Algorithm for graph sampling from large graph(第一题)

The name of algorithm that I invented is *"Customized Hybrid random walk and jumping sampling"*

Customized Hybrid random walking and jumping sampling

This is how it works.

- 1.Before we do sampling, first calculate correlation coefficiences between two certain kind of nodes under certain kind of edge from original graph. For example, the coefficient between number of user nodes (用户） and number of music nodes(音乐） that user complimented. (点赞）
- 2.Then, we randomly pick any starting node.
- 3.With probability of *p*(this value can be adjusted later), we simulate random walk or with probability of *(1-p)* we 'jump' to another starting point, that is **not connected** with previous starting node.
- 4.After sampling, we calculate correlation coefficiences specified in step 1 from sampled graph, and compare those coefficiences with original graph.
- 5.If the concordance rate of those two sets of coefficiences are high enough(it is determined by TA or other executor), end sampling, otherwise do sampling again.

Requirements from TA

Before I explain the advantages and disadvantages of my algorithm, let's check what kind of sampling graph TA wants, and what characteristics this graph has.

> Requirements1 **"TA希望采样得到的数据能尽量保持原始数据的特性"**

Because TA wants sample graph to have similar graph properties as original graph, we should use **probability sampling techniques!**(not non-probability sampling).

> Solution1 I adapted random walking as it is probability sampling and has been proved it's efficiency.

> Requirements2 **"图上有若干类型的点(node):用户， 短视频，音乐、特效等等。它们彼此之间也有若干类型的连边(edge)，代表着不同的关系，比如观 看、点赞、使用，创作。"**

This graph has "different kind of nodes", and also different kind of "edges". Thus, we **should capture the relationship between one node with another**, not just consider the basic properties of graph -- degree, distance, ratio of numbers of nodes to edges... etc. And it also implies us that our algorithm **shouldn't be biased to high degree noes, be free from problem of getting stuck!!**

> Solution2 Here I though of ***correlation coefficient***, to enable examination of sampled graph, whether it captures the relatiosnhip between different nodes. And introduced *p* and "jumping" to handle the shortcoming of random walking -- being biased to high degree node, since they have more edges incident to them.

Features of my algorithm(Advantages and Disadvantages)

Finally let see what are the advantages and disadvantages of my algorithm.

> Advantages

- Can well conserve the relationship between two certain nodes as it introduced correlation coefficiences.
- Not biased toward high-degree node by introduction of jumping
- Do not have problem of being stuck (isolated) also by the introduction of jumping
- Probability sampling

> Disadvantages

- Possibly completely leave one isolated community. Thus, could fail in capturing community structure.
- Inefficient in terms of time and cost.

Correlation Analysis (第二题)

Import libraries

```
library(DescTools)
library(dplyr)
library(ggplot2)
library(corrplot)
```

Import user_data_csv

I changed columns' name each to *"age"* and *"time"* for convenience.

```
user_data<-
read.csv("/Users/wooseokhan/Desktop/datascience/hw1/user_data.csv")
colnames(user_data) = c("age","time")
```

1. Data Cleaning

First I craeted dataframe to do data cleaning more conveniently.

```
#create dataframe
user_data_ex <- user_data
df_data <-data.frame(user_data_ex)
```

1-1 *"Age"* column data cleaning

I did data cleaning for both columns.

```
# change class as numeric
age <- user_data_ex[,1]
df_data$age <- as.numeric(age)
```

First, check how many Na are there

```
#check Na
table(is.na(df_data$age))
```

```
> table(is.na(df_data$age)) # 4 NULL

FALSE  TRUE
  996     4
```

we get 4 NAs, because there are 1000 rows total, its better to eliminate Na rows

```
# Na removal
df_data <- df_data %>% filter(!is.na(age))
```

We can also replace NULL with some representative data -- mode,mean...etc. here, I will use mode. Here is how I get mode

```
#get mode
age_num <- df_data$age
age_num_mode <-Mode(age_num)
age_num_mode[1]
```

```
> age_num_mode[1]
[1] 25
```

and replce Na.

```
# Replacement
df_data_2 <-data.frame(user_data_ex)
df_data_2$age <-as.numeric(age)
x <-df_data_2$age
df_data_2$age <- ifelse(is.na(x),age_num_mode[1],x)
```

Next, I will remove outliers, which affects correlation coefficient in a bad way, especially with Pearson's correlation coefficient, that I will adapt later.

Here, I adapted IQR Rule to detect and define outlier. I did it both for *df_data*(Na removed) and *df_data_2*(Na replaced by representative data)

```
# Detecting outliers
age_outlier <-boxplot.stats(df_data$age)$out
age_outlier # <=2.0, >=47.0
df_data$age <-
ifelse(df_data$age>46.0,NA,ifelse(df_data$age<3.0,NA,df_data$age))
df_data <- df_data %>% filter(!is.na(age))
****
df_data_2$age <-
ifelse(df_data_2$age>46.0,NA,ifelse(df_data_2$age<3.0,NA,df_data_2$age))
df_data_2 <- df_data_2 %>% filter(!is.na(age))
```

```
> age_outlier # <=2.0, >=47.0
[1]  47.0   0.0  -1.0 999.0  48.0 120.0   2.0   0.5
```

1-2 "*Time*" column data cleaning

It's similar with age column data cleaning.

```
# Change class of numeric
df_data$time <- as.numeric(df_data$time)
df_data_2$time <- as.numeric(df_data_2$time)
```

There was no Na in "*time*" column, thus I only did outlier removal.

```
> table(is.na(df_data$time))

FALSE
  965

> table(is.na(df_data_2$time))

FALSE
  969
```

```
# time outlier processing // IQR Rule
time_outlier <- boxplot.stats(df_data$time)$out
```

```
time_outlier # <=-1641/ >=122
time <-df_data$time
df_data$time <-
ifelse(df_data$time>121.0,NA,ifelse(df_data$time<10.0,NA,df_data$time))
df_data <-df_data %>% filter(!is.na(time))
table(is.na(df_data$time))

time_outlier_2 <- boxplot.stats(df_data_2$time)$out
time_outlier_2 # <=-1641/ >=122
time_2<-df_data_2$time
df_data_2$time <-
ifelse(df_data_2$time>121.0,NA,ifelse(df_data_2$time<10.0,NA,df_data_2$tim
e))
df_data_2 <-df_data_2 %>% filter(!is.na(time))
table(is.na(df_data_2$time))
```

```
> time_outlier # <=-1641/ >=122
[1] 2147483642      -1641        122        126        126        125  411535466

> time_outlier_2 # <=-1641/ >=122
[1] 2147483642      -1641        122        126        126        125  411535466
```

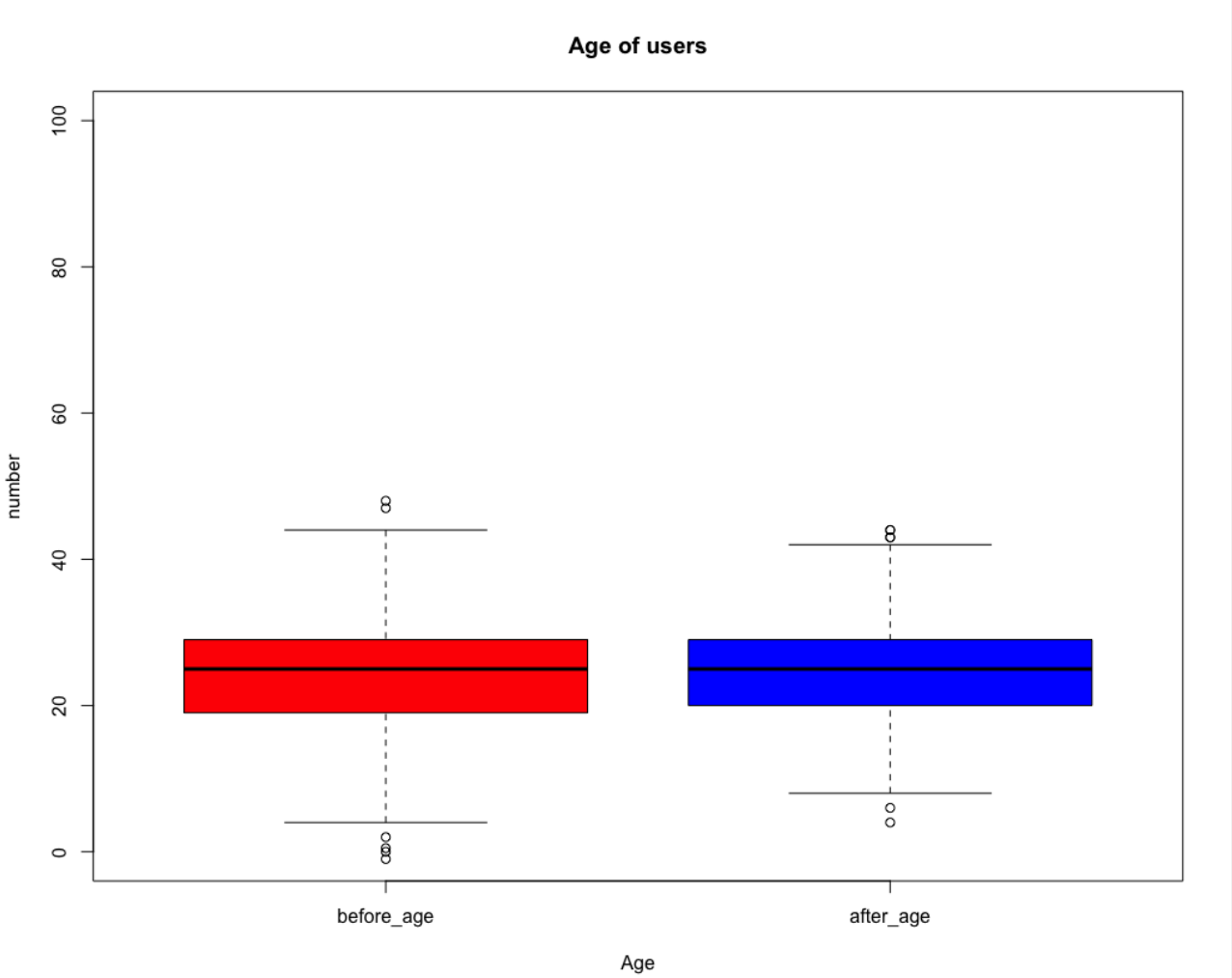
3. Evaluation of data cleaning

I draw two boxplots, before and after data cleaning, to evaluate whether my data cleaning process is appropriate process. Since I applied data cleaning for two columns, there are 4 boxplots overall.

3-1 *age* boxplot

```
age <- as.numeric(age)
boxplot(age,df_data$age,
        col = c("red","blue"),
        main = "Age of users",
        xlab = "Age",
        ylab = "number",
        ylim = c(0,100),
        names = c("before_age","after_age")
        )
```

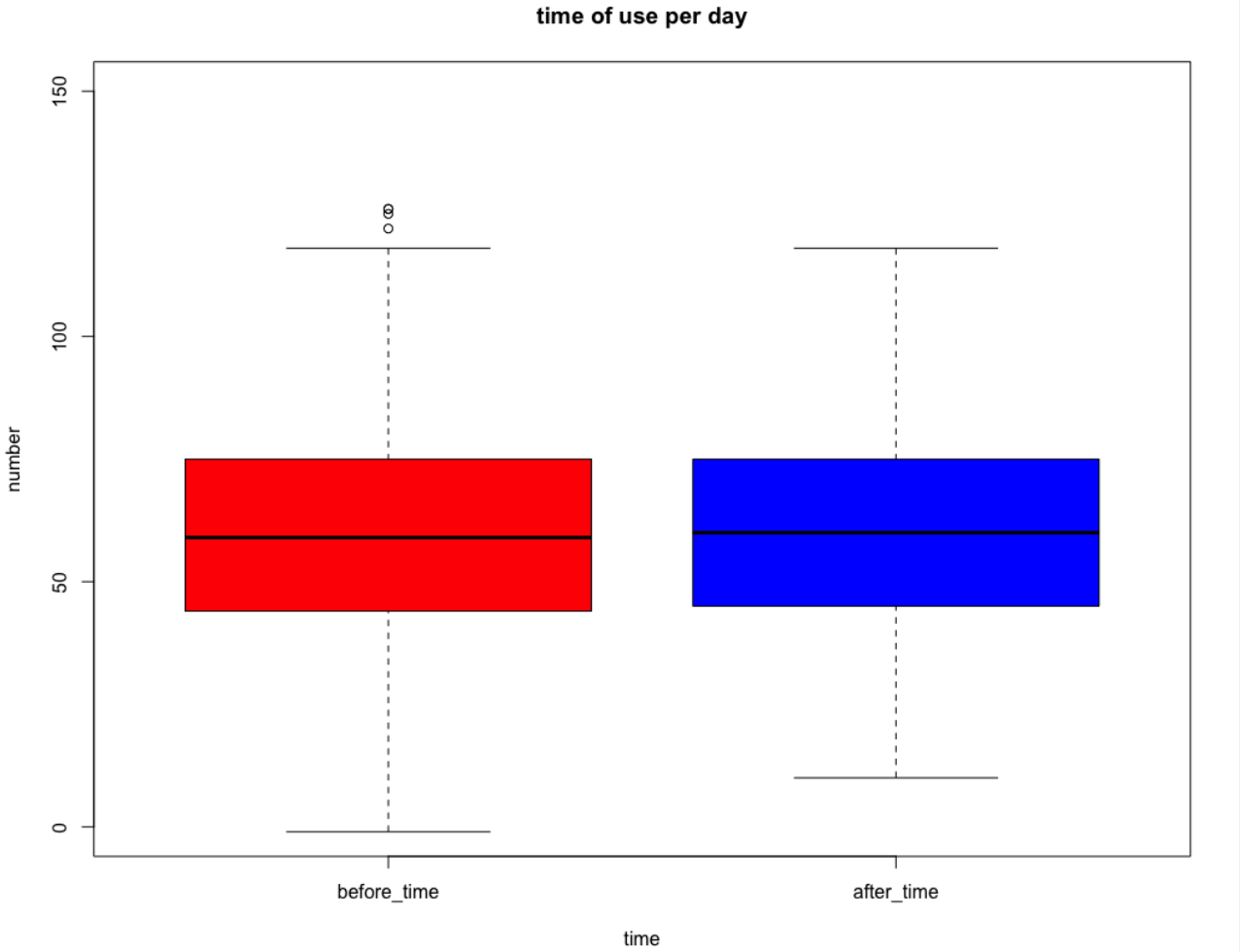
Age boxplot



3-2 *time* boxplot

```
boxplot(time, df_data$time,
        col = c("red","blue"),
        main = "time of use per day",
        xlab = "time",
        ylab = "number",
        ylim = c(0,150),
        names = c("before_time","after_time")
)
```

Time boxplot



Also, I compared the size of data after cleaning to see whether there was too much removal so that can affect the analyzing process, which shouldn't happen.

```
before = nrow(data.frame(user_data_ex)) #1000
after = nrow(df_data) #965
percent of removed data = (1-(after/before))*100 #3.5%
```

```
> (1-(after/before))*100 #3.5%
[1] 3.5
```

There were only 25 datas removed, which is 3.5% of whole data, so it's trivial.

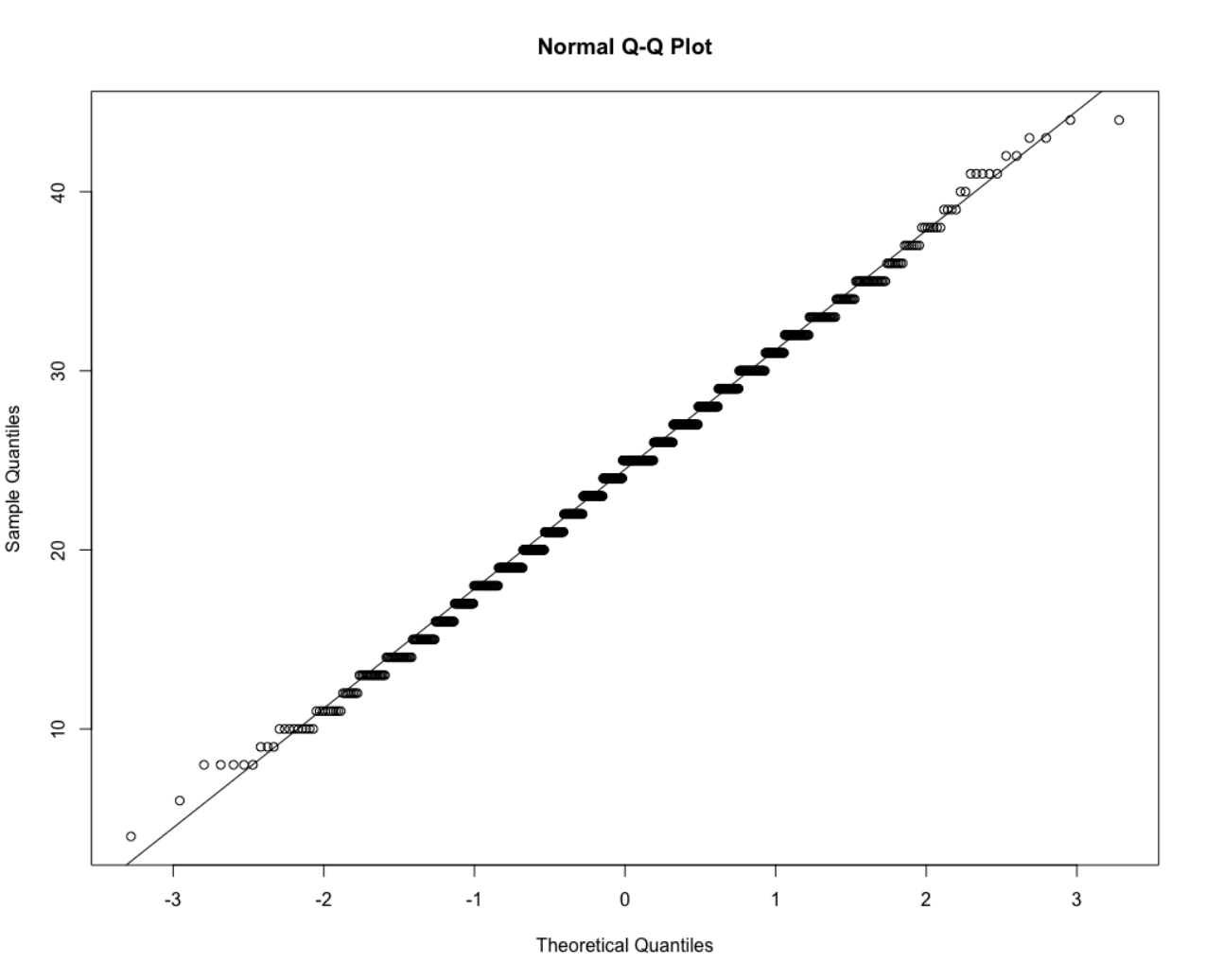
4. Correlation Analysis

I only covered *df_data*(Na removed)

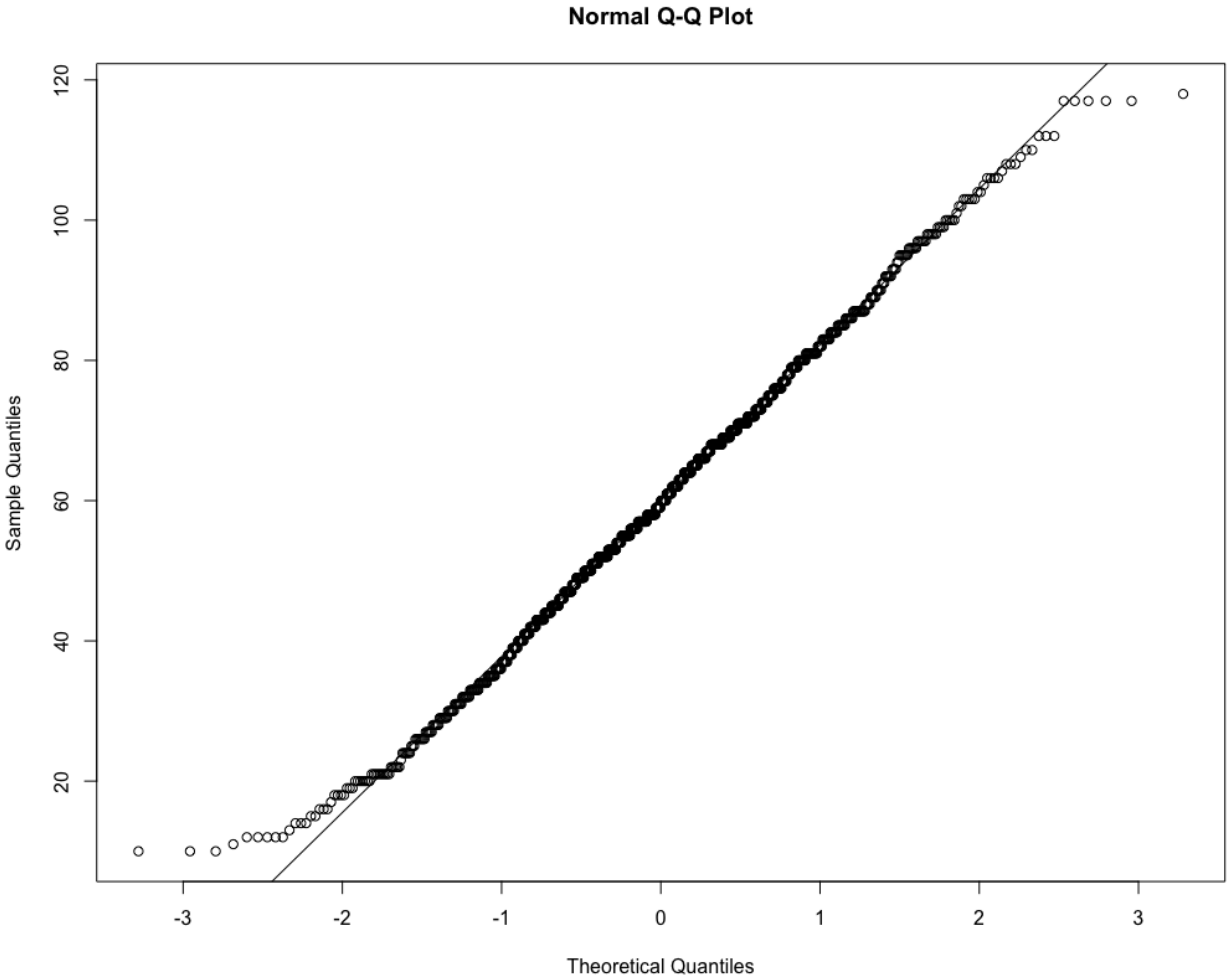
First, I did normality test to check whether the given data has been drawn from a normally distributed population. I used *Q-Q plot* for normality test, for it's convinent and very intuitive.

```
# Normality test - QQ plot
qqnorm(df_data$age)
qqline(df_data$age)
qqnorm(df_data$time)
qqline(df_data$time)
```

Age Q-Q plot



Time Q-Q plot



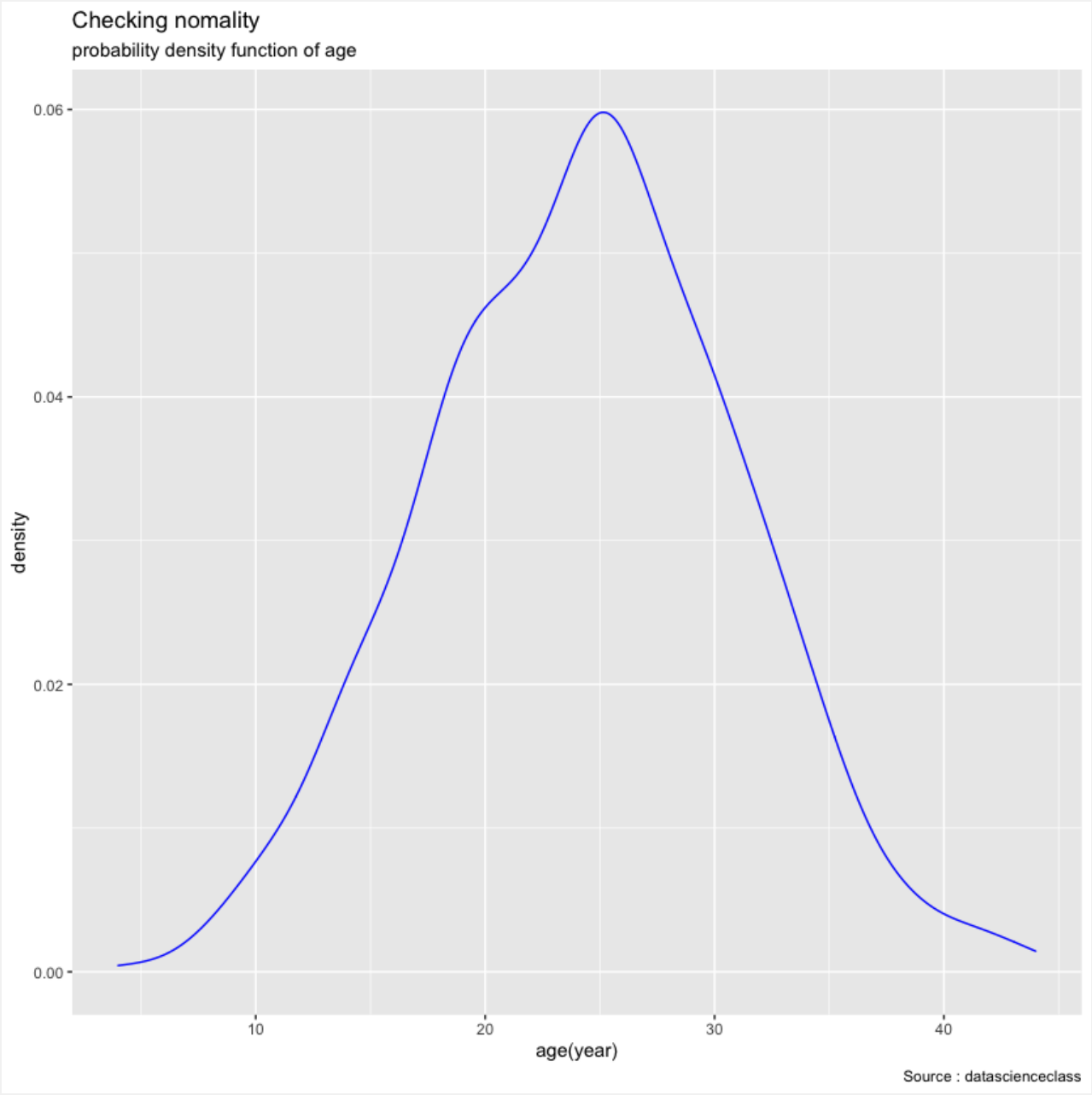
From the graph,we can easily find that both data follow normal distribution. Now,we can use *Pearson correlation coefficient* to explain the relationship between user's *age* and *time of use per day*!!

We can also check the data follows normal distribution by drawing *probability density function*.

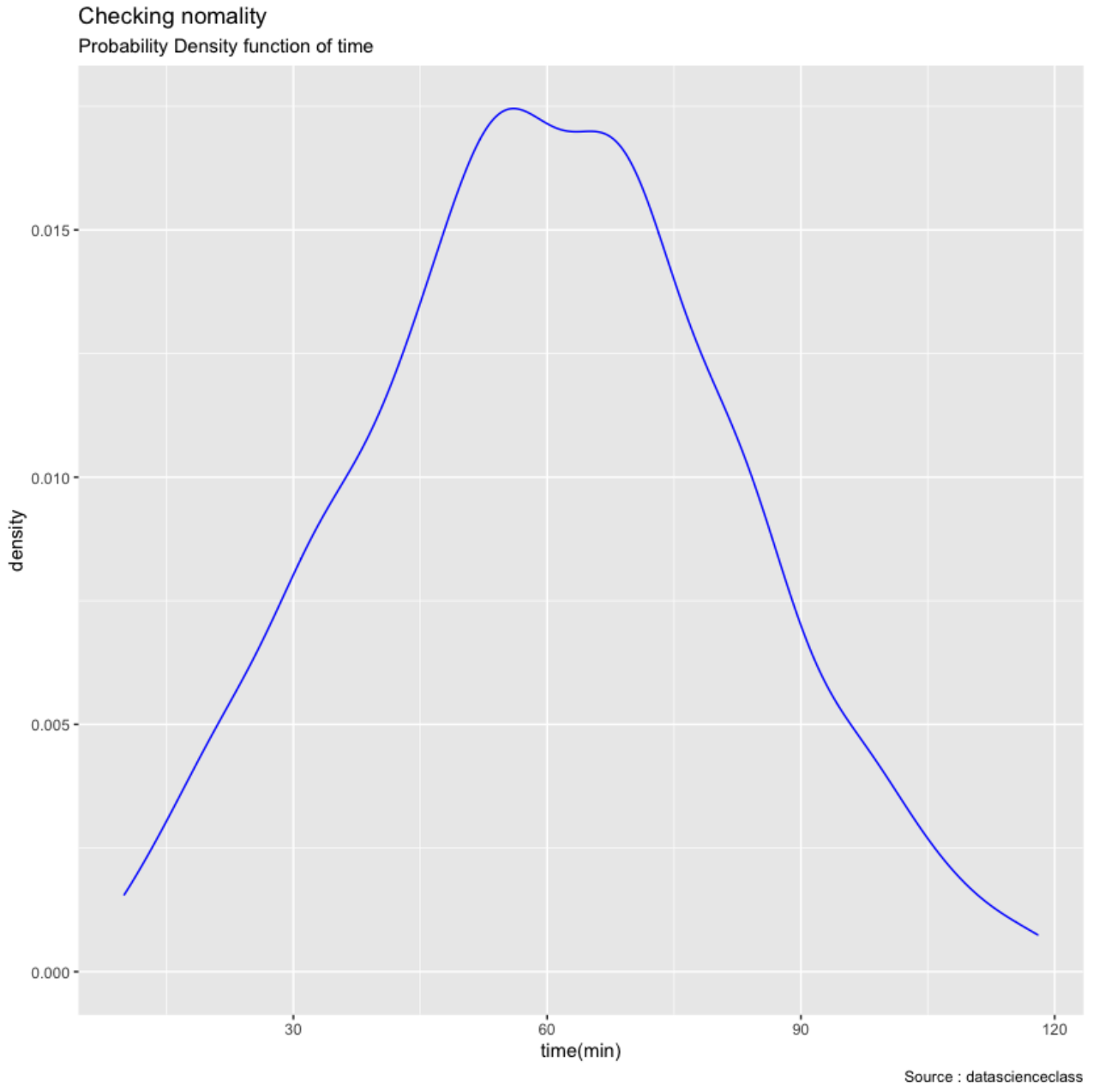
```
# Density plot
ggplot(data=df_data)+
  geom_density(mapping =aes(x=age),colour = 'blue')+
  labs(subtitle = "Relationship between age and time",
       y="density",
       x="age(year)",
       title="Density line",
       caption = "Source : datascienceclass")

ggplot(data = df_data)+
  geom_density(mapping = aes(x=time),colour = "blue")+
  labs(subtitle = "Probability Density function of time",
       y="density",
       x="time(min)",
       title="Checking nomality",
       caption = "Source : datascienceclass")
```

Probability density function(age)



Probabilty density function(time)



Next, I calculated *covariance* to roughly know the relationship between two variables, before specifically explore it.

```
# Covariance
cov(df_data$time,df_data$age) #109.5592
```

```
> cov(df_data$time,df_data$age) #109.5592
[1] 109.5592
```

Since covariance is 109.5592, we can think that two variables are positively related, and they move in the same direction in general.

Finally, let's calculate *pearson correlation coefficient*

```
cor(df_data$age,df_data$time,use='complete.obs',method='pearson')
#0.7456772
```

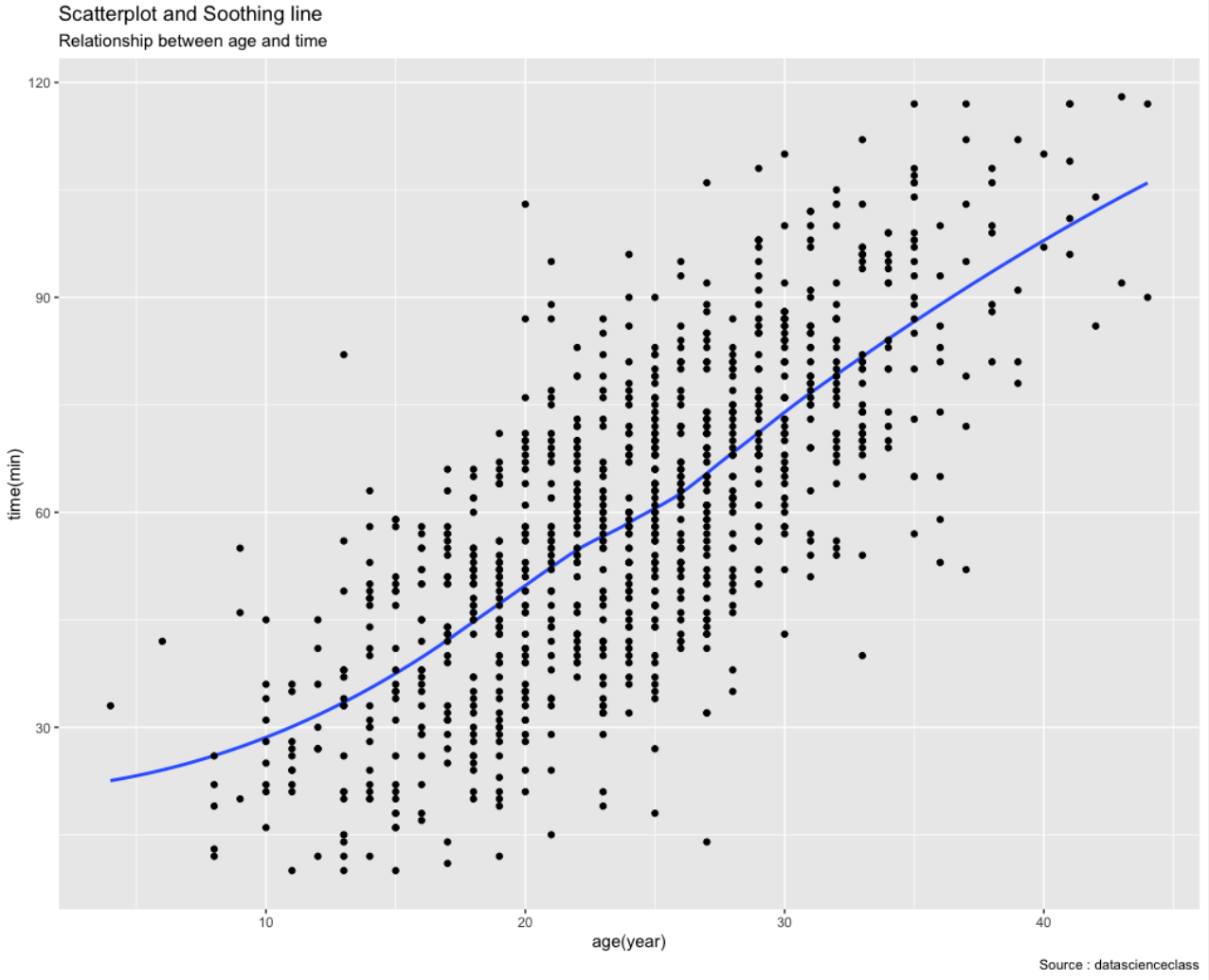
```
> cor(df_data$age,df_data$time,use='complete.obs',method='pearson') #0.7456772
[1] 0.7456772
```

The result is .... 0.7456772!!! It says there is a **strong positive linear relationship** between these two variables(age and time).

Through drawing *soothing line* and *scatter plot*, we can double-check it.

```
# Soothing line + scatter plot
ggplot(data=df_data,aes(x=age,y=time))+
  geom_smooth(method='auto',se=F)+
  geom_point(mapping = aes(x=age,y=time))+
  labs(subtitle = "Relationship between age and time",
       y="time(min)",
       x="age(year)",
       title="Scatterplot and Soothing line",
       caption = "Source : datascienceclass")
```

Scatter plot and Soothing line



Before end, I also checked p-value to check whether this relationship is credential.

```
# p-value
cor.test(df_data$age,df_data$time,use='complete.obs',method='pearson') #
p-value < 2.2e-16
```

P-value

```
> cor.test(df_data$age,df_data$time,use='complete.obs',method='pearson')

        Pearson's product-moment correlation

data:  df_data$age and df_data$time
t = 34.729, df = 963, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7162760 0.7724357
sample estimates:
      cor
0.7456772
```

Fortunately,P-value was smaller than 2.2e-16,which means this pearson coefficient is credential!!!

Main reasons why I choose pearson correlation coefficient are following:

- There are two quantitative variables
- We want to see if there is a linear relationship between these two variables
- Through data cleaning, I removed Na and outlier which largely affect pearson correlation coefficient.