

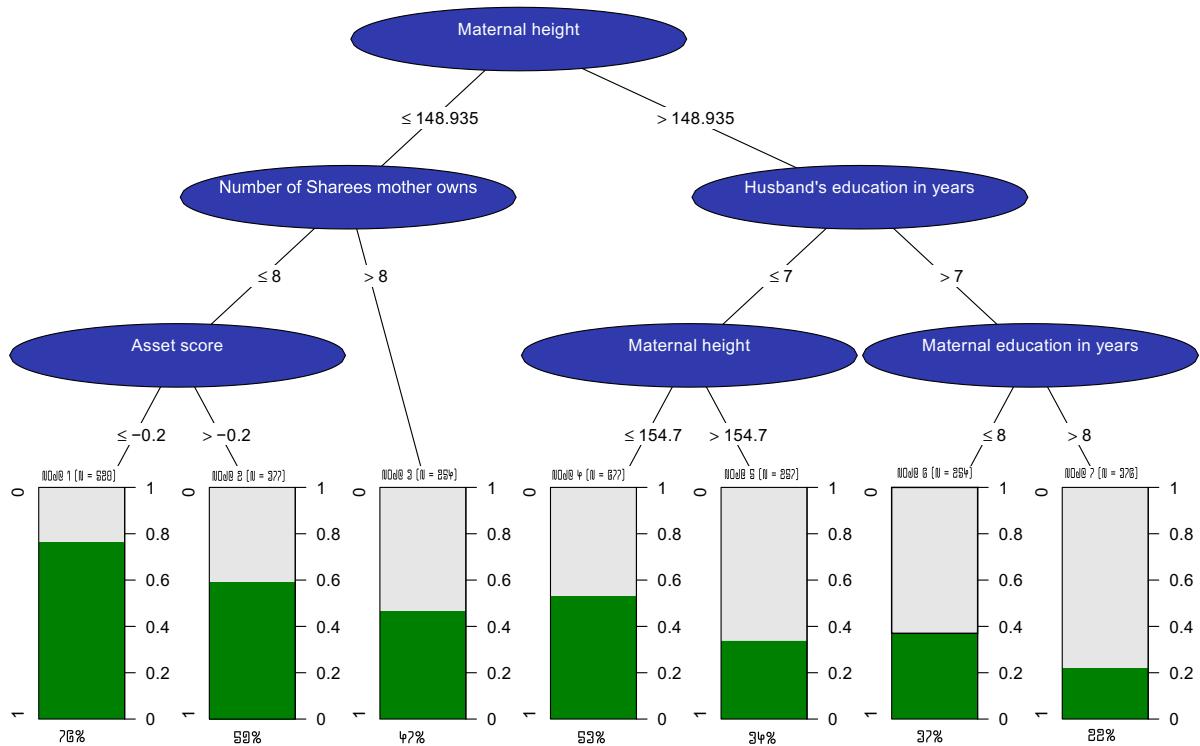
Lab1

Priya Kurian(priku577) and Andreas christopoulos(andch552)

17 September 2018

Assignment 1

Made the required changes to the tree.pdf The final one is tree_fixed.svg



Assignment 2

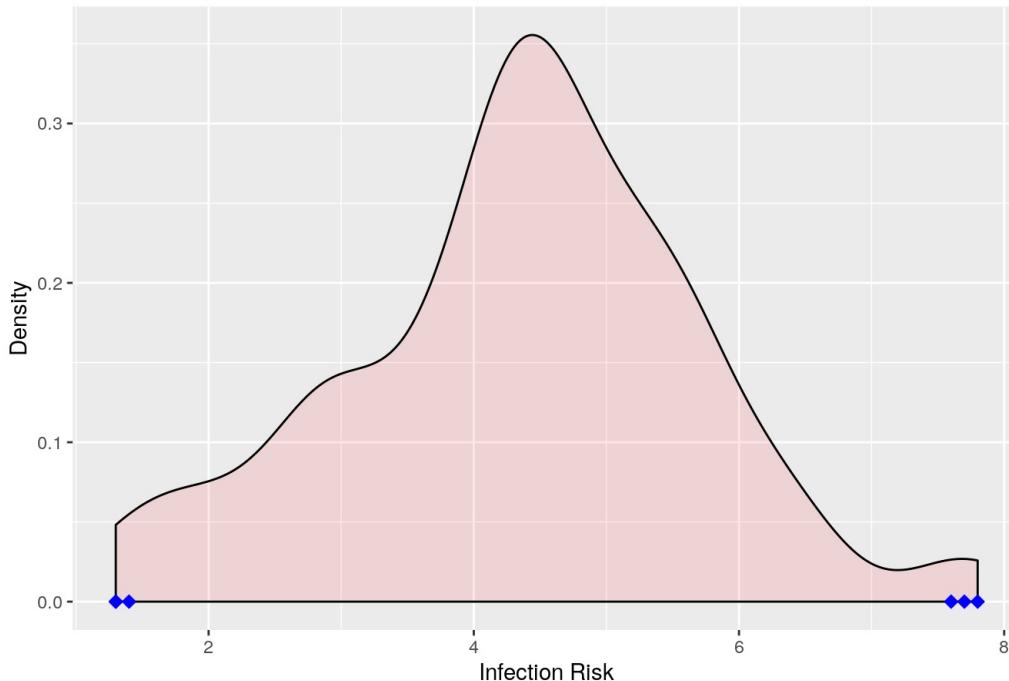
Read Data from file and load libraries

creating function to return indices

```
# Assignment 1 -----  
  
# 1 -----  
  
senic=read.table("SENIC.txt")  
  
# 2 -----  
  
outlier_function=function(X){  
  q1=as.numeric(quantile(X,0.25))  
  q3=as.numeric(quantile(X,0.75))  
  greater=q3+1.5*(q3-q1)  
  smaller=q1-1.5*(q3-q1)  
  indices=which(X>greater | X<smaller)  
  return(indices)  
}
```

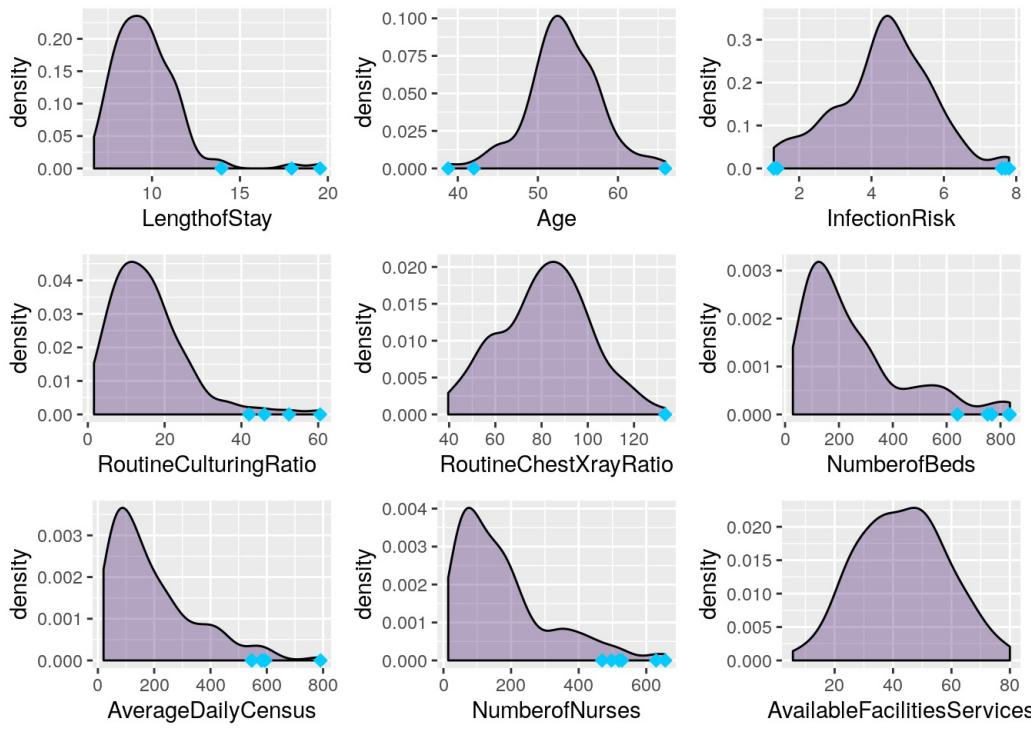
Density Plotting of Infection Risk

Density plot of infection risk with outliers



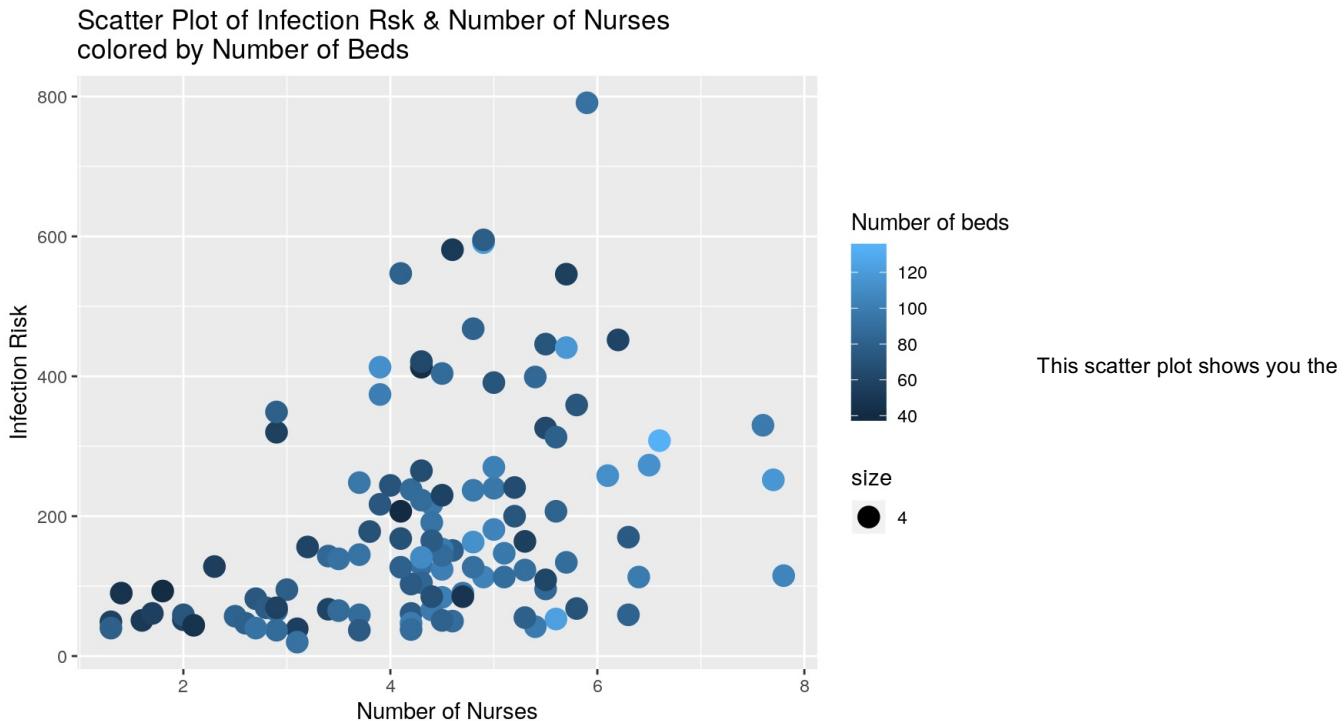
The density of infection rate has 5 outliers.the outliers are on either side of the graph. the probability of infection risk lies in a band of 2-6 and there is a very minimal probability for infection risk to cross the above mentioned limit.

Producce graphs for all other variables



We could see that all graphs except AvailableFacilitiesServices have outliers. We feel that the graphs of Age, InfectionRisk, RoutineChestXRayRatio, AvailableFacilitiesServices follow a normal distribution. Also the LengthofStay, RoutineCulturingRatio, NumberofBeds, AverageDailyCensus, NumberofNurses follow a chi-square distribution. The outliers skews the data.

Scatter Plot



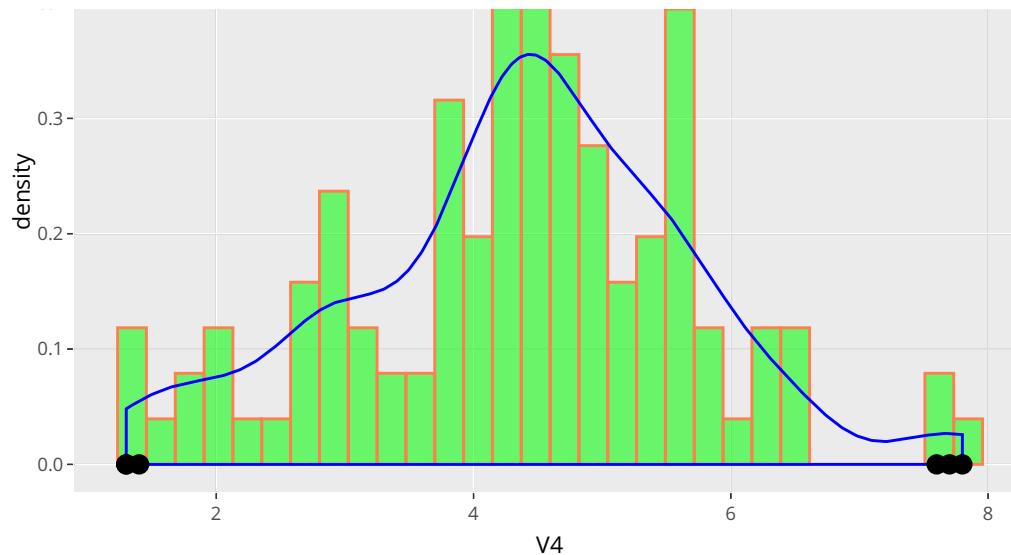
dependence of Infection Risk on the Number of Nurses where the points are colored by Number of Beds. Here the graph shows number of nurses and number of bed increases the infection rate. More number of beds means more number of patients can be accommodated. As the patients increases the infections also increases as the patients coming to hospital will be sick. The color scale itself is confusing as we usually think the light colours give a smaller value and dark a higher. Since the colours are very similar it is difficult to differentiate as well.

Plotly Graph

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

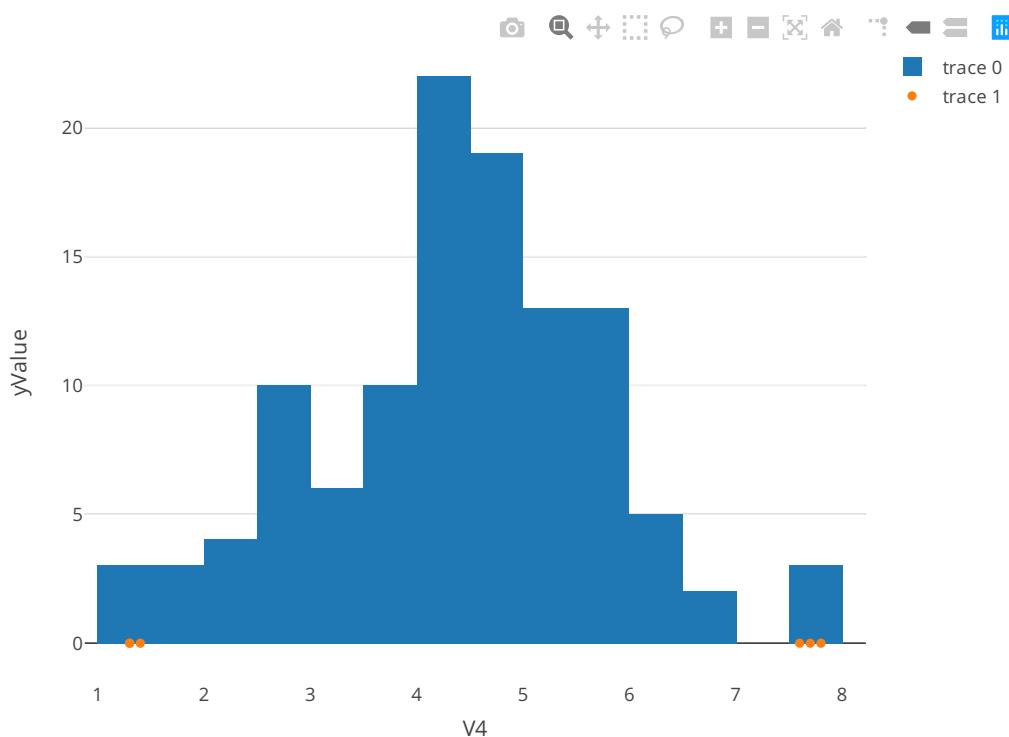
Density with Histogram overlay





The plotly has features like zoom in,zoom out,reset the axis,autoscaling,box slection,lassso selection etc.As a whole we feel like there are lot more that we can do with plotly graphs when compared to ggplot.

Plotly Plot Made with Pipe operator



Shiny App

```

library(shiny)
library(gridExtra)
df<-read.table("SENIC.txt")
quantile_function<-function(dat)
{
  Q1<-quantile(dat,0.25)
  Q3<-quantile(dat,0.75)
  greater<-Q3+1.5*(Q3-Q1)
  smaller<-Q1-1.5*(Q3-Q1)
  result<-subset(dat,dat > greater | dat < smaller)
  indices<-which(dat %in% result)
  return(result)
}

# Define UI for application that draws a histogram

ui <- fluidPage(

  # Application title
  titlePanel("Density Plots"),

  # Sidebar with checkbox of variables
  sidebarLayout(
    sidebarPanel(
      checkboxGroupInput(inputId = "input_value",
                         label = "Select Variables:",
                         choiceValues = paste("V", c(2:7,10,11), sep = ""),
                         choiceNames = c("LengthofStay", "Age", "InfectionRisk",
                                        "RoutineCulturingRatio", "Routine Chest X-ray Ratio",
                                        "Number of Beds", "Average Daily Census",
                                        "Number of Nurses"),
                         selected = "V4"),
      sliderInput(inputId = "bw_value", label = "Bandwidth", min = 0, max = 20, value = 1, step = .1)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("output_val")
    )
  )
)

server <- function(input, output){

  output$output_val <- renderPlot({
    # Render plot
    p <- lapply(input$input_value, function(i){ggplot() +
      geom_density(aes(df[,i]), fill = "green", bw = input$bw_value) +
      geom_point(aes(x = df[quantile_function(df[,i]),i], y = 0), shape = 23, col = "black", fill = "black") +
      labs(x = "") +
      theme_minimal()})
    if(length(p) <= 4){
      n_col <- length(p)
      n_row = 1} else{
      n_col <- 4
      n_row <- 2
    }
    marrangeGrob(p, ncol = n_col, nrow = n_row)
  })
}

# Run the application
shinyApp(ui = ui, server = server)

```

Seems like a bandwidth of 4 is a good choice.

Appendix

```
## ----echo=FALSE-----
knitr::opts_chunk$set(echo = F)
```

```

## ----message=FALSE-----

# libraries -----
library(ggplot2)
library(gridExtra)
library(plotly)
library(shiny)
library(gridExtra)
library(tidyverse)

## ----echo=TRUE-----

# Assignment 1 ----

# 1 -----
senic=read.table("SENIC.txt")

# 2 -----
outlier_function=function(X){
  q1=as.numeric(quantile(X,0.25))
  q3=as.numeric(quantile(X,0.75))
  greater=q3+1.5*(q3-q1)
  smaller=q1-1.5*(q3-q1)
  indices=which(X>greater | X<smaller)
  return(indices)
}

## -----
# 3-----

outliersInf=outlier_function(senic$V4)
infRisk=data.frame("outliers"=senic$V4[outliersInf],"rep"=0)

datfrm<-data.frame(senic$V4)
names(datfrm)<-c('InfectionRisk')
ggplot_value<-ggplot(datfrm,aes(x=InfectionRisk))+geom_density(color="black",fill="red",alpha=0.1)+geom_point(data=infRisk,aes(x=outliers,y=rep),shape=23,color="blue",size=2,fill="blue")+
  labs(x="Infection Risk",y="Density")+ggtitle("Density plot of infection risk with outliers")
ggplot_value

## -----
# 4 -----
dt<-senic[c(-1,-8,-9)]
names(dt)<-c('LengthofStay','Age','InfectionRisk','RoutineCulturingRatio',
  'RoutineChestXrayRatio','NumberofBeds','AverageDailyCensus',
  'NumberofNurses','AvailableFacilitiesServices')

my_plots=list()

for(name in names(dt)){
  indices=outlier_function(dt[[name]])
  outliers=dt[[name]][indices]
  outliers_names=tibble("x"=outliers,"y"=0)
  my_plots[[name]]=ggplot(dt, aes_string(x =name)) + geom_density(color="black",fill="#330066",alpha=0.3)+geom_point(data=outliers_names,aes(x,y),size=2,color="#00CCFF",pch=23,fill="#00CCFF")
}
grid.arrange(grobs=my_plots)

## -----
# 5 -----
ggplot(senic,aes(V4,V10))+geom_point(aes(size=4,color=V6))+ggtitle("Scatter Plot of Infection Rsk & Number of Nurses \n colored by Number of Beds")+
  labs(x="Number of Nurses",y="Infection Risk")+
  scale_colour_continuous("Number of beds")

## -----
# 6 -----

```

```

indices_infection=outlier_function(senic$V4)
outliers_infection=senic$V4[indices_infection]
outliers = tibble(x = outliers_infection, y = 0)

plot<-ggplot(senic,aes(V4))+geom_histogram(aes(y=..density..,alpha=0.7),col="coral",fill="green")+
  geom_density(col="blue")+ggtitle("Density with Histogram overlay")+
  geom_point(data = outliers, aes(x,y),size=3)

ggplotly(plot)

# using the plot from 3
#p<-ggplot_value+geom_histogram(aes(y=..density..,alpha=0.5),col="gray",fill="blue",alpha=0.5)
#ggplotly(p)

## -----
# 7 -----

Outlier_indices= outlier_function(senic$V4)
Outlier_values<-senic$V4[Outlier_indices]
yValue <- rep(0,length(Outlier_values))

hisPlot <- senic %>% select(V4) %>% plot_ly(x=~V4,type="histogram") %>%
  add_markers(x=~Outlier_values, y=~yValue)

hisPlot

## ----eval=FALSE,echo=T-----
## 
## library(shiny)
## library(gridExtra)
## df<-read.table("SENIC.txt")
## quantile_function<-function(dat)
## {
##   Q1<-quantile(dat,0.25)
##   Q3<-quantile(dat,0.75)
##   greater<-Q3+1.5*(Q3-Q1)
##   smaller<-Q1-1.5*(Q3-Q1)
##   result<-subset(dat,dat > greater | dat < smaller)
##   indices<-which(dat %in% result)
##   return(result)
## }
## 
## 
## # Define UI for application that draws a histogram
## 
## ui <- fluidPage(
## 
##   # Application title
##   titlePanel("Density Plots"),
## 
##   # Sidebar with checkbox of variables
##   sidebarLayout(
##     sidebarPanel(
##       checkboxGroupInput(inputId = "input_value",
##                         label = "Select Variables:",
##                         choiceValues = paste("V", c(2:7,10,11), sep = ""),
##                         choiceNames = c("LengthofStay", "Age", "InfectionRisk",
##                                       "RoutineCulturingRatio", "Routine Chest X-ray Ratio",
##                                       "Number of Beds", "Average Daily Census",
##                                       "Number of Nurses"),
##                         selected = "V4"),
## 
##       sliderInput(inputId = "bw_value", label = "Bandwidth", min = 0, max = 20, value = 1, step = .1)
##     ),
## 
##     # Show a plot of the generated distribution
##     mainPanel(
##       plotOutput("output_val")
##     )
##   )
## }

## server <- function(input, output){

```

```

## 
##   output$output_val <- renderPlot({
##     # Render plot
##     p <- lapply(input$input_value, function(i){ggplot() +
##       geom_density(aes(df[,i]), fill = "green", bw = input$bw_value) +
##       geom_point(aes(x = df[quantile_function(df[,i]),i], y = 0), shape = 23, col = "black", fill = "black")
##     +
##       labs(x = "") +
##       theme_minimal()})
## 
##     if(length(p) <= 4){
##       n_col <- length(p)
##       n_row = 1} else{
##         n_col <- 4
##         n_row <- 2
##       }
## 
##     marrangeGrob(p, ncol = n_col, nrow = n_row)
##   })
## }

## # Run the application
## shinyApp(ui = ui, server = server)
## 
## 
## 

## ----code = readLines(knitr::purl("~/Courses/Visualization_Group_labs/Group25_Lab1/Group25_lab1.Rmd",documentation = 1)), echo = T, eval = F----
## NA

```

Lab2

Priya(priku577) and Andreas(andch552)

19 September 2018

###Assignment 1

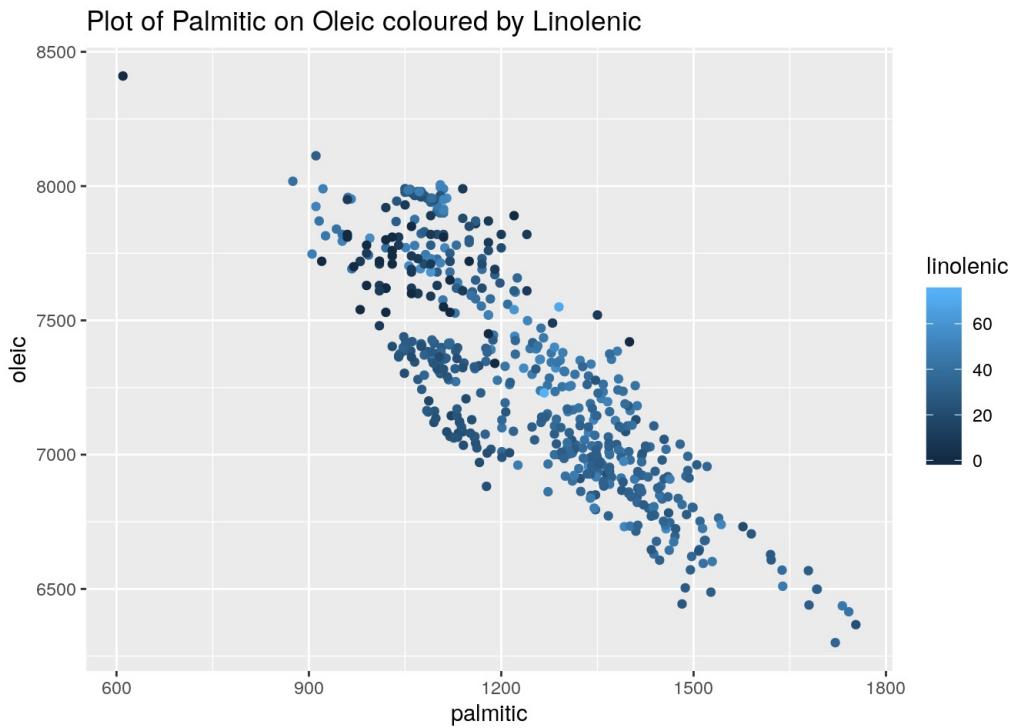
##1

scatterplot in Ggplot2 that shows dependence of Palmitic on Oleic in which observations are colored by Linolenic

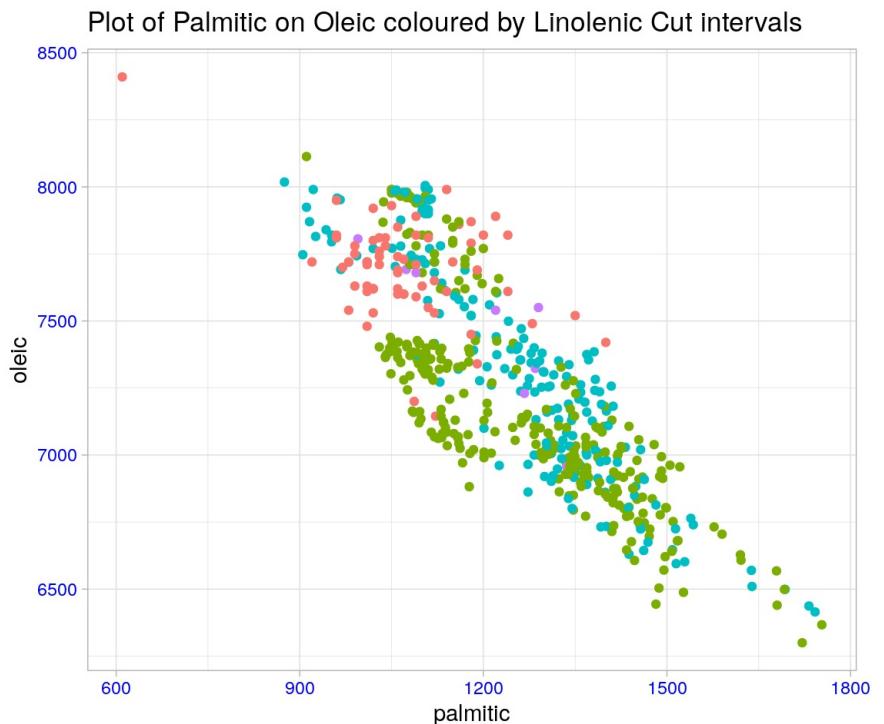
```
library(ggplot2)
library(plotly)
library(MASS)

olive<-read.csv("olive.csv",header=TRUE, sep=",")

p1<-ggplot(olive,aes(palmitic,oleic,col=linolenic))+geom_point()+labs(x="palmitic",y="oleic")+ggtitle("Plot of Palmitic on Oleic coloured by Linolenic")
p1
```



```
interval_cut <- cut_interval(olive$linolenic, 4)
p2<-ggplot(olive,aes(palmitic,oleic,col=interval_cut))+geom_point()+theme_light()+labs(x="palmitic",y="oleic")+ggtitle("Plot of Palmitic on Oleic coloured by Linolenic Cut intervals")+theme(axis.text = element_text(colour = "blue"))
p2
```



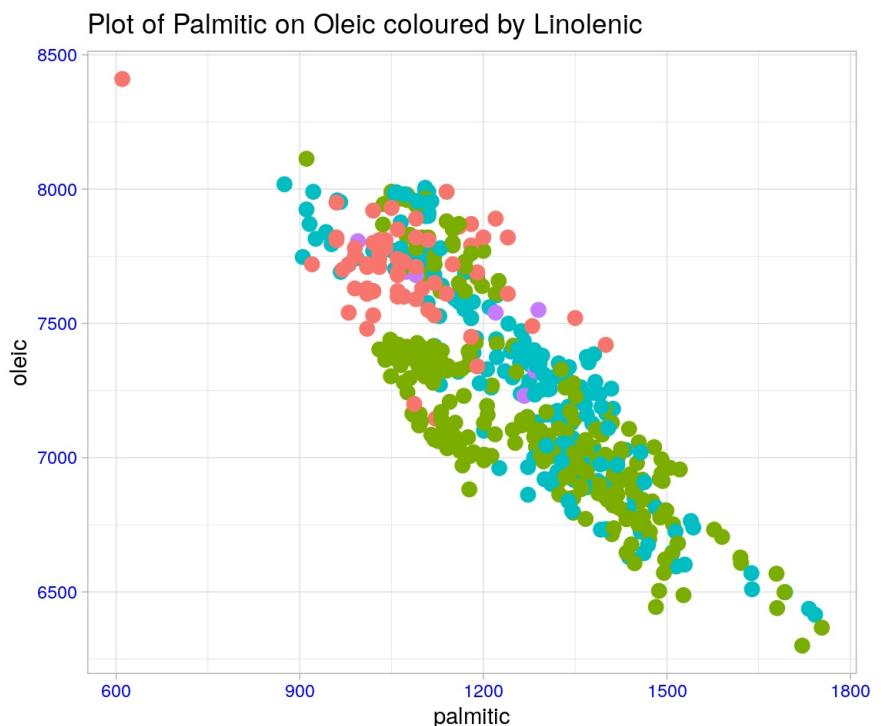
Channel capacity of the human perception is the perception problem told here. The perception consists of three steps: recognizing, organizing and interpreting. Grouping is hard to do in the first plot as different shades of same hue are used. The groups can be easily identified by perception in the second plot. Its difficult to distinguish between different shades of same hue while its easier to distinguish between different hues (channel capacity is here).

##2

scatterplots of Palmitic vs Oleic in which you map the discretized Linolenic with four classes to:

a. Color

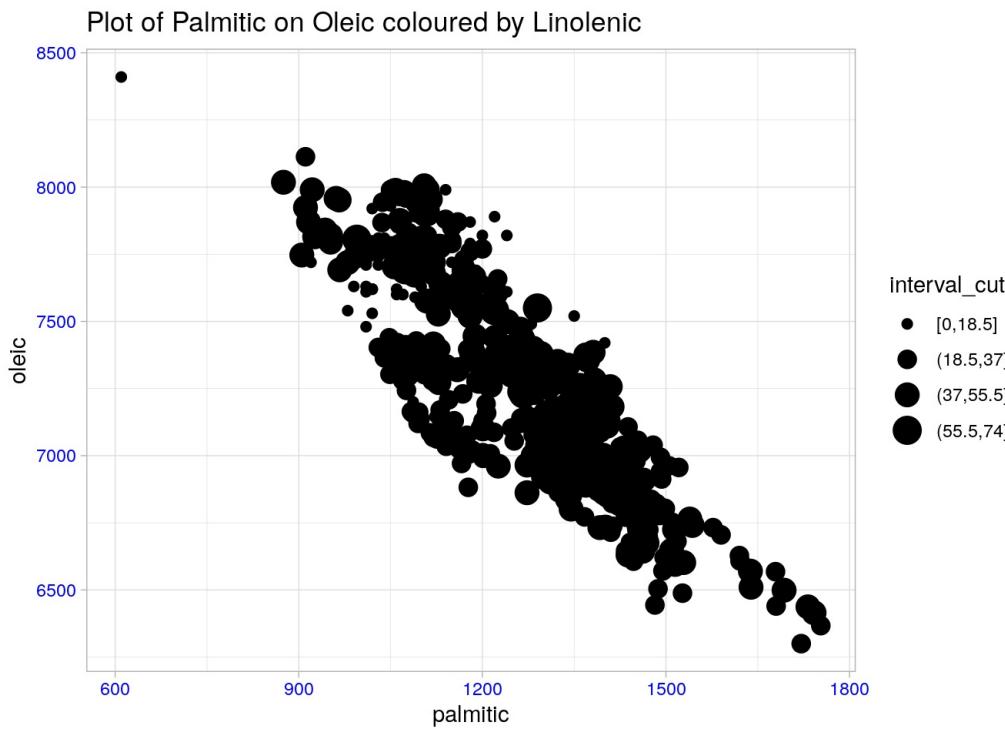
```
p3<-s2<-ggplot(olive,aes(palmitic,oleic,col=interval_cut))+geom_point()+theme_light()+labs(x="palmitic",y="oleic")  
+geom_point(size=3)+ggtitle("Plot of Palmitic on Oleic coloured by Linolenic")+theme(axis.text = element_text(color = "blue"))  
p3
```



b. Size

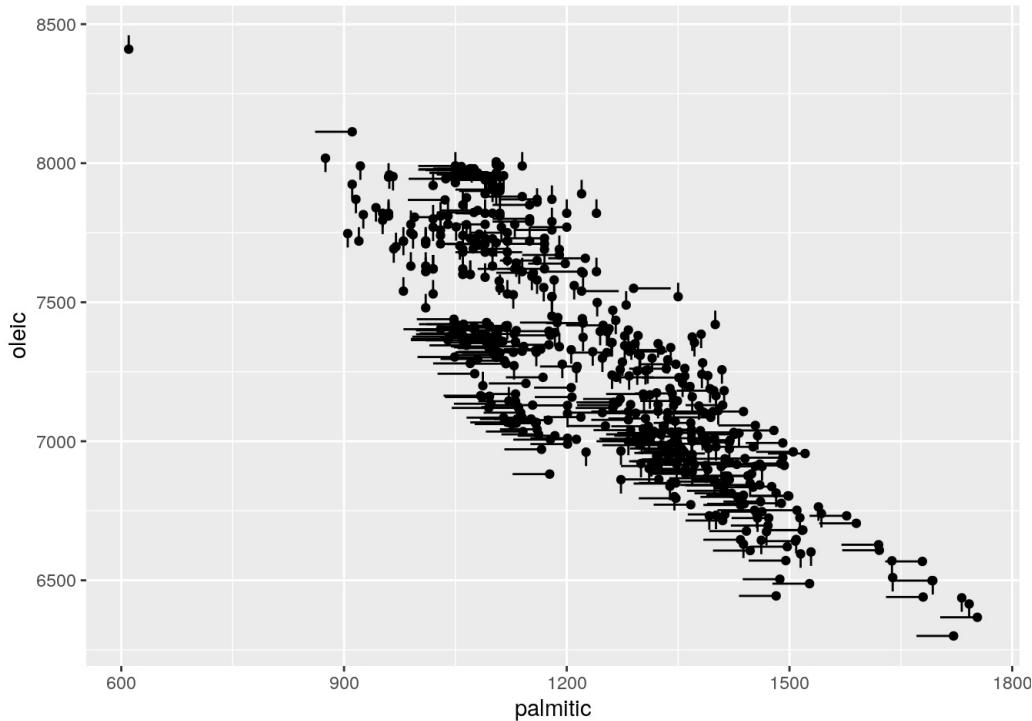
```
p4<-s2<-ggplot(olive,aes(palmitic,oleic))+geom_point(aes(size=interval_cut))+theme_light()+labs(x="palmitic",y="oleic")  
+geom_point()+ggtitle("Plot of Palmitic on Oleic coloured by Linolenic")+theme(axis.text = element_text(color = "blue"))  
p4
```

```
## Warning: Using size for a discrete variable is not advised.
```



c. Orientation angle

```
angleplot<-ggplot(olive,aes(x=palmitic,y=oleic)) +  
  geom_point() +geom_spoke(aes(angle = as.numeric(cut_interval(linolenic, 4))*pi/2), radius = 50)  
angleplot
```



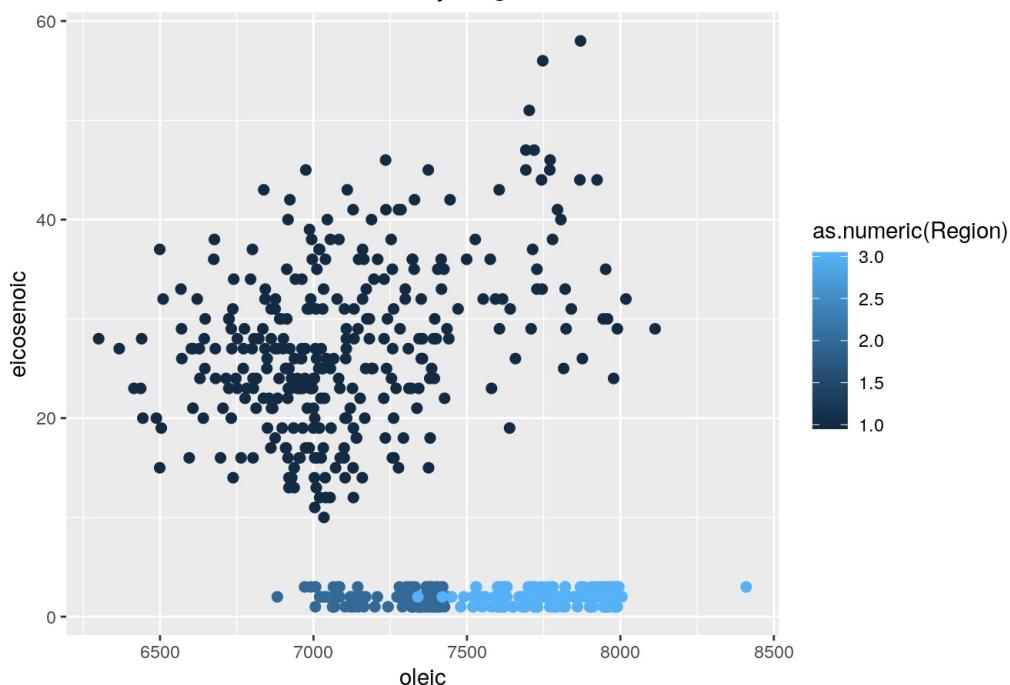
We feel it is very difficult to differentiate between categories in the second plot with size (b.size).The third plot is not good either.The first graph is obviously the best. The coloured graph is 3.1 bits the size plot 2.2 bits and the angle plot is 2.8 bits

#3

Create a scatterplot of Oleic vs Eicosenoic in which color is defined by numeric values of Region.

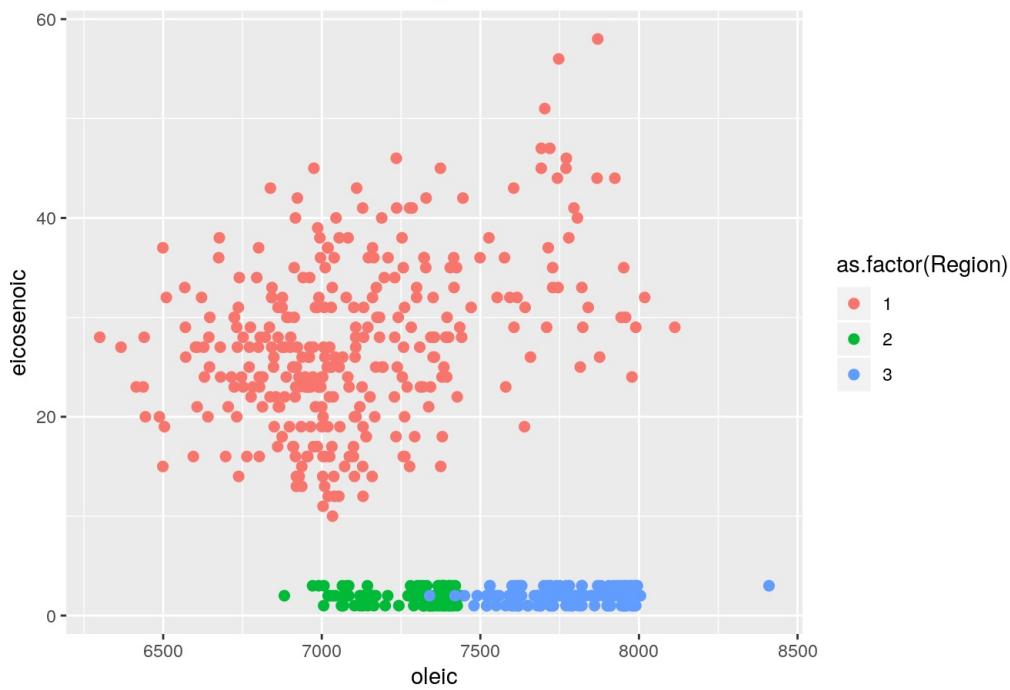
```
p6<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=as.numeric(Region)))+labs(x="oleic",y="eicosenoic")+geom_point(size=2)+ggtitle("oleic on eicosenoic coloured by Region")  
p6
```

oleic on eicosenoic coloured by Region



```
p7<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=as.factor(Region)))+labs(x="oleic",y="eicosenoic")+geom_point(size=2)+ggtitle("oleic on eicosenoic coloured by Region")
p7
```

oleic on eicosenoic coloured by Region



In the second plot it is very easy to find the decision boundaries as using discrete hues is a preattentive feature and identifying the boundaries take very less time.

##4

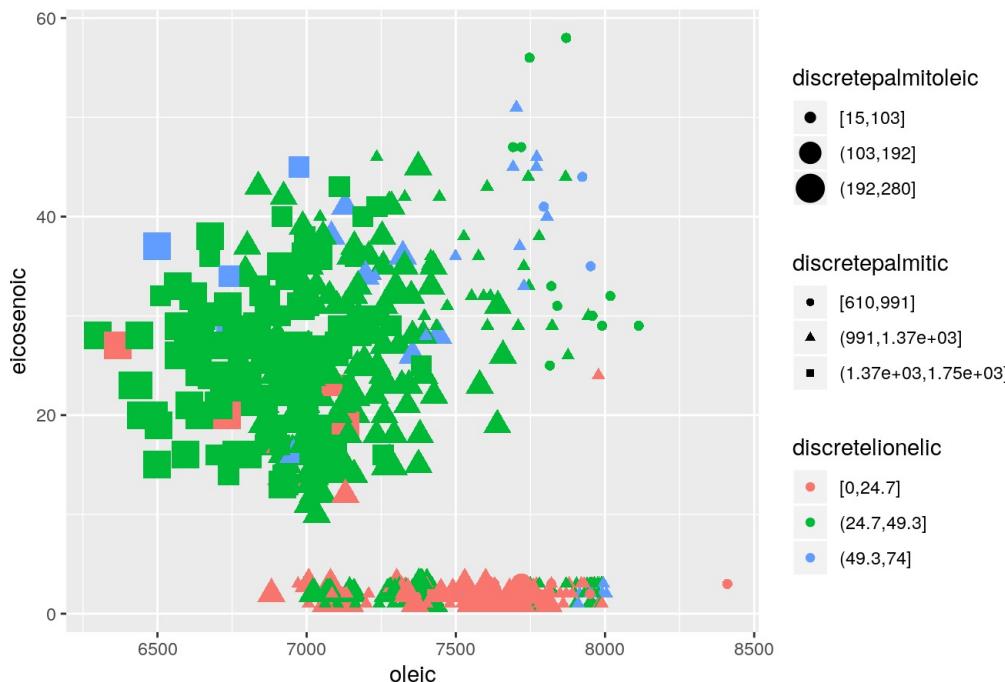
scatterplot of Oleic vs Eicosenoic in which color is defined by a discretized Linoleic (3 classes), shape is defined by a discretized Palmitic (3 classes) and size is defined by a discretized Palmitoleic (3 classes)

```
discretelionelic<-cut_interval(olive$linolenic, 3)
discretepalmitic<-cut_interval(olive$palmitic, 3)
discretepalmitoleic<-cut_interval(olive$palmitoleic, 3)
```

```
p8<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=discretelionelic,shape=discretepalmitic,size=discretepalmitoleic))
+geom_point()+ggtitle("oleic vs eicosenoic with linolenic, palmitic,palmitoleic")
p8
```

Warning: Using size for a discrete variable is not advised.

oleic vs eicosenoic with linolenic, palmitic,palmitoleic



It is really difficult to differentiate between 27 different types of observations as there are no clear cut boundaries. Here there are three shades of colour and three types of shapes. Their summation is not the same as summation of the channel capacity of three shapes and three colours.

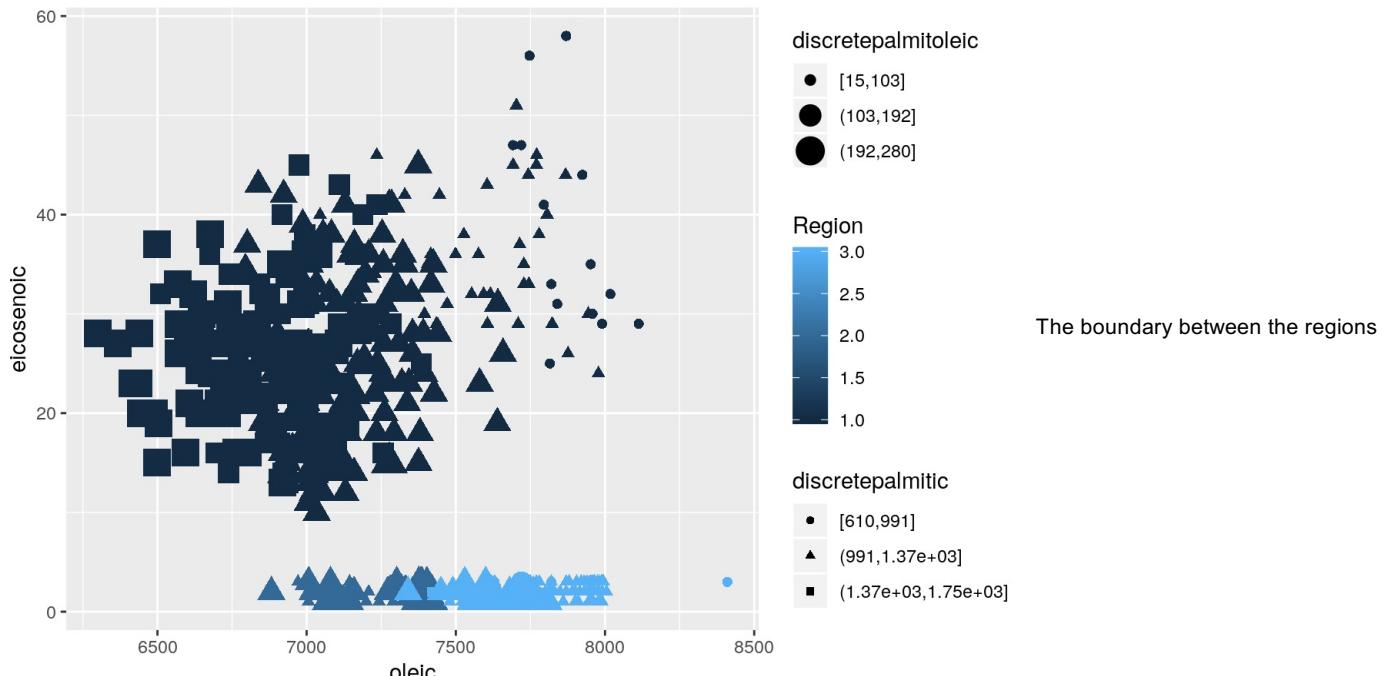
##5

scatterplot of Oleic vs Eicosenoic in which color is defined by Region,shape is defined by a discretized Palmitic (3 classes) and size is defined by a discretized Palmitoleic (3 classes)

```
p9<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=Region,shape=discretepalmitic,size=discretepalmitoleic))+geom_point()
t() + ggtitle("oleic vs eicosenoic with linolenic, palmitic,palmitoleic")
p9
```

Warning: Using size for a discrete variable is not advised.

oleic vs eicosenoic with linolenic, palmitic,palmitoleic



is easily identified as we have colour to distinguish the boundaries. Colour, orientation and intensity can be automatically perceived by human eye according to treistman theory of integration and the combination of these may require time to distinguish as seen in the question 4. Here colour is the one that helps as distinguish faster.

##6

create a pie chart that shows the proportions of oils coming from different Areas using plotly

```

library(plotrix)
library(plotly)

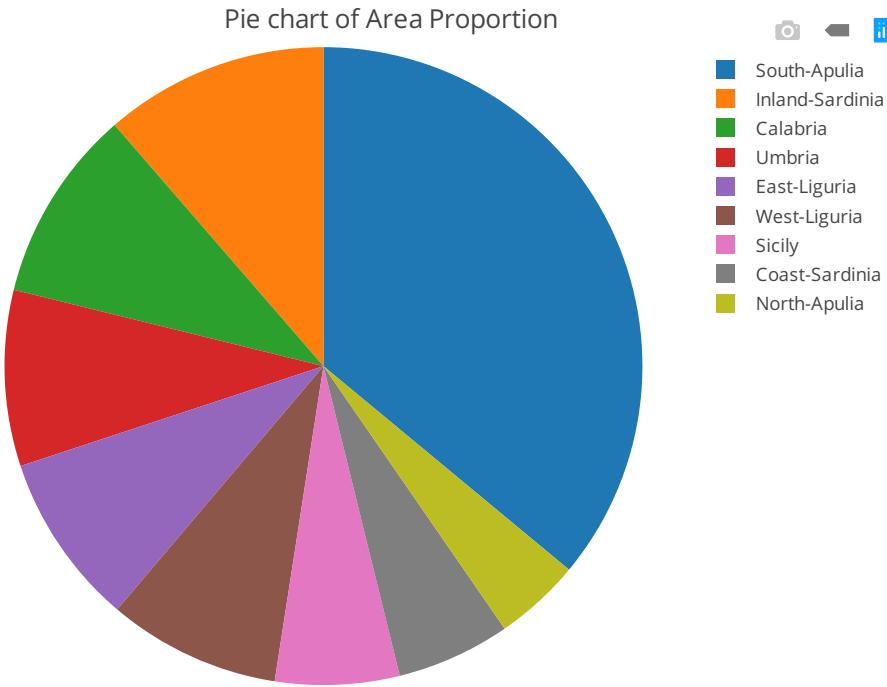
mytable <- table(olive$Area)
dframe<-as.data.frame(mytable)

#pie3D(dframe$Freq,labels=dframe$Var1,explode=0.1,main="Pie Chart of Area Proportion ")

p10 <- plot_ly(dframe, labels = ~Var1, values = ~Freq, type = 'pie',textinfo = "none", hoverinfo = "text",text=~Var1) %>%
  layout(title = 'Pie chart of Area Proportion',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))

p10

```



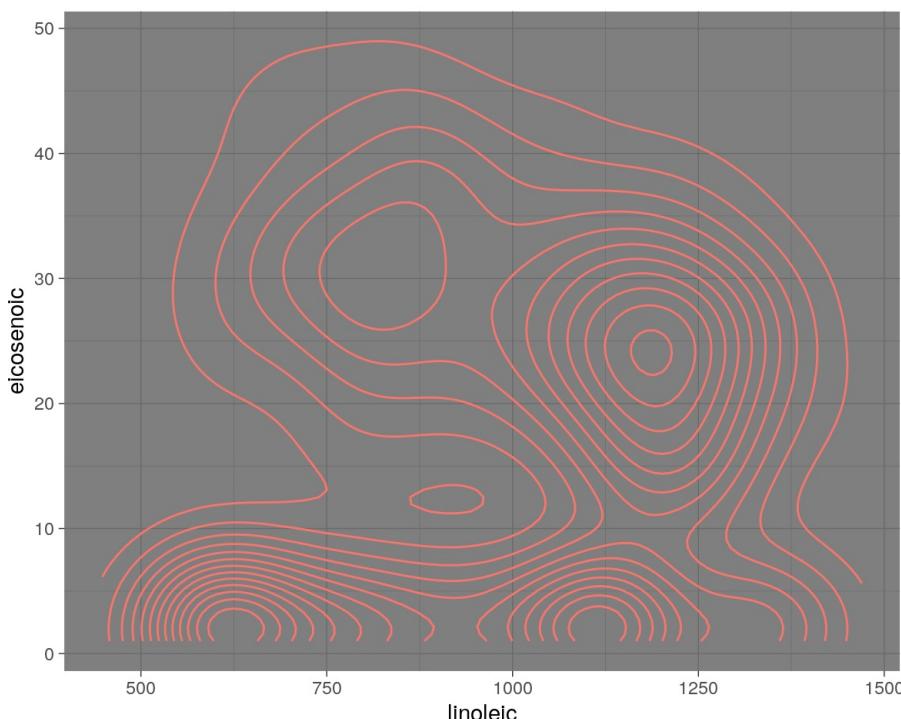
The pie chart is less effective than the bar chart. When the values differ by small number it is difficult to predict which area's frequency was more without hovering and finding for the exact value. In barchart just looking at the length we can find this. ##7

2d-density contour plot

```

p11<-ggplot(olive, aes(x=linoleic, y=eicosenoic,col="red")) +geom_density_2d(position="identity") +theme_dark()
p11

```



Outliers are missed in contour plots.

##Assignment 2

Load the baseball data in r

```
library(readxl)
library(ggplot2)
library(plotly)
library(MASS)

df<-read_xlsx("baseball-2016.xlsx",col_names=TRUE)
```

It is reasonable to scale the data because we have different range of features in the data.

```
library(readxl)

df<-read_xlsx("baseball-2016.xlsx",col_names=TRUE)
df<-data.frame(df)

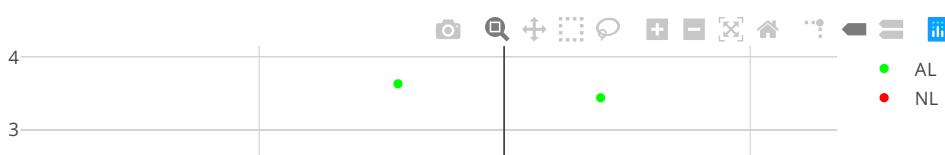
scaled_df<-scale(df[,-c(1,2)])
d<-dist(scaled_df)
mds<-isoMDS(d,k=2)
```

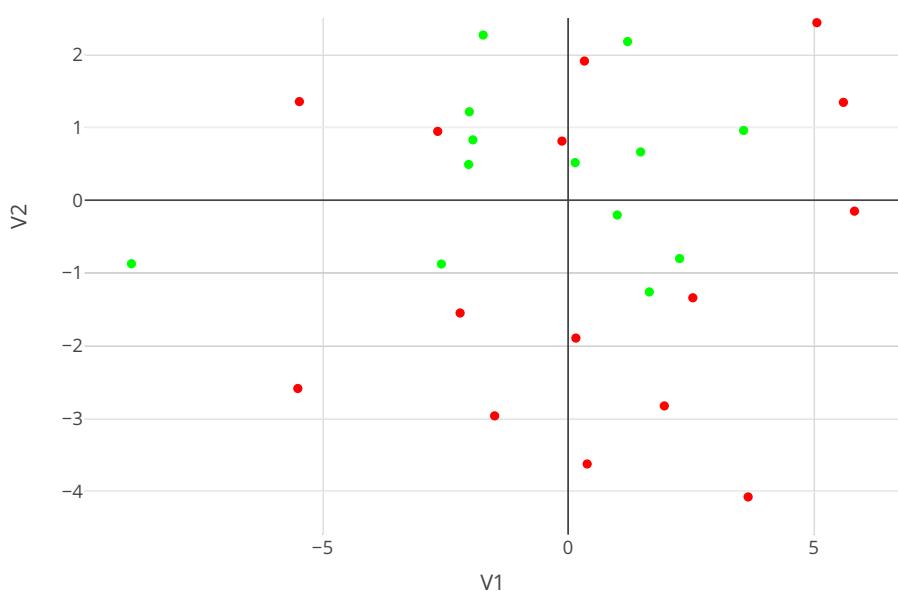
```
## initial value 19.856833
## iter 5 value 16.319153
## iter 10 value 16.046215
## final value 15.935476
## converged
```

```
dt<-(mds$points)
dt_new<-as.data.frame(dt)
dt_new$League<-df[,2]
dt_new$Team<-df[,1]
```

```
plot_ly(dt_new, x=~V1, y=~V2, type="scatter", hovertext=~Team, color=~League, colors=c("green", "red"))
```

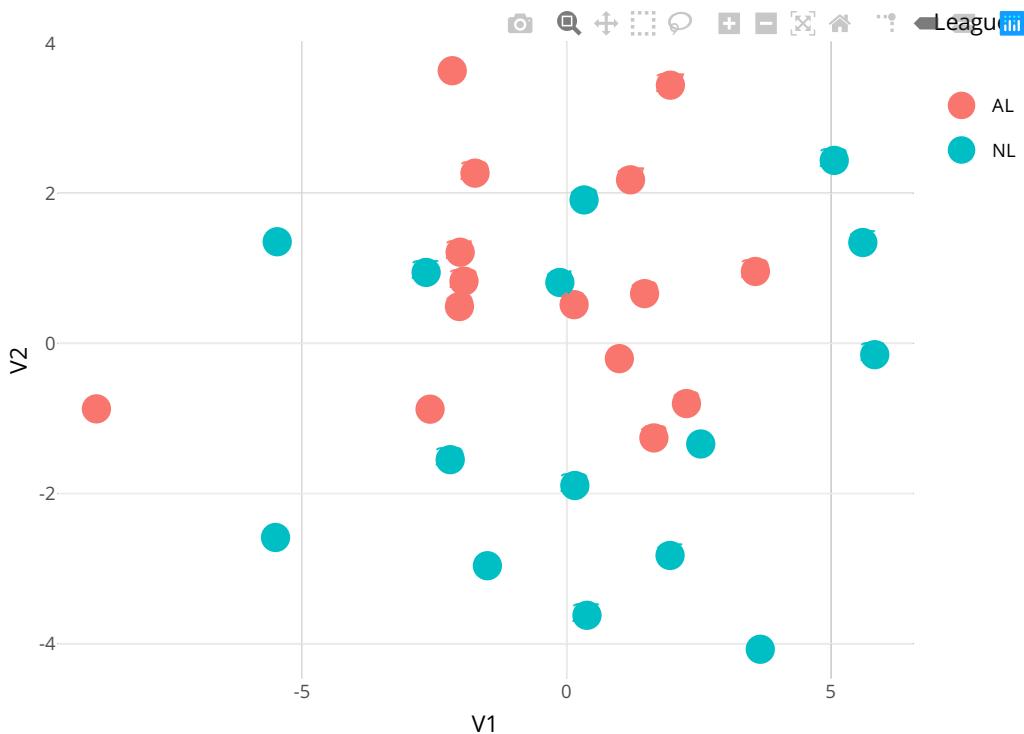
```
## No scatter mode specified:
## Setting the mode to markers
## Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```





From the graph we can first observe that Boston Red Sox, Baltimore Orioles, Philadelphia Phillies seems to be outliers. There's seems to be some kind of difference between the Leagues. V2 seems to provide a better differentiation between leagues. Also the NL is more scattered than AL.

```
p <- ggplot(data=dt_new,aes(x=V1,y=V2,col=League,size=5))+geom_point()+theme_minimal()+
  geom_text(aes(label=rownames(dt_new),size=5))
ggplotly(p)
```



#3

```
library(readxl)
library(ggplot2)
library(plotly)
library(MASS)

df<-read_xlsx("baseball-2016.xlsx",col_names=TRUE)
scaled_df<-scale(df[,-c(1,2)])
d<-dist(scaled_df)
mds<-isoMDS(d,k=2)
```

```
## initial value 19.856833
## iter 5 value 16.319153
## iter 10 value 16.046215
## final value 15.935476
## converged
```

```

dt<-(mds$points)
dt_new<-as.data.frame(dt)
dt_new$League<-df$League
rownames(dt_new)<-df$Team

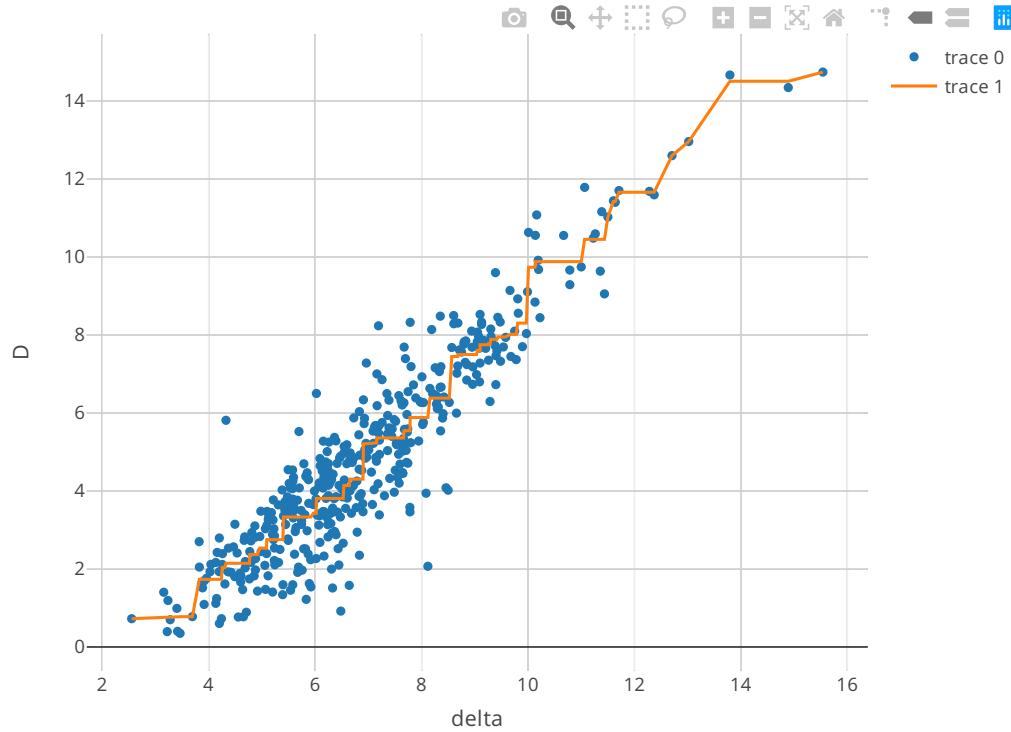
sh <- Shepard(d,dt)
delta <-as.numeric(d)
D<- as.numeric(dist(dt))

n=nrow(dt)
index=matrix(1:n, nrow=n, ncol=n)
index1=as.numeric(index[lower.tri(index)])

n=nrow(dt)
index=matrix(1:n, nrow=n, ncol=n, byrow = T)
index2=as.numeric(index[lower.tri(index)])

plot_ly()%>%
  add_markers(x=~delta, y=~D, hoverinfo ='text', text = ~paste('Obj1: ', df[index1,1],
  '  
 Obj 2:', df[index2,1]))%>%add_lines(x=~sh$x, y=~sh$yf)

```



The data points away from the line are the ones that are problematic. They are like Minnesota Twins and Arizona Diamondbacks, Oakland Athletics and Milwaukee Brewers were hardest to plot from Shepard plot.

```

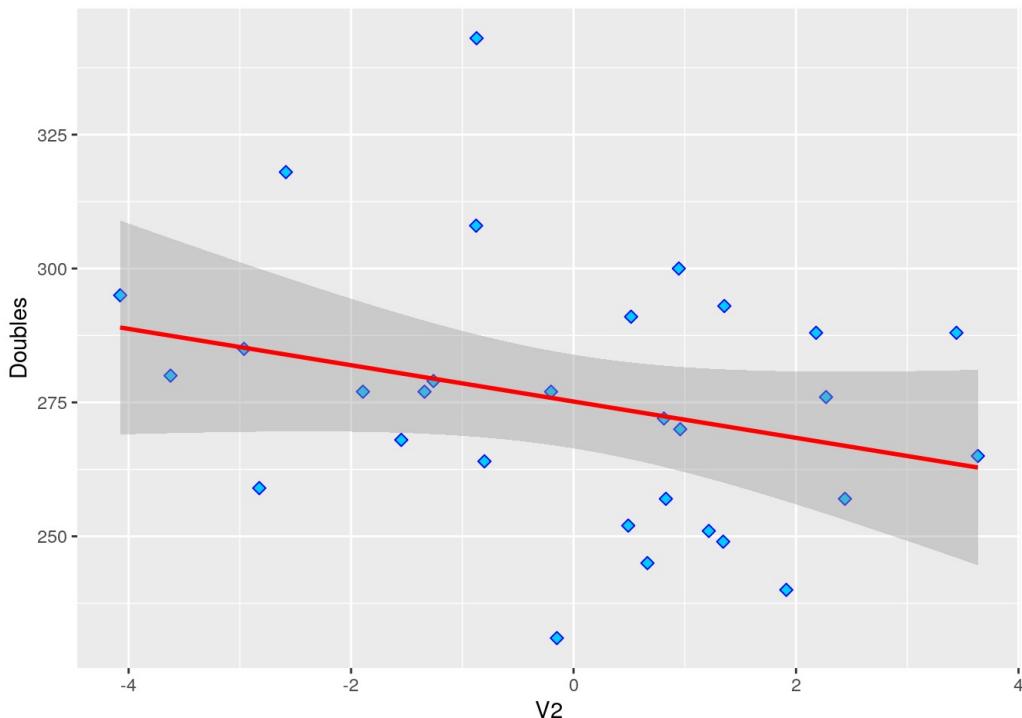
library(gridExtra)
num_df<-df[,-c(1,2)]
num_df$V2<-dt_new$V2

## numeric(0)

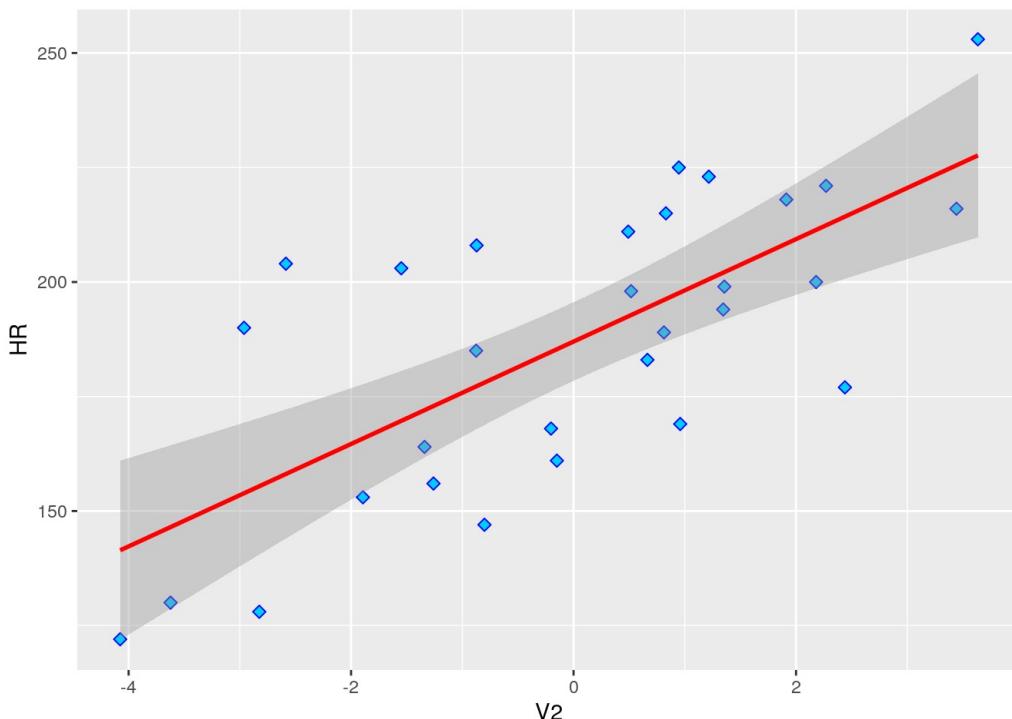
num_df<-as.data.frame(num_df)
names(num_df)[8]<-"Doubles"
names(num_df)[9]<-"Triples"

ggplot(num_df,aes(x=dt_new$V2,y=Doubles))+geom_point(size=2,color="blue",pch=23,fill="#00CCFF") +stat_smooth(method = "lm", col = "red") +xlab("V2")

```



```
ggplot(num_df,aes(x=dt_new$V2,y=HR))+geom_point(size=2,color="blue",pch=23,fill="#00CCFF")+stat_smooth(method = "lm", col = "red")+xlab("V2")
```



Doubles and Home Run could be observed as the two components that showed the strongest connection. This is true when considering the game itself especially for scoring. ####Appendix

```
## ----setup, include=FALSE,message=FALSE,warning=FALSE-----
knitr::opts_chunk$set(echo = TRUE)

## ----message=FALSE,warning=FALSE-----
library(ggplot2)
library(plotly)
library(MASS)

olive<-read.csv("olive.csv",header=TRUE, sep=",")

p1<-ggplot(olive,aes(palmitic,oleic,col=linolenic))+geom_point()+labs(x="palmitic",y="oleic")+ggtitle("Plot of Palmitic on Oleic coloured by Linolenic")
```

```

p1

interval_cut <- cut_interval(olive$linolenic, 4)
p2<-ggplot(olive,aes(palmitic,oleic,col=interval_cut))+geom_point()+theme_light()+labs(x="palmitic",y="oleic")+ggt
itle("Plot of Palmitic on Oleic coloured by Linolenic Cut intervals")+theme(axis.text = element_text(colou
e"))
p2

## -----
p3<-s2<-ggplot(olive,aes(palmitic,oleic,col=interval_cut))+geom_point()+theme_light()+labs(x="palmitic",y="oleic")
+geom_point(size=3)+ggttitle("Plot of Palmitic on Oleic coloured by Linolenic") + theme(axis.text = element_text(colou
r = "blue"))
p3

## -----
p4<-s2<-ggplot(olive,aes(palmitic,oleic))+geom_point(aes(size=interval_cut))+theme_light()+labs(x="palmitic",y="ol
eic") + geom_point() + ggttitle("Plot of Palmitic on Oleic coloured by Linolenic") + theme(axis.text = element_text(colou
r = "blue"))
p4

## -----
angleplot<-ggplot(olive,aes(x=palmitic,y=oleic)) +
  geom_point() + geom_spoke(aes(angle = as.numeric(cut_interval(linolenic, 4))*pi/2), radius = 50)
angleplot

## -----
p6<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=as.numeric(Region)))+labs(x="oleic",y="eicosenoic") + geom_point(siz
e=2) + ggttitle("oleic on eicosenoic coloured by Region")
p6

## -----
p7<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=as.factor(Region)))+labs(x="oleic",y="eicosenoic") + geom_point(size
=2) + ggttitle("oleic on eicosenoic coloured by Region")
p7

## -----
discretelionelic<-cut_interval(olive$linolenic, 3)
discretepalmitic<-cut_interval(olive$palmatic, 3)
discretepalmoleic<-cut_interval(olive$palmoleic, 3)

p8<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=discretelionelic,shape=discretepalmitic,size=discretepalmoleic))
+geom_point() + ggttitle("oleic vs eicosenoic with linolenic, palmitic,palmoleic")
p8

## -----
p9<-ggplot(olive,aes(x=oleic,y=eicosenoic,color=Region,shape=discretepalmitic,size=discretepalmoleic))+geom_poin
t() + ggttitle("oleic vs eicosenoic with linolenic, palmitic,palmoleic")
p9

## ----message=FALSE,warning=FALSE-----
library(plotrix)
library(plotly)

mytable <- table(olive$Area)
dframe<-as.data.frame(mytable)

#pie3D(dframe$Freq,labels=dframe$Var1,explode=0.1,main="Pie Chart of Area Proportion ")

p10 <- plot_ly(dframe, labels = ~Var1, values = ~Freq, type = 'pie',textinfo = "none", hoverinfo = "text",text=~Va
r1) %>%
  layout(title = 'Pie chart of Area Proportion',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))

p10

## -----

```

```

p11<-ggplot(olive, aes(x=linoleic, y=eicosenoic,col="red") ) +geom_density_2d(position="identity")+theme_dark()
p11

## ----message=FALSE,warning=FALSE-----
library(readxl)
library(ggplot2)
library(plotly)
library(MASS)

df<-read_xlsx("baseball-2016.xlsx",col_names=TRUE)

## -----
library(readxl)

df<-read_xlsx("baseball-2016.xlsx",col_names=TRUE)
df<-data.frame(df)

scaled_df<-scale(df[,-c(1,2)])
d<-dist(scaled_df)
mds<-isoMDS(d,k=2)

dt<-(mds$points)
dt_new<-as.data.frame(dt)
dt_new$League<-df[,2]
dt_new$Team<-df[,1]

plot_ly(dt_new, x=~V1, y=~V2, type="scatter", hovertext=~Team, color=~League, colors=c("green", "red"))

## ----message=FALSE,warning=FALSE-----
library(readxl)
library(ggplot2)
library(plotly)
library(MASS)

df<-read_xlsx("baseball-2016.xlsx",col_names=TRUE)
scaled_df<-scale(df[,-c(1,2)])
d<-dist(scaled_df)
mds<-isoMDS(d,k=2)

dt<-(mds$points)
dt_new<-as.data.frame(dt)
dt_new$League<-df$League
rownames(dt_new)<-df$Team

sh <- Shepard(d,dt)
delta <-as.numeric(d)
D<- as.numeric(dist(dt))

n=nrow(dt)
index=matrix(1:n, nrow=n, ncol=n)
index1=as.numeric(index[lower.tri(index)])

n=nrow(dt)
index=matrix(1:n, nrow=n, ncol=n, byrow = T)
index2=as.numeric(index[lower.tri(index)])

plot_ly()%>%
  add_markers(x=~delta, y=~D, hoverinfo ='text', text = ~paste('Obj1: ', df[index1,1],
  '  
 Obj 2:', df[index2,1]))%>%add_lines(x=~sh$x, y=~sh$y)

## ----message=FALSE,warning=FALSE-----
library(gridExtra)
num_df<-df[,-c(1,2)]
num_df$V2<-dt_new$V2
num_df<-as.data.frame(num_df)
names(num_df)[8]<-"Doubles"
names(num_df)[9]<-"Triples"

```

```
ggplot(num_df,aes(x=dt_new$V2,y=Doubles))+geom_point(size=2,color="blue",pch=23,fill="#00CCFF")+stat_smooth(method = "lm", col = "red")+xlab("V2")
```

```
## -----
```

```
ggplot(num_df,aes(x=dt_new$V2,y=HR))+geom_point(size=2,color="blue",pch=23,fill="#00CCFF")+stat_smooth(method = "lm", col = "red")+xlab("V2")
```

Lab3

Priya (priku577) and Andreas(andch552)

26 September 2018

Assignment 1

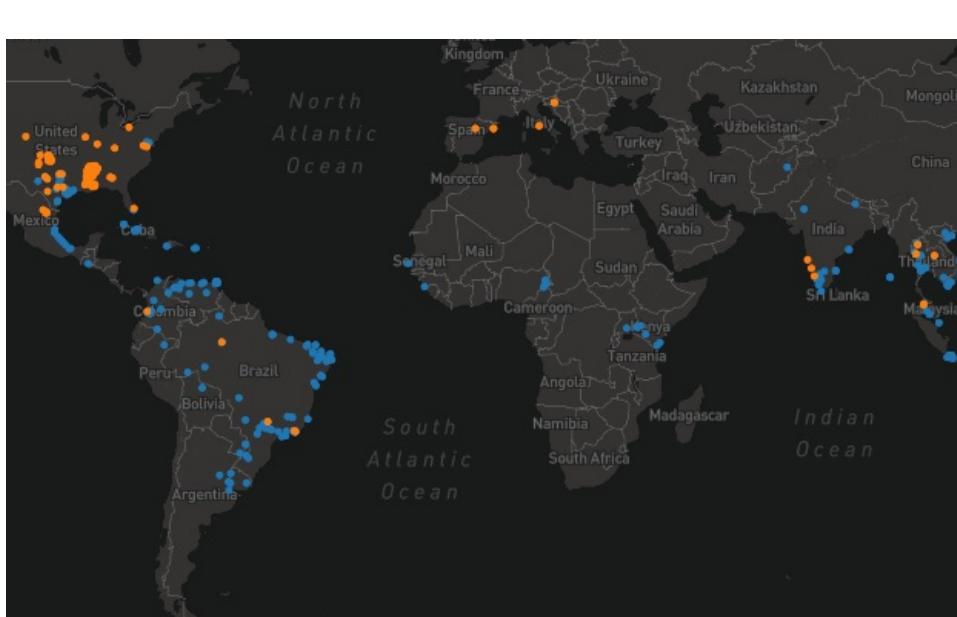
1. Mapbox plotly

MAPBOX PLOT FOR YEAR 2004

```
library(tidyverse)
library(plotly)
library(RCurl)
library(gridExtra)
library(RColorBrewer)

Sys.setenv('MAPBOX_TOKEN'='pk.eyJ1IjoicXVhcnRlcmlhaW5lIiwiYSI6ImNqbWJucjh4MjA2dm0zd25xMmp4ejZzMnQifQ.-FXHcA1t8b_YZ
kdRSUXuGw')

mosquitos<-read.csv('aegypti_albopictus.csv',header=TRUE)
p_2004<-mosquitos%>%filter(YEAR=='2004')%>%plot_mapbox(lat = ~Y, lon = ~X,
                                                               split=~VECTOR,colors = 'Set3',
                                                               mode = 'scattermapbox', hoverinfo='name')%>%
layout(title = 'MOSQUITOS PLOTS FOR YEAR 2004',
       font = list(color='white'),
       plot_bgcolor = '#191A1A', paper_bgcolor = '#191A1A',
       mapbox = list(style = 'dark'),
       legend = list(orientation = 'h',font = list(size = 8)),
       margin = list(l = 25, r = 25,
                     b = 25, t = 25,
                     pad = 2))
p_2004
```

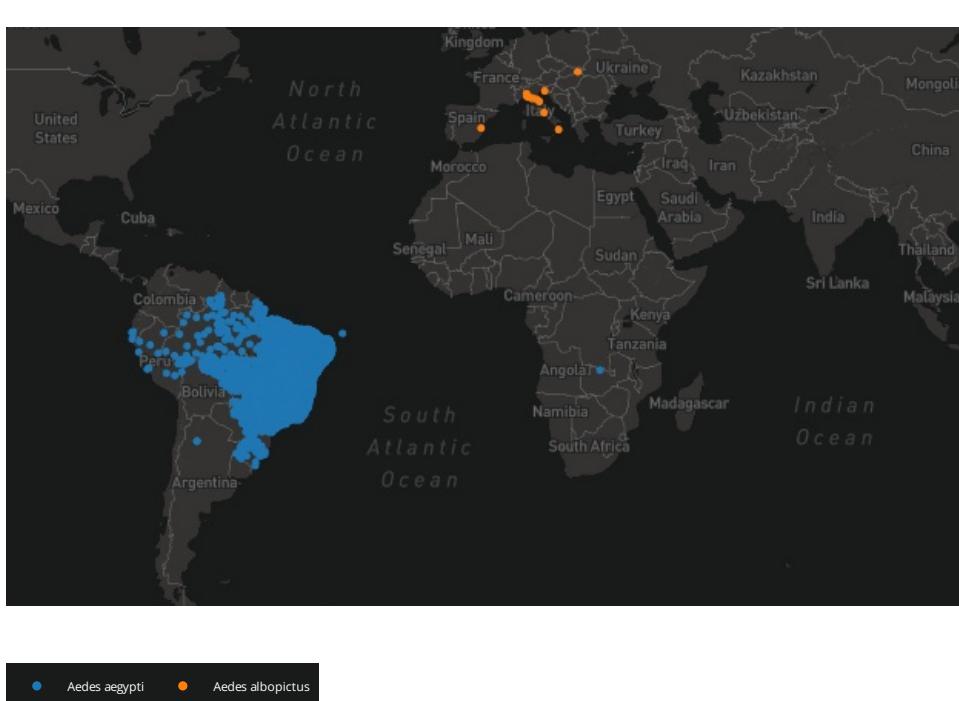


MAPBOX PLOT FOR YEAR 2013

```

p_2013<-mosquitos %>%filter(YEAR=='2013')%>%plot_mapbox(lat = ~Y, lon = ~X,
                                                               split=~VECTOR,colors = 'Set3',mode = 'scattermapbox', hove
rinfo='name')%>
  layout(title = 'MOSQUITOS PLOTS FOR YEAR 2013',
         font = list(color='white'),
         plot_bgcolor = '#191A1A', paper_bgcolor = '#191A1A',
         mapbox = list(style = 'dark'),
         legend = list(orientation = 'h',
                      font = list(size = 8)),
         margin = list(l = 25, r = 25,
                      b = 25, t = 25,
                      pad = 2))

```



Comparing the two plots we can come to conclusions like

- 1.The amount of Aedes aegypti in Brazil has increased from 2004 to 2013.Brazil had one of the highest number of Aedes aegypti in 2013 as wellThe regions of rio di Janeiro had more cases.
 - 2.The amount of aedes albopictus was high in the US and Taiwan during the yer 2004.The regions most affected were mississippi in US and kaohsiung,Tainam in vietnam.Seems like US could eradicate the aedes albopictus from the mississippi area completey but taiwan on other hand in taiwan their number increased.

The perception problems are overplotting.comparison on the population of mosquitoes in each region is difficult.

2. Choropleth Map

CHOROPLETH MAP Equirectangular Projection

```

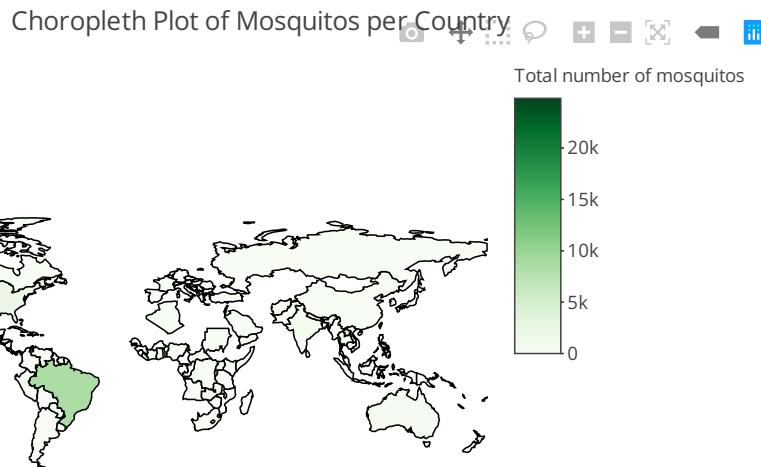
d<-mosquitos%>%select(c("VECTOR","COUNTRY_ID"))
x<-group_by(d,COUNTRY_ID)%>%count()
s<-as.data.frame(x)

g1 <- list(
  showframe = FALSE,
  showcoastlines = FALSE,
  projection = list(type = 'equirectangular')
)

p1 <- s%>% plot_geo()  %>%
  add_trace(
    z = ~n, color=~n,colors = 'Greens',
    text = ~COUNTRY_ID,locations=~COUNTRY_ID
  ) %>% colorbar(title = 'Total number of mosquitos')%>%
  layout(title = 'Choropleth Plot of Mosquitos per Country',
    geo = g1
  )

p1

```



Here the colour scale is from 0-20k .It is very difficult to percieve the data based on this colous scale as instead of different hues intensity varies.

3.Choropleth Map with LOG(Z) transformation

a.Equidistant projection log(z)

```

s$log_n<-log(s$n)

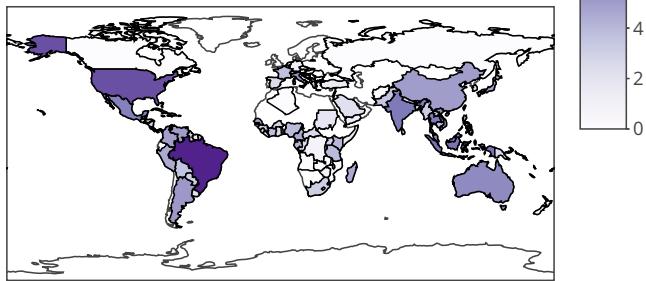
g2 <- list(
  projection = list(type = "equirectangular")
)

p2 <- s%>% plot_geo()  %>%
  add_trace(
    z = ~log_n, color=~log_n,colors = 'Purples',
    text = ~COUNTRY_ID,locations=~COUNTRY_ID
  ) %>% colorbar(title = 'Logarithmic Total number of mosquitos')%>%
  layout(title = 'Choropleth Plot of Mosquitos per Country \n with Log Transformation',
    geo = g2
  )

```

p2





Here we can see the colour scale is from 0-10 instead of 0-20K. The changes can be noted in a better way. A point to be noted here is Russia had a mosquito count of 1 earlier and now it log(1) and hence 0. Countries like India, USA can be differentiated better now.

b. Conic Equal Projection log(Z)

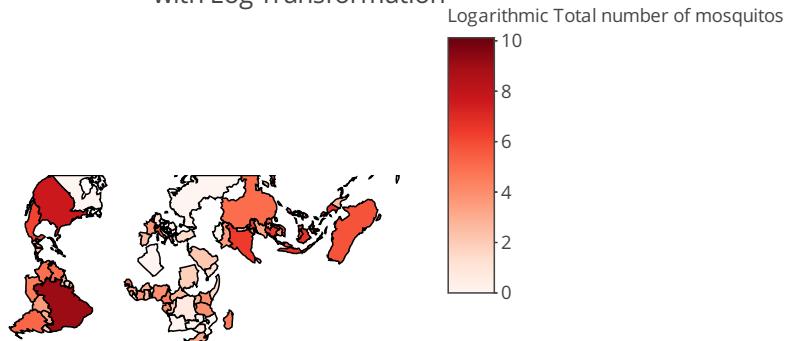
```
s$log_n<-log(s$n)

g3 <- list(
  showframe = FALSE,
  showcoastlines = FALSE,
  projection = list(type = "conic equal area")
)

p3 <- s%>% plot_geo() %>%
  add_trace(
    z = ~log_n, color=~log_n, colors = 'Reds',
    text = ~COUNTRY_ID, locations=~COUNTRY_ID
  ) %>% colorbar(title = 'Logarithmic Total number of mosquitos')%>%
  layout(title = 'Choropleth Plot of Mosquitos per Country \n with Log Transformation',
         geo = g3
  )

p3
```

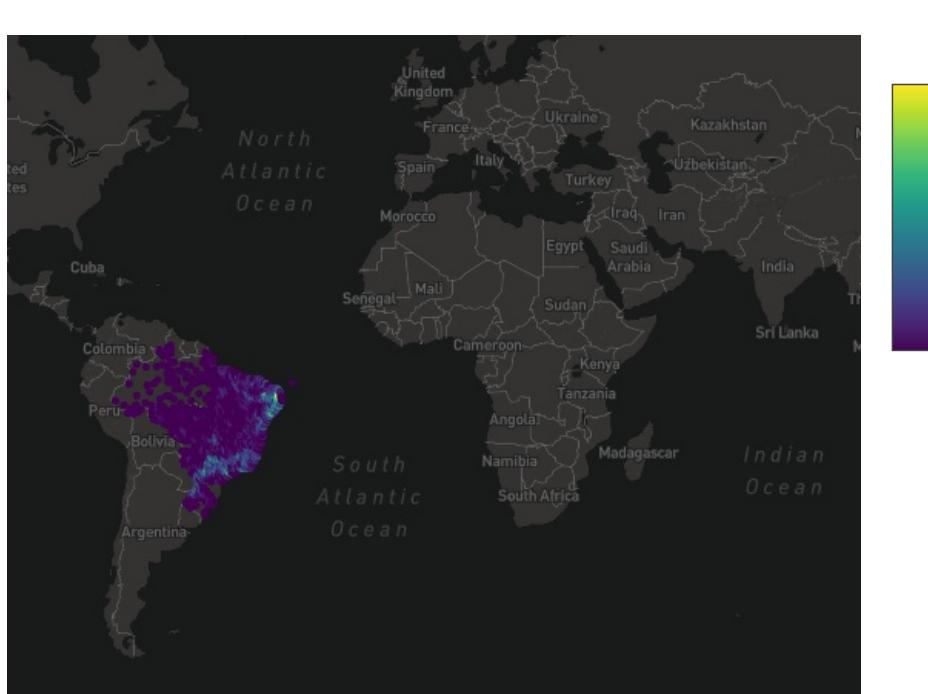
Choropleth Plot of Mosquitos per Country
with Log Transformation



comparing both the map it is clear that the map 3.a is better.The map 3b leads to misinterpretations.In conical projections as we move away from regions where cone intersects the globe the distortions become more.

Mapbox of Brazil

```
mosquitos_brazil<-mosquitos[(mosquitos$COUNTRY == "Brazil" & mosquitos$YEAR == "2013"), ]  
  
mosquitos_brazil$X1<-cut_interval(mosquitos_brazil$X,100)  
mosquitos_brazil$Y1<-cut_interval(mosquitos_brazil$Y,100)  
  
#mos_group_X1<-as.data.frame(mosquitos_brazil%>%group_by(X1)%>%summarise(mean_group_X=mean(X),n_group_X=n()))  
#mos_group_Y1<-as.data.frame(mosquitos_brazil%>%group_by(Y1)%>%summarise(mean_group_Y=mean(Y),n_group_Y=n()))  
  
mos<-as.data.frame(mosquitos_brazil)%>%group_by(X1,Y1)%>%summarise(m1=mean(X),m2=mean(Y),N=n())  
  
## Warning: Factor `X1` contains implicit NA, consider using  
## `forcats::fct_explicit_na`  
  
## Warning: Factor `Y1` contains implicit NA, consider using  
## `forcats::fct_explicit_na`  
  
mos<-as.data.frame(mos)  
  
br <-mos %>%plot_mapbox(lat = ~m2, lon = ~m1,color = ~N ,mode = 'scattermapbox', hoverinfo='name')%>%  
  layout(title = 'Mean values of X and Y per group (X1,Y1) and amount of obs per group (X1,Y1) ',  
         font = list(color='white'),  
         plot_bgcolor = '#191A1A', paper_bgcolor = '#191A1A',  
         mapbox = list(style = 'dark'),  
         legend = list(orientation = 'h',  
                      font = list(size = 8)),  
         margin = list(l = 25, r = 25,  
                      b = 25, t = 25,  
                      pad = 2))  
  
br  
## Warning: Ignoring 1 observations
```



As we can observe from the plot Nova Cruz, Guarabira and Sao Paulo are some cities with high levels of Mosquitos. This discretization defenately help in analyzing the distribution of mosquitoes accross Brazil.

Assignment 2

1.Young,Adult and senior of the swedish counties are grouped.

```
library(RCurl)
library(tidyverse)

swedish<-read.csv("kd.csv", header=TRUE,encoding = "latin1")

swedish_wide <- spread(swedish, age,X2016)
swedish_wide =swedish_wide[,-2]

names(swedish_wide)<-c("Region","Young","Adult","Senior")
```

2.Violin plot

```
ee <- swedish_wide %>%
  plot_ly(type = 'violin') %>%
  add_trace(y = ~Young, name = 'Young', box = list(visible = T),
            meanline = list(visible = T), line = list(color = 'pink'))
) %>%
  add_trace(y = ~Adult, name = 'Adults', box = list(visible = T),
            meanline = list(visible = T), line = list(color = 'blue')) %>%
  add_trace(
    y = ~Senior, name = 'Senior', box = list(visible = T),
    meanline = list(visible = T), line = list(color = 'green'))
) %>%
  layout(yaxis = list(title = "income",
                      zeroline = F), xaxis = list(title = "Age Group"), title = "Income vs Age Group")
)
```

ee



The mean income is lowest for age 18-24. This is true may be coz they just start working. All three groups have a small group of people having very high income range (outliers). The age groups of 30-49 and 50-64 have almost similar mean and distribution, with majority having income in range 500 K SEK.

3.Surface plot

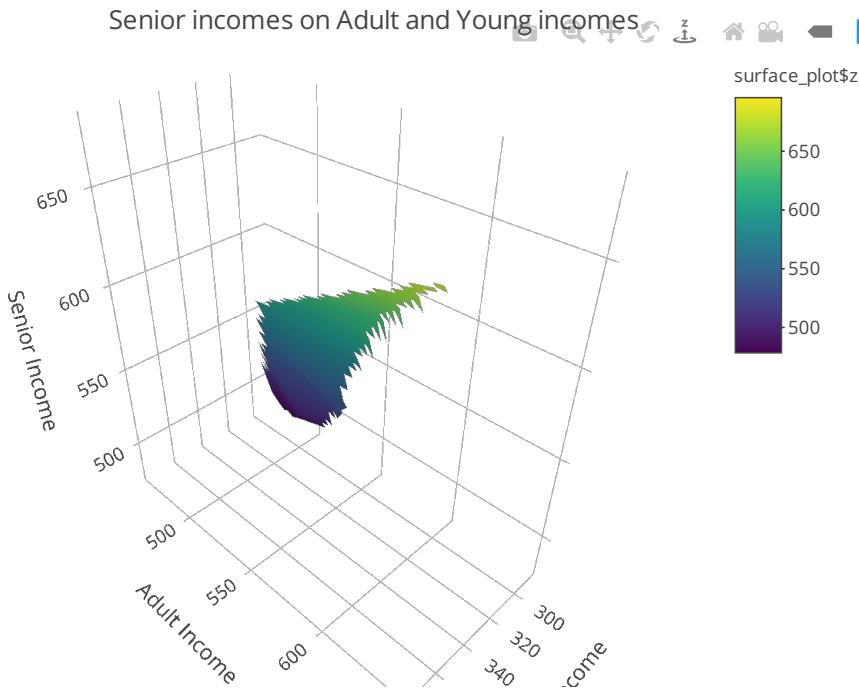
```

library(akima)

surface_plot=interp(swedish_wide$Young, swedish_wide$Adult, swedish_wide$Senior, duplicate = "mean")

plot_ly(x=~surface_plot$x, y=~surface_plot$y, z=~surface_plot$z, type="surface") %>% layout(
  title = "Senior incomes on Adult and Young incomes",
  scene = list(
    xaxis = list(title = "Young Income"),
    yaxis = list(title = "Adult Income"),
    zaxis = list(title = "Senior Income")
  )
)

```



There is linear relationship and hence linear model fits well.

4. Choropleth Plots of Adults and Youngs

```

library(RCurl)
library(tidyverse)
#library(sf)
library(plotly)
library(RCurl)
library(tidyverse)
library(sf)
library(plotly)

swedish<-read.csv("kd.csv", header=TRUE, encoding = "latin1")

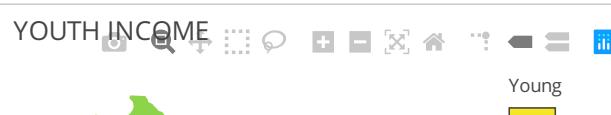
swedish_wide <- spread(swedish, age, X2016)
swedish_wide =swedish_wide[,-2]

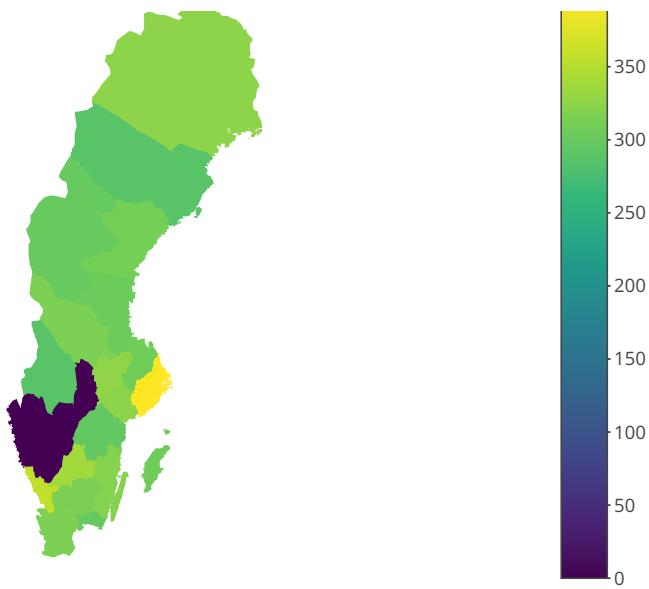
names(swedish_wide)<-c("Region", "Young", "Adult", "Senior")
map<-readRDS("gadm36_SWE_1_sf.rds")

swedish <- swedish_wide %>% separate(Region,c("Num", "Region", "County"), extra = 'drop')%>%select("Region", "Young", "Adult", "Senior")

rownames(swedish)=swedish$Region
map$Young=swedish[map$NAME_1, "Young"]
map$Young[is.na(map$Young)]=0
map$Adult=swedish[map$NAME_1, "Adult"]
map$Adult[is.na(map$Adult)]=0
Young_value <- plot_ly() %>% add_sf(data=map, split=~NAME_1, color=~Young, showlegend=F, alpha=1) %>%
  layout(title="YOUTH INCOME")
Young_value

```



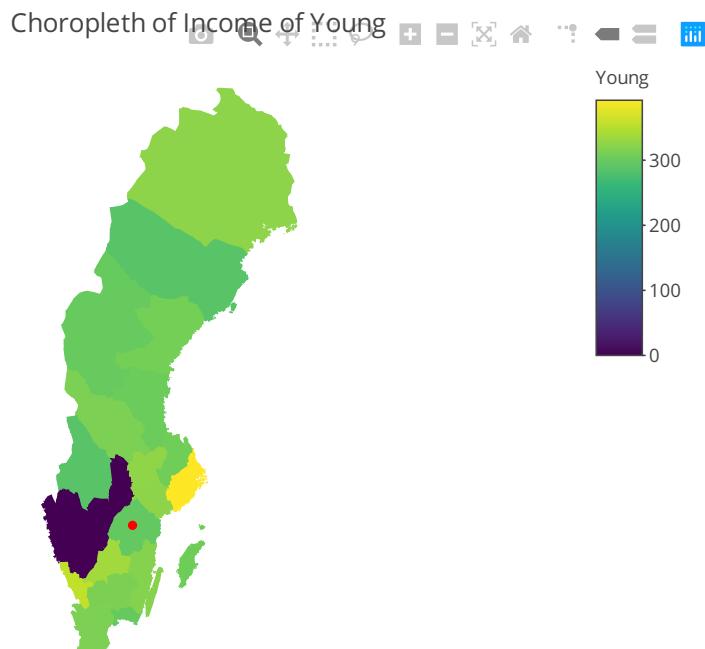


As we can observe from both Choropleth Plots both Young and Adults in Stockholm have higher income regarding rest of Sweden. This seems to be reasonable as Stockholm is the capital city and there would be lot of job opportunities there. It is easier to compare between regions. BUT when we look at the country map it is difficult to say about the income distribution because of the continuous colour scale of colour blue.

5. Choropleth Plots of Adults and Youngs with Linkoping coords

```
library(sf)
lat= 58.409814
longitude = 15.624525
name = "Linkoping"
desc = "Linköping, Östergötlands län, SE"

data_point <- data.frame(lat, longitude, name, desc)
plot_ly() %>% add_sf(data=map, split=~NAME_1, color=~Young, showlegend=F, alpha=1,
                       type = "scatter") %>% layout(title = "Choropleth of Income of Young") %>% add_markers(data =
data_point,
t, x = ~longitude, color = I("red"), text='Linkoping')
y = ~la
```



APPENDIX

```

# Assignment 1 ----

# 1 -----
library(tidyverse)
library(plotly)
library(RCurl)
library(gridExtra)
library(RColorBrewer)

Sys.setenv('MAPBOX_TOKEN'='pk.eyJ1IjoicXVhcnRlcmlhaW5lIiwiYSI6ImNqbWJucjh4MjA2dm0zd25xMmp4ejZzMnQifQ.-FXHcA1t8b_YZkdRSUXuGw')

mosquitos<-read.csv('aegypti_albopictus.csv',header=TRUE)
p_2004<-mosquitos%>%filter(YEAR=='2004')%>%plot_mapbox(lat = ~Y, lon = ~X,
                                                               split=~VECTOR,colors = 'Set3',
                                                               mode = 'scattermapbox', hoverinfo='name')%>%
layout(title = 'MOSQUITOS PLOTS FOR YEAR 2004',
       font = list(color='white'),
       plot_bgcolor = '#191A1A', paper_bgcolor = '#191A1A',
       mapbox = list(style = 'dark'),
       legend = list(orientation = 'h',font = list(size = 8)),
       margin = list(l = 25, r = 25,
                     b = 25, t = 25,
                     pad = 2))
p_2004

p_2013<-mosquitos %>%filter(YEAR=='2013')%>%plot_mapbox(lat = ~Y, lon = ~X,
                                                               split=~VECTOR,colors = 'Set3',mode = 'scattermapbox', hoverinfo='name')%>%
layout(title = 'MOSQUITOS PLOTS FOR YEAR 2013',
       font = list(color='white'),
       plot_bgcolor = '#191A1A', paper_bgcolor = '#191A1A',
       mapbox = list(style = 'dark'),
       legend = list(orientation = 'h',
                     font = list(size = 8)),
       margin = list(l = 25, r = 25,
                     b = 25, t = 25,
                     pad = 2))
p_2013

# 2 ----

d<-mosquitos%>%select(c("VECTOR","COUNTRY_ID"))
x<-group_by(d,COUNTRY_ID)%>%count()
s<-as.data.frame(x)

g1 <- list(
  showframe = FALSE,
  showcoastlines = FALSE,
  projection = list(type = 'equirectangular')
)

p1 <- s%>% plot_geo()  %>%
  add_trace(
    z = ~n, color=~n,colors = 'Greens',
    text = ~COUNTRY_ID,locations=~COUNTRY_ID
  ) %>% colorbar(title = 'Total number of mosquitos')%>%
  layout(title = 'Choropleth Plot of Mosquitos per Country',
         geo = g1
  )

p1

# 3 ----

s$log_n<-log(s$n)

g2 <- list(
  projection = list(type = "equirectangular")
)

p2 <- s%>% plot_geo()  %>%
  add_trace(

```

```

z = ~log_n, color=~log_n,colors = 'Purples',
text = ~COUNTRY_ID,locations=~COUNTRY_ID
) %>% colorbar(title = 'Logarithmic Total number of mosquitos')%>%
layout(title = 'Choropleth Plot of Mosquitos per Country \n with Log Transformation',
      geo = g2
)

p2

s$log_n<-log(s$n)

g3 <- list(
  showframe = FALSE,
  showcoastlines = FALSE,
  projection = list(type = "conic equal area")
)

p3 <- s%>% plot_geo()  %>%
  add_trace(
    z = ~log_n, color=~log_n,colors = 'Reds',
    text = ~COUNTRY_ID,locations=~COUNTRY_ID
) %>% colorbar(title = 'Logarithmic Total number of mosquitos')%>%
layout(title = 'Choropleth Plot of Mosquitos per Country \n with Log Transformation',
      geo = g3
)

p3

# 4 -------

mosquitos_brazil<-mosquitos[(mosquitos$COUNTRY == "Brazil" & mosquitos$YEAR == "2013" ), ]

mosquitos_brazil$X1<-cut_interval(mosquitos_brazil$X,100)
mosquitos_brazil$Y1<-cut_interval(mosquitos_brazil$Y,100)

#mos_group_X1<-as.data.frame(mosquitos_brazil%>%group_by(X1)%>%summarise(mean_group_X=mean(X),n_group_X=n()))
#mos_group_Y1<-as.data.frame(mosquitos_brazil%>%group_by(Y1)%>%summarise(mean_group_Y=mean(Y),n_group_Y=n()))

mos<-as.data.frame(mosquitos_brazil)%>%group_by(X1,Y1)%>%summarise(m1=mean(X),m2=mean(Y),N=n())
mos<-as.data.frame(mos)

br <-mos %>plot_mapbox(lat = ~m2, lon = ~m1,color = ~N ,mode = 'scattermapbox', hoverinfo='name')%>%
  layout(title = 'Mean values of X and Y per group (X1,Y1) and amount of obs per group (X1,Y1) ',
         font = list(color='white'),
         plot_bgcolor = '#191A1A', paper_bgcolor = '#191A1A',
         mapbox = list(style = 'dark'),
         legend = list(orientation = 'h',
                      font = list(size = 8)),
         margin = list(l = 25, r = 25,
                      b = 25, t = 25,
                      pad = 2))

br

# Assignment 2 -------

# 1 -----
library(RCurl)
library(tidyverse)

swedish<-read.csv("kd.csv", header=TRUE,encoding = "latin1")

swedish_wide <- spread(swedish, age,X2016)
swedish_wide =swedish_wide[,-2]

names(swedish_wide)<-c("Region","Young","Adult","Senior")

# 2 -----
ee <- swedish_wide %>%
  plot_ly(type = 'violin') %>%
  add_trace(y = ~Young,name = 'Young',box = list(visible = T),
            meanline = list(visible = T),line = list(color = 'pink'))
) %>%
  add_trace(y = ~Adult,name = 'Adults',box = list(visible = T),
            meanline = list(visible = T),line = list(color = 'blue')) %>%

```

```

add_trace(
  y = ~Senior, name = 'Senior', box = list(visible = T),
  meanline = list(visible = T), line = list(color = 'green')
) %>%
layout(yaxis = list(title = "income ",
                     zeroline = F), xaxis = list(title = "Age Group"), title = "Income vs Age Group"
)

ee

# 3 ----

library(akima)

surface_plot=interp(swedish_wide$Young, swedish_wide$Adult, swedish_wide$Senior, duplicate = "mean")

plot_ly(x=~surface_plot$x, y=~surface_plot$y, z=~surface_plot$z, type="surface") %>% layout(
  title = "Senior incomes on Adult and Young incomes",
  scene = list(
    xaxis = list(title = "Young Income"),
    yaxis = list(title = "Adult Income"),
    zaxis = list(title = "Senior Income")
))

# 4 ----

library(RCurl)
library(tidyverse)
#library(sf)
library(plotly)
library(RCurl)
library(tidyverse)
library(sf)
library(plotly)

swedish<-read.csv("kd.csv", header=TRUE,encoding = "latin1")

swedish_wide <- spread(swedish, age,X2016)
swedish_wide =swedish_wide[, -2]

names(swedish_wide)<-c("Region","Young","Adult","Senior")
map<-readRDS("gadm36_SWE_1_sf.rds")

swedish <- swedish_wide %>% separate(Region,c("Num","Region","County"),extra = 'drop')%>%select("Region","Young","Adult","Senior")

rownames(swedish)=swedish$Region
map$Young=swedish[map$NAME_1, "Young"]
map$Young[is.na(map$Young)]=0
map$Adult=swedish[map$NAME_1, "Adult"]
map$Adult[is.na(map$Adult)]=0
Young_value <- plot_ly() %>% add_sf(data=map, split=~NAME_1, color=~Young, showlegend=F, alpha=1) %>%
  layout(title="YOUTH INCOME")
Young_value

# 4 ----
library(sf)
lat= 58.409814
longitude = 15.624525
name = "Linkoping"
desc = "Linköping, Östergötlands län, SE"

data_point <- data.frame(lat, longitude, name, desc)
plot_ly() %>% add_sf(data=map, split=~NAME_1, color=~Young, showlegend=F, alpha=1,
                      type = "scatter") %>% layout(title = "Choropleth of Income of Young") %>% add_markers(data =
data_point,
t, x = ~longitude, color = I("red"), text='Linkoping')                                     y = ~la

```

Lab4

Priya(priku577) and Andreas(andch552)

October 3, 2018

Assignment1

1.Importing the data and filtering it

```
library(tidyverse)
```

```
## -- Attaching packages -----
----- tidyverse 1.2.1 --
```

```
## <U+221A> ggplot2 3.0.0      <U+221A> purrr    0.2.5
## <U+221A> tibble   1.4.2      <U+221A> dplyr    0.7.6
## <U+221A> tidyr    0.8.1      <U+221A> stringr  1.3.1
## <U+221A> readr    1.1.1      <U+221A>forcats  0.3.0
```

```
## -- Conflicts -----
----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(RColorBrewer)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

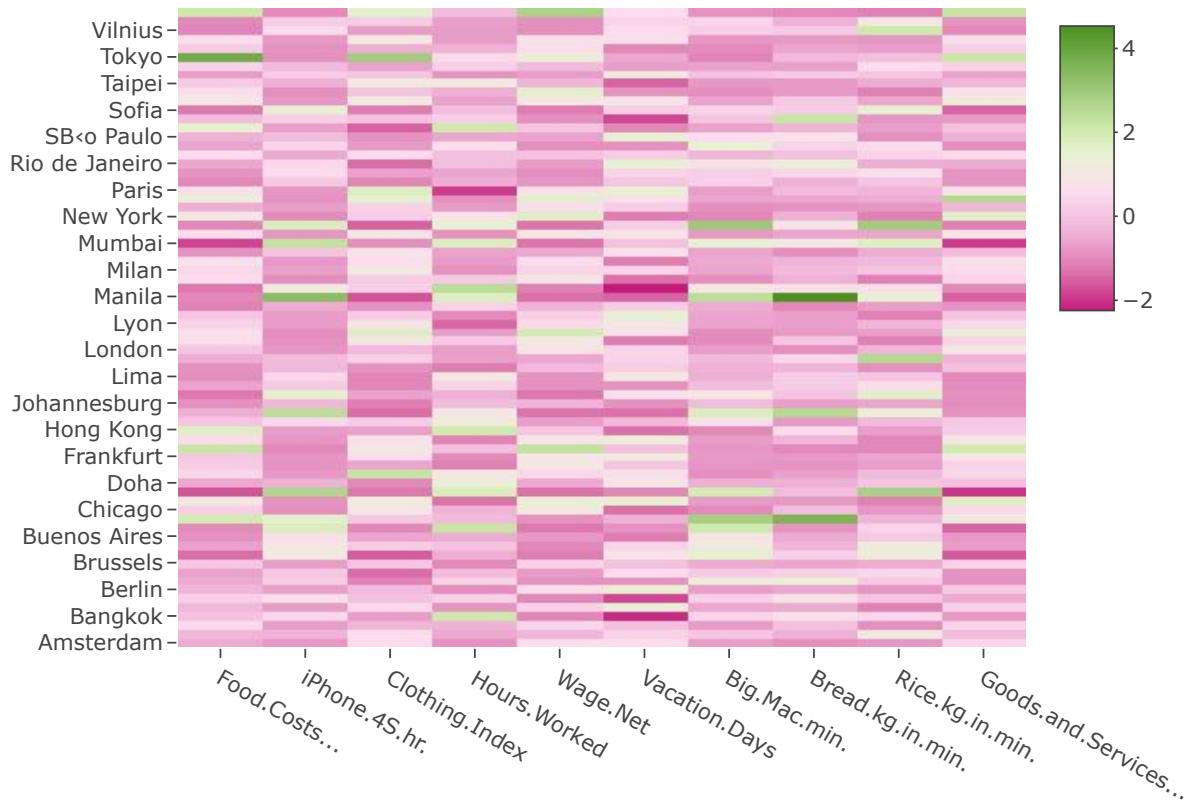
```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(seriation)
df<-read.csv("prices-and-earnings.csv")
adults<-df[c(1,2,5,6,7,9,10,16,17,18,19)]

rownames(adults)<-df[,1]
adults<-adults[,-c(1)]
```

2. Heat map of the data

```
coul = colorRampPalette(brewer.pal(8, "PiYG"))(25)
adscaled=scale(adults)
p2<-plot_ly(x=colnames(adscaled), y=rownames(adscaled), z=adscaled, type="heatmap", colors = coul)
p2
```



some outliers can be spotted. But it is very difficult to plot clusters since the data is unordered.

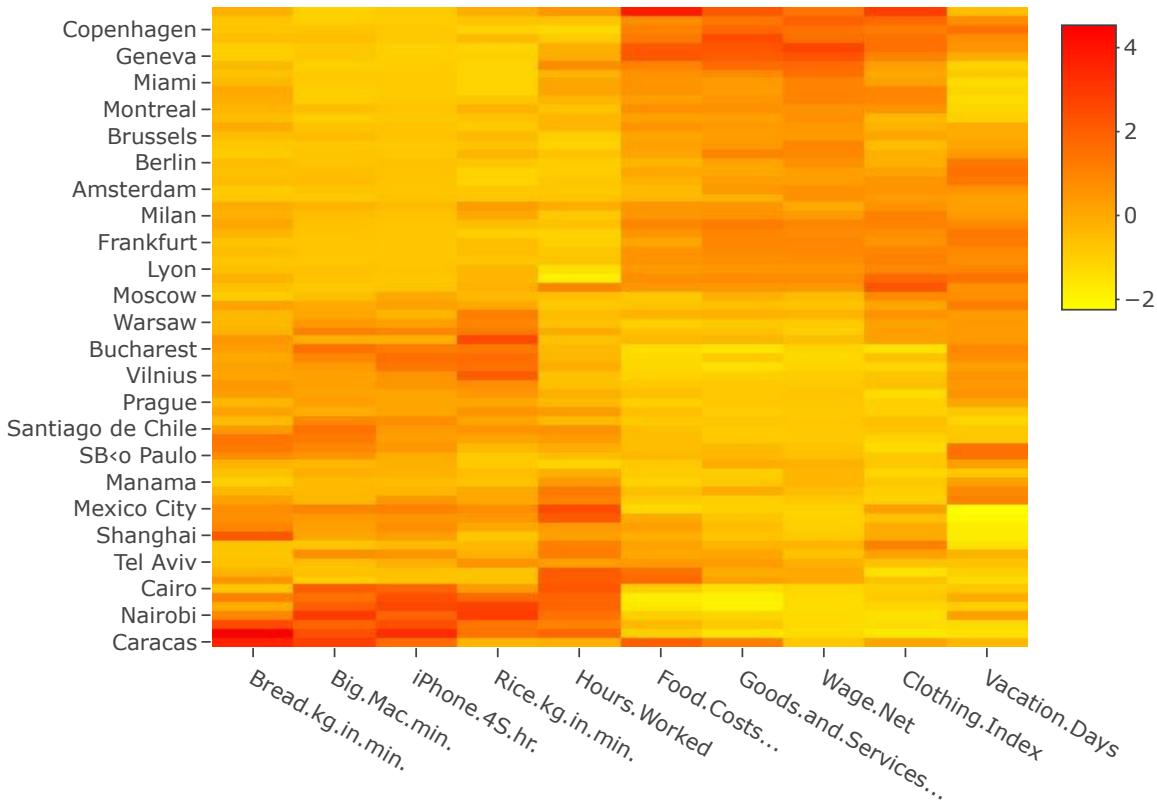
3. Heatmaps

Heat map using euclidian distance

```

rowdist_e<-dist(adscaled,method = "minkowski", p=2)
coldist_e<-dist(t(adscaled),method = "minkowski", p=2)
order1_e<-seriate(rowdist_e, "GW")
order2_e<-seriate(coldist_e, "GW")
ord1_e<-get_order(order1_e)
ord2_e<-get_order(order2_e)
reordmatr_euc<-adscaled[rev(ord1_e),ord2_e]
plot_ly(x=colnames(reordmatr_euc), y=rownames(reordmatr_euc),
        z=as.matrix(reordmatr_euc), type="heatmap", colors =colorRamp(c("yellow", "red")))

```



Heat Map with 1-Correlation HC

```

coldist_c<-as.dist(1-cor(adscaled))
rowdist_c<-as.dist(1-cor(t(adscaled)))
order1_c<-seriate(rowdist_c, "GW")
order2_c<-seriate(coldist_c, "GW")
ord1_c<-get_order(order1_c)
ord2_c<-get_order(order2_c)
reordmatr_c<-adscaled[rev(ord1_c),ord2_c]
plot_ly(x=colnames(reordmatr_c), y=rownames(reordmatr_c),
        z=as.matrix(reordmatr_c), type="heatmap", colors =colorRamp(c("yellow", "red")))

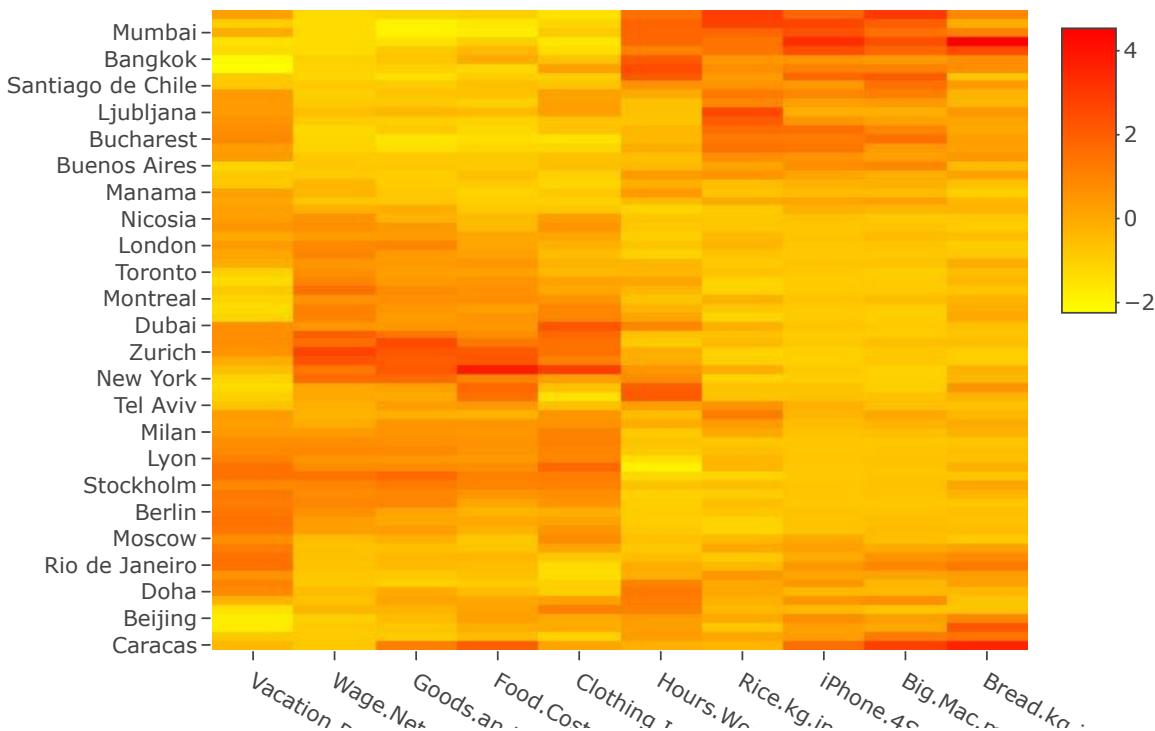
```



THe two plots arrive at same conclusions as both of the plots are related.

4. Heat Map with Euclidean Distance TSP

```
rowdist_tsp<-dist(adscaled,method = "minkowski", p=2)
coldist_tsp<-dist(t(adscaled),method = "minkowski", p=2)
order1_tsp<-seriate(rowdist_tsp, "TSP")
order2_tsp<-seriate(coldist_tsp, "TSP")
ord1_tsp<-get_order(order1_tsp)
ord2_tsp<-get_order(order2_tsp)
reordmatr_tsp<-adscaled[rev(ord1_tsp),ord2_tsp]
plot_ly(x=colnames(reordmatr_tsp), y=rownames(reordmatr_tsp),
        z=as.matrix(reordmatr_tsp), type="heatmap", colors =colorRamp(c("yellow", "red")))
```



TSP during each run produce a different result due to the randomness.

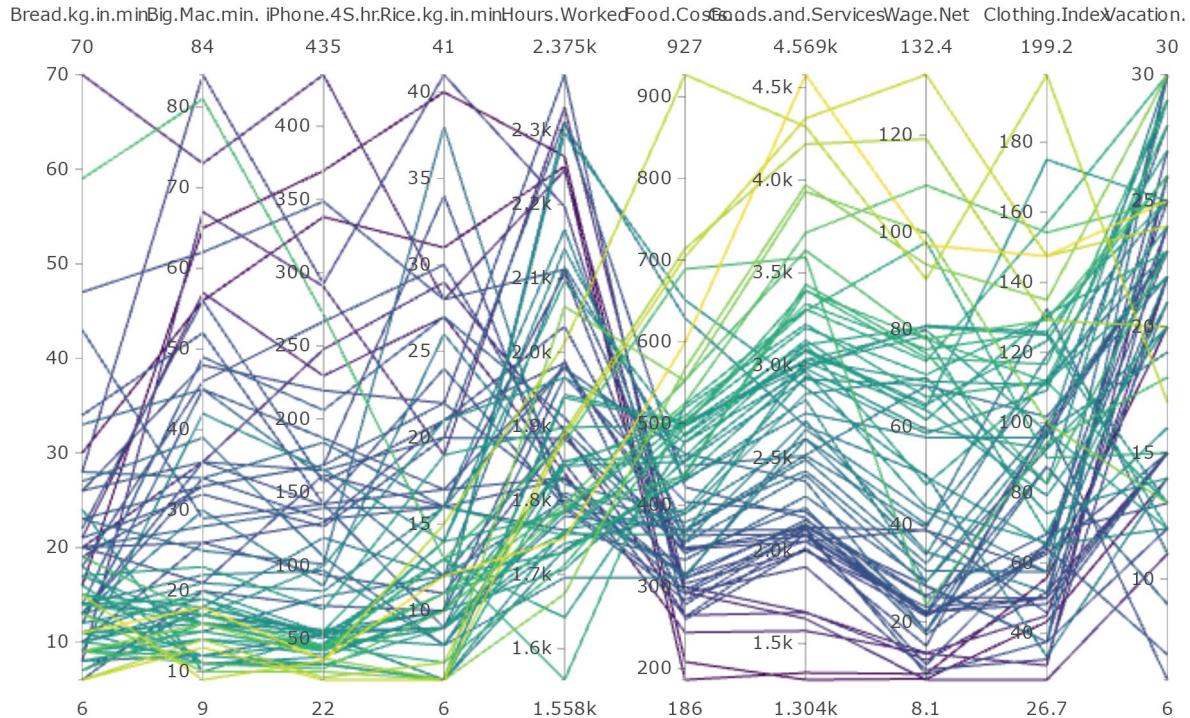
5. Parallel Coordinate Plot for Unsorted data

```
v<-1:11
ord=sample(v)
#ord=ord2

dims0=list()
for( i in 1:ncol(adults)){
  dims0[[i]]=list( label=colnames(adults)[ord2_e[i]],
                  values=as.formula(paste("~",colnames(adults)[ord2_e[i]])))
}
p <-adults %>%
  plot_ly(type = 'parcoords',
         line = list(color = ~as.numeric(Goods.and.Services...)),
         dimensions = dims0
  )%>%layout(title = "Plots from sorted data")

p
```

Plots from sorted data



6. Radar plot

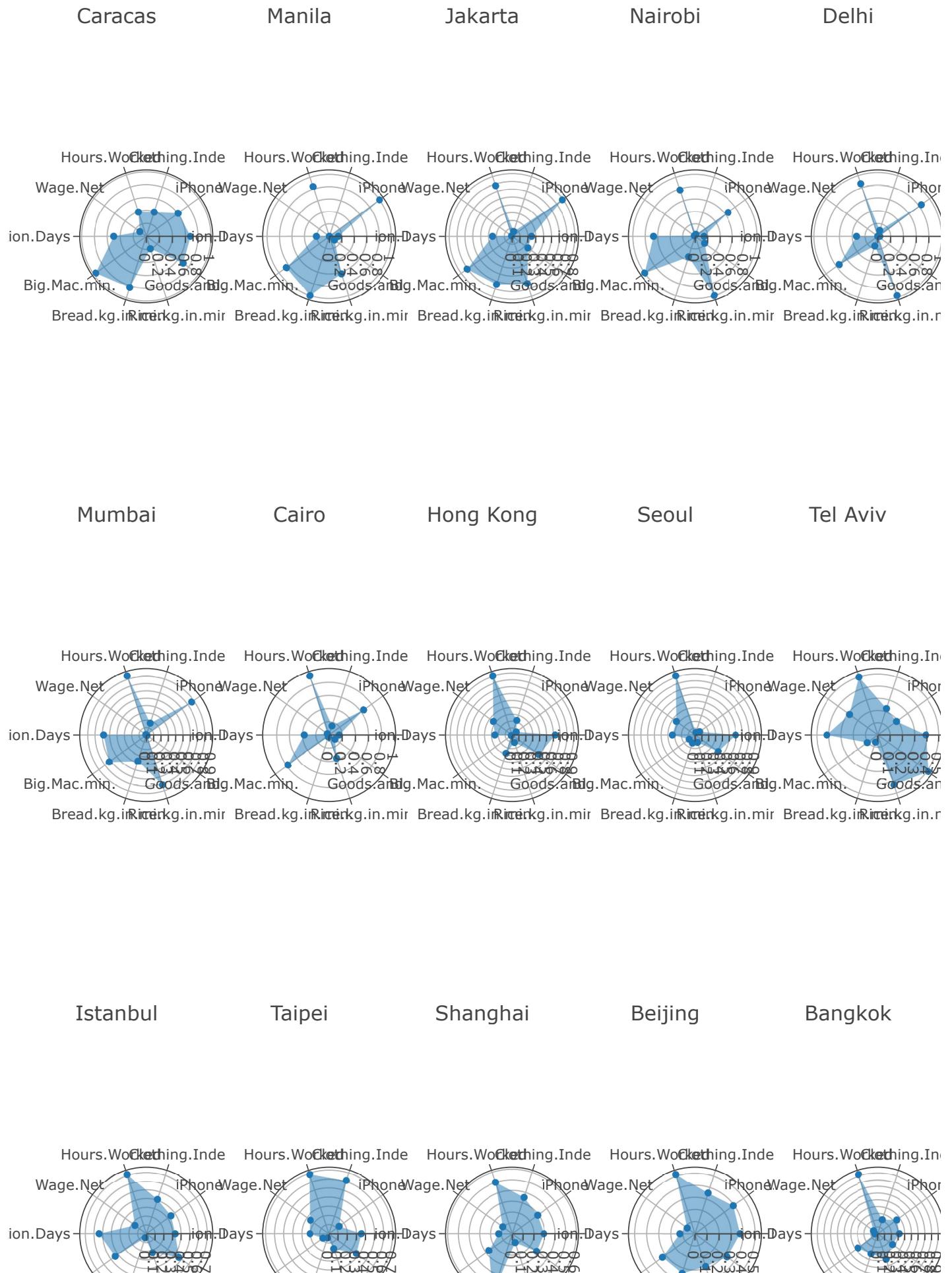
```
library(scales)
Ps=list()
nPlot=72

ad<-as.data.frame(adscaled)
ad[rev(ord1_e), ]%>%
  add_rownames( var = "group" ) %>%
  mutate_each(funns(rescale), -group) -> pe_radar

## Warning: Deprecated, use tibble::rownames_to_column() instead.

for (i in 1:nPlot){
  Ps[[i]] <- htmltools::tags$div(
    plot_ly(type = 'scatterpolar',
            r=as.numeric(pe_radar[i,-1]),
            theta= colnames(pe_radar)[-1],
            fill="toself")%>%
    layout(title=pe_radar$group[i]), style="width: 20%;") # 4 plots per row
}

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



```
#Using Juxtaposed and set scale to True
# stars(adults[rev(ord1_e),],
#        key.loc=c(23,2),
#        col.stars =rep("lightblue", nrow(df)),
#        cex = 0.5, lwd = 0.25, lty = par("lty"),
#        scale = TRUE)
```

7.Best one

We think radarchart is good.Radar charts are best when we need to compare small amount of observations.Plotting one over the other we can come to conclusions.BUt if the data is dissimilar then it is hard to compare.

Assignment 2

1.Scatterplot

```
library(tidyverse)
library(RColorBrewer)
library(plotly)
library(ggplot2)

adults<-read.csv("adult.csv",sep=",")
names(adults)<-c("Age","Workclass","PolpulationIndex","Education","EducationNum","MaritalStatus",
,"Occupation","Relationship",
"Race","Sex","CapitalGain","CapitalLoss","HoursPerWeek","NativeCountry","Income
Level")

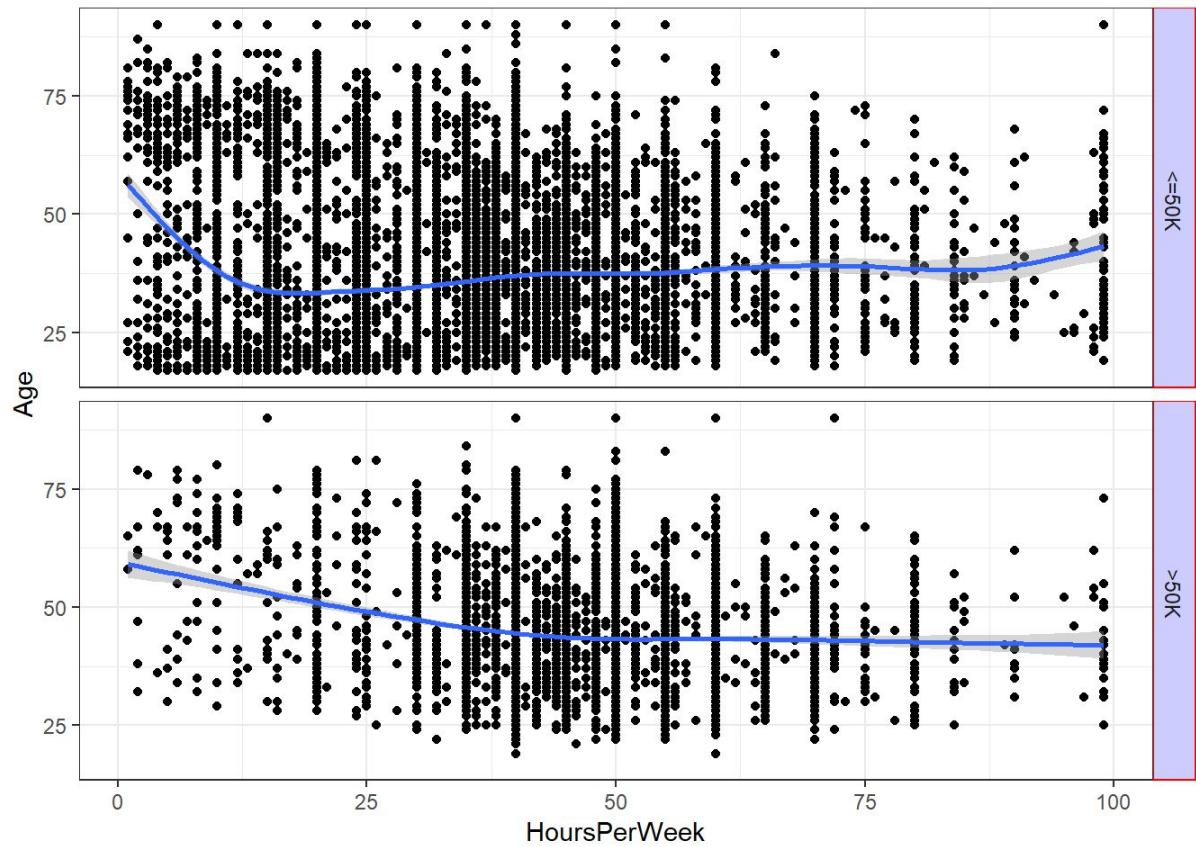
scl<-ggplot(adults,aes(x=HoursPerWeek,y=Age,col=IncomeLevel))+geom_point()
scl
```



trillis plot

```
tr1<-ggplot(adults,aes(x=HoursPerWeek,y=Age))+geom_point()+facet_grid(IncomeLevel~.)+theme_bw()+
  theme(strip.background = element_rect(colour="red", fill="#CCCCFF"))+geom_smooth()
tr1
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



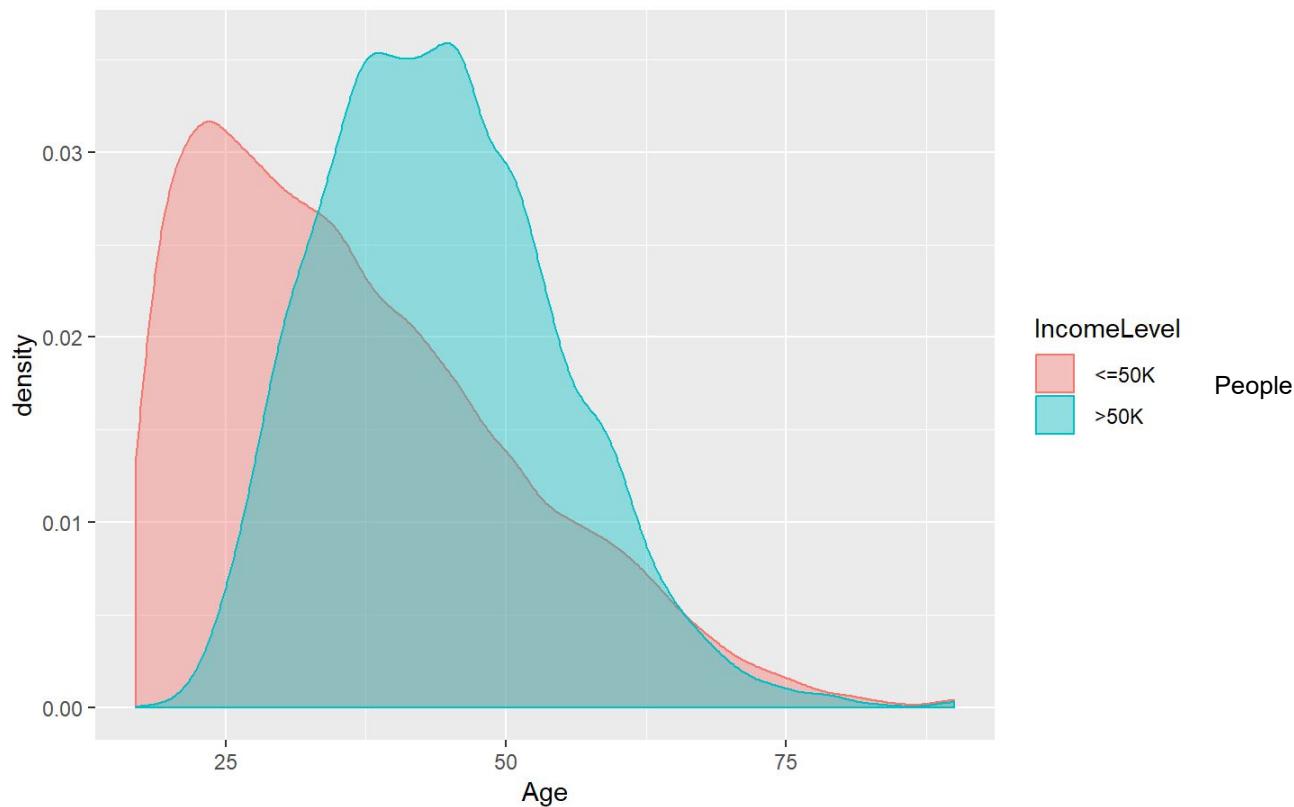
The first plot is a clear case of overplotting and hence difficult to make conclusions. People having less income work more hours per week. The people having less income also seem to work for a longer age. We can make this conclusion easily using trillis plot.

2.

Use ggplot2 to create a density plot of age grouped by the Income level.

```
d1<-ggplot(adults,aes(x=Age))+geom_density(aes(fill=IncomeLevel,color=IncomeLevel),alpha=0.4)+gg  
title("Density Plot")  
d1
```

Density Plot



earning less than 50k are younger age group mostly between 20 and 35 and those with income level greater than 50k are in range of age group 30-50

Create a trellis plot of the same kind where you condition on Marital Status

```
d2<-ggplot(adults,aes(x=Age))+
  geom_density(aes(col=IncomeLevel,fill=IncomeLevel),alpha=0.4)+facet_wrap(~MaritalStatus, label
  ler = "label_both")+ggtitle("Trellis Plot ")
d2
```



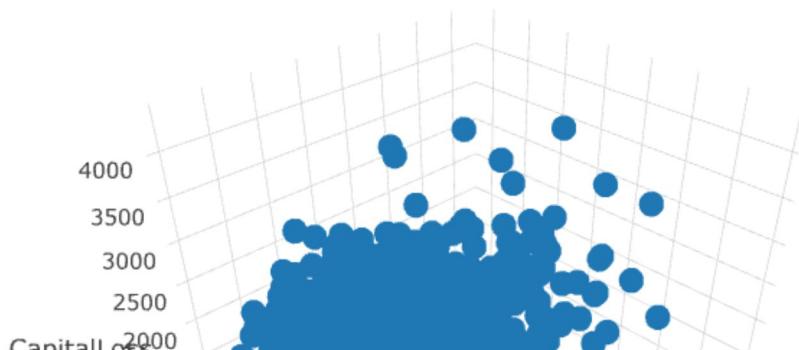
There are 7 different plots for different married status. Never married and Married AF spouse have the same pattern. Rest five plots follow another same pattern of income distribution for different age group and also these plots are similar to the normal density plot of income vs age group.

2.3

3 d scatter plot

```
pp <- adults %>% filter(CapitalLoss != 0) %>% plot_ly( x = ~EducationNum, y = ~Age, z = ~CapitalLoss)
%>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'EducationNum'),
                      yaxis = list(title = 'Age'),
                      zaxis = list(title = 'CapitalLoss')))
```

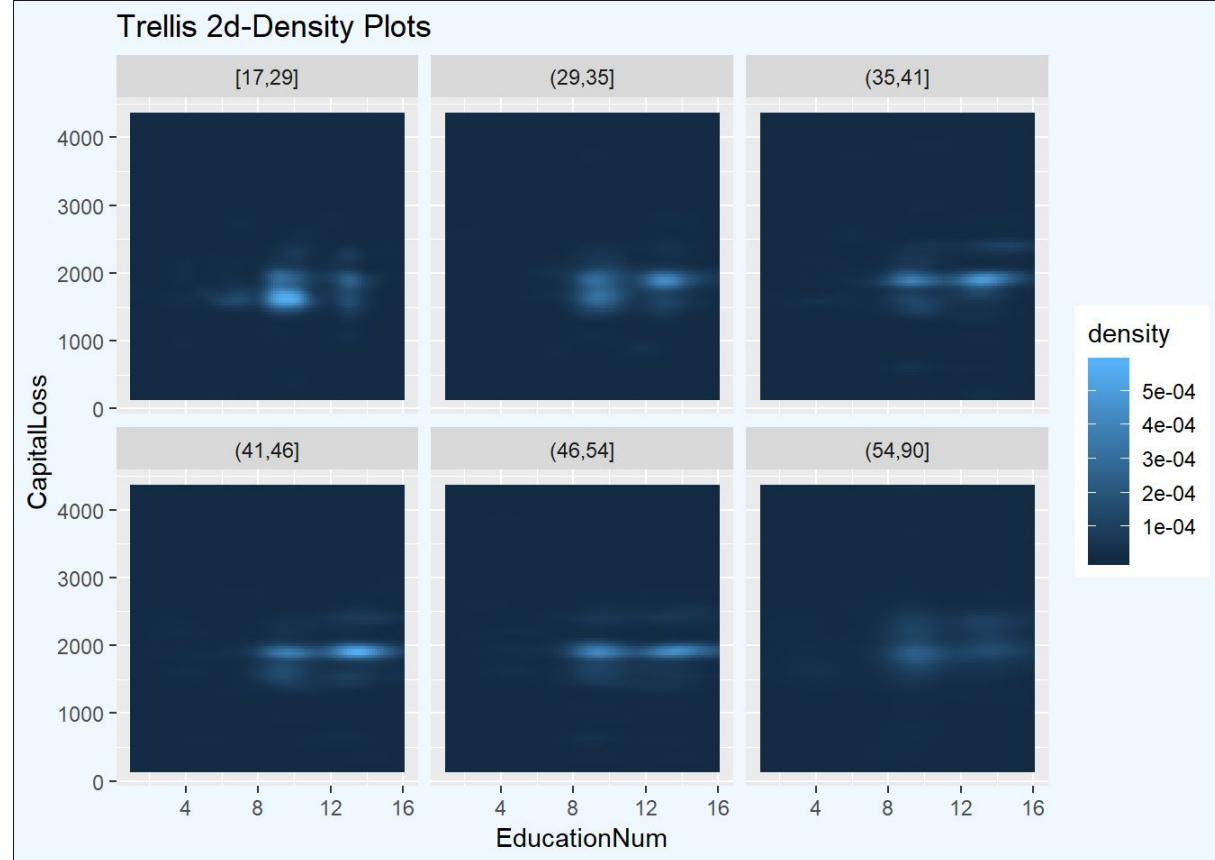
pp



The above plot is a clear example of overplotting. Hence we cannot get a clear understanding of the data.

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())

```
tr2<-adults[which(adults$CapitalLoss!=0), ]%>%ggplot(aes(y=CapitalLoss,x=EducationNum))+stat_density2d(aes(fill = stat(density)), geom = "raster",contour = FALSE)+facet_wrap(~cut_number(Age,n=6))
tr2+ggtitle("Trellis 2d-Density Plots") +theme_gray()%>%replace%theme(plot.background = element_rect(fill = "aliceblue"))
```



The data is better understandable in this plot when compared to the 3 d plot. We can say there is high density of capital loss around 2000 for age groups [17,29],(46,54].

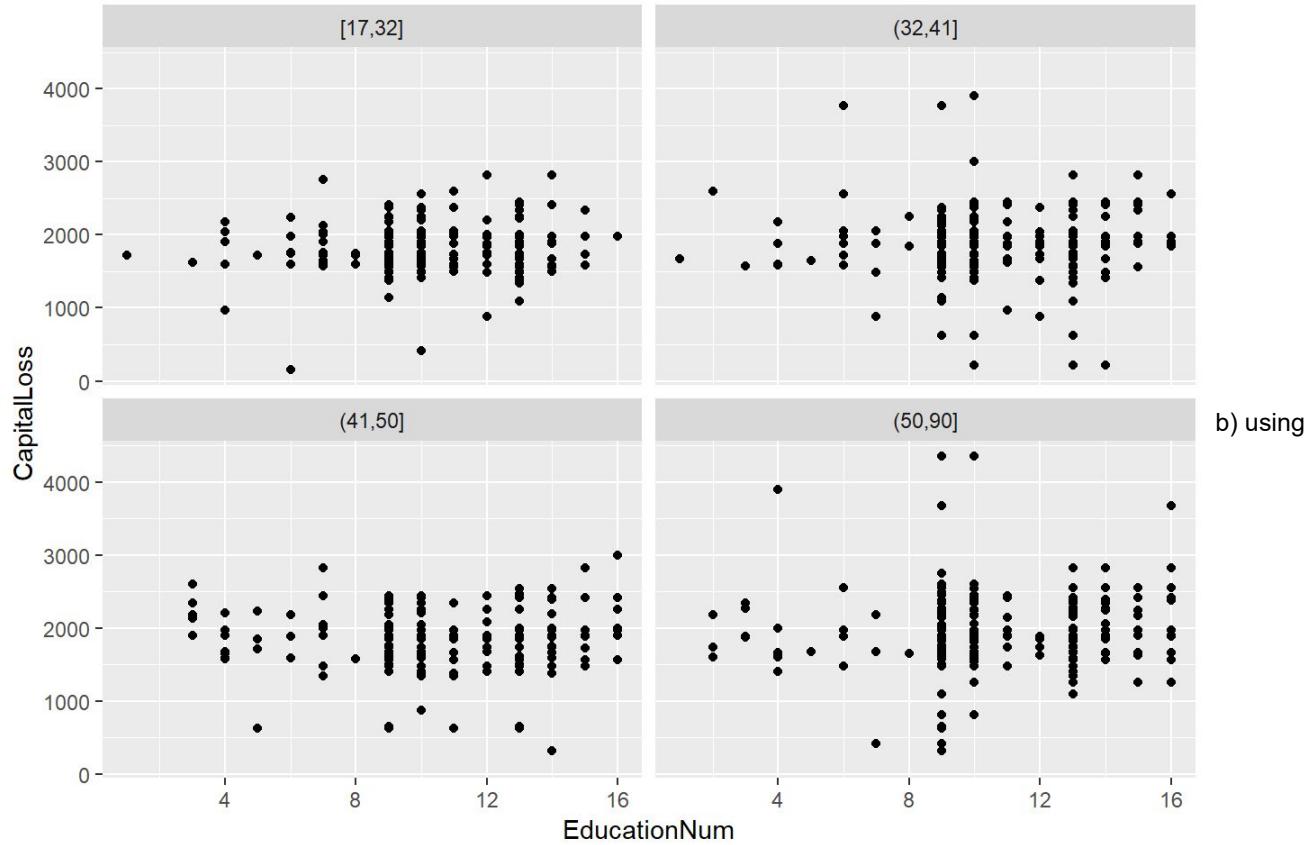
2.4

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()`

```
adults %>% filter(CapitalLoss != 0) %>% ggplot(aes(x=EducationNum, y=CapitalLoss)) + geom_point() + facet_wrap(~cut_number(Age, n=4))
```

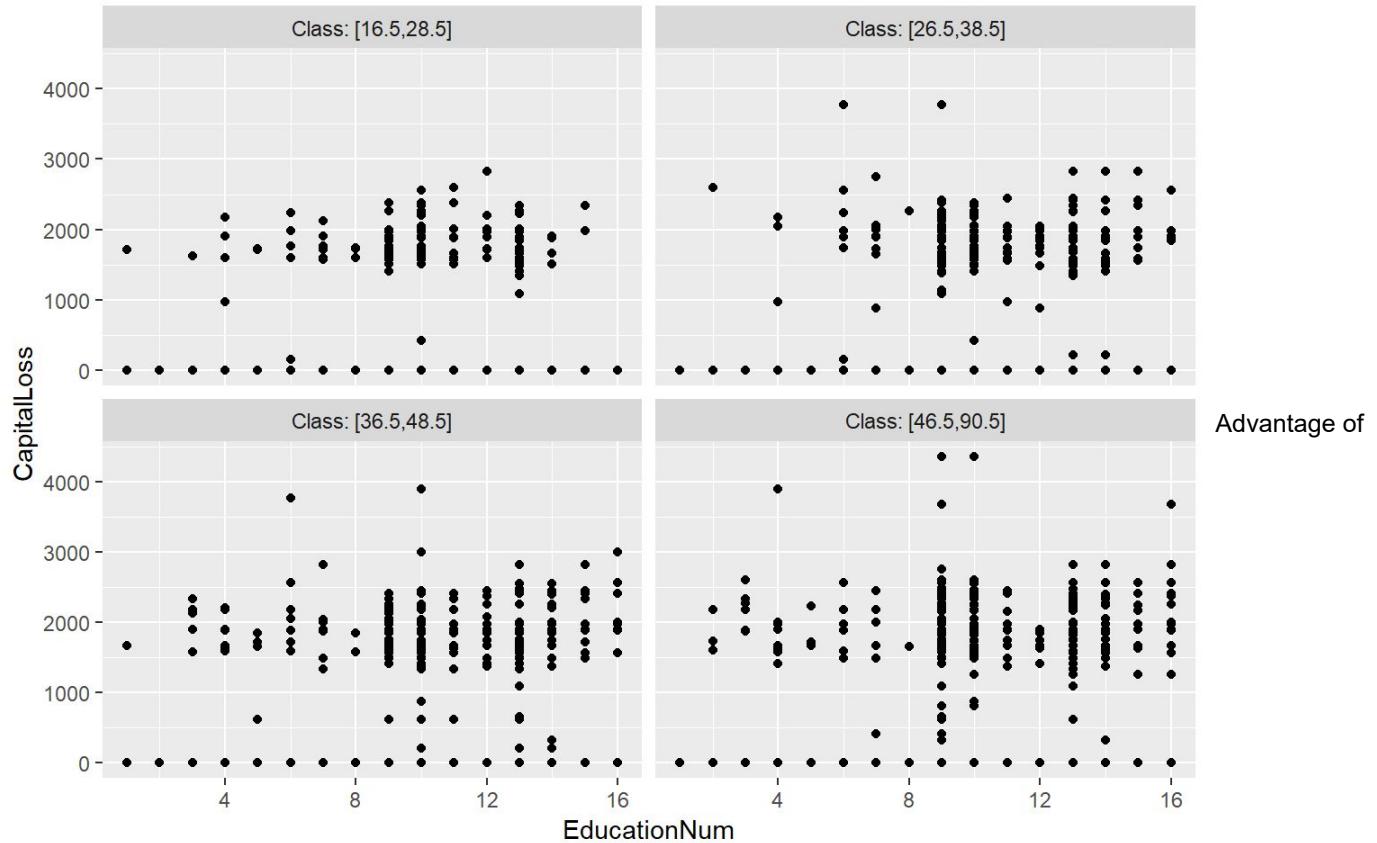


b) using

Shingles with 10% overlap.

```
Agerange<-lattice::equal.count(adults$Age, number=4, overlap=0.10) #overlap is 10%
L<-matrix(unlist(levels(Agerange)), ncol=2, byrow = T)
L1<-data.frame(Lower=L[,1],Upper=L[,2], Interval=factor(1:nrow(L)))
index=c()
Class=c()
for(i in 1:nrow(L)) {
  Cl=paste("[", L1$Lower[i], ", ", L1$Upper[i], "]", sep="")
  ind=which(adults$Age>=L1$Lower[i] & adults$Age<=L1$Upper[i])
  index=c(index, ind)
  Class=c(Class, rep(Cl, length(ind))) }

df4<-adults[index,]
df4$Class<-as.factor(Class)
h1<-ggplot(df4, aes(x=EducationNum, y=CapitalLoss)) + geom_point() + facet_wrap(~Class, labeller = "label_both")
h1
```



the shingles is that sudden jumps in data would not be missed if it occurred in the transition period .One disadvantage is that amount of coding is high and the next is that sometimes the data will be overlapped and hence be present in both plots and can lead to misleading calculations.

Appendix

```
library(tidyverse)
library(RColorBrewer)
library(plotly)
library(seriation)

df<-read.csv("prices-and-earnings.csv")
adults<-df[c(1,2,5,6,7,9,10,16,17,18,19)+1]

rownames(adults)<-df[,1]
coul = colorRampPalette(brewer.pal(8, "PiYG"))(25)
adscaled=scale(as.matrix(adults))
p2<-plot_ly(x=colnames(adscaled), y=rownames(adscaled), z=adscaled, type="heatmap", colors = coul)
p2

rowdist<-dist(adscaled,method = "minkowski", p=2)
coldist<-dist(t(adscaled),method = "minkowski", p=2)
order1<-seriate(rowdist, "HC")
order2<-seriate(coldist, "HC")
ord1<-get_order(order1)
ord2<-get_order(order2)
reordmatr_euc<-adscaled[rev(ord1),ord2]
plot_ly(x=colnames(reordmatr_euc), y=rownames(reordmatr_euc),
        z=as.matrix(reordmatr_euc), type="heatmap", colors =colorRamp(c("yellow", "red")))

v<-1:11
ord=sample(v)
#ord=ord2
dims0=list()
for( i in 1:ncol(adults)){
  dims0[[i]]=list( label=colnames(adults)[ord[i]],
                  values=as.formula(paste("~",colnames(adults)[ord[i]])))
}
p <-adults%>%
  plot_ly(type = 'parcoords',
          line = list(color = ~as.numeric(Hours.Worked)),dimensions = dims0)
p

radar_value <- reordmatr_euc%>%
  as.tibble(rownames = "name")%>%
  tidyrr::gather(variable, value, -name, factor_key=T)%>%
  mutate(name = factor(name,levels = rownames(reordmatr_euc)))

radar_value %>%
  ggplot(aes(x=variable, y=value, group=name)) +
  coord_polar() + theme_bw() + facet_wrap(~name, ncol=13) +
  theme(axis.text.x = element_text(size = rel(.5)))
Ps<-list()
for (i in 1:12){
  Ps[[i]] <- htmltools::tags$div(
    plot_ly(type = 'scatterpolar',
            r=as.numeric(reordmatr_euc[i,-1]),
            theta= colnames(reordmatr_euc)[-1],
            fill="toself")%>%
```

```
layout(title=rownames(reorderedmatr_euc)[i]), style="width: 25%;")  
}  
  
h <- htmltools::tags$div(style = "display: flex; flex-wrap: wrap", Ps)  
  
htmltools::browsable(h)  
  
library(tidyverse)  
library(RColorBrewer)  
library(plotly)  
  
adults<-read.csv("adult.csv",sep=",")  
names(adults)<-c("Age","Workclass","PopulationIndex","Education","EducationNum","MaritalStatus",  
,"Occupation","Relationship",  
"Race","Sex","CapitalGain","CapitalLoss","HoursPerWeek","NativeCountry","Income  
Level")  
  
### scatter plot of Hours per Week versus age where observations are colored by Income level.  
# Make a trellis plot of the same kind where you condition on Income Level  
  
sc1<-ggplot(adults,aes(x=HoursPerWeek,y=Age,col=IncomeLevel))+geom_point()  
sc1  
  
tr1<-ggplot(adults,aes(x=HoursPerWeek,y=Age))+geom_point()+facet_grid(IncomeLevel~.)+theme_bw() +  
theme(strip.background = element_rect(colour="red", fill="#CCCCFF"))+geom_smooth()  
tr1  
  
##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  
  
d1<-ggplot(adults,aes(x=Age))+geom_density(aes(fill=IncomeLevel,color=IncomeLevel),alpha=0.4)+gg  
title("Density Plot")  
d1  
  
d2<-ggplot(adults,aes(x=Age)) +  
  geom_density(aes(col=IncomeLevel,fill=IncomeLevel),alpha=0.4)+facet_wrap(~MaritalStatus, label  
ler = "label_both")+ggtitle("Trellis Plot ")  
d2  
  
##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 Capital Loss  
  
pp <- adults%>%filter(CapitalLoss!=0)%>%plot_ly( x = ~EducationNum, y = ~Age, z = ~CapitalLoss,c  
olor=~EducationNum) %>%  
  add_markers() %>%  
  layout(scene = list(xaxis = list(title = 'EducationNum'),  
                      yaxis = list(title = 'Age'),  
                      zaxis = list(title = 'CapitalLoss')))  
pp  
  
###Create a trellis plot with 6 panels in ggplot2 in which each panel shows a raster-type 2d-den  
sity plot of Capital Loss
```

```
####versus Education-num conditioned on values of Age (use cut_number())  
  
tr2<-adults[which(adults$CapitalLoss!=0), ]%>%ggplot(aes(x=CapitalLoss,y=EducationNum))+geom_dens  
ity_2d()+facet_wrap(~cut_number(Age,n=6))  
tr2+ggtitle("Trellis 2d-Density Plots")+theme_gray()%+replace%theme(plot.background = element_re  
ct(fill = "aliceblue"))  
  
###Make a trellis plot containing 4 panels where each panel should show a scatter plot of Capita  
l Loss versus Education-num  
##conditioned on the values of Age by a) using cut_number() b) using Shingles with 10% overlap.  
  
#a)  
adults%>%filter(CapitalLoss!=0)%>%ggplot(aes(x=CapitalLoss,y=EducationNum))+geom_point()+facet_w  
rap(~cut_number(Age,n=4))  
  
#b)  
  
#cap_adults<-adults[which(adults$CapitalLoss!=0),]  
Agerange<-lattice::equal.count(adults$Age, number=4, overlap=0.10) #overlap is 10%  
L<-matrix(unlist(levels(Agerange)), ncol=2, byrow = T)  
  
index=c()  
Class=c()  
for(i in 1:nrow(L)){  
  Cl=paste("[", L1$Lower[i], ", ", L1$Upper[i], "]", sep="")  
  ind=which(df3$age>=L1$Lower[i] &df3$age<=L1$Upper[i])  
  index=c(index,ind)  
  Class=c(Class, rep(Cl, length(ind))) }  
  
df4<-adults[index,]  
df4$Class<-as.factor(Class)  
h1<-ggplot(df4, aes(x=CapitalLoss, y=EducationNum))+ geom_point()+ facet_wrap(~Class, labeller =  
"label_both")  
h1
```

lab5

Andreas and Priya

11 October 2018

##Assignment 1

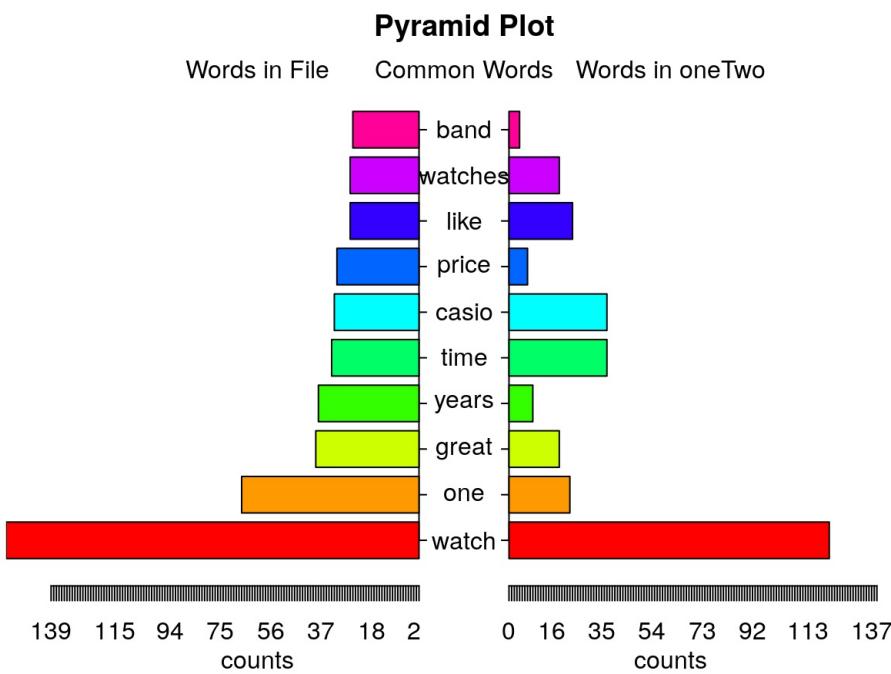
###Word Cloud Plot for five



###Word Cloud Plot for One Two



###Pyramid Plot



```
## [1] 5.1 4.1 4.1 2.1
```

The word cloud for five.txt which is the good feedback has more frequent words like watch,great,one,price,casio,years,battery etc.This makes sense as words watch and casio will be often coming in the words as this is the product.As it is a positive feedback,people are happy with the price.Great is a posisive word and may be telling it last for long with word years.

The oneTwo.txt is the negative feedback and the frequent words are amazon,one,back,watch,casio,battery etc.casio and watch comes frequent like before as this is the product.Amazon ,may be because they are not happy with the amazon more than product(may be the deal,return policy,shipping etc).They would like to return back the product and hence **back** has come multiple times.

###Phrasenets

Phrase nets of five.text

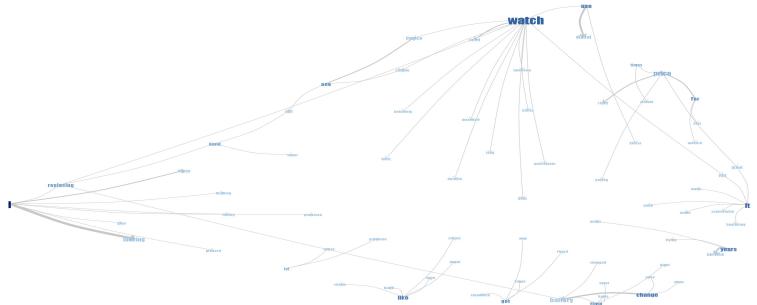


Figure 1. positive.

phrase nets of negative words

Figure 2. negative

word trees

word tree of battery with the positive feedback

battery and **replacement** changed for the period. Oh, I think the rubber wristband dried up and broke a few years. Just battery replacement for the period. Casio makes this same watch with the atomic movement, solar power. I wanted to make sure that my son would like this style and size. A couple of times. The watch repair guy said that once he opens up the watch to change the battery, it won't stay waterproof. So when he goes to swim, he just takes it off. Then the crystal broke for so 3 or 4 times. The last time the battery was changed, the jeweler told me the push button won't stay waterproof. Maybe I'm prejudiced, but for more than 25 years I've depended on my Casios and they have never failed me. The battery lasted 11 years. Replaced it 3-4 years ago still working. Its slim and love it also. Dropped it face down on concrete. One of the bands came loose. Dont drop this watch face about that many straps. I've traveled the planet, dived in three oceans with it, it has gone through change, no service, no nothing. My wife's Cartier keeps breaking and needs a 700\$ service every 5 years. goes. I just got a new one cause its rare when a watch shop makes it waterproof again after replacing the battery how many watches go thru more than one generation of batteries? I think my black one (about 15 years) every few years are minor issues. Pretty good for the money. In Australia, it cost about 3 times more. I was changed, the jeweler told me the pushers had become clogged. He cleaned them, and I wore it another 1 life around 2 years. Casio makes great watches. Also have a casio digital that came w/lithium battery. The battery lasted 11 years. Replaced it 3-4 years ago still working. Its slim and love it also. If were to spend more money, I dropped and broke it. So, I finally got a new one, the black face version this time for a change of pa

still works fine and with great accuracy after quite a few years. Just battery replacement for the period. Oh, I think the rubber wristband dried up and broke at one point after years of wearing it. I had to replace it. It is beginning to show some wear and tear - mostly scratches on the crystal. The rest of the case seems to be fine. But nice complements made about the old watch recently - never had that happen before. So that's why I got another one with the effect of free shipping. I like the price, it's a solid watch. Sturdy and durable. You never have to worry about it , like you would an expensive watch. This product has 5 stars is very nice, excellent product. i recommend it to every bahu

Figure 4. wordtree battery pos feedback

word tree with the word battery in negative feedback

battery and after that it start to be slower I dont know what the hell..... this did not fix the problem, so I returned the watch to the store and exchanged it for an id all seems well. You do need special tool to get back off of watch. I will update if it quits working. The analog portion does not keep time. I replaced the battery and this did not fix the problem, so I If you just have someone replace the battery without pressure testing it will fail. This watch was given At the least I would recommend trying this monitor on before purchasing it to make sure you are changed and it worked well for a few months, but then stopped again. Sad, because I love the look. shortly after PURCHASING this watch. After replacing the battery, it did not too long after that so I'm done. It did not too long after that so I'm done fooling with it. It's a shame because it's a really nice looking watch quit in less than 12 months and after it was replaced, the watch died completely. Looks and price are my top priority. The watch is still under warranty, but I'm just going to return it and get my money back from Casio. I had the battery replaced and the watch wouldn't keep time. I returned the watch to Casio for repair and replaced the watch wouldn't keep time. I returned the watch to Casio for repair and was told it would be \$ only last a year. It will cost more than the watch to properly replace the battery. If you just have someone replace, work fine for another couple of months, then the same problem happened again - hence the issue because it will run fine for days keeping time well and then I will look at it a day later and then it will be already. The back is designed to be opened with special tools so you cannot do it yourself making it a bit more in the first watch. Had this watch for one and a half years when the push buttons started sticking while depre

Figure 4. wordtree battery neg feedback

Analysis

Looking at the above graphs we come to conclusions like

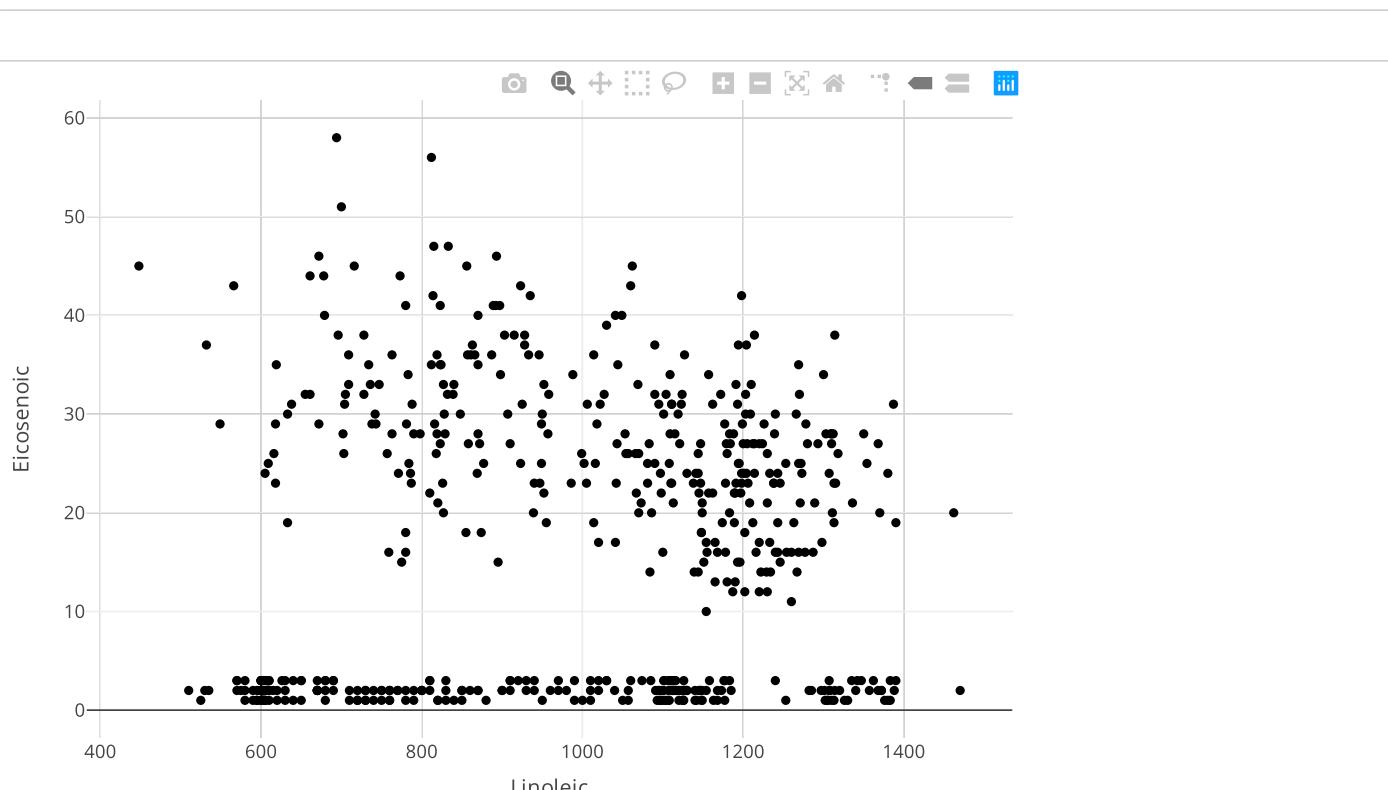
The watch has analog and digital display and it appears in black,gold and white. It has sporty look,rubber band and waterprrof and good value for money. The satisfied customers are talking about modern look,water resistance,some of them say the battery last long and it is good priced. Disatisfied customers talk about the defective alrams,battery once changed the watch does not work well,The analog portions becomes defective etc. Also Replacing battery is costly .

Seems like the customers are satisfied with the look and feel and price of the watch .With some changes in the battery and all it can be a good product.

Assignment 2

1.Scatter Plot eicosenoic vs linoleic

Choose region



The value of eicosenoic is between one and three. # ## 2.Linked Scatterplot and Bar chart

values of stearic



Brush color

rgba(228, 1

152 248 344

60

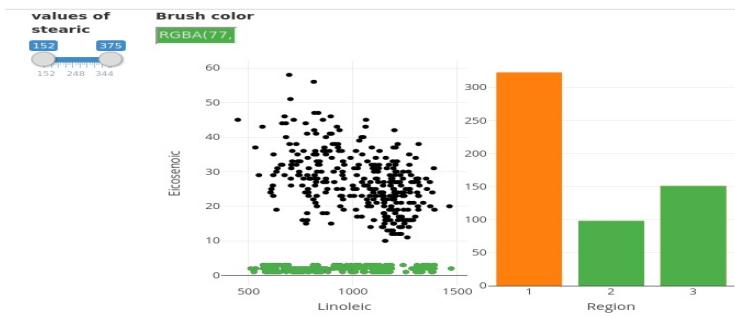
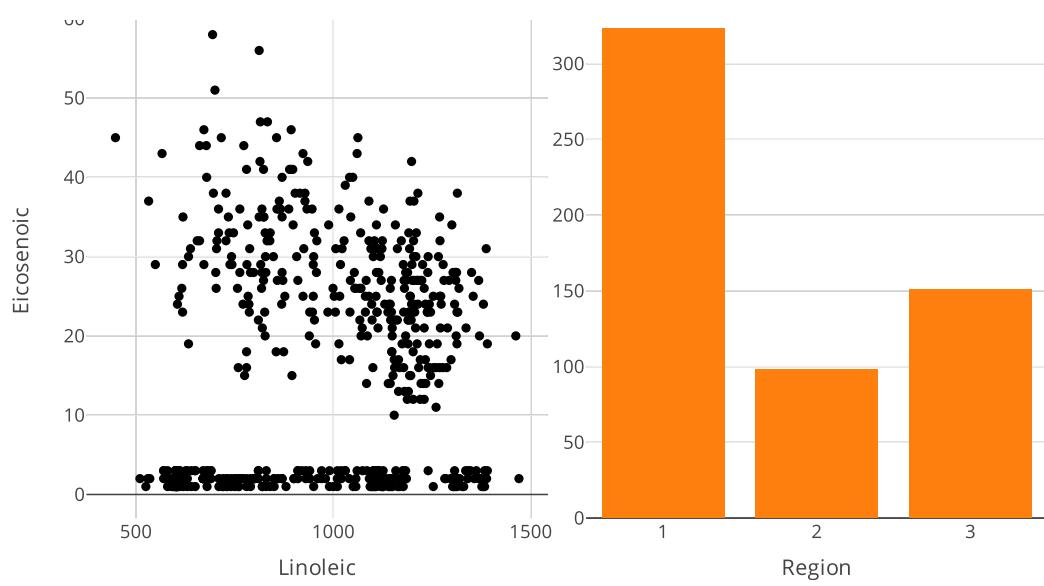


Figure 1. coloured

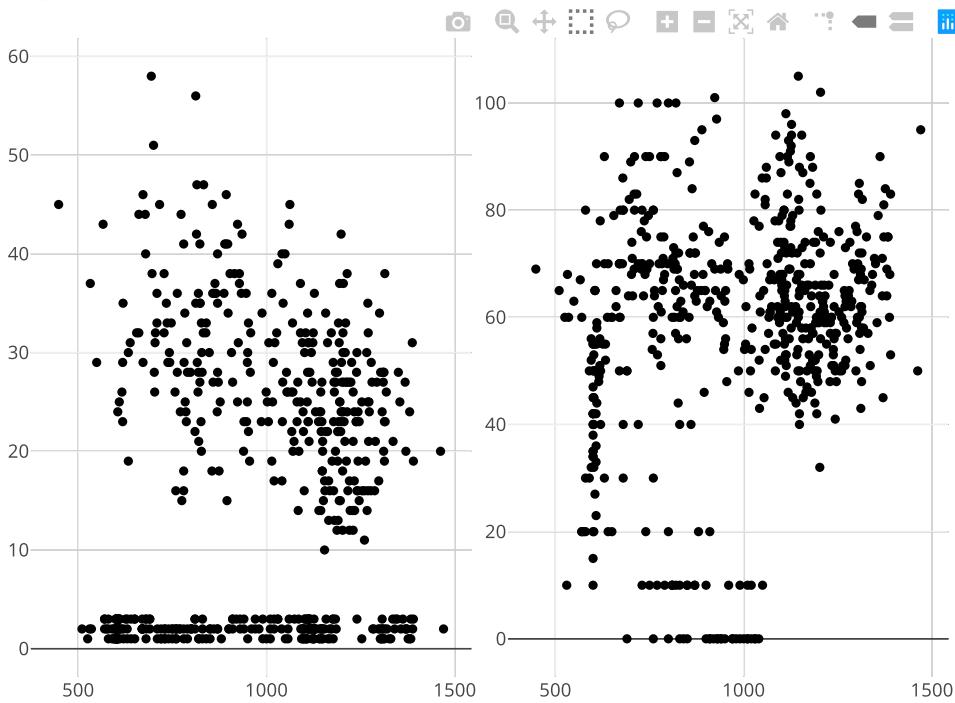
Using brushing when we take out the lower values of eicosenoic(between 1 and 3),we can see they correspond to region south and sardinia Island.In south region the amount of stearic acid is between 190 and 270.

The operators used here are selection operators for brushing,Connection operator for connecting the plots and Filtering operator for filtering stearic values

##3.Linked Scatter Plots

Brush color

rgba(228, 228, 228)



Brush color
rgba(228,2)

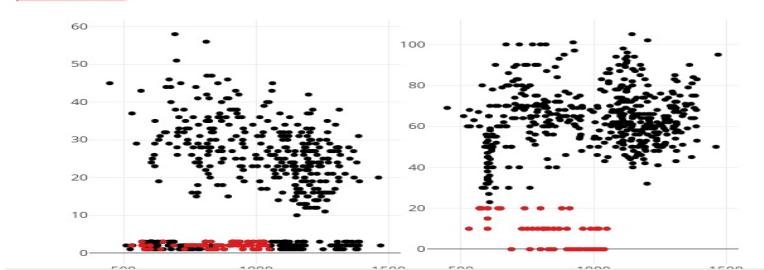


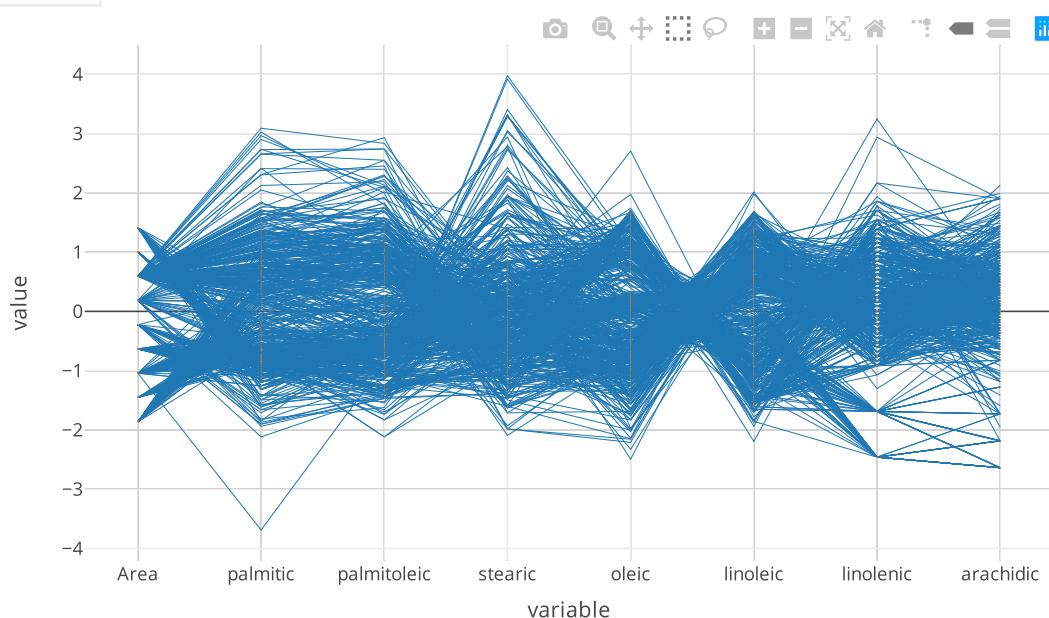
Figure 1. linked scatter

The values of arachidic below 40 are outliers in the plot2 are also outliers in plot 1. IN plot one their eicosenoic values are between 1 and 3.

####4.Parallel Coordinate Plot linked Barplot and Scatterplot

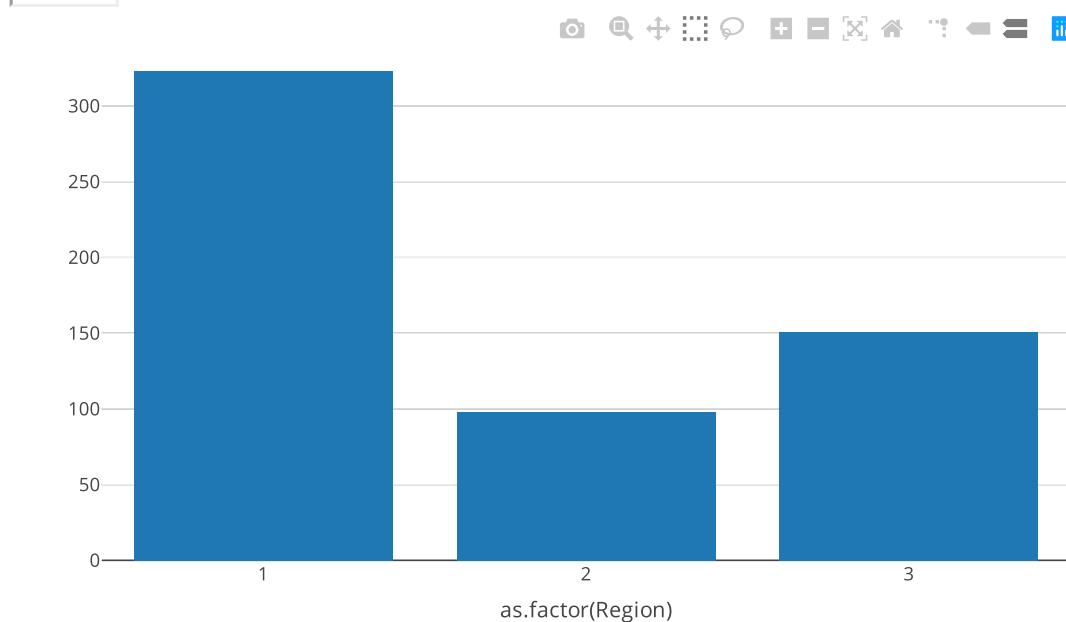
Brush color

rgba(228,2)



Brush color

rgba(228,2)



Brush color

rgba(228,2)

palmitic ▼

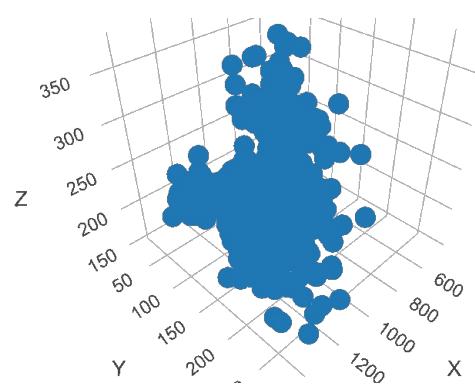
Distribution of Oil



palmitic ▼



palmitic ▼



When we analysed the parallel coordinate plot, linolenic, oleic and palmitic seemed to have more clusters hence they are influential variables to differentiate regions.

In South region we can observe multiple clusters. linolenic, oleic and stearic forms cluster, steraric, oleic, linoleic forms another cluster.

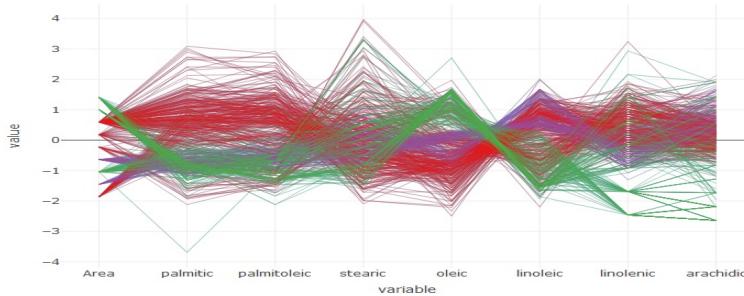


Figure 1. plot_123

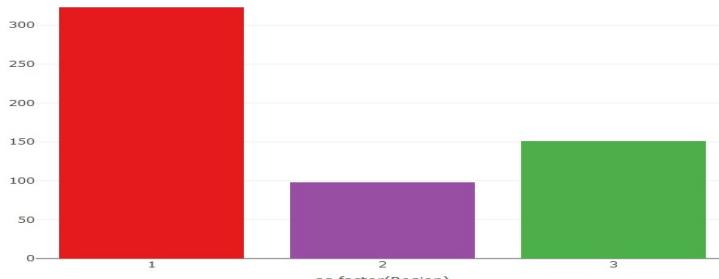


Figure 1. plot_123

oleic ▼

Distribution of Oil

linolenic ▼

palmitic ▼

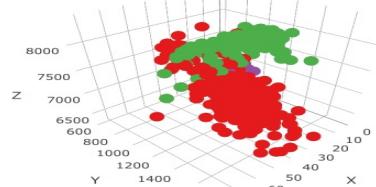


Figure 1. plot_123

As seen in the above screenshots, when we select the 3 variables and brush the regions with 3 different colours, the plot is almost clustered in 3 separate groups.

##5.Analysis

In step 4 we have used:

selection operator while brushing the plots ,connection operator to connect plots, Reconfiguring operator dynamically selecting axis variables in 3D plot on data value space operand

An additional interaction operator which can be used is Filtering operator.This operator can be used to filter based on regions and analyse the concentration of acids in oil from each region. This data can be used for analysing the region of origin of an oil given the concentration of acids in that oil.s

##Appendix

```
## ----message=FALSE,echo=FALSE-----  
library(tm)  
library(wordcloud)
```

```

library(RColorBrewer)
#library(viridisLite)
library(plotrix)
library(tidyverse)

## ----message=FALSE,echo=F-----
five<-read.table("Five.txt",header=F, sep='\n',encoding="latin1")#Read file
five$doc_id=1:nrow(five)
colnames(five)[1]<-"text"
mycorpus <- Corpus(DataframeSource(five)) #Creating corpus (collection of text data)

clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus,function(x) removeWords(x,stopwords("english")))
  return(corpus)
}

mycorpus <- clean_corpus(mycorpus)
tdm <- TermDocumentMatrix(mycorpus) #Creating term-document matrix
m <- as.matrix(tdm)

#here we merge all rows
v <- sort(rowSums(m),decreasing=TRUE) #Sum up the frequencies of each word
d <- data.frame(word = names(v),freq=v) #Create one column=names, second=frequencies

pal<-brewer.pal(6,"PRGn") #Create palette of colors
#color_pal <- cividis(n = 8)

wordcloud(d$word,d$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))

wordcloud(d$word,d$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))

## ----message=F,echo=F-----
oneTwo<-read.table("OneTwo.txt",header=F,sep="\n")
oneTwo$doc_id=1:nrow(oneTwo)
colnames(oneTwo)[1]<-"text"

mycor <- Corpus(DataframeSource(oneTwo))

mycor <- clean_corpus(mycor)
tdm1<- TermDocumentMatrix(mycor) #Creating term-document matrix
m1 <- as.matrix(tdm1)

#here we merge all rows
v1 <- sort(rowSums(m1),decreasing=TRUE) #Sum up the frequencies of each word
d1<- data.frame(word = names(v1),freq=v1)
pal<-brewer.pal(8,"Dark2")

wordcloud(d1$word,d1$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))

## ----message=FALSE,warning=FALSE,echo=F-----
j<-left_join(d,d1,by="word")

pyramid.plot(
  j$freq.x[1:10],j$freq.y[1:10],
  # Words
  labels = j$word[1:10],
  top.labels = c("Words in File", "Common Words", "Words in oneTwo"),
  main = "Pyramid Plot",gap=17,unit="counts")

## ----message=FALSE,echo=FALSE-----
library(tidyverse)
library(plotly)
library(plyr)
library(crosstalk)
library(GGally)

```

```

olive<-read.csv("olive.csv")
olive$Region<-as.factor(olive$Region)
levels(olive$Region)<-c("1","2","3")
o<-SharedData$new(olive)
oo<-SharedData$new(olive,~Region,group = "Choose region")

## ----set-options,message=F,echo=F-----
scatterolive <- plot_ly(oo, y = ~eicosenoic, x = ~linoleic)%>%group_by(Region)%>%
  add_markers(color = I("black"),name="hollow")%>%highlight(on="plotly_hover",persistent = F,selectize = T) %>% l
ayout(xaxis = list(title = "Linoleic"),
      yaxis = list(title = "Eicosenoic"))

scatterolive

## ----message=FALSE,warning=FALSE,echo=F-----

barolive <- plot_ly(oo, x=~as.factor(Region))%>%add_histogram()%>%layout(barmode="overlay",xaxis=list(title="Regio
n"))

bscols(widths=c(2, NA),filter_slider("stearic", "values of stearic", oo, ~stearic),subplot(scatterolive,barolive,t
itleY = TRUE, titleX = TRUE)%>%highlight(on="plotly_select", dynamic=T, persistent = T, opacityDim = I(1))%>%hide
_legend())

## ----message=FALSE,echo=F-----
scatter1 <- plot_ly(o, x = ~linoleic, y = ~eicosenoic) %>%
  add_markers(color = I("black")) %>% layout(xaxis = list(title = "linoleic"),
                                                yaxis = list(title = "Eicosenoic"))
scatter2 <- plot_ly(o, x = ~linoleic, y = ~arachidic) %>%
  add_markers(color = I("black"))%>% layout(xaxis = list(title = "linoleic"),
                                                yaxis = list(title = "arachidic"))

subplot(scatter1,scatter2)%>%
  highlight(on="plotly_select", dynamic=T, persistent=T, opacityDim = I(1))%>%hide_legend()

## ----message=FALSE,fig.width=20, fig.height=18,echo=F-----
oliveparallelcord<-ggparcoord(olive, columns = c(3:10))

d<-plotly_data(ggplotly(oliveparallelcord))%>%group_by(.ID)
d1<-SharedData$new(d, ~.ID, group="olive")

p1<-plot_ly(d1, x=~variable, y=~value)%>%
  add_lines(line=list(width=0.3))%>%
  add_markers(marker=list(size=0.3),
              text=~.ID, hoverinfo="text")

oildata <- olive
oildata$.ID=1:nrow(oildata)
d2<-SharedData$new(oildata, ~.ID, group="olive")
p2 <- plot_ly(d2, x=~as.factor(Region))%>%add_histogram()%>%layout(barmode="overlay")

ButtonsX=list()
for (i in 4:10){
  ButtonsX[[i-3]]= list(method = "restyle",
                        args = list( "x", list(olive[[i]])),
                        label = colnames(olive)[i])
}

ButtonsY=list()
for (i in 4:10){
  ButtonsY[[i-3]]= list(method = "restyle",

```

```

        args = list( "y", list(olive[[i]])),
        label = colnames(olive)[i])
    }

ButtonsZ=list()
for (i in 4:10){
  ButtonsZ[[i-3]]= list(method = "restyle",
                        args = list( "z", list(olive[[i]])),
                        label = colnames(olive)[i])
}

axx <- list(
  title = "X"
)

axy <- list(
  title = "Y"
)

axz <- list(
  title = "Z"
)
p3 <- plot_ly(d2,x= ~linoleic,y = ~palmitoleic, z = ~stearic)%>%add_markers()%>%
  layout(xaxis=list(title=""), yaxis=list(title=""),
         title = "Distribution of Oil",
         updatemenus = list(
           list(y=0.9, buttons = ButtonsX),
           list(y=0.6, buttons = ButtonsY),
           list(y=1.2, buttons = ButtonsZ)
         )%>%layout(scene = list(xaxis=axx,yaxis=axy,zaxis=axz))

ps<-htmltools::tagList(p1%>%
                          highlight(on="plotly_select", dynamic=T, persistent = T, opacityDim = I(1))%>%
                          hide_legend(),
                          p2%>%
                          highlight(on="plotly_select", dynamic=T, persistent = T, opacityDim = I(1))%>%
                          hide_legend(),
                          p3%>%
                          highlight(on="plotly_click", dynamic=T, persistent = T, opacityDim = I(1))%>%
                          hide_legend())
)
htmltools::browsable(ps)

```

Lab6

Andreas

29 Οκτωβρίου 2018

Assignment 1

This assignment is all about network visualisation of the terrorist connections.

The files given are *trainData.dat* and *trainMeta.dat*. The files have the data about the network of people involved in Madrid bombing.

We plot a graph using the *visNetwork* package

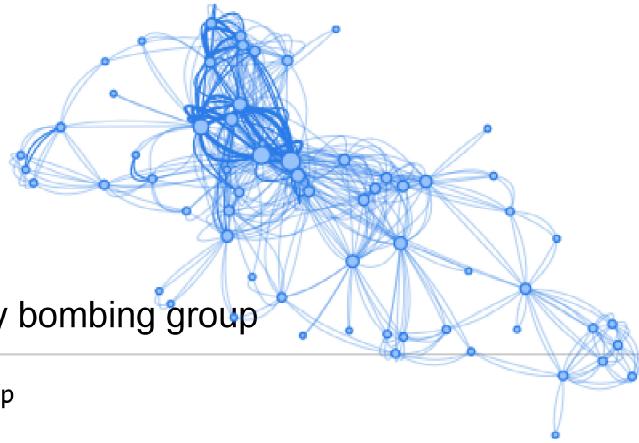
The basic graph is as below

```
library(ggraph)
library(igraph)
library(visNetwork)
library(seriation)
library(plotly)

#setwd("C:/Users/quartermaine/Documents/Visualization/Lab_6")
edges <- read.delim("trainData.dat", header = FALSE, sep = " ")
nodes <- read.delim("trainMeta.dat", header = FALSE, sep = " ")
set.seed(12345)

nodes$id <- rownames(nodes)
colnames(nodes) <- c("bombers", "b_group", "id")
colnames(edges) <- c("temp", "from", "to", "value")
edges$temp <- NULL
graph <- graph.data.frame(edges, directed = T)
degree_value <- degree(graph)
nodes$value <- degree_value[match(nodes$id, names(degree_value))]
nodes <- na.omit(nodes)
nodes$label<-nodes$bombers
#### a.Basic
visNetwork(nodes=nodes,edges=edges,main = "Madrid bombing people network")
```

Madrid bombing people network

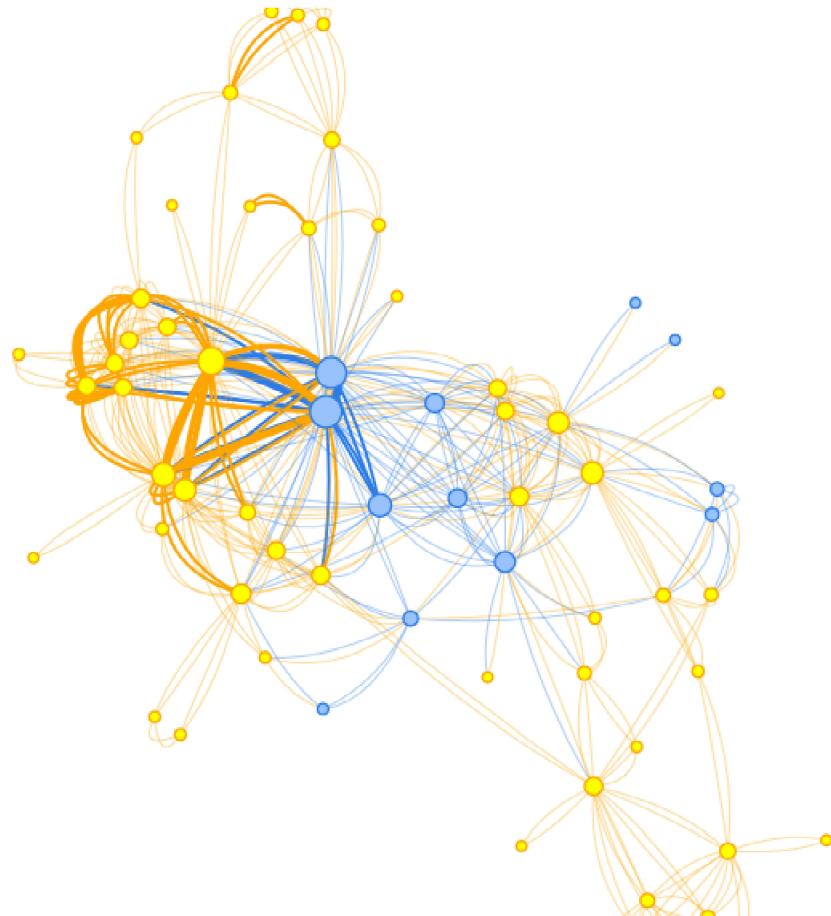


b.nodes are coloured by bombing group

```
nodes$group<-nodes$b_group
```

```
visNetwork(nodes=nodes,edges=edges,main = "Madrid bombing people network")
```

Madrid bombing people network



c. Strength

Strength can be also showed by degree

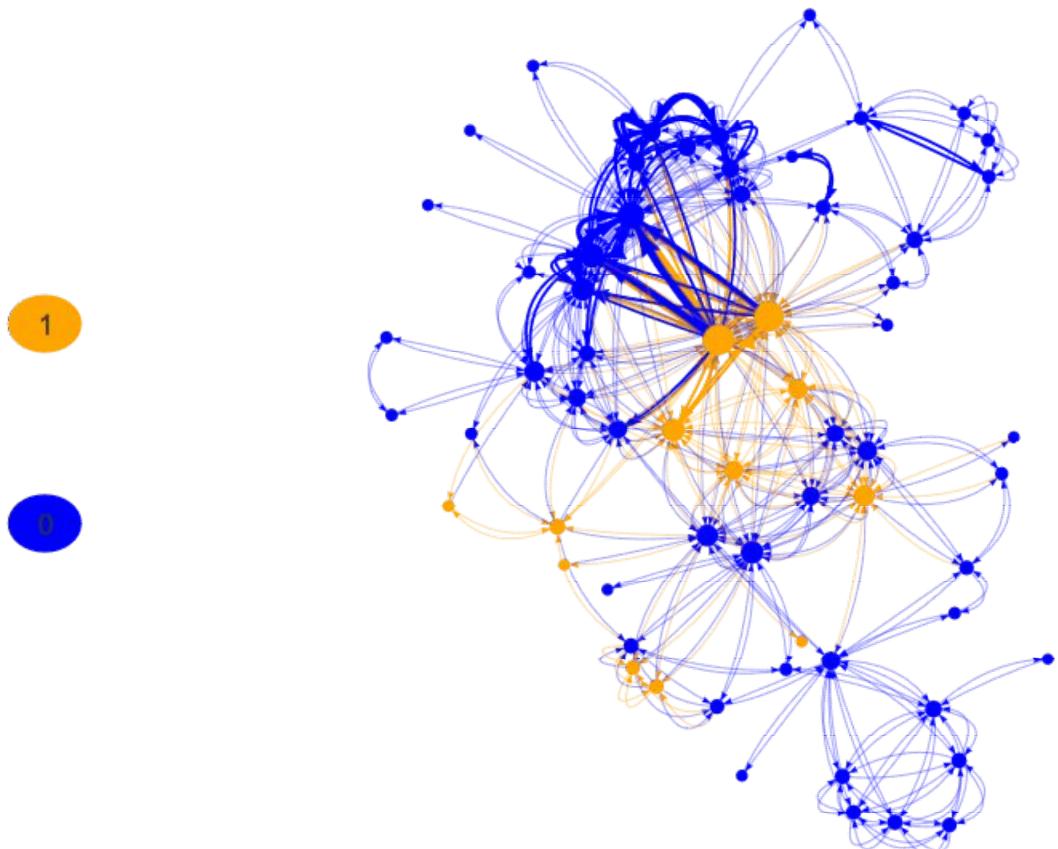
```
graph <- graph.data.frame(edges, directed = T)
degree_value <- degree(graph)
```

d.replusion

```
visNetwork(nodes = nodes, edges = edges, main = "Madrid bombing people network") %>%
  visGroups(groupname = "0", color = "blue") %>%
  visGroups(groupname = "1", color = "orange") %>%
  visEdges(arrows = "from") %>%
  visOptions(collapse = TRUE,
             selectedBy = "group") %>%
  visPhysics(solver= "repulsion") %>%
  visLegend() %>% addFontAwesome()
```

Madrid bombing people network

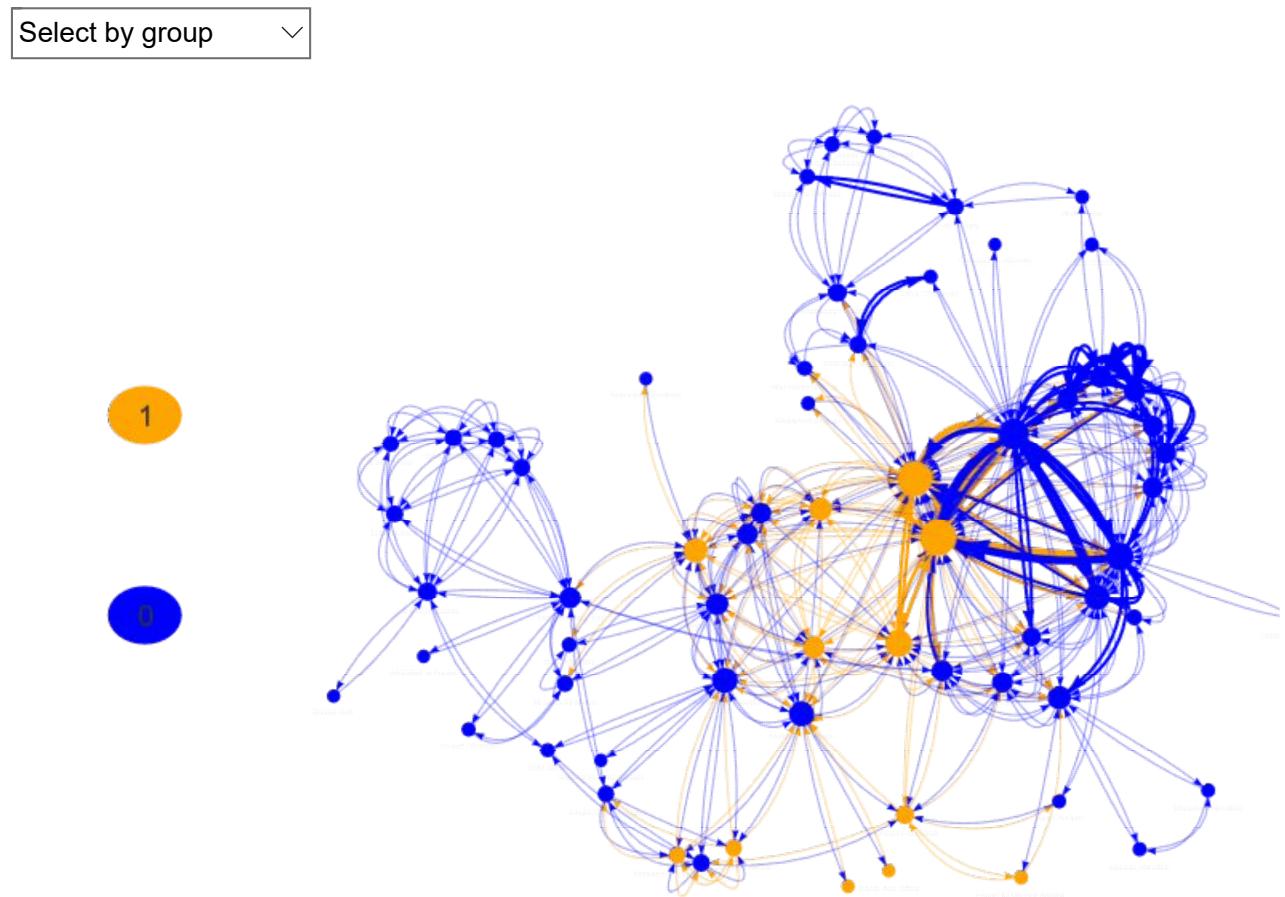
Select by group ▾



e.Highlighted nodes ans the end graph

```
visNetwork(nodes = nodes, edges = edges, main = "Madrid bombing people network") %>%
  visGroups(groupname = "0", color = "blue") %>%
  visGroups(groupname = "1", color = "orange") %>%
  visEdges(arrows = "to") %>%
  visOptions(highlightNearest = list(enabled = TRUE,
                                     degree = 1),
             collapse = TRUE,
             selectedBy = "group") %>%
  visPhysics(solver = "repulsion") %>%
  visLegend() %>% addFontAwesome()
```

Madrid bombing people network



Group 1 is for those involved in blasting and 0 for others. We could see one main cluster centered around Mohamed chaoui,Jamal Zougam,Basel chayoun ,SB abdelmajid Fakher. They are the people actively involved in bombing. Also there are three other small clusters of family members one with abdel karim,Tayser,mohamed bekkali etc.another with seeman gaby eid,el gitano etc and last one with mohamed chedadi,mohamed oulat akcha etc.

2.

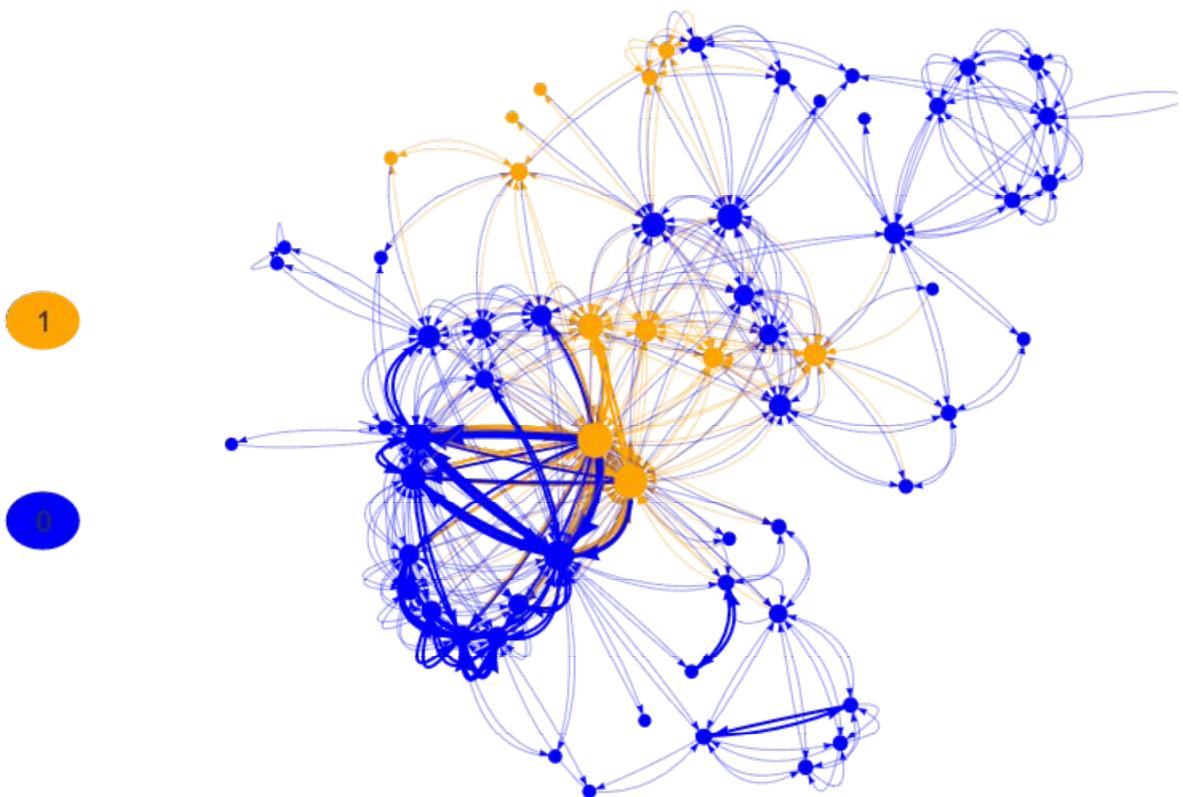
degree = list(from = 1, to = 2)) gives the highlight required

```

visNetwork(nodes = nodes, edges = edges) %>%
  visGroups(groupname = "0", color = "blue") %>%
  visGroups(groupname = "1", color = "orange") %>%
  visEdges(arrows = "from") %>%
  visOptions(highlightNearest = list(enabled = TRUE,
                                      degree = list(from = 1, to = 2)),
             collapse = TRUE,
             selectedBy = "group") %>%
  visPhysics(solver= "repulsion") %>%
  visLegend() %>% addFontAwesome()

```

Select by group ▾



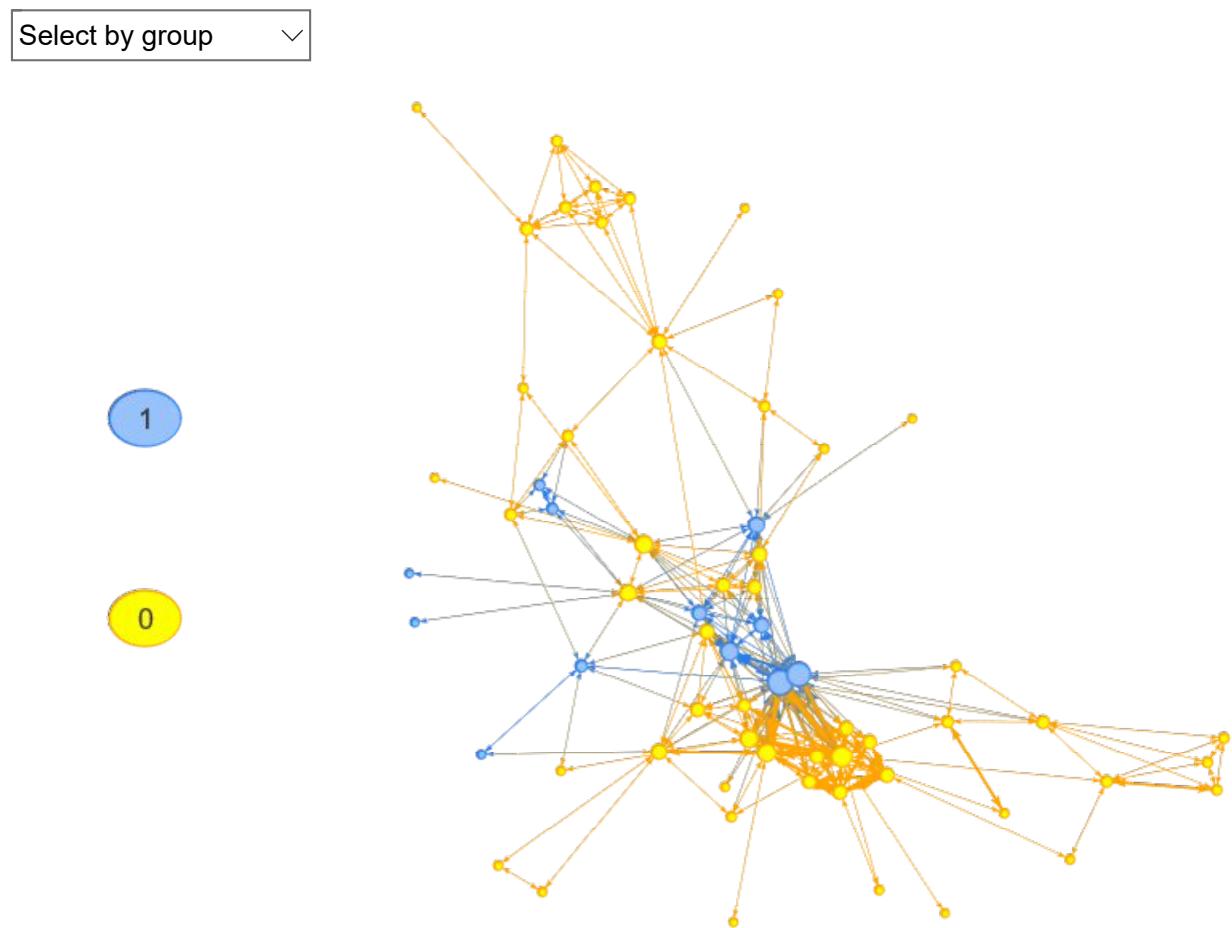
We could see jamal Zougam and mohamed chaoui were best in spreading news. The BBC 18-4-2014 (<http://news.bbc.co.uk/2/hi/europe/3515790.stm>) shows him as an early suspect of the attack and he owned a mobile shop so he could be the point of contact to get all the fake sim cards that connected the people.

3.

```
graph_data_frame <- graph.data.frame(edges, directed = FALSE)
clusters <- cluster_edge_betweenness(graph_data_frame, directed = T)
nodes$clusters <- clusters$membership

visNetwork(nodes = nodes, edges = edges, main = "Madrid Bombing") %>%
  visEdges(arrows = "from") %>%
  visOptions(highlightNearest = list(enabled = TRUE,
                                      degree = list(from = 1, to = 2)),
             collapse = TRUE,
             selectedBy = "group") %>%
  visPhysics(solver= "repulsion") %>%
  visLegend() %>% addFontAwesome() %>% visIgraphLayout()
```

Madrid Bombing



The main cluster with Jamal Zougam is clearly evident in both the graphs. The two other clusters can be also seen one with seeman gaby eid and other with mohamed chedadi,mohamed oulat akcha the third small cluster is not so visible. In conclusion the prominent cluster of Jamal Zougma can be definitely seen in both the clusters.

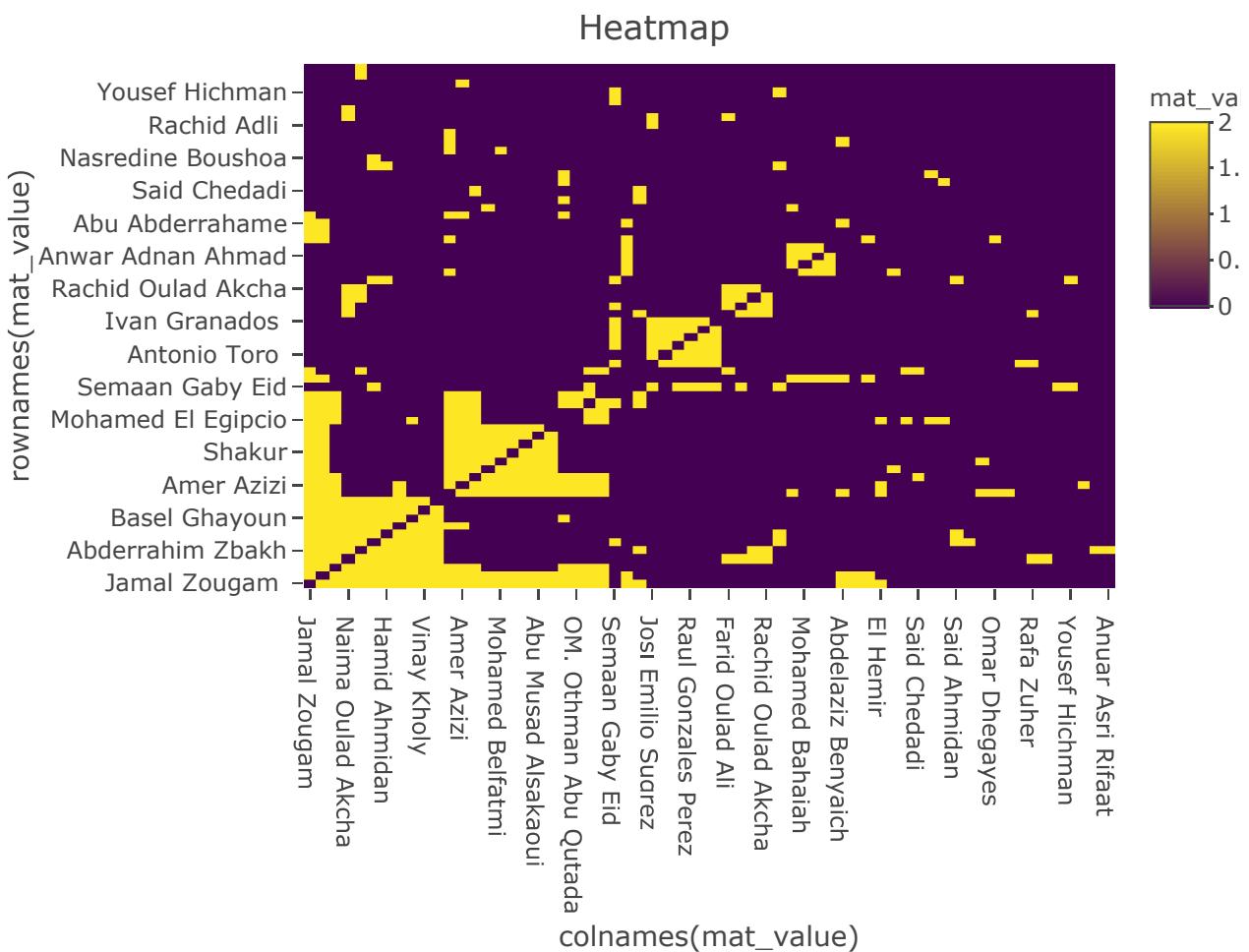
4.

```
adjacency <- get.adjacency(graph_data_frame, sparse=F)
colnames(adjacency) <- nodes$label
rownames(adjacency) <- nodes$label
rowdist<-dist(adjacency)
```

```
reord<-get_order(seriate(rowdist, "HC"))
mat_value<-adjacency[reord,reord]
```

##Since using ubuntu system using the below line of code.

```
plot_ly(z=~mat_value, x=~colnames(mat_value),  
        y=~rownames(mat_value), type="heatmap") %>% layout(title = "Heatmap")
```



The cluster that is most prominent identified here is same identified in step 1 and step 3. That is the one containing Jamal zougham, Mohamed chaoui etc.

Assignment 2

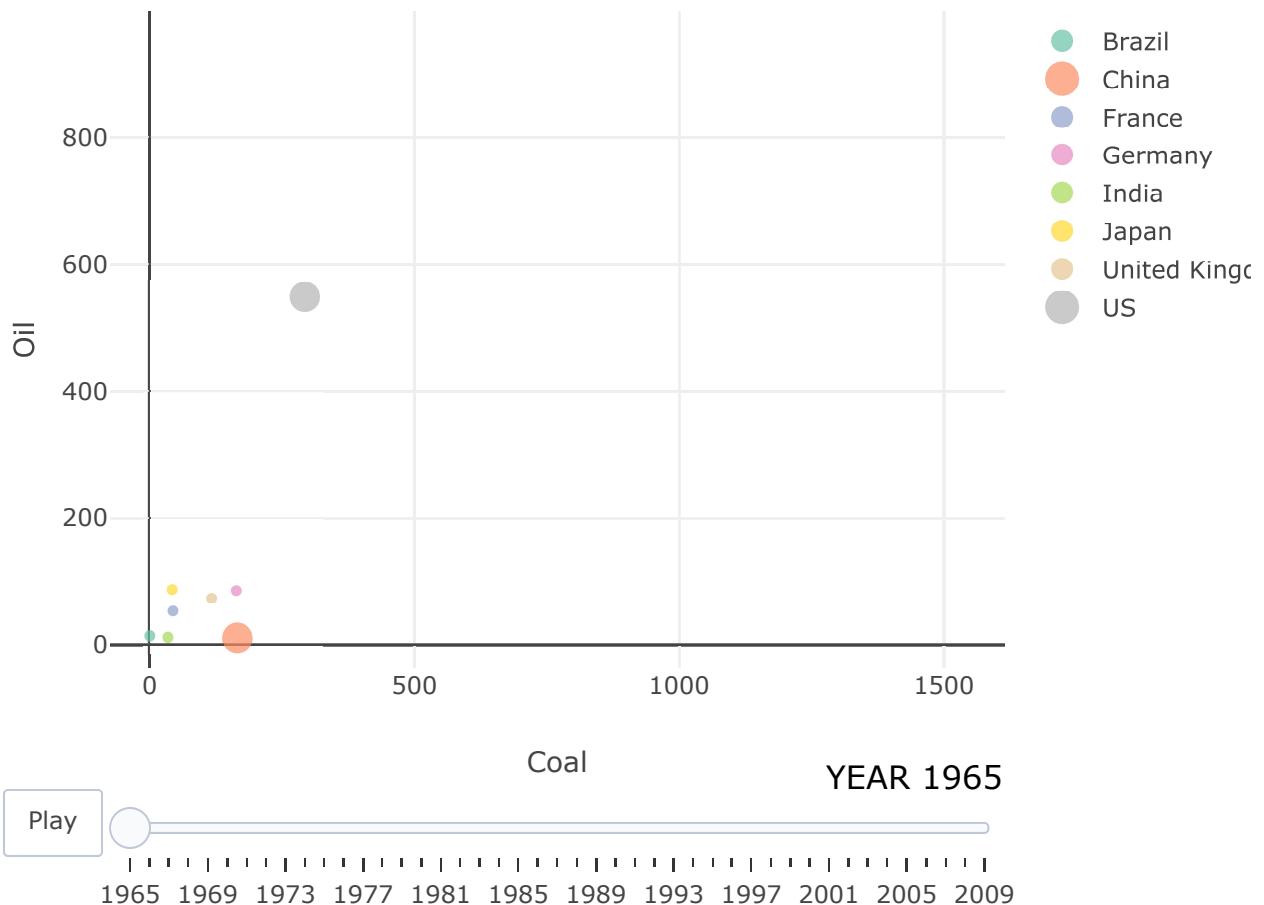
The consumption of oil and coal in many countries within a time period is given in the oilcoal.csv.

1.plotly visualisation

```
library(tidyr)
library(tourr)
oilcoal<-read.csv("Oilcoal.csv",header=T,sep=";",stringsAsFactors = F)
oilcoal<-oilcoal[,c(1:5)]
oilcoal$Coal<-as.numeric(gsub(",",".",oilcoal$Coal))
oilcoal$Oil<-as.numeric(gsub(",",".",oilcoal$Oil))
oilcoal$Marker.size<-as.numeric(gsub(",",".",oilcoal$Marker.size))
#Visualize data in Plotly as an animated bubble chart of Coal versus Oil
#in which the bubble size corresponds to the country size.
#List several noteworthy features of the investigated animation.

####1

base<-oilcoal%>%plot_ly(x=~Coal,y=~Oil,size=~Marker.size,text=~Country,hoverinfo="text")%>%
  add_markers(color=~Country,frame=~Year,ids=~Country)%>%
  animation_opts(8,redraw = F)%>%animation_slider(
    currentvalue = list(prefix = "YEAR ", font = list(color="black")))
base
```



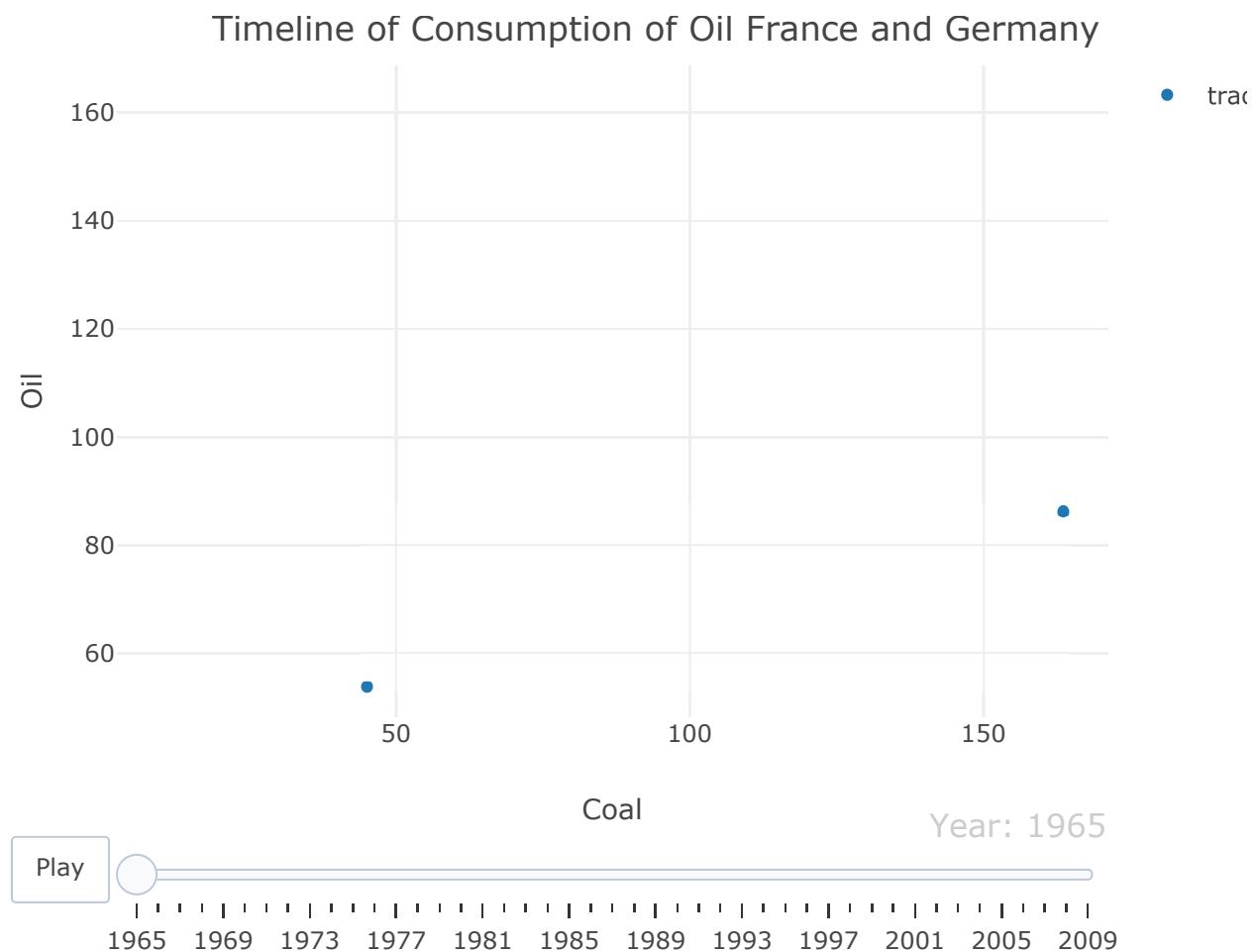
We can say like three countries grab our attention in the animation mostly.US,China and India.We could see that in the start of the animation,that is year 1965,the US had the highest oil and coal consumption ,then germany,uk,britain and china.Coal is a dwindling resource.May be the europe

countries and US have started coal mining earlier and over a period of time they would have found it is a bad source of energy because of the bad working condition of miners, price increased as the coal resource decreases and the pollution. So they might have opted other cleaner source of energy. US has a high consumption of oil maybe because the population has tripled or so by 2007 and hence domestic consumption increases.

We think the increased consumption of oil and coal and oil in India and China are related to the decreased consumption in other countries. As time progressed India, China, Japan etc became industrialised and other countries have outsourced everything to these countries. We can say that production is very less in the Europe and US now. They import a lot from India and China. As industries increase the consumption becomes high. Of course the domestic need in India and China has increased is another thing because their population is way up the ladder when compared to 1968.

2. Motion chart

```
oilcoal %>% filter(Country %in% c("France", "Germany")) %>% plot_ly(x=~Coal, y=~Oil, frame =~Year, type = 'scatter', text = ~Country, mode = 'markers') %>% animation_opts(100, easing = "cubic", redraw = F) %>% layout(title="Timeline of Consumption of Oil France and Germany")
```



One reason could be that they understood they need to focus on renewable sources of energy and another would be the industries were shifted to the developing countries like China and India.

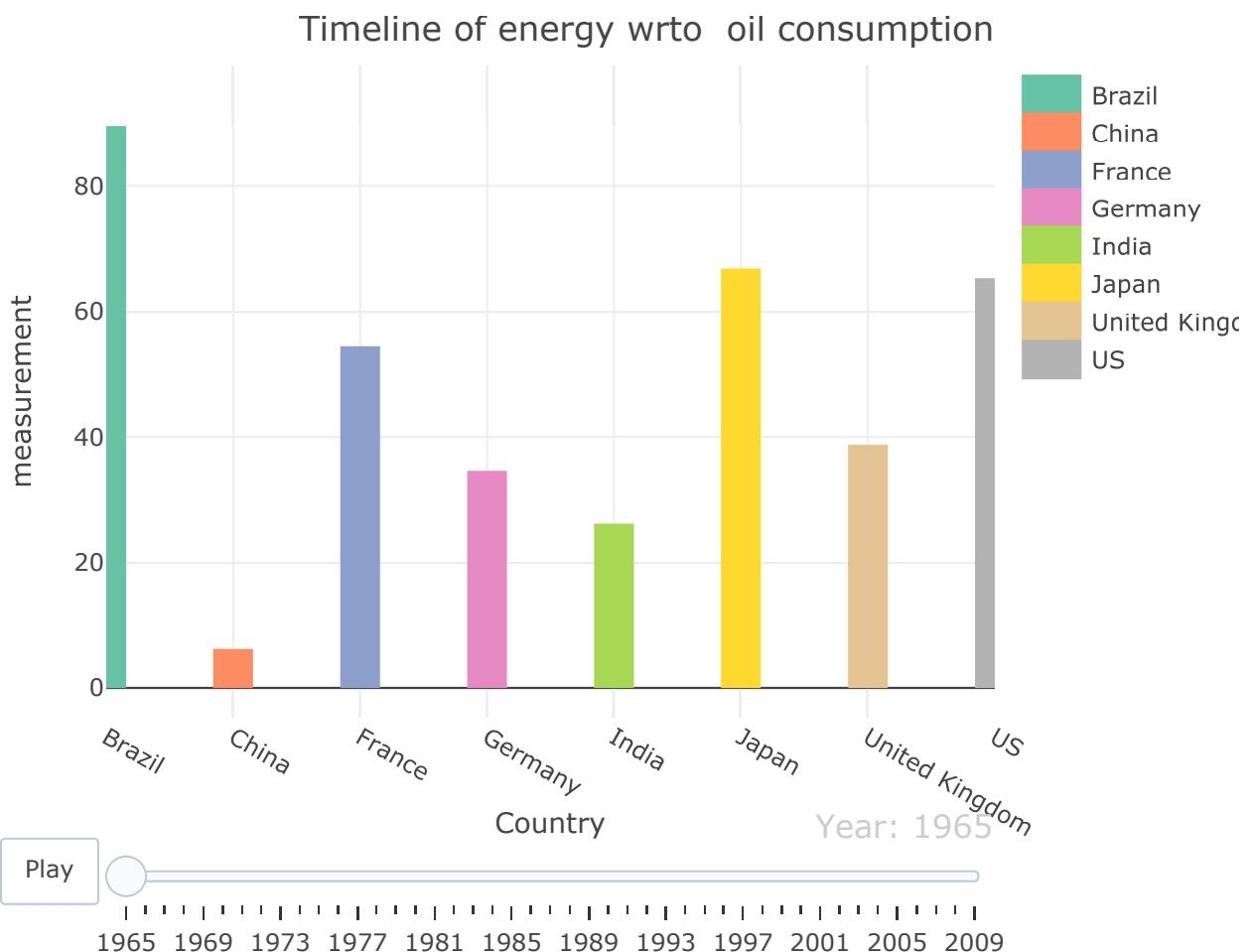
3.

```
oilcoal$oilprop<-100*oilcoal$Oil/(oilcoal$Oil+oilcoal$Coal)

oilcoal$oilprop_0<-0

melt_oil<-gather(oilcoal, condition, measurement,oilprop,oilprop_0)

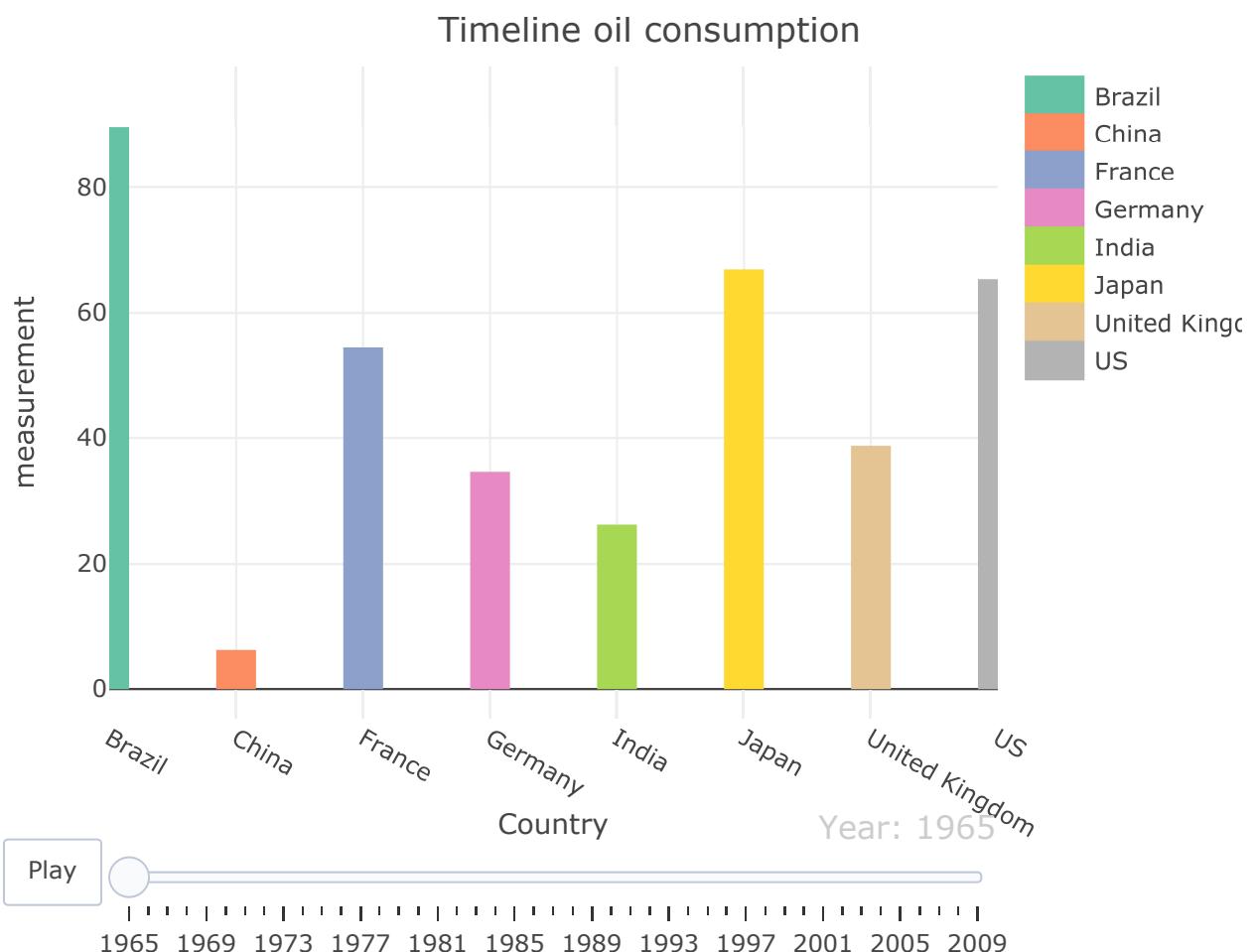
melt_oil %>% plot_ly(x=~Country, y=~measurement, frame =~Year, type = 'scatter', mode='lines', text = ~Country, color = ~Country, line = list(width = 20)) %>%
  animation_opts(300, easing = "cubic", redraw = F) %>% layout(title="Timeline of energy wrto oil consumption")
```



When we use the line plot it is easier to compare because the length will help us to visualize easier. So the variations can also be visualized better using this plot. But it does not take into account the oilconsumption and the coalconsumption separately. Or in bubble chart the x and y axis had oil and coal consumption respectively. But in the line plot together of the oil and coal is given as measurement in the y axis.

4.

```
melt_oil %>% plot_ly(x=~Country, y=~measurement, frame =~Year, type = 'scatter', mode='lines', text = ~Country, color = ~Country, line = list(width = 20)) %>% animation_opts(300, easing = "elastic", redraw = F) %>% layout(title="Timeline oil consumption")
```



The transitions are visible more clearly here we think.

5.

```

oilcoal_c <- oilcoal[, c("Country", "Year", "Coal")]
oilcoal_tour <- oilcoal_c %>%spread(Country, Coal)
oilcoal_scale <- rescale(oilcoal_tour[, 2:9])

rownames(oilcoal_scale) <- oilcoal_tour$Year
colnames(oilcoal_scale) <- names(oilcoal_tour)[-1]

set.seed(12345)
tour <- new_tour(oilcoal_scale, grand_tour(), NULL)
#tour<- new_tour(mat, guided_tour(cmass), NULL)

steps <- c(0, rep(1/15, 200))

Projs<-lapply(steps, function(step_size){
  step <- tour(step_size)
  if(is.null(step)) {
    .GlobalEnv$tour<- new_tour(oilcoal_scale, guided_tour(cmass), NULL)
    step <- tour(step_size)
  }
  step
})

# projection of each observation
tour_dat <- function(i) {
  step <- Projs[[i]]
  proj <- center(oilcoal_scale %*% step$proj)
  data.frame(x = proj[,1], y = proj[,2], state = rownames(oilcoal_scale))
}

# projection of each variable's axis
proj_dat <- function(i) {
  step <- Projs[[i]]
  data.frame(
    x = step$proj[,1], y = step$proj[,2], variable = colnames(oilcoal_scale)
  )
}

stepz <- cumsum(steps)
# tidy version of tour data
tour_dats <- lapply(1:length(steps), tour_dat)
tour_datz <- Map(function(x, y) cbind(x, step = y), tour_dats, stepz)
tour_dat <- dplyr::bind_rows(tour_datz)

# tidy version of tour projection data
proj_dats <- lapply(1:length(steps), proj_dat)
proj_datz <- Map(function(x, y) cbind(x, step = y), proj_dats, stepz)
proj_dat <- dplyr::bind_rows(proj_datz)

ax <- list(
  title = "", showticklabels = FALSE,
  zeroline = FALSE, showgrid = FALSE,
  range = c(-1.1, 1.1)
)

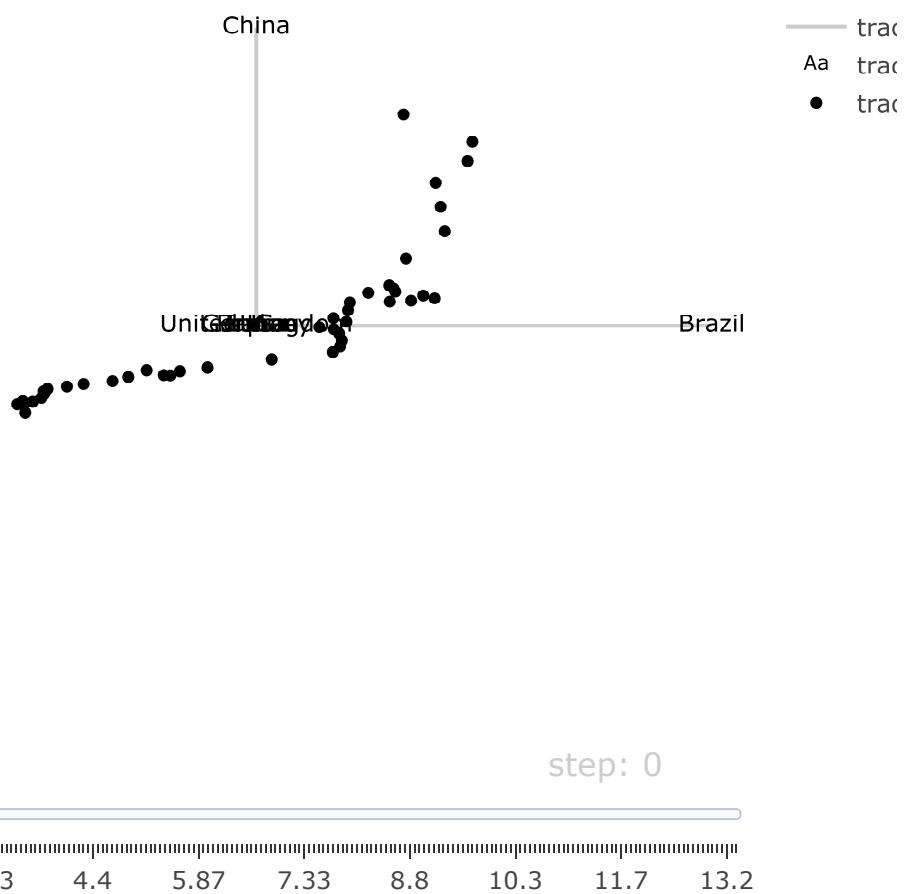
```

```

)
# for nicely formatted slider Labels
options(digits = 3)
tour_dat <- highlight_key(tour_dat, ~state, group = "A")
tour <- proj_dat %>%
  plot_ly(x = ~x, y = ~y, frame = ~step, color = I("black")) %>%
  add_segments(xend = 0, yend = 0, color = I("gray80")) %>%
  add_text(text = ~variable) %>%
  add_markers(data = tour_dat, text = ~state, ids = ~state, hoverinfo = "text") %>%
  layout(xaxis = ax, yaxis = ax, title = "Animation by country")
tour

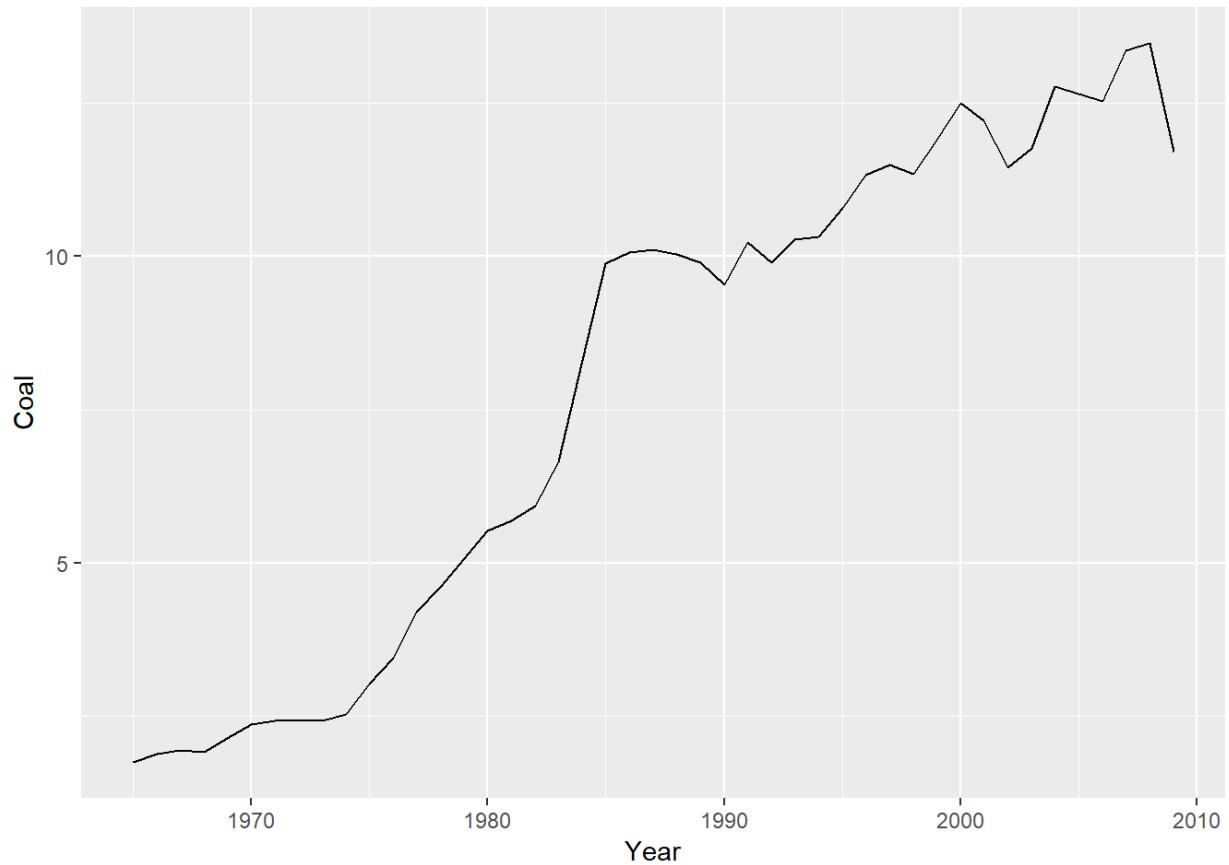
```

Animation by country



Yes the clusters correspond to different year ranges. We think brazil has the largest contribution to this projection

```
oilcoal%>%filter(Country=="Brazil")%>%ggplot(aes(y=Coal,x=Year))+geom_line()
```



Appendix

```

library(ggraph)
library(igraph)
library(visNetwork)
library(serialization)
library(plotly)

#setwd("C:/Users/quartermaine/Documents/Visualization/Lab_6")
edges <- read.delim("trainData.dat", header = FALSE, sep = " ")
nodes <- read.delim("trainMeta.dat", header = FALSE, sep = " ")
set.seed(12345)

nodes$id <- rownames(nodes)
colnames(nodes) <- c("bombers", "b_group", "id")
colnames(edges) <- c("temp", "from", "to", "value")
edges$temp <- NULL
graph <- graph.data.frame(edges, directed = T)
degree_value <- degree(graph)
nodes$value <- degree_value[match(nodes$id, names(degree_value))]
nodes <- na.omit(nodes)
nodes$label<-nodes$bombers
##### a.Basic
visNetwork(nodes=nodes,edges=edges,main = "Madrid bombing people network")
nodes$group<-nodes$b_group

visNetwork(nodes=nodes,edges=edges,main = "Madrid bombing people network")
visNetwork(nodes = nodes, edges = edges, main = "Madrid bombing people network") %>%
  visGroups(groupname = "0", color = "blue") %>%
  visGroups(groupname = "1", color = "orange") %>%
  visEdges(arrows = "from") %>%
  visOptions(collapse = TRUE,
             selectedBy = "group") %>%
  visPhysics(solver= "repulsion") %>%
  visLegend() %>% addFontAwesome()
visNetwork(nodes = nodes, edges = edges,main = "Madrid bombing people network") %>%
  visGroups(groupname = "0", color = "blue") %>%
  visGroups(groupname = "1", color = "orange") %>%
  visEdges(arrows = "to") %>%
  visOptions(highlightNearest = list(enabled =TRUE,
                                      degree = 1),
             collapse = TRUE,
             selectedBy = "group") %>%
  visPhysics(solver= "repulsion") %>%
  visLegend() %>% addFontAwesome()

visNetwork(nodes = nodes, edges = edges) %>%
  visGroups(groupname = "0", color = "blue") %>%
  visGroups(groupname = "1", color = "orange") %>%
  visEdges(arrows = "from") %>%
  visOptions(highlightNearest = list(enabled =TRUE,

```

```

degree = list(from = 1, to = 2)),
collapse = TRUE,
selectedBy = "group") %>%
visPhysics(solver= "repulsion") %>%
visLegend() %>% addFontAwesome()

graph_data_frame <- graph.data.frame(edges, directed = FALSE)
clusters <- cluster_edge_betweenness(graph_data_frame, directed = T)
nodes$clusters <- clusters$membership

visNetwork(nodes = nodes, edges = edges, main = "Madrid Bombing") %>%
visEdges(arrows = "from") %>%
visOptions(highlightNearest = list(enabled =TRUE,
degree = list(from = 1, to = 2)),
collapse = TRUE,
selectedBy = "group") %>%
visPhysics(solver= "repulsion") %>%
visLegend() %>% addFontAwesome() %>%visIgraphLayout()

adjacency <- get.adjacency(graph_data_frame, sparse=F)
colnames(adjacency) <- nodes$label
rownames(adjacency) <- nodes$label
rowdist<-dist(adjacency)

reord<-get_order(seriate(rowdist, "HC"))
mat_value<-adjacency[reord,reord]

##Since using ubuntu system using the below line of code.

plot_ly(z=~mat_value, x=~colnames(mat_value),
y=~rownames(mat_value), type="heatmap") %>% layout(title = "Heatmap")
library(tidyr)
library(tourr)
oilcoal<-read.csv("Oilcoal.csv",header=T,sep=";",stringsAsFactors = F)
oilcoal<-oilcoal[,c(1:5)]
oilcoal$Coal<-as.numeric(gsub(",",".",oilcoal$Coal))
oilcoal$Oil<-as.numeric(gsub(",",".",oilcoal$Oil))
oilcoal$Marker.size<-as.numeric(gsub(",",".",oilcoal$Marker.size))
#Visualize data in Plotly as an animated bubble chart of Coal versus Oil
#in which the bubble size corresponds to the country size.
#List several noteworthy features of the investigated animation.

###1

base<-oilcoal%>plot_ly(x=~Coal,y=~Oil,size=~Marker.size,text=~Country,hoverinfo="text")%>%
add_markers(color=~Country,frame=~Year,ids=~Country)%>%
animation_opts(8,redraw = F)%>%animation_slider(
currentvalue = list(prefix = "YEAR ", font = list(color="black")))

```

```
base
```

```
oilcoal %>% filter(Country %in% c("France", "Germany")) %>% plot_ly(x=~Coal, y=~Oil, frame =~Year, type = 'scatter', text = ~Country, mode = 'markers') %>% animation_opts(100, easing = "cubic", redraw = F) %>% layout(title="Timeline of Consumption of Oil France and Germany")

oilcoal$oilprop<-100*oilcoal$Oil/(oilcoal$Oil+oilcoal$Coal)

oilcoal$oilprop_0<-0

melt_oil<-gather(oilcoal, condition, measurement,oilprop,oilprop_0)

melt_oil %>% plot_ly(x=~Country, y=~measurement, frame =~Year, type = 'scatter', mode='lines', text = ~Country, color = ~Country, line = list(width = 20)) %>% animation_opts(300, easing = "cubic", redraw = F) %>% layout(title="Timeline of energy wrto oil consumption")

melt_oil %>% plot_ly(x=~Country, y=~measurement, frame =~Year, type = 'scatter', mode='lines', text = ~Country, color = ~Country, line = list(width = 20)) %>% animation_opts(300, easing = "elastic", redraw = F) %>% layout(title="Timeline of wrto and oil consumption")
oilcoal_c <- oilcoal[, c("Country", "Year", "Coal")]
oilcoal_tour <- oilcoal_c %>% spread(Country, Coal)
oilcoal_scale <- rescale(oilcoal_tour[, 2:9])

rownames(oilcoal_scale) <- oilcoal_tour$Year
colnames(oilcoal_scale) <- names(oilcoal_tour)[-1]

set.seed(12345)
tour <- new_tour(oilcoal_scale, grand_tour(), NULL)
#tour<- new_tour(mat, guided_tour(cmass), NULL)

steps <- c(0, rep(1/15, 200))

Projs<-lapply(steps, function(step_size){
  step <- tour(step_size)
  if(is.null(step)) {
    .GlobalEnv$tour<- new_tour(oilcoal_scale, guided_tour(cmass), NULL)
    step <- tour(step_size)
  }
  step
})

# projection of each observation
tour_dat <- function(i) {
  step <- Projs[[i]]
  proj <- center(oilcoal_scale %*% step$proj)
  data.frame(x = proj[,1], y = proj[,2], state = rownames(oilcoal_scale))
```

```

}

# projection of each variable's axis
proj_dat <- function(i) {
  step <- Projs[[i]]
  data.frame(
    x = step$proj[,1], y = step$proj[,2], variable = colnames(oilcoal_scale)
  )
}
stepz <- cumsum(steps)
# tidy version of tour data
tour_dats <- lapply(1:length(steps), tour_dat)
tour_datz <- Map(function(x, y) cbind(x, step = y), tour_dats, stepz)
tour_dat <- dplyr::bind_rows(tour_datz)

# tidy version of tour projection data
proj_dats <- lapply(1:length(steps), proj_dat)
proj_datz <- Map(function(x, y) cbind(x, step = y), proj_dats, stepz)
proj_dat <- dplyr::bind_rows(proj_datz)

ax <- list(
  title = "", showticklabels = FALSE,
  zeroline = FALSE, showgrid = FALSE,
  range = c(-1.1, 1.1)
)
# for nicely formatted slider labels
options(digits = 3)
tour_dat <- highlight_key(tour_dat, ~state, group = "A")
tour <- proj_dat %>%
  plot_ly(x = ~x, y = ~y, frame = ~step, color = I("black")) %>%
  add_segments(xend = 0, yend = 0, color = I("gray80")) %>%
  add_text(text = ~variable) %>%
  add_markers(data = tour_dat, text = ~state, ids = ~state, hoverinfo = "text") %>%
  layout(xaxis = ax, yaxis = ax, title = "Animation by country")
tour

oilcoal%>%filter(Country=="Brazil")%>%ggplot(aes(y=Coal,x=Year))+geom_line()

```

Data Transformation with dplyr :: CHEAT SHEET



dplyr functions work with pipes and expect **tidy data**. In tidy data:



Each **variable** is in its own **column**



Each **observation**, or **case**, is in its own **row**



`x %>% f(y)` becomes `f(x, y)`

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).



`summarise(.data, ...)`
Compute table of summaries.
`summarise(mtcars, avg = mean(mpg))`

`count(x, ..., wt = NULL, sort = FALSE)`
Count number of rows in each group defined by the variables in ... Also **tally()**.
`count(iris, Species)`

VARIATIONS

`summarise_all()` - Apply funs to every column.

`summarise_at()` - Apply funs to specific columns.

`summarise_if()` - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



`mtcars %>%`
`group_by(cyl) %>%`
`summarise(avg = mean(mpg))`

`group_by(.data, ..., add = FALSE)`
Returns copy of table grouped by ...
`g_iris <- group_by(iris, Species)`

`ungroup(x, ...)`
Returns ungrouped copy of table.
`ungroup(g_iris)`

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.



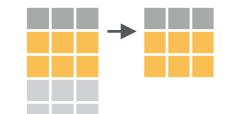
`filter(.data, ...)` Extract rows that meet logical criteria. `filter(iris, Sepal.Length > 7)`



`distinct(.data, ..., .keep_all = FALSE)` Remove rows with duplicate values.
`distinct(iris, Species)`



`sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame())` Randomly select fraction of rows.
`sample_frac(iris, 0.5, replace = TRUE)`



`slice(.data, ...)` Select rows by position.
`slice(iris, 10:15)`



`top_n(x, n, wt)` Select and order top n entries (by group if grouped data).
`top_n(iris, 5, Sepal.Width)`

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.



`pull(.data, var = -1)` Extract column values as a vector. Choose by name or index.
`pull(iris, Sepal.Length)`



`select(.data, ...)` Extract columns as a table. Also `select_if()`.
`select(iris, Sepal.Length, Species)`

Use these helpers with `select()`,
e.g. `select(iris, starts_with("Sepal"))`

`contains(match)` `num_range(prefix, range)` : e.g. `mpg:cyl`
`ends_with(match)` `one_of(...)` -, e.g. `-Species`
`matches(match)` `starts_with(match)`

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).



`mutate(.data, ...)`
Compute new column(s).
`mutate(mtcars, gpm = 1/mpg)`

`transmute(.data, ...)`
Compute new column(s), drop others.
`transmute(mtcars, gpm = 1/mpg)`

`mutate_all(.tbl, .funs, ...)` Apply funs to every column. Use with `funs()`. Also `mutate_if()`.
`mutate_all(faithful, funs(log(.), log2(.)))`
`mutate_if(iris, is.numeric, funs(log(.)))`

`mutate_at(.tbl, .cols, .funs, ...)` Apply funs to specific columns. Use with `funs()`, `vars()` and the helper functions for `select()`.
`mutate_at(iris, vars(-Species), funs(log(.)))`

`add_column(.data, ..., .before = NULL, .after = NULL)` Add new column(s). Also `add_count()`, `add_tally()`.
`add_column(mtcars, new = 1:32)`

`rename(.data, ...)` Rename columns.
`rename(iris, Length = Sepal.Length)`

Logical and boolean operators to use with filter()

< <= is.na() %in% | xor()
> >= !is.na() ! &

See `?base:::logic` and `?Comparison` for help.

ARRANGE CASES



`arrange(.data, ...)` Order rows by values of a column or columns (low to high), use with `desc()` to order from high to low.
`arrange(mtcars, mpg)`
`arrange(mtcars, desc(mpg))`

ADD CASES



`add_row(.data, ..., .before = NULL, .after = NULL)`
Add one or more rows to a table.
`add_row(faithful, eruptions = 1, waiting = 1)`



Vector Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function →

OFFSETS

dplyr::lag() - Offset elements by 1
dplyr::lead() - Offset elements by -1

CUMULATIVE AGGREGATES

dplyr::cumall() - Cumulative all()
dplyr::cumany() - Cumulative any()
 cummax() - Cumulative max()
dplyr::cummean() - Cumulative mean()
 cummin() - Cumulative min()
 cumprod() - Cumulative prod()
 cumsum() - Cumulative sum()

RANKINGS

dplyr::cume_dist() - Proportion of all values <=
dplyr::dense_rank() - rank with ties = min, no gaps
dplyr::min_rank() - rank with ties = min
dplyr::ntile() - bins into n bins
dplyr::percent_rank() - min_rank scaled to [0,1]
dplyr::row_number() - rank with ties = "first"

MATH

+, -, *, /, ^, %/%, %% - arithmetic ops
log(), **log2()**, **log10()** - logs
<, <=, >, >=, !=, == - logical comparisons
dplyr::between() - x >= left & x <= right
dplyr::near() - safe == for floating point numbers

MISC

dplyr::case_when() - multi-case if_else()
dplyr::coalesce() - first non-NA values by element across a set of vectors
dplyr::if_else() - element-wise if() + else()
dplyr::na_if() - replace specific values with NA
 pmax() - element-wise max()
 pmin() - element-wise min()
dplyr::recode() - Vectorized switch()
dplyr::recode_factor() - Vectorized switch() for factors

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function →

COUNTS

dplyr::n() - number of values/rows
dplyr::n_distinct() - # of uniques
 sum(!is.na()) - # of non-NA's

LOCATION

mean() - mean, also **mean(!is.na())**
median() - median

LOGICALS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

dplyr::first() - first value
dplyr::last() - last value
dplyr::nth() - value in nth location of vector

RANK

quantile() - nth quantile
min() - minimum value
max() - maximum value

SPREAD

IQR() - Inter-Quartile Range
mad() - median absolute deviation
sd() - standard deviation
var() - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

A	B
1	a
2	b
3	c

rownames_to_column()
Move row names into col.
a <- rownames_to_column(iris, var = "C")

A	B	C
1	a	t
2	b	u
3	c	v

column_to_rownames()
Move col in row names.
column_to_rownames(a, var = "C")

Also **has_rownames()**, **remove_rownames()**

Combine Tables

COMBINE VARIABLES

X	A B C a t 1 b u 2 c v 3	+	y	A B D a t 3 b u 2 d w 1	=	A B C A B D a t 1 a t 3 b u 2 b u 2 c v 3 d w 1
---	----------------------------------	---	---	----------------------------------	---	--

Use **bind_cols()** to paste tables beside each other as they are.

bind_cols(...) Returns tables placed side by side as a single table.
BE SURE THAT ROWS ALIGN.

Use a "**Mutating Join**" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

A B C D a t 1 3 b u 2 2 c v 3 NA	left_join(x, y, by = NULL, copy=FALSE, suffix=c("x","y"),...) Join matching values from y to x.
---	--

A B C D a t 1 3 b u 2 2 d w NA 1	right_join(x, y, by = NULL, copy = FALSE, suffix=c("x","y"),...) Join matching values from x to y.
---	---

A B C D a t 1 3 b u 2 2	inner_join(x, y, by = NULL, copy = FALSE, suffix=c("x","y"),...) Join data. Retain only rows with matches.
-------------------------------	---

A B C D a t 1 3 b u 2 2 d w NA 1	full_join(x, y, by = NULL, copy=FALSE, suffix=c("x","y"),...) Join data. Retain all values, all rows.
---	--

Use **by = c("col1", "col2", ...)** to specify one or more common columns to match on.
left_join(x, y, by = "A")

Use a named vector, **by = c("col1" = "col2")**, to match on columns that have different names in each table.
left_join(x, y, by = c("C" = "D"))

Use **suffix** to specify the suffix to give to unmatched columns that have the same name in both tables.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

COMBINE CASES

X	A B C a t 1 b u 2 c v 3	+	y	A B C C v 3 d w 4
---	----------------------------------	---	---	-------------------------

Use **bind_rows()** to paste tables below each other as they are.

bind_rows(..., .id = NULL)
Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured)

intersect(x, y, ...)
Rows that appear in both x and y.

setdiff(x, y, ...)
Rows that appear in x but not y.

union(x, y, ...)
Rows that appear in x or y.
(Duplicates removed). **union_all()** retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

EXTRACT ROWS

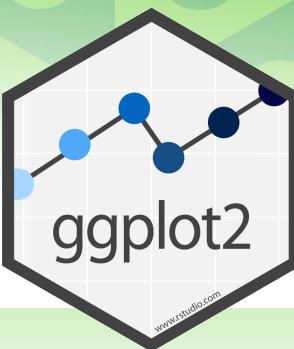
X	A B C a t 1 b u 2 c v 3	+	y	A B D a t 3 b u 2 d w 1	=
---	----------------------------------	---	---	----------------------------------	---

Use a "**Filtering Join**" to filter one table against the rows of another.

semi_join(x, y, by = NULL, ...)
Return rows of x that have a match in y.
USEFUL TO SEE WHAT WILL BE JOINED.

anti_join(x, y, by = NULL, ...)
Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.

Data Visualization with ggplot2 :: CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings **data** **geom**

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", **width** = 5, **height** = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

- a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
- a + geom_blank()**
(Useful for expanding limits)
- b + geom_curve(aes(yend = lat + 1, xend = long + 1, curvature = z))** - x, yend, alpha, angle, color, curvature, linetype, size
- a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1)** - x, y, alpha, color, group, linetype, size
- a + geom_polygon(aes(group = group))** - x, y, alpha, color, fill, group, linetype, size
- b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

- b + geom_abline(aes(intercept = 0, slope = 1))**
- b + geom_hline(aes(yintercept = lat))**
- b + geom_vline(aes(xintercept = long))**
- b + geom_segment(aes(yend = lat + 1, xend = long + 1))**
- b + geom_spoke(aes(angle = 1:1155, radius = 1))**

ONE VARIABLE continuous

- c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
- c + geom_area(stat = "bin")** - x, y, alpha, color, fill, linetype, size
- c + geom_density(kernel = "gaussian")** - x, y, alpha, color, fill, group, linetype, size, weight
- c + geom_dotplot()** - x, y, alpha, color, fill
- c + geom_freqpoly()** - x, y, alpha, color, group, linetype, size
- c + geom_histogram(binwidth = 5)** - x, y, alpha, color, fill, linetype, size, weight
- c2 + geom_qq(aes(sample = hwy))** - x, y, alpha, color, fill, linetype, size, weight

discrete

- d <- ggplot(mpg, aes(f1))
- d + geom_bar()** - x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

- e <- ggplot(mpg, aes(cty, hwy))
- e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

- e + geom_jitter(height = 2, width = 2)** - x, y, alpha, color, fill, shape, size

- e + geom_point()** - x, y, alpha, color, fill, shape, size, stroke

- e + geom_quantile()** - x, y, alpha, color, group, linetype, size, weight

- e + geom_rug(sides = "bl")** - x, y, alpha, color, linetype, size

- e + geom_smooth(method = lm)** - x, y, alpha, color, fill, group, linetype, size, weight

- e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

- f <- ggplot(mpg, aes(class, hwy))

- f + geom_col()** - x, y, alpha, color, fill, group, linetype, size

- f + geom_boxplot()** - x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

- f + geom_dotplot(binaxis = "y", stackdir = "center")** - x, y, alpha, color, fill, group

- f + geom_violin(scale = "area")** - x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

- g <- ggplot(diamonds, aes(cut, color))

- g + geom_count()** - x, y, alpha, color, fill, shape, size, stroke

THREE VARIABLES

- seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
l <- ggplot(seals, aes(long, lat))

- l + geom_contour(aes(z = z))** - x, y, z, alpha, colour, group, linetype, size, weight

continuous bivariate distribution

- h <- ggplot(diamonds, aes(carat, price))
- h + geom_bin2d(binwidth = c(0.25, 500))** - x, y, alpha, color, fill, linetype, size, weight

- h + geom_density2d()** - x, y, alpha, colour, group, linetype, size

- h + geom_hex()** - x, y, alpha, colour, fill, size

continuous function

- i <- ggplot(economics, aes(date, unemploy))

- i + geom_area()** - x, y, alpha, color, fill, linetype, size

- i + geom_line()** - x, y, alpha, color, group, linetype, size

- i + geom_step(direction = "hv")** - x, y, alpha, color, group, linetype, size

visualizing error

- df <- data.frame(grp = c("A", "B"), fit = 4.5, se = 1.2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

- j + geom_crossbar(fatten = 2)** - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

- j + geom_errorbar()** - x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

- j + geom_linerange()** - x, ymin, ymax, alpha, color, group, linetype, size

- j + geom_pointrange()** - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

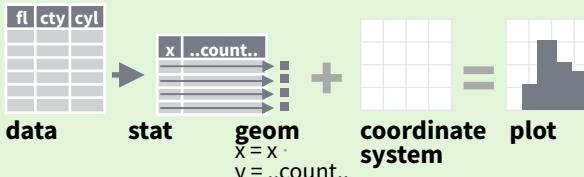
- data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

- k + geom_map(aes(map_id = state), map = map) + expand_limits(x = map\$long, y = map\$lat)** - map_id, alpha, color, fill, linetype, size

Stats

An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).



Visualize a stat by changing the default stat of a geom function, `geom_bar(stat="count")` or by using a stat function, `stat_count(geom="bar")`, which calls a default geom to make a layer (equivalent to a geom function). Use `..name..` syntax to map stat variables to aesthetics.



`c + stat_bin(binwidth = 1, origin = 10)`
`x, y | ..count.., ..ncount.., ..density.., ..ndensity..`

`c + stat_count(width = 1) x, y, | ..count.., ..prop..`

`c + stat_density(adjust = 1, kernel = "gaussian")`
`x, y, | ..count.., ..density.., ..scaled..`

`e + stat_bin_2d(bins = 30, drop = T)`
`x, y, fill | ..count.., ..density..`

`e + stat_bin_hex(bins=30) x, y, fill | ..count.., ..density..`

`e + stat_density_2d(contour = TRUE, n = 100)`
`x, y, color, size | ..level..`

`e + stat_ellipse(level = 0.95, segments = 51, type = "t")`

`l + stat_contour(aes(z = z)) x, y, z, order | ..level..`

`l + stat_summary_hex(aes(z = z), bins = 30, fun = max)`
`x, y, z, fill | ..value..`

`l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)`
`x, y, z, fill | ..value..`

`f + stat_boxplot(coef = 1.5) x, y | ..lower.., ..middle.., ..upper.., ..width.., ..ymin.., ..ymax..`

`f + stat_ydensity(kernel = "gaussian", scale = "area") x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..`

`e + stat_ecdf(n = 40) x, y | ..x.., ..y..`

`e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") x, y | ..quantile..`

`e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95) x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..`

`ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd = 0.5)) x | ..x.., ..y..`

`e + stat_identity(na.rm = TRUE)`

`ggplot() + stat_qq(aes(sample = 1:100), dist = qt, dparam = list(df = 5)) sample, x, y | ..sample.., ..theoretical..`

`e + stat_sum(x, y, size | ..n.., ..prop..)`

`e + stat_summary(fun.data = "mean_cl_boot")`

`h + stat_summary_bin(fun.y = "mean", geom = "bar")`

`e + stat_unique()`

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



GENERAL PURPOSE SCALES

Use with most aesthetics

`scale_*_continuous()` - map cont' values to visual ones

`scale_*_discrete()` - map discrete values to visual ones

`scale_*_identity()` - use data values as visual ones

`scale_*_manual(values = c())` - map discrete values to manually chosen visual ones

`scale_*_date(date_labels = "%m/%d")`, `date_breaks = "2 weeks"` - treat data values as dates.

`scale_*_datetime()` - treat data x values as date times. Use same arguments as `scale_x_date()`. See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

`scale_x_log10()` - Plot x on log10 scale

`scale_x_reverse()` - Reverse direction of x axis

`scale_x_sqrt()` - Plot x on square root scale

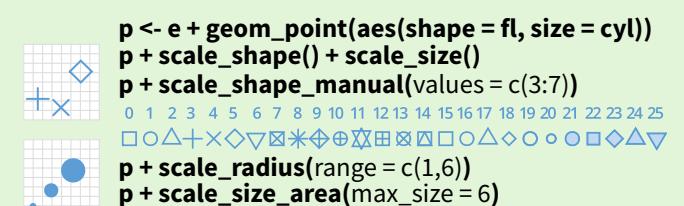
COLOR AND FILL SCALES (DISCRETE)



COLOR AND FILL SCALES (CONTINUOUS)



SHAPE AND SIZE SCALES



Coordinate Systems

`r <- d + geom_bar()`

`r + coord_cartesian(xlim = c(0, 5))`
The default cartesian coordinate system

`r + coord_fixed(ratio = 1/2)`
Cartesian coordinates with fixed aspect ratio between x and y units

`r + coord_flip()`
Flipped Cartesian coordinates

`r + coord_polar(theta = "x", direction = 1)`
theta, start, direction
Polar coordinates

`r + coord_trans(xtrans = "sqrt")`
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

`π + coord_quickmap()`

`π + coord_map(projection = "ortho", orientation = c(41, -74, 0))`
projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

`t <- ggplot(mpg, aes(cty, hwy)) + geom_point()`

`t + facet_grid(~ fl)`
facet into columns based on fl

`t + facet_grid(year ~ .)`
facet into rows based on year

`t + facet_grid(year ~ fl)`
facet into both rows and columns

`t + facet_wrap(~ fl)`
wrap facets into a rectangular layout

Set `scales` to let axis limits vary across facets

`t + facet_grid(drv ~ fl, scales = "free")`
x and y axis limits adjust to individual facets
`"free_x"` - x axis limits adjust
`"free_y"` - y axis limits adjust

Set `labeler` to adjust facet labels

`t + facet_grid(. ~ fl, labeler = label_both)`
fl: c fl: d fl: e fl: p fl: r

`t + facet_grid(fl ~ ., labeler = label_bquote(alpha ^ .(fl)))`
α^c α^d α^e α^p α^r

`t + facet_grid(. ~ fl, labeler = label_parsed)`
c d e p r

Labels

`t + labs(x = "New x axis label", y = "New y axis label", title = "Add a title above the plot", subtitle = "Add a subtitle below title", caption = "Add a caption below plot", <AES> = "New <AES> legend title")`

`t + annotate(geom = "text", x = 8, y = 9, label = "A")`

`geom to place manual values for geom's aesthetics`

Legends

`n + theme(legend.position = "bottom")`
Place legend at "bottom", "top", "left", or "right"

`n + guides(fill = "none")`
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

`n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"))`
Set legend title and labels with a scale function.

Zooming

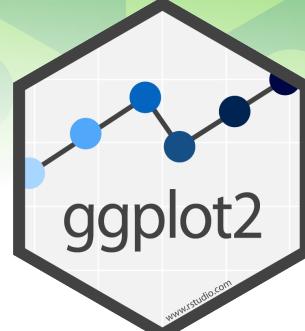
`Without clipping (preferred)`

`t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))`

`With clipping (removes unseen data points)`

`t + xlim(0, 100) + ylim(10, 20)`

`t + scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 100))`



Data Wrangling with dplyr and tidyr

Cheat Sheet



Syntax - Helpful conventions for wrangling

`dplyr::tbl_df(iris)`

Converts data to `tbl` class. `tbl`'s are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1          5.1        3.5         1.4
2          4.9        3.0         1.4
3          4.7        3.2         1.3
4          4.6        3.1         1.5
5          5.0        3.6         1.4
...
Variables not shown: Petal.Width (dbl), Species (fctr)
```

`dplyr::glimpse(iris)`

Information dense summary of `tbl` data.

`utils::View(iris)`

View data set in spreadsheet-like display (note capital V).

iris x					
<input type="button" value="Filter"/> <input type="button" value="Print"/> <input type="button" value="Copy"/> <input type="button" value="Save"/>					
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

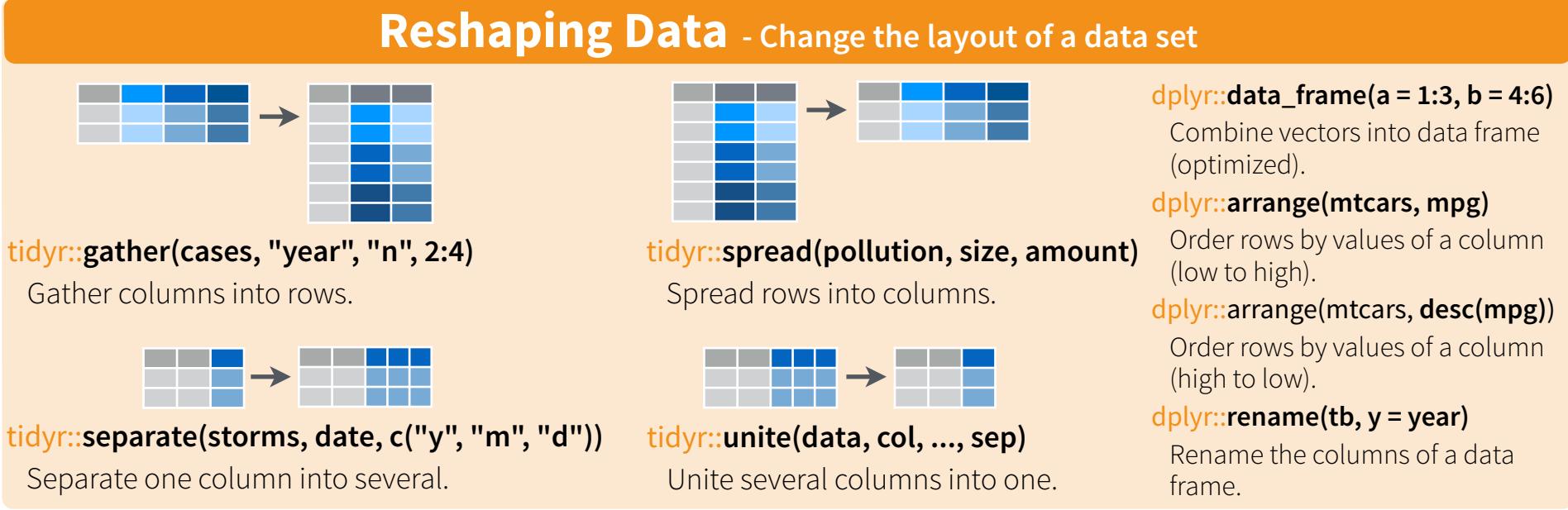
`dplyr::%>%`

Passes object on left hand side as first argument (or . argument) of function on righthand side.

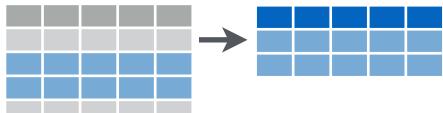
`x %>% f(y)` is the same as `f(x, y)`
`y %>% f(x, ., z)` is the same as `f(x, y, z)`

"Piping" with `%>%` makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```



Subset Observations (Rows)



`dplyr::filter(iris, Sepal.Length > 7)`

Extract rows that meet logical criteria.

`dplyr::distinct(iris)`

Remove duplicate rows.

`dplyr::sample_frac(iris, 0.5, replace = TRUE)`

Randomly select fraction of rows.

`dplyr::sample_n(iris, 10, replace = TRUE)`

Randomly select n rows.

`dplyr::slice(iris, 10:15)`

Select rows by position.

`dplyr::top_n(storms, 2, date)`

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic			
<code><</code>	Less than	<code>!=</code>	Not equal to
<code>></code>	Greater than	<code>%in%</code>	Group membership
<code>==</code>	Equal to	<code>is.na</code>	Is NA
<code><=</code>	Less than or equal to	<code>!is.na</code>	Is not NA
<code>>=</code>	Greater than or equal to	<code>&, , !, xor, any, all</code>	Boolean operators

Subset Variables (Columns)



`dplyr::select(iris, Sepal.Width, Petal.Length, Species)`

Select columns by name or helper function.

Helper functions for select - ?select

`select(iris, contains("."))`

Select columns whose name contains a character string.

`select(iris, ends_with("Length"))`

Select columns whose name ends with a character string.

`select(iris, everything())`

Select every column.

`select(iris, matches(".t.))`

Select columns whose name matches a regular expression.

`select(iris, num_range("x", 1:5))`

Select columns named x1, x2, x3, x4, x5.

`select(iris, one_of(c("Species", "Genus")))`

Select columns whose names are in a group of names.

`select(iris, starts_with("Sepal"))`

Select columns whose name starts with a character string.

`select(iris, Sepal.Length:Petal.Width)`

Select all columns between Sepal.Length and Petal.Width (inclusive).

`select(iris, -Species)`

Select all columns except Species.

Summarise Data



dplyr::summarise(iris, avg = mean(Sepal.Length))

Summarise data into single row of values.

dplyr::summarise_each(iris, funs(mean))

Apply summary function to each column.

dplyr::count(iris, Species, wt = Sepal.Length)

Count number of rows with each unique value of variable (with or without weights).



Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::first

First value of a vector.

dplyr::last

Last value of a vector.

dplyr::nth

Nth value of a vector.

dplyr::n

of values in a vector.

dplyr::n_distinct

of distinct values in a vector.

IQR

IQR of a vector.

min

Minimum value in a vector.

max

Maximum value in a vector.

mean

Mean value of a vector.

median

Median value of a vector.

var

Variance of a vector.

sd

Standard deviation of a vector.

Group Data

dplyr::group_by(iris, Species)

Group data into rows with the same value of Species.

dplyr::ungroup(iris)

Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(...)

Compute separate summary row for each group.



Make New Variables



dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)

Compute and append one or more new columns.

dplyr::mutate_each(iris, funs(min_rank))

Apply window function to each column.

dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)

Compute one or more new columns. Drop original columns.



Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::lead

Copy with values shifted by 1.

dplyr::lag

Copy with values lagged by 1.

dplyr::dense_rank

Ranks with no gaps.

dplyr::min_rank

Ranks. Ties get min rank.

dplyr::percent_rank

Ranks rescaled to [0, 1].

dplyr::row_number

Ranks. Ties got to first value.

dplyr::ntile

Bin vector into n buckets.

dplyr::between

Are values between a and b?

dplyr::cume_dist

Cumulative distribution.

dplyr::cumall

Cumulative **all**

dplyr::cumany

Cumulative **any**

dplyr::cummean

Cumulative **mean**

cumsum

Cumulative **sum**

cummax

Cumulative **max**

cummin

Cumulative **min**

cumprod

Cumulative **prod**

pmax

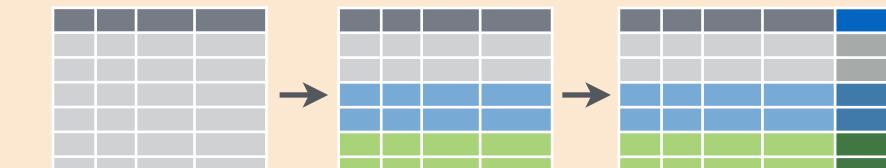
Element-wise **max**

pmin

Element-wise **min**

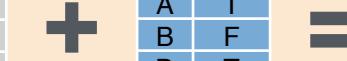
iris %>% group_by(Species) %>% mutate(...)

Compute new variables by group.



Combine Data Sets

a	b		
x1	x2	x1	x3
A	1	A	T
B	2	B	F
C	3	D	T



Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

x1	x2	x3
A	T	1
B	F	2
D	NA	T

Filtering Joins

x1	x2
A	1
B	2

x1	x2
C	3

dplyr::semi_join(a, b, by = "x1")

Join matching rows from b to a.

dplyr::right_join(a, b, by = "x1")

Join matching rows from a to b.

dplyr::inner_join(a, b, by = "x1")

Join data. Retain only rows in both sets.

dplyr::full_join(a, b, by = "x1")

Join data. Retain all values, all rows.

y	z		
x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4



dplyr::intersect(y, z)

Rows that appear in both y and z.

dplyr::union(y, z)

Rows that appear in either or both y and z.

dplyr::setdiff(y, z)

Rows that appear in y but not z.

x1	x2
A	1
B	2
C	3
B	2
C	3
D	4

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C</td			

Basic Regular Expressions in R

Cheat Sheet

Character Classes

<code>[:digit:]</code> or <code>\d</code>	Digits; [0-9]
<code>\D</code>	Non-digits; [^0-9]
<code>[:lower:]</code>	Lower-case letters; [a-z]
<code>[:upper:]</code>	Upper-case letters; [A-Z]
<code>[:alpha:]</code>	Alphabetic characters; [A-z]
<code>[:alnum:]</code>	Alphanumeric characters [A-z0-9]
<code>\w</code>	Word characters; [A-z0-9_]
<code>\W</code>	Non-word characters
<code>[:xdigit:]</code> or <code>\x</code>	Hexadec. digits; [0-9A-Fa-f]
<code>[:blank:]</code>	Space and tab
<code>[:space:]</code> or <code>\s</code>	Space, tab, vertical tab, newline, form feed, carriage return
<code>\S</code>	Not space; [^[:space:]]
<code>[:punct:]</code>	Punctuation characters; !#\$%&'()*+, -./; <=>?@[]^_`{ }~
<code>[:graph:]</code>	Graphical characters; [[:alnum:][:punct:]]
<code>[:print:]</code>	Printable characters; [[:alnum:][:punct:]\s]
<code>[:cntrl:]</code> or <code>\c</code>	Control characters; \n, \r etc.

Special Metacharacters

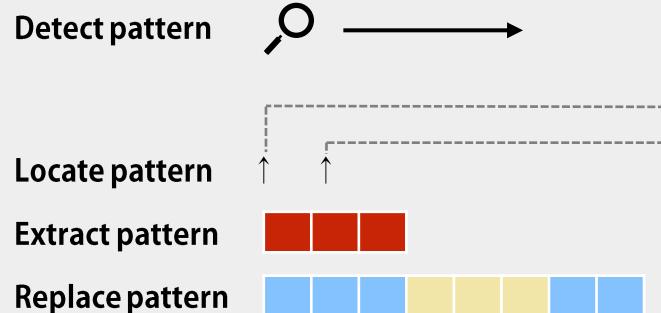
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Tab
<code>\v</code>	Vertical tab
<code>\f</code>	Form feed

Lookarounds and Conditionals*

<code>(?=)</code>	Lookahead (requires PERL = TRUE), e.g. (?=yx): position followed by 'xy'
<code>(?!)</code>	Negative lookahead (PERL = TRUE); position NOT followed by pattern
<code>(?<=)</code>	Lookbehind (PERL = TRUE), e.g. (?<=yx): position following 'xy'
<code>(?<!=)</code>	Negative lookbehind (PERL = TRUE); position NOT following pattern
<code>?(if)then</code>	If-then-condition (PERL = TRUE); use lookaheads, optional char. etc in if-clause
<code>?(if)then else</code>	If-then-else-condition (PERL = TRUE)

*see, e.g. <http://www.regular-expressions.info/lookaround.html>
<http://www.regular-expressions.info/conditional.html>

Functions for Pattern Matching



```
> string <- c("Hipopopotamus", "Rhymenoceros", "time for bottomless lyrics")
> pattern <- "t.m"
```

Detect Patterns

`grep(pattern, string)`

```
[1] 1 3
```

`grep(pattern, string, value = TRUE)`

```
[1] "Hipopopotamus"
[2] "time for bottomless lyrics"
```

`grepl(pattern, string)`

```
[1] TRUE FALSE TRUE
```

`stringr::str_detect(string, pattern)`

```
[1] TRUE FALSE TRUE
```

Split a String using a Pattern

`strsplit(string, pattern)` or `stringr::str_split(string, pattern)`

Locate Patterns

`regexpr(pattern, string)`

find starting position and length of first match

`gregexpr(pattern, string)`

find starting position and length of all matches

`stringr::str_locate(string, pattern)`

find starting and end position of first match

`stringr::str_locate_all(string, pattern)`

find starting and end position of all matches

Extract Patterns

`regmatches(string, regexpr(pattern, string))`

extract first match

```
[1] "tam" "tim"
```

`regmatches(string, gregexpr(pattern, string))`

extract all matches, outputs a list

```
[[1]] "tam" [[2]] character(0) [[3]] "tim" "tom"
```

`stringr::str_extract(string, pattern)`

extract first match

```
[1] "tam" NA "tim"
```

`stringr::str_extract_all(string, pattern)`

extract all matches, outputs a list

`stringr::str_extract_all(string, pattern, simplify = TRUE)`

extract all matches, outputs a matrix

`stringr::str_match(string, pattern)`

extract first match + individual character groups

`stringr::str_match_all(string, pattern)`

extract all matches + individual character groups

Replace Patterns

`sub(pattern, replacement, string)`

replace first match

`gsub(pattern, replacement, string)`

replace all matches

`stringr::str_replace(string, pattern, replacement)`

replace first match

`stringr::str_replace_all(string, pattern, replacement)`

replace all matches

Character Classes and Groups

.

Any character except \n

|

Or, e.g. (a|b)

[...]

List permitted characters, e.g. [abc]

[a-z]

Specify character ranges

[^...]

List excluded characters

(...)

Grouping, enables back referencing using \N where N is an integer

Anchors

^

Start of the string

\$

End of the string

\b

Empty string at either edge of a word

\B

NOT the edge of a word

\<

Beginning of a word

\>

End of a word

Quantifiers

*

Matches at least 0 times

+

Matches at least 1 time

?

Matches at most 1 time; optional string

{n}

Matches exactly n times

{n,}

Matches at least n times

{n,m}

Matches between n and m times

General Modes

By default R uses *extended regular expressions*. You can switch to *PCRE regular expressions* using `PERL = TRUE` for base or by wrapping patterns with `perl()` for stringr.

All functions can be used with literal searches using `fixed = TRUE` for base or by wrapping patterns with `fixed()` for stringr.

All base functions can be made case insensitive by specifying `ignore.cases = TRUE`.

Escaping Characters

Metacharacters (. * + etc.) can be used as literal characters by escaping them. Characters can be escaped using \\ or by enclosing them in \Q...\\E.

Case Conversions

Regular expressions can be made case insensitive using (?i). In backreferences, the strings can be converted to lower or upper case using \\L or \\U (e.g. \\L\\1). This requires `PERL = TRUE`.

Greedy Matching

By default the asterisk * is greedy, i.e. it always matches the longest possible string. It can be used in lazy mode by adding ?, i.e. *?.

Greedy mode can be turned off using (?U). This switches the syntax, so that (?U)a* is lazy and (?U)a*? is greedy.

Note

Regular expressions can conveniently be created using e.g. the packages `rex` or `rebus`.

R Markdown :: CHEAT SHEET

What is R Markdown?

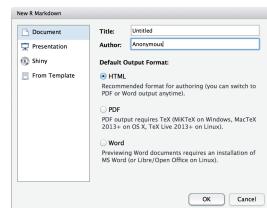


.Rmd files • An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.

Reproducible Research • At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.

Dynamic Documents • You can choose to export the finished report in a variety of formats, including html, pdf, MS Word, or RTF documents; html or pdf based slides, Notebooks, and more.

Workflow



① Open a new .Rmd file at File ► New File ► R Markdown. Use the wizard that opens to pre-populate the file with a template

② Write document by editing template

③ Knit document to create report; use knit button or render() to knit

④ Preview Output in IDE window

⑤ Publish (optional) to web server

⑥ Examine build log in R Markdown console

⑦ Use output file that is saved along side .Rmd

render

Use rmarkdown::render() to render/knit at cmd line. Important args:

input - file to render
output_format

output_options - List of render options (as in YAML)

output_file
output_dir

params - list of params to use

envir - environment to evaluate code chunks in

encoding - of input file

Embed code with knitr syntax

INLINE CODE

Insert with `r <code>`. Results appear as text without code.

Built with `r getRVersion()` → Built with 3.2.3

CODE CHUNKS

One or more lines surrounded with `{{r}}` and `{{ }}`. Place chunk options within curly braces, after r. Insert with {{getRVersion()}}

```
```{r echo=TRUE}
getRVersion()
```

```

GLOBAL OPTIONS

Set with knitr::opts_chunk\$set(), e.g.

```
```{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```

IMPORTANT CHUNK OPTIONS

cache - cache results for future knits (default = FALSE)

cache.path - directory to save cached results in (default = "cache/")

child - file(s) to knit and then include (default = NULL)

collapse - collapse all output into single block (default = FALSE)

comment - prefix for each line of results (default = "#")

dependson - chunk dependencies for caching (default = NULL)

echo - Display code in output document (default = TRUE)

engine - code language used in chunk (default = 'R')

error - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = FALSE)

eval - Run code in chunk (default = TRUE)

fig.align - 'left', 'right', or 'center' (default = 'default')

fig.cap - figure caption as character string (default = NULL)

fig.height, fig.width - Dimensions of plots in inches

highlight - highlight source code (default = TRUE)

include - Include chunk in doc after running (default = TRUE)

message - display code messages in document (default = TRUE)

results (default = 'markup')
'asis' - passthrough results

'hide' - do not display results
'hold' - put all results below all code

tidy - tidy code for display (default = FALSE)

warning - display code warnings in document (default = TRUE)

Options not listed above: R.options, aniopts, autodep, background, cache.comments, cache.lazy, cache.rebuild, cache.vars, dev, dev.args, dpi, engine.opts, engine.path, fig.asp, fig.env, fig.ext, fig.keep, fig.lp, fig.path, fig.pos, fig.process, fig.retina, fig.scap, fig.show, fig.showtext, fig.subcap, interval, out.extra, out.height, out.width, prompt, purl, ref.label, render, size, split, tidy.opts

.rmd Structure



YAML Header

Optional section of render (e.g. pandoc) options written as key:value pairs (YAML).

At start of file

Between lines of ---

Text

Narration formatted with markdown, mixed with:

Code Chunks

Chunks of embedded code. Each chunk:

Begins with `{{r}}`

ends with `{{ }}`

R Markdown will run the code and append the results to the doc. It will use the location of the .Rmd file as the working directory

Parameters

Parameterize your documents to reuse with different inputs (e.g., data, values, etc.)

1. **Add parameters** • Create and set parameters in the header as sub-values of params

```
---
params:
  n: 100
  d: ! Sys.Date()
---
```

2. **Call parameters** • Call parameter values in code as params\$<name>

Today's date is `r params\$d`

3. **Set parameters** • Set values with Knit with parameters or the params argument of render():

render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01")))

Knit to HTML
Knit to PDF
Knit to Word
Knit with Parameters...

Interactive Documents

Turn your report into an interactive Shiny document in 4 steps

1. Add runtime: shiny to the YAML header.
2. Call Shiny input functions to embed input objects.
3. Call Shiny render functions to embed reactive output.
4. Render with rmarkdown::run or click Run Document in RStudio IDE

```
---
output: html_document
runtime: shiny
---

```{r, echo = FALSE}
numericInput("n", "How many cars?", 5)
renderTable({
 head(cars, input$n)
})
```

```

| How many cars? | |
|----------------|-------|
| speed | dist |
| 1 4.00 | 2.00 |
| 2 4.00 | 10.00 |
| 3 7.00 | 4.00 |
| 4 7.00 | 22.00 |
| 5 8.00 | 16.00 |

Embed a complete app into your document with shiny::shinyAppDir()

NOTE: Your report will be rendered as a Shiny app, which means you must choose an html output format, like html_document, and serve it with an active R Session.





Pandoc's Markdown

Write with syntax on the left to create effect on right (after render)

```
Plain text
End a line with two spaces
to start a new paragraph.
*italics* and **bold**
`verbatim` code
sub/superscript22
~~strikethrough~~
escaped: `*` \\
endash: --, emdash: ---
equation: $A = \pi * r^2$
```

```
equation block:
```

```
$$E = mc^2$$
```

```
> block quote
```

```
# Header1 {#anchor}
## Header 2 {#css_id}
### Header 3 {.css_class}
```

```
#### Header 4
```

```
##### Header 5
```

```
##### Header 6
```

```
<!--Text comment-->
```

```
\textbf{Text ignored in HTML}
```

```
<em>HTML ignored in pdfs</em>
```

```
http://www.rstudio.com
```

```
link
```

```
Jump to Header 1
```

```
image:
```

```
Plain text
End a line with two spaces
to start a new paragraph.
italics and bold
`verbatim` code
sub/superscript22
~~strikethrough~~
escaped: `*` \\
endash: --, emdash: ---
equation: A = \pi * r^2
```

```
equation block:
```

```
E = mc^2
```

```
block quote
```

Header1

Header 2

Header 3

Header 4

Header 5

Header 6

```
HTML ignored in pdfs
```

```
http://www.rstudio.com
```

```
link
```

```
Jump to Header 1
```

```
image:
```



```
Caption
```

- unordered list
 - sub-item 1
 - sub-item 2
 - sub-sub-item 1
- item 2

```
Continued (indent 4 spaces)
```

1. ordered list
2. item 2
 - i. sub-item 1
 - A. sub-sub-item 1

```
1. A list whose numbering
```

```
continues after
```

```
2. an interruption
```

```
Term 1
```

```
Definition 1
```

| Right | Left | Default | Center |
|-------|------|---------|--------|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

- slide bullet 1
- slide bullet 2

```
(>- to have bullets appear on click)
```

```
horizontal rule/slide break:
```

```
***
```

```
A footnote [^1]
```

```
[^1]: Here is the footnote.
```

1. Here is the footnote.[^1](#)

Set render options with YAML

When you render, R Markdown

1. runs the R code, embeds results and text into .md file with knitr
2. then converts the .md file into the finished format with pandoc



Set a document's default output format in the YAML header:

```
---  
output: html_document  
---  
# Body
```

output value

creates

| | |
|-----------------------|----------------------------------|
| html_document | html |
| pdf_document | pdf (requires Tex) |
| word_document | Microsoft Word (.docx) |
| odt_document | OpenDocument Text |
| rtf_document | Rich Text Format |
| md_document | Markdown |
| github_document | Github compatible markdown |
| ioslides_presentation | ioslides HTML slides |
| slidy_presentation | slidy HTML slides |
| beamer_presentation | Beamer pdf slides (requires Tex) |

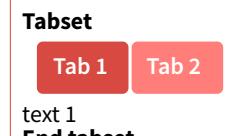
Customize output with sub-options (listed to the right):

```
---  
output: html_document:  
  code_folding: hide  
  toc_float: TRUE  
---  
# Body
```

html tabs

Use tablet css class to place sub-headers into tabs

```
# Tabset {.tabset .tabset-fade .tabset-pills}  
## Tab 1  
text 1  
## Tab 2  
text 2  
### End tabset
```



Create a Reusable Template

1. Create a new package with a `inst/rmarkdown/templates` directory

2. In the directory, Place a folder that contains:

`template.yaml` (see below)

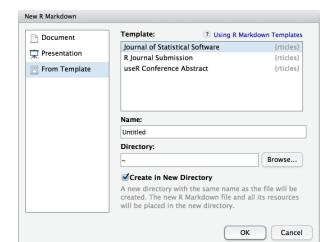
`skeleton.Rmd` (contents of the template)

any supporting files

3. Install the package

4. Access template in wizard at File ▶ New File ▶ R Markdown template.yaml

```
name: My Template
```



sub-option

description

| | | html | pdf | word | odt | rtf | md | gitlab | ioslides | slidy | beamer |
|-----------------------|---|------|-----|------|-----|-----|----|--------|----------|-------|--------|
| citation_package | The LaTeX package to process citations, natbib, biblatex or none | X | | | | | | | | | |
| code_folding | Let readers to toggle the display of R code, "none", "hide", or "show" | | X | | | | | | | | |
| colortheme | Beamer color theme to use | | | | | | | | | | X |
| css | CSS file to use to style document | | X | | | | | | X | X | |
| dev | Graphics device to use for figure output (e.g. "png") | | X | X | | | | | X | X | X |
| duration | Add a countdown timer (in minutes) to footer of slides | | | | | | | | | | X |
| fig_caption | Should figures be rendered with captions? | X | X | X | X | | | | X | X | X |
| fig_height, fig_width | Default figure height and width (in inches) for document | X | X | X | X | X | X | X | X | X | X |
| highlight | Syntax highlighting: "tango", "pygments", "kate", "zenburn", "textmate" | X | X | X | | | | | X | X | |
| includes | File of content to place in document (in_header, before_body, after_body) | X | X | X | X | X | X | X | X | X | X |
| incremental | Should bullets appear one at a time (on presenter mouse clicks)? | | | | | | | | X | X | X |
| keep_md | Save a copy of .md file that contains knitr output | | X | X | X | X | | | X | X | |
| keep_tex | Save a copy of .tex file that contains knitr output | | | | | | | | | | X |
| latex_engine | Engine to render latex, "pdflatex", "xelatex", or "lualatex" | | | | | | | | | | X |
| lib_dir | Directory of dependency files to use (Bootstrap, MathJax, etc.) | | X | | | | | | X | X | |
| mathjax | Set to local or a URL to use a local/URL version of MathJax to render equations | | X | | | | | | X | X | |
| md_extensions | Markdown extensions to add to default definition or R Markdown | X | X | X | X | X | X | X | X | X | X |
| number_sections | Add section numbering to headers | | X | X | | | | | | | |
| pandoc_args | Additional arguments to pass to Pandoc | X | X | X | X | X | X | X | X | X | X |
| preserve_yaml | Preserve YAML front matter in final document? | | | | | | | | | | X |
| reference_docx | docx file whose styles should be copied when producing docx output | | | | | | | | | | X |
| self_contained | Embed dependencies into the doc | | | | | | | | | | X |
| slide_level | The lowest heading level that defines individual slides | | | | | | | | | | X |
| smaller | Use the smaller font size in the presentation? | | | | | | | | | | X |
| smart | Convert straight quotes to curly, dashes to em-dashes, ... to ellipses, etc. | X | | | | | | | | X | X |
| template | Pandoc template to use when rendering file quarterly_report.html | X | X | X | | | | | | X | X |
| theme | Bootswatch or Beamer theme to use for page | X | | | | | | | | | X |
| toc | Add a table of contents at start of document | X | X | X | X | X | X | X | X | X | X |
| toc_depth | The lowest level of headings to add to table of contents | X | X | X | X | X | X | X | X | X | X |
| toc_float | Float the table of contents to the left of the main content | X | | | | | | | | | |

Table Suggestions

Several functions format R data into tables

| Table with kable | |
|------------------|---------|
| eruptions | waiting |
| 3.600 | 79 |
| 1.800 | 54 |
| 3.333 | 74 |
| 2.283 | 62 |

| eruptions waiting | |
|-------------------|------|
| 1 | 3.60 |
| 2 | 1.80 |
| 3 | 3.33 |
| 4 | 2.28 |



R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. **Markdown Syntax**
2. Knitr chunk options
3. Pandoc options

Syntax

Plain text

End a line with two spaces to start a new paragraph.

italics and _italics_

bold and __bold__

superscript²

~~strikethrough~~

[link](www.rstudio.com)

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

endash: --

emdash: ---

ellipsis: ...

inline equation: \$A = \pi * r^2\$

image:

horizontal rule (or slide break):

> block quote

* unordered list
* item 2

+ sub-item 1
+ sub-item 2

1. ordered list

2. item 2

+ sub-item 1
+ sub-item 2

Table Header | Second Header

----- | -----

Table Cell | Cell 2

Cell 3 | Cell 4

Becomes

Plain text

End a line with two spaces to start a new paragraph.

italics and *italics*

bold and **bold**

superscript²

~~strikethrough~~

[link](#)

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

endash: –

emdash: —

ellipsis: ...

inline equation: $A = \pi * r^2$



horizontal rule (or slide break):

block quote

- unordered list
- item 2
 - sub-item 1
 - sub-item 2

1. ordered list

2. item 2
- sub-item 1
 - sub-item 2

Table Header

Second Header

Table Cell

Cell 2

Cell 3

Cell 4



R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. Markdown Syntax
- 2. Knitr chunk options**
3. Pandoc options

Syntax

Make a code chunk with three back ticks followed by an r in braces. End the chunk with three back ticks:

```
```{r}
paste("Hello", "World!")
```

```

Place code inline with a single back ticks. The first back tick must be followed by an R, like this `r paste("Hello", "World!")`.

Add chunk options within braces. For example, `echo=FALSE` will prevent source code from being displayed:

```
```{r eval=TRUE, echo=FALSE}
paste("Hello", "World!")
```

```

Becomes

Make a code chunk with three back ticks followed by an r in braces. End the chunk with three back ticks:

```
paste("Hello", "World!")
```

```
## [1] "Hello World!"
```

Place code inline with a single back ticks. The first back tick must be followed by an R, like this Hello World!.

Add chunk options within braces. For example, `echo=FALSE` will prevent source code from being displayed:

```
## [1] "Hello World!"
```

Learn more about chunk options at <http://yihui.name/knitr/options>

Chunk options

| option | default value | description |
|--------------------------|---------------|---|
| Code evaluation | | |
| <code>child</code> | NULL | A character vector of filenames. Knitr will knit the files and place them into the main document. |
| <code>code</code> | NULL | Set to R code. Knitr will replace the code in the chunk with the code in the code option. |
| <code>engine</code> | 'R' | Knitr will evaluate the chunk in the named language, e.g. <code>engine = 'python'</code> . Run <code>names(knitr::knit_engines\$get())</code> to see supported languages. |
| <code>eval</code> | TRUE | If FALSE, knitr will not run the code in the code chunk. |
| <code>include</code> | TRUE | If FALSE, knitr will run the chunk but not include the chunk in the final document. |
| <code>purl</code> | TRUE | If FALSE, knitr will not include the chunk when running <code>purl()</code> to extract the source code. |
| Results | | |
| <code>collapse</code> | FALSE | If TRUE, knitr will collapse all the source and output blocks created by the chunk into a single block. |
| <code>echo</code> | TRUE | If FALSE, knitr will not display the code in the code chunk above it's results in the final document. |
| <code>results</code> | 'markup' | If 'hide', knitr will not display the code's results in the final document. If 'hold', knitr will delay displaying all output pieces until the end of the chunk. If 'asis', knitr will pass through results without reformatting them (useful if results return raw HTML, etc.) |
| <code>error</code> | TRUE | If FALSE, knitr will not display any error messages generated by the code. |
| <code>message</code> | TRUE | If FALSE, knitr will not display any messages generated by the code. |
| <code>warning</code> | TRUE | If FALSE, knitr will not display any warning messages generated by the code. |
| Code Decoration | | |
| <code>comment</code> | '##' | A character string. Knitr will append the string to the start of each line of results in the final document. |
| <code>highlight</code> | TRUE | If TRUE, knitr will highlight the source code in the final output. |
| <code>prompt</code> | FALSE | If TRUE, knitr will add > to the start of each line of code displayed in the final document. |
| <code>strip.white</code> | TRUE | If TRUE, knitr will remove white spaces that appear at the beginning or end of a code chunk. |
| <code>tidy</code> | FALSE | If TRUE, knitr will tidy code chunks for display with the <code>tidy_source()</code> function in the <code>formatR</code> package. |



R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. Markdown Syntax
2. Knitr chunk options
3. Pandoc options

Chunk options (Continued)

| option | default value | description |
|------------------------------------|-----------------|--|
| Chunks | | |
| opts.label | NULL | The label of options set in <code>knitr:: opts_template()</code> to use with the chunk. |
| R.options | NULL | Local R options to use with the chunk. Options are set with <code>options()</code> at start of chunk. Defaults are restored at end. |
| ref.label | NULL | A character vector of labels of the chunks from which the code of the current chunk is inherited. |
| Cache | | |
| autodep | FALSE | If TRUE , knitr will attempt to figure out dependencies between chunks automatically by analyzing object names. |
| cache | FALSE | If TRUE , knitr will cache the results to reuse in future knits. Knitr will reuse the results until the code chunk is altered. |
| cache.comments | NULL | If FALSE , knitr will not rerun the chunk if only a code comment has changed. |
| cache.lazy | TRUE | If TRUE , knitr will use <code>lazylload()</code> to load objects in chunk. If FALSE , knitr will use <code>load()</code> to load objects in chunk. |
| cache.path | 'cache/' | A file path to the directory to store cached results in. Path should begin in the directory that the .Rmd file is saved in. |
| cache.vars | NULL | A character vector of object names to cache if you do not wish to cache each object in the chunk. |
| dependson | NULL | A character vector of chunk labels to specify which other chunks a chunk depends on. Knitr will update a cached chunk if its dependencies change. |
| Animation | | |
| anipots | 'controls,loop' | Extra options for animations (see the <code>animate</code> package). |
| interval | 1 | The number of seconds to pause between animation frames. |
| Plots | | |
| dev | 'png' | The R function name that will be used as a graphical device to record plots, e.g. <code>dev='CairoPDF'</code> . |
| dev.args | NULL | Arguments to be passed to the device, e.g. <code>dev.args=list(bg='yellow', pointsize=10)</code> . |
| dpi | 72 | A number for knitr to use as the dots per inch (dpi) in graphics (when applicable). |
| external | TRUE | If TRUE , knitr will externalize tikz graphics to save LaTex compilation time (only for the <code>tikzDevice::tikz()</code> device). |
| fig.align | 'default' | How to align graphics in the final document. One of 'left', 'right', or 'center'. |
| fig.cap | NULL | A character string to be used as a figure caption in LaTex. |
| fig.env | 'figure' | The Latex environment for figures. |
| fig.ext | NULL | The file extension for figure output, e.g. <code>fig.ext='png'</code> . |
| fig.height, fig.width | 7 | The width and height to use in R for plots created by the chunk (in inches). |
| fig.keep | 'high' | If 'high', knitr will merge low-level changes into high level plots. If 'all', knitr will keep all plots (low-level changes may produce new plots). If 'first', knitr will keep the first plot only. If 'last', knitr will keep the last plot only. If 'none', knitr will discard all plots. |
| fig.lp | 'fig:' | A prefix to be used for figure labels in latex. |
| fig.path | 'figure/' | A file path to the directory where knitr should store the graphics files created by the chunk. |
| fig.pos | " | A character string to be used as the figure position arrangement in LaTex. |
| fig.process | NULL | A function to post-process a figure file. Should take a filename and return a filename of a new figure source. |
| fig.retina | 1 | Dpi multiplier for displaying HTML output on retina screens. |
| fig.scap | NULL | A character string to be used as a short figure caption. |
| fig.subcap | NULL | A character string to be used as captions in sub-figures in LaTex. |
| fig.show | 'asis' | If 'hide', knitr will generate the plots created in the chunk, but not include them in the final document. If 'hold', knitr will delay displaying the plots created by the chunk until the end of the chunk. If 'animate', knitr will combine all of the plots created by the chunk into an animation. |
| fig.showtext | NULL | If TRUE , knitr will call <code>showtext::showtext.begin()</code> before drawing plots. |
| out.extra | NULL | A character string of extra options for figures to be passed to LaTex or HTML. |
| out.height, out.width | NULL | The width and height to scale plots to in the final output. Can be in units recognized by output, e.g. <code>8\ linewidth, 50px</code> |
| resize.height, resize.width | NULL | The width and height to resize like graphics in LaTex, passed to <code>\resizebox{}`{`}</code> . |
| sanitize | FALSE | If TRUE , knitr will sanitize like graphics for LaTex. |



R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. Markdown Syntax
2. Knitr chunk options
- 3. Pandoc options**

| Templates | Basic YAML | Template options | Latex options | Interactive Docs |
|---|------------|------------------|---------------|------------------|
| html_document
pdf_document
word_document
md_document
ioslides_presentation
slidy_presentation
beamer_presentation | --- | --- | --- | --- |

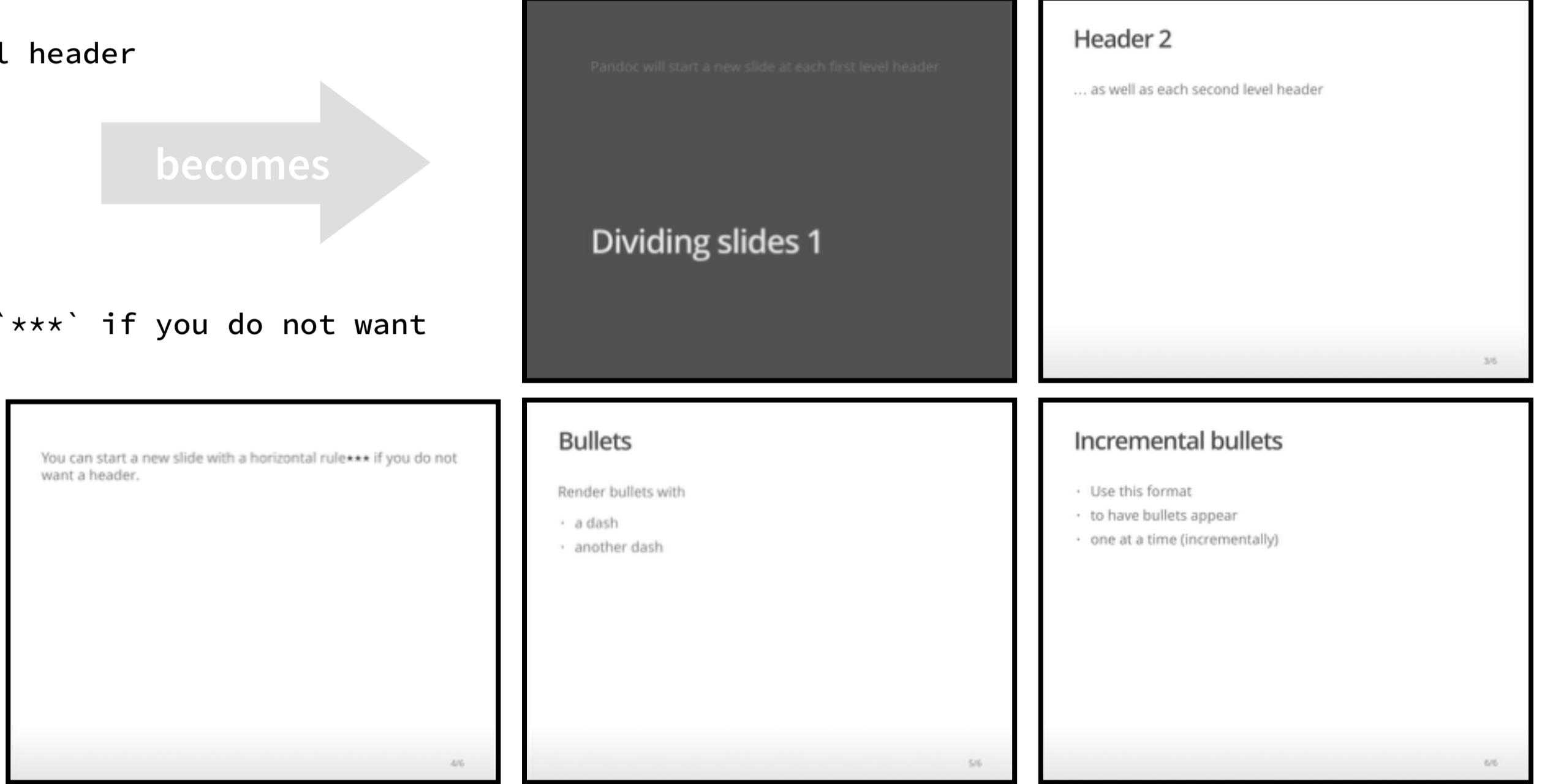
Syntax for slide formats (ioslides, slidify, beamer)

```
# Dividing slides 1
Pandoc will start a new slide at each first level header
## Header 2
... as well as each second level header
***
You can start a new slide with a horizontal rule`***` if you do not want
a header.

## Bullets
Render bullets with
- a dash
- another dash

## Incremental bullets
>- Use this format
>- to have bullets appear
>- one at a time (incrementally)
```

becomes



Slide display modes

Press a key below during presentation to enter display mode. Press **esc** to exit display mode.

ioslides

- f** - enable fullscreen mode
- w** - toggle widescreen mode
- o** - enable overview mode
- h** - enable code highlight mode
- p** - show presenter notes

slidy

- C** - show table of contents
- F** - toggle display of the footer
- A** - toggle display of current vs all slides
- S** - make fonts smaller
- B** - make fonts bigger

Top level options to customize LaTex (pdf) output

| option | description |
|---|---|
| lang | Document language code |
| fontsize | Font size (e.g. 10pt, 11pt, 12 pt) |
| documentclass | Latex document class (e.g. article) |
| classoption | Option for document class (e.g. oneside); may be repeated |
| geometry | Options for geometry class (e.g. margin=1in); may be repeated |
| mainfont, sansfont, monofont, mathfont | Document fonts (works only with xelatex and lualatex, see the <code>latex_engine</code> option) |
| linkcolor, urlcolor, citecolor | Color for internal, external, and citation links (red, green, magenta, cyan, blue, black) |



R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. Markdown Syntax
2. Knitr chunk options
3. Pandoc options

| option | html | pdf | word | md | ioslides | slidy | beamer | description |
|-----------------|------|-----|------|----|----------|-------|--------|---|
| colortheme | | | | | | | X | Beamer color theme to use (e.g., <code>colortheme: "dolphin"</code>). |
| css | X | | | | X | X | | Filepath to CSS style to use to style document (e.g., <code>css: styles.css</code>). |
| duration | | | | | | X | | Add a countdown timer (in minutes) to footer of slides (e.g., <code>duration: 45</code>). |
| fig_caption | X | X | X | | X | X | X | Should figures be rendered with captions? |
| fig_crop | | X | | | | | X | Should pdfcrop utility be automatically applied to figures (when available)? |
| fig_height | X | X | X | X | X | X | X | Default figure height (in inches) for document. |
| fig_retina | X | | | X | X | X | | Scaling to perform for retina displays (e.g., <code>fig_retina: 2</code>). |
| fig_width | X | X | X | X | X | X | X | Default figure width (in inches) for document. |
| font_adjustment | | | | | | X | | Increase or decrease font size for entire presentation (e.g., <code>font_adjustment: -1</code>). |
| fonttheme | | | | | | | X | Beamer font theme to use (e.g., <code>fonttheme: "structurebold"</code>). |
| footer | | | | | | X | | Text to add to footer of each slide (e.g., <code>footer: "Copyright (c) 2014 RStudio"</code>). |
| highlight | X | X | | | X | X | | Syntax highlighting style (e.g. "tango", "pygments", "kate", "zenburn", and |
| includes | X | X | | X | X | X | | See below |
| -in_header | X | X | | | X | X | X | File of content to place in document header (e.g., <code>in_header: header.html</code>). |
| -before_body | X | X | | | X | X | X | File of content to place before document body (e.g., <code>before_body:</code> |
| -after_body | X | X | | | X | X | X | File of content to place after document body (e.g., <code>after_body: doc_suffix.html</code>). |
| incremental | | | | | X | X | X | Should bullets appear one at a time (on presenter mouse clicks)? |
| keep_md | X | | | | X | X | | Save a copy of .md file that contains knitr output (in addition to the .Rmd and HTML files)? |
| keep_tex | | X | | | | X | | Save a copy of .tex file that contains knitr output (in addition to the .Rmd and PDF files)? |
| latex_engine | | X | | | | | | Engine to render latex. Should be one of "pdflatex", "xelatex", and "lualatex". |
| lib_dir | X | | | | X | X | | Directory of dependency files to use (Bootstrap, MathJax, etc.) (e.g., <code>lib_dir: libs</code>). |
| logo | | | | | X | | | File path to a logo (at least 128 x 128) to add to presentation (e.g., <code>logo: logo.png</code>). |
| mathjax | X | | | | X | X | | Set to <code>local</code> or a URL to use a local/URL version of MathJax to render equations |
| number_section | X | X | | | | | | Add section numbering to headers (e.g., <code>number_sections: true</code>). |
| pandoc_args | X | X | X | X | X | X | X | Arguments to pass to Pandoc (e.g., <code>pandoc_args: ["--title-prefix", "Foo"]</code>). |
| preserve_yaml | | | | X | | | | Preserve YAML front matter in final document? |
| reference_docx | | | X | | | | | A .docx file whose styles should be copied to use (e.g., <code>reference_docx:</code> |
| self_contained | X | | | | X | X | | Embed dependencies into the doc? Set to <code>false</code> to keep dependencies in external files. |
| slide_level | | | | | | X | | The lowest heading level that defines individual slides (e.g., <code>slide_level: 2</code>). |
| smaller | | | | | X | | | Use the smaller font size in the presentation? |
| smart | X | | | | X | X | | Convert straight quotes to curly, dashes to em-dashes, ... to ellipses, and so on? |
| template | X | X | | | X | X | | Pandoc template to use when rendering file (e.g., <code>template:</code> |
| theme | X | | | | | X | | Bootswatch or Beamer theme to use for page. Valid bootswatch themes include "cerulean", "journal", "flatly", "readable", "spacelab", "united", and "cosmo". |
| toc | X | X | | X | | X | | Add a table of contents at start of document? (e.g., <code>toc: true</code>). |
| toc_depth | X | X | | X | | | | The lowest level of headings to add to table of contents (e.g., <code>toc_depth: 2</code>). |
| transition | | | | | X | | | Speed of slide transitions should be "slower", "faster" or a number in seconds. |
| variant | | | X | | | | | The flavor of markdown to use; one of "markdown", "markdown_strict", "markdown_github", "markdown_mmd", and "markdown_phpextra" |
| widescreen | | | | X | | | | Display presentation in widescreen format? |

Shiny Cheat Sheet

learn more at shiny.rstudio.com

Shiny 0.10.0 Updated: 6/14



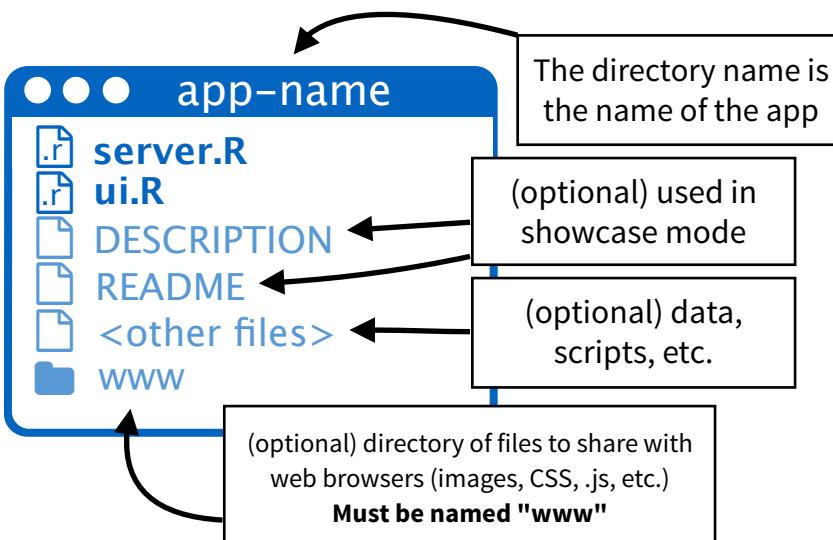
2. server.R

A set of instructions that build the R components of your app. To write server.R:

- A** Provide server.R with the minimum necessary code, `shinyServer(function(input, output) {})`
- B** Define the R components for your app between the braces that follow `function(input, output)`
- C** Save each R component in your UI as `output$<component name>`
- D** Create each output component with a render* function.
- E** Give each render* function the R code the server needs to build the component. The server will note any reactive values that appear in the code and will rebuild the component whenever these values change.
- F** Refer to widget values with `input$<widget name>`

1. Structure

Each app is a directory that contains a `server.R` file and usually a `ui.R` file (plus optional extra files)



render* functions

| function | expects | creates |
|-----------------|--------------------------|---------------------|
| renderDataTable | any table-like object | DataTables.js table |
| renderImage | list of image attributes | HTML image |
| renderPlot | plot | plot |
| renderPrint | any printed output | text |
| renderTable | any table-like object | plain table |
| renderText | character string | text |
| renderUI | Shiny tag object or | UI element (HTML) |

input values are reactive.

They must be surrounded with one of:

- render*** - creates a shiny UI component
- reactive** - creates a reactive expression
- observe** - creates a reactive observer
- isolate** - creates a non-reactive copy of a reactive object

server.R

```
# load libraries, scripts, data
A shinyServer(function(input, output) {B
  # make user specific variables
  output$text <- renderText({
    input$title
  })
  C output$plot <- renderPlot({
    D x <- mtcars[, input$x]F
    E y <- mtcars[, input$y]
    plot(x, y, pch = 16)
  })
})
```

3. Execution

Place code where it will be run the minimum necessary number of times

Run once - code placed *outside* of `shinyServer` will be run once, when you first launch your app. Use this code to set up the tools that your server will only need one copy of.

Run once per user - code placed *inside* `shinyServer` will be run once each time a user visits your app (or refreshes his or her browser). Use this code to set up the tools that your server will need a unique copy of for each user.

Run often - code placed within a `render*`, `reactive`, or `observe` function will be run many times. Place here only the code that the server needs to rebuild a UI component after a widget changes.

4. Reactivity

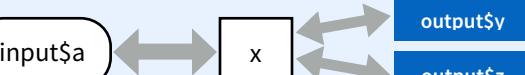
When an input changes, the server will rebuild each output that depends on it (even if the dependence is indirect). You can control this behavior by shaping the chain of dependence.

render* - An output will automatically update whenever an input in its `render*` function changes.



```
output$z <- renderText({
  input$a
})
```

Reactive expression - use reactive to create objects that will be used in multiple outputs.



```
x <- reactive({
  input$a
})
output$y <- renderText({
  x()
})
output$z <- renderText({
  x()
})
```

isolate - use isolate to use an input without depending on it. Shiny will not rebuild the output when the isolated input changes.



```
output$z <- renderText({
  paste(
    isolate(input$a),
    input$b
  )
})
```

observe - use observe to create code that runs when an input changes, but does not create an output object.



```
observe({
  input$a
  # code to run
})
```

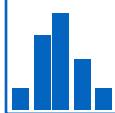
A shinyUI(fluidPage)

```

titlePanel("mtcars data"),
B sidebarLayout(
  sidebarPanel(
    CtextInput("title", "Plot title:",
      value = "x v y"),
    selectInput("x", "Choose an x var:",
      choices = names(mtcars),
      selected = "disp"),
    selectInput("y", "Choose a y var:",
      choices = names(mtcars),
      selected = "mpg")
  ),
  mainPanel(
    h3(textOutput("text")),
    plotOutput("plot")
  )
)
)

```

C In each panel or column, place...



R components - These are the output objects that you defined in `server.R`. To place a component:

1. Select the `*Output` function that builds the type of object you want to place in the UI.
2. Pass the `*Output` function a character string that corresponds to the name you assigned the object in `server.R`, e.g.

`output$plot <- renderPlot({ ... })` `plotOutput("plot")`

*Output functions

| | |
|-----------------|--------------------|
| dataTableOutput | tableOutput |
| htmlOutput | textOutput |
| imageOutput | uiOutput |
| plotOutput | verbatimTextOutput |

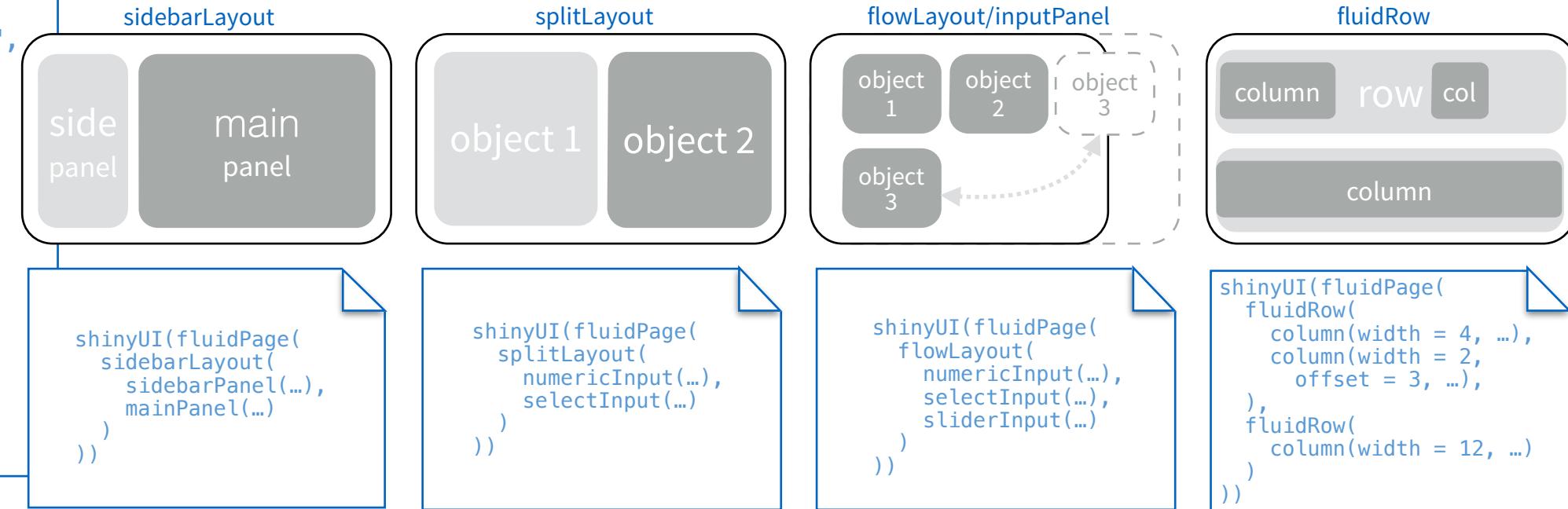
5. ui.R A description of your app's User Interface (UI), the web page that displays your app.

To write ui.R:

- A** Include the minimum necessary code for ui.R, `shinyUI(fluidPage())`

* note: use `navbarPage` instead of `fluidPage` if you'd like your app to have multiple pages connected by a navbar

- B** Build a layout for your UI. `sidebarLayout` provides a default layout when used with `sidebarPanel` and `mainPanel`. `splitLayout`, `flowLayout`, and `inputLayout` divide the page into equally spaced regions. `fluidRow` and `column` work together to create a grid-based layout, which you can use to layout a page or a panel.



6. Run your app

`runApp` - run from local files

`runGitHub` - run from files hosted on www.github.com

`runGist` - run from files saved as a gist (gist.github.com)

`runURL` - run from files saved at any URL

7. Share your app

Launch your app as a live web page that users can visit online.

ShinyApps.io

Host your apps on RStudio's server. Free and paid options
www.shinyapps.io

Shiny Server

Build your own linux server to host apps. Free and open source.
shiny.rstudio.com/deploy

Shiny Server Pro

Build a commercial server with authentication, resource management, and more.
shiny.rstudio.com/deploy

Rcookbook

Andreas

29 OCT 2018

Cookbook for R

Manipulating Data Converting data between wide and long format

Converting data between wide and long format

Problem Solution Sample data tidyR From wide to long From long to wide reshape2 From wide to long From long to wide

Problem

You want to do convert data from a wide format to a long format.

Many functions in R expect data to be in a long format rather than a wide format. Programs like SPSS, however, often use wide-formatted data. Solution

There are two sets of methods that are explained below: `gather()` and `spread()` from the `tidyR` package. This is a newer interface to the `reshape2` package. `melt()` and `dcast()` from the `reshape2` package.

There are a number of other methods which aren't covered here, since they are not as easy to use: The `reshape()` function, which is confusingly not part of the `reshape2` package; it is part of the base install of R. `stack()` and `unstack()`

Sample data

These data frames hold the same data, but in wide and long formats. They will each be converted to the other format below.

```
#Wide data
olddata_wide <- read.table(header=TRUE, text='
    subject sex control cond1 cond2
    1      M     7.9   12.3  10.7
    2      F     6.3   10.6  11.1
    3      F     9.5   13.1  13.8
    4      M    11.5   13.4  12.9
    ')
# Make sure the subject column is a factor
olddata_wide$subject <- factor(olddata_wide$subject)

#Long data
olddata_long <- read.table(header=TRUE, text='
    subject sex condition measurement
    1      M   control        7.9
    1      M   cond1       12.3
    1      M   cond2       10.7
    2      F   control        6.3
    2      F   cond1       10.6
    2      F   cond2       11.1
    3      F   control        9.5
    3      F   cond1       13.1
    3      F   cond2       13.8
    4      M   control      11.5
    4      M   cond1       13.4
    ')
```

```

        4   M      cond2      12.9
      ')
# Make sure the subject column is a factor
olddata_long$subject <- factor(olddata_long$subject)

```

tidyr

From wide to long

Use gather:

```

# olldata_wide
#>   subject sex control cond1 cond2
#> 1       1   M     7.9  12.3  10.7
#> 2       2   F     6.3  10.6  11.1
#> 3       3   F     9.5  13.1  13.8
#> 4       4   M    11.5  13.4  12.9

library(tidyr)

# The arguments to gather():
# - data: Data object
# - key: Name of new key column (made from names of data columns)
# - value: Name of new value column
# - ...: Names of source columns that contain values
# - factor_key: Treat the new key column as a factor (instead of character vector)
data_long <- gather(olldata_wide, condition, measurement, control:cond2, factor_key=TRUE)
data_long
#>   subject sex condition measurement
#> 1       1   M   control      7.9
#> 2       2   F   control      6.3
#> 3       3   F   control      9.5
#> 4       4   M   control     11.5
#> 5       1   M   cond1      12.3
#> 6       2   F   cond1      10.6
#> 7       3   F   cond1      13.1
#> 8       4   M   cond1      13.4
#> 9       1   M   cond2      10.7
#> 10      2   F   cond2      11.1
#> 11      3   F   cond2      13.8
#> 12      4   M   cond2      12.9

```

In this example, the source columns that are gathered are specified with control:cond2. This means to use all the columns, positionally, between control and cond2. Another way of doing it is to name the columns individually, as in: gather(olldata_wide, condition, measurement, control, cond1, cond2)

If you need to use gather() programmatically, you may need to use variables containing column names. To do this, you should use the gather_() function instead, which takes strings instead of bare (unquoted) column names.

```

keycol <- "condition"
valuecol <- "measurement"
gathercols <- c("control", "cond1", "cond2")

gather_(olldata_wide, keycol, valuecol, gathercols)

```

Optional: Rename the factor levels of the variable column, and sort.

```
# Rename factor names from "cond1" and "cond2" to "first" and "second"
levels(data_long$condition)[levels(data_long$condition)=="cond1"] <- "first"
levels(data_long$condition)[levels(data_long$condition)=="cond2"] <- "second"

# Sort by subject first, then by condition
data_long <- data_long[order(data_long$subject, data_long$condition), ]
data_long
#>   subject sex condition measurement
#> 1       1   M    control      7.9
#> 5       1   M     first     12.3
#> 9       1   M    second     10.7
#> 2       2   F    control      6.3
#> 6       2   F     first     10.6
#> 10      2   F    second     11.1
#> 3       3   F    control      9.5
#> 7       3   F     first     13.1
#> 11      3   F    second     13.8
#> 4       4   M    control     11.5
#> 8       4   M     first     13.4
#> 12      4   M    second     12.9
```

From long to wide

Use spread:

```
#olddata_long
#>   subject sex condition measurement
#> 1       1   M    control      7.9
#> 2       1   M    cond1     12.3
#> 3       1   M    cond2     10.7
#> 4       2   F    control      6.3
#> 5       2   F    cond1     10.6
#> 6       2   F    cond2     11.1
#> 7       3   F    control      9.5
#> 8       3   F    cond1     13.1
#> 9       3   F    cond2     13.8
#> 10      4   M    control     11.5
#> 11      4   M    cond1     13.4
#> 12      4   M    cond2     12.9

library(tidyr)

# The arguments to spread():
# - data: Data object
# - key: Name of column containing the new column names
# - value: Name of column containing values
data_wide <- spread.olddata_long, condition, measurement)
data_wide
#>   subject sex cond1 cond2 control
#> 1       1   M  12.3  10.7     7.9
```

```
#> 2      2   F  10.6  11.1    6.3
#> 3      3   F  13.1  13.8    9.5
#> 4      4   M  13.4  12.9   11.5
```

Optional: A few things to make the data look nicer.

```
# Rename cond1 to first, and cond2 to second
names(data_wide)[names(data_wide)=="cond1"] <- "first"
names(data_wide)[names(data_wide)=="cond2"] <- "second"

# Reorder the columns
data_wide <- data_wide[, c(1,2,5,3,4)]
data_wide
#>   subject sex control first second
#> 1      1   M     7.9  12.3  10.7
#> 2      2   F     6.3  10.6  11.1
#> 3      3   F     9.5  13.1  13.8
#> 4      4   M    11.5  13.4  12.9
```

The order of factor levels determines the order of the columns. The level order can be changed before reshaping, or the columns can be re-ordered afterward.

reshape2

From wide to long

Use melt:

```
#olddata_wide
#>   subject sex control cond1 cond2
#> 1      1   M     7.9  12.3  10.7
#> 2      2   F     6.3  10.6  11.1
#> 3      3   F     9.5  13.1  13.8
#> 4      4   M    11.5  13.4  12.9

library(reshape2)

# Specify id.vars: the variables to keep but not split apart on
melt.olddata_wide, id.vars=c("subject", "sex"))
#>   subject sex variable value
#> 1      1   M   control   7.9
#> 2      2   F   control   6.3
#> 3      3   F   control   9.5
#> 4      4   M   control  11.5
#> 5      1   M   cond1  12.3
#> 6      2   F   cond1  10.6
#> 7      3   F   cond1  13.1
#> 8      4   M   cond1  13.4
#> 9      1   M   cond2  10.7
#> 10     2   F   cond2  11.1
#> 11     3   F   cond2  13.8
#> 12     4   M   cond2  12.9
```

There are options for melt that can make the output a little easier to work with:

```
data_long <- melt.olddata_wide,
          # ID variables - all the variables to keep but not split apart on
```

```

id.vars=c("subject", "sex"),
# The source columns
measure.vars=c("control", "cond1", "cond2" ),
# Name of the destination column that will identify the original
# column that the measurement came from
variable.name="condition",
value.name="measurement")

data_long
#>   subject sex condition measurement
#> 1      1   M    control      7.9
#> 2      2   F    control      6.3
#> 3      3   F    control      9.5
#> 4      4   M    control     11.5
#> 5      1   M    cond1      12.3
#> 6      2   F    cond1      10.6
#> 7      3   F    cond1      13.1
#> 8      4   M    cond1      13.4
#> 9      1   M    cond2      10.7
#> 10     2   F    cond2      11.1
#> 11     3   F    cond2      13.8
#> 12     4   M    cond2      12.9

```

If you leave out the measure.vars, melt will automatically use all the other variables as the id.vars. The reverse is true if you leave out id.vars.

If you don't specify variable.name, it will name that column "variable", and if you leave out value.name, it will name that column "measurement".

Optional: Rename the factor levels of the variable column.

```

# Rename factor names from "cond1" and "cond2" to "first" and "second"
levels(data_long$condition)[levels(data_long$condition)=="cond1"] <- "first"
levels(data_long$condition)[levels(data_long$condition)=="cond2"] <- "second"

# Sort by subject first, then by condition
data_long <- data_long[ order(data_long$subject, data_long$condition), ]
data_long
#>   subject sex condition measurement
#> 1      1   M    control      7.9
#> 5      1   M    first       12.3
#> 9      1   M    second      10.7
#> 2      2   F    control      6.3
#> 6      2   F    first       10.6
#> 10     2   F    second      11.1
#> 3      3   F    control      9.5
#> 7      3   F    first       13.1
#> 11     3   F    second      13.8
#> 4      4   M    control     11.5
#> 8      4   M    first       13.4
#> 12     4   M    second      12.9

```

From long to wide

The following code uses dcast to reshape the data. This function is meant for data frames; if you are working with arrays or matrices, use acast instead.

```
#olddata_long
```

```

#>      subject sex condition measurement
#> 1      1     M   control       7.9
#> 2      1     M   cond1        12.3
#> 3      1     M   cond2        10.7
#> 4      2     F   control       6.3
#> 5      2     F   cond1        10.6
#> 6      2     F   cond2        11.1
#> 7      3     F   control       9.5
#> 8      3     F   cond1        13.1
#> 9      3     F   cond2        13.8
#> 10     4     M   control      11.5
#> 11     4     M   cond1        13.4
#> 12     4     M   cond2        12.9

# From the source:
# "subject" and "sex" are columns we want to keep the same
# "condition" is the column that contains the names of the new column to put things in
# "measurement" holds the measurements
library(reshape2)

data_wide <- dcast(olddata_long, subject + sex ~ condition, value.var="measurement")
data_wide
#>      subject sex cond1 cond2 control
#> 1      1     M  12.3  10.7    7.9
#> 2      2     F  10.6  11.1    6.3
#> 3      3     F  13.1  13.8    9.5
#> 4      4     M  13.4  12.9   11.5

```

Optional: A few things to make the data look nicer.

```

# Rename cond1 to first, and cond2 to second
names(data_wide)[names(data_wide)=="cond1"] <- "first"
names(data_wide)[names(data_wide)=="cond2"] <- "second"

# Reorder the columns
data_wide <- data_wide[, c(1,2,5,3,4)]
data_wide
#>      subject sex control first second
#> 1      1     M    7.9  12.3  10.7
#> 2      2     F    6.3  10.6  11.1
#> 3      3     F    9.5  13.1  13.8
#> 4      4     M   11.5  13.4  12.9

```

The order of factor levels determines the order of the columns. The level order can be changed before reshaping, or the columns can be re-ordered afterward.

Cookbook for R This site is powered by knitr and Jekyll. If you find any errors, please email winston@stdout.org

lab5

Andreas and Priya

11 Οκτωβρίου 2018

- Assignment 1
 - Word Cloud Plot for five
 - Word Cloud Plot for One Two
 - Pyramid Plot
 - Phrasenets
- Assignment 2
 - 1.Scatter Plot eicoseinoic vs linoleic
 - 3.Linked Scatter Plots
 - 4.Parallel Coordinate Plot linked Barplot and Scatterplot
- Appendix

Assignment 1

```
library(tm)
library(wordcloud)
library(RColorBrewer)
library(viridisLite)
library(plotrix)
library(tidyverse)
```

Word Cloud Plot for five

```
five<-read.table("Five.txt", header=F, sep='\n')#Read file
five$doc_id=1:nrow(five)
colnames(five)[1]<-"text"
mycorpus <- Corpus(DataframeSource(five)) #Creating corpus (collection of text data)

clean_corpus <- function(corporus){
  corporus <- tm_map(corporus, removePunctuation)
  corporus <- tm_map(corporus, content_transformer(tolower))
  corporus <- tm_map(corporus, stripWhitespace)
  corporus <- tm_map(corporus,function(x) removeWords(x,stopwords("english")))
  return(corporus)
}

mycorpus <- clean_corpus(mycorpus)
tdm <- TermDocumentMatrix(mycorpus) #Creating term-document matrix
m <- as.matrix(tdm)

#here we merge all rows
v <- sort(rowSums(m),decreasing=TRUE) #Sum up the frequencies of each word
d <- data.frame(word = names(v),freq=v) #Create one column=names, second=frequencies

pal<-brewer.pal(6,"PRGn") #Create palette of colors
#color_pal <- cividis(n = 8)

#wordCloud(d$word,d$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans #serif","plain"))

wordcloud(d$word,d$freq, scale=c(8,.3),min.freq=2,max.words=100,
          random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))
```



Word Cloud Plot for One Two

```

oneTwo<-read.table("oneTwo.txt",header=F,sep="\n")
oneTwo$doc_id=1:nrow(oneTwo)
colnames(oneTwo)[1]<-"text"

mycor <- Corpus(DataframeSource(oneTwo))

mycor <- clean_corpus(mycor)
tdm1<- TermDocumentMatrix(mycor) #Creating term-document matrix
m1 <- as.matrix(tdm1)

#here we merge all rows
v1 <- sort(rowSums(m1),decreasing=TRUE) #Sum up the frequencies of each word
d1<- data.frame(word = names(v1),freq=v1)
pal<-brewer.pal(8,"Dark2")

wordcloud(d1$word,d1$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=pal, vfont=c("sans serif","plain"))

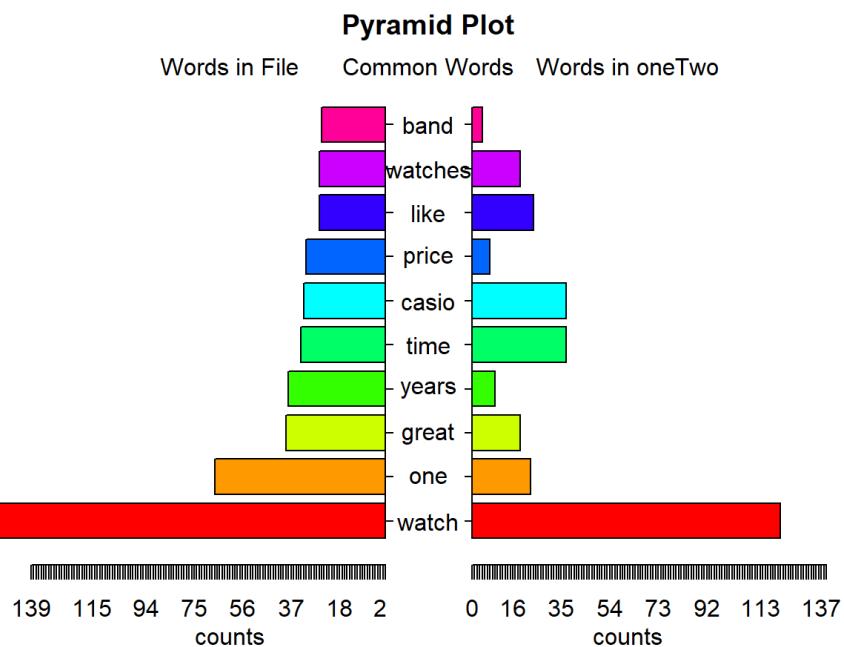
```



Pyramid Plot

```
j<-left_join(d,d1,by="word")
## Warning: Column `word` joining factors with different levels, coercing to
## character vector

pyramid.plot(
  j$freq.x[1:10],j$freq.y[1:10],
  # Words
  labels = j$word[1:10],
  top.labels = c("Words in File", "Common Words", "Words in oneTwo"),
  main = "Pyramid Plot",gap=17,unit="counts")
```



```
## [1] 5.1 4.1 4.1 2.1
```

The word cloud for five.txt which is the good feedback has more frequent words like watch,great,one,price,casio,years,battery etc.This makes sense as words watch and casio will be often coming in the words as this is the product.As it is a positive feedback,people are happy with the price.Great is a posisive word and may be telling it last for long with word years.

The oneTwo.txt is the negative feedback and the frequent words are amazon,one,back,watch,casio,battery etc.casio and watch comes frequent like before as this is the product.Amazon ,may be because they are not happy with the amazon more than product(may be the deal,return policy,shipping etc).They would like to return back the product and hence **back** has come multiple times.

Phrasenets

Phrase nets of five.text

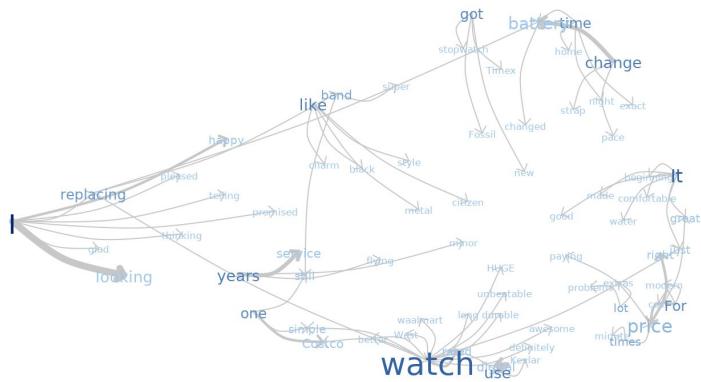


Figure 1. positive.

phrase nets of negative words

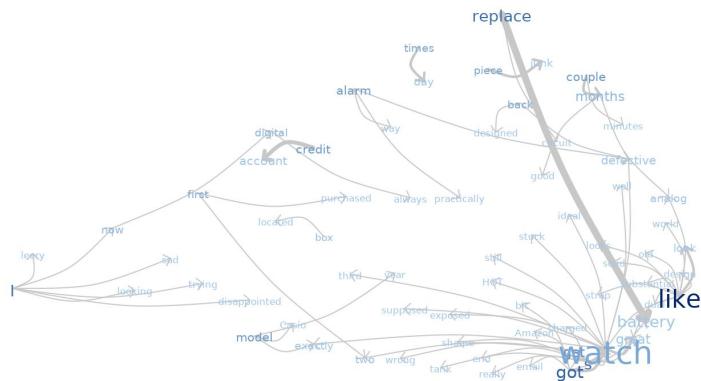


Figure 2. negative

word trees

word tree of not with the positive feedback

TOO BIG OR SMALL OR TOO HEAVY. QUITE THIN I
 , comfortable. looks snazzy. Priced right. Ok..
 and chunky. The dual time feature keeps trac
 large, easy to read and has both analog and easy to set digital r
 expensive, and not too big and chunky. The dual time feature k
 woks great but looks like a \$500 dive watch. It so far seems very
 does it look sexy but its durable super high quality and doesnt sc
 looks great (kinda like an Omega) but it is very durable. This wa
 how long I have had it, but a guess would be easily over 6 years. I
 I ever want another model. I'm surprised that so many people cri
 diver but I don't want a watch which is afraid of the water. This one is no
 a Casio) broke after a basketball hit it, popping off the glass and knocking
 beat. Looks good, feels robust but not heavy and from the other reviews should
 heavy and from the other reviews should last quite a while. WATCH IS NOT TOO
 noticeable in the pictures. IMO the dial should be prominent instead of the bezel.
 tried to see if it is waterproof but it looks like it will hold up very well This is a great
 in a good way!! This watch met all my requirements: it's a face watch, it's got a sto
 chic but the most durable watches you can buy! It fulfilled completely my expecta
 disappointed. This one has a larger dial, which is easier to see without my glasses,
 . I haven't put it to a 100-m test but it shrugs off casual water from boating, water
 great in the luminosity dept. but just viewable at night. Limited functionality, but i
 find it cheaper elsewhere. It's perfect for surfing, swimming, etc. and has a stylish
 water resistant)This is a great watch, well worth the money.I am here looking to re
 go wrong owning this piece. It will serve you well for many, many years. I have see
 really. I like to know that I'm wearing a watch when I have one on. Bottom line gr
 partial to strap bands so I have transferred the metal band from the old one so tha
 strong or long. But for \$50, it is a great watch still. My previous Casio served for te
 recommend scuba diving unless the watch is rated 200 or 300m, I have been scub
 offer as much as this Casio. For the cost, the features, and the overall value, I don't
 lost a second. The digital display is a little small, but I don't use it much anyway ar

Figure 4. wordtree not pos feedback

word tree with the word amazon in negative feedback

Both times, the watch movement stopped working correct
 The watch battery only last a year. It will cost more than t
 Amazon sent it again, to the main USPS facility, different
 If Amazon wants to credit the 49.00, I would be willing to
 com, certainly not. Thank you all. REVIEW: I hope this re
 nope, UPS would not deliver to my PO Box. Well, I went to
 I don't want this watch anymore, and I want it replaced, n
 to save the situation and keep the customer. And you win :
 'S warranty so now I'll have to chase down Casio for a repl
 money or anyone's. I paid for the watch, I got the watch,
 team even if you don't want to replace it, or give me c
 was very courteous and friendly. The Casio guys
 decides to reason and have the courtesy to credi
 correct this situation, and take back t
 (THANK YOU AMAZON for not even a e-mail response). My strong
 for not even a e-mail response). My strong recommendation: SKIP
 ! The watch is great, but there's more to this case than just a senten
 sent it again, to the main USPS facility, different place, that's where
 /UPS delivery policies, it's more like a world away. So I call Amazo
 back and talk again with another representative in India, very frier
 did have the goodwill to send a replacement, then a second replace
 expires, and now I'm on my own. Well, Casio will cover for it. Yeah,
 courteously to reconsider this issue, and give me a replacement iter
 sends me an email (one of those that look like spam and people del
 representative I just spoke to (today is September 9, 2010), has "N
 does make an effort to satisfy every customer, sometimes there are
 nor Casio want to deal with it. I would warn every Amazon custome
 customer to research the watches well. This particular watch retails
 AGAIN. I would pay the premium somewhere else. Sadly, the once
 has been eroded with so many irregularities in this case. By the way
 wants to credit the 49.00, I would be willing to purchase another, n
 this time. If Amazon decides to reason and have the courtesy to cre

Figure 4. wordtree amazon neg feedback

Assignment 2

```
library(tidyverse)
library(plotly)
library(plyr)
library(crosstalk)
library(GGally)

olive<-read.csv("olive.csv")
olive$Region<-as.factor(olive$Region)
levels(olive$Region)<-c("North","South","Sardinia island")
oo<-SharedData$new(olive)
oo<-SharedData$new(olive,~Region,group = "Choose region")
```

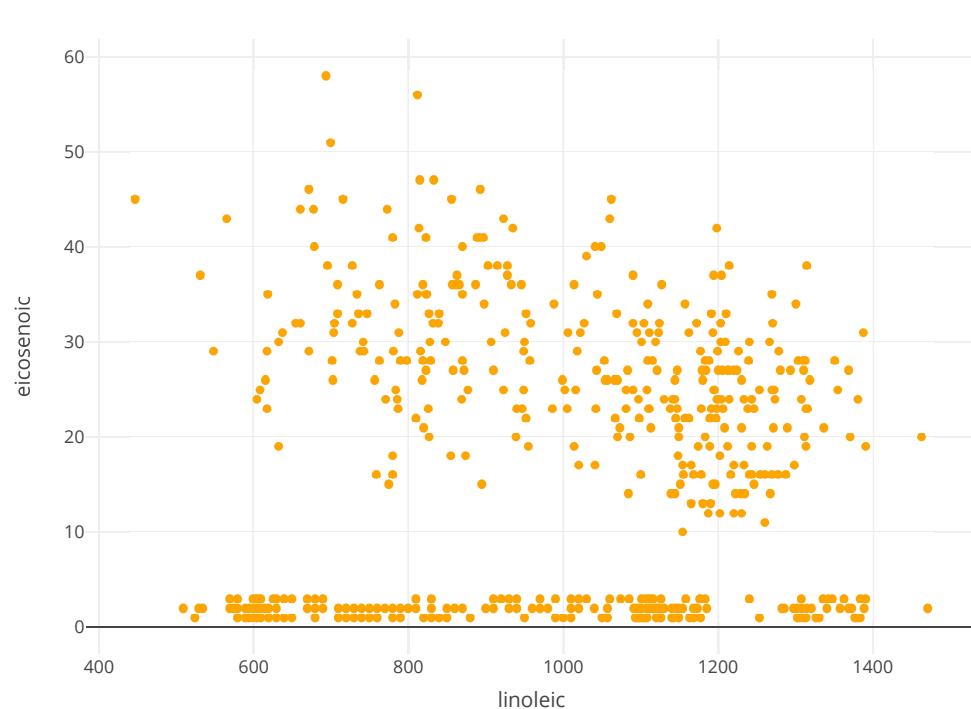
1.Scatter Plot eicoseinoic vs linoleic

```
scatterolive <- plot_ly(oo, y = ~eicosenoic, x = ~linoleic) %>%group_by(Region)%>%
  add_markers(color = I("orange"),name="hollow")%>%highlight(on="plotly_hover",persistent = F,selectize = T)

scatterolive
```

Setting the `off` event (i.e., 'plotly_doubleclick') to match the `on` event (i.e., 'plotly_hover'). You can change this default via the `highlight()` function.

Choose region



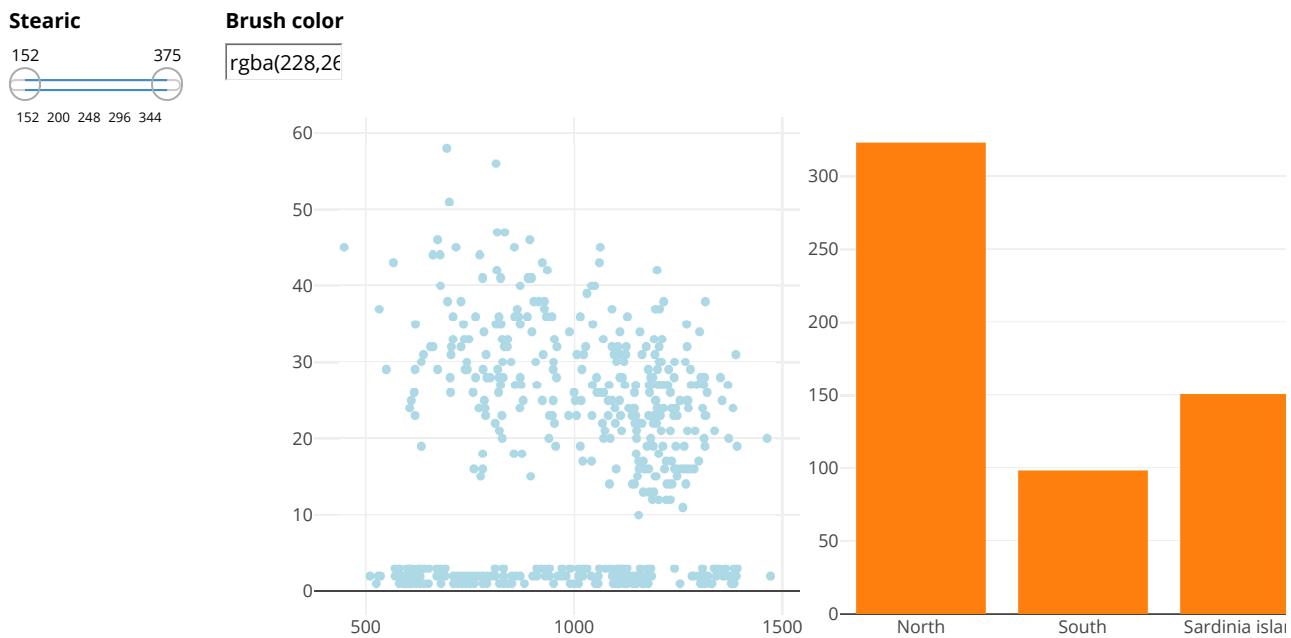
The value of eicosenoic is between one and three. ####2.Linked Scatterplot and Bar chart

```
scatterolive2 <- plot_ly(o, y = ~eicosenoic, x = ~linoleic) %>%
  add_markers(color = I("lightblue"))

barolive <-plot_ly(o, x=~Region,group="Select your region")%>%add_histogram()%>%layout(barmode="overlay")
```

Warning in plot_ly(o, x = ~Region, group = "Select your region"): The group argument has been deprecated. Use `group_by()` or split instead.
See `help('plotly_data')` for examples

```
bscols(widths=c(2, NA),filter_slider("Slider", "Stearic", o, ~stearic)
      ,subplot(scatterolive2,barolive)%>%
        highlight(on="plotly_select", dynamic=T, persistent = T, opacityDim = I(1))%>%hide_legend())
```



The brush colour and the slider/filter were the interaction operators used in this step.

Using brushing when we take out the lower values of eicosenoic(between 1 and 3),we can see they correspond to region south and sardinia Island.Now when we use slider to find the higher values of stearic we can see that they correspond to the region of north and Sardinia Island

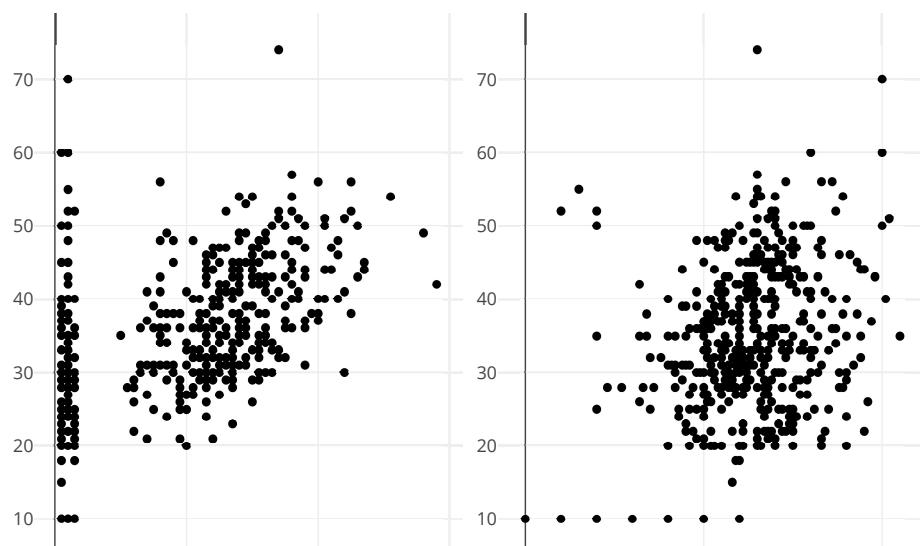
3.Linked Scatter Plots

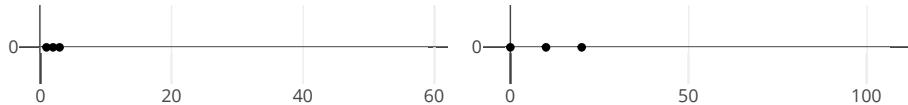
```
scatter1 <- plot_ly(o, x_= ~eicosenoic, y_= ~linolenic) %>%
  add_markers(color = I("black"))
scatter2 <- plot_ly(o, x_= ~arachidic, y_= ~linolenic) %>%
  add_markers(color = I("black"))

subplot(scatter1,scatter2)%>%
  highlight(on="plotly_select", dynamic=T, persistent=T, opacityDim = I(1))%>%hide_legend()
```

Brush color

rgba(228,26)





The values of arachidic below 40 are outliers in the plot2 are also outliers in plot 1. IN plot one their eicosenoic values are between 1 and 3.

4. Parallel Coordinate Plot linked Barplot and Scatterplot

```

p<-ggparcoord(olive,columns = c(4:11))

d<-plotly_data(ggplotly(p))%>%group_by(.ID)
d1<-SharedData$new(d, ~.ID, group="olive")
p1<-plot_ly(d1, x=~variable, y=~value)%>%
  add_lines(line=list(width=0.3))%>%
  add_markers(marker=list(size=0.3),
              text=~.ID, hoverinfo="text")

ButtonsX=list()
for (i in 4:11){
  ButtonsX[[i-3]]= list(method = "restyle",
                        args = list( "x", list(olive[[i]])),
                        label = colnames(olive)[i])
}

ButtonsY=list()
for (i in 4:11){
  ButtonsY[[i-3]]= list(method = "restyle",
                        args = list( "y", list(olive[[i]])),
                        label = colnames(olive)[i])
}

ButtonsZ=list()
for (i in 4:11){
  ButtonsZ[[i-3]]= list(method = "restyle",
                        args = list( "z", list(olive[[i]])),
                        label = colnames(olive)[i])
}

p3<-plot_ly(o,x=~palmitic,y=~stearic,z=~oleic)%>%add_markers() %>%
  layout(xaxis=list(title=""), yaxis=list(title=""),
         title = "Select variable:",
         updatemenus = list(
           list(y=0.9, buttons = ButtonsX),
           list(y=0.7, buttons = ButtonsY),
           list(y=0.5, buttons = ButtonsZ)
         )
       )

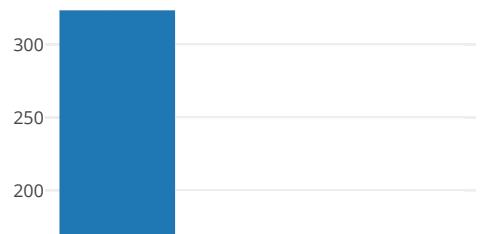
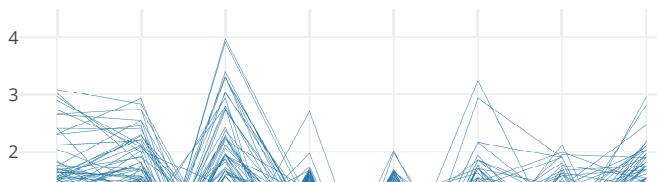
bscols(p1%>%highlight(on="plotly_select", dynamic=T, persistent = T, opacityDim = I(1))%>%
  hide_legend(),barolive,
  p3%>%highlight(on="plotly_click", dynamic=T, persistent = T)%>%hide_legend(),
  widths = c(7,5,9))

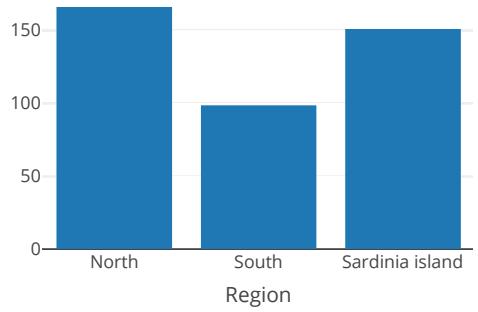
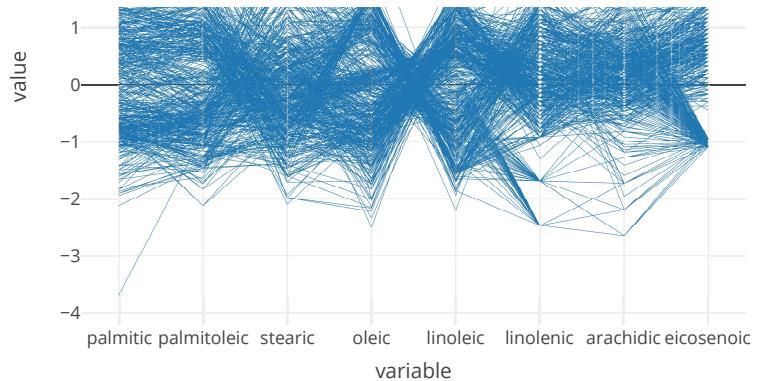
```

Warning in bscols(p1 %>% highlight(on = "plotly_select", dynamic = T, persistent = T, : Sum of bscolumn width units is greater than 12

Brush color

rgba(228,26





Brush color

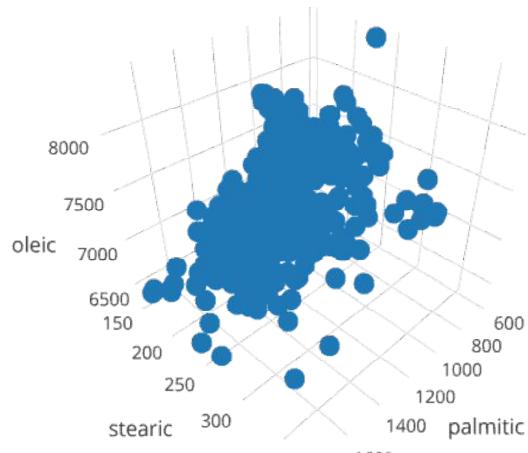
`rgba(228,26`

Select variable:

palmitic ▼

palmitic ▼

palmitic ▼



The parallel coordinate plot demonstrate clusters among observations that belong to the same region. i think oleic,lieoneic,Eicosenoic are three influential variables.while using this three variable we think we can differentiate regions.

Appendix

```

library(tm)
library(wordcloud)
library(RColorBrewer)
library(viridisLite)
library(plotrix)
library(tidyverse)
five<-read.table("Five.txt",header=F, sep='\n')#Read file
five$doc_id=1:nrow(five)
colnames(five)[1]<-"text"
mycorpus <- Corpus(DataframeSource(five)) #Creating corpus (collection of text data)

clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus,function(x) removeWords(x,stopwords("english")))
  return(corpus)
}

mycorpus <- clean_corpus(mycorpus)
tdm <- TermDocumentMatrix(mycorpus) #Creating term-document matrix
m <- as.matrix(tdm)

#here we merge all rows
v <- sort(rowSums(m),decreasing=TRUE) #Sum up the frequencies of each word
d <- data.frame(word = names(v),freq=v) #Create one column=names, second=frequencies

pal<-brewer.pal(6,"PRGn") #Create palette of colors
#color_pal <- cividis(n = 8)

#wordCloud(d$word,d$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))

wordcloud(d$word,d$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))

oneTwo<-read.table("oneTwo.txt",header=F,sep="\n")
oneTwo$doc_id=1:nrow(oneTwo)
colnames(oneTwo)[1]<-"text"

mycor <- Corpus(DataframeSource(oneTwo))

mycor <- clean_corpus(mycor)
tdm1<- TermDocumentMatrix(mycor) #Creating term-document matrix
m1 <- as.matrix(tdm1)

#here we merge all rows
v1 <- sort(rowSums(m1),decreasing=TRUE) #Sum up the frequencies of each word
d1<- data.frame(word = names(v1),freq=v1)
pal<-brewer.pal(8,"Dark2")

wordcloud(d1$word,d1$freq, scale=c(8,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=col_pal, vfont=c("sans serif","plain"))

j<-left_join(d,d1,by="word")

pyramid.plot(
  j$freq.x[1:10],j$freq.y[1:10],
  # Words
  labels = j$word[1:10],
  top.labels = c("Words in File", "Common Words", "Words in oneTwo"),
  main = "Pyramid Plot",gap=17,unit="counts")

library(tidyverse)
library(plotly)
library(plyr)
library(crosstalk)

```

