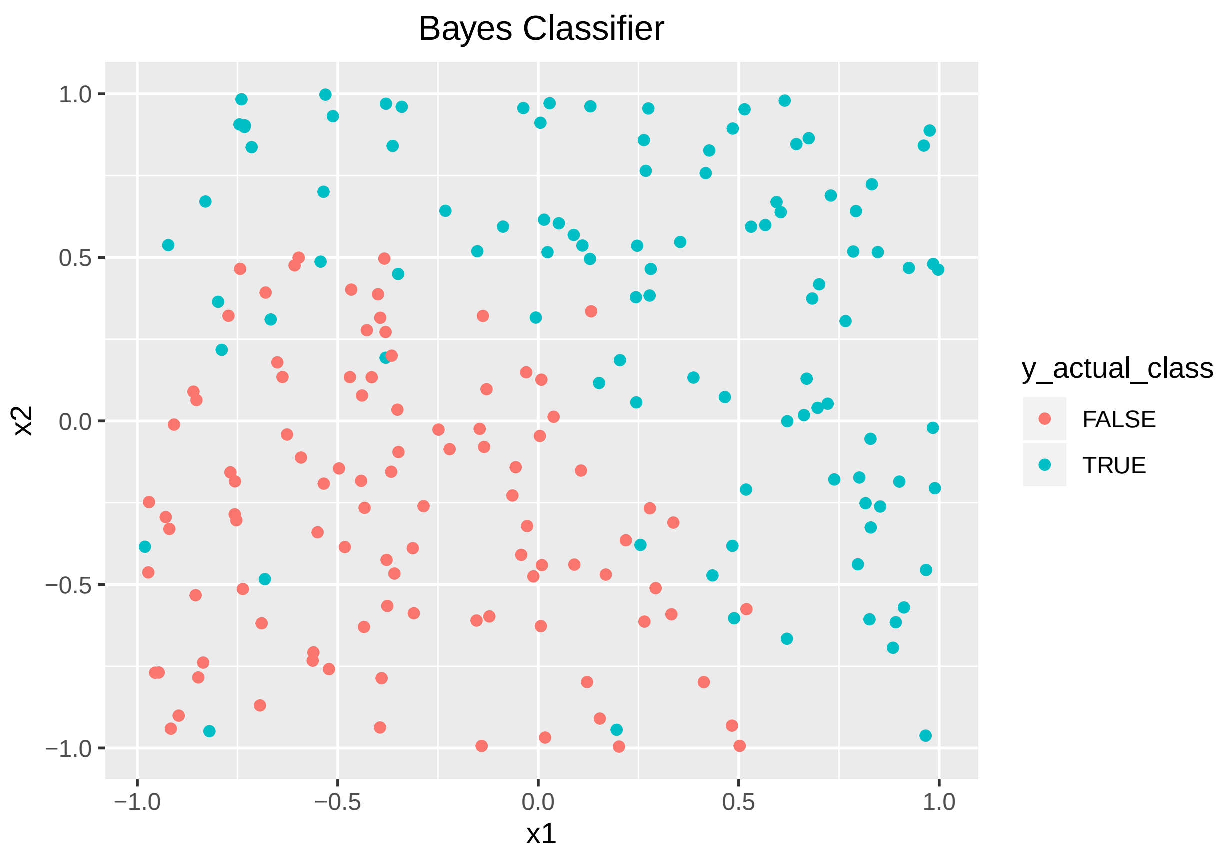
Xu_Yilun_HW02

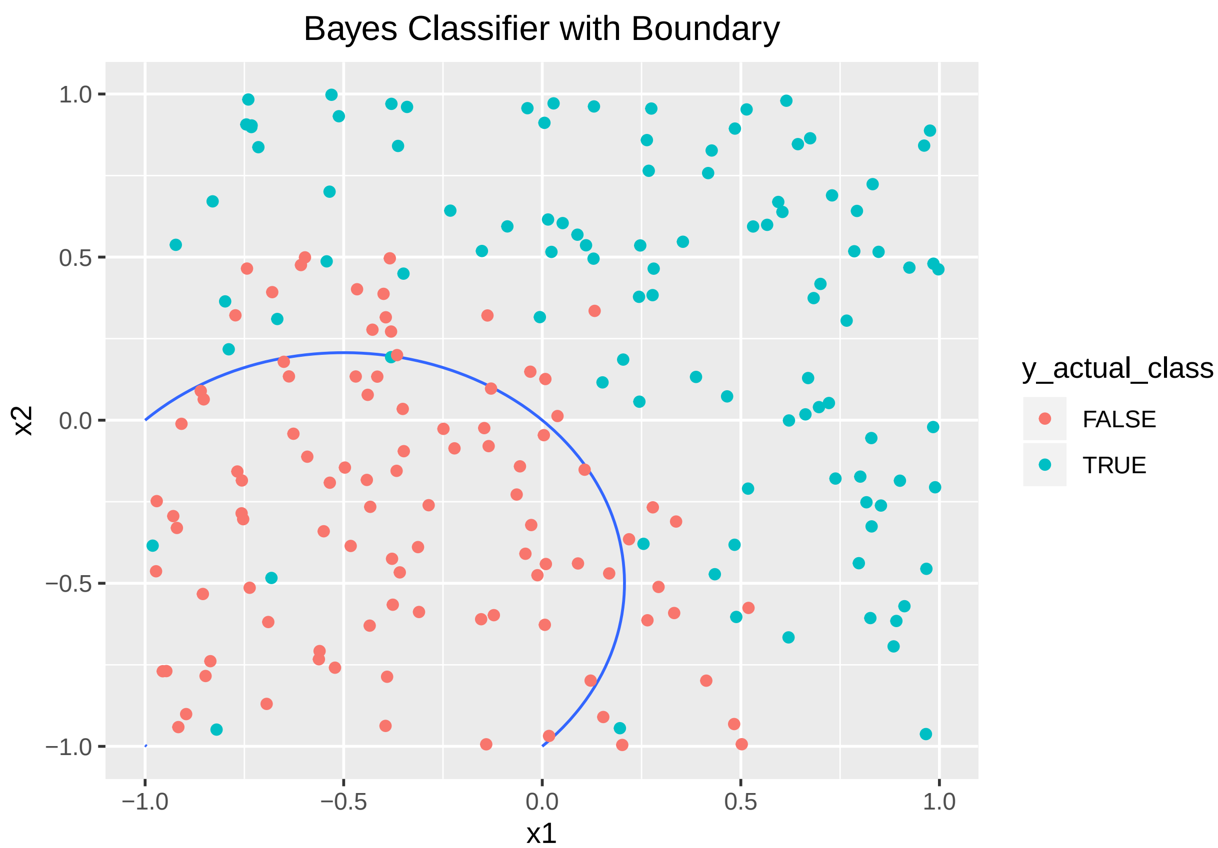
```r
library(tidyverse)
library(ggeffects)
library(tidyverse)
library(ISLR)
library(broom)
library(rsample)
library(rcfss)
library(yardstick)
library(patchwork)
library(corrplot)
library(dplyr)
library(ISLR)
library(knitr)
library(furrr)
library(e1071)
library(pROC)

set.seed(1234)
```

1. The Bayes Classifier

```r
set.seed(1234)
Model <- function(x1, x2) {
    x1 + x1^2 + x2 + x2^2}

bayes_c <- function(size = 200){
  bound_sim <- tibble(
    x1 = runif(size, -1, 1),
    x2 = runif(size, -1, 1),
    y_actual_value = Model(x1, x2) + rnorm(size, 0, 0.5),
    y_actual_class = y_actual_value > .5,
    y_model_class = Model(x1, x2) > 0.5)}

sim_bayes_c <- rerun(.n = 1, bayes_c()) %>%
  bind_rows()

##1.d
logit2prob<-function(x){
  exp(x)/(1+exp(x))
}
askedprobability <- logit2prob(sim_bayes_c$y_actual_value)

#1.e
ggplot(sim_bayes_c, aes(x=x1, y=x2, color = y_actual_class))+ geom_point()+
  labs(title = "Bayes Classifier",
       x = "x1", y = "x2")+
  theme(plot.title = element_text(hjust = 0.5))
```



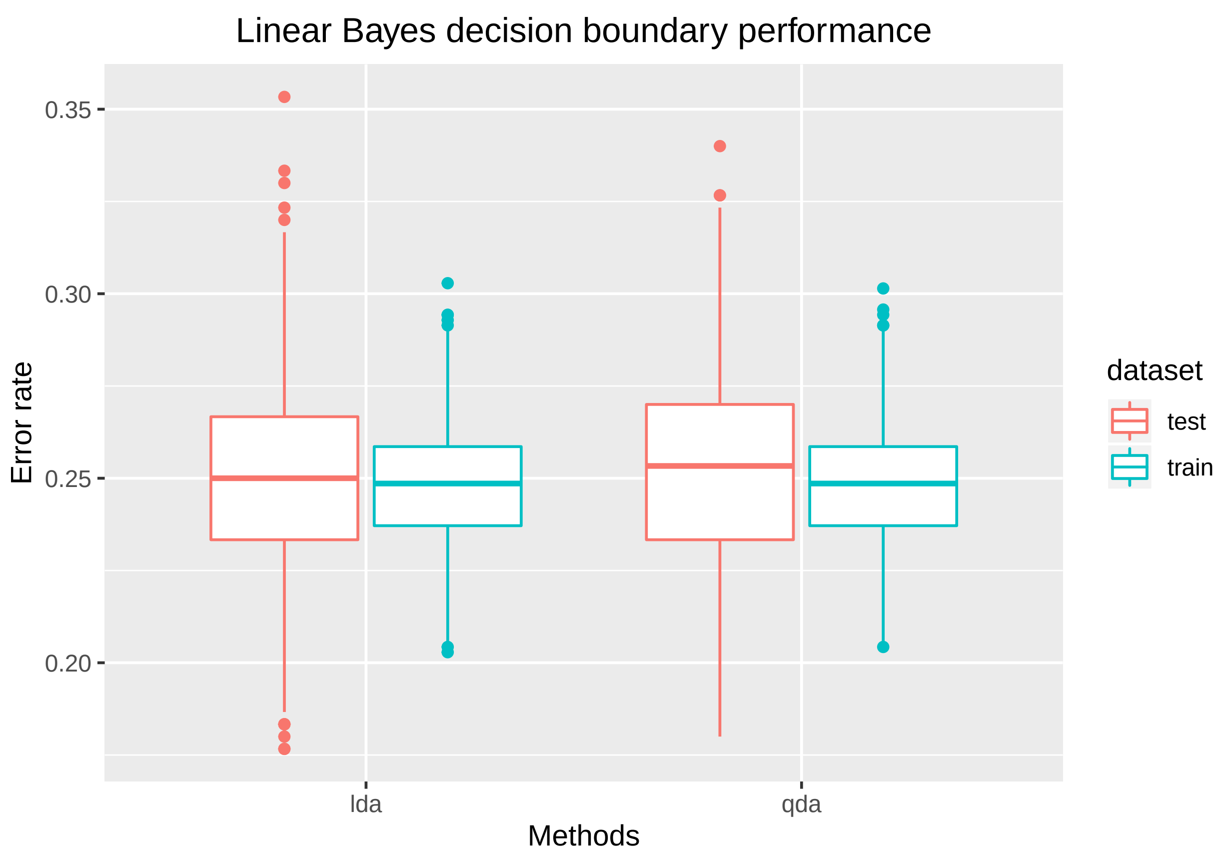

```r
grid<-expand.grid(x1=seq(-1,1,length.out = 100),x2=seq(-1,1,length.out = 100))

bayes_c_grid <- function(size = 200){
  bound_sim <- tibble(
    x1 = grid$x1,
    x2 = grid$x2,
    true_y = Model(x1, x2) + rnorm(size, 0, 0.5),
    y_fact_class = Model(x1, x2) >0.5,
    y = true_y > .5,
    y_model_value = Model(x1, x2))}
sim_bayes_c_grid <- rerun(.n = 1, bayes_c_grid()) %>%
  bind_rows()

ggplot(sim_bayes_c_grid, aes(x = x1, y=x2)) +
  geom_contour(aes(z = y_model_value),bins = 1) +
  geom_point(data = sim_bayes_c, aes(color = y_actual_class))+
  labs(title = "Bayes Classifier with Boundary",
       x = "x1", y = "x2")+
  theme(plot.title = element_text(hjust = 0.5))
```



2. Exploring Simulated Differences between LDA and QDA

```
set.seed(1234)
linear <- function(size = 1e03){
  Model <- function(x1, x2) {
    x1 + x2
  }
  data_set <- tibble(
    x1 = runif(size, -1, 1),
    x2 = runif(size, -1, 1),
    y_actual_value = Model(x1, x2) + rnorm(size, 0, 1),
    y_actual_class = y_actual_value > .5,
    y_model_class = Model(x1, x2) > .5)

  data_split <- initial_split(data_set, prop = 0.7)
  data_train <- training(data_split)
  data_test <- testing(data_split)

  model_lda <- MASS::lda(y_actual_class ~ x1 + x2, data = data_train)
  model_qda <- MASS::qda(y_actual_class ~ x1 + x2, data = data_train)

  tibble(
    lda_train = mean(predict(model_lda, newdata = data_train)$class!=data_train$y_actual_class),
    lda_test = mean(predict(model_lda, newdata = data_test)$class != data_test$y_actual_class),
    qda_train = mean(predict(model_qda, newdata = data_train)$class != data_train$y_actual_class),
```

```
    qda_test = mean(predict(model_qda, newdata = data_test)$class != data_test$y_actual_class))}

run_linear <- rerun(.n = 1000, linear()) %>%
  bind_rows() %>%
  gather(key = variable, value = error) %>%
  separate(col = variable, into = c("method", "dataset"))

ggplot(run_linear, aes(method, error, color = dataset)) + geom_boxplot() +
  labs(title = "Linear Bayes decision boundary performance",
       x = "Methods", y = "Error rate")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
run_linear %>%
  group_by(method, dataset) %>%
  summarize(error = mean(error))
```

```
## # A tibble: 4 x 3
## # Groups:   method [2]
##   method dataset error
##   <chr>  <chr>   <dbl>
## 1 lda    test    0.251
## 2 lda    train   0.248
## 3 qda    test    0.252
## 4 qda    train   0.248
```

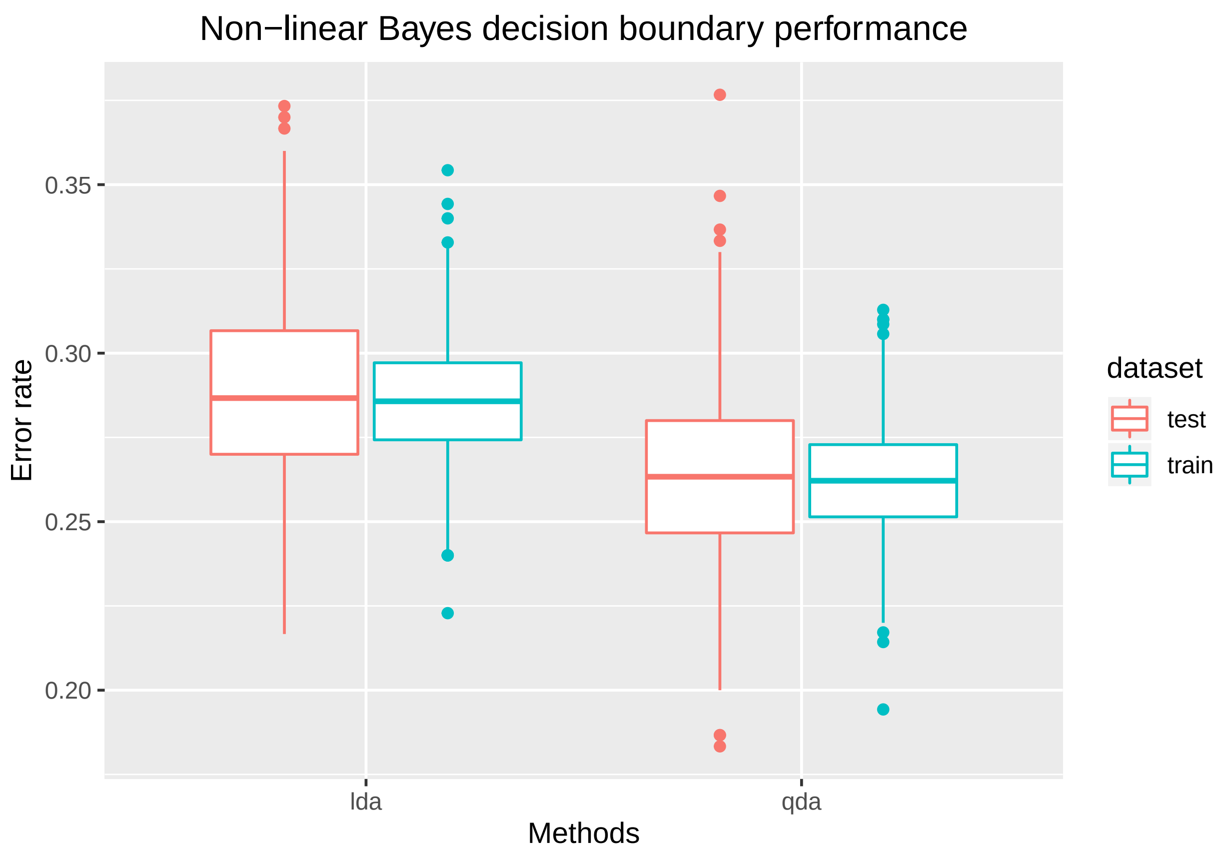
According to the above analysis, for train data, QDA performs better. For test data, IDA performs better.

3. Exploring Simulated Differences between LDA and QDA

```
set.seed(1234)
non_linear <- function(.nobs = 1e03){
  Model <- function(x1, x2) {
    x1 + x1^2 + x2 + x2^2}

  data_set <- tibble(x1 = runif(.nobs, -1, 1),
                     x2 = runif(.nobs, -1, 1),
                     y_actual_value = Model(x1, x2) + rnorm(.nobs, 0, 1),
                     y_actual_class = y_actual_value > .5,
                     y_model_class = Model(x1, x2) > .5)

  data_split <- initial_split(data_set, prop = .7)
  data_train <- training(data_split)
  data_test <- testing(data_split)

  model_lda <- MASS::lda(y_actual_class ~ x1 + x1^2 + x2 + x2^2, data = data_train)
  model_qda <- MASS::qda(y_actual_class ~ x1 + x1^2 + x2 + x2^2, data = data_train)

  tibble(
    lda_train = mean(predict(model_lda, newdata = data_train)$class!=data_train$y_actual_class),
    lda_test = mean(predict(model_lda, newdata = data_test)$class != data_test$y_actual_class),
    qda_train = mean(predict(model_qda, newdata = data_train)$class != data_train$y_actual_class),
    qda_test = mean(predict(model_qda, newdata = data_test)$class != data_test$y_actual_class))}

run_non_linear <- rerun(.n = 1000, non_linear()) %>%
  bind_rows() %>%
  gather(key = variable, value = error) %>%
  separate(col = variable, into = c("method", "dataset"))

run_non_linear %>%
  group_by(method, dataset) %>%
  summarize(error = mean(error))
```

```
## # A tibble: 4 x 3
## # Groups:   method [2]
##   method dataset error
##   <chr>  <chr>   <dbl>
## 1 lda    test    0.288
## 2 lda    train   0.286
## 3 qda    test    0.265
## 4 qda    train   0.262
```

```
ggplot(run_non_linear, aes(method, error, color = dataset)) + geom_boxplot() +
  labs(title = "Non-linear Bayes decision boundary performance",
       x = "Methods", y = "Error rate")+
  theme(plot.title = element_text(hjust = .5))
```



According to the above analysis, for train data and test data, QDA performs better.

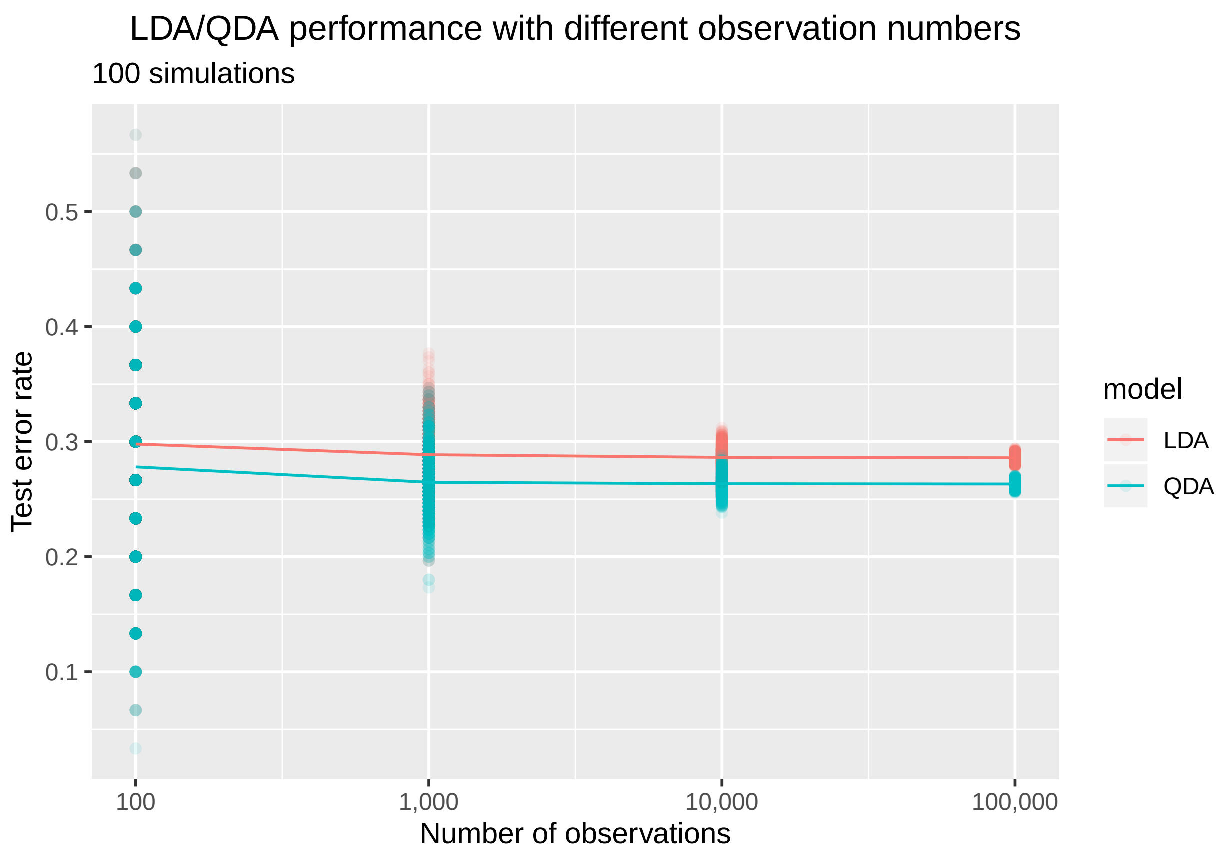
4. Exploring Simulated Differences between LDA and QDA

```
set.seed(1234)
more_non_linear <- function(.nobs = 1e03){
  Model <- function(x1, x2) {x1 + x1^2 + x2 + x2^2}
  data_set <- tibble(
    x1 = runif(.nobs, -1, 1),
    x2 = runif(.nobs, -1, 1),
    y_actual_value = Model(x1, x2) + rnorm(.nobs, 0, 1),
    y_actual_class = y_actual_value > .5,
    y_model_class = Model(x1, x2) > .5)

  data_split <- initial_split(data_set, prop = .7)
  data_train <- training(data_split)
  data_test <- testing(data_split)

  model_lda <- MASS::lda(y_actual_class ~ x1 + x1^2 + x2 + x2^2, data = data_train)
  model_qda <- MASS::qda(y_actual_class ~ x1 + x1^2 + x2 + x2^2, data = data_train)

  tibble(
    lda_test = mean(predict(model_lda, newdata = data_test)$class != data_test$y_actual_class),
    qda_test = mean(predict(model_qda, newdata = data_test)$class != data_test$y_actual_class),
    diff = qda_test - lda_test)}
```

```r
system.time({
  run_more_non_linear <- tibble(
    nobs = c(1e02, 1e03, 1e04, 1e05)) %>%
    mutate(sims = future_map(nobs, ~ rerun(.n = 1000, more_non_linear(.nobs = .x)))) %>% unnest(sims) %
  })
```

```
##    user  system elapsed
##  582.32    4.17  593.56
```

```r
run_df <- run_more_non_linear %>%
  select(-diff) %>%
  gather(model, error, ends_with("test")) %>%
  mutate(model = str_remove(model, "_test"),
         model = str_to_upper(model))

ggplot(run_df, aes(nobs, error, color = model)) +
  geom_point(alpha = .1) +
  geom_line(data = run_df %>%
              group_by(nobs, model) %>%
              summarize(error = mean(error))) +
  labs(title = "LDA/QDA performance with different observation numbers",
       subtitle = "100 simulations",
       x = "Number of observations",
       y = "Test error rate") +
  theme(plot.title = element_text(hjust = .5)) +
  scale_x_log10(labels = scales::comma)
```



5. Modeling voter turnout

```
set.seed(1234)
# 1
df <- read_csv("mental_health.csv") %>% na.omit
```

```
## Parsed with column specification:
## cols(
##   vote96 = col_double(),
##   mhealth_sum = col_double(),
##   age = col_double(),
##   educ = col_double(),
##   black = col_double(),
##   female = col_double(),
##   married = col_double(),
##   inc10 = col_double()
## )
```

```
df_split <- initial_split(df, prop = .7)
df_train <- training(df_split)
df_test <- testing(df_split)

#2
df_logit <- glm(vote96 ~ ., data = df_train)
```

```r
df_lda <- MASS::lda(vote96 ~ ., data = df_train)
df_qda <- MASS::qda(vote96 ~ ., data = df_train)
df_nb <- naiveBayes(vote96 ~ ., data = mutate(df_train, vote96 = factor(vote96)))
df_knn_pred <- map(1:10, ~ class::knn(select(df_train, -vote96),
                                      cl = df_train$vote96,
                                      test = select(df_test, -vote96),
                                      k = .x, prob = TRUE)) %>%
  map(~ attr(.x, "prob"))

#3
df_logit_pred <- predict(df_logit, newdata = df_test)
df_lda_pred <- predict(df_lda, newdata = df_test)$posterior[, 2]
df_qda_pred <- predict(df_qda, newdata = df_test)$posterior[, 2]
df_nb_pred <- predict(df_nb, newdata = mutate(df_test, vote96 = factor(vote96)), type = "raw")[, 1]

test_performance <- list(logit = df_logit_pred,
                  lda = df_lda_pred,
                  qda = df_qda_pred,
                  nb = df_nb_pred) %>%
  enframe(name = "type", value = "test_performance") %>%
  bind_rows(tibble(type = str_c("knn", 1:10, sep = "_"),test_performance = df_knn_pred))

#error rate
test_error_rate <- test_performance %>%
  mutate(error = map_dbl(test_performance, ~ mean(round(.x) != df_test$vote96)))

#roc/auc
test_roc_auc <- test_performance %>%
  mutate(roc = map(test_performance, ~ roc(df_test$vote96, .x)),
         auc = map(test_performance, ~ auc(df_test$vote96, .x)))
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
test_roc_auc$roc
```

```
## [[1]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.779
##
## [[2]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.779
##
## [[3]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.7647
##
## [[4]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) > 233 cases (df_test$vote96 1).
## Area under the curve: 0.7547
##
## [[5]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.4956
##
## [[6]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.5028
##
## [[7]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
```

```
## Area under the curve: 0.5706
##
## [[8]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.5915
##
## [[9]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.599
##
## [[10]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.6183
##
## [[11]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.6263
##
## [[12]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.6422
##
## [[13]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.6617
##
## [[14]]
##
## Call:
## roc.default(response = df_test$vote96, predictor = .x)
```

```
##
## Data: .x in 116 controls (df_test$vote96 0) < 233 cases (df_test$vote96 1).
## Area under the curve: 0.6633
```

```r
#4
arrange(test_error_rate,error)
```

```
## # A tibble: 14 x 3
##    type   test_performance error
##    <chr>  <list>           <dbl>
##  1 qda    <dbl [349]>      0.269
##  2 lda    <dbl [349]>      0.281
##  3 logit  <dbl [349]>      0.292
##  4 knn_7  <dbl [349]>      0.330
##  5 knn_8  <dbl [349]>      0.335
##  6 knn_3  <dbl [349]>      0.338
##  7 knn_9  <dbl [349]>      0.338
##  8 knn_5  <dbl [349]>      0.347
##  9 knn_1  <dbl [349]>      0.350
## 10 knn_10 <dbl [349]>      0.350
## 11 knn_4  <dbl [349]>      0.355
## 12 knn_6  <dbl [349]>      0.355
## 13 knn_2  <dbl [349]>      0.441
## 14 nb     <dbl [349]>      0.702
```

```r
##According to test error rates, QDA performs best, since it has the smallest test error rate.

test_roc_auc$auc
```

```
## [[1]]
## Area under the curve: 0.779
##
## [[2]]
## Area under the curve: 0.779
##
## [[3]]
## Area under the curve: 0.7647
##
## [[4]]
## Area under the curve: 0.7547
##
## [[5]]
## Area under the curve: 0.4956
##
## [[6]]
## Area under the curve: 0.5028
##
## [[7]]
## Area under the curve: 0.5706
##
## [[8]]
## Area under the curve: 0.5915
##
```

```
## [[9]]
## Area under the curve: 0.599
##
## [[10]]
## Area under the curve: 0.6183
##
## [[11]]
## Area under the curve: 0.6263
##
## [[12]]
## Area under the curve: 0.6422
##
## [[13]]
## Area under the curve: 0.6617
##
## [[14]]
## Area under the curve: 0.6633
```

```
## According to AUC, the first and second models are the best. They are logistic regression model and I
```