Lab 4

Group 22 - balra340, tejma768

October 3, 2018

Assignment 1

For this question we a datafile **prices-and-earnings.txt**.

1.1

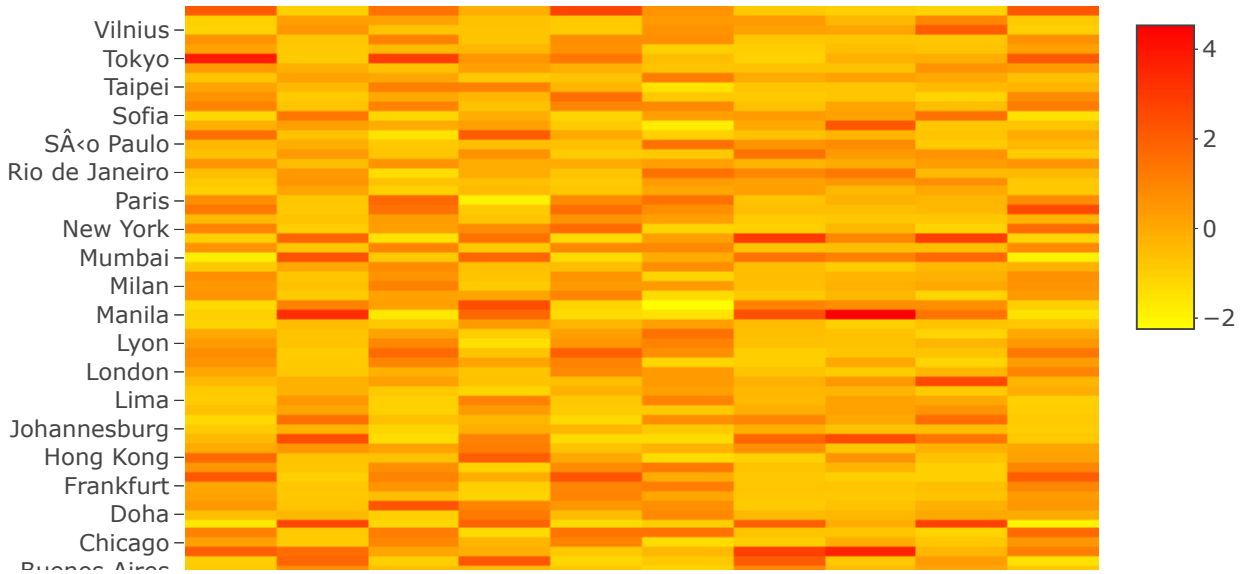
We are supposed to import the data and filter the data with required columns [1,2,3,6,7,9,10,16,17,18,19],and first column is used as labels.

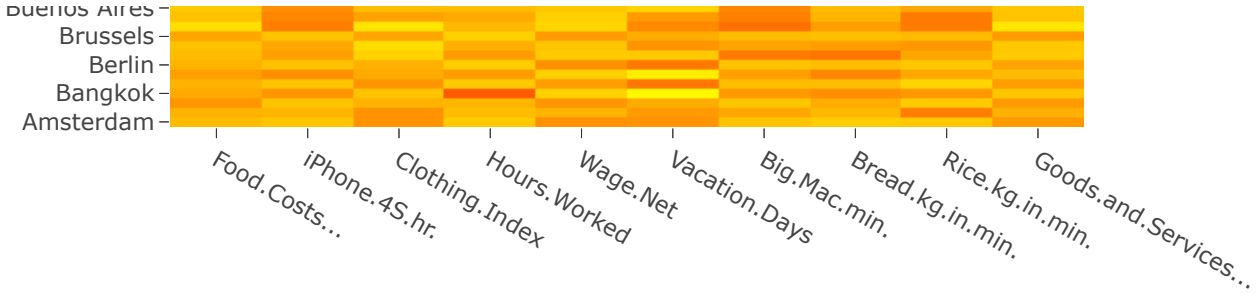
```
my_data <- read.csv2("prices-and-earnings.txt" ,header = T, sep = "\t" , stringsAsFactors = T,dec =
".")


my_data <- my_data[ , c(1,2,5,6,7,9,10,16,17,18,19)]

row.names(my_data) <- my_data[,1]

#my_data
```

1.2

We plot a heat map in this question without any reordering.

```
library(plotly)

my_data_scaled <- scale(my_data[, 2:11])

plot_ly(x= colnames(my_data_scaled), y= factor(my_data[,1])
,
        z=my_data_scaled, type="heatmap", colors = colorRamp(c("yellow", "red")))
```





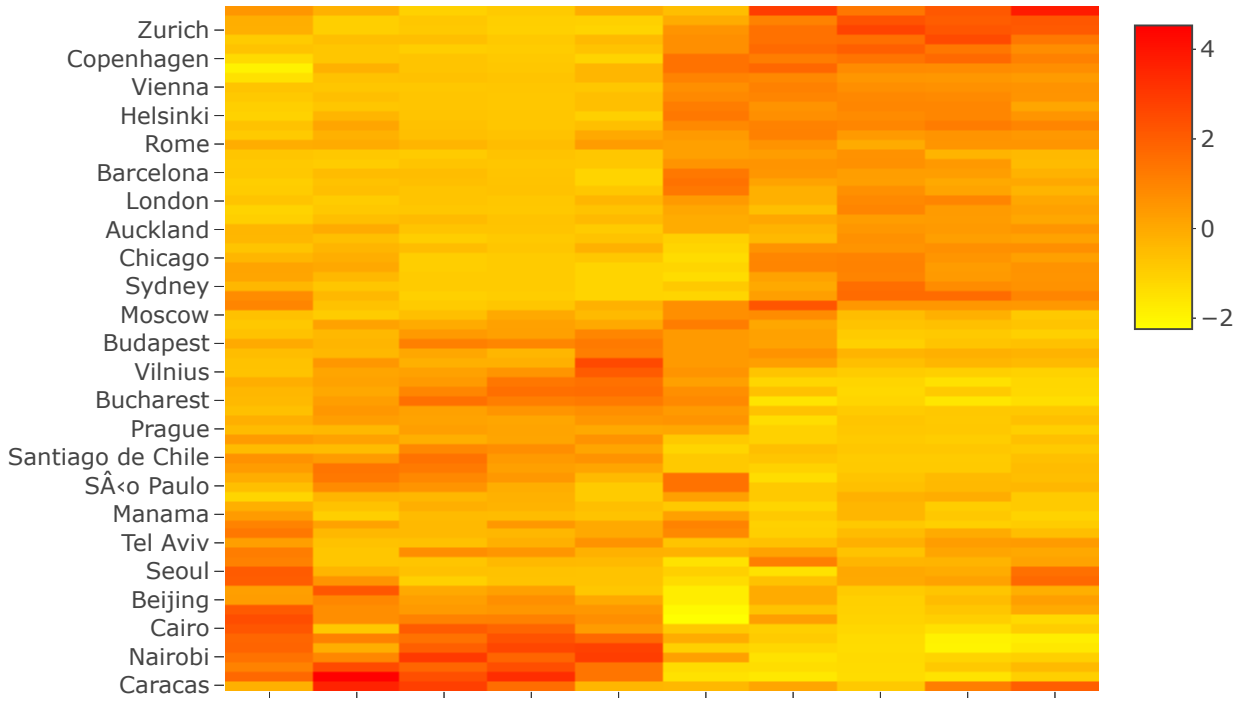
Its not easy to find clusters but finding outliers is easy.

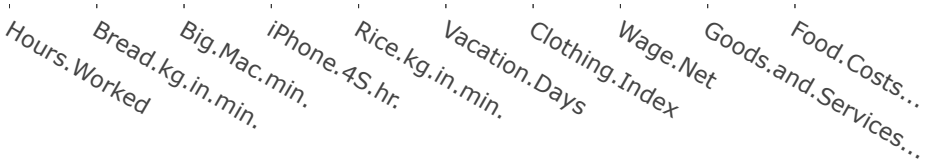
1.3

In this part we are supposed to compute distance matrices using Euclidean and one minus correlation. For both cases we compute Hamiltonian Path Length and use Hirearchial clustering as optimization algorithm.

Euclidean

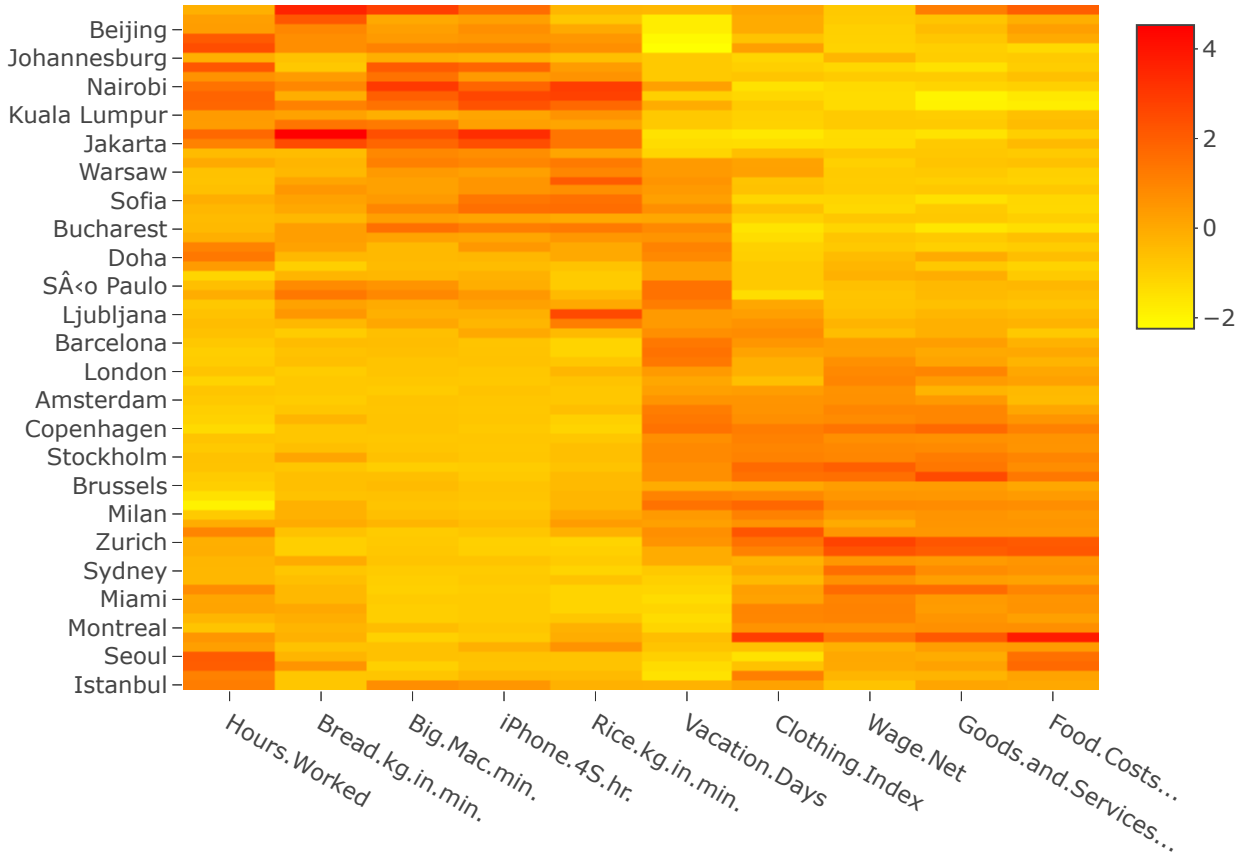
```
library(seriation)

rdist_euc <- dist(my_data_scaled)
rdist_ser <- seriate(rdist_euc, method = "OLO")
rorder_euc <-get_order(rdist_ser)

cdist_euc <- dist(t(my_data_scaled))
cdist_ser <- seriate(cdist_euc, method = "OLO")
corder_euc <- get_order(cdist_ser)

reordered_euc <- my_data_scaled[rev(rorder_euc),corder_euc]

plot_1 <- plot_ly(x=colnames(reordered_euc), y=rownames(reordered_euc),
              z=reordered_euc, type="heatmap", colors = colorRamp(c("yellow","red")))

plot_1
```





One minus correlation

```
rdist_cor <- 1 - as.dist(cor(t(my_data_scaled)))
rdist_ser <- seriate(rdist_cor, method = "OLO")
rorder_cor <- get_order(rdist_ser)

cdist_cor <- 1 - as.dist(cor(my_data_scaled))
cdist_ser <- seriate(cdist_cor, method = "OLO")
corder_cor <- get_order(cdist_ser)

reordered_cor <- my_data_scaled[rev(rorder_cor),corder_cor]

plot_2 <- plot_ly(x=colnames(reordered_cor), y=rownames(reordered_cor),
              z=reordered_cor, type="heatmap", colors = colorRamp(c("yellow","red")))


plot_2
```



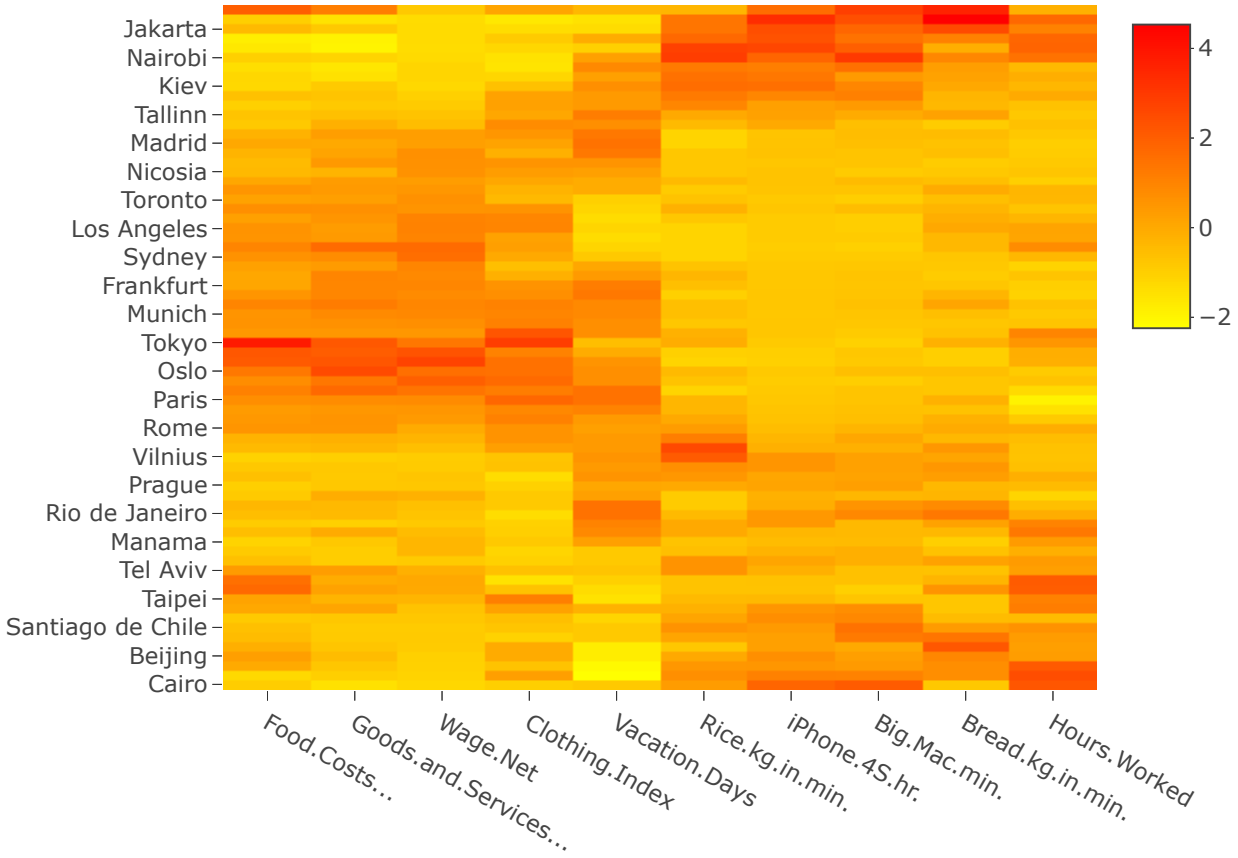
We feel the one using euclidean distance was better because according to the plot it is easy to find the outliers and classify clusters easily.

The graph shows a very clear clustering in the euclidean cases. We are able to judge which cities havr large working hours. And because of good clustering we find all high working hours cities together. And this helps us evaluate the dependent variables easily. The countries with high net wages has less work hours required to buy an iphone. This is clearly shown in the graph.We also find those countries spending more on food and clothing. In jakarta the price of bread is very high when compared to rice.

1.4

For this part of the question we compute a permutation that optimizes Hamiltonian Path Length but uses Traveling Salesman Problem as solver.

```
rorder_tsp <- get_order(seriate(rdist_euc, method = "TSP"))
corder_tsp <- get_order(seriate(cdist_euc, method = "TSP"))
reordered_tsp <- my_data_scaled[rev(rorder_tsp),corder_tsp]


plot_3 <- plot_ly(x=colnames(reordered_tsp), y=rownames(reordered_tsp),
                  z=reordered_tsp, type="heatmap", colors = colorRamp(c("yellow","red")))

plot_3
```



```
HC <- criterion(rdist_euc,order=seriate(rdist_euc, method = "GW"),method = c("Gradient_raw","Path_len
gth"))

TSP <- criterion(rdist_euc,order=seriate(rdist_euc, method = "TSP"),method = c("Gradient_raw","Path_l
ength"))

HC
```

```
## Gradient_raw  Path_length
##   59506.0000     128.5114
```

```
TSP
```

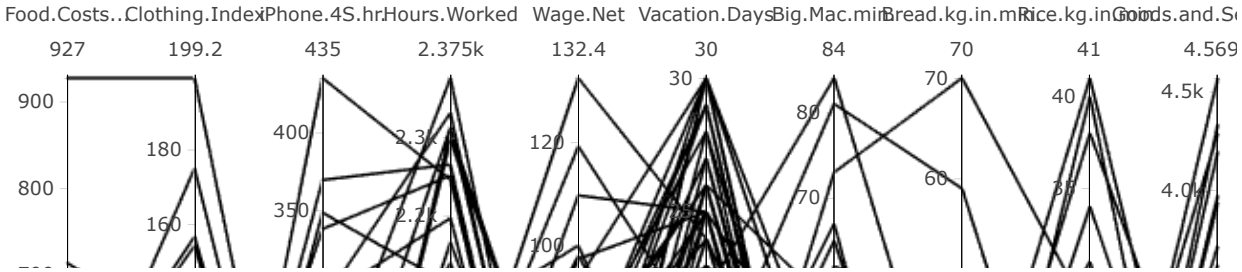
```
## Gradient_raw  Path_length
##     30658.00       122.26
```

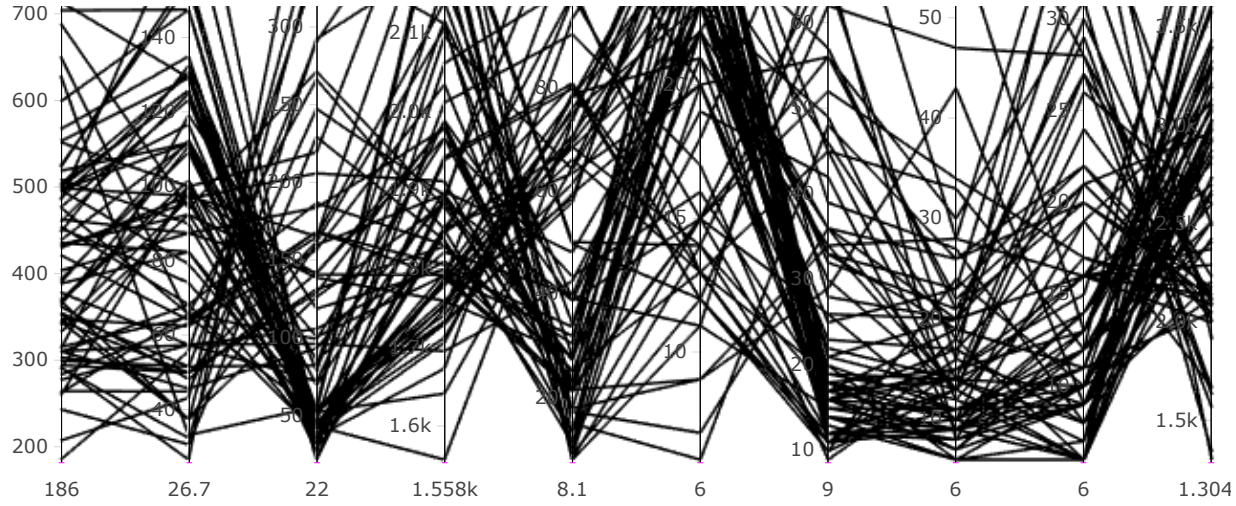
We feel the TSP plot is better. By comparing the objective function also we found that TSP was working better.Even the gradient and path length says TSP is better. Gradient Raw is considered the merit and path length is the loss. Higher the Gradient Raw makes the algorithm a better one hence TSP is the better problem solver.

1.5

Now we are supposed to use paralled coordinate plots from unsorted data and try to compute it manually.

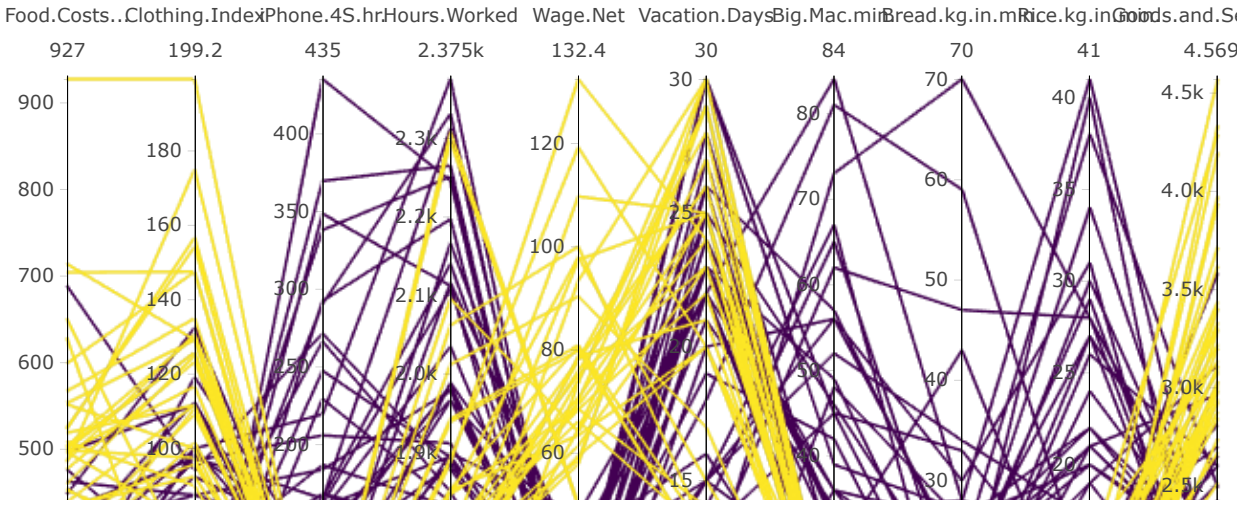
```
library(ggplot2)
library(plotly)
p1 <- as.data.frame(my_data) %>%
  plot_ly(type = 'parcoords',
          dimensions = list(
            list(label = "Food.Costs...", values =~Food.Costs...),
            list(label = "Clothing.Index", values =~Clothing.Index),
            list(label = "iPhone.4S.hr.", values =~iPhone.4S.hr.),
            list(label = "Hours.Worked", values =~Hours.Worked),
            list(label = "Wage.Net", values =~Wage.Net),
            list(label = "Vacation.Days", values =~Vacation.Days),
            list(label = "Big.Mac.min.", values =~Big.Mac.min.),
            list(label = "Bread.kg.in.min.", values =~Bread.kg.in.min.),
            list(label = "Rice.kg.in.min.", values =~Rice.kg.in.min.),
            list(label = "Goods.and.Services...", values =~Goods.and.Services...)
          )
  )
p1
```

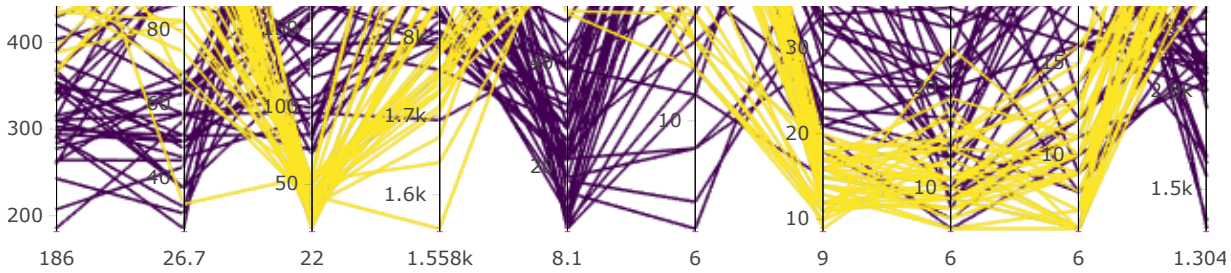




we feel wage.net is the most important variable. The influence of this on the other varibales is very strong. We were able to find clusters also in variables like iphone, big mac, bread.

```
p2 <-my_data %>%
  mutate(my_data = as.integer(Wage.Net > 48.6875)) %>%
  plot_ly(type = 'parcoords',
          dimensions = list(
            list(label = "Food.Costs...", values =~Food.Costs...),
            list(label = "Clothing.Index", values =~Clothing.Index),
            list(label = "iPhone.4S.hr.", values =~iPhone.4S.hr.),
            list(label = "Hours.Worked", values =~Hours.Worked),
            list(label = "Wage.Net", values =~Wage.Net),
            list(label = "Vacation.Days", values =~Vacation.Days),
            list(label = "Big.Mac.min.", values =~Big.Mac.min.),
            list(label = "Bread.kg.in.min.", values =~Bread.kg.in.min.),
            list(label = "Rice.kg.in.min.", values =~Rice.kg.in.min.),
            list(label = "Goods.and.Services...", values =~Goods.and.Services...)
          ),
            line = list(color = ~as.numeric(my_data))
  )
p2
```



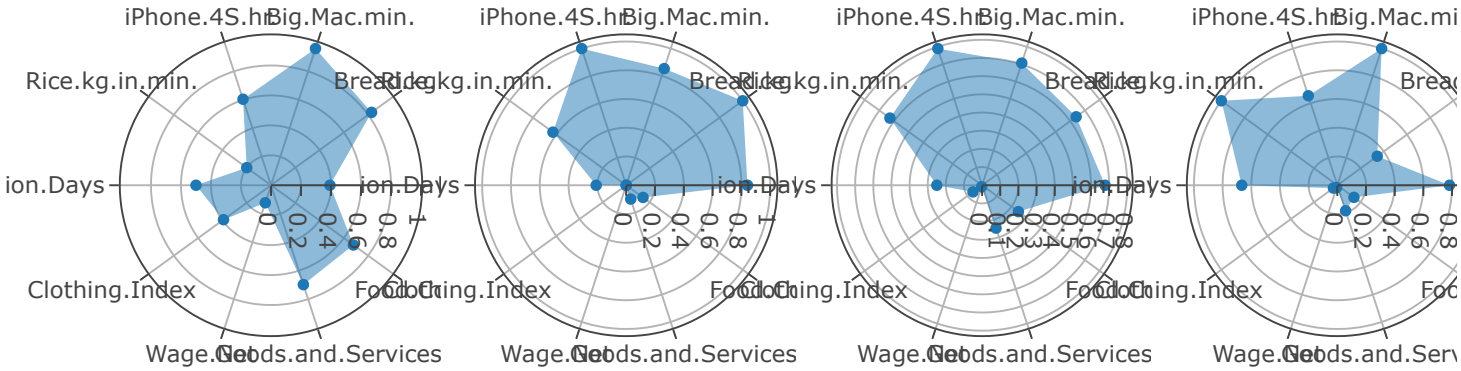


** 1.6**

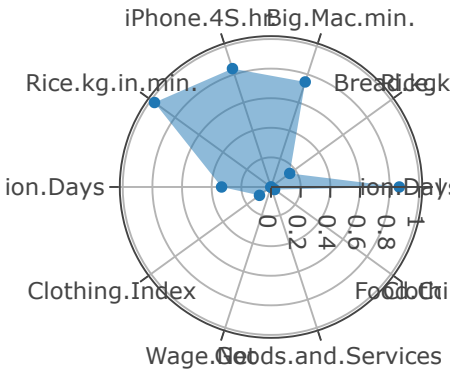
We use the data from HC solver abd create a radar chart diagram with juxtaposed radars.

```
library(plotly)
library(dplyr)
library(scales)

Ps <- list()
nPlot=72

as.data.frame(reordered_euc) %>%
  add_rownames( var = "group" ) %>%
  mutate_each(funs(rescale), -group) -> data_radar

for (i in 1:nPlot){
  Ps[[i]] <- htmltools::tags$div(
    plot_ly(type = 'scatterpolar',
            r=as.numeric(data_radar[i,-1]),
            theta= colnames(data_radar)[-1],
            fill="toself")%>%
      layout(title=data_radar$group[i]), style="width: 25%;")
}

h <-htmltools::tags$div(style = "display: flex; flex-wrap: wrap", Ps)

htmltools::browsable(h)
```

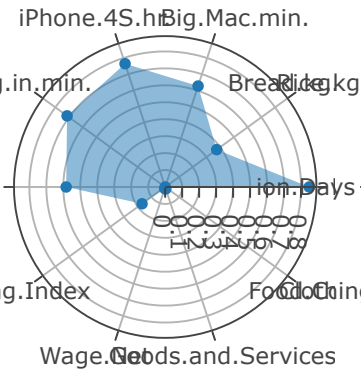
Caracas Manila Jakarta Nairobi



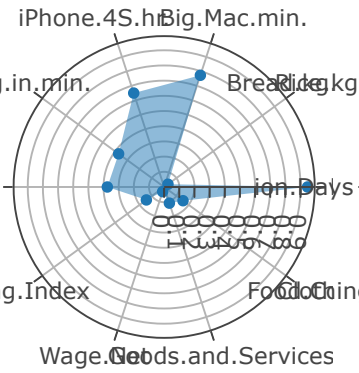
Delhi



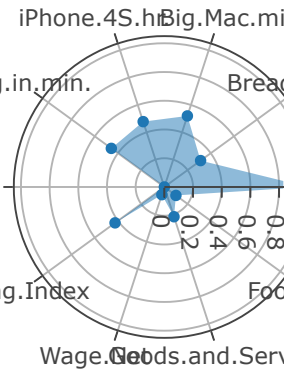
Mumbai



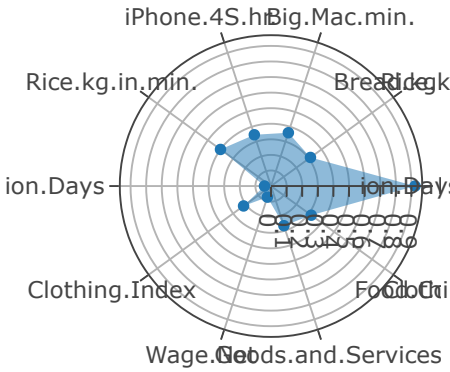
Cairo



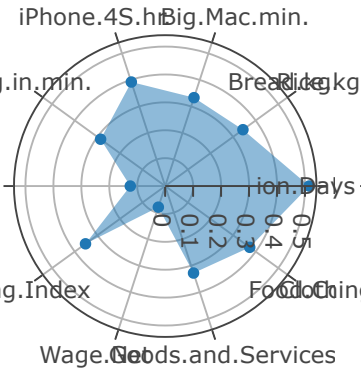
Mexico City



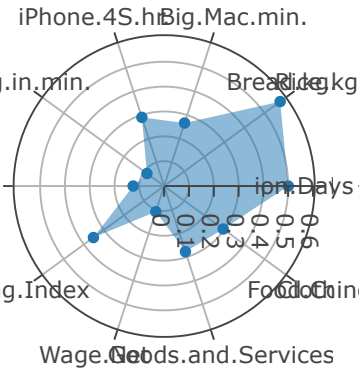
Bangkok



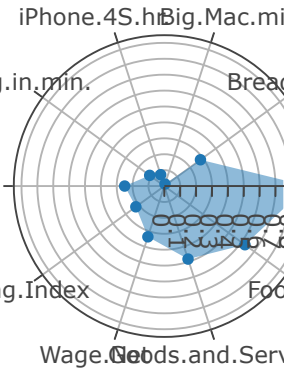
Beijing



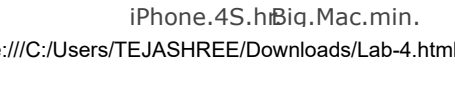
Shanghai



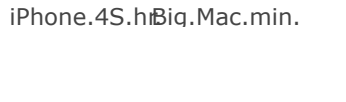
Hong Kong



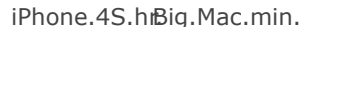
Seoul



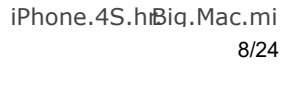
Taipei

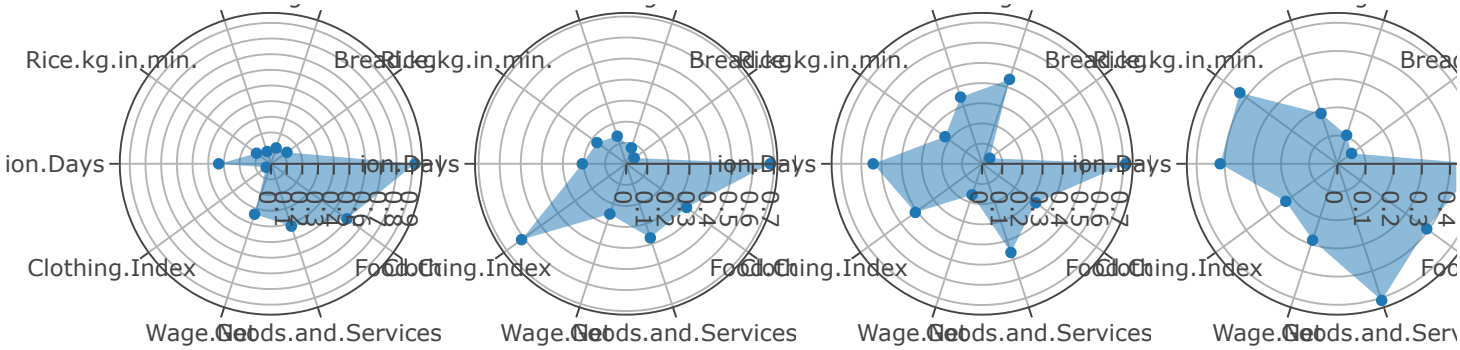


Istanbul

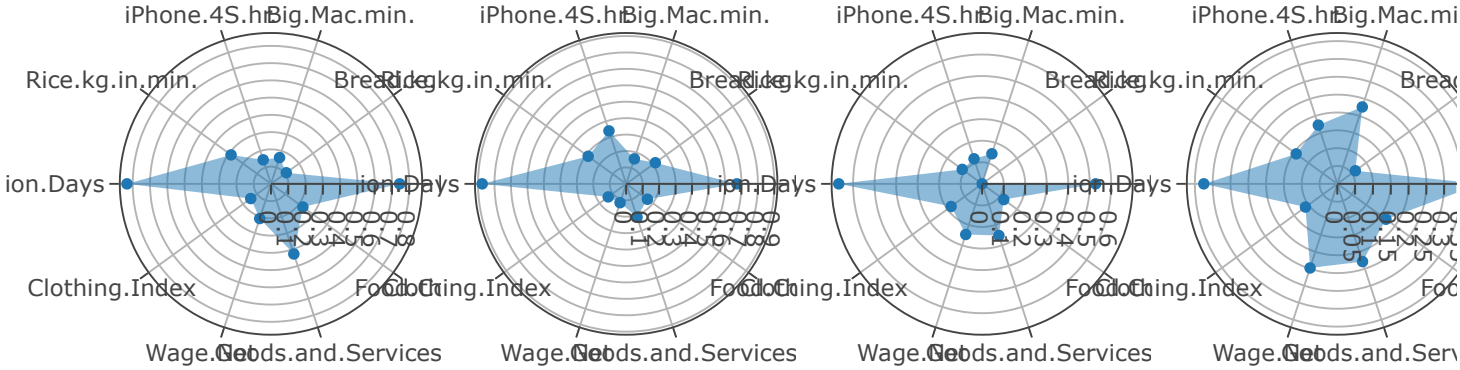


Tel Aviv

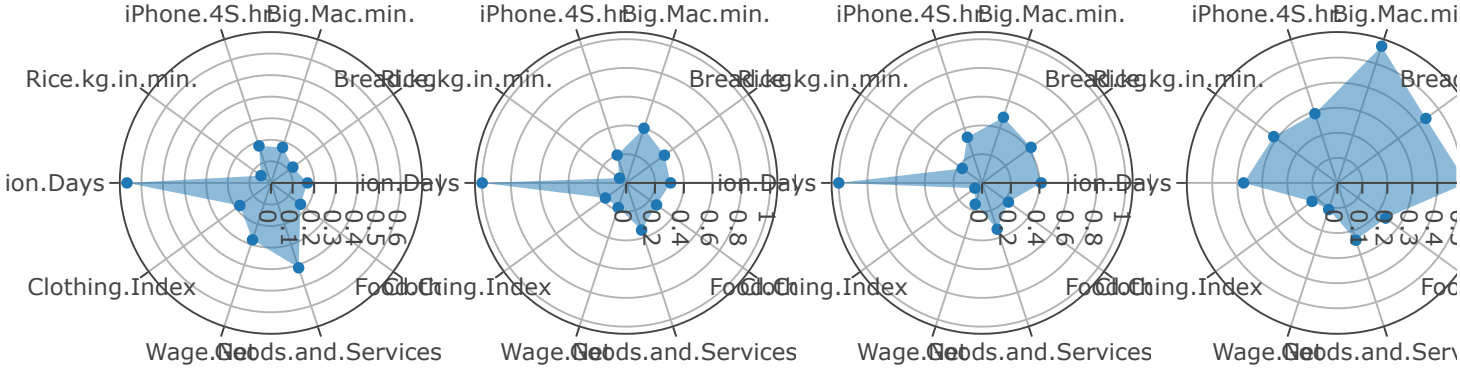




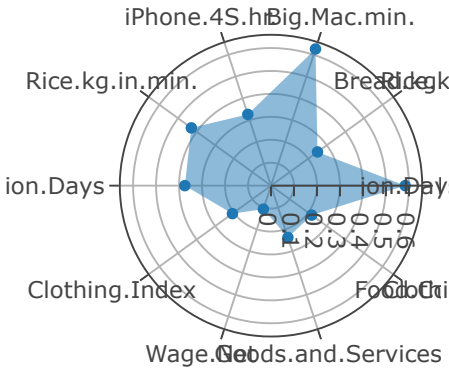
Doha Lima Manama Johannesburg



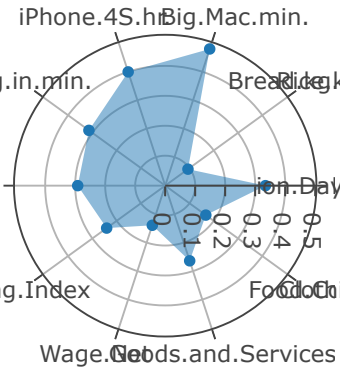
Lisbon SA‹o Paulo Rio de Janeiro BogotA‡



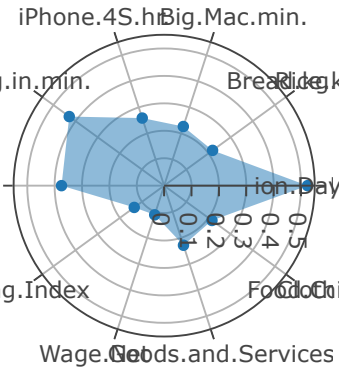
Santiago de Chile



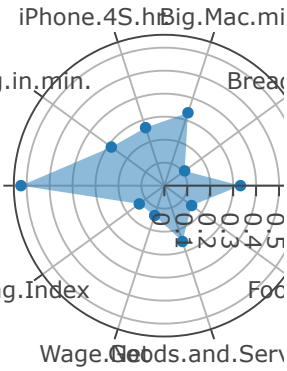
Buenos Aires



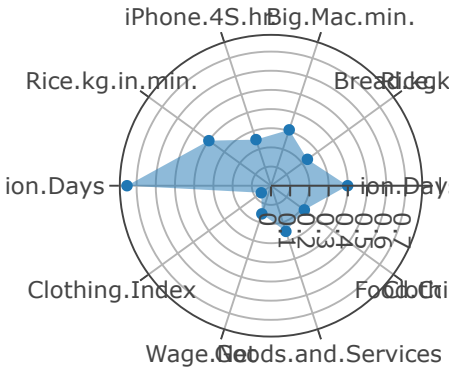
Kuala Lumpur



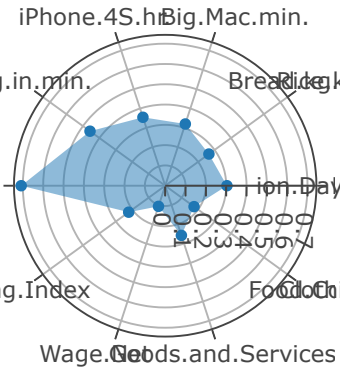
Prague



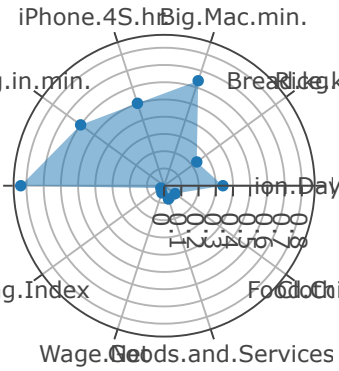
Bratislava



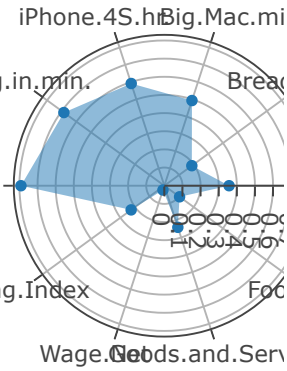
Riga



Bucharest



Kiev

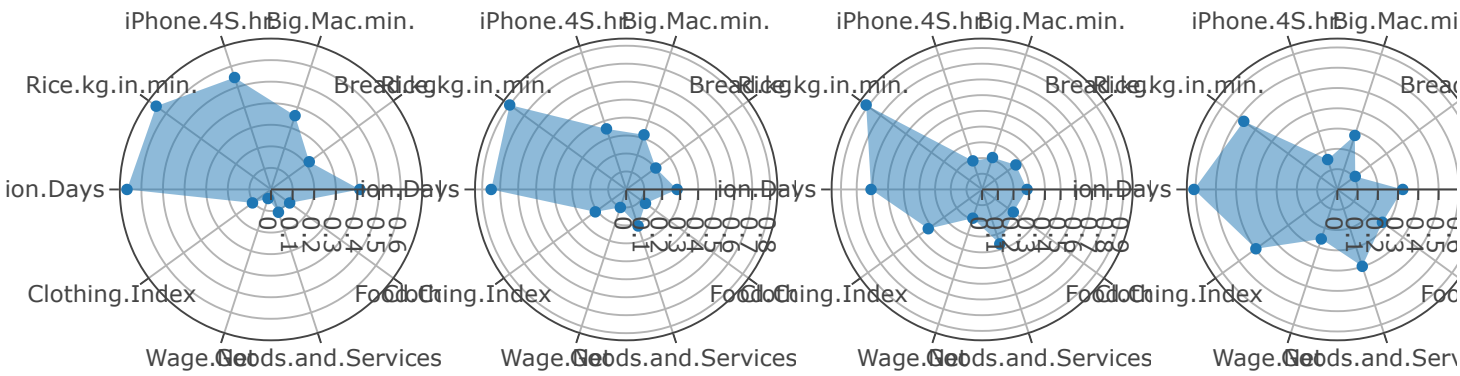


Sofia

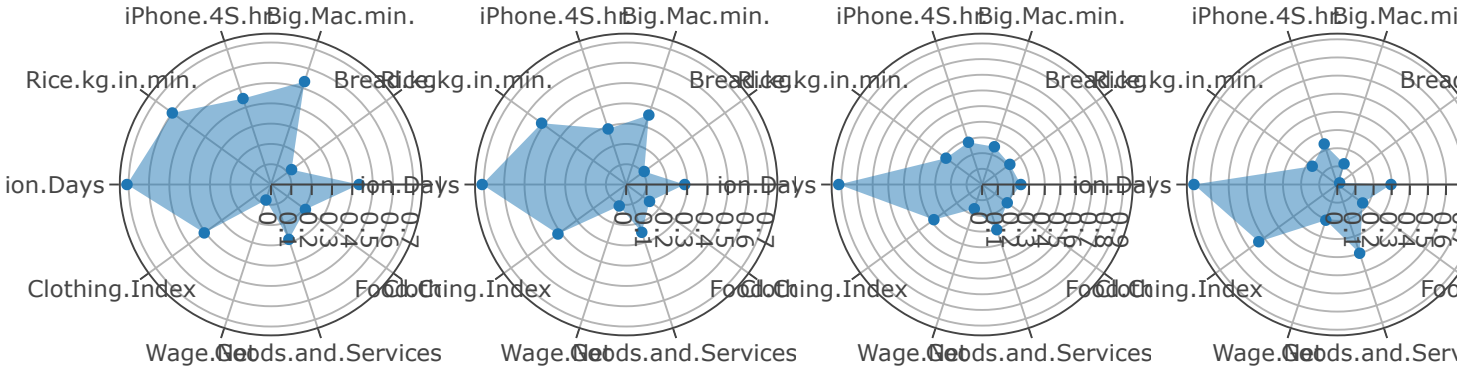
Vilnius

Ljubljana

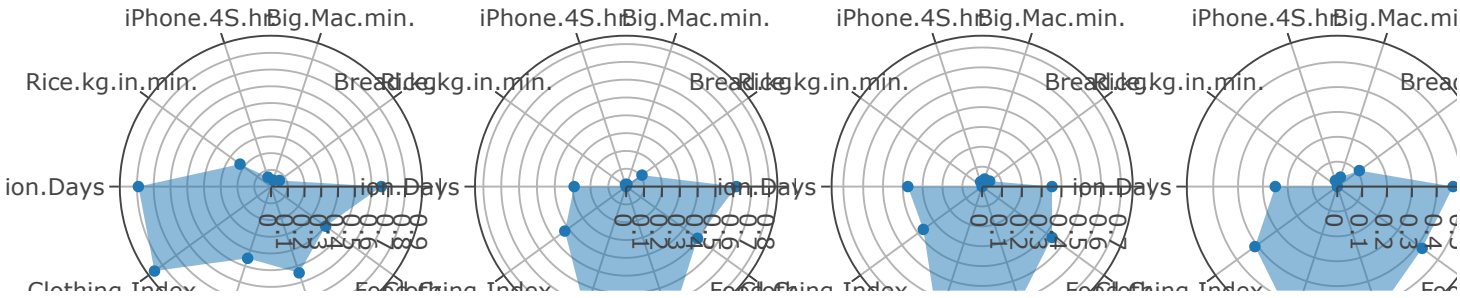
Athens



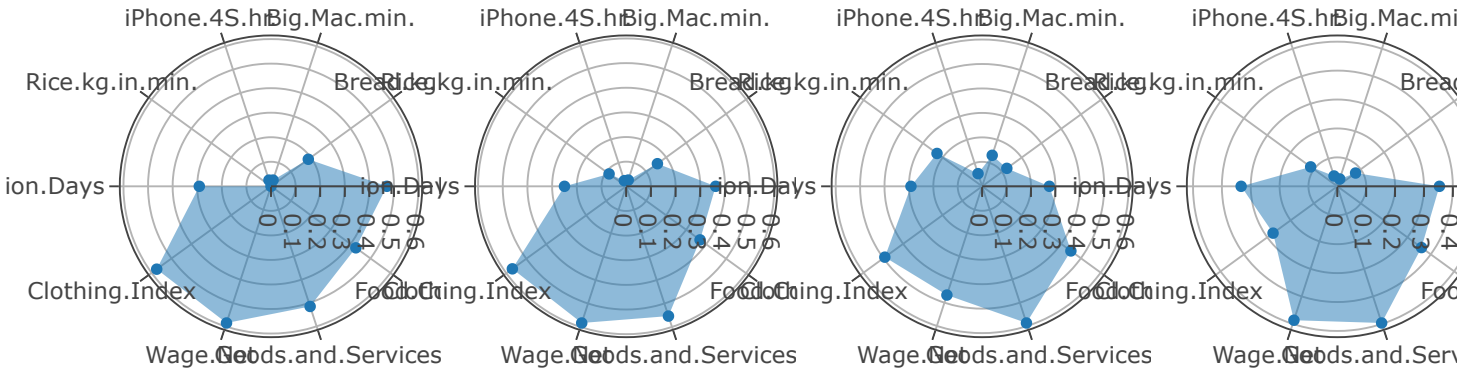
Budapest Warsaw Tallinn Moscow



Dubai New York Sydney Miami



Los Angeles

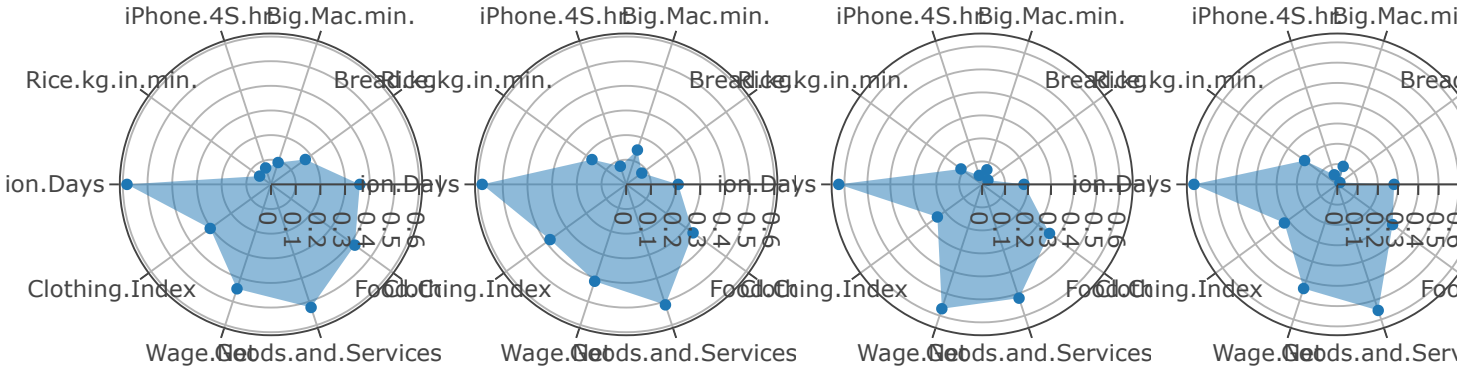


Chicago

Montreal

Toronto

Auckland



Brussels

Dublin

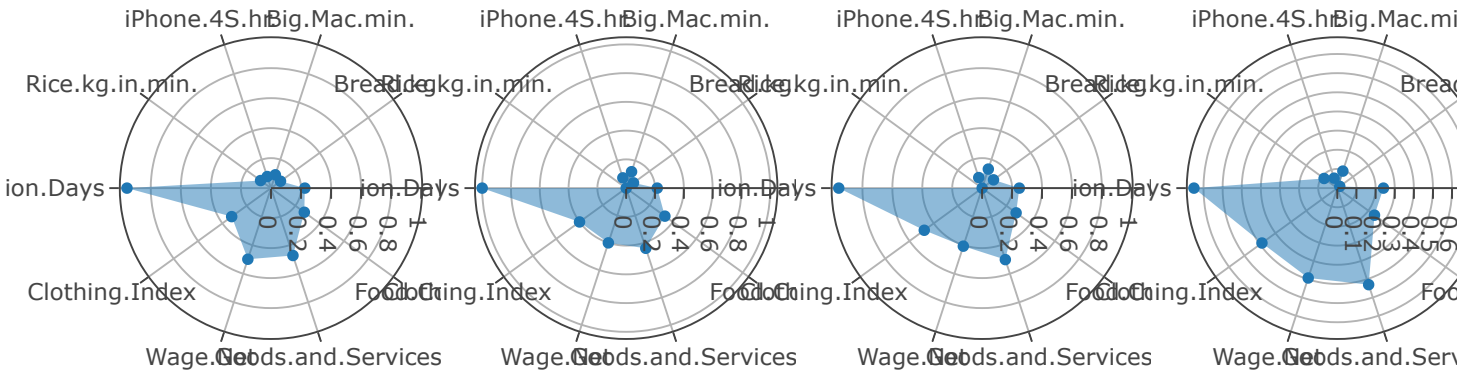
London

Berlin

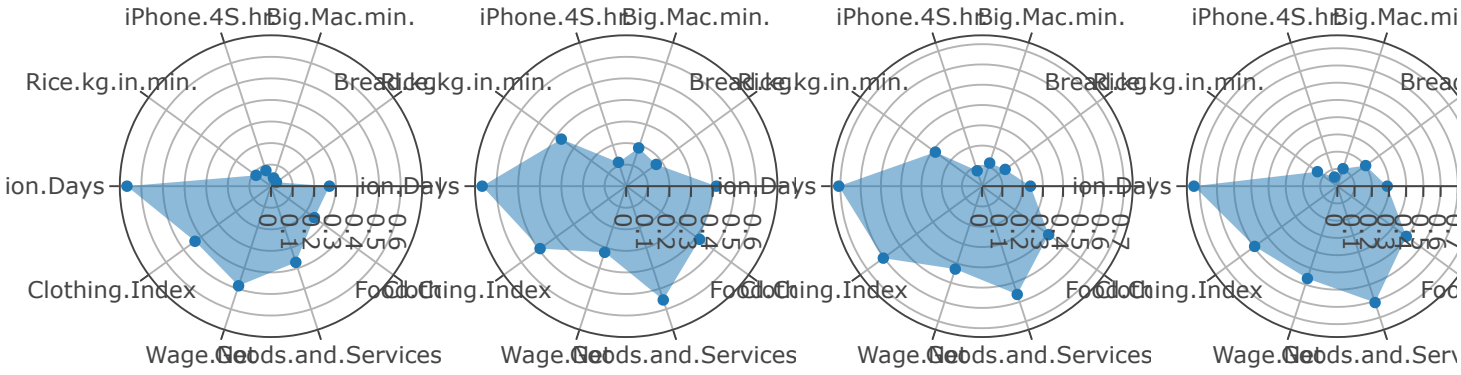
Madrid

Barcelona

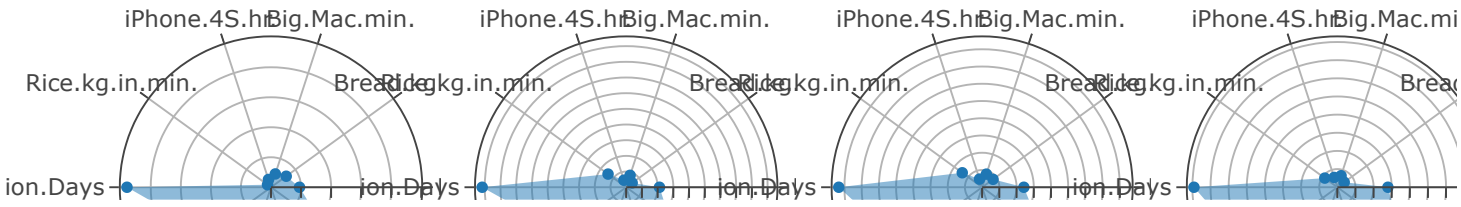
Amsterdam



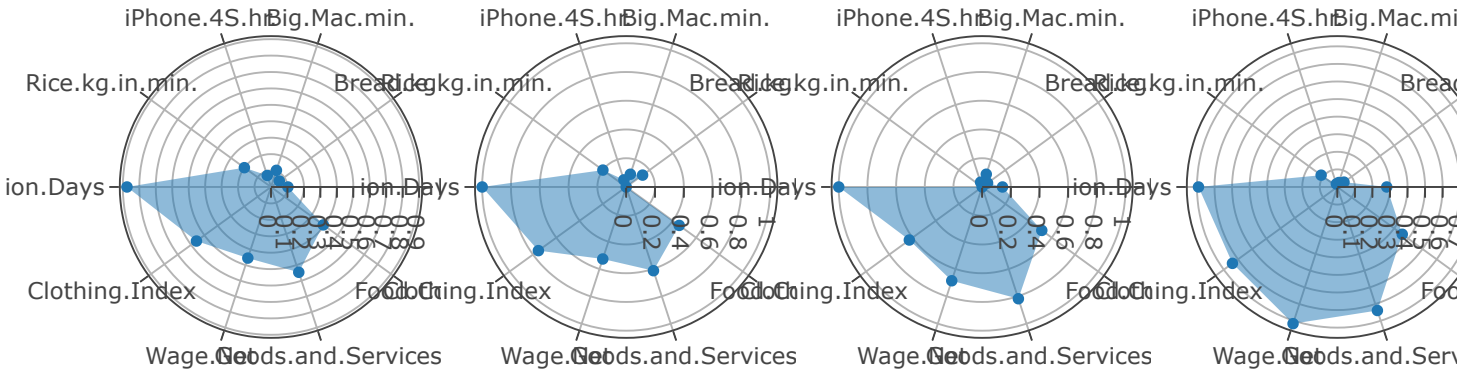
Nicosia Rome Milan Stockholm



Helsinki Frankfurt Munich Vienna



Lyon

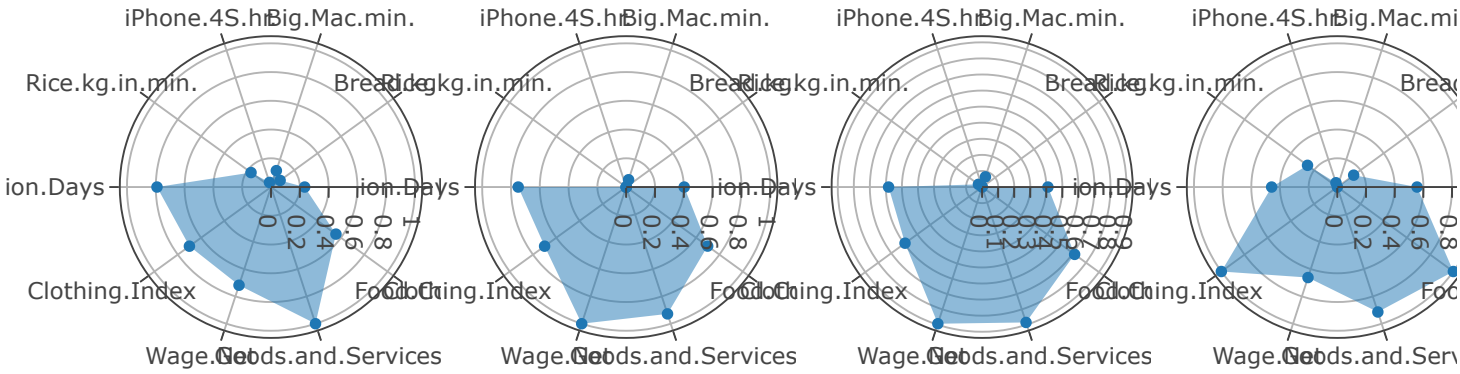


Paris

Copenhagen

Luxembourg

Oslo



Zurich

Geneva

Tokyo

Berlin,Madrid , Barcelona make a cluster with vaction variable in consideration

Lisbon, Sao Paulo, Rio de Janeiro also forms a cluster.

Caracas is the outlier.

1.7

We feel heatmaps were better as it was more informative about clusters and outliers.

Assignment 2: Trellis plots for population analysis

```
library(ggplot2)
library(plotly)
#install.packages("scatterplot3d")
library(scatterplot3d)
require(scatterplot3d)
library(lattice)
```

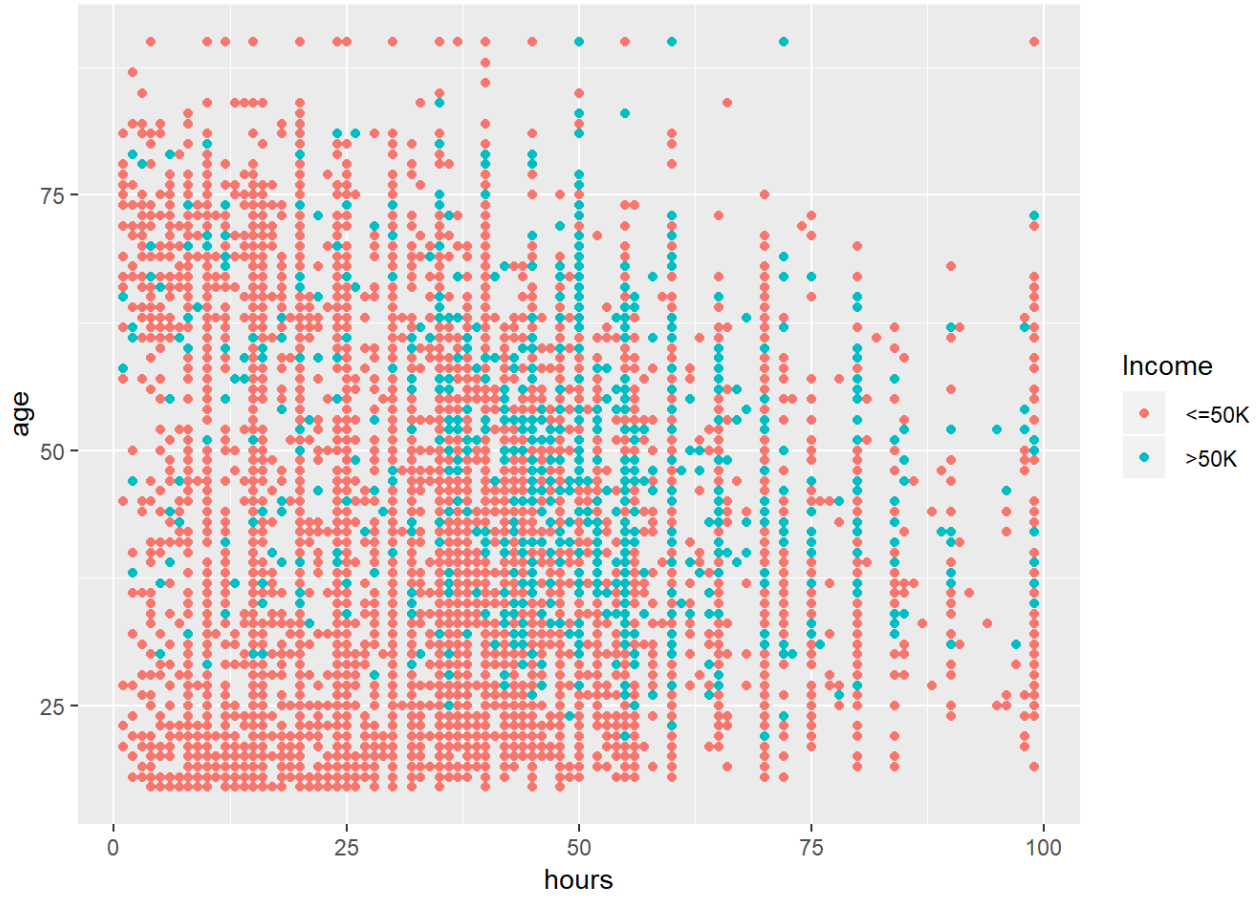
```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:seriation':
##
##     panel.lines
```

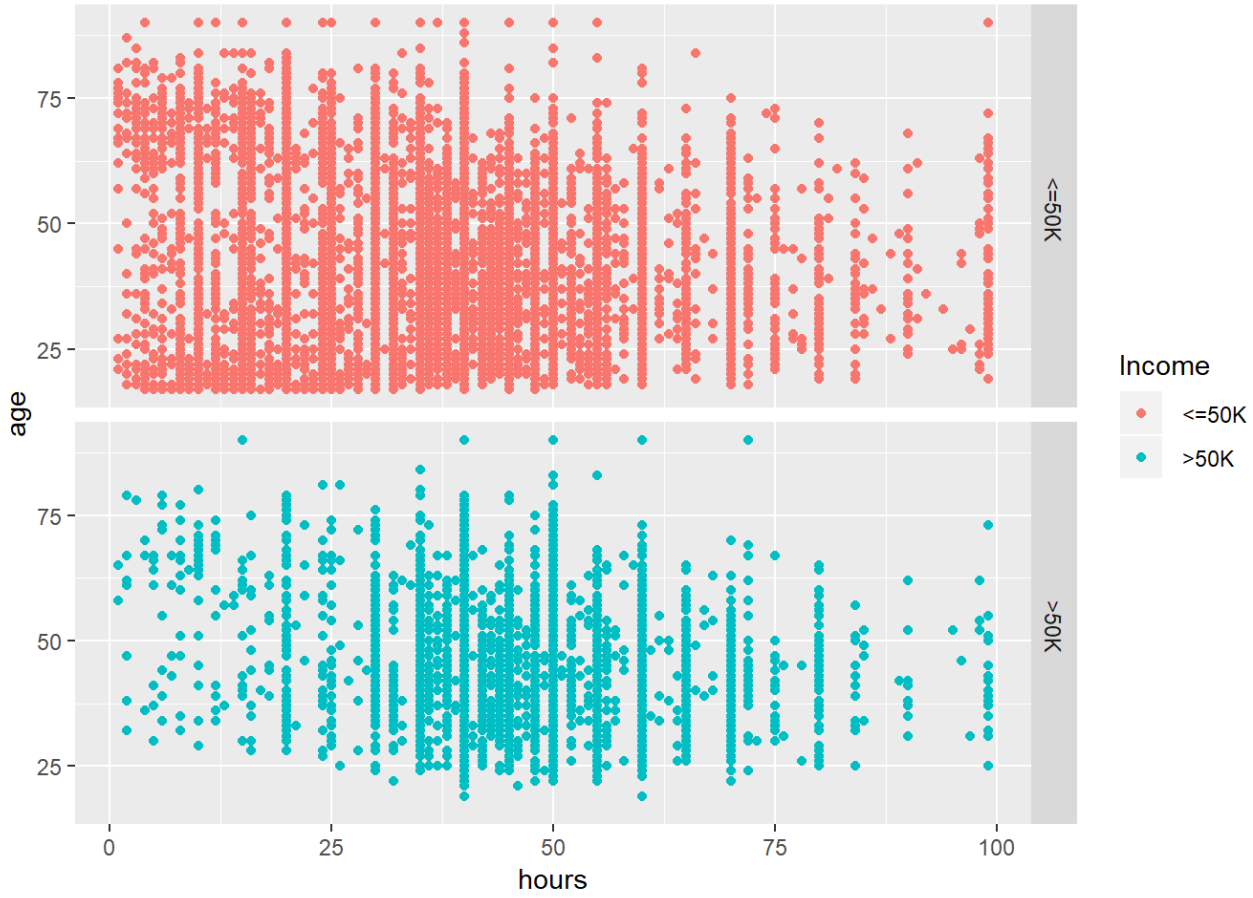
```
Sys.setenv('MAPBOX_TOKEN' = 'pk.eyJ1IjoidGVqYXNocmVlcm0iLCJhIjoiY2pta21ud212MHBxdTNsbmoxeGp1OGp4ZCJ9.
mn_aZzsqDMNS9cXtkAhfIw')
```

1. Use ggplot2 to make a scatter plot of Hours per Week versus age where observations are colored by Income level. Why it is problematic to analyze this plot? Make a trellis plot of the same kind where you condition on Income Level. What new conclusions can you make here?

```
library(ggplot2)

my_data <- read.csv("adult.csv")

colnames(my_data) <- c("age", "workclass", "fnlwgt", "education", "num","status", "occupation", "rela
tionship", "race", "sex", "gain", "loss", "hours", "country", "Income")

#scatter plot
p <- ggplot(my_data, aes(x = hours, y = age)) + geom_point(aes(color = Income))
p
```




```
#trellis plot
p1 <- ggplot(my_data, aes(x = hours, y = age)) + geom_point(aes(color = Income)) +
                              facet_grid(Income~.)
p1
```



From the scatter plot we can only say that people who have been working for more hours a week even with lesse than 50k income are at their old ages. Not much information can be depicted from the scatter plot as the data is overlapped. To avoid overlapping and to get a better understanding of the plot we use Trellis plot.

Trellis plot makes it easier to analyze the plot, the plot is split into 2 panels on the basis of income, one panel shows people whose income is less than or equal to 50k and the other shows people whose income is greater than 50k. Hence trellis plot is a better option.

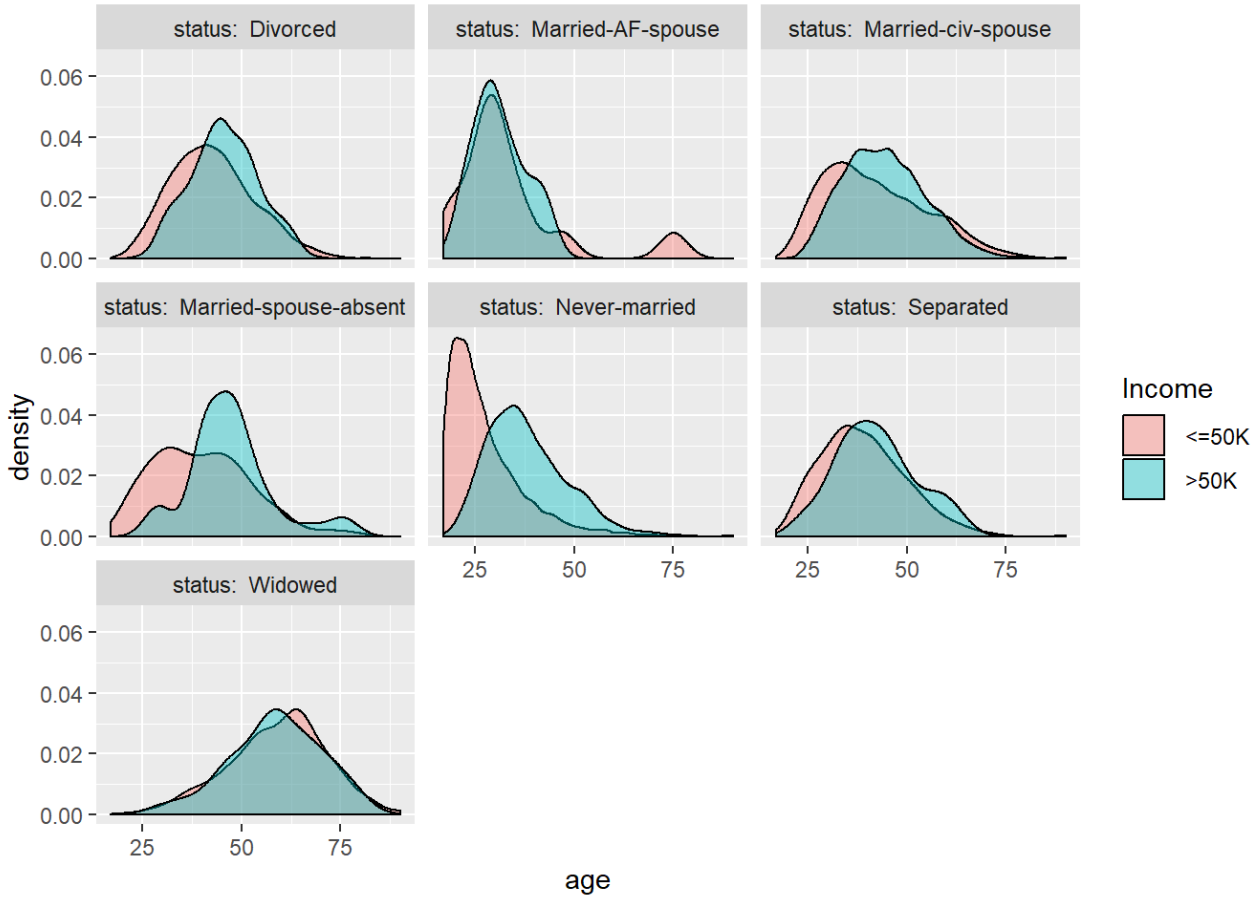
We can see from the Trellis plot people who are getting an income of more than 50k work for 30-55 hours a week who fall under the age group of 25-75 as the plot is denser in that region.

And people who are getting income of lesser than 50k, are in the age group 25-60 and they seem to work for more than 75 hours a week.

2. Use ggplot2 to create a density plot of age grouped by the Income level. Create a trellis plot of the same kind where you condition on Marital Status. Analyze these two plots and make conclusions.

```
#2.
#density plot
p <- ggplot(my_data, aes(age, fill = Income)) + geom_density(alpha=0.4)
p
```

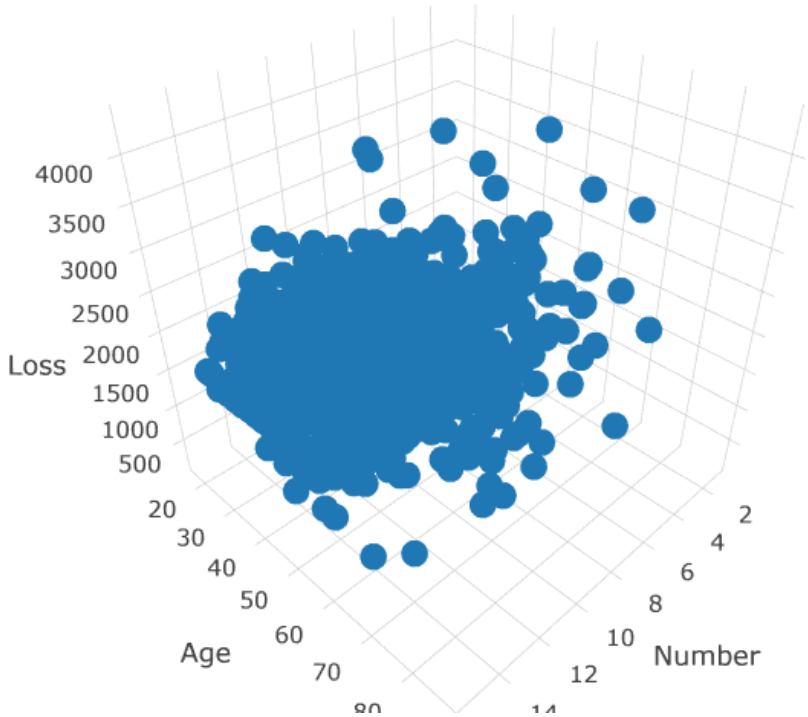

```
#trellis plot
p1 <- ggplot(my_data, aes(age, group=Income, fill = Income)) + geom_density(alpha = 0.4) +
      facet_wrap(~status, labeller = "label_both")
p1
```



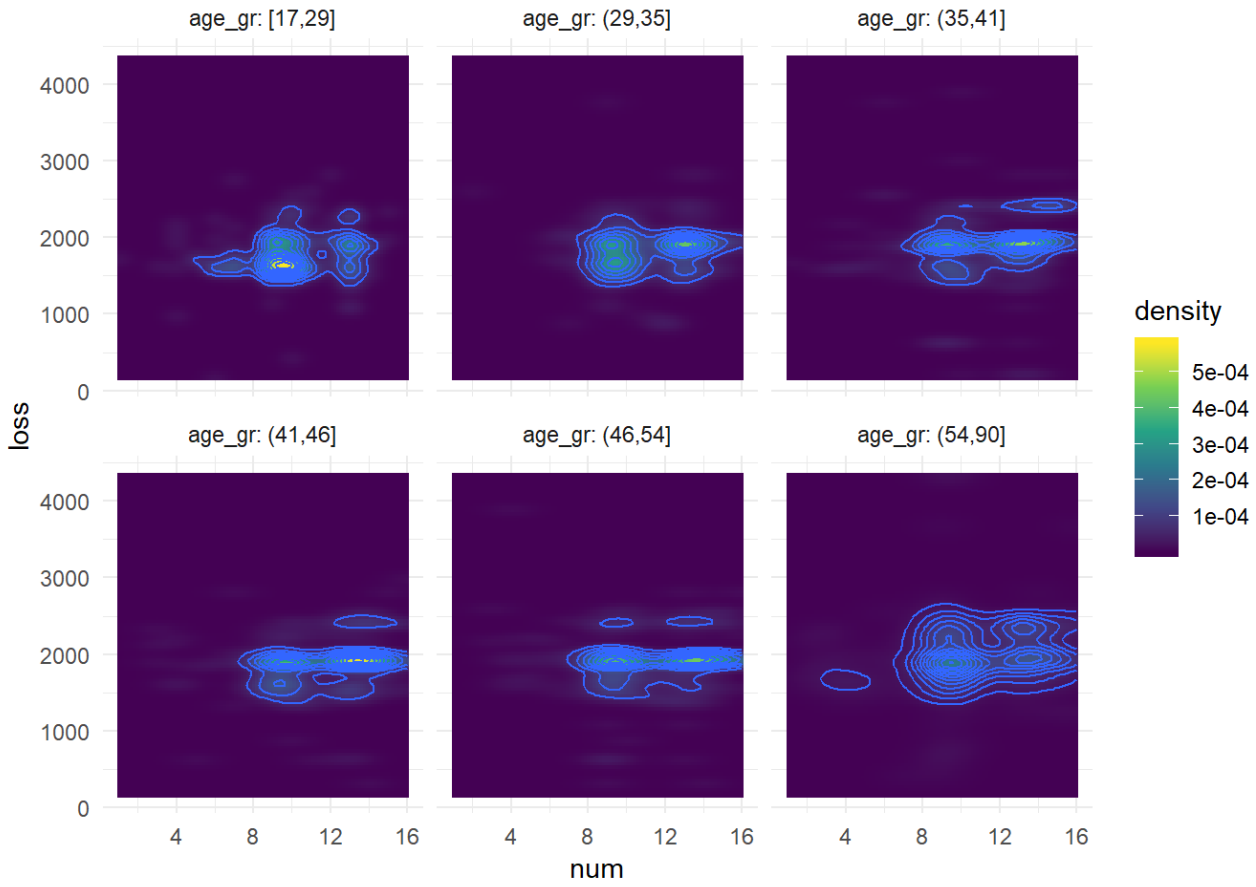
Density plot is used to avoid overplotting as in scatterplot. From the Trellis plot we can depict that the income is not affecting the marital status of the people. Divorced people have been working from the age of 25-70 and most of them have an income lesser than 50k. Married-AF-spouse, separated and Widowed people match both income groups. Married-spouse-absent at the age of 47 have an income level of greater than 50k. More number of Never-married people at the age of about 19 tp 20 have an income level lesser than 50k. Widowed people are working more in comaparision to other age groups.

3. Filter out all observations having Capital loss equal to zero. For the remaining data, use Plotly to create a 3D-scatter plot of Education-num vs Age vs Captial Loss. Why is it difficult to analyze this plot? Create a trellis plot with 6 panels in ggplot2 in which each panel shows a raster-type 2d-density plot of Capital Loss versus Education-num conditioned on values of Age (use cut_number()) . Analyze this plot.

```
library(plotly)

data1 <- subset(my_data, my_data$loss > 0)

#3D scatterplot
p <- plot_ly(data1, x = ~num, y = ~age, z = ~loss) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Number'),
                      yaxis = list(title = 'Age'),
                      zaxis = list(title = 'Loss')))

p
```




```
#trellis plot with raster type 2d density plot
data1 = my_data[-which(my_data$loss == 0), ]
data1$age_gr = cut_number(data1$age, n = 6)

p1 <- ggplot(data=data1, aes(num, loss)) + stat_density2d(aes(fill = stat(density)), geom = "raster",
      contour = FALSE) + geom_density_2d(aes(num, loss), data=data1) + theme_minimal() +
      facet_wrap(~age_gr, labeller = "label_both") + scale_fill_viridis_c()

p1
```

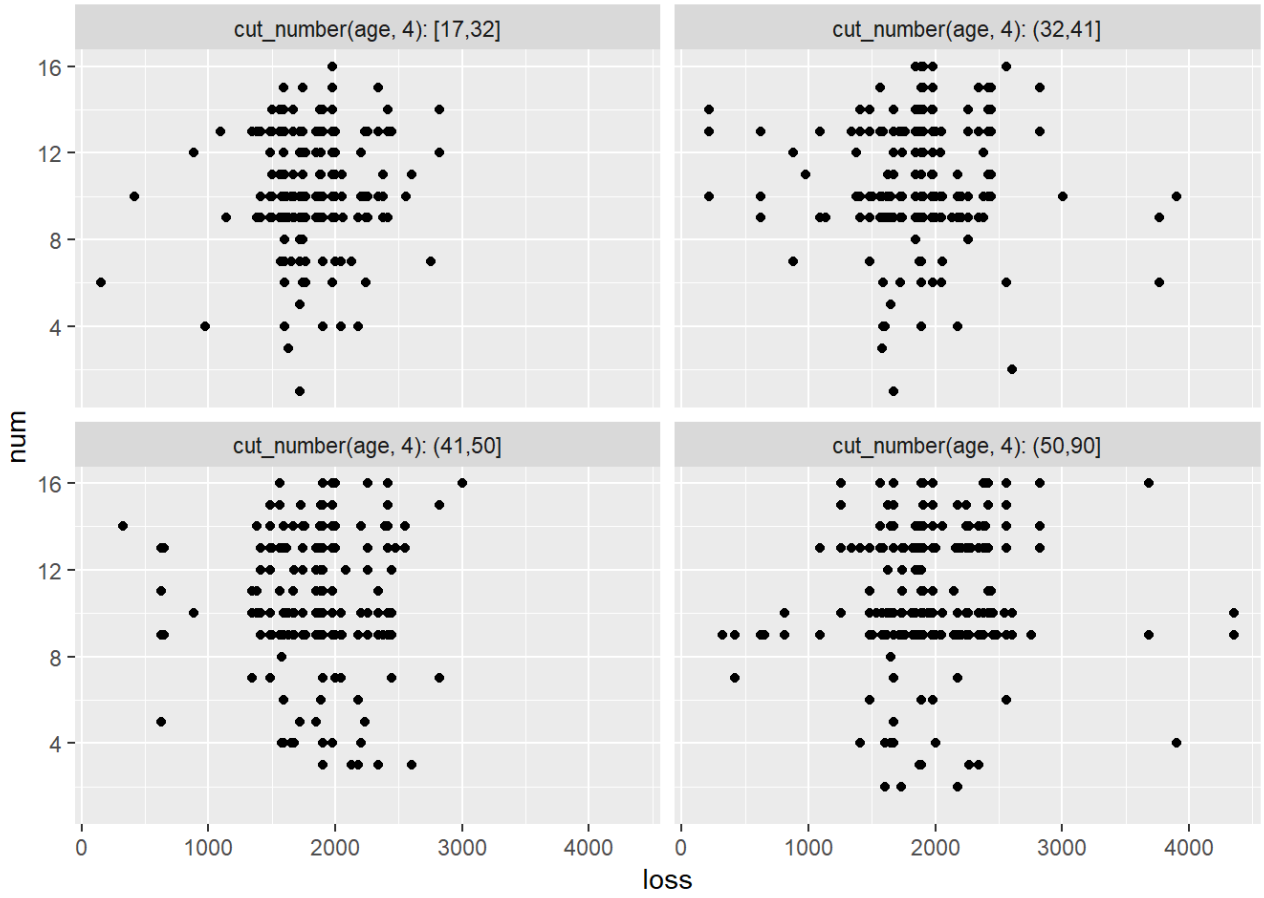


The first plot depicts a 3D scatterplot, and it seems to be overplotted. Hence to avoid over plotting in a scatterplot we use 2D density plot in the second plot.

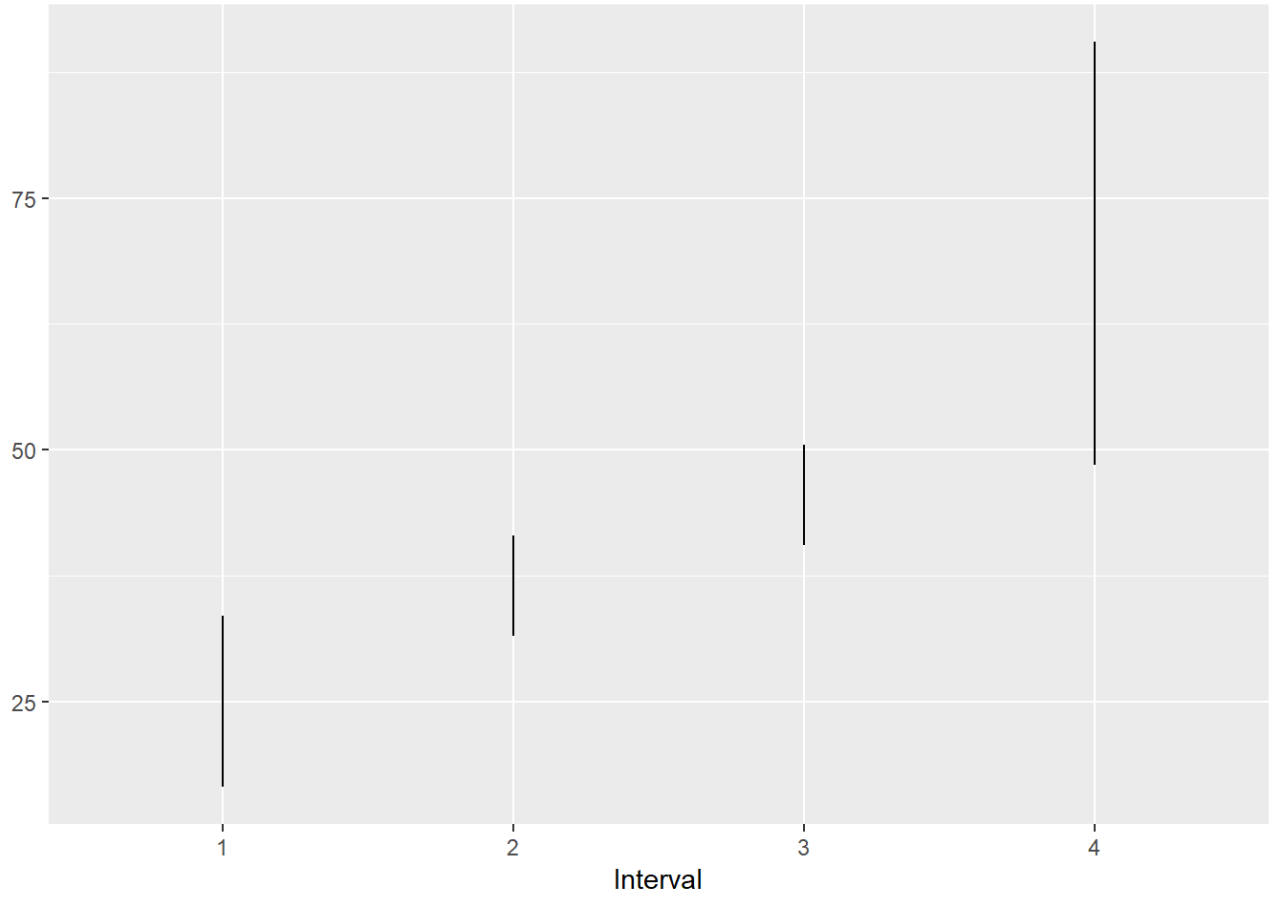
The trellis plot shows that loss is more for those who have taken 8-12 years of education and they belong to 17-29 years age group. For other age groups from 29-54, loss is more common among people who have taken 8-15 years of eucation. Raster plot does not have overlapping of points hence, making it easy for analysis when compared to the scatter plot.

4. Make a trellis plot containing 4 panels where each panel should show a scatter plot of Capital Loss versus Education-num conditioned on the values of Age by a) using cut_number() b) using Shingles with 10% overlap. Which advantages and disadvantages you see in using Shingles?

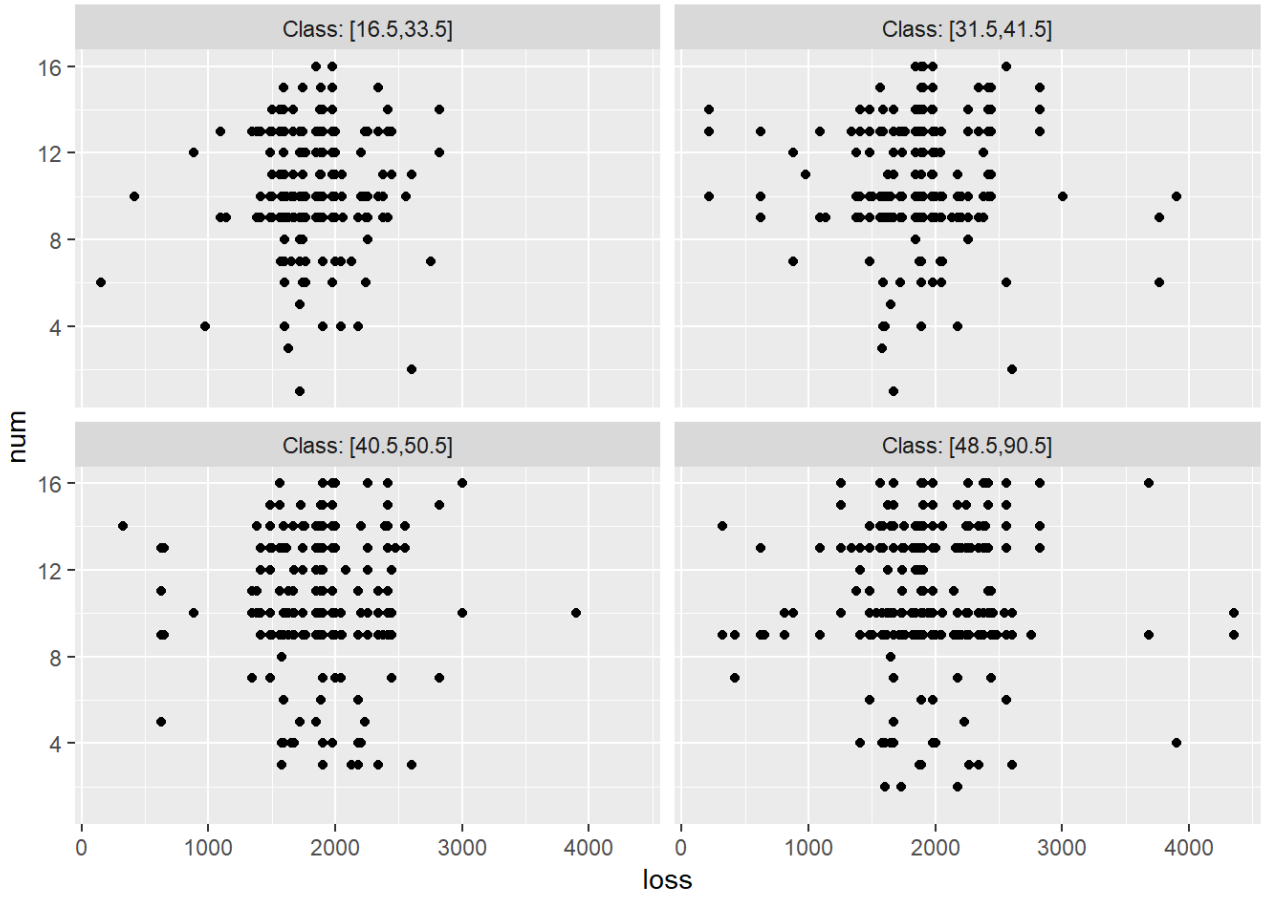
```
#4a. Trellis plot showing a scatterplot using cut_number
a <- ggplot(data1, aes(loss, num)) + geom_point() +
  facet_wrap(~cut_number(age,4), labeller = "label_both")
a
```




```
#4b. Trellis plot showing a scatterplot using shingles
library(lattice)
data1$shingle <- equal.count(data1$age, 4, overlap = 0.10)

M <- matrix(unlist(levels(data1$shingle)), ncol=2, byrow = T)

M1<-data.frame(Lower=M[,1],Upper=M[,2], Interval=factor(1:nrow(M)))

ggplot(M1)+geom_linerange(aes(ymin = Lower, ymax = Upper, x=Interval))
```




```
index=c()
Class=c()
for(i in 1:nrow(M)){
  Cl=paste("[", M1$Lower[i], ",", M1$Upper[i], "]", sep="")
  ind=which(data1$age>=M1$Lower[i] & data1$age<=M1$Upper[i])
  index=c(index,ind)
  Class=c(Class, rep(Cl, length(ind)))
}


data<-data1[index,]
data$Class<-as.factor(Class)

plot <- ggplot(data, aes(x=loss, y=num))+ geom_point()+
     facet_wrap(~Class, labeller = "label_both")
plot
```



Shingle is used to handle real valued variables as it creates overlap, and also helps in avoiding Boundary Effect.