# Data Visualization with R

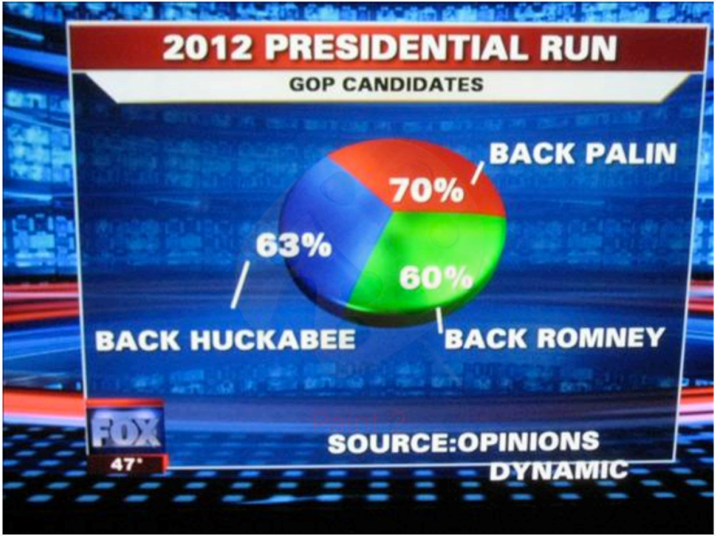
Dhafer Malouche

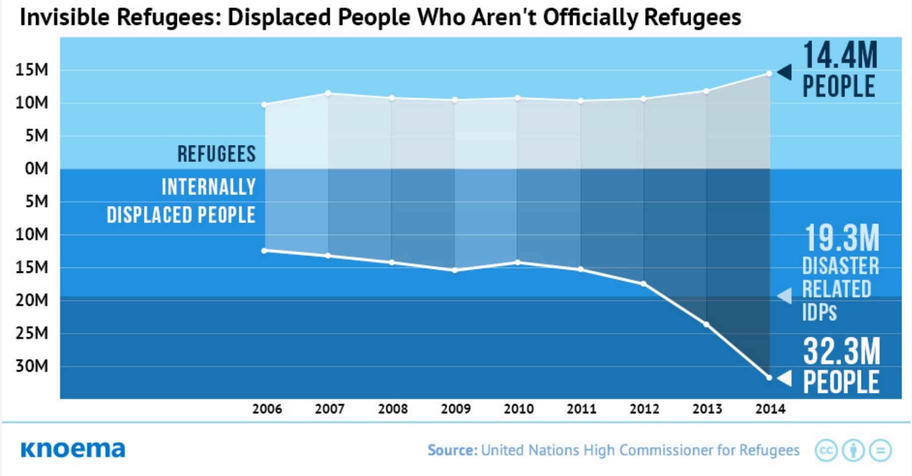
essai.academia.edu/DhaferMalouche

Center of Political Studies,
Institute of Social Research
University of Michigan

Ecole Supérieure de la Statistique
et de l'Analyse de l'Information,
University of Carthage

March 29th, 2017, 12:00-1:30 PM 5670 and 5769 Haven Hall
Department of Political Science, University of Michigan


2012 PRESIDENTIAL RUN
GOP CANDIDATES
BACK PALIN
70%
63%
60%
BACK HUCKABEE
BACK ROMNEY
FOX
47°
SOURCE:OPINIONS
DYNAMIC




Source: Knoema website

R package: `Knoema` on `Github`

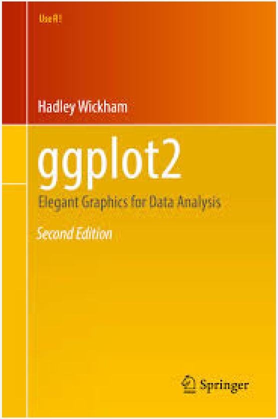
# Outline

1. R packages
   - ggplot2
   - sjPlot
   - tabplot
2. Visualizing multivariate:
   - Categorical Data
   - Quantitative Data
3. Visualizing Data with target variable and results of statistical models.

## R packages

- `ggplot2`, programming graphs
- `sjPlot`, for Social Scientists
- `fsmb`, Radar Charts
- `tabplot`, Large data

# ggplot2




**Hadley Wickham, 2005**

# ggplot2

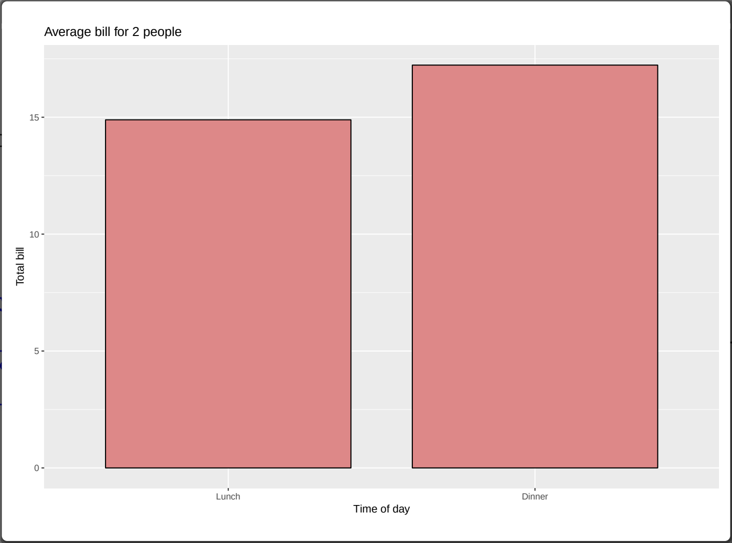
```
> dat <- data.frame(
+   time = factor(c("Lunch","Dinner"), levels=c("Lunch","Dinner")),
+   total_bill = c(14.89, 17.23)
+ )
> dat
    time total_bill
1  Lunch      14.89
2 Dinner      17.23
```

# ggplot2

```
> dat <- data.frame(
+   time = factor(c("Lunch","Dinner"), levels=c("Lunch","Dinner")),
+   total_bill = c(14.89, 17.23)
+ )
> dat
    time total_bill
1  Lunch      14.89
2 Dinner      17.23

> library(ggplot2)
> ggplot(data=dat, aes(x=time, y=total_bill, fill=time)) +
+   geom_bar(colour="black", fill="#DD8888", width=.8, stat="identity") +
+   guides(fill=FALSE) +
+   xlab("Time of day") + ylab("Total bill") +
+   ggtitle("Average bill for 2 people")
```

# ggplot2

```
> dat <-
+   time
+   total
+ )
> dat
    time
1  Lunch
2 Dinner

> library
> ggplot
+   geom
+   guide
+   xlab
+   ggti
```



# ggplot2

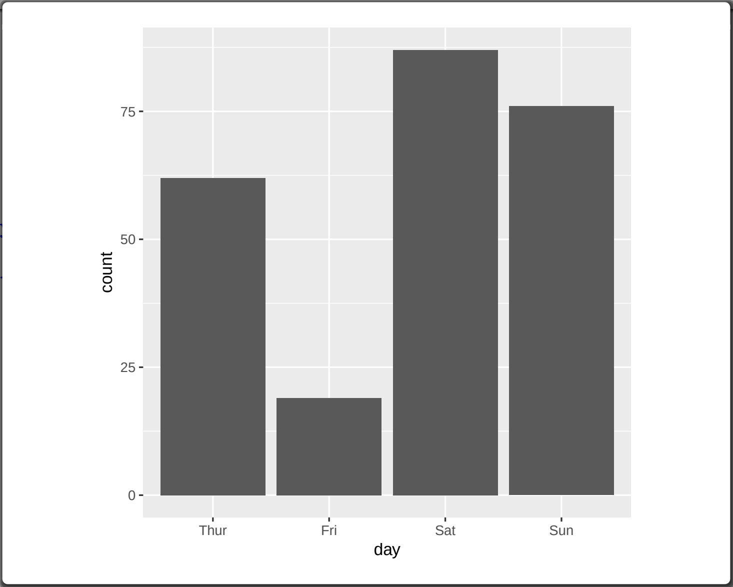
```
> library(reshape2)
> data(tips)
> head(tips)
  total_bill  tip    sex smoker day   time size
1      16.99 1.01 Female     No Sun Dinner    2
2      10.34 1.66   Male     No Sun Dinner    3
3      21.01 3.50   Male     No Sun Dinner    3
4      23.68 3.31   Male     No Sun Dinner    2
5      24.59 3.61 Female     No Sun Dinner    4
6      25.29 4.71   Male     No Sun Dinner    4
> levels(tips$day)
[1] "Fri"  "Sat"  "Sun"  "Thur"
> tips$day=factor(tips$day,levels=levels(tips$day)[c(4,1,2,3)])
```

# ggplot2

```
> library(ggplot2)
> ggplot(data=tips, aes(x=day)) +
+   geom_bar(stat="count")
```

# ggplot2

```
> librar
> ggplot
+   geom
```



# ggplot2

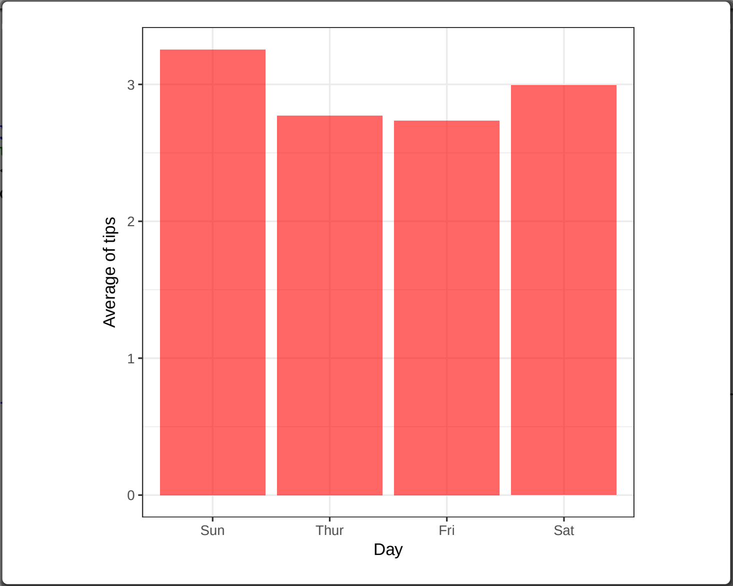
```
> library(plyr)
> # Calculate the mean of tip for each day
> mtips <- ddply(tips, "day", summarise, mtip = mean(tip))
> mtips$day=factor(mtips$day,levels=levels(mtips$day)[c(4,1,2,3)])
> mtips
   day     mtip
1 Thur 2.771452
2  Fri 2.734737
3  Sat 2.993103
4  Sun 3.255132
```

# ggplot2

```
> library(plyr)
> # Calculate the mean of tip for each day
> mtips <- ddply(tips, "day", summarise, mtip = mean(tip))
> mtips$day=factor(mtips$day,levels=levels(mtips$day)[c(4,1,2,3)])
> mtips
   day     mtip
1 Thur 2.771452
2  Fri 2.734737
3  Sat 2.993103
4  Sun 3.255132

> ggplot(data=mtips, aes(x=day,y=mtip)) +
+   geom_bar(stat="identity",fill="red",alpha=.6)+theme_bw()+xlab("Day")+
+   ylab("Average of tips")
```

# ggplot2

```
> library
> # Calc
> mtips <
> mtips$
> mtips
   day
1 Thur 2
2  Fri 2
3  Sat 2
4  Sun 3

> ggplot
+   geom
+   ylab
```



# ggplot2

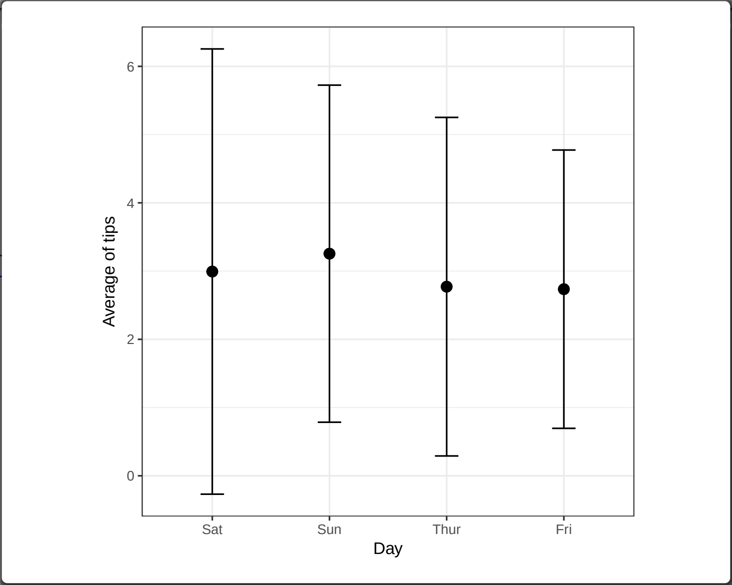
```
> library(plyr)
> # Calculate the mean of tip for each day
> mtips <- ddply(tips, "day", summarise, mtip = mean(tip),stip=sd(tip))
> mtips$day=factor(mtips$day,levels=levels(mtips$day)[c(4,1,2,3)])
> mtips$lower=mtips$mtip-2*mtips$stip
> mtips$upper=mtips$mtip+2*mtips$stip
> mtips$day=factor(mtips$day,levels=levels(mtips$day)[c(4,1,2,3)])
> mtips
   day     mtip     stip      lower    upper
1 Thur 2.771452 1.240223  0.2910052 5.251898
2  Fri 2.734737 1.019577  0.6955827 4.773891
3  Sat 2.993103 1.631014 -0.2689252 6.255132
4  Sun 3.255132 1.234880  0.7853710 5.724892
```

# ggplot2

```
> ggplot(mtips,aes(x=day,y=mtip,group=day))+
+   geom_errorbar(aes(ymin=lower,ymax=upper,width=.2))+
+   geom_point(size=3)+theme_bw()+xlab("Day")+ylab("Average of tips")
```

# ggplot2

```
> ggplot
+    geom
+    geom
```



# ggplot2

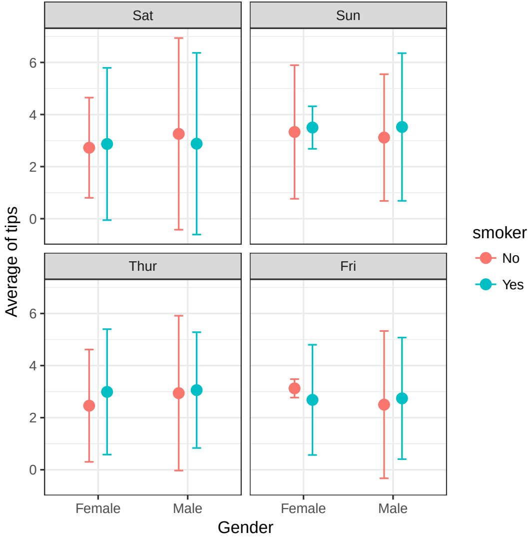
```
> library(plyr)
> # Calculate the mean of tip for each day
> mtips <- ddply(tips, c("day","sex","smoker"), summarise, mtip = mean
+                (tip),stip=sd(tip))
> mtips$day=factor(mtips$day,levels=levels(mtips$day)[c(4,1,2,3)])
> mtips$lower=mtips$mtip-2*mtips$stip
> mtips$upper=mtips$mtip+2*mtips$stip
> mtips$day=factor(mtips$day,levels=levels(mtips$day)[c(4,1,2,3)])
> mtips
    day    sex smoker     mtip      stip       lower    upper
1  Thur Female     No 2.459600 1.0783687  0.30286265 4.616337
2  Thur Female    Yes 2.990000 1.2040487  0.58190255 5.398097
3  Thur   Male     No 2.941500 1.4856233 -0.02974659 5.912747
4  Thur   Male    Yes 3.058000 1.1115735  0.83485308 5.281147
5   Fri Female     No 3.125000 0.1767767  2.77144661 3.478553
6   Fri Female    Yes 2.682857 1.0580125  0.56683212 4.798882
7   Fri   Male     No 2.500000 1.4142136 -0.32842712 5.328427
8   Fri   Male    Yes 2.741250 1.1668081  0.40763386 5.074866
9   Sat Female     No 2.724615 0.9619045  0.80080640 4.648424
10  Sat Female    Yes 2.868667 1.4613783 -0.05409002 5.791423
11  Sat   Male     No 3.256562 1.8397486 -0.42293469 6.936060
12  Sat   Male    Yes 2.879259 1.7443379 -0.60941660 6.367935
13  Sun Female     No 3.329286 1.2823564  0.76457293 5.893998
14  Sun Female    Yes 3.500000 0.4082483  2.68350342 4.316497
15  Sun   Male     No 3.115349 1.2164005  0.68254779 5.548150
16  Sun   Male    Yes 3.521333 1.4174316  0.68647010 6.356197
```

# ggplot2

```
> pd <- position_dodge(0.4)
> ggplot(mtips,aes(x=sex,y=mtip,col=smoker,group=smoker))+
+   geom_errorbar(aes(ymin=lower,ymax=upper),position=pd,width=.2)+
+   geom_point(size=3,position=pd)+theme_bw()+xlab("Gender")+
+   ylab("Average of tips")+facet_wrap(~day)
```

# ggplot2

```
> pd <-
> ggplot
+   geom
+   geom
+   ylab
```

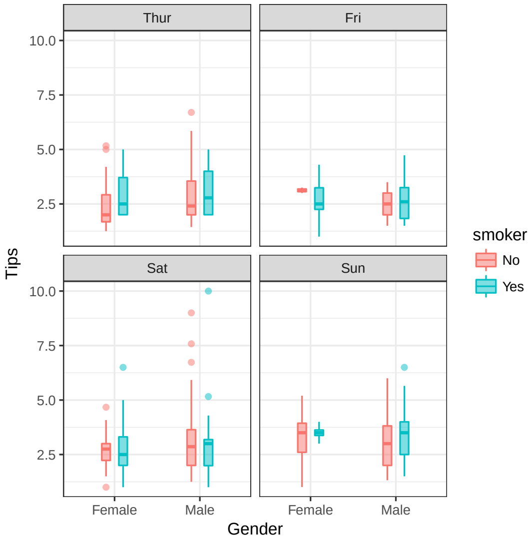


# ggplot2

```
> ggplot(tips,aes(x=sex,y=tip,col=smoker,fill=smoker))+
+   geom_boxplot(position=pd,width=.2,alpha=.5)+theme_bw()+xlab("Gender")+
+   ylab("Tips")+facet_wrap(~day)
```

# ggplot2

```
> ggplot
+   geom
+   ylab
```
+

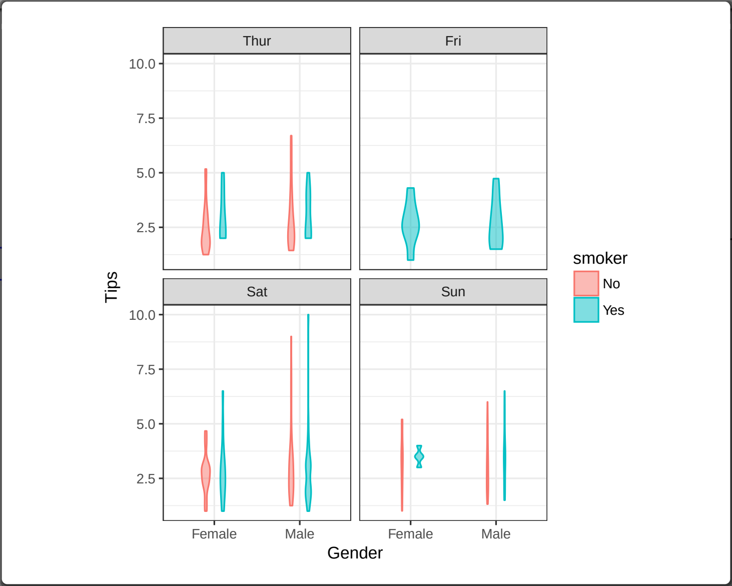


# ggplot2

```
> ggplot(tips,aes(x=sex,y=tip,col=smoker,fill=smoker))+
+   geom_violin(position=pd,width=.2,alpha=.5)+theme_bw()+xlab("Gender")+
+   ylab("Tips")+
+   facet_wrap(~day)
```

# ggplot2

```
> ggplot
+   geom_
+   ylab
+   facet
```

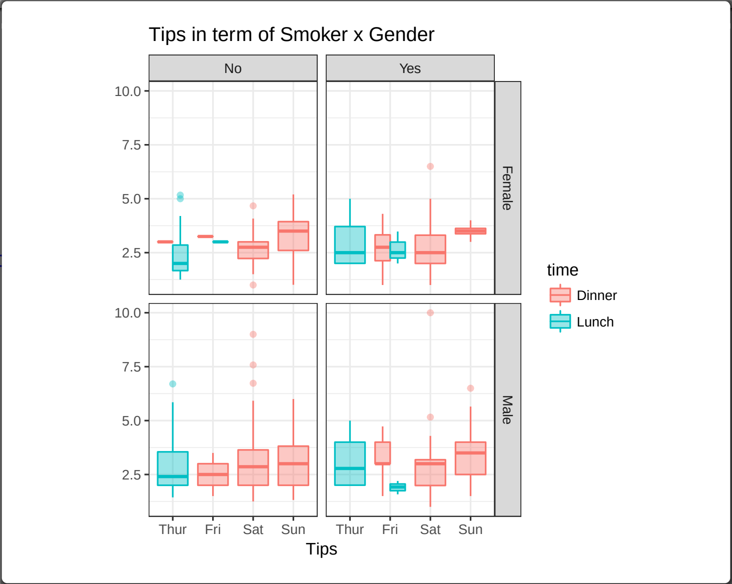


# ggplot2

```
> ggplot(tips,aes(x=day,y=tip,col=time,fill=time))+
+   geom_boxplot(alpha=.4)+theme_bw()+xlab("Tips")+ylab("")+
+   facet_grid(sex~smoker)+ggtitle("Tips in term of Smoker x Gender")
```

## ggplot2

```
> ggplot
+   geom
+   face
```

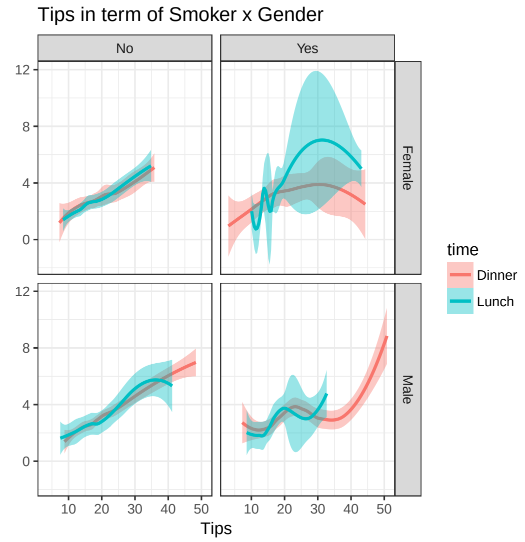


# ggplot2

```
> ggplot(tips,aes(x=total_bill,y=tip,col=time,fill=time))+
+   geom_smooth(alpha=.4)+theme_bw()+xlab("Tips")+ylab("")+
+   facet_grid(sex~smoker)+ggtitle("Tips in term of Smoker x Gender")
```

# ggplot2

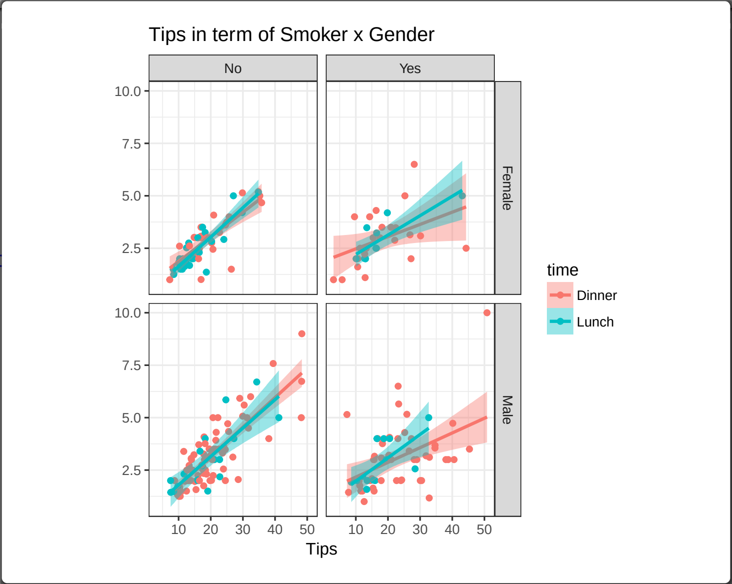
```
> ggplot
+   geom
+   face
```



# ggplot2

```
> ggplot(tips,aes(x=total_bill,y=tip,col=time,fill=time))+geom_point()+
+   geom_smooth(method='lm',alpha=.4)+theme_bw()+xlab("Tips")+ylab("")+
+   facet_grid(sex~smoker)+ggtitle("Tips in term of Smoker x Gender")
```

# ggplot2




```
> ggplot
+   geom
+   face
```

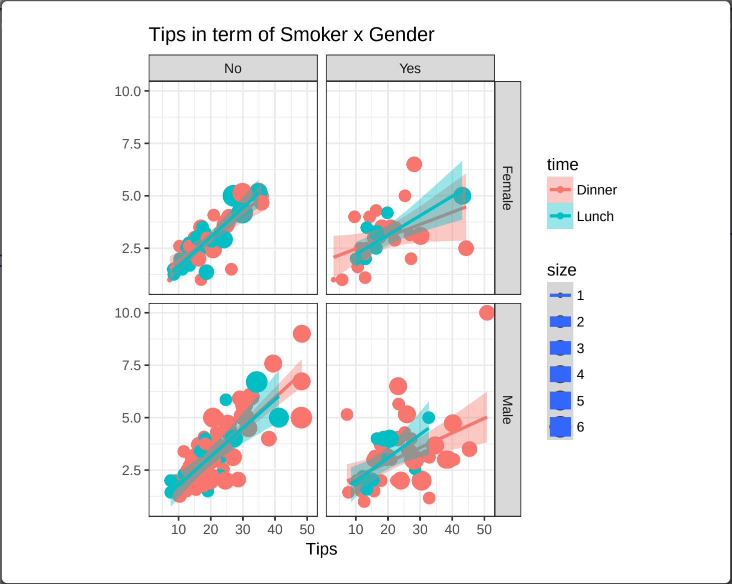
# ggplot2

```
> ggplot(tips,aes(x=total_bill,y=tip,col=time,fill=time,size=size))+geom_point()+
+   geom_smooth(method='lm',alpha=.4)+theme_bw()+xlab("Tips")+ylab("")+
+   facet_grid(sex~smoker)+ggtitle("Tips in term of Smoker x Gender")
```

# ggplot2

```
> ggplot
+   geom
+   face
```

.point()+



# GUI for ggplot2

# GUI for ggplot2

**`Rcmdr, RmcdrPlugin.KMggplot2`**

# GUI for ggplot2

**JGR, Deducer...**

# `sjPlot`

# sjPlot

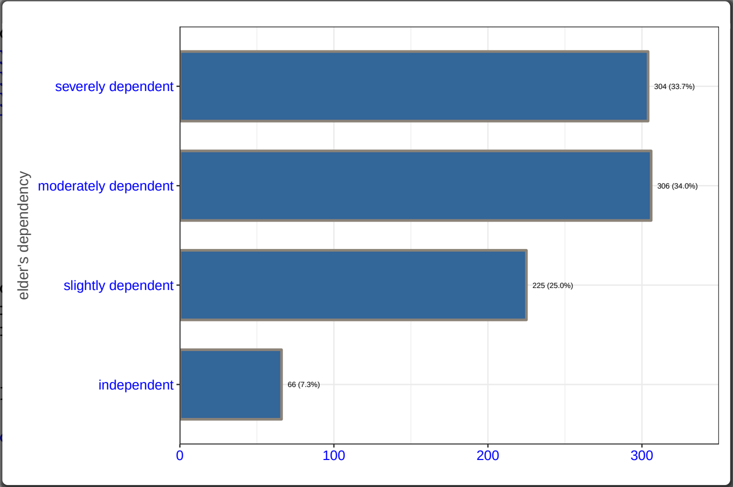
- Author: Daniel Lüdecke d.luedecke@uke.de
- Website: http://www.strengejacke.de/sjPlot/
- It's a Data Visualization package for Statistics in Social Science
- It contains functions to import data from different formats: SPSS, STATA, SAS. . . etc.
- Labeling and handling factor variables in the data.

# sjPlot, Bar charts

```
> ## Load the package and define your theme (there are a lot...).
> library(sjPlot)
> library(sjmisc)
> library(ggplot2)
> sjp.setTheme(geom.outline.color = "antiquewhite4",
+              geom.outline.size = 1,
+              geom.label.size = 2,
+              geom.label.color = "black",
+              title.color = "red",
+              title.size = 1.5,
+              axis.textcolor = "blue",
+              base = theme_bw())
> ## Load data and represent the bar chart of one the variables.
> data(efc)
> attr(efc$e42dep,"labels")
       independent   slightly dependent moderately dependent
                 1                    2                    3
  severely dependent
                 4
> sjp.frq(efc$e42dep,coord.flip = T,geom.size = .4)
```

# sjPlot, Bar charts

```
> ## Loa
> librar
> librar
> librar
> sjp.se
+
+
+
+
+
+
+
> ## Loa
> data(e
> attr(e

  severe

> sjp.fr
```

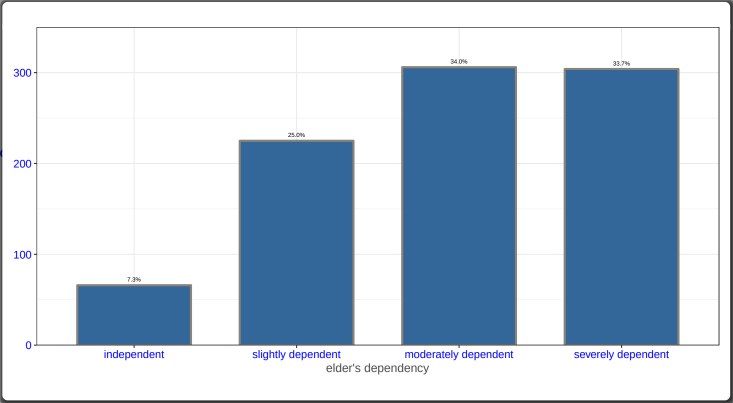


# sjPlot, Bar charts

```
> sjp.frq(efc$e42dep,show.prc = T,show.n = F)
```

# sjPlot, Bar charts

```
> sjp.fr
```



# sjPlot, Contingency tables

```
> xtabs(~efc$e16sex+efc$e42dep)
          efc$e42dep
efc$e16sex   1   2   3   4
         1  23  70 109  93
         2  43 154 197 211
```

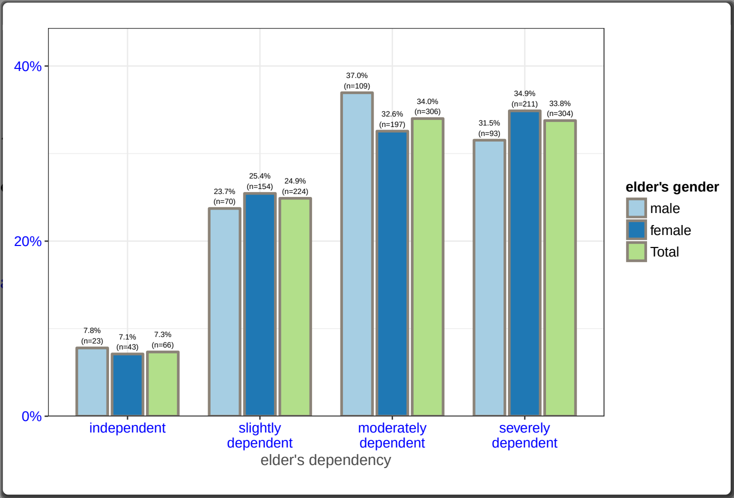
# sjPlot, Contingency tables

```
> xtabs(~efc$e16sex+efc$e42dep)
           efc$e42dep
efc$e16sex   1   2   3   4
         1  23  70 109  93
         2  43 154 197 211
```

```
> sjp.xtab(x = efc$e42dep, grp = efc$e16sex)
```

# sjPlot, Contingency tables

```
> xtabs(
efc$e16s
> sjp.xt
```

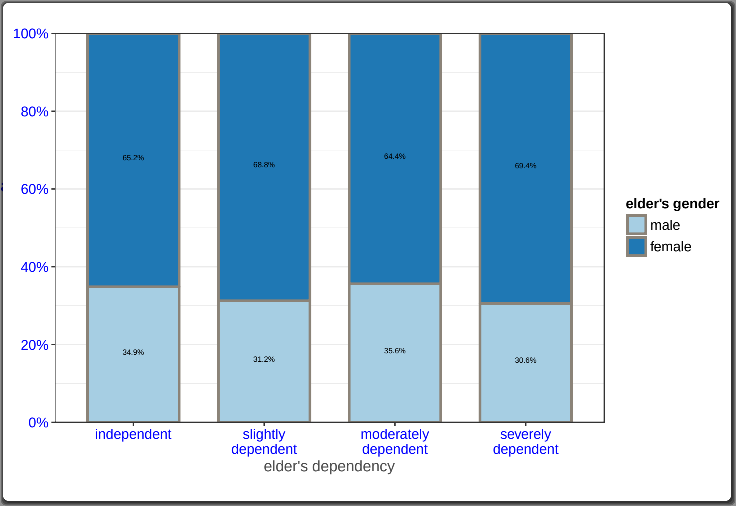


# sjPlot, Contingency tables, Other options

```
> sjp.xtab(x = efc$e42dep, grp = efc$e16sex,bar.pos = "stack",
+          margin = "row",show.n = F,show.total = F,
+          summary.pos = "l")
```

# sjPlot, Contingency tables, Other options

```
> sjp.xta
+
+
```



# sjPlot, Contingency tables, Other options

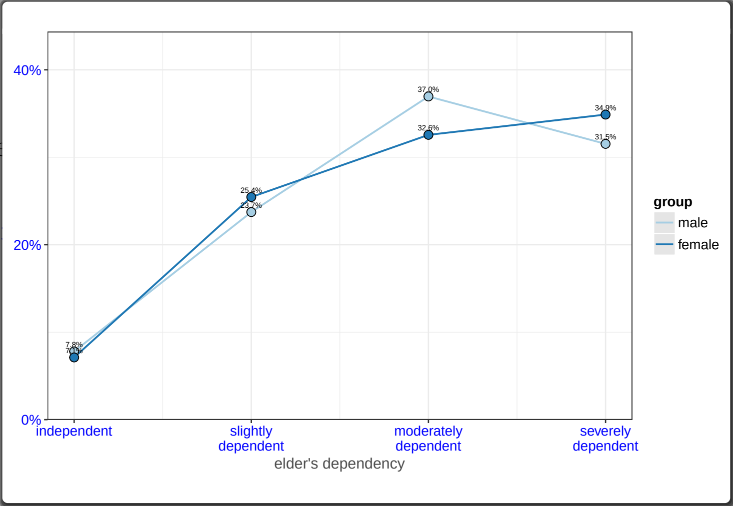
- We replace bars with lines

```
> sjp.xtab(x = efc$e42dep, grp = efc$e16sex,
+          show.n = F,show.total = F,
+          type="line")
```

# sjPlot, Contingency tables, Other options

- We

```
> sjp.
+
+
```



# Stacked bar plot

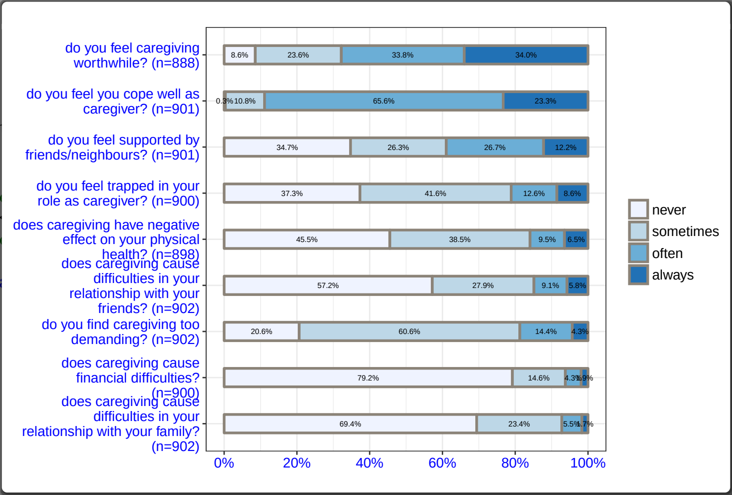
- Plot multiple variables with same categories.

```
> # recveive first item of COPE-index scale
> start <- which(colnames(efc) == "c82cop1")
> # recveive first item of COPE-index scale
> end <- which(colnames(efc) == "c90cop9")
> sjp.stackfrq(efc[, start:end], expand.grid = TRUE,
+              geom.size = .4,sort.frq = "last.desc")
```

# Stacked bar plot

- Plo

```
> # recv
> start
> # recv
> end <-
> sjp.st
+
```



# sjPlot, Likert-scales plots

- Create a dummy data set with
  - five items (columns)
  - 500 observations.
  - Each items has 4 category values, two so-called “positive” values (agree and strongly agree) versus two negative values (disagree and strongly disagree).

# sjPlot, Likert-scales plots

```
> ## Data
> mydf <- data.frame(
+   question1 = as.factor(sample(1:4, 500, replace = TRUE,
+                               prob = c(0.25, 0.33, 0.14, 0.28))),
+   question2 = as.factor(sample(1:4, 500, replace = TRUE,
+                               prob = c(0.5, 0.25, 0.15, 0.1))),
+   question3 = as.factor(sample(1:4, 500, replace = TRUE,
+                               prob = c(0.25, 0.1, 0.39, 0.26))),
+   question4 = as.factor(sample(1:4, 500, replace = TRUE,
+                               prob = c(0.17, 0.27, 0.38, 0.16))),
+   question5 = as.factor(sample(1:4, 500, replace = TRUE,
+                               prob = c(0.37, 0.26, 0.16, 0.21)))
+ )
```

# sjPlot, Likert-scales plots

```
> ## Create labels
> labels <- c("Strongly agree", "Agree", "Disagree",
+             "Strongly disagree")
```

# sjPlot, Likert-scales plots

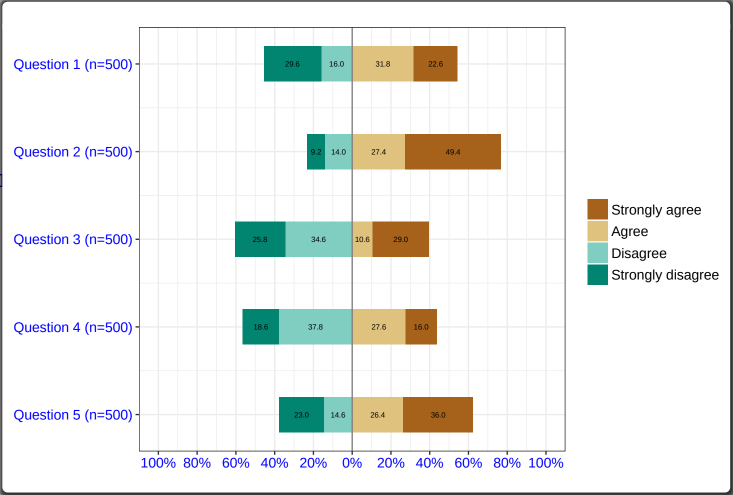
```
> ## Create labels
> labels <- c("Strongly agree", "Agree", "Disagree",
+              "Strongly disagree")

> ## Create item labels
> items <- c("Question 1", "Question 2", "Question 3",
+             "Question 4", "Question 5")
```

# sjPlot, Likert-scales plots

```
> sjp.likert(mydf, axis.labels = items,
+            legend.labels = labels,
+            geom.size = 0.4)
```

# sjPlot, Likert-scales plots

```
> sjp.li
+
+
```



# Radar Charts

# Radar Charts

- Radar charts are
  - called Spider or Web or Polar charts.
  - a way of comparing multiple quantitative variables.
  - are also useful for seeing which variables are scoring high or low within a dataset.
- We can use **`fmsb`** package to draw radar charts.

# Radar Charts

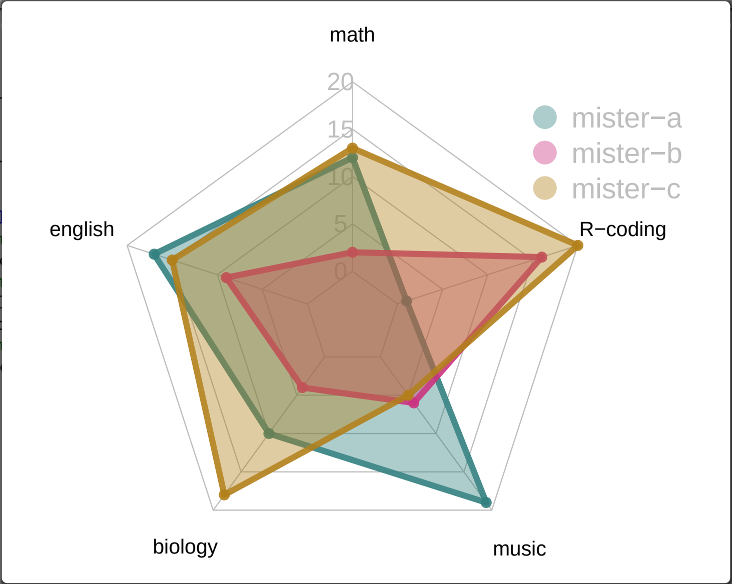
```
> library(fmsb)
>
> # Create data: note in High school for several students
> set.seed(99)
> data=as.data.frame(matrix( sample( 0:20 , 15 , replace=F) , ncol=5))
> colnames(data)=c("math" , "english" , "biology" , "music" , "R-coding" )
> rownames(data)=paste("mister" , letters[1:3] , sep="-")
> # We add 2 lines to the dataframe: the max and min of each
> # topic to show on the plot!
> data=rbind(rep(20,5) , rep(0,5) , data)
> data
         math english biology music R-coding
1          20      20      20    20       20
2           0       0       0     0        0
mister-a   12      17      10    19        1
mister-b    2       9       4     6       16
mister-c   13      15      18     5       20
```

# Radar Charts

```
> colors_border=c( rgb(0.2,0.5,0.5,0.9),
+                  rgb(0.8,0.2,0.5,0.9) ,
+                  rgb(0.7,0.5,0.1,0.9) )
> colors_in=c( rgb(0.2,0.5,0.5,0.4),
+              rgb(0.8,0.2,0.5,0.4) ,
+              rgb(0.7,0.5,0.1,0.4) )
> radarchart( data  , axistype=1 ,
+     #custom polygon
+     pcol=colors_border , pfcol=colors_in , plwd=4 , plty=1,
+     #custom the grid
+     cglcol="grey", cglty=1, axislabcol="grey",
+     caxislabels=seq(0,20,5), cglwd=0.8,
+     #custom labels
+     vlcex=0.8
+     )
> legend(x=0.7, y=1,
+        legend = rownames(data[-c(1,2),]),
+        bty = "n", pch=20 ,
+        col=colors_in , text.col = "grey", cex=1.2, pt.cex=3)
```

# Radar Charts

```
> colors_
+
+
> colors_
+
+
> radarc
+     #c
+     pc
+     #c
+     cg
+     ca
+     #c
+     vl
+     )
> legend
+
+
+
```



# `tabplot`, Large Data

# tabplot, Large data visualization

1. Explore and analyse large datasets.
2. Discover strange data patterns.
3. Check the occurrence and selectivity of missing values.

# Data

```
> require(ggplot2)
Loading required package: ggplot2
> data(diamonds)
> head(diamonds)
# A tibble: 6 » 10
  carat       cut color clarity depth table price     x     y     z
  <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23     Ideal     E     SI2  61.5    55   326  3.95  3.98  2.43
2  0.21   Premium     E     SI1  59.8    61   326  3.89  3.84  2.31
3  0.23      Good     E     VS1  56.9    65   327  4.05  4.07  2.31
4  0.29   Premium     I     VS2  62.4    58   334  4.20  4.23  2.63
5  0.31      Good     J     SI2  63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good     J    VVS2  62.8    57   336  3.94  3.96  2.48
```

# Data

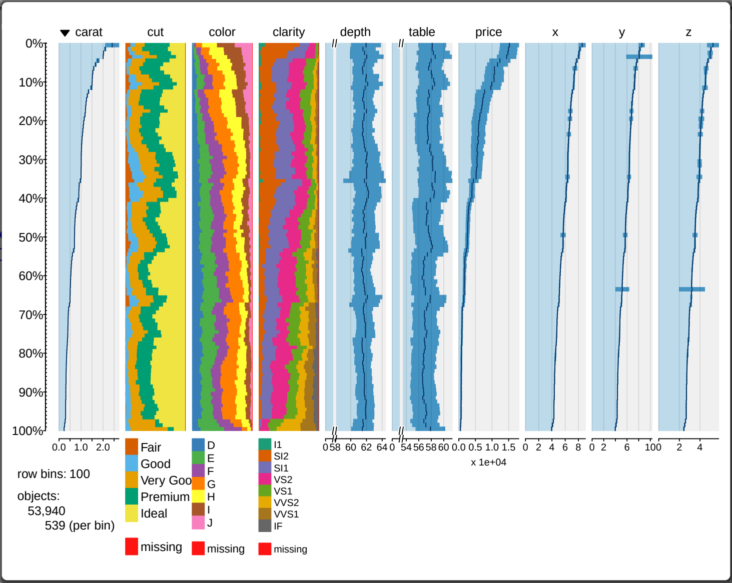
```
> summary(diamonds)
     carat                 cut        color        clarity
 Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065
 1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258
 Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194
 Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171
 3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066
 Max.   :5.0100                     I: 5422   VVS1   : 3655
                                    J: 2808   (Other): 2531
     depth           table           price             x
 Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
 1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
 Median :61.80   Median :57.00   Median : 2401   Median : 5.700
 Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
 3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
 Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740

       y                z
 Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.710   Median : 3.530
 Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :58.900   Max.   :31.800
```

# Exploring Data

```
> require(tabplot)
> tableplot(diamonds)
```

# Exploring Data

```
> requir
> tablep
```

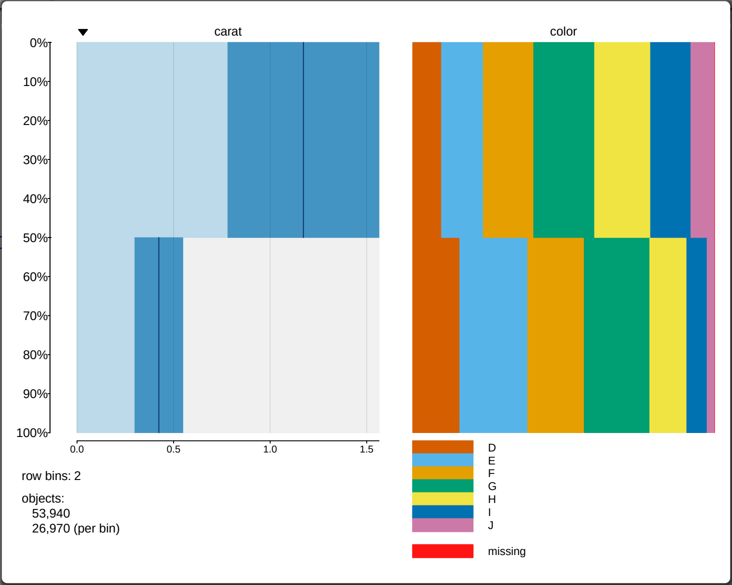


# Exploring Data, how it works?

```
> tableplot(diamonds, nBins=2,select =c(carat,color),decreasing = T)
```

# Exploring Data, how it works?

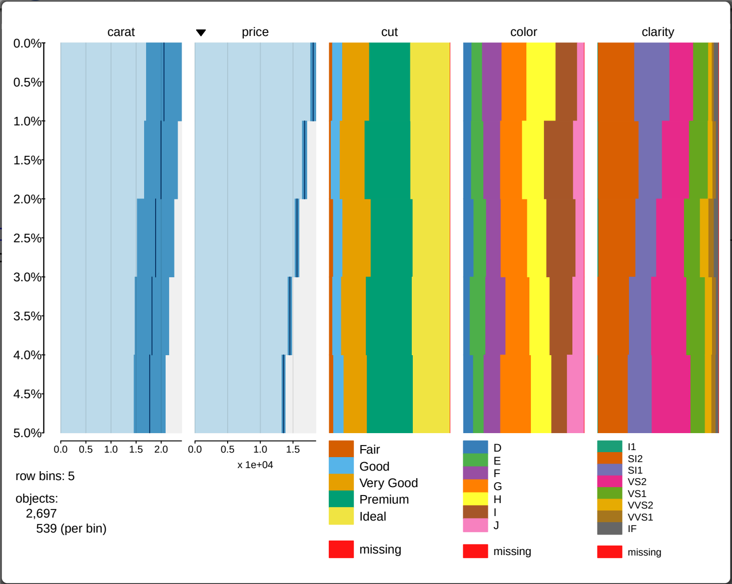
```
> tablep
```



# Zooming data,

```
> tableplot(diamonds, nBins=5, select = c(carat, price, cut, color, clarity),
+     sortCol = price, from = 0, to = 5)
```

# Zooming data,



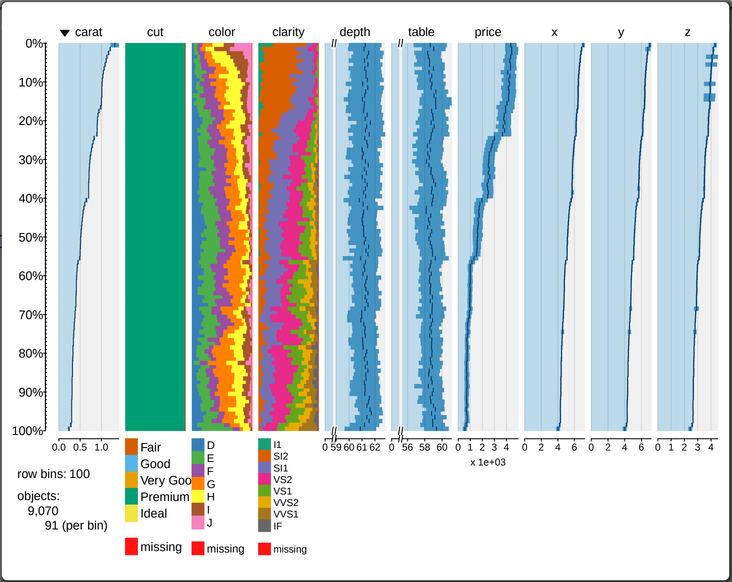

```
> tablep                                                        ty),
+       so
```

# Filtering data

```
> tableplot(diamonds, subset = price < 5000 & cut == "Premium")
```

# Filtering data

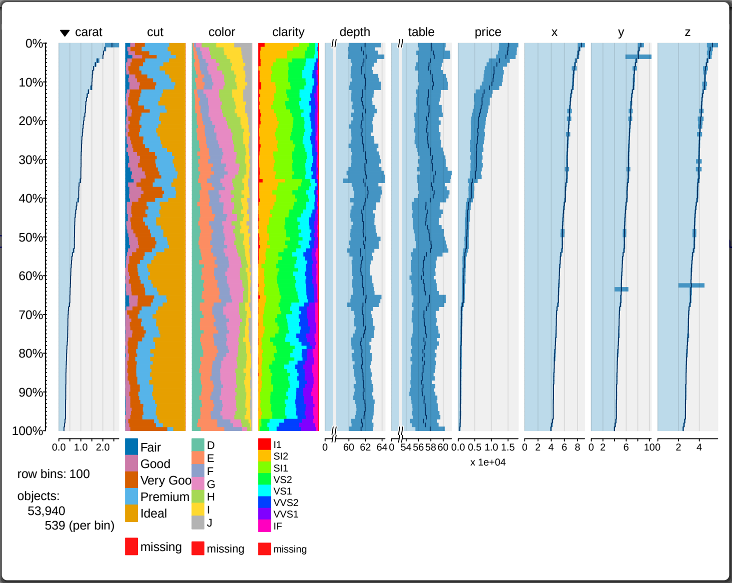
> tablep



# Change colors

```
> tableplot(diamonds, pals = list(cut="Set1(6)", color="Set5", clarity=rainbow(8)))
```

# Change colors




```
> tablep                                                      inbow(8)))
```

# Visualizing multivariate:

- Categorical Data
- Quantitative Data

# Categorical Data, Mosaic plots

# Mosaic Plots with ggmosaic, Titanic Data

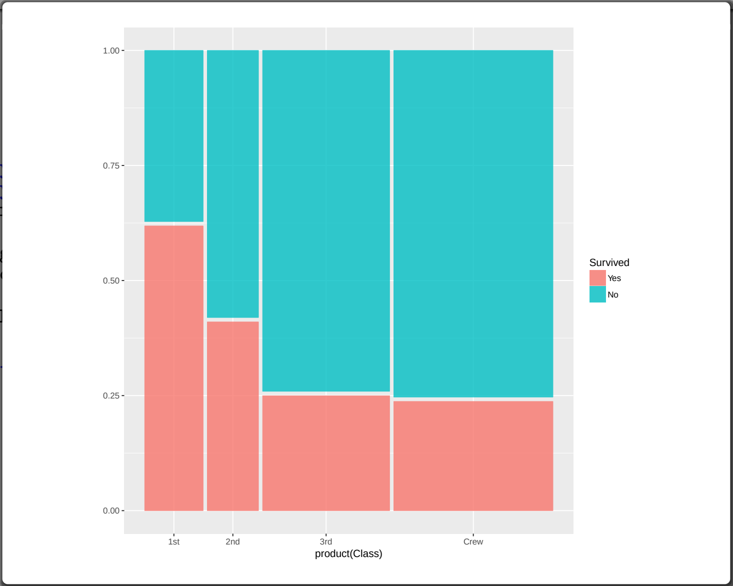
```
> data(Titanic)
> titanic <- as.data.frame(Titanic)
> titanic$Survived <- factor(titanic$Survived, levels=c("Yes", "No"))
> head(titanic)
  Class    Sex   Age Survived Freq
1   1st   Male Child       No    0
2   2nd   Male Child       No    0
3   3rd   Male Child       No   35
4  Crew   Male Child       No    0
5   1st Female Child       No    0
6   2nd Female Child       No    0
```

# Mosaic of table Class x Survived

```
> library(ggplot2)
> library(ggmosaic)
Loading required package: productplots

Attaching package: 'ggmosaic'
The following objects are masked from 'package:productplots':

    ddecker, hspine, mosaic, prodcalc, spine, vspine
> ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Class), fill=Survived))
```

# Mosaic of table Class x Survived

```
> librar
> librar
Loading 

Attachin
The foll

    ddecl
> ggplot
+   geom
```



# Mosaic of table Class x Survived, how it works?

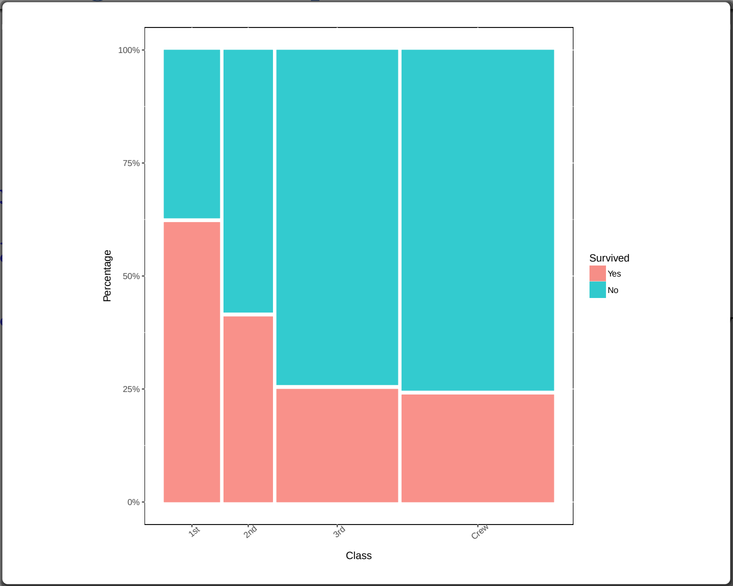
```
> margin.table(Titanic,margin = c(1,4))
      Survived
Class   No Yes
  1st  122 203
  2nd  167 118
  3rd  528 178
  Crew 673 212
> prop.table(margin.table(Titanic,margin = c(1,4)),1)
      Survived
Class         No       Yes
  1st  0.3753846 0.6246154
  2nd  0.5859649 0.4140351
  3rd  0.7478754 0.2521246
  Crew 0.7604520 0.2395480
```

# Customizing the Mosaic plot

```
> library(scales)
> ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Class), fill=Survived))+
+   scale_y_continuous(labels=percent) +
+   labs(x = "Class",
+        y = "Percentage") +
+   theme(panel.background = NULL, axis.text.x = element_text(angle=40, vjust=1))
```

# Customizing the Mosaic plot

```
> librar
> ggplot
+   geom
+   scal
+   labs
+
+   them                                        just=1))
```

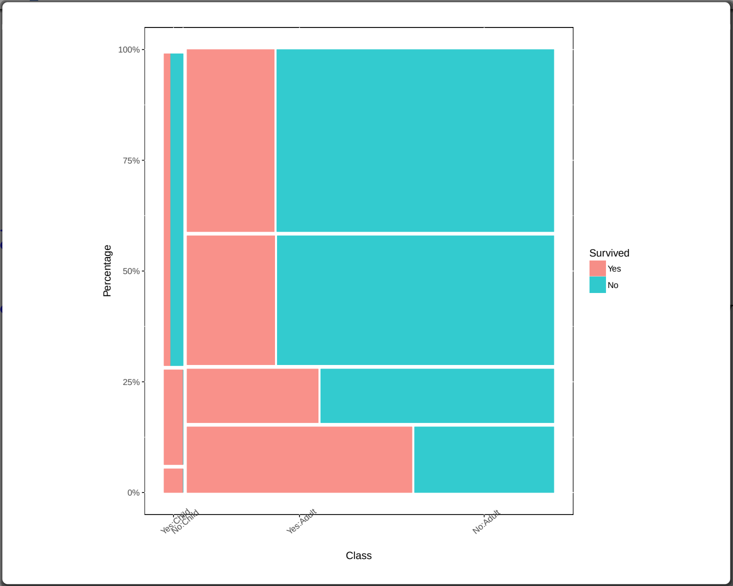


# Mosaic plot with 3 variables and more

```
> ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Class, Age), fill=Survived))+
+   scale_y_continuous(labels=percent) +
+   labs(x = "Class",
+        y = "Percentage") +
+   theme(panel.background = NULL, axis.text.x = element_text(angle=40, vjust=1))
```

# Mosaic plot with 3 variables and more

```
> ggplot
+   geom
+   scal
+   labs
+
+   them                                            just=1))
```

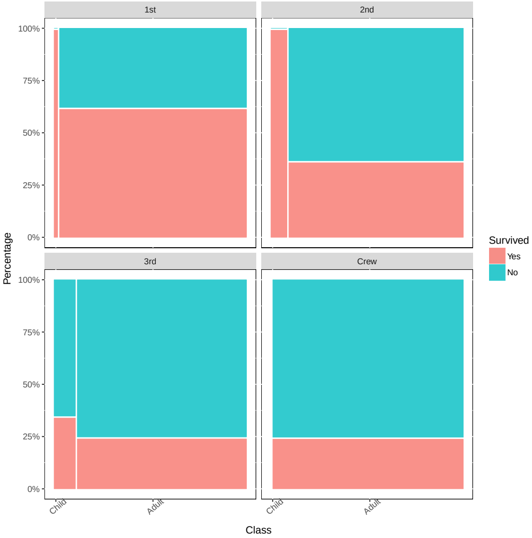


# Mosaic plot with 3 variables and more

```
> ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Age), fill=Survived))+
+   scale_y_continuous(labels=percent) +
+   labs(x = "Class",
+        y = "Percentage") +
+   theme(panel.background = NULL, axis.text.x = element_text(angle=40, vjust=1))+
+   facet_wrap(~Class)
```

# Mosaic plot with 3 variables and more

```
> ggplot
+   geom
+   scal
+   labs
+
+   them                                    just=1))+
+   face
```



# Mosaic plot with 3 variables and more

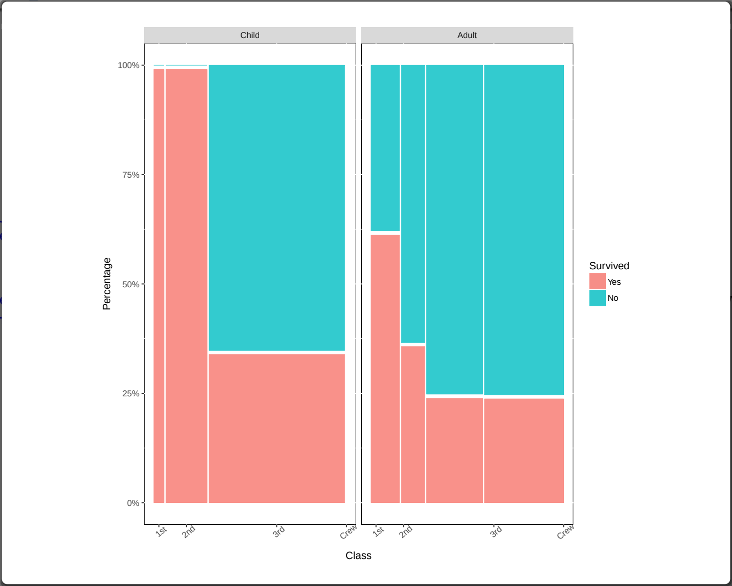
```
> margin.table(Titanic,margin = c(1,3,4))
, , Survived = No

      Age
Class  Child Adult
  1st      0   122
  2nd      0   167
  3rd     52   476
  Crew     0   673

, , Survived = Yes

      Age
Class  Child Adult
  1st      6   197
  2nd     24    94
  3rd     27   151
  Crew     0   212
```

# Mosaic plot with 3 variables and more

```
> ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Class),fill=Survived))+
+   scale_y_continuous(labels=percent) +
+   labs(x = "Class",
+        y = "Percentage") +
+   theme(panel.background = NULL, axis.text.x = element_text(angle=40, vjust=1))+
+   facet_wrap(~Age)
```

# Mosaic plot with 3 variables and more

```
> ggplot
+   geom
+   scal
+   labs
+
+   them                                                    just=1))+
+   face
```



# Adding frequencies to the mosaic plot

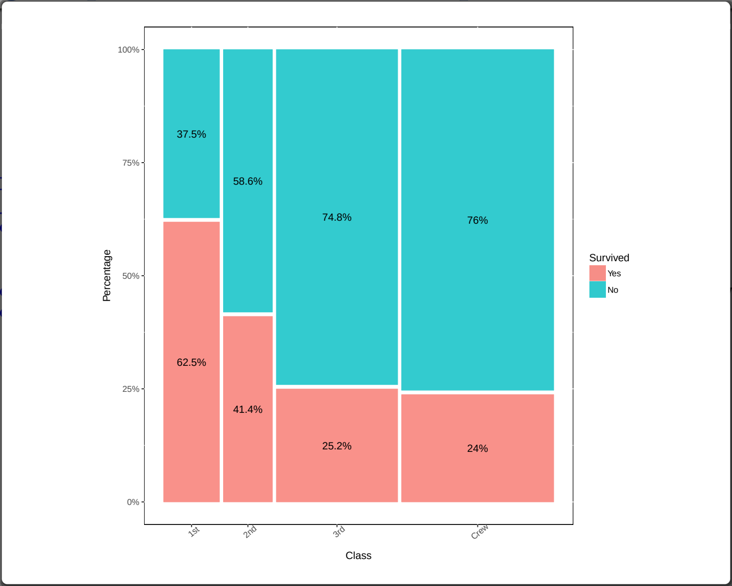
```
> p<-ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Class),fill=Survived))
> x=ggplot_build(p)
> z=prop.table(margin.table(Titanic,margin = c(1,4)),1)
> z=z[,levels(titanic$Survived)]
> z1=paste(round(100*as.vector(t(z)),1),"%",sep="")
> df=data.frame(xtext=(x$data[[1]]$xmin+x$data[[1]]$xmax)/2,
+               ytext=(x$data[[1]]$ymin+x$data[[1]]$ymax)/2,
+               value=z1)
> df
       xtext     ytext value
1 0.07161517 0.3093300 62.5%
2 0.07161517 0.8140973 37.5%
3 0.21603135 0.2050437 41.4%
4 0.21603135 0.7098110 58.6%
5 0.44440254 0.1248604 25.2%
6 0.44440254 0.6296277 74.8%
7 0.80498637 0.1186320   24%
8 0.80498637 0.6233993   76%
```

# Adding frequencies to the mosaic plot

```
> p<-ggplot(data=titanic) +
+   geom_mosaic(aes(weight=Freq, x=product(Class),fill=Survived))+
+   scale_y_continuous(labels=percent) +
+   labs(x = "Class",
+        y = "Percentage") +
+   theme(panel.background = NULL, axis.text.x = element_text(angle=40, vjust=1))
> p<-p+geom_text(data=df,aes(x=xtext,y=ytext,label=value))
> p
```

# Adding frequencies to the mosaic plot

```
> p<-ggp
+   geom
+   scal
+   labs
+
+   them                                              just=1))
> p<-p+g
> p
```



# Quantitative Data, Correlation matrix

# corrplot package

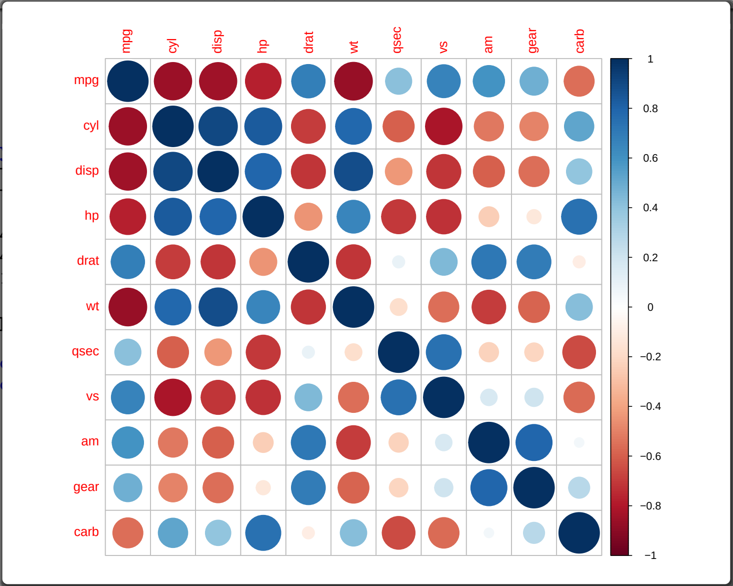
- Display of a correlation matrix, confidence interval.
- Contains some algorithms to do matrix reordering.
- Good at details, including choosing color, text labels, color labels, layout, etc.

# Visualization Methods, circles

```
> library(corrplot)
> data(mtcars)
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> M <- cor(mtcars)
> corrplot(M, method = "circle")
```

# Visualization Methods, circles

```
> librar
> data(m
> head(m

Mazda RX
Mazda RX
Datsun 7
Hornet 4
Hornet S
Valiant
> M <- c
> corrpl
```

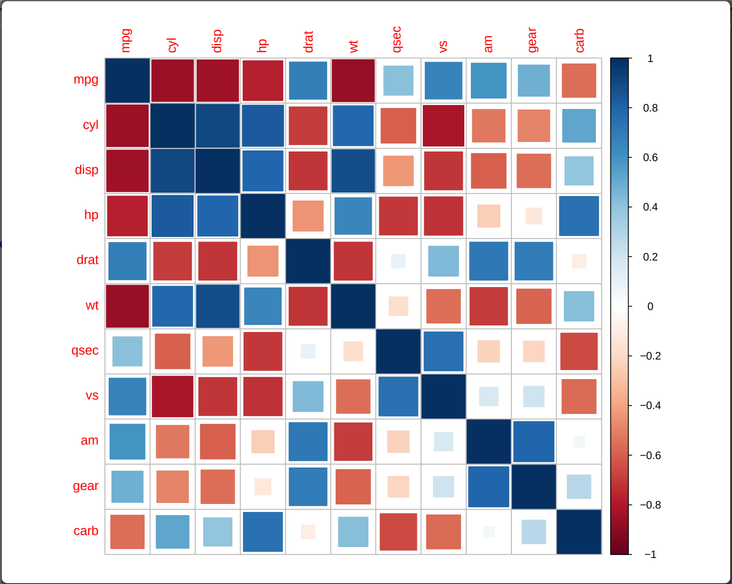


# Visualization Methods, squares

```
> corrplot(M, method = "square")
```

# Visualization Methods, squares

> corrpl

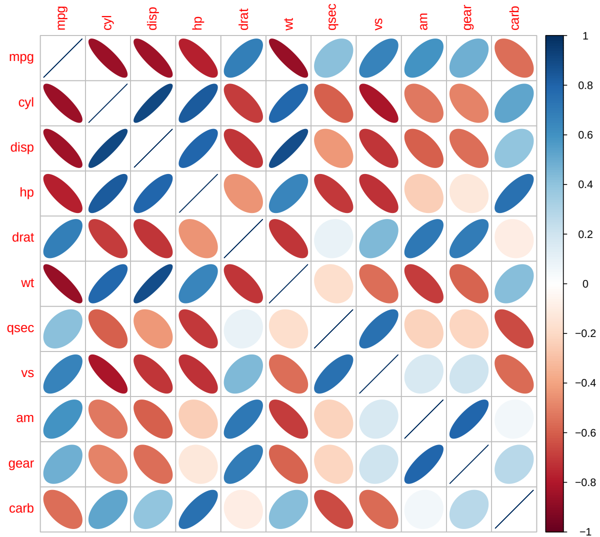


# Visualization Methods, ellipses

```
> corrplot(M, method = "ellipse")
```

# Visualization Methods, ellipses

> corrpl

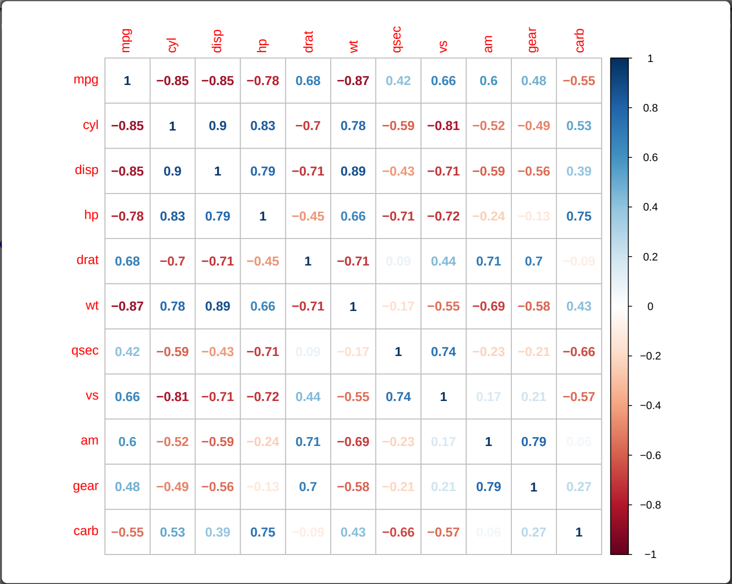


# Visualization Methods, numbers

```
> corrplot(M, method = "number")
```

# Visualization Methods, numbers

```
> corrpl
```

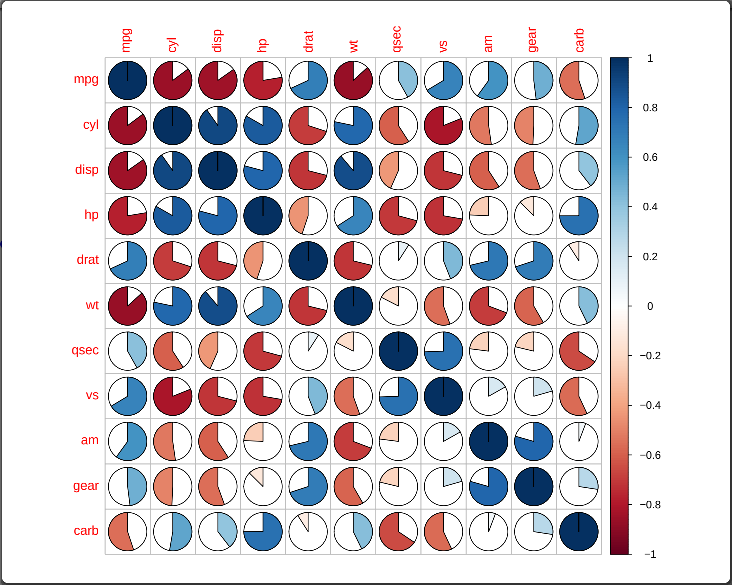


# Visualization Methods, pies

```
> corrplot(M, method = "pie")
```

# Visualization Methods, pies

```
> corrpl
```

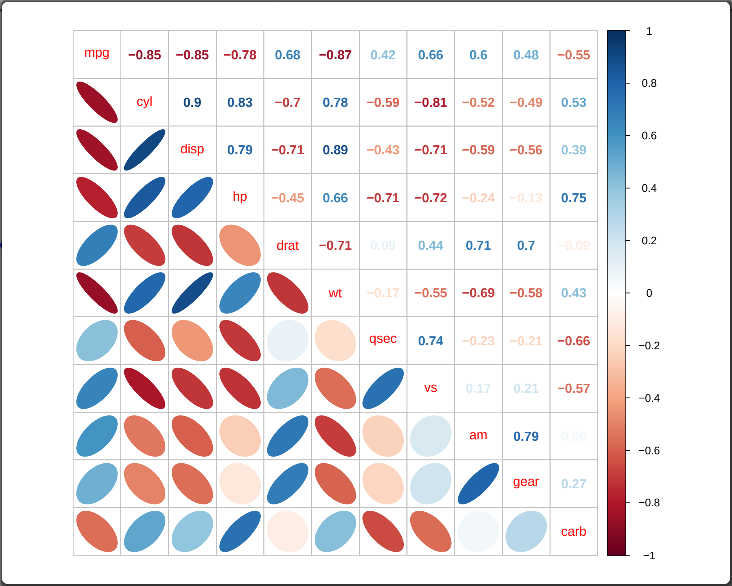


# Visualization Methods, mixed

```
> corrplot.mixed(M, lower = "ellipse", upper = "number")
```

# Visualization Methods, mixed

```
> corrpl
```



# Reorder A Correlation Matrix

- `AOE` based on the angle of eigen vector of the correlation matrix.
- `FPC` for the first principal component order.
- `hclust` for hierarchical clustering order, and `hclust.method` for the agglomeration method to be used.

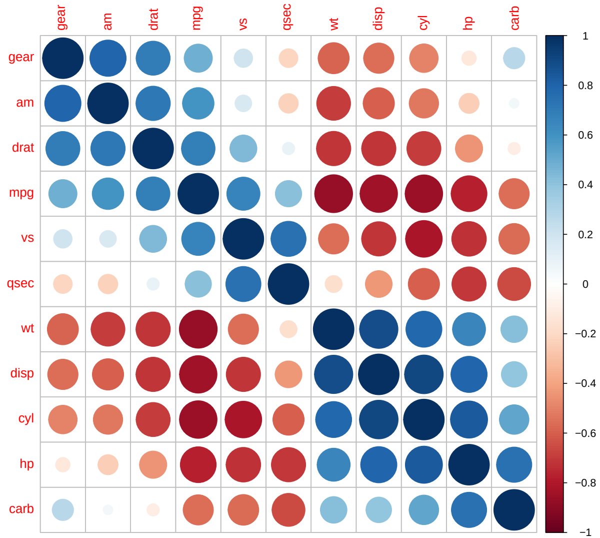
  `hclust.method` should be one of `ward`, `single`, `complete`, `average`, `mcquitty`, `median` or `centroid`.
- `alphabet` for alphabetical order.

# Reorder A Correlation Matrix, AOC

```
> corrplot(M, order = "AOE")
```

# Reorder A Correlation Matrix, AOC

```
> corrpl
```

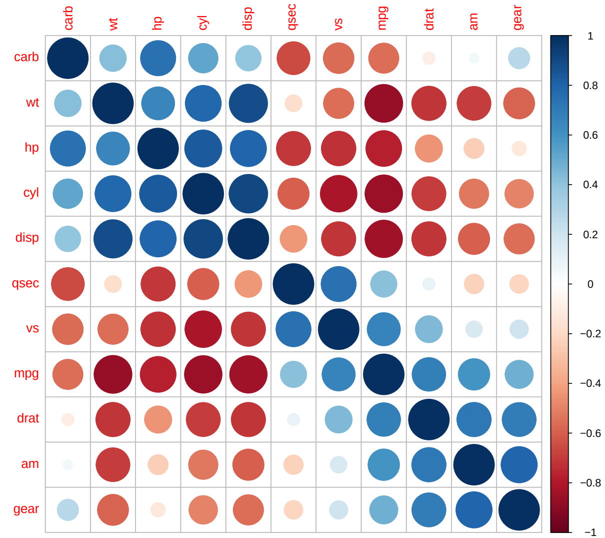


# Reorder A Correlation Matrix, hclust

```
> corrplot(M, order = "hclust")
```

# Reorder A Correlation Matrix, hclust

> corrpl

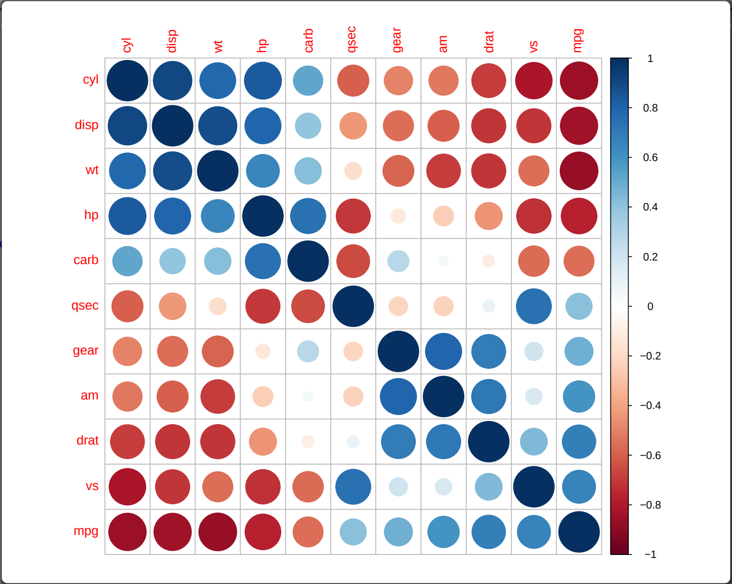


# Reorder A Correlation Matrix, FPC

```
> corrplot(M, order = "FPC")
```

# Reorder A Correlation Matrix, FPC

```
> corrpl
```

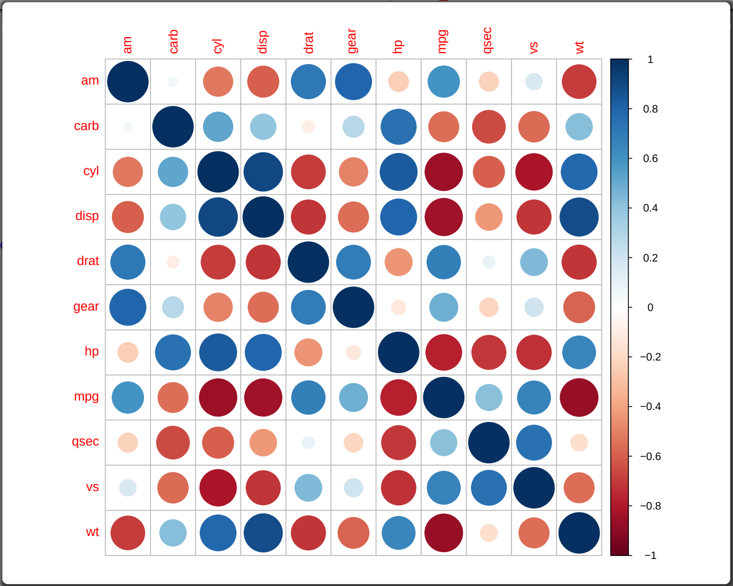


# Reorder a Correlation Matrix, alphabet

```
> corrplot(M, order = "alphabet")
```

# Reorder a Correlation Matrix, alphabet

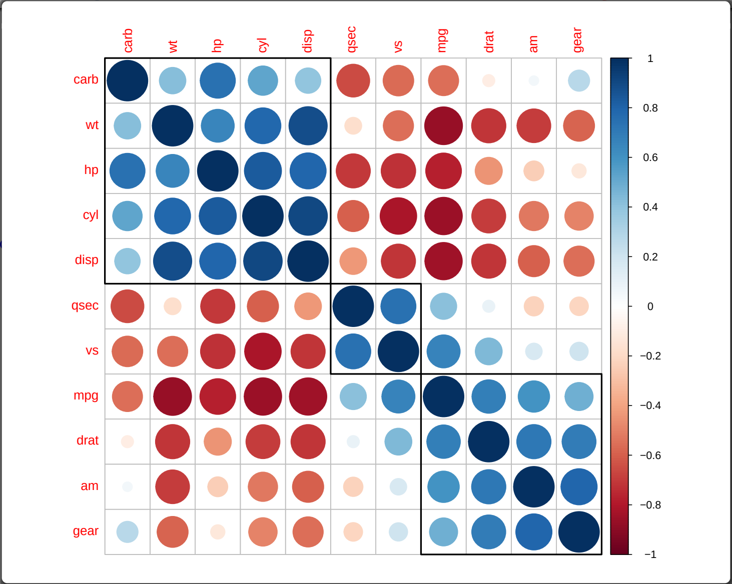
```
> corrpl
```



# Customizing the correlation matrix, adding rectangles

```
> corrplot(M, order = "hclust",addrect = 3)
```

# Customizing the correlation matrix, adding rectangles



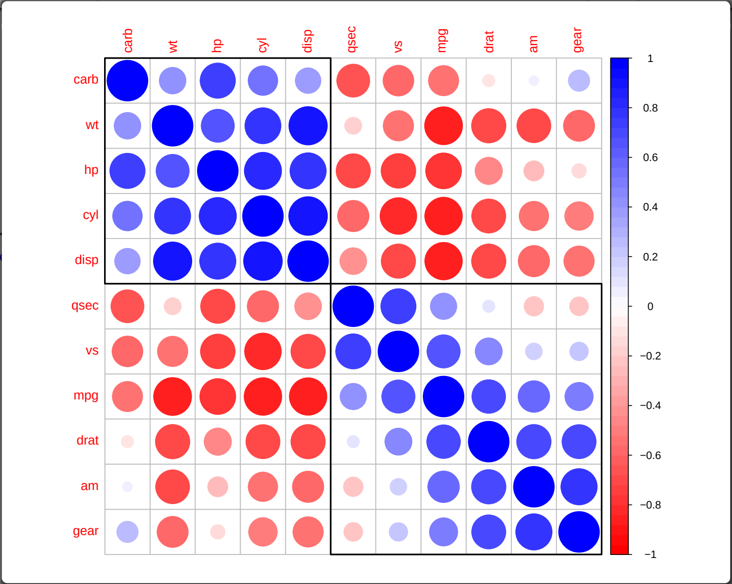

```
> corrpl
```

# Customizing the correlation matrix, changing colors

```
> mycol <- colorRampPalette(c("red", "white", "blue"))
> corrplot(M, order = "hclust",addrect = 2,col=mycol(50))
```

# Customizing the correlation matrix, changing colors

```
> mycol
> corrpl
```

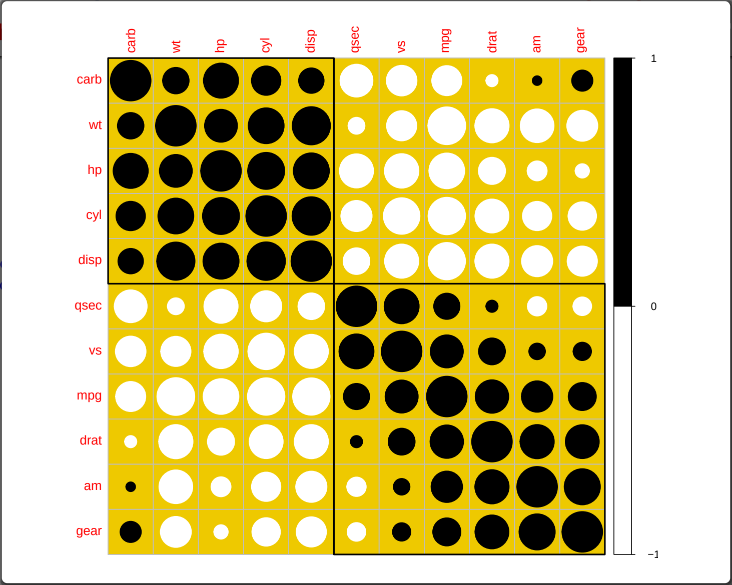


# Customizing the correlation matrix, changing background

```
> wb <- c("white", "black")
> corrplot(M, order = "hclust", addrect = 2, col = wb, bg = "gold2")
```

# Customizing the correlation matrix, changing backg

```
> wb <- 
> corrpl
```



# Correlation Independence test

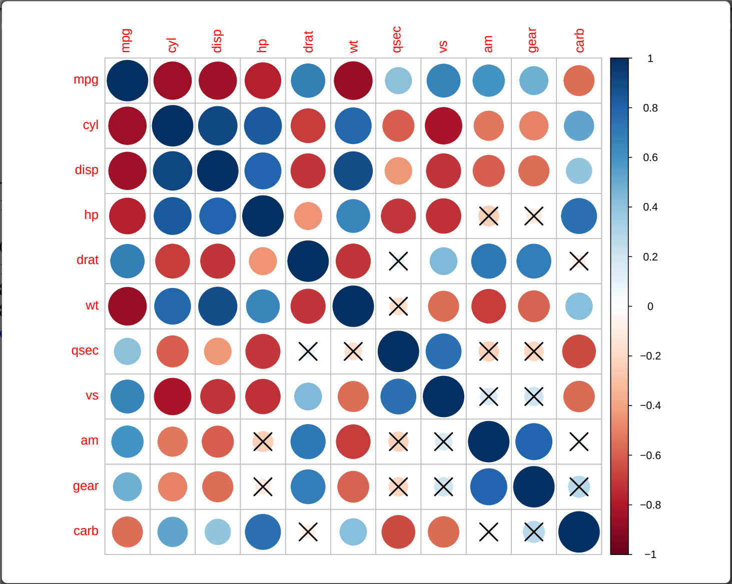
```
> cor.mtest <- function(mat, conf.level = 0.95) {
+     mat <- as.matrix(mat)
+     n <- ncol(mat)
+     p.mat <- lowCI.mat <- uppCI.mat <- matrix(NA, n, n)
+     diag(p.mat) <- 0
+     diag(lowCI.mat) <- diag(uppCI.mat) <- 1
+     for (i in 1:(n - 1)) {
+         for (j in (i + 1):n) {
+             tmp <- cor.test(mat[, i], mat[, j], conf.level = conf.level)
+             p.mat[i, j] <- p.mat[j, i] <- tmp$p.value
+             lowCI.mat[i, j] <- lowCI.mat[j, i] <- tmp$conf.int[1]
+             uppCI.mat[i, j] <- uppCI.mat[j, i] <- tmp$conf.int[2]
+         }
+     }
+     return(list(p.mat, lowCI.mat, uppCI.mat))
+ }
```

# Correlation Independence test

```
> res1 <- cor.mtest(mtcars, 0.95)
> res1[[1]][1:4,1:4]
             [,1]         [,2]         [,3]         [,4]
[1,] 0.000000e+00 6.112687e-10 9.380327e-10 1.787835e-07
[2,] 6.112687e-10 0.000000e+00 1.802838e-12 3.477861e-09
[3,] 9.380327e-10 1.802838e-12 0.000000e+00 7.142679e-08
[4,] 1.787835e-07 3.477861e-09 7.142679e-08 0.000000e+00
> corrplot(M, p.mat = res1[[1]], sig.level = 0.1)
```

# Correlation Independence test

```
> res1 <-
> res1[[
[1,] 0.0
[2,] 6.1
[3,] 9.3
[4,] 1.7
> corrpl
```

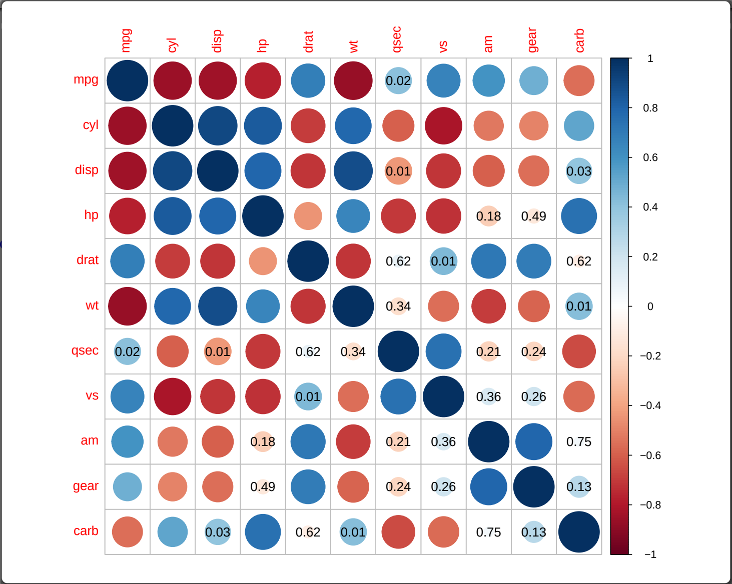


# Correlation Independence test

```
> corrplot(M, p.mat = res1[[1]], sig.level = 0.01,insig = "p-value")
```

# Correlation Independence test

```
> corrpl
```

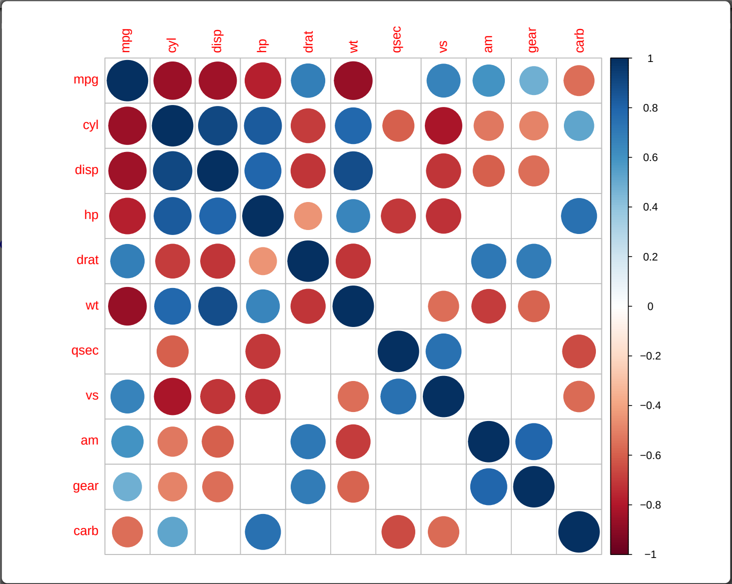


# Correlation Independence test

```
> corrplot(M, p.mat = res1[[1]], sig.level = 0.01,insig = "blank")
```

# Correlation Independence test

```
> corrpl
```



# Visualizing data with target variable and results of Statistical Models.

# Regression models, `sjPlot`

# ANOVA

```
> library(sjmisc)
> library(sjPlot)
#refugeeswelcome
> data(efc)
> attr(efc$e42dep,"labels")
        independent   slightly dependent moderately dependent
                  1                    2                    3
  severely dependent
                  4
```

# ANOVA

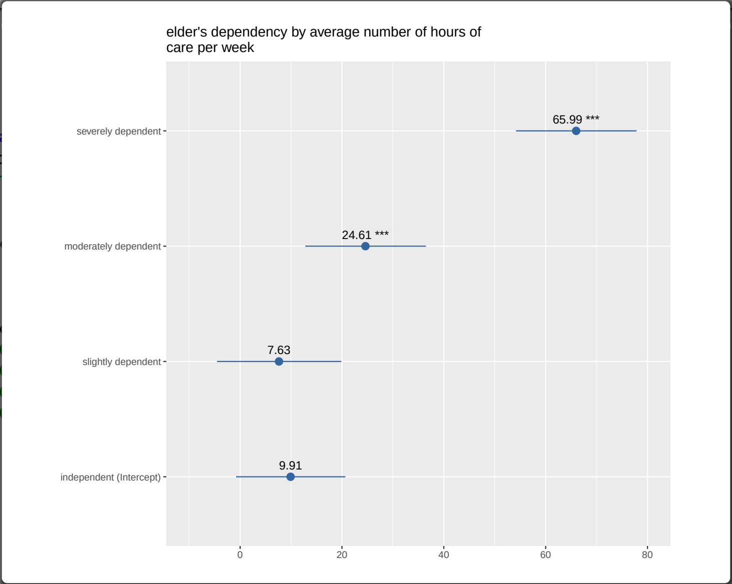
```
> summary(lm(efc$c12hour~as.factor(efc$e42dep)))

Call:
lm(formula = efc$c12hour ~ as.factor(efc$e42dep))

Residuals:
    Min      1Q  Median      3Q     Max 
-71.901 -24.520  -7.538   9.099 150.462 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)               9.909      5.445   1.820   0.0691 .  
as.factor(efc$e42dep)2    7.629      6.193   1.232   0.2183    
as.factor(efc$e42dep)3   24.611      6.004   4.099 4.52e-05 ***
as.factor(efc$e42dep)4   65.992      6.007  10.985  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 44.24 on 897 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared:  0.2448,    Adjusted R-squared:  0.2422 
F-statistic: 96.91 on 3 and 897 DF,  p-value: < 2.2e-16
>
```

# ANOVA

```
> x=sjp.aov1(efc$c12hour, efc$e42dep)
> names(x)
[1] "plot" "data"
> x$data
         term  estimate   conf.low conf.high      p.value  p.string xpos
1 (Intercept)  9.909091 -0.7779303  20.59611 6.913035e-02      9.91    1
2    var.grp2  7.628687 -4.5251081  19.78248 2.183125e-01      7.63    2
3    var.grp3 24.610517 12.8272036  36.39383 4.523872e-05 24.61 ***    3
4    var.grp4 65.992225 54.2020366  77.78241 1.994596e-26 65.99 ***    4
  geom.color
1    #3366a0
2    #3366a0
3    #3366a0
4    #3366a0
> x$plot ## to plot the ANOVA
```

# ANOVA

```
> x=sjp.a
> names(
[1] "plo
> x$data

1 (Inter
2    var
3    var
4    var
  geom.c
1    #33
2    #33
3    #33
4    #33
> x$plot
```

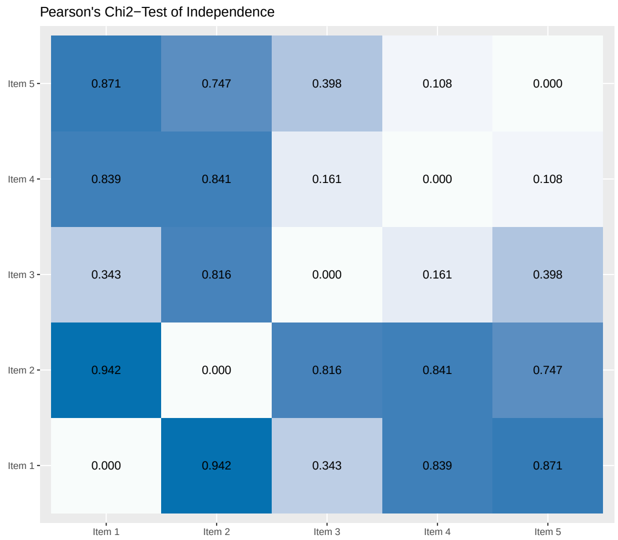


# Pearson's Chi2-tests

```
> # create data frame with 5 dichotomous (dummy) variables
> mydf <- data.frame(as.factor(sample(1:2, 100, replace=TRUE)),
+                    as.factor(sample(1:2, 100, replace=TRUE)),
+                    as.factor(sample(1:2, 100, replace=TRUE)),
+                    as.factor(sample(1:2, 100, replace=TRUE)),
+                    as.factor(sample(1:2, 100, replace=TRUE)))
> colnames(mydf)=c("x1","x2","x3","x4","x5")
> # create variable labels
> items <- list(c("Item 1", "Item 2", "Item 3", "Item 4", "Item 5"))
>
> # plot Chi2-contingency-table
> x=sjp.chi2(mydf, axis.labels = items)
> x$mydf[1:2,]
  Row Column Chi.Square df p.value
1  x1     x1    95.9370  1  0.0000
2  x2     x1     0.0054  1  0.9417
> chisq.test(xtabs(~mydf$x1+mydf$x2))

    Pearson's Chi-squared test with Yates' continuity correction

data:  xtabs(~mydf$x1 + mydf$x2)
X-squared = 0.0053545, df = 1, p-value = 0.9417
> x$plot
```

# Pearson's Chi2-tests

```
> # crea
> mydf <
+
+
+
+
> colnam
> # crea
> items
>
> # plot
> x=sjp.
> x$mydf
  Row Co
1  x1
2  x2
> chisq.

    Pear

data:  x
X-square
> x$plot
```



# Linear models, $\beta$ coefficients

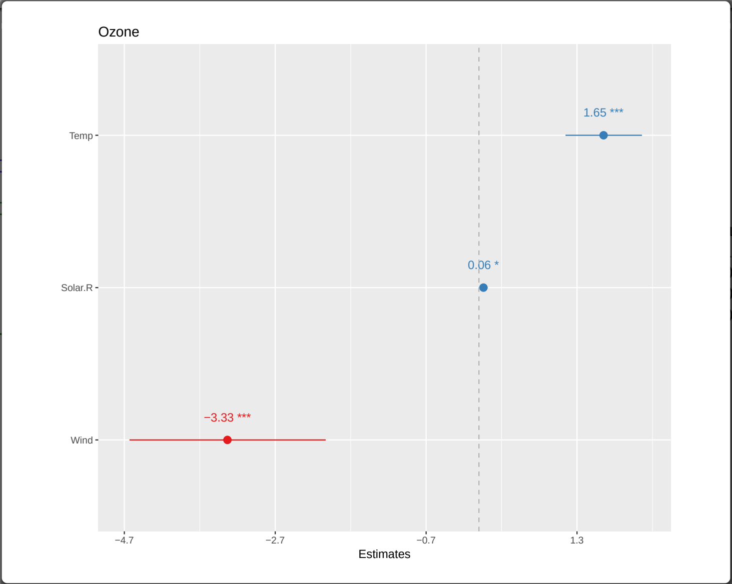
```
> # fit linear model
> fit <- lm(Ozone ~ Wind + Temp + Solar.R, data=airquality)
> summary(fit)

Call:
lm(formula = Ozone ~ Wind + Temp + Solar.R, data = airquality)

Residuals:
    Min      1Q  Median      3Q     Max
-40.485 -14.219  -3.551  10.097  95.619

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -64.34208   23.05472  -2.791  0.00623 **
Wind         -3.33359    0.65441  -5.094 1.52e-06 ***
Temp          1.65209    0.25353   6.516 2.42e-09 ***
Solar.R       0.05982    0.02319   2.580  0.01124 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.18 on 107 degrees of freedom
  (42 observations deleted due to missingness)
Multiple R-squared:  0.6059,    Adjusted R-squared:  0.5948
F-statistic: 54.83 on 3 and 107 DF,  p-value: < 2.2e-16
```

# Linear models, $\beta$ coefficients

```
> x=sjp.lm(fit, grid.breaks = 2)
> x$data
# A tibble: 3 » 8
    xpos    term    estimate    conf.low  conf.high  p.string      p.value
* <fctr>   <chr>       <dbl>       <dbl>      <dbl>     <chr>        <dbl>
1      1    Wind -3.33359131 -4.63087706 -2.0363055 -3.33 *** 1.515934e-06
2      2 Solar.R  0.05982059  0.01385613  0.1057851    0.06 * 1.123664e-02
3      3    Temp  1.65209291  1.14949967  2.1546862  1.65 *** 2.423506e-09
# ... with 1 more variables: group <lgl>
> x$plot
```

# Linear models, $\beta$ coefficients

```
> x=sjp.
> x$data
# A tibb
   xpos                                                              e
* <fctr>                                                             >
1       1                                                            6
2       2                                                            2
3       3                                                            9
# ... wi
> x$plot
```

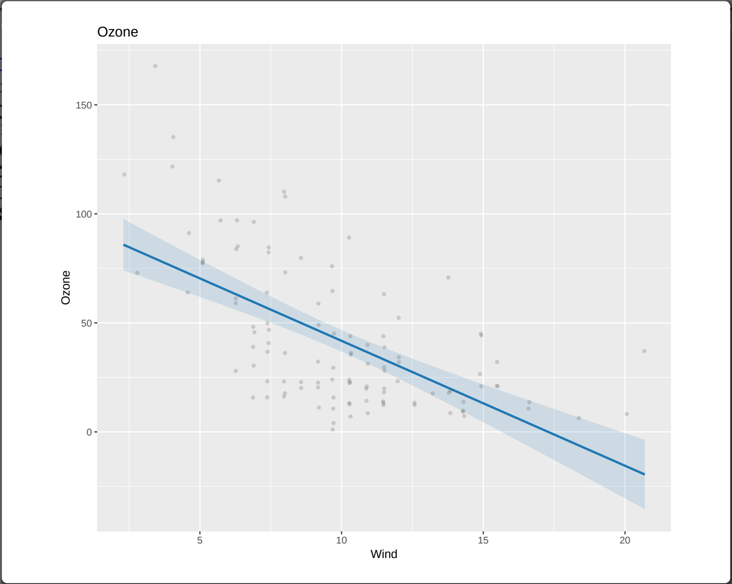


# Linear models, $\beta$ coefficients, slopes for each predictor

```
> x=sjp.lm(fit, grid.breaks = 2,type = "slope")
> x$df.list[[1]][1:3,]
     x  y
1  7.4 41
2  8.0 36
3 12.6 12
> airquality[1:3,]
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> x$plot.list[[1]]
> x$plot.list[[2]]
> x$plot.list[[3]]
> x$plot.list[[1]]
> x$plot.list[[2]]
> x$plot.list[[3]]
```

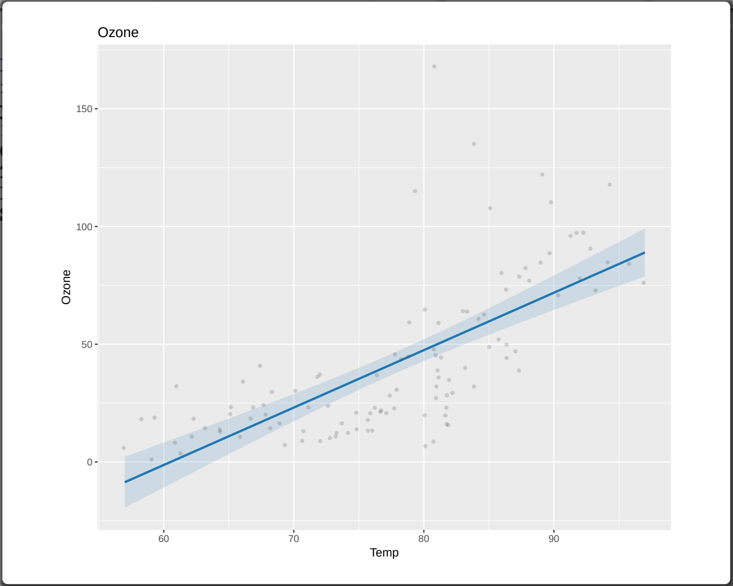
# Linear models, $\beta$ coefficients, slopes for each predictor

```
> x=sjp.l
> x$df.li
    x   
1  7.4 4
2  8.0 3
3 12.6 1
> airqual
  Ozone S
1    41
2    36
3    12
> x$plot
> x$plot
> x$plot
> x$plot
> x$plot
> x$plot
```



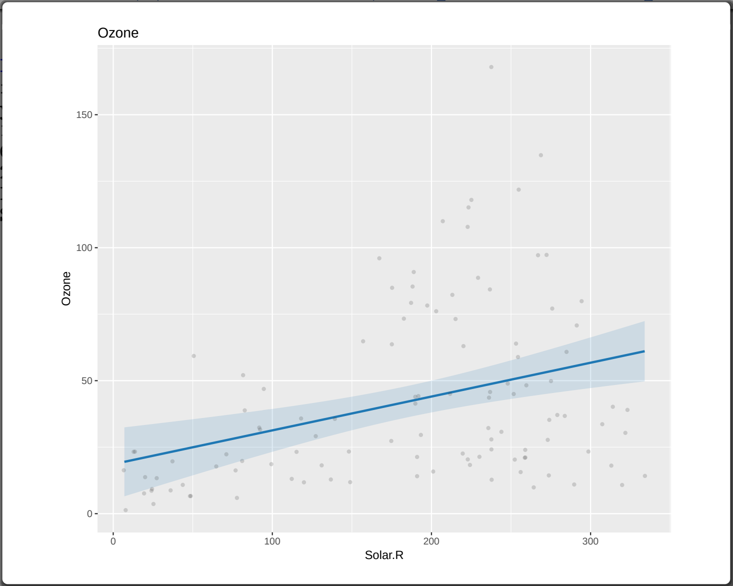
# Linear models, $\beta$ coefficients, slopes for each predictor

```
> x=sjp.l
> x$df.li
     x  
1  7.4 4
2  8.0 3
3 12.6 1
> airqual
  Ozone S
1    41
2    36
3    12
> x$plot
> x$plot
> x$plot
> x$plot
> x$plot
> x$plot
```



# Linear models, $\beta$ coefficients, slopes for each predictor

```
> x=sjp.l
> x$df.li
    x   
1  7.4 41
2  8.0 36
3 12.6 12
> airqual
  Ozone S
1    41
2    36
3    12
> x$plot
> x$plot
> x$plot
> x$plot
> x$plot
> x$plot
```

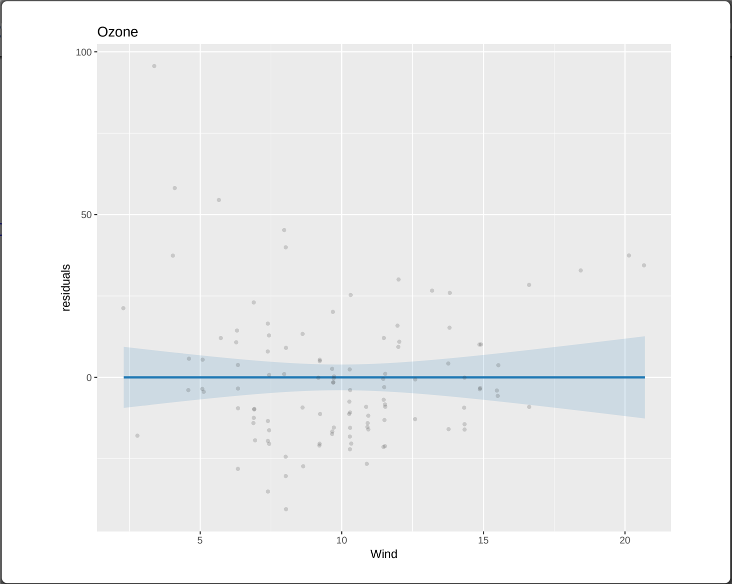


# Linear models, $\beta$ coefficients, residuals for each predictor

```
> x=sjp.lm(fit, grid.breaks = 2,type = "resid")
```

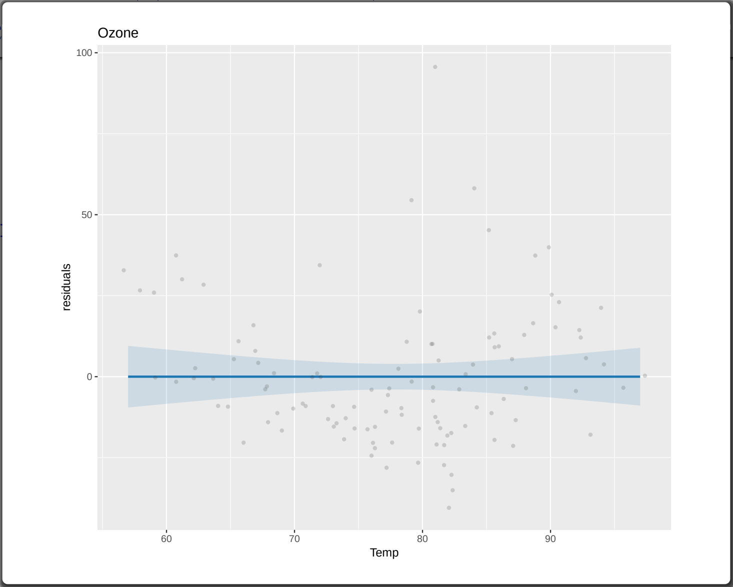
# Linear models, $\beta$ coefficients, residuals for each predic

> x=sjp.



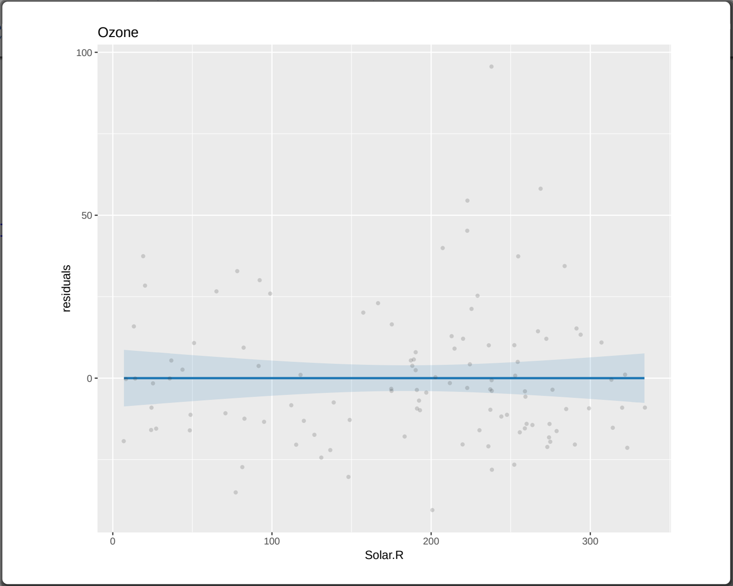
# Linear models, $\beta$ coefficients, residuals for each predic

```
> x=sjp.
```



# Linear models, $\beta$ coefficients, residuals for each predic

```
> x=sjp.l
```

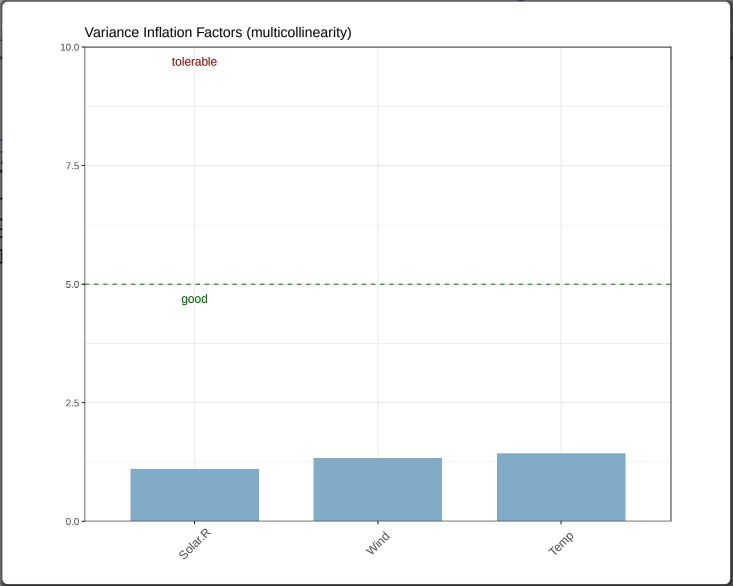


# Linear models, $\beta$ coefficients, checking model assumptions

```
> x=sjp.lm(fit, type = "ma")
Removed 3 cases during 1 step(s).
R^2 / adj. R^2 of original model: 0.605895 / 0.594845
R^2 / adj. R^2 of updated model:  0.663962 / 0.654268
AIC of original model: 998.717103
AIC of updated model:  926.512020
```

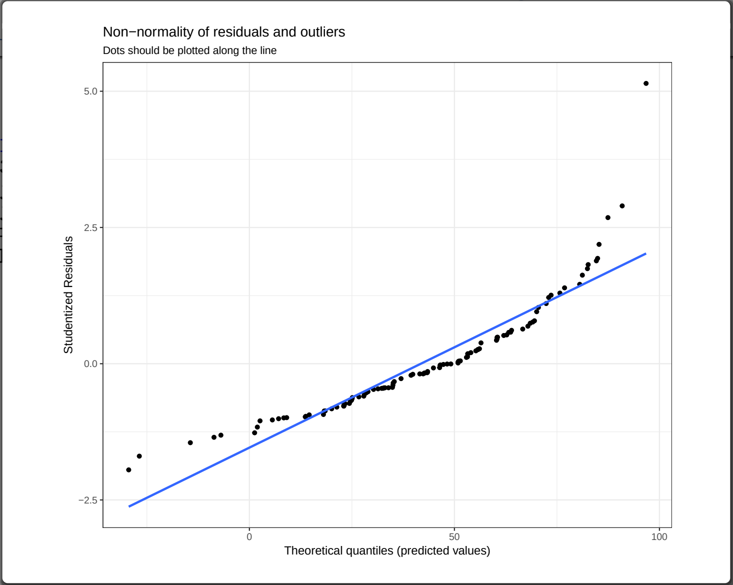
# Linear models, $\beta$ coefficients, checking model assum

```
> x=sjp.
Removed
R^2 / ad
R^2 / ad
AIC of o
AIC of u
```



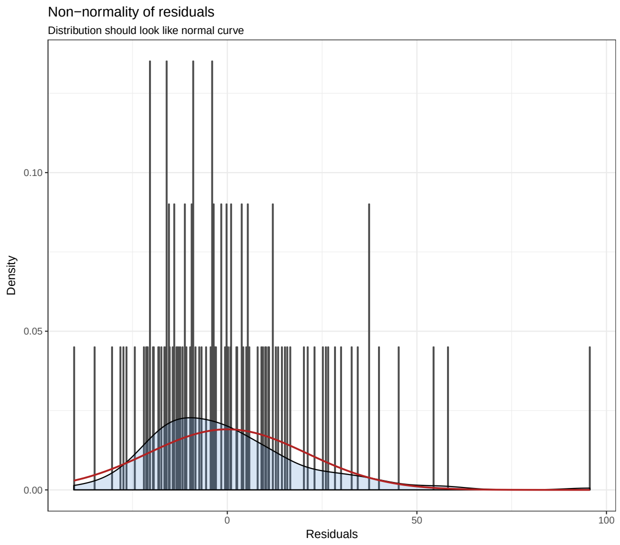
# Linear models, $\beta$ coefficients, checking model assum

```
> x=sjp.
Removed
R^2 / ad
R^2 / ad
AIC of o
AIC of u
```

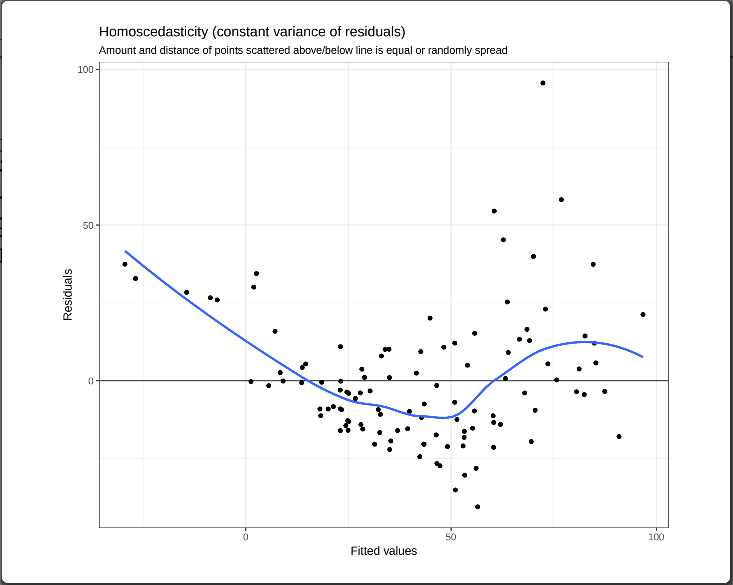


# Linear models, $\beta$ coefficients, checking model assum

```
> x=sjp.
Removed
R^2 / ad
R^2 / ad
AIC of o
AIC of u
```



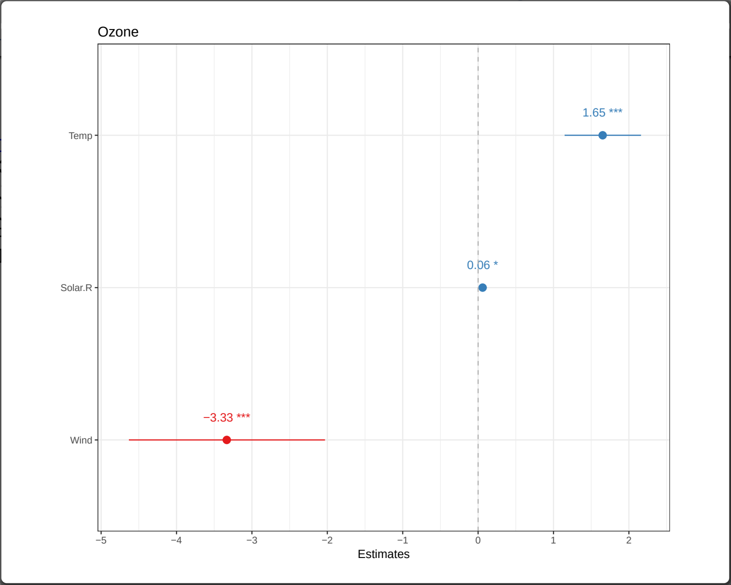
# Linear models, $\beta$ coefficients, checking model assum




```
> x=sjp.
Removed
R^2 / ad
R^2 / ad
AIC of o
AIC of u
```

# Linear models, $\beta$ coefficients, checking model assum

```
> x=sjp.
Removed 
R^2 / ad
R^2 / ad
AIC of o
AIC of u
```

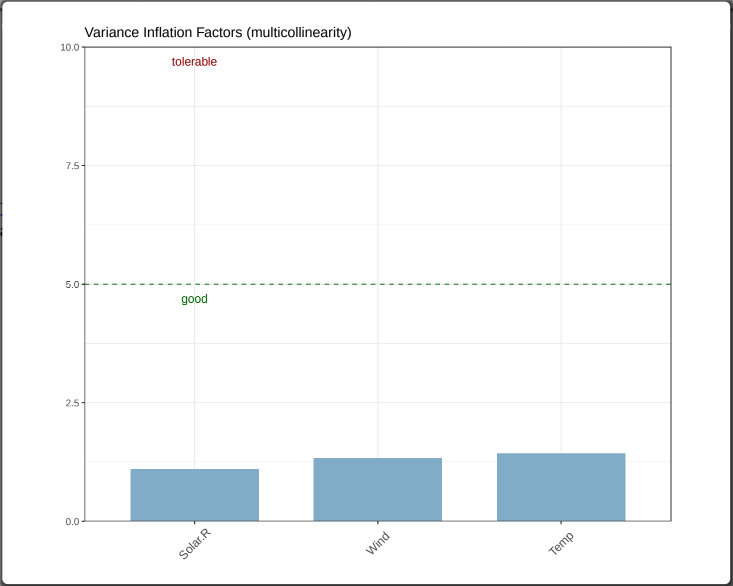


# Linear models, $\beta$ coefficients, Variance Inflation factor

```
> x=sjp.lm(fit, type = "vif")
> x$vifval
    Wind     Temp  Solar.R
1.329070 1.431367 1.095253
> x$plot
```

# Linear models, $\beta$ coefficients, Variance Inflation factor

```
> x=sjp.
> x$vifv
    Wind
1.329070
> x$plot
```



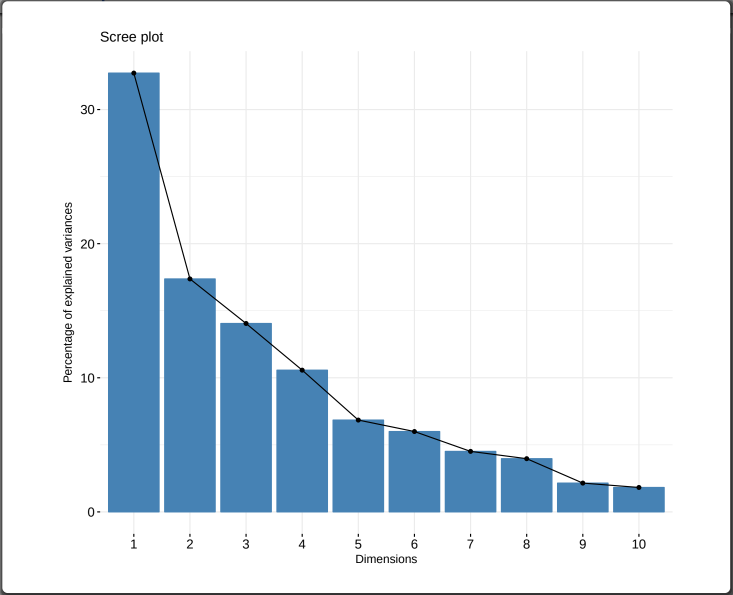
# PCA, CA and MCA

# PCA

```
> library(FactoMineR)
> library(factoextra)
Loading required package: ggplot2
> data(decathlon)
> head(decathlon)
         100m Long.jump Shot.put High.jump  400m 110m.hurdle Discus
SEBRLE  11.04      7.58    14.83      2.07 49.81       14.69  43.75
CLAY    10.76      7.40    14.26      1.86 49.37       14.05  50.72
KARPOV  11.02      7.30    14.77      2.04 48.37       14.09  48.95
BERNARD 11.02      7.23    14.25      1.92 48.93       14.99  40.87
YURKOV  11.34      7.09    15.19      2.10 50.42       15.31  46.26
WARNERS 11.11      7.60    14.31      1.98 48.68       14.23  41.10
        Pole.vault Javeline 1500m Rank Points Competition
SEBRLE        5.02    63.19 291.7    1   8217    Decastar
CLAY          4.92    60.15 301.5    2   8122    Decastar
KARPOV        4.92    50.31 300.2    3   8099    Decastar
BERNARD       5.32    62.77 280.1    4   8067    Decastar
YURKOV        4.72    63.44 276.4    5   8036    Decastar
WARNERS       4.92    51.77 278.1    6   8030    Decastar
> pc1=PCA(decathlon,ncp=3,scale.unit = T,quanti.sup=11:12,quali.sup=13,graph = F)
```

# PCA, scree plot

```
> fviz
```

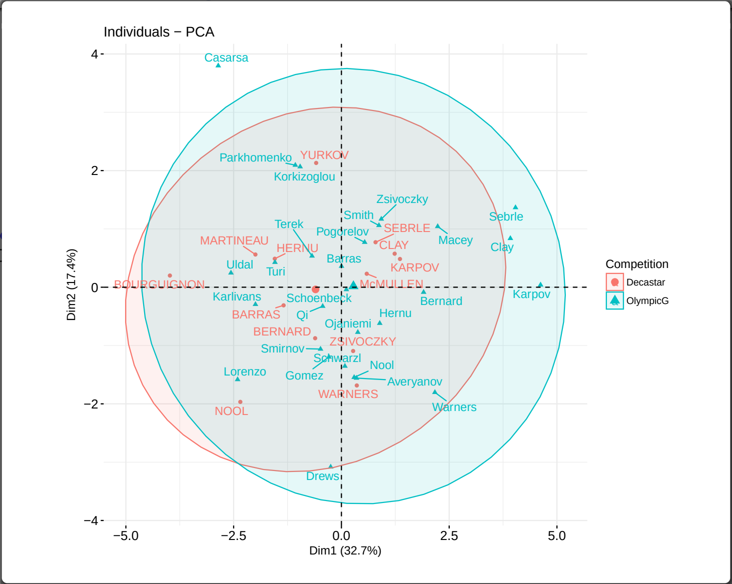


# PCA, Representing individuals

```
> fviz_pca_ind(pc1,axes=c(1,2),repel = T,habillage = "Competition",
+  addEllipses=TRUE, ellipse.level=0.95)
```

# PCA, Representing individuals

```
> fviz_p
+  addEl
```

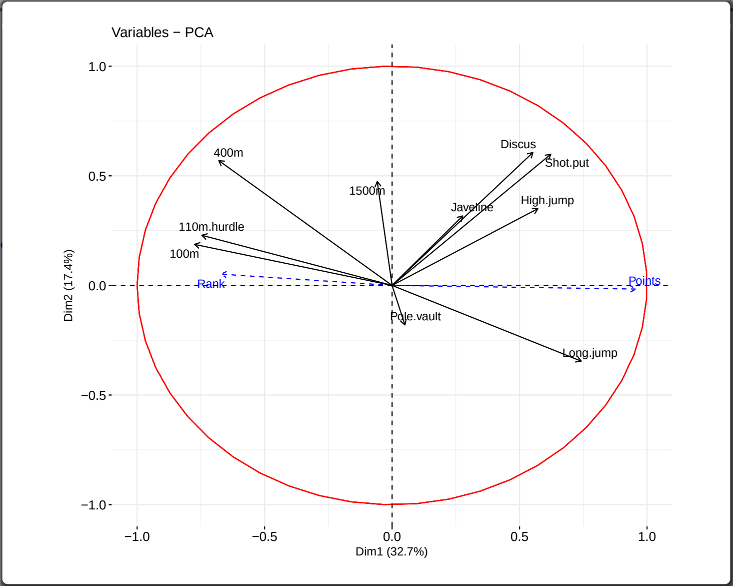


# PCA, Circle of correlations

```
> fviz_pca_var(pc1,axes=c(1,2),repel = T,col.circle = "red")
```

# PCA, Circle of correlations

> fviz_p

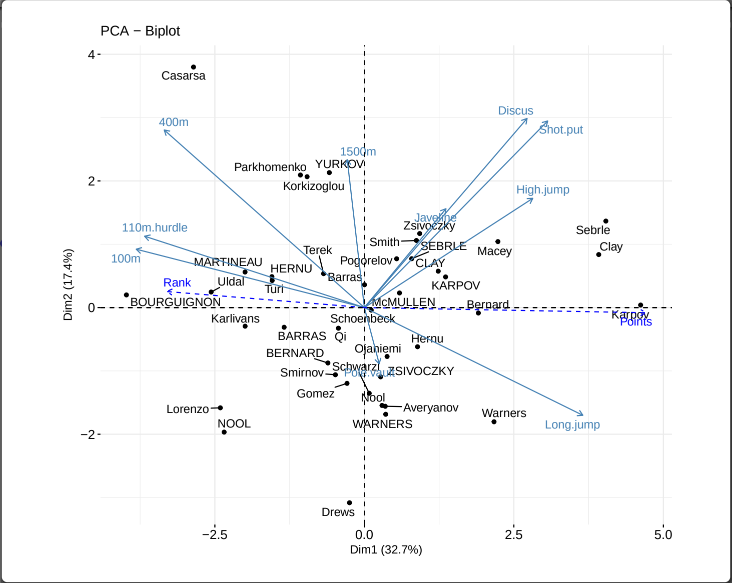


# PCA, Biplot

```
> fviz_pca_biplot(pc1,axes=c(1,2),repel = T)
```

# PCA, Biplot

```
> fviz_p
```



# CA, Correspondence Analysis

```
> library(vcd)
Loading required package: grid
> data("Suicide")
> head(Suicide)
  Freq  sex   method age age.group method2
1    4 male   poison  10     10-20  poison
2    0 male  cookgas  10     10-20     gas
3    0 male toxicgas  10     10-20     gas
4  247 male     hang  10     10-20    hang
5    1 male    drown  10     10-20   drown
6   17 male      gun  10     10-20     gun
> suicide.tab1=xtabs(Freq~sex+method2,data=Suicide)
> suicide.tab1
        method2
sex      poison   gas  hang drown   gun knife  jump other
  male     8917  2089 14740   946  2945   628  1340  2214
  female   8648   318  5637  1703   173   309  1505  1070
> suicide.tab2=xtabs(Freq~age.group+method2,data=Suicide)
> suicide.tab2
         method2
age.group poison  gas hang drown  gun knife jump other
    10-20   2081  375 1736    97  537    58  320   564
    25-35   4495  996 3326   352  916   180  642  1038
    40-50   4689  716 5417   601  927   263  571   839
    55-65   3814  246 5595   886  506   257  661   590
    70-90   2486   74 4303   713  232   179  651   253
> suicide.tab=rbind(suicide.tab2,suicide.tab1)
```

# CA

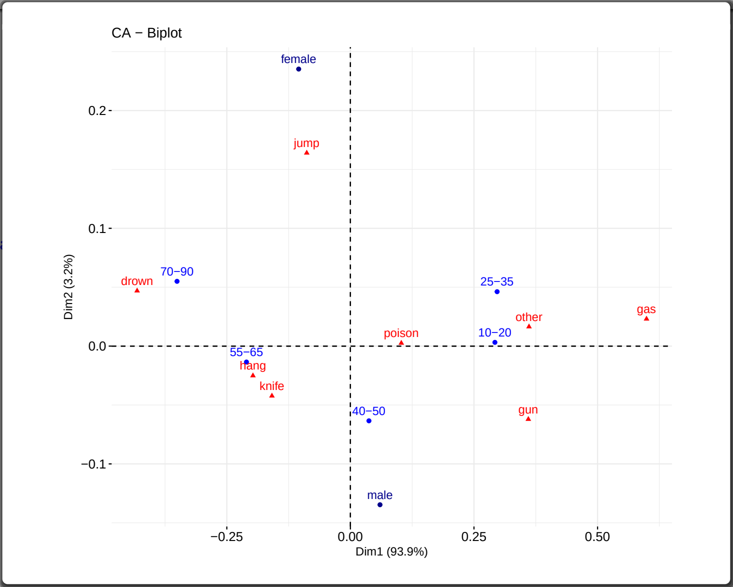
```
> suicide.ca=CA(suicide.tab,row.sup = 6:7,graph = F)
> summary(suicide.ca)

Call:
CA(X = suicide.tab, row.sup = 6:7, graph = F)

The chi square of independence between the two variables is equal to 3422.466
(p-value =  0 ).

Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4
Variance               0.060   0.002   0.001   0.000
% of var.             93.901   3.248   2.298   0.554
Cumulative % of var.  93.901  97.149  99.446 100.000

Rows
          Iner*1000    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
10-20   |    10.361 |  0.292 15.339  0.895 |  0.003  0.053  0.000 | -0.100
25-35   |    20.579 |  0.297 32.748  0.962 |  0.046 22.935  0.023 |  0.037
40-50   |     1.563 |  0.038  0.614  0.237 | -0.063 50.755  0.679 |  0.016
55-65   |    10.683 | -0.210 17.271  0.977 | -0.014  2.064  0.004 | -0.001
70-90   |    21.168 | -0.351 34.028  0.971 |  0.055 24.193  0.024 | -0.009
           ctr   cos2
10-20   73.762  0.105 |
25-35   20.723  0.015 |
40-50    4.607  0.044 |
55-65    0.008  0.000 |
70-90    0.900  0.001 |
```

# CA

```
> fviz_ca_biplot(suicide.ca)
```

# CA

```
> fviz_ca
```



CA − Biplot
female
jump
70−90
drown
25−35
gas
other
poison
10−20
55−65
hang
knife
40−50
gun
male
0.2
0.1
0.0
−0.1
−0.25
0.00
0.25
0.50
Dim1 (93.9%)
Dim2 (3.2%)

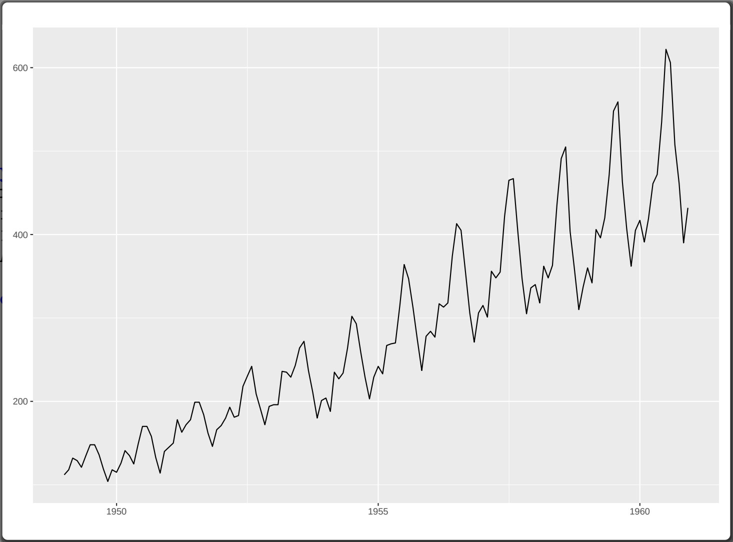
# ggfortify

# Time series

```
> library(ggfortify)
Loading required package: ggplot2
> head(AirPassengers)
[1] 112 118 132 129 121 135
> class(AirPassengers)
[1] "ts"
> autoplot(AirPassengers)
```

# Time series

```
> library
Loading i
> head(Ai
[1] 112 1
> class(A
[1] "ts"
> autoplo
```

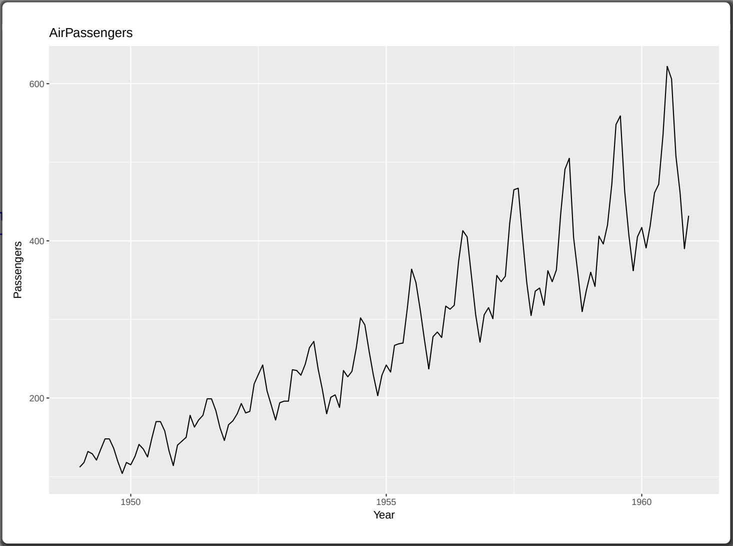


# Time series, Customizing

```
> p <- autoplot(AirPassengers)
> p + ggtitle('AirPassengers') + xlab('Year') + ylab('Passengers')
```

# Time series, Customizing

```
> p <- a
> p + gg
```

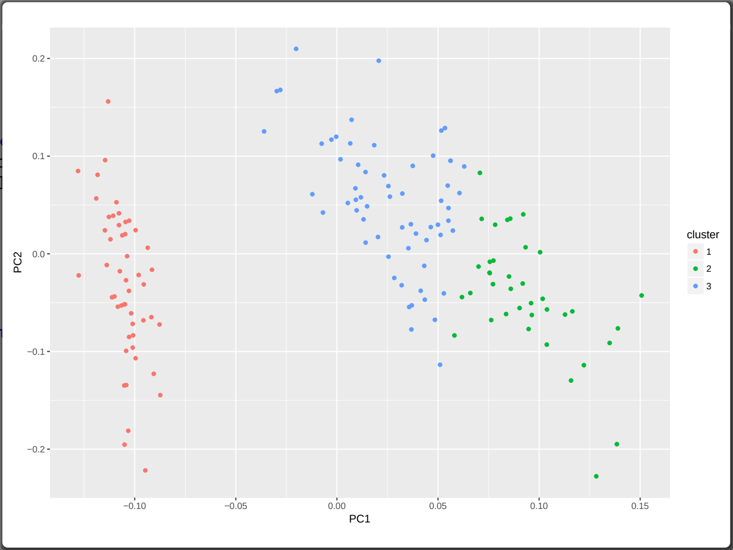


# Clustering

```
> set.seed(1)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> p <- autoplot(kmeans(iris[-5], 3), data = iris)
> p
```

# Clustering

```
> set.se
> head(i
  Sepal.
1
2
3
4
5
6
> p <- a
> p
```

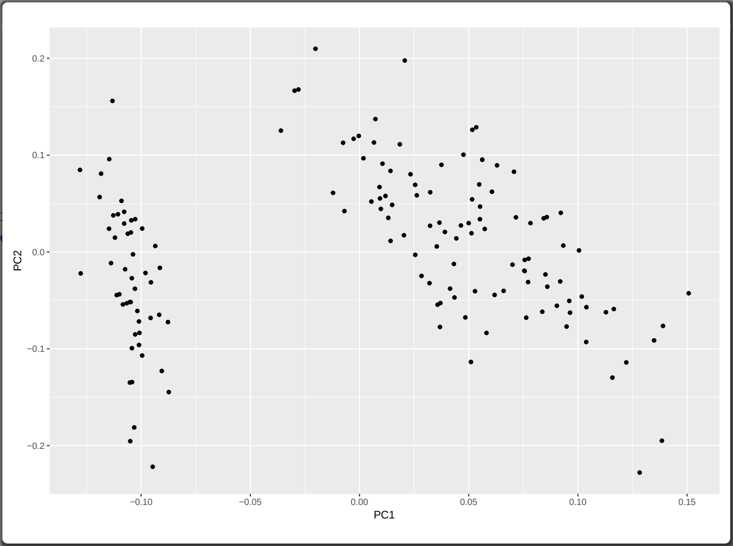


# PCA

```
> df <- iris[c(1, 2, 3, 4)]
> autoplot(prcomp(df))
```

# PCA

```
> df <- i
> autopl
```

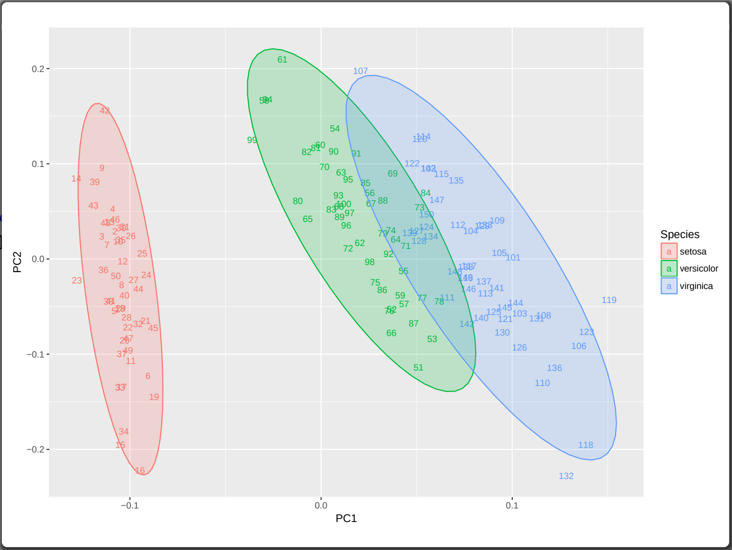


# PCA, by showing groups! Ellipses

```
> autoplot(prcomp(df), data = iris, colour = 'Species',
+     shape = FALSE, label.size = 3, frame=T, frame.type = 'norm')
```

# PCA, by showing groups! Ellipses

```
> autopl
+     sha
```

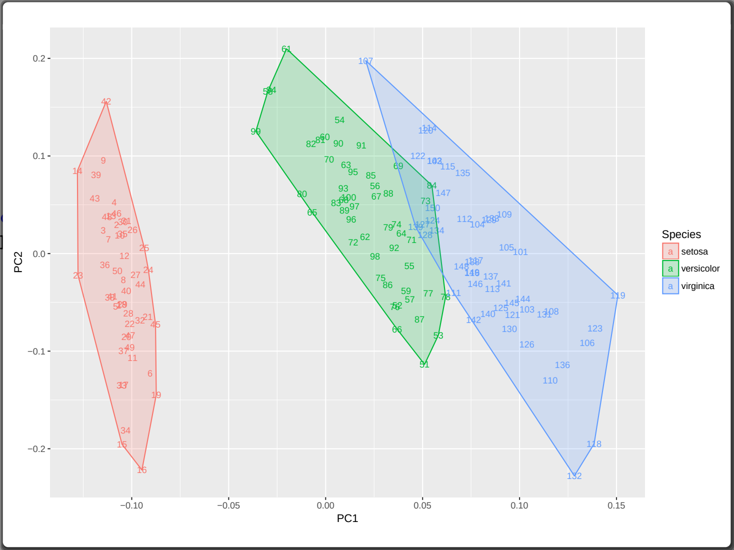


# PCA, by showing groups! Convexes

```
> autoplot(prcomp(df), data = iris, colour = 'Species',
+    shape = FALSE, label.size = 3, frame=T)
```

# PCA, by showing groups! Convexes

```
> autopl
+     sha
```

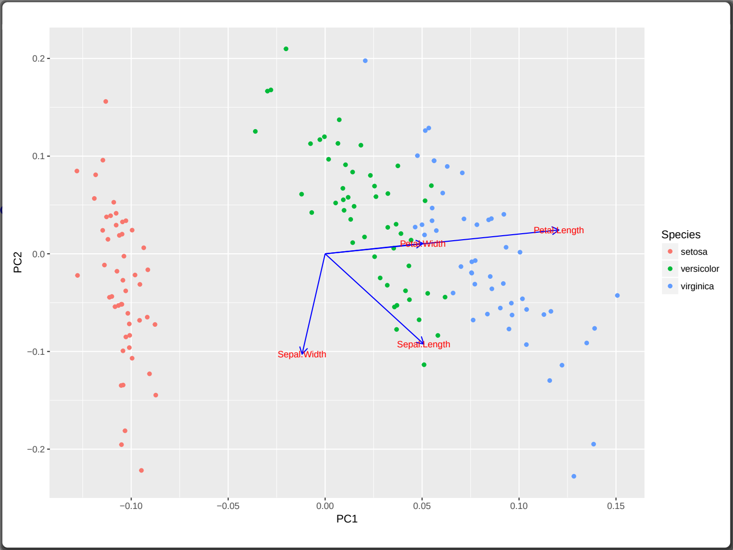


# Biplot for a PCA

```
> autoplot(prcomp(df), data = iris, colour = 'Species',
+           loadings = TRUE, loadings.colour = 'blue',
+           loadings.label = TRUE, loadings.label.size = 3)
```

# Biplot for a PCA

```
> autopl
+
+
```

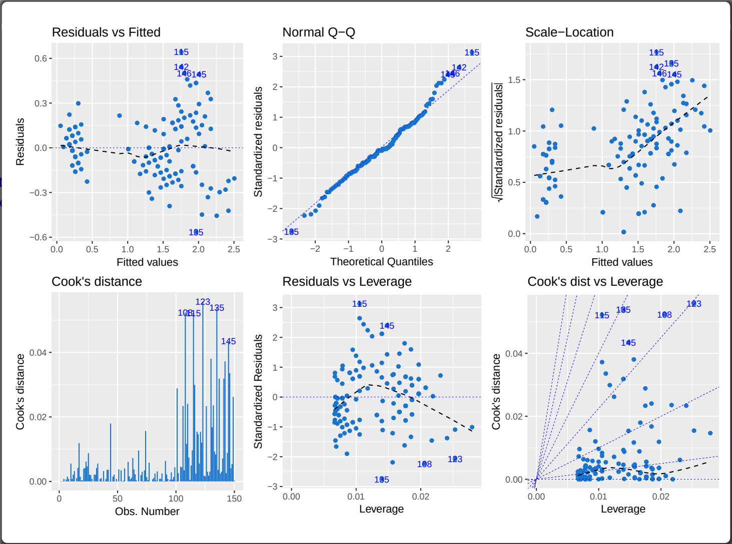


# Regression diagnostic

```
> m <- lm(Petal.Width ~ Petal.Length, data = iris)
> autoplot(m, which = 1:6, colour = 'dodgerblue3',
+          smooth.colour = 'black', smooth.linetype = 'dashed',
+          ad.colour = 'blue',
+          label.size = 3, label.n = 5, label.colour = 'blue',
+          ncol = 3)
```

# Regression diagnostic

```
> m <- lı
> autopl
+
+
+
+
```

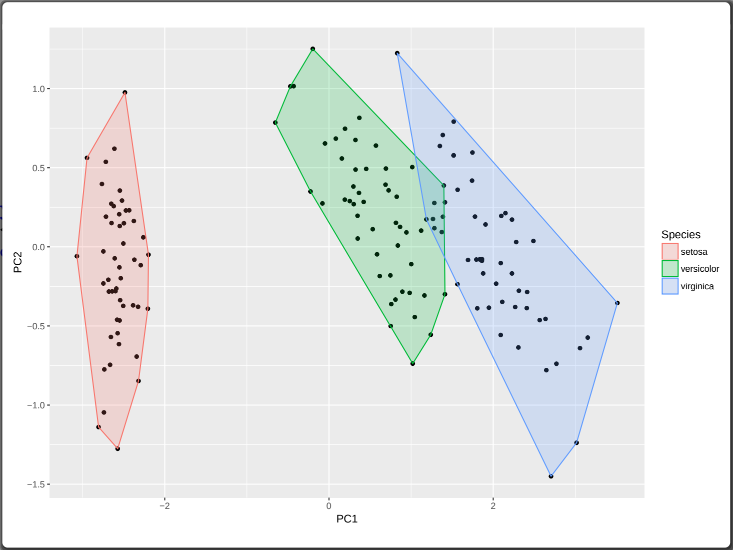


# Local Fisher Discriminant Analysis

```
> library(lfda)
> model <- lfda(x = iris[-5], y = iris[, 5], r = 3, metric="plain")
> autoplot(model, data = iris, frame = TRUE, frame.colour = 'Species')
```

# Local Fisher Discriminant Analysis

```
> library
> model
> autopl
```

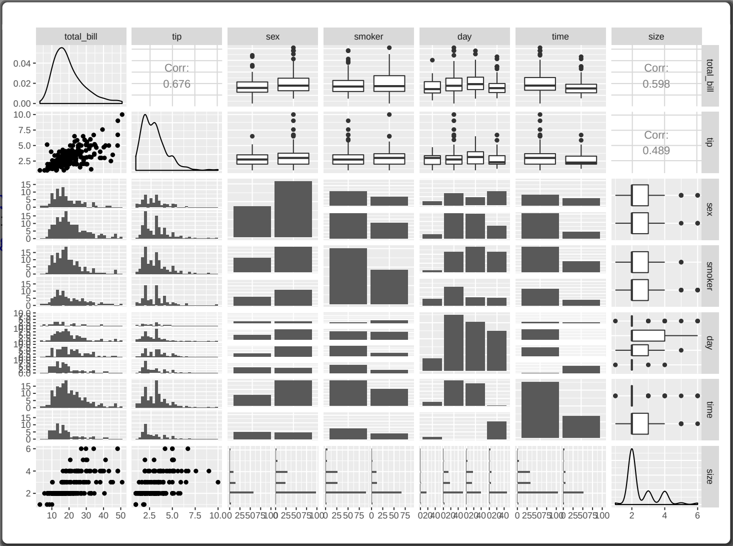


# GGally package, showing the whole data!

```
> library(GGally)
> data(tips, package = "reshape")
> pm <- ggpairs(tips,bins=10)
> pm
```

# GGally package, showing the whole data!

```
> library
> data(ti
> pm <- g
> pm
```

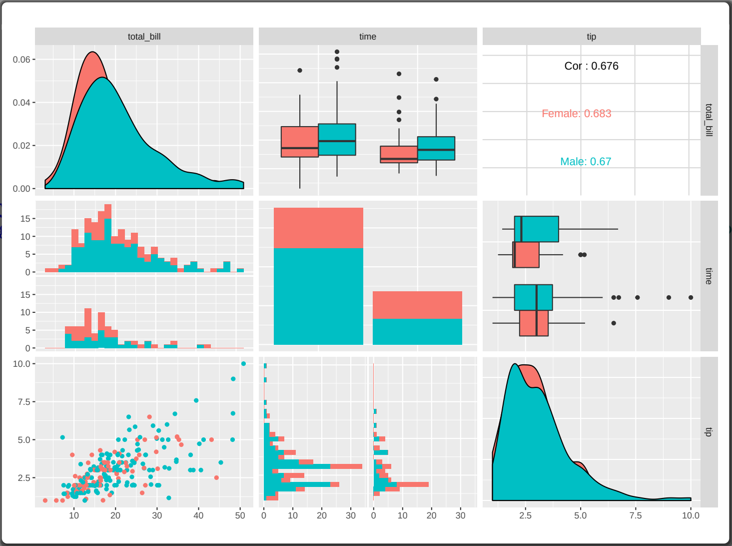


# GGally package, selecting some variables

```
> library(ggplot2)
> pm <- ggpairs(tips, bins=5, mapping = aes(color = sex), columns = c("total_bill",
> pm
```

# GGally package, selecting some variables

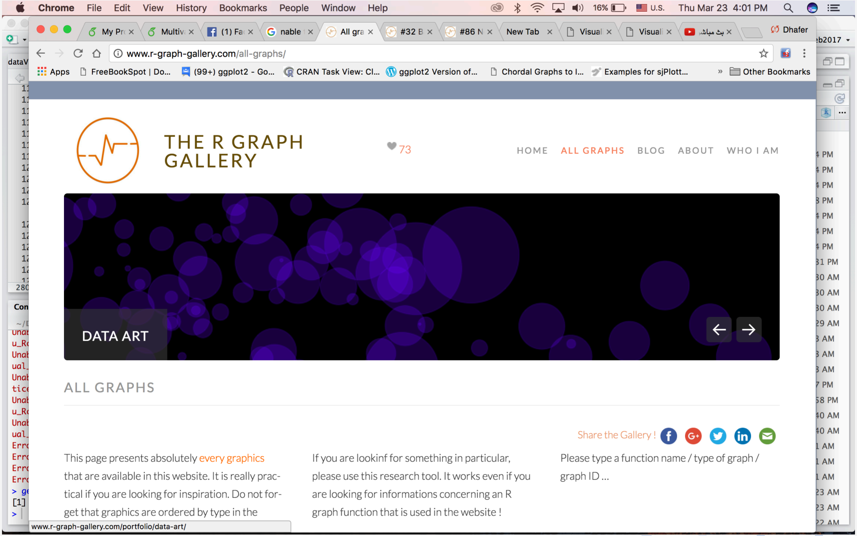
```
> librar
> pm <- g                                           tal_bill",
> pm
```



# Resources

# The R Graph Gallery

http://www.r-graph-gallery.com/all-graphs/



# R for data sciences

http://r4ds.had.co.nz/