Introduction to Stan and Bayesian Inference

Paris Machine Learning Meetup

Dataiku User Meetup

21 September 2016

Eric Novik enovik@stan.fit

StanGroup

Outline

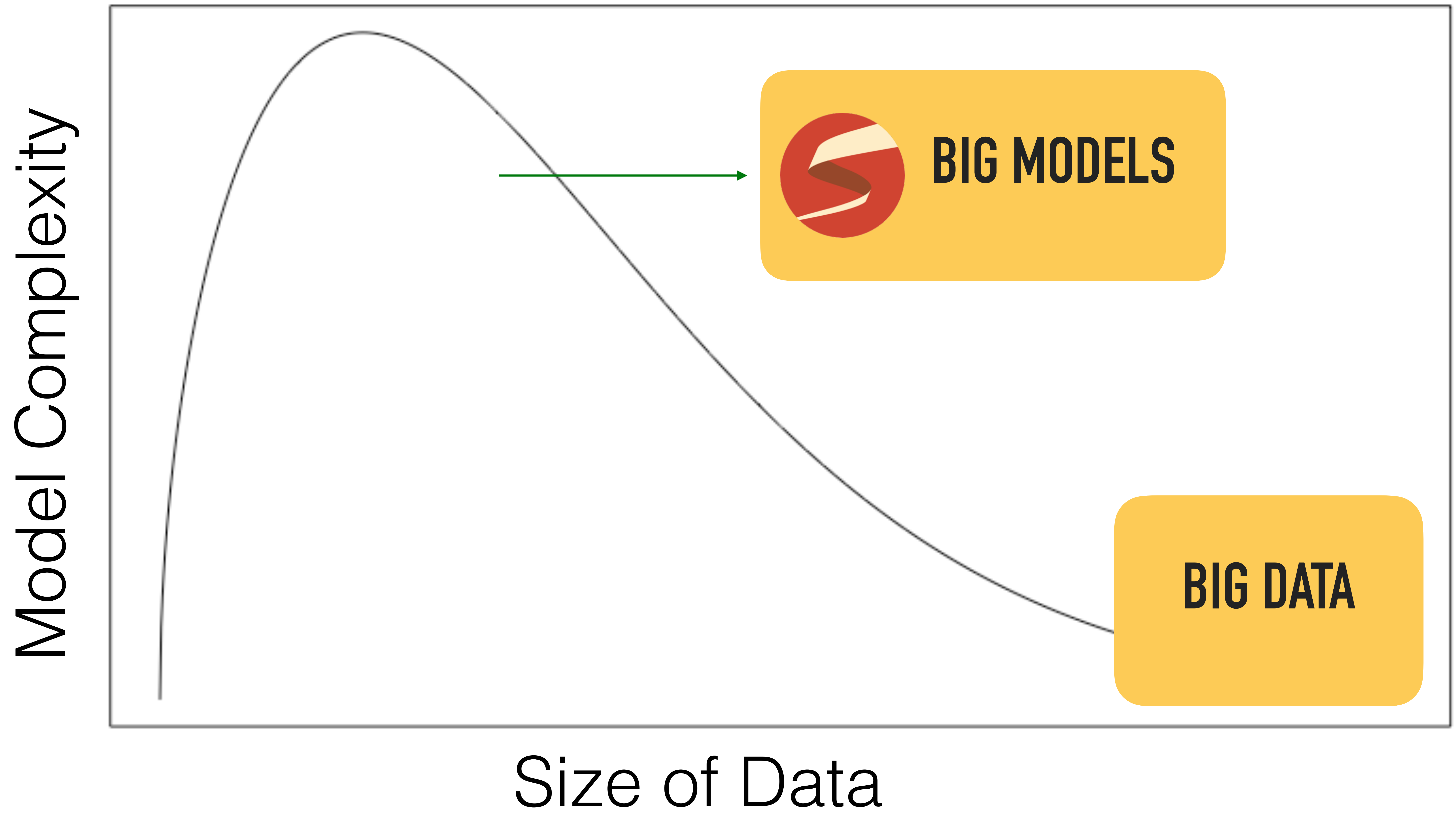
- Why should you bother with Bayes
- Why should you use Stan
- Introduction to modern Bayesian workflow
- Building up a Stan model
- Brief introduction to pooling and magic of multi-level models
- Pricing books using Stan and rstanarm package
- References and guide to getting started

Why Bayes

Benefits of Bayesian Approach

- Express your beliefs about parameters **and** the data generating process
- Properly account for uncertainty at the individual and group level
- Do not collapse grouping variables (e.g. sales for of multiple products over time) and do not fit a separate model to each group
- Small data is fine if you have a strong model
- But what about Big Data?

Big Data Need Big Models



Traditional Machine Learning and Causal Models

- **Problem A**: A large retailer wants to know how many units of each product they are going to sell tomorrow
- **Problem B**: A large retailer wants to find a revenue maximizing price for all of their products

- **Data**: We observe quantity sold of each product of time, meta data about the products, and price variation
- **Question**: Which one needs a causal model?

Why Stan

What Is Stan

What	What For
C++ Math/Stats Library	Mathematical specification of models; Automatic calculations of gradients
Imperative Model Specification Language	Fast and simple way to specify complex models
Algorithm Toolbox	Fit with full Bayes, approximate Bayes, optimization (HMC NUTS, ADVI, L-BFGS)
Interfaces (Command Line, R, Python, Julia, Matlab, Stata, …)	Work in the language of your choice
Interpretation Tools (shinystan)	Model critisism, algorithm evaluation

Who Is Using Stan

- 2,000+ members on the user list
- Over 10,000 manual downloads during the new release
- Stan is used for fitting climate models, clinical drug trials, genomics and cancer biology, population dynamics, psycholinguistics, social networks, finance and econometrics, professional sports, publishing, recommender systems, educational testing, and many more.

NOVARTIS

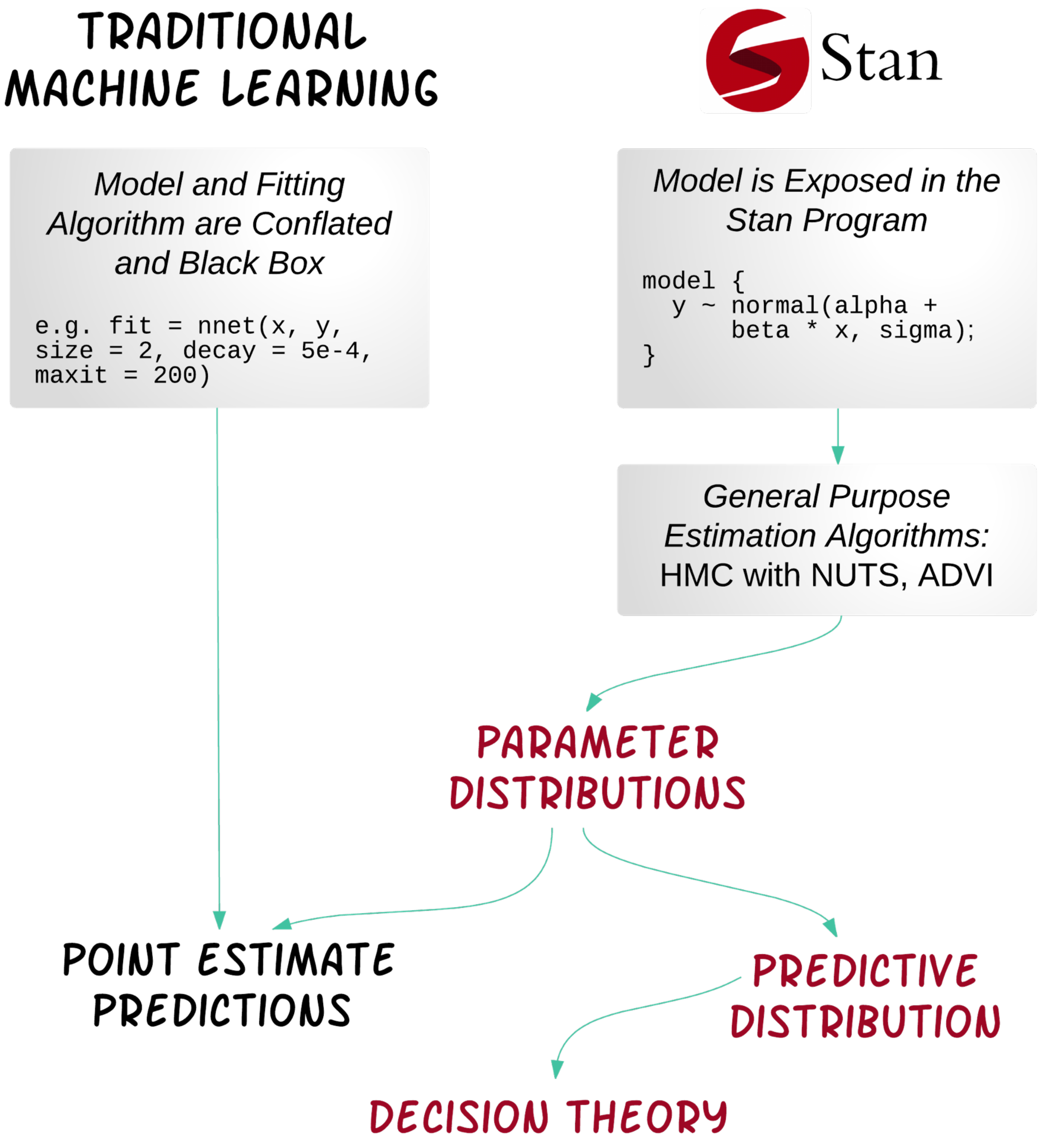
Penguin Random House

MAJOR LEAGUE BASEBALL

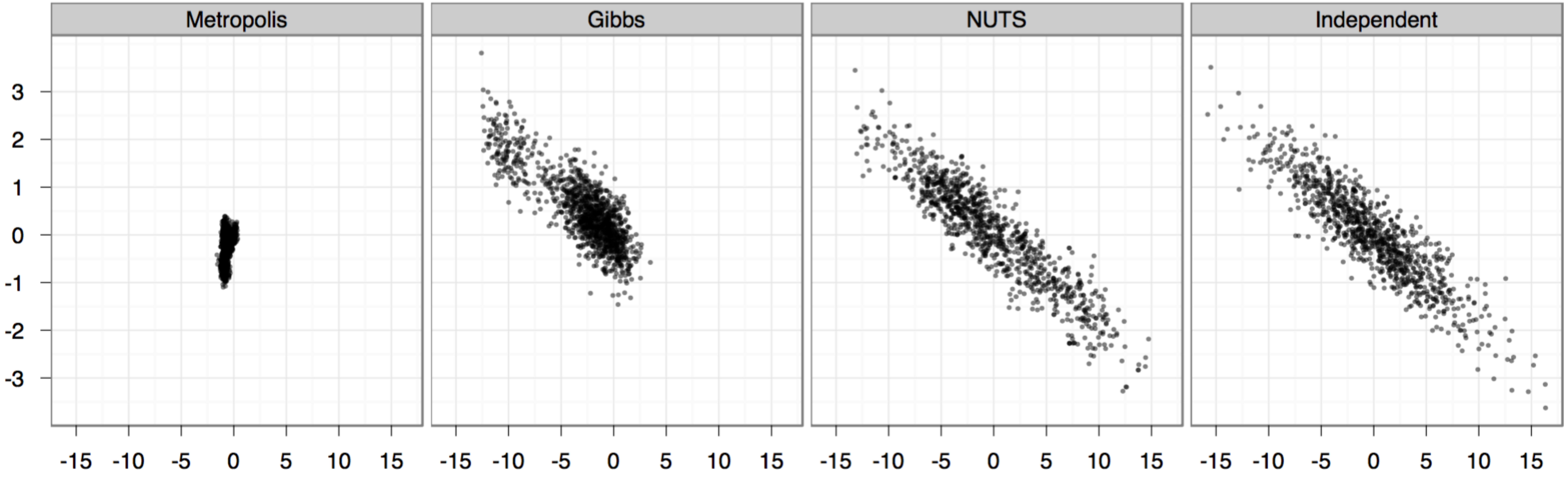
YouGov®
What the world thinks

Stan vs Traditional Machine Learning

- Model is directly expressed in Stan
- When in MCMC mode Stan produces produces draws from posterior distribution, not point estimates (MLE) of the parameters
- Fit complex models with millions of parameters
- Express and fit hierarchical models



Stan vs Gibbs and Metropolis



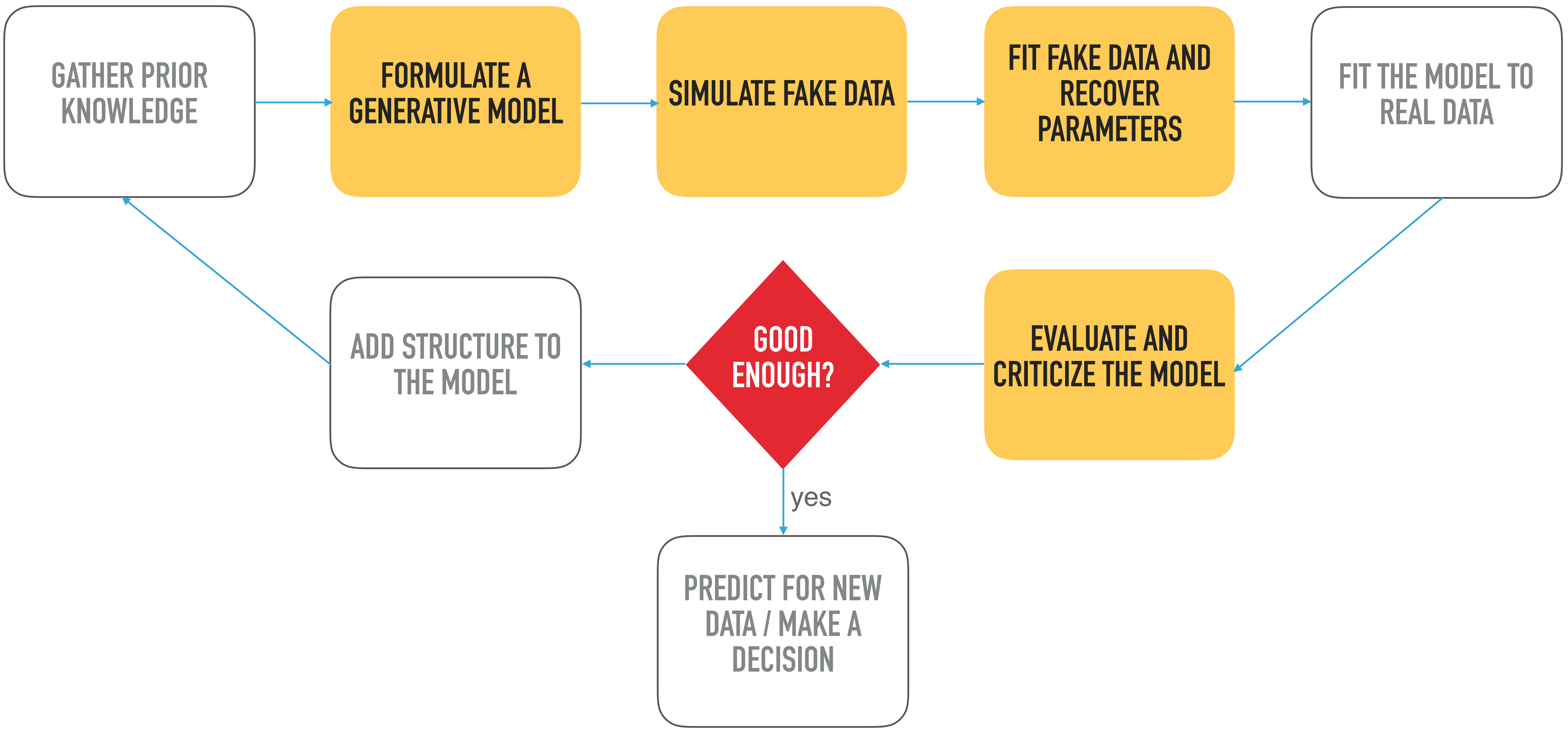
- 2-d projection of a highly correlated 250-d distribution
- 1M samples from Metropolis and 1M samples from Gibbs
- 1K samples from NUTS

Hamiltonian Simulation

Intro to Bayes

with Modern Bayesian Workflow

Bayesian Workflow



Bayesian Machinery

- The joint probability of data **y** and unknown parameter **theta**:

$$p(y,\theta) = p(y|\theta) * p(\theta)$$
$$p(y,\theta) = p(\theta|y) * p(y)$$

- The conditional probability of **theta** given **y**:

Likelihood Prior

$$p(\theta|y) = \frac{p(y|\theta) * p(\theta)}{p(y)} = \frac{p(y|\theta) * p(\theta)}{\int p(y,\theta)d\theta} = \frac{p(y|\theta) * p(\theta)}{\int p(y|\theta) * p(\theta)d\theta}$$

Marginal Likelihood

$$\propto p(y|\theta) * p(\theta)$$

Bernoulli Model

- If we model each occurrence as independent, the joint model can be written as:

Bernoulli Likelihood $p(y|\theta)$

$$p(y,\theta) = \prod_{n=1}^{N} \theta^{y_n} * (1-\theta)^{1-y_n} = \theta^{\sum_{n=1}^{N} y_n} * (1-\theta)^{\sum_{n=1}^{N}(1-y_n)}$$

- What happened to the prior, $p(\theta)$
- On the log scale:

$$\log(p(y,\theta)) = \sum_{n=1}^{N} y_n * \log(\theta) + \sum_{n=1}^{N}(1-y_n) * \log(1-\theta)$$

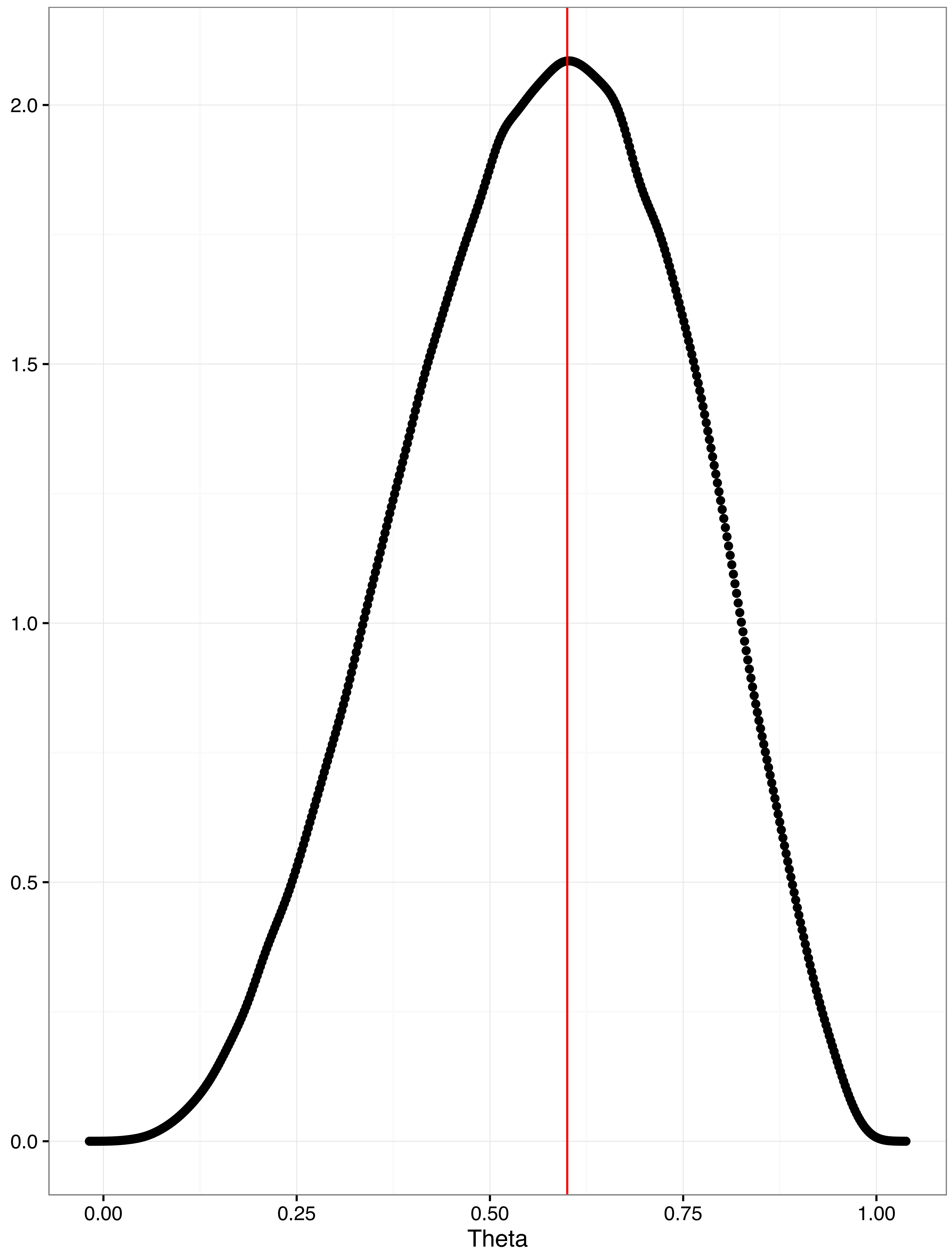
```
data <- list(N = 5,
             y = c(0, 1, 1, 0, 1))

# log probability function
lp <- function(theta, data) {
  lp <- 0
  for (i in 1:data$N) {
    lp <- lp + log(theta) * data$y[i] +
      log(1 - theta) * (1 - data$y[i])
  }
  return(lp)
}
```

Bernoulli Model

```
# using dbinom()
lp_dbinom <- function(theta, d) {
  lp <- 0
  for (i in 1:length(theta))
    lp[i] <- sum(dbinom(d$y, size = 1,
                        prob = theta[i],
                        log = TRUE))
  return(lp)
}

> lp(c(0.6, 0.7), data)
[1] -3.365058 -3.477970

> lp_dbinom(c(0.6, 0.7), data)
[1] -3.365058 -3.477970
```

Bernoulli Model

```
# generate the parameter grid
theta <- seq(0.001, 0.999,
             length.out = 250)

# log p(theta | y)
posterior <- lp(theta = theta, data)
posterior <- exp(log_prob)

# normalize
posterior <- posterior / sum(posterior)

# sample from the posterior
post_draws <- sample(theta, size = 1e5,
                     replace = TRUE,
                     prob = posterior)
post_dens <- density(post_draws)
mle <- sum(data$y) / data$N
> mle
[1] 0.6
```



Same Model in Stan

```
data {
  int<lower=1> N;
  int<lower=0, upper=1> y[N];
}
parameters {
  real theta;
}
model {
  for (n in 1:N)
    target += y[n] * log(theta) +
              (1 - y[n]) * log(1 - theta);
}
```

```
data {
  int<lower=1> N;
  int<lower=0, upper=1> y[N];
}
parameters {
  real<lower=0, upper=1> theta;
}
model {
  y ~ bernoulli(theta);
}
```

$$\log(p(y,\theta)) = \sum_{n=1}^{N} y_n * \log(\theta) + \sum_{n=1}^{N} (1 - y_n) * \log(1 - \theta)$$

Product Pricing

Basic Model and Data Simulation

Anlytical Problem

- A large publisher has hundreds of thousands of books in the catalog
- Every year, thousands of new books (products) are published
- There are also new authors, repeat authors, genres, seasonality, and so on
- Publisher wants to maximize revenue but if uncertainty is high, maximize quantity sold
- How should we model this? (and what is this)?

Basic Model for Quantity Sold

$$qty = f(price, price^2, product\_age, ...)$$

- For a Gaussian model, and one product:

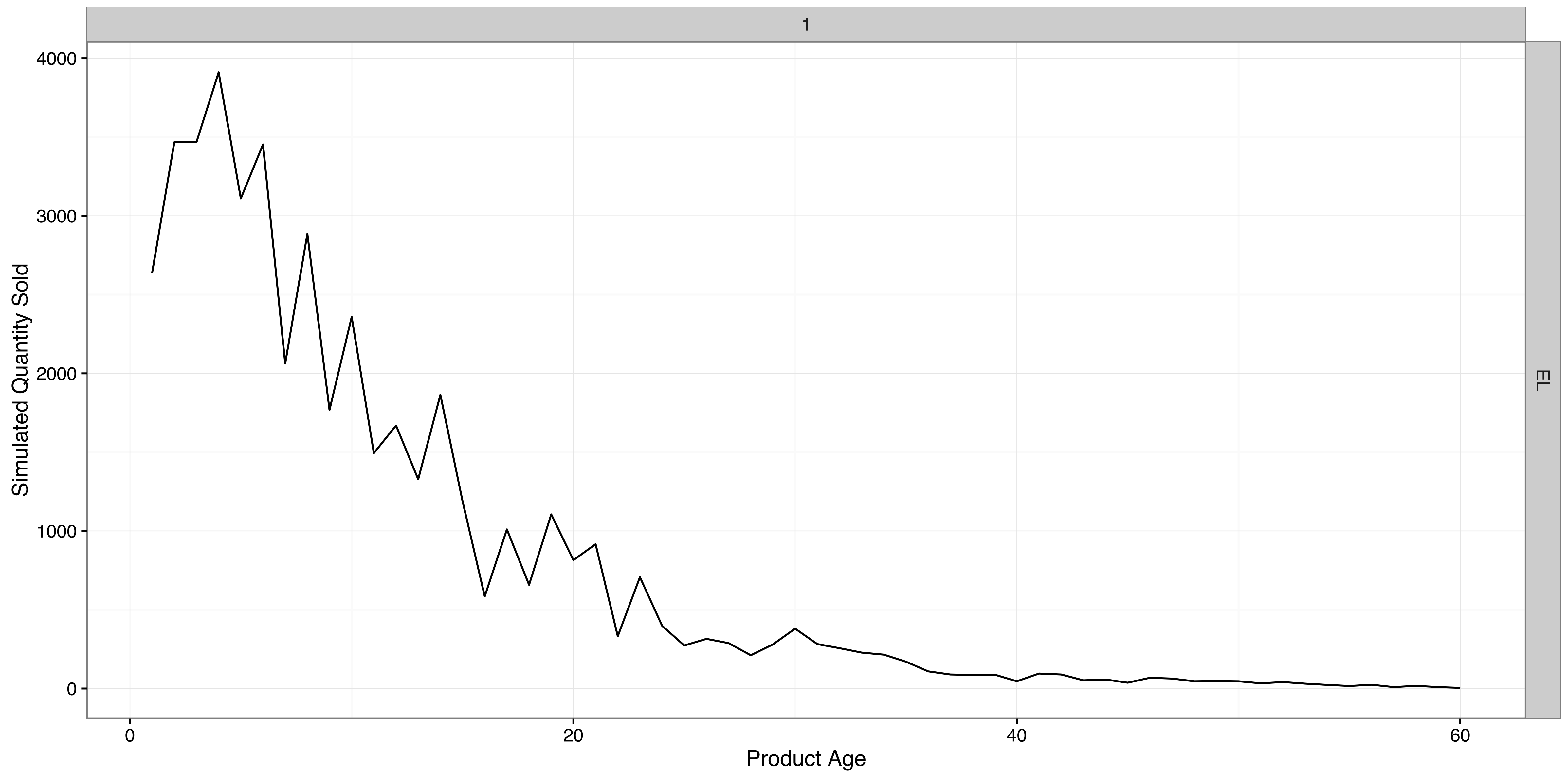
$$qty_i \sim N(X_i\beta, \sigma^2)$$

- For products that sell thousands of units we would fit a log-log model
- For lower volume products that sometimes sell zero units, we fit a count model that does not force the mean to be equal to the variance

$$qty \sim NegativeBinomial(\mu, \phi)$$

$$\mu = exp(\alpha + \beta_1 * product\_age + \beta_2 * price + ...)$$

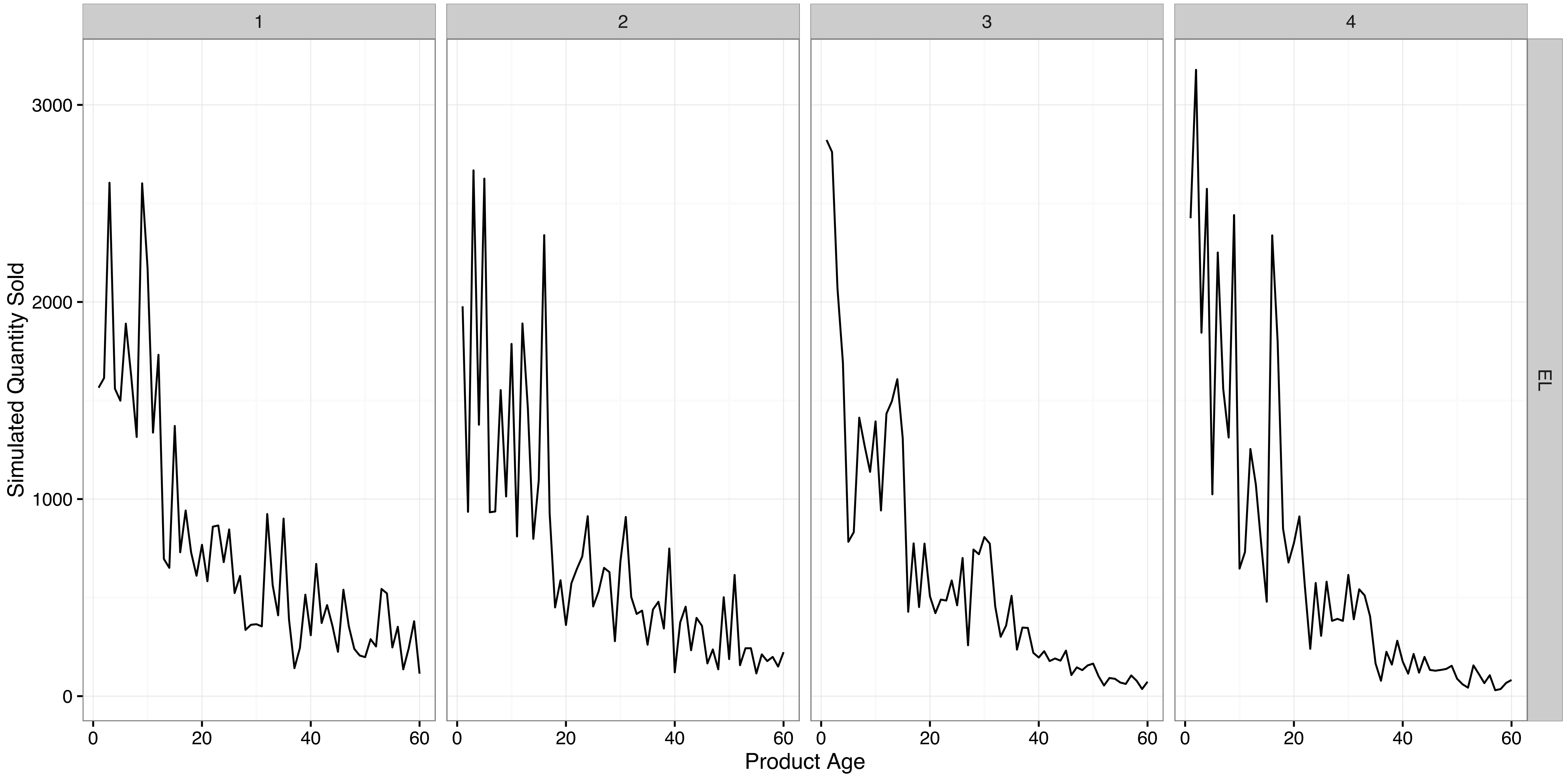
$$\sigma^2 = \mu + \mu^2/\phi$$



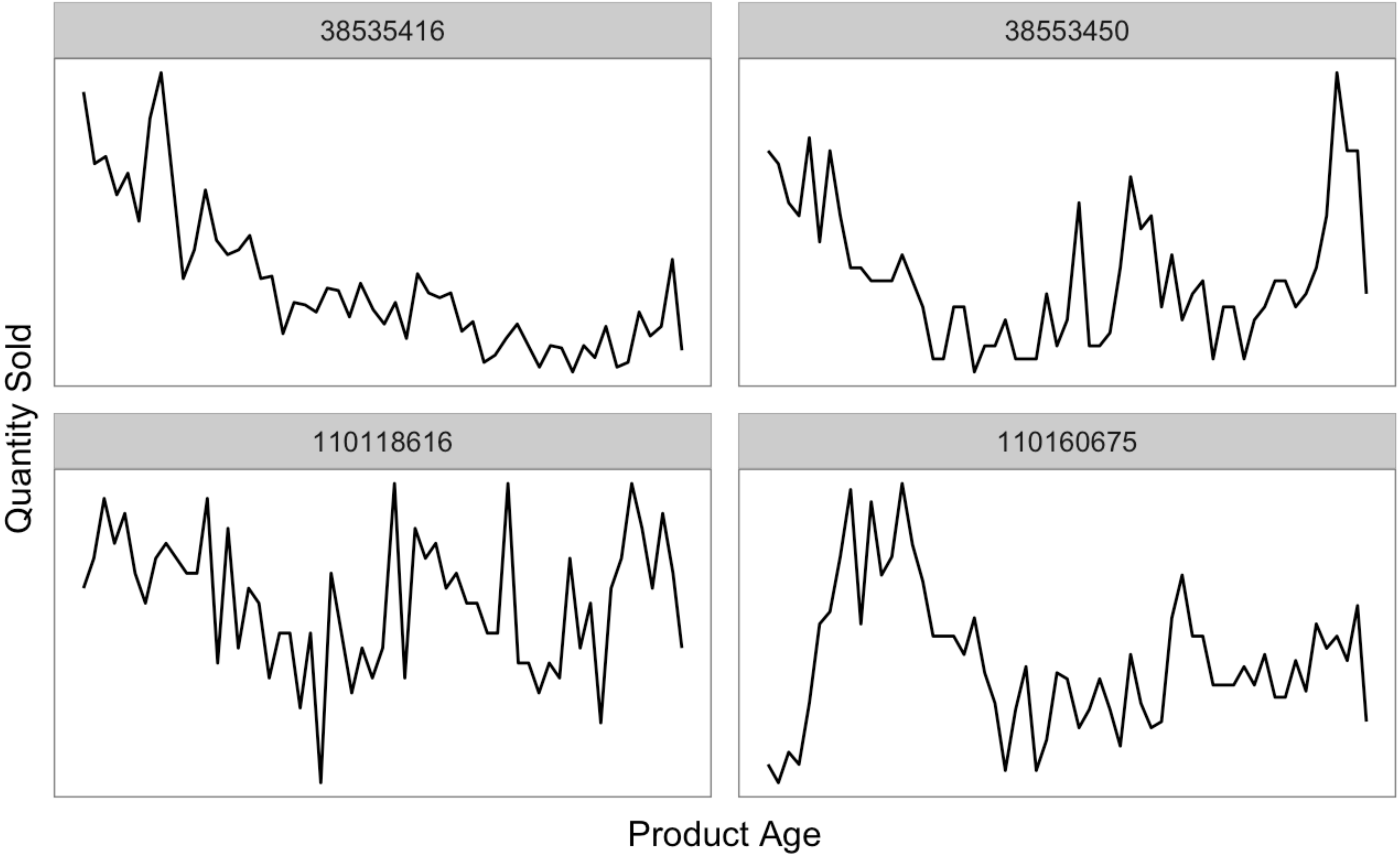

```
simd2 <- hir_data_sim(N_prod = 1,
                      global_intercept = 8.5,
                      theta = 10,
                      qty_process = "negbinom",
                      primary_price_process = "none",
                      …
                      linkinv = exp)
```

Simulating Data

```
if (process == "normal") {
  data <- data %>%
    mutate(qty = linkinv(product_intercept + product_beta_time * days + product_beta_price * price +
                   error_sd * rnorm(sum(n)))) %>%
    mutate(qty = ifelse(qty <= 0, 0, round(qty)))
} else { # negative binomial
  data <- data %>%
    mutate(mu = linkinv(product_intercept + product_beta_time * day + product_beta_price * price)
           qty = MASS::rnegbin(n = sum(n), mu = mu, theta = theta))
}
```



What About the Real Data?



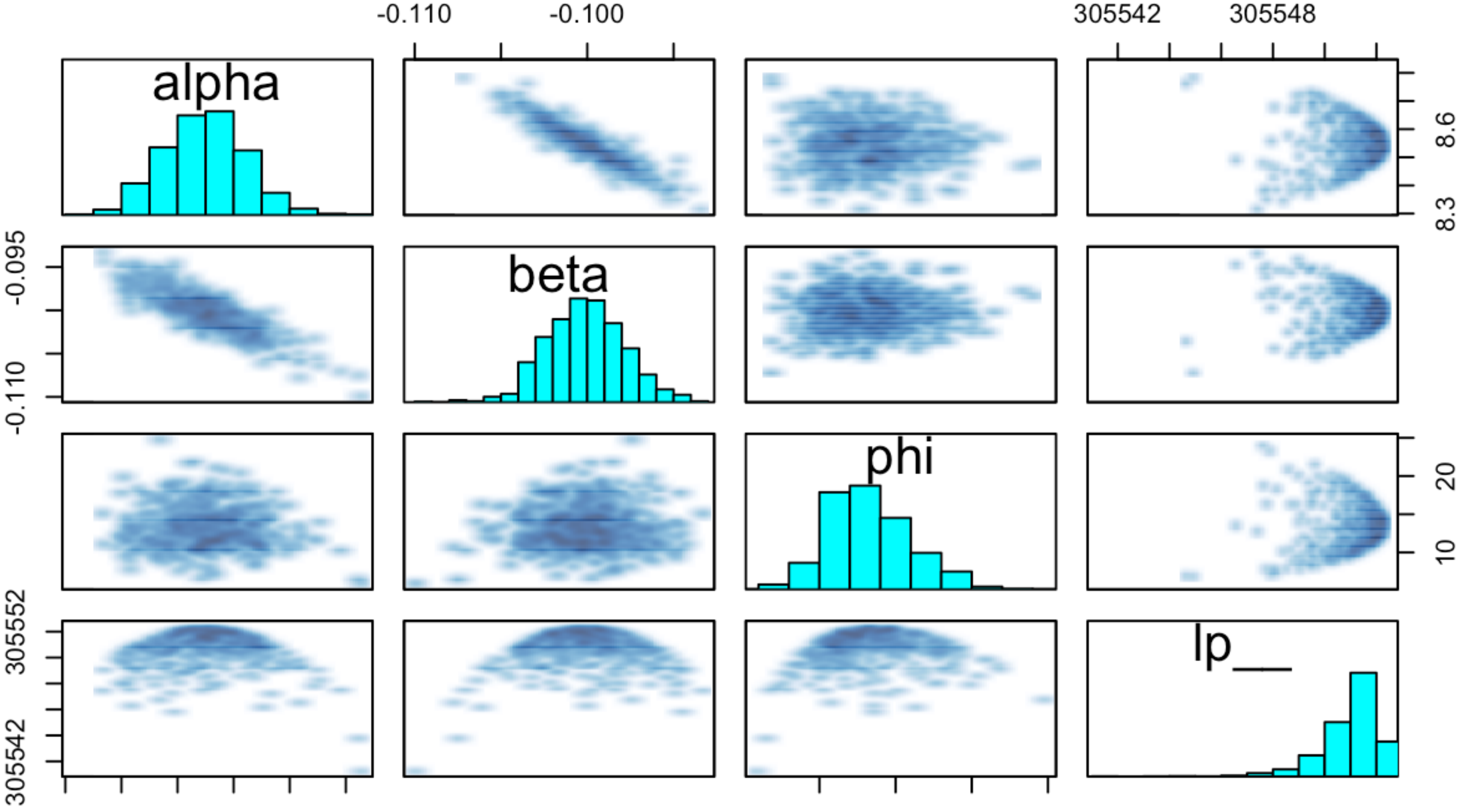
Baseline Stan Model for Single Product

```
data {
  int<lower=0> N;
  int<lower=0> y[N];
  vector[N] t;
}
parameters {
  real alpha;          // overall mean
  real beta;           // time beta
  real<lower=0> phi; // dispersion
}
model {
  vector[N] eta;
  // linear predictor
  eta = alpha + t * beta;
  // priors
  alpha ~ normal(0, 10);
  phi ~ cauchy(0, 2.5);
  beta ~ normal(0, 1);
  // likelihood
  y ~ neg_binomial_2_log(eta, phi);
}
```

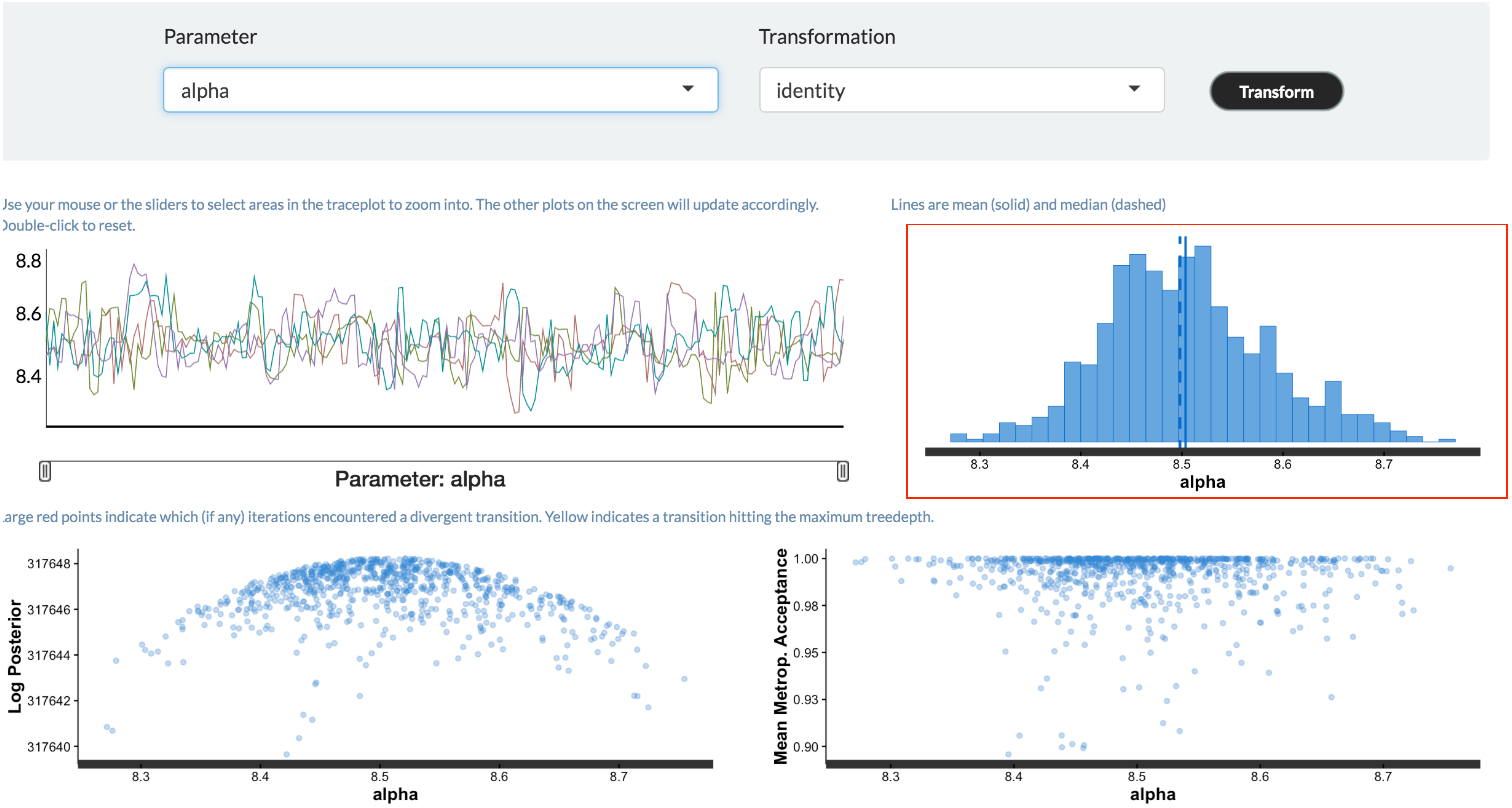
```
simd2_m2 <- stan('m2_self_stan_nbinom.stan'
                 data = list(N = nrow(simd2$data),
                             y = simd2$data$qty,
                             t = simd2$data$day),
                 control = list(stepsize = 0.01,
                                adapt_delta = 0.99),
                 cores = 4,
                 iter = 400)

# truth: alpha = 8.5, beta = -0.10, phi = 10
samples <- rstan::extract(simd2_m2,
                          pars = c('alpha',
                          'beta',
                          'phi'))

> lapply(samples, quantile)
$alpha
  0%  25%  50%  75% 100%
 8.3  8.4  8.5  8.6  8.8

$beta
    0%     25%     50%     75%    100%
-0.107 -0.102 -0.100 -0.099 -0.092

$phi
  0%  25%  50%  75% 100%
 6.2 10.1 11.5 13.0 24.1
```

Looking at Posterior Draws

```
> pairs(simd2_m2)
```



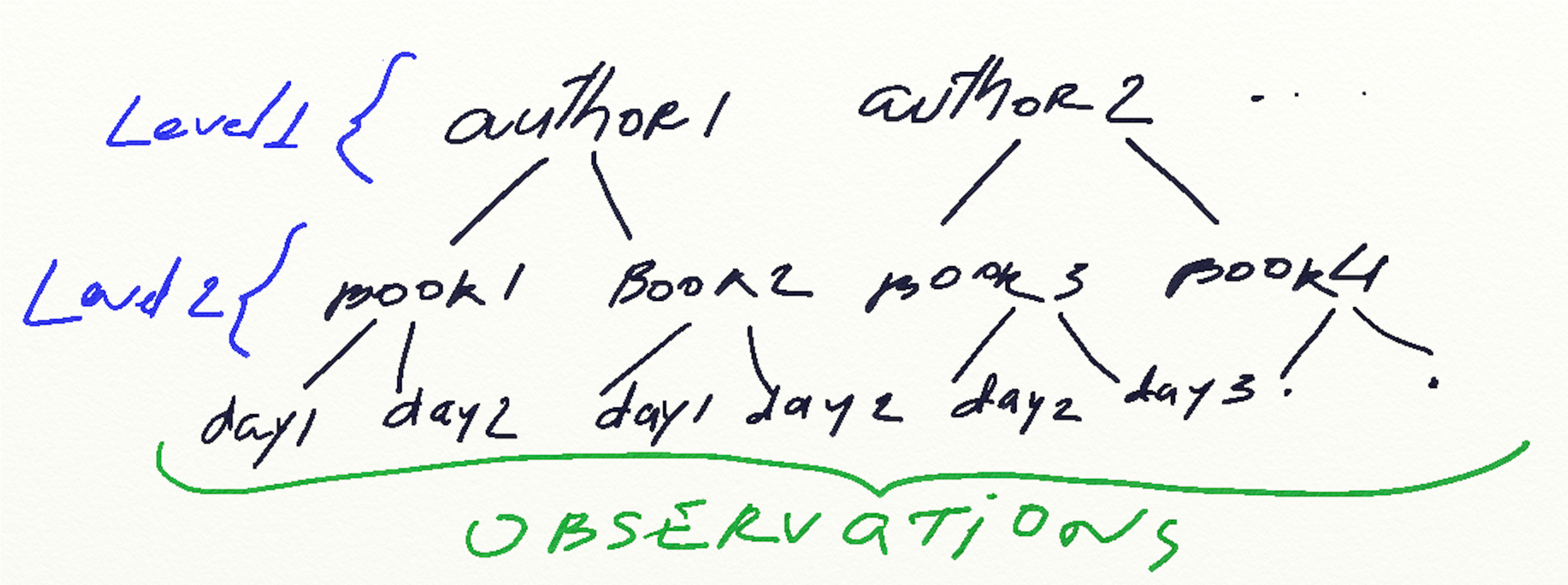
Diagnostics with Shinystan



Product Pricing

Introduction to Pooling and Hirarchical Models

We Have Multiple Products, Authors, Genres



Hierarchical Pooling in One Slide

Average sales across all books

Average sales for book j

Number of observations for book j

$$\widehat{\alpha}_j^{multilevel} \approx \frac{\frac{n_j}{\sigma_y^2}\bar{y}_j + \frac{1}{\sigma_\alpha^2}\bar{y}_{all}}{\frac{n_j}{\sigma_y^2} + \frac{1}{\sigma_\alpha^2}}$$

Indexes books

Estimate of average sales for book j

Within-book variance

Variance among average sales of different books

Multi-Level Models using Lmer Syntax

Formula	Alternative	Meaning
`(1 | g)`	`1 + (1 | g)`	Random intercept with fixed mean
`0 + offset(o) + (1 | g)`	`-1 + offset(o) + (1 | g)`	Random intercept with *a priori* means
`(1 | g1/g2)`	`(1 | g1)+(1 | g1:g2)`	Intercept varying among `g1` and `g2` within `g1`
`(1 | g1)+(1 | g2)`	`1 + (1 | g1) + (1 | g2)`	Intercept varying among `g1` and `g2`
`x + (x | g)`	`1 + x + (1 + x | g)`	Correlated random intercept and slope
`x + (x || g)`	`1 + x + (1 | g) + (0 + x | g)`	Uncorrelated random intercept and slope

Table 2: Examples of the right-hand sides of mixed-effects model formulas. The names of grouping factors are denoted `g`, `g1`, and `g2`, and covariates and *a priori* known offsets as `x` and `o`.

Fitting Multi-Level Models in rstanarm

“Fixed Effects”

```
fit <- stan_glmer.nb(qty ~ product_age + price + price_sqr +
             (1 + product_age + price + price_sqr | product),
                            algorithm = "sampling",
                            seed = 123,
                            cores = 4,
                            iter = 600,
                            data = data)
```

Random Seed

Varying Intercepts

1 Core per Chain

Varying Slopes

Number of Iterations per Chain

Fit using MCMC

Prediction and Checking: Posterior Predictive Distribution

- How can we tell if our model is sufficient for our task?
- We can simulate from the model and compare to observed data
- We can predict across interesting co-variates (e.g. change prices and observe how the model predicts qty over time)

Average Over Theta

Posterior Predictive Distribution

$$p(\tilde{y}|y) = \int_{\Theta} p(\tilde{y}|\theta)p(\theta|y)d\theta$$

New Data

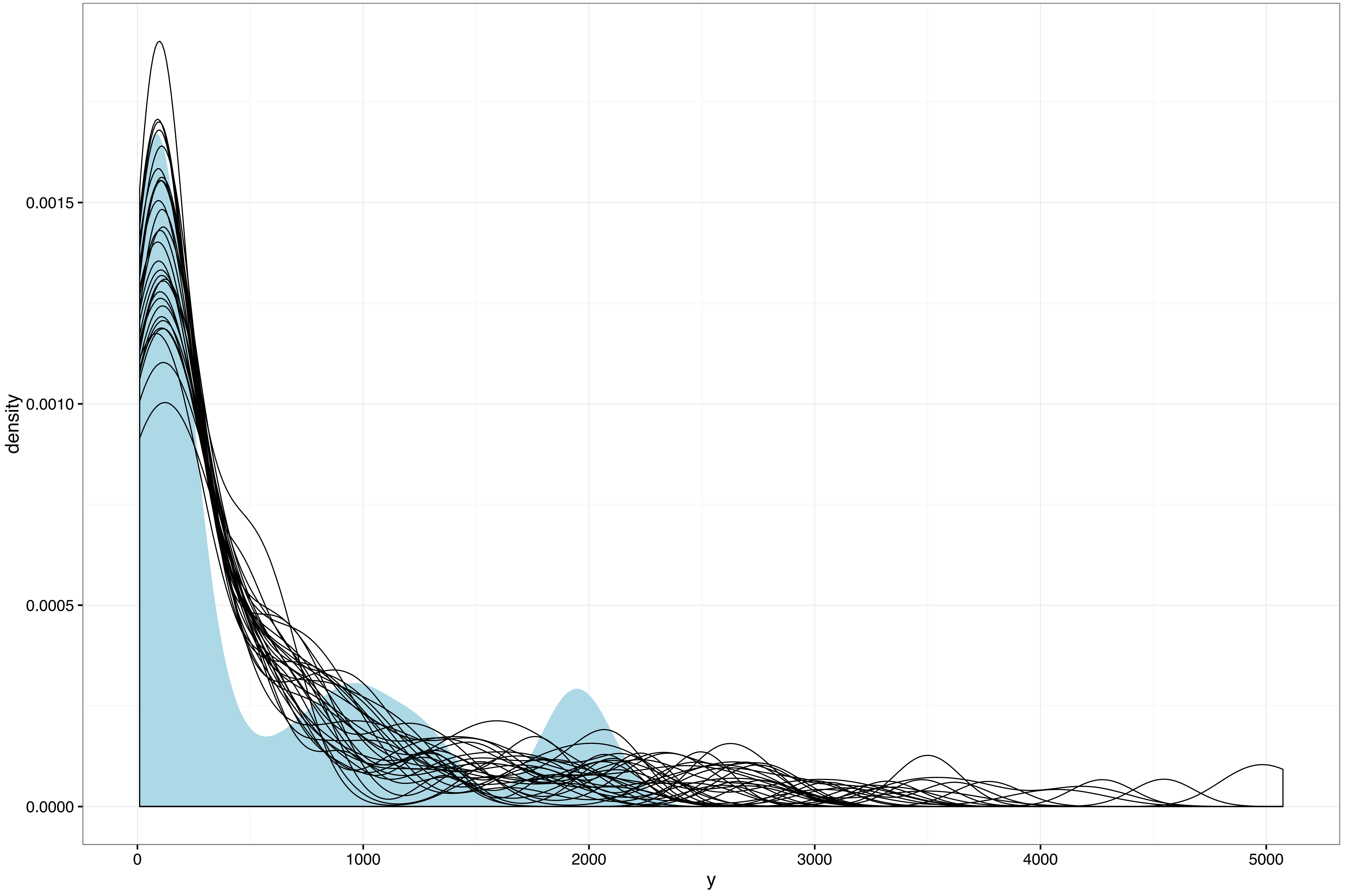
Data Used to Fit the Model

Likelihood Function

Weighted by the Posterior

Assessing Model Performance: Posterior Predictive Checks, Calibration

```
> pp_check(fit, check = “dist", overlay = TRUE)
```




```
> check_calib(d)
       in_90     in_50
1 0.9573893 0.7125305
> check_calib(d, TRUE)
Source: local data frame [203 x 3]

          id     in_90     in_50
       (dbl)     (dbl)     (dbl)
1   aaaaaaa1 0.9333333 0.7166667
2   aaaaaaa2 0.9500000 0.8333333
3   aaaaaaa3 0.9833333 0.8500000
4   aaaaaaa4 0.9666667 0.6500000
5   aaaaaaa5 0.9666667 0.7000000
6   aaaaaaa6 0.9833333 0.8833333
7   aaaaaaa7 0.9666667 0.6833333
8   aaaaaaa8 1.0000000 0.7666667
9   aaaaaaa9 0.8833333 0.6166667
10 aaaaaa10 0.9500000 0.8500000
..       ...       ...       ...
```

Prediction for Observed Prices

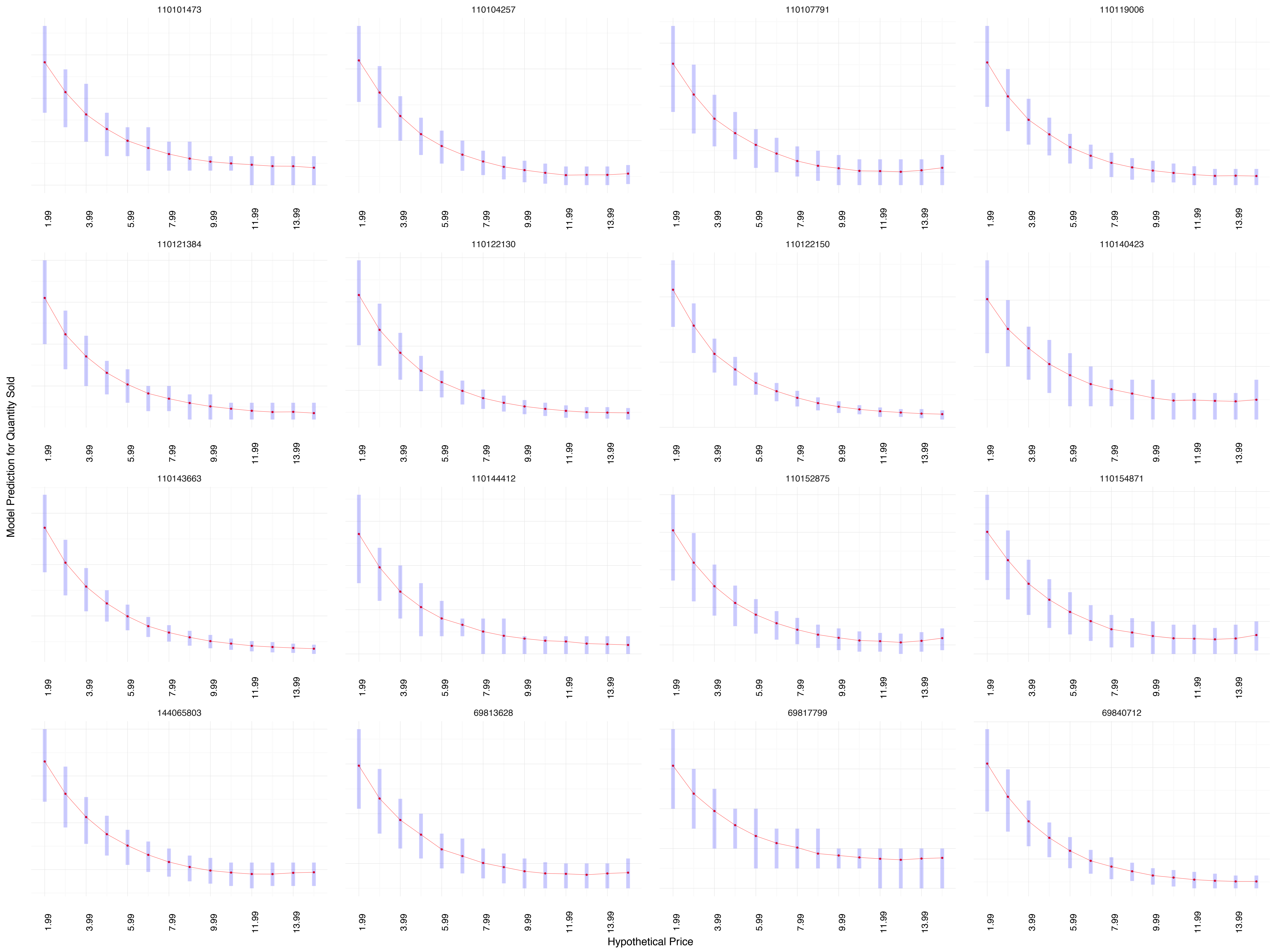
Revenue Optimisation

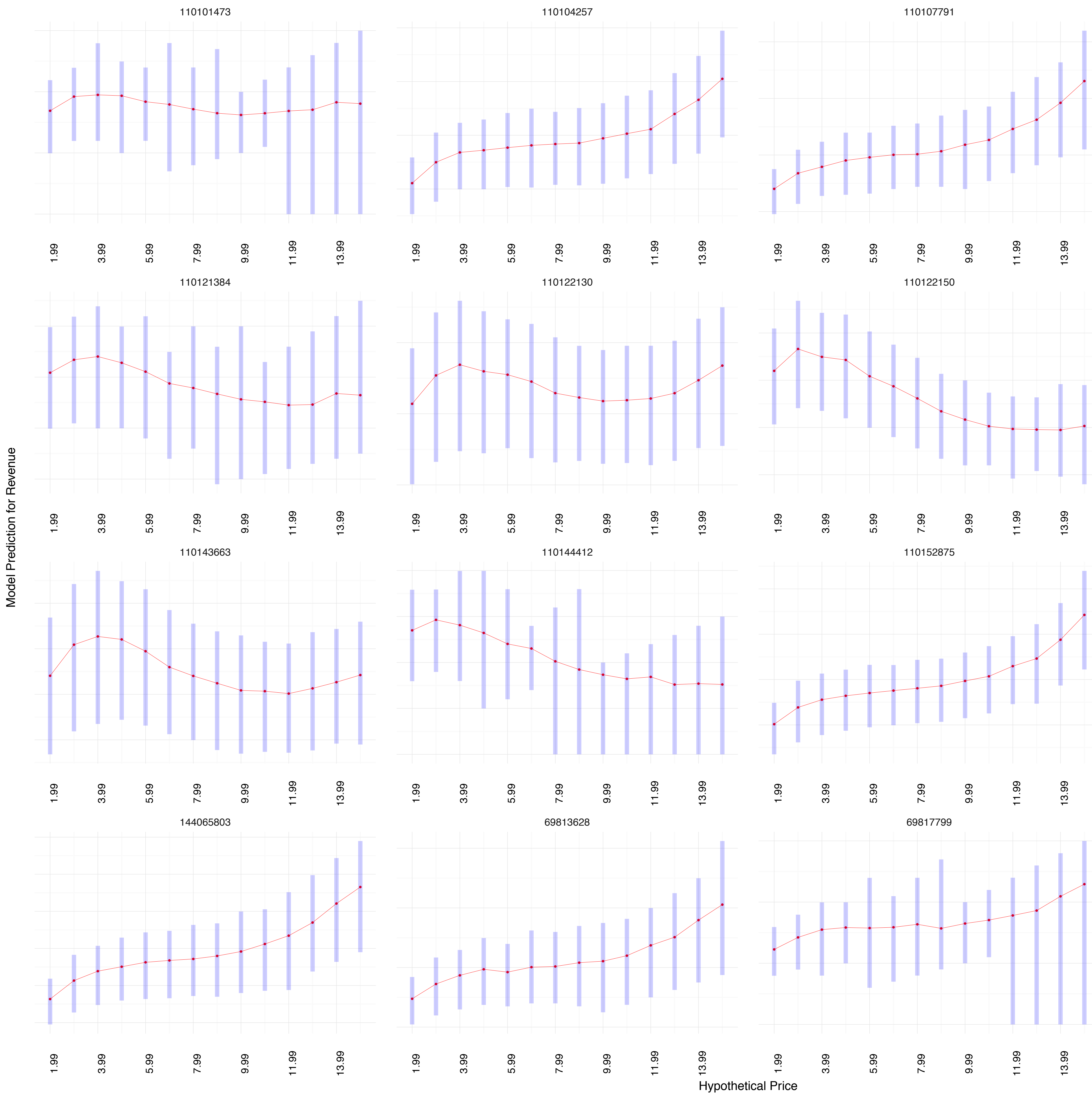
Generating Model Counterfactuals

Generating New Prices

```
new_data <- generate_new_prices(data, price_grid = seq(1.99, 14.99, by = 1))

> new_data
# A tibble: 1,946 x 7
   prod_key_factor prod_key price ysd_scaled price_scaled price_sqr_scaled month
            <fctr>    <chr> <dbl>      <dbl>        <dbl>            <dbl> <dbl>
1         aaaaaaaa aaaaaaaa  1.99   1.595587  -3.58149701       -2.5113317     8
2         aaaaaaaa aaaaaaaa  2.99   1.595587  -3.19446355       -2.4144849     8
3         aaaaaaaa aaaaaaaa  3.99   1.595587  -2.80743009       -2.2787437     8
4         aaaaaaaa aaaaaaaa  4.99   1.595587  -2.42039662       -2.1041082     8
5         aaaaaaaa aaaaaaaa  5.99   1.595587  -2.03336316       -1.8905783     8
6         aaaaaaaa aaaaaaaa  6.99   1.595587  -1.64632970       -1.6381541     8
7         aaaaaaaa aaaaaaaa  7.99   1.595587  -1.25929624       -1.3468357     8
8         aaaaaaaa aaaaaaaa  8.99   1.595587  -0.87226277       -1.0166228     8
9         aaaaaaaa aaaaaaaa  9.99   1.595587  -0.48522931       -0.6475157     8
10        aaaaaaaa aaaaaaaa 10.99   1.595587  -0.09819585       -0.2395142     8
# ... with 1,936 more rows

pred_q <- posterior_predict(fit, newdata = new_data)
```

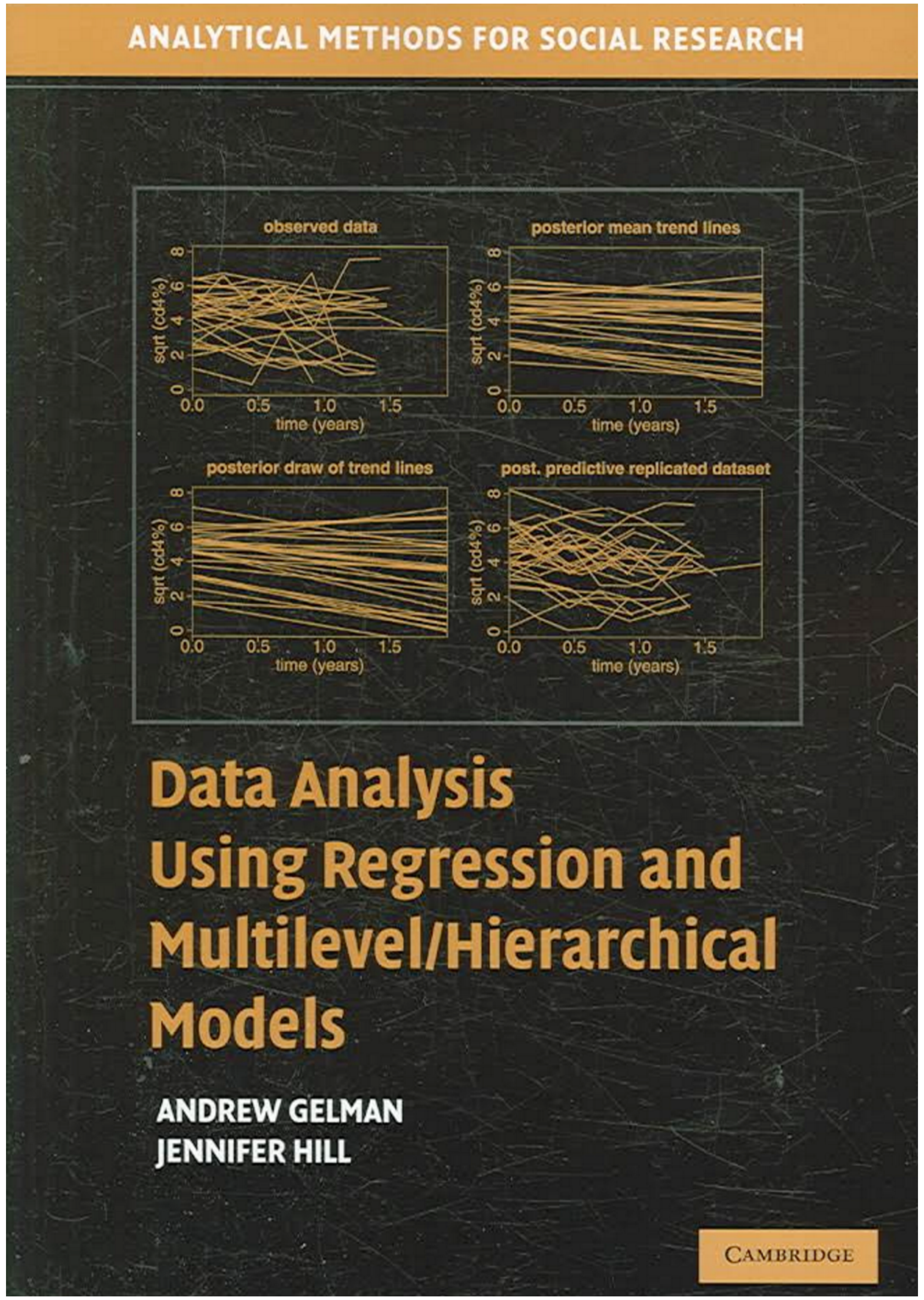
Computing Demand Curves and Revenue Predictions

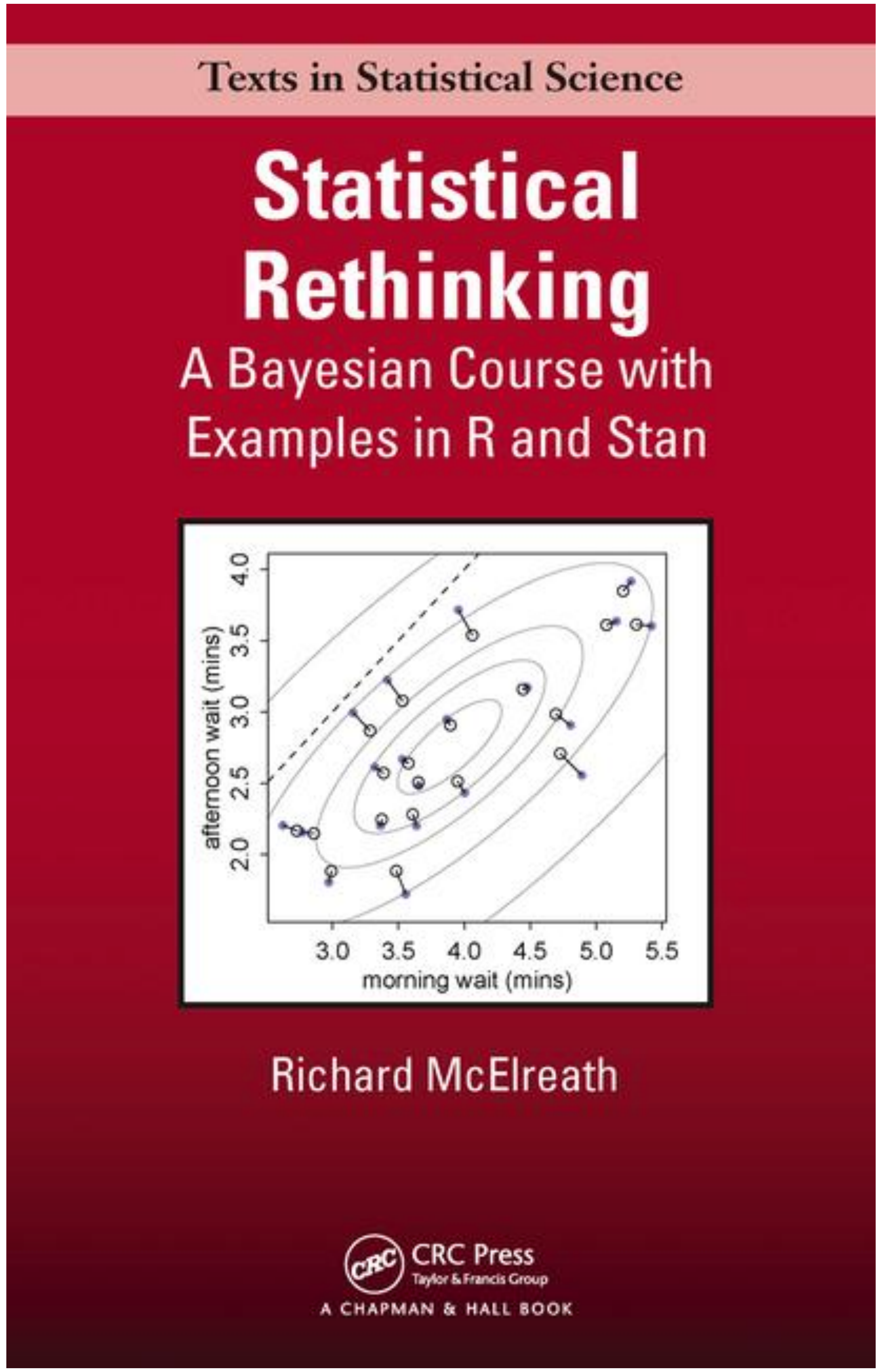


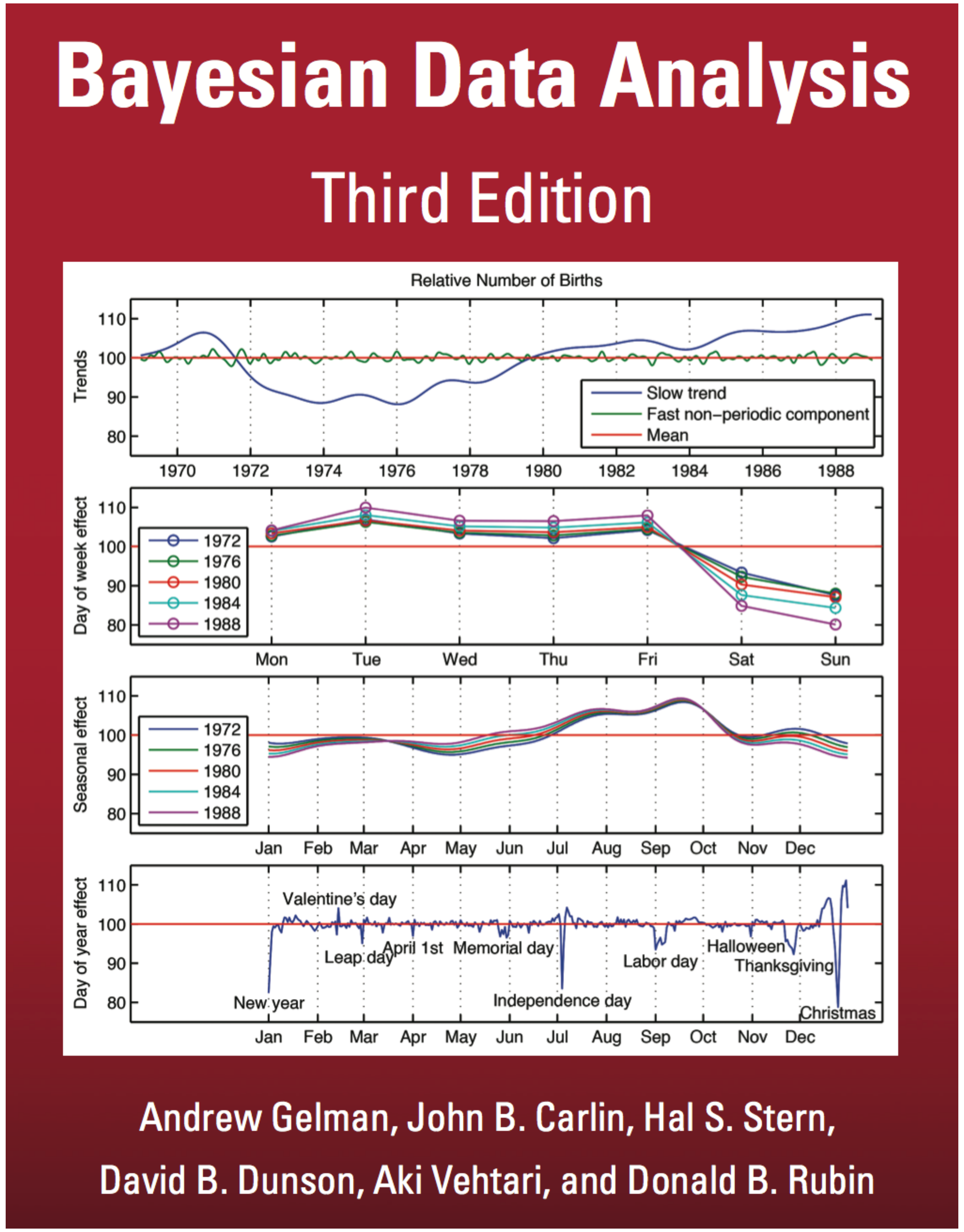


References

Books


ANALYTICAL METHODS FOR SOCIAL RESEARCH
Data Analysis Using Regression and Multilevel/Hierarchical Models
ANDREW GELMAN
JENNIFER HILL
CAMBRIDGE


Texts in Statistical Science
Statistical Rethinking
A Bayesian Course with Examples in R and Stan
Richard McElreath
CRC Press
A CHAPMAN & HALL BOOK


Bayesian Data Analysis
Third Edition
Relative Number of Births
Andrew Gelman, John B. Carlin, Hal S. Stern,
David B. Dunson, Aki Vehtari, and Donald B. Rubin

Some Papers and Videos

- Stan: A probabilistic programming language for Bayesian inference and optimization (Andrew Gelman, et. al.) http://www.stat.columbia.edu/~gelman/research/published/stan_jebs_2.pdf
- Stan: A Probabilistic Programming Language (Bob Carpenter, et. al.) http://www.stat.columbia.edu/~gelman/research/published/stan-paper-revision-feb2015.pdf
- Hamiltonian Monte Carlo (Michael Betancourt) https://www.youtube.com/watch?v=pHsuIaPbNbY
- Stan Hands-on with Bob Carpenter https://www.youtube.com/watch?v=6NXRCtWQNMg
- A lot more available on mc-stan.org

Merci Beaucoup!

- eric@stan.fit
- @ericnovik
- mc-stan.org / stan.fit
- www.linkedin.com/in/enovik

StanGroup