Flexible Bayesian Regression Models for the Detection of Interactions in Exposure Mixture Studies

Elizabeth Zhang
April 19, 2024

Submitted to the Department of
Mathematics and Statistics
of Amherst College in partial fulfillment
of the requirements for the degree of
Bachelor of Arts with honors.

Advisor:
Amy Wagaman

Abstract

We encounter toxins everyday — in the food we eat, the water we drink, and the air we breathe. Studies of environmental hazards have typically focused on identifying the health effects of single exposures. However, we are invariably exposed to whole mixtures of chemicals, and analyses of their joint effects can provide more accurate estimates of true health risk. The study of these exposure mixtures presents unique statistical challenges, though, including the detection of non-additive interactions. This thesis explores emerging flexible Bayesian regression models for quantifying a wide range of interactions between multiple pollutants, and between pollutants and sociodemographic factors. We use simulation studies to compare two such methods: Bayesian Kernel Machine Regression (BKMR) and Bayesian Semiparametric Regression (BSR). Based on our findings, we recommend first using BKMR for estimating the overall exposure-response relationship, and then using BSR for formal inference on potential interactions. However, our simulations also illustrate the difficulties of detecting interactions, including a reliance on large sample sizes. As such, we highlight the importance of further methodological research in this area in order to progress toward a study of health that better acknowledges the relationality of our bodies to the environment and to each other.

Acknowledgments

First, this work was performed in part using high-performance computing equipment at Amherst College obtained under National Science Foundation Grant Number 2117377. The data used for simulations in this thesis was supported by the National Institute of Environmental Health Sciences of the National Institutes of Health under Award Numbers U2CES026555 and U2CES026553. The content is solely the responsibility of the author and does not necessarily represent the official views of the National Science Foundation or the National Institutes of Health.

Next, I have been extremely fortunate to receive the support of so many people, without whom this thesis would not have been possible. Thank you so much to my thesis advisor, Amy Wagaman, for her tireless support. Your encouragement, feedback, and guidance gave me the confidence to see this project through. Thank you to Alex Keil for first inspiring my interest in exposure mixture studies and thereafter providing invaluable advice on my ideas. Thank you to Shu-Min Liao for her advice on copulas, and to Victoria Nguyen for her guidance on how to think humanistically about toxicity. Thank you to Doug Hall for troubleshooting my code for the cluster with me. I would also like to acknowledge Brittney Bailey, whose early support encouraged me to first pursue a statistics major, and then to write this thesis. And to everyone in the Department of Mathematics and Statistics — the faculty, staff, and my peers — thank you for shaping the wonderful journey that I've taken through statistics at Amherst.

Finally, I want to express my gratitude for my family for their love and support. I would not be here without you. Thank you to my mom for believing in me and giving me the opportunity to take risks. Thank you to my friends for their support, energy, and enthusiasm. You all inspire me so much. And to Sara, I am so grateful to you for always encouraging me to keep going and for listening to me talk on and on about statistics. Thank you for sticking by my side through it all.

Table of Contents

Abstract . . . . . **i**

Acknowledgments . . . . . **iii**

List of Tables . . . . . **vii**

List of Figures . . . . . **x**

Chapter 1: Introduction **1**

Chapter 2: Humanistic Perspective **5**

Chapter 3: Bayesian Regression Methods **9**

3.1 Motivation 9

3.1.1 Interactions from a statistical perspective 9

3.1.2 Mechanistic and public health relevance 11

3.2 Bayesian Kernel Machine Regression (BKMR) 12

3.2.1 Kernel machine regression 14

3.2.2 Connection to mixed models 16

3.2.3 Toy example 17

3.2.4 Variable selection 20

3.2.5 Prior specification 23

3.2.6 The MCMC algorithm 24

3.3 Bayesian Semiparametric Regression (BSR) 25

3.3.1 Spline regression 26

3.3.2 Toy example . . . 28
3.3.3 Model formulation in BSR . . . 31
3.3.4 Sparsity inducing priors . . . 32
3.3.5 Prior specification . . . 33
3.3.6 The MCMC algorithm . . . 34
3.4 Detecting interactions . . . 35
3.4.1 BKMR . . . 36
3.4.2 BSR . . . 37
3.4.3 Differences between BKMR and BSR . . . 37
Chapter 4: Simulations . . . **39**
4.1 Past simulation studies . . . 39
4.2 Methods . . . 41
4.2.1 MADRES data . . . 41
4.2.2 Using copulas to simulate predictor data . . . 44
4.2.3 Simulating predictor-response relationships . . . 49
4.2.4 Models . . . 51
4.2.5 Model assessment . . . 53
4.3 Results . . . 55
4.3.1 Base case . . . 55
4.3.2 Univariate sensitivity . . . 57
4.3.3 Two-way interactions between chemicals . . . 60
4.3.4 Three-way interactions between chemicals . . . 66
4.3.5 Interactions between race and an exposure . . . 67
4.3.6 Run-time analysis . . . 70
4.4 Discussion . . . 73
Conclusion . . . **77**

Appendix A: Supplemental output **81**
A.1 Methods 81
A.2 Results 93
Appendix B: Code **109**
B.1 Code for Chapter 3 109
B.2 Code for Chapter 4.2 112
B.2.1 Cleaning MADRES data 113
B.2.2 Simulating predictor data 114
B.2.3 Simulating response data 122
B.2.4 Fitting models 131
B.3 Code for Chapter 4.3 and Appendix A.2 149
B.3.1 Extracting results 149
B.3.2 Presenting results 183
B.4 Code for Appendix A.1 219
Corrections **235**
References **237**

List of Tables

4.1 Specification of interaction terms in simulations. 50

4.2 Sensitivity and false discovery rate (FDR) of chemicals in base case scenario. 56

4.3 Overall sensitivity for univariate chemicals in all scenarios with interactions between chemicals. Multiplicative and polynomial are abbreviated mult. and poly., respectively. 58

4.4 Sensitivity to interactions in all scenarios with two-way interactions between exposures. 61

4.5 False discovery rate of interactions in all scenarios with two-way interactions between exposures. 64

4.6 Sensitivity to trivariate interactions between Hg, Ni, and Tl. 67

4.7 Sensitivity to interactions between the categorical race variable and Hg. 68

4.8 Average run-times in all scenarios with interactions between two or more chemicals, as well as the base case. 71

4.9 Average run-times in scenarios with an interaction between the categorical race covariate and Hg. 72

A.1 Overall false discovery rate for univariate chemicals in all scenarios with interactions between chemicals. 100

A.2 Sensitivity for the univariate Hg term in all scenarios with an interaction between the categorical race covariate and Hg. Sensitivities for BKMR and BSR models are stratified by race. 108

List of Figures

3.1 Non-linear data with a true relationship (orange) and a fitted linear regression (blue). 17

3.2 A query point of 12.5 and the weights of neighboring observations based on a Gaussian kernel. 18

3.3 Fitted kernel machine regression (blue) with $\rho = 2$ compared to the true relationship (orange). 19

3.4 Fitted kernel machine regression with $\rho = 0.02$ (undersmoothed) and $\rho = 50$ (oversmoothed). 20

3.5 Linear spline regression (blue) with four knots (dotted lines) compared to the true relationship (orange). 29

3.6 Cubic spline regression (blue) with four knots (dotted lines) compared to the true relationship (orange). 29

3.7 Natural spline regression (blue) with four knots (dotted lines) compared to the true relationship (orange). 30

3.8 Natural and cubic spline regression (blue) compared to the true relationship (orange) extrapolated outside the bounds of x (dotted lines). 31

4.1 Distributions of original (a) and natural log transformed (b) concentrations of metals in MADRES cohort (n=252). 42

4.2 Distributions of continuous (a) and categorical (b) covariates in the MADRES cohort (n=252). 44

4.3 Association between race by ethnicity and birth place and metal exposures in the MADRES cohort (n=252). 46

4.4 Distributions of log-transformed exposures from observed data (blue) and 2100 simulated smaller size (n=252) datasets (gray). 47

4.5 Distributions of continuous (a) and categorical (b) covariates from observed data (blue) and 2100 simulated smaller size (n=252) datasets (gray). 47

4.6 Spearman's correlation heat maps of exposures from observed data (a) and averaged across 2100 smaller size (n=252) simulated datasets (b). 48

4.7 P-value distributions from smaller (a) and larger (b) size datasets and PIP distributions from smaller (c) and larger (d) size datasets. 56

4.8 Flow chart showing relationship between detection of the univariate effects of As and Cd (a) and Ni and Co (b) with detection of their interactions. Only cases with a higher effect size, multiplicative interaction in a larger size dataset are included. 63

4.9 Exposure-response relationships estimated by BKMR in small (a) and large (b) datasets and BSR in small (c) and large (d) datasets, using the first chemical fixed at quantiles of another to assess interactions between Hg and Ni. All other chemicals are fixed at 0.5 quantiles. . . 65

4.10 Relationship between Hg and response estimated by stratified BKMR and BSR models in smaller and larger datasets. All other chemicals are fixed at 0.5 quantiles. Lower and higher refer to effect sizes, while small and large refer to dataset sizes. 69

A.1 Exposure-response relationship for univariate exposures in all models. Exposure values are log-scaled and then standardized. 81

A.2 Exposure-response surface for base case: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})}$. 82

A.3 Exposure-response surface for a multiplicative interaction between Hg and Ni at the lower effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.35\text{Hg}*\text{Ni}$. . . 82

A.4 Exposure-response surface for a multiplicative interaction between Hg and Ni at the higher effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.7\text{Hg}*\text{Ni}$. . . 83

A.5 Exposure-response surface for a polynomial interaction between Hg and Ni at the lower effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.3\text{Hg}*(\text{Ni}-1)^2$. 83

A.6 Exposure-response surface for a polynomial interaction between Hg and Ni at the higher effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.6\text{Hg}*(\text{Ni}-1)^2$. 84

A.7 Exposure-response surface for a multiplicative interaction between Cd and As at the lower effect size: $Y = 0.35\text{Cd}*\text{As}$. 84

A.8 Exposure-response surface for a multiplicative interaction between Cd and As at the higher effect size: $Y = 0.7\text{Cd}*\text{As}$. 84

A.9 Exposure-response surface for a polynomial interaction between Cd and As at the lower effect size: $0.125\text{Cd}*(\text{As}-1)^2$. 85

A.10 Exposure-response surface for a polynomial interaction between Cd and As at the higher effect size: $0.25\text{Cd}*(\text{As}-1)^2$. 85

A.11 Exposure-response surface for a multiplicative interaction between Ni and Co at the lower effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.3\text{Ni}*\text{Co}$. 85

A.12 Exposure-response surface for a multiplicative interaction between Ni and Co at the higher effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.6\text{Ni}*\text{Co}$. 86

A.13 Exposure-response surface for a polynomial interaction between Ni and Co at the lower effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.09\text{Ni}*(\text{Co}-1)^2$. 86

A.14 Exposure-response surface for a polynomial interaction between Ni and Co at the lower effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.18\text{Ni}*(\text{Co}-1)^2$. 87

A.15 Distributions of Spearman's correlation from 2100 smaller size (n=252) simulated datasets. 87

A.16 Distributions of exposures from observed data (blue) and simulated larger size (n=1000) datasets (gray). 88

A.17 Distributions of covariates from observed data (blue) and simulated larger size (n=1000) datasets (gray). 88

A.18 Spearman's correlation heat maps of exposures from observed data (a) and averaged from 2100 larger size (n=1000) simulated datasets (b), as well as distributions of correlations from larger size simulated datasets (c). 89

A.19 R^2 values from multiple linear regressions with only the true functional form of significant exposures in smaller size (a) and larger size (b) simulated datasets. 90

A.20 Test comparing WAIC selection of degrees of freedom from BSR models fit with either 5,000 or 50,000 MCMC iterations. 91

A.21 Examples of trace plots from smaller size BKMR (a) and BSR (b) as well as larger size BKMR (c) and BSR (d) in scenarios with larger effect size interactions between Hg and Ni. 92

A.22 P-value distributions of univariate chemicals from naive MLRs run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles. 94

A.23 P-value distributions of univariate chemicals from naive MLRs run on larger size (n=1000) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles. 94

A.24 P-value distributions of univariate chemicals from oracle MLRs run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Sensitivities are displayed above a point-range with the median and first and third quartiles. 95

A.25 P-value distributions of univariate chemicals from oracle MLRs run on larger size (n=1000) datasets, in all scenarios with interactions between chemicals. Sensitivities are displayed above a point-range with the median and first and third quartiles. 95

A.26 PIP distributions of univariate chemicals from BKMR models run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles. 96

A.27 PIP distributions of univariate chemicals from BKMR models run on larger size (n=1000) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles. 96

A.28 PIP distributions of univariate chemicals from BSR models run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles. 97

A.29 PIP distributions of univariate chemicals from BSR models run on larger size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles. 97

A.30 P-value distributions of interaction terms from oracle MLRs run on smaller size (n=252) datasets, from all scenarios with interactions between chemicals. 98

A.31 P-value distributions of interaction terms from oracle MLRs run on larger size (n=1000) datasets, from all scenarios with interactions between chemicals. 99

A.32 Exposure-response relationships estimated by BKMR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles. 101

A.33 Exposure-response relationships estimated by BSR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles. 101

A.34 Exposure-response relationships estimated by BKMR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles. 102

A.35 Exposure-response relationships estimated by BSR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles. 102

A.36 Exposure-response relationships estimated by BKMR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles. 103

A.37 Exposure-response relationships estimated by BSR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles. 103

A.38 Exposure-response relationships estimated by BKMR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles. 104

A.39 Exposure-response relationships estimated by BSR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles. 104

A.40 Exposure-response relationships estimated by BKMR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles. 105

A.41 Exposure-response relationships estimated by BSR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles. 105

A.42 Exposure-response relationships estimated by BKMR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles. 106
A.43 Exposure-response relationships estimated by BSR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles. 106

Chapter 1 Introduction

Rapid industrial development has created conditions of cumulative chronic toxicity which pose an acute risk to the wellbeing of humans and our living environment. In fact, it has been estimated that, globally, human activity releases chemicals at a rate of 220 billion tons per annum (Cribb, 2016). These staggering levels of pollution have led scholars to formally declare that humanity has surpassed the safe operating space of the planetary boundary for novel entities (Persson et al., 2022). As a result, exposure to low levels of pollutants has become an inevitable peril of daily life (Naidu et al., 2021; Vineis, 2018). In this new era of pervasive toxicity, understanding the nature and severity of health effects associated with chemical exposures is especially timely.

For this, we turn to epidemiological studies. The broad field of preventive epidemiology involves the identification of potentially modifiable risk factors that contribute to the burden of disease within human populations. Environmental epidemiology, in particular, considers the effect of environmental exposures — chemical or otherwise. However, studies concerning chemical pollutants in environmental epidemiology have historically focused on elucidating the effect and mechanisms of exposures to a single pollutant. In reality, humans are invariably exposed to numerous complex exposure mixtures which together contribute to the progression of adverse health outcomes. Therefore, risk assessments of single pollutants likely fail to capture the true consequences of these complex exposures (Heys, Shore, Pereira, Jones, & Mar-

tin, 2016). Assessing mixtures of chemicals can also have more direct implications for public health interventions. The United States Environmental Protection Agency (U.S. EPA) currently passes regulations for individual pollutants. In practice, though, regulation occurs by controlling the source of pollution, which is responsible for the production of a whole mixture of chemicals with specific joint effects on human health. As a result, the National Academies of Science (NAS) has advocated for a multipollutant regulatory approach, which is likely to be more protective of human health (NAS et al., 2017).

There are clear practical motivations for studies that examine the health effects of exposure to co-occurring mixtures of chemicals, hereafter referred to as exposure mixtures. However, expanding the focus of analysis from one exposure to multiple exposures introduces unique statistical challenges (Dominici, Peng, Barr, & Bell, 2010). In addition to a common issue of small effect sizes and small sample sizes present in most exposure analyses, multiple exposure analyses must also contend with high-dimensionality, collinearity, non-linear effects, and non-additive interactions (Yu et al., 2022). In particular, data with numerous pollutants, or predictors, require exponentially greater levels of complexity and time cost in analysis. Collinearity between exposures is common when analyzing pollutants from a single source and can lead to unstable estimates in a generalized linear model if left unaccounted for. Finally, exposures can have both non-linear single effects and non-additive interaction effects, which are difficult to capture unless explicitly specified in the model.

The classic multiple linear regression framework often fails to capture the true effects in this setting. In the past few years, a wide variety of statistical methods have been developed to overcome these challenges (see reviews, Gibson et al., 2019; Yu et al., 2022), which have been accompanied by a host of comparative simulation studies for general mixture scenarios (e.g., Hoskovec, Benka-Coker, Severson, Magza-

men, & Wilson, 2021; Lazarevic, Knibbs, Sly, & Barnett, 2020; Pesenti et al., 2023). However, to our knowledge, there has yet to be a simulation study which provides conclusive guidance about the ability of recently developed methods to conduct inference on non-additive interactions between exposures when the nature and effect sizes of these interactions vary. Moreover, there is no guidance in the literature on assessing interactions between covariates and exposures in exposure mixtures, including the potential of stratified models for detecting such interactions.

The goal of this thesis is to fill this gap in the literature by exploring the theory and performance of Bayesian regression techniques for quantifying complex interactions between multiple environmental exposures and related covariates. Specifically, we will compare two recently developed models for estimating the health effects of exposure mixtures: Bayesian Kernel Machine Regression (BKMR) (Bobb et al., 2015) and Bayesian Semiparametric Regression (BSR) (Antonelli et al., 2020).

In an age where anthropogenic actions have radically reshaped the earth, humanistic inquiry can offer critical insights into how we navigate and comprehend the hazards of our rapidly changing environment. We begin in Chapter 2 by contextualizing this thesis with a brief overview of cultural and social understandings of toxicity. Chapter 3 explains the motivation for studying interactions and provides background on the theory of Bayesian methods for analyzing exposure mixtures. Chapter 4 assesses the performance of these methods using a simulation study, based on a dataset with information on the relationship between prenatal exposure to heavy metals and gestational weight. We conclude with a discussion of the implications of this work for the future study of complex interactions in exposure mixture studies.

Chapter 2 Humanistic Perspective

In this section, we briefly introduce ideas from science and technology studies (STS), which track habits of thought that have fundamentally shaped the ways in which we think about and study toxicity. Our goal, here, is to contextualize exposure mixture studies — the motivation for this thesis — within larger regimes of knowledge production. We aim to recognize the limitations in current norms of understanding toxicity in order to inform research that more accurately reflects the complex realities of communities who bear the burdens of toxic exposure.

Modern scientific thinking encourages the onlooker to see objects in the world as distinct entities. For instance, when we talk about chemicals, we tend to talk about them as disconnected molecules: lead, mercury, per- and polyfluoroalkyl substances, etc. Clouds of pollution are conceived through their individual components, each of which becomes separated from its surroundings through the lenses of our imaginations (Myers, 2015). While it is easy to accept this ontology as inevitable, its origins can be located in the history of chemistry. The Industrial Revolution first turned chemistry into a profitable field, leading practitioners to become keenly interested in how structural representations of chemicals could be most useful for industrial and corporate technoscientific disciplines (Bensaude-Vincent & Stengers, 1996). Reducing chemical mixtures amidst complex environments down into discrete molecules made the quantification of their effects simpler — certain side effects or relations could be masked or hidden in technical reports. Such reports were then used by industrial

lobbyists as evidence for reducing or withholding environmental regulations (Murphy, 2017).

These lines of thinking have filtered out of industry and into scientific studies of environmental health, which have historically been permeated by a focus on studying single chemicals (Alaimo, 2010). The end goal has often been to obtain mechanistic explanations of their modes of toxicity. By isolating attention to a single chemical entity within a purely biological framework, one can simplify the problem; but, in so doing, one also captures only a narrow sliver of the complex social and physical realities of exposure (Murphy, 2004). This is not to dismiss such studies or their contributions when, in fact, they are necessary for understanding the mechanisms of exposure. Rather, we hope to acknowledge that there exist additional possibilities for the study of exposures when we shift perspective away from isolated molecules and into chemical relations.

We start by recognizing that an object cannot exist without also taking on relations to other beings — existence implies relationality. This concept of relationality disrupts the notion that objects can be bounded, the archetypal project of intellectual modernity. Indeed, the modern Cartesian split between body and mind idealizes humanness as a quality that is separable from the physical human form (Venn, 2010). In other words, the ideal human can be defined not by her corporeal body, or its physical relations to the environment, but instead by her intellect. Modernity insists that humans are not enmeshed in relations. However, toxicity is embodied in materials and in living beings and, as such, is inherently relational (Theriault & Kang, 2021). Evidence of toxic exposure, then, reinforces the dependence of the human on its corporeal form.

This is a problematic way to view chemical relations. Bodies that bear the burden of toxicity are cast as more porous to their environments, and thus less qualified

or worthy of humanness (Roberts, 2017). Such designations often take on a racial valence; structural preconditions place racial minorities in closer proximity to toxins, creating co-constituted geographies of toxicity and racialization (Packer, 2022). As a result, toxicity becomes embedded in racial hierarchies that qualify racialized groups as less human. Relationality encourages us to move away from this damaging epistemology. To this, we call on Michelle Murphy's definition of *alterlife*, which "names life already altered... [and] being in the contradictions of existing in worlds that demand chemical exposures as the conditions for eating, drinking, breathing," (2017, p. 497). Alterlife acknowledges that man-made chemical entities persist in every living being — it is no longer possible to live without being exposed. Moreover, it also highlights that the geographies of chemical exposures are inevitably mediated by unequal social conditions. Human life is alterlife. We are all bound together by a web of complex and unequal chemical relations.

It is critical that we explore modes of co-habitation that make these toxic conditions more livable. This task does not fall purely to science — other forms of knowledge are crucial for navigating the chemical hazards of our everyday life (Nguyen, 2020; Shapiro, 2015). The responsibility of science, though, is to become aware of how such hazards are lived and experienced. STS scholars have advocated for a departure from linear thinking, which singles out isolated causes, chemicals, and bodies (Lock & Nguyen, 2018; Murphy, 2004; Roberts, 2017). We argue that the methods for exposure mixtures examined in this thesis take us one step closer to this goal. In particular, exposure mixtures account for the realities of co-occuring chemical compounds and their non-linear effects. Testing for interactions between chemicals acknowledges that the relationality of these molecules with each other might affect their modes of action. In so doing, exposure mixtures push us to think beyond the limiting constructs of boundedness and isolation.

We also consider potential interactions between sociodemographic covariates and chemical exposures, a previously understudied form of interaction. As we have discussed, chemicals — despite being inanimate objects — are not fully separable from the societal context in which they exist. Pollutants become entangled in bodies that are themselves entangled in social relations. We link this idea to practices in social epidemiology, which posit that one's health is embodied within structural conditions (Krieger, 2001, 2011). In the context of exposure studies, this requires acknowledging that the effects of chemical exposure can be modulated along lines of race, class, and other discriminative categories that privilege certain groups over others.

Still, there is more work to do. We make a multitude of assumptions when describing human livelihoods with numbers and models. For instance, the race variable that we use attempts to capture the full lived experience of racism within five categories. Moreover, the studies of exposure that we focus on can also be forms of damage-centered research, which Tuck (2009) defines as work that documents unjust harms enacted against communities. While aimed at accountability and justice, such research can also unintentionally perpetuate norms of thinking about certain communities as inherently damaged. These harms can arise when we start to unconsciously attribute toxicity as the defining feature of the target populations of our studies, and toxic exposure becomes the primary reason that they are valuable to science. To remedy this, the research community must explore collaborative forms of inquiry that engage and empower communities in the research process, through a framework that Tuck (2009) describes as "desire-centered." And, there are certainly other considerations that we have missed. So, we hope that this is just the start of a conversation that continues to contend with how scientific studies of exposure can help us better make sense of our new toxic realities.

Chapter 3 Bayesian Regression Methods

3.1 Motivation

We are interested in using Bayesian regression techniques to characterize the nature of non-additive interactions in exposure mixture studies. We begin by reviewing definitions for what constitutes an interaction and why interactions are relevant from both a public health and biological viewpoint.

3.1.1 Interactions from a statistical perspective

First, we define additivity and non-additivity in the traditional statistical paradigm (Siemiatycki & Thomas, 1981). Suppose we have two variables x_1 and x_2, and we want to consider their effect on some outcome of interest. If specifying [effect due to x_1 and x_2] = [effect due to x_1] + [effect due to x_2] can adequately capture this relationship, then we say that x_1 and x_2 each have an **additive effect** on the outcome and that there is no interaction between them. On the other hand, if there is variability in the outcome that can be captured by an additional term equal to some function of x_1 and x_2, we say that there is a **non-additive interaction** between x_1 and x_2. In this case, [effect due to x_1 and x_2] = [effect due to x_1] + [effect due to x_2] + [effect due to $f(x_1, x_2)$], where f is a non-zero function.

For our sake, when we refer to "interaction," we mean any non-additive interaction. We consider such non-additive interactions to be complex, meaning that they are

difficult to detect. To see why, let us consider running a linear regression for Y on x_1 and x_2. The theoretical model would be defined as

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} f(x_1, x_2) + \varepsilon,$$

where the β's represent the effect sizes, and ε has a normal distribution with mean 0 and variance equal to the residual variance. We can see that the form of the interaction must be explicitly specified in the formulation of the model. Most commonly, a multiplicative interaction is assessed, where $f(x_1, x_2) = x_1 * x_2$. However, a non-additive interaction can take on many different forms, the true nature of which is difficult to determine analytically.

We used a two-predictor case above, but interactions can also exist between more variables (i.e., two-way by $f(x_1, x_2)$, three-way by $f(x_1, x_2, x_3)$, etc.). So, if we wanted to assess all possible interactions, the number to consider quickly becomes intractable in high-dimensional settings. For instance, consider modelling 10 predictors in the above linear regression setting. In order to be assessed, each interaction must be explicitly specified as a new term in the model. Even if we only considered one form for each interaction, including all possible two-way interactions would involve adding $\binom{10}{2} = 45$ additional terms to the model, and all possible three-way interactions would add $\binom{10}{3} = 120$ additional terms.

It is important also to acknowledge, here, that there is a limit to how many variables can be included in an interaction before it becomes incomprehensible to most humans. For instance, Halford, Baker, McCredden, & Bain (2005) suggest that there is a steep decline in interpretability from three- to four-way interactions, and that five-way interactions are only interpreted correctly at chance level. Hence, for practical purposes, we will limit our exploration to two- and three-way interactions.

3.1.2 Mechanistic and public health relevance

Thus far, we have discussed interactions within a statistical paradigm. However, in addition to being an interesting estimation challenge, non-additive interactions are also relevant in exposure mixture studies from both a mechanistic and public health point of view.

From a mechanistic perspective, a non-additive statistical interaction between two chemical exposures suggests that these compounds may be functionally interacting with each other. Theoretical models propose that such interactions can be classified as either synergistic or antagonistic (Heys et al., 2016; Plackett & Hewlett, 1952). In a synergistic interaction, the joint effects of a mixture exceed the independent effects of each component. This usually occurs if a chemical induces an enzyme involved with the activation of a second chemical or if a chemical inhibits an enzyme that would have otherwise degraded a second chemical. For example, it has been shown that organophosphates slow the degradation of pyrethoids by inhibiting detoxifying enzymes — these two classes of chemicals are often found together in commercial insecticide mixtures (Hernández et al., 2013).

On the other hand, in an antagonistic interaction, the joint effects of a mixture are less than their independent effects. This can occur either through competition at the target site of an enzyme or through direct chemical reactions with each other. In general, synergistic interactions are more concerning in risk assessments, as they lead to underestimation of the true toxicity of a mixture.

It should be noted, though, that while statistical interactions may provide some insight into how exposure mixtures are related to health, they cannot confirm their underlying biology (VanderWeele & Knol, 2014). If the goal is to assess a meaningful biological interaction, then the discovery of a statistical interaction should be followed

up by a functional study.

Now, from a public health perspective, we might be interested in how exposure mixtures interact with other covariates, or, in other words, how social and health factors might mediate the relationship between a health outcome and chemical exposures (VanderWeele & Knol, 2014). In our case, we can include these additional covariates in the exposure mixture model, where, statistically, they would contribute to the model in the same manner as another chemical exposure: a predictor. A statistical interaction in our model between a covariate and an exposure would indicate that the *magnitude* of the effect of reducing the level of an exposure might differ across various levels of the covariate. This finding could be relevant to public health policy makers, as the potential benefit of regulating a pollutant might differ across groups. For instance, it has been suggested that nutritional intake may modify susceptibility to chemical exposures (e.g., Kannan, Misra, Dvonch, & Krishnakumar, 2006; Kordas, Lönnerdal, & Stoltzfus, 2007).

In many cases, we might assess a covariate related to health inequity, such as socioeconomic status. We provide a cautionary comment, here, that an interaction term should not be the *sole* measure used to measure a health disparity (Ward et al., 2019). In this case, we should first consider the independent, additive association between the covariate and levels of exposure or rates of a health outcome, in order to contextualize the meaning of a potential interaction term.

3.2 Bayesian Kernel Machine Regression (BKMR)

In this section, we introduce the theory of BKMR. First, we define the notation that we will be using for kernel machine regression:

- X_m is an exposure in the exposure matrix $\mathbf{X}$ with $m = 1, \ldots, M$

- $\mathbf{x}_i$ is a vector of values for a single observation in $\mathbf{X}$ with $i = 1, \ldots, n$
- x_{im} is the ith observation of X_m
- $\mathbf{z}_i$ is a vector of covariates for a single observation in the matrix $\mathbf{Z}$, which contains a set of covariates, with $i = 1, \ldots, n$
- Y_i is an observation of $\mathbf{Y}$, measuring the health outcome in this case
- $h(\cdot)$ is the flexible function relating $\mathbf{x}$ to $\mathbf{Y}$
- k is the kernel function, the Gaussian in this case
- $\mathbf{K}$ is the $n \times n$ kernel matrix, with (i, j)th element $k(\mathbf{x}_i, \mathbf{x}_j)$
- ρ is the parameter which controls smoothness, associated with the kernel function
- τ is the parameter multiplied by the kernel matrix to relate $\mathbf{K}$ to h
- $\boldsymbol{\beta}_{\mathbf{z}}$ is a vector of the weights on the covariates, and
- $\boldsymbol{\varepsilon}_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ are the residuals of the response.

And, we define the notation that we will be using specific to BKMR:

- $r_m = 1/\rho_m$ is an augmented variable in $\mathbf{r}$ in the kernel matrix, controlling smoothness of the exposure-response relationship
- δ_m is an indicator variable in $\boldsymbol{\delta}$ which represents inclusion of the corresponding exposure in the model
- $\mathcal{S}_g$ is a group of partitioned predictors with $g = 1, \ldots, G$
- $\{\delta_m | \mathbf{x}_m \in \mathcal{S}_g\}$ is an indicator variable in $\boldsymbol{\delta}_{\mathcal{S}_g}$ which represents inclusion of a parameter in group g in the model
- π is the prior probability of inclusion of a predictor in the model, and
- $\lambda \equiv \tau\sigma^{-2}$ is used as a convenient way to define the prior on τ.

3.2.1 Kernel machine regression

We begin by introducing kernel machine regression, with attention to its specific implementation in BKMR. First proposed independently by Nadaraya (1964) and Watson (1964), kernel machine regression is a nonparametric regression technique that can be used to capture non-linear effects and non-additive interactions. In this introduction, we follow the presentation of kernel machine regression provided by Bobb et al. (2015).

To contextualize this method, we start at the typical linear regression setting,

$$Y_i = \mathbf{x}_i^\top \boldsymbol{\beta}_\mathbf{x} + \mathbf{z}_i^\top \boldsymbol{\beta}_\mathbf{z} + \varepsilon_i,$$

where Y_i measures a health outcome at a given point, $\mathbf{x}_i = [x_{i1}, \ldots, x_{iM}]$ is a vector of M exposures, $\mathbf{z}_i$ is a vector of covariates, $\boldsymbol{\beta}_\mathbf{x}$ and $\boldsymbol{\beta}_\mathbf{z}$ are vectors of weights for the exposures and covariates, respectively, and ε_i is a random variable where $\boldsymbol{\varepsilon} \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We can see that this function assumes that there is a linear relationship between the exposure and the response, and that the combined effects of multiple exposures are additive.

Kernel machine regression defines this relationship using a flexible function $h : \mathbb{R}^M \to \mathbb{R}$, where

$$Y_i = h(\mathbf{x}_i) + \mathbf{z}_i^\top \boldsymbol{\beta}_\mathbf{z} + \varepsilon_i.$$

Here, $h(\cdot)$ is represented by the function $k(\cdot, \cdot)$, a kernel. The kernel controls the covariance, or the similarity, between values of $h(\mathbf{x})$ and as such ensures that points near each other on the prediction surface will have similar values — or, in other words, that the prediction surface will be smooth. In the case of kernel machine regression,

we define a positive definite kernel where $k : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$. Note also that covariates are assumed to have a linear, additive effect on the response.

There are many choices of functions for k. BKMR uses the Gaussian kernel, also known as the radial basis function or, sometimes, the squared exponential kernel. The Gaussian kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = \exp\left\{ - \frac{||\mathbf{x} - \mathbf{x}'||^2}{\rho} \right\},$$

where $||\mathbf{x} - \mathbf{x}'||^2 = \sum_{m=1}^{M} (x_m - x'_m)^2$ for a set of exposure values $\mathbf{x}$ and the exposure values of another subject $\mathbf{x}'$, and ρ is a tuning parameter that controls the relationship of the correlation between two points with their distance. Greater values of ρ will enforce more dependence between points and make the resulting function smoother. h is related to k by a multiplicative constant τ, a tuning parameter which controls the vertical scale of h.

Now that we have defined h and k, we can think about how to characterize the relationship between our response and exposures. Kernel machine regression is a nonparametric technique because it does not specify a functional form for this relationship. Hence, we will think about estimating the response at a particular query point. Operationally, Müller (1987) demonstrated that kernel machine regression uses a weighted average of all the observations in the dataset to estimate the response, defined as

$$\bar{Y} = \frac{\sum_{i=1}^{n} w_i Y_i}{\sum_{i=1}^{n} w_i},$$

with some set of weights $\{w_i\}_{i=1}^{n}$. Intuitively, we want to weight the observations that are closer to the query point more heavily. Using the Gaussian kernel as a weight allows us to achieve this. Replacing the weight with the Gaussian kernel, we get

$$\bar{Y} = \frac{\sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i) Y_i}{\sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i)}.$$

As we move through the predictor space, we can think of the prediction as a continuous moving average of local points in the dataset. The correlation between two values of h is defined as

$$\text{cor}(h_i, h_j) = \exp\left\{ - \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\rho} \right\},$$

which allows us to see that values of h near each other will have a higher correlation and thus similar values. This is also why the resulting function is smooth.

3.2.2 Connection to mixed models

It is useful to make connections between this definition of kernel machine regression and mixed models. Liu, Lin, & Ghosh (2007) demonstrated this by representing $h(\mathbf{x})$ as following a Gaussian process probability distribution,

$$h(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \tau k(\mathbf{x}, \mathbf{x}')),$$

with covariance function k, where $\mathbf{x}$ is a vector of the exposure values, and $\mathbf{x}'$ contains the exposure values of another subject. A Gaussian process is a collection of random variables, of which any finite number follow a multivariate normal distribution (Schulz, Speekenbrink, & Krause, 2018). Here, we assume that the expected value of the h function with input $\mathbf{x}$ is $\mathbf{0}$. We use k for the covariance function, which represents the dependence between the function values with inputs $\mathbf{x}$ and $\mathbf{x}'$: $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(h(\mathbf{x}) - \mathbf{0})(h(\mathbf{x}') - \mathbf{0})]$.

Now, we can represent h as a collection of variables from a Gaussian process. h follows a multivariate normal distribution,

$$h(\mathbf{x}) \sim N(\mathbf{0}, \tau\mathbf{K}),$$

where $h(\mathbf{x}) = [h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_n)]^\top$ and $\mathbf{K}$ is the kernel matrix. The kernel matrix is an $n \times n$ matrix with (i, j)th element $k(\mathbf{x}_i, \mathbf{x}_j)$. Now, returning back to the regression view, we can think of each Y_i as following the distribution,

$$Y_i \overset{\text{ind}}{\sim} N(h(\mathbf{x}_i) + \mathbf{z}_i^\top \boldsymbol{\beta}_\mathbf{z}, \sigma^2) \text{ for } i = 1, \ldots, n,$$

where σ^2 comes from the variance of the residuals. Here, h can be interpreted as a random effect.

3.2.3 Toy example

In the following section, we illustrate kernel machine regression with a simulated toy example.

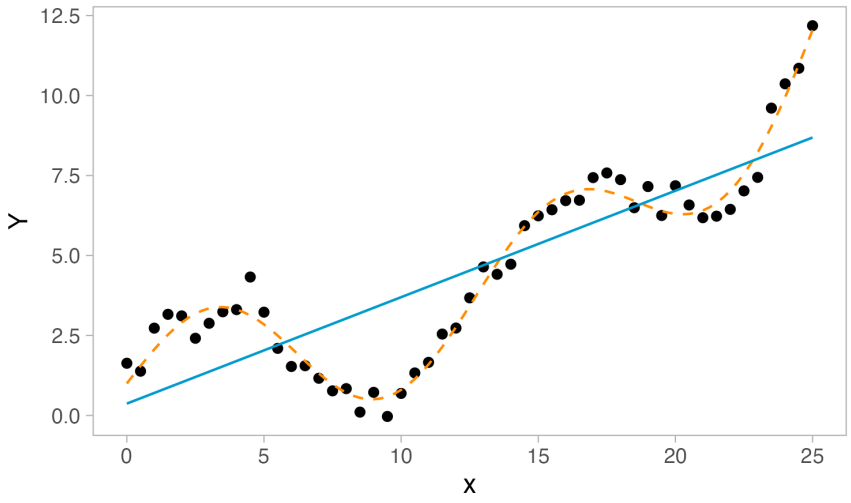

Figure 3.1: Non-linear data with a true relationship (orange) and a fitted linear regression (blue).

Consider the following case where we want to model the relationship between a

single predictor and a response variable. Suppose the true relationship between x and Y is defined $Y = e^{\frac{x}{10}} + 2\sin(\frac{x}{2})$. We simulate 51 equally spaced observations of x from 0 to 25, with error $\varepsilon_i \overset{\text{iid}}{\sim} N(0, 0.25)$.

Figure 3.1 illustrates the shape of our simulated non-linear data and the fit proposed by a simple linear regression. We can observe that the linear regression fails to capture the true non-linear relationship. In this case, this would lead to an underestimation of the true association between x and Y. Now, we will try to capture this relationship using kernel machine regression.

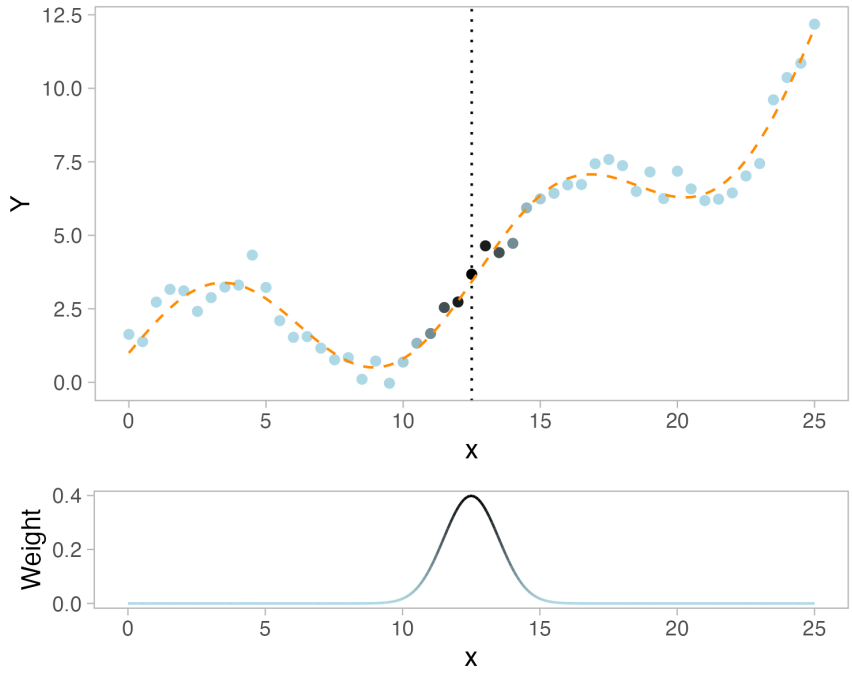

Figure 3.2: A query point of 12.5 and the weights of neighboring observations based on a Gaussian kernel.

To visualize how kernel machine regression works as a moving weighted average, we can consider a query point of 12.5. Figure 3.2 identifies the query point and assigns corresponding weights to the neighboring points based on a normal distribution, which shares the same density as the Gaussian kernel. In this case, we will specify $\rho = 2$,

which is synonymous with assigning the weights using a normal distribution with $\sigma^2 = 1$. We can see how an appropriate estimate for $h(12.5)$ can be obtained by taking a weighted average of the Y's, with those observations nearby weighted the most heavily.

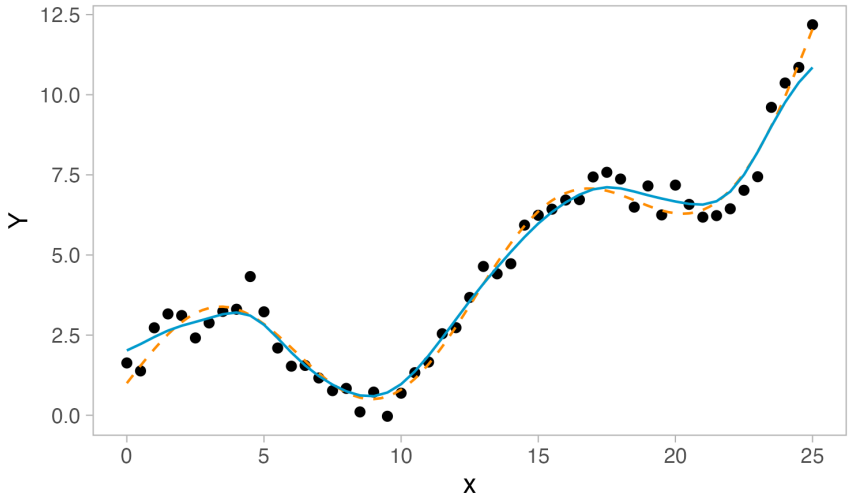

Figure 3.3: Fitted kernel machine regression (blue) with $\rho = 2$ compared to the true relationship (orange).

Now, we fit a kernel machine regression on this data with $\rho = 2$ using the `stats` package in R. We can see in Figure 3.3 that kernel machine regression captures the complex non-linear relationship between Y and x and closely follows the true relationship. We do note, though, that the estimation is less precise at the tails, where there is less information provided by local observations. We can also use this example to consider the effect of various values of ρ on the smoothness of the h function.

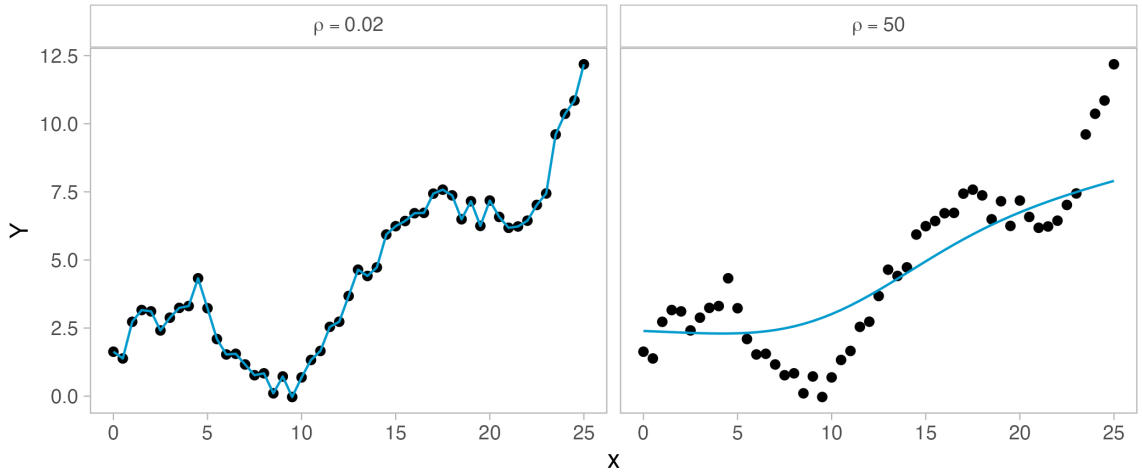

Figure 3.4: Fitted kernel machine regression with $\rho = 0.02$ (undersmoothed) and $\rho = 50$ (oversmoothed).

Figure 3.4 demonstrates the effect of relatively smaller and larger values of ρ on h. Decreasing the value of ρ allows kernel machine regression to overfit to the noise in the data by relaxing the dependence of neighboring values of h to each other. On the other hand, increasing the value of ρ enforces more dependence in h and as such results in an underfit estimation. Hence, the choice of ρ has a strong effect on the performance of kernel machine regression.

3.2.4 Variable selection

Now that we have defined kernel machine regression, we can extend it to the Bayesian paradigm. Bobb et al. (2015) showed that the Bayesian approach can outperform frequentist kernel machine regression because simultaneous variable selection and estimation can better capture the exposure-response relationship. In this section, we discuss the two methods for Bayesian variable selection in BKMR: hierarchical variable selection and component-wise variable selection (Bobb et al., 2015).

In order to perform variable selection, we define a parameter that puts a weight on each exposure. Each weight controls the degree to which its associated exposure

contributes to the model. In component-wise selection, we do this by augmenting the kernel function as

$$k(\mathbf{x}, \mathbf{x}'|\mathbf{r}) = \exp\Big\{ -\sum_{m=1}^{M} r_m(x_m - x'_m)^2 \Big\},$$

where $\mathbf{r} = [r_1, \ldots, r_M]^\top$. We define $r_m = 1/\rho_m$, the inverse of the tuning parameter ρ_m for each $\mathbf{x}_m$. Now, we can imagine that an exposure that is not closely associated with the response will be be assigned a value of r_m close to 0, which corresponds to a larger value of ρ_m. This larger value of ρ_m means that this exposure would contribute less to the exposure-response relationship, as depicted in the second panel of Figure 3.4.

We now define the kernel matrix $\mathbf{K}_{\mathbf{X},\mathbf{r}}$ as the $n \times n$ matrix with (i, j)th element $k(\mathbf{x}, \mathbf{x}'|\mathbf{r})$. To allow r_m to equal 0 with non-zero probability, we first define an indicator variable determining whether or not a predictor is included in the model, which is denoted and distributed as

$$\delta_m \sim \text{Bernoulli}(\pi),$$

where π is the prior probability of inclusion. Now, we can assume a "slab-and-spike" prior on r_m, distributed as

$$r_m|\delta_m \sim \delta_m f(r_m) + (1 - \delta_m)P_0,$$

where $f(\cdot)$ is some pdf with support $\mathbb{R}^+$, and P_0 denotes the density with point mass at 0.

While this process of component-wise variable selection works well in a typical multiple regression setting, it can lead to unreliable estimates in situations where the exposures are highly correlated with each other, which is common in exposure mixture

studies. In this case, the correlated components contribute similar information to the model, and component-wise variable selection is not able to distinguish which exposure is important. BKMR deals with this problem by introducing hierarchical variable selection (Bobb et al., 2015).

Hierarchical variable selection involves partitioning the predictors $\mathbf{x}_1, \ldots, \mathbf{x}_M$ into G groups, denoted $\mathcal{S}_g$ with $g = 1, \ldots, G$. These groups should be selected by the user based on prior knowledge, with the aim of keeping within-group correlation high and between-group correlation low (Bobb et al., 2015). For instance, consider a situation with 4 chemicals, Hg, Pb, As, and Sn. If Hg, Pb, and As were strongly correlated with each other and each weakly correlated with Sn, we might define $\mathcal{S}_1 = \{\text{Hg, Pb, As}\}$ and $\mathcal{S}_2 = \{\text{Sn}\}$.

The indicators from $r_m|\delta_m$ are now distributed as

$$\boldsymbol{\delta}_{\mathcal{S}_g}|\omega_g \sim \text{Multinomial}(\omega_g, \boldsymbol{\pi}_{\mathcal{S}_g}), g = 1, \ldots, G, \text{ and}$$
$$\omega_g \sim \text{Bernoulli}(\pi),$$

where $\boldsymbol{\delta}_{\mathcal{S}_g} = \{\delta_m | \mathbf{x}_m \in \mathcal{S}_g\}$ and $\boldsymbol{\pi}_{\mathcal{S}_g}$ are vectors of indicator variables and prior probabilities, respectively, of a exposure $\mathbf{x}_m$ in group $\mathcal{S}_g$ entering the model. By this approach, at most one exposure in each group is allowed to enter the model (Bobb et al., 2015).

While hierarchical variable selection resolves the issue of multicollinearity, it requires specifying subgroups of predictors a priori and assumes that one exposure in each group can capture the information of the rest. Hence, care should be taken to justify the partitioning of exposures when taking this approach (Bobb et al., 2015).

Note also that the posterior means of δ_m generated from these variable selection procedures represent the posterior probability of inclusion of $\mathbf{x}_m$. We can interpret

these posterior inclusion probabilities (PIPs) as measures of the relative importance of each exposure. These measures can be used to understand the contribution of each exposure to the health outcome of interest in the model (Bobb et al., 2015).

3.2.5 Prior specification

In this section, we specify the default prior distributions and parameters used by the BKMR algorithm (Bobb et al., 2015).

BKMR, by default, assumes $\rho_m = 1/r_m \sim \text{Unif}(a_r, b_r)$, a flat prior between a_r and b_r for which the default values are 0 and 100, respectively (Bobb, 2017a). This defines the prior probability of ρ as equally distributed across any value from 0 to 100. The inverse of this prior corresponds to the slab component of the "slab-and-spike" prior, where $r_m|\delta_m \sim \delta_m \text{Unif}^{-1}(a_r, b_r) + (1-\delta_m)P_0$. As a flat prior, this distribution should be chosen when we have no prior knowledge about the smoothness of the exposure-response function, with hyperparameters a_r and b_r selected to represent the range of values we expect ρ to potentially span.

We have seen that the smoothness of a kernel machine regression responds strongly to different values of $r_m = 1/\rho$, and, accordingly, the model fit of BKMR is sensitive to their prior distribution. In general, the PIPs generated from the variable selection procedure are particularly sensitive to this prior, though their relative importance tends to remain stable (Bobb, Claus Henn, Valeri, & Coull, 2018). As such, the BKMR algorithm also offers the options to define uniform and gamma priors for $r_m = 1/\rho$.

Moreover, BKMR assumes that the prior probability of including a predictor (δ_m) or group of predictors (ω_g) in the model is distributed $\pi \sim \text{Beta}(a_\pi, b_\pi)$. The default hyperparameters are $a_\pi = b_\pi = 1$, which represent a flat, uninformative prior between 0 and 1. When the hierarchical selection approach is applied, equal values for $\boldsymbol{\pi}_{\mathcal{S}_g}$,

the probabilities of inclusion for each component in group $\mathcal{S}_g$, are assumed.

Finally, BKMR assumes that the inverse of the variance of the residuals is distributed $\sigma^{-2} \sim \text{Gamma}(a_\sigma, b_\sigma)$, with default values of $a_\sigma = b_\sigma = 0.001$, and that the vertical scale of h is parameterized by $\lambda \equiv \tau\sigma^{-2} \sim \text{Gamma}(a_\lambda, b_\lambda)$, with default values of a_λ and b_λ such that the mean and variance of λ are both equal to 10.

3.2.6 The MCMC algorithm

Briefly, we discuss the algorithm used to find the solution in the BKMR package (Bobb et al., 2015, 2018), with commentary on its implications for the model fitting process.

BKMR uses a Markov chain Monte Carlo (MCMC) algorithm with a mix of Gibbs and Metropolis-Hastings samplers to estimate the posterior distributions of the parameters. In particular, a Gibbs step is used to update the distribution of σ^2, while a Metropolis-Hastings step is used to update the distribution of λ. For component-wise and hierarchical variable selection, $(\mathbf{r}, \boldsymbol{\delta}, \boldsymbol{\omega})$ are sampled jointly using a Metropolis-Hastings sampling scheme.

While each distribution generated by the Gibbs step is always accepted, the distributions for λ and r_m generated by the Metropolis-Hastings steps are accepted based on an acceptance rate (Wagaman & Dobrow, 2021). The standard deviation of the proposal distribution controls the acceptance rate and as such acts as a tuning parameter (Bobb, 2017b). In general, increasing the standard deviation leads to lower acceptance rates. Acceptance rates that are too low lead to slower convergence, but rates that are too high can cause convergence to a non-optimal distribution.

A major computational limitation of BKMR is that at each iteration of the MCMC algorithm, the $n \times n$ augmented kernel matrix $\mathbf{K}_{\mathbf{Z},\mathbf{r}}$ must be inverted multiple times. To offset this, BKMR can employ a Gaussian predictive process which involves spec-

ifying a set of l points, or "knots," that are a subset of the predictor space. The vector of predictors can be approximated by projection onto this lower dimensional space, which allows the algorithm to perform inversions on an $l \times l$ matrix. A general suggestion is to use this approach to speed up the algorithm when n is large and to specify $l \approx n/10$ (Bellavia, 2021).

3.3 Bayesian Semiparametric Regression (BSR)

In this section, we introduce the theory of BSR. First, we define the notation that we will be using for spline regression (much of it designed to be consistent with our BKMR presentation):

- X_m is a predictor variable in the predictor matrix $\mathbf{X}$ with $m = 1, \ldots, M$, measuring exposure variables or covariates
- $\mathbf{x}_i$ is a vector of values for a single observation in $\mathbf{X}$ with $i = 1, \ldots, n$
- $\mathbf{z}_i$ is a vector of covariates for a single observation in the matrix $\mathbf{Z}$, which contains a set of covariates, with $i = 1, \ldots, n$
- Y_i is an observation of $\mathbf{Y}$, measuring the health outcome in this case
- $f(\cdot)$ relates $\mathbf{x}_i$ to Y_i by a set of basis functions, $b_j(X)$
- β_j is a weight on the jth basis function
- P is the order of the basis expansion
- K is a set of ξ_k, $k = 1, \ldots, K$, interior knots defining $K + 1$ disjoint intervals, and
- $\boldsymbol{\varepsilon}_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ are the residuals of the response.

And, we define the notation that we will be using specific to BSR:

- $\widetilde{X_m}$ is a d-dimensional basis function expansion of X_m

- $\widetilde{X}_{m_1 m_2}$ is a d^2-dimensional basis expansion of the interaction between X_{m_1} and X_{m_2}
- $f^{(h)}(\cdot)$, where $h = 1, \ldots, H$ are a set of functions that sum up to $f(\cdot)$
- $\boldsymbol{\zeta} = \{\zeta_{mh}\}$ is an indicator for whether the mth predictor is included in the hth function
- $\boldsymbol{\beta}_S^{(h)}$ is a vector of all the coefficients on the predictors in function h
- σ_{β}^2 is the prior variance on the coefficients, and
- $\boldsymbol{\Sigma}_{\beta}$ is a diagonal matrix with the variances of the multivariate slab prior, $\sigma^2\sigma_{\beta}^2$, on the diagonals.

3.3.1 Spline regression

We begin by introducing spline regression, with attention to its specific implementation in BSR. Spline regression is a semiparametric regression technique that can be used to capture non-linear effects. In this introduction, we follow the presentation of spline regression provided by Antonelli et al. (2020), with additional details and explanation from Hastie, Tibshirani, & Friedman (2009).

BSR uses spline regression to define the regression relationship as

$$Y_i = f(\mathbf{x}_i) + \mathbf{z}_i^{\top}\boldsymbol{\beta}_{\mathbf{z}} + \varepsilon_i,$$

where f is defined by a set of basis functions on the exposures, $\mathbf{x}_i$, $\mathbf{z}_i$ and $\boldsymbol{\beta}_{\mathbf{z}}$ are the covariates and their associated weights, and ε_i is a random variable where $\boldsymbol{\varepsilon} \overset{\text{iid}}{\sim} N(0, \sigma^2)$. BSR uses natural spline bases (Antonelli et al., 2020). In order to understand how these are constructed, we start with a broad definition of basis expansions, before exploring linear, cubic, and then natural spline bases.

We determine the basis expansion by considering a piece-wise function of X_m with

some order P and some set of K knots defining $K+1$ disjoint intervals. BSR places knots at uniformly sized quantiles within the boundaries of X_m. The most commonly used orders are $P = 1, 2, \text{and } 4$, the constant, linear, and cubic splines, respectively. To begin, let us consider a continuous piece-wise linear spline basis (i.e., $P = 2$) of a one-dimensional X with two interior knots. In this case, we use the following four basis functions:

$$b_1(X) = 1, \quad b_2(X) = X, \quad b_3(X) = (X - \xi_1)_+, \quad b_4(X) = (X - \xi_2)_+,$$

where ξ_1 and ξ_2 are the two interior knots, and t_+ denotes the positive part. These bases are used to construct the regression model $f(X) = \sum_{j=1}^{4} \beta_j b_j(X)$, which requires estimating $K + P = 4$ parameters. We can check the continuity restrictions at the knots by seeing that $f(\xi_1^-) = \beta_1 + \xi_1\beta_2$ and $f(\xi_1^+) = \beta_1 + \xi_1\beta_2 + (\xi_1 - \xi_1)\beta_3$ are equal, and likewise at the second knot (Hastie et al., 2009).

Now, in the case of exposure mixtures, we want smoother functions that can capture the non-linear relationship between the response and the predictors. We can achieve this by increasing the order to $P = 4$ and using a cubic spline, with continuous first and second derivatives at the knots. The cubic spline is the lowest-order spline for which knot-discontinuity cannot be detected by the human eye (Hastie et al., 2009). For example, for one X with two interior knots, we use the following six basis functions:

$$b_1(X) = 1, \quad b_2(X) = X, \quad b_3(X) = X^2,$$
$$b_4(X) = X^3, \quad b_5(X) = (X - \xi_1)_+^3, \quad b_6(X) = (X - \xi_2)_+^3.$$

Now, the regression model is defined as $f(X) = \sum_{j=1}^{6} \beta_j b_j(X)$ and requires estimating

$K + P = 6$ parameters. It can be shown that $f'(\xi_i^-) = f'(\xi_i^+)$ and $f''(\xi_i^-) = f''(\xi_i^+)$ for $i = 1, 2$ (Hastie et al., 2009).

However, the behavior of polynomials near the boundaries of X, where there is less information, can be erratic. Natural cubic splines, also referred to as just natural splines, address this by imposing an additional restriction of linearity at the boundaries of X. Paradoxically, this also leads to a simpler model with four fewer parameters to estimate. A general definition of the K basis functions for a natural spline with interior knots ξ_j, $j = 1, \ldots, K$, is given by:

$$b_1(X) = 1, \qquad b_2(X) = X, \qquad b_{k+2}(X) = d_k(X) - d_{K-1}(X),$$
$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}.$$

Here, the regression model is defined as $f(X) = \sum_{j=1}^{K} \beta_j b_j(X)$, with K parameters (Hastie et al., 2009). BSR uses natural splines to specify the regression relationship.

3.3.2 Toy example

In the following section, we illustrate spline regression using the same toy example used to introduce kernel machine regression. See Chapter 3.2.3 and Figure 3.1 for details on the parameters used to generate simulated data.

As in Chapter 3.2.3, we consider a case where we want to model the relationship between a single exposure and a response variable, where the true relationship between x and Y is defined as $Y = e^{\frac{x}{10}} + 2\sin(\frac{x}{2})$. We fit a series of linear, cubic, and then natural spline regressions to illustrate the general framework of a natural spline regression.

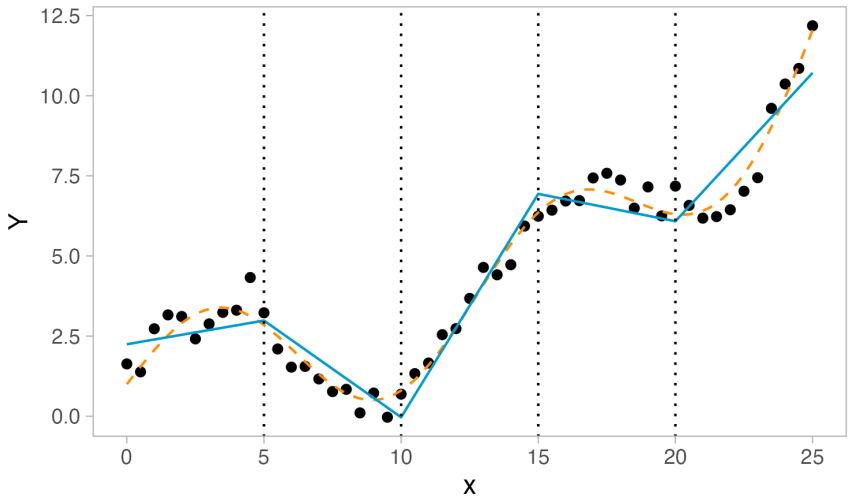

Figure 3.5: Linear spline regression (blue) with four knots (dotted lines) compared to the true relationship (orange).

Figure 3.5 illustrates the fit proposed by a linear spline regression with order $P = 2$. We can see that the implementation of knots allows for even a linear fit to capture more of the nuances in this nonlinear relationship, as compared to a standard linear regression.

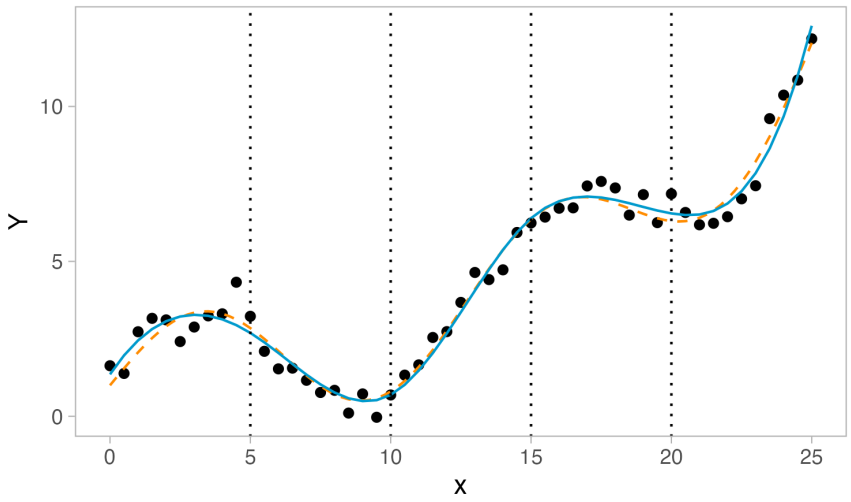

Figure 3.6: Cubic spline regression (blue) with four knots (dotted lines) compared to the true relationship (orange).

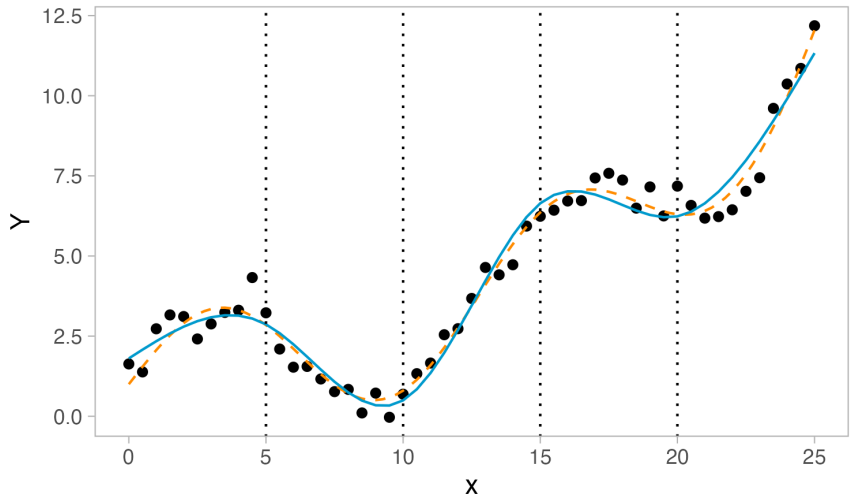

Figure 3.7: Natural spline regression (blue) with four knots (dotted lines) compared to the true relationship (orange).

However, this linear spline regression is still unable to fully estimate the nonlinearity in our example. Increasing the order to $P = 4$ with a cubic spline regression offers additional flexibility. Figure 3.6 illustrates the fit proposed by this model. Here, we can see the benefits of using a cubic polynomial relationship in a nonlinear setting: the estimated relationship is continuous at the knots, and the nonlinear relationship has been flexibly captured.

Our final modification involves imposing linearity constraints on the boundaries of x to implement a natural spline regression. Figure 3.7 shows the fit estimated using a natural spline regression. The fitted line is only slightly different than that proposed by a cubic spline regression in Figure 3.6. The most noticeable difference is that the slopes of the tails of the natural spline regression are less extreme than for the cubic spline regression.

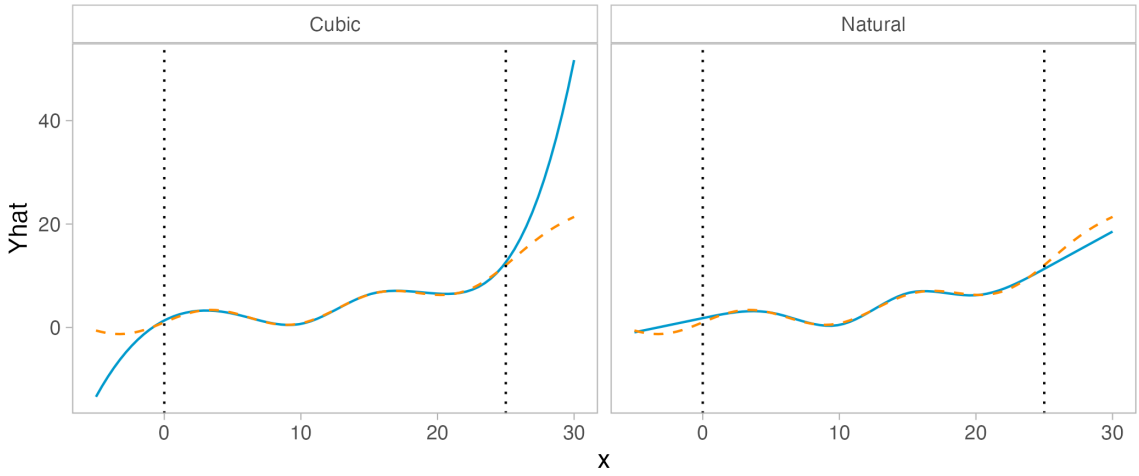

Figure 3.8: Natural and cubic spline regression (blue) compared to the true relationship (orange) extrapolated outside the bounds of x (dotted lines).

Extrapolation outside the scope of x allows us to see the effect of the linearity constraints imposed by natural regression. Figure 3.8 demonstrates how the cubic spline regression behaves erratically outside the bounds of x, as the cubic polynomial lines tend toward $\pm\infty$, while the natural spline regression follows a more appropriate linear trend. As the natural spline regression is more reliable near the boundaries of x and also simpler to estimate, BSR adopts the use of natural splines.

3.3.3 Model formulation in BSR

Now that we have defined natural splines, we introduce BSR, following the presentation in Antonelli et al. (2020). We demonstrate the construction of f in BSR by first assuming a two-dimensional case with exposures X_1 and X_2. We define

$$f(X_1) = \widetilde{X}_1\boldsymbol{\beta}_1, \qquad f(X_2) = \widetilde{X}_2\boldsymbol{\beta}_2,$$
$$f(X_1, X_2) = \widetilde{X}_1\boldsymbol{\beta}_1 + \widetilde{X}_2\boldsymbol{\beta}_2 + \widetilde{X}_{12}\boldsymbol{\beta}_{12},$$

where $\widetilde{X}_m = [b_{m1}(X_m), \ldots, b_{md}(X_m)]$ represents a d-dimensional basis function expansion for $m = 1, 2$, and $\widetilde{X}_{12} = [b_{11}(X_1)b_{21}(X_2), b_{11}(X_1)b_{22}(X_2), \ldots, b_{1d}(X_1)b_{2d}(X_2)]$ represents a d^2-dimensional basis expansion of the interaction between X_1 and X_2. d is an influential tuning parameter. BSR by default assumes that all exposures have the same number of degrees of freedom and uses the Watanabe-Akaike (WAIC) model selection criterion to select d, which approximates leave one out cross validation. Note that we must explicitly model the effect of the interaction term by assuming a multiplicative interaction between the basis functions of the predictors.

Extending to the multi-dimensional setting, BSR assumes the following general model formulation:

$$f(\mathbf{x}_i) = \sum_{h=1}^{H} f^{(h)}(\mathbf{x}_i),$$
$$f^{(h)}(\mathbf{x}_i) = \sum_{m_1=1}^{M} \widetilde{x}_{im_1}\boldsymbol{\beta}_{m_1}^{(h)} + \sum_{m_1=2}^{M}\sum_{m_2<m_1} \widetilde{x}_{im_1m_2}\boldsymbol{\beta}_{m_1m_2}^{(h)} + \ldots,$$

where $f^{(h)}(\mathbf{x}_i)$ includes a summation of all M-way interactions. The inclusion of all M-way interactions makes the model far too overparameterized. Moreover, $f(\mathbf{x}_i)$ is a sum of k different functions $f^{(h)}(\mathbf{x}_i)$ where a value for H is selected in order to capture all exposure effects in the model. Each of the H functions has the same functional form, and so the regression coefficients for a function $f^{(h)}(\mathbf{x}_i)$ are only identifiable up to a constant — this means that there are multiple sets of coefficients that could be estimated from the same data.

3.3.4 Sparsity inducing priors

In order to handle the overparameterization and non-identifiability of the model, BSR implements multivariate sparsity inducing priors. In this section, we follow the presentation provided in Antonelli et al. (2020).

First, we define indicators $\boldsymbol{\zeta} = \{\zeta_{mh}\}$ representing whether the mth exposure is included in the hth function:

$$P(\zeta_{mh} = 1) = \tau_h^{\zeta_{mh}}(1 - \tau_h)^{1-\zeta_{mh}} I(A_h \not\subset A_{h'} \forall h' \neq h \text{ or } A_h = \{\}),$$
$$\text{where } A_h = \{m : \zeta_h = 1\}.$$

Here, the indicators follow a Bernoulli distribution with prior probability of inclusion τ_h. The posterior means of $\boldsymbol{\zeta}$, i.e. the PIPs, can be interpreted as measures of relative variable importance. We include an indicator function $I()$ that represents whether the function h contains a unique set of predictors. This indicator ensures that no function contains exposures that are a subset of those in another function, h', in which case this function would be redundant and thus removed from the model entirely.

Now, we assume a multivariate slab-and-spike prior on the regression coefficients:

$$P(\boldsymbol{\beta}_S^{(h)}|\boldsymbol{\zeta}) = \left(1 - \prod_{m \in S} \zeta_{mh}\right) P_{\mathbf{0}} + \left(\prod_{m \in S} \zeta_{mh}\right) \psi_1(\boldsymbol{\beta}_S^{(h)}),$$
$$\text{where } S \text{ is some subset of } 1, 2, \ldots, m.$$

Here, $P_{\mathbf{0}}$ denotes the density with point mass at $\mathbf{0}$, and $\psi_1()$ is a multivariate normal distribution with mean $\mathbf{0}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$, a diagonal matrix with $\sigma^2\sigma_{\boldsymbol{\beta}}^2$ on the diagonals.

3.3.5 Prior specification

In this section, we discuss the priors and their default specifications in BSR (Antonelli et al., 2020).

The priors on $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$, the diagonal matrix with $\sigma^2\sigma_{\boldsymbol{\beta}}^2$ on the diagonals, control the shrinkage of $\boldsymbol{\beta}_S^{(h)}$. Variable selection is sensitive to the choice of prior on this parame-

ter, so BSR implements an empirical Bayes strategy to obtain a prior distribution for σ^2_β based on the data. While this is not a fully Bayesian approach, it has been shown that this strategy works better in practice (Antonelli et al., 2020). Additionally, the default prior for σ^2 is assumed to follow a Gamma$(0.001, 0.001)$ distribution.

However, when there is a weak relationship between the exposures and relationship, the estimated prior variance for the slab σ^2_β can be very small. In this case, the shape of the slab approximates the point mass of 0 at the spike, and the PIPs become difficult to accurately estimate. BSR avoids this by imposing a lower bound on the variance. This is determined by establishing a constant value for τ_h, the prior probability of inclusion, for all h and then permuting the rows of Y (i.e., breaking up the relationship). Then, a grid of values for σ^2_β are tested until some predefined threshold of the posterior probability of inclusion is obtained (e.g., 0.25 for a main effect and 0.05 for a two-way interaction). If the empirical Bayes estimate for σ^2_β is less than this lower bound, then the lower bound is used instead.

Finally, BSR assumes that $\tau_h \sim \text{Beta}(L, \gamma)$, which defines the prior probability of including a predictor for all functions h. If L is some predefined constant, and $\gamma = m$, the number of predictors, then the prior amount of sparsity should increase as the number of predictors increases (Antonelli et al., 2020).

3.3.6 The MCMC algorithm

We also briefly discuss the MCMC algorithm employed by BSR (Antonelli et al., 2020).

BSR uses an MCMC algorithm to obtain posterior distributions of σ^2 and τ_h. In particular, Gibbs samplers are employed to sample σ^2 and τ_h from their full distributions and to update $\boldsymbol{\zeta}$ and $\boldsymbol{\beta}^{(h)}_S$. Every T MCMC iterations, BSR uses a Monte Carlo expectation maximization algorithm with a Gibbs sampler to update σ^2_β. The

empirical Bayes estimate is obtained once σ^2_β converges, at which point the MCMC runs conditional on this estimated variance.

Notably, this algorithm must deal with the explicit specification of interaction terms in the model. Any additive univariate effect or lower-order interaction term is, by definition, a subset of some higher-order interaction term. As the MCMC algorithm searches the model space, it might accept a move to a higher-order interaction and get stuck in a local mode when, in reality, a simpler model should be preferred. BSR handles this challenge by imposing a constraint in the MCMC algorithm: if the inclusion of a pth order interaction term is being considered, then the algorithm must also evaluate all $(p-1)$th order models. If the truth is some lower-order model, then this strategy avoids the undesirable convergence to a local mode. When the model is complex, maintaining reversibility of updates under this strategy can be computationally challenging with a Gibbs sampler; in this case, using a Metropolis Hastings sampler is computationally faster (Antonelli et al., 2020).

3.4 Detecting interactions

In Chapter 3.1, we highlighted the challenges of analytically testing for the presence of interactions in exposure mixture studies. These challenges motivated a theoretical exploration of BKMR and BSR in Chapters 3.2 and 3.3. Now, we discuss and compare the options that BKMR and BSR provide for inference on the presence of interactions. We also include discussion on theoretical advantages and disadvantages to each. Importantly, we note that it is not possible to detect interactions between a covariate and an exposure within the model specifications of BKMR or BSR.

3.4.1 BKMR

Since the flexible h function in kernel machine regression allows us to forgo any assumptions about the nature of the relationship between the health outome and exposures, BKMR can potentially capture complex interactions between exposures The challenge with using BKMR to do this, however, is that there is no formal framework for conducting inference on the presence of interactions.

Currently, the most common approach for detecting interactions is through a qualitative assessment of visual diagnostic plots (Bobb, 2017a). Two- or three-way interactions can be assessed by plotting the estimated exposure-response relation for one/two exposures at various quantiles of another exposure, while setting all other exposures at fixed quantile values. For instance, if we are interested in the interaction between X_1 and X_2, we can plot the estimated regression line against X_1 at the 0.1, 0.5, and 0.9 quantiles of X_2 and vice versa. In the three-way case of X_1, X_2, and X_3, we can plot the estimated regression surface against X_1 and X_2 at the 0.1, 0.5, and 0.9 quantiles of X_3. If the shape of the estimation changes meaningfully, then there might be evidence of an interaction.

A slightly more formal inferential approach for two-way interactions involves using summary statistics (Bobb, 2017a, 2017b). In this case, we can calculate the difference in estimated response values for X_1 at two quantiles, say, 0.25 and 0.75, of X_2 and then generate a confidence interval. If we observe that the interval does not contain 0, then there is evidence of an interaction. The choice of quantiles here is important. If there is a parachute-like regression surface between the response and two exposures, the summary statistics might mask the true nature of the relationship.

Notably, if we specify hierarchical variable selection to handle multicollinearity, then only one exposure in each a priori defined group can enter the model. If there

exists some true interaction between exposures in one group, then BKMR will be unable to incorporate it into the final model. Moreover, interactions between exposures in separate groups can only be identified if both are selected into the final model based on their within-group PIPs. Hence, if detecting interactions is a goal when using BKMR with hierarchical variable selection, groups should be carefully selected, and the influence of group membership should be considered in model interpretation.

3.4.2 BSR

Providing formal inference on the presence of interactions was one of the primary motivations for the development of BSR. BSR explicitly incorporates interaction terms in its model formulation, and the model fitting process assigns PIPs for any m-dimensional interaction from the posterior means of the $\boldsymbol{\zeta}$ matrix. Such probabilities can be used as a quantifiable measure of the significance of a potential interaction. We can also compare PIPs for interactions with the PIPs for their individual components, which can be used to compare exposures' interactive effects with their marginal effects.

The visualization options available in BKMR are also possible in BSR. In particular, it is helpful to follow up the identification of a potential interaction with a visual assessment using the estimated exposure-response relationship at fixed quantiles of other predictors. The major benefit of BSR is that the PIPs serve as a quantifiable uncertainty metric for the presence of interactions.

3.4.3 Differences between BKMR and BSR

We have explained how the inferential framework of BSR improves upon the BKMR approach for the detection of complex interactions. We also briefly compare general features of their model formulations.

While BKMR is a fully nonparametric approach, BSR is a semiparametric approach because it makes distributional assumptions about the data (i.e., that the relationship can be adequately captured by a d-dimensional natural spline basis expansion). As BKMR uses the kernel technique, its implementation can become computationally intensive for datasets with large n, as it scales with n^2, while BSR is able to scale with n (Antonelli et al., 2020).

BSR is highly sensitive to the choice of d, the degrees of freedom. While it employs a WAIC approach to selecting the best d, this parameter introduces an additional tuning step in the analysis. Both approaches can be highly sensitive to the specification of certain priors. BSR offers an empirical Bayes strategy to estimate σ^2_β, the variance of a exposure's prior probability of inclusion. On the other hand, the choice of prior on the influential smoothing parameter, ρ, in BKMR is up to the user.

We also note that BKMR offers a hierarchical variable selection approach to dealing with collinearity between exposures. While this is an additional decision that must be specified by the user, it offers a formal approach for dealing with collinear exposures. BSR does not explicitly account for collinearity in its model formulation, which can lead to erroneous interpretations of variable importance if unaccounted for.

Chapter 4 Simulations

4.1 Past simulation studies

Here, we preface our simulation study with an overview of examples in the literature which compare various methods for exposure mixture studies using simulations. Taylor et al. (2016) conclude that, in general for exposure mixture studies, no single method consistently outperforms others across all situations and, importantly, that a method should be chosen based on the question of interest. Thus, for each study, we highlight not only the findings, but also the data-generating scenarios and the identified question of interest.

Lazarevic et al. (2020) compare the performance of a broad range of methods for accurate variable selection of important exposures. They simulated exposure data using a multivariate copula based on real-world data and the response by specifying a regression model with only a subset of truly significant exposures and a normal error term. Two correlation structures were considered — one with the original Spearman correlation matrix and one with the values halved — as well as two signal-to-noise ratios — one with an R^2 for the true model at 10% and one at 30%. They found that BKMR, along with three other flexible regression methods that allow for nonlinearity, provided more accurate variable selection results compared to two machine learning methods. Moreover, they observed that, in general, low signal-to-noise ratios had a stronger impact on performance than did increasing multicollinearity.

Hoskovec et al. (2021) compare Bayesian methods, including BKMR, while considering 4 research questions: accurate estimation, selection of important exposures, exclusion of unimportant exposures, and identification of interactions. They use observed exposure and covariate data to simulate response data using regression relationships; they considered three exposure-response scenarios of varying complexity and included two-way multiplicative interaction terms. For each simulated dataset, they randomly assigned exposures to be active components of the mixture to incorporate variability in the data. Overall, they found that Bayesian methods outperformed traditional linear regressions, and that BKMR performed best when the exposure-response function takes on a complex form.

Most recently, Pesenti et al. (2023) compare BKMR, BSR, and the Bayesian Least Absolute Shrinkage and Selection Operator (LASSO) for variable selection. Data were generated using a multivariate normal with moderate and strong correlation structures specified manually by the researchers. They found that, in situations with additive and linear exposure-response relationships, Bayesian LASSO was appropriate. Across the other scenarios, BKMR generally performed best, while BSR selected exposures with high heterogeneity when the sample size was smaller due to the influence of the degrees of freedom, d, tuning parameter. Notably, multicollinearity did not generally lead to spurious variable selection.

Finally, we briefly comment on studies by Sun et al. (2013) and Barrera-Gómez et al. (2017), whose explicit goal is to compare methods for identifying interactions. Both studies generate exposure data using the correlation structure from an existing dataset; Sun et al. (2013) uses a multivariate lognormal, while Barrera-Gómez et al. (2017) uses a multivariate normal. Both only consider two-way, multiplicative interactions. While neither of these studies consider the more recently developed methods used in this thesis, they find that, in general, models that formally allow for

interaction effects perform better than models that only allow for univariate additive effects, when an interaction is present.

4.2 Methods

The goal of our simulation study is to provide guidance on the choice between BSR and BKMR for characterizing a diverse set of complex interactions between predictors. In particular, we aim to extend findings from previous simulation studies by considering a more comprehensive range of interaction types, including different effect sizes, non-multiplicative interactions, and three-way interactions. We also explore interactions between exposures and categorical covariates, a previously understudied form of interaction in exposure mixture studies. Hereafter, we refer to exposures and chemicals interchangably.

4.2.1 MADRES data

In order to make our simulations comparable to real-world exposure mixture studies, we based our simulation data on the Maternal And Developmental Risks from Environmental and Social Stressors (MADRES) pregnancy cohort. The MADRES cohort is a prospective pregnancy cohort of predominantly lower-income, Hispanic persons in Los Angeles, California, which began in 2015 (Bastain et al., 2019). Urine samples were collected by participants at their first visit, and questionnaires were administered during their first visit, with follow-ups at the first, second, and third trimesters. See Bastain et al. (2019) for further details on study design.

Howe et al. (2020) previously examined the effect of prenatal metal mixtures of birth weight (BW) for gestational age (GA) in this cohort. They used BKMR to identify associations between metal mixtures and BW for GA, as well as BSR to con-

duct inference on interactions between metals. Using BKMR, they found that, of the metals in the mixture, mercury and nickel were most strongly associated with BW for GA. Moreover, BKMR results suggested that a potential interaction between mercury and nickel exists; however, when run through BSR, the PIP for this interaction was extremely small, despite being the highest of all two-way interactions.

Data from the study by Howe et al. (2020) were obtained from publicly available data in the Human Health Exposure Resource (HHEAR) Data Repository, which has been approved under Icahn School of Medicine at Mount Sinai IRB Protocol #16-00947. The Digital Object Identifiers associated with the urinary trace element data and epidemiological data are 10.36043/1945_159 and 10.36043/1945_177, respectively. All analyses were conducted in R v4.3.2 (R Core Team, 2013).

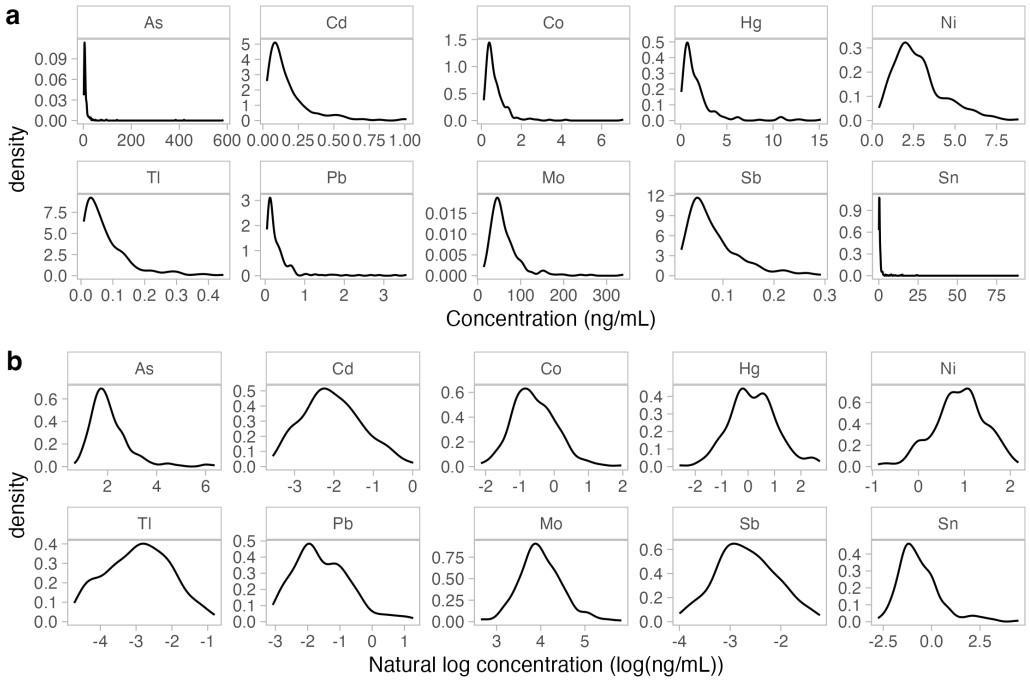

Figure 4.1: Distributions of original (a) and natural log transformed (b) concentrations of metals in MADRES cohort (n=252).

We followed the approach by Howe et al. (2020) for preparing the data for anal-

ysis. This resulted in retaining 10 metals in analysis: arsenic (As), cadmium (Cd), cobalt (Co), mercury (Hg), nickel (Ni), molybdenum (Mo), lead (Pb), antimony (Sb), tin (Sn), and thallium (Tl). Howe et al. (2020) used speciated As, but this was not available in HHEAR, so we used total As. Metals were expressed in nanograms per milliliter (ng/mL) and natural log transformed to reduce right-skewness (Figure 4.1). Among the full range of covariates considered by Howe et al. (2020), we used the subset of 4 that were available in HHEAR: any smoke exposure during pregnancy, maternal prepregnancy body mass index (BMI), maternal age during first trimester, and maternal race by ethnicity and birth place. We chose not to include study site, as there was a study site with only 1 participant. Race by ethnicity and birth place was collapsed into the following categories: non-Hispanic white, non-Hispanic black, non-Hispanic other, Hispanic born in the US, and Hispanic born outside the US. We observed 8 missing values for BMI in the data from HHEAR, which were not reported by Howe et al. (2020). We mean imputed these missing values. Distributions of covariates are shown in Figure 4.2. Our final analytic dataset included 252 participants, which was 10 fewer than in Howe et al. (2020), likely due to small discrepancies in their dataset and the one made available in HHEAR.

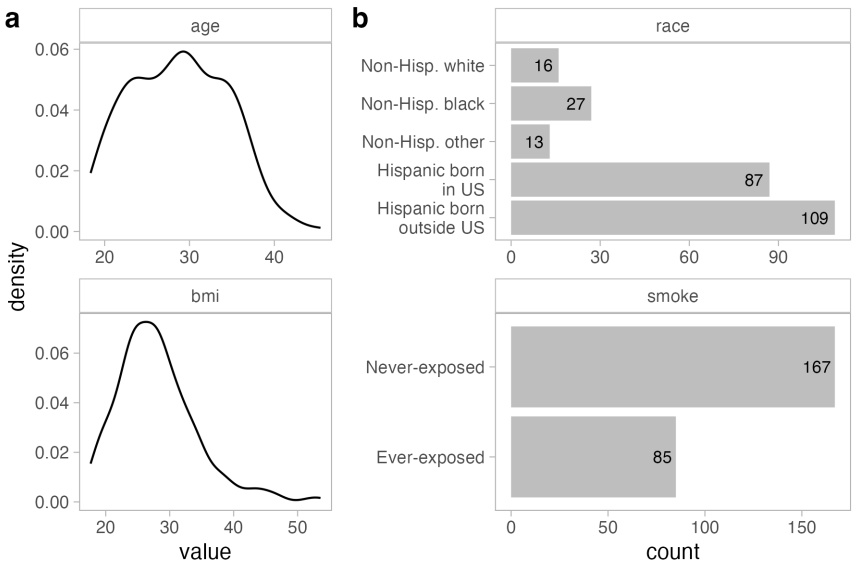

Figure 4.2: Distributions of continuous (a) and categorical (b) covariates in the MADRES cohort (n=252).

4.2.2 Using copulas to simulate predictor data

We simulated exposure and covariate data (hereafter referred to collectively as predictors) using a multivariate Gaussian copula fit on the 252 participants in the MADRES cohort. We used copulas as they can preserve both the correlation structure and marginal distributions from the observed data, allowing us to replicate conditions in a real-world scenario.

First, we briefly introduce copulas in the context of their use in this simulation, based on the presentation in Nelsen (2006). Copulas are joint cumulative distribution functions (CDFs) defined on the unit cube $[0, 1]^n$ that capture the dependence between n uniformly distributed marginals. Sklar's theorem allows us to apply copulas to our observed data. Sklar's theorem states that, if $H(x_1, \ldots x_n)$ is a joint CDF of the marginal CDFs $F_1(x_1), \ldots, F_n(x_n)$, then there exists a copula C such that, for all $(x_1, \ldots, x_n)$ in $(X_1, \ldots, X_n)$,

$$H(x_1, \ldots x_n) = C(F_1(x_1), \ldots, F_n(x_n)).$$

Note that, by the probability integral transform, or the universality of the uniform, the CDFs $F_1(x_1), \ldots, F_n(x_n)$ are distributed uniformly.

We used the `copula` package in R to fit copulas and generate random data (Hofert, Kojadinovic, Maechler, & Yan, 2023). We transformed the observed continuous predictor values to uniform distributions based on their empirical marginal CDFs, a process called generating "pseudo-random" samples. We used the checkerboard copula approach for generating pseudo-random samples for smoke exposure, a binary variable (Genest & Nešlehovà, 2007). We coded smoke exposure as 0's and 1's, generated a pseudo-random sample, and then "jittered" the values with uniform random noise. There is currently no widely accepted approach for generating pseudo-random samples from unordered categorical variables with more than two levels. Thus, we excluded race by ethnicity and birthplace from the copula model. While this means that our simulated datasets did not preserve any potential association between race and exposures, Figure 4.3 suggests that there is little to no visible association between race and exposures in the observed dataset.

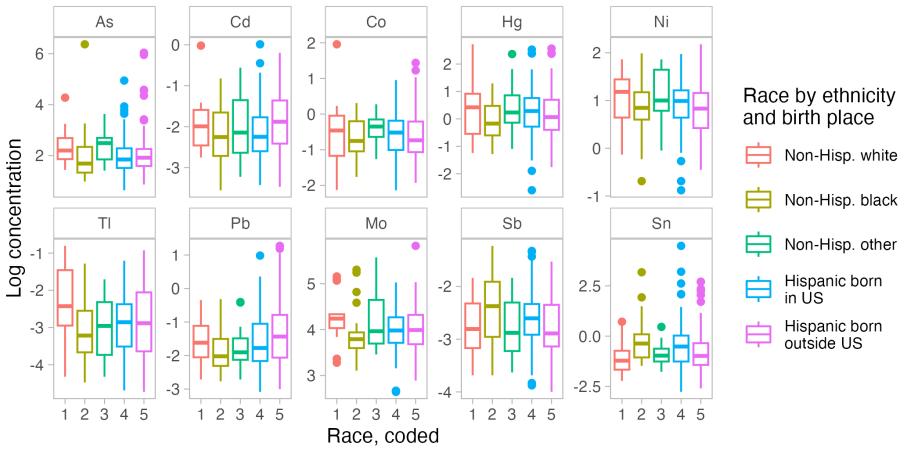

Figure 4.3: Association between race by ethnicity and birth place and metal exposures in the MADRES cohort (n=252).

Various families of copulas have been described, each of which specifies a different shape for the dependence structure. We performed model selection to identify the copula that best approximates the dependence structure of our data. We fit the set of multivariate copulas used by Lazarevic et al. (2020) in their simulation study, which included the Gaussian, t, Gumbel, Frank, Clayton, and Joe copulas. We fit two t copulas with 4 and 10 degrees of freedom, which controls dependence at the tails of the distributions, as well as a t copula where the degrees of freedom was determined during the fitting process. The Gumbel, Frank, Clayton, and Joe copulas require a θ parameter, which controls dependence between the distributions. We fit two versions of these copulas with $\theta = \{2, 4\}$. Among these, the Gaussian copula minimized the Akaike information criterion and maximized the likelihood, so we proceeded with this model. The Gaussian copula assumes a bivariate normal dependence structure between the marginal CDFs.

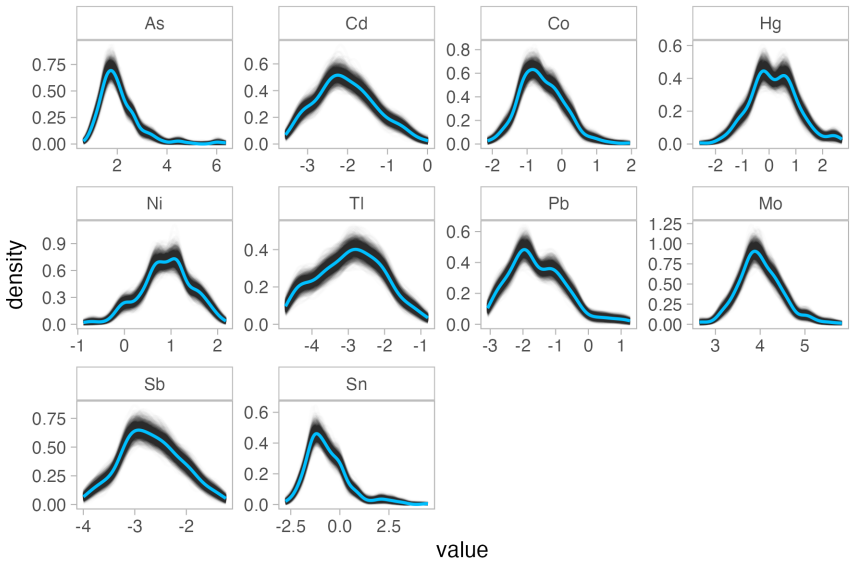

Figure 4.4: Distributions of log-transformed exposures from observed data (blue) and 2100 simulated smaller size (n=252) datasets (gray).

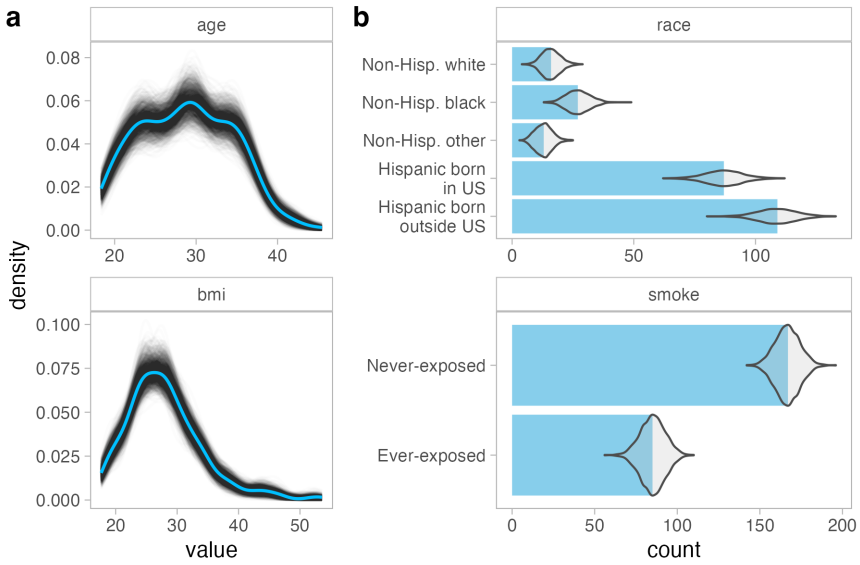

Figure 4.5: Distributions of continuous (a) and categorical (b) covariates from observed data (blue) and 2100 simulated smaller size (n=252) datasets (gray).

We simulated predictor data by randomly sampling from the fitted multivariate Gaussian copula distribution. All pseudo-random samples were then

back-transformed to their original distributions using empirical marginal CDFs. We simulated the race by ethnicity and birthplace variable by randomly assigning observations to each of the five categories based on proportions in the observed dataset.

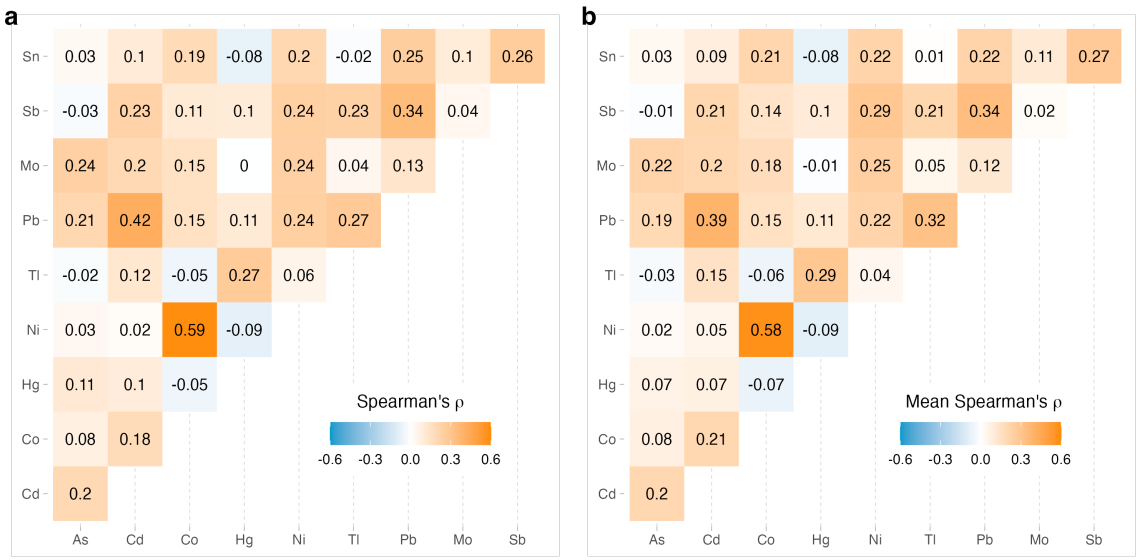

Figure 4.6: Spearman's correlation heat maps of exposures from observed data (a) and averaged across 2100 smaller size (n=252) simulated datasets (b).

We generated one set of simulated datasets with the same sample size as the observed dataset (n=252), which is typical in many cohort studies. We also generated another set of simulated datasets with a larger sample size (n=1000), which has become increasingly common with the rise of larger-scale studies. The goal of this choice was to inform sample size considerations in study design. We verified that the original structure of the observed dataset were preserved by visually comparing marginal distributions of exposures (Figure 4.4) and covariates (Figure 4.5), as well as the correlation structure using Spearman's ρ (Figure 4.6). Distributions of Spearman's correlation were approximately normal (Figure A.15). Plots for the larger size simulated datasets were similar (Figures A.16, A.17, and A.18).

4.2.3 Simulating predictor-response relationships

Health outcome responses were simulated under several different scenarios, each of which included different effect sizes and functional forms for the interactions. All scenarios were run for both the smaller (n=252) and larger (n=1000) sample sizes. In the first scenario, we specified a "base case" model:

$$Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + \frac{1.5}{1+\exp(-4\text{Sn})} - \text{Sb}^2 + 0.5\text{Sb} \\ + \text{age} + 0.5\text{bmi} + 0.5\text{race}_{\text{black}} + 0.5\text{race}_{\text{hisp.non}} + 1.5\text{smoke} + \varepsilon,$$

where $\varepsilon \overset{\text{iid}}{\sim} N(0, 5)$. This model includes a linear term for Hg, two S-shaped logistic terms for Ni and Sn with varying effect sizes, and a symmetric inverse U-shaped quadratic term for Sb (Figure A.1). Moreover, we included covariate terms as linear effects in the model. We chose the standard deviation on the normal random error term in order to achieve an R^2 of around 0.1-0.3 in a multiple linear regression that included only the true functional form of the significant chemicals (Figure A.19). This R^2 range approximates realistic signal-to-noise ratios in exposure mixture studies (Lazarevic et al., 2020).

In subsequent scenarios, we added an additional interaction term to the base case model. First, we considered interactions between two or more exposures. We defined four cases of interest: a two-way interaction between exposures that are univariately significant, a two-way interaction between exposures that are univariately insignificant, a two-way interaction between exposures that are moderately collinear, and a three-way interaction. For each case, we considered two functional forms — multiplicative and polynomial — and a lower and higher effect size, which we set by defining the weight on the interaction term in the model. The higher effect sizes were

Table 4.1: Specification of interaction terms in simulations.

	Effect size	
	Lower	Higher
Univariately significant		
Multiplicative	0.35Hg∗Ni	0.7Hg∗Ni
Polynomial	$0.13\text{Hg}*(\text{Ni}-1)^2$	$0.26\text{Hg}*(\text{Ni}-1)^2$
Univariately insignificant		
Multiplicative	0.35Cd∗As	0.7Cd∗As
Polynomial	$0.125\text{Cd}*(\text{As}-1)^2$	$0.25\text{Cd}*(\text{As}-1)^2$
Highly correlated		
Multiplicative	0.3Ni∗Co	0.6Ni∗Co
Polynomial	$0.1\text{Ni}*(\text{Co}-1)^2$	$0.2\text{Ni}*(\text{Co}-1)^2$
Three-way interaction		
Multiplicative	0.3Hg∗Ni∗Tl	0.6Hg∗Ni∗Tl
Polynomial	$0.09\text{Hg}*(\text{Ni}-1)^2*\text{Tl}$	$0.18\text{Hg}*(\text{Ni}-1)^2*\text{Tl}$

selected in order to achieve a power of approximately 0.5 at $\alpha = 0.05$ in the smaller sample size (n=252) case, using a multiple linear regression with the true functional form of the chemicals specified and the covariate terms included. The weights on the lower effect sizes were set equal to half of the higher effect sizes. Table 4.1 shows the specification of interaction terms. See Appendix A.1, Figures A.2-A.14, for 3D surfaces of the two-way interaction terms. Next, we considered interactions between the race by ethnicity and birthplace covariate (hereafter referred to as race for concision) and an exposure. We are interested in cases where the health effects of an exposure are higher in one group compared to the rest. In a real-world scenario, such interactions can arise from excess amounts of social stress experienced by a group due to racism. To model this, we increased the coefficient on Hg in non-Hispanic black individuals (n=27 in the original MADRES cohort) for the first scenario, and in Hispanic individuals born outside the US (n=109 in the original MADRES cohort) for the second scenario. The goal of this choice was to assess the impact of group size

on detectability of an interaction, and to quantify the potential value of oversampling the minority group. For each scenario, we specified a lower effect size by increasing the coefficient on Hg by 1.5× (i.e. from 1∗Hg to 1.5∗Hg) in the target group, and a higher effect size by increasing the coefficient on Hg by 2× (i.e. from 1∗Hg to 2∗Hg).

This resulted in a total of 42 scenarios ([1 base case + 5 interaction cases × 2 effect sizes × 2 functional forms] × 2 sample sizes = 42). For each scenario, we generated 100 simulated datasets to fit our models on, resulting in a total of 4200 datasets. Hereafter, we refer to the two different effect sizes of interactions using "lower" and "higher," and we refer to the two different sample sizes as "small" and "large," or "smaller" and "larger."

4.2.4 Models

We ran four methods on our simulated datasets. All metal concentrations and continuous covariates were standardized in analysis to keep values scale-free.

To obtain a baseline, we ran a multiple linear regression, including all exposures and covariates as linear, additive terms in the model. We refer to this model as the naive MLR. Then, we ran a multiple linear regression with the true model explicitly specified by excluding non-significant exposures and specifying the known form of non-linear terms and non-additive interactions. We refer to this model as the oracle MLR. In scenarios with an interaction between race and Hg, we collapsed race into a binary variable indicating whether or not the original race category was interacting with Hg. Then, we specified the interaction term in the oracle MLR using this binary variable, in order to simplify the detection of the interaction.

Next, we ran BKMR using the `bkmr` package in R (Bobb, 2022; Bobb et al., 2018). We chose to implement component-wise variable selection rather than hierarchical selection to make simulation results more interpretable, and because there was only

moderate multicollinearity in the observed and simulated data. We specified the default priors (Bobb et al., 2015, and as listed in Chapter 3.2.5), which is common in the literature for BKMR (Howe et al., 2020; e.g., Lazarevic, Barnett, Sly, & Knibbs, 2019; Pesenti et al., 2023). We ran the MCMC sampler for 50,000 iterations, as recommended by Bobb et al. (2018), and discarded the first 25,000 iterations for burn-in. BKMR does not provide the option to run multiple chains or to thin chains. For larger size datasets, we sped up computations by employing a Gaussian predictive process on 100 knots specified evenly across the predictor space.

We ran BSR using the `NLinteraction` package in R (Antonelli, 2018). We specified the default priors (Antonelli et al., 2020, and as listed in Chapter 3.3.5), which is common in the literature for BSR (e.g., Howe et al., 2020; Pesenti et al., 2023). Antonelli et al. (2020) suggests separately fitting models for degrees of freedom $d = \{1, 2, 3, 4\}$ and selecting the value for d which minimizes WAIC. We used the default Gibbs sampler, as opposed to the Metropolis Hasting sampler. Due to time constraints for this thesis, we first fit BSR on the grid of values for d using 5,000 MCMC iterations to obtain the empirical Bayes estimate for σ^2_β and then another 5,000 MCMC iterations to obtain the posterior distributions, discarding the first 2,500 iterations for burn-in each time. We selected d based on the WAIC criterion on these preliminary models. Then, we fit the full BSR model using 50,000 MCMC iterations to obtain the empirical Bayes estimate and then another 50,000 MCMC iterations to obtain the posterior distributions, discarding the first 25,000 iterations for burn-in each time. We ran two chains to verify convergence, thinning each chain by selecting every 8th iteration to reduce autocorrelation based on default settings. In a small test run on five smaller size datasets for each scenario containing interactions between exposures, as well as the base case, we found that using 5,000 iterations selected the same degrees of freedom as using 50,000 iterations 86% of the time (see

Appendix A.1, Figure A.20).

Finally, we ran stratified BKMR and BSR models in scenarios where we simulated an interaction between race and Hg. This involved running five separate models for each race category, each with the same settings specified above. For the smaller size datasets, we often observed convergence issues in BKMR within the smaller race categories. As such, for smaller size datasets, we also assess the impact of collapsing the three smaller race categories (Non-Hispanic white, black, and other) into one category before stratifying. This is a common practice in real-world studies where sample sizes for certain categories are low.

For all models, we saved run-times for the fitting process on the high-performance computing cluster. Due to an error in our code and time constraints for this thesis, we were not able to obtain run times for stratified BSR models. We checked convergence for a selection of BKMR and BSR models using trace plots (Appendix A.2, Figure A.21).

4.2.5 Model assessment

We assessed model performance based on detection of significant univariate chemicals as well as detection of interactions. For the naive and oracle MLRs, we considered a p-value less than 0.05 to indicate detection of a significant term. For BKMR and BSR, we used the median probability model, which considers a PIP greater than or equal to 0.5 to indicate detection of a significant term (Barbieri & Berger, 2004).

While BSR provides PIP's to quantify detection of interactions, BKMR does not. As such, for BKMR, we considered formal detection of an interaction based on confidence intervals constructed around the estimated response. Specifically, we first calculated the difference in estimated response at a chemical's 0.25 and 0.75 quantiles. Then, we assessed whether this quantity differed at the 0.25 and 0.75 quantiles of one

(or two, for three-way interactions) other chemical(s) in the interaction, while holding all other chemicals at their 0.5 quantiles, by constructing a 95% confidence interval of the difference in differences. We followed the code in the `SingVarIntSummaries()` function in the `bkmr` package for constructing confidence intervals (Bobb, 2022). Notably, for two-way interactions, the ordering of the chemicals in this comparison does not matter. To see why, see Equation (A.1) in Appendix A.1. However, for three-way interactions, the order does matter. Thus, we considered a three-way interaction to be detected if at least one of the three confidence intervals was significantly greater than zero. We used the Bonferroni correction to adjust for multiple comparisons, creating 98.3% confidence intervals to achieve an overall 95% confidence level (Dunn, 1961; VanderWeele & Mathur, 2019).

For both BKMR and BSR, we also visually assessed detection of interactions by plotting the estimated exposure-response surface for one chemical while fixing one (or two, for three-way interactions) other chemical(s) at their 0.1, 0.5, and 0.9 quantiles. In all scenarios, we calculated the sensitivity as the proportion of times a significant term was correctly detected. Due to time constraints for this thesis, we only calculated the false discovery rates, or the proportion of times a significant term is incorrectly detected, for two-way interactions between chemicals.

For stratified models, we compared the estimated response across each separate model. Specifically, for BKMR, we computed a confidence interval for the difference in estimated response at the 0.25 and 0.75 quantiles of Hg on each subcategory of race, following the code in the `SingVarRiskSummaries()` function in the `bkmr` package (Bobb, 2022). We adjusted for multiple comparisons based on the Bonferroni procedure by constructing 5 simultaneous 99% confidence intervals, in order to achieve an overall 95% confidence level (Dunn, 1961; VanderWeele & Mathur, 2019). We considered an interaction as correctly detected if (1) there was at least one non-overlap

between the target group's confidence interval and all other confidence intervals, and (2) all other confidence intervals overlapped. Due to time constraints for this thesis, we only considered sensitivity, as opposed to also considering false discovery rates.

While the kernel machine regression method used by BKMR generates a covariance matrix for estimates at various predictor values, which are then used to generate confidence intervals, the same is not true for the spline regression method used by BSR. We were not able to find a method in the literature for combining variances from estimated responses at two different sets of predictor values in the spline regression framework. Therefore, we visualized and compared the estimated exposure-response relationship in each of the stratified models as a qualitative way to assess for interactions. We also generated these diagnostic plots for BKMR.

See Appendix B.3.1 for the code we wrote to extract some of the aforementioned results from BKMR and BSR, as some functions were not directly available from the `bkmr` and `NLinteraction` packages (Antonelli, 2018; Bobb, 2022).

4.3 Results

4.3.1 Base case

We start by presenting results from models run on the base case scenario, in which the true relationship contained no interactions. Figure 4.7 displays the distribution of p-values and PIPs from this scenario, while Table 4.2 summarizes model sensitivity and false discovery rates based on these values. Note that insignificant chemicals were not included in the oracle MLR, which is why their distributions are omitted from this output.

Table 4.2: Sensitivity and false discovery rate (FDR) of chemicals in base case scenario.

	Sensitivity				FDR					
	Hg	Ni	Sb	Sn	As	Cd	Co	Mo	Pb	Tl
Naive MLR										
Small	0.80	0.70	0.12	0.24	0.05	0.07	0.02	0.07	0.03	0.08
Large	1.00	1.00	0.51	0.71	0.02	0.04	0.08	0.13	0.03	0.03
Oracle MLR										
Small	0.84	0.94	0.95	0.35	-	-	-	-	-	-
Large	1.00	1.00	1.00	0.89	-	-	-	-	-	-
BKMR										
Small	0.86	0.92	0.95	0.52	0.30	0.33	0.36	0.38	0.41	0.40
Large	1.00	1.00	1.00	0.77	0.12	0.12	0.14	0.13	0.08	0.13
BSR										
Small	0.51	0.67	0.88	0.17	0.06	0.04	0.04	0.04	0.04	0.05
Large	1.00	1.00	1.00	0.62	0.01	0.00	0.03	0.02	0.03	0.00

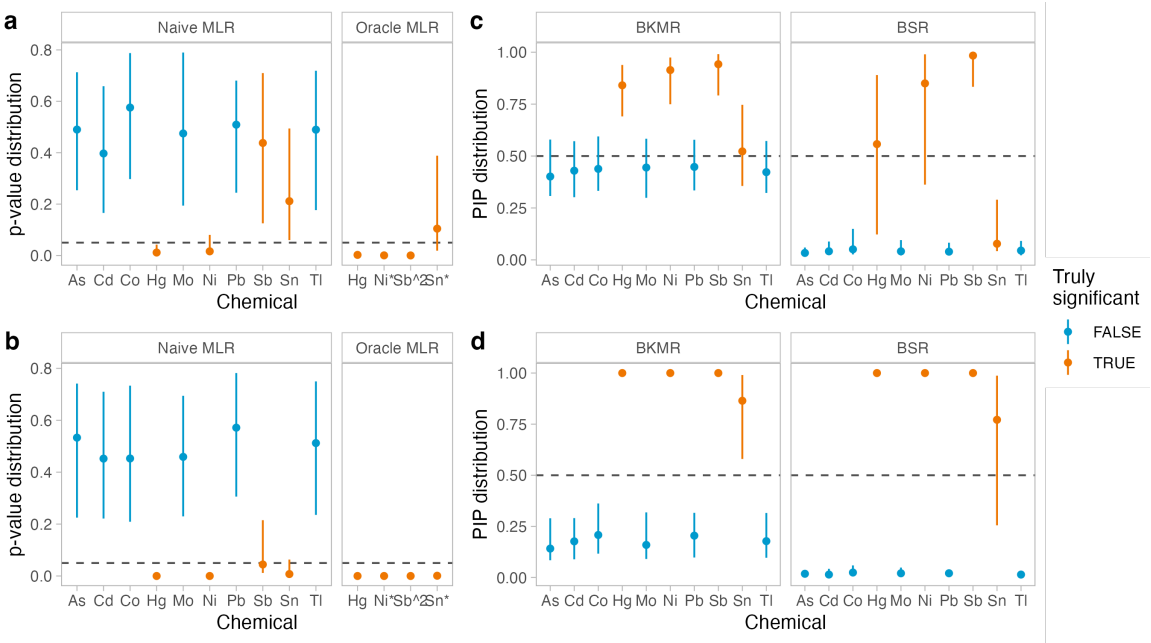

Figure 4.7: P-value distributions from smaller (a) and larger (b) size datasets and PIP distributions from smaller (c) and larger (d) size datasets.

The strongest detected effects in the naive MLR come from Hg, with a sensitivity

of 0.8 and 1 in the smaller and larger size datasets, respectively. This is likely because Hg is the only linear term in the model. Ni and Sn have an S-shaped curve with higher and lower effect sizes, respectively, so the naive MLR detects a slight linear signal from them. Sb, which has a U-shaped curve, is the hardest to pick up. On the other hand, the oracle MLR consistently detects Hg, Ni, and Sb. The smaller size oracle MLRs only occasionally pick up Sn, likely due to the lower effect size.

BKMR has similar sensitivity rates as the oracle MLR, ranging from 0.52-0.95 in the smaller size datasets and 0.77-1.00 in the larger size datasets. However, BKMR also has, by far, the highest false discovery rates, ranging from 0.30-0.41 in the smaller size datasets and 0.08-0.14 in the larger size datasets. This is likely due to the default choice of an inverse uniform distribution from 0 to 100 for the "slab" component on the "slab-and-spike" prior on r_m. Choosing a prior that assigns higher probability to smaller values of r_m should reduce the false discovery rate. In contrast, BSR tends to have slightly lower sensitivity rates than BKMR, ranging from 0.17 to 0.88 in the smaller size datasets and 0.62-1.00 in the larger size datasets, but the false discovery rates are much lower, ranging from 0.04-0.05 in the smaller size datasets and 0-0.03 in the larger size datasets.

Overall, this base case scenario confirms that the multiple regression and Bayesian models behave as expected. We also recognize that, for univariate significance metrics, BKMR tends to produce higher sensitivity and false discovery rates, while the opposite is true for BSR.

4.3.2 Univariate sensitivity

Now, we provide a brief overview of univariate sensitivity metrics from all scenarios with an interaction between chemicals. We are particularly interested in cases where it appears that the inclusion of an interaction term influences the detection rate of

Table 4.3: Overall sensitivity for univariate chemicals in all scenarios with interactions between chemicals. Multiplicative and polynomial are abbreviated mult. and poly., respectively.

		Small (n=252)				**Large (n=1000)**			
Type	Effect size	Naive	Oracle	BKMR	BSR	Naive	Oracle	BKMR	BSR
Hg-Ni									
Mult.	Lower	0.48	0.62	0.86	0.58	0.80	0.84	0.94	0.91
Mult.	Higher	0.50	0.70	0.82	0.68	0.80	0.95	0.91	0.92
Poly.	Lower	0.50	0.70	0.87	0.64	0.80	0.92	0.93	0.92
Poly.	Higher	0.46	0.62	0.80	0.57	0.80	0.88	0.92	0.93
Cd-As									
Mult.	Lower	0.46	0.78	0.81	0.56	0.78	0.94	0.94	0.92
Mult.	Higher	0.43	0.80	0.84	0.55	0.80	0.95	0.97	0.93
Poly.	Lower	0.46	0.78	0.78	0.56	0.82	0.94	0.96	0.92
Poly.	Higher	0.46	0.76	0.82	0.52	0.78	0.93	0.94	0.90
Ni-Co									
Mult.	Lower	0.49	0.65	0.85	0.60	0.80	0.84	0.95	0.92
Mult.	Higher	0.52	0.73	0.81	0.60	0.79	0.94	0.93	0.91
Poly.	Lower	0.52	0.67	0.85	0.64	0.80	0.89	0.93	0.89
Poly.	Higher	0.45	0.60	0.78	0.56	0.85	0.88	0.93	0.92
Hg-Ni-Tl									
Mult.	Lower	0.48	0.69	0.84	0.58	0.81	0.90	0.93	0.91
Mult.	Higher	0.44	0.73	0.80	0.56	0.82	0.94	0.95	0.92
Poly.	Lower	0.44	0.68	0.84	0.55	0.79	0.89	0.95	0.92
Poly.	Higher	0.48	0.73	0.80	0.62	0.77	0.94	0.92	0.92

univariate chemicals. Note that we use "univariate" to refer to the additive effects of a chemical.

Table 4.3 summarizes the sensitivity of univariate chemicals in all scenarios with interactions between chemicals, comparing the form of interactions, effect sizes, size of datasets, and models. See Table A.1 in Appendix A.2 for false discovery rates, as well as Figures A.22-A.29 in Appendix A.2 for the full p-value and PIP distributions for all scenarios with interactions between chemicals. In general, the detection rates of univariate chemicals that participate in an interaction differ in comparison to the

base case. First, we briefly note that the sensitivity for univariate chemicals that participate in interactions is reduced for oracle MLRs, compared to the base case (Table 4.3 and Figures A.24 and A.25). However, as the oracle MLRs contain explicit interaction terms, the p-values for univariate terms that participate in an interaction are not interpretable.

Now, for the naive MLR, BKMR, and BSR, chemicals that are univariately insignificant but participate in an interaction can occasionally be detected, depending on their form, effect size, and the sample size of the data. For instance, in scenarios with a higher effect polynomial interaction between As and Cd, both of which are univariately insignificant, the larger size naive MLR has a sensitivity of 0.67 for Cd (Figure A.23), BKMR 0.78, and BSR 0.55. In comparison, the sensitivities are all lower in scenarios with the higher effect multiplicative interaction between As and Cd. This is likely because, in the theoretical model for this scenario, Cd exhibits a strong positive relationship with the health response at lower levels of As, which is then attenuated at higher levels of As (Figure A.10). On the other hand, in the case of a multiplicative interaction, the positive relationship between Cd and the health response at lower levels of As becomes inverted at higher levels of As (Figure A.8), which would be more difficult to detect when only considering additive effects. This demonstrates that non-additive interactions may occasionally be detected through univariate terms depending on their form, and when the effect size and sample size are large enough.

Finally, univariate sensitivity for chemicals that are univariately significant — in our case, Hg and Ni — are not noticably affected by participation in an interaction. For instance, the naive MLR, BKMR, and BSR had sensitivities of 0.80, 0.86, and 0.51, respectively, for Hg in the base case with the smaller size dataset. In scenarios with a higher effect size multiplicative interaction between Hg and Ni, the sensitivities

were 0.77, 0.88, and 0.58 for the naive MLR, BKMR, and BSR, respectively. As such, we confirm that participation in an interaction does not affect univariate sensitivity, in our simulation scenarios.

The findings from this section suggest the impact of interpreting univariate effects from models without considering potential interactions. If the chemical is itself significant, the detection of its univariate effect should not be affected by participation in an interaction. On the other hand, if a chemical participates in an interaction but is, itself, not significant, then it can occasionally be detected as univariately significant, depending on the form of the interaction, its effect size, and the sample size.

4.3.3 Two-way interactions between chemicals

This section describes the ability of BKMR and BSR to detect two-way interactions between chemicals, using results from the oracle MLR as a baseline. We recall that sensitivities for BKMR were generated by the confidence interval approach, which required an arbitrary choice of the 0.75 and 0.25 quantiles for comparing the estimated response. On the other hand, sensitivities for BSR were based on the PIPs that it provides for interaction terms.

Table 4.4 summarizes the sensitivity of the oracle MLR, BKMR, and BSR for two-way interactions. The oracle MLR consistently has much greater sensitivity for interactions than both BKMR and BSR. First, we discuss smaller size datasets. At the lower effect size of interactions, the oracle MLR has sensitivities of 0.15-0.24, while BKMR and BSR have much lower sensitivities of 0-0.03, excluding BSR's sensitivity of 0.07 for the multiplicative interaction between Hg and Ni. Now, for the higher effect size interactions in the smaller size datasets, the oracle MLR has a sensitivity of around 0.5, which is as we designed. In comparison, for Hg and Ni, BKMR has sensitivities of only 0.11 and 0.03 for the multiplicative and polynomial interactions,

Table 4.4: Sensitivity to interactions in all scenarios with two-way interactions between exposures.

		Small (n=252)			**Large (n=1000)**		
Interaction type	Effect size	Oracle	BKMR	BSR	Oracle	BKMR	BSR
Hg-Ni							
Multiplicative	Lower	0.24	0.01	0.07	0.48	0.21	0.28
	Higher	0.50	0.11	0.10	1.00	0.88	0.77
Polynomial	Lower	0.14	0.01	0.03	0.54	0.18	0.27
	Higher	0.48	0.03	0.14	1.00	0.76	0.76
Cd-As							
Multiplicative	Lower	0.15	0.00	0.00	0.59	0.04	0.01
	Higher	0.52	0.01	0.02	0.99	0.57	0.39
Polynomial	Lower	0.18	0.00	0.00	0.57	0.00	0.00
	Higher	0.52	0.00	0.00	0.99	0.03	0.21
Ni-Co							
Multiplicative	Lower	0.18	0.01	0.01	0.52	0.03	0.00
	Higher	0.50	0.02	0.01	0.98	0.52	0.09
Polynomial	Lower	0.16	0.00	0.00	0.53	0.00	0.00
	Higher	0.54	0.00	0.02	0.97	0.05	0.05

respectively, while BSR has sensitivities of only 0.10 and 0.14. For interactions between Cd and As, and between Ni and Co, BKMR and BSR have sensitivities ranging from only 0-0.02. Therefore, in smaller size datasets, it is very difficult for BKMR and BSR to detect interactions, particularly when the effect size is low, or both chemicals are not univariately significant (which is the case for the pair Cd and As, and the pair Ni and Co). Now, in larger size datasets, interactions are detected much more frequently by BKMR and BSR. In general, for interactions between Hg and Ni, the sensitivities of BKMR and BSR have values around 0.2-0.3 less than those for the oracle MLR. As a result, for the higher effect size interactions, BKMR and BSR have relatively high sensitivites ranging from 0.76-0.88. While both do not achieve the sensitivity of 1 that the oracle MLR does, they can still detect these interactions

relatively reliably. Therefore, larger sample sizes can considerably increase sensitivity for interactions. We also note that neither BKMR nor BSR consistently outperforms the other in this case — BSR would thus be more preferable as it has a more formal framework for conducting inference.

In comparison, for interactions between Cd and As, which are both univariately insignificant, BKMR and BSR struggle to detect a significant effect even in the larger size datasets. At the lower effect size, sensitivities range from 0-0.04, and at the higher effect size, sensitivities range from 0.03-0.57. We note that cases with higher sensitivities for interactions (e.g., the higher effect size multiplicative interaction) are associated with higher univariate detection rates for Cd and As in both BKMR (Figure A.27) and BSR (Figure A.29). Moreover, BKMR and BSR's sensitivities for Ni and Co, which have moderate to strong multicollinearity, are similar to their sensitivities for Cd and As in the larger size dataset. Since Co is not univariately significant, it is difficult to determine whether low detection rates for this case are due to Co's univariate insignificance or its multicollinearity.

To investigate this further, Figure 4.8 visualizes the association between univariate significance and significance of an interaction. For both As and Cd, and Ni and Co, it appears that interactions are typically only detected by BKMR and BSR when their univariate effects are detected. On the other hand, for As and Cd, almost all the cases in which As and Cd's univariate effects are detected lead to detection of a significant interaction. However, for Ni and Co, only a little over half of cases in which Ni and Co's univariate effects are detected lead to detection of a significant interaction. We attribute this pattern to the effects of multicollinearity between Ni and Co.

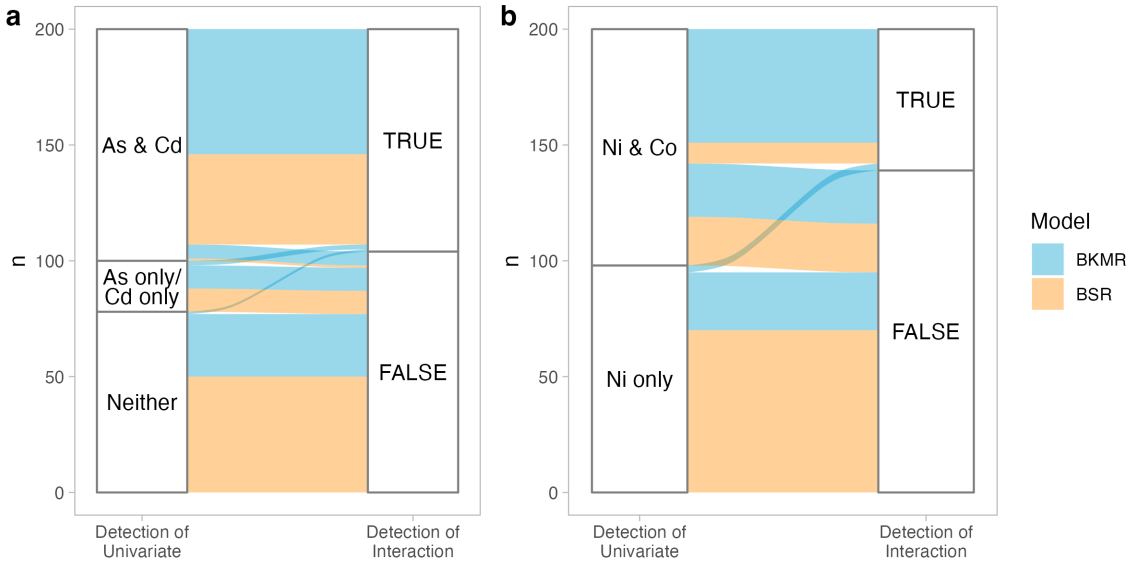

Figure 4.8: Flow chart showing relationship between detection of the univariate effects of As and Cd (a) and Ni and Co (b) with detection of their interactions. Only cases with a higher effect size, multiplicative interaction in a larger size dataset are included.

Thus, we suggest that detection of interactions is associated with detection of univariate effects. Interactions between Hg and Ni are easier to detect because they are, themselves, univariately significant. However, interactions between univariately insignificant chemicals are likely to only be detected if the form and effect size of the interaction are such that both chemicals can also be detected univariately. Moreover, it appears that multicollinearity between Ni and Co reduces the ability of BKMR and BSR to detect their interaction.

Next, Table 4.5 summarizes the false discovery rate of BKMR and BSR for two-way interactions. Both methods perform well in all scenarios, with false discovery rates below 0.02. This confirms that the above results for sensitivity did not occur by chance. For smaller size datasets, both BKMR and BSR have almost negligible false discovery rates, ranging from 0.0002-0.0025 for BKMR and 0.0009-0.0034 for BSR. In larger size datasets, false discovery rates increase slightly for BKMR, ranging

Table 4.5: False discovery rate of interactions in all scenarios with two-way interactions between exposures.

		Small (n=252)		**Large (n=1000)**	
Interaction type	Effect size	BKMR	BSR	BKMR	BSR
Hg-Ni					
Multiplicative	Lower	0.0011	0.0030	0.0034	0.0125
	Higher	0.0020	0.0020	0.0048	0.0159
Polynomial	Lower	0.0014	0.0014	0.0034	0.0148
	Higher	0.0009	0.0032	0.0064	0.0180
Cd-As					
Multiplicative	Lower	0.0011	0.0032	0.0039	0.0123
	Higher	0.0018	0.0020	0.0059	0.0150
Polynomial	Lower	0.0025	0.0016	0.0036	0.0136
	Higher	0.0002	0.0018	0.0034	0.0168
Ni-Co					
Multiplicative	Lower	0.0009	0.0023	0.0025	0.0120
	Higher	0.0011	0.0009	0.0025	0.0150
Polynomial	Lower	0.0020	0.0034	0.0036	0.0141
	Higher	0.0011	0.0030	0.0016	0.0173

from 0.0016-0.0064. On the other hand, false discovery rates increase notably for BSR in the larger size datasets, ranging from 0.012-0.018. Therefore, false discovery rates for interactions, while still low, should be taken into consideration for BSR with larger sample sizes. In exposure mixture studies, a visual assessment of the estimated exposure-response relationship is often used to characterize the form of a potential interaction, and a common choice for diagnostic plots involves estimating the exposure-response relationship for one chemical at various quantiles of the second (e.g., Howe et al., 2020; Valeri et al., 2017). Figure 4.9 depicts such plots for all scenarios with interactions between Hg and Ni.

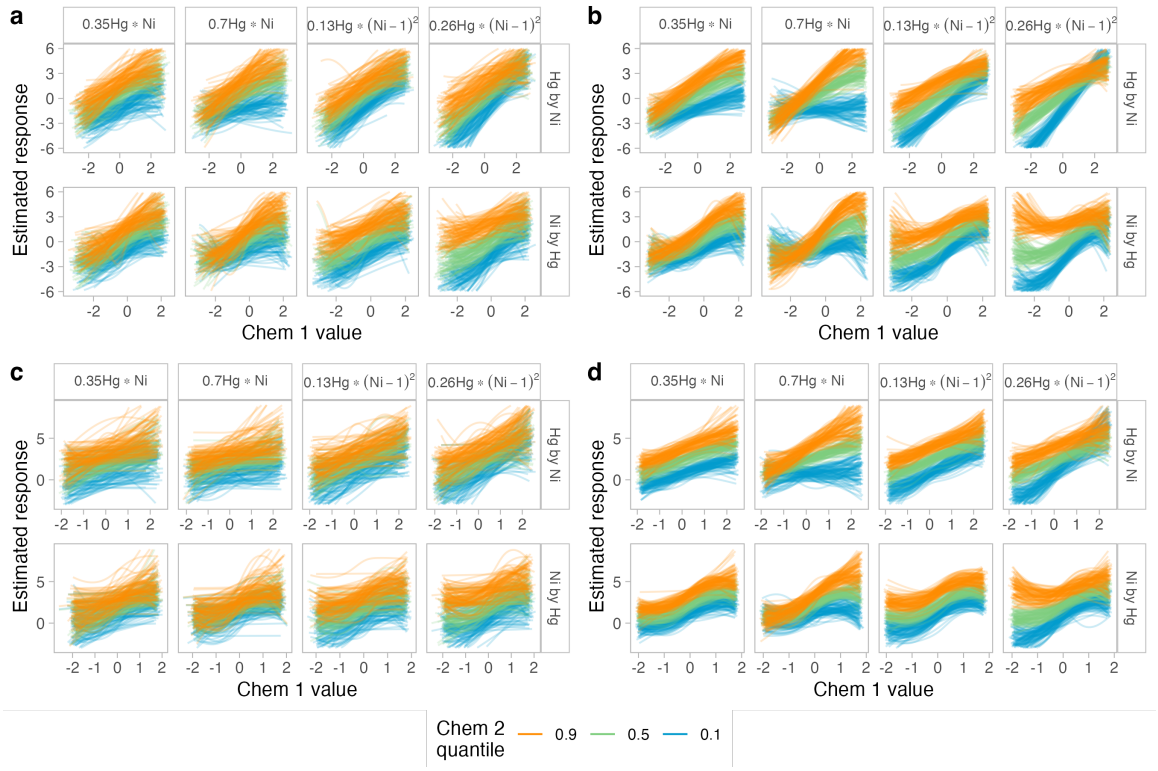

Figure 4.9: Exposure-response relationships estimated by BKMR in small (a) and large (b) datasets and BSR in small (c) and large (d) datasets, using the first chemical fixed at quantiles of another to assess interactions between Hg and Ni. All other chemicals are fixed at 0.5 quantiles.

The sensitivities in Table 4.4 are reflected by the patterns in the diagnostic plots. For instance, the second column of Figures 4.9b and 4.9d visualizes the higher effect size, multiplicative interaction between Hg and Ni estimated by BKMR and BSR, respectively. Both models have a sensitivity of 0.76 for this interaction, which is confirmed by the clear difference in the exposure-response relationship between various quantiles of the second chemical. Moreover, the exposure-response relationships are shaped similarly to their associated cross-sections in the 3D theoretical model displayed in Figure A.6.

Assessing both directions of the pairwise comparison in these plots (e.g., Hg at

quantiles of Ni, as well as Ni at quantiles of Hg) is important as, depending on the true form of the interaction, one direction may have more separation between the estimated relationship at various quantiles. For instance, in the aforementioned example with Hg and Ni, it appears that assessing the relationship between Hg and the health outcome at quantiles of Ni leads to clearer separation than the other way around.

Similar diagnostic plots for all other scenarios, which have similar patterns as those discussed here for Hg and Ni, are shown in Figures A.32-A.39 in Appendix A.2. In particular, we confirm that the low sensitivities observed in scenarios with interactions between Cd and As, as well as Ni and Co are reflected by estimated exposure-response relationships that do not appear to be associated with interactions. Moreover, it appears that, in Figures A.34 and A.38, the higher effect size, multiplicative interactions between Cd and As and Ni and Co are somewhat captured by BKMR, which reflects their associated sensitivities of 0.57 and 0.52, respectively. Overall, we confirm that patterns in visual diagnostic plots reflect metrics of importance for interactions in both BKMR and BSR.

4.3.4 Three-way interactions between chemicals

Next, we discuss the ability of BKMR and BSR to detect three-way interactions between chemicals. Table 4.6 summarizes the sensitivity of the oracle MLR, BKMR, and BSR for three-way interactions between Hg, Ni, and Tl. Visual plots of the estimated exposure response relationship for one chemical, while holding the two others in the interaction at their 0.1, 0.5, and 0.9 quantiles, are shown in Figures A.40-A.43. Due to time constraints, we did not obtain false discovery rates for these interactions. The oracle MLR sensitivities are similar to the two-way scenarios. As in the case of two-way interactions with univariately insignificant chemicals, BKMR

Table 4.6: Sensitivity to trivariate interactions between Hg, Ni, and Tl.

		Small (n=252)			**Large (n=1000)**		
Interaction type	Effect size	Oracle	BKMR	BSR	Oracle	BKMR	BSR
Multiplicative	Lower	0.19	0.00	0	0.45	0.01	0.00
	Higher	0.50	0.01	0	0.97	0.00	0.01
Polynomial	Lower	0.10	0.00	0	0.48	0.00	0.00
	Higher	0.49	0.00	0	0.96	0.05	0.00

and BSR are generally unable to detect these interactions, with sensitivities ranging from 0-0.05. While Hg and Ni are both univariately significant, Tl is not. Similar to the Ni-Co case, it is difficult to determine whether these patterns are due to inherent difficulties in detecting three-way interactions, or the univariate insignificance of Tl.

4.3.5 Interactions between race and an exposure

This section includes results for scenarios with an interaction between race, a categorical covariate, and Hg.

Table 4.7 summarizes the sensitivity of the oracle MLR and BKMR. Note that we were unable to provide formal metrics of importance for interactions from BSR. In total, BKMR failed to converge in either the non-Hispanic white or non-Hispanic other category in 311 out of 400 of the smaller size datasets, which are captured in the "Uncollapsed" column of this table. All BKMR models run on datasets with the three smallest race categories, collapsed into one group, converged. In general, interactions between a categorical variable and a continuous variable are harder to detect. Even in larger datasets where the effect of Hg was doubled in the largest race category (i.e., Hispanic born outside US), the oracle MLR had a sensitivity of 0.83, while BKMR had a much lower sensitivity of 0.21. This is the only scenario in which BKMR's sensitivity exceeded 0.03. Thus, BKMR is generally not able to detect these

Table 4.7: Sensitivity to interactions between the categorical race variable and Hg.

		Small (n=252)			**Large (n=1000)**	
		Uncollapsed		Collapsed*		
Interaction in	Effect size	Oracle	BKMR	BKMR	Oracle	BKMR
Original n=27†	Lower	0.07	0.00	0.00	0.21	0.01
	Higher	0.19	0.00	0.00	0.51	0.03
Original n=109‡	Lower	0.12	0.00	0.00	0.39	0.03
	Higher	0.24	0.02	0.03	0.83	0.21

* "Collapsed" refers to scenarios where the smallest three race categories are collapsed into one stratified model.
† Non-Hispanic black
‡ Hispanic born outside US

interactions with high sensitivity.

Similar to interactions between chemicals, we can create visual plots of the estimated relationship between Hg and the response within each stratified BKMR and BSR model. Figure 4.10 displays such plots; in each panel, the estimated relationship in the race category with the interaction is plotted on top, to distinguish it from the other categories.

In some cases, it appears that some degree of interaction is detected by BKMR and BSR. For instance, in the large dataset, when the effect of Hg is increased by 150% (the lower effect size) in the category Hispanic born outside US, there appears to be some separation in the estimated relationship in that category (shown in red), compared to the other four. However, BKMR's sensitivity in this scenario was only 0.03, indicating that it was under-powered to formally detect this interaction. Based on these plots, it also appears that BSR detects a significant effect from Hg less often than BKMR does, as most of the estimated exposure-response relationships are flatter. This is confirmed in Table A.2, which displays the sensitivity for Hg in each of the stratified BKMR and BSR model. Thus, BSR is likely more affected by

the smaller sample sizes from stratification than BKMR, at least with respect to its univariate metrics of variable importance.

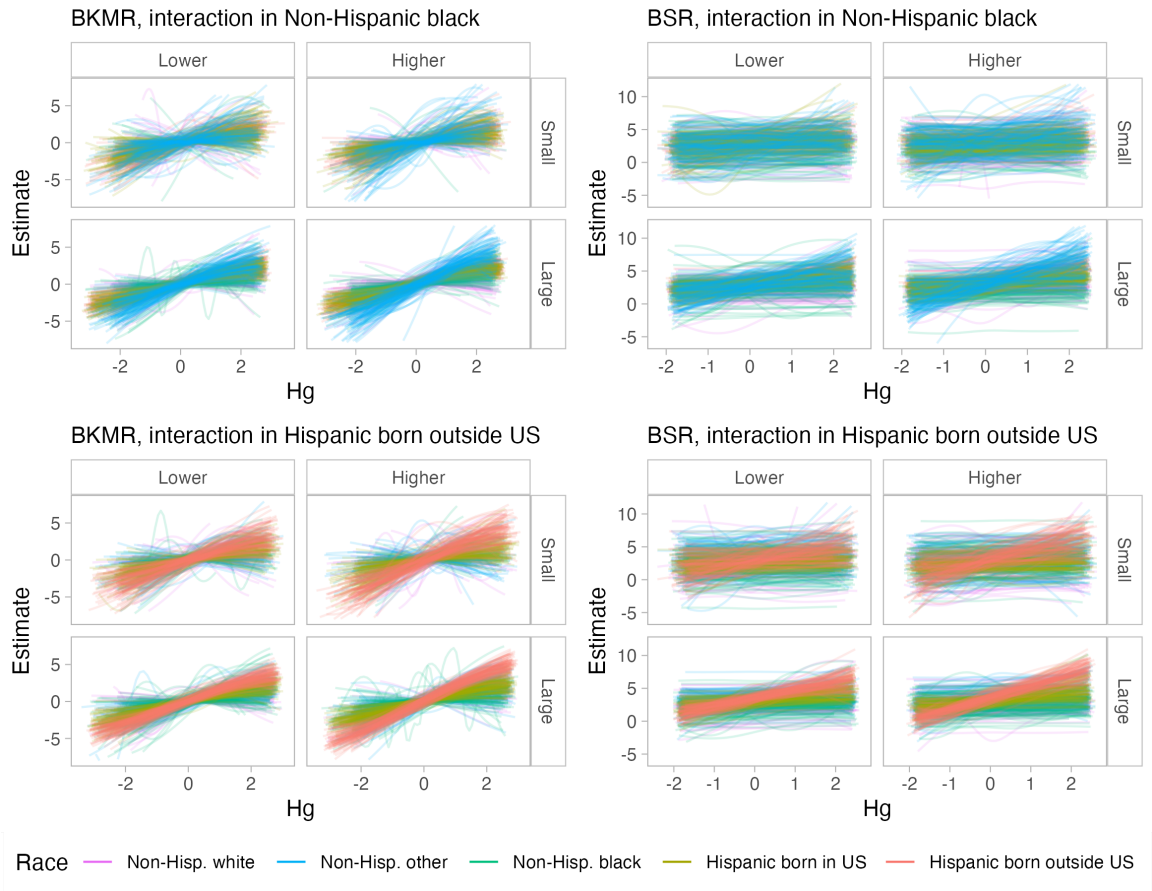

Figure 4.10: Relationship between Hg and response estimated by stratified BKMR and BSR models in smaller and larger datasets. All other chemicals are fixed at 0.5 quantiles. Lower and higher refer to effect sizes, while small and large refer to dataset sizes.

Therefore, while stratified BKMR and BSR models are, in theory, capable of detecting interactions between a categorical covariate and an exposure, it is difficult to use them for this purpose in practice. The flexibility of BKMR and BSR means that it is not possible to directly conduct inference on the effect sizes of specific chemicals. For BKMR, we are able to construct confidence intervals around the difference of the estimated response at two quantiles of a chemical, and to then compare these

confidence intervals between race categories. However, this is a more informal method of inference, as the choice of quantiles is arbitrary. Moreover, for both BKMR and BSR, it is likely that the reduced sample sizes in each stratified model significantly under-power the model for accurately estimating the exposure-response relationship.

Thus, this section highlights the difficulty of using current flexible Bayesian regression methods for detecting interactions between a categorical covariate and a chemical in exposure mixture studies.

4.3.6 Run-time analysis

Finally, we present the run-times for fitting models on the high-performance computing cluster.

Table 4.8 summarizes the average run-time for each model and sample size, under different interaction scenarios (i.e., various effect sizes and forms of interactions). For model fitting, we used 50,000 MCMC iterations. "BSR df" refers to the process for determining the optimal degrees of freedom for the spline regression, while "BSR mod" refers to model fitting process. We determined the degrees of freedom in BSR by fitting a grid of four values, each with 1/10 of the MCMC iterations we used for the full model. However, in a full analysis, one should determine the degrees of freedom using the full set of MCMC iterations, so the full run-time for BSR would be approximately four times that reported in the "BSR mod" row. We also recall that BKMR was fit using a Gaussian predictive process with 100 knots in the larger size datasets to reduce computational time.

The naive and oracle MLRs take a fraction of a second to run, in both the smaller and larger size datasets, while run-times range from around 15 minutes for BKMR in the smaller size datasets to around 4 hours for BSR (which includes the degrees of freedom selection process) in the larger size datasets. While both Bayesian models

Table 4.8: Average run-times in all scenarios with interactions between two or more chemicals, as well as the base case.

			Multiplicative		**Polynomial**	
Model	Sample size	Base	Lower	Higher	Lower	Higher
Naive	Small	0.0032 s	0.0016 s	0.002 s	0.0016 s	0.0017 s
	Large	0.002 s	0.002 s	0.0035 s	0.0031 s	0.0025 s
Oracle	Small	0.0015 s	0.0015 s	0.0015 s	0.0019 s	0.0016 s
	Large	0.0044 s	0.0018 s	0.0018 s	0.0019 s	0.0024 s
BKMR	Small	13.56 m	15.21 m	15.51 m	15.20 m	15.17 m
	Large	1.64 h	1.68 h	1.62 h	1.64 h	1.59 h
BSR df	Small	30.11 m	33.15 m	32.92 m	33.81 m	35.11 m
	Large	57.22 m	58.02 m	1.06 h	57.75 m	1.05 h
BSR mod	Small	1.37 h	1.36 h	1.42 h	1.44 h	1.49 h
	Large	2.92 h	2.81 h	3.06 h	2.87 h	3.04 h

are slow, BKMR is still faster than BSR, even when BSR's lengthy degrees of freedom selection process is not factored in. In the smaller size datasets, BKMR takes about 1/6 of the time BSR does, and in the larger size datasets, BKMR takes about 1/2 of the time BSR does. BKMR's speed in comparison to BSR is more pronounced in the smaller size datasets, as BKMR scales with n^2, while BSR scales with n. The Gaussian predictive process likely improves BKMR's speed in the larger size datasets by a wide margin. Moreover, BSR's time may also be attributed to the empirical Baye's strategy for estimating the influential σ_β^2 parameter, which requires a second set of 50,000 MCMC iterations to run. We also note that BSR appears to take slightly longer in cases with an interaction of higher effect size, compared to the lower effect size, while BKMR's run-time does not appear to be influenced by effect size. Overall, BKMR may be a slightly better choice in settings with smaller datasets, if computational burden is a limiting factor. Table 4.9 summarizes the average run-times of the naive MLR, oracle MLR, and stratified BKMR in scenarios with an

Table 4.9: Average run-times in scenarios with an interaction between the categorical race covariate and Hg.

Model	Race	Small (n=250)	Large (n=1000)
Naive	-	0.0021 s	0.0035 s
Oracle	-	0.0019 s	0.0022 s
BKMR	Non-Hispanic white	2.50 m	3.20 m
	Non-Hispanic black	4.01 m	4.24 m
	Non-Hispanic other	1.42 m	3.34 m
	Hispanic born in US	5.37 m	29.24 m
	Hispanic born outside US	5.65 m	52.20 m
	Collapsed non-Hispanic	4.17 m	-

Note:
Original sample sizes of race category: Non-Hispanic white (n=16), Non-Hispanic black (n=27), Non-Hispanic other (n=13), Hispanic born in US (n=87), Hispanic born outside US (n=109), and Collapsed non-Hispanic (n=56=16+27+13).

interaction between race and Hg. Again, we note that we were unable to get run-times for BSR within the timeframe of this thesis. However, based on observations of stratified BSR in the high-performance computing cluster, we estimate that it took around 20 hours for all stratified models on one smaller size dataset to run, and 15 hours for one larger size dataset. We suspect that the longer run-time required for smaller size datasets is a result of the MCMC algorithm. We used a Gibbs sampler to fit BSR — in cases where the model is complex, maintaining reversibility of updates can be computationally challenging (Antonelli et al., 2020). As in the previous case, the MLRs took a fraction of a second to run. The average run-time for BKMR on the smaller size dataset, when added up across all stratified models, is just slightly higher than for the unstratified models. The average run-time for BKMR on the larger size dataset, again added up across all stratified models, is slightly lower than the for the unstratified models. This demonstrates the dependency of BKMR's run-time on n^2, as we did not use a Gaussian predictive process to speed up the stratified

models. Finally, based on rough observations of BSR, it was clear that the stratified BSR models took much longer to run than the unstratified BSR models, likely due to challenges with the MCMC algorithm in the reduced sample size. This is a major limitation for using stratified BSR models to detect interactions between a categorical covariate and an exposure.

4.4 Discussion

Our simulation study provides guidance on the detection and characterization of a wide range of interactions using flexible Bayesian regression models.

First, the base case scenario, with no interactions, demonstrates that BKMR and BSR behave as expected. We confirm the results of previous studies, which find that BKMR is able to flexibly capture complex additive exposure-response relationships (i.e., no interactions), though false discovery rates can be higher than other methods (Hoskovec et al., 2021; Lazarevic et al., 2020). Similar to Pesenti et al. (2023), we also find that BKMR generally has higher sensitivity than BSR, especially in smaller size datasets.

Next, in cases with interactions between chemicals, we also assessed the univariate signals of chemicals that participated in an interaction. We found that univariately significant chemicals that participated in interactions could occasionally be detected through their univariate effects, depending on the true form of the interaction and if its effect size was large enough. Moreover, the univariate effects of univariately significant chemicals were not detected less often when they participated in an interaction. Together, these results suggest that, at least in the scenarios explored in this simulation, failing to assess for an interaction that would otherwise have been detected should not affect univariate signals, and interactions may sometimes be detected as

univariate effects.

Overall, we found that BKMR and BSR did not detect interactions between chemicals as well as the oracle MLR. This suggests that the flexibility afforded by BKMR and BSR leads to drawbacks in sensitivity compared to an ideal (and unrealistic) case, where we know the true form of the interaction. Still, in smaller size datasets with an interaction between chemicals that were themselves univariately significant, BKMR and BSR occasionally detected the signal. In larger size datasets, BKMR and BSR produced sensitivities within 0.2-0.3 of the sensitivities produced by the oracle MLR; and, sensitivities for the larger effect sizes of interactions ranged from 0.76 to 0.88, which is relatively reliable. These findings show that increasing sample size can considerably increase sensitivity for interactions.

However, in cases where at least one of the chemicals in the interaction was not univariately significant, BKMR and BSR generally only detected an interaction if there were also significant univariate effects for each chemical. It is typical practice in exposure mixture studies to only search for interactions between chemicals that have significant univariate effects. Thus, we confirm that, for BKMR and BSR, this analytic process should capture most interactions, as interactions between univariately insignificant chemicals were rare in our simulations. Moreover, we suggest that multicolllinearity between chemicals reduced sensitivity for interaction. Though, the effects of multicollinearity were difficult to ascertain, as one of the chemicals was univariately insignificant in our simulations. Finally, sensitivity for three-way interactions was very low, though that could also be due to the inclusion of a univariately insignificant chemical in our simulated interactions, rather than inherent challenges with three-way interactions. Additional simulations with multicollinearity and three-way interactions in different scenarios are necessary to verify our results.

Our simulation study extends the findings from Pesenti et al. (2023) by providing

more comprehensive guidance for the use of BKMR and BSR for the detection of interactions. Moreover, to our knowledge, this is the first simulation study of interactions in exposure mixtures since the study by Barrera-Gómez et al. (2017). While the methods they considered were only capable of detecting interactions when assuming linearity, the flexible Bayesian regression methods that we considered in our study can incorporate non-linear effects, and can thus be used in a wider range of scenarios.

To our knowledge, this is also the first study to consider interactions between a categorical, sociodemographic covariate and a chemical in an exposure mixture study. Overall, we found that it is very difficult to detect such interactions. We were not able to conduct inference on the presence of these interactions using BSR, as we could not find guidance in the literature for constructing confidence intervals on contrasts between values of a continuous predictor in spline regression. For BKMR, interactions were only detected when the overall sample size was 1000 and when the effect of exposure differed in the largest racial/ethnic category. Thus, the detection of such interactions likely requires large sample sizes and sufficient sampling of the group in which the effect of exposure differs compared to the others. This is evidence for oversampling the minority group, as it is often this group that experiences excess effects of exposure due to discriminative social and structural conditions. Lastly, it appeared that the confidence interval approach that we used was under-powered, as there were cases in which there was visual evidence of an interaction in the estimated exposure-response relationship, but BKMR did not produce significant results.

We also briefly comment on run-times. As both BKMR and BSR are Bayesian models that use MCMC algorithms, it is computationally expensive to arrive at a solution. Moreover, BKMR and BSR require prior specifications that can affect the model fit. It would be time-intensive to fit a grid of prior settings in order to assess the impact of, say, the prior on the influential r parameter in BKMR in order to

check the robustness of PIP values. Running BSR can be especially time intensive on stratified models, as it can become more difficult for the MCMC algorithm to converge in a smaller sample size. Thus, time cost can be a major limitation for model implementation, and it should be taken into consideration particularly when datasets are large, or access to high-performance computing equipment is limited.

Conclusion

We will continue to encounter mixtures of novel chemical entities in our day-to-day lives. Statistical studies of the health risks associated with exposure to chemical pollutants are critical for making modern toxic landscapes more habitable. Historically, the focus of risk assessments has been to identify the toxicity of single chemicals. However, as discussed in Chapter 2, such studies stem from habits of thought which seek to associate outcomes with isolated causes. An approach that considers the joint effects of multiple chemical pollutants — exposure mixtures — is more representative of reality, and also more likely to be protective of human health. However, exposure mixture studies come with a host of statistical challenges. This thesis focuses on the task of detecting non-additive interactions between exposures and sociodemographic covariates.

Beyond posing an interesting statistical estimation challenge, interactions are also important for accurately estimating health risk from exposure mixtures and informing policymakers about the potential benefits of environmental regulations. In Chapter 3, we introduce two flexible Bayesian regression models, BKMR and BSR, which both have the capacity to detect interactions. We discuss how the two models take different approaches to estimation. BKMR employs kernel machine regression, a nonparametric modeling technique that can flexibly capture complex exposure-response relationships and, as such, allows for non-additive interactions between exposures. BSR employs spline regression, a semiparametric modeling technique, in a model

formulation that explicitly incorporates interaction terms. This parameterization of interaction terms makes BSR's inferential approach for interactions more formal than BKMR's.

Our simulation study in Chapter 4 compares the ability of BKMR and BSR to detect interactions between chemicals. Our results emphasize the challenges of detecting interactions. In general, BKMR and BSR had lower sensitivites compared to a multiple linear regression with the true form of interactions specified. Still, we were able to achieve high sensitivities using BKMR and BSR to detect interactions between univariately significant chemicals. Sensitivity for univariately insignificant chemicals was much lower, and these interactions were only detected when univariate signals were also detected. Multicollinearity between interacting chemicals likely reduces sensitivity, and three-way interactions are generally very difficult to detect. Increasing the sample size from around 250 to 1000 in our simulations considerably increased sensitivity for interactions.

We conclude that BKMR and BSR are generally comparable for detecting interactions between chemicals, as neither consistently outperforms the other. For future studies of interactions in exposure mixture studies using Bayesian models, we recommend increasing the target sample size in study design, in order to increase power for detecting interactions. For data analysis, we recommend using BKMR first to identify both univariate effects and potential interactions with visual diagnostic plots. BKMR is generally more sensitive to univariate signals, more reliable in smaller sample sizes, and also computationally less intensive. We recommend using BSR to verify these results, as it tends to have lower false discovery rates for univariate effects and also provides a more formal framework of inference for any potential interactions. As most interactions were only detected when the chemicals were also univariately significant, limiting the search space to those chemicals with univariate PIPs greater than 0.5

should generally capture all significant interactions.

We also assessed interactions between chemicals and a categorical race covariate. Testing for such interactions can identify cases where certain groups may experience excess harmful effects of exposure due to discriminative social and structural conditions. However, we found that it was generally very difficult to detect such interactions using stratified BKMR and BSR models. Even in large sample sizes, and when the differential effect of exposure was in the largest race category, sensitivity was still very low. As such, we highlight that current Bayesian methods are under-powered for detecting potential interactions between categorical covariates and an exposure.

There are multiple areas for future simulation studies that arise from our findings. It would be beneficial to complete the full analyses for scenarios considered in our simulation by obtaining false discovery rates for three-way interactions and interactions between race and Hg, as well as recording run-times for the stratified BSR models. Considering interactions between collinear chemicals that are, themselves, univariately significant, could strengthen our finding that multicollinearity reduces sensitivity for interactions. This is also the case for three-way interactions. Identifying a method for constructing confidence intervals on the estimated magnitude of response for BSR would allow us to conduct inference on stratified BSR models for interactions between a categorical covariate and an exposure. Finally, our simulated datasets included 10 chemicals and 4 covariates, but exposure mixture studies can often include many more variables. Future simulations could consider how sensitivity for interactions changes in higher-dimensional data.

Overall, this thesis summarizes the current capabilities of modern Bayesian statistical methods for characterizing complex interactions in exposure mixture studies. The flexibility of recently developed techniques enables models to successfully identify interactions in noisy and high-dimensional data. However, we also highlight areas

where current methods fall short, particularly for interactions between covariates and exposure. Thus, it is important that we continue to develop methods that account for the unique statistical challenges of exposure mixtures, in order to progress toward a study of exposure that better considers how bodies are related to the environment and to each other.

Appendix A Supplemental output

This first appendix includes supplemental output for Chapter 4.

A.1 Methods

Figure A.1 depicts the exposure-response relationship for univariate exposures included in all models in simulation.

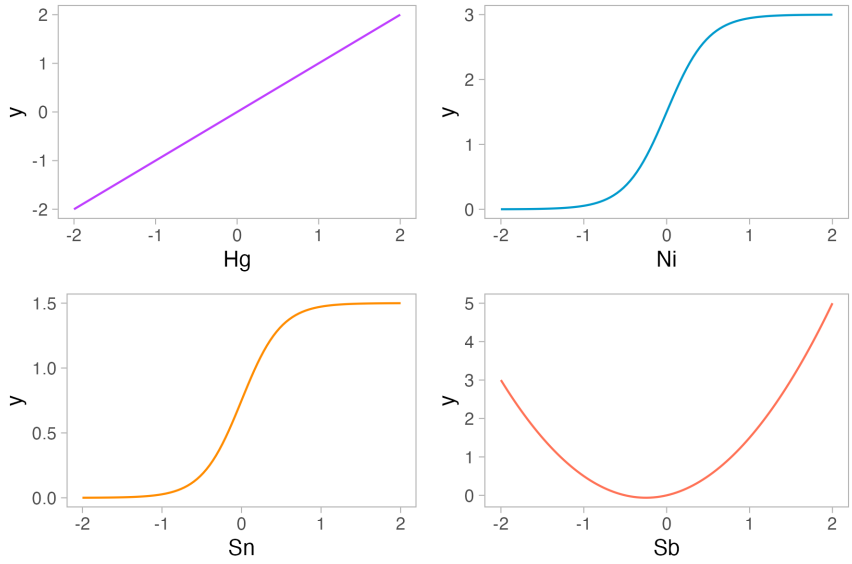

Figure A.1: Exposure-response relationship for univariate exposures in all models. Exposure values are log-scaled and then standardized.

Figures A.2-A.14 depict the exposure-response relationship for all two-way interactions specified in the simulation study.

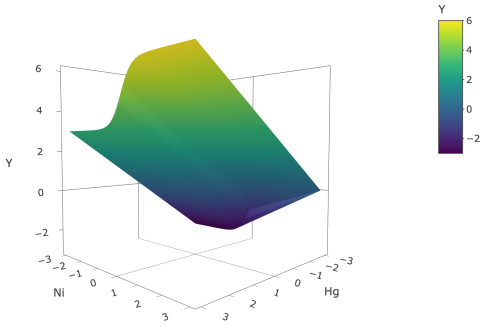

Figure A.2: Exposure-response surface for base case: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})}$.

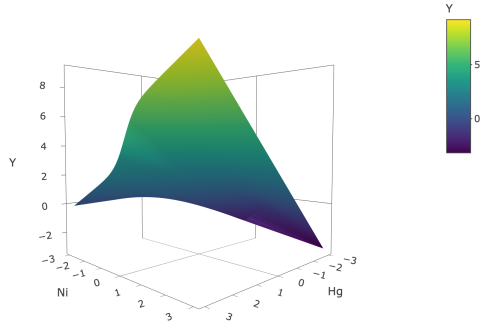

Figure A.3: Exposure-response surface for a multiplicative interaction between Hg and Ni at the lower effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.35\text{Hg}*\text{Ni}$.

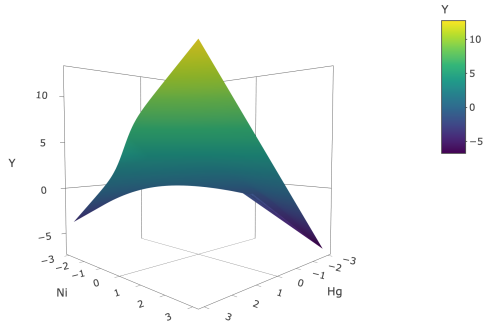

Figure A.4: Exposure-response surface for a multiplicative interaction between Hg and Ni at the higher effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.7\text{Hg}*\text{Ni}$.

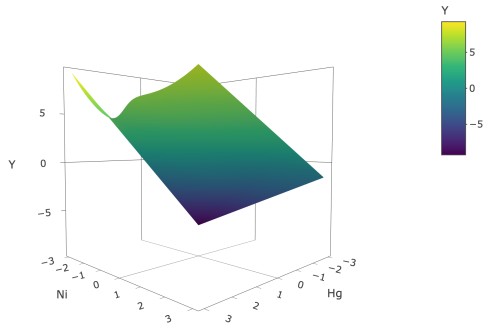

Figure A.5: Exposure-response surface for a polynomial interaction between Hg and Ni at the lower effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.3\text{Hg}*(\text{Ni}-1)^2$.

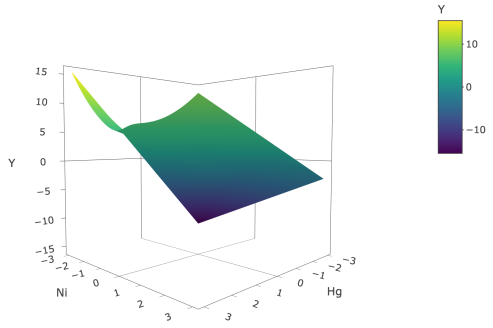

Figure A.6: Exposure-response surface for a polynomial interaction between Hg and Ni at the higher effect size: $Y = \text{Hg} + \frac{3}{1+\exp(-4\text{Ni})} + 0.6\text{Hg}*(\text{Ni}-1)^2$.

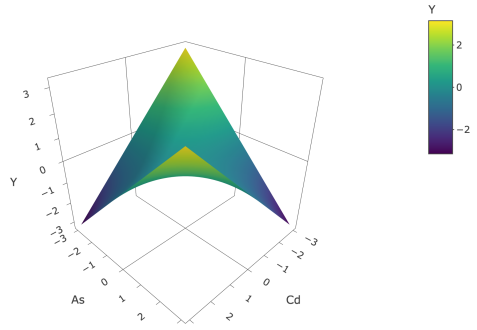

Figure A.7: Exposure-response surface for a multiplicative interaction between Cd and As at the lower effect size: $Y = 0.35\text{Cd}*\text{As}$.

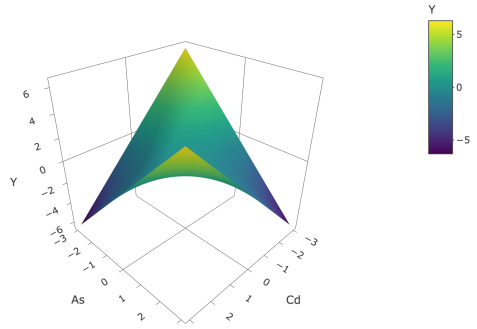

Figure A.8: Exposure-response surface for a multiplicative interaction between Cd and As at the higher effect size: $Y = 0.7\text{Cd}*\text{As}$.

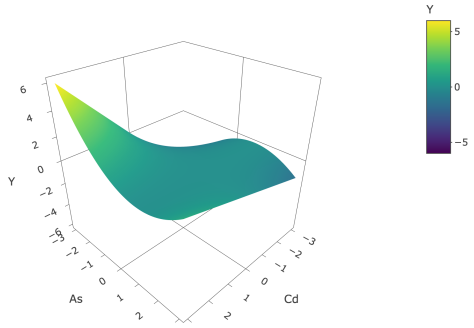

Figure A.9: Exposure-response surface for a polynomial interaction between Cd and As at the lower effect size: $0.125\text{Cd}*(\text{As}-1)^2$.

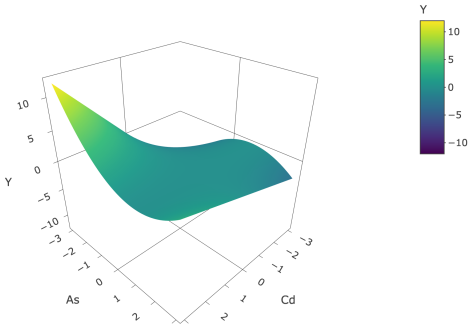

Figure A.10: Exposure-response surface for a polynomial interaction between Cd and As at the higher effect size: $0.25\text{Cd}*(\text{As}-1)^2$.

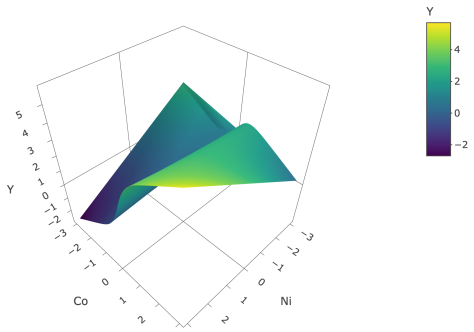

Figure A.11: Exposure-response surface for a multiplicative interaction between Ni and Co at the lower effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.3\text{Ni}*\text{Co}$.

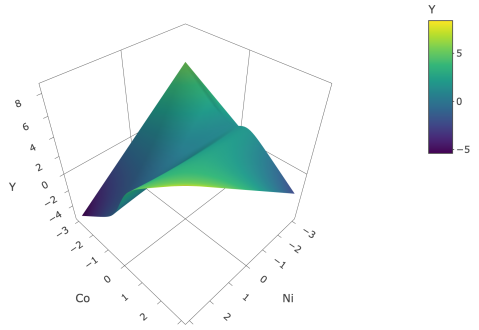

Figure A.12: Exposure-response surface for a multiplicative interaction between Ni and Co at the higher effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.6\text{Ni}*\text{Co}$.

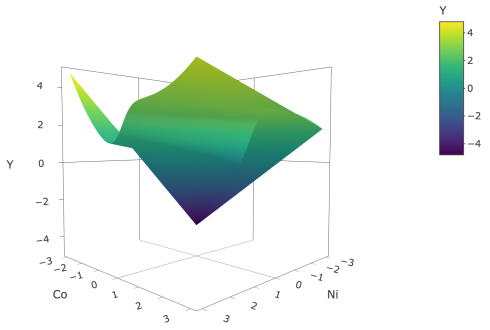

Figure A.13: Exposure-response surface for a polynomial interaction between Ni and Co at the lower effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.09\text{Ni}*(\text{Co}-1)^2$.

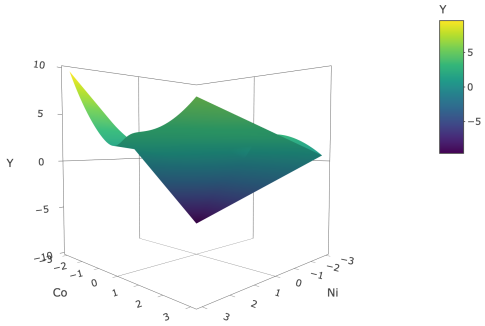

Figure A.14: Exposure-response surface for a polynomial interaction between Ni and Co at the lower effect size: $Y = \frac{3}{1+\exp(-4\text{Ni})} + 0.18\text{Ni}*(\text{Co}-1)^2$.

Figure A.15 shows the detailed distribution of correlations in smaller size simulated datasets.

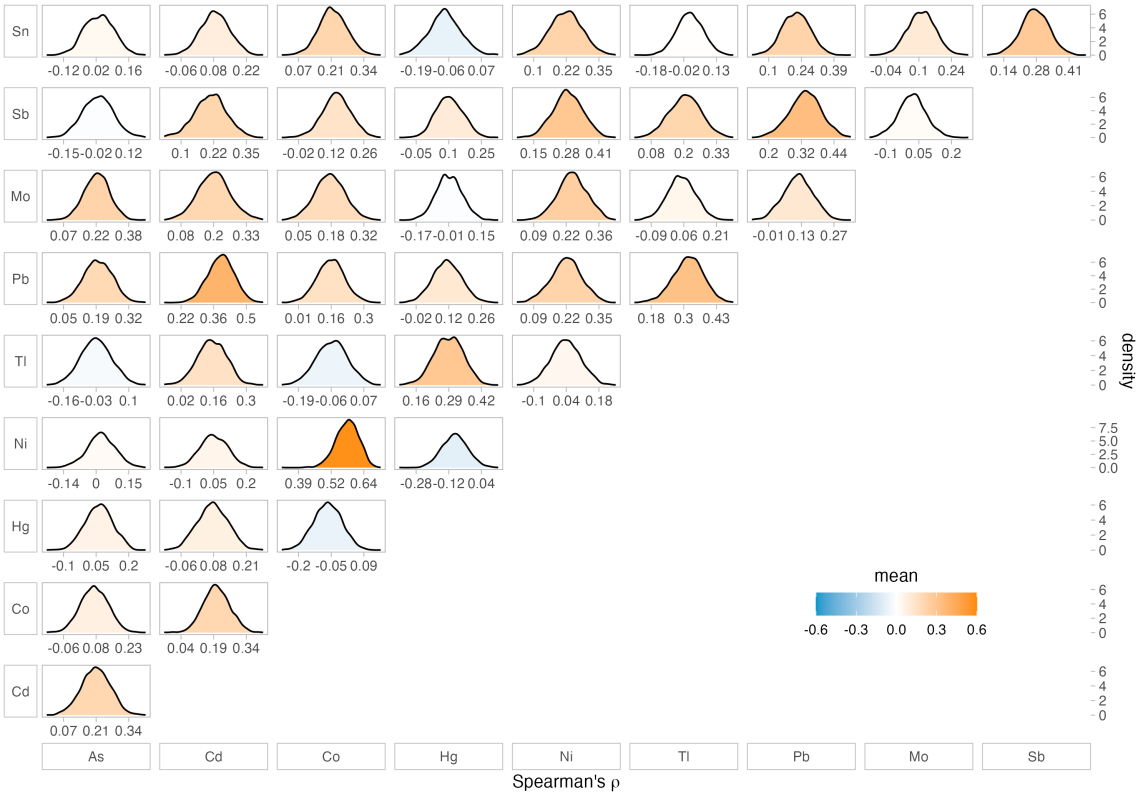

Figure A.15: Distributions of Spearman's correlation from 2100 smaller size (n=252) simulated datasets.

Figures A.16, A.17, and A.18 compare the marginal distributions of predictors and dependence structure between exposures of the observed dataset and simulated datasets of larger size (n=1000).

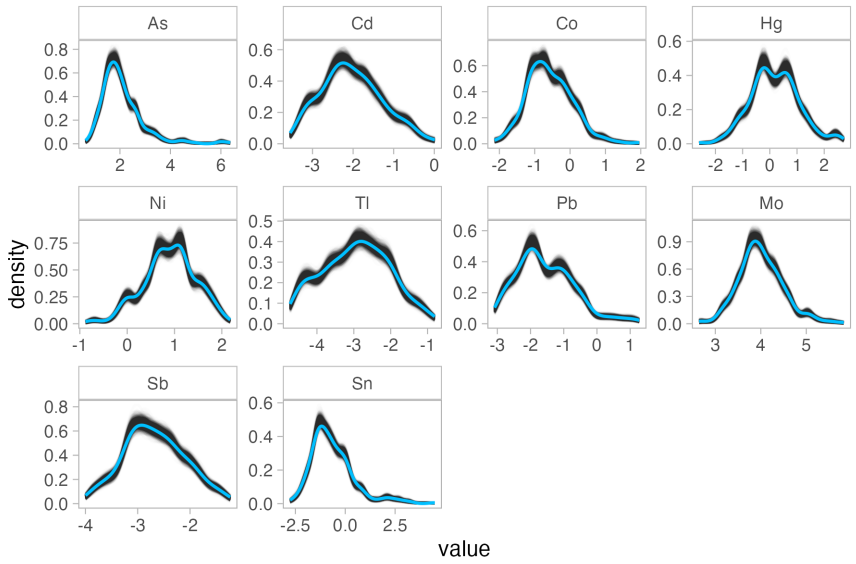

Figure A.16: Distributions of exposures from observed data (blue) and simulated larger size (n=1000) datasets (gray).

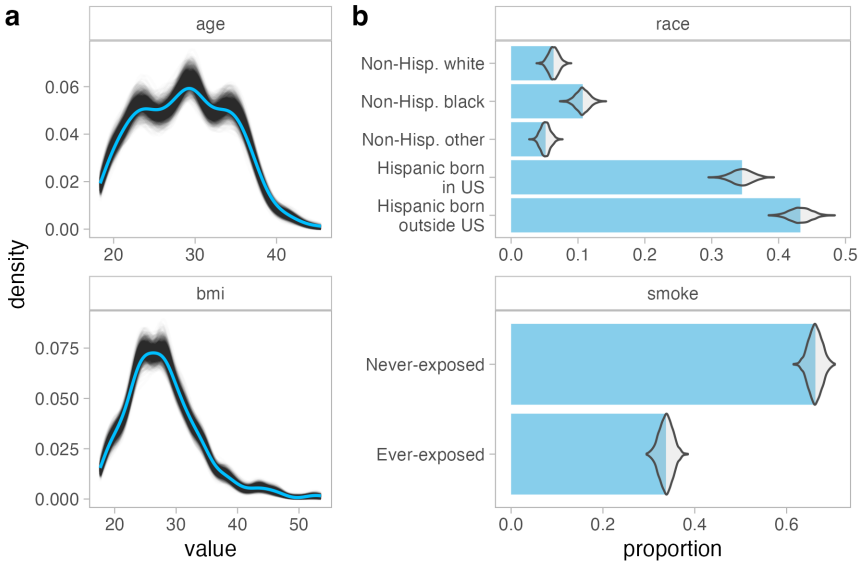

Figure A.17: Distributions of covariates from observed data (blue) and simulated larger size (n=1000) datasets (gray).

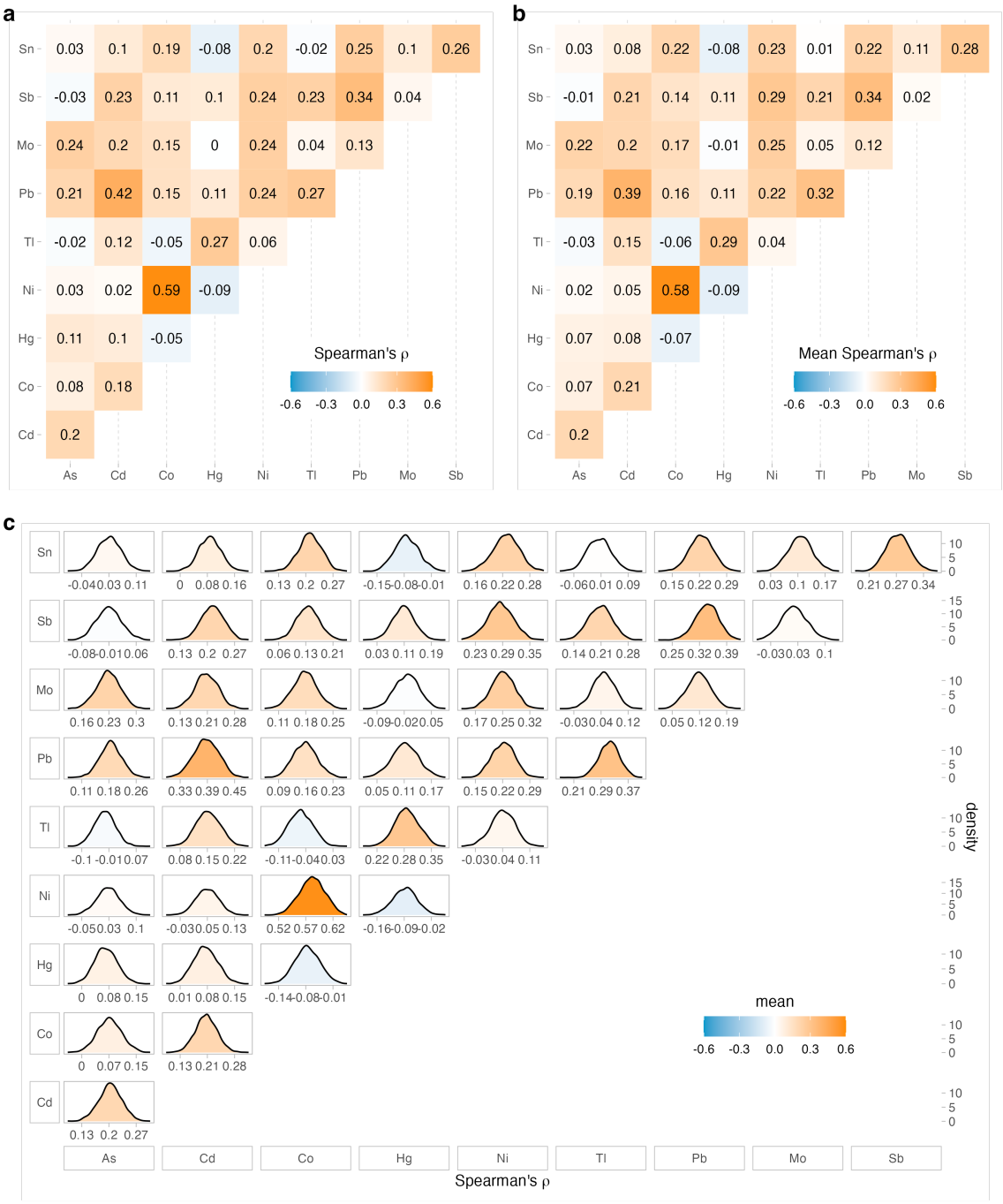

Figure A.18: Spearman's correlation heat maps of exposures from observed data (a) and averaged from 2100 larger size (n=1000) simulated datasets (b), as well as distributions of correlations from larger size simulated datasets (c).

Figure A.19 visualizes the distribution of R^2 values in smaller and larger size simulated datasets.

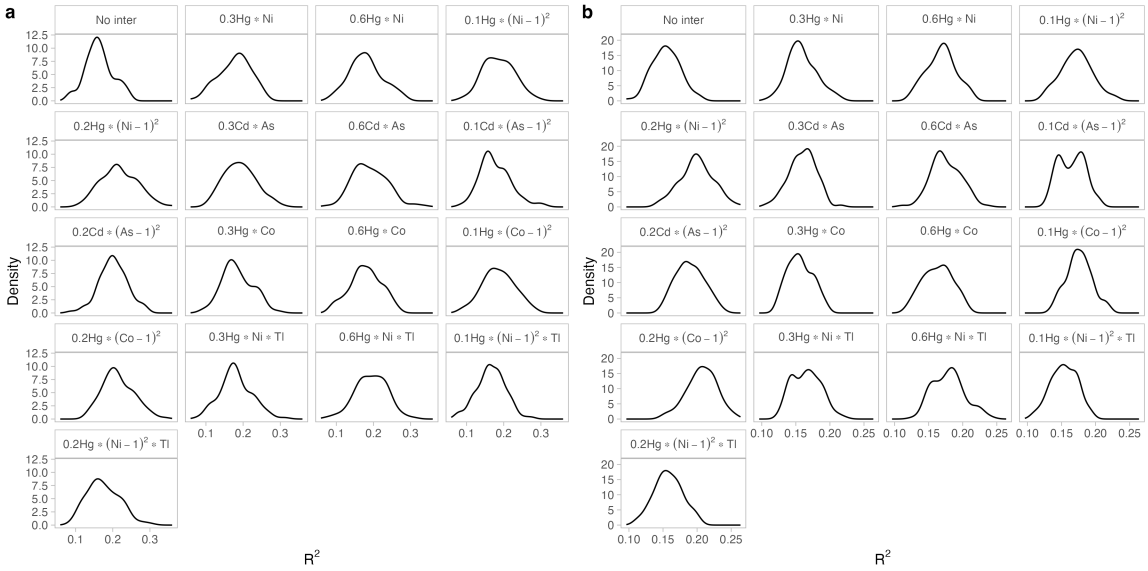

Figure A.19: R^2 values from multiple linear regressions with only the true functional form of significant exposures in smaller size (a) and larger size (b) simulated datasets.

Figure A.20 compares the degrees of freedom selected using the WAIC criterion when fitting BSR using 5,000 MCMC iterations to 50,000 MCMC iterations. We ran this test on five smaller size datasets from the 16 scenarios containing interactions between exposures, as well as from the base case.

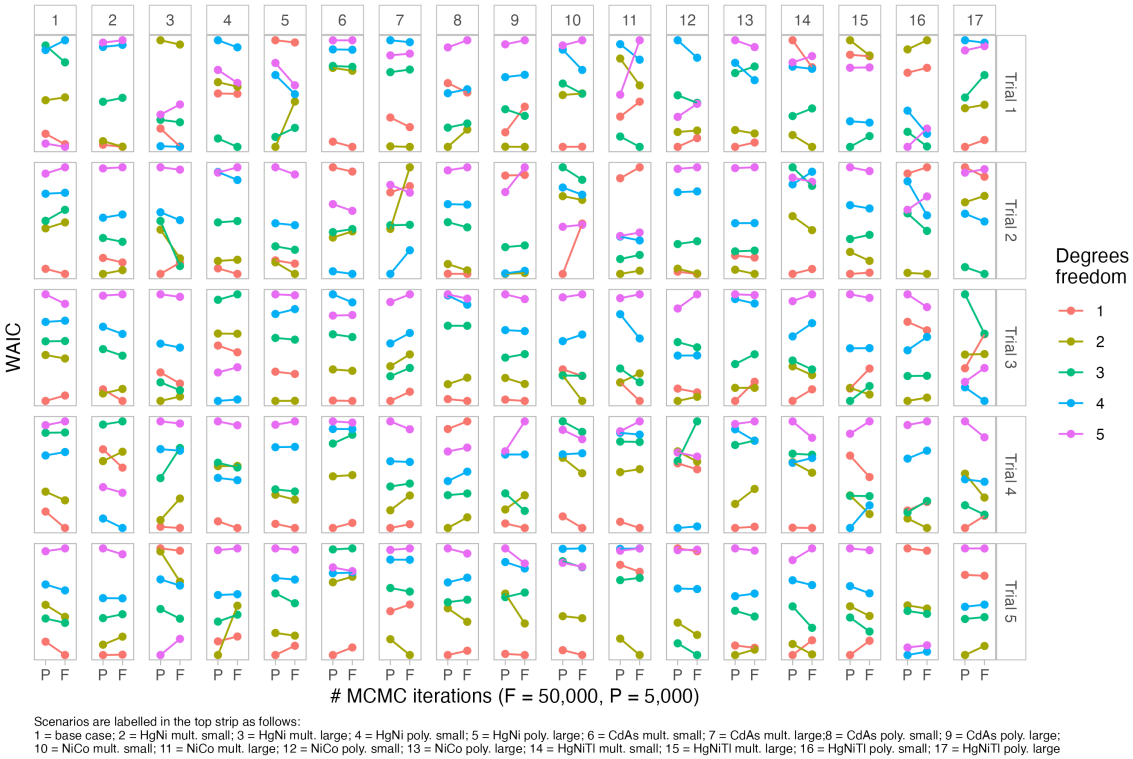

Figure A.20: Test comparing WAIC selection of degrees of freedom from BSR models fit with either 5,000 or 50,000 MCMC iterations.

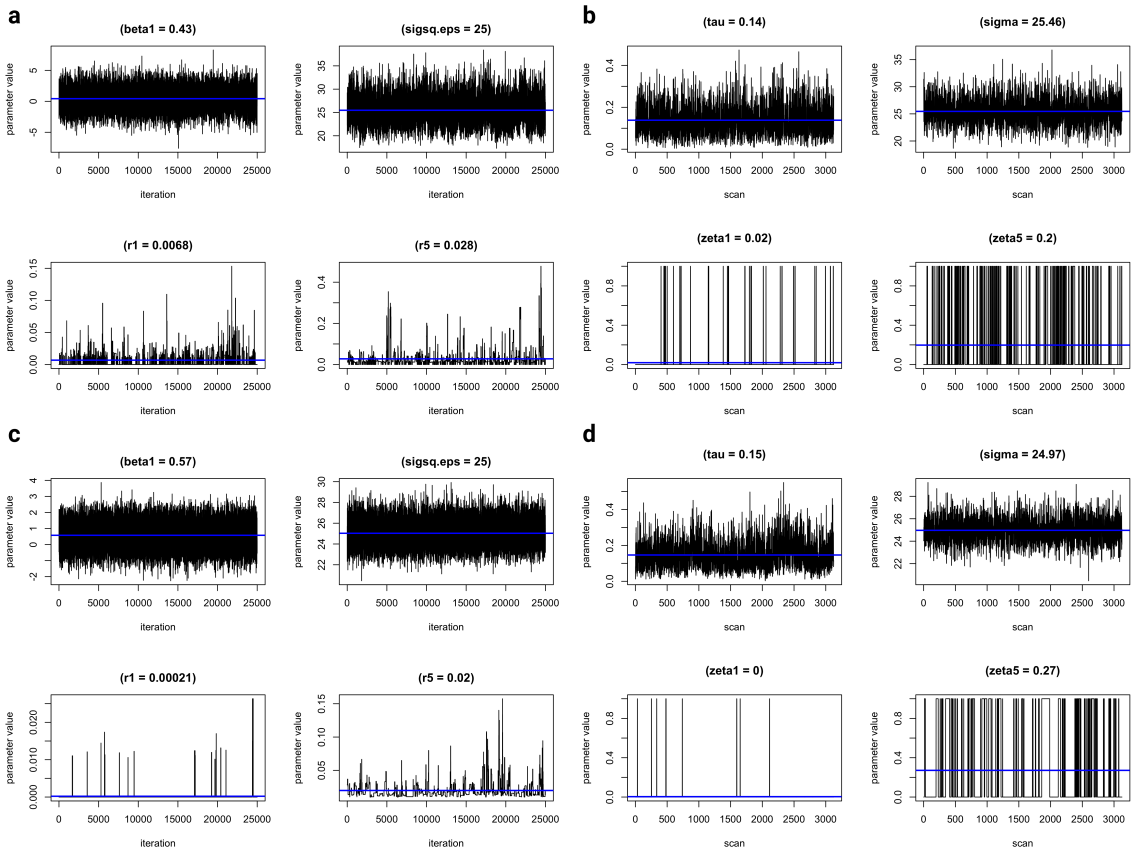

Figure A.21: Examples of trace plots from smaller size BKMR (a) and BSR (b) as well as larger size BKMR (c) and BSR (d) in scenarios with larger effect size interactions between Hg and Ni.

Finally, we show why the order of chemicals in the confidence intervals we use to assess two-way interactions in BKMR is arbitrary. Let $\widehat{Y}(X1_a, X2_b)$ be the estimated response at the ath quantile of $X1$, the first chemical, the bth quantile of $X2$, the second chemical, while holding all other chemicals at their 0.5 quantiles. Then, the estimated confidence interval of the difference in the response X_1's 0.75 and 0.25 quantiles, at both the 0.75 and 0.25 quantiles of X_2, can be shown to be equivalent

to swapping the order of $X1$ and $X2$:

$$\begin{aligned}
&[\widehat{Y}(X1_{0.75}, X2_{0.75}) - \widehat{Y}(X1_{0.25}, X2_{0.75})] - [\widehat{Y}(X1_{0.75}, X2_{0.25}) - \widehat{Y}(X1_{0.25}, X2_{0.25})] \\
&= \widehat{Y}(X1_{0.75}, X2_{0.75}) - \widehat{Y}(X1_{0.25}, X2_{0.75}) - \widehat{Y}(X1_{0.75}, X2_{0.25}) + \widehat{Y}(X1_{0.25}, X2_{0.25}) \\
&= \widehat{Y}(X1_{0.75}, X2_{0.75}) - \widehat{Y}(X1_{0.75}, X2_{0.25}) - \widehat{Y}(X1_{0.25}, X2_{0.75}) + \widehat{Y}(X1_{0.25}, X2_{0.25}) \\
&= [\widehat{Y}(X1_{0.75}, X2_{0.75}) - \widehat{Y}(X1_{0.75}, X2_{0.25})] - [\widehat{Y}(X1_{0.25}, X2_{0.75}) - \widehat{Y}(X1_{0.25}, X2_{0.25})].
\end{aligned} \tag{A.1}$$

A.2 Results

Figures A.22-A.29 display the full distributions of univariate p-values and PIPs for all models in scenarios with interactions between chemicals that were not included in Chapter 4.3.2.

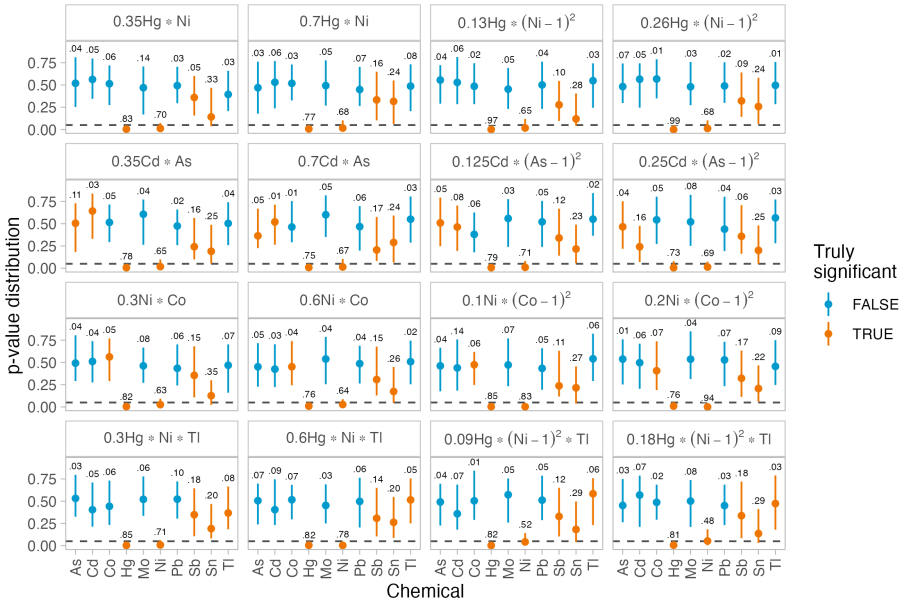

Figure A.22: P-value distributions of univariate chemicals from naive MLRs run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles.

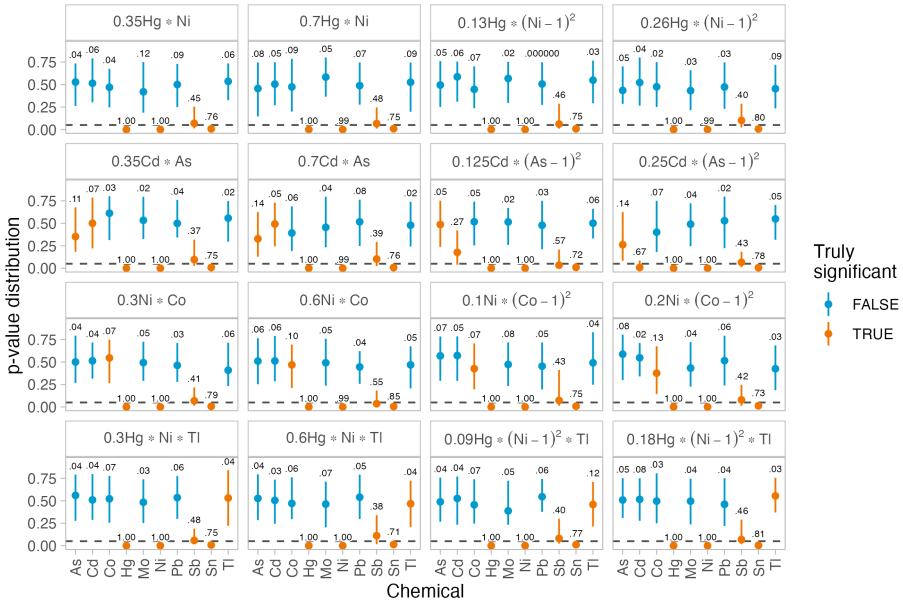

Figure A.23: P-value distributions of univariate chemicals from naive MLRs run on larger size (n=1000) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles.

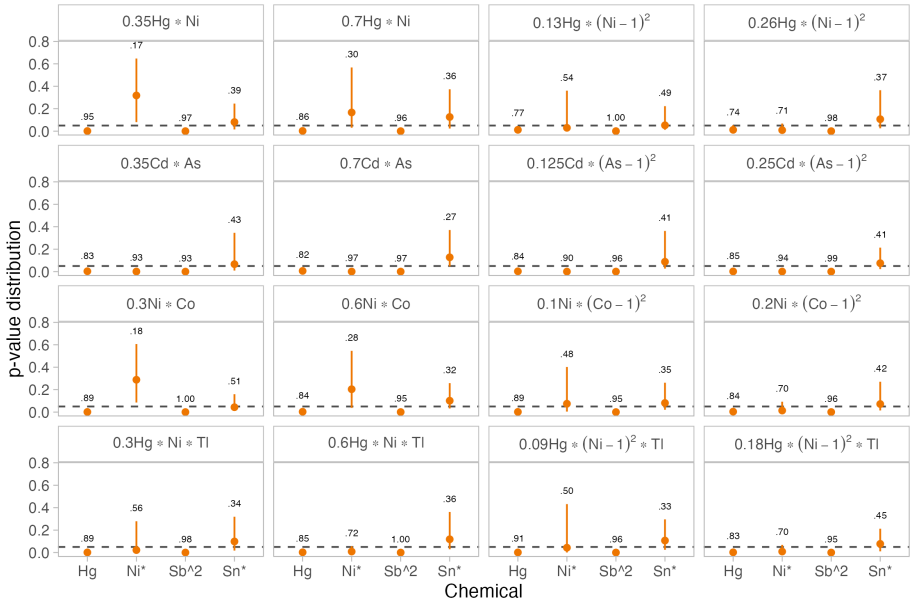

Figure A.24: P-value distributions of univariate chemicals from oracle MLRs run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Sensitivities are displayed above a point-range with the median and first and third quartiles.

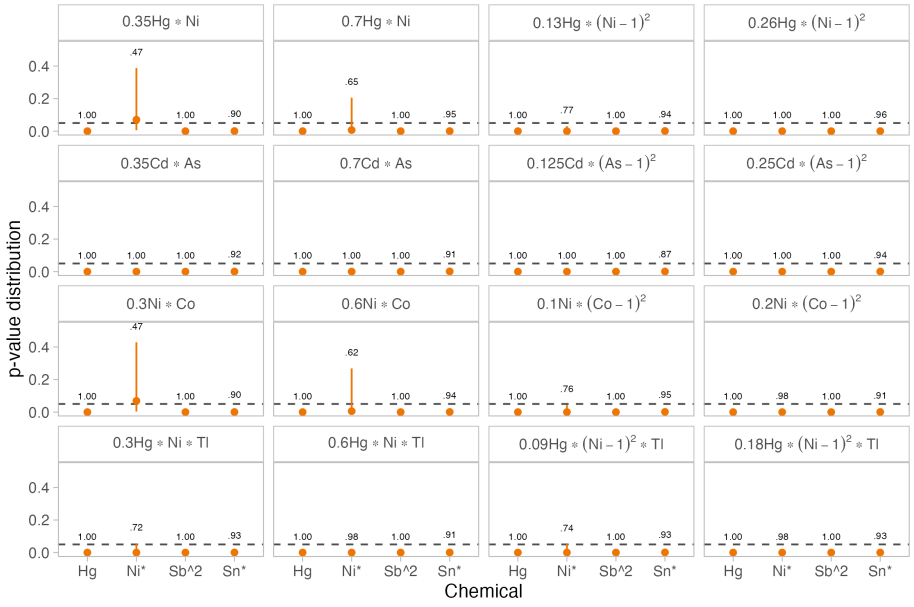

Figure A.25: P-value distributions of univariate chemicals from oracle MLRs run on larger size (n=1000) datasets, in all scenarios with interactions between chemicals. Sensitivities are displayed above a point-range with the median and first and third quartiles.

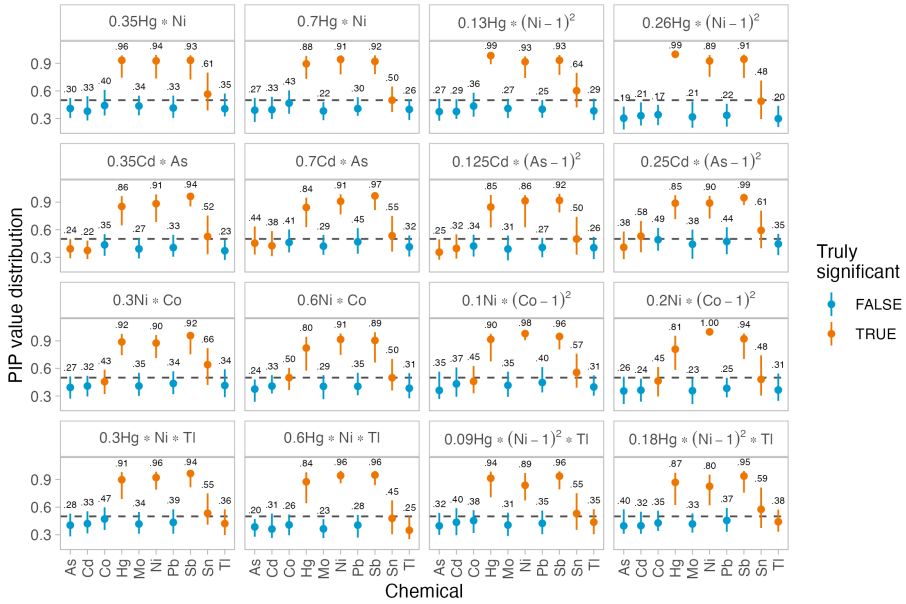

Figure A.26: PIP distributions of univariate chemicals from BKMR models run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles.

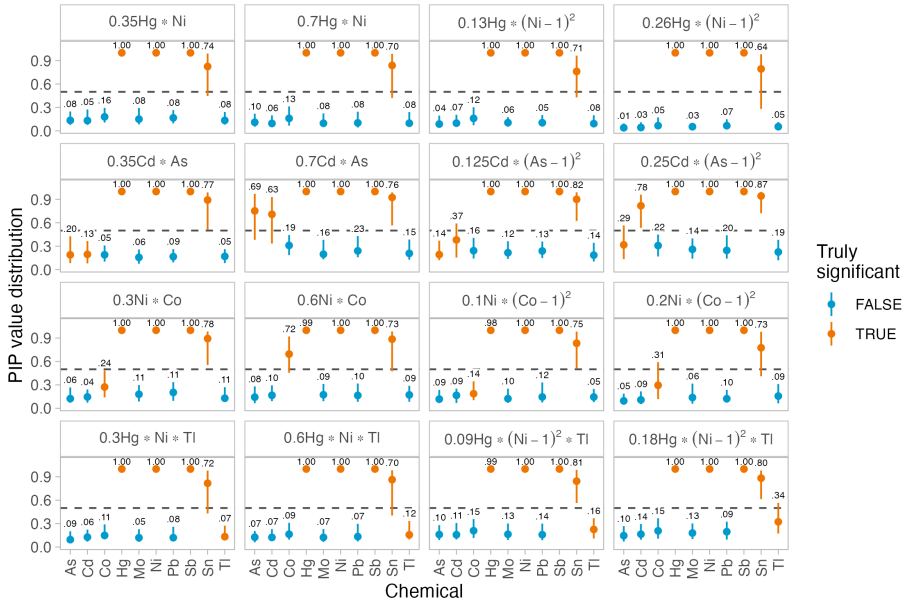

Figure A.27: PIP distributions of univariate chemicals from BKMR models run on larger size (n=1000) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles.

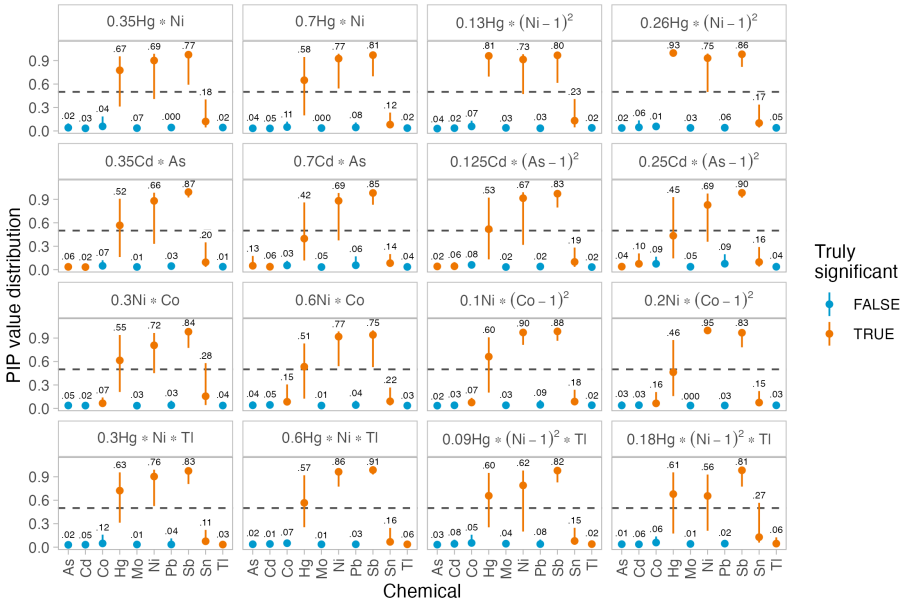

Figure A.28: PIP distributions of univariate chemicals from BSR models run on smaller size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles.

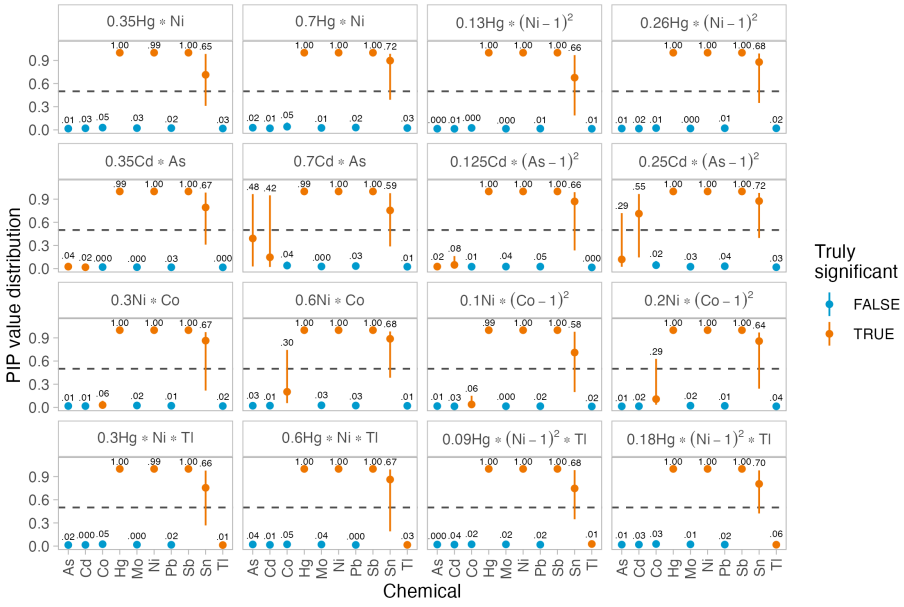

Figure A.29: PIP distributions of univariate chemicals from BSR models run on larger size (n=252) datasets, in all scenarios with interactions between chemicals. Detection rates are displayed above a point-range with the median and first and third quartiles.

Figures A.30 and A.31 display the full p-value distributions on interaction terms between chemicals in the oracle MLR models.

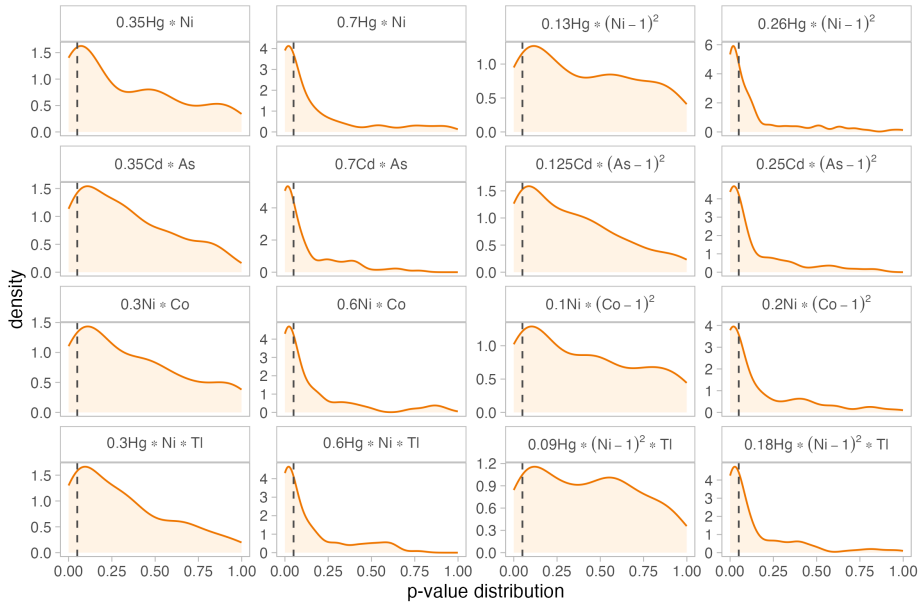

Figure A.30: P-value distributions of interaction terms from oracle MLRs run on smaller size (n=252) datasets, from all scenarios with interactions between chemicals.

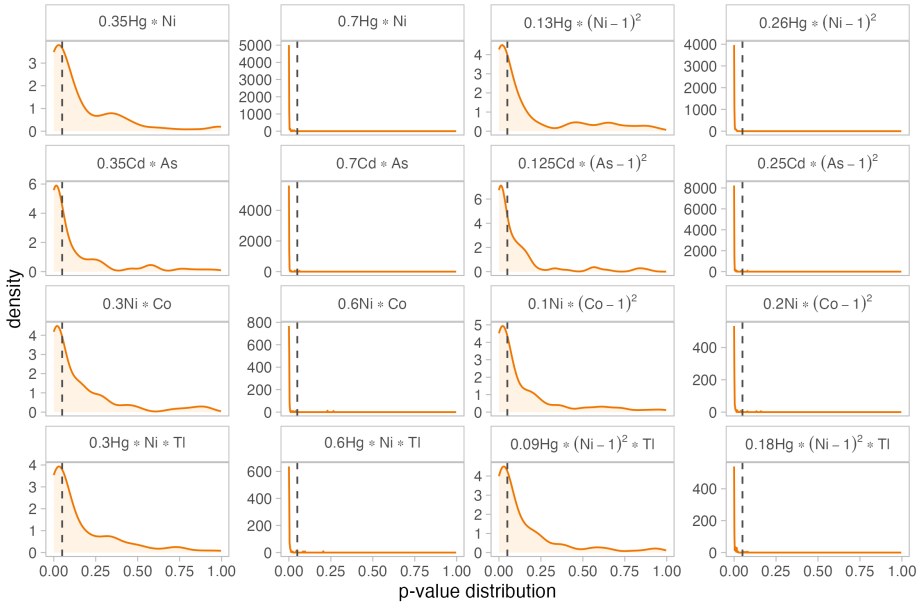

Figure A.31: P-value distributions of interaction terms from oracle MLRs run on larger size (n=1000) datasets, from all scenarios with interactions between chemicals.

Table A.1 summarizes the sensitivities and false discovery rates for univariate terms in scenarios with interactions between chemicals. Figures A.32-A.43 display the estimated exposure-response surface for one chemical while fixing one (or two, for three-way interactions) other chemicals at their 0.1, 0.5, and 0.9 quantiles, for all scenarios not included in Chapter 4.3.2.

Table A.1: Overall false discovery rate for univariate chemicals in all scenarios with interactions between chemicals.

		Small			Large		
Type	Effect size	Naive	BKMR	BSR	Naive	BKMR	BSR
Hg-Ni							
Mult.	Lower	0.06	0.34	0.03	0.07	0.09	0.03
Mult.	Higher	0.03	0.20	0.04	0.04	0.04	0.01
Poly.	Lower	0.04	0.29	0.04	0.04	0.07	0.00
Poly.	Higher	0.05	0.30	0.05	0.07	0.09	0.02
Cd-As							
Mult.	Lower	0.05	0.27	0.03	0.05	0.10	0.01
Mult.	Higher	0.07	0.44	0.07	0.16	0.30	0.16
Poly.	Lower	0.05	0.29	0.04	0.08	0.18	0.03
Poly.	Higher	0.04	0.38	0.06	0.06	0.34	0.16
Ni-Co							
Mult.	Lower	0.06	0.34	0.04	0.05	0.11	0.02
Mult.	Higher	0.06	0.29	0.05	0.06	0.12	0.06
Poly.	Lower	0.07	0.37	0.04	0.06	0.10	0.02
Poly.	Higher	0.04	0.34	0.05	0.06	0.20	0.07
Hg-Ni-Tl							
Mult.	Lower	0.06	0.36	0.04	0.05	0.08	0.02
Mult.	Higher	0.04	0.36	0.04	0.04	0.16	0.03
Poly.	Lower	0.05	0.35	0.05	0.06	0.13	0.02
Poly.	Higher	0.06	0.26	0.03	0.05	0.08	0.03

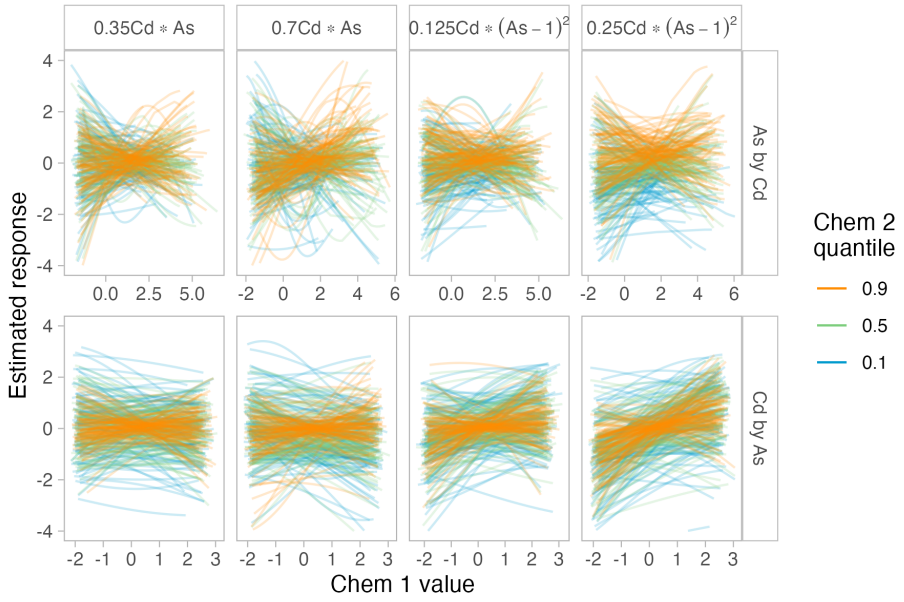

Figure A.32: Exposure-response relationships estimated by BKMR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles.

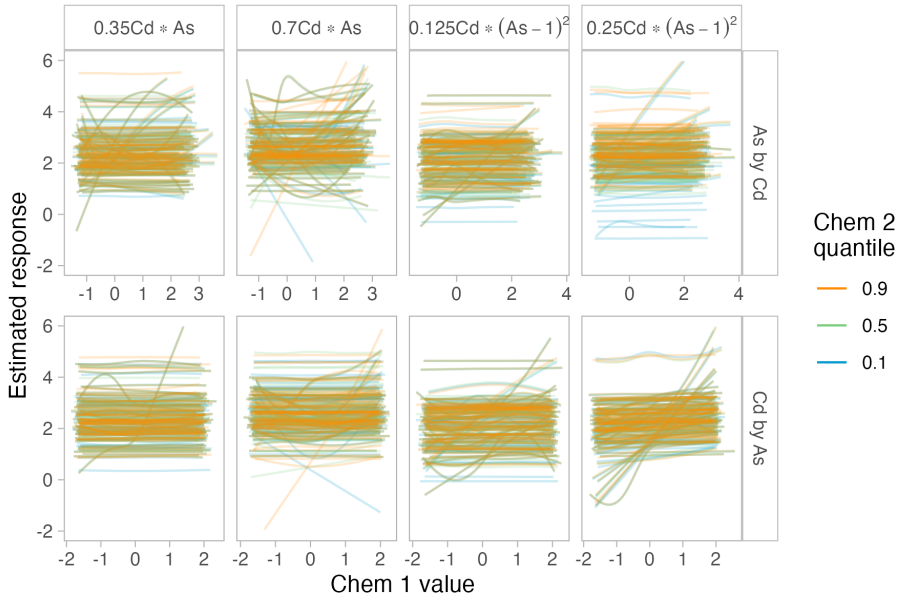

Figure A.33: Exposure-response relationships estimated by BSR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles.

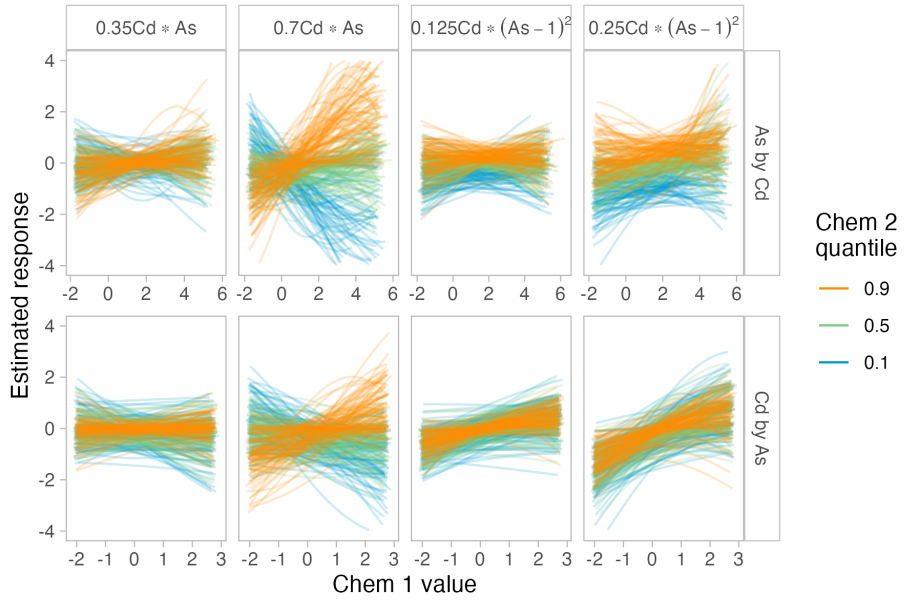

Figure A.34: Exposure-response relationships estimated by BKMR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles.

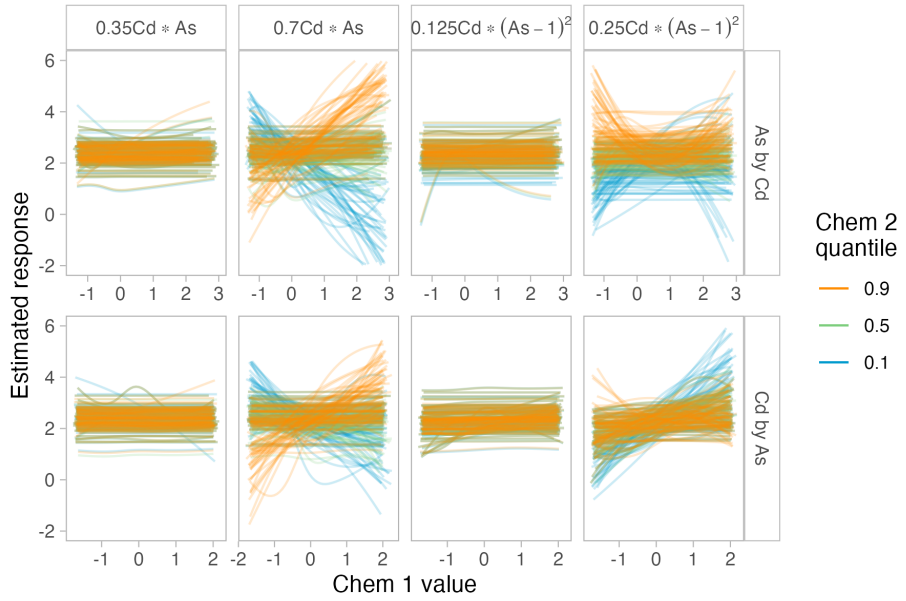

Figure A.35: Exposure-response relationships estimated by BSR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Cd and As. All other chemicals are fixed at 0.5 quantiles.

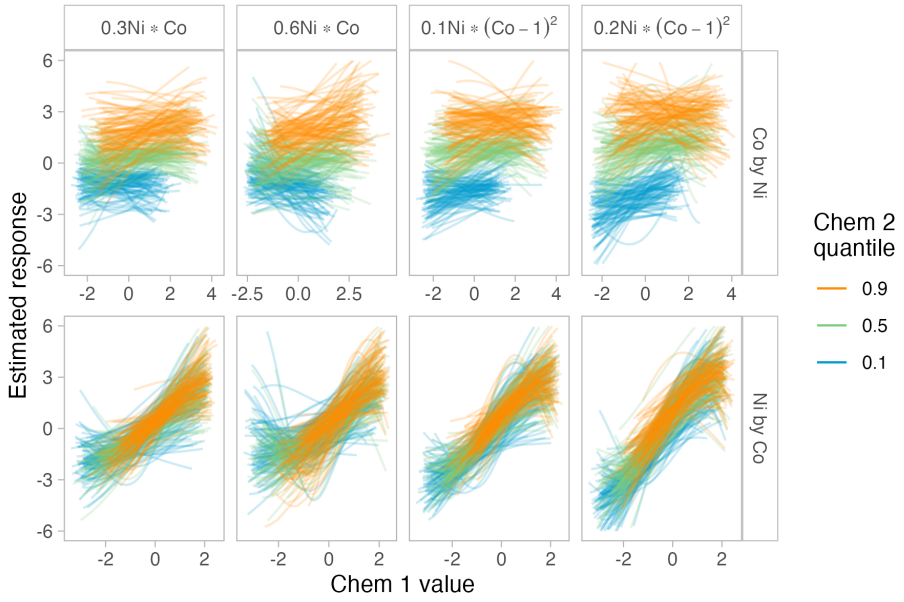

Figure A.36: Exposure-response relationships estimated by BKMR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles.

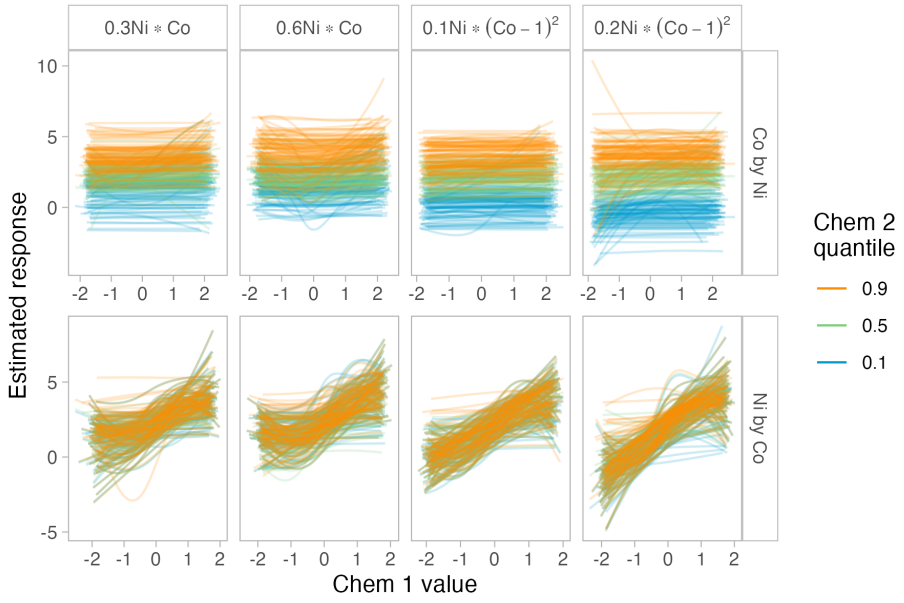

Figure A.37: Exposure-response relationships estimated by BSR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles.

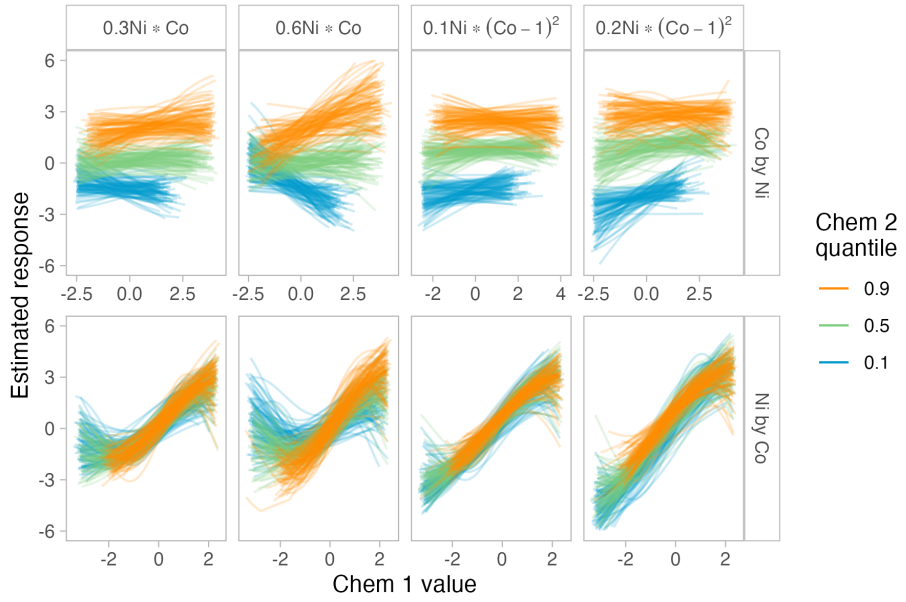

Figure A.38: Exposure-response relationships estimated by BKMR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles.

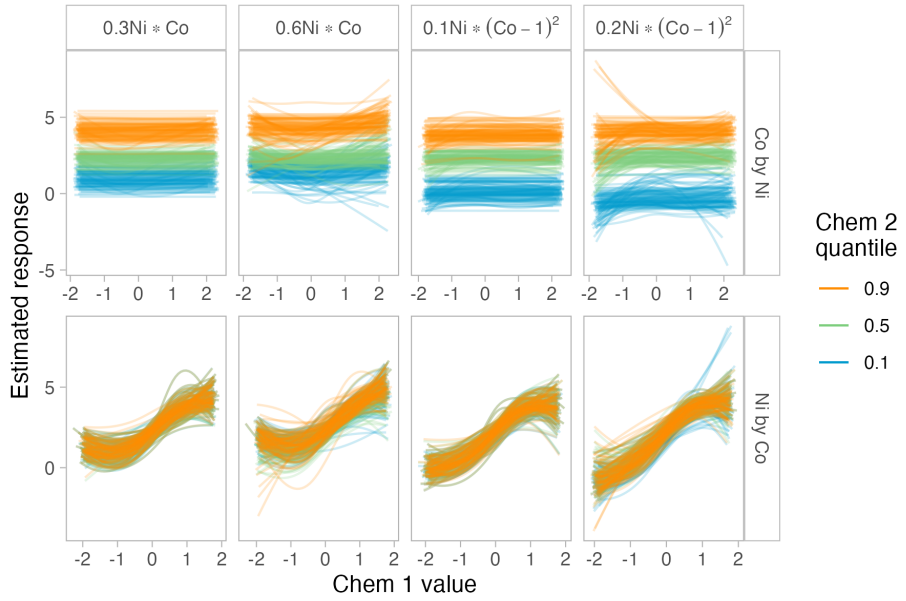

Figure A.39: Exposure-response relationships estimated by BSR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of another to assess interactions between Ni and Co. All other chemicals are fixed at 0.5 quantiles.

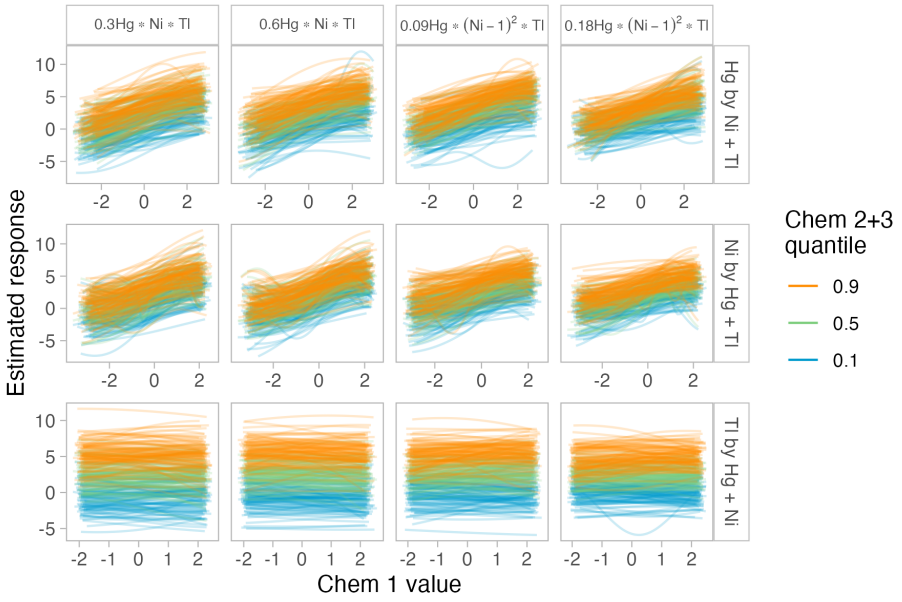

Figure A.40: Exposure-response relationships estimated by BKMR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles.

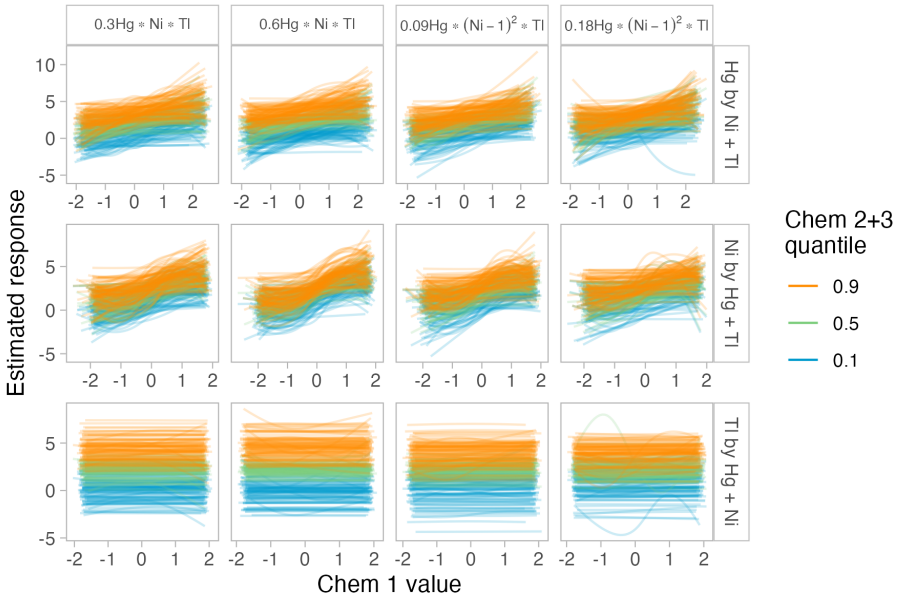

Figure A.41: Exposure-response relationships estimated by BSR in smaller size (n=252) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles.

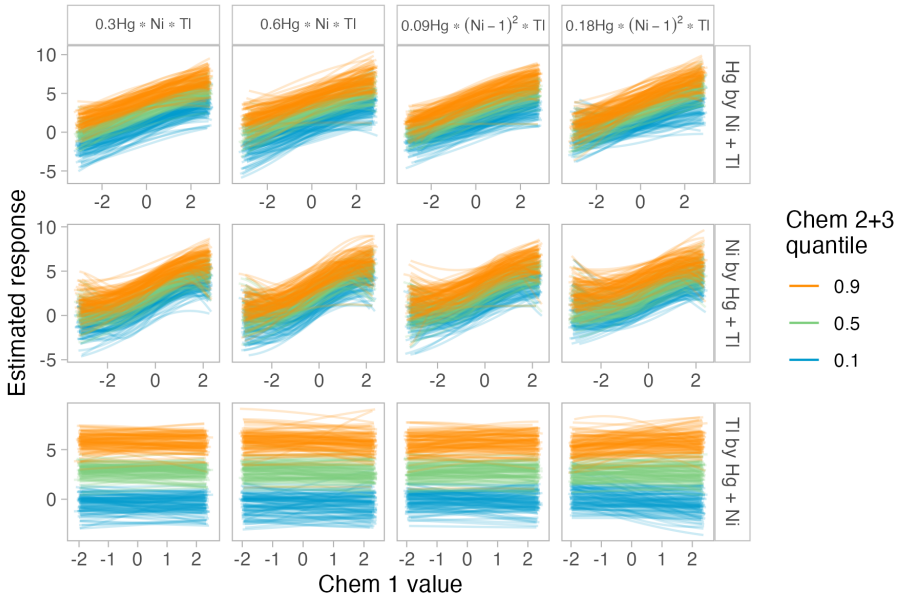

Figure A.42: Exposure-response relationships estimated by BKMR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles.

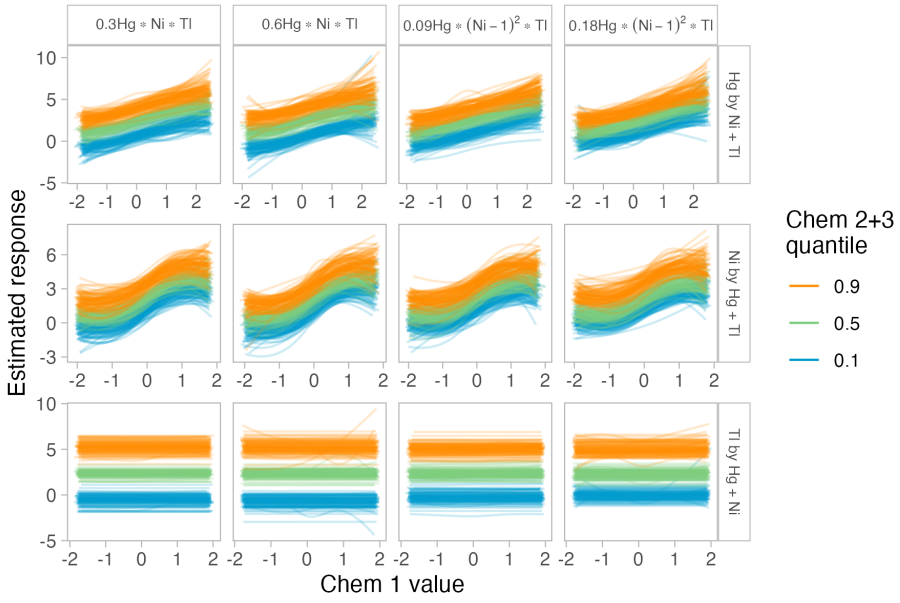

Figure A.43: Exposure-response relationships estimated by BSR in larger size (n=1000) datasets, using the first chemical fixed at quantiles of two others to assess interactions between Ni, Hg, and Tl. All other chemicals are fixed at 0.5 quantiles.

Table A.2 summarizes the sensitivities for Hg in scenarios with interactions between the categorical race covariate and Hg. Note that the naive and oracle MLRs do not have an associated race category, because we did not stratify them in our analysis.

Table A.2: Sensitivity for the univariate Hg term in all scenarios with an interaction between the categorical race covariate and Hg. Sensitivities for BKMR and BSR models are stratified by race.

		Small race cat.		**Large race cat.**	
Model	Race	Lower	Higher	Lower	Higher
Small					
Naive MLR	-	0.84	0.79	0.93	0.99
Oracle MLR	-	0.82	0.78	0.70	0.56
BKMR	Non-Hisp. white (n=16)	0.52	0.47	0.42	0.51
	Non-Hisp. black (n=27)	0.60	0.69	0.50	0.56
	Non-Hisp. other (n=13)	0.30	0.27	0.27	0.20
	Hisp. born in US (n=87)	0.56	0.50	0.57	0.60
	Hisp. born outside US (n=109)	0.68	0.63	0.89	0.97
	Collapsed non-Hisp. (n=56)	0.54	0.65	0.55	0.46
BSR	Non-Hisp. white (n=16)	0.32	0.30	0.42	0.40
	Non-Hisp. black (n=27)	0.26	0.41	0.22	0.23
	Non-Hisp. other (n=13)	0.36	0.35	0.32	0.40
	Hisp. born in US (n=87)	0.23	0.15	0.13	0.18
	Hisp. born outside US (n=109)	0.23	0.25	0.53	0.80
	Collapsed non-Hisp. (n=56)	0.17	0.33	0.23	0.16
Large					
Naive MLR	-	1.00	1.00	1.00	1.00
Oracle MLR	-	1.00	1.00	0.99	1.00
BKMR	Non-Hisp. white (n=16)	0.45	0.56	0.50	0.43
	Non-Hisp. black (n=27)	0.88	0.93	0.59	0.59
	Non-Hisp. other (n=13)	0.55	0.41	0.52	0.53
	Hisp. born in US (n=87)	0.90	0.93	0.88	0.95
	Hisp. born outside US (n=109)	0.92	0.94	1.00	1.00
BSR	Non-Hisp. white (n=16)	0.16	0.18	0.18	0.13
	Non-Hisp. black (n=27)	0.56	0.82	0.24	0.30
	Non-Hisp. other (n=13)	0.17	0.11	0.18	0.17
	Hisp. born in US (n=87)	0.76	0.70	0.71	0.70
	Hisp. born outside US (n=109)	0.79	0.85	1.00	1.00

Appendix B Code

This second appendix includes all of the R chunks of code that were hidden throughout the document.

B.1 Code for Chapter 3

The code for this section generates a toy example, used to demonstrate the kernel machine regression and spline regression techniques.

```
# load packages
library(tidyverse)
library(stats)
library(splines)
```

```
# set theme for plots
theme_set(theme_light())
theme_update(panel.grid.major = element_blank(),
             panel.grid.minor = element_blank())
theme_update(
  strip.background = element_rect(color="gray", fill="white"),
  strip.text = element_text(color = "gray30")
  )
```

```
########
# generate simulated points
########

# generate data from distribution
set.seed(0) # reproducibility
x <- seq(0, 25, length.out = 51)
Y <- exp(x/10) + 2*sin(x/2) + rnorm(51, mean = 0, sd = 0.5)
df <- data.frame(x, Y)

# plot data and linear regression line
q1 <- ggplot(df, aes(x, Y)) +
  geom_point() +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
```

```
  geom_smooth(method = "lm", formula = "y~x",
              color = "deepskyblue3", fill = "gray70",
              linewidth = 0.5, se = F)

# save plot
ggsave("index/figures/ch3_toy1.png", plot = q1, device = "png",
       width = 5, height = 3)
```

```
########
# kernel regression
########

# get normal distribution of weights around query points
df$Weight <- dnorm(df$x, mean = 12.5, sd = 1)

# plot points colored by their weights
p1 <- ggplot(df, aes(x, Y)) +
  geom_point(aes(color = Weight)) +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
  geom_vline(xintercept = 12.5, linetype = "dotted") +
  theme(legend.position = "none")

# plot a curve of weights
normcurv <- data.frame(x = seq(0, 25, length.out = 250))
normcurv$Weight <- dnorm(normcurv$x, mean = 12.5, sd = 1)
p2 <- ggplot(normcurv, aes(x, Weight, color = Weight)) +
  geom_line() +
  scale_y_continuous(breaks = c(0, 0.2, 0.4)) +
  theme(legend.position = "none")

# stitch plots together
q2 <- cowplot::plot_grid(p1, p2, ncol = 1, rel_heights = c(0.7, 0.3))
q2

# save plot
ggsave("index/figures/ch3_toy2.png", plot = q2, device = "png",
       width = 5, height = 4)

# fit kernel regression with sigma = 1, bandwidth = 8/3
kmr_toy <- ksmooth(df$x, df$Y, kernel = "normal",
                   bandwidth = 8/3, x.points = df$x)
df <- df |>
  left_join(as.data.frame(kmr_toy), by = "x") |>
  rename(Yhat = y)

# plot kernel regression estimation
q3 <- ggplot(df) +
  geom_point(aes(x, Y)) +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
  geom_line(aes(x, Yhat), color = "deepskyblue3")
q3

# save plot
ggsave("index/figures/ch3_toy3.png", plot = q3, device = "png",
       width = 5, height = 3)

# fit kernel regression with sigma = 5, bandwidth = 40/3
kmr_toy_5 <- ksmooth(df$x, df$Y, kernel = "normal",
                     bandwidth = 40/3, x.points = df$x)

# fit kernel regression with sigma = 0.1, bandwith = 8/30
```

```r
kmr_toy_1 <- ksmooth(df$x, df$Y, kernel = "normal",
                     bandwidth = 8/30, x.points = df$x)

# re-join data
dfrho <- df |>
  left_join(as.data.frame(kmr_toy_5), by = "x") |>
  rename("rho = 50" = y) |>
  left_join(as.data.frame(kmr_toy_1), by = "x") |>
  rename("rho = 0.02" = y) |>
  select(-Yhat) |>
  pivot_longer(cols = c("rho = 50", "rho = 0.02"), values_to = "Yhat")

# labeller for rho in facet
rho_labeller <- function(name) {
  return(ifelse(grepl("50", name),
                latex2exp::TeX("$\\rho=50$"),
                latex2exp::TeX("$\\rho=0.02$")))
}

# plot kernel regression with two values of rho
qrho <- ggplot(dfrho) +
  geom_point(aes(x, Y)) +
  geom_line(aes(x, Yhat), color = "deepskyblue3") +
  facet_wrap(~name, labeller = as_labeller(rho_labeller, default = label_parsed))
qrho

# save plot
ggsave("index/figures/ch3_toyrho.png", plot = qrho, device = "png",
       width = 7, height = 3)
```

```r
########
# spline regression
########

kn <- c(5, 10, 15, 20) # 4 knots of equal width

# fit linear spline regression
spline_toy_line <- lm(Y ~ bs(x, knots = kn, degree = 1), data = df)
p_line <- predict(spline_toy_line, se = T)
df$Yhats_line <- p_line$fit

q4 <- ggplot(df) +
  geom_point(aes(x, Y)) +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
  geom_line(aes(x, Yhats_line), color = "deepskyblue3") +
  geom_vline(xintercept = kn, linetype = "dotted")
q4

# save plot
ggsave("index/figures/ch3_toy4.png", plot = q4, device = "png",
       width = 5, height = 3)

# fit cubic spline regression
spline_toy_cub <- lm(Y ~ bs(x, knots = kn), data = df)
p_cub <- predict(spline_toy_cub, se = T)
df$Yhats_cub <- p_cub$fit

# plot spline regression estimation
q5 <- ggplot(df) +
  geom_point(aes(x, Y)) +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
```

```
  geom_line(aes(x, Yhats_cub), color = "deepskyblue3") +
  geom_vline(xintercept = kn, linetype = "dotted")
q5

# save plot
ggsave("index/figures/ch3_toy5.png", plot = q5, device = "png",
       width = 5, height = 3)

# fit natural spline regression
spline_toy_nat <- lm(Y ~ ns(x, knots = kn), data = df)
p_nat <- predict(spline_toy_nat, se = T)
df$Yhats_nat <- p_nat$fit

# plot spline regression estimation
q6 <- ggplot(df) +
  geom_point(aes(x, Y)) +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
  geom_line(aes(x, Yhats_nat), color = "deepskyblue3") +
  geom_vline(xintercept = c(5, 10, 15, 20), linetype = "dotted")
q6

# save plot
ggsave("index/figures/ch3_toy6.png", plot = q6, device = "png",
       width = 5, height = 3)

# see what happens outside of the bounds
x_longer <- seq(-5, 30, length.out = 81)
y_longer_cub <- predict(spline_toy_cub,
                        newdata = data.frame(x = x_longer))
y_longer_nat <- predict(spline_toy_nat,
                        newdata = data.frame(x = x_longer))

df_longer <- data.frame(
  x = c(x_longer, x_longer),
  spline = c(rep("Cubic", 81), rep("Natural", 81)),
  Yhat = c(y_longer_cub, y_longer_nat)
)

# plot outside of bounds
qbounds <- ggplot(df_longer) +
  geom_line(aes(x, Yhat), color = "deepskyblue3") +
  geom_function(fun = function(x) exp(x/10) + 2*sin(x/2),
                linetype = "dashed", color = "darkorange") +
  geom_vline(xintercept = c(0, 25), linetype = "dotted") +
  facet_wrap(~spline)
qbounds

# save plot
ggsave("index/figures/ch3_toybounds.png", plot = qbounds,
       device = "png", width = 7, height = 3)
```

B.2 Code for Chapter 4.2

The code for this section prepares the data from the MADRES study, generates simulated data, and fits multiple linear regressions, BKMR, and BSR on the simulated

data.

B.2.1 Cleaning MADRES data

First, we clean the data from the MADRES study. Note that this code requires two publicly available datasets, available from the HHEAR Data Repository, as outlined in Chapter 4.2.1.

```
# load packages
library(tidyverse)
```

```
# read in data
target <- read_csv("madres_data/1945_TARGETED_DATA.csv")
epi <- read_csv("madres_data/1945_EPI_DATA.csv")
```

```
########
# clean target data
########

target_small <- target |>
  # if below LOD, use LOD / sqrt(2)
  mutate(conc_mod = ifelse(Comment_code == 37,
                           LOD / sqrt(2),
                           Concentration)) |>
  # adjust for urine specific gravity: Ac = A × [(SGmean -1)/(SG-1)]
  mutate(conc_mod = conc_mod * ((mean(target$SG)-1)/(SG-1))) |>
  select(Project_ID, SID, PID, child_PID, Analyte_Code, conc_mod) |>
  group_by(SID) |>
  mutate(Project_ID = min(Project_ID)) |>
  ungroup() |>
  pivot_wider(names_from = Analyte_Code, values_from = conc_mod) |>
  # howe kept As, Cd, Co, Hg, Ni, Tl, and Pb in main, Mo, Sb, and Sn in supp
  # don't have modified version of As used in their paper
  select(Project_ID, SID, PID, child_PID,
         As, Cd, Co, Hg, Ni, Tl, Pb, Mo, Sb, Sn)

# save
write_csv(target_small, "madres_data/target_small.csv")

# only keep data from first trimester
target_first <- target_small |>
  group_by(child_PID) |>
  filter(Project_ID == min(Project_ID)) |>
  ungroup()

# save
write_csv(target_first, "madres_data/target_first.csv")
```

```
########
# clean epi data
########

# select relevant variables
```

```
epi_small <- epi |>
  # make new categorical variables
  mutate(mom_site = as.factor(mom_site),
         race = as.factor(case_when(
           t1_demo_hispanic == 0 & t1_demo_race == 2 ~ 1, #non-hisp white
           t1_demo_hispanic == 0 & t1_demo_race == 4 ~ 2, #non-hisp black
           t1_demo_hispanic == 0 ~ 3, #other, non-hispanic
           t1_demo_hispanic == 1 & t1_demo_usa == 1 ~ 4, #hispanic in US
           t1_demo_hispanic == 1 & t1_demo_usa == 0 ~ 5, #hispanic NOT in US
           .default = NA
         )),
         smoke = as.factor(ifelse(
           t1_smoke_preg == 1 | t2_smoke_preg == 1 | t3_smoke_preg == 1 |
             t1_smoke == 1 | t2_smoke == 1 | t3_smoke == 1, 1, 0
         ))) |>
  # replace -99 with NA
  mutate(across(where(is.numeric), ~ifelse(. == -99, NA, .)))  |>
  dplyr::select(child_pid, mom_site,
                age = t1_mat_age, # age, trimester 1
                bmi = t1_pre_BMI, # bmi
                race, # maternal r/e
                smoke, # ever-exposure to smoke
                gender, birthweight, GA # birthweight + gestational age
                # can't find anemia measure or AsB
  )

# handle NA values
epi_imp <- epi_small |>
  # exclude birthweight (observed response)
  # exclude study site because of small categories
  select(-c(gender, birthweight, GA, mom_site)) |>
  # na's for smoke during preg, set to 0
  mutate(smoke = as.factor(ifelse(is.na(smoke), 0, smoke))) |>
  # impute mean for BMI
  mutate(across(where(is.numeric),
                ~ifelse(is.na(.), mean(.,na.rm = TRUE), .)))
```

```
########
# combine epi and target data
########
comb <- epi_imp |>
  left_join(target_first, by = c("child_pid" = "child_PID")) |>
  relocate(child_pid, Project_ID, SID, PID, mom_site, race, smoke)

# remove outliers
comb_small <- comb |>
  filter(Mo >=1, Sb <= 1.4)

# save
write_csv(comb_small, "madres_data/base_data.csv")
```

B.2.2 Simulating predictor data

Next, we use copulas to simulate predictor data, as described in Chapter 4.2.2. We use the `copula` and `rslurm` packages in this section. This code was run on FrostByte, the Amherst high-performance computing cluster (HPC) RStudio server.

```r
# load packages
library(tidyverse)
library(copula)
library(rslurm)
```

```r
# read data back in
comb_small <- read_csv("madres_data/base_data.csv")

# log-transform target data
comb_log <- comb_small |>
  mutate(across(10:19, log)) |>
  # factors back to numeric
  mutate(across(where(is.factor), as.numeric))

# check spearman's rho
cor(comb_log[, 7:19], method = "spearman")
```

```r
########
# fit copulas
########

# create pseudo observations for continuous variables
u <- pobs(comb_log[, 7:19])

# fit checkerboard copula on smoke
prop_smoke0 <- 1 - mean(comb_log$smoke)
# jitter 0's and 1's uniformly within quantile
set.seed(0)
u_smoke <- comb_log$smoke |>
  map_dbl(\(x) {
    ifelse(x == 0, runif(1, 0, prop_smoke0), runif(1, prop_smoke0, 1))
  })
u[, 1] <- u_smoke

# fit copulas
cfit_gaus <- fitCopula(normalCopula(dim = 13, dispstr = "un"), u)
cfit_t1 <- fitCopula(tCopula(dim = 13, dispstr = "un",
                             df.fixed = FALSE), u)
cfit_t2 <- fitCopula(tCopula(dim = 13, dispstr = "un",
                             df = 4, df.fixed = TRUE), u)
cfit_t3 <- fitCopula(tCopula(dim = 13, dispstr = "un",
                             df = 10, df.fixed = TRUE), u)
cfit_gum1 <- fitCopula(gumbelCopula(4, dim = 13), u)
cfit_gum2 <- fitCopula(gumbelCopula(2, dim = 13), u)
cfit_frank1 <- fitCopula(frankCopula(4, dim = 13), u)
cfit_frank2 <- fitCopula(frankCopula(2, dim = 13), u)
cfit_clay1 <- fitCopula(claytonCopula(4, dim = 13), u)
cfit_clay2 <- fitCopula(claytonCopula(2, dim = 13), u)
cfit_joe1 <- fitCopula(joeCopula(4, dim = 13), u)
cfit_joe2 <- fitCopula(joeCopula(2, dim = 13), u)

# evaluate fit using AIC
aic_values <- sapply(list(cfit_gaus, cfit_t1, cfit_t2, cfit_t3,
                          cfit_gum1, cfit_gum2, cfit_frank1, cfit_frank2,
                          cfit_clay1, cfit_clay2, cfit_joe1, cfit_joe2
                          ), AIC)
names(aic_values) <- c("cfit_gaus", "cfit_t1", "cfit_t2", "cfit_t3",
                       "cfit_gum", "cfit_gum2", "cfit_frank1", "cfit_frank2",
                       "cfit_clay1", "cfit_clay2", "cfit_joe1", "cfit_joe2")
sort(aic_values)

# evaluate fit using likelihood
```

```r
aic_values <- sapply(list(cfit_gaus, cfit_t1, cfit_t2, cfit_t3,
                          cfit_gum1, cfit_gum2, cfit_frank1, cfit_frank2,
                          cfit_clay1, cfit_clay2, cfit_joe1, cfit_joe2
                          ), logLik)
names(lik_values) <- c("cfit_gaus", "cfit_t1", "cfit_t2", "cfit_t3",
                       "cfit_gum", "cfit_gum2", "cfit_frank1", "cfit_frank2",
                       "cfit_clay1", "cfit_clay2", "cfit_joe1", "cfit_joe2")
sort(lik_values)

# guassian copula performs best, proceed with this
write_rds(cfit_gaus, "sim/gauscop.RDS")
```

```r
########
# simulate predictor data
########

# read copula back in
cfit_gaus <- read_rds("sim/gauscop.RDS")

# extract rho
rho <- coef(cfit_gaus)

# create function for simulation
simulate_data <- function(data, n, rho, prop_smoke, prop_race) {
  #' data = original observed data
  #' n = sample size
  #' rho = rho values from normal copula
  #' prop_smoke = proportion smoke from observed dataset
  #' prop_race = table with race/eth values

  # simulate pseudo-observations from copula
  samp <- rCopula(n, normalCopula(rho, dim = ncol(data), dispstr = "un"))

  # transform pseudo-observations to observed marginal distributions
  sampt <- 1:ncol(data) |>
    purrr::map_dfc(
      \(x) {
        if(names(data)[x] == "smoke") {
          # use observed probability threshold for smoke
          df <- data.frame(ifelse(samp[,x] < prop_smoke, 0, 1),
                           row.names = NULL)
        } else {
          # use empirical marginal CDF's for continuous
          df <- data.frame(quantile(data[[x]], probs = samp[,x]),
                           row.names = NULL)
        }
        names(df) <- names(data)[x]
        return(df)
      }
    ) |>
    # randomly sample race
    mutate(race = sample(x = names(prop_race), prob = prop_race,
                         size = n, replace = T)) |>
    relocate(race)
  return(sampt)
}

# create function to run size 252 samples on hpc
run_sim1 <- function() {
  set.seed(0)
  out <- 1:2100 |>
    purrr::map(\(x) {
      mutate(simulate_data(comb_log_clip, n = nrow(comb_log_clip), rho = rho,
```

```
                          prop_smoke = 1-mean(comb_log_clip$smoke),
                          prop_race = table(comb_log$race)),
             race = as.numeric(race),
             sim = x)
    })
  return(out)
}

# send job to hpc for size 252 samples
sjob1 <- slurm_call(run_sim1,
                    global_objects = c('comb_log', 'comb_log_clip',
                                       'rho', 'simulate_data'),
                    jobname = 'sim_data1')

# get output
out1 <- get_slurm_out(sjob1)
write_rds(out1, "sim/sim_preds_sm.RDS")

# create function to run size 1000 samples on hpc
run_sim2 <- function() {
  set.seed(1)
  out <- 1:2100 |>
    purrr::map(\(x) {
      mutate(simulate_data(comb_log_clip, n = 1000, rho = rho,
                          prop_smoke = 1-mean(comb_log_clip$smoke),
                          prop_race = table(comb_log$race)),
             race = as.numeric(race),
             sim = x)
    })
  return(out)
}

# send job to hpc for size 1000 samples
sjob2 <- slurm_call(run_sim2,
                    global_objects = c('comb_log', 'comb_log_clip',
                                       'rho', 'simulate_data'),
                    jobname = 'sim_data2')

# get output
out2 <- get_slurm_out(sjob2)
write_rds(out2, "sim/sim_preds_lg.RDS")
```

Here, we visualize the observed and simulated predictor data.

```
# load packages
library(tidyverse)
library(latex2exp) # for printing latex on ggplot

# set theme for plots
theme_set(theme_light())
theme_update(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
theme_update(
  strip.background = element_rect(color="gray", fill="white"),
  strip.text = element_text(color = "gray30")
)
```

```
########
# observed data
########
```

```r
# read target data back in
target_first <- read_csv("madres_data/target_first.csv")

# create spearman's correlation matrix
cor_mat <- cor(target_first[, 5:14], method = "spearman")
cor_mat[lower.tri(cor_mat)] <- NA

# reshape correlation matrix to longer format
melt_cor <- reshape2::melt(cor_mat) |>
  mutate(label = ifelse(value == 1, NA, round(value, 2))) |>
  na.omit()

# create correlation heatmap
cor_orig <- melt_cor |>
  ggplot(aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  geom_text(aes(label = label), size = 3.5) +
  scale_fill_gradient2(
    limit = c(-0.6, 0.6), breaks = c(-0.6, -0.3, 0, 0.3, 0.6),
    low = "deepskyblue3", mid = "white", high = "darkorange",
    na.value = NA) +
  coord_fixed() +
  labs(x = NULL, y = NULL, fill = TeX(r"( Spearman's $\rho$ )")) +
  theme(
    panel.grid.major.x = element_line(color = "grey85",
                                      linewidth = 0.25,
                                      linetype = 2),
    panel.border = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.9, 0.1),
    legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))
cor_orig

ggsave("index/figures/ch4_corr.png", width = 5, height = 5)
```

```r
# read target and epi data back in
comb_small <- read_csv("madres_data/base_data.csv")

# log-transform target data
comb_log <- comb_small |>
  mutate(across(10:19, log)) |>
  # factors back to numeric
  mutate(across(where(is.factor), as.numeric))

# look at densities of exposures before log-transform
univ1 <- comb_small |>
  select(10:19) |>
  pivot_longer(cols = 1:10) |>
  mutate(name = factor(name, levels = names(comb_small)[10:19])) |>
  ggplot(aes(x = value)) +
  geom_density() +
  facet_wrap(~name, scales = "free", nrow = 2) +
  labs(x = "Concentration (ng/mL)")
# look at densities of exposures after log-transform
univ2 <- comb_log |>
  select(10:19) |>
  pivot_longer(cols = 1:10) |>
  mutate(name = factor(name, levels = names(comb_small)[10:19])) |>
  ggplot(aes(x = value)) +
  geom_density() +
  facet_wrap(~name, scales = "free", nrow = 2) +
```

```
  labs(x = "Natural log concentration (log(ng/mL))")

# plot in grid and save
cowplot::plot_grid(univ1, univ2, labels = "auto", nrow = 2)
ggsave("index/figures/ch4_univlog.png", width = 7.5, height = 5)
```

```
# density plot of continuous covariates
cov_cont <- comb_log |>
  select(age, bmi) |>
  pivot_longer(cols = 1:2) |>
  ggplot(aes(x = value)) +
  geom_density() +
  facet_wrap(~name, scales = "free", ncol = 1)

# create new dataset for dist. of categorical covariates
df_forcovcat <- comb_log |>
  select(smoke, race) |>
  mutate(smoke = ifelse(smoke == 0, "Never-exposed", "Ever-exposed"),
         race = case_when(
           race == 1 ~ "Non-Hisp. white",
           race == 2 ~ "Non-Hisp. black",
           race == 3 ~ "Non-Hisp. other",
           race == 4 ~ "Hispanic born\nin US",
           race == 5 ~ "Hispanic born\noutside US"
         )) |>
  pivot_longer(cols = 1:2) |>
  mutate(value = factor(
    value, levels = rev(c("Never-exposed", "Ever-exposed",
                          "Non-Hisp. white", "Non-Hisp. black", "Non-Hisp. other",
                          "Hispanic born\nin US", "Hispanic born\noutside US"))
  ))

# bar plot of categorical covariates
cov_cat <- df_forcovcat |>
  ggplot(aes(x = value)) +
  geom_bar(stat = "count", fill = "gray") +
  geom_text(aes(label = after_stat(count)), stat = "count",
            size = 3, hjust = 1, nudge_y = -2) +
  facet_wrap(~name, scales = "free", ncol = 1) +
  coord_flip() +
  labs(x = NULL)

# plot and save
cowplot::plot_grid(cov_cont, cov_cat, labels = "auto", nrow = 1,
                   rel_widths = c(0.4, 0.6))
ggsave("index/figures/ch4_covdist.png", width = 6, height = 4)
```

```
#look at association between race and chemicals
name_order <- c("As", "Cd", "Co", "Hg", "Ni", "Tl", "Pb", "Mo", "Sb", "Sn")
comb_log |>
  select(c(6, 10:19)) |>
  pivot_longer(cols = 2:11, names_to = "key", values_to = "value") |>
  mutate(key = factor(key, levels = name_order)) |>
  mutate(race = as.factor(race)) |>
  ggplot(aes(x = race, y = value, color = race)) +
  geom_boxplot() +
  scale_color_discrete(
    name = "Race by ethnicity\nand birth place",
    labels = c("Non-Hisp. white", "Non-Hisp. black", "Non-Hisp. other",
               "Hispanic born\nin US", "Hispanic born\noutside US")) +
  theme(legend.spacing.y = unit(0.25, 'cm')) +
  guides(color = guide_legend(byrow = TRUE)) +
```

```
  labs(x = "Race, coded", y = "Log concentration") +
  facet_wrap(~key, scales = "free_y")

# save
ggsave("index/figures/ch4_race_exp.png", width = 7, height = 3.5)
```

```
########
# look at simulated data, smaller size
########

# read smaller size simulation back in
out1 <- read_rds("sim/sim_preds_sm.RDS")
comb_sim1 <- bind_rows(out1)

# density plots for exposures
name_order <- c("As", "Cd", "Co", "Hg", "Ni", "Tl", "Pb", "Mo", "Sb", "Sn")
comb_sim1 |>
  mutate(sim = as.factor(sim)) |>
  select(5:15) |>
  pivot_longer(cols = 1:10) |>
  mutate(name = factor(name, levels = name_order)) |>
  ggplot(aes(x = value, group = sim)) +
  geom_line(stat = "density", color = "grey10", alpha = 0.01) +
  # reference density from observed data
  geom_density(
    data = comb_log |> select(10:19) |> pivot_longer(cols = 1:10) |>
      mutate(name = factor(name, levels = name_order)),
    mapping = aes(x = value),
    color = "deepskyblue", linewidth = 0.75, inherit.aes = FALSE
  ) +
  facet_wrap(~name, scales = "free")
# save
ggsave("index/figures/ch4_univ_exp_sim.png", width = 6, height = 4)
```

```
# density plot for continuous covariates
cov_sim_p <- comb_sim1 |>
  mutate(sim = as.factor(sim)) |>
  select(age, bmi, sim) |>
  pivot_longer(cols = 1:2) |>
  ggplot(aes(x = value, group = sim)) +
  geom_line(stat = "density", color = "grey10", alpha = 0.01) +
  geom_density(
    data = comb_log |> select(age, bmi) |> pivot_longer(cols = 1:2),
    mapping = aes(x = value),
    color = "deepskyblue", linewidth = 0.75, inherit.aes = FALSE
  ) +
  facet_wrap(~name, scales = "free", ncol = 1)

# bar + violin plot for categorical covariates
cov_sim_q <- comb_sim1 |>
  mutate(sim = as.factor(sim)) |>
  select(sim, smoke, race) |>
  mutate(smoke = ifelse(smoke == 0, "Never-exposed", "Ever-exposed"),
         race = case_when(
           race == 1 ~ "Non-Hisp. white",
           race == 2 ~ "Non-Hisp. black",
           race == 3 ~ "Non-Hisp. other",
           race == 4 ~ "Hispanic born\nin US",
           race == 5 ~ "Hispanic born\noutside US"
         )) |>
  pivot_longer(cols = 2:3) |>
  group_by(sim, name, value) |>
```

```
  summarize(count = n()) |>
  mutate(value = factor(
    value, levels = rev(c("Never-exposed", "Ever-exposed",
                          "Non-Hisp. white", "Non-Hisp. black", "Non-Hisp. other",
                          "Hispanic born\nin US", "Hispanic born\noutside US"))
  )) |>
  ggplot(aes(x = value, y = count)) +
  geom_bar(data = df_forcovcat, aes(x = value), inherit.aes = FALSE,
           stat = "count", fill = "skyblue") +
  geom_violin(color = "gray30", fill = "gray", alpha = 0.25) +
  facet_wrap(~name, scales = "free", ncol = 1) +
  coord_flip() +
  labs(x = NULL)

# plot in grid and save
cowplot::plot_grid(cov_sim_p, cov_sim_q, labels = "auto", nrow = 1,
                   rel_widths = c(0.4, 0.6))

ggsave("index/figures/ch4_univ_cov_sim.png", width = 6, height = 4)
```

```
# look at correlation structure

# extract correlation structure from simulated data
cors <- out1 |>
  purrr::map_df(\(x) {
    cor_mat <- cor(x[, 5:14], method = "spearman")
    cor_mat[lower.tri(cor_mat)] <- NA
    melt_cor <- reshape2::melt(cor_mat) |>
      mutate(value = ifelse(value == 1, NA, value)) |>
      na.omit() |>
      mutate(sim = x$sim[1])
    return(melt_cor)
  })

# correlation heatmap of average correlation in simulated data
cor_sim <- cors |>
  group_by(Var1, Var2) |>
  summarize(value = mean(value)) |>
  mutate(label = round(value, 2)) |>
  ggplot(aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  geom_text(aes(label = label), size = 3.5) +
  scale_fill_gradient2(
    limit = c(-0.6, 0.6), breaks = c(-0.6, -0.3, 0, 0.3, 0.6),
    low = "deepskyblue3", mid = "white", high = "darkorange",
    na.value = NA) +
  coord_fixed() +
  labs(x = NULL, y = NULL, fill = TeX(r"( Mean Spearman's $\rho$ )")) +
  theme(
    panel.grid.major.x = element_line(color = "grey85",
                                      linewidth = 0.25,
                                      linetype = 2),
    panel.border = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.9, 0.1),
    legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))

# plot and save
cor_sim
ggsave("index/figures/ch4_corr_avg_sim.png", width = 5, height = 5)
```

```
# put original and simulated correlation heatmaps together
top_row <- cowplot::plot_grid(cor_orig, cor_sim, labels = "auto", label_size = 16,
                   nrow = 1, scale = 0.95)
top_row
ggsave("index/figures/ch4_corr_sim+orig.png", width = 10, height = 5)
```

B.2.3 Simulating response data

Next, we simulate the response data, as described in Chapter 4.2.3. We use the `rslurm` package in this section. This code was run on the Amherst HPC RStudio server.

```
# load packages
library(tidyverse)
library(rslurm)
```

```
###############
# create functions for various response variables
###############

# base case, no interactions
base_case <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5,
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
           ifelse(smoke == 1, -1, 0.5) +
           rnorm(nrow(df), 0, 5))
}

# HgxNi, mult, small
am1 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           0.35*Hg*Ni +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5,
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
           ifelse(smoke == 1, -1, 0.5) +
           rnorm(nrow(df), 0, 5))
}

# HgxNi, mult, large
am2 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           0.7*Hg*Ni +
           age + 0.5*bmi +
```

```
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# HgxNi, poly, small
ap1 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.13*Hg*((Ni-1)^2) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# HgxNi, poly, large
ap2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.26*Hg*((Ni-1)^2) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# CdxAs, mult, small
bm1 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.35*Cd*As +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# CdxAs, mult, large
bm2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.7*Cd*As +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
```

```
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# CdxAs, poly, small
bp1 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.125*Cd*((As-1)^2) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# CdxAs, poly, large
bp2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.25*Cd*((As-1)^2) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# HgxCo, mult, small
cm1 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.3*Hg*Co +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# HgxCo, mult, large
cm2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.6*Hg*Co +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}
```

```
# HgxCo, poly, small
cp1 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           0.15*Hg*((Co-1)^2) +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5,
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
           ifelse(smoke == 1, -1, 0.5) +
           rnorm(nrow(df), 0, 5))
}

# HgxCo, poly, large
cp2 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           0.3*Hg*((Co-1)^2) +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5,
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
           ifelse(smoke == 1, -1, 0.5) +
           rnorm(nrow(df), 0, 5))
}

# three-way, multi, small
dm1 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           0.3*Hg*Ni*Tl +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5,
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
           ifelse(smoke == 1, -1, 0.5) +
           rnorm(nrow(df), 0, 5))
}

# three-way, multi, large
dm2 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           0.6*Hg*Ni*Tl +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5,
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
           ifelse(smoke == 1, -1, 0.5) +
           rnorm(nrow(df), 0, 5))
}

# three-way, poly, small
dp1 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
```

```
          0.09*Hg*((Ni-1)^2)*Tl +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# three-way, poly, large
dp2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          0.18*Hg*((Ni-1)^2)*Tl +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}
```

```
##############
# create response variables for exposure-exposure interxn
##############

# read output back in, size 252
out1 <- read_rds("sim/sim_preds_sm.RDS")

# create function for responses at size 252
run_resp1 <- function() {
  set.seed(0)
  out1_resp1 <- out1 |>
    purrr::map(\(x) {
      # get dataset number
      no <- x$sim[1]
      x <- x |>
        # scale log-transformed exposures and covariates
        mutate(across(age:Sn, ~c(scale(.))))
      df <- case_when(
        no <= 100 ~ base_case(x),
        no <= 200 ~ am1(x),
        no <= 300 ~ am2(x),
        no <= 400 ~ ap1(x),
        no <= 500 ~ ap2(x),
        no <= 600 ~ bm1(x),
        no <= 700 ~ bm2(x),
        no <= 800 ~ bp1(x),
        no <= 900 ~ bp2(x),
        no <= 1000 ~ cm1(x),
        no <= 1100 ~ cm2(x),
        no <= 1200 ~ cp1(x),
        no <= 1300 ~ cp2(x),
        no <= 1400 ~ dm1(x),
        no <= 1500 ~ dm2(x),
        no <= 1600 ~ dp1(x),
        no <= 1700 ~ dp2(x),
        .default = x #note 1701 - 2100 is for cov-exp interxn
      )
```

```
    }) |>
    purrr::set_names(nm = c(
      rep("_base", 100),
      rep("am1", 100),
      rep("am2", 100),
      rep("ap1", 100),
      rep("ap2", 100),
      rep("bm1", 100),
      rep("bm2", 100),
      rep("bp1", 100),
      rep("bp2", 100),
      rep("cm1", 100),
      rep("cm2", 100),
      rep("cp1", 100),
      rep("cp2", 100),
      rep("dm1", 100),
      rep("dm2", 100),
      rep("dp1", 100),
      rep("dp2", 100),
      rep("unset", 400)
    ))
  return(out1_resp1)
}

# run to hpc
runrespsm <- slurm_call(
  run_resp1,
  global_objects = c('out1', 'base_case',
                     'am1', 'am2', 'ap1', 'ap2',
                     'bm1', 'bm2', 'bp1', 'bp2',
                     'cm1', 'cm2', 'cp1', 'cp2',
                     'dm1', 'dm2', 'dp1', 'dp2'),
  jobname = 'sim_resp1')

# get output
out1_resp1 <- get_slurm_out(runrespsm)
# only save for exp-exp interxns
out1_resp1 <- out1_resp1[1:1700]
write_rds(out1_resp1, "sim/sim_resp_sm_a.RDS")

# read output back in, size 1000
out2 <- read_rds("sim/sim_preds_lg.RDS")

# create function for response at size 1000
run_resp2 <- function() {
  set.seed(0)
  out2_resp1 <- out2 |>
    purrr::map(\(x) {
      # get dataset number
      no <- x$sim[1]
      x <- x |>
        mutate(across(age:Sn, ~c(scale(.))))
      df <- case_when(
        no <= 100 ~ base_case(x),
        no <= 200 ~ am1(x),
        no <= 300 ~ am2(x),
        no <= 400 ~ ap1(x),
        no <= 500 ~ ap2(x),
        no <= 600 ~ bm1(x),
        no <= 700 ~ bm2(x),
        no <= 800 ~ bp1(x),
        no <= 900 ~ bp2(x),
        no <= 1000 ~ cm1(x),
        no <= 1100 ~ cm2(x),
```

```
        no <= 1200 ~ cp1(x),
        no <= 1300 ~ cp2(x),
        no <= 1400 ~ dm1(x),
        no <= 1500 ~ dm2(x),
        no <= 1600 ~ dp1(x),
        no <= 1700 ~ dp2(x),
        .default = x #note 1701 - 2100 is for cov-exp interxn
      )
    }) |>
    purrr::set_names(nm = c(
      rep("_base", 100),
      rep("am1", 100),
      rep("am2", 100),
      rep("ap1", 100),
      rep("ap2", 100),
      rep("bm1", 100),
      rep("bm2", 100),
      rep("bp1", 100),
      rep("bp2", 100),
      rep("cm1", 100),
      rep("cm2", 100),
      rep("cp1", 100),
      rep("cp2", 100),
      rep("dm1", 100),
      rep("dm2", 100),
      rep("dp1", 100),
      rep("dp2", 100),
      rep("unset", 400)
    ))
  return(out2_resp1)
}

# send to HPC
runresplg <- slurm_call(
  run_resp2,
  global_objects = c('out2', 'base_case',
                     'am1', 'am2', 'ap1', 'ap2',
                     'bm1', 'bm2', 'bp1', 'bp2',
                     'cm1', 'cm2', 'cp1', 'cp2',
                     'dm1', 'dm2', 'dp1', 'dp2'),
  jobname = 'sim_resp2')

# get output
out2_resp1 <- get_slurm_out(runresplg)
# only save output for exp-exp interxns for now
out2_resp1 <- out2_resp1[1:1700]
write_rds(out2_resp1, "sim/sim_resp_lg_a.RDS")
```

```
#############
# create response variables for exposure-covariate interxn
#############

# functions for creating response
# interxn in smaller group, smaller effect size
em1 <- function(df) {
  mutate(df, y =
           Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
           age + 0.5*bmi +
           case_when(race == 1 ~ 1,
                     race == 2 ~ 1.5 + 0.5*Hg, # 1.5x in group 2
                     race == 3 ~ 1,
                     race == 4 ~ 1,
                     race == 5 ~ 1.5) +
```

```
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# interxn in smaller group, larger effect size
em2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5 + Hg, # double in group 2
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5) +
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# interxn in larger group, smaller effect size
ep1 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5 + 0.5*Hg) + # 1.5x in group 5
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# interxn in larger group, larger effect size
ep2 <- function(df) {
  mutate(df, y =
          Hg + 3/(1+exp(-4*Ni)) - (Sb^2) + 0.5*Sb + 1.5/(1+exp(-4*Sn)) +
          age + 0.5*bmi +
          case_when(race == 1 ~ 1,
                    race == 2 ~ 1.5,
                    race == 3 ~ 1,
                    race == 4 ~ 1,
                    race == 5 ~ 1.5 + Hg) + # double in group 5
          ifelse(smoke == 1, -1, 0.5) +
          rnorm(nrow(df), 0, 5))
}

# read output back in, size 252
out1 <- read_rds("sim/sim_preds_sm.RDS")

# create function to simulate response for smaller size
run_resp1_re <- function() {
  set.seed(0)
  out1_resp1 <- out1 |>
    purrr::map(\(x) {
      # get dataset number
      no <- x$sim[1]
      x <- x |>
        mutate(across(age:Sn, ~c(scale(.))))
      df <- case_when(
        no <= 1700 ~ x, #note 1 - 1700 are chemxchem
        no <= 1800 ~ em1(x),
        no <= 1900 ~ em2(x),
        no <= 2000 ~ ep1(x),
        no <= 2100 ~ ep2(x),
```

```
        .default = x
      )
    }) |>
    purrr::set_names(nm = c(
      rep("unset", 1700),
      rep("em1", 100),
      rep("em2", 100),
      rep("ep1", 100),
      rep("ep2", 100)
    ))
  return(out1_resp1)
}

# send to HPC
runrespsm_re <- slurm_call(
  run_resp1_re,
  global_objects = c('out1',
                     'em1', 'em2', 'ep1', 'ep2'),
  jobname = 'sim_resp1_re')

# get output
out1_resp1_re <- get_slurm_out(runrespsm_re)
out1_resp1_re <- out1_resp1_re[1701:2100]
write_rds(out1_resp1_re, "sim/sim_resp_sm_re.RDS")

# read output back in, size 1000
out2 <- read_rds("sim/sim_preds_lg.RDS")

# create function to simulate response for larger size
run_resp2_re <- function() {
  set.seed(0)
  out2_resp1 <- out2 |>
    purrr::map(\(x) {
      # get dataset number
      no <- x$sim[1]
      x <- x |>
        mutate(across(age:Sn, ~c(scale(.))))
      df <- case_when(
        no <= 1700 ~ x, #note 1 - 1700 are chemxchem
        no <= 1800 ~ em1(x),
        no <= 1900 ~ em2(x),
        no <= 2000 ~ ep1(x),
        no <= 2100 ~ ep2(x),
        .default = x
      )
    }) |>
    purrr::set_names(nm = c(
      rep("unset", 1700),
      rep("em1", 100),
      rep("em2", 100),
      rep("ep1", 100),
      rep("ep2", 100)
    ))
  return(out2_resp1)
}

# send to HPC
runresplg_re <- slurm_call(
  run_resp2_re,
  global_objects = c('out2',
                     'em1', 'em2', 'ep1', 'ep2'),
  jobname = 'sim_resp2_re')

# get output
```

```
out2_resp1_re <- get_slurm_out(runresplg_re)
out2_resp1_re <- out2_resp1_re[1701:2100]
write_rds(out2_resp1_re, "sim/sim_resp_lg_re.RDS")
```

B.2.4 Fitting models

Here, we fit the models to our simulated data, as described in Chapter 4.2.4. We use the `rslurm`, `bkmr`, and `NLinteraction` packages in this section. This code was run on the Amherst HPC RStudio server.

```
# load packages
library(tidyverse)
library(bkmr)
library(NLinteraction)
```

First, we fit the naive and oracle MLRs.

```
#########
# naive and oracle MLRs
#########

### smaller size

# read in simulated data
out1_resp1 <- read_rds("sim/sim_resp_sm_a.RDS")

run_mlr_sm <- function() {
  # initialize vectors
  mlrs <- vector(mode='list', length = 1700)
  names(mlrs) <- names(out1_resp1)
  mlrtimes <- vector(mode = 'list', length = 1700)
  names(mlrtimes) <- names(out1_resp1)

  oracles <- vector(mode='list', length = 1700)
  names(oracles) <- names(out1_resp1)
  oracletimes <- vector(mode = 'list', length = 1700)
  names(oracletimes) <- names(out1_resp1)

  for(i in 1:1700) {
    df <- out1_resp1[[i]] |>
      mutate(race = as.factor(race), smoke = as.factor(smoke)) |>
      select(-sim)

    start.time <- Sys.time()
    mlrs[[i]] <- lm(y ~ ., data = df)
    end.time <- Sys.time()
    mlrtimes[[i]] <- end.time - start.time

    if(i <= 100) {
      start.time <- Sys.time()
      oracles[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           age + bmi + race + smoke, data = df)
```

```
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 300) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         Hg*Ni +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 500) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         I(Hg*((Ni-1)^2)) +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 700) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         Cd*As +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 900) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         I(Cd*((As-1)^2)) +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 1100) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         Ni*Co +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 1300) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         I(Ni*((Co-1)^2)) +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 1500) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         Hg:Ni:Tl +
                         age + bmi + race + smoke, data = df)
    end.time <- Sys.time()
    oracletimes[[i]] <- end.time - start.time
  } else if (i <= 1700) {
    start.time <- Sys.time()
    oracles[[i]] <- lm(y ~ Hg + Sb +
                         I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                         I(Hg*((Ni-1)^2)*Tl) +
                         age + bmi + race + smoke, data = df)
```

```
      end.time <- Sys.time()
      oracletimes[[i]] <- end.time - start.time
    }
  }
  return(list(mlrs, mlrtimes, oracles, oracletimes))
}

# send to hpc
sjob5 <- slurm_call(
  run_mlr_sm,
  global_objects = c('out1_resp1'),
  jobname = 'mlr_sm')

# get output
mlr_sm <- get_slurm_out(sjob5)
mlr_mods <- mlr_sm[[1]]
mlr_times <- mlr_sm[[2]]
oracle_mods <- mlr_sm[[3]]
oracle_times <- mlr_sm[[4]]
write_rds(mlr_mods, "sim/mlr_mods_sm.RDS")
write_rds(mlr_times, "sim/mlr_mods_sm_times.RDS")
write_rds(oracle_mods, "sim/oracle_mods_sm.RDS")
write_rds(oracle_times, "sim/oracle_mods_sm_times.RDS")

### larger sample size

# read in simulated data
out2_resp1 <- read_rds("sim/sim_resp_lg_a.RDS")

run_mlr_lg <- function() {
  # initialize vectors
  mlrl <- vector(mode='list', length = 1700)
  names(mlrl) <- names(out2_resp1)
  mlrtimel <- vector(mode = 'list', length = 1700)
  names(mlrtimel) <- names(out2_resp1)

  oraclel <- vector(mode='list', length = 1700)
  names(oraclel) <- names(out2_resp1)
  oracletimel <- vector(mode = 'list', length = 1700)
  names(oracletimel) <- names(out2_resp1)

  for(i in 1:1700) {
    df <- out2_resp1[[i]] |>
      mutate(race = as.factor(race), smoke = as.factor(smoke)) |>
      select(-sim)

    start.time <- Sys.time()
    mlrl[[i]] <- lm(y ~ ., data = df)
    end.time <- Sys.time()
    mlrtimel[[i]] <- end.time - start.time

    if(i <= 100) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 300) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           Hg*Ni +
                           age + bmi + race + smoke, data = df)
```

```
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 500) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           I(Hg*((Ni-1)^2)) +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 700) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           Cd*As +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 900) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           I(Cd*((As-1)^2)) +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 1100) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           Ni*Co +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 1300) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           I(Ni*((Co-1)^2)) +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 1500) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           Hg:Ni:Tl +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    } else if (i <= 1700) {
      start.time <- Sys.time()
      oraclel[[i]] <- lm(y ~ Hg + Sb +
                           I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                           I(Hg*((Ni-1)^2)*Tl) +
                           age + bmi + race + smoke, data = df)
      end.time <- Sys.time()
      oracletimel[[i]] <- end.time - start.time
    }
  }
  return(list(mlrl, mlrtimel, oraclel, oracletimel))
}

# send to hpc
```

```
sjob6 <- slurm_call(
  run_mlr_lg,
  global_objects = c('out2_resp1'),
  jobname = 'mlr_lg')

# get output
mlr_lg <- get_slurm_out(sjob6)
mlr_modl <- mlr_lg[[1]]
mlr_timel <- mlr_lg[[2]]
oracle_modl <- mlr_lg[[3]]
oracle_timel <- mlr_lg[[4]]
write_rds(mlr_modl, "sim/mlr_mods_lg.RDS")
write_rds(mlr_timel, "sim/mlr_mods_lg_times.RDS")
write_rds(oracle_modl, "sim/oracle_mods_lg.RDS")
write_rds(oracle_timel, "sim/oracle_mods_lg_times.RDS")
```

Next, we fit BKMR and BSR models with the base case and with interactions between chemicals.

```
#########
# run bkmr
#########

### smaller sample size

out1_resp1 <- read_rds("sim/sim_resp_sm_a.RDS")

run_bkmr_sm <- function(vector) {
  # initiate vector of times
  bkmr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  # for each simulated dataset...
  for(i in vector) {
    print(paste0("------------- run ", i, "--------------"))

    # prepare data
    df <- out1_resp1[[i]]
    Z <- df |>
      select(As:Sn)
    X <- df |>
      bind_cols(
        data.frame(model.matrix(~ race-1, data =
                                  mutate(df, race = as.factor(race))))
      ) |>
      select(race2:race5, smoke:bmi)
    y <- df$y

    # fit model and save time
    set.seed(0)
    start.time <- Sys.time()
    mod <- kmbayes(y = y, Z = Z, X = X,
                           iter = 50000, verbose = FALSE, varsel = TRUE)
```

```
    end.time <- Sys.time()
    bkmr_times[[i]] <- end.time - start.time

    # save model and remove from memory
    write_rds(mod, file = paste0("mods/bkmr_sm_", names(out1_resp1)[i], "_", i, ".RDS"))
    write_rds(bkmr_times[[i]], file =
                paste0("times/bkmr_sm_", names(out1_resp1)[i], "_", i, ".RDS"))
    rm(mod)
  }
  # save times
  write_rds(bkmr_times, file = "bkmr_sm_times.RDS")
}

# send all simulated datasets to hpc
for(i in seq(1, 1601, by = 100)) {
  slurm_call(
    run_bkmr_sm,
    params = list(vector = i:(i+99)),
    global_objects = c('out1_resp1'),
    jobname = paste0("bkmr_sm", str_pad(ceiling(i/100), 2, pad = "0")),
    slurm_options = list(mem = '8G')
  )
}
```

```
### larger sample size
out2_resp1 <- read_rds("sim/sim_resp_lg_a.RDS")

run_bkmr_lg <- function(vector) {
  # initiate vector of times
  bkmr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  # sometimes this code would stop prematurely...
  # so this chunk finds the index of the last model that was run
  list_files <- list.files("mods", full.names = TRUE)
  nums <- as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", list_files))
  print(nums)
  if(length(nums) == 0) {
    final <- 0; starting <- min(vector)
  } else {
    final <- max(nums); starting <- final + 1
  }
  # keep note of when the loop was stopped and re-started
  if(file.exists("final.txt")) {
    write(paste0("final ran = ", final, ", starting at ", starting), "final.txt", append = T)
  } else {
    writeLines(paste0("final ran = ", final, ", starting at ", starting), "final.txt")
  }

  # for each simulated dataset...
  for(i in starting:max(vector)) {
    print(paste0("------------ run ", i, "--------------"))

    # prepare data
    df <- out2_resp1[[i]]
    Z <- df |>
```

```
      select(As:Sn)
    X <- df |>
      bind_cols(
        data.frame(model.matrix(~ race-1, data =
                                  mutate(df, race = as.factor(race))))
      ) |>
      select(race2:race5, smoke:bmi)
    y <- df$y
    knots <- fields::cover.design(Z, nd = 100)$design

    # fit model and save time
    set.seed(0)
    start.time <- Sys.time()
    mod <- kmbayes(y = y, Z = Z, X = X, knots = knots,
                   iter = 50000, verbose = FALSE, varsel = TRUE)
    end.time <- Sys.time()
    bkmr_times[[i]] <- end.time - start.time

    # save model and remove from memory
    write_rds(mod, file = paste0("mods/bkmr_lg_", names(out2_resp1)[i], "_", i, ".RDS"))
    write_rds(bkmr_times[[i]], file =
                paste0("times/bkmr_lg_", names(out2_resp1)[i], "_", i, ".RDS"))
    rm(mod)
  }
  # save times
  write_rds(bkmr_times, file = "bkmr_lg_times.RDS")
}

# send all simulated datasets to hpc
for(i in seq(1, 1601, by = 100)) {
  slurm_call(
    run_bkmr_lg,
    params = list(vector = i:(i+99)),
    global_objects = c('out2_resp1'),
    jobname = paste0("bkmr_lg", str_pad(ceiling(i/100), 2, pad = "0")),
    slurm_options = list(mem = '8G')
  )
}
```

```
########
# run bsr
########

### smaller sample size

out1_resp1 <- read_rds("sim/sim_resp_sm_a.RDS")

run_bsr_sm <- function(vector) {
  bsr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  # sometimes this code would stop prematurely...
  # so this chunk finds the index of the last model that was run
  list_files <- list.files("mods", full.names = TRUE)
  nums <- as.numeric(sub(".+_(.+)d.+", "\\1", list_files))
  if(length(nums) == 0) {
```

```
  final <- 0; starting <- min(vector)
} else {
  final <- max(nums); starting <- final + 1
}
if(file.exists("final.txt")) {
  write(paste0("final ran = ", final, ", starting at ", starting), "final.txt", append = T)
} else {
  writeLines(paste0("final ran = ", final, ", starting at ", starting), "final.txt")
}

# for each simulated dataset...
for(i in starting:max(vector)) {
  print(paste0("------------- run ", i, "--------------"))

  # prepare data
  df <- out1_resp1[[i]]
  X <- df |>
    select(As:Sn) |>
    as.matrix.data.frame()
  C <- df |>
    bind_cols(
      data.frame(model.matrix(~ race-1, data =
                                mutate(df, race = as.factor(race))))
    ) |>
    select(race2:race5, smoke:bmi) |>
    as.matrix.data.frame()
  Y <- df$y

  # fit model for d = {1, 2, 3, 4, 5} and save time
  list_times <- vector(mode = "list", length = 2)

  set.seed(0)
  start.time <- Sys.time()
  mod1 <- NLint(Y = Y, X = X, C = C,
                nIter = 5000, nBurn = 2500, ns = 1)
  mod2 <- NLint(Y = Y, X = X, C = C,
                nIter = 5000, nBurn = 2500, ns = 2)
  mod3 <- NLint(Y = Y, X = X, C = C,
                nIter = 5000, nBurn = 2500, ns = 3)
  mod4 <- NLint(Y = Y, X = X, C = C,
                nIter = 5000, nBurn = 2500, ns = 4)
  end.time <- Sys.time()
  list_times[[1]] <- end.time - start.time

  ind <- which.min(c(mod1$waic, mod2$waic, mod3$waic, mod4$waic))

  print(paste0("-------- chose ", ind, " -----------"))

  # fit with selected d from waic
  start.time <- Sys.time()
  mod <- NLint(Y = Y, X = X, C = C,
               nIter = 50000, nBurn = 25000, ns = ind)
  end.time <- Sys.time()
  list_times[[2]] <- end.time - start.time

  bsr_times[[i]] <- list_times

  # save model and remove from memory
  write_rds(mod, file =
              paste0("mods/bsr_sm_", names(out1_resp1)[i], "_", i,
                     "df", ind, ".RDS"))
  write_rds(list_times, file =
               paste0("times/bsr_sm_", names(out1_resp1)[i], "_", i,
                      "df", ind, ".RDS"))
```

```
    rm(mod1, mod2, mod3, mod4, mod)
  }

  write_rds(bsr_times, file = "bsr_smf_times.RDS")
  return(bsr_times)
}

# send all simulated datasets to hpc
for(i in seq(1, 1601, by = 100)) {
  slurm_call(
    run_bsr_lg,
    params = list(vector = i:(i+99)),
    global_objects = c('out1_resp1'),
    jobname = paste0("bsr_sm", str_pad(ceiling(i/100), 2, pad = "0")),
    slurm_options = list(mem = '8G')
  )
}
```

```
### larger sample size

out2_resp1 <- read_rds("sim/sim_resp_lg_a.RDS")

run_bsr_lg <- function(vector) {
  bsr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  # sometimes this code would stop prematurely...
  # so this chunk finds the index of the last model that was run
  list_files <- list.files("mods", full.names = TRUE)
  nums <- as.numeric(sub(".+_(.+)d.+", "\\1", list_files))
  if(length(nums) == 0) {
    final <- 0; starting <- min(vector)
  } else {
    final <- max(nums); starting <- final + 1
  }
  if(file.exists("final.txt")) {
    write(paste0("final ran = ", final, ", starting at ", starting), "final.txt", append = T)
  } else {
    writeLines(paste0("final ran = ", final, ", starting at ", starting), "final.txt")
  }

  for(i in starting:max(vector)) {
    print(paste0("------------- run ", i, "--------------"))

    # prepare data
    df <- out2_resp1[[i]]
    X <- df |>
      select(As:Sn) |>
      as.matrix.data.frame()
    C <- df |>
      bind_cols(
        data.frame(model.matrix(~ race-1, data =
                                  mutate(df, race = as.factor(race))))
      ) |>
      select(race2:race5, smoke:bmi) |>
      as.matrix.data.frame()
```

```
    Y <- df$y

    # fit model for d = {1, 2, 3, 4, 5} and save time
    list_times <- vector(mode = "list", length = 2)

    set.seed(0)
    start.time <- Sys.time()
    mod1 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 1)
    mod2 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 2)
    mod3 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 3)
    mod4 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 4)
    end.time <- Sys.time()
    list_times[[1]] <- end.time - start.time

    ind <- which.min(c(mod1$waic, mod2$waic, mod3$waic, mod4$waic))

    print(paste0("-------- chose ", ind, " -----------"))

    start.time <- Sys.time()
    mod <- NLint(Y = Y, X = X, C = C,
                 nIter = 50000, nBurn = 25000, ns = ind)
    end.time <- Sys.time()
    list_times[[2]] <- end.time - start.time

    bsr_times[[i]] <- list_times

    # save model and remove from memory
    write_rds(mod, file =
                paste0("mods/bsr_lgf_", names(out2_resp1)[i], "_", i,
                       "df", ind, ".RDS"))
    write_rds(list_times, file =
                paste0("times/bsr_lgf_", names(out2_resp1)[i], "_", i,
                       "df", ind, ".RDS"))

    rm(mod1, mod2, mod3, mod4, mod)
  }

  write_rds(bsr_times, file = "bsr_lgf_times.RDS")
  # return(bsr_times)
}

# send all simulated datasets to hpc
for(i in seq(1, 1601, by = 100)) {
  slurm_call(
    run_bsr_sm,
    params = list(vector = i:(i+99)),
    global_objects = c('out2_resp1'),
    jobname = paste0("bsr_lg", str_pad(ceiling(i/100), 2, pad = "0")),
    slurm_options = list(mem = '8G')
  )
}
```

Next, we fit stratified models with an interaction between the categorical race variable and a chemical.

```
##########
# stratified bkmr, smaller size
##########

out1_resp1_re <- read_rds("sim/sim_resp_sm_re.RDS")

run_bkmr_sm_re <- function(vector) {
  bkmr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  for(i in vector) {
    print(paste0("------------- run ", i, "--------------"))

    # prepare data
    df_full <- out1_resp1_re[[i]]

    # for each race level, run bmkr
    list_times <- vector(mod = "list", length = 6)
    list_mods <- vector(mod = "list", length = 6)

    set.seed(0)
    for(j in 1:5) {
      df <- df_full[df_full$race == j, ]
      Z <- df |>
        select(As:Sn)
      X <- df |>
        select(smoke:bmi)
      y <- df$y

      # fit model and save time
      start.time <- Sys.time()
      mod <- tryCatch({
        kmbayes(y = y, Z = Z, X = X,
                iter = 50000, verbose = FALSE, varsel = TRUE)
      }, error = function(e) {
        NA
      })
      end.time <- Sys.time()
      list_times[[j]] <- end.time - start.time
      list_mods[[j]] <- mod
    }

    # combine 1-3 r/e
    df <- df_full[df_full$race %in% c(1, 2, 3), ]
    Z <- df |>
      select(As:Sn)
    X <- df |>
      select(smoke:bmi)
    y <- df$y

    # fit model and save time
    start.time <- Sys.time()
    mod <- tryCatch({
      kmbayes(y = y, Z = Z, X = X,
              iter = 50000, verbose = FALSE, varsel = TRUE)
    }, error = function(e) {
      NA
```

```
    })
    end.time <- Sys.time()
    list_times[[6]] <- end.time - start.time
    list_mods[[6]] <- mod


    bkmr_times[[i]] <- list_times
    # save model and remove from memory
    write_rds(list_mods, file =
                paste0("mods/bkmr_sm_", names(out1_resp1_re)[i], "_", i, ".RDS"))
    write_rds(list_times, file =
                paste0("times/bkmr_smf_", names(out1_resp1_re)[i], "_", i, ".RDS"))
    rm(mod)
  }
  # write_rds(bkmr_times, file = "bkmr_sm_times.RDS")
  return(bkmr_times)
}

# send to hpc
ujob01 <- slurm_call(
  run_bkmr_sm_re, params = list(vector = 1:100),
  global_objects = c('out1_resp1_re'),
  jobname = 'ksmre01',
  slurm_options = list(mem = '8G'))

ujob02 <- slurm_call(
  run_bkmr_sm_re, params = list(vector = 101:200),
  global_objects = c('out1_resp1_re'),
  jobname = 'ksmre02',
  slurm_options = list(mem = '8G'))

ujob03 <- slurm_call(
  run_bkmr_sm_re, params = list(vector = 201:300),
  global_objects = c('out1_resp1_re'),
  jobname = 'ksmre03',
  slurm_options = list(mem = '8G'))

ujob04 <- slurm_call(
  run_bkmr_sm_re, params = list(vector = 301:400),
  global_objects = c('out1_resp1_re'),
  jobname = 'ksmre04',
  slurm_options = list(mem = '8G'))
```

```
##########
# stratified bkmr, larger size
##########

out2_resp1_re <- read_rds("sim/sim_resp_lg_re.RDS")

run_bkmr_lg_re <- function(vector) {
  bkmr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  for(i in vector) {
    print(paste0("------------ run ", i, "--------------"))
```

```
    # prepare data
    df_full <- out2_resp1_re[[i]]

    # for each race level, run bmkr
    list_times <- vector(mod = "list", length = 5)
    list_mods <- vector(mod = "list", length = 5)

    set.seed(0)
    for(j in 1:5) {
      df <- df_full[df_full$race == j, ]
      Z <- df |>
        select(As:Sn)
      X <- df |>
        select(smoke:bmi)
      y <- df$y

      # fit model and save time
      start.time <- Sys.time()
      mod <- kmbayes(y = y, Z = Z, X = X,
                     iter = 50000, verbose = FALSE, varsel = TRUE)
      end.time <- Sys.time()
      list_times[[j]] <- end.time - start.time
      list_mods[[j]] <- mod
    }

    bkmr_times[[i]] <- list_times
    # save model and remove from memory
    write_rds(list_mods, file =
                paste0("mods/bkmr_lg_", names(out2_resp1_re)[i], "_", i, ".RDS"))
    write_rds(list_times, file =
                paste0("times/bkmr_lgf_", names(out2_resp1_re)[i], "_", i, ".RDS"))
    rm(mod)
  }
  # write_rds(bkmr_times, file = "bkmr_lg_times_re.RDS")
  return(bkmr_times)
}

# send to hpc
tjob01 <- slurm_call(
  run_bkmr_lg_re, params = list(vector = 1:100),
  global_objects = c('out2_resp1_re'),
  jobname = 'klgre01',
  slurm_options = list(mem = '8G'))

tjob02 <- slurm_call(
  run_bkmr_lg_re, params = list(vector = 101:200),
  global_objects = c('out2_resp1_re'),
  jobname = 'klgre02',
  slurm_options = list(mem = '8G'))

tjob03 <- slurm_call(
  run_bkmr_lg_re, params = list(vector = 201:300),
  global_objects = c('out2_resp1_re'),
  jobname = 'klgre03',
  slurm_options = list(mem = '8G'))

tjob04 <- slurm_call(
  run_bkmr_lg_re, params = list(vector = 301:400),
  global_objects = c('out2_resp1_re'),
  jobname = 'klgre04',
  slurm_options = list(mem = '8G'))
```

```
###########
# stratified bsr, smaller size
###########

out1_resp1_re <- read_rds("sim/sim_resp_sm_re.RDS")

run_bsr_sm_re <- function(vector) {
  bsr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  # sometimes bsr would stop prematurely...
  # so this code finds the index of the last model run
  list_files <- list.files("mods", full.names = TRUE)
  nums <- as.numeric(sub(".+_(.+)d.+", "\\1", list_files))
  if(length(nums) == 0) {
    final <- 0
    starting <- min(vector)
  } else {
    final <- max(nums)
    starting <- final + 1
  }
  if(file.exists("final.txt")) {
    write(paste0("final ran = ", final, ", starting at ", starting), "final.txt", append = T)
  } else {
    writeLines(paste0("final ran = ", final, ", starting at ", starting), "final.txt")
  }


  for(i in starting:max(vector)) {
    print(paste0("------------- run ", i, "--------------"))

    # prepare data
    df_full <- out1_resp1_re[[i]]

    # for each race level, run bsr
    list_times <- vector(mod = "list", length = 6)
    list_mods <- vector(mod = "list", length = 6)

    set.seed(0)
    for(j in 1:5) {
      df <- df_full[df_full$race == j, ]
      X <- df |>
        select(As:Sn) |>
        as.matrix.data.frame()
      C <- df |>
        select(smoke:bmi) |>
        as.matrix.data.frame()
      Y <- df$y

      # waic for choosing df
      list_times_small <- vector(mode = "list", length = 2)

      start.time <- Sys.time()
      mod1 <- NLint(Y = Y, X = X, C = C,
                    nIter = 5000, nBurn = 2500, ns = 1)
      mod2 <- NLint(Y = Y, X = X, C = C,
                    nIter = 5000, nBurn = 2500, ns = 2)
```

```
  mod3 <- NLint(Y = Y, X = X, C = C,
                nIter = 5000, nBurn = 2500, ns = 3)
  mod4 <- NLint(Y = Y, X = X, C = C,
                nIter = 5000, nBurn = 2500, ns = 4)
  end.time <- Sys.time()
  list_times_small[[1]] <- end.time - start.time

  ind <- which.min(c(mod1$waic, mod2$waic, mod3$waic, mod4$waic))

  # fit model and save time
  start.time <- Sys.time()
  mod <- tryCatch({
    NLint(Y = Y, X = X, C = C,
          nIter = 50000, nBurn = 25000, ns = ind)
  }, error = function(e) {
    NA
  })
  end.time <- Sys.time()
  list_times_small[[2]] <- end.time - start.time

  bsr_times[[j]] <- list_times_small
  list_mods[[j]] <- mod
}

# combine 1-3 r/e
df <- df_full[df_full$race %in% c(1, 2, 3), ]
X <- df |>
  select(As:Sn) |>
  as.matrix.data.frame()
C <- df |>
  select(smoke:bmi) |>
  as.matrix.data.frame()
Y <- df$y

# waic for choosing df
list_times_small <- vector(mode = "list", length = 2)

start.time <- Sys.time()
mod1 <- NLint(Y = Y, X = X, C = C,
              nIter = 5000, nBurn = 2500, ns = 1)
mod2 <- NLint(Y = Y, X = X, C = C,
              nIter = 5000, nBurn = 2500, ns = 2)
mod3 <- NLint(Y = Y, X = X, C = C,
              nIter = 5000, nBurn = 2500, ns = 3)
mod4 <- NLint(Y = Y, X = X, C = C,
              nIter = 5000, nBurn = 2500, ns = 4)
end.time <- Sys.time()
list_times_small[[1]] <- end.time - start.time

ind <- which.min(c(mod1$waic, mod2$waic, mod3$waic, mod4$waic))

# fit model and save time
start.time <- Sys.time()
mod <- tryCatch({
  NLint(Y = Y, X = X, C = C,
        nIter = 50000, nBurn = 25000, ns = ind)
}, error = function(e) {
  NA
})
end.time <- Sys.time()
list_times_small[[2]] <- end.time - start.time

list_times[[6]] <- list_times_small
list_mods[[6]] <- mod
```

```
    # save model and remove from memory
    write_rds(list_mods, file =
                paste0("mods/bsr_sm_", names(out1_resp1_re)[i], "_", i,
                       "df", ind, ".RDS"))
    write_rds(list_times, file =
                paste0("times/bsr_sm_", names(out1_resp1_re)[i], "_", i,
                       "df", ind, ".RDS"))

    rm(mod1, mod2, mod3, mod4, mod)
  }

  return(bsr_times)
}

# send to hpc
vjob01 <- slurm_call(
  run_bsr_sm_re, params = list(vector = 1:100),
  global_objects = c('out1_resp1_re'),
  jobname = 'ssmre01',
  slurm_options = list(mem = '8G'))

vjob02 <- slurm_call(
  run_bsr_sm_re, params = list(vector = 101:200),
  global_objects = c('out1_resp1_re'),
  jobname = 'ssmre02',
  slurm_options = list(mem = '8G'))

vjob03 <- slurm_call(
  run_bsr_sm_re, params = list(vector = 201:300),
  global_objects = c('out1_resp1_re'),
  jobname = 'ssmre03',
  slurm_options = list(mem = '8G'))

vjob04 <- slurm_call(
  run_bsr_sm_re, params = list(vector = 301:400),
  global_objects = c('out1_resp1_re'),
  jobname = 'ssmre04',
  slurm_options = list(mem = '8G'))
```

```
###########
# stratified bsr, larger size
###########

out2_resp1_re <- read_rds("sim/sim_resp_lg_re.RDS")

run_bsr_lg_re <- function(vector) {
  bsr_times <- vector(mode = "list", length = length(vector))

  # create folder for model output
  if(!dir.exists("mods")) {
    dir.create("mods")
  }
  if(!dir.exists("times")) {
    dir.create("times")
  }

  # sometimes bsr would stop prematurely...
  # so this code finds the index of the last model run
  list_files <- list.files("mods", full.names = TRUE)
  nums <- as.numeric(sub(".+_(.+)d.+", "\\1", list_files))
  if(length(nums) == 0) {
    final <- 0
    starting <- min(vector)
```

```
} else {
  final <- max(nums)
  starting <- final + 1
}
if(file.exists("final.txt")) {
  write(paste0("final ran = ", final, ", starting at ", starting), "final.txt", append = T)
} else {
  writeLines(paste0("final ran = ", final, ", starting at ", starting), "final.txt")
}


for(i in starting:max(vector)) {
  print(paste0("------------- run ", i, "--------------"))

  # prepare data
  df_full <- out2_resp1_re[[i]]

  # for each race level, run bsr
  list_times <- vector(mod = "list", length = 6)
  list_mods <- vector(mod = "list", length = 6)

  set.seed(0)
  for(j in 1:5) {
    df <- df_full[df_full$race == j, ]
    X <- df |>
      select(As:Sn) |>
      as.matrix.data.frame()
    C <- df |>
      select(smoke:bmi) |>
      as.matrix.data.frame()
    Y <- df$y

    # waic for choosing df
    list_times_small <- vector(mode = "list", length = 2)

    start.time <- Sys.time()
    mod1 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 1)
    mod2 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 2)
    mod3 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 3)
    mod4 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 4)
    end.time <- Sys.time()
    list_times_small[[1]] <- end.time - start.time

    ind <- which.min(c(mod1$waic, mod2$waic, mod3$waic, mod4$waic))

    # fit model and save time
    start.time <- Sys.time()
    mod <- tryCatch({
      NLint(Y = Y, X = X, C = C,
            nIter = 50000, nBurn = 25000, ns = ind)
    }, error = function(e) {
      NA
    })
    end.time <- Sys.time()
    list_times_small[[2]] <- end.time - start.time

    bsr_times[[j]] <- list_times_small
    list_mods[[j]] <- mod
  }
```

```
    # combine 1-3 r/e
    df <- df_full[df_full$race %in% c(1, 2, 3), ]
    X <- df |>
      select(As:Sn) |>
      as.matrix.data.frame()
    C <- df |>
      select(smoke:bmi) |>
      as.matrix.data.frame()
    Y <- df$y

    # waic for choosing df
    list_times_small <- vector(mode = "list", length = 2)

    start.time <- Sys.time()
    mod1 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 1)
    mod2 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 2)
    mod3 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 3)
    mod4 <- NLint(Y = Y, X = X, C = C,
                  nIter = 5000, nBurn = 2500, ns = 4)
    end.time <- Sys.time()
    list_times_small[[1]] <- end.time - start.time

    ind <- which.min(c(mod1$waic, mod2$waic, mod3$waic, mod4$waic))

    # fit model and save time
    start.time <- Sys.time()
    mod <- tryCatch({
      NLint(Y = Y, X = X, C = C,
            nIter = 50000, nBurn = 25000, ns = ind)
    }, error = function(e) {
      NA
    })
    end.time <- Sys.time()
    list_times_small[[2]] <- end.time - start.time

    list_times[[6]] <- list_times_small
    list_mods[[6]] <- mod

    # save model and remove from memory
    write_rds(list_mods, file =
                paste0("mods/bsr_lg_", names(out2_resp1_re)[i], "_", i,
                       "df", ind, ".RDS"))
    write_rds(list_times, file =
                paste0("times/bsr_lg_", names(out2_resp1_re)[i], "_", i,
                       "df", ind, ".RDS"))

    rm(mod1, mod2, mod3, mod4, mod)
  }

  # write_rds(bsr_times, file = "bsr_lgf_times.RDS")
  return(bsr_times)
}

# send to hpc
wjob01 <- slurm_call(
  run_bsr_lg_re, params = list(vector = 1:100),
  global_objects = c('out2_resp1_re'),
  jobname = 'slgre01',
  slurm_options = list(mem = '8G'))

wjob02 <- slurm_call(
```

```
  run_bsr_lg_re, params = list(vector = 101:200),
  global_objects = c('out2_resp1_re'),
  jobname = 'slgre02',
  slurm_options = list(mem = '8G'))

wjob03 <- slurm_call(
  run_bsr_lg_re, params = list(vector = 201:300),
  global_objects = c('out2_resp1_re'),
  jobname = 'slgre03',
  slurm_options = list(mem = '8G'))

wjob04 <- slurm_call(
  run_bsr_lg_re, params = list(vector = 301:400),
  global_objects = c('out2_resp1_re'),
  jobname = 'slgre04',
  slurm_options = list(mem = '8G'))
```

B.3 Code for Chapter 4.3 and Appendix A.2

This section includes code for extracting and presenting results in Chapter 4.3 and the supplemental results, Appendix A.2.

B.3.1 Extracting results

Here, we extract results from our simulation. We use the `rslurm`, `bkmr`, and `NLinteraction` packages in this section. This code was run on the Amherst HPC RStudio server.

First, we write custom functions to extract output from BKMR and BSR models. These functions are stored in the `extract_fxns.R` file and loaded in later in analysis.

```
# this function calculates confidence intervals for two-way interactions in BKMR
bivarinter_bkmr <- function(fit, z1, z2, qs.diff = c(0.25, 0.75),
                            qs.fixed = c(0.25, 0.75), q.rest = 0.5) {
  #' fit = bkmr model
  #' z1 = index of chemical 1
  #' z2 = index of chemical 2
  #' qs.diff = quantiles to calculate response diff for
  #' qs.fixed = quantiles to fix other chemical at
  #' q.rest = quantile to fix rest of chemicals at
  #'
  #' NOTE that order of z1 and z2 does not matter

  # extract fit
  y <- fit$y
  Z <- fit$Z
  X <- fit$X
```

```
  # fix all chems at q.rest
  point2 <- point1 <- apply(Z, 2, quantile, q.rest)
  # fix z2 at lower
  point2[z2] <- point1[z2] <- quantile(Z[, z2], qs.fixed[1])
  # fix z1 at lower and upper
  point2[z1] <- quantile(Z[, z1], qs.diff[2])
  point1[z1] <- quantile(Z[, z1], qs.diff[1])
  newz.q1 <- rbind(point1, point2) # has all lower quantiles of z2
  # fix all chems at q.rest
  point2 <- point1 <- apply(Z, 2, quantile, q.rest)
  # fix z2 at higher
  point2[z2] <- point1[z2] <- quantile(Z[, z2], qs.fixed[2])
  # fix z1 at lower and upper
  point2[z1] <- quantile(Z[, z1], qs.diff[2])
  point1[z1] <- quantile(Z[, z1], qs.diff[1])
  newz.q2 <- rbind(point1, point2) # has all upper quantiles of z2

  # prepare
  cc <- c(-1 * c(-1, 1), c(-1, 1))
  newz <- rbind(newz.q1, newz.q2)

  # default to using approximate calculation
  preds <- ComputePostmeanHnew(fit = fit, y = y, Z = Z, X = X, Znew = newz)

  # extract intervals
  int <- drop(cc %*% preds$postmean)
  int.se <- drop(sqrt(cc %*% preds$postvar %*% cc))
  ints <- c(est = int, sd = int.se)

  return(data.frame(z1 = colnames(Z)[z1], z2 = colnames(Z)[z2],
                    est = ints["est"], sd = ints["sd"], row.names = NULL))
}

# this function calculates confidence intervals for three-way interactions in BKMR
trivarinter_bkmr <- function(fit, z1, z2, z3, qs.diff = c(0.25, 0.75),
                             qs.fixed = c(0.25, 0.75), q.rest = 0.5) {
  #' fit = bkmr model
  #' z1 = index of chemical 1
  #' z2 = index of chemical 2
  #' z3 = index of chemical 3
  #' qs.diff = quantiles to calculate response diff for
  #' qs.fixed = quantiles to fix other chemical at
  #' q.rest = quantile to fix rest of chemicals at
  #'
  #' NOTE that order of z1, z2, z3 does not matter

  # extract fit
  y <- fit$y
  Z <- fit$Z
  X <- fit$X

  df <- purrr::map_df(1:3, \(x) {
    if(x == 1) {
      c1 <- z1; c2 <- z2; c3 <- z3
    } else if (x == 2) {
      c1 <- z2; c2 <- z3; c3 <- z1
    } else {
      c1 <- z3; c2 <- z1; c3 <- z2
    }

    # fix all chems at q.rest
    point2 <- point1 <- apply(Z, 2, quantile, q.rest)
    # fix c2 at lower
    point2[c2] <- point1[c2] <- quantile(Z[, c2], qs.fixed[1])
```

```
    # fix c3 at lower
    point2[c3] <- point1[c3] <- quantile(Z[, c3], qs.fixed[1])
    # fix c1 at lower and upper
    point2[c1] <- quantile(Z[, c1], qs.diff[2])
    point1[c1] <- quantile(Z[, c1], qs.diff[1])
    newz.q1 <- rbind(point1, point2) # has all lower quantiles of c2, c3
    # fix all chems at q.rest
    point2 <- point1 <- apply(Z, 2, quantile, q.rest)
    # fix c2 at higher
    point2[c2] <- point1[c2] <- quantile(Z[, c2], qs.fixed[2])
    # fix c3 at higher
    point2[c3] <- point1[c3] <- quantile(Z[, c3], qs.fixed[2])
    # fix c1 at lower and upper
    point2[c1] <- quantile(Z[, c1], qs.diff[2])
    point1[c1] <- quantile(Z[, c1], qs.diff[1])
    newz.q2 <- rbind(point1, point2) # has all upper quantiles of c2, c3

    # prepare
    cc <- c(-1 * c(-1, 1), c(-1, 1))
    newz <- rbind(newz.q1, newz.q2)

    # default to using approximate calculation
    preds <- ComputePostmeanHnew(fit = fit, y = y, Z = Z, X = X, Znew = newz)

    # extract intervals
    int <- drop(cc %*% preds$postmean)
    int.se <- drop(sqrt(cc %*% preds$postvar %*% cc))
    ints <- c(est = int, sd = int.se)

    return(data.frame(variable = colnames(Z)[c1], fixedat1 = colnames(Z)[c2],
                      fixedat2 = colnames(Z)[c3],
                      est = ints["est"], sd = ints["sd"], row.names = NULL))
  })

 return(df)
}

# this function generates estimated exposure-response relationships
# to assess three-way interactions in BKMR
trivarsurf_bkmr <- function(fit, z1, z2, z3, qs.diff = c(0.1, 0.5, 0.9),
                            q.fixed = 0.5, ngrid = 50) {
  #' fit = bkmr model
  #' z1 = index of chemical 1
  #' z2 = index of chemical 2
  #' z3 = index of chemical 3
  #' qs.diff = quantiles to calculate response diff for
  #' q.fixed = quantiles to fix rest of chemicals at
  #'
  #' NOTE that order of z1, z2, z3 does not matter

  # call from fit
  y <- fit$y
  Z <- fit$Z
  X <- fit$X
  z.names <- colnames(Z)

  df <- purrr::map_df(1:3, \(x) {
    if(x == 1) {
      c1 <- z1; c2 <- z2; c3 <- z3
    } else if (x == 2) {
      c1 <- z2; c2 <- z3; c3 <- z1
    } else {
      c1 <- z3; c2 <- z1; c3 <- z2
    }
```

```
    # create new ordering
    ord <- c(c1, c2, c3, setdiff(1:ncol(Z), c(c1, c2, c3)))

    # create grid of z-values to evaluate at
    z1.grid <- seq(min(Z[, ord[1]]), max(Z[, ord[1]]), length = ngrid)
    z2.grid <- quantile(Z[, ord[2]], probs = qs.diff)
    z3.grid <- quantile(Z[, ord[3]], probs = qs.diff)
    z.all <- c(list(z1.grid), list(z2.grid), list(c(-99)))
    if (ncol(Z) > 3) {
      z.others <- lapply(4:ncol(Z), function(x) quantile(Z[, ord[x]], q.fixed))
      z.all <- c(z.all, z.others)
    }
    newz.grid <- expand.grid(z.all)
    newz.grid[, 3] <- rep(z3.grid, each = ngrid)
    z1save <- newz.grid[, 1]
    colnames(newz.grid) <- colnames(Z)[ord]
    newz.grid <- newz.grid[, colnames(Z)]

    # evaluate prediction, assume approx fit
    preds <- ComputePostmeanHnew(fit = fit, y = y, Z = Z, X = X, Znew = newz.grid)
    preds.mean <- preds$postmean
    preds.se <- sqrt(diag(preds$postvar))

    # return
    return(data.frame(z1_val = z1save,
                      z23_q = rep(qs.diff, each = ngrid),
                      est = preds.mean,
                      se = preds.se,
                      z1_name = rep(colnames(Z)[c1], length(z1save)),
                      z2_name = rep(colnames(Z)[c2], length(z1save)),
                      z3_name = rep(colnames(Z)[c3], length(z1save))))
  })

  return(df)
}

# this function estimates exposure-response relationship for one chemical in BSR
univarsurf_bsr <- function(NLmod, X, C, j1, gridLength = 50,
                           quantile_rest = 0.5) {
  #' NLmod = bsr model
  #' X = matrix or dataframe of chemical values used to fit model
  #' C = matrix or dataframe of covariate values used to fit model
  #' gridLength = number of points to estimate response at
  #' j1 = index of chemical
  #' quantile_rest = quantile to fix other chemicals at

  # define parameters
  n  <-  dim(X)[1]
  ns <- NLmod$ns
  k <- NLmod$k
  p <- dim(X)[2]
  Xstar <- array(NA, dim = c(n, p, ns + 1))
  Xstar[, , 1] <- 1
  for (j in 1:p) {
    Xstar[, j, 2:(ns + 1)] <- scale(splines::ns(X[, j], df = ns))
  }

  # define posteriors
  zetaPost <- NLmod$posterior$zeta
  betaList <- NLmod$posterior$beta
  betaCPost <- NLmod$posterior$betaC
  totalScans <- dim(NLmod$posterior$betaC)[2]
  nChains <- dim(NLmod$posterior$betaC)[1]
```

```r
  # create design of covariates
  pc <- dim(C)[2]
  NewDesignC <- matrix(NA, gridLength, pc + 1)
  NewDesignC[, 1] <- 1
  for (jc in 1:pc) {
    NewDesignC[, jc + 1] <- mean(C[, jc])
  }

  # create design of chemicals
  n <- dim(X)[1]
  NewDesignMat <- matrix(NA, gridLength, p)
  for (j in 1:p) {
    NewDesignMat[, j] <- quantile(X[, j], quantile_rest)
  }
  NewDesignMat[, j1] <- seq(quantile(X[, j1], 0.025),
                            quantile(X[, j1], 0.975), length = gridLength)
  NewDesign <- array(NA, dim = c(gridLength, p, ns + 1))
  NewDesign[, , 1] <- 1
  for (j in 1:p) {
    temp_ns_object <- splines::ns(X[, j], df = ns)
    temp_sds <- apply(temp_ns_object, 2, sd)
    temp_means <- apply(temp_ns_object, 2, mean)
    NewDesign[, j, 2:(ns + 1)] <- t((t(
      predict(temp_ns_object,
              NewDesignMat[, j])
    ) - temp_means) / temp_sds)
  }

  # generate predictions
  predictions <- NLinteraction:::PredictionsMixture(
    XstarOld = Xstar,
    XstarNew = NewDesign,
    designC = NewDesignC,
    totalScans = totalScans,
    nChains = nChains,
    zetaPost = zetaPost,
    betaList = betaList,
    betaCPost = betaCPost,
    k = k,
    ns = ns
  )

  return(data.frame(
    j1val = NewDesignMat[, j1],
    est = apply(predictions$PredictedPost, 3, mean),
    lower = apply(predictions$PredictedPost, 3, quantile, 0.025),
    upper = apply(predictions$PredictedPost, 3, quantile, 0.975)))
}

# this function generates estimated exposure-response relationships
# to assess two-way interactions in BSR
bivarsurf_bsr <- function(NLmod, X, C, j1, j2, gridLength = 50,
                          quantile_j2 = c(0.1, 0.5, 0.9), quantile_rest = 0.5) {
  #' NLmod = bsr model
  #' X = matrix or dataframe of chemical values used to fit model
  #' C = matrix or dataframe of covariate values used to fit model
  #' gridLength = number of points to estimate response at
  #' j1 = index of first chemical
  #' j2 = index of second chemical
  #' quantile_j2 = vector of quantiles to fix second chemical at
  #' quantile_rest = quantile to fix other chemicals at
  #'
  #' NOTE order of j1 and j2 does not matter
```

```r
# define parameters
n  <-  dim(X)[1]
ns <- NLmod$ns
k <- NLmod$k
p <- dim(X)[2]
Xstar <- array(NA, dim = c(n, p, ns + 1))
Xstar[, , 1] <- 1
for (j in 1:p) {
  Xstar[, j, 2:(ns + 1)] <- scale(splines::ns(X[, j], df = ns))
}

# define posteriors
zetaPost <- NLmod$posterior$zeta
betaList <- NLmod$posterior$beta
betaCPost <- NLmod$posterior$betaC
totalScans <- dim(NLmod$posterior$betaC)[2]
nChains <- dim(NLmod$posterior$betaC)[1]

# create design of covariates
pc <- dim(C)[2]
NewDesignC <- matrix(NA, gridLength, pc + 1)
NewDesignC[, 1] <- 1
for (jc in 1:pc) {
  NewDesignC[, jc + 1] <- mean(C[, jc])
}

# for each quantile of j2
df <- purrr::map_df(quantile_j2, \(quantile_j2) {
  # create design of chemicals
  n <- dim(X)[1]
  NewDesignMat <- matrix(NA, gridLength, p)
  for (j in 1:p) {
    NewDesignMat[, j] <- quantile(X[, j], quantile_rest)
  }
  NewDesignMat[, j1] <- seq(quantile(X[, j1], 0.025),
                            quantile(X[, j1], 0.975), length = gridLength)
  NewDesignMat[, j2] <- quantile(X[, j2], quantile_j2)
  NewDesign <- array(NA, dim = c(gridLength, p, ns + 1))
  NewDesign[, , 1] <- 1
  for (j in 1:p) {
    temp_ns_object <- splines::ns(X[, j], df = ns)
    temp_sds <- apply(temp_ns_object, 2, sd)
    temp_means <- apply(temp_ns_object, 2, mean)
    NewDesign[, j, 2:(ns + 1)] <- t((t(predict(temp_ns_object,
                                               NewDesignMat[, j])) - temp_means)/temp_sds)
  }

  # generate predictions
  predictions <- NLinteraction:::PredictionsMixture(
    XstarOld = Xstar, XstarNew = NewDesign,
    designC = NewDesignC, totalScans = totalScans, nChains = nChains,
    zetaPost = zetaPost, betaList = betaList, betaCPost = betaCPost,
    k = k, ns = ns)

  # get surface
  return(data.frame(
    j1val = NewDesignMat[, j1],
    j2quant = rep(quantile_j2, gridLength),
    est = apply(predictions$PredictedPost, 3, mean),
    lower = apply(predictions$PredictedPost, 3, quantile, 0.025),
    upper = apply(predictions$PredictedPost, 3, quantile, 0.975)
  ))
})
return(df)
```

```
}

# this function generates estimated exposure-response relationships
# to assess three-way interactions in BSR
trivarsurf_bsr <- function(NLmod, X, C, j1, j2, j3, gridLength = 50,
                           quantile_j23 = c(0.1, 0.5, 0.9), quantile_rest = 0.5) {
  #' NLmod = bsr model
  #' X = matrix or dataframe of chemical values used to fit model
  #' C = matrix or dataframe of covariate values used to fit model
  #' gridLength = number of points to estimate response at
  #' j1 = index of first chemical, chemical used as primary predictor
  #' j2 = index of second chemical
  #' j3 = index of third chemical
  #' quantile_j23 = vector of quantiles to fix second and third chemicals at
  #' quantile_rest = quantile to fix other chemicals at
  #'
  #' NOTE order of j1, j2, j3 matters

  # define parameters
  n  <-  dim(X)[1]
  ns <- NLmod$ns
  k <- NLmod$k
  p <- dim(X)[2]
  Xstar <- array(NA, dim = c(n, p, ns + 1))
  Xstar[, , 1] <- 1
  for (j in 1:p) {
    Xstar[, j, 2:(ns + 1)] <- scale(splines::ns(X[, j], df = ns))
  }

  # define posteriors
  zetaPost <- NLmod$posterior$zeta
  betaList <- NLmod$posterior$beta
  betaCPost <- NLmod$posterior$betaC
  totalScans <- dim(NLmod$posterior$betaC)[2]
  nChains <- dim(NLmod$posterior$betaC)[1]

  # create design of covariates
  pc <- dim(C)[2]
  NewDesignC <- matrix(NA, gridLength, pc + 1)
  NewDesignC[, 1] <- 1
  for (jc in 1:pc) {
    NewDesignC[, jc + 1] <- mean(C[, jc])
  }

  # for each quantile of j2, j3
  df <- purrr::map_df(quantile_j23, \(quantile_j23) {
    # create design of chemicals
    n <- dim(X)[1]
    NewDesignMat <- matrix(NA, gridLength, p)
    for (j in 1:p) {
      NewDesignMat[, j] <- quantile(X[, j], quantile_rest)
    }
    NewDesignMat[, j1] <- seq(quantile(X[, j1], 0.025),
                              quantile(X[, j1], 0.975), length = gridLength)
    NewDesignMat[, j2] <- quantile(X[, j2], quantile_j23)
    NewDesignMat[, j3] <- quantile(X[, j3], quantile_j23)
    NewDesign <- array(NA, dim = c(gridLength, p, ns + 1))
    NewDesign[, , 1] <- 1
    for (j in 1:p) {
      temp_ns_object <- splines::ns(X[, j], df = ns)
      temp_sds <- apply(temp_ns_object, 2, sd)
      temp_means <- apply(temp_ns_object, 2, mean)
      NewDesign[, j, 2:(ns + 1)] <- t((t(predict(temp_ns_object,
                                                 NewDesignMat[, j])) - temp_means)/temp_sds)
```

```
    }

    # generate predictions
    predictions <- NLinteraction:::PredictionsMixture(
      XstarOld = Xstar, XstarNew = NewDesign,
      designC = NewDesignC, totalScans = totalScans, nChains = nChains,
      zetaPost = zetaPost, betaList = betaList, betaCPost = betaCPost,
      k = k, ns = ns)

    # get surface
    return(data.frame(
      j1val = NewDesignMat[, j1],
      j23quant = rep(quantile_j23, gridLength),
      est = apply(predictions$PredictedPost, 3, mean),
      lower = apply(predictions$PredictedPost, 3, quantile, 0.025),
      upper = apply(predictions$PredictedPost, 3, quantile, 0.975)
    ))
  })
  return(df)
}
```

Now, we extract p-values from the multiple linear regression models.

```
# naive small, chemxchem models
mlr_sm <- read_rds("sim/mlr_mods_sm.RDS")

mlrsm_pval <- 1:1700 |>
  map_df(\(x) {
    mod <- mlr_sm[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(mlrsm_pval, "sim/_mlr/pvalsm.csv")

# naive large, chemxchem models
mlr_lg <- read_rds("sim/mlr_mods_lg.RDS")

mlrlg_pval <- 1:1700 |>
  map_df(\(x) {
    mod <- mlr_lg[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(mlrlg_pval, "sim/_mlr/pvallg.csv")

# oracle small, chemxchem models
orac_sm <- read_rds("sim/oracle_mods_sm.RDS")

oracsm_pval <- 1:1700 |>
  map_df(\(x) {
    mod <- orac_sm[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(oracsm_pval, "sim/_oracle/pvalsm.csv")
```

```
# oracle large, chemxchem models
orac_lg <- read_rds("sim/oracle_mods_lg.RDS")

oraclg_pval <- 1:1700 |>
  map_df(\(x) {
    mod <- orac_lg[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(oraclg_pval, "sim/_oracle/pvallg.csv")
```

Next, we extract output from BKMR models run on smaller and larger size simulated datasets, including the base case and models with interactions between chemicals.

```
# load packages
library(tidyverse)
library(bkmr)

######
# extract bkmr smalls!
######

# get file paths of models
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ksm <- list_files[grepl("_rslurm_bkmr_sm", list_files)]

ksm_subf <- list.dirs(list_ksm, full.names = TRUE, recursive = TRUE)
ksm_mod <- ksm_subf[grepl("mods", ksm_subf)]

ksm_labels <- gsub("\\D", "", ksm_mod)
ksm_labels <- ifelse(ksm_labels == "", 1, as.numeric(ksm_labels))

# get paths
ksm_paths <- ksm_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ksm_labels)

# extract PIP's
ksm_pips <- names(ksm_paths) |>
  purrr::map_df(\(x) {
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- data.frame(
          ExtractPIPs(bkmr)
        ) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_pips, "sim/bkmr_sm/pips.csv")
```

```r
# extract univariate relationships
ksm_univ <- names(ksm_paths) |>
  purrr::map_df(\(x) {
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- data.frame(
          PredictorResponseUnivar(bkmr)
        ) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_univ, "sim/bkmr_sm/univ_expresp.csv")

# extract bivariate relationships
ksm_biv <- names(ksm_paths)[2:13] |> # only for 2-way interaction
  purrr::map_df(\(x) {
    indices <- case_when(
      x %in% 2:5 ~ c(4, 5), # Hg and Ni
      x %in% 6:9 ~ c(1, 2), # Cd and As
      x %in% 10:13 ~ c(3, 5) # Co and Ni
    )
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        bivar <- PredictorResponseBivar(bkmr,
                                z.pairs = rbind(indices), verbose = FALSE)
        result <- data.frame(
          PredictorResponseBivarLevels(
            pred.resp.df = bivar,
            Z = bkmr$Z, qs = c(0.1, 0.5, 0.9))

        ) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
ksm_biv_nona <- na.omit(ksm_biv)
write_csv(ksm_biv_nona, "sim/bkmr_sm/biv_expresp.csv")

# load in fxn
source("extract_fxns.R")

# extract trivariate relationships
ksm_triv <- names(ksm_paths)[14:17] |> # only for 3-way
  purrr::map_df(\(x) {
    message("starting ", x)
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- trivarsurf_bkmr(bkmr, 4, 5, 6,
                                  qs.diff = c(0.1, 0.5, 0.9),
                                  q.fixed = 0.5, ngrid = 50) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_triv, "sim/bkmr_sm/triv_expresp.csv")
```

```r
# extract one vs. rest bivariate interactions
ksm_ints <- names(ksm_paths)[2:17] |>
  purrr::map_df(\(x) {
    print(paste0("starting ", x))
    indices <- case_when(
      x %in% 2:5 ~ list(c(4, 5)), # Hg and Ni
      x %in% 6:9 ~ list(c(1, 2)), # Cd and As
      x %in% 10:13 ~ list(c(3, 5)), # Co and Ni
      x %in% 14:17 ~ list(c(4, 5, 6)) # Hg, Ni, Tl
    )
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        ints <- SingVarIntSummaries(bkmr,
                                    which.z = indices[[1]],
                                    qs.diff = c(0.25, 0.75),
                                    qs.fixed = c(0.25, 0.75),
                                    method = "approx")
        result <- data.frame(ints) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_ints, "sim/bkmr_sm/ints.csv")

# load in fxn
source("extract_fxns.R")

# extract one vs. other bivariate interactions
ksm_intb <- names(ksm_paths)[2:13] |> # only for two-way
  purrr::map_df(\(x) {
    print(paste0("starting ", x))
    indices <- case_when(
      x %in% 2:5 ~ list(c(4, 5)), # Hg and Ni
      x %in% 6:9 ~ list(c(1, 2)), # Cd and As
      x %in% 10:13 ~ list(c(3, 5)) # Co and Ni
    )
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        ints <- bivarinter_bkmr(bkmr,
                                z1 = indices[[1]][1],
                                z2 = indices[[1]][2],
                                qs.diff = c(0.25, 0.75),
                                qs.fixed = c(0.25, 0.75),
                                q.rest = 0.5)
        result <- data.frame(ints) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_intb, "sim/bkmr_sm/int_bivar.csv")

# FDR for Hg-Ni
indices <- t(combn(1:10, 2))
ksm_intb_hgni <- names(ksm_paths)[2:5] |> # only for Hg-Ni
  purrr::map_df(\(x) {
    message("starting ", x)
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
```

```
        bkmr <- read_rds(y)
        result <- 1:45 |>
          purrr::map_df(\(z) {
            ints <- bivarinter_bkmr(bkmr,
                                    z1 = indices[z, ][1],
                                    z2 = indices[z, ][2],
                                    qs.diff = c(0.25, 0.75),
                                    qs.fixed = c(0.25, 0.75),
                                    q.rest = 0.5)
            data.frame(ints)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_intb_hgni, "sim/bkmr_sm/int_bivar_fullhgni.csv")

# FDR for rest
indices <- t(combn(1:10, 2))
ksm_intb_rest <- names(ksm_paths)[6:13] |> # for rest
  purrr::map_df(\(x) {
    message("starting ", x)
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- 1:45 |>
          purrr::map_df(\(z) {
            ints <- bivarinter_bkmr(bkmr,
                                    z1 = indices[z, ][1],
                                    z2 = indices[z, ][2],
                                    qs.diff = c(0.25, 0.75),
                                    qs.fixed = c(0.25, 0.75),
                                    q.rest = 0.5)
            data.frame(ints)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ksm_intb_rest, "sim/bkmr_sm/int_bivar_fullrest.csv")

# extract one vs. 2 others trivariate interactions
ksm_intt <- names(ksm_paths)[14:17] |> # only for three-way
  purrr::map_df(\(x) {
    print(paste0("starting ", x))
    ksm_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        ints <- trivarinter_bkmr(bkmr,
                                 z1 = 4, z2 = 5, z3 = 6,
                                 qs.diff = c(0.25, 0.75),
                                 qs.fixed = c(0.25, 0.75),
                                 q.rest = 0.5)
        result <- data.frame(ints) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
```

```r
write_csv(ksm_intt, "sim/bkmr_sm/int_trivar.csv")


######
# extract bkmr larges!
######

list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_klg <- list_files[grepl("_rslurm_bkmr_lg", list_files)]

klg_subf <- list.dirs(list_klg, full.names = TRUE, recursive = TRUE)
klg_mod <- klg_subf[grepl("mods", klg_subf)]

klg_labels <- gsub("\\D", "", klg_mod)
klg_labels <- ifelse(klg_labels == "", 1, as.numeric(klg_labels))

# get paths
klg_paths <- klg_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = klg_labels)

# extract PIP's
klg_pips <- names(klg_paths) |>
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- data.frame(
          ExtractPIPs(bkmr)
        ) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_pips, "sim/bkmr_lg/pips.csv")

# extract univariate relationships
klg_univ <- names(klg_paths) |>
  purrr::map_df(\(x) {
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- data.frame(
          PredictorResponseUnivar(bkmr)
        ) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_univ, "sim/bkmr_lg/univ_expresp.csv")

# extract bivariate relationships
klg_biv <- names(klg_paths)[2:13] |> # only for 2-way interaction
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    indices <- case_when(
      x %in% 2:5 ~ c(4, 5), # Hg and Ni
```

```
      x %in% 6:9 ~ c(1, 2), # Cd and As
      x %in% 10:13 ~ c(3, 5) # Co and Ni
    )
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        bivar <- PredictorResponseBivar(bkmr,
                                        z.pairs = rbind(indices), verbose = FALSE)
        result <- data.frame(
          PredictorResponseBivarLevels(
            pred.resp.df = bivar,
            Z = bkmr$Z, qs = c(0.1, 0.5, 0.9))

        ) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
# write_csv(klg_biv, "sim/bkmr_lg/biv_expresp.csv")
# write_rds(klg_biv, "sim/bkmr_lg/biv_expresp.rds")
klg_biv_nona <- na.omit(klg_biv)
write_csv(klg_biv_nona, "sim/bkmr_lg/biv_expresp.csv")

# load in fxn
source("extract_fxns.R")

# extract trivariate relationships
klg_triv <- names(klg_paths)[14:17] |> # only for 3-way
  purrr::map_df(\(x) {
    message("starting ", x)
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- trivarsurf_bkmr(bkmr, 4, 5, 6,
                                  qs.diff = c(0.1, 0.5, 0.9),
                                  q.fixed = 0.5, ngrid = 50) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_triv, "sim/bkmr_lg/triv_expresp.csv")

# extract one vs. rest bivariate interactions
klg_ints <- names(klg_paths)[2:17] |>
  purrr::map_df(\(x) {
    print(paste0("starting ", x))
    indices <- case_when(
      x %in% 2:5 ~ list(c(4, 5)), # Hg and Ni
      x %in% 6:9 ~ list(c(1, 2)), # Cd and As
      x %in% 10:13 ~ list(c(3, 5)), # Co and Ni
      x %in% 14:17 ~ list(c(4, 5, 6)) # Hg, Ni, Tl
    )
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        ints <- SingVarIntSummaries(bkmr,
                                    which.z = indices[[1]],
                                    qs.diff = c(0.25, 0.75),
                                    qs.fixed = c(0.25, 0.75),
                                    method = "approx")
        result <- data.frame(ints) |>
```

```
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_ints, "sim/bkmr_lg/ints.csv")

# load in fxn
source("extract_fxns.R")

# extract one vs. other bivariate interactions
klg_intb <- names(klg_paths)[2:13] |> # only for two-way
  purrr::map_df(\(x) {
    print(paste0("starting ", x))
    indices <- case_when(
      x %in% 2:5 ~ list(c(4, 5)), # Hg and Ni
      x %in% 6:9 ~ list(c(1, 2)), # Cd and As
      x %in% 10:13 ~ list(c(3, 5)) # Co and Ni
    )
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        ints <- bivarinter_bkmr(bkmr,
                                z1 = indices[[1]][1],
                                z2 = indices[[1]][2],
                                qs.diff = c(0.25, 0.75),
                                qs.fixed = c(0.25, 0.75),
                                q.rest = 0.5)
        result <- data.frame(ints) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_intb, "sim/bkmr_lg/int_bivar.csv")

# FDR for Hg-Ni
indices <- t(combn(1:10, 2))
klg_intb_hgni <- names(klg_paths)[2:5] |> # only for Hg-Ni
  purrr::map_df(\(x) {
    message("starting ", x)
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- 1:45 |>
          purrr::map_df(\(z) {
            ints <- bivarinter_bkmr(bkmr,
                                    z1 = indices[z, ][1],
                                    z2 = indices[z, ][2],
                                    qs.diff = c(0.25, 0.75),
                                    qs.fixed = c(0.25, 0.75),
                                    q.rest = 0.5)
            data.frame(ints)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_intb_hgni, "sim/bkmr_lg/int_bivar_fullhgni.csv")
```

```
# FDR for rest
indices <- t(combn(1:10, 2))
klg_intb_rest <- names(klg_paths)[6:13] |> # for Cd-As and Ni-Co
  purrr::map_df(\(x) {
    message("starting ", x)
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        result <- 1:45 |>
          purrr::map_df(\(z) {
            ints <- bivarinter_bkmr(bkmr,
                                    z1 = indices[z, ][1],
                                    z2 = indices[z, ][2],
                                    qs.diff = c(0.25, 0.75),
                                    qs.fixed = c(0.25, 0.75),
                                    q.rest = 0.5)
            data.frame(ints)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_intb_rest, "sim/bkmr_lg/int_bivar_fullrest.csv")

# extract one vs. 2 others trivariate interactions
klg_intt <- names(klg_paths)[14:17] |> # only for three-way
  purrr::map_df(\(x) {
    print(paste0("starting ", x))
    klg_paths[[x]] |>
      purrr::map_df(\(y) {
        bkmr <- read_rds(y)
        ints <- trivarinter_bkmr(bkmr,
                                 z1 = 4, z2 = 5, z3 = 6,
                                 qs.diff = c(0.25, 0.75),
                                 qs.fixed = c(0.25, 0.75),
                                 q.rest = 0.5)
        result <- data.frame(ints) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
        rm(bkmr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(klg_intt, "sim/bkmr_lg/int_trivar.csv")
```

Next, we extract output from BSR models run on smaller and larger size simulated datasets, including the base case and models with interactions between chemicals.

```
# load packages
library(NLinteraction)
library(tidyverse)

# create indices of chemical names
cnames <- c("As", "Cd", "Co", "Hg", "Ni", "Tl", "Pb", "Mo", "Sb", "Sn")

######
# extract bsr smalls!
######
```

```r
# extracting file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ssm <- list_files[grepl("_rslurm_bsr_sm", list_files)]

ssm_subf <- list.dirs(list_ssm, full.names = TRUE, recursive = TRUE)
ssm_mod <- ssm_subf[grepl("mods", ssm_subf)]

ssm_labels <- gsub("\\D", "", ssm_mod)
ssm_labels <- ifelse(ssm_labels == "", 1, as.numeric(ssm_labels))

# get paths
ssm_paths <- ssm_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ssm_labels)

# extract PIPs
ssm_pips <- names(ssm_paths) |>
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    ssm_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- data.frame(variable = cnames, PIP = bsr$MainPIP) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns)
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ssm_pips, "sim/bsr_sm/pips.csv")

# extract bivariate pip's
ssm_pip_biv <- names(ssm_paths) |>
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    ssm_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- reshape2::melt(bsr$InteractionPIP,
                                 na.rm = TRUE,
                                 value.name = "PIP") |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns)
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ssm_pip_biv, "sim/bsr_sm/pip_biv.csv")

# extract trivariate pip's
ssm_pip_triv <- names(ssm_paths)[14:17] |> # trivariate only
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    ssm_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- data.frame(
          PIP = InteractionProb(NLmod = bsr, Xsub = c(4, 5, 6)),
          trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
          df = bsr$ns
```

```
        )
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(ssm_pip_triv, "sim/bsr_sm/pip_triv.csv")

# read back in data
out1_resp1 <- read_rds("sim/sim_resp_sm_a.RDS")

# load in functions for extracting results
source("extract_fxns.R")

# extract bivariate relationships
ssm_biv <- names(ssm_paths)[2:13] |> # bivariate only
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    indices <- case_when(
      x %in% 2:5 ~ c(4, 5), # Hg and Ni
      x %in% 6:9 ~ c(1, 2), # Cd and As
      x %in% 10:13 ~ c(3, 5) # Co and Ni
    )
    ssm_paths[[x]] |>
      purrr::map_df(\(y) {
        trial <- as.numeric(sub(".+_(.+)d.+", "\\1", y))
        if(trial %% 5 == 0) print(paste0("index ", y))
        df <- out1_resp1[[trial]]
        X <- df |>
          select(As:Sn) |>
          as.matrix.data.frame()
        C <- df |>
          bind_cols(
            data.frame(model.matrix(~ race-1, data =
                                      mutate(df, race = as.factor(race))))
          ) |>
          select(race2:race5, smoke:bmi) |>
          as.matrix.data.frame()
        Y <- df$y

        bsr <- read_rds(y)

        result1 <- bivarsurf_bsr(bsr, X = X, C = C, j1 = indices[1], j2 = indices[2],
                                 gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                 quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[indices[1]],
                 j2 = cnames[indices[2]])
        result2 <- bivarsurf_bsr(bsr, X = X, C = C, j1 = indices[2], j2 = indices[1],
                                 gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                 quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[indices[2]],
                 j2 = cnames[indices[1]])

        rm(bsr)
        return(bind_rows(result1, result2))
      }) |>
      mutate(case = x)
  })
write_csv(ssm_biv, "sim/bsr_sm/biv_expresp.csv")
```

```r
# extract trivariate relationships
ssm_triv <- names(ssm_paths)[14:17] |> # trivariate only
  purrr::map_df(\(x) {
    message("starting ", x)
    ssm_paths[[x]] |>
      purrr::map_df(\(y) {
        trial <- as.numeric(sub(".+_(.+)d.+", "\\1", y))
        if(trial %% 5 == 0) message("   index ", trial)
        df <- out1_resp1[[trial]]
        X <- df |>
          select(As:Sn) |>
          as.matrix.data.frame()
        C <- df |>
          bind_cols(
            data.frame(model.matrix(~ race-1, data =
                                      mutate(df, race = as.factor(race))))
          ) |>
          select(race2:race5, smoke:bmi) |>
          as.matrix.data.frame()
        Y <- df$y

        bsr <- read_rds(y)

        result1 <- trivarsurf_bsr(bsr, X = X, C = C, j1 = 4, j2 = 5, j3 = 6,
                                  gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                  quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[4],
                 j2 = cnames[5],
                 j3 = cnames[6])
        result2 <- trivarsurf_bsr(bsr, X = X, C = C, j1 = 5, j2 = 6, j3 = 4,
                                  gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                  quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[5],
                 j2 = cnames[6],
                 j3 = cnames[4])
        result3 <- trivarsurf_bsr(bsr, X = X, C = C, j1 = 6, j2 = 4, j3 = 5,
                                  gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                  quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[6],
                 j2 = cnames[4],
                 j3 = cnames[5])

        rm(bsr)
        return(bind_rows(result1, result2, result3))
      }) |>
      mutate(case = x)
  })
write_csv(ssm_triv, "sim/bsr_sm/triv_expresp.csv")

######
# extract bsr larges!
######

# extracting file names

list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_slg <- list_files[grepl("_rslurm_bsr_lg", list_files)]
```

```r
slg_subf <- list.dirs(list_slg, full.names = TRUE, recursive = TRUE)
slg_mod <- slg_subf[grepl("mods", slg_subf)]

slg_labels <- gsub("\\D", "", slg_mod)
slg_labels <- ifelse(slg_labels == "", 1, as.numeric(slg_labels))

# get paths
slg_paths <- slg_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = slg_labels)

# extract PIPs
slg_pips <- names(slg_paths) |>
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    slg_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- data.frame(variable = cnames, PIP = bsr$MainPIP) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns)
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(slg_pips, "sim/bsr_lg/pips.csv")

# extract bivariate pip's
slg_pip_biv <- names(slg_paths) |>
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    slg_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- reshape2::melt(bsr$InteractionPIP,
                                 na.rm = TRUE,
                                 value.name = "PIP") |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns)
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })
write_csv(slg_pip_biv, "sim/bsr_lg/pip_biv.csv")

# extract trivariate pip's
slg_pip_triv <- names(slg_paths)[14:17] |> # trivariate only
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    slg_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- data.frame(
          PIP = InteractionProb(NLmod = bsr, Xsub = c(4, 5, 6)),
          trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
          df = bsr$ns
        )
        rm(bsr)
        return(result)
      }) |>
```

```
      mutate(case = x)
  })
write_csv(slg_pip_triv, "sim/bsr_lg/pip_triv.csv")

# read back in data
out2_resp1 <- read_rds("sim/sim_resp_lg_a.RDS")
source("extract_fxns.R")

# extract bivariate relationships
slg_biv <- names(slg_paths)[2:13] |> # bivariate only
  purrr::map_df(\(x) {
    print(paste0("starting at ", x))
    indices <- case_when(
      x %in% 2:5 ~ c(4, 5), # Hg and Ni
      x %in% 6:9 ~ c(1, 2), # Cd and As
      x %in% 10:13 ~ c(3, 5) # Co and Ni
    )
    slg_paths[[x]] |>
      purrr::map_df(\(y) {
        trial <- as.numeric(sub(".+_(.+)d.+", "\\1", y))
        if(trial %% 5 == 0) print(paste0("index ", y))
        df <- out2_resp1[[trial]]
        X <- df |>
          select(As:Sn) |>
          as.matrix.data.frame()
        C <- df |>
          bind_cols(
            data.frame(model.matrix(~ race-1, data =
                                      mutate(df, race = as.factor(race))))
          ) |>
          select(race2:race5, smoke:bmi) |>
          as.matrix.data.frame()
        Y <- df$y

        bsr <- read_rds(y)

        result1 <- bivarsurf_bsr(bsr, X = X, C = C, j1 = indices[1], j2 = indices[2],
                                 gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                 quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[indices[1]],
                 j2 = cnames[indices[2]])
        result2 <- bivarsurf_bsr(bsr, X = X, C = C, j1 = indices[2], j2 = indices[1],
                                 gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                 quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[indices[2]],
                 j2 = cnames[indices[1]])

        rm(bsr)
        return(bind_rows(result1, result2))
      }) |>
      mutate(case = x)
  })
write_csv(slg_biv, "sim/bsr_lg/biv_expresp.csv")

# extract trivariate relationships
slg_triv <- names(slg_paths)[14:17] |> # trivariate only
  purrr::map_df(\(x) {
    message("starting ", x)
    slg_paths[[x]] |>
      purrr::map_df(\(y) {
```

```
        trial <- as.numeric(sub(".+_(.+)d.+", "\\1", y))
        if(trial %% 5 == 0) message("   index ", trial)
        df <- out2_resp1[[trial]]
        X <- df |>
          select(As:Sn) |>
          as.matrix.data.frame()
        C <- df |>
          bind_cols(
            data.frame(model.matrix(~ race-1, data =
                                      mutate(df, race = as.factor(race))))
          ) |>
          select(race2:race5, smoke:bmi) |>
          as.matrix.data.frame()
        Y <- df$y

        bsr <- read_rds(y)

        result1 <- trivarsurf_bsr(bsr, X = X, C = C, j1 = 4, j2 = 5, j3 = 6,
                                  gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                  quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[4],
                 j2 = cnames[5],
                 j3 = cnames[6])
        result2 <- trivarsurf_bsr(bsr, X = X, C = C, j1 = 5, j2 = 6, j3 = 4,
                                  gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                  quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[5],
                 j2 = cnames[6],
                 j3 = cnames[4])
        result3 <- trivarsurf_bsr(bsr, X = X, C = C, j1 = 6, j2 = 4, j3 = 5,
                                  gridLength = 50, quantile_j2 = c(0.1, 0.5, 0.9),
                                  quantile_rest = 0.5) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)),
                 df = bsr$ns,
                 j1 = cnames[6],
                 j2 = cnames[4],
                 j3 = cnames[5])

        rm(bsr)
        return(bind_rows(result1, result2, result3))
      }) |>
      mutate(case = x)
  })
write_csv(slg_triv, "sim/bsr_lg/triv_expresp.csv")
```

Next, we extract output from stratified models.

```
# naive small, chemxcov models
mlr_sm_re <- read_rds("sim/mlr_mods_sm_re.RDS")

mlrsmre_pval <- 1:400 |>
  map_df(\(x) {
    mod <- mlr_sm_re[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
```

```r
write_csv(mlrsmre_pval, "sim/_mlr/pvalsmre.csv")

# naive large, chemxcov models
mlr_lg_re <- read_rds("sim/mlr_mods_lg_re.RDS")

mlrlgre_pval <- 1:400 |>
  map_df(\(x) {
    mod <- mlr_lg_re[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(mlrlgre_pval, "sim/_mlr/pvallgre.csv")

# oracle small, chemxcov models
orac_sm_re <- read_rds("sim/oracle_mods_sm_re.RDS")

oracsmre_pval <- 1:400 |>
  map_df(\(x) {
    mod <- orac_sm_re[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(oracsmre_pval, "sim/_oracle/pvalsmre.csv")

# oracle large, chemxcov models
orac_lg_re <- read_rds("sim/oracle_mods_lg_re.RDS")

oraclgre_pval <- 1:400 |>
  map_df(\(x) {
    mod <- orac_lg_re[[x]]
    data.frame(summary(mod)$coefficients) |>
      rownames_to_column(var = "var") |>
      select(var, p = 5) |>
      mutate(case = x)
  })
write_csv(oraclgre_pval, "sim/_oracle/pvallgre.csv")
```

```r
#########
# extract bkmr stratified
#########

## small ##

list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ksmre <- list_files[grepl("_rslurm_ksmre", list_files)]

ksmre_subf <- list.dirs(list_ksmre, full.names = TRUE, recursive = TRUE)
ksmre_mod <- ksmre_subf[grepl("mods", ksmre_subf)]

ksmre_labels <- gsub("\\D", "", ksmre_mod)
ksmre_labels <- ifelse(ksmre_labels == "", 1, as.numeric(ksmre_labels))

# get paths
ksmre_paths <- ksmre_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ksmre_labels)
```

```
one <- read_rds(ksmre_paths[[1]][[1]])
oney <- SingVarRiskSummaries(one[[2]], q.fixed = 0.5)
onex <- SingVarRiskSummaries(one[[2]], which.z = 4, q.fixed = 0.5)
onez <- PredictorResponseUnivar(one[[2]], which.z = 4)

# extract PIPs
ksmre_pips <- names(ksmre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    ksmre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
        1:6 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(variable = NA, PIP = NA, race = z)
            } else {
              # get pips
              df <- ExtractPIPs(mod) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(ksmre_pips, "sim/re/ksmre_pips.csv")

# get confidence intervals
ksmre_ints <- names(ksmre_paths) |>
  purrr::map_df(\(x) { # for each case
    ksmre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
        1:6 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(est = NA, sd = NA, race = z)
            } else {
              # compute estimate of 0.75 vs. 0.25 quantiles
              df <- SingVarRiskSummaries(mod, which.z = 4, q.fixed = 0.5) |>
                select(est, sd) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(ksmre_ints, "sim/re/ksmre_ints.csv")

# get exposure response relationships
ksmre_univ <- names(ksmre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    ksmre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
```

```
        1:6 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(z1 = NA, est = NA, se = NA, race = z)
            } else {
              # compute estimated Hg-response relationship
              df <- PredictorResponseUnivar(mod, which.z = 4) |>
                select(z1 = z, est, se) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(ksmre_univ, "sim/re/ksmre_expresp.csv")

## large ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_klgre <- list_files[grepl("_rslurm_klgre", list_files)]

klgre_subf <- list.dirs(list_klgre, full.names = TRUE, recursive = TRUE)
klgre_mod <- klgre_subf[grepl("mods", klgre_subf)]

klgre_labels <- gsub("\\D", "", klgre_mod)
klgre_labels <- ifelse(klgre_labels == "", 1, as.numeric(klgre_labels))

# get paths
klgre_paths <- klgre_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = klgre_labels)

# get pips
klgre_pips <- names(klgre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    klgre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
        1:5 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(variable = NA, PIP = NA, race = z)
            } else {
              # get pips
              df <- ExtractPIPs(mod) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(klgre_pips, "sim/re/klgre_pips.csv")
```

```
# get confidence intervals
klgre_ints <- names(klgre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    klgre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
        1:5 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(est = NA, sd = NA, race = z)
            } else {
              # compute estimate of 0.75 vs. 0.25 quantiles
              df <- SingVarRiskSummaries(mod, which.z = 4, q.fixed = 0.5) |>
                select(est, sd) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(klgre_ints, "sim/re/klgre_ints.csv")

# get exposure response relationships
klgre_univ <- names(klgre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    klgre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
        1:5 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(z1 = NA, est = NA, se = NA, race = z)
            } else {
              # compute estimated Hg-response relationship
              df <- PredictorResponseUnivar(mod, which.z = 4) |>
                select(z1 = z, est, se) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(klgre_univ, "sim/re/klgre_expresp.csv")
```

```
######
# extract bsr stratified
#######

# first, move all stratified models into one folder
# because we ran each model as one separate job

## small ##
```

```r
# extracting file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_bsri <- list_files[grepl("_rslurm_ssmre(\\d+)_", list_files)]

bsri_subf <- list.dirs(list_bsri, full.names = TRUE, recursive = TRUE)
bsri_mod <- bsri_subf[grepl("mods", bsri_subf)]
bsri_time <- bsri_subf[grepl("times", bsri_subf)]

# copy mods to correct folder
for (f in bsri_mod) {
  file.copy(from = list.files(f, full.names = TRUE),
            to   = paste0(substr(f, 1, 15), "/mods"))
}

## large ##
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_bsri <- list_files[grepl("_rslurm_slgre(\\d+)_", list_files)]

bsri_subf <- list.dirs(list_bsri, full.names = TRUE, recursive = TRUE)
bsri_mod <- bsri_subf[grepl("mods", bsri_subf)]
bsri_time <- bsri_subf[grepl("times", bsri_subf)]

# copy mods to correct folder
for (f in bsri_mod) {
  file.copy(from = list.files(f, full.names = TRUE),
            to   = paste0(substr(f, 1, 15), "/mods"))
}

# next, extract results

# load in functions for extracting results
source("extract_fxns.R")

## small ##

# read back in data
out1_resp1_re <- read_rds("sim/sim_resp_sm_re.RDS")

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ssmre <- list_files[grepl("_rslurm_ssmre", list_files)]

ssmre_subf <- list.dirs(list_ssmre, full.names = TRUE, recursive = TRUE)
ssmre_mod <- ssmre_subf[grepl("mods", ssmre_subf)]

ssmre_labels <- gsub("\\D", "", ssmre_mod)
ssmre_labels <- ifelse(ssmre_labels == "", 1, as.numeric(ssmre_labels))

# get paths
ssmre_paths <- ssmre_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ssmre_labels)

# extract PIPs
ssmre_pips <- names(ssmre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    ssmre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)

        1:6 |> # for each stratified model
```

```
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod) == "logical") {
              result <- data.frame(variable = NA, PIP = NA,
                                   df = NA, race = z)
            } else {
              result <- data.frame(variable = cnames, PIP = mod$MainPIP) |>
                mutate(df = mod$ns, race = z)
            }
            return(result)
          }) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(ssmre_pips, "sim/re/ssmre_pips.csv")
write.csv(ssmre_pips, "sim/re/ssmre_pips.csv")

# get exposure response relationships
ssmre_univ <- names(ssmre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    ssmre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        trial <- as.numeric(sub(".+_(.+)d.+", "\\1", y))
        if(trial %% 10 == 0) message("index ", y)
        df <- out1_resp1_re[[trial]]
        X <- df |>
          select(As:Sn) |>
          as.matrix.data.frame()
        C <- df |>
          select(smoke:bmi) |>
          as.matrix.data.frame()
        Y <- df$y

        list_mod <- read_rds(y)
        1:6 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(j1val = NA, est = NA,
                               lower = NA, upper = NA, race = z)
            } else {
              # compute estimated Hg-response relationship
              df <- univarsurf_bsr(mod, X, C, j1 = 4) |>
                mutate(race = z)
            }
            # print(df)
            return(df)
          }) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(ssmre_univ, "sim/re/ssmre_expresp.csv")

## large ##

# read back in data
out2_resp1_re <- read_rds("sim/sim_resp_lg_re.RDS")

list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_slgre <- list_files[grepl("_rslurm_slgre", list_files)]
```

```r
slgre_subf <- list.dirs(list_slgre, full.names = TRUE, recursive = TRUE)
slgre_mod <- slgre_subf[grepl("mods", slgre_subf)]

slgre_labels <- gsub("\\D", "", slgre_mod)
slgre_labels <- ifelse(slgre_labels == "", 1, as.numeric(slgre_labels))

# get paths
slgre_paths <- slgre_mod |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = slgre_labels)

# extract PIPs
slgre_pips <- names(slgre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    slgre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        list_mod <- read_rds(y)
        1:6 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod) == "logical") {
              result <- data.frame(variable = NA, PIP = NA,
                                   df = NA, race = z)
            } else {
              result <- data.frame(variable = cnames, PIP = mod$MainPIP) |>
                mutate(df = mod$ns, race = z)
            }
            return(result)
          }) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)))
      }) |>
      mutate(case = x)
  })
write_csv(slgre_pips, "sim/re/slgre_pips.csv")

# get exposure response relationships
slgre_univ <- names(slgre_paths) |>
  purrr::map_df(\(x) { # for each case
    message("starting ", x)
    slgre_paths[[x]] |>
      purrr::map_df(\(y) { # for each trial
        trial <- as.numeric(sub(".+_(.+)d.+", "\\1", y))
        if(trial %% 10 == 0) message("index ", y)
        df <- out2_resp1_re[[trial]]
        X <- df |>
          select(As:Sn) |>
          as.matrix.data.frame()
        C <- df |>
          select(smoke:bmi) |>
          as.matrix.data.frame()
        Y <- df$y

        list_mod <- read_rds(y)
        1:5 |> # for each stratified model
          purrr::map_df(\(z) {
            mod <- list_mod[[z]]
            if(class(mod)[1] == "logical") { # if output failed
              df <- data.frame(j1val = NA, est = NA,
                               lower = NA, upper = NA, race = z)
            } else {
              # compute estimated Hg-response relationship
```

```
            df <- univarsurf_bsr(mod, X, C, j1 = 4) |>
              mutate(race = z)
          }
          # print(df)
          return(df)
        }) |>
        mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)))
    }) |>
    mutate(case = x)
})
write_csv(slgre_univ, "sim/re/slgre_expresp.csv")
```

Finally, we extract information on run-times for BKMR and BSR. Note that run-times for MLRs were already combined in one file earlier, as they were all run on one job sent to the HPC.

```
######
# BKMR run times
#####

## small ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ksm <- list_files[grepl("_rslurm_bkmr_sm", list_files)]

ksm_subf <- list.dirs(list_ksm, full.names = TRUE, recursive = TRUE)
ksm_times <- ksm_subf[grepl("times", ksm_subf)]

ksm_labelt <- gsub("\\D", "", ksm_times)
ksm_labelt <- ifelse(ksm_labelt == "", 1, as.numeric(ksm_labelt))

# get paths
ksm_patht <- ksm_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ksm_labelt)

times <- names(ksm_patht) |>
  map_df(\(x) {
    ksm_patht[[x]] |>
      map_df(\(y) {
        # time = read_rds(y)
        # print(time)
        return(data.frame(
          time = read_rds(y),
          trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y))))
      }) |>
      mutate(case = x)
  })

write_rds(times, "sim/bkmr_sm/times.RDS")

## large ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
```

```
list_klg <- list_files[grepl("_rslurm_bkmr_lg", list_files)]

klg_subf <- list.dirs(list_klg, full.names = TRUE, recursive = TRUE)
klg_times <- klg_subf[grepl("times", klg_subf)]

klg_labelt <- gsub("\\D", "", klg_times)
klg_labelt <- ifelse(klg_labelt == "", 1, as.numeric(klg_labelt))

# get paths
klg_patht <- klg_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = klg_labelt)

times <- names(klg_patht) |>
  map_df(\(x) {
    klg_patht[[x]] |>
      map_df(\(y) {
        # time = read_rds(y)
        # print(time)
        return(data.frame(
          time = read_rds(y),
          trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y))))
      }) |>
      mutate(case = x)
  })

write_rds(times, "sim/bkmr_lg/times.RDS")

######
# BSR run times
######

## small ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ssm <- list_files[grepl("_rslurm_bsr_sm", list_files)]

ssm_subf <- list.dirs(list_ssm, full.names = TRUE, recursive = TRUE)
ssm_times <- ssm_subf[grepl("times", ssm_subf)]

ssm_labelt <- gsub("\\D", "", ssm_times)
ssm_labelt <- ifelse(ssm_labelt == "", 1, as.numeric(ssm_labelt))

# get paths
ssm_patht <- ssm_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ssm_labelt)

times <- names(ssm_patht) |>
  map_df(\(x) {
    ssm_patht[[x]] |>
      map_df(\(y) {
        time = read_rds(y)
        # print(time)
        return(data.frame(
          time_selection = time[[1]],
          time = time[[2]],
          trial = as.numeric(sub(".+_(.+)d.+", "\\1", y))))
      }) |>
```

```
      mutate(case = x)
  })

write_rds(times, "sim/bsr_sm/times.RDS")

## large ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_slg <- list_files[grepl("_rslurm_bsr_lg", list_files)]

slg_subf <- list.dirs(list_slg, full.names = TRUE, recursive = TRUE)
slg_times <- slg_subf[grepl("times", slg_subf)]

slg_labelt <- gsub("\\D", "", slg_times)
slg_labelt <- ifelse(slg_labelt == "", 1, as.numeric(slg_labelt))

# get paths
slg_patht <- slg_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = slg_labelt)

times <- names(slg_patht) |>
  map_df(\(x) {
    slg_patht[[x]] |>
      map_df(\(y) {
        time = read_rds(y)
        # print(time)
        return(data.frame(
          time_selection = time[[1]],
          time = time[[2]],
          trial = as.numeric(sub(".+_(.+)d.+", "\\1", y))))
      }) |>
      mutate(case = x)
  })
write_rds(times, "sim/bsr_lg/times.RDS")

######
# BKMR stratified run times
######

## small ##

# extract run times
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ksmre <- list_files[grepl("_rslurm_ksmre", list_files)]

ksmre_subf <- list.dirs(list_ksmre, full.names = TRUE, recursive = TRUE)
ksmre_times <- ksmre_subf[grepl("times", ksmre_subf)]

ksmre_labelt <- gsub("\\D", "", ksmre_times)
ksmre_labelt <- ifelse(ksmre_labelt == "", 1, as.numeric(ksmre_labelt))

# get paths
ksmre_patht <- ksmre_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ksmre_labelt)

times <- names(ksmre_patht) |>
  map_df(\(x) {
```

```
    ksmre_patht[[x]] |>
      map_df(\(y) {
        list_time <- read_rds(y)
        df <- 1:6 |>
          map_df(\(z) {
            data.frame(time = list_time[[z]],
                       race = z)
          })
        return(mutate(df, trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y))))
      }) |>
      mutate(case = x)
  })

write_rds(times, "sim/re/ksmre_times.RDS")

## large ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_slgre <- list_files[grepl("_rslurm_slgre", list_files)]

klgre_subf <- list.dirs(list_klgre, full.names = TRUE, recursive = TRUE)
klgre_times <- klgre_subf[grepl("times", klgre_subf)]

klgre_labelt <- gsub("\\D", "", klgre_times)
klgre_labelt <- ifelse(klgre_labelt == "", 1, as.numeric(klgre_labelt))

# get paths
klgre_patht <- klgre_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = klgre_labelt)

t <- read_rds(klgre_patht[[1]][1])

times <- names(klgre_patht) |>
  map_df(\(x) {
    klgre_patht[[x]] |>
      map_df(\(y) {
        # time = read_rds(y)
        # print(time)
        list_time <- read_rds(y)
        df <- 1:5 |>
          map_df(\(z) {
            data.frame(time = list_time[[z]],
                       race = z)
          })
        return(mutate(df, trial = as.numeric(sub(".*_(\\d+)\\.RDS", "\\1", y))))
      }) |>
      mutate(case = x)
  })

write_rds(times, "sim/re/klgre_times.RDS")

########
# BSR stratified run-times
#######

## small ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_ssmre <- list_files[grepl("_rslurm_ssmre", list_files)]
```

```
ssmre_subf <- list.dirs(list_ssmre, full.names = TRUE, recursive = TRUE)
ssmre_times <- ssmre_subf[grepl("times", ssmre_subf)]

ssmre_labelt <- gsub("\\D", "", ssmre_times)
ssmre_labelt <- ifelse(ssmre_labelt == "", 1, as.numeric(ssmre_labelt))

# get paths
ssmre_patht <- ssmre_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = ssmre_labelt)

times <- names(ssmre_patht) |>
  map_df(\(x) {
    ssmre_patht[[x]] |>
      map_df(\(y) {
        list_time = read_rds(y)
        df <- 1:5 |>
          map_df(\(z) {
            data.frame(time_selection = list_time[[z]][[1]],
                       time = list_time[[z]][[2]],
                       race = z)
          }) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)))
        return(df)
      }) |>
      mutate(case = x)
  })

write_rds(times, "sim/re/ssmre_times.RDS")

## large ##

# file names
list_files <- list.dirs(".", full.names = FALSE, recursive = FALSE)
list_slgre <- list_files[grepl("_rslurm_slgre", list_files)]

slgre_subf <- list.dirs(list_slgre, full.names = TRUE, recursive = TRUE)
slgre_times <- slgre_subf[grepl("times", slgre_subf)]

slgre_labelt <- gsub("\\D", "", slgre_times)
slgre_labelt <- ifelse(slgre_labelt == "", 1, as.numeric(slgre_labelt))

# get paths
slgre_patht <- slgre_times |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = slgre_labelt)

times <- names(slgre_patht) |>
  map_df(\(x) {
    slgre_patht[[x]] |>
      map_df(\(y) {
        list_time = read_rds(y)
        df <- 1:5 |>
          map_df(\(z) {
            data.frame(time_selection = list_time[[z]][[1]],
                       time = list_time[[z]][[2]],
                       race = z)
          }) |>
          mutate(trial = as.numeric(sub(".+_(.+)d.+", "\\1", y)))
        return(df)
```

```
      }) |>
      mutate(case = x)
  })

write_rds(times, "sim/re/slgre_times.RDS")
```

B.3.2 Presenting results

Here, we create the tables and figures presented in Chapter 4.3. Code for supplemental results included in Appendix A.2 are also included in this section.

We start with some setup.

```
library(tidyverse)
library(latex2exp)

# set up ------------------------------------------------------------------

# set theme for plots
theme_set(theme_light())
theme_update(
  panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  strip.background = element_rect(color="gray", fill="white"),
  strip.text = element_text(color = "gray30")
)

# create labeller
equations1 <-  c(TeX("No inter", output = "character"),
                 TeX("0.35Hg$*$Ni"), TeX("0.13Hg$*($Ni$-1)^2$"),
                 TeX("0.35Cd$*$As"), TeX("0.125Cd$*($As$-1)^2$"),
                 TeX("0.3Ni$*$Co"), TeX("0.1Ni$*($Co$-1)^2$"),
                 TeX("0.3Hg$*$Ni$*$Tl"), TeX("0.09Hg$*($Ni$-1)^2*$Tl"),
                 TeX("0.7Hg$*$Ni"), TeX("0.26Hg$*($Ni$-1)^2$"),
                 TeX("0.7Cd$*$As"), TeX("0.25Cd$*($As$-1)^2$"),
                 TeX("0.6Ni$*$Co"), TeX("0.2Ni$*($Co$-1)^2$"),
                 TeX("0.6Hg$*$Ni$*$Tl"), TeX("0.18Hg$*($Ni$-1)^2*$Tl"))
namesa <- c(1, 2, 4, 6, 8, 10, 12, 14, 16, 3, 5, 7, 9, 11, 13, 15, 17)
appendera <- function(string) {
  return(equations1[match(string, namesa)])
}

# identify true significance based on variable name
get_sign <- function(case, chem) {
  case_when(
    chem %in% c("Hg", "Ni", "Sb", "Sn") ~ TRUE,
    case %in% 6:9 & chem %in% c("Cd", "As") ~ TRUE,
    case %in% 10:13 & chem == "Co" ~ TRUE,
    case %in% 14:17 & chem == "Tl" ~ TRUE,
    .default = FALSE
  )
}

# identify true significance based on index (BSR)
get_sign_bsr <- function(case, chem) {
  case_when(
    case %in% 2:5 & chem %in% c(4, 5) ~ TRUE,
    case %in% 6:9 & chem %in% c(1, 2) ~ TRUE,
```

```
    case %in% 10:13 & chem %in% c(3, 5) ~ TRUE,
    case %in% 14:17 & chem %in% c(4, 5, 6) ~ TRUE,
    .default = FALSE
  )
}

# identify true significance for race by ethnicity models
getsign_re <- function(chem) {
  return(chem %in% c("Hg", "Ni", "Sb", "Sn"))
}

# go from index of race to name
which_race <- function(race) {
  case_when(
    race == 1 ~ "Non-Hisp. white",
    race == 2 ~ "Non-Hisp. black",
    race == 3 ~ "Non-Hisp. other",
    race == 4 ~ "Hispanic born in US",
    race == 5 ~ "Hispanic born outside US",
    race == 6 ~ "Collapsed non-Hisp."
  )
}

# go from index of race to size of category
size_race <- function(race) {
  case_when(
    race == 1 ~ "16",
    race == 2 ~ "27",
    race == 3 ~ "13",
    race == 4 ~ "87",
    race == 5 ~ "109",
    race == 6 ~ "56"
  )
}
```

This code creates output for the base case.

```
# base case ------------------------------------------------------------------

# smaller size
nsm_pval <- read_csv("sim/_mlr/pvalsm.csv")
osm_pval <- read_csv("sim/_oracle/pvalsm.csv")
ksm_pips <- read_csv("sim/bkmr_sm/pips.csv")
ssm_pips <- read_csv("sim/bsr_sm/pips.csv")

# p-value visualization
osm_pvalc <- osm_pval |>
  mutate(var = case_when(
    grepl("Ni", var) ~ "Ni*",
    grepl("Sn", var) ~ "Sn*",
    grepl("I\\(Sb", var) ~ "Sb^2",
    .default = var
  )) |>
  filter(var %in% c("Hg", "Ni*", "Tl", "Pb", "Mo", "Sn*", "Sb^2")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = TRUE,
         mod = "Oracle MLR")
nsm_pvalc <- nsm_pval |>
  filter(var %in% c("As", "Cd", "Co", "Hg", "Ni",
                    "Tl", "Pb", "Mo", "Sb", "Sn")) |>
  mutate(trial = case,
```

```
         case = ceiling(case/100),
         sign = get_sign(case, var),
         mod = "Naive MLR")

base_mlrs <- bind_rows(nsm_pvalc, osm_pvalc) |>
  filter(case == 1)

p1 <- base_mlrs |>
  ggplot(aes(var, p)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  labs(y = "p-value distribution",
       color = "Truly\nsignificant",
       x = "Chemical") +
  facet_grid(.~mod, scales = "free_x", space = "free")

# pip visualization
base_pips <- bind_rows(
  mutate(ksm_pips, mod = "BKMR"),
  mutate(ssm_pips, mod = "BSR")
) |>
  filter(case == 1) |>
  mutate(sign = get_sign(case, variable))

p2 <- base_pips |>
  ggplot(aes(variable, PIP)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey30") +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  labs(y = "PIP distribution",
       color = "Truly significant",
       x = "Chemical") +
  facet_wrap(~mod, scales = "free_x")


# larger size
nlg_pval <- read_csv("sim/_mlr/pvallg.csv")
olg_pval <- read_csv("sim/_oracle/pvallg.csv")
klg_pips <- read_csv("sim/bkmr_lg/pips.csv")
slg_pips <- read_csv("sim/bsr_lg/pips.csv")

# p-val visualization
nlg_pvalc <- nlg_pval |>
  filter(var %in% c("As", "Cd", "Co", "Hg", "Ni",
                    "Tl", "Pb", "Mo", "Sb", "Sn")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = get_sign(case, var),
         mod = "Naive MLR")
olg_pvalc <- olg_pval |>
  mutate(var = case_when(
    grepl("Ni", var) ~ "Ni*",
    grepl("Sn", var) ~ "Sn*",
```

```
    grepl("I\\(Sb", var) ~ "Sb^2",
    .default = var
  )) |>
  filter(var %in% c("Hg", "Ni*", "Tl", "Pb", "Mo", "Sn*", "Sb^2")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = TRUE,
         mod = "Oracle MLR")

base_mlrl <- bind_rows(nlg_pvalc, olg_pvalc) |>
  filter(case == 1)

p3 <- base_mlrl |>
  ggplot(aes(var, p)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  labs(y = "p-value distribution",
       color = "Truly significant",
       x = "Chemical") +
  facet_grid(.~mod, scales = "free_x", space = "free")

# pip visualization
base_pipl <- bind_rows(
  mutate(klg_pips, mod = "BKMR"),
  mutate(slg_pips, mod = "BSR")
) |>
  filter(case == 1) |>
  mutate(sign = get_sign(case, variable))

p4 <- base_pipl |>
  ggplot(aes(variable, PIP)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey30") +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  labs(y = "PIP distribution",
       color = "Truly significant",
       x = "Chemical") +
  facet_wrap(~mod, scales = "free_x")

# plot grid
pone <- cowplot::plot_grid(
  p1 + theme(legend.position = "none"),
  p2 + theme(legend.position = "none"),
  p3 + theme(legend.position = "none"),
  p4 + theme(legend.position = "none"),
  labels = c("a", "c", "b", "d"), nrow = 2, rel_widths = c(4, 5)
)
pone

legend <- cowplot::get_legend(p1)

# stitch together
cowplot::plot_grid(pone, legend, rel_widths = c(5, 0.5))
```

```
ggsave("index/figures/ch4_basecasesig.png", width = 9, height = 5)
```

```
# create table of sensitivities
base_mlr <- bind_rows(
  mutate(base_mlrs, size = "Small"),
  mutate(base_mlrl, size = "Large")
)

sum_base_mlr <- base_mlr |>
  group_by(mod, size, var, sign) |>
  summarize(sensitivity = sum(p < 0.05)/n())

base_bay <- bind_rows(
  mutate(base_pips, size = "Small"),
  mutate(base_pipl, size = "Large")
)

sum_base_bay <- base_bay |>
  rename(var = variable) |>
  group_by(mod, size, var, sign) |>
  summarize(sensitivity = sum(PIP >= 0.5)/n())

sum_base <- bind_rows(sum_base_mlr, sum_base_bay)
write_csv(sum_base, "index/data/base_sens.csv")

base_sens <- read_csv("data/base_sens.csv")

base_sens2 <- base_sens |>
  mutate(var = gsub("[^[:alpha:]]","", var),
         mod = factor(mod, levels = c("Naive MLR", "Oracle MLR", "BKMR", "BSR"))) |>
  arrange(desc(sign), var, mod, desc(size)) |>
  select(-sign) |>
  pivot_wider(names_from = var, values_from = sensitivity)

base_mod_ord <- table(base_sens2$mod)

options(knitr.kable.NA = '-')
base_sens2 |>
  select(-mod) |>
  kbl(booktabs = TRUE, escape = FALSE,
      col.names = c("", names(base_sens2)[3:12]),
      align = "lccccccccc",
      caption = "Sensitivity and false discovery rate (FDR) of chemicals in base case scenario.") |>
  column_spec(1, width = "6em") |>
  add_header_above(header = c(" " = 1, "Sensitivity" = 4, "FDR" = 6)) |>
  pack_rows(index = base_mod_ord)
```

This code formats results for the naive MLR on scenarios with interactions between chemicals.

```
# naive mlr ---------------------------------------------------------------

# p-values small
nsm_pval <- read_csv("sim/_mlr/pvalsm.csv")
nsm_pvalc <- nsm_pval |>
  filter(var %in% c("As", "Cd", "Co", "Hg", "Ni",
                    "Tl", "Pb", "Mo", "Sb", "Sn")) |>
  mutate(trial = case,
         case = ceiling(case/100),
```

```
         sign = get_sign(case, var),
         mod = "Naive MLR")
nsm_pvalc_sens <- nsm_pvalc |>
  mutate(imp = p < 0.05) |>
  group_by(case, var) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2),
            upper = quantile(p, 0.75))

# line plot of p-values
nsm_pvalc |>
  filter(case != 1) |>
  ggplot(aes(x = var, y = p)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(nsm_pvalc_sens, case != 1),
             mapping = aes(x = var, y = upper+0.1, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(0.05, "lines")) +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(y = "p-value distribution",
       color = "Truly\nsignificant",
       x = "Chemical") +
  facet_wrap(~case,
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_nsm_univ_pval.png", width = 7.5, height = 5)

# p-values large
nlg_pval <- read_csv("sim/_mlr/pvallg.csv")
nlg_pvalc <- nlg_pval |>
  filter(var %in% c("As", "Cd", "Co", "Hg", "Ni",
                    "Tl", "Pb", "Mo", "Sb", "Sn")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = get_sign(case, var),
         mod = "Naive MLR")
nlg_pvalc_sens <- nlg_pvalc |>
  mutate(imp = p < 0.05) |>
  group_by(case, var) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2),
            upper = quantile(p, 0.75))

# line plot of p-values
nlg_pvalc |>
  filter(case != 1) |>
  ggplot(aes(x = var, y = p)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(nlg_pvalc_sens, case != 1),
           mapping = aes(x = var, y = upper+0.1, label = sensitivity),
           size = 2, label.size = NA, label.padding = unit(0.05, "lines")) +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(y = "p-value distribution",
```

```
       color = "Truly\nsignificant",
       x = "Chemical") +
  facet_wrap(~case,
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_nlg_univ_pval.png", width = 7.5, height = 5)

# save sensitivity as table
naive_all <- bind_rows(
  mutate(nsm_pvalc, size = "Small", mod = "Naive MLR"),
  mutate(nlg_pvalc, size = "Large", mod = "Naive MLR")
) |>
  group_by(mod, case, size, var, sign) |>
  summarize(sensitivity = sum(p < 0.05)/n())
write_csv(naive_all, "index/data/naive_sens.csv")
```

This code formats results for the oracle MLR in scenarios with interactions between chemicals.

```
# oracle mlr ------------------------------------------------------------------

# p-values small
osm_pval <- read_csv("sim/_oracle/pvalsm.csv")
osm_pvalc <- osm_pval |>
  mutate(var = case_when(
    grepl("Ni", var) ~ "Ni*",
    grepl("Sn", var) ~ "Sn*",
    grepl("I\\(Sb", var) ~ "Sb^2",
    .default = var
  )) |>
  filter(var %in% c("Hg", "Ni*", "Tl", "Pb", "Mo", "Sn*", "Sb^2")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = TRUE)
osm_pvalc_sens <- osm_pvalc |>
  mutate(imp = p < 0.05) |>
  group_by(case, var) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2),
            upper = quantile(p, 0.75))

# line plot of p-values
osm_pvalc |>
  filter(case != 1) |>
  ggplot(aes(x = var, y = p)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(osm_pvalc_sens, case != 1),
             mapping = aes(x = var, y = upper+0.12, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(0.05, "lines")) +
  geom_pointrange(aes(color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2, color = "darkorange2") +
  labs(y = "p-value distribution",
       color = "Truly\nsignificant",
       x = "Chemical") +
  facet_wrap(~case,
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
```

```
ggsave("index/figures/ch4_osm_univ_pval.png", width = 7.5, height = 5)

# interactions

keepnames <- c('(Intercept)', 'Hg', 'Sb', 'I(1/(1 + exp(-4 * Ni)))', 'Ni',
               'I(Sb^2)', 'I(1/(1 + exp(-4 * Sn)))', 'age', 'bmi',
               'race2', 'race3', 'race4', 'race5', 'smoke1', 'Cd', 'As', 'Co')

osm_pvali <- osm_pval |>
  filter(!(var %in% keepnames)) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = get_sign(case, var))
osm_pvali_sens <- osm_pvali |>
  mutate(imp = p < 0.05) |>
  group_by(case, var) |>
  summarize(sensitivity = sum(imp)/n())

# density plot of p-values
osm_pvali |>
  ggplot(aes(x = p)) +
  geom_density(color = "darkorange2", fill = "darkorange", alpha = 0.1) +
  geom_vline(xintercept = 0.05, linetype = "dashed", color = "grey30") +
  labs(x = "p-value distribution") +
  facet_wrap(~case, scales = "free_y",
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_osm_biv_pval.png", width = 7.5, height = 5)

# put interactions and univ together
osm_comb <- bind_rows(
  mutate(osm_pvali, variable = "Int"),
  mutate(osm_pvalc[osm_pvalc$case != 1, ], variable = var)
)

osm_comb |>
  mutate(variable = factor(variable, levels = c(unique(osm_pvalc$var), "Int")),
         interaction = (variable == "Int")) |>
  ggplot(aes(x = variable, y = p, color = interaction)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_pointrange(stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2"))+
  labs(y = "p-value distribution",
       color = "Interaction",
       x = "Term") +
  facet_wrap(~case,
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_osm_pval.png", width = 7.5, height = 5)

# p-values large
olg_pval <- read_csv("sim/_oracle/pvallg.csv")
olg_pvalc <- olg_pval |>
  mutate(var = case_when(
    grepl("Ni", var) ~ "Ni*",
    grepl("Sn", var) ~ "Sn*",
    grepl("I\\(Sb", var) ~ "Sb^2",
    .default = var
  )) |>
  filter(var %in% c("Hg", "Ni*", "Tl", "Pb", "Mo", "Sn*", "Sb^2")) |>
```

```
  mutate(trial = case,
         case = ceiling(case/100),
         sign = get_sign(case, var))
olg_pvalc_sens <- olg_pvalc |>
  mutate(imp = p < 0.05) |>
  group_by(case, var) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2),
            upper = quantile(p, 0.75))

# line plot of p-values
olg_pvalc |>
  filter(case != 1) |>
  ggplot(aes(x = var, y = p)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(olg_pvalc_sens, case != 1),
             mapping = aes(x = var, y = upper+0.1, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(0.05, "lines")) +
  geom_pointrange(stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2, color = "darkorange2") +
  labs(y = "p-value distribution",
       color = "Truly significant",
       x = "Chemical") +
  facet_wrap(~case,
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_olg_univ_pval.png", width = 7.5, height = 5)

# interactions

keepnames <- c('(Intercept)', 'Hg', 'Sb', 'I(1/(1 + exp(-4 * Ni)))', 'Ni',
               'I(Sb^2)', 'I(1/(1 + exp(-4 * Sn)))', 'age', 'bmi',
               'race2', 'race3', 'race4', 'race5', 'smoke1', 'Cd', 'As', 'Co')

olg_pvali <- olg_pval |>
  filter(!(var %in% keepnames)) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = get_sign(case, var))
olg_pvali_sens <- olg_pvali |>
  mutate(imp = p < 0.05) |>
  group_by(case, var) |>
  summarize(sensitivity = sum(imp)/n())

# density plot of p-values
olg_pvali |>
  ggplot(aes(x = p)) +
  geom_density(color = "darkorange2", fill = "darkorange", alpha = 0.1) +
  geom_vline(xintercept = 0.05, linetype = "dashed", color = "grey30") +
  labs(x = "p-value distribution") +
  facet_wrap(~case, scales = "free_y",
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_olg_biv_pval.png", width = 7.5, height = 5)

# put together
olg_comb <- bind_rows(
  mutate(olg_pvali, variable = "Int"),
  mutate(olg_pvalc[olg_pvalc$case != 1, ], variable = var)
)

olg_comb |>
```

```
  mutate(variable = factor(variable, levels = c(unique(olg_pvalc$var), "Int")),
         interaction = (variable == "Int")) |>
  ggplot(aes(x = variable, y = p, color = interaction)) +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "grey30") +
  geom_pointrange(stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2"))+
  labs(y = "p-value distribution",
       color = "Interaction",
       x = "Term") +
  facet_wrap(~case,
             labeller = labeller(
               case = as_labeller(appendera, default = label_parsed)))
ggsave("index/figures/ch4_olg_pval.png", width = 7.5, height = 5)

# save sensitivity as table
oracle_all <- bind_rows(
  mutate(osm_comb, size = "Small", mod = "Oracle MLR"),
  mutate(olg_comb, size = "Large", mod = "Oracle MLR")
) |>
  group_by(mod, case, size, var, variable) |>
  summarize(sensitivity = sum(p < 0.05)/n())
write_csv(oracle_all, "index/data/oracle_sens.csv")
```

This code formats results from BKMR models in scenarios with interactions between chemicals.

```
# smaller size bkmr ------------------------------------------------------

# plot pips bkmr small
ksm_pips <- read_csv("sim/bkmr_sm/pips.csv")

ksm_pip_sig <- ksm_pips |>
  mutate(sign = get_sign(case, variable))
ksm_pip_sen <- ksm_pips |>
  mutate(imp = PIP >= 0.5) |>
  group_by(case, variable) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2),
            upper = quantile(PIP, 0.75))

# point and lineplot
ksm_pip_sig |>
  filter(case != 1) |>
  ggplot(aes(x = variable)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(ksm_pip_sen, case != 1),
             mapping = aes(x = variable, y = upper+0.1, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(0, "lines")) +
  geom_pointrange(aes(y = PIP, color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
```

```
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(y = "PIP value distribution",
       color = "Truly\nsignificant",
       x = "Chemical")
ggsave("index/figures/ch4_ksm_univ_pips.png", width = 7.5, height = 5)

# plot bivariate relationships bkmr small
ksm_biv <- read_csv("sim/bkmr_sm/biv_expresp.csv")

# plot Hg and Ni
ksm_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(variable1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_ksm_biv_expresp_1.png", width = 6, height = 4)

# plot Cd and As
ksm_biv |>
  filter(case %in% 6:9) |>
  mutate(variable1 = ifelse(variable1 == "Cd", "Cd by As", "As by Cd"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-4, 4) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_ksm_biv_expresp_2.png", width = 6, height = 4)

# plot Ni and Co
ksm_biv |>
  filter(case %in% 10:13) |>
  mutate(variable1 = ifelse(variable1 == "Ni", "Ni by Co", "Co by Ni"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
```

```
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_ksm_biv_expresp_3.png", width = 6, height = 4)

# plot trivariate relationships bkmr small
ksm_triv <- read_csv("sim/bkmr_sm/triv_expresp.csv")
ksm_triv <- ksm_triv |>
  mutate(variable1 = case_when(
    z1_name == "Hg" ~ "Hg by Ni + Tl",
    z1_name == "Ni" ~ "Ni by Hg + Tl",
    z1_name == "Tl" ~ "Tl by Hg + Ni"),
  quantile = as.factor(z23_q))

ksm_triv |>
  ggplot(aes(z1_val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2+3\nquantile") +
  theme(strip.text.x = element_text(size = 7))
ggsave("index/figures/ch4_ksm_triv_expresp.png", width = 6, height = 4)

# one vs. other significant data
ksm_intb <- read_csv("sim/bkmr_sm/int_bivar.csv")

ksm_intb <- ksm_intb |>
  mutate(cond = paste0(z1, "+", z2)) |>
  rowwise() |>
  mutate(signif = ifelse(
    between(0, est - 1.96*sd, est + 1.96*sd), FALSE, TRUE
  ))

# one vs. two others significant data
ksm_intt <- read_csv("sim/bkmr_sm/int_trivar.csv")

ksm_intt <- ksm_intt |>
  mutate(cond = paste0(variable, " by ", fixedat1, "+", fixedat2)) |>
  rowwise() |>
  mutate(signif = ifelse( #bonferroni correction: critical value 2.39398
    between(0, est - 2.39398*sd, est + 2.39398*sd), FALSE, TRUE
  ))

# bivar and trivar together
int_combs <- bind_rows(
  select(ksm_intb, cond, trial, case, signif),
  select(ksm_intt, cond, trial, case, signif)
)

# larger size bkmr ----------------------------------------------------

# plot pips bkmr large
klg_pips <- read_csv("sim/bkmr_lg/pips.csv")

klg_pip_sig <- klg_pips |>
  mutate(sign = get_sign(case, variable))
klg_pip_sen <- klg_pips |>
  mutate(imp = PIP >= 0.5) |>
  group_by(case, variable) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2),
```

```
            upper = quantile(PIP, 0.75))

# point and lineplot
klg_pip_sig |>
  filter(case != 1) |>
  ggplot(aes(x = variable)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(klg_pip_sen, case != 1),
             mapping = aes(x = variable, y = upper+0.1, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(.05, "lines")) +
  geom_pointrange(aes(y = PIP, color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(y = "PIP value distribution",
       color = "Truly\nsignificant",
       x = "Chemical")
ggsave("index/figures/ch4_klg_univ_pips.png", width = 7.5, height = 5)

# plot bivariate relationships bkmr large
klg_biv <- read_csv("sim/bkmr_lg/biv_expresp.csv")

# plot Hg and Ni
klg_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(variable1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_klg_biv_expresp_1.png", width = 6, height = 4)

# plot Cd and As
klg_biv |>
  filter(case %in% 6:9) |>
  mutate(variable1 = ifelse(variable1 == "Cd", "Cd by As", "As by Cd"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-4, 4) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
```

```
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_klg_biv_expresp_2.png", width = 6, height = 4)

# plot Ni and Co
klg_biv |>
  filter(case %in% 10:13) |>
  mutate(variable1 = ifelse(variable1 == "Ni", "Ni by Co", "Co by Ni"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_klg_biv_expresp_3.png", width = 6, height = 4)

# plot trivariate relationships bkmr large
klg_triv <- read_csv("sim/bkmr_lg/triv_expresp.csv")
klg_triv <- klg_triv |>
  mutate(variable1 = case_when(
    z1_name == "Hg" ~ "Hg by Ni + Tl",
    z1_name == "Ni" ~ "Ni by Hg + Tl",
    z1_name == "Tl" ~ "Tl by Hg + Ni"),
    quantile = as.factor(z23_q))

klg_triv |>
  ggplot(aes(z1_val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2+3\nquantile") +
  theme(strip.text.x = element_text(size = 7))
ggsave("index/figures/ch4_klg_triv_expresp.png", width = 6, height = 4)

# one vs. other significant data
klg_intb <- read_csv("sim/bkmr_lg/int_bivar.csv")

klg_intb <- klg_intb |>
  mutate(cond = paste0(z1, "+", z2)) |>
  rowwise() |>
  mutate(signif = ifelse(
    between(0, est - 1.96*sd, est + 1.96*sd), FALSE, TRUE
  ))

# one vs. two others significant data
klg_intt <- read_csv("sim/bkmr_lg/int_trivar.csv")

klg_intt <- klg_intt |>
  mutate(cond = paste0(variable, " by ", fixedat1, "+", fixedat2)) |>
  rowwise() |>
  mutate(signif = ifelse( #bonferroni correction: critical value 2.39398
    between(0, est - 2.39398*sd, est + 2.39398*sd), FALSE, TRUE
```

```
  ))

# bivar and trivar together
int_combl <- bind_rows(
  select(klg_intb, cond, trial, case, signif),
  select(klg_intt, cond, trial, case, signif)
)

# bkmr combine sensitivity ------------------------------------------------

# univariate pips
bkmr_pips <- bind_rows(
  mutate(ksm_pips, size = "Small"),
  mutate(klg_pips, size = "Large")
) |>
  group_by(size, case, variable) |>
  summarize(sensitivity = sum(PIP >= 0.5)/n()) |>
  mutate(sign = get_sign(case, variable))
write_csv(bkmr_pips, "index/data/bkmr_pip_sens.csv")

# interactions
bkmr_comb <- bind_rows(
  mutate(int_combs, size = "Small"),
  mutate(int_combl, size = "Large")
)

bkmr_sens <- bkmr_comb |>
  group_by(size, case, cond) |>
  summarize(sensitivity = sum(signif)/n())
write_csv(bkmr_sens, "index/data/bkmr_int_sens.csv")
```

This code formats results from BSR models in scenarios with interactions between chemicals.

```
# smaller size bsr --------------------------------------------------------

# plot pip's bsr small
ssm_pips <- read_csv("sim/bsr_sm/pips.csv")

ssm_pip_sig <- ssm_pips |>
  mutate(sign = get_sign(case, variable))
ssm_pip_sen <- ssm_pips |>
  mutate(imp = PIP >= 0.5) |>
  group_by(case, variable) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2,
                                      max = 2),
            upper = quantile(PIP, 0.75))

# point and lineplot
ssm_pip_sig |>
  filter(case != 1) |>
  ggplot(aes(x = variable)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(ssm_pip_sen, case != 1),
             mapping = aes(x = variable, y = upper+0.1, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(0.05, "lines")) +
  geom_pointrange(aes(y = PIP, color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
```

```
                fun = median,
                size = 0.2) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  # scale_y_continuous(sec.axis = sec_axis(trans = ~., name = "Sensitivity")) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
        # legend.position = c(0.95, 0.05)) +
  labs(y = "PIP value distribution",
       color = "Truly\nsignificant",
       x = "Chemical")
ggsave("index/figures/ch4_ssm_univ_pips.png", width = 7.5, height = 5)

# plot bivariate pip's small
ssm_pipb <- read_csv("sim/bsr_sm/pip_biv.csv")

ssm_pipb |>
  # filter(case == 2) |>
  mutate(sign = get_sign_bsr(case, Var1) & get_sign_bsr(case, Var2),
         inter = paste0(Var1, "_", Var2)) |>
  ggplot(aes(x = inter)) +
  geom_pointrange(aes(y = PIP, color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z, 0.25)},
                  fun.max = function(z) {quantile(z, 0.75)},
                  fun = median,
                  size  = 0.05) +
  scale_color_manual(values = c("#92D0E4", "darkorange2")) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = c(0.9, 0.1)) +
  labs(y = "PIP value distribution",
       color = "Truly significant",
       x = NULL)
ggsave("index/figures/ch4_ssm_biv_pips.png", width = 7.5, height = 5)

# plot trivariate pip's small
ssm_pipt <- read_csv("sim/bsr_sm/pip_triv.csv")

ssm_pipt |>
  ggplot(aes(x = "")) +
  geom_pointrange(aes(y = PIP),
                  stat = "summary",
                  fun.min = function(z) {quantile(z, 0.25)},
                  fun.max = function(z) {quantile(z, 0.75)},
                  fun = median,
                  size  = 0.05) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = c(0.9, 0.1)) +
  labs(y = "PIP value distribution",
       color = "Truly significant",
       x = NULL)

# plot bivariate relationships bkmr small
ssm_biv <- read_csv("sim/bsr_sm/biv_expresp.csv")
```

```
# plot Hg and Ni
ssm_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(j1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-3, 9) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_ssm_biv_expresp_1.png", width = 6, height = 4)

# plot Cd and As
ssm_biv |>
  filter(case %in% 6:9) |>
  mutate(variable1 = ifelse(j1 == "Cd", "Cd by As", "As by Cd"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-2, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_ssm_biv_expresp_2.png", width = 6, height = 4)

# plot Ni and Co
ssm_biv |>
  filter(case %in% 10:13) |>
  mutate(variable1 = ifelse(j1 == "Ni", "Ni by Co", "Co by Ni"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  # ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_ssm_biv_expresp_3.png", width = 6, height = 4)

# plot trivariate relationships bsr small
ssm_triv <- read_csv("sim/bsr_sm/triv_expresp.csv")
ssm_triv <- ssm_triv |>
  mutate(variable1 = case_when(
    j1 == "Hg" ~ "Hg by Ni + Tl",
    j1 == "Ni" ~ "Ni by Hg + Tl",
    j1 == "Tl" ~ "Tl by Hg + Ni"),
```

```
    quantile = as.factor(j23quant))

ssm_triv |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2+3\nquantile") +
  theme(strip.text.x = element_text(size = 7))
ggsave("index/figures/ch4_ssm_triv_expresp.png", width = 6, height = 4)


# larger size bsr ---------------------------------------------------------

# plot pip's bsr large
slg_pips <- read_csv("sim/bsr_lg/pips.csv")

slg_pip_sig <- slg_pips |>
  mutate(sign = get_sign(case, variable))
slg_pip_sen <- slg_pips |>
  mutate(imp = PIP >= 0.5) |>
  group_by(case, variable) |>
  summarize(sensitivity = weights::rd(sum(imp)/n(), 2, max = 0),
            upper = quantile(PIP, 0.75))

# point and lineplot
slg_pip_sig |>
  filter(case != 1) |>
  ggplot(aes(x = variable)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey30") +
  geom_label(data = filter(slg_pip_sen, case != 1),
             mapping = aes(x = variable, y = upper+0.1, label = sensitivity),
             size = 2, label.size = NA, label.padding = unit(0.05, "lines")) +
  geom_pointrange(aes(y = PIP, color = sign),
                  stat = "summary",
                  fun.min = function(z) {quantile(z,0.25)},
                  fun.max = function(z) {quantile(z,0.75)},
                  fun = median,
                  size = 0.2) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  scale_color_manual(values = c("deepskyblue3", "darkorange2")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(y = "PIP value distribution",
       color = "Truly\nsignificant",
       x = "Chemical")
ggsave("index/figures/ch4_slg_univ_pips.png", width = 7.5, height = 5)

# plot bivariate pip's large
slg_pipb <- read_csv("sim/bsr_lg/pip_biv.csv")

slg_pipb |>
  # filter(case == 2) |>
  mutate(sign = get_sign_bsr(case, Var1) & get_sign_bsr(case, Var2),
         inter = paste0(Var1, "_", Var2)) |>
  ggplot(aes(x = inter)) +
  geom_pointrange(aes(y = PIP, color = sign),
```

```
                  stat = "summary",
                  fun.min = function(z) {quantile(z, 0.25)},
                  fun.max = function(z) {quantile(z, 0.75)},
                  fun = median,
                  size  = 0.05) +
  scale_color_manual(values = c("#92D0E4", "darkorange2")) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = c(0.9, 0.1)) +
  labs(y = "PIP value distribution",
       color = "Truly significant",
       x = NULL)
ggsave("index/figures/ch4_slg_biv_pips.png", width = 7.5, height = 5)

# plot trivariate pip's large
slg_pipt <- read_csv("sim/bsr_lg/pip_triv.csv")

slg_pipt |>
  ggplot(aes(x = "")) +
  geom_pointrange(aes(y = PIP),
                  stat = "summary",
                  fun.min = function(z) {quantile(z, 0.25)},
                  fun.max = function(z) {quantile(z, 0.75)},
                  fun = median,
                  size  = 0.05) +
  facet_wrap(~case,
             labeller = as_labeller(appendera,
                                    default = label_parsed)) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = c(0.9, 0.1)) +
  labs(y = "PIP value distribution",
       color = "Truly significant",
       x = NULL)

# plot bivariate relationships bkmr large
slg_biv <- read_csv("sim/bsr_lg/biv_expresp.csv")

# plot Hg and Ni
slg_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(j1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-3, 9) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_slg_biv_expresp_1.png", width = 6, height = 4)

# plot Cd and As
slg_biv |>
  filter(case %in% 6:9) |>
  mutate(variable1 = ifelse(j1 == "Cd", "Cd by As", "As by Cd"),
```

```
        quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  ylim(-2, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_slg_biv_expresp_2.png", width = 6, height = 4)

# plot Ni and Co
slg_biv |>
  filter(case %in% 10:13) |>
  mutate(variable1 = ifelse(j1 == "Ni", "Ni by Co", "Co by Ni"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  # ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")
ggsave("index/figures/ch4_slg_biv_expresp_3.png", width = 6, height = 4)

# plot trivariate relationships bsr large
slg_triv <- read_csv("sim/bsr_lg/triv_expresp.csv")
slg_triv <- slg_triv |>
  mutate(variable1 = case_when(
    j1 == "Hg" ~ "Hg by Ni + Tl",
    j1 == "Ni" ~ "Ni by Hg + Tl",
    j1 == "Tl" ~ "Tl by Hg + Ni"),
    quantile = as.factor(j23quant))

slg_triv |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2+3\nquantile") +
  theme(strip.text = element_text(size = 7))
ggsave("index/figures/ch4_slg_triv_expresp.png", width = 6, height = 4)


# bsr combine sens ------------------------------------------------------

# univariate pips
bsr_pips <- bind_rows(
  mutate(ssm_pips, size = "Small"),
```

```
  mutate(slg_pips, size = "Large")
) |>
  group_by(size, case, variable) |>
  summarize(sensitivity = sum(PIP >= 0.5)/n()) |>
  mutate(sign = get_sign(case, variable))
write_csv(bsr_pips, "index/data/bsr_pip_sens.csv")

# interactions
cnames <- c("As", "Cd", "Co", "Hg", "Ni", "Tl", "Pb", "Mo", "Sb", "Sn")

bsr_comb <- bind_rows(
  mutate(ssm_pipb, size = "Small"),
  mutate(slg_pipb, size = "Large")
) |>
  mutate(sign = get_sign_bsr(case, Var1) & get_sign_bsr(case, Var2),
         v1 = cnames[Var1], v2 = cnames[Var2],
         inter = paste0(v1, "-", v2)) |>
  mutate(inter2 = ifelse(sign, inter, "none")) |>
  group_by(size, case, inter2, sign) |>
  summarize(sensitivity = sum(PIP >= 0.5)/n())
write_csv(bsr_comb, "index/data/bsr_int_sens.csv")
```

This code creates tables with univariate sensitivity and false discovery rates in scenarios with interactions between chemicals.

```
naive_sens <- read_csv("data/naive_sens.csv")
oracle_sens <- read_csv("data/oracle_sens.csv")
bkmr_sens <- read_csv("data/bkmr_pip_sens.csv")
bsr_sens <- read_csv("data/bsr_pip_sens.csv")

comb_sens <- bind_rows(
  rename(naive_sens, variable = var),
  filter(select(oracle_sens,-var), variable != "Int"),
  mutate(bkmr_sens, mod = "BKMR"),
  mutate(bsr_sens, mod = "BSR")
) |>
  filter(case != 1) |>
  mutate(
    variable = gsub("[^[:alpha:]]", "", variable),
    mod = factor(mod, levels = c("Naive MLR", "Oracle MLR", "BKMR", "BSR")) ,
    sign = ifelse(mod == "Oracle MLR", TRUE, sign),
    new_sign = factor(ifelse(
      variable %in% c("Hg", "Ni", "Sb", "Sn"),
      "Sen",
      "FDR"
    ), levels = c("Sen", "FDR")),
    cond = case_when(
      case %in% 2:5 ~ "Hg-Ni",
      case %in% 6:9 ~ "Cd-As",
      case %in% 10:13 ~ "Ni-Co",
      .default = "Hg-Ni-Tl"
    ),
    cond = factor(cond, levels = c("Hg-Ni", "Cd-As", "Ni-Co", "Hg-Ni-Tl")),
    inter_type = ifelse(case %% 4 %in% 1:2, "Mult.", "Poly."),
    effect_size = factor(
      ifelse(case %% 2 == 1, "Higher", "Lower"),
      levels = c("Lower", "Higher")
    )
  ) |>
  group_by(cond, inter_type, effect_size, mod, size, new_sign) |>
```

```
  summarize(detection = mean(sensitivity)) |>
  arrange(new_sign, desc(size)) |>
  pivot_wider(names_from = c(mod, size, new_sign), values_from = detection,
              names_sep = " ") |>
  ungroup()

comb_sens_ord <- table(comb_sens$cond)

# sensitivities
comb_sens |>
  select(-cond) |>
  select(1:10) |>
  kbl(booktabs = TRUE, digits = 2, #align = "lcc",
      caption = "Overall sensitivity for univariate chemicals in all scenarios with interactions between chemicals. Multiplicati
      col.names = c("Type", "Effect size",
                    rep(c("Naive", "Oracle", "BKMR", "BSR"), 2))
      ) |>
  add_header_above(header = c(" " = 2, "Small (n=252)" = 4, "Large (n=1000)" = 4), bold = TRUE) |>
  pack_rows(index = comb_sens_ord)

# false discovery rates
comb_sens |>
  select(-cond) |>
  select(c(1, 2, 11:16)) |>
  kbl(booktabs = TRUE, digits = 2, #align = "lcc",
      caption = "Overall false discovery rate for univariate chemicals in all scenarios with interactions between chemicals.",
      col.names = c("Type", "Effect size",
                    rep(c("Naive", "BKMR", "BSR"), 2))
      ) |>
  add_header_above(header = c(" " = 2, "Small" = 3, "Large" = 3)) |>
  # add_header_above(header = c(" " = 2, "Sensitivity" = 8, "False discovery rate" = 6)) |>
  pack_rows(index = comb_sens_ord)
```

This code creates a figure with the full estimated exposure-response relationship output for Hg and Ni only, which is included in the main results section

```
# HgNi only ------------------------------------------------------------------

# bivariate surfaces

# bkmr small
hgni3 <- ksm_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(variable1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  theme(strip.text = element_text(size = 8)) +
  ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")

# bkmr large
```

```r
hgni4 <- klg_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(variable1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(quantile)) |>
  ggplot(aes(z1, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  theme(strip.text = element_text(size = 8)) +
  ylim(-6, 6) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")

# bsr small
hgni5 <- ssm_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(j1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  theme(strip.text = element_text(size = 8)) +
  ylim(-3, 9) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")

# bsr large
hgni6 <- slg_biv |>
  filter(case %in% 2:5) |>
  mutate(variable1 = ifelse(j1 == "Hg", "Hg by Ni", "Ni by Hg"),
         quantile = as.factor(j2quant)) |>
  ggplot(aes(j1val, est, color = quantile)) +
  geom_line(aes(group = interaction(trial, quantile)), alpha = 0.2) +
  ggh4x::facet_grid2(variable1~case, scales = "free", independent = "x",
                     labeller = labeller(
                       case = as_labeller(appendera, default = label_parsed))) +
  scale_color_manual(values = c("deepskyblue3", "palegreen3", "darkorange"),
                     guide = guide_legend(override.aes = list(alpha = 1),
                                          reverse = TRUE)) +
  theme(strip.text = element_text(size = 8)) +
  ylim(-3, 9) +
  labs(x = "Chem 1 value",
       y = "Estimated response",
       color = "Chem 2\nquantile")

hgnitwo <- cowplot::plot_grid(
  hgni3 + theme(legend.position = "none"),
  hgni4 + theme(legend.position = "none"),
  hgni5 + theme(legend.position = "none"),
  hgni6 + theme(legend.position = "none"),
  labels = "auto", nrow = 2
)
hgnitwo
```

```
hgnilegend2 <- cowplot::get_legend(hgni3 + theme(legend.position = "bottom"))

# stitch together
cowplot::plot_grid(hgnitwo, hgnilegend2, nrow = 2, rel_heights = c(5, 0.4))
ggsave("index/figures/ch4_hgni_biv.png", width = 9, height = 6)
```

This code creates a table of sensitivities for two-way interactions between chemicals.

```
oracle_comb <- read_csv("data/oracle_sens.csv")
bkmr_comb <- read_csv("data/bkmr_int_sens.csv")
bsr_comb <- read_csv("data/bsr_int_sens.csv")

comb_int <- bind_rows(
  oracle_comb |> filter(variable == "Int") |>
    select(-variable, cond = var),
  bkmr_comb |> mutate(mod = "BKMR"),
  bsr_comb |> mutate(mod = "BSR") |> filter(sign) |> rename(cond = inter2) |> select(-sign)
) |>
  filter(case %in% 2:13) |>
  mutate(cond = gsub(":", "-", cond),
         cond = gsub("\\+", "-", cond),
         cond = case_when(
           cond == "I(Hg * ((Ni - 1)^2))" ~ "Hg-Ni",
           cond == "I(Cd * ((As - 1)^2))" ~ "Cd-As",
           cond == "I(Ni * ((Co - 1)^2))" ~ "Ni-Co",
           cond == "As-Cd" ~ "Cd-As",
           cond == "Co-Ni" ~ "Ni-Co",
           .default = cond
         ),
         cond = factor(cond, levels = c("Hg-Ni", "Cd-As", "Ni-Co")),
         inter_type = ifelse(case %in% c(2, 3, 6, 7, 10, 11), "Multiplicative", "Polynomial"),
         effect_size = factor(ifelse(case %% 2 == 1, "Higher", "Lower"),
                              levels = c("Lower", "Higher"))) |>
  select(-case) |>
  arrange(desc(size)) |>
  pivot_wider(names_from = c(mod, size), values_from = sensitivity,
              names_sep = " ") |>
  arrange(cond, inter_type, effect_size)

# create table
comb_int_ord <- table(comb_int$cond)
comb_int |>
  select(-cond) |>
  kbl(booktabs = TRUE, escape = FALSE,
      align = "llccccccc",
      caption = "Sensitivity to interactions in all scenarios with two-way interactions between exposures.",
      col.names = c("Interaction type", "Effect size", rep(c("Oracle", "BKMR", "BSR"), 2))) |>
  add_header_above(header = c(" " = 2, "Small (n=252)" = 3, "Large (n=1000)" = 3), bold = TRUE) |>
  pack_rows(index = comb_int_ord) |>
  collapse_rows(columns = 1, valign = "middle", latex_hline = "linespace")
```

This code creates a flow chart of detection of univariate and interaction signals for scenarios with larger size datasets and higher effect size, multiplicative interactions between Cd and As and Ni and Co.

```r
library(ggalluvial)

# Cd-As

# extract from bkmr
bkmr_comb1 <- bkmr_comb |>
  filter(cond == "As+Cd", case == 7) |>
  select(case, size, trial, sign = signif, variable = cond)
ksm_pip_cdas <- ksm_pip_sig |>
  filter(case == 7, variable %in% c("As", "Cd")) |>
  mutate(size = "Small") |>
  select(case, size, trial, PIP, variable)
klg_pip_cdas <- klg_pip_sig |>
  filter(case == 7, variable %in% c("As", "Cd")) |>
  mutate(size = "Large") |>
  select(case, size, trial, PIP, variable)

# combine
bkmr_cdas <- bind_rows(bkmr_comb1, ksm_pip_cdas, klg_pip_cdas) |>
  mutate(mod = "BKMR")

# extract from bsr
bsr_comb1 <- bind_rows(
  mutate(ssm_pipb, size = "Small"),
  mutate(slg_pipb, size = "Large")
) |>
  mutate(sign = get_sign_bsr(case, Var1) & get_sign_bsr(case, Var2),
         v1 = cnames[Var1], v2 = cnames[Var2],
         inter = paste0(v1, "-", v2)) |>
  mutate(inter2 = ifelse(sign, inter, "none"))
ssm_pip_cdas <- ssm_pip_sig |>
  filter(case == 7, variable %in% c("As", "Cd")) |>
  mutate(mod = "BSR", size = "Small") |>
  select(case, size, trial, PIP, variable)
slg_pip_cdas <- slg_pip_sig |>
  filter(case == 7, variable %in% c("As", "Cd")) |>
  mutate(mod = "BSR", size = "Large") |>
  select(case, size, trial, PIP, variable)

# combine
bsr_cdas <- bsr_comb1 |>
  filter(inter2 == "As-Cd", case == 7) |>
  select(case, size, trial, PIP, variable = inter2) |>
  bind_rows(ssm_pip_cdas, slg_pip_cdas) |>
  mutate(mod = "BSR")

# bind Cd As together
all_cdas <- bind_rows(bkmr_cdas, bsr_cdas) |>
  mutate(sign = ifelse(is.na(PIP), sign, PIP >= 0.5),
         variable = gsub("\\+", "-", variable)) |>
  select(-PIP) |>
  pivot_wider(names_from = variable, values_from = sign) |>
  mutate(univ = case_when(
    As & Cd ~ "As & Cd",
    As | Cd~ "As only/\nCd only",
    .default = "Neither"
  )) |>
  rename(interaction = 5) |>
  mutate(univ = factor(univ, levels = c("As & Cd", "As only/\nCd only", "Neither")),
         interaction = factor(interaction, levels = c(TRUE, FALSE)))

# alluvial plot for cd as
large_cdas <- all_cdas |>
  filter(size == "Large") |>
```

```
  group_by(mod, univ, interaction) |>
  count()
cdasp <- large_cdas |>
  rename(Univariate = univ, Interaction = interaction)  |>
  ggplot(aes(axis1 = Univariate, axis2 = Interaction, y = n)) +
  ggalluvial::geom_alluvium(aes(fill = mod), alpha = 0.4) +
  ggalluvial::geom_stratum(color = "gray50") +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)),
            lineheight = 0.75) +
  scale_x_discrete(limits = c("Univariate", "Interaction"), expand = c(0.2, 0.05)) +
  scale_fill_manual(values = c("deepskyblue3", "darkorange")) +
  labs(fill = "Model")

# Ni-Co

# extract from bkmr
bkmr_comb2 <- bkmr_comb |>
  filter(cond == "Co+Ni", case == 11) |>
  select(case, size, trial, sign = signif, variable = cond)
ksm_pip_nico <- ksm_pip_sig |>
  filter(case == 11, variable %in% c("Ni", "Co")) |>
  mutate(size = "Small") |>
  select(case, size, trial, PIP, variable)
klg_pip_nico <- klg_pip_sig |>
  filter(case == 11, variable %in% c("Ni", "Co")) |>
  mutate(size = "Large") |>
  select(case, size, trial, PIP, variable)

bkmr_nico <- bind_rows(bkmr_comb2, ksm_pip_nico, klg_pip_nico) |>
  mutate(mod = "BKMR")

# extract from bsr
bsr_comb2 <- bind_rows(
  mutate(ssm_pipb, size = "Small"),
  mutate(slg_pipb, size = "Large")
) |>
  mutate(sign = get_sign_bsr(case, Var1) & get_sign_bsr(case, Var2),
         v1 = cnames[Var1], v2 = cnames[Var2],
         inter = paste0(v1, "-", v2)) |>
  mutate(inter2 = ifelse(sign, inter, "none"))
ssm_pip_nico <- ssm_pip_sig |>
  filter(case == 11, variable %in% c("Ni", "Co")) |>
  mutate(mod = "BSR", size = "Small") |>
  select(case, size, trial, PIP, variable)
slg_pip_nico <- slg_pip_sig |>
  filter(case == 11, variable %in% c("Ni", "Co")) |>
  mutate(mod = "BSR", size = "Large") |>
  select(case, size, trial, PIP, variable)

bsr_nico <- bsr_comb2 |>
  filter(inter2 == "Co-Ni", case == 11) |>
  select(case, size, trial, PIP, variable = inter2) |>
  bind_rows(ssm_pip_nico, slg_pip_nico) |>
  mutate(mod = "BSR")

# bind together
all_nico <- bind_rows(bkmr_nico, bsr_nico) |>
  mutate(sign = ifelse(is.na(PIP), sign, PIP >= 0.5),
         variable = gsub("\\+", "-", variable)) |>
  select(-PIP) |>
  pivot_wider(names_from = variable, values_from = sign) |>
  mutate(univ = case_when(
    Ni & Co ~ "Ni & Co",
    Ni ~ "Ni only",
```

```
    Co ~ "Co only",
    .default = "Neither"
  )) |>
  rename(interaction = 5) |>
  mutate(univ = factor(univ, levels = c("Ni & Co", "Ni only", "Co only", "Neither")),
         interaction = factor(interaction, levels = c(TRUE, FALSE)))

# alluvial plot for Ni Co
large_nico <- all_nico |>
  filter(size == "Large") |>
  group_by(mod, univ, interaction) |>
  count()
nicop <- large_nico |>
  rename(Univariate = univ, Interaction = interaction)  |>
  ggplot(aes(axis1 = Univariate, axis2 = Interaction, y = n)) +
  ggalluvial::geom_alluvium(aes(fill = mod), alpha = 0.4) +
  ggalluvial::geom_stratum(color = "gray50") +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)),
            lineheight = 0.75) +
  scale_x_discrete(limits = c("Univariate", "Interaction"), expand = c(0.2, 0.05)) +
  scale_fill_manual(values = c("deepskyblue3", "darkorange")) +
  labs(fill = "Model")

# combine nico and cdas together
cowplot::plot_grid(cdasp + theme(legend.position = "none"),
                   nicop, rel_widths = c(4.5, 6), labels = "auto")
ggsave("index/figures/ch4_flowchartboth.png", width = 8, height = 4)
```

This code creates a table of false discovery rates for two-way interactions between chemicals.

```
# bkmr false discovery rates
ksmfdr<- read_csv("sim/bkmr_sm/int_bivarfull.csv")
klgfdr <- read_csv("sim/bkmr_lg/int_bivarfull.csv")

bkmr_comb_fdr <- bind_rows(
  mutate(ksmfdr, size = "Small"),
  mutate(klgfdr, size = "Large")
) |>
  rowwise() |>
  mutate(inter1 = paste0(z1, "-", z2),
         sign = ifelse(
           (inter1 == "Hg-Ni" & case %in% 2:5) |
             (inter1 == "As-Cd" & case %in% 6:9) |
             (inter1 == "Co-Ni" & case %in% 10:13), TRUE, FALSE
         ),
         signif = ifelse(
           between(0, est - 1.96*sd, est + 1.96*sd), FALSE, TRUE
         ),
         inter2 = ifelse(sign, inter1, "none")) |>
  group_by(size, case, inter2, sign) |>
  summarize(sensitivity = sum(signif)/n())

# bsr false discovery rates
bsr_comb_fdr <- read_csv("index/data/bsr_int_sens.csv")

# combine
fdr_comb <- bind_rows(
  mutate(bkmr_comb_fdr, mod = "BKMR"),
  mutate(bsr_comb_fdr, mod = "BSR")
```

```
) |>
  relocate(mod)
write_csv(fdr_comb, "index/data/comb_int_fdr.csv")

# read data back in
fdr_comb <- read_csv("data/comb_int_fdr.csv")

fdr_comb2 <- fdr_comb |>
  filter(inter2 == "none", case %in% 2:13) |>
  mutate(
    cond = case_when(
      case %in% 2:5 ~ "Hg-Ni",
      case %in% 6:9 ~ "Cd-As",
      case %in% 10:13 ~ "Ni-Co"
    ),
    cond = factor(cond, levels = c("Hg-Ni", "Cd-As", "Ni-Co")),
    inter_type = ifelse(case %in% c(2, 3, 6, 7, 10, 11), "Multiplicative", "Polynomial"),
    effect_size = factor(
      ifelse(case %% 2 == 1, "Higher", "Lower"),
      levels = c("Lower", "Higher")
    )
  ) |>
  select(-inter2, -sign, -case) |>
  pivot_wider(names_from = c(mod, size), values_from = sensitivity) |>
  arrange(cond, inter_type, effect_size) |>
  relocate(cond, inter_type, effect_size, BKMR_Small, BSR_Small, BKMR_Large, BSR_Large)

# create table
fdr_ord <- table(fdr_comb2$cond)
fdr_comb2 |>
  select(-cond) |>
  kbl(booktabs = TRUE, escape = FALSE,
      align = "llcccc", digits = 4,
      caption = "False discovery rate of interactions in all scenarios with two-way interactions between exposures.",
      col.names = c("Interaction type", "Effect size", rep(c("BKMR", "BSR"), 2))) |>
  add_header_above(header = c(" " = 2, "Small (n=252)" = 2, "Large (n=1000)" = 2), bold = TRUE) |>
  pack_rows(index = fdr_ord) |>
  collapse_rows(columns = 1, valign = "middle", latex_hline = "linespace")
```

This code creates a table of sensitivities for three-way interactions.

```
# get trivariate sensitivity from oracle, bkmr, bsr
oracle_senst <- bind_rows(
  filter(mutate(osm_comb, size = "Small"), case %in% 14:17),
  filter(mutate(olg_comb, size = "Large"), case %in% 14:17)
) |>
  filter(variable == "Int") |>
  group_by(case, size) |>
  summarize(sensitivity = sum(p < 0.05)/n()) |>
  mutate(mod = "Oracle") |>
  relocate(mod)
bkmr_senst <- bkmr_comb |>
  filter(case %in% 14:17) |>
  pivot_wider(names_from = cond, values_from = signif) |>
  mutate(signif = unname(pick(4) | pick(5) | pick(6))) |>
  rename(signif = 7) |>
  group_by(case, size) |>
  summarize(sensitivity = sum(signif)/n()) |>
  mutate(mod = "BKMR") |>
  relocate(mod)
bsr_senst <- bind_rows(
  mutate(ssm_pipt, size = "Small"),
```

```
  mutate(slg_pipt, size = "Large")
) |>
  group_by(case, size) |>
  summarize(sensitivity = sum(PIP >= 0.5)/n()) |>
  mutate(mod = "BSR") |>
  relocate(mod)

# bind together
all_senst <- bind_rows(oracle_senst, bkmr_senst, bsr_senst) |>
  mutate(inter_type = ifelse(case %in% c(14, 15), "Multiplicative", "Polynomial"),
         effect_size = ifelse(case %in% c(14, 16), "Lower", "Higher"))
write_csv(all_senst, "sim/tables/triv_sens.csv")

# read back in
triv <- read_csv("data/triv_sens.csv")

# create table
triv |>
  select(-case) |>
  pivot_wider(names_from = c(mod, size), values_from = sensitivity) |>
  relocate(1, 2, 4, 6, 8, 3, 5, 7) |>
  kbl(booktabs = TRUE,
      caption = "Sensitivity to trivariate interactions between Hg, Ni, and Tl.",
      col.names = c("Interaction type", "Effect size", rep(c("Oracle", "BKMR", "BSR"), 2))) |>
  add_header_above(header = c(" " = 2, "Small (n=252)" = 3, "Large (n=1000)" = 3), bold = TRUE) |>
  collapse_rows(columns = 1, valign = "middle", latex_hline = "linespace")
```

This code formats results for scenarios with an interaction between race and Hg.

```
# naive ------------------------------------------------------------------

nsmre_pval <- read_csv("sim/_mlr/pvalsmre.csv")
nlgre_pval <- read_csv("sim/_mlr/pvallgre.csv")

# p-values small
nsmre_pvalc <- nsmre_pval |>
  filter(var %in% c("As", "Cd", "Co", "Hg", "Ni",
                    "Tl", "Pb", "Mo", "Sb", "Sn")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = getsign_re(var),
         mod = "Naive MLR")

# p-values large
nlgre_pvalc <- nlgre_pval |>
  filter(var %in% c("As", "Cd", "Co", "Hg", "Ni",
                    "Tl", "Pb", "Mo", "Sb", "Sn")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = getsign_re(var),
         mod = "Naive MLR")

# save sensitivity as table
naive_allre <- bind_rows(
  mutate(nsmre_pvalc, size = "Small", mod = "Naive MLR"),
  mutate(nlgre_pvalc, size = "Large", mod = "Naive MLR")
) |>
  group_by(mod, case, size, var, sign) |>
  summarize(sensitivity = sum(p < 0.05)/n())
write_csv(naive_allre, "index/data/naive_re_sens.csv")
```

```
# oracle ----------------------------------------------------------------

# interaction terms
keepnames <- c('(Intercept)', 'Hg', 'Sb', 'I(1/(1 + exp(-4 * Ni)))', 'Ni',
               'I(Sb^2)', 'I(1/(1 + exp(-4 * Sn)))', 'age', 'bmi',
               'race2', 'race3', 'race4', 'race5', 'smoke1', 'Cd', 'As', 'Co')

# small
osmre_pval <- read_csv("sim/_oracle/pvalsmre.csv")

# univariate p-values
osmre_pvalc <- osmre_pval |>
  mutate(var = case_when(
    grepl("Ni", var) ~ "Ni*",
    grepl("Sn", var) ~ "Sn*",
    grepl("I\\(Sb", var) ~ "Sb^2",
    .default = var
  )) |>
  filter(var %in% c("Hg", "Ni*", "Tl", "Pb", "Mo", "Sn*", "Sb^2")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = TRUE)

osmre_pvali <- osmre_pval |>
  filter(!(var %in% keepnames)) |>
  mutate(trial = case,
         case = ceiling(case/100))

# combine univariate and interactions
osmre_comb <- bind_rows(
  mutate(osmre_pvali, variable = "Int"),
  mutate(osmre_pvalc, variable = var)
)

# large
olgre_pval <- read_csv("sim/_oracle/pvallgre.csv")

# univariate p-values
olgre_pvalc <- olgre_pval |>
  mutate(var = case_when(
    grepl("Ni", var) ~ "Ni*",
    grepl("Sn", var) ~ "Sn*",
    grepl("I\\(Sb", var) ~ "Sb^2",
    .default = var
  )) |>
  filter(var %in% c("Hg", "Ni*", "Tl", "Pb", "Mo", "Sn*", "Sb^2")) |>
  mutate(trial = case,
         case = ceiling(case/100),
         sign = TRUE)

olgre_pvali <- olgre_pval |>
  filter(!(var %in% keepnames)) |>
  mutate(trial = case,
         case = ceiling(case/100))

# combine univariate and interactions
olgre_comb <- bind_rows(
  mutate(olgre_pvali, variable = "Int"),
  mutate(olgre_pvalc, variable = var)
)

# combine both oracle outputs
oracle_re <- bind_rows(
  mutate(osmre_comb, size = "Small"),
```

```
  mutate(olgre_comb, size = "Large")
) |>
  select(size, case, trial, variable, p) |>
  arrange(case, trial, desc(size))

write_csv(oracle_re, "index/data/oracle_re_sens.csv")

# bkmr ------------------------------------------------------------------

library(intervals)
# critical value of 2.57583 for bonferroni corrected 95% CI

# function for checking overlap of confidence intervals
check_overlap <- function(data) {
  num_na <- sum(is.na(data$lower))
  sign <- data$sign
  unsign <- !(data$sign)
  sign_intervals <- Intervals(na.omit(data[sign, c("lower", "upper")]))
  unsign_intervals <- Intervals(na.omit(data[unsign, c("lower", "upper")]))
  return(data.frame(
    num_na = num_na,
    sig_overlaps = length(interval_overlap(sign_intervals, unsign_intervals)[[1]]),
    insig_overlaps = (length(unlist(interval_overlap(unsign_intervals, unsign_intervals))) ==
                        (length(unsign_intervals)/2)^2)
  ))
}

# detection of interactions

# small
ksmre_ints <- read_csv("sim/re/ksmre_ints.csv")

# uncollapsed
ksmre_det1 <- ksmre_ints |>
  filter(race != 6) |>
  mutate(lower = est - 2.57583*sd,
         upper = est + 2.57583*sd,
         sign = (case <= 2 & race == 2) | (case >= 3 & race == 5)) |>
  group_by(case, trial) |>
  group_modify(~ check_overlap(.))

# check number of models where at least one was unable to be fitted
table(ksmre_det1$num_na != 0)

# use collapsed
ksmre_det2 <- ksmre_ints |>
  filter(race %in% c(4, 5, 6)) |>
  mutate(lower = est - 2.57583*sd,
         upper = est + 2.57583*sd,
         sign = (case <= 2 & race == 6) | (case >= 3 & race == 5)) |>
  group_by(case, trial) |>
  group_modify(~ check_overlap(.))

# large
klgre_ints <- read_csv("sim/re/klgre_ints.csv")

klgre_det <- klgre_ints |>
  mutate(lower = est - 2.57583*sd,
         upper = est + 2.57583*sd,
         sign = (case <= 2 & race == 2) | (case >= 3 & race == 5)) |>
  group_by(case, trial) |>
  group_modify(~ check_overlap(.))

# save the trials that produced significant results
```

```r
klgre_det |>
  filter(sig_overlaps != 4) |>
  write_csv("sim/re/trials_klg_sig.csv")

# get sensitivity
ksmre_sens1 <- ksmre_det1 |>
  group_by(case) |>
  summarize(sens = sum((sig_overlaps/(4-num_na)) != 1 & insig_overlaps)/n()) |>
  mutate(size = "Small uncollapsed")
ksmre_sens2 <- ksmre_det2 |>
  group_by(case) |>
  summarize(sens = sum((sig_overlaps/(2-num_na)) != 1 & insig_overlaps)/n()) |>
  mutate(size = "Small collapsed")
# no trials that had a significant non-overlap AlSO had non-significant non-overlap
klgre_sens <- klgre_det |>
  group_by(case) |>
  summarize(sens = sum(sig_overlaps != 4 & insig_overlaps)/n()) |>
  mutate(size = "Large")

# combine sensitivities together
bkmrre_senscomb <- bind_rows(ksmre_sens1, ksmre_sens2, klgre_sens)
write_csv(bkmrre_senscomb, "index/data/bkmr_re_sens.csv")

# pip's
ksmre_pips <- read_csv("sim/re/ksmre_pips.csv")
klgre_pips <- read_csv("sim/re/klgre_pips.csv")

# summarize pip's
ksmre_pips2 <- ksmre_pips |>
  filter(is.na(variable) | variable == "Hg") |>
  group_by(case, race) |>
  summarize(num_na = sum(is.na(variable))/100,
            hg_detect = sum(PIP >= 0.5, na.rm = T)/100)

klgre_pips2 <- klgre_pips |>
  filter(is.na(variable) | variable == "Hg") |>
  group_by(case, race) |>
  summarize(num_na = sum(is.na(variable))/n(),
            hg_detect = sum(PIP >= 0.5, na.rm = T)/n())

bkmrre_pipscomb <- bind_rows(
  mutate(ksmre_pips2, size = "Small"),
  mutate(klgre_pips2, size = "Large")
)  |>
  mutate(which_race = which_race(race),
         size_race = size_race(race))
write_csv(bkmrre_pipscomb, "index/data/bkmr_re_pips.csv")


# bsr ----------------------------------------------------------------------

# pip's

# pip's
ssmre_pips <- read_csv("sim/re/ssmre_pips.csv")
slgre_pips <- read_csv("sim/re/slgre_pips.csv")

# summarize pip's
ssmre_pips2 <- ssmre_pips |>
  filter(is.na(variable) | variable == "Hg") |>
  group_by(case, race) |>
  summarize(num_na = sum(is.na(variable))/100,
            hg_detect = sum(PIP >= 0.5, na.rm = T)/100)
```

```
slgre_pips2 <- slgre_pips |>
  filter(is.na(variable) | variable == "Hg") |>
  group_by(case, race) |>
  summarize(num_na = sum(is.na(variable))/n(),
            hg_detect = sum(PIP >= 0.5, na.rm = T)/n())

bsrre_pipscomb <- bind_rows(
  mutate(ssmre_pips2, size = "Small"),
  mutate(slgre_pips2, size = "Large")
)  |>
  mutate(which_race = which_race(race),
         size_race = size_race(race))
write_csv(bsrre_pipscomb, "index/data/bsr_re_pips.csv")
```

This code creates a table of sensitivities for interactions between race and Hg.

```
# table
bkmr_re_sens <- read_csv("data/bkmr_re_sens.csv") |>
  rename(sensitivity = sens)
oracle_re_int <- read_csv("data/oracle_re_sens.csv") |>
  filter(variable == "Int") |>
  group_by(size, case) |>
  summarize(sensitivity = sum(p<0.05)/n()) |>
  mutate(size = ifelse(size == "Small", "Small uncollapsed", size))

re_ints <- bind_rows(
  mutate(bkmr_re_sens, mod = "BKMR"),
  mutate(oracle_re_int, mod = "Oracle MLR")
) |>
  arrange(desc(size), desc(mod)) |>
  pivot_wider(names_from = c(mod, size), values_from = sensitivity) |>
  mutate(effect_size = ifelse(case %in% c(1, 3), "Lower", "Higher"),
         case = ifelse(case %in% c(1, 2),
                       paste0("Original n=27", footnote_marker_symbol(2)),
                       paste0("Oirignal n=100", footnote_marker_symbol(3)))
         ) |>
  relocate(case, effect_size)

re_ints |> kbl(booktabs = TRUE, escape = FALSE,
      align = "llccccc",
      caption = "Sensitivity to interactions between the categorical race variable and Hg.",
      col.names = c("Interaction in", "Effect size",
                    "Oracle", "BKMR", "BKMR",
                    "Oracle", "BKMR")
      ) |>
  add_header_above(header = c(" " = 2, "Uncollapsed" = 2,
                              "Collapsed*" = 1,
                              " " = 2)) |>
  add_header_above(header = c(" " = 2, "Small (n=252)" = 3, "Large (n=1000)" = 2),
                   bold = TRUE) |>
  collapse_rows(columns = 1, valign = "middle", latex_hline = "linespace") |>
  # column_spec(7, width = "6em") |>
  add_footnote(c(paste0("\"Collapsed\" refers to scenarios where the smallest three race categories",
                        "are collapsed into one stratified model."),
                 "Non-Hispanic black", "Hispanic born outside US"), notation = "symbol", threeparttable = TRUE)
```

This code plots the estimated exposure-response relationship in scenarios with an interaction between race and Hg.

```r
# exposure response relationship ----------------------------------------

#### BKMR #####

# small
ksmre_expresp <- read_csv("sim/re/ksmre_expresp.csv")
# large
klgre_expresp <- read_csv("sim/re/klgre_expresp.csv")
# combine
bkmr_re_expresp <- bind_rows(
  mutate(ksmre_expresp, size = "Small"),
  mutate(klgre_expresp, size = "Large")
) |>
  filter(race != 6)

# first two cases
kre12 <- bkmr_re_expresp |>
  filter(case <= 2) |>
  mutate(which_race = which_race(race),
         size_race = size_race(race),
         size = factor(size, levels = c("Small", "Large")))

k12 <- kre12 |>
  mutate(
    race2 = case_when(race == 2 ~ 5,
                      race == 5 ~ 2,
                      .default = race),
    race2 = factor(
      race2,
      levels = c(1, 2, 3, 4, 5),
      labels = c(1, 5, 3, 4, 2)
    ),
    case = factor(ifelse(case == 1, "Lower", "Higher"),
                  levels = c("Lower", "Higher"))
  ) |>
  ggplot(aes(z1, est, color = race2)) +
  geom_line(aes(group = interaction(trial, race2)), alpha = 0.15) +
  ylim(-8, 8) +
  facet_grid(size ~ case) +
  scale_color_manual(values = rev(c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6", "#E76BF3")),
                     breaks = c(1, 2, 3, 4, 5)) +
  guides(color = guide_legend(override.aes = list(alpha = 1))) +
  labs(x = "Hg", y = "Estimate", subtitle = "BKMR, interaction in Non-Hispanic black")

# second two cases
kre34 <- bkmr_re_expresp |>
  filter(case > 2) |>
  mutate(which_race = which_race(race),
         size_race = size_race(race),
         size = factor(size, levels = c("Small", "Large")))

k34 <- kre34 |>
  mutate(
    race2 = factor(
      race,
      levels = c(1, 2, 3, 4, 5),
    ),
    case = factor(ifelse(case == 3, "Lower", "Higher"),
                  levels = c("Lower", "Higher"))
  ) |>
  ggplot(aes(z1, est, color = race2)) +
  geom_line(aes(group = interaction(trial, race2)), alpha = 0.15) +
  ylim(-8, 8) +
  facet_grid(size ~ case) +
```

```
  scale_color_manual(values = rev(c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6", "#E76BF3")),
                     breaks = c(1, 2, 3, 4, 5)) +
  guides(color = guide_legend(override.aes = list(alpha = 1))) +
  labs(x = "Hg", y = "Estimate", subtitle = "BKMR, interaction in Hispanic born outside US")


##### BSR ######

# small
ssmre_expresp <- read_csv("sim/re/ssmre_expresp.csv")
# large
slgre_expresp <- read_csv("sim/re/slgre_expresp.csv")
# combine
bsr_re_expresp <- bind_rows(
  mutate(ssmre_expresp, size = "Small"),
  mutate(slgre_expresp, size = "Large")
) |>
  filter(race != 6)

# first two cases
sre12 <- bsr_re_expresp |>
  filter(case <= 2) |>
  mutate(which_race = which_race(race),
         size_race = size_race(race),
         size = factor(size, levels = c("Small", "Large")))

s12 <- sre12 |>
  mutate(
    race2 = case_when(race == 2 ~ 5,
                      race == 5 ~ 2,
                      .default = race),
    race2 = factor(
      race2,
      levels = c(1, 2, 3, 4, 5),
      labels = c(1, 5, 3, 4, 2)
    ),
    case = factor(ifelse(case == 1, "Lower", "Higher"),
                  levels = c("Lower", "Higher"))
  ) |>
  ggplot(aes(j1val, est, color = race2)) +
  geom_line(aes(group = interaction(trial, race2)), alpha = 0.15) +
  ylim(-6, 12) +
  facet_grid(size ~ case) +
  scale_color_manual(values = rev(c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6", "#E76BF3")),
                     breaks = c(1, 2, 3, 4, 5)) +
  guides(color = guide_legend(override.aes = list(alpha = 1))) +
  labs(x = "Hg", y = "Estimate", subtitle = "BSR, interaction in Non-Hispanic black")

# second two cases
sre34 <- bsr_re_expresp |>
  filter(case > 2) |>
  mutate(which_race = which_race(race),
         size_race = size_race(race),
         size = factor(size, levels = c("Small", "Large")))

s34 <- sre34 |>
  mutate(
    race2 = factor(
      race,
      levels = c(1, 2, 3, 4, 5),
      labels = c("Non-Hisp. white", "Non-Hisp. other", "Non-Hisp. black",
                 "Hispanic born in US", "Hispanic born outside US")
    ),
    case = factor(ifelse(case == 3, "Lower", "Higher"),
```

```
                  levels = c("Lower", "Higher"))
  ) |>
  ggplot(aes(j1val, est, color = race2)) +
  geom_line(aes(group = interaction(trial, race2)), alpha = 0.15) +
  ylim(-6, 12) +
  facet_grid(size ~ case) +
  scale_color_manual(values = rev(c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6", "#E76BF3"))) +
  guides(color = guide_legend(override.aes = list(alpha = 1))) +
  labs(x = "Hg", y = "Estimate", color = "Race",
       subtitle = "BSR, interaction in Hispanic born outside US") +
  theme(legend.position = "bottom")

re_expresp <- cowplot::plot_grid(
  k12 + theme(legend.position = "none"),
  s12 + theme(legend.position = "none"),
  k34 + theme(legend.position = "none"),
  s34 + theme(legend.position = "none"),
  nrow = 2
)

reerlegend <- cowplot::get_legend(s34)
cowplot::plot_grid(re_expresp, reerlegend, rel_heights = c(5, 0.35), ncol = 1)
ggsave("index/figures/ch4_re_expresp.png", width = 7.75, height = 6)
```

This code creates a table of sensitivities for the univariate Hg term in scenarios where it interacts with race.

```
# read in data
naive_re_sens <- read_csv("data/naive_re_sens.csv") |>
  filter(var == "Hg") |>
  select(-sign, -var)
oracle_re_sens <- read_csv("data/oracle_re_sens.csv") |>
  filter(variable == "Hg") |>
  group_by(size, case) |>
  summarize(sensitivity = sum(p<0.05)/n())
bkmr_re_pips <- read_csv("data/bkmr_re_pips.csv") |>
  rename(sensitivity = hg_detect) |>
  select(-num_na)
bsr_re_pips <- read_csv("data/bsr_re_pips.csv") |>
  filter(size != "Large" | race != 6) |>
  rename(sensitivity = hg_detect) |>
  select(-num_na)

# combine together
univ_re_comb <- bind_rows(
  naive_re_sens,
  mutate(oracle_re_sens, mod = "Oracle MLR"),
  mutate(bkmr_re_pips, mod = "BKMR"),
  mutate(bsr_re_pips, mod = "BSR")
)

# pivot wider
univ_re_wide <- univ_re_comb |>
  arrange(desc(size)) |>
  pivot_wider(names_from = case, values_from = sensitivity) |>
  mutate(
    which_race = gsub("Hispanic", "Hisp.", which_race),
    race = ifelse(is.na(which_race), NA,
                  paste0(which_race, " (n=", size_race, ")"))
  ) |>
```

```
  select(-which_race, -size_race) |>
  relocate(size)

# print
univre_ord <- table(univ_re_wide$size)[c(2, 1)]
options(knitr.kable.NA = '-')

univ_re_wide |>
  select(-size) |>
  kbl(booktabs = TRUE, escape = FALSE,
      align = "llcccc",
      caption = paste0("Sensitivity for the univariate Hg term in all scenarios with an interaction", "
                       between the categorical race covariate and Hg. Sensitivities for BKMR and BSR", "
                       models are stratified by race."),
      col.names = c("Model", "Race", rep(c("Lower", "Higher"), 2))
      ) |>
  add_header_above(header = c(" " = 2, "Small race cat." = 2, "Large race cat." = 2), bold = TRUE) |>
  pack_rows(index = univre_ord) |>
  collapse_rows(columns = 1, valign = "middle", latex_hline = "linespace")
```

B.4 Code for Appendix A.1

Finally, we create the supplemental methods output that is included in Appendix A.1.

First, we visualize relationships between significant univariate exposures and the response.

```
library(tidyverse)

p1 <- ggplot(NULL) +
  geom_function(fun = function(x) x,
                color = "darkorchid1") +
  xlim(-2, 2) +
  labs(x = "Hg")

p2 <- ggplot(NULL) +
  geom_function(fun = function(x) 3/(1+exp(-4*x)),
                color = "deepskyblue3") +
  xlim(-2, 2) +
  labs(x = "Ni")

p3 <- ggplot(NULL) +
  geom_function(fun = function(x) 1.5/(1+exp(-4*x)),
                color = "darkorange") +
  xlim(-2, 2) +
  labs(x = "Sn")

p4 <- ggplot(NULL) +
  geom_function(fun = function(x) (x^2) + 0.5*x,
                color = "coral1") +
  xlim(-2, 2) +
  labs(x = "Sb")
```

```
cowplot::plot_grid(p1, p2, p3, p4)
ggsave("index/figures/univariatelines.png", width = 6, height = 4)
```

Next, we visualize 3D surfaces for pair-wise interactions between exposures.

```
# load packages
library(tidyverse)
library(plotly)
# use reticulate to save plotly 3d plots
if(!require(reticulate)) {
  install.packages('reticulate')
  reticulate::install_miniconda()
  reticulate::conda_install('r-reticulate', 'python-kaleido')
  reticulate::conda_install('r-reticulate', 'plotly', channel = 'plotly')
  Sys.setenv(RETICULATE_PYTHON =
               '/Users/elizabethzhang/Library/r-miniconda-arm64/envs/r-reticulate/bin/python')
  reticulate::use_miniconda('r-reticulate')
}
```

```
# load in observed data
comb_small <- read_csv("madres_data/base_data.csv")

# log-transform and scale target data
comb_scale <- comb_small |>
  mutate(across(10:19, ~scale(log(.))))

# check ranges of scaled predictors
range(comb_scale$Hg)
range(comb_scale$Ni)
range(comb_scale$Cd)
range(comb_scale$As)
range(comb_scale$Co)

# generate data covering 2d predictor surface
data <- expand.grid(x1 = seq(-3, 3, by = 0.1),
                    x2 = seq(-3, 3, by = 0.1))
x1 <- data$x1
x2 <- data$x2

# function to create plot
create_plot <- function(xax, yax, Y, xname = NA, yname = NA) {
  plot_ly(x = ~xax, y = ~yax, z = ~Y, intensity = ~Y) |>
    add_trace(type = "mesh3d") |>
    layout(scene = list(
      xaxis = list(rangemode = "normal",
                   showgrid = FALSE,
                   showline = TRUE,
                   mirror = TRUE,
                   ticks = "outside",
                   title = xname),
      yaxis = list(rangemode = "normal",
                   showgrid = FALSE,
                   showline = TRUE,
                   mirror = TRUE,
                   ticks = "outside",
                   title = yname),
      zaxis = list(rangemode = "normal",
                   showgrid = FALSE,
                   showline = TRUE,
                   mirror = TRUE,
```

```
                  ticks = "outside",
                  title = "Y"),
      aspectmode = "cube"
    ))
}

##########
# marginally significant (Hg and Ni)
##########

# no interaction
y00 <- with(data, x1 + 3/(1+exp(-4*x2)))
fig00 <- create_plot(x1, x2, y00, "Hg", "Ni")  |>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = 0.1))))
fig00
save_image(fig00, "index/figures/surfaces/p00.png",
           width = 720, height = 480, scale = 3)

# multiplicative interaction, smaller effect size
yam1 <- with(data, x1 + 3/(1+exp(-4*x2)) + 0.35*x1*x2)
figam1 <- create_plot(x1, x2, yam1, "Hg", "Ni") |>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = .1))))
figam1
save_image(figam1, "index/figures/surfaces/am1.png",
           width = 720, height = 480, scale = 3)

# multiplicative interaction, larger effect size
yam2 <- with(data, x1 + 3/(1+exp(-4*x2)) + 0.75*x1*x2)
figam2 <- create_plot(x1, x2, yam2, "Hg", "Ni") |>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = .1))))
figam2
save_image(figam2, "index/figures/surfaces/am2.png",
           width = 720, height = 480, scale = 3)

# polynomial interaction, smaller effect size
yap1 <- with(data, x1 + 3/(1+exp(-4*x2)) + 0.13*x1*((x2-1)^2))
figap1 <- create_plot(x1, x2, yap1, "Hg", "Ni")|>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = .1))))
figap1
save_image(figap1, "index/figures/surfaces/ap1.png",
           width = 720, height = 480, scale = 3)

# polynomial interaction, larger effect size
yap2 <- with(data, x1 + 3/(1+exp(-4*x2)) + 0.26*x1*((x2-1)^2))
figap2 <- create_plot(x1, x2, yap2, "Hg", "Ni") |>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = .1))))
figap2
save_image(figap2, "index/figures/surfaces/ap2.png",
           width = 720, height = 480, scale = 3)

##########
# marginally insignificant (Cd and As)
##########

# multiplicative interaction, smaller effect size
ybm1 <- with(data, 0.35*x1*x2)
figbm1 <- create_plot(x1, x2, ybm1, "Cd", "As") |>
  layout(scene = list(camera = list(eye = list(x = 1.4, y = 1.4, z = 1.2))))
figbm1
save_image(figbm1, "index/figures/surfaces/bm1.png",
           width = 720, height = 480, scale = 3)

# multiplicative interaction, larger effect size
ybm2 <- with(data, 0.7*x1*x2)
```

```
figbm2 <- create_plot(x1, x2, ybm2, "Cd", "As") |>
  layout(scene = list(camera = list(eye = list(x = 1.4, y = 1.4, z = 1.2))))
figbm2
save_image(figbm2, "index/figures/surfaces/bm2.png",
           width = 720, height = 480, scale = 3)

# polynomial interaction, smaller effect size
ybp1 <- with(data, 0.125*x1*((x2-1)^2))
figbp1 <- create_plot(x1, x2, ybp1, "Cd", "As") |>
  layout(scene = list(camera = list(eye = list(x = 1.4, y = 1.4, z = 1.2))))
figbp1
save_image(figbp1, "index/figures/surfaces/bp1.png",
           width = 720, height = 480, scale = 3)

# polynomial interaction, larger effect size
ybp2 <- with(data, 0.25*x1*((x2-1)^2))
figbp2 <- create_plot(x1, x2, ybp2, "Cd", "As") |>
  layout(scene = list(camera = list(eye = list(x = 1.4, y = 1.4, z = 1.2))))
figbp2
save_image(figbp2, "index/figures/surfaces/bp2.png",
           width = 720, height = 480, scale = 3)

##########
# highly correlated (Ni and Co)
##########

# multiplicative interaction, smaller effect size
ycm1 <- with(data, 3/(1+exp(-4*x2)) + 0.3*x1*x2)
figcm1 <- create_plot(x1, x2, ycm1, "Ni", "Co") |>
  layout(scene = list(camera = list(eye = list(x = 1.2, y = 1.2, z = 1.5))))
figcm1
save_image(figcm1, "index/figures/surfaces/cm1.png",
           width = 720, height = 480, scale = 3)

# multiplicative interaction, larger effect size
ycm2 <- with(data, 3/(1+exp(-4*x2)) + 0.6*x1*x2)
figcm2 <- create_plot(x1, x2, ycm2, "Ni", "Co") |>
  layout(scene = list(camera = list(eye = list(x = 1.2, y = 1.2, z = 1.5))))
figcm2
save_image(figcm2, "index/figures/surfaces/cm2.png",
           width = 720, height = 480, scale = 3)

# polynomial interaction, smaller effect size
ycp1 <- with(data, 3/(1+exp(-4*x2)) + 0.1*x1*((x2-1)^2))
figcp1 <- create_plot(x1, x2, ycp1, "Ni", "Co") |>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = .1))))
figcp1
save_image(figcp1, "index/figures/surfaces/cp1.png",
           width = 720, height = 480, scale = 3)

# polynomial interaction, larger effect size
ycp2 <- with(data, 3/(1+exp(-4*x2)) + 0.2*x1*((x2-1)^2))
figcp2 <- create_plot(x1, x2, ycp2, "Ni", "Co") |>
  layout(scene = list(camera = list(eye = list(x = 1.5, y = 1.5, z = .1))))
figcp2
save_image(figcp2, "index/figures/surfaces/cp2.png",
           width = 720, height = 480, scale = 3)
```

Next, we look at simulated exposure and covariate data for the larger size dataset. We also create the grid of density plots for Spearman's correlation values between

exposures in the simulated smaller size datasets.

```
# create density plot of correlation for smaller size dataset
# read small size back in
out1 <- read_rds("sim/sim_preds_sm.RDS")

# extract correlation
cors <- out1 |>
  purrr::map_df(\(x) {
    cor_mat <- cor(x[, 5:14], method = "spearman")
    cor_mat[lower.tri(cor_mat)] <- NA
    melt_cor <- reshape2::melt(cor_mat) |>
      mutate(value = ifelse(value == 1, NA, value)) |>
      na.omit() |>
      mutate(sim = x$sim[1])
    return(melt_cor)
  })

# function for plotting x-axis
newbreaks <- function(lims) {
  range <- diff(lims)
  return(c(lims[1] + range/5, mean(lims), lims[2] - range/5))
}

# create grid of density plots
cors_dens <- cors |>
  group_by(Var1, Var2) |>
  mutate(mean = mean(value)) |>
  ungroup() |>
  ggplot(aes(x = value, fill = mean)) +
  geom_density() +
  scale_x_continuous(breaks = newbreaks,
                     labels = ~round(.x, 2)) +
  scale_y_continuous(position = "right") +
  scale_fill_gradient2(
    limit = c(-0.6, 0.6), breaks = c(-0.6, -0.3, 0, 0.3, 0.6),
    low = "deepskyblue3", mid = "white", high = "darkorange",
    na.value = NA) +
  ggh4x::facet_grid2(fct_rev(Var2) ~ Var1, scales = "free", independent = "x",
                     render_empty = FALSE, switch = "both") +
  labs(x = TeX(r"( Spearman's $\rho$ )")) +
  theme(strip.placement = "outside",
        strip.text.y.left = element_text(angle = 0),
        legend.justification = c(1, 0),
        legend.position = c(0.9, 0.1),
        legend.direction = "horizontal") +
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))
cors_dens

ggsave("index/figures/ch4_corr_sim.png", width = 10, height = 7)
```

```
##############
# look at simulated data, larger size
##############

# read larger size data back in
out2 <- read_rds("sim/sim_preds_lg.RDS")
comb_sim2 <- bind_rows(out2)
```

```r
# density plots for exposures
name_order <- c("As", "Cd", "Co", "Hg", "Ni", "Tl", "Pb", "Mo", "Sb", "Sn")
comb_sim2 |>
  mutate(sim = as.factor(sim)) |>
  select(5:15) |>
  pivot_longer(cols = 1:10) |>
  mutate(name = factor(name, levels = name_order)) |>
  ggplot(aes(x = value, group = sim)) +
  geom_line(stat = "density", color = "grey10", alpha = 0.01) +
  # reference observed densities
  geom_density(
    data = comb_log |> select(10:19) |> pivot_longer(cols = 1:10) |>
      mutate(name = factor(name, levels = name_order)),
    mapping = aes(x = value),
    color = "deepskyblue", linewidth = 0.75, inherit.aes = FALSE
  ) +
  facet_wrap(~name, scales = "free")
# save
ggsave("index/figures/ch4_univ_exp_sim_lg.png", width = 6, height = 4)
```

```r
# density plot for continuous covariates
cov_sim_p2 <- comb_sim2 |>
  mutate(sim = as.factor(sim)) |>
  select(age, bmi, sim) |>
  pivot_longer(cols = 1:2) |>
  ggplot(aes(x = value, group = sim)) +
  geom_line(stat = "density", color = "grey10", alpha = 0.01) +
  geom_density(
    data = comb_log |> select(age, bmi) |> pivot_longer(cols = 1:2),
    mapping = aes(x = value),
    color = "deepskyblue", linewidth = 0.75, inherit.aes = FALSE
  ) +
  facet_wrap(~name, scales = "free", ncol = 1)

# bar + violin plot for continuous covariates
cov_sim_q2 <- comb_sim2 |>
  mutate(sim = as.factor(sim)) |>
  select(sim, smoke, race) |>
  mutate(smoke = ifelse(smoke == 0, "Never-exposed", "Ever-exposed"),
         race = case_when(
           race == 1 ~ "Non-Hisp. white",
           race == 2 ~ "Non-Hisp. black",
           race == 3 ~ "Non-Hisp. other",
           race == 4 ~ "Hispanic born\nin US",
           race == 5 ~ "Hispanic born\noutside US"
         )) |>
  pivot_longer(cols = 2:3) |>
  group_by(sim, name, value) |>
  summarize(prop = n()/1000) |>
  mutate(value = factor(
    value, levels = rev(c("Never-exposed", "Ever-exposed",
                          "Non-Hisp. white", "Non-Hisp. black", "Non-Hisp. other",
                          "Hispanic born\nin US", "Hispanic born\noutside US"))
  )) |>
  ggplot(aes(x = value, y = prop)) +
  geom_bar(data = df_forcovcat, aes(x = value, y = after_stat(prop), group = 1),
           inherit.aes = FALSE, stat = "count", fill = "skyblue") +
  geom_violin(color = "gray30", fill = "gray", alpha = 0.25) +
  facet_wrap(~name, scales = "free", ncol = 1) +
  coord_flip() +
  labs(x = NULL, y = "proportion")

# plot in grid and save
```

```
cowplot::plot_grid(cov_sim_p2, cov_sim_q2, labels = "auto", nrow = 1,
                   rel_widths = c(0.4, 0.6))
ggsave("index/figures/ch4_univ_cov_sim_lg.png", width = 6, height = 4)
```

```
# extract spearman's correlation from large simulated data
corl <- out2 |>
  purrr::map_df(\(x) {
    cor_mat <- cor(x[, 5:14], method = "spearman")
    cor_mat[lower.tri(cor_mat)] <- NA
    melt_cor <- reshape2::melt(cor_mat) |>
      mutate(value = ifelse(value == 1, NA, value)) |>
      na.omit() |>
      mutate(sim = x$sim[1])
    return(melt_cor)
  })

# grid of density plots of correlation in large simulated data
cors_dens2 <- corl |>
  group_by(Var1, Var2) |>
  mutate(mean = mean(value)) |>
  ungroup() |>
  ggplot(aes(x = value, fill = mean)) +
  geom_density() +
  scale_x_continuous(breaks = newbreaks,
                     labels = ~round(.x, 2)) +
  scale_y_continuous(position = "right") +
  scale_fill_gradient2(
    limit = c(-0.6, 0.6), breaks = c(-0.6, -0.3, 0, 0.3, 0.6),
    low = "deepskyblue3", mid = "white", high = "darkorange",
    na.value = NA) +
  ggh4x::facet_grid2(fct_rev(Var2) ~ Var1, scales = "free", independent = "x",
                     render_empty = FALSE, switch = "both") +
  labs(x = TeX(r"( Spearman's $\rho$ )")) +
  theme(strip.placement = "outside",
        strip.text.y.left = element_text(angle = 0),
        legend.justification = c(1, 0),
        legend.position = c(0.9, 0.1),
        legend.direction = "horizontal") +
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))

# heatmap of average correlation in large simulated data
cor_sim2 <- corl |>
  group_by(Var1, Var2) |>
  summarize(value = mean(value)) |>
  mutate(label = round(value, 2)) |>
  ggplot(aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  geom_text(aes(label = label), size = 3.5) +
  scale_fill_gradient2(
    limit = c(-0.6, 0.6), breaks = c(-0.6, -0.3, 0, 0.3, 0.6),
    low = "deepskyblue3", mid = "white", high = "darkorange",
    na.value = NA) +
  coord_fixed() +
  labs(x = NULL, y = NULL, fill = TeX(r"( Mean Spearman's $\rho$ )")) +
  theme(
    panel.grid.major.x = element_line(color = "grey85",
                                      linewidth = 0.25,
                                      linetype = 2),
    panel.border = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.9, 0.1),
    legend.direction = "horizontal")+
```

```
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))

# put original and simulated correlation heatmaps together
top_row2 <- cowplot::plot_grid(cor_orig, cor_sim2, labels = "auto", label_size = 16,
                               nrow = 1, scale = 0.95)
# put original, simulated, and density plots of correlation together
cowplot::plot_grid(top_row2, cors_dens2, labels = c("", "c"), label_size = 16,
                   nrow = 2, rel_heights = c(5, 7), scale = c(1, 0.95))
ggsave("index/figures/ch4_corr_lg_simorigdens.png", width = 10, height = 12)
```

Here, we create the visualization of R^2 values in multiple linear regressions with the true functional form of the chemicals specified, in order to ensure we are achieving a realistic signal-to-noise ratio. This code was run on the Amherst HPC RStudio server.

```
### smaller size

# read in simulated datasets
out1_resp1 <- read_rds("sim/sim_resp_sm_a.RDS")
run_co_sm <- function() {
  # initialize vectors
  chems_oracle <- vector(mode = 'list', length = 1700)
  names(chems_oracle) <- names(out1_resp1)

  for(i in 1:1700) {
    df <- out1_resp1[[i]] |>
      mutate(race = as.factor(race), smoke = as.factor(smoke)) |>
      select(-sim)

    if(i <= 100) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) +
                                I(1/(1+exp(-4*Sn))), data = df)
    } else if (i <= 300) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Hg*Ni, data = df)
    } else if (i <= 500) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Hg*((Ni-1)^2)), data = df)
    } else if (i <= 700) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Cd*As, data = df)
    } else if (i <= 900) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Cd*((As-1)^2)), data = df)
    } else if (i <= 1100) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Ni*Co, data = df)
    } else if (i <= 1300) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
```

```
                                I(Ni*((Co-1)^2)), data = df)
    } else if (i <= 1500) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Hg:Ni:Tl, data = df)
    } else if (i <= 1700) {
      chems_oracle[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Hg*((Ni-1)^2)*Tl), data = df)
    }
  }
  return(chems_oracle)
}

# send to hpc
cosjob <- slurm_call(
  run_mlr_sm,
  global_objects = c('out1_resp1'),
  jobname = 'co_sm')

# get output
chem_oracle <- get_slurm_out(cosjob)
write_rds(chem_oracle, "sim/chem_oracle_sm.RDS")

### larger sample size

# read in simulated datasets
out2_resp1 <- read_rds("sim/sim_resp_lg_a.RDS")

run_co_lg <- function() {
  # initialize vectors
  chems_onlyl <- vector(mode = 'list', length = 1700)
  names(chems_onlyl) <- names(out2_resp1)
  chems_oraclel <- vector(mode = 'list', length = 1700)
  names(chems_oraclel) <- names(out2_resp1)

  for(i in 1:1700) {
    df <- out2_resp1[[i]] |>
      mutate(race = as.factor(race), smoke = as.factor(smoke)) |>
      select(-sim)

    if(i <= 100) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) +
                                I(1/(1+exp(-4*Sn))), data = df)
    } else if (i <= 300) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Hg*Ni, data = df)
    } else if (i <= 500) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Hg*((Ni-1)^2)), data = df)
    } else if (i <= 700) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Cd*As, data = df)
    } else if (i <= 900) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Cd*((As-1)^2)), data = df)
    } else if (i <= 1100) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
```

```
                                Ni*Co, data = df)
    } else if (i <= 1300) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Ni*((Co-1)^2)), data = df)
    } else if (i <= 1500) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                Hg:Ni:Tl, data = df)
    } else if (i <= 1700) {
      chems_oraclel[[i]] <- lm(y ~ Hg + Sb +
                                I(1/(1+exp(-4*Ni))) + I(Sb^2) + I(1/(1+exp(-4*Sn))) +
                                I(Hg*((Ni-1)^2)*Tl), data = df)
    }
  }
  return(chems_oraclel)
}

# send to hpc
coljob <- slurm_call(
  run_co_lg,
  global_objects = c('out2_resp1'),
  jobname = 'co_lg')

# get output
chem_oraclel <- get_slurm_out(coljob)
write_rds(chem_oraclel, "sim/chem_oracle_lg.RDS")
```

```
# extract rsq from smaller size datasets
chem_mods <- read_rds("sim/mlr/chem_oracle_sm.RDS")
rsq_chem <- chem_mods |>
  purrr::map_df(\(x) {
    data.frame(
      rsq = summary(x)$r.squared
    )
  }) |>
  mutate(name = names(chem_mods))

# plot for smaller size
rsqsmplot <- rsq_chem |>
  ggplot(aes(x = rsq)) +
  geom_density() +
  facet_wrap(~name,
             labeller = as_labeller(appender1,
                                    default = label_parsed),
             ncol = 4) +
  labs(y = "Density", x = latex2exp::TeX("R$^2$"))

# extract rsq from larger size datasets
chem_modl <- read_rds("sim/mlr/chem_oracle_lg.RDS")
rsq_cheml <- chem_modl |>
  purrr::map_df(\(x) {
    data.frame(
      rsq = summary(x)$r.squared
    )
  }) |>
  mutate(name = names(chem_modl))

# plot for larger size
rsqlgplot <- rsq_cheml |>
  ggplot(aes(x = rsq)) +
  geom_density() +
  facet_wrap(~name,
```

```
            labeller = as_labeller(appender1,
                                   default = label_parsed),
            ncol = 4) +
  labs(y = "Density", x = latex2exp::TeX("R$^2$"))

# plot both in grid and save
cowplot::plot_grid(rsqsmplot, rsqlgplot, labels = "auto", nrow = 1)
ggsave("index/figures/chem_rsq.png", width = 12, height = 6)
```

We also look at selecting degrees of freedom in BSR using 5,000 vs. 50,000 MCMC iterations. Refer to the code for fitting BSR on the smaller size datasets for how the models for this test were run.

```
# extract names of files for 50,000 iterations
main_folder <- "sim/bsr_df_sm"
subfolders <- list.dirs(main_folder, full.names = TRUE, recursive = TRUE)
mod_subfolders <- subfolders[grepl("mods", subfolders)]
mod_labels <- gsub("\\D", "", mod_subfolders)
mod_labels <- ifelse(mod_labels == "", 1, as.numeric(mod_labels))

mod_paths <- mod_subfolders |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = mod_labels)

# extract WAIC values from model output
waic <- names(mod_paths) |>
  purrr::map_df(\(x) {
    mod_paths[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- data.frame(
          df = c(1, 2, 3, 4, 5),
          waic = c(bsr[[1]]$waic,
                   bsr[[2]]$waic,
                   bsr[[3]]$waic,
                   bsr[[4]]$waic,
                   bsr[[5]]$waic),
          trial = rep(substr(y, nchar(y) - 4, nchar(y) - 4))
        )
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })

write_csv(waic, "sim/tables/test_waic.csv")

# look at waic values
waic <- read_csv("sim/tables/test_waic.csv")

# select degrees of freedom that minimizes waic
min_waic <- waic |>
  group_by(case, trial) |>
  filter(waic == min(waic)) |>
  arrange(as.numeric(case))
```

```r
# extract names of files for 5,000 iterations (trial 2)
main_folder2 <- "sim/bsr_df_sm2"
subfolders2 <- list.dirs(main_folder2, full.names = TRUE, recursive = TRUE)
mod_subfolders2 <- subfolders2[grepl("mods", subfolders2)]
mod_labels2 <- gsub("\\D", "", substr(mod_subfolders2, 16, nchar(mod_subfolders2)))
mod_labels2 <- ifelse(mod_labels2 == "", 1, as.numeric(mod_labels2))

mod_paths2 <- mod_subfolders2 |>
  purrr::map(\(x) {
    list.files(x, full.names = TRUE)
  }) |>
  setNames(nm = mod_labels2)

# extract waic from model outputs
waic2 <- names(mod_paths2) |>
  purrr::map_df(\(x) {
    mod_paths2[[x]] |>
      purrr::map_df(\(y) {
        bsr <- read_rds(y)
        result <- data.frame(
          df = c(1, 2, 3, 4, 5),
          waic = c(bsr[[1]]$waic,
                   bsr[[2]]$waic,
                   bsr[[3]]$waic,
                   bsr[[4]]$waic,
                   bsr[[5]]$waic),
          trial = rep(substr(y, nchar(y) - 4, nchar(y) - 4))
        )
        rm(bsr)
        return(result)
      }) |>
      mutate(case = x)
  })

write_csv(waic2, "sim/tables/test_waic2.csv")

# compare them
waic <- read_csv("sim/tables/test_waic.csv")
waic2 <- read_csv("sim/tables/test_waic2.csv")
waic_comb <- waic |>
  mutate(iter = 1) |>
  bind_rows(mutate(waic2, iter = 2))

# create plot
waic_comb |>
  filter(trial <= 5) |>
  mutate(df = as.factor(df),
         iter = factor(ifelse(iter == 1, "F", "P"), levels = c("P", "F"))) |>
  ggplot(aes(x = iter, y = waic, color = df)) +
  geom_point() +
  geom_line(aes(group = df)) +
  ggh4x::facet_grid2(paste0("Trial ", trial) ~ case,
                     scales = "free_y", independent = "y") +
  theme(axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        plot.caption = element_text(hjust = 0, size = 7)) +
  labs(y = "WAIC", x = "# MCMC iterations (F = 50,000, P = 5,000)",
       color = "Degrees\nfreedom",
       caption = paste0(
         "Scenarios are labelled in the top strip as follows:\n",
         "1 = base case; 2 = HgNi mult. small; 3 = HgNi mult. large; ",
         "4 = HgNi poly. small; 5 = HgNi poly. large; ",
         "6 = CdAs mult. small; 7 = CdAs mult. large;",
         "8 = CdAs poly. small; 9 = CdAs poly. large;\n",
```

```
         "10 = NiCo mult. small; 11 = NiCo mult. large; ",
         "12 = NiCo poly. small; 13 = NiCo poly. large; ",
         "14 = HgNiTl mult. small; 15 = HgNiTl mult. large; ",
         "16 = HgNiTl poly. small; 17 = HgNiTl poly. large"))

ggsave("index/figures/test_waic2.png", height = 6, width = 9)

# get proportion of correctly selected df's
waic_min <- waic_comb |>
  mutate(iter = ifelse(iter == 1, "full", "partial")) |>
  arrange(iter, case, trial) |>
  filter(trial <= 5) |>
  group_by(iter, trial, case) |>
  filter(waic == min(waic)) |>
  ungroup() |>
  pivot_wider(id_cols = c(trial, case),
              names_from = iter, values_from = df) |>
  mutate(equal = (full == partial))
mean(waic_min$equal)
```

We check convergence using trace plots for a selection of BKMR and BSR models.

```
library(bkmr)
library(NLinteraction)

# bkmr fits
bkmr_sm_am2_203 <- readRDS("testing/bkmr_sm_am2_203.RDS")
bkmr_lg_am2_284 <- readRDS("testing/bkmr_lg_am2_284.RDS")

# bsr fits
bsr_sm_am2_203df2 <- readRDS("testing/bsr_sm_am2_203df2.RDS")
bsr_lgf_am2_284df2 <- readRDS("testing/bsr_lgf_am2_284df2.RDS")

# bkmr sm
png("index/figures/traceplots/bksm_traceplot.png", width = 8, height = 6, units = "in", res = 360)
par(mfrow = c(2, 2))
TracePlot(fit = bkmr_sm_am2_203, par = "beta") # prior probability
TracePlot(fit = bkmr_sm_am2_203, par = "sigsq.eps") # variance of residuals
TracePlot(fit = bkmr_sm_am2_203, par = "r", comp = 1) # prob of each
TracePlot(fit = bkmr_sm_am2_203, par = "r", comp = 5)
dev.off()

# bkmr lg
png("index/figures/traceplots/bklg_traceplot.png", width = 8, height = 6, units = "in", res = 360)
par(mfrow = c(2, 2))
TracePlot(fit = bkmr_lg_am2_284, par = "beta")
TracePlot(fit = bkmr_lg_am2_284, par = "sigsq.eps")
TracePlot(fit = bkmr_lg_am2_284, par = "r", comp = 1)
TracePlot(fit = bkmr_lg_am2_284, par = "r", comp = 5)

# bsr sm
h <- bsr_sm_am2_203df2$posterior

png("index/figures/traceplots/bssm_traceplot.png", width = 8, height = 6, units = "in", res = 360)
par(mfrow = c(2, 2))
htau <- t(h[["tau"]][, , 1])
plot(htau[, 1], type = "l",
     main = paste0("(tau = ", round(mean(htau[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(htau[, 1]), col = "blue", lwd = 2)
```

```r
hsigma <- t(h[["sigma"]])
plot(hsigma[, 1], type = "l",
     main = paste0("(sigma = ", round(mean(hsigma[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(hsigma[, 1]), col = "blue", lwd = 2)

hzeta1 <- t(h[["zeta"]][, , 1, 2])
plot(hzeta1[, 1], type = "l",
     main = paste0("(zeta1 = ", round(mean(hzeta1[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(hzeta1[, 1]), col = "blue", lwd = 2)

hzeta5 <- t(h[["zeta"]][, , 5, 2])
plot(hzeta5[, 1], type = "l",
     main = paste0("(zeta5 = ", round(mean(hzeta5[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(hzeta5[, 1]), col = "blue", lwd = 2)
dev.off()

# bsr lg
h <- bsr_lgf_am2_284df2$posterior

png("index/figures/traceplots/bslg_traceplot.png", width = 8, height = 6, units = "in", res = 360)
par(mfrow = c(2, 2))
htau <- t(h[["tau"]][, , 1])
plot(htau[, 1], type = "l",
     main = paste0("(tau = ", round(mean(htau[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(htau[, 1]), col = "blue", lwd = 2)

hsigma <- t(h[["sigma"]])
plot(hsigma[, 1], type = "l",
     main = paste0("(sigma = ", round(mean(hsigma[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(hsigma[, 1]), col = "blue", lwd = 2)

hzeta1 <- t(h[["zeta"]][, , 1, 2])
plot(hzeta1[, 1], type = "l",
     main = paste0("(zeta1 = ", round(mean(hzeta1[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(hzeta1[, 1]), col = "blue", lwd = 2)

hzeta5 <- t(h[["zeta"]][, , 5, 2])
plot(hzeta5[, 1], type = "l",
     main = paste0("(zeta5 = ", round(mean(hzeta5[,1]), 2), ")"),
     xlab = "scan", ylab = "parameter value")
abline(h = mean(hzeta5[, 1]), col = "blue", lwd = 2)
dev.off()

# put all together

library(magick)

# list of input files
file_list <- c("index/figures/traceplots/bksm_traceplot.png",
               "index/figures/traceplots/bklg_traceplot.png",
               "index/figures/traceplots/bssm_traceplot.png",
               "index/figures/traceplots/bslg_traceplot.png")

# output file name
output_file <- "index/figures/traceplots/bksm_traceplotmerged.png"

# merge images
merge_and_save <- function(file_list, output_file) {
```

```
  images <- image_read(file_list)

  # two columns
  top_row <- image_append(images[1:2], stack = TRUE)
  bottom_row <- image_append(images[3:4], stack = TRUE)

  # combine columns
  combined_image <- image_append(c(top_row, bottom_row), stack = FALSE)

  # add labels
  combined_image <- image_annotate(combined_image, "a", location = "+30+20",
                                   size = 120, color = "black", font = "Roboto", weight = 700)
  combined_image <- image_annotate(combined_image, "b", location = "+2900+20",
                                   size = 120, color = "black", font = "Roboto", weight = 700)
  combined_image <- image_annotate(combined_image, "c", location = "+30+2110",
                                   size = 120, color = "black", font = "Roboto", weight = 700)
  combined_image <- image_annotate(combined_image, "d", location = "+2900+2110",
                                   size = 120, color = "black", font = "Roboto", weight = 700)

  image_write(combined_image, path = output_file, format = "png")
}

merge_and_save(file_list, output_file)
```

Corrections

A list of corrections after submission to department.

Corrections may be made to the body of the thesis, but every such correction will be acknowledged in a list under the heading "Corrections," along with the statement "When originally submitted, this honors thesis contained some errors which have been corrected in the current version. Here is a list of the errors that were corrected." This list will be given on a sheet or sheets to be appended to the thesis. Corrections to spelling, grammar, or typography may be acknowledged by a general statement such as "30 spellings were corrected in various places in the thesis, and the notation for definite integral was changed in approximately 10 places." However, any correction that affects the meaning of a sentence or paragraph should be described in careful detail. The files samplethesis.tex and samplethesis.pdf show what the "Corrections" section should look like. Questions about what should appear in the "Corrections" should be directed to the Chair.

References

Alaimo, S. (2010). *Bodily Natures: Science, Environment, and the Material Self*. Indiana University Press.

Antonelli, J. (2018). NLinteraction: Bayesian variable selection in multivariate semi-parametric regression models. Retrieved from https://github.com/jantonelli111/NLinteraction

Antonelli, J., Mazumdar, M., Bellinger, D., Christiani, D., Wright, R., & Coull, B. (2020). Estimating the health effects of environmental mixtures using Bayesian semiparametric regression and sparsity inducing priors. *The Annals of Applied Statistics*, *14*(1), 257–275. http://doi.org/10.1214/19-AOAS1307

Barbieri, M. M., & Berger, J. O. (2004). Optimal predictive model selection. *The Annals of Statistics*, *32*(3), 870–897. http://doi.org/10.1214/009053604000000238

Barrera-Gómez, J., Agier, L., Portengen, L., Chadeau-Hyam, M., Giorgis-Allemand, L., Siroux, V., ... Basagaña, X. (2017). A systematic comparison of statistical methods to detect interactions in exposome-health associations. *Environmental Health*, *16*(1), 74. http://doi.org/10.1186/s12940-017-0277-6

Bastain, T. M., Chavez, T., Habre, R., Girguis, M. S., Grubbs, B., Toledo-Corral, C., ... Breton, C. (2019). Study Design, Protocol and Profile of the Maternal And Developmental Risks from Environmental and Social Stressors (MADRES)

Pregnancy Cohort: A Prospective Cohort Study in Predominantly Low-Income Hispanic Women in Urban Los Angeles. *BMC Pregnancy and Childbirth*, *19*(1), 189. http://doi.org/10.1186/s12884-019-2330-7

Bellavia, A. (2021). *Statistical Methods for Environmental Mixtures*. Retrieved from https://bookdown.org/andreabellavia/mixtures/preface.html

Bensaude-Vincent, B., & Stengers, I. (1996). *A History of Chemistry*. Harvard University Press.

Bobb, J. F. (2017a, March). Introduction to Bayesian kernel machine regression and the bkmr R package. Retrieved from https://jenfb.github.io/bkmr/overview.html

Bobb, J. F. (2017b, December). Example using the bkmr R package with simulated data from the NIEHS mixtures workshop. Retrieved from https://jenfb.github.io/bkmr/SimData1.html#1_load_packages_and_download_data

Bobb, J. F. (2022). Bkmr: Bayesian Kernel Machine Regression. Retrieved from https://CRAN.R-project.org/package=bkmr

Bobb, J. F., Claus Henn, B., Valeri, L., & Coull, B. A. (2018). Statistical software for analyzing the health effects of multiple concurrent exposures via Bayesian kernel machine regression. *Environmental Health*, *17*(1), 67. http://doi.org/10.1186/s12940-018-0413-y

Bobb, J. F., Valeri, L., Claus Henn, B., Christiani, D. C., Wright, R. O., Mazumdar, M., ... Coull, B. A. (2015). Bayesian kernel machine regression for estimating the health effects of multi-pollutant mixtures. *Biostatistics*, *16*(3), 493–508. http://doi.org/10.1093/biostatistics/kxu058

Cribb, J. (2016). *Surviving the 21st century: Humanity's ten great challenges and how we can overcome them*. Springer.

Dominici, F., Peng, R. D., Barr, C. D., & Bell, M. L. (2010). Protecting Human Health From Air Pollution: Shifting From a Single-pollutant to a Multipollutant Approach. *Epidemiology*, *21*(2), 187. http://doi.org/10.1097/EDE.0b013e3181cc86e8

Dunn, O. J. (1961). Multiple Comparisons among Means. *Journal of the American Statistical Association*, *56*(293), 52–64. http://doi.org/10.1080/01621459.1961.10482090

Genest, C., & Nešlehovà, J. (2007). A Primer on Copulas for Count Data. *ASTIN Bulletin*, *37*(2), 475–515. http://doi.org/https://doi.org/10.2143/AST.37.2.2024077

Gibson, E. A., Nunez, Y., Abuawad, A., Zota, A. R., Renzetti, S., Devick, K. L., ... Kioumourtzoglou, M.-A. (2019). An overview of methods to address distinct research questions on environmental mixtures: An application to persistent organic pollutants and leukocyte telomere length. *Environmental Health*, *18*(1), 76. http://doi.org/10.1186/s12940-019-0515-1

Halford, G. S., Baker, R., McCredden, J. E., & Bain, J. D. (2005). How Many Variables Can Humans Process? *Psychological Science*, *16*(1), 70–76. http://doi.org/10.1111/j.0956-7976.2005.00782.x

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. New York, NY: Springer New York. http://doi.org/10.1007/978-0-387-84858-7

Hernández, A. F., Parrón, T., Tsatsakis, A. M., Requena, M., Alarcón, R., & López-Guarnido, O. (2013). Toxic effects of pesticide mixtures at a molecular level: Their relevance to human health. *Toxicology*, *307*, 136–145.

http://doi.org/10.1016/j.tox.2012.06.009

Heys, K., Shore, R., Pereira, M., Jones, K., & Martin, F. (2016). Risk assessment of environmental mixture effects. *RSC Advances*, *6*(53), 47844–47857. http://doi.org/10.1039/C6RA05406D

Hofert, M., Kojadinovic, I., Maechler, M., & Yan, J. (2023). Copula: Multivariate Dependence with Copulas. Retrieved from https://CRAN.R-project.org/package=copula

Hoskovec, L., Benka-Coker, W., Severson, R., Magzamen, S., & Wilson, A. (2021). Model choice for estimating the association between exposure to chemical mixtures and health outcomes: A simulation study. *PLOS ONE*, *16*(3), e0249236. http://doi.org/10.1371/journal.pone.0249236

Howe, C. G., Claus, H. B., Eckel, S. P., Farzan, S. F., Grubbs, B. H., Chavez, T. A., … Breton, C. V. (2020). Prenatal Metal Mixtures and Birth Weight for Gestational Age in a Predominately Lower-Income Hispanic Pregnancy Cohort in Los Angeles. *Environmental Health Perspectives*, *128*(11), 117001. http://doi.org/10.1289/EHP7201

Kannan, S., Misra, D. P., Dvonch, J. T., & Krishnakumar, A. (2006). Exposures to Airborne Particulate Matter and Adverse Perinatal Outcomes: A Biologically Plausible Mechanistic Framework for Exploring Potential Effect Modification by Nutrition. *Environmental Health Perspectives*, *114*(11), 1636–1642. http://doi.org/10.1289/ehp.9081

Kordas, K., Lönnerdal, B., & Stoltzfus, R. J. (2007). Interactions between nutrition and environmental exposures: Effects on health outcomes in women and children. *The Journal of Nutrition*, *137*(12), 2794–2797.

http://doi.org/10.1093/jn/137.12.2794

Krieger, N. (2001). Theories for social epidemiology in the 21st century: An ecosocial perspective. *International Journal of Epidemiology*, *30*(4), 668–677. http://doi.org/10.1093/ije/30.4.668

Krieger, N. (2011). *Epidemiology and the People's Health*. Oxford University Press. http://doi.org/10.1093/acprof:oso/9780195383874.001.0001

Lazarevic, N., Barnett, A. G., Sly, P. D., & Knibbs, L. D. (2019). Statistical Methodology in Studies of Prenatal Exposure to Mixtures of Endocrine-Disrupting Chemicals: A Review of Existing Approaches and New Alternatives. *Environmental Health Perspectives*, *127*(2), 026001. http://doi.org/10.1289/EHP2207

Lazarevic, N., Knibbs, L. D., Sly, P. D., & Barnett, A. G. (2020). Performance of variable and function selection methods for estimating the nonlinear health effects of correlated chemical mixtures: A simulation study. *Statistics in Medicine*, *39*(27), 3947–3967. http://doi.org/10.1002/sim.8701

Liu, D., Lin, X., & Ghosh, D. (2007). Semiparametric Regression of Multidimensional Genetic Pathway Data: Least-Squares Kernel Machines and Linear Mixed Models. *Biometrics*, *63*(4), 1079–1088. http://doi.org/10.1111/j.1541-0420.2007.00799.x

Lock, M. M., & Nguyen, V.-K. (2018). *An Anthropology of Biomedicine*. John Wiley & Sons.

Müller, H.-G. (1987). Weighted Local Regression and Kernel Methods for Nonparametric Curve Fitting. *Journal of the American Statistical Association*, *82*(397), 231–238. http://doi.org/10.1080/01621459.1987.10478425

Murphy, M. (2004). Uncertain Exposures and the Privilege of Imperception: Activist

Scientists and Race at the U.S. Environmental Protection Agency. *Osiris*, *19*(1), 266–282. http://doi.org/10.1086/649406

Murphy, M. (2017). Alterlife and Decolonial Chemical Relations. *Cultural Anthropology*, *32*(4), 494–503. http://doi.org/10.14506/ca32.4.02

Myers, N. (2015). *Rendering Life Molecular: Models, Modelers, and Excitable Matter*. Duke University Press.

Nadaraya, E. A. (1964). On Estimating Regression. *Theory of Probability & Its Applications*, *9*(1), 141–142. http://doi.org/10.1137/1109020

Naidu, R., Biswas, B., Willett, I. R., Cribb, J., Kumar Singh, B., Paul Nathanail, C., ... Aitken, R. J. (2021). Chemical pollution: A growing peril and potential catastrophic risk to humanity. *Environment International*, *156*, 106616. http://doi.org/10.1016/j.envint.2021.106616

National Academies of Sciences, Engineering, and Medicine, Division on Earth and Life Studies, Board on Environmental Studies and Toxicology, & Committee on Incorporating 21st Century Science into Risk-Based Evaluations. (2017). *Using 21st Century Science to Improve Risk-Related Evaluations*. Washington, D.C.: National Academies Press. http://doi.org/10.17226/24635

Nelsen, R. B. (2006). *An introduction to copulas* (2nd ed). New York: Springer.

Nguyen, V. (2020). Breathless in Beijing: Aerial Attunements and China's New Respiratory Publics. *Engaging Science, Technology, and Society*, *6*, 439–461. http://doi.org/10.17351/ests2020.437

Packer, M. (2022). Becoming with Toxicity: Chemical Epigenetics as "Racializing and Sexualizing Assemblage." *Hypatia*, *37*(1), 2–26. http://doi.org/10.1017/hyp.2021.68

Persson, L., Carney Almroth, B. M., Collins, C. D., Cornell, S., De Wit, C. A., Diamond, M. L., ... Hauschild, M. Z. (2022). Outside the Safe Operating Space of the Planetary Boundary for Novel Entities. *Environmental Science & Technology*, *56*(3), 1510–1521. http://doi.org/10.1021/acs.est.1c04158

Pesenti, N., Quatto, P., Colicino, E., Cancello, R., Scacchi, M., & Zambon, A. (2023). Comparative efficacy of three Bayesian variable selection methods in the context of weight loss in obese women. *Frontiers in Nutrition*, *10*, 1203925. http://doi.org/10.3389/fnut.2023.1203925

Plackett, R. L., & Hewlett, P. S. (1952). Quantal Responses to Mixtures of Poisons. *Journal of the Royal Statistical Society: Series B (Methodological)*, *14*(2), 141–154. http://doi.org/10.1111/j.2517-6161.1952.tb00108.x

R Core Team. (2013). *R: A language and environment for statistical computing: Reference index*. Vienna: R Foundation for Statistical Computing.

Roberts, E. F. S. (2017). What Gets Inside: Violent Entanglements and Toxic Boundaries in Mexico City. *Cultural Anthropology*, *32*(4), 592–619. http://doi.org/10.14506/ca32.4.07

Schulz, E., Speekenbrink, M., & Krause, A. (2018). A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, *85*, 1–16. http://doi.org/10.1016/j.jmp.2018.03.001

Shapiro, N. (2015). Attuning to the Chemosphere: Domestic Formaldehyde, Bodily Reasoning, and the Chemical Sublime. *Cultural Anthropology*, *30*(3), 368–393. http://doi.org/10.14506/ca30.3.02

Siemiatycki, J., & Thomas, D. C. (1981). Biological Models and Statistical Interactions: An Example from Multistage Carcinogenesis. *International Journal of*

Epidemiology, *10*(4), 383–387. http://doi.org/10.1093/ije/10.4.383

Sun, Z., Tao, Y., Li, S., Ferguson, K. K., Meeker, J. D., Park, S. K., … Mukherjee, B. (2013). Statistical strategies for constructing health risk models with multiple pollutants and their interactions: Possible choices and comparisons. *Environmental Health*, *12*(1), 85. http://doi.org/10.1186/1476-069X-12-85

Taylor, K. W., Joubert, B. R., Braun, J. M., Dilworth, C., Gennings, C., Hauser, R., … Carlin, D. J. (2016). Statistical Approaches for Assessing Health Effects of Environmental Chemical Mixtures in Epidemiology: Lessons from an Innovative Workshop. *Environmental Health Perspectives*, *124*(12), A227–A229. http://doi.org/10.1289/EHP547

Theriault, N., & Kang, S. (2021). Toxic Research: Political Ecologies and the Matter of Damage. *Environment and Society*, *12*(1), 5–24. http://doi.org/10.3167/ares.2021.120102

Tuck, E. (2009). Suspending Damage: A Letter to Communities. *Harvard Educational Review*, *79*(3), 409–428. http://doi.org/10.17763/haer.79.3.n0016675661t3n15

Valeri, L., Mazumdar, M. M., Bobb, J. F., Claus Henn, B., Rodrigues, E., Sharif, O. I. A., … Wright, R. O. (2017). The Joint Effect of Prenatal Exposure to Metal Mixtures on Neurodevelopmental Outcomes at 20–40 Months of Age: Evidence from Rural Bangladesh. *Environmental Health Perspectives*, *125*(6), 067015. http://doi.org/10.1289/EHP614

VanderWeele, T. J., & Knol, M. J. (2014). A Tutorial on Interaction. *Epidemiologic Methods*, *3*(1), 33–72. http://doi.org/10.1515/em-2013-0005

VanderWeele, T. J., & Mathur, M. B. (2019). Some desirable properties of the Bonferroni correction: Is the Bonferroni correction really so bad? *American Journal of Epidemiology*, *188*(3), 617–618. http://doi.org/10.1093/aje/kwy250

Venn, C. (2010). Individuation, Relationality, Affect: Rethinking the Human in Relation to the Living. *Body & Society*, *16*(1), 129–161. http://doi.org/10.1177/1357034X09354770

Vineis, P. (2018). From John Snow to omics: The long journey of environmental epidemiology. *European Journal of Epidemiology*, *33*(4), 355–363. http://doi.org/10.1007/s10654-018-0398-4

Wagaman, A. S., & Dobrow, R. P. (2021). *Probability: With Applications and R* (1st ed.). Wiley. http://doi.org/10.1002/9781119692430

Ward, J. B., Gartner, D. R., Keyes, K. M., Fliss, M. D., McClure, E. S., & Robinson, W. R. (2019). How do we assess a racial disparity in health? Distribution, interaction, and interpretation in epidemiological studies. *Annals of Epidemiology*, *29*, 1–7. http://doi.org/10.1016/j.annepidem.2018.09.007

Watson, G. S. (1964). Smooth Regression Analysis. *Sankhyā: The Indian Journal of Statistics*, *26*(4), 359–372.

Yu, L., Liu, W., Wang, X., Ye, Z., Tan, Q., Qiu, W., ... Chen, W. (2022). A review of practical statistical methods used in epidemiological studies to estimate the health effects of multi-pollutant mixture. *Environmental Pollution*, *306*, 119356. http://doi.org/10.1016/j.envpol.2022.119356