Deliverable 1

Data Processing, Description, Validation and Profiling

Júlia Gasull i Claudia Sánchez

October 18, 2020

Contents

1 Data description

- Description http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
- Data Dictionary - SHL Trip Records -This data dictionary describes SHL trip data in visit http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml:

1.1 Variables

- VendorID
  - A code indicating the LPEP provider that provided the record.
  - Values:
    - 1= Creative Mobile Technologies, LLC
    - 2= VeriFone Inc.
- lpep_pickup_datetime
  - The date and time when the meter was engaged.
- lpep_dropoff_datetime
  - The date and time when the meter was disengaged.
- Passenger_count
  - The number of passengers in the vehicle.
  - This is a driver-entered value.
- Trip_distance
  - The elapsed trip distance in miles reported by the taximeter.
- Pickup_longitude
  - Longitude where the meter was engaged.
- Pickup_latitude
  - Latitude where the meter was engaged.
- RateCodeID
  - The final rate code in effect at the end of the trip.
  - Values:
    - 1=Standard rate

    - 2=JFK
    - 3=Newark
    - 4=Nassau or Westchester
    - 5=Negotiated fare
    - 6=Group ride

- Store_and_fwd_flag
  - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server:
  - Values
    - Y= store and forward trip
    - N= not a store and forward trip

- Dropoff_longitude
  - Longitude where the meter was timed off.

- Dropoff_latitude
  - Latitude where the meter was timed off.

- Payment_type
  - A numeric code signifying how the passenger paid for the trip.
  - Values:
    - 1= Credit card
    - 2= Cash
    - 3= No charge
    - 4= Dispute
- Fare_amount
  - The time-and-distance fare calculated by the meter.

- Extra
  - Miscellaneous extras and surcharges.
  - Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
- MTA_tax
  - $0.50 MTA tax that is automatically triggered based on the metered rate in use.

- Improvement_surcharge
  - $0.30 improvement surcharge assessed on hailed trips at the flag drop.
  - The improvement surcharge began being levied in 2015.

- Tip_amount
  - This field is automatically populated for credit card tips.
  - Cash tips are not included.

- Tolls_amount
  - Total amount of all tolls paid in trip.

- Total_amount
  - The total amount charged to passengers.
  - Does not include cash tips.

- Trip_type
  - A code indicating whether the trip was a street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver.
  - Values:
    - 1= Street-hail
    - 2= Dispatch

2 Load Required Packages for this deliverable

We load the necessary packages and set working directory

```
#setwd("~/Documents/uni/FIB-ADEI-LAB/deliverable1")
setwd("C:/Users/Claudia Sánchez/Desktop/FIB/TARDOR 2020-2021/ADEI/DELIVERABLE1/FIB-ADEI-LAB/deliverable1

# Load Required Packages
options(contrasts=c("contr.treatment","contr.treatment"))

requiredPackages <- c("missMDA","chemometrics","mvoutlier","effects","FactoMineR","car", "factoextra","F
missingPackages <- requiredPackages[!(requiredPackages %in% installed.packages()[,"Package"])]

if(length(missingPackages)) install.packages(missingPackages)
lapply(requiredPackages, require, character.only = TRUE)
```

2.1 Select a sample of 5000 records

From the proposed database, we need to select a sample of 5000 records randomly so we can start analyzing our data.

```
if(!is.null(dev.list())) dev.off()  # Clear plots
rm(list=ls())                       # Clean workspace
```

Data: green_tripdata_2016-01

```
#setwd("~/Documents/uni/FIB-ADEI-LAB/deliverable1")
#filepath<-"~/Documents/uni/FIB-ADEI-LAB/deliverable1"
setwd("C:/Users/Claudia Sánchez/Desktop/FIB/TARDOR 2020-2021/ADEI/DELIVERABLE1/FIB-ADEI-LAB/deliverable1
filepath<-"C:/Users/Claudia Sánchez/Desktop/FIB/TARDOR 2020-2021/ADEI/DELIVERABLE1/FIB-ADEI-LAB/delivera
df<-read.table(paste0(filepath,"/green_tripdata_2016-01.csv"),header=T, sep=",")
# dim(df)       # Displays the sample size
# names(df)     # Displays the names of the sample variables
# summary(df)
```

Select your 5000 register sample (random sample). Use birthday of 1 member of the group –> Júlia's one

```
set.seed(180998)
sam<-as.vector(sort(sample(1:nrow(df),5000)))
```

Verification and storage of the sample

```
head(df)
```

```
##   VendorID lpep_pickup_datetime Lpep_dropoff_datetime Store_and_fwd_flag
## 1        2  2016-01-01 00:29:24   2016-01-01 00:39:36                  N
## 2        2  2016-01-01 00:19:39   2016-01-01 00:39:18                  N
## 3        2  2016-01-01 00:19:33   2016-01-01 00:39:48                  N
## 4        2  2016-01-01 00:22:12   2016-01-01 00:38:32                  N
## 5        2  2016-01-01 00:24:01   2016-01-01 00:39:22                  N
## 6        2  2016-01-01 00:32:59   2016-01-01 00:39:35                  N
##   RateCodeID Pickup_longitude Pickup_latitude Dropoff_longitude
## 1          1        -73.92864        40.68061         -73.92428
## 2          1        -73.95267        40.72318         -73.92392
## 3          1        -73.97161        40.67611         -74.01316
## 4          1        -73.98950        40.66958         -74.00065
## 5          1        -73.96473        40.68285         -73.94072
## 6          1        -73.89114        40.74646         -73.86774
##   Dropoff_latitude Passenger_count Trip_distance Fare_amount Extra MTA_tax
## 1         40.69804               1          1.46         8.0   0.5     0.5
## 2         40.76138               1          3.56        15.5   0.5     0.5
## 3         40.64607               1          3.79        16.5   0.5     0.5
## 4         40.68903               1          3.01        13.5   0.5     0.5
## 5         40.66301               1          2.55        12.0   0.5     0.5
## 6         40.74211               1          1.37         7.0   0.5     0.5
##   Tip_amount Tolls_amount Ehail_fee improvement_surcharge Total_amount
## 1       1.86            0        NA                   0.3        11.16
## 2       0.00            0        NA                   0.3        16.80
## 3       4.45            0        NA                   0.3        22.25
## 4       0.00            0        NA                   0.3        14.80
```

```
## 5       0.00            0        NA                  0.3        13.30
## 6       0.00            0        NA                  0.3         8.30
##   Payment_type Trip_type
## 1            1         1
## 2            2         1
## 3            1         1
## 4            2         1
## 5            2         1
## 6            2         1
```

```
df<-df[sam,]
summary(df)
```

```
##     VendorID     lpep_pickup_datetime Lpep_dropoff_datetime Store_and_fwd_flag
##  Min.   :1.000   Length:5000          Length:5000           Length:5000
##  1st Qu.:2.000   Class :character     Class :character      Class :character
##  Median :2.000   Mode  :character     Mode  :character      Mode  :character
##  Mean   :1.788
##  3rd Qu.:2.000
##  Max.   :2.000
##    RateCodeID  Pickup_longitude Pickup_latitude Dropoff_longitude
##  Min.   :1.0   Min.   :-75.39   Min.   : 0.00   Min.   :-75.31
##  1st Qu.:1.0   1st Qu.:-73.96   1st Qu.:40.70   1st Qu.:-73.97
##  Median :1.0   Median :-73.95   Median :40.75   Median :-73.94
##  Mean   :1.1   Mean   :-73.89   Mean   :40.72   Mean   :-73.80
##  3rd Qu.:1.0   3rd Qu.:-73.92   3rd Qu.:40.80   3rd Qu.:-73.91
##  Max.   :5.0   Max.   :  0.00   Max.   :41.04   Max.   :  0.00
##  Dropoff_latitude Passenger_count Trip_distance     Fare_amount
##  Min.   : 0.00    Min.   :0.000   Min.   : 0.000   Min.   :-52.0
##  1st Qu.:40.70    1st Qu.:1.000   1st Qu.: 1.020   1st Qu.:  6.0
##  Median :40.75    Median :1.000   Median : 1.800   Median :  9.0
##  Mean   :40.67    Mean   :1.375   Mean   : 2.765   Mean   : 11.9
##  3rd Qu.:40.79    3rd Qu.:1.000   3rd Qu.: 3.420   3rd Qu.: 14.5
##  Max.   :41.18    Max.   :6.000   Max.   :52.790   Max.   :200.0
##      Extra            MTA_tax          Tip_amount      Tolls_amount
##  Min.   :-1.0000   Min.   :-0.5000   Min.   : 0.000   Min.   : 0.00000
##  1st Qu.: 0.0000   1st Qu.: 0.5000   1st Qu.: 0.000   1st Qu.: 0.00000
##  Median : 0.5000   Median : 0.5000   Median : 0.000   Median : 0.00000
##  Mean   : 0.3517   Mean   : 0.4857   Mean   : 1.217   Mean   : 0.08369
##  3rd Qu.: 0.5000   3rd Qu.: 0.5000   3rd Qu.: 2.000   3rd Qu.: 0.00000
##  Max.   : 1.0000   Max.   : 0.5000   Max.   :96.000   Max.   :18.04000
##  Ehail_fee      improvement_surcharge  Total_amount     Payment_type
##  Mode:logical   Min.   :-0.3000       Min.   :-52.80   Min.   :1.00
##  NA's:5000      1st Qu.: 0.3000       1st Qu.:  7.80   1st Qu.:1.00
##                 Median : 0.3000       Median : 11.16   Median :2.00
##                 Mean   : 0.2914       Mean   : 14.33   Mean   :1.52
##                 3rd Qu.: 0.3000       3rd Qu.: 17.16   3rd Qu.:2.00
##                 Max.   : 0.3000       Max.   :260.00   Max.   :4.00
##    Trip_type
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.023
##  3rd Qu.:1.000
##  Max.   :2.000
```

Save the image

```
save.image("Taxi5000_raw.RData")
```

2.2 Some useful functions

```
calcQ <- function(x) { # Function to calculate the different quartiles
  s.x <- summary(x)
  iqr<-s.x[5]-s.x[2]
```

```
  list(souti=s.x[2]-3*iqr, mouti=s.x[2]-1.5*iqr, min=s.x[1], q1=s.x[2], q2=s.x[3],
       q3=s.x[5], max=s.x[6], mouts=s.x[5]+1.5*iqr, souts=s.x[5]+3*iqr )
}

countNA <- function(x) { # Function to count the NA values
  mis_x <- NULL
  for (j in 1:ncol(x)) {mis_x[j] <- sum(is.na(x[,j])) }
  mis_x <- as.data.frame(mis_x)
  rownames(mis_x) <- names(x)
  mis_i <- rep(0,nrow(x))
  for (j in 1:ncol(x)) {mis_i <- mis_i + as.numeric(is.na(x[,j])) }
  list(mis_col=mis_x,mis_ind=mis_i)
}

countX <- function(x,X) { # Function to count a specific number of appearences
  n_x <- NULL
  for (j in 1:ncol(x)) {n_x[j] <- sum(x[,j]==X) }
  n_x <- as.data.frame(n_x)
  rownames(n_x) <- names(x)
  nx_i <- rep(0,nrow(x))
  for (j in 1:ncol(x)) {nx_i <- nx_i + as.numeric(x[,j]==X) }
  list(nx_col=n_x,nx_ind=nx_i)
}
```

---

3 Initiating missings, outliers and errors

Initialization of counts for missings, outliers and errors. All numerical variables have to be checked before

```
imis<-rep(0,nrow(df))  # rows - trips
jmis<-rep(0,2*ncol(df))  # columns - variables

mis1<-countNA(df)
imis<-mis1$mis_ind
# mis1$mis_col # Number of missings for the current set of variables

iouts<-rep(0,nrow(df))  # rows - trips
jouts<-rep(0,2*ncol(df))  # columns - variables

ierrs<-rep(0,nrow(df))  # rows - trips
jerrs<-rep(0,2*ncol(df))  # columns - variables
```

4 Univariate Descriptive Analysis

```
summary(df)
```

```
##     VendorID     lpep_pickup_datetime Lpep_dropoff_datetime Store_and_fwd_flag
##  Min.   :1.000   Length:5000          Length:5000           Length:5000
##  1st Qu.:2.000   Class :character     Class :character      Class :character
##  Median :2.000   Mode  :character     Mode  :character      Mode  :character
##  Mean   :1.788
##  3rd Qu.:2.000
##  Max.   :2.000
##    RateCodeID  Pickup_longitude Pickup_latitude Dropoff_longitude
##  Min.   :1.0   Min.   :-75.39   Min.   : 0.00   Min.   :-75.31
##  1st Qu.:1.0   1st Qu.:-73.96   1st Qu.:40.70   1st Qu.:-73.97
##  Median :1.0   Median :-73.95   Median :40.75   Median :-73.94
##  Mean   :1.1   Mean   :-73.89   Mean   :40.72   Mean   :-73.80
##  3rd Qu.:1.0   3rd Qu.:-73.92   3rd Qu.:40.80   3rd Qu.:-73.91
##  Max.   :5.0   Max.   :  0.00   Max.   :41.04   Max.   :  0.00
##  Dropoff_latitude Passenger_count Trip_distance     Fare_amount
```

```
##  Min.   : 0.00    Min.   :0.000   Min.   : 0.000   Min.   :-52.0
##  1st Qu.:40.70    1st Qu.:1.000   1st Qu.: 1.020   1st Qu.:  6.0
##  Median :40.75    Median :1.000   Median : 1.800   Median :  9.0
##  Mean   :40.67    Mean   :1.375   Mean   : 2.765   Mean   : 11.9
##  3rd Qu.:40.79    3rd Qu.:1.000   3rd Qu.: 3.420   3rd Qu.: 14.5
##  Max.   :41.18    Max.   :6.000   Max.   :52.790   Max.   :200.0
##      Extra            MTA_tax          Tip_amount      Tolls_amount
##  Min.   :-1.0000   Min.   :-0.5000   Min.   : 0.000   Min.   : 0.00000
##  1st Qu.: 0.0000   1st Qu.: 0.5000   1st Qu.: 0.000   1st Qu.: 0.00000
##  Median : 0.5000   Median : 0.5000   Median : 0.000   Median : 0.00000
##  Mean   : 0.3517   Mean   : 0.4857   Mean   : 1.217   Mean   : 0.08369
##  3rd Qu.: 0.5000   3rd Qu.: 0.5000   3rd Qu.: 2.000   3rd Qu.: 0.00000
##  Max.   : 1.0000   Max.   : 0.5000   Max.   :96.000   Max.   :18.04000
##  Ehail_fee      improvement_surcharge  Total_amount     Payment_type
##  Mode:logical   Min.   :-0.3000       Min.   :-52.80   Min.   :1.00
##  NA's:5000      1st Qu.: 0.3000       1st Qu.:  7.80   1st Qu.:1.00
##                 Median : 0.3000       Median : 11.16   Median :2.00
##                 Mean   : 0.2914       Mean   : 14.33   Mean   :1.52
##                 3rd Qu.: 0.3000       3rd Qu.: 17.16   3rd Qu.:2.00
##                 Max.   : 0.3000       Max.   :260.00   Max.   :4.00
##    Trip_type
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.023
##  3rd Qu.:1.000
##  Max.   :2.000
```

```
names(df)
```

```
##  [1] "VendorID"              "lpep_pickup_datetime"  "Lpep_dropoff_datetime"
##  [4] "Store_and_fwd_flag"    "RateCodeID"            "Pickup_longitude"
##  [7] "Pickup_latitude"       "Dropoff_longitude"     "Dropoff_latitude"
## [10] "Passenger_count"       "Trip_distance"         "Fare_amount"
## [13] "Extra"                 "MTA_tax"               "Tip_amount"
## [16] "Tolls_amount"          "Ehail_fee"             "improvement_surcharge"
## [19] "Total_amount"          "Payment_type"          "Trip_type"
```

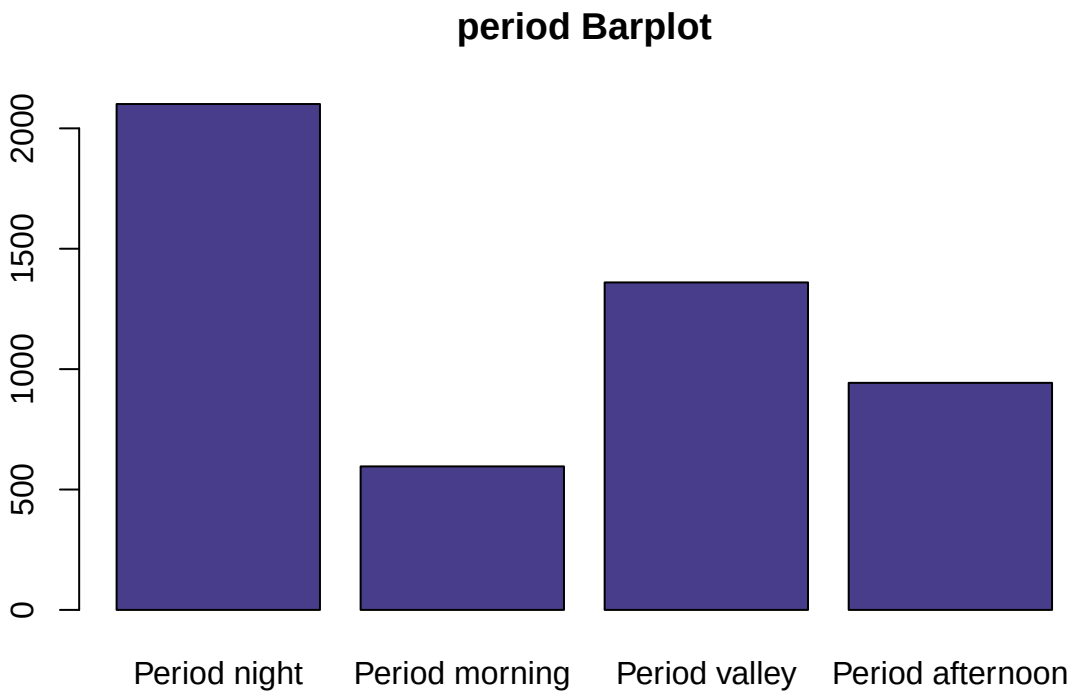
4.1 Qualitative Variables (Factors) / Categorical

Description: Original numeric variables corresponding to qualitative concepts have to be converted to factors. New factors grouping original levels will be considered very positively.

We need to do an analysis of all the variables to be able to identify missings, errors and outliers. We will also try to factorize each variable to make it easier to understand the sample.

4.1.1 New variable: Period

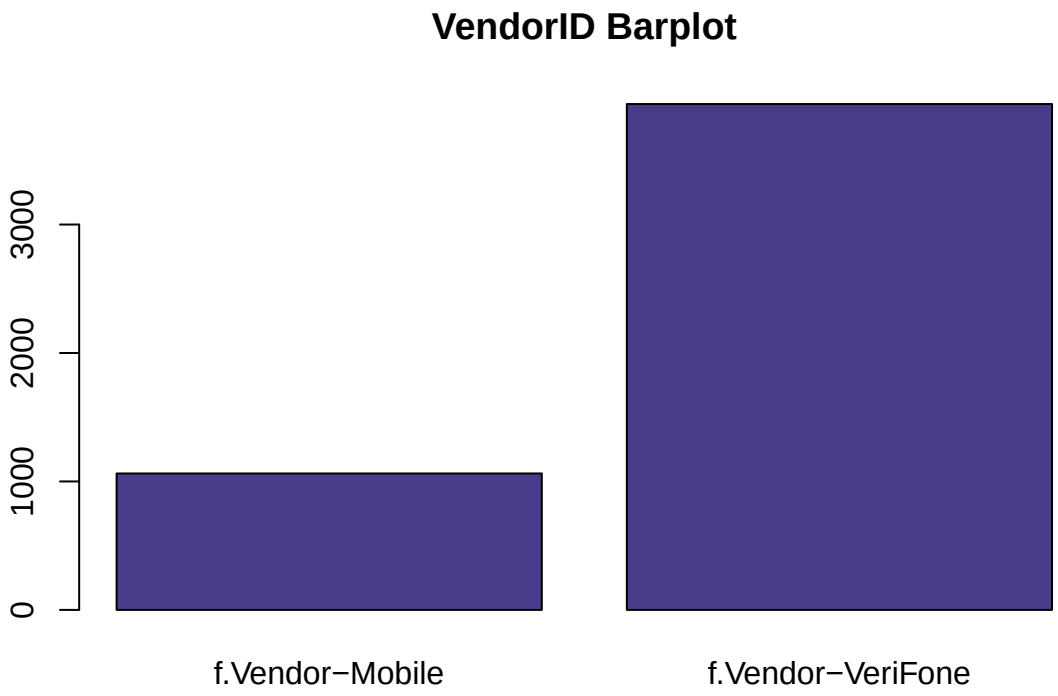
```
df$hour<-as.numeric(substr(strptime(df$lpep_pickup_datetime, "%Y-%m-%d %H:%M:%S"),12,13))
df$period<-1
df$period[df$hour>7]<-2
df$period[df$hour>10]<-3
df$period[df$hour>16]<-4
df$period[df$hour>19]<-1
df$period<-factor(df$period,labels=paste("Period",c("night","morning","valley","afternoon")))
barplot(summary(df$period),main="period Barplot",col = "DarkSlateBlue")
```



4.1.2 1. VendorID

This variable expresses the Creative Mobile Technologies, LLC as 1 and Verifone Inc as 2, so we create a factor to make it more readable. With the initial summary we see that this variable does not have any missing value, so we proceed to factor it.

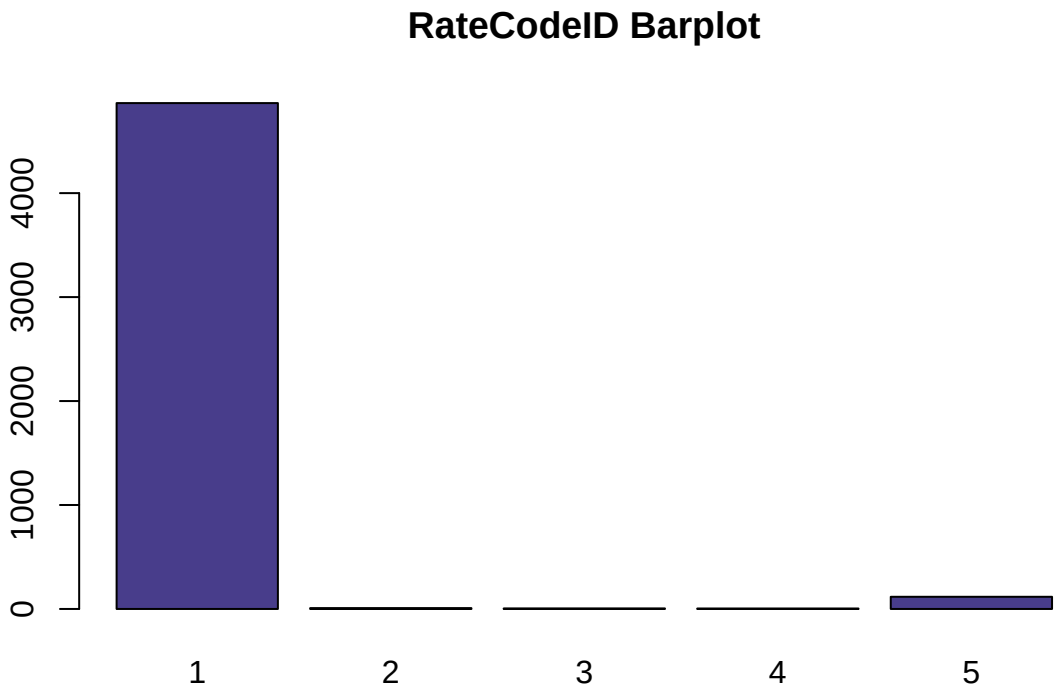
```
df$VendorID<-factor(df$VendorID,labels=c("Mobile","VeriFone"))
# nlevels(df$VendorID)
levels(df$VendorID)<-paste0("f.Vendor-",levels(df$VendorID))
# summary(df$VendorID)
barplot(summary(df$VendorID),main="VendorID Barplot",col = "DarkSlateBlue")
```



4.1.3 8. RateCodeID

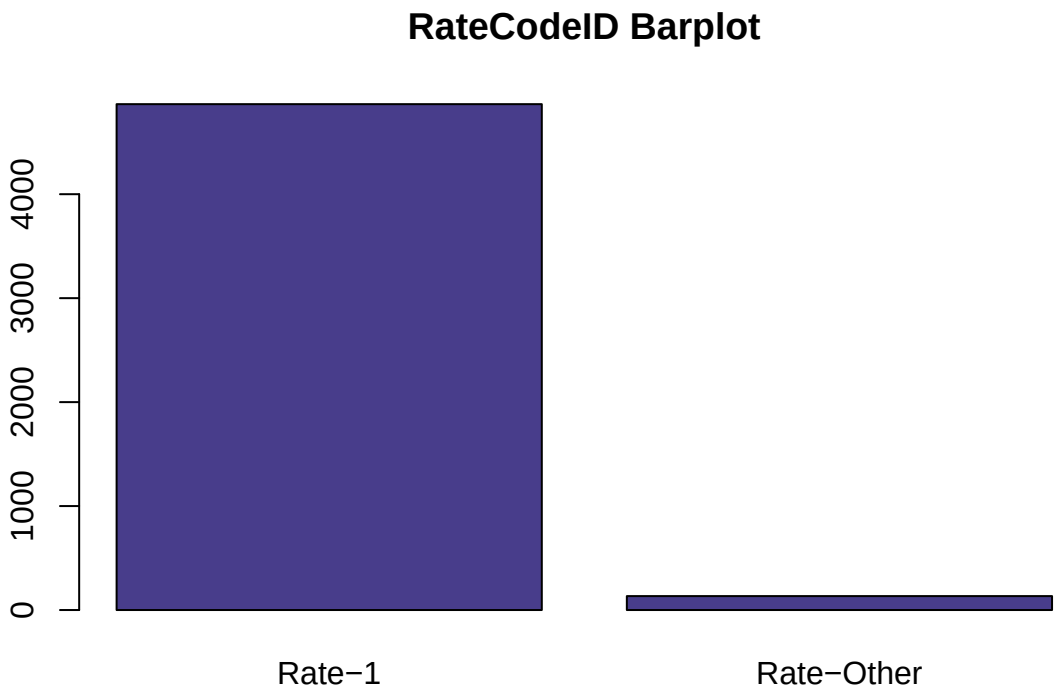
This variable expresses the different RateCodeIDs that we can have as numerical values, so we need to categorize them in order to be able to work with them.

```
# summary(df$RateCodeID)
df$RateCodeID<-factor(df$RateCodeID)
barplot(summary(df$RateCodeID),main="RateCodeID Barplot",col = "DarkSlateBlue")
```



We see that most samples are in RateCodeID = 1, which is what we are interested in. Therefore, we factorize and create only two groups, the one with RateCodeID = 1 and the rest.

```
df$RateCodeID[df$RateCodeID != 1] = 2
df$RateCodeID <- factor(df$RateCodeID, labels =c("Rate-1","Rate-Other"))
barplot(summary(df$RateCodeID),main="RateCodeID Barplot",col = "DarkSlateBlue")
```



Now is more balanced.

4.1.4 9. Store_and_fwd_flag

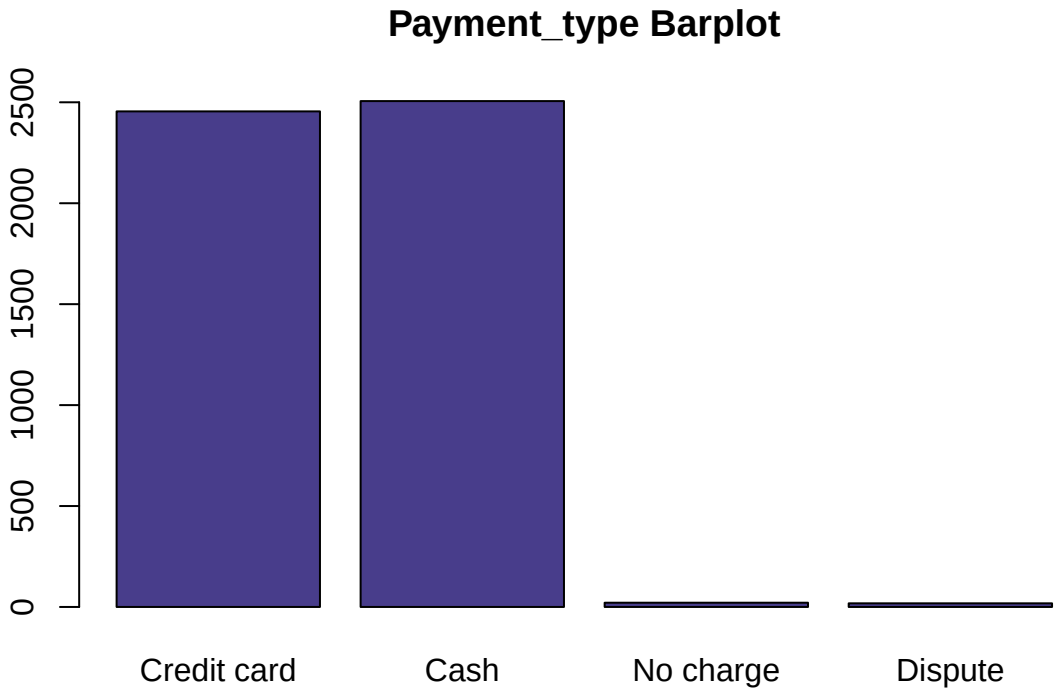
This is a categorical variable with the values Y and N, so we need to factor it.

```
# summary(df$Store_and_fwd_flag)
df$Store_and_fwd_flag<-factor(df$Store_and_fwd_flag)
barplot(summary(df$Store_and_fwd_flag),main="Store_and_fwd_flag Barplot",col = "DarkSlateBlue")
```

4.1.5 12. Payment_type

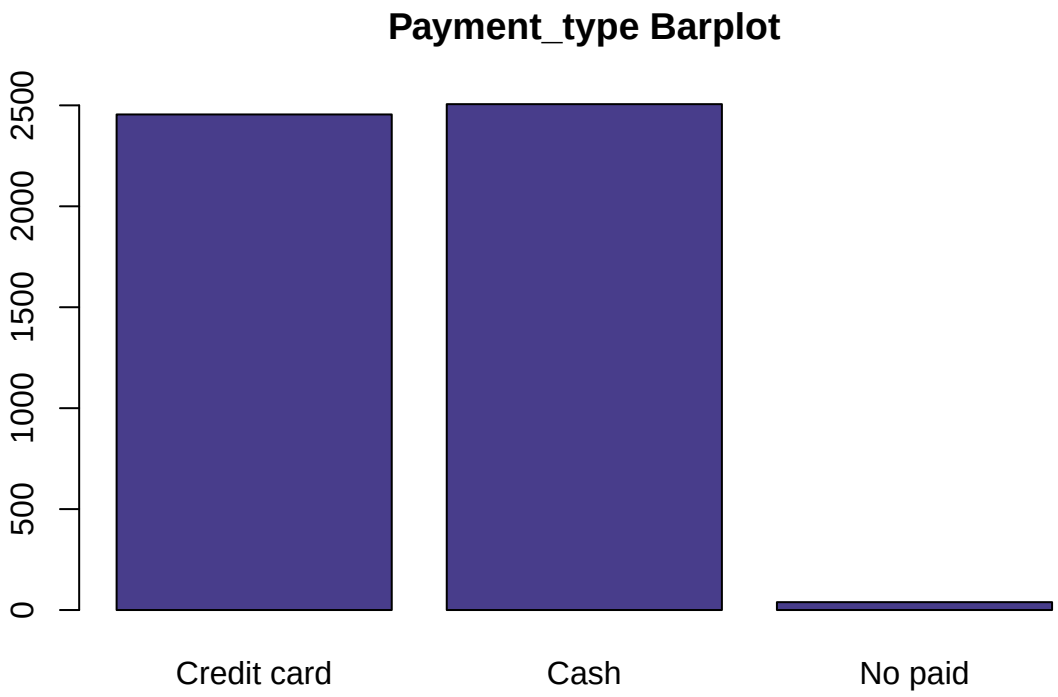
This variable is categorical but it is expressed as numerical, so we need to factor it in order to be able to work with it.

```
df$Payment_type<-factor(df$Payment_type,labels=c("Credit card","Cash","No charge","Dispute"))
# summary(df$Payment_type)
barplot(summary(df$Payment_type),main="Payment_type Barplot",col = "DarkSlateBlue")
```



As we can see, there are few values with “No charge” or “Dispute” category, so we decided to categorize it into a new category (“No paid”).

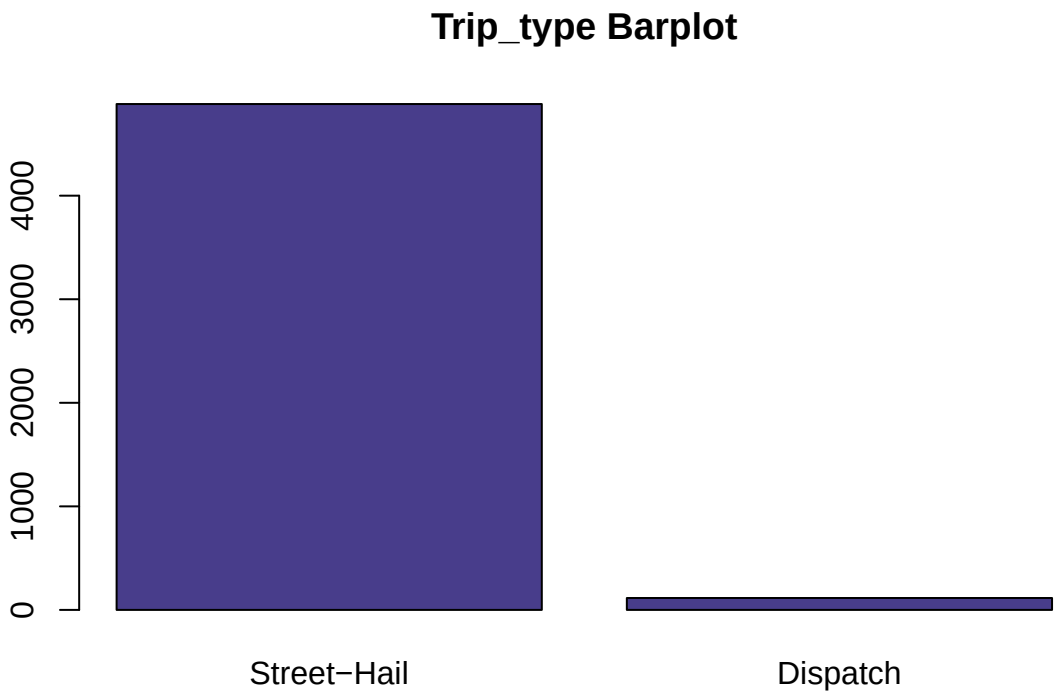
```
levels(df$Payment_type) <- c("Credit card","Cash","No paid","No paid")
# summary(df$Payment_type)
barplot(summary(df$Payment_type),main="Payment_type Barplot",col = "DarkSlateBlue")
```



4.1.6 21. Trip_type

This variable is categorical but it is expressed as numerical, so we need to factor it in order to be able to work with it.

```
df$Trip_type<-factor(df$Trip_type,labels=c("Street-Hail","Dispatch"))
barplot(summary(df$Trip_type),main="Trip_type Barplot",col = "DarkSlateBlue")
```




```
# summary(df$Trip_type)
```

4.2 Quantitative Variables

Description: Original numeric variables corresponding to real quantitative concepts are kept as numeric but additional factors should also be created as a discretization of each numeric variable.

We only keep the hours (variables 2 and 3) to be able to work with time slots in the future.

Create new variables derived from the original ones, as effective speed, travel time, hour of request, period of request, effective trip distance (in km)

4.2.1 New variables: Trip Length in km, Travel time un min and Effective speed

```
df$tlenkm<-df$Trip_distance*1.609344 # Miles to km
```

4.2.1.1 Trip length in km

```
df$traveltime<-(as.numeric(as.POSIXct(df$Lpep_dropoff_datetime)) - as.numeric(as.POSIXct(df$lpep_pickup_
```

4.2.1.2 Travel time in min

```
df$espeed<-(df$tlenkm/(df$traveltime))*60
```

4.2.1.3 Effective speed in km/h

```
sel<-which(is.na(df$espeed<=0)) #;length(sel)
#imis[sel]<-imis[sel]+1
#jmis[26]<-length(sel)
```

4.2.1.4 Missing data

```
summary(df$espeed)
```

4.2.1.5 Error detection

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   14.60   18.58   23.07   23.70 3881.74       2
```

```
sel<-which((df$espeed<=0)|(df$espeed=="Inf"))
ierrs[sel]<-ierrs[sel]+1
jerrs[26]<-length(sel)
# sel
```

Sel contains the rownames of the individuals with "0" as value for longitude

```
df[sel,"espeed"]<-NA
```

```
# summary(df$espeed)
calcQ(df$espeed)
```

4.2.1.6 Check outliers

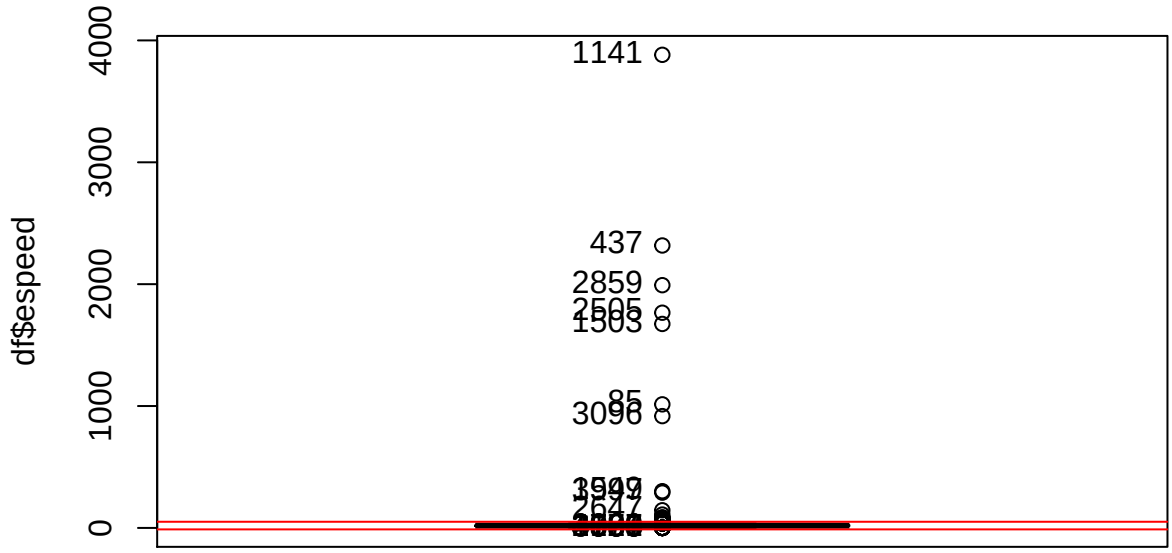
```
## $souti
##   1st Qu.
## -12.05656
##
## $mouti
##  1st Qu.
## 1.375907
##
## $min
##       Min.
## 0.03530885
##
## $q1
##  1st Qu.
## 14.80837
##
## $q2
##   Median
## 18.66159
##
## $q3
##  3rd Qu.
## 23.76335
##
## $max
##     Max.
## 3881.738
##
## $mouts
##  3rd Qu.
## 37.19582
##
## $souts
##  3rd Qu.
## 50.62828
```

```
Boxplot(df$espeed)
```

4.2.1.7 Outlier detection

```
##  [1] 4780 3001 3066 1936  120 3578 1767 4824 2685 3009 1141  437 2859 2505 1503
## [16]   85 3096 1549 3997 2647
```

```
var_out<-calcQ(df$espeed)
abline(h=var_out$souts,col="red")
abline(h=var_out$souti,col="red")
```




```
llout<-which((df$espeed<=3)|(df$espeed>80))
iouts[llout]<-iouts[llout]+1
jouts[26]<-length(llout)
df[llout,"espeed"]<-NA
```

4.2.2 2. lpep_pickup_datetime

We just keep the hours

```
df$pickup<-substr(strptime(df$lpep_pickup_datetime, "%Y-%m-%d %H:%M:%S"), 12, 13) # table(df$pickup)
```

4.2.3 3. lpep_dropoff_datetime

We just keep the hours

```
df$dropoff<-substr(strptime(df$Lpep_dropoff_datetime, "%Y-%m-%d %H:%M:%S"), 12, 13) # table(df$pickup)
```

4.2.4 4. Passenger_count

```
summary(df$Passenger_count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.375   1.000   6.000
```

We set the 0 as an error because it is not possible to have a trip without passengers

```
sel<-which(df$Passenger_count == 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[10]<-length(sel)
# sel
```

Sel contains the rownames of the individuals with “0” as value for passengers

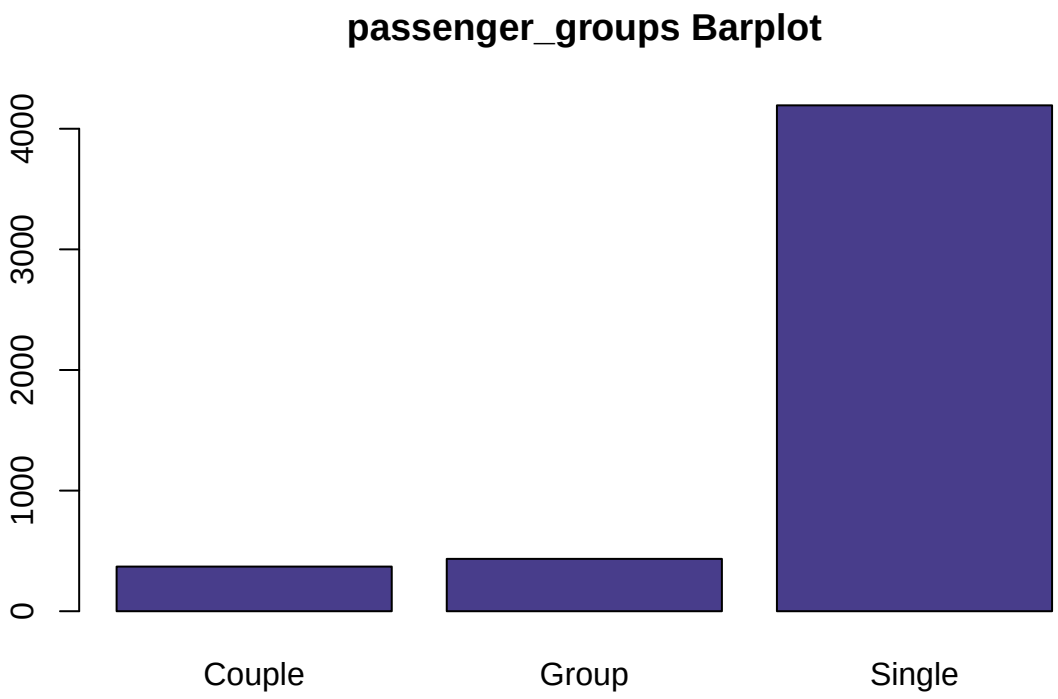
```
df[sel,"Passenger_count"]<-NA
```

We decided to create categorical for this variable so we categorize it for single passengers, couple and groups (3 or more)

```
df$passenger_groups[df$Passenger_count == 1] = "Single"
df$passenger_groups[df$Passenger_count == 2] = "Couple"
df$passenger_groups[df$Passenger_count >= 3] = "Group"
df$passenger_groups <- factor(df$passenger_groups)
```

We see the barplot in order to see the distribution of passenger per trip

```
barplot(table(df$passenger_groups),main="passenger_groups Barplot",col = "DarkSlateBlue")
```



4.2.5 5. Trip_distance

```
summary(df$Trip_distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.020   1.800   2.765   3.420  52.790
```

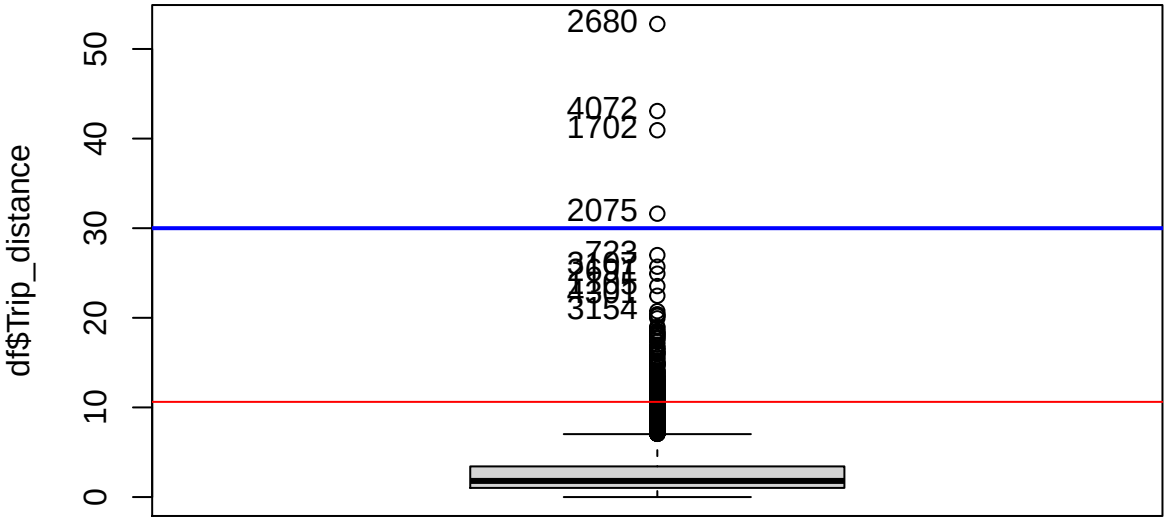
We see on the summary that there are not NA values, so we proceed to the outlier and error detection.

4.2.5.1 Outlier detection In order to evalute or data, we decide to set the maximum trip distance to 30, so we proceed to delete the outliers.

```
Boxplot(df$Trip_distance)
```

```
##  [1] 2680 4072 1702 2075  723 3107 2691 1105 4301 3154
```

```
var_out<-calcQ(df$Trip_distance)
abline(h=var_out$souts,col="red")
abline(h=var_out$souti,col="red")
abline(h=30,col="blue",lwd=2)
```




```
llout<-which(df$Trip_distance>30)
iouts[llout]<-iouts[llout]+1
# names(df)
jouts[11]<-length(llout)
```

4.2.5.2 Error detection We decide that an incorrect trip distance is the one with 0 miles or less. In order to be aware of this error we store it at ierrs, and jerrs. ierrs stores the number of errors in a row, and jerrs stores the total amount of errors in a variable.

```
sel<-which(df$Trip_distance <= 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[11]<-length(sel)
# sel
```

4.2.5.3 Errors and outliers Now, we set NA values in order to remove errors and outliersfrom the dataset

```
setNA<-which((df$Trip_distance<=0) | (df$Trip_distance > 30))
df[setNA,"Trip_distance"]<-NA
```

4.2.5.4 Caterogial variable for Trip_distance We are going to set a categorical variable for the Trip_distancerange. We decided to create 3 levels: “Short_dist”, “Medium_dist” and“Long_dist”. - Short_dist <= 2.5 - Medium_dist 2.5 < Trip_distance <= 5 - Long_dist > 5

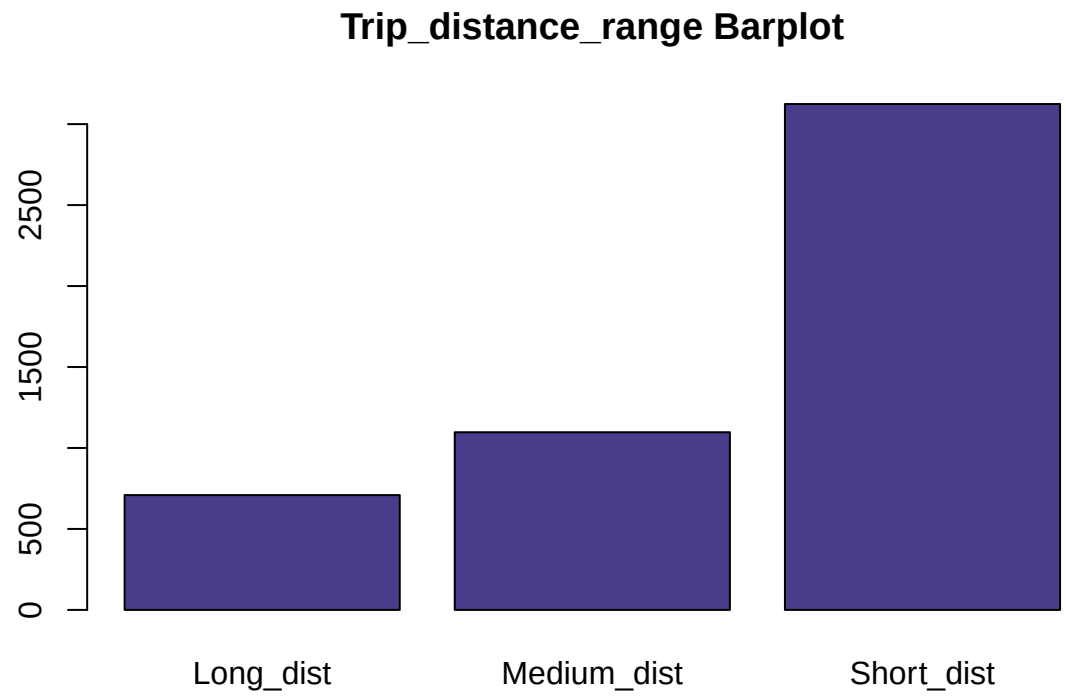
```
df$Trip_distance_range[df$Trip_distance <= 2.5] = "Short_dist"
df$Trip_distance_range[(df$Trip_distance > 2.5) & (df$Trip_distance <= 5)] = "Medium_dist"
df$Trip_distance_range[df$Trip_distance > 5] = "Long_dist"
# summary(df$Trip_distance_range)
```

We see, though, that it is not a factor yet, so we factor it.

```
df$Trip_distance_range <- factor(df$Trip_distance_range)
```

We see a barplot for the factor we created.

```
barplot(table(df$Trip_distance_range),main="Trip_distance_range Barplot",col = "DarkSlateBlue")
```



4.2.6 6. Pickup_longitude

We know that New York's longitude is -73.9385, so values that differ a lot from this value is an error or an outlier.

```
summary(df$Pickup_longitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -75.39  -73.96  -73.95  -73.89  -73.92    0.00
```

0.00 looks to be an error Seeing the individuals with this "0" value: df[which(df[,"Pickup_longitude"]==0),] it is a quantitive variable. Non-possible values will be recoded as errors, so will be transformed to NA.

```
sel<-which(df$Pickup_longitude == 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[6]<-length(sel)
# sel
```

Sel contains the rownames of the individuals with "0" as value for longitude.

```
df[sel,"Pickup_longitude"]<-NA
```

Non-possible values are replaced by NA, missing value symbol in R.

4.2.6.1 Which trips are not running in New-York? Consider if, at least, one of the pick-up and drop-off points belong to New-York area. if not, this trip is an "out-of-scope" individual and has to be eliminated of the basis. Nevertheless, you have to justify thiselimination and count how many individuals were in this situation. Look at that!! possibly, starting from the outliers... "0" is missing value, outliers can help to detect trips running outside of New York...

We are deleting trips from outside New York. This means we are not using longitudes bigger than -73.80 and smaller than -74.02.

```
llout <-which((df$Pickup_longitude < -74.02) | (df$Pickup_longitude > -73.80))
iouts[llout]<-iouts[llout]+1
# names(df)
jouts[6]<-length(llout)
```

Now that we have the outliers, we are setting them as NA

```
df[llout,"Pickup_longitude"]<-NA
```

4.2.7 7. Pickup_latitude

We know that New York's latitude is 40.6643, so values that differ a lot from this value is an error or an outlier.

```
summary(df$Pickup_latitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   40.70   40.75   40.72   40.80   41.04
```

0.00 looks to be an error. Seeing the individuals with this "0" value: df[which(df[,"Pickup_latitude"]==0),] it is a quantitive variable. non-possible values will be recoded as errors, so will be transformed to NA.

```
sel<-which(df$Pickup_latitude == 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[7]<-length(sel)
# sel
```

Sel contains the rownames of the individuals with "0" as value for longitude

```
df[sel,"Pickup_longitude"]<-NA
```

Non-possible values are replaced by NA, missing value symbol in R. We are deleting trips from outside New York. This means we are not using latitudes bigger than 40.54 and smallerthan 40.86

```
llout <-which((df$Pickup_latitude < 40.54) | (df$Pickup_latitude > 40.86))
iouts[llout]<-iouts[llout]+1
# names(df)
jouts[7]<-length(llout)
```

Now that we have the outliers, we are setting them as NA

```
df[llout,"Pickup_latitude"]<-NA
```

4.2.8 10. Dropoff_longitude

We know that New York's longitude is -73.9385, so values that differ a lot from this value is an error or an outlier.

```
summary(df$Dropoff_longitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -75.31  -73.97  -73.94  -73.80  -73.91    0.00
```

0.00 looks to be an error Seeing the individuals with this "0" value: df[which(df[,"Dropoff_longitude"]==0),] it is a quantitive variable.
Non-possible values will be recoded as errors, so will be transformed to NA.

```
sel<-which(df$Dropoff_longitude == 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[8]<-length(sel)
# sel
```

Sel contains the rownames of the individuals with "0" as value for longitude

```
df[sel,"Dropoff_longitude"]<-NA
```

Non-possible values are replaced by NA, missing value symbol in R. We are deleting trips from outside New York. This means we are not using longitudes bigger than -73.80 and smaller than -74.02.

```
llout <-which((df$Dropoff_longitude < -74.02) | (df$Dropoff_longitude > -73.80))
iouts[llout]<-iouts[llout]+1
# names(df)
jouts[8]<-length(llout)
# llout
```

Now that we have the outliers, we are setting them as NA

```
df[llout,"Dropoff_longitude"]<-NA
```

4.2.9 11. Dropoff_latitude

We know that New York's latitude is 40.6643, so values that differ a lot from this value is an error or an outlier.

```
summary(df$Dropoff_latitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   40.70   40.75   40.67   40.79   41.18
```

0.00 looks to be an error Seeing the individuals with this "0" value: df[which(df[,"Dropoff_latitude"]==0),] it is a quantitive variable. Non-possible values will be recoded as errors, so will be transformed to NA.

```
sel<-which(df$Dropoff_latitude == 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[8]<-length(sel)
# sel
```

Sel contains the rownames of the individuals with "0" as value for longitude

```
df[sel,"Dropoff_latitude"]<-NA
```

Non-possible values are replaced by NA, missing value symbol in R. We are deleting trips from outside New York. This means we are not using latitude bigger than 40.54 and smaller than 40.86

```
llout <-which((df$Dropoff_latitude < 40.54) | (df$Dropoff_latitude > 40.86))
iouts[llout]<-iouts[llout]+1
#names(df)
jouts[9]<-length(llout)
# llout
```

Now that we have the outliers, we are setting them as NA

```
df[llout,"Dropoff_latitude"]<-NA
```

4.2.10 13. Fare_amount

We know that the fare should be positive, as it is the price of the trip, so we'll treat as error those values. The next we'll do is decide the outliers.

```
summary(df$Fare_amount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -52.0     6.0     9.0    11.9    14.5   200.0
```

```
sel<-which(df$Fare_amount <= 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[12]<-length(sel)
# sel
```

```
df[sel,"Fare_amount"]<-NA
```

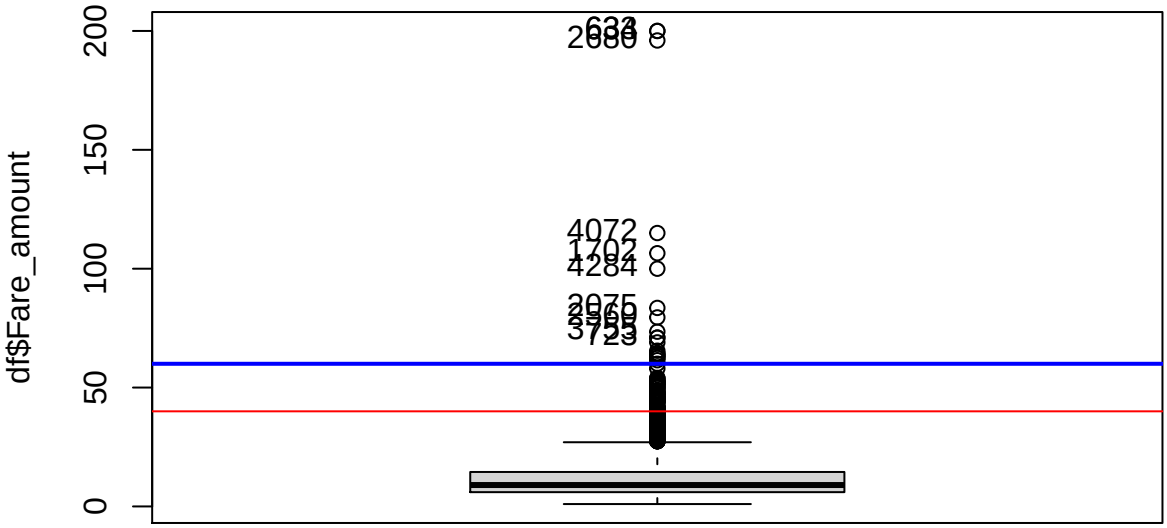
Non-possible values are replaced by NA, missing value symbol in R

```
Boxplot(df$Fare_amount)
```

4.2.10.1 Outlier detection

```
##  [1]  633  634 2680 4072 1702 4284 2075 2560 3755  723
```

```
var_out<-calcQ(df$Fare_amount)
abline(h=var_out$souts,col="red")
abline(h=var_out$souti,col="red")
abline(h=60,col="blue",lwd=2)
```



We decide to set outliers for fare amounts bigger than 60, because the majority of the values are concentrated between 0 and 60.

```
llout<-which(df$Fare_amount>60)
iouts[llout]<-iouts[llout]+1
jouts[12]<-length(llout)
df[llout,"Fare_amount"]<-NA
# llout
```

4.2.11 14. Extra

As this variable is price related, it cannot have negative values, so this individuals will be treated as errors.

```
summary(df$Extra)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.0000  0.0000  0.5000  0.3517  0.5000  1.0000
```

We execute table in order to see every different value in the sample

```
table(df$Extra)
```

```
##
##   -1 -0.5    0  0.5    1
##    2    5 2296 1868  829
```

As it is a price related variable, negative values should be treated as errors, and the other values are the ones defined for this variable, so there are not outliers.

```
# df[which(df[, "Extra"] < 0),]
sel<-which(df$Extra < 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[13]<-length(sel)
df[sel,"Extra"]<-NA
# sel
```

4.2.12 15. MTA_tax

This variable corresponds to a tax that must be charged in every trip and its cost is $0.50, so values different from this are errors, and we don't have to take into account outliers because after the errors detection all values should be the MTA_tax.

```
summary(df$MTA_tax)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.5000  0.5000  0.5000  0.4857  0.5000  0.5000
```

```
# df[which(df[, "MTA_tax"] != 0.50),]
```

Important note: We assume that when this tax is 0, it means there has been no payment. Therefore, we say that payment in these cases is equivalent to "no paid".

```
sel<-which(df$MTA_tax != 0.50)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[14]<-length(sel)
df[sel,"MTA_tax"]<-NA
# sel
```

If we execute a summary, we'll see that every value should be 0.5, so we proceed to categorize this variable.

```
summary(df$MTA_tax)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      0.5     0.5     0.5     0.5     0.5     0.5     133
```

```
df$MTA_tax <- factor(df$MTA_tax)
```

4.2.13 16. Improvement_surcharge

This variable corresponds to a charge that must be charged in every trip and its cost is $0.30, so values different from this are errors, and we don't have to take into account outliers because after the errors detection all values should be the Improvement surcharge.

```
summary(df$improvement_surcharge)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3000  0.3000  0.3000  0.2914  0.3000  0.3000
```

```
table(df$improvement_surcharge)
```

```
##
## -0.3    0  0.3
##   11  121 4868
```

We know that this surcharge was leived in 2015, so we need to check if the 0 values correspond to trips before this year. That is what we are going to do.

```
df$yearGt2015[(df$lpep_pickup_datetime >= "2015-01-01 00:00:00") & (df$improvement_surcharge == 0.3)] =
df$yearGt2015[(df$lpep_pickup_datetime < "2015-01-01 00:00:00") | (df$improvement_surcharge != 0.3)] = 0

table(df$yearGt2015)
```

```
##
##    0    1
##  132 4868
```

We see that the 0 individuals are errors, so we proceed to set them has NA and categorize this variable.

```
sel<-which(df$improvement_surcharge <= 0)
ierrs[sel]<-ierrs[sel]+1
# names(df)
jerrs[18]<-length(sel)
df[sel,"improvement_surcharge"]<-NA
# sel
df$improvement_surcharge <- factor(df$improvement_surcharge)
```

4.2.14 17. Ehail_fee

We don't take this into account because every value of our sample is NA.

```
summary(df$Ehail_fee)
```

```
##    Mode    NA's
## logical    5000
```

4.2.15 18. Tip_amount

As this is a price related variable, negative values should be considered as errors, and big tips should be considered as outliers. Also tip amounts bigger than 0 for individuals with payment_type = "Cash" should be considered as errors as well.

```
summary(df$Tip_amount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   1.217   2.000  96.000
```

We proceed to check if the 0 values are related with payment_type = "Credit card" and the passenger did not tip.

```
df$CashTips[(df$Tip_amount > 0) & (df$Payment_type == "Cash")] = 1
df$CashTips[(df$Payment_type == "Credit card")] = 0
table(df$CashTips)
```

```
##
##    0
## 2455
```

We see that we have correct data, so we proceed to create the binary factor TipIsGiven.

```
df$TipIsGiven[(df$Tip_amount > 0)] = "Yes"
df$TipIsGiven[(df$Tip_amount == 0)] = "No"
df$TipIsGiven <- factor(df$TipIsGiven)
summary(df$TipIsGiven)
```

```
##   No  Yes
## 2902 2098
```

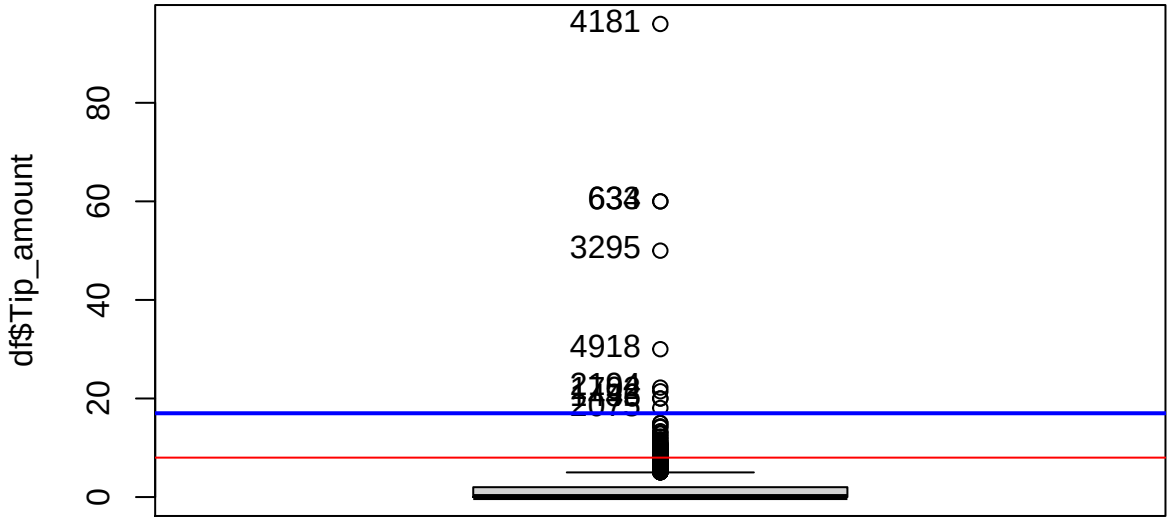
Now, we proceed to the outlier detection.

```
Boxplot(df$Tip_amount)
```

4.2.15.1 Outlier detection

```
##  [1] 4181  633  634 3295 4918 2194 1702   46 1433 2075
```

```
var_out<-calcQ(df$Tip_amount)
abline(h=var_out$souts,col="red")
abline(h=var_out$souti,col="red")
abline(h=17,col="blue",lwd=2)
```




```
llout<-which(df$Tip_amount>17)
iouts[llout]<-iouts[llout]+1
# names(df)
jouts[15]<-length(llout)
df[llout,"Tip_amount"]<-NA
# llout
```

4.2.16 19. Tolls_amount

As this is a price related variable, negative values should be considered as errors.

```
summary(df$Tolls_amount)
```

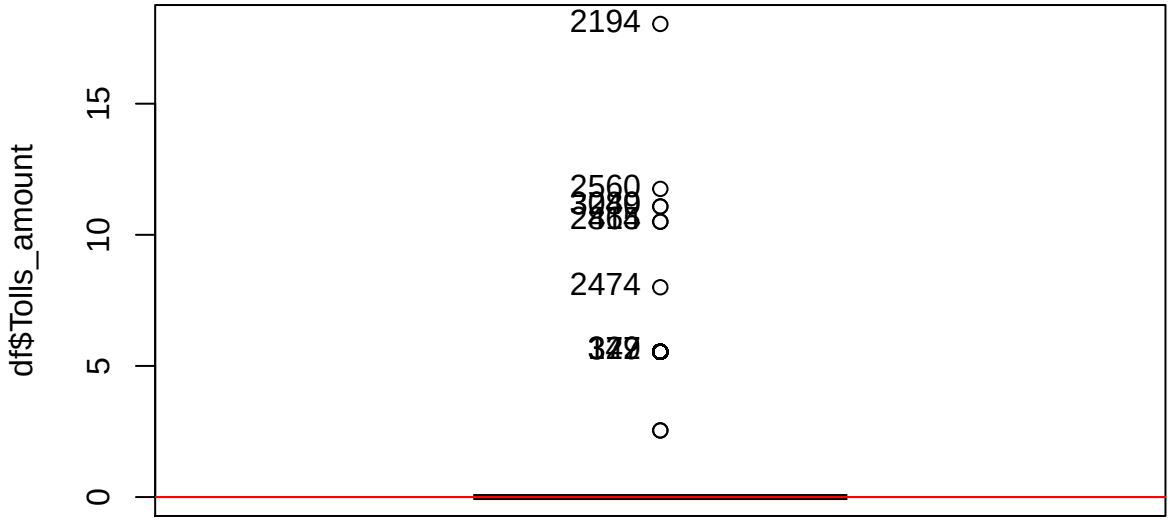
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.00000   0.00000   0.00000   0.08369   0.00000 18.04000
```

We see that there are not negative values, so we do not have errors. We proceed now to the outlier detection.

```
Boxplot(df$Tolls_amount)
```

```
##  [1] 2194 2560 3040 3289  415 2864 2474  122  347  379
```

```
var_out<-calcQ(df$Tolls_amount)
abline(h=var_out$souts,col="red")
abline(h=var_out$souti,col="red")
```




```
table(df$Tolls_amount)
```

```
##
##     0  2.54  5.54     8  10.5 11.08 11.75 18.04
##  4931     2    60     1     2     2     1     1
```

As we see in the boxplot and the table, the majority of the individuals are 0, so the values bigger than 0 will be outliers. After having the outliers, we proceed to categorize this variable.

```
llout<-which(df$Tolls_amount>0)
iouts[llout]<-iouts[llout]+1
# names(df)
jouts[16]<-length(llout)
df[llout,"Tolls_amount"]<-NA
# llout

df$Tolls_amount <- factor(df$Tolls_amount)
```

4.2.17 20. Total_amount

This is a price related variable, so negative values should be treated as errors. Also, we need to sum the "Fare_amount", "Extra","MTA_tax", "Improvement_surcharge", "Tip_amount" and the "Tolls_amount" in order to see if the Total_amount matches with this sum.

```
summary(df$Total_amount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -52.80    7.80   11.16   14.33   17.16  260.00
```

Negative values seem to be errors - 0 Total_amount is possible when Payment_type ==“No charge”

We proceed to check if total amount is correctsumming the other variables and checking negatives values:

```
df$Sum_total_amount = (df$Fare_amount + df$Extra + df$MTA_tax + df$improvement_surcharge + df$Tip_amount
```

```
## Warning in Ops.factor(df$Fare_amount + df$Extra, df$MTA_tax): '+' not meaningful
## for factors
```

```
## Warning in Ops.factor(df$Fare_amount + df$Extra + df$MTA_tax,
## df$improvement_surcharge): '+' not meaningful for factors
```

```
## Warning in Ops.factor(df$Fare_amount + df$Extra + df$MTA_tax +
## df$improvement_surcharge + : '+' not meaningful for factors
```

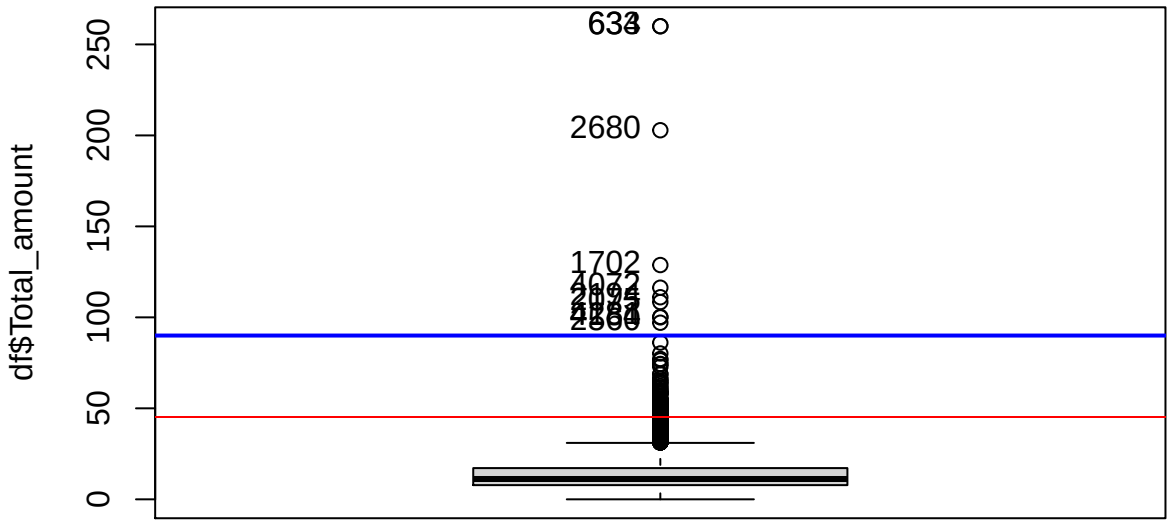
```
sel<-which((df$Total_amount != df$Sum_total_amount) | (df$Total_amount<0))
# names(df)
if (length(sel)>0) {
  ierrs[sel]<-ierrs[sel]+1
  jerrs[19]<-length(sel)
}
# sel
df[sel,"Total_amount"]<-NA
```

```
Boxplot(df$Total_amount)
```

4.2.17.1 Outlier detection

```
##  [1]  633  634 2680 1702 4072 2194 2075 4181 4284 2560
```

```
var_out<-calcQ(df$Total_amount)
abline(h=var_out$souts,col="red")
abline(h=var_out$souti,col="red")
abline(h=90,col="blue",lwd=2)
```




```
llout<-which(df$Total_amount>90)
iouts[llout]<-iouts[llout]+1
jouts[19]<-length(llout)
df[llout,"Total_amount"]<-NA
```

---

5 Data Quality Report

5.1 Per variable

Per each variable, we have to count the following:

- number of missing values

- number of errors (including inconsistencies)
- number of outliers
- rank variables according the sum of missing values (and errors).

5.1.1 Number of missing values of each variable

```
missings_ranking_sortlist <- sort.list(mis1$mis_col, decreasing = TRUE)
for (j in missings_ranking_sortlist) {
  print(paste(names(df)[j], " : ", mis1$mis_col$mis_x[j]))
}
```

```
## [1] "Ehail_fee  :  5000"
## [1] "VendorID  :  0"
## [1] "lpep_pickup_datetime  :  0"
## [1] "Lpep_dropoff_datetime  :  0"
## [1] "Store_and_fwd_flag  :  0"
## [1] "RateCodeID  :  0"
## [1] "Pickup_longitude  :  0"
## [1] "Pickup_latitude  :  0"
## [1] "Dropoff_longitude  :  0"
## [1] "Dropoff_latitude  :  0"
## [1] "Passenger_count  :  0"
## [1] "Trip_distance  :  0"
## [1] "Fare_amount  :  0"
## [1] "Extra  :  0"
## [1] "MTA_tax  :  0"
## [1] "Tip_amount  :  0"
## [1] "Tolls_amount  :  0"
## [1] "improvement_surcharge  :  0"
## [1] "Total_amount  :  0"
## [1] "Payment_type  :  0"
## [1] "Trip_type  :  0"
```

5.1.2 Number of errors per each variable

```
errors_ranking_sortlist <- sort.list(jerrs, decreasing = TRUE)
for (j in errors_ranking_sortlist) {
  if(!is.na(names(df)[j])) { print(paste(names(df)[j], " : ", jerrs[j])) }
}
```

```
## [1] "MTA_tax  :  133"
## [1] "improvement_surcharge  :  132"
## [1] "Trip_distance  :  66"
## [1] "espeed  :  64"
## [1] "Fare_amount  :  24"
## [1] "Total_amount  :  11"
## [1] "Dropoff_longitude  :  9"
## [1] "Extra  :  7"
## [1] "Pickup_longitude  :  3"
## [1] "Pickup_latitude  :  3"
## [1] "Passenger_count  :  2"
## [1] "VendorID  :  0"
## [1] "lpep_pickup_datetime  :  0"
## [1] "Lpep_dropoff_datetime  :  0"
## [1] "Store_and_fwd_flag  :  0"
## [1] "RateCodeID  :  0"
## [1] "Dropoff_latitude  :  0"
## [1] "Tip_amount  :  0"
## [1] "Tolls_amount  :  0"
## [1] "Ehail_fee  :  0"
## [1] "Payment_type  :  0"
## [1] "Trip_type  :  0"
## [1] "hour  :  0"
## [1] "period  :  0"
```

```
## [1] "tlenkm  :  0"
## [1] "traveltime  :  0"
## [1] "pickup  :  0"
## [1] "dropoff  :  0"
## [1] "passenger_groups  :  0"
## [1] "Trip_distance_range  :  0"
## [1] "yearGt2015  :  0"
## [1] "CashTips  :  0"
## [1] "TipIsGiven  :  0"
## [1] "Sum_total_amount  :  0"
```

5.1.3 Number of outliers per each variable

```
errors_ranking_sortlist <- sort.list(jouts, decreasing = TRUE)
for (j in errors_ranking_sortlist) {
  if(!is.na(names(df)[j])) print(paste(names(df)[j], " : ", jouts[j]))
}
```

```
## [1] "Dropoff_latitude  :  116"
## [1] "Dropoff_longitude  :  113"
## [1] "Pickup_latitude  :  87"
## [1] "Tolls_amount  :  69"
## [1] "espeed  :  48"
## [1] "Fare_amount  :  20"
## [1] "Pickup_longitude  :  19"
## [1] "Tip_amount  :  10"
## [1] "Total_amount  :  10"
## [1] "Trip_distance  :  4"
## [1] "VendorID  :  0"
## [1] "lpep_pickup_datetime  :  0"
## [1] "Lpep_dropoff_datetime  :  0"
## [1] "Store_and_fwd_flag  :  0"
## [1] "RateCodeID  :  0"
## [1] "Passenger_count  :  0"
## [1] "Extra  :  0"
## [1] "MTA_tax  :  0"
## [1] "Ehail_fee  :  0"
## [1] "improvement_surcharge  :  0"
## [1] "Payment_type  :  0"
## [1] "Trip_type  :  0"
## [1] "hour  :  0"
## [1] "period  :  0"
## [1] "tlenkm  :  0"
## [1] "traveltime  :  0"
## [1] "pickup  :  0"
## [1] "dropoff  :  0"
## [1] "passenger_groups  :  0"
## [1] "Trip_distance_range  :  0"
## [1] "yearGt2015  :  0"
## [1] "CashTips  :  0"
## [1] "TipIsGiven  :  0"
## [1] "Sum_total_amount  :  0"
```

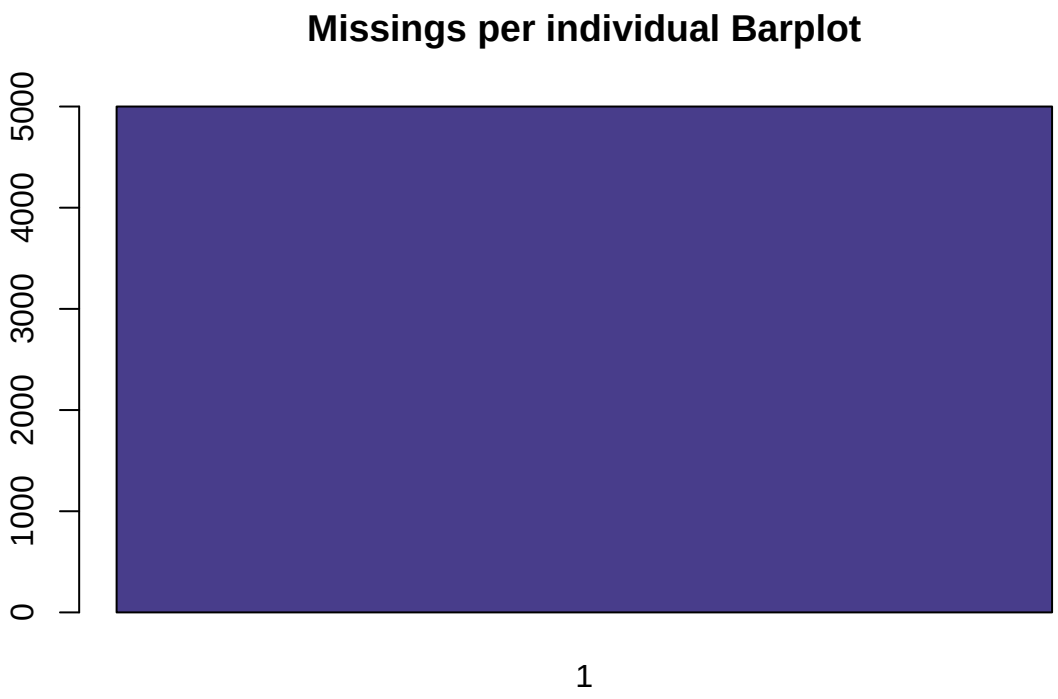
5.2 Per individual

Per each individuals, we have to count the following:

- number of missing values
- number of errors
- number of outliers
- identify individuals considered as multivariant outliers (we are leaving this step for the end of the document)

5.2.1 Number of missing values

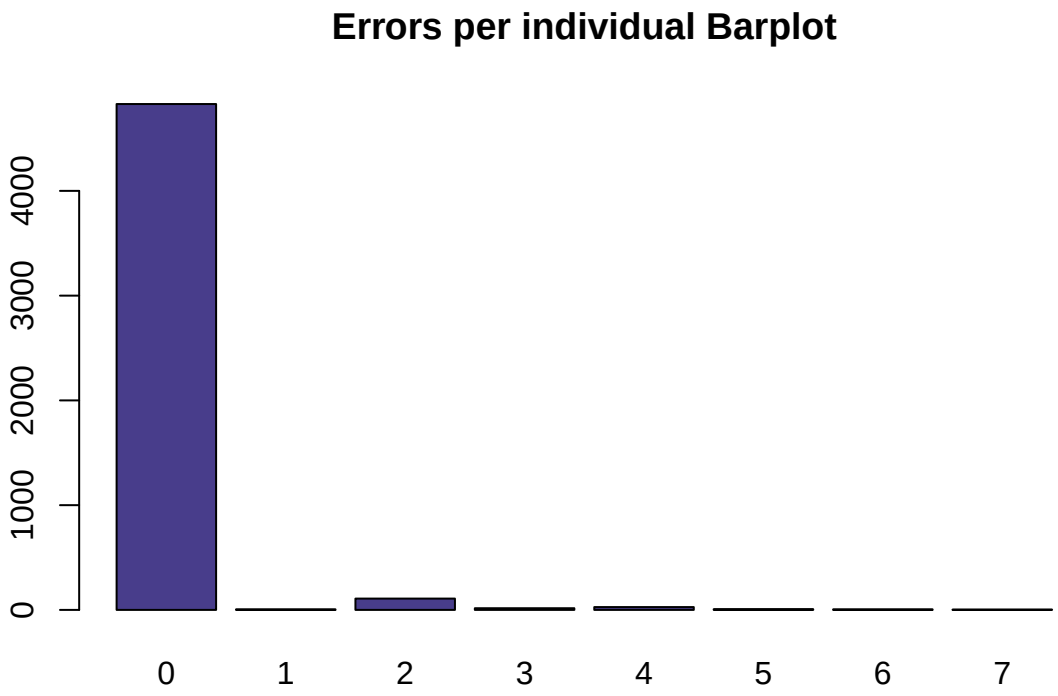
```
# table(imis)
barplot(table(imis),main="Missings per individual Barplot",col = "DarkSlateBlue")
```



5.2.2 Number of errors

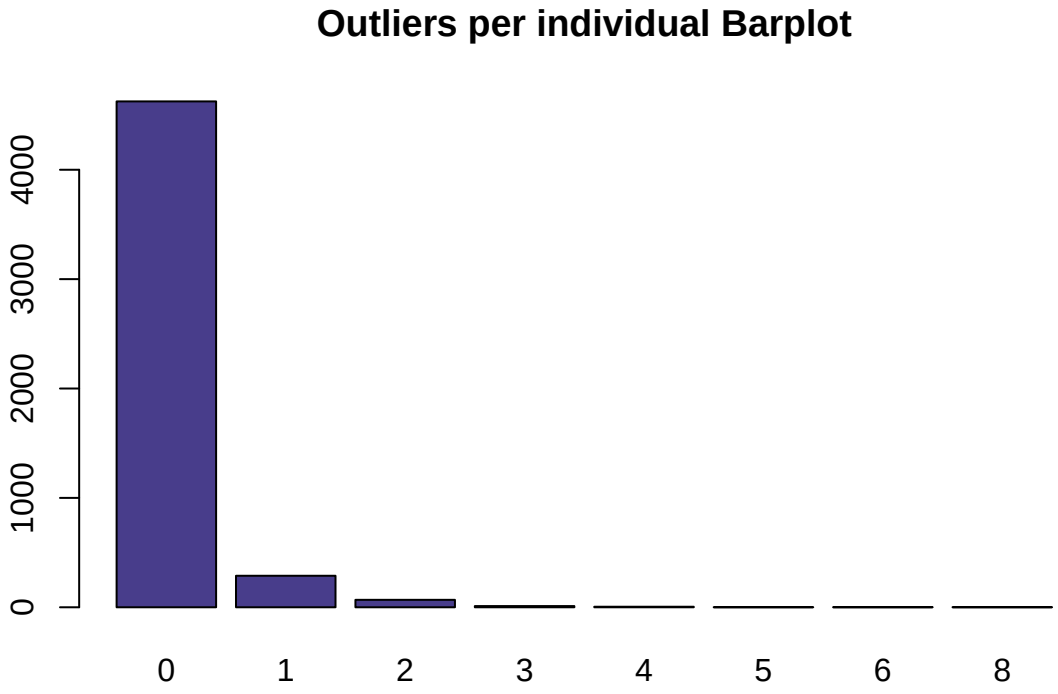
As we can see, most individuals have no mistakes. Those who do have errors, they tend to have more than one.

```
# table(ierrs)
barplot(table(ierrs),main="Errors per individual Barplot",col = "DarkSlateBlue")
```



5.2.3 Number of outliers

```
# table(iouts)
barplot(table(iouts),main="Outliers per individual Barplot",col = "DarkSlateBlue")
```



5.3 Create variable adding the total number missing values, outliers and errors

```
total_missings <- 0; total_outliers <- 0; total_errors <- 0;
for (m in imis) {total_missings <- total_missings + m}
for (o in iouts) {total_outliers <- total_outliers + o}
for (e in ierrs) {total_errors <- total_errors + e}
```

Now, let's print this variables:

```
total_missings
```

```
## [1] 5000
```

```
total_outliers
```

```
## [1] 496
```

```
total_errors
```

```
## [1] 463
```

---

6 Imputation

```
library(missMDA)
```

6.1 Numeric variables

We will now do the study by variables and try to impute the necessary observations.

```
# #vars_quantitatives<-names(df)[c(6:16, 18:19,2)]
vars_quantitatives<-names(df)[c(10:13,15,24:26)]
# we do not include MTA_tax (14), Tolls_amount(16) nor improvement_surcharge(18)
summary(df[,vars_quantitatives])
```

```
##  Passenger_count Trip_distance     Fare_amount        Extra
##  Min.   :1.000   Min.   : 0.010   Min.   : 1.00   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.: 1.050   1st Qu.: 6.00   1st Qu.:0.0000
##  Median :1.000   Median : 1.810   Median : 9.00   Median :0.5000
##  Mean   :1.376   Mean   : 2.770   Mean   :11.64   Mean   :0.3531
##  3rd Qu.:1.000   3rd Qu.: 3.458   3rd Qu.:14.50   3rd Qu.:0.5000
##  Max.   :6.000   Max.   :27.000   Max.   :60.00   Max.   :1.0000
##  NA's   :2       NA's   :70       NA's   :44      NA's   :7
##    Tip_amount        tlenkm         traveltime           espeed
##  Min.   : 0.00   Min.   : 0.000   Min.   :   0.000   Min.   : 3.239
##  1st Qu.: 0.00   1st Qu.: 1.642   1st Qu.:   5.917   1st Qu.:14.885
##  Median : 0.00   Median : 2.897   Median :   9.833   Median :18.689
##  Mean   : 1.14   Mean   : 4.450   Mean   :  20.059   Mean   :20.596
##  3rd Qu.: 2.00   3rd Qu.: 5.504   3rd Qu.:  16.246   3rd Qu.:23.759
##  Max.   :15.00   Max.   :84.957   Max.   :1438.183   Max.   :77.249
##  NA's   :10                                          NA's   :114
```

```
res.imputation<-imputePCA(df[,vars_quantitatives],ncp=5)
summary(res.imputation$completeObs)
```

```
##  Passenger_count Trip_distance      Fare_amount         Extra
##  Min.   :1.000   Min.   :-0.3415   Min.   :  1.00   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.: 1.0400   1st Qu.:  6.00   1st Qu.:0.0000
##  Median :1.000   Median : 1.8000   Median :  9.00   Median :0.5000
##  Mean   :1.376   Mean   : 2.7801   Mean   : 11.84   Mean   :0.3531
##  3rd Qu.:1.000   3rd Qu.: 3.4400   3rd Qu.: 14.50   3rd Qu.:0.5000
##  Max.   :6.000   Max.   :49.4812   Max.   :147.94   Max.   :1.0000
##    Tip_amount         tlenkm         traveltime           espeed
##  Min.   : 0.000   Min.   : 0.000   Min.   :   0.000   Min.   :-341.72
##  1st Qu.: 0.000   1st Qu.: 1.642   1st Qu.:   5.917   1st Qu.:  14.88
##  Median : 0.000   Median : 2.897   Median :   9.833   Median :  18.65
```

```
##  Mean   : 1.143   Mean   : 4.450   Mean   :  20.059   Mean   :  18.72
##  3rd Qu.: 2.000   3rd Qu.: 5.504   3rd Qu.:  16.246   3rd Qu.:  23.71
##  Max.   :15.000   Max.   :84.957   Max.   :1438.183   Max.   :  98.41
```

We proceed to impute all NAs in our numerical variables that are stored in: `res.imputation$completeObs`

```
#summary(res.imputation$completeObs)
df[,"Passenger_count"] <- res.imputation$completeObs[,"Passenger_count"]
df[,"Trip_distance"] <- res.imputation$completeObs[,"Trip_distance"]
df[,"Fare_amount"] <- res.imputation$completeObs[,"Fare_amount"]
df[,"Extra"] <- res.imputation$completeObs[,"Extra"]
df[,"Tip_amount"] <- res.imputation$completeObs[,"Tip_amount"]
df[,"tlenkm"] <- res.imputation$completeObs[,"tlenkm"]
df[,"traveltime"] <- res.imputation$completeObs[,"traveltime"]
df[,"espeed"] <- res.imputation$completeObs[,"espeed"]
```

6.2 Categorical variables / Factors

```
vars_categorical<-names(df)[c(1,4,5,20:21,23,29,30)]
summary(df[,vars_categorical])
```

```
##               VendorID    Store_and_fwd_flag      RateCodeID
##  f.Vendor-Mobile  :1062   N:4982             Rate-1    :4866
##  f.Vendor-VeriFone:3938   Y:  18             Rate-Other: 134
##
##
##       Payment_type        Trip_type                 period     passenger_groups
##  Credit card:2455   Street-Hail:4885   Period night    :2101   Couple: 370
##  Cash       :2506   Dispatch    : 115   Period morning  : 596   Group : 434
##  No paid    :  39                      Period valley   :1360   Single:4194
##                                        Period afternoon: 943   NA's  :   2
##   Trip_distance_range
##  Long_dist  : 709
##  Medium_dist:1097
##  Short_dist :3124
##  NA's       :  70
```

```
#nb <- estim_ncpMCA(df[, vars_categorical],ncp.max=25)
res.input<-imputeMCA(df[,vars_categorical],ncp=10)
summary(res.input$completeObs)
```

```
##               VendorID    Store_and_fwd_flag      RateCodeID
##  f.Vendor-Mobile  :1062   N:4982             Rate-1    :4866
##  f.Vendor-VeriFone:3938   Y:  18             Rate-Other: 134
##
##
##       Payment_type        Trip_type                 period     passenger_groups
##  Credit card:2455   Street-Hail:4885   Period night    :2101   Couple: 370
##  Cash       :2506   Dispatch    : 115   Period morning  : 596   Group : 434
##  No paid    :  39                      Period valley   :1360   Single:4196
##                                        Period afternoon: 943
##   Trip_distance_range
##  Long_dist  : 731
##  Medium_dist:1097
##  Short_dist :3172
##
```

We proceed to impute all NAs in our numerical variables that are stored in: `res.input$completeObs`

```
# summary(res.input$completeObs)
df[,"VendorID"] <- res.input$completeObs[,"VendorID"]
df[,"Store_and_fwd_flag"] <- res.input$completeObs[,"Store_and_fwd_flag"]
df[,"RateCodeID"] <- res.input$completeObs[,"RateCodeID"]
df[,"Payment_type"] <- res.input$completeObs[,"Payment_type"]
df[,"Trip_type"] <- res.input$completeObs[,"Trip_type"]
df[,"period"] <- res.input$completeObs[,"period"]
```

```
df[,"passenger_groups"] <- res.input$completeObs[,"passenger_groups"]
df[,"Trip_distance_range"] <- res.input$completeObs[,"Trip_distance_range"]
```

6.3 Result variables

```
vars_restult<-names(df)[c(19,33)]
```

6.4 Describe these variables, to which other variables exist higher associations

- compute the correlation with all other variables. Rank these variables according the correlation

```
library(mvoutlier)
library(FactoMineR)
vars_quantitatives<-names(df)[c(10:13,15,24:26)]
res <- cor(df[,vars_quantitatives])
round(res, 2)
```

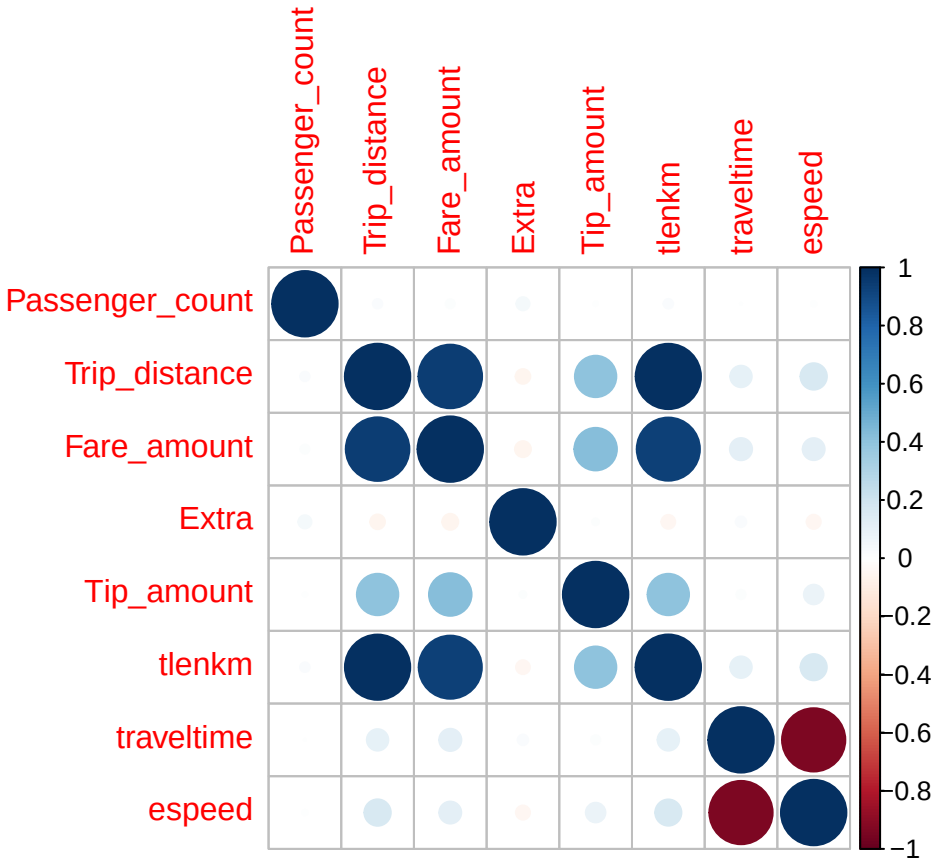
```
##                 Passenger_count Trip_distance Fare_amount Extra Tip_amount
## Passenger_count            1.00          0.02        0.02  0.04       0.00
## Trip_distance              0.02          1.00        0.94 -0.05       0.40
## Fare_amount                0.02          0.94        1.00 -0.06       0.42
## Extra                      0.04         -0.05       -0.06  1.00       0.01
## Tip_amount                 0.00          0.40        0.42  0.01       1.00
## tlenkm                     0.02          1.00        0.93 -0.04       0.40
## traveltime                 0.00          0.11        0.11  0.03       0.02
## espeed                     0.01          0.16        0.11 -0.05       0.09
##                 tlenkm traveltime espeed
## Passenger_count   0.02       0.00   0.01
## Trip_distance     1.00       0.11   0.16
## Fare_amount       0.93       0.11   0.11
## Extra            -0.04       0.03  -0.05
## Tip_amount        0.40       0.02   0.09
## tlenkm            1.00       0.11   0.16
## traveltime        0.11       1.00  -0.94
## espeed            0.16      -0.94   1.00
```

```
### Rank these variables by correlation
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.3
```

```
## corrplot 0.84 loaded
```

```
corrplot(res)
```



---

7 Profiling

7.1 Numeric target: Total_amount

Profiling is used to finish profiling our sample. We will now proceed to the profiling that asks us for our numeric target (Total_amount) and then we have to use the original variables and factors. In order to observe the relationship of our numerical target with the other variables we use the condes tool that provides us with information about the relationships between the indicated variables and the target.

```
library(FactoMineR)
library(mvoutlier)
# condes(df, which(names(df) == "Total_amount"))
```

Falta explicació de la correlació!!! + potser algun plot? (vars_resu <-names(df)[c(1,11)], aq.plot(df[,vars_resu])) Error: Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) : contrasts can be applied only to factors with 2 or more levels

7.2 Factor (Y.bin - TipIsGiven)

And now, we are profiling it with all other variables:

```
#df_catdes<-df[c(1:37)]
#catdes(df_catdes,37)
```

Falta explicació de la correlació!!! + potser algun plot? (vars_resu <-names(df)[c(1,11)], aq.plot(df[,vars_resu]))

7.2.1 Identify individuals considered as multivariant outliers

```
#library(chemometrics)
#multivariant_outliers <- Moutlier(df[, c(11:12, 19)], quantile = 0.995)

#!!!!! effective speed
#[1] ""               ""  "" ""
```

```
#[5] ""             ""       ""        ""
#[9] ""       ""        "Trip_distance"          "Fare_amount"
#[13] ""                   ""                "Tip_amount"             ""
#[17] ""              "" "Total_amount"           ""
#[21] ""              ""                  ""                  ""
#[25] ""    ""             ""                ""


# library(chemometrics)
# dis <- Moutlier(SwissLabor[,2:4], quantile = 0.995)
# dis$cutoff
# par(mfrow=c(1,1))
# plot(dis$md,dis$rd, type="n") text(dis$md,dis$rd,labels=rownames(SwissLabor[,2:4])) abline(h=qchisq(0.
# SwissLabor$mout<-0
# sel<-which(dis$rd>12)
# SwissLabor[sel,"mout"]<-1
```