Exploratory Data Analysis with R

Multivariate Data Visualization

Xuemao Zhang
East Stroudsburg University

October 26, 2022

Outline

- Scatter plot matrix
- Correlogram
- Heat plot
- Superheatplot
- 3D plot

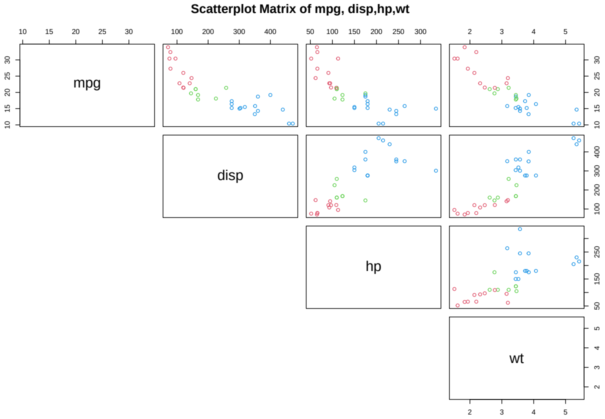
Multivariate data

- What is multivariate data?
- What makes multivariate analysis different from univariate analysis?
  - Data is multivariate, if we have more information than a single aspect for each entity/person/experimental unit. Mutivariate analysis takes **relationships** between these different aspects into account.

Scatterplot matrix

- pairs() function creates beautiful scatter plot matrix.

```
mtcars$cyl=factor(mtcars$cyl);
pairs(~mpg+disp+hp+wt,data=mtcars, col=c(2,3,4)[mtcars$cyl],
main="Scatterplot Matrix of mpg, disp,hp,wt ",lower.panel = NULL);
```



Scatterplot matrix

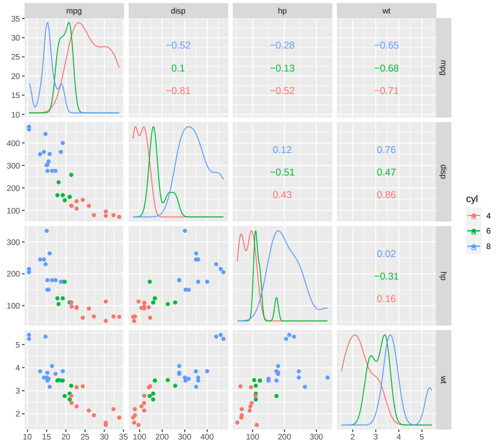
- GGally R package extends 'ggplot2' by adding several functions to reduce the complexity of combining geometric objects with transformed data. For example, pairwise plot matrix. The package is maintained by Barret Schloerke.
- The function `ggscatmat` and `ggpairs()` are used to produce a matrix of scatter plots for visualizing the correlation between variables.
- The function `ggcorr()` is used to make a nice correlation matrix plot.

Scatterplot matrix

- The function `GGally::ggscatmat` strictly take numeric variables.

```
library(dplyr); library(GGally);
mtcars %>% select(mpg,disp,hp,wt,cyl) %>% ggscatmat(color="cyl");
```

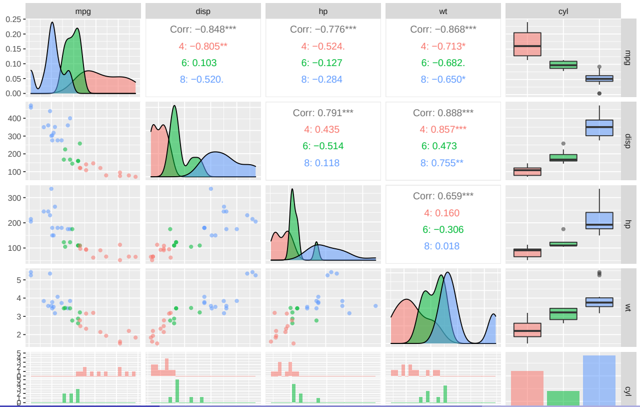
```
## Warning in ggscatmat(., color = "cyl"): Factor variables are omit
```



Scatterplot matrix

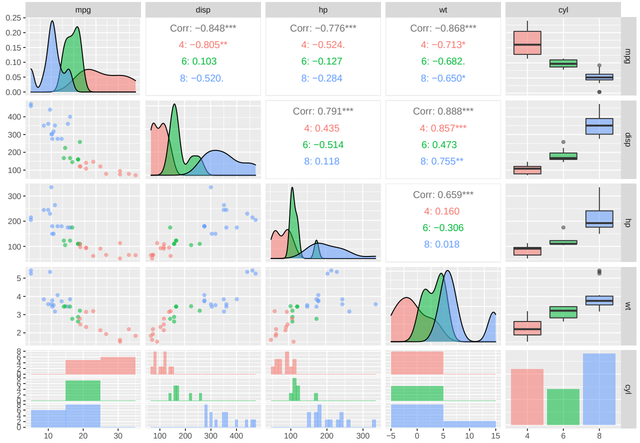
```
mtcars %>% select(mpg,disp,hp,wt,cyl) %>%
  ggpairs(aes(color=cyl,alpha = 0.4));
```

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



Scatterplot matrix

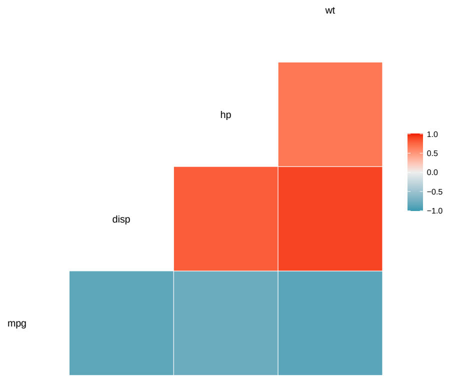
```
mtcars %>% select(mpg,disp,hp,wt,cyl) %>%
  ggpairs(aes(color=cyl,alpha = 0.4),
    lower = list(combo=wrap("facethist",binwidth=10)));
```



Correlogram

- The function GGally::ggcorr() is used to make a nice correlation matrix plot.

```
mtcars %>% select(mpg,disp,hp,wt) %>% ggcorr();
```



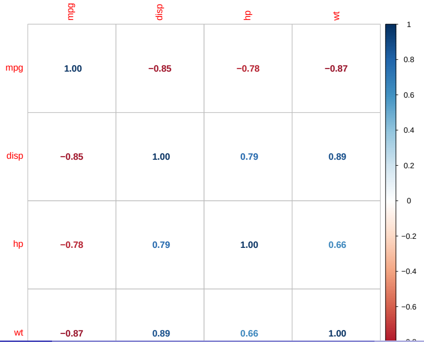
Correlogram

- corrplot can shows the linear correlation coefficients in the correlation matrix plot

```
library(corrplot);
```

```
## corrplot 0.92 loaded
```

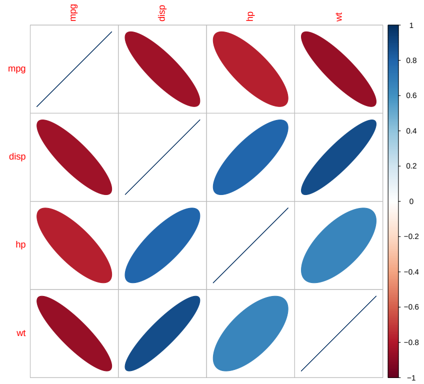
```
mtcars1 = mtcars %>% select(mpg,disp,hp,wt);
corrMat=cor(mtcars1);  #calculate the linear correlations
corrplot(corrMat, method = "number");
```



Correlogram

- corrplot can show the correlations using symbols and colors.

```
corrplot(corrMat, method = "ellipse");
```



Heatmap

- Heat maps is a very useful graphical tool to better understand or present data stored in **matrix** forms.
- A **heatmap** is basically a table that has colors in place of numbers.
- Colors correspond to the level of the measurement.
- It is quite straight forward to make a heat map, as shown on the examples in this lecture. However be careful to understand the underlying mechanisms.
- You might prefer to conduct **cluster analysis** and then permute the rows and the columns of the matrix to place similar values near each other according to the clustering. Cluster analysis will not be discussed.

Heat map using `heatmap()`

- Let's start with a very simple matrix

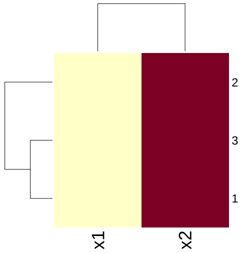
```r
A= matrix(c(1, 2, 3, 15, 2, 8), byrow=T, nrow = 3, ncol = 2)
rownames(A) =c("1","2","3")
colnames(A) =c("x1","x2")
print(A)
```

```
##   x1 x2
## 1  1  2
## 2  3 15
## 3  2  8
```

Heat map using `heatmap()`

- Next, we can prepare a basic heat map. In the following heat map,
  - The x axis represents columns in matrix. The first column is on the left (the lowest value on the axis), the second column is on the right (analogously - the highest value).
  - The y axis represents rows and the first row is on the bottom.
  - By default, red colour represents the highest values in our matrix, while the lowest are lighter.

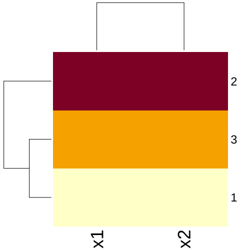
```
heatmap(A)
```



Heat map using `heatmap()`

- The `scale` argument is used to specify if values should be normalized (centered and scaled) in row/column direction. The variables are comparable after normalization.
- We now center and scale the columns to compare individuals (rows).

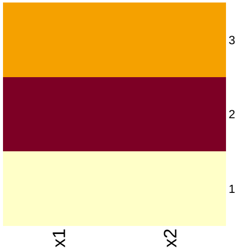
```
heatmap(A,scale="column") #By default, scale="row"
```



Heat map using `heatmap()`

- You may want to clean it up by removing the dendrograms produced by the cluster analysis.

```
heatmap(A,scale="column",Rowv=NA, Colv=NA)
```



Heat map using `heatmap()`

- Let's plot a heat map for the data set `mtcars`.

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ..
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Heat map using `heatmap()`

- Convert the data frame to a matrix.

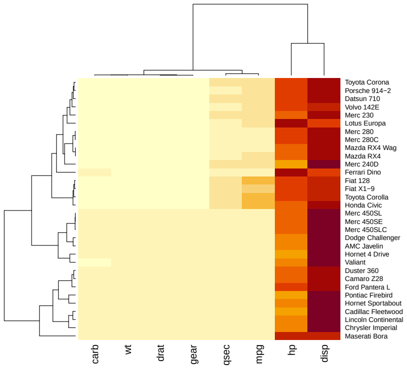
```
#remove the categorical variables cyl, vs, am
mtcars %>% select(-cyl,-vs, -am)  %>% as.matrix() -> data
head(data)
```

```
##                    mpg disp  hp drat    wt  qsec gear carb
## Mazda RX4         21.0  160 110 3.90 2.620 16.46    4    4
## Mazda RX4 Wag     21.0  160 110 3.90 2.875 17.02    4    4
## Datsun 710        22.8  108  93 3.85 2.320 18.61    4    1
## Hornet 4 Drive    21.4  258 110 3.08 3.215 19.44    3    1
## Hornet Sportabout 18.7  360 175 3.15 3.440 17.02    3    2
## Valiant           18.1  225 105 2.76 3.460 20.22    3    1
```

Heat map using `heatmap()`

- The default heatmap is not really informative. Indeed, the hp and disp variable have really high values which make that the other variables with small values all look the same.

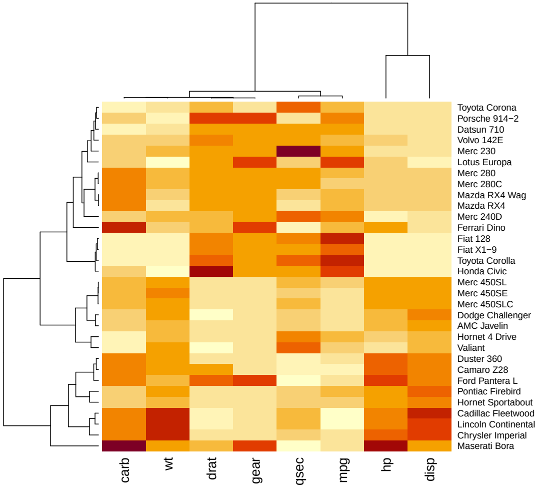
```
heatmap(data)
```



Heat map using heatmap()

- We need to normalize the data using scale.

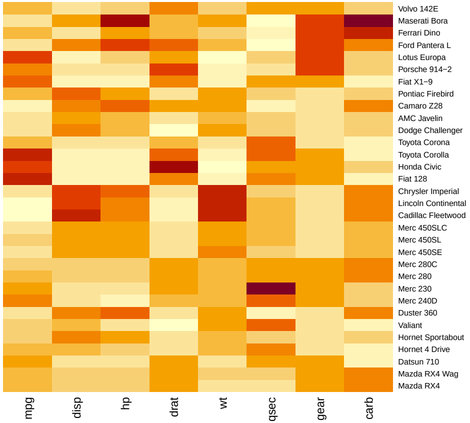
```
heatmap(data,scale="column")
```



Heat map using `heatmap()`

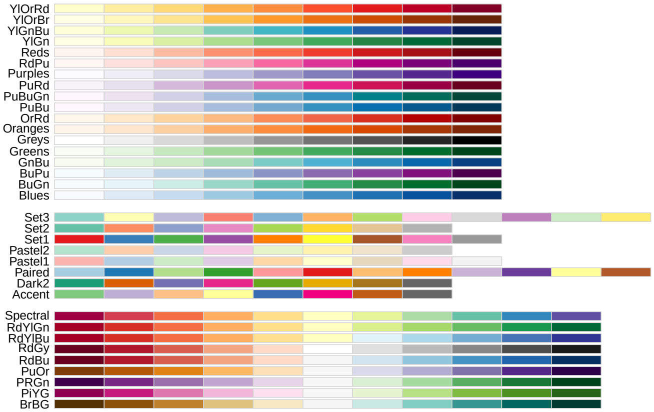
- It is noticed that order of both rows and columns is different compared to the raw mtcar matrix. This is due to cluserizations.
- To visualize the raw matrxi, we use the Rowv and Colv arguments.

```r
heatmap(data, Colv = NA, Rowv = NA, scale="column")
```



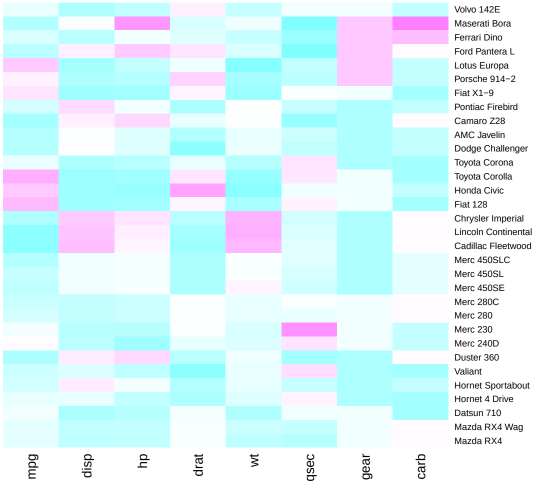
Heat map using `heatmap()`

- Custom colors: There are several ways to custom the color palette.
  - use the native palettes of R: terrain.color, rainbow, heat.colors, topo.colors or cm.colors (https://stat.ethz.ch/R-manual/R-devel/library/grDevices/html/palettes.html);
  - use the **Palettes** proposed by RColorBrewer.



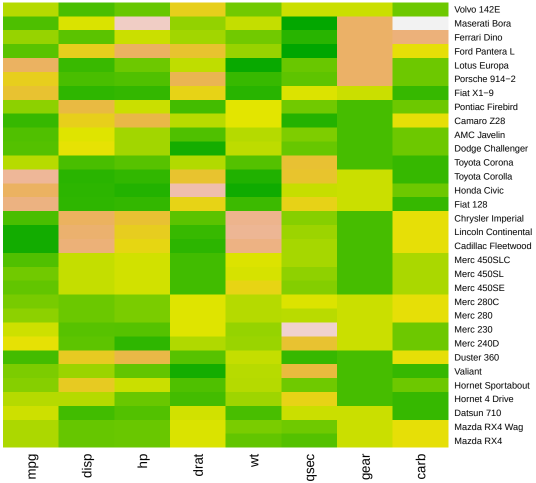
Heat map using `heatmap()`

```r
# native palette from R
heatmap(data, Colv = NA, Rowv = NA,scale="column",
        col = cm.colors(256));
```



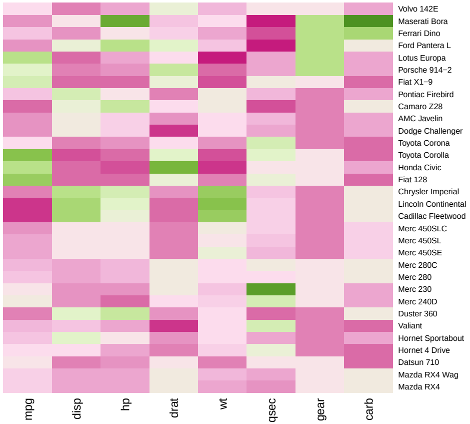
Heat map using heatmap()

```
# native palette from R
heatmap(data, Colv = NA, Rowv = NA, scale="column",
        col = terrain.colors(256));
```



Heat map using `heatmap()`

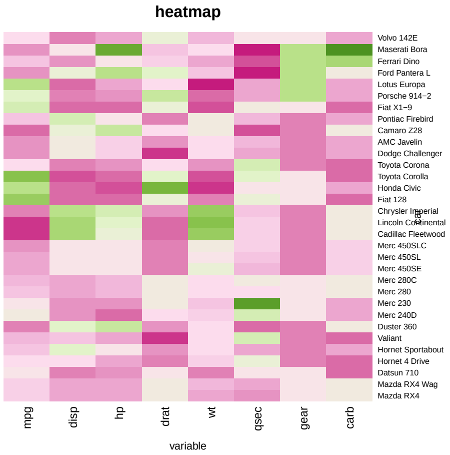
```r
#Rcolorbrewer palette
library(RColorBrewer)
coul = colorRampPalette(brewer.pal(8, "PiYG"))(25)
heatmap(data, Colv = NA, Rowv = NA, scale="column", col = coul)
```



Heat map using `heatmap()`

- We can custom title & axis titles with the usual `main` and `xlab`/`ylab` arguments.

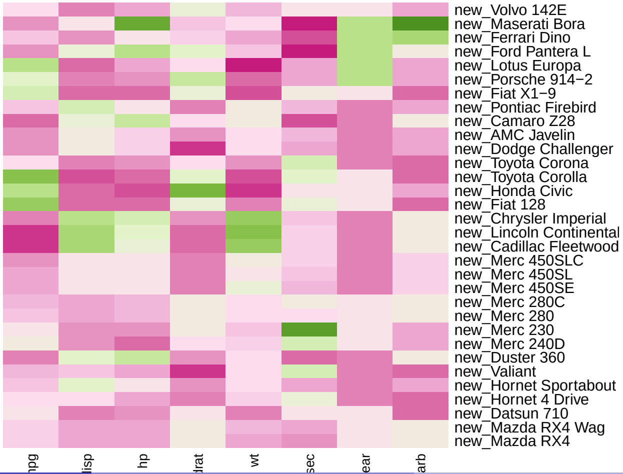
```
heatmap(data, Colv = NA, Rowv = NA, scale="column", col = coul,
        xlab="variable", ylab="car", main="heatmap")
```



Heat map using `heatmap()`

- You can also change labels with `labRow`/`colRow` and their size with `cexRow`/`cexCol`.

```
heatmap(data, Colv = NA, Rowv = NA, scale="column", col = coul,
    cexRow=1.5, labRow=paste("new_", rownames(data),sep=""))
```

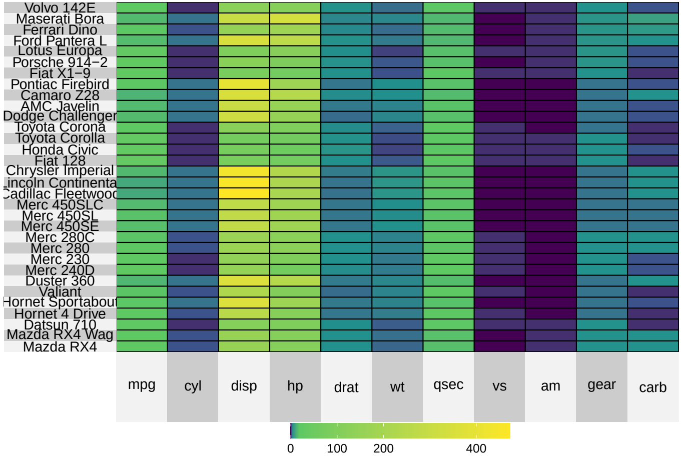


Superheat Map

- A heatmap maps a quantitative variable to colour, and displays in a matrix layout. Often accompanied by a clustering, or ordering of rows and columns. Colour mapping sometimes is difficult for a user to accurately read the information. The coordinate system underlying this form of mapping is not clear, its not Euclidean.
- The `superheat` package enhances the traditional heatmap by providing a platform to visualize a wide range of data types simultaneously, adding to the heatmap a response variable as a scatterplot, model results as boxplots, correlation information as barplots, text information, and more. Superheat allows the user to explore their data to greater depths and to take advantage of the heterogeneity present in the data to inform analysis decisions.
- The package consists of a single function: `superheat`

Superheat Map

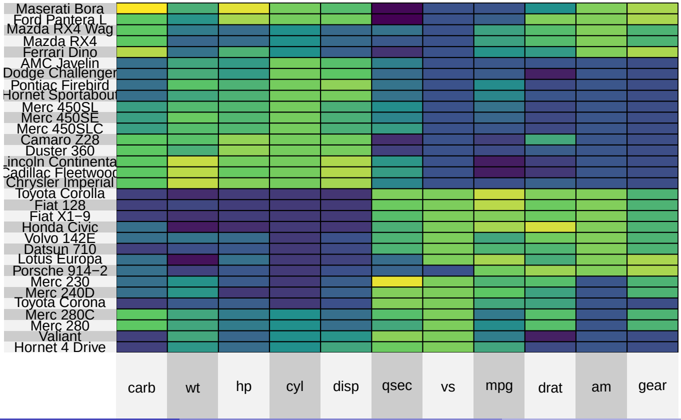
```r
library(superheat)
mtcars$cyl=as.numeric(mtcars$cyl)
#data must contain numeric entries only
superheat(mtcars)
```



Superheat Map

- You can set the ordering of the rows/columns based on a “pretty” hierarchical clustering by specifying pretty.order.rows = TRUE and pretty.order.cols = TRUE. Errors may arise when the matrix has missing values.

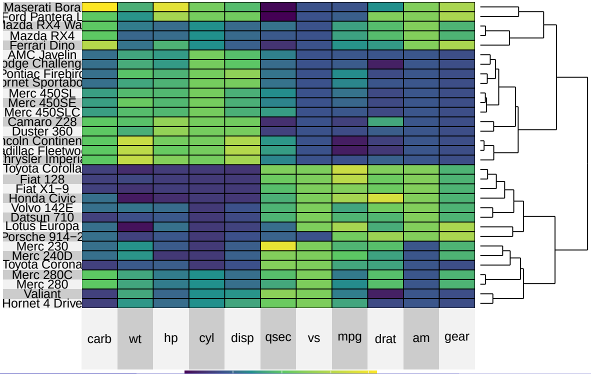
```
superheat(mtcars,scale=TRUE, pretty.order.rows=TRUE,
     pretty.order.cols=TRUE);
```



Superheat Map

- It is natural to supply a dendrogram that highlights the hierarchical clustering of the columns and/or rows using the `col.dendrogram` and row.dendrogram arguments.
- For more information, read https://rlbarter.github.io/superheat/index.html

```
superheat(mtcars,scale=TRUE, pretty.order.rows=TRUE,
    pretty.order.cols=TRUE, row.dendrogram = TRUE)
```



3D plot

- The rgl package is used to produce interactive 3-D plots.
- The package `rgl`(https://cran.r-project.org/web/packages/rgl/rgl.pdf) is maintained by Duncan Murdoch.
- Output may be on screen using OpenGL, or to various standard 3D file formats including WebGL, PLY, OBJ, STL as well as 2D image formats, including PNG, Postscript, SVG, PGF.

3D scatter plot using plot3d

- The function `plot3d(x,y,z)` can be used to draw a 3D scatter plot. It is similar to the classic `plot` function, but works in 3 dimensions. If you call `plot3d` again, it will overwrite the current plot.

```r
library(rgl)
x = iris$Sepal.Length
y = iris$Petal.Length
z = iris$Sepal.Width
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species));
# "s" is for spheres
```

Exporting images: Export images as png or pdf

- `rgl` has functions to save snapshots or other recordings of a scene, without any 3D information being saved
- The function `rgl.snapshot()` is used to save the screenshot as png file:

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species))
rgl.snapshot(filename = "plot1.png")
```

- The function `rgl.postscript()` is used to save the screenshot to a file in ps, eps, tex, pdf, svg or pgf format:

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species))
rgl.postscript("plot1.pdf", fmt="pdf")
```

Export the plot into an interactive PLY file

- writePLY() is used to export a plot into a PLY file commonly used in 3D printing.

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species))
writePLY("plot1.ply")
```

Export the plot into an interactive HTML file

- Currently the best way to embed an rgl scene in a document is to produce the document in HTML using R Markdown. Early in the document, you should have code like this in one of the setup code chunks:

~~~
```{r echo=FALSE, include=FALSE}

library(rgl);

setupKnitr(autoprint = TRUE);

```
~~~

License