Introduction to Heat Map

Xuemao Zhang
Department of Mathematics
East Stroudsburg University

June 27, 2019

Outline

- Overview
- Heat map using `heatmap()`
- Correlogram
- Contour plot for county population data (if time allows)

Overview

- Install the following packages if you don't have them.

```
install.packages("ggplot2");
install.packages("usmap");
install.packages("dplyr");
install.packages("RColorBrewer");
install.packages("car");
install.packages("corrplot");
install.packages("mvtnorm");
install.packages("MASS");
install.packages("scatterplot3d");
```

Overview

- 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 today.

Heat map using heatmap()

- Let's start with a very simple matrix

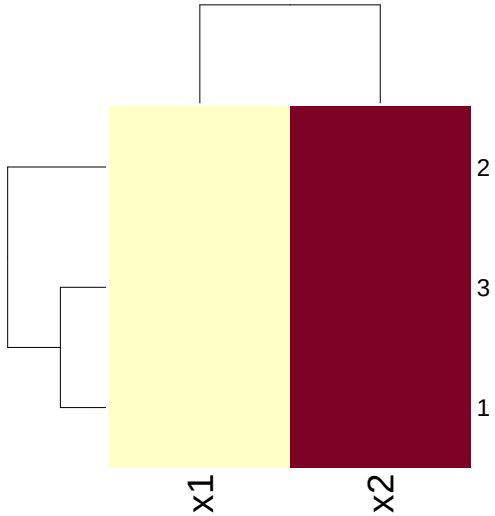
```
A= matrix(c(1, 2, 3, 15, 2, 8), byrow=T, nrow = 3, ncol = 2);
rownames(A) =c("1","2","3");
colnames(A) =c("x1","x2");
print(A);
```

```
##   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.

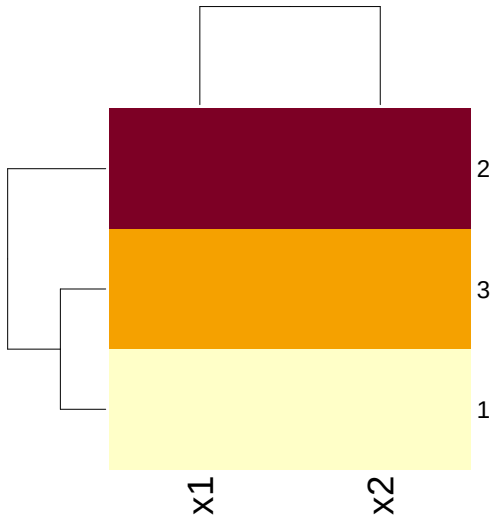
```
heatmap(A);
```



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).

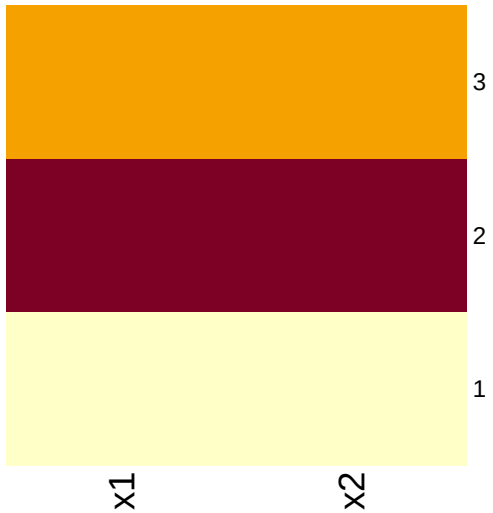
```
heatmap(A,scale="column"); #By default, scale="row"
```



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 a larger 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 : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ 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.

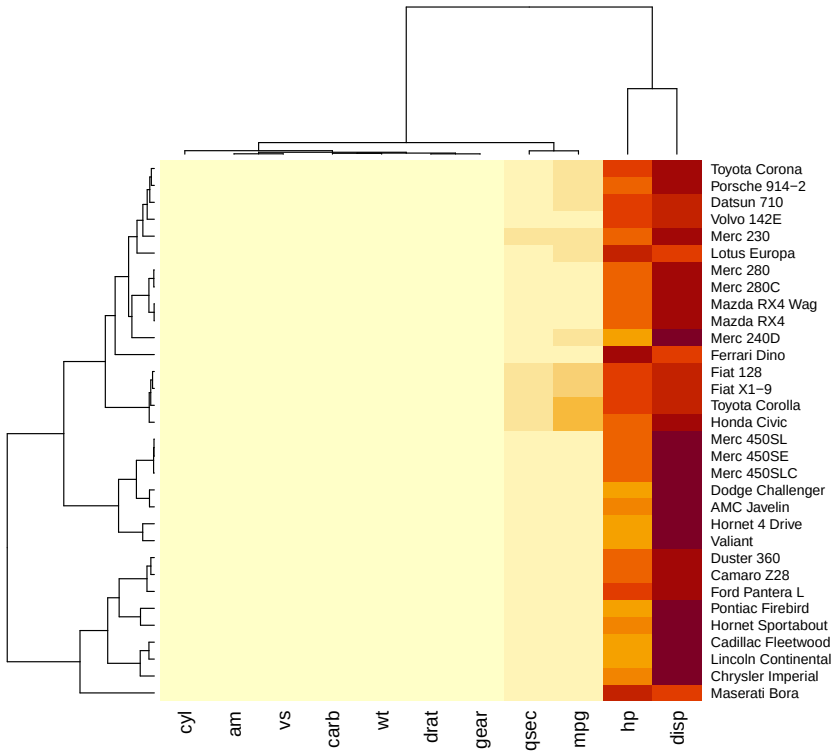
```
data=as.matrix(mtcars);
head(mtcars);
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear c
## Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4
## Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4
## Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4
## Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3
## Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0    3
## Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0    3
```

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.

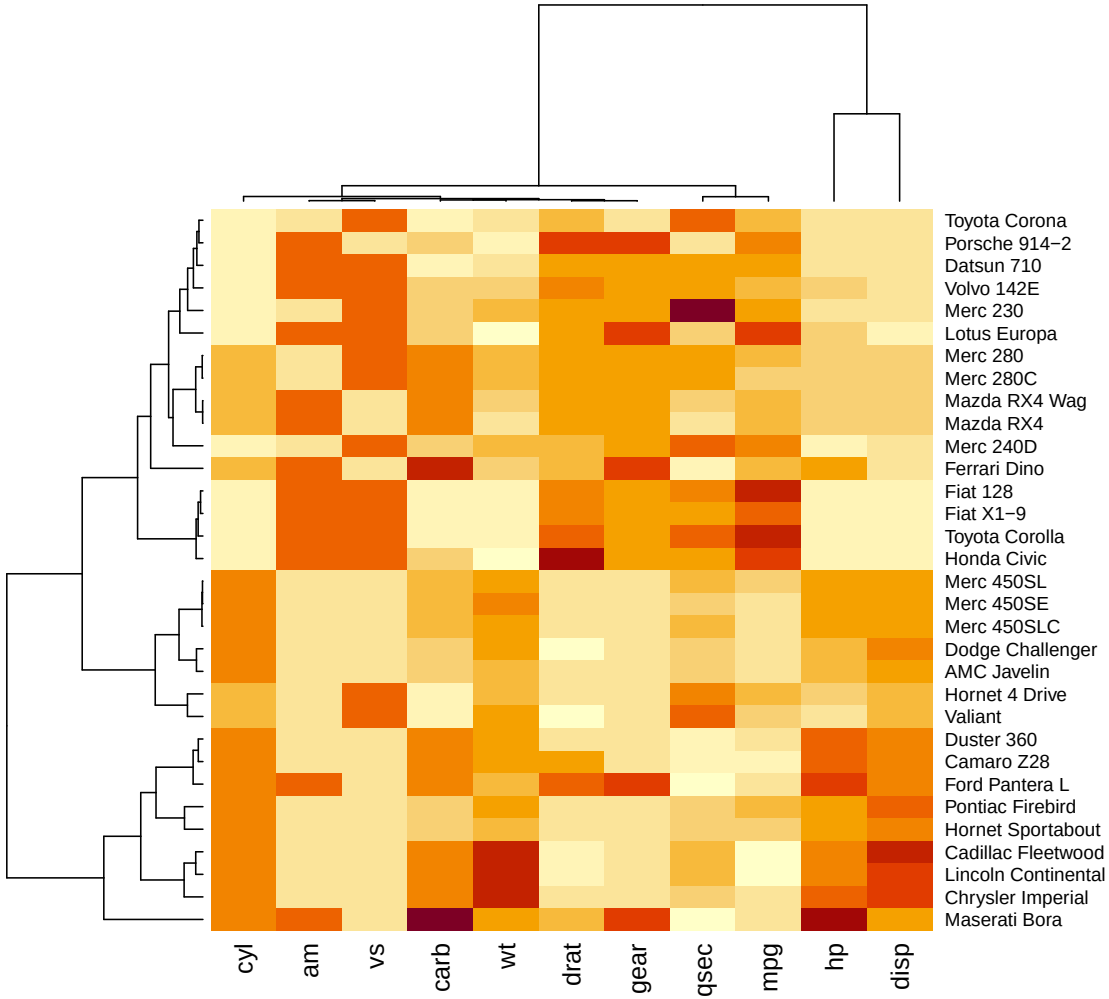
```
heatmap(data);
```



Heat map using `heatmap()`

- We need to normalize the data using `scale`.

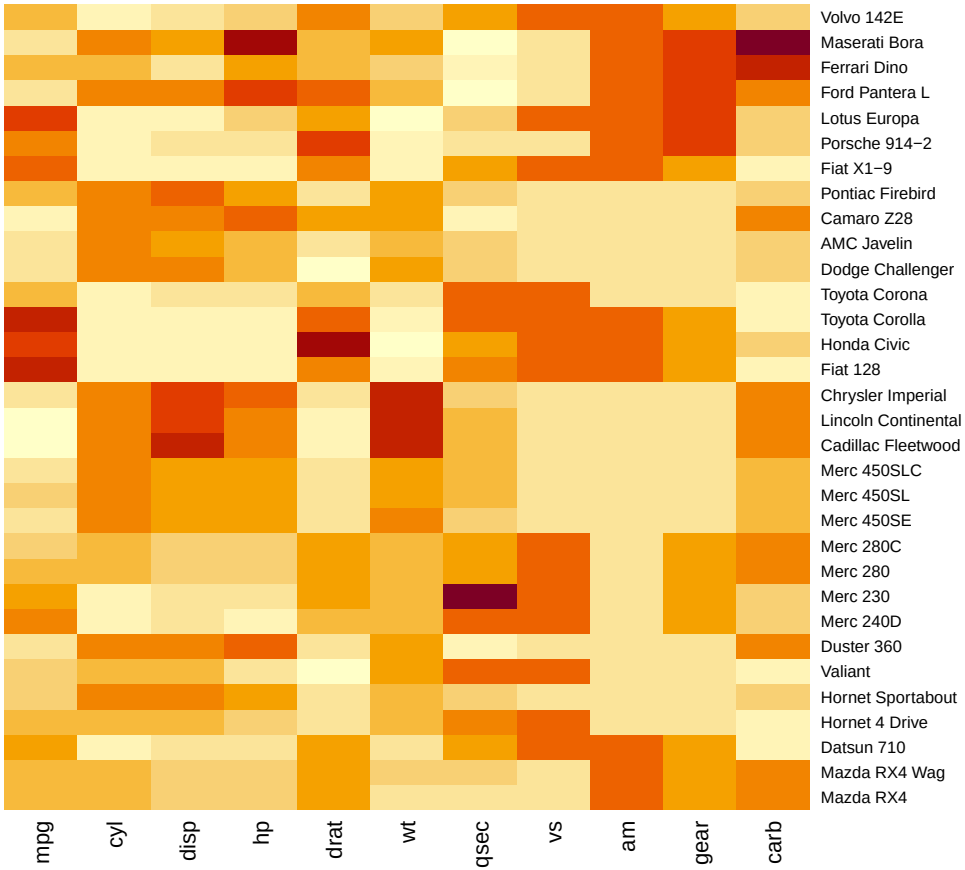
```
heatmap(data,scale="column");
```



Heat map using `heatmap()`

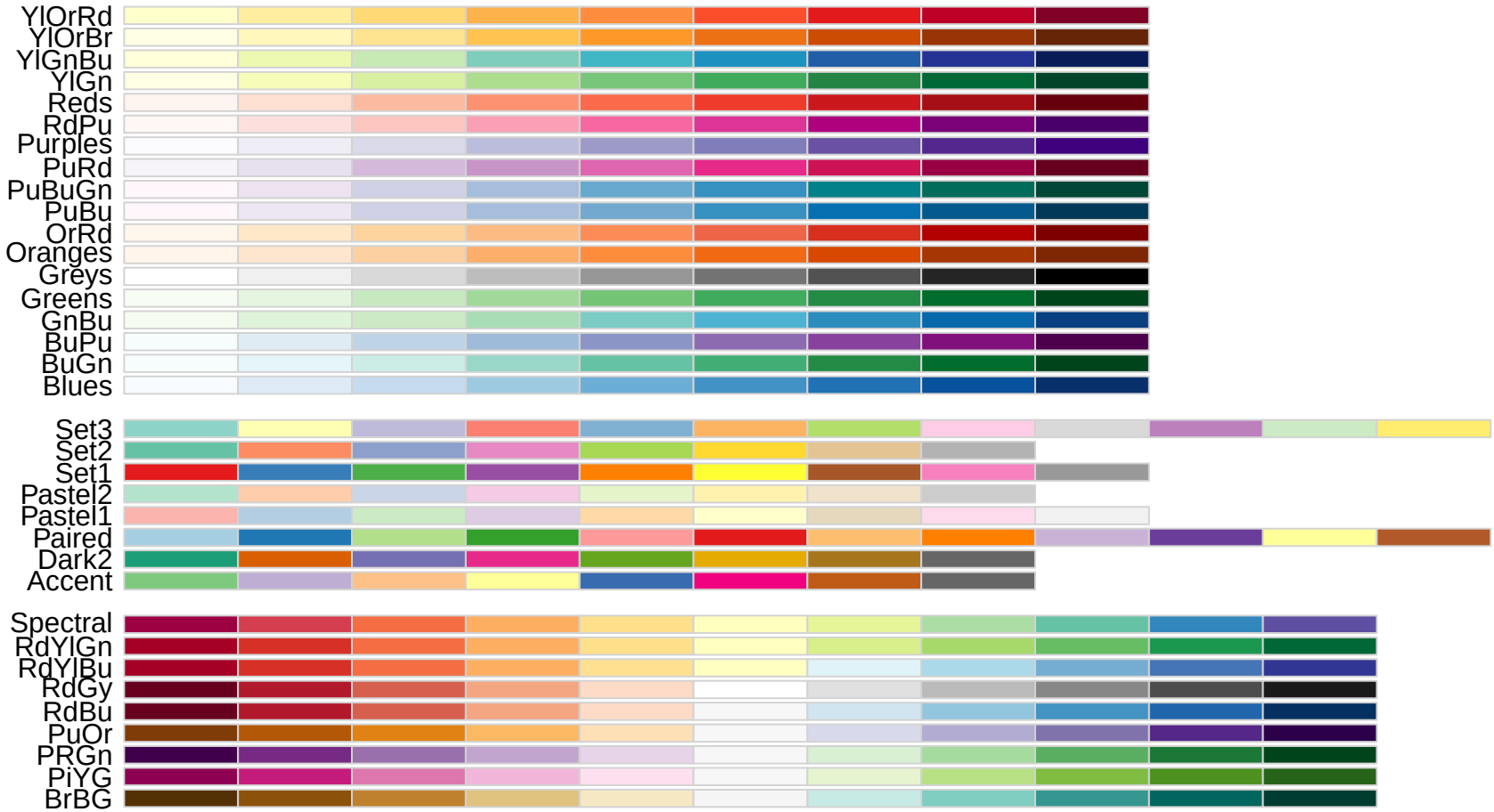
- It is noticed that order of both rows and columns is different compared to the raw mtcar matrix. This is due to cluserizations.
- To visualize the raw matrxi, we use the Rowv and Colv arguments.

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



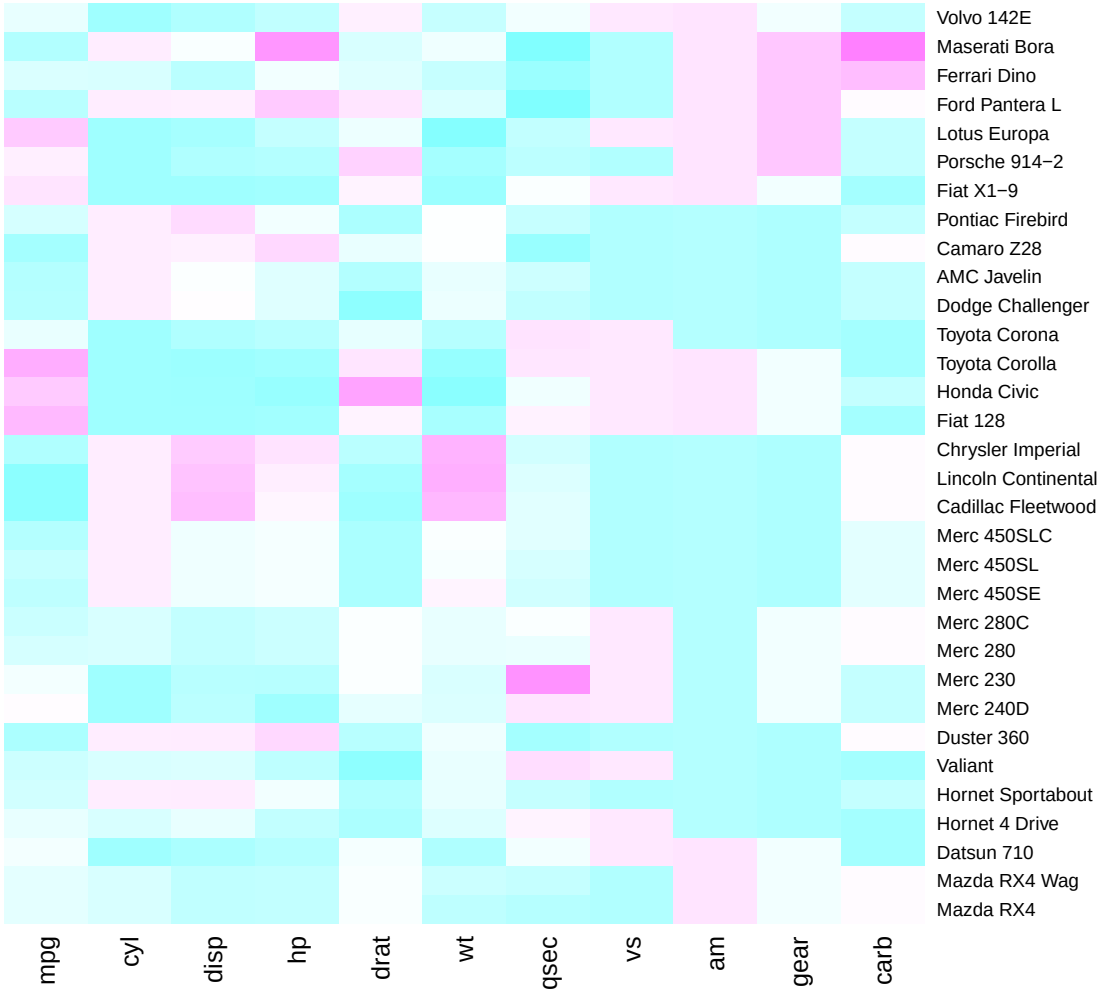
Heat map using `heatmap()`

- 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.



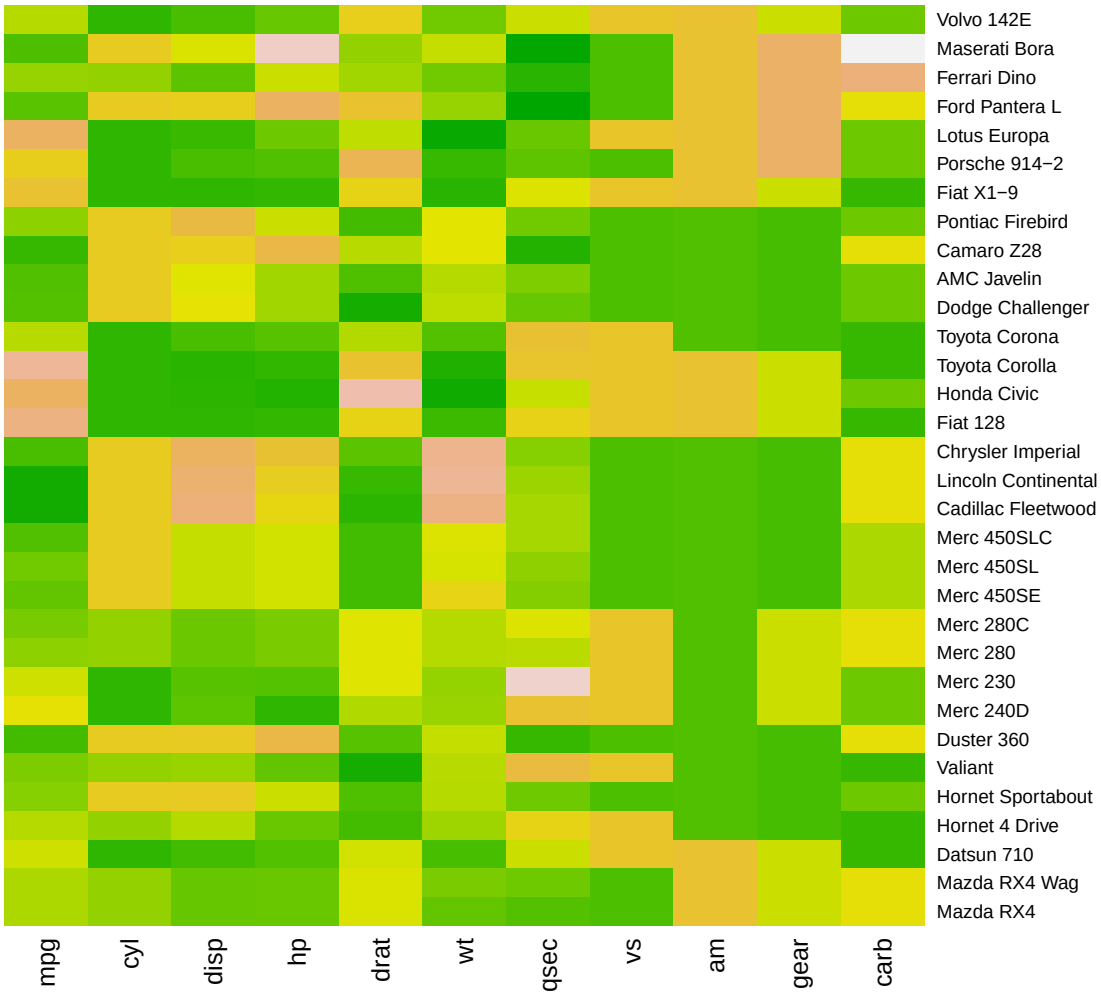
Heat map using heatmap()

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



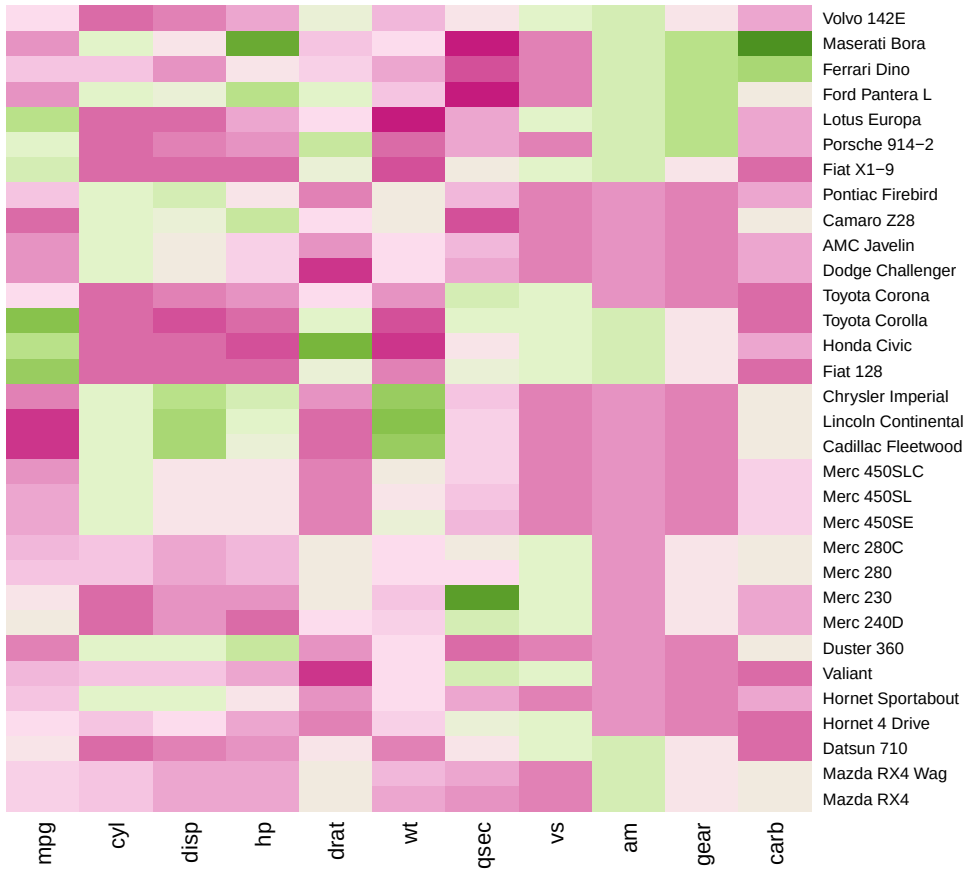
Heat map using heatmap()

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



Heat map using heatmap()

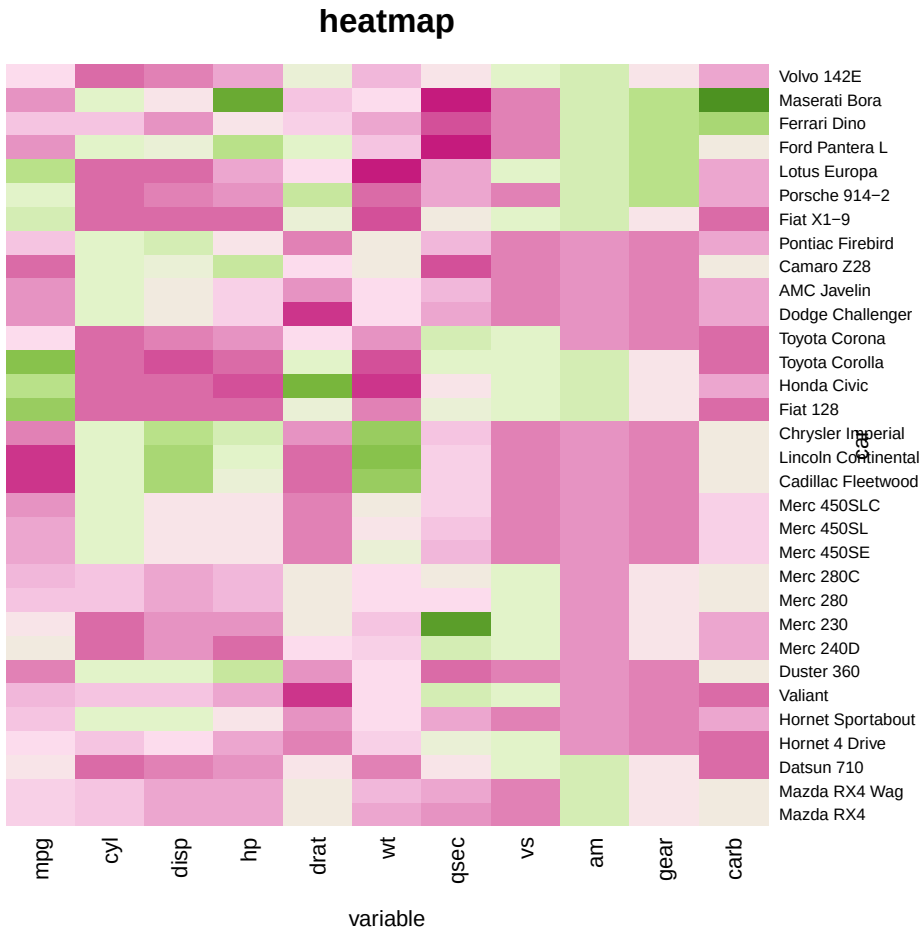
```
#Rcolorbrewer palette
library(RColorBrewer);
coul = colorRampPalette(brewer.pal(8, "PiYG"))(25);
heatmap(data, Colv = NA, Rowv = NA, scale="column", col = coul);
```



Heat map using `heatmap()`

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

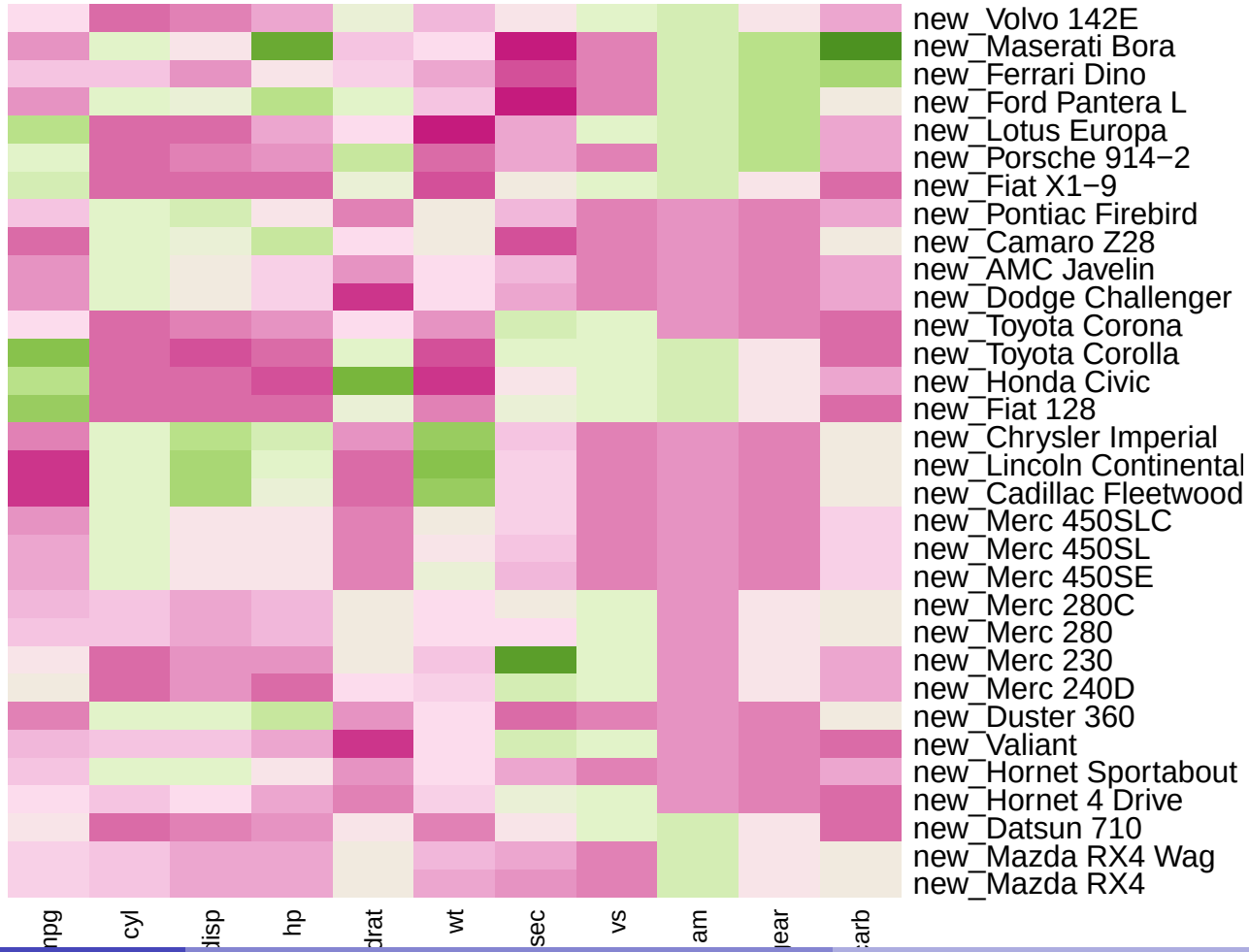
```
heatmap(data, Colv = NA, Rowv = NA, scale="column", col = coul,
        xlab="variable", ylab="car", main="heatmap");
```



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=""));
```



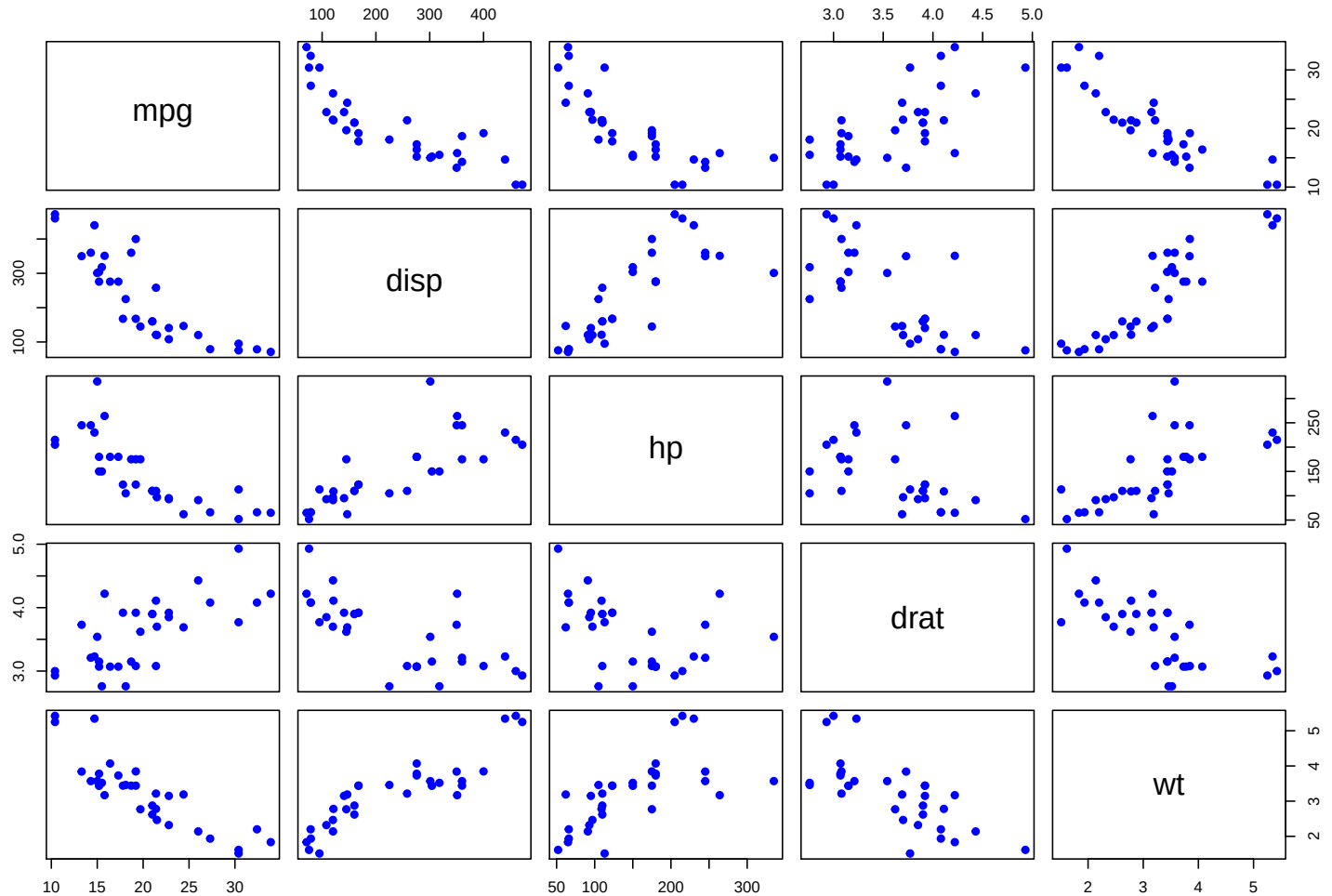
Correlogram

- A **correlogram** or correlation matrix allows to analyse the relationship between each pair of numerical variables of a matrix.
- The correlation between each pair of variable is visualise through a scatterplot, or a symbol that represents the correlation (bubble, line, number..).
- The diagonal represents the distribution of each variable, using an histogram or a density plot.

Correlogram

- Scatterplot matrices we have seen before.

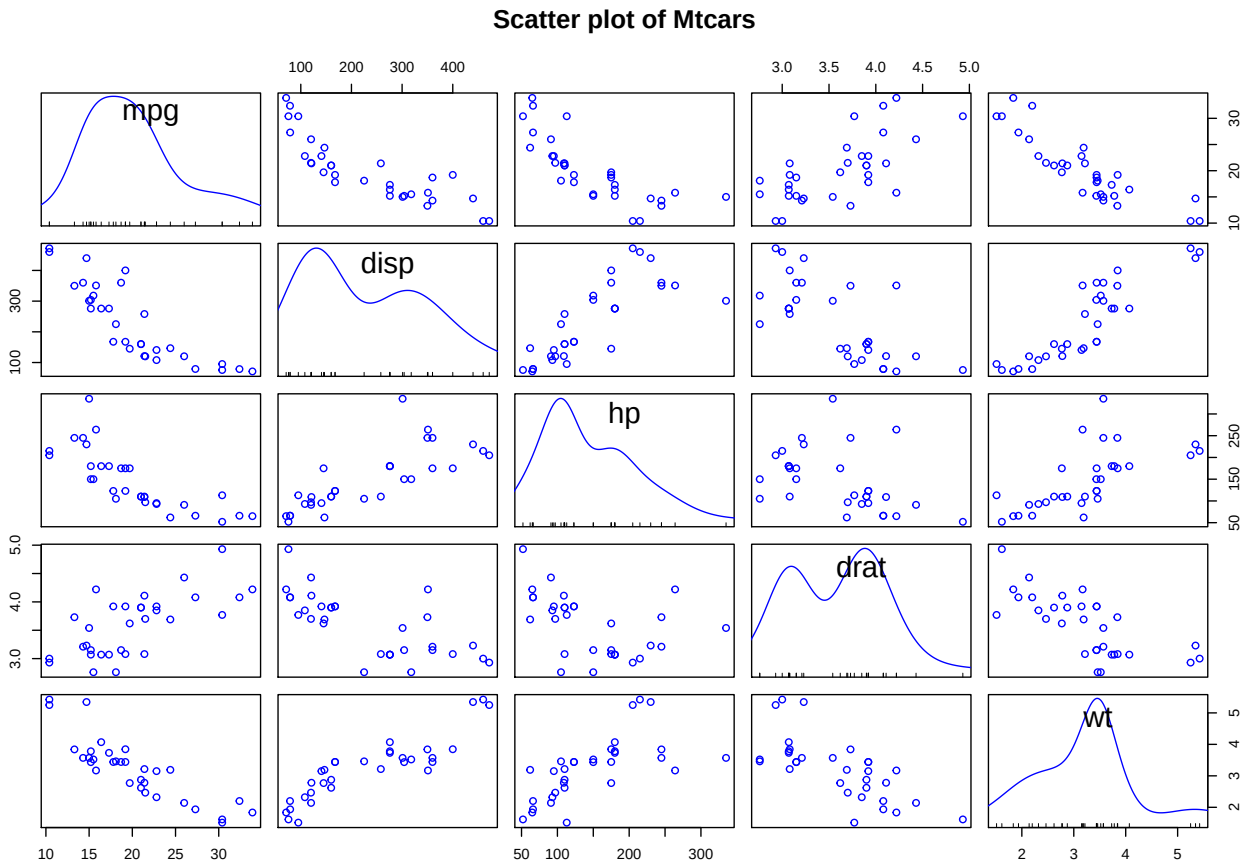
```
data2=mtcars[, c(1,3:6)];
plot(data2, pch=20, cex=1.5, col="blue");
```



Correlogram

- Scatterplot matrix using package car.

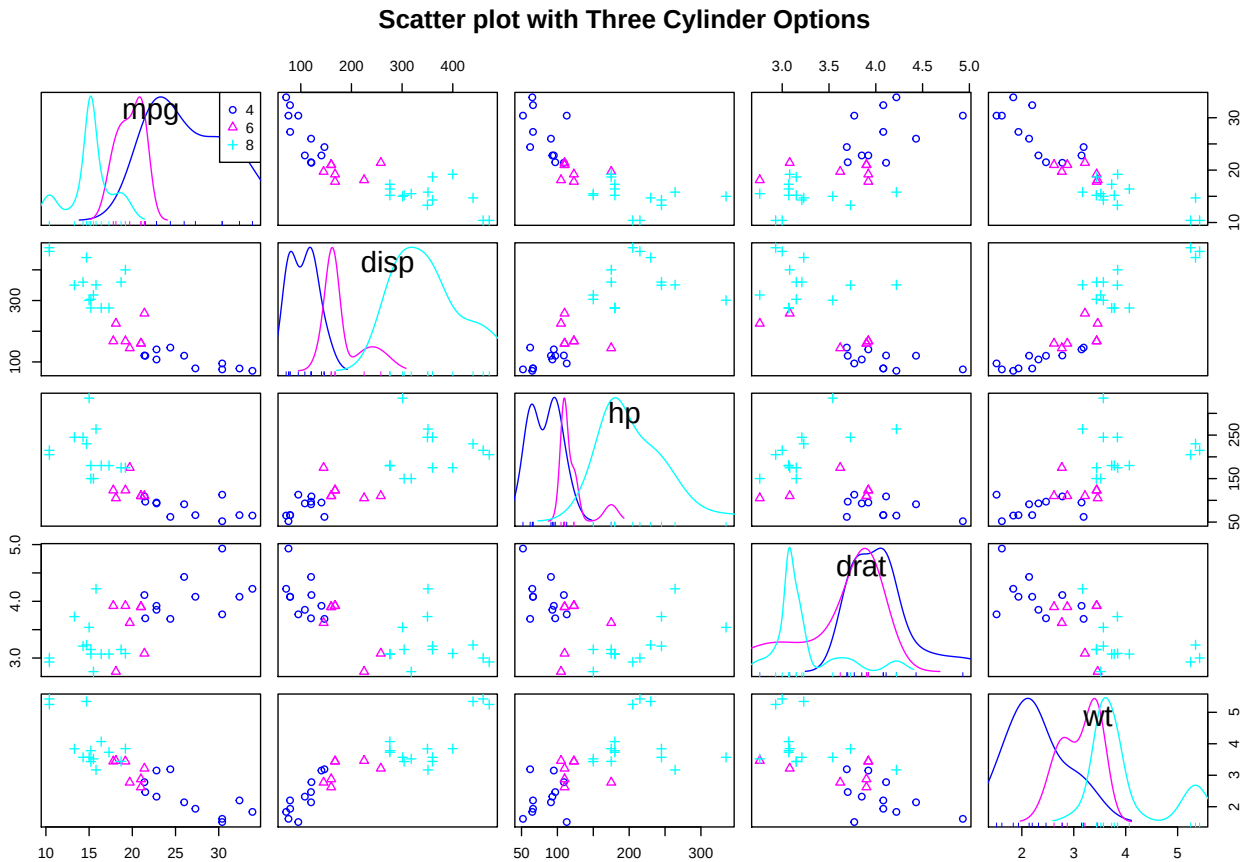
```
library(car);
scatterplotMatrix(~mpg+disp+hp+drat+wt, data=mtcars, smooth =F,
      regLine=F,main="Scatter plot of Mtcars");
```



Correlogram

- Scatterplot matrix using package car.

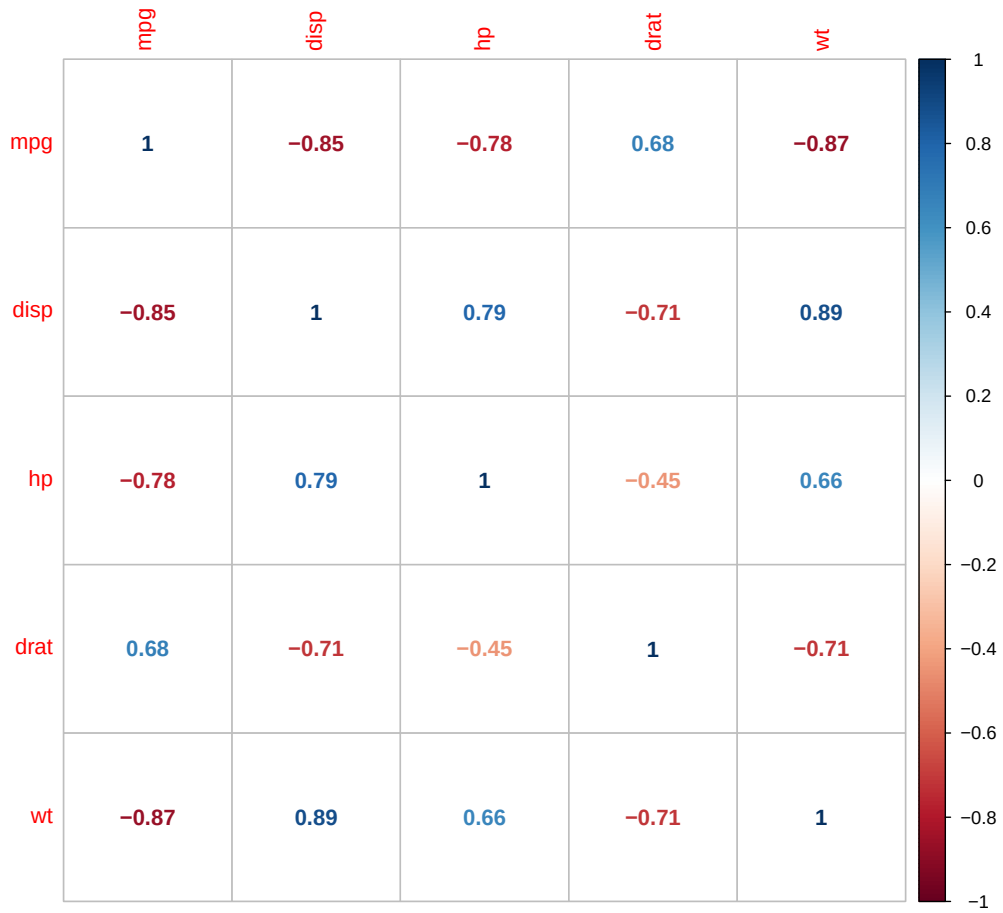
```
library(car); mtcars$cyl=as.factor(mtcars$cyl);
scatterplotMatrix(~mpg+disp+hp+drat+wt|cyl, data=mtcars, smooth =F,
      regLine=F,main="Scatter plot with Three Cylinder Options");
```



Correlogram

- Just shows the linear correlation coefficients.

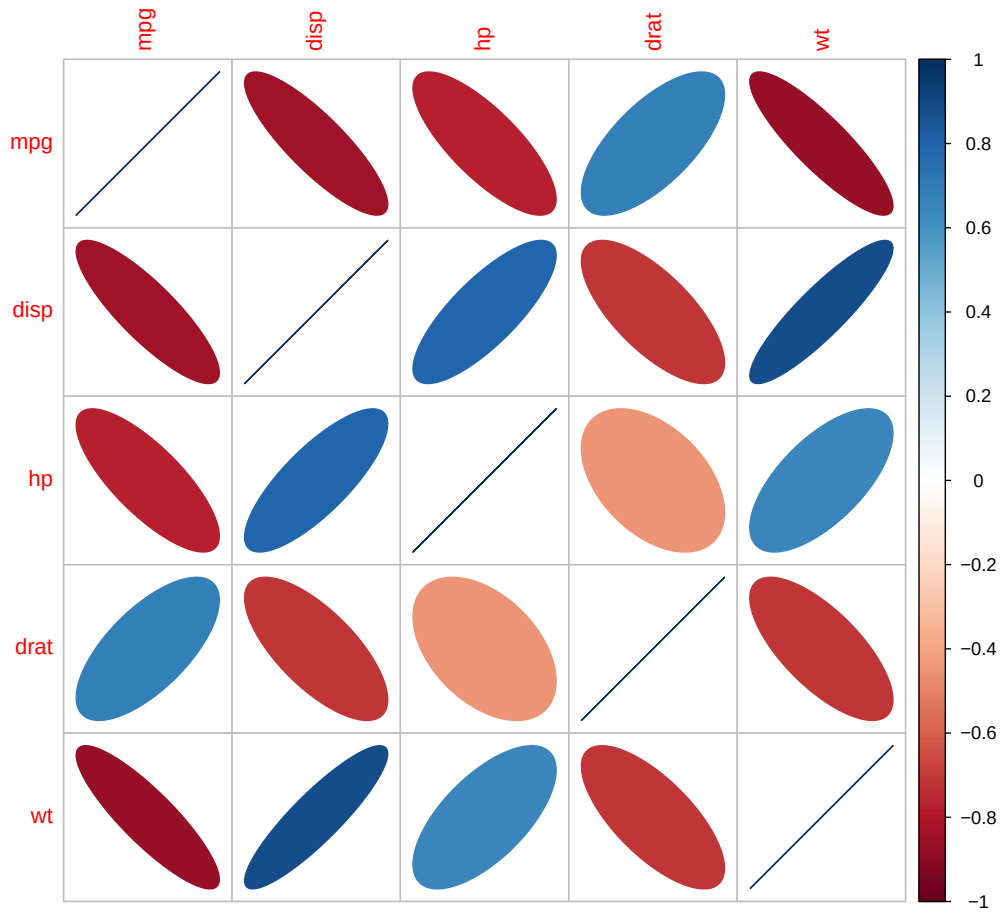
```
library(corrplot);
corrMax=cor(data2);   #calculate the linear correlations
corrplot(corrMax, method = "number");
```



Correlogram

- Show the correlations using symbols and colors.

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

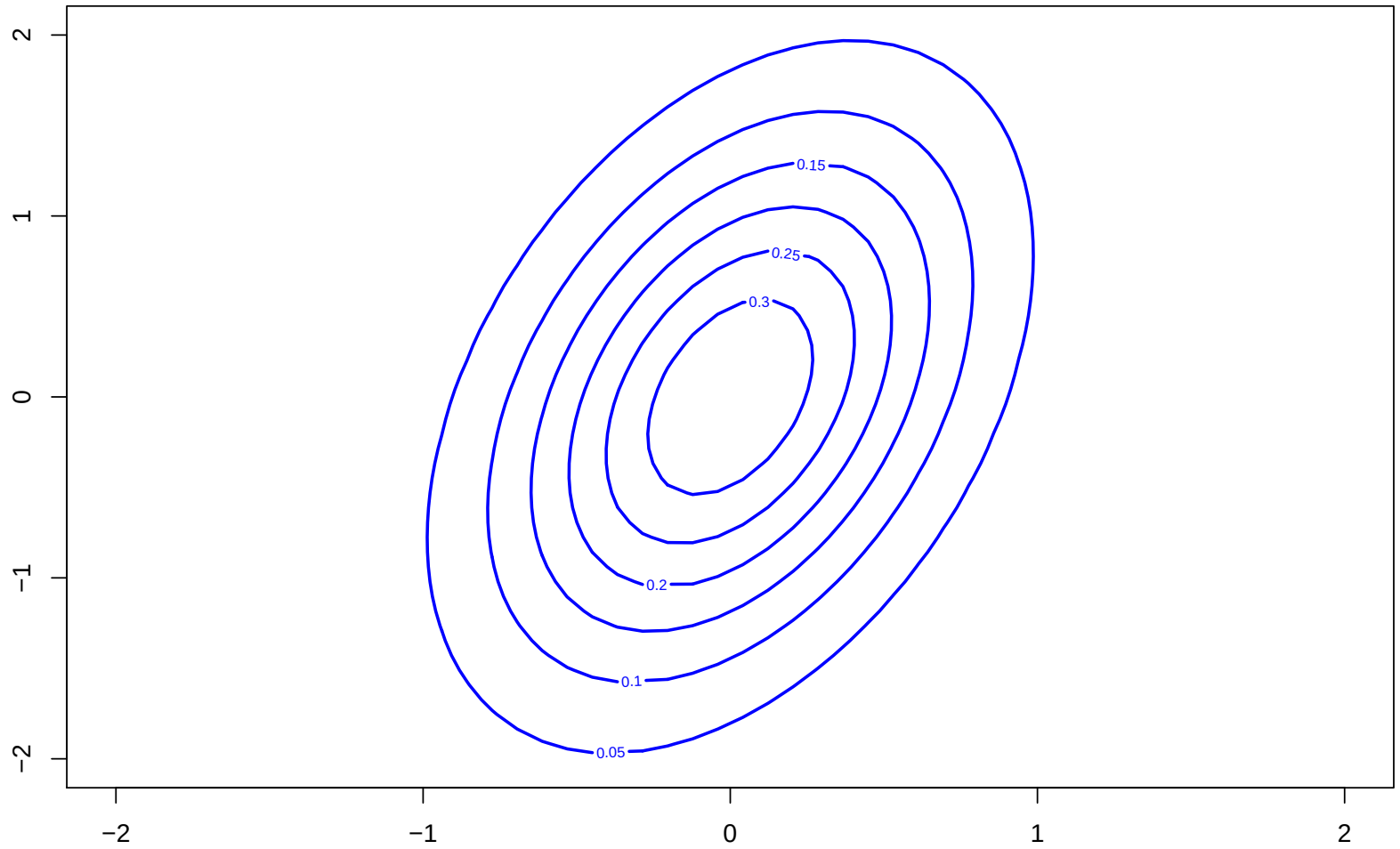


Contour plot for county population data

- A contour plot is a graphical technique for representing a 3-dimensional surface by plotting constant z slices, called contours, on a 2-dimensional format. That is, given a value for z, lines are drawn for connecting the (x, y) coordinates where that z value occurs.
- The contour plot is an alternative to a 3-D surface plot.
- The contour plot is formed by:
  - Vertical axis: Independent variable y
  - Horizontal axis: Independent variable x
  - Lines: response z values

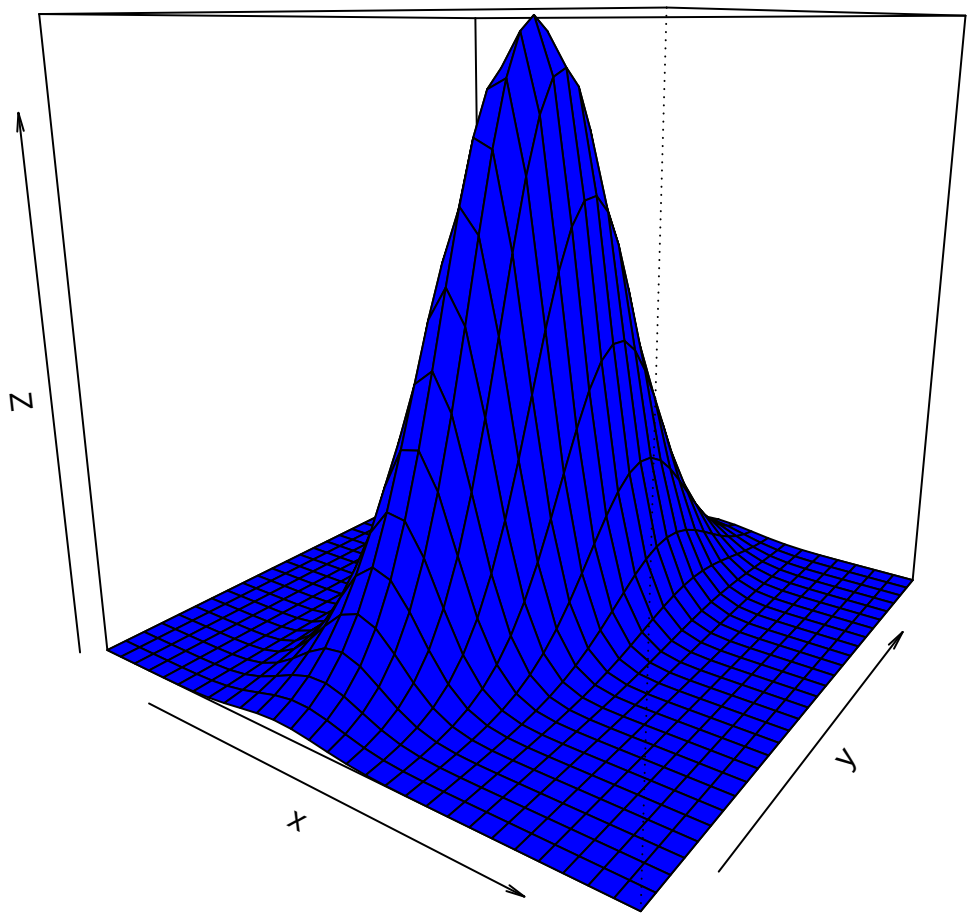
Contour plot for county population data

- The following is a contour plot of a bivariate normal distribution where the density surface is show in the next slide.



Contour plot for county population data

- Density surface of a multivariate normal distribution.



Contour plot for county population data

- Now we regard longitude and latitude as two independent variables and estimate the distribution of county populations.
- The function `kde2d` from the `MASS` package can be used to compute kernel density estimates. But it does not support the use of weights. A simple modification that does support weights is available here. Please download it to `F:/DataCamp/data`.
- Consider the county population data in 2017 only.

Contour plot for county population data

- First prepare the county population data in 2017 following the last lecture.

```
data = read.csv("F:/DataCamp/data/co-est2017-alldata.csv",
                header=TRUE, sep=",");
datapop1=data[,1:17]; #select the first 17 variables
rowsremoved=which(datapop1$CTYNAME%in%datapop1$STNAME);
datapop1 = datapop1[-rowsremoved,];
```

Contour plot for county population data

```
library(dplyr);
subdata1=dplyr::select(datapop1,STATE,COUNTY,Pop="POPESTIMATE2017");
CountyPop=dplyr::mutate(subdata1,Year=2017); #subdata for year 2017
```

```
#Then add county fips variable
CountyPop$STATE=as.character(CountyPop$STATE);
CountyPop$COUNTY=as.character(CountyPop$COUNTY);
CountyPop=mutate(CountyPop, fips=NA); #add the variable fips
CountyPop$fips=ifelse(nchar(CountyPop$COUNTY)==3,
paste(CountyPop$STATE, CountyPop$COUNTY, sep=""),
ifelse(nchar(CountyPop$COUNTY)==2,
paste(CountyPop$STATE, CountyPop$COUNTY, sep="0"),
paste(CountyPop$STATE, CountyPop$COUNTY, sep="00")));
str(CountyPop);
```

```
## 'data.frame':    3141 obs. of  5 variables:
##  $ STATE : chr  "1" "1" "1" "1" ...
##  $ COUNTY: chr  "1" "3" "5" "7" ...
##  $ Pop   : int  55504 212628 25270 22668 58013 10309 19825 114728
##  $ Year  : num  2017 2017 2017 2017 2017 ...
##  $ fips  : chr  "1001" "1003" "1005" "1007"
```

Contour plot for county population data

- In the usmap package, county fips is a character with lenth 5. We need to match the format.

```
CountyPop$fips=ifelse(nchar(CountyPop$fips)==4,
    paste("0", CountyPop$fips, sep=""), CountyPop$fips);
str(CountyPop);
```

```
## 'data.frame':    3141 obs. of  5 variables:
##  $ STATE : chr  "1" "1" "1" "1" ...
##  $ COUNTY: chr  "1" "3" "5" "7" ...
##  $ Pop   : int  55504 212628 25270 22668 58013 10309 19825 114728
##  $ Year  : num  2017 2017 2017 2017 2017 ...
##  $ fips  : chr  "01001" "01003" "01005" "01007" ...
```

Contour plot for county population data

- We need to add the long and lat information to the data. Use the data `usmap::us_map(regions = "counties"))`.

```
county_centroids = summarize(group_by(us_map(regions = "counties"),
    fips), x = mean(range(long)), y = mean(range(lat)));
str(county_centroids);
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3142 obs. of  3 vari
##  $ fips: chr  "01001" "01003" "01005" "01007" ...
##  $ x   : num  1251343 1181961 1380066 1200664 1232175 ...
##  $ y   : num  -1287672 -1495098 -1339238 -1237455 -1121131 ...
```

Contour plot for county population data

- Combine the CountyPop data and county_centroids data

```
CountyPop2017 =left_join(CountyPop, county_centroids, "fips");
str(CountyPop2017);
```

```
## 'data.frame':    3141 obs. of  7 variables:
##  $ STATE : chr  "1" "1" "1" "1" ...
##  $ COUNTY: chr  "1" "3" "5" "7" ...
##  $ Pop   : int  55504 212628 25270 22668 58013 10309 19825 114728
##  $ Year  : num  2017 2017 2017 2017 2017 ...
##  $ fips  : chr  "01001" "01003" "01005" "01007" ...
##  $ x     : num  1251343 1181961 1380066 1200664 1232175 ...
##  $ y     : num  -1287672 -1495098 -1339238 -1237455 -1121131 ...
```

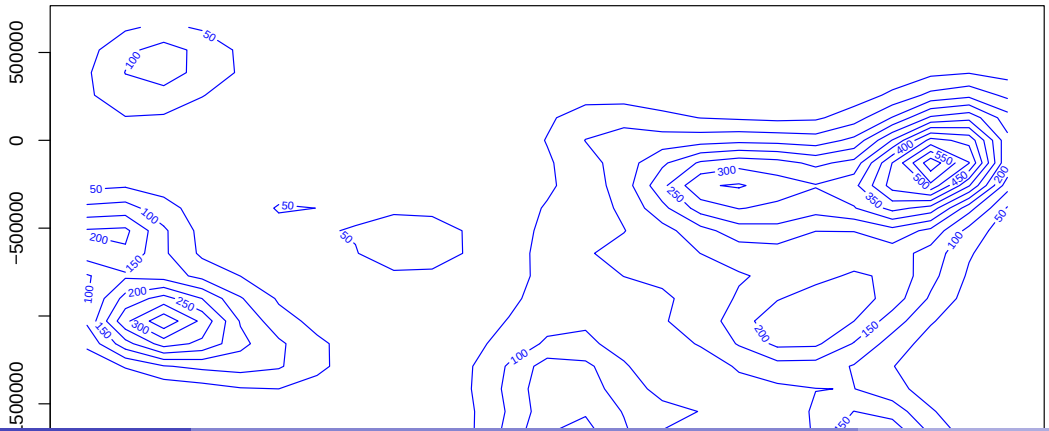
Contour plot for county population data

- We first estimate the population density on a grid over the range of the two variables, long and lat.

```
library(MASS); source("F:/DataCamp/data/kde2d.R");
ds = with(CountyPop2017, kde2d(x, y, weights=Pop*1e10));
str(ds);
```

```
## List of 3
##  $ x: num [1:25] -1979605 -1794642 -1609678 -1424714 -1239750 ...
##  $ y: num [1:25] -2443891 -2315277 -2186663 -2058048 -1929434 ...
##  $ z: num [1:25, 1:25] 0.0171 0.0766 0.1746 0.2295 0.2019 ...
```

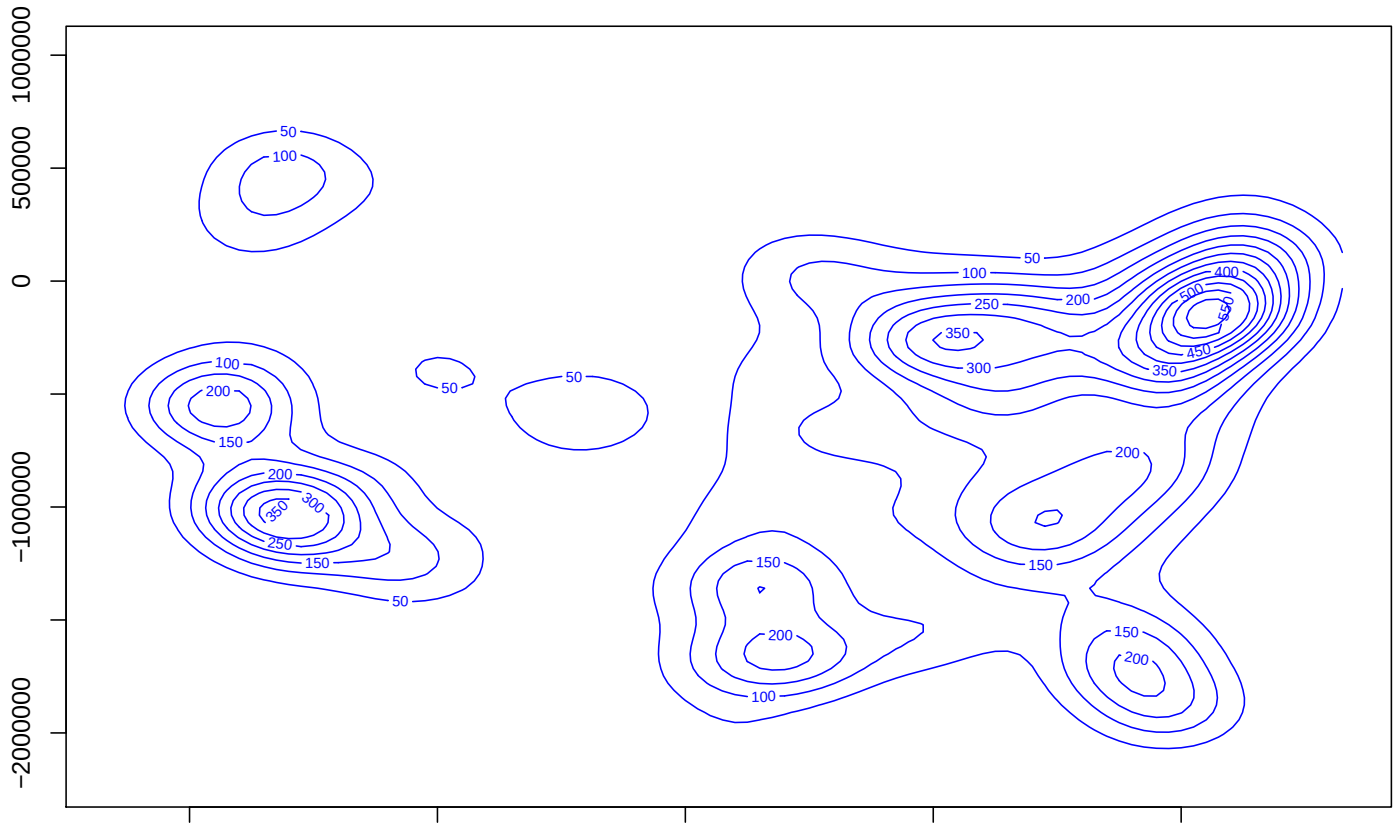
```
contour(ds,col="blue");
```



Contour plot for county population data

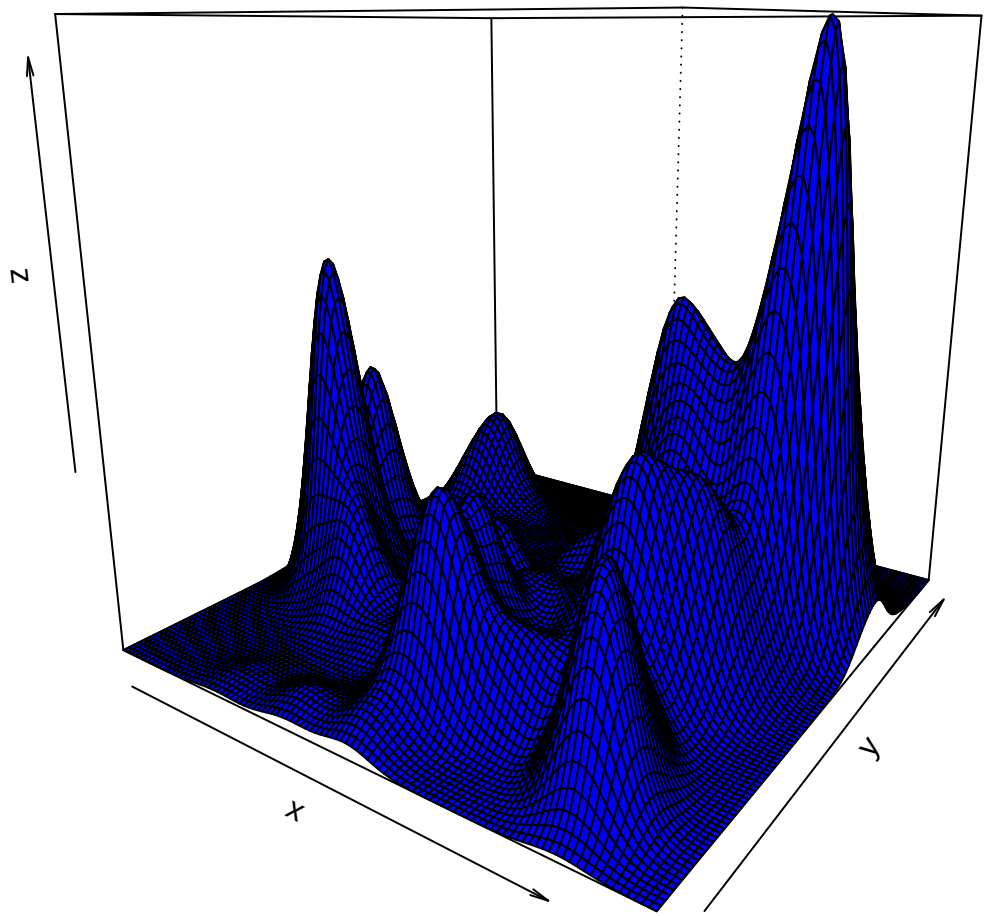
- To produce contours that work better when filled, it is useful to increase the number of grid points and enlarge the range:

```
ds = with(CountyPop2017, kde2d(x, y, weights=Pop*1e10, n=100,
          lims = c(-2.3e6,2.65e6, -2.2e6, 1e6)) );
contour(ds,col="blue");
```



Contour plot for county population data

- The following is the density suface of the couty population data.

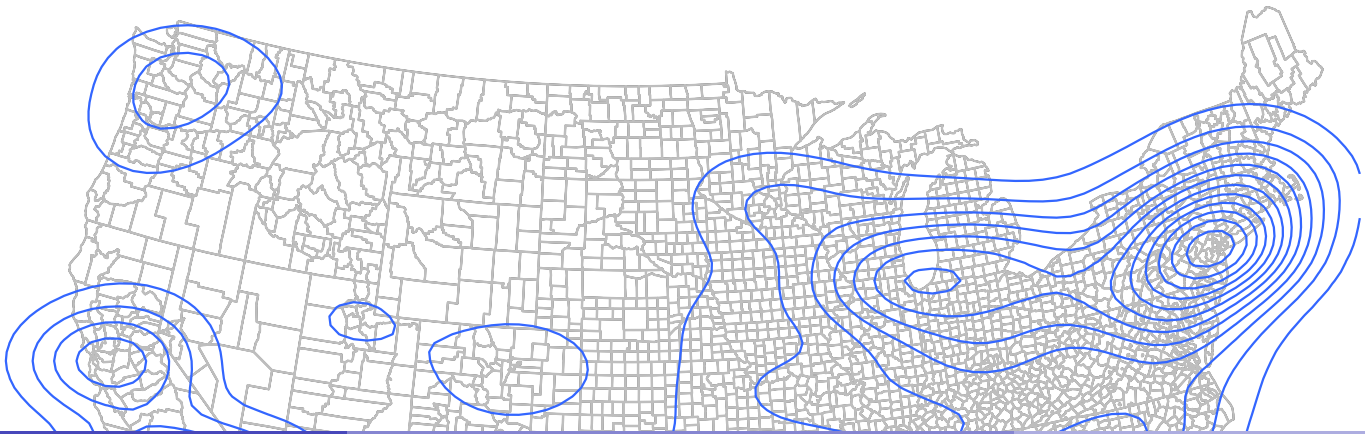


Contour plot for county population data

- We use the ggplot2 approach: `stat_contour()` function to plot the map and contour plot in the same graph.
- The ggplot approach requires the density is tidy form:

```
dsrfc = expand.grid(long = ds$x, lat = ds$y);
#Create a data frame from all combinations of factor variables
dsrfc$dens = as.vector(ds$z);
```

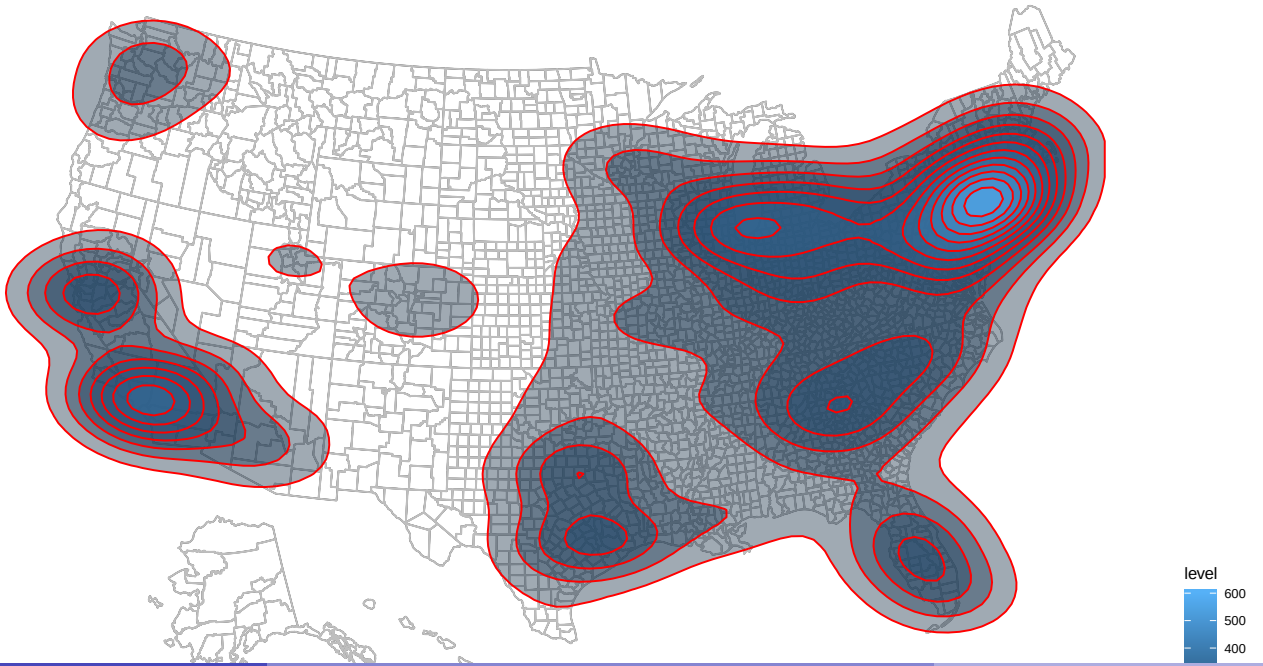
```
library(usmap);
plot_usmap(regions = "counties")+
  geom_polygon(aes(long,lat,group=group),fill=NA,color="grey")+
  stat_contour(aes(long,lat, z=dens),data=dsrfc);
```



Contour plot for county population data

- A filled contour version can be created using `stat_contour` and `fill = level`.

```
plot_usmap(regions = "counties")+
  geom_polygon(aes(long,lat,group=group),fill=NA,color="grey")+
  stat_contour(aes(long,lat, z=dens,fill = ..level..),
      data=dsrfc,geom = "polygon", alpha=0.4,col="red")+
  # alpha specifies the transparency level
  theme(legend.position = "right");
```



Questions?