Biochemical Reaction Networks from a Systems Biology Perspective

By

Erin Michelle Shockley

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Chemical and Physical Biology

May 10, 2019

Nashville, Tennessee

Approved:

Vito Quaranta, M.D.

Chris Fonnesbeck, Ph.D.

Todd Giorgio, Ph.D.

William Holmes, Ph.D.

Alissa Weaver, M.D., Ph.D.

To my Oma, who was a survivor

ACKNOWLEDGMENTS

Portions of this dissertation are reprinted from:

- reference [1] as allowed by the author license granted by *PNAS*.
- reference [2], as allowed by the author license granted by *Bioinformatics*. Content was originally published in *Bioinformatics*, published by Oxford University Press.

TABLE OF CONTENTS

Page

DEDICATION ii

ACKNOWLEDGMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

Chapter

1 Introduction 1

1.1 General Background 1

1.2 Introduction to Model Calibration 1

1.3 Introduction to Model Flux Analysis 3

1.4 Introduction to Model Sensitivity Analysis 4

1.5 Introduction to Information Theory 5

2 The COX-2 Reaction Model (CORM) 6

2.1 Biological Background 6

2.2 Model Scope and Implementation 7

2.3 Detailed Model Description 8

3 Advances in Model Calibration 11

3.1 Specific Advances from this Work 11

3.2 Background on PyDREAM Algorithm and Specific Implementation 12

3.2.1 Introduction to Bayesian Inference 12

3.2.2 Introduction to Markov Chain Monte Carlo (MCMC) 13

3.2.3 The DREAM Algorithm and Implementation in PyDREAM 14

3.2.4 PyDREAM Validation 19

3.2.4.1 10D Bimodal Mixture Model 19

3.2.4.2 200D Multivariate Normal 19

3.3 Calibration of CORM with PyDREAM 19

3.3.1 Available Calibration Constraints 22

3.3.1.1 Experimental Data and Experimentally Measured Kinetic Rates . 22

3.3.1.2 Thermodynamic Constraints 22

3.3.1.3 Kinetic Parameter Prior Distributions 24

3.3.2 Calibration Results 25

3.3.3 Insights into COX-2 Catalysis 28

3.4 Calibrating a Larger Model with PyDREAM 30

4 Advances in Model Flux Analysis 39
4.1 Specific Advances and Problems Solved 39
4.2 Calculating Network Pathway Flux 40
4.3 Applying Pathway Flux Analysis to CORM 43
5 An Analysis of Information Transfer in the Multi-Input Multi-Output COX-2 Network 54
5.1 Motivation 54
5.2 Pathway Entropy is Dynamic Across Input Concentrations 55
5.3 Input Output Behavior in CORM 58
5.4 Channel Capacity from Substrates to Products 59
5.5 COX-2 Integrates Information from Both AA and 2-AG 63
5.6 Information Flow is Dictated by Substrate Concentration 67
6 Perturbing the COX-2 Network: A Sensitivity Analysis of CORM 72
6.1 Motivation 72
6.2 Introduction to Sensitivity Analysis with RS-HDMR 73
6.3 Constructing an HDMR Meta Model of CORM 75
6.4 HDMR Based Sensitivity Indices for CORM 80
6.5 CORM Sensitivity to Perturbations at Different Substrate Levels 93
7 High Performance Computing with Amazon Web Services: A Case Study 98
7.1 Introduction 98
7.1.1 Overview of the protocol 99
7.2 Materials 100
7.3 Protocol 100
7.4 Troubleshooting 111
8 Discussion and Future Directions 113
8.1 Discussion 113
8.1.1 COX-2 as an Allosteric Enzyme 113
8.1.2 Role of COX-2 Allostery *In Vivo* 114
8.1.3 Relationship Between AA and 2-AG *In Vivo* 115
8.1.4 Ensemble Model Calibration 115
8.2 Future Directions 119
8.2.1 PyDREAM for Larger Models 119
8.2.2 The COX-2 Reaction Model 120
BIBLIOGRAPHY 122

APPENDICES 144

LIST OF TABLES

Table		Page
2.1	CORM Species	9
2.2	CORM Reactions	10
3.1	EARM Rate Constant Constraints	30
3.2	Experimentally Measured Parameters from Literature Used for EARM Calibration	34
3.3	Association Rate Parameters Predicted with TransComp and Used for EARM Calibration. All units are M^{-1} s^{-1}.	35
6.1	CORM HDMR Variable Names	77
7.1	Troubleshooting guidance for using AWS Batch.	112

LIST OF FIGURES

Figure Page

2.1 COX-2 Reaction Model (CORM) 8

3.1 Schematic representation of PyDREAM workflow. The experimental data, model setup, prior parameter distribution, and likelihood function are defined as algorithmic input (top). This serves as input to the technical heart of PyDREAM (middle) which uses differential evolution for chain evolution, with a Metropolis selection rule to decide whether to accept/reject candidate points. These sampled chain trajectories (bottom) are then returned to the user and used to summarize the marginal posterior parameter distributions. 18

3.2 PyDREAM samples for a 10D bimodal test case. Each subfigure depicts a parameter dimension. Samples drawn using PyDREAM are shaded while the true distribution is indicated by a black line. Sampled values are in arbitrary units. 20

3.3 PyDREAM samples for a 200D multivariate normal test case. Each subfigure depicts one of ten representative parameter dimensions. Samples drawn using PyDREAM are shaded while the true distribution is indicated by a black line. Sampled values are in arbitrary units. 21

3.4 Thermodynamic Cycles in CORM 23

3.5 Prior Distributions for Fitted Parameters 25

3.6 Simulated model PG and PGG values generated using the most probable parameter vector obtained during calibration at several concentrations of AA and 2-AG. Experimental data is shown in black and simulated results in red. . . . 26

3.7 Distributions of parameter values for each fitted kinetic rate in CORM. . . . 27

3.8 Distribution of catalytic rate (in s^{-1}) when AA is bound in both the catalytic site and allosteric site of COX-2. The experimentally measured catalytic rate for the enzyme in the absence of any allosteric modulator is indicated by the red line. 28

3.9 Distribution of catalytic rate (in s^{-1}) when AA is bound in the catalytic site and 2-AG in the allosteric site of COX-2. The experimentally measured catalytic rate for the enzyme in the absence of any allosteric modulator is indicated by the red line. 29

3.10 Distribution of catalytic rate (in s^{-1}) when 2-AG is bound in the catalytic site and AA in the allosteric site of COX-2. The experimentally measured catalytic rate for the enzyme in the absence of any allosteric modulator is indicated by the red line. 29
3.11 Uncoverged distributions of MCMC draws for a single parameter in the EARM model. Different colors indicate different chains run in parallel, which should reflect similar distributions for convergence. 31
3.12 The process used for selecting parameter priors for EARM calibration, using either experimental data or computational prediction. 35
3.13 Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The normal prior used was derived from experimental measurements in the literature. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence. 36
3.14 Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The calibrated parameter values are unchanged from the uniform prior. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence. 36
3.15 Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The original parameter prior was uniform across the range of values. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence. 37
3.16 Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The original parameter prior was uniform across the range of values. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence. 37
3.17 Unconverged (by Gelman-Rubin metric) distribution of MCMC draws for a single parameter in the EARM model (bottom) and the corresponding traceplot for that parameter and other parameters, some of which had reached convergence (top). Parameter values are log transformed. The original parameter prior was uniform across the range of values. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence. 38

4.1 An overview of the method for calculating network pathway flux using a graph theoretic analysis to determine network paths and an integrator to return the integrated flux through specific chemical reactions. 41
4.2 Calculating paths between reactants and products. Edges are directed towards the flow of the net integrated reaction flux (for bidirectional reactions) or the integrated forward reaction flux (for unidirectional reactions). Edge labels indicate the probability of selecting that reaction path into a node, calculated as the proportion of total flux leaving the node that came from a given reaction. . 42
4.3 Possible paths between COX-2 and PG in CORM. 44
4.4 Possible paths from COX-2 to PGG in CORM. 44
4.5 Dominant PG Production Paths in CORM. Colors correspond to path fluxes in Fig. 4.7*A*. (*D*) Dominant PG-G Production Paths in CORM. Colors correspond to path fluxes in Fig. 4.7*B*. 45
4.6 Dominant PG-G Production Paths in CORM. Colors correspond to path fluxes in Fig. 4.7*B*. 45
4.7 Concentration-dependent PG and PG-G production paths. (*A*) Dominant Reaction paths for PG Production Vary with AA and 2-AG Concentration. Each individual plot depicts the amount of flux through each path in 2.1*C* for a given concentration of 2-AG across varying concentrations of AA. Colors correspond to labeled paths in Fig. 4.5. The error bars in each plot indicates the flux variation resulting from inferred kinetic rates. (*B*) Dominant Mechanisms of PG-G Production Vary with AA Concentration. Each individual plot is at a given concentration of 2-AG. In all plots AA increases from left to right at concentrations of 0.5, 1, 2, 4, 8 and 16 μM in *A* and 0, 0.5, 1, 2, 4, 8, and 16 μM in *B*. Colors correspond to labeled paths in Fig. 4.6 The error bars in each plot indicates the flux variation from inferred kinetic rates. 48
4.8 Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PG from the path COX-2 $\rightarrow$ COX-2:AA $\rightarrow$ PG at different starting AA and 2-AG concentrations. 49
4.9 Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PG from the path COX-2 $\rightarrow$ COX-2:AA $\rightarrow$ 2-AG:COX-2:AA $\rightarrow$ PG at different starting AA and 2-AG concentrations. . . . 50
4.10 Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PG from the path COX-2 $\rightarrow$ COX-2:AA $\rightarrow$ AA:COX-2:AA $\rightarrow$ PG at different starting AA and 2-AG concentrations. . . . 51
4.11 Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PGG from the path COX-2 $\rightarrow$ COX-2:2-AG $\rightarrow$ PGG at different starting AA and 2-AG concentrations. 52

4.12 Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PGG from the path COX-2 $\rightarrow$ COX-2:2-AG $\rightarrow$ AA:COX-2:2-AG $\rightarrow$ PGG at different starting AA and 2-AG concentrations. . 53

5.1 Pathway entropy within CORM. (*A*) Pathway Entropy for Production of PG. The intensity indicates the pathway entropy in units of bits. (*B*) Pathway Entropy for Production of PG-G. Units are the same as in *A*. 57

5.2 Input vs output plots for substrates and products in CORM. (*A*) Input vs Output plots for AA to PG. 2-AG varies randomly. All concentrations are measured at steady-state (10 seconds). (*B*) Input vs Output plots for 2-AG to PG-G. AA varies randomly. All concentrations are measured at steady-state (10 seconds). 59

5.3 Estimated channel capacities from substrates to intermediates or products in CORM. (*A*) Estimated Channel Capacities from Input to intermediates and final products within CORM when levels of AA and 2-AG are strongly correlated (Pearson correlation coefficient = 1). Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration.(*B*) Estimated Channel Capacities from AA to intermediates and final products within CORM when AA and 2-AG are varied independently. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration.(*C*) Estimated Channel Capacities from 2-AG to intermediates and final products within CORM when AA and 2-AG are varied independently. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration. 62

5.4 Distributions of response entropy under different input correlation and signal/response pairs. While the response entropy spans a similar range with both independent and correlated inputs, the transfer efficiency increases with correlation. 64

5.5 Sum of channel capacity from AA and 2-AG to intermediates and final outputs when inputs are semi-correlated (Pearson correlation coefficient = .5). Distributions arise from uncertainty in calibrated parameter values. 65

5.6 Channel capacity from AA and 2-AG to intermediates and final outputs when inputs are present in a 2 to 1 AA to 2-AG ratio. Distributions arise from uncertainty in calibrated parameter values. 66

5.7 Effect of substrate level on estimated channel capacities between substrates and products in CORM. (*A*) Total Estimated Channel Capacity from AA and 2-AG combined to products across regions of substrate space. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration. (*B*) Estimated Channel Capacities from input to products when levels of AA and 2-AG are perfectly correlated across regions of substrate space. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration. 68
5.8 Sum of channel capacity from AA and 2-AG to intermediates when inputs are varied independently in different regions of substrate space. Distributions arise from uncertainty in calibrated parameter values. 69
5.9 Channel capacity from AA and 2-AG to intermediates when inputs are varied correlated (Pearson correlation coefficient = 1) in different regions of substrate space. Distributions arise from uncertainty in calibrated parameter values. . . 70
5.10 Left: channel capacity between independent inputs and PG intermediates at different substrate levels. Right: absolute flux between independent inputs and PG intermediates at different substrate levels. 70
5.11 Left: channel capacity between independent inputs and PGG intermediates at different substrate levels. Right: absolute flux between independent inputs and PGG intermediates at different substrate levels. 71

6.1 Prediction accuracy and error for a second order HDMR model of CORM simulations. In the outputs column, (c) and (a) denote the catalytic and allosteric subunits of COX-2, and 0 indicates that either site is unbound. 78
6.2 Original CORM simulation values versus values calculated from a second order HDMR model trained to the CORM. 79
6.3 Heatmap of first order S_i^a sensitivity indices calculated for all i parameter in CORM. Each row represents a model parameter (input) and each column an output (PG, PGG, or the ratio of the two). Lighter values indicate stronger sensitivity of the output to variation in an input. 80
6.4 Heatmap of first order S_i^b sensitivity indices calculated for all i parameter in CORM. Each row represents a model parameter (input) and each column an output (PG, PGG, or the ratio of the two). Color indicates either positive or negative error in estimates induced by random correlation in the data. 81
6.5 Heatmap of first order S_i sensitivity indices calculated for all i parameter in CORM. Each row represents a model parameter (input) and each column an output (PG, PGG, or the ratio of the two). Lighter values indicate stronger sensitivity of the output to variation in an input. 82

6.6 Heatmap of S_i^a sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs. 83
6.7 Heatmap of S_i^b sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the error in an estimated sensitivity value calculated for PGs to a change in the row and column model parameters. Color indicates either either positive or negative error in estimates induced by random correlation in the data. 84
6.8 Heatmap of S_i sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs. 85
6.9 Heatmap of S_i^a sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs. 86
6.10 Heatmap of S_i^b sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the error in an estimated sensitivity value calculated for PGGs to a change in the row and column model parameters. Color indicates either either positive or negative error in estimates induced by random correlation in the data. 87
6.11 Heatmap of S_i sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs. 88
6.12 Heatmap of S_i^a sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of the PG to PGG ratio to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs. 89
6.13 Heatmap of S_i^b sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the error in an estimated sensitivity value calculated for the ratio of PGs to PGGs to a change in the row and column model parameters. Color indicates either either positive or negative error in estimates induced by random correlation in the data. 90

6.14 Heatmap of S_i sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of the ratio of PGs to PGGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs. 91
6.15 HDMR functions for relationships between the most sensitive parameters in CORM and the ratio of PGs to PGGs. From left to right, the parameters are kcat_AG1, KD_AG_1, and KD_AA_1. 93
6.16 Levels of AA and 2-AG Used to Generate Data for Sensitivity Analysis . . . 94
6.17 HDMR Model Accuracy for Each Region of Substrate Space 94
6.18 First order sensitivity indices for PG in each region of substrate space. Each row represents sensitivity to perturbations in a particular model parameter, and each column is a different region of substrate space. Lighter colors indicate higher sensitivity. 95
6.19 First order sensitivity indices for PGG in each region of substrate space. Each row represents sensitivity to perturbations in a particular model parameter, and each column is a different region of substrate space. Lighter colors indicate higher sensitivity. 96

8.1 Predicted concentration of CORM species ten seconds after mixing as a function of AA concentration. Darker lines indicate the prediction generated from the most probable parameter vector in the ensemble and the shaded region encompasses predictions based on the entire ensemble of parameters. 116
8.2 A kernel density estimate of the joint parameter probability density for two kinetic parameters in CORM, indicating a nonlinear relationship between the variables. 118

Chapter 1

Introduction

1.1 General Background

Computational modeling of biochemical systems at the level of signaling networks has provided insights into the origins of biochemical heterogeneity [3, 4], the plausibility of different mechanistic hypotheses about biological processes [5, 6], and the efficacy of different drug perturbations [7]. Deriving useful insights from mechanistic models poses a number of challenges related to model creation, calibration, and analysis.

Many of the challenges related to model creation have been ameliorated by the advent of rule-based modeling. Instead of enumerating all possible model species, Rule-based modeling defines valid interactions as a set of rules and then determining the species that can form given the rule set. This avoids directly tackling the combinatorial complexity present in many biochemical systems, while still faithfully representing the known system interactions [8]. The rule-based approach has been used to create models containing as many as three billion species [9]; when combined with network-free simulation methods [10], the number of model species may theoretically be infinite. Numerous modeling packages [11, 12] simplify the creation of transparent, reusable, extensible rule-based models.

Rigorous, efficient methods for model calibration and analysis have lagged behind those for model creation. The next section features a discussion of model calibration in systems biology.

1.2 Introduction to Model Calibration

Model calibration is the process of quantifying the fit of a model to experimental data, and iteratively improving that fit by modifying kinetic rate constants, initial conditions, or model interactions. Uncertainty in all three components (kinetic rate constants, initial conditions, and interactions) complicates the process.

The model components to be modified, the means of quantifying data fit, the algorithm for improving that fit, and the quantification of uncertainty in final fitted values have all evolved with the systems biology field. One of the first mechanistic systems biology models simply compared simulated outputs to experimental data by eye, varying only model interactions (manually) to determine the effect on model output [13]. The kinetic rate constants were rough estimates based on the type of biological interaction, and the initial conditions were known.

The use of computational algorithms to adjust model elements iteratively and optimize the fit to experimental data has become standard as the rise in available computational power has made efficient optimization possible. These algorithms require an *objective function* which quantifies the fit of simulated output to experimental data. Systems biology optimization problems are frequently non-convex with many minima [14], stymieing attempts to use highly efficient *local optimization* (also called *determistic optimization*) methods, which can become stuck in local minima. One early proposed solution to this was the multi-start method, in which multiple local solutions obtained from different start locations were compared to (hopefully) find the global minima [15]. Because this quickly becomes inefficient as the same minima are repeatedly discovered, the more computationally intense but rigorous *global optimization* (or *stochastic optimization*) methods became preferred. *Hybrid optimization* methods, which combine an initial global search followed by a local search, were developed to capitalize on the advantages of both techniques. Global optimization algorithms utilized in systems biology have included the DIRECT algorithm [16], simulated annealing [17, 18], genetic algorithms [19], evolution strategy [20], differential evolution [21] and particle swarm optimization [22]. Hybrid methods have also been used [23].

If model predictions are desired, optimizing the fit of a model to experimental data is only a first step in the model calibration process. Because model predictions depend on model parameters, the uncertainty in those parameters must be quantified lest the predictions be of unknown confidence and thus meaningless. There are two main approaches to quantifying model parameter uncertainty [24]. In the first, parameter uncertainty is approached in terms of *identifiability*, and the ultimate goal is to calculate confidence intervals for unknown parameters given the experimental data. If the available experimental data is not sufficiently informative, some parameters will be unidentifiable, and their confidence intervals infinite. Any model predictions that depend on these unidentifiable parameters are then not determined (*non-observable*) and likely meaningless. This can be ameliorated by targeted collection of experimental data to produce identifiable calibrated parameters and reliable model predictions. The second approach is to use Bayesian inference methods to specify model parameter prior probability distributions, probability distributions that specify what is known about a model parameter before calibration, and then infer posterior probability distributions given the priors and a likelihood function that incorporates experimental data (see Chapter 3, section 2.1 for an introduction to Bayesian inference). Typically model calibration is then performed using a Markov Chain Monte Carlo (MCMC) walk (see Chapter 3, section 2.2 for an introduction to MCMC). Model parameter prior distributions may include constraints such as those imposed by biophysical limitations (for example, diffusion limits). In many cases, both approaches will give similar results [24]. The first is likely to be more computationally efficient; however, the second can be successful in cases where the available experimental data is sparse.

1.3 Introduction to Model Flux Analysis

The analysis of flux through a mathematical representation of a biological system is one approach to derive non-trivial insights into the system. One of the most common techniques applied is called flux balance analysis (FBA). In FBA, linear programming is used to solve for a distribution of fluxes for system reactions at steady state, given any constraints on the reaction rates[25]. Specific points in this constrained space can then be selected to optimize a given objective function (e.g. metabolic growth rate). An alternate approach, pathway flux analysis, is presented in this work. In this method, described in more detail in section 4.2, reaction fluxes are directly calculated after parameter calibration, and this information is used to determine the path by which a particular molecule is produced in the system. Because this does not require the system to be at steady state, it can uncover system signal and product dynamics at any simulation point and is more widely applicable than FBA.

1.4 Introduction to Model Sensitivity Analysis

In sensitivity analysis, the degree of response to each input is calculated for each output of interest. This can be useful both as a post-calibration analysis technique and as a prelude to model calibration, since any inputs that have no effect on an output of interest both alone and in tandem with other inputs may be safely fixed at a given value, decreasing the dimension of the calibration problem and saving computational time. Like optimization, sensitivity analysis also comes in two varieties, *local* and *global*. Local methods produce estimates of sensitivity at a particular point (or set of points) in parameter space [26]. Mathematically, a local sensitivity coefficient is the partial derivative of an output, y, with respect to a particular input parameter, p:

$$S_i = \frac{\delta y}{\delta p} = \lim_{p \to 0} \frac{y(p+\Delta p) - y(p)}{\Delta p} \tag{1.1}$$

The derivative may be numerically approximated using the finite difference approximation (in which S_i is approximated by a small perturbation of p) or analytically computed with the direct differential method (in which the differential equation representation of a given model is solved for the sensitivity coefficients) [27]. For systems biology models, the finite difference method has been shown to give inaccurate results due to numerical instability in the derivative calculations [24]. Global sensitivity analysis methods instead produce estimates of sensitivity over an entire region of parameter space deemed relevant for the model in question [26].

1.5 Introduction to Information Theory

In 1948, Claude Shannon introduced the information theory framework to quantify information transfer and signal processing by a generic input-response network or communication system subjected to some source of noise [28]. For a discrete Markov process with event probabilities equal to $p_1 ... p_n$, he proposed the Shannon entropy, H:

$$H = -\sum_{x=1}^{n} P(x_i) \log_2 P(x_i) \tag{1.2}$$

as a measure of the degree of uncertainty in a variable given its probability distribution across states. A higher entropy state is one that is more uncertain; entropy increases both with the inclusion of more possible states and with increased diffusivity of probability across the possible states.

Quantifying uncertainty provided a means to measure the degree to which knowledge of one random variable X decreases uncertainty in a second variable Y, the mutual information between X and Y:

$$I(X;Y) = \sum_{X}\sum_{Y} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{1.3}$$

In recent years, information theory has been leveraged to analyze biological signaling systems [29, 30, 31]. Because the input signaling distributions to which biological signaling systems are exposed are generally unknown, previous researchers have estimated the channel capacity:

$$C = \sup_{p_x(x)} I(X:Y) \tag{1.4}$$

for a given system by sampling reasonably exhaustively across possible input distributions. In this work, information theory is applied to a specific catalytic system to investigate the role multiple inputs and varying input levels play in information transfer.

Chapter 2

The COX-2 Reaction Model (CORM)

2.1 Biological Background

The enzyme cyclooxygenase-2 (COX-2) lies at the interface of the eicosanoid and endocannabinoid signaling pathways [32, 33] and its dysregulation has been implicated in the pathogenesis of many diseases, including cancer [34], neurodegenerative diseases [35], and artheriosclerosis [32]. COX-2 is a homodimeric enzyme that acts as a heterodimer, with one subunit serving as the catalytic site and the other as an allosteric site [36, 37, 38]. It processes multiple substrates, has multiple allosteric regulators (including some members of the class of nonsteroidal anti-inflammatory drugs) [39, 40, 1, 41, 42, 43], and yields multiple products that drive pro- and anti-inflammatory downstream processes [44, 45, 46]. Because of the identical structure of COX-2's two subunits, both substrates and allosteric modulators can bind to and compete for both sites simultaneously, enabling substrates to also allosterically modulate catalytic activity and some inhibitors to act both competitively and noncompetitively.

The development of the computational model of COX-2 dynamics utilized in this work, the COX-2 Reaction Model (CORM), was motivated by the observation that COX-2 in the presence of two of its substrates, arachidonic acid (AA) and 2-arachidonoyl-glycerol (2-AG), displays behavior that cannot be explained by simple competition between the two substrates. Both AA and 2-AG are cleaved from membrane phospholipids prior to processing by COX-2. COX-2 turnover of AA produces prostaglandins (PGs), while 2-AG turnover produces prostaglandin-glycerol (PGG). Initial turnover produces the intermediate prostaglandin H2 (PGH_2) or $PGGH_2$; prostaglandin synthases that are expressed in a cell-type specific manner then convert the intermediate to specific prostaglandin subtypes, each of which exert particular downstream effects through corresponding receptors [32, 44].

2.2 Model Scope and Implementation

CORM includes the enzyme COX-2 (with both subunits contained in a single species). In CORM, the COX-2 species contains an allosteric and a catalytic site. Either site may bind either of the two COX-2 substrates included in the model, AA and 2-AG. COX-2 species with AA in the catalytic site are processed into PG (meant to represent the PGH_2 intermediate), while those with 2-AG in the catalytic site are processed to PGG ($PGGH_2$). The exception to this is the COX-2 species with 2-AG bound in both sites; because 2-AG displays substrate-dependent inhibition, turnover of this complex does not occur. Because the model was meant to represent a system in which the enzyme and substrates are well-mixed (reflecting the experimental calibration *in vitro* data), all forward binding rate constants (k_fs) were set to be diffusion limited. The corresponding reverse binding rates (k_rs) were allowed to vary independently during calibration, as were all catalytic rates (k_{cat}s) except those that were experimentally determined (indicated in red in Table 2.2). CORM does not encode the prostaglandin synthases that produce various prostaglandin subtypes by processing PGs; nor does it include any other substrates of COX-2 or COX-2 inhibitors. CORM was implemented as a PySB [12] model within a single module and is available as Python code at http://github.com/LoLab-VU/CORM. A diagram of all model interactions is shown in Figure 2.1.

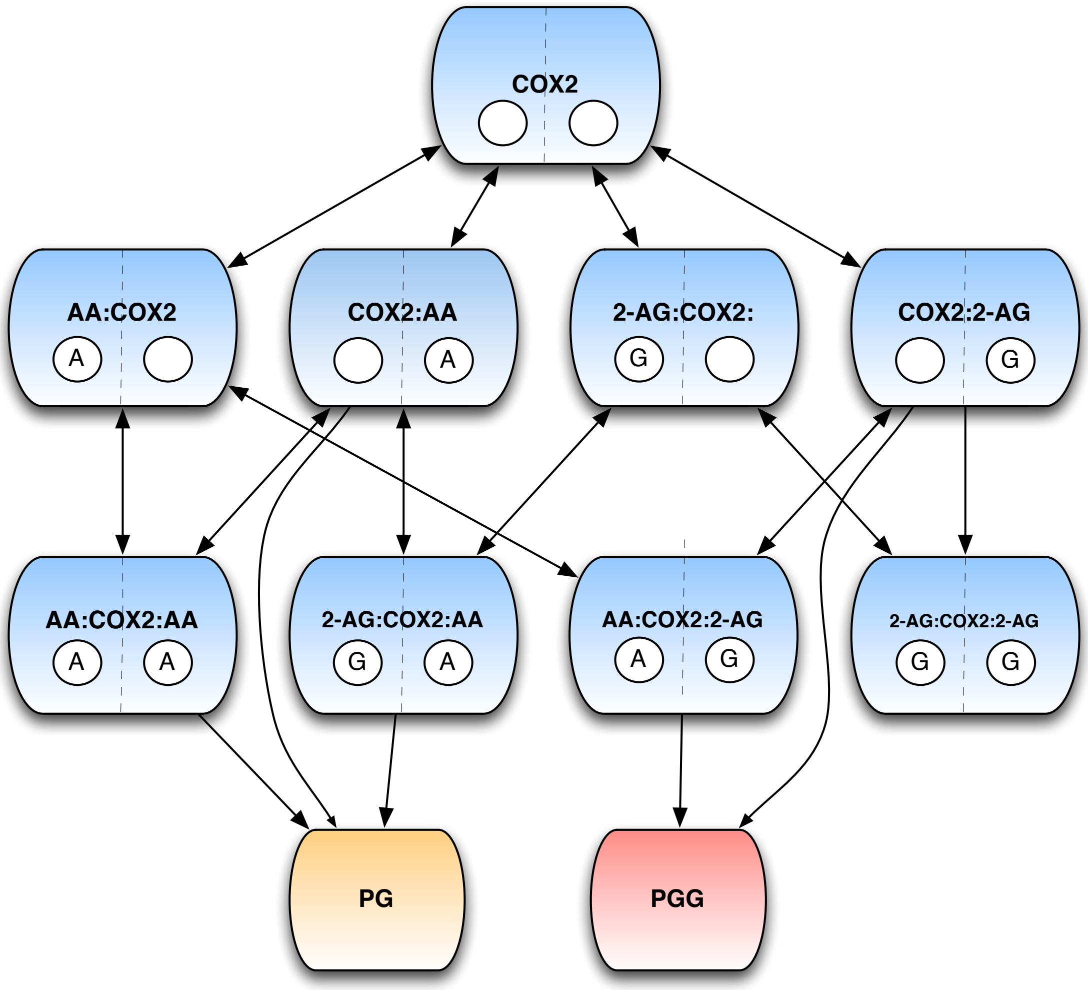


Figure 2.1: COX-2 Reaction Model (CORM)

2.3 Detailed Model Description

Model species are shown in Table 2.1 and model reactions in Table 2.2.

Species	Starting Concentration (μmolar)
COX-2	.015
AA	0, .5, 1, 2, 4, 8, 16
2-AG	0, .5, 1, 2, 4, 8, 16
PG	0
PGG	0
COX-2:AA(cat)	0
COX-2:2-AG(cat)	0
COX-2:AA(allo)	0
COX-2:2-AG(allo)	0
COX-2:AA(cat):AA(allo)	0
COX-2:2-AG(cat):2-AG(allo)	0
COX-2:AA(cat):2-AG(allo)	0
COX-2:2-AG(cat):AA(allo)	0

Table 2.1: CORM Species

PySB-generated Reaction	Rate or Equilibrium Constant
COX-2 + AA $\rightleftharpoons$ COX-2:AA(cat)	KD_AA_cat1 $=$ .83 μmolar
COX-2 + AA $\rightleftharpoons$ COX-2:AA(allo)	KD_AA_allo1
COX-2 + 2-AG $\rightleftharpoons$ COX-2:2-AG(cat)	KD_AG_cat1 $=$ .76 μmolar
COX-2 + 2-AG $\rightleftharpoons$ COX-2:2-AG(allo)	KD_AG_allo1
COX-2:2-AG(allo) + AA $\rightleftharpoons$ COX-2:AA(cat):2-AG(allo)	KD_AA_cat2
COX-2:AA(cat) + AA $\rightleftharpoons$ COX-2:AA(cat):AA(allo)	KD_AA_allo2
COX-2:2-AG(allo) + 2-AG $\rightleftharpoons$ COX-2:2-AG(cat):2-AG(allo)	KD_AG_cat2
COX-2:AA(cat) + 2-AG $\rightleftharpoons$ COX-2:AA(cat):2-AG(allo)	KD_AG_allo2
COX-2:AA(allo) + AA $\rightleftharpoons$ COX-2:AA(cat):AA(allo)	KD_AA_cat3
COX-2:2-AG(cat) + AA $\rightleftharpoons$ COX-2:2-AG(cat):AA(allo)	KD_AA_allo3
COX-2:AA(allo) + 2-AG $\rightleftharpoons$ COX-2:2-AG(cat):AA(allo)	KD_AG_cat3
COX-2:2-AG(cat) + 2-AG $\rightleftharpoons$ COX-2:2-AG(cat):2-AG(allo)	KD_AG_allo3 $=$ 63 μmolar
COX-2:AA(cat) $\rightarrow$ PG + COX-2	kcat_AA1 $=$ 1.3 s^{-1}
COX-2:2-AG(cat) $\rightarrow$ PGG + COX-2	kcat_AG1 $=$ 1.2 s^{-1}
COX-2:AA(cat):2-AG(allo) $\rightarrow$ PG + COX-2:2-AG(allo)	kcat_AA2
COX-2:2-AG(cat):2-AG(allo) $\rightarrow$ PGG + COX-2:2-AG(allo)	0 s^{-1}
COX-2:AA(cat):AA(allo) $\rightarrow$ PG + COX-2:AA(allo)	kcat_AA3
COX-2:2-AG(cat):AA(allo) $\rightarrow$ PGG + COX-2:AA(allo)	kcat_AG3

Table 2.2: CORM Reactions

Chapter 3

Advances in Model Calibration

3.1 Specific Advances from this Work

As described in Section 1.2, calibration of systems biology models is complicated by several factors, including parameter identifiability, large numbers of parameters, lack of experimental data for model calibration, and large uncertainties in individual parameter estimates. Because uncertainty in parameter estimates translates to uncertainty in all model predictions and is one metric of the trustworthiness and applicability of a model, interest in applying calibration methods that provide uncertainty estimates is increasing, although examples in the literature are still rare [5]. This chapter introduces PyDREAM, a Python implementation of the (Multiple-Try) Differential Evolution Adaptive Metropolis (DREAM$_{(ZS)}$) algorithm developed by [47] and [48]. The PyDREAM codebase was developed as a part of this thesis and resulted in a publication [2]. In addition, detailed results from two model calibrations performed with PyDREAM are presented, the first of which also resulted in a publication [1].

The original DREAM$_{(ZS)}$ algorithm [47, 48] encoded in PyDREAM has several characteristics that make it particularly suited for calibration of systems biology models. First, it provides probability distributions for fitted parameters, allowing quantification of uncertainty. Second, it is designed to fit large models containing many parameters, and models in which parameters may be multi-modal. Third, it employs parallel computation when possible; since simulation of many systems biology models is CPU-intensive, and calibration involves many model simulations, this is highly desirable. Finally, the DREAM algorithm (a form of Markov Chain Monte Carlo (MCMC)), obeys detailed balance (a characteristic of MCMC algorithms described in detail below, without which the results of an MCMC chain are likely unreliable). Because the only other systems biology model calibrated using MCMC methods violated detailed balance in order to obtain distributions that were deemed to be converged [5], this represents an important advance.

3.2 Background on PyDREAM Algorithm and Specific Implementation

3.2.1 Introduction to Bayesian Inference

The statistical methods most commonly applied in biology are frequentist, so named because in a frequentist context the probability that a particular event will take place is interpreted as the frequency of that event over a set of observations. This type of statistical analysis assumes that the parameters that describe the underlying population to be sampled remain fixed, while the observed data is sampled. Conclusions draw from a frequentist study then hinge upon the probability of again sampling data with similar characteristics. So a 95% confidence interval, for example, quantifies the region within which there is a 95% probability of getting a similar estimate for a given parameter, *if the experiment were repeated and new samples from the population were drawn.* Bayesian statistical analysis flips the convention; in a Bayesian context, probability is instead defined as a measure of belief or confidence that a given event will take place. When performing Bayesian analysis, observed data is fixed, while parameters describing the underlying population are sampled. A Bayesian 95% credible interval (the analogue of a frequentist confidence interval) quantifies the region within which there is a 95% probability that the true parameter lies. While the frequentist and Bayesian definition of probability align when a large enough sample of not improbable events is available, the Bayesian concept of prior belief is particularly powerful when data is sparse and can yield divergent conclusions in such cases.

The overarching goal of Bayesian inference is to calculate the *posterior probability*, that is, the probability of an occurrence (*e.g.*, a delayed flight) given both the *prior probability* of that event (*e.g.*, prior experience at an airport and knowledge of delay history) and the *likelihood* of observing the available data (*e.g.*, the frequency of flight delays from the airport) given a particular probability of delayed flights. The relationship between these terms is defined mathematically as:

$$\pi(\theta|D) = \frac{\pi(\theta)L(D|\theta)}{\int_\theta \pi(\theta)L(D|\theta)\,\mathrm{d}\theta} \tag{3.1}$$

where $\pi(\theta|D)$ is the *posterior probability* of an event θ after observing the data, D, $\pi(\theta)$ is the *prior probability* of the event θ before observing any data, and $L(D|\theta)$ is the *likelihood*, the probability of observing the data D if θ were true. The denominator in Equation (5.1) is termed the *evidence*, and is often computationally intensive to calculate. Fortunately, when only a single model is being studied, the *evidence* can be treated as a normalizing constant [49].

Occasionally, the posterior probability can be calculated analytically. More often, such as when θ is a high-dimensional vector, analytical solutions become intractable, and numerical approximations must be used. In this case, a suitable method to calculate posterior probabilities is to apply the Markov Chain Monte Carlo (MCMC) technique, discussed in the next section.

3.2.2 Introduction to Markov Chain Monte Carlo (MCMC)

Sometimes it is easier to estimate a deterministic quantity (such as the posterior probabilities discussed above) through random sampling rather than a direct analytical calculation. This approach was first envisioned by Stanislaw Ulam, a Polish American mathematician, while he passed the time recovering from an illness by playing Canfield solitaire. After considering for some time how to calculate the probability of winning a game of solitaire, he realized that sampling many solitaire games and recording win rates was a simpler approach than analytically calculating the win probability. He called the new random sampling method Monte Carlo sampling [50] in reference to the eponymous region of Monaco famous for its casinos and games of chance.

In order to generate random samples from a particular probability distribution, Markov

Chain Monte Carlo (MCMC) walk employs the *Markov chain* method. In a *Markov chain*, the next step taken depends only on the current location, and not on the path that led to the current location (*i.e.*, the previous steps). Markov chains have some useful mathematical properties for sampling of probability distributions, detailed in reference [51]. In MCMC, a Markov chain that has the probability distribution of interest as its equilibrium distribution is created. After a number of steps spent reaching equilibrium, the chain can be analyzed and tested for convergence to the target distribution, and all further chain steps are samples taken from the converged probability distribution. In order for a Markov chain to converge to its equilibrium distribution, it must be both *irreducible* and *aperiodic* [51]. An irreducible Markov chain is one in which it is possible to reach any state in the chain from any other state in the chain through positive probability transitions [51], while an aperiodic chain is one that does not contain periodic ”loops” within the chain [51]. While a Markov chain that meets these criteria will eventually converge to the desired equilibrium distribution, this convergence may be so slow as to make the analysis effectively computationally intractable. Adaptive MCMC is a class of algorithms designed to speed convergence to the target distribution by adapting the proposal distribution over the course of the algorithm. A variety of these algorithms have been developed [52, 53]; the specific adaptive MCMC method, DREAM, implemented in this work is discussed in the next section.

3.2.3 The DREAM Algorithm and Implementation in PyDREAM

PyDREAM provides a Python interface to several variants of a specific adaptive MCMC algorithm, DiffeRential Evolution Adaptive Metropolis (DREAM) [54, 55, 48]. PyDREAM includes the DREAM$_{(ZS)}$ [47, 53] and the MT-DREAM$_{(ZS)}$ [48] algorithms. All variants of the DREAM algorithm utilize a multi-chain structure in which multiple MCMC chains walk the parameter space simultaneously. In both implemented variants, a shared history of past chain states is maintained, and the next state is proposed by adding a perturbation derived from the distance between two randomly chosen history points. Occasionally, these parallel updates are supplemented by perpendicular snooker updates [47] in order to diversify proposal points. In the most recent MT-DREAM$_{\text{(ZS)}}$ [48] algorithm, multiple parallel proposals for each chain are created using parallel and snooker implementations of the multiple proposal schema introduced in [56]. The pseudo-code of the algorithm is given on the next page. PyDREAM follows [53] with the exception that the use of a single chain pair (e.g. $\delta = 1$) was hardwired for the parallel direction and snooker update. Scalars appear as lower-case italic, vectors as lower case letters, and matrices as upper case letters. Functions are typeset bold. Default values are used for the algorithmic variables. For a detailed list of all PyDREAM options, please see Appendix I.

```
 1: $c = 0.05$                                   ▷ Default values of DREAM algorithmic variables
 2: $c^* = 10^{-12}$
 3: $n_{\text{cr}} = 3$
 4: $p_\gamma = 0.2$
 5: $k = 10$                                     ▷ Default values of DREAM$_{\text{(ZS)}}$ algorithmic variables
 6: $p_{\text{sn}} = 0.1$
 7: $m_0 = 10d$
 8: $\text{cr} = \begin{bmatrix} 1/n_{\text{cr}} & 2/n_{\text{cr}} & \cdots & 1 \end{bmatrix}$   ▷ Crossover values
 9: $\text{p}_{\text{cr}} = 1/n_{\text{cr}}\textbf{ones}(n_{\text{cr}})$   ▷ Selection probability crossover values: $n_{\text{cr}}$ unity vector multiplied with $1/n_{\text{cr}}$
10: Z = **prior**($m_0$)                          ▷ Create initial archive with $m_0$ draws from the prior distribution
11: X = Z[$m_0 - n_{\text{chains}} + 1 : m_0$]    ▷ Use last $n_{\text{chains}}$ points of archive Z as initial chain states
12: $\text{p}_\text{X}$ = **pdf**(X)              ▷ Calculate posterior density of initial chain states
13: $m = m_0$                                    ▷ Set initial length archive equal to $m_0$
14: **for** *iter* in $n_{\text{iter}}$ **do**    ▷ **For each iteration**
15:     $\Lambda$ = **uniform**(-$c$, $c$,$n_{\text{chains}}$)   ▷ Draw $n_{\text{chains}}$ values from uniform distribution between $-c$ and $c$
16:     e = $c^*$**normal**(0,1,$n_{\text{chains}}$)   ▷ Vector of $n_{\text{chains}}$ standard normal draws multiplied with $c^*$
17:     **if** **uniform**(0,1) $\leq (1 - p_{\text{sn}})$ **then**   ▷ Parallel direction or snooker jump for this generation (iteration)?
18:         **for** *chain* in $n_{\text{chains}}$ (sequential) **do**   ▷ **Parallel direction jump for each chain**
19:             $a$, $b$ = **randsample**($m$, 2)   ▷ Sample two integers from 1, 2, ... $m$
20:             *cross* = **draw_crossover**(cr, $\text{p}_{\text{cr}}$)   ▷ Sample crossover from vector cr with probabilities $\text{p}_{\text{cr}}$
21:             $d^* = 0$                       ▷ Number of dimensions to update is zero
22:             **for** *dim* in $n_{\text{dim}}$ **do**   ▷ **For each dimension**
23:                 **if** **uniform**(0,1) $\leq$ *cross* **then**   ▷ Is dimension *dim* selected with current crossover, *cross*
24:                     $\text{dx}_{dim}$ = Z[$a$]$_{dim}$ - Z[$b$]$_{dim}$   ▷ Update dx dimension *dim* to difference of dim of points Z[$a$] and Z[$b$] of archive
25:                     $d^* = d^* + 1$     ▷ Increment dimensions updated
26:                 **else**
27:                     $\text{dx}_{dim} = 0$   ▷ Zero jump in respective dimension
28:                 **end if**
29:             **end for**
30:             **if** **uniform**(0,1) $\geq$ $p_\gamma$ **then**
31:                 $\gamma = 2.38/\sqrt{2\delta d^*}$   ▷ Use default jump factor for $d^*$ dimensions
32:             **else**
33:                 $\gamma = 1$                 ▷ Set jump factor to unity to simplify mode jumping
34:             **end if**
35:             dX[*chain*] = e[*chain*] $+ \gamma(1 + \Lambda[chain])$dx   ▷ Parallel direction jump vector of current chain
36:             $\alpha_{\text{sn}}[chain] = 1$   ▷ Symmetric proposal distribution (no snooker correction needed)
37:         **end for**
```

```
38:     else
39:         for chain in $n_{\text{chains}}$ (sequential) do                    ▷ Snooker jump for each chain
40:             $a, b, c$ = randsample(m, 3)                                  ▷ Sample three integers from 1, 2, ... $m$
41:             $\gamma$ = uniform(1.2, 2.2)                                  ▷ Sample randomly the jump factor
42:             F = X[$chain$] - Z[$a$]        ▷ Difference vector of current chain state and archive point Z[a]
43:             zp = orthog_proj(F, Z[$b$], Z[$c$])   ▷ Project orthogonally points Z[$b$] and Z[$c$] of archive Z onto F
44:             $\text{dX}[chain] = \text{e}[chain] + \gamma(1+\Lambda[chain])\text{zp}$        ▷ Snooker jump vector of current chain
45:             $\alpha_{\text{sn}}[chain] = \left((\text{X}[chain]+\text{dX}[chain]-\text{Z}[a])^2/(\text{X}[chain]-\text{Z}[a])^2\right)^{(d-1)}$   ▷ Snooker correction nonsymmetry
    jump
46:         end for
47:     end if
48:     Xp = X + dX                                                          ▷ Compute candidate points for the chains
49:     for chain in $n_{\text{chains}}$ (in parallel) do                          ▷ For each chain
50:         $p_{\text{Xp}}[chain]$ = pdf(Xp[$chain$])                          ▷ Calculate posterior density of proposal point
51:     end for
52:     for chain in $n_{\text{chains}}$ (sequential) do
53:         $p_{\text{acc}}[chain] = \textbf{min}\left(1, \alpha_{\text{sn}}[chain](p_{\text{Xp}}[chain]/p_{\text{X}}[chain])\right)$       ▷ Calculate acceptance probability
54:         if $p_{\text{acc}}[chain] \geq$ uniform(0,1) then                    ▷ If accept Xp of current chain
55:             X[$chain$] = Xp[$chain$]                                      ▷ Candidate point becomes new state of chain
56:             $p_{\text{X}}[chain] = p_{\text{Xp}}[chain]$     ▷ Density of proposal is equivalent to density of current state chain
57:         end if
58:     end for
59:     if modulus($iter$, $k$) == 0 then                ▷ Check whether to append current population to archive
60:         Z[$m+1 : m+n_{\text{chains}}$] = X                                ▷ Append current states of chains to archive
61:         $m = m + n_{\text{chains}}$                                       ▷ Increment number of points in archive
62:     end if
63: end for
```

Calibration of a kinetic model with PyDREAM involves three steps: specifying the model and any constraints on the model (such as from experimental data or physical constraints), sampling with PyDREAM, and analysis of inferred kinetic rate distributions. The process is outlined in Fig. 3.1.

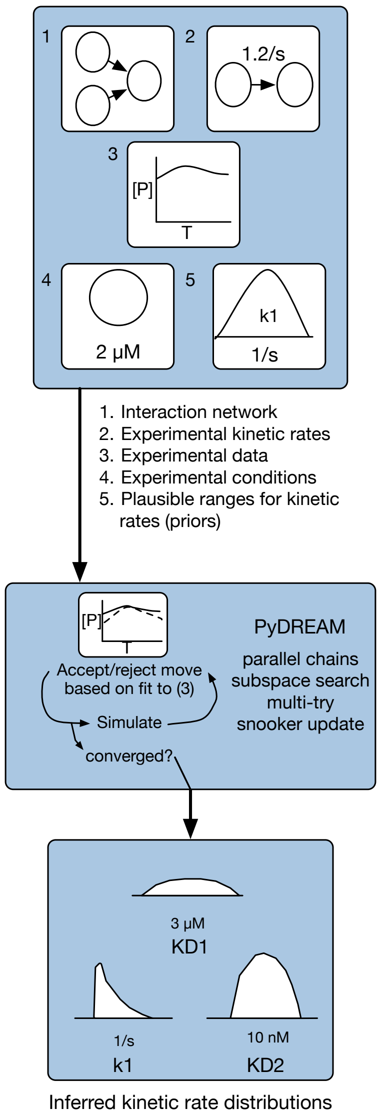

1
2
1.2/s
3
[P]
T
4
2 μM
5
k1
1/s
1. Interaction network
2. Experimental kinetic rates
3. Experimental data
4. Experimental conditions
5. Plausible ranges for kinetic rates (priors)
[P]
T
Accept/reject move based on fit to (3)
Simulate
converged?
PyDREAM
parallel chains
subspace search
multi-try
snooker update
3 μM
KD1
1/s
k1
10 nM
KD2
Inferred kinetic rate distributions

Figure 3.1: Schematic representation of PyDREAM workflow. The experimental data, model setup, prior parameter distribution, and likelihood function are defined as algorithmic input (top). This serves as input to the technical heart of PyDREAM (middle) which uses differential evolution for chain evolution, with a Metropolis selection rule to decide whether to accept/reject candidate points. These sampled chain trajectories (bottom) are then returned to the user and used to summarize the marginal posterior parameter distributions.

3.2.4 PyDREAM Validation

To ensure that the version of DREAM encoded in PyDREAM is consistent with original version of DREAM in MATLAB, PyDREAM performance for several test cases was computed and compared to the MATLAB results.

3.2.4.1 10D Bimodal Mixture Model

This test case, originally described in [55], is a ten-dimensional bimodal pdf with modes centered around -5 and 5, each with a variance of 5. Two thirds of the density is centered around the latter mode. PyDREAM was run for 50,000 iterations with 3 chains and 5 multi-try parallel tests, requiring 5 minutes on a six-core CPU. The first 50% of the samples were removed as burn-in. The sampled distributions for each parameter dimension are shown in Figure 3.2.

3.2.4.2 200D Multivariate Normal

This test case, described in [48], is a 200-dimensional multivariate normal distribution. The variance of the *j*th variable is equal to *j*, and the pairwise correlations are set to 0.5. PyDREAM was run for 150,000 iterations with 3 chains and 5 multi-try parallel tests, requiring 50 minutes on a six-core CPU. The first 50% of the samples were removed as burn-in. The sampled distributions for 10 representative dimensions are shown in Figure 3.3.

3.3 Calibration of CORM with PyDREAM

As mentioned in Section 2.1, the development of CORM was motivated by the observation that COX-2 in the presence of two of its substrates, AA and 2-AG, displays kinetics that cannot be explained by commonly used models of inhibition and competition [57] based on Michaelis-Menten assumptions [58]. It was hypothesized that in order to accurately predict experimentally observed COX-2 product (PG and PGG) kinetics when both substrates were

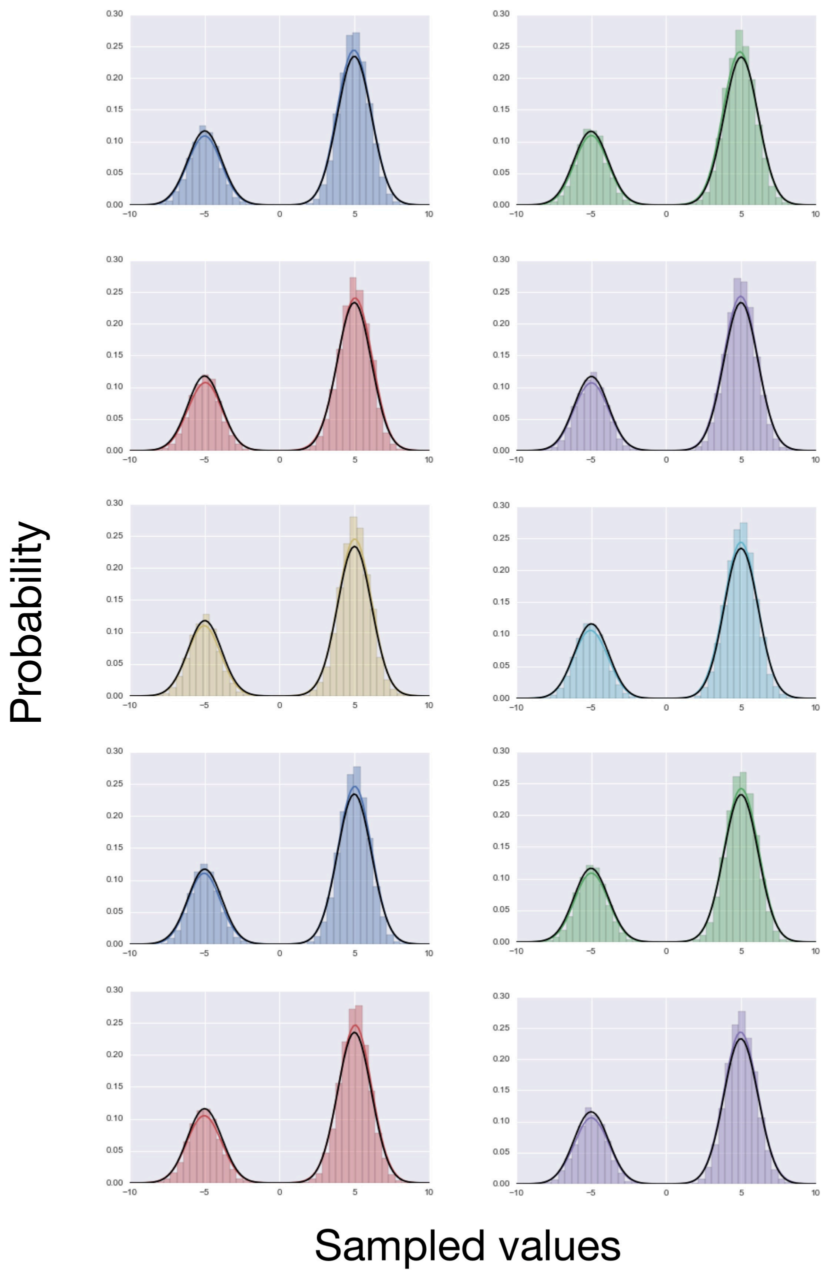


Figure 3.2: PyDREAM samples for a 10D bimodal test case. Each subfigure depicts a parameter dimension. Samples drawn using PyDREAM are shaded while the true distribution is indicated by a black line. Sampled values are in arbitrary units.

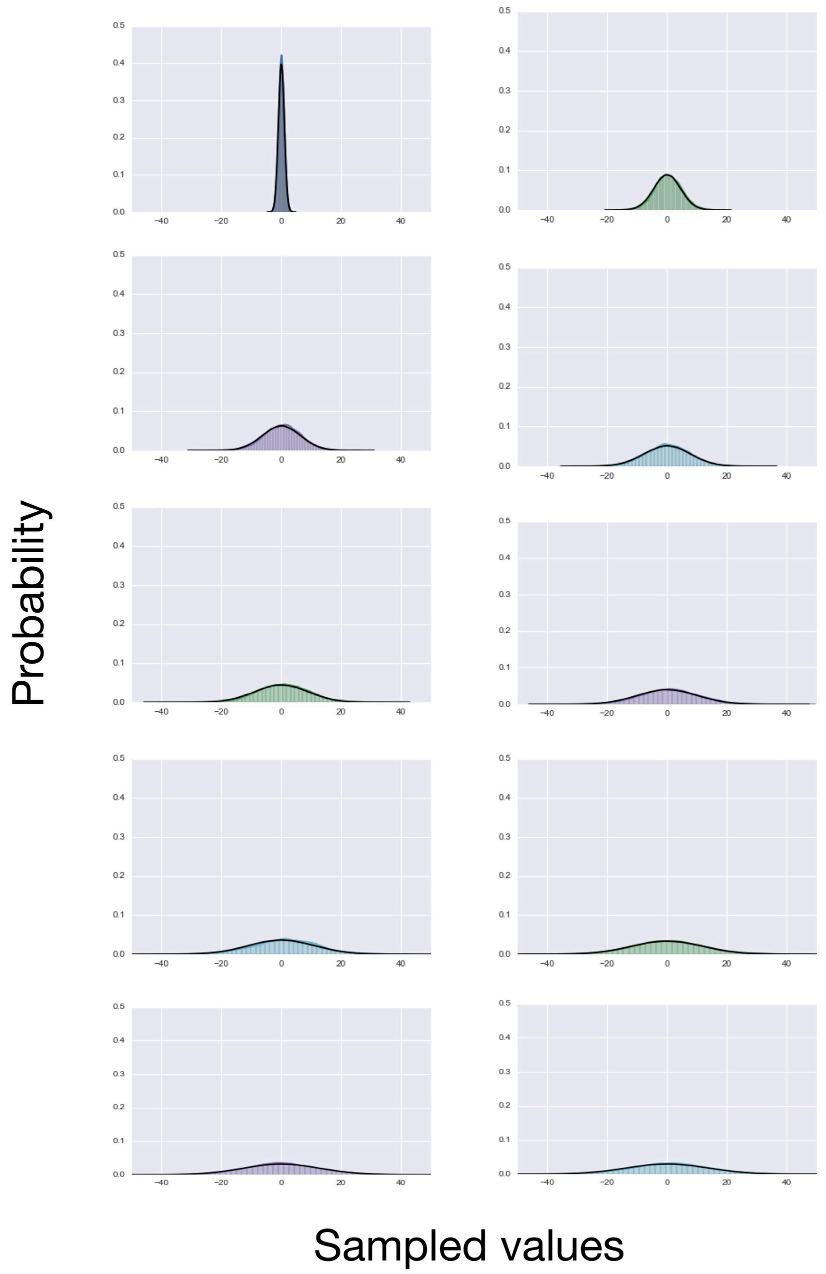


Figure 3.3: PyDREAM samples for a 200D multivariate normal test case. Each subfigure depicts one of ten representative parameter dimensions. Samples drawn using PyDREAM are shaded while the true distribution is indicated by a black line. Sampled values are in arbitrary units.

present, all possible interactions between both substrates and the enzyme at both catalytic and allosteric sites should be considered. It was further hypothesized that the reason for the observed kinetics was a difference in catalytic rates induced by allosteric interactions. Creating a computational model of COX-2, AA, and 2-AG dynamics (CORM) and fitting that model to available experimental data would provide distributions of kinetic parameters and make it possible to test whether first, a model including all interactions could accurately reproduce the experimentally observed dynamics, and second, whether kinetic rates inferred from the experimental data and model would support the role of allostery in catalysis.

3.3.1 Available Calibration Constraints

3.3.1.1 Experimental Data and Experimentally Measured Kinetic Rates

The available experimental data consisted of 49 measurements at different substrate concentrations of the products PG and PGG, all measured ten seconds after the mixing of enzyme and substrate. Each data point was collected in triplicate and reported as a mean and standard deviation. The experimental data can be found in reference [1], Figure 3. Six experimentally measured catalytic rates and disassociation constants were available from our collaborators. These are marked in red in Table 2.2.

3.3.1.2 Thermodynamic Constraints

When a kinetic scheme contains cycles with identical beginning and ending species, at equilibrium the net flux through the cycle vanishes. This constrains the product of the equilibrium constants for the reactions in the cycle to be equal to one. Alternatively, one may view the cycle as energy conserving (no net change in free energy). Within the specified interaction network, there are four such thermodynamic cycles, within which relative parameter values must be consistent with energy conservation. These cycles are shown in Figure 3.4.

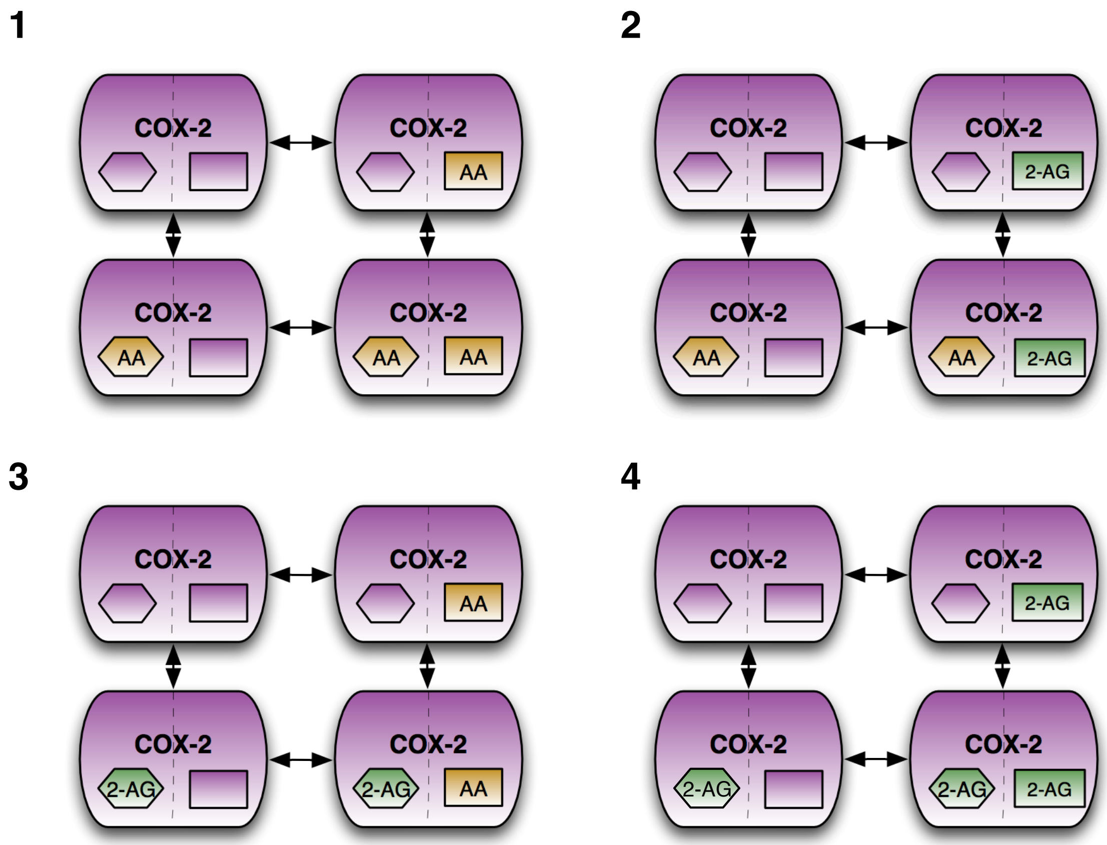


Figure 3.4: Thermodynamic Cycles in CORM

3.3.1.3 Kinetic Parameter Prior Distributions

Broad (spanning multiple orders of magnitude), normal prior distributions were specified for all parameters to be fitted. These were selected based on expert biological knowledge provided by collaborators. Disassociation constants were used rather than forward and reverse rates; the forward rates were assumed to be diffusion limited and the reverse rates were varied to give a particular K_D. Prior distributions for all parameters are shown in Figure 3.5.

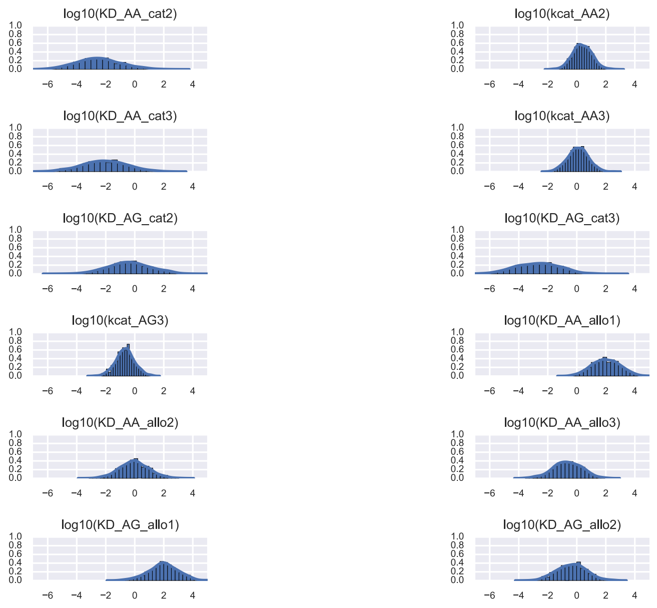


Figure 3.5: Prior Distributions for Fitted Parameters

3.3.2 Calibration Results

PyDREAM was initialized with five chains in random locations in parameter space drawn from prior parameter probability distributions. Sampling was performed for 2.5 million iterations; 100,000 samples were discarded as burn-in, and sample matrixes were thinned by a factor of 10. All chains converged to a limiting distribution as assessed by both the Geweke score [59] and Gelman Rubin convergence criterion [60]. After calibration, simulation at any vector in the ensemble of parameter sets produced results within experimental data error; representative simulation results generated using the most probable parameter vector from the ensemble overlaid with experimental data points are shown in Figure 3.6.

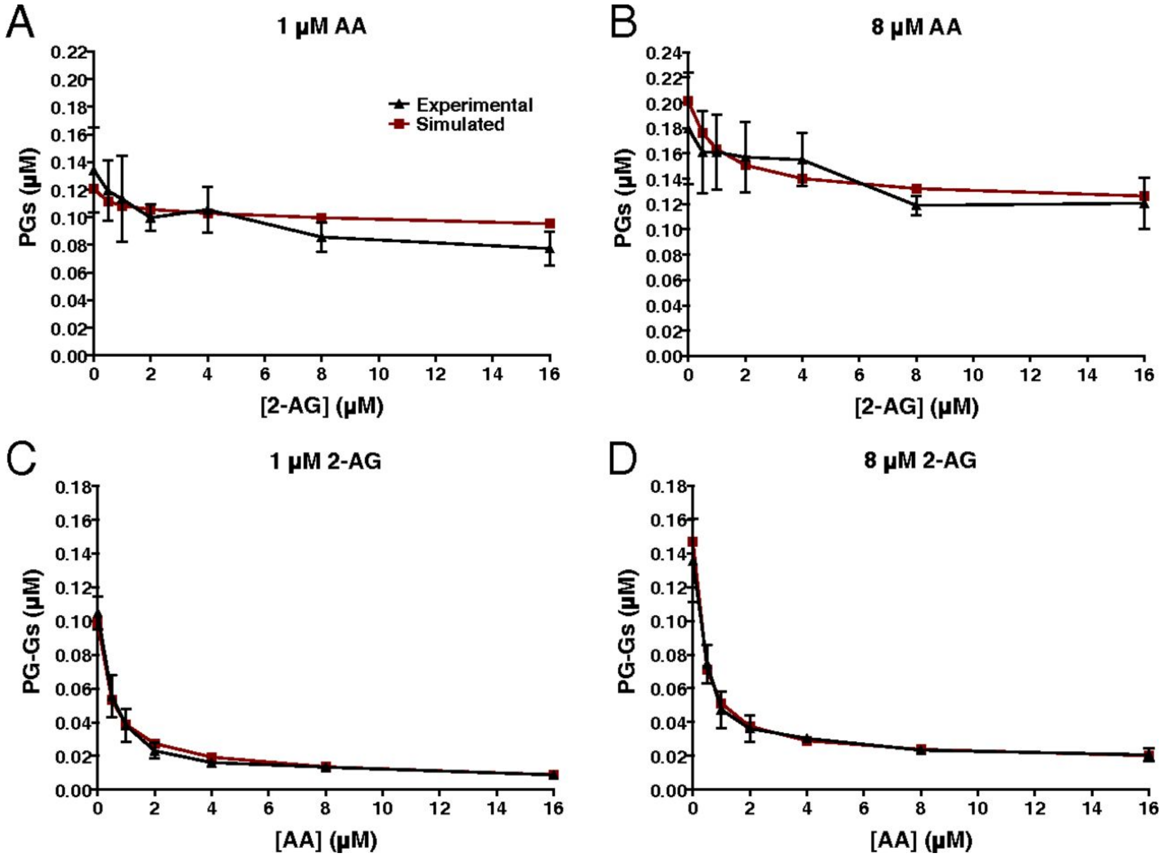


Figure 3.6: Simulated model PG and PGG values generated using the most probable parameter vector obtained during calibration at several concentrations of AA and 2-AG. Experimental data is shown in black and simulated results in red.

Parameter distributions from the calibrated parameter ensemble for each of the twelve fitted kinetic rates in CORM are shown in Figure 3.7.

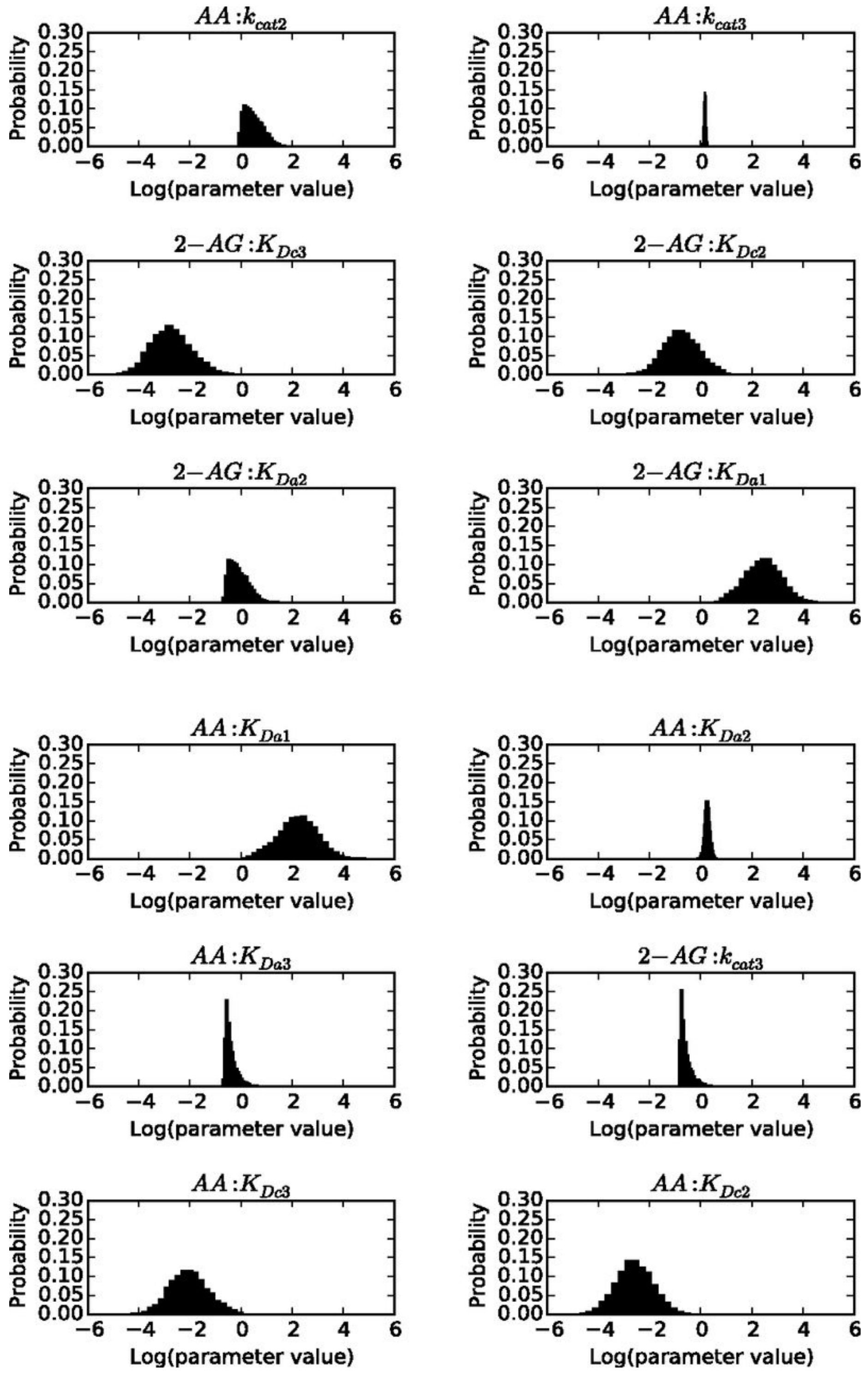


Figure 3.7: Distributions of parameter values for each fitted kinetic rate in CORM.

3.3.3 Insights into COX-2 Catalysis

CORM was designed to address two biological questions related to COX-2 catalysis. The first question was whether a model including all interactions at both catalytic and allosteric sites would be able to capture the nonlinear dynamics present in the experimental data. This did occur, as is evident by the close fit of simulations to data in Figure 3.6. The second question was whether the calibrated kinetic rates from CORM would support the importance of allostery in the system dynamics. The calibrated rates suggest that allostery does measurably impact the catalytic rates in the system, altering the system dynamics. The allosteric effects on catalysis vary depending on the substrate and allosteric regulator. As shown in Figures 3.8 and 3.9, the distributions of the catalytic rates for AA turnover when either 2-AG or AA is bound in the allosteric site indicate that binding of the allosteric modulator most likely increases the catalytic rate (as the bulk of the probability density lies to the right of the experimentally measured allosterically unbound enzyme indicated by the red line.

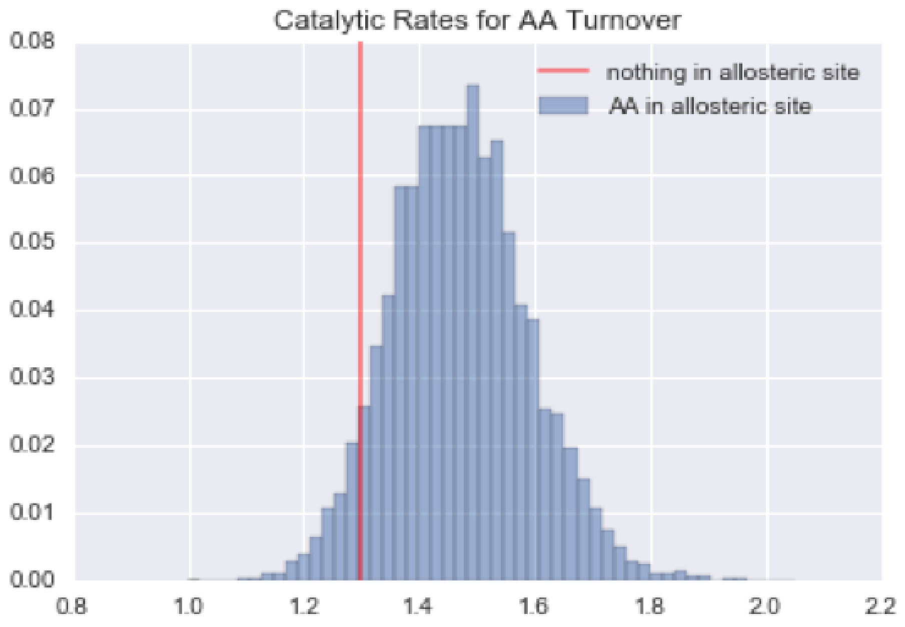


Figure 3.8: Distribution of catalytic rate (in s^{-1}) when AA is bound in both the catalytic site and allosteric site of COX-2. The experimentally measured catalytic rate for the enzyme in the absence of any allosteric modulator is indicated by the red line.

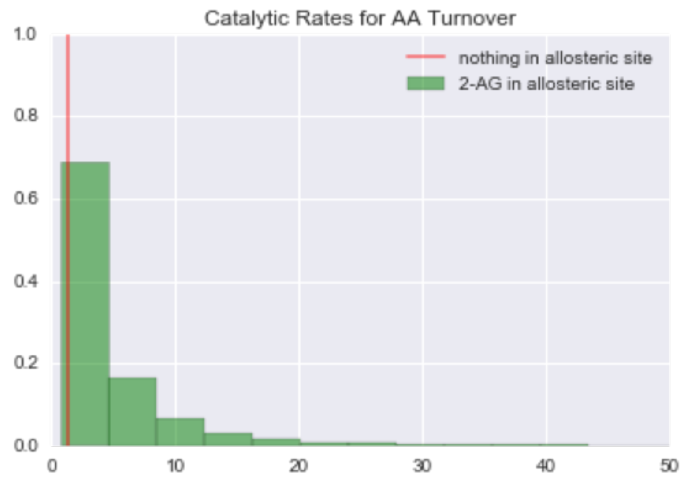


Figure 3.9: Distribution of catalytic rate (in s^{-1}) when AA is bound in the catalytic site and 2-AG in the allosteric site of COX-2. The experimentally measured catalytic rate for the enzyme in the absence of any allosteric modulator is indicated by the red line.

When 2-AG is in the catalytic site, however, binding of AA in the allosteric site most likely decreases the turnover rate for 2-AG, as indicated by the majority of the probability density present to the left of the allosterically unbound enzyme in Figure 3.10.

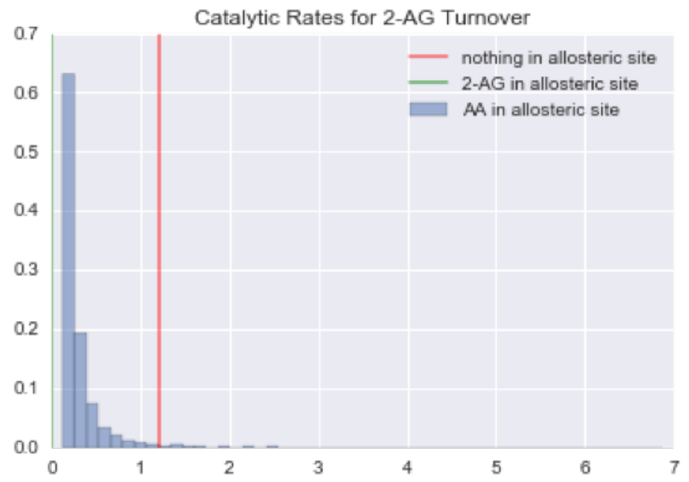


Figure 3.10: Distribution of catalytic rate (in s^{-1}) when 2-AG is bound in the catalytic site and AA in the allosteric site of COX-2. The experimentally measured catalytic rate for the enzyme in the absence of any allosteric modulator is indicated by the red line.

3.4 Calibrating a Larger Model with PyDREAM

After the successful calibration of CORM with PyDREAM, calibration of a larger model was initiated. While CORM has only twelve free rate parameters, models with many more free parameters occur in systems biology research [5, 18], and are generally far more challenging to calibrate to experimental data. The Extrinsic Apoptosis Reaction Model (EARM) was chosen as a test model because it had previously been calibrated and a set of parameter values with good fit to experimental data already identified, but further calibration to identify the uncertainty in these parameter values would be valuable. EARM contains 105 free rate parameters. The experimental data available for calibration of EARM is quantitatively different than that available for CORM calibration. While CORM calibration data consists of two outputs measured at a single time point across a range of different initial conditions, EARM experimental data consists of time courses for three outputs under a single initial condition.

Initial attempted calibration attempts for EARM with PyDREAM used a set of uniform priors that were chosen based on physiological constraints to particular types of rate constants, as defined in Table 3.1. A variety of different algorithmic settings (including PyDREAM chain numbers of 3, 5, and 12, the use of multi-try updates and snooker updates, and selection of start points with good fit to experimental data rather than random start points) were tested repeatedly to assess whether convergence of parameters was occurring.

Rate Constant Type	Lower Bound	Upper Bound
Forward association rate (k_f)	$10^{-4}/(M*s)$	$10^{6}/(M*s)$
Equilibrium dissociation constant (K_D)	10^{-12} M	10^{-3} M
Catalytic constant (k_{cat})	10^{-2}	10^{3}

Table 3.1: EARM Rate Constant Constraints

Unfortunately, inferring calibrated parameter distributions for EARM using only the generic physiologically relevant priors listed above was not successful; different chains did not converge to the same distribution visually (see Fig. 3.11 for an example).

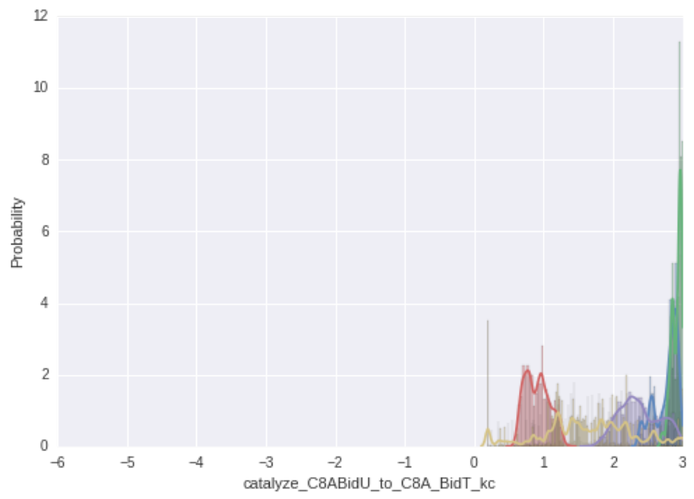


Figure 3.11: Uncoverged distributions of MCMC draws for a single parameter in the EARM model. Different colors indicate different chains run in parallel, which should reflect similar distributions for convergence.

As a way to provide further constraints to the parameters in the system and hopefully achieve convergence, two methods were applied. First, experimental parameter rates from the literature were located when possible (see Table 3.2 for experimentally measured parameter values and their literature source). All reverse rate constants (kr) values present in the table are actually K_D values used as described above. Values in the table are given in molar units, but for use in EARM all values were scaled by Avogadro's number and a cellular volume of 1.661 pL (chosen to match median experimental measurements of HeLa cell cytoplasmic volumes [61]). Parameters for kinetic rates operating in the mitochondrial membrane were set to 7% of the cytoplasmic volume.

As a second source of constraints, certain kf values for which structures were available were inferred using the program TransComp [62]. Complex association rates may be limited by either diffusion or conformational change. In addition, diffusion limited complex formation when strong electrostatic attraction between the two subunits is present may increase the association rate beyond that dictated by diffusion alone. Predictions generated by TransComp are generated based on transient-complex theory. The transient-complex represents the boundary between the unbound and bound states for a given complex of two proteins when the reaction is diffusion-limited (accounting for electrostatic interaction). In the unbound state, the two subunits in the complex have complete rotational and translational freedom, but few specific inter-subunit short-range interactions. In the bound state, the opposite is true; the subunits are translationally and rotationally constrained, but there are many specific inter-subunit short-range interactions. Forward association rates for diffusion-limited complex formation (with accompanying electrostatic interactions) were predicted well by the following equation:

$$k_a = k_{a0} \exp(\frac{-\Delta G^*_{el}}{k_B T}) \tag{3.2}$$

where k_{a0} is the basal association rate by diffusion alone, and the Boltzmann factor reflects the contribution to association from electrostatic interactions [63]. TransComp takes as input the structure of the complex and calculates both k_{a0} and ΔG^*_{el}. EARM association rate parameters which were predicted using TransComp, the PDB structure used as input, and the literature reference for the structure are shown in Table 3.3. In cases where multiple structures were available, TransComp was used to predict an association rate for each structure and the variance in the results was used to set the standard deviation in a normal prior around the average of all results.

A decision tree summarizing the process for selecting priors during EARM calibration using the available constraints is shown in Fig. 3.12. The calibration process when including these constraints was more successful than without. Multiple parameters converged, both as measured by the Gelman-Rubin metric and by visual comparison of distributions. These included parameters that had prior constraints (Fig. 3.13) as well as parameters that had no experimental or predicted prior constraints available (Figs. 3.14, 3.15, and 3.16). The latter category included both parameters for which the initial prior was returned after calibration (Fig. 3.14), indicating the available calibration data provided no information about the marginal distribution of the parameter, as well as parameters for which the available calibration data did further constrain the parameter posteriors (Figs. 3.15 and 3.16). Unfortunately, while including the constraints did allow some parameters to converge, there were still parameters that failed to converge after millions of iterations, and traceplots for these parameters showed evidence of poor mixing (Fig. 3.17). At this juncture resources were redirected to focus on further exploration of the COX-2 system, which was hoped to be more fruitful.

Parameter Name (as used in EARM)	Value	Reference
bind_L_R_to_LR_kr	10 M	[64]
catalyze_ApopC3pro_to_Apop_C3A_kc	.1 s^{-1}	[65, 66]
bind_Apop_XIAP_kr	11 nM	[67, 68]
bind_SmacA_XIAP_kr	1 μM	[69]
catalyze_C8AC3pro_to_C8A_C3A_kc	.37 s^{-1}	[65]
bind_XIAP_C3A_to_XIAPC3A_kf	2.5 μM	[70]
bind_XIAP_C3A_to_XIAPC3A_kr	.5 nM	[71, 72, 73, 74]
catalyze_C3APARPU_to_C3A_PARPC_kc	1 s^{-1}	[65, 75]
catalyze_C3AC6pro_to_C3A_C6A_kc	9.1 s^{-1}	[65]
equilibrate_BidT_to_BidM_kr	100 μM	[76, 77]
equilibrate_BaxC_to_BaxM_kr	50 nM	[78]
equilibrate_BclxLC_to_BclxLM_kr	50 nM	[78]
bind_BidM_BaxC_to_BidMBaxC_kr	600 nM	[79]
bind_BidM_BakM_to_BidMBakM_kr	5 mM	[80]
bind_BidM_Bcl2M_kr	100 nM	[81, 82, 83]
bind_BidM_BclxLM_kr	200 nM	[79, 81, 82, 84, 85]
bind_BidM_Mcl1M_kr	1 nM	[81]
bind_BaxA_Bcl2_kr	100 nM	[82, 86]
bind_BaxA_BclxLM_kr	150 nM	[82, 86]
bind_BaxA_Mcl1_kr	39.5 nM	[82]
bind_BakA_Bcl2_kr	95 nM	[82, 86]
bind_BakA_BclxLM_kr	15 nM	[82, 86]
bind_BakA_Mcl1M_kr	10 nM	[82, 86]
bind_BadM_Bcl2_kr	14 nM	[81, 82]
bind_BadM_BclxLM_kr	10 nM	[79, 81, 82, 85, 87, 88]
bind_NoxaM_Mcl1M_kr	24 nM	[81, 82]

Table 3.2: Experimentally Measured Parameters from Literature Used for EARM Calibration

Parameter Name (as used in EARM)	Predicted Value	PDB ID	Reference
bind_L_R_to_LR_kf	1 x 10^4	1D4V, 1D0G, 1DU3	[89, 90, 91]
convert_ApafA_C9_to_Apop_kf	34.9	4RHW	[92]
bind_Apop_XIAP_kf	2.52 x 10^3	1NW9	[93]
bind_SmacA_XIAP_kf	8.17 x 10^5	1G73	[94]
bind_BidM_BaxM_to_BidMBaxM_kf	1.75 x 10^4	4ZIG, 4BD2	[95, 96]
bind_BidM_BakM_to_BidMBakM_kf	7.31 x 10^5	2M5B	[80]
bind_BidM_BclxLM_kf	2.37 x 10^4	4QVE	[97]
bind_BidM_Mcl1M_kf	1.58 x 10^4	2KBW	[98]
bind_BaxA_Bcl2_kf	5.44 x 10^3	2XA0	[82]
bind_BaxA_Mcl1_kf	1.63 x 10^4	3PK1	[99]
bind_BakA_BclxLM_kf	1.1 x 10^5	1BXL, 2LP8	[100, 101]
bind_NoxaM_Mcl1M_kf	2.31 x 10^5	2ROD, 2JM6	[102, 103]

Table 3.3: Association Rate Parameters Predicted with TransComp and Used for EARM Calibration. All units are M^{-1} s^{-1}.

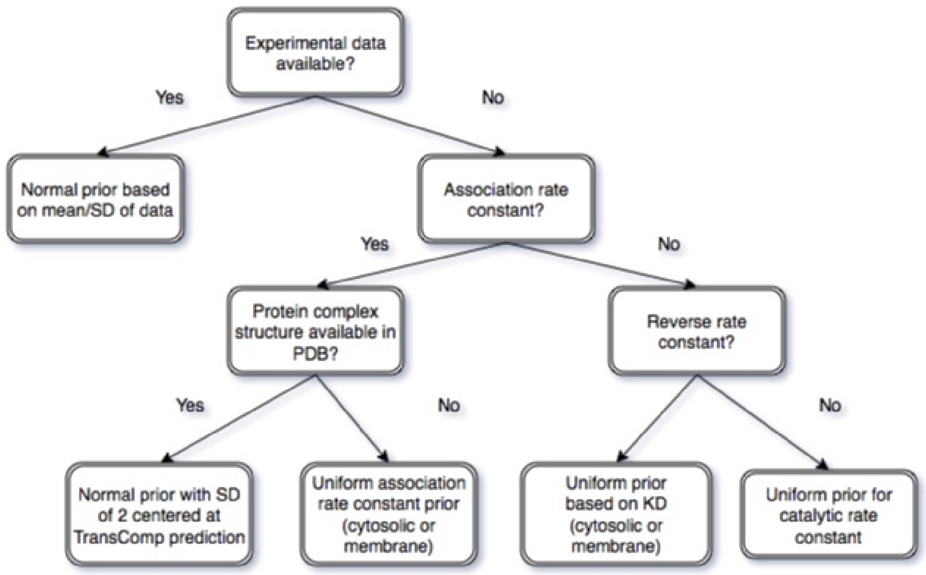

Experimental data available?
Yes
No
Normal prior based on mean/SD of data
Association rate constant?
Yes
No
Protein complex structure available in PDB?
Reverse rate constant?
Yes
No
No
Normal prior with SD of 2 centered at TransComp prediction
Uniform association rate constant prior (cytosolic or membrane)
Uniform prior based on KD (cytosolic or membrane)
Uniform prior for catalytic rate constant

Figure 3.12: The process used for selecting parameter priors for EARM calibration, using either experimental data or computational prediction.

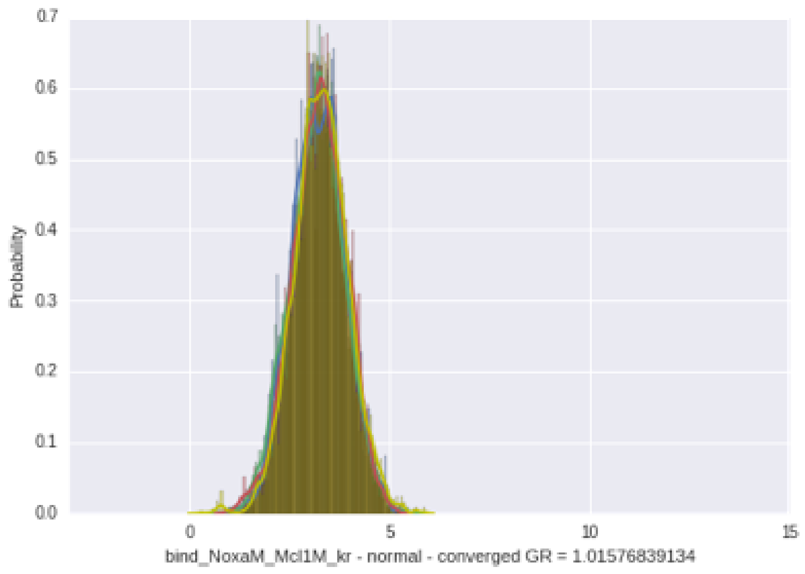


Figure 3.13: Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The normal prior used was derived from experimental measurements in the literature. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence.

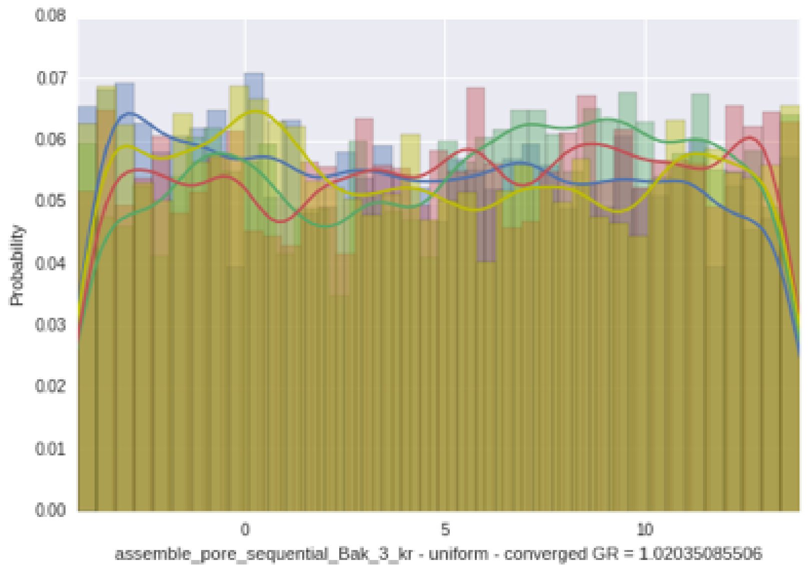


Figure 3.14: Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The calibrated parameter values are unchanged from the uniform prior. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence.

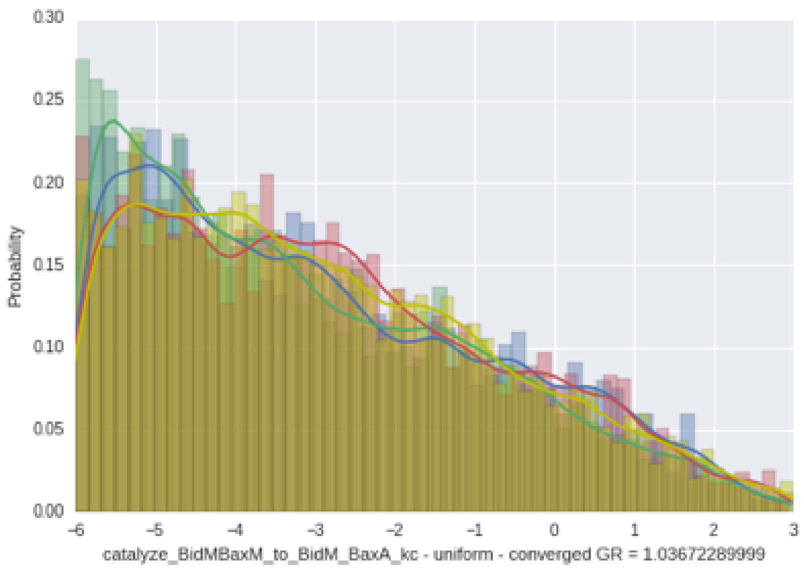


Figure 3.15: Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The original parameter prior was uniform across the range of values. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence.

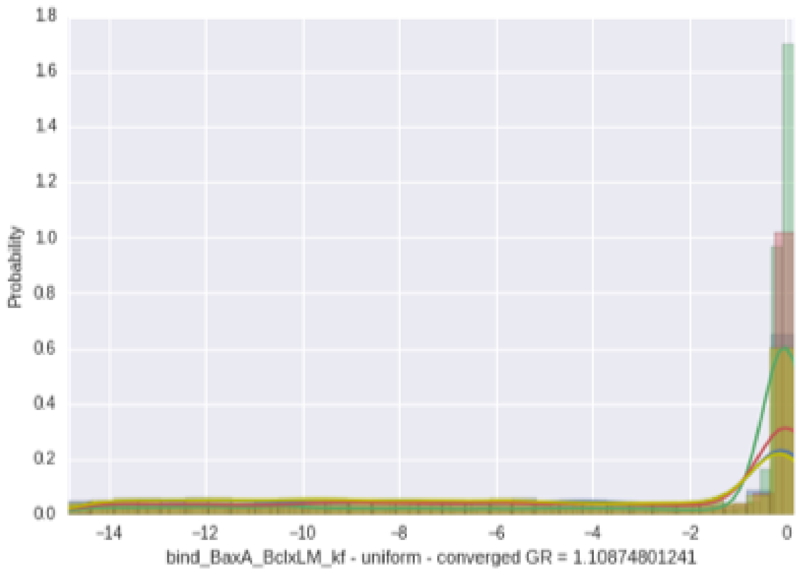


Figure 3.16: Converged (by Gelman-Rubin metric) distributions of MCMC draws for a single parameter in the EARM model. Parameter values are log transformed. The original parameter prior was uniform across the range of values. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence.

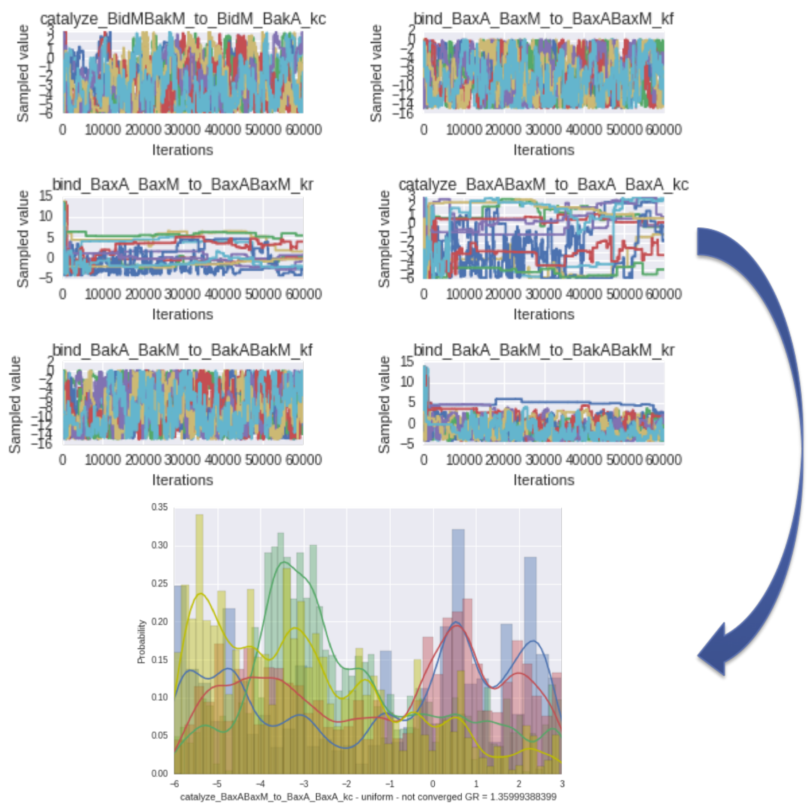


Figure 3.17: Unconverged (by Gelman-Rubin metric) distribution of MCMC draws for a single parameter in the EARM model (bottom) and the corresponding traceplot for that parameter and other parameters, some of which had reached convergence (top). Parameter values are log transformed. The original parameter prior was uniform across the range of values. Different colors indicate different chains run in parallel, which should reflect similar distributions after convergence.

Chapter 4

Advances in Model Flux Analysis

4.1 Specific Advances and Problems Solved

After publication of the work described in Sections 2.1 and 3.3, knowledge of COX-2 catalytic function in the presence of allosteric regulators was significantly increased. Computational modeling and rigorous model calibration provided the information that, for the two allosteric modulators examined, both positive and negative allosteric regulation of catalysis can occur, depending on the substrate in question. However, this analysis did not address the degree to which the system actually utilizes different catalytic complexes, and whether that degree of utilization depends on levels of key nodes within the network. A method to analyze the amount of chemical reaction flux through different paths within the network would be valuable for the COX-2 system, in which multiple complexes (containing combinations of different allosteric regulators and substrates) could theoretically produce product. It would also be applicable to any system in which multiple paths lead from system inputs to system outputs; given the combinatorial complexity present in many biochemical networks, a large number of applications in other biological systems would be expected. As one example, the extrinsic apoptosis model described previously, EARM, includes apoptosis induction through mitochondrial membrane pore formation by two different proteins, Bax and Bak [5, 12]. Application of such a tool could untangle the degree to which the death signal proceeds through pore formation using each protein, and whether this varies with the levels of network components, intrinsic noise, or other system features.

Such an analysis technique was not found in the literature. Flux balance analysis (perhaps the most common form of flux analysis in a network), is performed on a system at steady-state to calculate the possible combinations of fluxes into and out of a system that will conserve mass at steady-state [25, 104, 105]; for example, if a network consists of a single input node that is then processed by either of two downstream output branches to produce two outputs, the system has two possible outcomes at steady-state for every single molecule of input (assuming each branch consumes only a single input molecule): production of a single molecule of product one via path one, or production of a single molecule of product two via path two, and any linear combination thereof with increasing amounts of input. This process does not require kinetic rate information and is time independent; while these features make it more easily applicable, they also limit the insights derived to general rather than specific. Flux balance analysis enumerates the possible flux states the system could possibly access, but not the actual chemical traffic passing through the network with its unique kinetic constraints.

Because this level of detailed kinetic information was available for the COX-2 network, a different technique was needed. To this end, a method was developed to calculate *network pathway flux*, the chemical reaction flux through a given pathway in a network, given kinetic constraints, initial conditions, and, if desired, stochasticity. The method combines a graph theoretical analysis of the network to find paths between nodes with a modified PySB [12] integrator that returns integrated individual chemical reaction fluxes. A description of the method, its application to the CORM, and the insights derived from the analysis follows.

4.2 Calculating Network Pathway Flux

In this approach, a node edge graph is constructed in which each node is a set of reactants or products, and each edge represents a reaction connecting a reactant-product set. The edges are directed with the direction dictated by the net integrated reaction flux ($fluxk_f$ - $fluxk_r$) at a given time; these edges are therefore time dependent given the time dependence of the net reaction flux. We then determine all simple paths (paths with no repeating reactant-product nodes) connecting initial reactants (COX-2 + AA or COX-2 + 2-AG in the case of CORM) to final products (PG and PGG for CORM) via the net integrated flux directed edges, which are returned by the modified PySB [12] integrator. Simple paths are determined using the Python package `networkx` [106]. The overall process is outlined in Fig. 4.1. The relationship between paths then allows calculation of the proportion of flux each path contributes to the final product as the joint probability of all reactions that constitute that path (see Fig. 4.2 for an example).

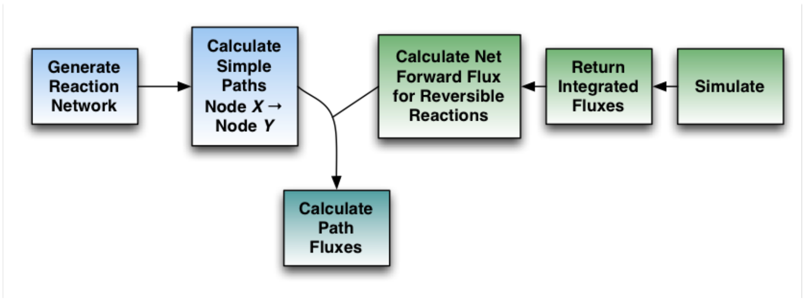

Generate Reaction Network
Calculate Simple Paths Node X → Node Y
Calculate Net Forward Flux for Reversible Reactions
Return Integrated Fluxes
Simulate
Calculate Path Fluxes

Figure 4.1: An overview of the method for calculating network pathway flux using a graph theoretic analysis to determine network paths and an integrator to return the integrated flux through specific chemical reactions.

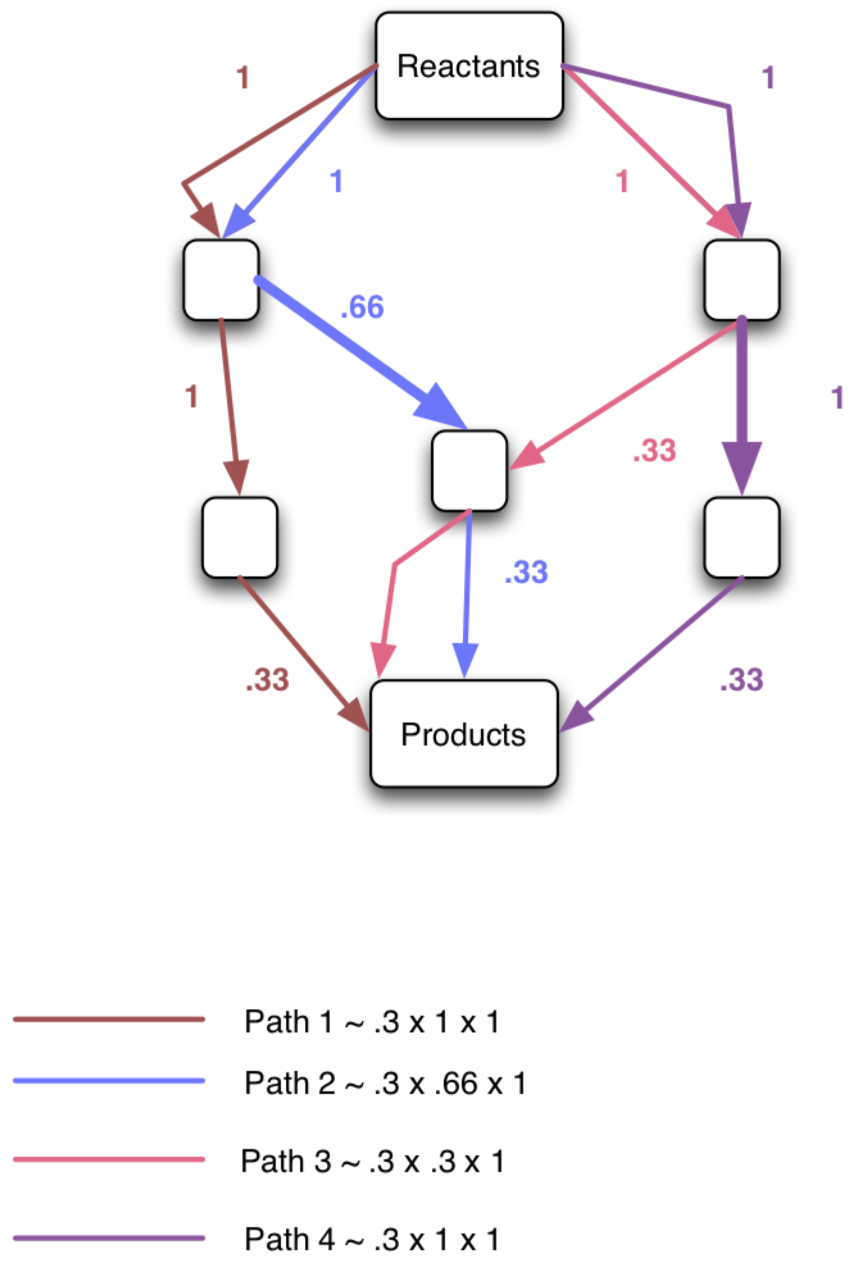


Figure 4.2: Calculating paths between reactants and products. Edges are directed towards the flow of the net integrated reaction flux (for bidirectional reactions) or the integrated forward reaction flux (for unidirectional reactions). Edge labels indicate the probability of selecting that reaction path into a node, calculated as the proportion of total flux leaving the node that came from a given reaction.

4.3 Applying Pathway Flux Analysis to CORM

All pathway fluxes were calculated for the first ten seconds of CORM catalysis after mixture with the substrates, a time chosen to match previous experimental work [1]. Path flux distributions were calculated for an ensemble of calibrated parameter values to quantify path flux uncertainty arising from parameter uncertainty. Path fluxes were also calculated over a range of initial substrate conditions, again chosen to match previous experimental work [1].

Our analysis indicates that there are six possible paths to produce PG (Fig. 4.3) and four possible paths to produce PG-G (Fig. 4.4) for all evaluated substrate concentration combinations. However, not all paths exhibit significant reaction flux during catalysis across all the concentrations. This occurs because paths in which binding of a species to the allosteric site precedes binding to the catalytic site are kinetically disfavored in CORM. As shown in Figs. 4.5 and 4.6, three paths dominate PG production and two paths dominate PG-G production. The dominant PG-producing paths (Fig. 4.5) include those with one or two intermediates, and the allosteric site empty or occupied by AA or 2-AG. Our results show that the dominant path is highly dependent on the substrate input concentrations (Fig. 4.7*A*). The presence of AA and 2-AG in the allosteric site enhances the production of PG [1]. The dominant PG-G-producing paths include one or two intermediates (Fig. 4.6) with the allosteric site empty or occupied by AA. The presence of AA in this site reduces the rate of PG-G production [1]. Similar to PG production, we also found that the flux through each dominant path for PG-G production is dependent on substrate concentration (Fig. 4.7*B*).

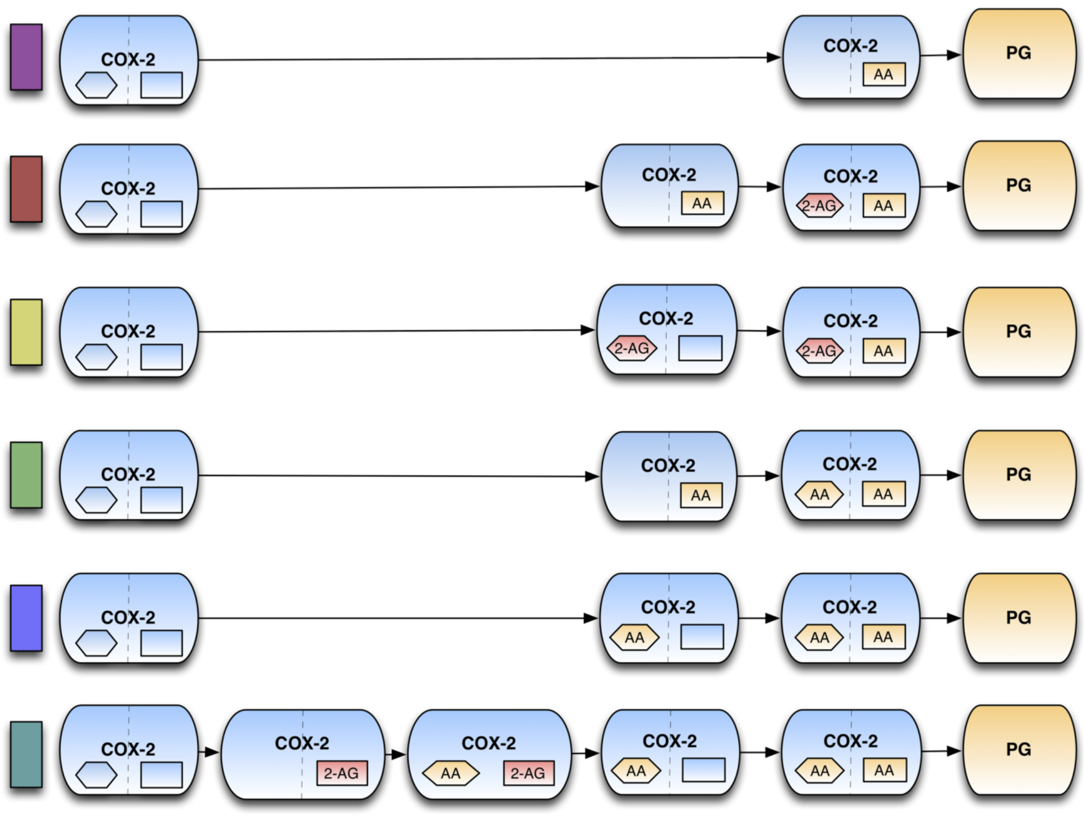


Figure 4.3: Possible paths between COX-2 and PG in CORM.

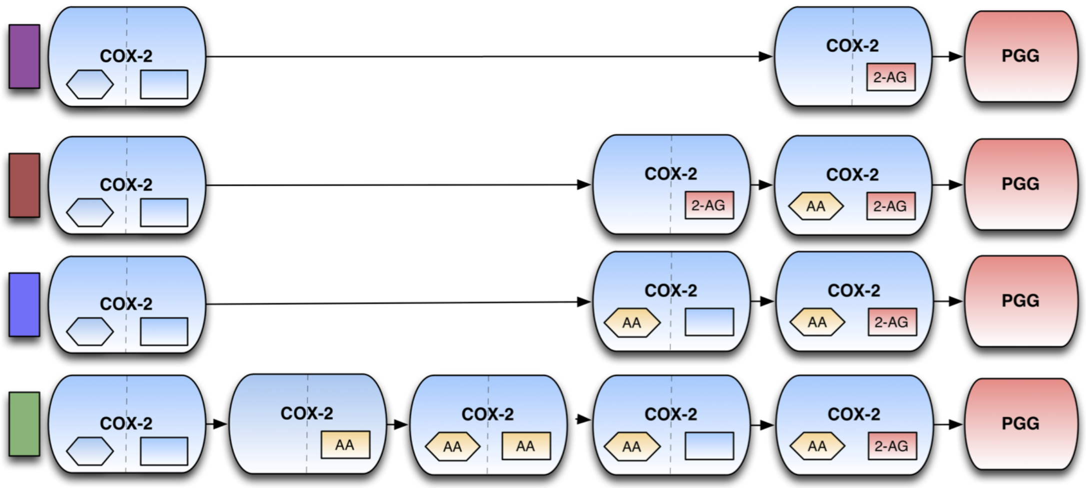


Figure 4.4: Possible paths from COX-2 to PGG in CORM.

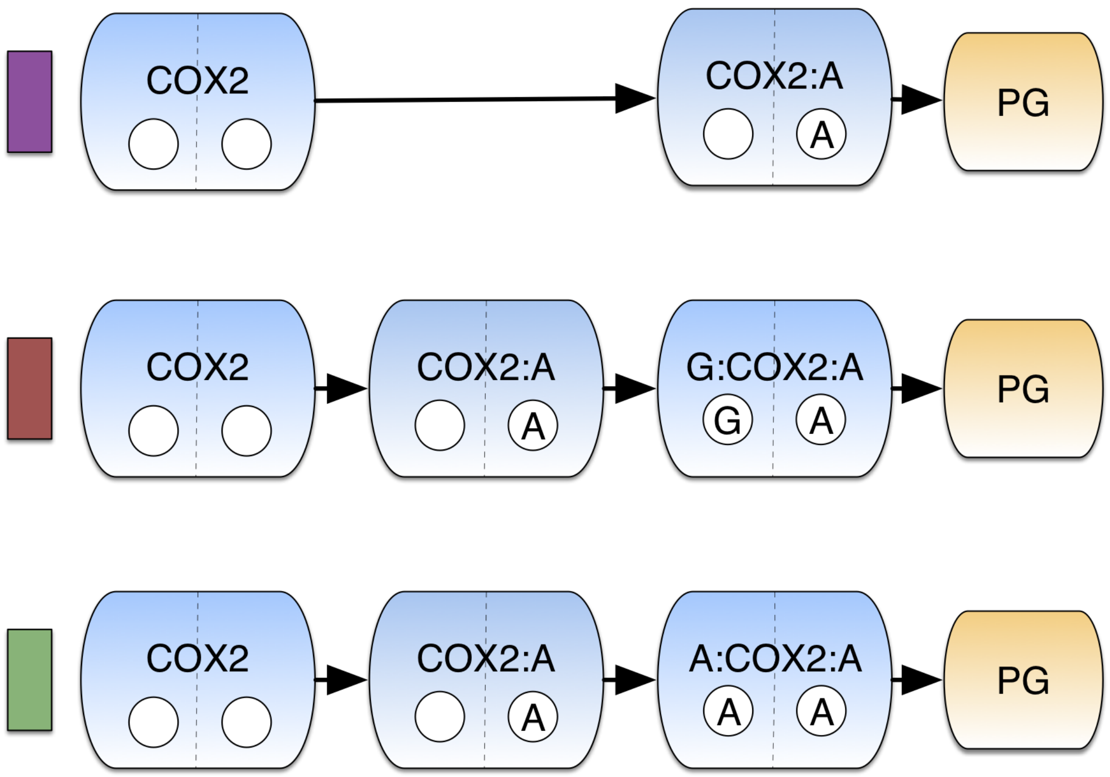


Figure 4.5: Dominant PG Production Paths in CORM. Colors correspond to path fluxes in Fig. 4.7*A*. (*D*) Dominant PG-G Production Paths in CORM. Colors correspond to path fluxes in Fig. 4.7*B*.

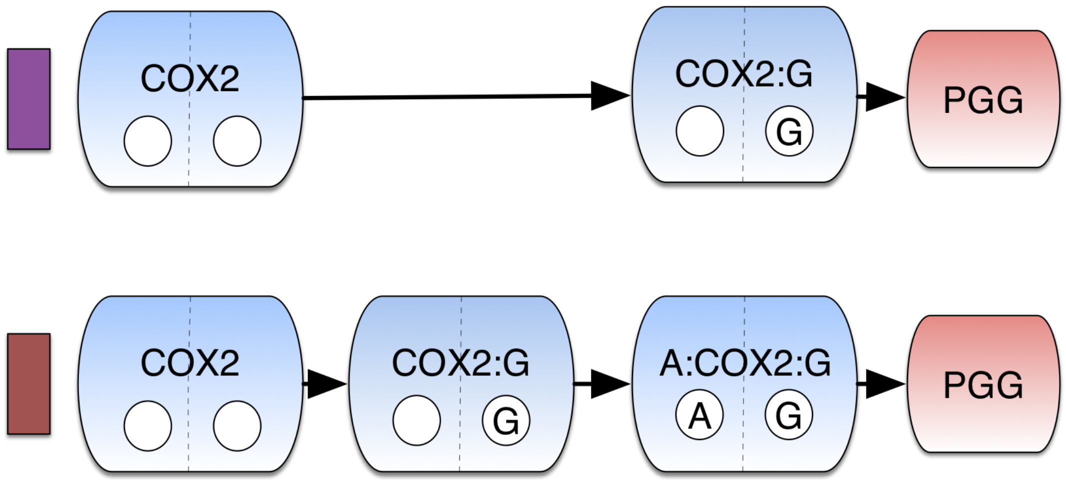


Figure 4.6: Dominant PG-G Production Paths in CORM. Colors correspond to path fluxes in Fig. 4.7*B*.

In the absence of 2-AG and at low (0.5 μM) AA, PG is produced without allosteric modulation (Fig. 4.7*A*, purple; purple-labeled path in Figure 4.5, top); as the concentration of AA increases, the proportion of PG produced with AA as an allosteric modulator also increases (Fig. 4.7*A*, green). When 2-AG is added to the system, PG production shifts to using 2-AG as an allosteric modulator (Fig. 4.7*A*, red), with this path favored to a greater extent as the concentration of 2-AG increases (Fig. 4.7*A*, lower plots). Even in the absence of 2-AG, about 20% of PG is produced by AA-modulated COX-2, and once even a small amount of 2-AG (0.5 μM) is added to the system, more than half of PG production occurs via a 2-AG or AA allosterically modulated path. In the presence of high concentrations of either modulator, as much as 90% of PG is produced via an allosterically modulated path.

Because 2-AG and COX-2 display substrate-dependent inhibition [1], the production of PG-G occurs via fewer paths than are available to PG. In the absence of AA, all PG-G produced is generated in the absence of an allosteric modulator (Fig. 4.7B, purple), because the intermediate with 2-AG bound in both catalytic and allosteric sites is not turned over. As AA is added to the system, the proportion of PG-G produced by the AA-modulated pathway (Fig. 4.7*B*, red) increases. Thus, in the range of tested substrate concentrations, the dominant mechanism of PG-G production depends entirely on the amount of AA present in the system. Compared to PG, a smaller proportion of PG-G produced by the system results from an allosterically regulated pathway because PG-G is only created via the AA-modulated species or the allosterically unbound species. Nevertheless, at high concentrations of AA, again as much as 90% of PG-G is produced by AA-modulated COX-2. For paths containing a species bound in the allosteric site, binding at the catalytic site followed by binding at the allosteric site is the favored mechanism.

At any given substrate concentration, the uncertainty arising from the calibrated kinetic parameter distributions never exceeds a 20% change in the percentage of product produced by a given path (Figs. 4.8- 4.12). We find that changes in substrate levels and their relative ratios have a much larger effect on the dominant reaction paths than changes in kinetic rates within the calibrated CORM parameter distributions. Overall, these findings suggest that variation of substrate concentrations in physiologically-relevant ranges has a significant impact on COX-2's mechanism of catalysis. In the next chapter I introduce a novel information theory based metric to quantify to what degree COX-2 utilizes the mechanisms available to it, and how this varies with substrate level.

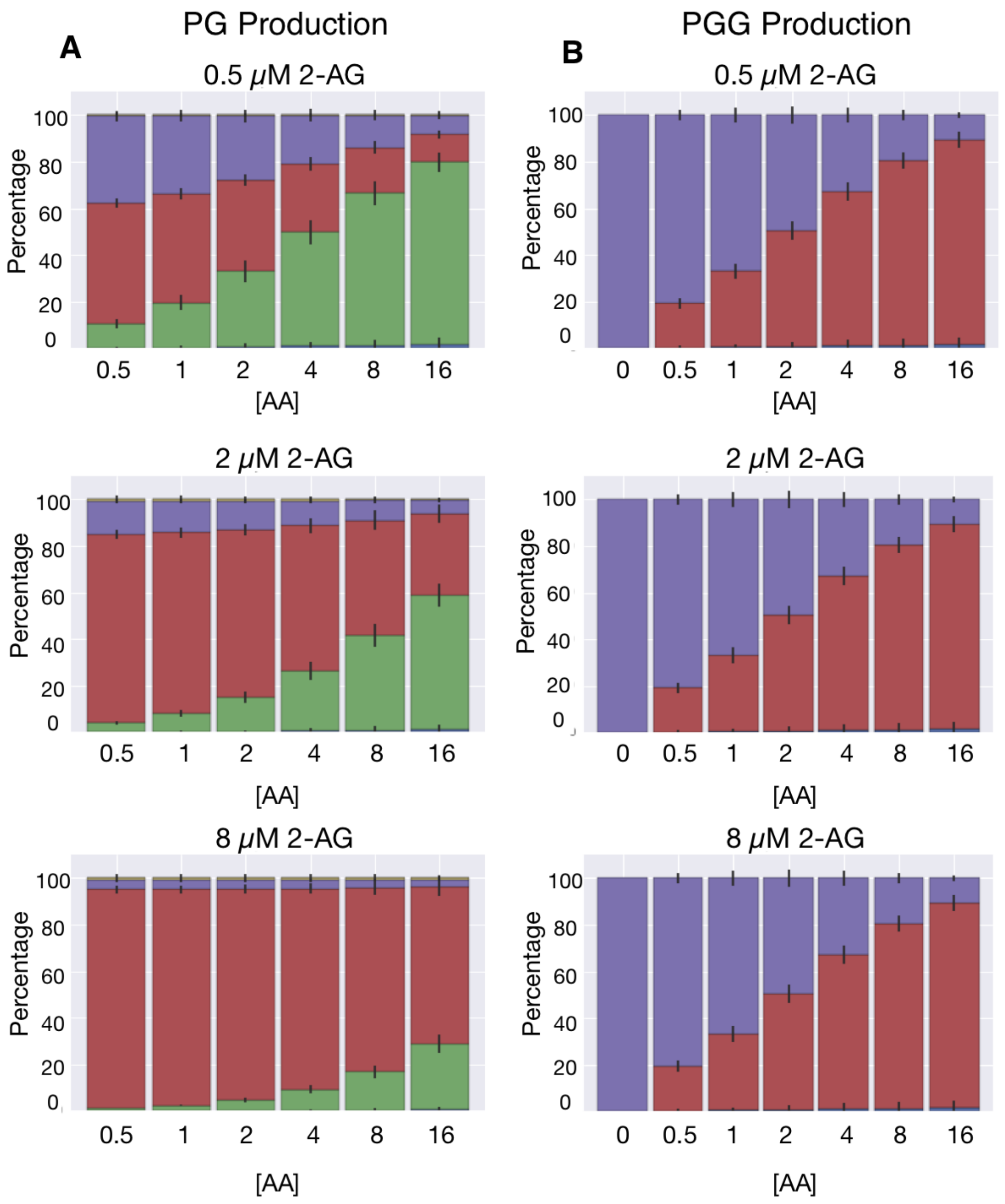


Figure 4.7: Concentration-dependent PG and PG-G production paths. (*A*) Dominant Reaction paths for PG Production Vary with AA and 2-AG Concentration. Each individual plot depicts the amount of flux through each path in 2.1*C* for a given concentration of 2-AG across varying concentrations of AA. Colors correspond to labeled paths in Fig. 4.5. The error bars in each plot indicates the flux variation resulting from inferred kinetic rates. (*B*) Dominant Mechanisms of PG-G Production Vary with AA Concentration. Each individual plot is at a given concentration of 2-AG. In all plots AA increases from left to right at concentrations of 0.5, 1, 2, 4, 8 and 16 μM in *A* and 0, 0.5, 1, 2, 4, 8, and 16 μM in *B*. Colors correspond to labeled paths in Fig. 4.6 The error bars in each plot indicates the flux variation from inferred kinetic rates.

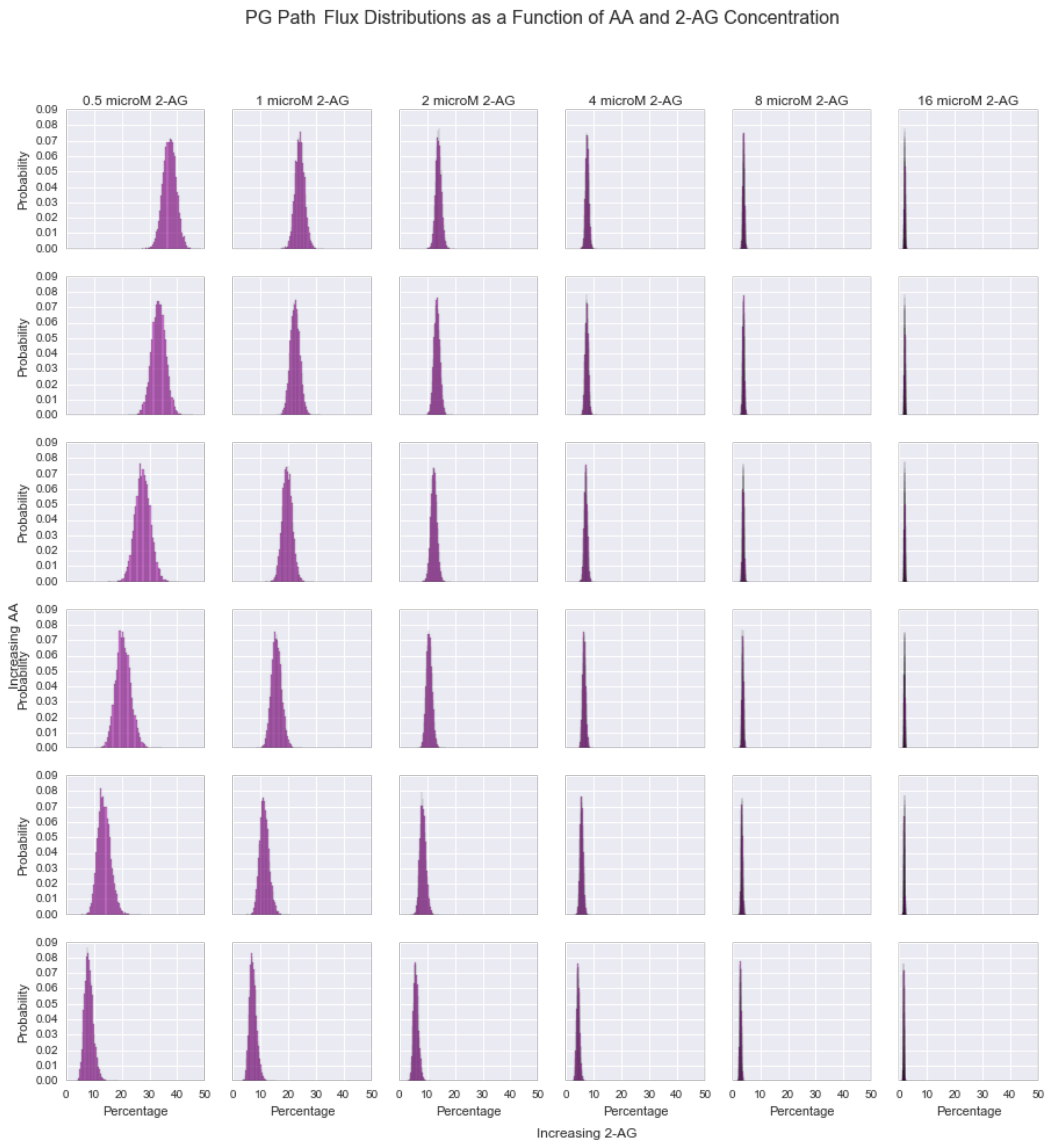


Figure 4.8: Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PG from the path COX-2 $\rightarrow$ COX-2:AA $\rightarrow$ PG at different starting AA and 2-AG concentrations.

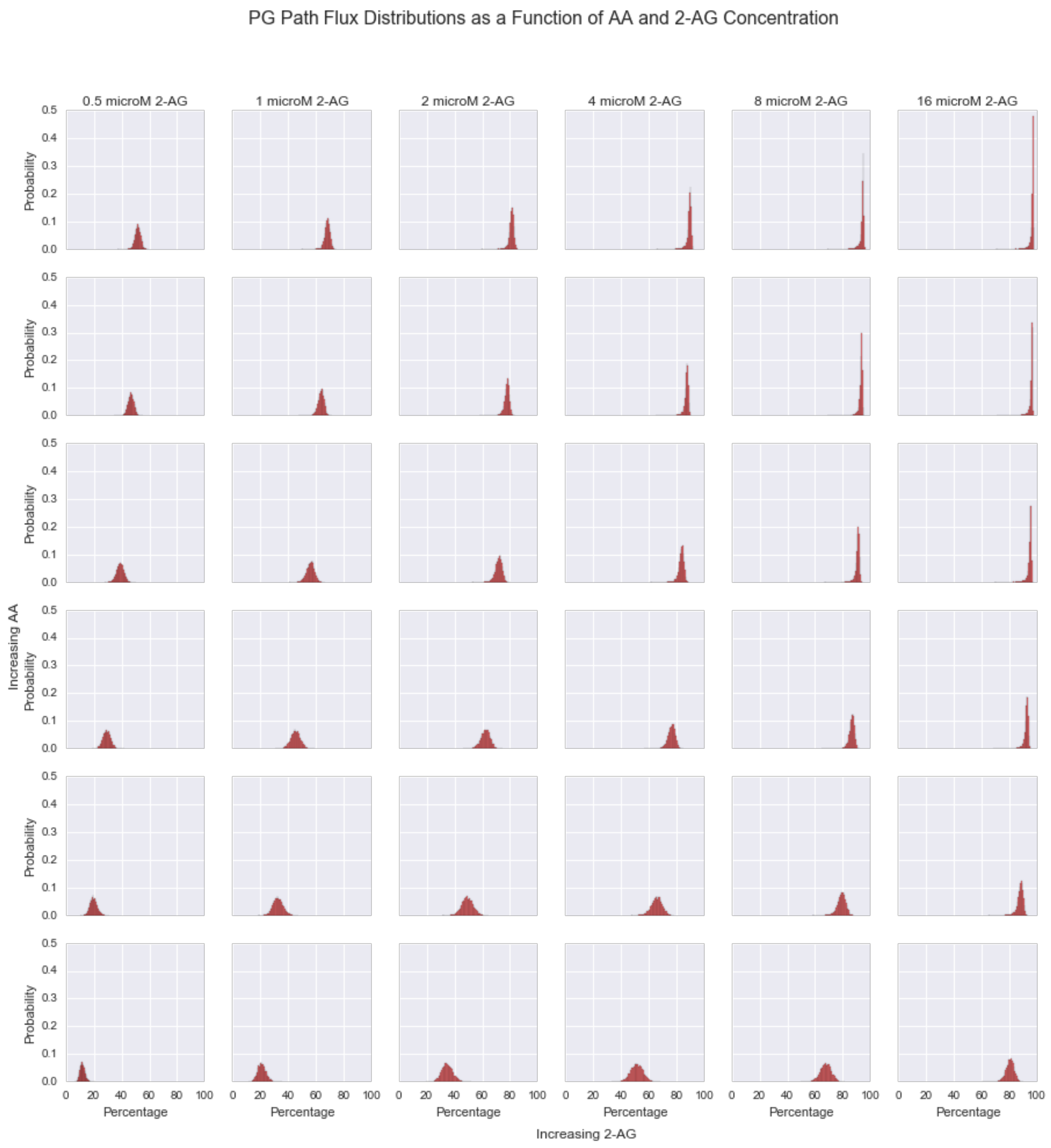


Figure 4.9: Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PG from the path COX-2 → COX-2:AA → 2-AG:COX-2:AA → PG at different starting AA and 2-AG concentrations.

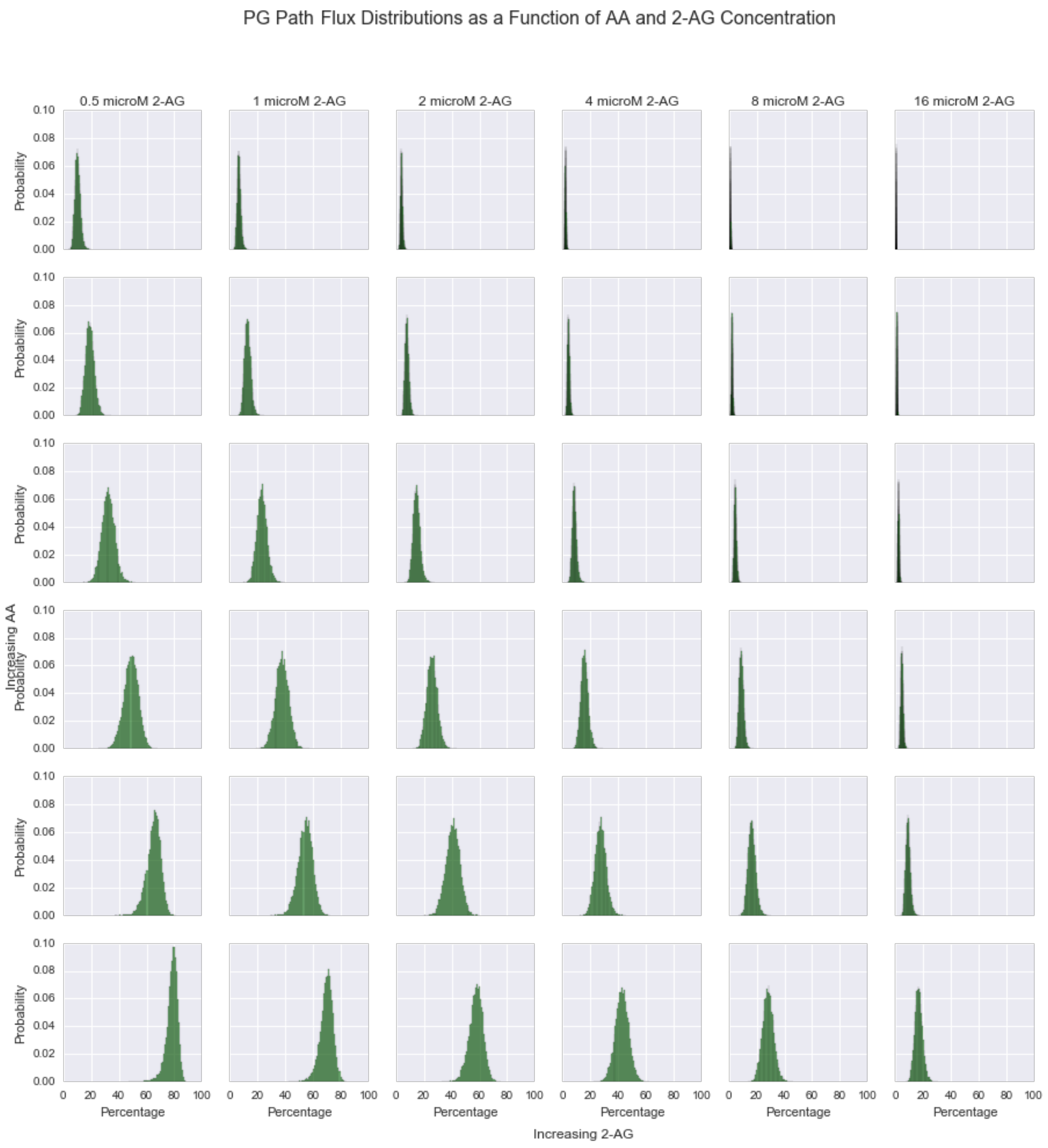


Figure 4.10: Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PG from the path COX-2 $\rightarrow$ COX-2:AA $\rightarrow$ AA:COX-2:AA $\rightarrow$ PG at different starting AA and 2-AG concentrations.

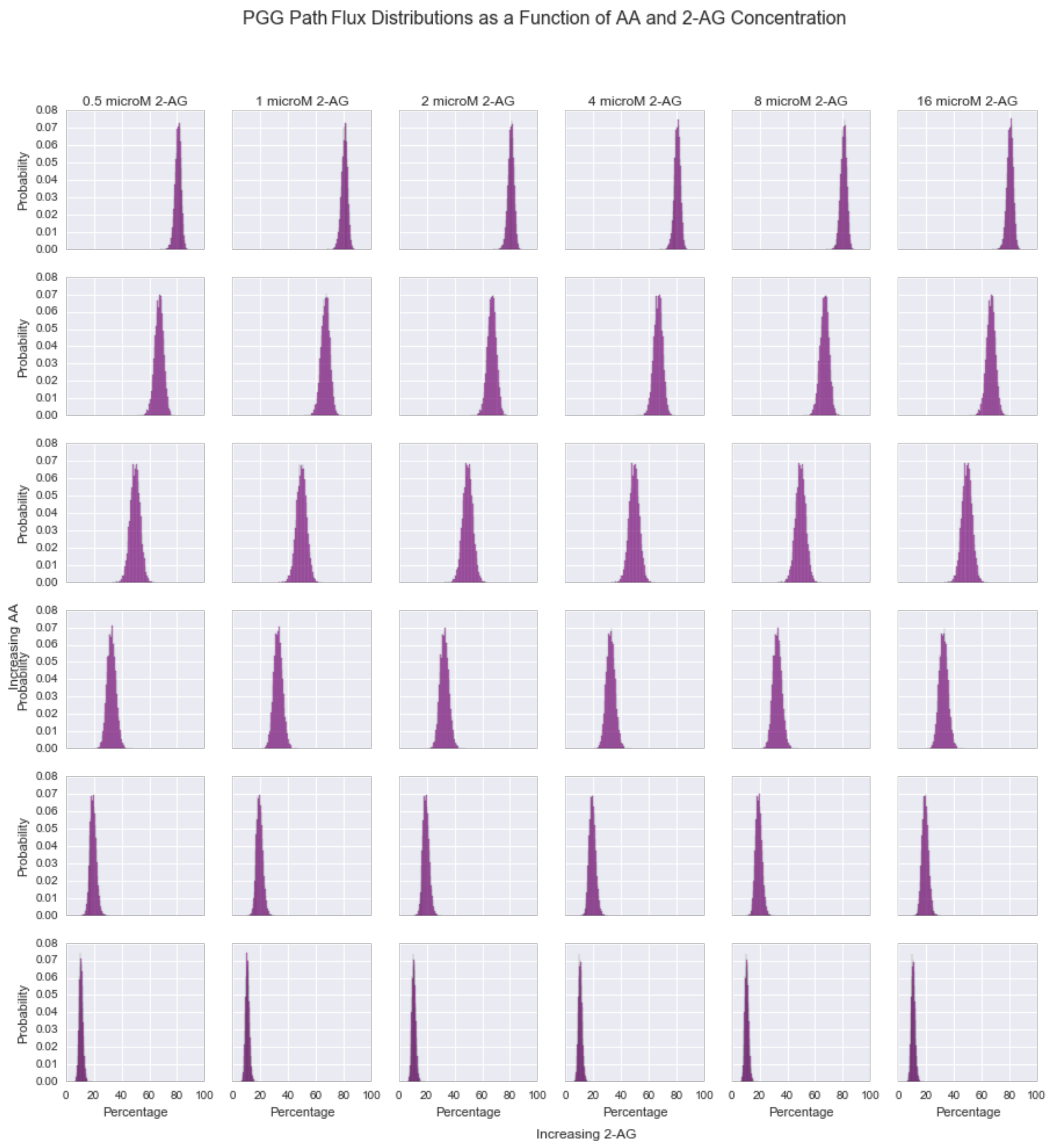


Figure 4.11: Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PGG from the path COX-2 $\rightarrow$ COX-2:2-AG $\rightarrow$ PGG at different starting AA and 2-AG concentrations.

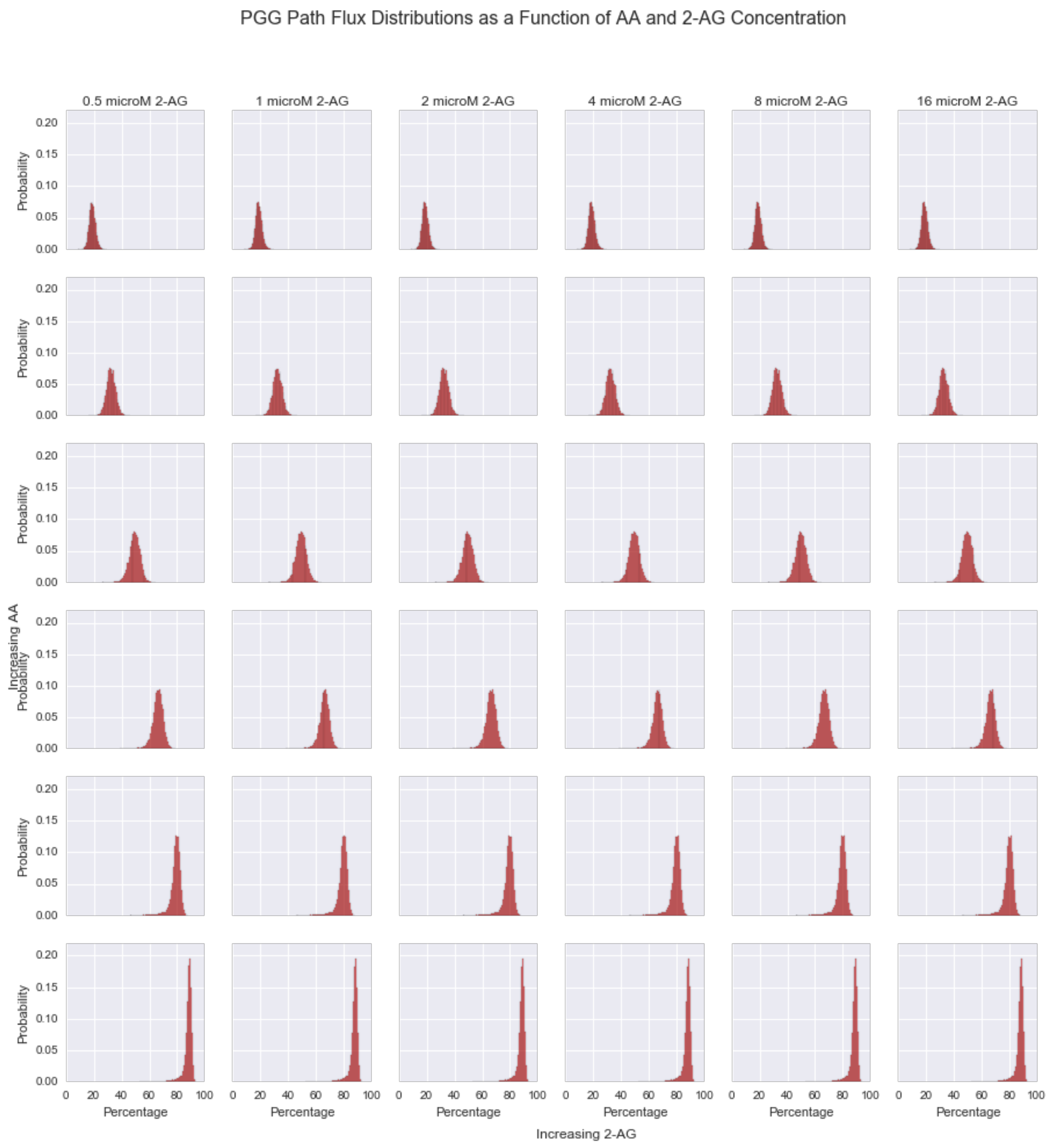


Figure 4.12: Distribution (arising from calibrated parameter uncertainty) of the percentage of the total flux to product PGG from the path COX-2 $\rightarrow$ COX-2:2-AG $\rightarrow$ AA:COX-2:2-AG $\rightarrow$ PGG at different starting AA and 2-AG concentrations.

Chapter 5

An Analysis of Information Transfer in the Multi-Input Multi-Output COX-2 Network

5.1 Motivation

Many biological signaling networks process multiple inputs and yield multiple outputs. Examples of multiple-input multiple-output (MIMO) biochemical systems include the mitogen-activated protein kinase (MAPK) network, which can respond to numerous ligands and yield a range of outputs including proliferation and differentiation[107]; the NF-κB pathway, which triggers pro- and anti-inflammatory responses to a variety of ligands [108]; and myriad metabolic networks, which respond to multiple substrates and allosteric regulators by producing energy and the building blocks of cellular components [109]. Recent work [3, 4, 110, 111, 112, 113, 114] has highlighted the fact that modulation of input concentrations in intracellular networks can yield markedly different outcomes. Despite this clear indication that MIMO systems are crucial to biological processes, few reports exist to date to explain how multiple inputs modulate reaction flux and information flow in a network to allow signal processing with a range of adaptive outputs.

To explore the properties of MIMO systems in biology, we chose to study the dynamics of cyclooxygenase-2 (COX-2), a key enzyme that controls the balance between pro- and anti-inflammatory signals in mammalian organisms. COX-2 lies at the interface of the eicosanoid and endocannabinoid signaling pathways [32, 33] and is itself the target of the widely used nonsteroidal anti-inflammatory drugs (NSAIDs). Although COX-2 is a structural homodimer, it behaves as a heterodimer. One subunit in the dimer harbors the catalytically active site, while the other subunit contains an allosteric site that modulates the overall activity of the enzyme [36, 37, 38]. An array of substrates, inhibitors, and allosteric modulators can bind to, and thus compete for, either site, giving rise to highly complex reaction kinetics [39, 40, 1, 41, 42, 43]. The various products from COX-2 activity drive multiple downstream pro- and anti-inflammatory processes that lead to diverse cellular fates including stress responses and apoptosis [44, 45, 46].

It is clear that COX-2 orchestrates a complex interplay between a variety of substrates (the enzyme *inputs*), various allosteric regulators, and the concentration of downstream products (the enzyme *outputs*) that control processes such as inflammation [44, 45, 46]. Previously, most studies of COX-2 function have used simplified models based on Michaelis-Menten kinetics [115]. Not surprisingly, these approaches have proved insufficient to capture the rich complexity of the COX-2 network of reactants, intermediates and products [1]. We posit that a systems approach to understand COX-2 mechanism will improve inhibitor design to achieve desired outcomes in clinical settings.

To explore the COX-2 MIMO signal processing mechanism, we used an information-theoretic approach [28] to understand the flow of information between network inputs, various intermediates, and the product outputs. This analysis reveals that competition between AA and 2-AG for the allosteric and active site generates highly complex concentration-dependence curves for COX-2 that are context-sensitive. In addition to providing insight into how COX-2 functions as a hub for the processing of inflammatory signals, our work suggests that our systems biochemistry framework provides useful information relevant to the study of other MIMO biological systems. This work also demonstrates that the extreme context-sensitivity of MIMO systems must be considered when attempting to modulate their behavior through targeted interventions.

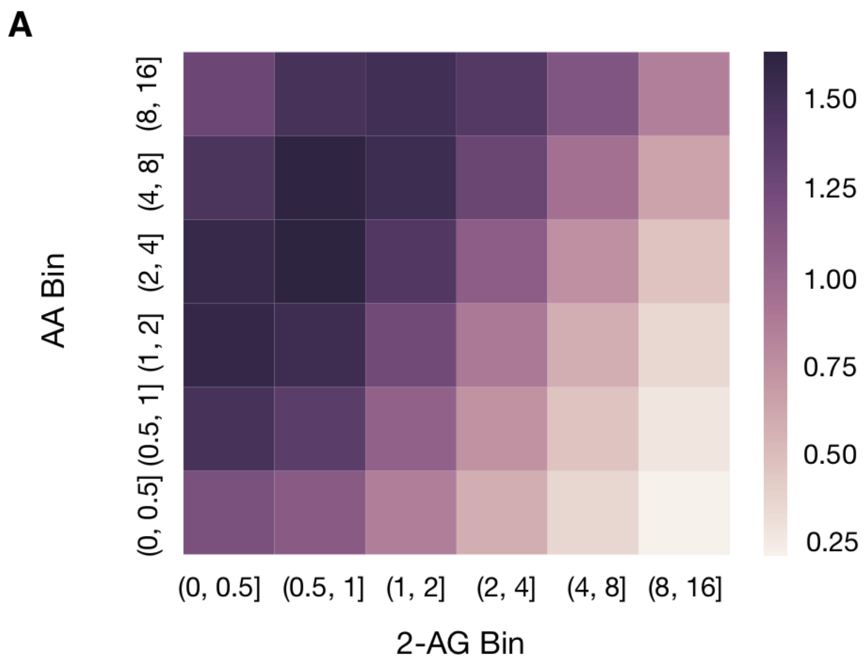
5.2 Pathway Entropy is Dynamic Across Input Concentrations

Calculating the flux through each path (as described in section 4.3) allows us to obtain information about the preferred sequences of reactions that the system executes while processing AA and 2-AG. However, these measurements do not provide an estimate of how chemical traffic (i.e. the flow of chemical signals in the network) is distributed throughout the network. To explore the distribution of biochemical network traffic, we introduce the *pathway entropy* to quantify the degree to which COX-2 utilizes multiple paths at different concentrations of substrates. Our definition of entropy, originally introduced by Claude Shannon [28] provides a measure of the uncertainty in a probability distribution across states as follows:

$$H = -\sum_{x=1}^{n} P(x_i)\log_2 P(x_i) \tag{5.1}$$

where H is entropy and $P(x_i)$ is the probability of any state x_i. To determine the degree of uncertainty associated with product production (the pathway entropy), we considered each pathway as a state and use the fraction of flux that a given pathway contributes to the product as a measure for the probability of that state. This analysis yields a measure of how evenly distributed production is across possible paths. In general, evenly distributed fluxes across paths in a network would maximize pathway entropy for a multi-path system.

Since the dominant paths vary with substrate concentration (Fig. 4.7), we would expect that pathway entropy would also vary. In Fig. 5.1 we present the pathway entropy dependence on input concentration for PG (Fig. 5.1*A*) and PG-G (Fig. 5.1*B*). The pathway entropy for PG production is highest at intermediate levels of AA and low levels of 2-AG, while the pathway entropy is highest for PG-G production at intermediate levels of AA and any level of 2-AG. These maxima correspond to states where the reaction flux is most spread across the possible paths from reactant to product (see Fig. 4.7*A*, top plot, center, and Fig. 4.7*B*, top plot, center). In contrast, in the lowest entropy states - low AA and high 2-AG for PG (Fig. 4.7*A*, bottom plot, far left) and low AA across the entire 2-AG spectrum for PG-G (Fig. 4.7*B*, bottom row), flux is concentrated in a single or a few paths. Reaction flow is thus highly distributed in some conditions yet highly concentrated in one path in other conditions. This finding suggests that MIMO networks utilize multiple execution modes across input concentrations. It also suggests that approaches to modulate or inhibit network activity, which focus on disrupting one or more of these paths, may need to be tailored to specific conditions. These behaviors could have physiological relevance. For example, high-entropy conditions with highly redundant path fluxes may require multiple targets for inhibition compared to a condition with low entropy.



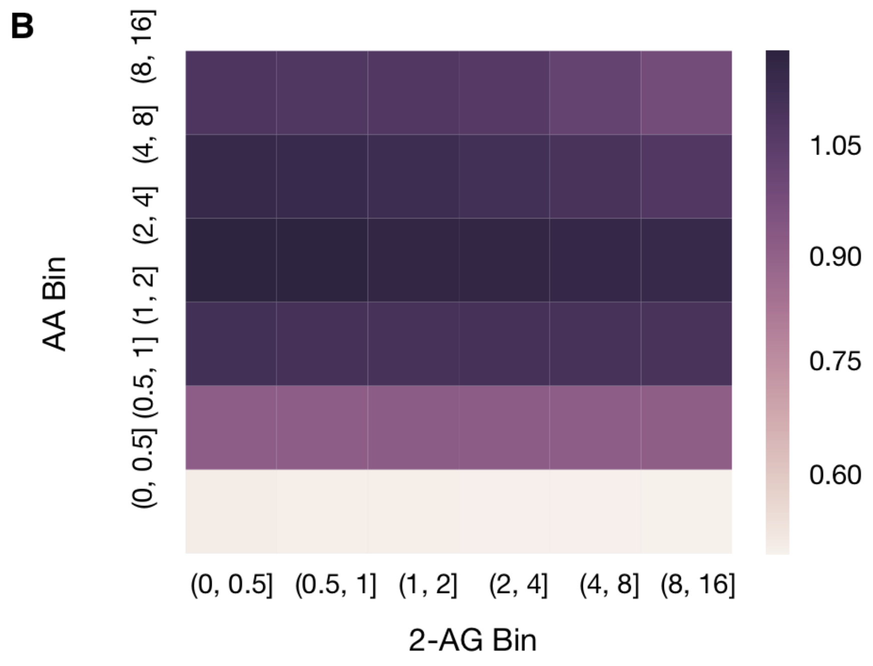


Figure 5.1: Pathway entropy within CORM. (*A*) Pathway Entropy for Production of PG. The intensity indicates the pathway entropy in units of bits. (*B*) Pathway Entropy for Production of PG-G. Units are the same as in *A*.

5.3 Input Output Behavior in CORM

The above findings on pathway entropy suggest a complex relationship between input concentrations, reaction intermediates, and product concentration in CORM. To understand these relationships, we next considered concentration-dependence curves derived from simulations using a fixed set of CORM kinetic parameters in which PG was calculated at increasing AA concentrations in the presence of random quantities of 2-AG (Fig. 5.2*A*) or PG-G was calculated at increasing 2-AG concentrations in the presence of random quantities of AA (Fig. 5.2*B*). Each data point was taken at steady-state (10 seconds) for consistency with experiments and previous work. Note that the presence of both substrates results in competitive inhibition with suppression of product formation from either one. Thus, the highest levels of output in each case occur when the concentration of the opposing substrate is low. These levels are similar for PG and PG-G because COX-2 utilizes the two substrates with similar catalytic efficiencies when they are present individually. As the concentration of the opposing substrate increases, competitive inhibition is partially balanced by positive allosteric modulation in the case of the conversion of AA to PG, but exacerbated by negative allosteric modulation in the case of the conversion of 2-AG to PG-G. Therefore, the suppression of PG-G formation is greater than that of AA formation as seen in the lower plateau level achieved in (Fig. 5.2*B*). In addition, the range of inputs over which the output varies depends significantly on which input/output pair is chosen (note the difference in that range in Fig. 5.2*A,B*). Clearly, variation of *both* inputs (e.g. changing AA *in addition* to changing 2-AG in Fig. 5.2*A*), results in significant variation in the outputs. Thus, while our simulations are deterministic, introducing uncertainty in the AA concentration generates a type of “extrinsic noise” in the relationship between 2-AG and PG-G (Fig. 5.2*B*), and *vice versa* for the impact of 2-AG on the relationship between AA and PG, Fig. 5.2*A*). This noise represents allosteric modulation in the network due to varying input concentrations.

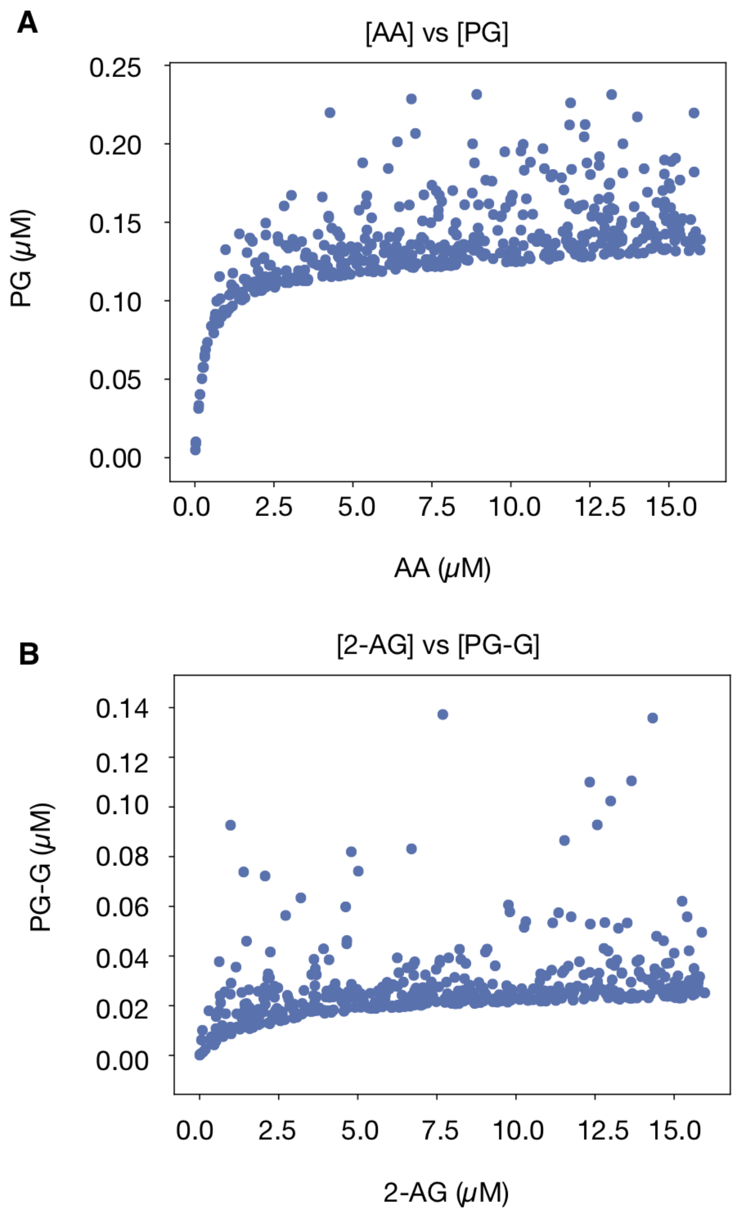


Figure 5.2: Input vs output plots for substrates and products in CORM. (*A*) Input vs Output plots for AA to PG. 2-AG varies randomly. All concentrations are measured at steady-state (10 seconds). (*B*) Input vs Output plots for 2-AG to PG-G. AA varies randomly. All concentrations are measured at steady-state (10 seconds).

5.4 Channel Capacity from Substrates to Products

To better understand how this output variation, combined with the shape of the concentration-dependence curves, influences the COX-2 reaction network, we applied an additional concept from information theory to measure dependence between inputs and outputs, namely the Mutual Information:

$$I(X;Y) = \sum_X \sum_Y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{5.2}$$

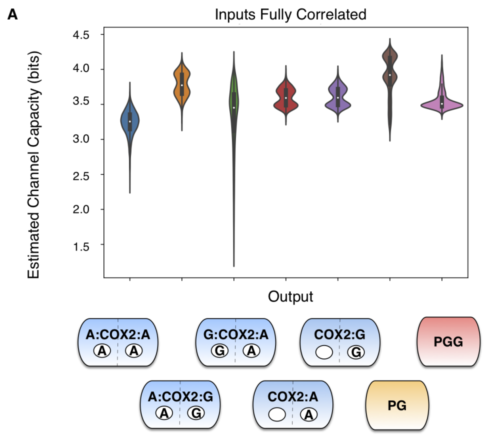
where X represents a given signal and Y the response to that signal [28]. Mutual information quantifies the degree to which one variable provides information about a second variable. Equivalently, it is a measure of how knowledge about one variable decreases uncertainty in the value of a second variable. For biological systems, quantifying mutual information is challenging because the input distribution is generally unknown. Previous work [29, 30, 31] has focused on estimating the "channel capacity," which is the maximum information attainable across all possible input distributions:

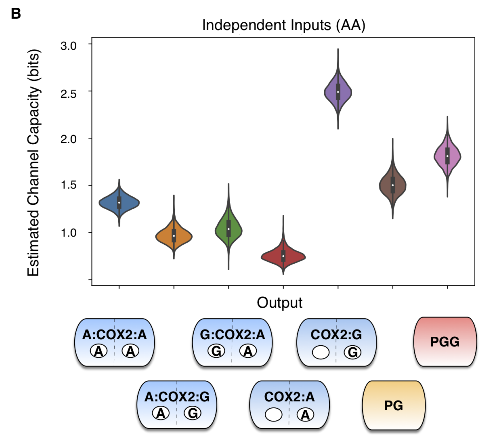
$$C = \sup_{p_x(x)} I(X:Y) \tag{5.3}$$

Note that any practical calculation provides a lower bound estimate for the channel capacity C, since only a finite set of input distributions is used to estimate I [31]. We calculated channel capacities using the approach and software published in Suderman *et al.* [31], which is similar to that used in Cheong *et al.* [29].

We applied this estimate to two different sets of simulations. In the first set of simulations, we considered a case where AA and 2-AG are perfectly correlated with each other; to do this, we sampled the AA concentration from a uniform distribution on $[0, 16\ \mu M]$ and set the 2-AG concentration to be exactly the same. In the second set, we *independently* sampled the input AA and 2-AG substrate concentrations from a uniform distribution on the interval $[0, 16\ \mu M]$. In each case, we sampled a total of 500 distinct input conditions and ran CORM simulations to 10s to agree with experiments and previous work [1]. The channel capacity was then estimated between the two different inputs (either AA or 2-AG) and every possible intermediate and product. The maximum theoretical channel capacity, $\log_2(500) \approx 9$ bits, would be obtained if each of the 500 inputs yielded a distinct response.

We repeated the channel capacity calculation for the top 5000 most probable parameter vectors from the calibrated parameter ensemble. This then allowed us to quantify the effect of kinetic parameter variation on channel capacities in the system. In total the analysis required approximately 1.5M CPU hours. An example of input data used for calculating channel capacities from AA to PG and 2-AG to PG-G for a single parameter set is shown in Fig. 5.2. Greater detail regarding the high performance computing methods used is provided in Chapter **??**.





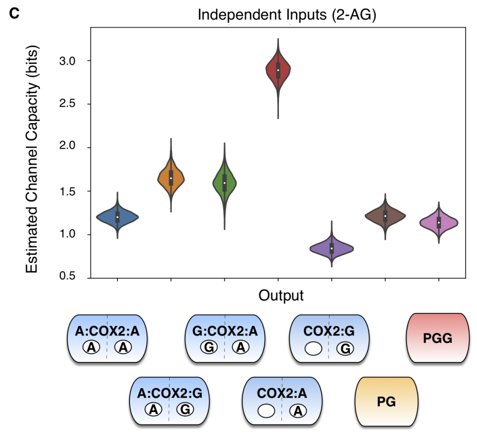


Figure 5.3: Estimated channel capacities from substrates to intermediates or products in CORM. (*A*) Estimated Channel Capacities from Input to intermediates and final products within CORM when levels of AA and 2-AG are strongly correlated (Pearson correlation coefficient = 1). Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration.(*B*) Estimated Channel Capacities from AA to intermediates and final products within CORM when AA and 2-AG are varied independently. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration.(*C*) Estimated Channel Capacities from 2-AG to intermediates and final products within CORM when AA and 2-AG are varied independently. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration.

5.5 COX-2 Integrates Information from Both AA and 2-AG

For ease of visualization, we estimated kernel densities of channel capacities given variation in calibrated kinetic parameters as shown in the violin plots in Fig. 5.3. In these plots, the data are represented by a central box plot that provides the mean, interquartile range and 95% credible interval, and the surrounding shape depicts the probability distribution, with wider regions indicating a higher probability. Because the input-output relationship in these simulations is deterministic, deviations from the theoretical maximum ($\approx$9 bits) arise from the two phenomena described above: either changes in the input do not really lead to significant changes in the output (*i.e.* the "flat" part of the concentration-dependence curves in Fig. 5.2) or the independent variation in one of the substrates generates variation in the output that is not due to the input being considered (*i.e.* the apparent noise in Fig. 5.2).

From Fig. 5.3, it is clear that the combination of these effects significantly reduces the observed channel capacities from the theoretical maximum. The highest observed value for any of the input/output pairs (AA to PG, 2-AG to PG-G, etc.) is at most half of the theoretical maximum (less than 4.5 bits). When input values are perfectly correlated ([AA] = [2-AG]), Fig. 5.3*A*, the channel capacity between the (correlated) inputs and the outputs is between 3 and 4.5 bits (depending on the parameters), indicating that, while not perfect, the concentration-dependence curves allow for high levels of information flow between inputs and outputs. It is interesting to note that the uncertainty in the kinetic parameters leads to some variation in the calculated channel capacities; since the inputs here are correlated, this variation is due to changes in the shape of the concentration-dependence curves between data sets. Many channel capacities in the correlated case are bimodal, suggesting that two specific concentration-dependent curve shapes are most likely.

When the inputs are varied independently, channel capacity values decrease even further (Fig. 5.3*B* and *C*). The channel capacity between AA and PG or PG-G is generally less than 2 bits, and the channel capacity between 2-AG and those outputs is generally less than 1.5 bits. This could occur for two reasons. First, a lack of correlation could result in less

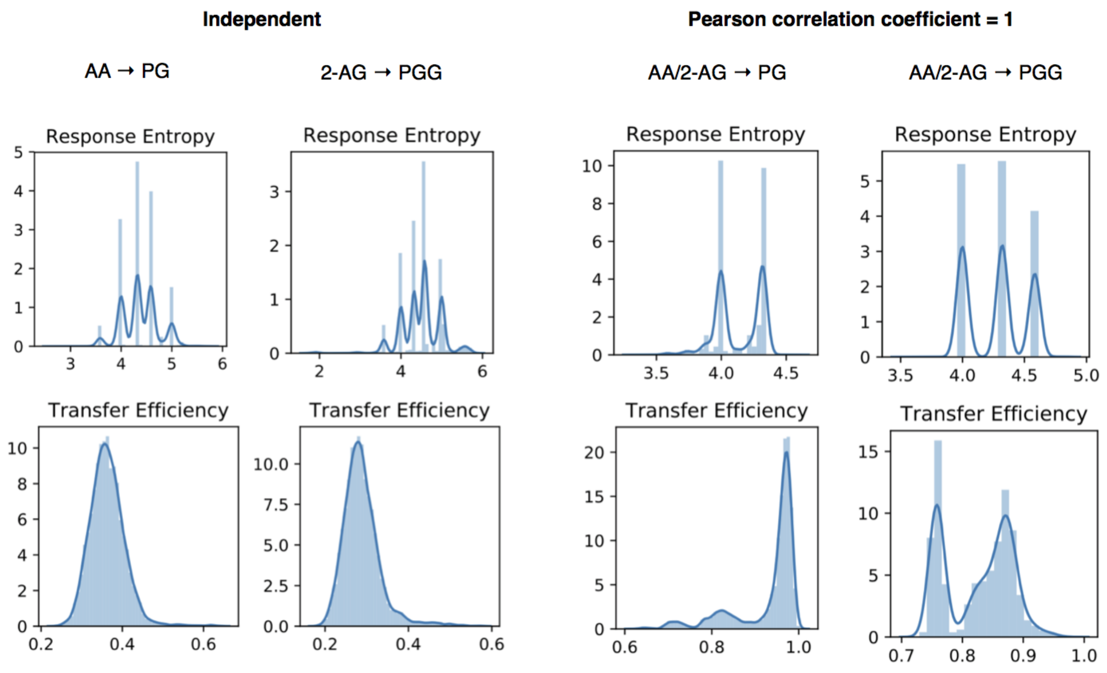


Figure 5.4: Distributions of response entropy under different input correlation and signal/response pairs. While the response entropy spans a similar range with both independent and correlated inputs, the transfer efficiency increases with correlation.

entropy in the response (*i.e.* less uncertainty in the value of the product). Since the mutual information is limited by the response entropy (eq. 5.2, [28, 29, 31]), this would cause a decrease in the mutual information. However, if the response entropy remains constant when there is no correlation between inputs, then mutual information can only decrease if information transfer through the network is less efficient. As shown in Fig. 5.4, the response entropy does not differ between the independent and correlated cases, indicating that independent variation in one of the inputs while the other input is known has a large effect on the output. In other words, COX-2 is truly an integrator of these signals, since accurate determination of the substrate concentrations given the output is considerably more difficult if the two substrates are independently varied.

Since perfect correlation and complete independence represent only the two extremes of the relationship between AA and 2-AG concentration, we also investigated the behavior of the system when the inputs exhibit moderate correlation (Pearson correlation coefficient

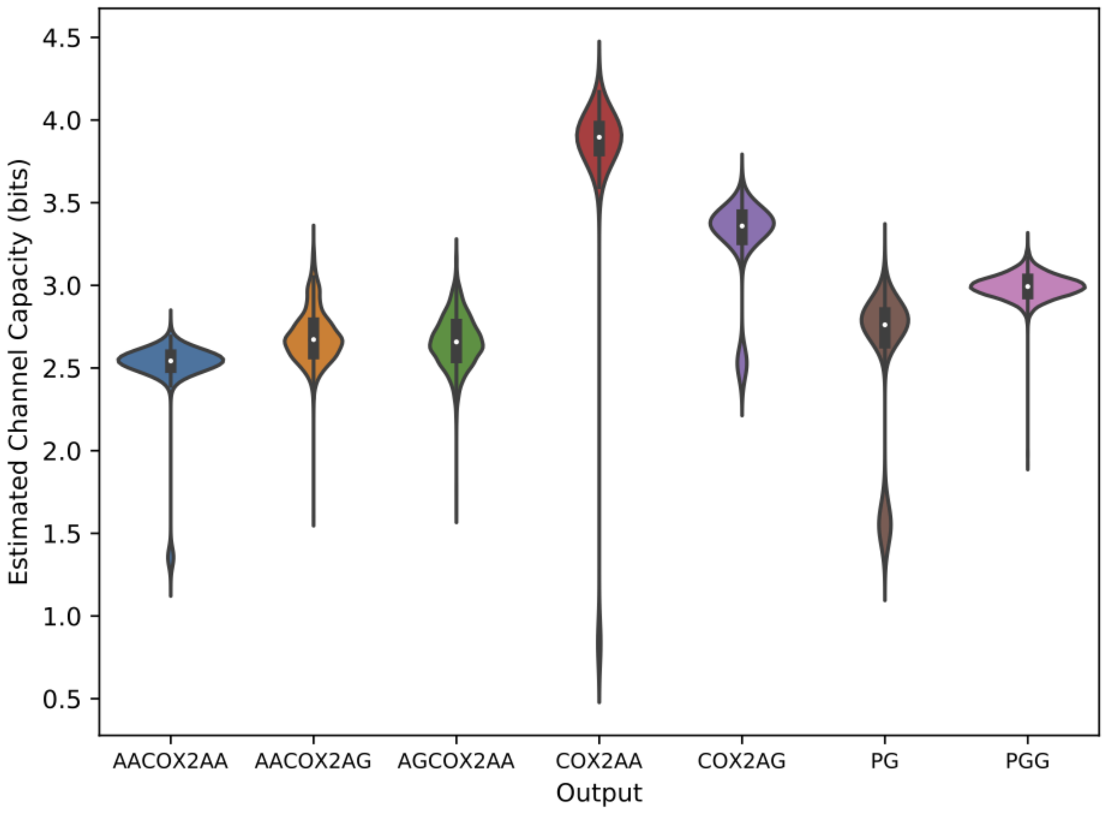


Figure 5.5: Sum of channel capacity from AA and 2-AG to intermediates and final outputs when inputs are semi-correlated (Pearson correlation coefficient = .5). Distributions arise from uncertainty in calibrated parameter values.

= 0.5), and when the inputs are consistently present in a 2-to-1 AA-to-2-AG ratio (Fig. 5.5 and Fig. 5.6). The behavior when input ratios were fixed was similar to that for the correlated values (when the input levels were fixed equal to each other); channel capacities were again higher than in the independent case and the effect of kinetic parameter variation on channel capacity was higher. When the inputs are moderately correlated, the system is still able to obtain high channel capacities for some kinetic parameter sets, although the overall distribution of channel capacities shifts to lower values compared to when input correlation is perfect, further confirming COX-2 input integration.

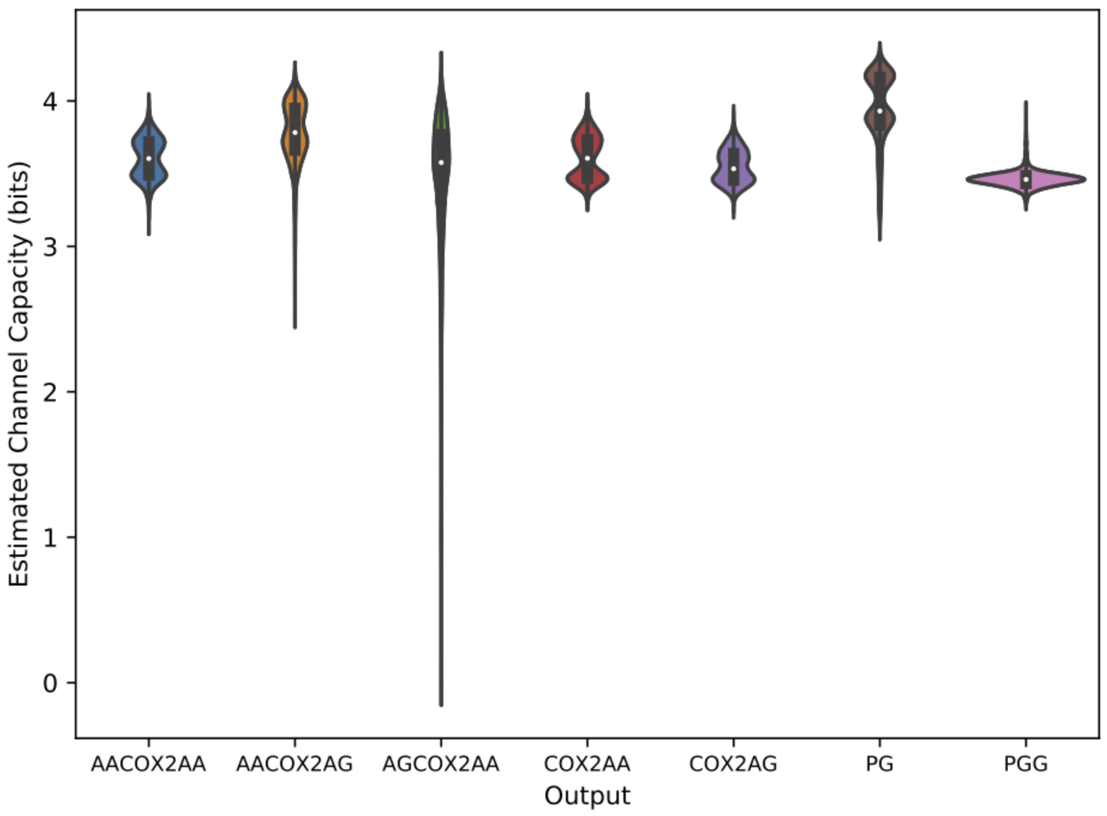


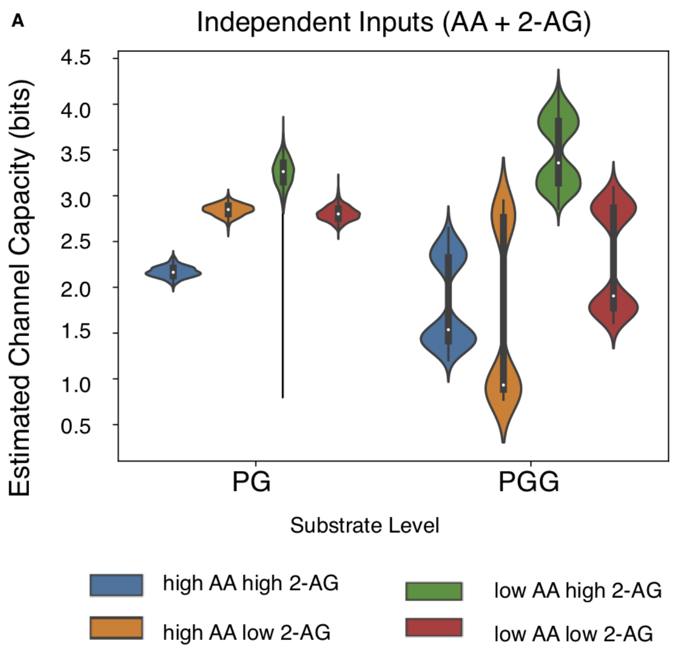
Figure 5.6: Channel capacity from AA and 2-AG to intermediates and final outputs when inputs are present in a 2 to 1 AA to 2-AG ratio. Distributions arise from uncertainty in calibrated parameter values.

5.6 Information Flow is Dictated by Substrate Concentration

We next tested whether the channel capacity between substrates and products varies with substrate level. We binned the input data into four quadrants (high or low values of either substrate) and calculated the channel capacity between inputs and outputs independently for each quadrant; input ranges were otherwise identical to those used for the calculations described above. Low substrate values spanned 0-8 μM and high substrate values 8-16μM.

Both independently varied inputs (Fig. 5.7*A*) and correlated inputs (Fig. 5.7*B*) yielded estimated channel capacities that were significantly different between the different regions of input space. In addition to differences in PG and PG-G channel capacity, we found that the distribution of information that passed through different intermediates changed with substrate concentration (Fig. 5.8 and Fig. 5.9); certain paths to product had greater information transfer capacity at particular levels of substrates. This echoes findings from our pathway analysis (Figs. 4.7 and 5.1), indicating that changes in substrate concentration result in significant changes in how the enzyme executes its catalytic mechanism. Interestingly, we found no detectable correlation between the flux through a pathway and the mutual information between an input and an intermediate in that path (Fig. 5.10 and Fig. 5.11). We leave further investigation of the relationship between information transfer and actual physical reaction fluxes for future work.

Splitting the input space into different quadrants also revealed signficant variation between different parameter sets, with most distributions showing significant bimodality across parameters (Fig. 5.7). This suggests that both the shape of the concentration-dependence curves, and the impact of “extrinsic noise” due to variation of one substrate independent of another, varies across parameter sets. Since all of these parameter sets are equally consistent with experimental data [1], this suggests that multiple modes of information flow are available to the COX-2 reaction network without significant changes to the core functionality of the enzyme.



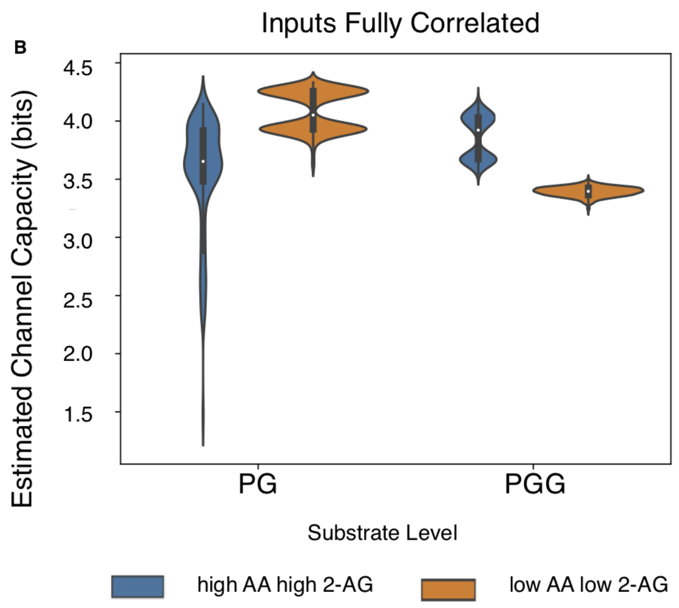


Figure 5.7: Effect of substrate level on estimated channel capacities between substrates and products in CORM. (*A*) Total Estimated Channel Capacity from AA and 2-AG combined to products across regions of substrate space. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration. (*B*) Estimated Channel Capacities from input to products when levels of AA and 2-AG are perfectly correlated across regions of substrate space. Distributions in the channel capacities arise from uncertainty in the kinetic parameter values after model calibration.

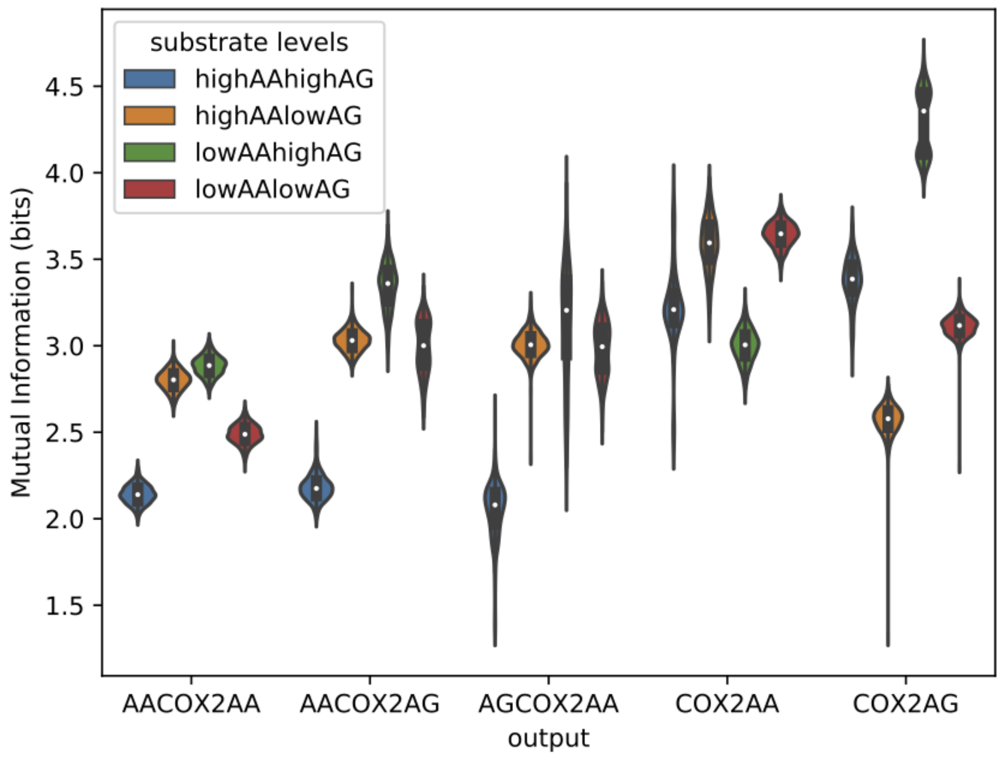


Figure 5.8: Sum of channel capacity from AA and 2-AG to intermediates when inputs are varied independently in different regions of substrate space. Distributions arise from uncertainty in calibrated parameter values.

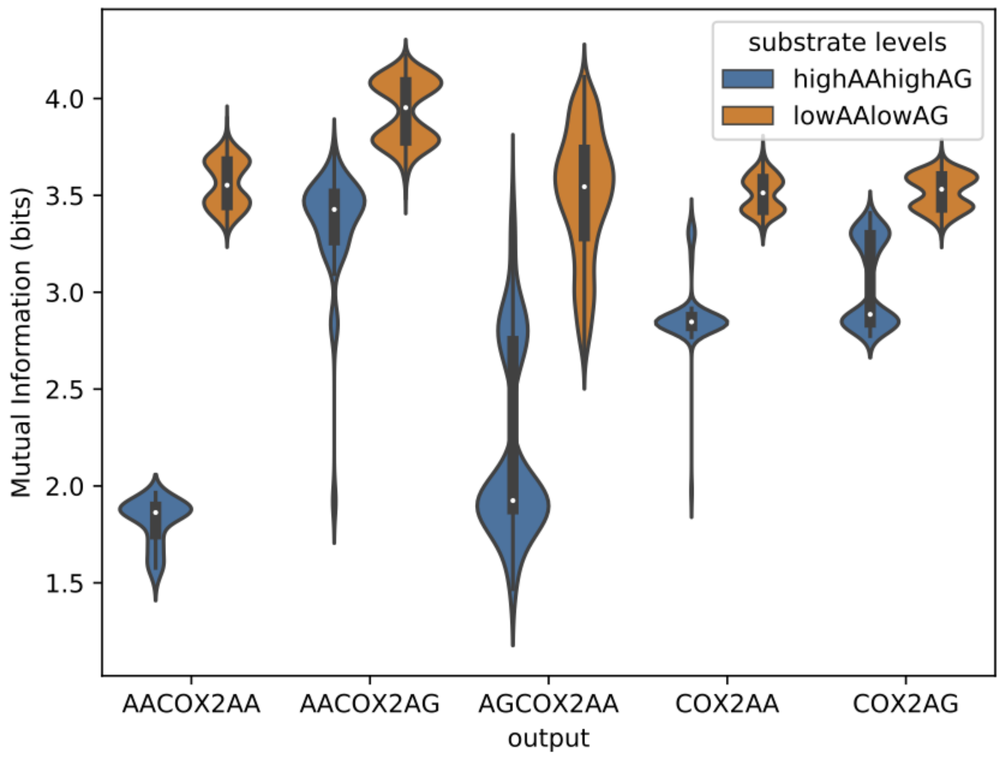


Figure 5.9: Channel capacity from AA and 2-AG to intermediates when inputs are varied correlated (Pearson correlation coefficient = 1) in different regions of substrate space. Distributions arise from uncertainty in calibrated parameter values.

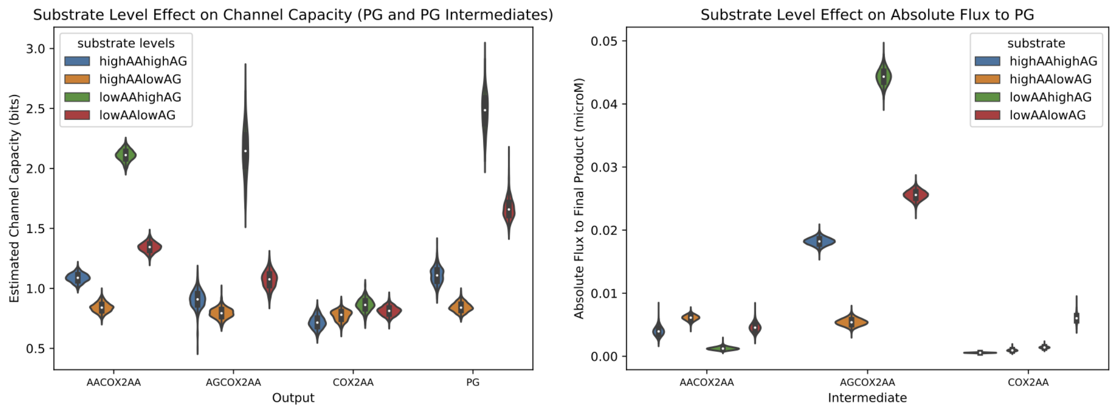
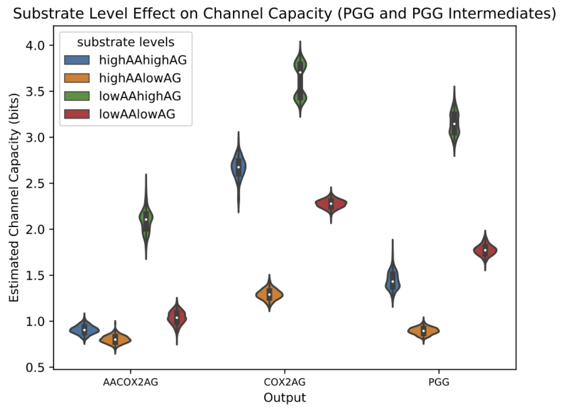


Figure 5.10: Left: channel capacity between independent inputs and PG intermediates at different substrate levels. Right: absolute flux between independent inputs and PG intermediates at different substrate levels.



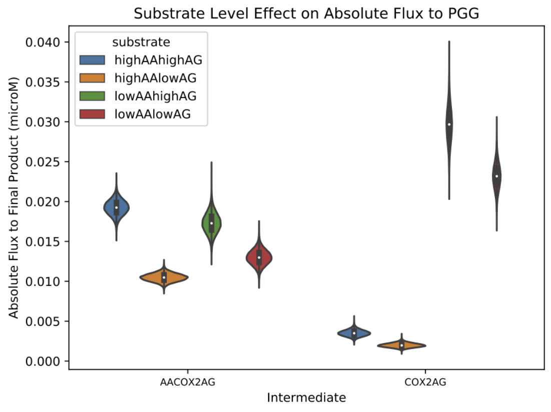


Figure 5.11: Left: channel capacity between independent inputs and PGG intermediates at different substrate levels. Right: absolute flux between independent inputs and PGG intermediates at different substrate levels.

Chapter 6

Perturbing the COX-2 Network: A Sensitivity Analysis of CORM

6.1 Motivation

The analyses in the preceding chapters are largely concerned either with model construction and calibration to experimental data, or with understanding how the system itself functions (*e.g.* the pathway and information theoretical analyses). In many cases, however, the ultimate goal of modeling and analyzing a biological system is to predict how best to change some output of interest in that system. For the COX-2 network previously discussed, modifying the final system outputs (prostaglandins and prostaglandin glycerols) would be the most obvious means to change the inflammatory phenotype tied to those products, and a greater understanding of the effect of different perturbations on the network could deepen the current understanding of why different nSAIDs do (or do not) affect inflammation *in vivo*.

Sensitivity analysis, discussed briefly in the introduction, is designed specifically to address the question of how a given output changes in response to variation in other model variables. For a model like CORM, in which there is some level of uncertainty in actual system input variables (uncertainty that has been quantified by the calibration methods used), global sensitivity analysis to assess the sensitivity of a given output across the entire range of plausible input values are preferable to local sensitivity analysis methods that restrict analysis to individual locations in input space. The specific global sensitivity analysis method used to assess sensitivities in CORM, Random Sampling High Dimensional Model Representation (RS-HDMR) is briefly introduced in the next section.

6.2 Introduction to Sensitivity Analysis with RS-HDMR

Random Sampling High Dimensional Model Representation (RS-HDMR) [116, 117, 118] is a global sensitivity analysis method suitable for discerning the fraction of output variation that can be tied to a specific input (or combination of inputs). Crucially, RS-HDMR can be applied to to models in which input variables are correlated, a challenging regime for many other methods [119].

As a variance-based sensitivity analysis method, RS-HDMR quantifies the sensitivity index of an output Y to a particular input X_i as a reduction in the variance of Y:

$$S_i = \frac{V_i}{V(Y)} = \frac{Var(E(Y|X_i))}{Var(Y)} \tag{6.1}$$

In addition, higher order interaction indices may be calculated to assess the sensitivity of an output to simultaneous variation in multiple inputs. For example, the second order interaction indices may be calculated as follows:

$$S_{ij} = \frac{V_{ij}}{V(Y)} = \frac{Var(E(Y|X_i,X_j)) - V_i - V_j}{Var(Y)} \tag{6.2}$$

Just as the variance in Y may be decomposed, Y itself (*f(x)*) may also be decomposed in a similar manner:

$$f(x) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{1\le i<j\le n} f_{ij}(x_i,x_j) + \cdots + f_{1,2\cdots n}(x_1,\cdots,x_n) = f_0 + \sum_{j=1}^{2^n-1} f_{p_j}(x_{p_j}) \tag{6.3}$$

When the inputs are independent, the component functions are mutually orthogonal, and the decomposition is unique. When this is the case, the function decomposition can be related to a parallel decomposition of the unconditional variance in Y:

$$V(Y) = \sum_{i=1}^{n} V_i + \sum_{1\leq i<j\leq n} V_{ij} + \cdots + V_{1,2\cdots n} = \sum_{j=1}^{2^n-1} V_{p_j} \tag{6.4}$$

When the inputs are not independent, the unconditional variance in Y can instead be decomposed into the sum of the covariances $Cov(f_{p_j}, Y)$ and the averaged square error ε^2:

$$V(Y) = \sum_{j=1}^{n_p} Cov(f_{p_j}, Y) + (\varepsilon, \varepsilon) = \sum_{j=1}^{n_p} [Var(Y) + Cov(f_{p_j}, \sum_{k=1,k\neq j}^{n_p} f_{p_k})] + \varepsilon^2 \tag{6.5}$$

Each covariance term $Cov(f_{p_j}, Y)$ is composed of a structural piece, $Var(Y)$, which is always positive, and a contribution from correlation, $Cov(f_{p_j}, \sum_{k=1,k\neq j}^{n_p} f_{p_k})$ which may be either positive or negative. When the inputs are independent, the calculation of $V(Y)$ simplifies to only include the structural piece, and the previously enumerated sensitivity indices are sufficient to characterize the sensitivity of this variance to changes in the output. In cases when the inputs are not independent, the sensitivity indices may be redefined to separate the contributions from structure and correlation:

$$S_{p_j} = S^a_{p_j} + S^b_{p_j} \tag{6.6}$$

where

$$S^a_{p_j} = Var(f_{p_j})/V(Y) \tag{6.7}$$

is the structural sensitivity index and

$$S^b_{p_j} = \frac{Cov(f_{p_j}, \sum_{k=1,k\neq j}^{n_p} f_{p_k})}{V(Y)} \tag{6.8}$$

is the correlation sensitivity index.

In contrast to classical sensitivity analysis methods [119] which directly sample inputs and outputs from the original system, RS-HDMR for reasons of efficiency first fits a meta-model to the input-output data available for a system and then uses this meta-model to calculate the sensitivity indices.

6.3 Constructing an HDMR Meta Model of CORM

CORM contains eighteen parameters (catalytic rates and equilibrium constants) for which the effect of a perturbation on outputs of interest may be tested. In addition, the effect of perturbing substrate levels (AA and 2-AG) was also tested in this analysis. Each parameter to be tested was assigned an HDMR variable name, listed in Table 6.1. CORM parameter names are identical to those used in Table 2.2. An svr-based HDMR model utilizing a radial basis function (rbf) kernel was built to model the relationships between inputs and outputs in CORM. To measure error between the HDMR model and the original model, three measurements were used. The first, the relative absolute average error (RAAE) is defined as:

$$RAAE = \frac{\frac{1}{N}\sum_{s=1}^{N} |\hat{y}_i^{(s)} - y_i^{(s)}|}{\sigma_i} \tag{6.9}$$

where σ_i is the standard deviation of the N testing or training data $y_i^{(s)}$ and $\hat{y}_i^{(s)}$ is the HDMR approximation of $y_i^{(s)}$. The second measure of error, relative maximum absolute error (RMAE) is defined as:

$$RMAE = \frac{\max_s |\hat{y}_i^{(s)} - y_i^{(s)}|}{\sigma_i} \tag{6.10}$$

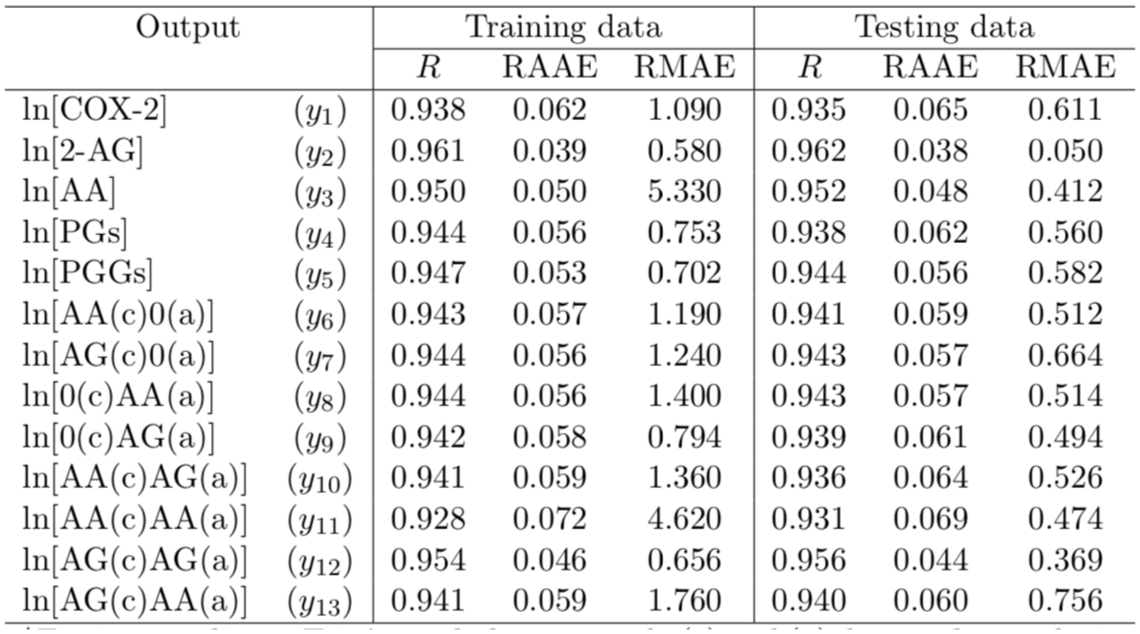
with variables having identical meanings to those in Equation 6.9. R, the correlation coefficient, defined as:

$$R = 1 - \frac{\frac{1}{N}\sum_{s=1}^{N} |\hat{y}_i^{(s)} - y_i^{(s)}|}{\sigma_i} \tag{6.11}$$

again with identical variables to those used in Equation 6.9, was used as the final measure of error stemming from the HDMR approximation of the CORM model. Several training and testing dataset sizes were examined to minimize training time while maximizing predictive power. Based on this analysis a training set of two thousand CORM simulations at parameter values randomly drawn from the marginal parameter distributions fitted during model calibration was chosen. After training a second order HDMR model, performance was tested on a testing dataset containing two hundred points. Error measurements for both the training and testing datasets are shown in Figure 6.1. Truth plots (original CORM simulation values versus values estimated from the HDMR model) are shown in Figure 6.2.

CORM Parameter Name	HDMR Input Variable
kcat_AA2	x1
kcat_AA3	x2
KD_AG_cat3	x3
KD_AG_cat2	x4
KD_AG_allo2	x5
KD_AG_allo1	x6
KD_AA_allo1	x7
KD_AA_allo2	x8
KD_AA_allo3	x9
kcat_AG3	x10
KD_AA_cat3	x11
KD_AA_cat2	x12
kcat_AA1	x13
KD_AG_cat1	x14
KD_AG_allo3	x15
kcat_AG1	x16
kcat_AG2	x17
KD_AA_cat1	x18
AA_0	x19
AG_0	x20

Table 6.1: CORM HDMR Variable Names



Output		Training data			Testing data		
		R	RAAE	RMAE	R	RAAE	RMAE
ln[COX-2]	(y_1)	0.938	0.062	1.090	0.935	0.065	0.611
ln[2-AG]	(y_2)	0.961	0.039	0.580	0.962	0.038	0.050
ln[AA]	(y_3)	0.950	0.050	5.330	0.952	0.048	0.412
ln[PGs]	(y_4)	0.944	0.056	0.753	0.938	0.062	0.560
ln[PGGs]	(y_5)	0.947	0.053	0.702	0.944	0.056	0.582
ln[AA(c)0(a)]	(y_6)	0.943	0.057	1.190	0.941	0.059	0.512
ln[AG(c)0(a)]	(y_7)	0.944	0.056	1.240	0.943	0.057	0.664
ln[0(c)AA(a)]	(y_8)	0.944	0.056	1.400	0.943	0.057	0.514
ln[0(c)AG(a)]	(y_9)	0.942	0.058	0.794	0.939	0.061	0.494
ln[AA(c)AG(a)]	(y_{10})	0.941	0.059	1.360	0.936	0.064	0.526
ln[AA(c)AA(a)]	(y_{11})	0.928	0.072	4.620	0.931	0.069	0.474
ln[AG(c)AG(a)]	(y_{12})	0.954	0.046	0.656	0.956	0.044	0.369
ln[AG(c)AA(a)]	(y_{13})	0.941	0.059	1.760	0.940	0.060	0.756

Figure 6.1: Prediction accuracy and error for a second order HDMR model of CORM simulations. In the outputs column, (c) and (a) denote the catalytic and allosteric subunits of COX-2, and 0 indicates that either site is unbound.

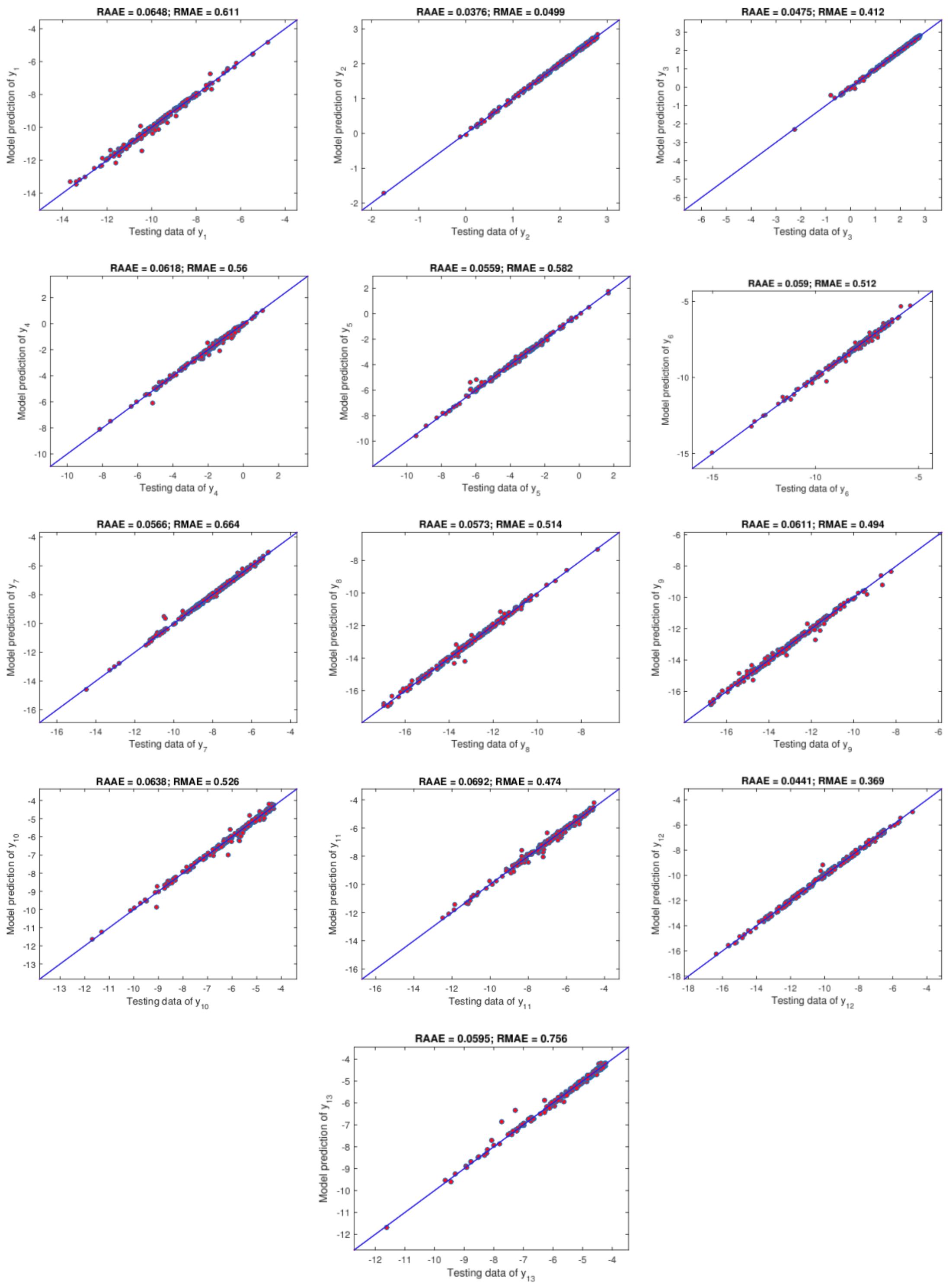


Figure 6.2: Original CORM simulation values versus values calculated from a second order HDMR model trained to the CORM.

6.4 HDMR Based Sensitivity Indices for CORM

The HDMR meta-model of CORM was used to calculate first and second order S_i^a (the structural sensitivity component), S_i^b (the correlation component), and S_i (the total sensitivity of an output to an input) for each input and output combination in CORM. Note that because the input simulations for HDMR training were sampled independently along marginal parameter distributions determined in CORM calibration, S_i^b serves here as a measure of error stemming from random correlations present in the non-infinite dataset rather than a measure of sensitivity resulting from correlations between parameters. Calculated first order S_i^a indices are shown in Figure 6.3, first order S_i^b indices in Figure 6.4, and first order S_i in Figure 6.5.

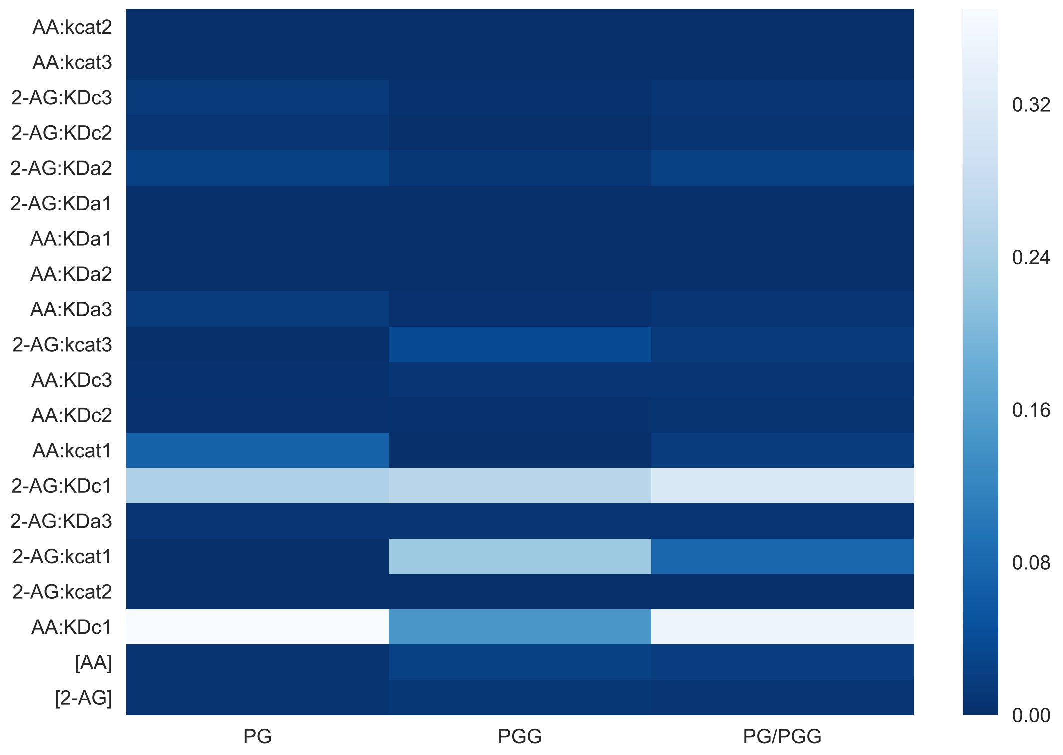


Figure 6.3: Heatmap of first order S_i^a sensitivity indices calculated for all i parameter in CORM. Each row represents a model parameter (input) and each column an output (PG, PGG, or the ratio of the two). Lighter values indicate stronger sensitivity of the output to variation in an input.

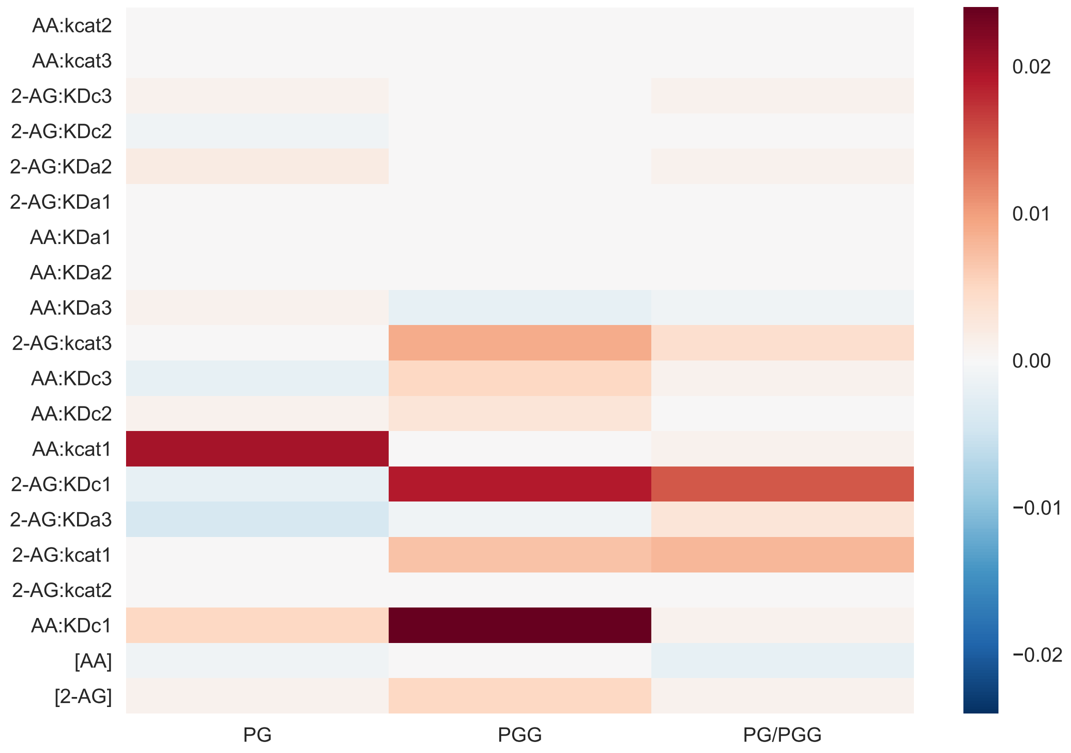


Figure 6.4: Heatmap of first order S_i^b sensitivity indices calculated for all i parameter in CORM. Each row represents a model parameter (input) and each column an output (PG, PGG, or the ratio of the two). Color indicates either positive or negative error in estimates induced by random correlation in the data.

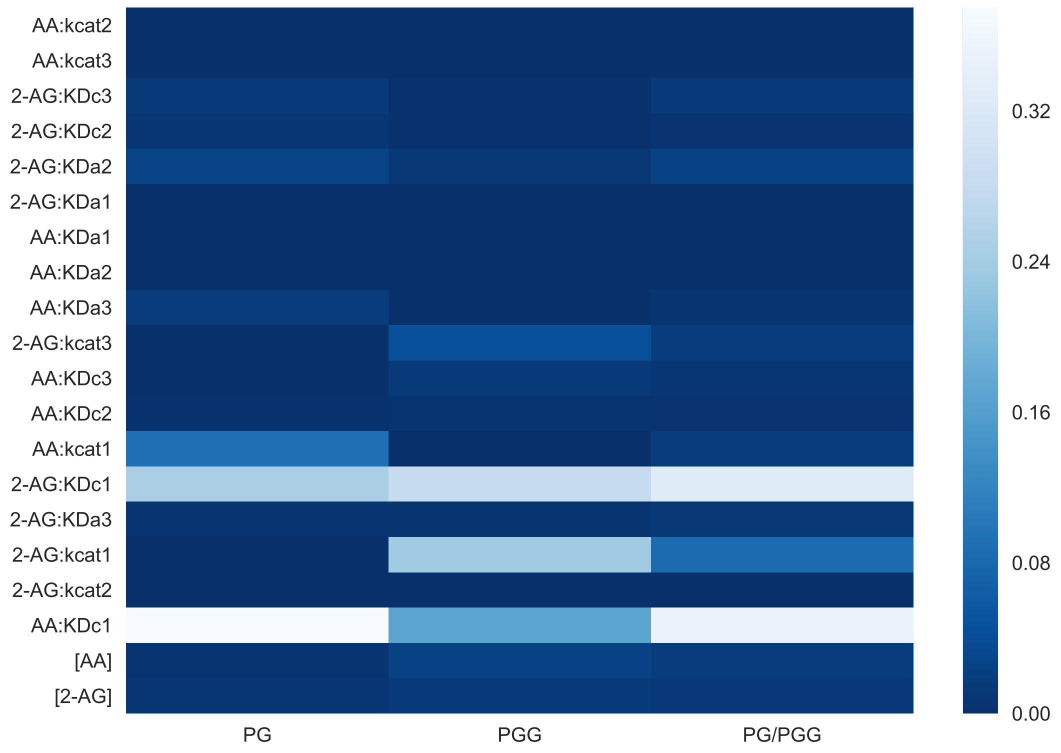


Figure 6.5: Heatmap of first order S_i sensitivity indices calculated for all i parameter in CORM. Each row represents a model parameter (input) and each column an output (PG, PGG, or the ratio of the two). Lighter values indicate stronger sensitivity of the output to variation in an input.

Estimated second order sensitivity indices for PG are shown in Figure 6.6 (S_i^a), Figure 6.7 (S_i^b), and Figure 6.8 (S_i). Second order sensitivity indices for PGG are found in Figure 6.9 (S_i^a), Figure 6.10 (S_i^b), and Figure 6.11 (S_i). Second order sensitivity indices for the ratio of PG to PGG are shown in Figure 6.12 (S_i^a), Figure 6.13 (S_i^b), and Figure 6.14 (S_i).

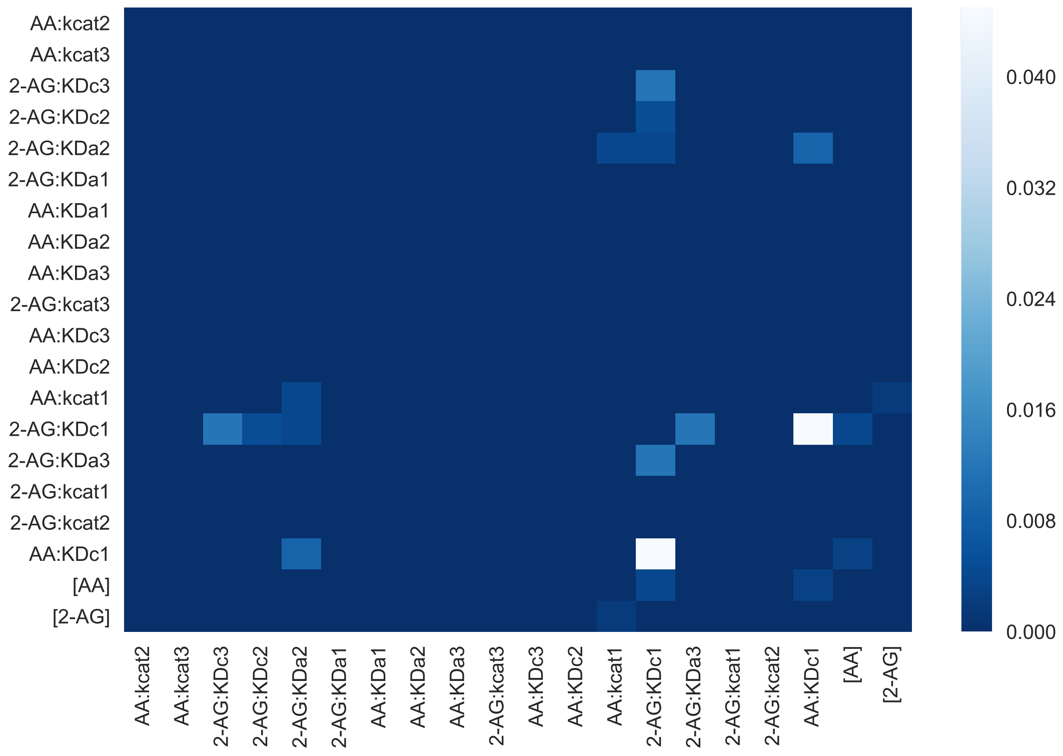


Figure 6.6: Heatmap of S_i^a sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs.

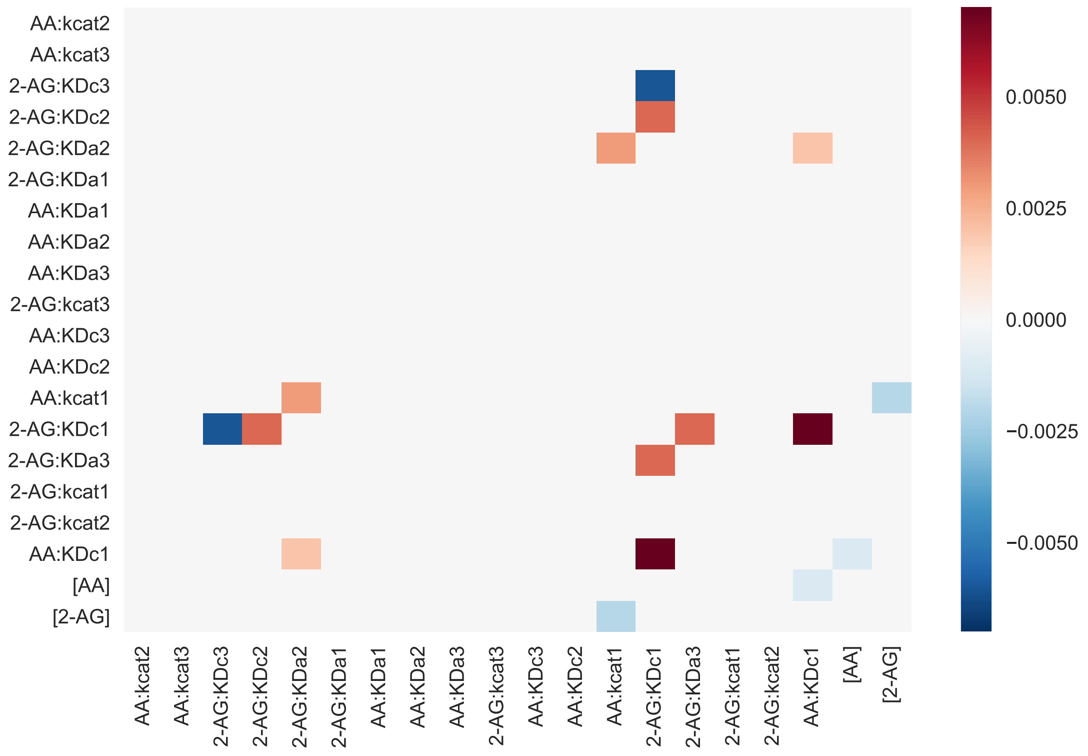


Figure 6.7: Heatmap of S_i^b sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the error in an estimated sensitivity value calculated for PGs to a change in the row and column model parameters. Color indicates either either positive or negative error in estimates induced by random correlation in the data.

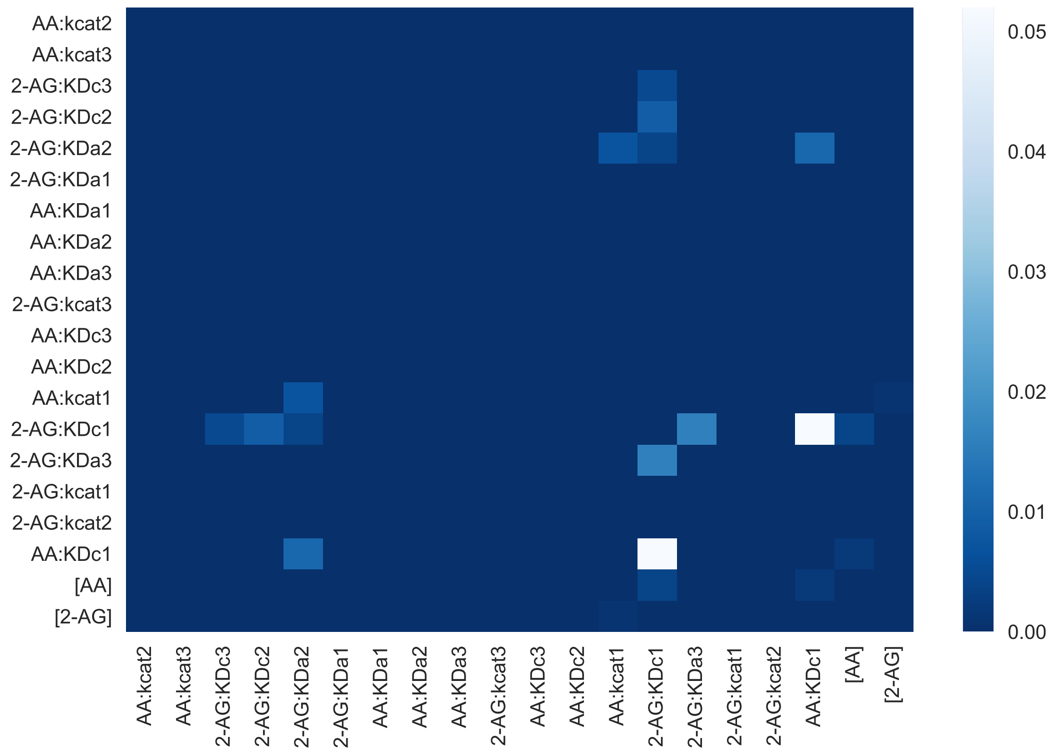


Figure 6.8: Heatmap of S_i sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs.

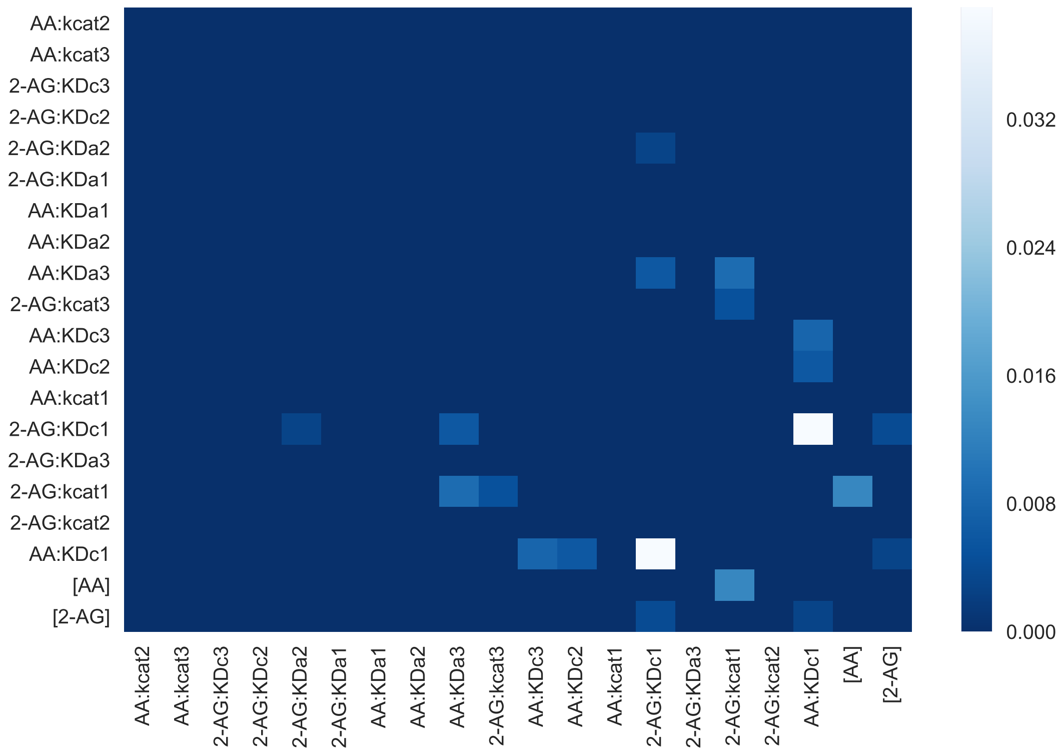


Figure 6.9: Heatmap of S_i^a sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs.

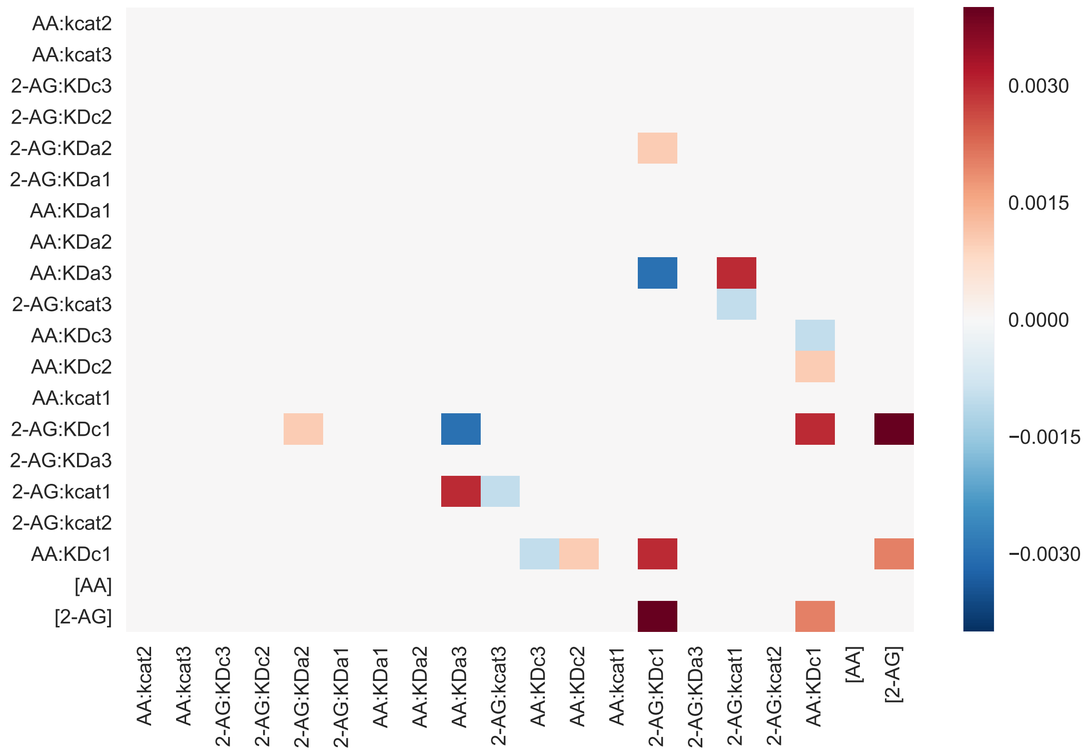


Figure 6.10: Heatmap of S_i^b sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the error in an estimated sensitivity value calculated for PGGs to a change in the row and column model parameters. Color indicates either either positive or negative error in estimates induced by random correlation in the data.

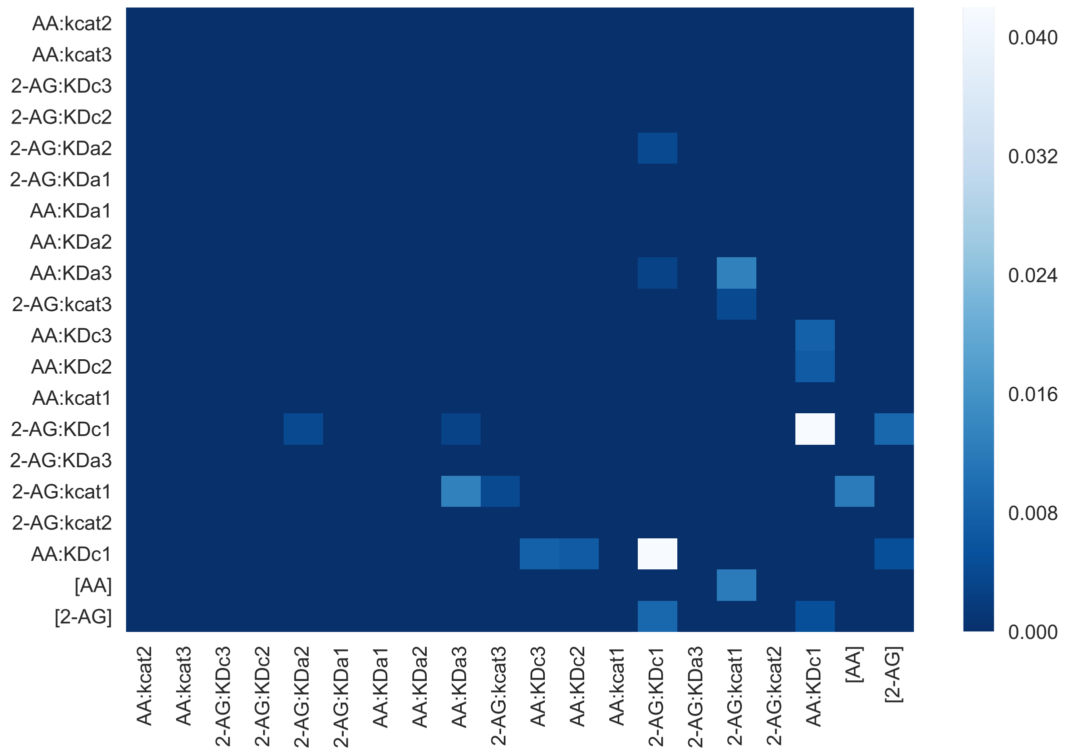


Figure 6.11: Heatmap of S_i sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of PGGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs.

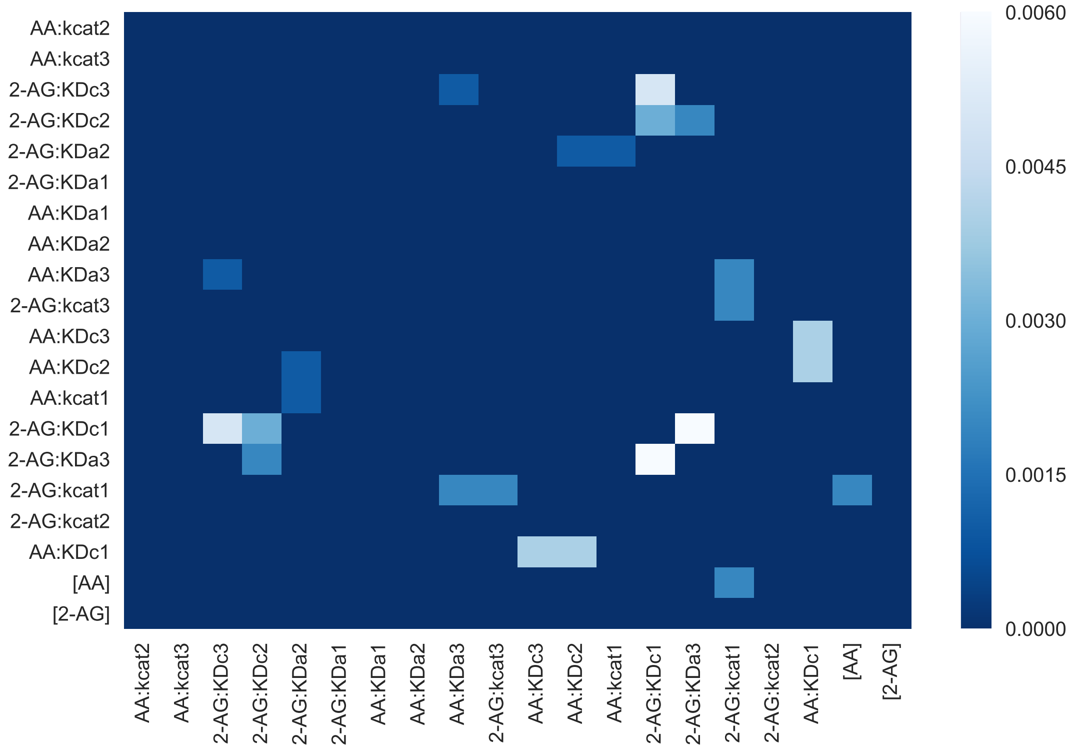


Figure 6.12: Heatmap of S_i^a sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of the PG to PGG ratio to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs.

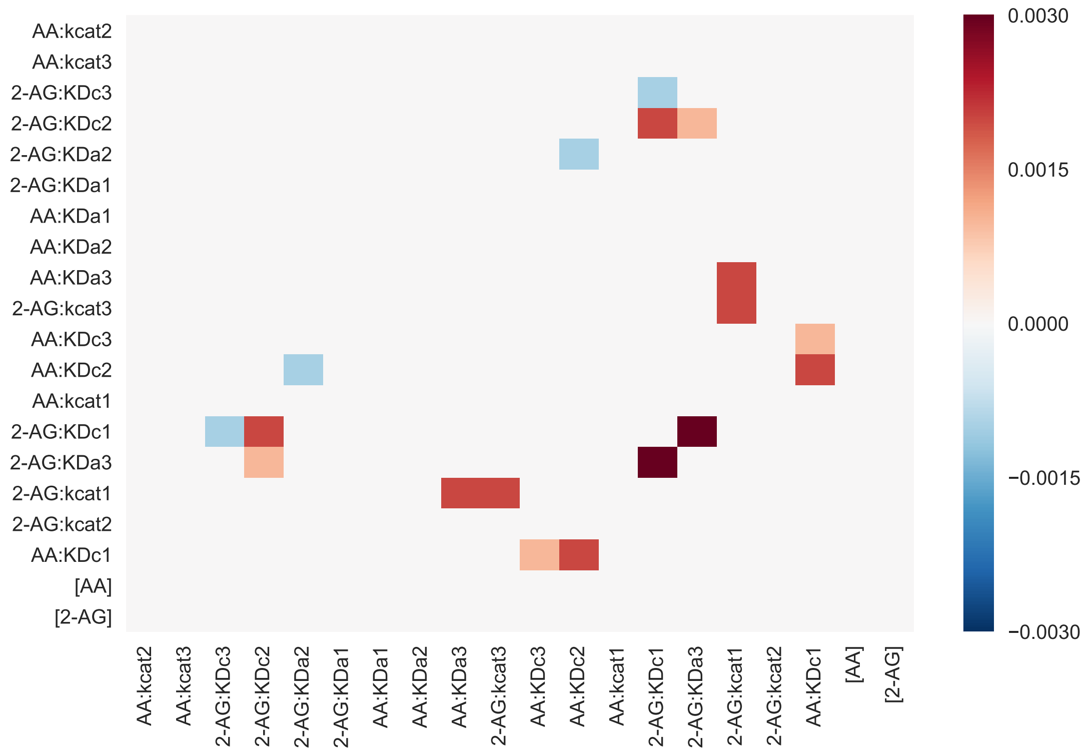


Figure 6.13: Heatmap of S_i^b sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the error in an estimated sensitivity value calculated for the ratio of PGs to PGGs to a change in the row and column model parameters. Color indicates either either positive or negative error in estimates induced by random correlation in the data.

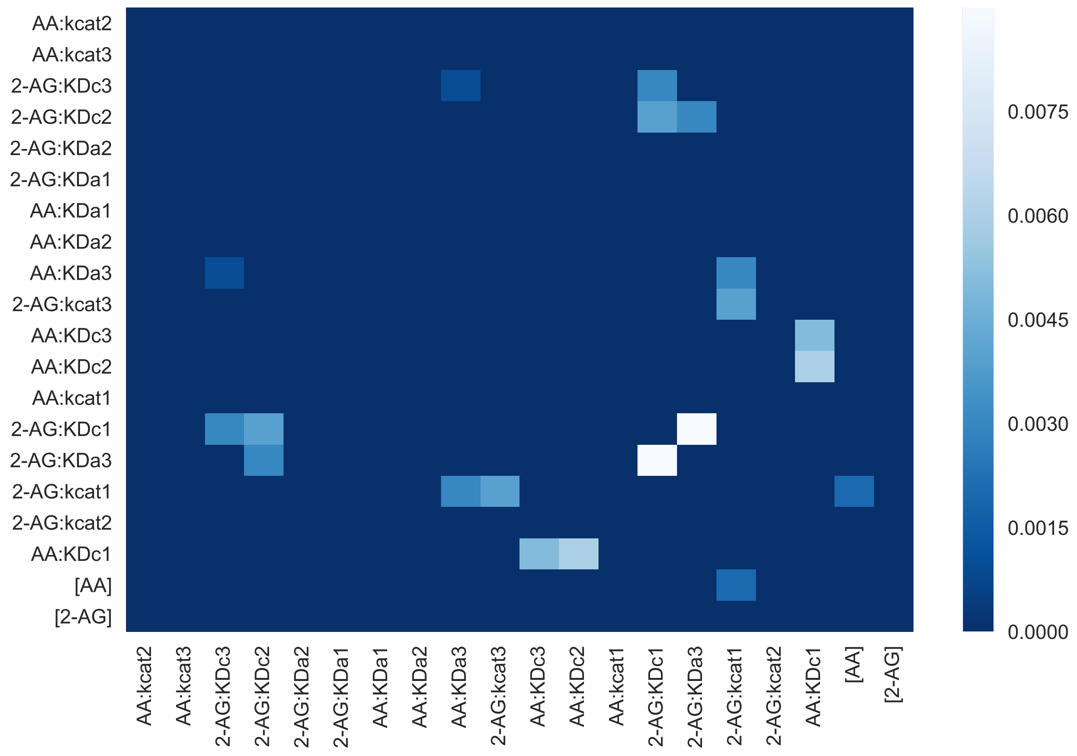


Figure 6.14: Heatmap of S_i sensitivity indices calculated for all i, j parameters in CORM. Each cell represents the sensitivity of the ratio of PGs to PGGs to a change in the row and column model parameters. Lighter values indicate stronger sensitivity of the output to variation in the combination of inputs.

For PG, the sum of all first and second order sensitivity indices is about .99, indicating that 99% of the variance in PG values can be captured by varying the parameters and substrate levels individually or in pairs. The sum of first order indices is .807, with the remaining variance coming from second order interactions, indicating that while 80% of the variance can be captured by varying single parameters, capturing final fifth requires simultaneous variation of parameters (though not higher than second order interactions). As can be seen in Figure 6.5 column 1 and Figure 6.8, PG values are most sensitive to the equilibrium constants for binding of each substrate to the catalytic site (the constant for AA captures 38% of the variance while the 2-AG constant captures 25%), as well as to the catalytic rate for AA turnover when nothing is bound in the allosteric site (10% of the variance). The single greatest contributor to PG variation is the equilibrium constant for

AA binding in the catalytic site, when nothing is bound in the allosteric site. It accounts for more than a third of the total variance, which suggests that affecting AA binding in the catalytic site would be the most effective means to perturb PG values (and would be superior to simply affecting the catalytic turnover rate for AA).

For PGG, the sum of all first and second order sensitivity indices is again about .99. As with PG, the bulk of PGG variation (about 82%) can be captured by changing only single parameters, with the remainder attributable to interactions between two parameters. As shown in Figure 6.5 column 2 and Figure 6.11, PGG depends roughly equally on the equilibrium constant for 2-AG binding in the catalytic site (28% of the variance) and the catalytic turnover rate of 2-AG when nothing is bound in the allosteric site (24% of the variance). There is a slightly smaller dependence on the AA catalytic site equilbrium binding constant (17%). To change PGG levels, changing the affinity of the catalytic site 2-AG and changing the catalytic turnover rate of 2-AG when the enzyme is allosterically unbound are expected to have roughly the same effect, though neither will have as large an effect as changing AA affinity for the catalytic site would have on PG levels.

Finally, the sum of all first and second order sensitivity indices is again nearly 1 when considering sensitivity of the ratio of PG to PGG. First order indices capture a greater proportion of the variance than for either individual product (91%), with the remainder depending on second order interactions between parameters. The ratio of products depends strongly and roughly equally on the equilibrium constants for binding of the substrate in the catalytic site (35% for AA and 33 % for 2-AG), with a much smaller contribution (8%) from the 2-AG turnover rate. The contribution from the AA turnover rate is even smaller (2%).

Note, however, that HDMR returns the most effective perturbations, but these pertrubations may be in either direction. As shown in Figure 6.15, an increase in the ratio (more PGs, fewer PGGs) could be accomplished by increasing the equilibrium constant for 2-AG catalytic site binding (making binding less tight), while a decrease in the ratio (fewer PGs, more PGGs) could be accomplished by increasing either the catalytic turnover rate for 2-AG with nothing in the allosteric site, or by increasing the equilibrium constant for AA binding in the catalytic site. In the final section we test whether the key perturbations for each output vary depending on the location in substrate space (as levels of AA and 2-AG are likely to vary *in vivo*).

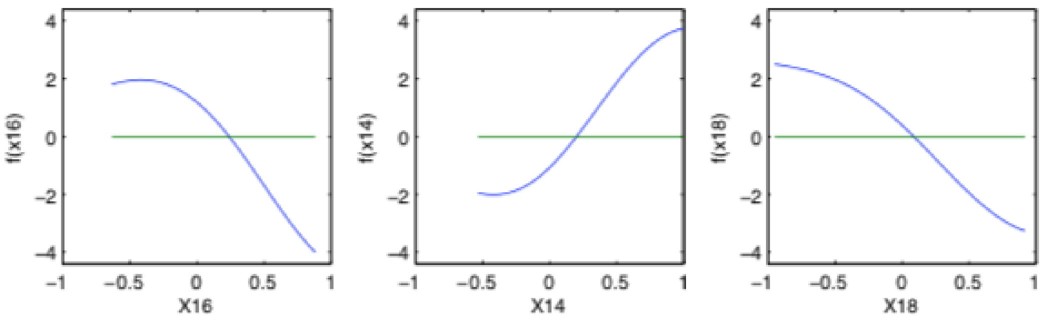
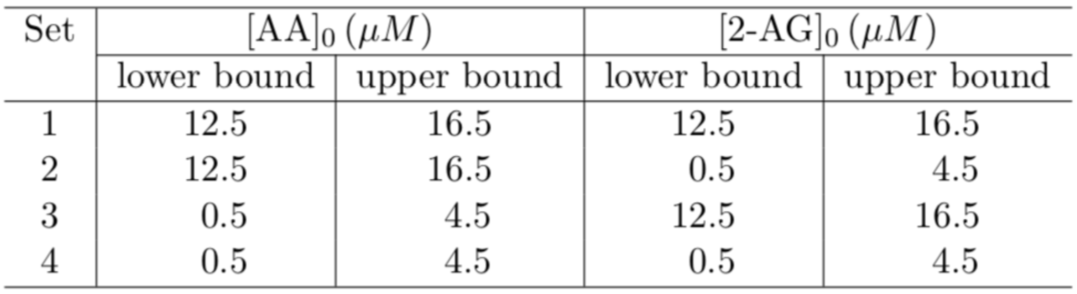


Figure 6.15: HDMR functions for relationships between the most sensitive parameters in CORM and the ratio of PGs to PGGs. From left to right, the parameters are kcat_AG1, KD_AG_1, and KD_AA_1.

6.5 CORM Sensitivity to Perturbations at Different Substrate Levels

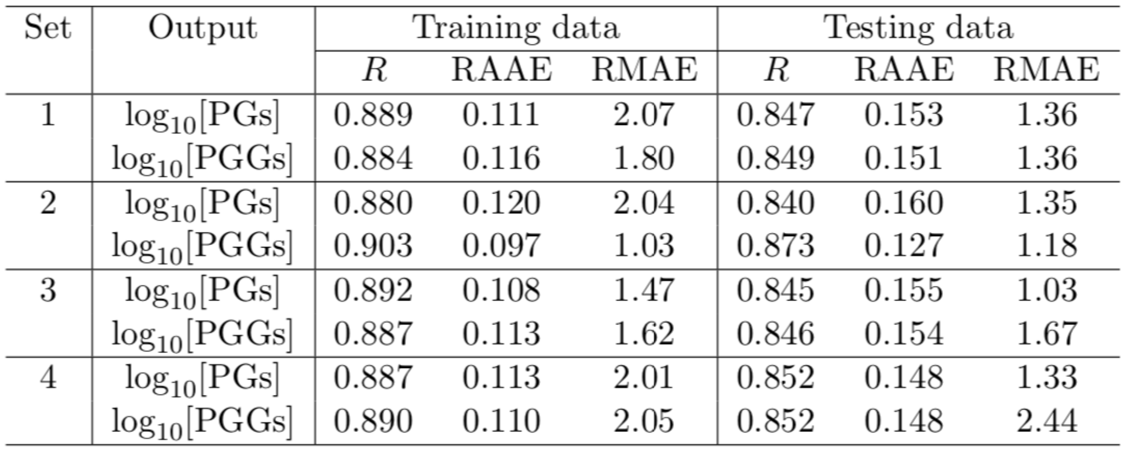
In order to test whether the levels of AA and 2-AG affect sensitivity to perturbations in the CORM network, a second analysis was performed in which four different datasets located in different regions of substrate space (as shown in Figure 6.16) were generated for both the PG and PGG outputs. Each dataset was used to fit an HDMR model, and a sensitivity analysis was performed for each.



Set	$[\mathrm{AA}]_0\,(\mu M)$		$[\text{2-AG}]_0\,(\mu M)$	
	lower bound	upper bound	lower bound	upper bound
1	12.5	16.5	12.5	16.5
2	12.5	16.5	0.5	4.5
3	0.5	4.5	12.5	16.5
4	0.5	4.5	0.5	4.5

Figure 6.16: Levels of AA and 2-AG Used to Generate Data for Sensitivity Analysis

For construction of each HDMR model, 2000 training points and 1000 testing points were used. HDMR error measures compared to the original data are shown in Figure 6.17.



Set	Output	Training data			Testing data		
		R	RAAE	RMAE	R	RAAE	RMAE
1	$\log_{10}$[PGs]	0.889	0.111	2.07	0.847	0.153	1.36
	$\log_{10}$[PGGs]	0.884	0.116	1.80	0.849	0.151	1.36
2	$\log_{10}$[PGs]	0.880	0.120	2.04	0.840	0.160	1.35
	$\log_{10}$[PGGs]	0.903	0.097	1.03	0.873	0.127	1.18
3	$\log_{10}$[PGs]	0.892	0.108	1.47	0.845	0.155	1.03
	$\log_{10}$[PGGs]	0.887	0.113	1.62	0.846	0.154	1.67
4	$\log_{10}$[PGs]	0.887	0.113	2.01	0.852	0.148	1.33
	$\log_{10}$[PGGs]	0.890	0.110	2.05	0.852	0.148	2.44

Figure 6.17: HDMR Model Accuracy for Each Region of Substrate Space

First order sensitivity indices for each region of substrate space for PG are shown in Figure 6.18 and for PGG in Figure 6.19. First order indices capture 85% of the variance, similar to in the previous analyses.

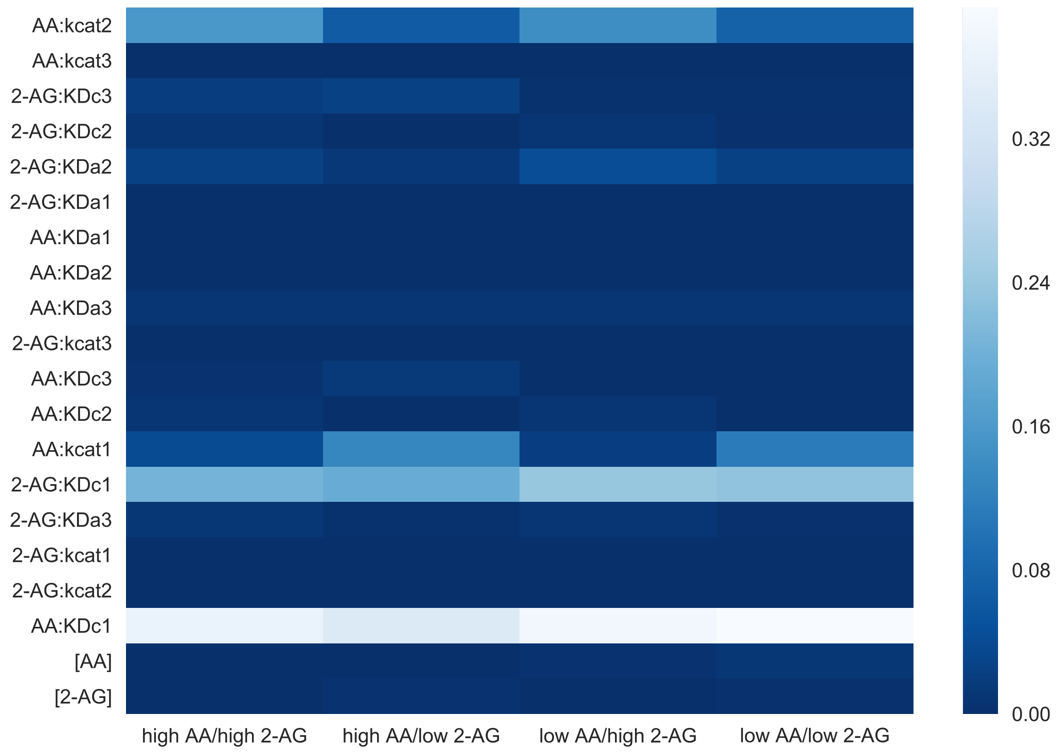


Figure 6.18: First order sensitivity indices for PG in each region of substrate space. Each row represents sensitivity to perturbations in a particular model parameter, and each column is a different region of substrate space. Lighter colors indicate higher sensitivity.

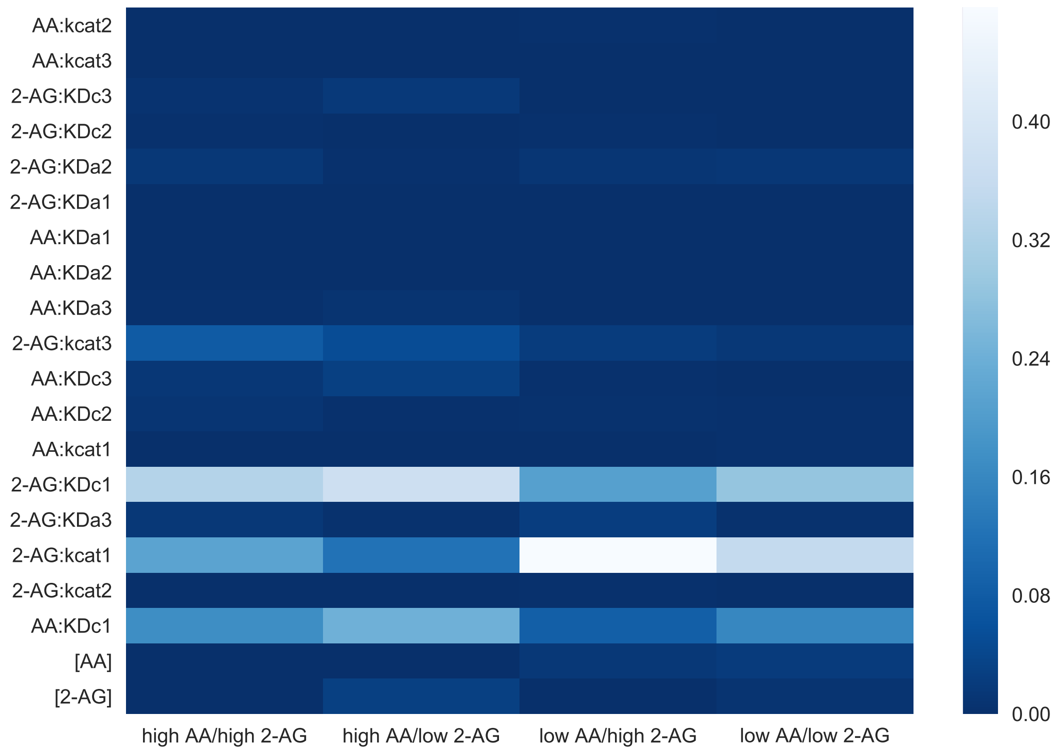


Figure 6.19: First order sensitivity indices for PGG in each region of substrate space. Each row represents sensitivity to perturbations in a particular model parameter, and each column is a different region of substrate space. Lighter colors indicate higher sensitivity.

These calculated sensitivities do appear to indicate that an output's sensitivity to perturbation in a given model parameter does depend on the levels of system substrates. Furthermore, the changes in sensitivity can be rationalized by considering what paths an output is predominately being produced (see Figs. 4.3, 4.4, and 4.5). For example, PG is more sensitive to perturbations in kcat_AA1, the catalytic rate for the turnover of AA in the absence of any allosteric regulator, when levels of 2-AG are low, which corresponds to conditions under which relatively more PG is produced by the unmodulated pathway. PG sensitivity to kcat_AA2 (the catalytic turnover rate for AA when 2-AG is bound in the allosteric site), increases under the opposite conditions, when 2-AG levels are high, corresponding to conditions when relatively more of the product is produced by the 2-AG-modulated pathway. PGG sensitivities can be similarly rationalized; for example, sensitivity to kcat_AG1, the turnover of 2-AG with no allosteric modulator present, is highest when AA is low and 2-AG is high, conditions corresponding to when production is predominately via that complex.

There are several caveats to this analysis. First, the binning of substrate values was somewhat arbitrary, and it would be useful to see what bin size maximizes the sensitivity information obtained while not requiring excessive samples. Secondly, for this analysis the parameters were perturbed within the ranges dictated by their calibrated distributions, meaning some parameters could vary considerably while others very little. This was done under the rationale that this was the regime under which the model captured the experimental data; however, since the goal of a perturbation is to *disrupt* the system behavior, a future analysis in which parameters were perturbed in a range determined by what could theoretically be achieved *in vivo*. Despite these caveats, this analysis indicates that the calculation of sensitivity indices in this relatively simple system for multiple input/output combinations is computationally feasible, and should inform similar analyses for larger models.

Chapter 7

High Performance Computing with Amazon Web Services: A Case Study

7.1 Introduction

Many of the sophisticated analyses presented in the preceding chapters are computationally intensive; for example, the information theoretical estimation of channel capacities required more than a million CPU hours. Such analyses are increasingly common in the era of *big data*, originally defined by the ”3Vs” suggested by the Gartner group (*volume*, *variety*, and *velocity*) [120], with the later addition of a fourth V, *veracity*, by IBM [121]. IBM has also suggested a fifth emerging quality of big data, *value* [121]. Experts have further noted that each quality should be defined relative to the needs and abilities of the user [122]; for example, for a user with proficiency only in data analysis via spreadsheets, data becomes big data at the point when the user can no longer effectively analyze it, regardless of whether the computer would be similarly stymied.

Much biomedical research now generates data with the defining features of big data. The *volume* of data available to biomedical researchers is growing exponentially and has been for more than a decade [123, 124, 125, 126, 127]. This data features a *variety* of data types and structures; some datasets, like ’omics’ data [128], are collected extremely systematically and consequently are highly structured, while other datasets (such as clinical notes in electronic health records) [129], are valuable but require an imposition of structure before analysis can begin. Huge volumes of data are generated at high *velocity* by methods such as next-generation sequencing [124, 125], requiring high-throughput computational methods to process. The *veracity* of this data may sometimes be in question, in part due to the current ”reproducibility crisis” in science [130]. Finally, the *value* of this data is becoming increasingly evident as related for-profit fields like healthcare leverage big data to improve quality and efficiency [126].

Two characteristics of big data, volume and velocity, have encouraged the adoption of parallel computing methods of analysis, in which large compute clusters or supercomputers are used to quickly analyze reams of data [131]. The related field of cloud computing, in which these resources are accessed remotely [131], has similarly grown in the wake of demand for big data processing capabilities. In addition to a host of academic and government-owned clusters, major companies such as Google (Google Cloud Platform) and Amazon (Amazon Web Services (AWS)) offer vast computing power to the general public. These latter commercially available services have led to the release of many cloud-based analysis packages developed in research labs [132, 133, 134, 135, 136, 137, 138, 139, 140, 141]. Such analysis pipelines are extremely useful when a user has similar data to analyze; however, general guidelines for utilizing cloud computing resources for arbitrary big data are lacking.

7.1.1 Overview of the protocol

In the following we introduce a flexible workflow for analysis of big data using AWS's Amazon Batch service, which generates an on-demand cluster in the cloud that is optimized to the computational needs of the user. While the specific application we utilized this approach for is detailed, the overall method is flexible enough to be applicable to any problem that would benefit from some degree of parallelization. although cost-effectiveness may vary (as discussed in detail in the protocol). The protocol assumes that the code needed for a specific analysis has already been generated and tested and now requires translation to the AWS Batch environment.

Preparing, running, and retrieving data for an arbitrary AWS Batch job consists of three major stages:

1. Prepare Docker image with any required software and small files and transfer large data files and code to AWS S3.

2. Setup AWS Batch variables: required access role(s), job definition, job queue, and compute environment.

3. Submit jobs to queue, monitor, and retrieve final output.

To facilitate the quick development of a AWS Batch workflow, template scripts have been generated for several steps and are provided in Appendix II.

7.2 Materials

- An AWS account (created at aws.amazon.com).
- Analysis code to be paralyzed (in programming language(s) of choice)
- A standard personal computer

7.3 Protocol

Stage 1: Prepare Docker image with required software, store the image on AWS, and transfer large files for analysis and analysis code to the cloud. Timing: variable depending on size of files to be transferred to cloud; less than an hour active interaction required.

1. Install the AWS Command Line Interface (CLI), a set of tools for interfacing with AWS from the command line. A platform-specific installation guide is available in the AWS CLI documentation at docs.aws.amazon.com/cli.

2. Download and install Docker [142], software that allows you to place any programs and small files you need for your computation in an enclosed container (called a Docker image) which can then be deployed independently on any computational node. This simplifies development of a consistent execution script across a possibly heterogeneous compute cloud. Docker is free and may be downloaded from docker.com.

3. Download and unzip AWS templates for creating a new Docker image from https://github.com/awslabs/aws-batch-helpers/archive/master.zip. Inside the unzipped folder (fetch-and-run) are two files:

   - Dockerfile
   - fetch_and_run.sh

   These templates allow the creation of a Docker image based on the latest Amazon Linux image that will download and run an arbitrary script from the cloud.

4. Modify the Dockerfile for the demands of your job. This includes:

   (a) If desired, change the base image (changing the base operating system of the Docker image from the default Amazon Linux by modifying the `FROM` line in the Dockerfile:

   ```
   FROM amazonlinux:latest
   ```

   Note that unless you have a compelling reason to use a different base image, the Amazon Linux default is likely to work well.

   (b) Install any necessary software. The template Dockerfile includes the line:

   ```
   RUN yum -y install unzip aws-cli
   ```

   which installs AWS command line tools, allowing the created Docker image to access other AWS tools, including S3. You can add additional `RUN` lines with shell commands to install other software, change directories, change permissions, or anything else needed. Note that the specific commands will be dependent on the base image selected in part (a); commands for the default Amazon Linux base image are shown. An example of a modified Dockerfile used in our own research is provided in Appendix II. It includes installation of Java and Python as well as several Python packages and commandline utilities.

(c) Add any small files that will be used in all analyses to the Docker image. The template Dockerfile includes an example of this:

```
ADD fetch_and_run.sh /usr/local/bin/fetch_and_run.sh
```

which adds the fetch_and_run.sh script to the Docker image. The first argument to the command is the path to the file on the current filesystem, and the second is the desired path on the created Docker image. The example Dockerfile included in Appendix II includes addition of several small text files to the Docker image.

(d) If desired, change the working directory (the directory the image will start in). By default this directory is `/tmp`:

```
WORKDIR /tmp
```

(e) If desired, change the user to execute commands in the image from the default:

```
USER nobody
```

5. Build the new Docker image. To do so, open a terminal, navigate to the directory containing your modified Dockerfile, and enter the following command:

```
docker build -t awsbatch/fetch_and_run .
```

If desired, the image name can be changed from awsbatch/fetch_and_run to a desired name. After the image builds successfully, you can issue the following command:

```
docker images
```

to list your current images. The new image (`awsbatch/fetch_and_run`, or whatever name selected) should be listed.

6. To use the Docker image in AWS Batch, it will need to be added (pushed) to the AWS Elastic Container Registry (ECR). Creating a repository allows version control for a given container image. To do so:

(a) Select AWS Elastic Container Service (ECS) from the Services menu in AWS.

(b) Under Amazon ECR on the left, select Repositories.

(c) Click Get Started or Create Repository.

(d) Enter a name for the new repository, copy the repository URI, and click Next step.

(e) Note the current AWS region located in the upper right of the screen, next to the current AWS username.

(f) Login to ECR from the command line using the following command (depending on operating system):

   i. `$(aws ecr get-login --region us-east-1)` (on macOS or Linux)

   ii. `Invoke-Expression -Command (Get-ECRLoginCommand -Region us-east-1).Command` (on Windows)

   If the AWS region you noted earlier is not US East (N. Virginia), replace us-east-1 with the correct AWS region code.

(g) Tag the Docker image you created in step 5 so that it can be pushed to the repository you just created:

```
docker tag awsbatch/fetch_and_run:latest REPOURI:latest
```

   replacing `REPOURI` with the URI of the repository copied in step 6d. If you altered the name of the Docker image from `awsbatch/fetch_and_run` when building the image in step 5, use your chosen name in place of `awsbatch/fetch_and_run`.

(h) Finally, push the tagged Docker image to the created repository:

```
docker push REPOURI:latest
```

   again replacing `REPOURI` with the repository URI from step 6d.

7. Any files required by the parallel jobs and not included in the created Docker image

will need to be uploaded to AWS S3 and then retrieved by individual jobs (instructions for the latter are detailed in step X). To do so:

(a) Select AWS S3 from the Services menu in AWS.

(b) Click Create bucket (a storage area with a given set of permissions and options).

(c) Enter a name for the bucket, select the same region used for the ECR repository in step 6e and click Next.

(d) Advanced options are configurable on the next screen but are unnecessary for a basic AWS Batch run and can be configured later if required. Click Next.

(e) The default selected options will give Read and Write access to the new bucket to your AWS username, as well as the ability to change who can access Read and Write privileges for the bucket. If you are creating a bucket to be accessed by a group of users rather than just by you, you can search for and add permissions for other AWS users on this screen. Once you've finished adding users, click Next.

(f) Review the summary of the bucket information for correctness and click Create bucket.

(g) In the console, click on the newly created bucket.

(h) Upload any files required by your jobs and not included in the created Docker image. This includes scripts as well as any large datasets to be accessed by individual jobs. If desired, you can organize code and input data files into folders in the S3 console. In addition to any input data files, one file that should be added to S3 if using the `fetch_and_run.sh` setup provided by AWS and discussed in step 3 is a shell script that will be run on initializing the Docker container. A basic template for this script is provided in Appendix II. The following Stage details the process for setting up the AWS Batch environment.

Stage 2: Setup AWS Batch variables, including required access role(s), job definition, a compute environment, and a job queue. Timing: about an hour.

1. Setting up an analysis with AWS Batch involves several steps:

   (a) If you will need to access any files on S3 during a job (as when using the `fetch_and_run.sh` setup), an AWS Identity and Access Management (IAM) role with permissions to access S3 will need to be created.

   (b) A template for a set of jobs, called a job definition, must be created.

   (c) A compute environment optimized for the jobs must be created.

   (d) A job queue must be created. Jobs will be submitted to the queue and then initialized on a particular compute environment.

   These steps are further detailed below.

2. To set up an AWS IAM role with access to S3 (if required):

   (a) Select AWS IAM from the Services menu in AWS.

   (b) Select Roles from the menu on the left.

   (c) Click Create role.

   (d) Choose Batch from the list of AWS services.

   (e) Click Next: Permissions.

   (f) AWS provides a default AWS Batch role, AWSBatchServiceRole, which provides access to the default services needed by AWS (but not S3). The permissions of this role have already been selected. Access to S3 will be added after role creation. Click Next:Review.

   (g) Enter a name for your new role and click Create role.

   (h) In the list of roles, select the role you just created.

   (i) If not already active, select the Permissions tab. Click Attach policies.

(j) Select AmazonS3FullAccess from the list of permissions. Note that if your job required access to any of AWS's other services, permissions for that access could be added here as well. Click Attach policy.

3. To create a job definition:

   (a) Select AWS Batch from the AWS services menu. On the left, click Job definitions.

   (b) Click Create.

   (c) Enter a name for the job definition.

   (d) If desired, change the maximum number of attempts for a given job (if failure occurs), which is 1 by default.

   (e) If you created a access role with new permissions in step 2, select it next to Job role.

   (f) Next to Container image, enter the ECR URI from stage 1, step 6d.

   (g) The Docker container image created previously will expect to be passed a shell script name, and any arguments to the script, in the Command value, just as you would call the same script from the command line. For example, to run a script called `runjob.sh`, with arguments `cat` and `3`, enter

   ```
   runjob.sh cat 1
   ```

   in the Space delimited tab next to Command. Below the text box, the JSON command that will be actually passed to the Docker container when the job runs is shown. For the example above, this would be:

   ```
   ["runjob.sh", "cat", "1"]
   ```

   If you have only a single job to run, entering a command using the Space delimited tab is the simplest method. However, if you have a large number of jobs to run that are identical except for the data to be analyzed, the simplest method is to create a job array using the JSON tab next to Command. A job array is a set of jobs, each of which launches with the `AWS_BATCH_JOB_ARRAY_INDEX` environment variable set to a different integer. The number of jobs in the array is set using a value called `ARRAYSIZE`. To create a job array:

   i. Instead of selecting the Space delimited tab next to Command, select the JSON tab.

   ii. In the JSON tab, enter the following (with exact capitalization and punctuation, including all brackets, commas, and quotes):
   `{"containerOverrides":{"command":COMMAND},`
   `"arrayProperties":{"size":ARRAYSIZE}}`
   where `COMMAND` is the command to be passed to the Docker image in JSON format (*e.g.* `["runjob.sh", "cat", "1"]` in the example above), and `ARRAYSIZE` is the number of datasets you need to analyze using this job definition. For example, a job definition with `ARRAYSIZE` of 100 will launch 100 jobs, each with the `AWS_BATCH_JOB_ARRAY_INDEX` set between 0 and 99.

(h) Next to vCPUs, enter the number of virtual CPUs required by each individual job in the array. Note that if your job requests more resources than listed in the job definition during execution, it will fail.

(i) Next to Memory (MiB), enter the amount of memory required by an individual job in the array in mebibytes. Note that if your job requests more resources (either vCPUs or memory) than listed in the job definition during execution, it will fail.

(j) If desired, change the privileges and username of the user who will execute commands in the Docker container. For most jobs the defaults should be fine.

(k) If desired, you can expose particular Mount points and Volumes to the container when each job runs. This is an alternative to using S3 for file storage for jobs that is not further detailed in this guide.

(l) If using the default `fetch_and_run.sh` job format provided by AWS and described when creating the Docker image in Stage 1, step 3, and a job script based on the template provided in Appendix II and added to S3 in Stage 1, step 7h, up to four environment variables will need to be set. To enter these, select Add environment variable and enter the variable name under Key and its value under Value. The variables that may be required are:

   i. `BATCH_FILE_S3_URL` : the S3 URL for the job script to be executed (based on the template in Appendix II). This can be found by selecting the uploaded file in the S3 console and clicking Copy path.

   ii. `BATCH_FILE_TYPE` : if you uploaded a shell script to S3 to be run by the job, enter `script`. Alternatively, you can upload a zip file containing a script to S3 and enter `zip` here.

   iii. `INPUTDIR` : the S3 URL for the directory containing your input files.

   iv. `OUTPUTDIR` : the S3 URL for the directory to which you wish to upload your output files.

(m) Click Create Job Definition.

4. To create a compute environment in AWS Batch:

   (a) Open the AWS Batch console.

   (b) On the left, click Compute environments.

   (c) Click Create environment.

   (d) The default selection of a Managed compute environment (in which AWS controls the scaling and configuration of the nodes in the environment) is likely to be ideal for all but advanced users.

   (e) Enter a name for the compute environment.
   (f) Leave Create new role selected to create a new Service role and a new Instance role for use by the environment.
   (g) Next to Provisioning model, select either
      i. On-Demand : to pay for instances on demand, at full price *or*
      ii. Spot : to pay for ”Spot” instances at a discounted price; however your jobs can be interrupted.
   (h) Leave default values for the other entries. If desired, you may wish to alter the Maximum vCPUs (the maximum number of virtual CPUs in your compute environment at one time). Note that new AWS accounts begin with a limit on the number of an instance type that can be launched simultaneously. These limits can be increased by entering a request with AWS Support.
   (i) Click Create.

5. To create a job queue to which to submit jobs:
   (a) Open the AWS Batch console.
   (b) On the left, select Job queues.
   (c) Enter a name for the job queue.
   (d) If desired, enter a number for the priority of the queue. Higher numbered queues are given higher priority. Note this is only useful if you intend to have multiple job queues.
   (e) Ensure the box for Enable Job queue is checked.
   (f) Select the new compute environment you just created.
   (g) Click Create Job Queue.

6. After completion of steps 1-5, all infrastructure is in place for running AWS Batch jobs. In Stage 3, the process for submitting and monitoring output for jobs will be detailed.

Stage 3: Submit a job and monitor and retrieve job output. Timing: Variable depending on job length; less than an hour active interaction required.

1. To submit a job to AWS Batch:
   (a) Open the AWS Batch Dashboard.
   (b) Click Create job.
   (c) Enter a name for the job.
   (d) Select the job definition you created in Stage 2, step 3.
   (e) Select the job queue you created in Stage 2, step 5.
   (f) Optionally, you may alter any of the variables defined in the job definition by altering on the Create job screen; if you do not enter text for an entry on this screen, the value from the job definition will be used by default. Note that you do not need to reenter information from the job definition on this screen if it is the same as that already entered when creating the job definition.
   (g) Press Submit job.
2. To monitor a submitted job:
   (a) Open the AWS Batch Dashboard.
   (b) The number of jobs in a given status will be shown next to the queue the jobs were submitted to on this dashboard. As the job progresses, it will move from a status of `SUBMITTED` to `PENDING`, to `RUNNABLE`, to `STARTING`, to `RUNNING`, and finally to either `FAILED` or `SUCCEEDED` depending on whether the process exits with an error code or not.

(c) To see details about jobs in a particular queue and status category:

   i. Click the job queue/job status combination in the Dashboard table. This will open a list of jobs with that status in that queue on the Jobs screen of AWS Batch. Alternatively, you could find these jobs by selecting the correct queue and status directly from the Jobs screen instead.

   ii. In the list of jobs on the Jobs screen, click on a Job ID.

   iii. If the job you selected is an Array job, you'll be taken to another list with the individual array jobs. Click any one on the list.

   iv. Details for the job are shown, including its current status, start and end times, and job ID. To see the command line output for the compute instance running the job, click View logs under Attempts. If the job was attempted multiple times, there will be multiple logs.

   v. Clicking the View logs link will open AWS CloudWatch, where the logs are stored. The output of the given job will be shown. If the job is currently running, you may click the refresh icon in the upper right corner to retrieve the most recent output. Note that these logs are the most useful resource for troubleshooting jobs that fail or yield unexpected output.

3. To retrieve job output:

   (a) The process for retrieving output logs for a given job is detailed in step 2c.

   (b) Any output files uploaded to S3 will be available for download on S3 after the job successfully runs.

7.4 Troubleshooting

Troubleshooting advice can be found in Table 7.1.

Step	Problem	Possible reason	Solution
Stage 3, Step 2	A job appears with a `FAILED` status.	The job exited with a status code other than 0. This could indicate an error in the code itself, or that the job requested more CPUs or memory than it was allotted in the job definition.	Check the output logs for the job by following the steps in Stage 3, step 2c. If the container has returned an `OutofMemoryError`, increase the memory requested in the job definition and resubmit. If the code has thrown an error, correct the code, reupload it to S3, and resubmit the job.
Stage 3, Step 2	A job remains in the `RUNNABLE` state indefinitely.	The compute environment is incorrectly configured.	Create a new compute environment and attach it to the job queue for that job.
Stage 3, Step 2	AWS Batch launches fewer simultaneous instances than are specified as the maximum number of instances when creating the compute environment.	AWS account limits are decreasing the number of simultaneously running instances of the type required for your job.	Create a new case for Service Limit Increases in the AWS Support Center (accessible in the upper right hand corner) and request a limit increase.

Table 7.1: Troubleshooting guidance for using AWS Batch.

Chapter 8

Discussion and Future Directions

8.1 Discussion

8.1.1 COX-2 as an Allosteric Enzyme

In 1997, Swinney *et al.* [143] reported that COX-1 behaves as an allosteric enzyme with a Hill coefficient of 1.3. Their interpretation of this apparent cooperativity was later disputed by Chen *et al.* [144], who explained the phenomenon on the basis of COXs requirement for product hydroperoxide to activate the enzymes active site. Since that time, researchers have generally agreed that the COX isoforms do not behave as allosteric enzymes with regard to AA. However, recent evidence strongly supports the concept that the enzymes are functional heterodimers and that activity is modulated by binding of non-substrate ligands to the allosteric subunit [37, 39, 145]. Through creation and analysis of CORM, we reported that AA and 2-AG, despite similar catalytic efficiencies with COX-2 when measured individually *in vitro*, differ markedly in their rate of oxygenation in the presence of the other substrate. Furthermore, we present data consistent with the hypothesis that this modulation of COX-2 activity not only occurs with purified protein but also in intact cells.

COX-2 has significant regulatory flexibility: it is an allosteric protein, with multiple substrates and multiple allosteric regulators, all of which can influence how COX-2 operates on its substrates *in vivo*. The pathway analysis (Fig. 2.1*B* and 2.1*C*) suggests that COX-2 functions by first binding a substrate at the catalytic site, followed by binding of an allosteric regulator. Allostery can be viewed as a shift in the conformational free-energy landscape sampled by COX-2 through preferential binding of the allosteric regulator to particular conformations [146, 147]. From this perspective, modulating the concentrations of allosteric regulators in the COX-2 system shifts the conformational ensemble towards conformations favored by particular regulators. In the case of PG, these conformations are more easily turned over to product than the unmodulated enzyme, while for PG-G, the allosteric influence makes catalysis less energetically favorable (shifts the ensemble towards conformations that are less active). This allows COX-2 to manage the balance between PG and PG-G production in a more complex (and potentially farther-reaching) fashion than that provided by simple competition between substrates. This added complexity suggests a physiological reason why the COX-2 system would integrate information from multiple inputs: by adding a second competitive input, the system can access different responses than with a single input. Furthermore, the response dynamics of COX-2 gain even greater complexity because its inputs act as allosteric modulators in addition to substrates. The situation *in vivo* is likely far more complicated (and flexible) than considered here, as COX-2 has potential substrates in addition to AA and 2-AG [148], and some nonsubstrate fatty acids that act as allosteric regulators [36, 37, 42, 43, 149]. In addition, many of the nonsteroidal anti-inflammatory drugs that target COX-2 also may bind at either the catalytic or allosteric site.

8.1.2 Role of COX-2 Allostery *In Vivo*

Our findings that AA inhibits 2-AG oxygenation *in vitro* and that cellular AA and PG-G levels are inversely correlated lead one to question the degree to which allosteric control modulates PG-G synthesis in cells. Most studies of cellular PG-G production have used stimuli that trigger the release of high concentrations of free AA, only a portion of which is converted to PGs. In most cases, the ratio of AA to 2-AG in whole-cell lysates is at least 10:1, whereas the ratio of PGs to PG-Gs has been in the range of 5001,000:1 [150, 151]. In the experiments reported in our work, the PG:PG-G ratio was from 20- to 100- fold higher than the AA:2-AG ratio. These findings are consistent with our evidence that AA suppresses 2-AG oxygenation by COX-2. However, allosteric regulation is likely only one of a number of factors contributing to the high PG:PG-G ratio found in cells. Others include the kinetics of release of AA and 2-AG, the local concentrations of each substrate, and the hydroperoxide tone in the immediate vicinity of the enzyme.

8.1.3 Relationship Between AA and 2-AG *In Vivo*

In vivo, COX-2, AA, and 2-AG concentrations vary across cells in different tissues [152, 153, 154]. In most tissues, AA processed by COX-2 is released from membrane phospholipids, predominantly through the action of cytosolic phospholipase A2 [155]. In some tissues, (particularly the brain) a major source of AA is hydrolysis of 2-AG [156, 157]. In turn, 2-AG is also sourced from membrane phospholipids; through the sequential action of phospholipase C, which forms diacylglycerol (DAG), followed by conversion of DAG to 2-AG by DAG lipase [158]. Both DAG lipase and cytosolic phospholipase A2 are stimulated by increases in intracellular Ca^{2+} [155, 159]. Thus, many stimuli (such as zymosan phagocytosis by macrophages [151]) promote the release of AA and 2-AG simultaneously, with concentrations of AA typically higher than those of 2-AG. Considering the precursor-product relationship between 2-AG and AA, however, it is conceivable that in some cells, the levels of the two substrates may change inversely to one another, or that the level of one may change while the other remains constant. These considerations suggest that the system features we find that vary with AA and 2-AG level (pathway entropy and information transfer capacity) are states accessible by the true biological system with the attendant repercussions for information transfer within that system. In addition, the postulated link between diet and the substrates available for COX-2 turnover [160] suggests that the information transfer properties of the system could be modulated by fatty acid intake.

8.1.4 Ensemble Model Calibration

Every aspect of COX-2 analysis presented herein required the use of an ensemble model calibration method like PyDREAM. Such methods have the obvious benefit of providing an estimate of certainty in all predictions generated from the ensemble. For many CORM species, prediction uncertainties are broad (Fig. 8.1), indicating that a prediction generated from a single (or a few) vector(s) would be extremely misleading. In the case of COX-2 (and we hypothesize other systems as well), a kinetic parameter ensemble also provides the advantage of an understanding of how relationships between parameters are constrained.

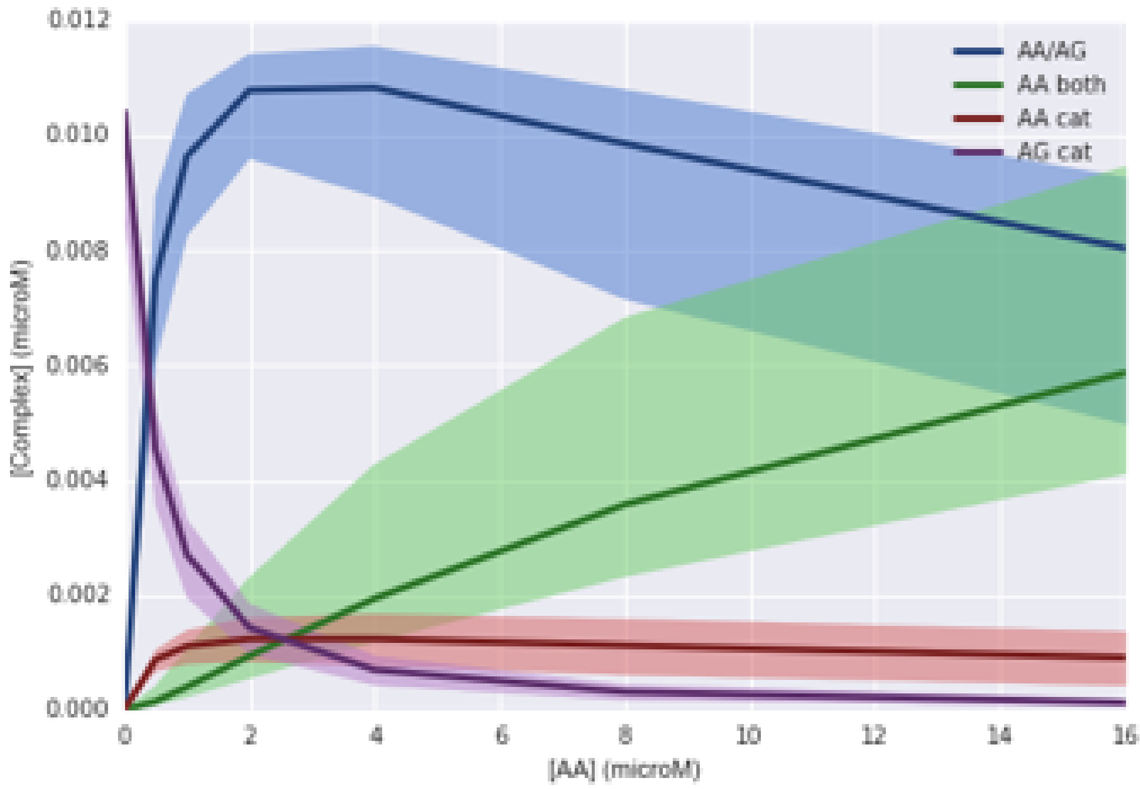


Figure 8.1: Predicted concentration of CORM species ten seconds after mixing as a function of AA concentration. Darker lines indicate the prediction generated from the most probable parameter vector in the ensemble and the shaded region encompasses predictions based on the entire ensemble of parameters.

In many systems biology models, even when individual parameters are not well constrained by available experimental data, relationships between the parameters are, a property referred to as *sloppiness* [161]. As in previous literature [162], many of the relationships between calibrated parameters in CORM are nonlinear (Fig. 8.2) and therefore not well approximated by linear analysis techniques such as PCA. While the quality and quantity of experimental calibration data and the predictions of interest will determine whether or not an ensemble method is critical to generate predictions from a given model, these results suggest that the importance of parameter ensembles for a given modeling problem should not be presumed unnecessary without first deploying a sampling method such as MCMC to determine if the problem would benefit from an ensemble analysis approach.

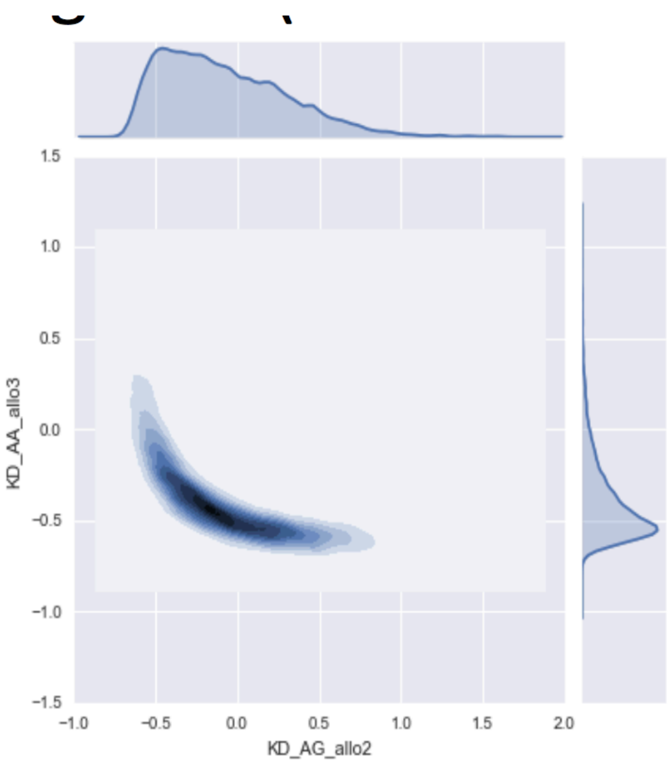


Figure 8.2: A kernel density estimate of the joint parameter probability density for two kinetic parameters in CORM, indicating a nonlinear relationship between the variables.

8.2 Future Directions

8.2.1 PyDREAM for Larger Models

As discussed in section 3.4, calibration of a larger model (more than 100 parameters) with PyDREAM was unsuccessful. Future work could focus on determining the necessary conditions to achieve convergence for a system like EARM within reasonable computational time constraints. Hypotheses for why these distributions failed to converge during this work include:

- Insufficient experimental data available to constrain parameter priors and allow convergence in a reasonable amount of time. The generic parameter priors selected are quite broad, spanning more than ten orders of magnitude in some cases (such as for generic k_f values, which are known to span this range biologically [62]). A recent article in *Nature Protocols* [163] describes a process for generating informative literature-based systems biology model priors that focuses largely on centrally massed, heavy-tailed priors. The authors suggest the use of such priors even for parameters without literature values, in which case priors are based on a generic parameter type (such as a k_f). Rigorous application of their process using the already curated set of literature-based parameter values discussed in this work could yield sufficient constraints to generate converged distributions for all parameters. It is also possible that convergence would be theoretically possible but only with sampling so long as to be essentially infinite, in which case MCMC methods that adapt more quickly than those available now would need to be developed and applied to this and similar problems.

- Possible bimodality in some parameters, causing certain chains to become stuck in particular modes and leading to failure of convergence checks (this behavior appears to be a possibility in the unconverged parameter shown in Fig. 3.17).

- Parameters in EARM that require violation of physiologically relevant ranges in order to fit experimental data; note that certain parameters (*i.e.* Fig. 3.16) that did converge with the added experimental and predictive constraints appear to be most probable on the edges of the boundaries set by the uniform prior. The parameter prior setting protocol discussed above [163] specifically advises against uniform priors because of the typical lack of such hard boundaries in actual biological systems. The parameter in Fig. 3.16 is a k_f and therefore the upper boundary of the prior is set by diffusion-limited association; converged values along this boundary could indicate that EARM requires values above the diffusion limit for this parameter in order to adequately reproduce experimental data, which could further suggest potential issues with the model itself.

Note that these possible explanations are not mutually exclusive. Future advances in superior adaptive MCMC methods, increased experimental data availability, and (perhaps, if the third possibility is an issue) greater accuracy in EARM itself could make achieving convergence possible, but fall outside the scope of this work.

8.2.2 The COX-2 Reaction Model

Further development of CORM could proceed in several directions:

- The addition of drug modules for COX-2 inhibitors with different binding modalities, including aspirin, ibuprofen, and others, as well as hypothetical drugs designed for maximum perturbation of the system.
- Upstream expansion of CORM to include synthesis of substrates, which introduces interactions both with dietary fatty acid intake and cellular phospholipids from which AA and 2-AG are cleaved.
- Downstream expansion of CORM to include specific PG and PG-Gs and their converting enzymes; because different prostaglandins can have opposite effects (on *e.g.* inflammation) this is key for tying CORM to an actual phenotypic outcome, rather

than the generic all prostaglandin proxy currently in use. Furthermore, different tissue types heterogeneously express the different prostaglandin synthases and the receptors that bind them to mediate downstream effects, meaning this effort is likely to create a more accurate picture of the actual biological function *in vivo* rather than a simplified ”average” picture across all tissues.

- Addition of other substrate and non-substrate regulators that bind COX-2. This complexity is difficult to measure *in vivo* or recreate *in vitro*, making it ideal for computational modeling.

Finally, a sensitivity analysis with CORM kinetic parameters and protein levels as inputs with a phenotypic outcome such as inflammation (as discussed above) as an output would be a useful means to probe the key drivers of system behavior and translate the model into actionable insights. The sensitivity analysis described in Chapter 6 represents a first step towards this goal.

BIBLIOGRAPHY

[1] Michelle M Mitchener, Daniel J Hermanson, Erin M Shockley, H Alex Brown, Craig W Lindsley, Jeff Reese, Carol A Rouzer, Carlos F Lopez, and Lawrence J Marnett. Competition and allostery govern substrate selectivity of cyclooxygenase-2. *Proceedings of the National Academy of Sciences of the United States of America*, 112(40):12366–12371, October 2015.

[2] Erin M Shockley, Jasper A Vrugt, and Carlos F Lopez. PyDREAM: high-dimensional parameter inference for biological models in python. *Bioinformatics*, 18:343, 2017.

[3] Sabrina L Spencer, Suzanne Gaudet, John G Albeck, John M Burke, and Peter K Sorger. Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature*, 459(7245):428–432, May 2009.

[4] Lorenz Adlung, Sandip Kar, Marie-Christine Wagner, Bin She, Sajib Chakraborty, Jie Bao, Susen Lattermann, Melanie Boerries, Hauke Busch, Patrick Wuchter, Anthony D Ho, Jens Timmer, Marcel Schilling, Thomas Höfer, and Ursula Klingmüller. Protein abundance of AKT and ERK pathway components governs cell type-specific regulation of proliferation. *Molecular Systems Biology*, 13(1):904, January 2017.

[5] Hoda Eydgahi, William W Chen, Jeremy L Muhlich, Dennis Vitkup, John N Tsitsiklis, and Peter K Sorger. Properties of cell death models calibrated and compared using Bayesian approaches. *Molecular Systems Biology*, 9(1):644–644, January 2013.

[6] Tian-Rui Xu, Vladislav Vyshemirsky, Amélie Gormand, Alex von Kriegsheim, Mark Girolami, George S Baillie, Dominic Ketley, Allan J Dunlop, Graeme Milligan, Miles D Houslay, and Walter Kolch. Inferring signaling pathway topologies

from multiple perturbation measurements of specific biochemical species. *Science Signaling*, 3(113):ra20–ra20, March 2010.

[7] Birgit Schoeberl, Emily A Pace, Jonathan B Fitzgerald, Brian D Harms, Lihui Xu, Lin Nie, Bryan Linggi, Ashish Kalra, Violette Paragas, Raghida Bukhalid, Viara Grantcharova, Neeraj Kohli, Kip A West, Magdalena Leszczyniecka, Michael J Feldhaus, Arthur J Kudla, and Ulrik B Nielsen. Therapeutically Targeting ErbB3: A Key Node in Ligand-Induced Activation of the ErbB Receptor–PI3K Axis. *Sci. Signal.*, 2(77):ra31–ra31, June 2009.

[8] Lily A Chylek, Leonard A Harris, James R Faeder, and William S Hlavacek. Modeling for (physical) biologists: an introduction to the rule-based approach. *Physical Biology*, 12(4):045007, July 2015.

[9] Ryan Suderman and Eric J Deeds. Machines vs. ensembles: effective MAPK signaling through heterogeneous sets of protein complexes. *PLoS computational biology*, 9(10):e1003278, 2013.

[10] Michael W Sneddon, James R Faeder, and Thierry Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8(2):177–183, February 2011.

[11] Leonard A Harris, Justin S Hogg, José-Juan Tapia, John A P Sekar, Sanjana Gupta, Ilya Korsunsky, Arshi Arora, Dipak Barua, Robert P Sheehan, and James R Faeder. BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, 32(21):3366–3368, October 2016.

[12] Carlos F Lopez, Jeremy L Muhlich, John A Bachman, and Peter K Sorger. Programming biological models in Python using PySB. *Molecular Systems Biology*, 9(1):646–646, January 2013.

[13] Boris N Kholodenko, Oleg V Demin, Gisela Moehren, and Jan B Hoek. Quantification of Short Term Signaling by the Epidermal Growth Factor Receptor. *Journal of Biological Chemistry*, 274(42):30169–30181, October 1999.

[14] Julio R Banga. Optimization in computational systems biology. *BMC Systems Biology*, 2(1):47, May 2008.

[15] Carmen G Moles, Pedro Mendes, and Julio R Banga. Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. *Genome Research*, 13(11):2467–2474, November 2003.

[16] J W Zwolak, J J Tyson, and L T Watson. Globally optimised parameters for a model of mitotic control in frog egg extracts. *IEE Proceedings - Systems Biology*, 152(2):81–92, June 2005.

[17] P Mendes and D Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14(10):869–883, 1998.

[18] William W Chen, Birgit Schoeberl, Paul J Jasper, Mario Niepel, Ulrik B Nielsen, Douglas A Lauffenburger, and Peter K Sorger. Input–output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Molecular Systems Biology*, 5:239, January 2009.

[19] Ivan Arisi, Antonino Cattaneo, and Vittorio Rosato. Parameter estimate of signal transduction pathways. *BMC Neuroscience*, 7(1):S6, October 2006.

[20] Yves Fomekong-Nanfack, Jaap A Kaandorp, and Joke Blom. Efficient parameter estimation for spatio-temporal models of pattern formation: case study of Drosophila melanogaster. *Bioinformatics*, 23(24):3356–3363, December 2007.

[21] Kuan-Yao Tsai and Feng-Sheng Wang. Evolutionary optimization with data collocation for reverse engineering of biological networks. *Bioinformatics*, 21(7):1180–1188, April 2005.

[22] Brandon R Thomas, Lily A Chylek, Joshua Colvin, Suman Sirimulla, Andrew H A Clayton, William S Hlavacek, and Richard G Posner. BioNetFit: a fitting tool compatible with BioNetGen, NFsim, and distributed computing environments. *Bioinformatics*, 32(5):btv655–800, November 2015.

[23] Maria Rodriguez-Fernandez, Jose A Egea, and Julio R Banga. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, 7(1):483, November 2006.

[24] Andreas Raue, Marcel Schilling, Julie Bachmann, Andrew Matteson, Max Schelke, Daniel Kaschek, Sabine Hug, Clemens Kreutz, Brian D Harms, Fabian J Theis, Ursula Klingmüller, and Jens Timmer. Lessons Learned from Quantitative Dynamical Modeling in Systems Biology. *PLoS ONE*, 8(9):e74335, September 2013.

[25] Jeffrey D Orth, Ines Thiele, and Bernhard Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248, March 2010.

[26] H Rabitz, M Kramer, and D Dacol. Sensitivity Analysis in Chemical Kinetics. *dx.doi.org*, 34(1):419–461, November 2003.

[27] Z Zi. Sensitivity analysis approaches applied to systems biology models. *IET Systems Biology*, 5(6):336–346, November 2011.

[28] C E Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423– 623–656, October 1948.

[29] Raymond Cheong, Alex Rhee, Chiaochun Joanne Wang, Ilya Nemenman, and An-

dre Levchenko. Information transduction capacity of noisy biochemical signaling networks. *Science*, 334(6054):354–358, October 2011.

[30] Jangir Selimkhanov, Brooks Taylor, Jason Yao, Anna Pilko, John Albeck, Alexander Hoffmann, Lev Tsimring, and Roy Wollman. Systems biology. Accurate information transmission through dynamic biochemical signaling networks. *Science*, 346(6215):1370–1373, December 2014.

[31] Ryan Suderman, John A Bachman, Adam Smith, Peter K Sorger, and Eric J Deeds. Fundamental trade-offs between information flow in single cells and cellular populations. *Proceedings of the National Academy of Sciences of the United States of America*, 114(22):5755–5760, May 2017.

[32] Mireille Alhouayek and Giulio G Muccioli. COX-2-derived endocannabinoid metabolites as novel inflammatory mediators. *Trends in Pharmacological Sciences*, 35(6):284–292, June 2014.

[33] Carol A Rouzer and Lawrence J Marnett. Endocannabinoid Oxygenation by Cyclooxygenases, Lipoxygenases, and Cytochromes P450: Cross-Talk between the Eicosanoid and Endocannabinoid Signaling Pathways. *Chemical Reviews*, 111(10):5899–5921, September 2011.

[34] C S Williams, M Mann, and R N DuBois. The role of cyclooxygenases in inflammation, cancer, and development. *Oncogene*, 18(55):7908–7916, December 1999.

[35] Vincenzo Di Marzo, Nephi Stella, and Andreas Zimmer. Endocannabinoid signalling and the deteriorating brain. *Nature reviews. Neuroscience*, 16(1):30–42, January 2015.

[36] Liang Dong, Narayan P Sharma, Brice J Jurban, and William L Smith. Pre-existent asymmetry in the human cyclooxygenase-2 sequence homodimer. *The Journal of biological chemistry*, 288(40):28641–28655, October 2013.

[37] Liang Dong, Alex J Vecchio, Narayan P Sharma, Brice J Jurban, Michael G Malkowski, and William L Smith. Human cyclooxygenase-2 is a sequence homodimer that functions as a conformational heterodimer. *The Journal of biological chemistry*, 286(21):19035–19046, May 2011.

[38] R J Kulmacz and W E Lands. Prostaglandin H synthase. Stoichiometry of heme cofactor. *Journal of Biological Chemistry*, 259(10):6358–6363, May 1984.

[39] Shalley N Kudalkar, Spyros P Nikas, Philip J Kingsley, Shu Xu, James J Galligan, Carol A Rouzer, Surajit Banerjee, Lipin Ji, Marsha R Eno, Alexandros Makriyannis, and Lawrence J Marnett. 13-Methylarachidonic Acid Is a Positive Allosteric Modulator of Endocannabinoid Oxygenation by Cyclooxygenase. *Journal of Biological Chemistry*, 290(12):7897–7909, March 2015.

[40] R J Kulmacz and W E Lands. Stoichiometry and kinetics of the interaction of prostaglandin H synthase with anti-inflammatory agents. *Journal of Biological Chemistry*, 260(23):12572–12578, October 1985.

[41] Gilad Rimon, Ranjinder S Sidhu, D Adam Lauver, Jullia Y Lee, Narayan P Sharma, Chong Yuan, Ryan A Frieler, Raymond C Trievel, Benedict R Lucchesi, and William L Smith. Coxibs interfere with the action of aspirin by binding tightly to one monomer of cyclooxygenase-1. *Proceedings of the National Academy of Sciences of the United States of America*, 107(1):28–33, January 2010.

[42] Chong Yuan, Ranjinder S Sidhu, Dmitry V Kuklev, Yuji Kado, Masayuki Wada, Inseok Song, and William L Smith. Cyclooxygenase Allosterism, Fatty Acid-mediated Cross-talk between Monomers of Cyclooxygenase Homodimers. *Journal of Biological Chemistry*, 284(15):10046–10055, April 2009.

[43] Liang Dong, Hechang Zou, Chong Yuan, Yu H Hong, Dmitry V Kuklev, and William L Smith. Different Fatty Acids Compete with Arachidonic Acid for Binding

to the Allosteric or Catalytic Subunits of Cyclooxygenases to Regulate Prostanoid Synthesis. *The Journal of biological chemistry*, 291(8):4069–4078, February 2016.

[44] C D Funk. Prostaglandins and leukotrienes: advances in eicosanoid biology. *Science*, 294(5548):1871–1875, November 2001.

[45] Carol A Rouzer and Lawrence J Marnett. Mechanism of free radical oxygenation of polyunsaturated fatty acids by cyclooxygenases. *Chemical Reviews*, 103(6):2239–2304, June 2003.

[46] W L Smith, D L DeWitt, and R M Garavito. Cyclooxygenases: structural, cellular, and molecular biology. *Annual review of biochemistry*, 69(1):145–182, 2000.

[47] J A Vrugt and C J F Ter Braak. Differential evolution Markov chain with snooker updater and fewer chains. *Statistics and Computing*, 18(4):435–446, 2008.

[48] Jasper A Vrugt and Eric Laloy. High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS)and high-performance computing. *Water Resources Research*, 48(1):W01526, 2012.

[49] William M Bolstad. *Introduction to Bayesian Statistics*. Bolstad/Bayesian Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA, January 2005.

[50] Roger Eckhardt. Stan ulam, john von neumann, and the monte carlo method. *Los Alamos Science*, 1987.

[51] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov Chains and Mixing Times*. American Mathematical Soc., 2008.

[52] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008.

[53] Jasper A Vrugt. Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB implementation. *Environmental Modelling & Software*, 75:273–316, January 2016.

[54] Jasper A Vrugt, Cajo J F ter Braak, Martyn P Clark, James M Hyman, and Bruce A Robinson. Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with Markov chain Monte Carlo simulation. *Water Resources Research*, 44(12):n/a–n/a, December 2008.

[55] J A Vrugt, CJF Ter Braak, and CGH Diks. Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of ...*, 10(3), 2009.

[56] Jun S Liu, Faming Liang, and Wing Hung Wong. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, 95(449):121–134, March 2000.

[57] J F Morrison. Kinetics of the reversible inhibition of enzyme-catalysed reactions by tight-binding inhibitors. *Biochimica et biophysica acta*, 185(2):269–286, 1969.

[58] L Michaelis and M L Menten. Die kinetik der invertinwirkung. *Biochemistry Zeitung*, 49:333–369, 1913.

[59] Alan E Gelfand and Adrian F M Smith. Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, 85(410):398–409, June 1990.

[60] A Gelman and D B Rubin. A single series from the Gibbs sampler provides a false sense of security. In J Bernardo, editor, *Bayesian Statistics*. 1992.

[61] Orane Guillaume-Gentil, Rashel V Grindberg, Romain Kooger, Livie Dorwling-Carter, Vincent Martinez, Dario Ossola, Martin Pilhofer, Tomaso Zambelli, and

Julia A Vorholt. Tunable Single-Cell Extraction for Molecular Analyses. *Cell*, 166(2):506–516, July 2016.

[62] Sanbo Qin, Xiaodong Pang, and Huan-Xiang Zhou. Automated Prediction of Protein Association Rate Constants. *Structure*, 19(12):1744–1751, December 2011.

[63] Ramzi Alsallaq and Huan-Xiang Zhou. Electrostatic rate enhancement and transient complex of protein–protein association. *Proteins: Structure, Function, and Bioinformatics*, 71(1):320–335, 2008.

[64] A Truneh, S Sharma, C Silverman, S Khandekar, M P Reddy, K C Deen, M M McLaughlin, S M Srinivasula, G P Livi, L A Marshall, E S Alnemri, W V Williams, and M L Doyle. Temperature-sensitive differential affinity of TRAIL for its receptors. DR5 is the highest affinity receptor. *Journal of Biological Chemistry*, 275(30):23319–23325, July 2000.

[65] Margarita Garcia-Calvo, Erin P Peterson, Dita M Rasper, John P Vaillancourt, Robert Zamboni, Donald W Nicholson, and Nancy A Thornberry. Purification and catalytic properties of human caspase family members. *, Published online: 08 April 1999; — doi:10.1038/sj.cdd.4400497*, 6(4):362–369, April 1999.

[66] Hua Zou, Ruomei Yang, Junshan Hao, Jean Wang, Chaohong Sun, Stephen W Fesik, Joe C Wu, Kevin J Tomaselli, and Robert C Armstrong. Regulation of the Apaf-1/caspase-9 apoptosome by caspase-3 and XIAP. *Journal of Biological Chemistry*, 278(10):8091–8098, March 2003.

[67] Kelly M Boatright, Martin Renatus, Fiona L Scott, Sabina Sperandio, Hwain Shin, Irene M Pedersen, Jean-Ehrland Ricci, Wade A Edris, Daniel P Sutherlin, Douglas R Green, and Guy S Salvesen. A Unified Model for Apical Caspase Activation. *Molecular Cell*, 11(2):529–541, February 2003.

[68] C Sun, M Cai, R P Meadows, N Xu, A H Gunasekera, J Herrmann, J C Wu, and S W Fesik. NMR structure and mutagenesis of the third Bir domain of the inhibitor of apoptosis protein XIAP. *Journal of Biological Chemistry*, 275(43):33777–33781, October 2000.

[69] Z Liu, C Sun, E T Olejniczak, R P Meadows, S F Betz, T Oost, J Herrmann, J C Wu, and S W Fesik. Structural basis for binding of Smac/DIABLO to the XIAP BIR3 domain. *Nature*, 408(6815):1004–1008, December 2000.

[70] S Riedl. Structural Basis for the Inhibition of Caspase-3 by XIAP. *Cell*, 104(5):791–800, March 2001.

[71] Q L Deveraux, R Takahashi, G S Salvesen, and J C Reed. X-linked IAP is a direct inhibitor of cell-death proteases. *Nature*, 388(6639):300–304, July 1997.

[72] Y Suzuki, Y Nakabayashi, K Nakata, J C Reed, and R Takahashi. X-linked inhibitor of apoptosis protein (XIAP) inhibits caspase-3 and -7 in distinct modes. *Journal of Biological Chemistry*, 276(29):27058–27063, July 2001.

[73] John Silke, Paul G Ekert, Catherine L Day, Christine J Hawkins, Manuel Baca, Joanne Chew, Miha Pakusch, Anne M Verhagen, and David L Vaux. Direct inhibition of caspase 3 is dispensable for the anti-apoptotic activity of XIAP. *The EMBO Journal*, 20(12):3114–3123, June 2001.

[74] Fiona L Scott, Jean-Bernard Denault, Stefan J Riedl, Hwain Shin, Martin Renatus, and Guy S Salvesen. XIAP inhibits caspase-3 and -7 using two binding sites: evolutionarily conserved mechanism of IAPs. *The EMBO Journal*, 24(3):645–655, February 2005.

[75] Cristina Pop, Brett Feeney, Ashutosh Tripathy, and A Clay Clark. Mutations in the Procaspase-3 Dimer Interface Affect the Activity of the Zymogen †. *Biochemistry*, 42(42):12311–12320, October 2003.

[76] Sanjeevan Shivakumar, Martin Kurylowicz, Nehad Hirmiz, Yaseen Manan, Ouided Friaa, Aisha Shamas-Din, Pourya Masoudian, Brian Leber, David W Andrews, and Cecile Fradin. The Proapoptotic Protein tBid Forms Both Superficially Bound and Membrane-Inserted Oligomers. *Biophysical Journal*, 106(10):2085–2095, May 2014.

[77] Aisha Shamas-Din, Scott Bindner, Weijia Zhu, Yehudit Zaltsman, Clinton Campbell, Atan Gross, Brian Leber, David W Andrews, and Cecile Fradin. tBid undergoes multiple conformational changes at the membrane required for Bax activation. *The Journal of biological chemistry*, 288(30):22111–22127, July 2013.

[78] Lieven P Billen, Candis L Kokoski, Jonathan F Lovell, Brian Leber, and David W Andrews. Bcl-XL Inhibits Membrane Permeabilization by Competing with Bax. *PLOS Biol*, 6(6):e147, June 2008.

[79] Loren D Walensky, Kenneth Pitter, Joel Morash, Kyoung Joon Oh, Scott Barbuto, Jill Fisher, Eric Smith, Gregory L Verdine, and Stanley J Korsmeyer. A stapled BID BH3 helix directly binds and activates BAX. *Molecular Cell*, 24(2):199–210, October 2006.

[80] Tudor Moldoveanu, Christy R Grace, Fabien Llambi, Amanda Nourse, Patrick Fitzgerald, Kalle Gehring, Richard W Kriwacki, and Douglas R Green. BID-induced structural changes in BAK promote apoptosis. *Nature Structural & Molecular Biology*, 20(5):589–597, May 2013.

[81] Michael Certo, Victoria Del Gaizo Moore, Mari Nishino, Guo Wei, Stanley Korsmeyer, Scott A Armstrong, and Anthony Letai. Mitochondria primed by death signals determine cellular addiction to antiapoptotic BCL-2 family members. *Cancer Cell*, 9(5):351–365, January 2006.

[82] Bonsu Ku, Chengyu Liang, Jae U Jung, and Byung-Ha Oh. Evidence that inhibition

of BAX activation by BCL-2 involves its tight and preferential interaction with the BH3 domain of BAX. *Cell Research*, 21(4):627–641, April 2011.

[83] Anthony Letai, Michael C Bassik, Loren D Walensky, Mia D Sorcinelli, Solly Weiler, and Stanley J Korsmeyer. Distinct BH3 domains either sensitize or activate mitochondrial apoptosis, serving as prototype cancer therapeutics. *Cancer Cell*, 2(3):183–192, September 2002.

[84] Ariele Viacava Follis, Fabien Llambi, Li Ou, Katherine Baran, Douglas R Green, and Richard W Kriwacki. The DNA-binding domain mediates both nuclear and cytosolic functions of p53. *Nature Structural & Molecular Biology*, 21(6):535–543, June 2014.

[85] Tomomi Kuwana, Lisa Bouchier-Hayes, Jerry E Chipuk, Christine Bonzon, Barbara A Sullivan, Douglas R Green, and Donald D Newmeyer. BH3 domains of BH3-only proteins differentially regulate Bax-mediated mitochondrial membrane permeabilization both directly and indirectly. *Molecular Cell*, 17(4):525–535, February 2005.

[86] Dayong Zhai, Chaofang Jin, Ziwei Huang, Arnold C Satterthwait, and John C Reed. Differential regulation of Bax and Bak by anti-apoptotic Bcl-2 family proteins Bcl-B and Mcl-1. *Journal of Biological Chemistry*, 283(15):9580–9586, April 2008.

[87] T Scott Chen, Hector Palacios, and Amy E Keating. Structure-Based Redesign of the Binding Specificity of Anti-Apoptotic Bcl-xL. *Journal of Molecular Biology*, 425(1):171–185, January 2013.

[88] Sean T Campbell, Kevin J Carlson, Carl J Buchholz, Mark R Helmers, and Indraneel Ghosh. Mapping the BH3 Binding Interface of Bcl-x L, Bcl-2, and Mcl-1 Using Split-Luciferase Reassembly. *Biochemistry*, 54(16):2632–2643, April 2015.

[89] J Mongkolsapaya, J M Grimes, N Chen, X N Xu, D I Stuart, E Y Jones, and G R Screaton. Structure of the TRAIL-DR5 complex reveals mechanisms conferring specificity in apoptotic initiation. *Nature structural biology*, 6(11):1048–1053, November 1999.

[90] S G Hymowitz, H W Christinger, G Fuh, M Ultsch, M O'Connell, R F Kelley, A Ashkenazi, and A M de Vos. Triggering cell death: the crystal structure of Apo2L/TRAIL in a complex with death receptor 5. *Molecular Cell*, 4(4):563–571, October 1999.

[91] S S Cha, B J Sung, Y A Kim, Y L Song, H J Kim, S Kim, M S Lee, and B-H Oh. Crystal structure of TRAIL-DR5 complex identifies a critical role of the unique frame insertion in conferring recognition specificity. *Journal of Biological Chemistry*, 275(40):31171–31177, October 2000.

[92] Qi Hu, Di Wu, Wen Chen, Zhen Yan, Chuangye Yan, Tianxi He, Qionglin Liang, and Yigong Shi. Molecular determinants of caspase-9 activation by the Apaf-1 apoptosome. *Proceedings of the National Academy of Sciences of the United States of America*, 111(46):16254–16261, November 2014.

[93] Eric N Shiozaki, Jijie Chai, Daniel J Rigotti, Stefan J Riedl, Pingwei Li, Srinivasa M Srinivasula, Emad S Alnemri, Robert Fairman, and Yigong Shi. Mechanism of XIAP-mediated inhibition of caspase-9. *Molecular Cell*, 11(2):519–527, February 2003.

[94] G Wu, J Chai, T L Suber, J W Wu, C Du, X Wang, and Y Shi. Structural basis of IAP recognition by Smac/DIABLO. *Nature*, 408(6815):1008–1012, December 2000.

[95] A Y Robin, K Krishna Kumar, D Westphal, A Z Wardak, G V Thompson, G Dewson, P M Colman, and P E Czabotar. Crystal structure of Bax bound to the BH3

peptide of Bim identifies important contacts for interaction. *Cell Death and Disease*, 6(7):e1809–e1809, July 2015.

[96] Peter E Czabotar, Dana Westphal, Grant Dewson, Stephen Ma, Colin Hockings, W Douglas Fairlie, Erinna F Lee, Shenggen Yao, Adeline Y Robin, Brian J Smith, David C S Huang, Ruth M Kluck, Jerry M Adams, and Peter M Colman. Bax crystal structures reveal how BH3 domains activate Bax and nucleate its oligomerization to induce apoptosis. *Cell*, 152(3):519–531, January 2013.

[97] Sreekanth Rajan, Minjoo Choi, Kwanghee Baek, and Ho Sup Yoon. Bh3 induced conformational changes in Bcl-Xl revealed by crystal structure and comparative analysis. *Proteins: Structure, Function, and Bioinformatics*, 83(7):1262–1272, July 2015.

[98] Qian Liu, Tudor Moldoveanu, Tara Sprules, Edna Matta-Camacho, Nura Mansur-Azzam, and Kalle Gehring. Apoptotic regulation by MCL-1 through heterodimerization. *The Journal of biological chemistry*, 285(25):19615–19624, June 2010.

[99] Peter E Czabotar, Erinna F Lee, Geoff V Thompson, Ahmad Z Wardak, W Douglas Fairlie, and Peter M Colman. Mutation to Bax beyond the BH3 domain disrupts interactions with pro-survival proteins and promotes apoptosis. *The Journal of biological chemistry*, 286(9):7123–7131, March 2011.

[100] M Sattler, H Liang, D Nettesheim, R P Meadows, J E Harlan, M Eberstadt, H S Yoon, S B Shuker, B S Chang, A J Minn, C B Thompson, and S W Fesik. Structure of Bcl-xL-Bak peptide complex: recognition between regulators of apoptosis. *Science*, 275(5302):983–986, February 1997.

[101] Piotr Wysoczanski, Robert J Mart, E Joel Loveridge, Christopher Williams, Sara B-M Whittaker, Matthew P Crump, and Rudolf K Allemann. NMR solution structure

of a photoswitchable apoptosis activating Bak peptide bound to Bcl-xL. *Journal of the American Chemical Society*, 134(18):7644–7647, May 2012.

[102] Catherine L Day, Callum Smits, F Cindy Fan, Erinna F Lee, W Douglas Fairlie, and Mark G Hinds. Structure of the BH3 domains from the p53-inducible BH3-only proteins Noxa and Puma in complex with Mcl-1. *Journal of Molecular Biology*, 380(5):958–971, July 2008.

[103] Peter E Czabotar, Erinna F Lee, Mark F van Delft, Catherine L Day, Brian J Smith, David C S Huang, W Douglas Fairlie, Mark G Hinds, and Peter M Colman. Structural insights into the degradation of Mcl-1 induced by BH3 domains. *Proceedings of the National Academy of Sciences*, 104(15):6217–6222, April 2007.

[104] Stefan Schuster and Claus Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *Journal of Biological Systems*, 02(02):165–182, June 1994.

[105] Jeremy S Edwards, Markus Covert, and Bernhard Palsson. Metabolic modelling of microbes: the flux-balance approach. *Environmental Microbiology*, 4(3):133–140, March 2002.

[106] A Hagberg, P Swart, and D S Chult. Exploring network structure, dynamics, and function using NetworkX. *Proceedings of the th Python in Science Conference SciPy*, 2008.

[107] Silvia D M Santos, Peter J Verveer, and Philippe I H Bastiaens. Growth factor-induced MAPK network topology shapes Erk response determining PC-12 cell fate. *Nature Cell Biology*, 9(3):324–330, March 2007.

[108] Toby Lawrence. The nuclear factor NF-kappaB pathway in inflammation. *Cold Spring Harbor Perspectives in Biology*, 1(6):a001651–a001651, December 2009.

[109] Doriane Lorendeau, Stefan Christen, Gianmarco Rinaldi, and Sarah-Maria Fendt. Metabolic control of signalling pathways and metabolic auto-regulation. *Biology of the cell*, 107(8):251–272, August 2015.

[110] Tujin Shi, Mario Niepel, Jason E McDermott, Yuqian Gao, Carrie D Nicora, William B Chrisler, Lye M Markillie, Vladislav A Petyuk, Richard D Smith, Karin D Rodland, Peter K Sorger, Wei-Jun Qian, and H Steven Wiley. Conservation of protein abundance patterns reveals the regulatory architecture of the EGFR-MAPK pathway. *Science Signaling*, 9(436):rs6–rs6, July 2016.

[111] Sui Huang. Non-genetic heterogeneity of cells in development: more than just noise. *Development (Cambridge, England)*, 136(23):3853–3862, December 2009.

[112] Adam James Waite, Nicholas W Frankel, Yann S Dufour, Jessica F Johnston, Junjiajia Long, and Thierry Emonet. Non-genetic diversity modulates population performance. *Molecular Systems Biology*, 12(12):895, December 2016.

[113] Simon Mitchell, Koushik Roy, Thomas A Zangle, and Alexander Hoffmann. Non-genetic origins of cell-to-cell variability in B lymphocyte proliferation. *Proceedings of the National Academy of Sciences of the United States of America*, 115(12):E2888–E2897, March 2018.

[114] Jia-Yun Chen, Jia-Ren Lin, Karlene A Cimprich, and Tobias Meyer. A two-dimensional ERK-AKT signaling code for an NGF-triggered cell-fate decision. *Molecular Cell*, 45(2):196–209, January 2012.

[115] G E Briggs and J B Haldane. A Note on the Kinetics of Enzyme Action. *Biochemical Journal*, 19(2):338–339, 1925.

[116] Herschel Rabitz. Global Sensitivity Analysis for Systems with Independent and/or Correlated Inputs. *Procedia - Social and Behavioral Sciences*, 2(6):7587–7589, 2010.

[117] Genyuan Li, Herschel Rabitz, Paul E Yelvington, Oluwayemisi O Oluwole, Fred Bacon, Charles E Kolb, and Jacqueline Schoendorf. Global Sensitivity Analysis for Systems with Independent and/or Correlated Inputs. *The Journal of Physical Chemistry A*, 114(19):6022–6032, May 2010.

[118] Genyuan Li, Xi Xing, William Welsh, and Herschel Rabitz. High dimensional model representation constructed by support vector regression. I. Independent variables with known probability distributions. *Journal of Mathematical Chemistry*, 55(1):278–303, 2016.

[119] Andrea Saltelli and Stefano Tarantola. On the Relative Importance of Input Factors in Mathematical Models. *Journal of the American Statistical Association*, 97(459):702–709, September 2002.

[120] Gartner group. Pattern-Based Strategy: getting value from Big Data , July 2011.

[121] IBM. Extracting business value from the 4 V's of big data.

[122] H V Jagadish. Big Data and Science: Myths and Reality. *Big Data Research*, 2(2):49–52, June 2015.

[123] Michael R Stratton, Peter J Campbell, and P Andrew Futreal. The cancer genome. *Nature*, 458(7239):719–724, April 2009.

[124] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, October 2008.

[125] Michael L Metzker. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11(1):31–46, December 2009.

[126] Travis B Murdoch and Allan S Detsky. The Inevitable Application of Big Data to Health Care. *JAMA*, 309(13):1351, April 2013.

[127] Clifford Lynch. How do your data grow? *Nature*, 455(7209):28–29, September 2008.

[128] Andrew R Joyce and Bernhard Ø Palsson. The model organism as a system: integrating 'omics' data sets. *Nature Reviews Molecular Cell Biology*, 7(3):198–210, March 2006.

[129] S T Rosenbloom, J C Denny, H Xu, N Lorenzi, W W Stead, and K B Johnson. Data from clinical notes: a perspective on the tension between structure and flexible documentation. *Journal of the American Medical Informatics Association*, 18(2):181–186, March 2011.

[130] Reality check on reproducibility. *Nature*, 533(7604):437–437, May 2016.

[131] Jake Luo, Min Wu, Deepika Gopukumar, and Yiqing Zhao. Big Data Application in Biomedical Research and Health Care: A Literature Review:. *Biomedical Informatics Insights*, 8:BII.S31559, January 2016.

[132] Cuiping Pan, Gregory McInnes, Nicole Deflaux, Michael Snyder, Jonathan Bingham, Somalee Datta, and Philip S Tsao. Cloud-based interactive analytics for terabytes of genomic variants data. *Bioinformatics*, 33(23):3709–3715, December 2017.

[133] Sheila M Reynolds, Michael Miller, Phyliss Lee, Kalle Leinonen, Suzanne M Paquette, Zack Rodebaugh, Abigail Hahn, David L Gibbs, Joseph Slagel, William J Longabaugh, Varsha Dhankani, Madelyn Reyes, Todd Pihl, Mark Backus, Matthew Bookman, Nicole Deflaux, Jonathan Bingham, David Pot, and Ilya Shmulevich. The ISB Cancer Genomics Cloud: A Flexible Cloud-Based Platform for Cancer Genomics Research. *Cancer Research*, 77(21):e7–e10, November 2017.

[134] Nick Weber, David Liou, Jennifer Dommer, Philip MacMenamin, Mariam Quiñones, Ian Misner, Andrew J Oler, Joe Wan, Lewis Kim, Meghan Coakley Mc-

Carthy, Samuel Ezeji, Karlynn Noble, and Darrell E Hurt. Nephele: a cloud platform for simplified, standardized and reproducible microbiome data analysis. *Bioinformatics*, 34(8):1411–1413, April 2018.

[135] Michael A Cianfrocco, Indrajit Lahiri, Frank DiMaio, and Andres E Leschziner. cryoem-cloud-tools: A software platform to deploy and manage cryo-EM jobs in the cloud. *Journal of structural biology*, 203(3):230–235, September 2018.

[136] Abhinav Nellore, Leonardo Collado-Torres, Andrew E Jaffe, José Alquicira-Hernández, Christopher Wilks, Jacob Pritt, James Morton, Jeffrey T Leek, and Ben Langmead. Rail-RNA: scalable analysis of RNA-seq splicing and coverage. *Bioinformatics*, 33(24):4033–4040, December 2017.

[137] Tara M Madhyastha, Natalie Koh, Trevor K M Day, Moises Hernández-Fernández, Austin Kelley, Daniel J Peterson, Sabreena Rajan, Karl A Woelfer, Jonathan Wolf, and Thomas J Grabowski. Running Neuroimaging Applications on Amazon Web Services: How, When, and at What Cost? *Frontiers in neuroinformatics*, 11:63, 2017.

[138] Jia Yu, Jochen Blom, Alexander Sczyrba, and Alexander Goesmann. Rapid protein alignment in the cloud: HAMOND combines fast DIAMOND alignments with Hadoop parallelism. *Journal of biotechnology*, 257:58–60, September 2017.

[139] Daniel Castaño-Díez. The Dynamo package for tomography and subtomogram averaging: components for MATLAB, GPU computing and EC2 Amazon Web Services. *Acta crystallographica. Section D, Structural biology*, 73(Pt 6):478–487, June 2017.

[140] Heath R Pardoe and Ruben Kuzniecky. NAPR: a Cloud-Based Framework for Neuroanatomical Age Prediction. *Neuroinformatics*, 16(1):43–49, January 2018.

[141] Yiqi Wang, Gen Li, Mark Ma, Fazhong He, Zhuo Song, Wei Zhang, and Chengkun

Wu. GT-WGS: an efficient and economic tool for large-scale WGS analyses based on the AWS cloud service. *BMC Genomics*, 19(Suppl 1):959, January 2018.

[142] Dirk Merkel. Docker: lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, March 2014.

[143] D C Swinney, A Y Mak, J Barnett, and C S Ramesha. Differential allosteric regulation of prostaglandin H synthase 1 and 2 by arachidonic acid. *Journal of Biological Chemistry*, 272(19):12393–12398, May 1997.

[144] W Chen, T R Pawelek, and R J Kulmacz. Hydroperoxide dependence and cooperative cyclooxygenase kinetics in prostaglandin H synthase-1 and -2. *Journal of Biological Chemistry*, 274(29):20301–20306, July 1999.

[145] Hechang Zou, Chong Yuan, Liang Dong, Ranjinder S Sidhu, Yu H Hong, Dmitry V Kuklev, and William L Smith. Human cyclooxygenase-1 activity and its responses to COX inhibitors are allosterically regulated by nonsubstrate fatty acids. *Journal of lipid research*, 53(7):1336–1347, July 2012.

[146] Bernhard C Lechtenberg, Stefan M V Freund, and James A Huntington. An ensemble view of thrombin allostery. *Biological chemistry*, 393(9):889–898, September 2012.

[147] Ruth Nussinov and Chung-Jung Tsai. Allostery in Disease and in Drug Discovery. *Cell*, 153(2):293–305, April 2013.

[148] Carol A Rouzer and Lawrence J Marnett. Cyclooxygenases: structural and functional insights. *Journal of lipid research*, 50 Suppl(Supplement):S29–34, April 2009.

[149] Liang Dong, Hechang Zou, Chong Yuan, Yu H Hong, Charis L Uhlson, Robert C Murphy, and William L Smith. Interactions of 2-O-arachidonylglycerol ether and

ibuprofen with the allosteric and catalytic subunits of human COX-2. *Journal of lipid research*, 57(6):1043–1050, June 2016.

[150] Carol A Rouzer, Susanne Tranguch, Haibin Wang, Hao Zhang, Sudhansu K Dey, and Lawrence J Marnett. Zymosan-induced glycerylprostaglandin and prostaglandin synthesis in resident peritoneal macrophages: roles of cyclo-oxygenase-1 and -2. *Biochemical Journal*, 399(1):91–99, October 2006.

[151] Carol A Rouzer and Lawrence J Marnett. Glycerylprostaglandin Synthesis by Resident Peritoneal Macrophages in Response to a Zymosan Stimulus. *Journal of Biological Chemistry*, 280(29):26690–26700, July 2005.

[152] Karen Seibert, Yan Zhang, Kathleen Leahy, Scott Hauser, Jaime Masferrer, and Peter Isakson. Distribution of Cox-1 and Cox-2 in Normal and Inflamed Tissues. In *Eicosanoids and Other Bioactive Lipids in Cancer, Inflammation, and Radiation Injury 2*, pages 167–170. Springer US, Boston, MA, 1997.

[153] A M Monjazeb. Arachidonic acid-induced gene expression in colon cancer cells. *Carcinogenesis*, 27(10):1950–1960, August 2006.

[154] T Sugiura, S Kishimoto, S Oka, and M Gokoh. Biochemistry, pharmacology and physiology of 2-arachidonoylglycerol, an endogenous cannabinoid receptor ligand. *Progress in Lipid Research*, 45(5):405–446, September 2006.

[155] Christina C Leslie. Cytosolic phospholipase A2: physiological function and role in disease. *Journal of lipid research*, 56(8):1386–1402, August 2015.

[156] B M Ignatowska-Jankowska, S Ghosh, M S Crowe, S G Kinsey, M J Niphakis, R A Abdullah, Q Tao, S T O' Neal, D M Walentiny, J L Wiley, B F Cravatt, and A H Lichtman. In vivocharacterization of the highly selective monoacylglycerol lipase inhibitor KML29: antinociceptive activity without cannabimimetic side effects. *British Journal of Pharmacology*, 171(6):1392–1407, March 2014.

[157] Jonathan Z Long, Weiwei Li, Lamont Booker, James J Burston, Steven G Kinsey, Joel E Schlosburg, Franciso J Pavón, Antonia M Serrano, Dana E Selley, Loren H Parsons, Aron H Lichtman, and Benjamin F Cravatt. Selective blockade of 2-arachidonoylglycerol hydrolysis produces cannabinoid behavioral effects. *Nature Chemical Biology*, 5(1):37–44, November 2008.

[158] Filomena Fezza, Monica Bari, Rita Florio, Emanuela Talamonti, Monica Feole, and Mauro Maccarrone. Endocannabinoids, Related Compounds and Their Metabolic Routes. *Molecules*, 19(11):17078–17106, November 2014.

[159] Tiziana Bisogno, Fiona Howell, Gareth Williams, Alberto Minassi, Maria Grazia Cascio, Alessia Ligresti, Isabel Matias, Aniello Schiano-Moriello, Praveen Paul, Emma-Jane Williams, Uma Gangadharan, Carl Hobbs, Vincenzo Di Marzo, and Patrick Doherty. Cloning of the first sn1-DAG lipases points to the spatial and temporal regulation of endocannabinoid signaling in the brain. *The Journal of Cell Biology*, 163(3):463–468, November 2003.

[160] Chu Chen. Lipids: COX-2's new role in inflammation. *Nature Chemical Biology*, 6(6):nchembio.375–402, June 2010.

[161] Kevin S Brown and James P Sethna. Statistical mechanical approaches to models with many poorly known parameters. *Physical Review E*, 68(2 Pt 1):021904, August 2003.

[162] Ryan N Gutenkunst, Fergal P Casey, Joshua J Waterfall, Christopher R Myers, and James P Sethna. Extracting Falsifiable Predictions from Sloppy Models. *Annals of the New York Academy of Sciences*, 1115(1):203–211, December 2007.

[163] Areti Tsigkinopoulou, Aliah Hawari, Megan Uttley, and Rainer Breitling. Defining informative priors for ensemble modeling in systems biology. *Nature Protocols*, 1:396, October 2018.

APPENDICES

Appendix I: PyDREAM Options Reference

nchains The number of parallel DREAM chains to run. Default: 5

niterations The number of iterations of the algorithm to run. Default: 50000

start A location in parameter space to begin the algorithm. This may be either a list of locations the same length as nchains, or a single location at which to start all chains. Default: None. Start all chains in random locations drawn from the prior.

restart A flag to use when continuing an earlier run. When used, will attempt to load earlier history and fitted probability values using the model name specified by the `model_name` argument. Default: False

verbose Trigger printing of detailed information relating to the run, such as acceptance or rejection of a jump and the current acceptance rate. Default: True

tempering Whether to use parallel tempering of the parallel DREAM chains. This feature has been very minimally tested. Use at your own risk! Default: False

nseedchains The number of draws to seed the history with at the start of a new DREAM run. Default: The number of sampled parameter dimensions times ten

nCR The number of different crossover probability values to use. Default: 3

adapt_crossover Whether to adapt crossover probability values. Default: True

adapt_gamma Whether to adapt gamma values. Default: False

crossover_burnin How many iterations to adapt crossover and/or gamma values. Default: 10% of the total iterations.

DEpairs The number of pairs of sampled past points to use for determining the next jump size. Default: 1

lamb Small random error to ensure ergodicity in walk. Default: .05

zeta Randomization term. Default: 10^{-12}

history_thin How many iterations to take before saving a point to the history. Default: 10

snooker Probability of proposing a snooker update. Default: .10

p_gamma_unity Probability of setting γ=1. Default: .20

gamma_levels Levels of γ adaptation (decreases default γ value as levels increase. Useful to get universally smaller jump sizes.) Default: 1

start_random Whether to start from a random location in parameter space. Default: True

save_history Whether to save the history to file at the end of the run. This also controls whether crossover and gamma level probabilities are saved to file at the end of the run. Default: True

history_file The name of a file to load a history of chain states from a previous run. This will be set automatically if you specify the `model_name` with `restart = True`.

crossover_file The name of a file to load a set of previously fit crossover probabilities. This will be set automatically if you specify the `model_name` with `restart = True`.

gamma_file The name of a file to load a set of previously fit gamma level probabilities. This will be set automatically if you specify the `model_name` with `restart = True`.

multitry Whether to use multiple trials for each chain at each iteration. This can be set to True, False, or a value for the number of multiple trials per iteration. Default: False. If set to True, the default number of multiple trials is 5.

parallel Whether to execute multiple trials in parallel (different DREAM chains are always run in parallel). Default: False

model_name A string for the model_name. This will be used when saving history, crossover probabilities, and gamma level probability files. It will also be used for automatic loading of previous run files when `restart = True`. Default: Save files with the current date and time in the file name.

Appendix II: AWS Batch Examples

Example Dockerfile

```
FROM amazonlinux:latest}

RUN yum -y install epel-release}

RUN yum -y update

RUN yum -y install unzip

RUN yum -y install aws-cli

RUN yum -y install java-1.8.0*

RUN yum -y install zip

RUN yum -y install sudo

RUN yum -y install gcc

RUN curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"

RUN python get-pip.py

RUN pip install plumbum

RUN pip install numpy

RUN yum -y install python27-devel

RUN yum -y install tcl

RUN yum -y install tcl-devel

RUN yum -y install tkinter27

RUN pip install seaborn

RUN pip install matplotlib

RUN yum -y install which

RUN mkdir /data

ADD EstCC/EstCC.jar /data

ADD fetch_and_run.sh /data

RUN chmod 777 /data/fetch_and_run.sh
```

```
ADD CORM_columns.txt /data
ADD modified_params.txt /data
WORKDIR /data
USER root

ENTRYPOINT ["/data/fetch_and_run.sh"]
```

Example Shell Script for a Containerized Job Run with AWS Batch

```
#!/usr/bin/env bash
#This script assumes the use of a job array and that the datasets
#to be analyzed have been uploaded to S3 and are numbered from 0
#to ARRAYSIZE-1 (a given job has environment variable
#$AWS_BATCH_JOB_ARRAY_INDEX with this number)
#It also assumes that the environment variable $INPUTDIR was set
#to the S3 path of your input files,
# and the environment variable $OUTPUTDIR
#was set to the S3 path to which you wish to save output files.
#If you passed any arguments to the script when setting the job
#definition, these will be stored in variables $1, $2, etc.
#The template below assumes no extra arguments to the script were
#included.

FILENAME=inputdata_$AWS_BATCH_JOB_ARRAY_INDEX.csv

#This command copies the input file in the given S3 input directory an
aws s3 cp $INPUTDIR$FILENAME /data
```

```
#Add specific analysis code to be run on the individual dataset.
#Remember that any programs required for analysis will need to be
#installed in the Docker container when creating the container image.
#For example, to run a Python script with an argument of the FILENAME 

cd /data
python analyze_data.py $FILENAME

#This command uploads the newly created output file to S3.
aws s3 cp outputdata_$AWSBATCH_JOB_ARRAY_INDEX.csv $OUTPUTDIR
```