final_result

Abdelrehim Sabri (500998232)

2019-11-25

Install R packages

```
#install.packages("rpart")
#install.packages("caret")
#install.packages("e1071")
#install.packages("randomForest")
#install.packages("corrplot")
#install.packages("caretEnsemble")
#install.packages("lsr")
```

Exploratory Data Analysis

```
options(warm=-1)
library(rsample)  # data splitting

library(lubridate)

library(ggplot2)  # data visualization

library(randomForest)

library(dplyr) # data transformation

#library(rpart)
library(caret) # implementing with caret

library(caretEnsemble)

library(e1071)

library(corrplot)

library(unbalanced)

library(mlbench)

library(pROC)

library(lsr)
```

Includes functions to clean datasets

Read datasets from csv file

```
build_clean_dataset <- function() {
  datasetloc = "C:/Users/abdel/Desktop/Ryerson
```

```
University/capstone/capstone/R/Health_Care_History.csv"
  if (file.exists(datasetloc)) {
    alldata <- read.csv(file=datasetloc, header = T)
  }
  return(alldata)
}
```

Convert the date to age and group them into four groups (0-25, 26-40, 41-50, 50-65, 65+)

```
age <- function(dob, age.day = today(), units = "years", floor = TRUE) {
  calc.age = interval(dob, age.day) / duration(num = 1, units = units)
  if (floor) return(as.integer(floor(calc.age)))

  return(calc.age)
}

get_age_group <- function(a) {
  ifelse(a<25,25, ifelse(a<40, 40, ifelse(a<50,50,65)))
}
```

Group the countries of the patients based on ethnic groups

```
east_europe <- c('Ukraine','Russia','Poland','Czech Republic','Hungary')
west_europe <-
c('Austria','Belgium','France','Germany','Italy','Netherlands','Portugal','Sp
ain','Switzerland')
north_europe <- c('Sweden', 'Finland', 'Denmark')
british <- c('England','Scotland','Ireland')

get_ethnic_group <- function(country) {
  ifelse((country %in% east_europe), 'east_europe',
         ifelse((country %in% west_europe) ,'west_europe',
                ifelse((country %in% north_europe), 'north_europe',
                       ifelse((country %in% british), 'british',
                              country))))
}
```

Read the dataset and remove patient ids from the analysis

```
patients <- build_clean_dataset()

#remove the patient ids from the dataset
patients <- patients[,-1]
str(patients)

## 'data.frame':    2000 obs. of  13 variables:
##  $ gender            : Factor w/ 2 levels "female","male": 1 1 2 2 1 1 1 1
1 2 ...
##  $ dob               : Factor w/ 1877 levels "1923-10-10","1924-03-28",..:
505 1502 1811 545 327 1120 628 1378 631 1176 ...
##  $ zipcode           : int  89136 94105 89127 44101 89136 94105 60612
```

```
43221 89127 43210 ...
##  $ employment_status : Factor w/ 4 levels "employed","retired",..: 2 1 1 2
2 4 2 1 2 1 ...
##  $ education         : Factor w/ 6 levels "bachelors","highschool",..: 1 5
4 1 4 2 5 1 4 2 ...
##  $ marital_status    : Factor w/ 2 levels "married","single": 1 1 1 1 1 1
1 1 2 1 ...
##  $ children          : int  1 4 2 2 3 2 0 2 2 7 ...
##  $ ancestry          : Factor w/ 20 levels "Austria","Belgium",..: 14 18 8
4 1 1 9 10 1 20 ...
##  $ avg_commute       : num  13.4 15.2 23.6 19.6 36.5 ...
##  $ daily_internet_use: num  2.53 6.77 3.63 5 7.75 3.34 6.75 3.01 4.12 3.15
...
##  $ available_vehicles: int  2 2 1 3 1 0 2 3 1 1 ...
##  $ military_service  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1
...
##  $ disease           : Factor w/ 13 levels "Alzheimer disease",..: 8 4 11
10 13 1 9 2 1 7 ...

summary(patients)

##     gender             dob          zipcode       employment_status
##  female: 975   1946-02-22:   3   Min.   :10001   employed  :769
##  male  :1025   1949-09-15:   3   1st Qu.:43221   retired   :955
##                1954-12-31:   3   Median :60612   student   : 21
##                1959-09-22:   3   Mean   :63388   unemployed:255
##                1960-08-01:   3   3rd Qu.:90008
##                1961-04-10:   3   Max.   :94110
##                (Other)   :1982
##       education    marital_status    children            ancestry
##  bachelors :1076   married:1496   Min.   :0.000   Ireland    : 121
##  highschool: 459   single : 504   1st Qu.:1.000   Switzerland: 115
##  highscool :   4                  Median :2.000   Sweden     : 114
##  masters   : 280                  Mean   :2.267   Portugal   : 112
##  phd/md    : 169                  3rd Qu.:3.000   Belgium    : 109
##  phD/MD    :  12                  Max.   :7.000   Germany    : 106
##                                                   (Other)    :1323
##   avg_commute    daily_internet_use available_vehicles military_service
##  Min.   :-2.47   Min.   :1.010      Min.   :0.000      no :1817
##  1st Qu.:23.46   1st Qu.:4.020      1st Qu.:1.000      yes: 183
##  Median :30.32   Median :5.010      Median :2.000
##  Mean   :30.38   Mean   :4.993      Mean   :1.746
##  3rd Qu.:37.13   3rd Qu.:5.973      3rd Qu.:3.000
##  Max.   :63.73   Max.   :8.820      Max.   :4.000
##
##               disease
##  Alzheimer disease:339
##  hypertension     :298
##  skin cancer      :233
##  kidney disease   :185
```

```
##  prostate cancer  :180
##  breast cancer    :145
##  (Other)          :620
```

From the summary, there are no NaN nor missing data in the dataset.

Fix the education column values by fixing the misspelled words

```
patients$education <- ifelse(patients$education == 'highscool',
as.character('highschool'), as.character(patients$education))
patients$education <- ifelse(as.factor(patients$education) == 'phD/MD',
as.character('phd/md'), as.character(patients$education))
patients$education <- as.factor(patients$education)
```

Group the ancestry countries to ethnic groups

```
patients$ancestry <- as.factor(get_ethnic_group(patients$ancestry))
```

Convert the date of birth into age and group them into 25 40 50 65

```
patients$age <- age(patients$dob)
patients$age <- get_age_group(age(patients$dob))
```

For the analysis purposes, move each disease to separate column with binary values, where 0: patient does not has the disease and 1: patient has the disease

```
get_binary_value <- function(value, compare_to) {
  ifelse(value==compare_to,1,0)
}
patients$prostate_cancer <- get_binary_value(patients$disease,'prostate
cancer')
patients$skin_cancer <- get_binary_value(patients$disease,'skin cancer')
patients$breast_cancer <- get_binary_value(patients$disease,'breast cancer')
patients$hiv_aids <- get_binary_value(patients$disease,'HIV/AIDS')
patients$diabetes <- get_binary_value(patients$disease,'diabetes')
patients$heart_disease <- get_binary_value(patients$disease,'heart disease')
patients$hypertension <- get_binary_value(patients$disease,'hypertension')
patients$endometriosis <- get_binary_value(patients$disease,'endometriosis')
patients$multiple_sclerosis <- get_binary_value(patients$disease,'multiple
sclerosis')
patients$schizophrenia <- get_binary_value(patients$disease,'schizophrenia')
patients$kidney_disease <- get_binary_value(patients$disease,'kidney
disease')
patients$gastritis <- get_binary_value(patients$disease,'gastritis')
patients$alzheimer <- get_binary_value(patients$disease,'Alzheimer disease')
str(patients)
```

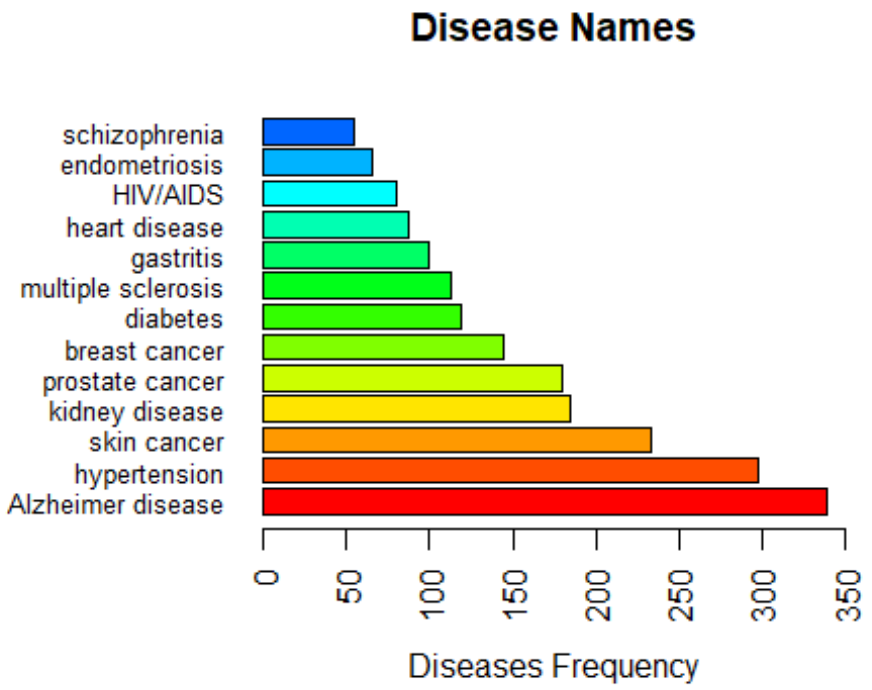
```
## 'data.frame':    2000 obs. of  27 variables:
##  $ gender            : Factor w/ 2 levels "female","male": 1 1 2 2 1 1 1 1
1 2 ...
##  $ dob               : Factor w/ 1877 levels "1923-10-10","1924-03-28",..:
505 1502 1811 545 327 1120 628 1378 631 1176 ...
```

```
##  $ zipcode           : int  89136 94105 89127 44101 89136 94105 60612
43221 89127 43210 ...
##  $ employment_status : Factor w/ 4 levels "employed","retired",..: 2 1 1 2
2 4 2 1 2 1 ...
##  $ education         : Factor w/ 4 levels "bachelors","highschool",..: 1 4
3 1 3 2 4 1 3 2 ...
##  $ marital_status    : Factor w/ 2 levels "married","single": 1 1 1 1 1 1
1 1 2 1 ...
##  $ children          : int  1 4 2 2 3 2 0 2 2 7 ...
##  $ ancestry          : Factor w/ 4 levels "british","east_europe",..: 4 3
4 3 4 4 2 1 4 2 ...
##  $ avg_commute       : num  13.4 15.2 23.6 19.6 36.5 ...
##  $ daily_internet_use: num  2.53 6.77 3.63 5 7.75 3.34 6.75 3.01 4.12 3.15
...
##  $ available_vehicles: int  2 2 1 3 1 0 2 3 1 1 ...
##  $ military_service  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1
...
##  $ disease           : Factor w/ 13 levels "Alzheimer disease",..: 8 4 11
10 13 1 9 2 1 7 ...
##  $ age               : num  65 65 40 65 65 65 65 65 65 65 ...
##  $ prostate_cancer   : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ skin_cancer       : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ breast_cancer     : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ hiv_aids          : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ diabetes          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ heart_disease     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ hypertension      : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ endometriosis     : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ multiple_sclerosis: num  0 0 0 1 0 0 0 0 0 0 ...
##  $ schizophrenia     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kidney_disease    : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ gastritis         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ alzheimer         : num  0 0 0 0 0 1 0 0 1 0 ...
```

Use barplots for the distribution of the categorical columns

Draw a bar plot to count the total number of diseases in the dataset

```r
par(las=2) # make label text perpendicular to axis
par(mar=c(5,8,4,2)) # increase y-axis margin.

disease_counts <- table(patients$disease)
barplot(sort(disease_counts, decreasing = TRUE), main="Disease Names",
        xlab="Diseases Frequency",
        col=rainbow(20),
        horiz=TRUE,
        cex.names=0.8,
        xlim = c(0, 350))
```



Observation : Male are more sick than Female

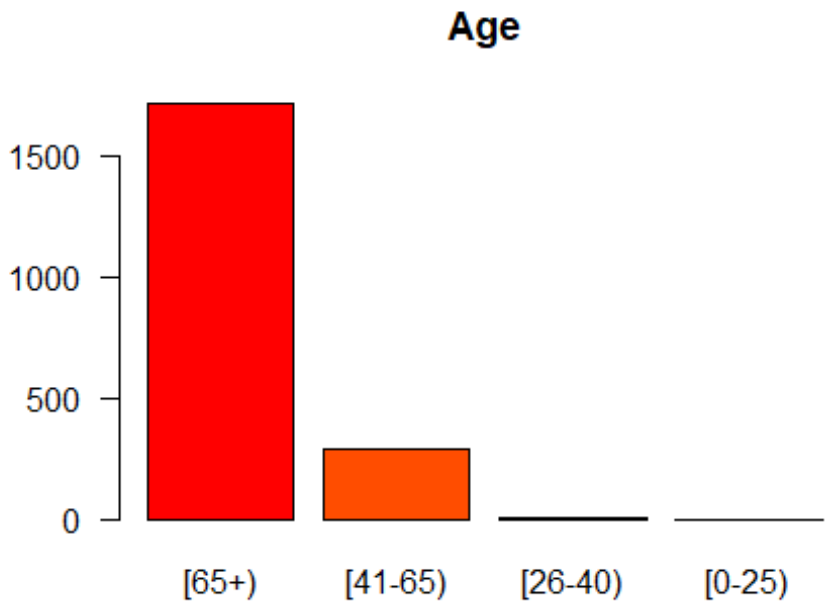
```
gender_counts <- table(patients$gender)
barplot(sort(gender_counts, decreasing = TRUE), main="Gender",
        col=rainbow(20), las=1)
```

Observation : age group that are more sick

```
age_breaks <- c(0,25,40,65,100)
tags <- c("[0-25)","[26-40)", "[41-65)", "[65+)")
age_group_tags <- cut(patients$age,
                      breaks=age_breaks,
                      include.lowest=TRUE,
                      right=FALSE,
                      labels=tags)
summary(age_group_tags)
```

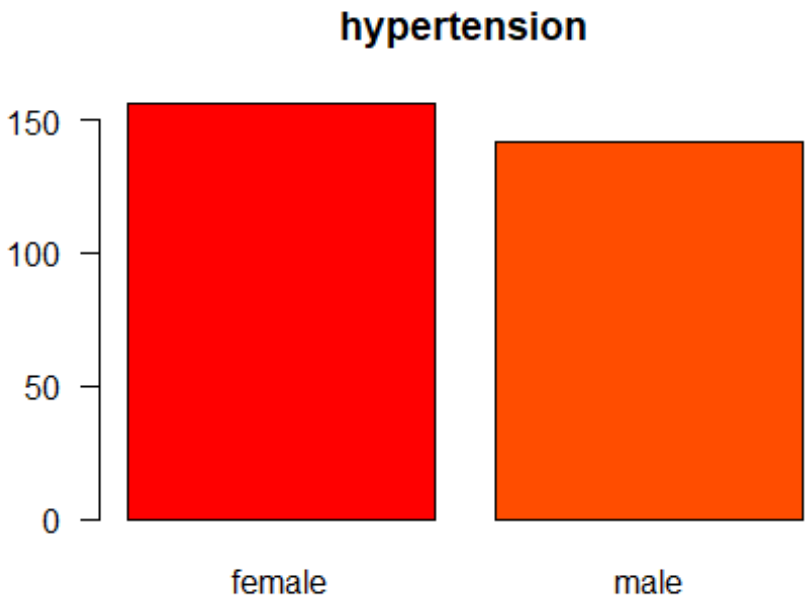
```
##  [0-25) [26-40) [41-65)   [65+)
##       0       4     285    1711
```

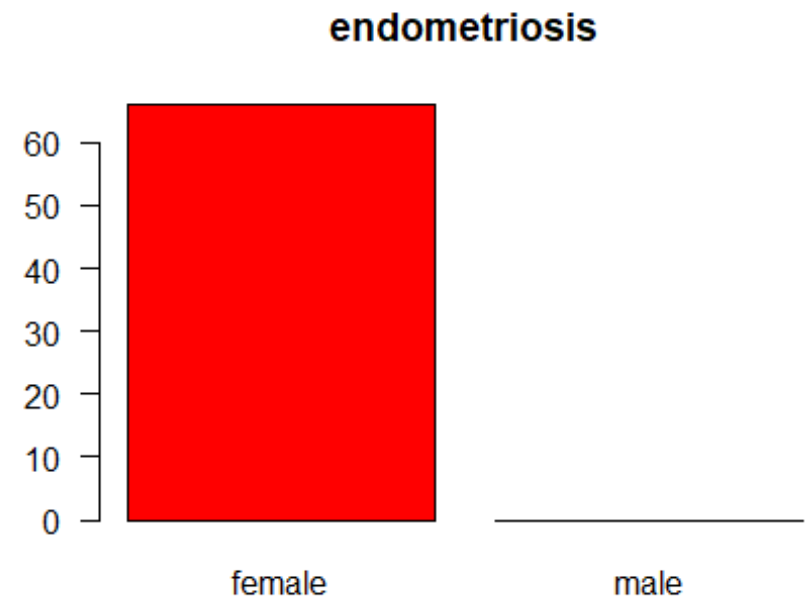
```
#age_counts <- table(patients$age)
age_counts <- table(age_group_tags)
barplot(sort(age_counts, decreasing = TRUE), main="Age",
        col=rainbow(20), las=1)
```

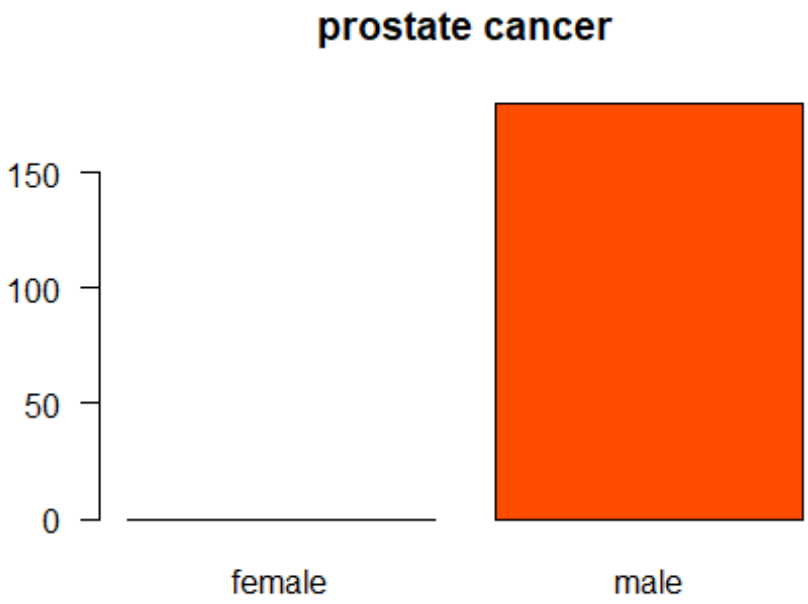


Observation : Disease and Gender distrubution

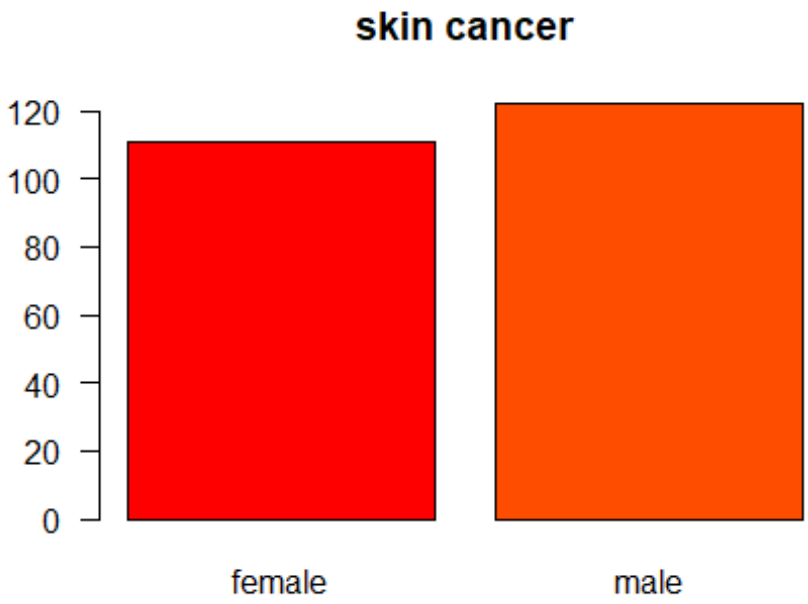
```
disease_name = c(as.character(unique(patients$disease)))

for (d in disease_name) {
  gender_disease_counts <- subset(patients, patients$disease == d)
  gender_disease_counts <- table(gender_disease_counts$gender)
  barplot(gender_disease_counts, main=d, col=rainbow(20), las=1)
}
```

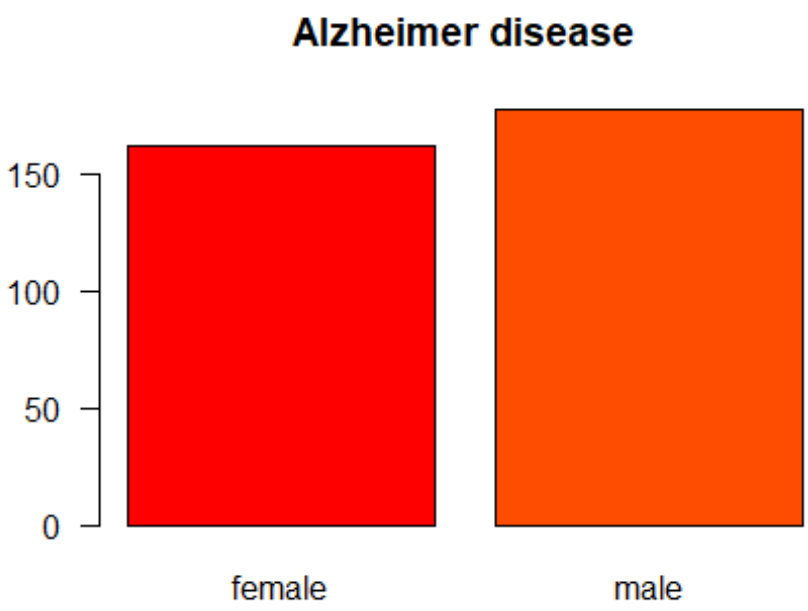

hypertension
150
100
50
0
female
male

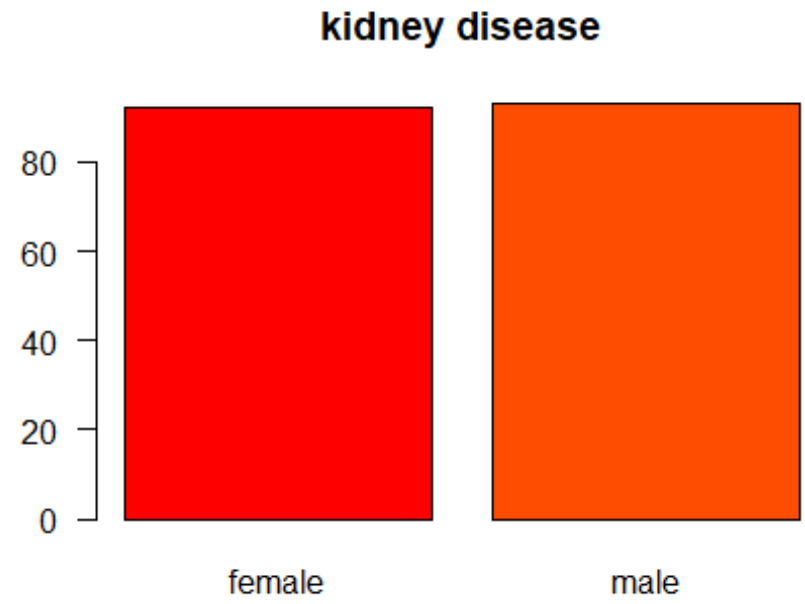

endometriosis
60
50
40
30
20
10
0
female
male

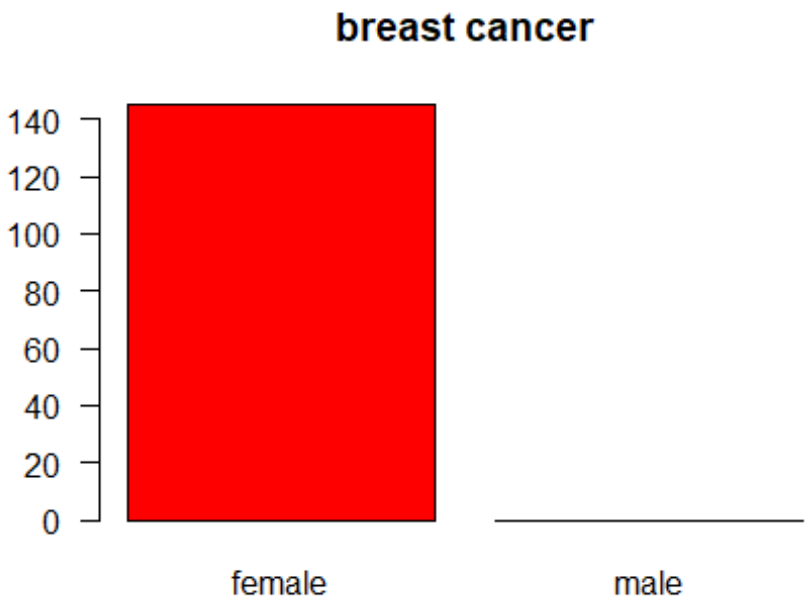

prostate cancer
150
100
50
0
female
male

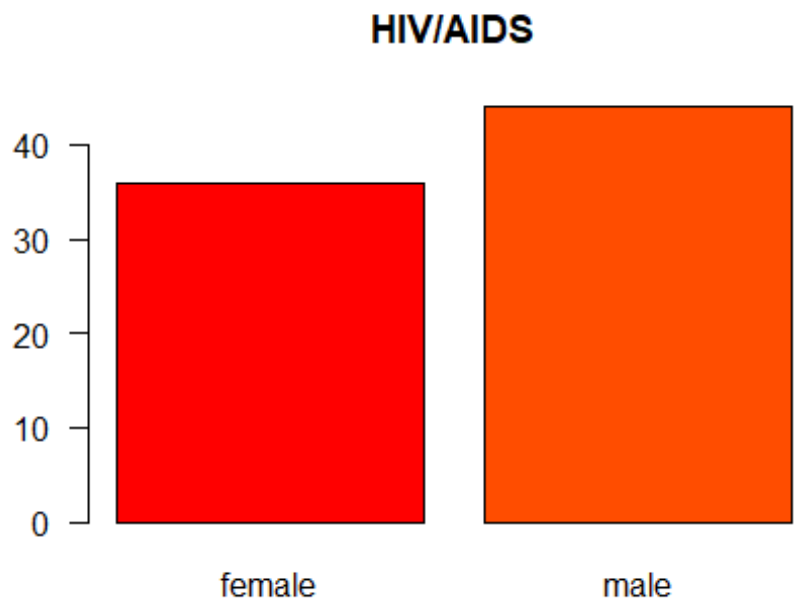

multiple sclerosis
60
50
40
30
20
10
0
female
male

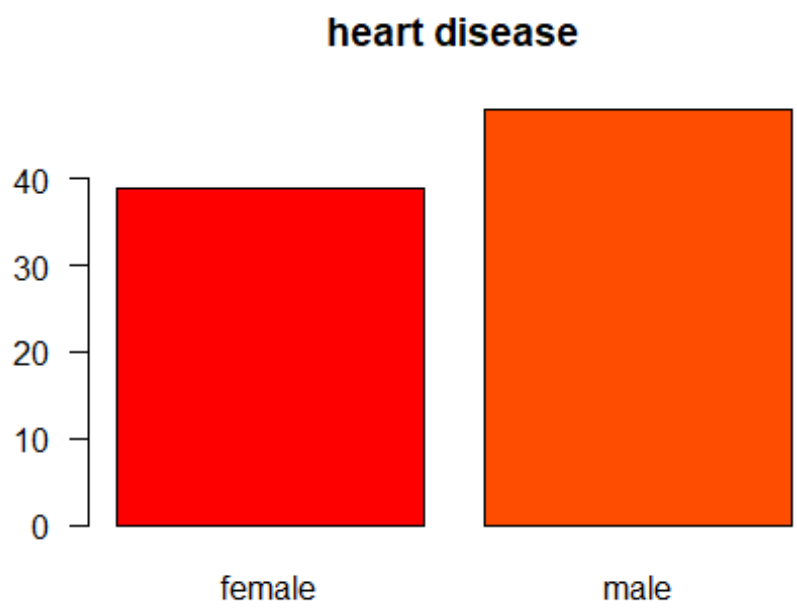

skin cancer
120
100
80
60
40
20
0
female
male

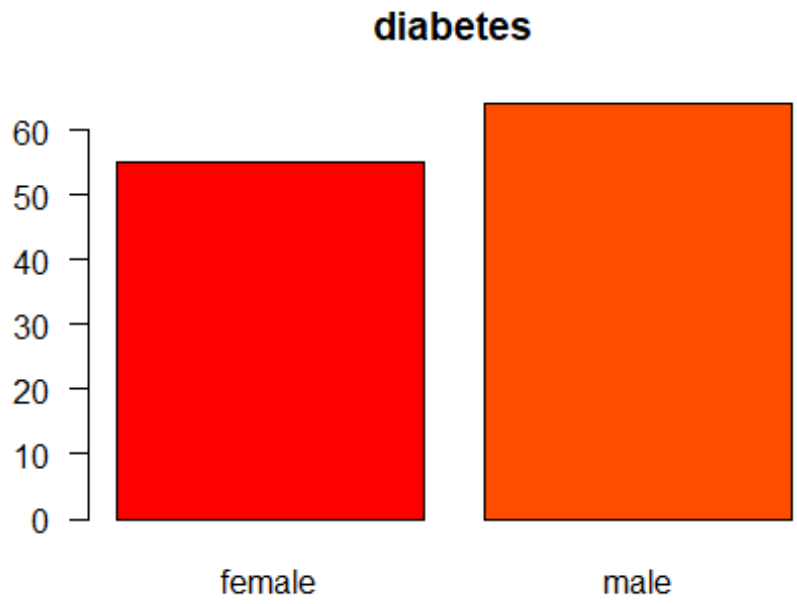

Alzheimer disease
150
100
50
0
female
male

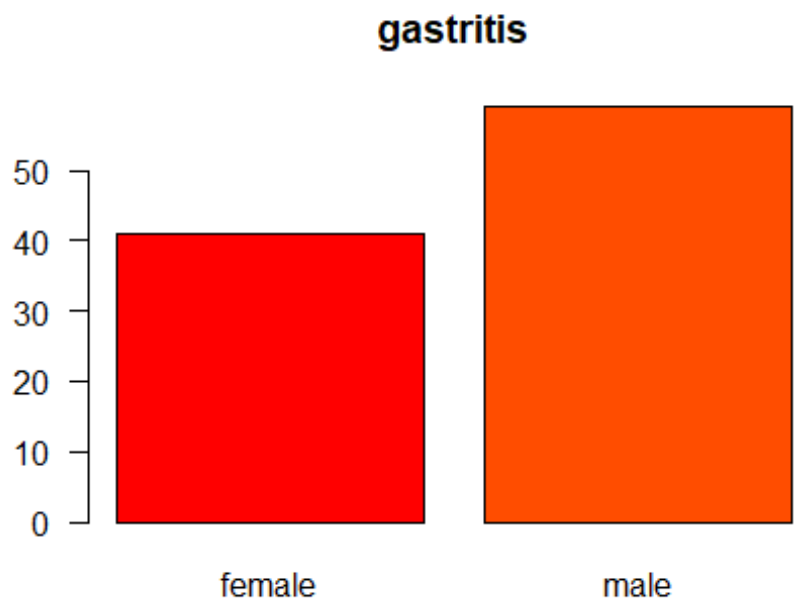

kidney disease
80
60
40
20
0
female
male

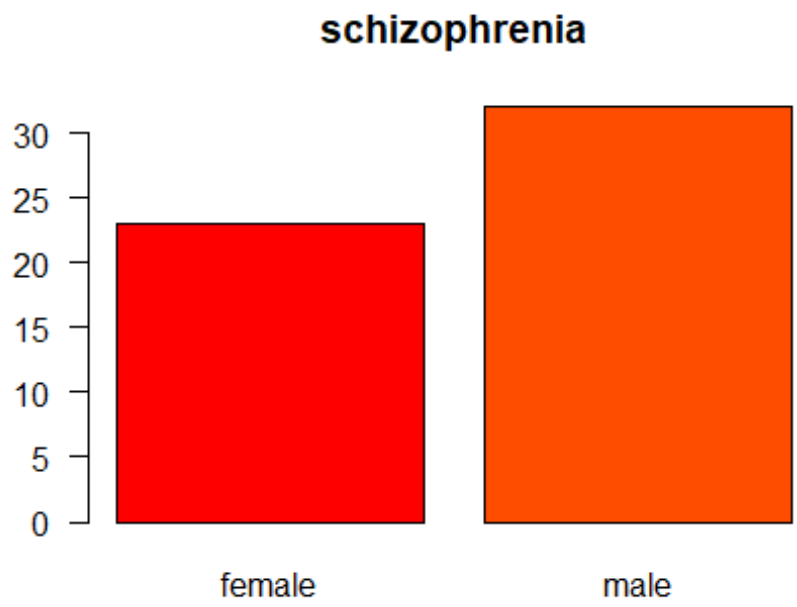

breast cancer
140
120
100
80
60
40
20
0
female
male


HIV/AIDS
40
30
20
10
0
female
male


heart disease
40
30
20
10
0
female
male

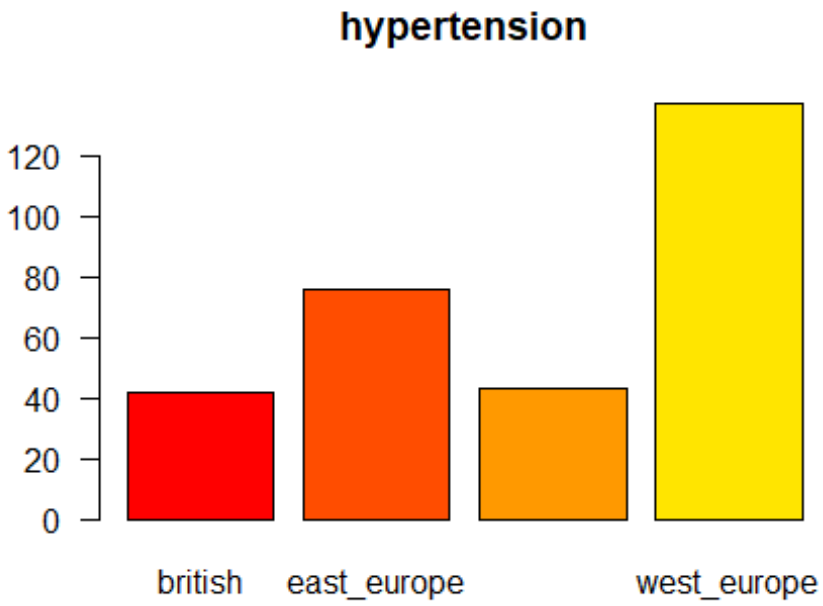

diabetes
60
50
40
30
20
10
0
female
male

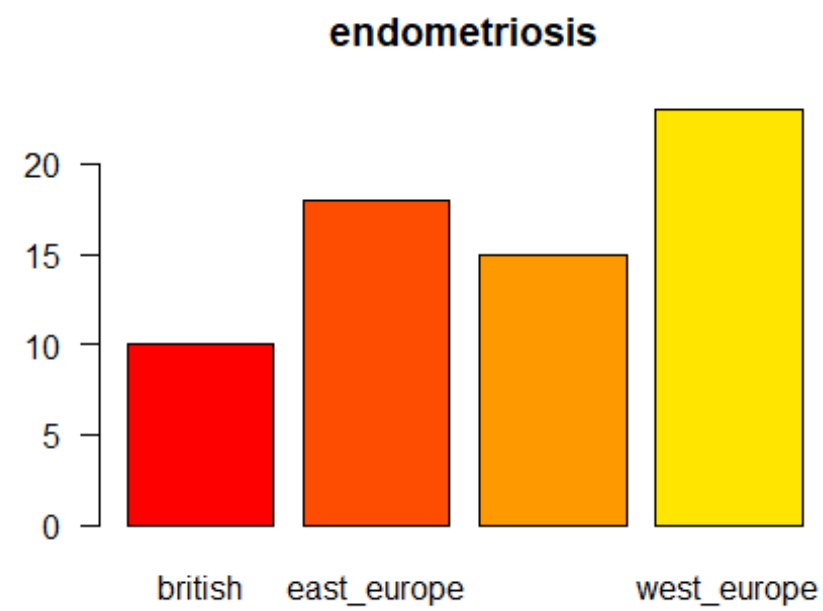

gastritis
50
40
30
20
10
0
female
male

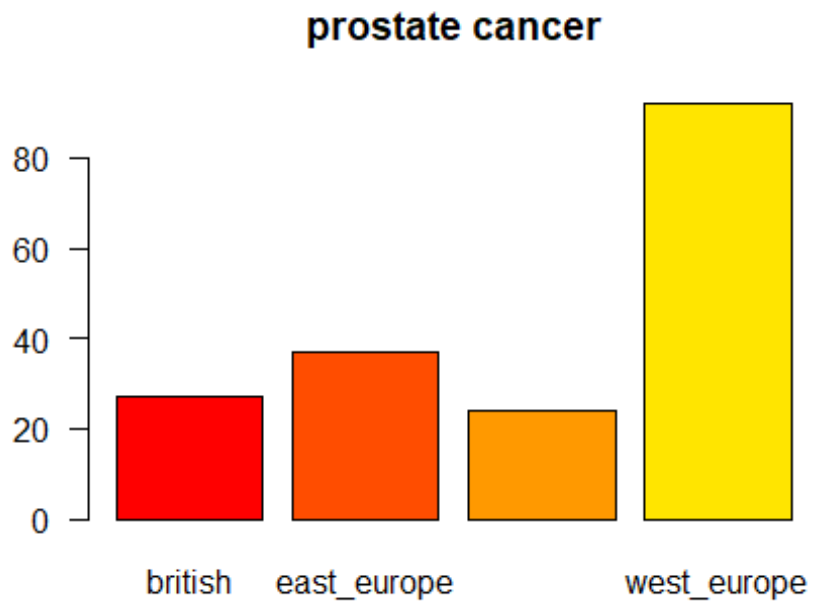


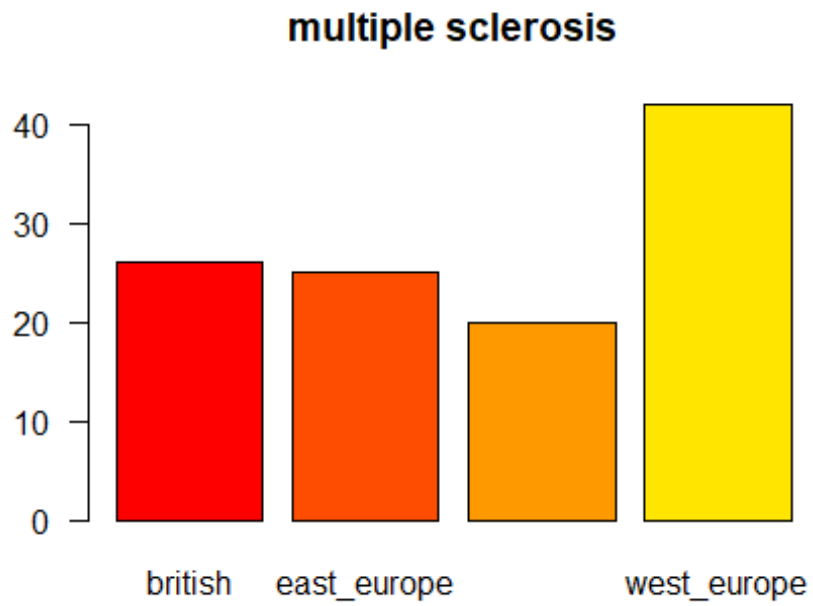
Observation : Disease and ancestry distrubution

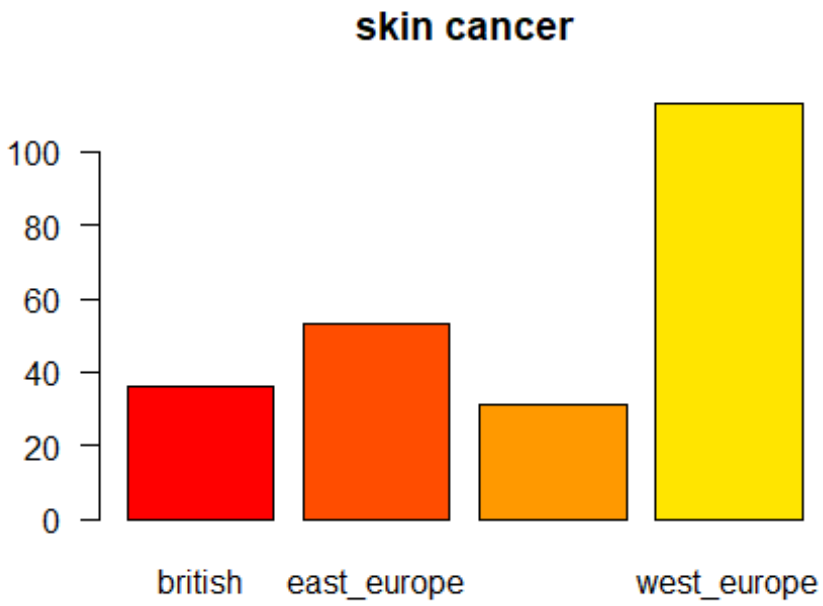
```
for (d in disease_name) {
  ancestry_disease_counts <- subset(patients, patients$disease == d)
  ancestry_disease_counts <- table(ancestry_disease_counts$ancestry)
  barplot(ancestry_disease_counts, main=d, col=rainbow(20), las=1)
}
```

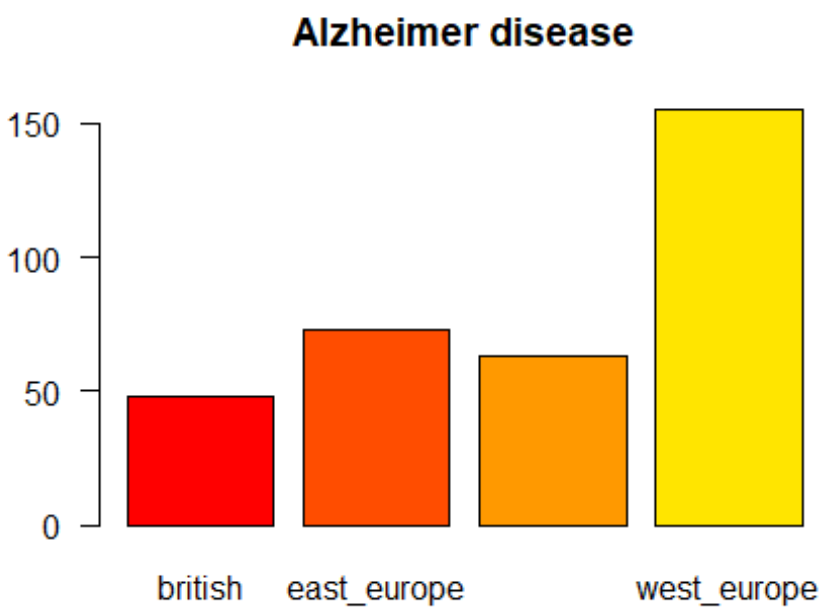

hypertension
120
100
80
60
40
20
0
british
east_europe
west_europe

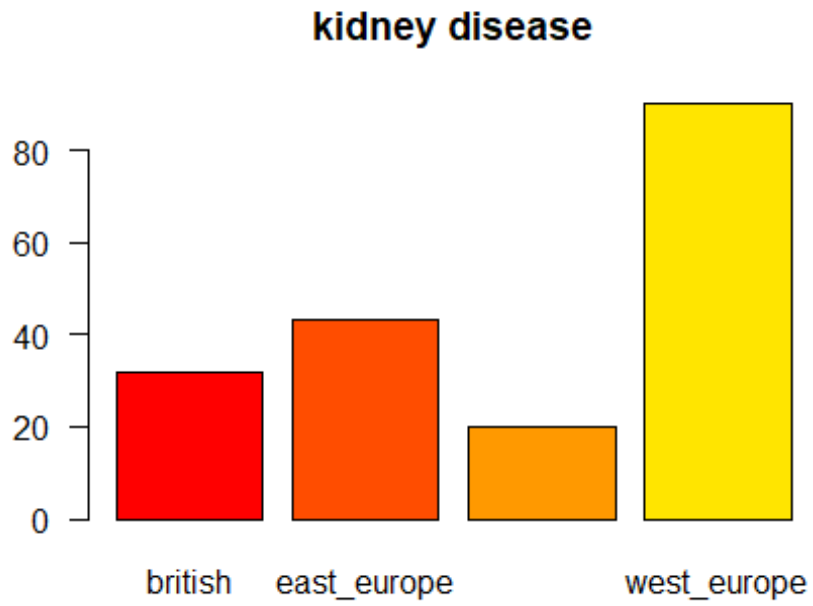

endometriosis
20
15
10
5
0
british
east_europe
west_europe

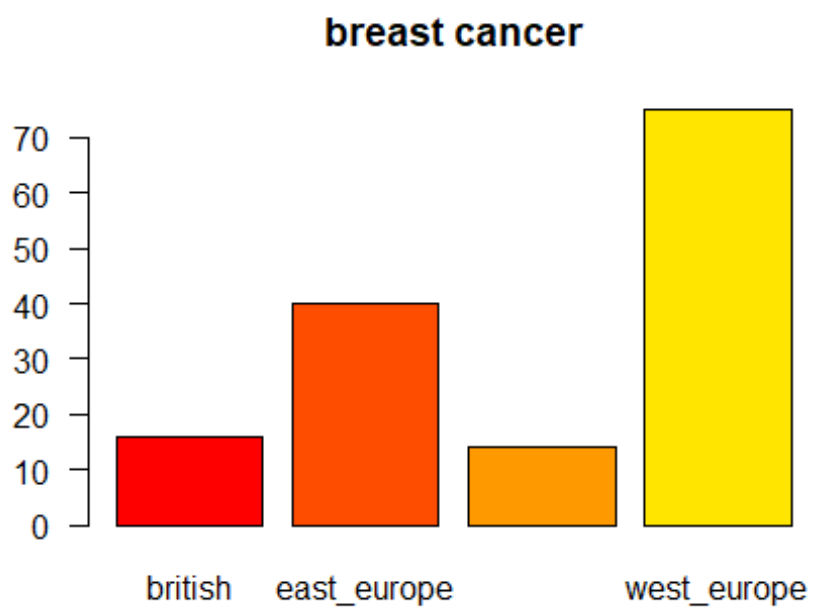

prostate cancer
80
60
40
20
0
british
east_europe
west_europe

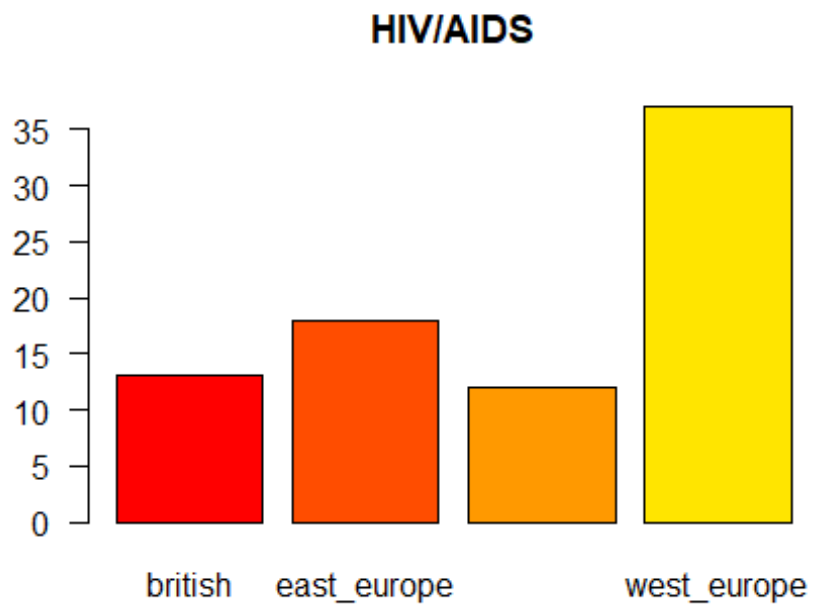

multiple sclerosis
40
30
20
10
0
british
east_europe
west_europe

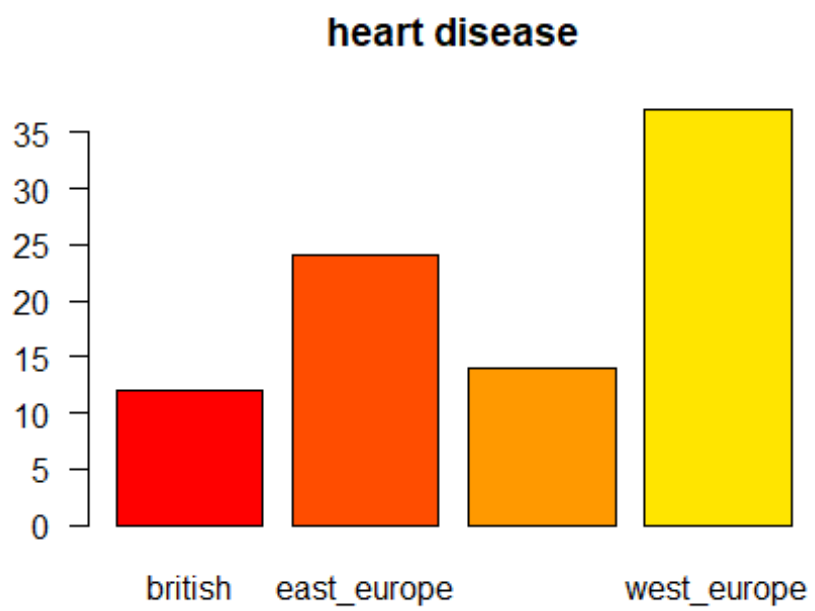

skin cancer
100
80
60
40
20
0
british
east_europe
west_europe

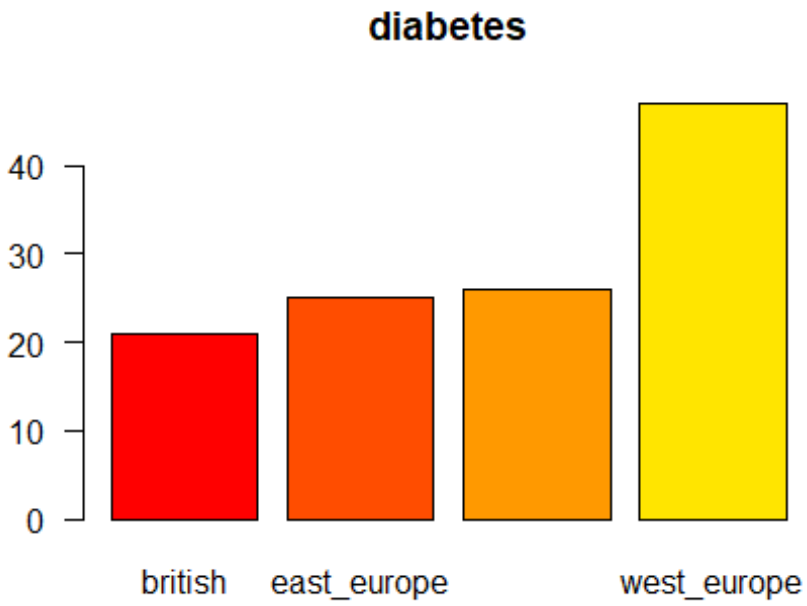

Alzheimer disease
150
100
50
0
british
east_europe
west_europe

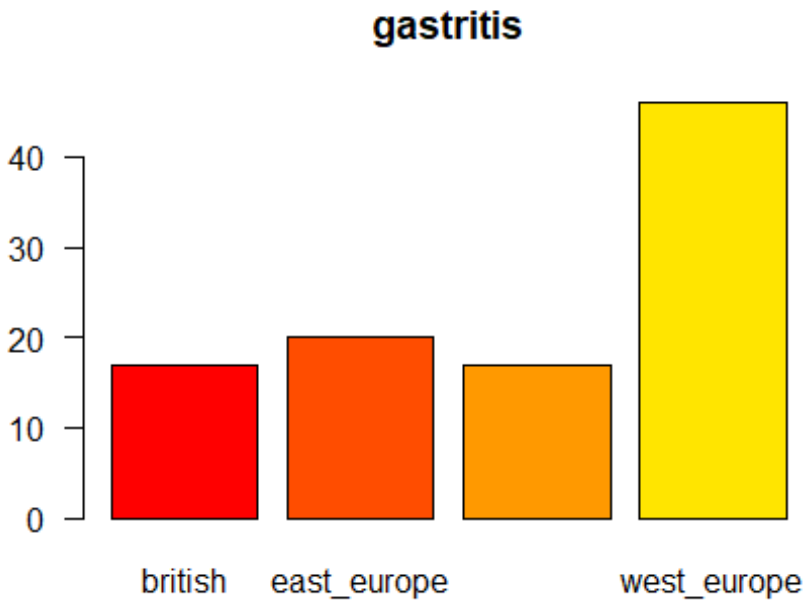

kidney disease
80
60
40
20
0
british
east_europe
west_europe

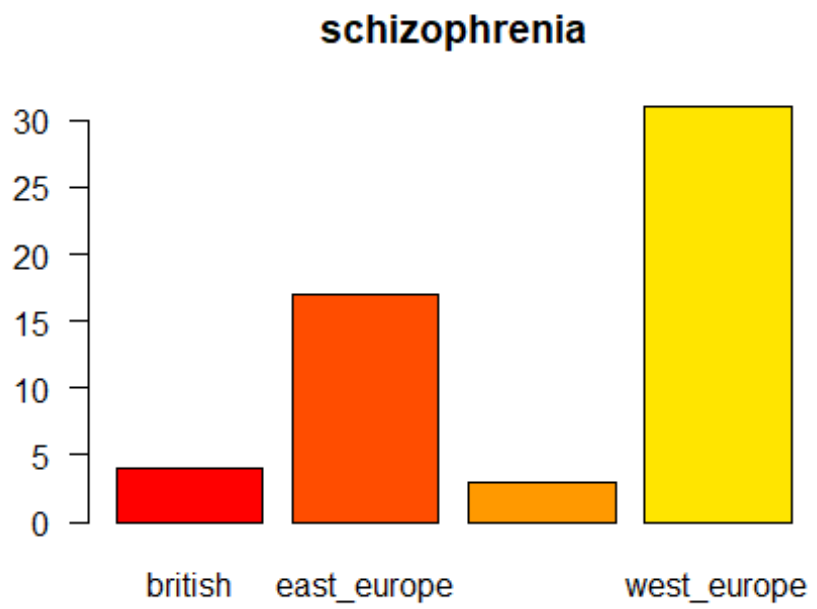

breast cancer
70
60
50
40
30
20
10
0
british
east_europe
west_europe


HIV/AIDS
35
30
25
20
15
10
5
0
british
east_europe
west_europe


heart disease
35
30
25
20
15
10
5
0
british
east_europe
west_europe


diabetes
40
30
20
10
0
british
east_europe
west_europe

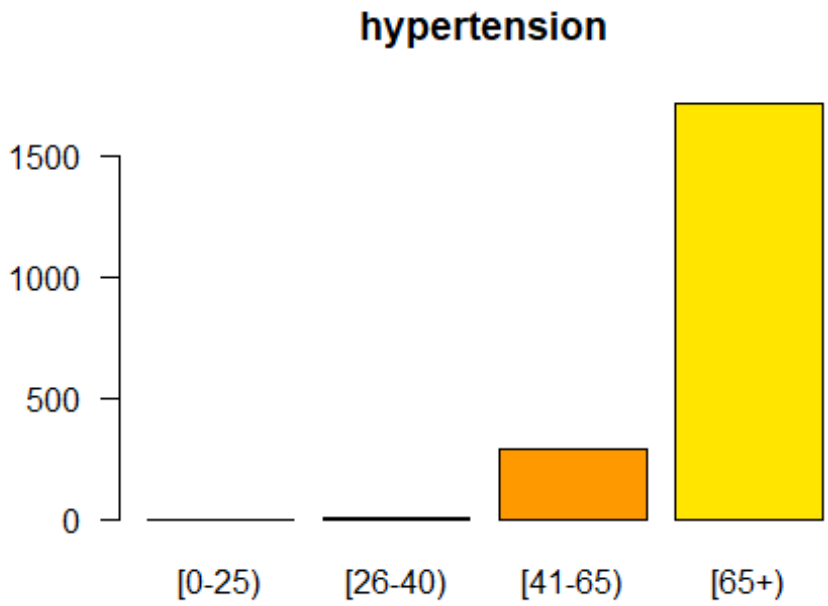

gastritis
40
30
20
10
0
british
east_europe
west_europe

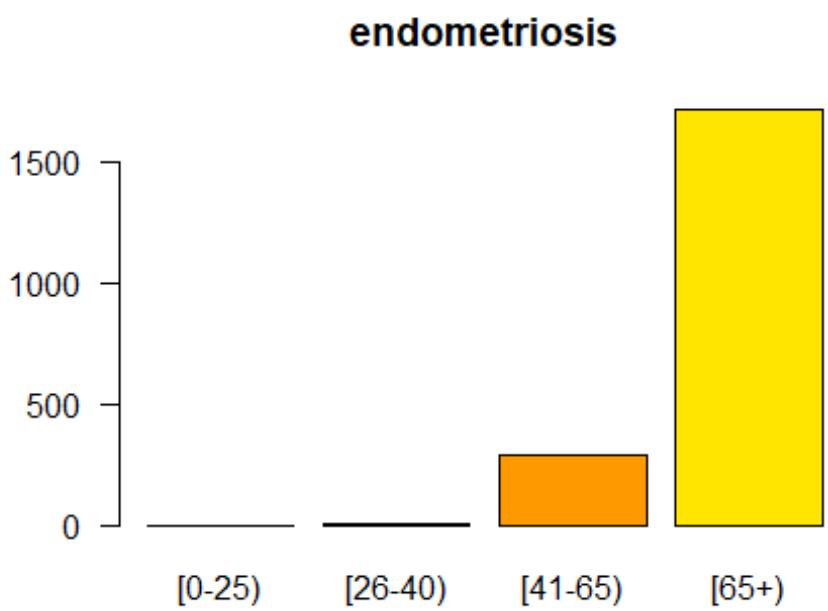


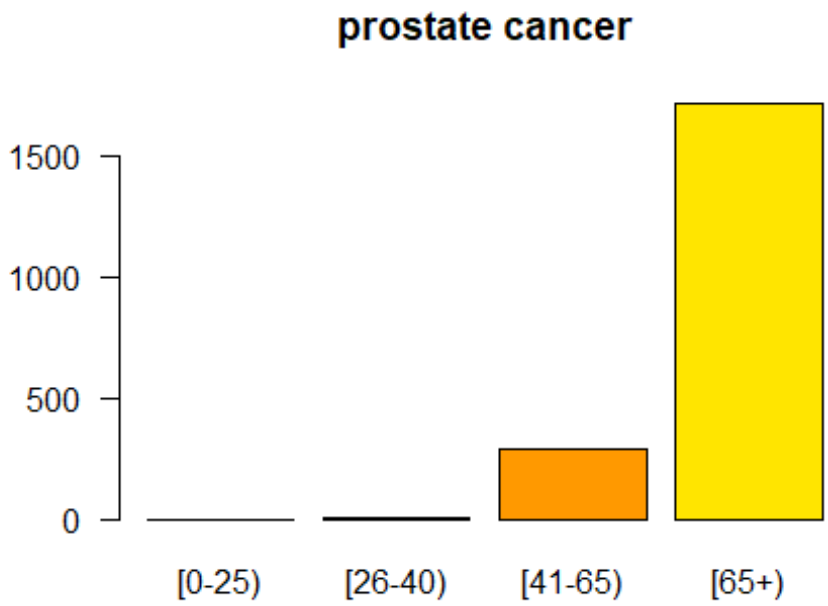
Plots for dependent variables

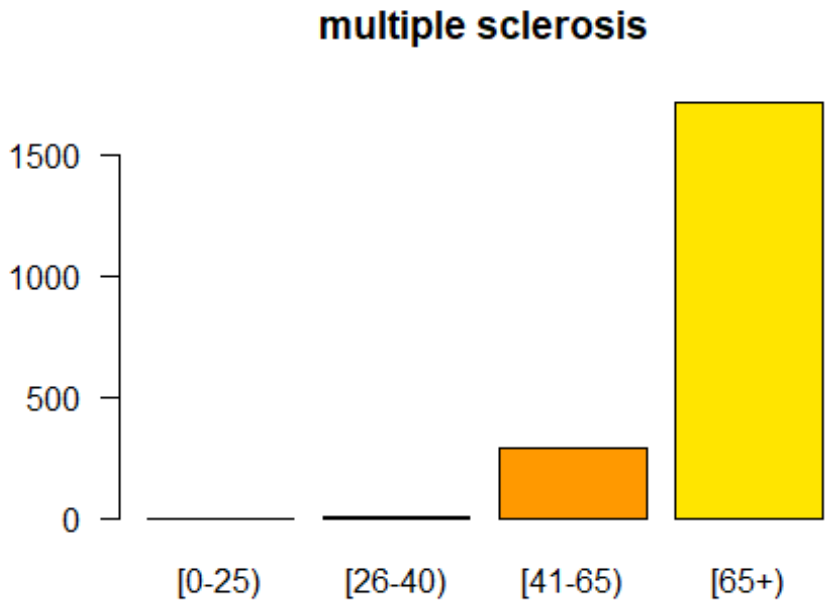
Observation : Disease and age distrubution

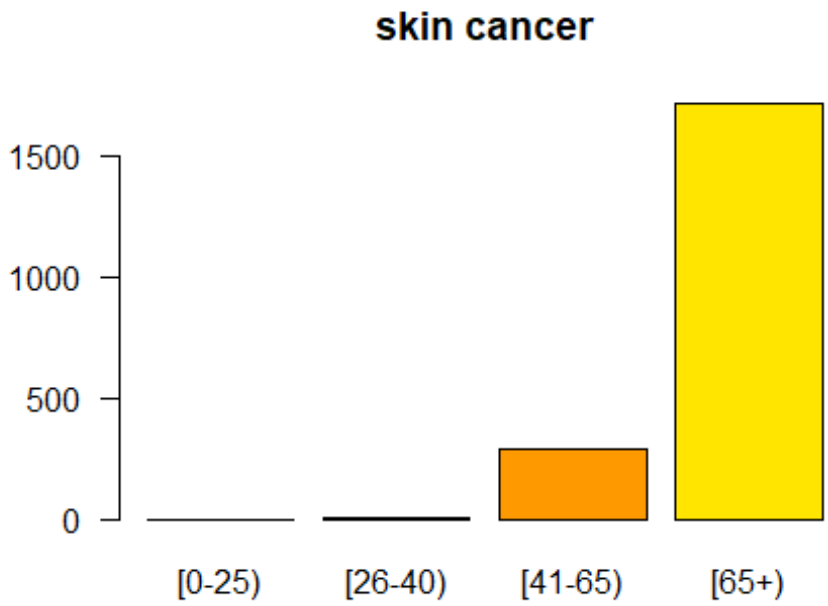
```
for (d in disease_name) {
  age_disease_counts <- subset(patients, patients$disease == d)
  #age_disease_counts <- table(age_disease_counts$age_group_tags)
  age_disease_counts <- table(age_disease_counts$age)
  barplot(age_counts, main=d, col=rainbow(20), las=1)
}
```

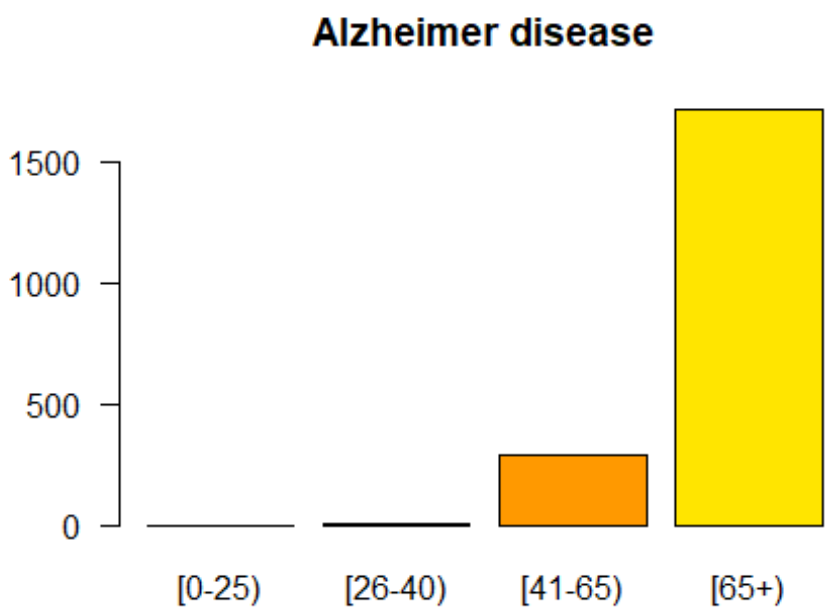

hypertension
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

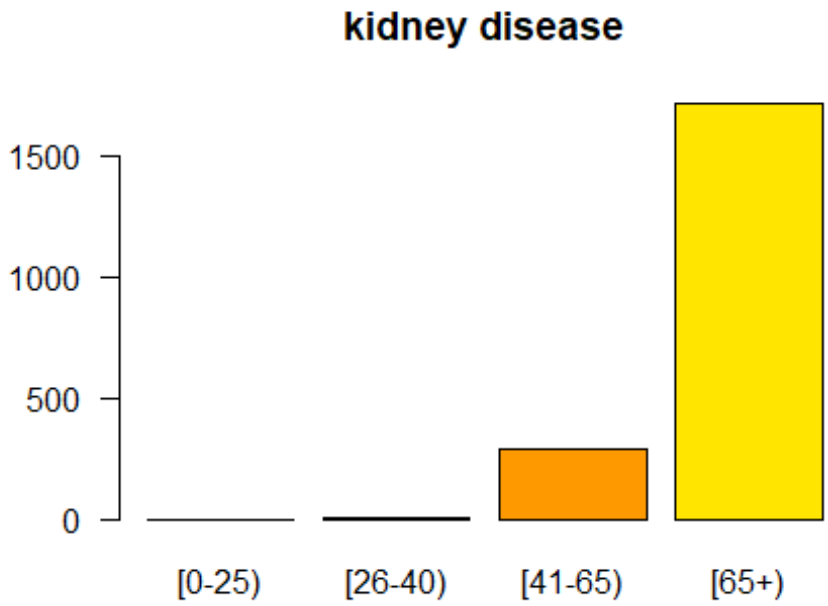

endometriosis
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

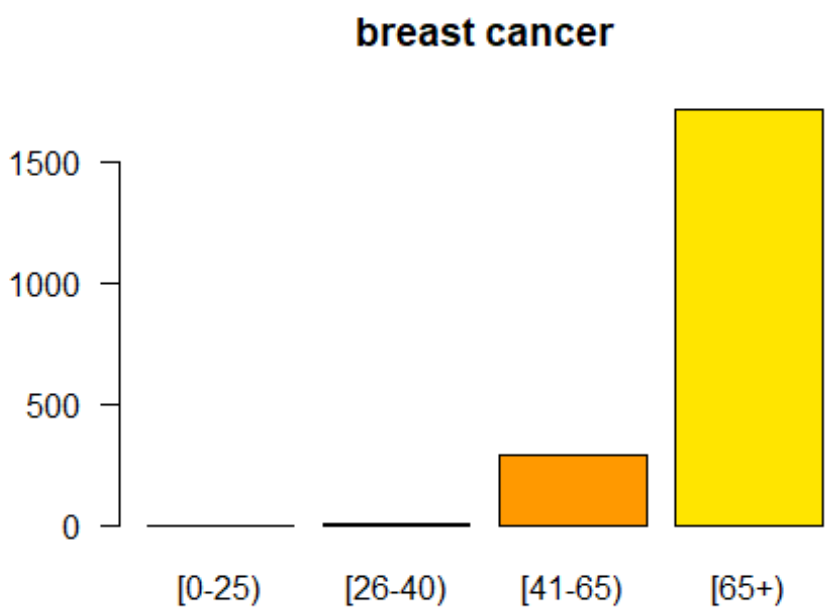

prostate cancer
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

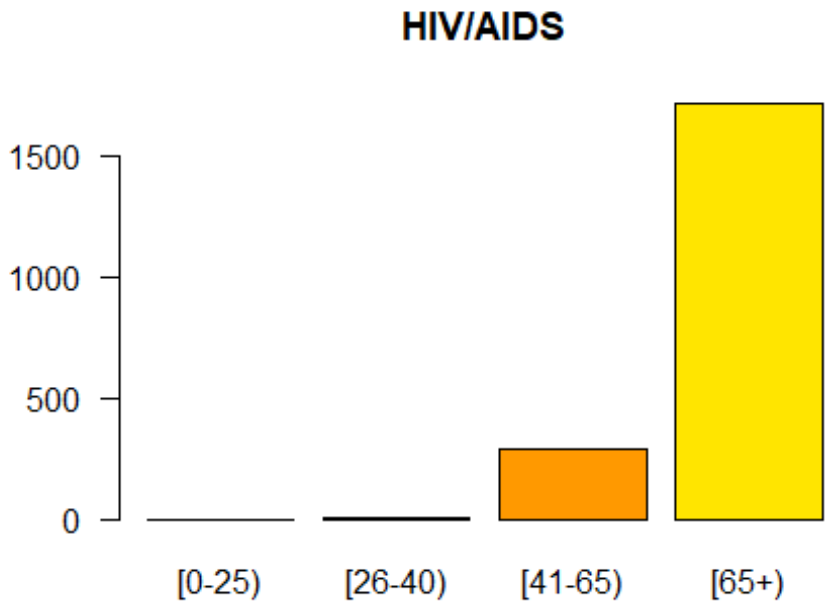

multiple sclerosis
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

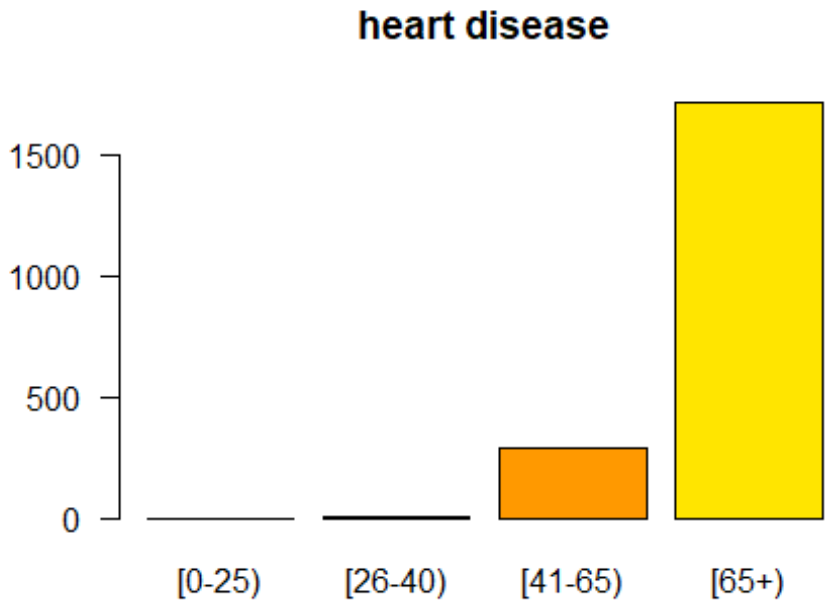

skin cancer
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

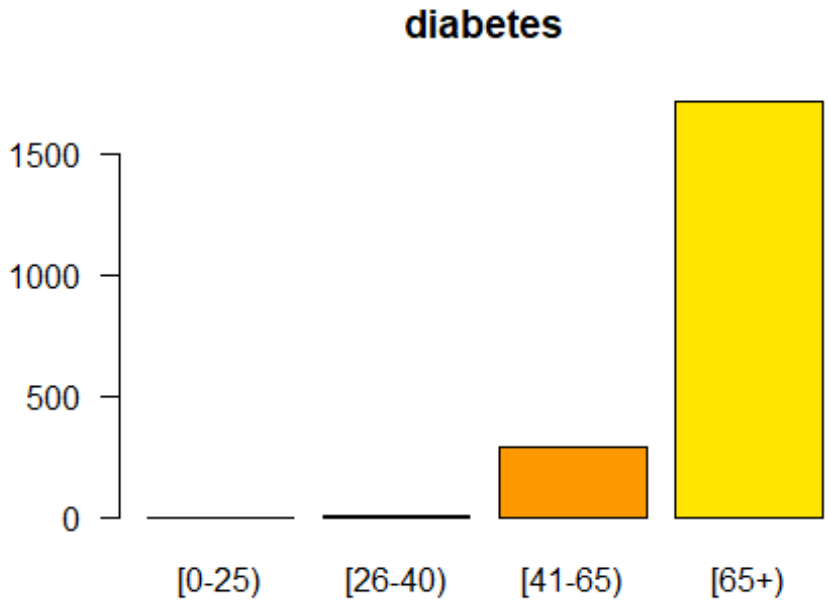

Alzheimer disease
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

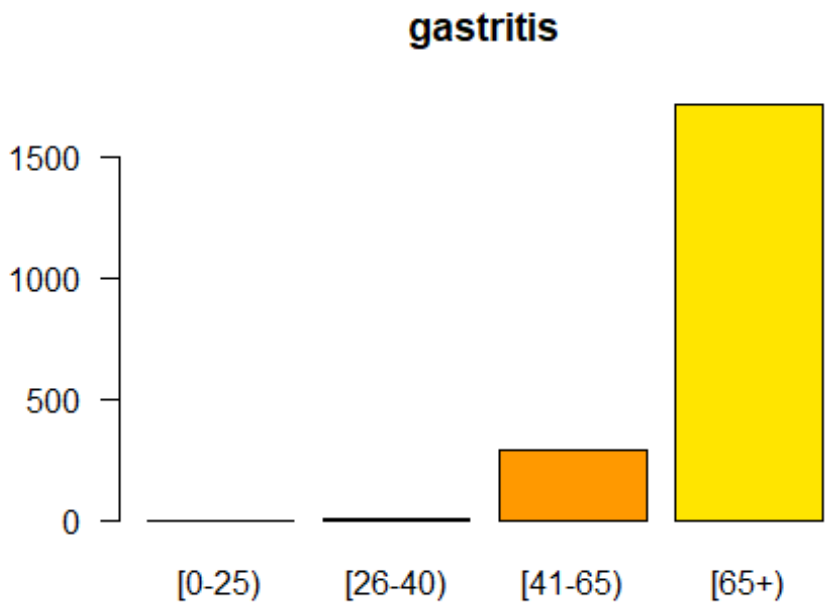

kidney disease
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

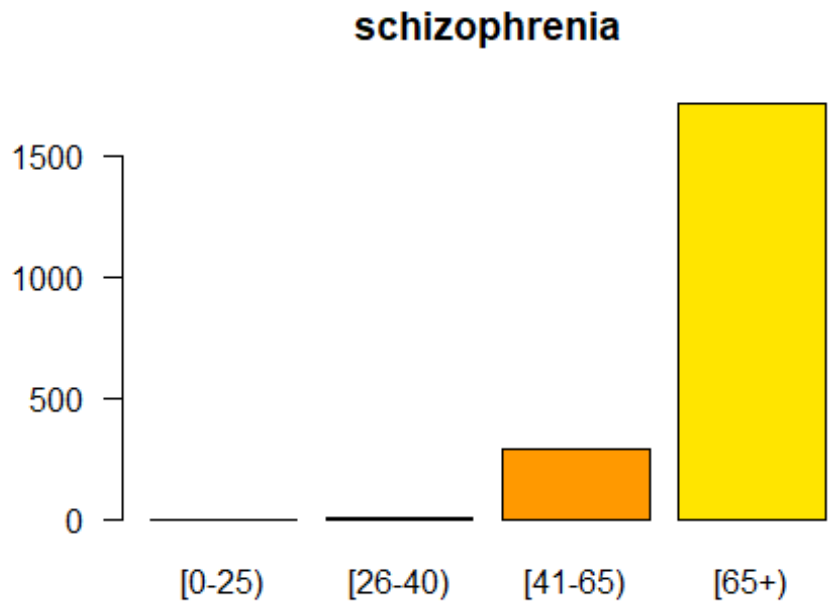

breast cancer
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)


HIV/AIDS
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)


heart disease
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

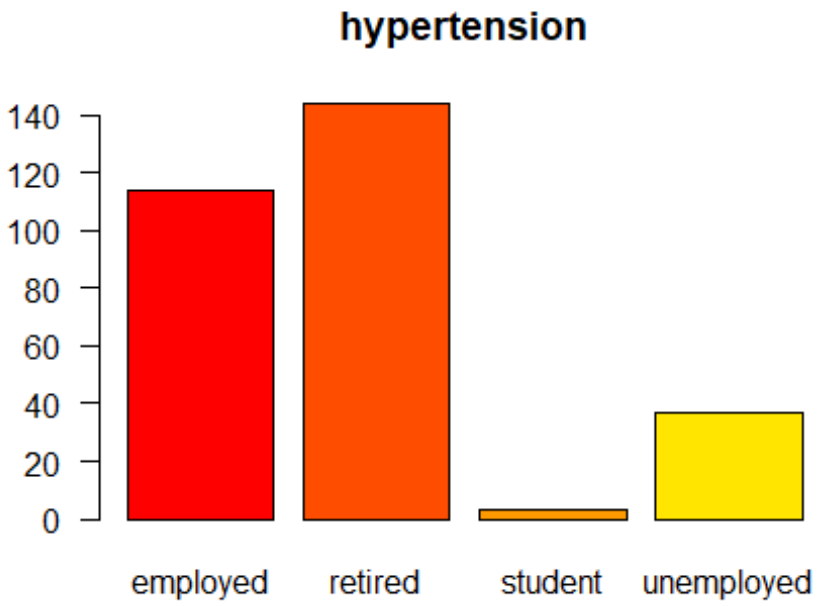

diabetes
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

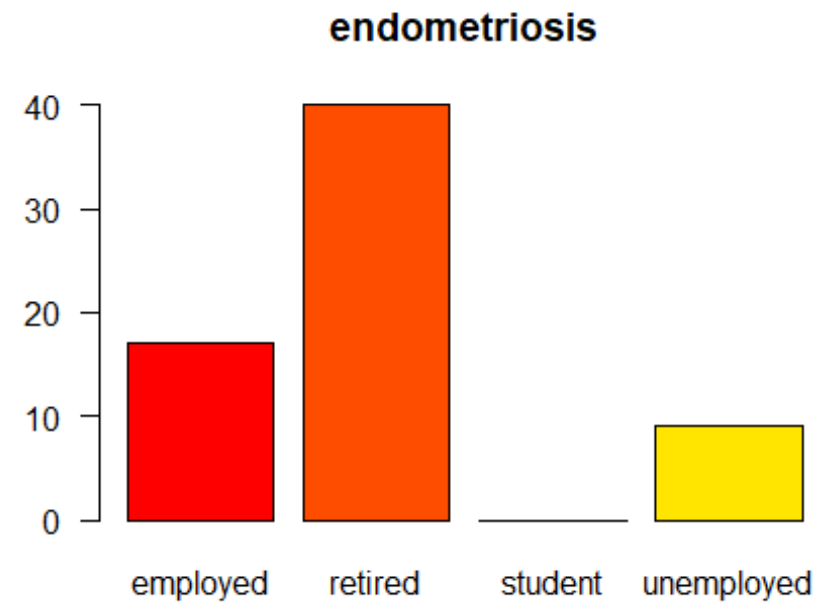

gastritis
1500
1000
500
0
[0-25)
[26-40)
[41-65)
[65+)

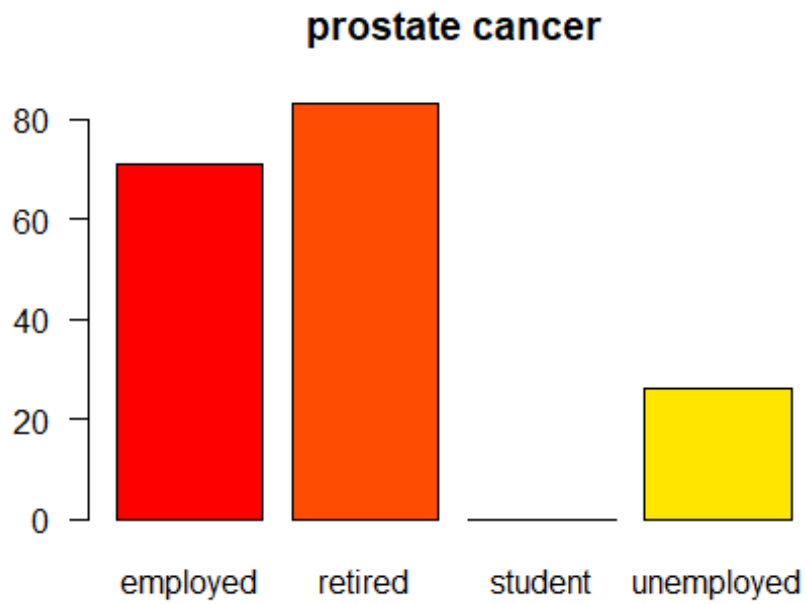


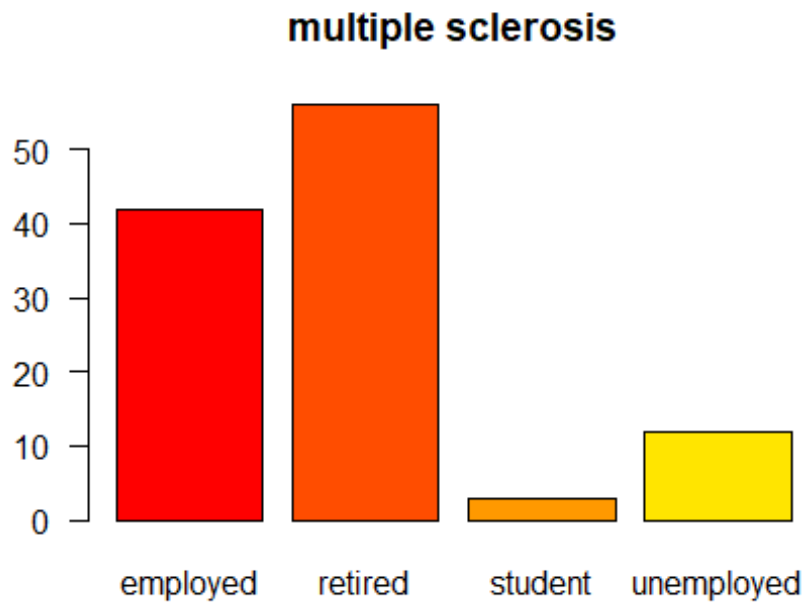
Observation : Disease and employment status distrubution

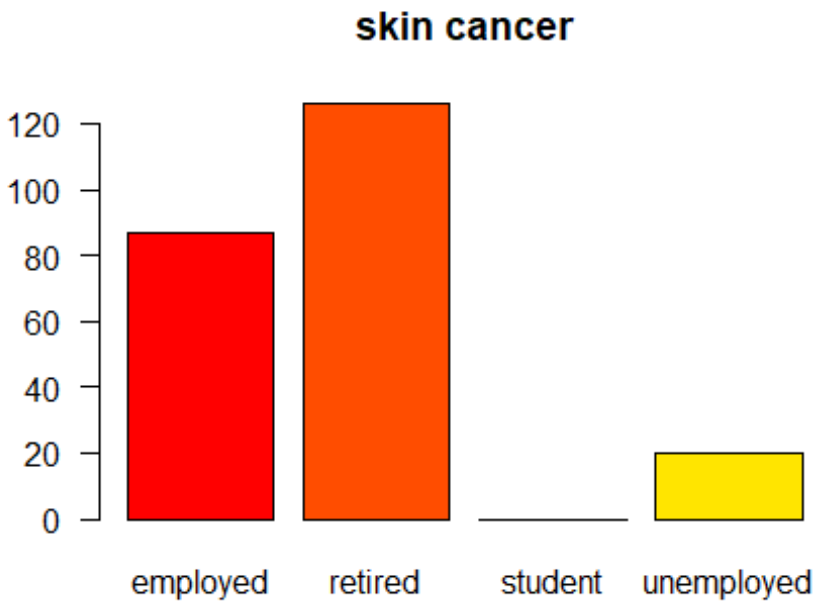
```
for (d in disease_name) {
  emp_disease_counts <- subset(patients, patients$disease == d)
  emp_disease_counts <- table(emp_disease_counts$employment_status)
  barplot(emp_disease_counts, main=d, col=rainbow(20), las=1)
}
```

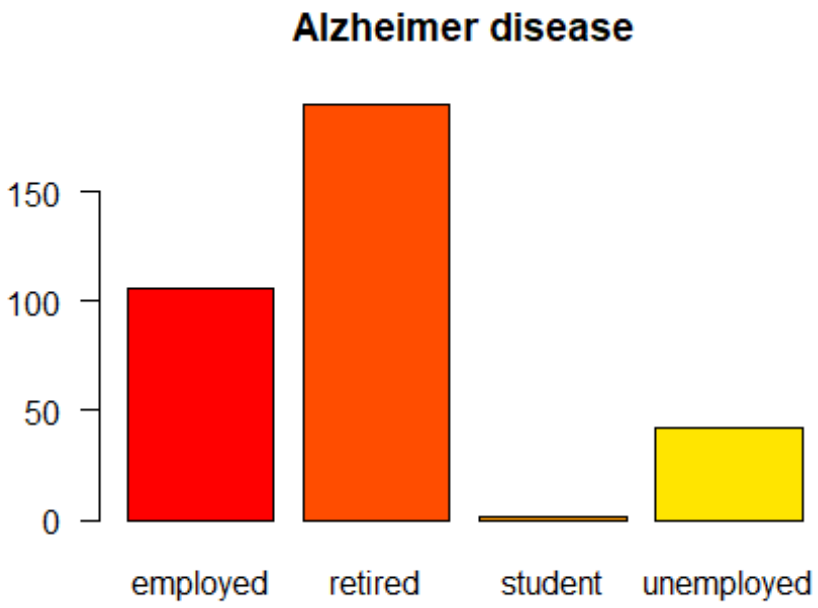

hypertension
140
120
100
80
60
40
20
0
employed
retired
student
unemployed

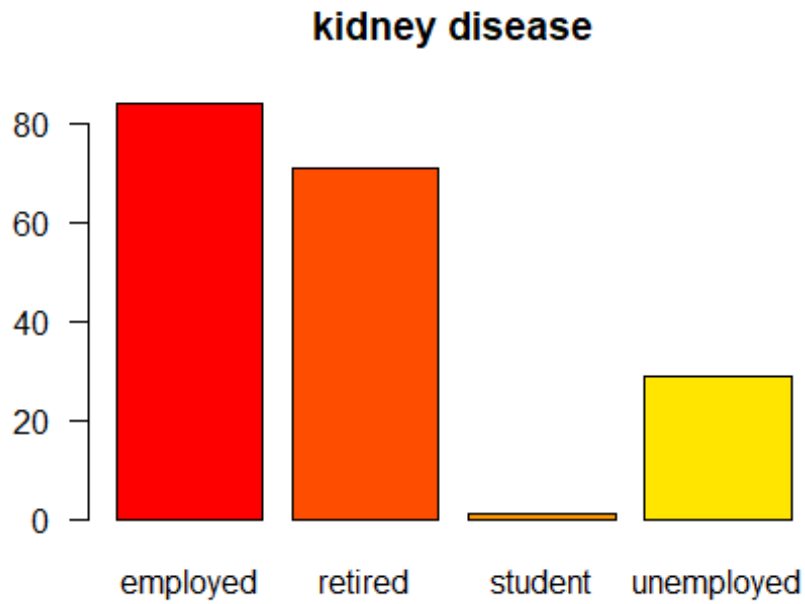

endometriosis
40
30
20
10
0
employed
retired
student
unemployed

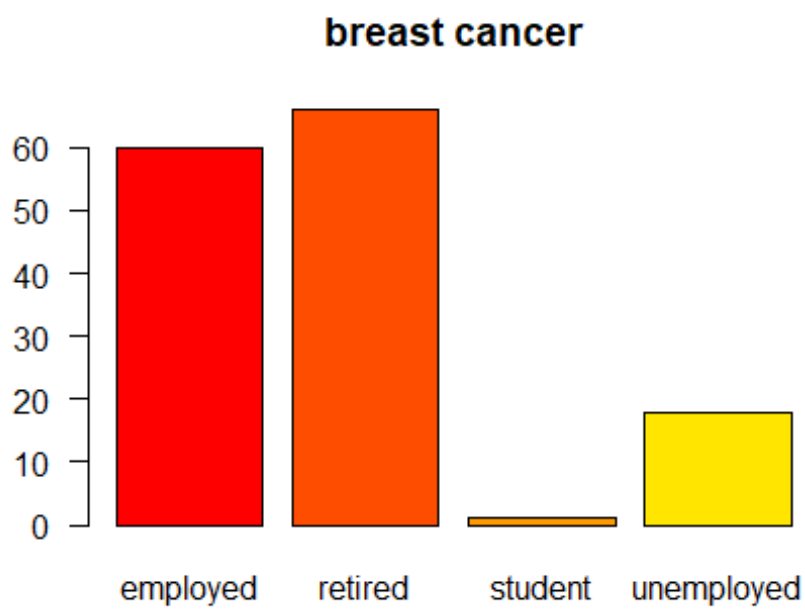

prostate cancer
80
60
40
20
0
employed
retired
student
unemployed

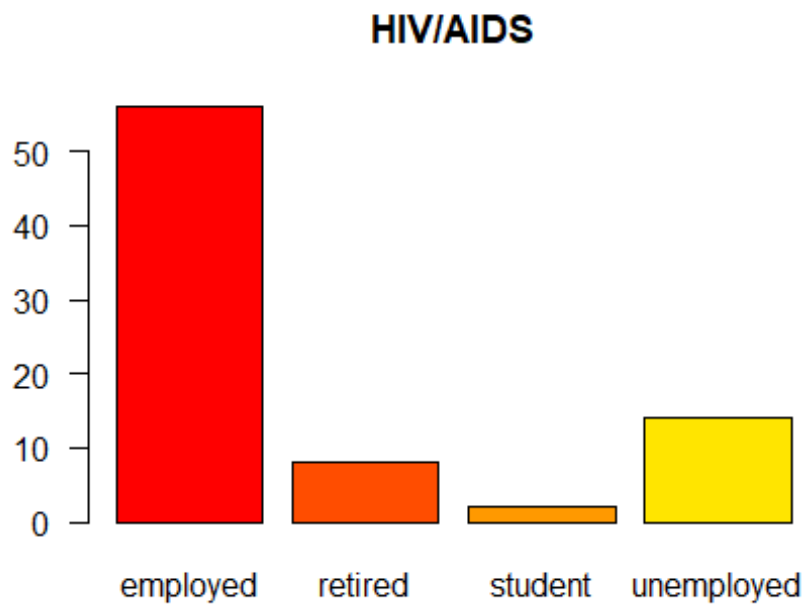

multiple sclerosis
50
40
30
20
10
0
employed
retired
student
unemployed

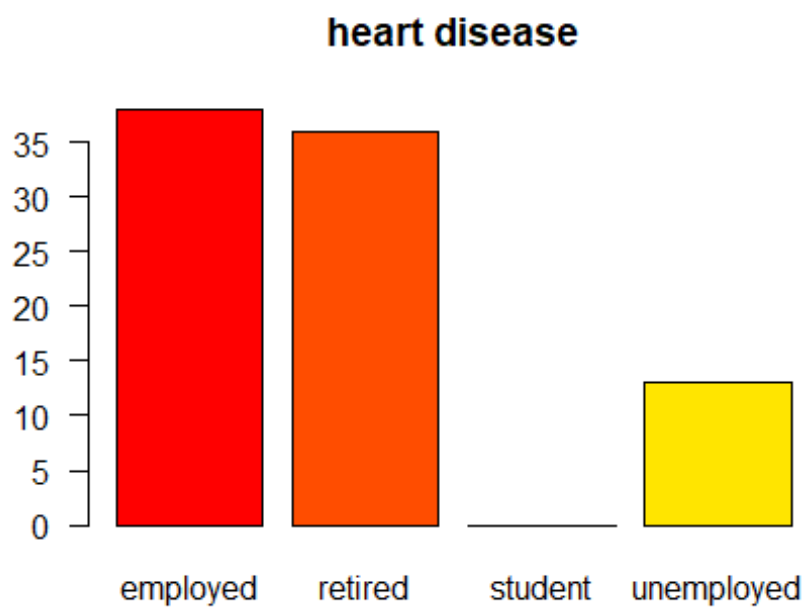

skin cancer
120
100
80
60
40
20
0
employed
retired
student
unemployed

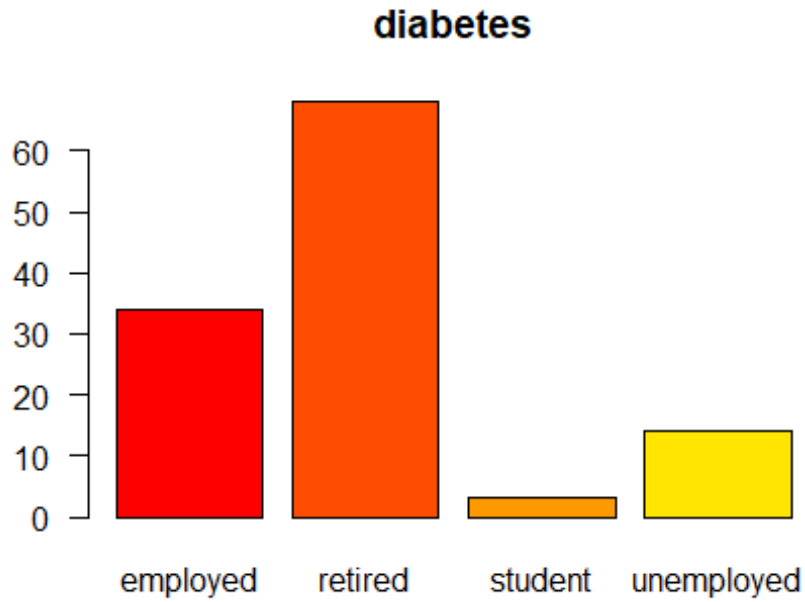

Alzheimer disease
150
100
50
0
employed
retired
student
unemployed

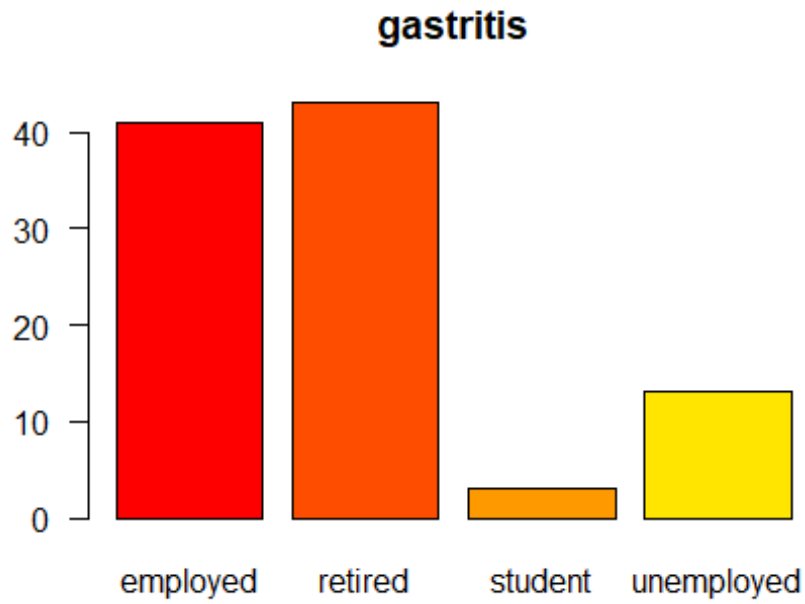

kidney disease
80
60
40
20
0
employed
retired
student
unemployed

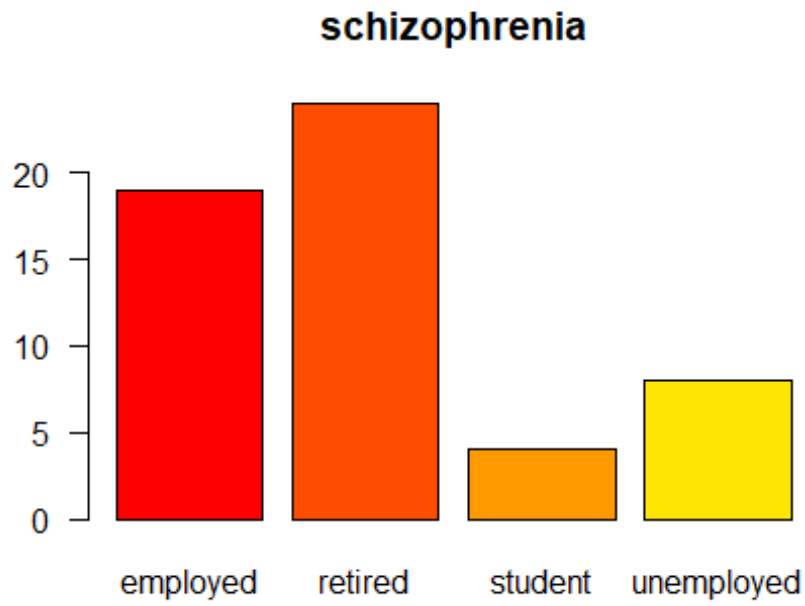

breast cancer
60
50
40
30
20
10
0
employed
retired
student
unemployed


HIV/AIDS
50
40
30
20
10
0
employed
retired
student
unemployed


heart disease
35
30
25
20
15
10
5
0
employed
retired
student
unemployed


diabetes
60
50
40
30
20
10
0
employed
retired
student
unemployed

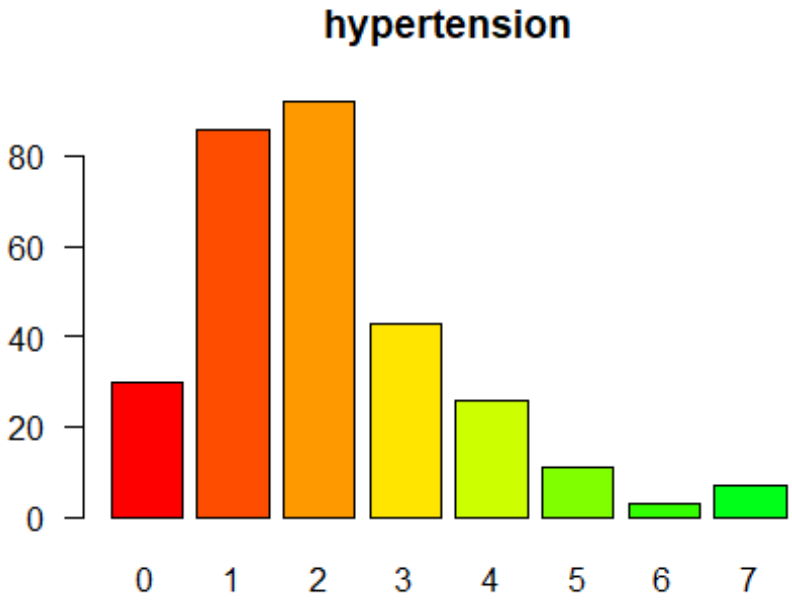

gastritis
40
30
20
10
0
employed
retired
student
unemployed

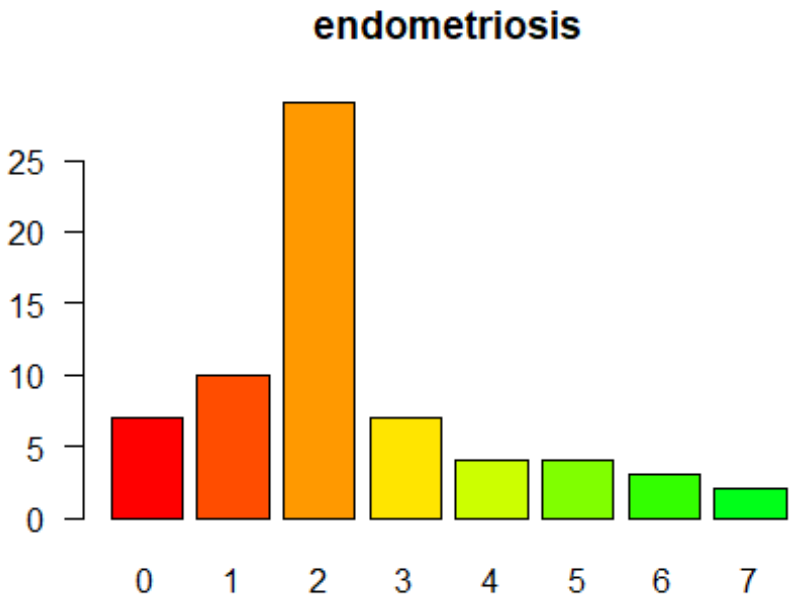


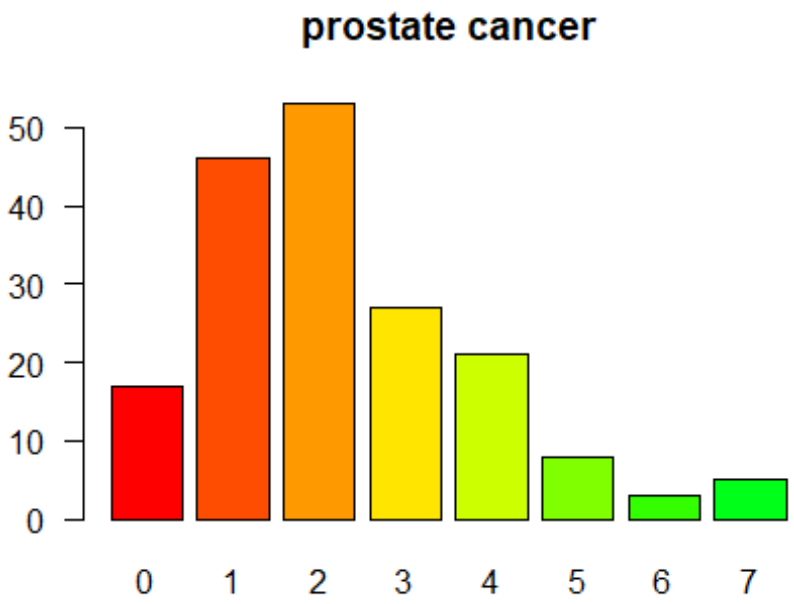
Observation :

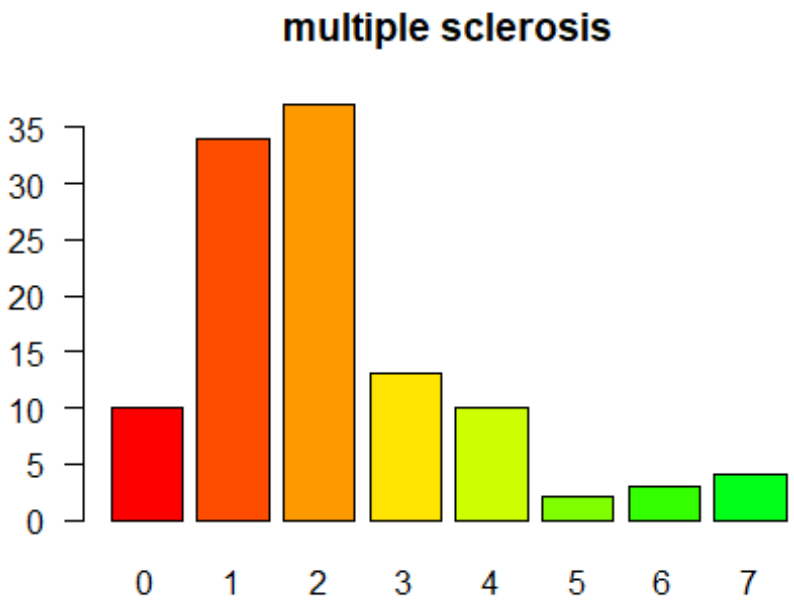
Disease and number of children

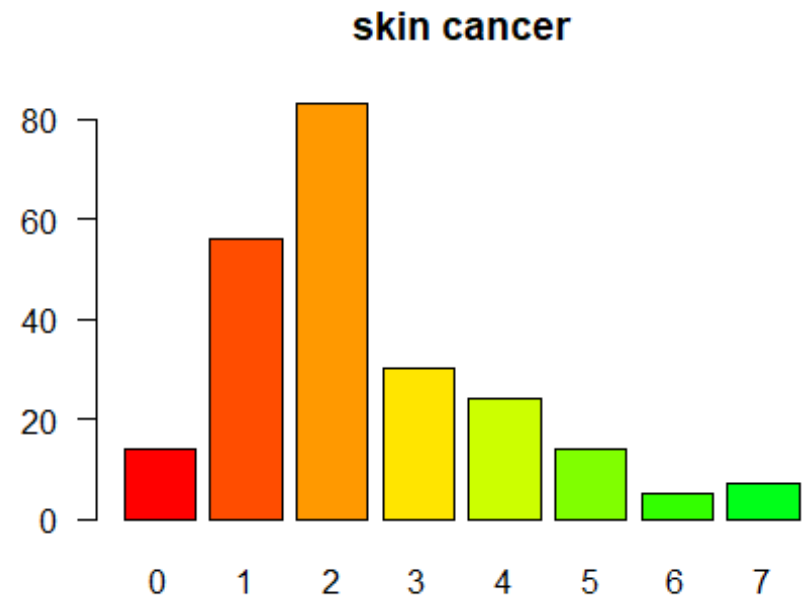
```
for (d in disease_name) {
  child_disease_counts <- subset(patients, patients$disease == d)
  child_disease_counts <- table(child_disease_counts$children)
  barplot(child_disease_counts, main=d, col=rainbow(20), las=1)
}
```

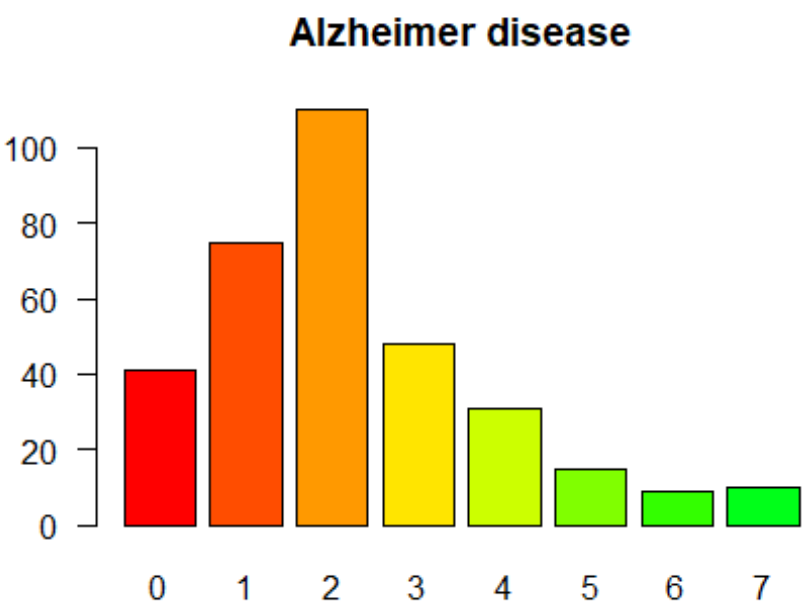

hypertension
80
60
40
20
0
0
1
2
3
4
5
6
7

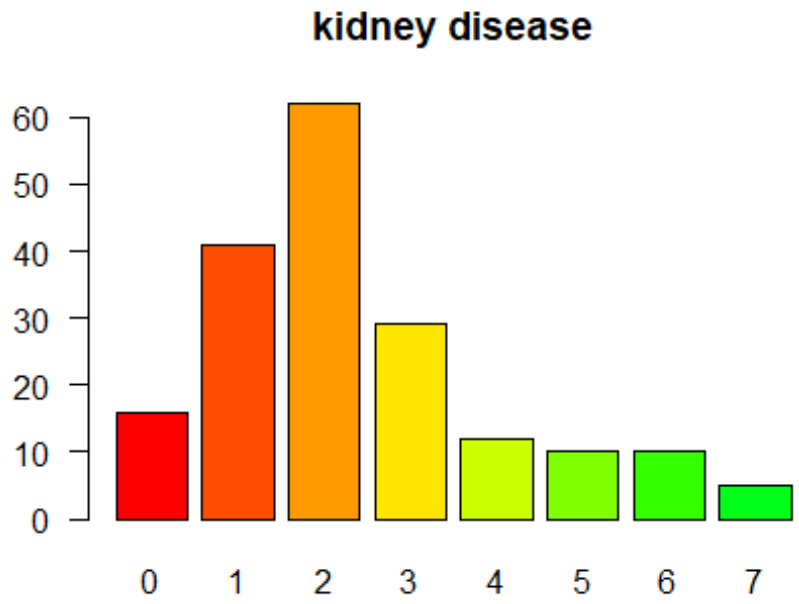

endometriosis
25
20
15
10
5
0
0
1
2
3
4
5
6
7

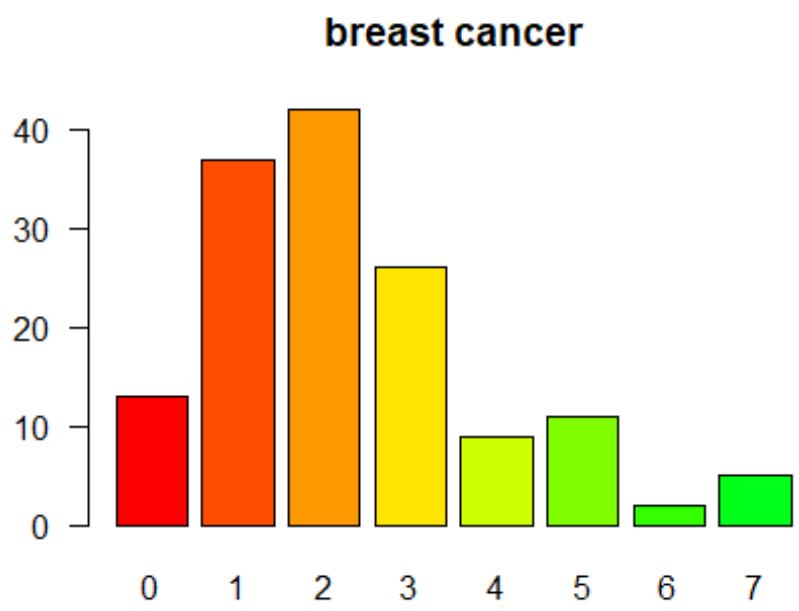

prostate cancer
50
40
30
20
10
0
0
1
2
3
4
5
6
7

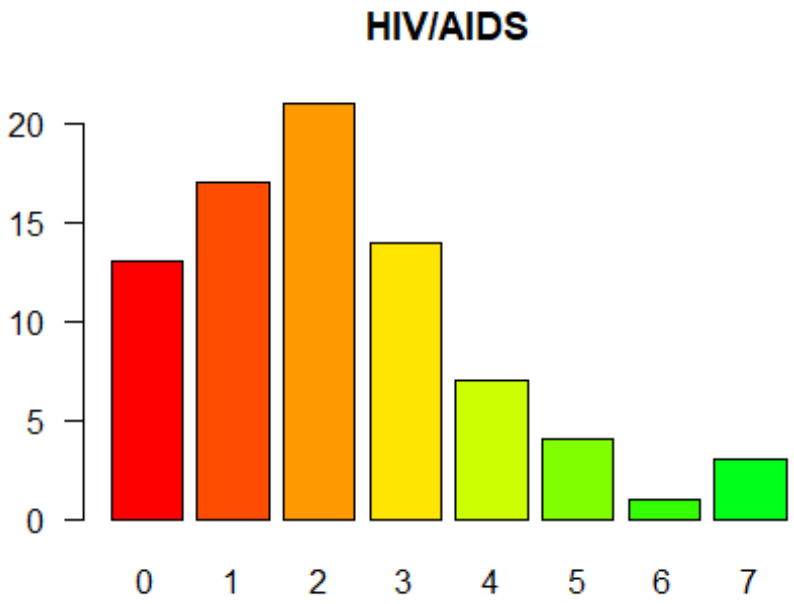

multiple sclerosis
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7

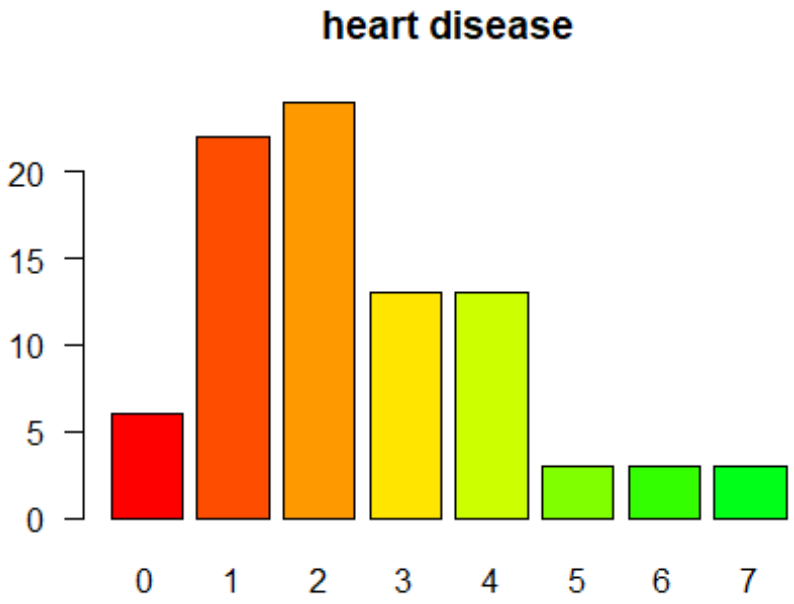

skin cancer
80
60
40
20
0
0
1
2
3
4
5
6
7

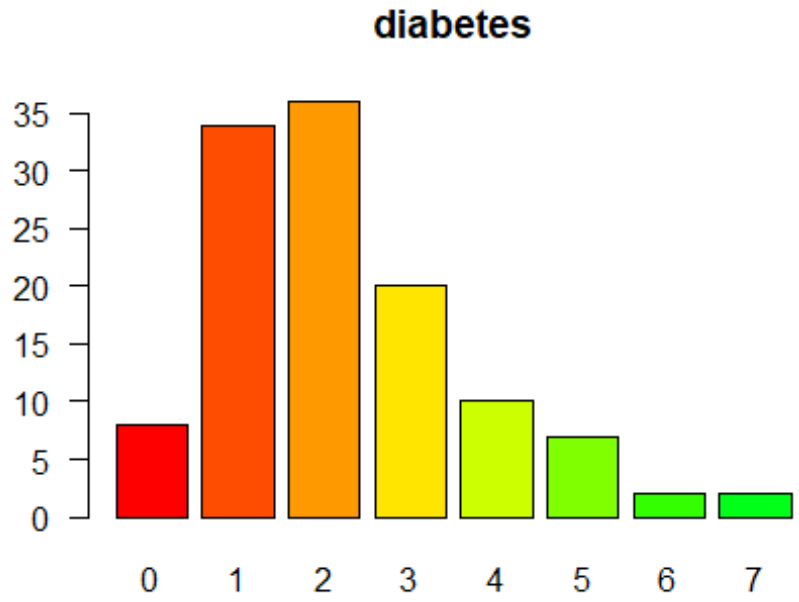

Alzheimer disease
100
80
60
40
20
0
0
1
2
3
4
5
6
7

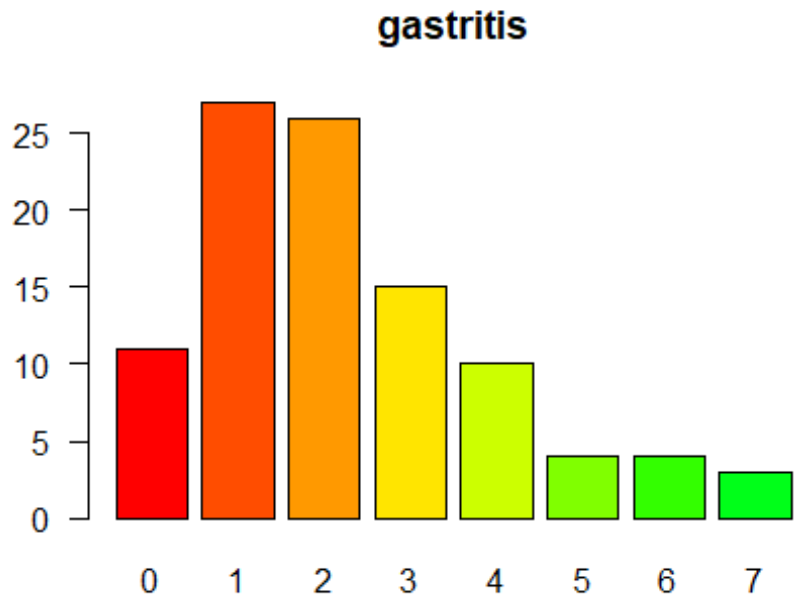

kidney disease
60
50
40
30
20
10
0
0
1
2
3
4
5
6
7

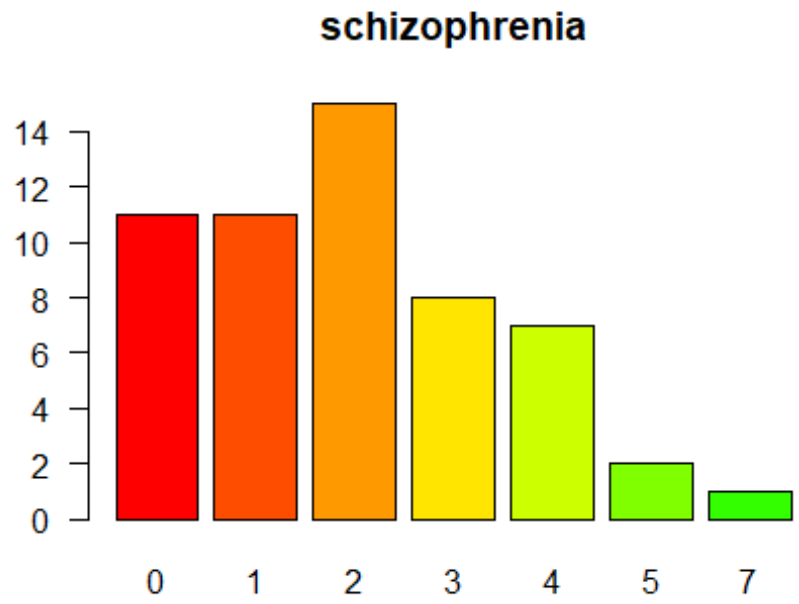

breast cancer
40
30
20
10
0
0
1
2
3
4
5
6
7


HIV/AIDS
20
15
10
5
0
0
1
2
3
4
5
6
7


heart disease
20
15
10
5
0
0
1
2
3
4
5
6
7


diabetes
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7

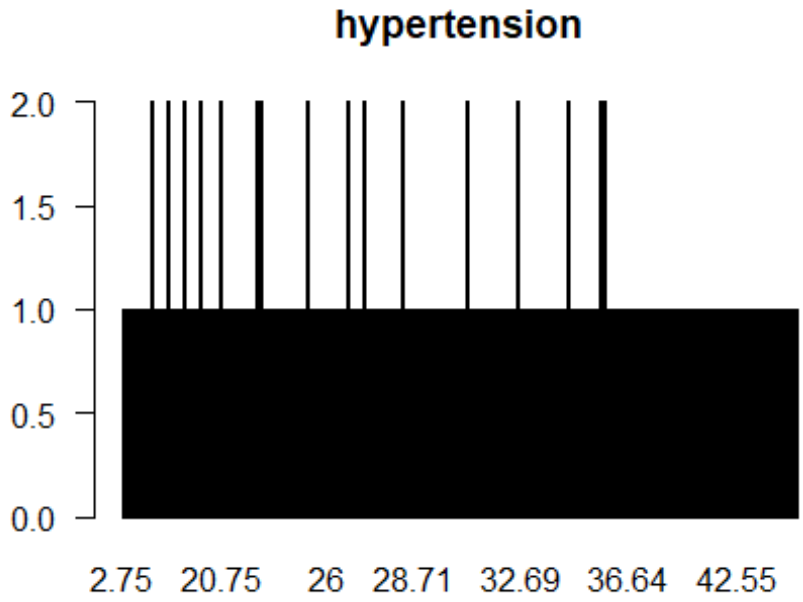

gastritis
25
20
15
10
5
0
0
1
2
3
4
5
6
7

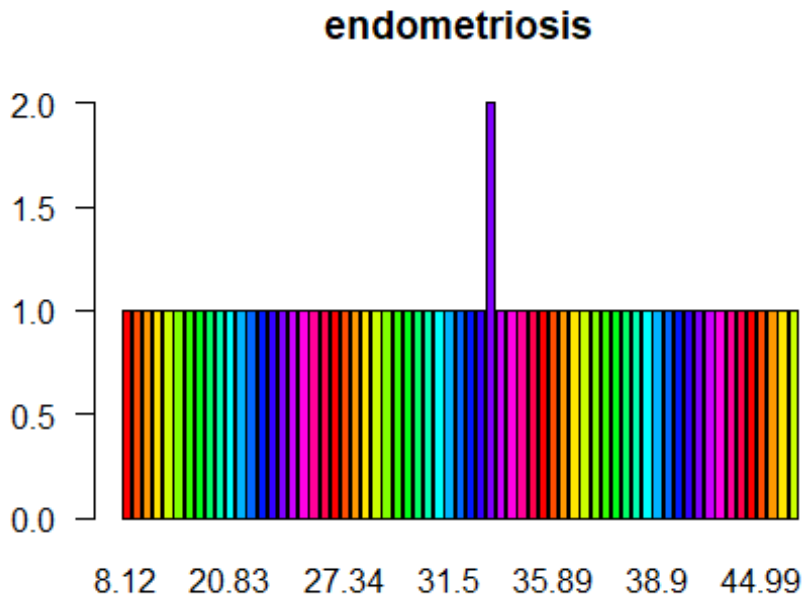


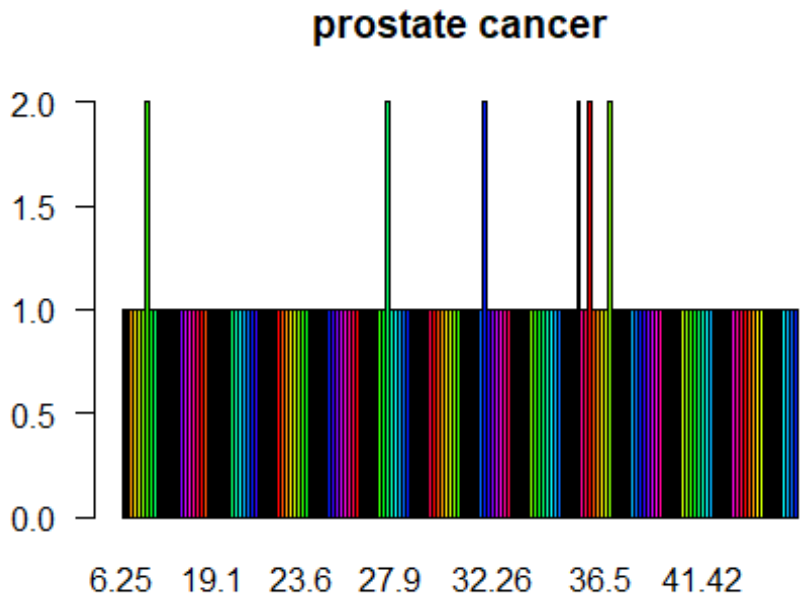
Observation :

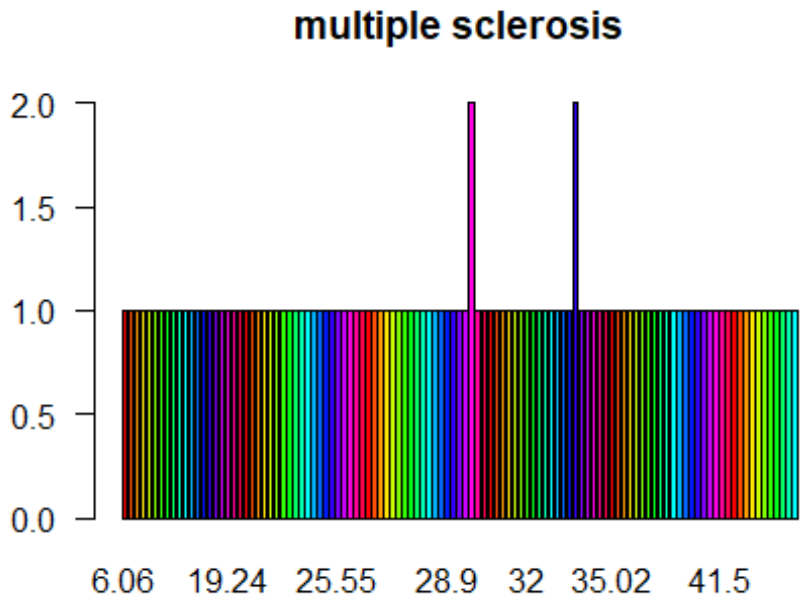
Disease and avg commute

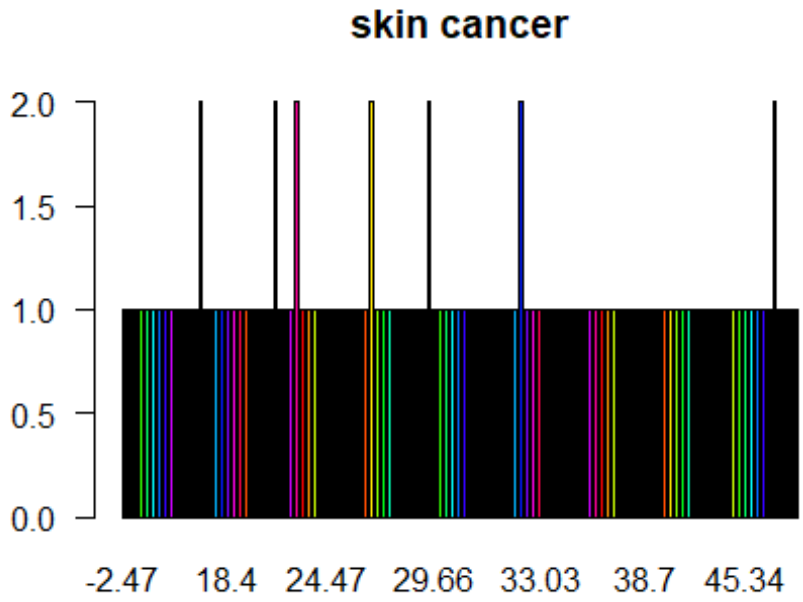
```
for (d in disease_name) {
  comm_disease_counts <- subset(patients, patients$disease == d)
  comm_disease_counts <- table(comm_disease_counts$avg_commute)
  barplot(comm_disease_counts, main=d, col=rainbow(20), las=1)
}
```

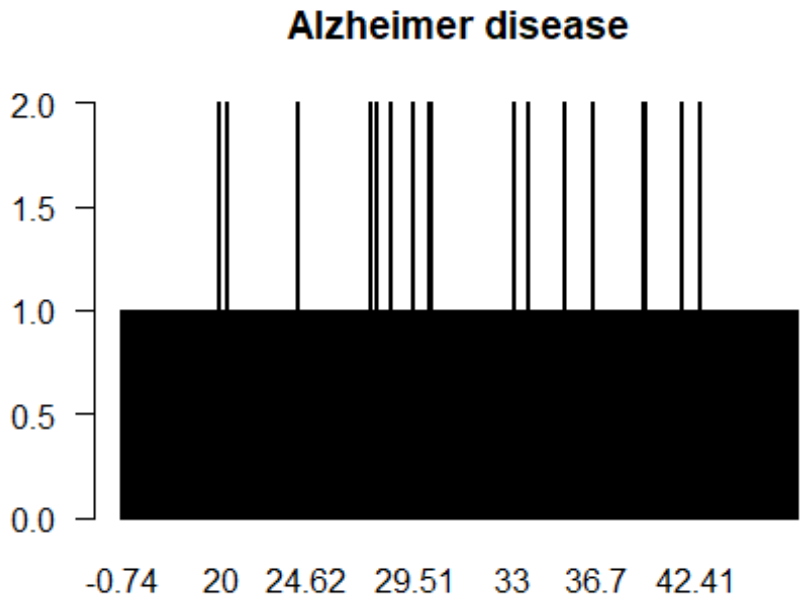

hypertension
2.0
1.5
1.0
0.5
0.0
2.75
20.75
26
28.71
32.69
36.64
42.55

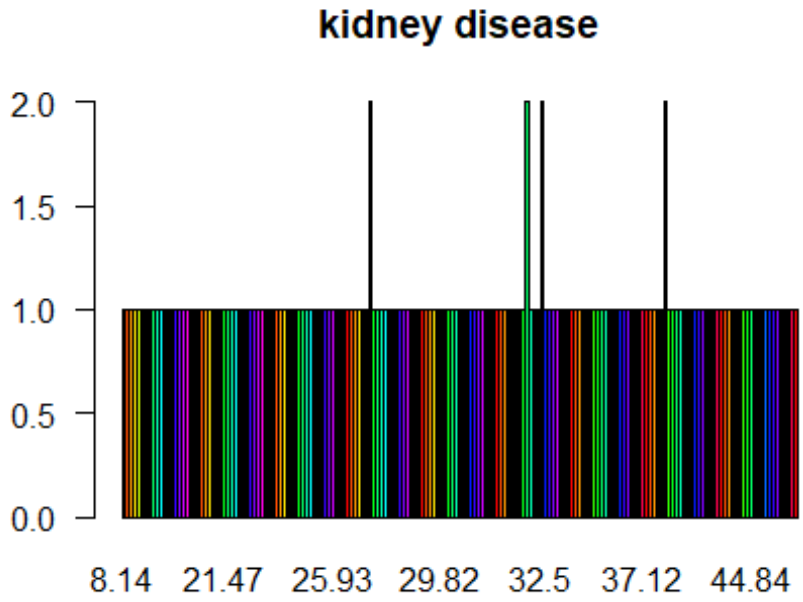

endometriosis
2.0
1.5
1.0
0.5
0.0
8.12
20.83
27.34
31.5
35.89
38.9
44.99

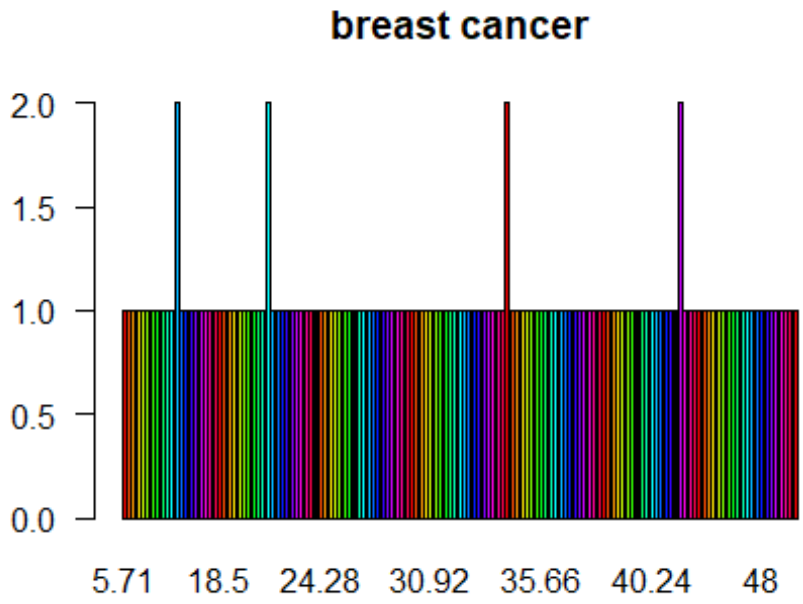

prostate cancer
2.0
1.5
1.0
0.5
0.0
6.25
19.1
23.6
27.9
32.26
36.5
41.42

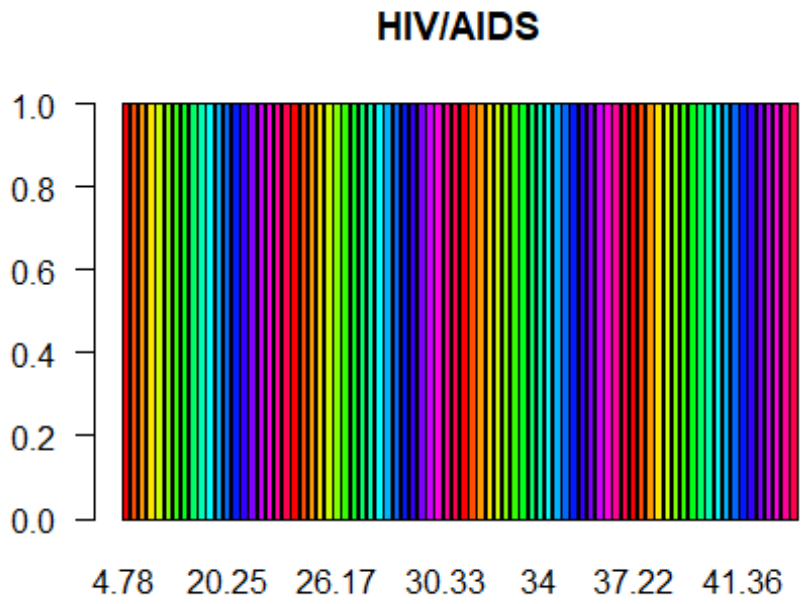

multiple sclerosis
2.0
1.5
1.0
0.5
0.0
6.06
19.24
25.55
28.9
32
35.02
41.5

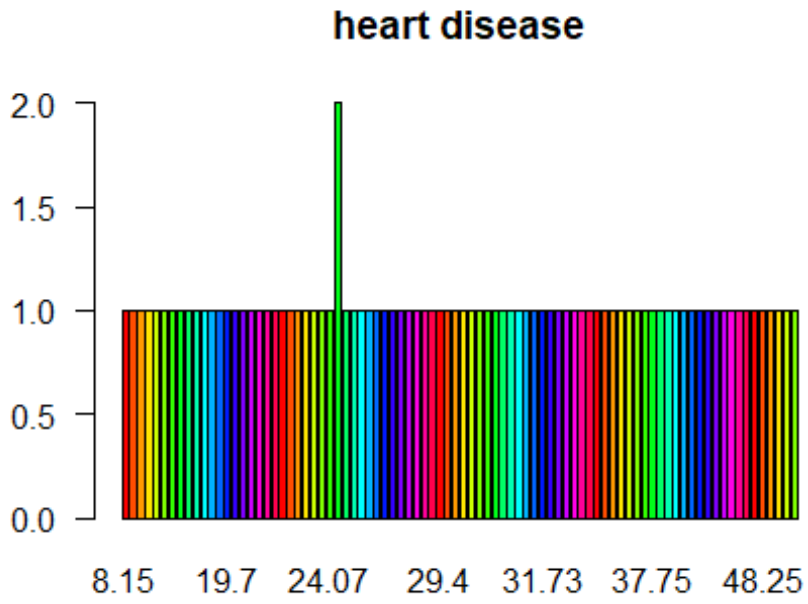

skin cancer
2.0
1.5
1.0
0.5
0.0
-2.47
18.4
24.47
29.66
33.03
38.7
45.34

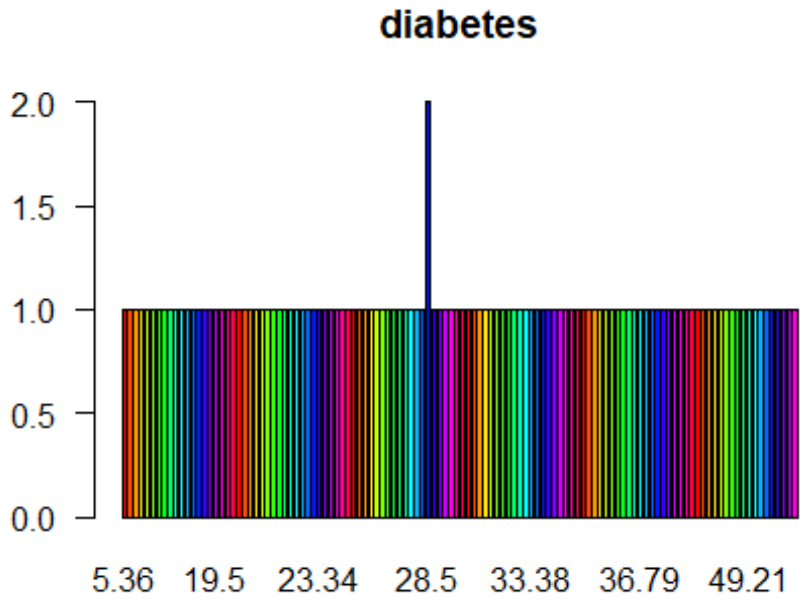

Alzheimer disease
2.0
1.5
1.0
0.5
0.0
-0.74
20
24.62
29.51
33
36.7
42.41

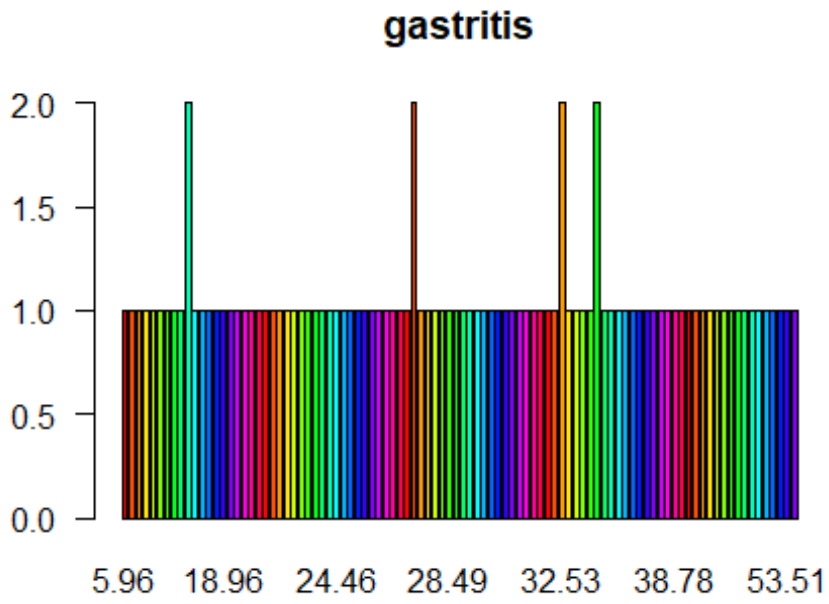

kidney disease
2.0
1.5
1.0
0.5
0.0
8.14
21.47
25.93
29.82
32.5
37.12
44.84

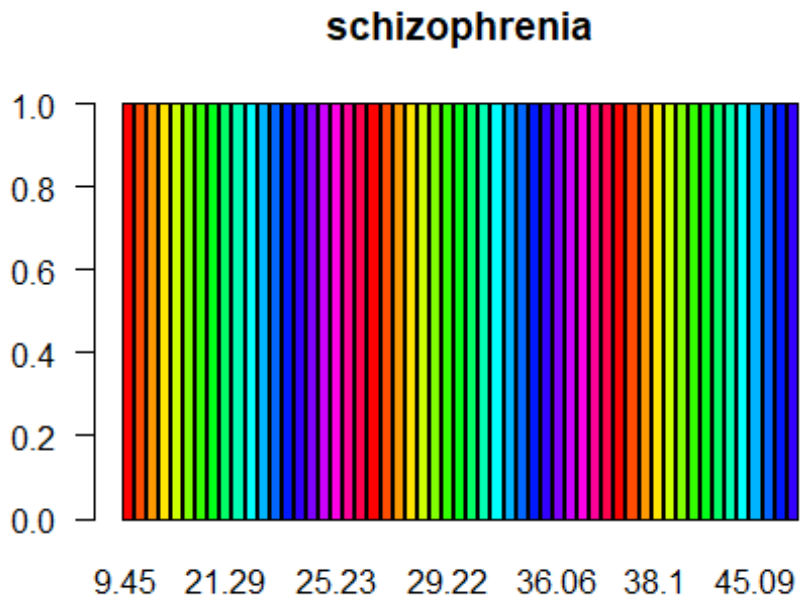

breast cancer
2.0
1.5
1.0
0.5
0.0
5.71
18.5
24.28
30.92
35.66
40.24
48


HIV/AIDS
1.0
0.8
0.6
0.4
0.2
0.0
4.78
20.25
26.17
30.33
34
37.22
41.36


heart disease
2.0
1.5
1.0
0.5
0.0
8.15
19.7
24.07
29.4
31.73
37.75
48.25


diabetes
2.0
1.5
1.0
0.5
0.0
5.36
19.5
23.34
28.5
33.38
36.79
49.21

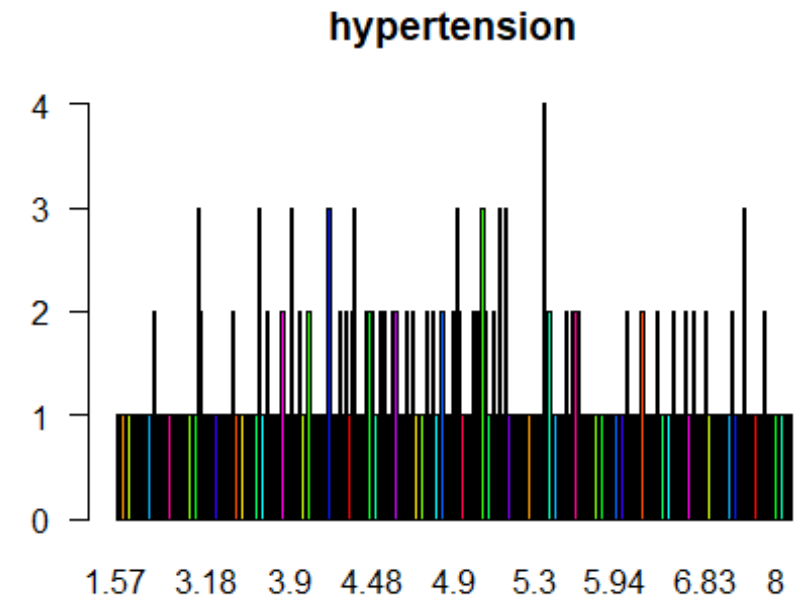

gastritis
2.0
1.5
1.0
0.5
0.0
5.96
18.96
24.46
28.49
32.53
38.78
53.51

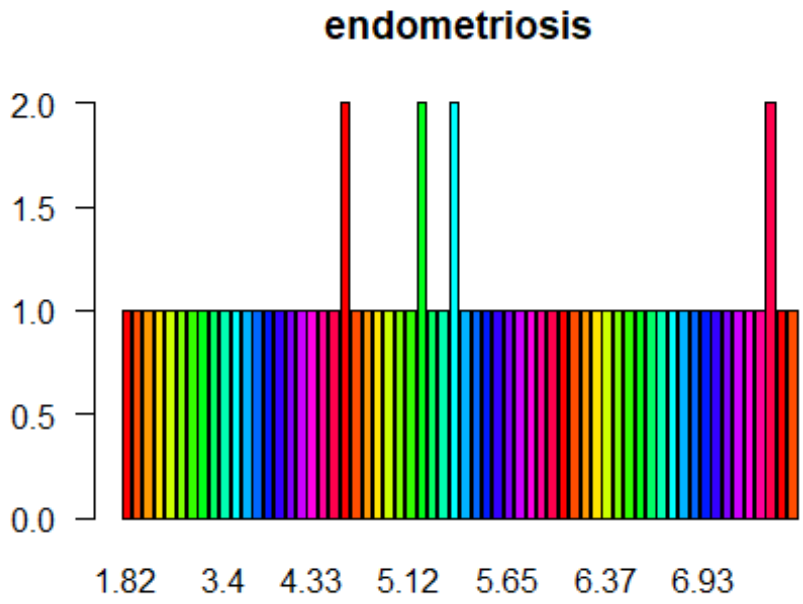


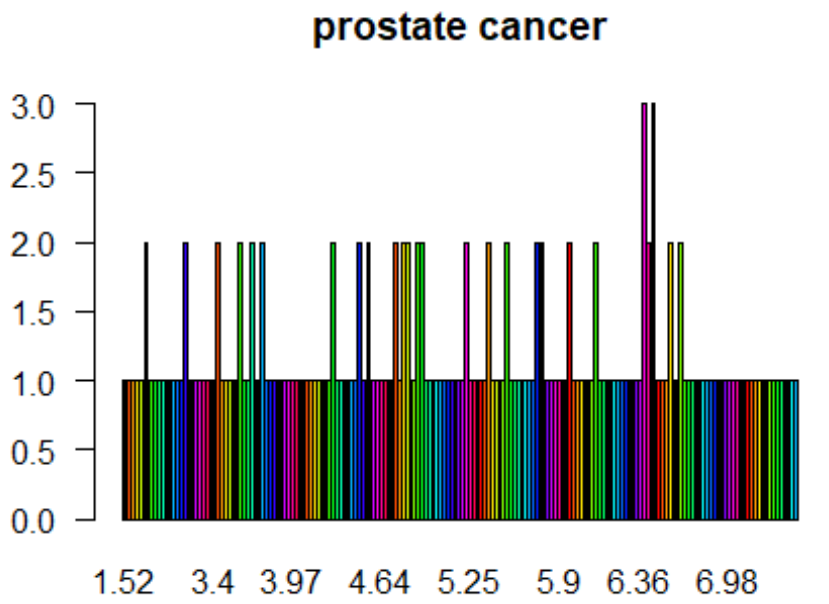
Observation :

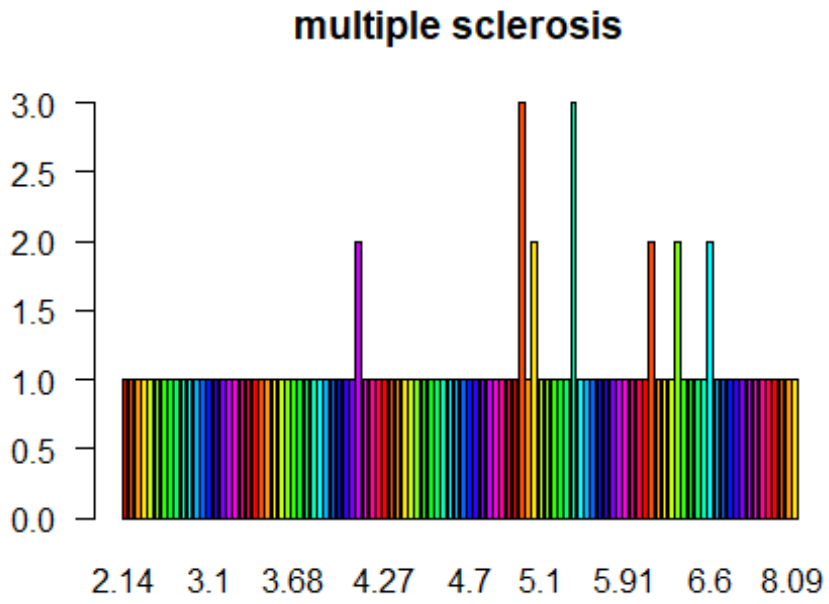
Disease and daily internet use

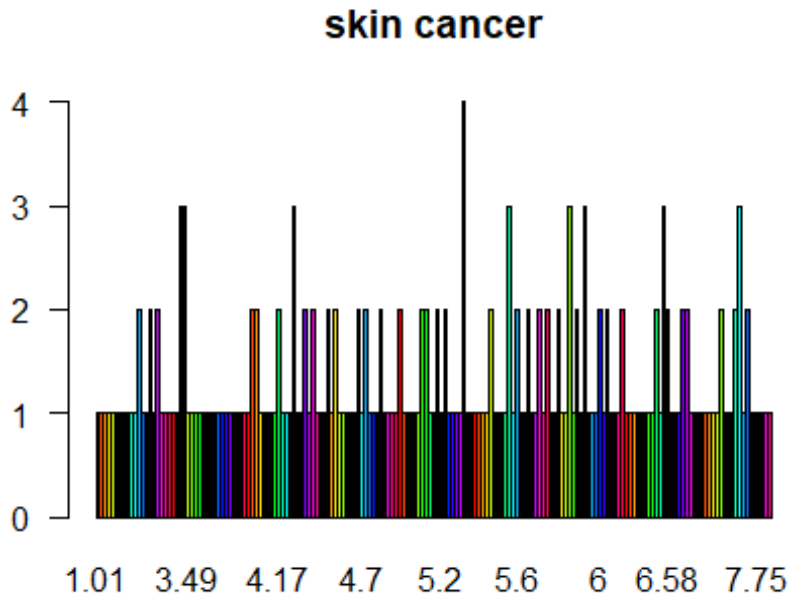
```
for (d in disease_name) {
  net_disease_counts <- subset(patients, patients$disease == d)
  net_disease_counts <- table(net_disease_counts$daily_internet_use)
  barplot(net_disease_counts, main=d, col=rainbow(20), las=1)
}
```

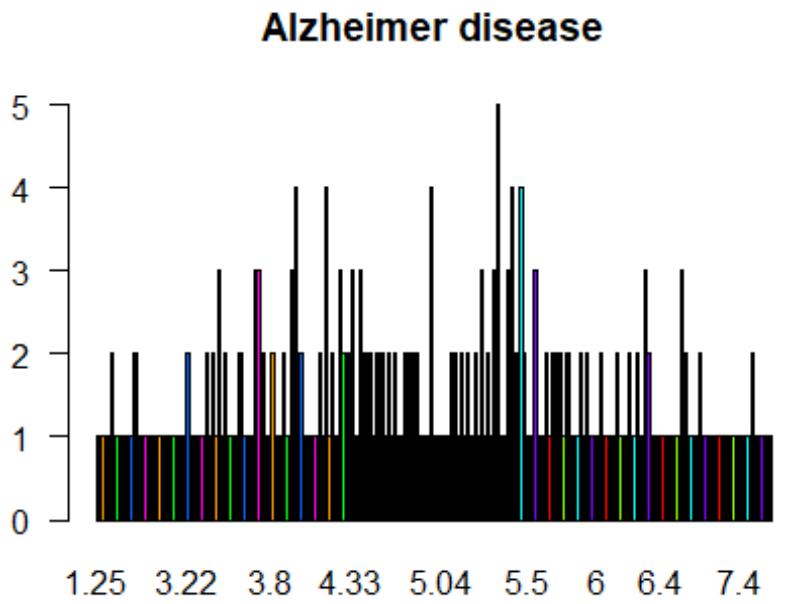

hypertension
4
3
2
1
0
1.57
3.18
3.9
4.48
4.9
5.3
5.94
6.83
8

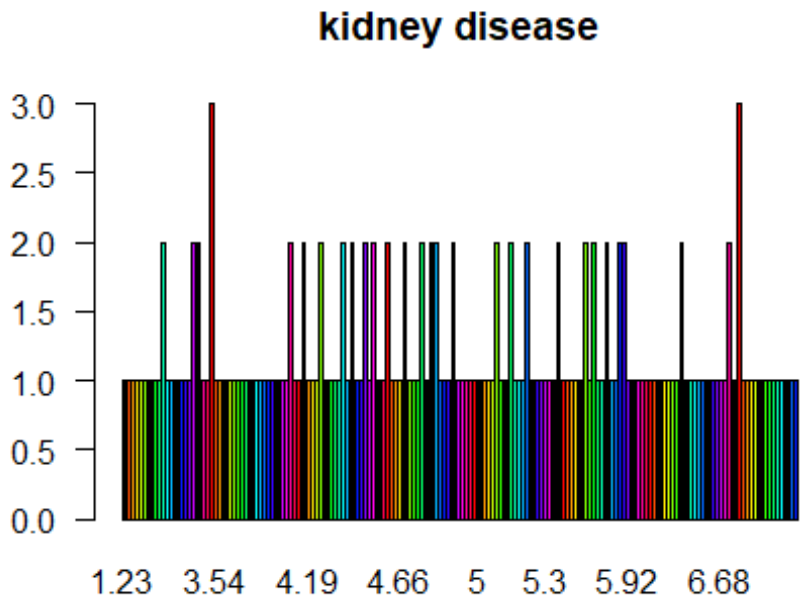

endometriosis
2.0
1.5
1.0
0.5
0.0
1.82
3.4
4.33
5.12
5.65
6.37
6.93

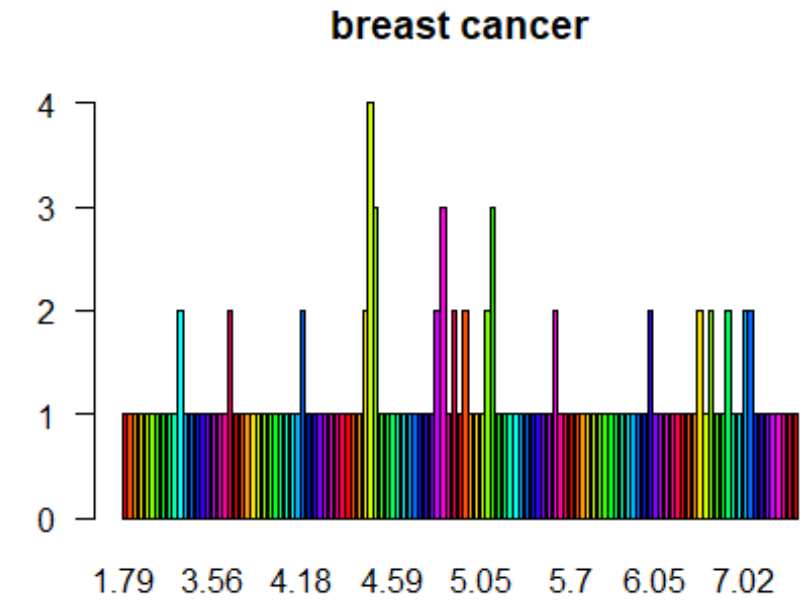

prostate cancer
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.52
3.4
3.97
4.64
5.25
5.9
6.36
6.98

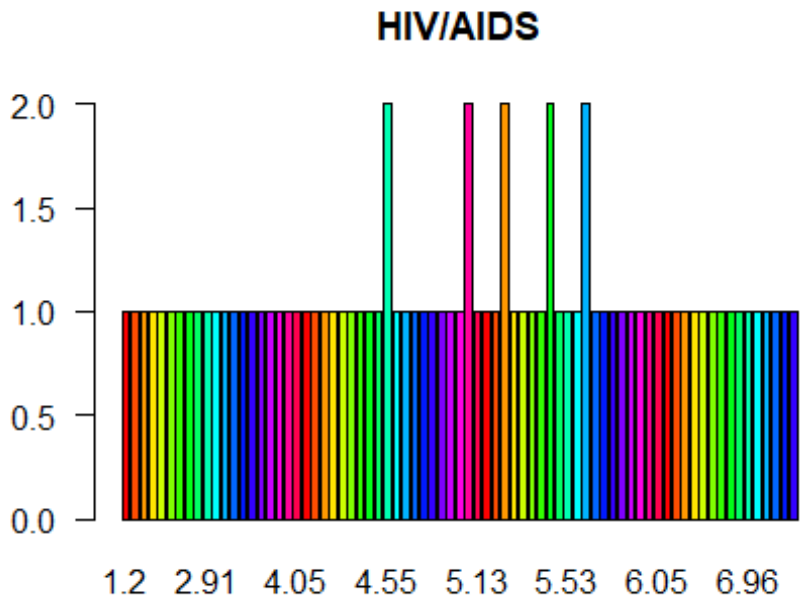

multiple sclerosis
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2.14
3.1
3.68
4.27
4.7
5.1
5.91
6.6
8.09

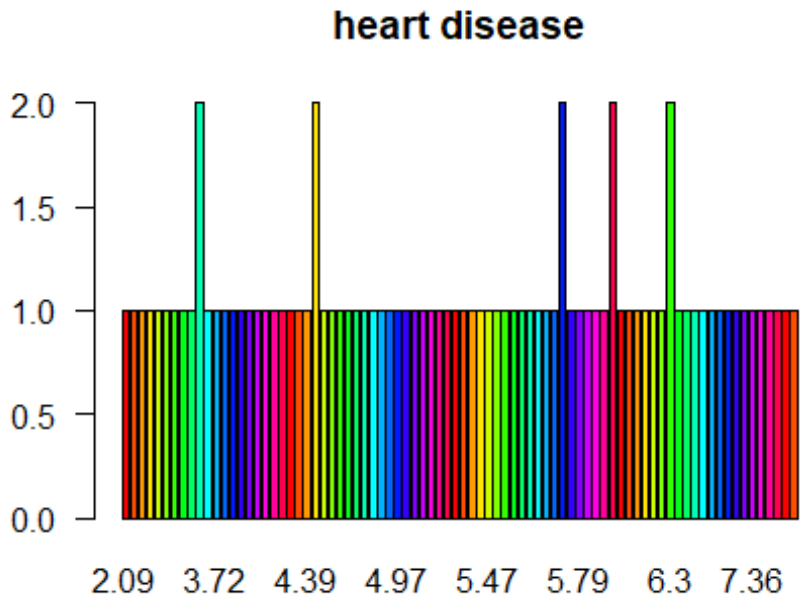

skin cancer
4
3
2
1
0
1.01
3.49
4.17
4.7
5.2
5.6
6
6.58
7.75

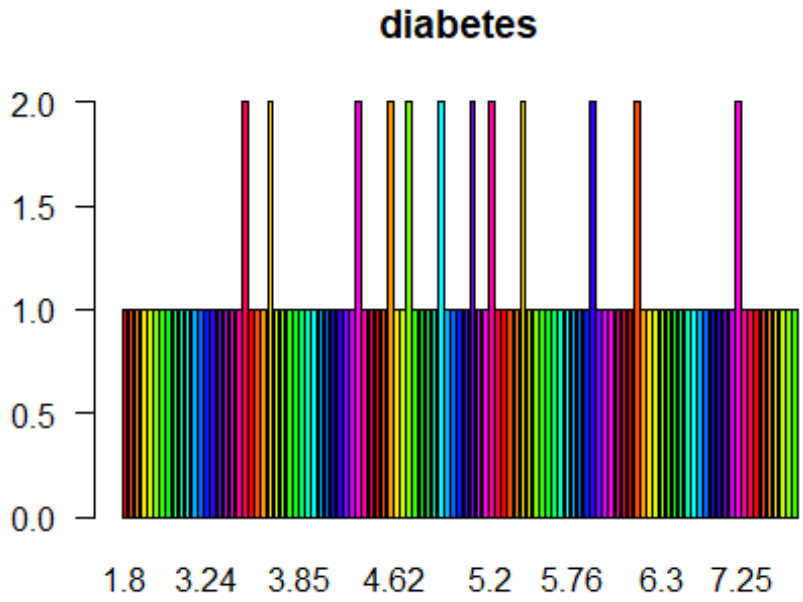

Alzheimer disease
5
4
3
2
1
0
1.25
3.22
3.8
4.33
5.04
5.5
6
6.4
7.4

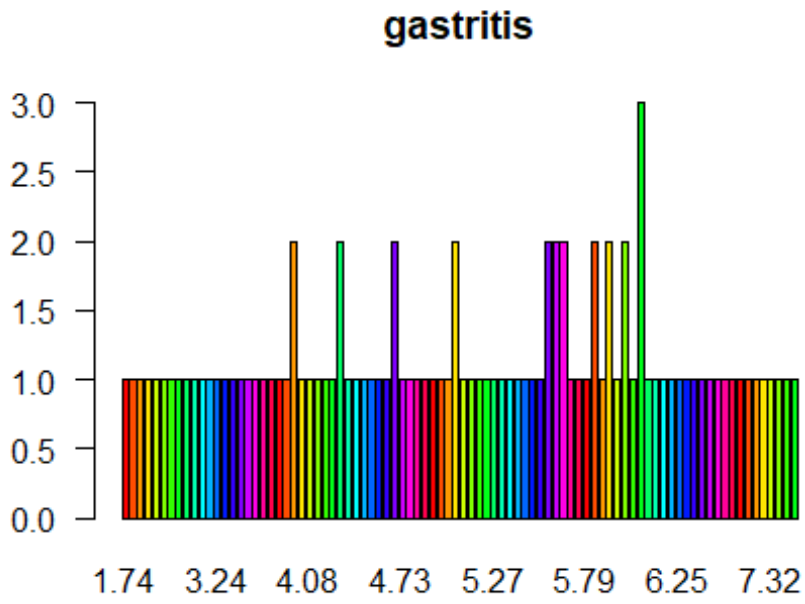

kidney disease
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.23
3.54
4.19
4.66
5
5.3
5.92
6.68

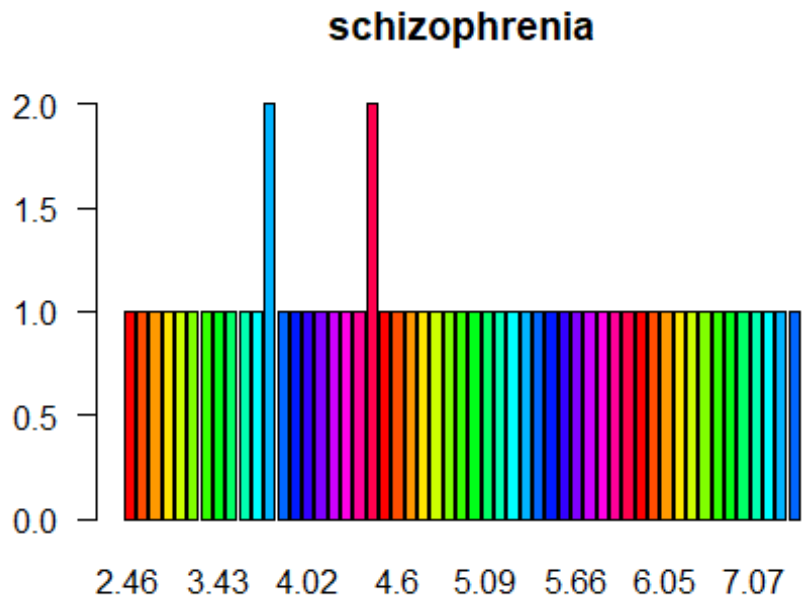

breast cancer
4
3
2
1
0
1.79
3.56
4.18
4.59
5.05
5.7
6.05
7.02


HIV/AIDS
2.0
1.5
1.0
0.5
0.0
1.2
2.91
4.05
4.55
5.13
5.53
6.05
6.96


heart disease
2.0
1.5
1.0
0.5
0.0
2.09
3.72
4.39
4.97
5.47
5.79
6.3
7.36


diabetes
2.0
1.5
1.0
0.5
0.0
1.8
3.24
3.85
4.62
5.2
5.76
6.3
7.25

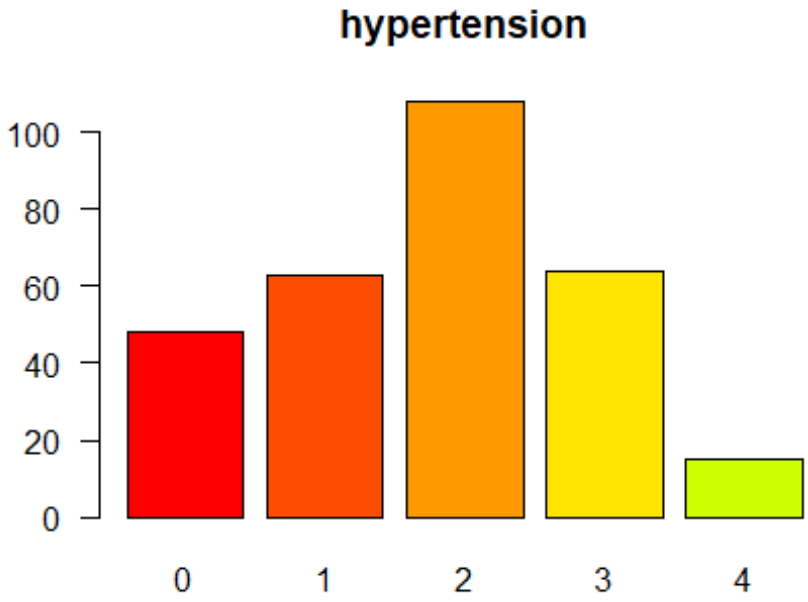

gastritis
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.74
3.24
4.08
4.73
5.27
5.79
6.25
7.32

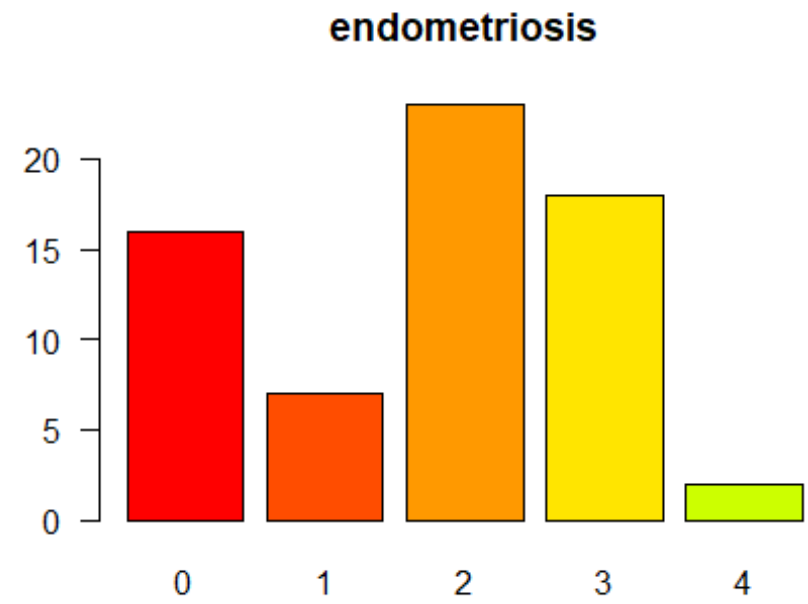


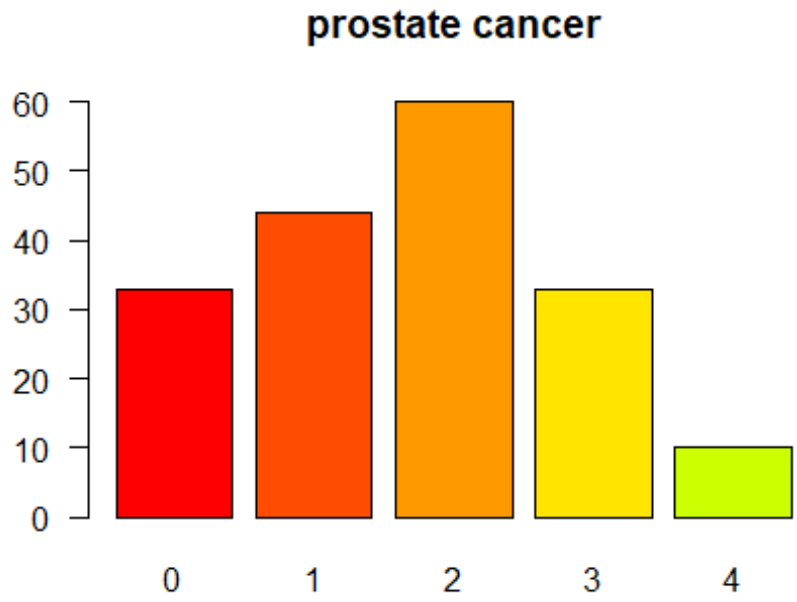
Observation :

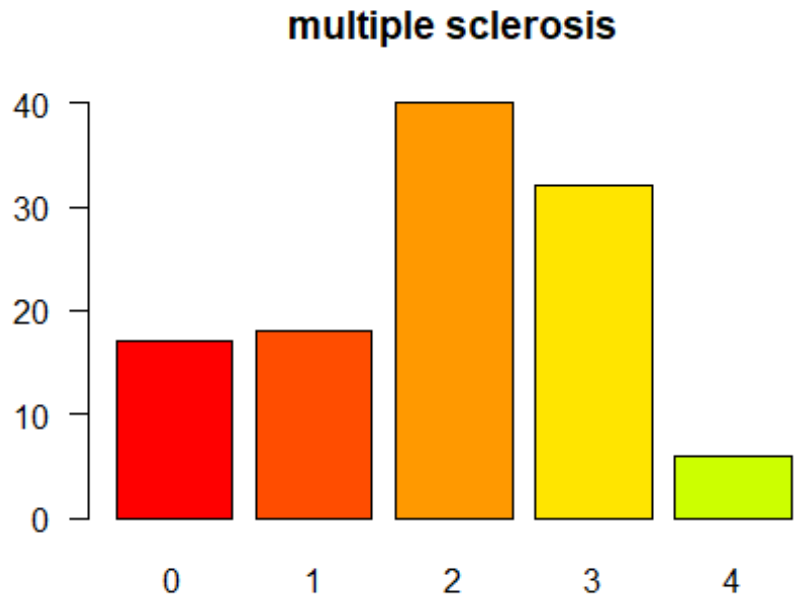
Disease and available vehicles

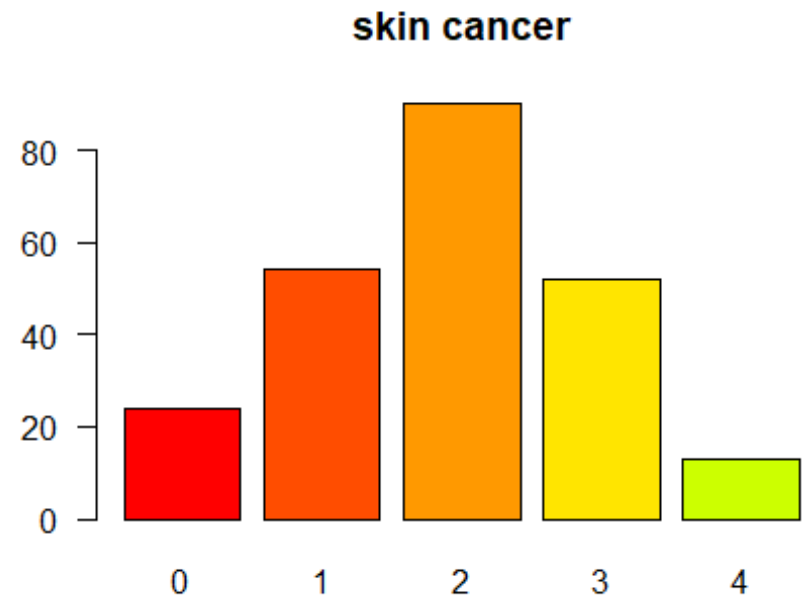
```
for (d in disease_name) {
  veh_disease_counts <- subset(patients, patients$disease == d)
  veh_disease_counts <- table(veh_disease_counts$available_vehicles)
  barplot(veh_disease_counts, main=d, col=rainbow(20), las=1)
}
```

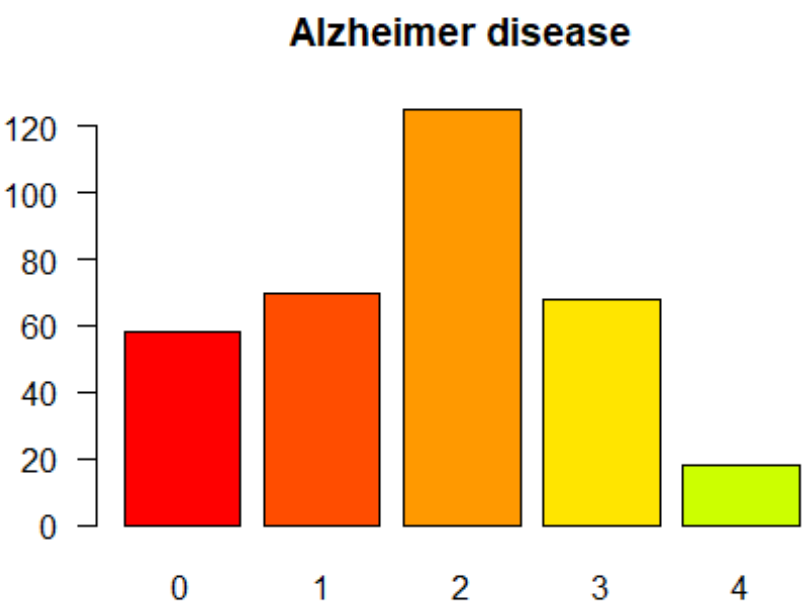

hypertension
100
80
60
40
20
0
0
1
2
3
4

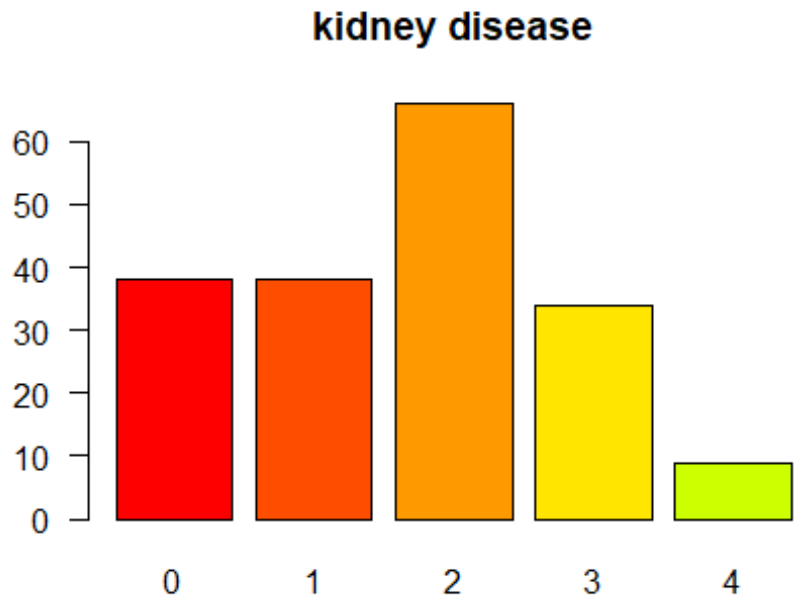

endometriosis
20
15
10
5
0
0
1
2
3
4

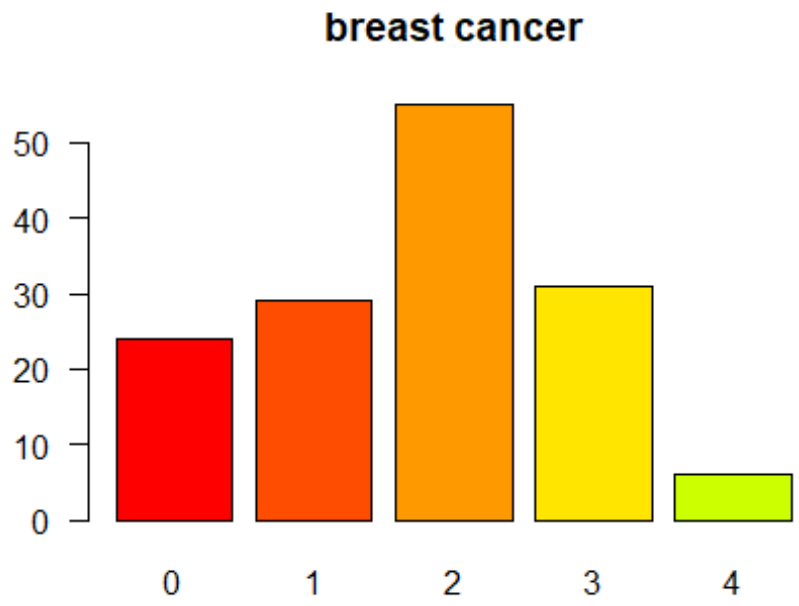

prostate cancer
60
50
40
30
20
10
0
0
1
2
3
4

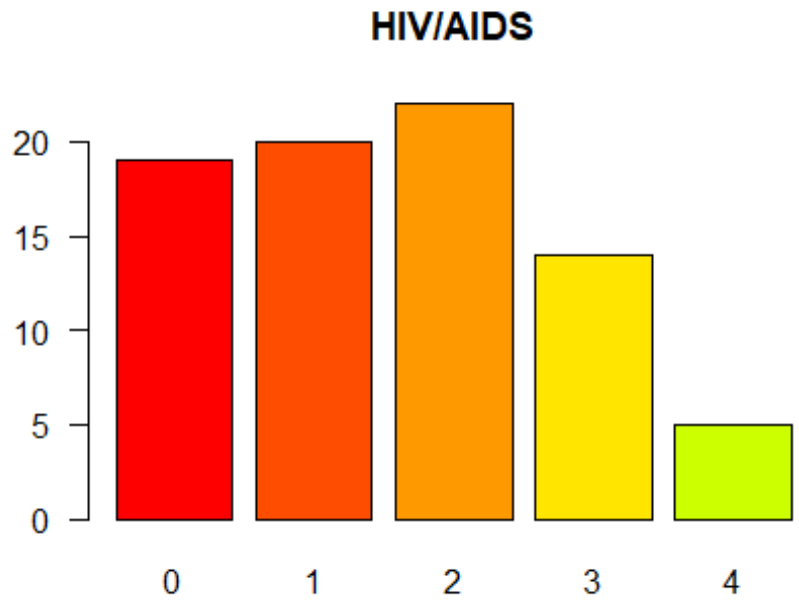

multiple sclerosis
40
30
20
10
0
0
1
2
3
4

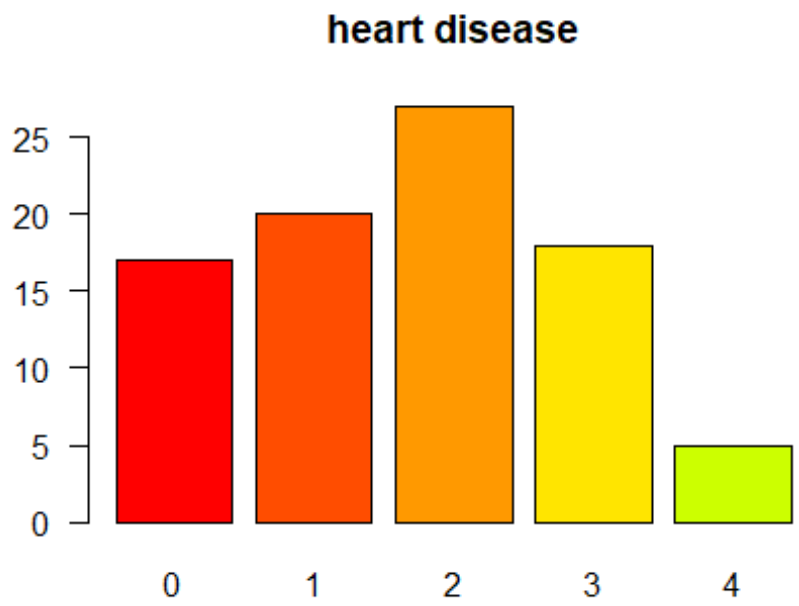

skin cancer
80
60
40
20
0
0
1
2
3
4

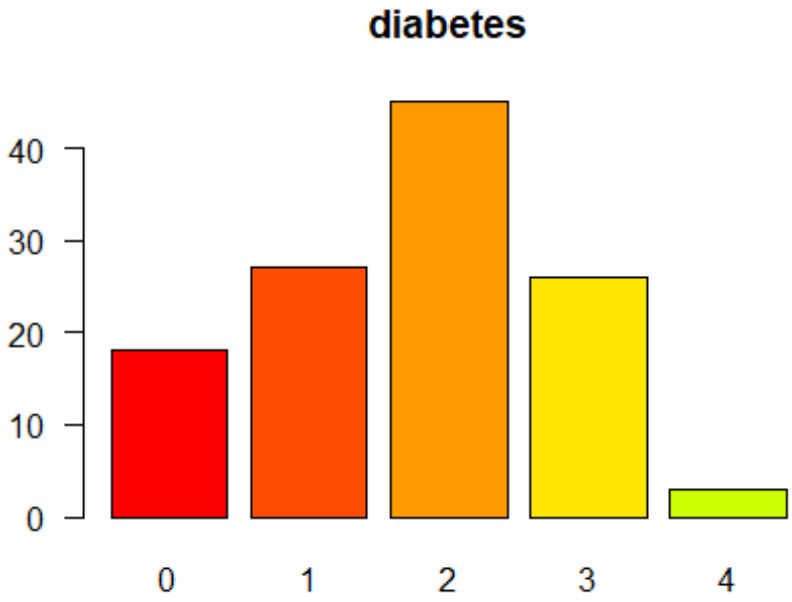

Alzheimer disease
120
100
80
60
40
20
0
0
1
2
3
4

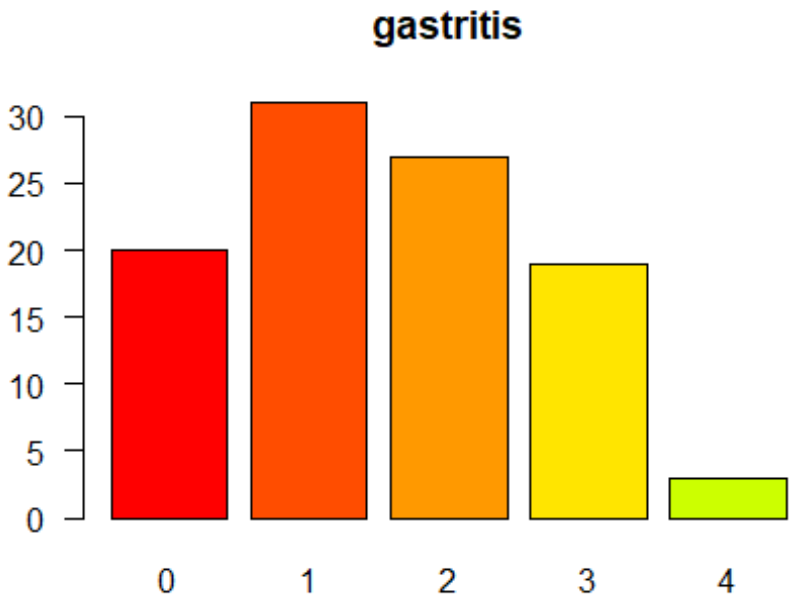

kidney disease
60
50
40
30
20
10
0
0
1
2
3
4

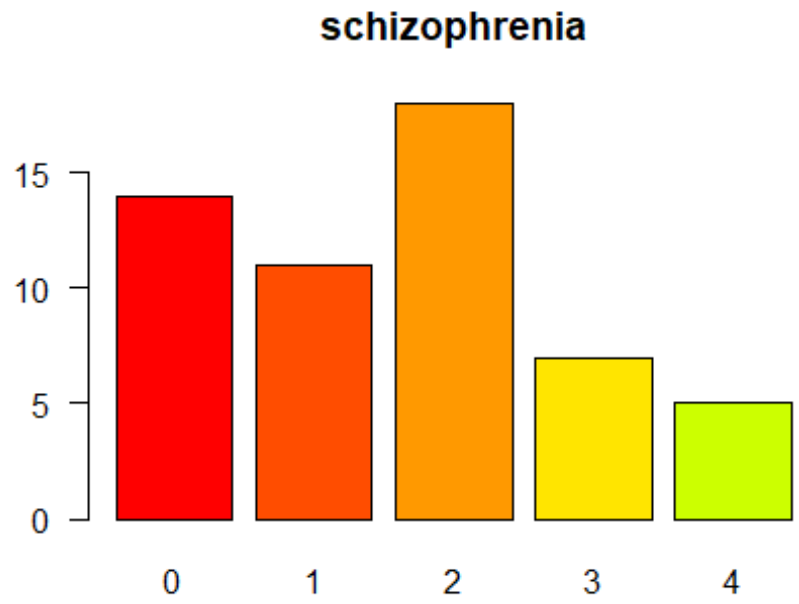

breast cancer
50
40
30
20
10
0
0
1
2
3
4


HIV/AIDS
20
15
10
5
0
0
1
2
3
4


heart disease
25
20
15
10
5
0
0
1
2
3
4


diabetes
40
30
20
10
0
0
1
2
3
4

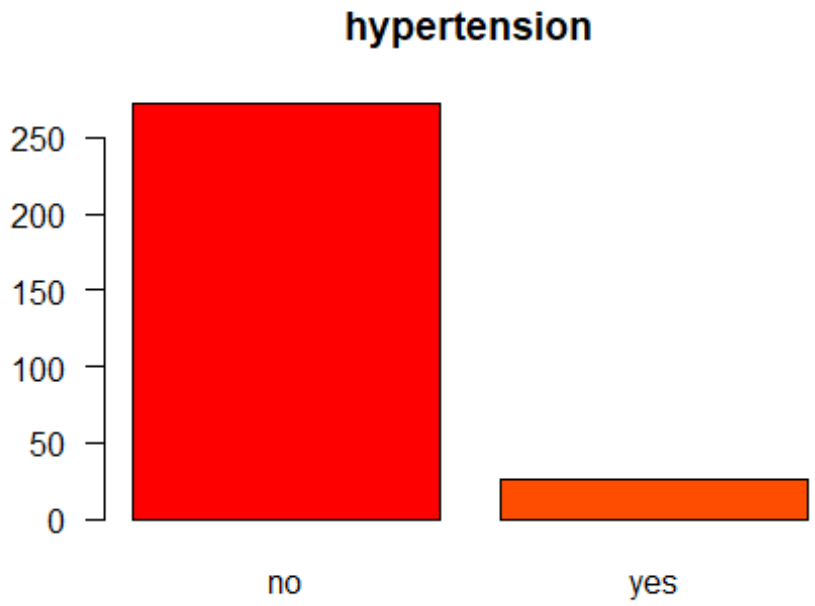

gastritis
30
25
20
15
10
5
0
0
1
2
3
4

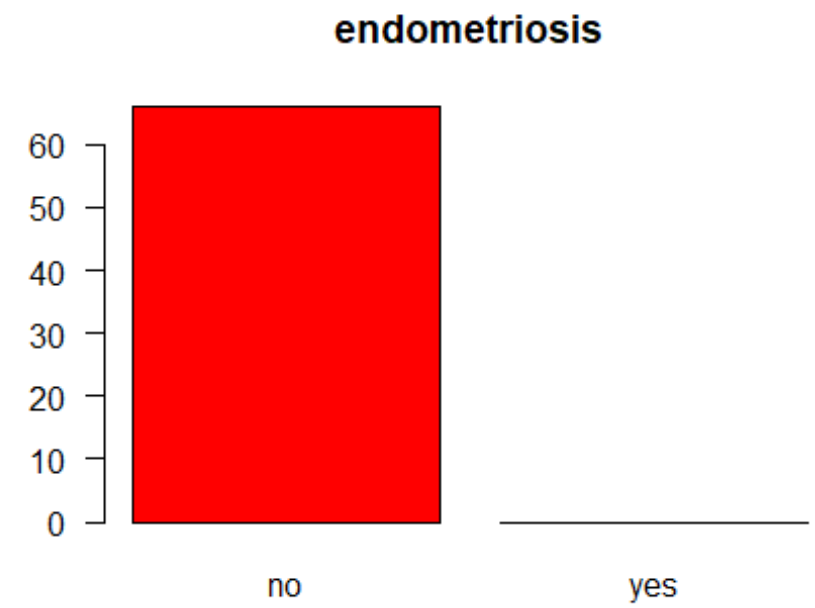


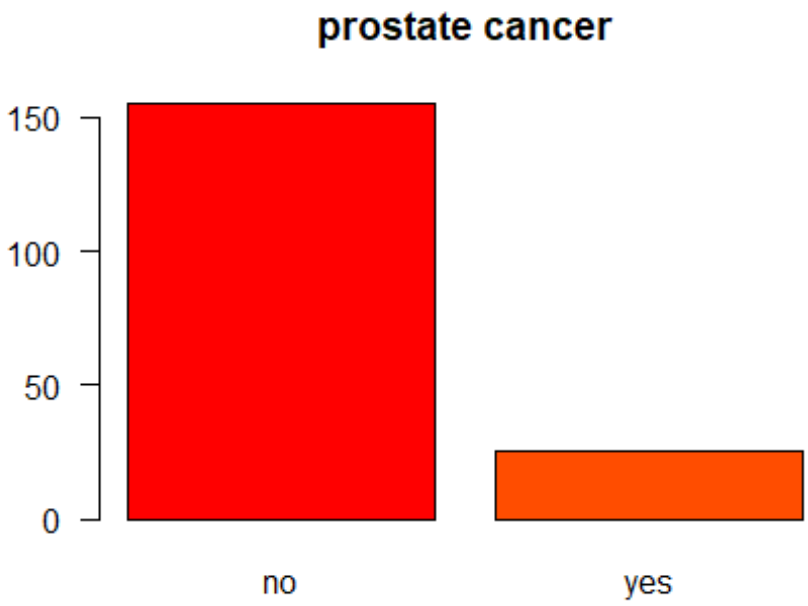
Observation :

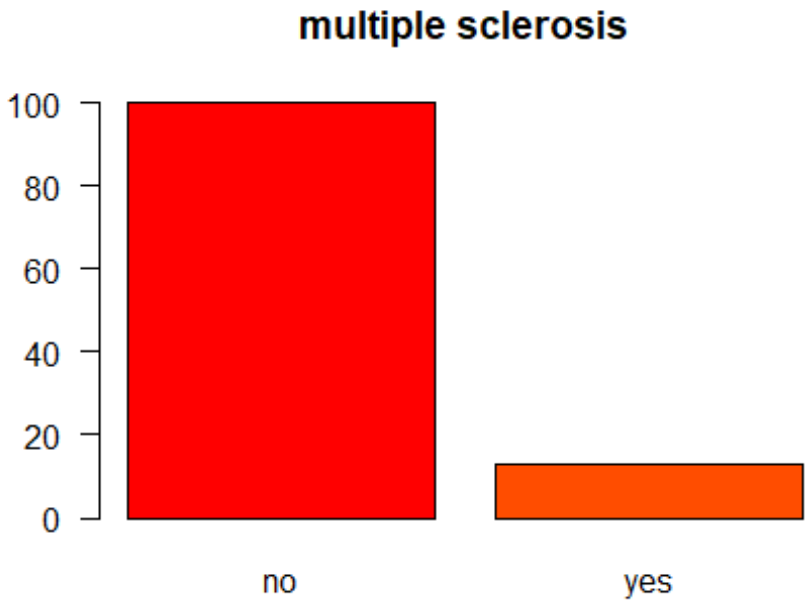
Disease and military service

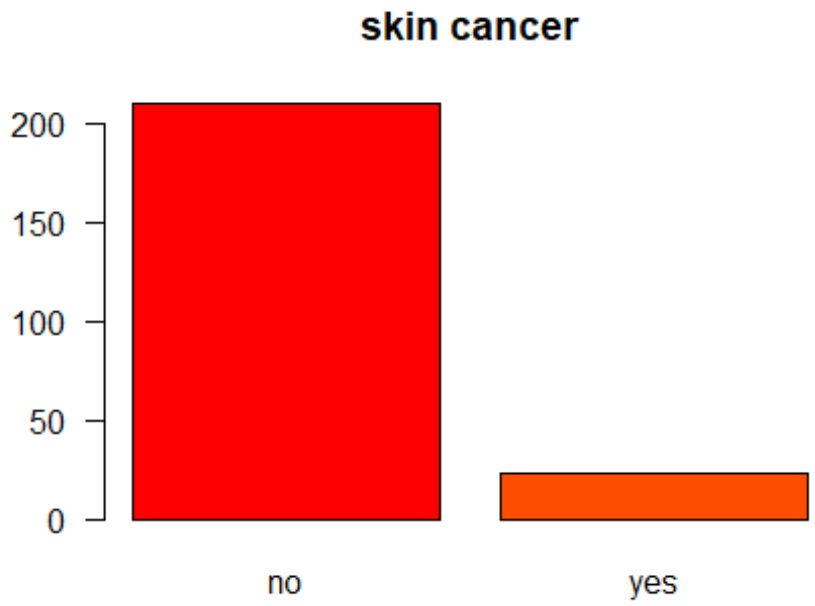
```
for (d in disease_name) {
  mil_disease_counts <- subset(patients, patients$disease == d)
  mil_disease_counts <- table(mil_disease_counts$military_service)
  barplot(mil_disease_counts, main=d, col=rainbow(20), las=1)
}
```

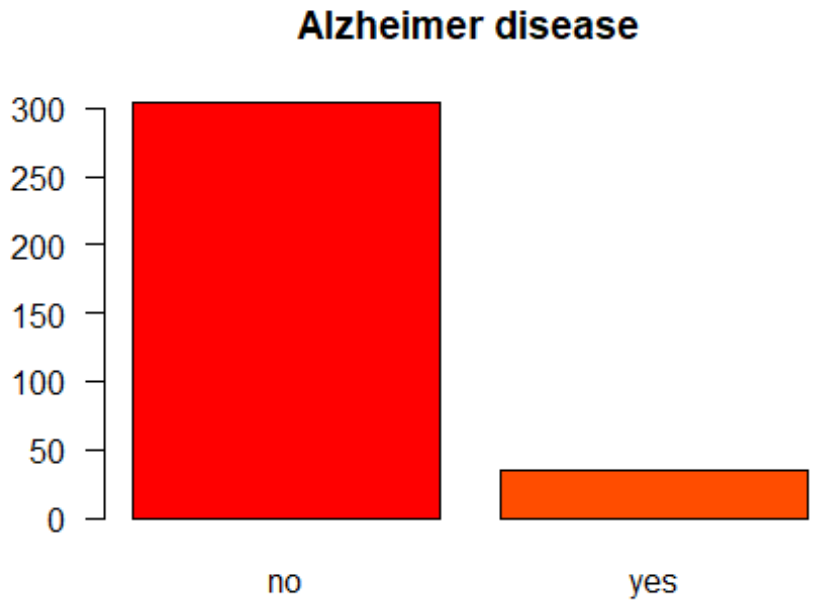

hypertension
250
200
150
100
50
0
no
yes

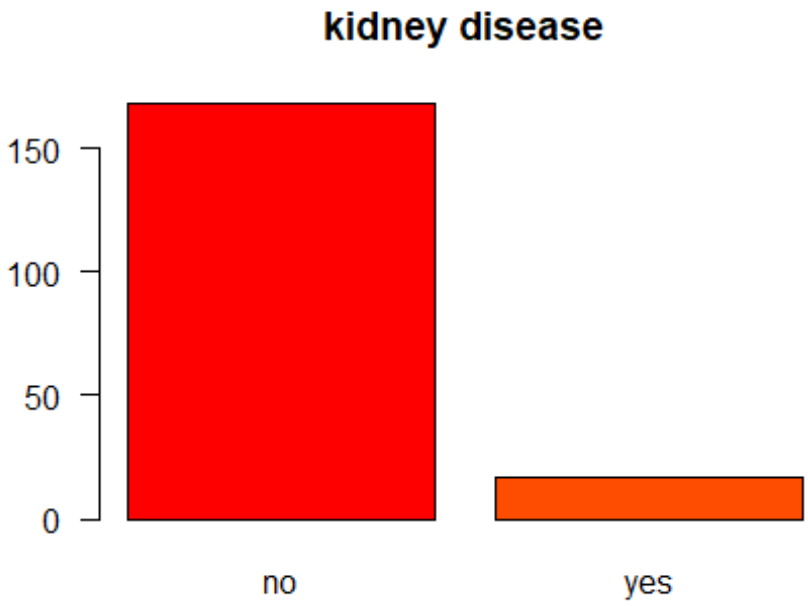

endometriosis
60
50
40
30
20
10
0
no
yes


prostate cancer
150
100
50
0
no
yes

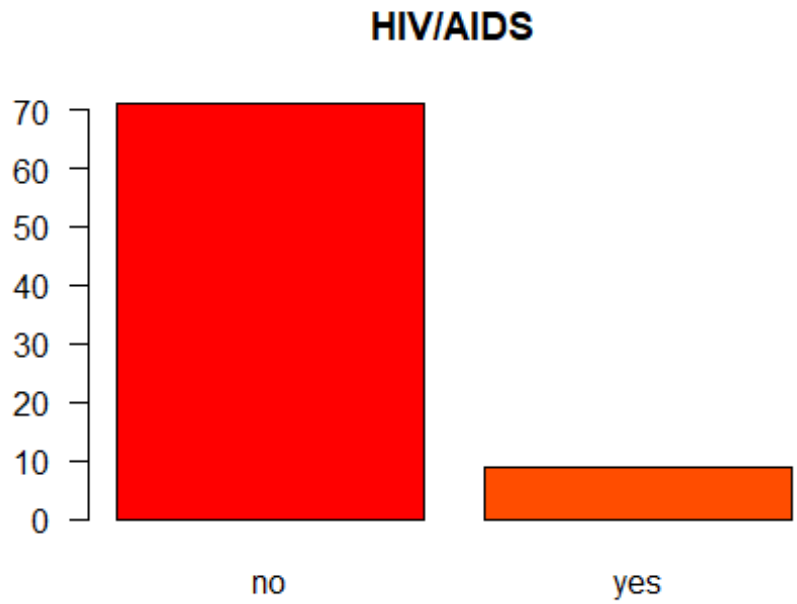

multiple sclerosis
100
80
60
40
20
0
no
yes


skin cancer
200
150
100
50
0
no
yes

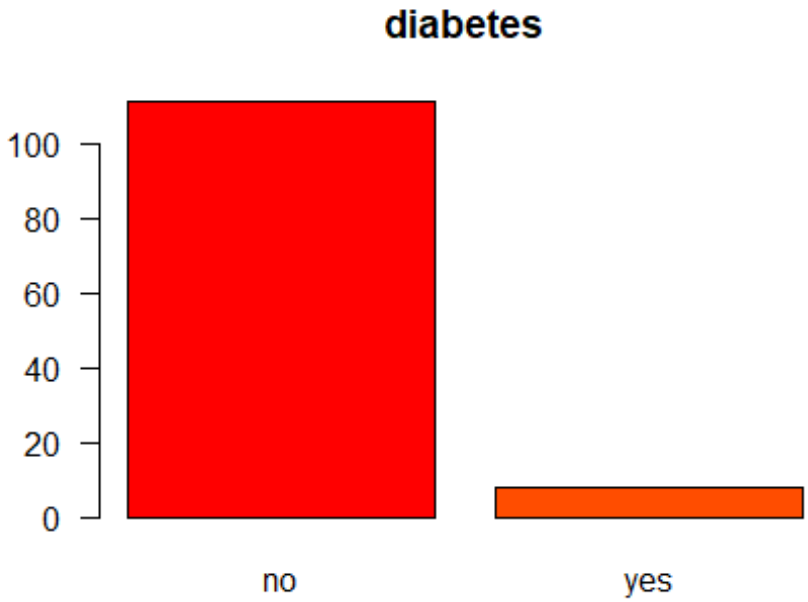

Alzheimer disease
300
250
200
150
100
50
0
no
yes

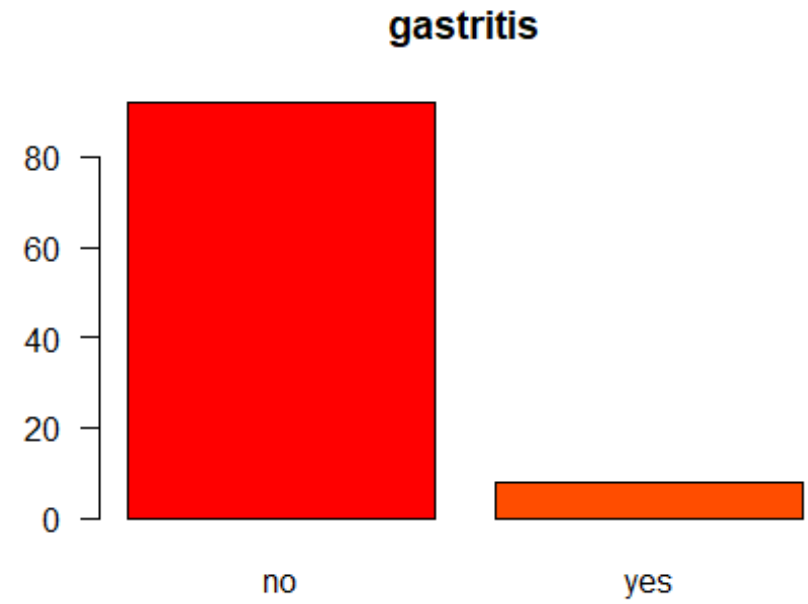

kidney disease
150
100
50
0
no
yes

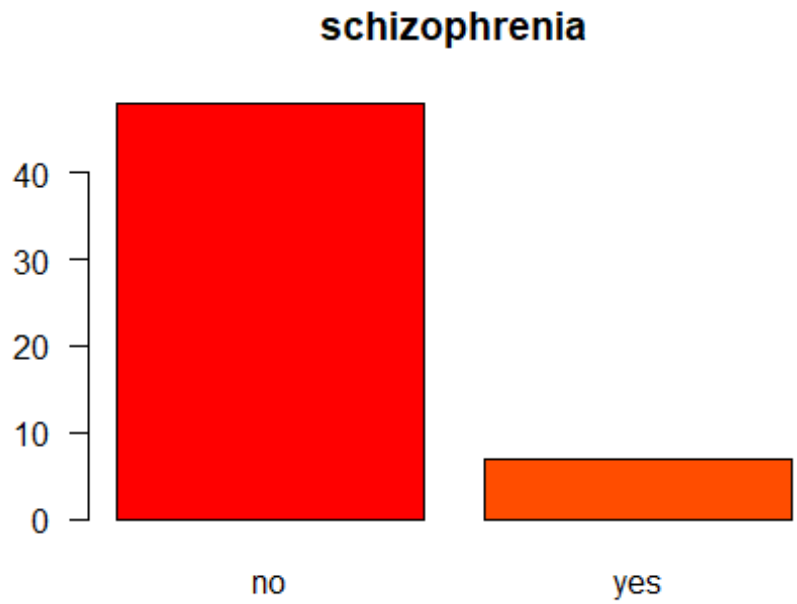

breast cancer
140
120
100
80
60
40
20
0
no
yes


HIV/AIDS
70
60
50
40
30
20
10
0
no
yes


heart disease
70
60
50
40
30
20
10
0
no
yes


diabetes
100
80
60
40
20
0
no
yes


gastritis
80
60
40
20
0
no
yes



Feature Selection

Will do feature selection using two methods, Chi-squared and Cramer's V after splitting and balancing the dataset for three diseases (alzheimer, hypertension, skin cancer)

Feature the selection Alsheimer using Chi-squared

```
alzheimer_set <- select(patients, gender, age, employment_status, education,
marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service, alzheimer)
FeatureTrain <- sample(nrow(alzheimer_set), 0.7*nrow(alzheimer_set), replace
= FALSE)
FeatureTrainSet <- alzheimer_set[FeatureTrain,]
FeatureValidSet <- alzheimer_set[-FeatureTrain,]

response <- as.factor(patients$alzheimer)
input <- select(patients, gender, age, employment_status, education,
marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service)

data <- ubOver(X=input, Y=response)
alzheime_os_dataset <- cbind(data$X, class=data$Y)

chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$gender)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$gender
## X-squared = 0.030121, df = 1, p-value = 0.8622
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$age)
```

```
## Warning in chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$age):
## Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$age
## X-squared = 12.362, df = 3, p-value = 0.006241
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$education)
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$education
## X-squared = 1.2066, df = 3, p-value = 0.7514
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$marital_status)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$marital_status
## X-squared = 8.6472, df = 1, p-value = 0.003276
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$zipcode)
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$zipcode
## X-squared = 48.141, df = 12, p-value = 2.953e-06
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$employment_status)
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$employment_status
## X-squared = 37.411, df = 3, p-value = 3.767e-08
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$children)
```

```
##
##  Pearson's Chi-squared test
```

```
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$children
## X-squared = 17.862, df = 7, p-value = 0.01261
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$ancestry)
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$ancestry
## X-squared = 17.201, df = 3, p-value = 0.0006427
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$avg_commute)
```

```
## Warning in chisq.test(alzheime_os_dataset$class,
## alzheime_os_dataset$avg_commute): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and alzheime_os_dataset$avg_commute
## X-squared = 2853.5, df = 1520, p-value < 2.2e-16
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$daily_internet_use)
```

```
## Warning in chisq.test(alzheime_os_dataset$class,
## alzheime_os_dataset$daily_internet_use): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and
alzheime_os_dataset$daily_internet_use
## X-squared = 1612.5, df = 573, p-value < 2.2e-16
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$available_vehicles)
```

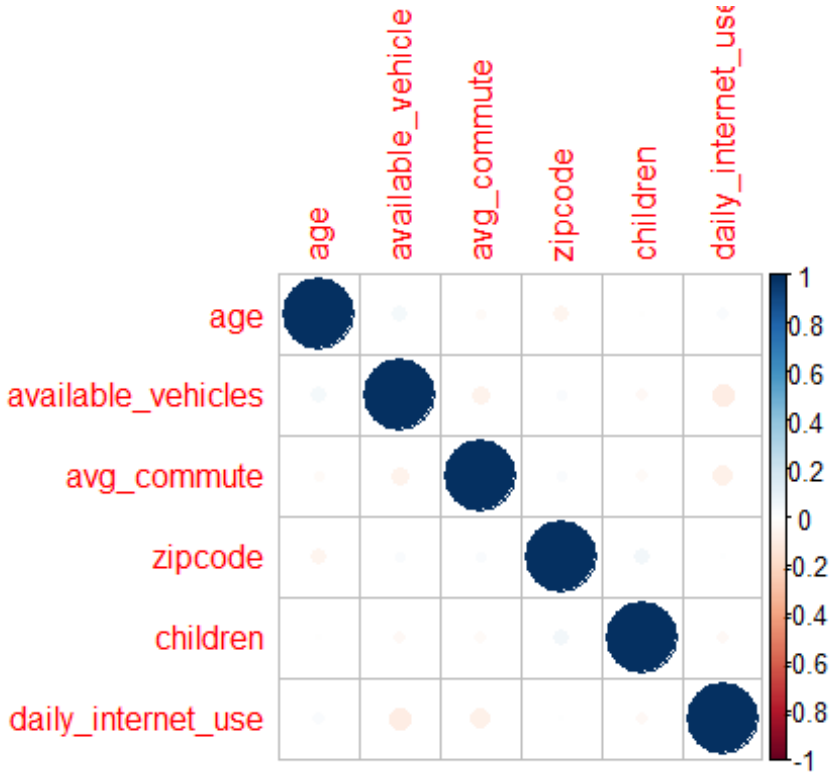
```
##
##  Pearson's Chi-squared test
##
## data:  alzheime_os_dataset$class and
alzheime_os_dataset$available_vehicles
## X-squared = 8.9663, df = 4, p-value = 0.06195
```

```
chisq.test(alzheime_os_dataset$class, alzheime_os_dataset$military_service)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
```

```
## data:  alzheime_os_dataset$class and alzheime_os_dataset$military_service
## X-squared = 2.1493, df = 1, p-value = 0.1426
```

```
alzheime_os_dataset %>%
  filter(class == "1") %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot::corrplot()
```



Feature the selection Alsheimer using Cramer's V

```
cramersV(alzheime_os_dataset$class, alzheime_os_dataset$gender)
```

```
## [1] 0.003011152
```

```
cramersV(alzheime_os_dataset$class, alzheime_os_dataset$age)
```

```
## Warning in chisq.test(...): Chi-squared approximation may be incorrect
```

```
## [1] 0.06100143
```

```
cramersV(alzheime_os_dataset$class, alzheime_os_dataset$education)
```

```
## [1] 0.0190584
```

```
cramersV(alzheime_os_dataset$class, alzheime_os_dataset$marital_status)
```

```
## [1] 0.05101965
```

```
cramersV(alzheime_os_dataset$class, alzheime_os_dataset$zipcode)

## [1] 0.1203814

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$employment_status)

## [1] 0.1061203

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$children)

## [1] 0.07332783

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$ancestry)

## [1] 0.07195677

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$avg_commute)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.9268049

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$daily_internet_use)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.6967101

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$available_vehicles)

## [1] 0.05195255

cramersV(alzheime_os_dataset$class, alzheime_os_dataset$military_service)

## [1] 0.02543615
```

Feature the selection Hypertension using Chi-squared

```
hypertension_set <- select(patients, gender, age, employment_status,
education, marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service, hypertension)
FeatureTrain <- sample(nrow(hypertension_set), 0.7*nrow(hypertension_set),
replace = FALSE)
FeatureTrainSet <- hypertension_set[FeatureTrain,]
FeatureValidSet <- hypertension_set[-FeatureTrain,]

response <- as.factor(patients$hypertension)
input <- select(patients, gender, age, employment_status, education,
marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service)

data <- ubOver(X=input, Y=response)
hypertension_os_dataset <- cbind(data$X, class=data$Y)
```

```r
chisq.test(hypertension_os_dataset$class, hypertension_os_dataset$gender)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  hypertension_os_dataset$class and hypertension_os_dataset$gender
## X-squared = 6.2623, df = 1, p-value = 0.01233
```

```r
chisq.test(hypertension_os_dataset$class, hypertension_os_dataset$age)
```

```
## Warning in chisq.test(hypertension_os_dataset$class,
## hypertension_os_dataset$age): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and hypertension_os_dataset$age
## X-squared = 35.943, df = 3, p-value = 7.698e-08
```

```r
chisq.test(hypertension_os_dataset$class, hypertension_os_dataset$education)
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and hypertension_os_dataset$education
## X-squared = 1.463, df = 3, p-value = 0.6908
```

```r
chisq.test(hypertension_os_dataset$class,
hypertension_os_dataset$marital_status)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  hypertension_os_dataset$class and
hypertension_os_dataset$marital_status
## X-squared = 1.2361, df = 1, p-value = 0.2662
```

```r
chisq.test(hypertension_os_dataset$class, hypertension_os_dataset$zipcode)
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and hypertension_os_dataset$zipcode
## X-squared = 45.135, df = 12, p-value = 9.771e-06
```

```r
chisq.test(hypertension_os_dataset$class,
hypertension_os_dataset$employment_status)
```

```
##
##  Pearson's Chi-squared test
##
```

```
## data:  hypertension_os_dataset$class and
hypertension_os_dataset$employment_status
## X-squared = 0.81971, df = 3, p-value = 0.8447
```

```
chisq.test(hypertension_os_dataset$class, hypertension_os_dataset$children)
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and hypertension_os_dataset$children
## X-squared = 36.927, df = 7, p-value = 4.842e-06
```

```
chisq.test(hypertension_os_dataset$class, hypertension_os_dataset$ancestry)
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and hypertension_os_dataset$ancestry
## X-squared = 2.9499, df = 3, p-value = 0.3994
```

```
chisq.test(hypertension_os_dataset$class,
hypertension_os_dataset$avg_commute)
```

```
## Warning in chisq.test(hypertension_os_dataset$class,
## hypertension_os_dataset$avg_commute): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and
hypertension_os_dataset$avg_commute
## X-squared = 2986.7, df = 1521, p-value < 2.2e-16
```

```
chisq.test(hypertension_os_dataset$class,
hypertension_os_dataset$daily_internet_use)
```

```
## Warning in chisq.test(hypertension_os_dataset$class,
## hypertension_os_dataset$daily_internet_use): Chi-squared approximation may
## be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and
hypertension_os_dataset$daily_internet_use
## X-squared = 1605.3, df = 573, p-value < 2.2e-16
```

```
chisq.test(hypertension_os_dataset$class,
hypertension_os_dataset$available_vehicles)
```

```
##
##  Pearson's Chi-squared test
##
## data:  hypertension_os_dataset$class and
hypertension_os_dataset$available_vehicles
## X-squared = 2.3449, df = 4, p-value = 0.6726
```

```
chisq.test(hypertension_os_dataset$class,
hypertension_os_dataset$military_service)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  hypertension_os_dataset$class and
hypertension_os_dataset$military_service
## X-squared = 0.031941, df = 1, p-value = 0.8582
```

Feature the selection Hypertension using Cramer's V

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$gender)
```

```
## [1] 0.04289146
```

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$age)
```

```
## Warning in chisq.test(...): Chi-squared approximation may be incorrect
```

```
## [1] 0.1027576
```

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$education)
```

```
## [1] 0.02073145
```

```
cramersV(hypertension_os_dataset$class,
hypertension_os_dataset$marital_status)
```

```
## [1] 0.01905634
```

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$zipcode)
```

```
## [1] 0.1151492
```

```
cramersV(hypertension_os_dataset$class,
hypertension_os_dataset$employment_status)
```

```
## [1] 0.015518
```

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$children)
```

```
## [1] 0.1041548
```

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$ancestry)
```

```
## [1] 0.02943802
```

```
cramersV(hypertension_os_dataset$class, hypertension_os_dataset$avg_commute)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.9367087

cramersV(hypertension_os_dataset$class,
hypertension_os_dataset$daily_internet_use)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.6867196

cramersV(hypertension_os_dataset$class,
hypertension_os_dataset$available_vehicles)

## [1] 0.02624609

cramersV(hypertension_os_dataset$class,
hypertension_os_dataset$military_service)

## [1] 0.003063235
```

Feature the selection Skin Cancer using Chi-squared

```
skin_cancer_set <- select(patients, gender, age, employment_status,
education, marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service, skin_cancer)
FeatureTrain <- sample(nrow(skin_cancer_set), 0.7*nrow(skin_cancer_set),
replace = FALSE)
FeatureTrainSet <- skin_cancer_set[FeatureTrain,]
FeatureValidSet <- skin_cancer_set[-FeatureTrain,]

response <- as.factor(patients$skin_cancer)
input <- select(patients, gender, age, employment_status, education,
marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service)

data <- ubOver(X=input, Y=response)
skin_cancer_os_dataset <- cbind(data$X, class=data$Y)

chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$gender)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$gender
## X-squared = 0.0045293, df = 1, p-value = 0.9463

chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$age)

## Warning in chisq.test(skin_cancer_os_dataset$class,
## skin_cancer_os_dataset$age): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$age
## X-squared = 21.309, df = 3, p-value = 9.082e-05
```

```
chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$education)
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$education
## X-squared = 1.0819, df = 3, p-value = 0.7814
```

```
chisq.test(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$marital_status)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  skin_cancer_os_dataset$class and
skin_cancer_os_dataset$marital_status
## X-squared = 1.6657, df = 1, p-value = 0.1968
```

```
chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$zipcode)
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$zipcode
## X-squared = 25.445, df = 12, p-value = 0.01285
```

```
chisq.test(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$employment_status)
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and
skin_cancer_os_dataset$employment_status
## X-squared = 41.343, df = 3, p-value = 5.53e-09
```

```
chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$children)
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$children
## X-squared = 45.33, df = 7, p-value = 1.18e-07
```

```
chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$ancestry)
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$ancestry
## X-squared = 3.4317, df = 3, p-value = 0.3297
```

```
chisq.test(skin_cancer_os_dataset$class, skin_cancer_os_dataset$avg_commute)
```

```
## Warning in chisq.test(skin_cancer_os_dataset$class,
## skin_cancer_os_dataset$avg_commute): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and skin_cancer_os_dataset$avg_commute
## X-squared = 3188, df = 1522, p-value < 2.2e-16
```

```
chisq.test(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$daily_internet_use)
```

```
## Warning in chisq.test(skin_cancer_os_dataset$class,
## skin_cancer_os_dataset$daily_internet_use): Chi-squared approximation may
## be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and
skin_cancer_os_dataset$daily_internet_use
## X-squared = 1911.4, df = 573, p-value < 2.2e-16
```

```
chisq.test(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$available_vehicles)
```

```
##
##  Pearson's Chi-squared test
##
## data:  skin_cancer_os_dataset$class and
skin_cancer_os_dataset$available_vehicles
## X-squared = 31.832, df = 4, p-value = 2.071e-06
```

```
chisq.test(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$military_service)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  skin_cancer_os_dataset$class and
skin_cancer_os_dataset$military_service
## X-squared = 0.5588, df = 1, p-value = 0.4547
```

Feature the selection Skin Cancer using Cramer's V

```
cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$gender)

## [1] 0.001132097

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$age)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.07765085

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$education)

## [1] 0.01749679

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$marital_status)

## [1] 0.02171011

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$zipcode)

## [1] 0.08485364

cramersV(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$employment_status)

## [1] 0.1081607

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$children)

## [1] 0.113256

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$ancestry)

## [1] 0.0311616

cramersV(skin_cancer_os_dataset$class, skin_cancer_os_dataset$avg_commute)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.9497831

cramersV(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$daily_internet_use)

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## [1] 0.7354225

cramersV(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$available_vehicles)

## [1] 0.09490698
```

```
cramersV(skin_cancer_os_dataset$class,
skin_cancer_os_dataset$military_service)

## [1] 0.01257463
```

Modeling

After having better understanding of the data, will begin preparing for modeling in order to predict diseases.

The process will be divided the process into steps: 1. Dealing with the Imbalance 2. Define algorithms 3. Testing algorithms

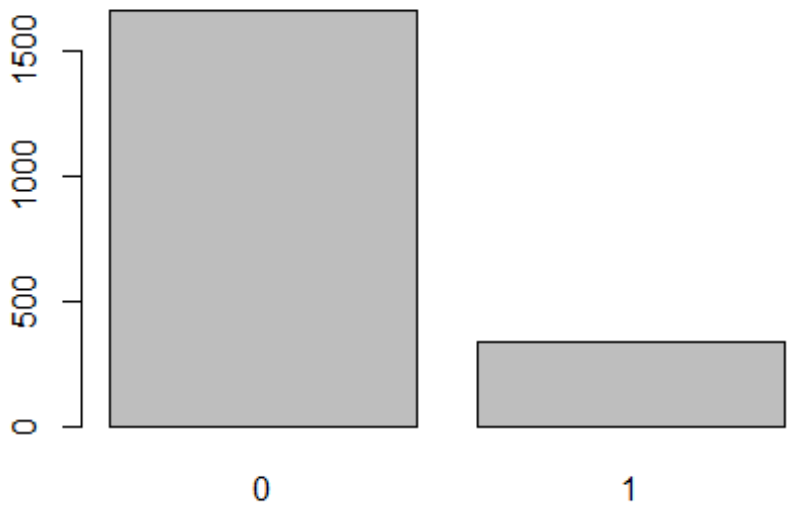
Dealing with the Imbalance

From the exploratory analysis abovem the dependent variable is imbalanced. There are many alternatives to tackle this problem: * Over-sampling * Under-sampling * Synthetic Minority Over-Sampling Technique (SMOTE) Sampling * Cost Sensitive Learning

For this dataset, will use over-sampling and SMOTE technique.

Patients with alzheimer

```
#Convert all columns to factor
patients[] <- lapply( patients, factor) # the "[]" keeps the dataframe
structure
 col_names <- names(patients)
 patients[col_names] <- lapply(patients[col_names], factor)

#See the data before balancing
barplot(table(patients$alzheimer), xlab=colnames(patients$alzheimer))
```




```
#filter the dataset and have only alzheimer disease as the target
alzheimer_set <- select(patients, gender, age, employment_status, education,
marital_status, ancestry, available_vehicles,zipcode,
children,military_service, alzheimer)

#The data were partitioned into a test and training set using a 70/30 split.
train <- sample(nrow(alzheimer_set), 0.7*nrow(alzheimer_set), replace =
FALSE)
  TrainSet <- alzheimer_set[train,]
  ValidSet <- alzheimer_set[-train,]

response <- as.factor(patients$alzheimer)
input <- select(patients, gender, age, employment_status, education,
marital_status, ancestry)
```

Excerice the Undersampling, oversampling, and smote against the test dataset

Logistic Regression, Randomforest, and Naive Bayes Models

```
  #initialize variables
  us_glm_accuracy <- c()
  us_glm_precision <- c()
  us_glm_recall <- c()
  us_glm_f1 <- c()

  os_glm_accuracy <- c()
  os_glm_precision <- c()
```

```r
  os_glm_recall <- c()
  os_glm_f1 <- c()

  smote_glm_accuracy <- c()
  smote_glm_precision <- c()
  smote_glm_recall <- c()
  smote_glm_f1 <- c()

  us_rf_accuracy <- c()
  us_rf_precision <- c()
  us_rf_recall <- c()
  us_rf_f1 <- c()

  os_rf_accuracy <- c()
  os_rf_precision <- c()
  os_rf_recall <- c()
  os_rf_f1 <- c()

  smote_rf_accuracy <- c()
  smote_rf_precision <- c()
  smote_rf_recall <- c()
  smote_rf_f1 <- c()

  us_nb_accuracy <- c()
  us_nb_precision <- c()
  us_nb_recall <- c()
  us_nb_f1 <- c()

  os_nb_accuracy <- c()
  os_nb_precision <- c()
  os_nb_recall <- c()
  os_nb_f1 <- c()

  smote_nb_accuracy <- c()
  smote_nb_precision <- c()
  smote_nb_recall <- c()
  smote_nb_f1 <- c()

  #use the 10-fold cross-validation and repeate the step 3 times
  train_control <- trainControl(method = "cv", number = 10)
  metric <- "Accuracy"
  mtry <- sqrt(ncol(alzheimer_set))
  tunegrid <- expand.grid(.mtry=mtry)

  # iterate throug the sampling and model 10 times and get the mean to get
the best model for the dataset prediction
  for (i in 1:10) {

    seed <- 999+i
```

```
    set.seed(seed)

    #run the undersampling
    data <- ubUnder(X=input, Y=response, perc=40, method="percPos")
    us_dataset <- cbind(data$X, class=data$Y)

    #run the oversampling
    data <- ubOver(X=input, Y=response)
    os_dataset <- cbind(data$X, class=data$Y)

    #run the smote
    data <- ubSMOTE(X=input, Y=response)
    smote_dataset <- cbind(data$X, class=data$Y)

    #run the logistic regression for the undersampling
    glm_mod <- caret::train(class~.,data=us_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    us_glm_accuracy <- c(us_glm_accuracy, us_cm$overall['Accuracy'])
    us_glm_precision <- c(us_glm_precision, us_cm$byClass['Precision'])
    us_glm_recall <- c(us_glm_recall, us_cm$byClass['Recall'])
    us_glm_f1 <- c(us_glm_f1, us_cm$byClass['F1'])

    #run the logistic regression for the oversampling
    glm_mod <- caret::train(class~.,data=os_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    os_glm_accuracy <- c(os_glm_accuracy, os_cm$overall['Accuracy'])
    os_glm_precision <- c(os_glm_precision, os_cm$byClass['Precision'])
    os_glm_recall <- c(os_glm_recall, os_cm$byClass['Recall'])
    os_glm_f1 <- c(os_glm_f1, os_cm$byClass['F1'])

    #run the logistic regression for the smote
    glm_mod <- caret::train(class~.,data=smote_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    smote_glm_accuracy <- c(smote_glm_accuracy, cm_smote$overall['Accuracy'])
    smote_glm_precision <- c(smote_glm_precision,
cm_smote$byClass['Precision'])
    smote_glm_recall <- c(smote_glm_recall, cm_smote$byClass['Recall'])
    smote_glm_f1 <- c(smote_glm_f1, cm_smote$byClass['F1'])
```

```
    #run the random forest for the undersampling
    rf_mod <- caret::train(class~., data=us_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    us_rf_accuracy <- c(us_rf_accuracy, us_cm$overall['Accuracy'])
    us_rf_precision <- c(us_rf_precision, us_cm$byClass['Precision'])
    us_rf_recall <- c(us_rf_recall, us_cm$byClass['Recall'])
    us_rf_f1 <- c(us_rf_f1, us_cm$byClass['F1'])

    #run the random forest for the oversampling
    rf_mod <- caret::train(class~., data=os_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    os_rf_accuracy <- c(os_rf_accuracy, os_cm$overall['Accuracy'])
    os_rf_precision <- c(os_rf_precision, os_cm$byClass['Precision'])
    os_rf_recall <- c(os_rf_recall, os_cm$byClass['Recall'])
    os_rf_f1 <- c(os_rf_f1, os_cm$byClass['F1'])

    #run the random forest for the smote
    rf_mod <- caret::train(class~., data=smote_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    smote_rf_accuracy <- c(smote_rf_accuracy, cm_smote$overall['Accuracy'])
    smote_rf_precision <- c(smote_rf_precision,
cm_smote$byClass['Precision'])
    smote_rf_recall <- c(smote_rf_recall, cm_smote$byClass['Recall'])
    smote_rf_f1 <- c(smote_rf_f1, cm_smote$byClass['F1'])

    #run the naive byes for the undersampling
    nb_mod <- caret::train(class~., data=us_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    us_nb_accuracy <- c(us_nb_accuracy, us_cm$overall['Accuracy'])
    us_nb_precision <- c(us_nb_precision, us_cm$byClass['Precision'])
    us_nb_recall <- c(us_nb_recall, us_cm$byClass['Recall'])
    us_nb_f1 <- c(us_nb_f1, us_cm$byClass['F1'])

    #run the naive byes for the oversampling
    nb_mod <- caret::train(class~., data=os_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
```

```
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    os_nb_accuracy <- c(os_nb_accuracy, os_cm$overall['Accuracy'])
    os_nb_precision <- c(os_nb_precision, os_cm$byClass['Precision'])
    os_nb_recall <- c(os_nb_recall, os_cm$byClass['Recall'])
    os_nb_f1 <- c(os_nb_f1, os_cm$byClass['F1'])

    #run the naive byes for the smote
    nb_mod <- caret::train(class~., data=smote_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$alzheimer),
mode='everything')
    smote_nb_accuracy <- c(smote_nb_accuracy, cm_smote$overall['Accuracy'])
    smote_nb_precision <- c(smote_nb_precision,
cm_smote$byClass['Precision'])
    smote_nb_recall <- c(smote_nb_recall, cm_smote$byClass['Recall'])
    smote_nb_f1 <- c(smote_nb_f1, cm_smote$byClass['F1'])
  }
```

Result of the alzheimer analysis

The data were partitioned into a test and training set using a 70/30 split.

```
df <- data.frame(us_glm_accuracy, os_glm_accuracy, smote_glm_accuracy,
us_rf_accuracy, os_rf_accuracy, smote_rf_accuracy, us_nb_accuracy,
os_nb_accuracy, smote_nb_accuracy)

us_glm_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7900000 0.8116667 0.7900000 0.7850000 0.7883333 0.7966667 0.7966667
##  Accuracy  Accuracy  Accuracy
## 0.7866667 0.7816667 0.7800000
```

```
os_glm_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.5550000 0.5500000 0.5483333 0.5266667 0.5200000 0.5450000 0.5350000
##  Accuracy  Accuracy  Accuracy
## 0.5366667 0.5450000 0.5316667
```

```
smote_glm_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.6983333 0.7033333 0.7283333 0.7100000 0.7250000 0.7016667 0.7133333
##  Accuracy  Accuracy  Accuracy
## 0.7000000 0.7250000 0.7016667
```

```
us_rf_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8116667 0.8183333 0.8150000 0.7966667 0.8300000 0.8266667 0.8033333
##  Accuracy  Accuracy  Accuracy
## 0.8016667 0.8150000 0.8116667
```

```
os_rf_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.5966667 0.5783333 0.5566667 0.5916667 0.5850000 0.5800000 0.6083333
##  Accuracy  Accuracy  Accuracy
## 0.5800000 0.5700000 0.5966667
```

```
smote_rf_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7650000 0.7683333 0.7833333 0.7483333 0.7766667 0.7433333 0.7750000
##  Accuracy  Accuracy  Accuracy
## 0.7500000 0.7683333 0.7666667
```

```
us_nb_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.6866667 0.8366667 0.8283333 0.8300000 0.7966667 0.8366667 0.6716667
##  Accuracy  Accuracy  Accuracy
## 0.8350000 0.8366667 0.7966667
```

```
os_nb_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.1866667 0.2233333 0.2266667 0.2350000 0.2783333 0.2150000 0.1983333
##  Accuracy  Accuracy  Accuracy
## 0.2383333 0.3133333 0.1983333
```

```
smote_nb_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7500000 0.7600000 0.7783333 0.7950000 0.7850000 0.6033333 0.7483333
##  Accuracy  Accuracy  Accuracy
## 0.6566667 0.8066667 0.7433333
```

```
us_glm_precision
```

```
## Precision Precision Precision Precision Precision Precision Precision
## 0.8443223 0.8406305 0.8405797 0.8409506 0.8415301 0.8429603 0.8442029
## Precision Precision Precision
## 0.8488806 0.8391225 0.8464419
```

```
os_glm_precision
```

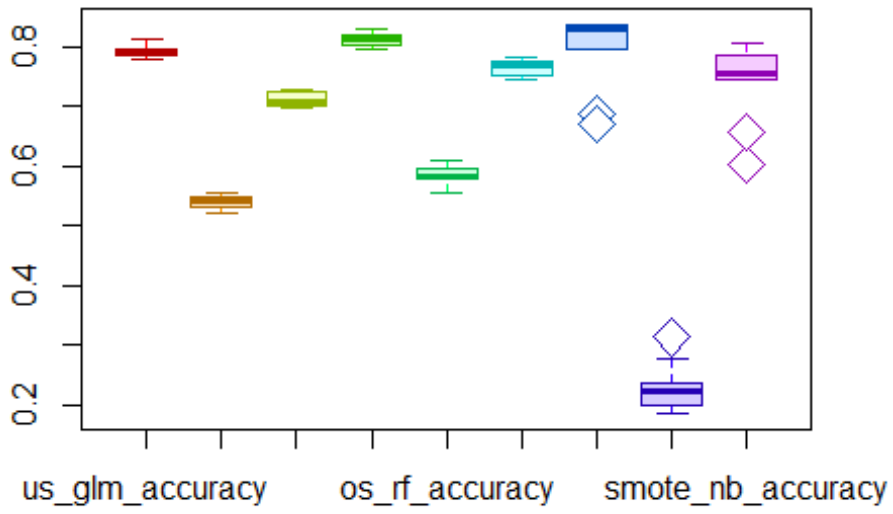
```
## Precision Precision Precision Precision Precision Precision Precision
## 0.8827362 0.8625000 0.8713826 0.8784722 0.8689655 0.8566978 0.8655738
```

```
## Precision Precision Precision
## 0.8758389 0.8459215 0.8646865

smote_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8350731 0.8333333 0.8410463 0.8374486 0.8363273 0.8414376 0.8353659
## Precision Precision Precision
## 0.8340249 0.8404040 0.8357588

us_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8394415 0.8441331 0.8447972 0.8467153 0.8436426 0.8442907 0.8529412
## Precision Precision Precision
## 0.8450450 0.8484848 0.8454707

os_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8939394 0.8952381 0.8907285 0.9028213 0.8940810 0.9058442 0.8985075
## Precision Precision Precision
## 0.8930818 0.8935484 0.9062500

smote_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8412098 0.8405253 0.8444444 0.8368522 0.8445693 0.8411765 0.8391867
## Precision Precision Precision
## 0.8397683 0.8392523 0.8376866

us_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8536036 0.8366667 0.8375635 0.8378378 0.8333333 0.8366667 0.8588235
## Precision Precision Precision
## 0.8375209 0.8366667 0.8442029

os_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
## 1.0000000 0.9285714 0.9318182 0.8909091 0.9259259 0.9696970 1.0000000
## Precision Precision Precision
## 0.8947368 0.8813559 1.0000000

smote_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8333333 0.8314815 0.8348457 0.8365897 0.8348294 0.8419689 0.8342857
## Precision Precision Precision
## 0.8303571 0.8327586 0.8307985

us_glm_recall
```

```
##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.9183267 0.9561753 0.9243028 0.9163347 0.9203187 0.9302789 0.9282869
##    Recall    Recall    Recall
## 0.9063745 0.9143426 0.9003984

os_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.5398406 0.5498008 0.5398406 0.5039841 0.5019920 0.5478088 0.5258964
##    Recall    Recall    Recall
## 0.5199203 0.5577689 0.5219124

smote_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.7968127 0.8067729 0.8326693 0.8107570 0.8346614 0.7928287 0.8187251
##    Recall    Recall    Recall
## 0.8007968 0.8286853 0.8007968

us_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.9581673 0.9601594 0.9541833 0.9243028 0.9780876 0.9721116 0.9243028
##    Recall    Recall    Recall
## 0.9342629 0.9482072 0.9482072

os_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.5876494 0.5617530 0.5358566 0.5737052 0.5717131 0.5557769 0.5996016
##    Recall    Recall    Recall
## 0.5657371 0.5517928 0.5776892

smote_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8864542 0.8924303 0.9083665 0.8685259 0.8984064 0.8545817 0.9043825
##    Recall    Recall    Recall
## 0.8665339 0.8944223 0.8944223

us_nb_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.7549801 1.0000000 0.9860558 0.9880478 0.9462151 1.0000000 0.7270916
##    Recall    Recall    Recall
## 0.9960159 1.0000000 0.9282869

os_nb_recall

##     Recall     Recall     Recall     Recall     Recall     Recall
## 0.02788845 0.07768924 0.08167331 0.09760956 0.14940239 0.06374502
##     Recall     Recall     Recall     Recall
## 0.04183267 0.10159363 0.20717131 0.04183267
```

```
smote_nb_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8764940 0.8944223 0.9163347 0.9382470 0.9262948 0.6474104 0.8725100
##    Recall    Recall    Recall
## 0.7410359 0.9621514 0.8705179

us_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8797710 0.8946878 0.8804554 0.8770257 0.8791627 0.8844697 0.8842505
##        F1        F1        F1
## 0.8766859 0.8751192 0.8725869

os_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.6699629 0.6715328 0.6666667 0.6405063 0.6363636 0.6682868 0.6542751
##        F1        F1        F1
## 0.6525000 0.6722689 0.6509317

smote_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8154944 0.8198381 0.8368368 0.8238866 0.8354935 0.8164103 0.8269618
##        F1        F1        F1
## 0.8170732 0.8345035 0.8179044

us_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8948837 0.8984157 0.8961646 0.8838095 0.9059041 0.9037037 0.8871893
##        F1        F1        F1
## 0.8874172 0.8955786 0.8938967

os_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.7091346 0.6903305 0.6691542 0.7015834 0.6974484 0.6888889 0.7192354
##        F1        F1        F1
## 0.6926829 0.6822660 0.7055961

smote_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8632396 0.8657005 0.8752399 0.8523949 0.8706564 0.8478261 0.8705657
##        F1        F1        F1
## 0.8529412 0.8659595 0.8651252

us_nb_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8012685 0.9110708 0.9057640 0.9067642 0.8861940 0.9110708 0.7874865
```

```
##        F1        F1        F1
## 0.9099181 0.9110708 0.8842505

os_nb_f1

##         F1         F1         F1         F1         F1         F1
## 0.05426357 0.14338235 0.15018315 0.17594255 0.25728988 0.11962617
##         F1         F1         F1         F1
## 0.08030593 0.18246869 0.33548387 0.08030593

smote_nb_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8543689 0.8618042 0.8736942 0.8845070 0.8781870 0.7319820 0.8529698
##        F1        F1        F1
## 0.7831579 0.8927911 0.8501946

c1 <- rainbow(10)
c2 <- rainbow(10, alpha=0.2)
c3 <- rainbow(10, v=0.7)
boxplot(df, col=c2, medcol=c3, whiskcol=c1, staplecol=c3, boxcol=c3,
outcol=c3, pch=23, cex=2)
```




```
mean(us_nb_accuracy)

## [1] 0.7955

mean(us_nb_precision)
```

```
## [1] 0.8412886
```

```
mean(us_nb_recall)
```

```
## [1] 0.9326693
```

```
mean(us_nb_f1)
```

```
## [1] 0.8814858
```

```
mean(os_nb_accuracy)
```

```
## [1] 0.2313333
```

```
mean(os_nb_precision)
```

```
## [1] 0.9423014
```

```
mean(os_nb_recall)
```

```
## [1] 0.08904382
```

```
mean(os_nb_f1)
```

```
## [1] 0.1579252
```

```
mean(smote_nb_accuracy)
```

```
## [1] 0.7426667
```

```
mean(smote_nb_precision)
```

```
## [1] 0.8341249
```

```
mean(smote_nb_recall)
```

```
## [1] 0.8645418
```

```
mean(smote_nb_f1)
```

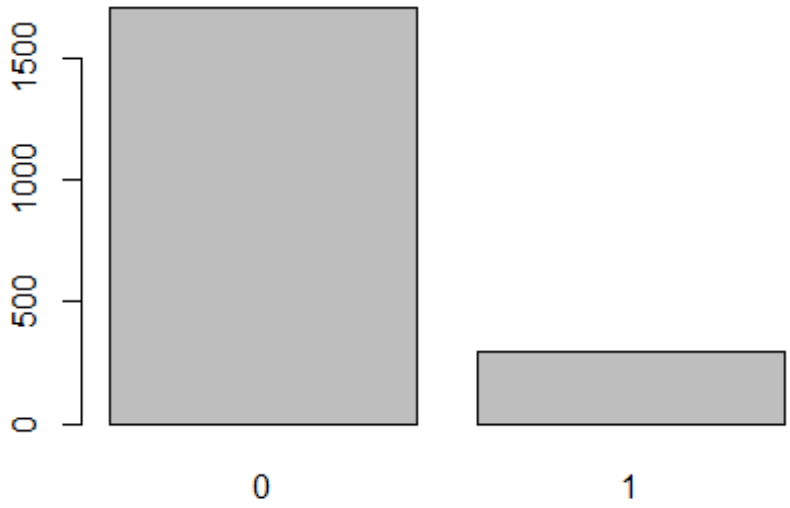
```
## [1] 0.8463657
```

```
a <- matrix(

c(mean(us_glm_accuracy),mean(us_glm_precision),mean(us_glm_recall),mean(us_gl
m_f1),

mean(os_glm_accuracy),mean(os_glm_precision),mean(os_glm_recall),mean(os_glm_
f1),

mean(smote_glm_accuracy),mean(smote_glm_precision),mean(smote_glm_recall),mea
n(smote_glm_f1)),
  nrow=3,
  ncol=4,
  byrow = TRUE
```

```
)

a

##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.7906667 0.8429621 0.9215139 0.8804215
## [2,] 0.5393333 0.8672775 0.5308765 0.6583295
## [3,] 0.7106667 0.8370220 0.8123506 0.8244403
```

Patients with hypertension

```
#See the data before balancing
barplot(table(patients$hypertension), xlab=colnames(patients$hypertension))
```




```
#filter the dataset and have only hypertension disease as the target
hypertension_set <- select(patients, gender, age, employment_status,
education, marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service, hypertension)

#The data were partitioned into a test and training set using a 70/30 split.
train <- sample(nrow(hypertension_set), 0.7*nrow(hypertension_set), replace =
FALSE)
  TrainSet <- hypertension_set[train,]
  ValidSet <- hypertension_set[-train,]

response <- as.factor(patients$hypertension)
```

```
input <- select(patients, gender, age, employment_status, education,
marital_status, ancestry)
```

Excerice the Undersampling, oversampling, and smote against the test dataset

Logistic Regression, Randomforest, and Naive Bayes Models

```
  #initialize variables
  us_glm_accuracy <- c()
  us_glm_precision <- c()
  us_glm_recall <- c()
  us_glm_f1 <- c()

  os_glm_accuracy <- c()
  os_glm_precision <- c()
  os_glm_recall <- c()
  os_glm_f1 <- c()

  smote_glm_accuracy <- c()
  smote_glm_precision <- c()
  smote_glm_recall <- c()
  smote_glm_f1 <- c()

  us_rf_accuracy <- c()
  us_rf_precision <- c()
  us_rf_recall <- c()
  us_rf_f1 <- c()

  os_rf_accuracy <- c()
  os_rf_precision <- c()
  os_rf_recall <- c()
  os_rf_f1 <- c()

  smote_rf_accuracy <- c()
  smote_rf_precision <- c()
  smote_rf_recall <- c()
  smote_rf_f1 <- c()

  us_nb_accuracy <- c()
  us_nb_precision <- c()
  us_nb_recall <- c()
  us_nb_f1 <- c()

  os_nb_accuracy <- c()
  os_nb_precision <- c()
  os_nb_recall <- c()
  os_nb_f1 <- c()

  smote_nb_accuracy <- c()
  smote_nb_precision <- c()
```

```r
  smote_nb_recall <- c()
  smote_nb_f1 <- c()

  #use the 10-fold cross-validation and repeate the step 3 times
  train_control <- trainControl(method = "cv", number = 10)
  metric <- "Accuracy"
  mtry <- sqrt(ncol(alzheimer_set))
  tunegrid <- expand.grid(.mtry=mtry)

  # iterate throug the sampling and model 10 times and get the mean to get
the best model for the dataset prediction
  for (i in 1:10) {

    #run the undersampling
    data <- ubUnder(X=input, Y=response, perc=40, method="percPos")
    us_dataset <- cbind(data$X, class=data$Y)

    #run the oversampling
    data <- ubOver(X=input, Y=response)
    os_dataset <- cbind(data$X, class=data$Y)

    #run the smote
    data <- ubSMOTE(X=input, Y=response)
    smote_dataset <- cbind(data$X, class=data$Y)

    #use the 10-fold cross-validation and repeate the step 3 times
    train_control <- trainControl(method = "repeatedcv", number = 10,
repeats=3, savePredictions = TRUE)

    #run the logistic regression for the undersampling
    glm_mod <- caret::train(class~.,data=us_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    us_glm_accuracy <- c(us_glm_accuracy, us_cm$overall['Accuracy'])
    us_glm_precision <- c(us_glm_precision, us_cm$byClass['Precision'])
    us_glm_recall <- c(us_glm_recall, us_cm$byClass['Recall'])
    us_glm_f1 <- c(us_glm_f1, us_cm$byClass['F1'])

    #run the logistic regression for the oversampling
    glm_mod <- caret::train(class~.,data=os_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    os_glm_accuracy <- c(os_glm_accuracy, os_cm$overall['Accuracy'])
    os_glm_precision <- c(os_glm_precision, os_cm$byClass['Precision'])
    os_glm_recall <- c(os_glm_recall, os_cm$byClass['Recall'])
```

```
    os_glm_f1 <- c(os_glm_f1, os_cm$byClass['F1'])

    #run the logistic regression for the smote
    glm_mod <- caret::train(class~.,data=smote_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    smote_glm_accuracy <- c(smote_glm_accuracy, cm_smote$overall['Accuracy'])
    smote_glm_precision <- c(smote_glm_precision,
cm_smote$byClass['Precision'])
    smote_glm_recall <- c(smote_glm_recall, cm_smote$byClass['Recall'])
    smote_glm_f1 <- c(smote_glm_f1, cm_smote$byClass['F1'])

    #run the random forest for the undersampling
    rf_mod <- caret::train(class~., data=us_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    us_rf_accuracy <- c(us_rf_accuracy, us_cm$overall['Accuracy'])
    us_rf_precision <- c(us_rf_precision, us_cm$byClass['Precision'])
    us_rf_recall <- c(us_rf_recall, us_cm$byClass['Recall'])
    us_rf_f1 <- c(us_rf_f1, us_cm$byClass['F1'])

    #run the random forest for the oversampling
    rf_mod <- caret::train(class~., data=os_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    os_rf_accuracy <- c(os_rf_accuracy, os_cm$overall['Accuracy'])
    os_rf_precision <- c(os_rf_precision, os_cm$byClass['Precision'])
    os_rf_recall <- c(os_rf_recall, os_cm$byClass['Recall'])
    os_rf_f1 <- c(os_rf_f1, os_cm$byClass['F1'])

    #run the random forest for the smote
    rf_mod <- caret::train(class~., data=smote_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    smote_rf_accuracy <- c(smote_rf_accuracy, cm_smote$overall['Accuracy'])
    smote_rf_precision <- c(smote_rf_precision,
cm_smote$byClass['Precision'])
    smote_rf_recall <- c(smote_rf_recall, cm_smote$byClass['Recall'])
    smote_rf_f1 <- c(smote_rf_f1, cm_smote$byClass['F1'])

    #run the naive byes for the undersampling
```

```
    nb_mod <- caret::train(class~., data=us_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    us_nb_accuracy <- c(us_nb_accuracy, us_cm$overall['Accuracy'])
    us_nb_precision <- c(us_nb_precision, us_cm$byClass['Precision'])
    us_nb_recall <- c(us_nb_recall, us_cm$byClass['Recall'])
    us_nb_f1 <- c(us_nb_f1, us_cm$byClass['F1'])

    #run the naive byes for the oversampling
    nb_mod <- caret::train(class~., data=os_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    os_nb_accuracy <- c(os_nb_accuracy, os_cm$overall['Accuracy'])
    os_nb_precision <- c(os_nb_precision, os_cm$byClass['Precision'])
    os_nb_recall <- c(os_nb_recall, os_cm$byClass['Recall'])
    os_nb_f1 <- c(os_nb_f1, os_cm$byClass['F1'])

    #run the naive byes for the smote
    nb_mod <- caret::train(class~., data=smote_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$hypertension),
mode='everything')
    smote_nb_accuracy <- c(smote_nb_accuracy, cm_smote$overall['Accuracy'])
    smote_nb_precision <- c(smote_nb_precision,
cm_smote$byClass['Precision'])
    smote_nb_recall <- c(smote_nb_recall, cm_smote$byClass['Recall'])
    smote_nb_f1 <- c(smote_nb_f1, cm_smote$byClass['F1'])
  }
```

Result of the hypertension analysis

The data were partitioned into a test and training set using a 70/30 split.

```
df <- data.frame(us_glm_accuracy, os_glm_accuracy, smote_glm_accuracy,
us_rf_accuracy, os_rf_accuracy, smote_rf_accuracy, us_nb_accuracy,
os_nb_accuracy, smote_nb_accuracy)

us_glm_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8666667 0.8300000 0.8683333 0.8200000 0.8283333 0.8700000 0.8600000
##  Accuracy  Accuracy  Accuracy
## 0.8700000 0.8300000 0.8016667
```

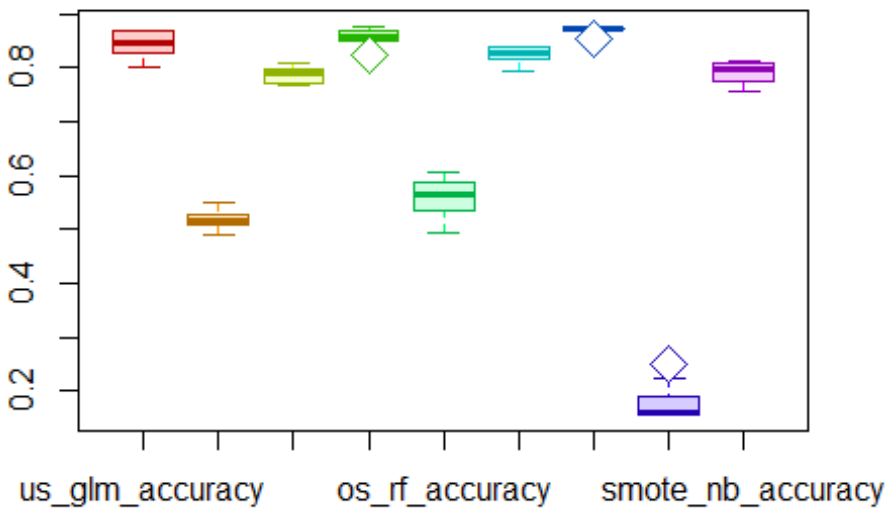
```
os_glm_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.4950000 0.5283333 0.5283333 0.5200000 0.5116667 0.5183333 0.5483333
##  Accuracy  Accuracy  Accuracy
## 0.5166667 0.5083333 0.4900000

smote_glm_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7933333 0.7716667 0.7683333 0.7700000 0.7916667 0.8100000 0.7850000
##  Accuracy  Accuracy  Accuracy
## 0.7850000 0.7983333 0.8083333

us_rf_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8566667 0.8233333 0.8666667 0.8516667 0.8583333 0.8683333 0.8750000
##  Accuracy  Accuracy  Accuracy
## 0.8766667 0.8483333 0.8600000

os_rf_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.5216667 0.4933333 0.5883333 0.5650000 0.5866667 0.5866667 0.6066667
##  Accuracy  Accuracy  Accuracy
## 0.5583333 0.5633333 0.5366667

smote_rf_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8250000 0.8250000 0.7950000 0.8400000 0.8350000 0.8400000 0.8066667
##  Accuracy  Accuracy  Accuracy
## 0.8316667 0.8150000 0.8383333

us_nb_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8716667 0.8550000 0.8716667 0.8716667 0.8716667 0.8716667 0.8716667
##  Accuracy  Accuracy  Accuracy
## 0.8716667 0.8716667 0.8716667

os_nb_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.1566667 0.1566667 0.1566667 0.1683333 0.1583333 0.2483333 0.1583333
##  Accuracy  Accuracy  Accuracy
## 0.1916667 0.2250000 0.1566667

smote_nb_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8116667 0.7833333 0.7566667 0.7733333 0.8016667 0.8016667 0.7966667
```

```
##  Accuracy  Accuracy  Accuracy
## 0.8100000 0.7750000 0.8116667

us_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8735245 0.8779174 0.8712375 0.8779599 0.8763441 0.8739496 0.8884956
## Precision Precision Precision
## 0.8739496 0.8752228 0.8754647

os_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8956835 0.8973510 0.9054054 0.8929766 0.8859060 0.8774194 0.8962264
## Precision Precision Precision
## 0.8949153 0.8958333 0.8945455

smote_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8858801 0.8906883 0.8855422 0.8857715 0.8871595 0.8865784 0.8877953
## Precision Precision Precision
## 0.8862745 0.8910506 0.8893130

us_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8894831 0.8897196 0.8892794 0.8931159 0.8996350 0.8867596 0.8985765
## Precision Precision Precision
## 0.8824532 0.9044944 0.8969259

os_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.9436090 0.9294118 0.9339623 0.9366667 0.9421222 0.9283489 0.9335347
## Precision Precision Precision
## 0.9271523 0.9364548 0.9328622

smote_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8899254 0.8943396 0.8891051 0.8917431 0.8925926 0.8903108 0.8921002
## Precision Precision Precision
## 0.8892989 0.8916350 0.8915441

us_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8716667 0.8758621 0.8716667 0.8729097 0.8716667 0.8716667 0.8716667
## Precision Precision Precision
## 0.8716667 0.8716667 0.8716667

os_nb_precision
```

```
## Precision Precision Precision Precision Precision Precision Precision
## 0.9047619 0.9047619 0.9047619 0.8750000 0.8461538 0.8829787 0.9090909
## Precision Precision Precision
## 0.9130435 0.8918919 0.9047619

smote_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8867925 0.8891089 0.8854806 0.8862275 0.8854962 0.8869732 0.8893204
## Precision Precision Precision
## 0.8880455 0.8864542 0.8867925

us_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.9904398 0.9349904 0.9961759 0.9216061 0.9349904 0.9942639 0.9598470
##    Recall    Recall    Recall
## 0.9942639 0.9388145 0.9005736

os_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.4760994 0.5181644 0.5124283 0.5105163 0.5047801 0.5200765 0.5449331
##    Recall    Recall    Recall
## 0.5047801 0.4933078 0.4703633

smote_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8757170 0.8413002 0.8432122 0.8451243 0.8718929 0.8967495 0.8623327
##    Recall    Recall    Recall
## 0.8642447 0.8757170 0.8910134

us_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.9541109 0.9101338 0.9674952 0.9426386 0.9426386 0.9732314 0.9655832
##    Recall    Recall    Recall
## 0.9904398 0.9235182 0.9483748

os_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.4799235 0.4531549 0.5678776 0.5372849 0.5602294 0.5697897 0.5908222
##    Recall    Recall    Recall
## 0.5353728 0.5353728 0.5047801

smote_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.9120459 0.9063098 0.8738050 0.9292543 0.9216061 0.9311663 0.8852772
##    Recall    Recall    Recall
## 0.9216061 0.8967495 0.9273423
```

```
us_nb_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 1.0000000 0.9713193 1.0000000 0.9980880 1.0000000 1.0000000 1.0000000
##    Recall    Recall    Recall
## 1.0000000 1.0000000 1.0000000

os_nb_recall

##     Recall     Recall     Recall     Recall     Recall     Recall
## 0.03632887 0.03632887 0.03632887 0.05353728 0.04206501 0.15869981
##     Recall     Recall     Recall     Recall
## 0.03824092 0.08030593 0.12619503 0.03632887

smote_nb_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8986616 0.8585086 0.8279159 0.8489484 0.8871893 0.8852772 0.8757170
##    Recall    Recall    Recall
## 0.8948375 0.8508604 0.8986616

us_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.9283154 0.9055556 0.9295272 0.8992537 0.9047179 0.9302326 0.9227941
##        F1        F1        F1
## 0.9302326 0.9059041 0.8878417

os_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.6217228 0.6569697 0.6544567 0.6496350 0.6431181 0.6530612 0.6777646
##        F1        F1        F1
## 0.6454768 0.6362515 0.6165414

smote_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8807692 0.8652901 0.8638590 0.8649706 0.8794600 0.8916350 0.8748788
##        F1        F1        F1
## 0.8751210 0.8833173 0.8901624

us_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.9206642 0.8998110 0.9267399 0.9172093 0.9206349 0.9279854 0.9308756
##        F1        F1        F1
## 0.9333333 0.9139073 0.9219331

os_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.6362484 0.6092545 0.7063020 0.6828676 0.7026379 0.7061611 0.7236534
```

```
##        F1        F1        F1
## 0.6787879 0.6812652 0.6550868

smote_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.9008499 0.9002849 0.8813886 0.9101124 0.9068674 0.9102804 0.8886756
##        F1        F1        F1
## 0.9051643 0.8941849 0.9090909

us_nb_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.9314337 0.9211242 0.9314337 0.9313113 0.9314337 0.9314337 0.9314337
##        F1        F1        F1
## 0.9314337 0.9314337 0.9314337

os_nb_f1

##         F1         F1         F1         F1         F1         F1
## 0.06985294 0.06985294 0.06985294 0.10090090 0.08014572 0.26904376
##         F1         F1         F1         F1
## 0.07339450 0.14762742 0.22110553 0.06985294

smote_nb_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8926876 0.8735409 0.8557312 0.8671875 0.8863419 0.8861244 0.8824663
##        F1        F1        F1
## 0.8914286 0.8682927 0.8926876

c1 <- rainbow(10)
c2 <- rainbow(10, alpha=0.2)
c3 <- rainbow(10, v=0.7)
boxplot(df, col=c2, medcol=c3, whiskcol=c1, staplecol=c3, boxcol=c3,
outcol=c3, pch=23, cex=2)
```




```
mean(us_nb_accuracy)
## [1] 0.87
mean(us_nb_precision)
## [1] 0.8722105
mean(us_nb_recall)
## [1] 0.9969407
mean(us_nb_f1)
## [1] 0.9303905
mean(os_nb_accuracy)
## [1] 0.1776667
mean(os_nb_precision)
## [1] 0.8937206
mean(os_nb_recall)
## [1] 0.06443595
mean(os_nb_f1)
```

```
## [1] 0.117163
```

```
mean(smote_nb_accuracy)
```

```
## [1] 0.7921667
```

```
mean(smote_nb_precision)
```

```
## [1] 0.8870691
```

```
mean(smote_nb_recall)
```

```
## [1] 0.8726577
```

```
mean(smote_nb_f1)
```

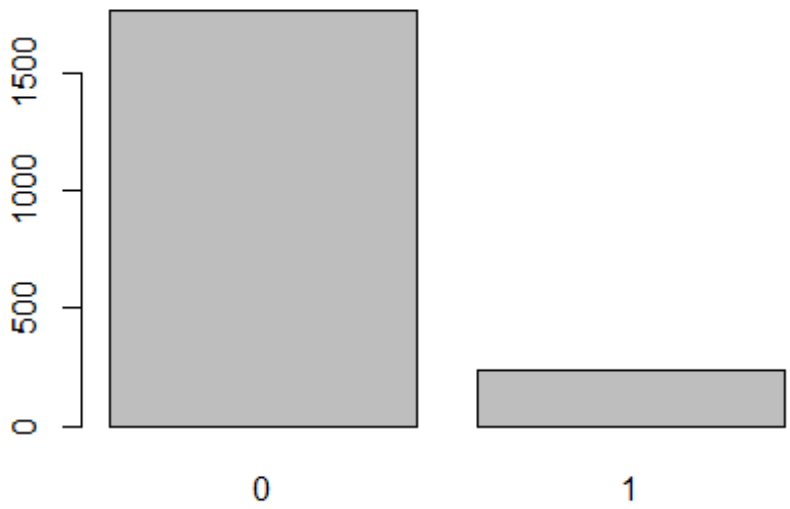
```
## [1] 0.8796489
```

```
a <- matrix(

c(mean(us_glm_accuracy),mean(us_glm_precision),mean(us_glm_recall),mean(us_gl
m_f1),

mean(os_glm_accuracy),mean(os_glm_precision),mean(os_glm_recall),mean(os_glm_
f1),

mean(smote_glm_accuracy),mean(smote_glm_precision),mean(smote_glm_recall),mea
n(smote_glm_f1)),
  nrow=3,
  ncol=4,
  byrow = TRUE
)

a
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.8445000 0.8764066 0.9565966 0.9144375
## [2,] 0.5165000 0.8936262 0.5055449 0.6454998
## [3,] 0.7881667 0.8876053 0.8667304 0.8769463
```

Patients with skin cancer

```
#See the data before balancing
barplot(table(patients$skin_cancer), xlab=colnames(patients$skin_cancer))
```




```
#filter the dataset and have only skin_cancer disease as the target
skin_cancer_set <- select(patients, gender, age, employment_status,
education, marital_status, ancestry, available_vehicles, avg_commute,zipcode,
children,daily_internet_use,military_service, skin_cancer)

#The data were partitioned into a test and training set using a 70/30 split.
train <- sample(nrow(skin_cancer_set), 0.7*nrow(skin_cancer_set), replace =
FALSE)
  TrainSet <- skin_cancer_set[train,]
  ValidSet <- skin_cancer_set[-train,]

response <- as.factor(patients$skin_cancer)
input <- select(patients, gender, age, employment_status, education,
marital_status, ancestry)
```

Excerice the Undersampling, oversampling, and smote against the test dataset

Logistic Regression, Randomforest, and Naive Bayes Models

```
  #initialize variables
  us_glm_accuracy <- c()
  us_glm_precision <- c()
  us_glm_recall <- c()
  us_glm_f1 <- c()

  os_glm_accuracy <- c()
  os_glm_precision <- c()
```

```r
  os_glm_recall <- c()
  os_glm_f1 <- c()

  smote_glm_accuracy <- c()
  smote_glm_precision <- c()
  smote_glm_recall <- c()
  smote_glm_f1 <- c()

  us_rf_accuracy <- c()
  us_rf_precision <- c()
  us_rf_recall <- c()
  us_rf_f1 <- c()

  os_rf_accuracy <- c()
  os_rf_precision <- c()
  os_rf_recall <- c()
  os_rf_f1 <- c()

  smote_rf_accuracy <- c()
  smote_rf_precision <- c()
  smote_rf_recall <- c()
  smote_rf_f1 <- c()

  us_nb_accuracy <- c()
  us_nb_precision <- c()
  us_nb_recall <- c()
  us_nb_f1 <- c()

  os_nb_accuracy <- c()
  os_nb_precision <- c()
  os_nb_recall <- c()
  os_nb_f1 <- c()

  smote_nb_accuracy <- c()
  smote_nb_precision <- c()
  smote_nb_recall <- c()
  smote_nb_f1 <- c()

  #use the 10-fold cross-validation and repeate the step 3 times
  train_control <- trainControl(method = "cv", number = 10)
  metric <- "Accuracy"
  mtry <- sqrt(ncol(skin_cancer_set))
  tunegrid <- expand.grid(.mtry=mtry)

  # iterate throug the sampling and model 10 times and get the mean to get
the best model for the dataset prediction
  for (i in 1:10) {

    #run the undersampling
```

```
    data <- ubUnder(X=input, Y=response, perc=40, method="percPos")
    us_dataset <- cbind(data$X, class=data$Y)

    #run the oversampling
    data <- ubOver(X=input, Y=response)
    os_dataset <- cbind(data$X, class=data$Y)

    #run the smote
    data <- ubSMOTE(X=input, Y=response)
    smote_dataset <- cbind(data$X, class=data$Y)

    #use the 10-fold cross-validation and repeate the step 3 times
    train_control <- trainControl(method = "repeatedcv", number = 10,
repeats=3, savePredictions = TRUE)

    #run the logistic regression for the undersampling
    glm_mod <- caret::train(class~.,data=us_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    us_glm_accuracy <- c(us_glm_accuracy, us_cm$overall['Accuracy'])
    us_glm_precision <- c(us_glm_precision, us_cm$byClass['Precision'])
    us_glm_recall <- c(us_glm_recall, us_cm$byClass['Recall'])
    us_glm_f1 <- c(us_glm_f1, us_cm$byClass['F1'])

    #run the logistic regression for the oversampling
    glm_mod <- caret::train(class~.,data=os_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    os_glm_accuracy <- c(os_glm_accuracy, os_cm$overall['Accuracy'])
    os_glm_precision <- c(os_glm_precision, os_cm$byClass['Precision'])
    os_glm_recall <- c(os_glm_recall, os_cm$byClass['Recall'])
    os_glm_f1 <- c(os_glm_f1, os_cm$byClass['F1'])

    #run the logistic regression for the smote
    glm_mod <- caret::train(class~.,data=smote_dataset, trControl =
train_control, method="glm", family="binomial", tuneLength = 5)
    pred = predict(glm_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    smote_glm_accuracy <- c(smote_glm_accuracy, cm_smote$overall['Accuracy'])
    smote_glm_precision <- c(smote_glm_precision,
cm_smote$byClass['Precision'])
    smote_glm_recall <- c(smote_glm_recall, cm_smote$byClass['Recall'])
    smote_glm_f1 <- c(smote_glm_f1, cm_smote$byClass['F1'])
```

```
    #run the random forest for the undersampling
    rf_mod <- caret::train(class~., data=us_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    us_rf_accuracy <- c(us_rf_accuracy, us_cm$overall['Accuracy'])
    us_rf_precision <- c(us_rf_precision, us_cm$byClass['Precision'])
    us_rf_recall <- c(us_rf_recall, us_cm$byClass['Recall'])
    us_rf_f1 <- c(us_rf_f1, us_cm$byClass['F1'])

    #run the random forest for the oversampling
    rf_mod <- caret::train(class~., data=os_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    os_rf_accuracy <- c(os_rf_accuracy, os_cm$overall['Accuracy'])
    os_rf_precision <- c(os_rf_precision, os_cm$byClass['Precision'])
    os_rf_recall <- c(os_rf_recall, os_cm$byClass['Recall'])
    os_rf_f1 <- c(os_rf_f1, os_cm$byClass['F1'])

    #run the random forest for the smote
    rf_mod <- caret::train(class~., data=smote_dataset, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=train_control)
    pred = predict(rf_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    smote_rf_accuracy <- c(smote_rf_accuracy, cm_smote$overall['Accuracy'])
    smote_rf_precision <- c(smote_rf_precision,
cm_smote$byClass['Precision'])
    smote_rf_recall <- c(smote_rf_recall, cm_smote$byClass['Recall'])
    smote_rf_f1 <- c(smote_rf_f1, cm_smote$byClass['F1'])

    #run the naive byes for the undersampling
    nb_mod <- caret::train(class~., data=us_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    us_cm <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    us_nb_accuracy <- c(us_nb_accuracy, us_cm$overall['Accuracy'])
    us_nb_precision <- c(us_nb_precision, us_cm$byClass['Precision'])
    us_nb_recall <- c(us_nb_recall, us_cm$byClass['Recall'])
    us_nb_f1 <- c(us_nb_f1, us_cm$byClass['F1'])

    #run the naive byes for the oversampling
    nb_mod <- caret::train(class~., data=os_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
```

```
    os_cm <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    os_nb_accuracy <- c(os_nb_accuracy, os_cm$overall['Accuracy'])
    os_nb_precision <- c(os_nb_precision, os_cm$byClass['Precision'])
    os_nb_recall <- c(os_nb_recall, os_cm$byClass['Recall'])
    os_nb_f1 <- c(os_nb_f1, os_cm$byClass['F1'])

    #run the naive byes for the smote
    nb_mod <- caret::train(class~., data=smote_dataset, method="nb",
trControl=train_control)
    pred = predict(nb_mod, newdata=ValidSet)
    cm_smote <- confusionMatrix(data=pred, as.factor(ValidSet$skin_cancer),
mode='everything')
    smote_nb_accuracy <- c(smote_nb_accuracy, cm_smote$overall['Accuracy'])
    smote_nb_precision <- c(smote_nb_precision,
cm_smote$byClass['Precision'])
    smote_nb_recall <- c(smote_nb_recall, cm_smote$byClass['Recall'])
    smote_nb_f1 <- c(smote_nb_f1, cm_smote$byClass['F1'])
  }
```

Result of the skin cancer analysis

The data were partitioned into a test and training set using a 70/30 split.

```
df <- data.frame(us_glm_accuracy, os_glm_accuracy, smote_glm_accuracy,
us_rf_accuracy, os_rf_accuracy, smote_rf_accuracy, us_nb_accuracy,
os_nb_accuracy, smote_nb_accuracy)

us_glm_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8583333 0.8633333 0.7966667 0.8566667 0.8416667 0.8550000 0.7816667
##  Accuracy  Accuracy  Accuracy
## 0.7866667 0.8000000 0.8583333

os_glm_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.4583333 0.5216667 0.5066667 0.4950000 0.5166667 0.5700000 0.5550000
##  Accuracy  Accuracy  Accuracy
## 0.5300000 0.5383333 0.5150000

smote_glm_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7600000 0.7483333 0.7266667 0.8016667 0.7000000 0.7566667 0.7316667
##  Accuracy  Accuracy  Accuracy
## 0.7266667 0.7750000 0.7833333

us_rf_accuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7933333 0.7916667 0.8316667 0.7883333 0.7833333 0.7950000 0.7266667
##  Accuracy  Accuracy  Accuracy
## 0.7800000 0.7816667 0.7833333

os_rf_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.6200000 0.6166667 0.5983333 0.6150000 0.6033333 0.6583333 0.5933333
##  Accuracy  Accuracy  Accuracy
## 0.5716667 0.6116667 0.6133333

smote_rf_accuracy

##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.7583333 0.7600000 0.7650000 0.8150000 0.7583333 0.7716667 0.7750000
##  Accuracy  Accuracy  Accuracy
## 0.7900000 0.7800000 0.8016667

us_nb_accuracy

## Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy
##     0.89     0.89     0.89     0.89     0.89     0.89     0.89     0.89
## Accuracy Accuracy
##     0.89     0.89

os_nb_accuracy

## Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy
##     0.89     0.89     0.89     0.89     0.89     0.89     0.89     0.89
## Accuracy Accuracy
##     0.89     0.89

smote_nb_accuracy

## Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy Accuracy
##     0.89     0.89     0.89     0.89     0.89     0.89     0.89     0.89
## Accuracy Accuracy
##     0.89     0.89

us_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8904348 0.8937282 0.8872180 0.8902439 0.8884956 0.8955752 0.8912621
## Precision Precision Precision
## 0.8949416 0.8950382 0.8945518

os_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.9300412 0.9363958 0.9190141 0.9325843 0.9178082 0.9259259 0.9320388
## Precision Precision Precision
## 0.9405594 0.9415808 0.9323843
```

```
smote_glm_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.8963415 0.8900204 0.8886555 0.8922495 0.8915929 0.8927126 0.8909853
## Precision Precision Precision
## 0.8936170 0.8966203 0.8976378

us_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.9133065 0.9081836 0.9046729 0.9178645 0.9122449 0.9134809 0.9166667
## Precision Precision Precision
## 0.9068826 0.9137577 0.9105691

os_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.9526627 0.9418605 0.9535604 0.9291785 0.9539877 0.9506849 0.9503106
## Precision Precision Precision
## 0.9482201 0.9492537 0.9548193

smote_rf_precision

## Precision Precision Precision Precision Precision Precision Precision
## 0.9026915 0.9079498 0.9051546 0.9043977 0.9043659 0.9042770 0.9079755
## Precision Precision Precision
## 0.8953488 0.9068826 0.9060665

us_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
##      0.89      0.89      0.89      0.89      0.89      0.89      0.89
## Precision Precision Precision
##      0.89      0.89      0.89

os_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
##      0.89      0.89      0.89      0.89      0.89      0.89      0.89
## Precision Precision Precision
##      0.89      0.89      0.89

smote_nb_precision

## Precision Precision Precision Precision Precision Precision Precision
##      0.89      0.89      0.89      0.89      0.89      0.89      0.89
## Precision Precision Precision
##      0.89      0.89      0.89

us_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.9588015 0.9606742 0.8838951 0.9569288 0.9400749 0.9475655 0.8595506
```

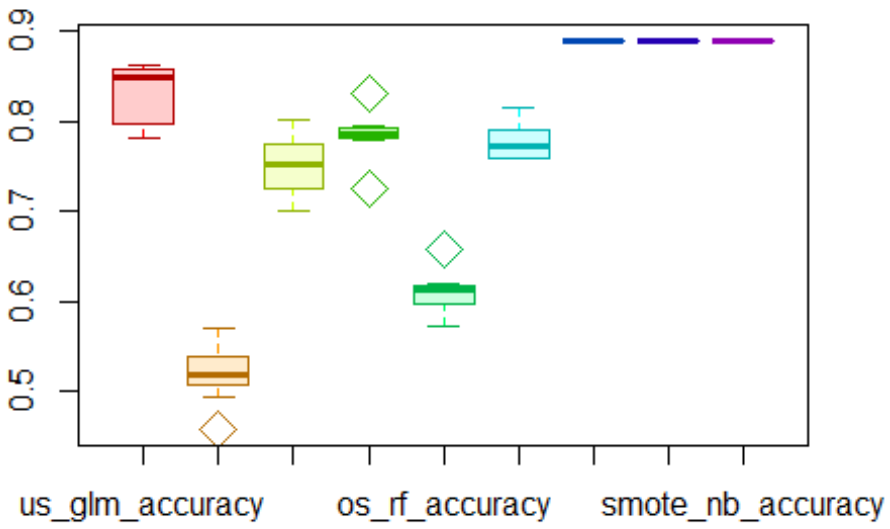
```
##    Recall    Recall    Recall
## 0.8614232 0.8782772 0.9531835

os_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.4232210 0.4962547 0.4887640 0.4662921 0.5018727 0.5617978 0.5393258
##    Recall    Recall    Recall
## 0.5037453 0.5131086 0.4906367

smote_glm_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8258427 0.8183521 0.7921348 0.8838951 0.7546816 0.8258427 0.7958801
##    Recall    Recall    Recall
## 0.7865169 0.8445693 0.8539326

us_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8483146 0.8520599 0.9063670 0.8370787 0.8370787 0.8501873 0.7621723
##    Recall    Recall    Recall
## 0.8389513 0.8333333 0.8389513

os_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.6029963 0.6067416 0.5767790 0.6142322 0.5823970 0.6498127 0.5730337
##    Recall    Recall    Recall
## 0.5486891 0.5955056 0.5936330

smote_rf_recall

##    Recall    Recall    Recall    Recall    Recall    Recall    Recall
## 0.8164794 0.8127341 0.8220974 0.8857678 0.8146067 0.8314607 0.8314607
##    Recall    Recall    Recall
## 0.8651685 0.8389513 0.8670412

us_nb_recall

## Recall Recall Recall Recall Recall Recall Recall Recall Recall Recall
##      1      1      1      1      1      1      1      1      1      1

os_nb_recall

## Recall Recall Recall Recall Recall Recall Recall Recall Recall Recall
##      1      1      1      1      1      1      1      1      1      1

smote_nb_recall

## Recall Recall Recall Recall Recall Recall Recall Recall Recall Recall
##      1      1      1      1      1      1      1      1      1      1

us_glm_f1
```

```
##        F1        F1        F1        F1        F1        F1        F1
## 0.9233544 0.9259928 0.8855535 0.9223827 0.9135578 0.9208371 0.8751192
##        F1        F1        F1
## 0.8778626 0.8865784 0.9229374

os_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.5817246 0.6487148 0.6381418 0.6217228 0.6489104 0.6993007 0.6832740
##        F1        F1        F1
## 0.6560976 0.6642424 0.6429448

smote_glm_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8596491 0.8526829 0.8376238 0.8880527 0.8174442 0.8579767 0.8407517
##        F1        F1        F1
## 0.8366534 0.8698168 0.8752399

us_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8796117 0.8792271 0.9055192 0.8756121 0.8730469 0.8806984 0.8323108
##        F1        F1        F1
## 0.8715953 0.8716944 0.8732943

os_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.7385321 0.7380410 0.7187865 0.7395716 0.7232558 0.7719689 0.7149533
##        F1        F1        F1
## 0.6951364 0.7318757 0.7321016

smote_rf_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.8574238 0.8577075 0.8616290 0.8949858 0.8571429 0.8663415 0.8680352
##        F1        F1        F1
## 0.8800000 0.8715953 0.8861244

us_nb_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989
##        F1        F1        F1
## 0.9417989 0.9417989 0.9417989

os_nb_f1

##        F1        F1        F1        F1        F1        F1        F1
## 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989
##        F1        F1        F1
## 0.9417989 0.9417989 0.9417989
```

```
smote_nb_f1
```

```
##        F1        F1        F1        F1        F1        F1        F1
## 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989 0.9417989
##        F1        F1        F1
## 0.9417989 0.9417989 0.9417989
```

```
c1 <- rainbow(10)
c2 <- rainbow(10, alpha=0.2)
c3 <- rainbow(10, v=0.7)
boxplot(df, col=c2, medcol=c3, whiskcol=c1, staplecol=c3, boxcol=c3,
outcol=c3, pch=23, cex=2)
```




```
mean(us_nb_accuracy)
```

```
## [1] 0.89
```

```
mean(us_nb_precision)
```

```
## [1] 0.89
```

```
mean(us_nb_recall)
```

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

```
mean(us_nb_f1)
```

```
## [1] 0.9417989
```

```r
mean(os_nb_accuracy)
```

```
## [1] 0.89
```

```r
mean(os_nb_precision)
```

```
## [1] 0.89
```

```r
mean(os_nb_recall)
```

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

```r
mean(os_nb_f1)
```

```
## [1] 0.9417989
```

```r
mean(smote_nb_accuracy)
```

```
## [1] 0.89
```

```r
mean(smote_nb_precision)
```

```
## [1] 0.89
```

```r
mean(smote_nb_recall)
```

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

```r
mean(smote_nb_f1)
```

```
## [1] 0.9417989
```

```r
a <- matrix(

c(mean(us_glm_accuracy),mean(us_glm_precision),mean(us_glm_recall),mean(us_glm_f1),

mean(os_glm_accuracy),mean(os_glm_precision),mean(os_glm_recall),mean(os_glm_f1),

mean(smote_glm_accuracy),mean(smote_glm_precision),mean(smote_glm_recall),mean(smote_glm_f1)),
  nrow=3,
  ncol=4,
  byrow = TRUE
)

a
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.8298333 0.8921490 0.9200375 0.9054176
## [2,] 0.5206667 0.9308333 0.4985019 0.6485074
## [3,] 0.7510000 0.8930433 0.8181648 0.8535891
```