732A98: Visualization - Cheatbook

Anubhav Dikshit (anudi287)

02 Nov 2018

Contents

```
# Loading required R packages
library(ggplot2)
library(plotly)
library(shiny)
library(gridExtra)
library(xlsx)
library(MASS)
library(sf)
library(akima)
library(scales)
library(seriation)
library(dplyr)
library(crosstalk)
library(GGally)
library(tm)
library(wordcloud)
library(RColorBrewer)
library(htmltools)
library(tourr)
library(reshape)
library(ggraph)
library(igraph)
library(visNetwork)
library(data.table)
library(reshape2)

Sys.setenv('MAPBOX_TOKEN' = 'pk.eyJ1IjoibGFrc2hpZGFhIiwiYSI6ImNqbWIyOHN2NTRlZ3kzam10aTljeGNybWgifQ.8EG9
```

Reading Data

Data Mugging

Quantile Computation

```
get_outliers <- function(x){
  quantile_values = quantile(x, probs = c(0.25, 0.75))
  q1 = quantile_values["25%"]
  q3 = quantile_values["75%"]

  return(c(which((x > (q3+1.5*(q3-q1)))), which(x < (q1-1.5*(q3-q1)))))
}
```

Scaling the Data

```
baseball_scaled <- scale(baseball_data[,3:length(baseball_data)])
```

Distance Matrix between rows

```
distance_matrix <- dist(baseball_scaled, method = "euclidean")
```

Non-metric MDS

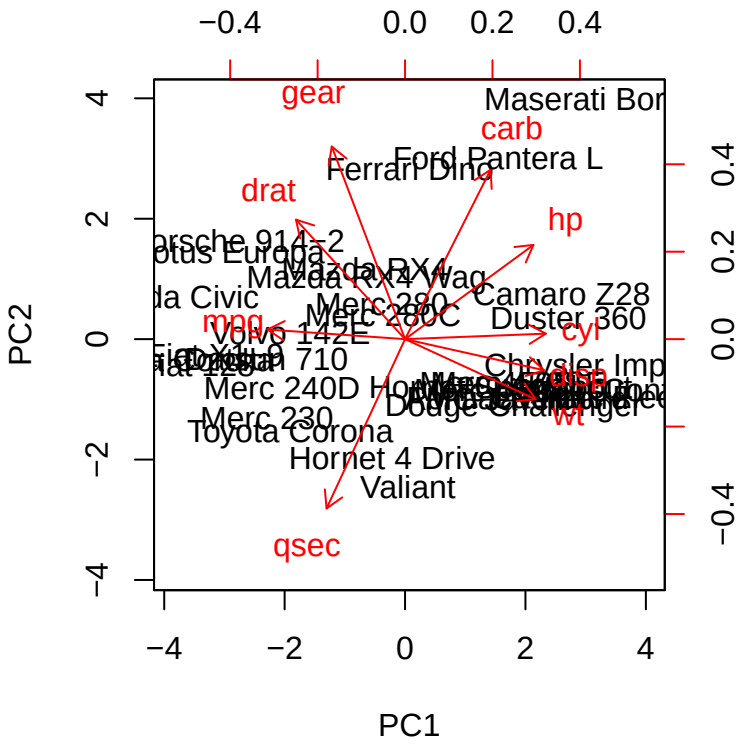
```
mds_result <- isoMDS(distance_matrix, k=2, p=2)
```

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

```
coords <- mds_result$points
coords_mds <- as.data.frame(coords)
baseball_data_with_mds <- baseball_data
baseball_data_with_mds$MDS_V1 <- coords_mds$V1
baseball_data_with_mds$MDS_V2 <- coords_mds$V2
```

Principle Component Analysis

```
mtcars.pca <- prcomp(mtcars[,c(1:7,10,11)], center = TRUE,scale. = TRUE)
biplot(mtcars.pca, scale = 0)
```



Types of Projection

```
#projection = list(type = "mercator"))
#projection = list(type = "albers usa"))
#projection = list(type = "equirectangular"))
#projection = list(type = "conic equal area"))
#projection = list(type = "azimuthal equal area"))
#projection = list(type = "equirectangular"))
#projection = list(type = "orthographic"))
```

Types of easing

```
#animation_opts(500, easing = 'linear', redraw = F)
# animation_opts(500, easing = 'quad', redraw = F)
# animation_opts(500, easing = 'cubic', redraw = F)
# animation_opts(500, easing = 'sin', redraw = F)
# animation_opts(500, easing = 'exp', redraw = F)
# animation_opts(500, easing = 'circle', redraw = F)
# animation_opts(500, easing = 'elastic', redraw = F)
# animation_opts(500, easing = 'back', redraw = F)
# animation_opts(500, easing = 'bounce', redraw = F)
# animation_opts(500, easing = 'linear-in', redraw = F)
# animation_opts(500, easing = 'quad-in', redraw = F)
```

```
# animation_opts(500, easing = 'cubic-in', redraw = F)
# animation_opts(500, easing = 'sin-in', redraw = F)
# animation_opts(500, easing = 'exp-in', redraw = F)
# animation_opts(500, easing = 'circle-in', redraw = F)
# animation_opts(500, easing = 'elastic-in', redraw = F)
# animation_opts(500, easing = 'back-in', redraw = F)
# animation_opts(500, easing = 'bounce-in', redraw = F)
# animation_opts(500, easing = 'linear-out', redraw = F)
# animation_opts(500, easing = 'quad-out', redraw = F)
# animation_opts(500, easing = 'cubic-out', redraw = F)
# animation_opts(500, easing = 'sin-out', redraw = F)
# animation_opts(500, easing = 'exp-out', redraw = F)
# animation_opts(500, easing = 'circle-out', redraw = F)
# animation_opts(500, easing = 'elastic-out', redraw = F)
# animation_opts(500, easing = 'back-out', redraw = F)
# animation_opts(500, easing = 'bounce-out', redraw = F)
# animation_opts(500, easing = 'linear-in-out', redraw = F)
# animation_opts(500, easing = 'quad-in-out', redraw = F)
# animation_opts(500, easing = 'cubic-in-out', redraw = F)
# animation_opts(500, easing = 'sin-in-out', redraw = F)
# animation_opts(500, easing = 'exp-in-out', redraw = F)
# animation_opts(500, easing = 'circle-in-out', redraw = F)
# animation_opts(500, easing = 'elastic-in-out', redraw = F)
# animation_opts(500, easing = 'back-in-out', redraw = F)
# animation_opts(500, easing = 'bounce-in-out', redraw = F)
```

Sorting dataset

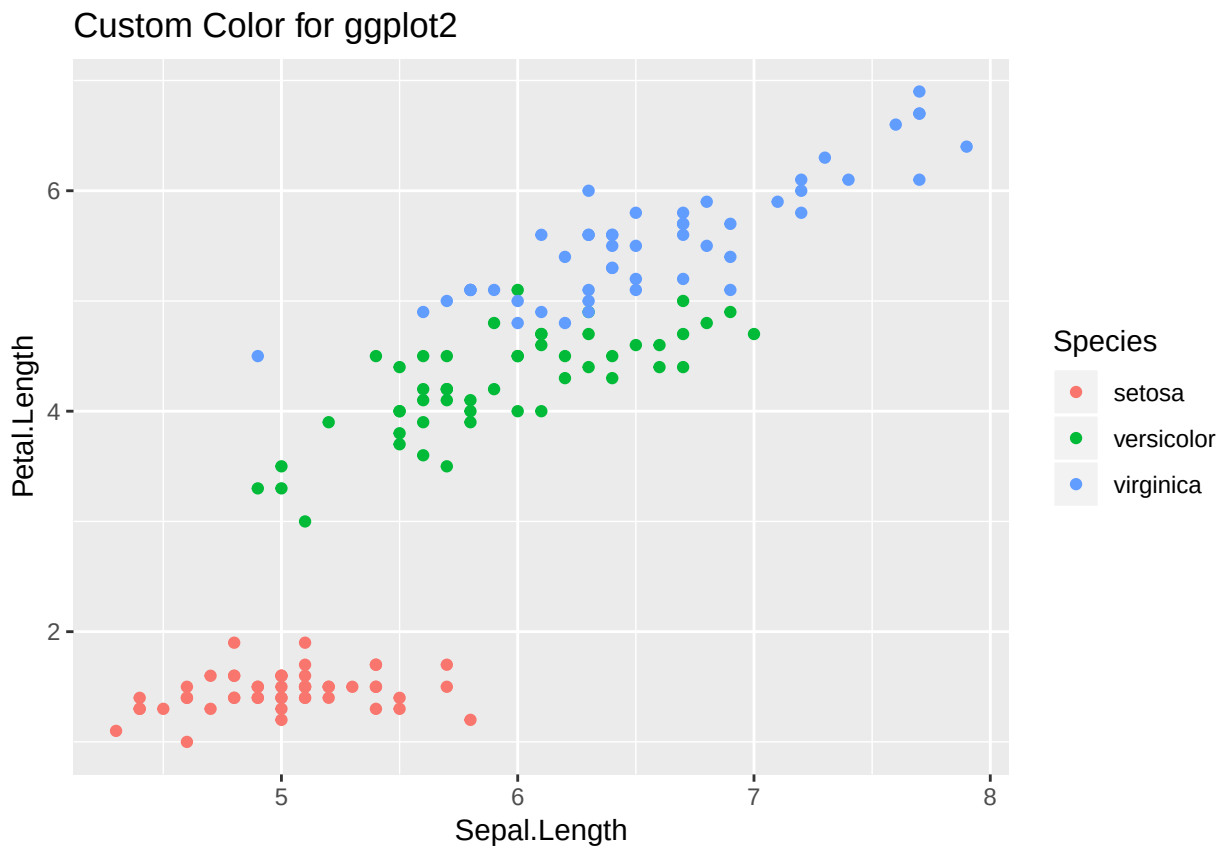
```
mtcars <- as.data.table(mtcars)
mtcars <- mtcars[order(-mpg, cyl)]

head(mtcars, 10)
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
##  1: 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
##  2: 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
##  3: 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
##  4: 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
##  5: 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
##  6: 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
##  7: 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
##  8: 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
##  9: 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## 10: 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
```

Colour selection palette

```
# The palette with grey:
default_colors <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79
```

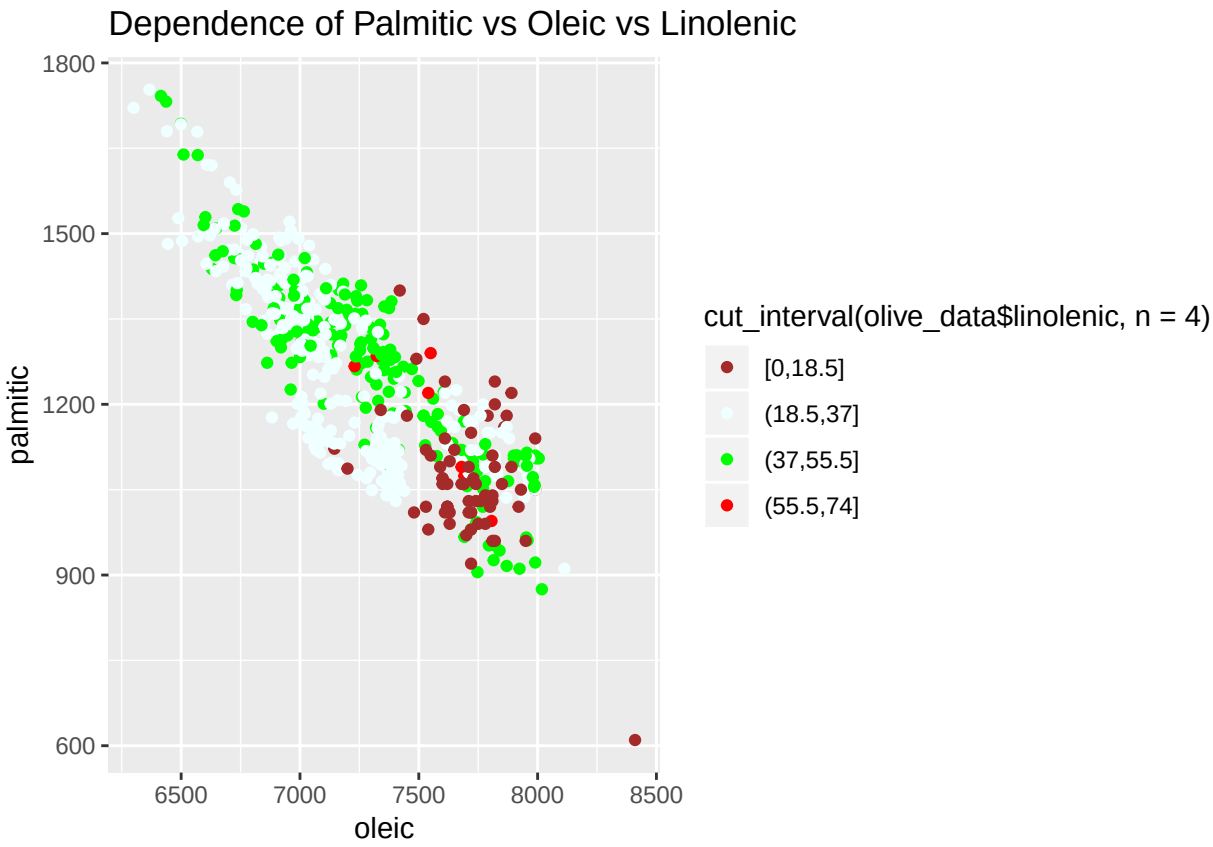
```
# ggplot2
ggplot(data=iris, aes(x= Sepal.Length, y = Petal.Length, color = Species)) + geom_point() + scale_fill_
```



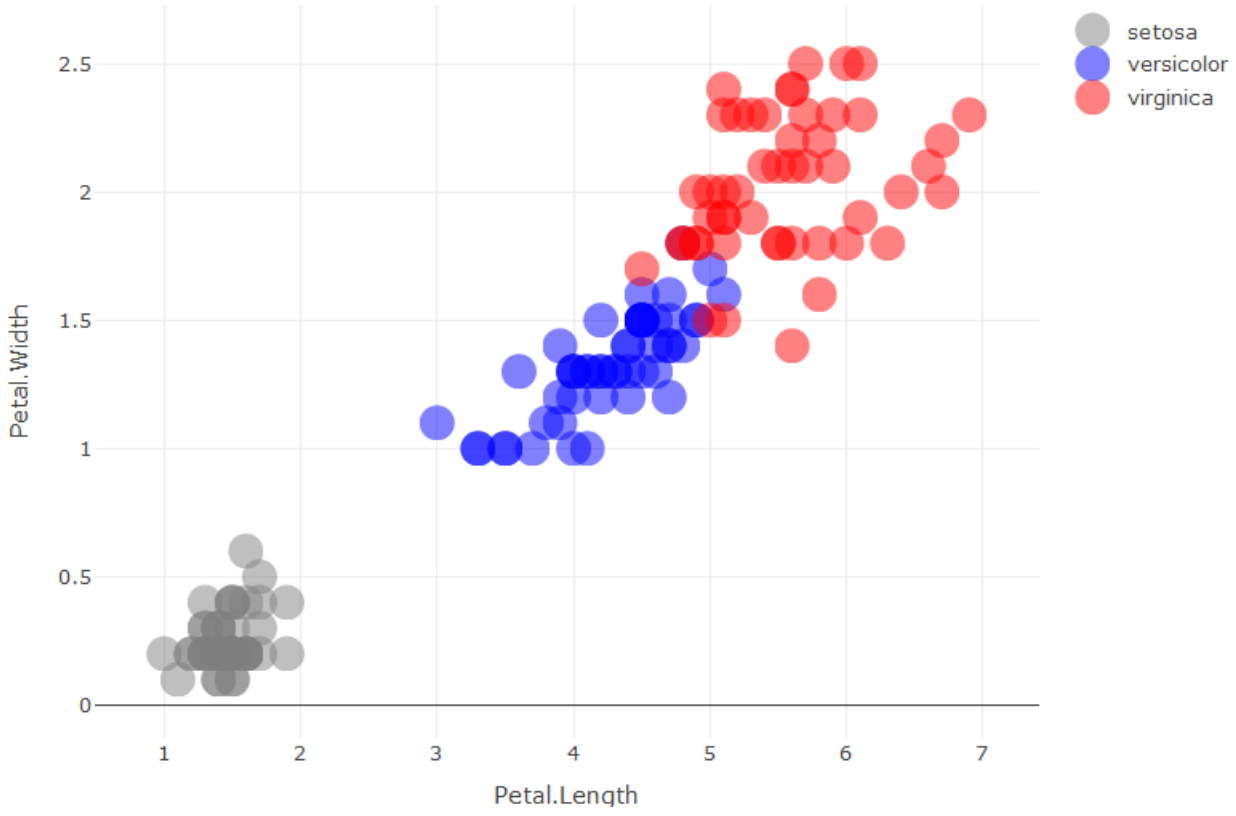

```
# plotly
x <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length, color = ~Species, colors = default_colo
```

Adding custom colours without palette

```
# ggplot2
ggplot(olive_data) +
  geom_point(aes(x = oleic, y = palmitic, color=cut_interval(olive_data$linolenic, n = 4))) +
  ggtitle("Dependence of Palmitic vs Oleic vs Linolenic") +
  scale_colour_manual(values=c("brown", "azure", "green", "red"))
```




```
# plotly
x <- plot_ly(iris, x = ~Petal.Length, y = ~Petal.Width,
        type="scatter", mode = "markers" , color = ~Species,
        colors = c("grey50", "blue", "red"), marker=list( size=20 , opacity=0.5))

knitr::include_graphics('./plotly_custom_color.png')
```



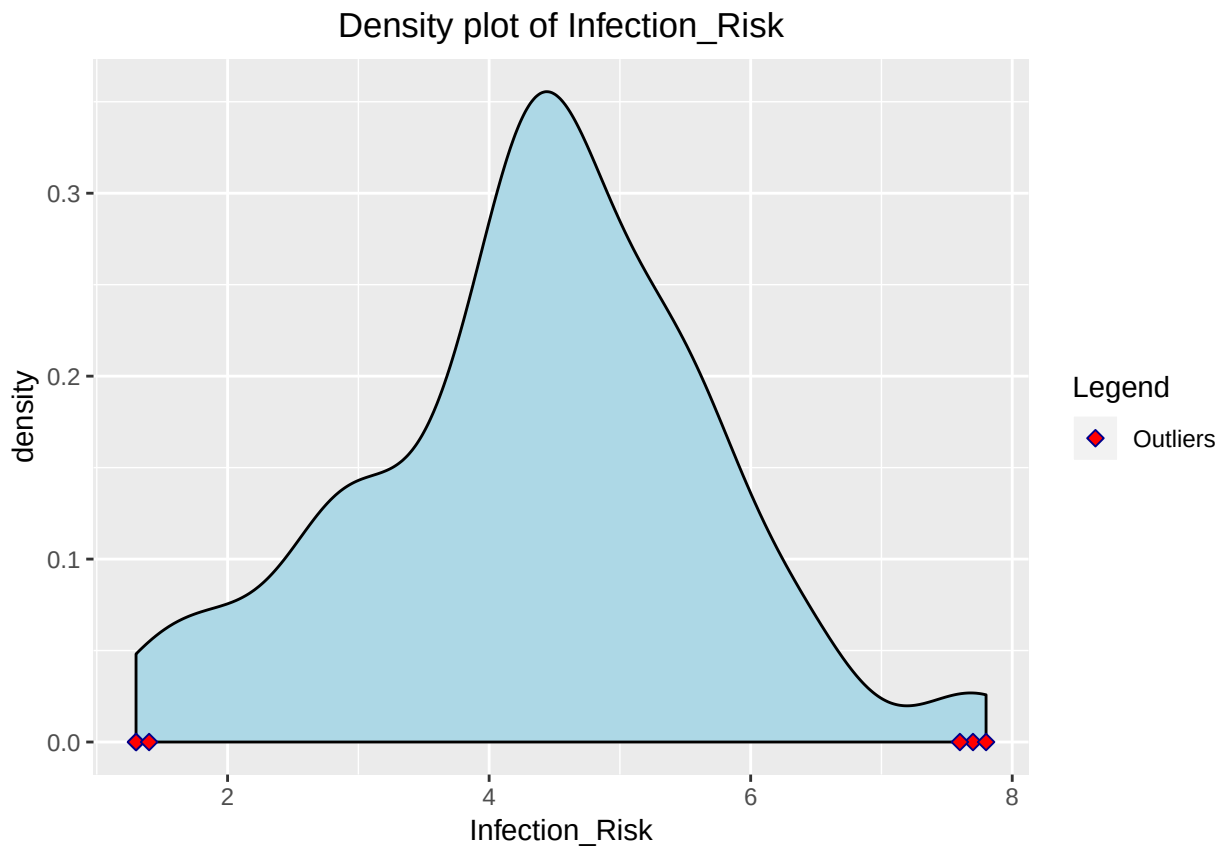
‘‘‘

Single Plots

Density Plot

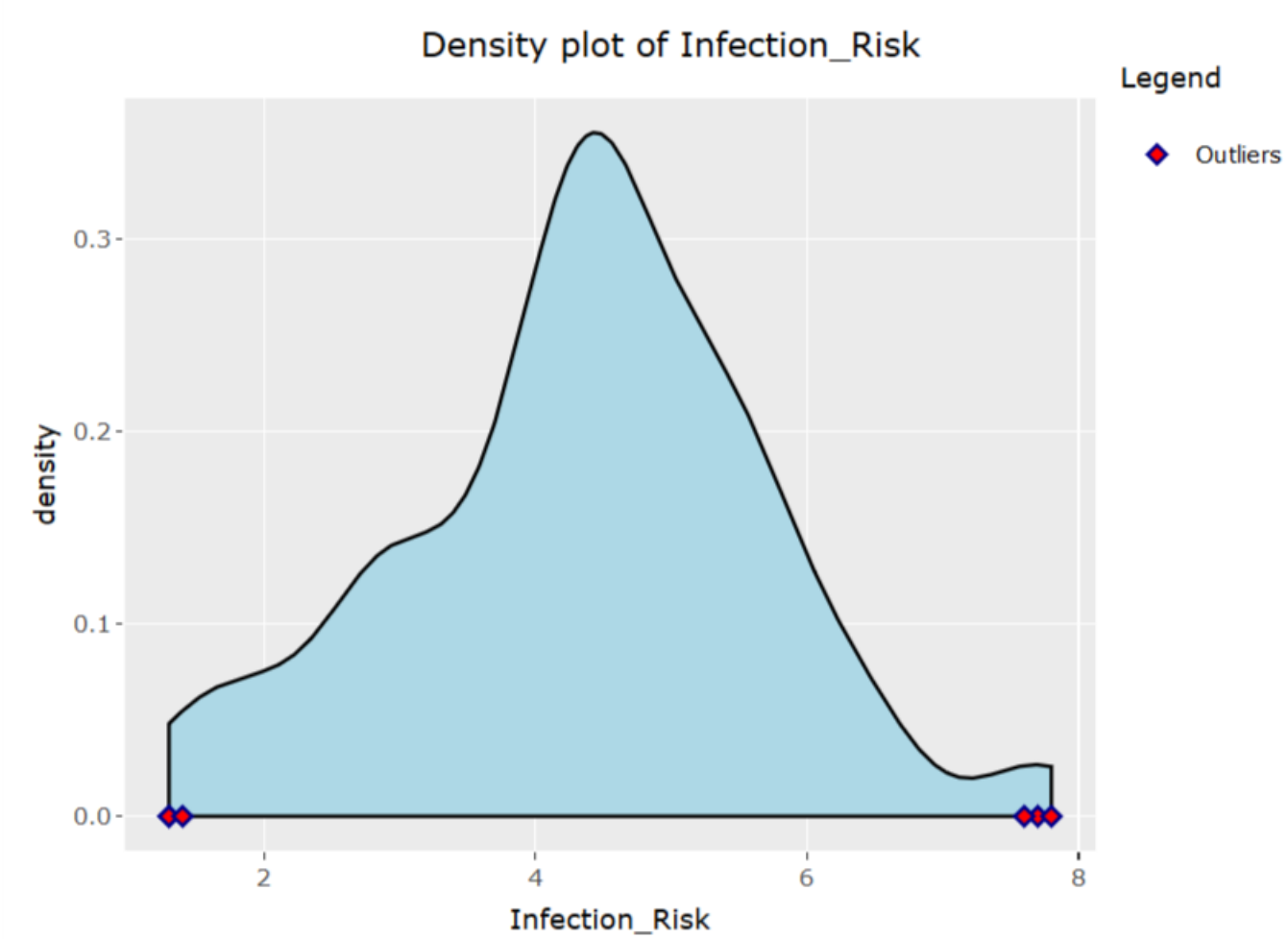
Density Plot with Outlier Highlight using GGplot2

```
density_plot_infection_risk = ggplot(senic_data) +
  ggtitle("Density plot of Infection_Risk")  +
  geom_density(aes(x=Infection_Risk), fill = "lightblue") +
  geom_point(data=senic_data[get_outliers(senic_data$Infection_Risk),],
             aes(x=Infection_Risk, y=0, colour="Outliers"),
             shape=23, size=2, fill="red") +
  scale_color_manual(values = c("darkblue","black")) +
  labs(colour="Legend") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")

density_plot_infection_risk
```



Density Plot with Outlier Highlight using Plotly (converting from ggplot2)

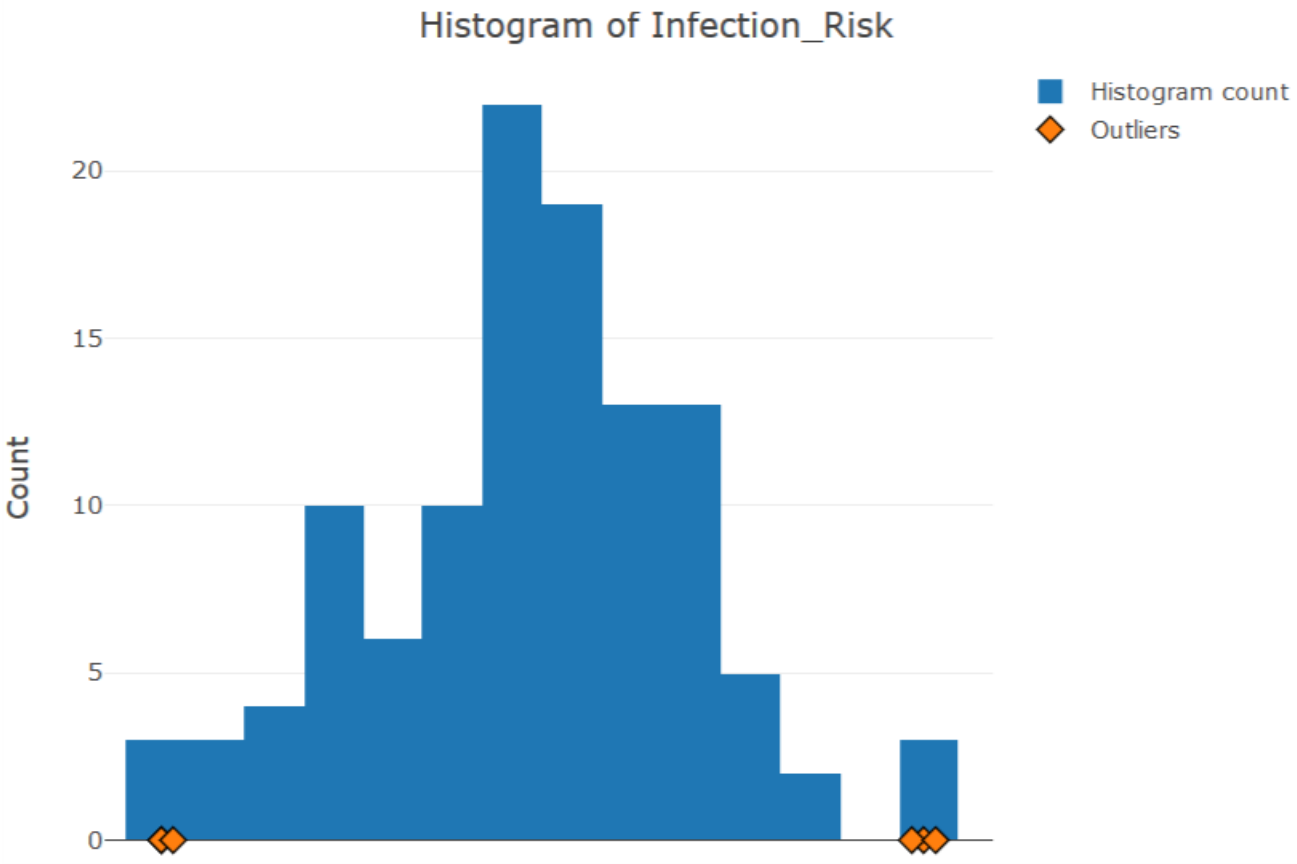
```
x <- ggplotly(p=density_plot_infection_risk)
knitr::include_graphics('./3.1.2.png')
```



Histogram Plot

Histogram plot with Outlier Highlight using Plotly

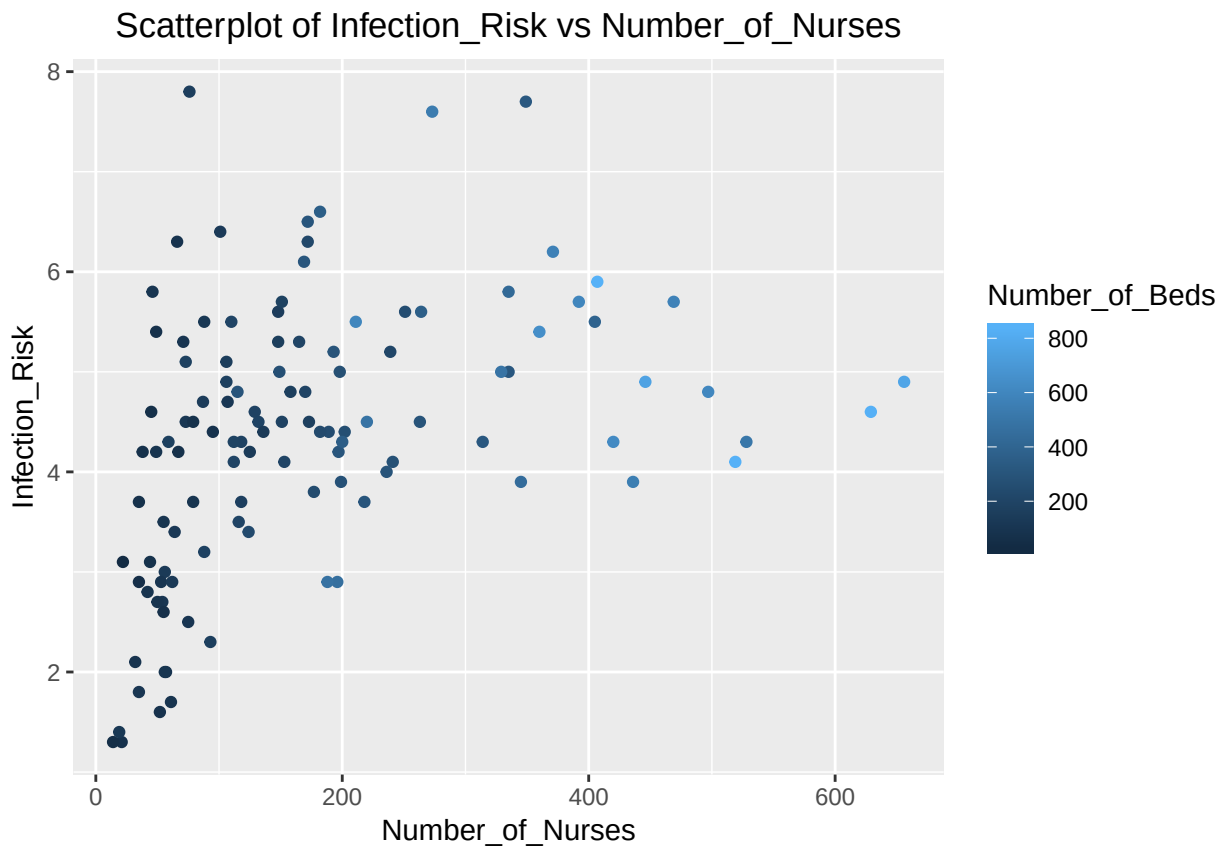
```
outliers = senic_data[get_outliers(senic_data$Infection_Risk),c("Infection_Risk")]
senic_data$zero = 0

x <- plot_ly(senic_data, x=~Infection_Risk) %>%
  add_histogram(name="Histogram count") %>%
  filter(is.element(Infection_Risk, outliers)) %>%
  add_markers(x=~Infection_Risk,y=~zero, name="Outliers",
              marker=list(symbol="diamond", size=10, line = list(color="black", width=1))) %>%
  layout(title="Histogram of Infection_Risk", yaxis=list(title="Count"))

knitr::include_graphics('./3.2.1.png')
```



Scatter Plot

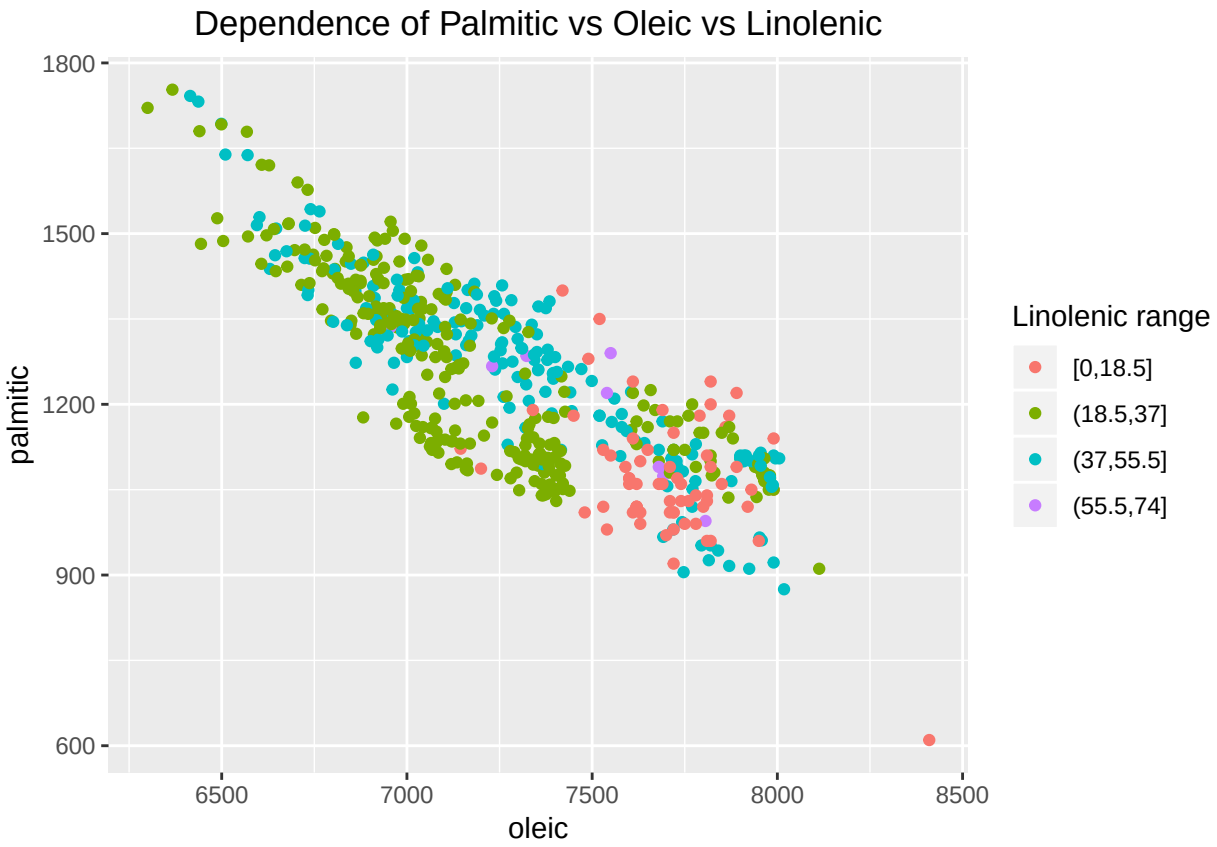
Simple scatter plot with colour

```
ggplot(senic_data) + geom_point(aes(x=Number_of_Nurses, y=Infection_Risk, color=Number_of_Beds)) +
  ggtitle("Scatterplot of Infection_Risk vs Number_of_Nurses") +
  theme(plot.title = element_text(hjust = 0.5))
```



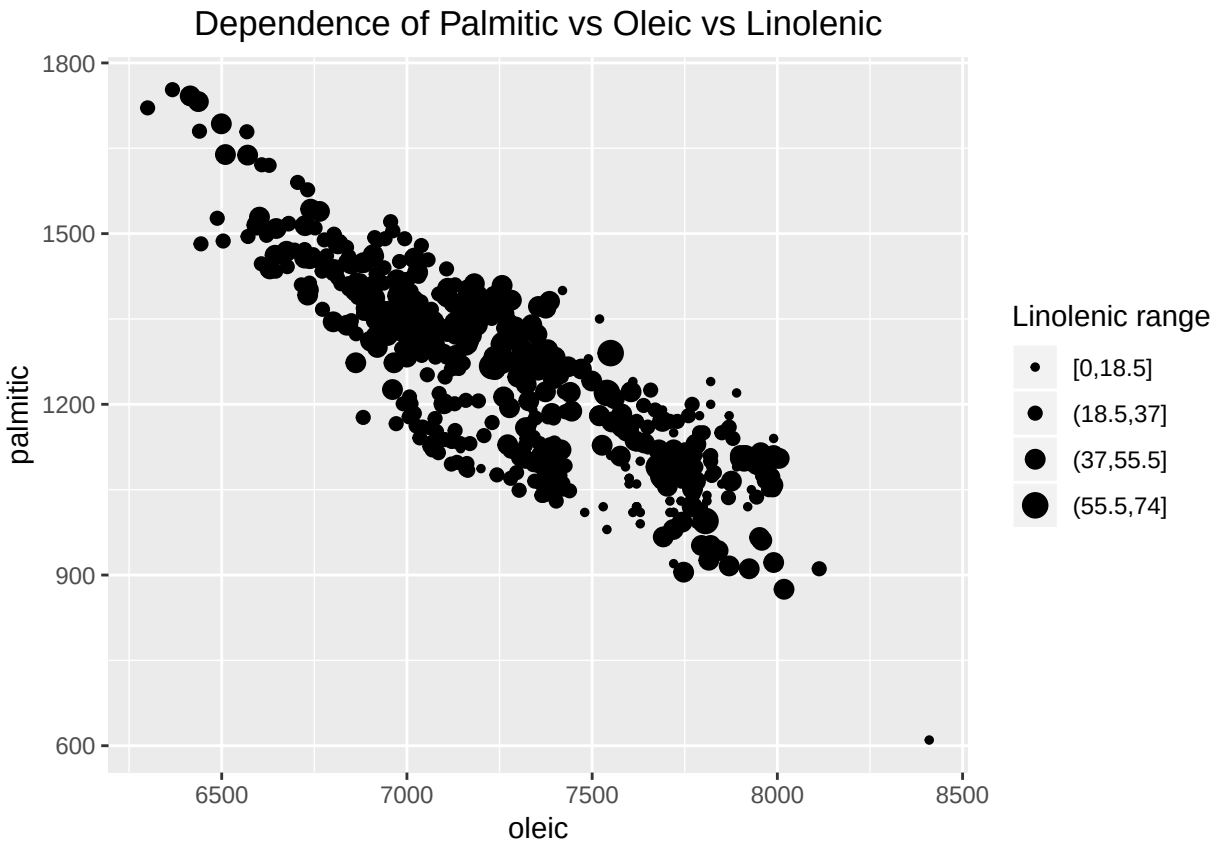
Scatter Plot with Discreetization (split a variable into classes)

```
ggplot(olive_data) +
  geom_point(aes(x = oleic, y = palmitic,
                 color=cut_interval(olive_data$linolenic, n = 4))) +
  ggtitle("Dependence of Palmitic vs Oleic vs Linolenic") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(color = 'Linolenic range')
```



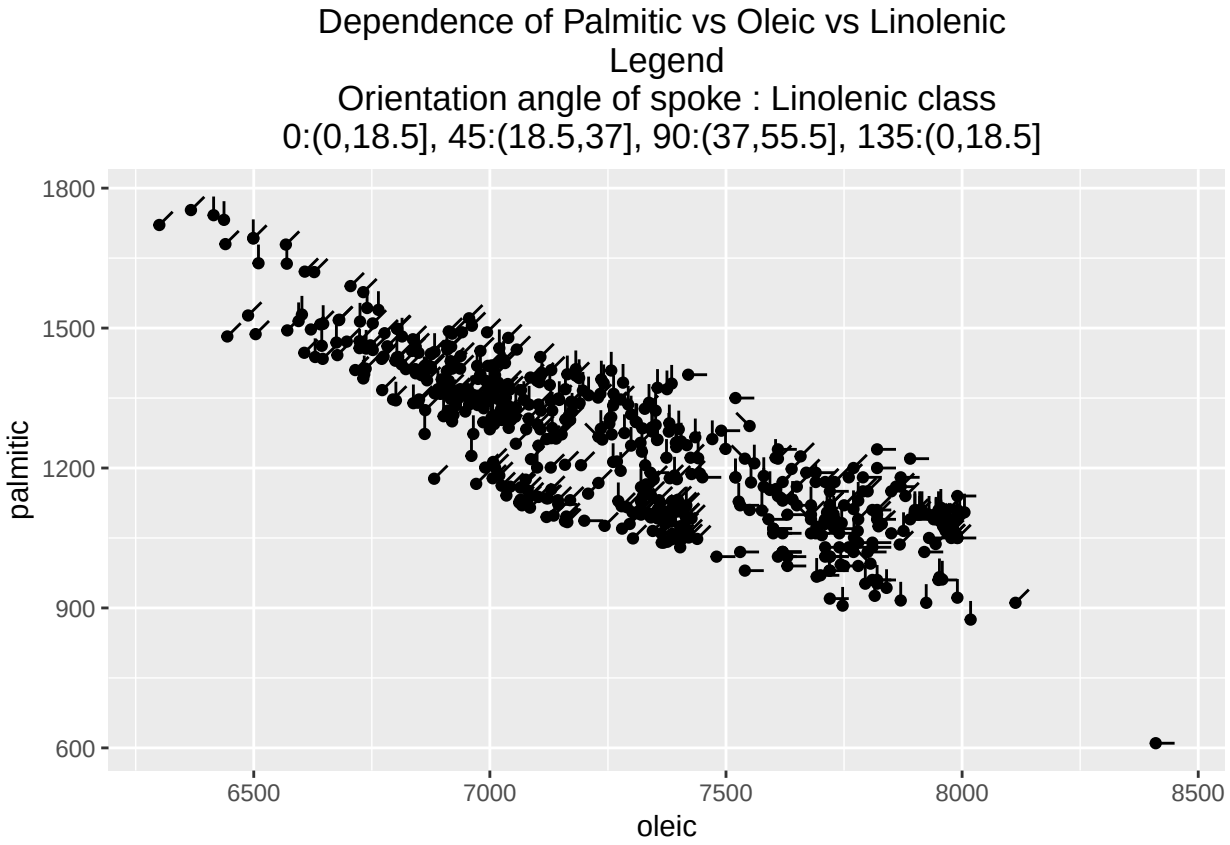
Scatter plot size varied

```
ggplot(olive_data) + geom_point(aes(x = oleic, y = palmitic, size = cut_interval(linolenic, n = 4))) +
  ggtitle("Dependence of Palmitic vs Oleic vs Linolenic") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_size_manual(name = "Linolenic range", values = c(1, 2, 3, 4))
```



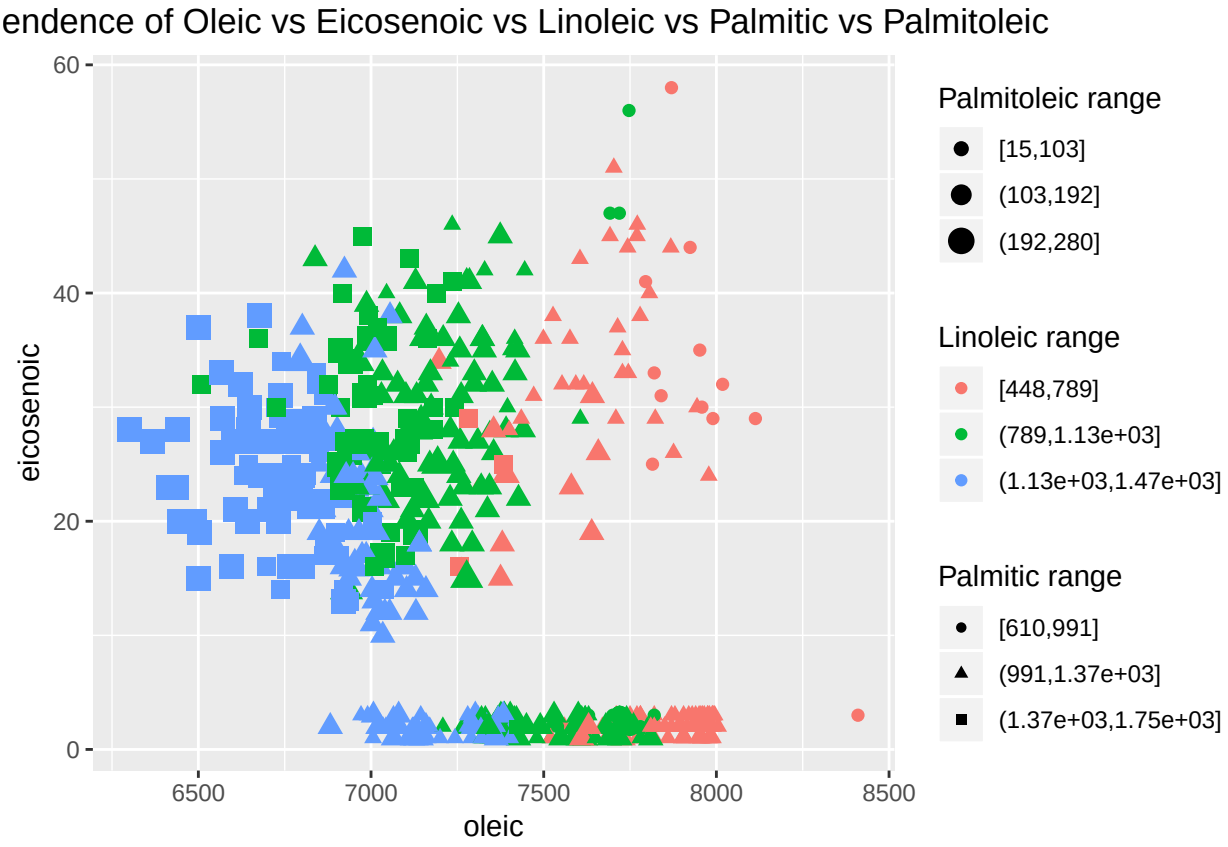
Scatter plot angle varied

```
# Pre-processing - Setting angle values based on category
olive_data$linolenic_class <- cut_interval(olive_data$linolenic, n = 4)
levels(olive_data$linolenic_class) <- (0:3) * (pi/4)
olive_data$linolenic_class <- as.numeric(as.character(olive_data$linolenic_class))

ggplot(olive_data, aes(x=oleic, y=palmitic)) + geom_point() +
  geom_spoke(aes(angle = olive_data$linolenic_class), radius=40) +
  ggtitle("Dependence of Palmitic vs Oleic vs Linolenic
Legend
Orientation angle of spoke : Linolenic class
0:(0,18.5], 45:(18.5,37], 90:(37,55.5], 135:(0,18.5] ") +
  theme(plot.title = element_text(hjust = 0.5))
```



Scatter plot with color, size and shape

```
ggplot(olive_data)+
  geom_point(aes(x = oleic, y = eicosenoic, color = cut_interval(linoleic, n = 3),
                            shape = cut_interval(palmitic, n = 3),
                            size = cut_interval(palmitoleic, n = 3))) +
  scale_size_manual(values = c(2,3,4)) +
  labs(shape = "Palmitic range", color = "Linoleic range", size = "Palmitoleic range") +
  ggtitle("Dependence of Oleic vs Eicosenoic vs Linoleic vs Palmitic vs Palmitoleic") +
  theme(plot.title = element_text(hjust = 0.5))
```



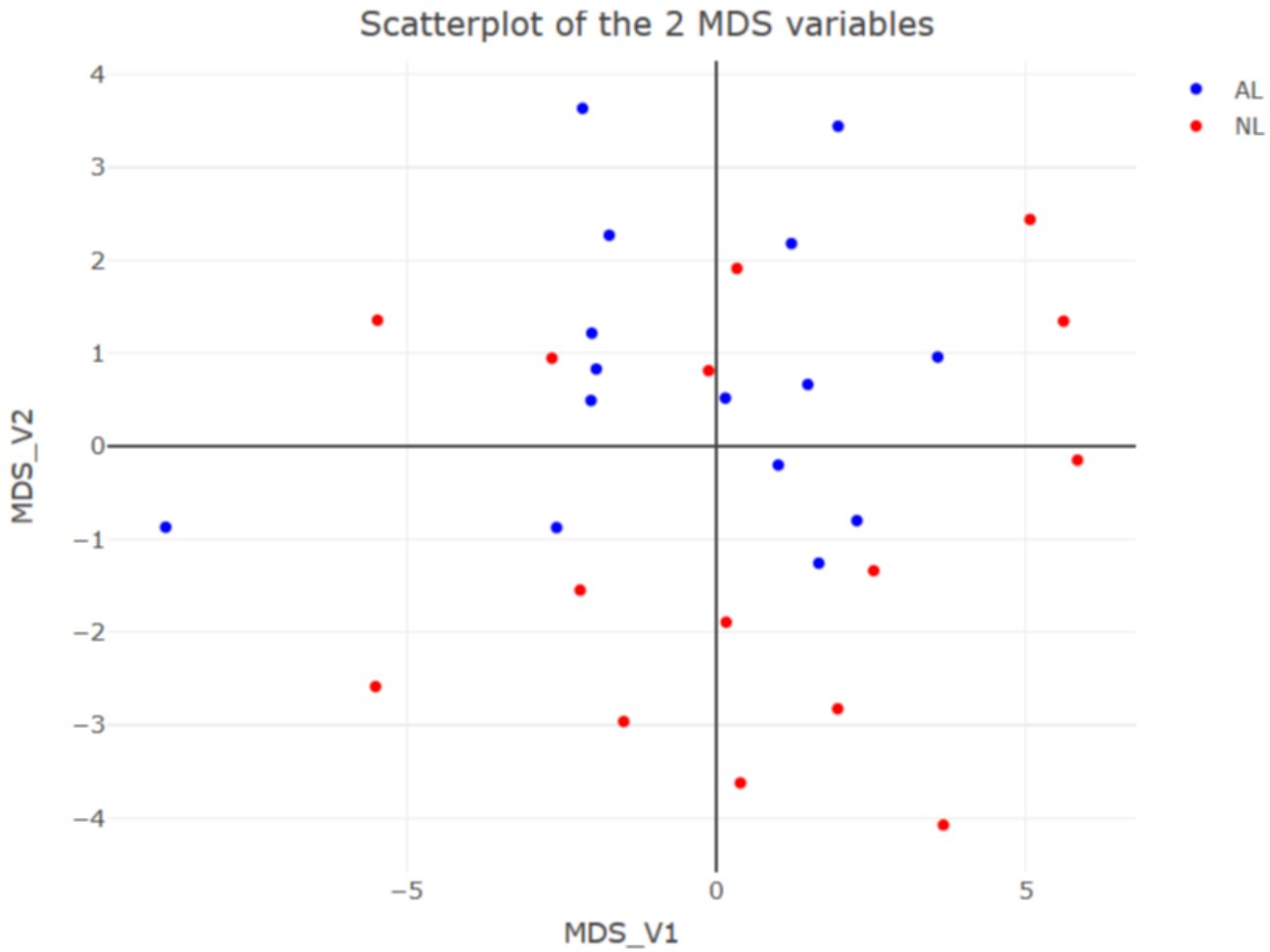
Scatter Plot of MDS

```
baseball_scaled <- scale(baseball_data[,3:length(baseball_data)])

### Distance Matrix between rows
distance_matrix <- dist(baseball_scaled, method = "euclidean")

### Non-metric MDS
mds_result <- isoMDS(distance_matrix, k=2, p=2)
```

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

```
coords <- mds_result$points
coords_mds <- as.data.frame(coords)
baseball_data_with_mds <- baseball_data
baseball_data_with_mds$MDS_V1 <- coords_mds$V1
baseball_data_with_mds$MDS_V2 <- coords_mds$V2



x <- plot_ly(baseball_data_with_mds, x=~MDS_V1, y=~MDS_V2) %>%
```

```
  add_markers(color=~League, colors = c("blue", "red"),
              text=baseball_data_with_mds$Team, hoverinfo="text") %>%
  layout(title="Scatterplot of the 2 MDS variables")

knitr::include_graphics('./3.3.6.png')
```



Shepard Plot

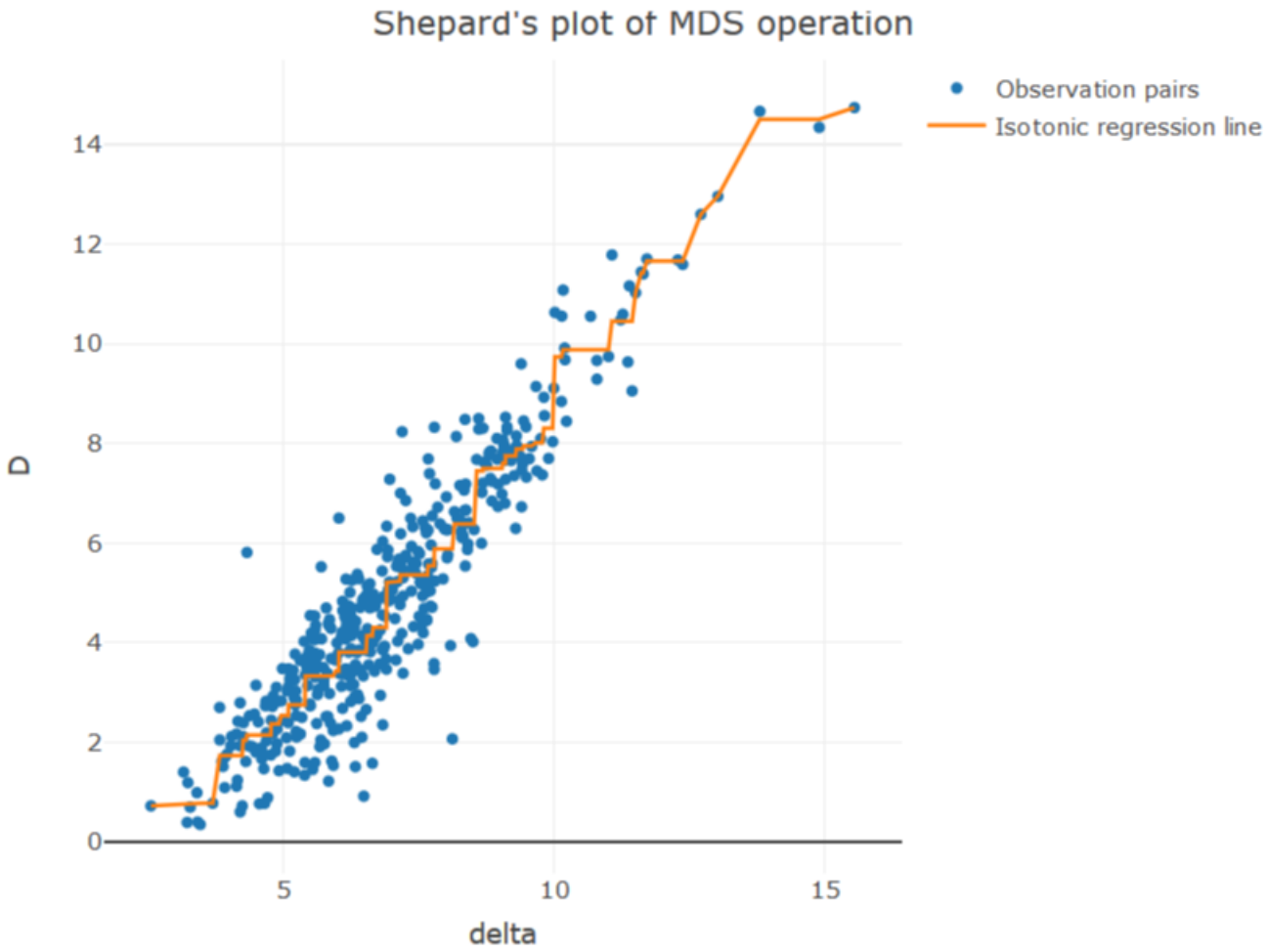
Shepard plot shows the fit of MDS, the distance in original dataset vs. distance in reordered dataset should be similar

```
baseball_scaled <- scale(baseball_data[,3:length(baseball_data)])

### Distance Matrix between rows
distance_matrix <- dist(baseball_scaled, method = "euclidean")
mds_result <- isoMDS(distance_matrix, k=2, p=2)
```

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

```
## converged
```

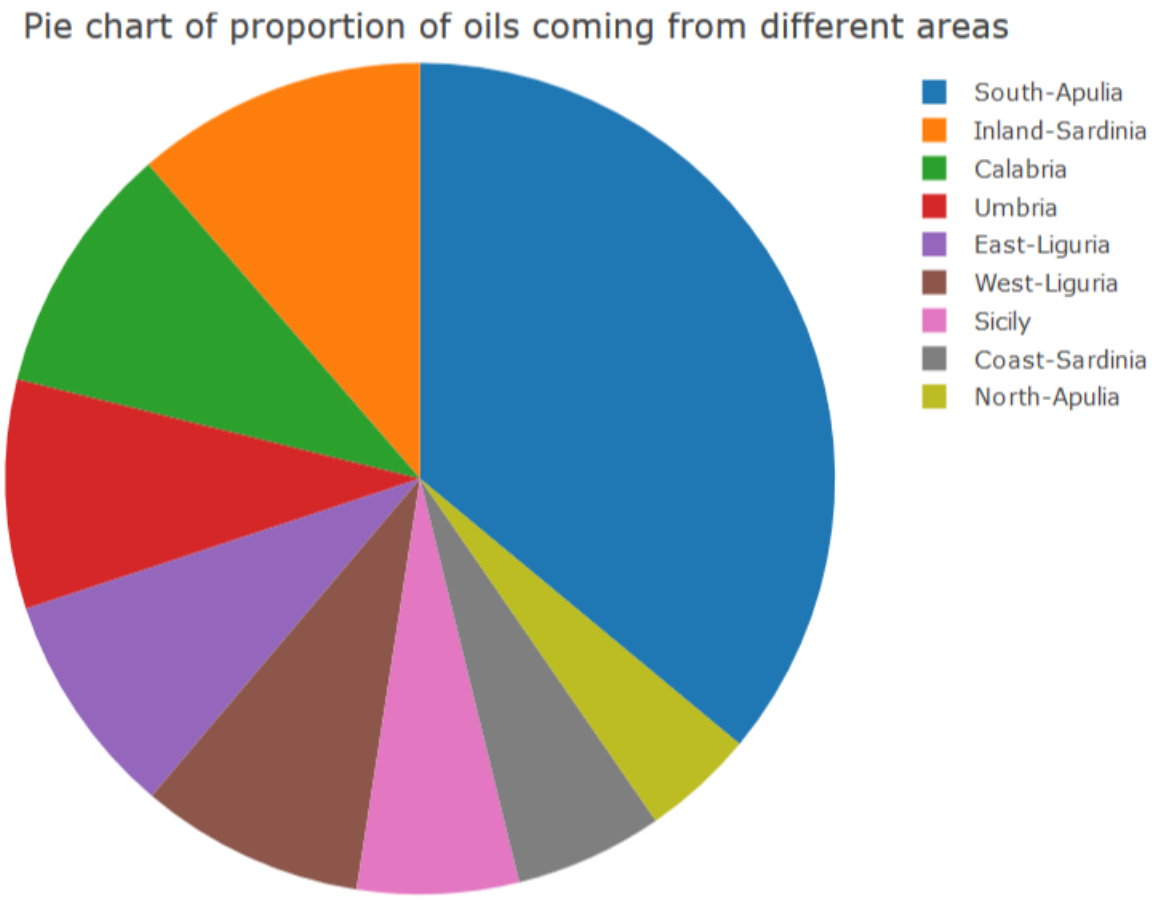
```
coords <- mds_result$points

shp <- Shepard(distance_matrix, coords)
delta <- as.numeric(distance_matrix)
D <- as.numeric(dist(coords, method = "euclidean"))

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

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


x <- plot_ly()%>%
  add_markers(x=~delta, y=~D, name="Observation pairs", hoverinfo = 'text',
              text = ~paste('Obj 1: ',
                            rownames(baseball_data_with_mds)[index1],
                            '<br> Obj 2: ',
                            rownames(baseball_data_with_mds)[index2])) %>%
  add_lines(x=~shp$x, y=~shp$yf, name="Isotonic regression line") %>%
  layout(title="Shepard's plot of MDS operation")

knitr::include_graphics('./3.4.1.png')
```



Pie Charts

Simple pie chart

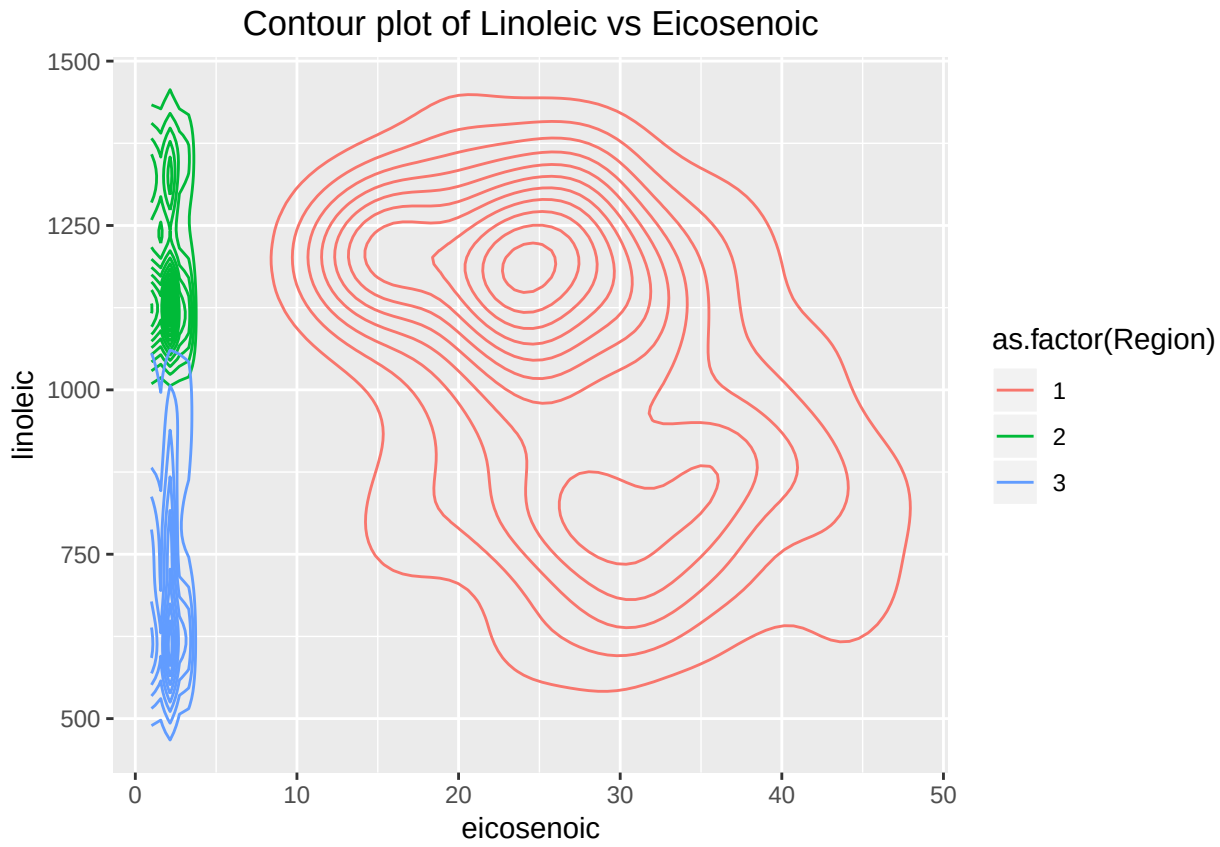
```
x <- plot_ly(olive_data,labels=~Area,type='pie',textinfo = "none") %>%
  layout(title = "Pie chart of proportion of oils coming from different areas")

knitr::include_graphics('./3.5.1.png')
```



2D density contour plot

Simple 2D plot using ggplot2

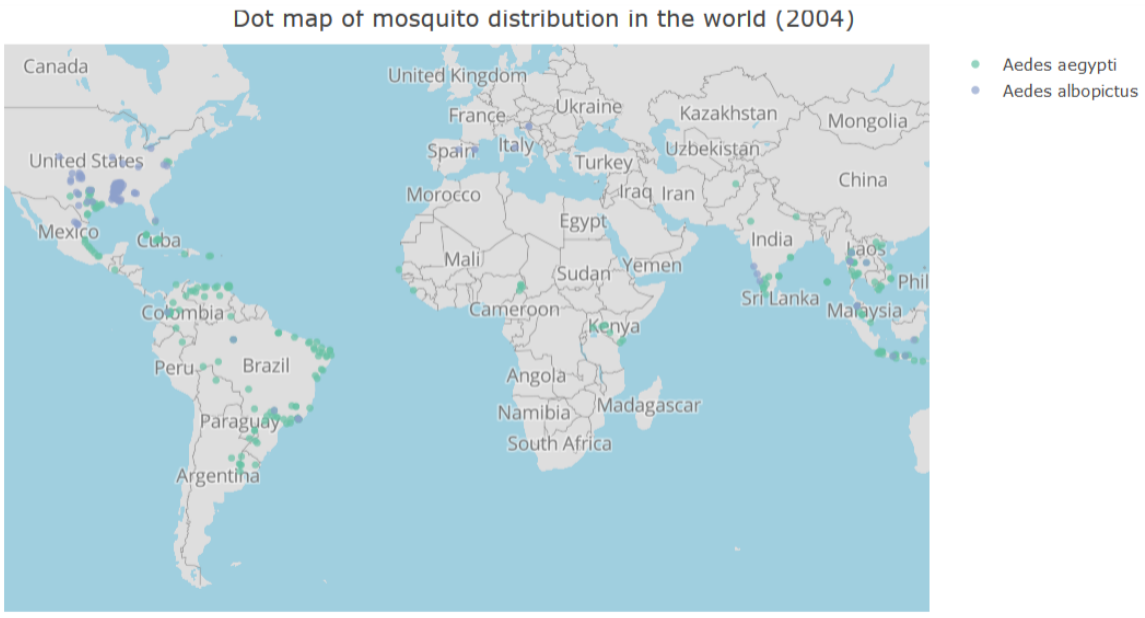
```
ggplot(olive_data)+geom_density_2d(aes(x=eicosenoic, y=linoleic, colour=as.factor(Region))) +
  ggtitle("Contour plot of Linoleic vs Eicosenoic") +
  theme(plot.title = element_text(hjust = 0.5))
```



Dot Map Plots (World Map)

Dot Map (Map with scatter plots)

```
x <- plot_mapbox(aegypti_data[aegypti_data$YEAR == 2004,], lat = ~Y, lon = ~X) %>%
  add_markers(color = ~VECTOR, hoverinfo = "text",
              text = ~paste(COUNTRY), alpha = 0.7) %>%
  layout(title = "Dot map of mosquito distribution in the world (2004)")

knitr::include_graphics('./3.7.1.png')
```



Draw line between two places

Plotly between two places

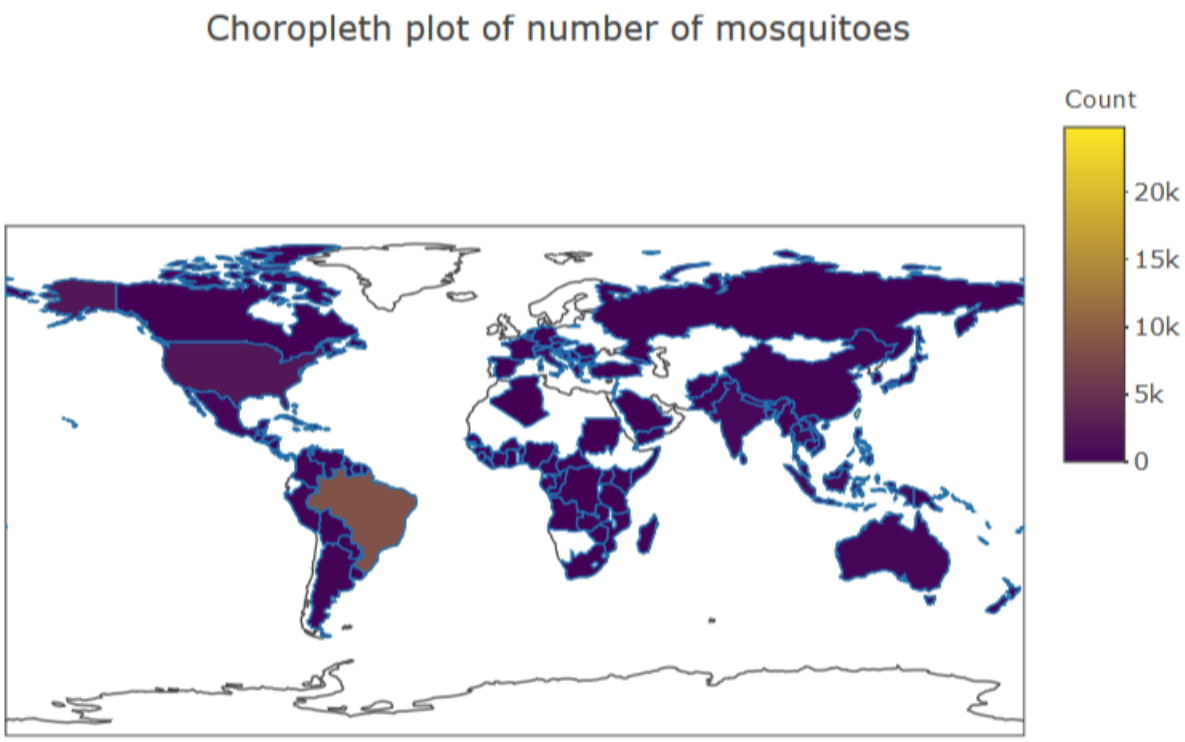
```
x <- plot_geo(lat = c(40.7127, 51.5072), lon = c(-74.0059, 0.1275)) %>%
  add_lines(color = I("blue"), size = I(2)) %>%
  layout(
    title = 'London to NYC Great Circle',
    showlegend = FALSE,
    geo = list(
      resolution = 50,
      showland = TRUE,
      showlakes = TRUE,
      landcolor = toRGB("grey80"),
      countrycolor = toRGB("grey80"),
      lakecolor = toRGB("white"),
      projection = list(type = "equirectangular"),
      coastlinewidth = 2,
      lataxis = list(
        range = c(20, 60),
        showgrid = TRUE,
        tickmode = "linear",
        dtick = 10
      ),
      lonaxis = list(
        range = c(-100, 20),
        showgrid = TRUE,
        tickmode = "linear",
        dtick = 20
      )
    )
```

```
  )

knitr::include_graphics('./world_map.png')
```

Choropleth Map

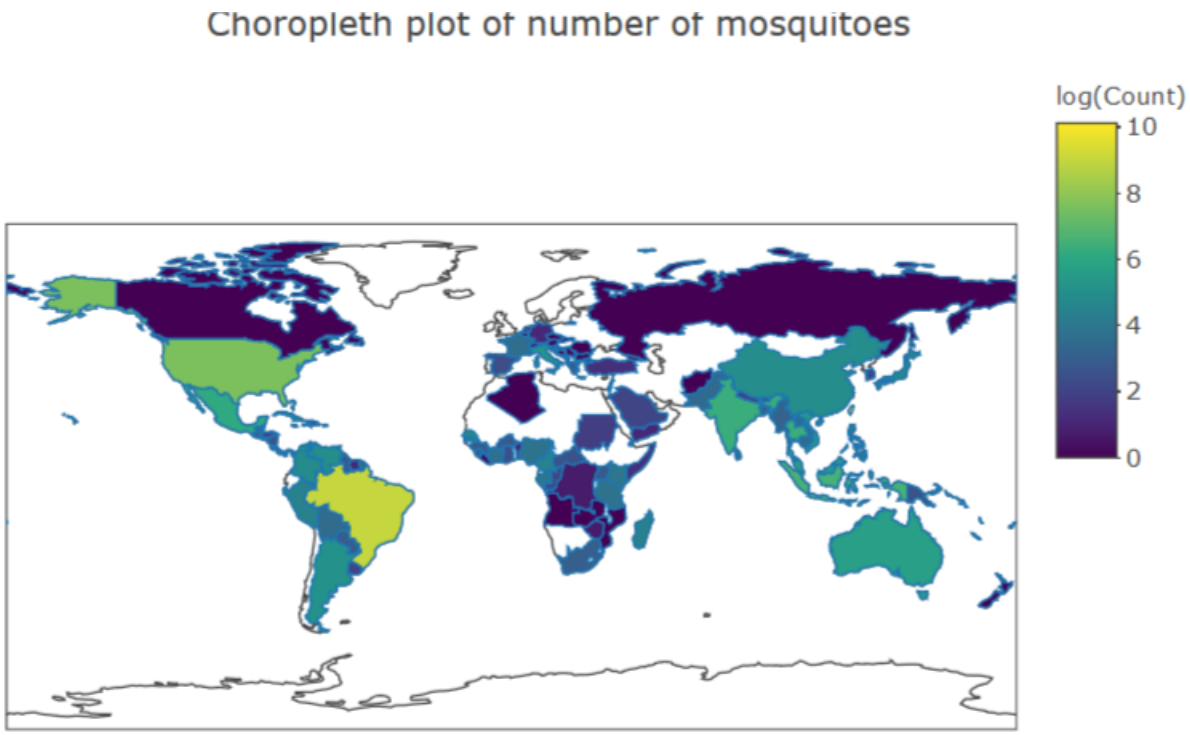
Chorpleth Map with Equirectangular Projection

```
# Data aggregation
country_aggregate = aggregate(aegypti_data[,c("COUNTRY", "COUNTRY_ID")],
                              by = list(aegypti_data$COUNTRY, aegypti_data$COUNTRY_ID), FUN=length)
country_aggregate$COUNTRY = NULL
colnames(country_aggregate) = c("COUNTRY", "COUNTRY_ID", "Count")

x <- plot_geo(country_aggregate) %>% add_trace(
    z = ~Count,
    text = ~COUNTRY, locations = ~COUNTRY_ID) %>%
  layout(title = "Choropleth plot of number of mosquitoes",
    geo = list(projection = list(type = "equirectangular")))

knitr::include_graphics('./3.8.1.png')
```



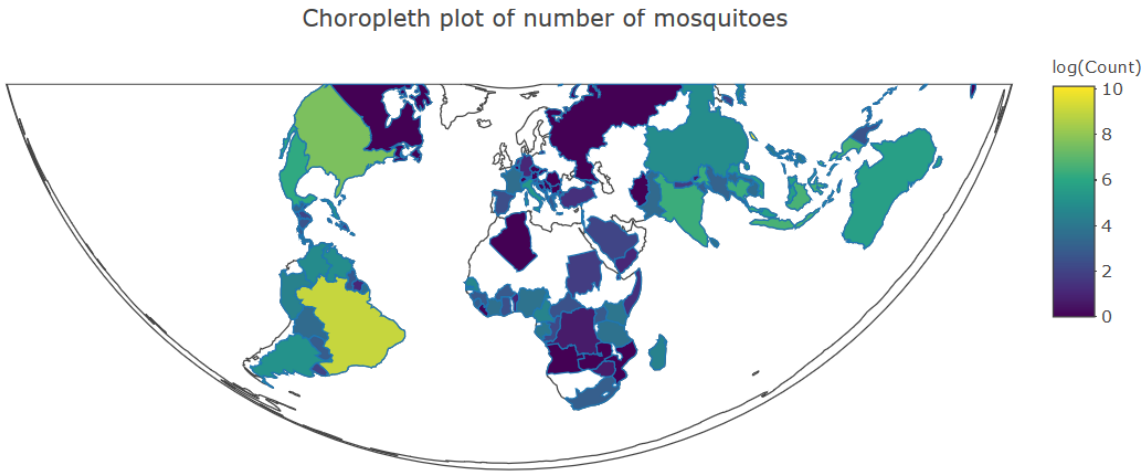
Choropleth plot with Equirectangular Projection and log

```
x <- plot_geo(country_aggregate) %>%
  add_trace(
    z = ~log(Count) ,
    text = ~paste(COUNTRY, "\n Count: ", Count), locations = ~COUNTRY_ID,
    hoverinfo = "text"
  ) %>%
  layout(
    title = "Choropleth plot of number of mosquitoes",
    geo = list(projection = list(type = "equirectangular")))

knitr::include_graphics('./3.8.2.png')
```



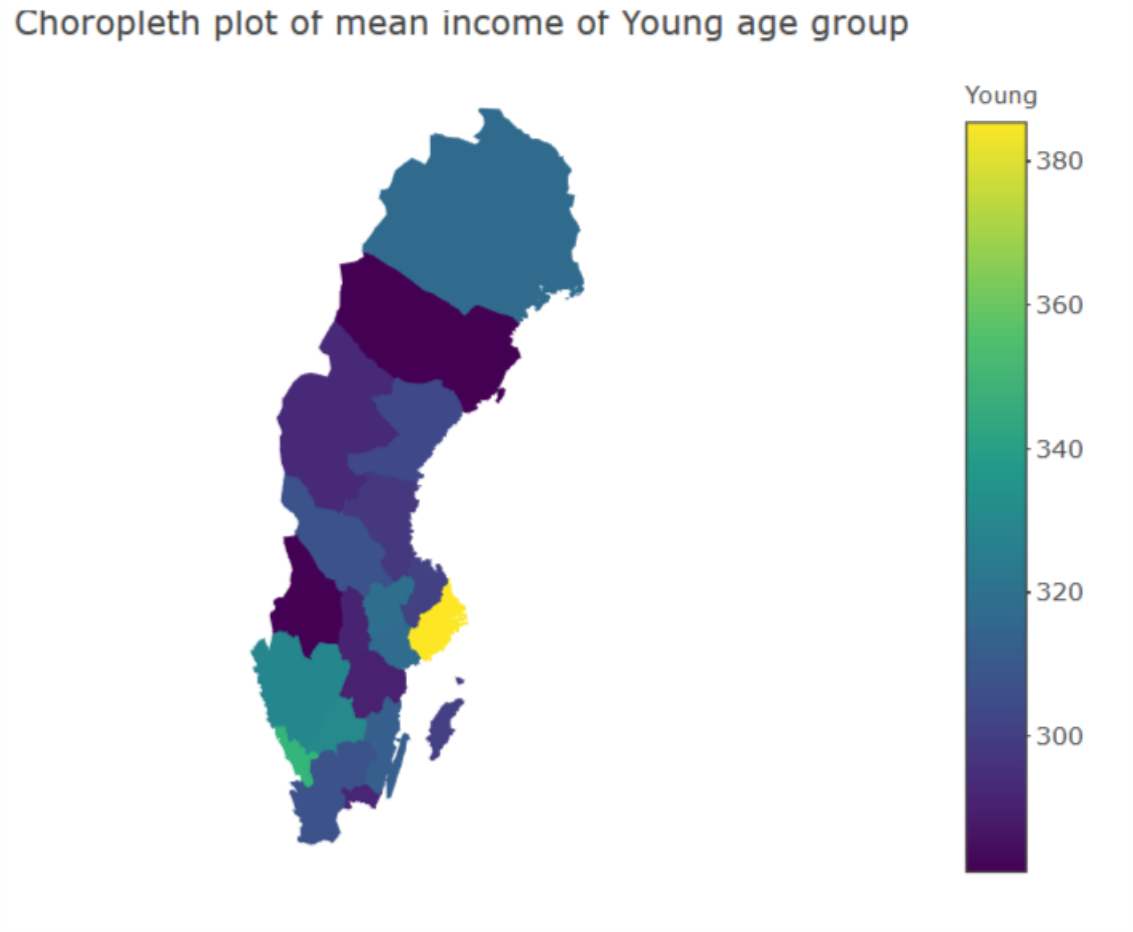
Choropleth plot with Conic Area Projection and log

```
x <- plot_geo(country_aggregate) %>%
  add_trace(
    z = ~log(Count) ,
    text = ~paste(COUNTRY, "\n Count: ", Count), locations = ~COUNTRY_ID
  ) %>%
  layout(
    title = "Choropleth plot of number of mosquitoes",
    geo = list(projection = list(type = "conic equal area")))

knitr::include_graphics('./3.8.3.png')
```



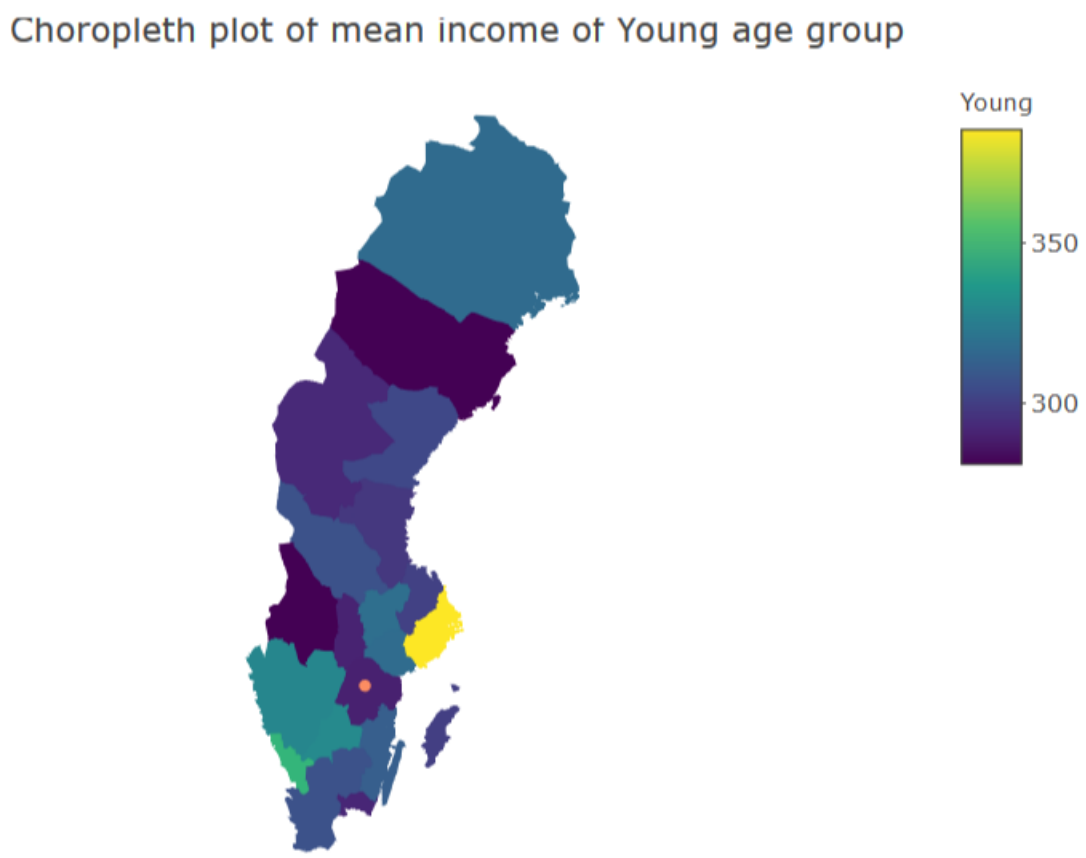
Choropleth plot using custom shape files(sf file)

```
swe_data = read.csv("000000KD.csv")

swe_data_processed = data.frame(region = unique(swe_data$region))
swe_data_split = split(swe_data, swe_data$age)
for (i in seq_along(swe_data_split)) {
  swe_data_processed[[names(swe_data_split)[i]]] = merge(swe_data_split[[i]],
                                                         swe_data_processed$region,
                                                         by.x = 'region',
                                                         by.y = 1, all = T)$X2016
}

colnames(swe_data_processed) = c("region", "Young", "Adult", "Senior")
swe_data_processed$region = gsub(" county", "", swe_data_processed$region)
swe_data_processed$region = gsub("\\d{2} ", "", swe_data_processed$region)
swe_data_processed$region = gsub("Örebro", "Orebro", swe_data_processed$region)
rownames(swe_data_processed) = swe_data_processed$region
rds = readRDS('gadm36_SWE_1_sf.rds')

rds$Young = swe_data_processed[rds$NAME_1, "Young"]

x <- plot_ly() %>% add_sf(data = rds, split = ~NAME_1,
                      color = ~Young, showlegend = F, alpha = 1) %>%
  layout(title = "Choropleth plot of mean income of Young age group")

knitr::include_graphics('./3.8.4.png')
```



Choropleth plot with a custom maker

```
rds$Young = swe_data_processed[rds$NAME_1, "Young"]

x <- plot_ly() %>% add_sf(data = rds, split = ~NAME_1, color = ~Young,
                          showlegend = F, alpha = 1) %>%
  add_markers(x = 15.621373, y = 58.410809, color = "red",
              hoverinfo = "text", text = "We are here!") %>%
  layout(title = "Choropleth plot of mean income of Young age group")

knitr::include_graphics('./3.8.5.png')
```



Violin Plots

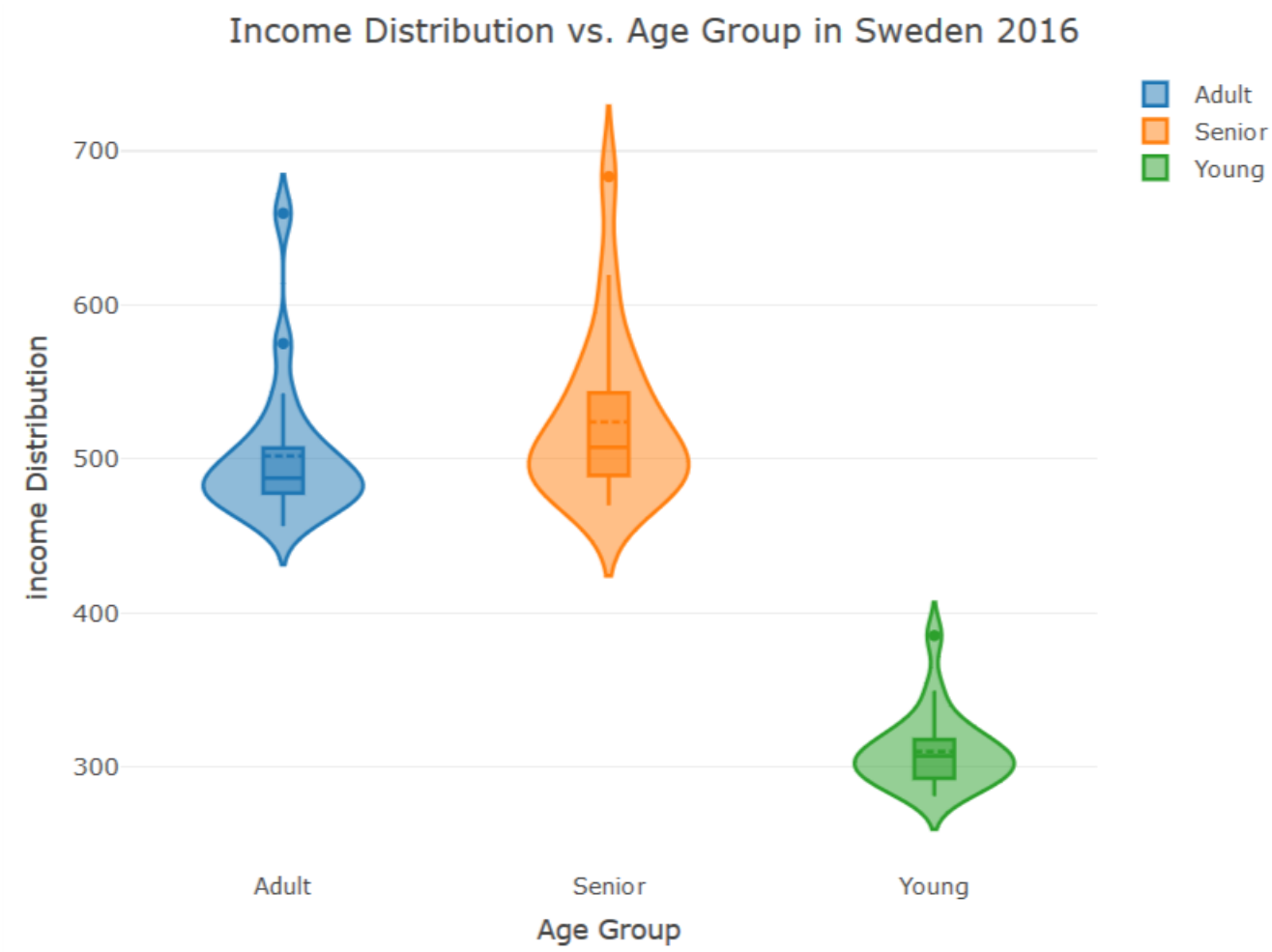
Violin Plot Simple 2 variable (one cateogrical and one numeric)

```
swe_data_processed = data.frame(region = unique(swe_data$region))

swe_data_split = split(swe_data, swe_data$age)
for (i in seq_along(swe_data_split)) {
  swe_data_processed[[names(swe_data_split)[i]]] = merge(swe_data_split[[i]],
                                                         swe_data_processed$region,
                                                         by.x = 'region',
                                                         by.y = 1, all = T)$X2016
}

colnames(swe_data_processed) = c("region", "Young", "Adult", "Senior")
swe_data_processed$region = gsub(" county", "", swe_data_processed$region)
swe_data_processed$region = gsub("\\d{2} ", "", swe_data_processed$region)
swe_data_processed$region = gsub("Örebro", "Orebro", swe_data_processed$region)
rownames(swe_data_processed) = swe_data_processed$region

swe_data$age_group <- ifelse(swe_data$age == "18-29 years", "Young", ifelse(swe_data$age == "30-49 years
```

```
ggplot(swe_data) +
  geom_violin(aes(x = age, y = X2016, fill = age_group),
              draw_quantiles = c(0.25, 0.5, 0.75)) +
  scale_x_discrete(labels=c("18_29" = "Young",
                            "30_49" = "Adult",
                            "50_69" = "Senior"),
                   name = "Age Group") +
  ylab("Mean Income (SEK thousands)") +
  ggtitle("Distribution of mean incomes in different age groups")
```

Violing using plotly

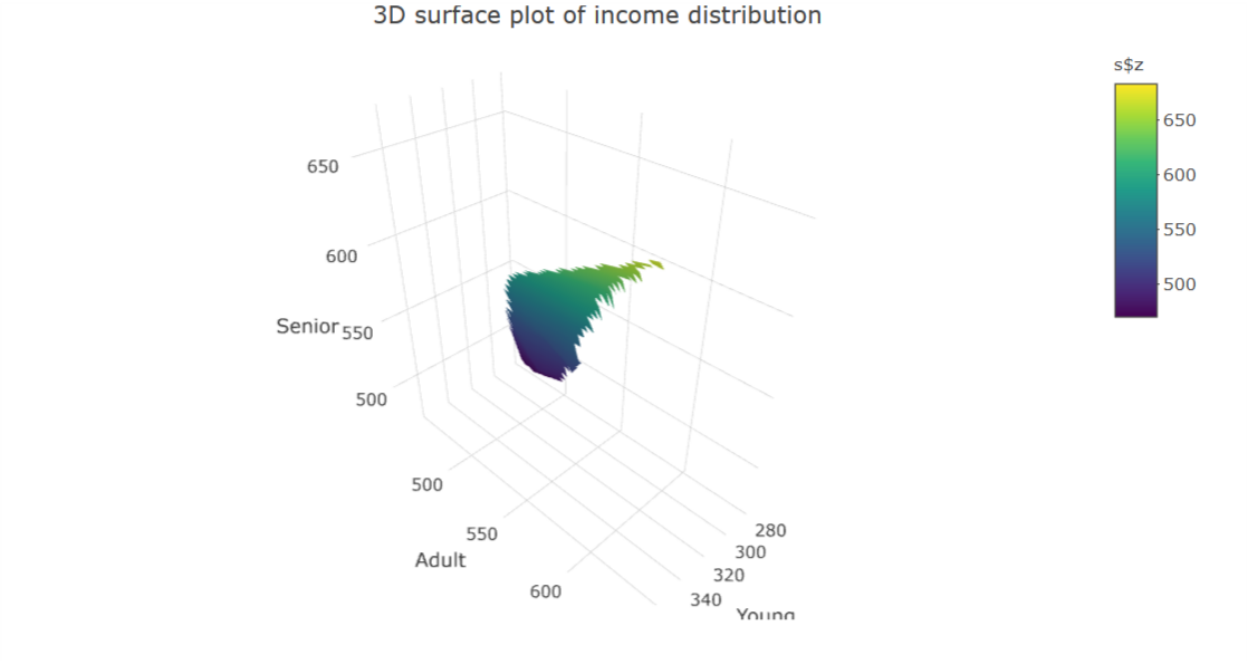
```
income_data_recasted <- dcast(income_data, region~age, value.var = "X2016")
setnames(income_data_recasted, old=c("18-29 years", "30-49 years", "50-64 years" ), new=c("Young", "Adul

income_data$age_group <- ifelse(income_data$age == "18-29 years", "Young",
                                ifelse(income_data$age == "30-49 years", "Adult","Senior"))

x <- income_data %>% plot_ly(x =~age_group ,y = ~X2016, type = 'violin', split = ~age_group, box = list

knitr::include_graphics('./3.9.1.png')
```



3D surface plots

3D surface plots using plotly

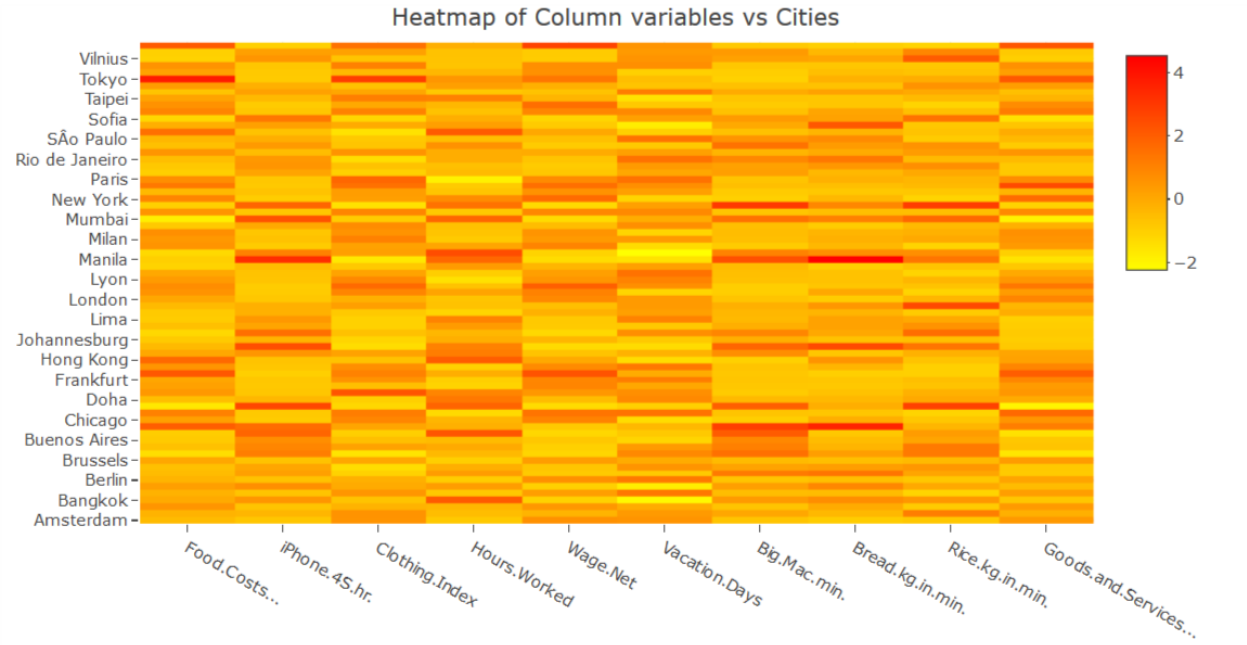
```
s = interp(swe_data_processed$Young, swe_data_processed$Adult,
           swe_data_processed$Senior, duplicate = "mean")

x <- plot_ly(x=~s$x, y=~s$y, z=~s$z, type="surface") %>% layout(
  scene=list(
  xaxis = list(title = "Young"),
  yaxis = list(title = "Adult"),
  zaxis = list(title = "Senior")),
  title = "3D surface plot of income distribution")

knitr::include_graphics('./3.10.1.png')
```



Heat Map

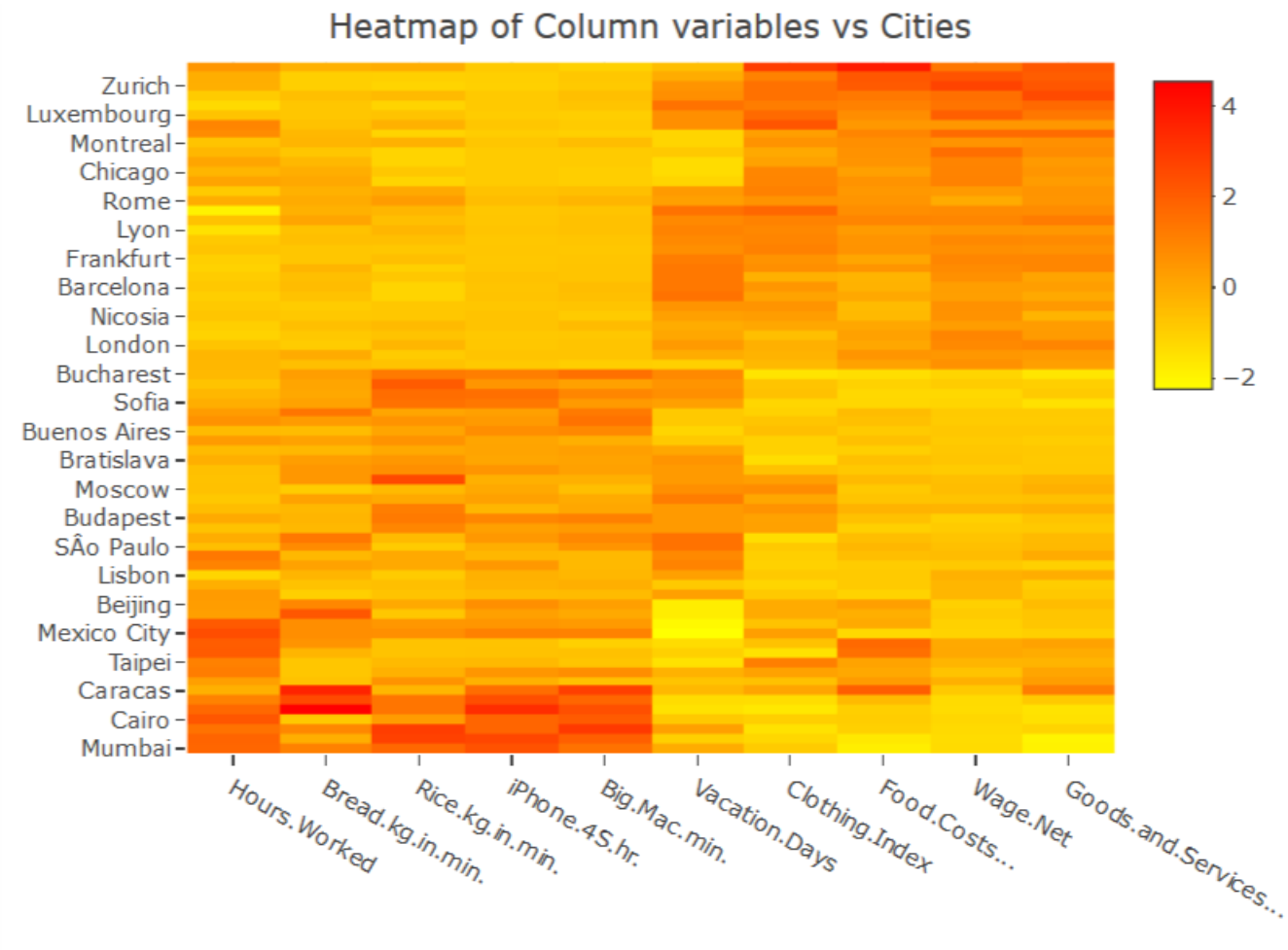
Heat Map without reordering

```
ubs_scaled = scale(ubs_data)

x <- plot_ly(x = colnames(ubs_scaled), y = rownames(ubs_scaled),
        z = ubs_scaled, type = "heatmap",
        colors = colorRamp(c("yellow", "red"))) %>%
  layout(title = "Heatmap of Column variables vs Cities")

knitr::include_graphics('./3.11.1.png')
```



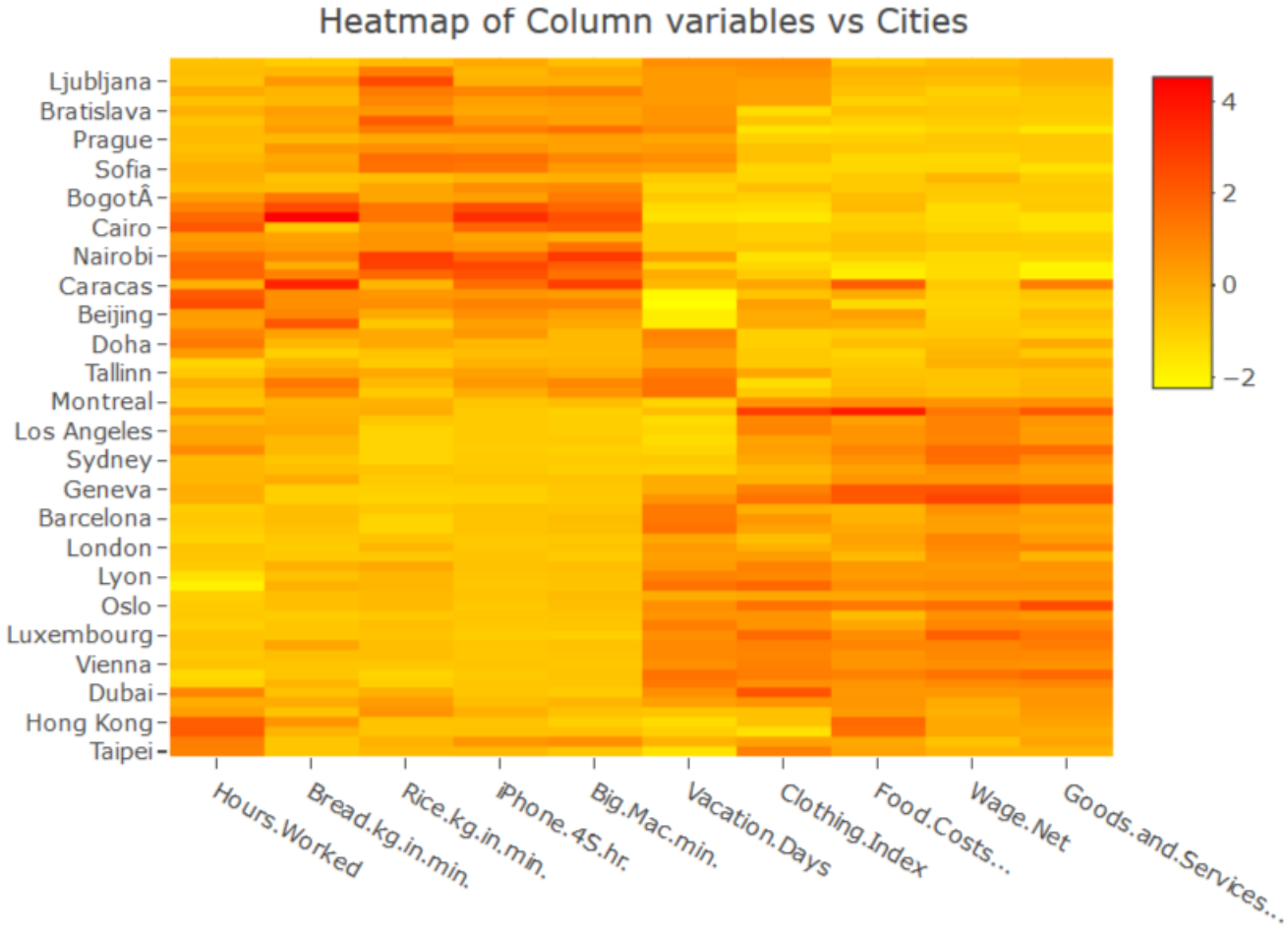
Heat Map with ordering using Hiearchical Clustering (HC) using Euclidean distance

```
ubs_scaled = scale(ubs_data)
row_dist_euc = dist(ubs_scaled, method = "euclidean")
col_dist_euc = dist(t(ubs_scaled), method = "euclidean")

seriate_row_euc = seriate(row_dist_euc, "HC")
seriate_col_euc = seriate(col_dist_euc, "HC")
ord_row_euc = get_order(seriate_row_euc)
ord_col_euc = get_order(seriate_col_euc)

ubs_reordered_euc = ubs_scaled[rev(ord_row_euc),ord_col_euc]

x <- plot_ly(x = colnames(ubs_reordered_euc),
        y = rownames(ubs_reordered_euc),
        z = ubs_reordered_euc, type = "heatmap",
        colors = colorRamp(c("yellow", "red"))) %>%
  layout(title = "Heatmap of Column variables vs Cities")

knitr::include_graphics('./3.11.2.png')
```



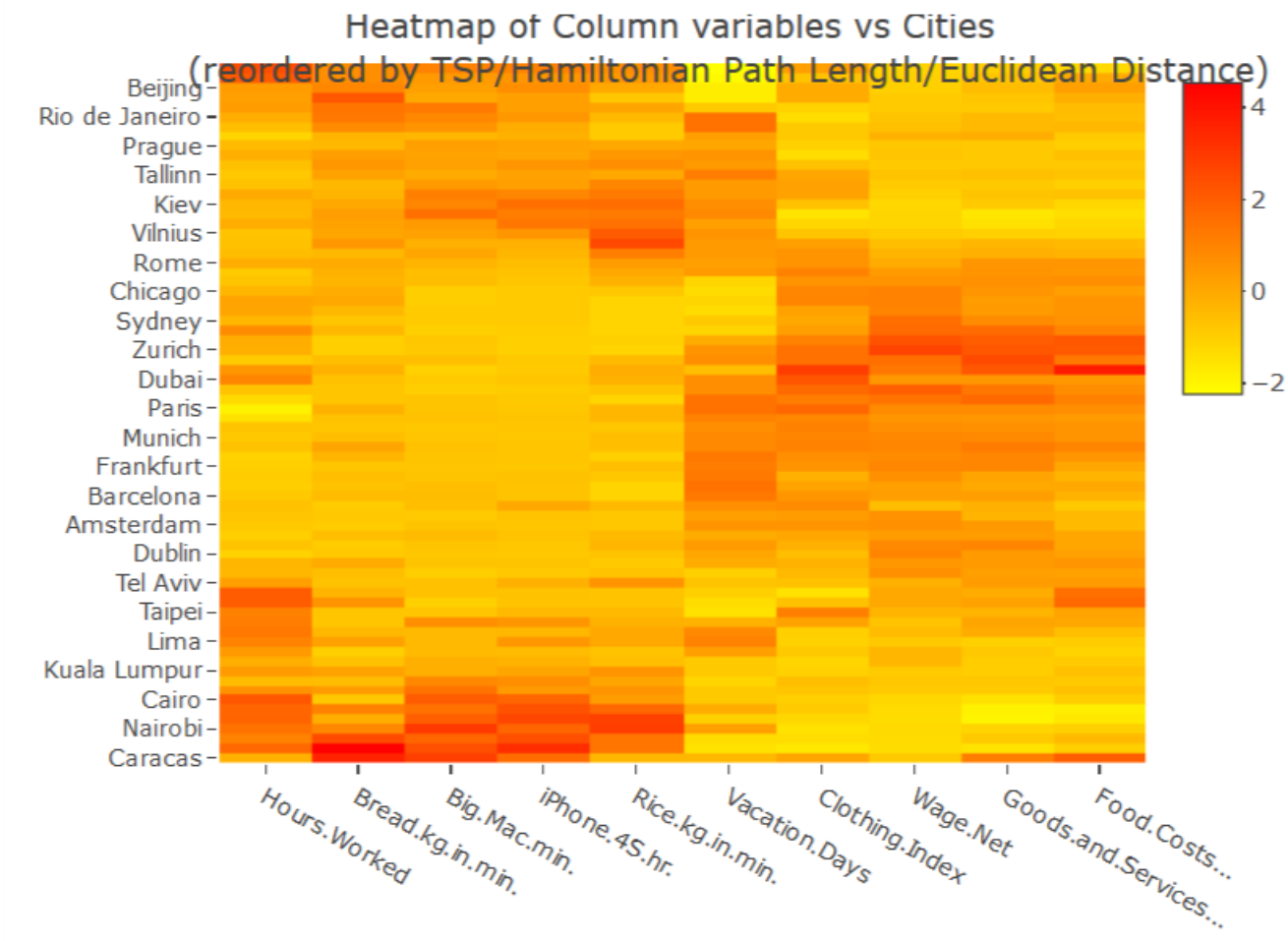
Heat Map with ordering using Corellation using Euclidean distance

```
row_dist_cor = as.dist(1 - cor(t(ubs_scaled)))
col_dist_cor = as.dist(1 - cor(ubs_scaled))

seriate_row_cor = seriate(row_dist_cor, method = "HC")
seriate_col_cor = seriate(col_dist_cor, method = "HC")
ord_row_cor = get_order(seriate_row_cor)
ord_col_cor = get_order(seriate_col_cor)

ubs_reordered_cor = ubs_scaled[rev(ord_row_cor), ord_col_cor]

x <- plot_ly(x = colnames(ubs_reordered_cor),
        y = rownames(ubs_reordered_cor),
        z = ubs_reordered_cor, type = "heatmap",
        colors = colorRamp(c("yellow", "red"))) %>%
  layout(title = "Heatmap of Column variables vs Cities")

knitr::include_graphics('./3.11.3.png')
```



Heat Map with ordering using Hamiltonian Path Length using Traveling Salesman Problem (TSP)

```
seriate_row_tsp = seriate(row_dist_euc, "TSP")
seriate_col_tsp = seriate(col_dist_euc, "TSP")
ord_row_tsp = get_order(seriate_row_tsp)
ord_col_tsp = get_order(seriate_col_tsp)

ubs_reordered_tsp = ubs_scaled[rev(ord_row_tsp), ord_col_tsp]

x <- plot_ly(x = colnames(ubs_reordered_tsp),
        y = rownames(ubs_reordered_tsp),
        z = ubs_reordered_tsp, type="heatmap",
        colors = colorRamp(c("yellow", "red"))) %>%
  layout(title = "Heatmap of Column variables vs Cities
         (reordered by TSP/Hamiltonian Path Length/Euclidean Distance)")

knitr::include_graphics('./3.11.4.png')
```



HeatMap QC and cluster information

```
library(heatmaply)
```

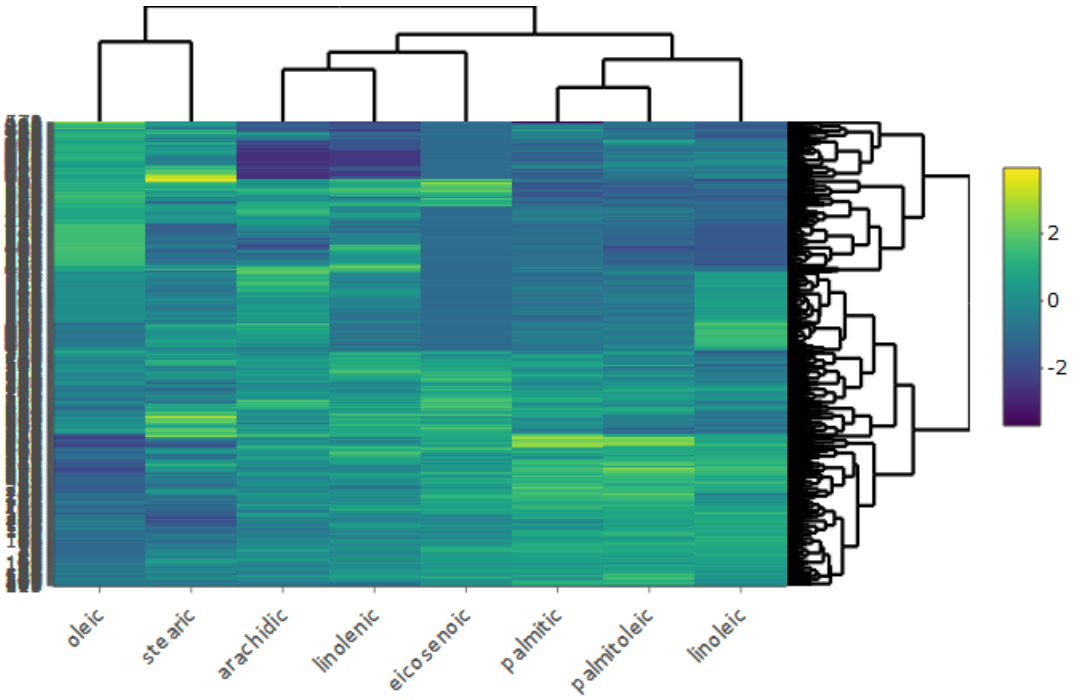
```
## Loading required package: viridis
## Loading required package: viridisLite
##
## Attaching package: 'viridis'
## The following object is masked from 'package:scales':
##
##     viridis_pal
##
## ======================
## Welcome to heatmaply version 0.15.2
##
## Type citation('heatmaply') for how to cite the package.
## Type ?heatmaply for the main documentation.
##
## The github page is: https://github.com/talgalili/heatmaply/
```

```
## Please submit your suggestions and bug-reports at: https://github.com/talgalili/heatmaply/issues
## Or contact: <tal.galili@gmail.com>
## ======================
```

```
##
## Attaching package: 'heatmaply'
```

```
## The following object is masked from 'package:igraph':
##
##     normalize
```

```
x <- heatmaply(ubs_reordered_tsp)
knitr::include_graphics('./heatmap_qc.png')
```



Comparison of two solvers for heatmap

```
#Hamiltonian Path Length
ord_tsp = seriate(row_dist_euc, "TSP")
ham_tsp = criterion(row_dist_euc, order = ord_row_tsp, "Path_length")
paste("Hamiltonian Path Length : ", ham_tsp)
```

```
## [1] "Hamiltonian Path Length :  123.487167919793"
```

```
#Gradient Measure
gm_tsp = criterion(row_dist_euc, order=ord_tsp, "Gradient_raw")
paste("Gradient Measure : ", gm_tsp)
```

```
## [1] "Gradient Measure :  40548"
```

```
#Hamiltonian Path Length
ord_hc = seriate(row_dist_euc, "HC")
ham_hc = criterion(row_dist_euc, order = ord_hc, "Path_length")
```

```r
paste("Hamiltonian Path Length : ", ham_hc)
```

```
## [1] "Hamiltonian Path Length :  144.492476381981"
```

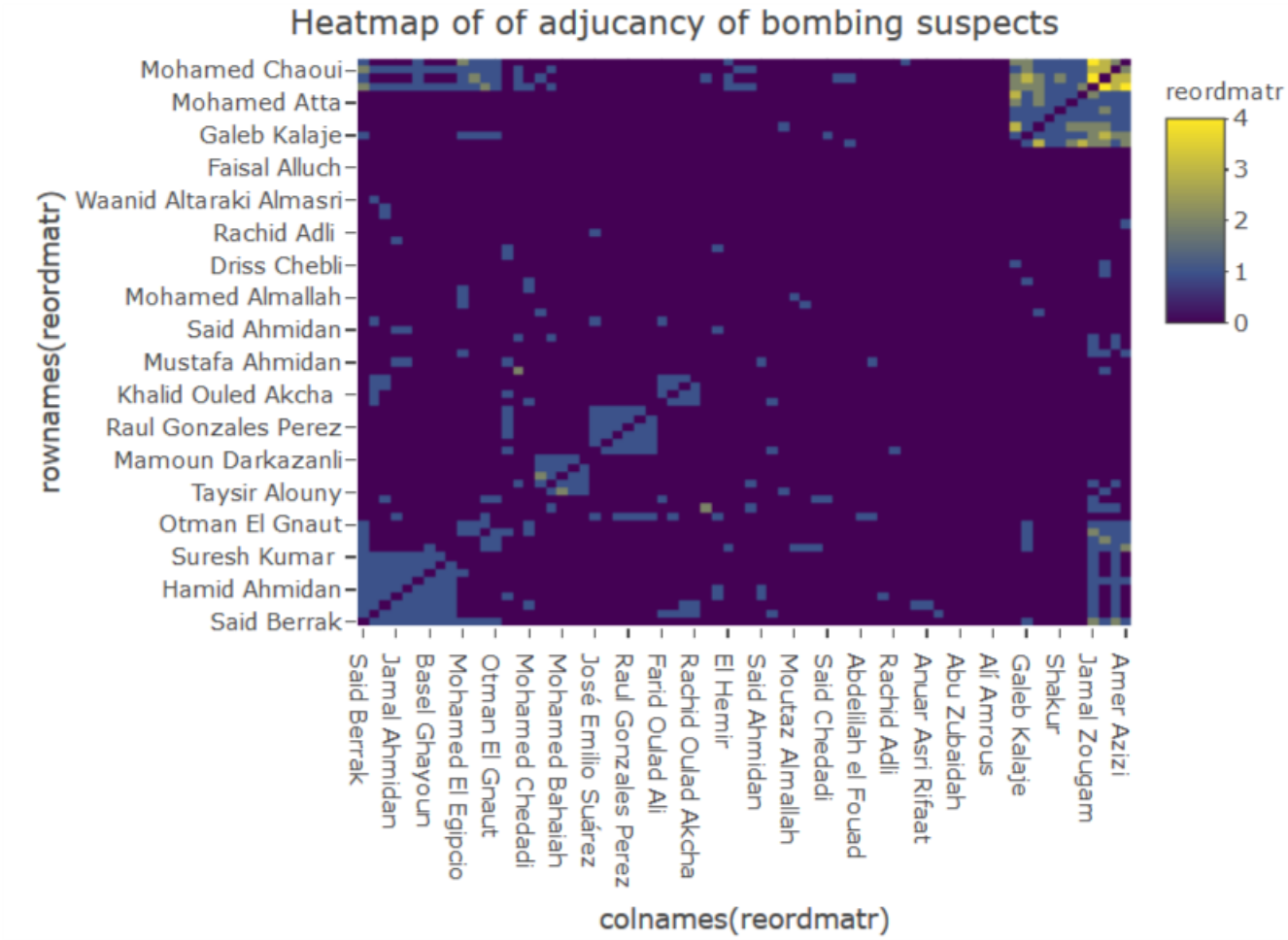
```r
#Gradient Measure
gm_hc = criterion(row_dist_euc, order = ord_hc, "Gradient_raw")
paste("Gradient Measure : ", gm_hc)
```

```
## [1] "Gradient Measure :  58432"
```

Heat Map using Adjacency

```r
colnames(links) <- c("from","to","strength")
links <- links[order(links$from, links$to),]
nodes$id <- 1:70
rownames(links) <- NULL

colnames(nodes)[2] <- "BombingGrp"

#Size of links based on "strength of links"
links$width <- links$strength*3

#Nodes colored based on Bombing Group
nodes$label <- nodes$V1
nodes$group <- nodes$BombingGrp

#Size of nodes proportional to the number of connections
graph <- graph.data.frame(links, directed = F)
strength_value <- strength(graph)
nodes$value <- strength_value[match(nodes$id, names(strength_value))]


for(i in 1:nrow(links)){
  links$from_name[i] <- nodes$V1[links$from[i]]
  links$to_name[i] <- nodes$V1[links$to[i]]
}

links <- links[,c("from_name","to_name","from","to","strength","width")]
nodes1 <- nodes
net <- graph_from_data_frame(d=links, vertices=nodes, directed=F)
ceb <- cluster_edge_betweenness(net)
nodes1$group <- ceb$membership


netm <- get.adjacency(net, attr="strength", sparse=F)
colnames(netm) <- V(net)$name
rownames(netm) <- V(net)$name

rowdist<-dist(netm)

order1<-seriate(rowdist, "HC")
ord1<-get_order(order1)
```

```
reordmatr<-netm[ord1,ord1]

x <- plot_ly(z=~reordmatr, x=~colnames(reordmatr),
        y=~rownames(reordmatr), type="heatmap") %>%
layout(title = "Heatmap of of adjucancy of bombing suspects")

knitr::include_graphics('./3.11.6.png')
```

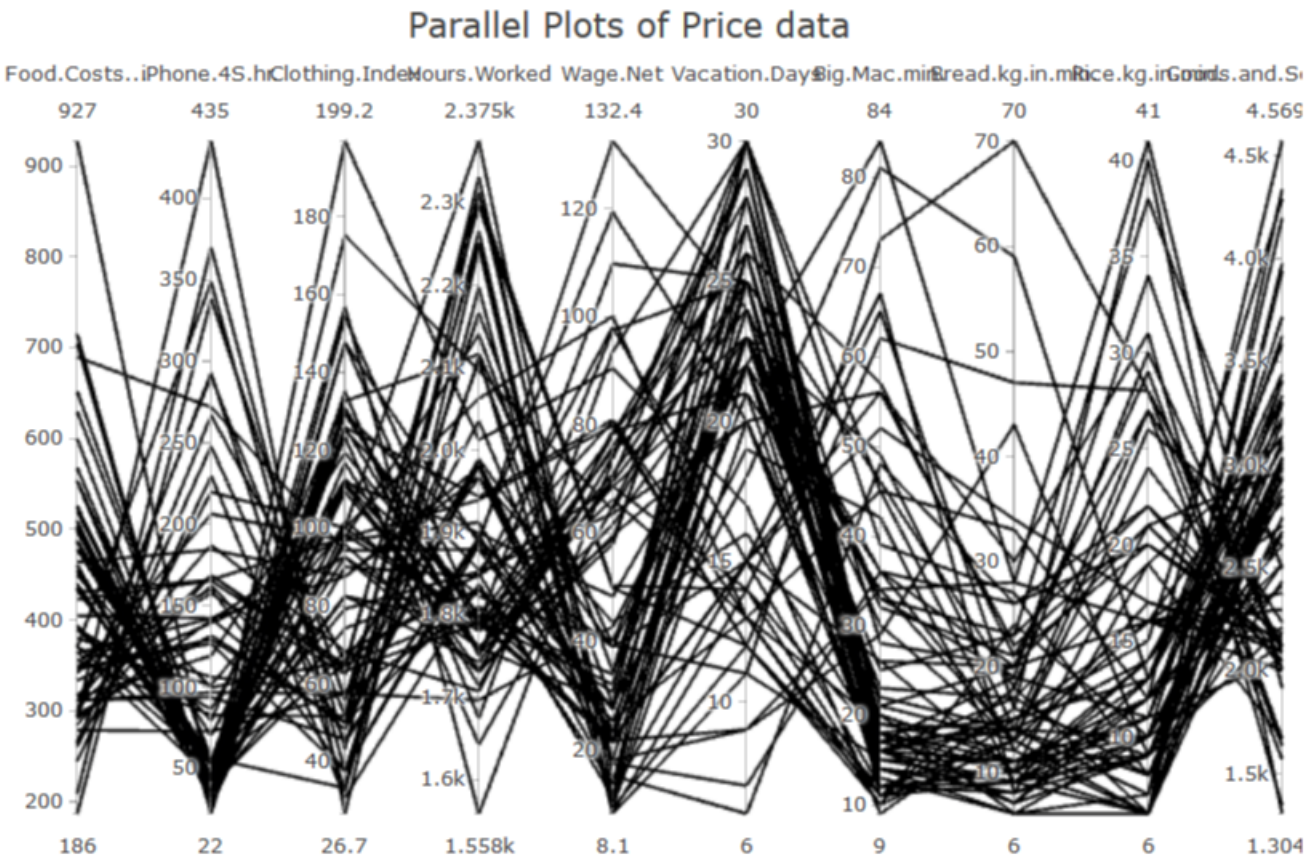


Parallel Plot

Parallel Plot using Plotly

```
# Function to create dimension list based on given variable order
get_dimension_list <- function(df, col_order){
  dim_list = list()
  i = 1
  for (col in col_order){
    dim_list[[i]] = list(label = col, values = df[[col]])
    i = i + 1
  }
```

```
  return(dim_list)
}

# Create dimension list for unordered data
unord_dim_list = get_dimension_list(ubs_data, colnames(ubs_data))

# Create dimension list for reordered data
var_order = c("Cloting.Index", "Wage.Net", "Goods.and.Services...", "Food.Costs...",
              "Vacation.Days", "iPhone.4S.hr.", "Big.Mac.min.",
              "Bread.kg.in.min.", "Rice.kg.in.min.", "Hours.Worked")

reord_dim_list = get_dimension_list(ubs_data, var_order)

# Brush different clusters with different colors
ubs_data$cluster = 1
ubs_data[ubs_data$Wage.Net < 24 , "cluster"] = 2

x <- plot_ly(data = ubs_data, type = 'parcoords',
        dimensions = unord_dim_list) %>%
  layout(title = "Parallel Plots of Price data")

knitr::include_graphics('./3.12.1.png')
```



Parallel Plot using Plotly and highlight one line

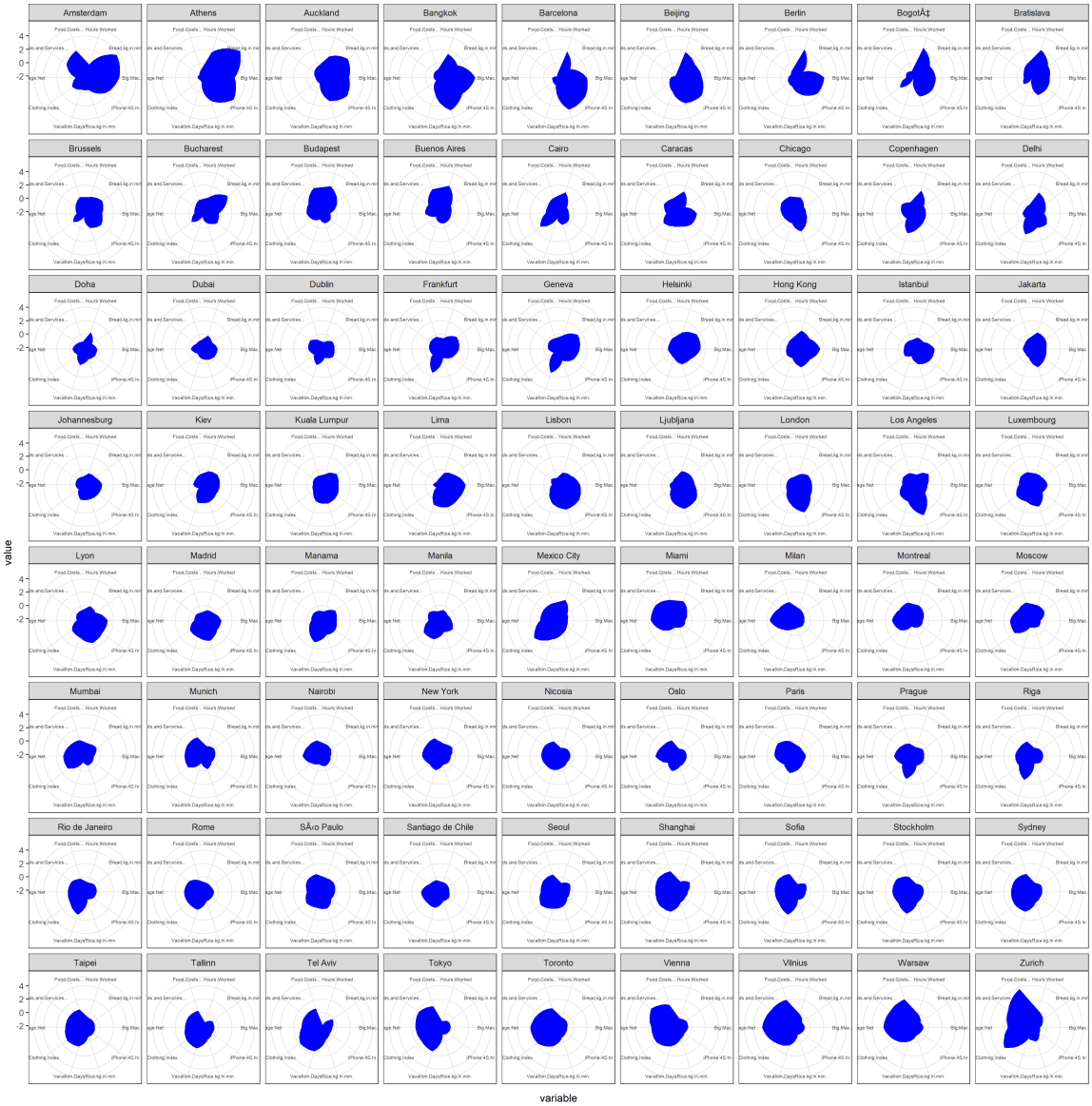
```
x <- plot_ly(data = ubs_data, type = 'parcoords',
        line = list(color = ~cluster, colorscale = list(c(0,"red"), c(1,"blue"))),
        dimensions = reord_dim_list) %>%
  layout(title = "Parallel Plot with one highlight")

knitr::include_graphics('./3.12.2.png')
```

Radar Plots

Radar Plots using Scatterpolar

```
price_data_scale <- scale(price_data[,-1])

price_row_dist <- dist(x=price_data_scale, method = "euclidean", diag = TRUE)
price_col_dist <- dist(x=t(price_data_scale), method = "euclidean", diag = TRUE)


order1 <- seriate(price_row_dist, "OLO")
order2 <- seriate(price_col_dist, "OLO")
ord1 <- get_order(order1)
```

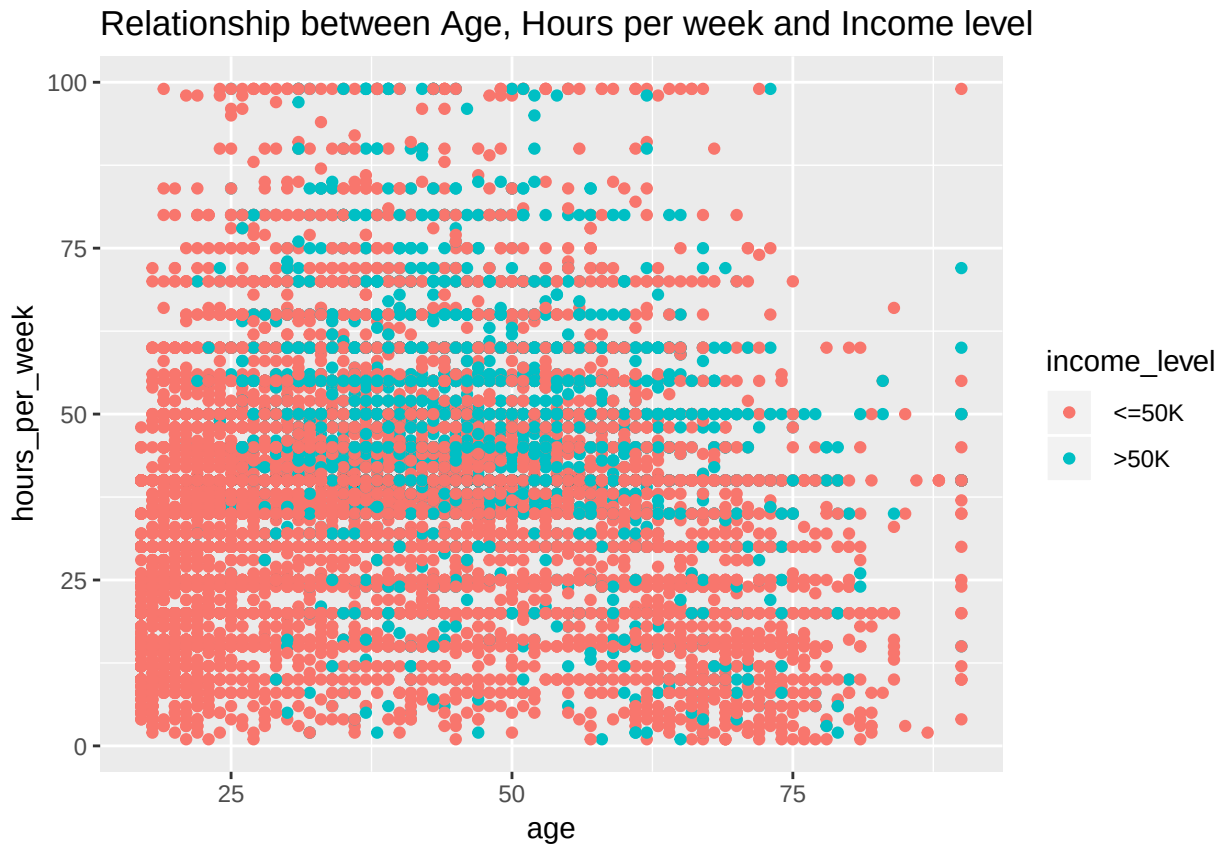
```r
ord2 <- get_order(order2)

reordmatr <- price_data_scale[rev(ord1),ord2]
reordmatr <- as.data.frame(reordmatr)
reordmatr$City = price_data$City

reordmatr_transformed <- reordmatr%>%tidyr::gather(variable, value, -City, factor_key=T)%>%arrange(City)

radar_plot <- reordmatr_transformed %>% ggplot(aes(x=variable, y=value, group=City)) + geom_polygon(fill

ggsave("radar_plot.png", width = 40, height = 60, units = "cm")

knitr::include_graphics('./radar_plot.png')
```


Amsterdam
Athens
Auckland
Bangkok
Barcelona
Beijing
Berlin
BogotÃ‡
Bratislava
Brussels
Bucharest
Budapest
Buenos Aires
Cairo
Caracas
Chicago
Copenhagen
Delhi
Doha
Dubai
Dublin
Frankfurt
Geneva
Helsinki
Hong Kong
Istanbul
Jakarta
Johannesburg
Kiev
Kuala Lumpur
Lima
Lisbon
Ljubljana
London
Los Angeles
Luxembourg
Lyon
Madrid
Manama
Manila
Mexico City
Miami
Milan
Montreal
Moscow
Mumbai
Munich
Nairobi
New York
Nicosia
Oslo
Paris
Prague
Riga
Rio de Janeiro
Rome
SÃ£o Paulo
Santiago de Chile
Seoul
Shanghai
Sofia
Stockholm
Sydney
Taipei
Tallinn
Tel Aviv
Tokyo
Toronto
Vienna
Vilnius
Warsaw
Zurich
value
variable

Trellis Plots

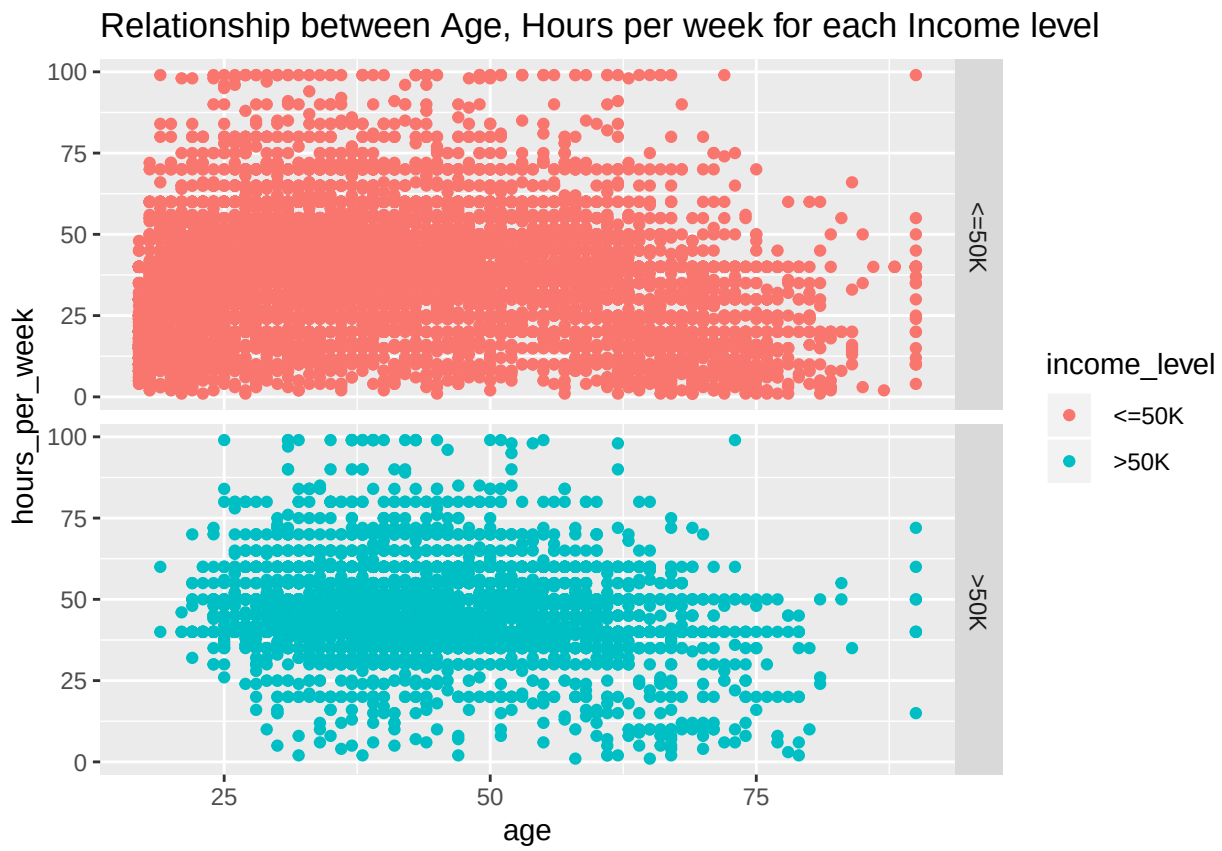
Trellis Plots using ggplot2

```
ggplot(adult_data) +
  geom_point(aes(x = age, y = hours_per_week, color = income_level)) +
  ggtitle("Relationship between Age, Hours per week and Income level")
```



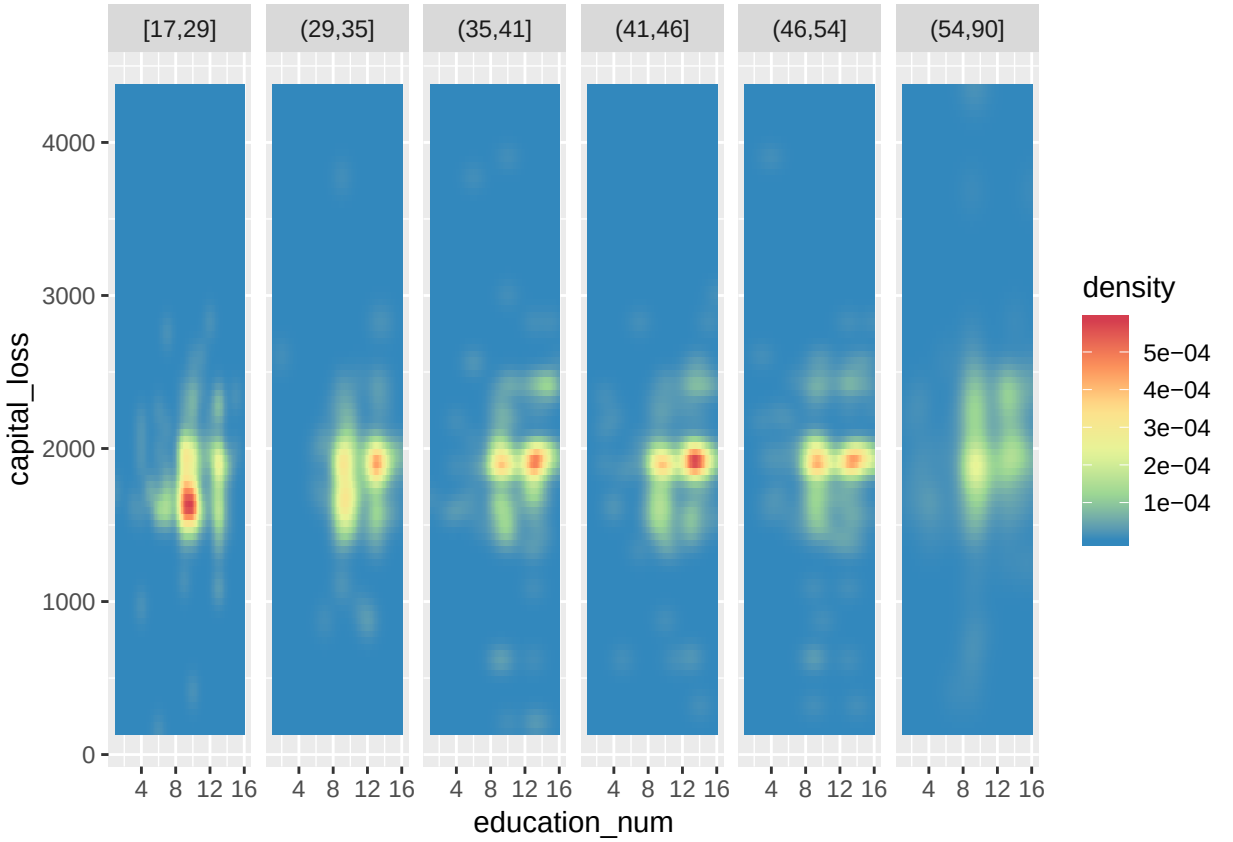
Trellis Plots with facet/condition on a variable

```
ggplot(adult_data) +
  geom_point(aes(x = age, y = hours_per_week, color = income_level)) +
  facet_grid(income_level~.) +
  ggtitle("Relationship between Age, Hours per week for each Income level")
```



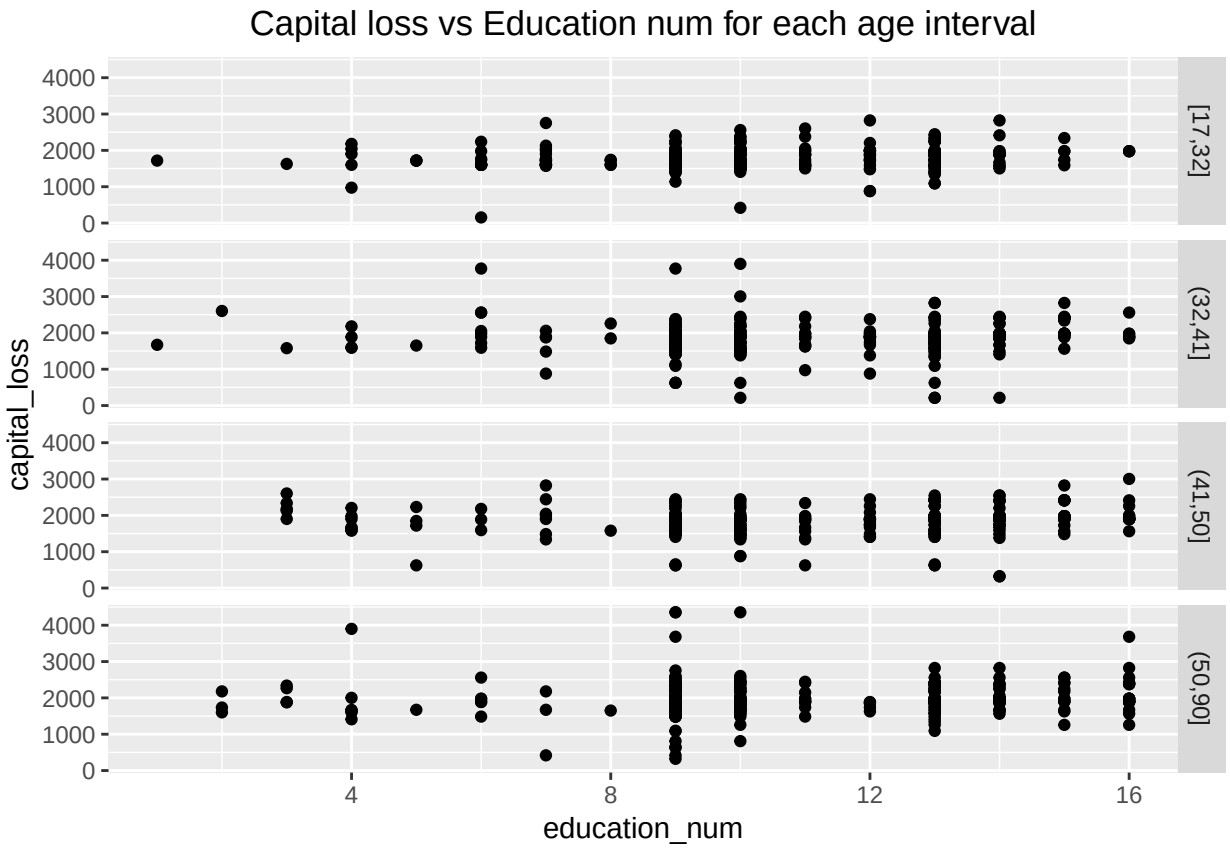
Trellis Plot with raster-type-2d-density plot

```
capital_loss_data = adult_data[adult_data$capital_loss != 0,]

ggplot(capital_loss_data) +
  stat_density_2d(geom = "raster",
                  aes(x = education_num, y = capital_loss,
                      fill = stat(density)), contour = FALSE) +
  scale_fill_distiller(palette = "Spectral") +
  facet_grid(cols = vars(cut_number(age, 6)))
```



Trellis Plot with raster-type-2d-density plot with discreetization(cut_number)

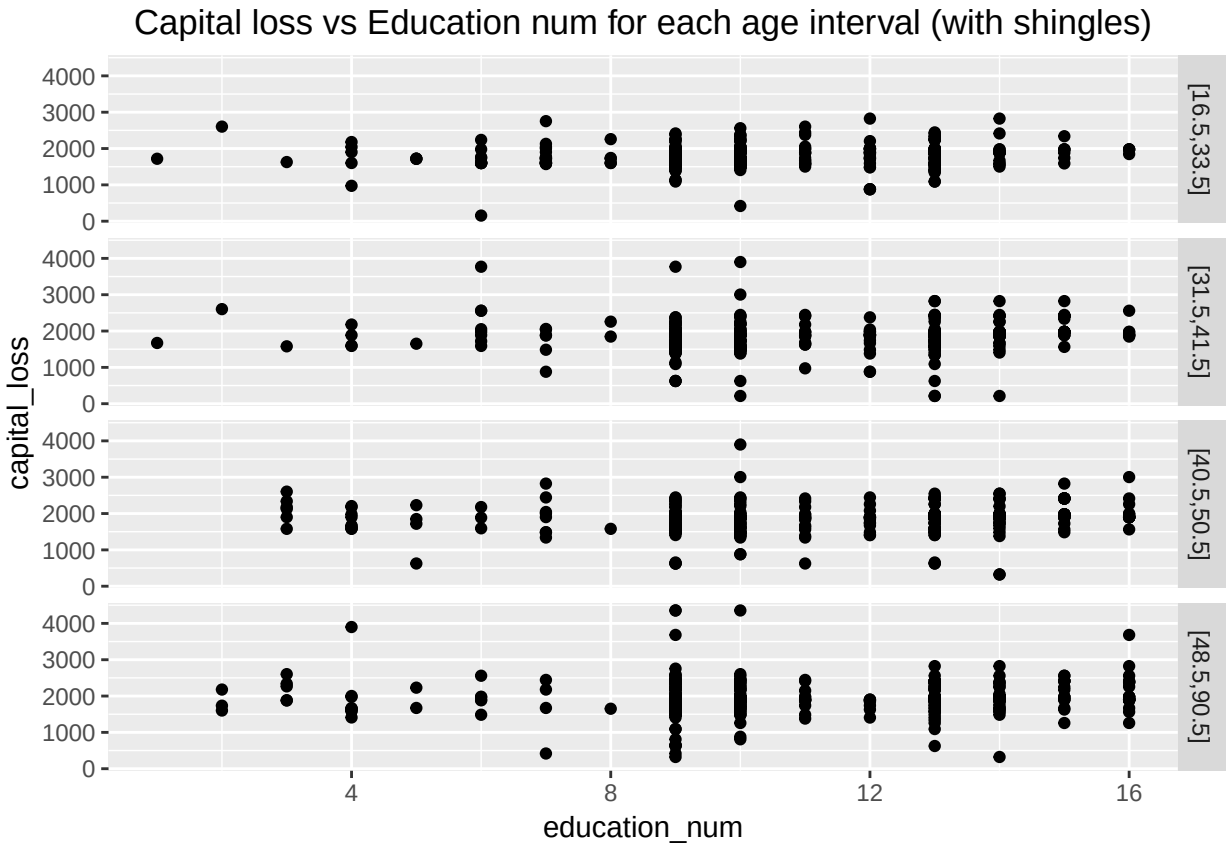
```
ggplot(capital_loss_data) +
  geom_point(aes(x = education_num, y = capital_loss)) +
  facet_grid(vars(cut_number(age, 4))) +
  ggtitle("Capital loss vs Education num for each age interval") +
  theme(plot.title = element_text(hjust = 0.5))
```



Trellis Plot with Shingles

```
age_ranges = lattice::equal.count(capital_loss_data$age, number = 4, overlap = 0.1)

age_range_matrix = matrix(unlist(levels(age_ranges)), ncol=2, byrow = T)
age_range_df = data.frame(Lower = age_range_matrix[,1],
                          Upper = age_range_matrix[,2],
                          Interval = factor(1:nrow(age_range_matrix)))

index = c()
age_interval = c()
for(i in 1:nrow(age_range_df)){
  interval_name = paste("[", age_range_df$Lower[i], ",",
                        age_range_df$Upper[i], "]", sep="")
  indices_in_interval = which(capital_loss_data$age >= age_range_df$Lower[i] &
                              capital_loss_data$age <= age_range_df$Upper[i])
  index = c(index, indices_in_interval)
  age_interval = c(age_interval, rep(interval_name, length(indices_in_interval)))
}

shingles_df = capital_loss_data[index,]
shingles_df = cbind(shingles_df, age_interval)

ggplot(shingles_df) +
```

```
  geom_point(aes(x = education_num, y = capital_loss)) +
  facet_grid(vars(age_interval)) +
  ggtitle("Capital loss vs Education num for each age interval (with shingles)") +
  theme(plot.title = element_text(hjust = 0.5))
```



Word Clouds

Word Clouds using tm

```
set.seed(1)
par(mfrow=c(1,2))
#Word cloud for positive reviews
positive_data$doc_id=1:nrow(positive_data)
colnames(positive_data)[1]<-"text"

#Here we interpret each line in the document as separate document
mycorpus <- Corpus(DataframeSource(positive_data)) #Creating corpus (collection of text data)
mycorpus <- tm_map(mycorpus, removePunctuation)
mycorpus <- tm_map(mycorpus, function(x) removeWords(x, stopwords("english")))
tdm <- TermDocumentMatrix(mycorpus) #Creating term-document matrix
m <- as.matrix(tdm)

#here we merge all rows
v1 <- sort(rowSums(m),decreasing=TRUE) #Sum up the frequencies of each word
```

```
v2 <- v1[-1]

d1 <- data.frame(word = names(v1),freq=v1) #Create one column=names, second=frequences
pal1 <- brewer.pal(8,"Dark2")
pal1 <- pal1[-(1:2)] #Create palette of colors
wordcloud(d1$word,d1$freq, scale=c(5,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=p

d2 <- data.frame(word = names(v2),freq=v2) #Create one column=names, second=frequences
pal2 <- brewer.pal(8,"Dark2")
pal2 <- pal2[-(1:2)] #Create palette of colors
wordcloud(d2$word,d2$freq, scale=c(2,.3),min.freq=2,max.words=100, random.order=F, rot.per=.15, colors=p
```

Linked Plots

Bar chart and Scatter Plot

Bar chart and scatter plot shared using crosstalk

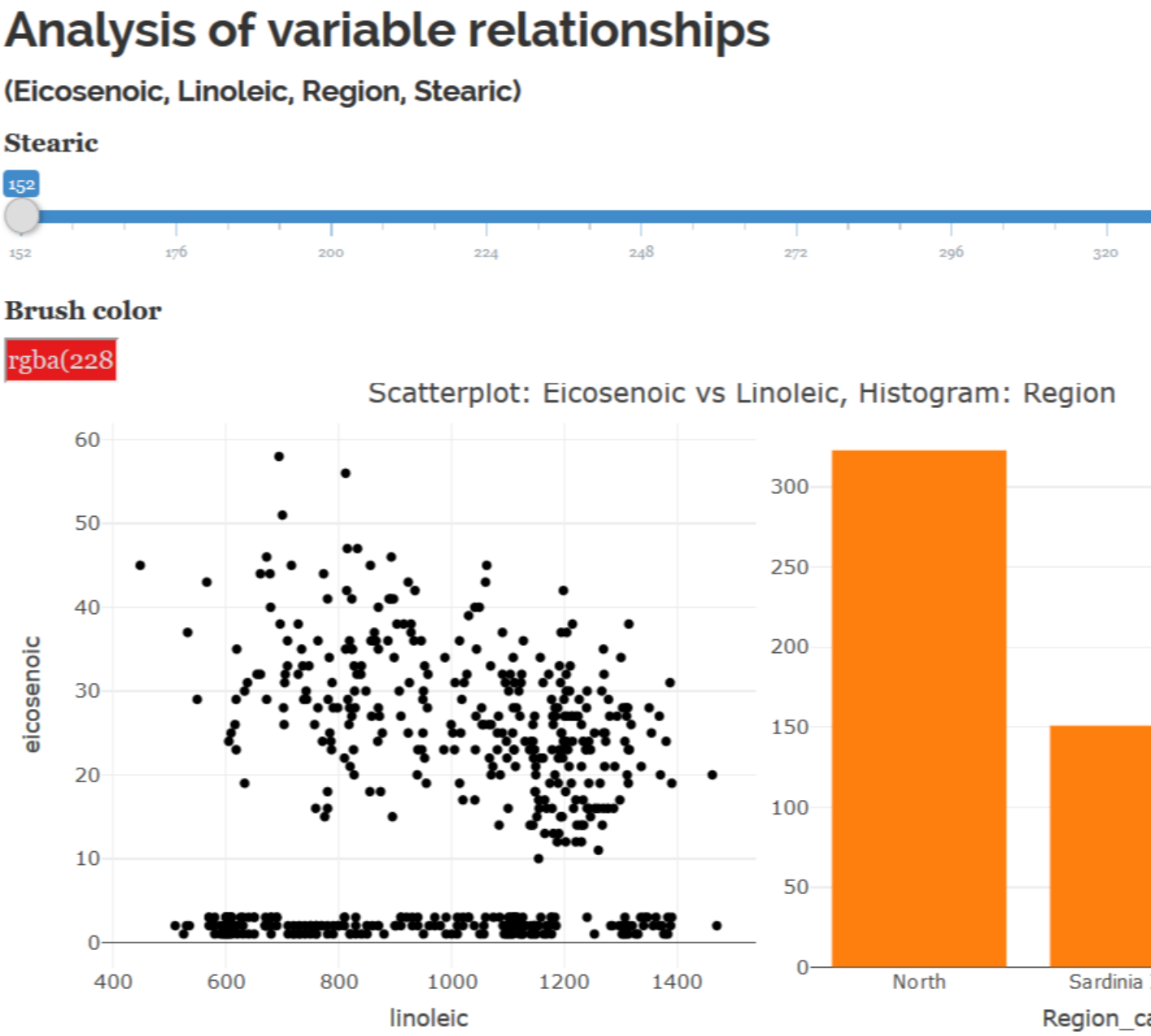
```
olive_data = read.csv("olive.csv")

olive_data$Region_category <- "Sardinia Island"
olive_data[olive_data$Region == 1, c("Region_category")] <- "North"
olive_data[olive_data$Region == 2, c("Region_category")] <- "South"
```

```
ct_olive_2 = SharedData$new(olive_data)

scatter_eico_lino = plot_ly(ct_olive_2, x = ~linoleic, y = ~eicosenoic) %>%
  add_markers(color = I("black"))

bar_region <-plot_ly(ct_olive_2, x = ~Region_category) %>%
  add_histogram() %>% layout(barmode = "overlay")

plots = subplot(scatter_eico_lino, bar_region, titleX = TRUE, titleY = TRUE) %>%
  highlight(on = "plotly_select", dynamic = T, persistent = T, opacityDim = I(1)) %>%
  hide_legend() %>%
  layout(title = "Scatterplot: Eicosenoic vs Linoleic, Histogram: Region")
```

```
## Adding more colors to the selection color palette.
```

```
## We recommend setting `persistent` to `FALSE` (the default) because persistent selection mode can now
```

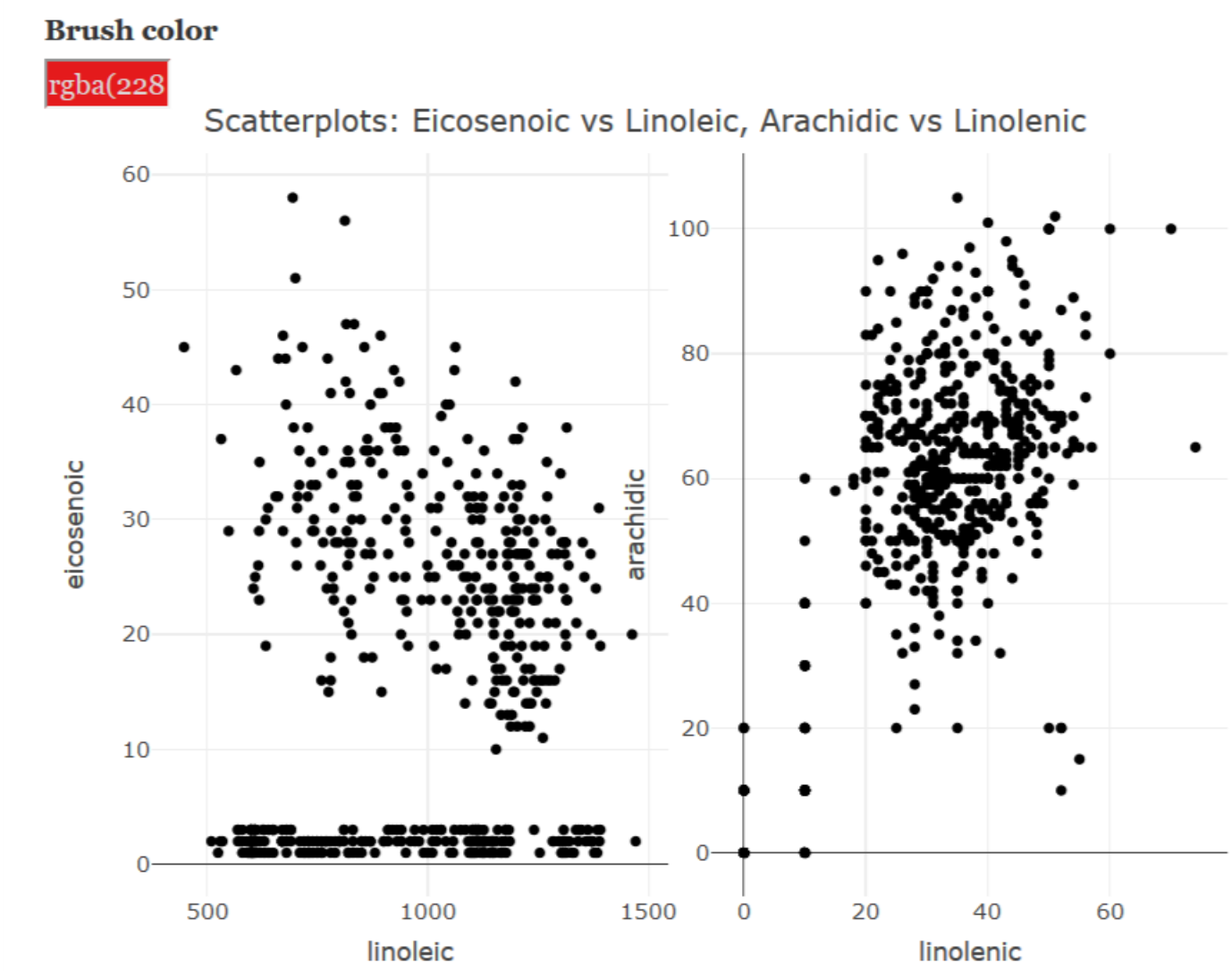
```
x <- bscols(widths = c(12), list(tags$h2("Analysis of variable relationships"),
                                 tags$h4("(Eicosenoic, Linoleic, Region, Stearic)"),
                                 filter_slider("stearic", "Stearic", ct_olive_2, ~stearic), plots))

knitr::include_graphics('./4.1.1.png')
```



Scatter plot shared using crosstalk

```
ct_olive_3 = SharedData$new(olive_data)

sctr_eico_lino = plot_ly(ct_olive_3, x = ~linoleic, y = ~eicosenoic) %>%
  add_markers(color = I("black"))

sctr_arac_lino = plot_ly(ct_olive_3, x = ~linolenic, y = ~arachidic) %>%
  add_markers(color = I("black"))
```

```
x <- subplot(sctr_eico_lino, sctr_arac_lino, titleX = TRUE, titleY = TRUE) %>%
  highlight(on = "plotly_select", dynamic = T, persistent = T, opacityDim = I(1)) %>%
  hide_legend() %>%
  layout(title = "Scatterplots: Eicosenoic vs Linoleic, Arachidic vs Linolenic")

knitr::include_graphics('./4.1.2.png')
```

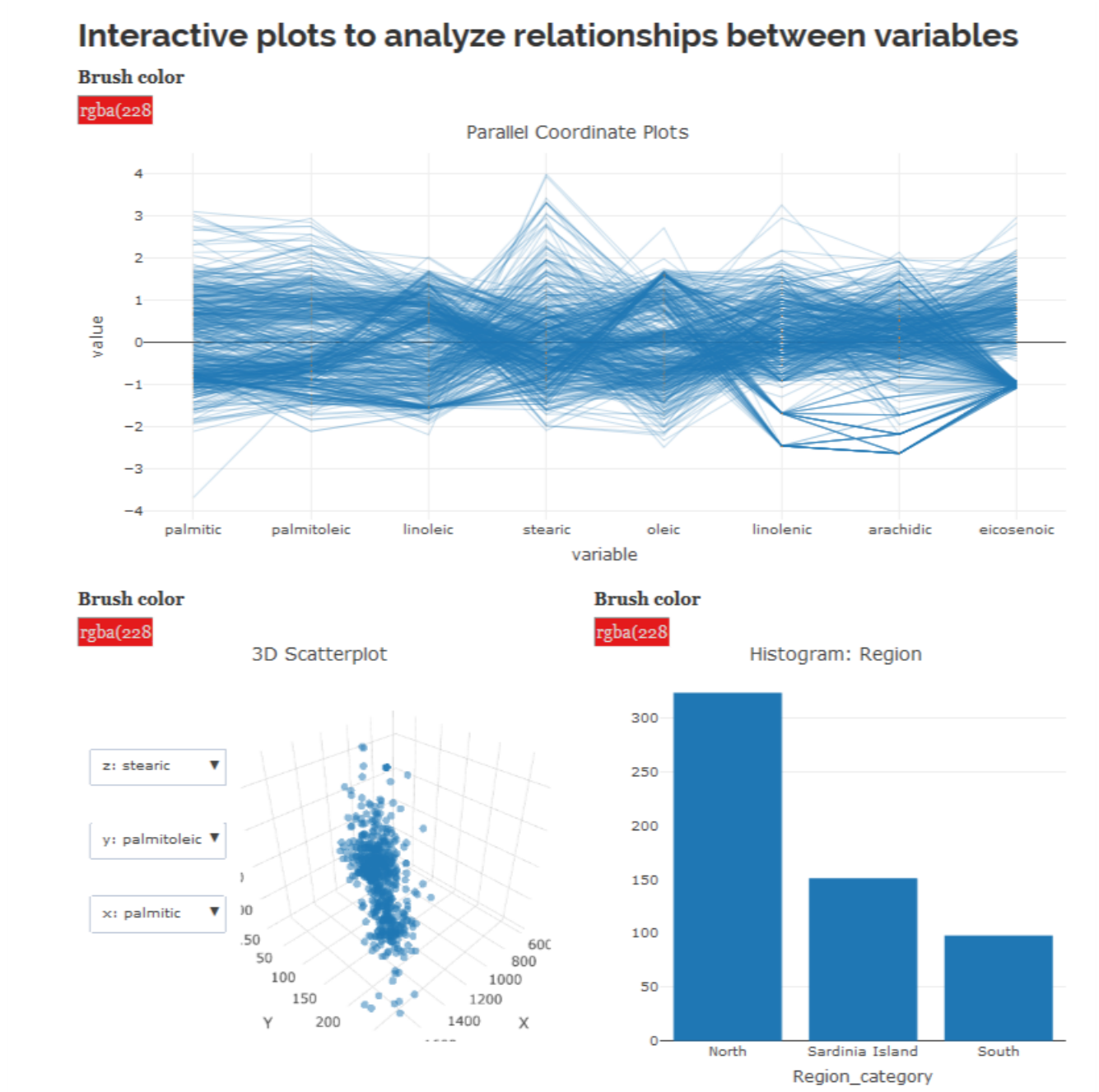


Parallel Chord/Trellis Plot Scatter plot and Bar chart

```
var_order = c("palmitic", "palmitoleic", "linoleic", "stearic",
              "oleic", "linolenic", "arachidic", "eicosenoic")

par_plot = ggparcoord(olive_data[var_order], columns = 1:8)

plot_data = plotly_data(ggplotly(par_plot)) %>% group_by(.ID)
ct_olive_4 = SharedData$new(plot_data, ~.ID, group = "olive")
par_coord_plot = plot_ly(ct_olive_4, x = ~variable, y = ~value) %>%
```

```
  add_lines(line = list(width = 0.3))%>%
  add_markers(marker = list(size = 0.3),
              text = ~.ID, hoverinfo = "text") %>%
  layout(title = "Parallel Coordinate Plots")

ct_olive = SharedData$new(olive_data, group = "olive")

get_buttons = function(df, axis){
  buttons = list()
  i = 1
  for(col in colnames(df)){
    buttons[[i]] = list(method = "restyle",
                        args = list(axis, list(olive_data[[col]])),
                        label = paste(axis , ": ", col, sep=""))
    i = i + 1
  }

  return(buttons)
}

buttons_x = get_buttons(olive_data[, 4:11], "x")
buttons_y = get_buttons(olive_data[, 4:11], "y")
buttons_z = get_buttons(olive_data[, 4:11], "z")

annot = list(list(text = "X", x=-0.6, y = 0.78, xref = 'paper', yref = 'paper', showarrow = FALSE),
             list(text = "Y", x=-0.6, y = 0.55, xref = 'paper', yref = 'paper', showarrow = FALSE),
             list(text = "Z", x=-0.6, y = 0.34, xref = 'paper', yref = 'paper', showarrow = FALSE))

scatter_plot_3d = plot_ly(ct_olive, x = ~palmitic, y = ~palmitoleic, z = ~stearic) %>%
  add_markers(marker = list(size=4), opacity = 0.5) %>%
  layout(scene = list(xaxis = list(title = "X"),
                      yaxis = list(title = "Y"),
                      zaxis = list(title = "Z")),
         title = "3D Scatterplot",
         # annotations = annot,
         updatemenus = list(
           list(y = 0.4, buttons = buttons_x, text = "X", active = 0),
           list(y = 0.6, buttons = buttons_y, text = "Y", active = 1),
           list(y = 0.8, buttons = buttons_z, text = "Z", active = 2)))

bar_chart = plot_ly(ct_olive, x = ~Region_category) %>%
  add_histogram() %>% layout(title = "Histogram: Region", barmode = "overlay")

x <- bscols(widths = c(12, 12,6,6),
  tags$h2("Interactive plots to analyze relationships between variables"),
       par_coord_plot %>%
         highlight(on = "plotly_select", dynamic = T, persistent = T, opacityDim = I(1)) %>%
         hide_legend(),
       scatter_plot_3d %>%
         highlight(on = "plotly_click", dynamic = T, persistent = T) %>% hide_legend(),
       bar_chart %>%
         highlight(on = "plotly_click", dynamic = T, persistent = T) %>% hide_legend())
```

```
knitr::include_graphics('./4.1.3.png')
```

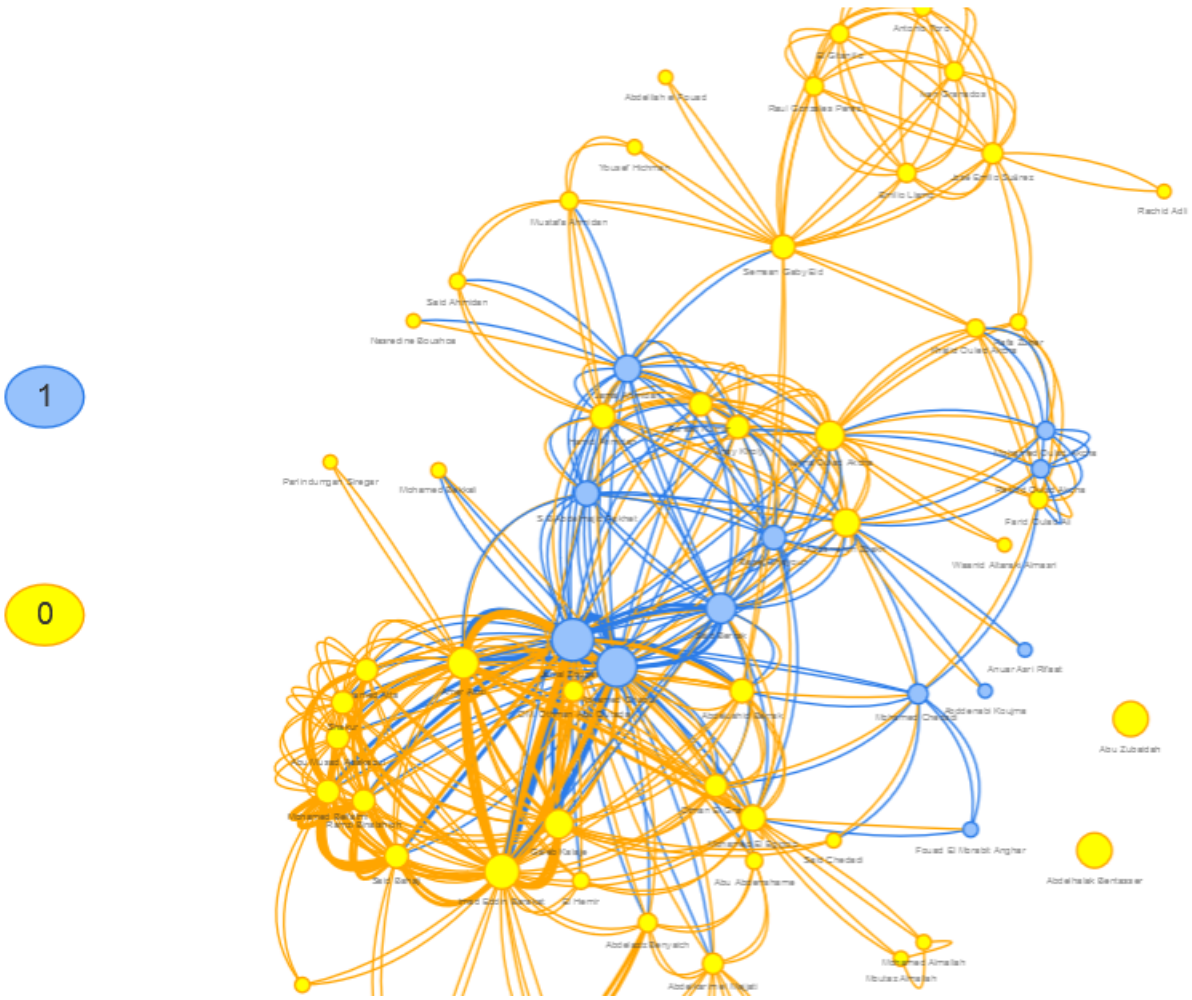


Network Graphs

Network Graph using visNetwork

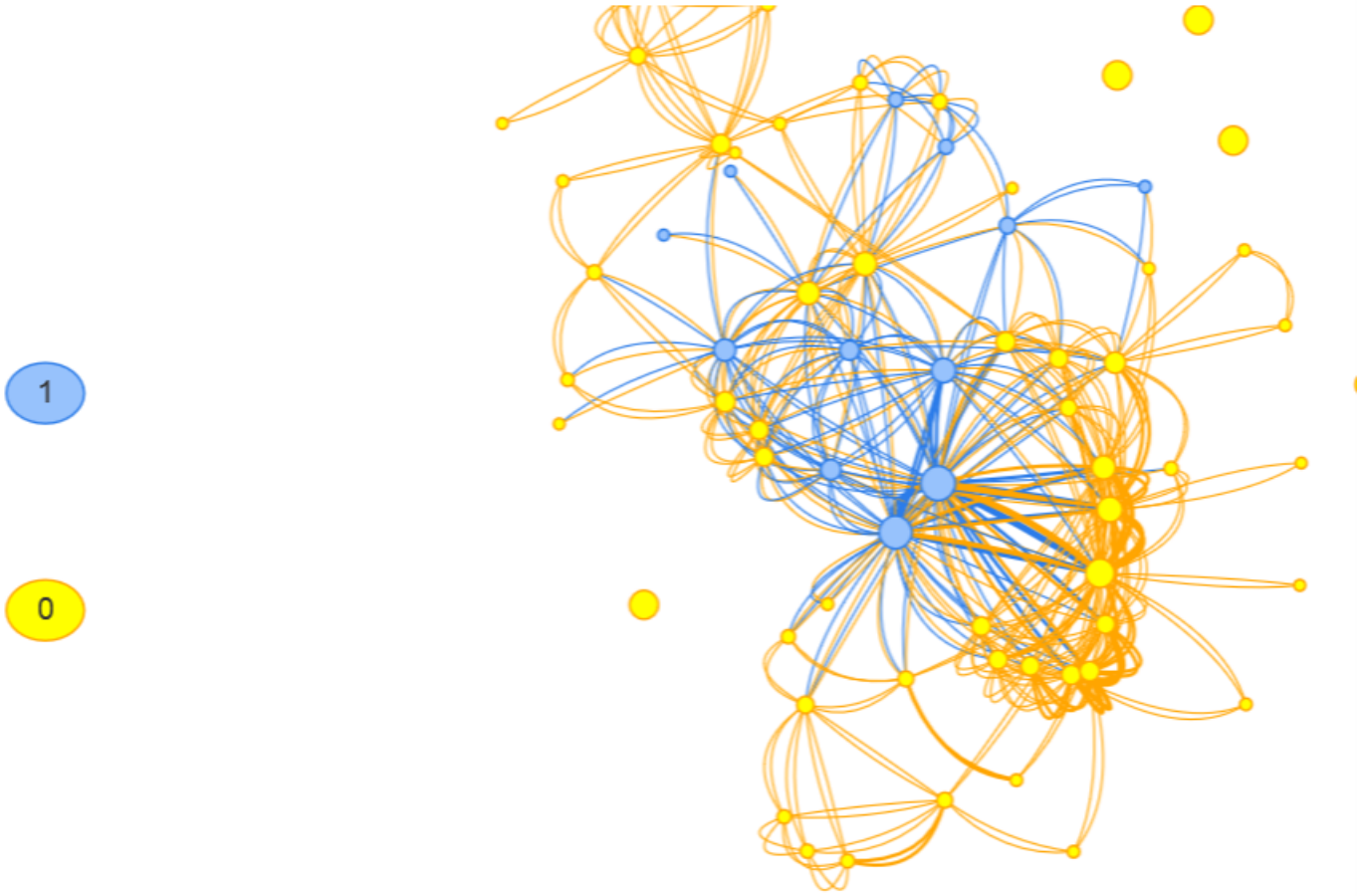
Replusion Optimzed Network Graph

```
links <- read.table("trainData.dat", as.is=T)
nodes <- as.data.frame(read.table("trainMeta.dat", as.is=T))

colnames(links) <- c("from","to","strength")
links <- links[order(links$from, links$to),]
nodes$id <- 1:70
rownames(links) <- NULL

colnames(nodes)[2] <- "BombingGrp"

#Size of links based on "strength of links"
links$width <- links$strength*3

#Nodes colored based on Bombing Group
nodes$label <- nodes$V1
nodes$group <- nodes$BombingGrp

#Size of nodes proportional to the number of connections
graph <- graph.data.frame(links, directed = F)
strength_value <- strength(graph)
nodes$value <- strength_value[match(nodes$id, names(strength_value))]

x <- visNetwork(nodes, links, main = "Network of people invloved in Madrid Bombing") %>%
  visLegend() %>%
  visLayout(randomSeed = 8) %>%
  visPhysics(solver="repulsion") %>%
  visOptions(highlightNearest = list(enabled = T, degree = 1, hover = T))

knitr::include_graphics('./5.1.1.png')
```



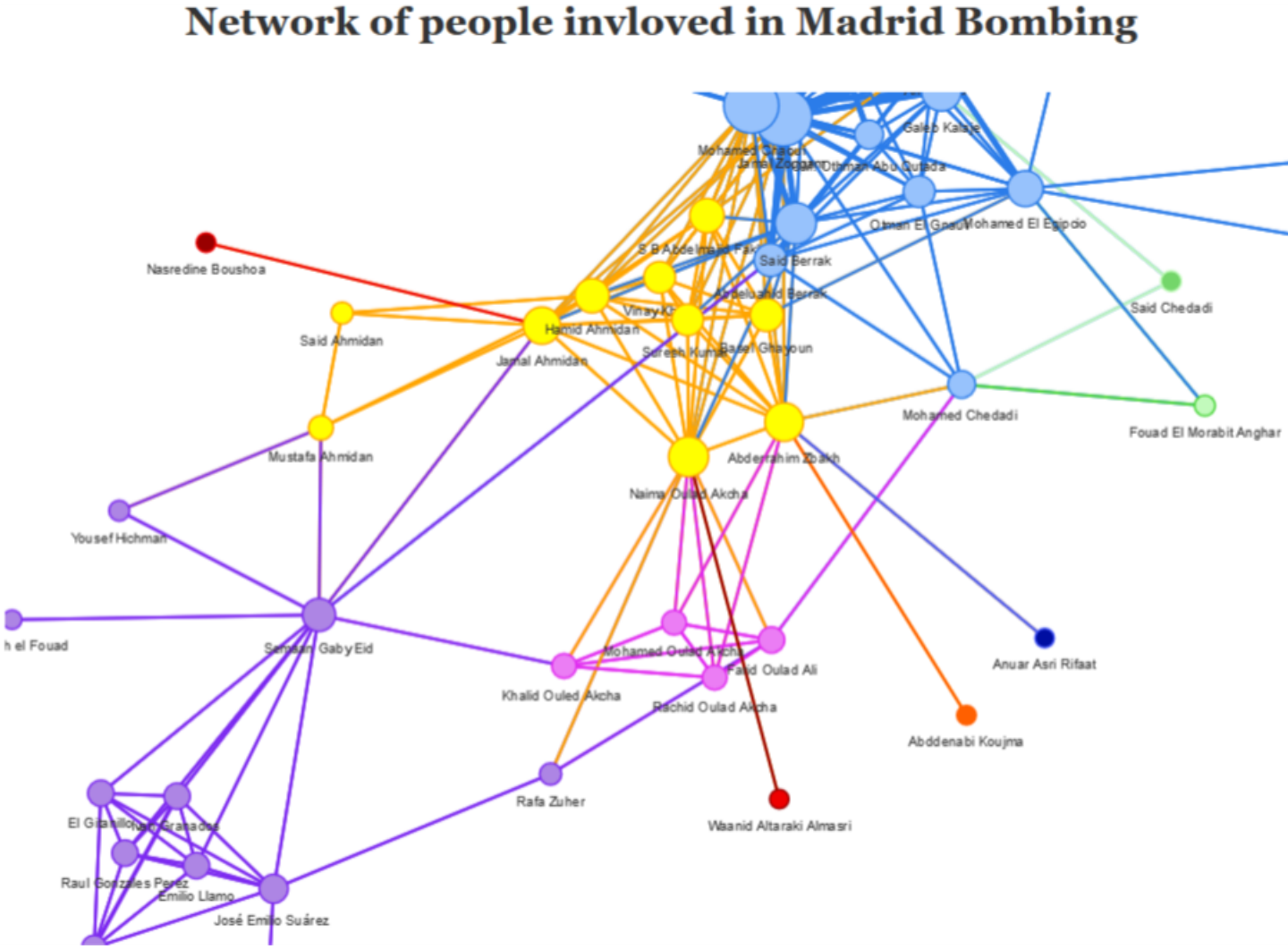
Replusion Optimzed Network Graph with highlights

```
x <- visNetwork(nodes, links, main = "Network of people invloved in Madrid Bombing") %>%
  visLegend() %>%
  visLayout(randomSeed = 12) %>%
  visPhysics(solver="repulsion") %>%
  visOptions(highlightNearest = list(enabled = T, degree = 2, hover = T))

knitr::include_graphics('./5.1.2.png')
```



Edge Between Optimizing Network Graph

```
for(i in 1:nrow(links)){
  links$from_name[i] <- nodes$V1[links$from[i]]
  links$to_name[i] <- nodes$V1[links$to[i]]
}

links <- links[,c("from_name","to_name","from","to","strength","width")]
nodes1 <- nodes
net <- graph_from_data_frame(d=links, vertices=nodes, directed=F)
ceb <- cluster_edge_betweenness(net)
nodes1$group <- ceb$membership

x <- visNetwork(nodes1,links, main = "Network of people invloved in Madrid Bombing") %>%
  visIgraphLayout() %>%
  visOptions(highlightNearest = list(enabled = T, degree = 2, hover = T))

knitr::include_graphics('./5.1.3.png')
```

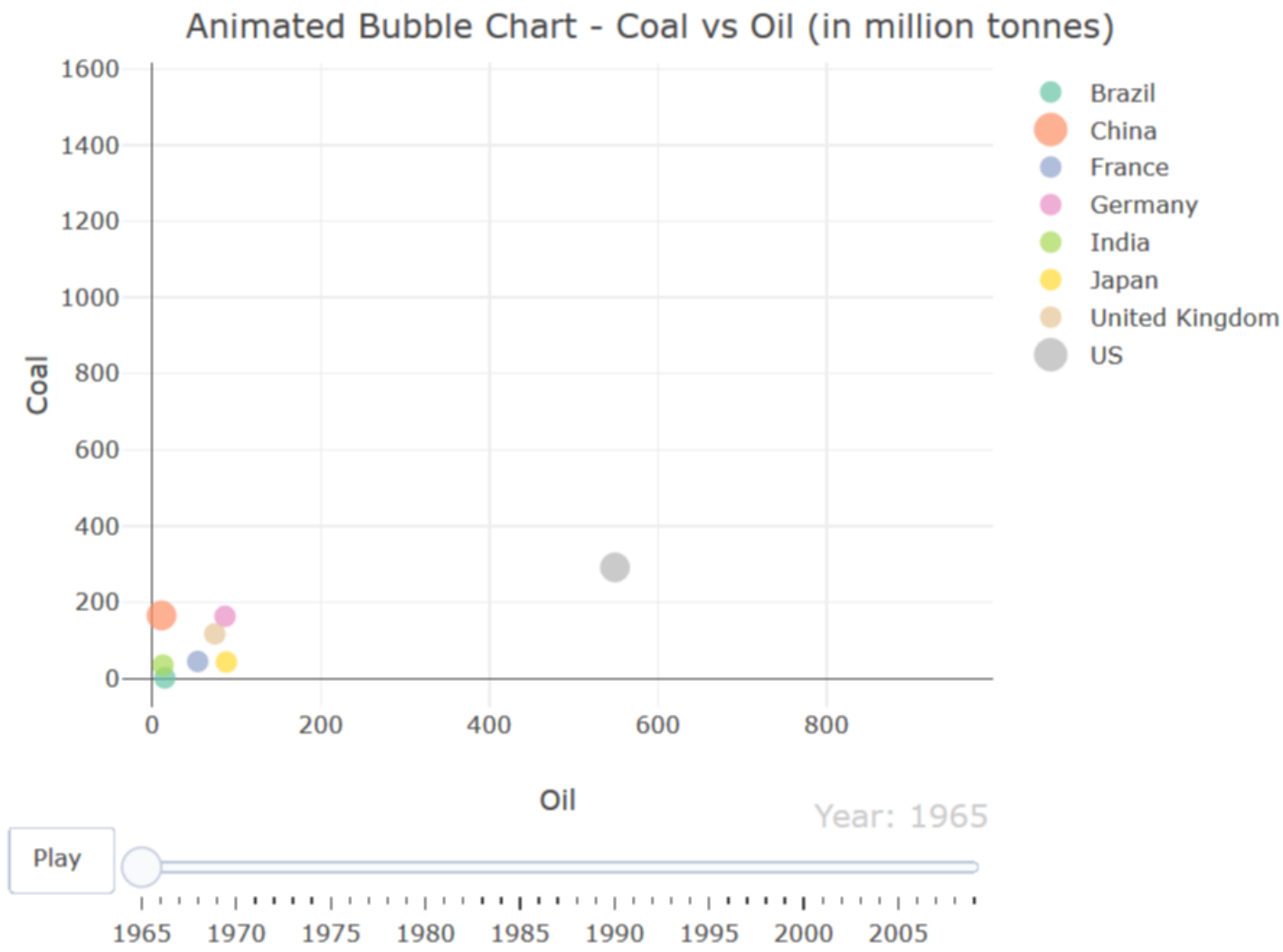


Animated Plots

Bubble Chart

Bubble Chart Animated with Cubic Easing

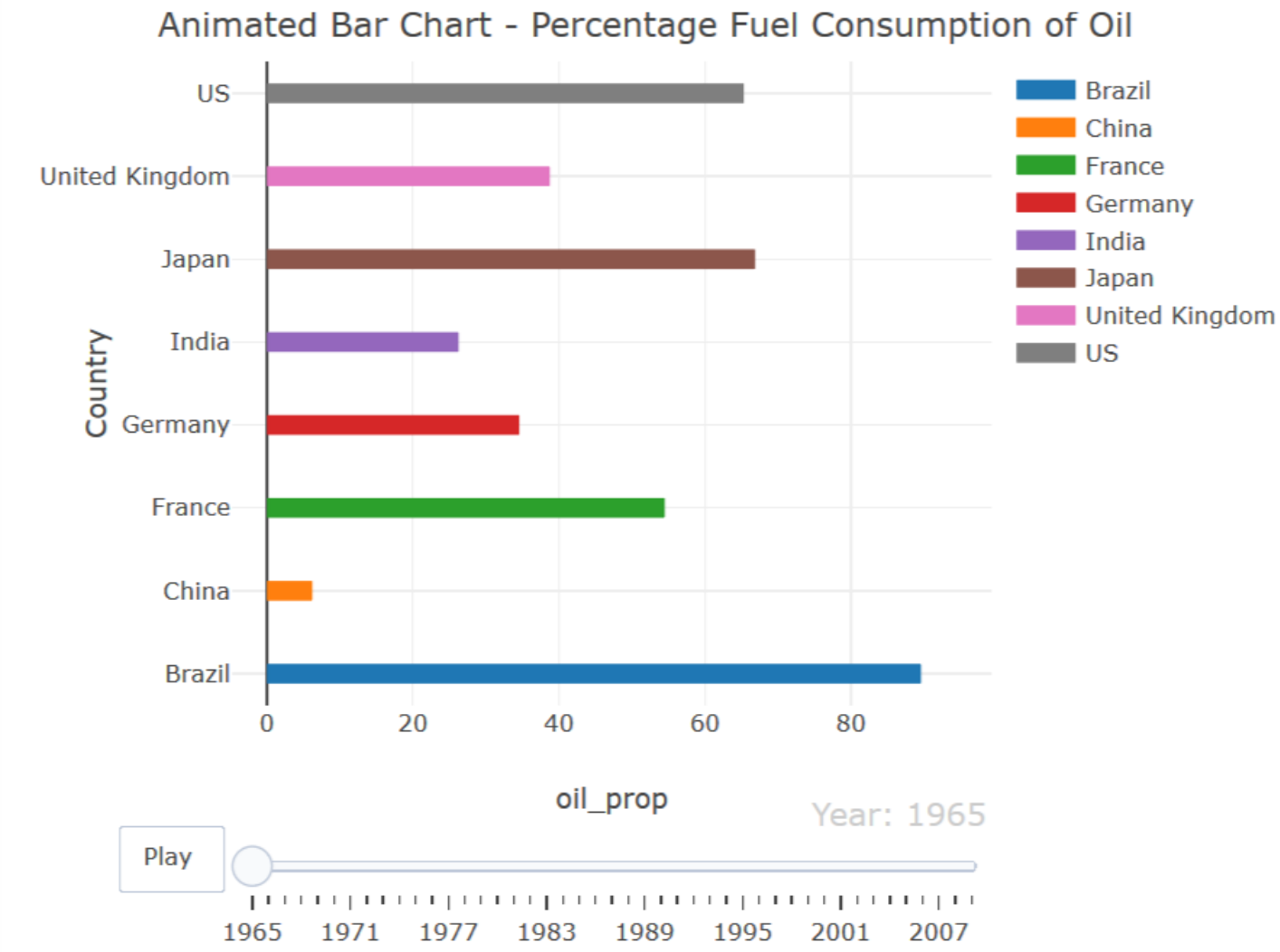
```
oc_data = read.csv2("Oilcoal.csv")

x <- plot_ly(oc_data, x= ~Oil, y= ~Coal, frame = ~Year, color = ~Country) %>%
  add_markers(size = ~Marker.size, marker = list(sizemin = 5)) %>%
  animation_opts(500, easing = "cubic", redraw = F) %>%
  layout(title = "Animated Bubble Chart - Coal vs Oil (in million tonnes)")

knitr::include_graphics('./6.1.1.png')
```



Bar Chart

Bar Chart Animated with Elastic Easing

```
oc_data$oil_prop = 100* oc_data$Oil / (oc_data$Oil + oc_data$Coal)
oc_data_zero = oc_data
oc_data_zero$oil_prop = 0
oc_line_anim_data = rbind(oc_data, oc_data_zero)

x <- plot_ly(oc_line_anim_data, x = ~oil_prop, y = ~Country, frame = ~Year) %>%
  add_lines(split = ~Country, line = list(width = 10)) %>%
  animation_opts(500, easing = "elastic", redraw = F) %>%
  layout(title = "Animated Bar Chart - Percentage Fuel Consumption of Oil")

knitr::include_graphics('./6.2.1.png')
```



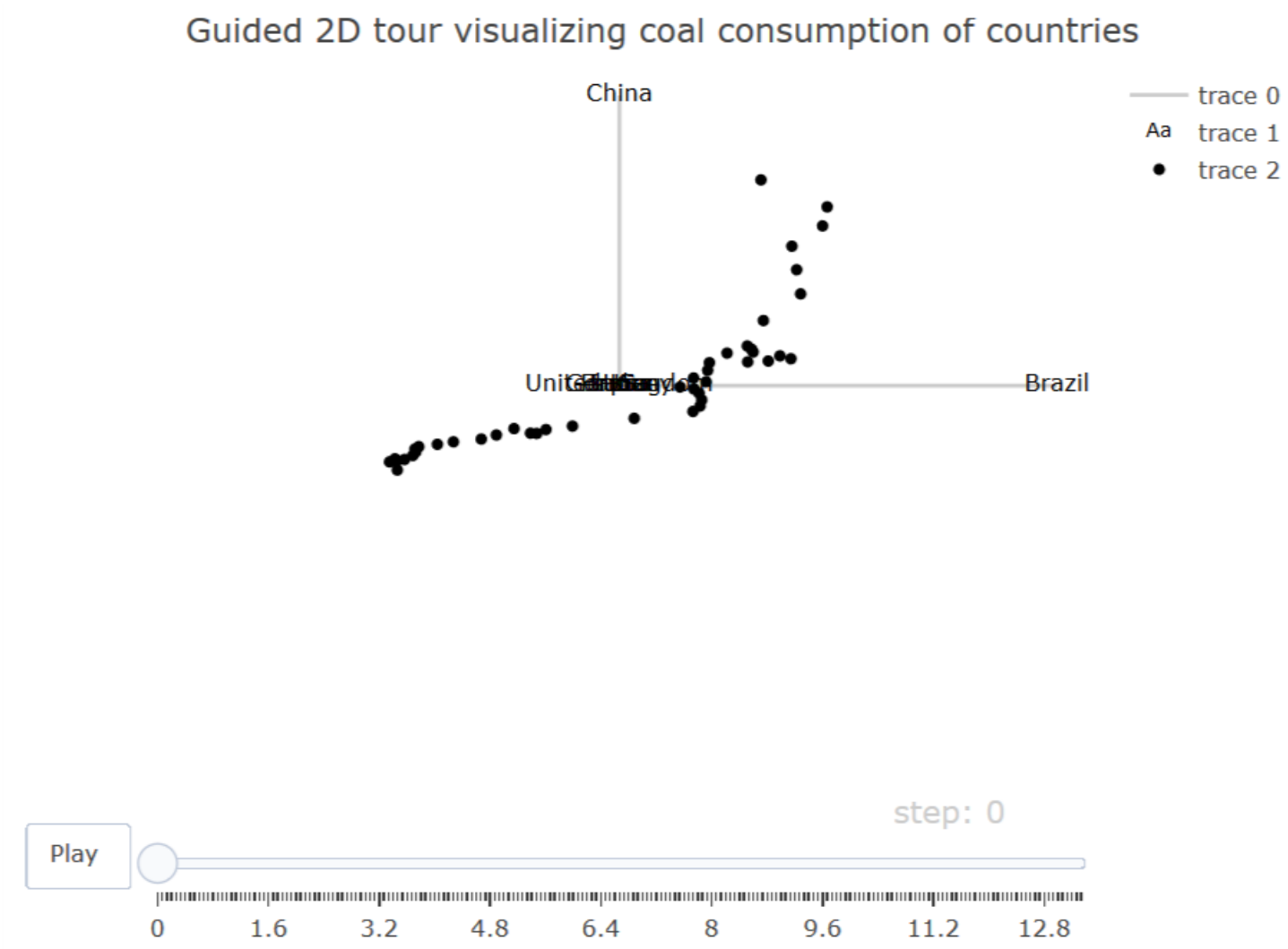
Tour and Projection

Animated with grand tour

```
oc_tour_data = cast(oc_data[,1:3], Year ~ Country, value = "Coal")

mat <- rescale(oc_tour_data)
set.seed(12345)
#tour <- new_tour(mat, 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(mat, guided_tour(cmass), NULL)
    step <- tour(step_size)
```

```
  }
  step
})

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

# 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(mat)
  )
}

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")

# step = 7.67
```

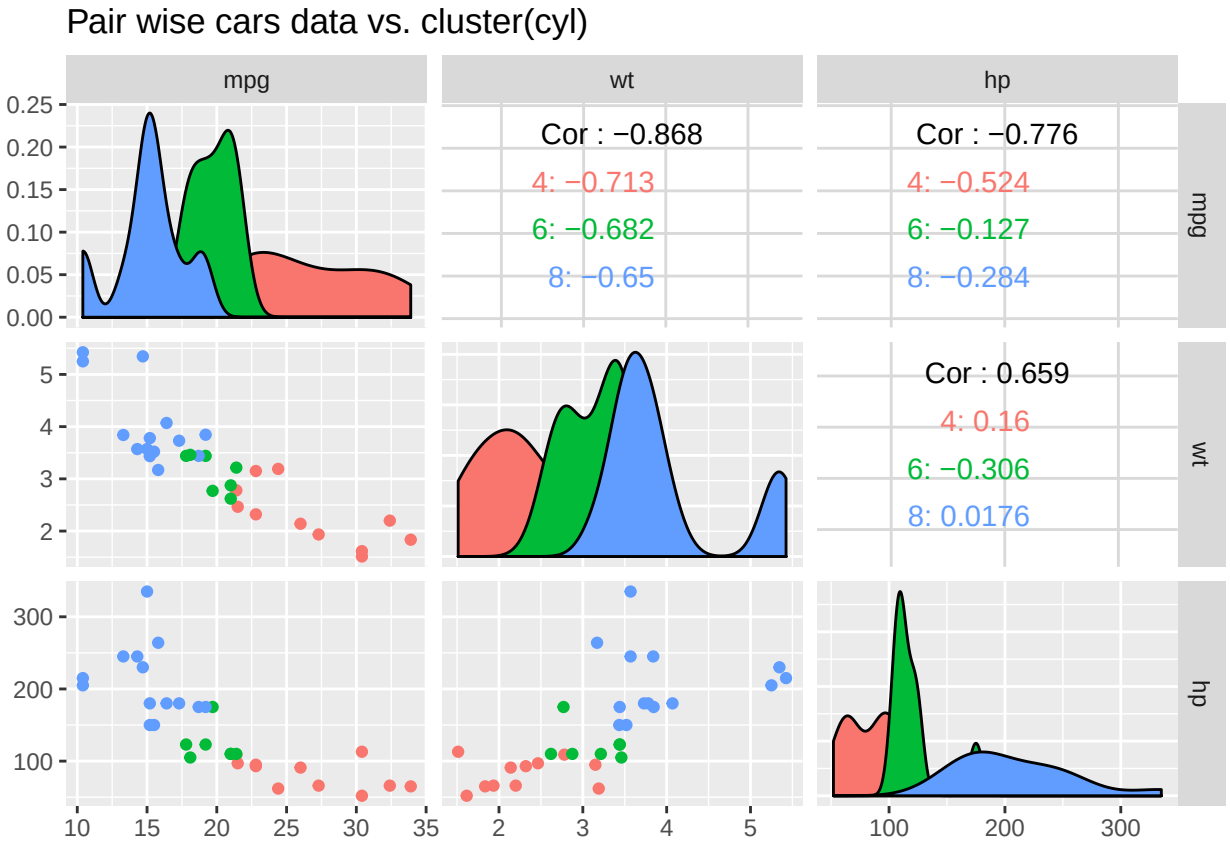
```
x <- proj_dat %>%
  plot_ly(x = ~x, y = ~y, frame = ~step, color = I("black")) %>%
  add_segments(xend = 0, yend = 0, color = I("gray80")) %>%
  add_text(text = ~variable) %>%
  add_markers(data = tour_dat, text = ~state, ids = ~state, hoverinfo = "text") %>%
  layout(xaxis = ax, yaxis = ax, title = "Guided 2D tour visualizing coal consumption of countries")

knitr::include_graphics('./7.1.1.png')
```



Multiple Plots

QC Scatter Matrix Plots using GGally library

```
ggpairs(mtcars, columns = c("mpg", "wt", "hp"),
        title = "Pair wise cars data vs. cluster(cyl)",
        ggplot2::aes(colour=as.factor(cyl)))
```

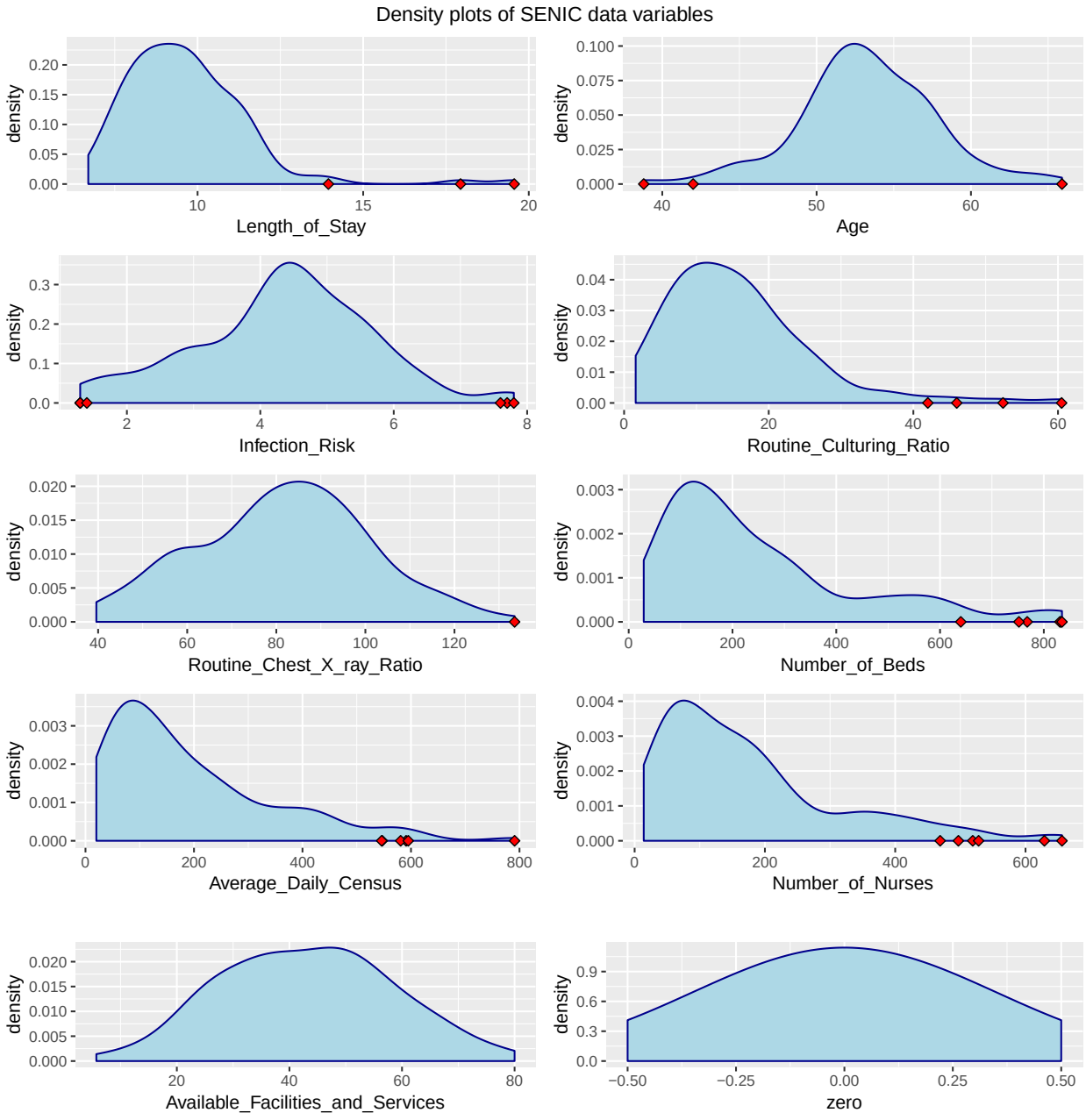


Density Plots

Density Plot with Outlier Highlight

```
plot_density_with_outliers <- function(var_data, col_name){
  p <- NULL
  df_data = setNames(data.frame(var_data),col_name)
  if(length(get_outliers(df_data[[col_name]])) > 0){
    p <- ggplot(df_data) +
      geom_density(aes_string(x=col_name), fill = "lightblue", color = "darkblue") +
      geom_point(data=df_data[get_outliers(df_data[[col_name]]),,drop=FALSE],
                 aes_string(x=col_name), y=0, shape=23, size=2, colour="black", fill="red")
  }
  else{
    p <- ggplot(df_data) +
      ggtitle(paste("")) +
      theme(plot.title = element_text(hjust = 0.5)) +
      geom_density(aes_string(x=col_name), fill = "lightblue", color = "darkblue")
  }

  return(p)
}

categorical_columns = c("Medical_School_Affiliation", "Region")
```

```
ID_columns = c("Identification_Number")
quantitative_columns = setdiff(colnames(senic_data), c(categorical_columns, ID_columns))

plot_list = mapply(plot_density_with_outliers, senic_data[, quantitative_columns],
                   colnames(senic_data[, quantitative_columns]), SIMPLIFY = FALSE)
plot_matrix <- arrangeGrob(grobs = plot_list, ncol = 2)
grid.arrange(plot_matrix, respect=TRUE, top="Density plots of SENIC data variables")
```



Shiny

```
#UI component
ui <- fluidPage(
  sliderInput(inputId="bw_value", label="Choose bandwidth size", value=4.5, min=0.1, max=80),
  checkboxGroupInput("selected_variables", "Variables to show: ", quantitative_columns, inline=TRUE),
  plotOutput("densPlot", height = "650px")
)

plot_density_with_outliers_shiny <- function(df_data, col_name, bw){
  p <- NULL
  if(length(get_outliers(senic_data[[col_name]])) > 0){
    p <- ggplot(df_data) +
      ggtitle(paste("Density plot of ", col_name)) +
      theme(plot.title = element_text(hjust = 0.5)) +
      geom_density(aes_string(x=col_name), fill = "lightblue", color = "darkblue", bw=bw) +
      geom_point(data=df_data[get_outliers(df_data[[col_name]]),],
                 aes_string(x=col_name, y=0), shape=23, size=2, colour="black", fill="red")
  }
  else{
    p <- ggplot(df_data) +
      ggtitle(paste("Density plot of ", col_name)) +
      theme(plot.title = element_text(hjust = 0.5)) +
      geom_density(aes_string(x=col_name), fill = "lightblue", color = "darkblue", bw=bw)
  }

  return(p)
}


server <- function(input, output) {

  output$densPlot <- renderPlot({

    selected_columns = input$selected_variables
    plot_list = vector("list", length(selected_columns))

    if(length(selected_columns) > 0){
      for(i in 1:length(selected_columns)){
        plot_list[[i]] = plot_density_with_outliers_shiny(senic_data, selected_columns[i],
                                                          bw = input$bw_value)
      }
      plot_matrix <- arrangeGrob(grobs = plot_list, ncol = 2)
      grid.arrange(plot_matrix)
    }

  })
}

shinyApp(ui = ui, server = server, options = list(width="800px", height="900px"))
```

Appendix Code

```
# ```{r, ref.label=knitr::all_labels(), echo=TRUE, eval=FALSE}
# ```
```