Statistical Learning Project

Sarah, Pavel, Rose, Catherine, Shravya East Statistical Learning Project

```
#load the necessary packages
library(plyr)
library(tidyverse)
library(reshape2)
library(readxl)
library(caret)
library(rpart)
library(partykit)
library(randomForest)
library(class)
library (rminer)
library(e1071)
library(mlbench)
library(plyr)
library(DMwR)

#Read in the data
dat <- read_excel("Absenteeism_at_work.xls")

#View the data
glimpse(dat)

## Observations: 740
## Variables: 21
## $ ID                              <dbl> 11, 36, 3, 7, 11, 3, 10, 20,...
## $ `Reason for absence`            <dbl> 26, 0, 23, 7, 23, 23, 22, 23...
## $ `Month of absence`              <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7...
## $ `Day of the week`               <dbl> 3, 3, 4, 5, 5, 6, 6, 6, 2, 2...
## $ Seasons                         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ `Transportation expense`        <dbl> 289, 118, 179, 279, 289, 179...
## $ `Distance from Residence to Work` <dbl> 36, 13, 51, 5, 36, 51, 52, 5...
## $ `Service time`                  <dbl> 13, 18, 18, 14, 13, 18, 3, 1...
## $ Age                             <dbl> 33, 50, 38, 39, 33, 38, 28, ...
## $ `Work load Average/day`         <dbl> 239554, 239554, 239554, 2395...
## $ `Hit target`                    <dbl> 97, 97, 97, 97, 97, 97, 97, ...
## $ `Disciplinary failure`          <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Education                       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 3...
## $ Son                             <dbl> 2, 1, 0, 2, 2, 0, 1, 4, 2, 1...
## $ `Social drinker`                <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0...
## $ `Social smoker`                 <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0...
## $ Pet                             <dbl> 1, 0, 0, 0, 1, 0, 4, 0, 0, 1...
## $ Weight                          <dbl> 90, 98, 89, 68, 90, 89, 80, ...
## $ Height                          <dbl> 172, 178, 170, 168, 172, 170...
```

```
## $ `Body mass index`                   <dbl> 30, 31, 31, 24, 30, 31, 27, ...
## $ `Absenteeism time in hours`         <dbl> 4, 0, 2, 4, 2, 2, 8, 4, 40, ...
```

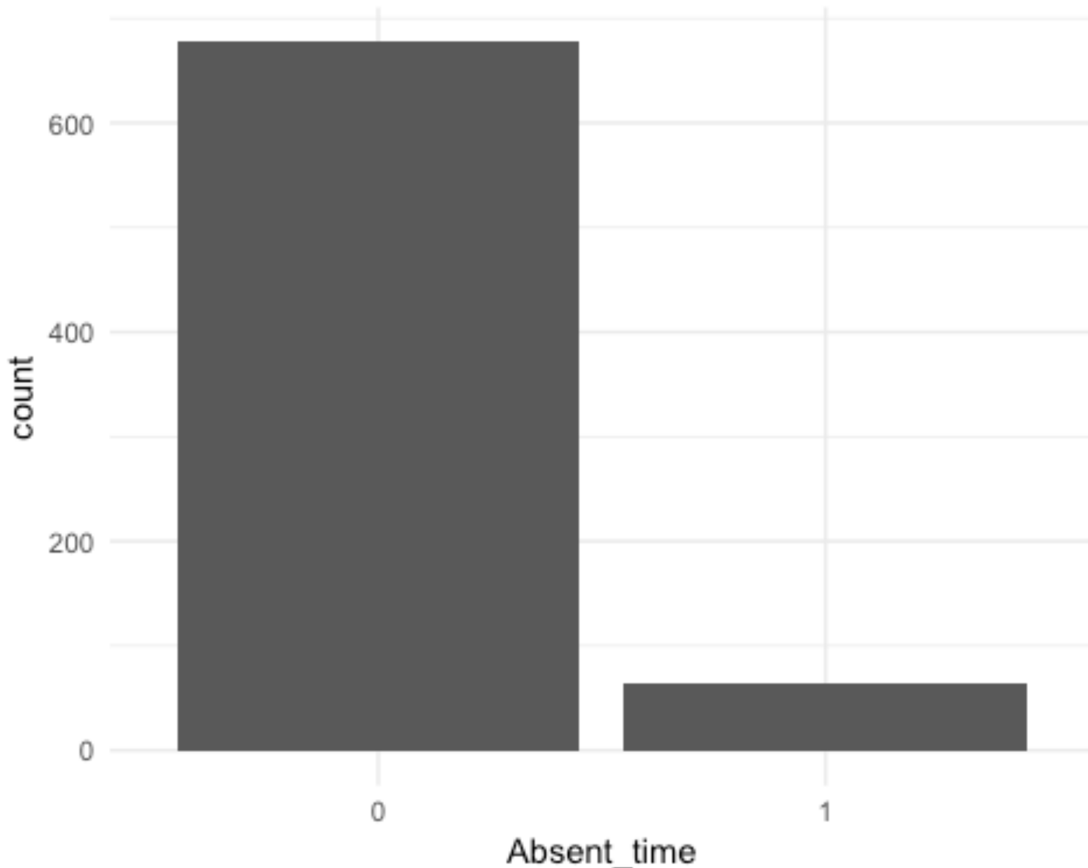
Pre-Processing Data

```
#Set factored variables as factors
col <- c("ID", "Reason for absence", "Month of absence", "Day of the week",
"Seasons", "Disciplinary failure", "Education", "Social drinker",   "Social
smoker")
#set all categorical variables as ordered factors
dat[col] <- lapply(dat[col], as.factor)
dat[col] <- lapply(dat[col], ordered)

#Rename the columns for easier use
colnames(dat) <- c("ID", "Reason", "Month", "Day", "Seasons",
"Transportation_expense", "Distance", "Service_time", "Age", "Work_load",
"Hit_target", "Disciplinary_failure", "Education", "Children",
"Social_drinker", "Social_smoker", "Pet", "Weight", "Height", "BMI",
"Absent_time")

#View the data
glimpse(dat)

## Observations: 740
## Variables: 21
## $ ID                     <ord> 11, 36, 3, 7, 11, 3, 10, 20, 14, 1, 20,...
## $ Reason                 <ord> 26, 0, 23, 7, 23, 23, 22, 23, 19, 22, 1...
## $ Month                  <ord> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, ...
## $ Day                    <ord> 3, 3, 4, 5, 5, 6, 6, 6, 2, 2, 2, 3, 4, ...
## $ Seasons                <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Transportation_expense <dbl> 289, 118, 179, 279, 289, 179, 361, 260,...
## $ Distance               <dbl> 36, 13, 51, 5, 36, 51, 52, 50, 12, 11, ...
## $ Service_time           <dbl> 13, 18, 18, 14, 13, 18, 3, 11, 14, 14, ...
## $ Age                    <dbl> 33, 50, 38, 39, 33, 38, 28, 36, 34, 37,...
## $ Work_load              <dbl> 239554, 239554, 239554, 239554, 239554,...
## $ Hit_target             <dbl> 97, 97, 97, 97, 97, 97, 97, 97, 97, 97,...
## $ Disciplinary_failure   <ord> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ Education              <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ...
## $ Children               <dbl> 2, 1, 0, 2, 2, 0, 1, 4, 2, 1, 4, 4, 4, ...
## $ Social_drinker         <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, ...
## $ Social_smoker          <ord> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ Pet                    <dbl> 1, 0, 0, 0, 1, 0, 4, 0, 0, 1, 0, 0, 0, ...
## $ Weight                 <dbl> 90, 98, 89, 68, 90, 89, 80, 65, 95, 88,...
## $ Height                 <dbl> 172, 178, 170, 168, 172, 170, 172, 168,...
## $ BMI                    <dbl> 30, 31, 31, 24, 30, 31, 27, 23, 25, 29,...
## $ Absent_time            <dbl> 4, 0, 2, 4, 2, 2, 8, 4, 40, 8, 8, 8, 8,...

#create a list of the numeric variables in the data set
nums <- unlist(lapply(dat, is.numeric))
#create a smaller data set of just numeric variables
dat.num <- dat[ , nums]
```

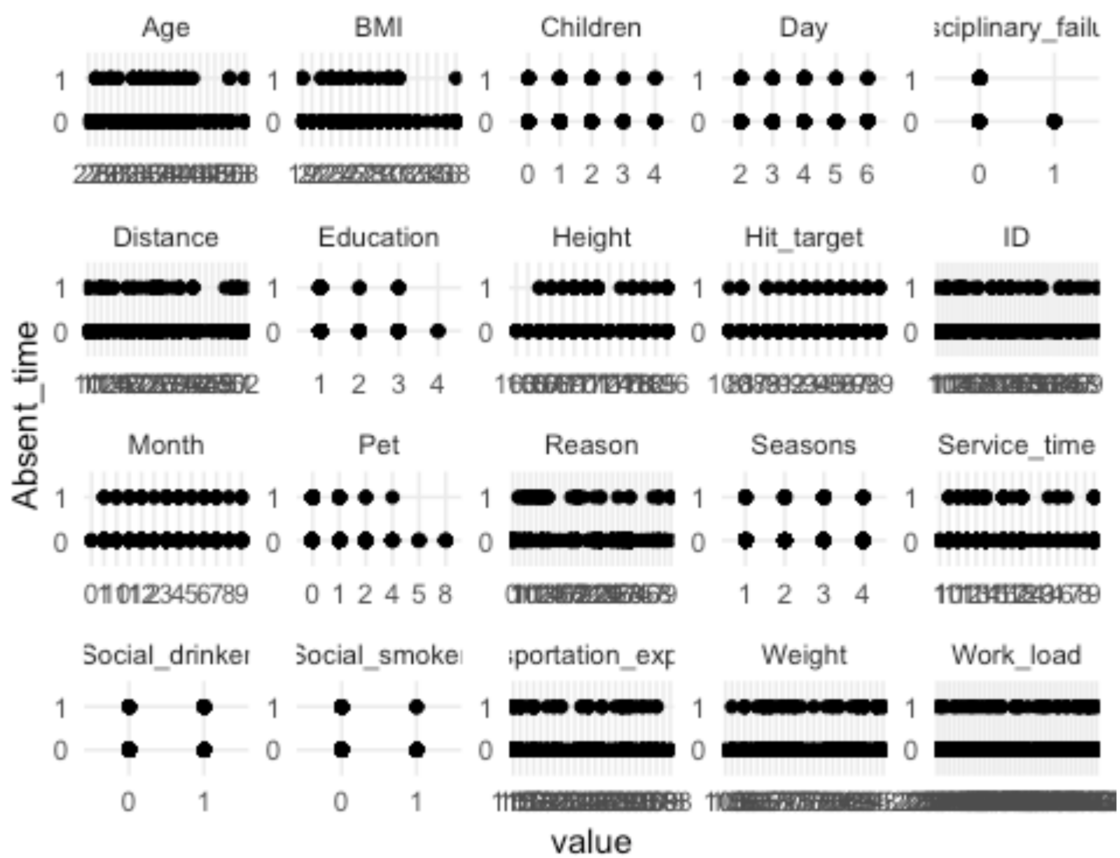
EDA Response Variable

Absent_time

```r
summary(dat$Absent_time)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   6.924   8.000 120.000

dat %>%
  count(Absent_time)

## # A tibble: 19 x 2
##    Absent_time     n
##          <dbl> <int>
##  1           0    44
##  2           1    88
##  3           2   157
##  4           3   112
##  5           4    60
##  6           5     7
##  7           7     1
##  8           8   208
##  9          16    19
## 10          24    16
## 11          32     6
## 12          40     7
## 13          48     1
## 14          56     2
## 15          64     3
## 16          80     3
## 17         104     1
## 18         112     2
## 19         120     3

#change variable represent missed time one day or greater
dat <- dat %>%
  mutate(Absent_time = ifelse(dat$Absent_time <= 8,0,1))

#save Absent_time as a factor in the data set
dat$Absent_time <- as.factor(dat$Absent_time)
#Transforming to Data Frame
dat <- as.data.frame(dat)

#plot the Absent_time
ggplot(data = dat,
       aes(x = Absent_time)) +
  geom_bar() +
  theme_minimal()
```




```r
#plot all variables vs. Absent_time
dat %>%
  gather(-Absent_time, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Absent_time)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



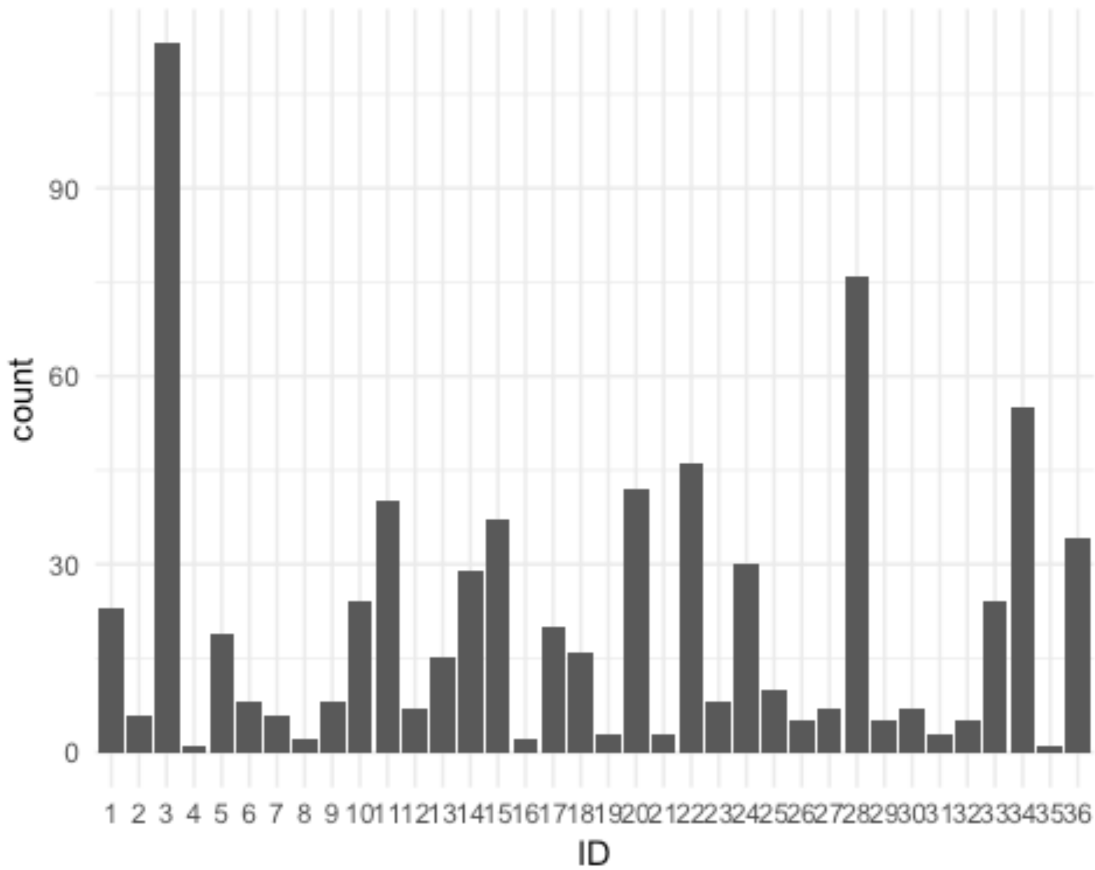
EDA Predictors

ID

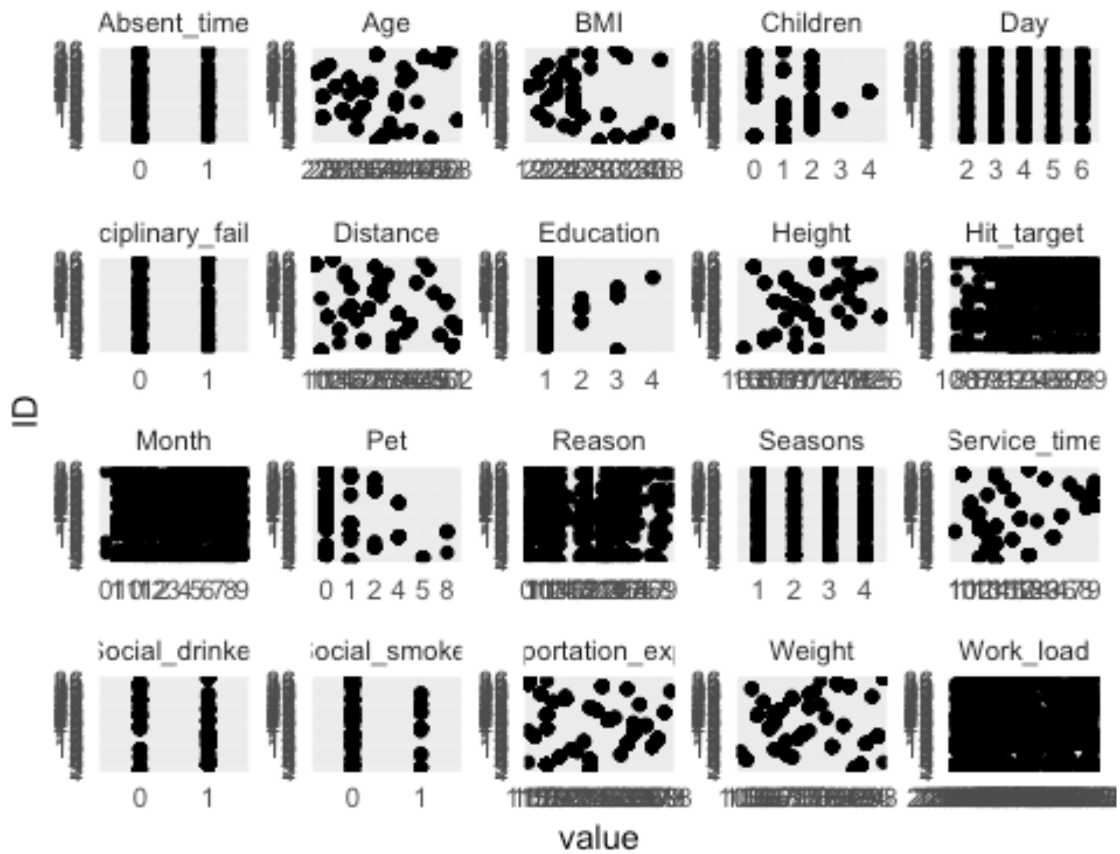
```
#frequency table by ID
dat %>%
  count(ID)
```

```
## # A tibble: 36 x 2
##    ID        n
##    <ord> <int>
##  1 1        23
##  2 2         6
##  3 3       113
##  4 4         1
##  5 5        19
##  6 6         8
##  7 7         6
##  8 8         2
##  9 9         8
## 10 10       24
## # ... with 26 more rows
```

```r
#bar chart
dat %>%
  ggplot(aes(x = ID)) +
  geom_bar() +
  theme_minimal()
```



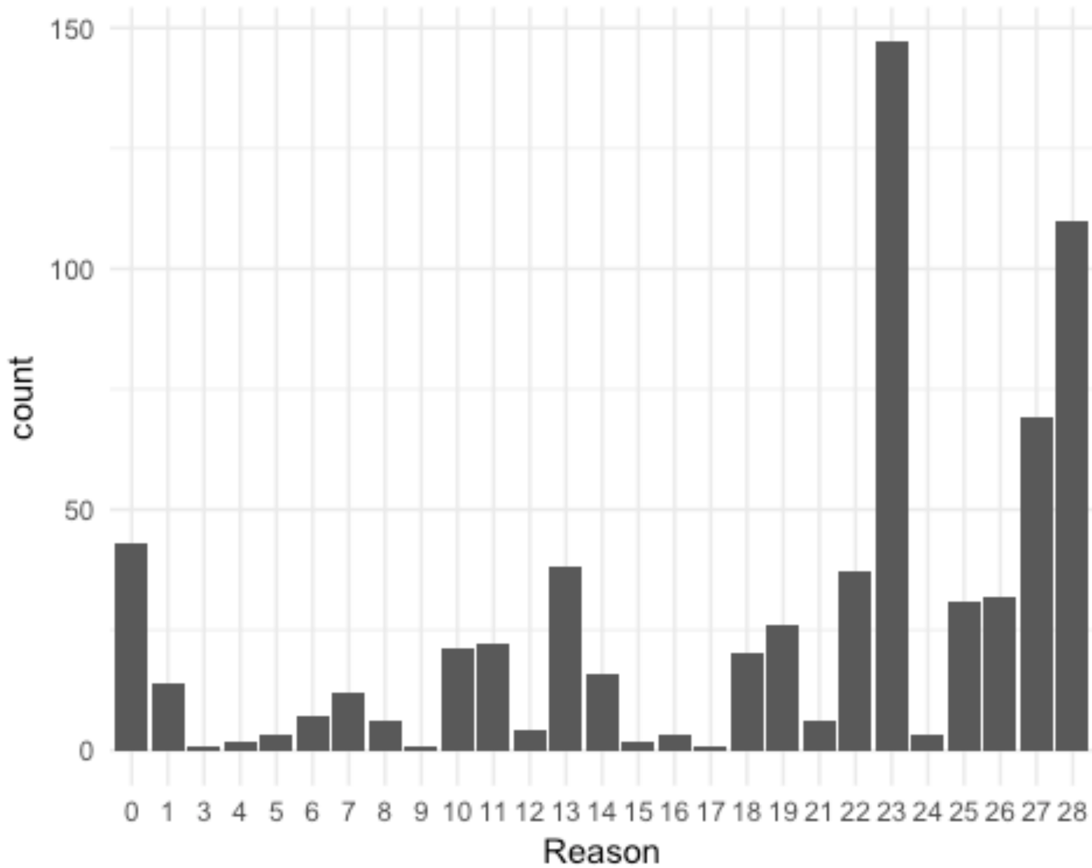

```r
#ID
dat %>%
  gather(-ID, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = ID)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



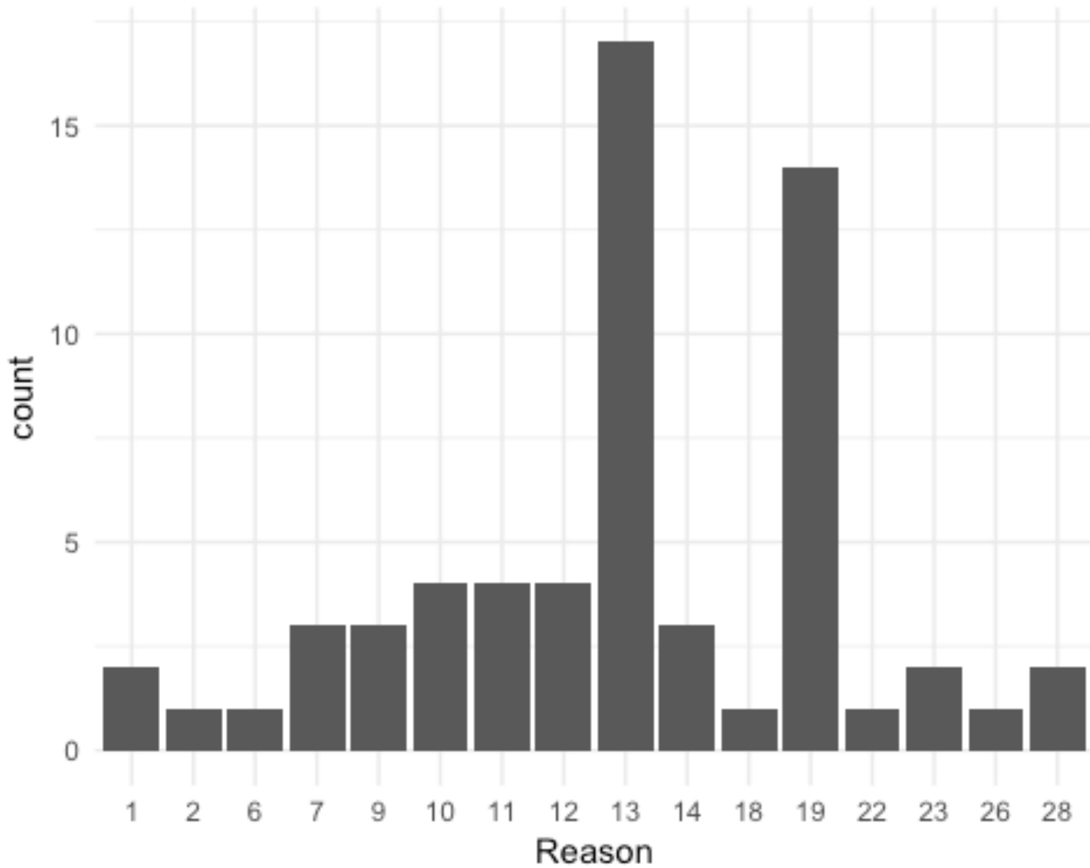
Reason

```
#frequency table by Reason for Absence
dat %>%
  count(Reason)

## # A tibble: 28 x 2
##    Reason     n
##    <ord>  <int>
##  1 0         43
##  2 1         16
##  3 2          1
##  4 3          1
##  5 4          2
##  6 5          3
##  7 6          8
##  8 7         15
##  9 8          6
## 10 9          4
## # ... with 18 more rows

#bar chart
dat %>%
  filter(Absent_time==0) %>%
```

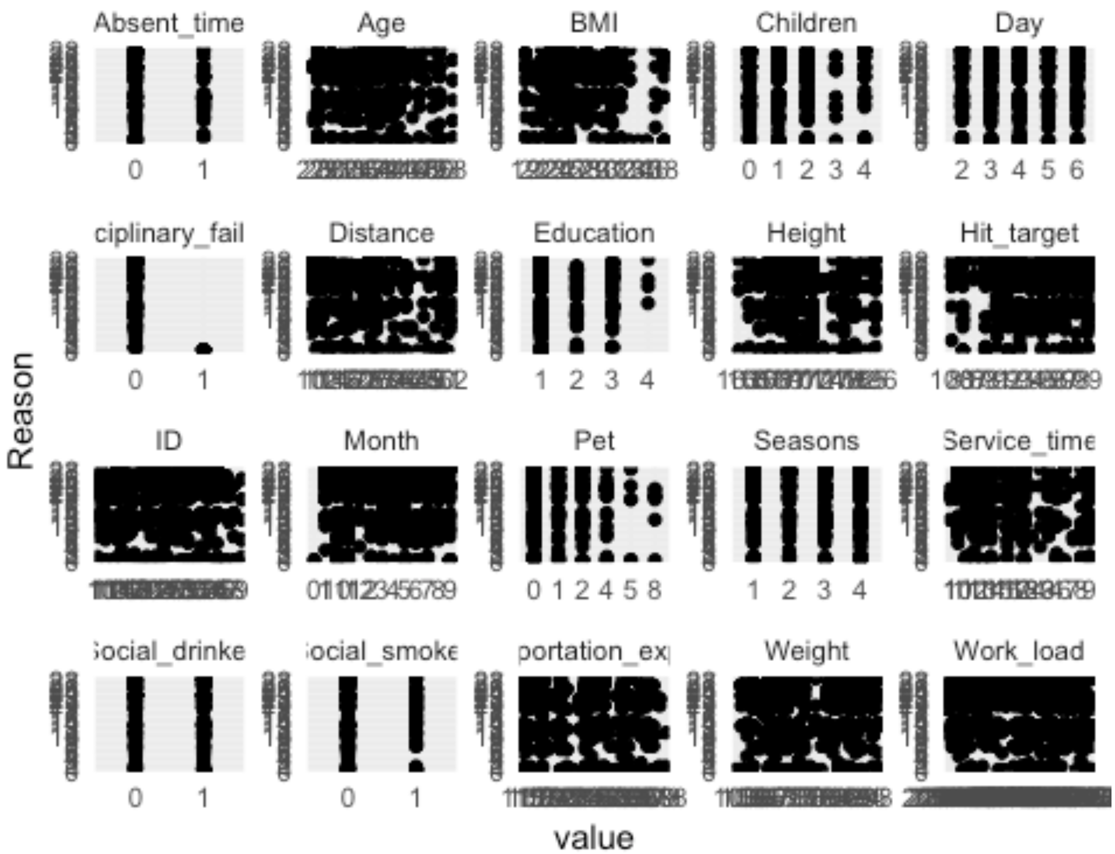
```
  ggplot(aes(x=Reason)) +
  geom_bar() +
  theme_minimal()
```




```
dat %>%
  filter(Absent_time==1) %>%
  ggplot(aes(x=Reason)) +
  geom_bar() +
  theme_minimal()
```




```r
#Reason for absence
table(dat %>%
  filter(Reason==0) %>%
  select(Absent_time))

##
##  0  1
## 43  0

dat %>%
  gather(-Reason, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Reason)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

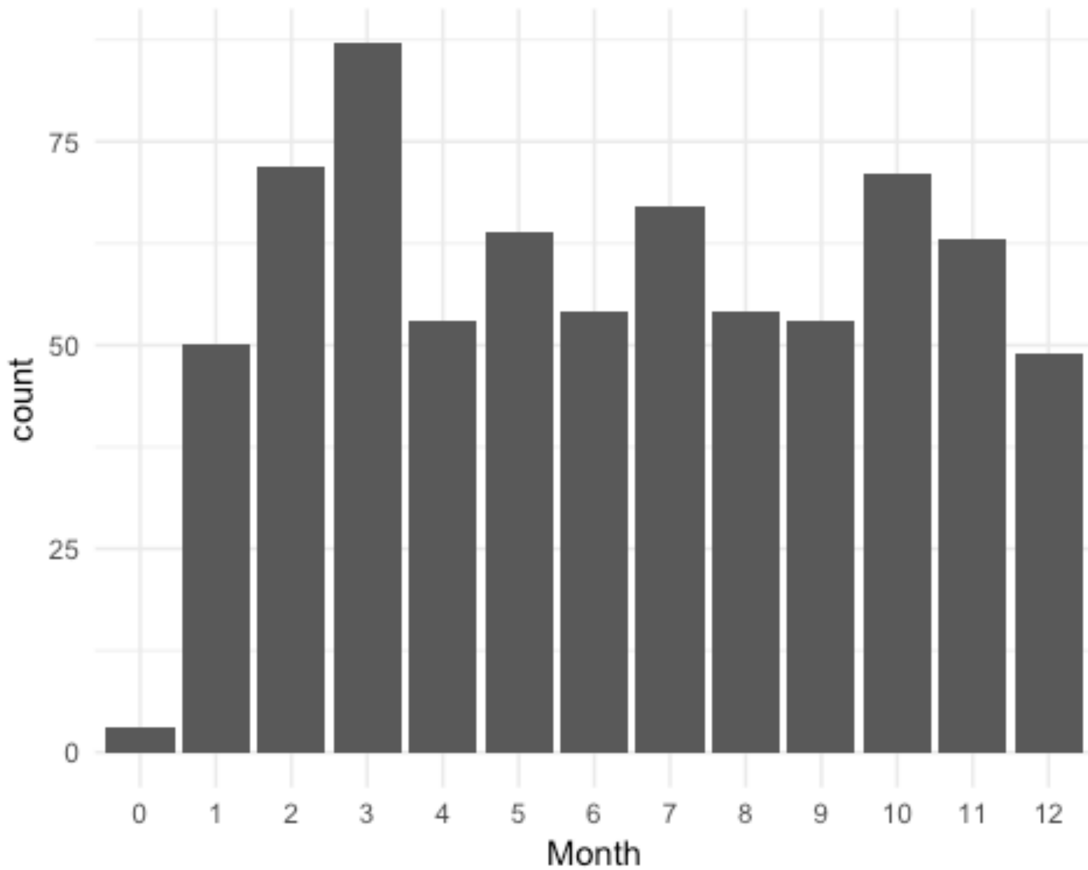


Month

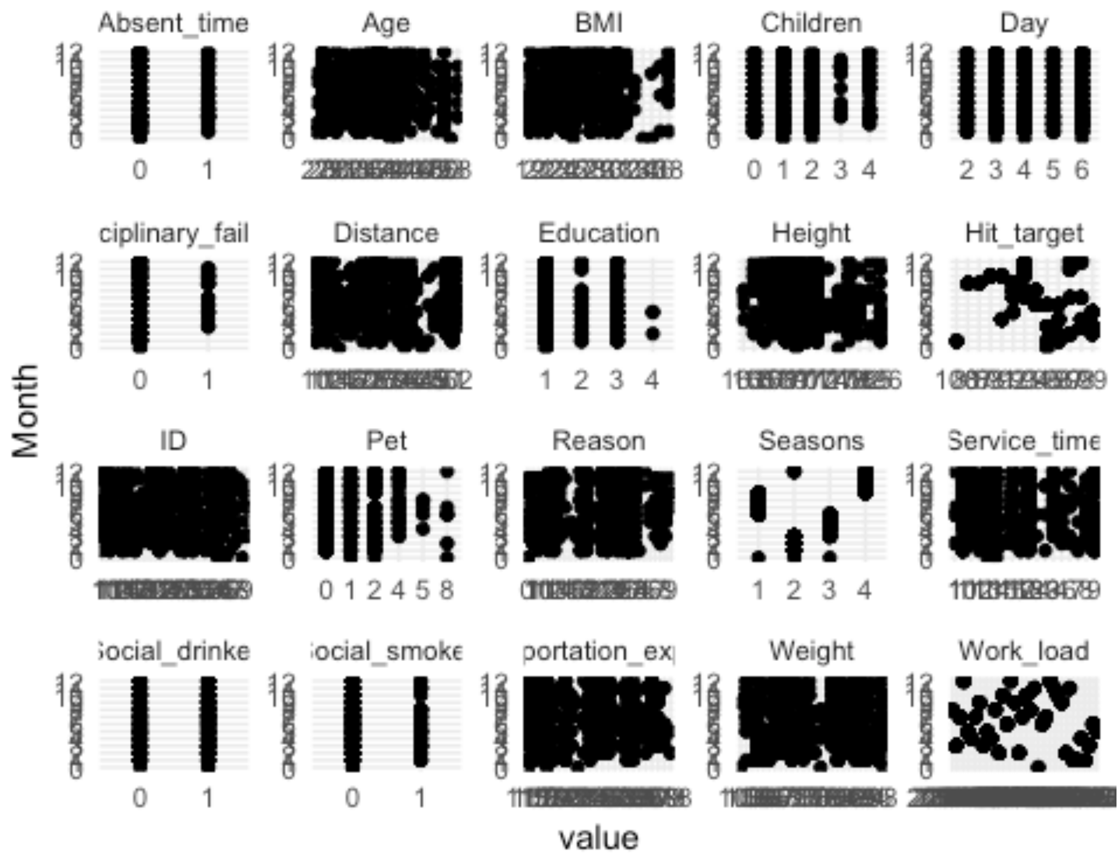
```
#frequency table by Month of Absence
dat %>%
  count(Month)
```

```
## # A tibble: 13 x 2
##    Month     n
##    <ord> <int>
##  1 0         3
##  2 1        50
##  3 2        72
##  4 3        87
##  5 4        53
##  6 5        64
##  7 6        54
##  8 7        67
##  9 8        54
## 10 9        53
## 11 10       71
## 12 11       63
## 13 12       49
```

```
#bar chart
dat %>%
  ggplot(aes(x=Month)) +
  geom_bar() +
  theme_minimal()
```

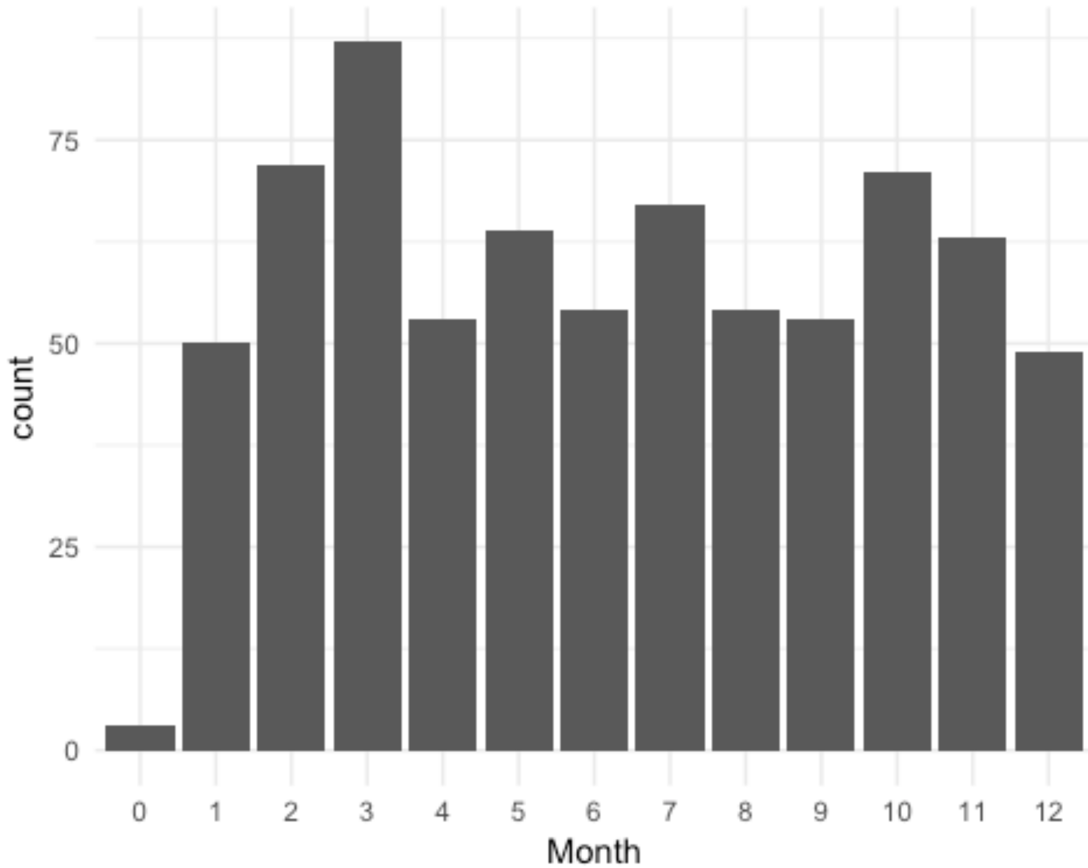



```
dat %>%
  gather(-Month, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Month)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

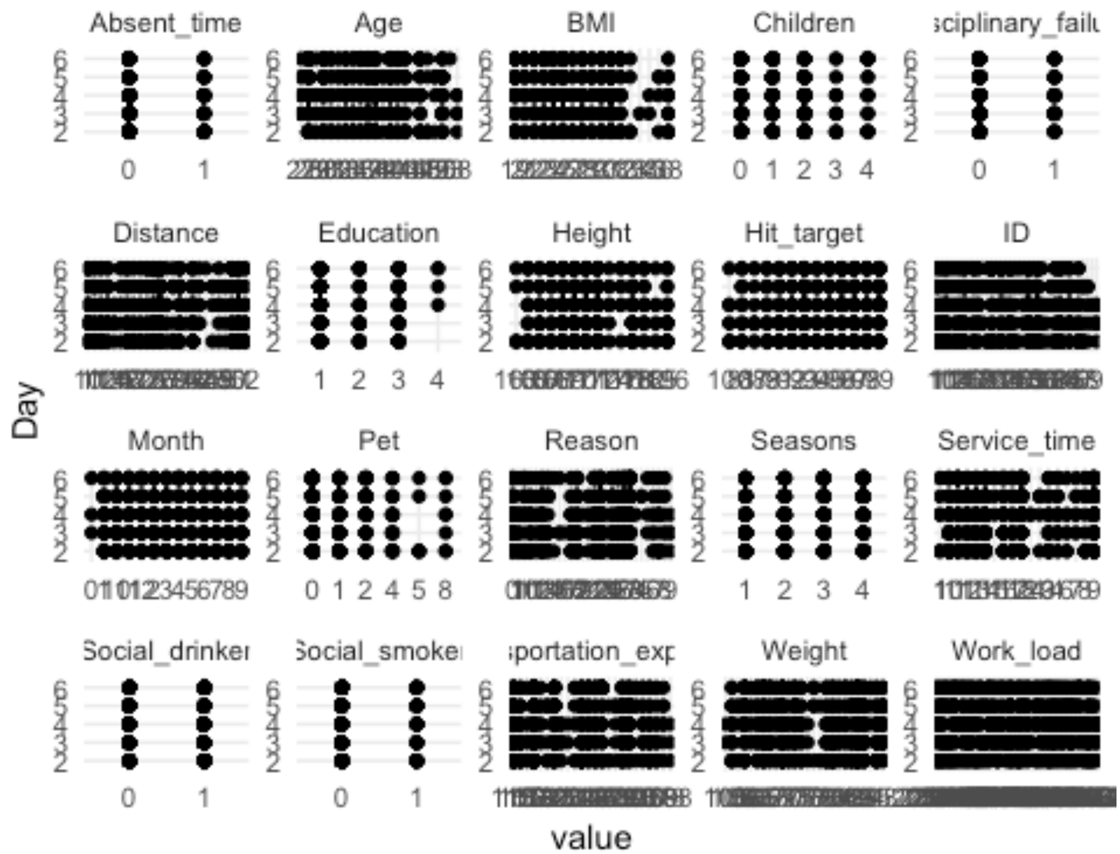


Day

```
#frequency table by Day of Absence
dat %>%
  count(Day)

## # A tibble: 5 x 2
##   Day       n
##   <ord> <int>
## 1 2       161
## 2 3       154
## 3 4       156
## 4 5       125
## 5 6       144

#bar chart
dat %>%
  ggplot(aes(x=Month)) +
  geom_bar() +
  theme_minimal()
```

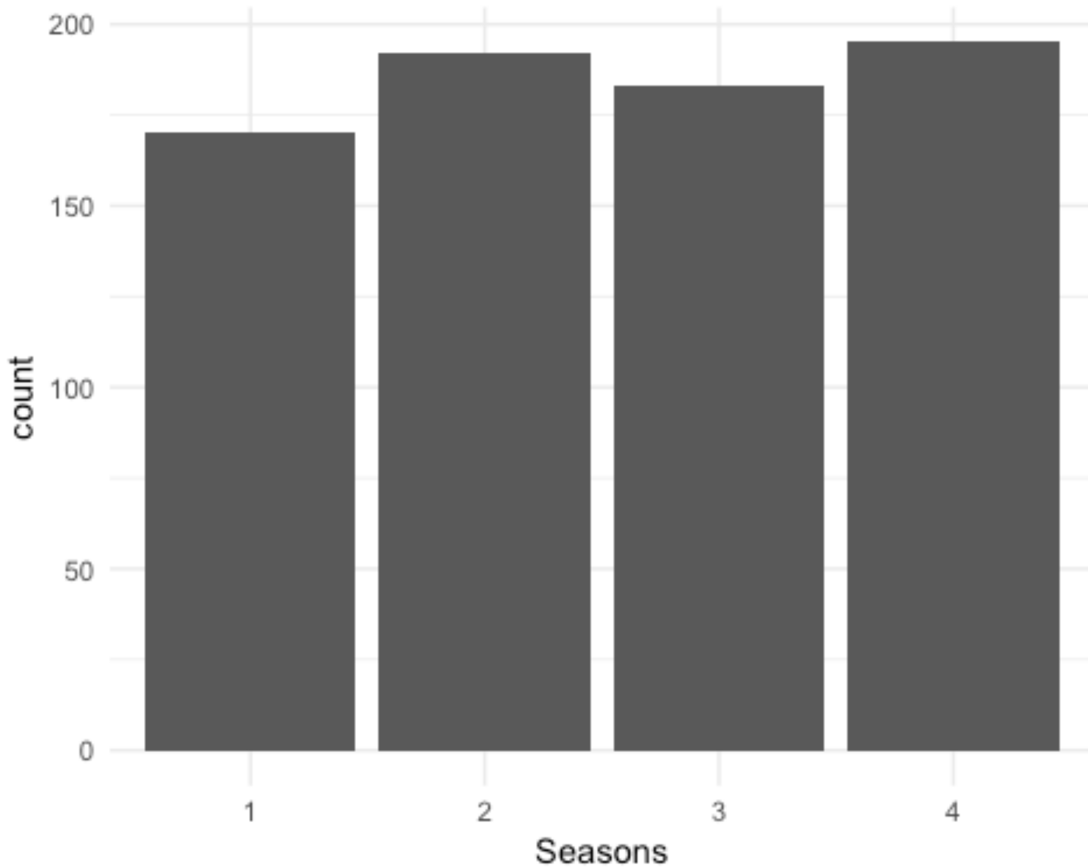



```r
dat %>%
  gather(-Day, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Day)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

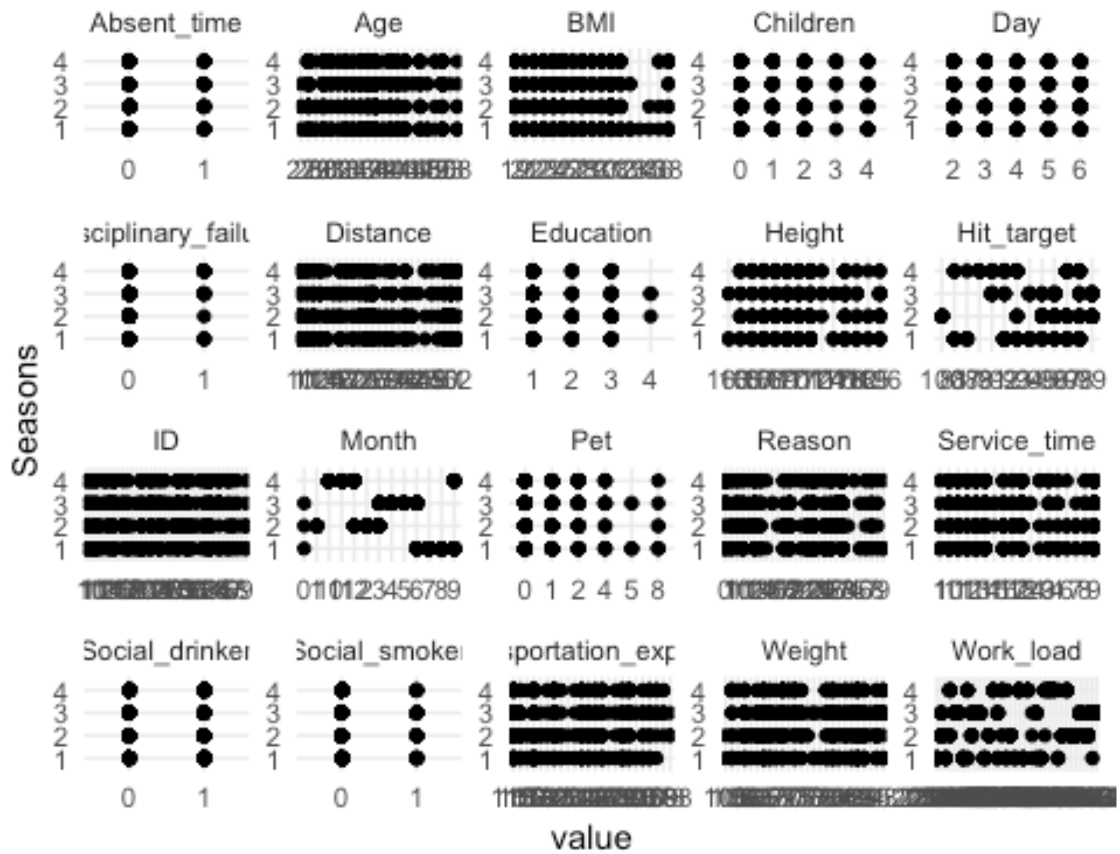


Seasons

```
#frequency table by Season of Absence
dat %>%
  count(Seasons)

## # A tibble: 4 x 2
##   Seasons     n
##   <ord>   <int>
## 1 1         170
## 2 2         192
## 3 3         183
## 4 4         195

#bar chart
dat %>%
  ggplot(aes(x=Seasons)) +
  geom_bar() +
  theme_minimal()
```

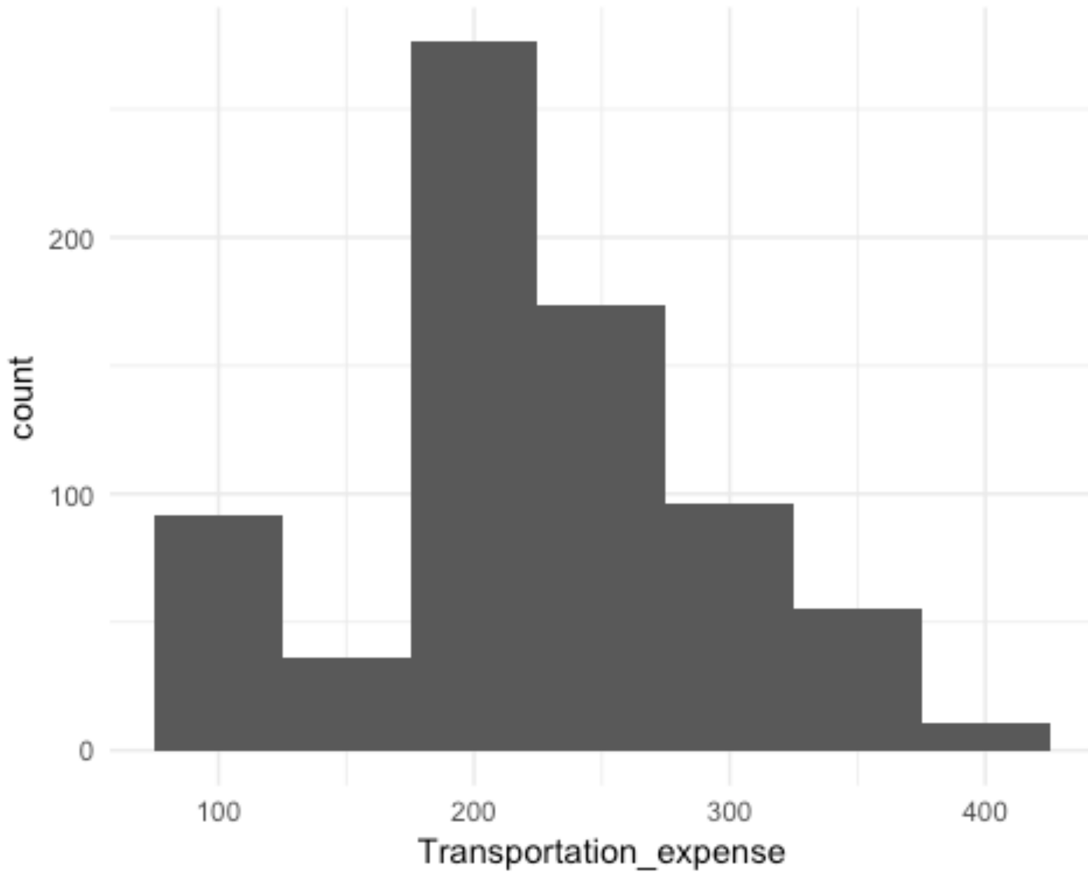



```
#Scatterplots for variable 'Seasons'
dat %>%
  gather(-Seasons, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Seasons)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

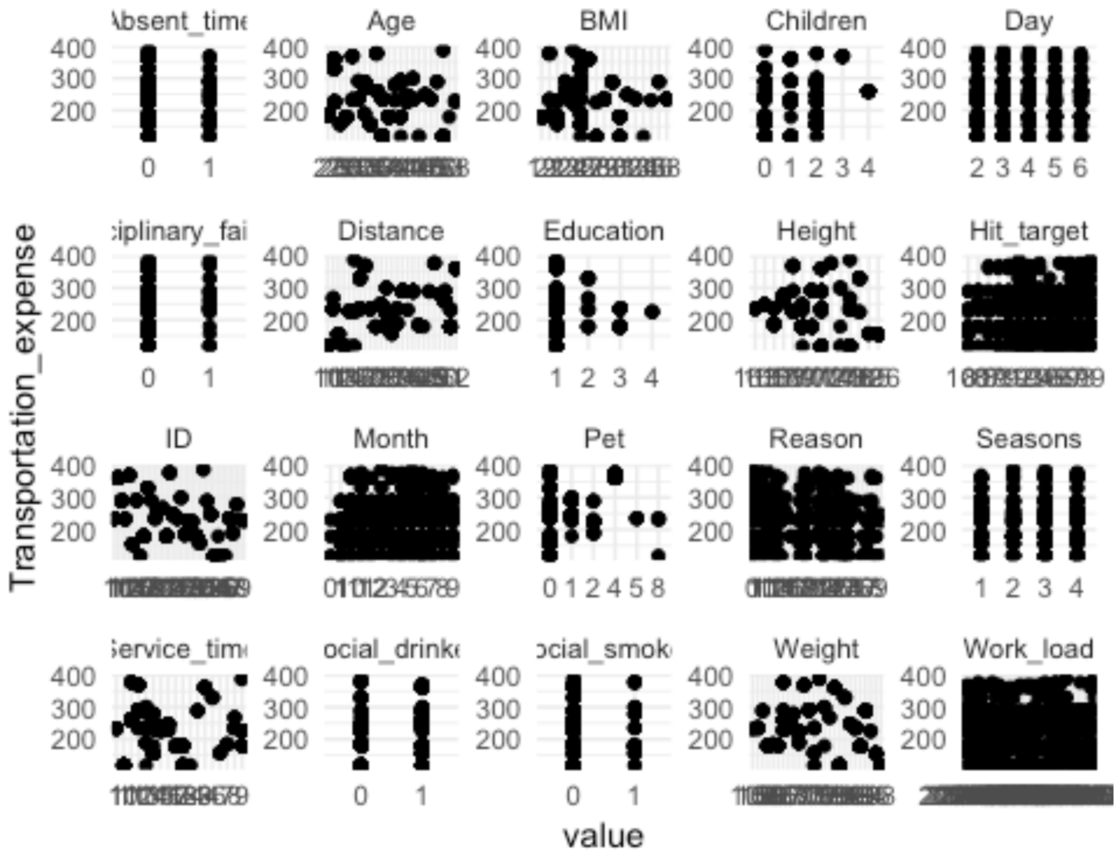


Transportation Expense

```
#summary of transportation expenses
summary(dat$Transportation_expense)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   118.0   179.0   225.0   221.3   260.0   388.0

#histograph
ggplot(data = dat,
       aes(x = Transportation_expense)) +
  geom_histogram(binwidth = 50) +
  theme_minimal()
```



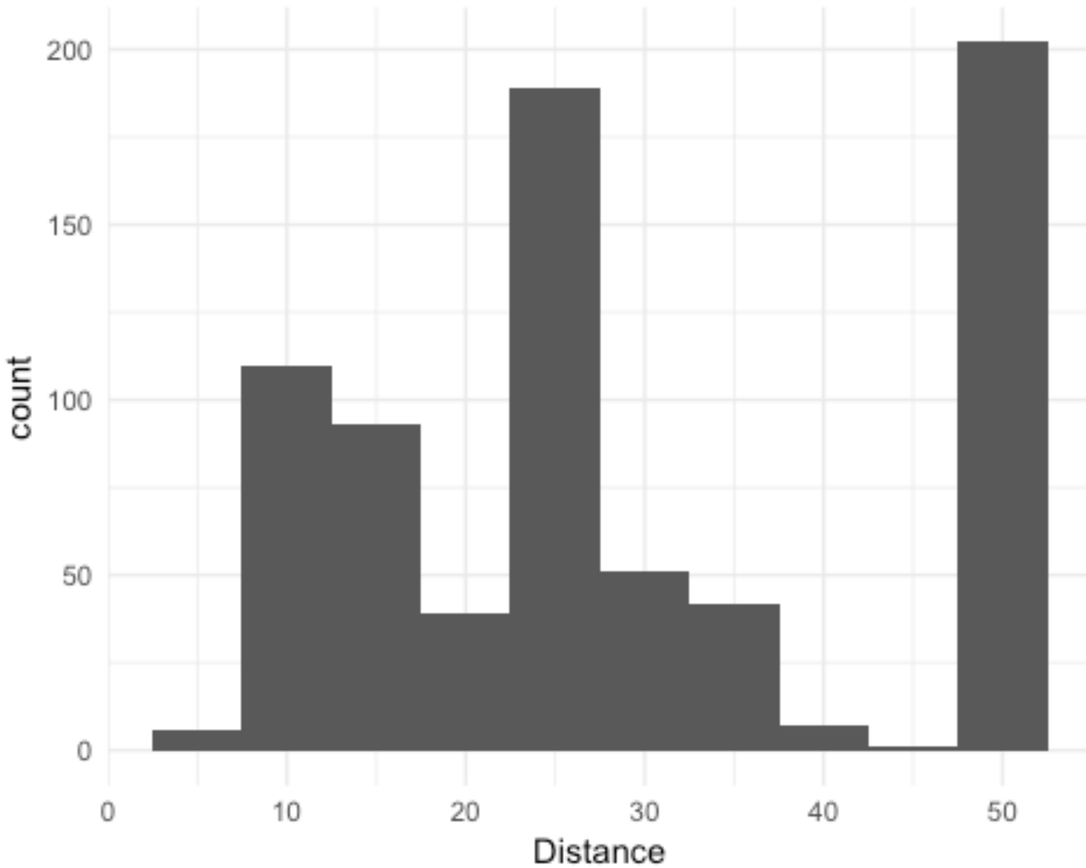

```
#Scatterplots for variable 'Transportation_expense'
dat %>%
  gather(-Transportation_expense, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Transportation_expense)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



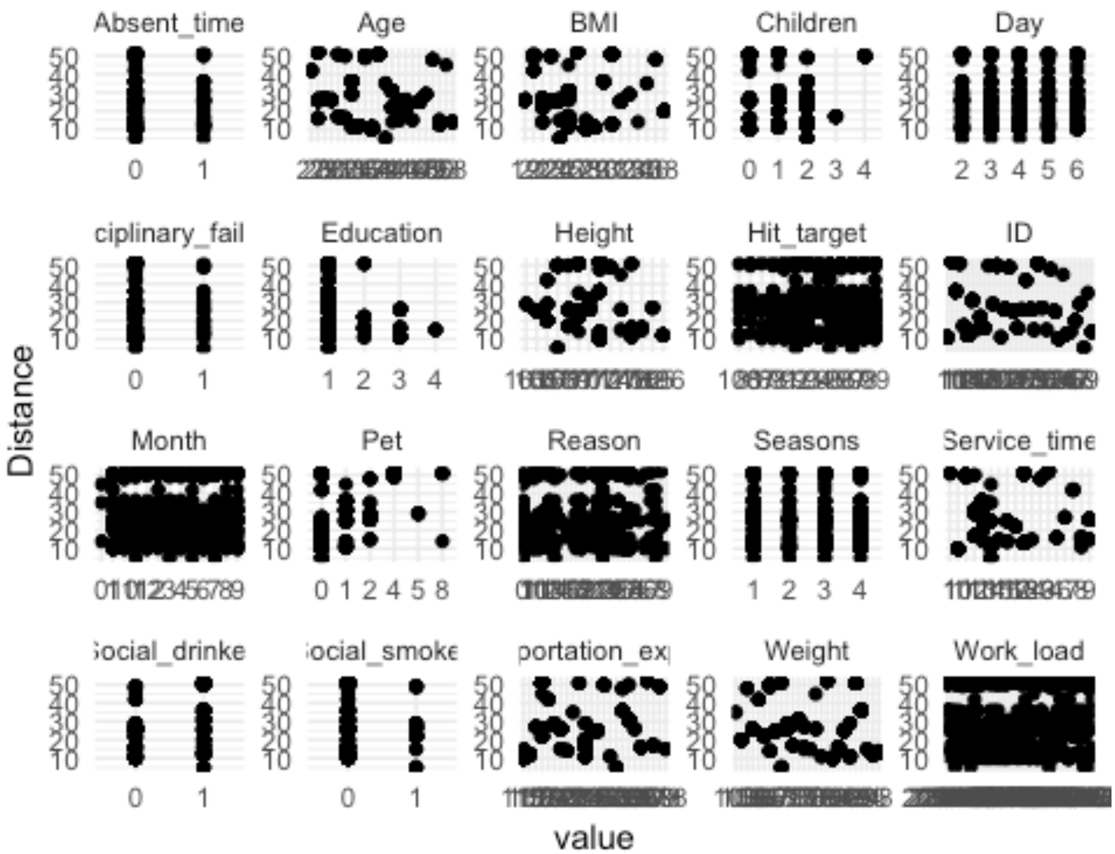

```
# Possible positive correlation seen between distance and
Transportation_expense
```

Distance

```
#summary of distance
summary(dat$Distance)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   16.00   26.00   29.63   50.00   52.00

#histogram
ggplot(data = dat,
       aes(x = Distance)) +
  geom_histogram(binwidth = 5) +
  theme_minimal()
```




```
#Scatterplots for variable 'Distance'
dat %>%
  gather(-Distance, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Distance)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



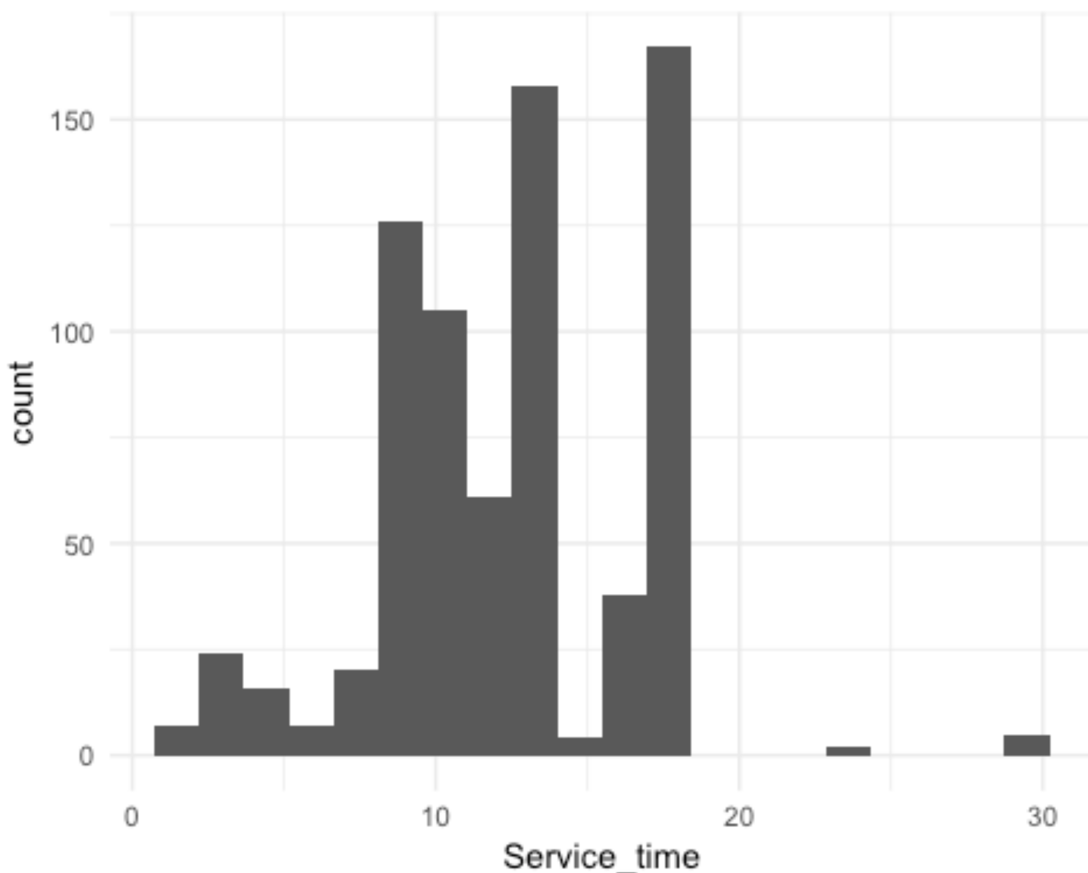

```
#Possible Positive correlation seen between distance and
Transportation_expense
```

Service Time

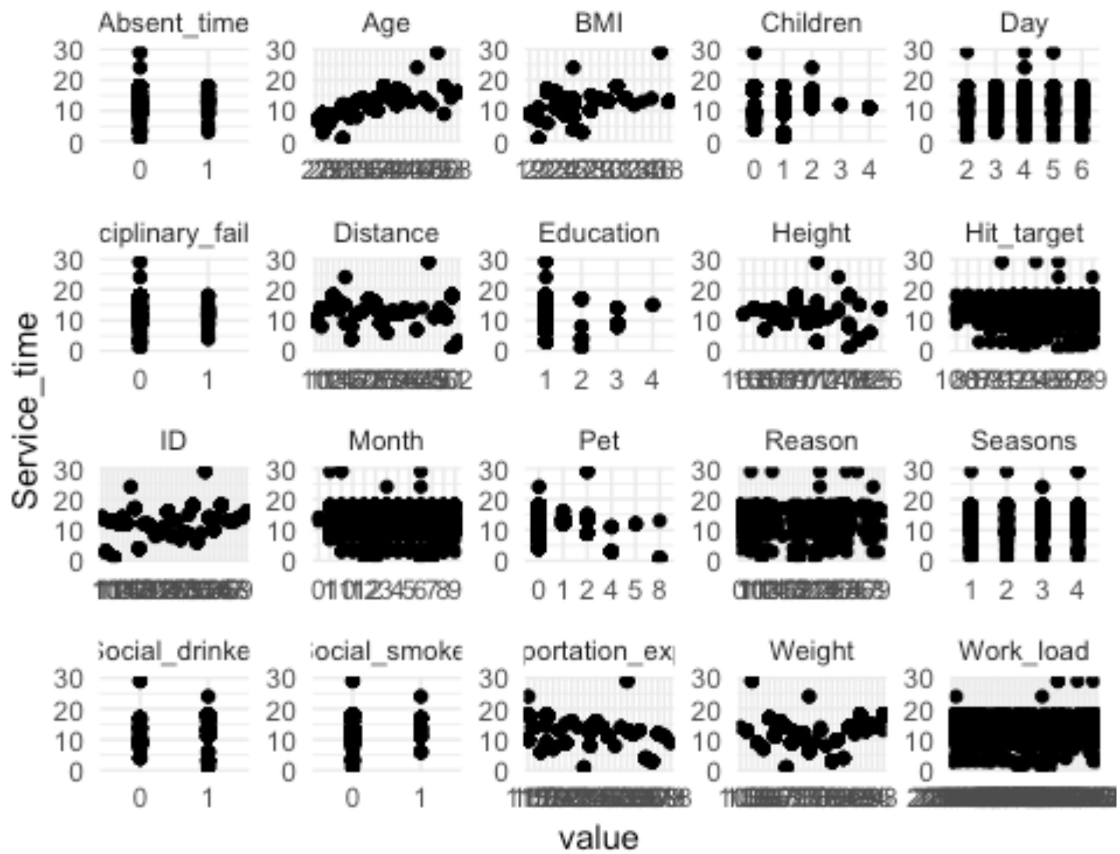
```
#summary for Service_time
summary(dat$Service_time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    9.00   13.00   12.55   16.00   29.00
```

```
#histogram
ggplot(data = dat,
       aes(x = Service_time)) +
  geom_histogram(bins = 20) +
  theme_minimal()
```

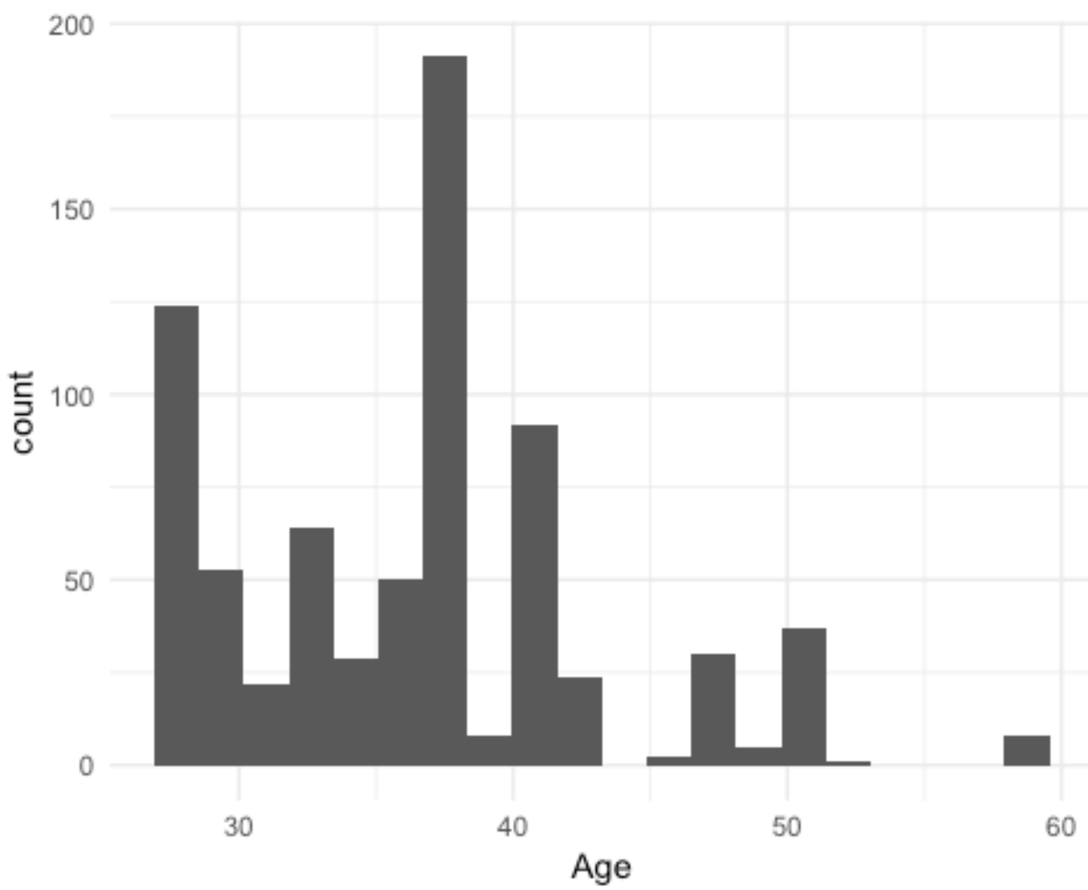



```
#Scatterplots for variable 'Service_time'
dat %>%
  gather(-Service_time, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Service_time)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

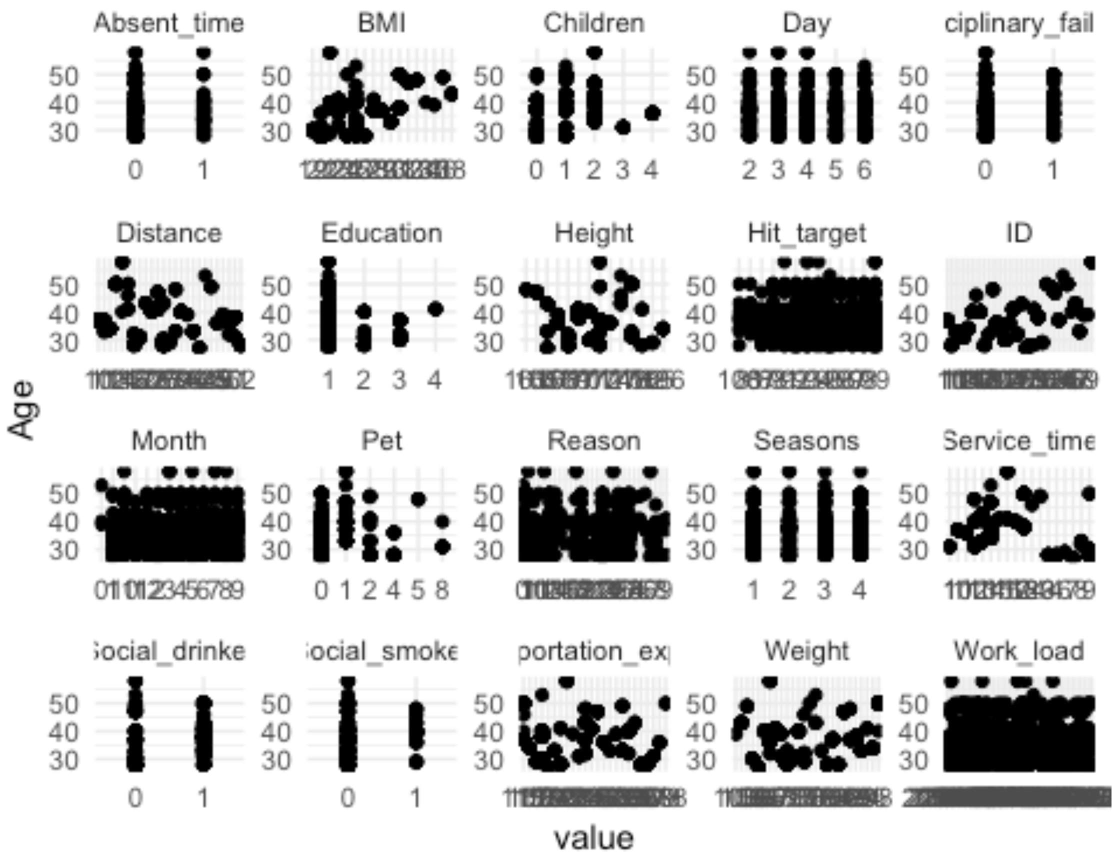


Age

```r
#summary for Age
summary(dat$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.00   31.00   37.00   36.45   40.00   58.00

#histogram
ggplot(data = dat,
       aes(x = Age)) +
  geom_histogram(bins = 20) +
  theme_minimal()
```

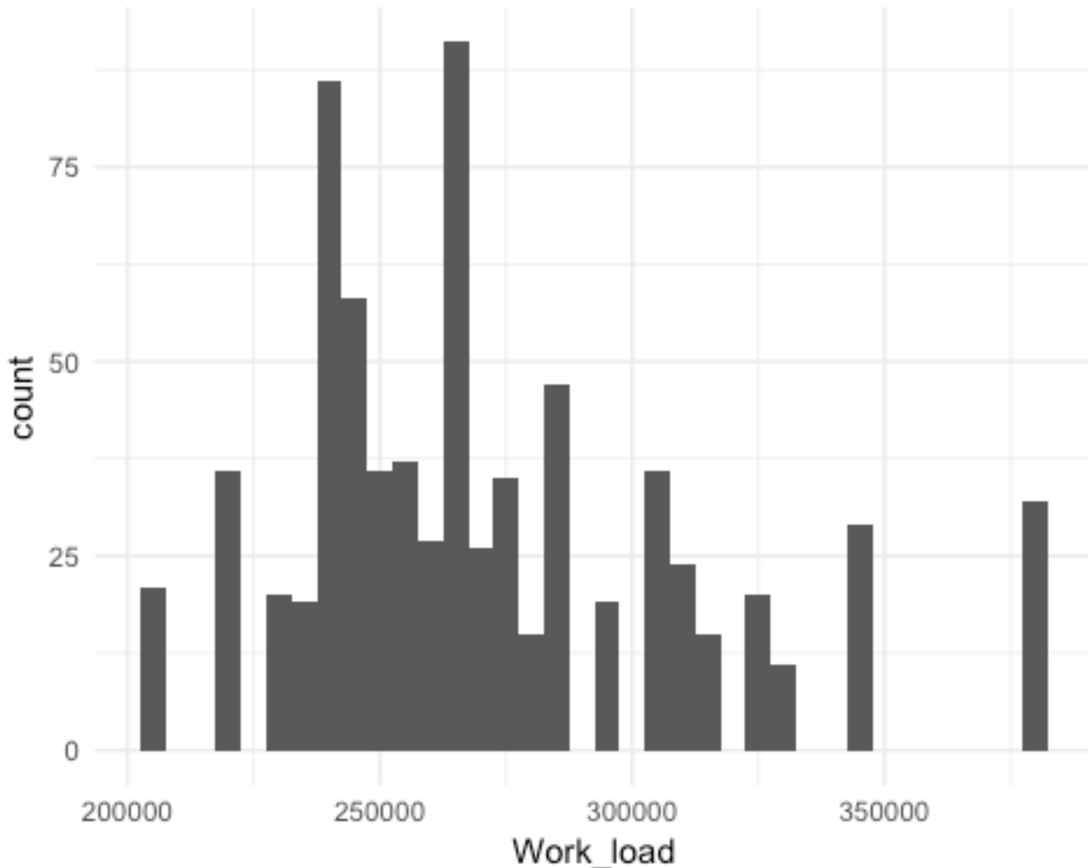



```
#Scatterplots for variable 'Age'
dat %>%
  gather(-Age, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Age)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

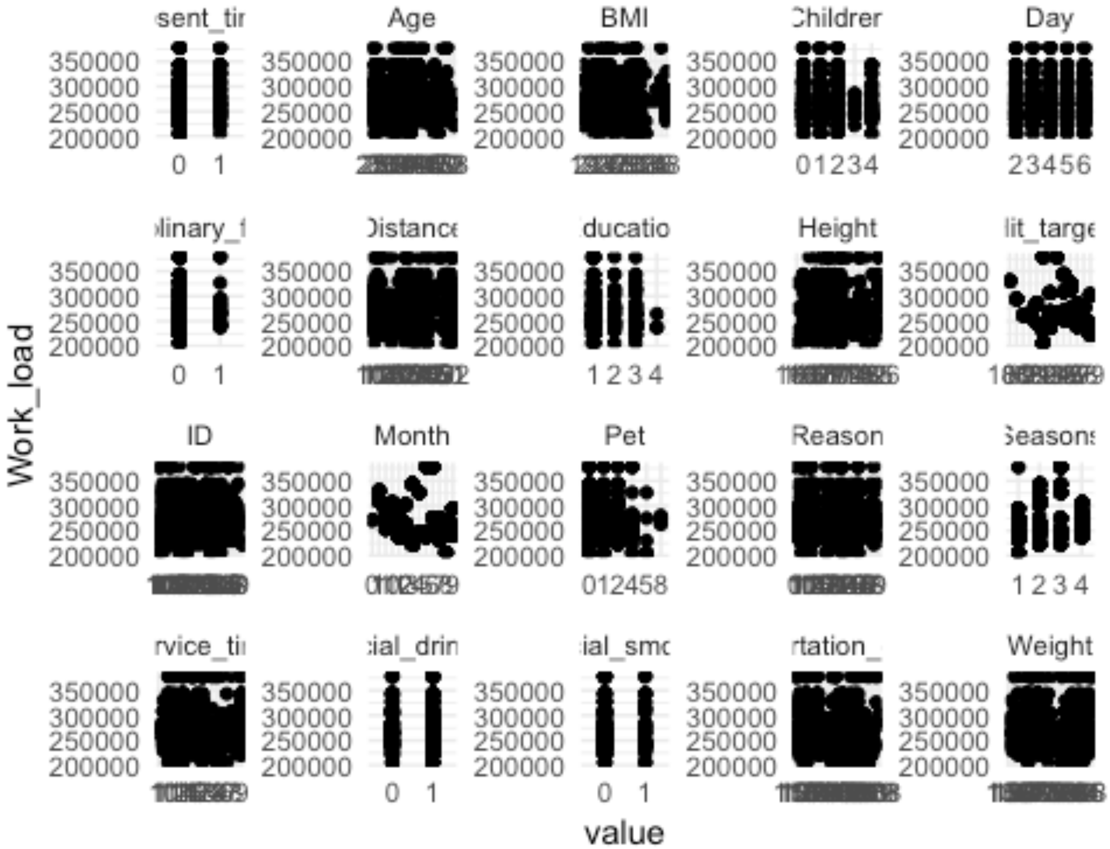


Workload

```r
#summary for work load
summary(dat$Work_load)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  205917  244387  264249  271490  294217  378884

#histogram
ggplot(data = dat,
       aes(x = Work_load)) +
  geom_histogram(binwidth = 5000) +
  theme_minimal()
```

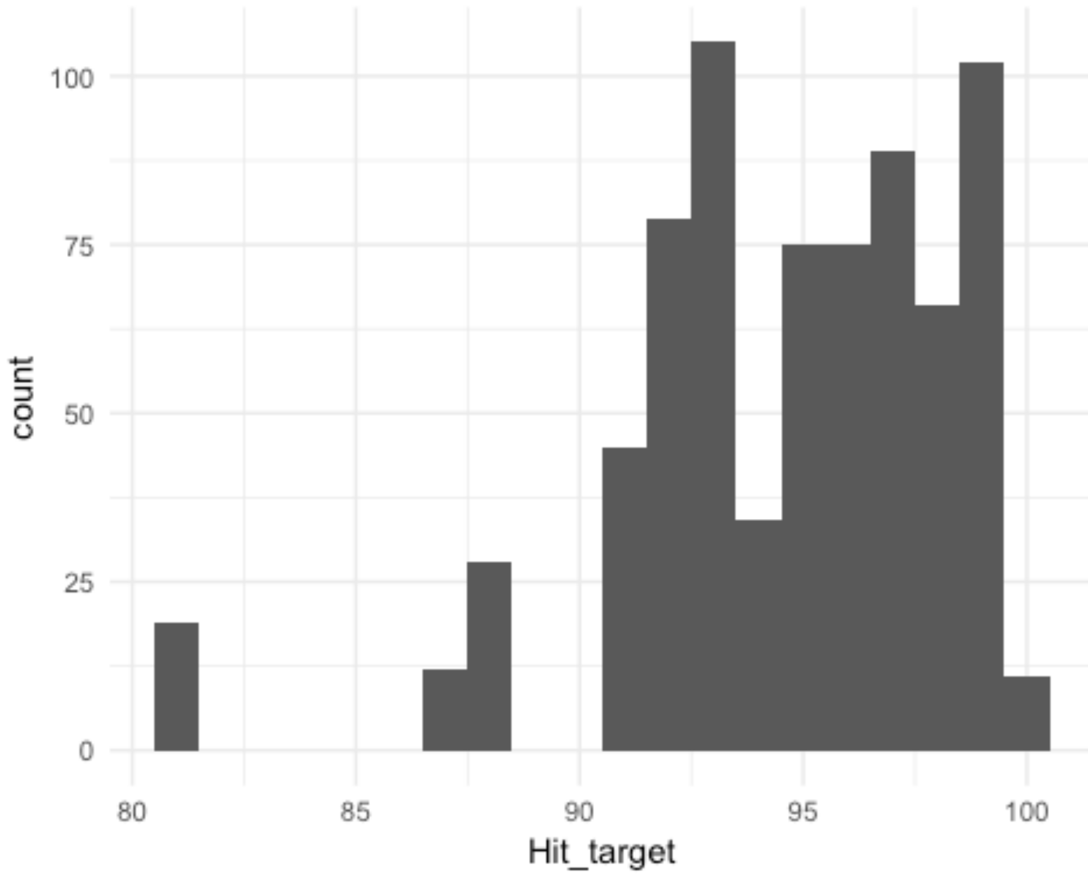



```
#Scatterplots for variable 'Work_load'
dat %>%
  gather(-Work_load, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Work_load)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

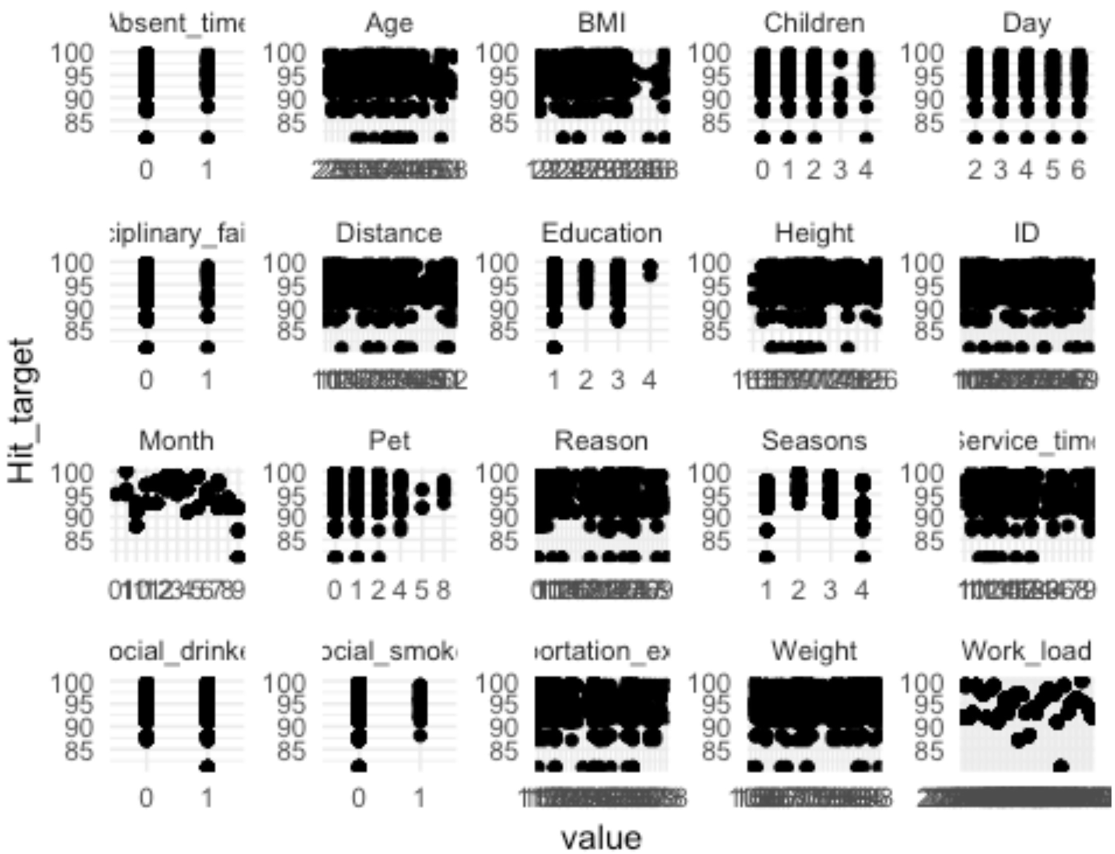


Hit Target

```r
#summary for hit target
summary(dat$Hit_target)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81.00   93.00   95.00   94.59   97.00  100.00

#histogram
ggplot(data = dat,
       aes(x = Hit_target)) +
  geom_histogram(bins = 20) +
  theme_minimal()
```

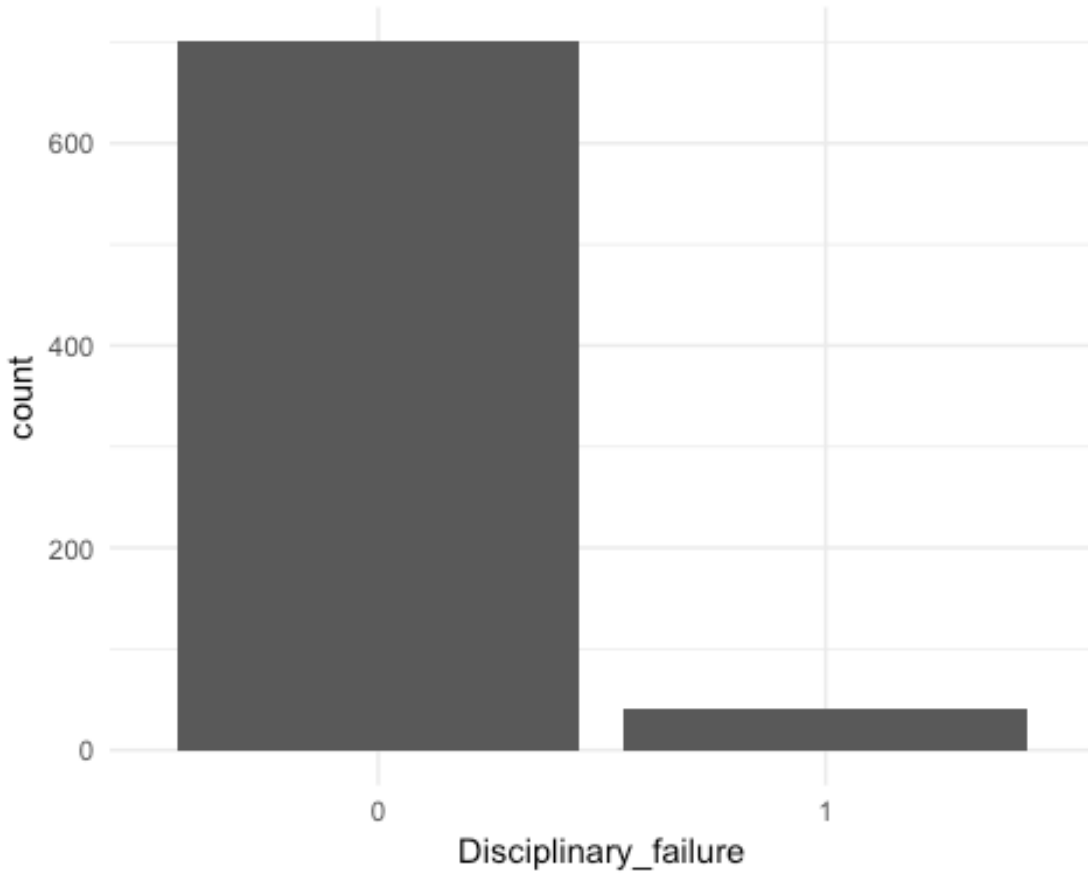



```
#Scatterplots for variable 'Hit_target'
dat %>%
  gather(-Hit_target, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Hit_target)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

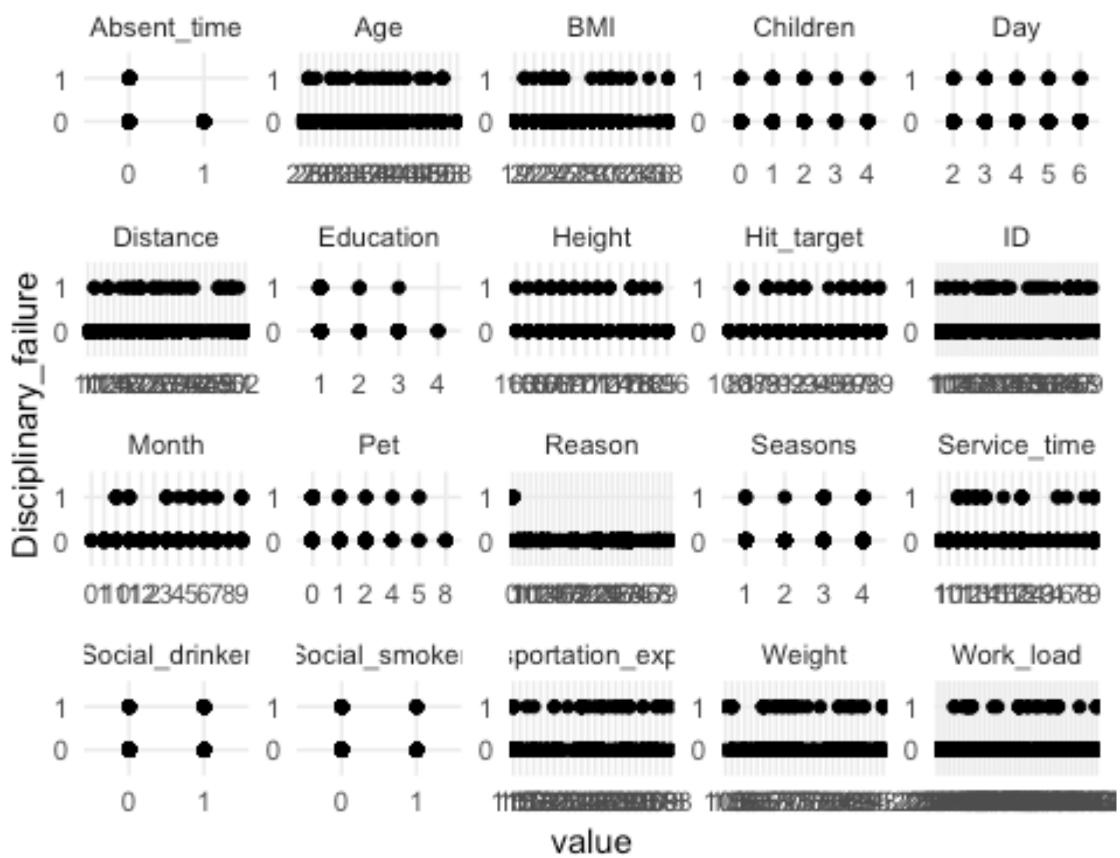


Disciplinary Failure

```
#table for disciplinary failure
dat %>%
  count(Disciplinary_failure)

## # A tibble: 2 x 2
##   Disciplinary_failure     n
##   <ord>                <int>
## 1 0                      700
## 2 1                       40

#bar chart
ggplot(data = dat,
       aes(x = Disciplinary_failure)) +
  geom_bar() +
  theme_minimal()
```

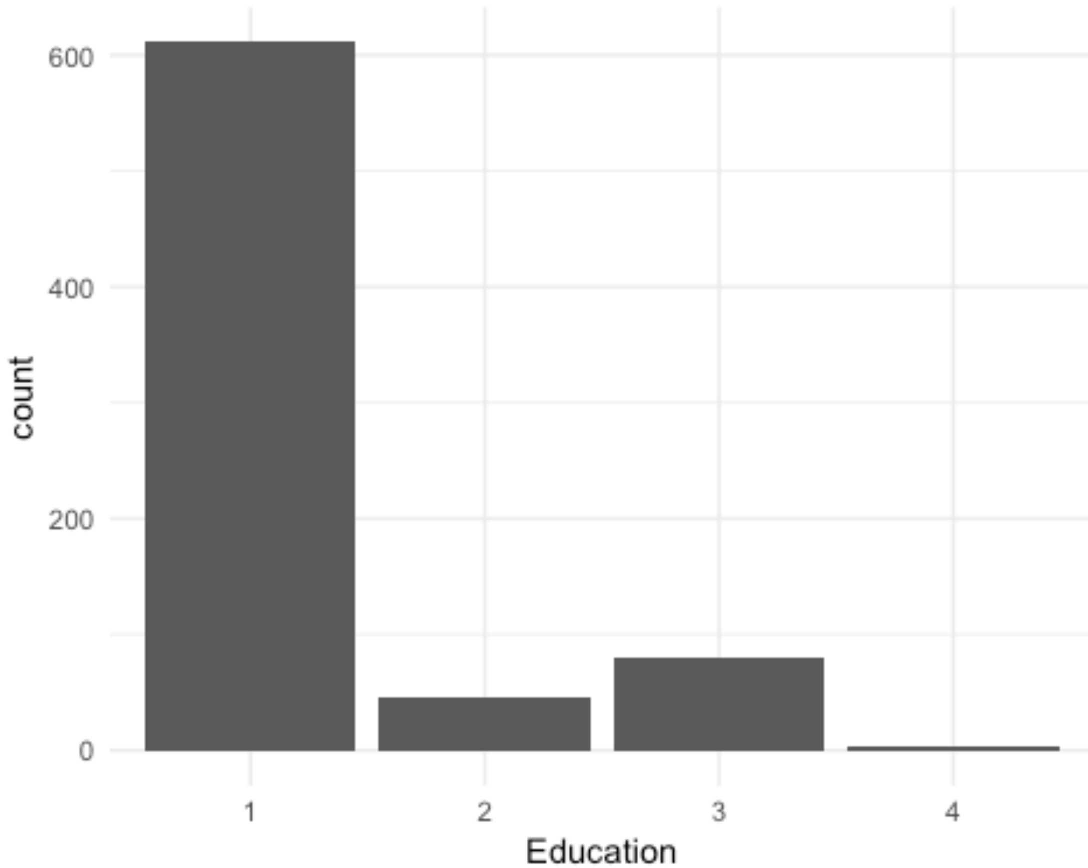



```r
#Scatterplots for variable 'Disciplinary_failure'
dat %>%
  gather(-Disciplinary_failure, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Disciplinary_failure)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

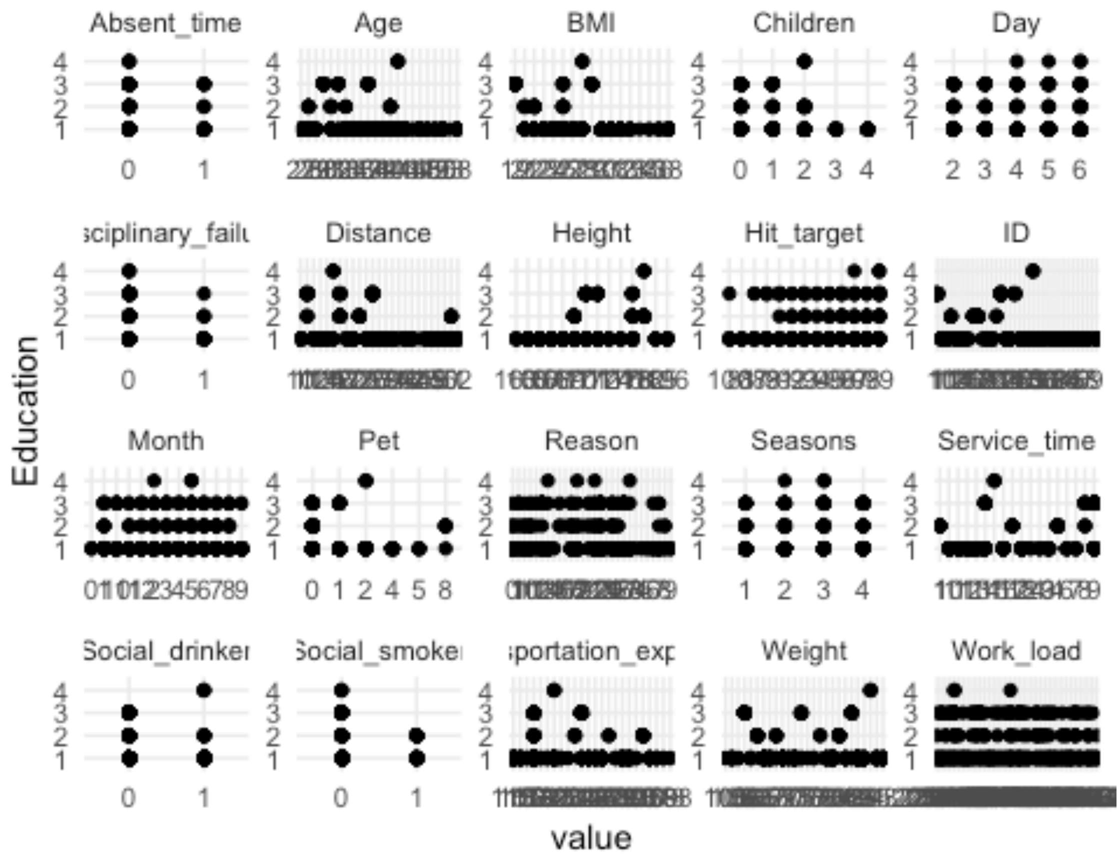


Education

```
#table for education
dat %>%
  count(Education)

## # A tibble: 4 x 2
##   Education     n
##   <ord>     <int>
## 1 1           611
## 2 2            46
## 3 3            79
## 4 4             4

#bar chart
ggplot(data = dat,
       aes(x = Education)) +
  geom_bar() +
  theme_minimal()
```

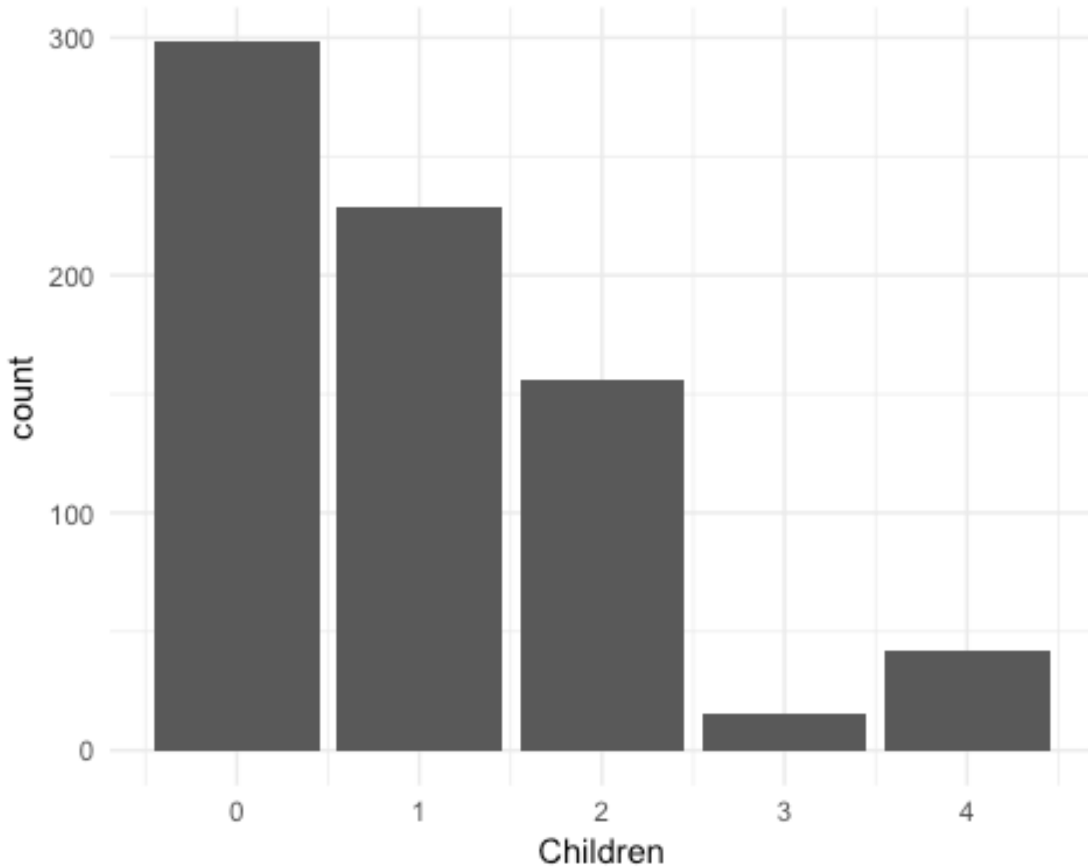



```
#Scatterplots for variable 'Education'
dat %>%
  gather(-Education, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Education)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

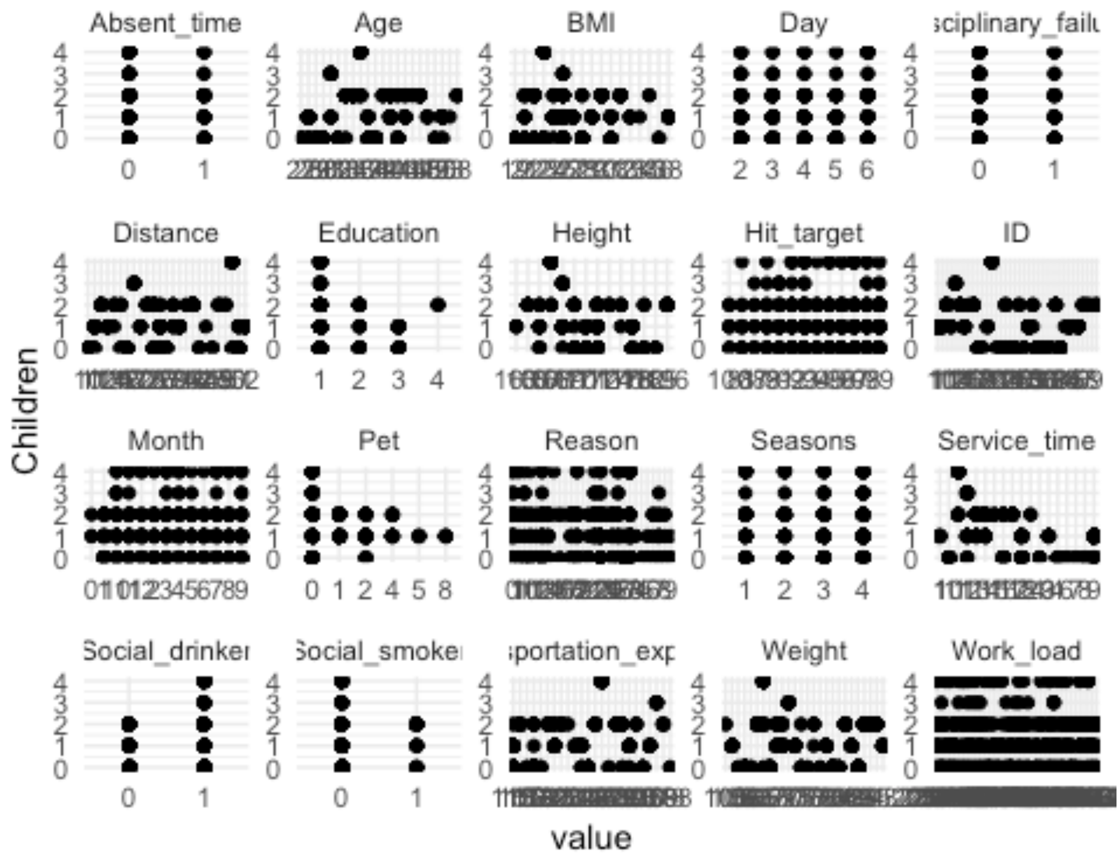


Children

```
#table for number of children
dat %>%
  count(Children)

## # A tibble: 5 x 2
##   Children     n
##      <dbl> <int>
## 1        0   298
## 2        1   229
## 3        2   156
## 4        3    15
## 5        4    42

#bar chart
ggplot(data = dat,
       aes(x = Children)) +
  geom_bar() +
  theme_minimal()
```

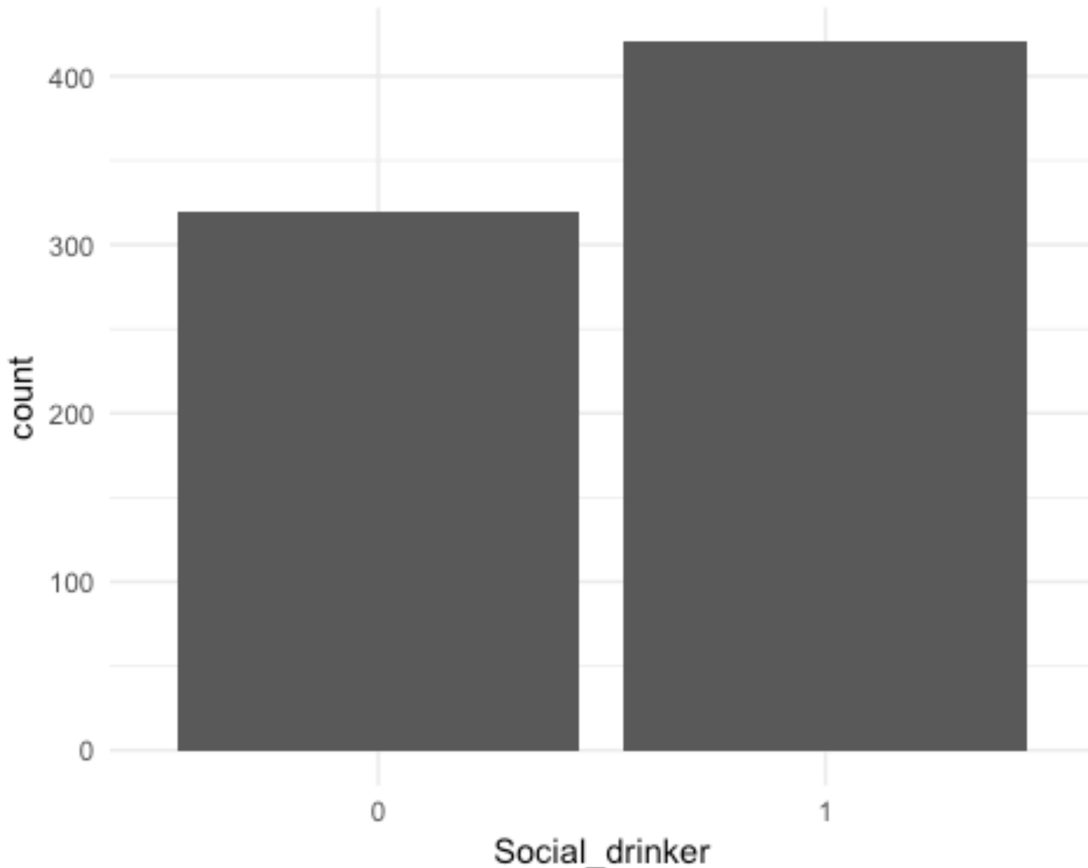



```
#Scatterplots for variable 'Children'
dat %>%
  gather(-Children, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Children)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

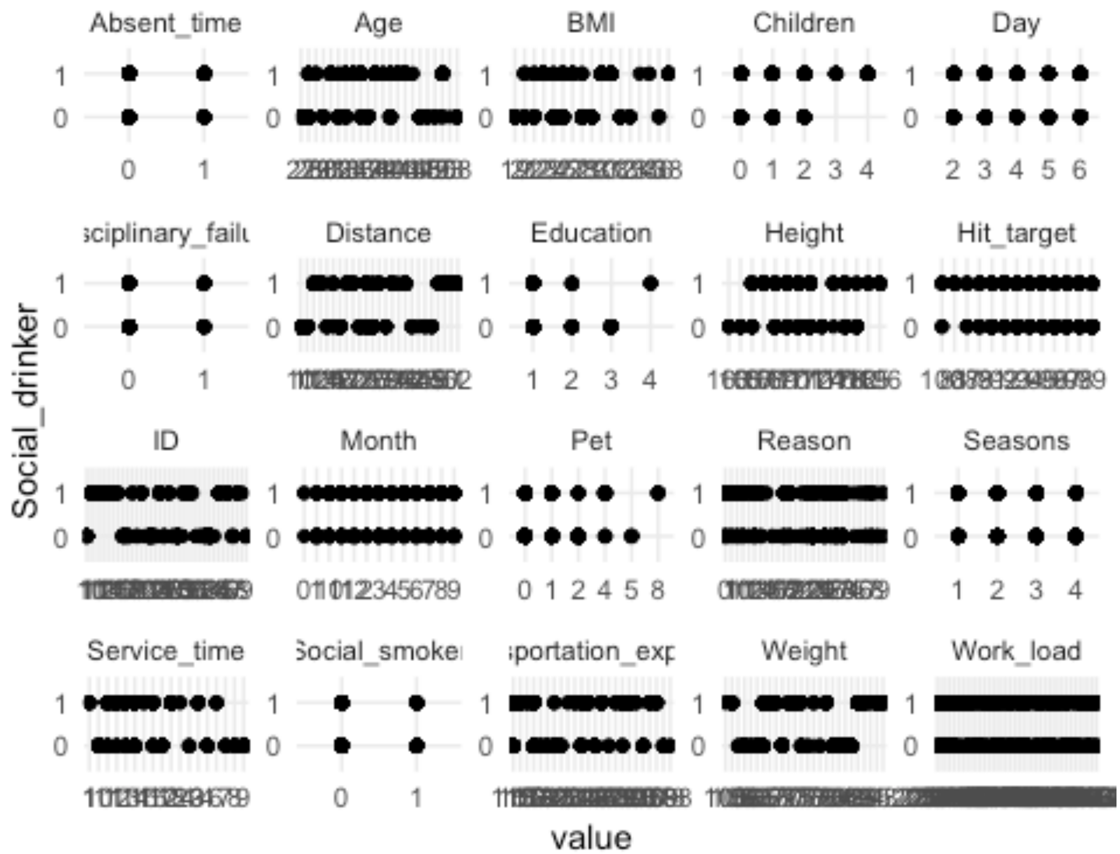


Social Drinker

```
#table for social drinking
dat %>%
  count(Social_drinker)

## # A tibble: 2 x 2
##   Social_drinker     n
##   <ord>          <int>
## 1 0                320
## 2 1                420

#bar chart
ggplot(data = dat,
       aes(x = Social_drinker)) +
  geom_bar() +
  theme_minimal()
```

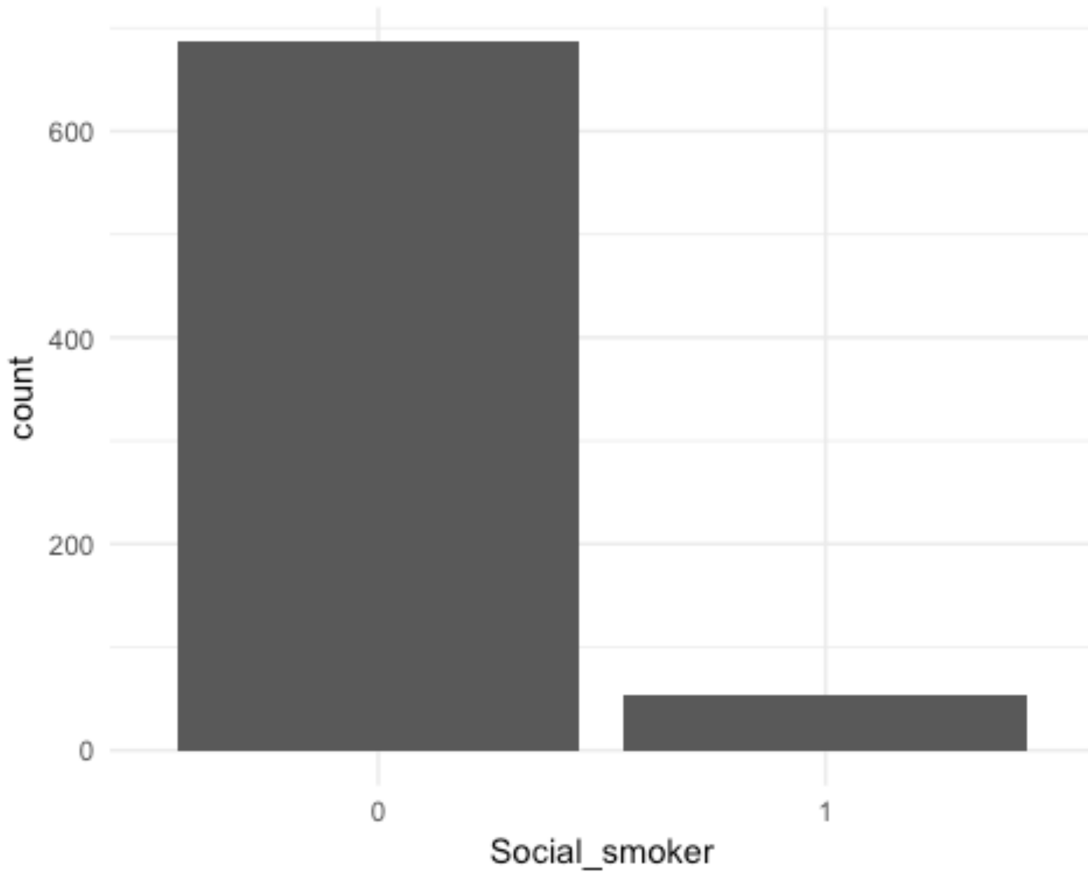



```
#Scatterplots for variable 'Social_drinker'
dat %>%
  gather(-Social_drinker, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Social_drinker)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

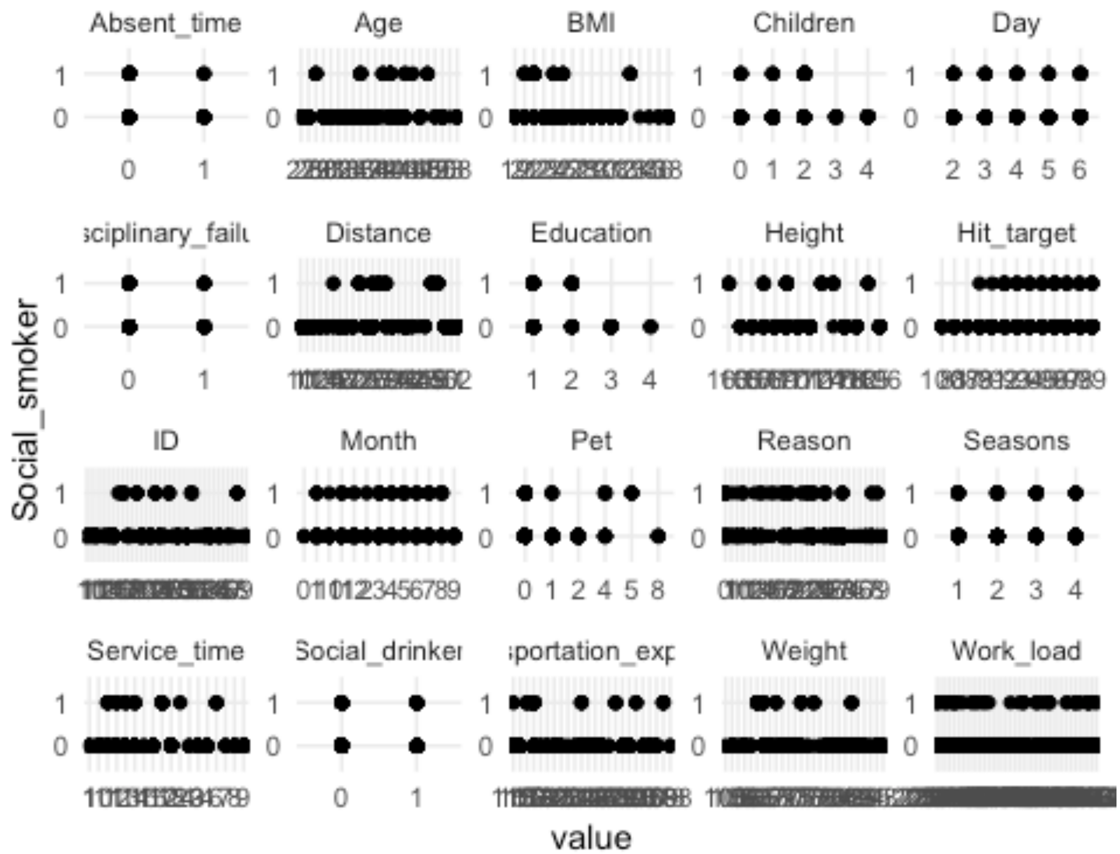


Social Smoker

```
#table for social smokers
dat %>%
  count(Social_smoker)

## # A tibble: 2 x 2
##   Social_smoker     n
##   <ord>         <int>
## 1 0               686
## 2 1                54

#bar chart
ggplot(data = dat,
       aes(x = Social_smoker)) +
  geom_bar() +
  theme_minimal()
```

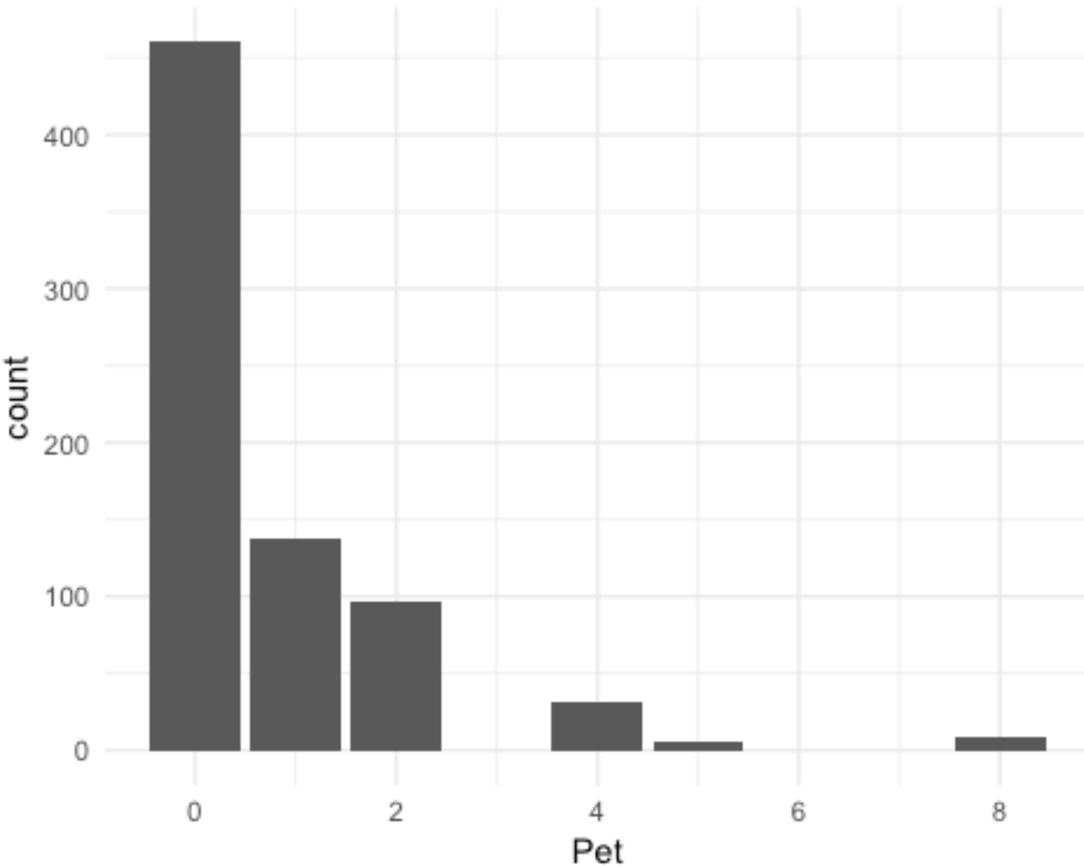



```r
#Scatterplots for variable 'Social_smoker'
dat %>%
  gather(-Social_smoker, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Social_smoker)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

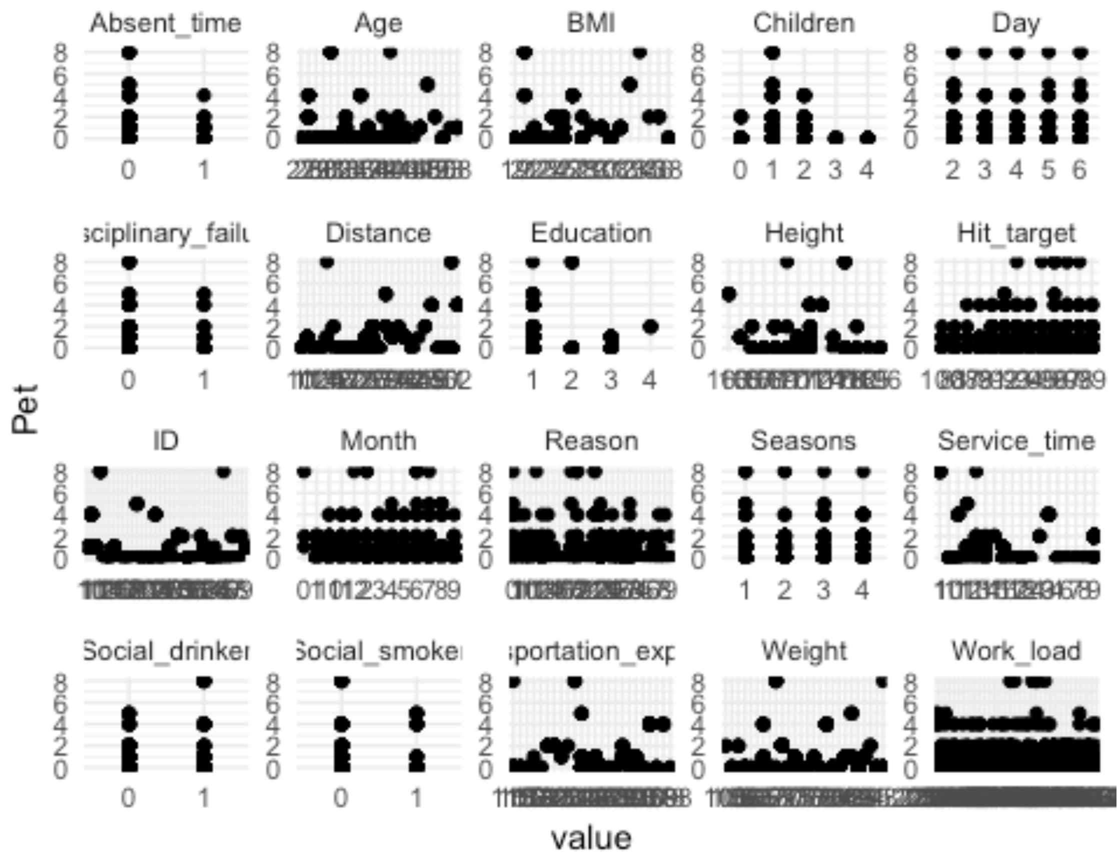


Pet

```r
#summary for pets
summary(dat$Pet)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.7459  1.0000  8.0000

#histogram
ggplot(data = dat,
       aes(x = Pet)) +
  geom_bar() +
  theme_minimal()
```

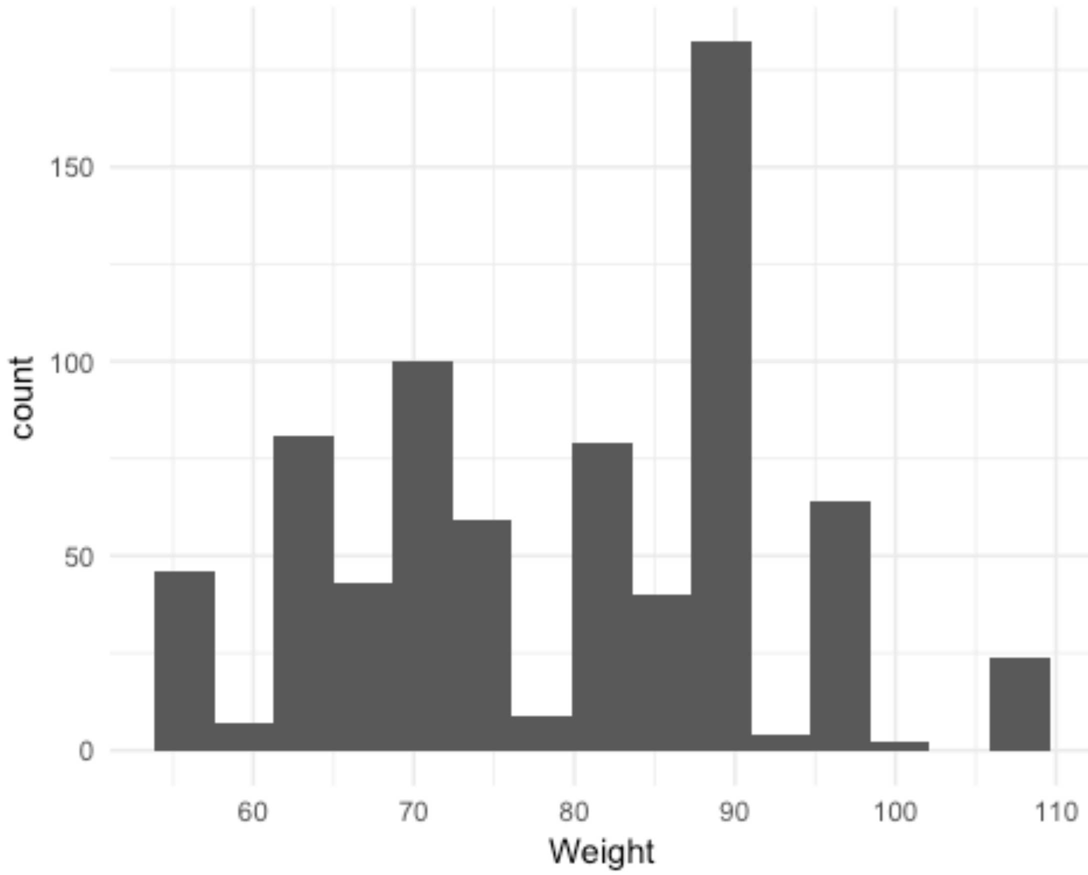



```r
#Scatterplots for variable 'Pet'
dat %>%
  gather(-Pet, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Pet)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

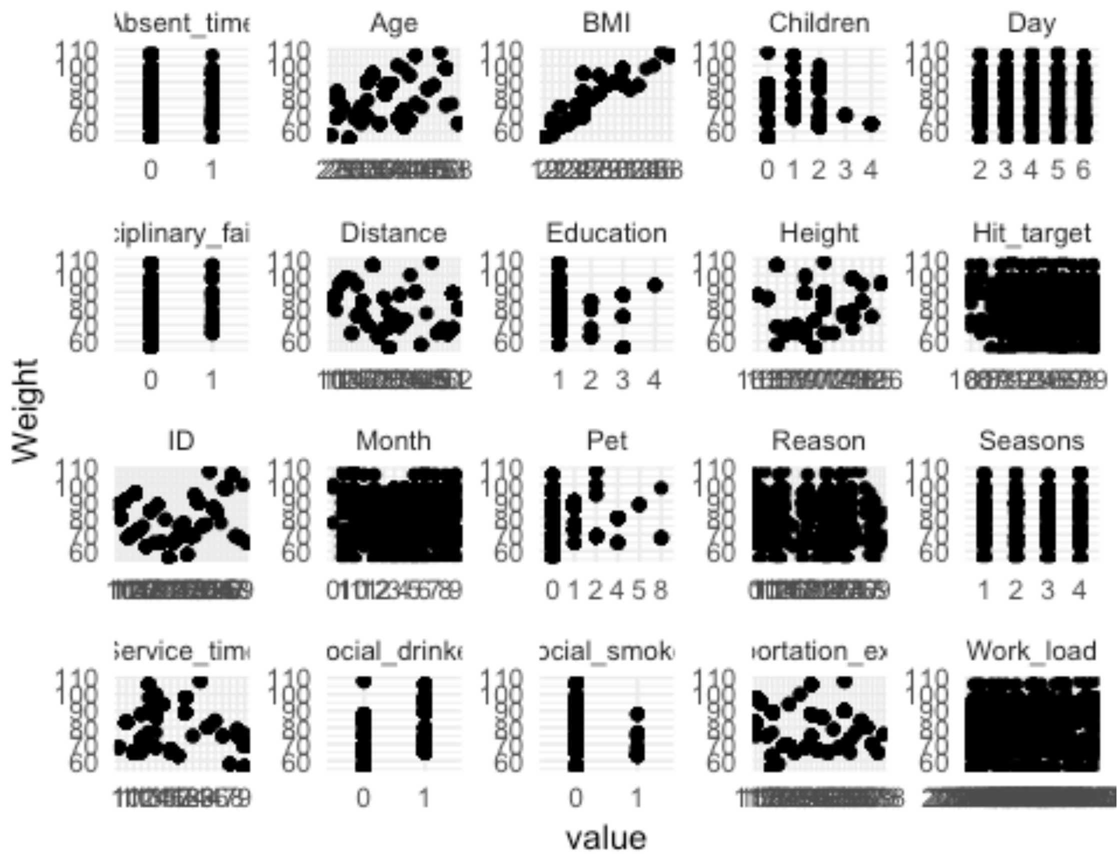


Weight

```r
#summary of weight
summary(dat$Weight)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   56.00   69.00   83.00   79.04   89.00  108.00

#histogram
ggplot(data = dat,
       aes(x = Weight)) +
  geom_histogram(bins = 15) +
  theme_minimal()
```




```r
#Scatterplots for variable 'Weight'
dat %>%
  gather(-Weight, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Weight)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

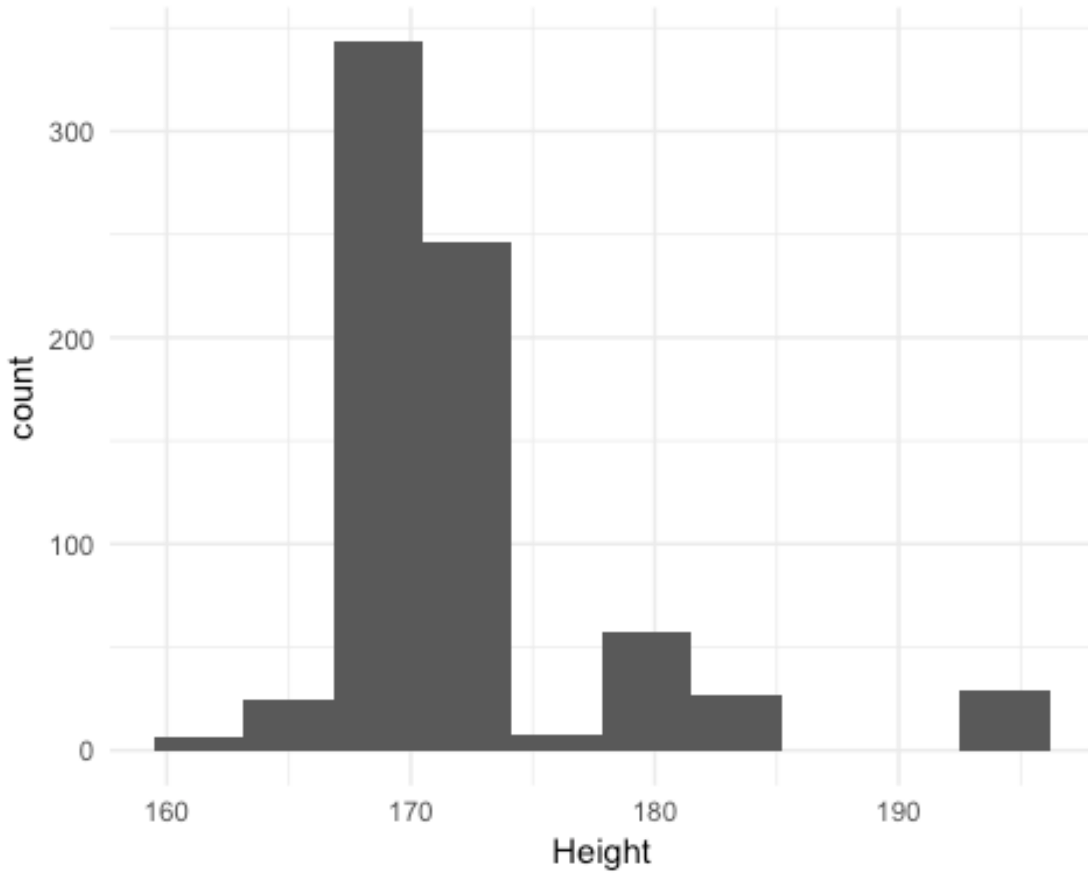


Height

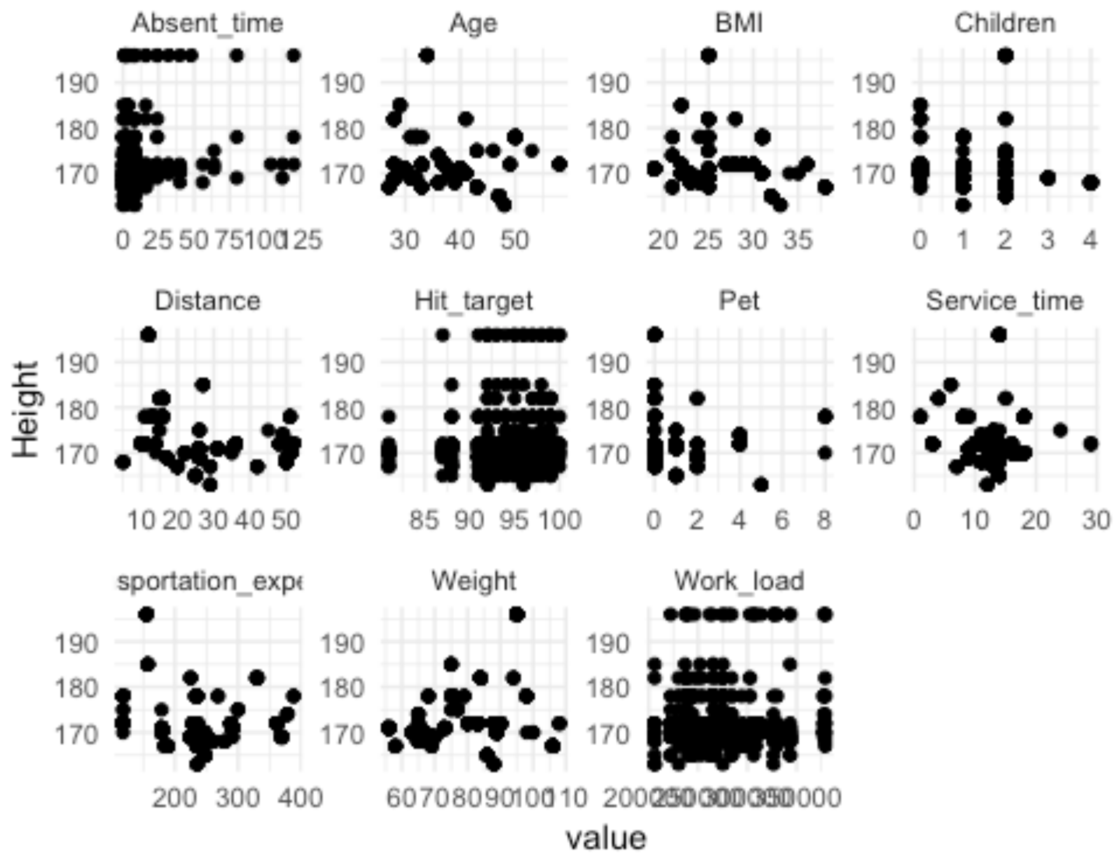
```r
#summary of height
summary(dat$Height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   163.0   169.0   170.0   172.1   172.0   196.0
```

```r
#histogram
ggplot(data = dat,
       aes(x = Height)) +
  geom_histogram(bins = 10) +
  theme_minimal()
```

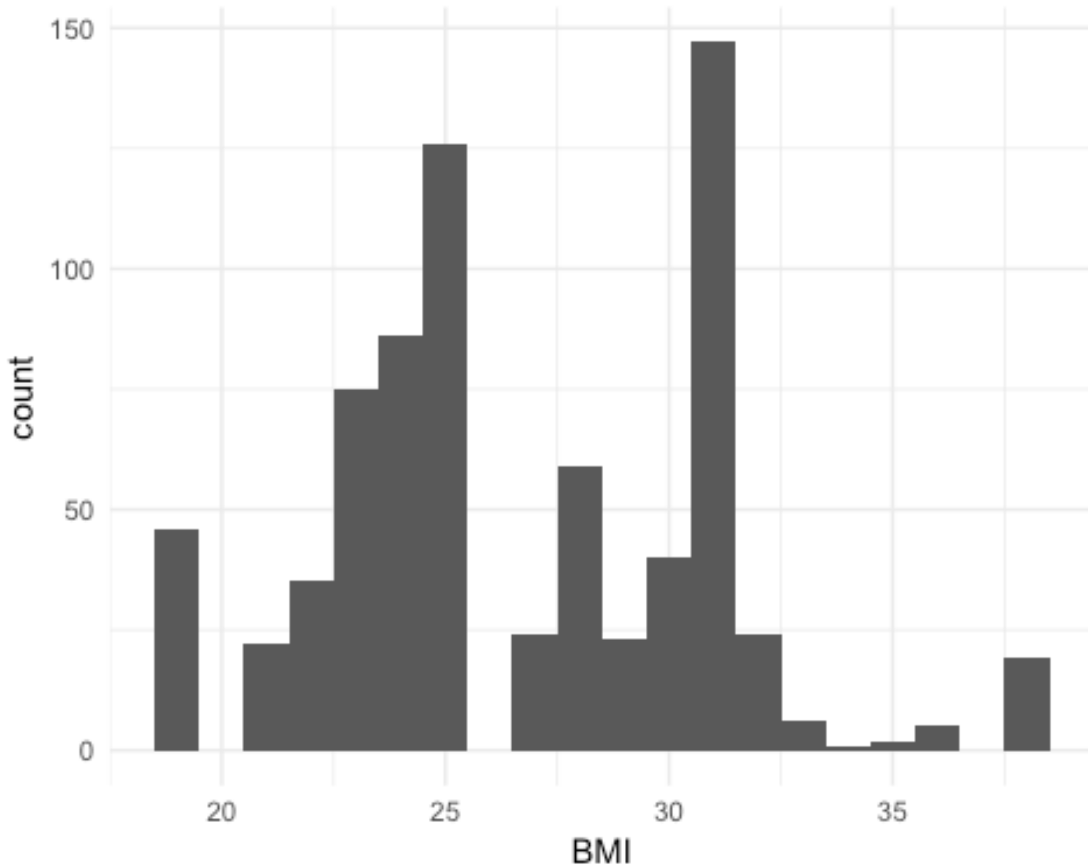



```
#Scatterplots for variable 'Height'
dat.num %>%
  gather(-Height, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Height)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

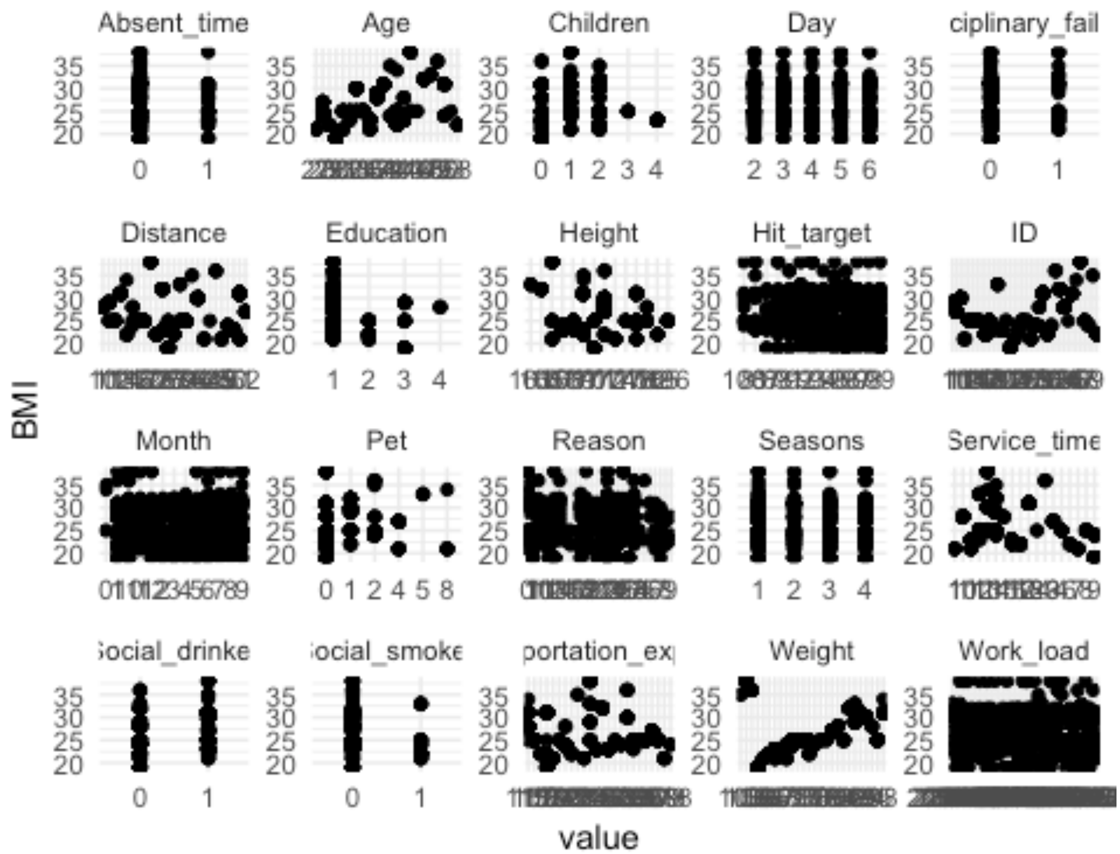


BMI

```
#summary for BMI
summary(dat$BMI)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   19.00   24.00   25.00   26.68   31.00   38.00

#histogram
ggplot(data = dat,
       aes(x = BMI)) +
  geom_histogram(binwidth = 1) +
  theme_minimal()
```




```
#Scatterplots for variable 'BMI'
dat %>%
  gather(-BMI, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = BMI)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



Additional Preprocessing

```
dat1 <- dat[-1]

#scale
scale <- sapply(dat1, is.numeric)
dat1[scale] <- lapply(dat1[scale],scale)
```

Initial Method Testing

```
R <- 50 # replications

# create the matrix to store values 1 row per model
err_matrix <- matrix(0, ncol=5, nrow=R)

sensitivity_matrix <- matrix(0, ncol=5, nrow=R)

fmeasure_matrix <- matrix(0, ncol=5, nrow=R)

gmean_matrix <- matrix(0, ncol=5, nrow=R)

# these are optional but I like to see how the model did each run so I can
check other output
KNNcm <- matrix(0, ncol=4, nrow=R)
```

```
glmcm <- matrix(0, ncol=4, nrow=R)
Treecm <- matrix(0, ncol=4, nrow=R)
rfcm <- matrix(0, ncol=4, nrow=R)
SVMcm <- matrix(0, ncol=4, nrow=R)

set.seed(1876)

for (r in 1:R){

# subsetting data to training and testing data
p <- .6 # proportion of data for training
w <- sample(1:nrow(dat1), nrow(dat1)*p, replace=F)
data_train <-dat1[w,]
data_test <- dat1[-w,]

  ################################################################ knn

#Running the classifier

  knn <- knn(data_train[-20],
                       test = data_test[-20],
                       cl=data_train$Absent_time, k=2)

#predict doesn't work with KNN for factors
 knntable <- table(knn, data_test$Absent_time)

#generate confusion matrix ( the 1 tells the model we care about that output)
 cm_KNN <-  confusionMatrix(data = knntable, reference = data_test[,-20],
positive = "1")

 KNNcm [[r,1]] <-  cm_KNN$table[1,1]
 KNNcm [[r,2]] <-  cm_KNN$table[1,2]
 KNNcm [[r,3]] <-  cm_KNN$table[2,1]
 KNNcm [[r,4]] <-  cm_KNN$table[2,2]

 err_matrix [[r,1]] <-  (cm_KNN$table[1,2]+cm_KNN$table[2,1])/nrow(
data_test)

  # store the errors (change the 1 to whichever model you have)

 sensitivity_matrix[[r, 1]] <- cm_KNN$byClass[1]

 fmeasure_matrix [[r, 1]] <- cm_KNN$byClass[7]

 gmean_matrix [[r, 1]] <- sqrt(cm_KNN$byClass[1]* cm_KNN$byClass[2])

  ################################################################### GLM
```

```
  model_glm_1 = suppressWarnings(
    train(Absent_time ~ .,
                        data = data_train,
                        method = "glm",
                        family = 'binomial')
                        )

  yhat_glm = predict(model_glm_1, newdata = data_test[,-20])

  cm_glm = confusionMatrix(data = yhat_glm, reference = data_test[,20],
positive = "1")

  glmcm [[r,1]] <-  cm_glm$table[1,1]
  glmcm [[r,2]] <-  cm_glm$table[1,2]
  glmcm [[r,3]] <-  cm_glm$table[2,1]
  glmcm [[r,4]] <-  cm_glm$table[2,2]

  err_matrix [[r,2]] <-  (cm_glm$table[1,2]+cm_glm$table[2,1])/nrow(
data_test)

  # store the errors (change the 1 to whichever model you have)

  sensitivity_matrix[[r, 2]] <- cm_glm$byClass[1]

  fmeasure_matrix [[r, 2]] <- cm_glm$byClass[7]

  gmean_matrix [[r, 2]] <- sqrt(cm_glm$byClass[1]* cm_glm$byClass[2])

  #####################################################Decision Tree

  tree_mod = rpart(Absent_time ~ ., data = data_train)

  #prediction
  yhat_tree = predict(tree_mod, data_test, type = 'class')

  #generate confusion matrix
 cm_tree <-  confusionMatrix(data = table(yhat_tree, data_test$Absent_time),
reference = data_test[,-20], positive = "1")

 Treecm[[r,1]] <-  cm_tree$table[1,1]
 Treecm[[r,2]] <-  cm_tree$table[1,2]
 Treecm[[r,3]] <-  cm_tree$table[2,1]
 Treecm[[r,4]] <-  cm_tree$table[2,2]

  #store the errors
  err_matrix[r, 3] = mean(yhat_tree != data_test$Absent_time)
```

```r
   # store the errors

 sensitivity_matrix[[r, 3]] <- cm_tree$byClass[1]

 cm_tree$byClass[1]

 fmeasure_matrix[[r, 3]] <- cm_tree$byClass[7]

 gmean_matrix[[r, 3]] <- sqrt(cm_tree$byClass[1]* cm_tree$byClass[2])

 #################################################### RF

   rf <- randomForest(Absent_time ~.,
                                   data=data_train,
                                   mtry=6,
                                   ntree=50,
                                   na.action=na.roughfix)

  yhat_rf = predict(rf, newdata = data_test, type= 'class')

  cm_rf = confusionMatrix(data = yhat_rf, reference = data_test[,20],
positive = "1")


  rfcm [[r,1]] <-  cm_rf$table[1,1]
  rfcm [[r,2]] <-  cm_rf$table[1,2]
  rfcm [[r,3]] <-  cm_rf$table[2,1]
  rfcm [[r,4]] <-  cm_rf$table[2,2]

 err_matrix [[r,4]] <-  (cm_glm$table[1,2]+cm_glm$table[2,1])/nrow(
data_test)

 sensitivity_matrix[[r, 4]] <- cm_rf$byClass[1]

 fmeasure_matrix[[r, 4]] <- cm_rf$byClass[7]

 gmean_matrix[[r, 4]] <- sqrt(cm_rf$byClass[1]* cm_rf$byClass[2])

  ################################################################ SVM

  csvm_absent =   svm(Absent_time~., data=data_train,
                   type='C-classification')

  #prediction
  y_hat_csvm = predict(csvm_absent, data_test[,-20])

#generate confusion matrix ( the 1 tells the model we care about that output)
  cm_SVM = confusionMatrix(data = y_hat_csvm, reference = data_test[,20],
```

```
positive = "1")

  SVMcm [[r,1]] <-  cm_SVM$table[1,1]
  SVMcm [[r,2]] <-  cm_SVM$table[1,2]
  SVMcm [[r,3]] <-  cm_SVM$table[2,1]
  SVMcm [[r,4]] <-  cm_SVM$table[2,2]

  # store the errors (change the 1 to whichever model you have)
  err_matrix[r,5] = (cm_SVM$table[1,2]+cm_SVM$table[2,1])/nrow(data_test)

  sensitivity_matrix[[r, 5]] <- cm_SVM$byClass[1]

  fmeasure_matrix [[r, 5]] <- cm_SVM$byClass[7]

  gmean_matrix [[r, 5]] <- sqrt(cm_SVM$byClass[1]* cm_SVM$byClass[2])

#statement indicates where in loop
  #cat("Finished Rep",r, "\n")
}
```

Change the matrix names to make easier to interpret

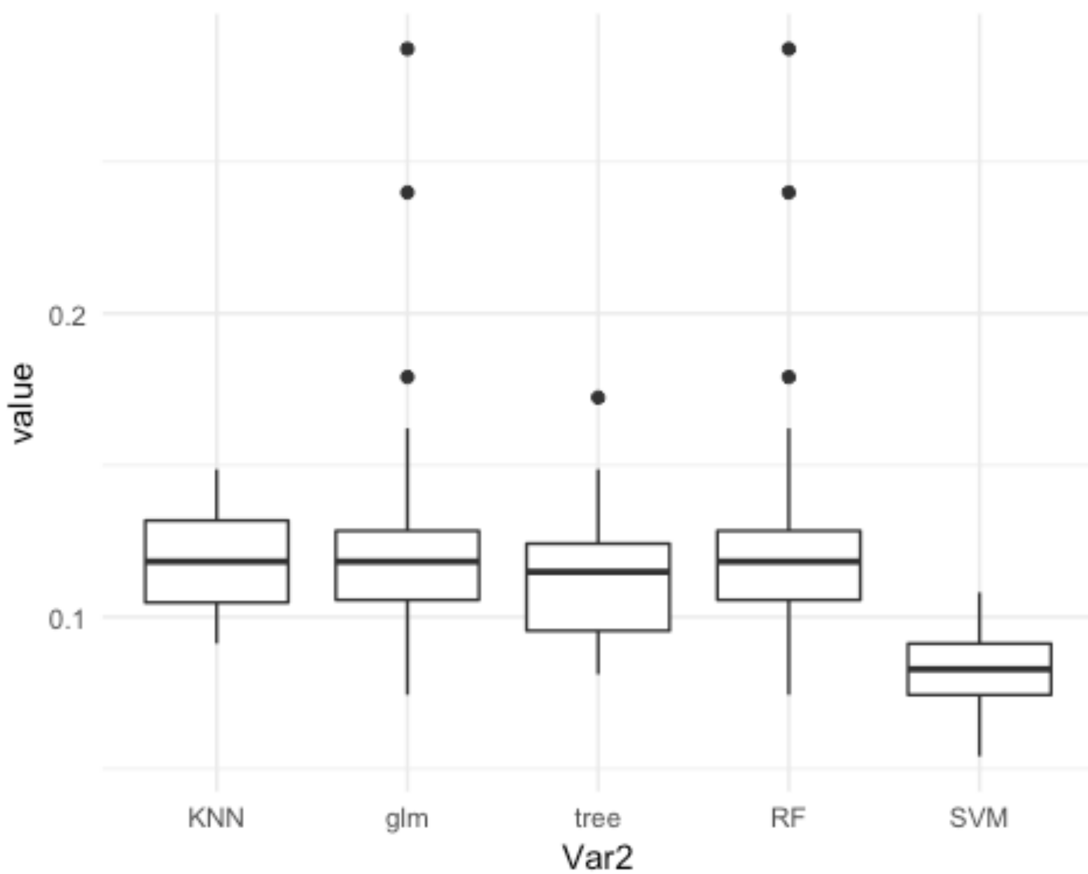
```
#rename the columns in the model
colnames(err_matrix) <- c("KNN","glm", "tree","RF", 'SVM')

colnames(sensitivity_matrix)<- c("KNN","glm", "tree","RF", 'SVM')

colnames(fmeasure_matrix) <- c("KNN","glm", "tree","RF", 'SVM')

colnames(gmean_matrix) <- c("KNN","glm", "tree","RF", 'SVM')


#rename the columns
colnames(KNNcm) <- c("True Negative","False Negative", "False Positive","True
Positive")

colnames(glmcm) <- c("True Negative","False Negative", "False Positive","True
Positive")

colnames(SVMcm) <- c("True Negative","False Negative", "False Positive","True
Positive")
```

save output

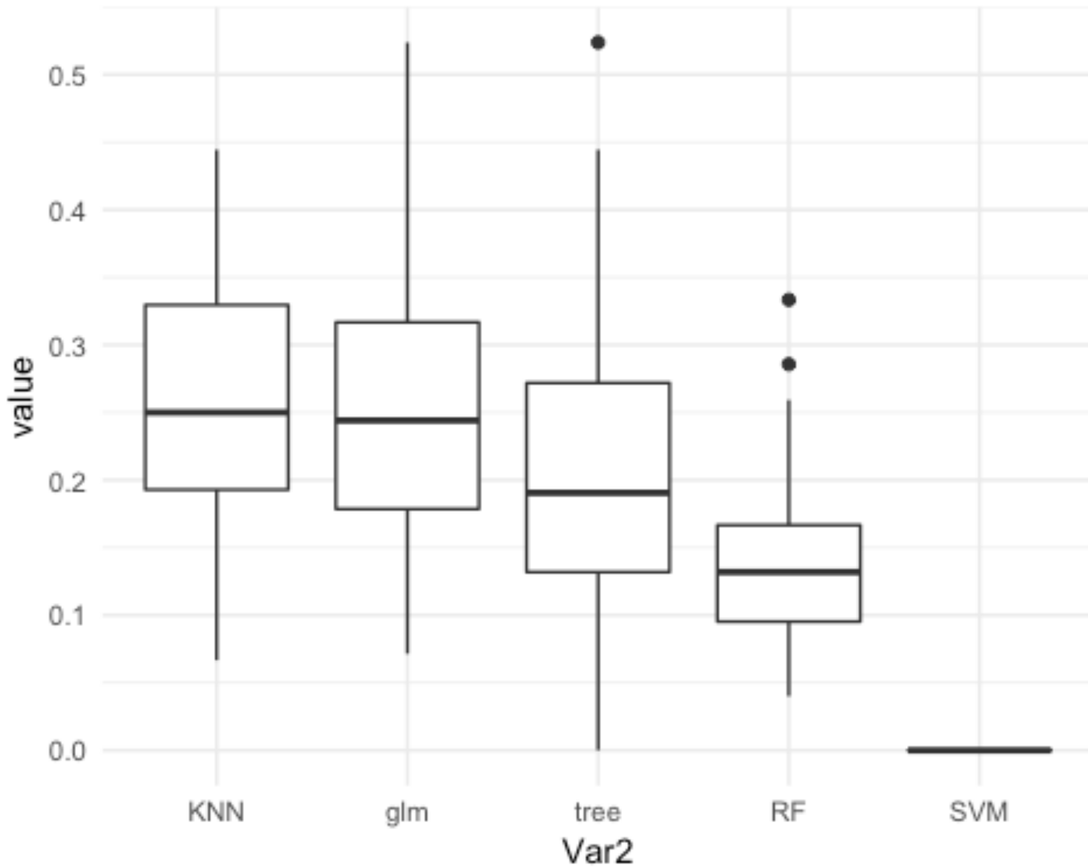
```
save(err_matrix, file='errmatrix.RData')
save(sensitivity_matrix, file='sensmatrix.RData')
save(fmeasure_matrix, file='fmeasmatrix.RData')
save(gmean_matrix, file='gmeanmatrix.RData')
```

load output

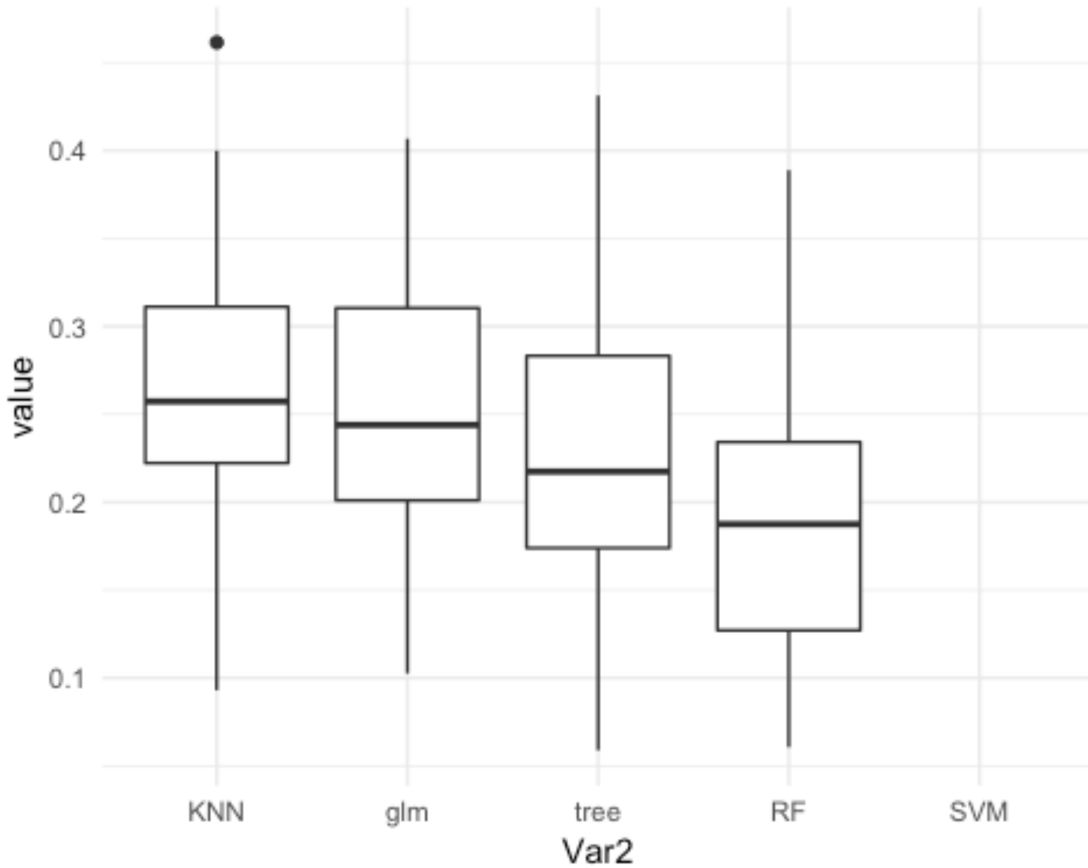
```r
load( file='errmatrix.RData')
load( file='sensmatrix.RData')
load( file='fmeasmatrix.RData')
load( file='gmeanmatrix.RData')

err_graph <- melt(err_matrix)

ggplot(err_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal()
```



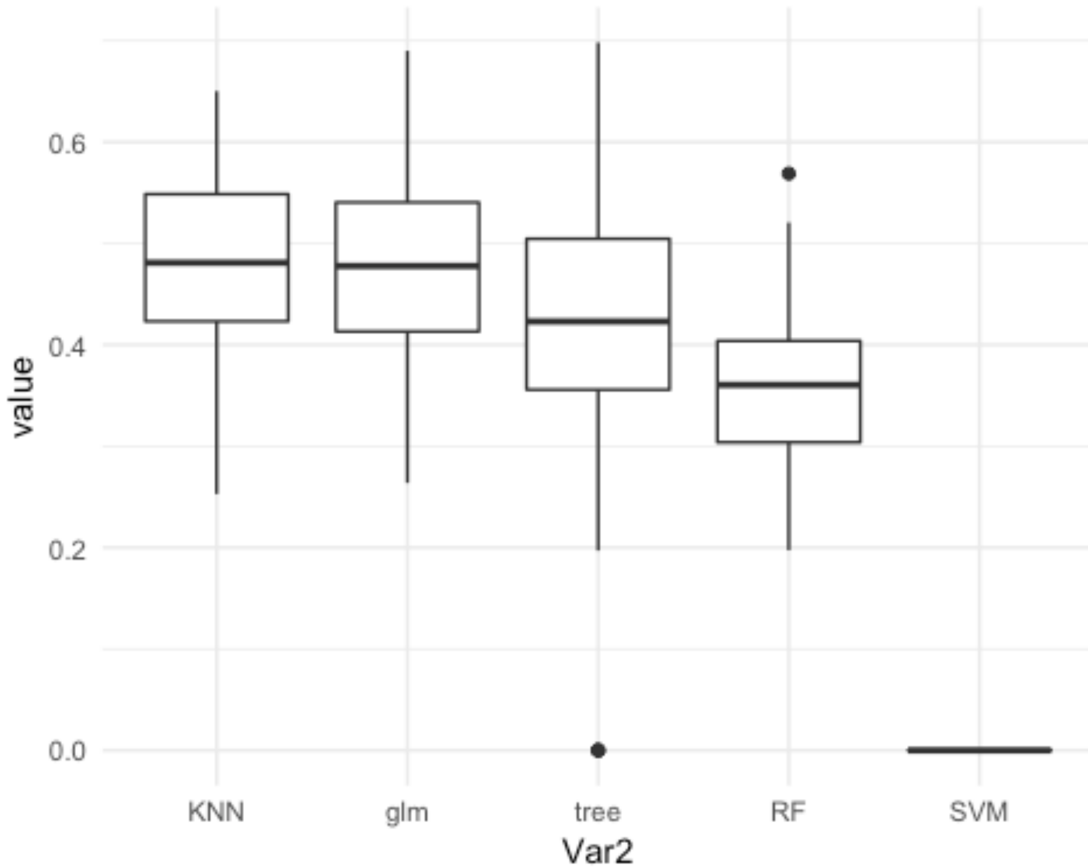

```r
sens_graph <- melt(sensitivity_matrix)

ggplot(sens_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal()
```




```
fmeas_graph <- melt(fmeasure_matrix)

ggplot(fmeas_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal()
```




```r
gmean_graph <- melt(gmean_matrix)

ggplot(gmean_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal()
```



KNN Optimization

```
set.seed(1876)

dat <- read_excel("Absenteeism_at_work.xls")
col <- c("ID", "Reason for absence", "Month of absence", "Day of the week",
"Seasons", "Disciplinary failure", "Education", "Social drinker",   "Social
smoker")
dat[col] <- lapply(dat[col], as.factor)
colnames(dat) <- c("ID", "Reason", "Month", "Day", "Seasons",
"Transportation_expense", "Distance", "Service_time", "Age", "Work_load",
"Hit_target", "Disciplinary_failure", "Education", "Children",
"Social_drinker", "Social_smoker", "Pet", "Weight", "Height", "BMI",
"Absent_time")

nums <- unlist(lapply(dat, is.numeric))
dat.num <- dat[ , nums]

#change variable represent missed time one day or greater
dat <- dat %>% mutate(Absent_time= ifelse(dat$Absent_time <=8,0,1))
str(dat)

## Classes 'tbl_df', 'tbl' and 'data.frame':    740 obs. of  21 variables:
##  $ ID                         : Factor w/ 36 levels "1","2","3","4",..: 11 36 3
```

```
7 11 3 10 20 14 1 ...
## $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 26 1 23
8 23 23 22 23 20 22 ...
## $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 8 8 8 ...
## $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 2 2 3 4
4 5 5 5 1 1 ...
## $ Seasons               : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
## $ Transportation_expense: num  289 118 179 279 289 179 361 260 155 235
...
## $ Distance              : num  36 13 51 5 36 51 52 50 12 11 ...
## $ Service_time          : num  13 18 18 14 13 18 3 11 14 14 ...
## $ Age                   : num  33 50 38 39 33 38 28 36 34 37 ...
## $ Work_load             : num  239554 239554 239554 239554 239554 ...
## $ Hit_target            : num  97 97 97 97 97 97 97 97 97 97 ...
## $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
## $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 3 ...
## $ Children              : num  2 1 0 2 2 0 1 4 2 1 ...
## $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
## $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1
...
## $ Pet                   : num  1 0 0 0 1 0 4 0 0 1 ...
## $ Weight                : num  90 98 89 68 90 89 80 65 95 88 ...
## $ Height                : num  172 178 170 168 172 170 172 168 196 172
...
## $ BMI                   : num  30 31 31 24 30 31 27 23 25 29 ...
## $ Absent_time           : num  0 0 0 0 0 0 0 0 1 0 ...
```

```
dat$Absent_time <- as.factor(dat$Absent_time)
#Transforming to Data Frame
dat <- as.data.frame(dat)

str(dat)
```

```
## 'data.frame':    740 obs. of  21 variables:
## $ ID                    : Factor w/ 36 levels "1","2","3","4",..: 11 36 3
7 11 3 10 20 14 1 ...
## $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 26 1 23
8 23 23 22 23 20 22 ...
## $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 8 8 8 ...
## $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 2 2 3 4
4 5 5 5 1 1 ...
## $ Seasons               : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
## $ Transportation_expense: num  289 118 179 279 289 179 361 260 155 235
```

```
...
##  $ Distance             : num  36 13 51 5 36 51 52 50 12 11 ...
##  $ Service_time         : num  13 18 18 14 13 18 3 11 14 14 ...
##  $ Age                  : num  33 50 38 39 33 38 28 36 34 37 ...
##  $ Work_load            : num  239554 239554 239554 239554 239554 ...
##  $ Hit_target           : num  97 97 97 97 97 97 97 97 97 97 ...
##  $ Disciplinary_failure : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Education            : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 3 ...
##  $ Children             : num  2 1 0 2 2 0 1 4 2 1 ...
##  $ Social_drinker       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
##  $ Social_smoker        : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1
...
##  $ Pet                  : num  1 0 0 0 1 0 4 0 0 1 ...
##  $ Weight               : num  90 98 89 68 90 89 80 65 95 88 ...
##  $ Height               : num  172 178 170 168 172 170 172 168 196 172
...
##  $ BMI                  : num  30 31 31 24 30 31 27 23 25 29 ...
##  $ Absent_time          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1
...

###Optimizing the KNN

#For the tunning of the KNN model, we are going to create another
traning/test data sets.

#scaling the data:
dat_v <- dat #we are going to use dat_v for the manipulation
scale <- sapply(dat_v, is.numeric)
dat_v[scale] <- lapply(dat_v[scale],scale)
head(dat_v)

##   ID Reason Month Day Seasons Transportation_expense   Distance
## 1 11     26     7   3       1              1.0107248  0.4292653
## 2 36      0     7   3       1             -1.5433353 -1.1209354
## 3  3     23     7   4       1             -0.6322379  1.4402658
## 4  7      7     7   5       1              0.8613645 -1.6601356
## 5 11     23     7   5       1              1.0107248  0.4292653
## 6  3     23     7   6       1             -0.6322379  1.4402658
##   Service_time        Age  Work_load Hit_target Disciplinary_failure
## 1    0.1017010 -0.5325083 -0.8176594  0.6382541                    0
## 2    1.2419848  2.0914456 -0.8176594  0.6382541                    1
## 3    1.2419848  0.2392429 -0.8176594  0.6382541                    0
## 4    0.3297577  0.3935931 -0.8176594  0.6382541                    0
## 5    0.1017010 -0.5325083 -0.8176594  0.6382541                    0
## 6    1.2419848  0.2392429 -0.8176594  0.6382541                    0
##   Education    Children Social_drinker Social_smoker        Pet     Weight
## 1         1  0.89311870              1             0  0.1927195  0.8510972
```

```
## 2         1 -0.01722267              1             0 -0.5658572  1.4720605
## 3         1 -0.92756405              1             0 -0.5658572  0.7734768
## 4         1  0.89311870              1             1 -0.5658572 -0.8565516
## 5         1  0.89311870              1             0  0.1927195  0.8510972
## 6         1 -0.92756405              1             0 -0.5658572  0.7734768
##        Height        BMI Absent_time
## 1 -0.01903313  0.7754078           0
## 2  0.97516826  1.0087554           0
## 3 -0.35043360  1.0087554           0
## 4 -0.68183407 -0.6246778           0
## 5 -0.01903313  0.7754078           0
## 6 -0.35043360  1.0087554           0
```

```
#predicting class:
AB_class <- dat_v[, 21]
names(AB_class) <- c(1:nrow(dat_v))
dat_v$ID <- c(1:nrow(dat_v))

dat_v <- dat_v[1:737,]
nrow(dat_v)
```

```
## [1] 737
```

```
rand_permute <- sample(x = nrow(dat_v), size = nrow(dat_v))

all_id_random <- dat_v[rand_permute, "ID"]
dat_v <- dat_v[,-1] #remove ID

#random samples for training test
validate_id <- as.character(all_id_random[1:248])
training_id <- as.character(all_id_random[249:737])

dat_v_train <- dat_v[training_id, ]
dat_v_val <- dat_v[validate_id, ]
AB_class_train <- AB_class[training_id]
AB_class_val <- AB_class[validate_id]
table(AB_class_train)
```

```
## AB_class_train
##   0   1
## 448  41
```

```
#Study significance of the variables
p <- .6 # proportion of data for training
w <- sample(1:nrow(dat_v), nrow(dat_v)*p, replace=F)
data_train <-dat_v[w,]
data_test <- dat_v[-w,]


rf <- randomForest(Absent_time ~.,
                   data=data_train,
```

```
                   mtry=6,
                   ntree=50,
                   na.action=na.roughfix)

impfact <- importance(rf)

impfact <- as.list(impfact)
names(impfact) <- colnames(dat_v[,-20])
impfact2 <- unlist(impfact)


most_sig_stats <- names(sort(desc(impfact2)))

#As per 'most_sig_stats' the 5 most significant variables for the prediction
are:
#'Seasons', 'Reason', 'Service_time', 'Month' and 'work_load'

#Re ordering variables by significance:

dat_v_train_ord <- dat_v_train[ c(most_sig_stats)]
str(dat_v_train_ord)

## 'data.frame':    489 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 1 23 25
25 24 18 27 2 28 26 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 4 8 6 4
9 4 3 11 5 7 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 3 4 1 3
2 2 2 1 3 5 ...
##  $ Work_load             : num [1:489, 1] -0.694 -0.818 -0.651 -1.262 -
1.679 ...
##  $ Hit_target            : num [1:489, 1] 0.903 0.638 1.167 1.167 -0.685
...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 2 1 3 2 1 2
2 4 3 3 ...
##  $ Age                   : num [1:489, 1] -0.841 -0.533 -0.996 3.326 -
0.533 ...
##  $ Distance              : num [1:489, 1] -0.851 0.429 -0.245 -1.054 0.429
...
##  $ Height                : num [1:489, 1] -0.516 -0.019 -0.185 -0.019 -
0.019 ...
##  $ Service_time          : num [1:489, 1] -0.126 0.102 -0.811 0.786 0.102
...
##  $ Transportation_expense: num [1:489, 1] 2.2056 1.0107 -0.6322 0.0996
1.0107 ...
##  $ Weight                : num [1:489, 1] -0.701 0.851 -1.788 -1.089 0.851
...
##  $ BMI                   : num [1:489, 1] -0.391 0.775 -1.791 -1.091 0.775
...
```

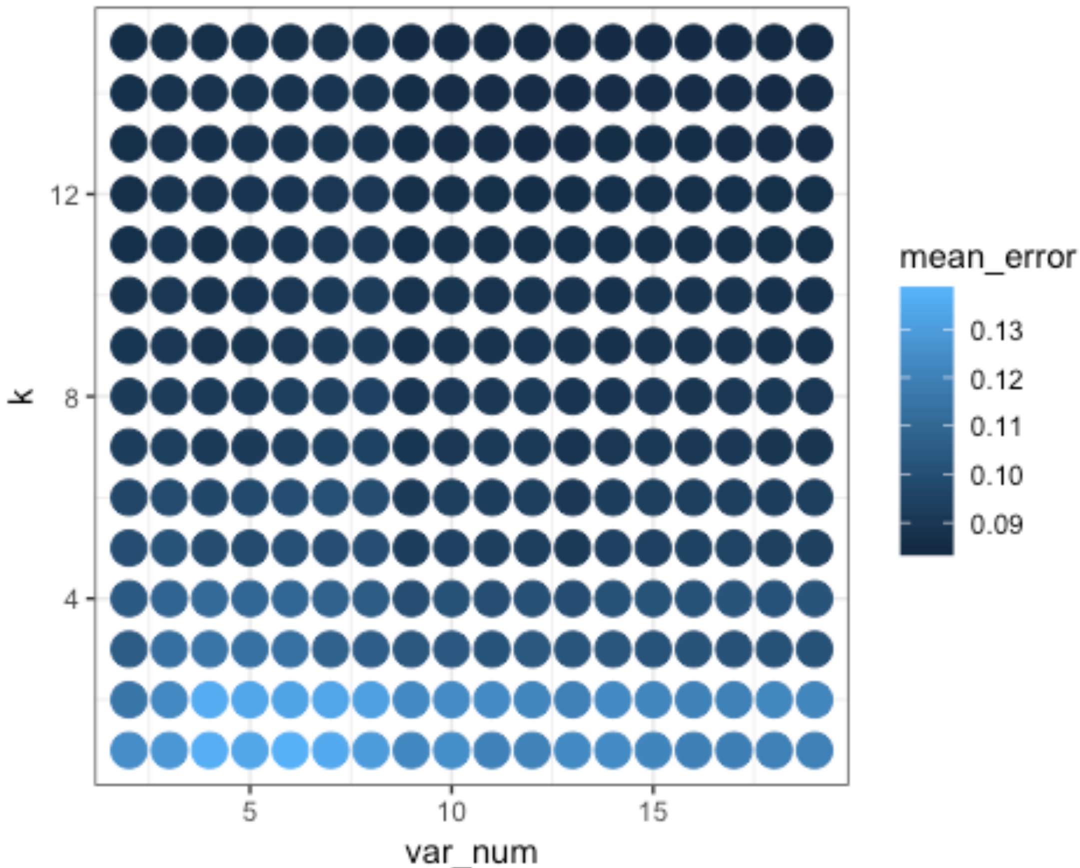
```
##  $ Children              : num [1:489, 1] 1.803 0.893 -0.928 0.893 0.893
...
##  $ Pet                   : num [1:489, 1] -0.566 0.193 -0.566 0.193 0.193
...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 1 2 1
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1
...
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 3 1 1 2
1 3 1 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1
...
```

```
dat_v_val_ord <- dat_v_val[, names(dat_v_train_ord)]
str(dat_v_val_ord)
```

```
## 'data.frame':    248 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 7 14 23
22 23 25 25 22 14 23 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 6 9 11
11 11 13 9 4 10 12 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 3 1 4 5
1 2 4 1 1 1 ...
##  $ Work_load             : num [1:248, 1] -0.866 -1.679 -0.166 -0.166 -
0.166 ...
##  $ Hit_target            : num [1:248, 1] 1.167 -0.685 -1.743 -1.743 -
1.743 ...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 3 1 4 4 4 4
1 2 4 4 ...
##  $ Age                   : num [1:248, 1] -1.304 -1.304 -0.996 -0.533
1.011 ...
##  $ Distance              : num [1:248, 1] -0.245 1.508 -0.245 0.429 -0.649
...
##  $ Height                : num [1:248, 1] -0.516 -0.019 -0.185 -0.019 -
0.848 ...
##  $ Service_time          : num [1:248, 1] -0.811 -2.179 -0.811 0.102 0.102
...
##  $ Transportation_expense: num [1:248, 1] 0.0548 2.0861 -0.6322 1.0107
0.2042 ...
##  $ Weight                : num [1:248, 1] -0.7789 0.0749 -1.788 0.8511
2.093 ...
##  $ BMI                   : num [1:248, 1] -0.6247 0.0754 -1.7914 0.7754
2.6422 ...
##  $ Children              : num [1:248, 1] -0.0172 -0.0172 -0.9276 0.8931 -
0.0172 ...
##  $ Pet                   : num [1:248, 1] 0.951 2.468 -0.566 0.193 -0.566
...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 1 2 1 2 2 1 2 1 2 1
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
```

```
...
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 3 1 1 1
1 3 1 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...

#Monte Carlo Validation:

size <- length(training_id)
(2/3) * length(training_id)

## [1] 326

training_family_L <- lapply(1:500, function(j) {
  perm <- sample(1:size, size = size, replace = F)
  shuffle <- training_id[perm]
  trn <- shuffle[1:326]
  trn
})

validation_family_L <- lapply(training_family_L,
                              function(x) setdiff(training_id, x))

#Finding an optimal set of variables and optimal k

N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 18.05547

K <- seq(from = 1, to = 15, by = 1)
times <- 500 * length(N) * length(K)

#Execution of the test with loops

paramter_errors_df <- data.frame(mc_index = as.integer(rep(NA, times =
times)),
                                 var_num = as.integer(rep(NA, times =
times)),
                                 k = as.integer(rep(NA, times = times)),
                                 error = as.numeric(rep(NA, times = times)))

#Core knn_model:
# j = index, n = length of range of variables, k=k
core_knn <- function(j, n, k) {
  knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
                     test = dat_v_train_ord[validation_family_L[[j]], 1:n],
                     cl = AB_class_train[training_family_L[[j]]],
                     k = k)
  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])
  err <- (tbl[1, 2] + tbl[2, 1])/(tbl[1, 2] + tbl[2, 1]+tbl[1, 1] + tbl[2,
```

```r
2])
  err
}

param_df1 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df <- merge(param_df1, data.frame(k = K))

knn_err_est_df <- ddply(param_df[1:times, ], .(mc_index, var_num, k),
function(df) {
  err <- core_knn(df$mc_index[1], df$var_num[1], df$k[1])
  err
})

head(knn_err_est_df)

##   mc_index var_num k        V1
## 1        1       2 1 0.1411043
## 2        1       2 2 0.1288344
## 3        1       2 3 0.1349693
## 4        1       2 4 0.1288344
## 5        1       2 5 0.1226994
## 6        1       2 6 0.1226994

names(knn_err_est_df)[4] <- "error"

mean_errs_df <- ddply(knn_err_est_df, .(var_num, k), function(df)
mean(df$error))
head(mean_errs_df)

##   var_num k         V1
## 1       2 1 0.12441718
## 2       2 2 0.11549693
## 3       2 3 0.10521472
## 4       2 4 0.10429448
## 5       2 5 0.09928834
## 6       2 6 0.09651534

names(mean_errs_df)[3] <- "mean_error"

library(ggplot2)
ggplot(data = mean_errs_df, aes(x = var_num, y = k, color = mean_error)) +
geom_point(size = 5) +
  theme_bw()
```




```
#This is the model that produces the lowest mean error var_num = 6 and k = 1:
mean_errs_df[which.min(mean_errs_df$mean_error), ]

##     var_num  k mean_error
## 165      12 15 0.08457669

mean_errs_df %>% arrange(mean_error)

##     var_num  k mean_error
## 1        12 15 0.08457669
## 2        11 15 0.08462577
## 3        13 15 0.08465031
## 4        18 15 0.08469939
## 5        19 15 0.08469939
## 6        14 15 0.08474847
## 7        10 15 0.08479755
## 8        17 15 0.08479755
## 9         9 15 0.08488344
## 10       15 15 0.08492025
## 11       16 15 0.08495706
## 12       18 14 0.08514110
## 13       13 14 0.08515337
## 14       13 13 0.08517791
## 15       12 13 0.08521472
```

```
## 16        11 14 0.08526380
## 17        19 14 0.08531288
## 18        12 14 0.08533742
## 19        10 14 0.08541104
## 20        11 13 0.08542331
## 21        14 14 0.08542331
## 22        18 13 0.08548466
## 23        19 13 0.08549693
## 24        15 14 0.08553374
## 25        17 14 0.08557055
## 26        10 13 0.08558282
## 27        16 14 0.08566871
## 28        17 13 0.08569325
## 29        14 13 0.08571779
## 30        16 13 0.08585276
## 31         9 13 0.08588957
## 32         9 14 0.08596319
## 33         2 15 0.08598773
## 34        13 12 0.08607362
## 35        15 13 0.08611043
## 36        12 12 0.08612270
## 37        18 12 0.08619632
## 38        18 11 0.08623313
## 39        19 11 0.08628221
## 40         2 13 0.08629448
## 41        11 12 0.08631902
## 42        19 12 0.08640491
## 43         9 11 0.08646626
## 44        13 11 0.08646626
## 45         2 14 0.08657669
## 46        17 12 0.08657669
## 47        14 11 0.08663804
## 48        14 12 0.08671166
## 49         2 11 0.08676074
## 50         9 12 0.08679755
## 51        11 11 0.08680982
## 52        16 12 0.08683436
## 53         2 12 0.08684663
## 54        17 11 0.08684663
## 55        12 11 0.08690798
## 56        16 11 0.08696933
## 57        15 12 0.08698160
## 58         4 11 0.08700613
## 59        10 12 0.08700613
## 60        15 11 0.08700613
## 61         4 15 0.08701840
## 62         6 15 0.08705521
## 63         3 15 0.08706748
## 64        10 11 0.08706748
## 65         5 15 0.08721472
```

```
## 66        18  9 0.08725153
## 67        19  9 0.08726380
## 68         4 13 0.08732515
## 69        14 10 0.08732515
## 70        19 10 0.08743558
## 71         8 15 0.08746012
## 72        18 10 0.08746012
## 73        14  9 0.08747239
## 74         9  9 0.08748466
## 75         3 14 0.08752147
## 76         9 10 0.08752147
## 77         7 15 0.08766871
## 78         4 14 0.08781595
## 79         4 12 0.08782822
## 80        11 10 0.08785276
## 81         4  9 0.08791411
## 82         3 13 0.08792638
## 83         5 13 0.08798773
## 84        13 10 0.08798773
## 85         2 10 0.08801227
## 86        16  9 0.08802454
## 87         6 14 0.08803681
## 88         4 10 0.08804908
## 89        16 10 0.08804908
## 90        15  9 0.08811043
## 91         5 11 0.08812270
## 92        17  9 0.08817178
## 93         6 13 0.08818405
## 94        13  9 0.08820859
## 95        11  9 0.08824540
## 96        17 10 0.08828221
## 97         8 13 0.08835583
## 98        10  9 0.08835583
## 99         5 14 0.08840491
## 100       10 10 0.08844172
## 101        5  9 0.08846626
## 102        3 12 0.08850307
## 103        7 13 0.08858896
## 104       12 10 0.08860123
## 105        8 14 0.08871166
## 106       15 10 0.08871166
## 107        5 12 0.08876074
## 108        7 14 0.08880982
## 109       12  9 0.08880982
## 110        6 12 0.08890798
## 111        6 11 0.08900613
## 112       13  7 0.08900613
## 113        9  8 0.08901840
## 114        5 10 0.08909202
## 115       13  8 0.08917791
```

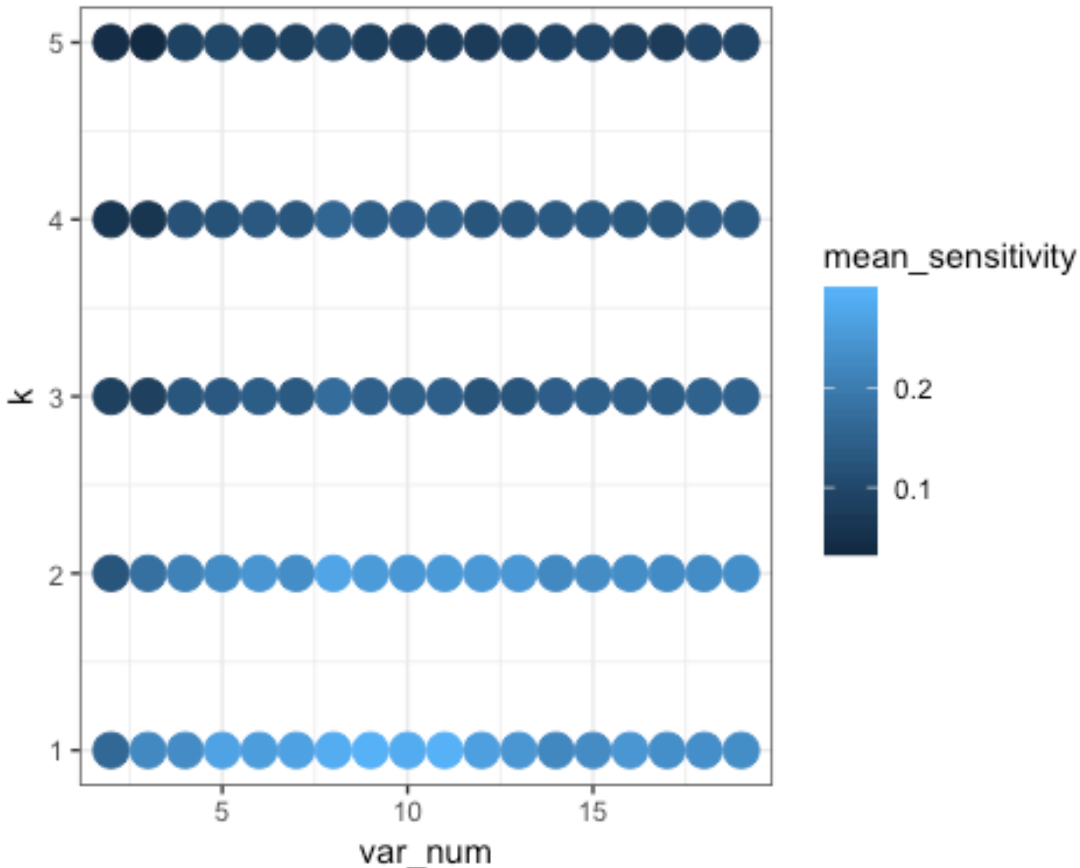
```
## 116        3 11 0.08921472
## 117       14  8 0.08923926
## 118        9  7 0.08928834
## 119        2  9 0.08939877
## 120       16  8 0.08949693
## 121       18  7 0.08965644
## 122       19  7 0.08965644
## 123        7 12 0.08980368
## 124        6 10 0.08982822
## 125        7 11 0.08986503
## 126       19  8 0.08988957
## 127        8 12 0.08996319
## 128        6  9 0.08998773
## 129        3 10 0.09001227
## 130       14  7 0.09002454
## 131       18  8 0.09024540
## 132       16  7 0.09028221
## 133       17  8 0.09029448
## 134       10  7 0.09033129
## 135        8 11 0.09034356
## 136       11  8 0.09034356
## 137       15  8 0.09042945
## 138        3  9 0.09044172
## 139       17  7 0.09050307
## 140       15  7 0.09066258
## 141       10  8 0.09071166
## 142       13  6 0.09087117
## 143       12  8 0.09107975
## 144        7 10 0.09109202
## 145        4  8 0.09114110
## 146       11  7 0.09125153
## 147        4  7 0.09126380
## 148       12  7 0.09133742
## 149       13  5 0.09134969
## 150        5  8 0.09136196
## 151        2  8 0.09144785
## 152        7  9 0.09153374
## 153        8  9 0.09175460
## 154        9  6 0.09184049
## 155        5  7 0.09185276
## 156        8 10 0.09191411
## 157        3  8 0.09207362
## 158        2  7 0.09266258
## 159       11  6 0.09268712
## 160       18  6 0.09273620
## 161        9  5 0.09278528
## 162       14  6 0.09278528
## 163       10  6 0.09293252
## 164       12  6 0.09298160
## 165       15  6 0.09301840
```

```
## 166          6  8 0.09303067
## 167         16  6 0.09311656
## 168         17  6 0.09319018
## 169          6  7 0.09321472
## 170         19  6 0.09334969
## 171          3  7 0.09336196
## 172         12  5 0.09350920
## 173         11  5 0.09364417
## 174          7  8 0.09377914
## 175         15  5 0.09390184
## 176         18  5 0.09406135
## 177         19  5 0.09411043
## 178         10  5 0.09420859
## 179          8  8 0.09422086
## 180         14  5 0.09434356
## 181          7  7 0.09446626
## 182          8  7 0.09466258
## 183         16  5 0.09477301
## 184         17  5 0.09499387
## 185          2  6 0.09651534
## 186          4  6 0.09674847
## 187          5  6 0.09811043
## 188          3  6 0.09826994
## 189          5  5 0.09834356
## 190          4  5 0.09910429
## 191          8  6 0.09916564
## 192          2  5 0.09928834
## 193          6  6 0.09937423
## 194         13  4 0.09944785
## 195          7  5 0.09955828
## 196          8  5 0.09975460
## 197         11  4 0.09977914
## 198          9  4 0.09979141
## 199          7  6 0.09998773
## 200          6  5 0.10025767
## 201         12  4 0.10058896
## 202         17  3 0.10072393
## 203         16  4 0.10117791
## 204         10  4 0.10141104
## 205         14  4 0.10141104
## 206         18  3 0.10148466
## 207         19  3 0.10152147
## 208         15  4 0.10158282
## 209         11  3 0.10171779
## 210         15  3 0.10173006
## 211         18  4 0.10193865
## 212         19  4 0.10212270
## 213          3  5 0.10213497
## 214         17  4 0.10214724
## 215         16  3 0.10217178
```

```
## 216       13  3 0.10240491
## 217       14  3 0.10287117
## 218        9  3 0.10338650
## 219       12  3 0.10370552
## 220       10  3 0.10382822
## 221        2  4 0.10429448
## 222        8  4 0.10498160
## 223        2  3 0.10521472
## 224        8  3 0.10629448
## 225        7  3 0.10776687
## 226        7  4 0.10777914
## 227        3  4 0.10907975
## 228        6  4 0.10971779
## 229        5  4 0.10984049
## 230        4  4 0.11077301
## 231        3  3 0.11341104
## 232        6  3 0.11352147
## 233        5  3 0.11403681
## 234        4  3 0.11504294
## 235        2  2 0.11549693
## 236       17  1 0.11896933
## 237       16  1 0.11942331
## 238       19  1 0.11973006
## 239       18  1 0.11979141
## 240       17  2 0.11986503
## 241       13  2 0.11990184
## 242       16  2 0.12008589
## 243       11  1 0.12034356
## 244       12  1 0.12036810
## 245       15  1 0.12125153
## 246       15  2 0.12157055
## 247       19  2 0.12157055
## 248       12  2 0.12158282
## 249       18  2 0.12226994
## 250        9  1 0.12231902
## 251        3  2 0.12271166
## 252        9  2 0.12299387
## 253       13  1 0.12304294
## 254       14  2 0.12314110
## 255       14  1 0.12344785
## 256       11  2 0.12358282
## 257       10  2 0.12380368
## 258        2  1 0.12441718
## 259       10  1 0.12492025
## 260        3  1 0.12812270
## 261        8  1 0.12987730
## 262        8  2 0.13150920
## 263        6  2 0.13267485
## 264        7  2 0.13336196
## 265        5  1 0.13376687
```

```
## 266       5  2 0.13398773
## 267       7  1 0.13505521
## 268       4  2 0.13568098
## 269       4  1 0.13633129
## 270       6  1 0.13739877

#load files from previous analysis
#load( file='errmatrix.RData')
#load( file='sensmatrix.RData')
#load( file='fmeasmatrix.RData')
#load( file='gmeanmatrix.RData')

#eventually run old to compare with new.
#We see that although error lower, other metrics hurt. We care about
identifying >8 hours  so modify

#Repeat with sensitivity

N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 18.05547

K <- seq(from = 1, to = 5, by = 1)
times <- 500 * length(N) * length(K)

core_knn_sen <- function(j, n, k) {
  knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
                     test = dat_v_train_ord[validation_family_L[[j]], 1:n],
                     cl = AB_class_train[training_family_L[[j]]],
                     k = k)

  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])

  #generate confusion matrix ( the 1 tells the model we care about that
output)
  cm_KNN <-  confusionMatrix(data = tbl, reference
=AB_class_train[validation_family_L[[j]]], positive = "1")

  sen <- cm_KNN$byClass[1]
  sen
}

param_df1_2 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df_2 <- merge(param_df1_2, data.frame(k = K))

knn_err_est_df_2 <- ddply(param_df_2[1:times, ], .(mc_index, var_num, k),
function(df) {
  sen <- core_knn_sen(df$mc_index[1], df$var_num[1], df$k[1])
```

```
  sen
})

head(knn_err_est_df_2)

##   mc_index var_num k Sensitivity
## 1        1       2 1   0.2941176
## 2        1       2 2   0.2352941
## 3        1       2 3   0.1176471
## 4        1       2 4   0.1176471
## 5        1       2 5   0.0000000
## 6        1       3 1   0.2941176

names(knn_err_est_df_2)[4] <- "Sensitivity"

mean_sens_df <- ddply(knn_err_est_df_2, .(var_num, k), function(df)
mean(df$Sensitivity))
head(mean_sens_df)

##   var_num k         V1
## 1       2 1 0.16291018
## 2       2 2 0.12607819
## 3       2 3 0.08472480
## 4       2 4 0.05907006
## 5       2 5 0.04329627
## 6       3 1 0.22416430

names(mean_sens_df)[3] <- "mean_sensitivity"

library(ggplot2)
ggplot(data = mean_sens_df, aes(x = var_num, y = k, color =
mean_sensitivity)) + geom_point(size = 5) +
  theme_bw()
```




```
#This is the model that produces the lowest mean error var_num = 9 and k = 1:
mean_sens_df[which.max(mean_sens_df$mean_sensitivity), ]

##    var_num k mean_sensitivity
## 46      11 1        0.2963108

mean_sens_df %>% arrange(desc(mean_sensitivity))

##    var_num k mean_sensitivity
## 1       11 1       0.29631075
## 2        9 1       0.29628928
## 3        8 1       0.28814862
## 4       10 1       0.28756153
## 5        8 2       0.27505663
## 6        5 1       0.26996587
## 7        7 1       0.26887811
## 8       12 1       0.26430625
## 9        6 1       0.26087161
## 10       9 2       0.25705663
## 11      11 2       0.25557954
## 12      12 2       0.25294873
## 13      13 2       0.25035087
## 14      10 2       0.25005292
## 15      13 1       0.24888999
```

```
## 16        16 1        0.24867340
## 17         6 2        0.24750280
## 18        19 2        0.23570491
## 19        18 1        0.23469156
## 20        17 1        0.23467278
## 21        19 1        0.23453448
## 22        16 2        0.23311113
## 23         7 2        0.23302910
## 24         4 1        0.23064666
## 25        15 2        0.23005933
## 26        15 1        0.23000072
## 27        18 2        0.22964669
## 28         5 2        0.22959821
## 29        17 2        0.22906893
## 30        14 2        0.22510482
## 31         3 1        0.22416430
## 32        14 1        0.22362974
## 33         4 2        0.21222337
## 34         3 2        0.17941810
## 35         8 3        0.17151230
## 36         2 1        0.16291018
## 37         8 4        0.15989484
## 38        18 3        0.15178067
## 39        19 3        0.15172991
## 40         9 3        0.14798958
## 41        11 4        0.14715629
## 42        10 3        0.14710532
## 43        11 3        0.14647028
## 44        16 3        0.14518043
## 45        15 3        0.14512881
## 46        10 4        0.14383488
## 47        14 3        0.14348699
## 48         6 3        0.14208558
## 49         9 4        0.14203962
## 50        17 3        0.14130806
## 51        18 4        0.13990686
## 52        15 4        0.13558766
## 53        19 4        0.13502490
## 54         7 3        0.13414150
## 55        14 4        0.13343245
## 56         6 4        0.13282813
## 57         5 3        0.13190753
## 58        16 4        0.13087010
## 59        17 4        0.12992587
## 60         7 4        0.12873013
## 61        13 4        0.12818373
## 62         4 3        0.12756461
## 63         2 2        0.12607819
## 64        13 3        0.12541192
## 65        12 4        0.12522221
```

```
## 66       12 3       0.12341400
## 67        5 4       0.12034291
## 68        4 4       0.11784016
## 69        8 5       0.10133393
## 70        5 5       0.09743037
## 71       15 5       0.09410982
## 72       19 5       0.09385348
## 73       18 5       0.09346801
## 74       14 5       0.08816972
## 75        6 5       0.08677699
## 76        4 5       0.08626123
## 77        2 3       0.08472480
## 78        7 5       0.08460925
## 79       16 5       0.08313597
## 80        3 3       0.08257101
## 81       13 5       0.08188636
## 82        9 5       0.08156346
## 83       11 5       0.07816682
## 84       10 5       0.07594044
## 85       17 5       0.07480186
## 86       12 5       0.07339861
## 87        3 4       0.06159546
## 88        2 4       0.05907006
## 89        2 5       0.04329627
## 90        3 5       0.03795239
```

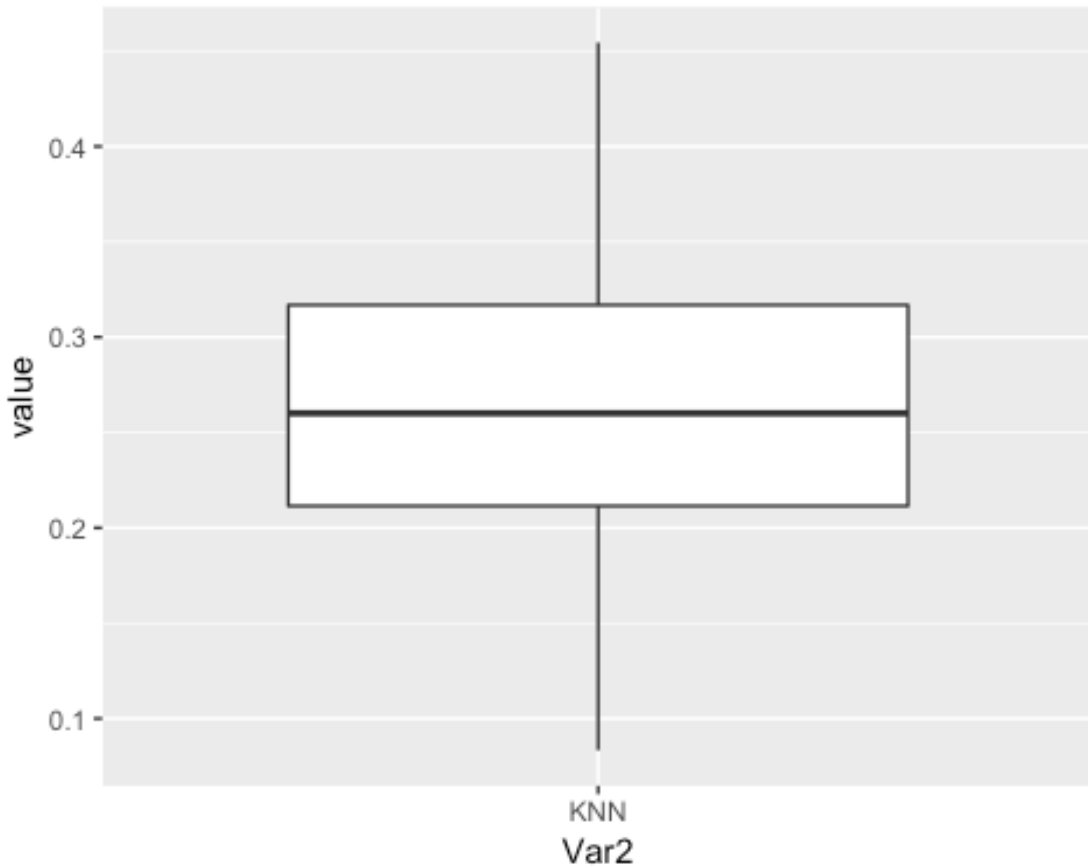
```
#Best KNN:

KNN_10_1 <- knn(train = dat_v_train_ord[, 1:9],
               dat_v_val_ord[, 1:9], AB_class_train,
               k = 1)

tbl_bm_val <- table(KNN_10_1, AB_class_val)
tbl_bm_val
```

```
##         AB_class_val
## KNN_10_1   0   1
##         0 210  17
##         1  16   5
```

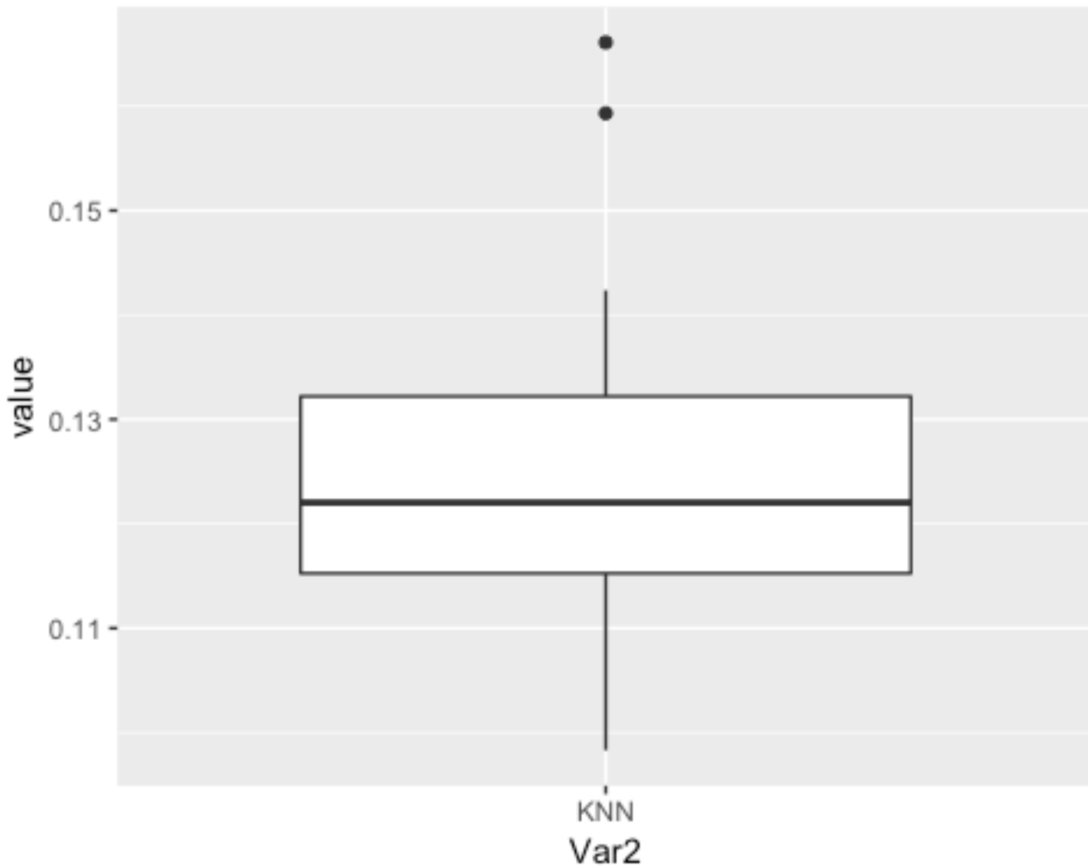
```
cm_KNN_opt <-  confusionMatrix(data = tbl_bm_val, reference = dat_v_val_ord[,
1:6], positive = "1")

R <- 50 # replications

# create the matrix to store values 1 row per model
err_matrix_opt <- matrix(0, ncol=1, nrow=R)

sensitivity_matrix_opt <- matrix(0, ncol=1, nrow=R)
```

```
fmeasure_matrix_opt <- matrix(0, ncol=1, nrow=R)

gmean_matrix_opt <- matrix(0, ncol=1, nrow=R)

# these are optional but I like to see how the model did each run so I can
check other output
KNNcm <- matrix(0, ncol=4, nrow=R)

dat_smaller <- dat[, names(dat_v_train_ord)]
dat_smaller[,20] <- dat$Absent_time

dat_smaller <- dat_smaller[1:737,] # remove lines with non-meaningful data

scale <- sapply(dat_smaller, is.numeric)
dat_smaller[scale] <- lapply(dat_smaller[scale],scale)
head(dat_smaller)

##   Reason Month Day  Work_load Hit_target Seasons        Age   Distance
## 1     26     7   3 -0.8160263  0.6374158       1 -0.5292037  0.4295322
## 2      0     7   3 -0.8160263  0.6374158       1  2.1019046 -1.1199466
## 3     23     7   4 -0.8160263  0.6374158       1  0.2446517  1.4400619
## 4      7     7   5 -0.8160263  0.6374158       1  0.3994228 -1.6588958
## 5     23     7   5 -0.8160263  0.6374158       1 -0.5292037  0.4295322
## 6     23     7   6 -0.8160263  0.6374158       1  0.2446517  1.4400619
##        Height Service_time Transportation_expense     Weight        BMI
## 1 -0.01930235    0.1025410              1.0078374  0.8561660  0.7818833
## 2  0.97319750    1.2406839             -1.5458897  1.4779119  1.0158452
## 3 -0.35013563    1.2406839             -0.6349110  0.7784478  1.0158452
## 4 -0.68096891    0.3301696              0.8584966 -0.8536352 -0.6218877
## 5 -0.01930235    0.1025410              1.0078374  0.8561660  0.7818833
## 6 -0.35013563    1.2406839             -0.6349110  0.7784478  1.0158452
##      Children        Pet Social_drinker Social_smoker Education
## 1  0.89294976  0.2057297              1             0         1
## 2 -0.01603363 -0.5678559              1             0         1
## 3 -0.92501702 -0.5678559              1             0         1
## 4  0.89294976 -0.5678559              1             1         1
## 5  0.89294976  0.2057297              1             0         1
## 6 -0.92501702 -0.5678559              1             0         1
##   Disciplinary_failure V20
## 1                    0   0
## 2                    1   0
## 3                    0   0
## 4                    0   0
## 5                    0   0
## 6                    0   0

set.seed(1876)


for (r in 1:R){
```

```
  # subsetting data to training and testing data
  p <- .6 # proportion of data for training
  w <- sample(1:nrow(dat_smaller), nrow(dat_smaller)*p, replace=F)
  data_train <-dat_smaller[w,]
  data_test <- dat_smaller[-w,]

  ################################################################ knn

  #Running the classifier

  knn <- knn(data_train[,1:9],
             test = data_test[,1:9],
             cl=data_train[,20], k=1)

  #predict doesn't work with KNN for factors
  knntable <- table(knn, data_test[,20])

  #generate confusion matrix ( the 1 tells the model we care about that
output)
  cm_KNN <-  confusionMatrix(data = knntable, reference = data_test[,1:2],
positive = "1")

  KNNcm [[r,1]] <-  cm_KNN$table[1,1]
  KNNcm [[r,2]] <-  cm_KNN$table[1,2]
  KNNcm [[r,3]] <-  cm_KNN$table[2,1]
  KNNcm [[r,4]] <-  cm_KNN$table[2,2]

  err_matrix_opt [[r,1]] <-  (cm_KNN$table[1,2]+cm_KNN$table[2,1])/nrow(
data_test)

  # store the errors (change the 1 to whichever model you have)

  sensitivity_matrix_opt[[r, 1]] <- cm_KNN$byClass[1]

  fmeasure_matrix_opt [[r, 1]] <- cm_KNN$byClass[7]

  gmean_matrix_opt [[r, 1]] <- sqrt(cm_KNN$byClass[1]* cm_KNN$byClass[2])

  #cat("Finished Rep",r, "\n")
}
colnames(sensitivity_matrix_opt)<- "KNN"
graph_sens <- melt(sensitivity_matrix_opt)

graph <- ggplot(graph_sens,aes(x=Var2, y=value) )+ geom_boxplot()
graph
```




```
colnames(err_matrix_opt)<- "KNN"
graph_err <- melt(err_matrix_opt)

graph <- ggplot(graph_err,aes(x=Var2, y=value) )+ geom_boxplot()
graph
```



Using SMOTE to optimize

```
set.seed(1876)

dat <- read_excel("Absenteeism_at_work.xls")
col <- c("ID", "Reason for absence", "Month of absence", "Day of the week",
"Seasons", "Disciplinary failure", "Education", "Social drinker",   "Social
smoker")
dat[col] <- lapply(dat[col], as.factor)
colnames(dat) <- c("ID", "Reason", "Month", "Day", "Seasons",
"Transportation_expense", "Distance", "Service_time", "Age", "Work_load",
"Hit_target", "Disciplinary_failure", "Education", "Children",
"Social_drinker", "Social_smoker", "Pet", "Weight", "Height", "BMI",
"Absent_time")


nums <- unlist(lapply(dat, is.numeric))
dat.num <- dat[ , nums]

#change variable represent missed time one day or greater
dat <- dat %>% mutate(Absent_time= ifelse(dat$Absent_time <=8,0,1))
str(dat)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    740 obs. of  21 variables:
##  $ ID                    : Factor w/ 36 levels "1","2","3","4",..: 11 36 3
7 11 3 10 20 14 1 ...
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 26 1 23
8 23 23 22 23 20 22 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 8 8 8 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 2 2 3 4
4 5 5 5 1 1 ...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
##  $ Transportation_expense: num  289 118 179 279 289 179 361 260 155 235
...
##  $ Distance              : num  36 13 51 5 36 51 52 50 12 11 ...
##  $ Service_time          : num  13 18 18 14 13 18 3 11 14 14 ...
##  $ Age                   : num  33 50 38 39 33 38 28 36 34 37 ...
##  $ Work_load             : num  239554 239554 239554 239554 239554 ...
##  $ Hit_target            : num  97 97 97 97 97 97 97 97 97 97 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 3 ...
##  $ Children              : num  2 1 0 2 2 0 1 4 2 1 ...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1
...
##  $ Pet                   : num  1 0 0 0 1 0 4 0 0 1 ...
##  $ Weight                : num  90 98 89 68 90 89 80 65 95 88 ...
##  $ Height                : num  172 178 170 168 172 170 172 168 196 172
...
##  $ BMI                   : num  30 31 31 24 30 31 27 23 25 29 ...
##  $ Absent_time           : num  0 0 0 0 0 0 0 0 1 0 ...
```

```
dat$Absent_time <- as.factor(dat$Absent_time)
#Transforming to Data Frame
dat <- as.data.frame(dat)

str(dat)
```

```
## 'data.frame':    740 obs. of  21 variables:
##  $ ID                    : Factor w/ 36 levels "1","2","3","4",..: 11 36 3
7 11 3 10 20 14 1 ...
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 26 1 23
8 23 23 22 23 20 22 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 8 8 8 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 2 2 3 4
4 5 5 5 1 1 ...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
```

```
1 1 1 1 ...
##  $ Transportation_expense: num  289 118 179 279 289 179 361 260 155 235
...
##  $ Distance              : num  36 13 51 5 36 51 52 50 12 11 ...
##  $ Service_time          : num  13 18 18 14 13 18 3 11 14 14 ...
##  $ Age                   : num  33 50 38 39 33 38 28 36 34 37 ...
##  $ Work_load             : num  239554 239554 239554 239554 239554 ...
##  $ Hit_target            : num  97 97 97 97 97 97 97 97 97 97 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 3 ...
##  $ Children              : num  2 1 0 2 2 0 1 4 2 1 ...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1
...
##  $ Pet                   : num  1 0 0 0 1 0 4 0 0 1 ...
##  $ Weight                : num  90 98 89 68 90 89 80 65 95 88 ...
##  $ Height                : num  172 178 170 168 172 170 172 168 196 172
...
##  $ BMI                   : num  30 31 31 24 30 31 27 23 25 29 ...
##  $ Absent_time           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1
...

###Optimizing the KNN

#For the tunning of the KNN model, we are going to create another
traning/test data sets.

#scaling the data:
dat_v <- dat #we are going to use dat_v for the manipulation
scale <- sapply(dat_v, is.numeric)
dat_v[scale] <- lapply(dat_v[scale],scale)
head(dat_v)

##   ID Reason Month Day Seasons Transportation_expense   Distance
## 1 11     26     7   3       1              1.0107248  0.4292653
## 2 36      0     7   3       1             -1.5433353 -1.1209354
## 3  3     23     7   4       1             -0.6322379  1.4402658
## 4  7      7     7   5       1              0.8613645 -1.6601356
## 5 11     23     7   5       1              1.0107248  0.4292653
## 6  3     23     7   6       1             -0.6322379  1.4402658
##   Service_time        Age  Work_load Hit_target Disciplinary_failure
## 1    0.1017010 -0.5325083 -0.8176594  0.6382541                    0
## 2    1.2419848  2.0914456 -0.8176594  0.6382541                    1
## 3    1.2419848  0.2392429 -0.8176594  0.6382541                    0
## 4    0.3297577  0.3935931 -0.8176594  0.6382541                    0
## 5    0.1017010 -0.5325083 -0.8176594  0.6382541                    0
## 6    1.2419848  0.2392429 -0.8176594  0.6382541                    0
```

```
##   Education    Children Social_drinker Social_smoker        Pet     Weight
## 1         1  0.89311870              1             0  0.1927195  0.8510972
## 2         1 -0.01722267              1             0 -0.5658572  1.4720605
## 3         1 -0.92756405              1             0 -0.5658572  0.7734768
## 4         1  0.89311870              1             1 -0.5658572 -0.8565516
## 5         1  0.89311870              1             0  0.1927195  0.8510972
## 6         1 -0.92756405              1             0 -0.5658572  0.7734768
##        Height        BMI Absent_time
## 1 -0.01903313  0.7754078           0
## 2  0.97516826  1.0087554           0
## 3 -0.35043360  1.0087554           0
## 4 -0.68183407 -0.6246778           0
## 5 -0.01903313  0.7754078           0
## 6 -0.35043360  1.0087554           0
```

```
#predicting class:
AB_class <- dat_v[, 21]
names(AB_class) <- c(1:nrow(dat_v))
dat_v$ID <- c(1:nrow(dat_v))

dat_v <- dat_v[1:737,]
nrow(dat_v)
```

```
## [1] 737
```

```
rand_permute <- sample(x = nrow(dat_v), size = nrow(dat_v))

all_id_random <- dat_v[rand_permute, "ID"]
dat_v <- dat_v[,-1] #remove ID

#######

splitIndex <- createDataPartition(dat_v$Absent_time, p = .50,
                                  list = FALSE,
                                  times = 1)

trainSplit <- dat_v[ splitIndex,]
testSplit <- dat_v[-splitIndex,]

trainSplit$Absent_time <- as.factor(trainSplit$Absent_time)
trainSplit <- SMOTE(Absent_time ~ ., trainSplit, perc.over = 100,
perc.under=200)

prop.table(table(trainSplit$Absent_time))
```

```
##
##   0   1
## 0.5 0.5
```

```
#######

#labels to make inserted code work
validate_id <- c(1:nrow(testSplit))
training_id <- c(1:nrow(trainSplit))

#rename to work with rest of code
dat_v_train <- trainSplit
dat_v_val <- testSplit
AB_class_train <- trainSplit$Absent_time
AB_class_val <- testSplit$Absent_time
#Confirms data comes out as expected
table(AB_class_train)

## AB_class_train
##  0  1
## 64 64

#Study significance of the variables

rf <- randomForest(Absent_time ~.,
                   data=dat_v_train,
                   mtry=6,
                   ntree=50,
                   na.action=na.roughfix)

impfact <- importance(rf)

impfact <- as.list(impfact)
names(impfact) <- colnames(dat_v[,-20])
impfact2 <- unlist(impfact)


most_sig_stats <- names(sort(desc(impfact2)))

#As per 'most_sig_stats' the 5 most significant variables for the prediction
are:
#'Seasons', 'Reason', 'Service_time', 'Month' and 'work_load'

#Re ordering variables by significance:

dat_v_train_ord <- dat_v_train[ c(most_sig_stats)]
str(dat_v_train_ord)

## 'data.frame':    128 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 28 23 1
8 23 23 14 8 19 7 ...
##  $ Work_load             : num [1:128, 1] -0.0761 -0.1657 -0.1657 -0.8663
-1.6789 ...
##   ..- attr(*, "dimnames")=List of 2
```

```
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 12 11
11 6 9 11 7 8 13 8 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 3 1 5 1
1 5 2 1 1 1 ...
##  $ Hit_target            : num [1:128, 1] -0.42 -1.743 -1.743 1.167 -0.685
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Distance              : num [1:128, 1] -1.323 -0.649 -0.649 1.508 1.508
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Height                : num [1:128, 1] -0.019 -0.848 -0.848 -0.019 -
0.019 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Weight                : num [1:128, 1] 0.3078 2.093 2.093 0.0749 0.0749
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Age                   : num [1:128, 1] 0.0849 1.011 1.011 -1.3043 -
1.3043 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Transportation_expense: num [1:128, 1] -1.543 0.204 0.204 2.086 2.086
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Service_time          : num [1:128, 1] -0.582 0.102 0.102 -2.179 -2.179
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Pet                   : num [1:128, 1] -0.566 -0.566 -0.566 2.468 2.468
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ BMI                   : num [1:128, 1] 0.3087 2.6422 2.6422 0.0754
0.0754 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
```

```
##   .. ..$ : NULL
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 4 4 4 3 1 4
1 1 4 1 ...
##  $ Children              : num [1:128, 1] -0.9276 -0.0172 -0.0172 -0.0172
-0.0172 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
3 2 2 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 1
...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1 2 1
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
```

```
dat_v_val_ord <- dat_v_val[, names(dat_v_train_ord)]
str(dat_v_val_ord)
```

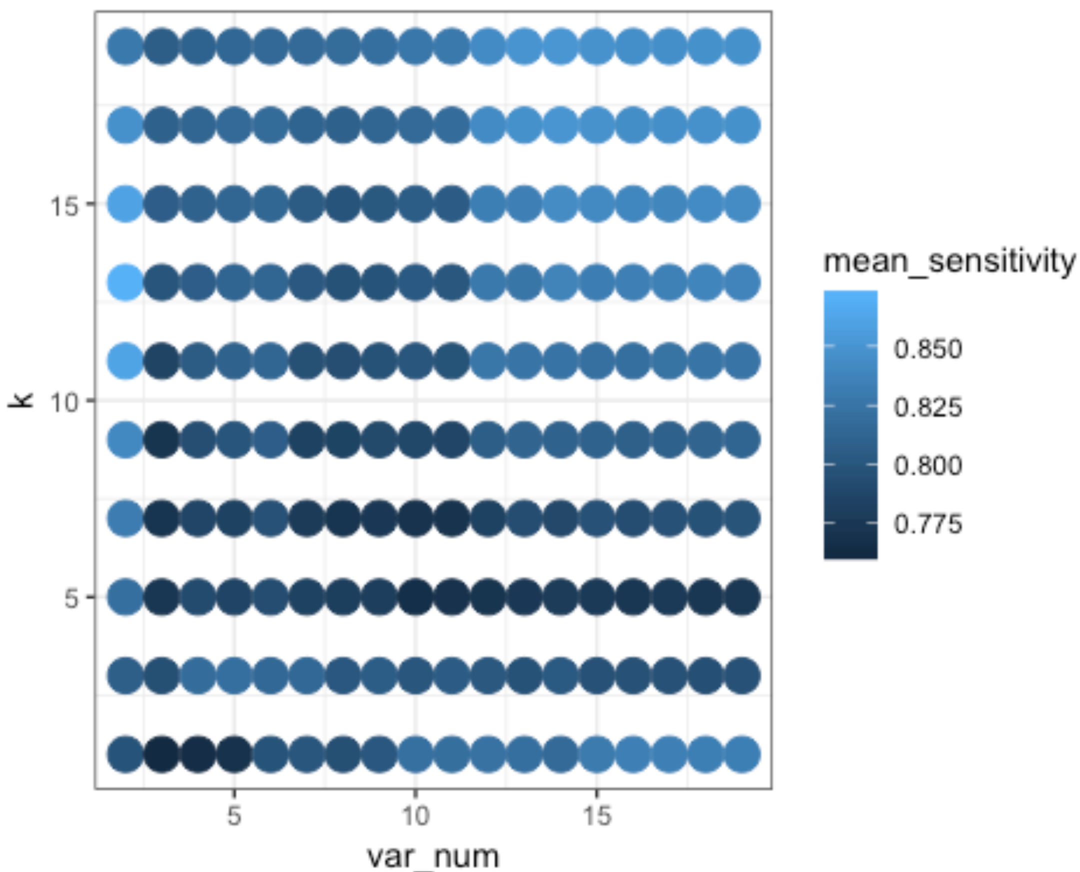
```
## 'data.frame':    368 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 26 1 22
23 20 22 2 11 19 28 ...
##  $ Work_load             : num [1:368, 1] -0.818 -0.818 -0.818 -0.818 -
0.818 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 9 9 9 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 2 2 5 5
1 1 2 3 1 3 ...
##  $ Hit_target            : num [1:368, 1] 0.638 0.638 0.638 0.638 0.638
...
##  $ Distance              : num [1:368, 1] 0.429 -1.121 1.508 1.373 -1.188
...
##  $ Height                : num [1:368, 1] -0.019 0.975 -0.019 -0.682 3.958
...
##  $ Weight                : num [1:368, 1] 0.8511 1.4721 0.0749 -1.0894
1.2392 ...
##  $ Age                   : num [1:368, 1] -0.5325 2.0914 -1.3043 -0.0695 -
0.3782 ...
##  $ Transportation_expense: num [1:368, 1] 1.011 -1.543 2.086 0.578 -0.991
...
##  $ Service_time          : num [1:368, 1] 0.102 1.242 -2.179 -0.354 0.33
...
##  $ Pet                   : num [1:368, 1] 0.193 -0.566 2.468 -0.566 -0.566
...
##  $ BMI                   : num [1:368, 1] 0.7754 1.0088 0.0754 -0.858 -
0.3913 ...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
##  $ Children              : num [1:368, 1] 0.8931 -0.0172 -0.0172 2.7138
```

```
0.8931 ...
##  $ Education            : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 3
1 2 1 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 2 1 2 2
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2
...

#######################
#######################

#Monte Carlo Validation:

size <- nrow(dat_v_train)
sub <- (2/3) * nrow(dat_v_train)

training_family_L <- lapply(1:500, function(j) {
  perm <- sample(1:size, size = size, replace = F)
  shuffle <- training_id[perm]
  trn <- shuffle[1:sub]
  trn
})

validation_family_L <- lapply(training_family_L,
                              function(x) setdiff(training_id, x))

#Finding an optimal set of variables and optimal k

N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 9.219544

K <- seq(from = 1, to = 19, by = 2)
times <- 500 * length(N) * length(K)

#Execution of the test with loops

paramter_errors_df <- data.frame(mc_index = as.integer(rep(NA, times =
times)),
                                 var_num = as.integer(rep(NA, times =
times)),
                                 k = as.integer(rep(NA, times = times)),
                                 error = as.numeric(rep(NA, times = times)))

#Core knn_model:
# j = index, n = length of range of variables, k=k
#core_knn <- function(j, n, k) {
```

```
 # knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
  #                   test = dat_v_train_ord[validation_family_L[[j]], 1:n],
   #                  cl = AB_class_train[training_family_L[[j]]],
    #                 k = k)
#  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])
 # err <- (tbl[1, 2] + tbl[2, 1])/(tbl[1, 2] + tbl[2, 1]+tbl[1, 1] + tbl[2,
2])
  #err}


param_df1 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df <- merge(param_df1, data.frame(k = K))

#knn_err_est_df <- ddply(param_df[1:times, ], .(mc_index, var_num, k),
function(df) {
 # err <- core_knn(df$mc_index[1], df$var_num[1], df$k[1])
  #err
#})

#head(knn_err_est_df)
#names(knn_err_est_df)[4] <- "error"

#mean_errs_df <- ddply(knn_err_est_df, .(var_num, k), function(df)
mean(df$error))
#head(mean_errs_df)
#names(mean_errs_df)[3] <- "mean_error"

#ggplot(data = mean_errs_df, aes(x = var_num, y = k, color = mean_error)) +
geom_point(size = 5) + theme_bw()

#mean_errs_df[which.min(mean_errs_df$mean_error), ]

#mean_errs_df %>% arrange(mean_error)



#eventually run old to compare with new.
#We see that although error lower, other metrics hurt. We care about
identifying >8 hours  so modify

#Repeat with sensitivity

N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 9.219544

K <- seq(from = 1, to = 19, by = 2)
times <- 500 * length(N) * length(K)
```

```
core_knn_sen <- function(j, n, k) {
  knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
                     test = dat_v_train_ord[validation_family_L[[j]], 1:n],
                     cl = AB_class_train[training_family_L[[j]]],
                     k = k)

  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])

  #generate confusion matrix ( the 1 tells the model we care about that
output)
  #cm_KNN <-  confusionMatrix(data = tbl, reference
=AB_class_train[validation_family_L[[j]]], positive = "1")

  sen <- (tbl[2, 2] )/(tbl[1, 2] + tbl[2, 2])
  sen
}


param_df1_2 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df_2 <- merge(param_df1_2, data.frame(k = K))

knn_err_est_df_2 <- ddply(param_df_2[1:times, ], .(mc_index, var_num, k),
function(df) {
  sen <- core_knn_sen(df$mc_index[1], df$var_num[1], df$k[1])
  sen
})

head(knn_err_est_df_2)

##   mc_index var_num  k        V1
## 1        1       2  1 0.8571429
## 2        1       2  3 0.8571429
## 3        1       2  5 0.8571429
## 4        1       2  7 0.8571429
## 5        1       2  9 0.9047619
## 6        1       2 11 0.9047619

names(knn_err_est_df_2)[4] <- "Sensitivity"

mean_sens_df <- ddply(knn_err_est_df_2, .(var_num, k), function(df)
mean(df$Sensitivity))
head(mean_sens_df)

##   var_num  k        V1
## 1       2  1 0.7978376
## 2       2  3 0.8077695
## 3       2  5 0.8213372
## 4       2  7 0.8307269
```

```
## 5        2  9 0.8405706
## 6        2 11 0.8610326

names(mean_sens_df)[3] <- "mean_sensitivity"

ggplot(data = mean_sens_df, aes(x = var_num, y = k, color =
mean_sensitivity)) + geom_point(size = 5) +
  theme_bw()
```




```
mean_sens_df[which.max(mean_sens_df$mean_sensitivity), ]

##   var_num  k mean_sensitivity
## 7       2 13        0.8706989

order <- mean_sens_df %>% arrange(desc(mean_sensitivity))

save(mean_sens_df, file='mean_sens_df.RData')

R <- 100 # replications


# create the matrix to store values 1 row per model
err_matrix_opt <- matrix(0, ncol=5, nrow=R)
```

```
sensitivity_matrix_opt <- matrix(0, ncol=5, nrow=R)

fmeasure_matrix_opt <- matrix(0, ncol=5, nrow=R)

gmean_matrix_opt <- matrix(0, ncol=5, nrow=R)

# these are optional but I like to see how the model did each run so I can
check other output
KNNcm <- matrix(0, ncol=4, nrow=R)
KNNcm2 <- matrix(0, ncol=4, nrow=R)
KNNcm3 <- matrix(0, ncol=4, nrow=R)
KNNcm4 <- matrix(0, ncol=4, nrow=R)
KNNcm5 <- matrix(0, ncol=4, nrow=R)

set.seed(1876)

for (r in 1:R){

  # subsetting data to training and testing data
  splitIndex <- createDataPartition(dat_v$Absent_time, p = .50,
                                    list = FALSE,
                                    times = 1)

  trainSplit <- dat_v[ splitIndex,]
  testSplit <- dat_v[-splitIndex,]

  trainSplit$Absent_time <- as.factor(trainSplit$Absent_time)
  trainSplit <- SMOTE(Absent_time ~ ., trainSplit, perc.over = 100,
perc.under=200)
  ################################################################ knn

  #Running the classifier

  #option 1

  knn <- knn(trainSplit[,1:order[1,1]],
             test = testSplit[,1:order[1,1]],
             cl=trainSplit[,20], k=order[1,2])

  #predict doesn't work with KNN for factors
  knntable <- table(knn, testSplit[,20])

  cm_KNN <-  confusionMatrix(data = knntable, reference = testSplit[,20],
positive = "1")

  KNNcm [[r,1]] <-  cm_KNN$table[1,1]
  KNNcm [[r,2]] <-  cm_KNN$table[1,2]
  KNNcm [[r,3]] <-  cm_KNN$table[2,1]
  KNNcm [[r,4]] <-  cm_KNN$table[2,2]
```

```
  err_matrix_opt [[r,1]] <-
(cm_KNN$table[1,2]+cm_KNN$table[2,1])/nrow(testSplit)

  # store the errors

  sensitivity_matrix_opt[[r, 1]] <- cm_KNN$byClass[1]

  fmeasure_matrix_opt [[r, 1]] <- cm_KNN$byClass[7]

  gmean_matrix_opt [[r, 1]] <- sqrt(cm_KNN$byClass[1]* cm_KNN$byClass[2])

  ######################
  #option 2

  knn <- knn(trainSplit[,1:order[2,1]],
             test = testSplit[,1:order[2,1]],
             cl=trainSplit[,20], k=order[2,2])

  #predict doesn't work with KNN for factors
  knntable2 <- table(knn, testSplit[,20])

  cm_KNN2 <-  confusionMatrix(data = knntable2, reference = testSplit[,20],
positive = "1")

  KNNcm2 [[r,1]] <-  cm_KNN2$table[1,1]
  KNNcm2 [[r,2]] <-  cm_KNN2$table[1,2]
  KNNcm2 [[r,3]] <-  cm_KNN2$table[2,1]
  KNNcm2 [[r,4]] <-  cm_KNN2$table[2,2]

  err_matrix_opt [[r,2]] <-
(cm_KNN2$table[1,2]+cm_KNN2$table[2,1])/nrow(testSplit)

  sensitivity_matrix_opt[[r, 2]] <- cm_KNN2$byClass[1]

  fmeasure_matrix_opt [[r, 2]] <- cm_KNN2$byClass[7]

  gmean_matrix_opt [[r, 2]] <- sqrt(cm_KNN2$byClass[1]* cm_KNN2$byClass[2])

  ##########
  #option 3

  knn <- knn(trainSplit[,1:order[3,1]],
             test = testSplit[,1:order[3,1]],
             cl=trainSplit[,20], k=order[3,2])

  #predict doesn't work with KNN for factors
  knntable <- table(knn, testSplit[,20])
```

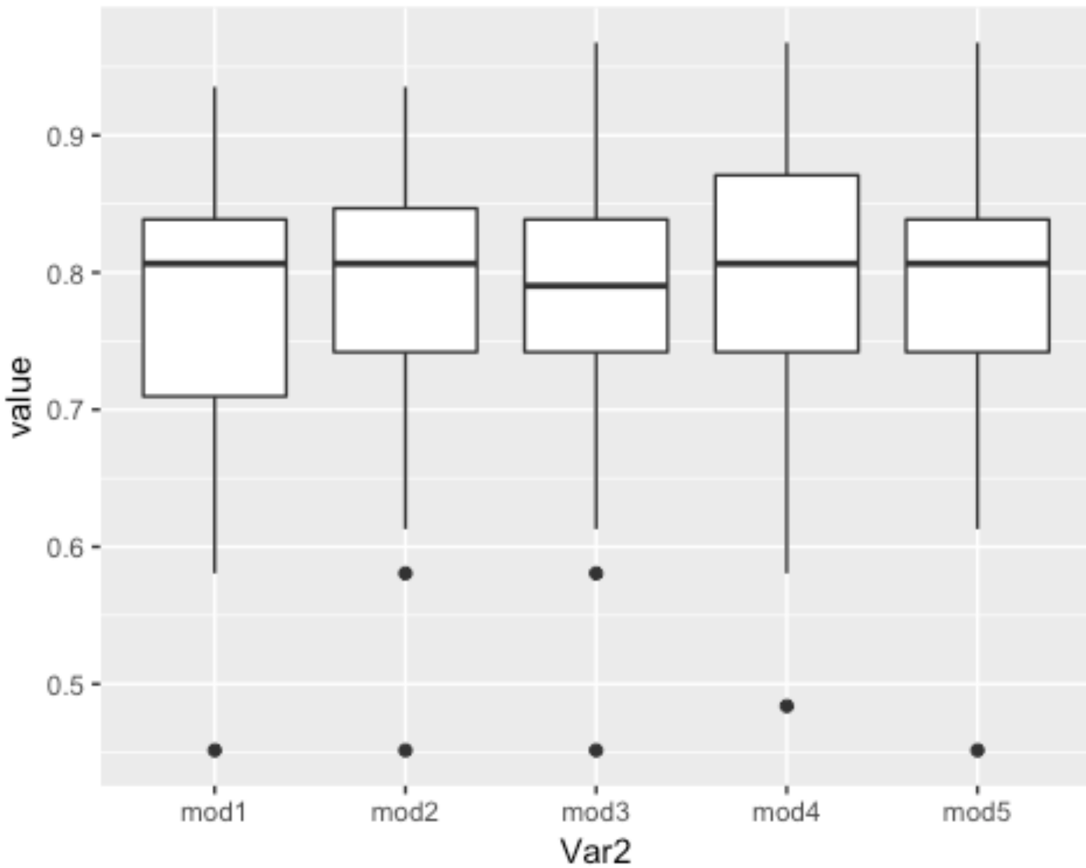
```
  cm_KNN3 <-  confusionMatrix(data = knntable, reference = testSplit[,20],
positive = "1")

  KNNcm3 [[r,1]] <-  cm_KNN3$table[1,1]
  KNNcm3 [[r,2]] <-  cm_KNN3$table[1,2]
  KNNcm3 [[r,3]] <-  cm_KNN3$table[2,1]
  KNNcm3 [[r,4]] <-  cm_KNN3$table[2,2]

  err_matrix_opt [[r,3]] <-
(cm_KNN3$table[1,2]+cm_KNN3$table[2,1])/nrow(testSplit)

  sensitivity_matrix_opt[[r, 3]] <- cm_KNN3$byClass[1]

  fmeasure_matrix_opt [[r, 3]] <- cm_KNN3$byClass[7]

  gmean_matrix_opt [[r, 3]] <- sqrt(cm_KNN3$byClass[1]* cm_KNN3$byClass[2])

  ################
  #option 4

  knn <- knn(trainSplit[,1:order[4,1]],
             test = testSplit[,1:order[4,1]],
             cl=trainSplit[,20], k=order[4,2])

  #predict doesn't work with KNN for factors
  knntable4 <- table(knn, testSplit[,20])

  cm_KNN4 <-  confusionMatrix(data = knntable4, reference = testSplit[,20],
positive = "1")

  KNNcm4 [[r,1]] <-  cm_KNN4$table[1,1]
  KNNcm4 [[r,2]] <-  cm_KNN4$table[1,2]
  KNNcm4 [[r,3]] <-  cm_KNN4$table[2,1]
  KNNcm4 [[r,4]] <-  cm_KNN4$table[2,2]

  err_matrix_opt [[r,4]] <-
(cm_KNN4$table[1,2]+cm_KNN4$table[2,1])/nrow(testSplit)

  # store the errors

  sensitivity_matrix_opt[[r, 4]] <- cm_KNN4$byClass[1]

  fmeasure_matrix_opt [[r, 4]] <- cm_KNN4$byClass[7]

  gmean_matrix_opt [[r, 4]] <- sqrt(cm_KNN4$byClass[1]* cm_KNN4$byClass[2])

  #####################
```

```
  #option 5

  knn <- knn(trainSplit[,1:order[5,1]],
             test = testSplit[,1:order[5,1]],
             cl=trainSplit[,20], k=order[5,2])

  knntable5 <- table(knn, testSplit[,20])

  cm_KNN5 <-  confusionMatrix(data = knntable5, reference = testSplit[,20],
positive = "1")

  KNNcm5 [[r,1]] <-  cm_KNN5$table[1,1]
  KNNcm5 [[r,2]] <-  cm_KNN5$table[1,2]
  KNNcm5 [[r,3]] <-  cm_KNN5$table[2,1]
  KNNcm5 [[r,4]] <-  cm_KNN5$table[2,2]

  err_matrix_opt [[r,5]] <-  (cm_KNN5$table[1,2]+cm_KNN5$table[2,1])/nrow(
testSplit)

  # store the errors

  sensitivity_matrix_opt[[r, 5]] <- cm_KNN5$byClass[1]

  fmeasure_matrix_opt [[r, 5]] <- cm_KNN5$byClass[7]

  gmean_matrix_opt [[r, 5]] <- sqrt(cm_KNN5$byClass[1]* cm_KNN5$byClass[2])

  cat("Finished Rep",r, "\n")
}

## Finished Rep 1
## Finished Rep 2
## Finished Rep 3
## Finished Rep 4
## Finished Rep 5
## Finished Rep 6
## Finished Rep 7
## Finished Rep 8
## Finished Rep 9
## Finished Rep 10
## Finished Rep 11
## Finished Rep 12
## Finished Rep 13
## Finished Rep 14
## Finished Rep 15
## Finished Rep 16
## Finished Rep 17
## Finished Rep 18
## Finished Rep 19
```

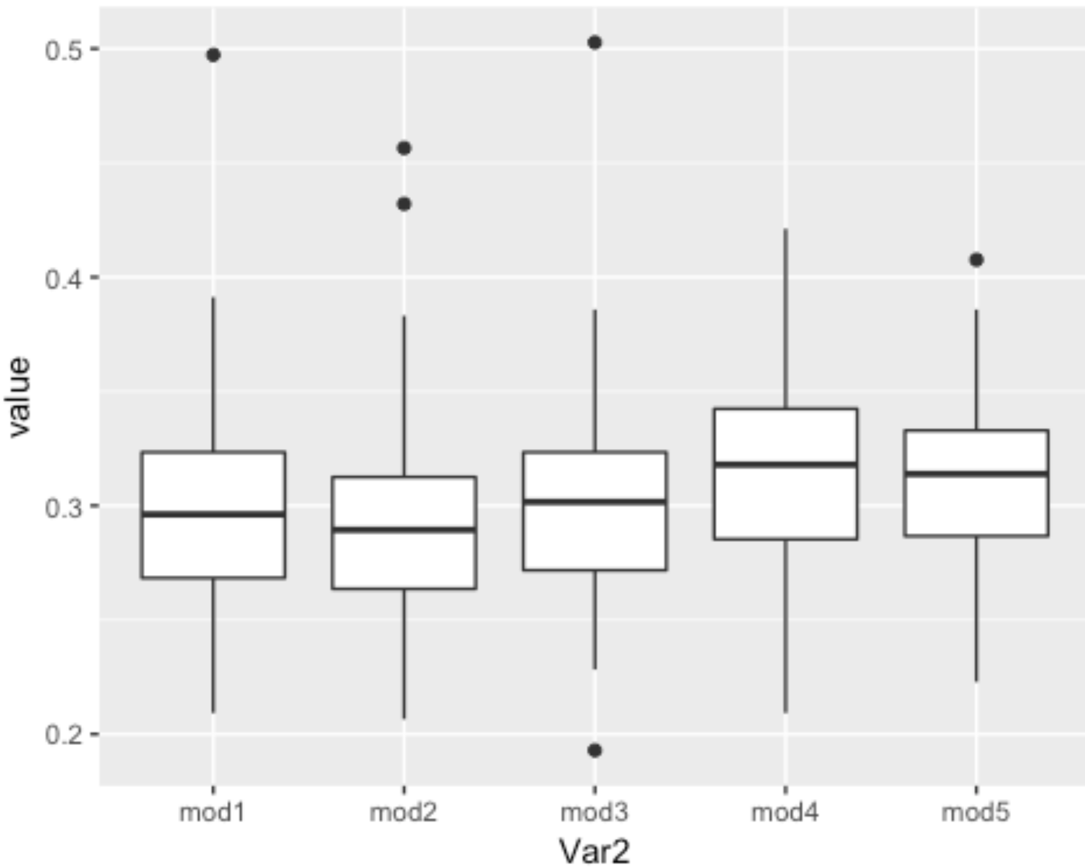
```
## Finished Rep 20
## Finished Rep 21
## Finished Rep 22
## Finished Rep 23
## Finished Rep 24
## Finished Rep 25
## Finished Rep 26
## Finished Rep 27
## Finished Rep 28
## Finished Rep 29
## Finished Rep 30
## Finished Rep 31
## Finished Rep 32
## Finished Rep 33
## Finished Rep 34
## Finished Rep 35
## Finished Rep 36
## Finished Rep 37
## Finished Rep 38
## Finished Rep 39
## Finished Rep 40
## Finished Rep 41
## Finished Rep 42
## Finished Rep 43
## Finished Rep 44
## Finished Rep 45
## Finished Rep 46
## Finished Rep 47
## Finished Rep 48
## Finished Rep 49
## Finished Rep 50
## Finished Rep 51
## Finished Rep 52
## Finished Rep 53
## Finished Rep 54
## Finished Rep 55
## Finished Rep 56
## Finished Rep 57
## Finished Rep 58
## Finished Rep 59
## Finished Rep 60
## Finished Rep 61
## Finished Rep 62
## Finished Rep 63
## Finished Rep 64
## Finished Rep 65
## Finished Rep 66
## Finished Rep 67
## Finished Rep 68
## Finished Rep 69
```

```
## Finished Rep 70
## Finished Rep 71
## Finished Rep 72
## Finished Rep 73
## Finished Rep 74
## Finished Rep 75
## Finished Rep 76
## Finished Rep 77
## Finished Rep 78
## Finished Rep 79
## Finished Rep 80
## Finished Rep 81
## Finished Rep 82
## Finished Rep 83
## Finished Rep 84
## Finished Rep 85
## Finished Rep 86
## Finished Rep 87
## Finished Rep 88
## Finished Rep 89
## Finished Rep 90
## Finished Rep 91
## Finished Rep 92
## Finished Rep 93
## Finished Rep 94
## Finished Rep 95
## Finished Rep 96
## Finished Rep 97
## Finished Rep 98
## Finished Rep 99
## Finished Rep 100
```

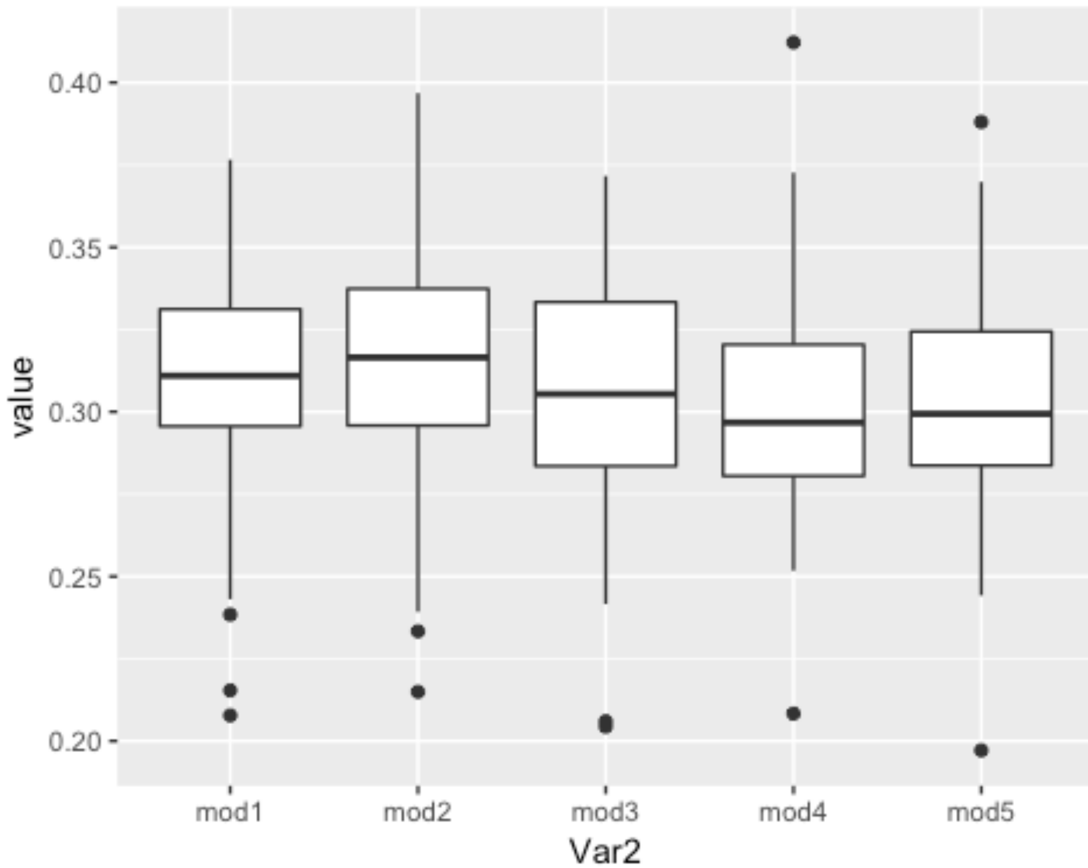
```
colnames(sensitivity_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_sens <- melt(sensitivity_matrix_opt)

ggplot(graph_sens,aes(x=Var2, y=value) )+ geom_boxplot()
```



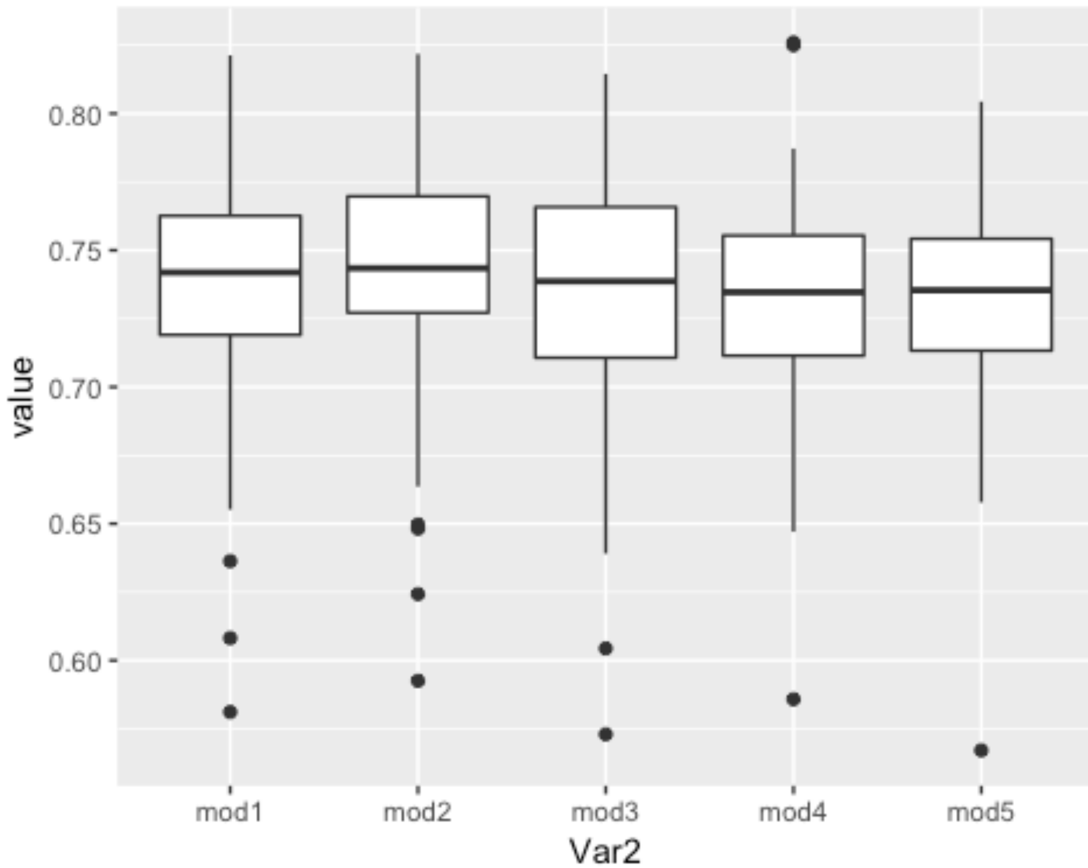

```
colnames(err_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_err <- melt(err_matrix_opt)

ggplot(graph_err,aes(x=Var2, y=value) )+ geom_boxplot()
```




```r
colnames(fmeasure_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_fmeasure <- melt(fmeasure_matrix_opt)

ggplot(graph_fmeasure,aes(x=Var2, y=value) )+ geom_boxplot()
```

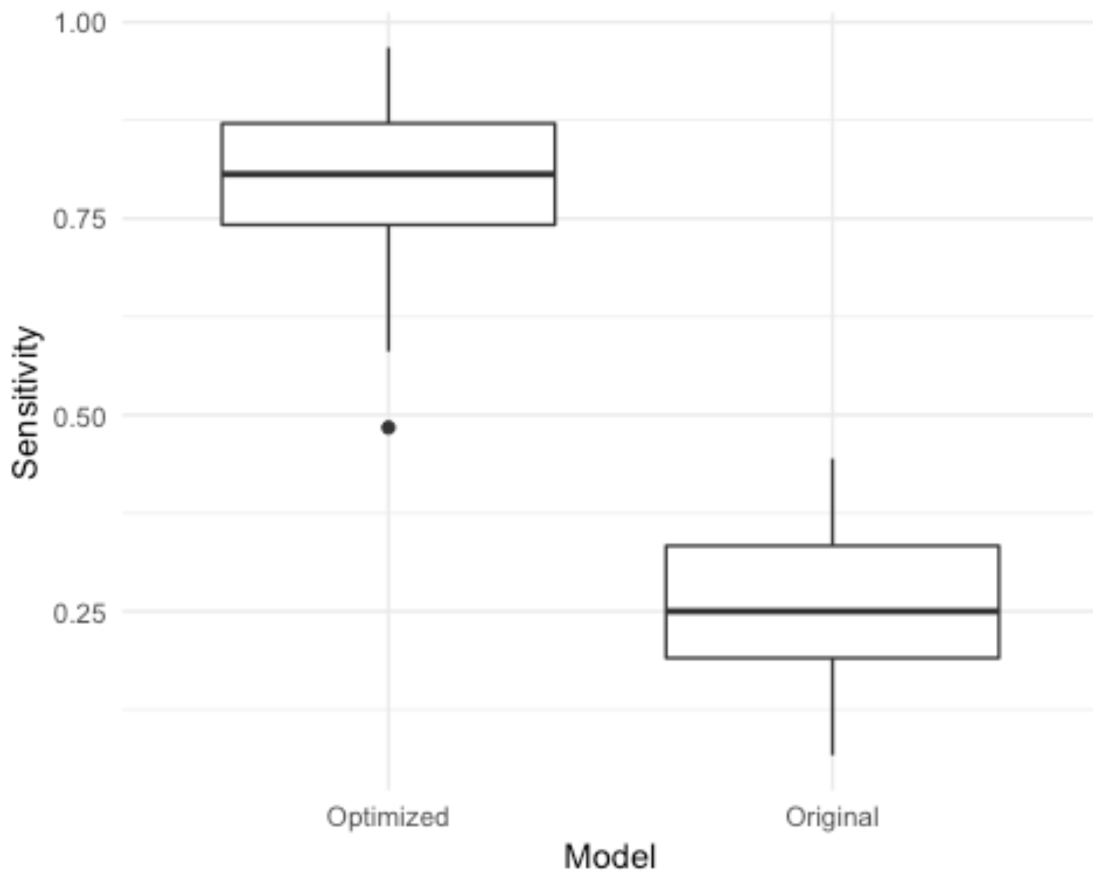



```
colnames(gmean_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_gmean <- melt(gmean_matrix_opt)

ggplot(graph_gmean,aes(x=Var2, y=value) )+ geom_boxplot()
```



Comparison to original model

```
comp_matrix_sens <- cbind(sensitivity_matrix_opt[,4], sensitivity_matrix[,1])

colnames(comp_matrix_sens)<- c("Optimized","Original")

graph_comparison <- melt(comp_matrix_sens)

ggplot(graph_comparison,aes(x=Var2, y=value) )+ geom_boxplot() +labs(x=
"Model", y= "Sensitivity") +
  theme_minimal()
```




```
set.seed(1876)
  splitIndex <- createDataPartition(dat_v$Absent_time, p = .50,
                                    list = FALSE,
                                    times = 1)

  trainSplit <- dat_v[ splitIndex,]
  testSplit <- dat_v[-splitIndex,]

  trainSplit$Absent_time <- as.factor(trainSplit$Absent_time)
  trainSplit <- SMOTE(Absent_time ~ ., trainSplit, perc.over = 100,
perc.under=200)

knn <- knn(trainSplit[,1:order[4,1]],
             test = testSplit[,1:order[4,1]],
             cl=trainSplit[,20], k=order[4,2])
  knntable4 <- table(knn, testSplit[,20])

  cm_KNN4 <-  confusionMatrix(data = knntable4, reference = testSplit[,20],
positive = "1")

cm_KNN4
```

```
## Confusion Matrix and Statistics
##
##
## knn   0   1
##   0 264   4
##   1  73  27
##
##                Accuracy : 0.7908
##                  95% CI : (0.7456, 0.8312)
##     No Information Rate : 0.9158
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3255
##  Mcnemar's Test P-Value : 9.239e-15
##
##             Sensitivity : 0.87097
##             Specificity : 0.78338
##          Pos Pred Value : 0.27000
##          Neg Pred Value : 0.98507
##              Prevalence : 0.08424
##          Detection Rate : 0.07337
##    Detection Prevalence : 0.27174
##       Balanced Accuracy : 0.82718
##
##        'Positive' Class : 1
##

set.seed(1876)
dat1 <- dat[-1]

#scale
scale <- sapply(dat1, is.numeric)
dat1[scale] <- lapply(dat1[scale],scale)
p <- .6 # proportion of data for training
w <- sample(1:nrow(dat1), nrow(dat1)*p, replace=F)
data_train <-dat1[w,]
data_test <- dat1[-w,]

  ################################################################ knn

#Running the classifier

  knn <- knn(data_train[-20],
                       test = data_test[-20],
                       cl=data_train$Absent_time, k=2)

#predict doesn't work with KNN for factors
 knntable <- table(knn, data_test$Absent_time)
```

```r
#generate confusion matrix ( the 1 tells the model we care about that output)
 cm_KNN <-  confusionMatrix(data = knntable, reference = data_test[,-20],
positive = "1")

cm_KNN
```

```
## Confusion Matrix and Statistics
##
##
## knn   0   1
##   0 253  16
##   1  19   8
##
##                Accuracy : 0.8818
##                  95% CI : (0.8394, 0.9162)
##     No Information Rate : 0.9189
##     P-Value [Acc > NIR] : 0.9900
##
##                   Kappa : 0.2493
##  Mcnemar's Test P-Value : 0.7353
##
##             Sensitivity : 0.33333
##             Specificity : 0.93015
##          Pos Pred Value : 0.29630
##          Neg Pred Value : 0.94052
##              Prevalence : 0.08108
##          Detection Rate : 0.02703
##    Detection Prevalence : 0.09122
##       Balanced Accuracy : 0.63174
##
##        'Positive' Class : 1
##
```