final_A3_machine_learning

Studygroup 5

2022-12-01

```
pacman::p_load(msm, tidyverse, brms, grid, gridExtra, readxl, metafor, dplyr, magrittr, reshape, tidymo
```

part 1 - 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.

Simulation for the informed dataset

```
n <- 100
trials <- 10

InformedEffectMean <- c(0.25, -0.55, 0.74, -1.26, 0.05, 1.89, 0, 0, 0, 0)


IndividualSD <-  1
TrialSD <- 0.5
Error <- 0.2


Schizophrenia <- rnorm(1, rnorm(1, 0.21, 0.5), 0.2)
Control <- rnorm(1, rnorm(1, -0.21, 0.5), 0.2)


for(i in seq(10)) {
  temp_informed <- tibble(
    ID = seq(n),
    TrueEffect = rnorm(n, InformedEffectMean[i], IndividualSD),
    Variable = paste0("v", i))
  if(i == 1) {
    d_informed_true <- temp_informed
  } else {
    d_informed_true <- rbind(d_informed_true, temp_informed)
  }
}
```

```
d_trial <- tibble(expand_grid(ID = seq(n), Trial = seq(trials), Group = c("Schizophrenia", "Control")))

d_informed <- merge(d_informed_true, d_trial)

for(i in seq(nrow(d_informed))){
  d_informed$measurement[i] <- ifelse(d_informed$Group[i]=="Schizophrenia",
                                      rnorm(1, rnorm(1, d_informed$TrueEffect[i]/2, TrialSD), Error),
                                      rnorm(1, rnorm(1, -d_informed$TrueEffect[i]/2, TrialSD), Error))
}

d_informed_wide <- d_informed %>%
  mutate(TrueEffect= NULL) %>%
  pivot_wider(names_from = Variable,
              values_from = measurement)
```

Simulation for the skeptic dataset

```
skeptic_EffectMean <- rep(0,10)

for(i in seq(10)) {
  temp_skeptic <- tibble(
    ID = seq(n),
    TrueEffect = rnorm(n, skeptic_EffectMean[i], IndividualSD),
    Variable = paste0("v", i))
  if(i == 1) {
    d_informed_true <- temp_informed
    d_skeptic_true <- temp_skeptic
  } else {
    d_skeptic_true <- rbind(d_skeptic_true, temp_skeptic)
  }
}

d_skeptic <- merge(d_skeptic_true, d_trial)

for(i in seq(nrow(d_skeptic))){
  d_skeptic$measurement[i] <- ifelse(d_skeptic$Group[i]=="Schizophrenia",
                                      rnorm(1, rnorm(1, d_skeptic$TrueEffect[i]/2, TrialSD), Error),
                                      rnorm(1, rnorm(1, -d_skeptic$TrueEffect[i]/2, TrialSD), Error))
}

d_skeptic_wide <- d_skeptic %>%
  mutate(TrueEffect= NULL) %>%
  pivot_wider(names_from = Variable,
              values_from = measurement)
```

##Part 2 - 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.

```
d_informed_wide <- d_informed_wide %>%
  mutate(ID = as.factor(ID)) %>%
  mutate(Trial = as.factor(Trial))

d_skeptic_wide <- d_skeptic_wide %>%
    mutate(ID = as.factor(ID)) %>%
    mutate(Trial = as.factor(Trial))
```

i) create a data budget (e.g. balanced training and test sets);

```
TestID <- sample(seq(n), 20)


train_informed <- d_informed_wide %>% subset(!(ID %in% TestID))
test_informed <- d_informed_wide %>% subset((ID %in% TestID))

train_skeptic <- d_skeptic_wide %>% subset(!(ID %in% TestID))
test_skeptic<- d_skeptic_wide %>% subset((ID %in% TestID))
```

ii) pre-process the data (e.g. scaling the features);

```
scaled_informed <- train_informed %>%
  recipe(Group ~ .) %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep(training = train_informed, retain = TRUE)


train_informed_scaled <- juice(scaled_informed)
test_informed_scaled <- bake(scaled_informed, new_data = test_informed)
```

```
scaled_skeptic <- train_skeptic %>%
  recipe(Group ~ .) %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep(training = train_skeptic, retain = TRUE)


train_skeptic_scaled <- juice(scaled_skeptic)
test_skeptic_scaled <- bake(scaled_skeptic, new_data = test_skeptic)
```

iii) fit and assess a classification algorithm on the training data (e.g. Bayesian multilevel logistic regression);

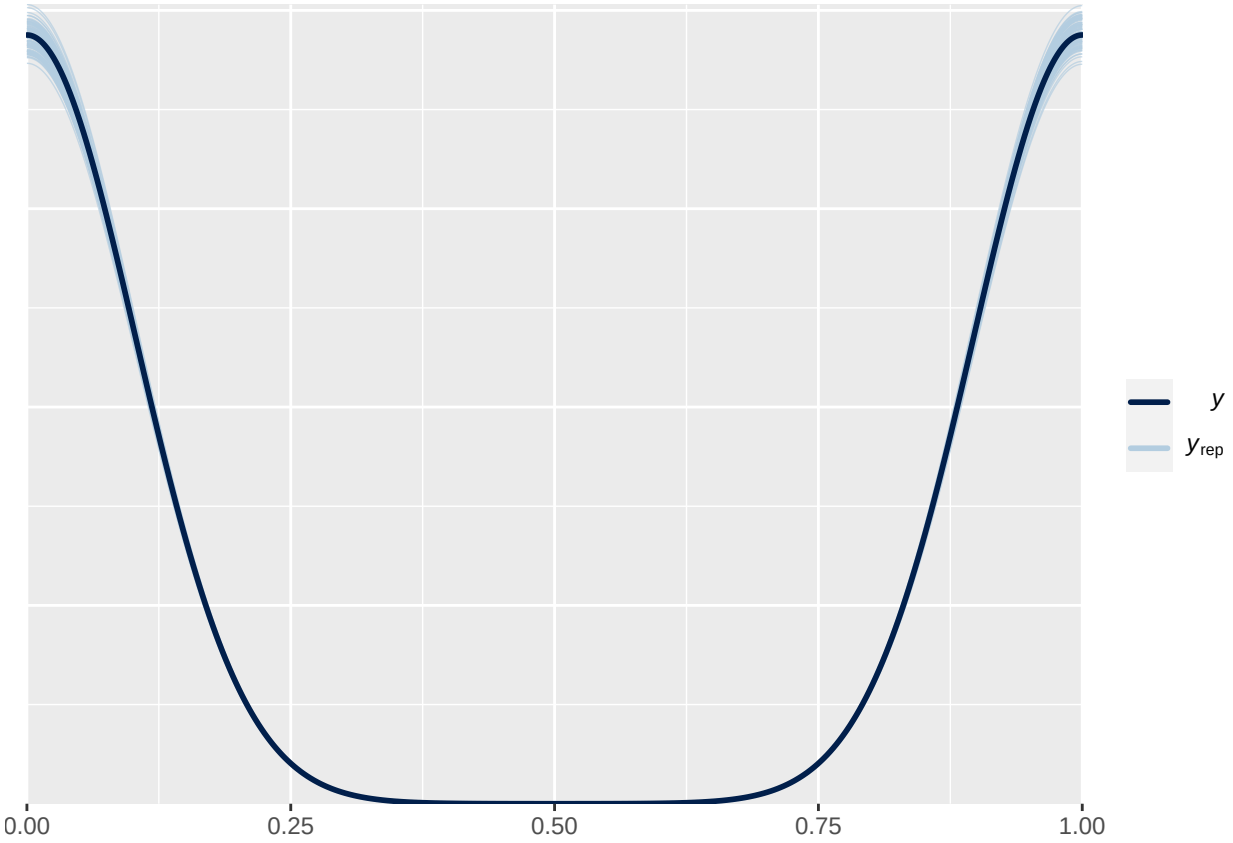
```
f1 <- bf(Group ~ 1 + v1 + v2 + v3 + v4 + v5 +v6 +v7 +v8 +v9 +v10)

s_f1 <- bf(Group ~ 1 + v1 + v2 + v3 + v4 + v5 +v6 +v7 +v8 +v9 +v10)
```

```
get_prior(f1,
    train_informed_scaled,
    family = bernoulli)

get_prior(s_f1,
    train_skeptic_scaled,
    family = bernoulli)
```

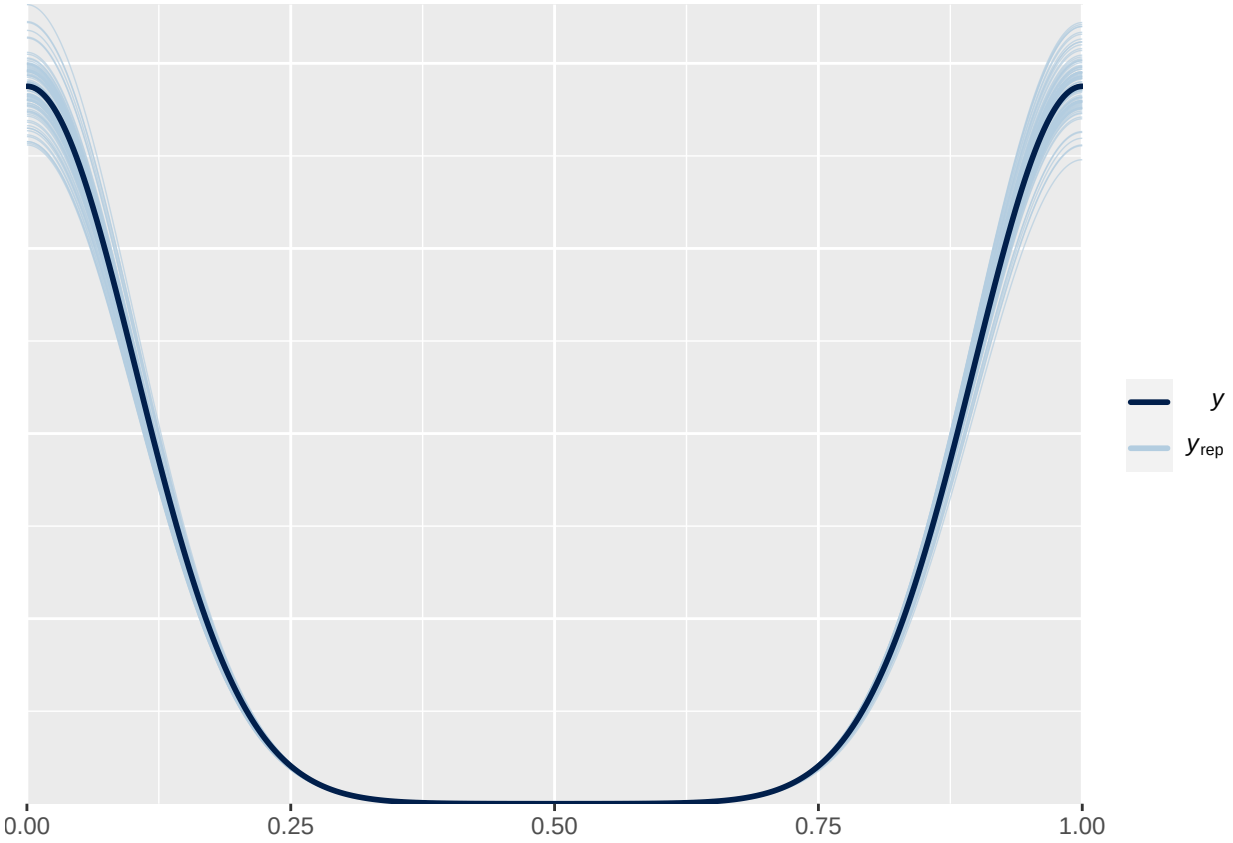
```
f1_prior <- c(
prior(normal(0, 1), class = Intercept),
prior(normal(0, 0.3), class = b)
)

s_f1_prior <- c(
prior(normal(0, 1), class = Intercept),
prior(normal(0, 0.3), class = b)
)
```

```
fitted_f1_prior <- brm(
    f1,
    train_informed_scaled,
    family = bernoulli,
    prior = f1_prior,
    sample_prior = T,
    iter = 4000,
    warmup = 2000,
    cores = 4,
    refresh=0,
    chains = 4,
    control = list(
      adapt_delta = 0.999,
      max_treedepth = 20))


fitted_s_f1_prior <-
  brm(
    s_f1,
    train_skeptic_scaled,
    family = bernoulli,
    prior = s_f1_prior,
    sample_prior = T,
    iter = 4000,
    warmup = 2000,
    cores = 4,
    refresh=0,
    chains = 4,
    control = list(
      adapt_delta = 0.999,
      max_treedepth = 20))
```

```
pp_check(fitted_f1_prior, ndraws = 100)
```




```
pp_check(fitted_s_f1_prior, ndraws = 100)
```




```
print(fitted_f1_prior)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: Group ~ 1 + v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 + v9 + v10
##    Data: train_informed_scaled (Number of observations: 1600)
##   Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup draws = 8000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.01      0.10    -0.19     0.21 1.00     8703     5407
## v1            0.33      0.10     0.14     0.52 1.00     8381     5733
## v2           -0.78      0.11    -0.99    -0.57 1.00     9391     6229
## v3            0.59      0.11     0.39     0.81 1.00     9564     6048
## v4           -1.26      0.12    -1.50    -1.02 1.00     9479     6020
## v5           -0.18      0.11    -0.38     0.03 1.00     9703     6029
## v6            2.91      0.15     2.62     3.22 1.00     8460     5960
## v7           -0.03      0.10    -0.23     0.16 1.00     9506     6043
## v8           -0.13      0.10    -0.32     0.06 1.00     9007     6090
## v9           -0.13      0.11    -0.34     0.08 1.00     8986     6329
## v10           0.11      0.10    -0.08     0.31 1.00     9331     5653
##
## 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).
```

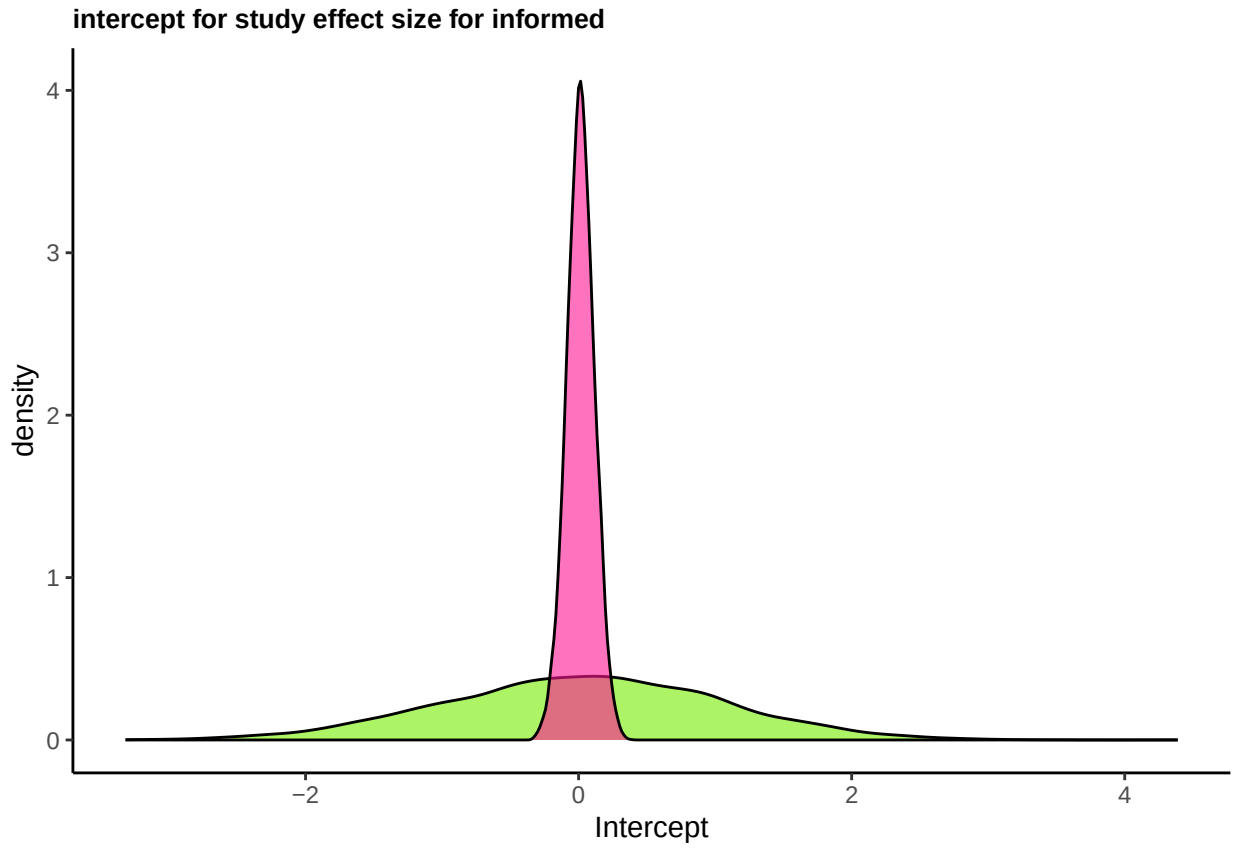
```
print(fitted_s_f1_prior)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: Group ~ 1 + v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 + v9 + v10
##    Data: train_skeptic_scaled (Number of observations: 1600)
##   Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup draws = 8000
##
## 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    11218     5878
## v1            0.30      0.05     0.20     0.41 1.00     9193     5446
## v2           -0.16      0.05    -0.26    -0.06 1.00    10022     5967
## v3           -0.20      0.05    -0.30    -0.09 1.00     9059     5708
## v4           -0.21      0.05    -0.31    -0.11 1.00     9791     5937
## v5           -0.20      0.05    -0.31    -0.10 1.00    10512     5821
## v6           -0.06      0.05    -0.17     0.04 1.00    10072     5987
## v7           -0.11      0.05    -0.22    -0.01 1.00     9996     5673
## v8           -0.07      0.05    -0.18     0.03 1.00     9904     6166
## v9           -0.02      0.05    -0.12     0.09 1.00     9707     5846
## v10          -0.20      0.05    -0.30    -0.10 1.00     9876     5862
##
## 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).
```

```
variables(fitted_f1_prior)
```

```
##  [1] "b_Intercept"     "b_v1"            "b_v2"            "b_v3"
##  [5] "b_v4"            "b_v5"            "b_v6"            "b_v7"
##  [9] "b_v8"            "b_v9"            "b_v10"           "prior_Intercept"
## [13] "prior_b"         "lprior"          "lp__"
```

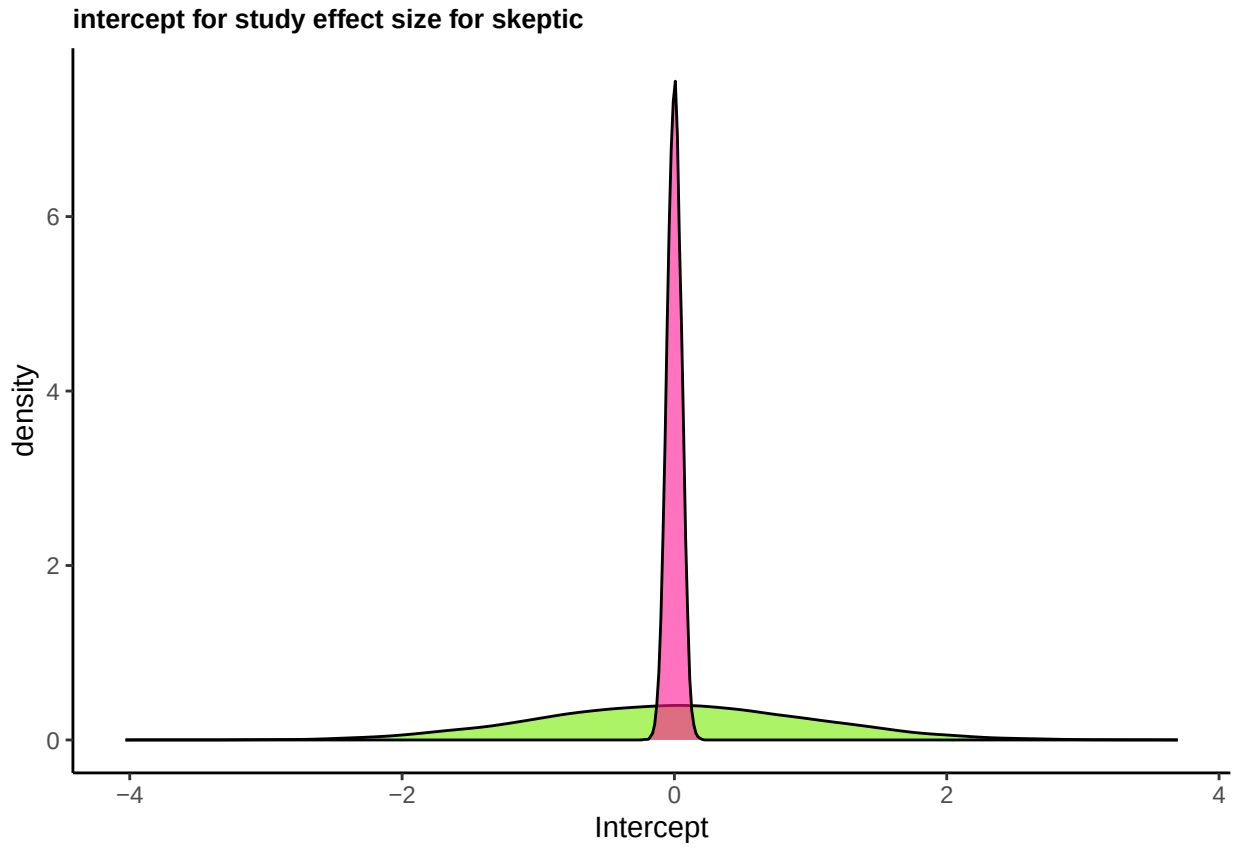
```
Posterior_f1 <- as_draws_df(fitted_f1_prior)

ggplot(Posterior_f1) +
  geom_density(aes(prior_Intercept), fill="chartreuse2", color="black",alpha=0.6) +
  geom_density(aes(b_Intercept), fill="deeppink", color="black",alpha=0.6) +
  xlab('Intercept') +
  theme_classic()+
  ggtitle("intercept for study effect size for informed")+
      theme(plot.title = element_text(size = 10, face = "bold"))
```




```
variables(fitted_s_f1_prior)
```

```
##  [1] "b_Intercept"     "b_v1"            "b_v2"            "b_v3"
##  [5] "b_v4"            "b_v5"            "b_v6"            "b_v7"
##  [9] "b_v8"            "b_v9"            "b_v10"           "prior_Intercept"
## [13] "prior_b"         "lprior"          "lp__"
```

```
Posterior_s_f1 <- as_draws_df(fitted_s_f1_prior)

ggplot(Posterior_s_f1) +
  geom_density(aes(prior_Intercept), fill="chartreuse2", color="black",alpha=0.6) +
  geom_density(aes(b_Intercept), fill="deeppink", color="black",alpha=0.6) +
  xlab('Intercept') +
  theme_classic()+
  ggtitle("intercept for study effect size for skeptic")+
      theme(plot.title = element_text(size = 10, face = "bold"))
```



iv) assess performance

```
PerformanceProb <- tibble(expand_grid(
    Sample = seq(4000),
    Model =c("Our_model"),
    Setup= c("informed", "skeptic"),
    Type= c("training", "test"))
    )


test_informed_scaled$PredictionsPerc0 <- predict(fitted_f1_prior, newdata = test_informed_scaled, allow
test_informed_scaled$Predictions0[test_informed_scaled$PredictionsPerc0 > 0.5] <- "Schizophrenia"
```

```
## Warning: Unknown or uninitialised column: `Predictions0`.
```

```
test_informed_scaled$Predictions0[test_informed_scaled$PredictionsPerc0 <= 0.5] <- "Control"

train_informed_scaled$PredictionsPerc0 <- predict(fitted_f1_prior)[,1]
train_informed_scaled$Predictions0[train_informed_scaled$PredictionsPerc0 > 0.5] <- "Schizophrenia"
```

```
## Warning: Unknown or uninitialised column: `Predictions0`.
```

```
train_informed_scaled$Predictions0[train_informed_scaled$PredictionsPerc0 <= 0.5] <- "Control"


test_skeptic_scaled$PredictionsPerc1 <- predict(fitted_s_f1_prior, newdata = test_skeptic_scaled, allow
test_skeptic_scaled$Predictions1[test_skeptic_scaled$PredictionsPerc1 > 0.5] <- "Schizophrenia"
```

```
## Warning: Unknown or uninitialised column: `Predictions1`.
```

```
test_skeptic_scaled$Predictions1[test_skeptic_scaled$PredictionsPerc1 <= 0.5] <- "Control"

train_skeptic_scaled$PredictionsPerc1 <- predict(fitted_s_f1_prior)[,1]
train_skeptic_scaled$Predictions1[train_skeptic_scaled$PredictionsPerc1 > 0.5] <- "Schizophrenia"
```

```
## Warning: Unknown or uninitialised column: `Predictions1`.
```

```
train_skeptic_scaled$Predictions1[train_skeptic_scaled$PredictionsPerc1 <= 0.5] <- "Control"



train0 <- inv_logit_scaled(posterior_linpred(fitted_f1_prior,
    summary = F))

test0 <- inv_logit_scaled(posterior_linpred(fitted_f1_prior,
    summary = F,
    newdata = test_informed_scaled,
    allow_new_levels = T ))


train1 <- inv_logit_scaled(posterior_linpred(fitted_s_f1_prior,
    summary = F))

test1 <- inv_logit_scaled(posterior_linpred(fitted_s_f1_prior,
    summary = F,
    newdata = test_skeptic_scaled,
    allow_new_levels = T ))


test_informed_scaled <- test_informed_scaled %>%
  mutate(Group = as.factor(Group),
         Predictions0 = as.factor((Predictions0)))

train_informed_scaled <- train_informed_scaled %>%
  mutate(Group = as.factor(Group),
         Predictions0 = as.factor((Predictions0)))


test_skeptic_scaled <- test_skeptic_scaled %>%
  mutate(Group = as.factor(Group),
         Predictions1 = as.factor(Predictions1))

train_skeptic_scaled <- train_skeptic_scaled %>%
  mutate(Group = as.factor(Group),
```

```
        Predictions1 = as.factor(Predictions1))


for (i in seq(4000)){

    train_informed_scaled$Predictions0 <- as.factor(ifelse(train0[i,] > 0.5, "Schizophrenia", "Control")

    test_informed_scaled$Predictions0 <- as.factor(ifelse(test0[i,] > 0.5, "Schizophrenia", "Control"))

    PerformanceProb$Accuracy[PerformanceProb$Sample==i & PerformanceProb$Model == "Our_model" & Performa
    accuracy(train_informed_scaled, truth = Group, estimate = Predictions0)[, ".estimate"]


    PerformanceProb$Accuracy[PerformanceProb$Sample==i & PerformanceProb$Model == "Our_model" & Performa
    accuracy(test_informed_scaled, truth = Group, estimate = Predictions0)[, ".estimate"]

    train_skeptic_scaled$Predictions1 <- as.factor(ifelse(train1[i,] > 0.5, "Schizophrenia", "Control"))
    test_skeptic_scaled$Predictions1 <- as.factor(ifelse(test1[i,] > 0.5, "Schizophrenia", "Control"))

    PerformanceProb$Accuracy[PerformanceProb$Sample == i & PerformanceProb$Model == "Our_model" & Perfo
    accuracy(train_skeptic_scaled, truth = Group, estimate = Predictions1)[, ".estimate"]


    PerformanceProb$Accuracy[PerformanceProb$Sample == i & PerformanceProb$Model == "Our_model" & Perfo
    accuracy(test_skeptic_scaled, truth = Group, estimate = Predictions1)[, ".estimate"]
}
```

```
## Warning: Unknown or uninitialised column: `Accuracy`.
```

Assesing average performance for the test set in the informed simualtion

```
conf_mat(
  test_informed_scaled,
  truth =  Group,
  estimate = Predictions0,
  dnn = c("Prediction", "truth"))
```

```
##                truth
## Prediction      Control Schizophrenia
##   Control           189            11
##   Schizophrenia      11           189
```

```
metrics(
  test_informed_scaled,
  truth =  Group,
  estimate = Predictions0)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.945
## 2 kap      binary         0.89
```

Assesing average performance for the test set in the skeptic simualtion

```
conf_mat(
  test_skeptic_scaled,
  truth =  Group,
  estimate = Predictions1,
  dnn = c("Prediction", "truth"))
```

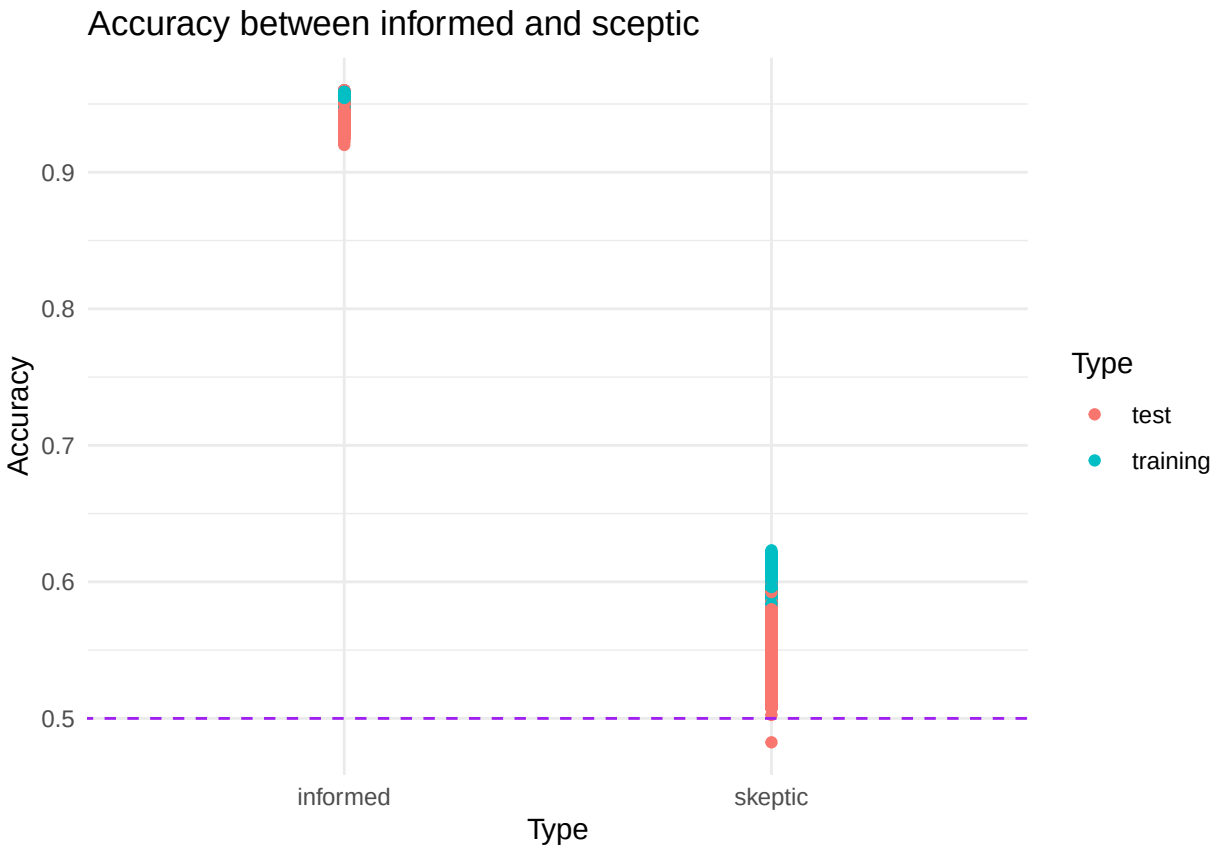
```
##                truth
## Prediction      Control Schizophrenia
##   Control           117            87
##   Schizophrenia      83           113
```

```
metrics(
  test_skeptic_scaled,
  truth =  Group,
  estimate = Predictions1)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.575
## 2 kap      binary         0.15
```

Plotting the accuracy

```
ggplot(PerformanceProb) +
  geom_point(aes(x = Setup, y = as.numeric(Accuracy), colour = Type)) + geom_abline(intercept = 0.5, sl
  theme_minimal() +
  ylab("Accuracy") +
  xlab("Type") +
  theme_minimal() +
  ggtitle("Accuracy between informed and sceptic")
```



Part 3 - 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).

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

```
## Rows: 1889 Columns: 398
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (5): NewID, Diagnosis, Language, Gender, Trial
## dbl (393): PatID, Corpus, Duration_Praat, F0_Mean_Praat, F0_SD_Praat, Intens...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data <- data %>%
  select(-Language) %>%
  select(-NewID) %>%
  select(-Corpus) %>%
```

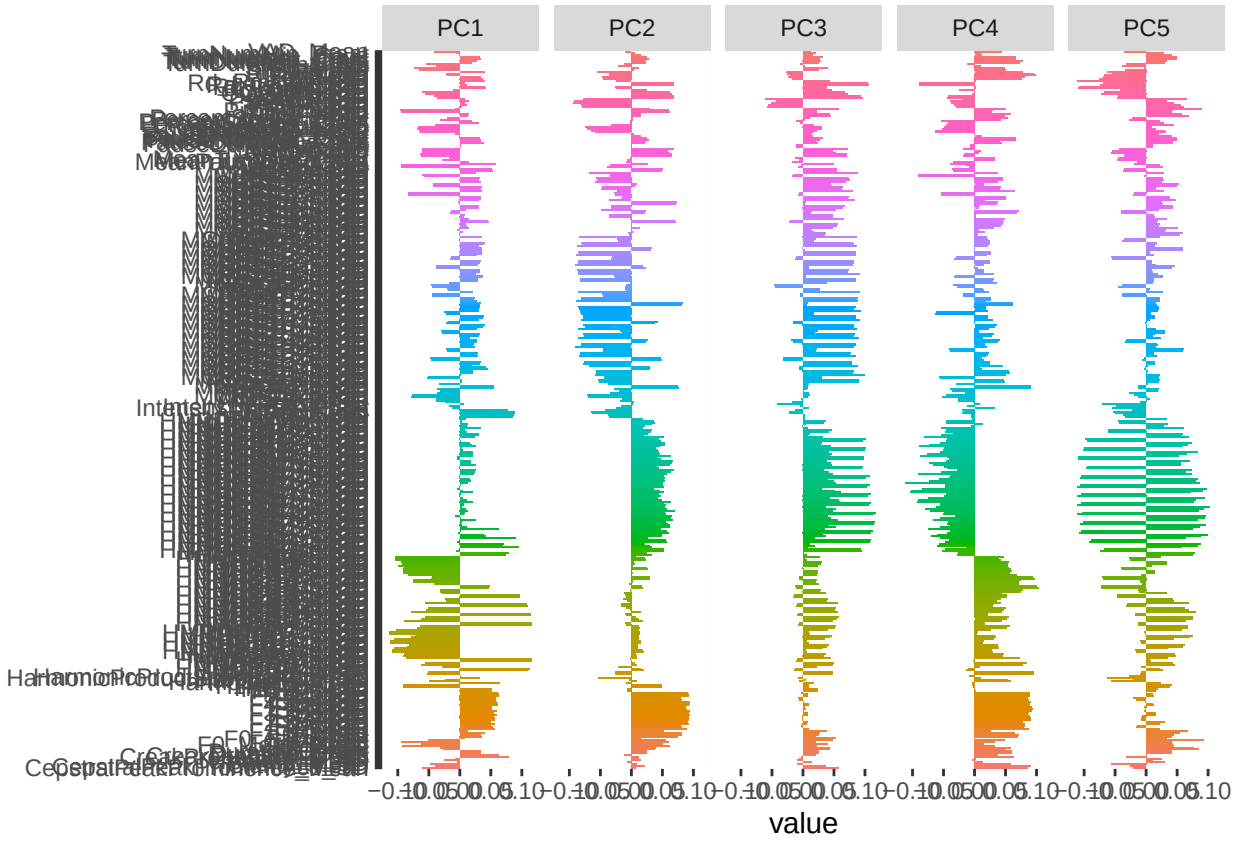
```
  mutate(Diagnosis= as.factor(Diagnosis)) %>%
  mutate(PatID = as.factor(PatID))
```

Data budgeting

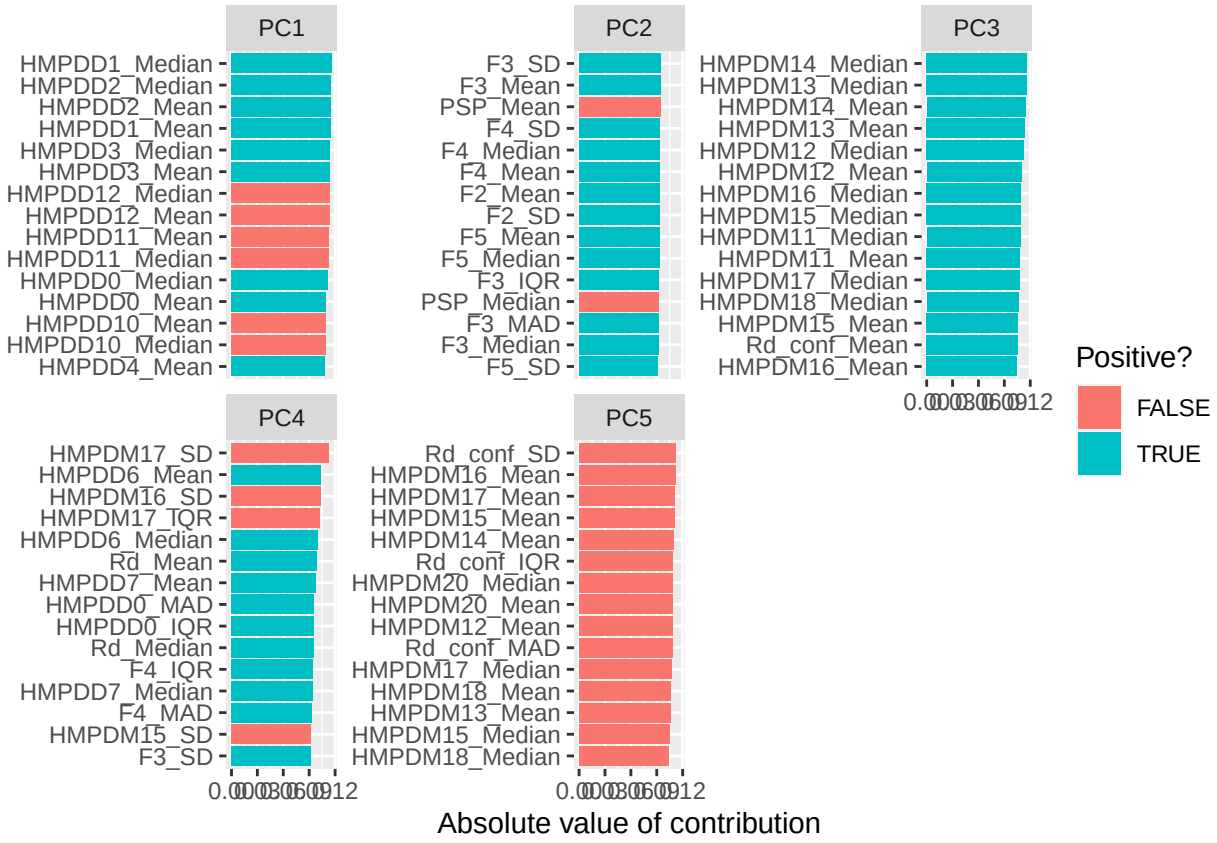
```
set.seed(222)

data_split <- initial_split(data, prop = 4/5, strata = Gender)

train_edata <- training(data_split)
test_edata  <- testing(data_split)

train_edata <- train_edata%>%
  select(-Gender) %>%
  select(-Trial)

test_edata <- test_edata%>%
  select(-Gender) %>%
  select(-Trial)


rec_train <- train_edata %>%
  recipe(Diagnosis ~ .) %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep(training = train_edata, retain = TRUE)

train_scaled <- juice(rec_train)
test_scaled <- bake(rec_train, new_data = test_edata)
```

Principal component analysis

```
pca_rec <- recipe(Diagnosis~., data = train_scaled) %>%
  update_role(PatID, new_role = "id") %>%
  step_pca(all_numeric(), id = "pca")%>%
  prep()

tidied_pca <- tidy(pca_rec, 1)

pca_bake <- bake(pca_rec, train_scaled)

pca_b_test <- bake(pca_rec, test_scaled)

tidied_pca %>%
  filter(component %in% paste0("PC", 1:5)) %>%
  mutate(component = fct_inorder(component)) %>%
  ggplot(aes(value, terms, fill = terms)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~component, nrow = 1) +
  labs(y = NULL)
```

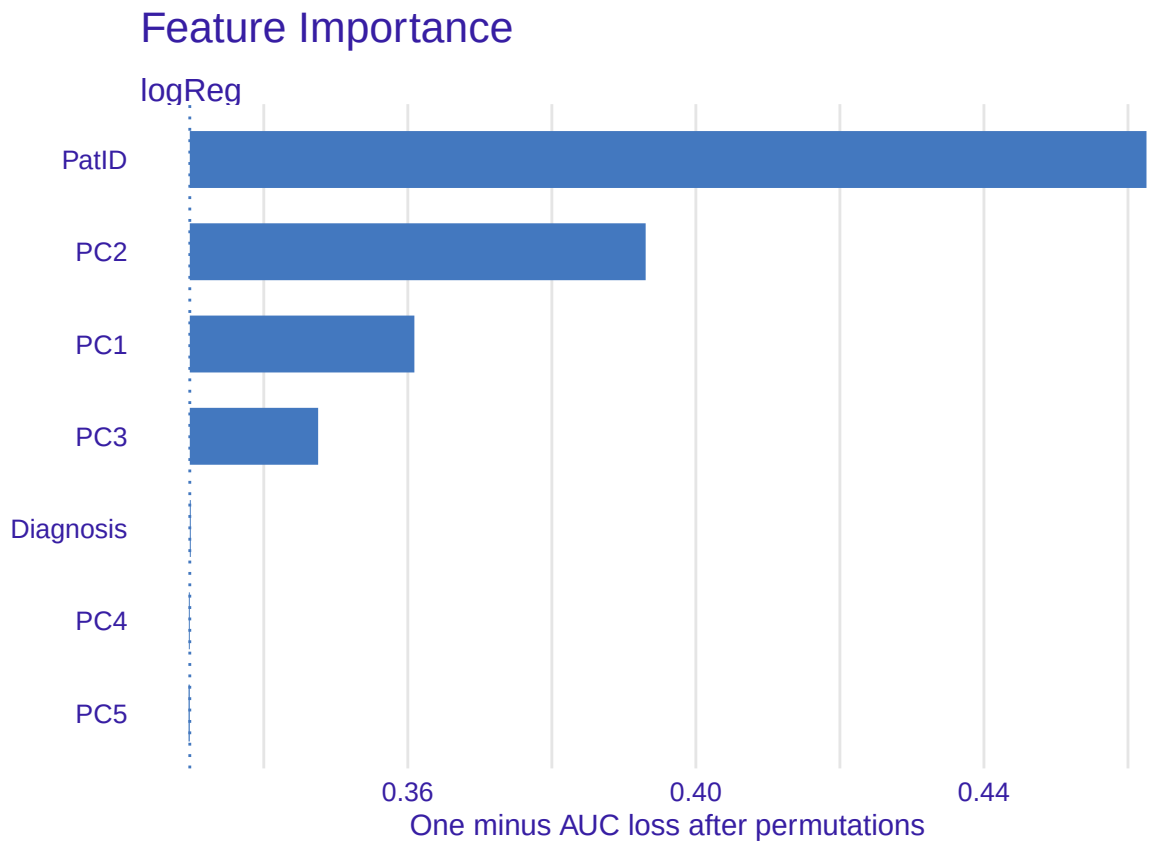



```
tidied_pca %>%
  filter(component %in% paste0("PC", 1:5)) %>%
  group_by(component) %>%
  top_n(15, abs(value)) %>%
  ungroup() %>%
  mutate(terms = reorder_within(terms, abs(value), component)) %>%
  ggplot(aes(abs(value), terms, fill = value > 0)) +
  geom_col() +
  facet_wrap(~component, scales = "free_y") +
  scale_y_reordered() +
  labs(
    x = "Absolute value of contribution",
    y = NULL, fill = "Positive?"
  )
```



Feature selection

```
pacman::p_load(DALEX, DALEXtra)
LogisticRegression_edata <- logistic_reg() %>%
        set_mode("classification") %>%
        set_engine("glm") %>%
        fit(Diagnosis ~ . , data = pca_bake)

explainer_lm <- explain_tidymodels(
        LogisticRegression_edata,
        data = pca_bake ,
        y = as.numeric(pca_bake$Diagnosis) -1,
        label = "logReg",
        verbose = FALSE)

explainer_lm %>%
        model_parts() %>%
        plot(show_boxplots = FALSE) +
        ggtitle("Feature Importance", "")
```




```
emp_f <- bf(Diagnosis ~ 1 + PC1 + PC2 + PC3 + (1 | PatID))
```

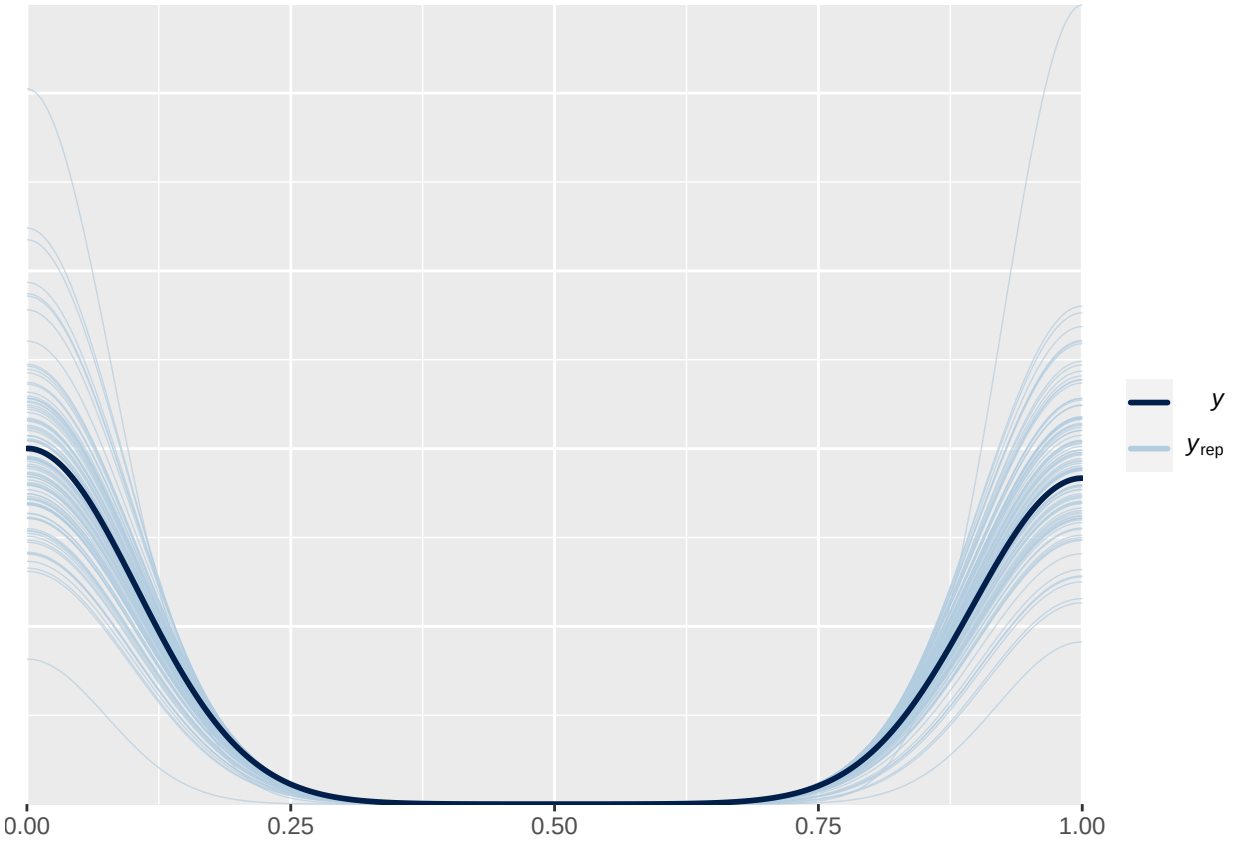
```
get_prior(emp_f,
    pca_bake,
    family = bernoulli)
```

```
##                 prior     class      coef group resp dpar nlpar lb ub
##                (flat)         b
##                (flat)         b       PC1
##                (flat)         b       PC2
##                (flat)         b       PC3
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd                                    0
##  student_t(3, 0, 2.5)        sd           PatID                    0
##  student_t(3, 0, 2.5)        sd Intercept PatID                    0
##        source
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##       default
##       default
##  (vectorized)
##  (vectorized)
```

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

```
emp_f_prior <-
  brm(
    emp_f,
    pca_bake,
    family = bernoulli,
    prior = emp_f_priors,
    sample_prior = "only",
    iter = 4000,
    warmup = 2000,
    cores = 4,
    refresh=0,
    chains = 4,
    control = list(
      adapt_delta = 0.999,
      max_treedepth = 20))
```

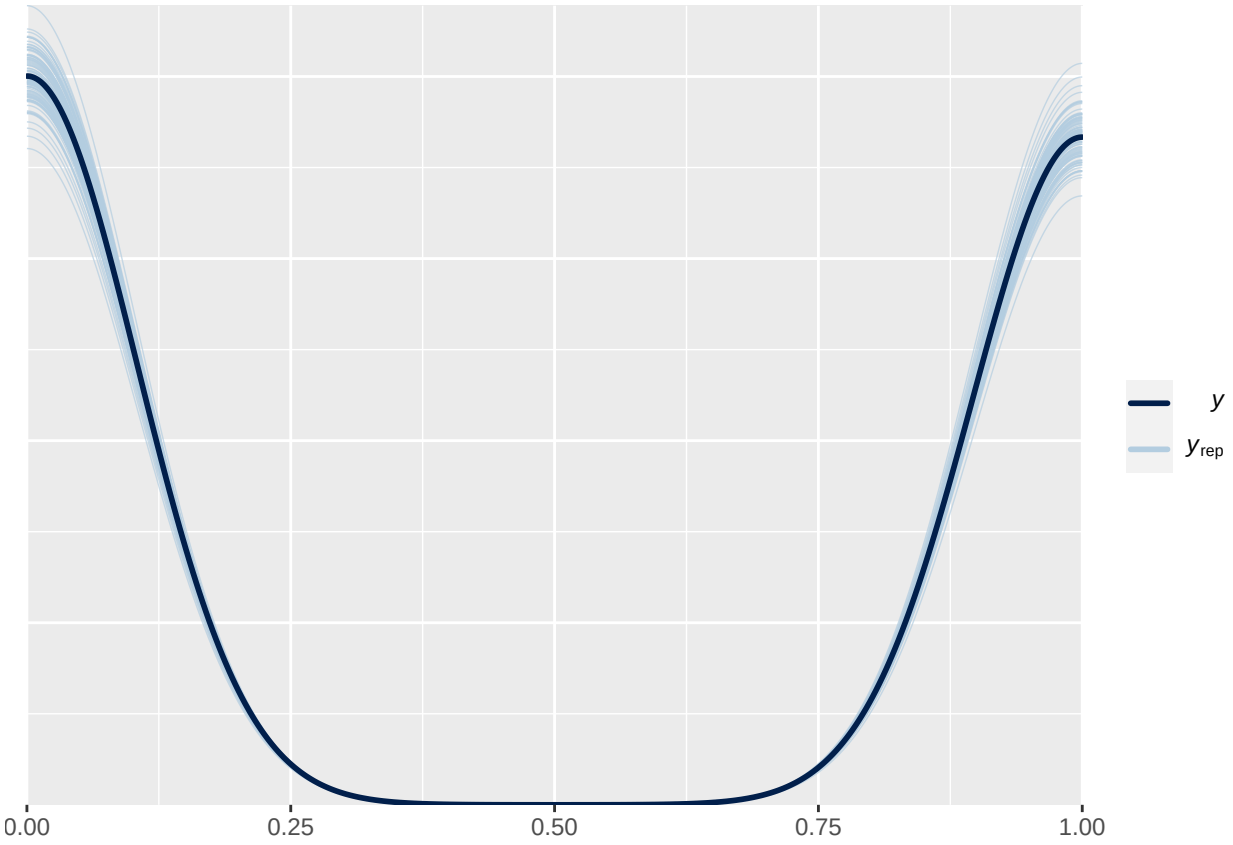
```
pp_check(emp_f_prior, ndraws = 100)
```



Fit the model

```
fitted_emp_f_prior <-
  brm(
    emp_f,
    pca_bake,
    family = bernoulli,
    prior = emp_f_priors,
    sample_prior = T,
    iter = 4000,
    warmup = 2000,
    cores = 4,
    refresh=0,
    chains = 4,
    control = list(
      adapt_delta = 0.999,
      max_treedepth = 20))
```

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

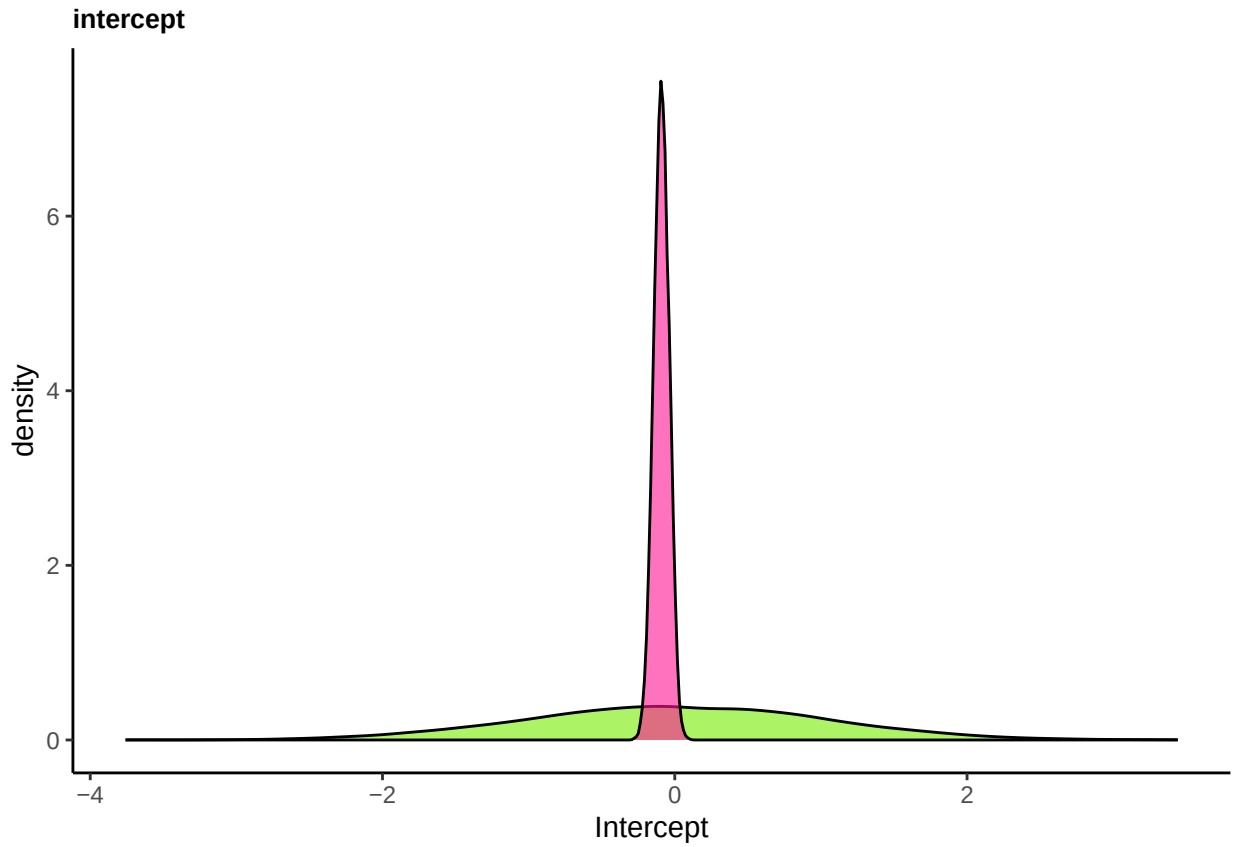



```
variables(fitted_emp_f_prior)
```

```
##   [1] "b_Intercept"            "b_PC1"                  "b_PC2"
##   [4] "b_PC3"                  "sd_PatID__Intercept"    "r_PatID[101,Intercept]"
```

```
##   [7] "r_PatID[102,Intercept]" "r_PatID[103,Intercept]" "r_PatID[104,Intercept]"
##  [10] "r_PatID[105,Intercept]" "r_PatID[106,Intercept]" "r_PatID[107,Intercept]"
##  [13] "r_PatID[108,Intercept]" "r_PatID[109,Intercept]" "r_PatID[110,Intercept]"
##  [16] "r_PatID[111,Intercept]" "r_PatID[112,Intercept]" "r_PatID[113,Intercept]"
##  [19] "r_PatID[114,Intercept]" "r_PatID[115,Intercept]" "r_PatID[116,Intercept]"
##  [22] "r_PatID[117,Intercept]" "r_PatID[118,Intercept]" "r_PatID[119,Intercept]"
##  [25] "r_PatID[120,Intercept]" "r_PatID[121,Intercept]" "r_PatID[122,Intercept]"
##  [28] "r_PatID[123,Intercept]" "r_PatID[124,Intercept]" "r_PatID[125,Intercept]"
##  [31] "r_PatID[126,Intercept]" "r_PatID[127,Intercept]" "r_PatID[128,Intercept]"
##  [34] "r_PatID[129,Intercept]" "r_PatID[130,Intercept]" "r_PatID[131,Intercept]"
##  [37] "r_PatID[132,Intercept]" "r_PatID[133,Intercept]" "r_PatID[134,Intercept]"
##  [40] "r_PatID[135,Intercept]" "r_PatID[136,Intercept]" "r_PatID[201,Intercept]"
##  [43] "r_PatID[202,Intercept]" "r_PatID[203,Intercept]" "r_PatID[204,Intercept]"
##  [46] "r_PatID[205,Intercept]" "r_PatID[206,Intercept]" "r_PatID[207,Intercept]"
##  [49] "r_PatID[208,Intercept]" "r_PatID[209,Intercept]" "r_PatID[210,Intercept]"
##  [52] "r_PatID[211,Intercept]" "r_PatID[212,Intercept]" "r_PatID[213,Intercept]"
##  [55] "r_PatID[214,Intercept]" "r_PatID[215,Intercept]" "r_PatID[216,Intercept]"
##  [58] "r_PatID[217,Intercept]" "r_PatID[218,Intercept]" "r_PatID[219,Intercept]"
##  [61] "r_PatID[242,Intercept]" "r_PatID[245,Intercept]" "r_PatID[246,Intercept]"
##  [64] "r_PatID[247,Intercept]" "r_PatID[248,Intercept]" "r_PatID[249,Intercept]"
##  [67] "r_PatID[301,Intercept]" "r_PatID[302,Intercept]" "r_PatID[303,Intercept]"
##  [70] "r_PatID[305,Intercept]" "r_PatID[306,Intercept]" "r_PatID[307,Intercept]"
##  [73] "r_PatID[310,Intercept]" "r_PatID[311,Intercept]" "r_PatID[314,Intercept]"
##  [76] "r_PatID[315,Intercept]" "r_PatID[316,Intercept]" "r_PatID[317,Intercept]"
##  [79] "r_PatID[318,Intercept]" "r_PatID[320,Intercept]" "r_PatID[323,Intercept]"
##  [82] "r_PatID[324,Intercept]" "r_PatID[326,Intercept]" "r_PatID[329,Intercept]"
##  [85] "r_PatID[330,Intercept]" "r_PatID[331,Intercept]" "r_PatID[332,Intercept]"
##  [88] "r_PatID[333,Intercept]" "r_PatID[334,Intercept]" "r_PatID[337,Intercept]"
##  [91] "r_PatID[339,Intercept]" "r_PatID[340,Intercept]" "r_PatID[342,Intercept]"
##  [94] "r_PatID[343,Intercept]" "r_PatID[344,Intercept]" "r_PatID[401,Intercept]"
##  [97] "r_PatID[402,Intercept]" "r_PatID[403,Intercept]" "r_PatID[404,Intercept]"
## [100] "r_PatID[405,Intercept]" "r_PatID[406,Intercept]" "r_PatID[407,Intercept]"
## [103] "r_PatID[408,Intercept]" "r_PatID[409,Intercept]" "r_PatID[410,Intercept]"
## [106] "r_PatID[411,Intercept]" "r_PatID[412,Intercept]" "r_PatID[413,Intercept]"
## [109] "r_PatID[414,Intercept]" "r_PatID[415,Intercept]" "r_PatID[416,Intercept]"
## [112] "r_PatID[417,Intercept]" "r_PatID[418,Intercept]" "r_PatID[419,Intercept]"
## [115] "r_PatID[420,Intercept]" "r_PatID[421,Intercept]" "r_PatID[422,Intercept]"
## [118] "r_PatID[423,Intercept]" "r_PatID[424,Intercept]" "r_PatID[440,Intercept]"
## [121] "r_PatID[441,Intercept]" "r_PatID[443,Intercept]" "r_PatID[444,Intercept]"
## [124] "r_PatID[445,Intercept]" "r_PatID[446,Intercept]" "r_PatID[447,Intercept]"
## [127] "r_PatID[448,Intercept]" "prior_Intercept"        "prior_b"
## [130] "prior_sd_PatID"         "lprior"                 "lp__"
```

```
Posterior_emp_f1 <- as_draws_df(fitted_emp_f_prior)

ggplot(Posterior_emp_f1) +
  geom_density(aes(prior_Intercept), fill="chartreuse2", color="black",alpha=0.6) +
  geom_density(aes(b_Intercept), fill="deeppink", color="black",alpha=0.6) +
  xlab('Intercept') +
  theme_classic()+
  ggtitle("intercept")+
      theme(plot.title = element_text(size = 10, face = "bold"))
```



Performance check

```
PerformanceProb2 <- tibble(expand_grid(
    Sample = seq(4000),
    Type= c("training", "test"))
    )

pca_b_test <- pca_b_test %>%
  mutate(Diagnosis = as.factor(Diagnosis))

pca_bake <- pca_bake %>%
  mutate(Diagnosis = as.factor(Diagnosis))



pca_bake$PredictionsPerc2 <- predict(fitted_emp_f_prior)[,1]
pca_bake$Predictions2[pca_bake$PredictionsPerc2 > 0.5] <- "SCZ"
```

```
## Warning: Unknown or uninitialised column: `Predictions2`.
```

```
pca_bake$Predictions2[pca_bake$PredictionsPerc2 <= 0.5] <- "CT"
```

```
pca_b_test$PredictionsPerc2 <- predict(fitted_emp_f_prior, newdata = pca_b_test, allow_new_levels = T)[
pca_b_test$Predictions2[pca_b_test$PredictionsPerc2 >= 0.5] <- "SCZ"
```

```
## Warning: Unknown or uninitialised column: `Predictions2`.
```

```
pca_b_test$Predictions2[pca_b_test$PredictionsPerc2 < 0.5] <- "CT"
```

```
train2 <- inv_logit_scaled(posterior_linpred(fitted_emp_f_prior,
    summary = F))

test2 <- inv_logit_scaled(posterior_linpred(fitted_emp_f_prior,
    summary = F,
    newdata = pca_b_test,
    allow_new_levels = T ))


for (i in seq(4000)){

    train_scaled$Predictions2 <- as.factor(ifelse(train2[i,] > 0.5, "SCZ", "CT"))
    test_scaled$Predictions2 <- as.factor(ifelse(test2[i,] > 0.5, "SCZ", "CT"))

    PerformanceProb2$Accuracy[PerformanceProb2$Sample == i & PerformanceProb2$Type =="training"] <-
    accuracy(train_scaled, truth = Diagnosis, estimate = Predictions2)[, ".estimate"]


    PerformanceProb2$Accuracy[PerformanceProb2$Sample == i & PerformanceProb2$Type =="test"] <-
    accuracy(test_scaled, truth = Diagnosis, estimate = Predictions2)[, ".estimate"]

}
```

```
## Warning: Unknown or uninitialised column: `Accuracy`.
```

```
pca_b_test <- pca_b_test %>%
  mutate(Diagnosis = as.factor(Diagnosis),
         Predictions2 = as.factor(Predictions2))

pca_bake <- pca_bake %>%
  mutate(Diagnosis = as.factor(Diagnosis),
         Predictions2 = as.factor(Predictions2))

conf_mat(
  pca_b_test,
  truth = Diagnosis,
  estimate = Predictions2,
  dnn = c("Predictions", "Truth")
)
```

```
##            Truth
## Predictions  CT SCZ
##         CT  120 100
##         SCZ  81  78
```

```
metrics(pca_b_test,
        truth = Diagnosis, estimate = Predictions2) %>%
  knitr::kable()
```

.metric	.estimator	.estimate
accuracy	binary	0.5224274
kap	binary	0.0354336

```
ggplot(PerformanceProb2) +
  geom_point(aes(x = Type, y = as.numeric(Accuracy))) + geom_abline(intercept = 0.5, slope = 0, col=c("
  theme_minimal() +
  ylab("Accuracy") +
  xlab("Type") +
  theme_minimal() +
  ggtitle("Accuracy between test and train")
```