DATA 582: Bayesian Inference

Lecture 6: Bayesian Linear Regression using `rstanarm`

Dr. Irene Vrbik

UBCO MDS

Introduction

- Last lecture we were introduced MCMC algorithms.
- These methods are often necessary to fit of realistic, complex, high-dimensional Bayesian models.
- A number of packages and libraries exist across the popular coding languages that implement commonly used models in Bayesian Inference (see a list here for packages in R)
- Today we'll look at the **rstanarm** package for fitting the Linear Regression model in the Bayesian framework.

Normal Regression Model

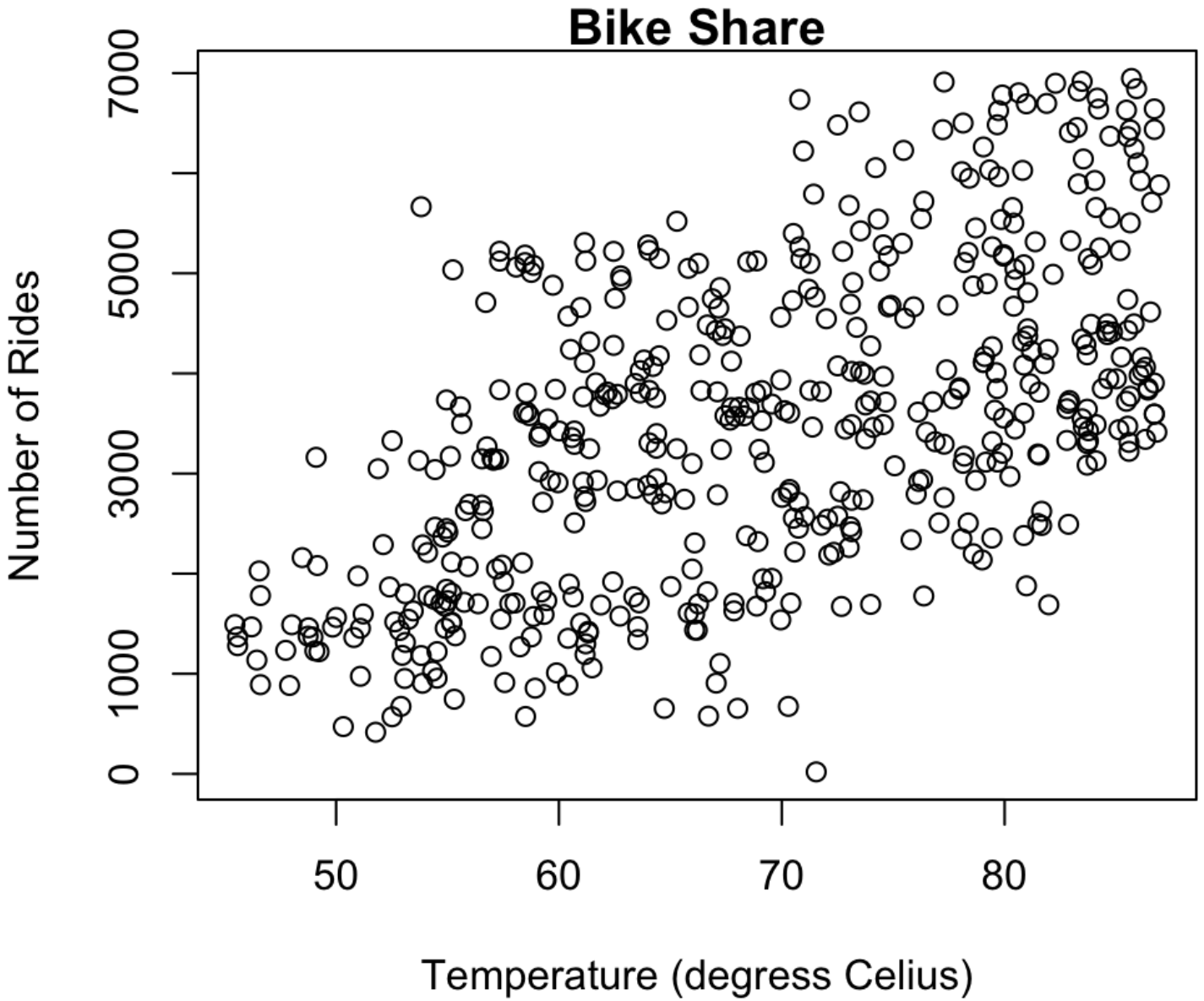
- Recall that regression analysis seeks to explain or predict a *response variable* (outcome), Y based on one or more *predictor variables*[1] X
- We begin by introducing the *Normal regression model* for a quantitative response variable Y.
- Assuming we are taking measurements from subjects $i = 1, 2, \ldots, n$, we let y_i and x_i denote the observed response and predictor values for the ith subject.

[1] aka explanatory/independent variables or covariates

Bike Share

- Let's consider the `bikes` data in the **bayesrules** package[2]
- For each of 500 days in the study, `bikes` contains the number of rides taken (`rides`) and a measure of what the temperature felt like when incorporating factors such as humidity (`temp_feel`).
- A scatterplot of `rides` by `temp_feel` supports our assumption of a positive linear relationship between the two, ...

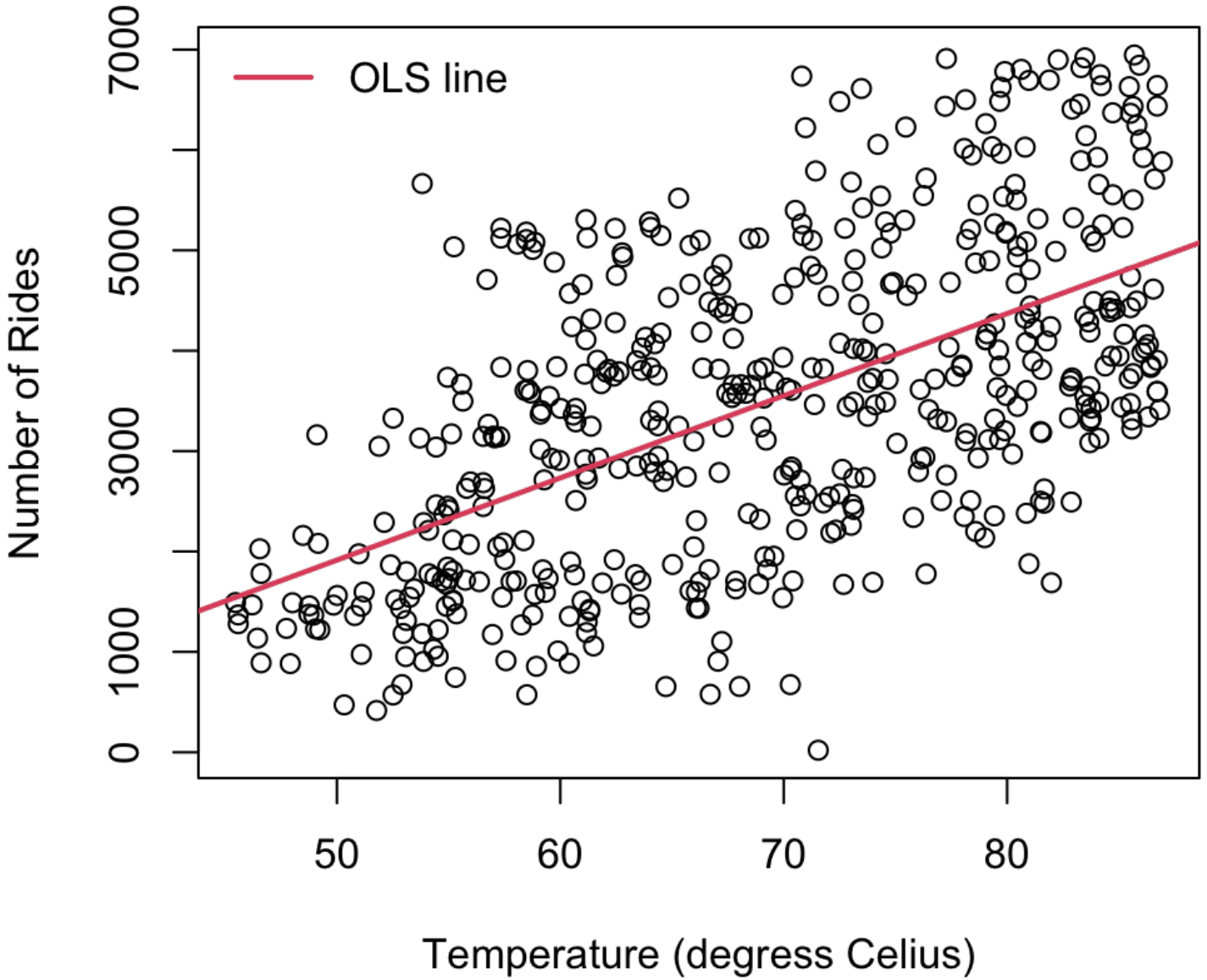
[2] this is a subset of the Bike Sharing dataset made available on the UCI Machine Learning Repository (2017) by Fanaee-T and Gama (2014).


Bike Share
Number of Rides
0
1000
3000
5000
7000
50
60
70
80
Temperature (degress Celius)

Using the classical Frequentist approach we could find the Ordinary Least Square (OLS) regression line:

```
fmod <- lm(rides ~ temp_feel, data = bikes)
summary(fmod)
```

Coefficients:	Estimate	Std. Error	t value	$\Pr(>\|t\|)$
(Intercept)	-2179.27	358.63	-6.077	2.44e-09 ***
temp_feel	81.88	5.12	15.992	<2e-16 ***


OLS line
Number of Rides
0
1000
3000
5000
7000
50
60
70
80
Temperature (degress Celius)

Model

- The regression equation with one covariate is *simple* linear regression model given by

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \tag{1}$$

  - β_0 is the intercept[3]
  - β_1 is the slope[4].
  - ϵ are the independent errors following a $N(0, \sigma^2)$

β_0 and β_1 are referred to as *regression coefficients* and can be stored in a vector $\beta = (\beta_0, \beta_1)$.

[3] the expected value of $Y \mid X = 0$

[4] the expected difference in Y for one unit increase in X

- You would have seen the Frequentist treatment of this model in Data 570.
- There we derived the so-called Least Squares Estimates for the regression coefficients which we denote $\hat{\beta}_i$.
- In the Bayesian treatment, we will arrive at a *posterior distribution for the slope and intercept*.
- Point estimates, of course, can be obtained from the posterior as we have done in previously.

- Bayesian simple linear regression is closely related to the normal models we've considered in previous lectures.

- In our normal-normal model we dealt with the simplified version which assumes σ^2 is known.

- We can generalize this to accommodate the more realistic problem that σ^2 is unknown by including a corresponding prior model:

$$\begin{aligned} Y_i|\mu,\sigma &\overset{ind}{\sim} N(\mu,\sigma^2) \\ \mu &\sim N(m,s^2) \\ \sigma &\sim \text{ some prior model.} \end{aligned} \tag{2}$$

- To use this in the linear regression setting, rather than the observations having a common population mean μ, the ith observation will have a mean dependant on the regression coefficients and the ith covariate.

- Let's look at the simple linear regression model with a single predictor/covariate X.

- In symbols we can write:

$$Y_i|\beta_0, \beta_1, \sigma \overset{ind}{\sim} N\left(\mu_i, \sigma^2\right) \tag{3}$$

$$\text{with } \mu_i = \beta_0 + \beta_1 X_i.$$

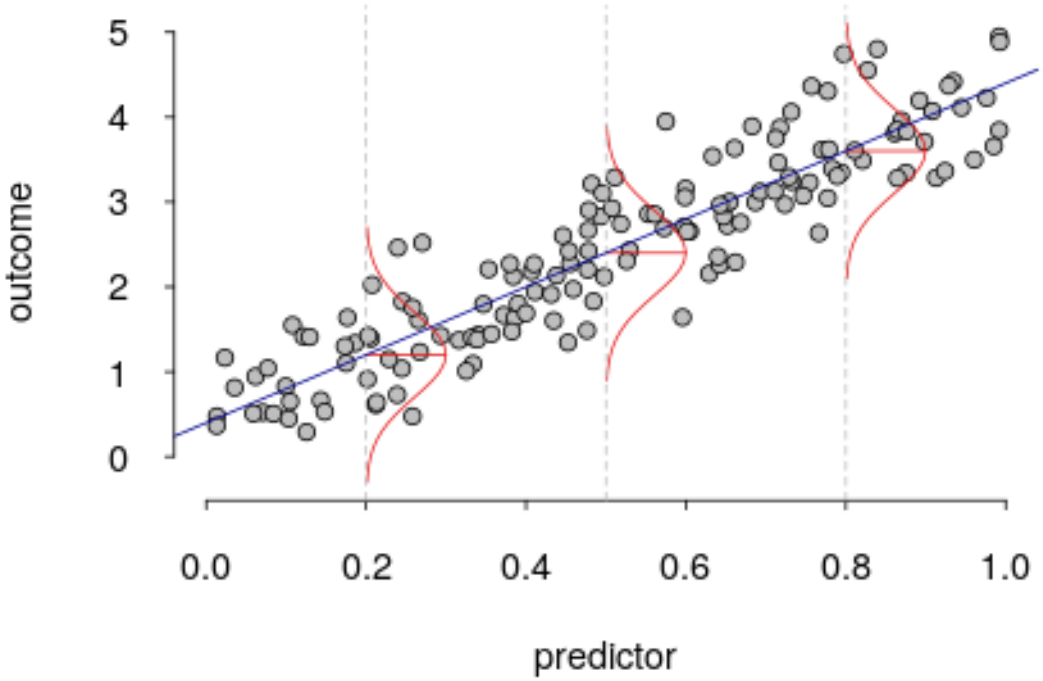


Figure: Source:
https://janhove.github.io/analysis/2019/04/11/assumptions-relevance

Normal regression assumptions

The appropriateness of the Bayesian Normal regression model (3) depends upon the following assumptions.

- Linearity: The relationship between X and the mean of Y is linear.
- Homoscedasticity: The variance of residual is the same for any value of X.
- Independence: Observations are independent of each other.
- Normality: For any fixed value of X, Y is normally distributed.

- To turn this into a Bayesian model, we must incorporate prior models for each of the unknown regression parameter
- There are countless approaches to this task, however, we adopt the default framework employed by **rstanarm**.
- The first assumption we'll make is that our prior models of β_0, β_1 and σ are independent[5].
- The intercept and slope regression parameters, β_0 and β_1 can technically take any values in the real line, while σ must be positive

[5] In practice we might have some prior notion about the combination of these parameters, however, this assumption greatly simplifies the model.

Centered covariates

- For a simplified run through of this Bayesian fit, let's consider *centering* the covariates so that our model becomes:

$$Y_i = \beta_{0c} + \beta_1(X_i - \overline{X_i}) + \epsilon_i \tag{4}$$

- While the interpretation of β_1 will not change, the intercept β_{0c} is now the expected value of Y when the *centered covariate* is 0, i.e. when $X = \overline{X}$.

- This is often much more meaningful than the regular β_0 interpretation (the expected difference in Y for one unit increase in X) and better facilitates specifying priors.[6]

[6] Furthermore, this transformation can improve the convergence of the corresponding MCMC algorithms that are used to fit these models.

- Assuming centered covariates, $i = 1, \ldots, n$, the sampling distribution used to write down the likelihood is:

$$y_i \mid x_i, \beta_{0c}, \beta_1, \sigma^2 \sim N(\beta_{0c} + \beta_1(x_i - \overline{x}), \sigma^2)$$

Hence our likelihood function could be written:

$$\begin{aligned}
&= \prod_{i=1}^{n} p(y_i \mid x_i, \beta_{0c}, \beta_1, \sigma^2) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y_i - (\beta_{0c} + \beta_1(x_i - \overline{x}))^2\right\} \\
&\propto \frac{1}{(\sigma^2)^{n/2}} \exp\left\{\frac{1}{2\sigma^2}\sum_i (y_i - \beta_{0c} - \beta_1(x_i - \overline{x}))^2\right\}
\end{aligned}$$

Thus, it's reasonable to utilize:

- Normal prior models for β_{0c} and β_1 which having real valued support:

$$\begin{aligned} \beta_{0c} &\sim N\left(m_0, s_0^2\right) \\ \beta_1 &\sim N\left(m_1, s_1^2\right) \end{aligned} \tag{5}$$

- Exponential[7] prior models for σ

$$\sigma \sim \text{Exp}(l). \tag{6}$$

where we can tune the m_0 , s_0 , m_1 , s_1 and l are hyperparameters to match our prior understanding of β_{0c}, β_1, and σ.

[7] Though this Exponential prior is currently the default in **rstanarm**, popular alternatives include the half-Cauchy and inverted Gamma models.

Hence the Bayesian simple linear regression model is given by

$$\begin{aligned}
\text{data:} \quad & Y_i|\beta_{0c},\beta_1,\sigma \overset{ind}{\sim} N\left(\mu_i,\sigma^2\right) \quad \text{with} \quad \mu_i = \beta_{0c} + \beta_1(X_i - \overline{X}) \\
\text{priors:} \quad & \beta_{0c} \sim N\left(m_0, s_0^2\right) \\
& \beta_1 \sim N\left(m_1, s_1^2\right) \\
& \sigma \sim \text{Exp}(l).
\end{aligned} \tag{7}$$

Bike Share

Based on past bikeshare analyses, suppose we have the following prior understanding of this relationship:

1. On an average temperature day, say 65 or 70 degrees for D.C., there are typically around 5000 riders, though this average could be somewhere between 3000 and 7000.
2. For every one degree increase in temperature, ridership typically increases by 100 rides, though this average increase could be as low as 20 or as high as 180.
3. At any given temperature, daily ridership will tend to vary with a moderate standard deviation of 1250 rides.

1. On an average temperature day, say 65 or 70 degrees for D.C., there are typically around 5000 riders, though this average could be somewhere between 3000 and 7000.

- Assumption 1 helps to inform the hyperparameters set for the centered intercept β_{0c}

- Hence we might adopt a prior assumption with a Normal model for β_{0c} which is centered at 5000 rides with a standard deviation of 1000 rides, and thus largely falls between 3000 and 7000 rides:

$$\beta_{0c} \sim N(5000, 1000^2)$$

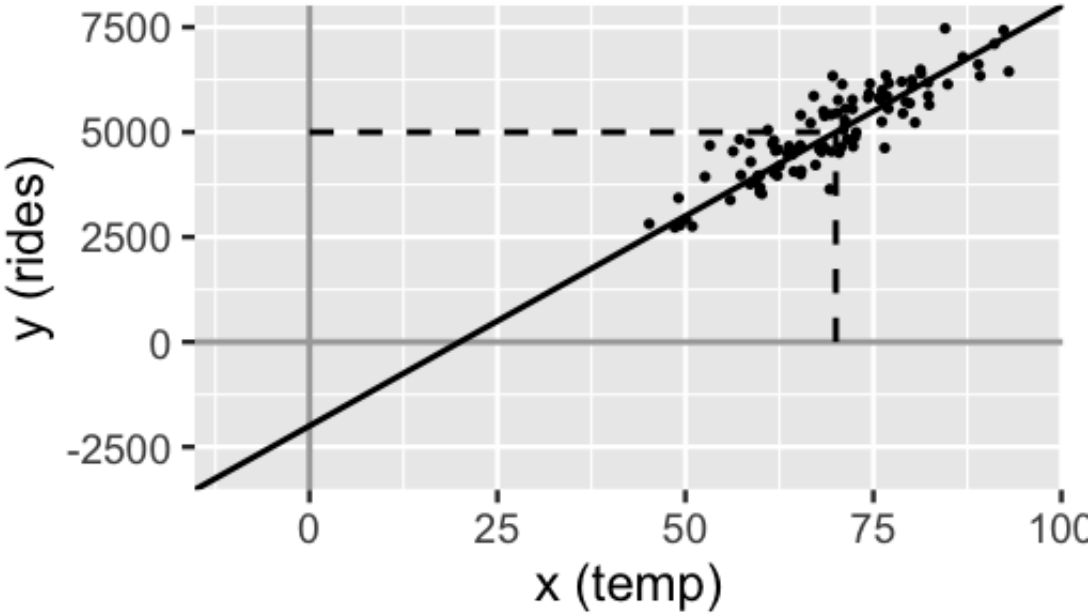


Figure: A simulated set of ridership data with intercept β_0 = - 2000 and centered intercept β_{0c}= 5000 at an average temperature of 70 degrees. Source: Ch9 of BR

2. For every one degree increase in temperature, ridership typically increases by 100 rides, though this average increase could be as low as 20 or as high as 180.

- This tells us about the rate of increase in ridership with temperature, and thus temperature coefficient β_1

- This prior understanding is well represented by a Normal model centered at 100 with a standard deviation of 40, and thus largely falls between 20 and 180:

$$\beta_1 \sim N(100, 40^2)$$

3. At any given temperature, daily ridership will tend to vary with a moderate standard deviation of 1250 rides.

- Recalling that the mean equations for an Explonential(l) r.v. we can set the hyperparameter such that the mean matches the expected standard deviation of 1250 rides

$$E(\sigma) = \frac{1}{l} = 1250$$

and thus $\sigma \sim Exp(0.0008)$

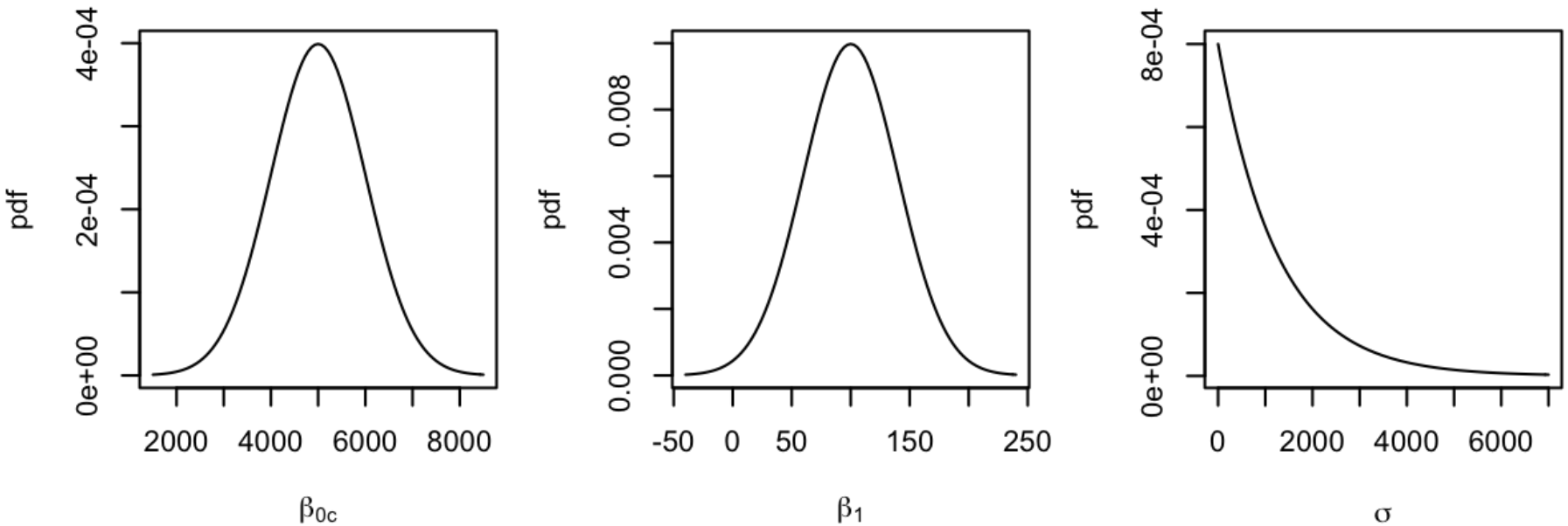


Figure: Prior models for the parameters in the regression analysis of bike ridership, (β_{0c}, β_1, and σ)

Plugging our tuned priors into (7), the Bayesian regression model of ridership (Y) by temperature (X) is specified as follows:

$$\begin{aligned} Y_i|\beta_0,\beta_1,\sigma &\overset{ind}{\sim} N\left(\mu_i,\sigma^2\right) \quad \text{with} \quad \mu_i = \beta_0 + \beta_1 X_i \\ \beta_{0c} &\sim N\left(5000, 1000^2\right) \\ \beta_1 &\sim N\left(100, 40^2\right) \\ \sigma &\sim \text{Exp}(0.0008). \end{aligned} \tag{8}$$

- Since we are assuming the independent priors we write the joint prior distribution as the product of the mariginal priors:

$$p(\beta_0, \beta_1, \sigma^2) = p(\beta_0)p(\beta_1)p(\sigma^2)$$

- To obtain our posterior we would need to multiply the likelihood found on slide 16 by the two normal priors from the β's times the Exponential prior to get ...

$$\begin{aligned} p(\beta_0, \beta_1, \sigma \mid \vec{y}) &\propto \text{prior} \cdot \text{likelihood} \\ &= p(\beta_0)p(\beta_1)p(\sigma) \cdot \left[\prod_{i=1}^{n} p(y_i \mid x_i, \beta_0, \beta_1, \sigma^2)\right] \\ &= \end{aligned}$$

- Since we are assuming the independent priors we write the joint prior distribution as the product of the mariginal priors:

$$p(\beta_0, \beta_1, \sigma^2) = p(\beta_0)p(\beta_1)p(\sigma^2)$$

- To obtain our posterior we would need to multiply the likelihood found on slide 16 by the two normal priors from the β's times the Exponential prior to get ...

$$\begin{aligned} p(\beta_0, \beta_1, \sigma \mid \vec{y}) &\propto \text{prior} \cdot \text{likelihood} \\ &= p(\beta_0)p(\beta_1)p(\sigma) \cdot \left[\prod_{i=1}^{n} p(y_i \mid x_i, \beta_0, \beta_1, \sigma^2)\right] \\ &= ??? \end{aligned}$$

If you went through the tedious work of plugging in the formulas you would *not* discover a familiar structure.

$$f(\beta_0, \beta_1, \sigma \mid \vec{y}) = \frac{\text{prior} \cdot \text{likelihood}}{\int \text{prior} \cdot \text{likelihood}}$$
$$= \frac{f(\beta_0)f(\beta_1)f(\sigma) \cdot [\prod_{i=1}^{n} f(y_i|\beta_0, \beta_1, \sigma)]}{\int\int\int f(\beta_0)f(\beta_1)f(\sigma) \cdot [\prod_{i=1}^{n} f(y_i|\beta_0, \beta_1, \sigma)]\, d\beta_0 d\beta_1 d\sigma}$$

If you really wanted to specify the posterior pdf, you'd need to calculate the normalizing constant as above ... but let's not.

- Instead, we can utilize Markov chain Monte Carlo simulation techniques to approximate the posterior.

- **rstanarm** uses **rstan**, R interface with Stan (see user guide here, YouTube channel here) on *applied regression models* (arm)

- To complete these exercises you will need to install[8] the **rstan** package by carefully following the directions at `https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started`.

Note: To complete these exercises you will need to install rstan by carefully following the directions here. This package requires a C++ compiler recognized by your system.

---

[8] this rstan which requires a C++ compiler recognized by your system

Rstan

- Stan is one of the popular[9] Bayesian engine (named after Stanislaw Ulam, the inventor of the modern version of MCMC) that employs a *Hamiltonian Monte Carlo algorithm*[10].
- Stan is an open source software that was designed to be faster and handle models that are out of reach for Gibbs samplers.
- Stan runs on various versions of Windows, Mac OS X and Linux.

[9] others include Bugs (Bayesian inference using Gibbs sampling) and JAGS (Just Another Gibbs Sampler), with Gibbs being another variation of the MH algorithm

[10] this is a variation of the Metropolis-Hastings (MH)algorithm

Using the `rstan` package we can fit the Bayesian linear model using the following:

```
> library(bayesrules)
> library(rstanarm)
> bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                         family = gaussian,
                         prior_intercept = normal(5000, 1000),
                         prior = normal(100, 40),
                         prior_aux = exponential(0.0008),
                         chains = 4, iter = 5000*2, seed = 84735)
```

Data information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

This species the conventional R formula as would have been used in the non-Bayesian `lm()` or `glm()`

`rides` our response variable Y

`temp_feel` our predictor variable (covariate) X

Data information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

This species where to find the data (if not `attach()`ed)

Data information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

- specifies that we are assuming a Normal data model.

Prior information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

- specifies the priors for β_{0c}, namely;

$$\beta_{0c} \sim N\left(5000, 1000^2\right)$$

Note: *please note that this is the centered intercept!*

Prior information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

- specifies the priors for β_1, namely;

$$\beta_1 \sim N\left(100, 40^2\right)$$

Prior information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

- specifies the priors for σ, namely;

$$\sigma \sim \text{Exp}\,(0.0008)$$

Note: *this prior is on standard deviation (not variance)*

Markov chain information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

the number of Markov chains to run

Markov chain information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

the number of iterations of each chain (Stan will use the first half of the chain as burn-in).

Markov chain information

```
bike_model <- stan_glm(rides ~ temp_feel, data = bikes,
                       family = gaussian,
                       prior_intercept = normal(5000, 1000),
                       prior = normal(100, 40),
                       prior_aux = exponential(0.0008),
                       chains = 4, iter = 5000*2, seed = 84735)
```

random seed for reproducibility

You can see which priors that uses by typing:

```
> prior_summary(bike_model)
Priors for model 'bike_model'
------
Intercept (after predictors centered)
 ~ normal(location = 5000, scale = 1000)

Coefficients
 ~ normal(location = 100, scale = 40)

Auxiliary (sigma)
 ~ exponential(rate = 8e-04)
------
See help('prior_summary.stanreg') for more details
```

```
> posterior <- as.array(bike_model)
> dim(posterior)
[1] 5000    4    3
> head(posterior)
, , parameters = (Intercept)

          chains
iterations   chain:1   chain:2   chain:3   chain:4
      [1,] -2656.894 -1909.885 -1945.143 -1948.812
      [2,] -2188.090 -2316.599 -2123.563 -2111.253
      [3,] -1983.816 -2298.671 -2063.761 -2159.332
      [4,] -2242.270 -2300.624 -2050.078 -1905.089
      [5,] -1640.718 -1917.427 -2240.187 -1823.851
      [6,] -3079.323 -1739.759 -2352.963 -1579.598
```

⋮

⋮

```
, , parameters = temp_feel

          chains
iterations  chain:1  chain:2  chain:3  chain:4
      [1,] 88.16061 78.42763 77.93586 78.39713
      [2,] 83.01252 83.50963 82.13081 82.46347
      [3,] 81.53971 84.60413 80.58994 82.73750
      [4,] 82.49201 83.74863 81.87918 78.83491
      [5,] 74.71256 78.50695 81.22644 77.56237
      [6,] 94.76976 76.36528 83.61065 74.16191
```

⋮

$\vdots$

```
, , parameters = sigma

          chains
iterations  chain:1  chain:2  chain:3  chain:4
      [1,] 1323.364 1235.235 1308.944 1169.699
      [2,] 1322.962 1338.478 1238.683 1365.358
      [3,] 1363.346 1222.411 1278.114 1364.053
      [4,] 1265.097 1253.676 1337.504 1295.273
      [5,] 1288.641 1318.117 1292.920 1276.152
      [6,] 1251.802 1353.870 1276.415 1282.493
```

Chain 1:

$$\texttt{(Intercept)} \left\{\beta_0^{(1)}, \beta_0^{(2)}, \ldots, \beta_0^{(5000)}\right\}$$

$$\texttt{temp\_feel} \left\{\beta_1^{(1)}, \beta_1^{(2)}, \ldots, \beta_1^{(5000)}\right\}$$

$$\texttt{sigma} \left\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(5000)}\right\}$$

Chain 2:

$$\texttt{(Intercept)} \left\{\beta_0^{(1)}, \beta_0^{(2)}, \ldots, \beta_0^{(5000)}\right\}$$

$$\texttt{temp\_feel} \left\{\beta_1^{(1)}, \beta_1^{(2)}, \ldots, \beta_1^{(5000)}\right\}$$

$$\texttt{sigma} \left\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(5000)}\right\}$$

Chain 3:

$$\texttt{(Intercept)} \left\{\beta_0^{(1)}, \beta_0^{(2)}, \ldots, \beta_0^{(5000)}\right\}$$

$$\texttt{temp\_feel} \left\{\beta_1^{(1)}, \beta_1^{(2)}, \ldots, \beta_1^{(5000)}\right\}$$

$$\texttt{sigma} \left\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(5000)}\right\}$$

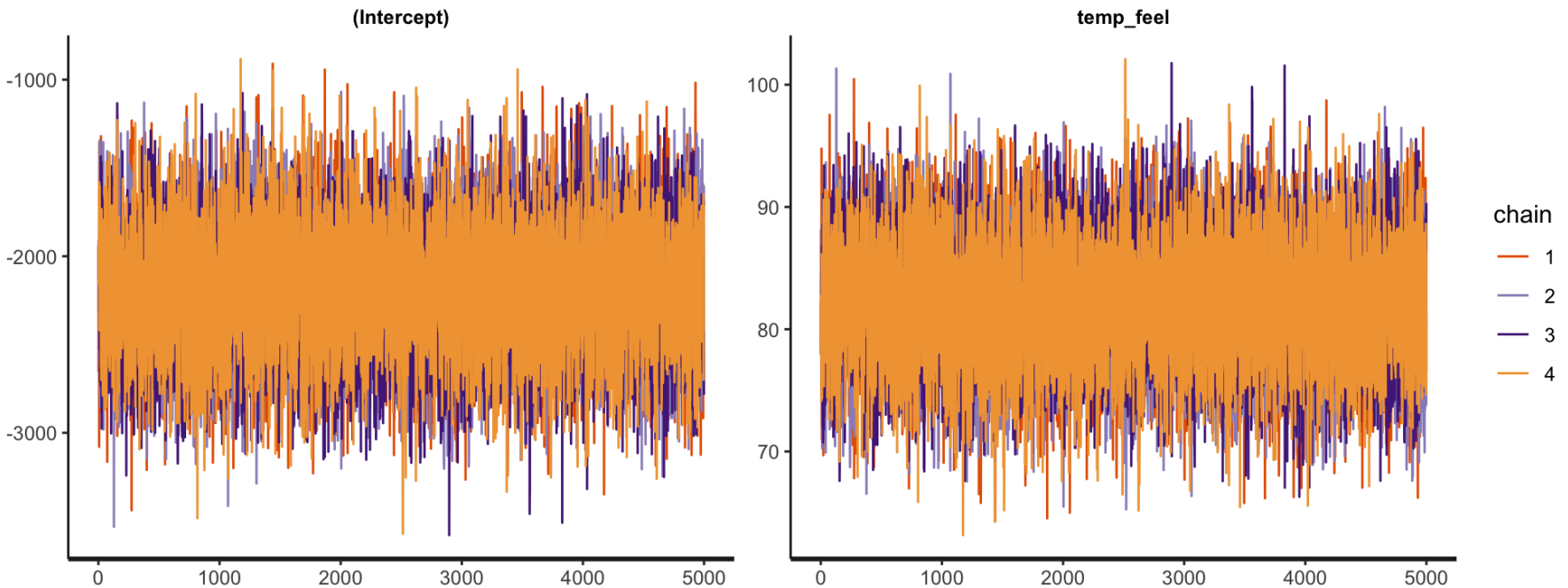
Chain 4:

$$\texttt{(Intercept)} \left\{\beta_0^{(1)}, \beta_0^{(2)}, \ldots, \beta_0^{(5000)}\right\}$$

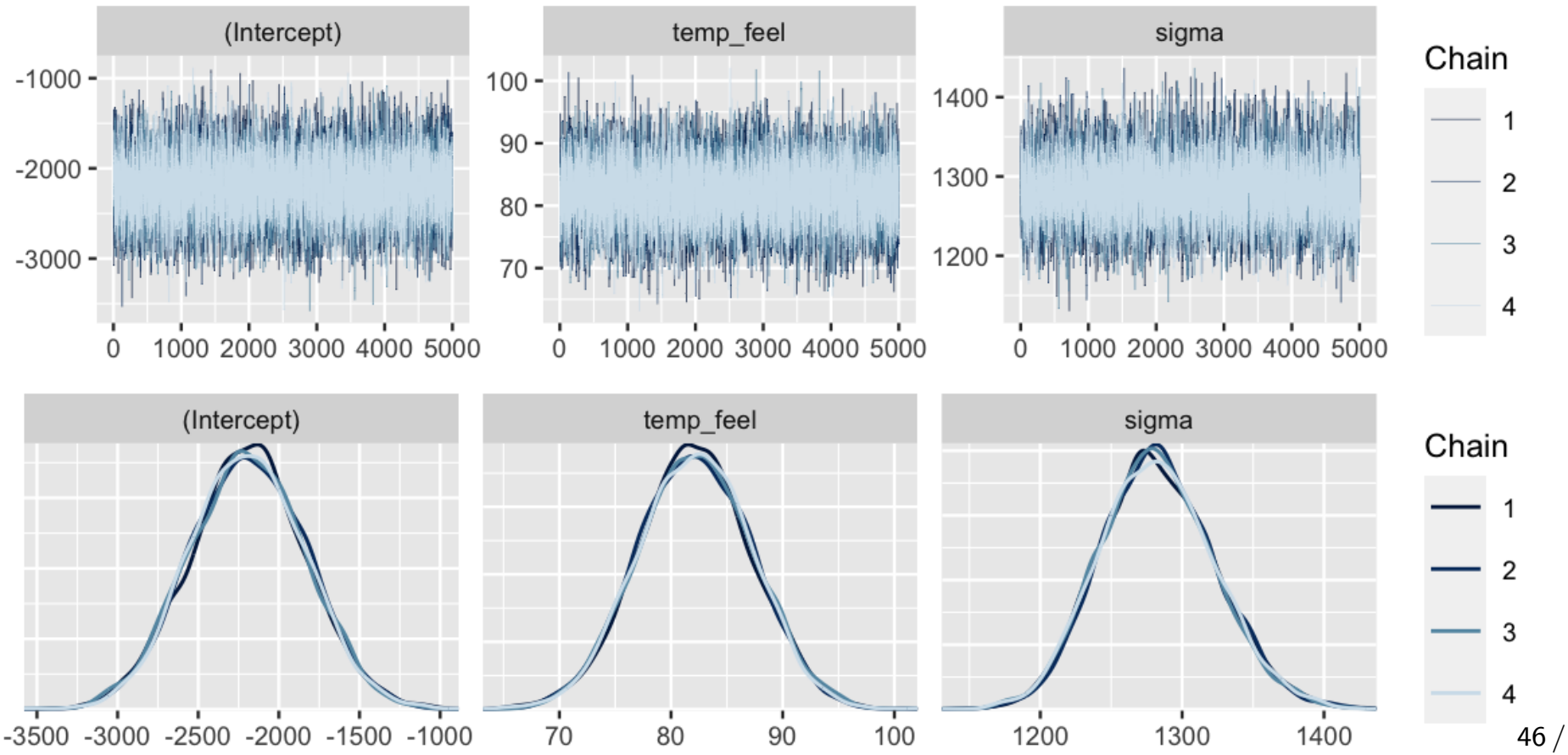
$$\texttt{temp\_feel} \left\{\beta_1^{(1)}, \beta_1^{(2)}, \ldots, \beta_1^{(5000)}\right\}$$

$$\texttt{sigma} \left\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(5000)}\right\}$$

```
library(rstan)
stan_trace(bike_model)
```



```
# Trace plots of parallel chains
mcmc_trace(bike_model)
# Density plots of parallel chains
mcmc_dens_overlay(bike_model)
```



- The split-$\hat{R}$ metric (or "*R-hat*" for short) provides a numerical supplement to the visual comparison of parallel chains.

- In addition to requiring that each individual Markov chain in our simulation be stable, we also want there to be consistency across the parallel chains.

- R-hat addresses this consistency by comparing the variability in sampled values across all chains *combined* to the variability *within* each individual chain.

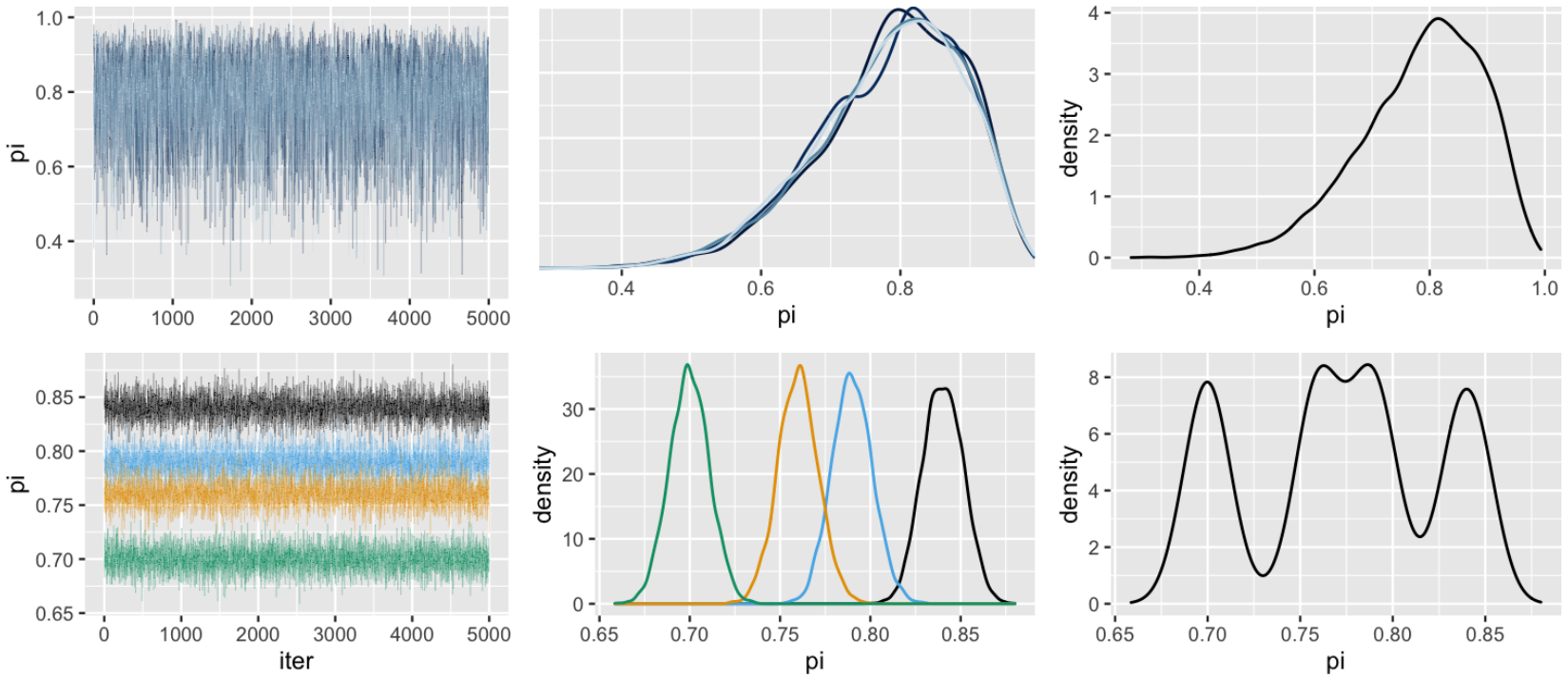


Figure: Simulation results for `bb_sim` (top row) and a hypothetical alternative (bottom row). Included are trace plots of the four parallel chains (left), density plots for each individual chain (middle), and a density plot of the combined chains (right). [Source: BR Fig 6.19]

- in a "good" Markov chain simulation, the variability across all parallel chains *combined* will be roughly comparable to the variability *within* any individual chain;

- in a "bad" Markov chain simulation, the variability across all parallel chains *combined* might exceed the typical variability *within* each chain.

Consider a Markov chain simulation of parameter θ which utilizes four parallel chains. Let $\text{Var}_{\text{combined}}$ denote the variability in θ across all four chains combined and $\text{Var}_{\text{within}}$ denote the typical variability within any individual chain. The R-hat metric calculates the ratio between these two sources of variability:

$$\text{R-hat} \approx \sqrt{\frac{\text{Var}_{\text{combined}}}{\text{Var}_{\text{within}}}}$$

Ideally, R-hat≈ 1, reflecting stability across the parallel chains. In contrast, R-hat> 1 indicates[11] instability, with the variability in the combined chains exceeding that within the chains.

[11] Though no golden rule exists, an R-hat ratio greater than 1.05 raises some red flags about the stability of the simulation.

The `rhat` function in the **bayesplot** package calculate the R-hat ratio

```
> rhat(bike_model)
(Intercept)   temp_feel       sigma
  0.9999203   0.9999217   1.0000107
```

The R-hat values are very close to 1, indicating that the chains are stable and consistent.

ESS

- Recall from our previous lecture, that an MCMC simulation produces a chain of *dependent* values.
- Let N be the length of a dependent Markov chain.
- The *effective sample size* (ESS) of this chain, N_{eff}, quantifies the number of independent samples it would take to produce an equivalently accurate posterior approximation (the bigger the better)

ESS

- It is typically true that the accuracy of a Markov chain approximation is only as good as that of a smaller independent sample.

- To put another way, its typically true that the *effective sample size ratio* is less than 1:

$$\frac{N_{eff}}{N} < 1$$

- As a general guide, we might be suspicious of a Markov chain for which the effective sample size ratio is less than 0.1, i.e., the effective sample size $< 10\%$

The `neff_ratio()` function in the **bayesplot** package provides the estimated effective sample size ratio

```
> neff_ratio(bike_model)
(Intercept)   temp_feel       sigma
    1.04185     1.03700     1.00360
```

The effective sample size ratios are slightly above 1 indicating that the chains are stable, mixing quickly, and behaving much like an independent sample.

- So now we have a sample from the posterior... what do we do now?
- While visualizations of the approximate posterior models are nice, we can also look at posterior summary statistics using the `tidy` function from the **broom.mixed** package.
- Let's first focus on the typical relationship between rideshare `rides` and temperature `temp_feel`
- Referring to the `tidy()` summary on the next slide, the posterior median relationship is

$$-2194 + 82.2X \tag{9}$$

```
# Posterior summary statistics
tidy(bike_model, effects = c("fixed", "aux"),
     conf.int = TRUE, conf.level = 0.80)
# A tibble: 4 x 5
  term        estimate std.error conf.low conf.high
  <chr>          <dbl>     <dbl>    <dbl>     <dbl>
1 (Intercept)  -2194.      362.    -2656.    -1732.
2 temp_feel       82.2      5.15     75.6      88.8
3 sigma         1281.      40.7    1231.     1336.
4 mean_PPD      3487.      80.4    3385.     3591.
```

Note: *mean_PPD is the sample mean of the posterior predictive distribution.*

Prediction

- An important feature of Bayesian inference is the existence of a predictive distribution for new observations.
- Suppose we observe $y_1, \ldots, y_n$ some population.
- If we want to predict a new observation $\tilde{y}$ from the same population that has yet to be observed.
- The predictive distribution of $\tilde{y}$ is the conditional distribution of $\tilde{y}$ given $y_1, \ldots, y_n$.

- Suppose a weather report indicates that tomorrow will be a 75-degree day in D.C. What's your posterior guess of the number of riders that Capital Bikeshare should anticipate?

- While it might be tempting to simply plop the value of $x = 75$ into (9) we can do a step better.

The *posterior predictive model* of a new data point Y_{new} accounts for:

- ***Sampling variability*** *in the data*
  How Y typically deviates from the model line (σ). That is, we don't expect every 75-degree day to have the same exact number of rides.
- ***Posterior variability*** *in parameters* (β_0 , β_1 , σ)
  The posterior median model is merely the center in a range of plausible model lines $\beta_0 + \beta_1 X$.

We should consider this entire range as well as that in σ, the degree to which observations might deviate from the model lines.

Prediction

- Let us make a distinction between two similar distributions:
- The distribution of the unknown but observable y is

$$p(y) = \int p(\theta, y) d\theta = \int p(y \mid \theta) p(\theta) d\theta$$
$$= \int \text{likelihood} \times \text{prior } d\theta$$

notice the use of the prior $p(\theta)$

- This is sometimes referred to as the *prior predictive distribution*[12]

[12] See 8.3 of BayesRules! for an example of this outside of the Regression settings

Prediction

The *posterior predictive distribution* for $\tilde{y}$ is *conditional on the observed data* and given by:

$$
\begin{aligned}
p(\tilde{y} \mid y) &= \int p(\tilde{y}, \theta \mid y) d\theta \\
&= \int p(\tilde{y}, \mid \theta, y) p(\theta \mid y) d\theta \\
&= \int p(\tilde{y}, \mid \theta) p(\theta \mid y) d\theta \qquad \text{since } \tilde{y} \text{ is independent of } y \\
&= \int \text{likelihood} \times \text{posterior } d\theta
\end{aligned}
$$

notice the use of the posterior $p(\theta \mid y)$

- Mathematically speaking, the posterior predictive model of a new data point Y_{new} for our Normal Regression model is:

$$f\left(y_{\text{new}}|\vec{y}\right) = \int\int\int f\left(y_{new}|\beta_0, \beta_1, \sigma\right) f(\beta_0, \beta_1, \sigma|\vec{y}) d\beta_0 d\beta_1 d\sigma.$$

- Mathematically speaking, the posterior predictive model of a new data point Y_{new} for our Normal Regression model is:

$$f\left(y_{\text{new}}|\vec{y}\right) = \int\int\int f\left(y_{new}|\beta_0, \beta_1, \sigma\right) f\left(\beta_0, \beta_1, \sigma|\vec{y}\right) d\beta_0 d\beta_1 d\sigma.$$

- But remember, our *posterior* did not have a functional form that we recognized.

- Hence we don't have a nice pdf that we could sub into the above equation, so we *approximate* the posterior predictive model.

Temptation:

$$\begin{array}{ccc} \beta_0^{(1)} & \beta_1^{(1)} & \sigma^{(1)} \\ \beta_0^{(2)} & \beta_1^{(2)} & \sigma^{(2)} \\ \vdots & \vdots & \vdots \\ \beta_0^{(20000)} & \beta_1^{(20000)} & \sigma^{(20000)} \\ \hline \hat{\beta}_0 & \hat{\beta}_1 & \hat{\sigma} \end{array}$$

(mean posterior estimates)

$$\hat{\beta}_0 + 75 \times \hat{\beta}_1$$

Using posterior predictive model we simulate using:

$$Y_{\text{new}}^{(i)} | \beta_0, \beta_1, \sigma \sim N\left(\beta_0^{(i)} + \beta_1^{(i)} \cdot 75, \left(\sigma^{(i)}\right)^2\right)$$

$\beta_0^{(1)}$	$\beta_1^{(1)}$	$\sigma^{(1)}$	$Y_{new}^{(1)}$	relies on $\mu^{(1)} = \beta_0^{(1)} + \beta_1^{(1)} \cdot 75$
$\beta_0^{(2)}$	$\beta_1^{(2)}$	$\sigma^{(2)}$	$Y_{new}^{(2)}$	relies on $\mu^{(2)} = \beta_0^{(2)} + \beta_1^{(2)} \cdot 75$
$\vdots$	$\vdots$	$\vdots$		
$\beta_0^{(20000)}$	$\beta_1^{(20000)}$	$\sigma^{(20000)}$	$Y_{new}^{(20000)}$	relies on $\mu^{(20000)} = \beta_0^{(20000)} + \beta_1^{(20000}$
$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\sigma}$	$\hat{Y}_{new}$	$\hat{\mu}$

The resulting collection of 20,000 predictions, $\left\{Y_{\text{new}}^{(1)}, Y_{\text{new}}^{(2)}, \ldots, Y_{\text{new}}^{(20000)}\right\}$ approximates the *posterior predictive* model of ridership Y on 75-degree days.

Note: *μ captures the uncertainty in the average ridership while $\hat{Y}$ incorporates the variability of the individual*

- You can build the posterior prediction model from scratch[13] (see 9.5.1 in BR), or use a shortcut from **rstanarm**

```
# Simulate a set of predictions
set.seed(84735)
shortcut_prediction <-
  posterior_predict(bike_model,
  newdata = data.frame(temp_feel = 75))
```

This object contains 20,000 predictions of ridership on 75-degree days.

[13] since we don't know the functional form of the posterior it requires simulation

We can both visualize and summarize the corresponding (approximate) posterior predictive model using our usual tricks.

```
# Construct a 95% posterior credible interval
posterior_interval(shortcut_prediction, prob = 0.95)
  2.5% 97.5%
1 1500  6482

# Plot the approximate predictive model
mcmc_dens(shortcut_prediction) +
  xlab("predicted ridership on a 75 degree day")
```

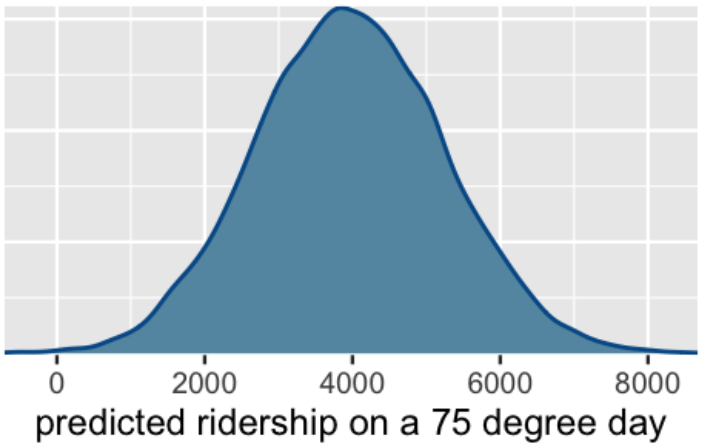


Figure: Posterior predictive model of ridership on a 75-degree day Fig 9.11

Comments

- The **rstanarm** package was written deliberately to mimic that of popular model fitting functions R. For example:

Popular function	rstan equivalent
`glm`	`stan_glm`
`aov`	`stan_aov`
`lme4::lmer`	`stan_lmer`

- However, Stan uses its own Stan syntax (not explicitly covered today) to tackle more general problems (even non-Bayesian ones).

- Other programming languages (eg. Python or Julia) also have interfaces to access Stan.