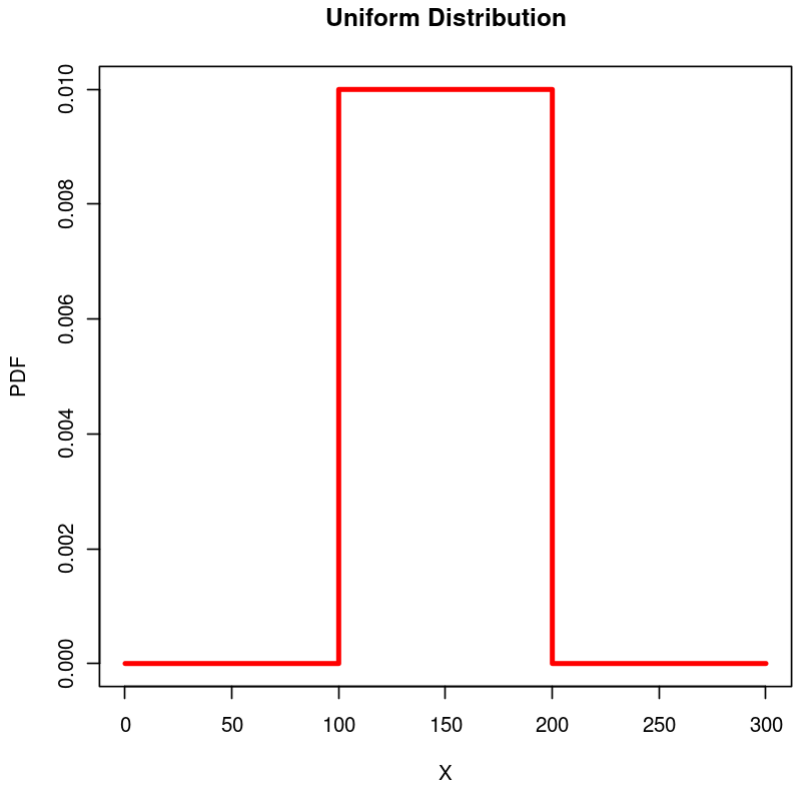
In [12]:

```
# Aditya Sarkar
# B19003

# Question 1
# considered a population here
population <- 1000000
a = 100
b = 200

# formed x and y
x <- seq(0, 300, length=population)
y <- dunif(x, min = a, max = b)

#Plotting Population
plot(x, y, type = 'l', lwd = 3, ylim = c(0, 0.01), col='red',
     xlab='X', ylab='PDF', main='Uniform Distribution')
```

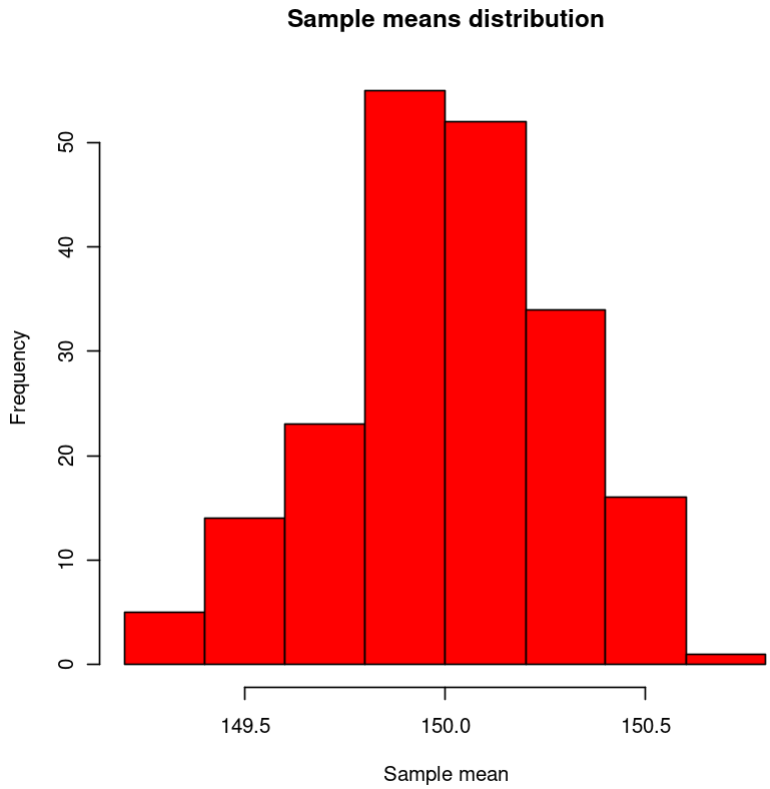


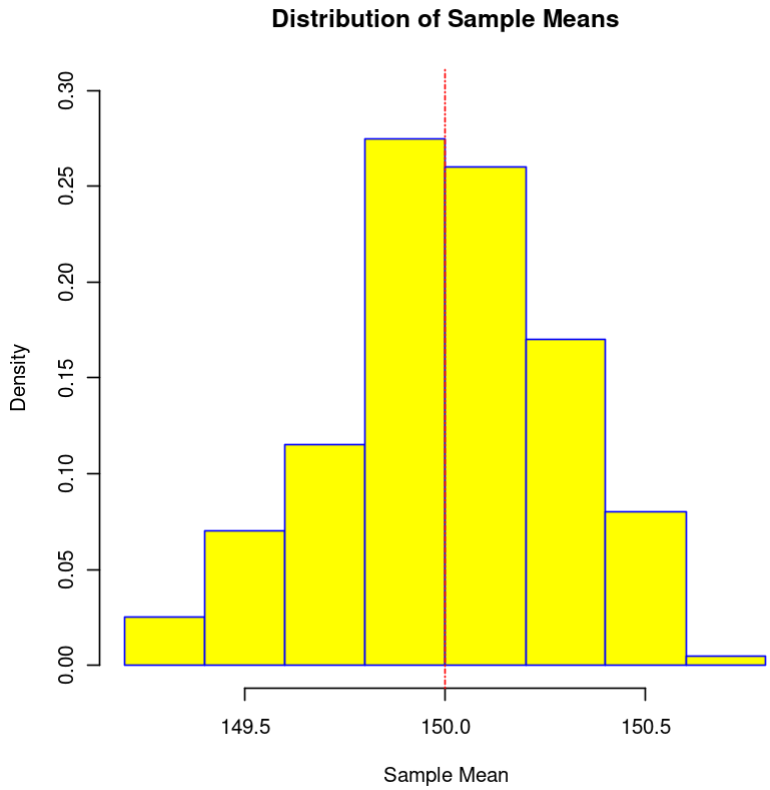
In [13]:

```
# Function for T-test and Rank-Sum test
Check_Ttest <- function(n, alpha = 0.05, itr = 200){

    # array of 0s
    nh = rep(0, itr)
    smean = rep(0,itr)
    nh1 = rep(0,itr)

    for(i in 1:itr){
#         print(i)

        # sampling a sample from population
        sample_pop = sample(x=x,size=n)

        # calculating the mean
        smean[i] = mean(sample_pop)
```

```
        # Ttest statistics
        tteststat <- t.test(sample_pop, mu = mean(x),
                            alternative = "two.sided", conf.level = 1-alpha )
        p_value <- tteststat$p.value

        # Rank Sum Test statistics
        rteststat <- wilcox.test (sample_pop, mu = mean(x),
                                  alternative = "two.sided",conf.level = 1
                                  exact = F)
        p_value1 <- rteststat$p.value

        # Reject the null hypothesis for pvalues smaller than 5% (as per alph
        # that are smaller than the reference value.
        if (p_value<alpha){
          nh[i] = 1
        }
        else{
          nh[i] = 0
        }

        # Same thing but for Rank Sum test
        if (p_value1<alpha){
          nh1[i] = 1
        }
        else{
          nh1[i] = 0
        }

    }

    # Printing Type 1 error
    cat('n=',n,'\n')
    cat('alpha = ',alpha,'\n')
    cat('Type1 error of T-test',sum(nh)/itr,'\n')
    cat('Type1 error of Rank Sum Test',sum(nh1)/itr,'\n')

    return(smean)
}

n = 100000
smean <- Check_Ttest(n=n)

# plotting the distribution of sample means
p2 <- hist(smean,main="Sample means distribution",xlab="Sample mean",col="red

# normlaized it to get the density
p2$counts=p2$counts/sum(p2$counts)

# plotted the distribution
plot(p2,main="Distribution of Sample Means",
       xlab="Sample Mean",ylab='Density',col="yellow",
       border="blue",ylim=c(0,0.3))

# Marked the horizonatal line
abline(v = mean(x), col="red", lty=4)
```

```
n= 1e+05
alpha =  0.05
Type1 error of T-test 0.06
Type1 error of Rank Sum Test 0.06
```





In [15]:

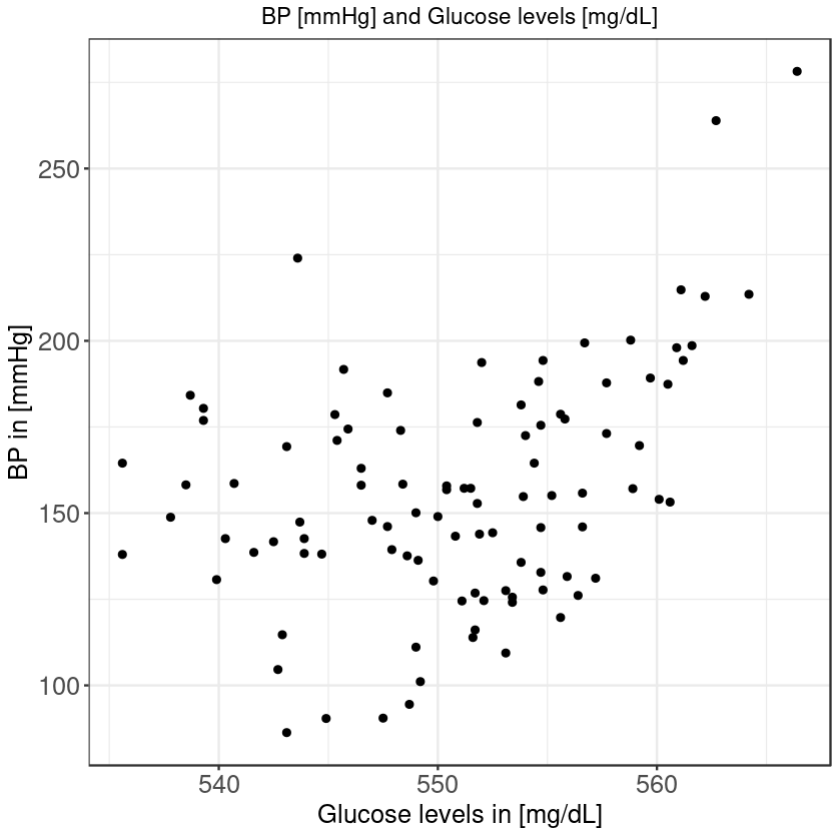
```
# Question 3

# libraries
library(readxl)
library(ggpubr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(MASS)
library(scales)
```

```
library(epitools)

# importing data and renaming columns
df <- data.frame(read_excel("Glucose_BP_levels.xlsx"))
names(df) = c("Datapoint","Glucose_L","BP_P")

# Considered two columns
Glu <- df$Glucose_L
BP <- df$BP_P

#Mean of two columns
cat("Mean of Glucose Level",mean(Glu))
cat('\n')
cat("Mean of Blood Pressure",mean(BP))
cat('\n')

#Variance of 2 columns
cat("Variance of Glucose Level",var(Glu))
cat('\n')
cat("Variance of Blood Pressure",var(BP))
cat('\n')

#Scatter plot
# Referred from Lab 10
plt1 <- ggplot() +
  geom_point(aes(x = df$Glucose_L, y = df$BP_P))+
  theme_bw()+
  labs( x = "Glucose levels in [mg/dL]",y = "BP in [mmHg]")+
  ggtitle("BP [mmHg] and Glucose levels [mg/dL]")+
  theme( axis.text = element_text( size =15) ,
         axis.title = element_text( size =15) ,
         plot.title = element_text( hjust = 0.5))
plot(plt1)

#  Compute the model parameters using glm(). This
# command computes generalized linear models.

model <- glm(df$BP_P ~ df$Glucose_L, data = df)
# summary(model)

# Model : y = mx +c
b = model$coefficients[1]
m = model$coefficients[2]
cat(c('m:',m))
cat('\n')
cat(c('b:',b))
cat('\n')

#Computation of Standard error of Intercept i.e. SEb
# Necessary for calculating the confidence interval of b and m
Seb <- summary(model)$coefficients[,2][1]
Sem <- summary(model)$coefficients[,2][2]
cat(c('SEm:',Sem))
cat('\n')
cat(c('SEb:',Seb))
cat('\n')

# Getting the regression line
# directly referred from lab 10
Reg <- (m*df$Glucose_L) + b

p11 <- ggplot () +
geom_point(aes (x = df$Glucose_L , y= df$BP_P))+
geom_line( aes (x= df$Glucose_L , y= Reg) , color = " blue ")+
```

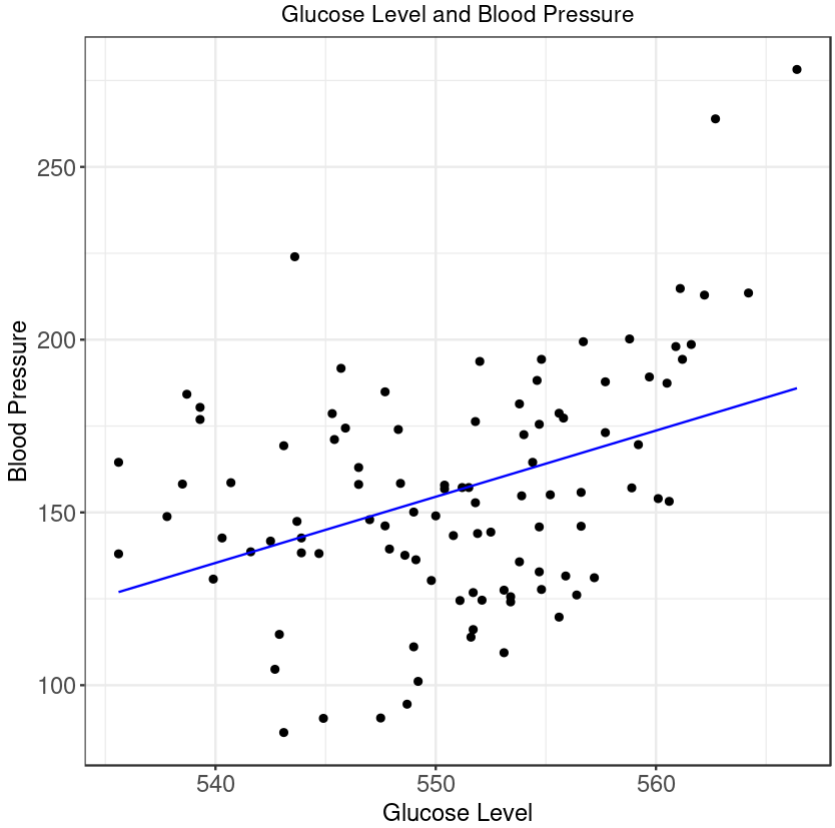
```
theme_bw() +
labs(x ="Glucose Level",y = "Blood Pressure")+
ggtitle("Glucose Level and Blood Pressure")+
theme(axis.text = element_text(size =14),
axis.title = element_text(size =14),
plot.title = element_text(hjust = 0.5) )

plot(p11)

# Threshold
alpha = 0.03

# Confidence Interval Formula
# Referred from Lab 10
CIm_max <- m + qt(1-alpha,(length(df$Datapoint)-2))*Sem
CIm_min <- m - qt(1-alpha,(length(df$Datapoint)-2))*Sem

# Confidence Interval for Intercept
CIb_max <- b + qt(1-alpha,(length(df$Datapoint)-2))*Seb
CIb_min <- b - qt(1-alpha,(length(df$Datapoint)-2))*Seb

# Confidence interval for slope and Intercept
cat(c('CI of m max:',CIm_max))
cat('\n')
cat(c('CI of m min:',CIm_min))
cat('\n')
cat(c('CI of b max:',CIb_max))
cat('\n')
cat(c('CI of b min:',CIb_min))
cat('\n')

# Regression maximum and minimum line
Reg_max <- (CIm_max * df$Glucose_L) + CIb_min
Reg_min <- (CIm_min * df$Glucose_L) + CIb_max

# Plotting the scatterplot
# Directly referred from Lab 10
plt2 <- ggplot() +
  geom_point(aes(x= df$Glucose_L , y= df$BP_P ))+
  # Optimal regression line
  geom_line( aes(x= df$Glucose_L , y= Reg ) , color = " orange ")+
  # regression with upper -limit slope
  geom_line( aes(x= df$Glucose_L , y= Reg_max ) , color = " green ")+
  # Regression with lower -limit slope
  geom_line( aes(x= df$Glucose_L , y= Reg_min ) , color = " green ")+
  #Auto - generated slope
  geom_smooth( aes(x= df$Glucose_L , y= df$BP_P ) ,formula = y ~ x ,method =
  # Inserting regression equation
  stat_regline_equation(aes(x=df$Glucose_L , y= df$BP_P))+
  theme_bw() +
  labs (x ="Glucose levels [mg/dL]",
        y = "BP [mmHg]")+
  ggtitle("BP [mmHg] and Glucose levels [mg/dL]")+
  theme( axis.text = element_text ( size =14) ,
         axis.title = element_text ( size =14) ,
         plot.title = element_text ( hjust = 0.5) )

plot(plt2)

# Calculting residuals
e <- Reg - df$BP_P

# Directly taken from Lab 10
plt3 <- ggplot() +
```

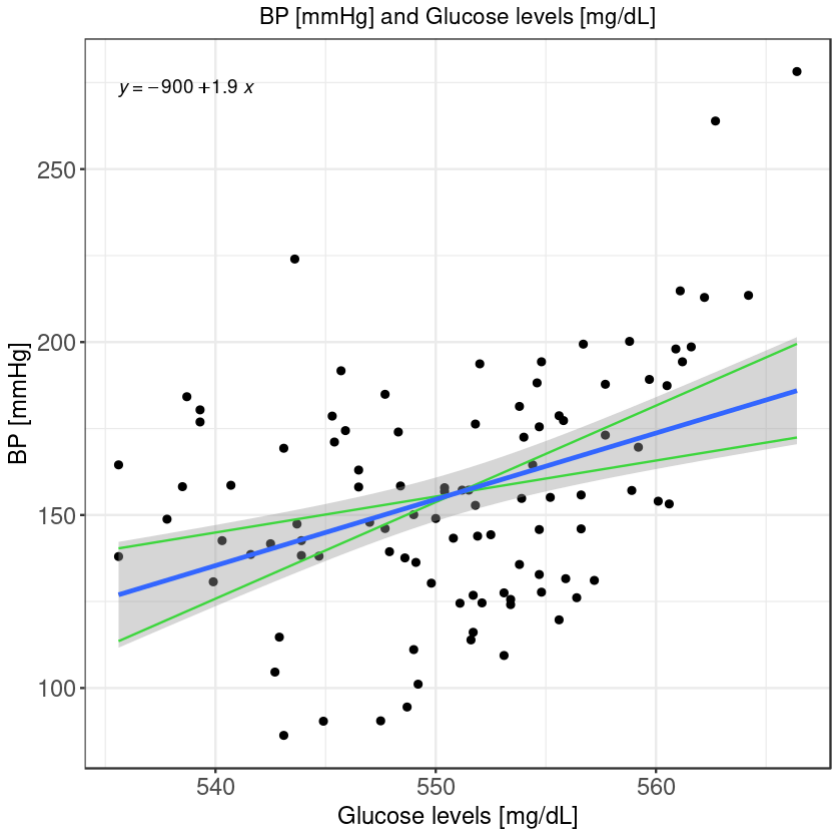
```
  geom_point(aes(x = df$BP_P, y = e))+
  geom_line(aes(x = df$BP_P, y = 0), color = "Red")+
  theme_bw()+
  labs( x = "BP [mmHg]",y = "Residual")+
  ggtitle("BP [mmHg] and Residual")+
  theme( axis.text = element_text( size =12) ,
         axis.title = element_text( size =12) ,
         plot.title = element_text( hjust = 0.5))
plot(plt3)
```

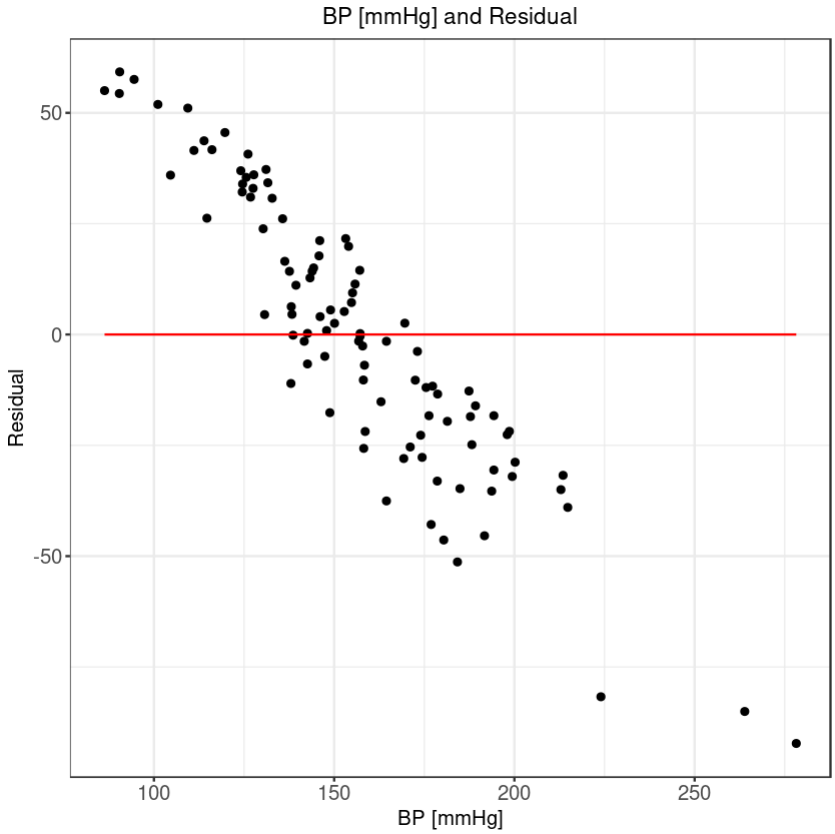
```
Mean of Glucose Level 550.883
Mean of Blood Pressure 156.227
Variance of Glucose Level 47.23718
Variance of Blood Pressure 1152.339
m: 1.91542223917039
b: -898.9465493809
SEm: 0.459879195963488
SEb: 253.359150077889
```




```
CI of m max: 2.79048731872174
CI of m min: 1.04035715961903
CI of b max: -416.850932175436
CI of b min: -1381.04216658636
```


Glucose Level and Blood Pressure
Blood Pressure
250
200
150
100
540
550
560
Glucose Level


BP [mmHg] and Glucose levels [mg/dL]
y = −900 +1.9 x
BP [mmHg]
250
200
150
100
540
550
560
Glucose levels [mg/dL]



In [16]:

```
# Libraries
library(readxl)
library(ggpubr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(MASS)
library(scales)
library(epitools)

# Dataframe
df <- data.frame(read_excel("India_Health_Database.xlsx",
                                    sheet = "All data"))
# Population Data
pop_data <- df[c("State","Population..2011","Population.2018")]

# Renaming columns
names(pop_data)[2]<-"population_2011"
names(pop_data)[3]<-"population_2018"

# Extrapolated the values for other missing years
# Assumed Linear relation
extrapolate <- function(year){
    l = length(pop_data$State)
    estm = rep(0,l)
    for(i in 1:l){
        temp = (pop_data[i,"population_2018"]-pop_data[i,"population_2011"])/
        estm[i] = pop_data[i,'population_2011'] + temp
    }
    return(estm)
}

# Plotting
plotting <- function(ncol=3, nrow=0){
    # defining a dataframe for making the quick plot
    df <- data.frame(matrix(ncol = ncol, nrow = nrow))
```

```
    colnames(df) <-c("Year","State", "Population")
    # iterating over the years
    for(i in 2011:2018){
        # iterating over the states
        for(j in 1:length(pop_data$State)){
            # Adding elements in the dataframe
            df[nrow(df) + 1,] <- c(i,pop_data$State[j],pop_data[j,i-2010])
        }
    }
    # Ensuring that the values are all numeric
    df$Population <- as.numeric(df$Population)
    # Making quick plot
    p<-qplot(Year,Population, group=State, data=df, col=State,geom="line",log
    plot(p)
}

# Code for extrapolation
year = list(2012,2013,2014,2015,2016,2017)
for(i in year){
    b = paste("population_", as.character(i), sep ="")
    pop_data[b] = extrapolate(i)
}

# ordering the columns
pop_data <- pop_data[,order(names(pop_data))]

# Calling the plotting function
plotting()

# Selecting required columns
col <- rep('skip',37)
col[0:13] = 'guess'

# Read the data
Malaria <- data.frame(read_excel("India_Health_Database.xlsx",
                                 sheet = "Vector borne diseases",
                                 col_types =col,skip=2))
# renamed the columns
names(Malaria) <- c("States","2013C","2013D","2014C","2014D","2015C","2015D",
                    "2016C","2016D","2017C","2017D","2018C","2018D")

# Calculated the mortality = Deaths / healthy patients
year = list(2013,2014,2015,2016,2017,2018)
for(i in year){
    b = paste(as.character(i),'_Mortality', sep ="")
    a = paste(as.character(i),'D',sep="")
    p = paste("population_", as.character(i), sep ="")
    Malaria[b] = Malaria[a]/pop_data[p]
}

# Calculting mean and std dev
mean1 = rep(0,length(Malaria$States))
conf_min = rep(0,length(Malaria$States))
conf_max = rep(0,length(Malaria$States))

# referred from lm documentation : https://www.rdocumentation.org/packages/st
for (i in 1:length(Malaria$States)){
  model <- lm(c(Malaria$'2013_Mortality'[i],Malaria$'2014_Mortality'[i],Malar
                Malaria$'2016_Mortality'[i],Malaria$'2017_Mortality'[i],M
  mean1[i]      = confint(model,level=0)[1]

  # 68.25% of data lies within 1 standard deviation in Gaussian curve
  conf_min[i] = confint(model,level=0.6825)[1]
  conf_max[i] = confint(model,level=0.6825)[2]
```

```
}

# Storing the means and confidence max and min in Malaria dataframe
Malaria["Mean"] <- mean1
Malaria["CImin"] <- conf_min
Malaria['CImax'] <- conf_max

cat("Mean Mortality & Confidence Interval of States:")
cat('\n')
print(Malaria[c("States","Mean","CImin","CImax")])

# directly referred from lm documentation : https://www.rdocumentation.org/pa
model <- lm(Malaria$'Mean'~1)
avg <- confint(model,level=0)[1]

# 68.25% of data lies within 1 standard deviation in Gaussian curve
CI_min <- confint(model,level=0.6825)[1]
CI_max <- confint(model,level=0.6825)[2]

cat("\n\n")
cat("Mean Mortality and CI computed:\n")
cat(c(avg,'(',CI_min,',',CI_max,')'))
cat('\n')

# Kruskal Test - small sample size, one way anova equivalent
kruskal_test <- kruskal.test(Malaria[c("2013_Mortality","2014_Mortality","201
                                       "2016_Mortality","2017_Mortality",
p_value <- kruskal_test$p.value
cat('\n')
cat(c("p value:",p_value))
cat('\n')

# defining the plotting function similar as above
plotting1 <- function(ncol=2, nrow=0){
    df <- data.frame(matrix(ncol = 2, nrow = 0))
    colnames(df) <-c("Year","Mortality")
    for (i in 2013:2018){
      for (j in 1:length(Malaria$States)){
        df[nrow(df) + 1,] <- c(i,Malaria[j,i-2013+14])
      }
    }
    df$Mortality <- as.numeric(df$Mortality)

    p1 <- ggboxplot(df, x = "Year", y = "Mortality",
              color = "Year",
              order = 2013:2018,
              ylab = "Mortality", xlab = "Year")
    plot(p1)
  }

# called the plotting function
plotting1()

# Calculating the correlation
urban_pop <- df$Urban.Population....
for(i in year){
    s = paste(as.character(i),'_Mortality', sep ="")
    t = cor(x=urban_pop, y=Malaria[s],use = "complete.obs")
    cat(c("Correlation b/w urbanisation vs mortality for year ",i,":",t))
    cat('\n')
}
```

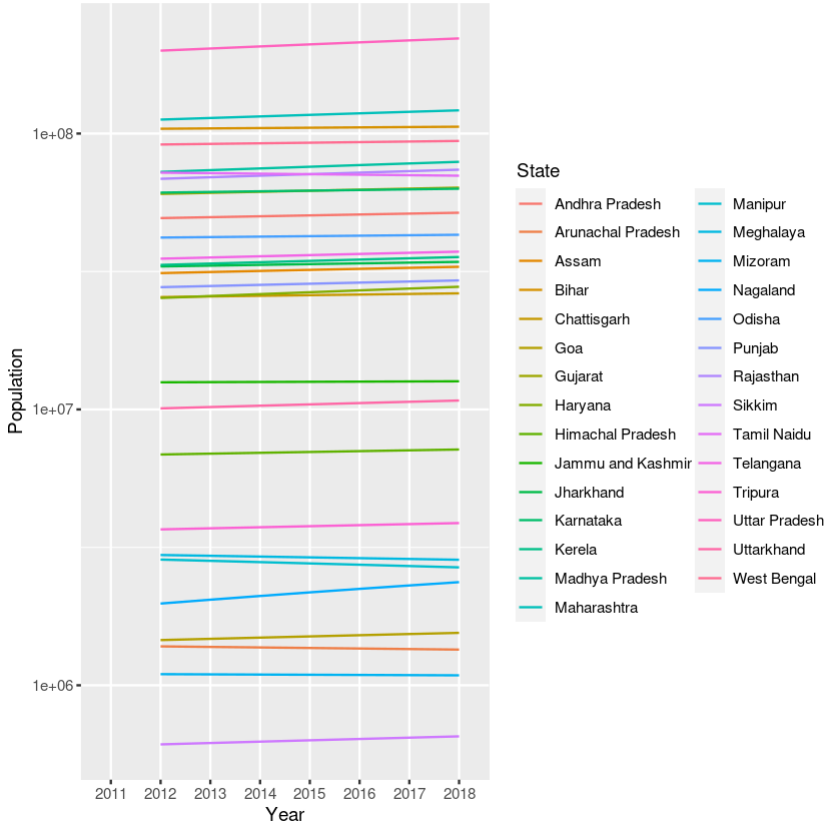
```
Warning message in plotting():
"NAs introduced by coercion"
```

```
Warning message:
"Removed 29 row(s) containing missing values (geom_path)."
New names:
* Cases -> Cases...2
* Deaths -> Deaths...3
* Cases -> Cases...4
* Deaths -> Deaths...5
* Cases -> Cases...6
* ...
```

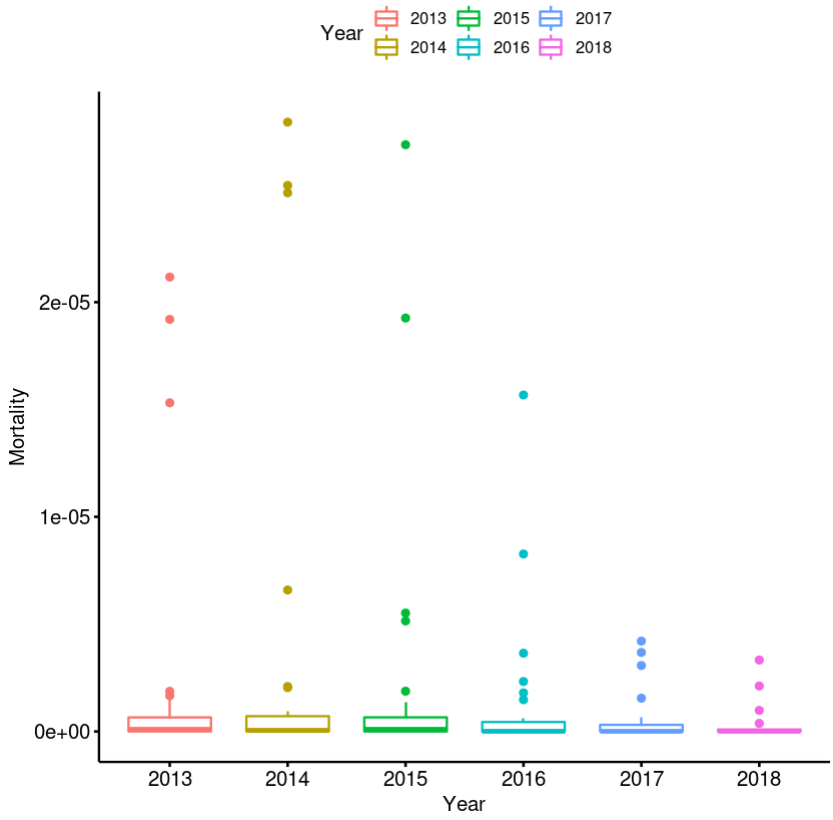
Mean Mortality & Confidence Interval of States:

	States	Mean	CImin	CImax
1	Andhra Pradesh	0.000000e+00	0.000000e+00	0.000000e+00
2	Arunachal Pradesh	4.755087e-06	2.105513e-06	7.404662e-06
3	Assam	1.552802e-07	9.995507e-08	2.106054e-07
4	Bihar	6.322572e-09	2.822379e-09	9.822765e-09
5	Chhattisgarh	1.815842e-06	1.429488e-06	2.202196e-06
6	Goa	2.173010e-07	6.473585e-08	3.698662e-07
7	Gujarat	2.039003e-07	1.060478e-07	3.017529e-07
8	Haryana	4.387707e-08	1.878436e-08	6.896978e-08
9	Himachal Pradesh	0.000000e+00	0.000000e+00	0.000000e+00
10	Jammu and Kashmir	0.000000e+00	0.000000e+00	0.000000e+00
11	Jharkhand	2.018412e-07	1.351296e-07	2.685529e-07
12	Karnataka	5.367857e-09	-5.905885e-10	1.132630e-08
13	Kerala	6.682118e-08	3.618834e-08	9.745402e-08
14	Madhya Pradesh	2.375137e-07	1.243023e-07	3.507251e-07
15	Maharashtra	3.767689e-07	2.655889e-07	4.879489e-07
16	Manipur	6.298816e-08	-6.930155e-09	1.329065e-07
17	Meghalaya	1.593457e-05	1.109696e-05	2.077218e-05
18	Mizoram	1.313342e-05	8.199917e-06	1.806693e-05
19	Nagaland	5.265717e-07	2.878620e-07	7.652813e-07
20	Odisha	1.334333e-06	9.664715e-07	1.702194e-06
21	Punjab	0.000000e+00	0.000000e+00	0.000000e+00
22	Rajasthan	6.321938e-08	2.751888e-08	9.891989e-08
23	Sikkim	0.000000e+00	0.000000e+00	0.000000e+00
24	Tamil Nadu	0.000000e+00	0.000000e+00	0.000000e+00
25	Telangana	2.271997e-08	2.890545e-09	4.254939e-08
26	Tripura	6.892222e-06	2.724303e-06	1.106014e-05
27	Uttarakhand	0.000000e+00	0.000000e+00	0.000000e+00
28	Uttar Pradesh	0.000000e+00	0.000000e+00	0.000000e+00
29	West Bengal	3.862224e-07	2.778247e-07	4.946201e-07

```
Mean Mortality and CI computed:
1.6014548617748e-06 ( 8.62533369824965e-07 , 2.34037635372463e-06 )

p value: 0.169557863650864
```




```
Correlation b/w urbanisation vs mortality for year  2013 : 0.0126469612185968
Correlation b/w urbanisation vs mortality for year  2014 : 0.0441934249494912
Correlation b/w urbanisation vs mortality for year  2015 : 0.020293975457508
Correlation b/w urbanisation vs mortality for year  2016 : -0.0568859890028357
Correlation b/w urbanisation vs mortality for year  2017 : 0.0466640845307434
Correlation b/w urbanisation vs mortality for year  2018 : -0.186801100889142
```



In [ ]:

In [ ]: