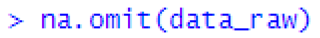
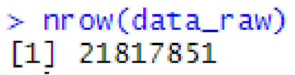
Task 1:

a. There are 21,817,851 rows in the dataset. Since the dataset is large, divide and conquer approach is suitable for this dataset. The dataset will be divided into smaller datasets and plots will be generated for each smaller dataset in order to identify noise points and outliers.
Invalid datapoints are those that are incomplete (with missing longitude or longitude values) and they should be omitted before the plots are generated. There is no datapoints with missing values in the dataset. The function na.omit() removes the datapoint with missing values. There seems to be no such datapoint no row was removed.

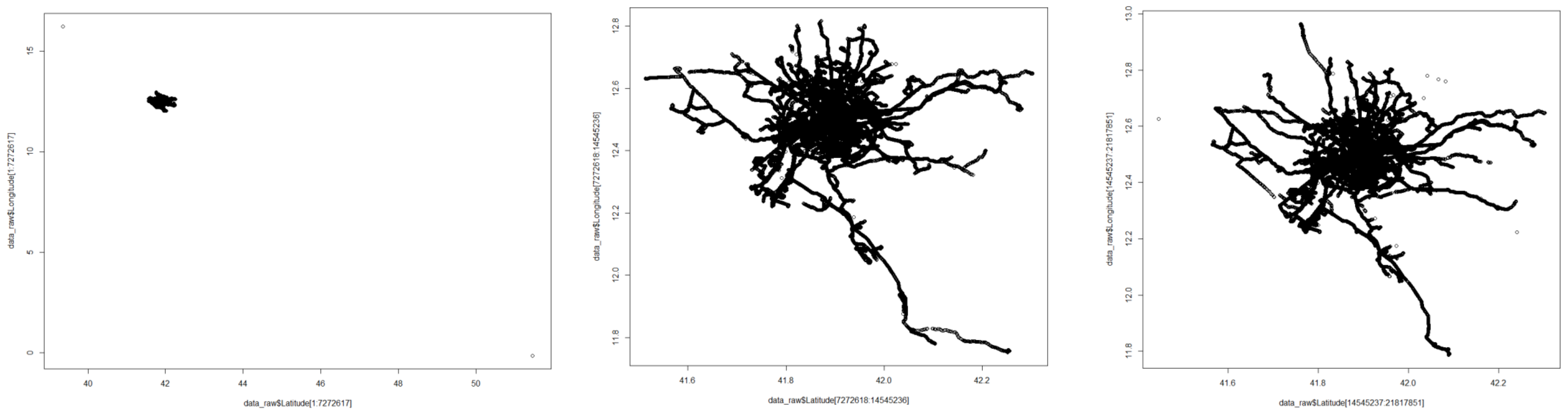


```
> na.omit(data_raw)
```



```
> nrow(data_raw)
[1] 21817851
```

The dataset is divided into three smaller datasets and plotted. The scatterplots below demonstrate the density of the location points using their longitudes and latitudes.



Figures 1.1: Density of the location points as the dataset is divided into thirds

Based on the three plots generated, there appear to be two possible noise points at around 16°E,39°N and at around 51°S,0.1°E as they appear completely separately from the area where the other datapoints appear densely. (They can be seen in the first plot). However, there is no certainty in using scatterplots to identify outliers and noise points in this case. For instance, datapoints that are densely distributed but appear to be in the water when plotted on a map are noise points because taxis cannot go underwater. Therefore, plotting the datapoints on the map of Italy or Europe would be a better alternative in this case. The dataset is divided into halves and plotted on the map of Italy. Noises are defined as the datapoints that are in the water areas, another city or country that are distinctively separated from the other datapoints. Outliers are the datapoints that are outside the border of Rome but still near Rome as it is possible that some taxi drivers travelled between cities or near the border areas.

The plots below demonstrate the datapoints plotted on a world map according to their longitudes and latitudes.

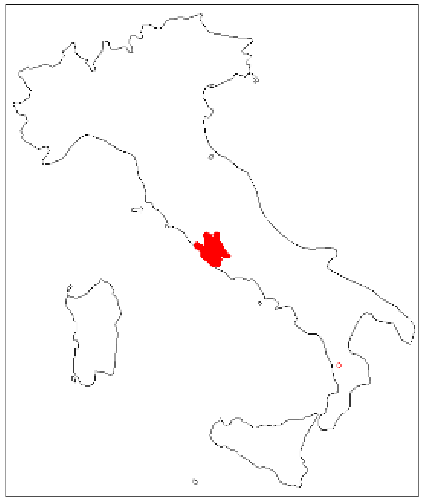
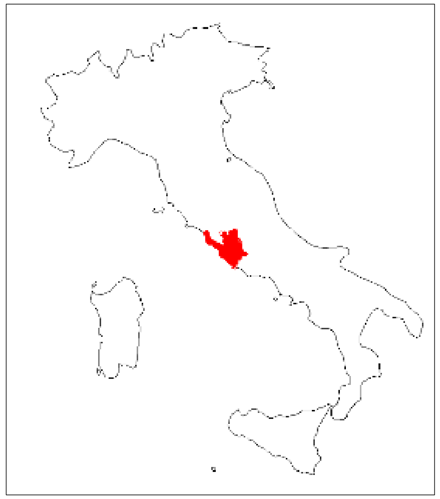



Figure 1.2: Datapoints plotted on Italy map

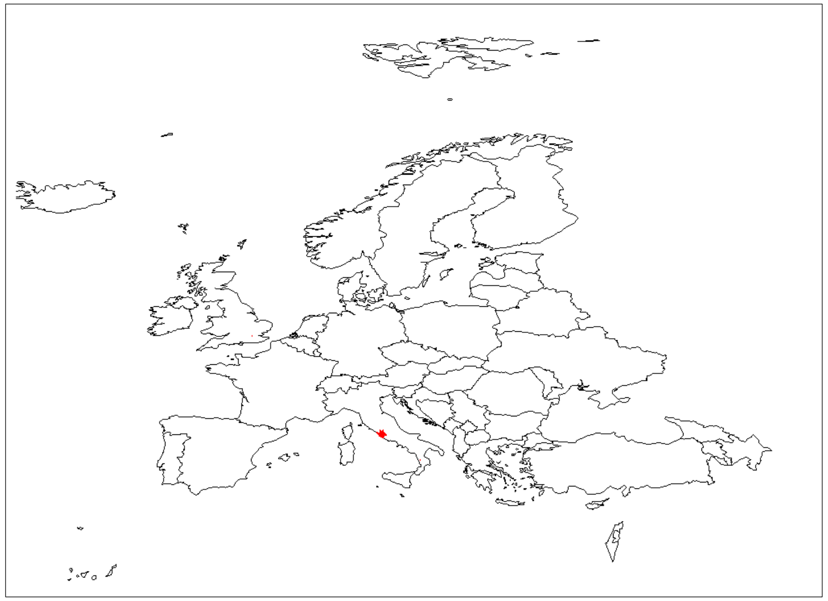
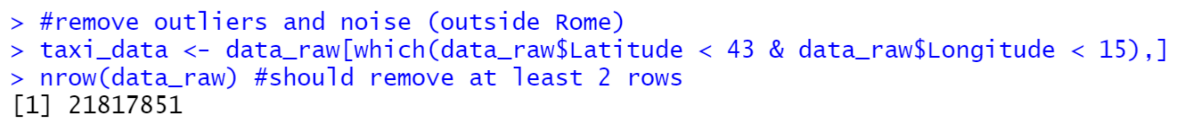


Figure 1.3: Datapoints plotted on Europe map

A noise point separate from the cluster can be seen in figure 1.2 in the area of Cosenzo (South of Italy) in as there no trail of the driver travelling from Rome to Cosenzo. The first map of Europe also shows a noise point in England. All four figures show no noise point in the water area. There might be outliers outside the border of Rome but still near Rome. These may be hard to find on a map.

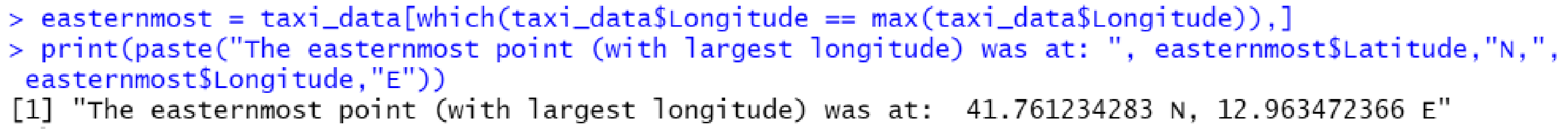
The outliers and noise points are those whose coordinates exceed a threshold (further than the extreme points of Rome) and will be removed using:



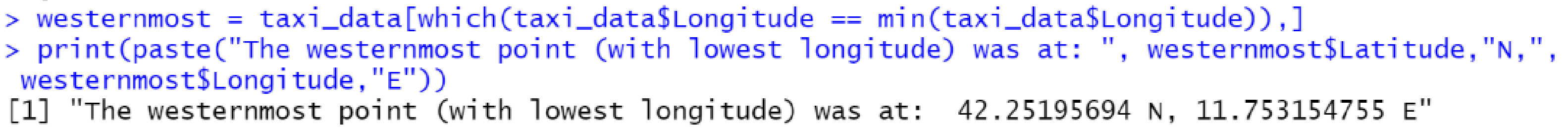
```
> #remove outliers and noise (outside Rome)
> taxi_data <- data_raw[which(data_raw$Latitude < 43 & data_raw$Longitude < 15),]
> nrow(data_raw) #should remove at least 2 rows
[1] 21817851
```

b. After outliers and noise points are removed:

#Identify the highest and lowest longitude

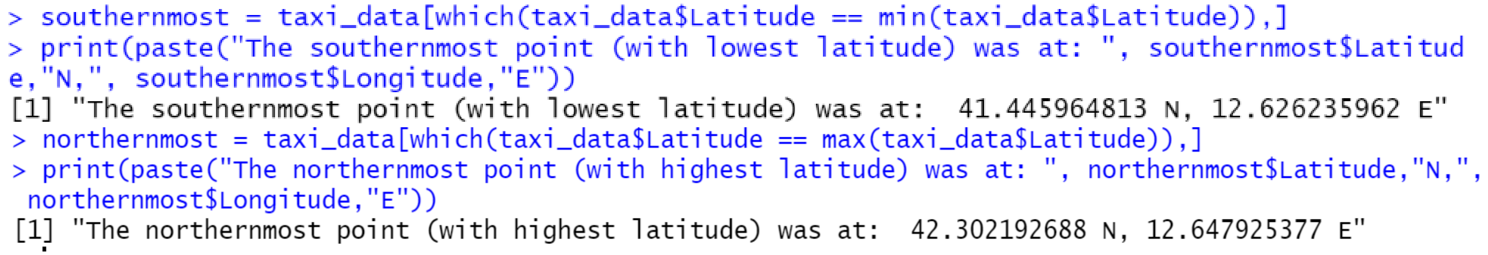


```
> easternmost = taxi_data[which(taxi_data$Longitude == max(taxi_data$Longitude)),]
> print(paste("The easternmost point (with largest longitude) was at: ", easternmost$Latitude,"N,",
 easternmost$Longitude,"E"))
[1] "The easternmost point (with largest longitude) was at:  41.761234283 N, 12.963472366 E"
```



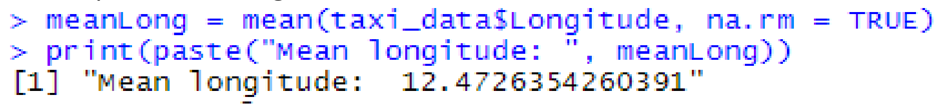
```
> westernmost = taxi_data[which(taxi_data$Longitude == min(taxi_data$Longitude)),]
> print(paste("The westernmost point (with lowest longitude) was at: ", westernmost$Latitude,"N,",
 westernmost$Longitude,"E"))
[1] "The westernmost point (with lowest longitude) was at:  42.25195694 N, 11.753154755 E"
```

#Identify the highest and lowest latitude

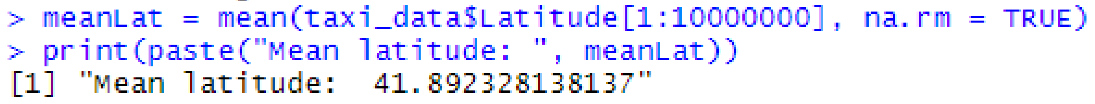


```
> southernmost = taxi_data[which(taxi_data$Latitude == min(taxi_data$Latitude)),]
> print(paste("The southernmost point (with lowest latitude) was at: ", southernmost$Latitud
e,"N,", southernmost$Longitude,"E"))
[1] "The southernmost point (with lowest latitude) was at:  41.445964813 N, 12.626235962 E"
> northernmost = taxi_data[which(taxi_data$Latitude == max(taxi_data$Latitude)),]
> print(paste("The northernmost point (with highest latitude) was at: ", northernmost$Latitude,"N,",
 northernmost$Longitude,"E"))
[1] "The northernmost point (with highest latitude) was at:  42.302192688 N, 12.647925377 E"
```

#Identify the mean longitude and latitude



```
> meanLong = mean(taxi_data$Longitude, na.rm = TRUE)
> print(paste("Mean longitude: ", meanLong))
[1] "Mean longitude:  12.4726354260391"
```



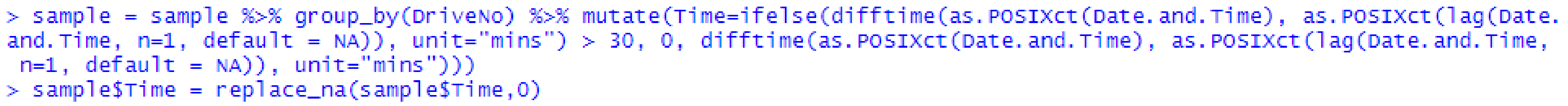
```
> meanLat = mean(taxi_data$Latitude[1:10000000], na.rm = TRUE)
> print(paste("Mean latitude: ", meanLat))
[1] "Mean latitude:  41.892328138137"
```

c. Approach to identify the most active, least active and the mean time driven:

A naïve approach would be using nested loops to calculate the time driven of each driver, then find the most, least and the mean time driven. However, using function of package “dplyr” allows us to compute the time driven faster using the similar logic. We will compute the active time of each driver and find the most active, least active and the mean active time of the entire dataset.

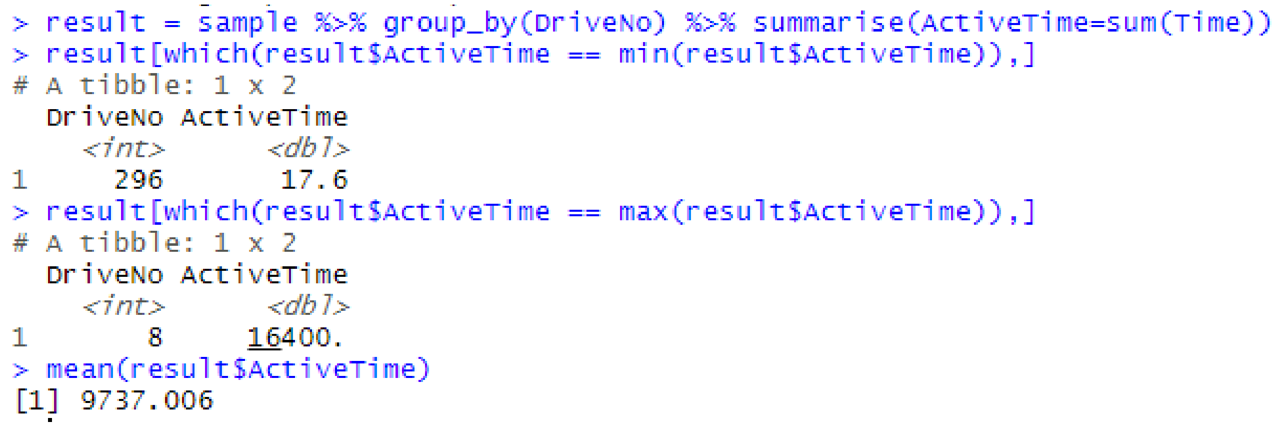
Firstly, using the functions in “dplyr”, we will group the rows by the Driver ID. Then, create a new column using “mutate” which calculate the time difference between two consecutively recorded moment in time with difftime in minutes. If the time difference computed is greater than 30 minutes, the result will be recorded as 0 minute of active time because this could mean that the drivers took a long break in between, and they were not working during those hours. Otherwise, the result will be stored in the newly created column called “Time”. The difftime function may result in a NA result; thus, replace_na is applied for the “Time” column to remove the NAs and replace them as 0s. Use group_by again to group the dataframe according to the Driver ID and calculate the total time each driver worked from the time differences archived above.

The figure below shows the R code as described above:



```
> sample = sample %>% group_by(DriveNo) %>% mutate(Time=ifelse(difftime(as.POSIXct(Date.and.Time), as.POSIXct(lag(Date.
and.Time, n=1, default = NA)), unit="mins") > 30, 0, difftime(as.POSIXct(Date.and.Time), as.POSIXct(lag(Date.and.Time,
 n=1, default = NA)), unit="mins")))
> sample$Time = replace_na(sample$Time,0)
```

After computing the total amount of active time of each driver by summing the time differences, we can extract the most, least and mean active time with the max, min and mean functions as follows:



```
> result = sample %>% group_by(DriveNo) %>% summarise(ActiveTime=sum(Time))
> result[which(result$ActiveTime == min(result$ActiveTime)),]
# A tibble: 1 x 2
  DriveNo ActiveTime
    <int>      <dbl>
1     296       17.6
> result[which(result$ActiveTime == max(result$ActiveTime)),]
# A tibble: 1 x 2
  DriveNo ActiveTime
    <int>      <dbl>
1       8     16400.
> mean(result$ActiveTime)
[1] 9737.006
```

The table below summarises the results:

	DriveNo	**Active Time**
Most active	296	17.6 minutes
Least active	8	16400 minutes
Mean	N/A	9737.006 minutes

d. Student ID: 6166994, Taxi ID: 268

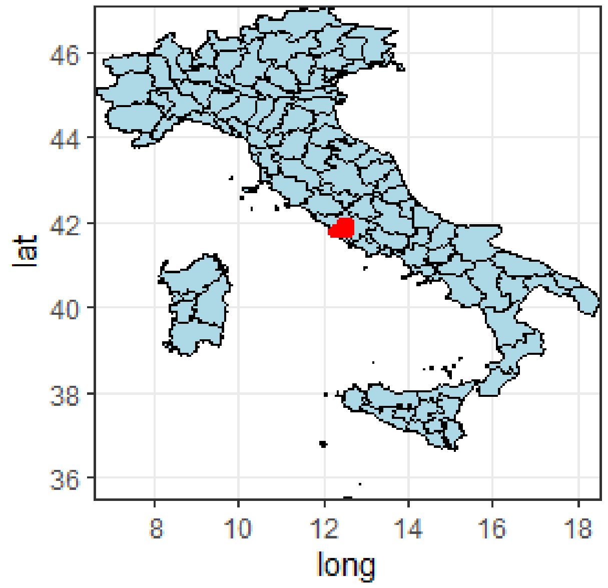


Figure 1.4: Locations of taxi driver 268

Driver 268 only travelled around within the centre of Rome. He did not travel inter-cities nor to other countries that are in the EU countries.

ii. Compare the min, max and mean location values of driver 268 with the global values

```
> min(studentTaxi$Longitude)
[1] 12.24792
> max(studentTaxi$Longitude)
[1] 12.60031
```

```
> min(studentTaxi$Latitude)
[1] 41.78515
> max(studentTaxi$Latitude)
[1] 41.99086
```

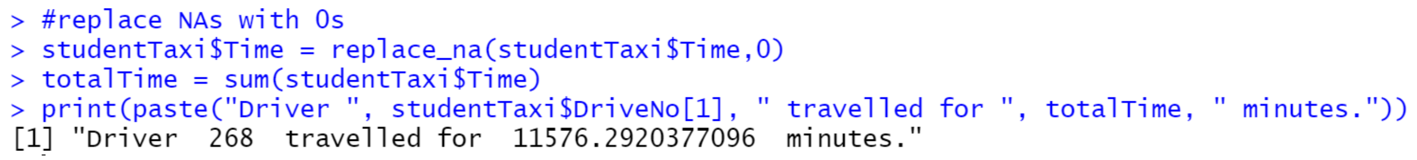
```
> mean(studentTaxi$Longitude)
[1] 12.46053
> mean(studentTaxi$Latitude)
[1] 41.88556
```

Figure 1.5: Min, max and mean location values of driver 268

The largest longitude, maximum latitude, mean longitude and the mean latitude of driver 268 are smaller than those of the global values. However, the lowest longitude and the lowest latitude of driver 268 are larger than the global values. This driver travelled further to the south of Rome than an average driver.

iii. The same approach in question 1c was used to compute the total time driven of driver 268. The result is as follows:

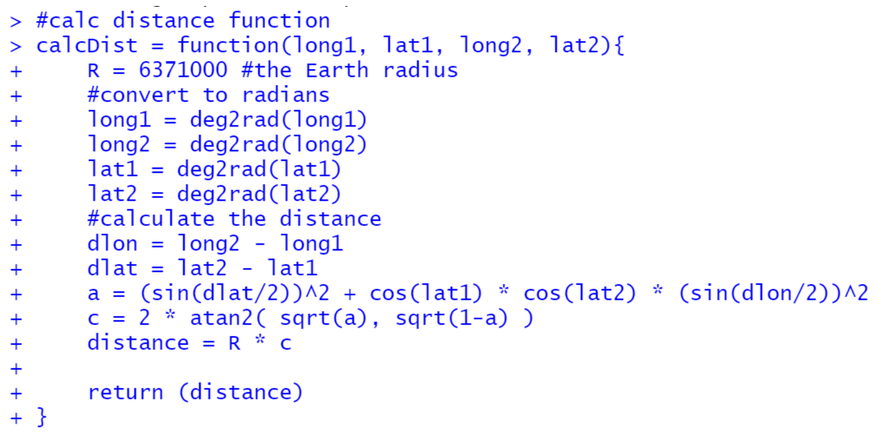
```
> #calc total distance travelled
> studentTaxi = taxi_data[which(taxi_data$DriveNo == 268),]
> studentTaxi = studentTaxi %>% mutate(Time=ifelse(difftime(as.POSIXct(Date.and.Time), as.POSIXct
(lag(Date.and.Time, n=1, default=NA)), unit="mins") > 30,0,difftime(as.POSIXct(Date.and.Time), a
s.POSIXct(lag(Date.and.Time, n=1, default=NA)), unit="mins")))
```



```
> #replace NAs with 0s
> studentTaxi$Time = replace_na(studentTaxi$Time,0)
> totalTime = sum(studentTaxi$Time)
> print(paste("Driver ", studentTaxi$DriveNo[1], " travelled for ", totalTime, " minutes."))
[1] "Driver  268  travelled for  11576.2920377096  minutes."
```

The total time driven of driver 268 is less than the maximum of the global values. However, it is higher than the minimum time and the mean time driven of the global values. This driver seemed to work hard. He even worked harder than an average driver whose working time is only 9737.006 minutes.

iv. The total travelled distance of driver 268 is calculated as follows:

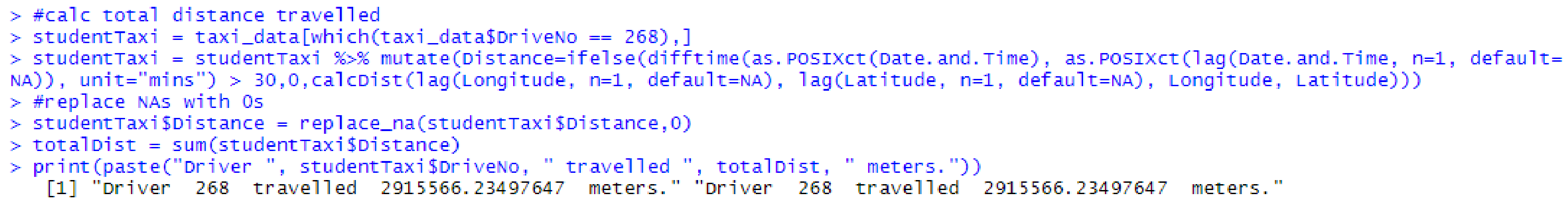


```
> #calc distance function
> calcDist = function(long1, lat1, long2, lat2){
+     R = 6371000 #the Earth radius
+     #convert to radians
+     long1 = deg2rad(long1)
+     long2 = deg2rad(long2)
+     lat1 = deg2rad(lat1)
+     lat2 = deg2rad(lat2)
+     #calculate the distance
+     dlon = long2 - long1
+     dlat = lat2 - lat1
+     a = (sin(dlat/2))^2 + cos(lat1) * cos(lat2) * (sin(dlon/2))^2
+     c = 2 * atan2( sqrt(a), sqrt(1-a) )
+     distance = R * c
+
+     return (distance)
+ }
```

The function calcDist was created to calculate the distance between two points on a globe based on their coordinates. It takes four parameters: the longitude and latitude of the two points, converts them to radians, computes and returns the distance.

Then, using the similar approach in question 1c. We will calculate the distance between two consecutive locations recorded of the driver using the created function above. We also calculate the time difference between two consecutive moments so as to ensure that we do not include the distance travelled when the drivers were not working. A threshold of 30 minutes is set. If the time difference exceeds this threshold, it means that the drivers were not working and the distance travelled will be recorded as 0 meter. The results will be stored in a new column called Distance. In order to calculate the total distance travelled by the driver, sum the previously computed distances.

Below is the R code for computing the total distance travelled. Driver 268 travelled around 2915566.23 meters.



```
> #calc total distance travelled
> studentTaxi = taxi_data[which(taxi_data$DriveNo == 268),]
> studentTaxi = studentTaxi %>% mutate(Distance=ifelse(difftime(as.POSIXct(Date.and.Time), as.POSIXct(lag(Date.and.Time, n=1, default=
NA)), unit="mins") > 30,0,calcDist(lag(Longitude, n=1, default=NA), lag(Latitude, n=1, default=NA), Longitude, Latitude)))
> #replace NAs with 0s
> studentTaxi$Distance = replace_na(studentTaxi$Distance,0)
> totalDist = sum(studentTaxi$Distance)
> print(paste("Driver ", studentTaxi$DriveNo, " travelled ", totalDist, " meters."))
   [1] "Driver  268  travelled  2915566.23497647  meters." "Driver  268  travelled  2915566.23497647  meters."
```

Task 2:

Question 1: This step mainly focuses on identifying the 5 most interesting variables through trial and error by investigating different possible clusters generated by SOMs. The key is to look for the attributes that make clear divisions in the dataset (i.e. possible clusters can be easily seen). This means that these attributes can be used for classification.

First, I trained a 35x35 SOM for the entire dataset with all variables in 2000 iterations. Five most interesting attributes noticed include Functionary [1], Rebalanced an overdrawn account [2], FICO score [3], Credit refused in the past [6], and Self-employed [9]. Five different heatmaps are plotted to demonstrate their distributions. There is a relatively clear distinction between the nodes according to each variable. The majority of the nodes in each heatmap below are either low or high and are well-grouped.

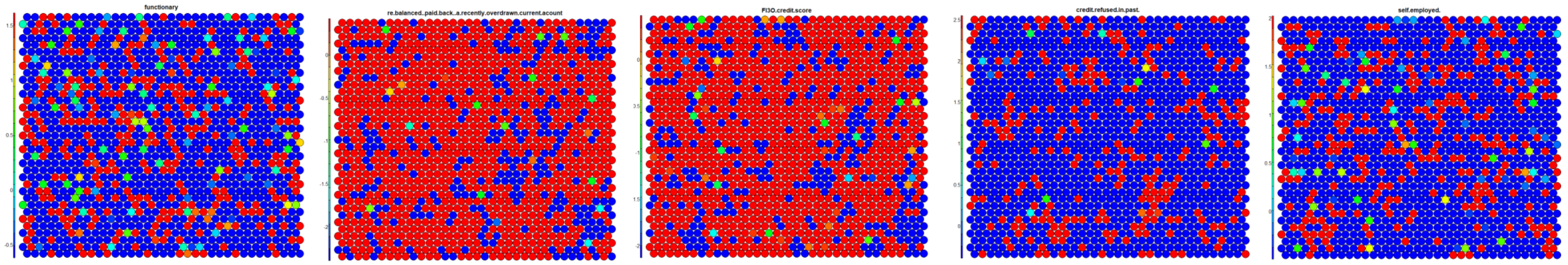


Figure 2.1: Heatmaps for five interesting variables with clusters identified by SOM
([1] Functionary, [2] Rebalanced, [3] FICO score, [4] Credit refused, [5] Self-employed)

These five attributes are then selected to train another 10x10 SOM in 1000 iterations. Again, there is a clear distinction between the nodes for each attribute.

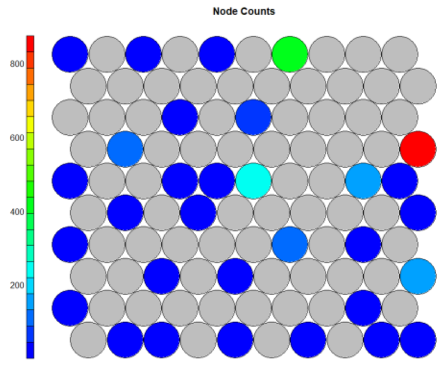


Figure 2.2: Node counts of the 10x10 SOM

The plot above demonstrates the distribution of the samples within each node. There is a number of nodes with no sample, while the others mainly have around 200 samples per node. This does not mean that the size of the SOM map is too large (21 million datapoints compared to 10x10 = 100 nodes in the map), but rather it means that the nodes are clearly distributed in some focused groups. This suggests that the 5 selected attributes are good variables and should be used for prediction task.

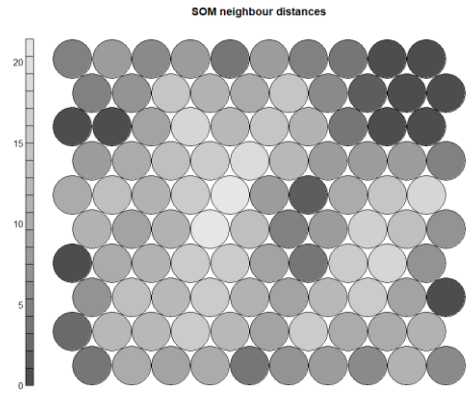


Figure 2.3: Neighbour distances for 10x10 SOM

The plot above demonstrates the distance of each node to its neighbours. Possible clusters can be identified through these nodes. For instance, there could be a cluster on the top right corner as the distance of each node to its neighbours is relatively close and uniform. This plot also suggests that the five variables are good for classification as there is a good division of the dataset using these attributes.

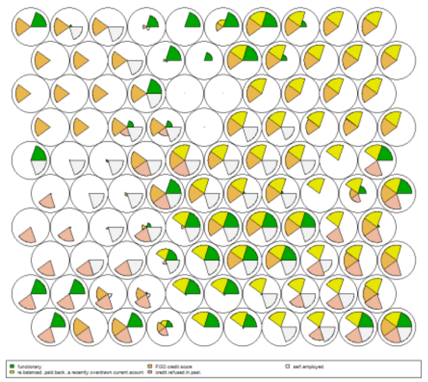


Figure 2.4: Code spread of 10x10 SOM

The plot above shows the training data as represented in each node. Each attribute is represented as a fan chart of the respective variable. This plot can be used to identify the clusters whose nodes have similar distributions of the five variables. It can also suggest the correlations between the variables. For instance, people with high FICO score often have little or no credit refused in the past, and vice versa. The plot suggests that the five variables seem to be closely related, and they can make a good division of the dataset. This means that they are good attributes for the prediction task.

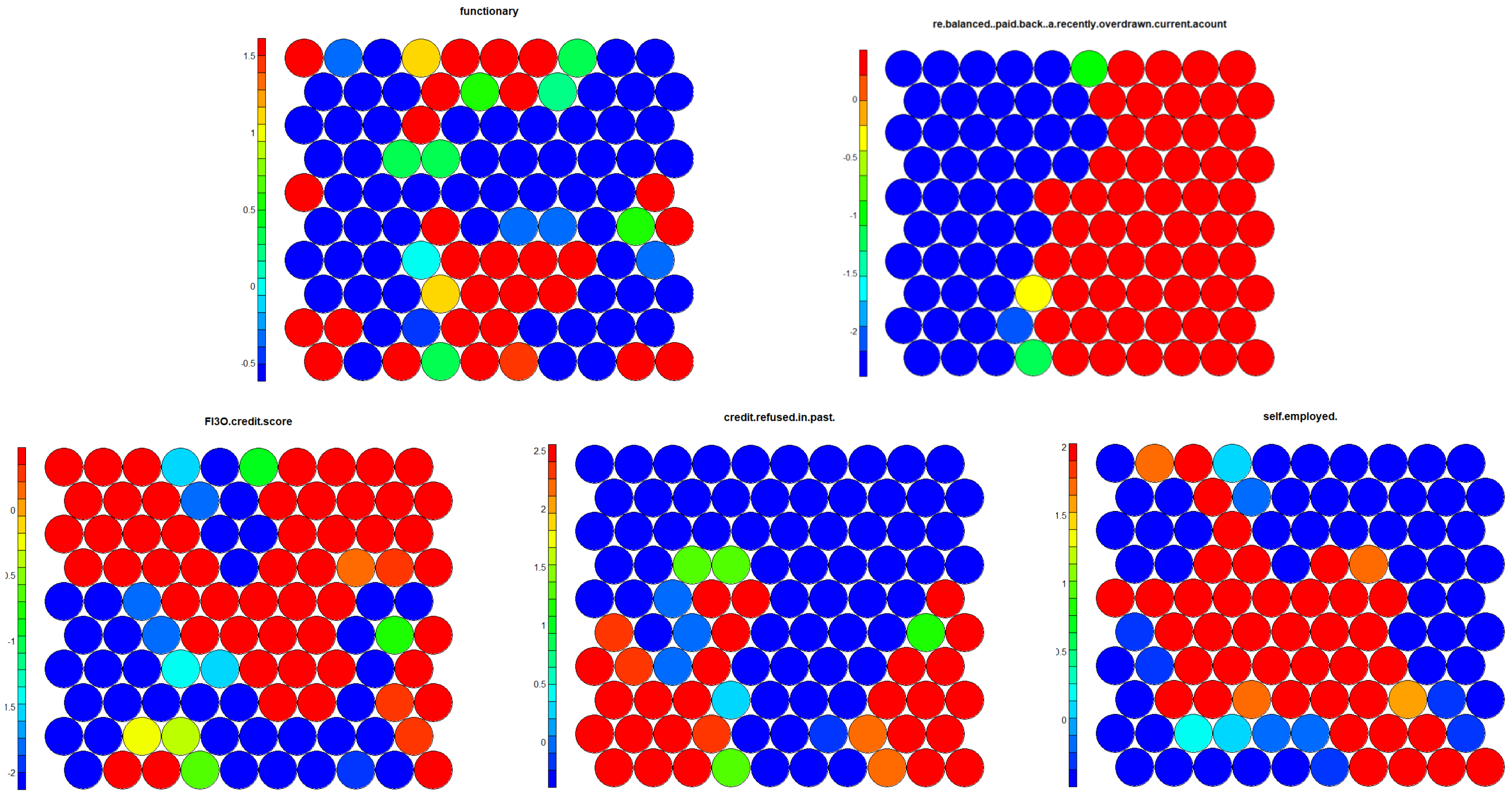


Figure 2.5: Heat maps for scaled variables
([1] Functionary, [2] Rebalanced, [3] FICO score, [4] Credit refused, [5] Self-employed)

The five heatmaps above demonstrates the distribution of the five variables, including Functionary, Rebalanced, FICO score, Credit refused and Self-employed respectively. Overall, the variables seem to be clearly distributed into groups. Possible clusters can also be identified easily.

These heatmaps can also be used to find the correlations between the variable. As mentioned above, nodes with low FICO score often has had credit refused in the past. This means that those who have been declined before often have low FICO score, which can result in a low credit rating.

However, there are still some exception nodes that do not follow the majority rules. This can result in difficulties in the prediction task as they can interfere with the result of the classification. For instance, on the bottom left, there are nodes that have credit refused in the past but they also have a relatively high FICO score. This can lead to a low accuracy of the prediction task.

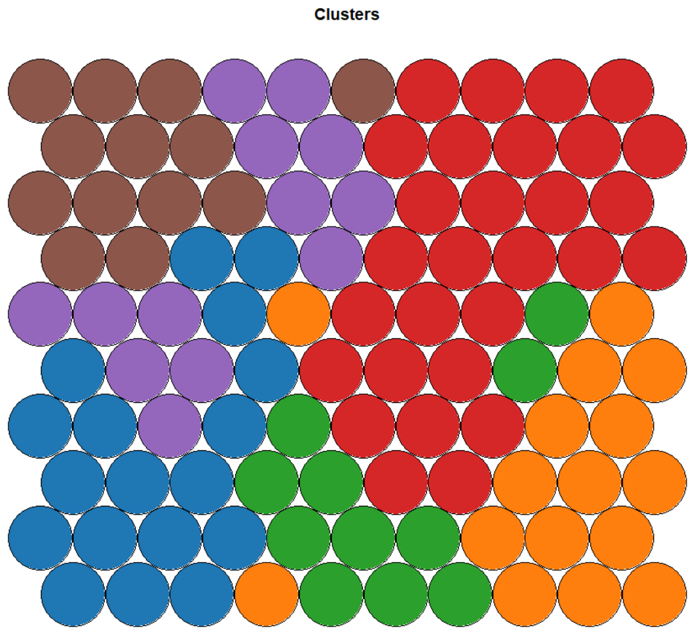


Figure 2.6: Clusters

The plot above demonstrates the six clusters formed based on the five variables. The plot reconfirms the assumption that these five variables are good for the prediction task. The divide the dataset into clearly distributed clusters. Nodes that are in a group are located close to their neighbours in the same group. However, there are some exceptions. One such example is that the node on the top row belongs to the cluster on the top left, but it is further from the others in the same group. This can lead to a reduction in the accuracy of the prediction task based on these variables.

Therefore, these five variables are the most suitable for the prediction task. However, it should be remembered that some variables that do not follow the majority trends can interfere with the accuracy of the model.

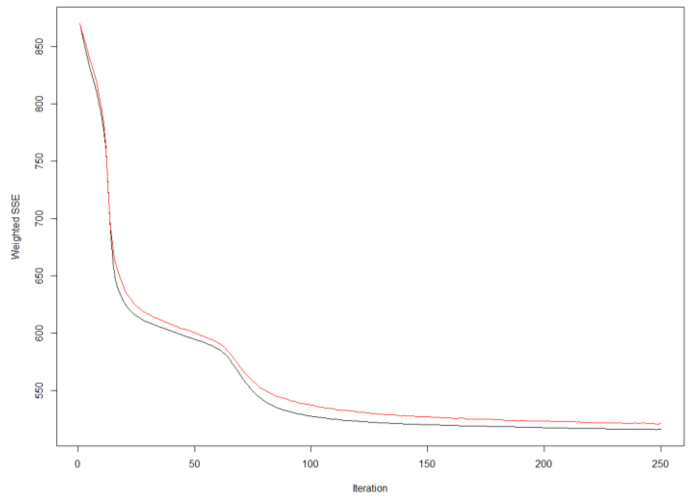
Question 2:

a. To optimise the accuracy of predicting the credit rating, instead of using the entire dataset to train the MLP, we should use the five attributes identified in Question 1 as they can clearly divides the dataset into notable groups. Thus, they can be used to rate each person's creditworthiness. Adjusting the weights to a reasonable range could also help optimising the model. The weights can be adjusted using the argument initFuncParam in mlp() function.
b. The dataset is divided into halves, one for training and the other for testing.

The best accuracy that I can obtained for the test set was around 61.37%. This can be calculated based on the confusion matrix below:

```
> confusionMatrix(trainset$targetsTest,predictTestSet)
       predictions
targets   1   2   3
      1 159  93   5
      2  79 367  21
      3  29 152  76
```

Compare to the accuracy of the MLP model in the lab task, there has been a significant improvement. The Iterative Error of the training set as well as the test set also reduced towards the end of the training stage. There also seems to be no overfitting problem where the model was overfitted to the training data while it did not fit well with the test set. The Regression Error plot also shows that the fitted model is nearly optimal. It nearly overlapped with the optimal line (black line), which means that the classification model is almost optimal.



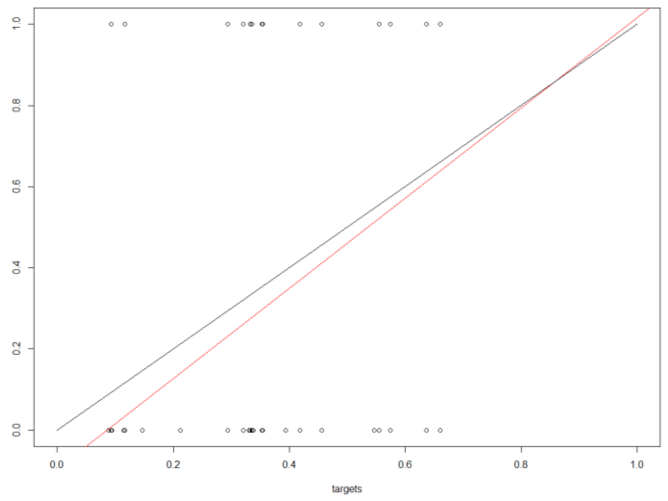
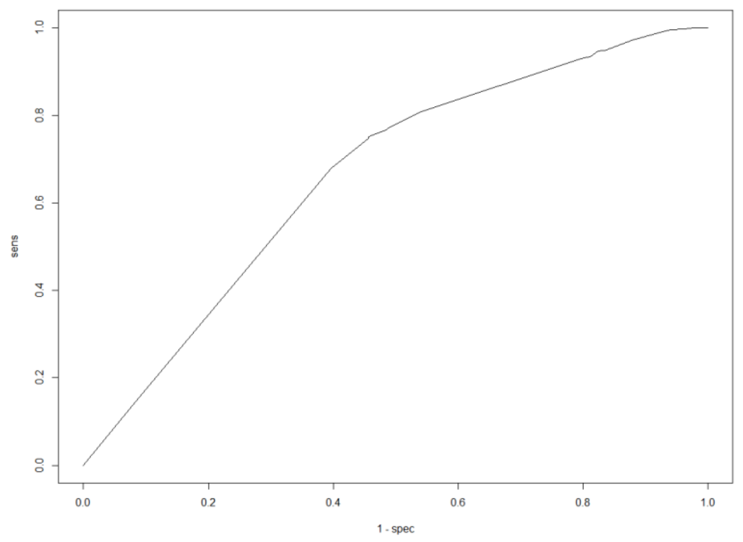


Figure 2.7: Itterative Error plot   Figure 2.8: Regression Error plot

The ROC curves below demonstrate the performance of the model for the training set and the test set. The ROC curves of both the training set and the test set are relatively accurate as they are further from the 45-degree baseline and closer to the top left corner. The Area Under Curve (AUC) of both sets are similar, meaning the performance of the model for both sets are relatively similar. This suggests that no overfitting problem has occurred during the training stage.



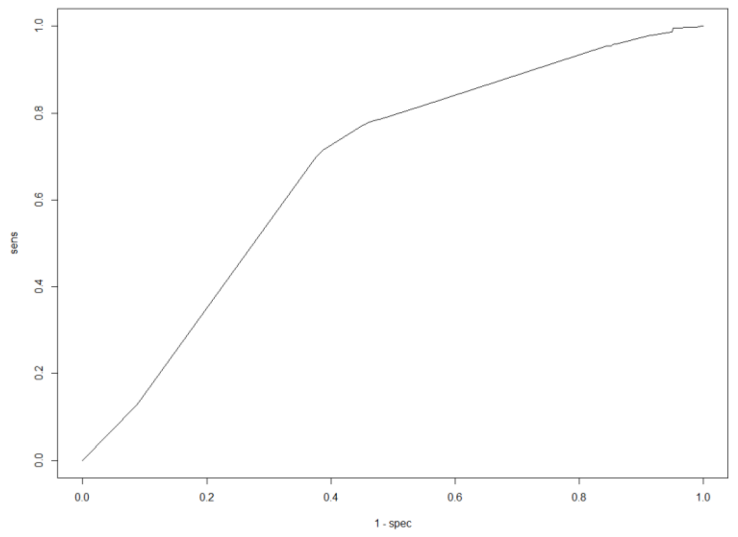


Figure 2.9: ROC curve for the training dataset   Figure 2.10: ROC curve for the test set

c. A 100% accuracy could not be obtained on this test dataset because the purpose of training in supervised machine learning, is to make a generalised model, that is not only specific for the training data or a specific test set. A dataset may contain outliers or invalid values which should not be fitted to the model, since it will lose its accuracy for general data. If a model fits perfectly for a specific test data (with an accuracy of 100%), an overfitting problem may have occurred although it happens on the test set. This could be due to some data leaking from the train set. A good model can predict relatively well on both training data and test data.

To get the prediction accuracy closer to 100%, there are several approaches:

- Parameters optimisation: Choosing a reasonable set of parameters for the MLP model such as the initial weights, biases or the learning rate for the MLP model can optimise the accuracy of the prediction task. It is important to consider the set of parameters as a whole rather than just individual parameters. A value might not be the best choice for a specific parameter, but it might be optimal in combination with another parameter.
- Remove noise and outliers: Noise and outliers should be removed in the data pre-processing stage as they can significantly distort the results of the classification task.
- Attributes optimisation: Choosing a good set of attributes of the dataset could help improve the prediction accuracy. The set of the attributes chosen should be able to make a clear division of the dataset into different groups, or clusters.