Alan Agresti, Maria Kateri, Ranjini Grove, and Antonietta Mira

R-Web Appendix of *Foundations of Bayesian Statistics for Data Scientists, with R and Python*

Contents

R0

Introduction to R-Web-Appendix

This web appendix provides supplementary material in R focusing on Bayesian analyses, complementing the examples and R-related information presented in the main text of the book as well as in its printed R-Appendix. Its aim is to further support readers in implementing Bayesian methods through additional code, explanations, and extended examples.

R is a widely used environment for statistical computing and data analysis, offering extensive capabilities for implementing modern statistical methodologies, including Bayesian approaches. It is freely available from the Comprehensive R Archive Network (CRAN) at , where users can also find a large collection of contributed packages. Its flexible syntax and rich system of available packages allow users to perform complex analyses, develop custom functions, and create high-quality visualizations. In particular, numerous packages available on CRAN facilitate Bayesian modeling, simulation, and inference.

A brief summary of the essentials of R, including basic examples, is available in Appendix A of Agresti and Kateri (2022) and in Chapter 0 of the R-Web-Appendix of that book, freely accessible at . Chapters 1 to 3 of the same Web-Appendix cover data handling and wrangling, exploratory data analysis, and probability distributions and data simulation, respectively, providing background material relevant to Chapter 1 of this book. Helpful introductory manuals for R include *An Introduction to R* and *R Data Import/Export*, both available on CRAN.

We do not provide an introduction to R here, but instead focus directly on its use for Bayesian analysis.

Sections marked with an asterisk (*) contain material that goes beyond the level and intended scope of this book. These sections are included for readers who wish to explore more advanced aspects of the topics and are not essential for understanding the main development of the text. Furthermore, note that the example analyses presented in the Web Appendix are sometimes fully worked-out. Hence, even though they may appear in earlier chapters, some of these analyses rely on methodological tools introduced later in the book, in particular the chapters on model assessment and diagnostics (Chapter 6). For this reason, readers may find it helpful to revisit certain Web Appendix examples after studying the diagnostic material in later chapters.

R1

Introduction to Bayesian Statistics with `R`

R1.1 `R` Software Packages for Bayesian Data Analysis

Modern Bayesian data analysis is often carried out using Stan (https://mc-stan.org), a highly capable platform designed for flexible and efficient probabilistic modeling. Within the R environment, Stan can be accessed via the `rstan` package.

In this book, many of the models presented in Chapters 3–7 were fitted using the `brms` package (Bayesian Regression Models using Stan: https://paulbuerkner.com/brms/). This package builds on Stan and offers a user-friendly interface that allows to specify models using familiar formula syntax, without the need to write Stan code directly. Users can define the model formula, supply the data, choose prior distributions, and select the likelihood family. Internally, `brms` translates these inputs into Stan code and performs the computation.

Another interface to Stan is provided by the `rstanarm` package, which offers a collection of precompiled models. This approach enables fast and convenient model fitting with a syntax similar to standard `R` functions such as `lm` or `glm`. The trade-off is that `rstanarm` supports only a limited set of model classes and provides less flexibility in the choice of prior distributions.

Compared to `rstanarm`, the `brms` package is more flexible. It can accommodate a broader spectrum of models, including non-linear, hierarchical, and multivariate specifications. In addition, it allows users to define custom prior distributions and offers extensive tools for model checking, diagnostics, and post-processing. As a result, `brms` is particularly well suited for more advanced or specialized modeling tasks. Instructions on how to install `rstan` and `brms` are provided in Section A.2.2 of the book's `R`-Appendix.

Beyond `rstan`, `brms`, and `rstanarm`, a number of other `R` packages are widely used in Bayesian workflows. The `coda` package provides essential tools for analyzing Markov chain Monte Carlo (MCMC) output, including convergence diagnostics, effective sample size calculations, and visualization methods (mainly discussed in Chapter 6). For model diagnostics and workflow integration, packages such as `bayesplot` and `posterior` are particularly useful; they are designed to work seamlessly with Stan-based outputs (including those from `rstan` and `brms`) and provide advanced visualization and summary tools for posterior distributions and convergence checks. The `loo` package implements approximate leave-one-out cross-validation (s. Chapters 4, 7), while `bridgesampling` enables estimation of marginal likelihoods for Bayesian model selection (s. Chapter 7).

The `MCMCpack` package offers a collection of Bayesian models with built-in MCMC algorithms, allowing users to fit standard models without relying on Stan. Furthermore, `LearnBayes` focuses on educational implementations of Bayesian methods and includes functions for posterior simulation and prior specification. The `BayesFactor` package, in contrast, is oriented toward Bayesian hypothesis testing and model comparison via Bayes factors (s. Chapter 2), providing convenient functionality for common experimental designs.

In addition, interfaces to other probabilistic programming frameworks are available, such as `rjags` for JAGS (Just Another Gibbs Sampler) and `nimble`. The latter is an extension

of the earlier software BUGS (Bayesian inference Using Gibbs Sampling) that allows flexible specification of hierarchical models and custom MCMC algorithms within R.

Together, these packages constitute a rich toolkit for Bayesian data analysis, facilitating model fitting, validation, comparison, and effective communication of results.

R1.2 Implementation and Visualization of R Functions for Probability Distributions

Figure 1.3 plots the Beta posterior *pdf* with parameters $\alpha_1 = 750$ and $\alpha_2 = 580$ for U.S. population proportion who believe a woman should be able to get an abortion for any reason, allocating the sample proportion with a the red diamond. The R code for deriving this figure follows.

```
> p = seq(0, 1, 0.0001)
> plot(p, dbeta(p,750,580),
+    ylab=expression("Posterior pdf  p("*pi*"| "*alpha[1]*","*alpha[2]*")"),
+    xlab=expression("Binomial parameter"~~italic(pi)),
+    type ="l", lwd=2, col="dodgerblue4")
 > points(0.564,0,pch=18,col="red4")    # sample proportion
```

In Section 1.3.5 we provided the R code for computing single quantiles or values of the cumulative density function of a beta distribution. Actually, we can provide a vector to the respective argument of the related functions, as shown below:

```
> qbeta(c(0.025, 0.975), 750, 580)
[1] 0.5371820 0.5904555 # 0.025 and 0.975 quantiles of beta distr. with param's 750, 580
> x <- seq(0,1,0.2)
> cdf <- pbeta(x, 750, 580)
> names(cdf) <- x
> cdf
            0           0.2           0.4           0.6           0.8             1
 0.000000e+00 2.377359e-187  1.066693e-33  9.962261e-01  1.000000e+00  1.000000e+00
```

Similarly can be evaluated the probability distribution functions for other probability distributions of Table A.1 of book's R-Appendix.

R1.3 The Multinomial Distribution in R

Next, we provide an example of a multinomial distribution $\text{Mult}(n, \boldsymbol{\pi})$. In particular, we compute the value of the probability mass function (pmf) of a $\text{Mult}(10, \boldsymbol{\pi})$ distribution with $\boldsymbol{\pi} = (0.2, 0.5, 0.3)$ at the point $\boldsymbol{x} = (2, 4, 4)$:

```
> n <- 10; p <- c(0.2, 0.5, 0.3)   # parameters of a multinomial distr.
> names(p) <- c("A", "B", "C")
> x <- c(2, 4, 4)  # evaluate the probability of a specific realization
> dmultinom(x, size = n, prob = p) # value of pmf at x
[1] 0.0637875
```

We can also evaluate multiple outcomes:

```
> X <- rbind(
+   c(2, 5, 3),  # observed multinomial sample 1
```

```
+   c(1, 6, 3),  # observed multinomial sample 2
+   c(3, 4, 3)   # observed multinomial sample 3
+ )
> apply(X, 1, function(row) dmultinom(row, size = n, prob = p))
[1] 0.085050 0.070875 0.056700
```

Next, we generate 5 samples from a Mult$(n, \boldsymbol{\pi})$ distribution, with $n = 10$ and $\boldsymbol{\pi} = (0.2, 0.5, 0.3)$.

```
> set.seed(2026)
> samples <- rmultinom(5, size = n, prob = p)   # random numbers generation
> samples
  [,1] [,2] [,3] [,4] [,5]
A    3    1    2    2    1
B    4    6    7    4    5
C    3    3    1    4    4
> sample1 <- samples[,1]     # sample 1 in a vector
```

The following code illustrates graphically (see Figure R1.1) the probability mass function (pmf) of the above used multinomial distribution with $n = 10$ and $\boldsymbol{\pi} = (0.2, 0.5, 0.3)$. Since the support of a multinomial distribution with three categories is constrained by $x_1+x_2+x_3 = n$, the plot is shown in the two-dimensional representation of the feasible set (x_1, x_2), with x_3 determined implicitly. Each point corresponds to a valid outcome of the experiment, and both the color intensity and point size are proportional to the value of the pmf at the respective point. The dashed line indicates the boundary of the support given by $x_1+x_2 = n$, beyond which no probability mass exists.

```
> n <- 10; p <- c(0.2, 0.5, 0.3)
> grid <- expand.grid(x1 = 0:n, x2 = 0:n)
> grid <- subset(grid, x1 + x2 <= n)
> grid$x3 <- n - grid$x1 - grid$x2
> grid$prob <- apply(grid, 1, function(x)
+   dmultinom(x, size = n, prob = p) )

> col_fun <- colorRampPalette(c("lightblue", "dodgerblue4"))  # color scale
> cols <- col_fun(100)[as.numeric(cut(grid\$prob, 100))]

> plot(grid\$x1, grid\$x2,
+      pch = 21, bg = cols,
+      cex = 1 + 6 * grid\$prob / max(grid\$prob), main = "",
+      xlab = expression(x[1]\~"(count category A)"),
+      ylab = expression(x[2]\~"(count category B)") )
> abline(a = n, b = -1, lty = 2)    # simplex boundary (x1 + x2 = n)
```

R1.4 Posterior Intervals in R

Highest posterior density (HPD) intervals are preferred when the posterior *pdf* is monotone increasing or decreasing from the boundary of the parameter space. An example is interval estimation of a binomial parameter π when all n trials are successes ($y = n$) or all are failures ($y = 0$). For such an example, see Section A.4.7 of the R-Web-Appendix at `https://stat4ds.rwth-aachen.de`, which obtains the HPD interval using the `hpd` function of `TeachingDemos`.

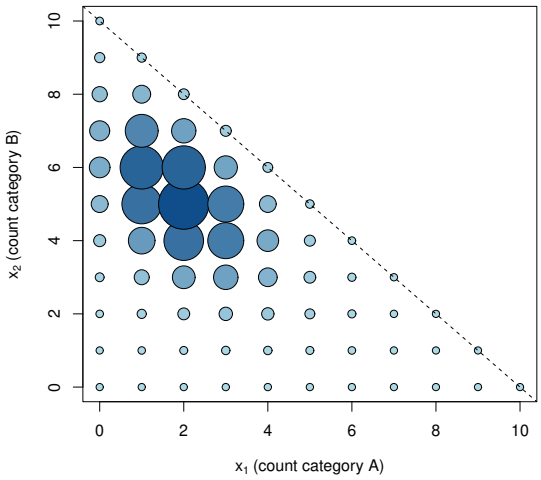

FIGURE R1.1: Scatterplot representation of the multinomial probability mass function for $n = 10$ and probability vector $\boldsymbol{\pi} = (0.2, 0.5, 0.3)$. Each point corresponds to a valid outcome $\boldsymbol{x} = (x_1, x_2, x_3)$ with $x_3 = n - x_1 - x_2$. Color intensity (light to darker blue) and point size are proportional to the probability mass. The dashed line indicates the boundary of the support $x_1 + x_2 = n$.

R2

R for Bayesian Inference for Proportions

R2.1 On the Logistic Distribution Prior for the Log-Odds

In computing the Bayes Factor in Section 2.3.5 of the book for testing a simple null hypothesis for a proportion for the example in Sections 2.3.2 and 2.3.4, we used in Section A.2.1 of the book appendix the `proportionBF` function in the `BayesFactor` package with an alternative prior. In particular, we considered a logistic distribution prior for the log-odds $\phi = \log(\gamma) = \log\left(\pi/(1-\pi)\right)$, centered on $\phi_0 = \log(\frac{\pi_0}{1-\pi_0})$, the log-odds corresponding to the null value π_0, i.e., we consider $\phi \sim \text{logistic}(\text{mean} = \phi_0,\ \text{scale} = r)$.

If $\pi_0 = 0.5$, as in our example, $\phi_0 = 0$ and the logistic prior distribution is symmetric around 0. The three options of logistic prior are visualized in Figure R2.2 (left). The default choice for r is $r = 1/2i$, corresponding to the `rscale = "medium"` argument of the `proportionBF` function. Alternative options are $r = \sqrt{2}/2$ (`rscale = "wide"`) and $r = 1$ (`rscale = "ultrawide"`). Using the option of the widest prior, we get $BF_{10} = 11.815$, very close to the value obtained with a uniform prior for π:

```
> bf = proportionBF(y = 63, N = 111, p = 0.5, rscale="ultrawide", nullInterval =c(0,0.5))
> bf
Bayes factor analysis
--------------
[1] Alt., p0=0.5, r=1 0<p<0.5    : 0.05050399 ±0%  # BF_{10} = 0.5967/0.050504 %= 11.815
[2] Alt., p0=0.5, r=1 !(0<p<0.5) : 0.596708   ±0%
Against denominator:
  Null, p = 0.5
```

R2.2 Bayes Factor for Comparison of Two Proportions

Sections 2.4.2 and 2.4.3 presented Bayesian methods for comparing two proportions. By viewing the data as a contingency table, we can use the methods of Section 2.5 for categorical variables having possibly multiple categories. We illustrate for the example in Section 2.4.2 of comparing the probability of belief in hell for females and males, expressing the data in a 2×2 contingency table. The equality of the probabilities $\pi_1 = P(\text{yes} \mid \text{female})$ and $\pi_2 = P(\text{yes} \mid \text{male})$ is then equivalent to the homogeneity of the distribution of "belief in hell" across gender, meaning that "belief in hell" does not depend on gender.

We can obtain the Bayes factor for evaluating the plausibility of independence, discussed in Section 2.5.5, with the `contingencyTableBF` function of the `BayesFactor` library for contingency tables, discussed there. Here, we treat the row totals of the tables, which are the gender sample sizes, as fixed, so we use the option `indepMulti` for the underlying sampling scheme.

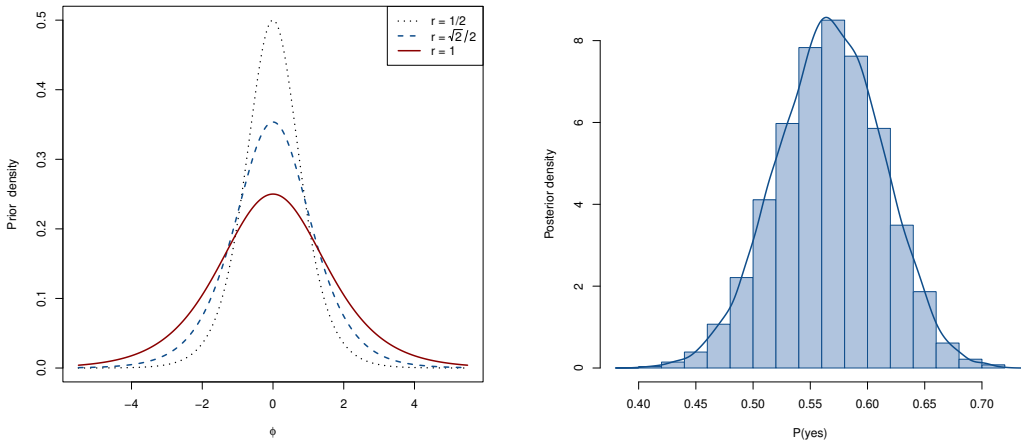

FIGURE R2.1: Logistic prior distributions for an odds ϕ, corresponding to a binomial probability π (left). Histogram of 10,000 samples from the posterior of π, given 63 yes responses (out of 111 subjects with graduate degree) and a logistic prior for the corresponding ϕ, centered at $\phi_0 = 0$ and with scale $r = 1$ (right).

```
> data_table <- matrix(c(498, 316, 176, 152), ncol=2)
> colnames(data_table) <- c("yes", "no"); rownames(data_table) <- c("female", "male")
> data_table
        yes  no
female 498 176
male   316 152
> library(BayesFactor)
> bf = contingencyTableBF(data_table, sampleType = "indepMulti", fixedMargin = "rows")
> bf
Bayes factor analysis
[1] Non-indep. (a=1) : 1.035295 ±0%
Against denominator:
  Null, independence, a = 1
```

With a Bayes factor of 1.035, these data provide very weak evidence against the null hypothesis that females and males are equally likely to believe in hell. This differs substantially from the Bayes factor reported in Section 2.4.3, because the two analyses address different hypotheses. Here, the hypotheses are $H_0 : \pi_1 = \pi_2$ versus $H_1 : \pi_1 \neq \pi_2$, so the Bayes factor is non-directional. In Section 2.4.3, the hypotheses are $H_0 : \pi_1 \geq \pi_2$ versus $H_1 : \pi_1 < \pi_2$, making the Bayes factor directional. One-sided alternatives focus the posterior on a smaller region, so even modest differences can yield a very large Bayes factor, reflecting strong evidence for that specific direction.

Directional information can be obtained from the non-directional posterior by examining the distribution of the difference $\pi_1 - \pi_2$. This is achieved by sampling from the posterior distribution of all cell probabilities, which yields samples of π_1 and π_2 for further inference:

```
> posterior_samples = posterior(bf, iterations = 10000)
> head(posterior_samples)
        pi[1,1]   pi[2,1]   pi[1,2]   pi[2,2]   pi[1,*]   pi[2,*] omega[1,1] omega[2,1] ...
 [1,] 0.4430906 0.2778787 0.1507173 0.1283134 0.5938079 0.4061921 0.7461850 0.6841066 ...
 [2,] 0.4284458 0.2954693 0.1487615 0.1273234 0.5772073 0.4227927 0.7422737 0.6988515 ...
 ...
```

```
# samples from posterior distribution:
> yes_female = posterior_samples[,"pi[1,1]"] / posterior_samples[,"pi[1,*]"] # of pi_1
> yes_male = posterior_samples[,"pi[2,1]"] / posterior_samples[,"pi[2,*]"]   # of pi_2
> quantile(yes_female - yes_male, c(0.025,0.975))
      2.5%      97.5%               # 95% posterior interval for pi_1 - pi_2
0.01027503 0.11719095
> hist(yes_female-yes_male, freq=FALSE, main="", xlab=expression(pi[1]-pi[2]),
+      ylab="Posterior density", col = "lightsteelblue", border="dodgerblue4")
> dens <- density(yes_female-yes_male)
> lines(dens, lwd=2, lty=1, col="dodgerblue4")                # Plot in Figure 2.2
sum(yes_female-yes_male<0)/10000    # proportion of simulated samples with pi_1 - pi_2 < 0
[1] 0.01
```

The posterior interval computed above closely matches the corresponding interval (0.01016, 0.11763) reported in Section 2.4.2.

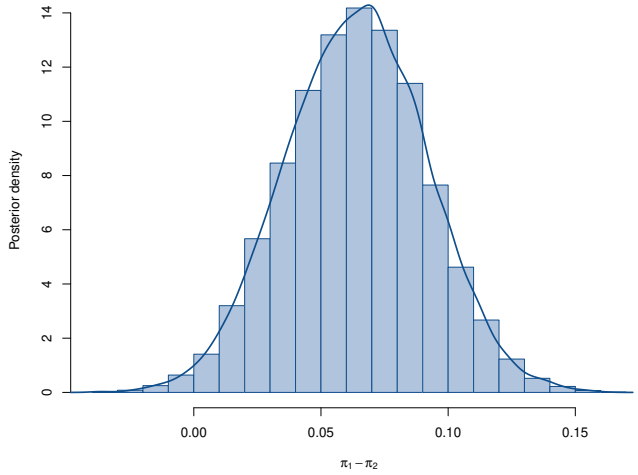

FIGURE R2.2: Histogram of 10,000 samples from the posterior distribution of $\pi_1 - \pi_2$, for the belief in hell data for females and males.

R2.3 Credible Interval for a Proportion

Revisiting the example of Section 2.3.2, we shall find posterior intervals for the population proportion π of those with a graduate degree who would believe in hell in the `brms` package, using the uniform Beta(1, 1) prior.

```
> y <- c(rep(1,63), rep(0,48))   # create vector of binary responses
> bin.data <- data.frame(y)      # define it as a data frame
> fit <- brm(data = bin.data, family = bernoulli(link = identity), formula = y ~ 1,
+          prior(beta(1, 1), class = Intercept, lb = 0, ub = 1), iter=6000, warmup=1000)
> print(summary(fit), digits = 7)
 Family: bernoulli
  Links: mu = identity
Formula: y ~ 1
   Data: bin.data (Number of observations: 111)
```

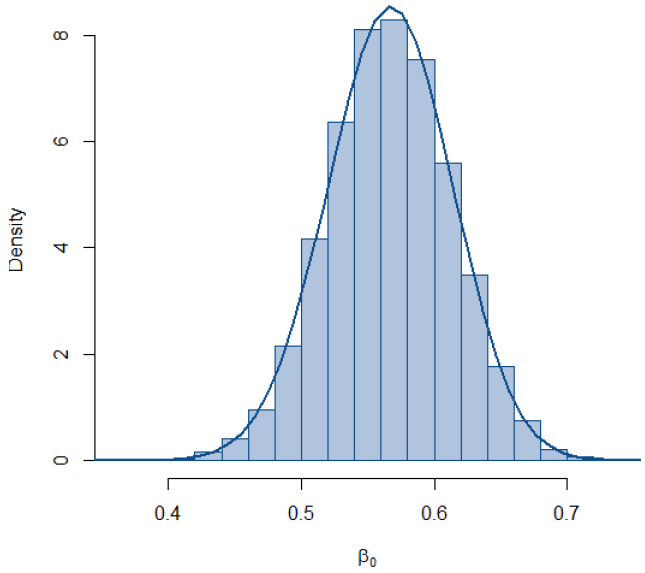

FIGURE R2.3: Histogramms of draws from the posterior distribution for the intercept β_0 of the model fitted on the Believe to Hell data of subjects with a graduate degree, along with the probability density function of the Beta(64, 49) posterior.

```
  Draws: 4 chains, each with iter = 6000; warmup = 1000; thin = 1;
         total post-warmup draws = 20000
Regression Coefficients:
           Estimate Est.Error  l-95% CI  u-95% CI
Intercept 0.5665342 0.0462126 0.4755825 0.6564730
```

We verify the results we obtained in Section 2.3.2 based on the Beta(64, 49) posterior. Figure R2.3 pictures the histogram of the draws from the posterior of π and is derived as follows.

```
> draws <- as.data.frame(fit)\$b_Intercept  # draws from the posdterior of the intercept
> hist(draws, freq=FALSE, main="", xlab= expression(beta[0]),  ylab="Density",
       col = "lightsteelblue", border="dodgerblue4")
p <- seq(0, 1, length=100)
lines(p, dbeta(p,64,49), lwd=2, lty=1, col="dodgerblue4")
```

R2.3.1 Simultaneous Confidence Region for a Probability Vector

We shall reconsider the Happiness example of Section 2.5.3, based on the 2021 General Social Survey. The sample frequencies in the (very happy, pretty happy, not too happy) categories were (778, 2304, 921) with corresponding proportions (0.194, 0.576, 0.230), which are also the ML estimates of the corresponding happiness category probabilities. That section obtained a posterior interval for the probability of *each* happiness category π_j, $j = 1, 2, 3$, based on using a beta conjugate prior distribution. Now, we present *simultaneous* confidence or posterior regions $\mathcal{C}$ for the entire probability vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3)$, i.e. $\mathcal{C}$ is a subset of the simplex $\Delta = \{(\pi_1, \pi_2, \pi_3) : \ \pi_i \in [0, 1], \ i = 1, 2, 3, \ \ \sum_{i=1}^{3} \pi_i = 1\}$. Confidence or posterior regions for a probability vector account for the dependencies between the categories that arise from the multinomial nature of the data (for every point (probability vector) in

the region, the sum of all its components (probabilities of multinomial categories) must be 1). Binomial confidence or posterior intervals are simpler, focusing on individual category probabilities, but they do not respect the underlying structure of the multinomial model and are inappropriate for capturing the joint uncertainty. Thus, the former are preferred when we are interested in the joint distribution of $\boldsymbol{\pi}$.

A simultaneous frequentist confidence region can for example be obtained in `R` package `MultinomialCI`, as illustrated below. [1]

```
> library(MultinomialCI)
> data <- c(778, 2304, 92)         # sample vector
> m = multinomialCI(data, 0.05)    # 95% simultaneous confidence region
> print(paste("very happy: [", m[1,1], m[1,2], "]"))
[1] "very happy: [ 0.178366225331002 0.210482448850811 ]"
> print(paste("pretty happy: [", m[2,1], m[2,2], "]"))
[1] "pretty happy: [ 0.559580314763927 0.591696538283736 ]"
> print(paste("not too happy: [", m[3,1], m[3,2], "]"))
[1] "not too happy: [ 0.214089432925306 0.246205656445115 ]"
```

The Bayesian posterior region, based on the Dirichlet posterior distribution with support the simplex Δ (see Section 2.5.2), can be visualized by a *ternary plot* (*triangular plot*), as shown in Figure R2.5, which is derived as follows.

```
> library(DirichletReg)      # package to simulate from a Dirichlet distribution
> library(ggtern)            # package for deriving ternary plots
> data <- c(778, 2304, 921)
> a.post <- data+1; a.post   # parameters of the Dirichlet posterior distribution
[1]  779 2305  922           # for prior Di(1,1,1)
> simul.data <- rdirichlet(100000, alpha = a.post)  # simulate a sample from the posterior
> dat <- as.data.frame(simul.data)         # for plotting the data must be in a data frame
> names(dat)[1] <- "X1"; names(dat) [2] <- "X2"; names(dat)[3] <- "X3";
> head(dat,3)                              # the first 3 rows of the data frame
     X1        X2        X3
1 0.1937692 0.5744340 0.2317968
2 0.1908700 0.5821952 0.2269347
3 0.1882806 0.5779661 0.2337532
> mle <- as.data.frame(t(data/sum(data))) # ML as data frame to be added on the plot
> names(mle)[1] <- "X1"; names(mle)[2] <- "X2"; names(mle)[3] <- "X3"; mle
     X1        X2        X3
1 0.1943542 0.5755683 0.2300774

> (p <- ggtern(data = dat, aes(X1, X2,X3)) +                 # ternary plot Fig.A.1 (left)
+   geom_point(alpha = 0.1, size = 1, color = "orange") +    # of simulated data (orange)
+   theme_rgbw() + geom_confidence_tern(breaks = c(0.95)) )  # with 95% contour

> p + scale_L_continuous(limits=c(0.15,0.32)) +    # focused ternary plot Fig.A.1 (middle)
+     scale_T_continuous(limits=c(0.50,0.67)) +    # with ML (black point)
+     scale_R_continuous(limits=c(0.18,0.35)) +
+     geom_point(data=mle, color = "black")

> ggtern(data = dat, aes(X1, X2,X3)) +                      # ternary plot Fig.A.1 (right)
+  theme_rgbw() + geom_confidence_tern(breaks = c(0.95)) +        # without simulated data
+  scale_L_continuous(limits=c(0.17,0.27)) +
+ scale_T_continuous(limits=c(0.55,0.65)) +
+ scale_R_continuous(limits=c(0.18,0.28)) +
+ geom_point(data=mle, color = "black")
```

A ternary plot is used to visualize three dependent variables (here, the components of the probability vector of a multinomial random vector with three categories) that sum

[1] Methodological details can be found in Sison CP and Glaz J (1995). Simultaneous confidence intervals and sample size determination for multinomial proportions. *Journal of the American Statistical Association*, 90:366-369.

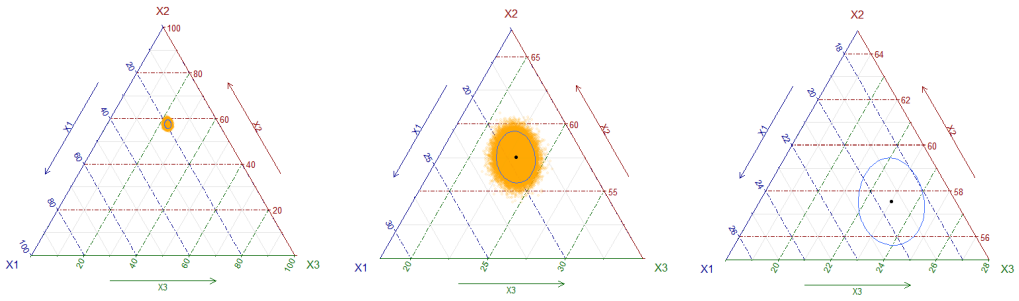

FIGURE R2.4: Ternary plots for the Dirichlet posterior distribution with estimated parameters $(\alpha_1, \alpha_2, \alpha_3) = (779, 2305, 922)$ for the Happiness data, along with the 95% posterior region (95% contour of the posterior) and simulated data from the posterior (in yellow, in the left and middle plots). The ML is marked in black. The left plot shows the complete ternary plot (the probabilities of the three categories of the multinomial distribution range from 0 to 100%), while the other two focus in the posterior region.

to a constant (here, 1 or 100%). Each corner represents one of the three variables (here, denoted by $X1$, $X2$, and $X3$). The sum of the three variables is constant, equal to 1 (100%) at every point (on the corners, edges or in the triangle). The corners correspond to the probability vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, while the edges represent probability vectors where one of the components (probabilities) is equal to 0. For example, the $X1 - X2$ edge corresponds to vectors in the set $\{(\pi_1, \pi_2, 0) : \pi_1, \pi_2 \in [0, 1] \text{ with } \pi_1 + \pi_2 = 1\}$. In such a plot, the 95% posterior region is framed by the 95% contour of the posterior distribution.

R2.4 Bayesian Prediction of a Future Observations: The Beta-Binomial Distribution

We have seen in Section 2.3.6 that for a binary random variable $Y \sim \mathcal{B}(n, \pi)$, the posterior predictive distribution for the future number of successes Y_f in some future number n_f of observations is obtained by averaging the binomial conditional distribution of Y_f for a given value of π with respect to the beta posterior distribution of π, yielding a *beta-binomial distribution* (s. also Exercise 2.43).

There, we illustrated the beta-binomial distribution for the example in Section 2.2.2 of the British tea taster, computing for the next $n_f = 5$ cups, the beta-binomial distribution for the number of correct guesses Y_f, using the `extraDistr` package.

Alternatively, the same analysis can be carried out using the `VGAM` package, which uses an equivalent parametrization that expresses the beta–binomial distribution in terms of a mean parameter $\pi = \alpha_1/(\alpha_1 + \alpha_2)$ and a dispersion parameter ρ, which controls overdispersion relative to the binomial model. Specifically,

$$\rho = \frac{1}{\alpha_1 + \alpha_2 + 1}.$$

In this formulation, for our example for $\alpha_1 = 11$ and $\alpha_2 = 1$, the argument `prob` corresponds to $\pi = 11/12$ and the dispersion parameter becomes $\rho = 1/13$:

```
> library(VGAM)
> y <- 0:5; dbetabinom(y, size=5, prob=11/12, rho=1/13) # rho = 1/(alpha_1 + alpha_2 + 1)
[1] 0.0002 0.0025 0.0151 0.0655 0.2292 0.6875
```

The parameter ρ can be interpreted as the degree of clustering or extra-binomial variation. When $\rho \to 0$, the beta–binomial converges to the standard binomial model with no overdispersion, whereas larger values of ρ correspond to stronger heterogeneity. In our case, we can compare the beta-binomial distribution to the binomial for the same n and success probability p:

```
> dbinom(y, 5, 11/12)
[1] 0.000004 0.00022 0.00486 0.053490 0.29419 0.64723
```

R2.5 Bayesian Posterior Interval for the Difference of Proportions in Independent Binomial Samples

In Section 2.4.2 we constructed a Bayesian posterior interval for $\pi_1 - \pi_2$ by assigning independent beta prior distributions for π_1 and π_2. This was illustrated using data from the 2018 General Social Survey by comparing the proportions of females and males who believe in hell. There, an equal-tailed credible interval was obtained by drawing independent samples from the Beta posterior distributions of π_1 and π_2, computing the resulting simulated differences, and taking empirical quantiles. This yielded a 95% percentile interval for $\pi_1 - \pi_2$ given by (0.01016046, 0.11763070).

Alternatively, the same Bayesian credible interval can be computed more directly using the `TeachingDemos` package, which implements Monte Carlo-based credible intervals for the difference of two binomial proportions under independent Beta posteriors. The function requires the observed counts $y_1 = 498$, $n_1 = 674$, $y_2 = 316$, $n_2 = 468$, Beta prior parameters $\alpha_1 = \alpha_2 = 1$ (for both priors), the desired credibility level, and the number of simulations:

```
> library(TeachingDemos)
> diffci.bayes(498, 674, 316, 468, 1.0, 1.0, 1.0, 1.0, 0.95, nsim = 1000000)
[1] 0.01014988 0.11761071          # simulated 95% percentile interval for pi1 - pi2
```

As expected, the result is very close to the manually computed percentile interval reported above, with minor differences due to Monte Carlo sampling variability.

R2.6 Bayesian Estimation of State-Specific Election Results Using `brms` *

Section 2.7.2 illustrated empirical Bayes estimation of a set of population proportions by first finding maximum likelihood estimates of parameters of a beta-binomial distribution for the counts. Alternatively, we could use a binomial model with a random effect over states to account for potential overdispersion. We can perform this using the `brms` package that is introduced in Section 2.7.1 and used throughout the following chapters.

*Advanced: with random effects models (see Sections 5.7 and 5.8) and model-fitting diagnostics of Chapter 6.

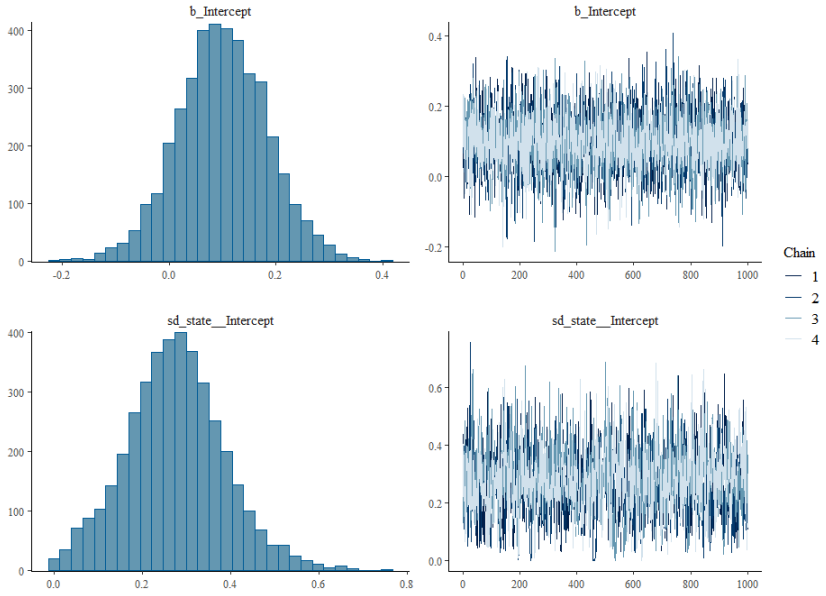

FIGURE R2.5: Histograms of draws from the posterior distributions for the parameters of the model fitted on the Election Poll data, along with corresponding trace plots.

For multi-level models, `brms` follows the syntax of the `R` package `lme4` for linear mixed effects models. In particular, the model formula we need has the form `successes|trials(total) ~ 1 + (1 | group)`, which specifies that we have binomial data (successes out of total trials) and we are modeling the success probability with a random intercept for each group. In our case, the grouping variable is `state`. Thus we fit the `brms` model as shown below:

```
> fit <- brm(y | trials(n) ~  (1|state), data = Election, family = binomial)
> summary(fit)
 Family: binomial
  Links: mu = logit
  Formula: y | trials(n) ~  (1 | state)
   Data: Election (Number of observations: 51)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Multilevel Hyperparameters:
~state (Number of levels: 51)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.27      0.11     0.06     0.51 1.01     1198     1181

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     0.10      0.08    -0.07     0.27 1.00     3916     3052
> plot(fit)     # Figure 2.5
```

The last command produces the histograms of the draws from the posterior distributions for the parameters of the fitted model, along with their trace plots, provided in Figure R2.5.

Instead of parameterizing the beta-binomial distribution in terms of the sample size and the beta hyperparameters α_1 and α_2, as shown in equation (2.13) in Exercise 2.43, `brms`, parametrizes it in terms of the mean probability of trial success $\mu = \alpha\beta$ and the precision parameter $\phi = (1-\mu)\beta$. The variance is then $\sigma^2 = \mu(1-\mu)/(\phi+1)$.

Though in Section 2.6.1 an empirical Bayes estimation was considered, here we adopt non-informative (flat) priors. Since we did not specify priors, the default flat priors were used, which can be obtained by typing

```
> prior_summary(fit)
                prior     class      coef group resp dpar nlpar lb ub
 student_t(3, 0, 2.5) Intercept
    gamma(0.01, 0.01)       phi                                   0
 student_t(3, 0, 2.5)        sd                                   0
 student_t(3, 0, 2.5)        sd           state                   0
 student_t(3, 0, 2.5)        sd Intercept state                   0
```

Summary statistics for the posterior distributions of all parameters are given by:

```
> fit$fit
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

                         mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
b_Intercept              0.10    0.00 0.08   -0.07    0.05    0.10    0.16    0.27  3832 1.00
sd_state__Intercept      0.27    0.00 0.11    0.06    0.20    0.27    0.34    0.51  1199 1.01
Intercept                0.10    0.00 0.08   -0.07    0.05    0.10    0.16    0.27  3832 1.00
r_state[AK,Intercept]   -0.08    0.00 0.30   -0.74   -0.24   -0.05    0.10    0.46  6073 1.00
r_state[AL,Intercept]   -0.23    0.00 0.28   -0.88   -0.39   -0.20   -0.02    0.24  4108 1.00
r_state[AR,Intercept]   -0.08    0.00 0.26   -0.65   -0.23   -0.06    0.07    0.42  6385 1.00
r_state[AZ,Intercept]    0.11    0.00 0.24   -0.33   -0.04    0.09    0.26    0.63  6603 1.00
r_state[CA,Intercept]    0.23    0.00 0.18   -0.08    0.10    0.22    0.35    0.61  3267 1.00
...
r_state[WV,Intercept]   -0.20    0.01 0.31   -0.95   -0.36   -0.16    0.00    0.31  3638 1.00
r_state[WY,Intercept]    0.00    0.00 0.28   -0.57   -0.15    0.00    0.16    0.58  6760 1.00
```

The estimated population proportions (posterior means) of voting for Biden for the states are derived as follows.

```
> posterior_epred_beta_binomial <- function(prep) { # prob. of success per state
+  mu <- brms::get_dpar(prep, "mu")}
> prep <- prepare_predictions(fit)
> ppe <- posterior_epred_beta_binomial(prep)
> post.mean <- colMeans(ppe); post.mean
 [1] 0.5127596 0.4980255 0.5130511 0.5357823 0.5461907 0.5324366 0.5436357 0.5321193
 [9] 0.5273214 0.4882267 0.5246237 0.5321098 0.5087954 0.5410646 0.5350738 0.5027366
[17] 0.5177487 0.5063891 0.5217188 0.5194388 0.5422357 0.5228721 0.5448441 0.5239577
[25] 0.5379401 0.5353174 0.5216972 0.5132093 0.5095107 0.5355278 0.5327991 0.5220982
[33] 0.5489298 0.5166280 0.5239972 0.5149562 0.4979670 0.5376236 0.5148163 0.5272200
[41] 0.5213493 0.5178903 0.5104646 0.5114481 0.5092150 0.5320436 0.5291564 0.5283704
[49] 0.5007476 0.5348193 0.5229450
```

Furthermore, we compute the root mean square of the above posterior means around the population proportions.

```
> Popul.prop <- Election$pi
> sqrt((sum(Election$n*(post.mean - Popul.prop)^2))/1000)
[1] 0.08137412
```

We observe that the quality of fit falls between the maximum likelihood fit (0.097) and the empirical Bayes fit (0.074) computed in Section 2.6.1.

R2.7 Normal Prior for Logit Transformation of a Probability in MCMCpack

In Section 2.7.1 we analyzed the Belief in afterlife example, using a $\mathcal{N}(0, 2^2)$ prior distribution for the logit. We derived in `brms` that the logit $L = \log[\pi/(1-\pi)]$ falls between 1.249 and 1.544 with a posterior probability of 0.95.

Here, we fit a logit model with intercept only and a vague normal prior for the logit using the `MCMCpack` package, resulting, as expected, to a comparable posterior interval:

```
> library(MCMCpack)
> fit.Bayes <- MCMClogit(y~1, mcmc=1000000, b0=0, B0=0) # logit model with intercept only
>               # using Monte Carlo methods; normal prior has mean b0 and variance 1/B0
> summary(fit.Bayes)
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
         Mean             SD       Naive SE Time-series SE # post. distr. of logit has
     1.393e+00      7.484e-02      7.484e-05      2.033e-04 # mean 1.393 and std. dev. 0.075
2. Quantiles for each variable:                          # based on normal prior for logit
 2.5%   25%   50%   75% 97.5%
1.248 1.342 1.392 1.443 1.542       # 95% posterior interval for logit L is (1.248, 1.542)
```

R3

R for Bayesian Inference for Means

R3.1 Revisiting Bayesian Inference for Anorexia Therapy

The R code for deriving Figure 3.2 in Section 3.2.6 is provided below:

```
> d1 <- rnorm(100000,mu0,sigma0)                  # prior
> d2 <- rnorm(100000,mean(y),sd(y)/sqrt(n))       # likelihood
> d3 <- postmean + (sigman / sqrt(n0 + n)) * rt(100000, df) # posterior
# densities:
> pdf1 <- density(d1); pdf2 <- density(d2); pdf3 <- density(d3)
> ylim_combined <- range(pdf1\$y, pdf2\$y, pdf3\$y)  # # plotting limits

> par(mar = c(5, 4, 5, 2) + 0.1)

> plot(pdf1, main = "", xlab = "Mean weight change",
+     ylab = "Density", col = "olivedrab4",
+     lwd = 2.5, xlim = c(-20, 40), ylim = ylim_combined)
> lines(pdf2, col = "red4", lwd = 2.5)
> lines(pdf3, col = "dodgerblue4", lwd = 2.5)
> legend("topright", legend = c("Prior", "Likelihood", "Posterior"),
+       col = c("olivedrab4", "red4", "dodgerblue4"), lwd = 3.75)
```

In the same section, we fitted a regression model on the weight change (after-before) with only an intercept using the `MCMCpack` package to incorporate inverse gamma prior distribution for σ^2. The analysis led to a posterior mean estimate of μ 3.36, and a 95% posterior interval of (0.73, 6.05). We obtain analogous results in `brms` using a normal prior for the mean and an exponential prior for σ:

```
> CB <- data.frame(y)
> library(brm)
> fit.bayes1 <- brm(y ~ 1, data=CB, family=gaussian, prior =
+              prior(normal(10,5.2), class=Intercept) + prior(exponential(0.05), class=sigma))
> summary(fit.bayes1)
Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI
Intercept     3.48      1.36     0.90     6.19  # 95% posterior interval (0.46, 5.96)
Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI
sigma     7.58      1.09     5.78    10.03
```

R3.2 Alternative Bayesian Implementation: Mean Housework Hours for Men and Women

For this household work example, in Section 3.5.3, we used an exponential prior distribution for σ and a $\mathcal{N}(\mu_0, \sigma_0^2)$ prior distribution for each of μ_1 and μ_2 with μ_0 = 10 and σ_0 = 4,

fitting a model using the `brm` function in the `brms` package. In Section A.3.2 of the book's R-Appendix, we provided an alternative fit of a model in `brms` using the same prior for the means but a different prior for σ. In particular, we used for σ and inverse inverse gamma prior, and saved the fitted model under `fit.bayes1`. We further generated an *ecdf* overlay plot using the `pp_check` function of `brms` (based on the `bayesplot` package), provided in Figure A.3.1. We mentioned there that the `pp_check` function cannot directly display posterior predictive checks separately by group (e.g., by gender). However, this can be done manually by subsetting the data (i.e., selecting only the rows of the data set corresponding to a specific group) and using the `newdata` argument, as shown below:

```
> household_men <- Household %>% filter(gender == "0")
 > household_women <- Household %>% filter(gender == "1")
 > pp_check(fit.bayes1, type = "ecdf_overlay", ndraws = 1000,
         newdata = household_men)
 > pp_check(fit.bayes1, type = "ecdf_overlay", ndraws = 1000,
         newdata = household_women)
```

We observe next that the results are very similar to those obtained using the *default priors* of `brms`, which are employed whenever priors are not specified:

```
> fit.bayes2 <- brm(time ~  gender, data = Household, family = gaussian(),
                    warmup = 2000, iter = 5000)
> summary(fit.bayes2)
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: time ~ -1 + gender
   Data: Household (Number of observations: 1276)
  Draws: 4 chains, each with iter = 5000; warmup = 2000; thin = 1;
         total post-warmup draws = 12000
Regression Coefficients:
        Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
gender0     8.31      0.47     7.39     9.23 1.00    11936     8747
gender1    11.88      0.43    11.03    12.73 1.00    12749     8564
Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    11.36      0.23    10.93    11.82 1.00    12167     8657
```

We can see the default priors used, by the following command and verify that they are non-informative and that σ has a lower bound (lb) at 0:

```
> prior_summary(fit.bayes2)
    prior class    coef group resp dpar nlpar lb ub       source
    (flat)     b                                          default
    (flat)     b gender0                                (vectorized)
    (flat)     b gender1                                (vectorized)
 student_t(3, 0, 5.9) sigma                   0         default
```

The output above shows that `brms` uses largely non-informative (flat) priors for the regression coefficients, together with a weakly informative prior for the residual standard deviation. In particular, the entries labeled `(flat)` for class `b` indicate that the regression coefficients (including the intercept and the effects associated with the levels of `gender`) are assigned improper uniform priors over the real line. These priors do not favor any particular values and thus contribute no prior information to the estimation.

In contrast, the residual standard deviation parameter `sigma` is assigned a Student-t prior with 3 degrees of freedom, location 0, and scale 5.9. This prior is weakly informative: it is centered at zero but has relatively heavy tails, allowing for a wide range of plausible values while still providing some regularization to stabilize estimation. Since `sigma` must be positive, this prior is implicitly truncated to the positive real line.

Here, we use the `R` package `MCMCpack` as well, which can specify an inverse gamma

prior distribution (3.9) for the common variance σ^2, corresponding to a Bayesian analysis with conjugate prior distributions. Since our main focus is on μ_1 and μ_2 and the difference between them, with σ^2 being a nuisance parameter, we take a highly disperse inverse gamma prior distribution for it. The improper prior $f(\sigma^2) \propto 1/\sigma^2$ is approximated by taking the shape parameter α and scale parameter β in the inverse gamma distribution to be close to 0, such as 0.001 each.[1]

```
> library(MCMCpack)
> fit.bayes3 <- MCMCregress(time~ -1+factor(gender), mcmc=5000000, b0=10, B0=1/16, c0=0.002,
+  d0=0.002, data=Household) # b0=prior mean, B0=precision, c0/2=shape, d0/2=scale
> summary(fit.bayes3)
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
                    Mean     SD
factor(gender)0   8.333 0.4680    # men
factor(gender)1  11.853 0.4291    # women
sigma2          129.279 5.1336    # variance
2. Quantiles for each variable:
                     2.5%     25%     50%     75%  97.5%
factor(gender)0   7.415   8.017   8.332   8.648   9.25   # men
factor(gender)1  11.012  11.564  11.853  12.143  12.70   # women
sigma2          119.598 125.750 129.143 132.663 139.72   # variance
```

R3.3 Bayes Factor for Comparing Two Means

In Section 3.5.4, we discussed Bayesian results based on highly dispersed or improper prior distributions for the parameters. In this setting, the posterior tail probability for $\mu_1 \leq \mu_2$ closely resembles the frequentist one-sided P-value for testing $H_0: \mu_1 = \mu_2$ against $H_1: \mu_1 > \mu_2$. Consequently, both the frequentist and Bayesian analyses lead to rejection of H_0 in favor of H_1.

For a detailed discussion on Bayes Factor for comparing two means, we refer to Section A.5.1 of the R-Web-Appendix of Agresti and Kateri (2022) at `https://stat4ds.rwth-aachen.de/pdf/DS_R_webAppendix`. A Bayesian t-test can be performed in the `BayesFactor` package applying the `ttest.tstat` function that requires only the sample sizes n_1 and n_2 along with the value of the t-test statistic. Alternatively, the more flexible `ttestBF` function can be used. It places a Cauchy prior on the standardized effect size $\delta = (\mu_1 - \mu_2)/\sigma$ and allows sampling from the corresponding posterior distribution. This, in turn, provides additional possibilities, such as constructing credible intervals or producing simulated posterior density plots.

Another useful option is the `BEST` package, which also relies on posterior simulation and provides highest posterior density (HPD) intervals for means, standard deviations, and their differences, as well as for the effect size.

All of these approaches are illustrated in Section A.5.1 of `DS_R_webAppendix.pdf` for the anorexia study example.

Furthermore, in the `Bolstad` package we can perform one and two sample Bayesian t-tests on vectors of data. We illustrate for the data vectors `y1` (women) and `y2` (men), as derived in Section 3.5.3 for the gender specific mean housework hours example, by comparing the corresponding means using Jeffreys priors, as shown below:

[1] The coding fits a linear model without an intercept, by putting −1 in the intercept position, to estimate the two means.

```
> library(Bolstad)
> t.Bayes <- bayes.t.test(y1,y2, prior='jeffreys')      # default prior choice
> t.Bayes
        Two Sample t-test
data:  y1 and y2
t = 5.5833, df = 1274, p-value = 2.88e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:           # compare to 95% post.interval of Section 3.4.3
  2.312596 4.818127                       # for improper priors: (2.312074, 4.818035)
sample estimates:
 posterior mean of x posterior mean of y
           11.874640            8.309278
```

R3.4 Comparing Means of Population Having Unequal Variances

In Section A.3.5 of the book's R Web-Appendix we obtained in `brms` a Bayesian comparison of drivers reaction times according to cell phone use, without assuming equal variances. For this, we used the `brmsformula` function `bf` to model the mean and standard deviation of the driver reaction times, with vague prior distributions, discussed there. The model fit was saved under `fit_uneqvar`.

We can derive the 95% credible interval for the difference $\mu_1 - \mu_2$ using the corresponding quantiles of the difference of the generated chains of posterior values for μ_1 and μ_2, as well as further summary statistics:

```
> mu1 <- as.data.frame(fit_uneqvar)$b_group1 # chain of the generated posterior
> mu2 <- as.data.frame(fit_uneqvar)$b_group2 #   dist's for mu1 and mu2
> quantile(mu1-mu2, c(0.025, 0.975))
#     2.5%    97.5%
# 11.52009 90.71371
> mean(mu1-mu2); sd(mu1-mu2)
# [1] 51.4397   [1] 20.07686
> summary(mu1-mu2)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  -26.41   38.25   51.65   51.44   64.83  137.91
```

Notice that the 95% posterior interval for $\mu_1 - \mu_2$ is somewhat different than the respective one in Exercise 3.10, which is (11.62, 91.36), when setting `set.seed(73)`. This difference is due to the use of different priors for the means. Exercise 3.10 we use improper priors for all parameters, as shown below:

```
> m1 <- 585.2; s1 <- 89.6; n1 <- 32 # summary stats for cell phone group
> m2 <- 533.7; s2 <- 65.3; n2 <= 32 # summary stats for control group
> set.seed(73)
> z1 <- rnorm(n1); y1 <- scale(z1)*s1 + m1 # scale fn standardizes values of z1
# multiplying standardized values by s1 and adding m1 gives mean m1, std. dev. s1
> z2 <- rnorm(n2); y2 <- scale(z2)*s2 + m2 # sample of size n2 with mean m2,
                                           # std. dev. s2
> sum1 <- (n1 - 1)*s1^2; sum2 <- (n2 - 1)*s2^2 # sums of squares in the two groups
> rsigma1 <- sum1/rchisq(20000000, n1-1) # simulate posterior variances from
> rsigma2 <- sum2/rchisq(20000000, n2-1) # scaled inverse chi-squared distributions
> mu1 <- rnorm(20000000, mean = mean(y1), sd = sqrt(rsigma1)/sqrt(n1)) # simulate
> mu2 <- rnorm(20000000, mean=mean(y2), sd=  sqrt(rsigma2)/sqrt(n2)) # posterior means
> quantile(mu1 - mu2, c(0.025, 0.975))
    2.5%    97.5%
11.62711 91.36701
```

R3.5 Comparing Mean Holding Times on Phone for Service in `MCMCpack`

In Section 3.6.3 we analyzed a randomized experiment of small size ($n = 15$) of an airline on whether subjects, when calling for making reservations, would remain on hold longer, on the average, if they heard (a) an advertisement about the airline and its current promotions (group A), (b) Muzak (simple instrumental versions of popular music, group M), or (c) classical music (Vivaldi's *Four Seasons*, group C). The data were analyzed by the frequentist approach (anova) with the associated P-value providing strong evidence that at least one of the population means differs from the others.

In a Bayesian fitting we modeled a quantitative response variable in terms of groups that are levels of a factor (categorical explanatory variable), using the `brms` package. Here, we fit the same model in `MCMCpack` using an inverse gamma distribution for the within-groups variance parameter σ^2 with mean 50 and standard deviation 50, and that distribution[2] has $\alpha = 3$ and $\beta = 100$.

```
> fit.bayes <- MCMCregress(time ~ -1 + factor(group), mcmc=5000000, b0=10, B0=1/25,
+                          c0=6, d0=200, data=Results)
> summary(fit.bayes)
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
                 Mean    SD  Naive SE Time-series SE
factor(group)A  6.048 1.889 0.0008446      0.0008556
factor(group)B  3.817 1.903 0.0008509      0.0008768
factor(group)C 11.718 1.879 0.0008403      0.0008436
sigma2         20.912 7.846 0.0035087      0.0040606
2. Quantiles for each variable:
                  2.5%    25%    50%    75%  97.5%
factor(group)A  2.3957  4.794  6.021  7.271  9.858
factor(group)B  0.1837  2.545  3.774  5.041  7.699
factor(group)C  7.9712 10.491 11.729 12.959 15.397
sigma2         10.6438 15.511 19.322 24.475 40.491
```

Notice that the `bayes.anova` function in the `bayesanova` package does Bayesian estimation of three or more means without assuming equal variances.

R3.5.1 A Model with Intercept for Multiple Means*

We now revisit Example 3.6.3 on comparing mean holding times on phone for service. In that section, to estimate directly the means instead of using the first group as a reference and estimating the difference of means between the other groups and that group, we fitted a model without an intercept. Here, we shall fit the same model on the data frame `Results` of Example 3.6.3 but with a different parameterization, including an intercept in the model and using the *sum-to-zero constraint* for the parameters of the factor `group`. One of the parameters corresponding to the levels of a factor is redundant, which means that for identifiability purposes a constraint has to be imposed on the factor levels' parameters. In the usual parameterization of factors in software (and `brms`), for a factor X with K levels, one of its levels is set to be the reference category (commonly the first one) and its parameter is set equal to 0. Thus, the vector of parameters that corresponds to X, is denoted by $\boldsymbol{\beta}_1 = (\beta_{11}, \beta_{12}, \ldots, \beta_{1K})'$ while if the first level of this factor is the reference category then $\beta_{11} = 0$. Under this parameterization, the intercept corresponds to the expected value of the response for the reference category, and the remaining factor parameters express

[2] Based on the formulas in a footnote in Section 3.2.3, p. 93.

the deviation of the expected value of the corresponding level to the reference category. If the sum-to-zero constraint is employed (i.e., instead of setting $\beta_{11} = 0$ for a factor of K levels, we set $\sum_{j=1}^{K} \beta_{1j} = 0$), the intercept corresponds to the overall mean, while the factor parameters refer to the expected deviation of each group mean from the overall mean. Notice that in this case $K - 1$ estimates are provided in the output, since due to the constraint, $\beta_J = -\sum_{j=1}^{K-1} \beta_{1j}$. In our example there are three groups present ($K = 3$). In the following code, we use a $\mathcal{N}(8, 1.658^2)$ prior distribution for the intercept and a $\mathcal{N}(0, 2.5^2)$ prior distribution for every factor level, corresponding to an a priori assumption of no differences among the groups means and to standard deviation of about 9 ($= \sqrt{(1.658^2 + 2.5^2)}$) for the mean priors of the first two groups, A and M, and of about 3.9 ($= \sqrt{(1.658^2 + 2 * 2.5^2)}$) for the mean prior of the third one, C, values comparable to the priors in Section 3.6.3.

```
> contrasts(Results$grp) <- "contr.sum"    # sum to zero constraint for group parameters
> fit.bayes <- brm(time ~ grp, data=Results, family=gaussian,
+            prior = prior(normal(8, 1.658), class=Intercept) + prior(normal(0, 2.5), class=b) +
+            prior(exponential(0.10), class=sigma))
> summary(fit.bayes)
Regression Coefficients:
          Estimate  Est.Error    l-95% CI u-95% CI
Intercept     7.05       0.85     5.43     8.82
grp1         -1.40       1.14    -3.64     0.84
grp2         -3.18       1.13    -5.31    -0.85

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI
sigma   3.65      0.81     2.46    5.55
```

The posterior estimated means for groups 1 and 2 are $7.05-1.40 = 5.65$ and $7.05-3.18 = 3.87$. The sum-to-zero constraint implies that the deviation of group3 from the overall mean (intercept) is equal to minus the sum of the parameters for group1 and group2. That is, the estimated posterior mean for group3 equals $7.05 + 1.40 + 3.18 = 11.63$.

However, for the estimation of the standard deviation and the credible interval for the deviation of group3 from the overall mean we need to draw from the posteriors.

```
> # 4000 draws from the posterior (after burn in):
> gr1.post <- as.data.frame(fit.bayes)$b_group1
> gr2.post <- as.data.frame(fit.bayes)$b_group2
> gr3.post <- -(gr1.post+gr2.post) # draws from the post. for additional effect of group3
> mean(gr3.post)
[1] 5.207173
> sd(gr3.post)
[1] 1.384074
> quantile(gr3.post, c(0.025, 0.975))  # 95% posterior CI
    2.5%    97.5%
 2.433295 8.054576
```

The overall mean is estimated by the posterior mean 6.76, while the posterior means for groups A, B and C are estimated by 6.76-1.37, 6.76-3.84 and 6.76+1.37+3.84.

For estimating standard deviations and deriving posterior intervals directly for the three groups and their differences, we need to draw also from the posterior of the intercept and synthesize samples from the posteriors:

```
> gr1.post <- as_draws_df(fit.bayes)$b_group1     # 4000 draws from the posterior
> gr2.post <- as_draws_df(fit.bayes)$b_group2
> gr3.post <- - (gr1.post+gr2.post)   # draws from posterior for effect of group3
> interc.post <- as_draws_df(fit.bayes)$b_Intercept
> A <- interc.post + gr1.post     # group 1 = A = advertisement
> M <- interc.post + gr2.post     # group 2 = M = Muzak
> C <- interc.post + gr3.post     # group 3 = C = Classical
```

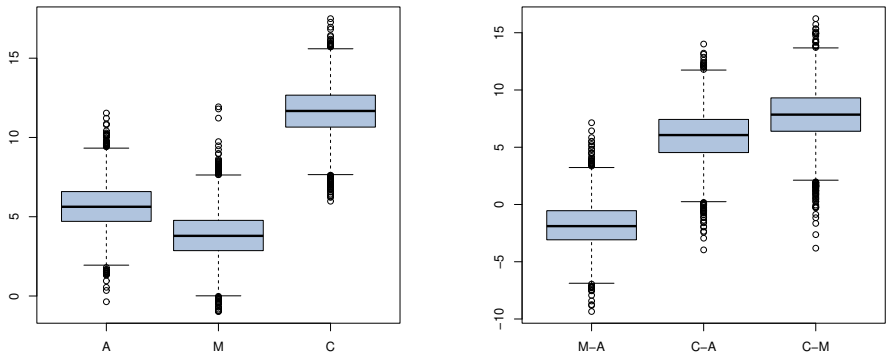

FIGURE R3.1: Box plots of draws from the posterior distributions of groups A, M, C (left) and their pairwise differences (right) for Example 3.6.3.

```
> post.groups <- rbind(A, M, C) # 3x4000 matrix (draws from posteriors of groups)
> library(matrixStats)   # to derive means, standard deviations and percentiles per row
> rowMeans(post.groups)
        A         M         C
  5.655583  3.840870 11.628584
> rowSds(post.groups)
      A        M        C
 1.425017 1.487112 1.564897
> CI.95 <- rowQuantiles(post.groups, probs=c(0.025, 0.975)); CI.95
       2.5%     97.5%
A 2.8963870  8.471675
M 0.9527152  6.959334
C 8.4245579 14.662446
> MA <- M - A;  CA <- C - A; CM <- C - M
> post.dif <- rbind(MA, CA, CM) # 3x4000 matrix (draws from posteriors of M-A, C-A, C-M)
> difCI.95 <- rowQuantiles(post.dif, probs=c(0.025,0.975))   # 95% CI for differences
> difCI.95
        2.5%     97.5%
MA -5.628688  2.171685
CA  1.376724 10.149384
CM  2.978873 11.955504
```

We can summarize graphically draws from the posterior distributions of the means for groups A, M, and C (and their differences) by histograms or smoothed posteriors. Here, we show the box-plots of the draws from the posteriors of groups A, M, C, and Figure R3.1 shows their pairwise differences, to resemble how multiple means differences are visualized in frequentist statistics. The R code for deriving the box-plots is provided below.

```
> group <- c(rep(1,4000),rep(2,4000),rep(3,4000))
> post.groups1 <- cbind(c(A,M,C),group)
> label=c("A","M","C")
> boxplot(post.groups1[,1]~post.groups1[,2], col="lightsteelblue", names=label,
+      xlab="", ylab="")
> post.dif1 <- cbind(c(MA,CA,CM),group)
> label1=c("M-A", "C-A", "C-M")
> boxplot(post.dif1[,1]~post.dif1[,2],, col="lightsteelblue", names=label1, xlab="", ylab="")
```

In case we fit the Bayesian model for multiple means with intercept but without the sum

to zero constraint (i.e., using group A as the reference category), the procedure described above applies analogously, with the posterior sample from the intercept being the posterior sample for group A, while the posterior samples for groups M or C are the sums of the posterior samples from the intercept and from group2 or group3 (i.e., deviations from A).

R3.5.2 Bayesian Comparison of Multiple Means with `stanarm`*

The `stanarm` package has the interesting option of helping with the specification of non-informative prior distributions. We shall illustrate for Example 3.6.3 of comparing three means.

```
> library(rstanarm)
> default_prior_test <- stan_glm(time ~ -1 + group, data = Results, chains = 1)
> prior_summary(default_prior_test)
Priors for model 'default_prior_test'
------
Coefficients
  Specified prior:
    ~ normal(location = [0,0,0], scale = [2.5,2.5,2.5])
  Adjusted prior:
    ~ normal(location = [0,0,0], scale = [26.01,26.01,26.01])

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.2)
------
See help('prior_summary.stanreg') for more details
```

With this information we proceed with fitting the model for multiple means in `stanarm`:

```
> fit.bayes <- stan_glm(time ~ -1 + group, data=Results, iter=2000, warmup=500, chains=4,
+      thin=2, refresh=0, prior_intercept=normal(6.7,13), prior=normal(0,26.01),
+      prior_aux=exponential(0.2))
> summary(fit.bayes)
Estimates:
         mean   sd   10%   50%   90%
groupA  5.3    1.6  3.3   5.3   7.3
groupB  2.8    1.7  0.8   2.8   4.9
groupC 12.0    1.6  9.9  12.0  14.0
sigma   3.6    0.8  2.7   3.5   4.7

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 6.7    1.4  5.1   6.7   8.4

MCMC diagnostics
              mcse Rhat n_eff
groupA        0.0  1.0  2611
groupB        0.0  1.0  2332
groupC        0.0  1.0  2771
sigma         0.0  1.0  2117
mean_PPD      0.0  1.0  2787
log-posterior 0.0  1.0  1610
```

The median and their standard deviations of the posterior distributions are provided by

```
print(fit.bayes)
```

Convergence diagnostics, plots and posterior multiple comparisons can be carried out with functions of `coda` and `bayesplot`, as illustrated in the next section.

R3.5.3 Bayesian Comparison of Multiple Means with `RStan`*

We can also employ `RStan` to implement a Bayesian comparison of multiple means, which we illustrate by revisiting Example 3.6.3. First we specify the model in a `modelString` and write it to a stan file ("anovaModel.stan"). The components needed are `data`, `parameters`, `transformed parameters`, `model` (containing the specification of the likelihood and the priors), and `generated quantities`.

```
> setwd("C:/... / ... /BayesANOVA")  # set the directory to read/save files
> modelString = "
   data {
   int<lower=1> n;
   int<lower=1> nX;
   vector [n] y;
   matrix [n,nX] X;
   }
   parameters {
    vector[nX] beta;
   real<lower=0> sigma;
   }
   transformed parameters {
   vector[n] mu;

   mu = X*beta;
   }
   model {
   //Likelihood
   y~normal(mu,sigma);

   //Priors
   beta ~ normal(0,1000);
   sigma~cauchy(0,5);
   }
   generated quantities {
   vector[n] log_lik;

   for (i in 1:n) {
   log_lik[i] = normal_lpdf(y[i] | mu[i], sigma);
   }
   }

   "
> writeLines(modelString, con = "anovaModel.stan")  # write model to a stan file
```

Next we read the data and transform them in a `list`, as required by Stan:

```
> time <- c(5, 1, 11, 2, 8, 0, 1, 4, 6, 3, 13, 9, 8, 15, 15)
> group <- c(rep("A", 5), rep("B", 5), rep("C", 5))
> group <- factor(group)  # the variable of levels needs to be a factor
> Results <- data.frame(time, group)
>  # Exploratory data analysis:
> boxplot(time ~ group, Results)           # see Fig.
> write.csv(Results, "ResultsAnova.csv")   # saves the data in an excel file
> Xmat <- model.matrix(~group, Results)
> data.list <- with(Results, list(y = time, X = Xmat, nX = ncol(Xmat), n = nrow(Results)))
```

In the sequel, the parameters to monitor and the Markov chain parameters are specified.

```
> params <- c("beta", "sigma", "log_lik")
> nChains = 2             # number of chains
> burnInSteps = 500       # burn in period
> thinSteps = 1
> numSavedSteps = 2000    #  saved replications across all chains
```

```
> nIter = ceiling(burnInSteps + (numSavedSteps * thinSteps)/nChains)
> nIter     # number of iterations per chain
[1] 1500    # 1000 + 500 (burn in)
```

Finally, we run the model:

```
> data.rstan <- stan(data = data.list, file = "anovaModel.stan", chains = nChains,
                     pars = params, iter = nIter, warmup = burnInSteps, thin = thinSteps)
> print(data.rstan, par = c("beta", "sigma"))  # print results for the posteriors of parmeters
Inference for Stan model: anon_model.
2 chains, each with iter=1500; warmup=500; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=2000.

         mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
beta[1]  5.45    0.06 1.63  2.35  4.42  5.38  6.46  8.75   659    1
beta[2] -2.65    0.09 2.35 -7.36 -4.21 -2.65 -1.14  1.90   749    1
beta[3]  6.59    0.08 2.32  1.91  5.15  6.66  8.09 10.87   795    1
sigma    3.64    0.02 0.78  2.44  3.09  3.53  4.04  5.46  1008    1

> pairs(data.rstan, pars = c("beta", "sigma"))  # posterior plots (not shown)
```

For each parameter, `n_eff` is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on split chains (at convergence, Rhat=1).

The next stage is to proceed to diagnostics:

```
library(mcmcplots)
s = as.array(data.rstan)
mcmc <- do.call(mcmc.list, plyr:::alply(s[, , -(length(s[1, 1, ]))], 2, as.mcmc))
denplot(mcmc, parms = c("beta","sigma"))   # plot of the posterior densities for the two chains
traplot(mcmc, parms = c("beta","sigma"))   # trace plots (not shown)
stan_ac(data.rstan, pars = c("beta"))      # autocorrelation plots (not shown)
stan_ess(data.rstan, pars = c("beta"))     # effective sample size
```

Results are better visualized graphically using packages `tidyr`, `bayesplot` and `multcomp` for multiple comparisons.

```
> install.packages("tidyr")
> library(tidyr)
> mcmc = as.data.frame(data.rstan) %>% dplyr:::select(contains("beta"),sigma) %>% as.matrix
> coefs = mcmc[, 1:3]      # post. median parameter estimates
> fit = coefs %*% t(Xmat)  # Xmat: model matrix from above
> # draw samples from this model:
> yRep = sapply(1:nrow(mcmc), function(i) rnorm(nrow(Results), fit[i,], mcmc[i, "sigma"]))
> newdata = data.frame(x = Results$group, yRep) %>% gather(key = Sample, value = Value, -x)
> ggplot(newdata) + geom_violin(aes(y = Value, x = x, fill = "Model"),
    alpha = 0.5) + geom_violin(data = Results, aes(y = time, x = group,
    fill = "Obs"), alpha = 0.5) + geom_point(data = Results, aes(y = time,
    x = group), position = position_jitter(width = 0.1, height = 0),
    color = "black") + theme_classic()
> # posterior ETP credible intervals:
> library(bayesplot)
> mcmc_intervals(as.matrix(data.rstan), regex_pars = "beta|sigma")

> mcmc = data.rstan
> betasigma <- as.matrix(mcmc)[, 1:4]
> mcmc_areas(betasigma)     # plot of parameters' posteriors
> coefs <- as.matrix(mcmc)[, 1:3]
v> newdata <- data.frame(x = levels(Results\$group))
> library(multcomp)  # for Tukey's multiple comparisons
> tuk.mat <- contrMat(n = table(newdata\$x), type = "Tukey") # table(): No. of replic./level
> Xmat <- model.matrix(~x, data = newdata)
> pairwise.mat <- tuk.mat %*% Xmat
> pairwise.mat
> mcmc_areas(coefs %*% t(pairwise.mat))
```

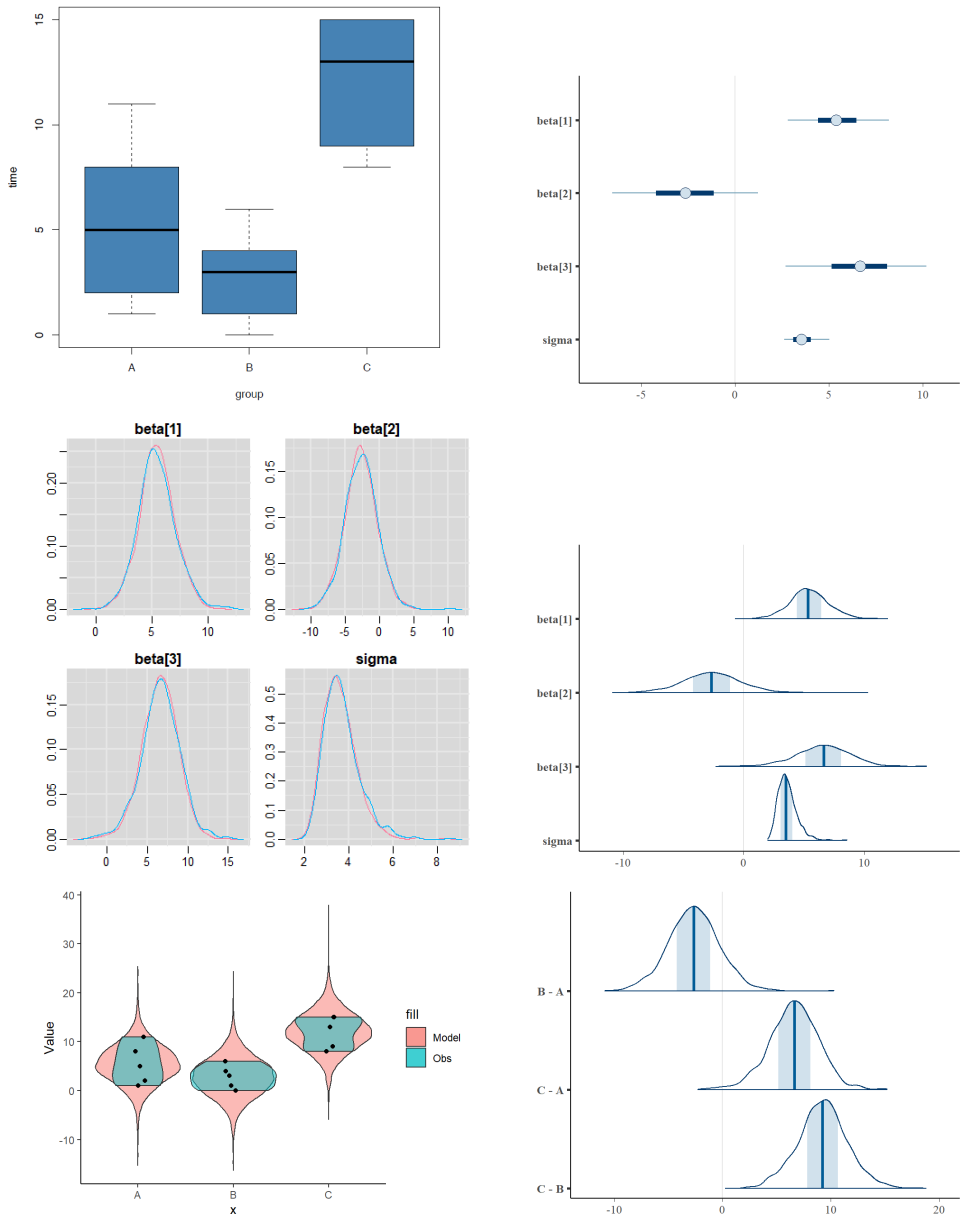

FIGURE R3.2: Diagnostic and plots of posterior distributions for Example in Section 3.6.3.

R4

Bayesian Linear Models in R

R4.1 The Scottish Hill Races Example in `rstanarm`*

Section 4.3.4 fitted a linear model in `brms` to the `ScotsRaces` dataset from the book's website. Here, we analyze this data using the `bayesglm` function in the `rstanarm` package, which uses t distribution priors. It provides normal priors by taking $df = \infty$ for the t distribution. That prior becomes flat when the prior scale parameter is infinite. For model fitting, rather than employing Gibbs sampling or the Metropolis–Hastings algorithm, it provides a very fast calculation that approximates a posterior mode and SE by incorporating an EM algorithm into the iteratively reweighted least squares algorithm[1].

```
> library(rstanarm)
> fit.bayes <- bayesglm(time ~ climb + distance + climb:distance,
+         family=gaussian, prior.mean=0, prior.scale=Inf, prior.df=Inf)
> summary(fit.bayes)
Coefficients:
               Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)     -0.7672      3.6758   -0.209    0.8359
climb            3.7133      2.2254    1.669    0.1041
distance         4.9623      0.4463   11.118  4.93e-13
climb:distance   0.6598      0.1640    4.023    0.0003
(Dispersion parameter for gaussian family taken to be 47.6967)
------------------------------------------------------------------------------
```

The Bayesian estimates, with such a vagiue prior, are essentially the same as the least squares estimates. Interpretations can use Bayesian posterior probability statements. For example, the probability is 0.95 that the interaction parameter falls between 0.30 and 1.01.

R4.2 A Simulation to Illustrate Collinearity

In Section 4.3.5, we discussed the effects of collinearlity in linear models. Here, we illustrate collinearity with a simple simulation. Suppose that Y is a continuous variable that is a measure of a person's intelligence, but it is unobservable. A variable that is unobservable but is assumed to underlie what we *can* measure is referred to as a *latent variable*.

In the population of interest, suppose that Y has a $\mathcal{N}(100, 16^2)$ distribution. To approximate it, suppose there are four versions (X_1, X_2, X_3, X_4) of an IQ test, where, conditional on $Y = y$, each X_j has a $\mathcal{N}(y, 2^2)$ distribution. If we could observe y, we now analyze how well we can estimate the effect of x_1 as a predictor of y, when it is the sole predictor and when the model also contains (x_2, x_3, x_4). Here we show least squares results for 100

[1] for details, see in https://www.unt.edu/rss/class/Jon/Benchmarks/BayesGLM_JDS_Jan2011.*pdf*

randomly generated observations, but similar results occur with Bayesian analyses using relatively disperse prior distributions:

```
> y <- rnorm(100, 100, 16)
> x1 <- rnorm(100,y,2); x2 <- rnorm(100,y,2); x3 <- rnorm(100,y,2); x4 <- rnorm(100,y,2)
> summary(lm(y ~ x1))
            Estimate Std. Error
(Intercept)  1.29092    1.24799
x1           0.98882    0.01231
Multiple R-squared:  0.985

> cor(cbind(y, x1, x2, x3, x4))
           y        x1        x2        x3        x4
y  1.0000000 0.9918715 0.9911635 0.9903682 0.9932046
x1 0.9918715 1.0000000 0.9830122 0.9824558 0.9843187
x2 0.9911635 0.9830122 1.0000000 0.9811606 0.9828595
x3 0.9903682 0.9824558 0.9811606 1.0000000 0.9816055
x4 0.9932046 0.9843187 0.9828595 0.9816055 1.0000000

> summary(lm(y ~ x1 + x2 + x3 + x4))
            Estimate Std. Error
(Intercept)  1.63086    0.68233
x1           0.30247    0.04655
x2           0.20023    0.04390
x3           0.33455    0.04479
x4           0.14772    0.04154
Multiple R-squared:  0.9957
```

The correlation is greater than 0.99 between y and each x_j, and the linear dependence of each x_j on the latent variable y induces very strong correlations among $\{x_j\}$. The model with x_1 alone as an explanatory variable gives a very precise approximation for y, with $R^2 = 0.985$. With all four explanatory variables in the model, its effect weakens quite dramatically, from 0.99 to 0.30, as it is now a conditional effect. Because the explanatory variables are so highly inter-correlated, its standard error nearly quadruples. We can predict y nearly as well with the simpler model ($R^2 = 0.985$, compared to $R^2 = 0.996$ with all four predictors), and its effect is simpler to interpret.

This example with such strong correlations is rather unrealistic, but collinearity can have a substantial impact in more common situations. Exercise 4.16 shows a real-data example.

R4.3 Posterior Predicted Responses and Residuals

For the impairment example in Section A.4.2 of the book's R-Appendix we demonstrated how to plot the posterior distributions for Bayesian models' parameters, using the packages `bayesplot` and `ggplot2` for a linear model fitted in `brms` and saved under `fit.bayes`. Next we show how to visualize *posterior predictive responses* and residuals based on posterior draws.

```
> yrep <- posterior_predict(fit.bayes)  # draws of post. predict. responses (dim:4000x40)
> y <- Mental$impair                    # observed responses
> ppc_error_hist(y, yrep[1:3, ])        # histogram of errors for the first three draws
> ppc_error_scatter(y, yrep[100:105, ]) # scatterplots of y vs resid. for draws 100-105
> ppc_error_scatter_avg(y, yrep)        # scatterplot of y vs mean resid. over all draws
> x <- Mental$life
> ppc_error_scatter_avg_vs_x(y, yrep, x) # scatterplot of mean residuals vs life events
```

The draws in `yrep` consists of s draws of posterior samples of the same size n as the data

sample. The number of draws s equals the number of iterations used in fitting the model (in our case, 4000). For the scatterplots in `ppc_error_scatter()`, not shown here, the residuals are calculated based on a single draw from the posterior while in `ppc_error_scatter_avg()` and `ppc_error_scatter_avg_x()`, given in Figure R4.1, the individual residuals of all posterior draws are averaged.

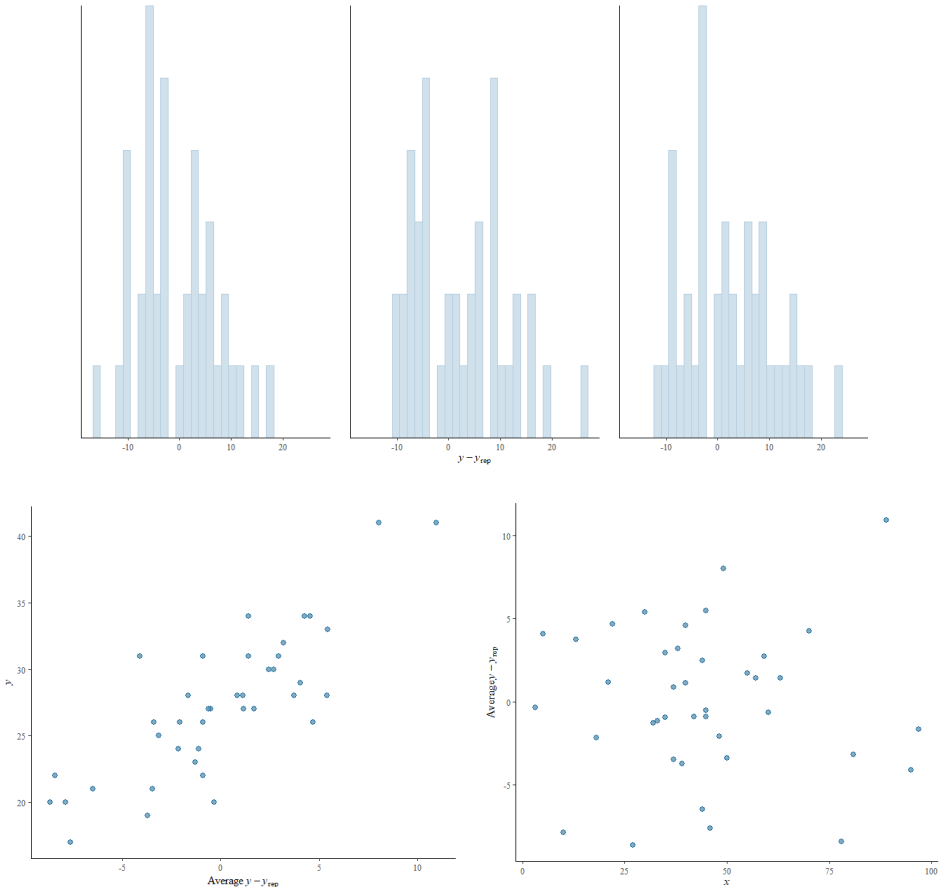

FIGURE R4.1: Histograms of residuals for the first three draws of posterior predicted responses and scatterplots of observed responses (impair) vs mean residual values over all posterior draws (left) and mean residuals vs the explanatory variable life (x) on the right.

R4.4 Bayesian Lasso in `R`

In its current version, `brms` does not support the lasso prior and proposes as shrinkage priors the horseshoe prior [2], which is symmetric around zero with an infinitely large spike at zero and fat tails. It corresponds to student-t prior imposed on the local shrinkage parameters

[2] Carvalho, Polson and Scott: Handling sparsity via the horseshoe. In *Artificial Intelligence and Statistics*, PMLR, p. 73-80 (2009).

with 1 degree of freedom. However, such a prior quite often leads to convergence problems and a high number of divergent transition in Stan. Increasing the degrees of freedom may solve the problem, however the prior does not resemble a horseshoe anymore.

In Section A.4.4 of the book's R Web-Appendix, we implemented, Bayesian Lasso in`rstanarm`. Notice that `rstan` (and consequently `rstanarm`) use a different reparameterization for the *pdf* of the Laplace distribution than the one we use in this book:

$$f(y|\mu,\lambda) = \frac{1}{2\lambda}\exp\left(-\frac{|y-\mu|}{\lambda}\right) ,$$

where μ and λ are the location and scale parameters, respectively. This corresponds to our notation for $\mu = 0$ and $\tau = 1\lambda$. Furthermore, if `autoscale=TRUE`, then the Laplace prior for the parameter β_j, corresponding to the regressor vector $\mathbf{x}_j$, $j = 1,\ldots,p$, is Laplace$(0,\lambda_j)$ with scale parameter $\lambda_j = \lambda sd(\mathbf{x}_j)$.

The code for deriving Figure 4.6, follows:

```
> library(glmnet)
> attach(Students)
> x <- cbind(gender, age, hsgpa, abor, dhome, dres, tv, sport, news, aids, veg, ideol,
+            relig, affirm)
> fit.lasso <- glmnet(x, cogpa, alpha=1, family="gaussian")
# set colors:
> custom_colors <- c("coral1", "goldenrod2", "chartreuse4", "darkturquoise","hotpink1",
+                    "sienna", "lightskyblue3", "gold1", "dodgerblue", "royalblue4",
+                    "lightpink", "magenta2", "red2","mediumpurple1")
> matplot(log(fit.lasso$lambda), t(fit.lasso$beta), type = "l", lty = 1, lwd=2,
+         col = custom_colors, xlab = expression(log(lambda)), ylab = "Coefficients")
# Add custom legend to indicate which variable corresponds to which color:
legend("bottomright", legend = colnames(x), col = custom_colors, lty = 1,
+      cex = 0.65, lwd=2, bty = "n")
```

R5

Bayesian Generalized Linear Models in **R**

R5.1 Another Gamma Regression Model

Section 5.2.2 presented various normal and gamma GLMs for the house selling price data. For the gamma model, we used the log link function, and BIC suggested that that model fitted better than the normal model. An alternative model, even better according to BIC, assumes a gamma-distributed response variable but models the mean directly (rather than its log) and removes the intercept term but includes an interaction term. Doing this has the appealing result of forcing the estimated selling price mean to be 0 at size = 0 both for new and older homes and still results in a higher estimated mean selling price for new homes at all size levels. Moreover, this formulation is more parsimonious than the models considered in Section 5.2.2 in the sense that it uses fewer free parameters. However, unlike other models presented, which follow the principle that main effects are included whenever their corresponding interaction terms are present, this model omits the main effect of new while retaining the interaction term size × new:

```
> fit.gamma2 <- brm(price ~ -1 + size + size:new, family=Gamma(link=identity), data=Houses,
          + prior = prior(normal(0,100), class=b) + prior(exponential(0.01), class=shape))

> summary(fit.gamma2)
Regression Coefficients:
         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
size         0.13      0.00     0.13     0.14 1.00     2951     2720
size:new     0.05      0.02     0.01     0.09 1.00     2912     2446
Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape     9.42      1.33     7.04    12.16 1.00     2389     2424

> BIC(glm(price ~ -1 + size + size:new, family=Gamma(link=identity), data=Houses))
[1] 1134.883

> BIC(lm(price ~ -1 + size + size:new, data=Houses)) # corresponding normal model
[1] 1171.261
```

The simple interpretation is that the estimated selling price is 0.13(size) for older homes and (0.13 + 0.05)(size) for new homes; i.e., \$130 times the number of square feet for older homes and \$180 times the number of square feet for new homes.

R5.2 Bayesian Poisson Regression with `brms`

Bayesian Poisson regression is used when the response variable represents counts, such as the number of events occurring within a fixed period of time. Let Y_i denote the random

count for response individual i, $i = 1, \ldots, n$. We assume

$$Y_i \sim \text{Poisson}(\mu_i), \quad i = 1, \ldots, n,$$

with $\mu_i > 0$ being the expected count of the random variable Y_i, which is also equal to its variance, i.e., $\text{E}(Y_i) = \text{Var}(Y_i) = \mu_i$.

Poisson regression is a generalized linear model (GLM) for count data. IN presence of p explanatory variables, the systematic component is given by the linear predictor

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \quad i = 1, \ldots, n,$$

where β_0 is the intercept. Using the canonical link function for the Poisson distribution yields

$$\log(\mu_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \quad i = 1, \ldots, n,$$

Equivalently, in matrix notation we have

$$\log(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\beta},$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top$ denotes the parameter vector and

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}$$

the $n \times p$ design matrix, collecting all observed values on the explanatory variables for every response item.

In the Bayesian framework, prior distributions are assigned to the regression coefficients, for example

$$\beta_j \sim \mathcal{N}(0, 1), \quad j = 1, \ldots p, \quad \text{and} \quad \beta_0 \sim \mathcal{N}(0, 2).$$

We next illustrate Bayesian fitting of a Poisson regression model using the `brms` package and the accompanying `epilepsy` dataset. The response variable `count`, represents the number of seizures. We consider two explanatory variables, namely `Age`, representing the patient's age, and `Trt`, an indicator of treatment group assignment.

```
> library(brms)
> data("epilepsy", package = "brms")    # load example data
>  head(epilepsy,3)                     # see variables in the dataset
  Age Base Trt patient visit count obs       zAge      zBase
1  31   11   0       1     1     5   1  0.4249950 -0.7571728
2  30   11   0       2     1     3   2  0.2652835 -0.7571728
3  25    6   0       3     1     2   3 -0.5332740 -0.9444033
> fit_poisson <- brm(count ~ Age + Trt, data = epilepsy, family = poisson(),
+             prior = c(            # weakly informative priors
+             prior(normal(0, 1), class="b"), prior(normal(0, 2), class="Intercept")),
+             chains = 4, iter = 2000, warmup = 1000, seed = 123)
> summary(fit_poisson)
 Family: poisson
  Links: mu = log
Formula: count ~ Age + Trt
   Data: epilepsy (Number of observations: 236)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
Intercept     2.53      0.11     2.32     2.74 1.00     4572     2782
Age          -0.01      0.00    -0.02    -0.01 1.00     4926     2933
Trt1         -0.09      0.05    -0.18    -0.00 1.00     3029     2495

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).

# Posterior summaries
posterior_summary(fit_poisson)
                 Estimate  Est.Error          Q2.5         Q97.5
b_Intercept  2.530834e+00 0.10779877     2.3236280  2.740066e+00  # in log-scale
b_Age       -1.327843e-02 0.00362227    -0.0202980 -6.385003e-03
b_Trt1      -9.310625e-02 0.04568990    -0.1847477 -3.418457e-03
Intercept    2.105617e+00 0.02310696     2.0583228  2.150003e+00
lprior      -4.009705e+00 0.01277966    -4.0346019 -3.984098e+00  # log prior density
lp__        -1.639470e+03 1.20436639 -1642.6174554 -1.638091e+03  # log posterior density

> plot(fit_poisson)                    # diagnostic plots   (Figure R.5.1, left)
> pp_check(fit_poisson)                # posterior predictive checks (Figure R.5.1, right)
Using 10 posterior draws for ppc type 'dens_overlay' by default.

ce <- conditional_effects(fit_poisson)  # marginal effects (Figure R.5.2)
plot(ce)
```

The histograms of samples from the posterior distributions of model's parameters, representing the empirical posterior distributions of the parameters, are provided in Figure R5.1 (left), along with trace plots of the MCMC draws for all model parameters. The trace plots are used to assess mixing and convergence of the chains. In Figure R5.1 (right), the model's posterior predictive check is provided. It shows the estimated density of the observed data (black line), along with densities of replicated datasets drawn from the posterior predictive distribution (light blue lines). The plot assesses whether the Poisson model can reproduce the distributional shape of the observed counts, with good overlap indicating an adequate model fit.

The plot in Figure R5.2 shows the marginal effect of `Age` on the expected response, obtained by averaging the model predictions over the observed distribution of `Trt`.

In this example, $\exp(b_{\text{Intercept}}) = \exp(2.53) \approx 12.56$ represents the model-implied baseline mean count for an individual with `Age` equal to zero and in the reference category of `Trt` (=0), whereas the reported `Intercept` (≈ 2.11) is a posterior summary of the expected response averaged over the observed distribution of the covariates in the data, and therefore corresponds to a marginal (population-averaged) mean rather than the baseline mean implied by the linear predictor.

The coefficient for `Age` is $b_{\text{Age}} \approx -0.0133$, implying that a one-unit increase in age changes the expected number of seizures by a factor of $\exp(-0.0133) \approx 0.987$, corresponding to an approximate 1.3% decrease in the expected count per unit increase in age, holding treatment constant. The coefficient for `Trt` is $b_{\text{Trt1}} \approx -0.0931$, so the expected number of seizures in the treatment group is multiplied by $\exp(-0.0931) \approx 0.911$ relative to the reference group, corresponding to an approximate 8.9% reduction in expected counts, holding age constant.

R5.3 Posterior Estimation of Random Effect Parameters

We noticed in Section A.5.3 of the book's `R` Appendix for the TV series data that in order to obtain the posterior point and interval estimates for β_j provided in Table A.4, we need

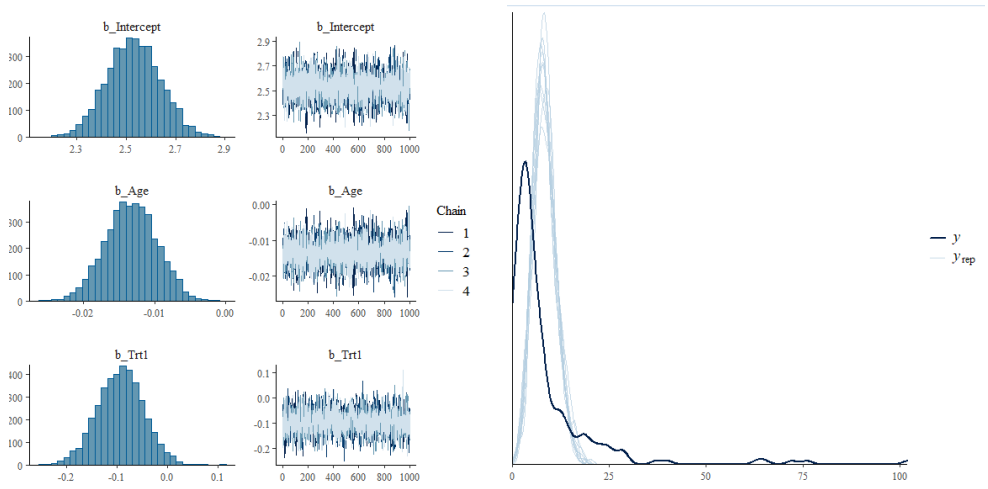

FIGURE R5.1: Left: Histograms of samples from the posterior distributions of model's parameters along with trace plots of the MCMC draws for all model parameters. Right: Posterior predictive check for the fitted model. The black line shows the estimated density of the observed data, while the light blue lines represent densities of 10 replicated datasets drawn from the posterior predictive distribution.

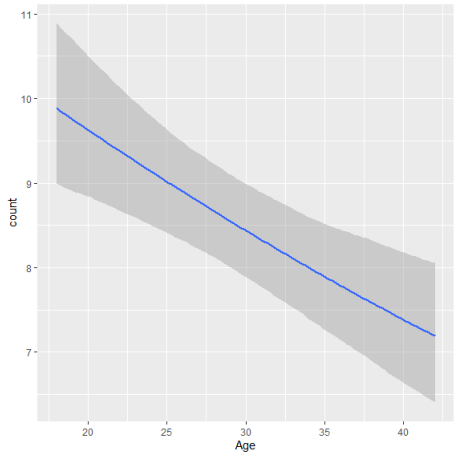

FIGURE R5.2: Solid blue line: posterior mean of the expected seizure count E(Y | Age), obtained by averaging over posterior draws and marginalising over `Trt`. Grey shaded area: pointwise 95% credible interval for the expected mean function.

first to split the posterior draws of our fitted model (saved under `fit.bayes`) by the levels of the factor (i.e., by the 11 levels of `series`) and compute β_j by adding the draws from the posterior distribution of the intercept `b_Intercept` and from the posterior distribution of the factor level `r_series`. Continuing the `R` code of the Section A.5.3, this procedure[1] is implemented as follows.

[1] for details see in https://bayesf22-notebook.classes.andrewheiss.com/bayes-rules/17-chapter.html

```
> library(tidyverse)
> library(tidybayes)
> beta_s <- fit.bayes |>
+   spread_draws(b_Intercept, r_series[series,]) |>
+   mutate(series_intercept = b_Intercept + r_series) |>
+   ungroup() |>
+   mutate(series = fct_reorder(factor(series), series_intercept, .fun = mean))
> series.means <- with(beta_s, tapply(series_intercept, series, mean)) # posterior means
> series.means                                                         # of beta_j
> q.fct <- function(x, prob=c(0.025,0.975)){quantile(x, probs = prob)} # 95% post. CI
> series.CI95 <- with(beta_s, tapply(series_intercept, series, q.fct))
> series.CI95
```

R6

Bayesian Computations and Diagnostics

R6.1 Gibbs Sampler for the Normal Linear Model*

In Section A.6.1 of the book's `R` Appendix, we illustrated how the Gibb's sampling algorithm samples from the full conditional distributions of the parameters of a normal linear model $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ and σ^2, in terms of the impairment example. We used the `Gibbs.regression` function of the package `lrgs` (linear regression by Gibbs sampling), assuming a multivariate normal prior distribution for $\boldsymbol{\beta}$ (Section 4.6.3) and an inverse gamma prior distribution for σ^2, which are the default priors in `lrgs`.

Actually, `lrgs` imposes by default a flat *inverse-Wishart* distribution for a covariance matrix $\boldsymbol{\Sigma}$, which has the benefit of producing always positive definite draws, if $n \geq p$. In the scalar case where the covariance matrix is a single variance term σ^2, the inverse-Wishart is simplified to the inverse-Gamma distribution (as in our example).

R6.2 Gibbs Sampler for Bayesian Linear Regression*

In the book and the section above, we implemented the Gibbs sampler in the `lrgs` package. Next, we provide a user defined `R` function for Gibbs sampling in the context of linear regression with two explanatory variables, a multivariate normal prior for the parameter vector $\boldsymbol{\beta}$ and an inverse gamma prior for the response's variance σ^2.

```
library(MASS)

gibbs_sampling <- function(y, X1, X2, n_iter = 2000, burn_in = 1000,
                           alpha_0 = 1, beta_0 = 1) {
  n <- length(y)
  X <- cbind(1, X1, X2)

  Sigma2.B <- c(100^2, 0.1^2, 0.1^2)  # prior variances

  XtX <- crossprod(X)
  Xty0 <- crossprod(X, y)

  beta <- c(0, 0.2, -0.2)   # initial values
  sigma2 <- 100

  # Storage
  beta_samples <- matrix(NA, nrow = n_iter - burn_in, ncol = 3)
  sigma_samples <- numeric(n_iter - burn_in)

  k <- 1
  for (iter in 1:n_iter) {

    A <- XtX/sigma2 + diag(1/Sigma2.B)  # posterior precision matrix
```

```
    Vbeta <- solve(A)                      # posterior covariance

    mbeta <- Vbeta %*% (Xty0/sigma2)    # posterior mean

    repeat {                           # draw beta until stationary
      beta_draw <- as.vector(
        mvrnorm(1, mu = mbeta, Sigma = Vbeta))

      # Companion matrix
      Bmat <- matrix(c(beta_draw[2], beta_draw[3], 1, 0),
                     2, 2, byrow = TRUE)
      eigmax <- max(Mod(eigen(Bmat, only.values = TRUE)$values))
      if (eigmax <= 1) {
        beta <- beta_draw
        break}   }

    # Update sigma2
    resid <- y - X %*% beta
    shape <- alpha_0 + n/2
    rate  <- beta_0 + sum(resid^2)/2
    sigma2 <- 1 / rgamma(1, shape = shape, rate = rate)

    if (iter > burn_in) {    # store after burn-in
      beta_samples[k, ] <- beta
      sigma_samples[k] <- sigma2
      k <- k + 1  }
  }
  list(beta_samples = beta_samples, sigma_samples = sigma_samples)
}
```

The `gibbs_sampling` function is illustrated below on a simulated data set.

```
set.seed(123)
n <- 100  # sample size
X1 <- rnorm(n)
X2 <- rnorm(n)
beta_true <- c(1, 2, 3)
y <- beta_true[1] + beta_true[2] * X1 + beta_true[3] * X2 + rnorm(n)
# run Gibbs sampling
samples <- gibbs_sampling(y, X1, X2, n_iter=2000, burn_in=1000)

summary(samples$beta_samples)   # summarize results
summary(samples$sigma_samples)
```

For the `Mental` dataset, we get:

```
Mental <- read.table("http://stat4ds.rwth-aachen.de/data/Mental.dat", header=TRUE)
y <- Mental$impair
X1 <- Mental$life
X2 <-Mental$ses
samples <- gibbs_sampling(y, X1, X2, n_iter = 4000, burn_in = 1000)
summary(samples$beta_samples)              # summarize results
summary(samples$sigma_samples)
```

Finally, we can plot the trace plots for the draws from the parameters' posterior samples

```
par(mfrow = c(3, 1))
plot(b0_post, type = "l", main = "Trace of beta_0")
plot(b1_post, type = "l", main = "Trace of beta_1")
plot(b2_post, type = "l", main = "Trace of beta_2")
plot(sigma2_post, type = "l", main = "Trace of sigma^2")
```

R7

Comparison and Choice of Bayesian Models

R7.1 Two Perspectives on Bayes Factor Computation: Analytical Testing vs. Simulation-Based Model Comparison

We have seen Bayes Factor (BF) computation implemented in two different ways in this book, which reflect two different perspectives on Bayesian model comparison.

First, in the context of contingency table analysis (Section 2.5.5) and hypothesis testing for a proportion (Section A.2.1 of the book's `R` Appendix), Bayes factors (BFs) were computed using the `BayesFactor` package. In this framework, BFs are obtained in a relatively direct way for a restricted class of models. For example, when testing independence in a contingency table that cross-classified random variables X and Y, we compare two models:

$$\mathcal{M}_0 : X \text{ and } Y \text{ are independent}, \quad \mathcal{M}_1 : X \text{ and } Y \text{ are dependent}.$$

The BF is then defined as the ratio of the marginal distributions (see (2.7) in Section 2.3.5):

$$BF_{10} = \frac{m_1(\boldsymbol{y})}{m_0(\boldsymbol{y})} = \frac{\int_{\Theta_1} f_1(\boldsymbol{y} \mid \theta) p_1(\theta) d\theta}{\int_{\Theta_0} f_0(\boldsymbol{y} \mid \theta) p_0(\theta) d\theta}.$$

In the `BayesFactor` package, these integrals are often evaluated using suitably chosen conjugate priors or analytic approximations, which makes computation fast and stable for standard designs such as proportions, contingency tables, t-tests, and ANOVA models. In these cases the marginal likelihoods are either available in closed form or approximated in a highly optimized way.

Second, in Section 7.1, BFs were used for Bayesian model choice via the `bridgesampling` approach. Here the perspective is more general and computationally intensive. We assume that two or more competing models have already been fitted, for example via MCMC methods in `brms`. The BF is again defined as the ratio of marginal likelihoods, but now each marginal likelihood is typically intractable. The `bridgesampling` package approximates this integral using importance sampling with a bridge distribution that connects the posterior and prior. The resulting estimator combines draws from the posterior distribution with a carefully chosen proposal distribution to stabilize the ratio of normalizing constants.

Although both approaches compute BFs, they differ in how the marginal likelihood is obtained. In the `BayesFactor` package, the BF is computed directly for restricted model classes where the marginal likelihood can be computed either in closed form or with highly optimized low-dimensional numerical integration. The prior structure is typically fixed and chosen to yield tractable solutions. On the other hand, in the `bridgesampling` package, the BF is computed indirectly by first fitting models using simulation-based methods (e.g., MCMC), and then numerically approximating the marginal likelihood, requiring a higher computational effort. Thus, bridgesampling provides a general-purpose method for estimating marginal likelihoods, thereby enabling Bayesian model comparison for essentially arbitrary models.".

To illustrate and compare the computation of Bayes factors using the `BayesFactor` and `bridgesampling` packages, we consider a simple example involving a two-sample t-test for independent groups under the assumption of equal variances. We use the data from Section A.3.5 of the book's `R` Appendix, where reaction times are compared between drivers using a cell phone while driving and a control group not using a cell phone. We demonstrate below both the analytic Bayes factor computation implemented in `BayesFactor` and the more general bridge-sampling approach based on Bayesian model fitting with `brms`.

```
> # ------------------------------------------------------------------------------
> # Simulated reaction-times, given mean and variance of the two independent groups
> # ------------------------------------------------------------------------------
> set.seed(123)
> m1 <- 585.2; s1 <- 89.6; n1 <- 32   # cell phone group
> m2 <- 533.7; s2 <- 65.3; n2 <- 32   # control group
> z1 <- rnorm(n1); y1 <- scale(z1) * s1 + m1
> z2 <- rnorm(n2); y2 <- scale(z2) * s2 + m2
> library(BayesFactor)
> bf_ttest <- ttestBF(x = y1, y = y2, paired = FALSE)
>bf_ttest
Bayes factor analysis
--------------
[1] Alt., r=0.707 : 4.378946 ±0%
Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

The Bayes factor analysis using the `BayesFactor` package yielded $BF_{1,0} = 4.38$, indicating that the observed data are approximately 4.4 times more likely under the alternative hypothesis $H_1 : (\mu_1 - \mu_2)/\sigma \neq 0$ than under the null hypothesis $H_0 : (\mu_1 - \mu_2)/\sigma = 0$, where $(\mu_1 - \mu_2)/\sigma$ is the standardized effect size μ_1 and μ_2 are the mean reaction times of drivers using a cell phone and drivers in the control group, respectively. According to commonly used guidelines for interpreting Bayes factors (see Table 2.1 in p. 58 of the book), values between 3 and 20 provide weak evidence against H_0. Thus, the present result suggests weak evidence for a difference in reaction times between the two groups. The analysis used a Jeffreys–Zellner–Siow (JZS) prior, corresponding to a Cauchy prior on the standardized effect size with default scale parameter $r = 0.707$, for a moderately dispersed prior distribution on the standardized effect size. The `rscale` argument controls the scale of the prior distribution, with `rscale=1` (or `"wide"`) yielding a standard Cauchy prior. Alternative options for `rscale` are `"mediun"` ($r = \sqrt{2}/2 = 0.707$, default) and `"ultrawide"` ($r = \sqrt{2}$). [1]

The BF is computed via Gaussian quadrature with essentially no numerical uncertainty (therefore the reported numerical error is ±0%).

Next, we compute the BF for the same example using the packages `bridgesampling` and `brms`. For the same seed as above we obtain a BF value close to that of `bf_ttest`.

```
> library(brms)
> library(bridgesampling)
> dat <- data.frame(        # create data frame
+   rt = c(y1, y2), group = factor(c(rep("cell-phone", n1), rep("control", n2)))
+  )
> fit1 <- brm(              # H_1 model: includes group effect
+         rt ~ group, data = dat, family = gaussian(),
+         prior = c(prior(normal(0, 100), class = "b"),
+         prior(student_t(3, 0, 100), class = "Intercept"),
+         prior(student_t(3, 0, 100), class = "sigma")),
```

[1] See the discussion and references in p. 51–54 in BayesFactor.pdf. See also the manual in BayesFactor manual.

```
+         iter = 4000, warmup = 1000, chains = 4, seed = 123,
+         save_pars = save_pars(all = TRUE) )

> fit0 <- brm(              # H_0 model: no group effect
+         rt ~ 1, data = dat, family = gaussian(),
+         prior = c(prior(student_t(3, 0, 100), class = "Intercept"),
+         prior(student_t(3, 0, 100), class = "sigma")),
+         iter = 4000, warmup = 1000, chains = 4, seed = 123,
+         save_pars = save_pars(all = TRUE))

>
> bridge1 <- bridge_sampler(fit1)    # bridge sampling
> bridge0 <- bridge_sampler(fit0)
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
>
>
> bf_bridge <- bf(bridge1, bridge0)  # BF (H_1 vs H_0)
> bf_bridge
Estimated Bayes factor in favor of bridge1 over bridge0: 4.67085
```

R7.2 The *k*-Nearest Neighbors Classification Method in R

In Section 7.3.5, we briefly discussed another simple classification method, the ***k-nearest neighbors*** (KNN) algorithm, without considering its implementation.

The prediction aspect of the KNN method is available with the `knn` function in the `class` package of `R`. For a random partition of the data into a training sample and a test sample, for each observation in the test sample, the function finds the majority class for the k nearest neighbors in the training sample. (For a binary response, selecting k to be an odd number avoids ties.) A classification table can then summarize the accuracy of the method as applied to the test sample. We illustrate with $k = 1$ nearest neighbor for predicting Iris species (data file: `Iris`).

```
> s_l2 <- scale(Iris$s_length); s_w2 <- scale(Iris$s_width)
> Iris2 <- data.frame(cbind(Iris\$y, s_l2, s_w2))
> train_index <- sample(nrow(Iris2), (1/2)*nrow(Iris2))
> Iris_train <- Iris2[train_index, ]
> Iris_test <- Iris2[-train_index, ]
> target_cat <- Iris2[train_index, 1]
> test_cat <- Iris2[-train_index, 1]
> library(class)
> predictions <- knn(Iris_train, Iris_test, cl=target_cat, k=1)
> table(predictions, test_cat)
           test_cat
predictions  0  1
          0 25  0
          1  0 25
```

The predictions for the 50 cases in the test set are perfect using only a single neighbor. The results depend on the sets chosen for training and test. For more details about nearest-neighbor methods, see James et al. (2021, Sec. 2.2.3, 3.5, and 4.7.6)[2].

[2] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R*, 2nd ed., Springer

The classical KNN algorithm is a non-parametric frequentist classification method. For a new observation x, the class assignment is determined by the most frequent class among the k observations in the training sample that are closest to x:

$$\hat{y}(x) = \arg\max_{c} \sum_{i \in N_k(x)} \mathbb{1}(y_i = c),$$

where $N_k(x)$ denotes the set of the k nearest neighbors of x, with the tuning parameter k being fixed in advance, and $\mathbb{1}(\cdot)$ is the indicator function.

In contrast, Bayesian KNN methods treat classification probabilistically. Instead of assigning only a class label, they model posterior probabilities of the form

$$p(y_{\text{new}} = c \mid x_{\text{new}}, \text{data}),$$

often by introducing a prior distribution on quantities such as the number of neighbors k or the strength of neighborhood dependence. Consequently, uncertainty about the classification rule can be incorporated into the inference process. Bayesian KNN methods are computationally substantially more involved than the classical algorithm and there is no standard, widely used `R` function for Bayesian KNN, comparable to `knn()` of the `class` package.[3]

R7.3 Neural Networks in `R`

We have commented in Section 7.3.5 that, by contrast with the simplicity of classification trees and the KNN method, the ***neural network*** method is a complex algorithm for classification and regression.

The fitting of a neural network is a black-box method that uses an algorithm, called *back-propagation*. The algorithm compares the prediction that a network produces with the actual response outcome and uses the difference to modify the weights of the connections between units in the network, working backward. As for any method, increasing complexity or the number of parameters, the predictions tend to be closer to the observed data. Performance is better assessed by testing the obtained prediction function on other data, such as with cross-validation. The method is commonly applied by randomly partitioning the data into a training sample and a test sample, performing the neural network analysis on the training sample, and checking its accuracy when applied to the test sample. We illustrate it here on the `Iris` data using the `neuralnet` package.

```
> train_index <- sample(nrow(Iris), (1/2)*nrow(Iris))
> Iris_train <- Iris[train_index, ]                    # train with 1/2 of data
> Iris_test <- Iris[-train_index, ]                    # test with 1/2 of data
> library(neuralnet)
> nn <- neuralnet(y=="1" ~ s_length + s_width, Iris_train, linear.output=F)
> pred.nn <- predict(nn, Iris_test)                 # results differ when repeat analysis,
> classtab <- table(Iris_test\$y==1, pred.nn[,1] > 0.50) # because test sample randomly
> classtab                                          # chosen using sample proportion cutoff
       FALSE TRUE
  FALSE    17    8
  TRUE     12   13
> sum(diag(classtab))/nrow(Iris_test)               # proportion of correct classification
[1] 0.6
> plot(nn, information=F)                           # plots the neural network (not shown)
```

[3] See, for example, Nuti, G. (2019). An Efficient Algorithm for Bayesian Nearest Neighbours. *Methodol Comput Appl Probab* **21**, 1251–1258.

Bayesian neural networks can also be formulated and fitted, but there is not available in `R` a simple, ready-to-use function, comparable to the frequentist implementation in the `neuralnet` package. Their fit is beyond the aims and level of our book.

R7.4 Bayesian Models with Missing Data: Multiple Imputation and Convergence

We have illustrated in Section 7.5.3 multiple imputation for treating missing data for the Mental Impairment Study using the `R` package `mice` (**m**ultiple **i**mputation by **c**hained **e**quations) together with the `brms` package for Bayesian model-fitting. Observing the output, provided in p. 298, we commented that the analysis automatically combines the results from fitting the model to all m samples obtained with multiple imputation. In the summary output, the *R-hat* values indicated possible convergence problems (close to or higher than 1.1). As mentioned by Bürkner[4], for models based on multiple imputed data sets, this may not indicate non-convergence but happen because chains of different sub-models are actually fitted to different data. For this, convergence should be investigated for the submodels separately, as shown next.

Recall that in Section 7.5.3 we fitted multiple models with $m = 100$ using the `brm_multiple()` function. Here, we set $m = 10$.

```
> Mental<-read.table("http://bayes4ds.rwth-aachen.de/data/Mental_missing.dat",header=TRUE)
> library(mice)
> library(brms)
> m <- 10  # m = the number of multiple imputations
> imp <- mice(Mental, m = m, print = FALSE) # generate m data sets with mult. imputation
> fit_imp10 <- brm_multiple(impair ~ life + ses, data = imp, family = gaussian,
+           prior=prior(normal(0,100), class=Intercept) + prior(normal(0, 100), class=b)
+           + prior(exponential(0.01), class = sigma))
> posterior_summary(fit_imp10)
                 Estimate  Est.Error          Q2.5         Q97.5
b_Intercept   28.95794204 2.61642544   23.80835590   34.13257411
b_life         0.09790251 0.03437300    0.02997053    0.16515395
b_ses         -0.10770478 0.03782913   -0.18157686   -0.03219648
sigma          4.77194433 0.60347649    3.75464709    6.11808347
Intercept     27.29880952 0.75599579   25.80121778   28.78954305
lprior       -21.26250680 0.00637446  -21.27668905  -21.25169754
lp__        -138.21691765 1.94432453 -142.58397431 -134.83290559
```

We can get the Rhat values and plot the chains fitted on the imputed data sets (see Figure R7.1, right), as follows:

```
> library(posterior)
> rhat(fit_imp10)  # alternative way to see the Rhat values for all parameters
b_Intercept      b_life       b_ses       sigma   Intercept      lprior        lp__
   1.036911    1.009730    1.066710    1.035477    1.001812    1.031669    1.351760
> plot(fit_imp10, variable = "^b", regex = TRUE)       # see Figure R7.1
```

Next, we compute convergence measures separately for each of the 10 submodels. Observe that none of the models fitted on each of the 10 samples has convergence issues (all Rhat values are 1.00).

```
> draws <- as_draws_array(fit_imp10)
```

[4]See `https://cran.r-project.org/web/packages/brms/vignettes/brms_missings.html`

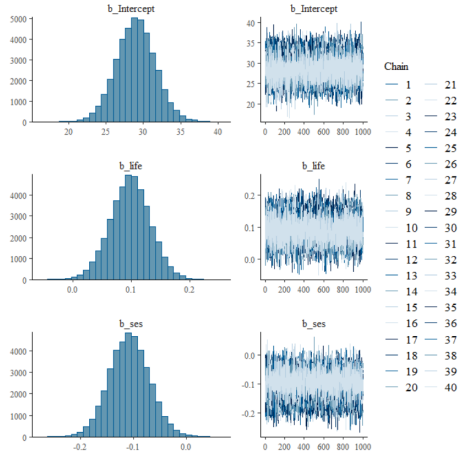

FIGURE R7.1: Left: Histograms of samples (after burn-in) from the posterior distributions of model's parameters. Each histogram is based on 40,000 samples, obtained from four chains of 1,000 iterations each for each of the $m = 10$ fitted submodels. Right: Trace plots for parameters for all chains.

```
> nc <- nchains(fit_imp10) / m    # every dataset has nc = 4 chains in this example
> draws_per_dat <- lapply(1:m, \(i) subset_draws(draws, chain = ((i-1)*nc+1):(i*nc)))
> lapply(draws_per_dat, summarise_draws, default_convergence_measures())
#  the output for each of the m=10 submodels follows:
[[1]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept  1.00    4561.    2852.
2 b_life       1.00    4012.    2908.
3 b_ses        1.00    4670.    2992.
4 sigma        1.00    3471.    2627.
5 Intercept    1.00    4628.    2613.
6 lprior       1.00    3725.    2770.
7 lp__         1.00    1770.    2231.

[[2]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept  1.00    4413.    3111.
2 b_life       1.00    4230.    2811.
3 b_ses        1.00    4147.    2681.
4 sigma        1.00    3693.    2686.
5 Intercept    1.00    4154.    2939.
6 lprior       1.00    3643.    2746.
7 lp__         1.00    1445.    2055.

[[3]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept 1.00     4748.    2981.
2 b_life      1.00     4129.    2815.
3 b_ses       1.00     4058.    2897.
4 sigma       1.00     3855.    2865.
5 Intercept   1.000    4066.    2687.
```

```
6 lprior      1.00     3735.    2838.
7 lp__        1.00     1895.    2763.

[[4]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept 1.000    4073.    2496.
2 b_life      1.000    3772.    2905.
3 b_ses       1.00     3996.    2878.
4 sigma       1.00     3544.    2617.
5 Intercept   1.00     4793.    2697.
6 lprior      1.00     3780.    2709.
7 lp__        1.00     1655.    2061.

[[5]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept  1.00    3980.    2255.
2 b_life       1.00    3705.    2748.
3 b_ses        1.00    4225.    2853.
4 sigma        1.00    3390.    2941.
5 Intercept    1.00    3559.    2825.
6 lprior       1.00    3407.    2907.
7 lp__         1.00    1828.    2224.

[[6]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept  1.00    4700.    2945.
2 b_life       1.00    4061.    2762.
3 b_ses        1.00    4320.    2942.
4 sigma        1.00    3724.    2710.
5 Intercept    1.00    3553.    2714.
6 lprior       1.00    3789.    2852.
7 lp__         1.00    1819.    2220.

[[7]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept 1.00     4750.    2564.
2 b_life      1.00     4165.    2777.
3 b_ses       1.00     4147.    2670.
4 sigma       1.00     3608.    2610.
5 Intercept   1.000    4242.    2814.
6 lprior      1.00     3596.    2675.
7 lp__        1.00     1642.    1924.

[[8]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept  1.00    4554.    2826.
2 b_life       1.00    4275.    2802.
3 b_ses        1.00    4419.    2990.
4 sigma        1.00    4041.    2918.
5 Intercept    1.00    4317.    2901.
6 lprior       1.00    4193.    2897.
7 lp__         1.00    2069.    2632.

[[9]]
```

```
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept 1.000    4067.    2676.
2 b_life      1.00     3225.    2610.
3 b_ses       1.00     3884.    2091.
4 sigma       1.00     3168.    2405.
5 Intercept   1.00     3562.    3220.
6 lprior      1.00     3022.    1975.
7 lp__        1.00     1608.    2114.

[[10]]
# A tibble: 7 × 4
  variable     rhat ess_bulk ess_tail
  <chr>       <dbl>    <dbl>    <dbl>
1 b_Intercept 1.00     4655.    3088.
2 b_life      1.00     4486.    2560.
3 b_ses       1.00     4378.    2929.
4 sigma       1.00     3949.    3110.
5 Intercept   1.00     4470.    2856.
6 lprior      1.00     4186.    2808.
7 lp__        1.000    1946.    3012.
```