Data science final guideline

Yi Huang, Yuchen Zhang, Shun Xie

2023-04-28

Contents

Final Project

Deliverables and deadlines

Due: May 10 by 11:59pm
Each team is required to submit one report.

Dataset

Please merge the midterm project datasets from two team members, and keep only the unique observations. If there are three team members, you can decide which two datasets to merge.

Background

To gain a better understanding of the factors that predict recovery time from COVID-19 illness, a study was designed to combine three existing cohort studies that have been tracking participants for several years. The study collects recovery information through questionnaires and medical records, and leverages existing data on personal characteristics prior to the pandemic. The ultimate goal is to develop a prediction model for recovery time and identify important risk factors for long recovery time.

Data

recovery.RData

The dataset in "recovery.RData" consists of 10000 participants. In your analysis, please draw a random sample of 2000 participants using the following R code:

set.seed([last four digits of your UNI])

dat <- dat[sample(1:10000, 2000),]

The resulting dat object will contain a random sample of 2000 participants that you can use for your analysis.

Here is a description of each variable:

Variable Description

- ID (id) :Participant ID
- Gender (gender): 1 = Male, 0 = Female
- Race/ethnicity (race): 1 = White, 2 = Asian, 3 = Black, 4 = Hispanic
- Smoking (smoking): Smoking status; 0 = Never smoked, 1 = Former smoker, 2 = Current smoker
- Height (height): Height (in centimeters)
- Weight (weight): Weight (in kilograms)
- BMI (bmi): Body Mass Index; BMI = weight (in kilograms) / height (in meters) squared
- Hypertension (hypertension): 0 = No, 1 = Yes
- Diabetes (diabetes): 0 = No, 1 = Yes
- Systolic blood pressure (SBP): Systolic blood pressure (in mm/Hg)
- LDL cholesterol (LDL): LDL (low-density lipoprotein) cholesterol (in mg/dL)
- Vaccination status at the time of infection (vaccine): 0 = Not vaccinated, 1 = Vaccinated
- Severity of COVID-19 infection (severity): 0 = Not severe, 1= Severe
- Study (study): The study (A/B/C) that the participant belongs to
- Time to recovery (tt__recovery__time): Time from COVID-19 infection to recovery in days

Report

Your report should be no more than five pages, excluding figures and tables. The total number of figures and tables should not exceed 8. You can submit an Appendix if you would like to include more details for modeling tuning.

For the primary analysis on time to recovery, you may revise your midterm report and incorporate new findings from the methods introduced in the second half of the semester.

As a secondary analysis, please consider time to recovery as a binary outcome (>30 days vs. <= 30 days) and develop a prediction model for this binary outcome.

Note

Similar to the midterm project report, your report should include the following sections: exploratory analysis and data visualization, model training, results, and conclusions.

```r
library(caret)
library(corrplot)
library(tidyverse)
library(ggplot2)
library(dplyr)
library(ISLR)
library(glmnet)
library(mgcv)
library(nlme)
library(earth)
library(vip)
```

Introduction & data manipulation

```
#load data
load("data/recovery.RData")

#get data from 2 team members uni
set.seed(3554)
dat1 <- dat[sample(1:10000, 2000),]
set.seed(4437)
dat2 <- dat[sample(1:10000, 2000),]

#combine the data and discard duplicates
dat_temp <- rbind(dat1, dat2)
dat <- dat_temp[!duplicated(dat_temp$id),]

#dat is now the final data and it has the length as following:
length(dat$id)
```

```
## [1] 3593
```

```
# save(dat, file = "data/covid_recovery.RData")
```

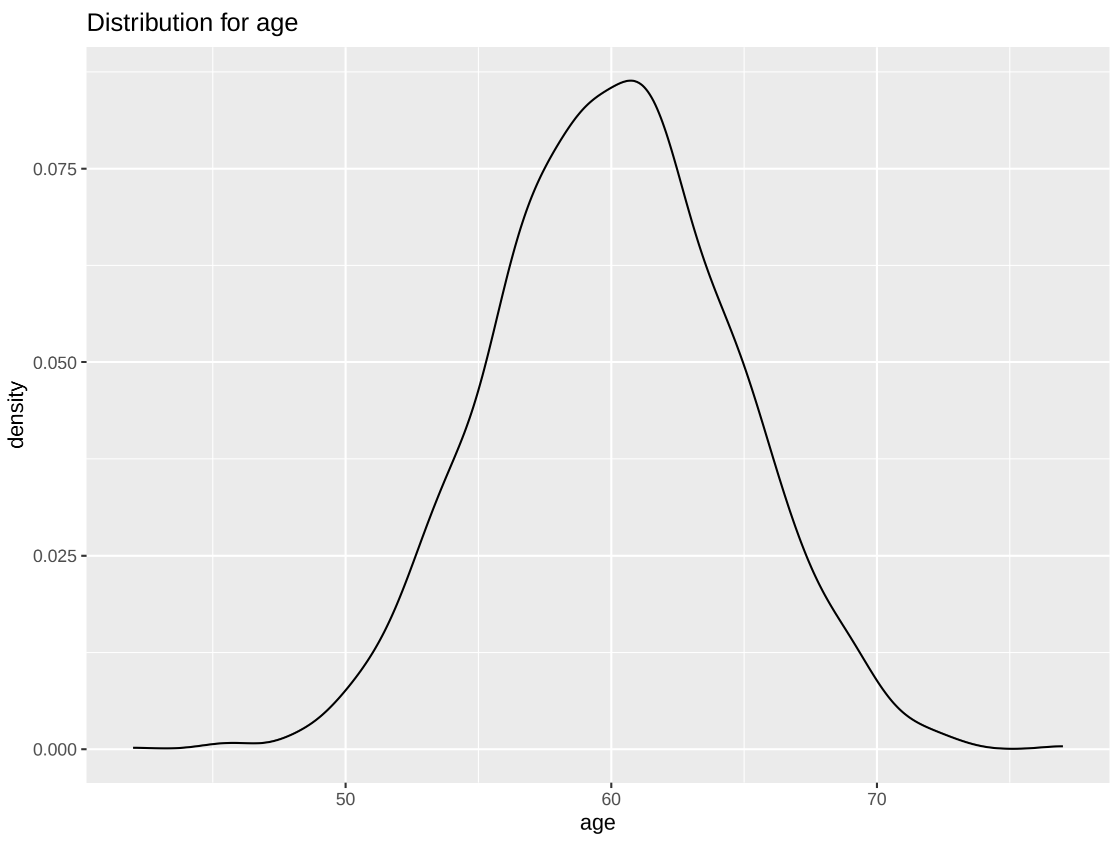
```
# data manipulation
data =
  dat %>%
  mutate(study = factor(dat$study),
         gender = factor(dat$gender),
         hypertension = factor(dat$hypertension),
         diabetes = factor(dat$diabetes),
         vaccine = factor(dat$vaccine),
         severity = factor(dat$severity)) %>%
  dplyr::select(-id)
head(data)
```

```
##      age gender race smoking height weight  bmi hypertension diabetes SBP LDL
## 2989  63      1    1       0  174.8   83.6 27.4            1        0 137 108
## 620   60      1    1       0  170.0   80.7 27.9            0        0 128  80
## 9226  64      0    1       0  167.5   89.1 31.7            1        0 137 126
## 4098  69      0    1       2  173.0   72.5 24.2            1        0 135 149
## 8349  62      0    1       0  172.6   82.4 27.7            1        0 138 136
## 184   60      1    1       0  166.5   82.9 29.9            0        0 128 118
##      vaccine severity study recovery_time
## 2989       1        0     B            14
## 620        1        0     A            36
## 9226       0        0     C            50
## 4098       0        0     B            65
## 8349       1        0     C            34
## 184         1         0      A             32
```

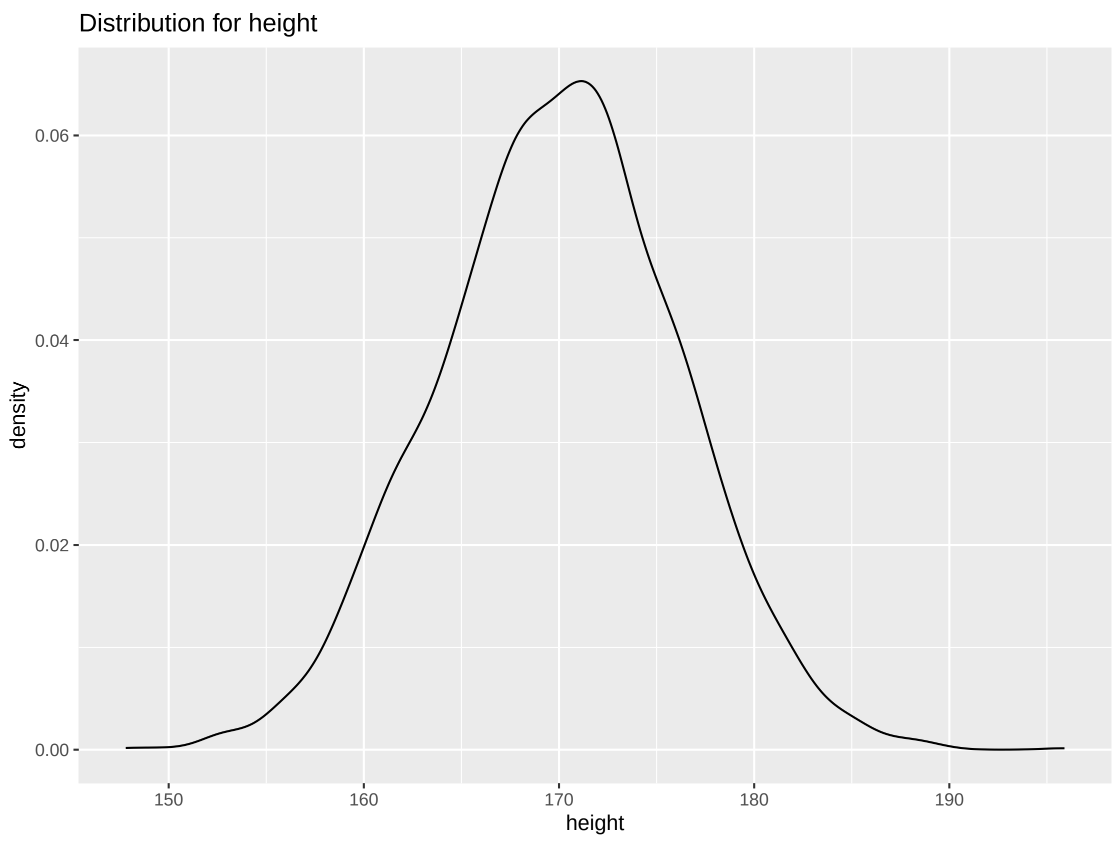
```
#Split data into 70-300, using the third member's uni
set.seed(2337)
indexTrain <- createDataPartition(y = data$recovery_time, p = 0.7, list = FALSE)

# training data
train_data <- data[indexTrain,]
# matrix of predictors
train_x <- model.matrix(recovery_time~.,train_data)[,-1]
# vector of response
train_y <- train_data$recovery_time

# test data
test_data <- data[-indexTrain,]
# matrix of predictors
test_x <- model.matrix(recovery_time~.,test_data)[,-1]
# vector of response
test_y <- test_data$recovery_time
```

Exploratory Data Analysis on train data

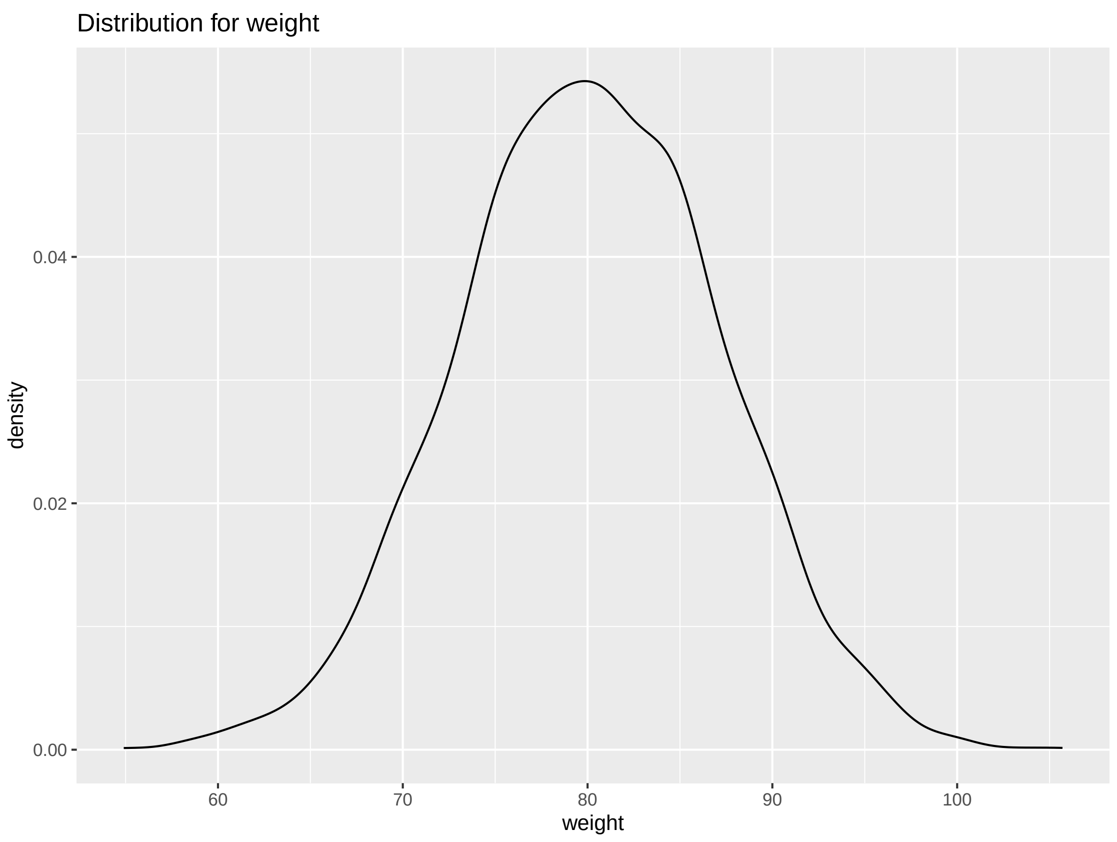
```
#continuous
ggplot(train_data, aes(x = age))+geom_density()+labs(title="Distribution for age")
```



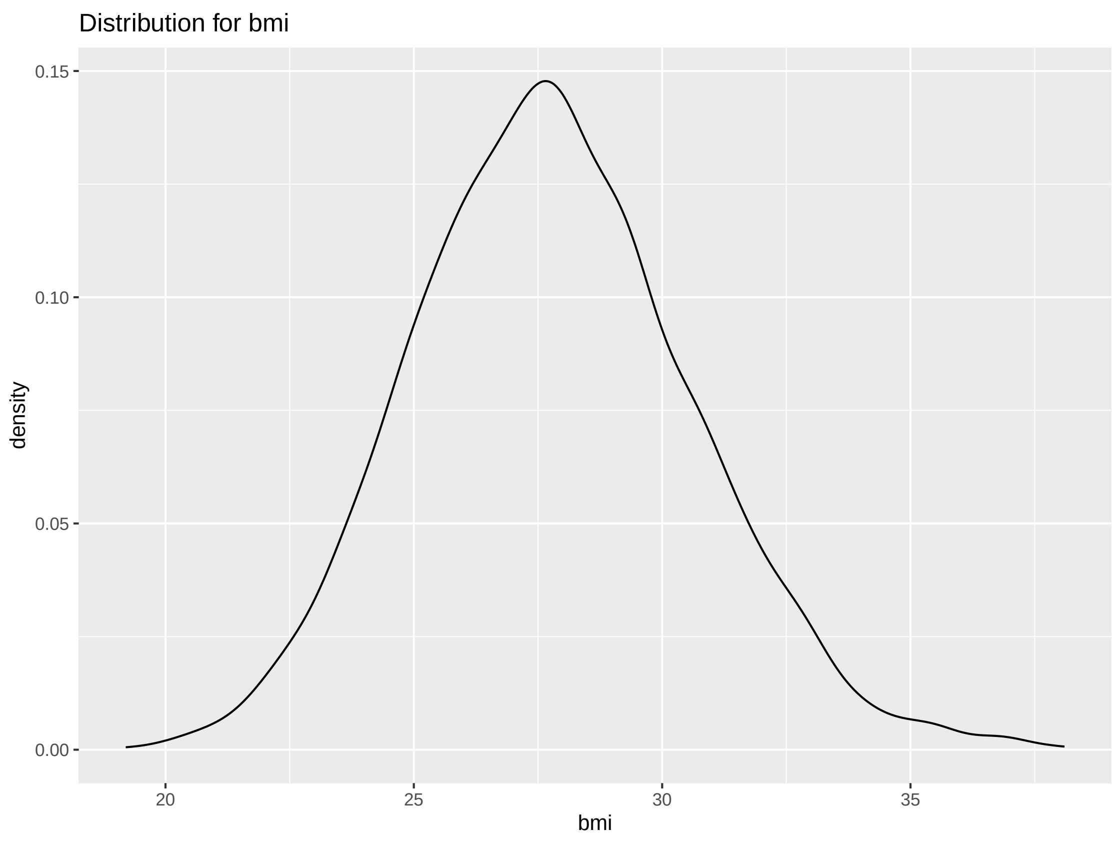
```
ggplot(train_data, aes(x = height))+geom_density()+labs(title="Distribution for height")
```



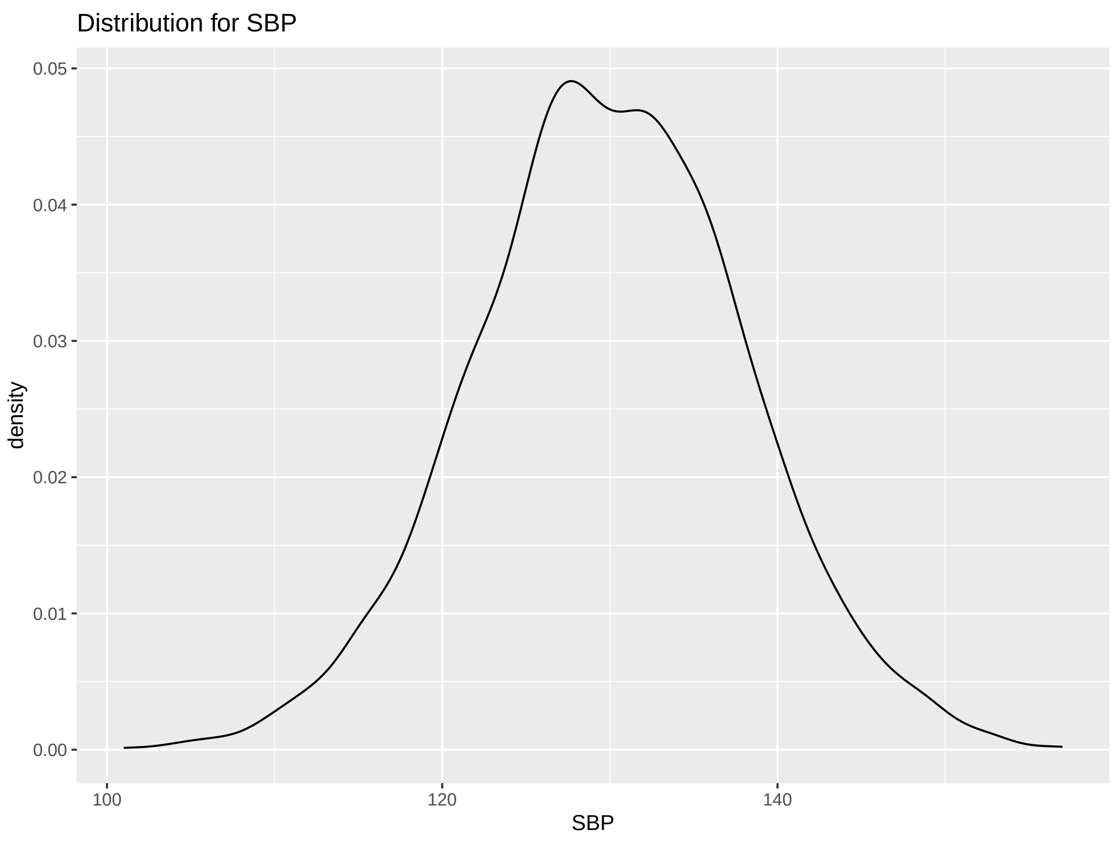
```
ggplot(train_data, aes(x = weight))+geom_density()+labs(title="Distribution for weight")
```



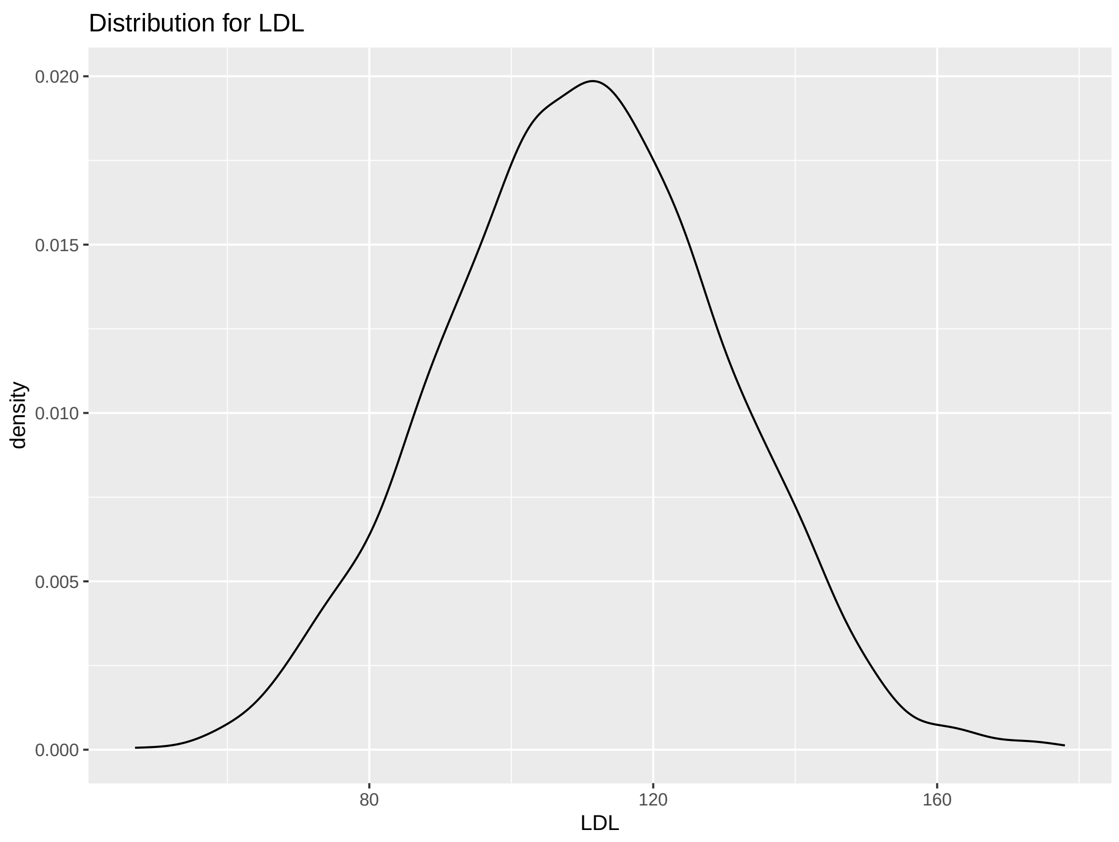
```
ggplot(train_data, aes(x = bmi))+geom_density()+labs(title="Distribution for bmi")
```



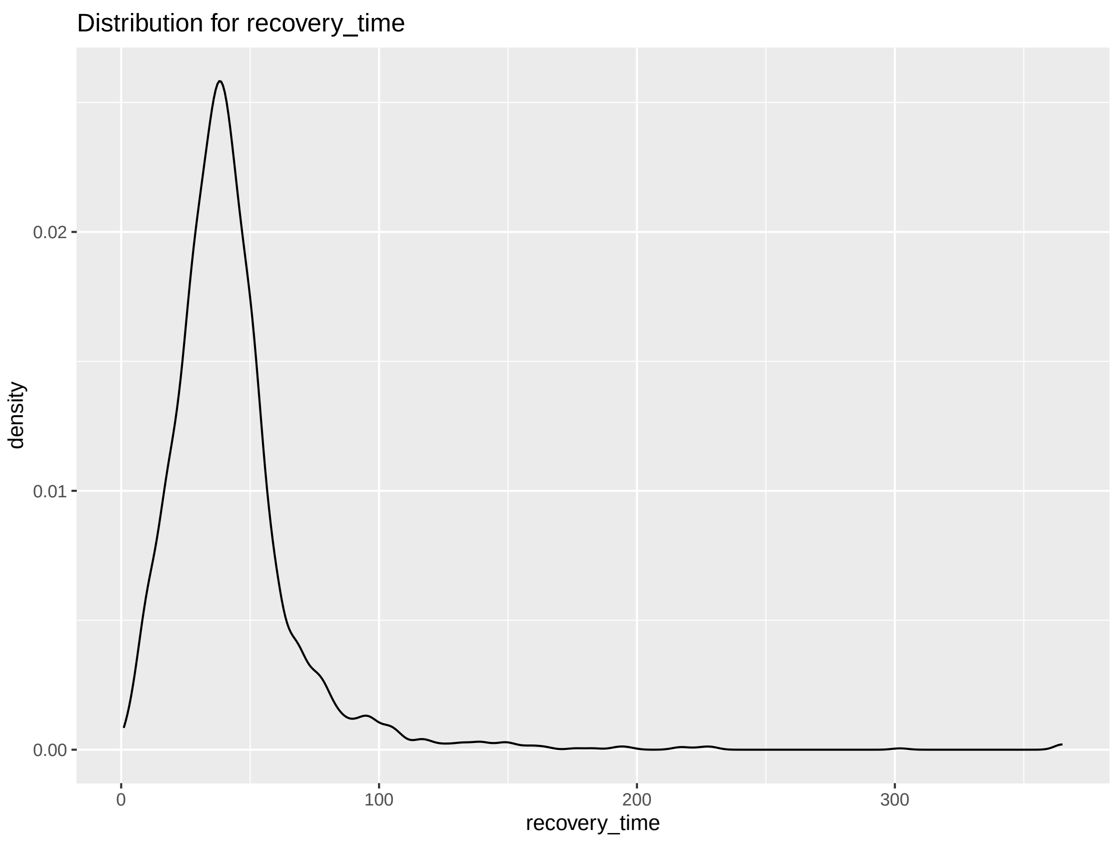
```
ggplot(train_data, aes(x = SBP))+geom_density()+labs(title="Distribution for SBP")
```



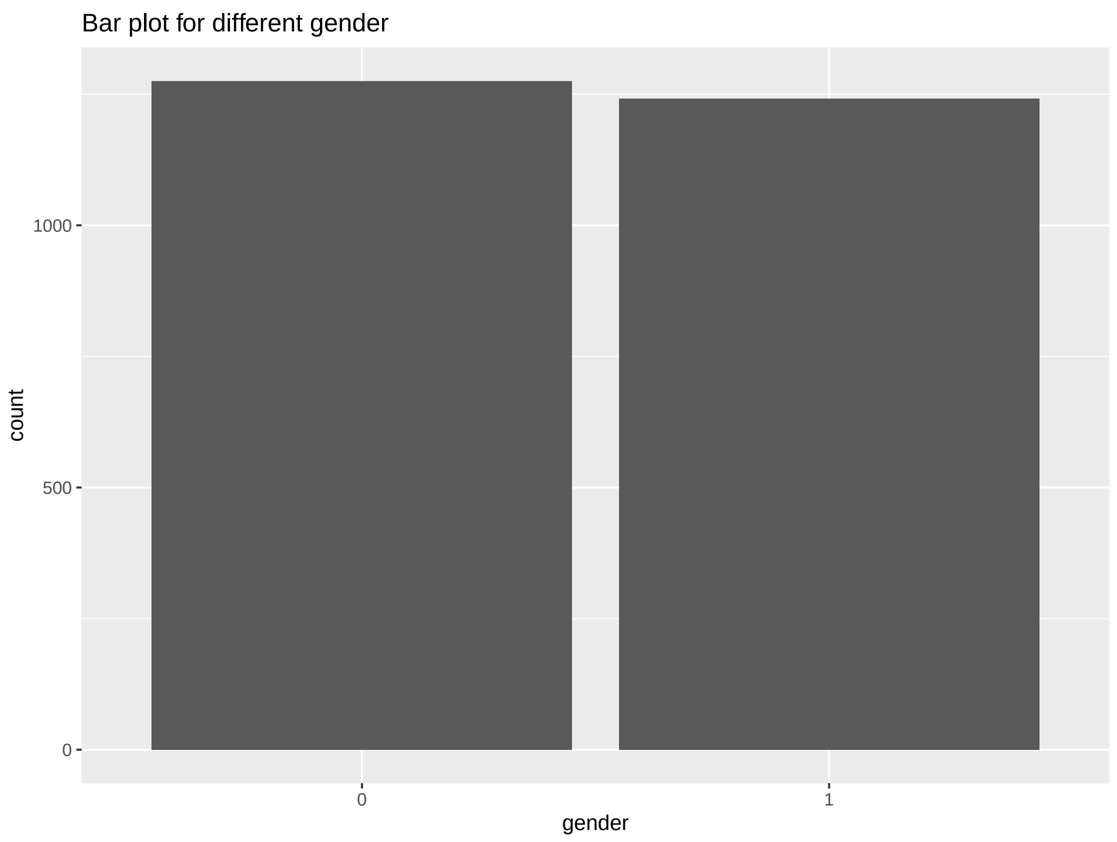
```
ggplot(train_data, aes(x = LDL))+geom_density()+labs(title="Distribution for LDL")
```



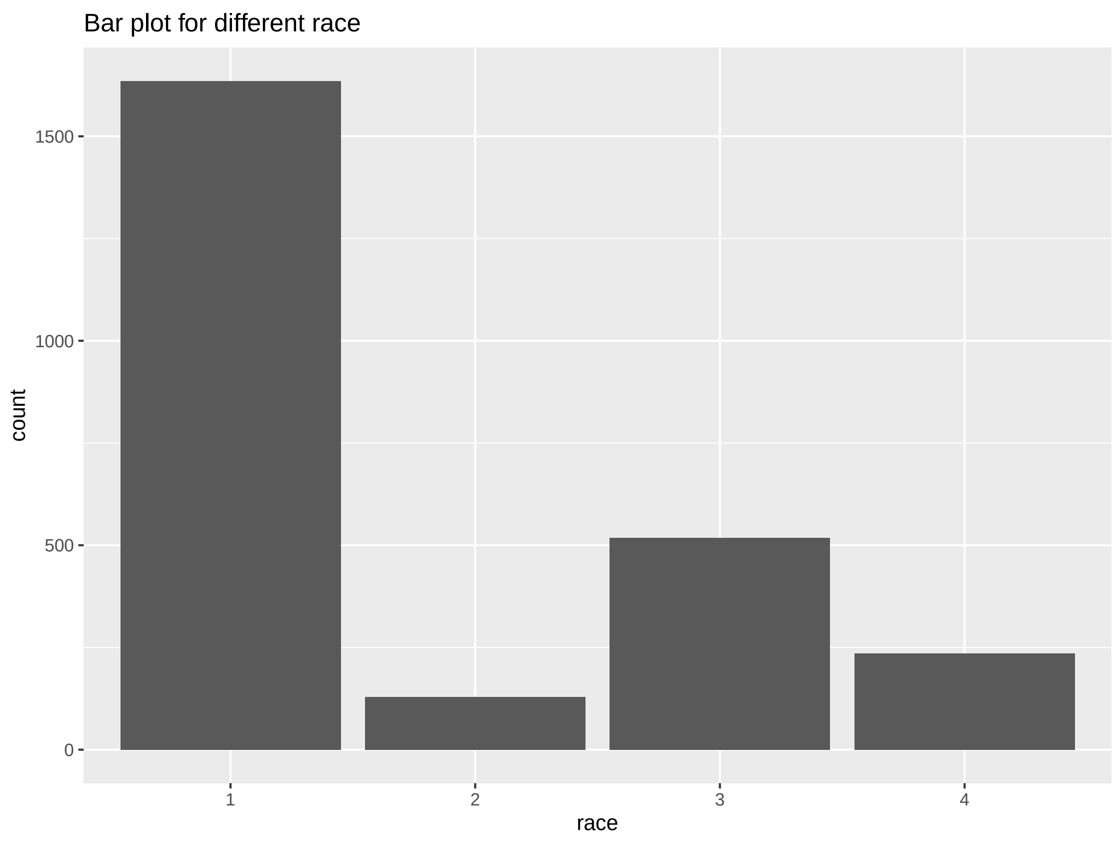
```
ggplot(train_data, aes(x = recovery_time))+geom_density()+labs(title="Distribution for recovery_time")
```



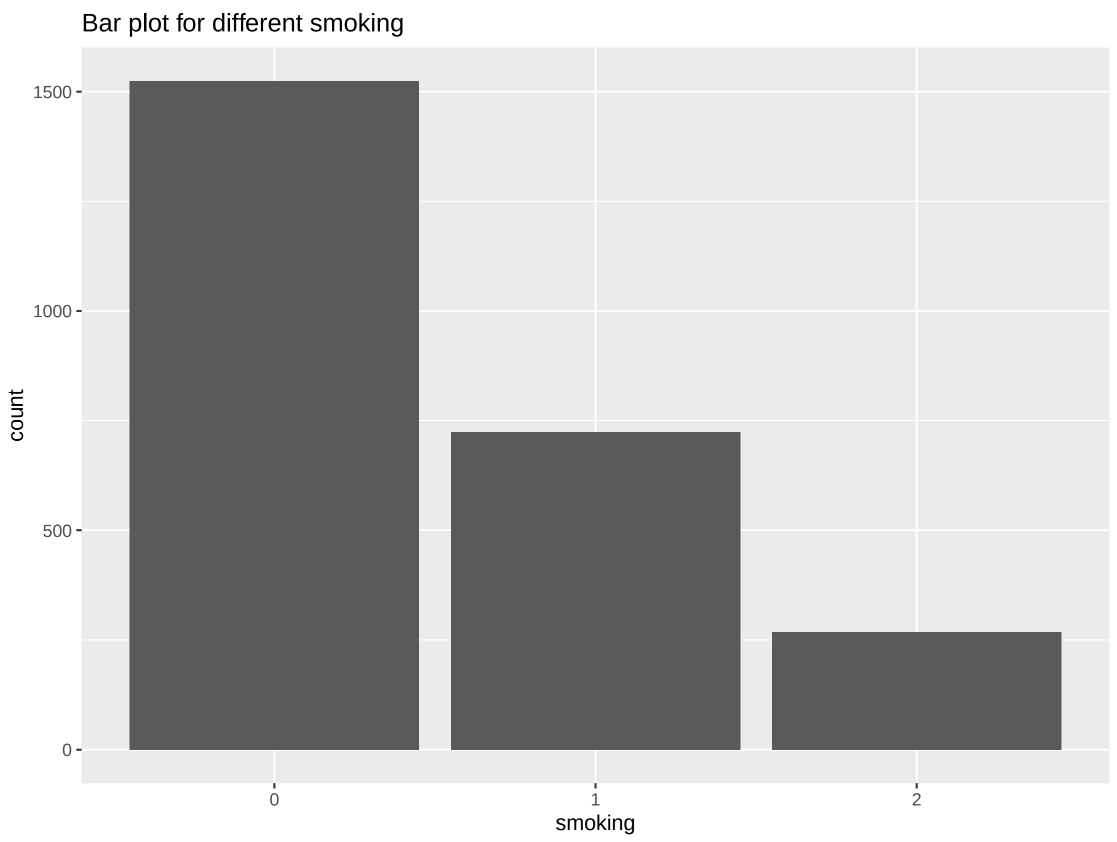
```r
#discrete
ggplot(train_data) + geom_bar(aes(x=gender))+labs(title="Bar plot for different gender")
```



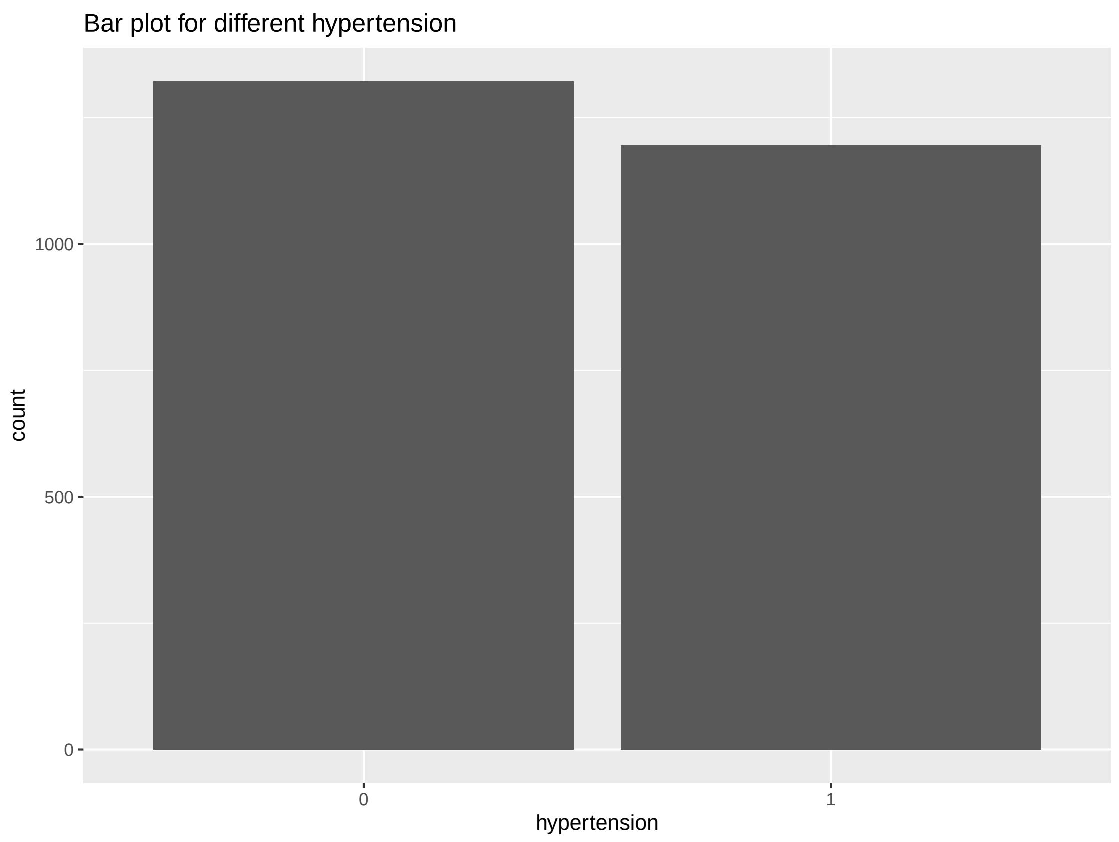
```r
ggplot(train_data) + geom_bar(aes(x=race))+labs(title="Bar plot for different race")
```



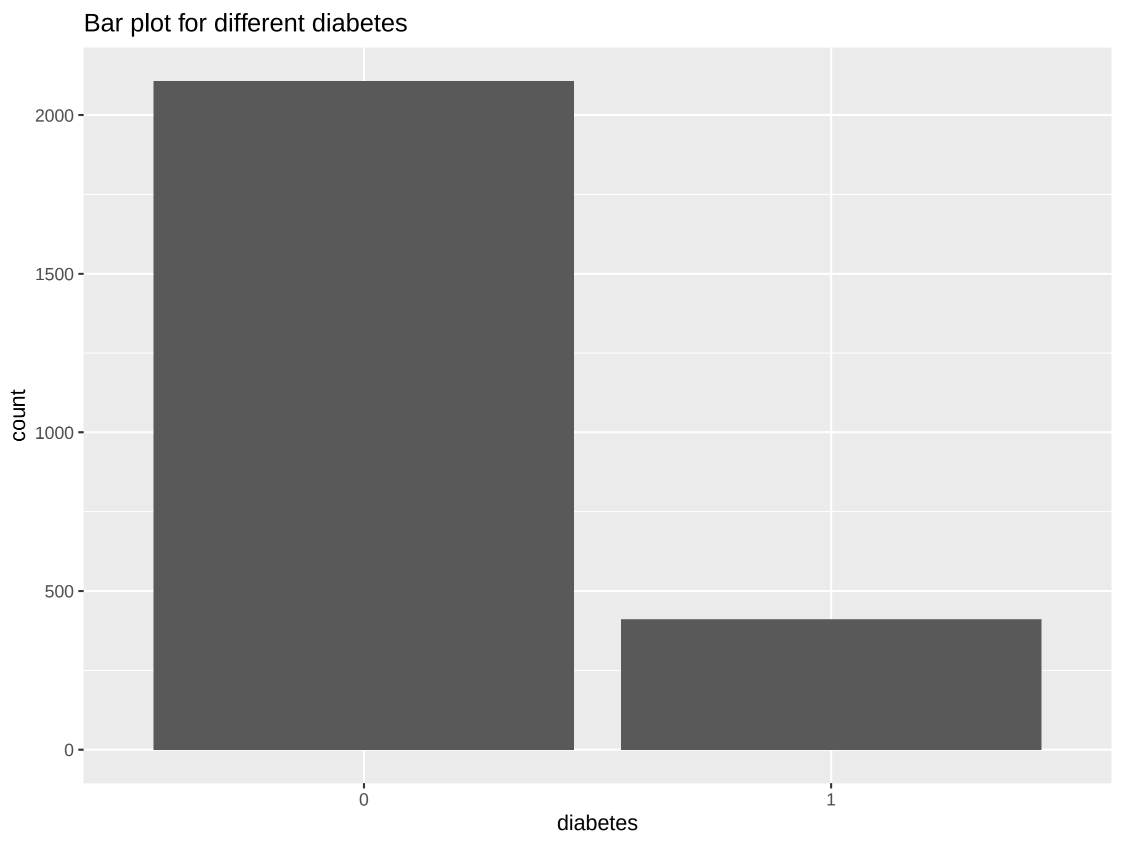
```
ggplot(train_data) + geom_bar(aes(x=smoking))+labs(title="Bar plot for different smoking")
```



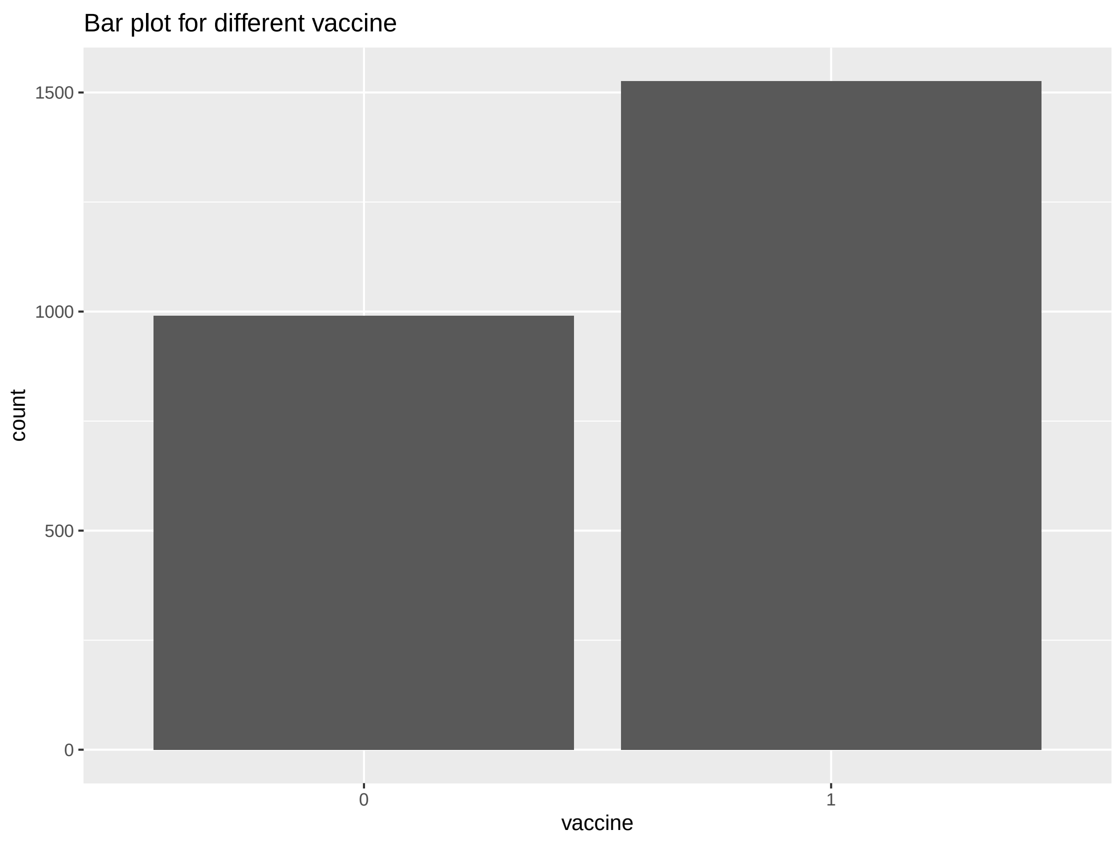
```
ggplot(train_data) + geom_bar(aes(x=hypertension))+labs(title="Bar plot for different hypertension")
```



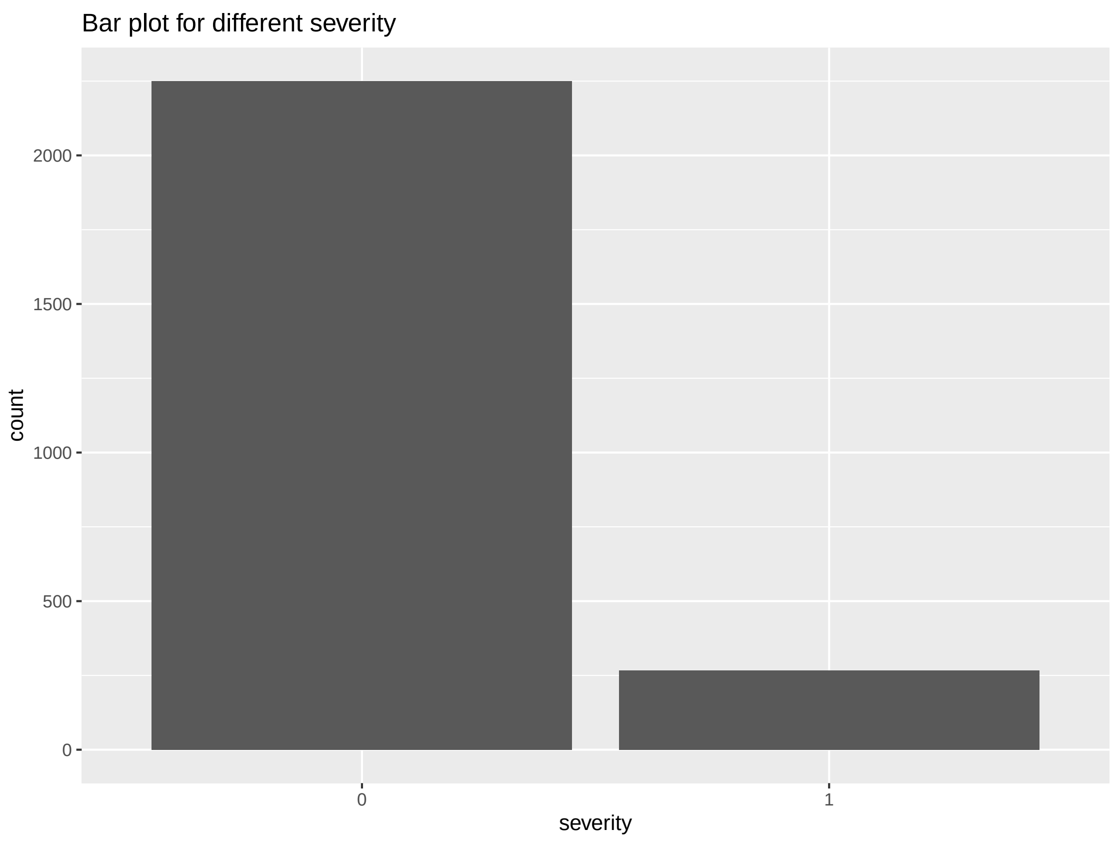
```r
ggplot(train_data) + geom_bar(aes(x=diabetes))+labs(title="Bar plot for different diabetes")
```



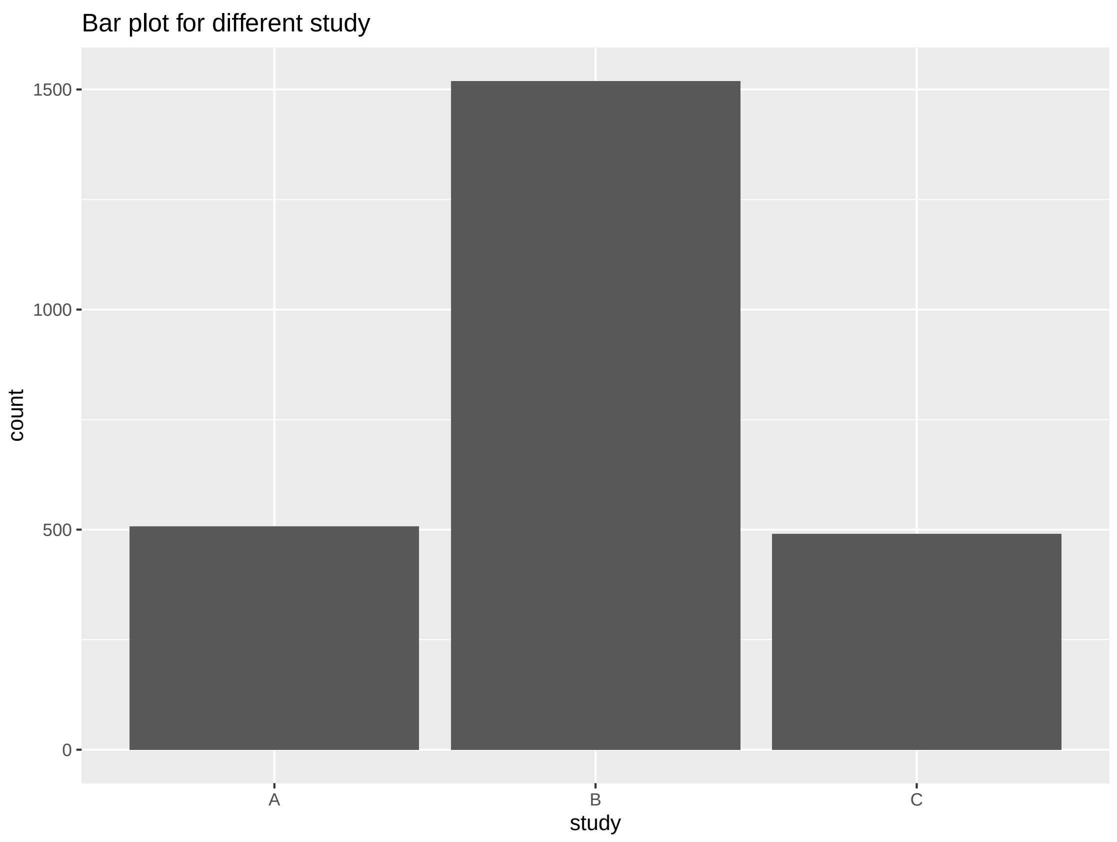
```
ggplot(train_data) + geom_bar(aes(x=vaccine))+labs(title="Bar plot for different vaccine")
```



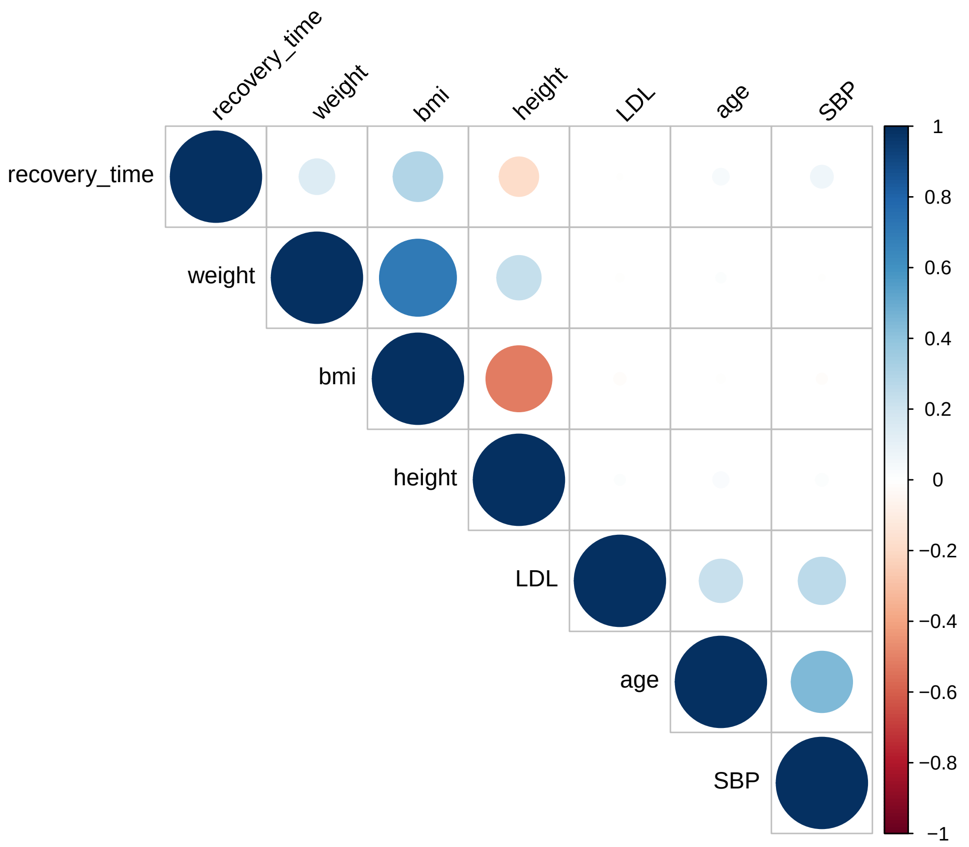
```r
ggplot(train_data) + geom_bar(aes(x=severity))+labs(title="Bar plot for different severity")
```



```
ggplot(train_data) + geom_bar(aes(x=study))+labs(title="Bar plot for different study")
```



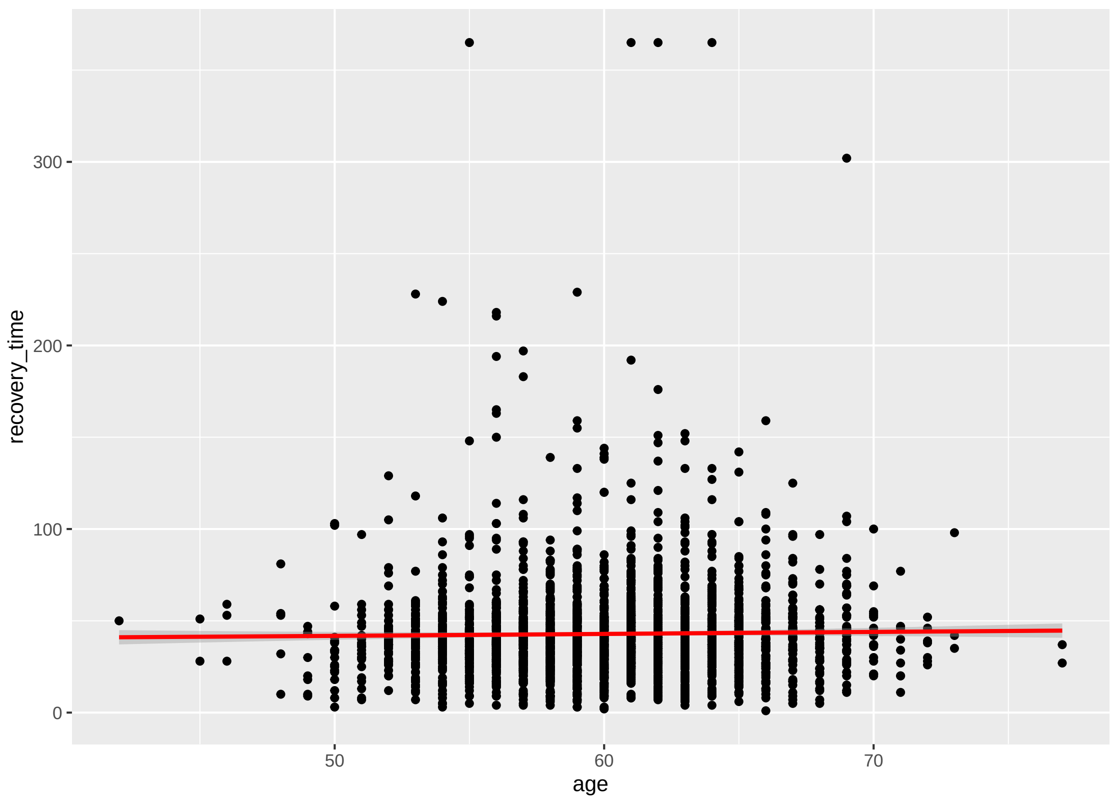
```r
num_df <-
  train_data %>%
  dplyr::select(age, height, weight, bmi, SBP, LDL, recovery_time)


# calulate the correlations
res <- cor(num_df, use="complete.obs")

corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



```
ggplot(train_data, aes(x = age, y = recovery_time))+geom_point()+geom_smooth(method = 'gam', se = TRUE,
```

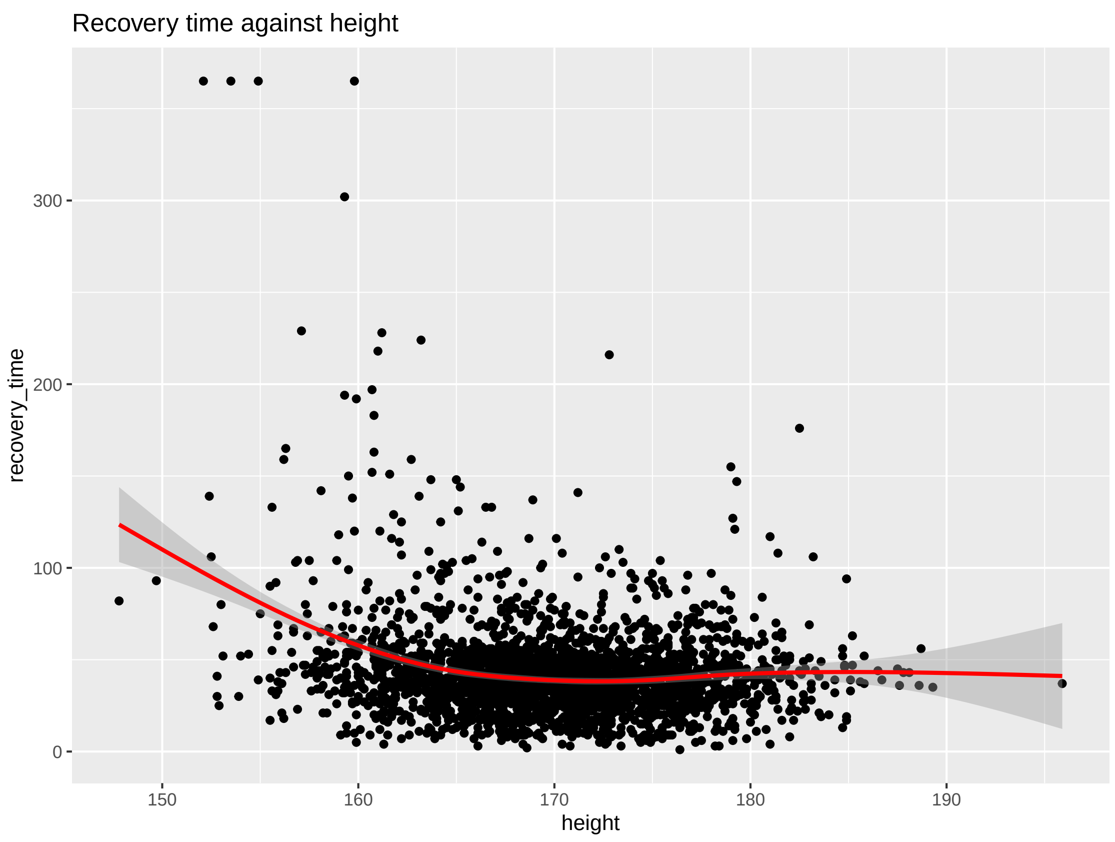
```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```

Recovery time against age



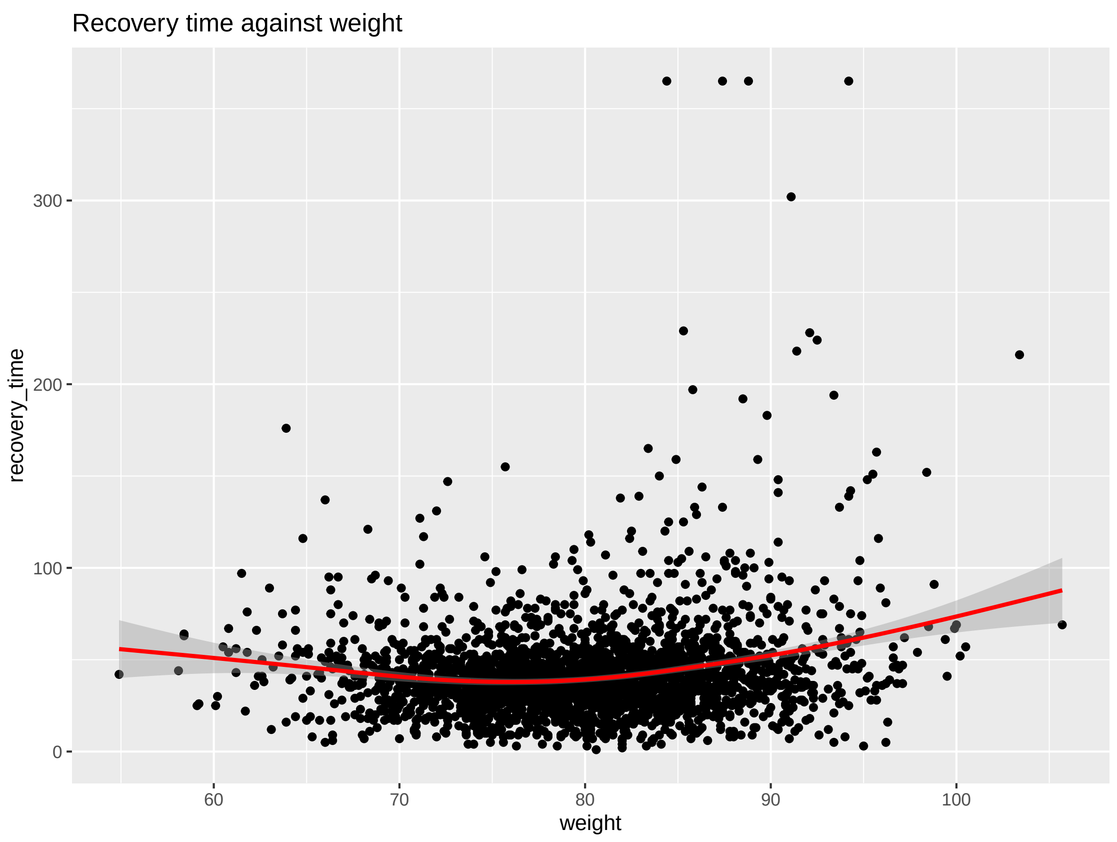
```
ggplot(train_data, aes(x = height, y = recovery_time))+geom_point()+geom_smooth(method = 'gam', se = TRU
```

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```



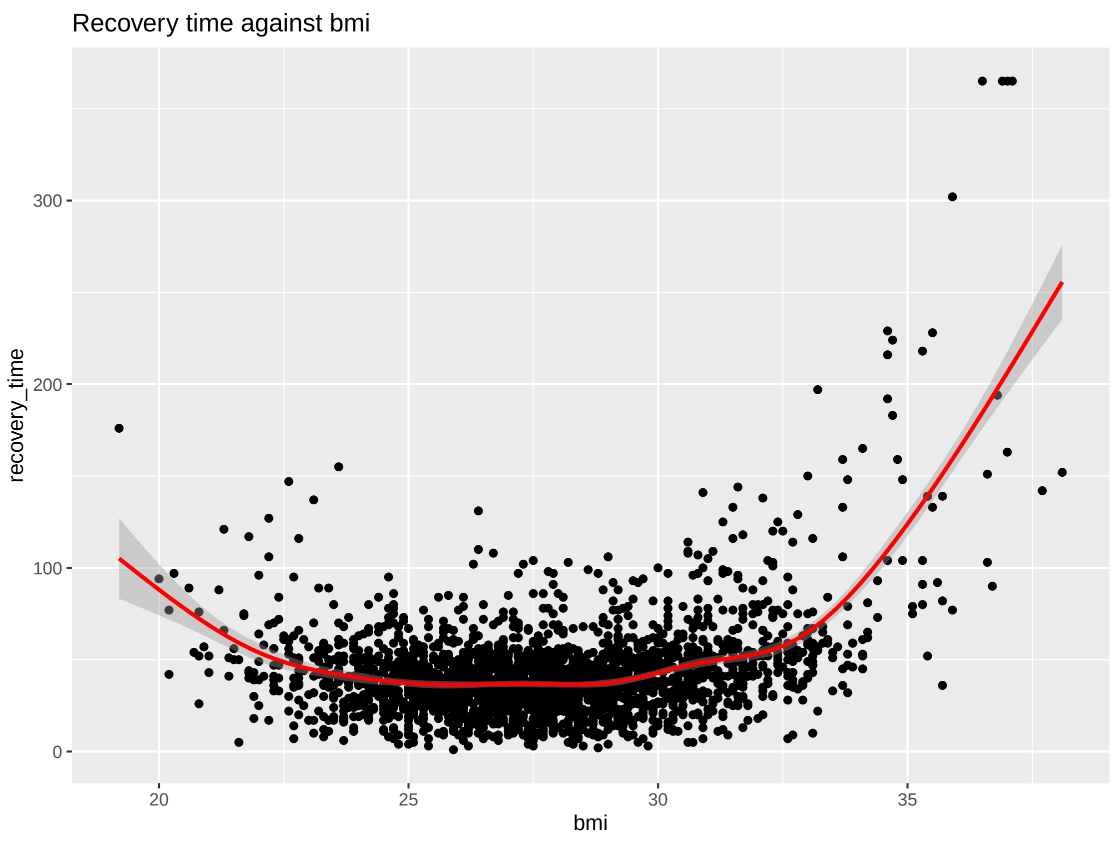
```
ggplot(train_data, aes(x = weight, y = recovery_time))+geom_point()+geom_smooth(method = 'gam', se = TRU
```

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```



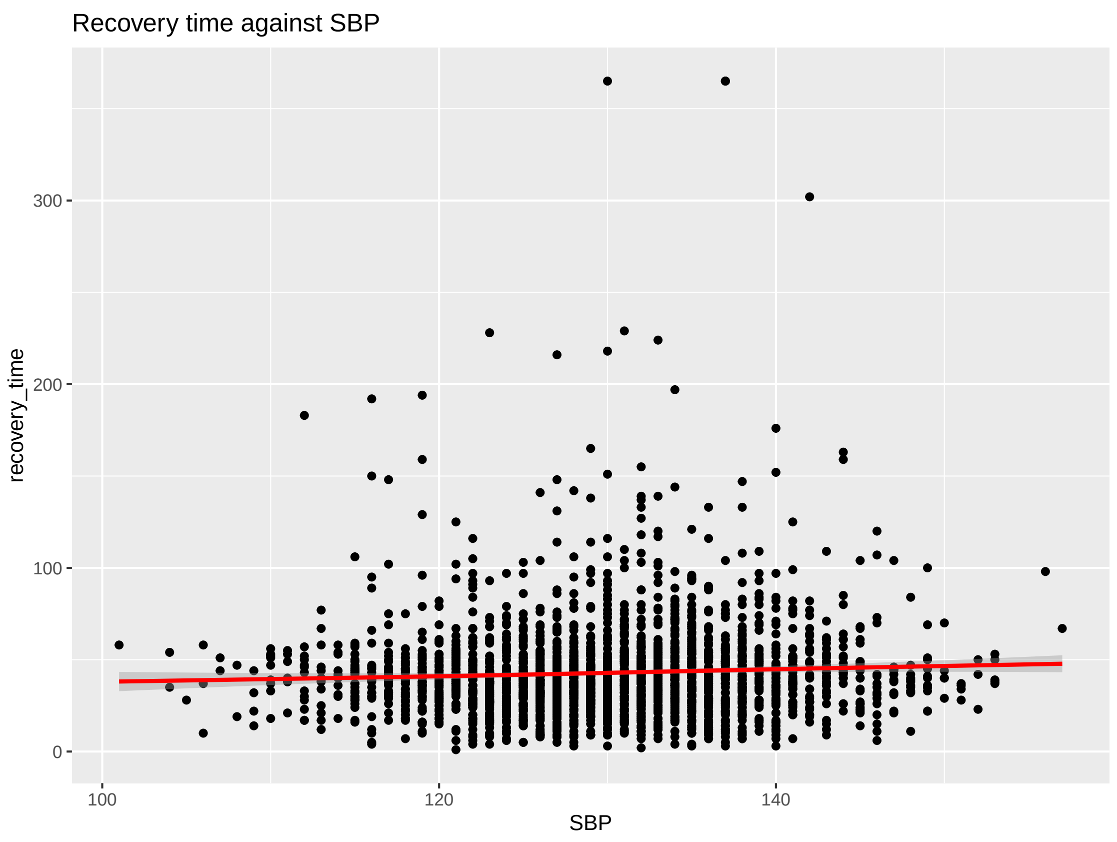
```
ggplot(train_data, aes(x = bmi, y = recovery_time))+geom_point()+geom_smooth(method = 'gam', se = TRUE,
```

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```



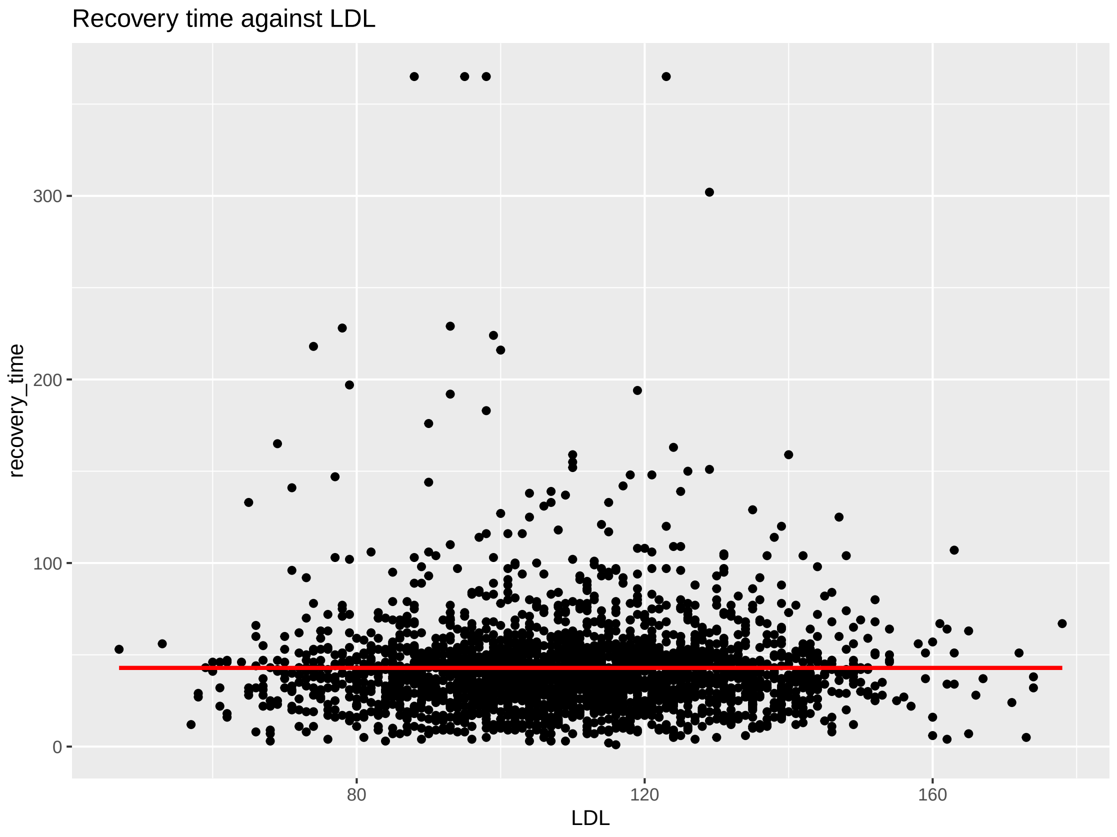
```
ggplot(train_data, aes(x = SBP, y = recovery_time))+geom_point()+geom_smooth(method = 'gam', se = TRUE,
```

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```



```
ggplot(train_data, aes(x = LDL, y = recovery_time))+geom_point()+geom_smooth(method = 'gam', se = TRUE,
```

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```



Model training

Resampling method

```
# repeated 10-fold cv
ctrl <- trainControl(method = "cv",
                     number = 10)
```

primary analysis (Regression)

- Linear regression
- Partial least squares (PLS)
- Generalized Additive Model (GAM)
- Multivariate Adaptive Regression Splines (MARS)
- Elastic net
- Boosting
- Random forest

```
## fit linear model on train data
set.seed(8)
linear_model <- train(train_x,
                      train_y,
                      method = "lm",
                      trControl = ctrl)
summary(linear_model)
```

Linear regression

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.848 -13.472  -1.414   9.399 257.759
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2.743e+03  1.295e+02 -21.171  < 2e-16 ***
## age           -1.353e-02  1.177e-01  -0.115 0.908505
## gender1       -4.538e+00  9.432e-01  -4.812 1.59e-06 ***
## race2          5.460e-01  2.166e+00   0.252 0.800997
## race3         -1.639e+00  1.193e+00  -1.374 0.169550
## race4         -9.671e-01  1.653e+00  -0.585 0.558518
## smoking1       3.052e+00  1.066e+00   2.863 0.004236 **
## smoking2       7.843e+00  1.563e+00   5.018 5.58e-07 ***
## height         1.608e+01  7.612e-01  21.120  < 2e-16 ***
## weight        -1.733e+01  8.043e-01 -21.541  < 2e-16 ***
## bmi            5.195e+01  2.300e+00  22.587  < 2e-16 ***
## hypertension1  3.229e+00  1.560e+00   2.070 0.038547 *
## diabetes1     -2.210e-01  1.277e+00  -0.173 0.862608
## SBP            3.232e-04  1.012e-01   0.003 0.997453
## LDL           -1.013e-02  2.499e-02  -0.405 0.685287
## vaccine1      -8.167e+00  9.684e-01  -8.434  < 2e-16 ***
## severity1      5.717e+00  1.534e+00   3.726 0.000199 ***
## studyB         4.315e+00  1.215e+00   3.552 0.000389 ***
## studyC        -3.479e-01  1.500e+00  -0.232 0.816566
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.58 on 2498 degrees of freedom
## Multiple R-squared:  0.2754, Adjusted R-squared:  0.2702
## F-statistic: 52.75 on 18 and 2498 DF,  p-value: < 2.2e-16
```

```r
# view performance on the test set (RMSE)
lm_test_pred <- predict(linear_model, newdata = test_x) # test dataset
lm_test_rmse <- sqrt(mean((lm_test_pred - test_y)^2))
sprintf("test error for lm is: %.3f",lm_test_rmse)
```

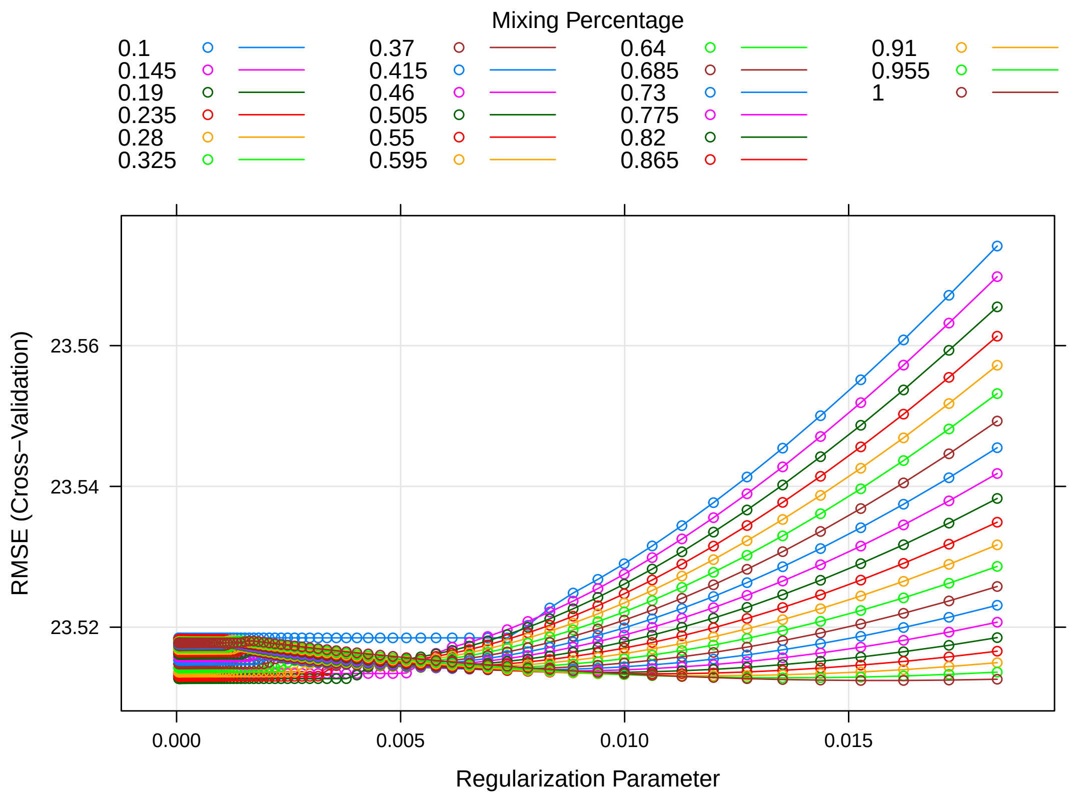
```
## [1] "test error for lm is: 23.569"
```

Enet

```r
#tuning glmnet alpha 0 and 0.05 have large RMSE. So alpha start from 0.1
set.seed(8)
enet.fit.min <- train(train_x, train_y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0.1, 1, length = 21),
                                         lambda = exp(seq(-10, -4, length = 100))),
                  trControl = ctrl)
enet.fit.min$bestTune
```

```
##      alpha     lambda
## 2097     1 0.01527072
```

```
plot(enet.fit.min)
```

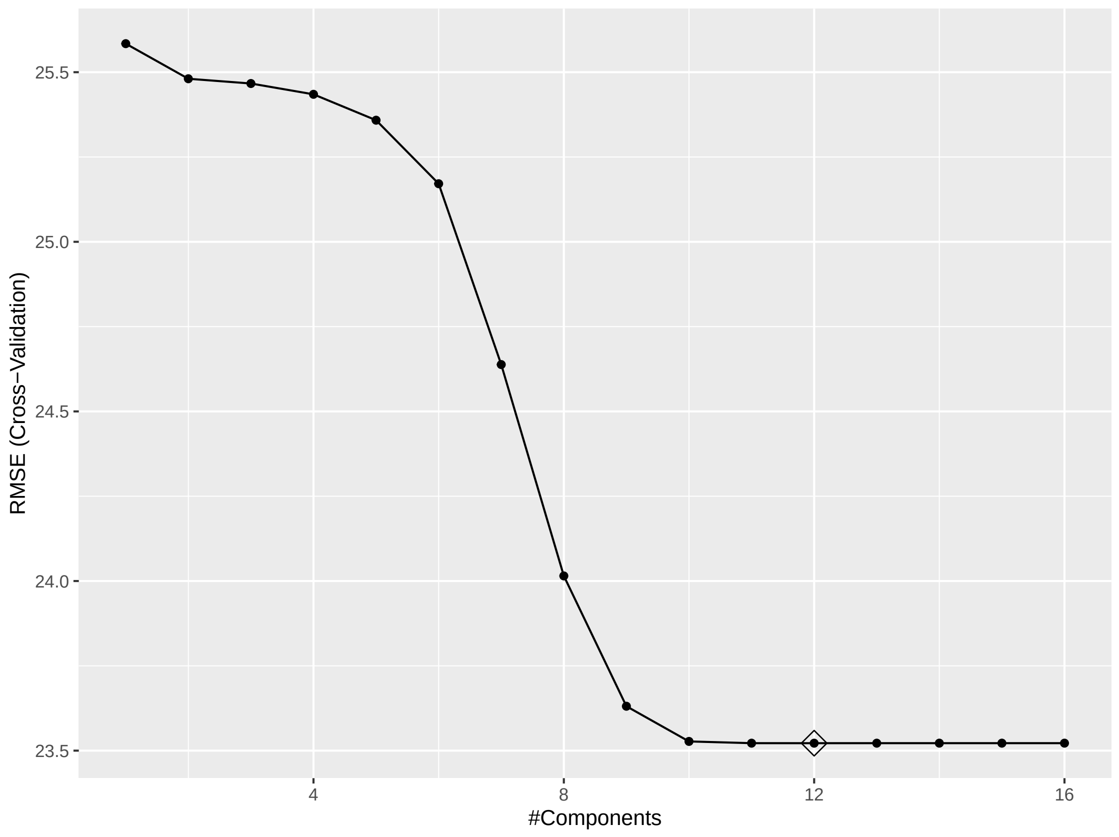


```
# view performance on the test set (RMSE)
enet_test_pred <- predict(enet.fit.min, newdata = test_x) # test dataset
enet_test_rmse <- sqrt(mean((enet_test_pred - test_y)^2))
sprintf("test error for enet is: %.3f",enet_test_rmse)
```

```
## [1] "test error for enet is: 23.525"
```

```
set.seed(8)
pls_model <- train(train_x,
                   train_y,
                   method = "pls",
                   tuneGrid = data.frame(ncomp = 1:16),
                   trControl = ctrl,
                   preProcess = c("center", "scale"))
ggplot(pls_model, highlight = TRUE)
```

PLS



```
ggsave(file = "image/pls_number_of_component.png", width = 10, height = 7)

# view performance on the test set (RMSE)
pls_test_pred <- predict(pls_model, newdata = test_x) # test dataset
pls_test_rmse <- sqrt(mean((pls_test_pred - test_y)^2))
sprintf("test error for pls is: %.3f",pls_test_rmse)
```

```
## [1] "test error for pls is: 23.569"
```

Modeling Strategy

- **Penalized logistic regression**
- **Generalized additive model (GAM)**
- **Multivariate adaptive regression splines (MARS)**
- **Linear discriminant analysis (LDA)**
- **Random Forest**
- **Boosting**

Optimal Tuning Parameters

Resampling Results and Model Selection

secondary analysis (Classification)

```

#consider time to recovery as a binary outcome (>30 days vs. <= 30 days) and develop a prediction model
train_class =
```

```r
  train_data %>%
  mutate(recovery_time = ifelse(recovery_time>30, "long", "short"))

test_class =
  test_data %>%
  mutate(recovery_time = ifelse(recovery_time>30, "long", "short"))

# matrix of predictors
x_train_class <- model.matrix(recovery_time~.,train_class)[,-1]
# vector of response
y_train_class <- train_class$recovery_time


# matrix of predictors
x_test_class <- model.matrix(recovery_time~.,test_class)[,-1]
# vector of response
y_test_class <- test_class$recovery_time


# ctrl for class
ctrl_class <- trainControl(method = "cv",
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)
```

Data Manipulation

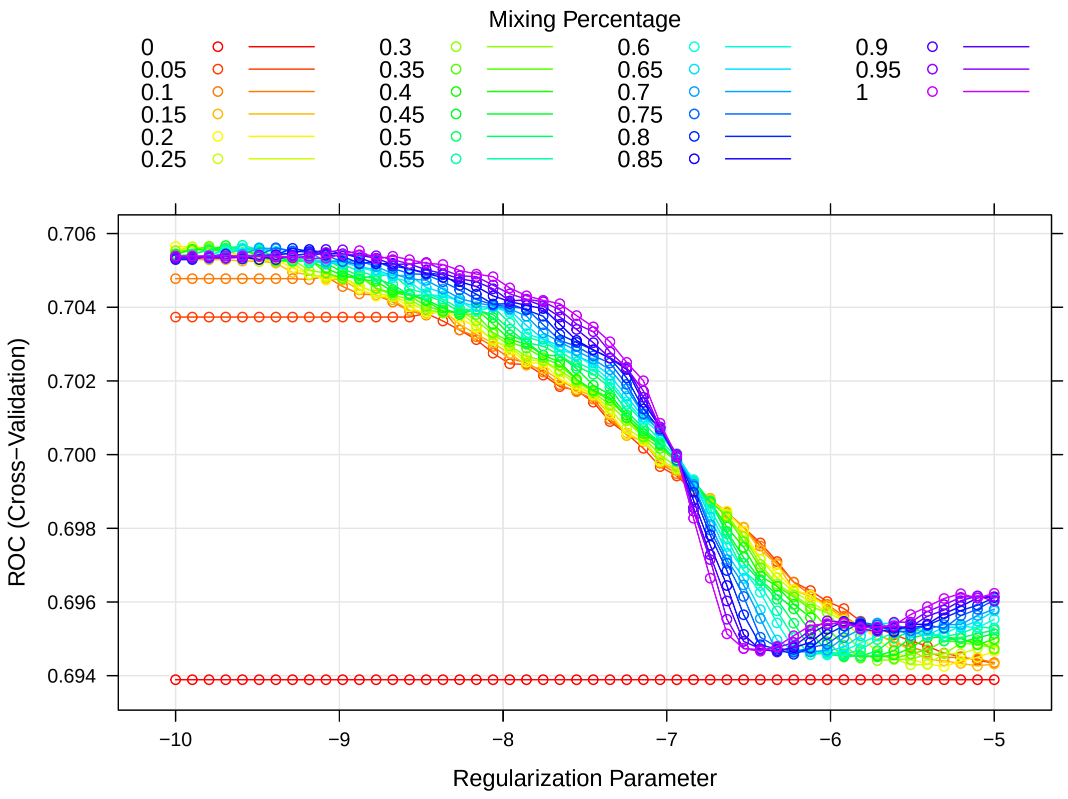
Modeling Strategy

- **Penalized logistic regression**

```r
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-10, -5, length = 50)))
set.seed(8)
model.glmn <- train(x = x_train_class,
                    y = y_train_class,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl_class)

model.glmn$bestTune
```

```
##     alpha       lambda
## 555  0.55 6.828389e-05
```

```r
myCol<- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(model.glmn, par.settings = myPar, xTrans = function(x) log(x))
```



```r
# view performance on the test set (accuracy)
glmn_class_test_pred <- predict(model.glmn, newdata=x_test_class) # test dataset
glmn_class_test_acc <- sum(glmn_class_test_pred==y_test_class)/length(y_test_class)
sprintf("test error for penalized logistic is: %.3f",1-glmn_class_test_acc)
```

```
## [1] "test error for penalized logistic is: 0.286"
```

Since the boundary value for λ is exp(-10)=4.54e-5 and exp(-5)=0.007, and the optimal lambda is not on the boundary. So local minimum is achieved.

- **Generalized additive model (GAM)**
- **Multivariate adaptive regression splines (MARS)**
- **Linear discriminant analysis (LDA)**
- **Classification and Regression Tree (CART)**
- **Random Forest**
- **Boosting**
- **Support Vector Machine (SVM with Linear and Radial Kernels)**

Optimal Tuning Parameters

Resampling Results and Model Selection

Results

primary analysis

Interpretation

Variable Importance

model's training/test performance

secondary analysis

Interpretion

Variable Importance

model's training/test performance

Conclusion

Appendix