Predictive Modeling of COVID-19 Outcomes

Shubham Kamboj (DSTC-22/23-014)

2023-05-29

Reading CSV file from System

```
 #rm(list= ls())

start_time <- Sys.time()

library(Boruta)
library(mlbench)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caTools)
library(e1071)
library(ggplot2)
```

```
data<- read.csv("C:/Users/USER/OneDrive - students.sau.ac.in/Desktop/Rfiles/University Project on Covid

# str(data)
```

factoring variables which are multicategorical

```
columns_to_factor <- c("sex","coronaseve","obesity","htn","dm","cad",
                       "copd","ckd","cva","lvdys","asthma","copd1","tb",
                       "malignancy","typeofmali","liverillne","fever",
                       "cough","dyspnoea","diarrhea","nauseavom","gitcomp",
                       "respirator","anemia","throbocyto","lft","sgot","sgpt",
                       "aki","abgacidos","ddimer","serumfibri","sferrtin",
                       "il6","lqtinter","sinustachy","arrhythmia","supreavent",
                       "pac","mat","at","af0","afl","psvt","ventricula", "vpc",
                       "nonsusvt"   ,"sustvt","vf","bradyarrhy","firstdegav",
                       "seconavblo", "thirdavblo" ,"oxygenrequ","cxrpneu",
                       "patchprsen","patchlobel", "pleuraleff", "oxygenreq1",
                       "ventilator", "noninvasie", "invasive" ,  "arrythmia" ,
                       "icushift",   "death", "lmwh","ramdesevir" ,"steroid" ,
                       "sitetb"  , "sarsseveri" , "cadtype", "patchside" ,
                       "lmwhdoseus", "ntprobnp", "tropt" , "sars",
                       "coviddeath", "noncovidde" , "procovidde", "discharge")
# colnames(data)


for(i in columns_to_factor){
  data[,i] <- factor(data[,i])
}

#summary(data)
```

counts of NA's

Count of missing values patient wise

```
data_sno <- cbind(sno = 1:nrow(data) , data)


# loop to count the missing values for each row

p_na_count<-c() # this is empty vector which will store count of NA's
for (i in 1:nrow(data_sno)){
  x<-0
  for(j in 1:ncol(data_sno)){
    if(is.na(data_sno[i,j]) == T){
      x <- x + 1
    }
    p_na_count[i] <- x
  }
}

na_count_p <- cbind(pt_sno = data_sno[,1] , total_na = p_na_count)
NA_count_p <- na_count_p[ order(-p_na_count) ,  ] # ordering of p_na_count
head(NA_count_p , 40)
```

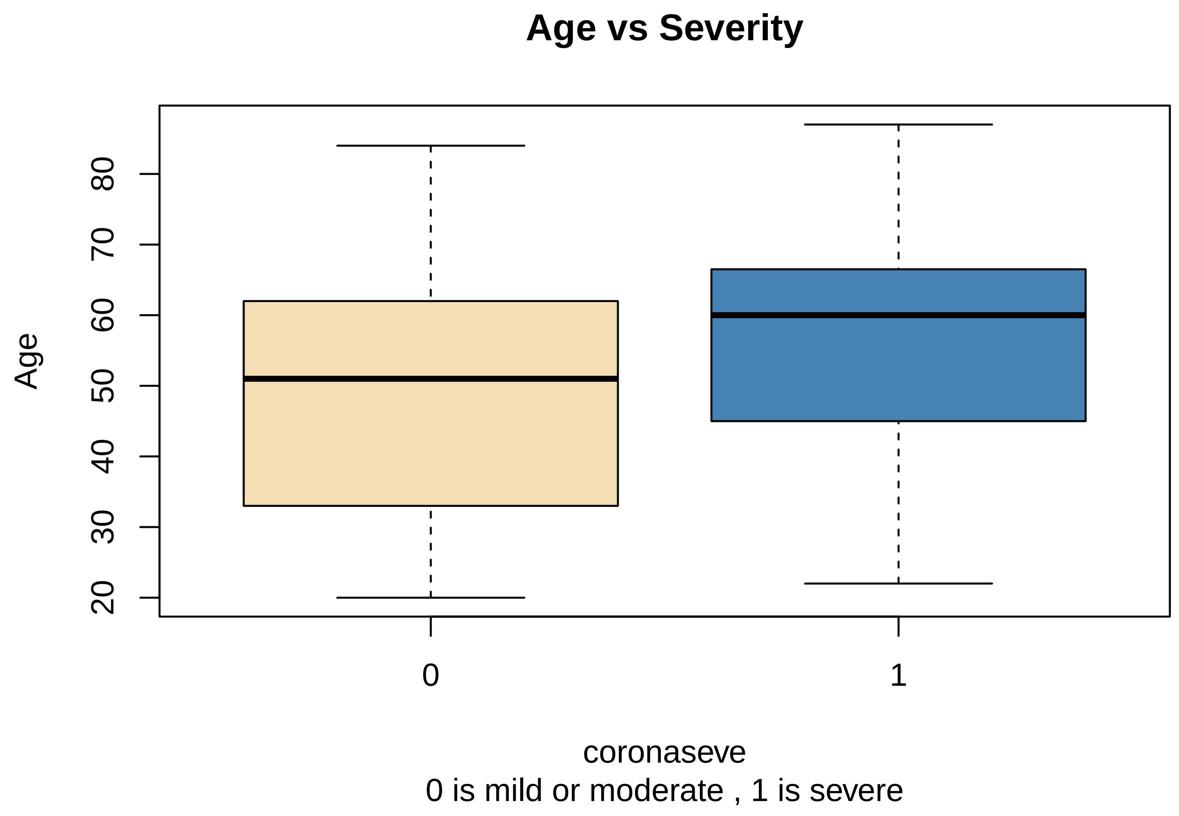
```
##       pt_sno total_na
##  [1,]     40       36
##  [2,]     43       36
##  [3,]     64       31
## [4,]      69       31
## [5,]      26       29
## [6,]      27       29
## [7,]      28       28
## [8,]      13       26
## [9,]       8       24
## [10,]      9       24
## [11,]     23       24
## [12,]     32       24
## [13,]     36       24
## [14,]     39       23
## [15,]     46       23
## [16,]      5       22
## [17,]     11       22
## [18,]     34       22
## [19,]     22       21
## [20,]     37       21
## [21,]      2       20
## [22,]     14       20
## [23,]     17       20
## [24,]     29       20
## [25,]    114       20
## [26,]    271       20
## [27,]     18       19
## [28,]     35       19
## [29,]     44       18
## [30,]    132       18
## [31,]    282       18
## [32,]     45       17
## [33,]     53       17
## [34,]    101       17
## [35,]      7       16
## [36,]     20       16
## [37,]     78       15
## [38,]     48       14
## [39,]     51       14
## [40,]     59       14
```

```
g <- NA_count_p[1:36 , 1] # refer NA_count_p to find number 36
# g contains row no of patient having 15 or more attributes missing


data_lna <- data_sno[-g , ]  # droping those patients from study

#write.csv(data_lna , file = "D:\\machine learning materials\\data_lna.csv" )
data<- data_lna[ , -1] # removing sno from the data set
dim(data)
```

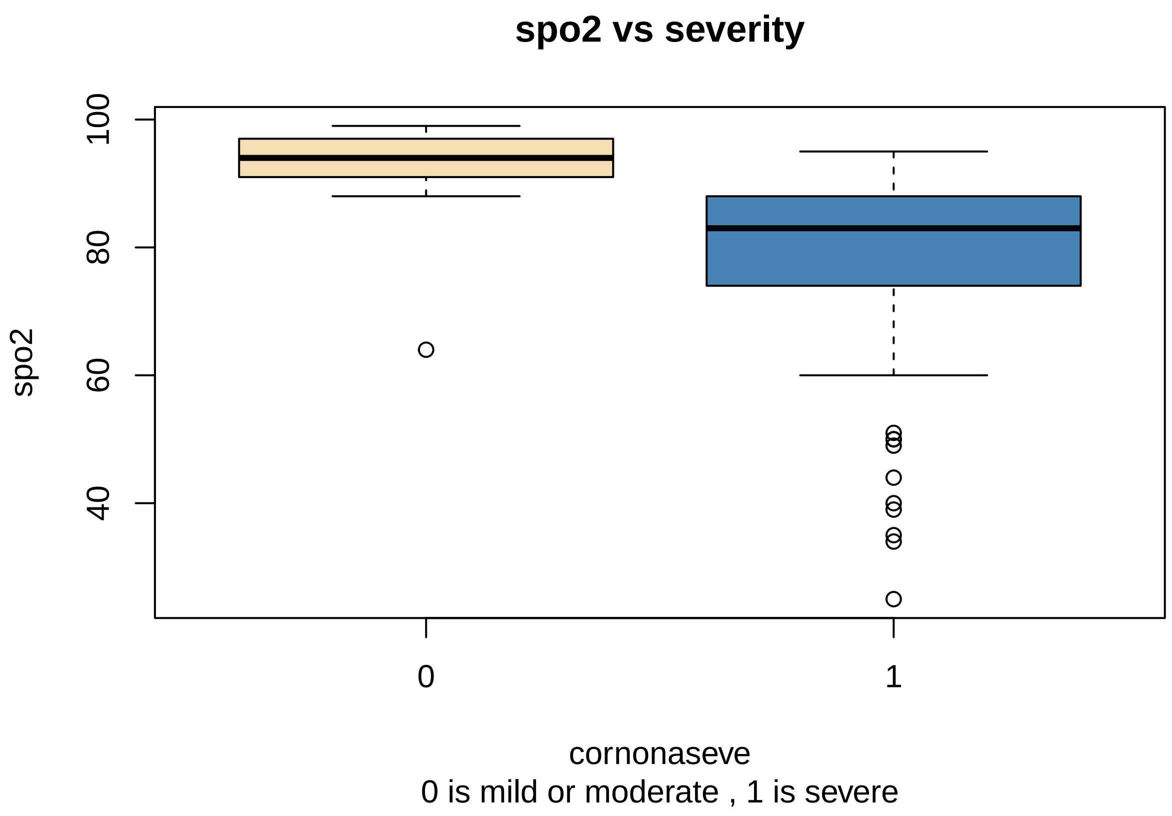
```
## [1] 269 111
```

```
#summary(data)
```

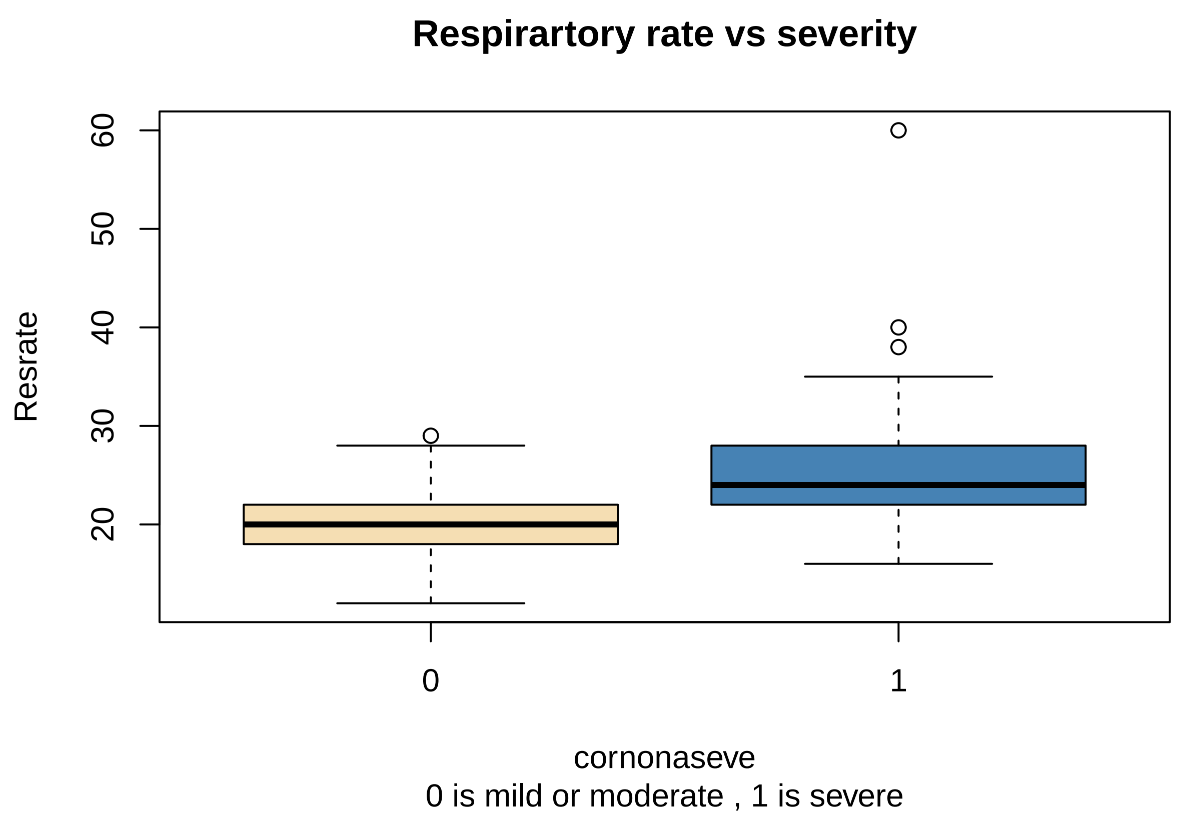
```r
boxplot(data$age ~ data$coronaseve ,
        xlab = "coronaseve" , ylab = "Age" ,
        main = "Age vs Severity" ,
        sub ="0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



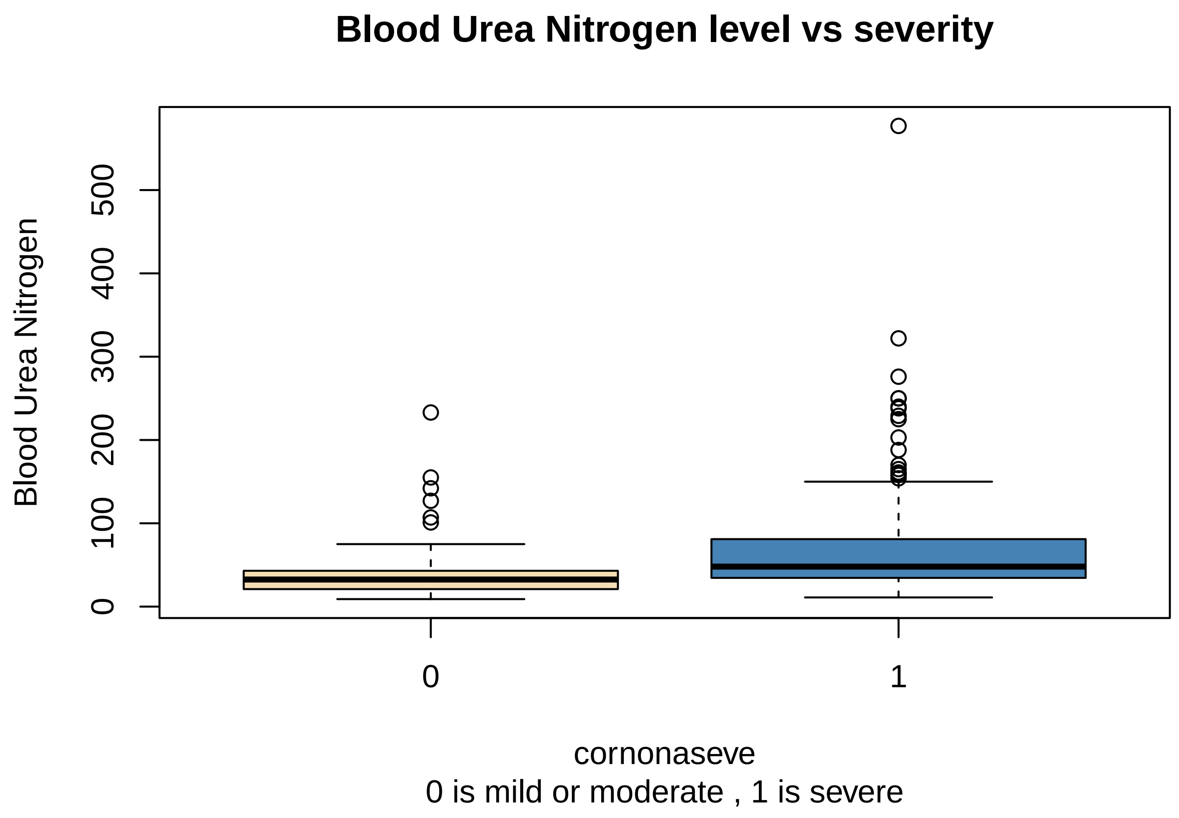
```r
boxplot(data$spo2 ~ data$coronaseve,
        xlab= "cornonaseve" , ylab = "spo2",
        main = "spo2 vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



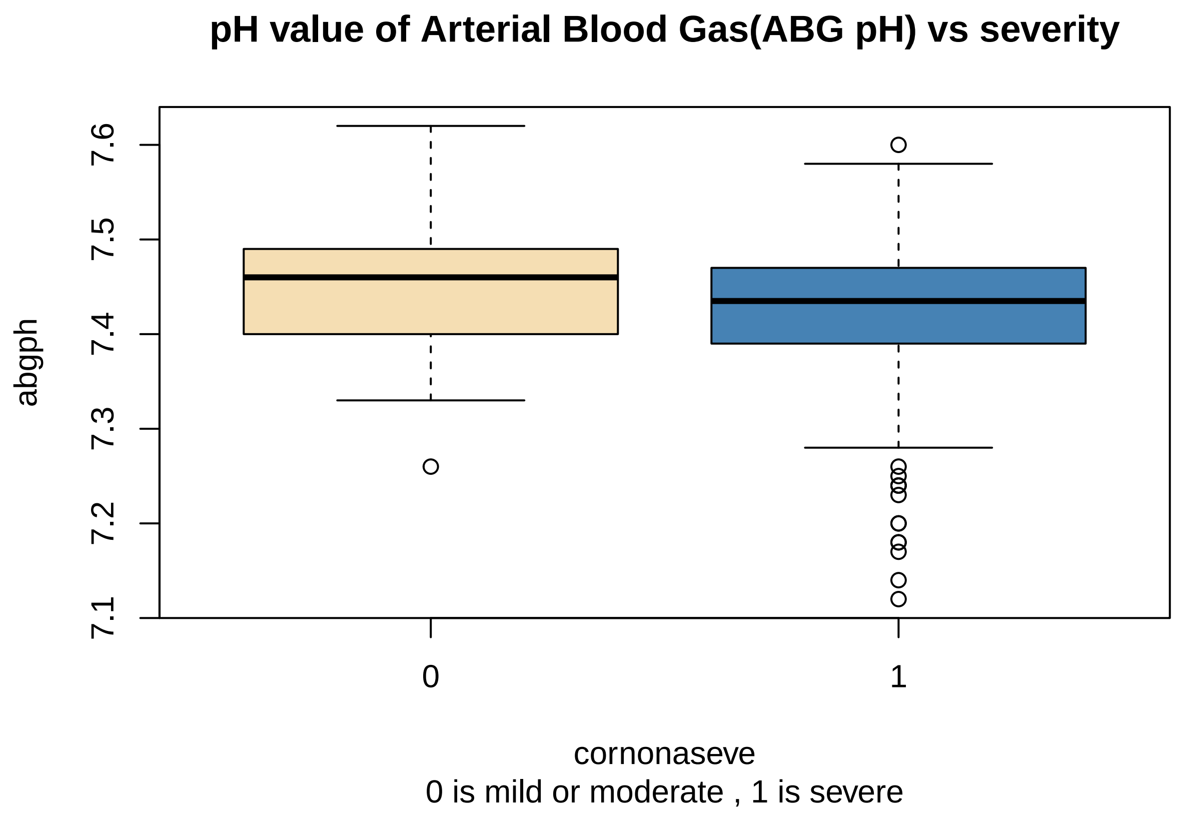
```r
boxplot(data$resrate ~ data$coronaseve,
        xlab= "cornonaseve" , ylab = "Resrate",
        main = "Respirartory rate vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



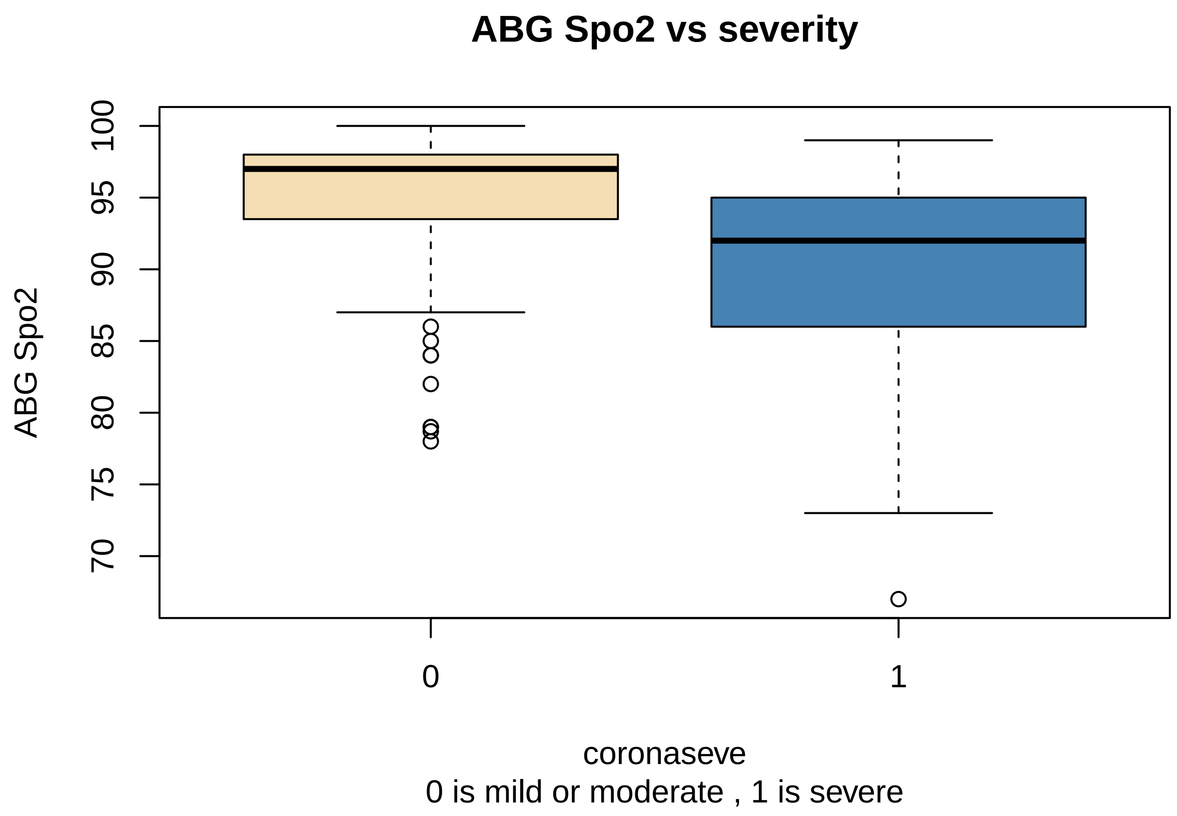
```r
boxplot(data$bun ~ data$coronaseve,
        xlab= "cornonaseve" , ylab = "Blood Urea Nitrogen",
        main = "Blood Urea Nitrogen level vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



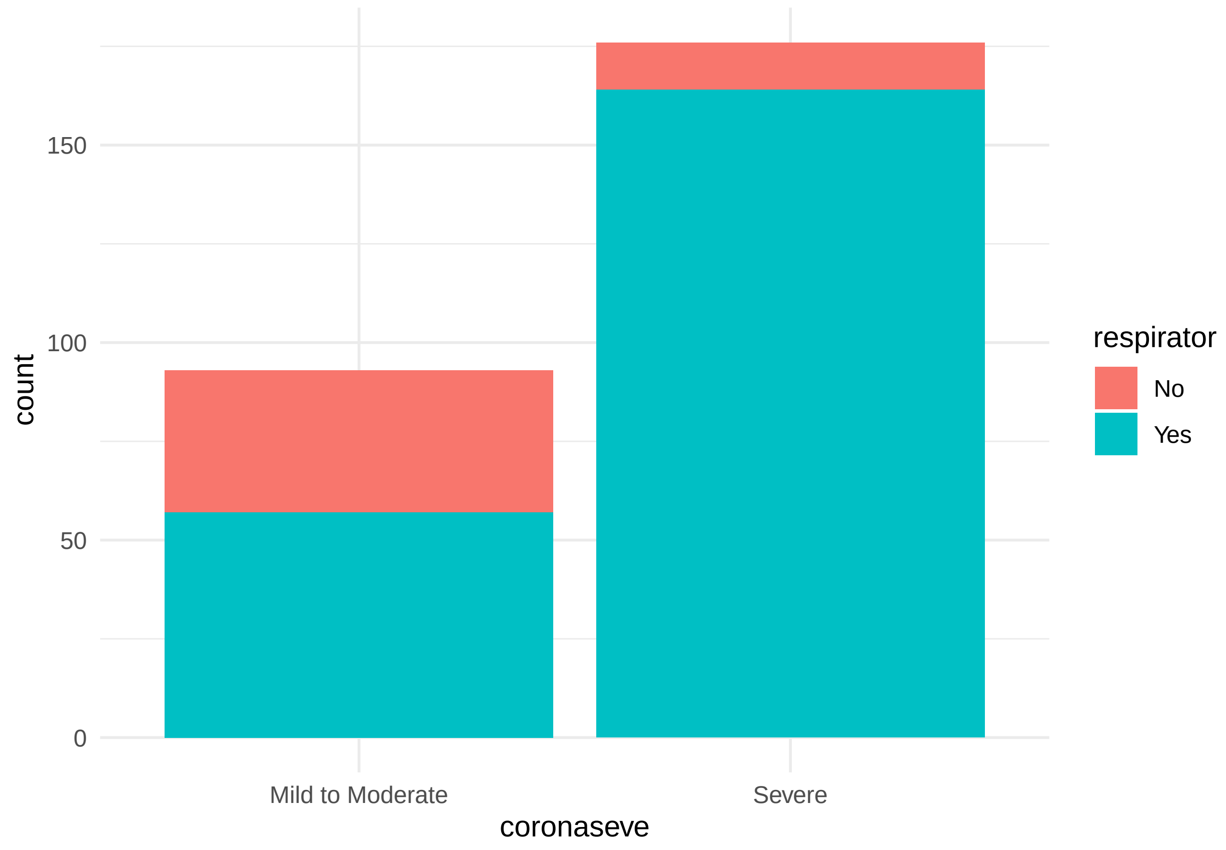
```
boxplot(data$abgph ~ data$coronaseve,
        xlab= "cornonaseve" , ylab = "abgph",
        main = "pH value of Arterial Blood Gas(ABG pH) vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



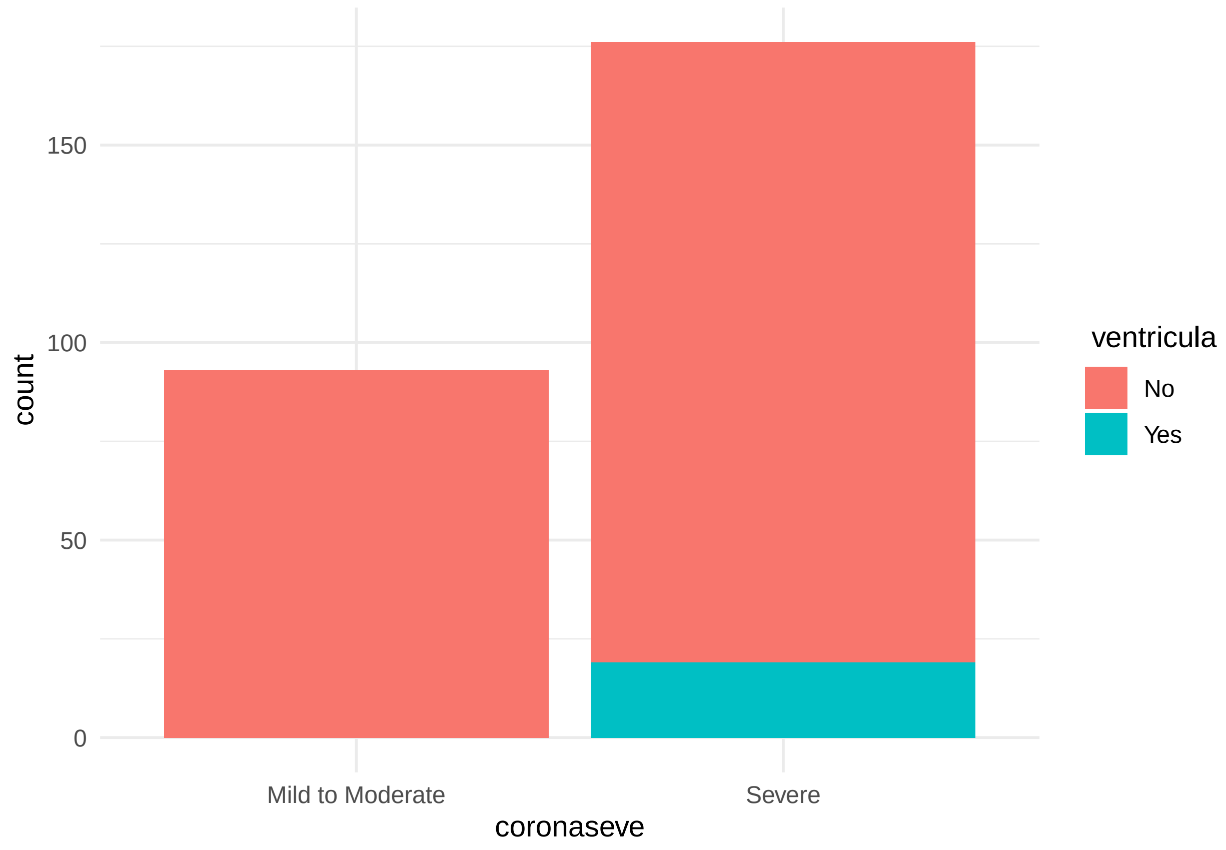
```r
boxplot(data$abgspo2 ~ data$coronaseve,
        xlab= "coronaseve" , ylab = "ABG Spo2",
        main = "ABG Spo2 vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



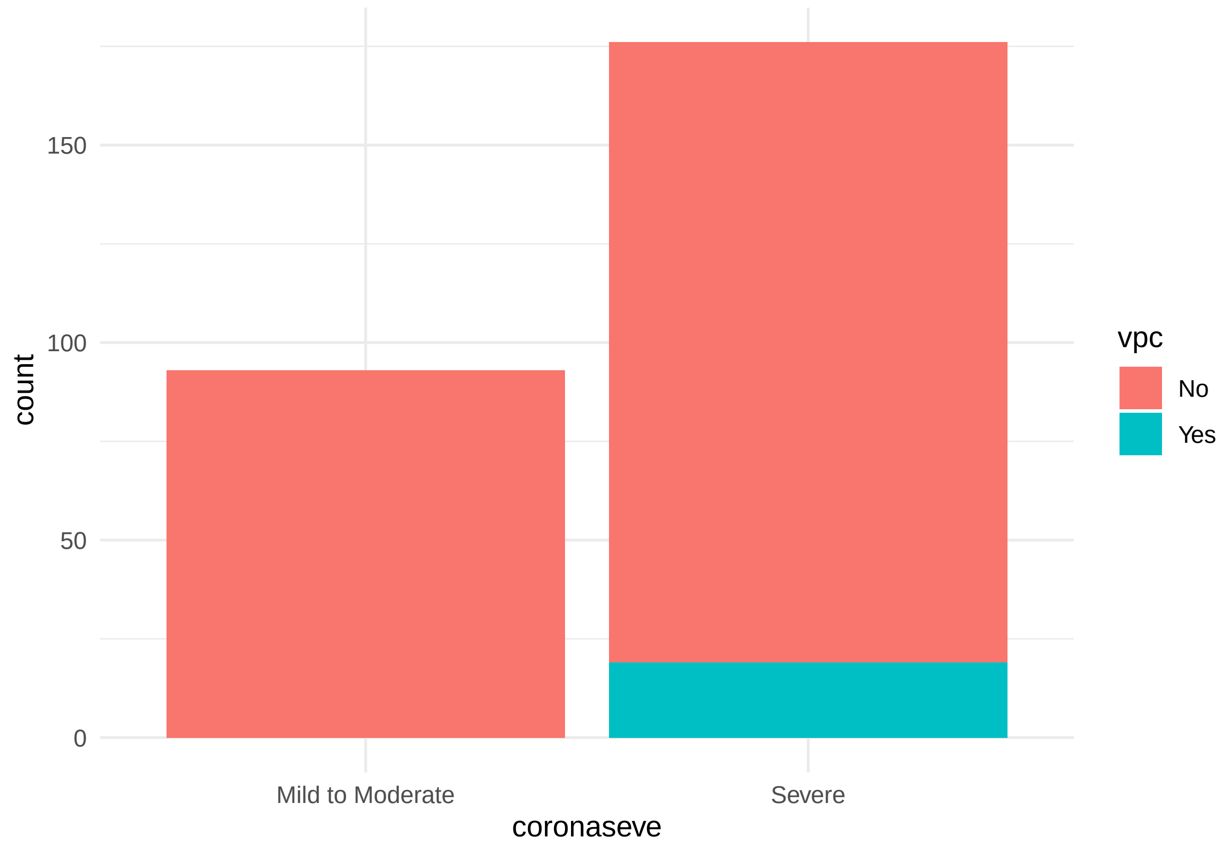
```r
# Create a bar plot using ggplot between coronaseve & respirator
ggplot(data, aes(x = coronaseve, fill =   respirator)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "respirator") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



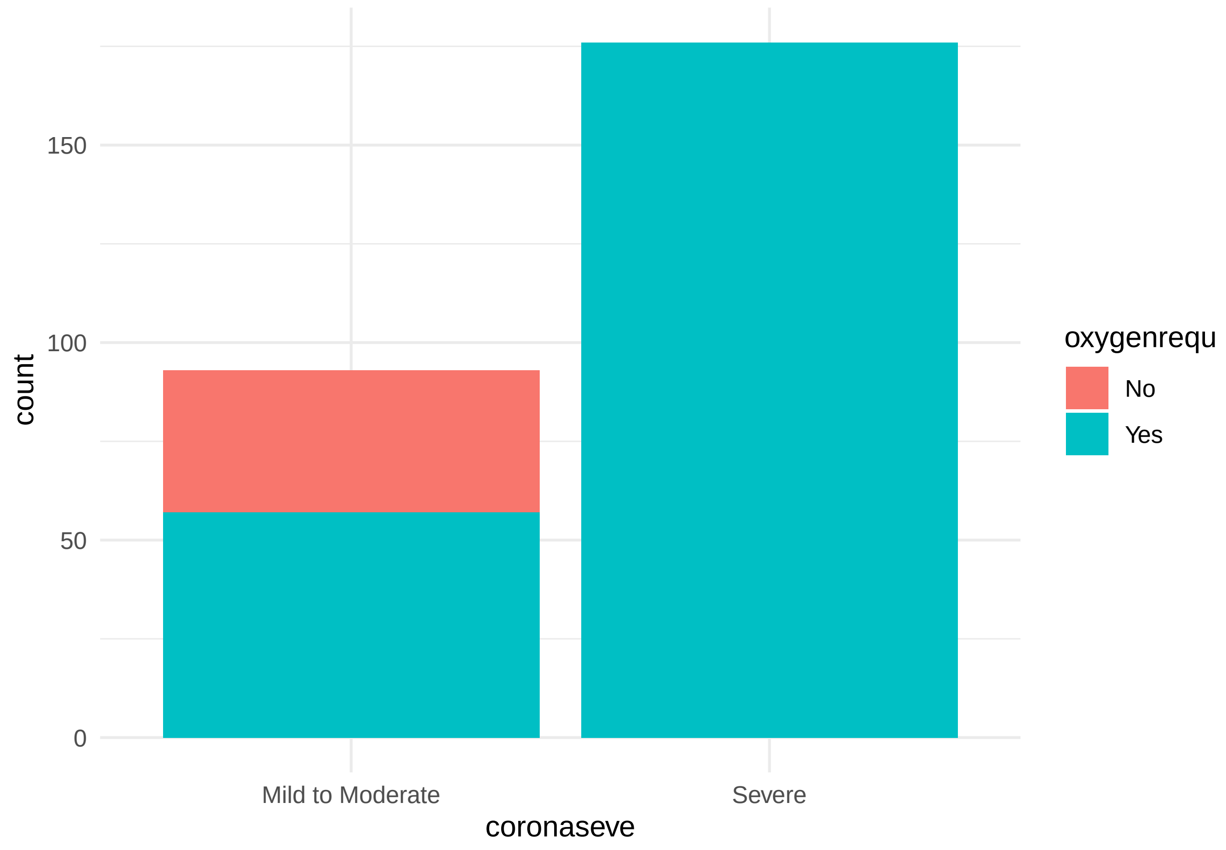
```r
# Create a bar plot using ggplot between coronaseve &  ventricula
ggplot(data, aes(x = coronaseve, fill =    ventricula)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " ventricula") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



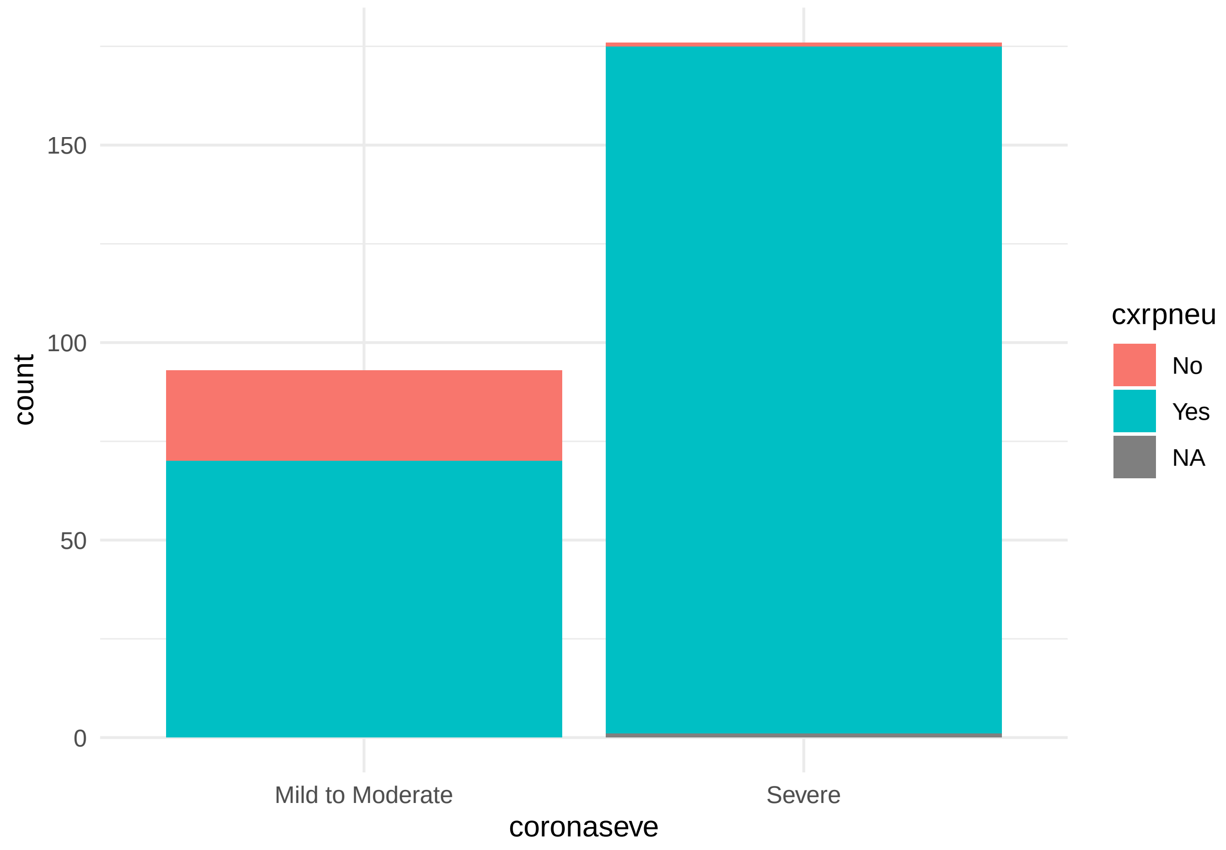
```r
# Create a bar plot using ggplot between coronaseve &  vpc
ggplot(data, aes(x = coronaseve, fill =    vpc)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "vpc") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



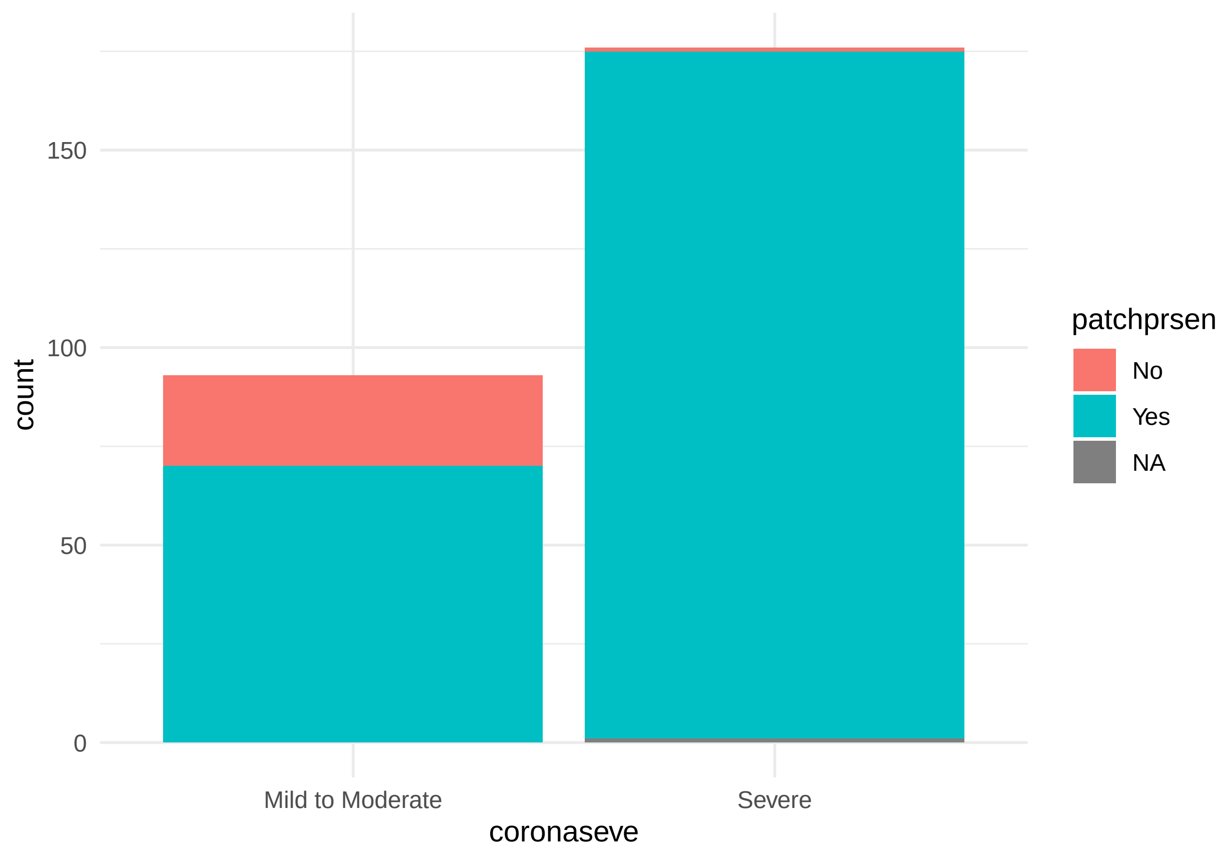
```r
# Create a bar plot using ggplot between coronaseve &  oxygenrequ
ggplot(data, aes(x = coronaseve, fill =    oxygenrequ)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "oxygenrequ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



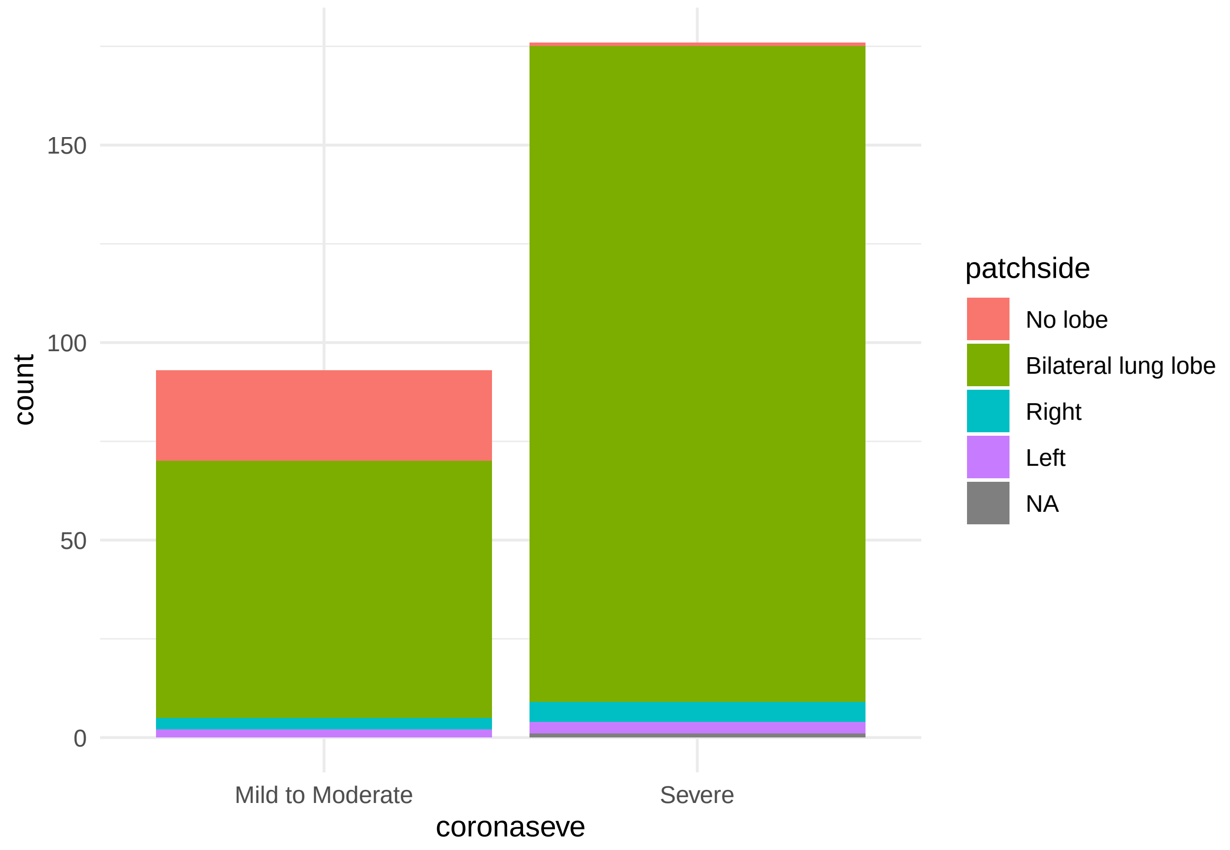
```r
# Create a bar plot using ggplot between coronaseve &  cxrpneu
ggplot(data, aes(x = coronaseve, fill =    cxrpneu)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "cxrpneu") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



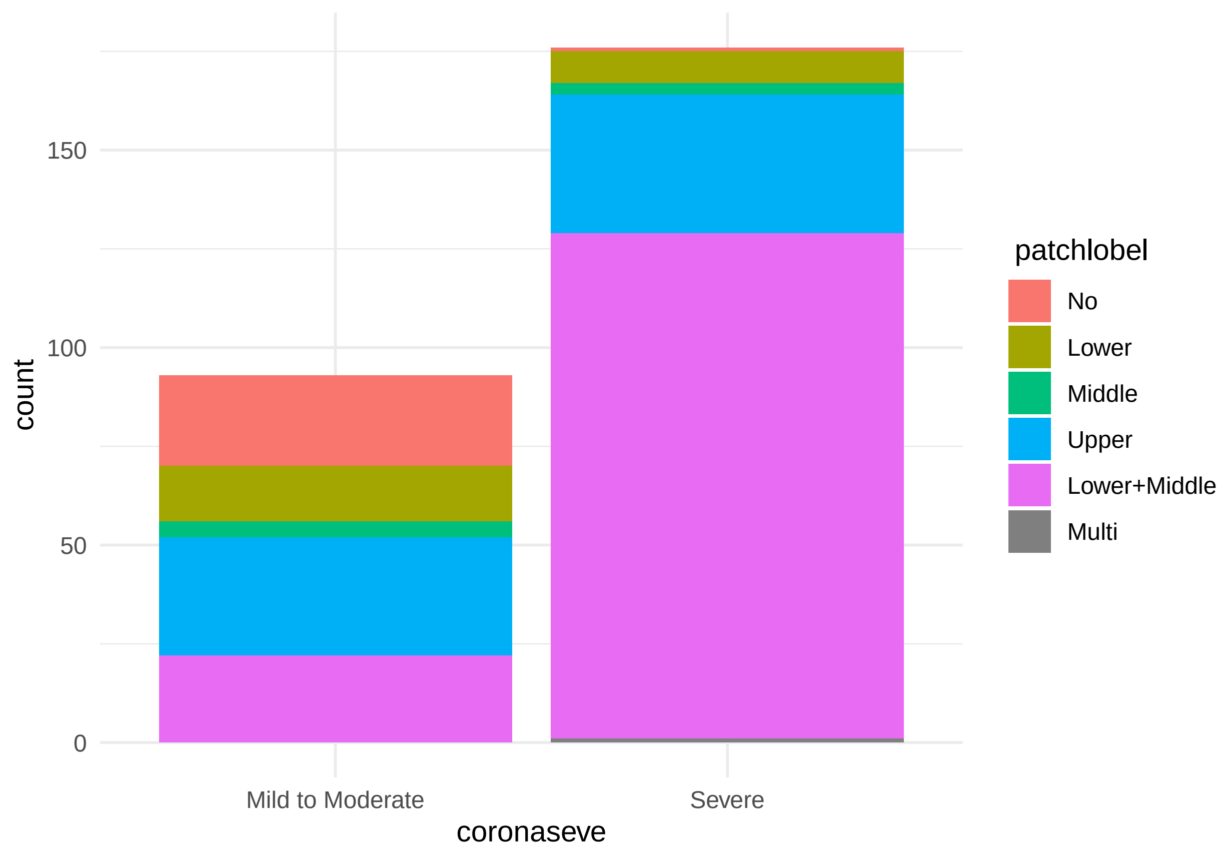
```r
# Create a bar plot using ggplot between coronaseve &  patchprsen
ggplot(data, aes(x = coronaseve, fill =     patchprsen)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "patchprsen") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



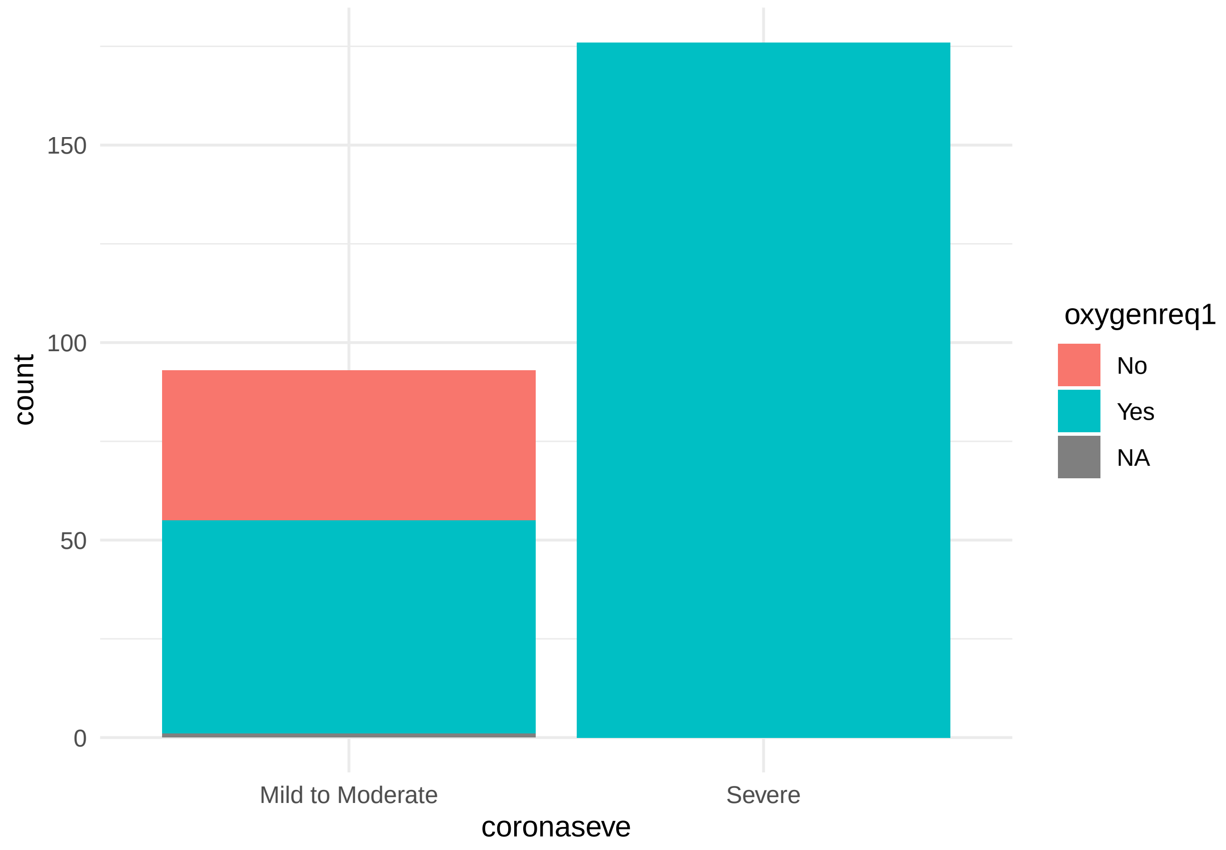
```r
# Create a bar plot using ggplot between coronaseve &  patchside
ggplot(data, aes(x = coronaseve, fill =    patchside)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "patchside") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No lobe", "Bilateral lung lobe",
                                    "Right", "Left")) +
     theme_minimal()
```



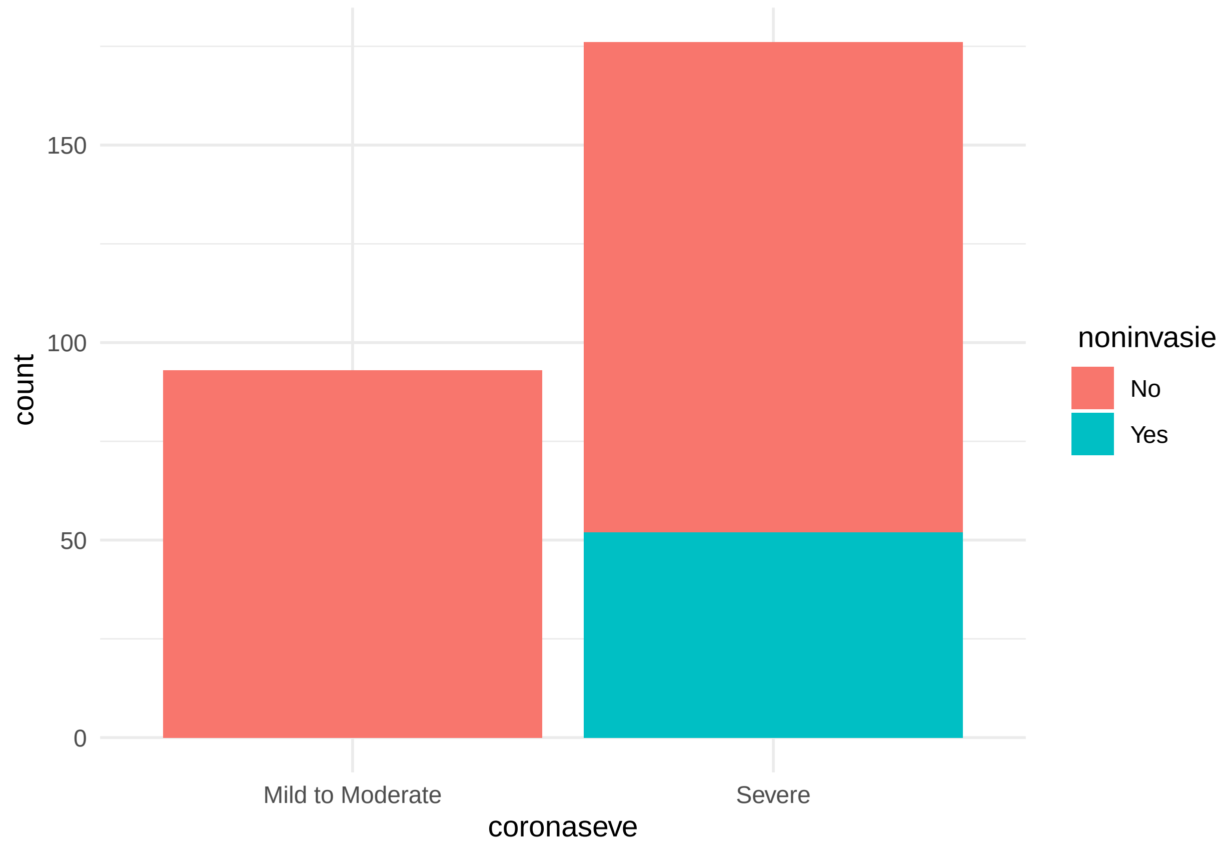
```r
# Create a bar plot using ggplot between coronaseve &  patchlobel
ggplot(data, aes(x = coronaseve, fill =    patchlobel)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " patchlobel") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Lower","Middle","Upper",
                                    "Lower+Middle" ,"Multi")) +
     theme_minimal()
```



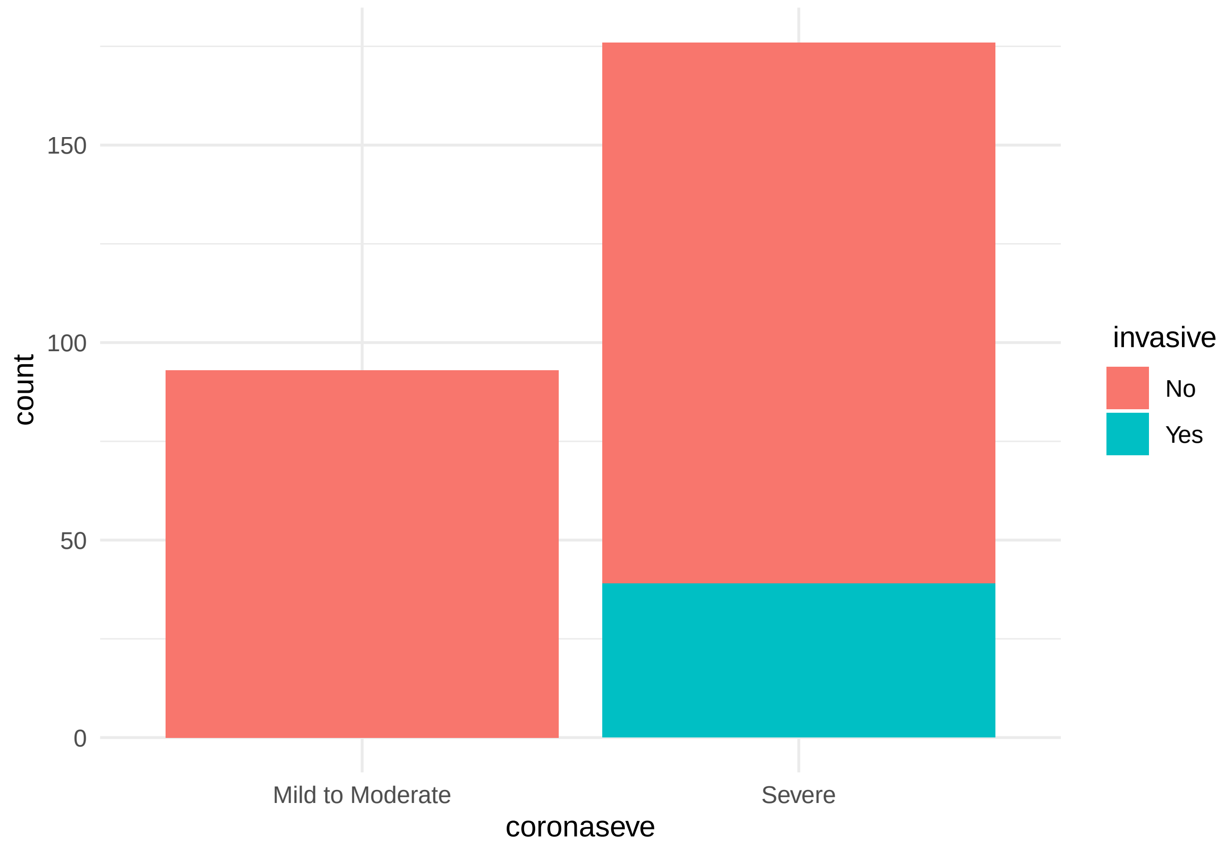
```r
# Create a bar plot using ggplot between coronaseve &  oxygenreq1
ggplot(data, aes(x = coronaseve, fill =    oxygenreq1)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " oxygenreq1") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



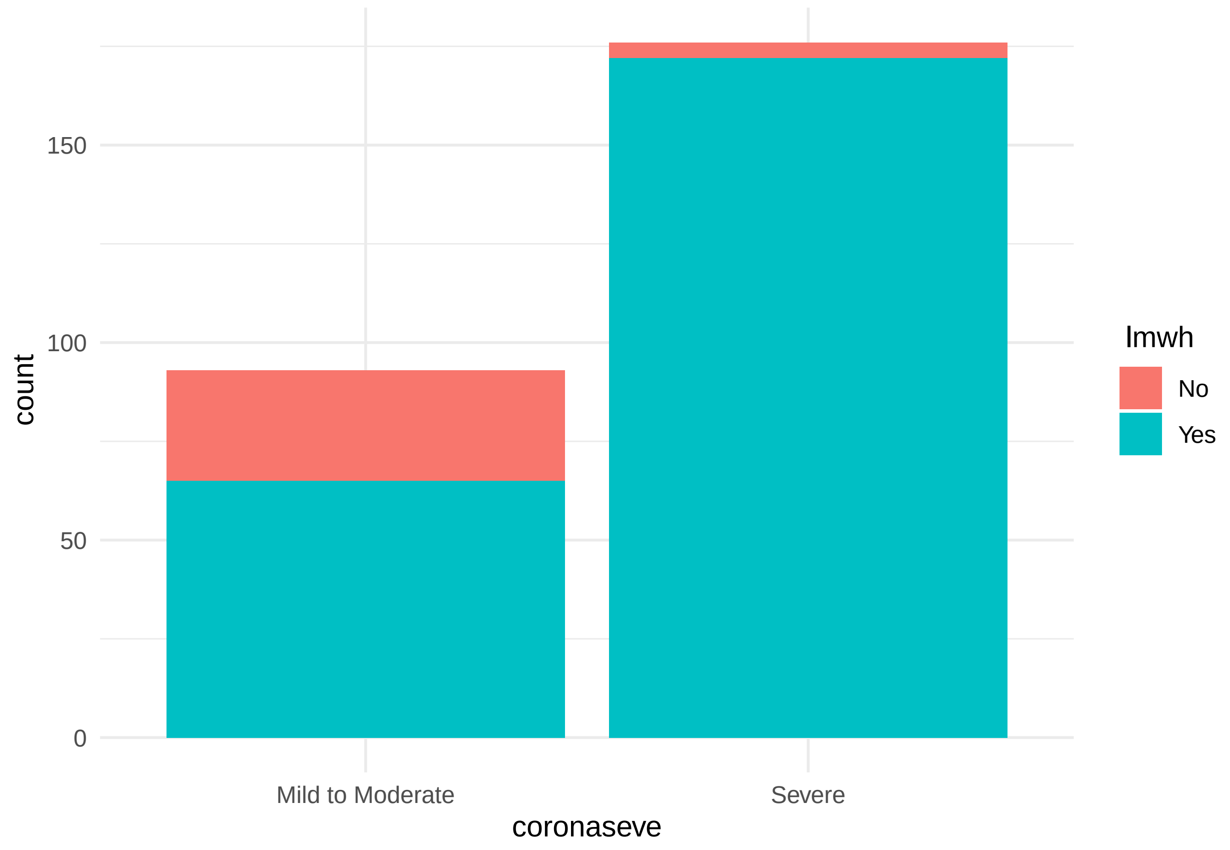
```r
# Create a bar plot using ggplot between coronaseve &  noninvasie
ggplot(data, aes(x = coronaseve, fill =    noninvasie)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " noninvasie") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



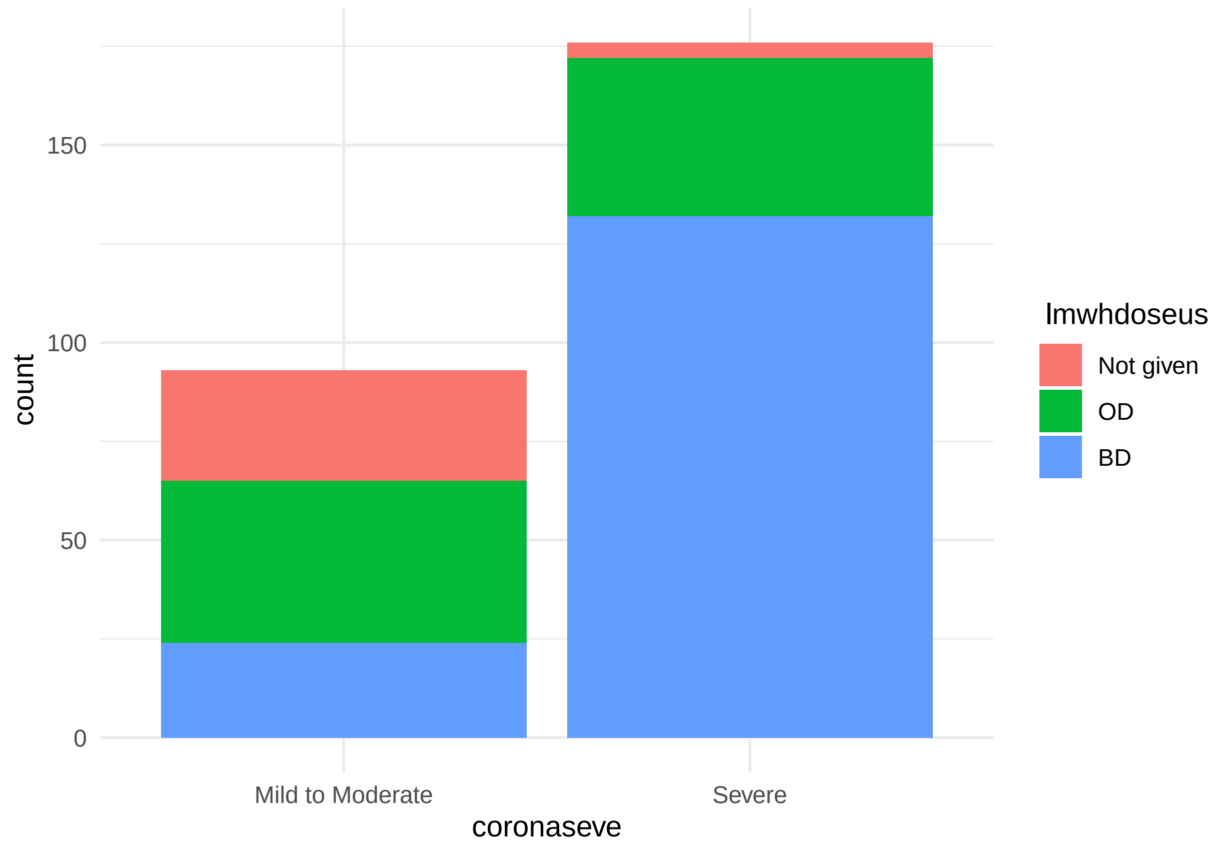
```r
# Create a bar plot using ggplot between coronaseve &  invasive
ggplot(data, aes(x = coronaseve, fill =    invasive)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " invasive") +
     scale_x_discrete(labels =c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



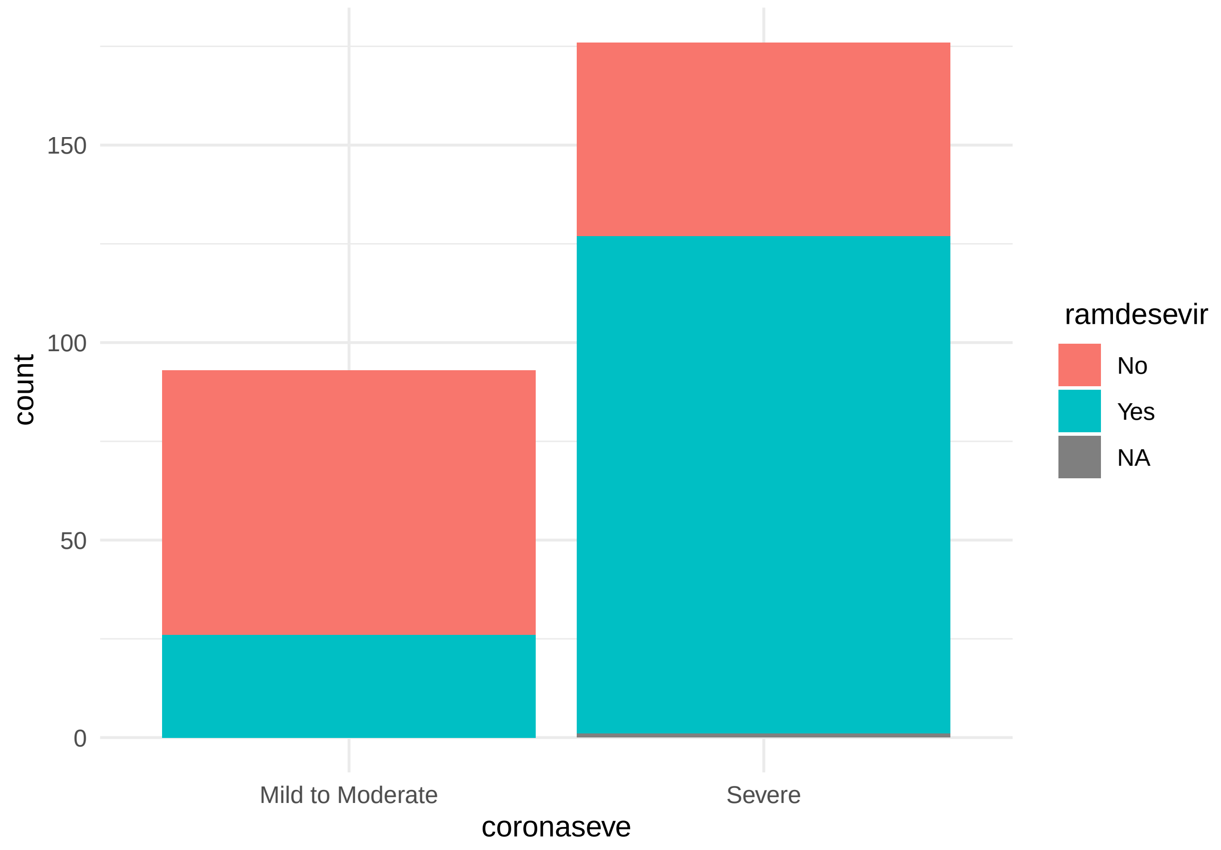
```
# Create a bar plot using ggplot between coronaseve &  lmwh
ggplot(data, aes(x = coronaseve, fill =    lmwh )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " lmwh ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



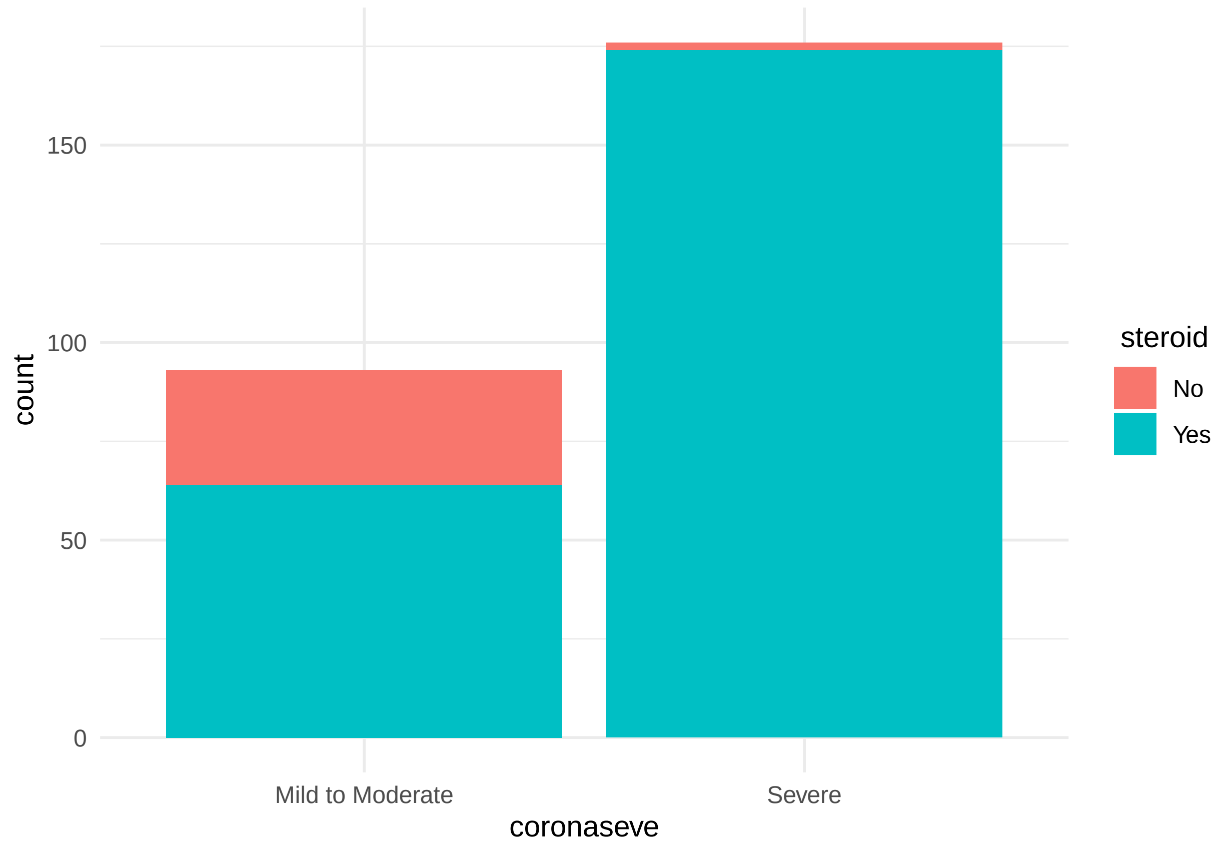
```r
# Create a bar plot using ggplot between coronaseve &  lmwhdoseus
ggplot(data, aes(x = coronaseve, fill =    lmwhdoseus )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " lmwhdoseus ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("Not given", "OD", "BD")) +
     theme_minimal()
```



```r
# Create a bar plot using ggplot between coronaseve &  ramdesevir
ggplot(data, aes(x = coronaseve, fill =    ramdesevir )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " ramdesevir ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



```
# Create a bar plot using ggplot between coronaseve &  steroid
ggplot(data, aes(x = coronaseve, fill =    steroid )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " steroid ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



Count of missing values predictor wise

```
a<-colnames(data)
na_count<-rep(NA,ncol(data))
for (i in 1:ncol(data)){
  na_count[i]<- sum(is.na(data[,i]))
}
na<-cbind(a , na_count)
na<-na[order(-na_count) , ]
## Arranging na_counts in descending order
## and suffling corresponding a's as well
NA_count<- na[which(na[,2] != 0) , ]
NA_count
```

```
##       a             na_count
##  [1,] "hba1c"       "205"
##  [2,] "obesity"     "139"
##  [3,] "bmi"         "129"
##  [4,] "ht"          "128"
##  [5,] "weig"        "126"
##  [6,] "pao2fio2"    "40"
##  [7,] "sars"        "39"
##  [8,] "sarsseveri"  "39"
##  [9,] "tropt"       "34"
## [10,] "tropt1"      "33"
## [11,] "ntprobnp"    "31"
## [12,] "ntprobnp1"  "31"
## [13,] "abgspo2"    "19"
## [14,] "serumfibri" "19"
## [15,] "serumfibr1" "19"
## [16,] "sferrtin"   "18"
## [17,] "serumferri" "18"
## [18,] "abgph"      "17"
## [19,] "il6"        "17"
## [20,] "il6absbva"  "17"
## [21,] "abgacidos"  "16"
## [22,] "cadtype"    "15"
## [23,] "ddimer"     "15"
## [24,] "ddimaerva"  "15"
## [25,] "resrate"    "14"
## [26,] "ecgheartra" "11"
## [27,] "qtinterval" "9"
## [28,] "correctedq" "9"
## [29,] "spo2"       "7"
## [30,] "tlc"        "6"
## [31,] "throbocyto" "6"
## [32,] "lqtinter"   "5"
## [33,] "hr"         "4"
## [34,] "plateletco" "4"
## [35,] "hb"         "3"
## [36,] "anemia"     "3"
## [37,] "creatinine" "3"
## [38,] "sodium"     "2"
## [39,] "potassium"  "2"
## [40,] "lft"        "2"
## [41,] "sgot"       "2"
## [42,] "sgpt"       "2"
## [43,] "bun"        "2"
## [44,] "aki"        "2"
## [45,] "sinustachy" "2"
## [46,] "htn"        "1"
## [47,] "cxrpneu"    "1"
## [48,] "patchprsen" "1"
## [49,] "patchside"  "1"
## [50,] "patchlobel" "1"
## [51,] "pleuraleff" "1"
## [52,] "oxygenreq1" "1"
## [53,] "ramdesevir" "1"
```

```r
dim(NA_count)
```

```
## [1] 53  2
```

```r
# NA_count gives us information about no of NA in given variable
```

Imputing *Height(ht)* and *Weight (weig)* of Patient assuming Normal Distribution and doing Random sampling from distribution of observed data

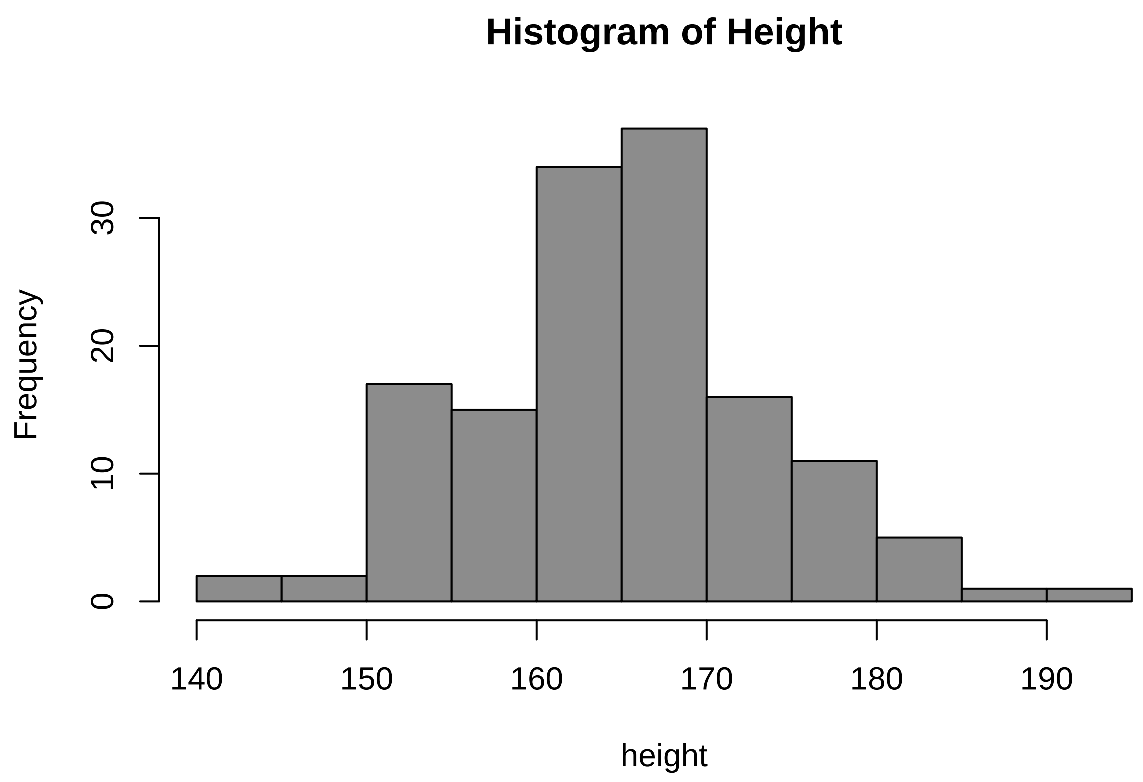
```r
data_imputed <- data
# dim(data_imputed)
#
# ## trying to omit observations with missing values
na_omit_data <- na.omit(data)
dim(na_omit_data)
```

```
## [1]  15 111
```

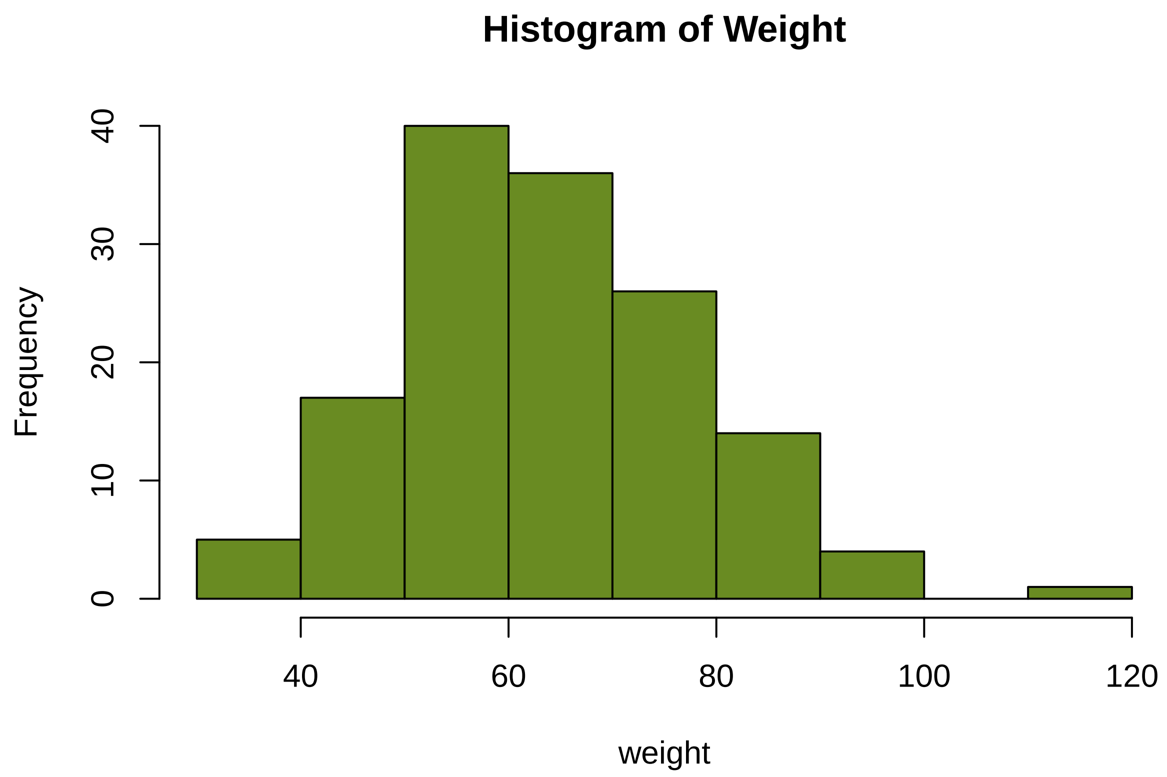
```r
print("Height and Weight can be assumed Normally Distributed")
```

```
## [1] "Height and Weight can be assumed Normally Distributed"
```

```r
hist(data$ht , main = "Histogram of Height" , col = "gray55" , border = T,
     xlab= "height")
```



```r
hist(data$weig, main = "Histogram of Weight", col = "olivedrab4" , border = T,
     xlab = "weight")
```



```r
rnormimpfun <- function(x){
  # makinga subset of x, to remove the effect of any potential outlier
  y <- subset(x ,
              x > quantile(x , 0.25 , na.rm = T) ,
              x < quantile(x , 0.75 , na.rm = T))

 for(i in 1:length(x)){
   if(is.na(x[i]) == TRUE ){
     x[i] <- round( rnorm(1, mean=mean(y , na.rm = T), sd=sd(y) ), 2)

    }
  }
  return(x)
}


summary(data$ht)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   142.2   160.0   165.1   164.4   168.0   190.5     128
```

Imputation of Height

```
data_imputed$ht <- rnormimpfun(data$ht)

# ## Summary of height (ht) after imputation
# summary(data_imputed$ht)
#
# ## summary of height before imputation
# summary(data$ht)
```

Imputation of weight

```
#summary(data$weig)

data_imputed$weig <- rnormimpfun(data$weig)

# ## Summary of weight (weig) after imputation
# summary(data_imputed$weig)
#
# ## Summary of weight (weig) before imputation
# summary(data$weig)
```

Impuation of BMI

```
for (i in 1:length(data_imputed$bmi)){
  if (is.na(data_imputed$bmi[i]) == T){
    data_imputed$bmi[i] <- (data_imputed$weig[i]/(data_imputed$ht[i]/100)^2)
  }
}


# ## Summary of bmi (bmi) after imputation
# summary(data_imputed$bmi)
#
# ## Summary of bmi (bmi) before imputation
# summary(data$bmi)
#
#
```

Imputation of obesity, based on bmi calculation

```
#
for( i in 1:length(data$obesity ) ){
  if(is.na(data$obesity[i]) == T ){
    if(data_imputed$bmi[i] >= 25 ){
      data_imputed$obesity[i] <- 1
    }else{data_imputed$obesity[i] <- 0}
  }
}
```

Imputation of continuous variable whose na_count is less than or equal to 20

We choosed to impute with random values form the range of observed data as number of missing values were less than 10%

```
col_names_cont <- c("hr" , "bun", "potassium" , "sodium", "creatinine",
                    "hb", "plateletco" , "tlc" , "spo2" , "correctedq" ,
                    "qtinterval" , "ecgheartra" , "resrate" , "ddimaerva",
                    "il6absbva", "abgph" , "serumferri" , "abgspo2" )



length(col_names_cont)
```

```
## [1] 18
```

```
for(i in col_names_cont){
  x <- x + 1
  summary(data[,i])
  data_imputed[, i][is.na(data_imputed[,i])] <-
    round(runif(sum(is.na(data_imputed[, i])),
                min = quantile(data[,i] , 0.25 , na.rm = T) ,
                max = quantile(data[,i] , 0.75 , na.rm = T)) ,2)
  summary(data_imputed[,i])
  }
```

Imputation of categorical variables

We can impute missing values with central measure for the variables which has 20 or less number of missing values. As 20 is less than even 10% of total *number of observations we have!**

```
# collection of col names that need to be imputed with Mode values

col_imputed_mode <- c("ramdesevir", "oxygenreq1", "pleuraleff", "patchlobel",
                  "patchside" , "patchprsen" , "cxrpneu" , "htn",
                  "sinustachy","aki" , "sgpt" , "sgot" , "lft", "anemia",
                  "lqtinter", "throbocyto", "ddimer" , "cadtype", "abgacidos" ,
                  "il6", "sferrtin", "serumfibri" )


# # Funtion for Statistical mode values in any vector x
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}


for(j in col_imputed_mode){
  mod<- Mode(data[,j]) # Mode is pre-defined function for Statistical mode
  for(i in 1:nrow(data_imputed)){
    if(is.na(data_imputed[i , j]) == T){
      data_imputed[i , j] <- mod
    }
  }
}
```

Let's check number of missing values left

```
a<-colnames(data_imputed)
na_count<-rep(NA,ncol(data_imputed))
for (i in 1:ncol(data_imputed)){
  na_count[i]<- sum(is.na(data_imputed[,i]))
}

na<-cbind(a , na_count)

na<-na[order(-na_count) , ] ## Arranging na_counts in descending order
## and suffling corresponding a's as well

NA_count<- na[which(na[,2] != 0) , ] # selecting which has non zero no of NA's
NA_count
```

```
##       a            na_count
##  [1,] "hba1c"      "205"
##  [2,] "pao2fio2"   "40"
##  [3,] "sars"       "39"
##  [4,] "sarsseveri" "39"
##  [5,] "tropt"      "34"
##  [6,] "tropt1"     "33"
##  [7,] "ntprobnp"   "31"
##  [8,] "ntprobnp1"  "31"
##  [9,] "serumfibr1" "19"
```

```
#dim(NA_count)
```

Variables with missing values more than **30**

One easy option is to drop these variables, will have to drop only 9 variables out of 111, and try building some model which can significantly help us in solving this classification problem. Another thing we can do is analyse the reason of missingness and proceed according to that.

For now let's drop those variables

```
# Check for null values in each column
null_columns <- colSums(is.na(data_imputed)) > 0

# Print the names of columns with null values
names_with_null <- names(null_columns[null_columns])
#print(names_with_null)
# Create a vector of column names to be removed (These columns have null values
# greater than 19)

columns_to_remove <- names_with_null
```

```r
# Remove the specified columns from the data frame

data_non_null <- data_imputed[, !names(data_imputed) %in% columns_to_remove]
#str(data_non_null)
dim(data_non_null)
```

```
## [1] 269 102
```

```r
# run the code below to verify no NA's left

# a<-colnames(data_non_null)
# na_count<-rep(NA,ncol(data_non_null))
# for (i in 1:ncol(data_non_null)){
#   na_count[i]<- sum(is.na(data_non_null[,i]))
# }
#
# na<-cbind(a , na_count)
#
# na<-na[order(-na_count) , ] ## Arranging na_counts in descending order
# ## and suffling corresponding a's as well
#
# NA_count<- na[which(na[,2] != 0),] #selecting which has non zero no of NA's
# NA_count
# #dim(NA_count)
```

Severity

```r
# Create a vector of column names to be removed (According to me these are
# not necessary for the classification models for severity )

remove_col_unnecessary<-c( "name","coviddeath","noncovidde","procovidde",
                          "discharge" , "los" , "ventilator", "icushift" ,
                          "death" )

data_seve<-data_non_null[, !names(data_non_null) %in%  remove_col_unnecessary]

dim(data_seve)
```

```
## [1] 269  93
```

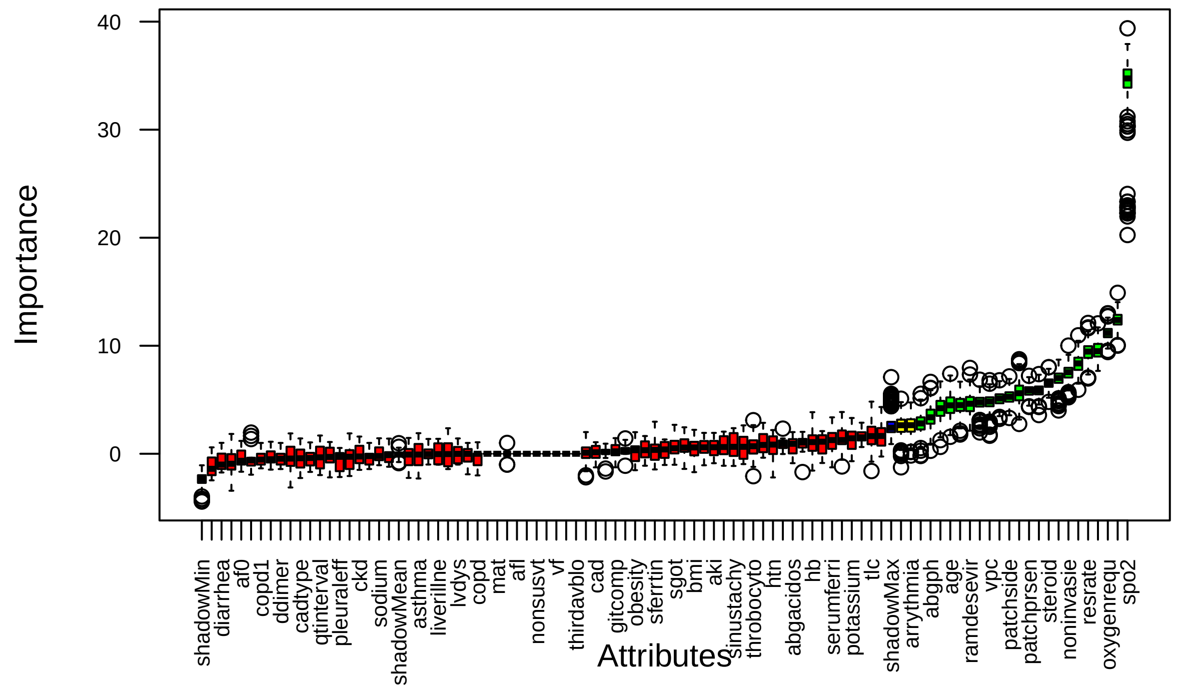
```r
#summary(data_seve)
```

Application of Boruta for feature selection

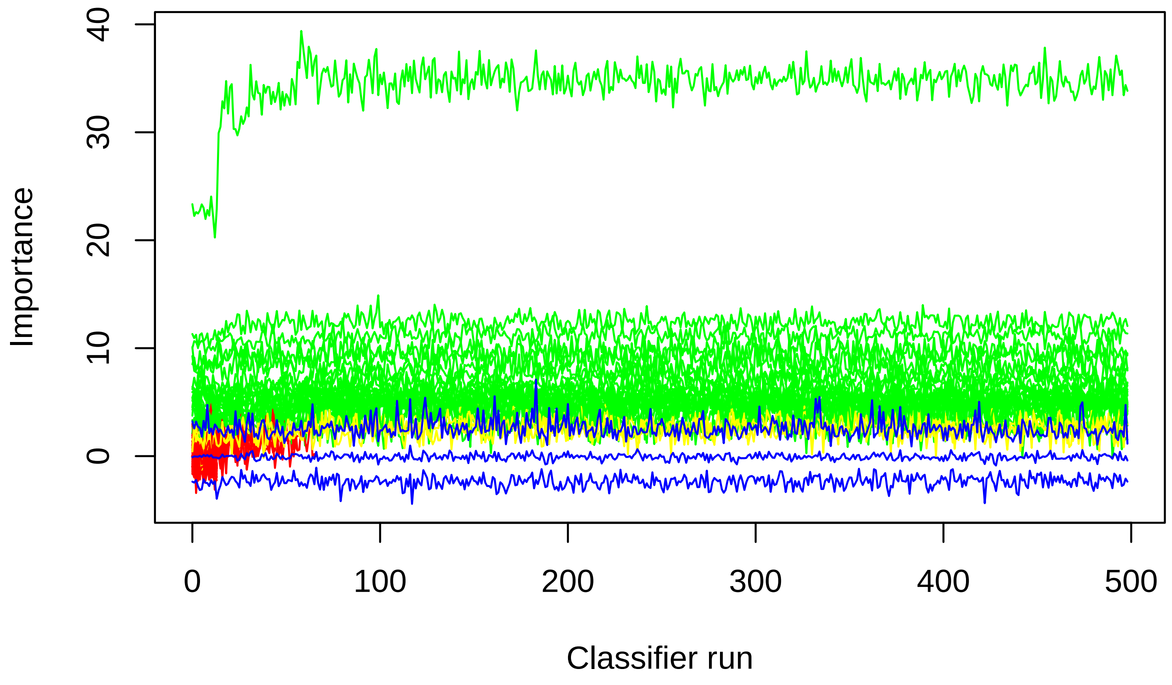
```r
# Feature Selection
set.seed(111)
boruta_seve <- Boruta(coronaseve ~ ., data = data_seve, maxRuns = 500)

# The above command applies the Boruta feature selection algorithm to
# the dataset data_seve . The formula death ~ . specifies that
# the variable death is the outcome variable, and . indicates that all other
# variables in the dataset are considered as potential predictors. The doTrace
# parameter controls the level of verbosity during the Boruta analysis, with
# a value of 2 indicating more detailed output. The maxRuns parameter
# specifies the maximum number of iterations to run the algorithm. The result
# of the Boruta analysis is stored in the boruta_seve object.

print(boruta_seve)
```

```
## Boruta performed 499 iterations in 26.00141 secs.
##  22 attributes confirmed important: abgph, abgspo2, age, bun, cxrpneu
## and 17 more;
##  68 attributes confirmed unimportant: abgacidos, af0, afl, aki, anemia
## and 63 more;
##  2 tentative attributes left: arrhythmia, arrythmia;
```

```r
#The above command prints the summary or information about the boruta object.
# It displays the variables considered in the Boruta analysis, their
# importance scores, and the final decision on whether they are selected
# as important predictors or not. This information helps assess the relevance
# of variables in predicting the outcome variable (death) based on the
# Boruta analysis.

plot(boruta_seve, las = 2, cex.axis = 0.7)
```



```
# Above command is used to create a plot of the Boruta analysis results.
# The boruta_seve object is passed as the argument, and additional parameters
#  las and cex.axis are used to modify the appearance of the plot. las controls
# the orientation of the axis labels, and cex.axis controls the size of the
# axis labels. By customizing these parameters, you can enhance the
# readability of the plot and visualize the importance of variables determined
# by Boruta.

plotImpHistory(boruta_seve)
```



```
bor <- TentativeRoughFix(boruta_seve)
print(bor)
```

```
## Boruta performed 499 iterations in 26.00141 secs.
## Tentatives roughfixed over the last 499 iterations.
##  24 attributes confirmed important: abgph, abgspo2, age, arrhythmia,
## arrythmia and 19 more;
##  68 attributes confirmed unimportant: abgacidos, af0, afl, aki, anemia
## and 63 more;
```

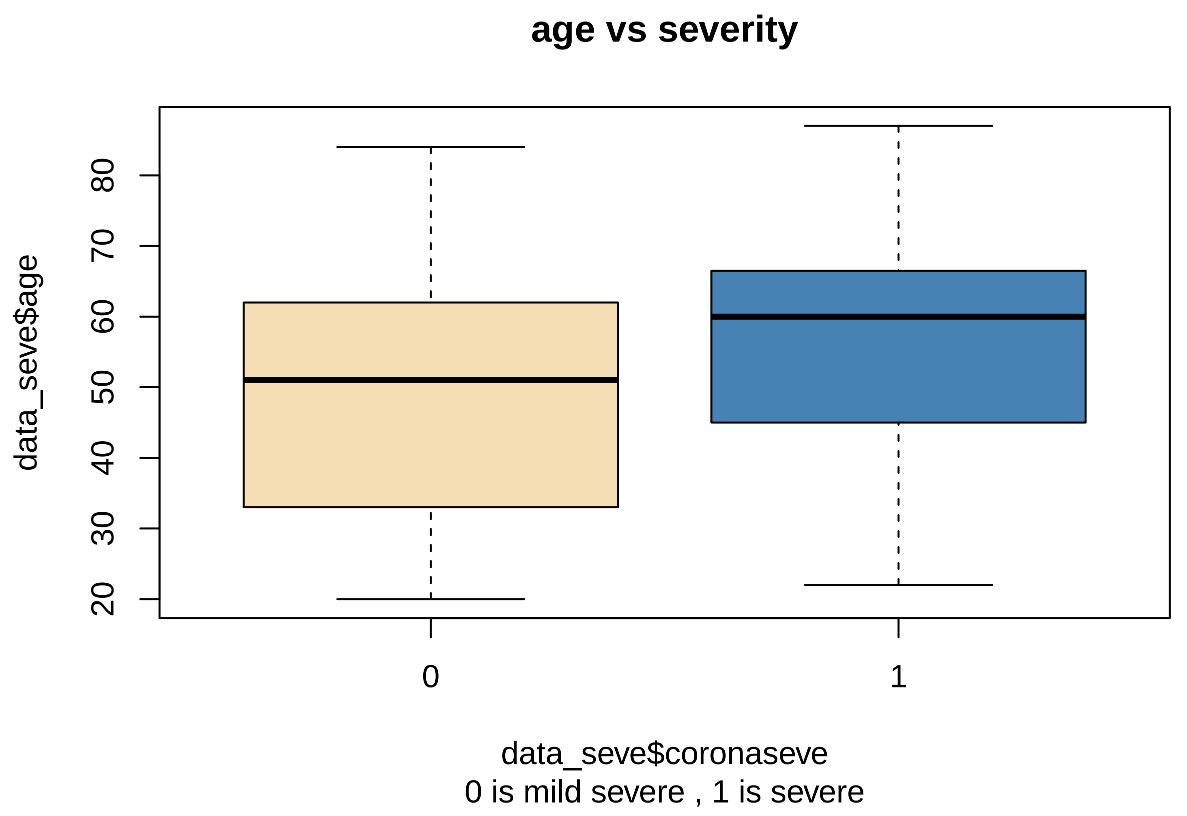
```
att_seve<-attStats(boruta_seve)
```

Data Visualization, of Important features with coronaseve

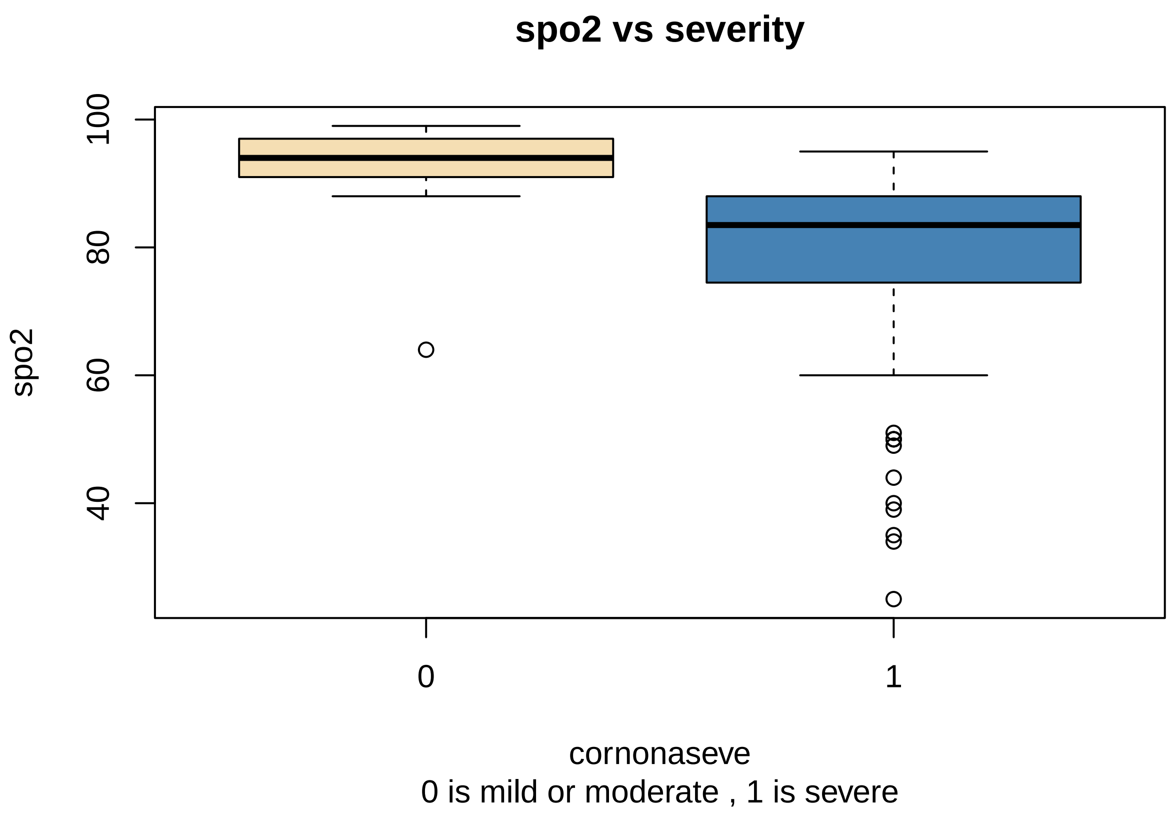
```
conf_var_seve <- c("coronaseve" , "age" , "respirator" , "spo2" , "resrate" ,
                   "bun" , "abgph" , "abgspo2" , "ventricula" , "vpc" ,
                   "oxygenrequ" , "cxrpneu" , "patchprsen" , "patchside" ,
                   "patchlobel" , "oxygenreq1" , "noninvasie" , "invasive" ,
                   "lmwh" , "lmwhdoseus" , "ramdesevir" , "steroid")
# conf_var_seve contains columns names those are important

data_seve_conf <- data_seve[, conf_var_seve]

summary(data_seve_conf)
```

```
##  coronaseve      age          respirator      spo2          resrate
##  0: 93      Min.   :20.00   0: 48      Min.   :25.00   Min.   :12.00
##  1:176      1st Qu.:42.00   1:221      1st Qu.:80.00   1st Qu.:20.00
##             Median :56.00              Median :88.00   Median :22.00
##             Mean   :54.28              Mean   :84.18   Mean   :22.94
##             3rd Qu.:65.00              3rd Qu.:92.00   3rd Qu.:26.00
##             Max.   :87.00              Max.   :99.00   Max.   :60.00
##       bun              abgph          abgspo2       ventricula vpc
##  Min.   :  9.00   Min.   :7.120   Min.   : 67.00   0:250      0:250
##  1st Qu.: 30.00   1st Qu.:7.400   1st Qu.: 88.00   1: 19      1: 19
##  Median : 41.00   Median :7.440   Median : 93.00
##  Mean   : 60.78   Mean   :7.426   Mean   : 91.58
##  3rd Qu.: 66.00   3rd Qu.:7.480   3rd Qu.: 97.00
##  Max.   :577.00   Max.   :7.620   Max.   :100.00
##  oxygenrequ cxrpneu patchprsen patchside patchlobel oxygenreq1 noninvasie
##  0: 36      0: 24   0: 24      0: 24     0: 24      0: 38      0:217
##  1:233      1:245   1:245      1:232     1: 22      1:231      1: 52
##                                2:  8     2:  7
##                                3:  5     4: 65
##                                          5:151
##
##  invasive lmwh    lmwhdoseus ramdesevir steroid
##  0:230    0: 32   0: 32      0:116      0: 31
##  1: 39    1:237   1: 81      1:153      1:238
##                   2:156
##
##
##
```

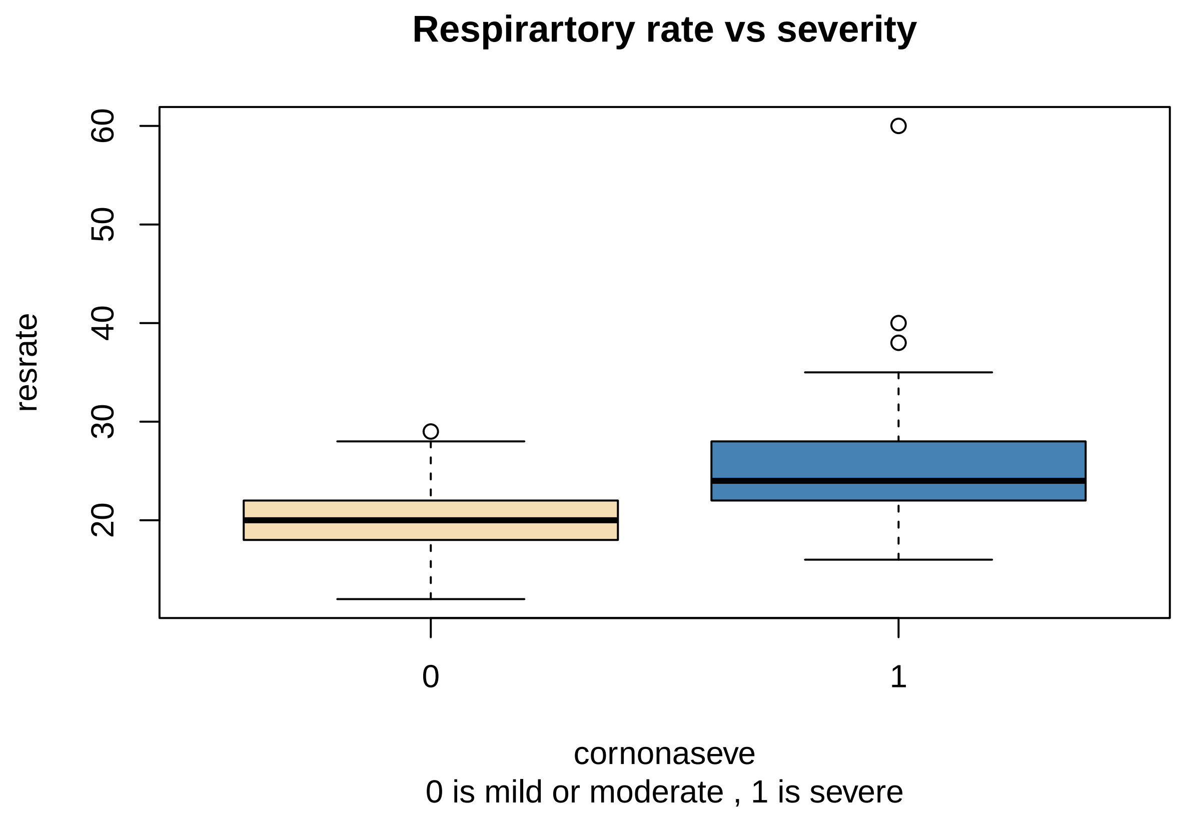
```
boxplot(data_seve$age ~ data_seve$coronaseve ,
        main = "age vs severity" ,
        sub ="0 is mild severe , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



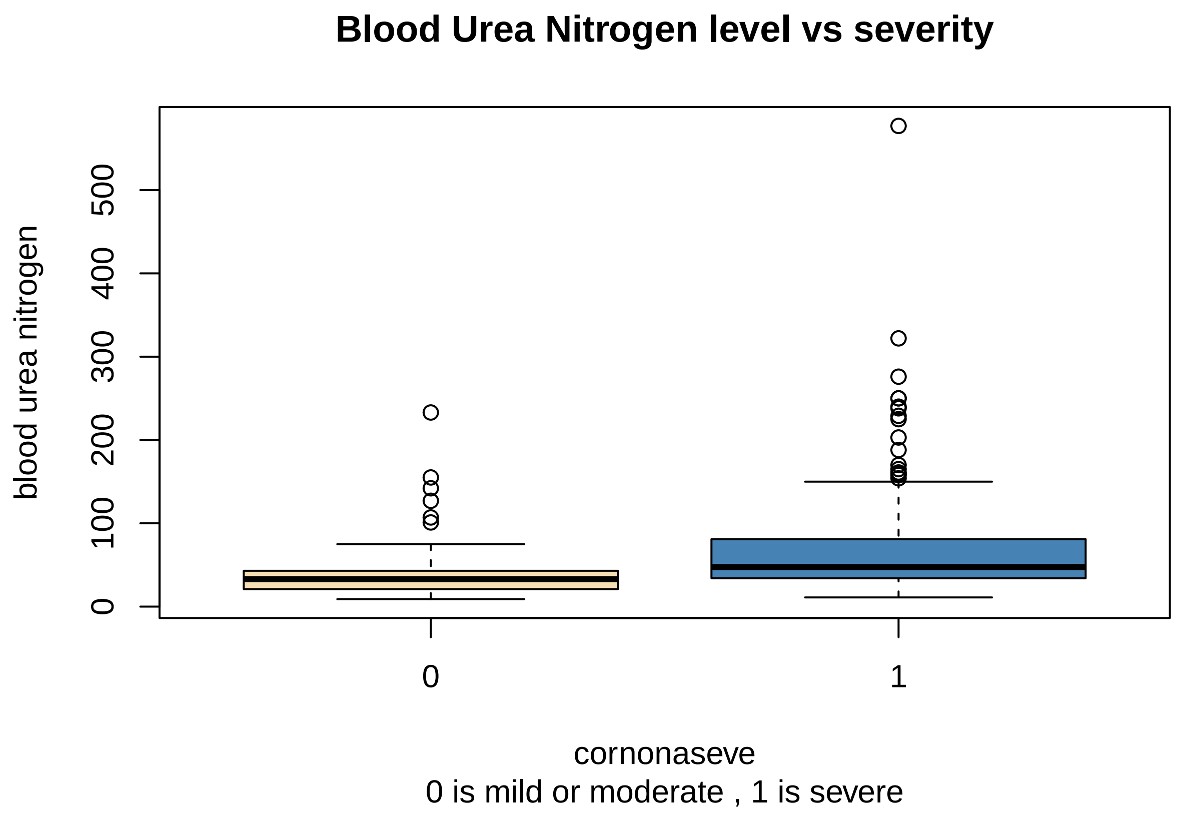
```r
boxplot(data_seve$spo2 ~ data_seve$coronaseve,
        xlab= "cornonaseve" , ylab = "spo2",
        main = "spo2 vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



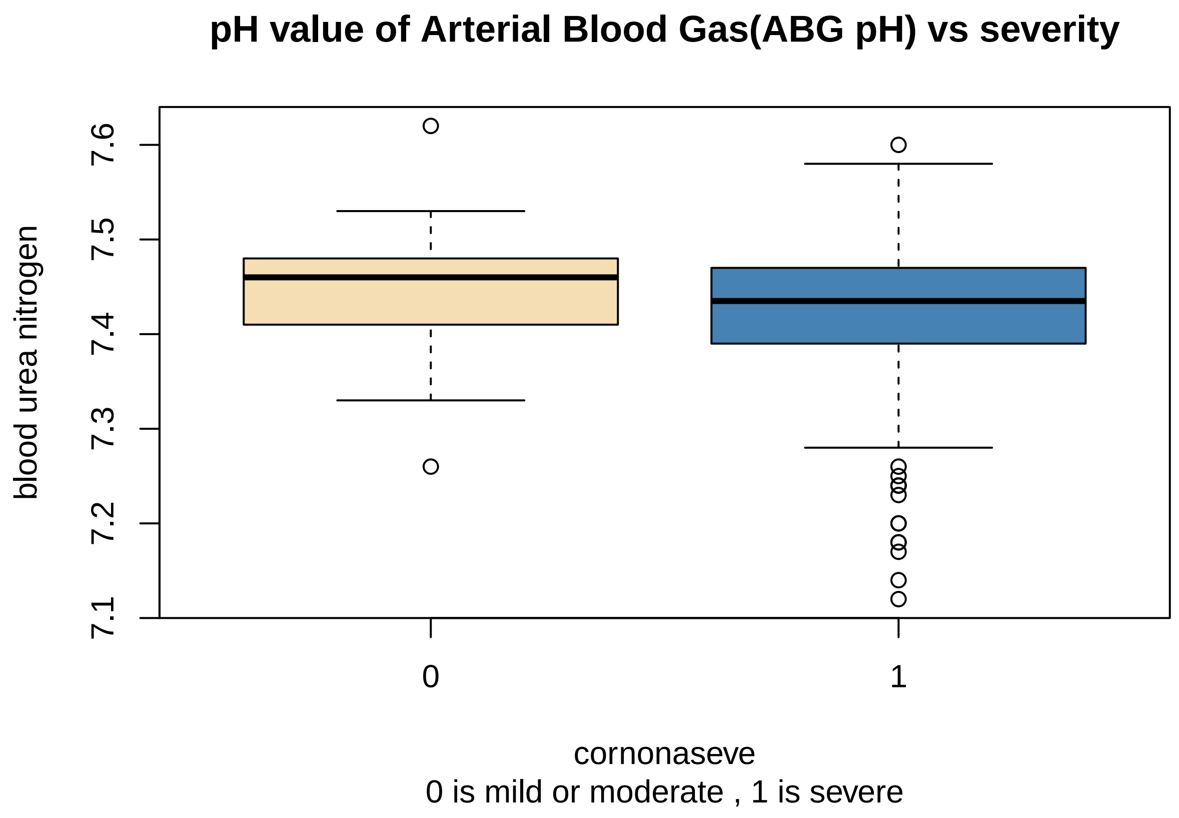
```r
boxplot(data_seve$resrate ~ data_seve$coronaseve,
        xlab= "cornonaseve" , ylab = "resrate",
        main = "Respirartory rate vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



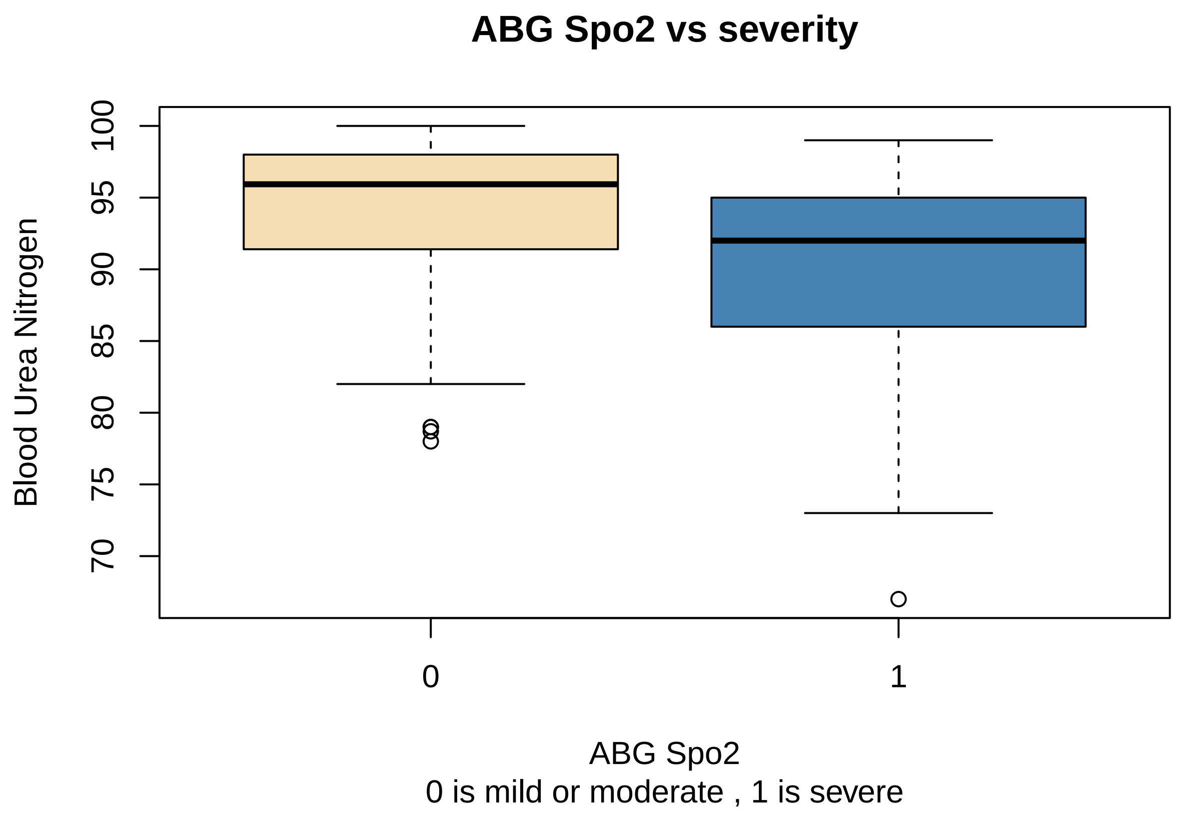
```r
boxplot(data_seve$bun ~ data_seve$coronaseve,
        xlab= "cornonaseve" , ylab = "blood urea nitrogen",
        main = "Blood Urea Nitrogen level vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



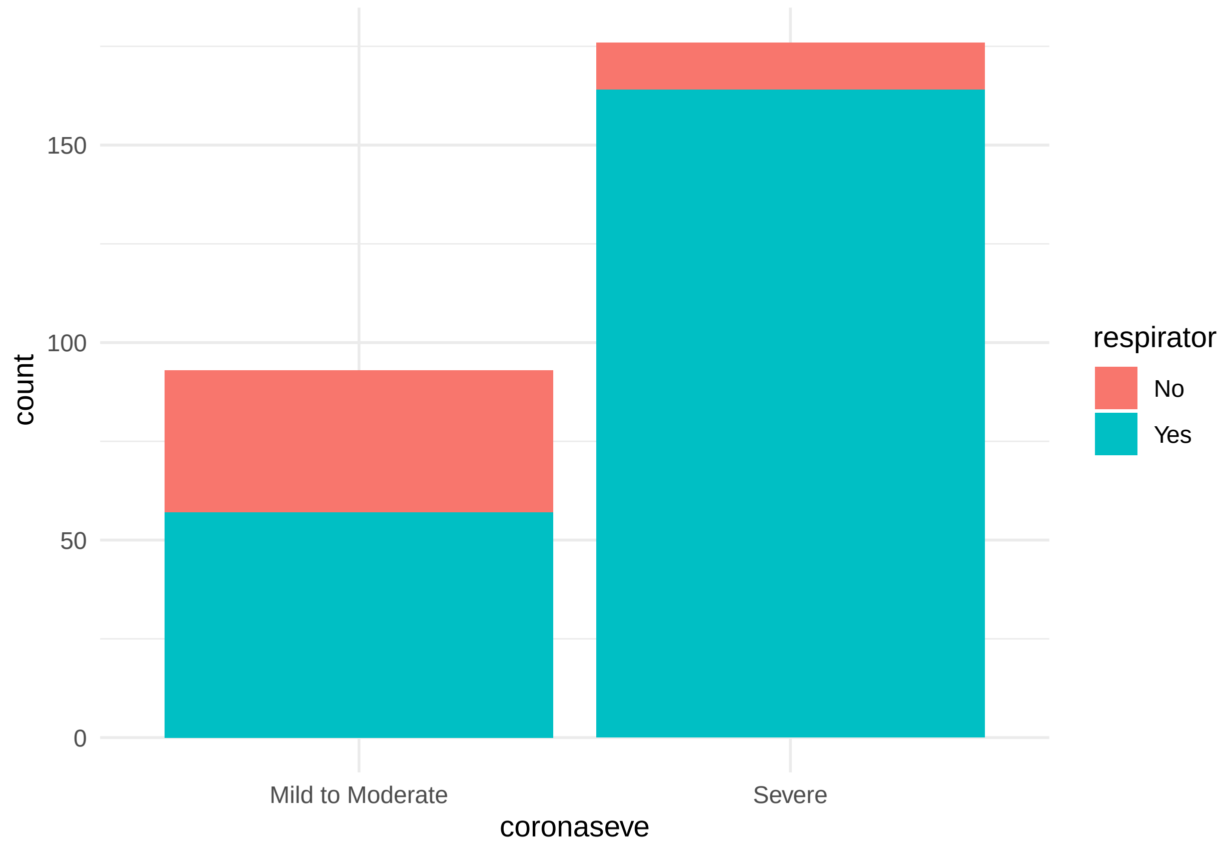
```r
boxplot(data_seve$abgph ~ data_seve$coronaseve,
        xlab= "cornonaseve" , ylab = "blood urea nitrogen",
        main = "pH value of Arterial Blood Gas(ABG pH) vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



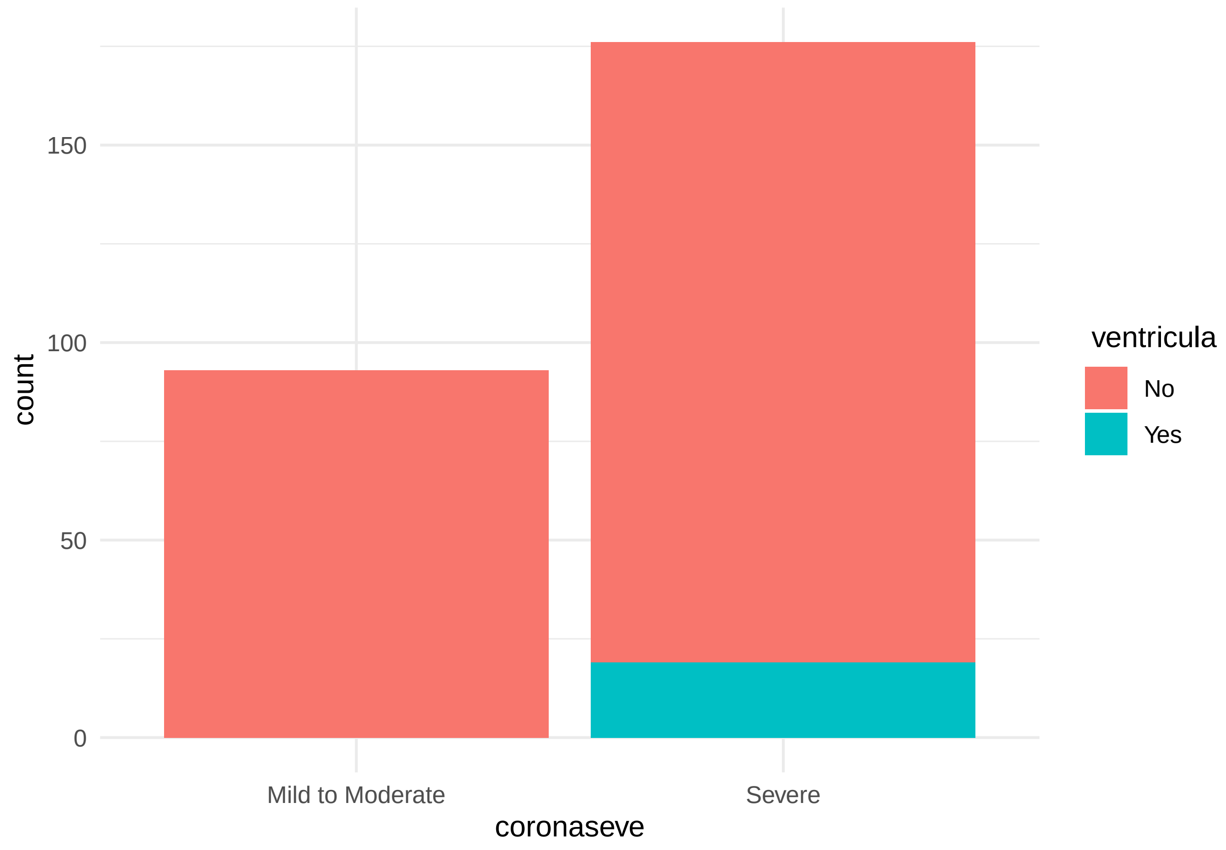
```
boxplot(data_seve$abgspo2 ~ data_seve$coronaseve,
        xlab= "ABG Spo2" , ylab = "Blood Urea Nitrogen",
        main = "ABG Spo2 vs severity" ,
        sub = "0 is mild or moderate , 1 is severe" ,
        col = c("wheat" , "steelblue"))
```



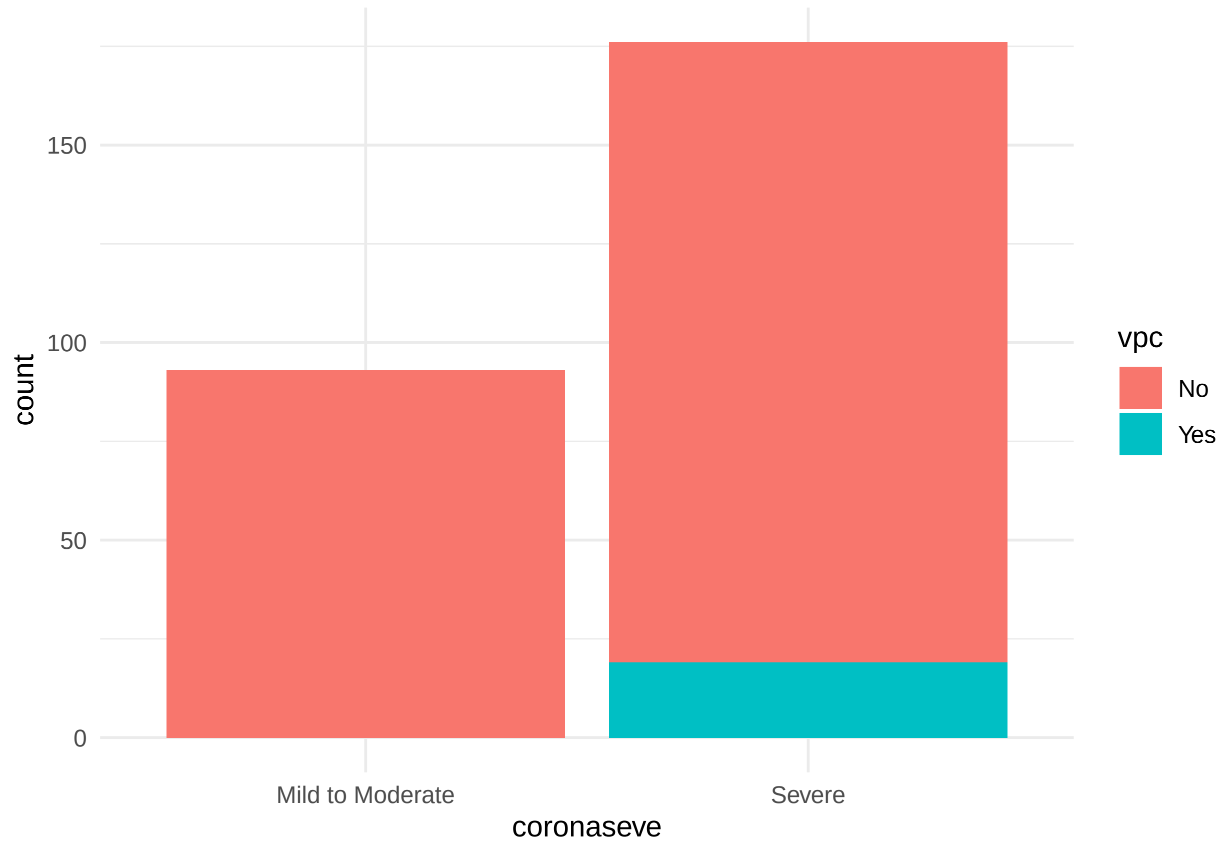
```r
# Create a bar plot using ggplot between coronaseve & respirator
ggplot(data_seve_conf, aes(x = coronaseve, fill =   respirator)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "respirator") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



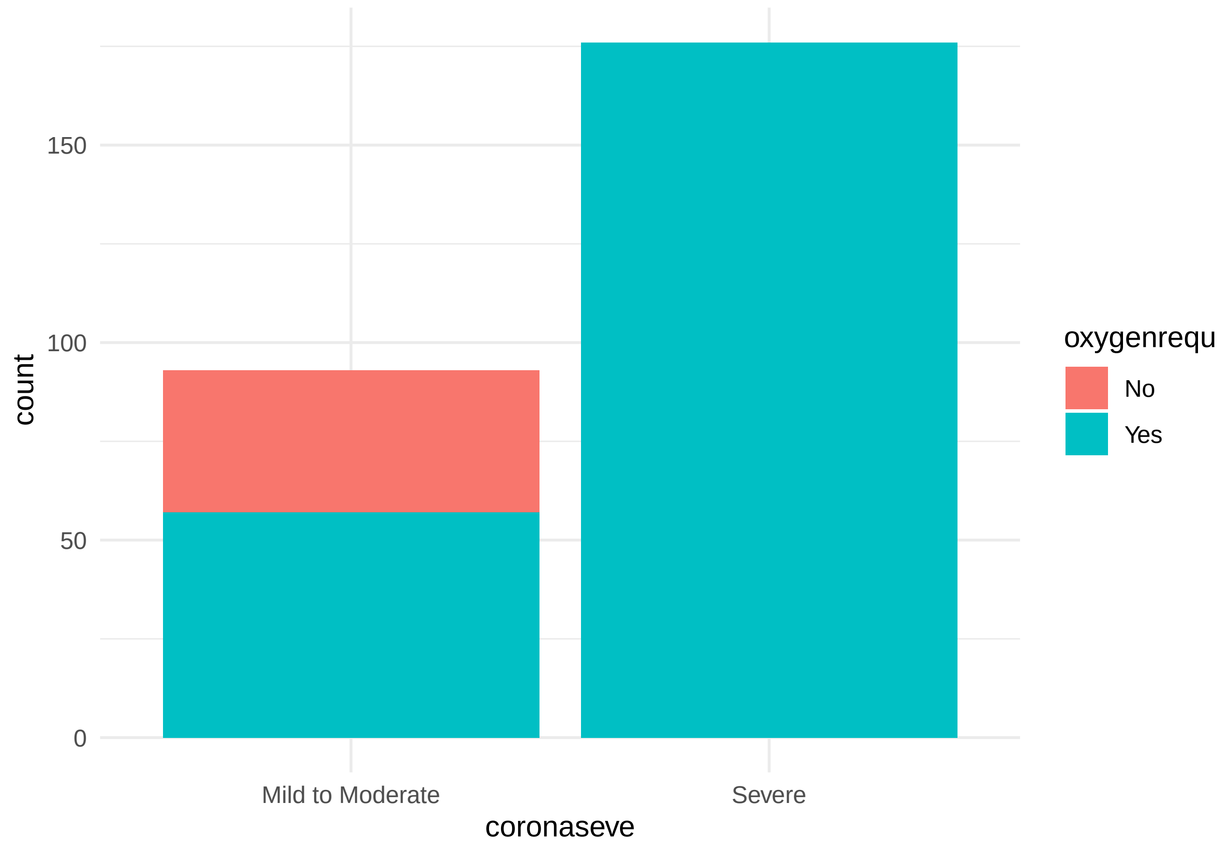
```r
# Create a bar plot using ggplot between coronaseve &  ventricula
ggplot(data_seve_conf, aes(x = coronaseve, fill =    ventricula)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " ventricula") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



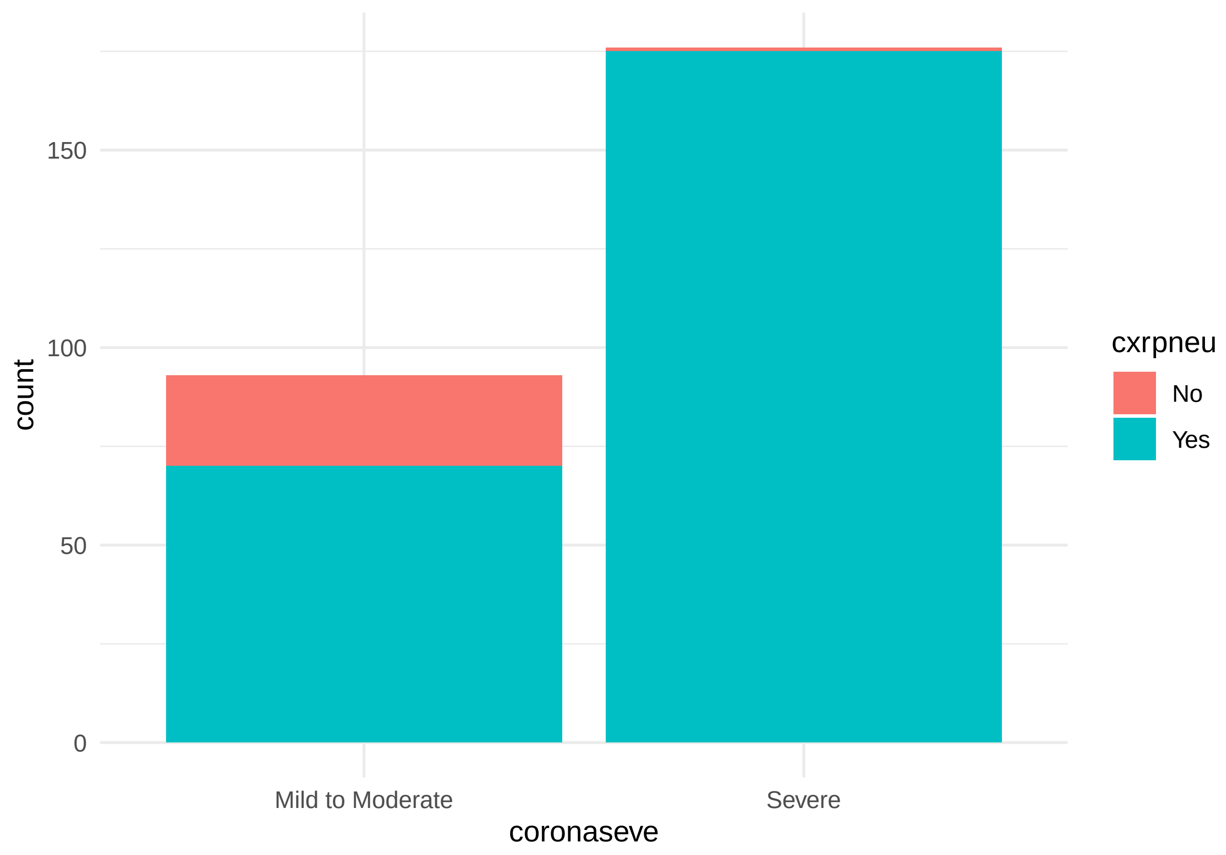
```r
# Create a bar plot using ggplot between coronaseve &  vpc
ggplot(data_seve_conf, aes(x = coronaseve, fill =    vpc)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "vpc") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



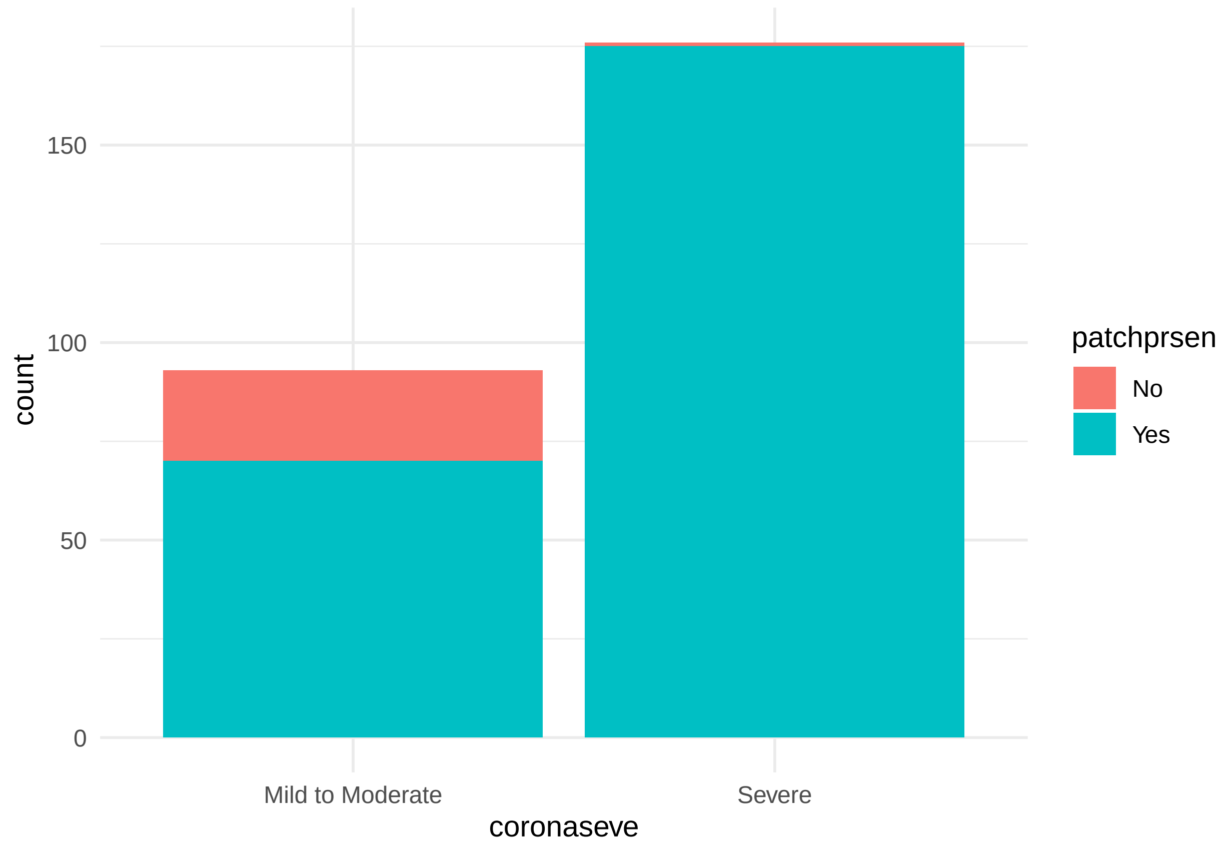
```r
# Create a bar plot using ggplot between coronaseve &  oxygenrequ
ggplot(data_seve_conf, aes(x = coronaseve, fill =    oxygenrequ)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "oxygenrequ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



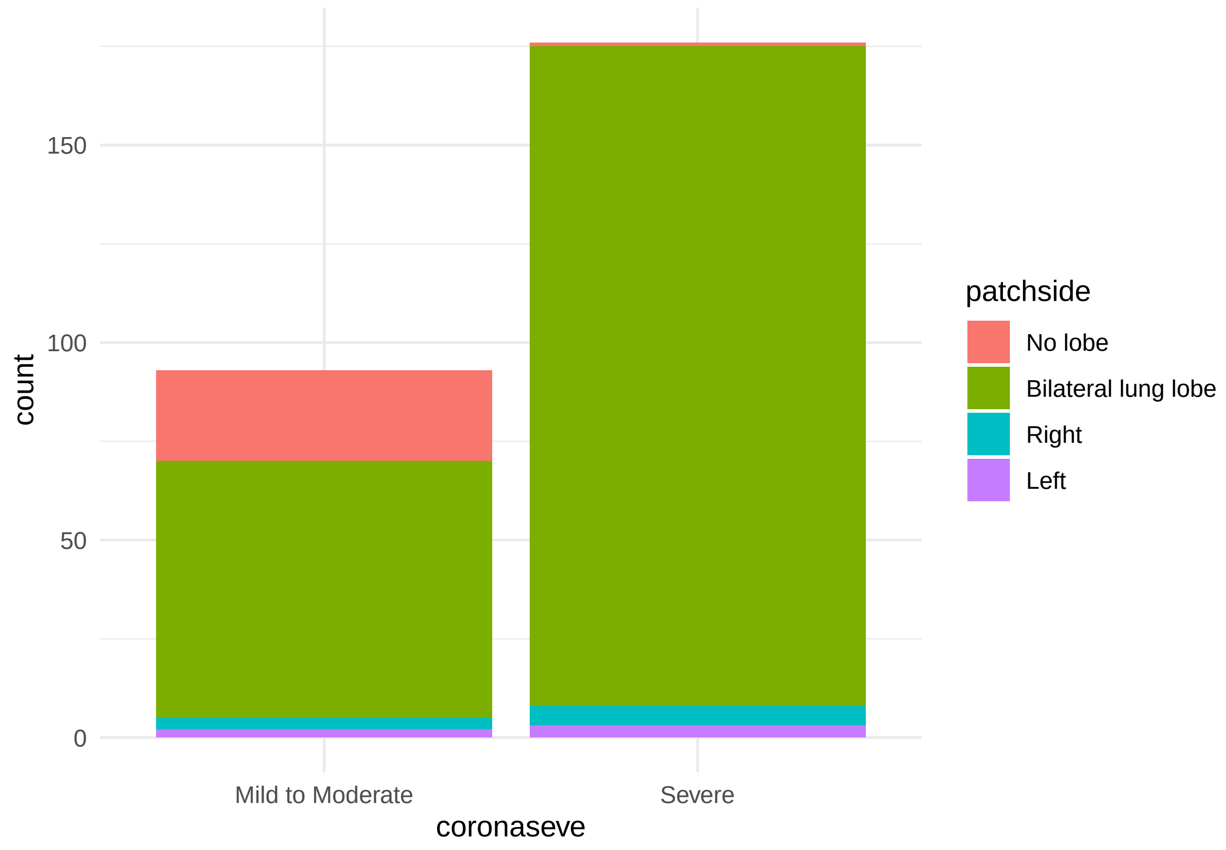
```r
# Create a bar plot using ggplot between coronaseve &  cxrpneu
ggplot(data_seve_conf, aes(x = coronaseve, fill =    cxrpneu)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "cxrpneu") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



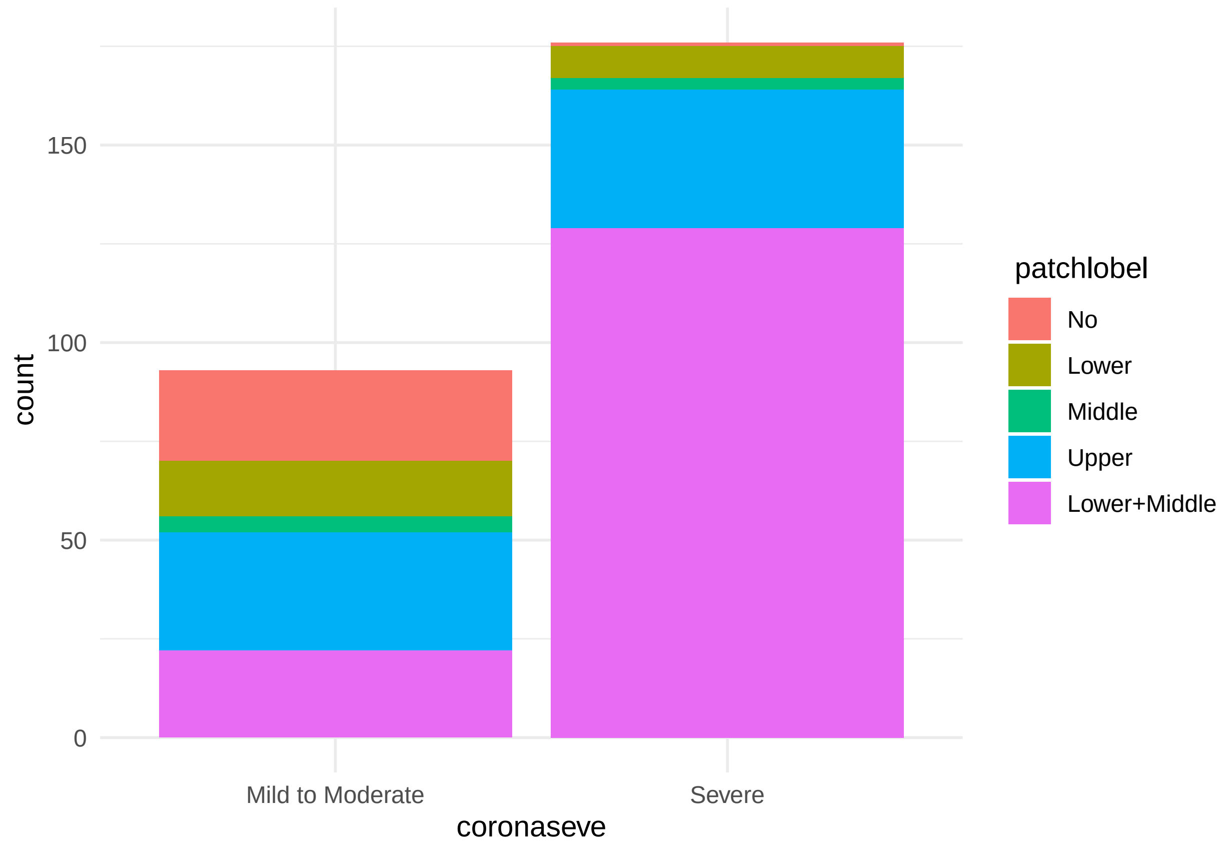
```r
# Create a bar plot using ggplot between coronaseve &  patchprsen
ggplot(data_seve_conf, aes(x = coronaseve, fill =    patchprsen)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "patchprsen") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



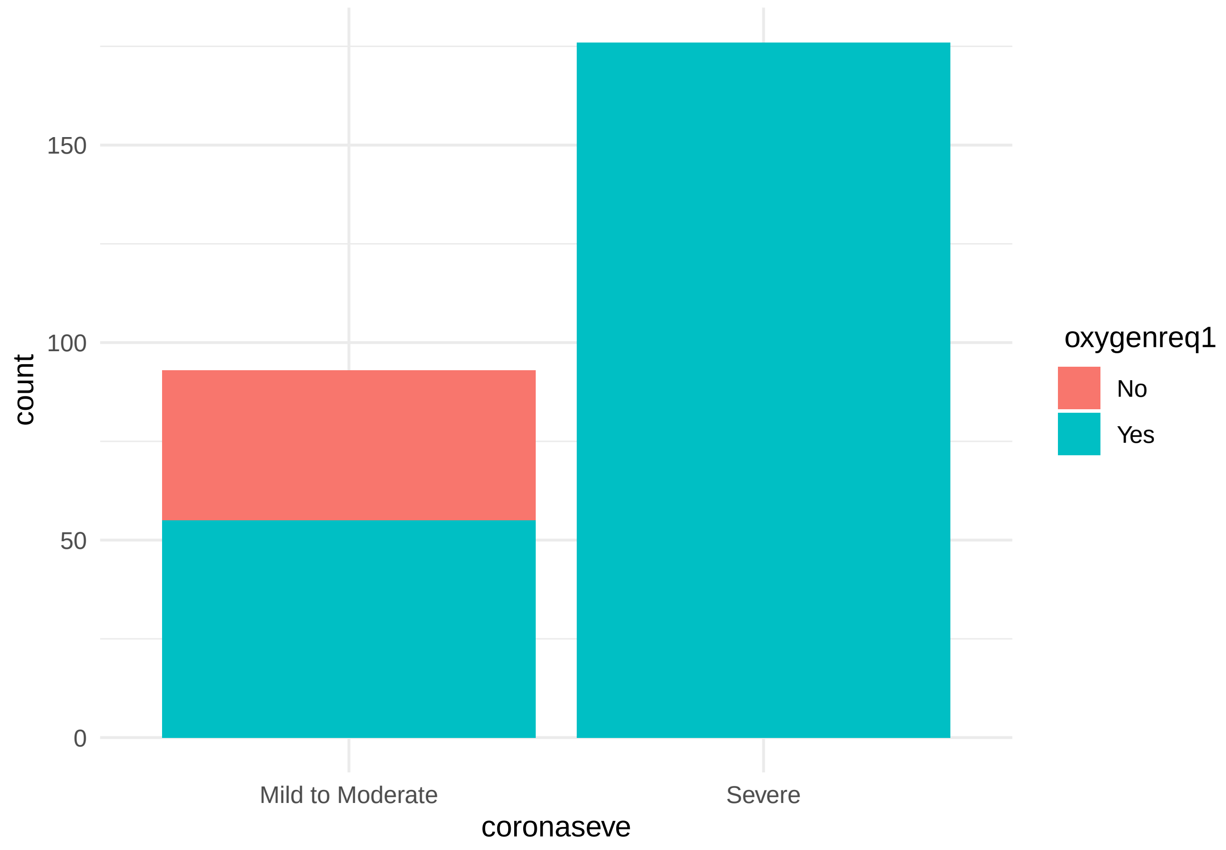
```r
# Create a bar plot using ggplot between coronaseve &  patchside
ggplot(data_seve_conf, aes(x = coronaseve, fill =    patchside)) +
     geom_bar() +
     labs(x = "coronaseve", fill = "patchside") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No lobe", "Bilateral lung lobe","Right", "Left")) +
     theme_minimal()
```



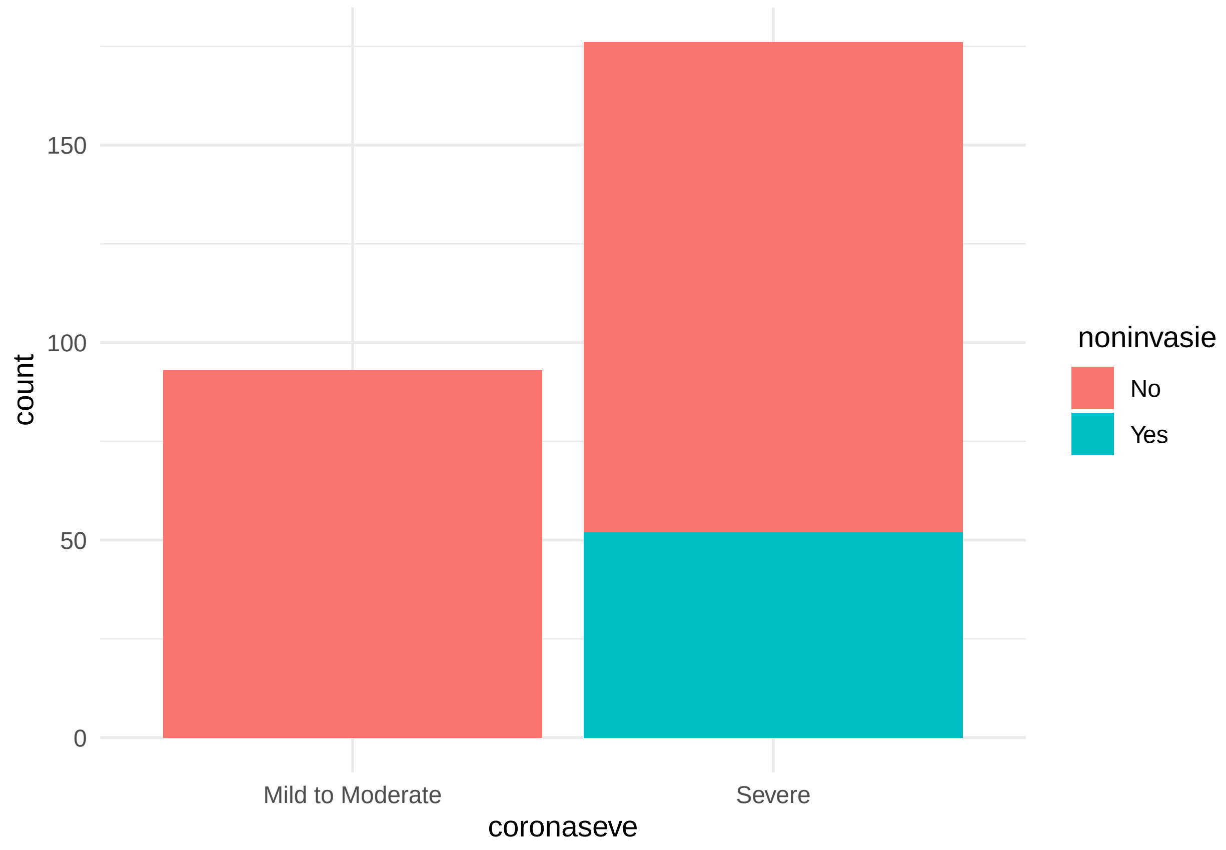
```r
# Create a bar plot using ggplot between coronaseve &  patchlobel
ggplot(data_seve_conf, aes(x = coronaseve, fill =    patchlobel)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " patchlobel") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Lower","Middle","Upper", "Lower+Middle" ,"Multi")) +
     theme_minimal()
```



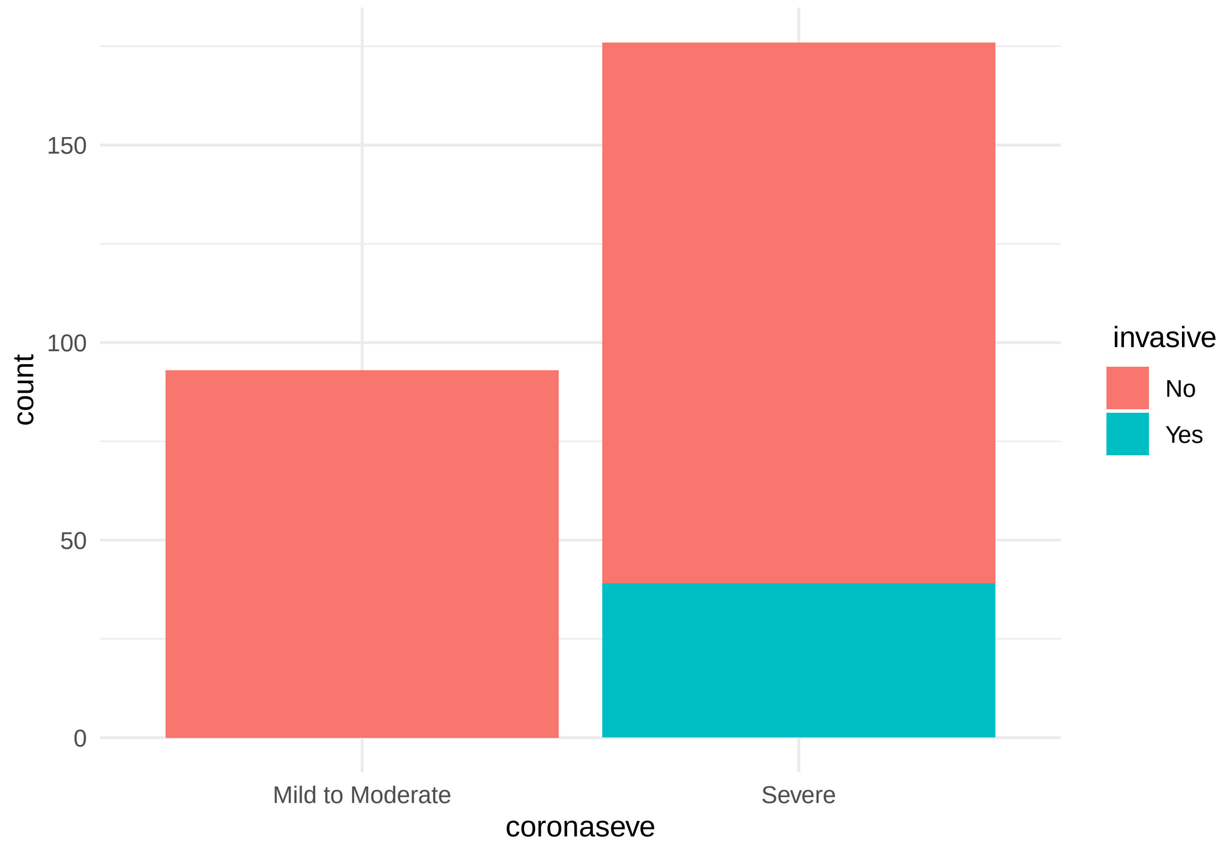
```
# Create a bar plot using ggplot between coronaseve &  oxygenreq1
ggplot(data_seve_conf, aes(x = coronaseve, fill =    oxygenreq1)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " oxygenreq1") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



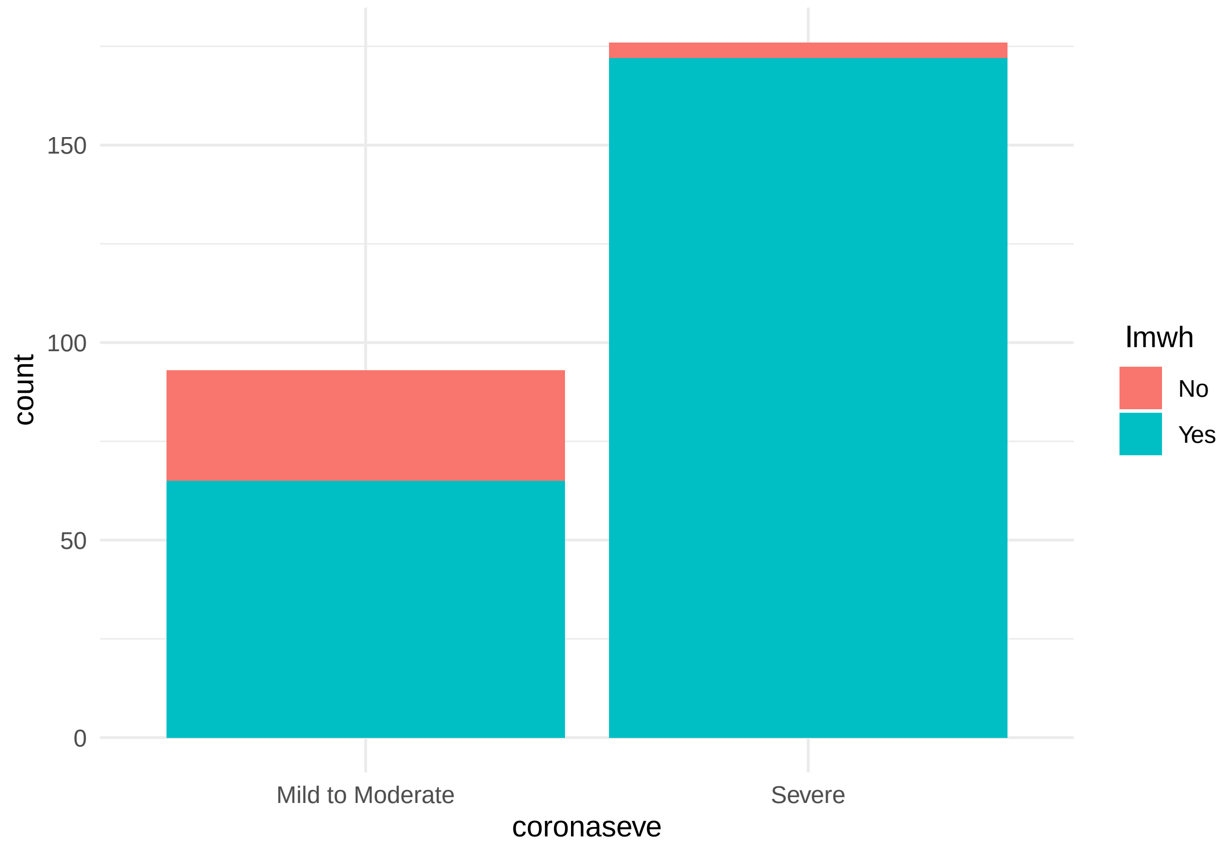
```
# Create a bar plot using ggplot between coronaseve &  noninvasie
ggplot(data_seve_conf, aes(x = coronaseve, fill =    noninvasie)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " noninvasie") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



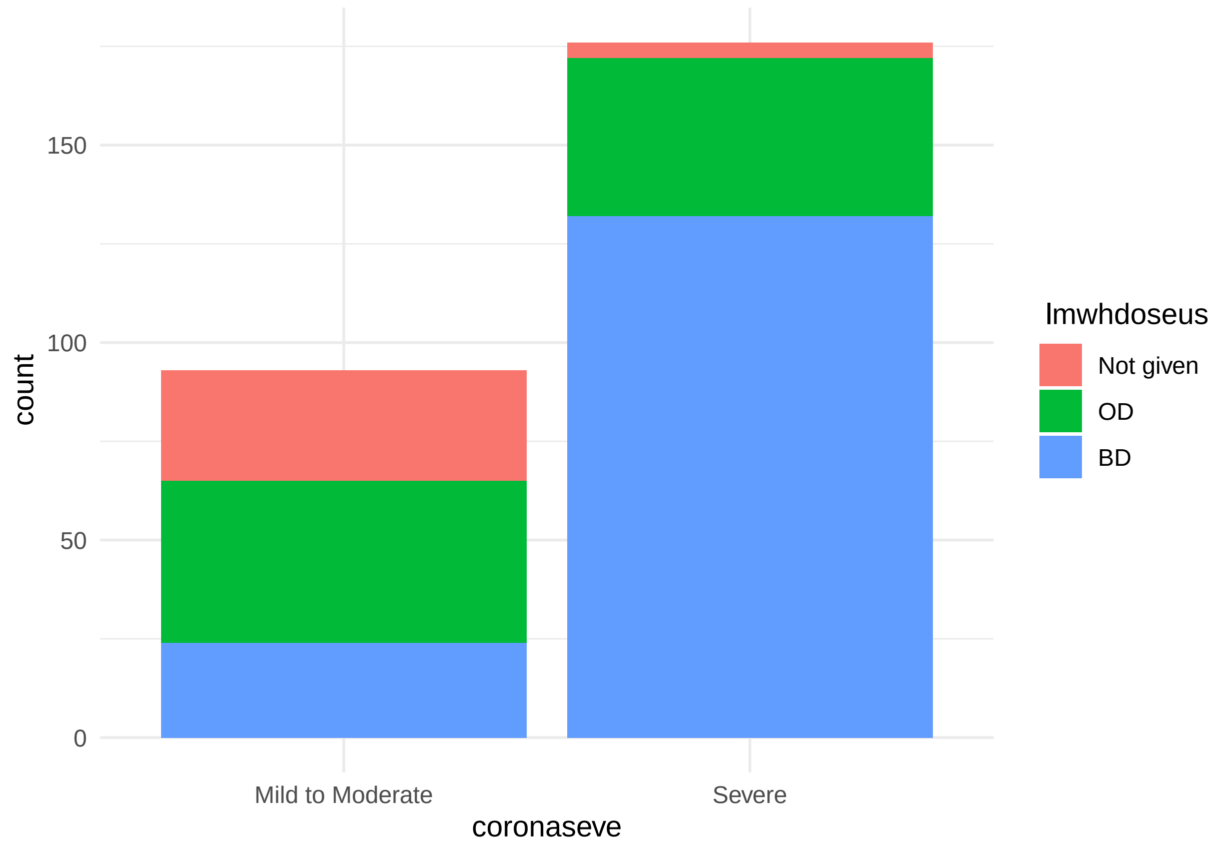
```r
# Create a bar plot using ggplot between coronaseve &  invasive
ggplot(data_seve_conf, aes(x = coronaseve, fill =    invasive)) +
     geom_bar() +
     labs(x = "coronaseve", fill = " invasive") +
     scale_x_discrete(labels =c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



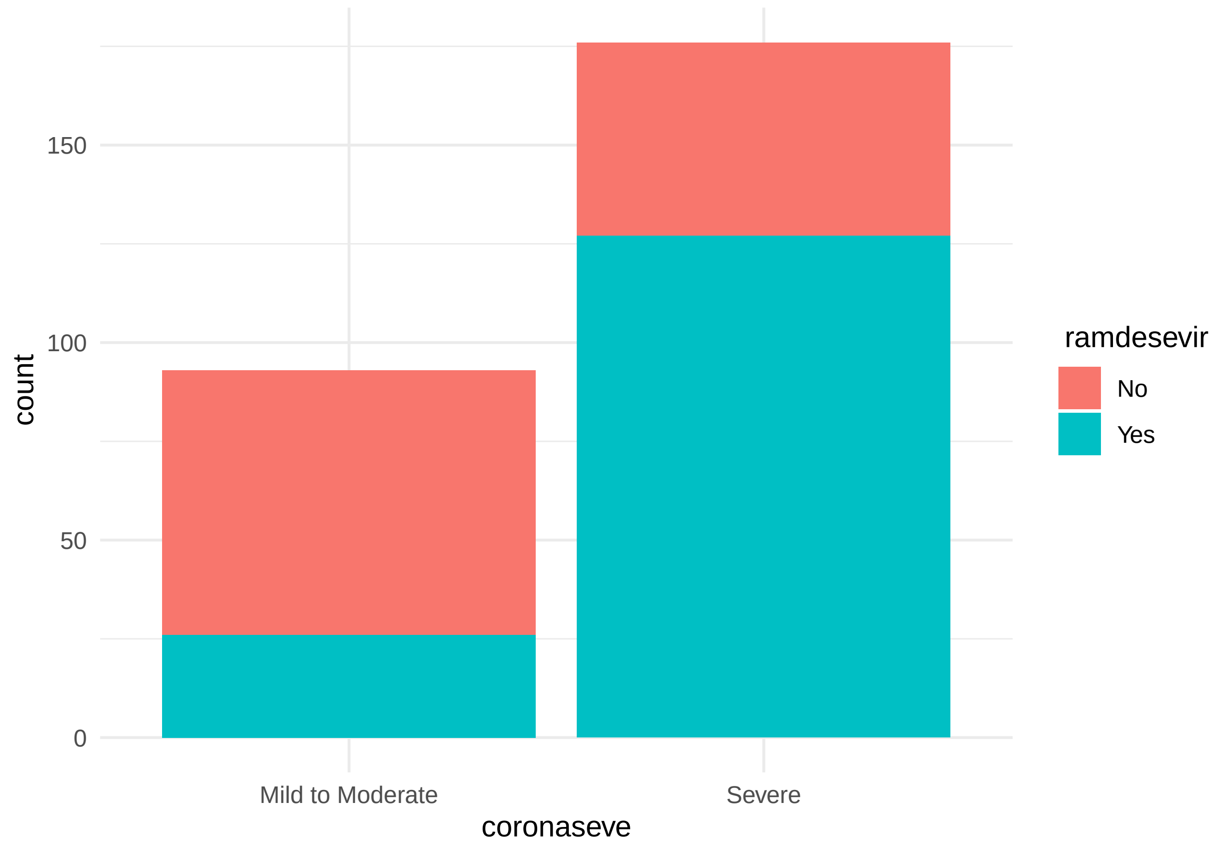
```r
# Create a bar plot using ggplot between coronaseve &  lmwh
ggplot(data_seve_conf, aes(x = coronaseve, fill =    lmwh )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " lmwh ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



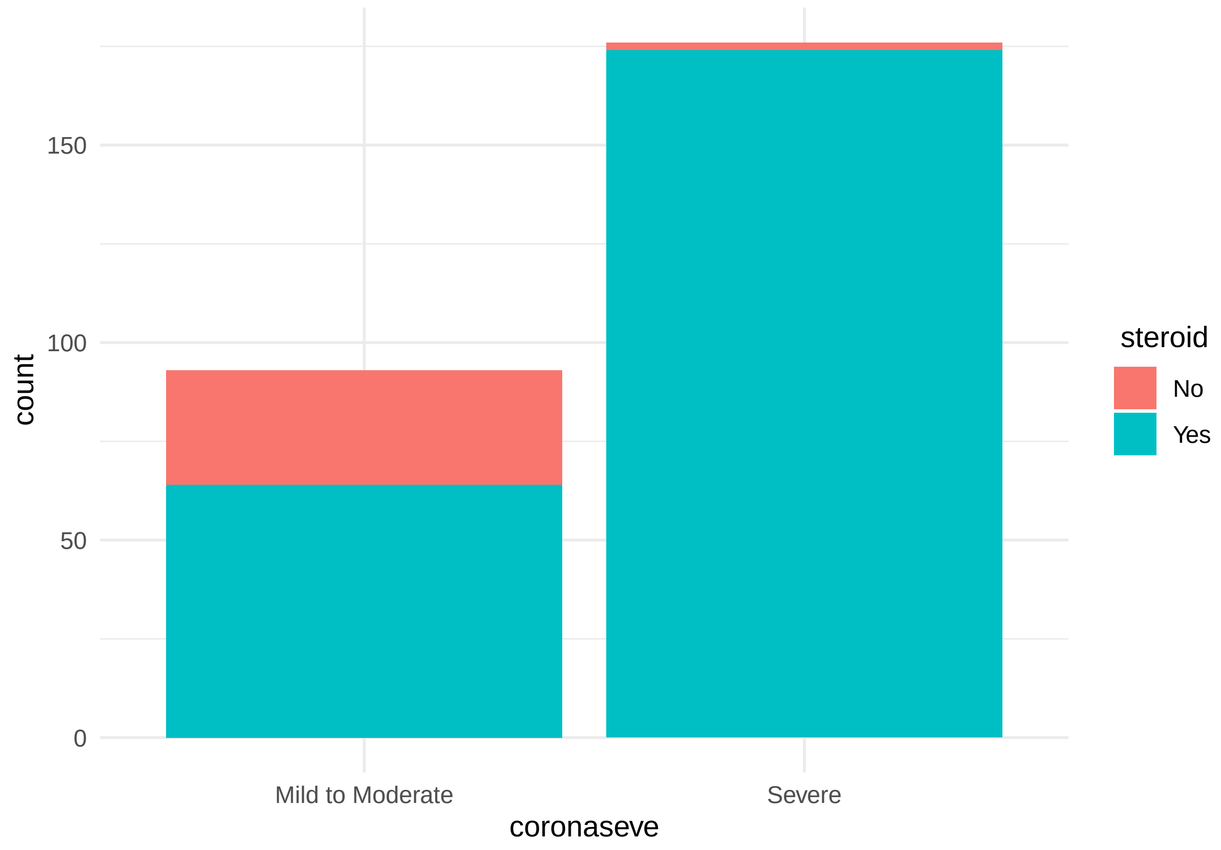
```r
# Create a bar plot using ggplot between coronaseve &  lmwhdoseus
ggplot(data_seve_conf, aes(x = coronaseve, fill =    lmwhdoseus )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " lmwhdoseus ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("Not given", "OD", "BD")) +
     theme_minimal()
```



```r
# Create a bar plot using ggplot between coronaseve &  ramdesevir
ggplot(data_seve_conf, aes(x = coronaseve, fill =    ramdesevir )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " ramdesevir ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



```
# Create a bar plot using ggplot between coronaseve &  steroid
ggplot(data_seve_conf, aes(x = coronaseve, fill =    steroid )) +
     geom_bar() +
     labs(x = "coronaseve", fill = " steroid ") +
     scale_x_discrete(labels = c("Mild to Moderate", "Severe")) +
     scale_fill_discrete(labels = c("No", "Yes")) +
     theme_minimal()
```



```r
#error metrics -- Confusion Matrix
err_metric=function(CM)
{
  TN =CM[2,2]
  TP =CM[1,1]
  FP =CM[2,1]
  FN =CM[1,2]
  Sensitivity = TP /(TP+FN)
  precision =(TP)/(TP+FP)
  recall_score =(FP)/(FP+TN)
  Specificity =  TN /(FP + TN)
  f1_score=2*((precision*recall_score)/(precision+recall_score))
  accuracy_model  =(TP+TN)/(TP+TN+FP+FN)
  False_positive_rate =(FP)/(FP+TN)
  False_negative_rate =(FN)/(FN+TP)
  print(paste("Precision value of the model: ",round(precision,2)))
  print(paste("Accuracy of the model: ",round(accuracy_model,2)))
  print(paste("Recall value of the model: ",round(recall_score,2)))
  print(paste("False Positive rate of the model: ",round(False_positive_rate,2)))
  print(paste("False Negative rate of the model: ",round(False_negative_rate,2)))
  print(paste("f1 score of the model: ",round(f1_score,2)))
  score_vec <- c(round(accuracy_model,2) , round(precision,2) ,
               round(Sensitivity , 2) ,  round(Specificity ,2))
  return(score_vec)
}
```

Data Partition

```r
library(caTools)
set.seed(9973)


sample <- sample.split(data_seve$coronaseve , SplitRatio = 0.6)
train_seve  <- subset(data_seve , sample == TRUE)
test_seve   <- subset(data_seve , sample == FALSE)
```

Random Forest Models

```r
#considering all variabls
set.seed(252525)
rf_seve92 <- randomForest(coronaseve~., data = train_seve)
rf_seve92
```

```
##
## Call:
##  randomForest(formula = coronaseve ~ ., data = train_seve)
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 4.32%
## Confusion matrix:
##    0   1 class.error
## 0 51   5  0.08928571
## 1  2 104  0.01886792
```

```r
# Prediction & Confusion Matrix - Test with all 92 variables
p <- predict(rf_seve92, test_seve)
confusionMatrix(p, test_seve$coronaseve)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32  1
##          1  5 69
##
##                Accuracy : 0.9439
##                  95% CI : (0.8819, 0.9791)
##     No Information Rate : 0.6542
##     P-Value [Acc > NIR] : 8.456e-13
##
##                   Kappa : 0.8728
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8649
##             Specificity : 0.9857
##          Pos Pred Value : 0.9697
##          Neg Pred Value : 0.9324
##              Prevalence : 0.3458
##          Detection Rate : 0.2991
##    Detection Prevalence : 0.3084
##       Balanced Accuracy : 0.9253
##
##        'Positive' Class : 0
##
```

```
CM_rf_seve92 <- as.matrix(confusionMatrix(p, test_seve$coronaseve))
CM_rf_seve92
```

```
##    0  1
## 0 32  1
## 1  5 69
```

```
err_metric(CM_rf_seve92)
```

```
## [1] "Precision value of the model:  0.86"
## [1] "Accuracy of the model:  0.94"
## [1] "Recall value of the model:  0.07"
## [1] "False Positive rate of the model:  0.07"
## [1] "False Negative rate of the model:  0.03"
## [1] "f1 score of the model:  0.13"

## [1] 0.94 0.86 0.97 0.93
```

```
score_vec_rf_seve92 <- err_metric(CM_rf_seve92)
```

```
## [1] "Precision value of the model:  0.86"
## [1] "Accuracy of the model:  0.94"
## [1] "Recall value of the model:  0.07"
## [1] "False Positive rate of the model:  0.07"
## [1] "False Negative rate of the model:  0.03"
## [1] "f1 score of the model:  0.13"
```

```
score_vec_rf_seve92
```

```
## [1] 0.94 0.86 0.97 0.93
```

getNonRejectedFormula() gives us the formula that should be used while building model

```
# formula_nonmrejected_seve contains the formula with non rejected variables
formula_nonrejected_seve <- getNonRejectedFormula(boruta_seve)
formula_nonrejected_seve
```

```
## coronaseve ~ age + dyspnoea + respirator + spo2 + resrate + bun +
##     abgph + abgspo2 + arrhythmia + ventricula + vpc + oxygenrequ +
##     cxrpneu + patchprsen + patchside + patchlobel + oxygenreq1 +
##     noninvasie + invasive + arrythmia + lmwh + lmwhdoseus + ramdesevir +
##     steroid
## <environment: 0x000001c6cf7a5228>
```

Random Forest model with 23 non rejected predictors

```r
rf_seve23<- randomForest(formula_nonrejected_seve, data=train_seve)

rf_seve23
```

```
##
## Call:
##  randomForest(formula = formula_nonrejected_seve, data = train_seve)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 2.47%
## Confusion matrix:
##    0   1 class.error
## 0 54   2  0.03571429
## 1  2 104  0.01886792
```

```r
# Prediction & Confusion Matrix - Test with 23 variables only
p <- predict(rf_seve23, test_seve)
confusionMatrix(p, test_seve$coronaseve)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 37  2
##          1  0 68
##
##                Accuracy : 0.9813
##                  95% CI : (0.9341, 0.9977)
##     No Information Rate : 0.6542
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9592
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9714
##          Pos Pred Value : 0.9487
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3458
##          Detection Rate : 0.3458
##    Detection Prevalence : 0.3645
##       Balanced Accuracy : 0.9857
##
##        'Positive' Class : 0
##
```

```
CM_rf_seve23  <- as.matrix(confusionMatrix(p, test_seve$coronaseve))
err_metric(CM_rf_seve23)
```

```
## [1] "Precision value of the model:  1"
## [1] "Accuracy of the model:  0.98"
## [1] "Recall value of the model:  0"
## [1] "False Positive rate of the model:  0"
## [1] "False Negative rate of the model:  0.05"
## [1] "f1 score of the model:  0"
```

```
## [1] 0.98 1.00 0.95 1.00
```

```
score_vec_rf_seve23 <- err_metric(CM_rf_seve23)
```

```
## [1] "Precision value of the model:  1"
## [1] "Accuracy of the model:  0.98"
## [1] "Recall value of the model:  0"
## [1] "False Positive rate of the model:  0"
## [1] "False Negative rate of the model:  0.05"
## [1] "f1 score of the model:  0"
```

```
score_vec_rf_seve23
```

```
## [1] 0.98 1.00 0.95 1.00
```

getConfirmedFormula() gives us the formula that should be used while building model with confirmed variables i.e. variables that was not rejected by *Boruta()*

```
formula_confirmed_seve <- getConfirmedFormula(boruta_seve)
formula_confirmed_seve
```

```
## coronaseve ~ age + dyspnoea + respirator + spo2 + resrate + bun +
##     abgph + abgspo2 + ventricula + vpc + oxygenrequ + cxrpneu +
##     patchprsen + patchside + patchlobel + oxygenreq1 + noninvasie +
##     invasive + lmwh + lmwhdoseus + ramdesevir + steroid
## <environment: 0x000001c6ca97a368>
```

model with 21 confirmed important predictors

```
rf_seve21<- randomForest(formula_confirmed_seve, data=train_seve)

rf_seve21
```

```
##
## Call:
##  randomForest(formula = formula_confirmed_seve, data = train_seve)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 2.47%
## Confusion matrix:
##    0   1 class.error
## 0 54   2  0.03571429
## 1  2 104  0.01886792
```

```
# Prediction & Confusion Matrix - Test with 21 important variables only
p <- predict(rf_seve21, test_seve)
confusionMatrix(p, test_seve$coronaseve)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 37  3
##          1  0 67
##
##                Accuracy : 0.972
##                  95% CI : (0.9202, 0.9942)
##     No Information Rate : 0.6542
##     P-Value [Acc > NIR] : 5.917e-16
##
##                   Kappa : 0.9392
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9571
##          Pos Pred Value : 0.9250
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3458
##          Detection Rate : 0.3458
##    Detection Prevalence : 0.3738
##       Balanced Accuracy : 0.9786
##
##        'Positive' Class : 0
##
```

```
CM_rf_seve21 <- as.matrix(confusionMatrix(p, test_seve$coronaseve))
err_metric(CM_rf_seve21)
```

```
## [1] "Precision value of the model:  1"
## [1] "Accuracy of the model:  0.97"
## [1] "Recall value of the model:  0"
## [1] "False Positive rate of the model:  0"
## [1] "False Negative rate of the model:  0.07"
## [1] "f1 score of the model:  0"
```

```
## [1] 0.97 1.00 0.92 1.00
```

```
score_vec_rf_seve21 <- err_metric(CM_rf_seve21)
```

```
## [1] "Precision value of the model:  1"
## [1] "Accuracy of the model:  0.97"
## [1] "Recall value of the model:  0"
## [1] "False Positive rate of the model:  0"
## [1] "False Negative rate of the model:  0.07"
## [1] "f1 score of the model:  0"
```

```
score_vec_rf_seve21
```

```
## [1] 0.97 1.00 0.92 1.00
```

SVM for classification of Severity

```
# library(e1071) # library required for svm

svmfit_seve <- svm(formula_confirmed_seve , data = train_seve,
                   kernal = "linear" )
summary(svmfit_seve)
```

```
##
## Call:
## svm(formula = formula_confirmed_seve, data = train_seve, kernal = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  75
##
##  ( 36 39 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
set.seed(9973)
# tune() performs cross validation
tune.out <- tune(svm , formula_confirmed_seve , data = train_seve,
                 kernal = "linear",
                 ranges = list(
                   cost = c(  1.8, 2, 3 ,3.5)

                 ))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     3
##
## - best performance: 0.1047794
##
## - Detailed performance results:
##   cost     error dispersion
## 1  1.8 0.1110294 0.07073531
## 2  2.0 0.1110294 0.07073531
## 3  3.0 0.1047794 0.07211153
## 4  3.5 0.1110294 0.08722185
```

```
p<- predict(tune.out$best.model, newdata = test_seve)
confusionMatrix(p , test_seve$coronaseve)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 30  2
##          1  7 68
##
##                Accuracy : 0.9159
##                  95% CI : (0.8463, 0.9608)
##     No Information Rate : 0.6542
##     P-Value [Acc > NIR] : 2.653e-10
##
##                   Kappa : 0.808
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.8108
##             Specificity : 0.9714
##          Pos Pred Value : 0.9375
##          Neg Pred Value : 0.9067
##              Prevalence : 0.3458
##          Detection Rate : 0.2804
##    Detection Prevalence : 0.2991
##       Balanced Accuracy : 0.8911
##
##        'Positive' Class : 0
##
```

```
CM_svmfit_seve <- as.matrix(confusionMatrix(p , test_seve$coronaseve))
err_metric((CM_svmfit_seve))
```

```
## [1] "Precision value of the model:  0.81"
## [1] "Accuracy of the model:  0.92"
## [1] "Recall value of the model:  0.09"
## [1] "False Positive rate of the model:  0.09"
## [1] "False Negative rate of the model:  0.06"
## [1] "f1 score of the model:  0.17"
```

```
## [1] 0.92 0.81 0.94 0.91
```

```
score_vec_svmfit_seve <- err_metric((CM_svmfit_seve))
```

```
## [1] "Precision value of the model:  0.81"
## [1] "Accuracy of the model:  0.92"
## [1] "Recall value of the model:  0.09"
## [1] "False Positive rate of the model:  0.09"
## [1] "False Negative rate of the model:  0.06"
## [1] "f1 score of the model:  0.17"
```

```
score_vec_svmfit_seve
```

```
## [1] 0.92 0.81 0.94 0.91
```

Death

Some variable that are not apt to include for the study of death Out of patient whose death occurred, only 6 death are noncovid or probably covid. So dropping those variables can be an option… Assuming all death are because of Covid only.

As we want to predict survival of patient so it's good to not keep *discharge* or *los* ,length of stay, as predictors.

Let's build model without them.

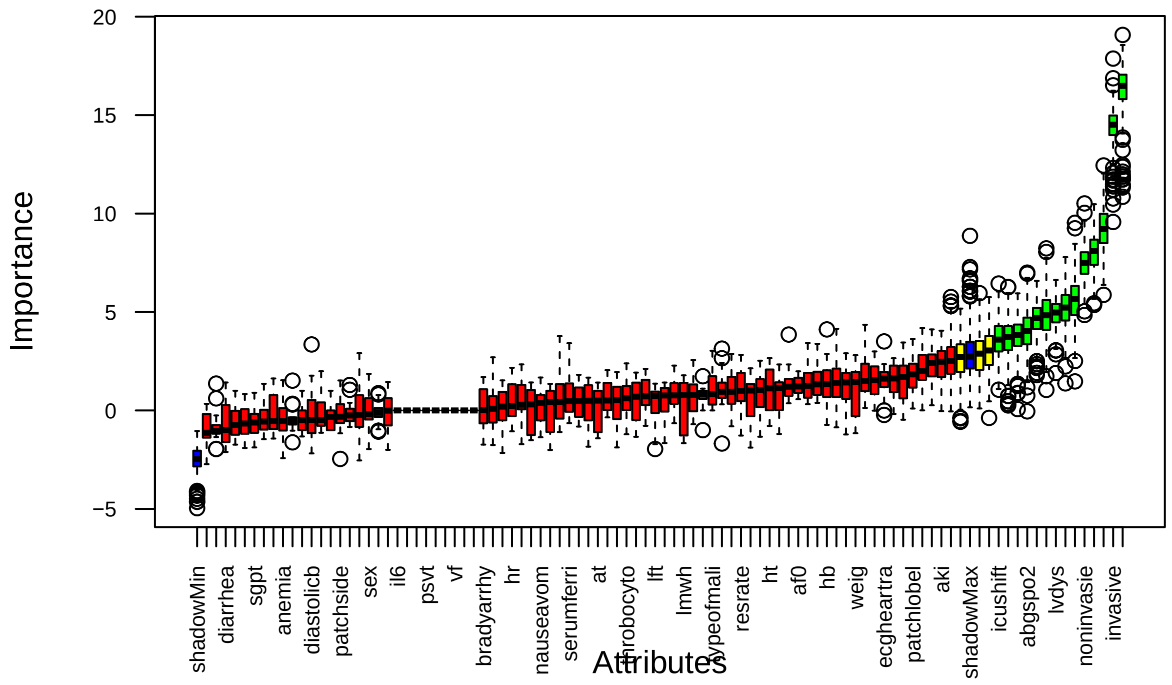
```
remove_col_unnecessary<-c( "name","coviddeath","noncovidde","procovidde",
                           "discharge" , "los")

data_death<-data_non_null[, !names(data_non_null) %in%  remove_col_unnecessary]

#dim(data_death)

#str(data_death)
```

Again we need to apply **Boruta()** to find out the features that are important to find out the class of variable death

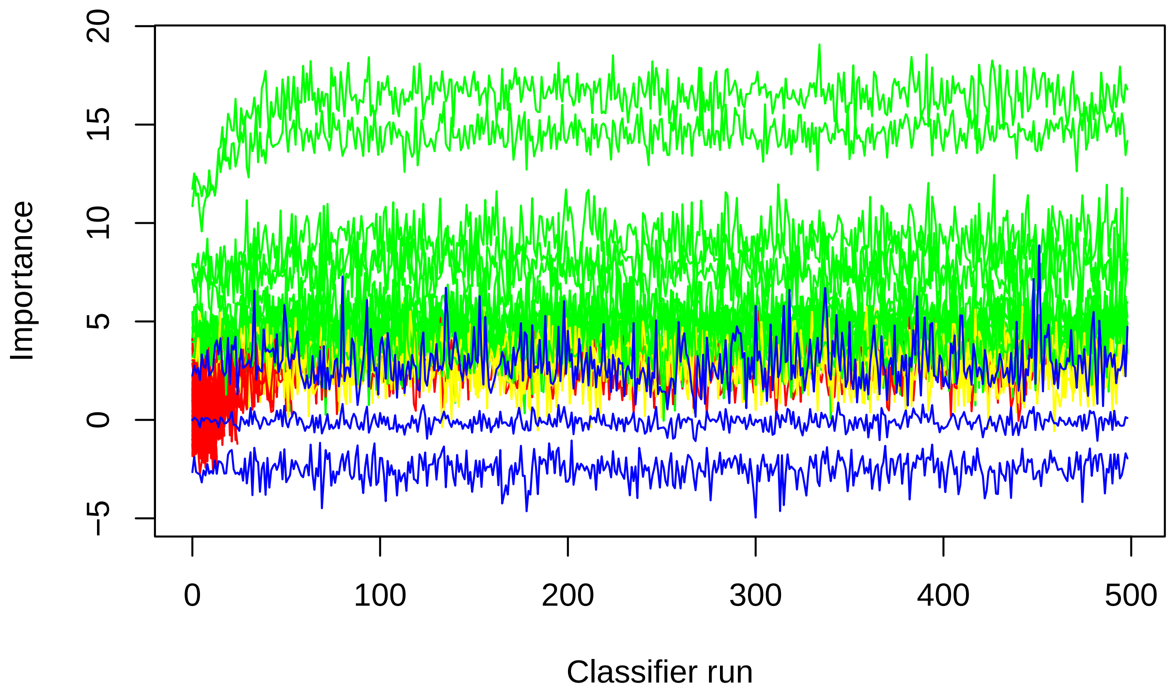
```
# Feature Selection
set.seed(111)
boruta <- Boruta(death ~ .,
                 data = data_death ,
                 maxRuns = 500)

# The above command applies the Boruta feature selection algorithm to
# the dataset data_death. The formula death ~ . specifies that
# the variable death is the outcome variable, and . indicates that all other
# variables in the dataset are considered as potential predictors. The doTrace
# parameter controls the level of verbosity during the Boruta analysis, with
# a value of 2 indicating more detailed output. The maxRuns parameter
# specifies the maximum number of iterations to run the algorithm. The result
# of the Boruta analysis is stored in the boruta object.

print(boruta)
```

```
## Boruta performed 499 iterations in 34.59672 secs.
##  14 attributes confirmed important: abgacidos, abgph, abgspo2, bun,
## coronaseve and 9 more;
##  78 attributes confirmed unimportant: af0, afl, age, aki, anemia and 73
## more;
##  3 tentative attributes left: arrhythmia, lqtinter, tlc;
```

```
#The above command prints the summary or information about the boruta object.
# It displays the variables considered in the Boruta analysis, their
# importance scores, and the final decision on whether they are selected
# as important predictors or not. This information helps assess the relevance
# of variables in predicting the outcome variable (death) based on the
# Boruta analysis.

plot(boruta, las = 2, cex.axis = 0.7)
```



```
# Above command is used to create a plot of the Boruta analysis results.
# The boruta object is passed as the argument, and additional parameters las
# and cex.axis are used to modify the appearance of the plot. las controls
# the orientation of the axis labels, and cex.axis controls the size of the
# axis labels. By customizing these parameters, you can enhance the
# readability of the plot and visualize the importance of variables determined
# by Boruta.

plotImpHistory(boruta)
```



```
#Above command is used to create a plot of the variable importance history
# during the Boruta analysis. The boruta object is passed as the argument,
# and the function generates a plot that shows the change in variable
# importance over iterations. This plot helps understand how the importance
# of variables evolves during the Boruta feature selection process and can
# assist in determining the optimal set of variables to include in the
# final model.

# Tentative Fix
bor <- TentativeRoughFix(boruta)
print(bor)
```

```
## Boruta performed 499 iterations in 34.59672 secs.
## Tentatives roughfixed over the last 499 iterations.
##  16 attributes confirmed important: abgacidos, abgph, abgspo2, bun,
## coronaseve and 11 more;
##  79 attributes confirmed unimportant: af0, afl, age, aki, anemia and 74
## more;
```

```r
att_death <- attStats(boruta)
```

Data Partition

```r
set.seed(9973)
sample <- sample.split(data_death$death , SplitRatio = 0.6)
train_death  <- subset(data_death , sample == TRUE)
test_death   <- subset(data_death , sample == FALSE)
```

Random Forest Models

```r
#considering all 95 variabls
set.seed(252525)
rf_death95 <- randomForest(death~., data = train_death)
rf_death95
```

```
##
## Call:
##  randomForest(formula = death ~ ., data = train_death)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 16.15%
## Confusion matrix:
##     0  1 class.error
## 0 111 10  0.08264463
## 1  16 24  0.40000000
```

```r
# Prediction & Confusion Matrix - Test with all 95 variables
p <- predict(rf_death95, test_death)
confusionMatrix(p, test_death$death)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 72 12
##          1  9 15
##
##                Accuracy : 0.8056
##                  95% CI : (0.7183, 0.8754)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : 0.1088
##
##                   Kappa : 0.4615
##
##  Mcnemar's Test P-Value : 0.6625
```

```
##
##             Sensitivity : 0.8889
##             Specificity : 0.5556
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.6250
##              Prevalence : 0.7500
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7778
##       Balanced Accuracy : 0.7222
##
##        'Positive' Class : 0
##
```

```
CM_rf_death95 <- as.matrix(confusionMatrix(p, test_death$death))
err_metric(CM_rf_death95)
```

```
## [1] "Precision value of the model:  0.89"
## [1] "Accuracy of the model:  0.81"
## [1] "Recall value of the model:  0.38"
## [1] "False Positive rate of the model:  0.38"
## [1] "False Negative rate of the model:  0.14"
## [1] "f1 score of the model:  0.53"

## [1] 0.81 0.89 0.86 0.62
```

```
score_vec_rf_death95 <- err_metric(CM_rf_death95)
```

```
## [1] "Precision value of the model:  0.89"
## [1] "Accuracy of the model:  0.81"
## [1] "Recall value of the model:  0.38"
## [1] "False Positive rate of the model:  0.38"
## [1] "False Negative rate of the model:  0.14"
## [1] "f1 score of the model:  0.53"
```

```
score_vec_rf_death95
```

```
## [1] 0.81 0.89 0.86 0.62
```

formula_nonrejected_death

```
formula_nonrejected_death <- getNonRejectedFormula(boruta)
formula_nonrejected_death
```

```
## death ~ coronaseve + lvdys + liverillne + spo2 + tlc + bun +
##     creatinine + abgph + abgacidos + abgspo2 + correctedq + lqtinter +
##     arrhythmia + ventilator + noninvasie + invasive + icushift
## <environment: 0x000001c6d24f6498>
```

model with 17 non rejected predictors

```r
rf_death17<- randomForest(formula_nonrejected_death, data=train_death)

rf_death17
```

```
##
## Call:
##  randomForest(formula = formula_nonrejected_death, data = train_death)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 13.66%
## Confusion matrix:
##     0  1 class.error
## 0 111 10  0.08264463
## 1  12 28  0.30000000
```

```r
# Prediction & Confusion Matrix - Test with 17 variables only
p <- predict(rf_death17, test_death)
confusionMatrix(p, test_death$death)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 69 11
##          1 12 16
##
##                Accuracy : 0.787
##                  95% CI : (0.6978, 0.86)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : 0.2206
##
##                   Kappa : 0.439
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8519
##             Specificity : 0.5926
##          Pos Pred Value : 0.8625
##          Neg Pred Value : 0.5714
##              Prevalence : 0.7500
##          Detection Rate : 0.6389
##    Detection Prevalence : 0.7407
##       Balanced Accuracy : 0.7222
##
##        'Positive' Class : 0
##
```

```
CM_rf_death17 <- as.matrix(confusionMatrix(p, test_death$death))
err_metric(CM_rf_death17)
```

```
## [1] "Precision value of the model:  0.85"
## [1] "Accuracy of the model:  0.79"
## [1] "Recall value of the model:  0.43"
## [1] "False Positive rate of the model:  0.43"
## [1] "False Negative rate of the model:  0.14"
## [1] "f1 score of the model:  0.57"
```

```
## [1] 0.79 0.85 0.86 0.57
```

```
score_vec_rf_death17 <- err_metric(CM_rf_death17)
```

```
## [1] "Precision value of the model:  0.85"
## [1] "Accuracy of the model:  0.79"
## [1] "Recall value of the model:  0.43"
## [1] "False Positive rate of the model:  0.43"
## [1] "False Negative rate of the model:  0.14"
## [1] "f1 score of the model:  0.57"
```

```
score_vec_rf_death17
```

```
## [1] 0.79 0.85 0.86 0.57
```

formula_confirmed death

```
formula_confirmed_death <- getConfirmedFormula(boruta)
formula_confirmed_death
```

```
## death ~ coronaseve + lvdys + liverillne + spo2 + bun + creatinine +
##     abgph + abgacidos + abgspo2 + correctedq + ventilator + noninvasie +
##     invasive + icushift
## <environment: 0x000001c6d48c5c38>
```

model with confirmed 15 important predictors

```
rf_death15<- randomForest(formula_confirmed_death, data=train_death)

rf_death15
```

```
##
## Call:
##  randomForest(formula = formula_confirmed_death, data = train_death)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 12.42%
## Confusion matrix:
##     0  1 class.error
## 0 112  9  0.07438017
## 1  11 29  0.27500000
```

```
# Prediction & Confusion Matrix - Test with 15 important variables only
p <- predict(rf_death15, test_death)
confusionMatrix(p, test_death$death)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 68 14
##          1 13 13
##
##                Accuracy : 0.75
##                  95% CI : (0.6575, 0.8283)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : 0.5515
##
##                   Kappa : 0.325
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8395
##             Specificity : 0.4815
##          Pos Pred Value : 0.8293
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7500
##          Detection Rate : 0.6296
##    Detection Prevalence : 0.7593
##       Balanced Accuracy : 0.6605
##
##        'Positive' Class : 0
##
```

```
CM_rf_death15 <- as.matrix(confusionMatrix(p, test_death$death))
err_metric(CM_rf_death15)
```

```
## [1] "Precision value of the model:  0.84"
## [1] "Accuracy of the model:  0.75"
## [1] "Recall value of the model:  0.5"
## [1] "False Positive rate of the model:  0.5"
## [1] "False Negative rate of the model:  0.17"
## [1] "f1 score of the model:  0.63"
```

```
## [1] 0.75 0.84 0.83 0.50
```

```
score_vec_rf_death15 <- err_metric(CM_rf_death15)
```

```
## [1] "Precision value of the model:  0.84"
## [1] "Accuracy of the model:  0.75"
## [1] "Recall value of the model:  0.5"
## [1] "False Positive rate of the model:  0.5"
## [1] "False Negative rate of the model:  0.17"
## [1] "f1 score of the model:  0.63"
```

```
score_vec_rf_death15
```

```
## [1] 0.75 0.84 0.83 0.50
```

SVM

```
svmfit_death <- svm(formula_confirmed_death , data = train_death,
                    kernal = "linear" )
summary(svmfit_death)
```

```
##
## Call:
## svm(formula = formula_confirmed_death, data = train_death, kernal = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  76
##
##  ( 41 35 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
set.seed(9973)
# tune() performs cross-validation
tune.out <- tune(svm , formula_confirmed_death , data = train_death,
                 kernal = "linear",
                 ranges = list(
                   cost = c(0.5 , 1, 1.5 , 2, 3, 10)

                 ))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1238971
##
## - Detailed performance results:
##   cost     error dispersion
## 1  0.5 0.1613971 0.11465311
## 2  1.0 0.1488971 0.11451942
## 3  1.5 0.1613971 0.10681397
## 4  2.0 0.1551471 0.09849465
## 5  3.0 0.1301471 0.08490799
## 6 10.0 0.1238971 0.08248286
```

```
p<- predict(tune.out$best.model, newdata = test_death)
confusionMatrix(p , test_death$death)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 73 12
##          1  8 15
##
##                Accuracy : 0.8148
##                  95% CI : (0.7286, 0.8831)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : 0.07101
##
##                   Kappa : 0.4805
##
##  Mcnemar's Test P-Value : 0.50233
##
##             Sensitivity : 0.9012
##             Specificity : 0.5556
##          Pos Pred Value : 0.8588
##          Neg Pred Value : 0.6522
##              Prevalence : 0.7500
##          Detection Rate : 0.6759
##    Detection Prevalence : 0.7870
##       Balanced Accuracy : 0.7284
##
##        'Positive' Class : 0
##
```

```
CM_svmfit_death <- as.matrix(confusionMatrix(p , test_death$death))
err_metric(CM_svmfit_death)
```

```
## [1] "Precision value of the model:  0.9"
## [1] "Accuracy of the model:  0.81"
## [1] "Recall value of the model:  0.35"
## [1] "False Positive rate of the model:  0.35"
## [1] "False Negative rate of the model:  0.14"
## [1] "f1 score of the model:  0.5"
```

```
## [1] 0.81 0.90 0.86 0.65
```

```
score_vec_svmfit_death <- err_metric(CM_svmfit_death)
```

```
## [1] "Precision value of the model:  0.9"
## [1] "Accuracy of the model:  0.81"
## [1] "Recall value of the model:  0.35"
## [1] "False Positive rate of the model:  0.35"
## [1] "False Negative rate of the model:  0.14"
## [1] "f1 score of the model:  0.5"
```

```
score_vec_svmfit_death
```

```
## [1] 0.81 0.90 0.86 0.65
```

Comaprision of models

```
df <- rbind(score_vec_rf_seve92 , score_vec_rf_seve23 , score_vec_rf_seve21,
            score_vec_svmfit_seve , score_vec_rf_death95, score_vec_rf_death17,
            score_vec_rf_death15 , score_vec_svmfit_death)
colnames(df) <- c("Accuracy" , "Precision" , "Sensitivity" , "Specificity")
df <- as.data.frame(df)
df
```

```
##                          Accuracy Precision Sensitivity Specificity
## score_vec_rf_seve92          0.94      0.86        0.97        0.93
## score_vec_rf_seve23          0.98      1.00        0.95        1.00
## score_vec_rf_seve21          0.97      1.00        0.92        1.00
## score_vec_svmfit_seve        0.92      0.81        0.94        0.91
## score_vec_rf_death95         0.81      0.89        0.86        0.62
## score_vec_rf_death17         0.79      0.85        0.86        0.57
## score_vec_rf_death15         0.75      0.84        0.83        0.50
## score_vec_svmfit_death       0.81      0.90        0.86        0.65
```

```
end_time <- Sys.time()
execution_time <- end_time - start_time
execution_time
```

```
## Time difference of 1.235875 mins
```