R Tutorial

Hai Yang

October 25, 2016

R

R is ‘GNU S’, a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques

- linear and nonlinear modeling
- statistical tests
- time series analysis
- classification
- clustering
- genomics

Download and install R (precompiled binary distribution) from the nearest R mirror site, `https://cran.stat.ucla.edu`

Look and feel of R

```
1+2
```

```
## [1] 3
```

```
2*2
```

```
## [1] 4
```

```
2*2*2*2
```

```
## [1] 16
```

```
2^4
```

```
## [1] 16
```

More calculations in R

```r
log(16, 2)
```

```
## [1] 4
```

```r
pi
```

```
## [1] 3.141593
```

```r
pi*(3^2)
```

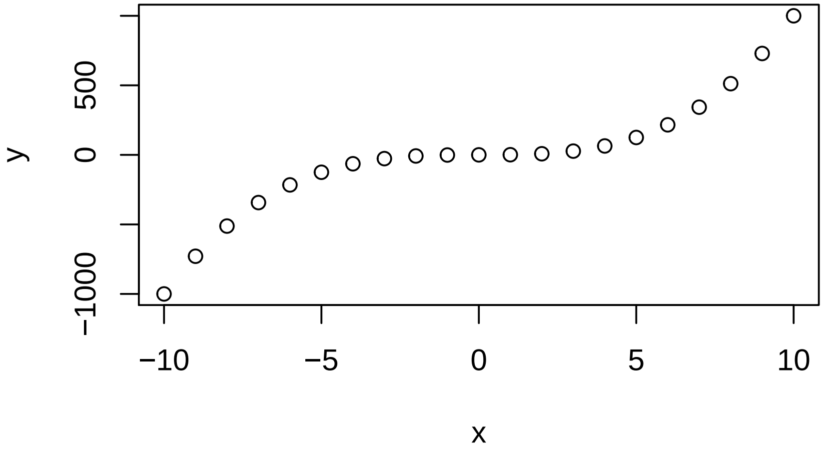
```
## [1] 28.27433
```

```r
sin(pi/2)
```

```
## [1] 1
```

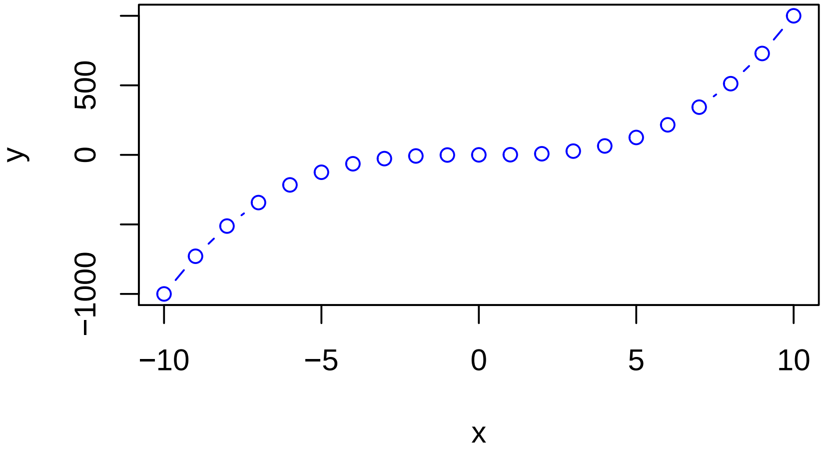
Simple plot in R

```
x = c(-10:10)
y = x^3
plot(x, y)
```



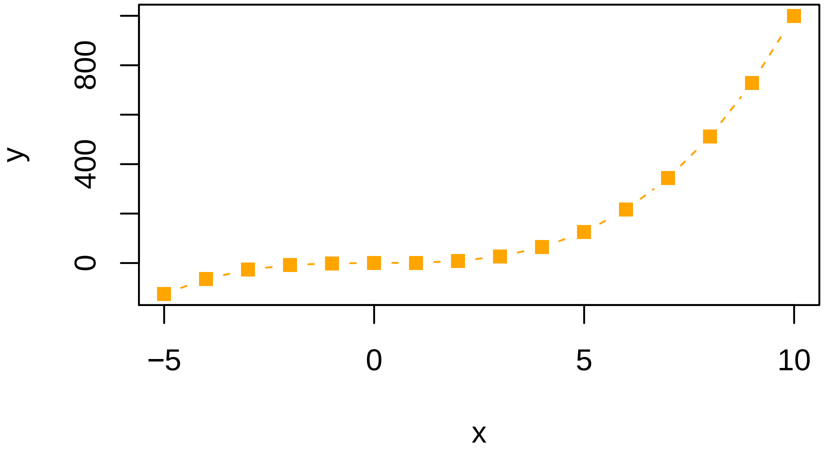
Make the figure fancier by adding more plotting parameters

```r
x = c(-10:10)
y = x^3
plot(x, y, pch=1, lty=1, type="b", col="blue")
```



Try changing parameter values

```
x = c(-5:10)
y = x^3
plot(x, y, pch=15, lty=2, type="b", col="orange")
```



R Studio

RStudio is a set of integrated tools designed to help you be more prudictive with R. It includes

- console
- syntax-highlighting editor that supports direct code execution
- plotting
- history
- debugging
- workspace management.

Download and install RStudio from
`https://www.rstudio.com/products/rstudio/download`

R Studio text editor

R Studio -> File -> New File -> R Script to create an R code file, type commands below in the text file, click the first line, and click Run button three times.

```
x = c(-10:10)
y = x^3
plot(x, y, pch=1, lty=1, type="b", col="blue")
```

R Packages

R packages are collections of functions, data, and complied code in a ready-to-use format with enriched documentation.

Use R Studio to install R packages R Studio -> Tools -> Install Packages… -> type in package name like “ggplot2”

- package name 1 “xlsx”
- package name 2 “foreign”
- package name 3 “ggplot2”

R Objects

R basic command

- Data input
- Data Management
- Basice Statistics
- Basic Graphs

Data input

Data input:

- Importing Data
- Data export
- Viewing Data
- Missing Data

Importing Data

```
## Keyboard input
# create a data frame from scratch
age <- c(25, 30, 56)
gender <- c("male", "female", "male")
weight <- c(160, 110, 220)
mydata <- data.frame(age,gender,weight)

## From web
# mydata <- read.csv("http://www.ats.ucla.edu/stat/data/hdp

## From text file
# mydata <- read.table("/Users/hai/Dropbox (DBMI)/Rtutoria

## From Excel
library(xlsx)
```

```
## Loading required package: rJava
```

Data export

```
# To A Tab Delimited Text File
write.table(mydata, "/Users/hai/Dropbox (DBMI)/Rtutorial/my


# To an Excel Spreadsheet
library(xlsx)
# write.xlsx(mydata, "/Users/hai/Dropbox (DBMI)/Rtutorial/r
```

Viewing Data

```r
# list objects in the working environment
ls()
```

```
## [1] "age"     "gender" "mydata" "weight" "x"      "y"
```

```r
# list the variables in mydata
names(mydata)
```

```
## [1] "Pat_id" "x"      "y"
```

```r
# list the structure of mydata
str(mydata)
```

```
## 'data.frame':    20 obs. of  3 variables:
##  $ Pat_id: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ x     : num  7163 6443 6801 6236 3998 ...
##  $ y     : num  5.27 4.98 5.2 4.87 4.65 ...
```

Missing Data

```
# Testing for Missing Values
x <- c(1,2,NA,3)
is.na(x)
```

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

```
y <- c(1,2,3,NA)
is.na(y)
```

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

```
# Excluding Missing Values from Analyses

mean(x) # returns NA
```

```
## [1] NA
```

```
mean(x, na.rm=TRUE) # returns 2
```

Data Management

Data Management:

- Creating new variables
- Control Structures
- Sorting Data and Merging Data
- Subsetting Data
- Using with( ) and by( )

Creating new variables

```
# Creating new variables
mydata$x <- c(1,2,NA,3)
mydata$y <- c(1,2,3,NA)
mydata$sum <- mydata$x + mydata$y
mydata$mean <- (mydata$x + mydata$y)/2

# Recoding variables
# create 2 age categories
mydata$age <- c(37,49,79,59,49,90,37,49,79,59,49,90,37,49,7
mydata$agecat <- ifelse(mydata$age > 70,
c("older"), c("younger"))
```

Control Structures

- if-else
  if (cond) expr
  if (cond) expr1 else expr2
- for
  for (var in seq) expr
- while
  while (cond) expr
- switch
  switch(expr, ...)
- ifelse
  ifelse(test,yes,no)

Sorting Data and Merging Data

```
# sort by mpg and cyl
attach(mtcars)
newdata <- mtcars[order(mtcars$mpg, mtcars$cyl),]

# Adding Columns
# merge two data frames by ID and Country
mtcars$ID <- rownames(mtcars)
data_frameA <- mtcars[,c(1,2,3,4,5,6,12)]
data_frameB <- mtcars[,c(6,7,8,9,10,11,12)]
total <- merge(data_frameA,data_frameB,by=c("ID","wt"))

# Adding Rows
data_frameA <- head(mtcars)
data_frameB <- tail(mtcars)
total <- rbind(data_frameA, data_frameB)
```

Subsetting Data part 1

```r
# Selecting (Keeping) Variables

#select by variable names "mpg", "cyl", "disp"
mydata <- mtcars
myvars <- c("mpg", "cyl", "disp")
newdata <- mydata[myvars]

# select 1st and 5th thru 10th variables
newdata <- mydata[c(1,5:10)]

# Excluding (DROPPING) Variables
#exclude variables "mpg", "cyl", "disp"
mydata <- mtcars
myvars <- names(mydata) %in% c("mpg", "cyl", "disp")
newdata <- mydata[!myvars]

# exclude 3rd and 5th variable
newdata <- mydata[c(-3,-5)]
```

Subsetting Data part 1

```
# Selecting Observations
#first 5 observations
newdata <- mydata[1:5,]

# based on variable values
newdata <- mydata[ which(gear== 3 & wt > 65),]

# Selection using the Subset Function
#using subset function
newdata <- subset(mydata, mpg >= 20 | mpg < 30,
select=c(ID, wt))
```

Using with( ) and by( )

- With
  with(data, expression)
  example applying a t-test to a data frame mydata
  with(mydata, t.test(y ~ group))
- By
  by(data, factorlist, function)
  example obtain variable means separately for
  each level of byvar in data frame mydata
  by(mydata, mydata$byvar, function(x) mean(x))

Basice Statistics

Basice Statistics
- t-test & Correlations
- ANOVA
- Multiple Regression
- Regression Diagnostics

t-test & Correlations

- Correlations

```
# Correlation matrix from mtcars
# with mpg, cyl, and disp as rows
# and hp, drat, and wt as columns
x <- mtcars[1:3]
y <- mtcars[4:6]
cor(x, y)
```

```
##              hp       drat         wt
## mpg  -0.7761684  0.6811719 -0.8676594
## cyl   0.8324475 -0.6999381  0.7824958
## disp  0.7909486 -0.7102139  0.8879799
```

- t-tests

```
# independent 2-group t-test
t.test(mtcars$mpg, mtcars$wt)
```

ANOVA

```r
# Analysis of Covariance
fit <- aov(mpg ~ wt + cyl, data=mtcars)
fit
```

```
## Call:
##    aov(formula = mpg ~ wt + cyl, data = mtcars)
##
## Terms:
##                       wt      cyl Residuals
## Sum of Squares  847.7252  87.1500  191.1720
## Deg. of Freedom        1        1        29
##
## Residual standard error: 2.567516
## Estimated effects may be unbalanced
```

Multiple Regression

```
## Fitting the Model

# Multiple Linear Regression Example
fit <- lm(mpg ~ wt + cyl + hp, data=mtcars)
summary(fit) # show results
```

```
##
## Call:
## lm(formula = mpg ~ wt + cyl + hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9290 -1.5598 -0.5311  1.1850  5.8986
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 38.75179    1.78686  21.687  < 2e-16 ***
## wt          -3.16697    0.74058  -4.276 0.000199 ***
```

Multiple Regression (cont)

```
## Variable Selection
#Stepwise Regression
library(MASS)
fit <- lm(mpg ~ wt + cyl + hp, data=mtcars)
step <- stepAIC(fit, direction="both")
```

```
## Start:  AIC=62.66
## mpg ~ wt + cyl + hp
##
##        Df Sum of Sq    RSS    AIC
## <none>              176.62 62.665
## - hp    1    14.551 191.17 63.198
## - cyl   1    18.427 195.05 63.840
## - wt    1   115.354 291.98 76.750
```

```
step$anova # display results
```

```
## Stepwise Model Path
```

Regression Diagnostics

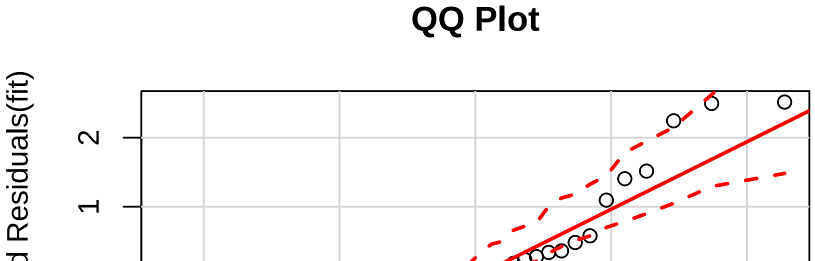
```
## Assume that we are fitting a multiple linear regression

library(car)
fit <- lm(mpg~disp+hp+wt+drat, data=mtcars)
```

Outliers

```r
# Assessing Outliers
outlierTest(fit) # Bonferonni p-value for most extreme obs
```

```
##
## No Studentized residuals with Bonferonni p < 0.05
## Largest |rstudent|:
##                rstudent unadjusted p-value Bonferonni p
## Toyota Corolla  2.51597           0.01838      0.58816
```

```r
qqPlot(fit, main="QQ Plot") #qq plot for studentized resid
```

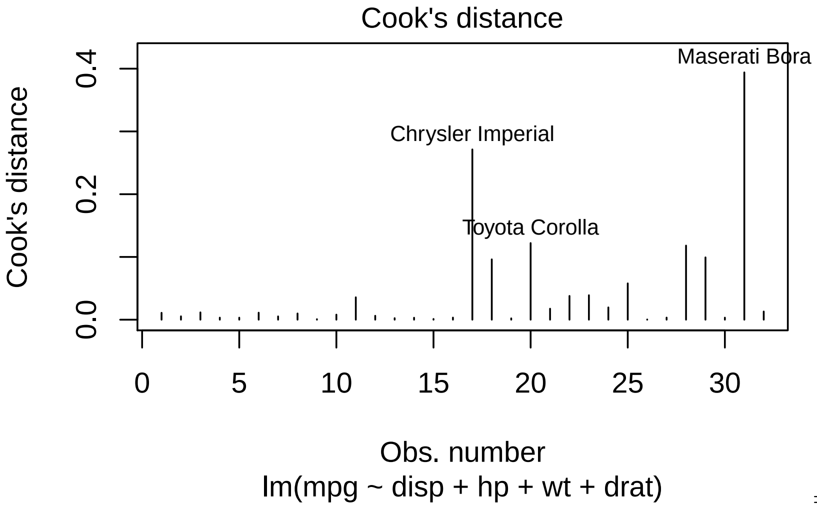


Influential Observations

```
# added variable plots
###av.Plots(fit)
```

Cook's D plot

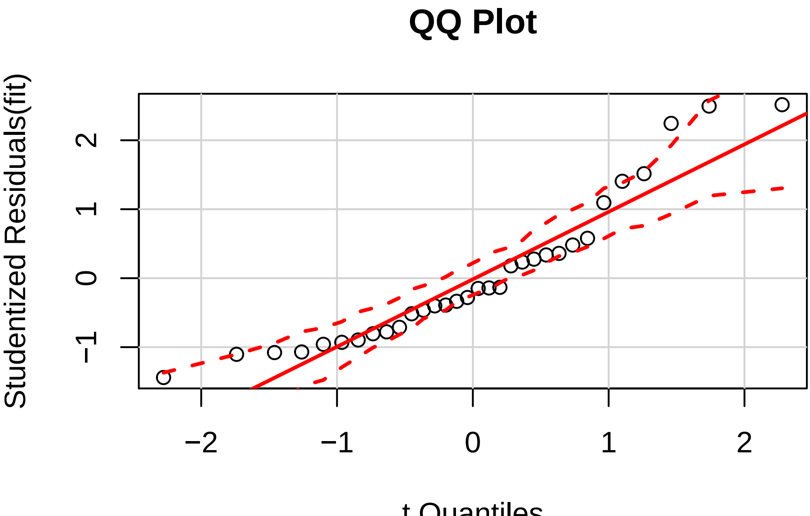
```
# identify D values > 4/(n-k-1)
cutoff <- 4/((nrow(mtcars)-length(fit$coefficients)-2))
plot(fit, which=4, cook.levels=cutoff)
```



##

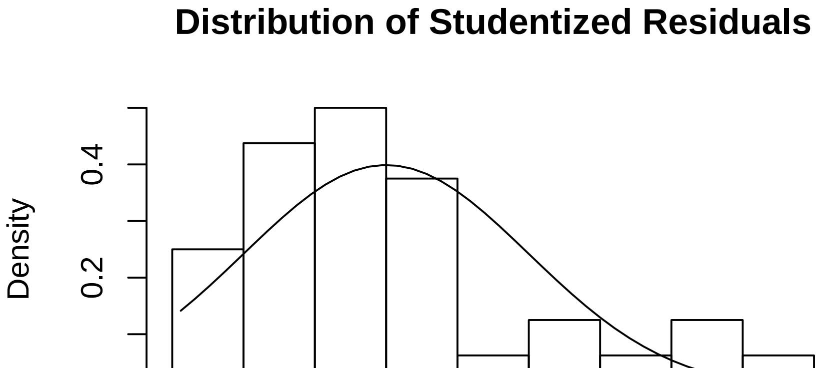
Non-normality

```
# Normality of Residuals
# qq plot for studentized resid
qqPlot(fit, main="QQ Plot")
```



distribution of studentized residuals

```r
library(MASS)
sresid <- studres(fit)
hist(sresid, freq=FALSE,
   main="Distribution of Studentized Residuals")
xfit<-seq(min(sresid),max(sresid),length=40)
yfit<-dnorm(xfit)
lines(xfit, yfit)
```



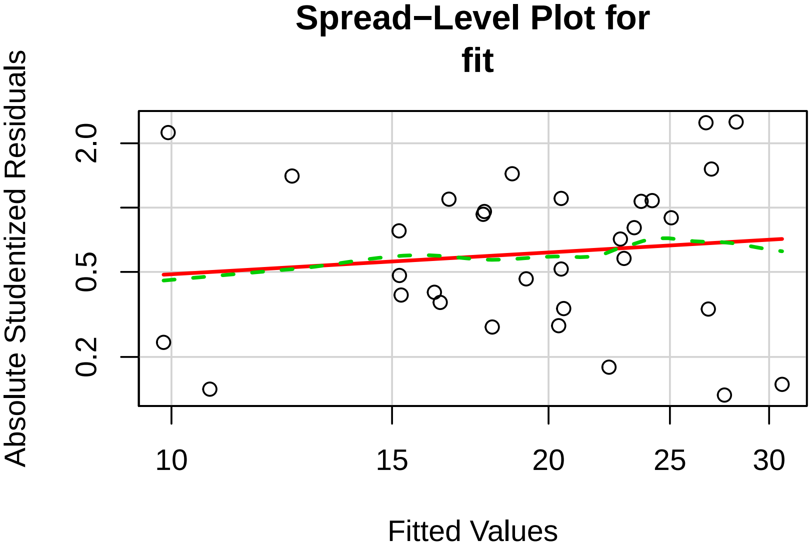
Non-constant Error Variance

```r
# Evaluate homoscedasticity
# non-constant error variance test
ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.429672    Df = 1     p = 0.231818
```

plot studentized residuals vs. fitted values

```
spreadLevelPlot(fit)
```



Multi-collinearity

```r
# Evaluate Collinearity
vif(fit) # variance inflation factors
```

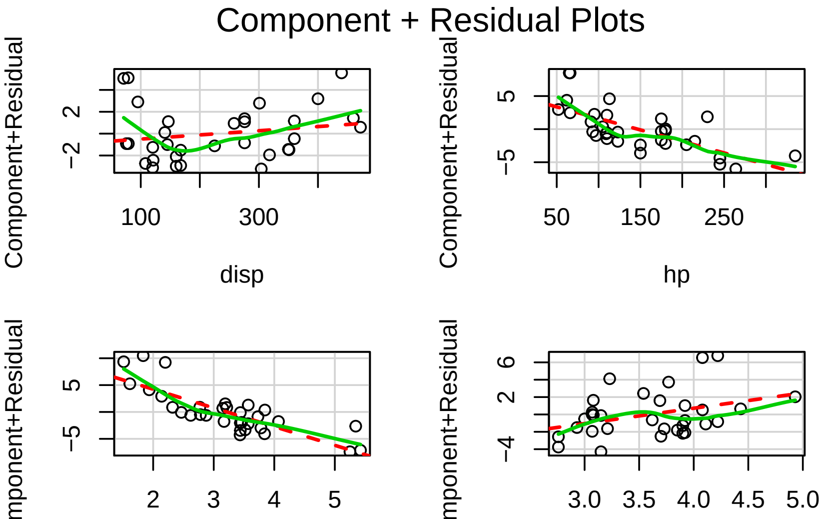
```
##     disp       hp       wt     drat
## 8.209402 2.894373 5.096601 2.279547
```

```r
sqrt(vif(fit)) > 2
```

```
##  disp    hp    wt  drat
##  TRUE FALSE  TRUE FALSE
```

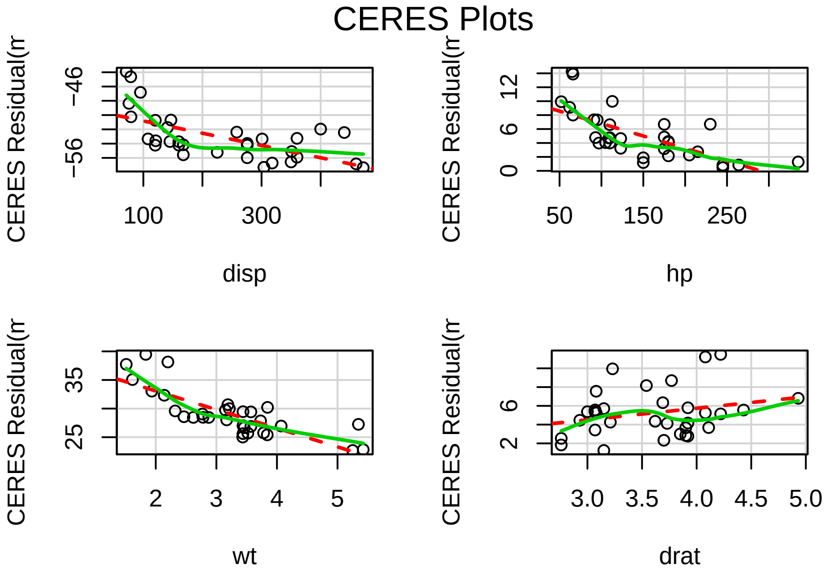
Nonlinearity

```
# Evaluate Nonlinearity
# component + residual plot
crPlots(fit)
```



Ceres plots

```
ceresPlots(fit)
```



Non-independence of Errors

```r
# Test for Autocorrelated Errors
durbinWatsonTest(fit)
```

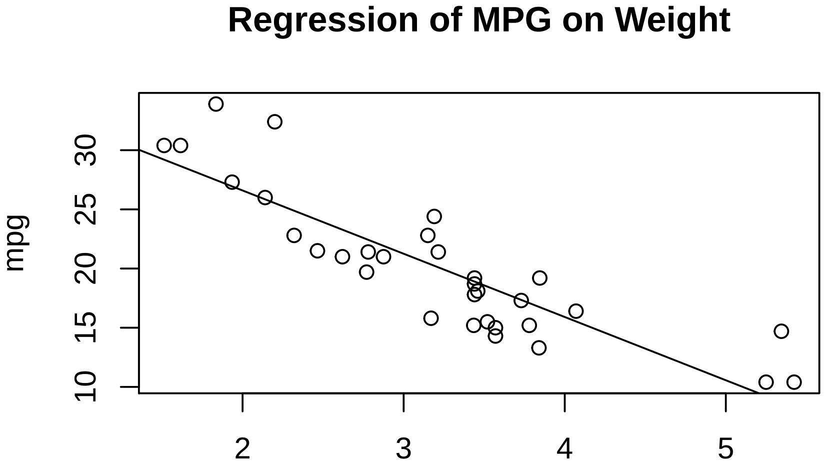
```
##  lag Autocorrelation D-W Statistic p-value
##    1        0.100862      1.735915   0.284
##  Alternative hypothesis: rho != 0
```

Basic Graphs

Basic Graphs:
- Creating a Graph
- Density Plots
- Dot Plots
- Bar Plots
- Boxplots
- Scatter Plots

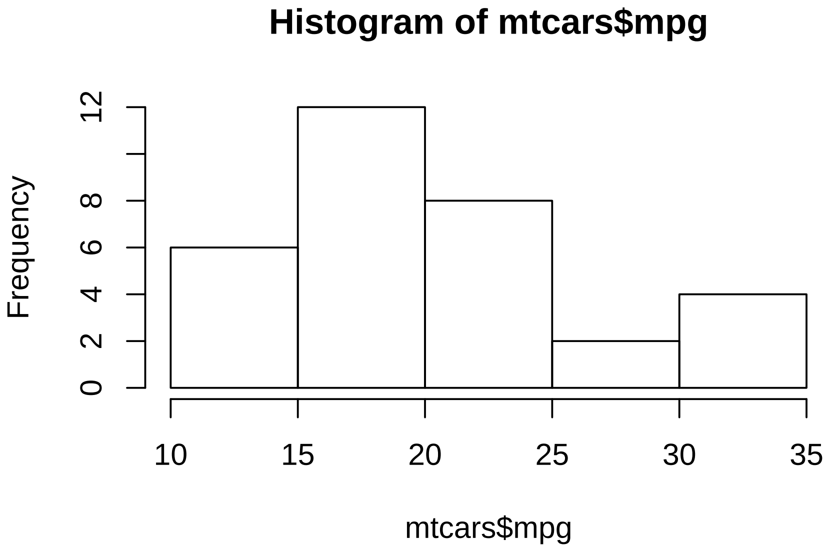
Creating a Graph

```r
with(mtcars, plot(wt, mpg) )
abline(lm(mpg~wt))
title("Regression of MPG on Weight")
```



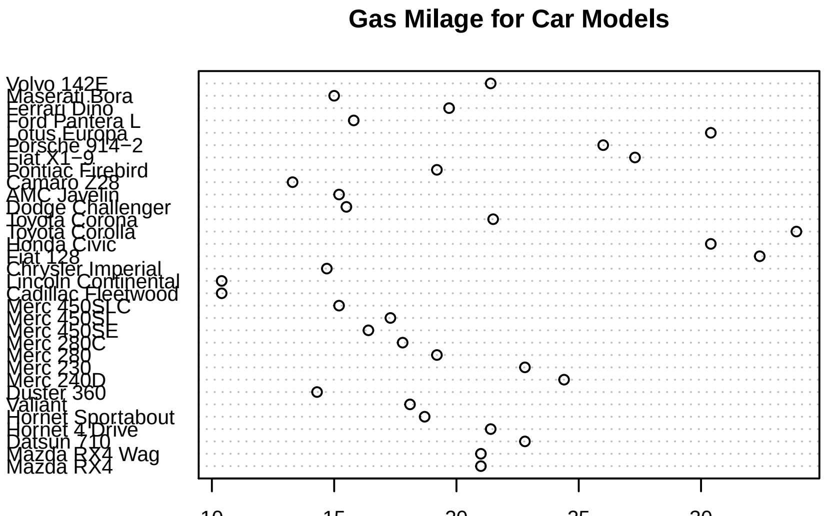
Density Plots

```r
hist(mtcars$mpg)
```



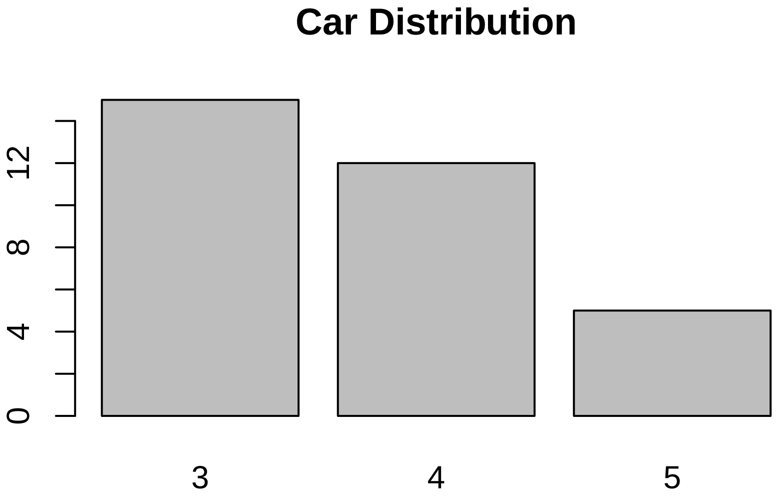
Dot Plots

```
dotchart(mtcars$mpg,labels=row.names(mtcars),cex=.7,
    main="Gas Milage for Car Models",
   xlab="Miles Per Gallon")
```



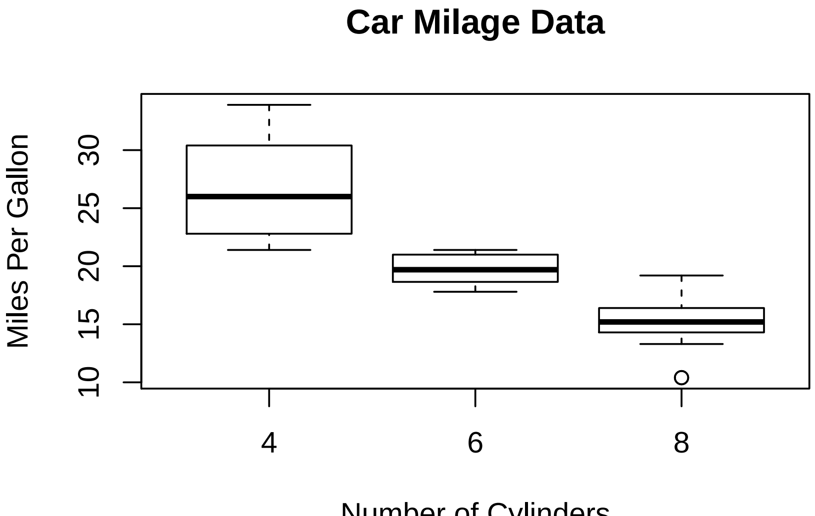
Bar Plots

```
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution",
    xlab="Number of Gears")
```



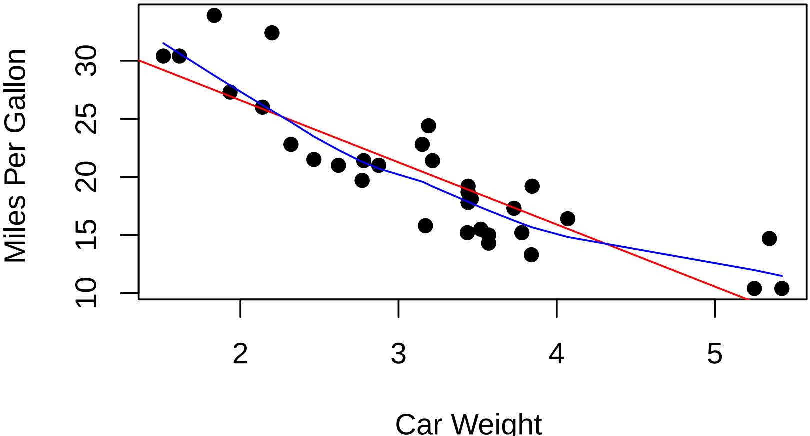
Boxplots

```
# Boxplot of MPG by Car Cylinders
boxplot(mpg~cyl,data=mtcars, main="Car Milage Data",
    xlab="Number of Cylinders", ylab="Miles Per Gallon")
```



Scatter Plots

```
plot(wt, mpg,
    xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
#Add fit lines
abline(lm(mpg~wt), col="red") # regression line (y~x)
lines(lowess(wt,mpg), col="blue") # lowess line (x,y)
```



Scatterplot Matrices

```r
pairs(~mpg+disp+drat+wt,data=mtcars)
```