Assignment 3 - Machine Learning

Elisius, Laurits, Tilde & Niels

02/11/2022

Assignment 3: Machine Learning

Part I - Simulating data

Use the meta-analysis reported in Parola et al (2020), create a simulated dataset with 100 matched pairs of schizophrenia and controls, each participant producing 10 repeated measures (10 trials with their speech recorded). for each of these “recordings” (data points) produce 10 acoustic measures: 6 from the meta-analysis, 4 with just random noise. Do the same for a baseline dataset including only 10 noise variables. Tip: see the slides for the code. #### Loading packages

```
pacman::p_load(tidyverse, brms, bayesplot, rstanarm, msm, cmdstanr, tidymodels, readr, broom.
mixed, dotwhisker, multilevelmod, recipes, caret, klaR, tidytext, DALEX, DALEXtra)
set.seed(123)
```

Data simulation (LL)

```
#defining sample size and trials
n <- 200
trials <- 10

#defining different effect sizes: 6 from meta analysis, 4 for random noise
InformedEffectMean <- c(rep(0,4), -0.23096087, -0.55698815, -0.05472132, -0.14332350, 0.20377
619, -0.41653998)
#introducing a skeptic effect mean
SkepticEffectMean <- rep(0,10)

#individual variability from population and across trials and measurement errors
IndividualSD <- 1
TrialSD <- 0.5
Error <- 0.2
```

```
# Setting up an empty data frame for the informed two conditions
Trial <- rep(1:10,100)
id_SZ <- rep(101:200, each = 10)
id_CON <- rep(1:100, each = 10)

CON <- data.frame(
  ID = id_CON,
  Trial = Trial,
  Condition = "Control") %>%
  mutate(
    v1 = NA,
    v2 = NA,
    V3 = NA,
    v4 = NA,
    v5 = NA,
    v6 = NA,
    v7 = NA,
    v8 = NA,
    v9 = NA,
    v10 = NA)

SZ <- data.frame(
  ID = id_SZ,
  Trial = Trial,
  Condition = "Schizophrenia") %>%
  mutate(
    v1 = NA,
    v2 = NA,
    V3 = NA,
    v4 = NA,
    v5 = NA,
    v6 = NA,
    v7 = NA,
    v8 = NA,
    v9 = NA,
    v10 = NA)
```

```
# Simulating the informed data
for (c in 1:1000){
  SZ[c,4:13] <- Map(rnorm,n=1,mean = InformedEffectMean/2, sd = IndividualSD)
}
for (c in 1:1000){
  CON[c,4:13] <- Map(rnorm,n=1,mean = (-InformedEffectMean)/2, sd = IndividualSD)
}

informed <- rbind(SZ,CON)
```

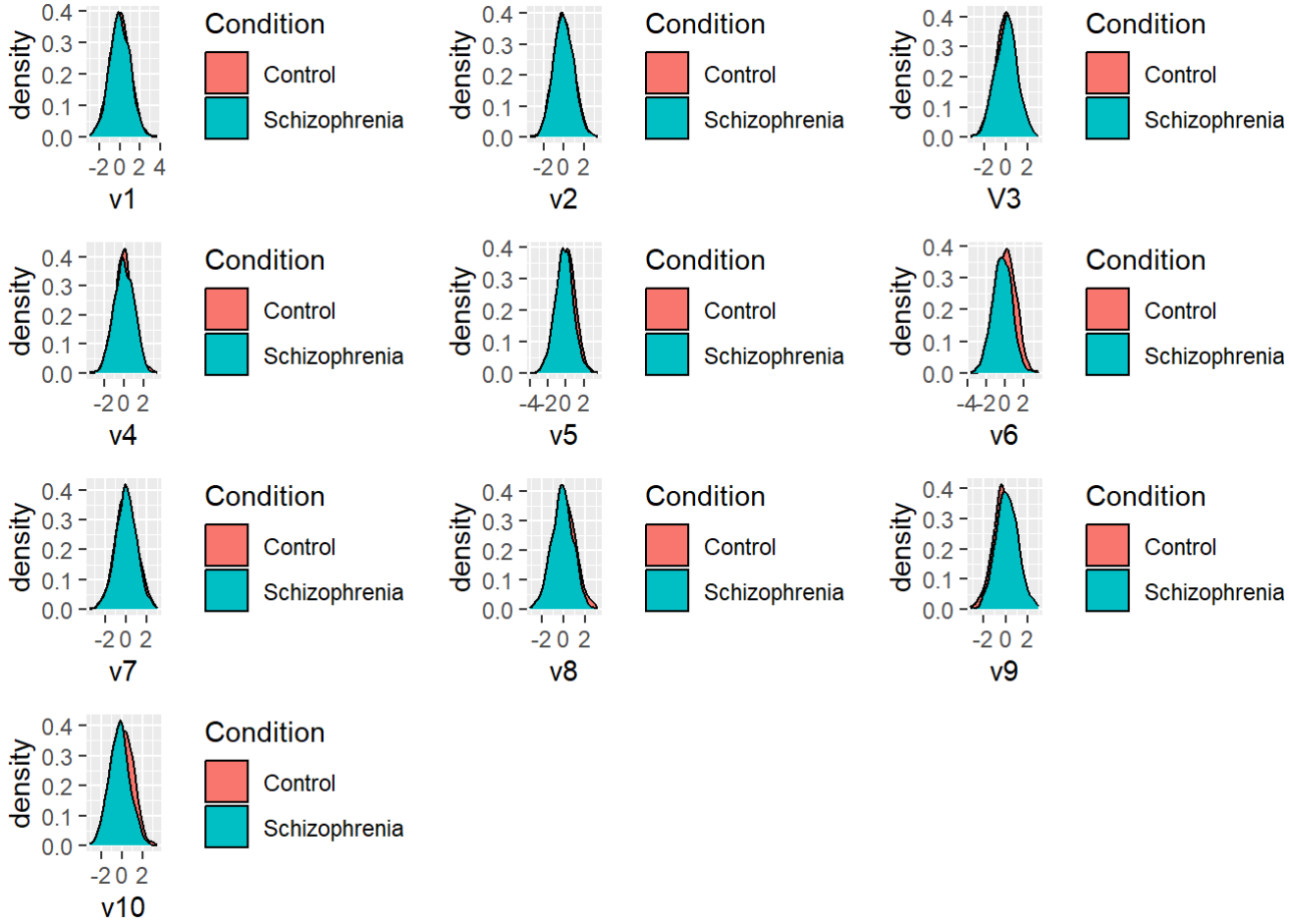
```
# Setting up the data frames for the two skeptic conditions
CON_S <- data.frame(
  ID = id_CON,
  Trial = Trial,
  Condition = "Control") %>%
  mutate(
    v1 = NA,
    v2 = NA,
    V3 = NA,
    v4 = NA,
    v5 = NA,
    v6 = NA,
    v7 = NA,
    v8 = NA,
    v9 = NA,
    v10 = NA)

SZ_S <- data.frame(
  ID = id_SZ,
  Trial = Trial,
  Condition = "Schizophrenia") %>%
  mutate(
    v1 = NA,
    v2 = NA,
    V3 = NA,
    v4 = NA,
    v5 = NA,
    v6 = NA,
    v7 = NA,
    v8 = NA,
    v9 = NA,
    v10 = NA)
```

```
# Simulating the data for the two skeptic conditions
for (c in 1:1000){
  SZ_S[c,4:13] <- Map(rnorm, n = 1, mean = SkepticEffectMean/2, sd = IndividualSD)
}
for (c in 1:1000){
  CON_S[c,4:13] <- Map(rnorm, n = 1, mean = (-SkepticEffectMean)/2, sd = IndividualSD)
}

Skeptic <- rbind(SZ_S,CON_S)
rm(SZ_S, SZ, CON_S, CON)
```

Plotting simulated data (TS)

Plotting the variables for the informed simulation

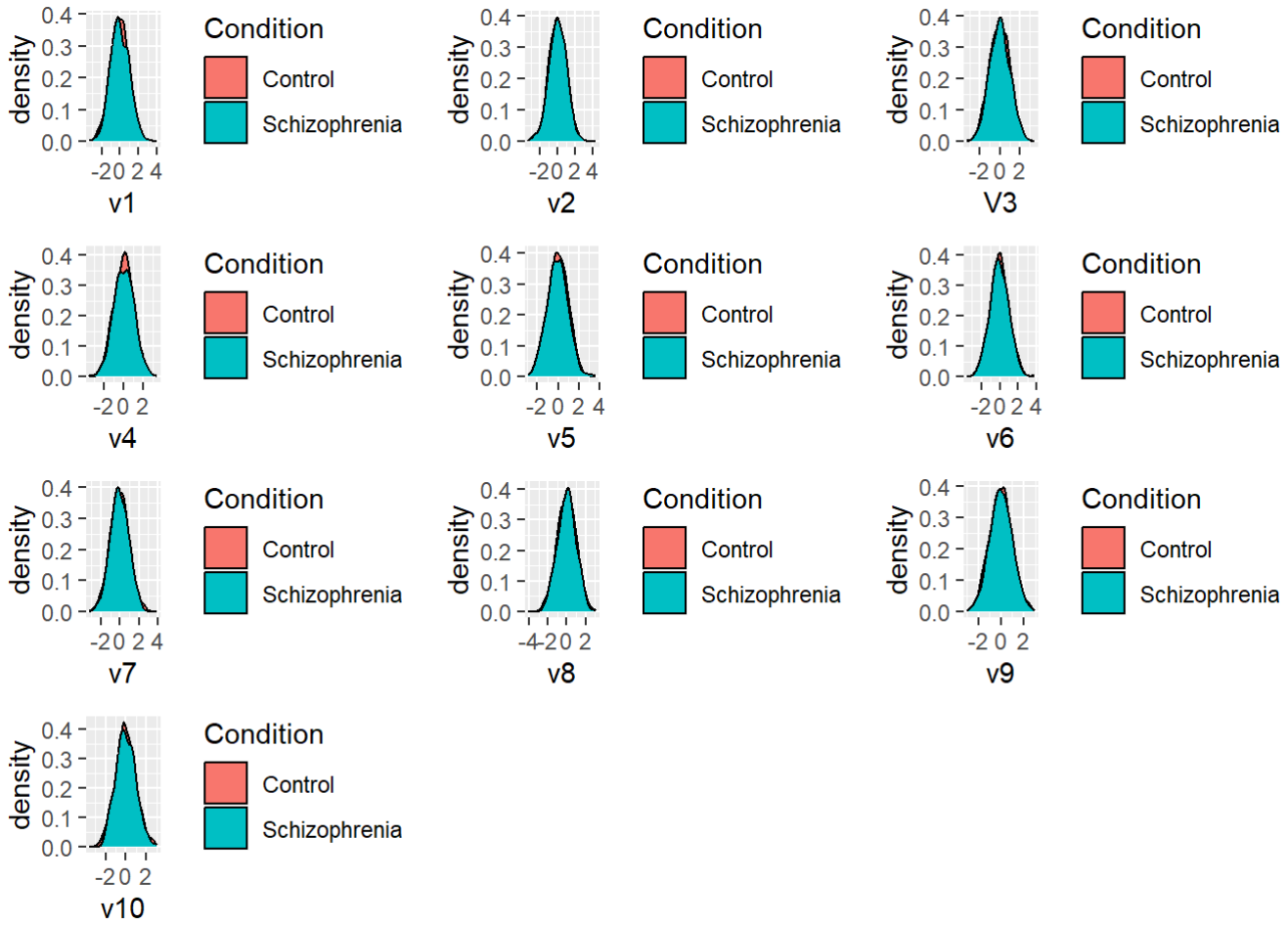
```
informed <- informed %>%
  mutate(Condition = as.factor(Condition))

plot1 <- informed %>%
  ggplot(aes(v1, fill = Condition)) + geom_density()

plot2 <- informed %>%
  ggplot(aes(v2, fill = Condition)) + geom_density()

plot3 <- informed %>%
  ggplot(aes(V3, fill = Condition)) + geom_density()

plot4 <- informed %>%
  ggplot(aes(v4, fill = Condition)) + geom_density()

plot5 <- informed %>%
  ggplot(aes(v5, fill = Condition)) + geom_density()

plot6 <- informed %>%
  ggplot(aes(v6, fill = Condition)) + geom_density()

plot7 <- informed %>%
  ggplot(aes(v7, fill = Condition)) + geom_density()

plot8 <- informed %>%
  ggplot(aes(v8, fill = Condition)) + geom_density()

plot9 <- informed %>%
  ggplot(aes(v9, fill = Condition)) + geom_density()

plot10 <- informed %>%
  ggplot(aes(v10, fill = Condition)) + geom_density()

gridExtra::grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, plot7, plot8, plot9, plot1
0)
```




```
rm(plot1, plot2, plot3, plot4, plot5, plot6, plot7, plot8, plot9, plot10)
```

Plotting the variables for the skeptic simulations

```
Skeptic <- Skeptic %>%
  mutate(Condition = as.factor(Condition))

plot1S <- Skeptic %>%
  ggplot(aes(v1, fill = Condition)) + geom_density()

plot2S <- Skeptic %>%
  ggplot(aes(v2, fill = Condition)) + geom_density()

plot3S <- Skeptic %>%
  ggplot(aes(V3, fill = Condition)) + geom_density()

plot4S <- Skeptic %>%
  ggplot(aes(v4, fill = Condition)) + geom_density()

plot5S <- Skeptic %>%
  ggplot(aes(v5, fill = Condition)) + geom_density()

plot6S <- Skeptic %>%
  ggplot(aes(v6, fill = Condition)) + geom_density()

plot7S <- Skeptic %>%
  ggplot(aes(v7, fill = Condition)) + geom_density()

plot8S <- Skeptic %>%
  ggplot(aes(v8, fill = Condition)) + geom_density()

plot9S <- Skeptic %>%
  ggplot(aes(v9, fill = Condition)) + geom_density()

plot10S <- Skeptic %>%
  ggplot(aes(v10, fill = Condition)) + geom_density()

gridExtra::grid.arrange(plot1S, plot2S, plot3S, plot4S, plot5S, plot6S, plot7S, plot8S, plot9
S, plot10S)
```




```
rm(plot1S, plot2S, plot3S, plot4S, plot5S, plot6S, plot7S, plot8S, plot9S, plot10S)
```

Part II - ML pipeline on simulated data

On the two simulated datasets (separately) build a machine learning pipeline: i) create a data budget (e.g. balanced training and test sets); ii) pre-process the data (e.g. scaling the features); iii) fit and assess a classification algorithm on the training data (e.g. Bayesian multilevel logistic regression); iv) assess performance on the test set; v) discuss whether performance is as expected and feature importance is as expected.

Bonus question: replace the bayesian multilevel regression with a different algorithm, e.g. SVM or random forest (but really, anything you'd like to try).

i) create a data budget (EL)

```
# creating pair columns, which pairs a CT with a SCZ participant for the informed and skeptic
data sets
# if ID is above 100 subtract 100 from ID if not keep the id
informed <- informed %>%
  mutate(pair=ID) %>%
  mutate(pair= ifelse(ID > 100, ID-100, ID) )

Skeptic <- Skeptic %>%
  mutate(pair=ID) %>%
  mutate(pair= ifelse(ID > 100, ID-100, ID))
```

```
# splitting informed into test and train based on the pair
sample1 <- sample(seq(100), 80)

informed_train <- subset(informed, pair %in% sample1)

informed_test <- subset(informed, !pair %in% sample1)

# splitting skeptic into test and train based on the pairs
sample2 <- sample(seq(100), 80)

Skeptic_train <- subset(Skeptic, pair %in% sample2)

Skeptic_test <- subset(Skeptic, !pair %in% sample2)
```

ii) pre-process the data (NV & TS)

```
# using tidymodels to pre-process the data using recipes.
rec_informed <- informed_train %>%
  recipe(Condition ~ .) %>%
  # excludes ID as a predictor
  update_role(ID, pair, new_role = "ID") %>%
  # normalizing the data
  step_normalize(all_numeric_predictors()) %>%
  # converting condition into dummy variables
  step_dummy(Condition) %>%
  prep(training = informed_train, retain = TRUE)

summary(rec_informed)
```

```
## # A tibble: 14 × 4
##    variable                type      role      source
##    <chr>                   <list>    <chr>     <chr>
##  1 ID                      <chr [2]> ID        original
##  2 Trial                   <chr [2]> predictor original
##  3 v1                      <chr [2]> predictor original
##  4 v2                      <chr [2]> predictor original
##  5 V3                      <chr [2]> predictor original
##  6 v4                      <chr [2]> predictor original
##  7 v5                      <chr [2]> predictor original
##  8 v6                      <chr [2]> predictor original
##  9 v7                      <chr [2]> predictor original
## 10 v8                      <chr [2]> predictor original
## 11 v9                      <chr [2]> predictor original
## 12 v10                     <chr [2]> predictor original
## 13 pair                    <chr [2]> ID        original
## 14 Condition_Schizophrenia <chr [2]> predictor derived
```

```
rec_skeptic <- Skeptic_train %>%
  recipe(Condition ~ .) %>%
  update_role(ID, pair, new_role = "ID") %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  step_dummy(Condition) %>%
  prep(training = Skeptic_train, retain = TRUE)

summary(rec_skeptic)
```

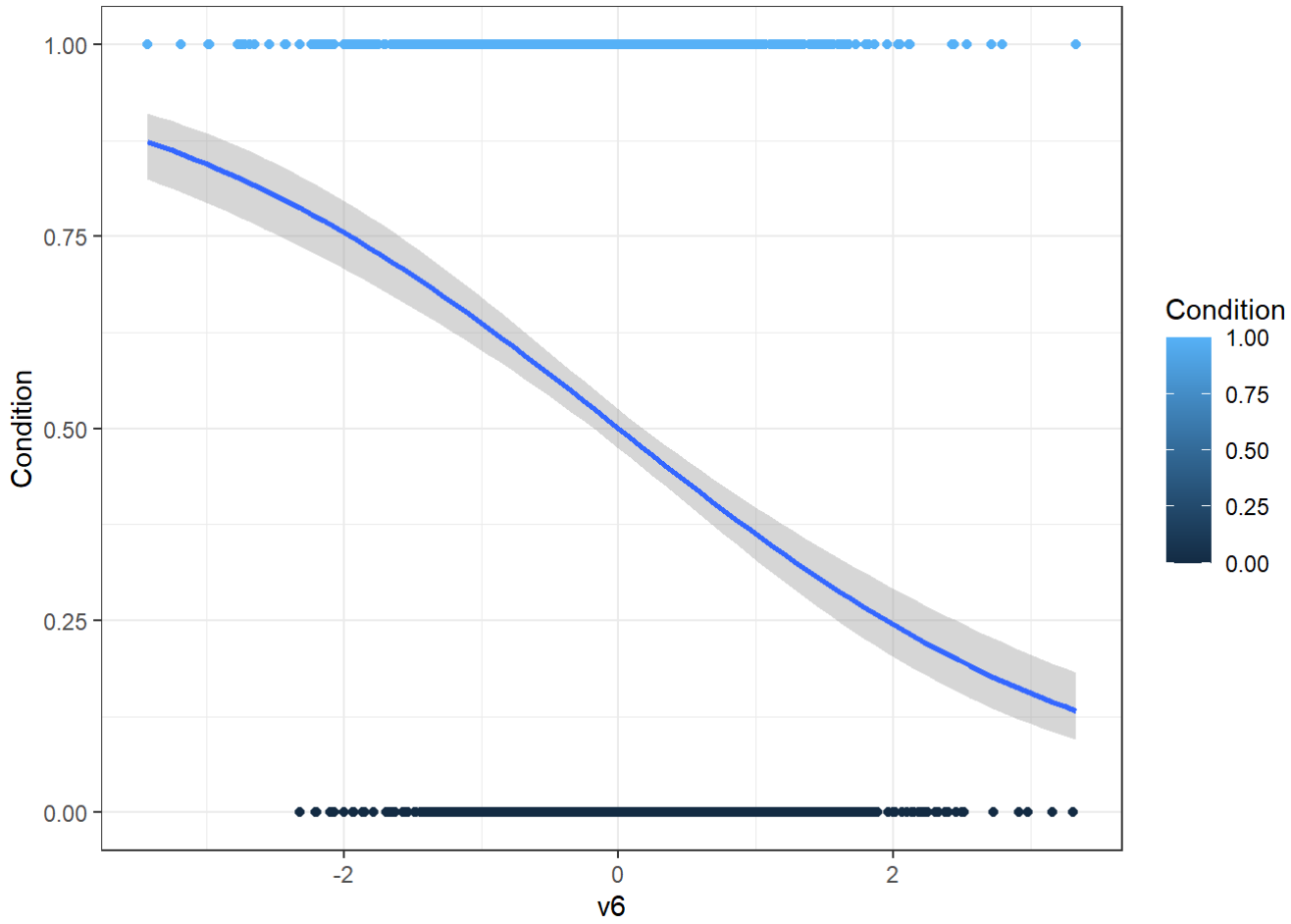
```
## # A tibble: 14 × 4
##    variable                type      role      source
##    <chr>                   <list>    <chr>     <chr>
##  1 ID                      <chr [2]> ID        original
##  2 Trial                   <chr [2]> predictor original
##  3 v1                      <chr [2]> predictor original
##  4 v2                      <chr [2]> predictor original
##  5 V3                      <chr [2]> predictor original
##  6 v4                      <chr [2]> predictor original
##  7 v5                      <chr [2]> predictor original
##  8 v6                      <chr [2]> predictor original
##  9 v7                      <chr [2]> predictor original
## 10 v8                      <chr [2]> predictor original
## 11 v9                      <chr [2]> predictor original
## 12 v10                     <chr [2]> predictor original
## 13 pair                    <chr [2]> ID        original
## 14 Condition_Schizophrenia <chr [2]> predictor derived
```

```
#Once the data are ready for transformation, the juices() extract transformed training set wh
ile the bake() function create a new testing set.

#Juice
informed_train_s <- juice(rec_informed)
skeptic_train_s <- juice(rec_skeptic)

#Bake
informed_test_s <- bake(rec_informed, new_data = informed_test)
skeptic_test_s <- bake(rec_skeptic, new_data = Skeptic_test)
```

```
# converting condition to factor
informed_train_s <- informed_train_s %>%
  mutate(Condition = as.factor(Condition_Schizophrenia))

skeptic_train_s <- skeptic_train_s %>%
  mutate(Condition = as.factor(Condition_Schizophrenia))

informed_test_s <- informed_test_s %>%
  mutate(Condition = as.factor(Condition_Schizophrenia))

skeptic_test_s <- skeptic_test_s %>%
  mutate(Condition = as.factor(Condition_Schizophrenia))
```

iii) fit and assess a classification algorithm on the training data (TS & EL)

```
# plotting the data
informed_train_s %>%
  mutate(Condition = as.numeric(Condition)-1) %>%
  ggplot() +
  geom_point(aes(v6, Condition, color = Condition))+
  geom_smooth(aes(v6, Condition), method = "glm", method.args = list(family = "binomial"))+
theme_bw()
```



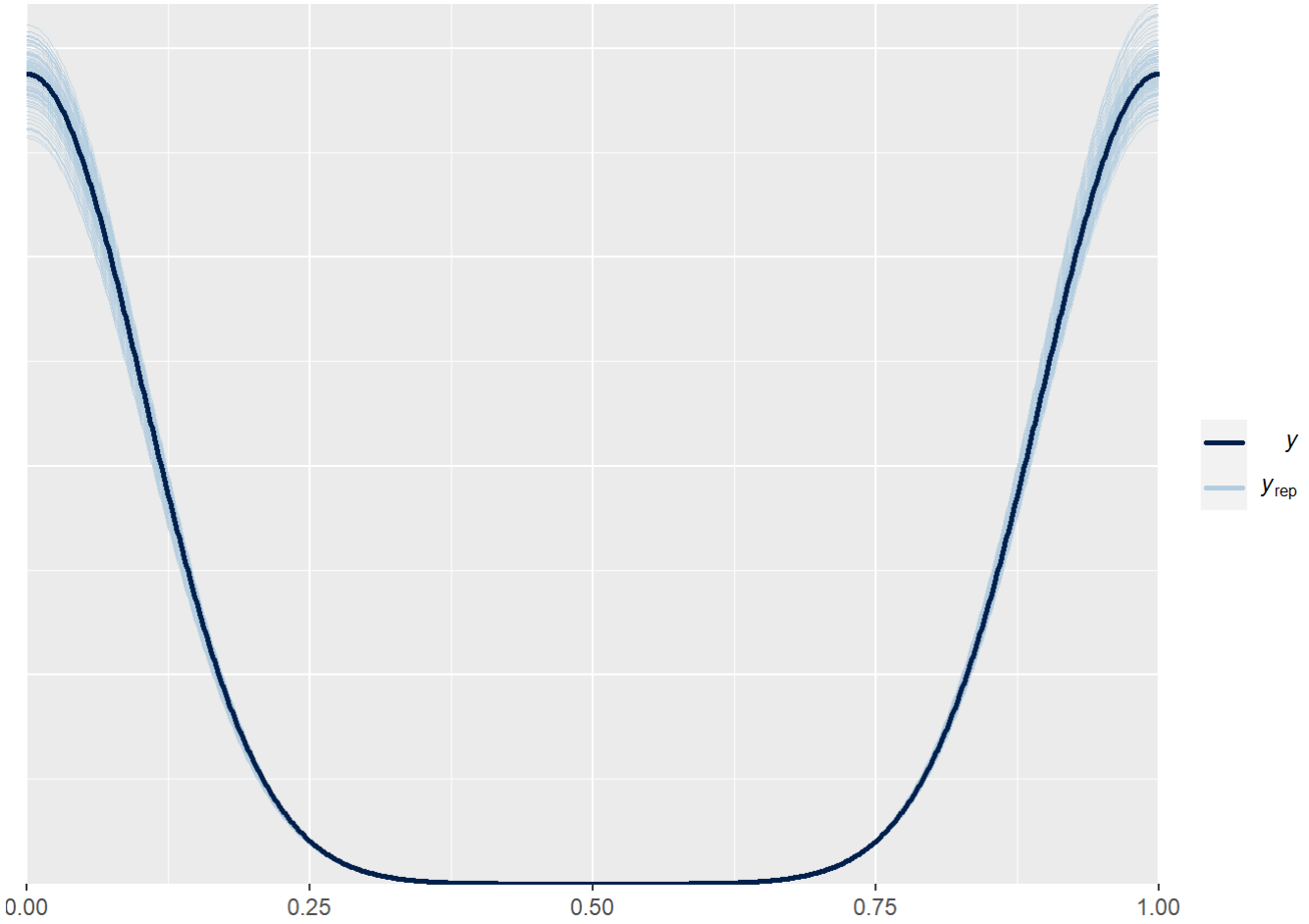

```
#informed model
pitch_f0 <- bf(Condition ~ 1 + v1 + v2 + V3 + v4 + v5 + v6 + v7 + v8 + v9 + v10)

get_prior(pitch_f0, informed_train_s, family = bernoulli)
```

```
##                 prior     class coef group resp dpar nlpar lb ub       source
##                (flat)         b                                       default
##                (flat)         b   v1                                (vectorized)
##                (flat)         b  v10                                (vectorized)
##                (flat)         b   v2                                (vectorized)
##                (flat)         b   V3                                (vectorized)
##                (flat)         b   v4                                (vectorized)
##                (flat)         b   v5                                (vectorized)
##                (flat)         b   v6                                (vectorized)
##                (flat)         b   v7                                (vectorized)
##                (flat)         b   v8                                (vectorized)
##                (flat)         b   v9                                (vectorized)
##  student_t(3, 0, 2.5) Intercept                                         default
```

```
pitch_p0 <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 0.3), class = b))

pitch_m0 <- brm(
  pitch_f0,
  informed_train_s,
  family = bernoulli,
  prior = pitch_p0,
  sample_prior = T,
  backend = "cmdstanr",
  chains = 2,
  cores = 8,
  threads = threading(2),
  control = list(adapt_delta = 0.9,
                 max_treedepth = 20),
  stan_model_args = list(stanc_options = list("O1"))
)
```

```
## Running MCMC with 2 chains, at most 8 in parallel, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 0.8 seconds.
## Chain 2 finished in 0.8 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.8 seconds.
## Total execution time: 1.0 seconds.
```

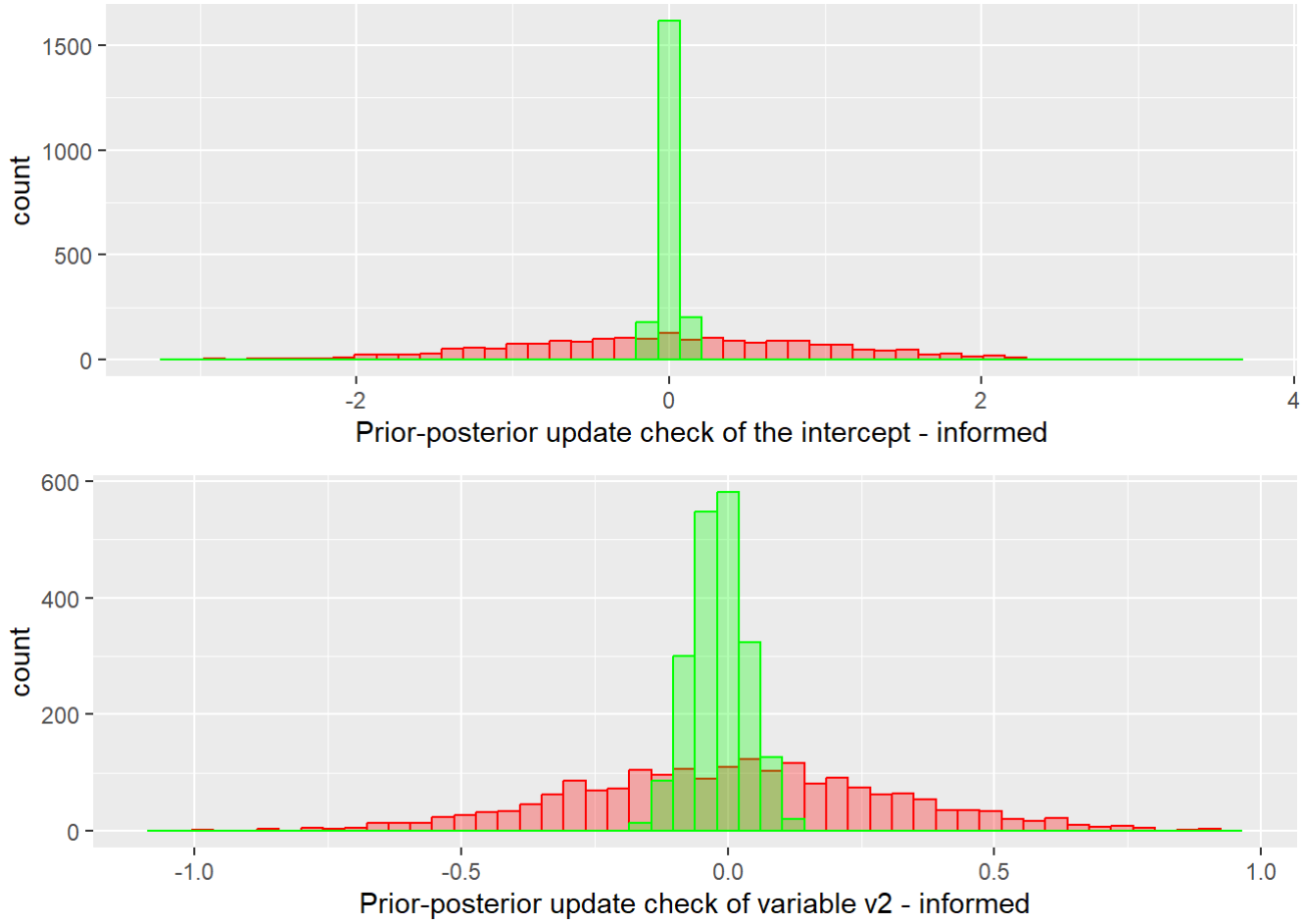
```
pp_check(pitch_m0, ndraws=100)
```




```
summary(pitch_m0)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: Condition ~ 1 + v1 + v2 + V3 + v4 + v5 + v6 + v7 + v8 + v9 + v10
##    Data: informed_train_s (Number of observations: 1600)
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.00      0.05    -0.10     0.11 1.00     3058     1376
## v1            0.01      0.05    -0.10     0.11 1.00     2534     1633
## v2           -0.02      0.05    -0.12     0.09 1.00     4206     1477
## V3            0.07      0.05    -0.04     0.17 1.00     4484     1443
## v4           -0.01      0.05    -0.12     0.09 1.00     3593     1677
## v5           -0.27      0.05    -0.38    -0.16 1.00     3773     1537
## v6           -0.55      0.06    -0.66    -0.43 1.00     4090     1411
## v7           -0.01      0.05    -0.11     0.09 1.00     3380     1381
## v8           -0.18      0.06    -0.28    -0.07 1.00     3256     1329
## v9            0.26      0.05     0.16     0.36 1.00     3503     1609
## v10          -0.38      0.05    -0.49    -0.27 1.00     4033     1407
##
## Draws were sampled using sample(hmc). 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).
```

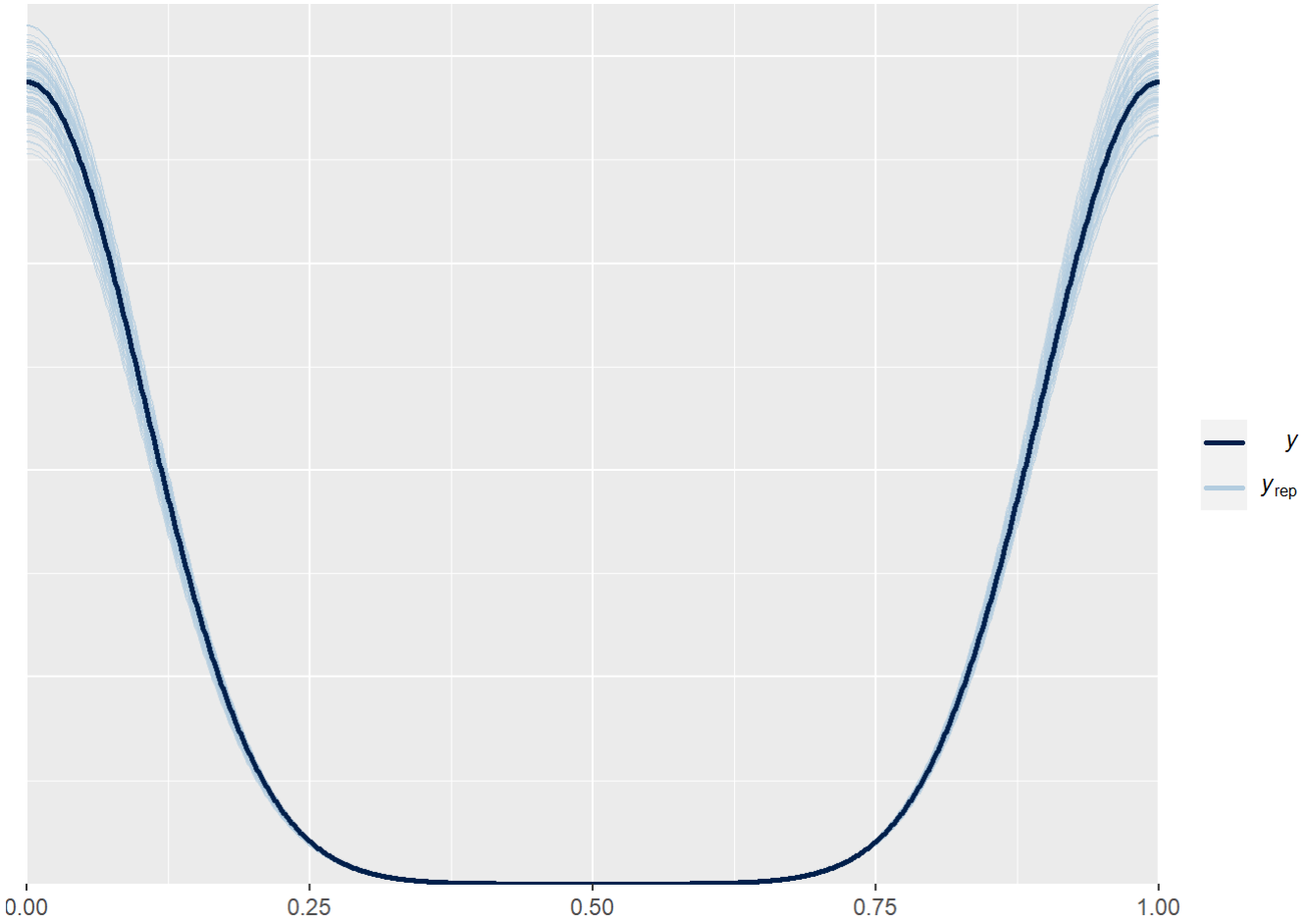
```
posterior <- as_draws_df(pitch_m0)

p1 <- ggplot(posterior) + geom_histogram(aes(prior_Intercept), fill = "red", color = "red", a
lpha = 0.3, bins = 50) + geom_histogram(aes(b_Intercept), fill = "green", color = "green", al
pha = 0.3, bins = 50) + xlab("Prior-posterior update check of the intercept - informed")

p2 <- ggplot(posterior) + geom_histogram(aes(prior_b), fill = "red", color = "red", alpha =
0.3, bins = 50) + geom_histogram(aes(b_v2), fill = "green", color = "green", alpha = 0.3, bin
s = 50) + xlab("Prior-posterior update check of variable v2 - informed")

gridExtra::grid.arrange(p1, p2)
```



```
#Skeptic model
pitch_m0_s <- brm(
  pitch_f0,
  skeptic_train_s,
  family = bernoulli,
  prior = pitch_p0,
  sample_prior = T,
  backend = "cmdstanr",
  chains = 2,
  cores = 8,
  threads = threading(2),
  control = list(adapt_delta = 0.9,
                 max_treedepth = 20),
  stan_model_args = list(stanc_options = list("O1"))
)
```

```
## Running MCMC with 2 chains, at most 8 in parallel, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 finished in 0.8 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.9 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.9 seconds.
## Total execution time: 1.0 seconds.
```

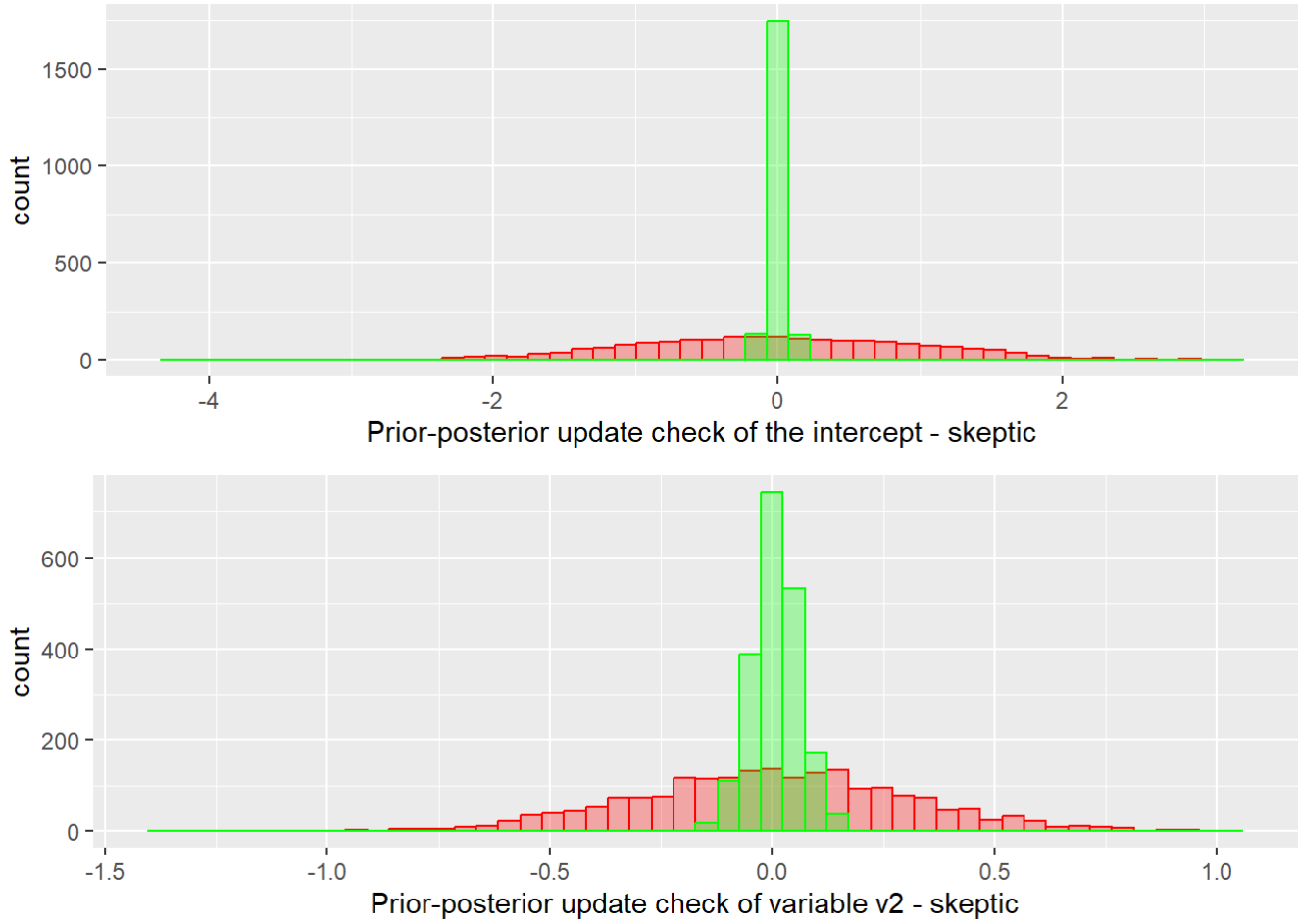
```
pp_check(pitch_m0_s, ndraws=100)
```




```
summary(pitch_m0_s)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: Condition ~ 1 + v1 + v2 + V3 + v4 + v5 + v6 + v7 + v8 + v9 + v10
##    Data: skeptic_train_s (Number of observations: 1600)
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.00      0.05    -0.10     0.10 1.00     5034     1359
## v1           -0.00      0.05    -0.10     0.09 1.00     4562     1383
## v2            0.01      0.05    -0.10     0.12 1.00     4259     1303
## V3            0.02      0.05    -0.08     0.11 1.00     4103     1398
## v4            0.01      0.05    -0.09     0.11 1.00     3822     1304
## v5           -0.10      0.05    -0.19     0.00 1.00     5430     1462
## v6           -0.05      0.05    -0.15     0.05 1.00     3880     1591
## v7            0.03      0.05    -0.07     0.13 1.00     5763     1374
## v8           -0.01      0.05    -0.11     0.08 1.00     3793     1698
## v9            0.02      0.05    -0.07     0.11 1.00     4485     1687
## v10          -0.03      0.05    -0.13     0.07 1.00     5533     1236
##
## Draws were sampled using sample(hmc). 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 <- as_draws_df(pitch_m0_s)

p1s <- ggplot(posterior) + geom_histogram(aes(prior_Intercept), fill = "red", color = "red", alpha = 0.3, bins = 50) + geom_histogram(aes(b_Intercept), fill = "green", color = "green", alpha = 0.3, bins = 50) + xlab("Prior-posterior update check of the intercept - skeptic")

p2s <- ggplot(posterior) + geom_histogram(aes(prior_b), fill = "red", color = "red", alpha = 0.3, bins = 50) + geom_histogram(aes(b_v2), fill = "green", color = "green", alpha = 0.3, bins = 50) + xlab("Prior-posterior update check of variable v2 - skeptic")

gridExtra::grid.arrange(p1s, p2s)
```



iv) assess performance on the test set (NV)

```
# AVERAGE CONFUSION MATRIX PREDICTIONS
# informed test accuracy
informed_test_s1 <- informed_test_s
informed_test_s1$PredictionPerc0 <- predict(pitch_m0, newdata = informed_test_s1, allow_new_l
evels = T)[, 1]
informed_test_s1$Predictions0[informed_test_s1$PredictionPerc0 > 0.5] <- "Schizophrenia"
informed_test_s1$Predictions0[informed_test_s1$PredictionPerc0 <= 0.5] <- "Control"

informed_test_s1 <- informed_test_s1 %>%
  mutate(Condition =
    ifelse(Condition == "1", "Schizophrenia", "Control"),
    Condition = as.factor(Condition),
    Predictions0 = as.factor(Predictions0)
  )
```

```
#Skeptic test accuracy
skeptic_test_s1 <- skeptic_test_s
skeptic_test_s1$PredictionPerc0 <- predict(pitch_m0_s, newdata = skeptic_test_s1, allow_new_l
evels = T)[, 1]
skeptic_test_s1$Predictions0[skeptic_test_s1$PredictionPerc0 > 0.5] <- "Schizophrenia"
skeptic_test_s1$Predictions0[skeptic_test_s1$PredictionPerc0 <= 0.5] <- "Control"

skeptic_test_s1 <- skeptic_test_s1 %>%
  mutate(Condition =
    ifelse(Condition == "1", "Schizophrenia", "Control"),
    Condition = as.factor(Condition),
    Predictions0 = as.factor(Predictions0)
  )
```

```
# Confusion matrices
confusionMatrix(data = informed_test_s1$Predictions0,
                reference = informed_test_s1$Condition)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           122            57
##   Schizophrenia      78           143
##
##                Accuracy : 0.6625
##                  95% CI : (0.6138, 0.7087)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 3.84e-11
##
##                   Kappa : 0.325
##
##  Mcnemar's Test P-Value : 0.08519
##
##             Sensitivity : 0.6100
##             Specificity : 0.7150
##          Pos Pred Value : 0.6816
##          Neg Pred Value : 0.6471
##              Prevalence : 0.5000
##          Detection Rate : 0.3050
##    Detection Prevalence : 0.4475
##       Balanced Accuracy : 0.6625
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(data = skeptic_test_s1$Predictions0,
                reference = skeptic_test_s1$Condition)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           104           104
##   Schizophrenia      96            96
##
##                Accuracy : 0.5
##                  95% CI : (0.4499, 0.5501)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.5199
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.6206
##
##             Sensitivity : 0.52
##             Specificity : 0.48
##          Pos Pred Value : 0.50
##          Neg Pred Value : 0.50
##              Prevalence : 0.50
##          Detection Rate : 0.26
##    Detection Prevalence : 0.52
##       Balanced Accuracy : 0.50
##
##        'Positive' Class : Control
##
```

v) discuss whether performance is as expected and feature importance is as expected (EL & LL)

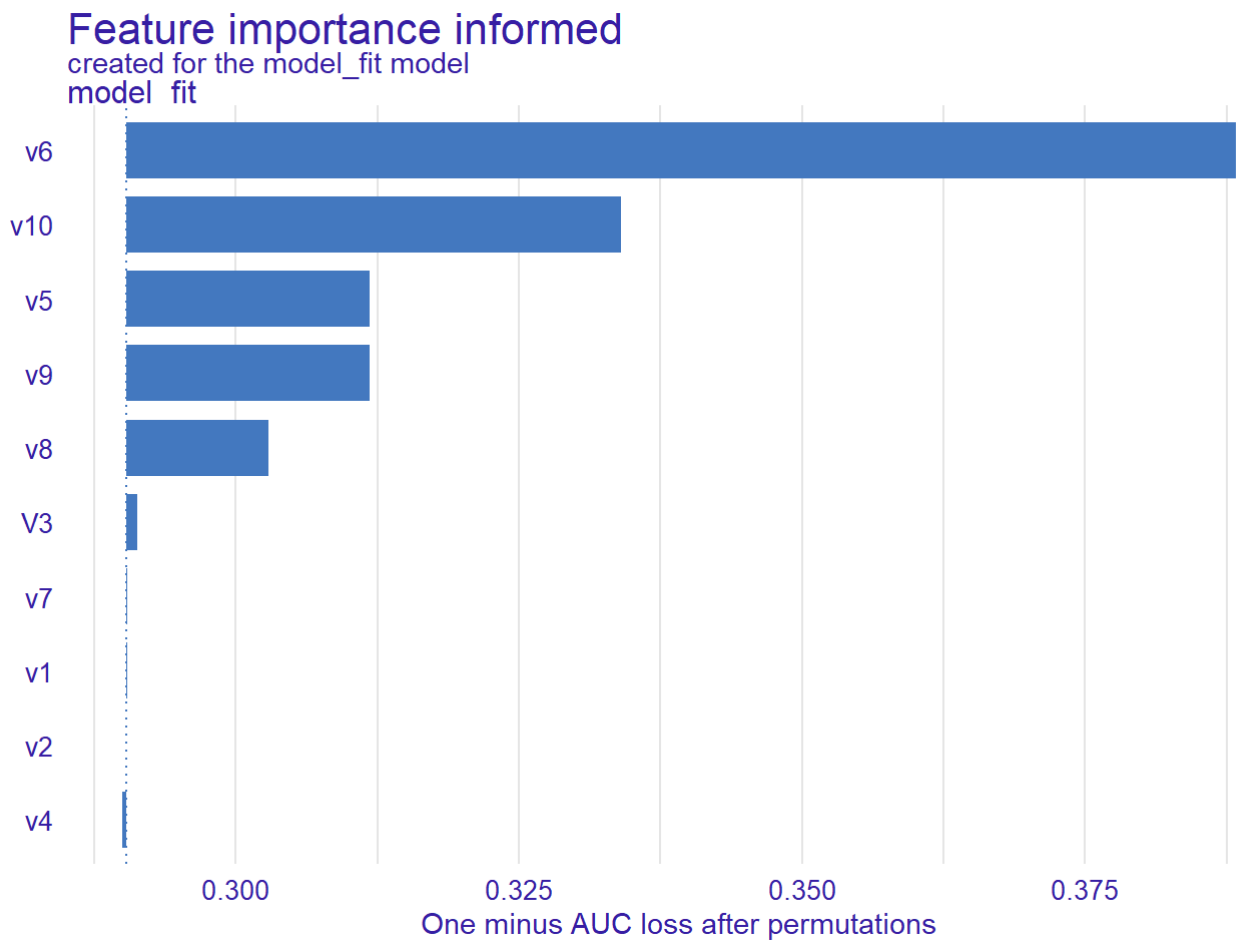
The performance for the skeptic classification model is 48 %, which is below chance. This makes sense since we have created the skeptic data set to not be able to predict anything (All variables have a mean of 0 - basically just noise).

The performance for the informed classification model is 67 %. This makes sense since the informed data set has informed 6 variables that should be able to predict Schizophrenia

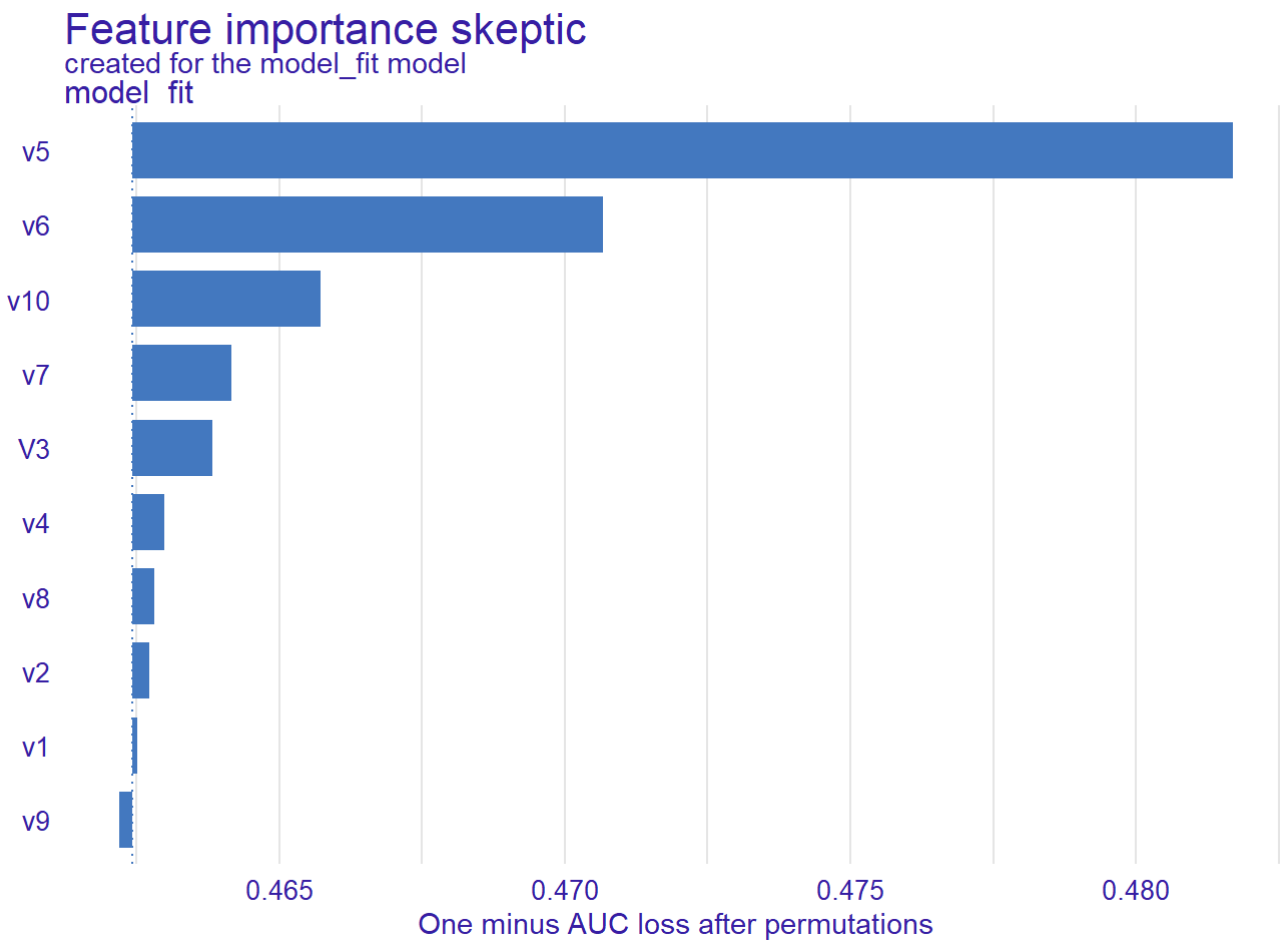
Feature selection

```
# excluding variables from the informed training set
d_inf <- informed_train_s %>%
  mutate(ID = NULL, Trial = NULL, Condition_Schizophrenia = NULL)

d_skep <- skeptic_train_s %>%
  mutate(ID = NULL, Trial = NULL, Condition_Schizophrenia = NULL)

# setting the model type and fitting the model with tidymodels
LogisticRegression_inf <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm") %>%
  fit(Condition ~., data = d_inf)

LogisticRegression_skep <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm") %>%
  fit(Condition ~., data = d_skep)

# using the model fit on the data with all information
explainer_lm <- explain_tidymodels(
  LogisticRegression_inf,
  data = informed_train_s,
  y = as.numeric(informed_train$Condition) -1,
  lable = "logReg",
  verbose = FALSE
)

explainer_lm_skep <- explain_tidymodels(
  LogisticRegression_skep,
  data = skeptic_train_s,
  y = as.numeric(skeptic_train_s$Condition) -1,
  lable = "logReg",
  verbose = FALSE
)
```

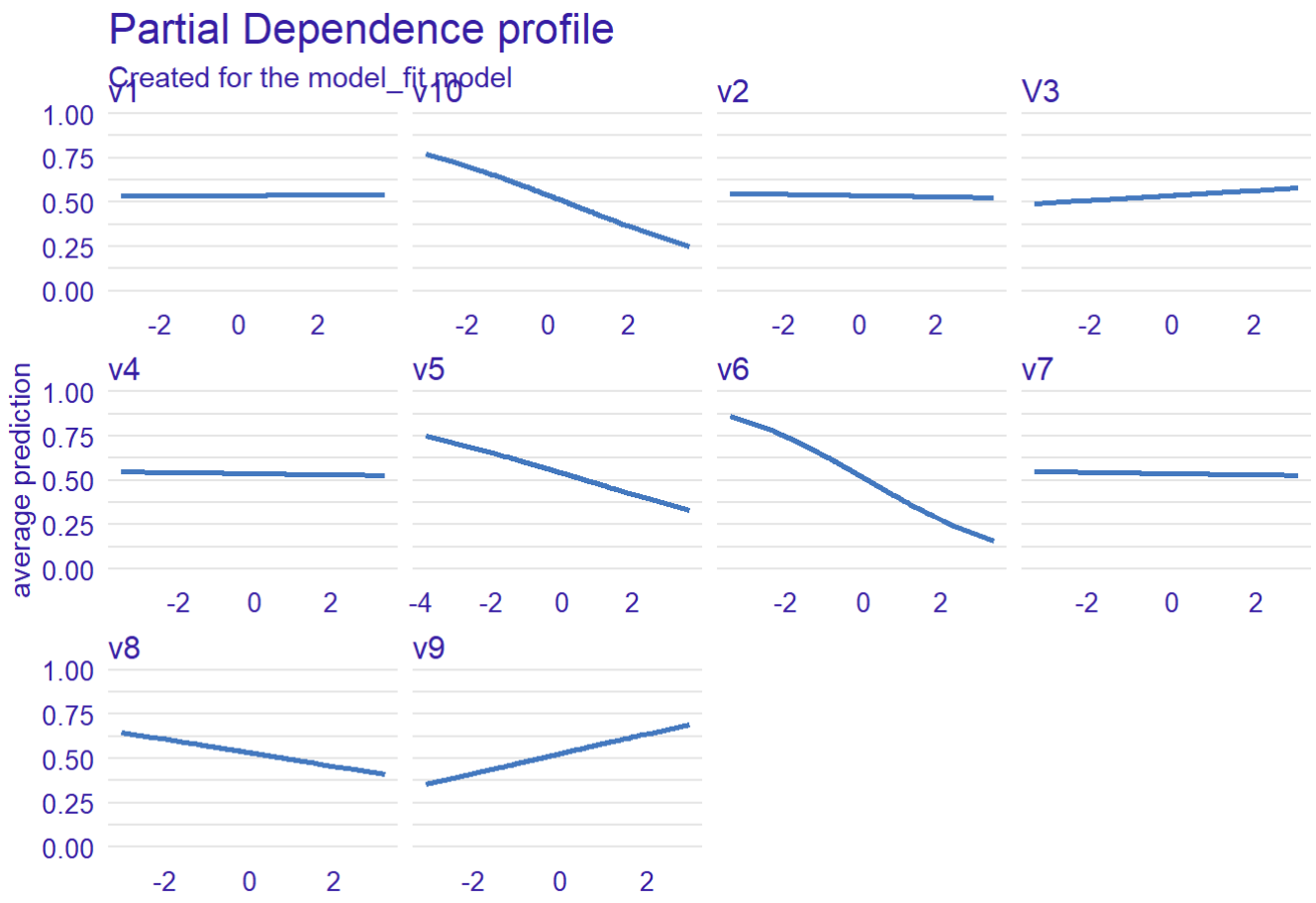
```
# plotting feature importance
set.seed(123)
# One for informed
explainer_lm %>% model_parts() %>%
  filter(variable != "pair" & # Discard the mentioned variables
           variable != "Trial" &
           variable != "ID" &
           variable != "Condition_Schizophrenia" &
           variable != "Condition") %>%
  plot(show_boxplots = FALSE) + ggtitle("Feature importance informed")
```



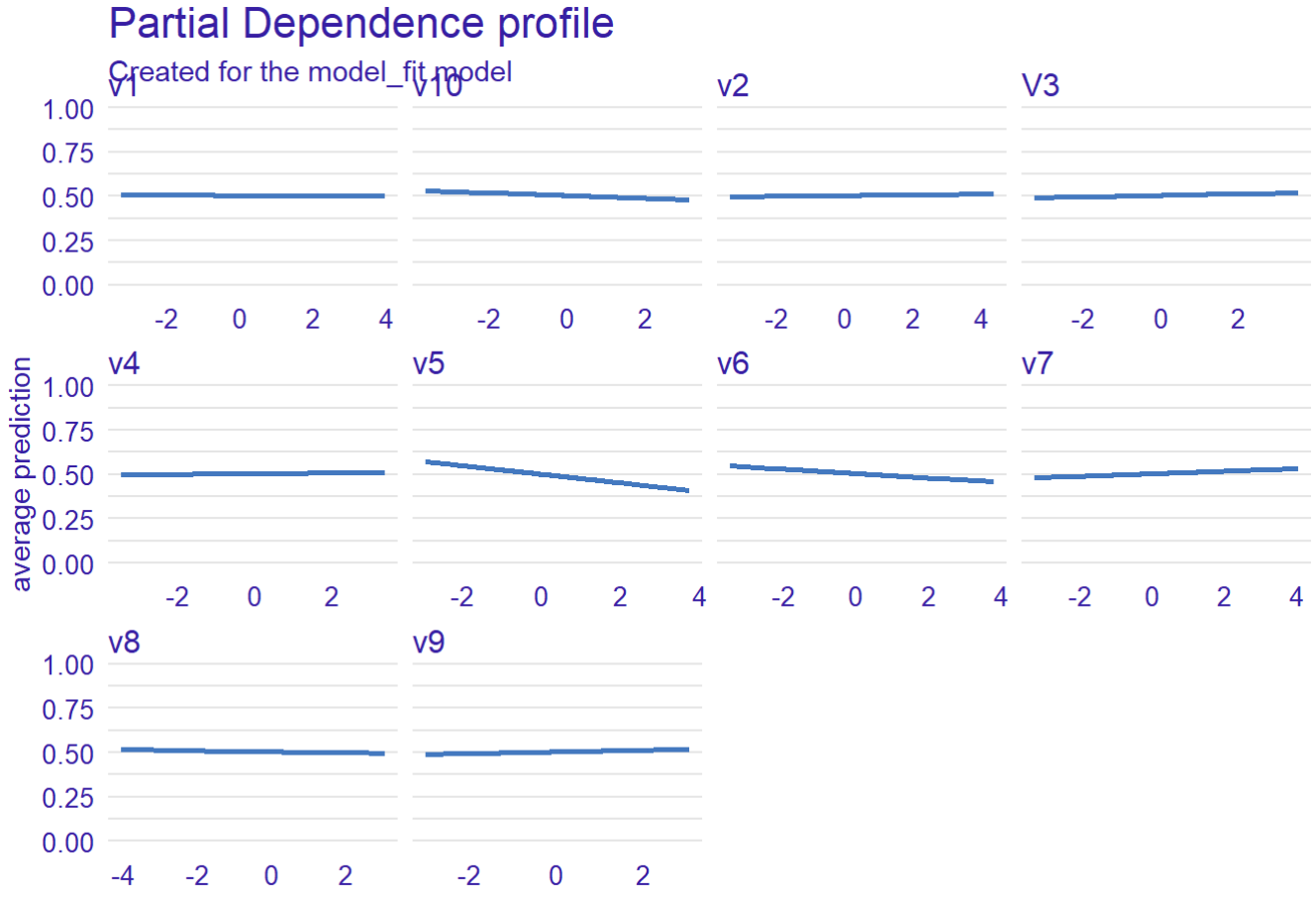

```
# One for skeptic
explainer_lm_skep %>% model_parts() %>%
  filter(variable != "pair" &
           variable != "Trial" &
           variable != "ID" &
           variable != "Condition_Schizophrenia" &
           variable != "Condition") %>%
  plot(show_boxplots = FALSE) + ggtitle("Feature importance skeptic")
```




```
#Plotting the slope for each variable according to how much it predicts Schizophrenia
model_profile_lm1 <- model_profile(explainer_lm, type = "partial", variables = c("v1", "v2",
"V3", "v4", "v5", "v6", "v7", "v8", "v9", "v10"))

model_profile_lm1_skep <- model_profile(explainer_lm_skep, type = "partial", variables = c("v
1", "v2", "V3", "v4", "v5", "v6", "v7", "v8", "v9", "v10"))

plot(model_profile_lm1, variables = c("v1", "v2", "V3", "v4", "v5", "v6", "v7", "v8", "v9",
"v10")) + ylim(0,1)
```




```
plot(model_profile_lm1_skep, variables = c("v1", "v2", "V3", "v4", "v5", "v6", "v7", "v8", "v
9", "v10")) + ylim(0,1)
```



```
# As expected there seems to be 6 variables that predcits Schizophrenia for the informed data set.

# For the skeptic data set we would expect that none of the variables predicted Schizophrenia or that they all predicted Schizophrenia equally. However the variation in how much each variable predicts varies quite a lot.
```

Part III - Applying the ML pipeline to empirical data

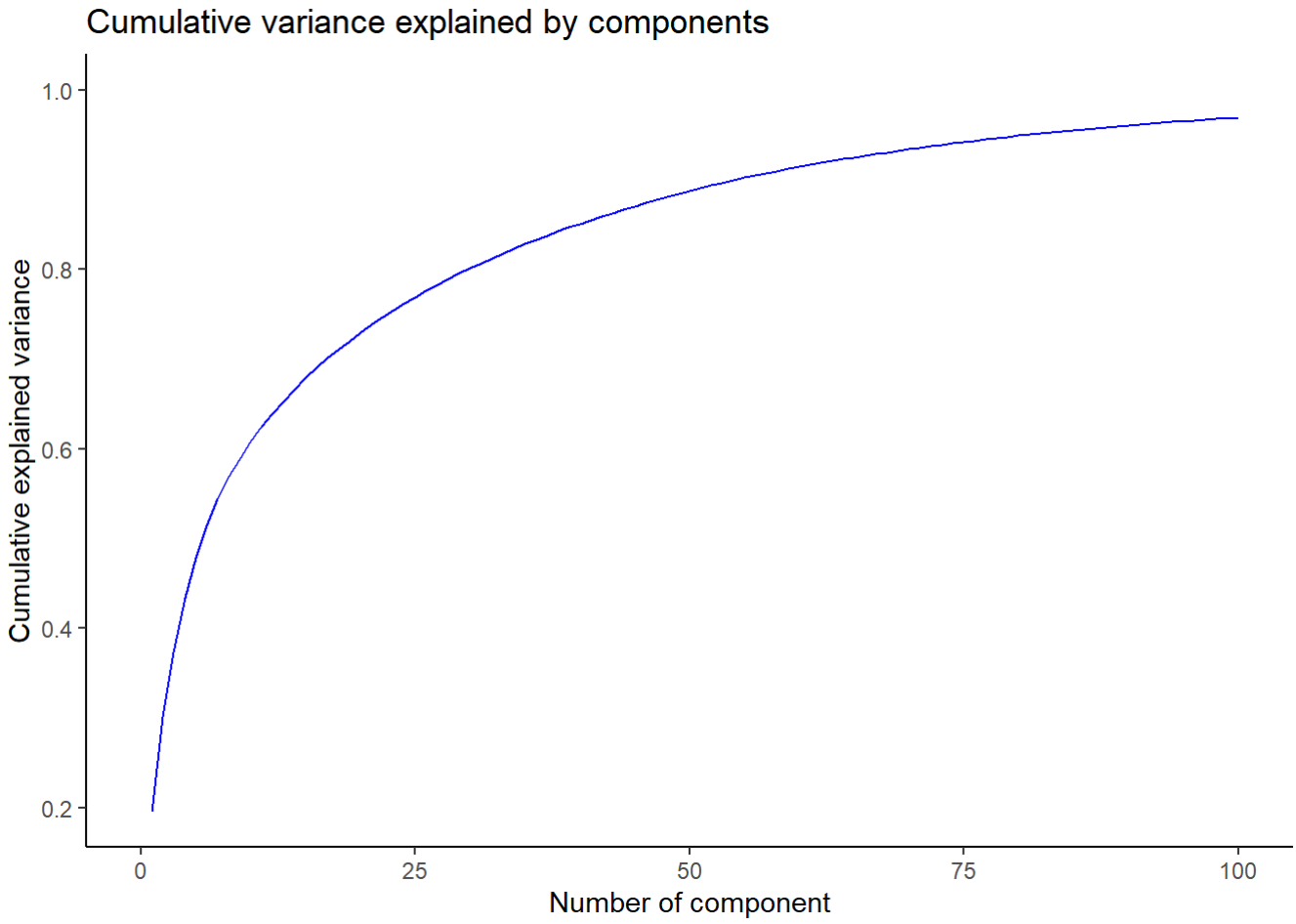
Download the empirical dataset from brightspace and apply your ML pipeline to the new data, adjusting where needed. Warning: in the simulated dataset we only had 10 features, now you have many more! Such is the life of the ML practitioner. Consider the impact a higher number of features will have on your ML inference, and decide whether you need to cut down the number of features before running the pipeline (or alternatively expand the pipeline to add feature selection).

```
d <- read_csv("Ass3_empiricalData1.csv")

# Only selecting the numeric variables, and adding Diagnosis afterwards.
d_num <- select_if(d, is.numeric)
d_num$Condition <- d$Diagnosis

#Splitting the data set into train and test
sample_emp <- sample(seq(100,448), 80)

train_empirical <- subset(d_num, PatID %in% sample_emp)

test_empirical <- subset(d_num, !PatID %in% sample_emp)
```

Cumulative variance explained by the number of components

(NV)

```
# preparing the recipe for
rec_PCA <- recipe(~., data = train_empirical) %>%
  update_role(PatID, new_role = "ID") %>%
  step_normalize(all_numeric_predictors()) %>%
  step_pca(all_numeric_predictors(), threshold = 1) %>%
  prep(training = train_empirical, retain = T)

# PLOTTING THE CUMULATIVE VARIANCE EXPLAINED BY NUMBER OF COMPONENTS
# extracting variance explained by the different components
PCA_variance <- as.tibble(t(summary(rec_PCA$steps[[2]]$res)$importance))

# adding number of PCA column
PCA_variance <- rowid_to_column(PCA_variance, "Number of component")

# plotting the cumulative variance explained
PCA_variance %>%
  ggplot(aes(x = `Number of component`, y = `Cumulative Proportion`)) +
  geom_line(col = "blue") + theme_classic() + xlim(0, 100) +
  labs(title = "Cumulative variance explained by components")+
  ylab("Cumulative explained variance")
```




```
# it looks like most of the variance is captured by the first 9 components
```

Principal compenent analysis (NV & EL)

Model with 3 PCAs

```
# Setting up a recipe using 3 PCAs
rec_PCA3 <- recipe(~., data = train_empirical) %>%
  update_role(PatID, new_role = "ID") %>%
  step_normalize(all_numeric_predictors()) %>%
  step_pca(all_numeric_predictors(), num_comp = 3) %>%
  prep(training = train_empirical, retain = T)

# Prepping the data frame, which is needed to move on
pca_rec_train3 <- juice(rec_PCA3)
pca_rec_test3 <- bake(rec_PCA3, new_data = test_empirical)

# Applying the model
real_f0 <- bf(Condition ~ 1 + .)

real_p0 <- c(
  prior(normal(0, 1), class = Intercept),
  prior(normal(0, 0.3), class = b))

real_m0_3 <- brm(
  real_f0,
  pca_rec_train3,
  family = bernoulli,
  prior = real_p0,
  sample_prior = T,
  backend = "cmdstanr",
  chains = 2,
  cores = 8,
  threads = threading(2),
  control = list(adapt_delta = 0.9,
                 max_treedepth = 20),
  stan_model_args = list(stanc_options = list("O1")))
```

```
## Running MCMC with 2 chains, at most 8 in parallel, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 finished in 0.8 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.8 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.8 seconds.
## Total execution time: 0.9 seconds.
```

Model with 6 PCAs

```
rec_PCA6 <- recipe(~., data = train_empirical) %>%
  update_role(PatID, new_role = "ID") %>%
  step_normalize(all_numeric_predictors()) %>%
  step_pca(all_numeric_predictors(), num_comp = 6) %>%
  prep(training = train_empirical, retain = T)

# Prepping the data frame, which is needed to move on
pca_rec_train6 <- juice(rec_PCA6)
pca_rec_test6 <- bake(rec_PCA6, new_data = test_empirical)

# Applying the model
real_m0_6 <- update(
  real_m0_3, newdata = pca_rec_train6)
```

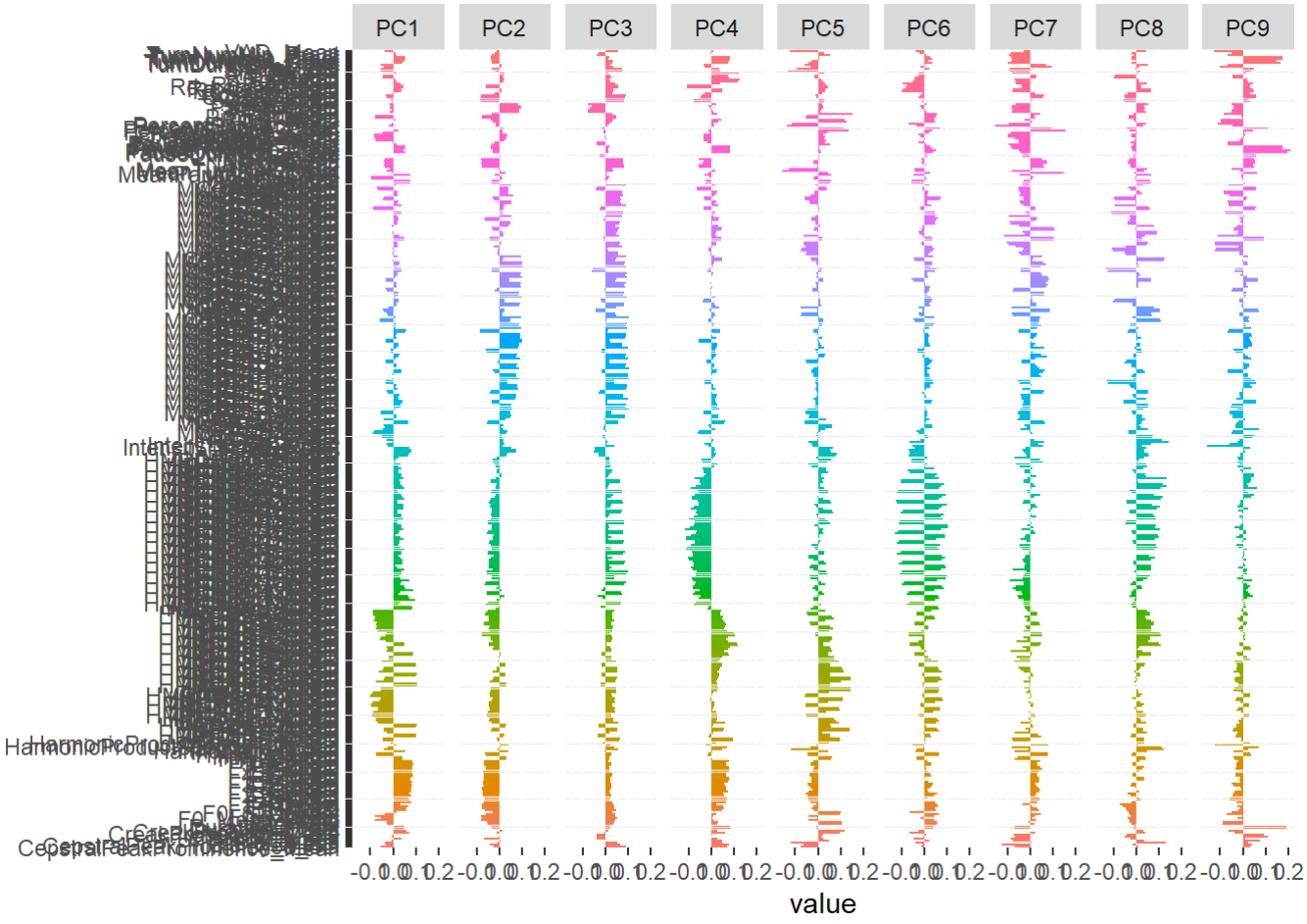
```
## Running MCMC with 2 sequential chains, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 0.9 seconds.
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.8 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.8 seconds.
## Total execution time: 1.9 seconds.
```

MODEL WITH 9 PCA

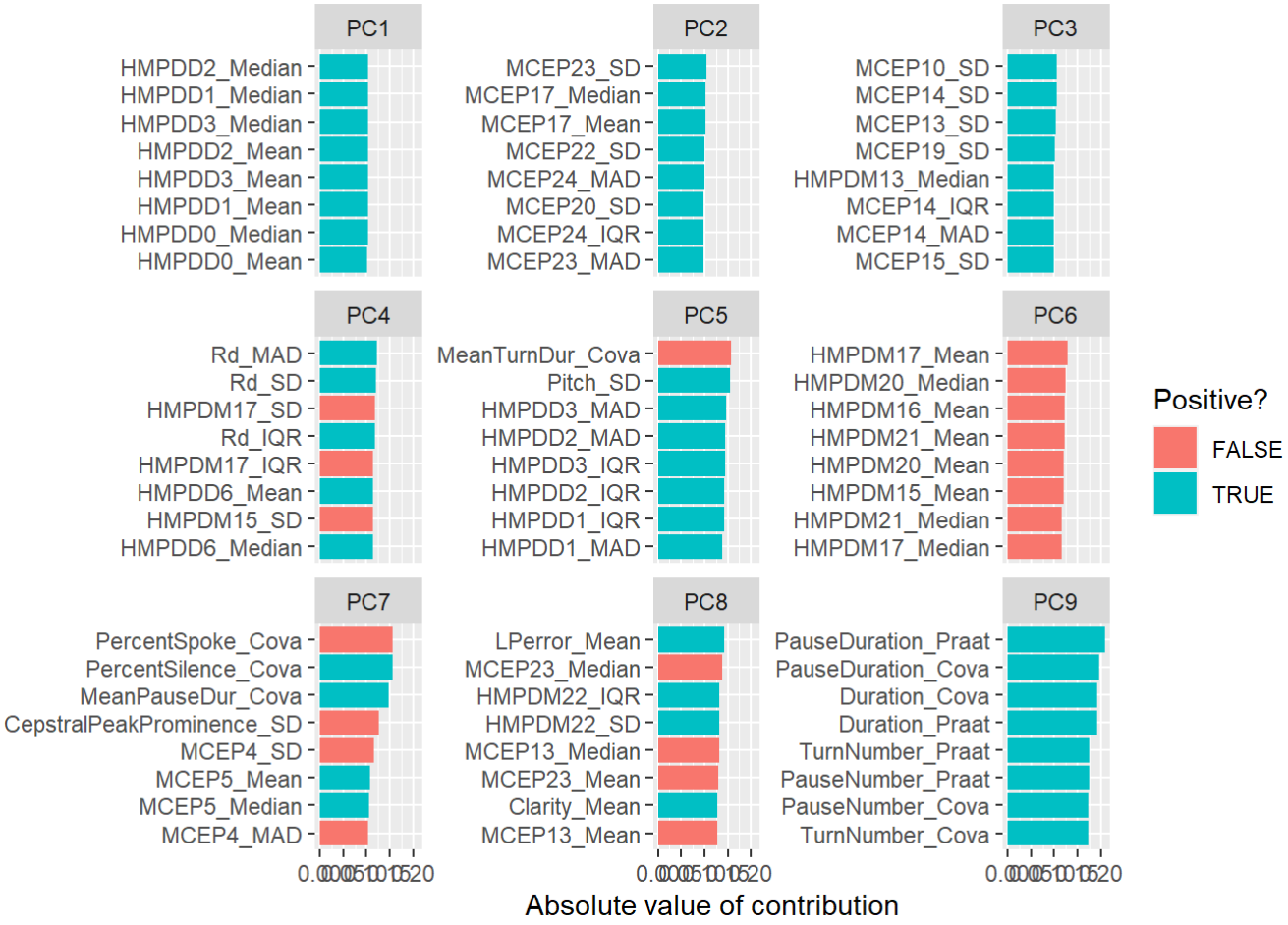
```
rec_PCA9 <- recipe(~., data = train_empirical) %>%
  update_role(PatID, new_role = "ID") %>%
  step_normalize(all_numeric_predictors()) %>%
  step_pca(all_numeric_predictors(), num_comp = 9) %>%
  prep(training = train_empirical, retain = T)

# Prepping the data frame, which is needed to move on
pca_rec_train9 <- juice(rec_PCA9)
pca_rec_test9 <- bake(rec_PCA9, new_data = test_empirical)

# Applying the model
real_m0_9 <- update(
  real_m0_3, newdata = pca_rec_train9)
```

```
## Running MCMC with 2 sequential chains, with 2 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 0.9 seconds.
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.9 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.9 seconds.
## Total execution time: 2.1 seconds.
```

```
# Expanding the recipe into a data frame, which is needed for it to be plotted below:
tidied_pca <- tidy(rec_PCA9, 2)

# Exploring results of PCA. These look at the top 9 components.
tidied_pca %>%
  filter(component %in% paste0("PC", 1:9)) %>%
  mutate(component = fct_inorder(component)) %>%
  ggplot(aes(value, terms, fill = terms)) +
  geom_col(show.legend = F) +
  facet_wrap(~ component, nrow = 1) +
  labs(y = NULL)
```



```
# Ranked results of PCA. Looks at the five most influencial components, and inside these it r
anks the most influential variables (If they are blue they are positive for the component, re
d negative.)
tidied_pca %>%
  filter(component %in% paste0("PC", 1:9)) %>%
  group_by(component) %>%
  top_n(8, abs(value)) %>%
  ungroup() %>%
  mutate(terms = reorder_within(terms, abs(value), component)) %>%
  ggplot(aes(abs(value), terms, fill = value > 0)) +
  geom_col() +
  facet_wrap(~ component, scales = "free_y") +
  scale_y_reordered() +
  labs(
    x = "Absolute value of contribution",
    y = NULL, fill = "Positive?"
  )
```



Evaluation

```
# Evaluation using predictions
# setting up a function for evaluating:
pca_eval<- function(test_data, model){

eval <- test_data
eval$Predictions0 <- NA
eval$PredictionPerc0 <- predict(model, newdata = test_data, allow_new_levels = T)[, 1]
eval$Predictions0[eval$PredictionPerc0 > 0.5] <- "Schizophrenia"
eval$Predictions0[eval$PredictionPerc0 <= 0.5] <- "Control"

eval1 <- eval %>%
  mutate(
    Condition =
      ifelse(Condition=="CT", "Control", "Schizophrenia"),
    Condition = as.factor(Condition),
    Predictions0 = as.factor(Predictions0)

  )
return(eval1)
}

eval_pca3 <- pca_eval(pca_rec_test3, real_m0_3)

eval_pca6 <- pca_eval(pca_rec_test6, real_m0_6)

eval_pca9 <- pca_eval(pca_rec_test9, real_m0_9)
```

Creating some confusion matrices to visualize accuracy of models

```
# setting up a function for confusion matrices
ConfMatrix <- function(testdata){
confusionMatrix(data = testdata$Predictions0,
                reference = testdata$Condition)
}

#3 PCAs
ConfMatrix(eval_pca3)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           612           563
##   Schizophrenia     161           178
##
##                Accuracy : 0.5218
##                  95% CI : (0.4963, 0.5472)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : 0.1982
##
##                   Kappa : 0.0323
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7917
##             Specificity : 0.2402
##          Pos Pred Value : 0.5209
##          Neg Pred Value : 0.5251
##              Prevalence : 0.5106
##          Detection Rate : 0.4042
##    Detection Prevalence : 0.7761
##       Balanced Accuracy : 0.5160
##
##        'Positive' Class : Control
##
```

```
#6 PCAs
ConfMatrix(eval_pca6)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           614           565
##   Schizophrenia     159           176
##
##                Accuracy : 0.5218
##                  95% CI : (0.4963, 0.5472)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : 0.1982
##
##                   Kappa : 0.0322
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7943
##             Specificity : 0.2375
##          Pos Pred Value : 0.5208
##          Neg Pred Value : 0.5254
##              Prevalence : 0.5106
##          Detection Rate : 0.4055
##    Detection Prevalence : 0.7787
##       Balanced Accuracy : 0.5159
##
##        'Positive' Class : Control
##
```

```
#9 PCAs
ConfMatrix(eval_pca9)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           605           561
##   Schizophrenia     168           180
##
##                Accuracy : 0.5185
##                  95% CI : (0.493, 0.5439)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : 0.2772
##
##                   Kappa : 0.0259
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7827
##             Specificity : 0.2429
##          Pos Pred Value : 0.5189
##          Neg Pred Value : 0.5172
##              Prevalence : 0.5106
##          Detection Rate : 0.3996
##    Detection Prevalence : 0.7701
##       Balanced Accuracy : 0.5128
##
##        'Positive' Class : Control
##
```