theta)$	Log-density function handle
θ	Position variable
r	Momentum variable
u	Slice variable u used to augment $\Pr(\theta, r);\ \Pr(\theta, r, u) \propto \mathbb{I}\left[\mathrm{u} \in \left[0, \exp\left\{\log\mathcal{L}(\theta) - \frac{1}{2}\mathrm{r}\cdot\mathrm{r}\right\}\right]\right]$
v	Direction (-1 backwards, 1 forwards)
j	Depth in tree
ε	Size of fictitious time steps
$\theta^{(0)}$	Initial values for the position variables (used for computing acceptance probability)
$r^{(0)}$	Initial values for the momentum variables (used for computing acceptance probability)
Outputs:	
θ^-	Leftmost leaf state's position
r^-	Leftmost leaf state's momentum
θ^+	Rightmost leaf state's position
r^+	Rightmost leaf state's momentum
θ'	Sample proposal
n'	number of valid points
s'	stopping criterion.
α'	acceptance probability statistic
n'_α	acceptance probability statistic
Line	**Pseudo-code statement**

	if j==0 {	// Base case—take one leapfrog step in the direction v
	$\theta', r' = LeapFrog(\log\mathcal{L}(\theta)\theta, r, v \cdot \varepsilon)$	
	$n' \leftarrow \mathbb{I}\left[u \leq \exp\left\{\log\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r'\right\}\right]$	
	$s' \leftarrow \mathbb{I}\left[u < \exp\left\{\Delta_{max} + \log\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r'\right\}\right]$	
	$\theta^- = \theta'; r^- = r'; \theta^+ = \theta'; r^+ = r'$	
	$\alpha' = \min\{1, \exp\left\{\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r' - \log\mathcal{L}(\theta^0) + \frac{1}{2}r^{(0)} \cdot r^{(0)}\right\}\}$	
	$n'_\alpha = 1$	
	}	
	else {	// Recursion—implicitly build the left and right subtrees at depth-1
	$\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha =$ *BuildTree*$(\log\mathcal{L}(\theta), \theta, r, u, v, j-1, \varepsilon, \theta^0, r^0)$	
	if $s' == 1$ {	// Stop criterion when s' is 0
	if $v = -1$ then {	// Backwards
	$t1, t2, t3, t4, \theta'', n'', s'', \alpha'', n''_\alpha =$ *BuildTree*$(\log\mathcal{L}(\theta), \theta^-, r^-, u, v, j-1, \varepsilon, \theta^{(0)}, r^{(0)})$	
	$\theta^- = t1; r^- = t2$	
	}	
	else {	// Forwards
	$t1, t2, t3, t4, \theta'', n'', s'', \alpha'', n''_\alpha =$ BuildTree$(\log\mathcal{L}(\theta), \theta^+, r^+, u, v, j-1, \varepsilon, \theta^{(0)}, r^{(0)})$	
	$\theta^+ = t3; r^+ = t4$	
	}	
	$U \sim Uniform(0,1)$	// Choose which subtree to propagate a sample up from using a Metropolis update
	$\alpha = \frac{n''}{n'+n''}$	
	if $(U < \alpha)$	
	$\theta' = \theta''$	
	$\alpha' = \alpha' + \alpha''$	// Update the acceptance probability statistics
	$n'_\alpha = \quad n'_\alpha + n''_\alpha$	
	$s' \leftarrow s''\mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \cdot \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$	// Update the stopping criterion.
	$n' \leftarrow n' + n''$	// Update the number of valid points.
	}	
	}	

E.4.3 MCMC summary statistics

MCMC readily estimates posterior distribution mean and standard deviation. Let $\theta^{(\ell)}$ designate the Monte Carlo ℓ^{th} step for variable θ. The sample mean (after B burn-in iterations) for θ supports estimation of the population mean using

$$E[\pi(\theta|\boldsymbol{D})] = \bar{\theta} \approx \frac{1}{T-B}\sum_{\ell=B+1}^{T}\theta^{(\ell)}, \qquad (E.27)$$

where the "burn-in" threshold is the sample, $\theta^{(B)}$, such that the ergodic mean[105] has converged to some stable value (Gilks, Richardson, and Spiegelhalter 1996, 1-20). Population standard deviation (SD, or the square root of the population variance, denoted Var) is likewise estimated from the sample mean (after B burn-in iterations)

$$SD[\pi(\theta|\boldsymbol{D})] \approx \sqrt{\frac{1}{T-B-1}\cdot\sum_{\ell=B+1}^{T}(\theta^{(\ell)}-\bar{\theta})^2}, \qquad (E.28)$$

(Gilks, Richardson, and Spiegelhalter 1996, 1-20).

Highest posterior density (HPD) intervals describe the range of values in which the majority of the MCMC samples for a particular model variable lie within a distribution (Box and Tiao 1992). As implied by the name, the variable distribution's densities of the points within the interval surpass their outside counterparts. Furthermore, the interval defines the shortest range of values for a specified probability total. To explain for a variable θ, the HPD interval, $\left(\theta^{\left(\frac{\alpha}{2}\right)}, \theta^{\left(1-\frac{\alpha}{2}\right)}\right)$, defines the range of values in $\pi(\theta|\boldsymbol{D})$ with total probability $1-\alpha$ (M. Chen, Shao, and Ibrahim 2000, 213-235). Table E-18 outlines the procedure for HPD estimation based on the Chen, Shao (M. Chen, Shao, and Ibrahim 2000, 213-235) algorithm.

Table E-18: Pseudo-code for highest posterior density estimation

Inputs: $\{\theta^{(i)}, \theta^{(ii)}, \dots, \theta^{(i\cdots T)}\}$	$\pi(\theta\|\boldsymbol{D})$ samples from MCMC
Outputs: R_*	Estimated interval for the range of values in $\pi(\theta\|\boldsymbol{D})$ with total probability $1-\alpha$

Line	**Pseudo-code statement**	
1	$\{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(T)}\} = Sort(\{\theta^{(i)}, \theta^{(ii)}, \dots, \theta^{(i\cdots T)}\})$	*//Sort* function orders $\{\theta^{(i)}, \theta^{(ii)}, \dots, \theta^{(i\cdots T)}\}$ *to obtain increasingly ordered* $\{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(T)}\}$
2	for $j = 1$ to $T - [(1-\alpha)T]$ {	
3	$R_j = \left(\theta^{(j)}, \theta^{(j+[(1-\alpha)T])}\right)$	*//compute the j^{th} CI*
4	if $(R_j < R_*)$	
5	$R_* = R_j$	*//save the new HPD estimate if j^{th} CI is smallest that achieves probability $1-\alpha$*

[105] The ergodic mean is the MCMC sample mean value up to the current iteration (Ntzoufras 2009).

Inputs: $\{\theta^{(i)}, \theta^{(ii)}, \dots, \theta^{(i\cdots T)}\}$	$\pi(\theta\|\boldsymbol{D})$ samples from MCMC
Outputs: R_*	Estimated interval for the range of values in $\pi(\theta\|\boldsymbol{D})$ with total probability $1-\alpha$

Line	**Pseudo-code statement**
6	else
7	continue
8	}

E.4.4 MCMC diagnostics

When using MCMC to estimate the posterior distribution for a variable θ, $\pi(\theta|\boldsymbol{D})$, MCMC results should be validated by performing various convergence diagnostic methods. This is because chain failure to converge to the stationary distribution is an indicator for MCMC failure. Unstable ergodic means, non-random patterns in trace plots, elevated Monte Carlo error (MCE) statistics, and correlated parameter samples all suggest convergence issues[106]. A converging algorithm will have stable and almost identical ergodic means (Ntzoufras 2009). Trace plots are plots of the raw MCMC samples and should appear random and have no obvious periodic indicators (Ntzoufras 2009). Pettitt and Hincksman (Pettitt and Hincksman 2012, 17-29) recommend that an adequate MCMC sample size retains $MCE \leq .01 \cdot SD\big(\pi(\theta|\boldsymbol{D})\big)$. Sample autocorrelation and cross-correlation should be low (near 0 to justify convergence) as Markov chains with high correlation (near 1) traverse the parameter space slowly and can take longer to converge (Hoff 2009). Likewise, Markov chains with low correlations (e.g., near zero), traverse the parameter space faster, and support claims for chain convergence (Ntzoufras 2009).

Monte Carlo error (MCE) is estimated by first dividing the samples in the range $\left[\theta^{(B+1)}, \theta^{(T)}\right]$ into $\mathcal{K}$ batches (usually $\mathcal{K} = 30$ or $\mathcal{K} = 50$) (Ntzoufras 2009). The sample size of each batch will then be $\nu = \frac{\mathrm{T-B+1}}{\mathcal{K}}$ (Ntzoufras 2009). The mean for the b^{th} batch of posterior samples for θ is then

$$\bar{\theta_b} = \frac{1}{\nu}\sum_{\ell=(b-1)\cdot\nu+1}^{b\nu} \theta^{(\ell)}, \qquad (E.29)$$

and the batch mean estimator of the MCE is the variance of $\bar{\theta_b}$

$$MCE[\theta] = \bar{\mathrm{S}}_{\mathrm{b}}^2 = \frac{1}{\mathcal{K}}\bar{\mathrm{S}}_{\mathrm{b}}^2 = \frac{1}{\mathcal{K}(\mathcal{K}-1)}\sum_{b=1}^{\mathcal{K}}(\bar{\theta_b} - \bar{\theta})^2, \qquad (E.30)$$

(Ntzoufras 2009). The covariance between two random variables X and Y is defined as

$$Cov(X,Y) = E[(X-\bar{X})(Y-\bar{Y})], \qquad (E.31)$$

[106] In general, practitioners cannot validate achievement of convergence; however, in some cases MCMC diagnostics imply non-convergence (Hoff 2009).

and the coefficient of correlation of between X and Y is

$$Corr(X,Y) = \frac{Cov(X,Y)}{SD_X SD_Y}, \qquad (E.32)$$

(Martinez and Martinez 2008). In MCMC, it is important to monitor two diagnostic measures: autocorrelation and cross-correlation. Once stationarity has been achieved, the sample autocorrelation (i.e., correlation between two values in the chain which are either consecutive or lag L steps apart) function is given by

$$ACorr_L(\theta) = \frac{\frac{1}{T-B-1-L}\sum_{\ell=B+1}^{T-L}\left(\theta^{(\ell)} - \bar{\theta}\right)\left(\theta^{(\ell+L)} - \bar{\theta}\right)}{\frac{1}{T-B-1}\sum_{\ell=B+1}^{T}(\theta^{(\ell)} - \bar{\theta})^{\,2}}, \qquad (E.33)$$

(Hoff 2009). Similarly, the sample cross-correlation (i.e., correlation between parameters θ_i and θ_j which are either consecutive or L steps apart) function is given by

$$XCorr_L(\theta_i,\theta_j) = \frac{\frac{1}{T-B-1-L}\sum_{\ell=B+1}^{T-L}\left(\theta_i^{(\ell)} - \bar{\theta}_i\right)\left(\theta_j^{(\ell+L)} - \bar{\theta}_j\right)}{\frac{1}{T-B-1}\sqrt{\sum_{\ell=B+1}^{T}(\theta_i^{(\ell)} - \bar{\theta}_i)^{\,2}}\cdot\sqrt{\sum_{\ell=B+1}^{T}(\theta_j^{(\ell)} - \bar{\theta}_j)^{\,2}}}. \qquad (E.34)$$

Appendix F. Model choice techniques

This section provides an overview for one of the model choice techniques and a measure for information gain in the Bayesian analysis. Both are implemented in MCMCBayes.

F.1 Power-posterior approach

This research estimates the marginal likelihood, denoted $Z_k = Pr(\boldsymbol{D})$, using a thermodynamic integration-based technique[107] (Ogata 1989, 137-157). The technique, referred to as the power-posterior approach (Friel and Pettitt 2008, 589-607), approximates the integration over the realm of possible values for the variable Θ; this method was independently discovered by both Lartillot et al. (Lartillot and Philippe 2004, 1095-1109; Lartillot and Philippe 2006, 195-207) and Friel et al. (Friel and Pettitt 2008, 589-607).

Details for the power-posterior approach are as follows. Lartillot et al. (Lartillot and Philippe 2006, 195-207) define the un-normalized density to be

$$\pi_\phi(\theta|\boldsymbol{D}) \propto \mathcal{L}(\theta|\boldsymbol{D})^\phi \cdot \pi(\theta), \qquad (F.1)$$

and explain its corresponding sampling path as the continuous and differentiable path[108] traversing the temperature parameter ϕ from the prior ($\phi = 0$) to the un-normalized posterior ($\phi = 1$). Letting Z_ϕ denote the normalizing constant for $\int_\theta \ \pi_\phi(\theta|\boldsymbol{D}) \cdot d\theta$, the corresponding key identity for this method is

$$\log\big(Pr(\boldsymbol{D})\big) = \log Z_1 - \log Z_0 = \int_0^1 E_{\theta|\boldsymbol{D},\phi}\big[\log\big(\mathcal{L}(\theta|\boldsymbol{D})\big)\big]d\phi\,, \qquad (F.2)$$

(Lartillot and Philippe 2006, 195-207)[109].

In this research application, the path across ϕ will be discretized into n_ϕ steps according to the serial MCMC approach presented by Friel et al. (Friel and Pettitt 2008, 589-607). For every discrete step $\phi_i = \left(\frac{i}{n_\phi}\right)^{c_\phi}$, [110], MCMC is performed for $\log\left(\pi_\phi(\theta|\boldsymbol{D})\right)$ to estimate $E_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, $Var_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, and $MCE_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, [111]. Then, the marginal log-

---

[107] Also known as path sampling by Gelman et al. (Gelman and Meng 1998, 163-185).

[108] In the space of un-normalized densities (Lartillot and Philippe 2006, 195-207)

[109] Here, $E_{\Theta|\boldsymbol{D},\phi}[\cdot]$ stands for the expectation when using the MCMC sampled values for θ, given $\boldsymbol{D}$ and ϕ.

[110] Friel et al. (Friel and Pettitt 2008, 589-607) define $c_\phi > 1$ as a constant which ensures the temperature steps are "chosen with high frequency close to 0" and they recommend $c_\phi = 5$ or $c_\phi = 3$. Their recommended range for the number of steps n_ϕ is $20 - 100$. To achieve reasonable simulation performance, I used $n_\phi = 30$ and $c_\phi = 5$.

[111] Note: $E_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, $Var_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, and $MCE_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$ above are not estimated using samples from the posterior distribution; rather, the MCMC samples for θ are drawn from the *power-posterior* distribution and these are used to estimate $E_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, $Var_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, and $MCE_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$. Estimation is performed by placing the sampled values into the log-likelihood function $\log\big(\mathcal{L}(\theta|\boldsymbol{D})\big)$ and then using these results to obtain $E_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, $Var_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$, and $MCE_{\Theta|\boldsymbol{D},\phi_i}[\cdot]$.

likelihood of $\boldsymbol{D}$ is the integration across all of the ϕ_i discrete steps. This integral is approximated numerically using the trapezoidal rule (Larson and Edwards 2014), and the corresponding bias-reduced marginal likelihood estimator is

$$\log\left(Pr(\boldsymbol{D})\right) \cong$$

$$\sum_{i=0}^{n-1}(\phi_{i+1}-\phi_i)\frac{\left[E_{\Theta|\boldsymbol{D},\phi_{i+1}}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]+E_{\Theta|\boldsymbol{D},\phi_i}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]\right]}{2} -$$

$$\sum_{i=0}^{n-1}\frac{(\phi_{i+1}-\phi_i)^2}{12}\left[\mathrm{Var}_{\Theta|\boldsymbol{D},\phi_{i+1}}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]-\mathrm{Var}_{\Theta|\boldsymbol{D},\phi_i}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]\right], \qquad (F.3)$$

(Friel, Hurn, and Wyse 2014, 709-723). Additionally, the corresponding Monte Carlo error for $\log\left(Pr(\boldsymbol{D})\right)$ is

$$\mathrm{MCE}\left[\log\left(Pr(\boldsymbol{D})\right)\right] \cong$$

$$\sqrt{\frac{(\phi_2-\phi_1)^2}{2}s_1^2+\sum_{i=2}^{n-1}\frac{(\phi_{i+1}-\phi_i)^2}{2}s_i^2+\frac{(\phi_n-\phi_{n-1})^2}{2}s_{n_\phi}^2}, \qquad (F.4)$$

where $s_i = \mathrm{MCE}_{\Theta|\boldsymbol{D},\phi_i}\left[E_{\Theta|\boldsymbol{D},\phi_i}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]\right]$ (Friel and Pettitt 2008, 589-607). Table F-1 contains pseudo-code for an example power-posterior sequence that estimates $\log\left(Pr(\boldsymbol{D})\right)$ and its corresponding MCE (also, see the example MATLAB session in Table I-12, Appendix I).

Table F-1: Pseudo-code for power-posterior sequence

Inputs:	
$\boldsymbol{\theta}_0^{(0)}$	Initial model parameter values for first temperature step (i.e., ϕ_0)
c_ϕ	Temperature schedule spacing parameter
n_ϕ	Number of discretization steps in the interval [0,1]
PPSampler($\log\left(\mathcal{L}(\theta\|\boldsymbol{D})\right)$,$\phi_i$)	Power-posterior sampling function handle that returns $E_{\Theta|\boldsymbol{D},\phi_i}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]$, $\mathrm{Var}_{\Theta|\boldsymbol{D},\phi_i}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]$, and $\mathrm{MCE}_{\Theta|\boldsymbol{D},\phi_i}\left[E_{\Theta|\boldsymbol{D},\phi_i}\left[\log\left(\mathcal{L}(\theta|\boldsymbol{D})\right)\right]\right]$ for a specified ϕ_i
Outputs:	
PP	Marginal log-likelihood estimate (i.e., $\log\left(Pr(\boldsymbol{D})\right)$)
$PPMCE$	Monte Carlo standard error for the marginal log-likelihood estimate (i.e., $\mathrm{MCE}\left[\log\left(Pr(\boldsymbol{D})\right)\right]$)

Line	**Pseudo-code statement**	
	$PP = 0$	
	$PPMCE = 0$	
	for i=0 to n_ϕ {	// Loop over the temperature steps
	$\phi_i = \left(\frac{i}{n_\phi}\right)^{c_\phi}$	// Set the temperature parameter value for this step according to a schedule that places more steps nearer to zero

	$E_i, Var_i, s_i =$ *PPSampler*$(\log(\mathcal{L}(\Theta\|\boldsymbol{D})),\phi_i)$	// Sample from the power-posterior; $E_i = E_{\Theta\|\boldsymbol{D},\phi_i}[\log(\mathcal{L}(\Theta\|\boldsymbol{D}))]$, $Var_i = \mathrm{Var}_{\Theta\|\boldsymbol{D},\phi_i}[\log(\mathcal{L}(\Theta\|\boldsymbol{D}))]$, and $s_i = \mathrm{MCE}_{\Theta\|\boldsymbol{D},\phi_i}\left[E_{\Theta\|\boldsymbol{D},\phi_i}[\log(\mathcal{L}(\Theta\|\boldsymbol{D}))]\right]$
	$\boldsymbol{\theta}_{i+1}^{(0)} = \hat{\boldsymbol{\theta}}_i$	// Initialize the next starting value using the point estimate from this step
	if $i \geq 1$	
	$PP = PP + \frac{(\phi_{i+1}-\phi_i)(E_i+E_{i-1})}{2} - \frac{(\phi_{i+1}-\phi_i)^2(Var_i+Var_{i-1})}{12}$	// Use trapezoidal rule to estimate $\log(Pr(\boldsymbol{D}))$
	if $i == 1$	// Estimate $MCE[\log(Pr(\boldsymbol{D}))]$
	$PPMCE = PPMCE + \frac{(\phi_2-\phi_1)^2 s_1^2}{2}$	
	elseif $i < n_\phi$	
	$PPMCE = PPMCE + \frac{(\phi_{i+1}-\phi_i)^2 s_i^2}{2}$	
	else	
	$PPMCE = PPMCE + \frac{\left(\phi_{n_\phi}-\phi_{n_\phi-1}\right)^2 s_{n_\phi}^2}{2}$	
	}	
	$PPMCE = \sqrt{PPMCE}$	

F.2 Kullback-Leibler distance

When performing Bayesian analysis with expert judgment elicited data, it is useful to quantify the discrepancy between the model variable prior distributions and their corresponding posterior distributions. A common measure for this, the Kullback-Leibler (KL) distance (Hastie 1987, 16-20; Kullback et al. 1987, 338-341), is easily computed using the already available marginal likelihood of the data and the posterior likelihood (both are artifacts from the power-posterior sequence). The interpretation of the KL is as follows. When updating from prior to posterior beliefs, larger KL distances are indicative of higher information gains (Friel and Pettitt 2008, 589-607). The equation,

$$KL(\pi(\Theta), \pi(\Theta|\boldsymbol{D})) = \int \log\left(\frac{Pr(\Theta|\boldsymbol{D})}{Pr(\Theta)}\right) Pr(\Theta|\boldsymbol{D}) d\Theta \qquad (F.5)$$

defines the Kullback-Leibler distance (Friel and Pettitt 2008, 589-607). The following equation,

$$KL(\pi(\Theta), \pi(\Theta|\boldsymbol{D})) = E_{\Theta|\boldsymbol{D}}[\log(Pr(\boldsymbol{D}|\Theta))] - \log(Pr(\boldsymbol{D})) \qquad (F.6)$$

represents key implementation details for its calculation (Friel and Pettitt 2008, 589-607).

Appendix G. Elicitation results supplement

Table G-1: x_j^1 scenarios

j	$x_{j_{41}}^1$	$x_{j_3}^1$	$x_{j_{28}}^1$	$x_{j_{23}}^1$	$x_{j_{29}}^1$	$x_{j_{39}}^1$	$x_{j_1}^1$	$x_{j_{18}}^1$	$x_{j_{20}}^1$	$x_{j_{17}}^1$
1	10000	5000	3	10	3	3	0.25	0.25	0.5	0.5
2	50	1000	3	10	3	3	0.25	0.25	0.5	0.5
3	1000	500	2	10	3	3	0.25	0.25	0.5	0.5
4	1000	500	3	30	3	3	0.25	0.25	0.5	0.5
5	1000	500	3	10	5	3	0.25	0.25	0.5	0.5
6	1000	500	3	10	3	2	0.25	0.25	0.5	0.5
7	1000	500	3	10	3	3	0.75	1	0.5	0.5
8	1000	500	3	10	3	3	0.1	0	0.5	0.5
9	1000	500	3	10	3	3	0.25	0.25	0.75	1
10	1000	500	3	10	3	3	0.25	0.25	0.75	1
11	50	50	3	10	3	3	0.25	0.25	0.5	0.5
12	1000	5000	3	10	3	3	0.25	0.25	0.5	0.5
13	1000	500	5	10	3	3	0.25	0.25	0.5	0.5
14	1000	500	3	30	3	3	0.25	0.25	0.5	0.5
15	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
16	1000	500	3	10	3	2	0.25	0.25	0.5	0.5
17	1000	500	3	10	3	3	0.75	0.25	0.5	0.5
18	1000	500	3	10	3	3	0.1	0.25	0.5	0.5
19	1000	500	3	10	3	3	0.25	0.25	0.75	1
20	1000	500	3	10	3	3	0.25	0.25	1	0.75
21	10000	100000	3	10	3	3	0.25	0.25	0.5	0.5
22	1000	5000	3	10	3	3	0.25	0.25	0.5	0.5
23	1000	500	1	10	3	3	0.25	0.25	0.5	0.5
24	1000	500	3	5	3	3	0.25	0.25	0.5	0.5
25	1000	500	3	10	1	3	0.25	0.25	0.5	0.5
26	1000	500	3	10	3	4	0.25	0.25	0.5	0.5
27	1000	500	3	10	3	3	0.5	0.25	0.5	0.5
28	1000	500	3	10	3	3	0.25	1	0.5	0.5
29	1000	500	3	10	3	3	0.25	0.25	0.75	0.25
30	1000	500	3	10	3	3	0.25	0.25	0.5	0.25
31	50	100000	3	10	3	3	0.25	0.25	0.5	0.5
32	10000	1000	3	10	3	3	0.25	0.25	0.5	0.5
33	1000	500	5	10	3	3	0.25	0.25	0.5	0.5
34	1000	500	3	3	3	3	0.25	0.25	0.5	0.5
35	1000	500	3	10	3	3	0.25	0.25	0.5	0.5

j	$x_{j_{41}}^1$	$x_{j_3}^1$	$x_{j_{28}}^1$	$x_{j_{23}}^1$	$x_{j_{29}}^1$	$x_{j_{39}}^1$	$x_{j_1}^1$	$x_{j_{18}}^1$	$x_{j_{20}}^1$	$x_{j_{17}}^1$
36	1000	500	3	10	3	1	0.25	0.25	0.5	0.5
37	1000	500	3	10	3	3	0.75	1	0.5	0.5
38	1000	500	3	10	3	3	0.75	0	0.5	0.5
39	1000	500	3	10	3	3	0.25	0.25	0.5	0.75
40	1000	500	3	10	3	3	0.25	0.25	0.25	0.5
41	10000	1000	3	10	3	3	0.25	0.25	0.5	0.5
42	250	1000	3	10	3	3	0.25	0.25	0.5	0.5
43	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
44	1000	500	3	3	3	3	0.25	0.25	0.5	0.5
45	1000	500	3	10	5	3	0.25	0.25	0.5	0.5
46	1000	500	3	10	3	5	0.25	0.25	0.5	0.5
47	1000	500	3	10	3	3	0.1	0.15	0.5	0.5
48	1000	500	3	10	3	3	0.25	0	0.5	0.5
49	1000	500	3	10	3	3	0.25	0.25	0.75	0.25
50	1000	500	3	10	3	3	0.25	0.25	1	1

Table G-2: $\boldsymbol{x}_j^2$ scenarios

j	$x_{j_{41}}^2$	$x_{j_3}^2$	$x_{j_{28}}^2$	$x_{j_{23}}^2$	$x_{j_{29}}^2$	$x_{j_{39}}^2$	$x_{j_1}^2$	$x_{j_{18}}^2$	$x_{j_{20}}^2$	$x_{j_{17}}^2$
1	100000	1000	3	10	3	3	0.25	0.25	0.5	0.5
2	50	50	3	10	3	3	0.25	0.25	0.5	0.5
3	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
4	1000	500	3	3	3	3	0.25	0.25	0.5	0.5
5	1000	500	3	10	2	3	0.25	0.25	0.5	0.5
6	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
7	1000	500	3	10	3	3	0.5	0.25	0.5	0.5
8	1000	500	3	10	3	3	0.75	0.5	0.5	0.5
9	1000	500	3	10	3	3	0.25	0.25	0.5	0.25
10	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
11	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
12	100000	100000	3	10	3	3	0.25	0.25	0.5	0.5
13	1000	500	1	10	3	3	0.25	0.25	0.5	0.5
14	1000	500	3	3	3	3	0.25	0.25	0.5	0.5
15	1000	500	3	10	5	3	0.25	0.25	0.5	0.5
16	1000	500	3	10	3	1	0.25	0.25	0.5	0.5
17	1000	500	3	10	3	3	0.1	0.5	0.5	0.5
18	1000	500	3	10	3	3	0.1	1	0.5	0.5
19	1000	500	3	10	3	3	0.25	0.25	0.5	0.25
20	1000	500	3	10	3	3	0.25	0.25	0	0.5

j	$x_{j_{41}}^2$	$x_{j_3}^2$	$x_{j_{28}}^2$	$x_{j_{23}}^2$	$x_{j_{29}}^2$	$x_{j_{39}}^2$	$x_{j_1}^2$	$x_{j_{18}}^2$	$x_{j_{20}}^2$	$x_{j_{17}}^2$
21	100000	5000	3	10	3	3	0.25	0.25	0.5	0.5
22	250	500	3	10	3	3	0.25	0.25	0.5	0.5
23	1000	500	4	10	3	3	0.25	0.25	0.5	0.5
24	1000	500	3	20	3	3	0.25	0.25	0.5	0.5
25	1000	500	3	10	3	3	0.25	0.25	0.5	0.5
26	1000	500	3	10	3	1	0.25	0.25	0.5	0.5
27	1000	500	3	10	3	3	0.75	1	0.5	0.5
28	1000	500	3	10	3	3	0.1	0.25	0.5	0.5
29	1000	500	3	10	3	3	0.25	0.25	0.5	1
30	1000	500	3	10	3	3	0.25	0.25	0.25	0.25
31	100000	1000	3	10	3	3	0.25	0.25	0.5	0.5
32	250	100000	3	10	3	3	0.25	0.25	0.5	0.5
33	1000	500	1	10	3	3	0.25	0.25	0.5	0.5
34	1000	500	3	30	3	3	0.25	0.25	0.5	0.5
35	1000	500	3	10	2	3	0.25	0.25	0.5	0.5
36	1000	500	3	10	3	5	0.25	0.25	0.5	0.5
37	1000	500	3	10	3	3	0.1	0	0.5	0.5
38	1000	500	3	10	3	3	0	0.15	0.5	0.5
39	1000	500	3	10	3	3	0.25	0.25	0.75	0
40	1000	500	3	10	3	3	0.25	0.25	1	0.75
41	100000	100000	3	10	3	3	0.25	0.25	0.5	0.5
42	1000	1000	3	10	3	3	0.25	0.25	0.5	0.5
43	1000	500	2	10	3	3	0.25	0.25	0.5	0.5
44	1000	500	3	20	3	3	0.25	0.25	0.5	0.5
45	1000	500	3	10	4	3	0.25	0.25	0.5	0.5
46	1000	500	3	10	3	1	0.25	0.25	0.5	0.5
47	1000	500	3	10	3	3	0.75	0.25	0.5	0.5
48	1000	500	3	10	3	3	0	0.5	0.5	0.5
49	1000	500	3	10	3	3	0.25	0.25	0.25	1
50	1000	500	3	10	3	3	0.25	0.25	0.75	0

Appendix H. Calibration and aggregation results supplement

Table H-1 and Table H-2 provide the Excalibur project (contains the expert calibration answers, and seed question realizations file contents). To recreate the files, insert the text from these two tables into text files respectively named "cleansed.dtt" and "cleansed.rls".

Table H-1: Excalibur "cleansed.dtt" project file containing the expert data

```
* CLASS ASCII OUTPUT FILE. NQ=   3   QU=   5  50  95
   1      A4    1  T1Car_Fastest UNI  1.00000E+0000  2.00000E+0000  3.00000E+0000
A4 (REuben, Proficient)
   1      A4    2   T1Car_Median UNI  8.00000E+0000  1.00000E+0001  1.20000E+0001
   1      A4    3  T1Car_Slowest UNI  2.50000E+0001  3.00000E+0001  5.00000E+0001
   1      A4    4  T4Car_Fastest UNI  6.00000E+0000  8.00000E+0000  1.00000E+0001
   1      A4    5   T4Car_Median UNI  1.70000E+0001  2.00000E+0001  2.50000E+0001
   1      A4    6  T4Car_Slowest UNI  4.00000E+0001  6.00000E+0001  8.00000E+0001
   1      A4    7 T1Chat_Fastest UNI  2.00000E+0000  3.00000E+0000  4.00000E+0000
   1      A4    8  T1Chat_Median UNI  1.00000E+0001  1.20000E+0001  1.60000E+0001
   1      A4    9 T1Chat_Slowest UNI  2.50000E+0001  3.00000E+0001  4.00000E+0001
   1      A4   10 T3Chat_Fastest UNI  1.20000E+0001  1.50000E+0001  1.90000E+0001
   1      A4   11  T3Chat_Median UNI  2.50000E+0001  2.80000E+0001  3.20000E+0001
   1      A4   12 T3Chat_Slowest UNI  4.50000E+0001  5.00000E+0001  6.50000E+0001
   2      B1    1  T1Car_Fastest UNI  1.00000E+0001  2.00000E+0001  2.50000E+0001
B1 (Competent)
   2      B1    2   T1Car_Median UNI  3.00000E+0001  4.50000E+0001  7.50000E+0001
   2      B1    3  T1Car_Slowest UNI  7.50000E+0001  9.00000E+0001  1.80000E+0002
   2      B1    4  T4Car_Fastest UNI  1.50000E+0001  2.00000E+0001  3.00000E+0001
   2      B1    5   T4Car_Median UNI  3.50000E+0001  4.50000E+0001  6.00000E+0001
   2      B1    6  T4Car_Slowest UNI  7.50000E+0001  9.00000E+0001  1.50000E+0002
   2      B1    7 T1Chat_Fastest UNI  7.00000E+0000  1.00000E+0001  1.50000E+0001
   2      B1    8  T1Chat_Median UNI  2.00000E+0001  2.50000E+0001  3.00000E+0001
   2      B1    9 T1Chat_Slowest UNI  3.00000E+0001  4.50000E+0001  6.00000E+0001
   2      B1   10 T3Chat_Fastest UNI  1.50000E+0001  2.00000E+0001  2.50000E+0001
   2      B1   11  T3Chat_Median UNI  3.00000E+0001  4.50000E+0001  5.00000E+0001
   2      B1   12 T3Chat_Slowest UNI  6.00000E+0001  9.00000E+0001  1.50000E+0002
   3      C2    1  T1Car_Fastest UNI  4.00000E+0000  5.00000E+0000  6.00000E+0000
C2 (Proficient)
   3      C2    2   T1Car_Median UNI  1.00000E+0001  1.20000E+0001  1.40000E+0001
   3      C2    3  T1Car_Slowest UNI  3.00000E+0001  6.00000E+0001  1.20000E+0002
   3      C2    4  T4Car_Fastest UNI  1.00000E+0001  1.20000E+0001  1.40000E+0001
   3      C2    5   T4Car_Median UNI  2.00000E+0001  2.40000E+0001  2.80000E+0001
   3      C2    6  T4Car_Slowest UNI  4.50000E+0001  8.00000E+0001  1.80000E+0002
   3      C2    7 T1Chat_Fastest UNI  1.00000E+0001  1.10000E+0001  1.20000E+0001
   3      C2    8  T1Chat_Median UNI  1.40000E+0001  1.60000E+0001  1.80000E+0001
   3      C2    9 T1Chat_Slowest UNI  2.00000E+0001  3.00000E+0001  6.00000E+0001
   3      C2   10 T3Chat_Fastest UNI  1.50000E+0001  1.80000E+0001  2.00000E+0001
   3      C2   11  T3Chat_Median UNI  2.50000E+0001  3.00000E+0001  4.00000E+0001
```

```
   3      C2   12 T3Chat_Slowest UNI  3.50000E+0001  4.50000E+0001  8.00000E+0001
   4      E5    1  T1Car_Fastest UNI  1.00000E+0000  5.00000E+0000  7.00000E+0000
E5 (Competent)
   4      E5    2   T1Car_Median UNI  5.00000E+0000  1.00000E+0001  2.50000E+0001
   4      E5    3  T1Car_Slowest UNI  2.00000E+0001  3.50000E+0001  4.00000E+0001
   4      E5    4  T4Car_Fastest UNI  1.00000E+0000  7.00000E+0000  1.50000E+0001
   4      E5    5   T4Car_Median UNI  1.00000E+0001  2.00000E+0001  3.00000E+0001
   4      E5    6  T4Car_Slowest UNI  3.00000E+0001  4.00000E+0001  4.50000E+0001
   4      E5    7 T1Chat_Fastest UNI  2.00000E+0000  1.00000E+0001  1.50000E+0001
   4      E5    8  T1Chat_Median UNI  5.00000E+0000  1.50000E+0001  2.00000E+0001
   4      E5    9 T1Chat_Slowest UNI  1.50000E+0001  2.50000E+0001  4.50000E+0001
   4      E5   10 T3Chat_Fastest UNI  1.00000E+0001  2.00000E+0001  2.50000E+0001
   4      E5   11  T3Chat_Median UNI  2.00000E+0001  4.00000E+0001  6.00000E+0001
   4      E5   12 T3Chat_Slowest UNI  3.00000E+0001  4.50000E+0001  9.00000E+0001
   5      F6    1  T1Car_Fastest UNI  5.00000E+0000  9.00000E+0000  1.00000E+0001
F6 (Proficient)
   5      F6    2   T1Car_Median UNI  5.00000E+0000  1.00000E+0001  2.00000E+0001
   5      F6    3  T1Car_Slowest UNI  2.00000E+0001  2.20000E+0001  3.00000E+0001
   5      F6    4  T4Car_Fastest UNI  1.00000E+0001  1.20000E+0001  2.00000E+0001
   5      F6    5   T4Car_Median UNI  1.50000E+0001  2.00000E+0001  3.00000E+0001
   5      F6    6  T4Car_Slowest UNI  2.00000E+0001  5.00000E+0001  6.00000E+0001
   5      F6    7 T1Chat_Fastest UNI  3.00000E+0001  4.00000E+0001  6.00000E+0001
   5      F6    8  T1Chat_Median UNI  4.00000E+0001  6.00000E+0001  1.20000E+0002
   5      F6    9 T1Chat_Slowest UNI  5.00000E+0001  1.00000E+0002  1.20000E+0002
   5      F6   10 T3Chat_Fastest UNI  4.00000E+0001  5.00000E+0001  7.00000E+0001
   5      F6   11  T3Chat_Median UNI  5.00000E+0001  7.00000E+0001  1.20000E+0002
   5      F6   12 T3Chat_Slowest UNI  6.00000E+0001  1.20000E+0002  1.50000E+0002
   6      G7    1  T1Car_Fastest UNI  3.00000E+0000  1.00000E+0001  1.20000E+0001
G7 (Competent)
   6      G7    2   T1Car_Median UNI  1.30000E+0001  1.50000E+0001  2.00000E+0001
   6      G7    3  T1Car_Slowest UNI  2.10000E+0001  3.00000E+0001  6.00000E+0001
   6      G7    4  T4Car_Fastest UNI  7.00000E+0000  1.50000E+0001  1.80000E+0001
   6      G7    5   T4Car_Median UNI  1.90000E+0001  2.50000E+0001  3.00000E+0001
   6      G7    6  T4Car_Slowest UNI  3.10000E+0001  4.50000E+0001  6.00000E+0001
   6      G7    7 T1Chat_Fastest UNI  3.00000E+0000  1.00000E+0001  1.50000E+0001
   6      G7    8  T1Chat_Median UNI  1.60000E+0001  2.00000E+0001  2.50000E+0001
   6      G7    9 T1Chat_Slowest UNI  2.60000E+0001  3.50000E+0001  4.50000E+0001
   6      G7   10 T3Chat_Fastest UNI  3.00000E+0000  1.00000E+0001  1.50000E+0001
   6      G7   11  T3Chat_Median UNI  1.60000E+0001  2.00000E+0001  2.50000E+0001
   6      G7   12 T3Chat_Slowest UNI  2.60000E+0001  3.00000E+0001  4.00000E+0001
```

Table H-2: Excalibur "cleansed.rls" file containing seed question realizations

```
   1   T1Car_Fastest  2.00000E+0000 UNI
   2   T1Car_Median  6.50000E+0000 UNI
   3   T1Car_Slowest  6.00000E+0001 UNI
   4   T4Car_Fastest  3.00000E+0000 UNI
   5   T4Car_Median  8.00000E+0000 UNI
```

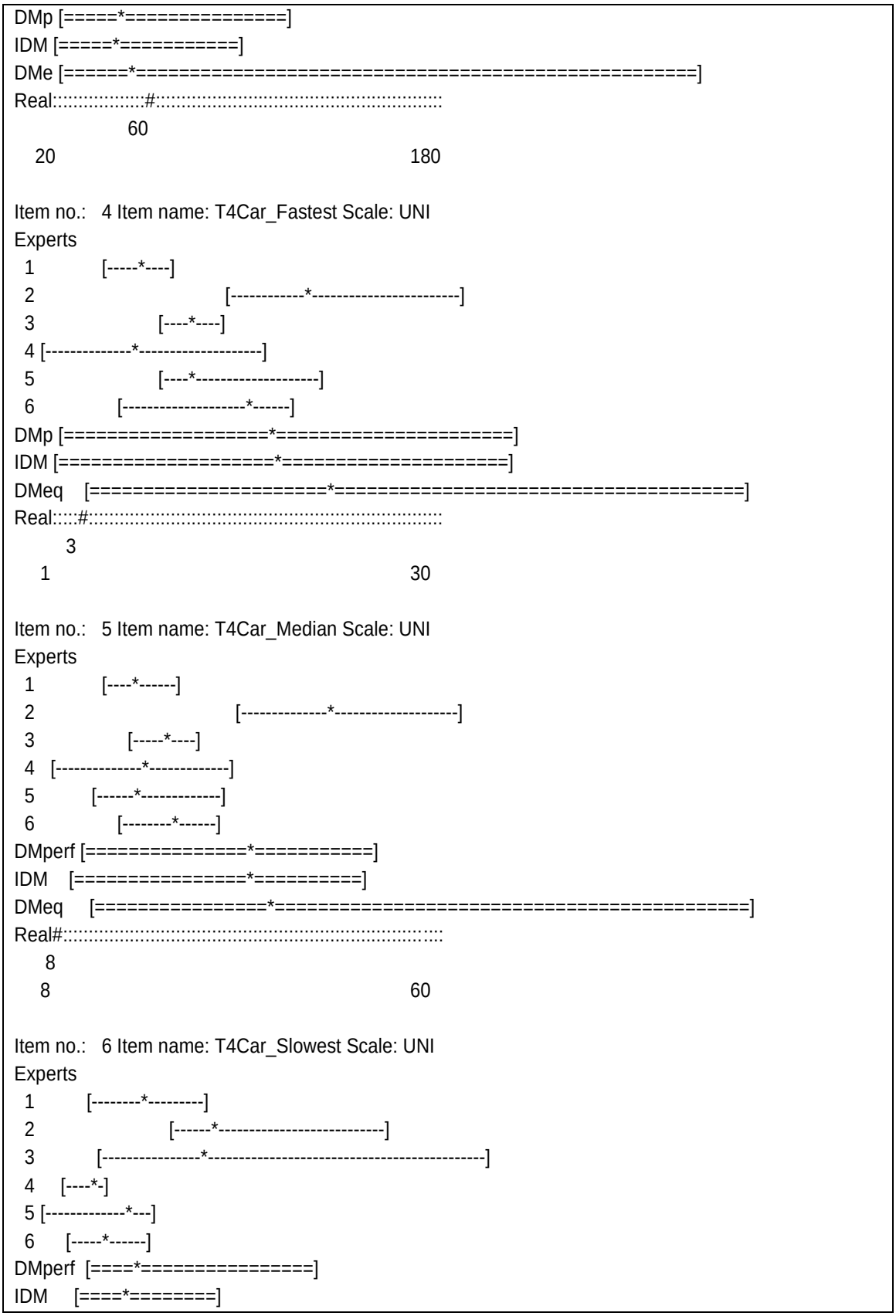
```
  6  T4Car_Slowest  2.80000E+0001 UNI
  7  T1Chat_Fastest  5.00000E+0000 UNI
  8  T1Chat_Median  2.00000E+0001 UNI
  9  T1Chat_Slowest  5.00000E+0001 UNI
 10  T3Chat_Fastest  4.00000E+0000 UNI
 11  T3Chat_Median  1.10000E+0001 UNI
 12  T3Chat_Slowest  2.00000E+0001 UNI
```

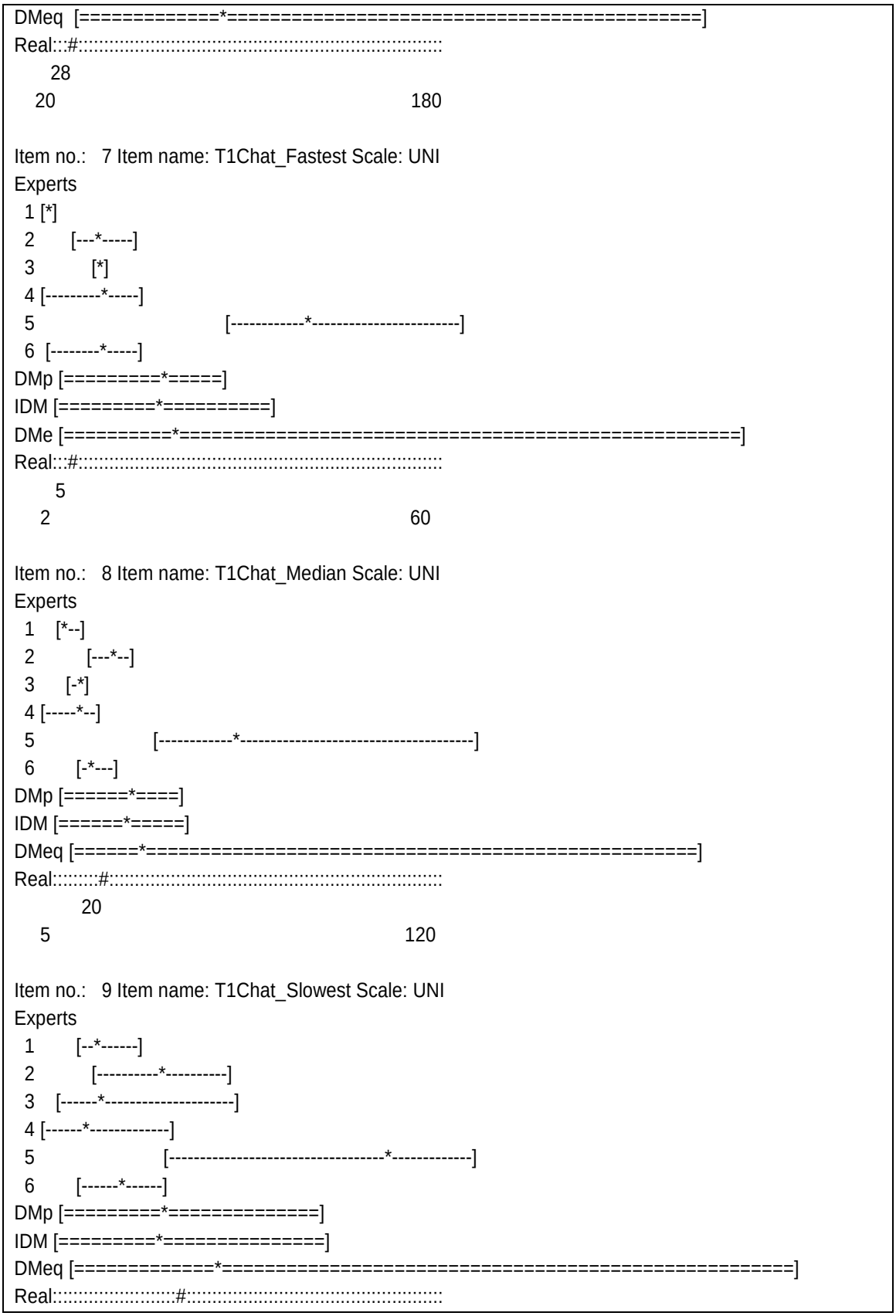
Table H-3: CCM range graph (items)

```
Item no.:   1 Item name: T1Car_Fastest Scale: UNI
Experts
 1 [--*--]
 2                        [------------------------------*--------------]
 3         [--*--]
 4 [-----------*-----]
 5            [----------*---]
 6      [----------------------*-----]
DMp [============*==================]
IDM [============*==================]
DMe [==============*==================================================]
Real:::#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
       2
   1                                                  25

Item no.:   2 Item name: T1Car_Median Scale: UNI
Experts
 1    [-*-]
 2                       [---------------*------------------------------]
 3     [-*-]
 4 [----*---------------]
 5 [----*----------]
 6        [-*-----]
DMp [=======*============]
IDM [========*===========]
DMe [=======*==========================================================]
Real:#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
    6.5
    5                                                  75

Item no.:   3 Item name: T1Car_Slowest Scale: UNI
Experts
 1   [-*---------]
 2                        [------*-----------------------------------------]
 3    [-------------*--------------------------]
 4 [------*-]
 5 *---]
 6 [---*-------------]
```


DMp [=====*===============]
IDM [=====*===========]
DMe [======*==]
Real:::::::::::::::::::#:::
60
20 180
Item no.: 4 Item name: T4Car_Fastest Scale: UNI
Experts
1 [-----*----]
2 [------------*------------------------]
3 [----*----]
4 [--------------*--------------------]
5 [----*--------------------]
6 [--------------------*------]
DMp [====================*=====================]
IDM [====================*====================]
DMeq [======================*====================================]
Real:::::#:::
3
1 30
Item no.: 5 Item name: T4Car_Median Scale: UNI
Experts
1 [----*------]
2 [--------------*--------------------]
3 [-----*----]
4 [--------------*-------------]
5 [------*-------------]
6 [--------*------]
DMperf [===============*==========]
IDM [================*==========]
DMeq [================*==]
Real#::
8
8 60
Item no.: 6 Item name: T4Car_Slowest Scale: UNI
Experts
1 [--------*---------]
2 [------*-------------------------]
3 [----------------*---]
4 [----*-]
5 [-------------*---]
6 [-----*------]
DMperf [====*================]
IDM [====*========]


DMeq [==========*==========]
Real:::#:::
28
20 180
Item no.: 7 Item name: T1Chat_Fastest Scale: UNI
Experts
1 [*]
2 [---*-----]
3 [*]
4 [---------*-----]
5 [----------*----------]
6 [--------*-----]
DMp [=========*=====]
IDM [=========*==========]
DMe [==========*==========]
Real:::#:::
5
2 60
Item no.: 8 Item name: T1Chat_Median Scale: UNI
Experts
1 [*--]
2 [---*--]
3 [-*]
4 [-----*--]
5 [----------*----------]
6 [-*---]
DMp [======*====]
IDM [======*=====]
DMeq [======*==========]
Real::::::::::#:::
20
5 120
Item no.: 9 Item name: T1Chat_Slowest Scale: UNI
Experts
1 [--*------]
2 [----------*----------]
3 [------*----------]
4 [------*----------]
5 [----------*----------]
6 [------*------]
DMp [=========*==========]
IDM [=========*==========]
DMeq [==========*==========]
Real::::::::::::::::::::::::::#:::

```
                    50
  15                                                120

Item no.:  10 Item name: T3Chat_Fastest Scale: UNI
Experts
 1          [--*---]
 2            [-----*----]
 3            [--*--]
 4       [-----------*----]
 5                                  [----------*----------------------]
 6 [------*-----]
DMpe [==============*========]
IDM  [==============*==========]
DMeq  [=============*================================================]
Real:#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
    4
   3                                                  70

Item no.:  11 Item name: T3Chat_Median Scale: UNI
Experts
 1         [-*--]
 2            [---------*--]
 3         [---*-----]
 4      [------------*-------------]
 5                    [-------------*---------------------------------]
 6   [--*--]
DMperf [===========*==================]
IDM    [=======*====================]
DMeq   [==========*==============================================]
Real#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
    11
   11                                                 120

Item no.:  12 Item name: T3Chat_Slowest Scale: UNI
Experts
 1             [--*-------]
 2                   [----------------*---------------------------------]
 3        [-----*-------------------]
 4     [--------*-------------------------]
 5                   [-----------------------------------*----------------]
 6   [-*-----]
DMperf [=======*============================]
IDM    [======*============================]
DMeq   [================*================================================]
Real#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
    20
   20                                                 150
```

Table H-4: Data results from simulating individual expert distributions

γ	**Expert 1**	**Expert 2**	**Expert 3**	**Expert 4**	**Expert 5**	**Expert 6**
148	[0, 6, 16, 20, 34, 37]	[0, 4, 12, 25, 49, 72]	[0, 19, 46, 73, 100, 113]	[0, 22, 45, 70, 85, 96]	[0, 19, 42, 69, 88, 100]	[0, 13, 30, 51, 60, 63]
55	[0, 3, 7, 9, 11, 13]	[0, 3, 6, 11, 19, 26]	[0, 3, 9, 18, 27, 35]	[0, 6, 17, 27, 34, 38]	[0, 4, 10, 19, 37, 48]	[0, 7, 19, 28, 34, 36]
20	[0, 1, 3, 5, 6, 7]	[0, 0, 1, 3, 7, 10]	[0, 1, 4, 9, 12, 14]	[0, 3, 6, 11, 13, 13]	[0, 1, 2, 7, 11, 16]	[0, 3, 8, 13, 14, 15]
7	[0, 0, 1, 1, 1, 2]	[0, 0, 0, 0, 0, 1]	[0, 0, 1, 2, 4, 4]	[0, 0, 2, 3, 3, 4]	[0, 0, 0, 2, 3, 5]	[0, 1, 3, 4, 5, 5]
3	[0, 0, 0, 0, 1, 1]	[0, 0, 0, 0, 0, 1]	[0, 0, 0, 1, 1, 2]	[0, 0, 0, 1, 1, 1]	[0, 0, 0, 0, 1, 2]	[0, 0, 0, 1, 2, 2]

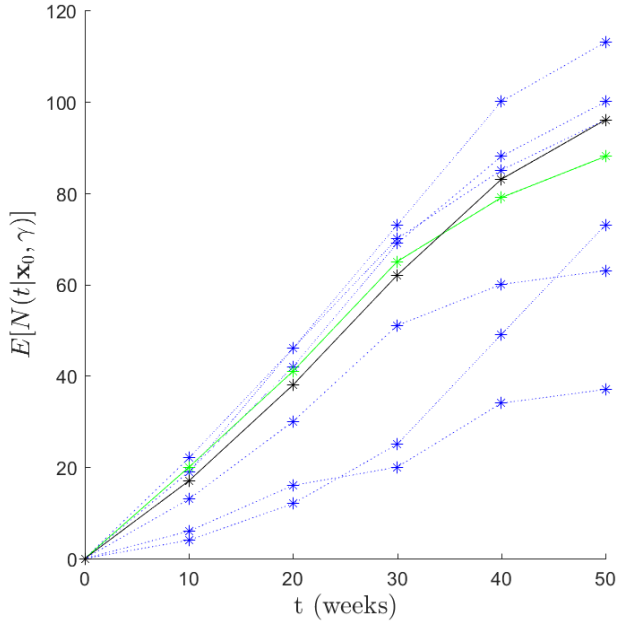


Figure H-1: Elicited cumulative discoveries over time, example 2

Baseline SR and SAP with $\gamma = 148$

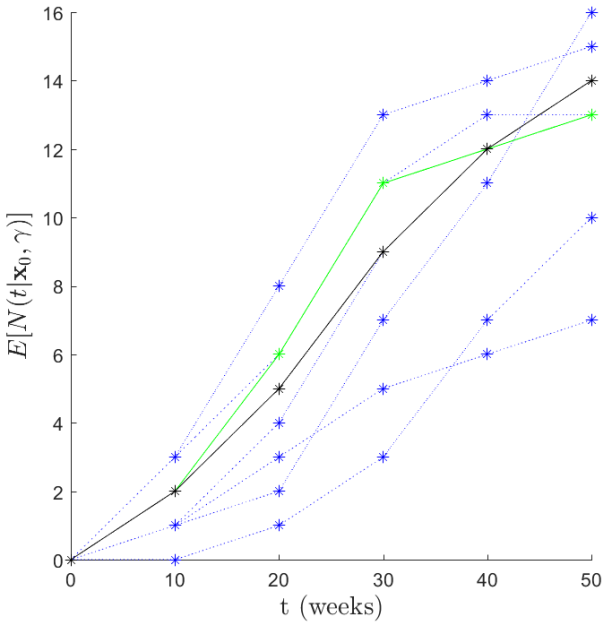


Figure H-2: Elicited cumulative discoveries over time, example 3

Baseline SR and SAP with $\gamma = 20$

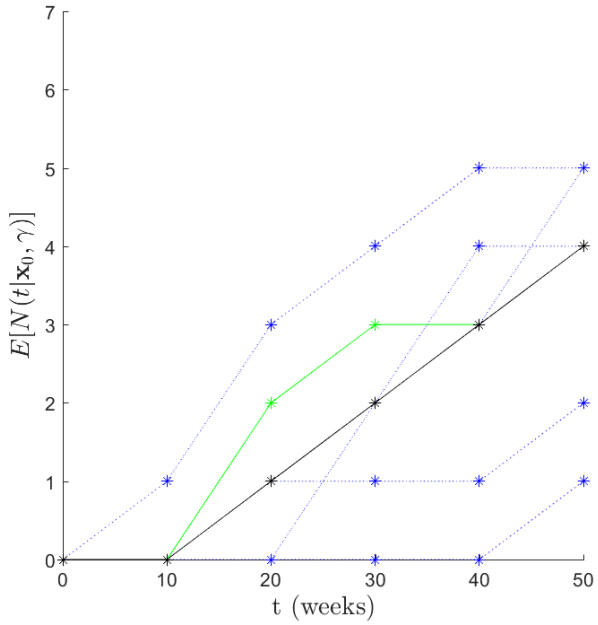


Figure H-3: Elicited cumulative discoveries over time, example 4

Baseline SR and SAP with $\gamma = 7$

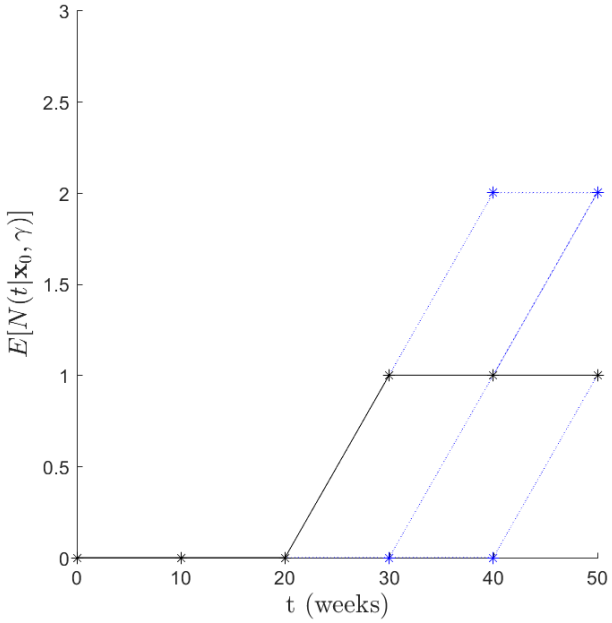


Figure H-4: Elicited cumulative discoveries over time, example 5

Baseline SR and SAP with $\gamma = 3$

Table H-5: Expert 1 $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
1	15	5
2	54	54
3	37	43
4	54	15
5	43	35
6	27	34
7	15	25
8	20	17
9	10	25
10	10	20
11	54	43
12	43	30
13	54	10
14	54	10
15	43	52
16	43	5
17	20	43
18	43	50
19	15	20

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
20	5	40
21	25	15
22	43	52
23	15	40
24	22	54
25	10	43
26	45	20
27	43	37
28	25	35
29	20	30
30	35	43
31	50	20
32	45	54
33	43	15
34	15	54
35	43	30
36	25	34
37	25	29
38	25	40
39	43	30
40	45	5
41	20	5
42	50	35
43	43	30
44	15	50
45	50	45
46	50	18
47	45	32
48	27	50
49	15	35
50	5	15

Table H-6: Expert 2 $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
1	1	1
2	20	20
3	1	1
4	3	0

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
5	6	0
6	1	1
7	3	1
8	1	1
9	1	1
10	1	1
11	20	1
12	1	1
13	1	1
14	3	0
15	1	6
16	1	1
17	1	1
18	1	3
19	1	1
20	1	3
21	1	1
22	1	4
23	1	1
24	0	2
25	0	1
26	1	1
27	1	3
28	3	1
29	1	1
30	1	1
31	20	1
32	1	4
33	4	1
34	0	3
35	1	0
36	1	1
37	3	1
38	1	1
39	1	1
40	1	1
41	1	1
42	4	1
43	1	1

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
44	0	2
45	6	2
46	1	1
47	1	1
48	1	1
49	1	1
50	1	1

Table H-7: Expert 3 $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
1	20	5
2	55	55
3	38	40
4	55	30
5	55	35
6	32	38
7	55	44
8	38	48
9	38	46
10	38	42
11	55	44
12	40	10
13	44	34
14	55	35
15	42	55
16	36	25
17	42	48
18	42	55
19	35	40
20	38	46
21	20	10
22	42	55
23	25	46
24	35	55
25	10	42
26	44	30
27	42	55
28	55	42

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
29	38	30
30	46	48
31	55	10
32	20	55
33	50	25
34	25	55
35	42	30
36	35	46
37	55	38
38	38	40
39	39	46
40	44	35
41	20	8
42	55	42
43	42	38
44	25	55
45	55	54
46	46	38
47	41	43
48	38	46
49	42	35
50	35	46

Table H-8: Expert 4 $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
1	25	17
2	46	46
3	30	36
4	45	12
5	42	25
6	20	27
7	50	33
8	26	40
9	20	38
10	20	36
11	47	35
12	33	15
13	45	10

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
14	38	20
15	35	45
16	20	3
17	38	38
18	30	45
19	20	42
20	28	38
21	25	15
22	35	48
23	15	42
24	20	41
25	15	35
26	37	5
27	36	44
28	45	32
29	45	20
30	43	38
31	50	13
32	23	40
33	45	5
34	28	40
35	35	27
36	5	49
37	48	23
38	23	26
39	25	45
40	38	25
41	23	15
42	43	35
43	35	26
44	27	45
45	47	42
46	39	5
47	30	37
48	29	40
49	43	26
50	20	30

Table H-9: Expert 5 $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
1	30	25
2	41	41
3	35	40
4	41	30
5	41	30
6	39	41
7	41	38
8	40	41
9	30	35
10	30	38
11	41	40
12	35	25
13	41	20
14	41	30
15	40	41
16	38	35
17	39	37
18	34	41
19	38	40
20	35	41
21	30	38
22	40	41
23	25	41
24	38	41
25	30	40
26	41	40
27	40	41
28	41	38
29	34	37
30	33	35
31	35	29
32	33	40
33	41	20
34	37	41
35	40	36
36	39	41
37	41	35
38	35	31

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
39	38	40
40	38	34
41	34	22
42	41	35
43	41	36
44	39	41
45	41	38
46	41	39
47	39	40
48	38	36
49	40	33
50	38	41

Table H-10: Expert 6 $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
1	20	10
2	45	45
3	20	25
4	18	5
5	32	18
6	22	25
7	48	36
8	18	36
9	18	25
10	19	25
11	45	25
12	26	5
13	27	5
14	35	10
15	25	32
16	22	10
17	32	28
18	21	36
19	19	24
20	20	30
21	20	12
22	26	42
23	15	30

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$
24	17	35
25	12	25
26	28	10
27	36	48
28	30	21
29	24	24
30	24	30
31	39	10
32	20	34
33	40	15
34	10	35
35	25	18
36	10	38
37	48	18
38	25	26
39	24	23
40	33	20
41	20	5
42	38	27
43	25	20
44	10	35
45	32	31
46	38	10
47	24	32
48	22	26
49	24	23
50	12	23

Table H-11: Aggregated $\boldsymbol{D}^1/\boldsymbol{D}^2$ dataset

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$	$E[N(\boldsymbol{x}_j^1)]/E[N(\boldsymbol{x}_j^2)]$
1	24.01902	15.74276	1.555711
2	45.65093	45.65093	0.999986
3	28.14729	33.93805	0.828908
4	40.18527	10.79509	3.694316
5	40.08617	23.67679	1.684755
6	20.2637	26.53371	0.766338
7	49.33727	33.31065	1.492998
8	24.53275	39.07961	0.627921

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$	$E[N(\boldsymbol{x}_j^1)]/E[N(\boldsymbol{x}_j^2)]$
9	19.56822	35.57079	0.562753
10	19.73612	33.93275	0.593354
11	46.47393	33.11505	1.538771
12	31.62369	13.25726	2.693827
13	41.68351	9.128993	4.622032
14	37.27857	18.21859	2.154976
15	33.11505	42.55517	0.774943
16	20.26105	4.247288	5.86536
17	36.7567	36.07979	1.024115
18	28.31254	43.20747	0.650953
19	19.75735	38.70815	0.533779
20	26.50129	36.43907	0.722763
21	24.01902	14.46705	1.660027
22	33.28295	46.69031	0.708239
23	14.93626	39.71821	0.385929
24	19.41511	39.74114	0.485474
25	14.43939	33.11505	0.435296
26	35.26741	5.906553	6.569472
27	35.78495	44.39927	0.804021
28	42.20007	29.96915	1.419362
29	41.16124	20.59419	2.028521
30	39.51258	36.41029	1.074544
31	47.91962	12.46137	3.952057
32	22.38098	38.76045	0.575787
33	43.88551	6.692993	7.885904
34	24.82116	38.92457	0.626466
35	33.11505	25.33871	1.302994
36	5.9039	46.82241	0.13712
37	47.69127	22.05049	2.187705
38	23.22579	25.85208	0.898909
39	24.71185	41.00925	0.641376
40	36.92195	24.02964	1.537393
41	22.38364	13.2493	1.944066
42	41.90371	33.43759	1.276258
43	33.11771	24.85794	1.327214
44	24.00347	43.03314	0.544295
45	44.20117	39.88458	1.116454
46	38.59241	5.9039	7.066642
47	28.8295	35.93635	0.802242

j	$E[N(\boldsymbol{x}_j^1)]$	$E[N(\boldsymbol{x}_j^2)]$	$E[N(\boldsymbol{x}_j^1)]/E[N(\boldsymbol{x}_j^2)]$
48	27.66805	37.38734	0.747978
49	39.53115	25.35368	1.545964
50	18.58205	28.66691	0.645159

Appendix I. Bayesian analysis results supplement

Table I-1: Parameter estimates for $\gamma = 148$ and release 19.0 in Mozilla Firefox

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
M_2	BM HPP	ζ	1.90435	1.89206	{0.000138499, 5.67833}	0.515286
M_3	GO NHPP	u	187.395	59.0254	{81.8025, 305.643}	18.1092
		ζ	0.0131889	0.00419253	{0.00570299, 0.0215399}	0.00119816
M_4	G's GO NHPP	u	101.064	32.0803	{42.5352, 164.332}	10.138
		ζ	0.00732708	0.00233355	{0.00294405, 0.011763}	0.000836691
		κ	1.44393	0.14459	{1.16911, 1.73161}	0.0489718
M_5	MO NHPP	ζ	2.49689	2.49048	{5.87851e-05, 7.49062}	0.769613
		κ	0.0067193	0.00676321	{4.72297e-08, 0.0203571}	0.00193899
M_6	Y's NHPP	u	99.5299	100.272	{0.0100736, 298.906}	28.5384
		ζ	0.0742642	0.0738009	{2.64318e-06, 0.220683}	0.0231361
M_7	Lin. Reg.	β_0	3.04964	3.16112	{-3.15585, 9.24125}	0.250262
		β_1	1.8293	1.41566	{-0.943836, 4.60849}	0.124291
		r	0.0100198	0.0100169	{1.20681e-08, 0.0300138}	0.000819074
M_8	Poly. (2) Reg.	β_0	-1.89476	3.16408	{-8.08998, 4.28038}	0.244606
		β_1	2.5723	3.16009	{-3.74058, 8.66951}	0.199561
		β_2	-0.0150341	0.0999884	{-0.211757, 0.180411}	0.0070982
		r	0.0099888	0.0099695	{8.06247e-12, 0.0298496}	0.000646257
M_9	Ver. GC	β_0	3.77167	2.40881	{4.85403e-05, 8.19299}	0.251175

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
		β_1	88.9144	9.99802	{69.3214, 108.512}	0.953848
		β_2	0.302104	0.214013	{2.60792e-08, 0.706185}	0.0171622
		r	0.0100037	0.0100118	{2.43171e-09, 0.0300193}	0.000792468
M_{10}	Gomp. GC	β_0	3.03053	2.14423	{9.77675e-06, 7.08285}	0.218156
		β_1	96.4835	9.99157	{76.9448, 116.11}	0.927344
		β_2	0.112332	0.074028	{1.647e-06, 0.249348}	0.00719125
		r	0.100254	0.0998826	{1.02429e-07, 0.299498}	0.0077083

The abbreviations are: Brooks-Motley (BM) HPP; Goel-Okumoto (GO) NHPP; Goel's generalized Goel-Okumoto (G's GO) NHPP; Musa-Okumoto (MO) NHPP; Yamada's (Y's) s-shaped NHPP; Linear Regression (Lin. Reg.); Polynomial degree-2 Regression (Poly. 2 Reg.); Verhulst growth curve (Ver. GC); and Gompertz growth curve (Gomp. GC). MCE for u parameters were higher than desired.

Table I-2: Parameter estimates for $\gamma = 55$ and release 3.0 in Firefox

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
M_1	KG NHPP	$\Theta(t)$	[0.000000, 6.027800, 17.031840, 27.080160, 33.085600, 37.070400]	[0.000000, 3.690763, 7.984636, 10.197649, 10.837332, 11.098539]	{[0.000000, 1.000000, 2.000000, 3.000000, 6.000000, 7.000000], [0.000000, 28.000000, 63.000000, 80.000000, 84.000000, 90.000000]}	[0.000000, 1.189605, 2.383160, 3.394593, 3.628923, 3.796131]
M_2	BM HPP	ζ	0.796452	0.795785	{1.7587e-05, 2.37428}	0.259332
M_3	GO NHPP	u	107.772	33.8688	{47.5972, 176.253}	11.4829
		ζ	0.00877336	0.00279959	{0.00352922, 0.0141498}	0.000945542
M_4	G's GO NHPP	u	39.3321	12.4191	{16.258, 63.3919}	3.5519

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
		ζ	0.00305623	0.000962727	{0.00133052, 0.00498584}	0.000315537
		κ	1.74222	0.174431	{1.40879, 2.08958}	0.0554559
M_5	MO NHPP	ζ	0.937989	0.937711	{3.14516e-06, 2.80968}	0.311366
		κ	0.0102893	0.0102454	{3.82993e-07, 0.0306584}	0.00305846
M_6	Y's NHPP	u	43.3352	43.3435	{0.000288372, 130.441}	14.4154
		ζ	0.0684338	0.0684184	{1.54801e-06, 0.206118}	0.0194884
M_7	Lin. Reg.	β_0	0.285796	0.0999927	{0.0898929, 0.481074}	0.00778284
		β_1	0.787747	0.316191	{0.166484, 1.40491}	0.0223514
		r	0.0100096	0.0100435	{1.76163e-09, 0.0300489}	0.000660982
M_8	Poly. (2) Reg.	β_0	-1.50312	3.15694	{-7.67715, 4.70245}	0.28002
		β_1	1.05698	0.999274	{-0.910365, 3.00458}	0.0859525
		β_2	-0.0053811	0.031681	{-0.0676132, 0.0566719}	0.00200192
		r	0.00999706	0.00999876	{3.38419e-09, 0.0300006}	0.000771382
M_9	Ver. GC	β_0	3.95764	2.47055	{1.61122e-05, 8.44872}	0.217449
		β_1	36.8719	9.98407	{17.0719, 56.1871}	0.936324
		β_2	0.308456	0.215956	{1.7487e-06, 0.71419}	0.0222386
		r	0.100151	0.100262	{5.40136e-07, 0.300181}	0.00756158
M_{10}	Gomp. GC	β_0	3.11741	2.1772	{2.31402e-05, 7.211}	0.212699
		β_1	39.7001	10.0037	{20.2444, 59.4474}	1.12737
		β_2	0.116568	0.0754584	{7.63723e-07, 0.25562}	0.00727568

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
		r	0.20034	0.199641	{1.11429e-06, 0.598951}	0.0121461

The abbreviation for M_1 is Kuo-Ghosh (KG) NHPP. MCE for u parameters were higher than desired.

Table I-3: Parameter estimates for $\gamma = 20$ and release 7.0 in Internet Explorer

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
M_1	KG NHPP	$\Theta(t)$	[0.000000, 1.988400, 5.976040, 10.955440, 11.955840, 12.956040]	[0.000000, 0.990951, 2.541142, 3.909535, 3.909550, 3.909537]	{[0.000000, 1.000000, 2.000000, 3.000000, 4.000000, 5.000000], [0.000000, 8.000000, 19.000000, 32.000000, 33.000000, 34.000000]}	[0.000000, 0.270776, 0.899399, 1.108174, 1.107041, 1.107266]
M_2	BM HPP	ζ	0.293406	0.292847	{3.40909e-06, 0.877421}	0.0981354
M_3	GO NHPP	u	28.0921	8.87189	{12.1255, 45.4265}	2.36709
		ζ	0.0133606	0.00419653	{0.00596626, 0.0218528}	0.0014306
M_4	G's GO NHPP	u	12.8544	4.07542	{5.57066, 21.0324}	1.39594
		ζ	0.000772052	0.000244812	{0.000338163, 0.00126554}	8.70455e-05
		κ	2.26488	0.227424	{1.81101, 2.70541}	0.0639444
M_5	MO NHPP	ζ	0.374934	0.371497	{3.52567e-06, 1.11543}	0.129374
		κ	0.0417417	0.0416585	{2.37833e-07, 0.125303}	0.0127847
M_6	Y's NHPP	u	14.9151	15.0238	{0.00059646, 44.4756}	4.02329
		ζ	0.0744225	0.0745153	{1.23397e-06, 0.222499}	0.02262
M_7	Lin. Reg.	β_0	0.190707	0.314757	{-0.429684, 0.804203}	0.0272628

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
		β_1	0.28532	0.315899	{-0.335769, 0.902439}	0.0262185
		r	0.0100044	0.010002	{2.37962e-08, 0.0299025}	0.000623679
M_8	Poly. (2) Reg.	β_0	-0.821621	0.315512	{-1.44726, -0.21071}	0.0294753
		β_1	0.436854	0.315196	{-0.176805, 1.05894}	0.0203836
		β_2	-0.00292361	0.0316208	{-0.0654649, 0.0583317}	0.00248241
		r	0.019995	0.0200114	{1.11543e-08, 0.0599926}	0.00174495
M_9	Ver. GC	β_0	4.34984	2.58574	{0.000337328, 8.98473}	0.299518
		β_1	14.7742	8.43645	{0.000147176, 29.7708}	0.921036
		β_2	0.32724	0.223696	{3.70786e-06, 0.745462}	0.0226203
		r	0.200521	0.200027	{2.63076e-06, 0.600968}	0.0170342
M_{10}	Gomp. GC	β_0	3.28012	2.24108	{3.19016e-05, 7.47349}	0.203159
		β_1	13.4547	3.15717	{7.29763, 19.6608}	0.259717
		β_2	0.294399	0.210473	{5.26501e-08, 0.693352}	0.0209017
		r	1.0009	0.317363	{0.433926, 1.63714}	0.0351743

MCE for u parameters were higher than desired.

Table I-4: Parameter estimates for $\gamma = 7$ and release 6.0 in Internet Explorer

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
M_2	BM HPP	ζ	0.0811011	0.0814098	{4.54761e-06, 0.242668}	0.0244476
M_3	GO NHPP	u	21.6041	6.82019	{9.38246, 35.3317}	2.12094
		ζ	0.00411051	0.00129883	{0.00172736, 0.00664401}	0.000396276
M_4	G's GO NHPP	u	3.53777	1.12163	{1.47252, 5.7333}	0.326998

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
		ζ	0.000277386	8.79921e-05	{0.000117161, 0.000449255}	2.90569e-05
		κ	2.6093	0.260337	{2.11379, 3.13249}	0.0990416
M_5	MO NHPP	ζ	0.0883894	0.0886055	{2.14478e-06, 0.264912}	0.0261189
		κ	0.0462264	0.046629	{2.46644e-06, 0.138948}	0.0146481
M_6	Y's NHPP	u	4.63214	4.60854	{7.33782e-06, 13.8064}	1.59473
		ζ	0.0642674	0.0637292	{5.73796e-06, 0.190843}	0.0184931
M_7	Lin. Reg.	β_0	-0.143732	0.315497	{-0.760106, 0.47695}	0.0274902
		β_1	0.0860139	0.100084	{-0.109691, 0.282213}	0.00966895
		r	0.0998866	0.100042	{4.37186e-07, 0.299013}	0.00930398
M_8	Poly. (2) Reg.	β_0	-0.321515	0.317144	{-0.942906, 0.296796}	0.0230562
		β_1	0.112306	0.316732	{-0.50643, 0.736586}	0.0234135
		β_2	-0.000519754	0.0315608	{-0.0619354, 0.0616987}	0.00164168
		r	0.0499535	0.0499819	{1.45511e-07, 0.149944}	0.00314521
M_9	Ver. GC	β_0	6.18004	2.94441	{0.432807, 11.5826}	0.338084
		β_1	4.21276	2.55302	{2.48763e-05, 8.82975}	0.273877
		β_2	0.404244	0.250151	{4.65271e-06, 0.858061}	0.0266764
		r	4.99786	0.706552	{3.63015, 6.3847}	0.0701047
M_{10}	Gomp. GC	β_0	3.70936	2.39198	{4.31958e-05, 8.10884}	0.227677
		β_1	4.34954	2.59235	{5.38872e-05, 9.01408}	0.282035
		β_2	0.855588	0.630181	{1.07046e-07, 2.06002}	0.0420978

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
		r	0.999379	0.317078	{0.426457, 1.62818}	0.0307138

MCE for u parameters were higher than desired.

Table I-5: Parameter estimates for $\gamma = 3$ and release 1.0 in Safari

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
M_2	BM HPP	ζ	0.0218818	0.0221474	{6.61026e-07, 0.0658275}	0.00714505
M_3	GO NHPP	u	226.115	71.2711	{95.4368, 364.27}	24.758
		ζ	9.67897e-05	3.08207e-05	{4.19618e-05, 0.000158123}	9.08867e-06
M_4	G's GO NHPP	u	1.06483	0.336199	{0.439912, 1.72196}	0.108866
		ζ	4.58073e-05	1.44243e-05	{1.99366e-05, 7.45078e-05}	4.60121e-06
		κ	2.99936	0.300999	{2.41297, 3.5876}	0.0917226
M_5	MO NHPP	ζ	0.0219053	0.0219004	{1.20508e-06, 0.065853}	0.00642742
		κ	2.15147e-08	2.16753e-08	{1.81547e-12, 6.46949e-08}	6.94768e-09
M_6	Y's NHPP	u	1.78719	1.7795	{1.13754e-05, 5.35493}	0.461697
		ζ	0.0432546	0.0425906	{5.1445e-06, 0.127832}	0.0128539
M_7	Lin. Reg.	β_0	-0.142697	0.316518	{-0.762834, 0.477898}	0.0268042
		β_1	0.0256433	0.0998722	{-0.172734, 0.218079}	0.00958084
		r	0.333755	0.333639	{1.76513e-06, 0.998664}	0.0260969
M_8	Poly. (2) Reg.	β_0	-0.143017	0.316388	{-0.767077, 0.471955}	0.0173232
		β_1	0.0259617	0.315534	{-0.595283, 0.641151}	0.0203851
		β_2	-1.70655e-05	0.0315658	{-0.0623345, 0.0613292}	0.00215606
		r	0.333081	0.333865	{1.39935e-06, 0.999749}	0.0199221

Model ID	Model name	Par. Name	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
M_9	Ver. GC	β_0	25.0993	3.15185	{18.8839, 31.2363}	0.23649
		β_1	1.29095	0.792451	{1.99663e-05, 2.7286}	0.0674396
		β_2	1.00071	0.315135	{0.385014, 1.62074}	0.0361741
		r	10.0006	3.16207	{4.29638, 16.315}	0.276676
M_{10}	Gomp. GC	β_0	21.4869	3.15568	{15.2451, 27.6196}	0.27243
		β_1	2.91854	2.09444	{7.55661e-05, 6.8917}	0.203661
		β_2	1.26949	0.786045	{1.73966e-05, 2.69845}	0.0810275
		r	10.004	1.41413	{7.27924, 12.7876}	0.0977371

MCE for u parameters were higher than desired. Despite the reasonable performance, parameter estimates for M_3 do not correspond with meaningful values (i.e., $u = 226$ is not realistic for $\gamma = 3$).

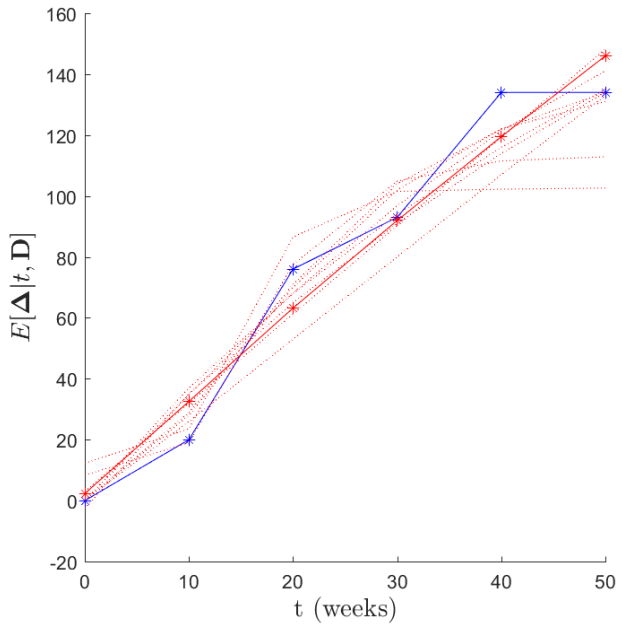


Figure I-1: Posterior-based averaged predictions for cumulative discoveries over time, example 2

The posterior-based predictions for the BMA used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_A}$ for $\gamma = 148$ and release 19.0 in Firefox to estimate $E[\boldsymbol{\Delta}|t, \boldsymbol{D}]$.

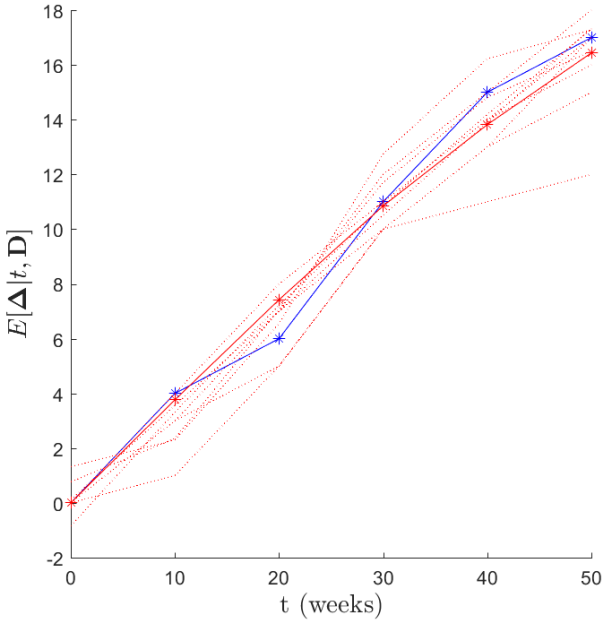


Figure I-2: Posterior-based averaged predictions for cumulative discoveries over time, example 3

The posterior-based predictions for the BMA used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_{\mathrm{A}}}$ for $\gamma = 20$ and release 7.0 in Internet Explorer to estimate $E[\boldsymbol{\Delta}|t, \boldsymbol{D}]$.

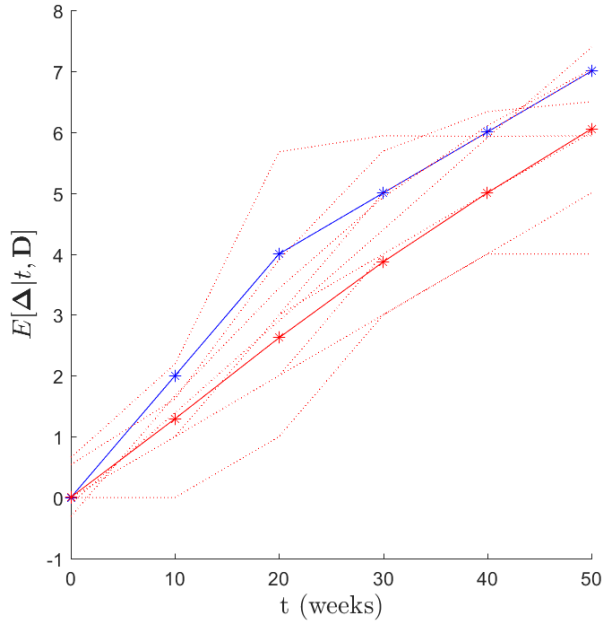


Figure I-3: Posterior-based averaged predictions for cumulative discoveries over time, example 4

The posterior-based predictions for the BMA used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_{\mathrm{A}}}$ for $\gamma = 7$ and release 6.0 in Internet Explorer to estimate $E[\boldsymbol{\Delta}|t, \boldsymbol{D}]$.

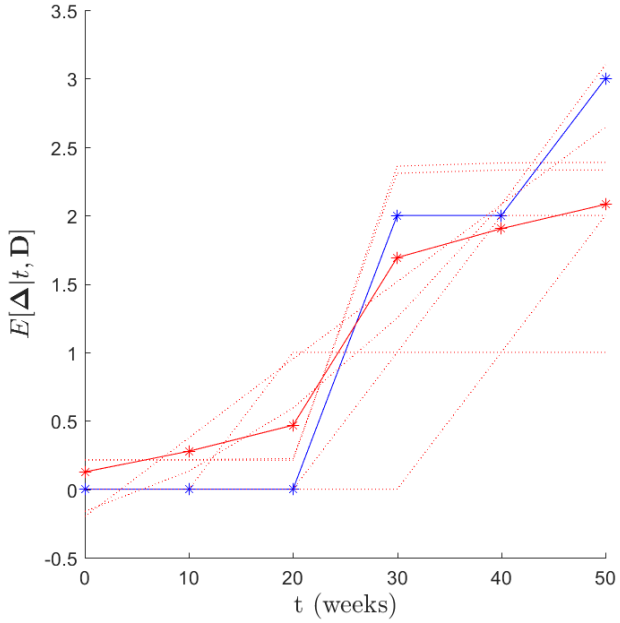


Figure I-4: Posterior-based averaged predictions for cumulative discoveries over time, example 5

The posterior-based predictions for the BMA used $\boldsymbol{D}_{\pi(\boldsymbol{\theta})} = \boldsymbol{D}_{w_A}$ for $\gamma = 3$ and release 1.0 in Safari to estimate $E[\boldsymbol{\Delta}|t, \boldsymbol{D}]$.

Table I-6: Model choice for $\gamma = 148$ and release 19.0 in Mozilla Firefox

Model ID	**Model name**	**Marginal log-likelihood (MCE)**	**BF**	**MSFE**
M_2	Brooks-Motley HPP	-58.1803 (0.0267)	5.79E+13	3.089e+02
M_3	Goel-Okumoto NHPP	-52.4672 (0.0278)	1.91E+11	1.860e+02
M_4	Goel's generalized Goel-Okumoto NHPP	-41.7287 (0.0425)	4.15E+06	1.373e+02
M_5	Musa-Okumoto logarithmic NHPP	-54.8959 (0.0323)	2.17E+12	1.999e+02
M_6	Yamada-Ohba-Osaki "s-shaped" NHPP	-43.1846 (0.0364)	1.78E+07	1.299e+02
M_7	Linear regression	-26.491 (0.0124)	1	2.785e+02
M_8	Polynomial (degree 2) regression	-27.8831 (0.0188)	4.0233	2.081e+02

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
M_9	Verhulst growth curve	-30.2633 (0.0163)	43.4809	6.616e+02
M_{10}	Gompertz growth curve	-31.1455 (0.0198)	105.064	3.301e+02
—	BMA	—	—	2.069e+02

Table I-7: Model choice for $\gamma = 55$ and release 3.0 in Firefox

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
M_1	Kuo-Ghosh NHPP	-18.0662 (0.0513)	1	2.815e+01
M_2	Brooks-Motley HPP	-19.744 (0.0254)	5.35362	6.055e+01
M_3	Goel-Okumoto NHPP	-21.7989 (0.0231)	41.7914	7.807e+01
M_4	Goel's generalized Goel-Okumoto NHPP	-19.7769 (0.0408)	5.53288	4.148e+01
M_5	Musa-Okumoto logarithmic NHPP	-20.9911 (0.0310)	18.6328	6.521e+01
M_6	Yamada-Ohba-Osaki "s-shaped" NHPP	-18.323 (0.0396)	1.29278	3.916e+01
M_7	Linear regression	-21.2616 (0.0111)	24.4199	8.103e+01
M_8	Polynomial (degree 2) regression	-22.5896 (0.0206)	92.147	4.698e+01
M_9	Verhulst growth curve	-21.5107 (0.0302)	31.3281	1.449e+01
M_{10}	Gompertz growth curve	-22.4914 (0.0242)	83.5288	2.291e+01
—	BMA	—	—	1.516e+01

Table I-8: Model choice for $\gamma = 20$ and release 7.0 in Internet Explorer

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
M_1	Kuo-Ghosh NHPP	-12.5299 (0.0214)	18.6699	1.259e+01
M_2	Brooks-Motley HPP	-10.2577 (0.0224)	1.92457	9.1123
M_3	Goel-Okumoto NHPP	-9.60304 (0.0149)	1	9.8602
M_4	Goel's generalized Goel-Okumoto NHPP	-11.6962 (0.0266)	8.11079	9.8458
M_5	Musa-Okumoto logarithmic NHPP	-10.2492 (0.0231)	1.90822	9.6968
M_6	Yamada-Ohba-Osaki "s-shaped" NHPP	-11.9052 (0.0300)	9.9956	1.043e+01
M_7	Linear regression	-19.0851 (0.0154)	13122.6	2.912e+01
M_8	Polynomial (degree 2) regression	-19.3586 (0.0201)	17249.6	1.670e+01
M_9	Verhulst growth curve	-15.756 (0.0285)	470.091	3.9136
M_{10}	Gompertz growth curve	-15.7177 (0.0249)	452.452	1.8025
—	BMA	—	—	3.1976

Table I-9: Model choice for $\gamma = 7$ and release 6.0 in Internet Explorer

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
M_2	Brooks-Motley HPP	-7.29059 (0.0207)	1.57313	4.036
M_3	Goel-Okumoto NHPP	-6.83752 (0.0151)	1	4.678
M_4	Goel's generalized Goel-Okumoto NHPP	-9.76274 (0.0243)	18.6384	5.886
M_5	Musa-Okumoto logarithmic NHPP	-7.11527 (0.0203)	1.32015	3.874

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
M_6	Yamada-Ohba-Osaki "s-shaped" NHPP	-8.30642 (0.0273)	4.34444	3.984
M_7	Linear regression	-12.7008 (0.0157)	351.891	3.476
M_8	Polynomial (degree 2) regression	-17.3804 (0.0212)	37907	6.637
M_9	Verhulst growth curve	-11.4067 (0.0227)	96.467	4.145e-1
M_{10}	Gompertz growth curve	-11.4651 (0.0226)	102.27	1.765
—	BMA	—	—	1.7767

Table I-10: Model choice for $\gamma = 3$ and release 1.0 in Safari

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
M_2	Brooks-Motley HPP	-6.41889 (0.0195)	3.18303	1.4471
M_3	Goel-Okumoto NHPP	-6.32939 (0.0134)	2.91053	1.5271
M_4	Goel's generalized Goel-Okumoto NHPP	-6.53378 (0.0189)	3.57058	1.5503
M_5	Musa-Okumoto logarithmic NHPP	-6.41225 (0.0185)	3.16198	1.4247
M_6	Yamada-Ohba-Osaki "s-shaped" NHPP	-6.81106 (0.0250)	4.71149	1.4015
M_7	Linear regression	-9.96643 (0.0152)	110.54	1.3186
M_8	Polynomial (degree 2) regression	-13.8879 (0.0232)	5579.56	1.3366
M_9	Verhulst growth curve	-5.26106 (0.0127)	1	2.150e-1
M_{10}	Gompertz growth curve	-6.11083 (0.0160)	2.33912	2.107e-1

Model ID	Model name	Marginal log-likelihood (MCE)	BF	MSFE
—	BMA	—	—	2.661e-1

Table I-11: Parameter estimates for the exponential scaling term

Parameter	Mean	SD	$\{HPD_{lo}, HPD_{hi}\}$	MCE
c_{41}	-0.80	0.13	{-1.02, -0.58}	0.0004
c_3	0.19	0.13	{-0.03, 0.40}	0.0004
c_{28}	1.58	0.17	{1.30, 1.87}	0.0005
c_{23}	0.94	0.16	{0.68, 1.20}	0.0005
c_{29}	0.87	0.26	{0.44, 1.30}	0.0008
c_{39}	2.13	0.17	{1.85, 2.42}	0.0005
c_1	0.17	0.19	{-0.15, 0.49}	0.0006
c_{18}	0.53	0.16	{0.27, 0.78}	0.0004
c_{20}	-0.15	0.19	{-0.47, 0.16}	0.0006
c_{17}	-0.72	0.14	{-0.95, -0.49}	0.0004
r	13.02	2.85	{8.72, 18.1}	0.009

Install MCMCBayes and from a MATLAB prompt, run the following code in Table I-12.

Table I-12: Example MATLAB session

Line	Code
1	MOL=42; %Using command encoding for less text
2	datStr=[62 55 68 58 59 66]; %Don't panic, we know the secret decoder key
3	msgStr=[41 59 59 -10 74 62 59 -10 ... 35 25 35 25 24 55 79 59 ... 73 -10 29 63 74 34 55 56 ... -10 70 72 69 64 59 57 74 ... -10 60 69 72 -10 58 59 74 ... 55 63 66 73 4 -10 -10 40 ... 27 75 56 -10 63 73 -10 69 ... 60 60 -10 74 69 -10 58 72 ... 63 68 65 -10 55 -10 74 55 ... 73 74 79 -10 31 38 23 -10 ... 68 69 77 2 -10 61 69 69 ... 58 -10 68 63 61 62 74 -9];
4	cmdStr=[70 66 55 79 59 72 10 19 ... -10 55 75 58 63 69 70 66 ... 55 79 59 72 -2 79 2 -10 ... 28 73 -1 17 -10 70 66 55 ...

	79 -2 70 66 55 79 59 72 ... -1 17];	
5	dataVar=char(datStr+MOL);	%Decode it (Obviously we use the answer to life, the Universe, well, everything)
6	eval(sprintf('load "%s"',dataVar))	%Load the data
7	eval(char(cmdStr+MOL));	%Run the demo
8	h=waitbar(0,char(msgStr+MOL));	%Statusbar for progress and a descriptive message
9	for i=1:8; pause(1); waitbar(i/8); end; close(h);	%Final loop