Assignment 5

Kerem Karagöz Immanuel Klein

GitHub: https://github.com/immanuel-klein/bayesian-assignments.git

```
# load packages here
library(dplyr)
library(tidyverse)
library(ggplot2)
library(tinytex)
library(rethinking)
library(rstan)
library(reshape2)
library(ggridges)
```

Task Set 1

Load the data set `RiskyChoice.csv` to solve the Task Set 1. Use the `read_csv2()` function instead of `read_csv()`.

```
# load data here
# Used read.csv2 instead of read_csv2
# to correctly get the decimal seperators for NegativeAffect
# Then use as numerics to convert to num from character
risky <- read.csv2("RiskyChoice.csv")
risky$NegativeAffect <- as.numeric(risky$NegativeAffect)
head(risky)
```

```
  Subject AgeGroup ItemID Position CorrectChoice RiskyChoice Gender
1       1  younger      1       18             0           0 female
2       1  younger      2       92             0           1 female
3       1  younger      3       73             1           1 female
4       1  younger      4       44             1           0 female
5       1  younger      5        9             0           0 female
6       1  younger      6       78             1          NA female
  NegativeAffect Numeracy
1           1.75        9
2           1.75        9
3           1.75        9
4           1.75        9
5           1.75        9
6           1.75        9
```

Task 1.1

Create a reduced data table with only one row per subject that shows the number of solved choices problems (`nChoice`) and the number of correct choices (`nCorrect`) for each subject along with the other variables. Remove the subjects with missing values. Print the data of the first 10 subjects.

```
# write code here
risky.reduced <- risky %>%
  filter(!is.na(CorrectChoice)) %>%
  group_by(Subject) %>%
  summarise(
    Gender = first(Gender),
```

```
    NegativeAffect = first(NegativeAffect),
    Numeracy = first(Numeracy),
    nChoice = n(),
    nCorrect = sum(CorrectChoice)
  ) %>% na.omit()

head(risky.reduced, 10)
```

```
# A tibble: 10 x 6
   Subject Gender NegativeAffect Numeracy nChoice nCorrect
     <int> <chr>           <dbl>    <int>   <int>    <int>
 1       1 female           1.75        9     104       61
 2       2 female           2.5         7     104       66
 3       3 male             1.25       10     104       80
 4       4 female           1.25       10     104       71
 5       5 male             4           9     104       58
 6       6 female           6           9     104       69
 7       7 male             1          10     104       75
 8       8 female           1.5        10     104       66
 9       9 female           1.75        9     104       62
10      10 female           1.5         7     104       63
```

Remark: We understood the task the following way: If a subject has at least one NA value for CorrectChoice, do not take any values of this subject into the new table. However, this left us with a tibble with the dimensions 0x3, i. e. there is no subject that has no NA value for CorrectChoice. Thus, we decided to just filter the rows with NA for CorrectChoice instead of the whole subject. Also, because risky.reduced should only have one row per subject, we could only include Gender, NegativeAffect, and Numeracy. The others have differing values within the subjects. We then omit NAs.

Task 1.2

Run a Bayesian regression model that predicts `nCorrect` from `Numeracy` using fixed intercepts and fixed slopes. Standardize the predictor before running the model and compute the WAIC of the model.

```
# write data list and model here
risky.reduced.regr <- risky.reduced %>%
  mutate(
    numeracy_std = (Numeracy - mean(Numeracy)) / sd(Numeracy),
```

```
    # Convert Subject to a numeric index
    subject_idx = as.numeric(as.factor(Subject))
)

data.list <- list(
  nCorrect = risky.reduced.regr$nCorrect,
  nChoice = risky.reduced.regr$nChoice,
  numeracy_std = risky.reduced.regr$numeracy_std
)

m1 <- ulam(
  alist(
    nCorrect ~ dbinom(nChoice, p),
    logit(p) <- a + b*numeracy_std,
    # Choosing intercept and slope priors:
    # When numeracy is 0, it's reasonable that correct choices come from guessing.
    # Thus, ca 50% of all choices might be correct (0.5 of ~ 100 choices).
    # Looking at the values where numeracy is 10, a 0.3 slope might make sense.
    a ~ dnorm(50, 10),
    b ~ dnorm(0.3, 0.1)
  ), data = data.list, chains = 4, cores = 4, log_lik = TRUE
)
```

```
#write code here
WAIC(m1)
```

```
      WAIC      lppd penalty std_err
1 1001.382 -494.0751 6.61607 71.8496
```

Task 1.3

Run a Bayesian regression model that predicts `nCorrect` from `Numeracy` using random intercepts and fixed slopes. Standardize the predictor before running the model and compute the WAIC of the model.

```
# write data list and model here

data.list2 <- list(
  nCorrect = risky.reduced.regr$nCorrect,
  nChoice = risky.reduced.regr$nChoice,
  numeracy_std = risky.reduced.regr$numeracy_std,
```

```
  subject = risky.reduced.regr$subject_idx
)

m2 <- ulam(
  alist(
    nCorrect ~ dbinom(nChoice, p),
    logit(p) <- a[subject] + beta*numeracy_std,
    a[subject] ~ dnorm(a_bar, tau_a),
    a_bar ~ dnorm(50, 10),
    tau_a ~ dexp(0.5),
    beta ~ dnorm(0.3, 0.1)
  ), data = data.list2, chains = 4, cores = 4, log_lik = TRUE
)
```

```
# write code here
WAIC(m2)
```

```
      WAIC     lppd  penalty std_err
1 796.0019 -339.411 58.58998 16.6611
```

Task Set 2

Task 2.1

Create a data table that entails 10,000 posterior samples (rows) for each subject-specific (columns) intercept. Convert the sampled values into probabilities and print the first 10 samples of the first 10 subjects.

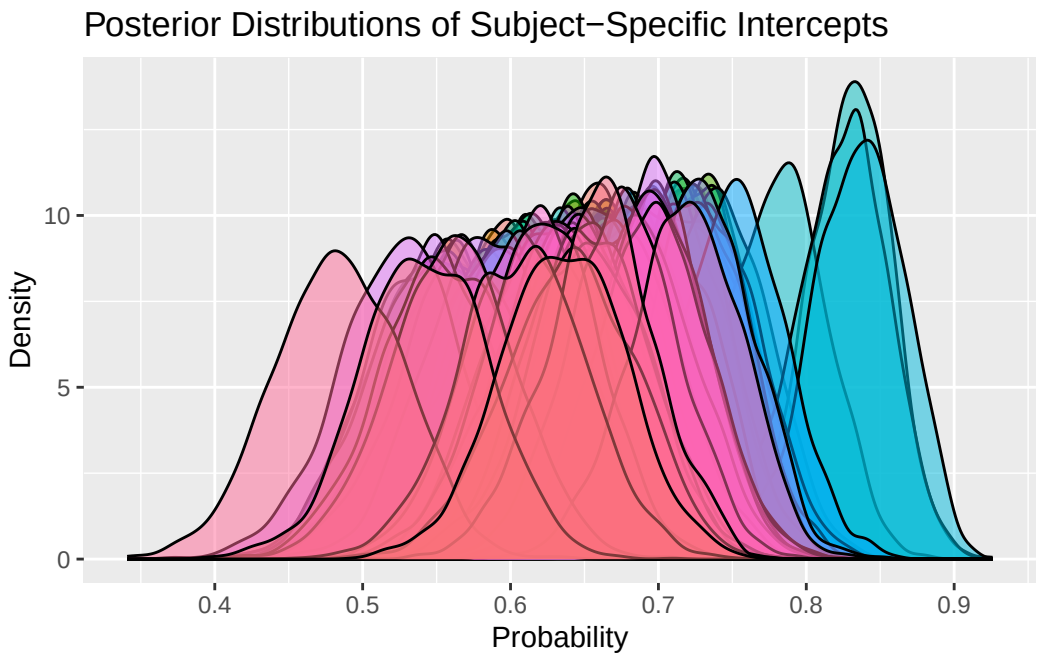
```
# write code here
samples <- extract.samples(m2, n = 10000)

prob.samples <- inv_logit(samples$a)

prob.samples.df <- as.data.frame(prob.samples)
colnames(prob.samples.df) <-
  paste0("Subject ", 1:ncol(prob.samples.df))

head(prob.samples.df[, 1:10], 10)
```

```
   Subject 1 Subject 2 Subject 3 Subject 4 Subject 5 Subject 6 Subject 7
1  0.5754439 0.7016720 0.7347092 0.6337369 0.5734770 0.6475287 0.6648301
2  0.6017943 0.6234707 0.6888083 0.6213743 0.6196853 0.6136659 0.6798422
3  0.5558088 0.6403710 0.7249455 0.7584280 0.5497451 0.6388814 0.6753602
4  0.6497846 0.7229613 0.7393021 0.6078877 0.4902690 0.6227069 0.7548778
5  0.6434495 0.6386334 0.6981084 0.6720511 0.5400521 0.5884192 0.7341647
6  0.4760311 0.6826883 0.6939088 0.6423397 0.5969645 0.6608435 0.6422491
7  0.5539647 0.6263647 0.7144651 0.6990872 0.5237200 0.6649359 0.6356159
8  0.6113872 0.6499650 0.6685675 0.6882346 0.5151118 0.6552968 0.6265920
9  0.6284138 0.6707631 0.7209547 0.6630184 0.5752882 0.7472701 0.6965919
10 0.6177177 0.5826505 0.7175217 0.7034498 0.5668636 0.6423184 0.6368311
   Subject 8 Subject 9 Subject 10
1  0.5507060 0.6104497  0.6096938
2  0.6195726 0.5933360  0.6964798
3  0.6231693 0.6601968  0.7028548
4  0.6827450 0.5823016  0.6571243
5  0.6631471 0.5887529  0.6207263
6  0.6649878 0.6087568  0.6274710
7  0.5927046 0.6667785  0.6108321
8  0.6583469 0.5167917  0.6210366
9  0.5697450 0.5998945  0.6755444
10 0.6993512 0.6044228  0.6078571
```

Task 2.2

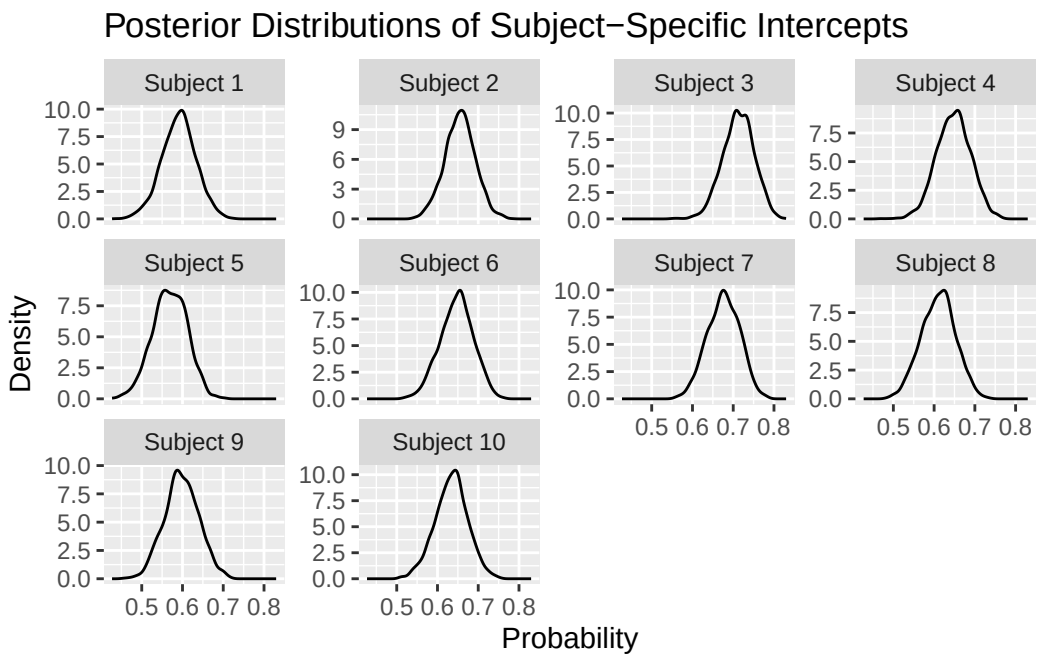
Use the posterior samples to plot the posterior distribution of all subject-specific intercepts to show the variability in the performance among subjects. Use the converted values (probabilities).

```
# write code here
# As there are 119 different subjects it is quite hard to visualize the intercepts

# Convert the wide format to long format for ggplot2
prob.samples.long <- melt(prob.samples.df)

# Plot the posterior distributions
ggplot(prob.samples.long, aes(x = value, fill = variable)) +
  geom_density(alpha = 0.5) +
  labs(title = "Posterior Distributions of Subject-Specific Intercepts",
       x = "Probability", y = "Density") +
  theme(legend.position = "none")
```




```
# An alternative version where we only use the first 10 subjects
prob.samples.subset <- prob.samples.df[, 1:10]
prob.samples.melt.subset <- melt(prob.samples.subset)
```

```
# Create a faceted plot for the first 10 subjects
ggplot(prob.samples.melt.subset, aes(x = value)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ variable, scales = "free_y") +
  labs(title = "Posterior Distributions of Subject-Specific Intercepts",
       x = "Probability", y = "Density")
```



Task 2.3

Consider the following posterior summaries and traceplots. Which model was estimated and what might be the cause of the convergence problems?

```
# precis(m3)
# traceplot_ulam(m3, pars = c("mu_a", "tau_a", "mu_b", "tau_b"))
```

The estimated model is most likely a hierarchical Bayesian regression model with random intercepts and slopes. It includes a Hierarchical structure with hyperparameters mu_a, tau_a, mu_b, and tau_b to model the population-level mean and standard deviation of the intercepts and slopes.

From the precis we can see that mu_a shows a good convergence with high effective sample size and rhat close to 1, meaning that chains are well mixed. When we go from mu_a to tau_a, mu_b and tau_b in order the effective sample size becomes smaller, rhat value is getting further away from 1 (except tau_a) and showing poorer convergences. So tau_b shows significant issues during chain mixing and convergence. Even though tau_a looks quite good from the parameters of precis its traceplot also collapses immediately.

The effective sample size (n_eff) for tau_b is very low (~44), suggesting that the sampler struggles to produce independent samples. The rhat value for tau_b is significantly above 1 (~1.106), indicating that the chains have not converged to the same distribution. tau_b might be poorly identified, leading to difficulties in sampling. The data may not provide enough information to reliably estimate this parameter, resulting in convergence problems.

The hierarchical nature of the model, with both random intercepts and slopes, increases the complexity of the parameter space. This complexity can make it challenging for the sampler to converge, especially if the data do not strongly inform all parameters.

The priors specified for the parameters (especially tau_a and tau_b) might be too restrictive or not well-aligned with the data. This misalignment can hinder the sampler's ability to explore the parameter space effectively.