

Basic guidelines to understand the R Manual

NEW TOPIC: Each new topic starts

1. Data Input : New sub-topic

For text file: Key points and class notes summary

```
a <- read.csv("C:Users/Desktop/abc.csv", header=TRUE): R command input  
abc.csv", header=TRUE): R output
```

```
# read.csv: R command explanation
```

***note: any notes from class lectures or additional key points*

INTRODUCTION TO R

1. Download R

<http://www.r-project.org/>

Some facts about R

- Completely FREE!!!
- R is case sensitive.
- For certain functions you have to install and load packages

2. Data Input

```
a <- read.csv("C:/Users/Desktop/abc.csv", header=TRUE)
```

***note: You are inputting the excel file named abc located on desktop in R and naming it a.
<- is similar to = function. R needs this!!!*

For text file use: `read.table ("filename.txt", header=TRUE)`

3. Basic Commands in R

Data entry and manipulation: (x can be any of several types; y and z are vectors)

<i>Command</i>	<i>Meaning</i>
<code>x<-c(1, 2, 3, 4)</code>	Create a vector of numbers
<code>x</code>	Prints contents of x
<code>attach(x)</code>	recognize the file x
<code>names(x)</code>	shows all the variable names in file x
<code>objects(x)</code>	provides a list of all objects in file x
<code>matrix(x)</code>	creating a matrix from file x with 2 rows and 3 columns
<code>y[2:5]</code>	Returns 2 nd to 5 th elements of vector y
<code>y[-3]</code>	Returns a vector of all elements in y except for the 3 rd
<code>y[y<10]</code>	Sub-vector of all entries in y less than 10
<code>z[y<10]</code>	Sub-vector of all entries in z for which the corresponding entries in y are less than 10 (x & y must be same length)
<code>x<-list(y,z)</code> <code>x\$y</code> <code>x\$z</code>	Construct of list with two vectors in it Returns vector y Returns vector z
<code>x<-data.frame(y,z)</code> <code>x\$y</code> <code>x\$z</code>	Construct of dataframe* with two vectors in it Returns vector y Returns vector z
<code>x<-factor(y)</code>	Converts numeric type y into a factor
<code>is.factor(y)</code>	Returns "TRUE" if y contains factors (numeric or symbolic)
<code>is.numeric(y)</code>	Returns "TRUE" if y contains numeric data
<code>is.na(y)</code>	Returns "TRUE" for each entry
<code>dimnames(x)</code>	Lists the different attributes of an array or dataframe
<code>levels(x)=c("a", "b", ...)</code>	Assign names to each factor value
<code>load("filename")</code>	Loads R data from <i>filename</i>
<code>save(x, "filename")</code>	Saves R object x into <i>filename</i>

Descriptive statistics: (x can be a vector or a data frame; y and z are vectors)

<i>Command</i>	<i>Meaning</i>
mean(x)	Calculate mean of vector x (or of all vectors in data frame x)
median(x)	Calculate median of vector x (“)
sd(x)	Calculate standard deviation of vector x (“)
var(x)	Calculate variance of vector x (“)
summary(x)	Calculate summary of vector x (“)
boxplot(x)	Create boxplot of vector x (“)
boxplot(x~y)	Create multiple boxplots of data in x, based on categories in y.
stripchart(x)	Create stripchart of vector x (“)
stripchart(x~y)	Create multiple stripcharts of data in x, based on categories in y.
hist(y)	Create histogram of vector
qqnorm(y)	Creates a “normal quantile-quantile” plot of y
plot(z~y)	Makes an “x-y” plot of vector z vs. vector y

Data analysis: (x,y,z are numeric vectors, f1, f2 are factors, M is a model)

<i>Command</i>	<i>Meaning</i>
t.test(x)	T-test
chisq.test	Chi-Square
aov(x~f1)	Analysis of Variance
lm(x~f1+f2+...fy)	Linear Model
lrm(x~f1+f2+...fy)	Logistic Linear Model
glm(x~f1+f2+...fy)	General Linear Model
summary(x)	Short description of model
anova(x)	ANOVA table
coef(x)	Fitted coefficients
resid(x)	Residuals
predict(x, newdata)	Predictions based on fitted Model

R system controls: Commands in square brackets ([]) are accessed through the R menus

<i>Command</i>	<i>Meaning</i>
help(command)	Opens a new window describing how to use command
library(package)	Make commands from a package available in R session
data(dataset)	Make load data items from dataset into R session
rm(x)	Remove R object x from current R session
rm(list=ls())	Remove all R objects from current R session

T-TEST

1. 1 sample t-test

Example #1

: A random sample of 50 golf balls of Brand X were hit by a Robot-Driver. Can we conclude that the mean distance obtained exceeds 300 yards?

(use data file GolfTest.csv)

Example #1:

- o $H_0 : \mu \leq 300$
- o $H_1 : \mu > 300$

Yards

303	317	284	305	300
282	297	290	277	293
289	308	304	278	300
298	317	290	301	276
283	293	311	304	318
303	295	299	296	315
309	294	303	285	292
293	291	299	288	303
316	297	282	279	301
302	300	318	310	292

```
golf <- read.csv("M:/TEACH/DataMining_Fall2009/data /GolfTest.csv",  
header=TRUE)
```

```
names(golf)
```

```
[1] "Yards"
```

```
attach(golf)
```

```
t1<- t.test(Yards, mu = 300, alt = "greater")
```

```
t1
```

```
qqnorm(Yards)
```

#names: R will bring the variable names from data file

#attach: R will recognize the data file

#t.test: R will perform a t test

#qqnorm: R will produce a qq plot for testing normality (assumption of t-test)

One Sample t-test

data: Yards

t = -1.5022, df = 49, p-value = 0.9303 (Null NOT rejected)

alternative hypothesis: true mean is greater than 300

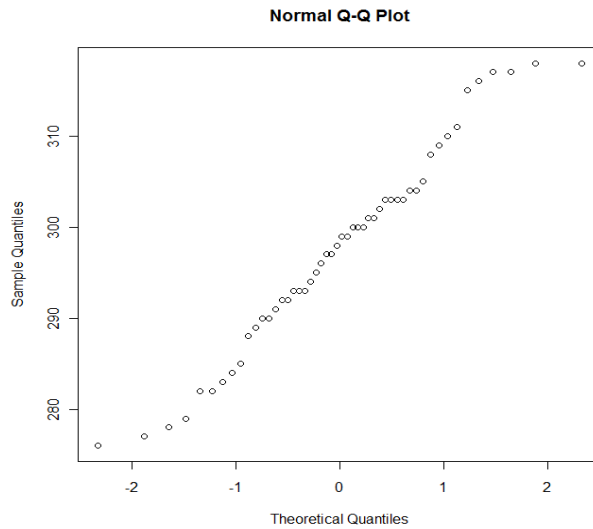
95 percent confidence interval:

294.9214 Inf

sample estimates:

mean of x

297.6



2. 2 sample t-test

Example #2

: The following table shows effectiveness of two brands of car wax. Compare the two brands in terms of mean wax effectiveness at test size 5%.

(use data file wax_effectiveness.csv)

- $H_0 : \mu_{\text{Sureglow}} = \mu_{\text{Mirrorsheen}}$
- $H_1 : \mu_{\text{Sureglow}} \neq \mu_{\text{Mirrorsheen}}$

Type	Effectiveness	Type	Effectiveness
Sureglow	93	Mirrorsheen	90
Sureglow	96	Mirrorsheen	97
Sureglow	87	Mirrorsheen	91
Sureglow	91	Mirrorsheen	94
Sureglow	88	Mirrorsheen	100
Sureglow	85	Mirrorsheen	95
Sureglow	88	Mirrorsheen	88
Sureglow	91	Mirrorsheen	92
Sureglow	82	Mirrorsheen	94
Sureglow	91	Mirrorsheen	89
Sureglow	86	Mirrorsheen	96
Sureglow	93	Mirrorsheen	91
Sureglow	91	Mirrorsheen	97
Sureglow	87	Mirrorsheen	92
Sureglow	88	Mirrorsheen	92

```
wax <-
read.csv("M:/TEACH/DataMining_Fall2009/data/wax_effectiveness.csv")
attach(wax)
x1 <- Effectiveness[Type=="Sureglow"]
x2 <- Effectiveness[Type=="Mirrorsheen"]
t2 <- t.test(x1, x2, var.equal = TRUE, alt = "two.sided")
t2

# var.equal = TRUE: equal sample size
# alt = "two.sided": indicates alternative is two sided
**note: you can either attach file (wax) and name x1 and x2

x1 <- wax$Effectiveness[wax$Type=="Sureglow"]
x2 <- wax$Effectiveness[wax$Type=="Mirrorsheen"]
**note: or either use $ to recognize the file

data: x1 and x2
t = -3.2048, df = 28, p-value = 0.003364 (Null rejected)
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval: -6.665932 -1.467401
sample estimates:
mean of x mean of y
89.13333 93.20000
```

3. Paired t-test

Example #3

: Test if average fish price has gone up.

- o $H_0 : \mu_{1970} = \mu_{1980}$,
- o $H_1 : \mu_{1970} < \mu_{1980}$

CaseNo	Type_Fish	Price_1970	Price_1980
1	COD	13.1	27.3
2	FLOUNDER	15.3	42.4
3	HADDOCK	25.8	38.7
4	MENHADEN	1.8	4.5
5	OCEAN PERCH	4.9	23.0
6	SALMON, CHINOOK	55.4	166.3
7	SALMON, COHO	39.3	109.7
8	TUNA, ALBACORE	26.7	80.1
9	CLAMS, SOFT-SHELLED	47.5	150.7
10	CLAMS, HARD-SHELLED	6.6	20.3
11	LOBSTERS, AMERICAN	94.7	189.7
12	OYSTERS, EASTERN	61.1	131.3
13	SEA SCALLOPS	135.6	404.2
14	SHRIMP	47.6	149.0

```
fish <- read.csv("M:/TEACH/DataMining_Fall2009/data/fish_prices.csv")
attach(fish)
t3 <- t.test(Price_1970, Price_1980, paired = TRUE, alt = "less")
t3
```

```
# paired = TRUE: R will perform T-test as a paired test.
# alt = "less": indicates alternative is less than
```

```
      Paired t-test
data:  Price_1970 and Price_1980
t = -3.7017, df = 13, p-value = 0.001331 (Null rejected)
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -35.83333
sample estimates:
mean of the differences
      -68.7
```

TEST FOR PROPORTIONS

Confidence Intervals and Test for Proportions in R

1. Single proportion

Example #1

: A large casino has collected data on a sample of 12844 loyal customers, and classified each as a "high roller" or not.

(use data file hi_rollers.txt)

- Calculate a 90% confidence interval for the proportion of 'high rollers' in the loyal customers database of this casino.
- Test if the true proportion of 'high rollers' > 0.15 . ($H_0: p = .15$ vs $> .15$)
 - `prop.test`
 - `binom.test`

```
hr<- read.table("G:/TEACH/DataMining_Fall2009/Data/Hi_rollers.txt")
y <- table(hr)
y
x
0      1
11434 1410
```

```
prop.test(1410,11434+1410, p = .15, alt = "greater", conf.level=.9)
# prop.test: run approximate z-test for binomial proportion
# conf.level=.9: confidence level 90%
**note: R needs x = number of successes , n = total number of trials for prop.test
```

```
1-sample proportions test with continuity correction
data: 1410 out of 11434 + 1410, null probability 0.15
X-squared = 55190.89, df = 1, p-value < 2.2e-16 (Null rejected)
alternative hypothesis: true p is greater than 0.15
90 percent confidence interval:
0.8865966 1.0000000
sample estimates:
p
0.8902211
```

```
binom.test(y, p=.15, conf.level=.9, alt="greater")
# binom.test: run exact binomial test
```

Exact binomial test

```
data: y
number of successes = 11434, number of trials = 12844, p-value <2.2e-16
alternative hypothesis: true probability of success is greater than
0.15
```

```

90 percent confidence interval:
 0.8866032 1.0000000
sample estimates:
probability of failure
      0.8902211

```

2. Comparison of two proportions

Example #2

: A blackjack player has collected the data on two blackjack dealers in one casino :

n = # hands dealt, x = # of blackjacks dealt to player

	n	x
Dealer 1	1000	59
Dealer 2	1200	61

The player wants to know if the proportion of blackjacks dealt by the two dealers are equal.

- o $H_0 : p_1 = p_2$
- o $H_1 : p_1 \neq p_2$

```

prop.test(c(59, 61), c(1000, 1200), alt="two.sided")
#c: concatenation

```

```

2-sample test for equality of proportions with continuity correction
data:  c(59, 61) out of c(1000, 1200)
X-squared = 0.5559, df = 1, p-value = 0.4559 (Null NOT rejected)
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.01192630  0.02825963
sample estimates:
  prop 1      prop 2
0.05900000 0.05083333

```

3. Chi-square

Example #3

: In planning her campaign strategy to determine how to approach a particular issue, a congressional candidate wants to know if there are any differences in the proportion of voters who favor the issue among her rural, suburban, and urban constituents. She has collected sample opinions and has obtained the following results: (use data file campaign.csv)

- o H_0 : P(In Favor) is same for Rural, Suburban, and Urban constituents.
- o H_a : H_0 is false.

	Rural	Suburban	Urban
In favor	65	63	52
Not in favor	35	37	48

```
campaign <-
read.csv("K:/TEACH/DataMining_Fall2009/Data/campaign.csv",header=FALSE)
campaign
  V1 V2 V3
1 65 63 52
2 35 37 48
```

```
chix <- chisq.test(x)
chix
```

chisq.test(): R will perform chi-square test

```
      Pearson's Chi-squared test
data:  x
X-squared = 4.0833, df = 2, p-value = 0.1298 (Null NOT rejected)
```

4. Chi-square Independence

Example #4

: For the following data, test if Age and GAME PREFERENCE are independent.
(use data file game_preference.csv)

Game	21-25	26-50	over 50
Multi-Line Slots	15	37	16
Video Poker	25	25	17
Wheel of Fortune	14	40	27
Sports Book	11	4	1
Blackjack	9	23	14
Megabucks	3	8	1

- H_0 : Age and game preference is independent.
- H_a : Age and game preference is not independent.

```
gp <-
read.csv("K:/TEACH/DataMining_Fall2009/Data/game_preference.csv",header
=FALSE)
chixx <- chisq.test(xx)
Warning message:
In chisq.test(xx) : Chi-squared approximation may be incorrect
```

```

chixx
      Pearson's Chi-squared test
data:  xx
X-squared = 28.5528, df = 10, p-value = 0.001471 (NULL rejected)

```

5. Chi-square Goodness of Fit (Homogeneity of proportions)

Example #5

: You are given the results of 360 rolls of a pair of fair dice (data file Pair-of_Dice.csv). Test the hypothesis that the two dice are fair.

(use data file Pair_of_Dice.csv)

- o H_0 : all die are fair.
- o H_a : at least one is not fair.

```

pod <-
read.csv("K:/TEACH/DataMining_Fall2009/Data/Pair_of_Dice.csv",header=TRUE)
ty <- table(pod)
ty
y
 2  3  4  5  6  7  8  9 10 11 12
14 22 23 39 52 75 48 33 31 14  9
tt <- as.matrix(ty)
chi <- chisq.test(tt, p=c(1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36,
4/36, 3/36, 2/36, 1/36))
#p=probability, c=concatenation

```

Chi

```

      Chi-squared test for given probabilities
data:  tt
X-squared = 10.5267, df = 10, p-value = 0.3956 (NULL not rejected)

```

ANOVA

1. One-Way ANOVA

Example #1

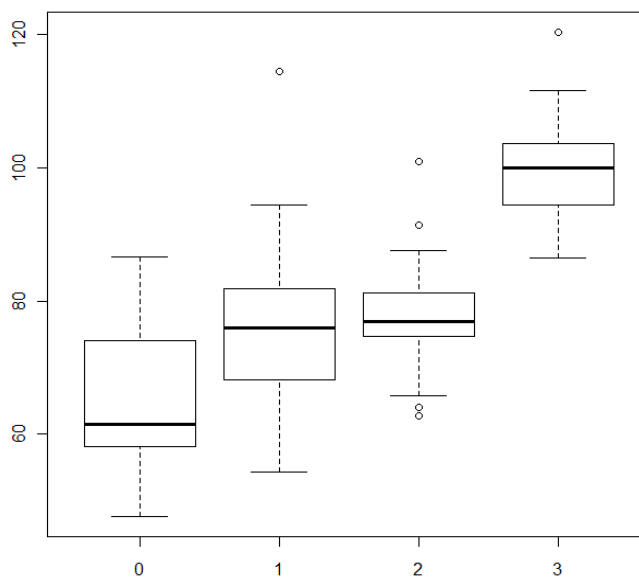
: Test if the average sales of the restaurant after 4 different promotions are equal.

(use data file restaurant4anova.csv)

- $H_0: \mu_1 = \mu_2 = \mu_3 = \mu_4$
- H_a : at least 1 μ is not the same.

No_promo	47.6032	48.9218	86.6995	60.9114	74.0193	58.4517	61.9995	81.6082	58.1103	73.6343
Promo1	83.243	114.432	74.916	70.12	80.039	66.66	70.878	94.427	68.167	82.858
	66.315	70.534	62.506	80.31	75.958	74.315	65.305	91.489	54.258	90.411
	75.991	81.846	81.554	80.38	64.139					
Promo2	79.674	79.314	80.435	80.187	91.444	70.974	75.036	72.623	72.995	75.494
	84.144	75.132	76.865	83.022	82.752	75.295	81.229	65.832	74.65	75.205
	78.09	62.722	64.07	100.902	87.548					
Promo3	98.504	96.154	95.688	111.455	105.033	100.643	98.913	101.709	111.569	86.526
	87.927	103.611	108.959	108.406	100.002	90.109	120.436	90.245	101.11	88.961
	94.453	100.07	91.999	102.255	102.313					

```
restaurant <-  
read.csv("K:/TEACH/DataMining_Fall2009/Data/restaurant4anova.csv",  
header=TRUE)  
attach(restaurant)  
boxplot(Sales~Promo)
```



```
outanova1 <- aov(Sales~Promo)
outanova1
summary(outanova1)
```

#boxplot: R will produce a boxplot graph

#aov: analysis of variance, DV ~ IV (DV: Sales, IV: Promo)

(outanova1: just by typing the file name will print your results)

#summary(): will give you a table of summary of outcome

Call:

```
aov(formula = Sales ~ Promo)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Promo	1	9984.5	9984.5	78.939	1.111e-13 *** (Null Rejected)
Residuals	83	10498.1	126.5		

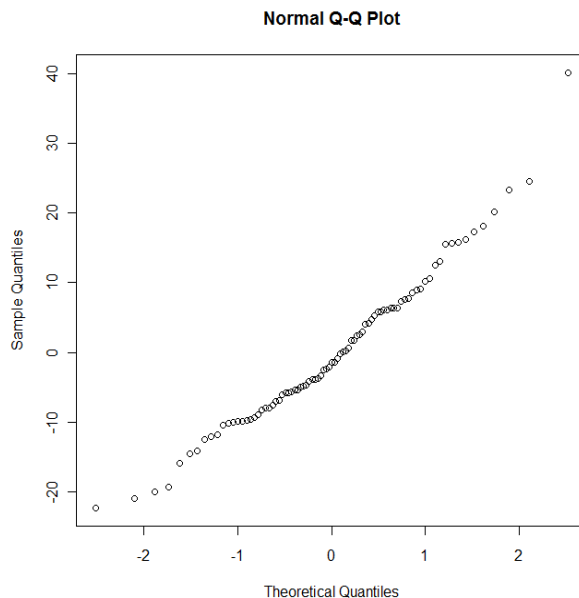
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
names(out1)
```

#names: you are bringing all the variable names from data file

```
[1] "coefficients" "residuals" "effects" "rank"
[5] "fitted.values" "assign" "qr" "df.residual"
[9] "xlevels" "call" "terms" "model"
```

```
qqnorm(out1$residuals)
```



****note:** Residuals from the linear model appear to be normally distributed (since data plots along a line)

2. Two-way ANOVA

Example #2:

Test if the average job satisfaction score is same for males and females.

(use data file twoway.csv)

- $H_0: \mu_m = \mu_f$
- $H_a: \mu_m \neq \mu_f$

Observation	Gender	Position	Score
1	m	S	39
2	m	S	51
3	m	S	54
4	m	S	51
5	m	M	33
6	m	M	36
7	m	M	84
8	m	M	57
9	f	S	60
10	f	S	51
11	f	S	81
12	f	S	57
13	f	M	60
14	f	M	51
15	f	M	69
16	f	M	81

```
twoway <- read.csv("K:/TEACH/DataMining_Fall2009/Data/twoway.csv",  
header=TRUE)
```

```
outanova2 <- aov(Score~Gender*Position)  
summary(outanova2)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Gender	1	689.06	689.06	2.9518	0.1115
Position	1	45.56	45.56	0.1952	0.6665
Gender:Position	1	0.56	0.56	0.0024	0.9617
Residuals	12	2801.25	233.44		

#*: running model with interaction

**note: Since the interaction term (Gender:Position) is not significant, it should be dropped and a two-way additive ANOVA model should be used.

```
outanova3 <- aov(Score~Gender+Position)  
summary(outanova3)
```

#+: dropping interaction: additive model

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Gender	1	689.06	689.06	3.1971	0.09709
Position	1	45.56	45.56	0.2114	0.65326
Residuals	13	2801.81	215.52		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

REGRESSION

R FUNCTIONS FOR REGRESSION ANALYSIS

: Here are some helpful R functions for regression analysis grouped by their goal. The name of package is in parentheses.

LM & GLM

<code>Anova (car)</code>	Anova Tables for Linear and Generalized Linear Models
<code>anova (stats)</code>	Compute an analysis of variance table for linear model fits
<code>coef (stats)</code>	generic function which extracts model coefficients
<code>coeftest (lmtest)</code>	Testing Estimated Coefficients
<code>confint (stats)</code>	Computes confidence intervals for parameters in a fitted model.
<code>deviance (stats)</code>	Returns the deviance of a fitted model object.
<code>effects (stats)</code>	Returns (orthogonal) effects from a fitted model, usually a linear model.
<code>fitted (stats)</code>	generic function which extracts fitted values
<code>formula (stats)</code>	provide a way of extracting formulae
<code>linear.hypothesis</code>	Test Linear Hypothesis
<code>lm (stats)</code>	Fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance
<code>glm</code>	Generalized Linear Model
<code>family</code>	Family objects provide a convenient way to specify the details of the models used by functions such as <code>glm</code>
<code>lm.ridge (MASS)</code>	Ridge Regression
<code>rlm (MASS)</code>	Robust Fitting of Linear Models
<code>model.matrix (stats)</code>	creates a design matrix
<code>predict (stats)</code>	Predicted values based on linear model object
<code>residuals (stats)</code>	generic function which extracts model residuals
<code>vcov (stats)</code>	Returns the variance-covariance matrix of a fitted model.

Model-Variable Selection

<code>add1 op1 / dr (stats)</code>	Compute all the single terms in the scope argument that can be added to or dropped from the model. Fit those models and compute a table of the changes in fit.
<code>AIC (stats)</code>	Generic function calculating the Akaike information criterion
<code>extractAIC (stats)</code>	Computes the (generalized) AIC for a fitted parametric model.
<code>coefficient (stats)</code>	coefficient
<code>step (stats)</code>	step: Select a formula-based model by AIC
<code>update.formula (stats)</code>	update model formulae. Typically involves adding or dropping terms.

Tests

<code>bartlett.test</code> (<code>stats</code>)	Bartlett's test of the null that variances in each of the groups are same.
<code>cvm.test</code> (<code>nortest</code>)	Cramer-von Mises test for normality
<code>durbin.watson</code> (<code>car</code>)	Durbin-Watson Test for Autocorrelated Errors
<code>dwtest</code> (<code>lmtest</code>)	Durbin-Watson Test
<code>levene.test</code> (<code>car</code>)	Levene's Test
<code>lillie.test</code> (<code>nortest</code>)	Lilliefors (Kolmogorov-Smirnov) test for normality
<code>ncv.test</code> (<code>car</code>)	Score Test for Non-Constant Error Variance
<code>pearson.test</code> (<code>nortest</code>)	Pearson chi-square test for normality

Diagnostics

<code>cookd</code> (<code>car</code>)	Cook's Distances for Linear and Generalized Linear Models
<code>cooks.distance</code> (<code>stats</code>)	Cook's distance
<code>covratio</code> (<code>stats</code>)	covariance ratio
<code>influence.measures</code> (<code>stats</code>)	This suite of functions can be used to compute some of the regression (leave-one-out deletion) diagnostics for linear and generalized linear models
<code>lm.influence</code> (<code>stats</code>)	This function provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of regression fits.
<code>ls.diag</code> (<code>stats</code>)	Computes basic statistics, including standard errors, t- and p-values for the regression coefficients (<code>stats</code>)
<code>outlier.test</code> (<code>car</code>)	Bonferroni Outlier Test
<code>rstandard</code> (<code>stats</code>)	standardized residuals
<code>rstudent</code> (<code>stats</code>)	studentized residuals
<code>vif</code> (<code>HH</code>)	Variance Inflation Factor

Graphics

<code>plot.lm</code>	Four plots provided: 1) residuals against fitted values, 2) Scale-Location plot of sqrt against fitted values, 3) Normal Q-Q plot, 4) plot of Cook's distances versus row
<code>prplot</code> (<code>faraway</code>)	Partial Residual Plot
<code>qq.plot</code> (<code>car</code>)	Quantile-Comparison Plots
<code>qqline</code>	adds a line to a normal quantile-quantile plot which passes through the first and third quartiles
<code>qqnorm</code>	produces a normal QQ plot
<code>reg.line</code> (<code>car</code>)	Plot Regression Line
<code>scatterplot.matrix</code> (<code>car</code>)	Scatterplot Matrices
<code>scatterplot</code> (<code>car</code>)	Scatterplots with Boxplots

1. Multiple Linear Regression

Example #1

: The data file football.csv has data on Position, Weight, Time (for 40 yard dash) and player's Rating. Fit a multiple linear regression model to $Y = \text{rating}$ as a function of the other 3 variables. (use data file football.csv)

subject	Position	Weight	Time	Rating	Dguard
1	Guard	322	5.38	7.4	1
2	Guard	303	5.18	7.0	1
3	Guard	317	5.34	6.8	1
4	Guard	330	5.46	6.7	1
5	Guard	334	5.18	6.3	1
6	Guard	308	5.32	6.1	1
7	Guard	310	5.28	6.0	1
8	Guard	318	5.37	6.0	1
9	Guard	321	5.25	6.0	1
10	Guard	295	5.34	5.8	1
11	Guard	328	5.31	5.3	1
12	Guard	320	5.64	5.0	1
13	Guard	304	5.20	5.0	1
14	Offensive tackle	325	4.95	8.5	0
15	Offensive tackle	361	5.50	8.0	0
16	Offensive tackle	315	5.39	7.8	0
17	Offensive tackle	307	4.98	7.6	0
18	Offensive tackle	326	5.20	7.3	0
19	Offensive tackle	320	5.36	7.1	0
20	Offensive tackle	287	5.05	6.8	0
21	Offensive tackle	332	5.26	6.8	0
22	Offensive tackle	334	5.55	6.4	0
23	Offensive tackle	312	5.15	6.3	0
24	Offensive tackle	299	5.35	6.1	0
25	Offensive tackle	333	5.59	6.0	0

```
football <- read.csv("M:/TEACH/DataMining_Fall2009/Data/football.csv")
attach(football)
names(football)
[1] "Position" "Weight"   "Time"     "Rating"   "Dguard"

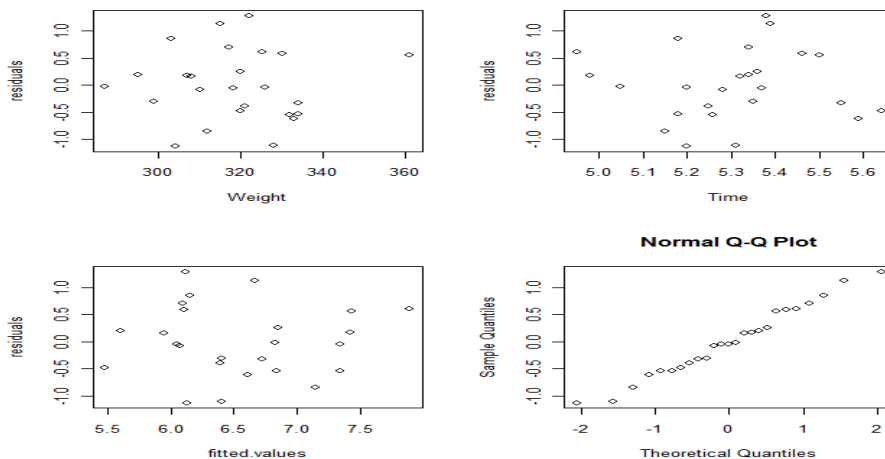
outreg1 <- lm(Rating~Weight+Time+Dguard)
# lm: R will perform linear model, DV: Rating, IVs: Weight, Time, Dguard

summary(outreg1)
Call:
```

```
lm(formula = Rating ~ Weight + Time + Dguard)
Residuals:
    Min       1Q   Median       3Q      Max
-1.12702 -0.48001 -0.04766  0.55881  1.28344
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.95563   4.49988   2.657  0.0148 *
Weight       0.02219   0.01039   2.135  0.0447 *
Time        -2.27755   0.92895  -2.452  0.0231 *
Dguard      -0.73237   0.28935  -2.531  0.0194 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.6936 on 21 degrees of freedom
Multiple R-squared: 0.4755,    Adjusted R-squared: 0.4005
F-statistic: 6.345 on 3 and 21 DF,  p-value: 0.003118 (Null rejected)
```

```
attach(outreg1)
nf <- layout(matrix(c(1, 2, 3, 4),2,2,byrow=TRUE))
layout.show(nf)
#matrix: setting up a matrix → 4 plots on 1 layout
```

```
plot(residuals~Weight)
plot(residuals~Time)
plot(residuals~fitted.values)
qqnorm(residuals)
```



Variance Inflation Factor(VIF) in R : You need to install and load R-package HH

```
library(HH)
vif(outreg1)
  Weight      Time      Dguard
1.296048 1.291599 1.086100
```

2. Regression Analysis using Dummy variables

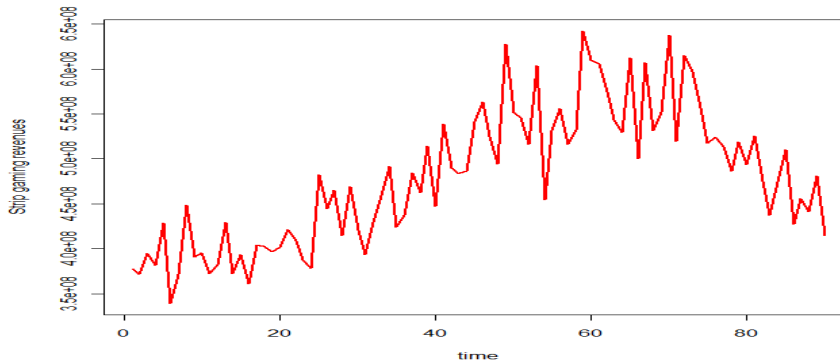
- Regression methods are also used for modeling time series data.

Example #2

The data file gaming_revenue.csv has monthly gaming revenue data for Clark County, Strip, Downtown and Boulder City, for 2002 – 2009. Fit a regression model using dummy variables to account for seasonality (months) and time t (# of month, t = 1, 2, ..., 90). Following table shows the Strip values (in million \$). (use data file gaming_revenue.csv)

642.34	616.36	656.11	647.07	681.19	561.97	627.4	665.12	643.2	654.48	592.87	642.44	635.88
703.26	620.35	684.98	608.89	651	656.39	655.9	635.8	683.85	664.5	631.44	635.03	652.61
747.66	731.18	776.29	678.72	748.98	704.33	647.23	726.32	753.27	766.72	720.4	709.89	725.91
793.1	765.8	864.99	723.52	860.38	797.15	765.63	808.36	837.1	884.08	846.69	762.6	809.12
987.7	869.98	908.67	824.2	962.62	757.74	850.26	886.02	807.85	889.91	989.65	908.63	886.93
967.78	901.82	889.67	892.76	968.44	789.66	964.73	838.03	879.14	1001.3	828.73	945.96	905.67
928.65	865.97	871.9	849.97	810.06	806.1	819.68	759.26	853.51	757.51	702.59	771.78	816.41
777.53	710.6	786.46	734.71	747.61	687.55							740.74
818.5	760.26	804.88	744.98	803.78	720.11	761.55	759.84	779.7	802.64	758.91	768.04	0

```
gr <- read.csv("K:/TEACH/DataMining_Fall2009/Data/gaming_revenue.csv")
attach(gr)
plot(Strip)
layout(1)
plot(Strip, type = "l", lwd=2, col="red", xlab="time", ylab="Strip
gaming revenues")
#type = "l" : graph type = line graph,
#lwd=2 : line width = 2
#col="red": color = red,
#xlab="time": x label = time
#ylab="Strip gaming revenues": y label = Strip gaming revenues
```



```
outreg2 <-
lm(Strip~Month_1+DJJan+DFeb+DFeb+DMar+DApr+DMay+DJun+DJJul+DAug+DSep+DOct
+DNov)
summary(outreg2)
Coefficients:
```

Estimate	Std. Error	t value	Pr(> t)
----------	------------	---------	----------

```

(Intercept) 391304739    18878186  20.728 < 2e-16 ***
T            1870717      232672    8.040  8.51e-12 ***
DJan         42593573      25400646  1.677  0.0976 .
DFeb        -3481394      25392763 -0.137  0.8913
DMar         3516764      25387010  0.139  0.8902
DApr        -22589217      25364451 -0.891  0.3759
DMay         29928816      25361834  1.180  0.2416
DJun        -39821526      25361351 -1.570  0.1205
DJul         3297002      26529926  0.124  0.9014
DAug         7823714      26521147  0.295  0.7688
DSep         14286569      26514406  0.539  0.5916
DOct         26474995      26509706  0.999  0.3211
DNov        -22075879      21655506 -1.019  0.3112

```

```

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 57160000 on 77 degrees of freedom
Multiple R-squared:  0.5006,    Adjusted R-squared:  0.4228
F-statistic: 6.433 on 12 and 77 DF,  p-value: 8.984e-08

```

```

t <- Month_1
t1 <- (t - mean(t))/sd(t)
t2 <- t1^2

```

****note: transforming data → you can transform data and increase Rsquare or lower VIF.**

```

outreg2t <-
lm(Strip~t1+t2++DJan+DFeb+DFeb+DMar+DApr+DMay+DJun+DJul+DAug+DSep+DOct+
DNov)
summary(outreg2t)

```

```

Coefficients:
              Estimate      Std. Error  t value Pr(>|t|)
(Intercept)  510066622  12503322  40.794 < 2e-16 ***
t1           48868019   4623458  10.57 < 2e-16 ***
t2          -39600607   5241056  -7.556 7.76e-11 ***
DJan         53176852   19371107  2.745  0.00754 **
DFeb         6869943    19362930  0.355  0.72372
DMar         13752205   19357486  0.71  0.47961
DApr        -12395257   19339985 -0.641  0.52351
DMay         40238971   19339078  2.081  0.04083 *
DJun        -29279130   19340905 -1.514  0.13421
DJul         3388589    20179365  0.168  0.86709
DAug         7683358    20172692  0.381  0.70436
DSep         14030316   20167585  0.696  0.48875
DOct         26218892   20164009  1.3  0.19743
DNov        -21742843   16471808 -1.32  0.1908

```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 43480000 on 76 degrees of freedom
Multiple R-squared: 0.7148, Adjusted R-squared: 0.6661
F-statistic: 14.66 on 13 and 76 DF, p-value: 8.716e-16

```
layout(1:2)
```

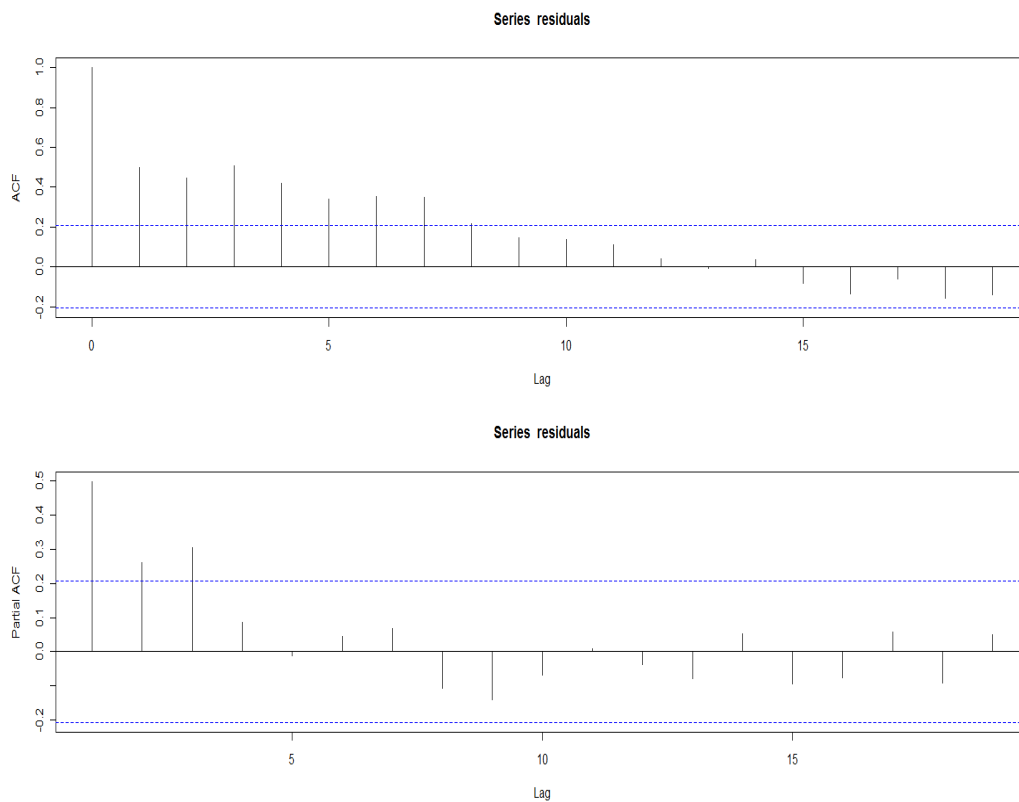
```
acf(residuals)
```

```
pacf(residuals)
```

#layout: setting graph or plot format → you want to see 2 plots on 1 layout here.

#acf: auto correlation function

#pacf: partial auto correlation function



The points where there are high spikes is where it tells you significance.

****note!!! When Lag 0 shows to be significant, it is its point by itself. So it is not significant.*

3. Robust Regression

- Robust: results will not change too much although you add more stuff.

- Robust: performs well even if its assumptions are somewhat violated by the true model from which the data were generated.
- Robust Regression is used when:
 - Normality of residuals not met
 - Non-Constant error variance
- To run ROBUST REGRESSION in R, install and load the R-package MASS.
- Huber's Method – (hybrid of OLS and LAD): R uses this.
 - OLS: Ordinary Least Squares-what we usually use in linear model.
 - Least Absolute Deviation (LAD)

1) Normality of Residuals not met

- 2 ways to test normality of residuals: install and load R-package **nortest**
 - `lillie.test(residuals)`
 - `cvm.test(residuals)`

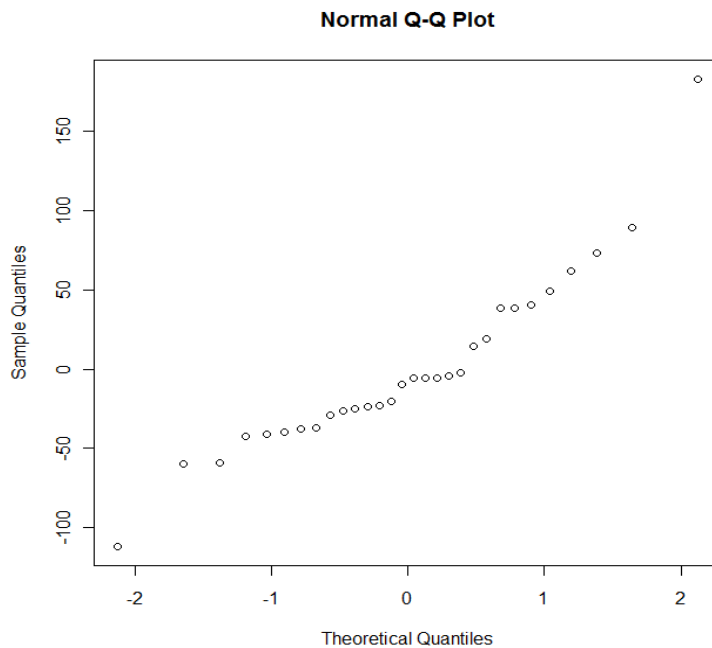
Example #3

```

Library(MASS)
gala <-
read.csv("M:/TEACH/DataMining_Fall2009/Data/galapagos.csv",header=TRUE)
attach(gala)
ols_reg3 <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
gala)
summary(ols_reg3)
Call:
lm(formula = Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
    data = gala)
Residuals:
    Min       1Q   Median       3Q      Max
-111.679  -34.898   -7.862   33.460  182.584
Coefficients:
                Estimate  Std. Error  t value Pr(>|t|)
(Intercept)  7.068221  19.154198   0.369  0.715351
Area         -0.023938   0.022422  -1.068  0.296318
Elevation     0.319465   0.053663   5.953  3.82e-06 ***
Nearest       0.009144   1.054136   0.009  0.993151
Scruz        -0.240524   0.215402  -1.117  0.275208
Adjacent     -0.074805   0.017700  -4.226  0.000297 ***
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 60.98 on 24 degrees of freedom
Multiple R-squared: 0.7658,    Adjusted R-squared: 0.7171
F-statistic: 15.7 on 5 and 24 DF,  p-value: 6.838e-07

attach(ols_reg3)
qqnorm(residuals)

```



```
library(nortest)
```

```
lillie.test(residuals)
```

***note: if $p < .05$ you are rejecting the H_0 which means not normal!*

Lilliefors (Kolmogorov-Smirnov) normality test

```
data: residuals
```

```
D = 0.1813, p-value = 0.01310 (Reject Null)
```

```
cvm.test(residuals)
```

***note: if $p < .05$ you are rejecting the H_0 which means not normal!*

Cramer-von Mises normality test

```
data: residuals
```

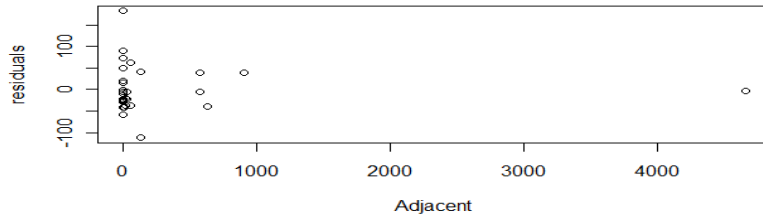
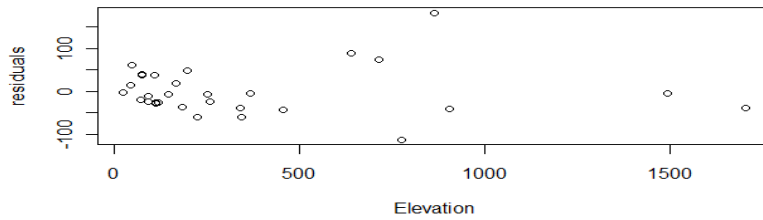
```
W = 0.1509, p-value = 0.02149 (Reject Null)
```

***note: Clearly, residuals are not normally distributed.*

```
layout(1:2)
```

```
plot(residuals~Elevation)
```

```
plot(residuals~Adjacent)
```



****note: So you run Robust regression instead.**

```
rob_reg3 <- rlm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent)
summary(rob_reg3)
```

rlm: robust linear model

```
Call: rlm(formula = Species ~ Area + Elevation + Nearest + Scruz +
  Adjacent)
```

Residuals:

Min	1Q	Median	3Q	Max
-74.389	-18.353	-6.364	21.187	229.082

Coefficients:

	Value	Std. Error	t value
(Intercept)	6.3611	12.3897	0.5134
Area	-0.0061	0.0145	-0.4214
Elevation	0.2476	0.0347	7.1320*
Nearest	0.3592	0.6819	0.5267
Scruz	-0.1952	0.1393	-1.4013
Adjacent	-0.0546	0.0114	-4.7648*

Residual standard error: 29.73 on 24 degrees of freedom (* significant)

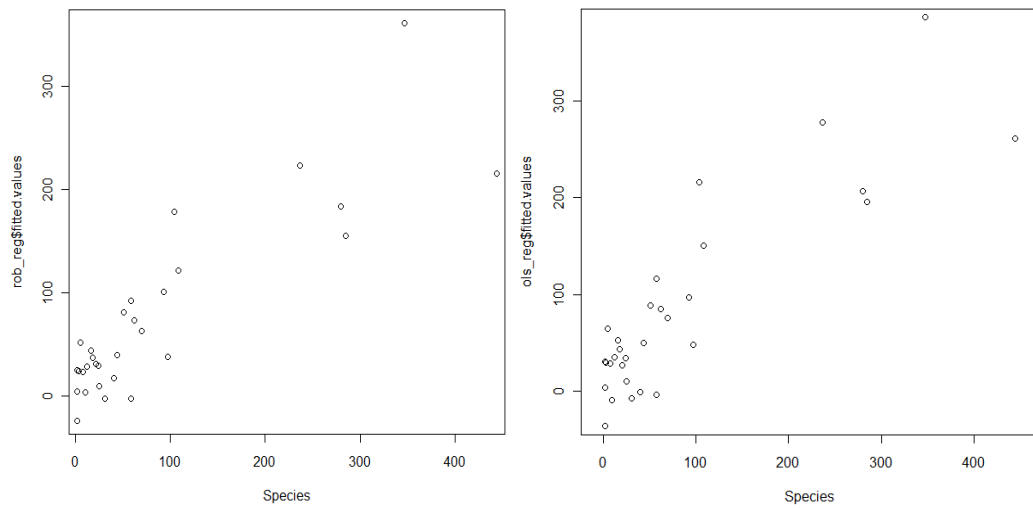
```
eqscplot(Species, ols_reg3$fitted.values)
```

```
eqscplot(Species, rob_reg3$fitted.values)
```

eqscplot: plots scatterplot on equal scales (R-package MASS)

**note: scatter plot for linear model

**note: scatterplot for robust model



***note: The first graph tells you that the linear Model takes account the outliers more. The second graph tells you that the Robust Model is ignoring the outliers more.*

2) Non-Constant Error Variance

Example #4

The data file Bid_Cost.csv shows the values of X = size of a bid in million \$, and Y = cost to the firm preparing the bid, for 12 recent bids.

(use data file Bid_Cost.csv)

BidSize	BidCost
2.13	15.5
1.21	11.1
11	62.6
6	35.4
5.6	24.9
6.91	28.1
2.97	15
3.35	23.2
10.39	42
1.1	10
4.36	20
8	47.5

```
bid <-
read.csv("K:/TEACH/DataMining_Fall2009/Data/Bid_Cost.csv",header=TRUE)
attach(bid)
names(bid)
[1] "BidSize" "BidCost"
```

```

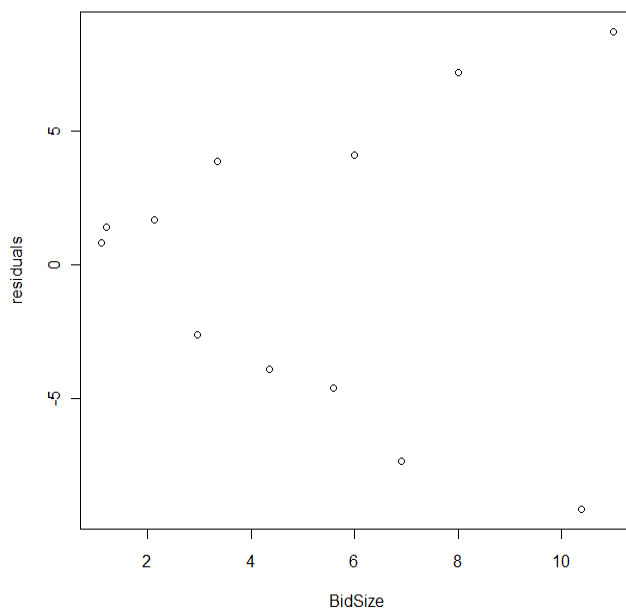
outreg4 <- lm(BidCost ~ BidSize)
summary(outreg4)
Call:
lm(formula = BidCost ~ BidSize)
Residuals:
    Min       1Q   Median       3Q      Max
-9.143 -4.090  1.106  3.904  8.703
Coefficients:
            Estimate          Std. Error t value Pr(>|t|)
(Intercept)  4.2289          3.2517     1.301   0.223
BidSize      4.5153          0.5285     8.544 6.59e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 5.87 on 10 degrees of freedom
Multiple R-squared:  0.8795,    Adjusted R-squared:  0.8675
F-statistic:   73 on 1 and 10 DF,  p-value: 6.588e-06

```

```

attach(outreg4)
plot(residuals~BidSize)

```



```
x2 <- Bidsize^2
```

***note: You have to create the new variable x2 first, you are minimizing error sum of square*

```

reg4_wtd <- lm(BidCost ~ BidSize, weights = 1/x2)
summary(reg4_wtd)
lm(formula = BidCost ~ BidSize, weights = 1/x2)
Residuals:
    Min       1Q   Median       3Q      Max
-1.04471 -0.79092  0.03284  0.82150  1.04621

```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    5.6569      0.9652   5.861 0.000159 ***
BidSize         4.1906      0.4037  10.381 1.13e-06 ***
```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8897 on 10 degrees of freedom

Multiple R-squared: 0.9151, Adjusted R-squared: 0.9066

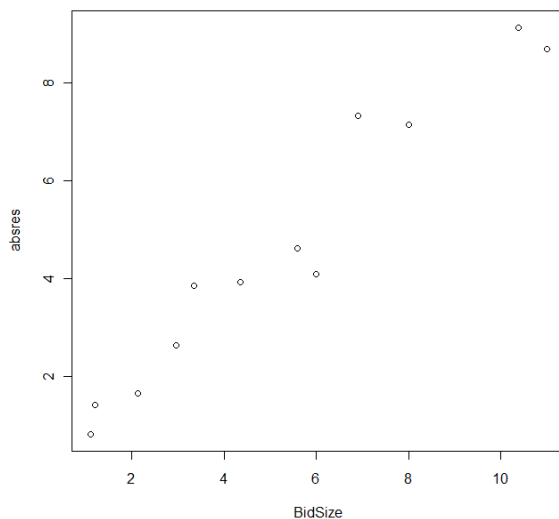
F-statistic: 107.8 on 1 and 10 DF, p-value: 1.127e-06

***note: Regression weighted: if error increase as $1/x^2 \rightarrow$ eventually you are minimizing error. You are trying to increase/decrease error by contributing different weight values.*

3) Weighted Regression

Example #5

```
bid <-
read.csv("K:/TEACH/DataMining_Fall2009/Data/Bid_Cost.csv",header=TRUE)
outreg5 <- lm(BidCost ~ BidSize)
absres <- abs(outreg5$residuals)
plot(absres~BidSize)
```



***note: When your plot is linear, your weight is $w = 1/x$*

#abs(residual) is a LINEAR function of BidSize (and not quadratic since BidSize² term is not significant).

```
BidSize2 <- BidSize^2
res_reg5a <- lm(absres~BidSize+BidSize2)
summary(res_reg5a)
Call:
lm(formula = absres ~ BidSize + BidSize2)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-1.29732	-0.34033	-0.08335	0.33580	1.21061

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.09063	0.65975	-0.137	0.8938
BidSize	0.99428	0.26562	3.743	0.0046 **
BidSize2	-0.01384	0.02179	-0.635	0.5410

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7118 on 9 degrees of freedom

Multiple R-squared: 0.9492, Adjusted R-squared: 0.938

F-statistic: 84.16 on 2 and 9 DF, p-value: 1.495e-06

```
res_reg 5b<- lm(BidCost~BidSize, weights = 1/BidSize)
summary(res_reg 5b)
```

****OUTPUT FROM OLS**

```
lm(formula = BidCost ~ BidSize)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-9.143	-4.090	1.106	3.904	8.703

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	4.2289	3.2517	1.301	0.223
BidSize	4.5153	0.5285	8.544	6.59e-06 ***

****OUTPUT FROM WEIGHTED LS**

```
lm(formula = BidCost ~ BidSize, weights = 1/BidSize)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	5.2573	1.7352	3.03	0.0127 *
BidSize	4.3195	0.4301	10.04	1.53e-06 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.9098, Adjusted R-squared: 0.9008

4. Polynomial Regression

- Polynomial Regression is used when your data is not linear.

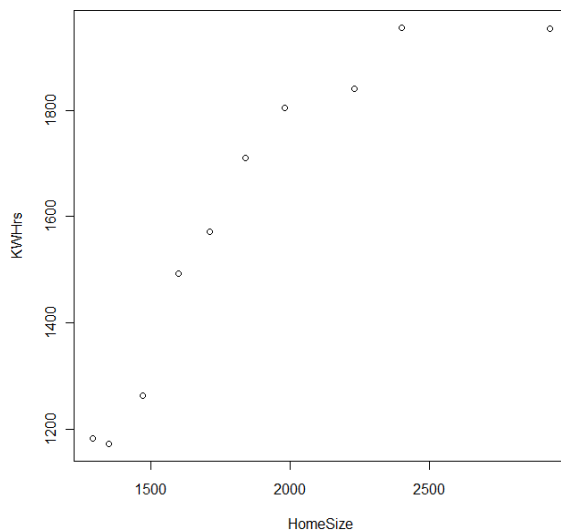
Example #6

Following table shows energy used (KWHrs) as a function of HomeSize (square feet). Fit a regression model to KWHrs as a function of HomeSize.

(use data file HomeEnergyUse.csv)

HomeSize	KWHrs
1290	1182
1350	1172
1470	1264
1600	1493
1710	1571
1840	1711
1980	1804
2230	1840
2400	1956
2930	1954

```
home <- read.csv("K:/TEACH/DataMining_Fall2009/Data/HomeEnergyUse.csv")
attach(home)
plot(home)
```



****note:** Look at the plot and discover it is not linear → Therefore, this is Polynomial Regression.
Since the graph shows 2 bends, we should try a cubic function:

```

KWHrs =  $\beta_0 + \beta_1 \text{HomeSize} + \beta_2 \text{HomeSize}^2 + \beta_3 \text{HomeSize}^3$ 
x <- (HomeSize-mean(HomeSize))/sd(HomeSize)
x2 <- x^2
x3 <- x^3
library(HH)
outreg6 <- lm(KWHrs~HomeSize+HS2+HS3)
summary(outreg6)
Call:
lm(formula = KWHrs ~ HomeSize + HS2 + HS3)
Residuals:
    Min       1Q   Median       3Q      Max
-75.068 -22.133   6.907  30.633  55.594
Coefficients:
              Estimate      Std. Error    t value Pr(>|t|)
(Intercept) -1.414e+03   1.300e+03   -1.088    0.318
HomeSize     2.707e+00   2.000e+00    1.354    0.225
HS2          -6.033e-04   9.876e-04   -0.611    0.564
HS3          2.436e-08   1.567e-07    0.156    0.882
Residual standard error: 50.45 on 6 degrees of freedom
Multiple R-squared: 0.982, Adjusted R-squared: 0.9729
F-statistic: 108.9 on 3 and 6 DF, p-value: 1.276e-05

vif(outreg6)
HomeSize      HS2      HS3
 3788.808    16000.454  4370.732
**note: VIF values TOO HIGH!!!: you need library(HH) for vif

```

1) Handling Multicollinearity:

- a. use **ORTHOGONAL POLYNOMIALS**.
- b. easier way: standardize the X-variable : $cX1 = (X - \text{mean}(X))/\text{sd}(X)$
and then calculate the square and cubic terms.

b. standardize the X-variable to reduce VIFs

```

x <- (HomeSize-mean(HomeSize))/sd(HomeSize)
x2 <- x^2
x3 <- x^3
**note: you are standardizing x-variables to reduce VIFs

outreg6a <- lm(KWHrs~x+x2+x3)
summary(outreg6a)
Call:
lm(formula = KWHrs ~ x + x2 + x3)
Residuals:
    Min       1Q   Median       3Q      Max
-75.068 -22.133   6.907  30.633  55.594
Coefficients:
              Estimate      Std. Error    t value Pr(>|t|)

```

```

(Intercept) 1704.872    24.553  69.436 6.00e-10 ***
x            360.832    38.349   9.409 8.19e-05 ***
x2          -124.828    32.205  -3.876  0.0082 **
x3             3.379    21.727   0.156  0.8815
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 50.45 on 6 degrees of freedom
Multiple R-squared: 0.982,    Adjusted R-squared: 0.9729
F-statistic: 108.9 on 3 and 6 DF,  p-value: 1.276e-05

```

```
vif(outreg6a)
```

```

x      x2      x3
5.200281  5.435543  13.034283

```

****note:** VIF values decreased significantly compared to previous model.

****note:** Perform linear model once again with only significant IVs.

```

outreg6b <- lm(KWHrs~x+x2)
summary(outreg6b)

```

```
Call:
```

```
lm(formula = KWHrs ~ x + x2)
```

```
Residuals:
```

```

      Min       1Q   Median       3Q      Max
-73.792 -22.426   5.886  31.689  52.436

```

```
Coefficients:
```

```

              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1703.22     20.54  82.914 9.77e-12 ***
x             365.84     19.27  18.985 2.80e-07 ***
x2           -120.58     15.83  -7.618 0.000124 ***
---

```

```

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 46.8 on 7 degrees of freedom
Multiple R-squared: 0.9819,    Adjusted R-squared: 0.9767
F-statistic: 189.7 on 2 and 7 DF,  p-value: 8e-07

```

```
vif(outreg6b)
```

```

x      x2
1.525748  1.525748

```

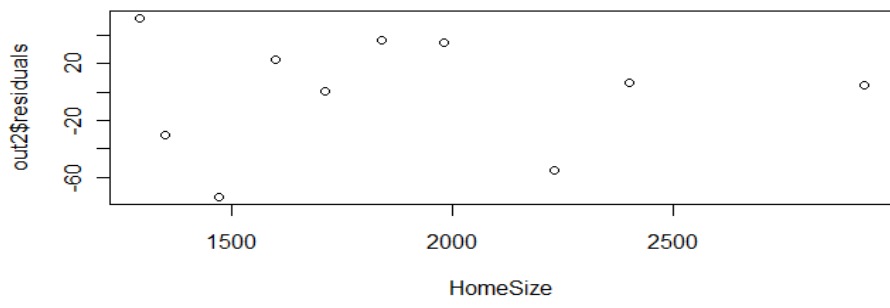
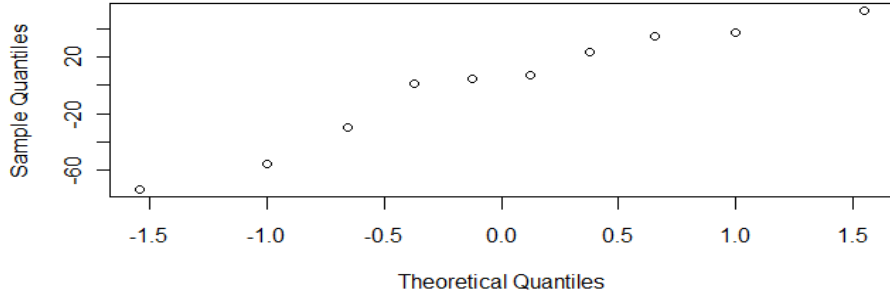
****note:** RESIDUAL DIAGNOSTIC PLOTS

```
layout(1:2)
```

```
qqnorm(outreg6b$residuals)
```

```
plot(outreg6b$residuals~HomeSize)
```

Normal Q-Q Plot



5. Logistic Regression

- Logistic Regression is good to use in predicting the DV when you have many IVs.
- Consider a simple example:
- $Y = 1$ (restaurant is a success, and 0 if restaurant fails).

Experience	N	#(Successes)	#(Failures)	Mean(Y)
2	10	2	8	0.2
4	40	10	30	0.25
6	50	20	30	0.4
8	50	30	20	0.6
10	50	35	15	0.7
15	50	40	10	0.8
20	50	45	5	0.9
22	100	95	5	0.95
25	100	98	2	0.98

- $\text{mean}(Y) = \#(\text{successes})/N = \text{estimated probability of success } p$
- **note: You see a pattern that as experience yrs increase \rightarrow # successes increase as well.*

- What is the probability of the rest to succeed as # experience as a predictor?
- How are $\beta_0, \beta_1, \dots, \beta_k$ estimated in logistic regression?

Y	X1	X2	X3	X4	X5
0	1.5	0	4.1
1	2.5	0	3.8
0	4.1	1	4.1
0	7.5	0	8.7
1	2.3	1	1.4
1	1.9	1	3.2
0	8.1	0	11.3
1	1.1	0	0.4
...
...
0	10.2	1	13.7
0	11.6	0	7.9
1	7.3	1	5.4

- The method of Maximum Likelihood is used.
- The method of maximum likelihood finds the values of $\beta_0, \beta_1, \dots, \beta_k$ which maximize the log-likelihood (computer search method is used to find the maximum).
- Output from Logistic Regression:
Deviance = $-2 * \text{log-likelihood}$

Akaike's Information Criterion (AIC) = - 2 * log-likelihood + 2 M

M = K + 1 = #(parameters estimated)

Bayesian Information Criterion (BIC) = - 2 * log-likelihood + M log(N)

***note: A model with small AIC or BIC is preferred.*

- o Install and upload R-package **MASS** to get AIC

1) Logistic Regression

Example #7

#Install MASS to perform variable selection using AIC criterion

```
library(MASS)
```

```
book <-
```

```
read.csv("M:/TEACH/DataMining_Fall2009/Data/Charles_BookClub.csv",  
header=TRUE)
```

```
outreg7a <- glm(Florence ~
```

```
Gender+M+R+F+FirstPurch+ChildBks+YouthBks+CookBks+DoltYBks+RefBks+ArtBk  
s+GeogBks+ItalCook+ItalHAtlas+ItalArt, family=binomial("logit"))
```

```
summary(outreg7a)
```

family=binomial("logit"): logistic regression

```
glm(formula = Florence ~ Gender + M + R + F + FirstPurch + ChildBks +  
YouthBks + CookBks + DoltYBks + RefBks + ArtBks + GeogBks +  
ItalCook + ItalHAtlas + ItalArt, family = binomial("logit"))
```

Deviance Residuals:

	Min	1Q	Median	3Q	Max	
	-1.8759	-0.4774	-0.3311	-0.1971	2.9345	
Estimate	Std. Error	z value	Pr(> z)			
(Intercept)	-1.3524835	0.3096589	-4.368	1.26e-05	***	
Gender	-0.8833497	0.1613001	-5.476	4.34e-08	***	
M		0.0003501	0.0009195	0.381	0.703404	
R		-0.0892098	0.0187102	-4.768	1.86e-06	***
F		0.2711517	0.0921058	2.944	0.003241	**
FirstPurch	0.0138329	0.0116336	1.189	0.234419		
ChildBks	-0.5059118	0.1367719	-3.699	0.000216	***	
YouthBks	-0.6457685	0.1889838	-3.417	0.000633	***	
CookBks	-0.6442635	0.1468503	-4.387	1.15e-05	***	
DoltYBks	-0.7815103	0.1742908	-4.484	7.33e-06	***	
RefBks	0.0171449	0.1790575	0.096	0.923719		
ArtBks	0.6449422	0.1545865	4.172	3.02e-05	***	
GeogBks	0.1943929	0.1428007	1.361	0.173423		
ItalCook	0.3067418	0.2598205	1.181	0.237765		
ItalHAtlas	0.0909080	0.3998221	0.227	0.820135		
ItalArt	0.3923821	0.3372949	1.163	0.244699		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1373.5 on 1999 degrees of freedom

Residual deviance: 1120.9 on 1984 degrees of freedom
 (2 observations deleted due to missingness)
AIC: 1152.9 = Residual Deviance + 2 M = 1120.9+ 32
 Number of Fisher Scoring iterations: 6

****note: Drop the non-significant variables and perform logistic regression again.**

```
outreg7b <- glm(Florence ~
Gender+R+F+ChildBks+YouthBks+CookBks+DoltYBks+ArtBks,
family=binomial("logit"))
summary(outreg7b)
Call:
glm(formula = Florence ~ Gender + R + F + ChildBks + YouthBks +
CookBks + DoltYBks + ArtBks, family = binomial("logit"))
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7523  -0.4749  -0.3344  -0.2028   2.9177
```

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.52130    0.22876  -6.650 2.93e-11 ***
Gender       -0.88362    0.16032  -5.512 3.56e-08 ***
R            -0.06604    0.01414  -4.671 2.99e-06 ***
F            0.40397     0.05851   6.904 5.04e-12 ***
ChildBks    -0.54516    0.12149  -4.487 7.21e-06 ***
YouthBks   -0.68522    0.17583  -3.897 9.74e-05 ***
CookBks    -0.62355    0.12153  -5.131 2.88e-07 ***
DoltYBks   -0.81585    0.15806  -5.162 2.45e-07 ***
ArtBks      0.63083     0.13085   4.821 1.43e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(Dispersion parameter for binomial family taken to be 1)
 Null deviance: 1373.5 on 1999 degrees of freedom
 Residual deviance: 1131.0 on 1991 degrees of freedom
 (2 observations deleted due to missingness)
AIC: 1149.0

Number of Fisher Scoring iterations: 6

****note: AIC value decreased after eliminating the IVs that were not significant from previous model. So this is a better model.**

- Logistic Regression Performance
 Confusion Matrix: contains information about actual and predicted values.

		PREDICTED	
		0	1
OBSERVED	0	a	b
	1	c	d

- a = # of correct predictions of response 0
- d = # of correct predictions of response 1
- b = # of incorrect predictions of response 0
- c = # of incorrect predictions of response 1
- Missclassification Rate = $(b+c)/(a+b+c+d)$
- Accuracy of prediction = AC = $(a+d)/(a+b+c+d)$
- Recall or true positive rate TP = $d/(c+d)$ = correctly classified 1's
- False positive rate FP = $b/(a+b)$
- True negative rate TN = $a/(a+b)$
- False negative rate FN = $c/(c+d)$
- Precision P = $d/(b+d)$ = proportion of predicted 1's that were correct

2) Logistic Regression and computing Confusion Matrix

Example #8

(use data file Charles_BookClub.csv)

```
charlesbook <-
read.csv("M:/TEACH/DataMining_Fall2009/Data/Charles_BookClub.csv",
header=TRUE)
outreg8 <- glm(Florence ~
Gender+M+R+F+FirstPurch+ChildBks+YouthBks+CookBks+DoltYBks+RefBks+ArtBk
s+GeogBks+ItalCook+ItalHAtlas+ItalArt, family=binomial("logit"))
names(outreg8)
[1] "coefficients"      "residuals"        "fitted.values"
"effects"           "R"                "rank"
[7] "qr"                "family"           "linear.predictors"
"deviance"         "aic"              "null.deviance"
[13] "iter"              "weights"          "prior.weights"
"df.residual"      "df.null"          "y"
[19] "converged"         "boundary"         "model"
"na.action"        "call"             "formula"
[25] "terms"             "data"             "offset"
"control"          "method"           "contrasts"
[31] "xlevels"
```

****note:** delete the 2 NA(missing) values in Y=Florence column (cases 2001 and 2002)

```
observed <- Florence[1:2000]
fitted <- round(outreg8$fitted.values)
xtabs(~observed+fitted)
```

****note:** you are only saving the cases from 1 to 2000 [1:2000].

#round: rounding the number

#xtabs: crosstabs → this will compute the confusion matrix.

observed	fitted	
	0	1
0	1764	19
1	189	28

****notes:**

FP = $19/(1764+19) = .01$: miss classified for 0

FN = $189/(189+28) = .87$: miss classified for 1

Overall Misclassification Rate = $(189+19)/(1764+189+19+28)$
 $= 208/2000 = .10 = 1 - AC$

Overall Correct classification Rate = $1 - .10$

3) Logistic Regression with Split Data set

Example #9 (continued from Example #8)

- Split data set into training set and test set to look the likelihood of prediction.
- Find # of cases in the data file Charles_BokokClub.csv

`length(Florence)`

length: tells you how many data points you have in file

```
[1] 2002
```

you have 2002 data points

`set.seed(10245)`

`holdout <- sample(1:nrow(charlesbook), 500, replace = FALSE)`

#set.seed: random selection, 10245 is any random number.

#500: 500 numbers from 1 to 2002.

Replace=FALSE: You are just subsetting the data. Do not bring data.

***note: Split data file so that training set has 75% of the data and test set has 25% of the data.*

```
charlesbook.train <- charlesbook [-holdout, ]
```

```
charlesbook.test <- charlesbook [holdout, ]
```

***note: Now split into training and test sets*

, :commanding to bring all column

[x, y] x: row, y: column → bring all x row all y column

[, 5]: bring all 5th column data

*** note: data points in training set # > test set #*

```

outreg9a <- glm(Florence ~
Gender+M+R+F+FirstPurch+ChildBks+YouthBks+CookBks+DoltYBks+RefBks+ArtBk
s+GeogBks+ItalCook+ItalHAtlas+ItalArt,data = book.train,
family=binomial("logit"))
summary(outreg9a)
Call:  glm(formula = Florence ~ Gender + R + F + FirstPurch + ChildBks
+      YouthBks + CookBks + DoltYBks + ArtBks + ItalCook, family =
binomial("logit"),      data = book.train)
Coefficients:
(Intercept)      Gender          R          F    FirstPurch
ChildBks      YouthBks
-1.04478      -1.06197      -0.11639      0.19435      0.02972      -
0.45224      -0.67728
      CookBks      DoltYBks      ArtBks      ItalCook
-0.64874      -0.77926      0.71032      0.62623
Degrees of Freedom: 1499 Total (i.e. Null); 1489 Residual
(2 observations deleted due to missingness)
Null Deviance:      984
Residual Deviance: 799.2      AIC: 821.2

```

****note: this number can change each time since you are selecting a different seed each time**

****note: Final model: use the final model built on training data to predict test data**

```

outreg9b <- glm(formula = Florence ~ Gender + R + F + FirstPurch +
ChildBks +      YouthBks + CookBks + DoltYBks + ArtBks + ItalCook,
family = binomial("logit"),      data = book.train)
outreg9c <- predict(outreg9b, book.test, type = 'response')
observed.train <- book.train$Florence[1:1500]
fitted.train <- outreg9b $fitted.values
xtabs(~observed.train+fitted.train)

```

****note: calculate correct confusion matrix and classification percentages on training set. this will only work on binary data**

	fitted.train	
observed.train	0	1
0	1340	8
1	136	16

```

correct_classification.train <- (1340+16)/(1340+136+8+16)
correct_classification.train

```

[1] 0.904

****note: similar to the concept of Rsquare value. 90.4% correctly classified.**

```

observed.test <- book.test$Florence
fitted.test <- round(outreg9c)
xtabs(~observed.test + fitted.test)

```

****note: calculate correct confusion matrix and classification percentages on training set**

	fitted.train	
observed.train	0	1
0	433	2
1	56	9

```
correct_classification.test <- (433+9)/(433+2+56+9)
correct_classification.test
```

```
[1] 0.884
```

***note: this value will never be the same but you are looking if they have a similar value. training value will always be higher.*

Example #10: Wetland Classification

- Data set has 26 variables, with DV = V26 = binary (wetland classification)
- V1, V2 = spatial coordinates of sample location
- Data set had over 6 million lines – 2 random samples, each of size 10000, was taken from this large data set.

```
wetland <- read.csv("M:/TEACH/DataMining_Fall2009/Data
/hcla.csv",header=TRUE)
names(wetland)
attach(wetland)
wetland.out <- glm(V26 ~
V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+V16+V17+V18+V19+V20+V21+V2
2+V23+V24, family=binomial("logit"))
summary(wetland.out)
Coefficients:
```

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-6.4971485	0.3698332	-17.568	< 2e-16	***
V3	0.9040565	0.3508822	2.577	0.009980	**
V4	-0.0065268	0.0018371	-3.553	0.000381	***
V5	0.7048252	0.0754529	9.341	< 2e-16	***
V6	0.0040051	0.0006381	6.276	3.47e-10	***
V7	-0.0358227	0.1856778	-0.193	0.847014	
V8	0.1506237	0.0217626	6.921	4.48e-12	***
V9	0.5306576	0.3440593	1.542	0.122990	
V10	0.2438647	0.1830803	1.332	0.182857	
V11	0.0429113	0.0438971	0.978	0.328300	
V12	0.0967733	0.5052403	0.192	0.848103	
V13	0.3521818	0.3441767	1.023	0.306186	
V14	-0.0804089	0.0421495	-1.908	0.056429	.
V15	0.0029327	0.0014124	2.076	0.037860	*
V16	0.0015271	0.0002880	5.302	1.15e-07	***
V17	-0.0045502	0.0009866	-4.612	3.99e-06	***
V18	-0.0046850	0.0014881	-3.148	0.001642	**

```

V19      -0.0582260  0.0273372  -2.130  0.033178  *
V20       0.2076367  0.0488683   4.249  2.15e-05  ***
V21      -0.1248815  0.0358366  -3.485  0.000493  ***
V22      -0.0021921  0.0052871  -0.415  0.678430
V23      -0.0124892  0.0127656  -0.978  0.327899
V24       0.0286177  0.0219352   1.305  0.192013
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
Null deviance: 1624.2 on 9999 degrees of freedom
Residual deviance: 1202.0 on 9977 degrees of freedom
AIC: 1248.0

```

```

observed <- V26
fitted <- round(wet.out$fitted.values)

```

	fitted	
observed	0	1
0	9829	13
1	143	15

```

FP <- 13/(13+9829): miss classified for 0
FP
0.001320870

```

```

FN <- 143/(143+15): miss classified for 1
FN
0.9050633

```

```

Overall misclassification = (143+13)/10000
= 0.0156

```

***note: this data set has very small % of 1's, all of the DVs were 0's.*

```

#(1) = 158, n = 10000

```

```

P(Y=1) = .0158 = 1.6%

```

***note: As a rule of thumb, you want to see at least 10% of 1's, so not a very good data set.*

6. Multinomial Logistic Regression

- Logistic regression: DV – binary variable
- Multinomial Logistic regression: DV – qualitative, can take K values.
 - e.g. DV = brand preference, IV = gender, age, education
- Diagnostics are complicated.
- Requires large sample sizes.
- Small # of observations in a cell may cause model to not run, or give very poor results.
- One can do a crosstab between Y and categorical predictors to find out if such is the case.
- Install and load R-package **VGAM**

Example #11

```
brand<-
read.csv("K:/TEACH/DataMining_Fall2009/Data/brand_preference.csv",
header=TRUE)
names(brand)
[1] "female" "age"      "educ"    "hotel"
library(VGAM)
outreg11 <- vglm(hotel~female+age+educ, family=multinomial(),
na.action=na.pass)
summary(outreg11)
```

Call:

```
vglm(formula = hotel ~ female + age + educ, family = multinomial(),
      na.action = na.pass)
```

Pearson Residuals:

			Min	1Q	Median
3Q	Max				
log(mu[,1]/mu[,3])	-5.6858	-0.44984	-0.32336	0.68053	7.7123
log(mu[,2]/mu[,3])	-4.8460	-0.67938	-0.44118	0.95906	1.8578

Coefficients:

		Value	Std. Error	t value
(Intercept):1	24.2960242	2.172153	11.18523	
(Intercept):2	11.7712627	1.556724	7.56156	
female:1	-0.4648455	0.214926	-2.16281	
female:2	0.0855761	0.189146	0.45243	
age:1	-0.6923699	0.062235	-11.12517	
age:2	-0.3244444	0.043282	-7.49610	
educ:1	0.0097041	0.033808	0.28704*	
educ:2	0.0103743	0.029177	0.35557*	

Number of linear predictors: 2

Names of linear predictors: log(mu[,1]/mu[,3]), log(mu[,2]/mu[,3])

Dispersion Parameter for multinomial family: 1

Residual Deviance: 1544.739 on 1600 degrees of freedom

Log-likelihood: -772.3694 on 1600 degrees of freedom

Number of Iterations: 5

****note: educ1 and educ2 is not significant (indicated by small magnitudes of t-values) – run model without educ1 & educ2.**

```
outreg11b <- vglm(hotel~female+age, family=multinomial(),
na.action=na.pass)
summary(outreg11b)
```

Call:

```
vglm(formula = hotel ~ female + age, family = multinomial(), na.action
= na.pass)
```

Pearson Residuals:

		Min	1Q	Median
3Q	Max			
log(mu[,1]/mu[,3])	-5.7433	-0.44984	-0.32533	0.67856
log(mu[,2]/mu[,3])	-4.9030	-0.68165	-0.44449	0.94920

Coefficients:

	Value	Std. Error	t value
(Intercept):1	24.354096	2.164362	11.25232
(Intercept):2	11.832644	1.547953	7.64406
female:1	-0.464434	0.214904	-2.16112
female:2	0.086084	0.189120	0.45518
age:1	-0.692652	0.062236	-11.12941
age:2	-0.324726	0.043277	-7.50348

Number of linear predictors: 2

Names of linear predictors: log(mu[,1]/mu[,3]), log(mu[,2]/mu[,3])

Dispersion Parameter for multinomial family: 1

Residual Deviance: 1544.874 on 1602 degrees of freedom

Log-likelihood: -772.437 on 1602 degrees of freedom

Number of Iterations: 5

$$\log\left(\frac{P(\text{hotel}=1)}{P(\text{hotel}=3)}\right) = 24.35 - .46 \text{female} - .69 \text{age}$$

$$\log\left(\frac{P(\text{hotel}=2)}{P(\text{hotel}=3)}\right) = 11.83 + .09 \text{female} - .32 \text{age}$$

- **Interpretation of multinomial logistic regression coefficients**

- With 1 unit change in age, log of P(Y=1)/P(Y=3) decreases by 0.69

log of P(Y=2)/P(Y=3) decreases by .32

- Exponentiate the coefficients:

```
exp(coef(outreg11b))
```

(Intercept):1	(Intercept):2	female:1	female:2
3.774414e+10	1.376740e+05	6.284910e-01	1.089898e+00
age:1	age:2		
5.002475e-01	7.227252e-01		

- With 1 unit change in age, P(Y=1)/P(Y=3) decreases by 0.50

P(Y=2)/P(Y=3) decreases by 0.72

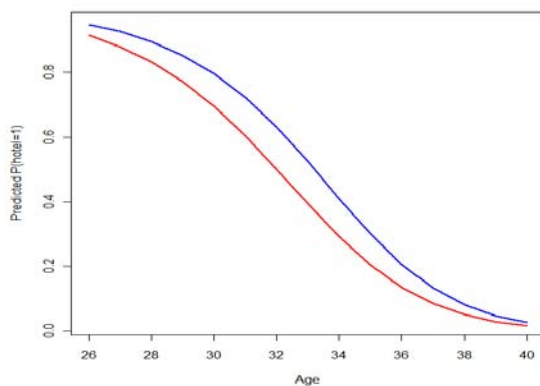
- For a binary predictor such as female:
 - $P(Y=1)/P(Y=3)$ for females (compared to males) is 0.63
 - $P(Y=2)/P(Y=3)$ for females (compared to males) is 1.09

- **Graphing multinomial logistic regression results**

```
nudatam <- data.frame(female=0, age=seq(26,40, 1))
male_predict <- predict(outregl0b, newdata = nudatam, type="response")
nudataf <- data.frame(female=1, age=seq(26,40, 1))
female_predict <- predict(outregl0b, newdata = nudataf, type="response")
```

```
male_predict
      1          2          3
1 0.94893382 0.04940055 0.001665625
2 0.92702440 0.06972287 0.003252726
3 0.89631858 0.09739456 0.006286859
4 0.85396579 0.13406055 0.011973658
5 0.79691970 0.18074375 0.022336544
6 0.72270019 0.23680731 0.040492501
7 0.63075561 0.29859757 0.070646816
8 0.52415605 0.35848746 0.117356488
9 0.41056429 0.40567921 0.183756498
10 0.30100145 0.42969310 0.269305453
11 0.20614582 0.42516013 0.368694046
12 0.13236428 0.39440010 0.473235618
13 0.08030981 0.34571881 0.573971380
14 0.04649828 0.28918749 0.664314227
15 0.02594381 0.23311200 0.740944187
```

```
male_predict1 <- male_predict[,1]
plot(male_predict1~nudatam$age, type="l",col="blue",lwd=2,ylab =
"Predicted P(hotel=1)", xlab="Age")
female_predict1 <- female_predict[,1]
lines(female_predict1~nudataf$age, col = "red", lwd=2)
```



7. Ordinal Logistic Regression

- Used when Y is ordinal – for example, Y is a measure of customer ratings of a restaurant on a 5-point Likert scale.
- Install and load R-package **Design**
- Install and load R-package **Hmisc**: you need this package for Design to work

Example #12

: A fast food chain wants to determine what factors affect size of soda (small, medium or large) customers order: factors being considered are.
(use data file soda_pref.csv)

X_1 = type of food ordered (chicken sandwich or hamburger)

X_2 = fries ordered or not

X_3 = age of customer

```
soda <- read.csv("K:/TEACH/DataMining_Fall2009/Data/soda_pref.csv",
header=TRUE)
attach(soda)
names(soda)
[1] "pref"  "food"  "fries" "age"
table(pref)
pref
 1    2    3
239 150  44
```

```
summary(age)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 20.00  21.00   21.00   21.03  21.00   22.00
```

```
xtabs(~food+pref)
      pref
food  1   2   3
 0 215 117  29
 1  24  33  15
```

```
xtabs(~fries+pref)
      pref
fries  1   2   3
 0 206 134  33
 1  33  16  11
```

- Many researchers use OLS regression: this is not recommended since OLS assumptions are violated when DV is not an interval scale variable.
- Ordinal Logistic Regression should be used – provided some of the cells do not have very low frequency counts: Xtabs on previous slide show this is not the case.

```

library(Design)
predictors <- soda(food, fries, age)
options(soda = 'predictors')
**note: the two lines above code is needed for Design

outreg12 <- lrm(pref~food+fries+age, data = xx, na.action = na.pass)
outreg12
#lrm: R performs logistic regression model
#na.action = na.pass: R skips missing data

Logistic Regression Model
lrm(formula = pref ~ food + fries + age, data = xx, na.action = na.pass)
Frequencies of Responses
  1  2  3
239 150 44
Obs Max Deriv Model L.R.      d.f.      P      C      Dxy
 433      1e-10      27.13      3      0      0.607      0.214
Gamma      Tau-a      R2      Brier
 0.322      0.121      0.072      0.234
Coef      S.E.      Wald      Z      P
y>=2 -12.116328  4.2666  -2.84  0.0045
y>=3 -14.185910  4.2810  -3.31  0.0009
food  1.033740    0.2496  4.14  0.0000
fries -0.004363    0.2872  -0.02  0.9879
age   0.558108    0.2030  2.75  0.0060

```

χ^2 - Likelihood Ratio value of 27.13 is significant (i.e., fitted model is better than no-predictor model)

****note: fries is not significant, food and age are.**

1 unit increase in age increases log-odd ratio of Y by .56.

```

exp(outreg12$coefficients)
      y>=2      y>=3
5.469473e-06  6.904583e-07
food      fries      age
2.811563e+00  9.956469e-01  1.747364e+00

```

- For 1 unit increase in age, the odds of the low and middle categories of pref versus the high category are 1.74 times larger (all other variables held constant).
- Proportional odds assumption implies that same is true (1.75 times increase) for low pref and combined middle+high pref.

8. Ridge Regression

- An alternative approach for multicollinearity data
- Look at IVs that are highly correlated, remove some of the variables
- Principal components regression compute principal component scores, use typically first 2-3 pc's instead of the original variables. In this case $vif = 1$, but interpreting pc's can become a challenge.
- Ridge regression yields a regression equation in terms of the original IVs.
- Install and upload R-package **MASS** for ridge regression
- Selecting the Ridge constant k
- When some of the predictors are correlated, the matrix $(X'X)$ is ill-conditioned, and adding a small positive k *reduces ill-conditioning*.
Another way to look at the ridge regression estimates is that it minimizes the error sum of squares while keeping all β 's from getting too large.
- Selecting the Ridge constant k
 - One way is to vary k from 0 to some upper value U (found by trial and error), plot the estimated coefficients of predictors as a function of k , and select smallest k for which the coefficients become stable.

Example #13

```
navy <-  
read.csv("K:/TEACH/DataMining_Fall2009/Data/Navy.csv", header=TRUE)  
attach(navy)  
outreg13a <- lm(Y~X1+X2+X3+X4+X5+X6+X7)  
summary(outreg12a)
```

Call:

```
lm(formula = Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	148.2206	221.6269	0.669	0.512613	
X1	-1.2874	0.8057	-1.598	0.128510	
X2	1.8096	0.5152	3.512	0.002673	**
X3	0.5904	1.8001	0.328	0.746931	
X4	-21.4817	10.2226	-2.101	0.050823	.
X5	5.6194	14.7562	0.381	0.708056	
X6	-14.5147	4.2262	-3.434	0.003163	**
X7	29.3603	6.3704	4.609	0.000250	***

```
library(HH)
```

```
vif(outreg13a)
```

X1	X2	X3	X4	X5	X6	X7
2.165539	4.500146	1.405882	2.352975	3.653326	37.184830	63.712775

****note: Variable X5, X6, X7 have high VIF values so let's look at the correlations among them.**

****note: combine X5, X6, X7 by columns to create a matrix v so correlation can be calculated.**

```
v <- cbind(X5, X6, X7)
```

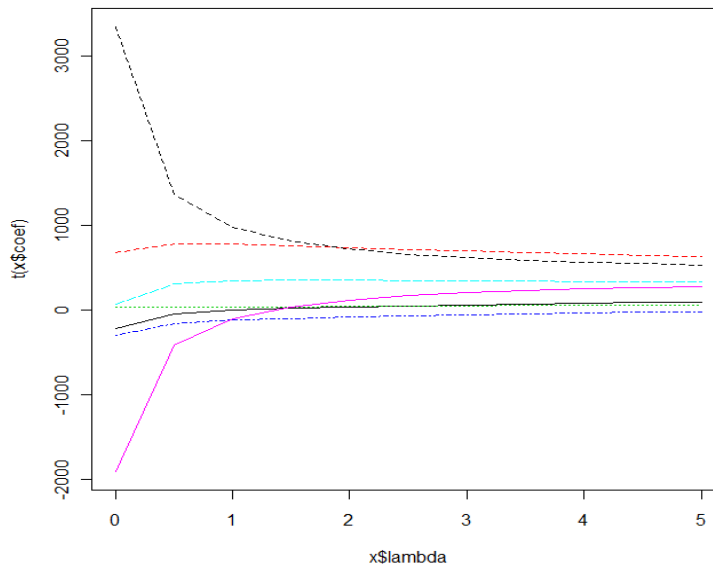
```
cor(v)
```

```
# cor: correlation
```

	X5	X6	X7
X5	1.0000000	0.6763194	0.7589389
X6	0.6763194	1.0000000	0.9781898
X7	0.7589389	0.9781898	1.0000000

```
library(MASS)
plot(lm.ridge(Yc~X1+X2+X3+X4+X5+X6+X7, lambda=seq(0, 5, 0.5)))
**note: run lm.ridge and plot the coefficients as function of k(lambda)
```

RIDGE TRACE PLOT for navy data



****note:** As λ increases β decreases.
 Choose smallest λ value where you discover β values become stable. From graph above, $\lambda = 1$.

****note:** Now run the linear model again → you will get new coefficient values for your equation model.

```
outreg13b <- lm.ridge(formula = Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7,
lambda=1)
outreg12b$coef
#lm.ridge: R will perform ridge regression
```

X1	X2	X3	X4
-1.169476	779.494163	36.992115	- 117.856248
X5	X6	X7	
346.294850	-95.989837	979.232177	

9. Poisson Regression

- You use Poisson regression when you have counted data.
- Install and load R-package **fields**, **lmtest**, **sandwich**
 - Function *stats* of library *fields* for obtaining descriptive statistics in a nicer format
 - **Waldtest** of library *lmtest* for testing overall model fit
 - **coefstest** of library *sandwich* for adjusting for heterogeneity (robust version of Poisson regression) – also needs *lmtest* .
- *predict* function used to predict using the fitted model.
- Poisson distribution can be thought of as the binomial distribution BIN(n,p) in the limiting case when number of trials n is very large, p is very small (i.e., we are looking at the occurrence of a rare event) in such a way that the mean np stays a constant : $np = \lambda > 0$.
- Variance of binomial = $np(1-p) = np - np^2$ will approach $\lambda - \lambda p$ or λ . In other words, for a Poisson random variable: MEAN = VARIANCE = λ

Example#14

: The data file poissonreg.csv has data on student's gender, math score, language/arts score and days absent (DV).

(use data file poissonreg.csv)

```
library(fields)
school2 <-
read.csv("K:/TEACH/DataMining_Fall2009/Data/poissonreg.csv",header=TRUE)
attach(school2)
sch <- school2[,3:7]
stats(sch)
```

****note:** compute summary stats of DV and lvs – use library fields and function stats to get a nicer output

****note:** no need to compute descriptives of id and school, so remove these columns [,3:7]

stats: summary stats

	male	math	langarts	daysatt
N	316	316	316	316
mean	0.487342	48.751	50.064	74.658
Std.Dev.	0.500633	17.881	17.939	11.467
Min	0	1.007	1.007	13
Q1	0	37.725	40.15	70
median	0	48.944	50	76
Q3	1	61.044	61.044	84
max	1	98.993	98.993	86
missing values	0	0	0	0

```
outreg14a<-glm(daysabs~math+langarts+male, family=poisson)
attach(outreg14a)
summary(outreg14a)
#family=poisson: run poisson regression in R
```

Call:

```
glm(formula = daysabs ~ math + langarts + male, family = poisson)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-4.0117	-2.5455	-1.1014	0.8956	11.0374

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.687666	0.072651	36.994	< 2e-16 ***
math	-0.003523	0.001821	-1.934	0.0531 .
langarts	-0.012152	0.001835	-6.623	3.52e-11 ***
male	-0.400921	0.048412	-8.281	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2409.8 on 315 degrees of freedom

Residual deviance: 2234.5 on 312 degrees of freedom

AIC: 3103.9

***note: Change in deviance from null (intercept only) model and the full model is a measure of model fit.*

```
library(lmtest)
```

```
waldtest(pr1, test="Chisq")
```

#lmtest: install and load lmtest for Wald's test to check model fit.

#waldtest: Wald's test to check model fit.

Wald test

Model 1: daysabs ~ math + langarts + male

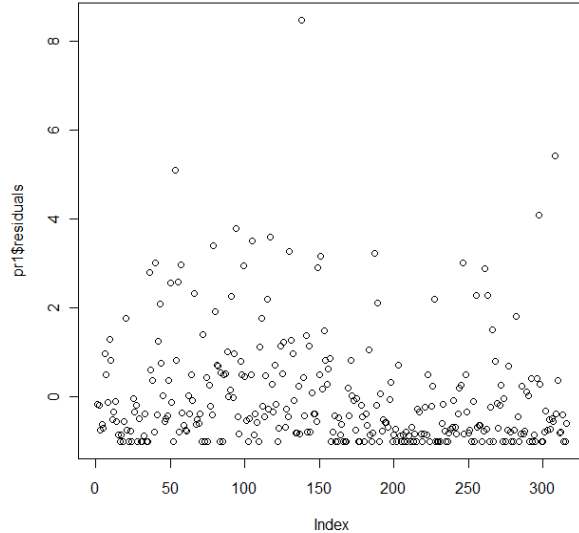
Model 2: daysabs ~ 1

	Res.Df	Df	Chisq	Pr(>Chisq)
1	312			
2	315	-3	176.24	< 2.2e-16 ***: (reject Null)

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

**note: Fitted model is doing significantly better than the null model. (= Model 2 is better than Model 1)*

```
plot(residuals)
```



****note:** You can see outliers → so you use the robust model of Poisson regression.

```
library(lmtest)
library(sandwich)
coefTest(outreg14a, vcov=sandwich)
```

#lmtest: you need this package to install sandwich

#sandwich: robust model of Poisson regression

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.6876659	0.2177980	12.3402	< 2.2e-16
math	-0.0035232	0.0076252	-0.4621	0.644044
langarts	-0.0121521	0.0052867	-2.2986	0.021527
male	-0.4009209	0.1393705	-2.8767	0.004019

****note:** Standard errors of estimates are quite different now, with math turning out to be not significant. We next run Poisson regression dropping math term.

```
outreg14b <- glm(daysabs~langarts+male, family=poisson)
attach(outreg14b)
summary(outreg14b)
```

Call:

```
glm(formula = daysabs ~ langarts + male, family = poisson)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-4.297	-2.510	-1.123	0.869	10.495

Coefficients:

Estimate	Std. Error	z value	Pr(> z)
----------	------------	---------	----------

```

(Intercept)  2.646976  0.069776  37.935  <2e-16 ***
langarts     -0.014670  0.001293  -11.342  <2e-16 ***
male         -0.409353  0.048219   -8.489  <2e-16 ***
Null deviance: 2409.8  on 315  degrees of freedom
Residual deviance: 2238.3  on 313  degrees of freedom
AIC: 3105.7

```

```
coefstest(outreg14b, cov=sandwich)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.6469765	0.0697764	37.9351	< 2.2e-16
langarts	-0.0146700	0.0012934	-11.3418	< 2.2e-16
male	-0.4093528	0.0482192	-8.4894	< 2.2e-16

```
waldtest(outreg14b, test = "Chisq")
```

Wald test

Model 1: daysabs ~ langarts + male

Model 2: daysabs ~ 1

Res.Df	Df	Chisq	Pr(>Chisq)
1	313		
2	315	-2 172.17	< 2.2e-16 ***

1 313

2 315 -2 172.17 < 2.2e-16 ***: ***note: this is the indicator whether your model is a good or not.*

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

- Once we have our final Poisson regression model, we can use it to predict values of days absent.
- Example: predict # of days for a student with math score = 75, language/arts score = 85.

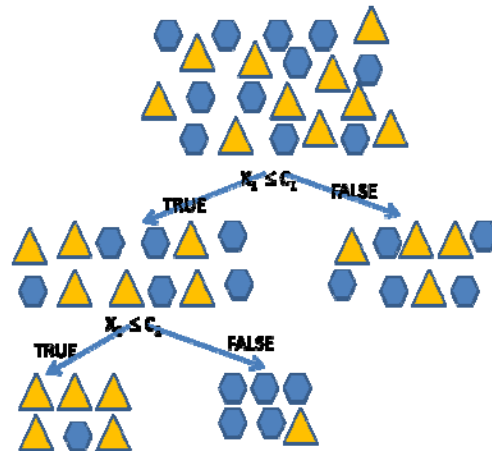
```

male<-c(0, 1)
langarts<-c(85)
new<-cbind(male, langarts)
fitted<-predict(outreg13b, newdata=data.frame(new),
type="response")
fitted
  1      2
4.05  2.693048

```

CART (Classification and Regression Trees)

- R-package: **tree**, **rpart**, **party**
- Idea behind CART
 - When data clustered and messy, fitting a smooth function may not be the thing to do.
 - We can instead split the data into homogeneous clusters, and then just compute the mean of each cluster (i.e., fit a constant).
- Recursive Partitioning is used to split data into homogeneous strata.
 - Parent node is split into 2 children nodes – process repeated for each child node.
 - An **impurity measure** is used to select splitting variable at each stage.



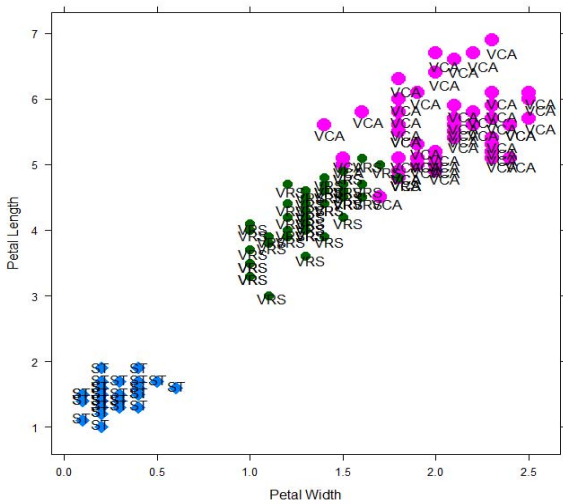
Possible Stopping Criteria

- Max #(nodes)
 - Min # cases in a node reached
- Impurity Measures
 - Continuous Variable – sample variance
 - Qualitative Variable : several approaches
 - Install and load R-package **tree**, **lattice**, **rpart**, **party**

Example #1

: use iris data

```
iris <- read.csv("M:/TEACH/DataMining_Fall2009/Data/iris.csv", header=TRUE)
attach(iris)
library(lattice)
library(tree)
xyplot(Petal.Length~Petal.Width, data=iris, groups=Species)
#xyplot: install and load R-package lattice for xyplot
```

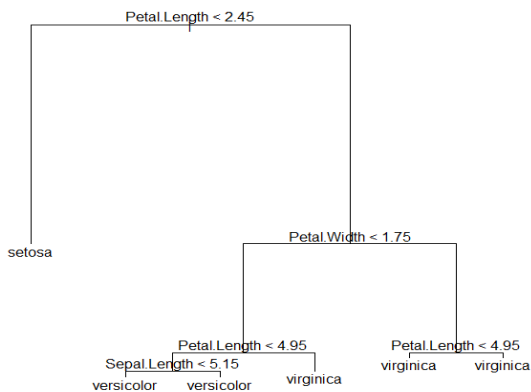


```
ir.tr <- tree(Species ~., iris)
```

***note: What does tree do?*

Classify into segments by mean value. You are predicting if they fall into the right group.

```
plot(ir.tr); text(ir.tr)
```



Example #2

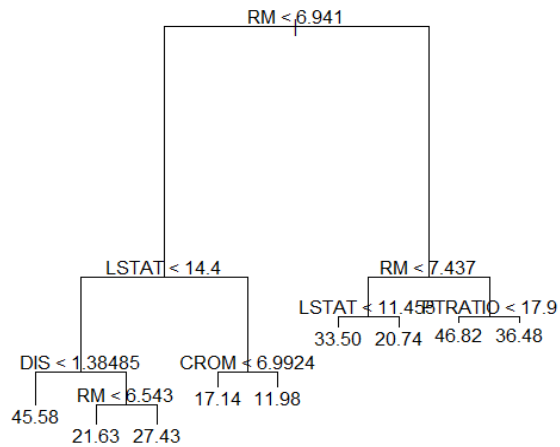
: use bostonhousing data

1) R-package Tree

```
library(tree)
```

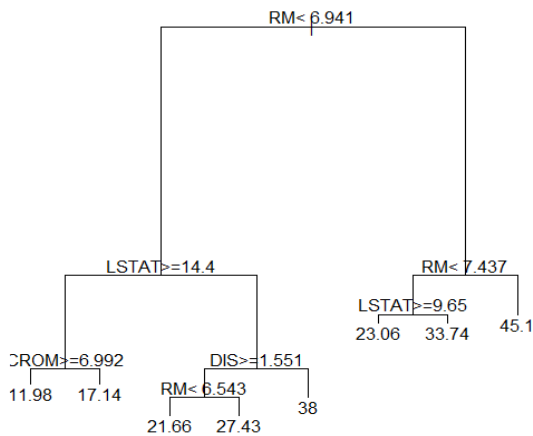
```
boston.tr <- tree(MEDV ~., boston)
```

```
plot(boston.tr); text(boston.tr)
```



2) R-package rpart

```
library(rpart)
fit <-
rpart(MEDV~CROM+ZN+INDUS+CHAS+NOX+RM+AGE+DIS+RAD+TAX+PTRATIO+B+LSTAT,
data=boston)
plot(fit)
text(fit)
```



3) R-package party

- R-package party: uses a parametric model in recursive partitioning
 - A parametric model is fitted to the full data.
 - Parameter instability is tested over a set of partitioning variables.
 - The variable with highest instability is used to split data.
 - Procedure is repeated for each child node.

```

library(party)
boston$CHAS <- factor(boston$CHAS, levels=0:1, labels = c("no", "yes"))
boston$RAD <- factor( boston$RAD, ordered = TRUE)
control.par <- mob_control(alpha=.05, bonferroni=TRUE, minsplit=40,
objfun=deviance, verbose=TRUE)
#factor: Convert CHAS and RAD to factors
#mob_control: OLS is used for the model fitting
#bonferroni: instability of parameter is assessed using Bonferroni

```

```

mob1 <- mob(MEDV~LSTAT+RM|CROM+CHAS+NOX+DIS+PTRATIO+B+RAD, control =
control.par, model = linearModel)

```

***note: model MEDV as a function of LSTAT and RM plus all partitioning variables*

Fluctuation tests of splitting variables:

	CROM	CHAS	NOX	DIS	PTRATIO
statistic	7.399234e+01	4.908879e+01	6.321138e+01	1.013319e+02	
	7.60051e+01				
p.value	1.006287e-13	3.258859e-08	2.555821e-11	6.655466e-20	3.55881e-14

	B	RAD
statistic	3.339619e+01	7.685939e+01
p.value	7.302362e-05	2.288228e-14

Best splitting variable: DIS
Perform split? yes

Node properties:

DIS <= 1.6582; criterion = 1, statistic = 101.332

Fluctuation tests of splitting variables:

	CROM	CHAS	NOX	DIS	PTRATIO
statistic	8.187771e+01	4.110237e+01	9.560163e+01	3.497670e+01	
	7.997064e+01				
p.value	1.685016e-15	1.708528e-06	1.323506e-18	3.388114e-05	4.530194e-15

	B	RAD
statistic	7.067411e+01	8.343392e+01
p.value	5.515470e-13	7.511532e-16

Best splitting variable: NOX
Perform split? yes

Node properties:

NOX <= 0.655; criterion = 1, statistic = 95.602

Fluctuation tests of splitting variables:

	CROM	CHAS	NOX	DIS	PTRATIO
statistic	3.457337e+01	12.8541418	22.21193255	19.97970227	8.014125e+01
p.value	4.080716e-05	0.4898428	0.01306554	0.03494108	4.104767e-15

	B	RAD

statistic 12.5715672 21.61332177
p.value 0.5283952 0.01705065

Best splitting variable: PTRATIO
Perform split? yes

Node properties:
PTRATIO <= 19.2; criterion = 1, statistic = 80.141

Fluctuation tests of splitting variables:

	CROM	CHAS	NOX	DIS	PTRATIO
B					
statistic	19.9675392	10.8026857	16.1420955	22.06771089	4.134665e+01
	11.8472459				
p.value	0.0265156	0.7113743	0.1323357	0.01021371	8.179616e-07
	0.5622984				

	RAD
statistic	17.23960984
p.value	0.08478723

Best splitting variable: PTRATIO
Perform split? yes

Node properties:
PTRATIO <= 14.9; criterion = 1, statistic = 41.347

Fluctuation tests of splitting variables:

	CROM	CHAS	NOX	DIS	PTRATIO	B
statistic	9.9762064	12.4129558	8.3077281	11.2807251	8.980504	10.1135121
p.value	0.7880782	0.4519443	0.9460855	0.6086947	0.896195	0.7705713

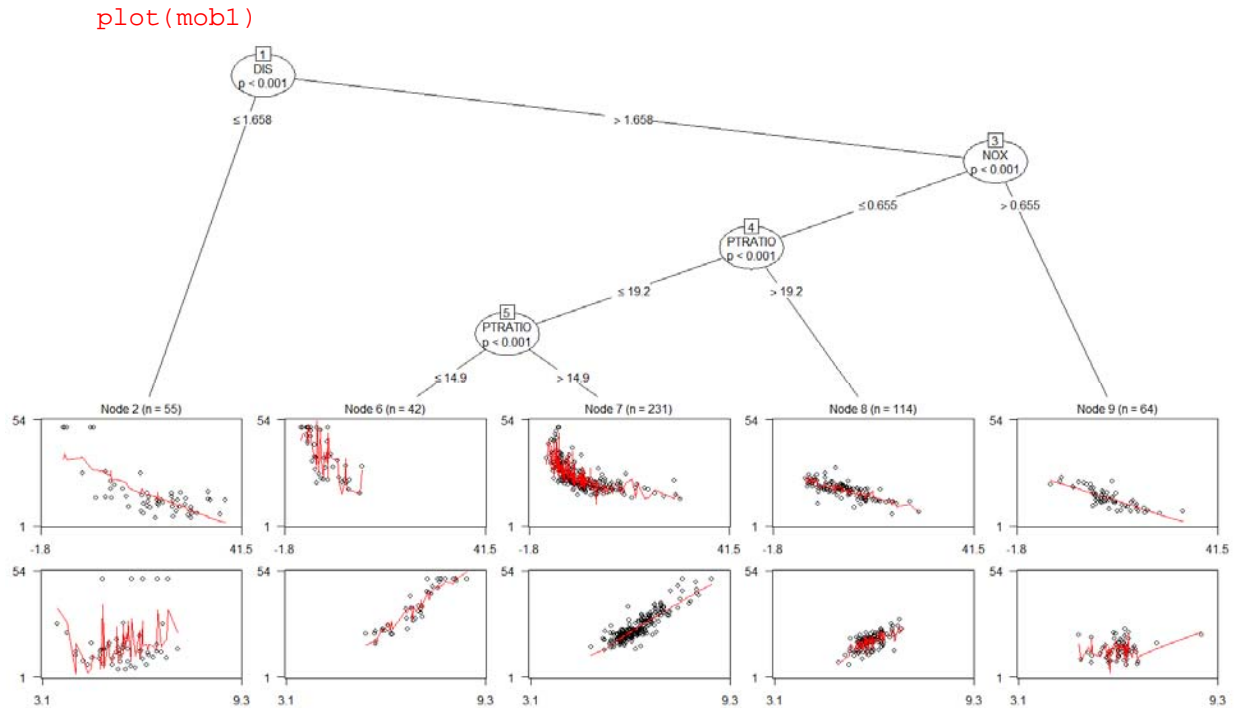
	RAD
statistic	6.6781977
p.value	0.9955196

Best splitting variable: CHAS
Perform split? no

Fluctuation tests of splitting variables:

	CROM	CHAS	NOX	DIS	PTRATIO	B	RAD
statistic	5.0424258	1.575127	4.2758081	10.1220794	8.649540	3.922884	9.342154
p.value	0.9951466	1.000000	0.9993386	0.5162606	0.725515	0.999801	0.627116

Best splitting variable: DIS
Perform split? no



Example #3

: Regression with CART

The R-package mlbench has a benchmark data set – Pima Indians Diabetes data.

Install and load mlbench

```

Library(mlbench)
data("PimaIndiansDiabetes2", package = "mlbench")
pid <- PimaIndiansDiabetes2
head(pid)

```

```

  pregnant glucose pressure triceps insulin mass pedigree age diabetes
1         6     148      72      35      NA  33.6   0.627  50      pos
2         1      85      66      29      NA  26.6   0.351  31      neg
3         8     183      64      NA      NA  23.3   0.672  32      pos
4         1      89      66      23     94  28.1   0.167  21      neg
5         0     137      40      35    168  43.1   2.288  33      pos
6         5     116      74      NA      NA  25.6   0.201  30      neg

```

head: prints the 1-st 6 lines of a dataframe

```

library(fields)
stats(pid)

```

#fields: install and load R-package fields for stats.

stats: yields descriptive statistics for each variable.

```

  pregnant  glucose  pressure  triceps  insulin      mass
N          768.000000 763.00000 733.00000 541.00000 394.0000
757.000000

```

```

mean          3.845052 121.68676  72.40518  29.15342 155.5482
32.457464
Std.Dev.     3.369578  30.53564  12.38216  10.47698 118.7759
6.924988
min          0.000000  44.00000  24.00000   7.00000 14.0000
18.200000
Q1           1.000000  99.00000  64.00000  22.00000 76.2500
27.500000
median       3.000000 117.00000  72.00000  29.00000 125.0000
32.300000
Q3           6.000000 141.00000  80.00000  36.00000 190.0000
36.600000
max          17.000000 199.00000 122.00000  99.00000 846.0000
67.100000
missing values 0.000000   5.00000  35.00000 227.00000 374.0000
11.000000

          pedigree      age diabetes
N          768.0000000 768.00000      NA
mean        0.4718763  33.24089      NA
Std.Dev.    0.3313286  11.76023      NA
min         0.0780000  21.00000      NA
Q1          0.2437500  24.00000      NA
median      0.3725000  29.00000      NA
Q3          0.6262500  41.00000      NA
max         2.4200000  81.00000      NA
missing values 0.0000000  0.00000      NA

```

****note:** Variables *triceps* and *insulin* have a large number of missing (NA) values → remove
Removing NA values: use *na.omit*

- 1) First, remove *triceps* and *insulin*
- 2) Then remove NA values

```

pid1 <- pid[,-c(4,5)]
pid2 <- na.omit(pid1)

```

****note:** first remove *triceps* and *insulin* then remove NA

```

dim(pid3)
[1] 768  7

```

```

pid3 <- na.omit(pid)
pid4 <- pid3[,-c(4,5)]

```

na.omit: removes NA values as all case

****note:** then remove *triceps* and *insulin*

```

dim(pid2)
[1] 392  7

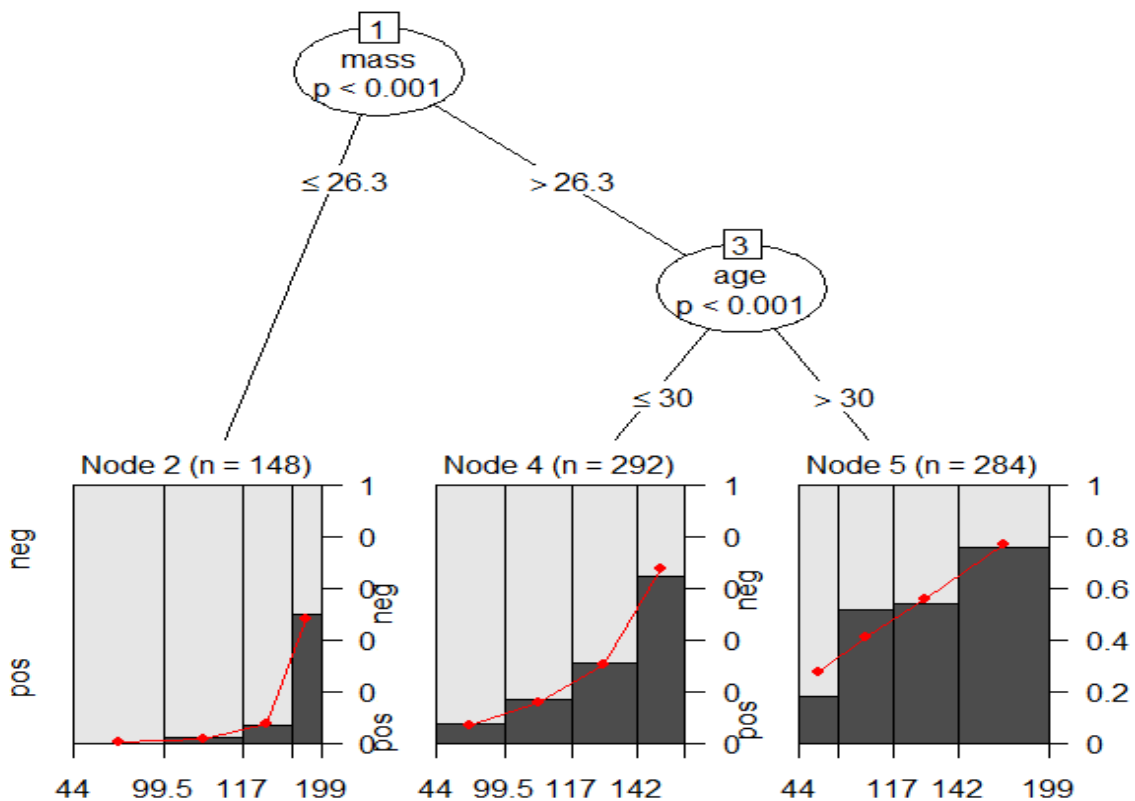
```

****note:** If the NA values are removed from the data first, and then the variables *triceps* and *insulin* removed, we lose too many observations

```

log1 <-
mob(diabetes~glucose|pregnant+glucose+pressure+mass+pedigree+age, data
= pid3, model=glinearModel,family=binomial())
plot(log1)

```



plots regression tree

- Node 2: women with low BMI have lower risk of diabetes, risk goes up with glucose.
- Node 4: women with average and high bmi , ≤ 30 years have higher average risk which increases with glucose.
- Node 5: women with average and high bmi , > 30 years have higher average risk which increases slowly with glucose.

```
coef(log1)
      (Intercept)  glucose
2 -10.999447  0.06456780
4  -6.573067  0.04504490
5  -3.318569  0.02748038
```

```
exp(coef(log1)[, 2])
      2      4      5
1.066698 1.046075 1.027861
```

**note: Odds go up by 6.5%, 4.5%, and 2.7% with glucose in the three partitions.

Random Forest

- Used for classification.
- In Random Forests, many classification trees are grown.
- The classification having the most votes (over all the trees in the forest) is the RF classification.
- Random Forest vs CART
 - CART artificially favor splits in variables with large number of categories or continuous variables, and hence are BIASED.
 - Random Forest uses an unbiased tree algorithm.
- Package **Party** (→cforest) and **randomForest** (→randomForest)
- **Variable Importance (VI) Measure in Random Forests**
 - Class membership of each OOB (out of bag) sample is predicted using each tree in the forest, and # of correctly classified samples (CCO) is counted.
 - Values of variable X_j are permuted in the OOB sample, and class membership of OOB samples are predicted again from the tree, and # of correctly classified samples after permutation (CCP) is counted.
 - $VI = (CCO - CCP) / \#(\text{OOB samples})$

Example #1

: data file readingSkills (download **party** to get data)

****note: First download two packages party and randomForest and install.**

1) Using package randomForest

```
library(party)
library(randomForest)
tree1.rf <- randomForest(score~., data = na.omit(readingSkills),
importance=TRUE, proximity=TRUE)
importance(tree1.rf)
```

```
      %IncMSE IncNodePurity
nativeSpeaker 48.93136      1985.680
age           31.07910      5618.695
shoeSize      23.36171      4623.487
```

#importance: print importance of variables

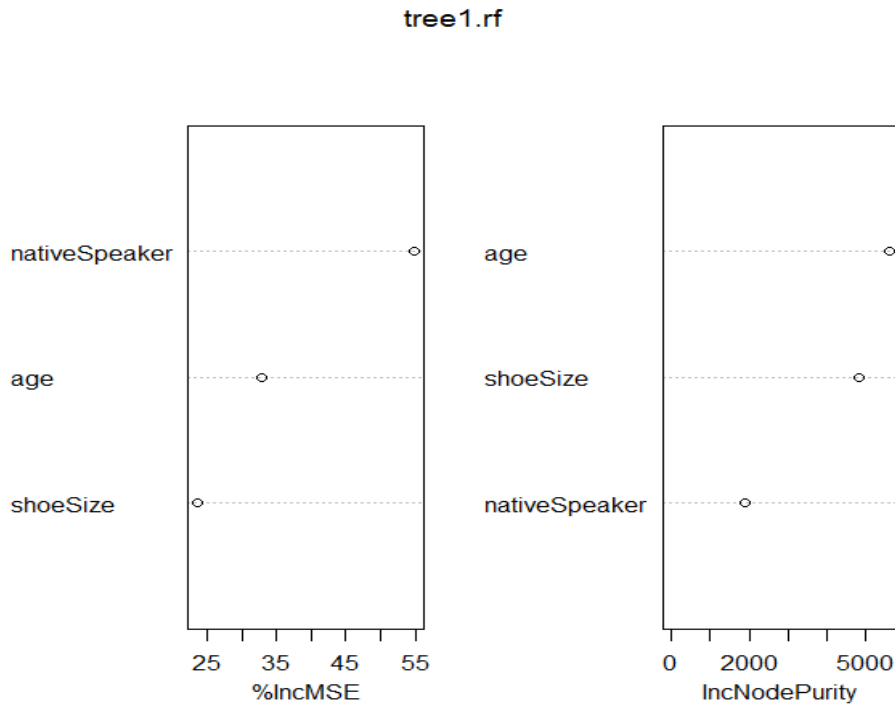
```
pred0 <- cbind(readingSkills$nativeSpeaker, readingSkills$age,
readingSkills$shoeSize)
pred00 <- na.omit(pred0)
cor(pred00)
```

```
      [,1]      [,2]      [,3]
[1,] 1.0000000 0.1355429 0.0958855
[2,] 0.1355429 1.0000000 0.8969335
[3,] 0.0958855 0.8969335 1.0000000
```

cor: checking correlation

****note:** Age and shoeSize are highly correlated ($r = 0.87$) – this is the reason for shoeSize to turn out as more important than nativeSpeaker

`varImpPlot(tree1.rf)`



***note: plot importance of variables*
%IncMSE: importance variable
IncNodePurity: actual value

- Variable importance
 - Unconditional variable importance: computed by permuting each predictor (1 at a time) in OOB sample. Can be misleading when predictors are highly correlated.
 - Conditional variable importance: Takes care of high correlation. You are not destroying the correlation on other variables.

UNCONDITIONAL IMPORTANCE			CONDITIONAL IMPORTANCE		
Y	Xj	Z	Y	Xj	Z
Y1	P	Z1	Y1	per	Z1=a
.	E	.	.	mute	Z12=a
.	r	.	.	per	Z37=a
.	M	.	.	mute	Z5=b
Yj	U	Zj	Yj	per	Z14=b
.	T	.	.	mute	Z140=b
.	e	.	.	per	
.	D	.	.	mute	
Yn		Zn	Yn	per	
				mute	

2) Using package party

```
tree1.party <- cforest(score~., data = na.omit(readingSkills),
control=cforest_unbiased(mtry=2, ntree=50))
varimp(tree1.party)
nativeSpeaker      age      shoeSize
      12.18944      84.92342      14.71297
```

***note: shoesize is more important than nativeSpeaker, since shoeSize is correlated with Age.*

```
varimp(tree1.party, conditional = TRUE)
nativeSpeaker      age      shoeSize
12.051017      46.037991      1.384335
```

***note: #conditional importance takes care of the problem.*

Example #2

: data file cu.summary (download **rpart** to get data)

The data frame cu.summary has 17 rows and 5 columns – data is from April 1990 issue of Consumer Reports.

```
library(rpart)
tree2.rf <- randomForest(Mileage~Price + Country + Reliability + Type,
data = na.omit(cu.summary), importance=TRUE, proximity=TRUE)
importance(tree2.rf)
```

	%IncMSE	IncNodePurity
Price	21.682851	406.36451
Country	2.257068	109.15904
Reliability	4.473107	98.42476
Type	17.310007	305.09807

```
print(tree2.rf)
```

Call:

```
randomForest(formula = Mileage ~ Price + Country + Reliability +
Type, data = na.omit(cu.summary), importance = TRUE, proximity = TRUE)
```

```
  Type of random forest: regression
```

```
  Number of trees: 500
```

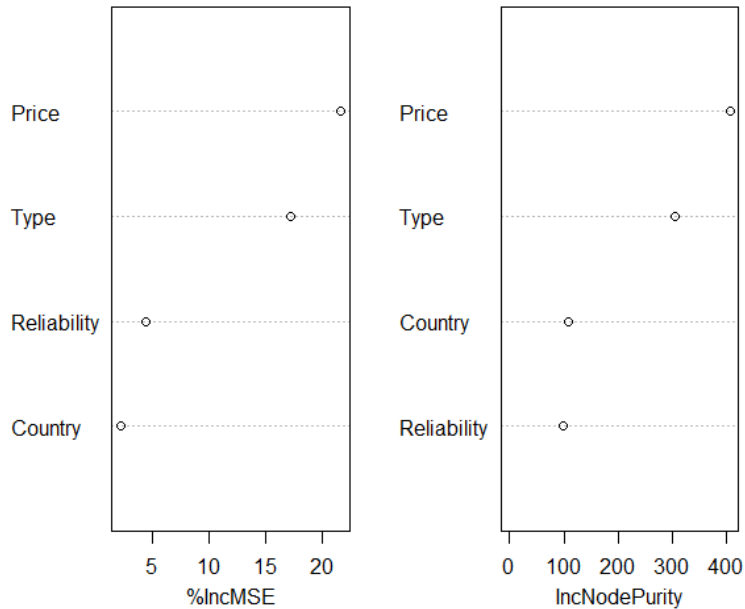
```
No. of variables tried at each split: 1
```

```
Mean of squared residuals: 9.813248
```

```
% Var explained: 56.76 →**note: similar to Rsquare
```

```
varImpPlot(tree2.rf)
```

tree2.rf



```
predcu <-  
cbind(cu.summary$Price,cu.summary$Country,cu.summary$Reliability,cu.summary$Type)  
predcul <- na.omit(predcu)  
cor(predcul)  
      [,1]      [,2]      [,3]      [,4]  
[1,] 1.00000000 0.006682906 -0.14709251 -0.20823617  
[2,] 0.006682906 1.000000000 -0.67467732 -0.02974314  
[3,] -0.147092508 -0.674677317 1.000000000 -0.02352221  
[4,] -0.208236170 -0.029743141 -0.02352221 1.00000000  
**note: Predictors are not highly correlated.
```

3) Predict test data using randomForest

Example #3

: use iris data

```
iris <- read.csv("K:/TEACH/DataMining_Fall2009/Data/iris.csv",  
header=TRUE)  
attach(iris)  
set.seed(43)  
r <- nrow(iris)  
ind <- sample(r, round(.25*r), replace=TRUE)
```

#set.seed: used to set the random number seed.

***note*: When we use `runif` to generate random number we almost get different set of random number.

```
runif(5)
[1] 0.2096388 0.3427873 0.5455948 0.7694844 0.4287647
```

```
runif(5)
[1] 0.6864617 0.5218690 0.7965364 0.9030520 0.4324572
```

****note:** But in some cases, we want the results reproducible → then we use `set.seed`

```
set.seed(100)
runif(5)
[1] 0.30776611 0.25767250 0.55232243 0.05638315 0.46854928
```

```
set.seed(100)
runif(5)
[1] 0.30776611 0.25767250 0.55232243 0.05638315 0.46854928
```

****note:** Let's try to split data into training and test, and generating approximately 25% in test

```
iris.train <- iris[-ind,]
iris.test <- iris[ind,]
iris.rf <- randomForest(Species ~ ., data=iris.train)
iris.pred <- predict(iris.rf, iris.test)
table(observed = iris.test[,5], predicted = iris.pred)
```

observed	predicted		
	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	10	1
virginica	0	0	12

```
iris.rf
Call:
randomForest(formula = Species ~ ., data = iris.train)
  Type of random forest: classification
    Number of trees: 500
No. of variables tried at each split: 2
OOB estimate of error rate: 5.98%
```

Confusion matrix:

	setosa	versicolor	virginica	class.error
setosa	41	0	0	0.00000000
versicolor	0	35	3	0.07894737
virginica	0	4	34	0.10526316

Time Series

Time series analysis is used on an ordered sequence of values of a quantitative random variable at equally spaced time points.

- Applications: Economic forecasting, Demand forecasting, Stock prices forecasting, Sales projections.
 - e.g. monthly time series of coin-in
- Procedure
 - Perform a time series analysis on data.
 - Check the data pattern(trend) by producing plots
 - Take care of the variance (stabilize data)
 - Data for Time Series Analysis: You need **STATIONARY** data!!!
 - You might need to **TRANSFORM** data depending on pattern (log, difference, etc).
 - Working series: the actual data series you work with → keep it simple!!!

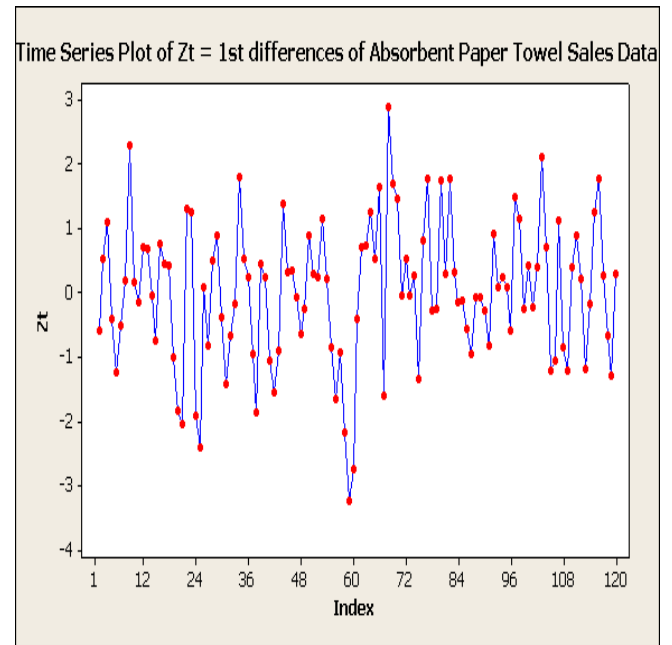
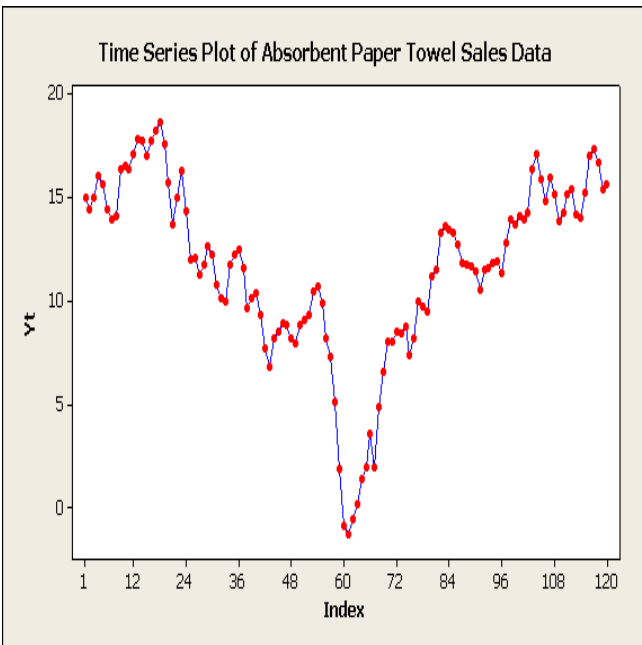


Figure 1(a): non-stationary time series

Figure 1(b): stationary time series

- Plot ACF and PACF and choose lags
- Use ARIMA or SARIMA for forecasting.

Example# 1

: The data file Maine.dat has monthly unemployment figures for maine.
(Use R-code time series1.txt file)

```
www <- http://www.massey.ac.nz/~pscowper/ts/Maine.dat
Maine.month <- read.table(www, header=TRUE)
attach(Maine.month)
class(Maine.month) # output = [1] "data.frame"
Maine.month.ts <- ts(unemploy, start = c(1996,1), freq = 12)
Maine.annual.ts <- aggregate(Maine.month.ts)/12
#ts: create a time series object
#aggregate: aggregate over year

layout(1:2)
plot(Maine.month.ts, main = "Monthly unemployment")
plot(Maine.annual.ts, main = "Yearly average unemployment")
```



***note: Monthly employment: you see spikes where in indicates there is a trend, thus we can use time series*

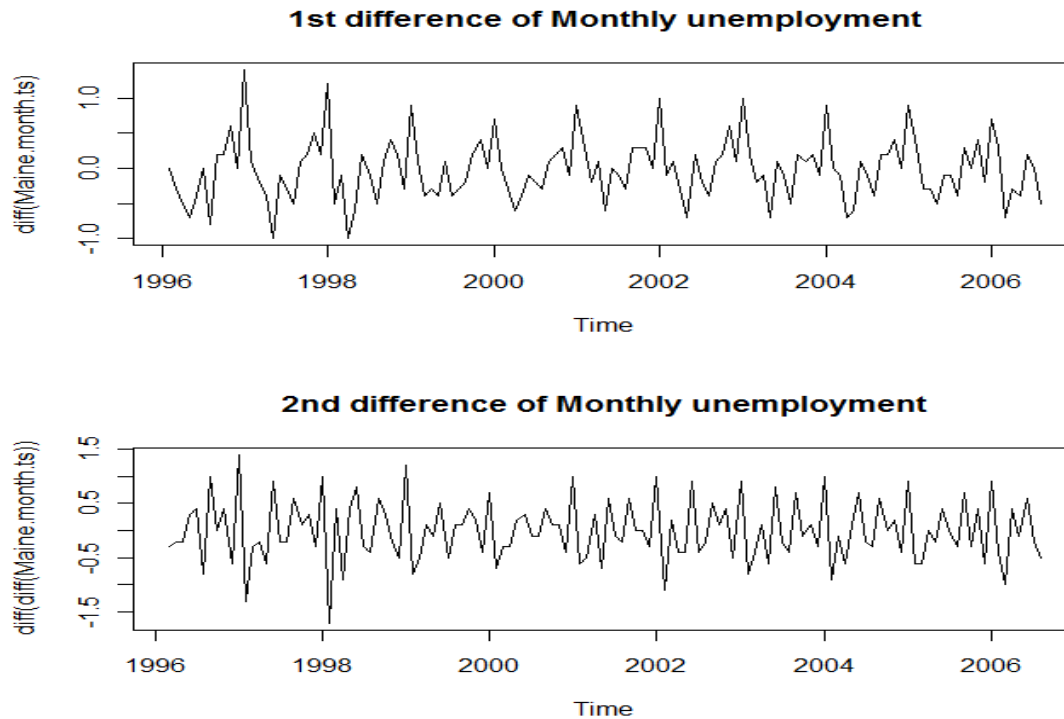
- 1) trend goes up
- 2) then variability is also going up with time t
Time series Y_t is NOT stationary.

- Data for ARIMA model

- ARIMA modeling requires Y_t to be STATIONARY.
 $E(Y_t) = \text{constant} = \text{Mean}(Y_t)$
 $Va(Y_t) = \text{constant}$
- What to do:
 - 1) either take a log transform or $\sqrt{Y_t}$ to get $\text{var}(Y_t) = \text{constant}$ (normally log and root will work)
 - 2) or fit a trend line to take care of non-constant mean by taking the difference

```
plot(diff(Maine.month.ts), main = "1st difference of Monthly unemployment")
plot(diff(diff(Maine.month.ts)), main = "2nd difference of Monthly unemployment")
```

****note:** plot 1-st and 2-nd order differences of the monthly time series



****note:**

<i>1st diff</i>	<i>2nd diff</i>
$Y_2 - Y_1 = U_1$	$U_2 - U_1$
$Y_3 - Y_2 = U_2$	$U_3 - U_2$

- 1) If the original time series exhibits a linear trend, and $\text{var}(Y_t)$ is constant, 1st difference will yield a stationary time series.
- 2) If the trend is quadratic (one bend in graph), take first difference of Y_t and calculate 2nd difference.
- 3) If $\text{var}(Y_t)$ is non-constant, a log-transform or a square-root transform may yield a time series with constant variance.

1) ARIMA Modeling

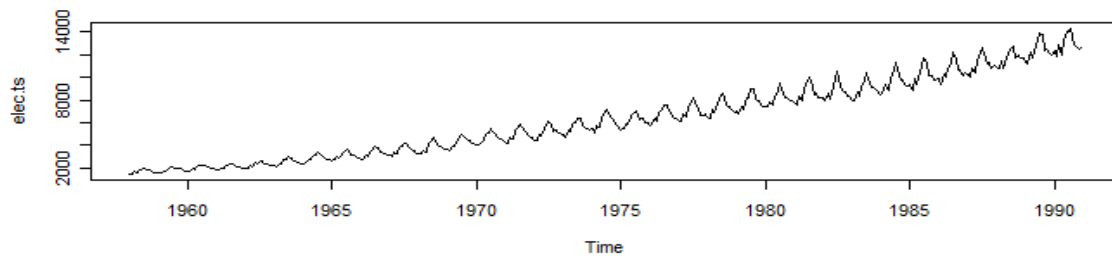
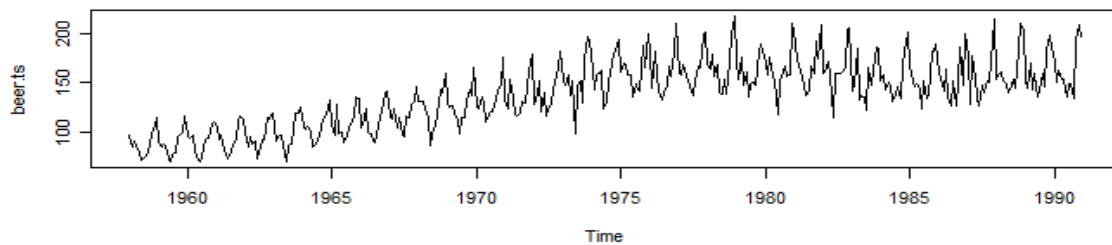
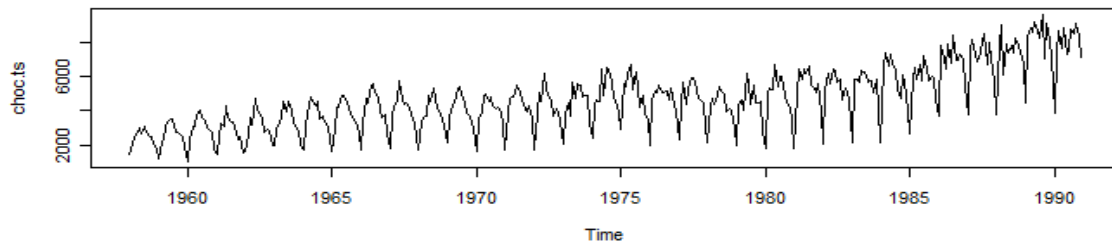
- A time series is stationary if: $E(Y_t) = \mu$, $\text{Var}(Y_t) = \sigma^2$ for all t
 - In other words, if y_1, y_2, \dots, y_n values of the time series fluctuate around a constant mean with constant variation, the time series is stationary.
 - If the n values do not seem to fluctuate around a constant mean or do not fluctuate with constant variation around a constant mean, then it is non-stationary.

Example #2

: monthly supply of chocolate and beer

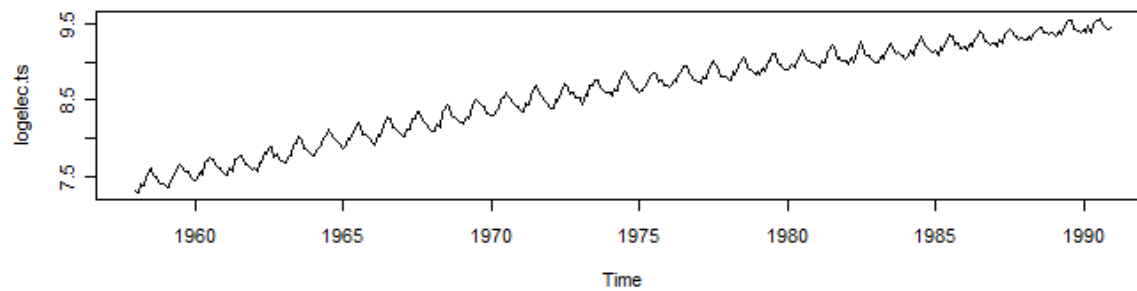
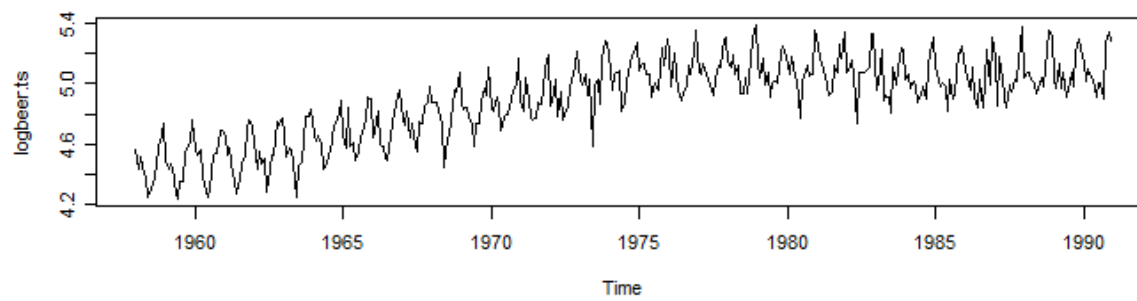
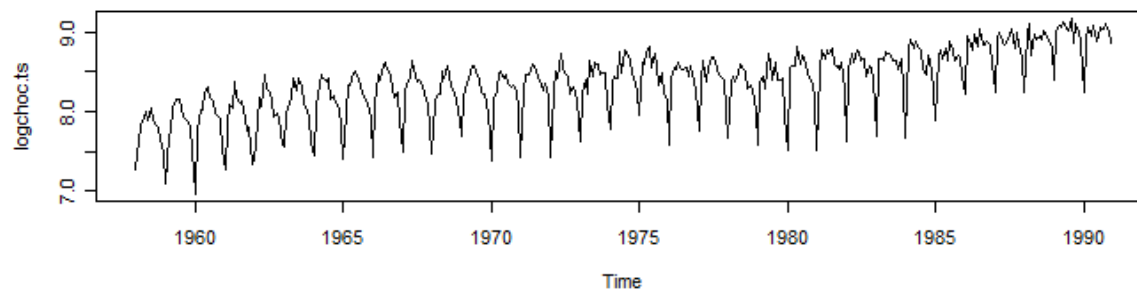
```
www <- "http://www.massey.ac.nz/~pscowper/ts/cbe.dat"
CBE <- read.table(www, header=T)
names(CBE)
1] "choc" "beer" "elec"
choc.ts <- ts(CBE[,1], start=1958, freq=12)
beer.ts <- ts(CBE[,2], start=1958, freq=12)
elec.ts <- ts(CBE[,3], start=1958, freq=12)
# freq: performing time series analysis on monthly data (12 months)

plot(choc.ts)
plot(beer.ts)
plot(elec.ts)
```



***note: each of the three time series is non-stationary (non-constant mean, non-constant variance). By using log, you are making the variability more constant.*

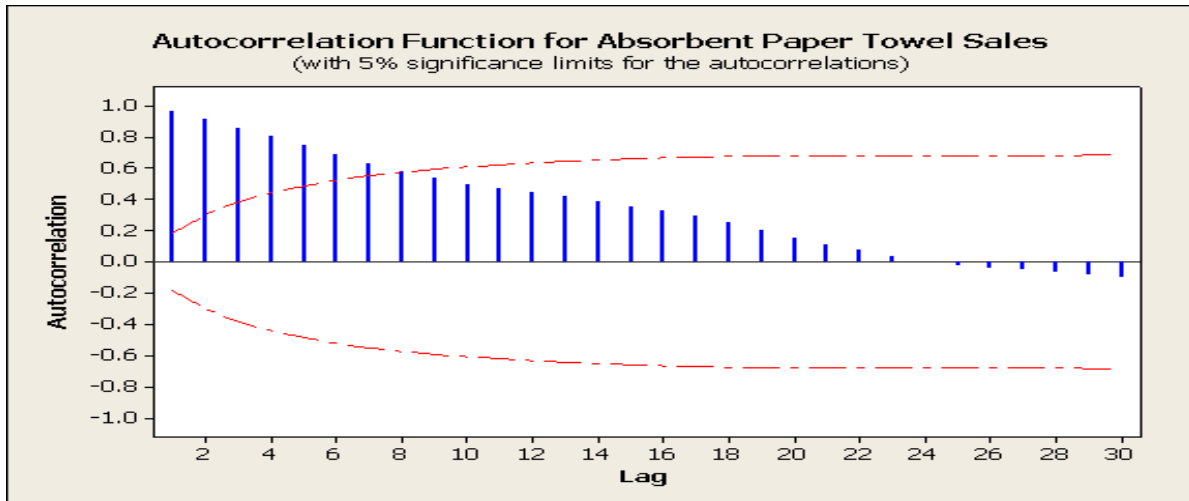
```
logchoc.ts <- log(choc.ts)
logbeer.ts <- log(beer.ts)
logelec.ts <- log(elec.ts)
plot(logchoc.ts)
plot(logbeer.ts)
plot(logelec.ts)
```



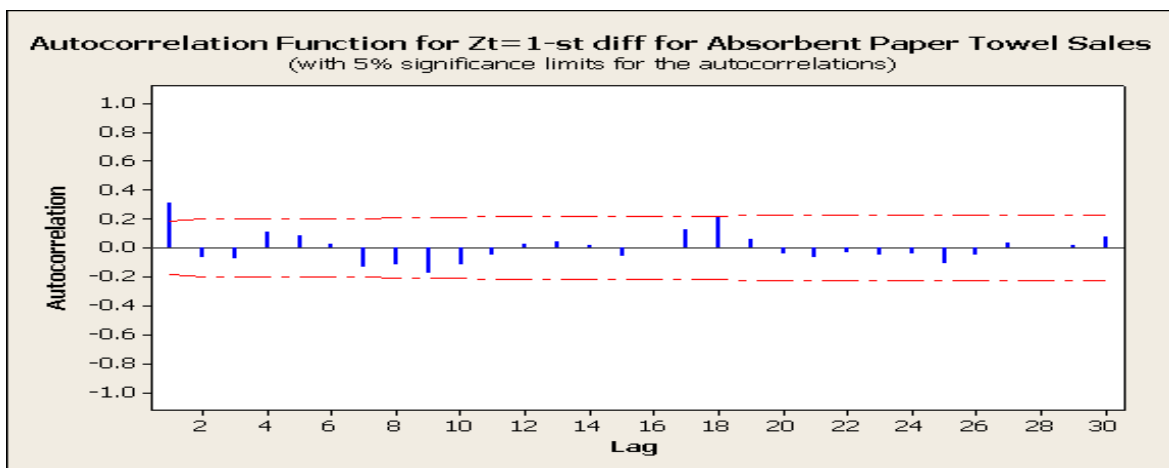
2) ARIMA MODEL IDENTIFICATION

Autoregressive Integrated Moving Average Process

Integrated: taking difference to get a stationary series



***note: SAC cuts off very slowly (after lag $k = 7$) as r_k is significantly different from 0 for $k \leq 7$; time series is non-stationary.*



***note: SAC cuts off quickly (after lag $k = 1$) as r_k is not significantly different from 0 for $k > 1$; working series is stationary.*

- **Determination of ARIMA(p,d,q) Model**

- ARIMA (p1, d1, q1) * (p2, d2, q2)
p1, d1, q1: non-seasonal , p2, d2, q2: seasonal
p1 = order of AR term (auto regressive)
d1 = order of differencing
q1 = order of MA term (moving average)

- How to identify p, d, q values:
 - Spikes on the ACF and PACF plots indicate significance (higher the spikes, more significant)
 - **NOTE!!! If plots do not show any spikes, it indicates that they lie in between the CI which means there is no significance: no need to proceed.*
 - Choose the value for p and q from either of the two ACF or PACF plots.
- Determine AR or MA model
 - **AR model (Autoregressive): (p, d, 0)**
 - ACF: if spikes die down
 - PACF: if spikes are high at first few lags and then cuts off

If it is an AR model, p will be the cut off lag value, and q will be 0
 - **MA model (Moving Average): (0, d, q)**
 - ACF: if spikes are high at first few lags and then cuts off
 - PACF: if spikes die down

If it is a MA model, q will be the cut off lag value, and p will be 0

***note: You are really trying to get 1 or 2 orders of spikes.*
- Difference (d): whether you used difference to stabilize data or not
 - 1: if you did
 - 0 : if you did not
- **Example:**
 - If **ACF dies down** and **PACF has spike at lag 2, cuts off past lag 2** (→ AR model)
 - No trend: fit ARIMA(2,0,0) : since this is an AR model, lag 2 indicates p
 - Trend: ARIMA (2,1,0)
 - If **PACF dies down** and **ACF has spike at lag 2, cuts off past lag 2** (→ MA model)
 - No trend: ARIMA(0,0,2): since this is a MA model, lag 2 indicates q
 - Trend: ARIMA (0,1,2)
- Other ARIMA models
 - ARIMA (0,1,0): Random walk model. The only effect is a non-seasonal differencing to remove a linear trend. ACF is either constant or is balanced between positive and negative. PACF is spiked only at lag 1.
 - ARIMA (p,0,q): ACF and PACF both decline slowly toward 0. PACF declines erratically due to shock effects.

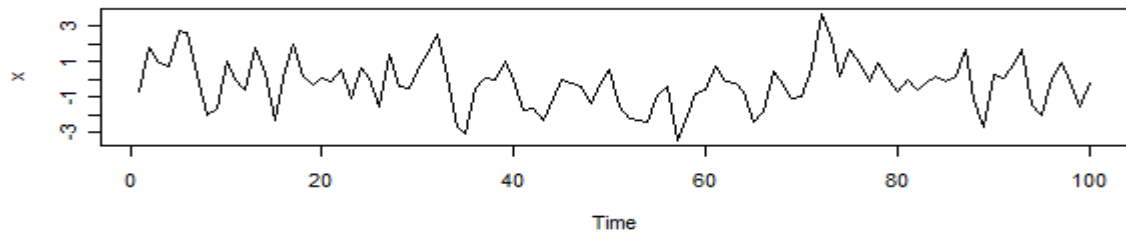
Example #3

: Time series analysis using ARIMA, you have a data set named x
(SIMULATION and FITTING)

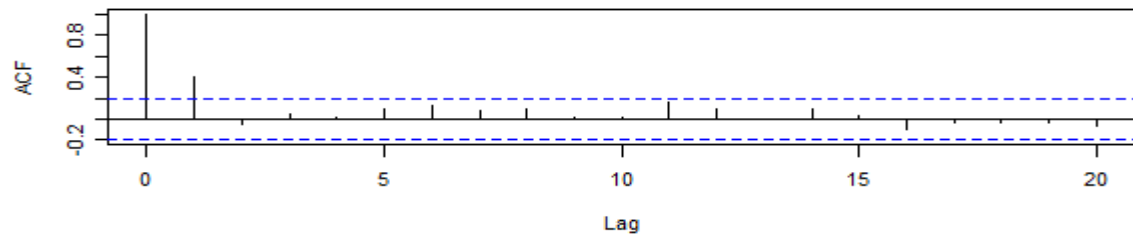
```
x.ts<- ts(x)
layout(1:3)
plot(x)
acf(x)
pacf(x)
x.ma<- arima(x.ts, order = c(0,0,1))
x.ar <- arima(x.ts, order = c(1,0,0))
x <- arima.sim(list(order=c(1,0,0), ar=.9), n=100)
# arima.sim: used when you do not have an actual data set but you are simulating
simulate ARIMA (1,0,0) time series when  $\alpha = .9$ .
**note: This is an AR model so you are providing ar value.
```

```
layout(1:3)
plot(x)
acf(x)
pacf(x)
x <- arima.sim(list(order=c(0,0,1), ma=.9), n=100)
# arima.sim: simulate ARIMA (0,0,1) time series when  $\beta = .9$ .
**note: This is an MA model so you are providing ma value.
```

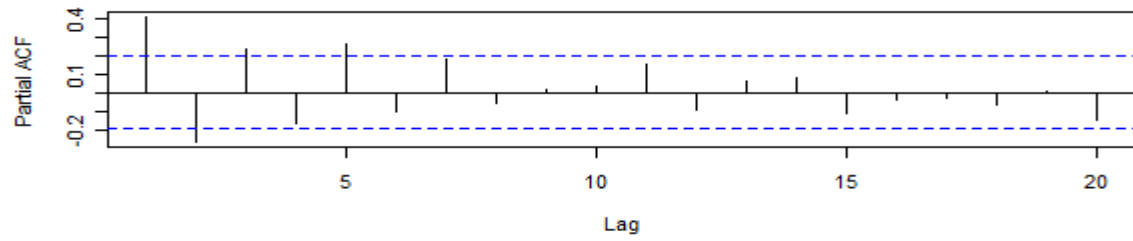
```
layout(1:3)
plot(x)
acf(x)
pacf(x)
```



Series x



Series x



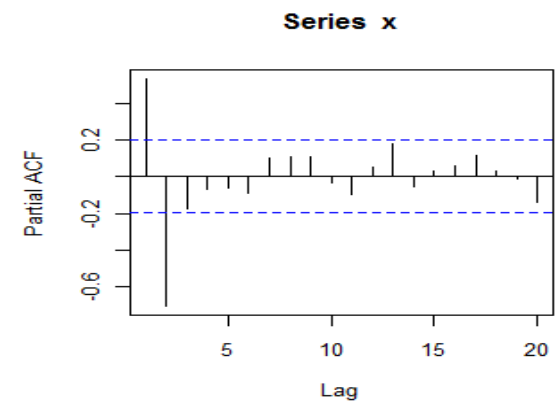
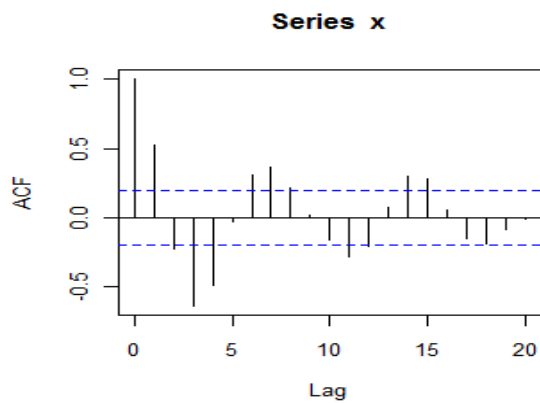
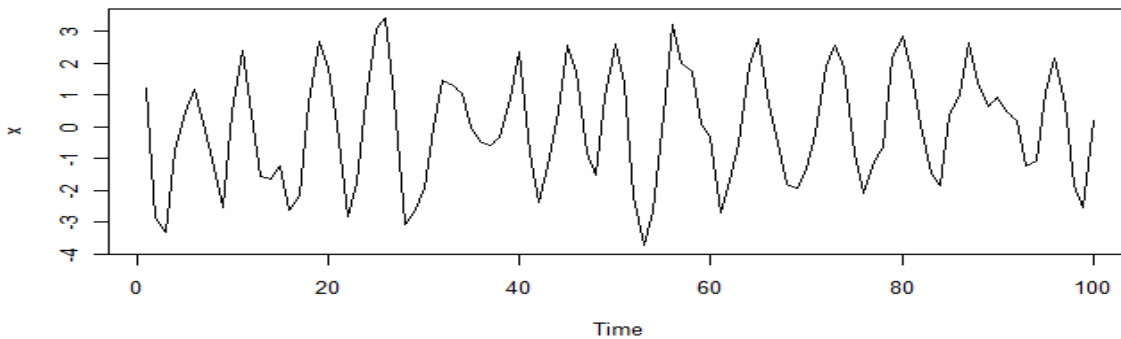
3) SARIMA

- similar to ARIMA but more advanced. Gives you better visual graphs.
- You need to download itall.R file for sarima
<http://www.stat.pitt.edu/stoffer/tsa2/Rissues.htm>

Example #1

: data file = ex1.csv

```
source(url("http://www.stat.pitt.edu/stoffer/tsa2/Rcode/itall.R"))
dl <- read.csv("K:/TEACH/DataMining_Fall2009/Data/ex1.csv",header=
FALSE)
tsdl <- ts(dl)
nf <- layout(matrix(c(1,1,2,3),2,2,byrow=TRUE), TRUE)→shows graphics
layout
layout.show(nf)
plot(tsdl)
acf(tsdl)
pacf(tsdl)
```



****note: acf dies out quickly**
pacf has spikes at lags 1 and 2, cuts off past lag 2,
acf/pacf show no seasonality
→ Therefore, this is an AR model with no seasonality: (2, 0, 0)
****note: so Try SARIMA (2,0,0)**

```
sarima(tsd1,2,0,0)
```

****note: you have to indicate time series data file**

Call:

```
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q),  
period = S),  
xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,  
REPORT = 1,  
reitol = tol))
```

Coefficients:

```
      ar1      ar2      xmean  
0.9580 -0.7628  0.0230  
s.e.  0.0663   0.0665  0.1228
```

sigma^2 estimated as 0.9513: log likelihood = -139.05, aic = 286.11

\$AIC

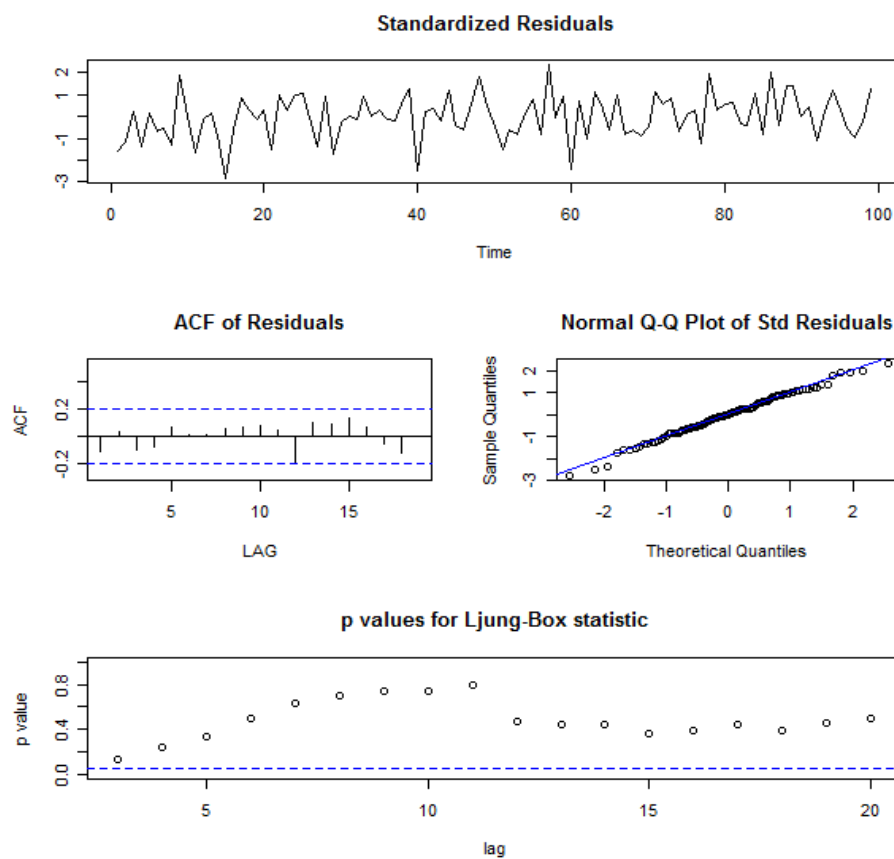
```
[1] 1.010726
```

\$AICc

```
[1] 1.035227
```

\$BIC

```
[1] 0.08936642
```



****note: What do we look for in the plot from SARIMA?**

- 1) Plot of standardized residuals show no regular pattern, is centered at 0,
- 2) acf(residuals from arima model) show no autocorrelation,
- 3) q-q plot of residuals indicate normality
- 4) Ljung-Box p-values are all > .05

Hence the ARIMA model provides good fit to the data.

****note: How do I know if I selected the best p (or q) value?**

→ aic.fit fits a series of arima models with various p values

```
aic.fit <- matrix(NA,nrow= 4,ncol = 1)
for (p in 1:4)
{a <- sarima(tsd1,p,0,0)
aic.fit[p] <- a$AIC}
print(aic.fit)
```

```
      [,1]
[1,] 1.841131
[2,] 1.019078
[3,] 1.016833
[4,] 1.035206
```

#for: fit a series of arima(p,0,0) models when p = 1, 2, 3, 4

****note: You can see [3,] is best because AIC is lowest. So you use sarima (tsd1, 3, 0,0)**

```
sarima(tsd1, 3, 0, 0)
```

Call:

```
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
Q), period = S),
xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
REPORT = 1,
reltol = tol))
```

Coefficients:

```
      ar1      ar2      ar3  xmean
      0.8437 -0.6163 -0.1541 0.0251
s.e.  0.0999  0.1171  0.1017 0.1056
```

sigma^2 estimated as 0.929: log likelihood = -137.92, aic = 285.84

\$AIC

```
[1] 1.007202
```

\$AICc

```
[1] 1.033921
```

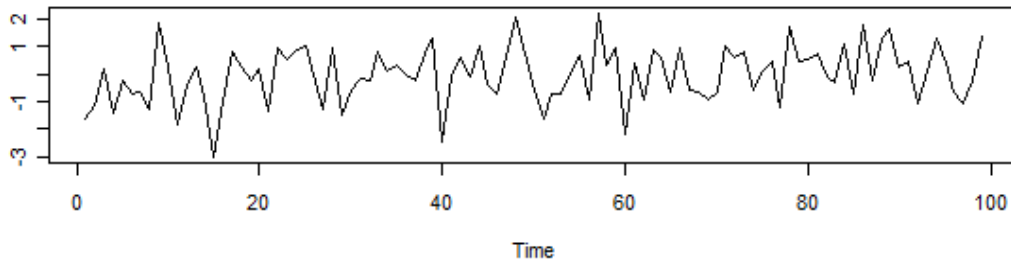
\$BIC

```
[1] 0.1120555
```

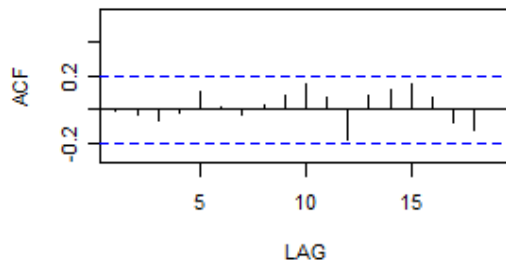
****note:**

- $ar1 \rightarrow$ significant, $0.8391/0.1004$
- $ar2 \rightarrow$ significant, $0.6311/0.1171$
- $ar3 \rightarrow$ not significant, $0.1506 / 0.1022$: s.e. is too big for $ar3$ to be significant.
- $xmean \rightarrow$ constant
 - So $ar1$ and $ar2$ is significant.
 - Therefore your equation model would be: $Y_t = .04 + .8391Y_{t-1} - .6311Y_{t-2}$
 - Overall, although $sarima(tsd1,2,0,0)$ was a good model, $sarima(tsd1,3,0,0)$ is a slightly better model.

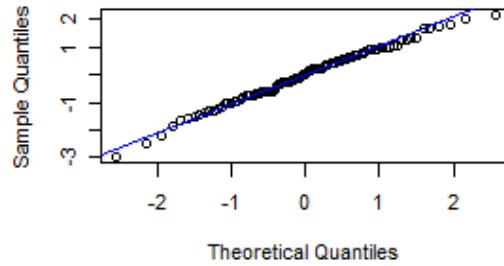
Standardized Residuals



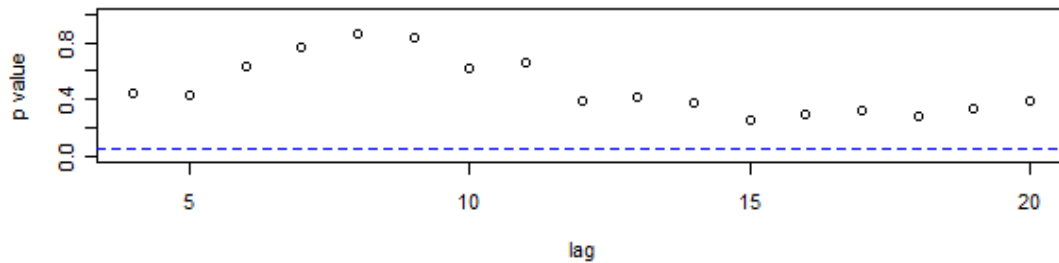
ACF of Residuals



Normal Q-Q Plot of Std Residuals

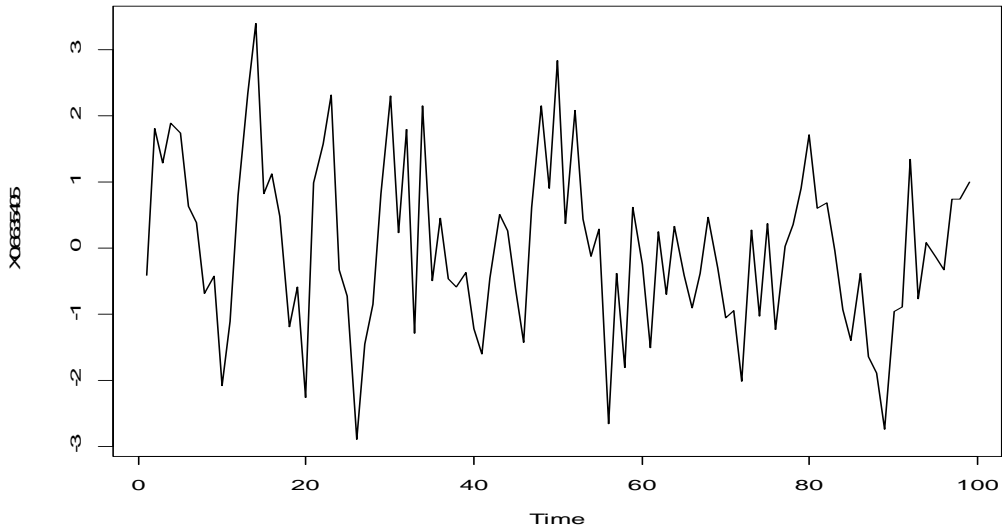


p values for Ljung-Box statistic



Example #2

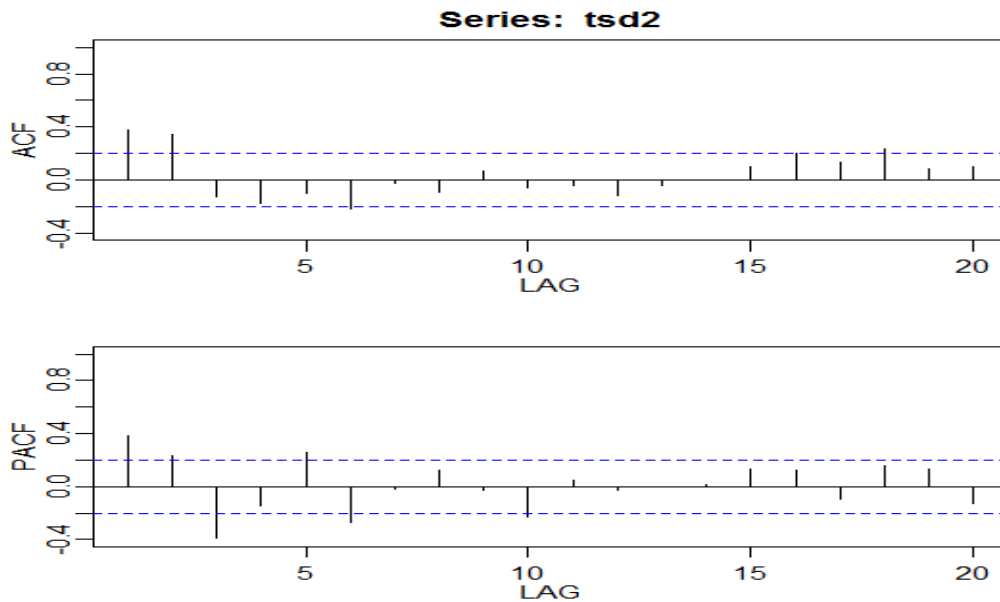
```
dd2 <- read.csv("K:/TEACH/DataMining_Fall2009/Data/ex2.csv",header=
FALSE)
tsd2 <- ts(dd2)
plot(tsd2)
```



***note: There is no trend*

```
acf2(tsd2)
```

acf2: will give you both the sample ACF and PACF of a series



***note: ACF shows few high spikes and then cuts off at lag 2, and PACF dies down
→ MA model, let's try (0,0,2)*

```
sarima(tsd2,0,0,2)
```

Call:

```
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),  
xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,  
REPORT = 1,  
reitol = tol))
```

Coefficients:

	ma1	ma2	xmean
	0.6178	1.0000	-0.0045
s.e.	0.0309	0.0466	0.2311

sigma^2 estimated as 0.7946: log likelihood = -134.38, aic = 276.76

\$AIC

[1] 0.8300965

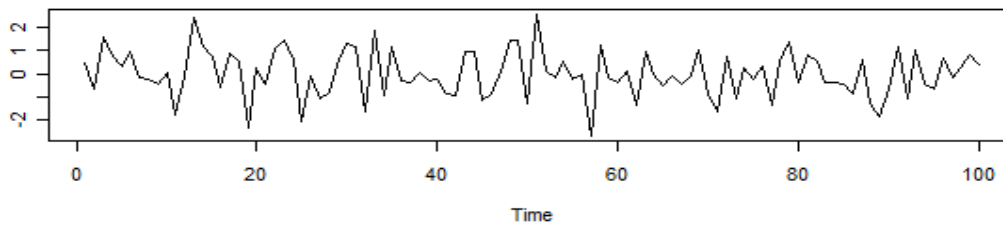
\$AICc

[1] 0.854307

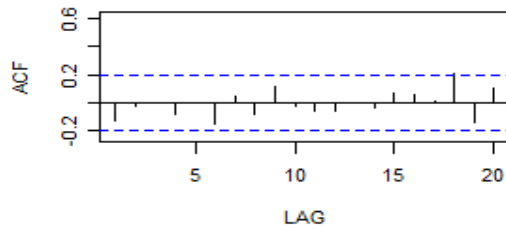
\$BIC

[1] -0.0917484

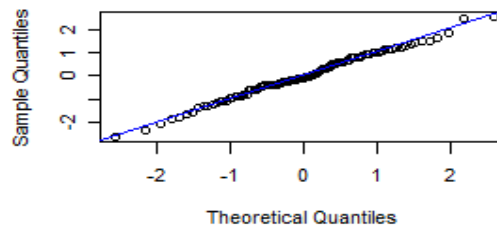
Standardized Residuals



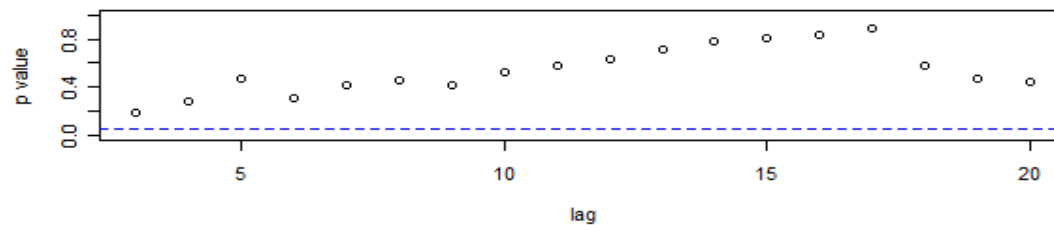
ACF of Residuals



Normal Q-Q Plot of Std Residuals



p values for Ljung-Box statistic



```

aic.fit <- matrix(NA,nrow= 4,ncol = 1)
for (q in 1:4)
{a <- sarima(tsd2,0,0,q)
aic.fit[q] <- a$AIC}
print(aic.fit)
      [,1]
[1,] 1.4215336
[2,] 0.8300965
[3,] 0.8308361
[4,] 0.8509893

```

****note:** In this case, second model is the best so stop, you already used $q=2$ for your model.

****note:** when comparing different models, choose one with smaller AIC

```

aic.fit <- matrix(NA,nrow= 5,ncol = 5)
for (p in 1:5)
for (q in 1:5)
{a <- sarima(tsd2,p-1,0,q-1)
aic.fit[p,q] <- a$AIC}
print(aic.fit)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.494082 1.421534 0.8300965 0.8308361 0.8509893
[2,] 1.356532 1.358005 0.8311777 0.8456972 0.8325783
[3,] 1.318556 1.289338 0.8512970 0.8284488 0.8482068
[4,] 1.170582 1.179390 0.8711699 0.8715350 0.8569698
[5,] 1.167200 1.101633 0.8797141 0.8844786 0.8543481

```

#ncol=5: we are looking by changing both p, q for 5 values.

Typically, 4 values for each would be enough.

****note:** Find the smallest value and run sarima again with those p and q values

This is a way to find the p and q value without looking for yourself from the ACF and PACF plots. So $p = 3$ $q = 2$

```

sarima(tsd2,3,0,2)
Call:
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
Q), period = S),
xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
REPORT = 1, reltol = tol))

```

Coefficients:

	ar1	ar2	ar3	ma1	ma2	xmean
	-0.1497	-0.0350	-0.0043	0.6285	1.000	-0.0072
s.e.	0.1039	0.1033	0.1039	0.0351	0.044	0.1937

sigma² estimated as 0.7797: log likelihood = -133.32, aic = 280.63

```

$AIC
[1] 0.8711699

```

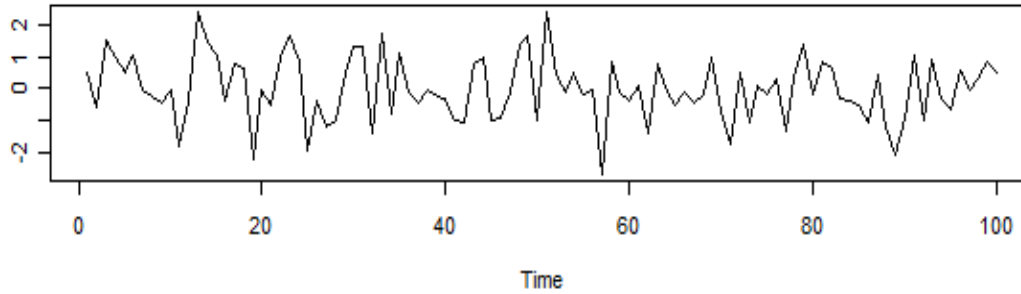
```

$AICc
[1] 0.9033438

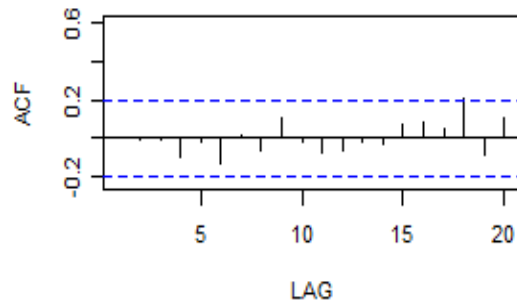
```

```
$BIC
[1] 0.02748009
```

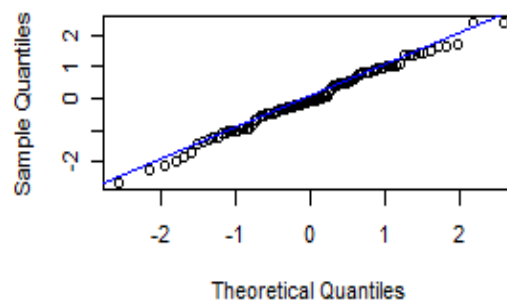
Standardized Residuals



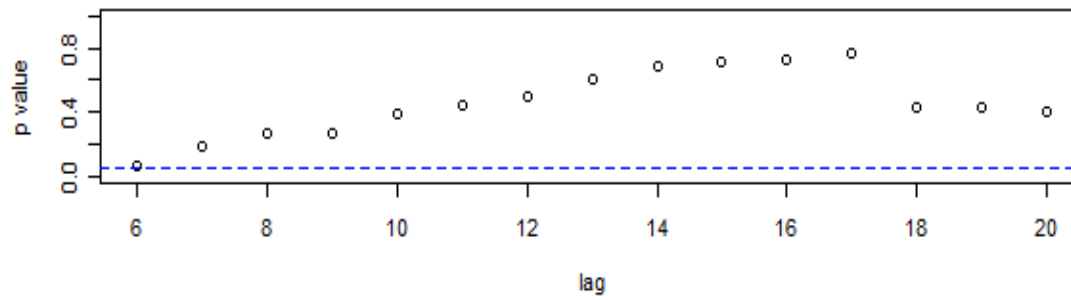
ACF of Residuals



Normal Q-Q Plot of Std Residuals



p values for Ljung-Box statistic

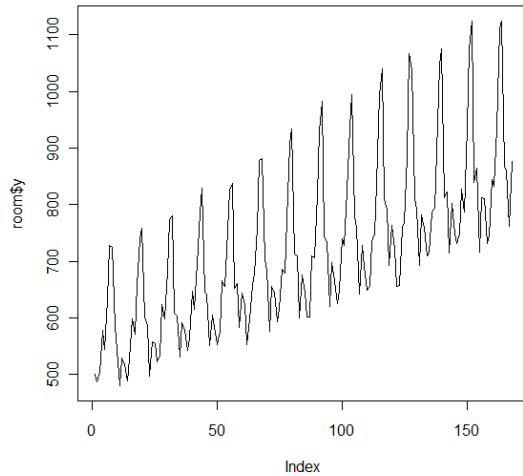


4) SEASONALITY in TIME SERIES ANALYSIS

Example #1

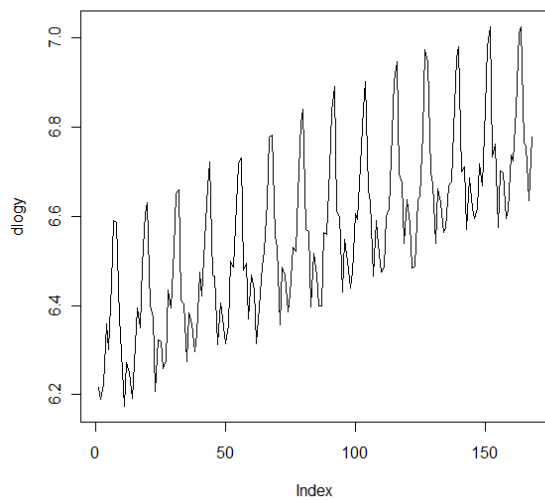
: Hotel room occupancy

```
room <- read.csv ("C:/Documents and  
Settings/User/Desktop/hotel_rooms.csv, header = T)  
plot(room$y,type="l")
```



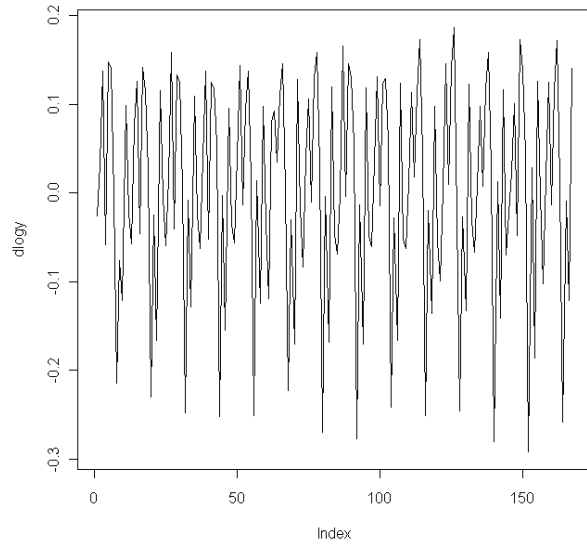
***note: Since the time series plot shows increasing variance, try log*

```
dlogy <- log(room$y)  
plot(log(room$y),type="l")
```



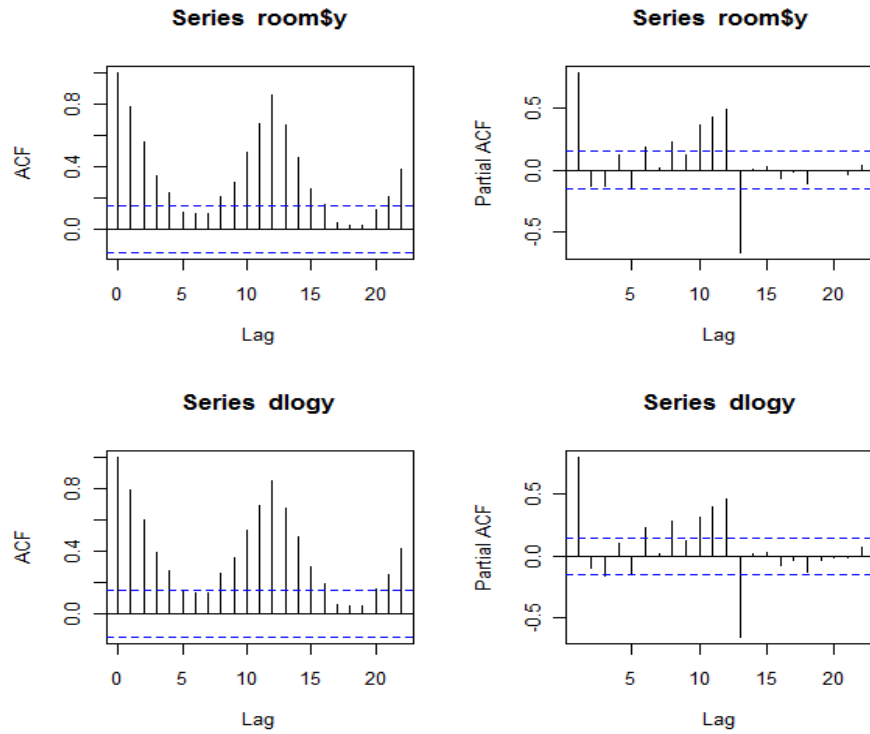
***note: Variance of dlogy appears to be constant.
Linear trend is present, so consider 1st difference.*

```
sdlogy <- diff(log(room$y))
plot(sdlogy,type="l")
```

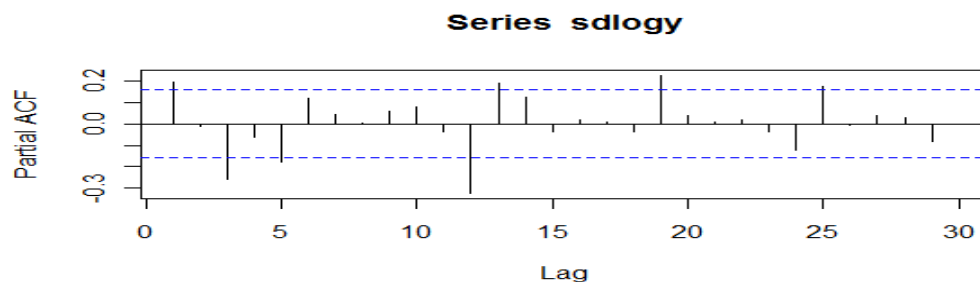
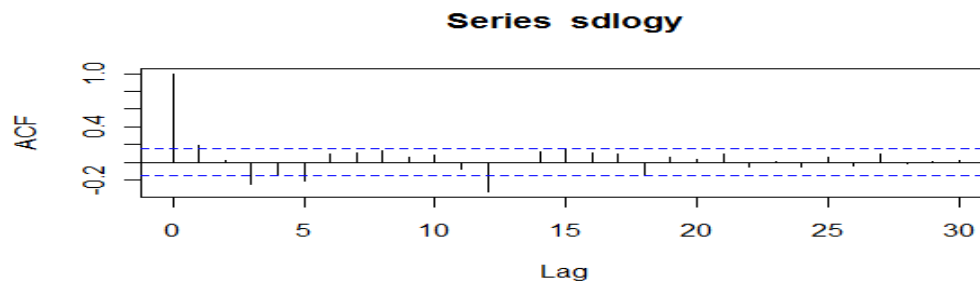


****note: Trend is completely gone now.**

```
layout(matrix(c(1,2,3,4),2,2,byrow=TRUE), TRUE)
acf(room$y)
pacf(room$y)
acf(dlogy)
pacf(dlogy)
```



```
layout(1:2)
acf(sdlogy, lag.max=30)
pacf(sdlogy, lag.max=30)
```



```
print(pacf(sdlogy, lag.max=30))
```

Partial autocorrelations of series 'sdlogy', by lag

1	2	3	4	5	6	7	8	9	10	11
-0.010	0.026	-0.218	0.001	-0.318	-0.041	-0.357	-0.189	-0.381	-0.507	-0.727
12	13	14	15	16	17	18	19	20	21	22
0.560	-0.128	-0.181	-0.015	-0.068	0.114	-0.072	0.071	-0.008	-0.025	-0.008
23	24	25	26	27	28	29	30			
0.011	0.087	-0.008	-0.069	0.135	-0.045	-0.035	0.051			

- **Identifying (p1, d1, q1)(p2, d2, q2) for SEASONALITY in SARIMA**

- Examine both ACF and PACF
 - Non-Seasonal: lags 1,2,...,9 (for monthly data)
 - Seasonal: lags 12, 24 (for monthly data)
 - Non-Seasonal → AR model
 - ACF: non-seasonal lags 1, 2, ..., 9: dies out fast at
 - PACF: non-seasonal lags 1, 2, ..., 9: spikes up to 5, cuts off past 5
 - Seasonal → MA model
 - ACF: seasonal lags 12,24: spike at lag 12, cuts off past 12
 - PACF: seasonal lags 12, 24: dies out fast

****note:** Non-Seasonal order = (5,0,0), Seasonal order = (0,1,1), let's try:

- o For sarima: `sarima(sdlog(room$y),5,0,0,0,1,1,12)`
- o For arima: `ts1 <- arima(room$y,order=c(5,0,0), seasonal = c(0,1,1))`

```
sarima(sdlog(room$y),5,0,0,0,1,1,12)
```

#12: you need 12 for indicating monthly data

```
ts1 <- arima(room$y,order=c(5,0,0), seasonal = c(0,1,1))
```

Call:

```
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),  
xreg = constant, optim.control = list(trace = trc, REPORT = 1, reltol = tol))
```

Coefficients:

	ar1	ar2	ar3	ar4	ar5	sma1	constant
	-0.5202	-0.2719	-0.3800	-0.2784	-0.2998	-0.5854	0e+00
s.e.	0.0788	0.0906	0.0863	0.0867	0.0797	0.0832	1e-04

sigma² estimated as 0.0004230: log likelihood = 379.22, aic = -742.43

\$AIC

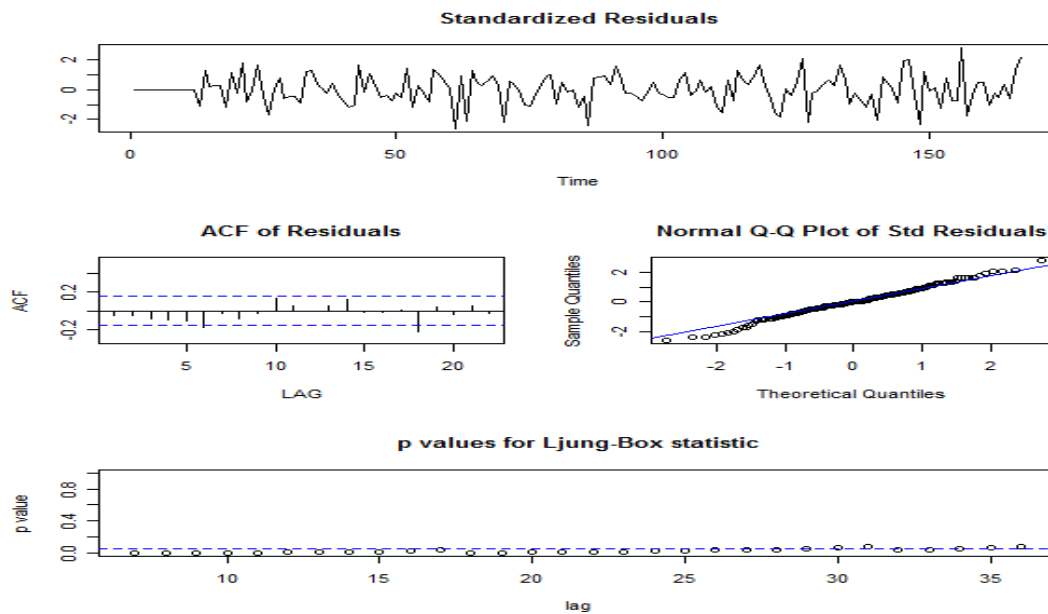
[1] -6.684409

\$AICc

[1] -6.666975

\$BIC

[1] -7.553714

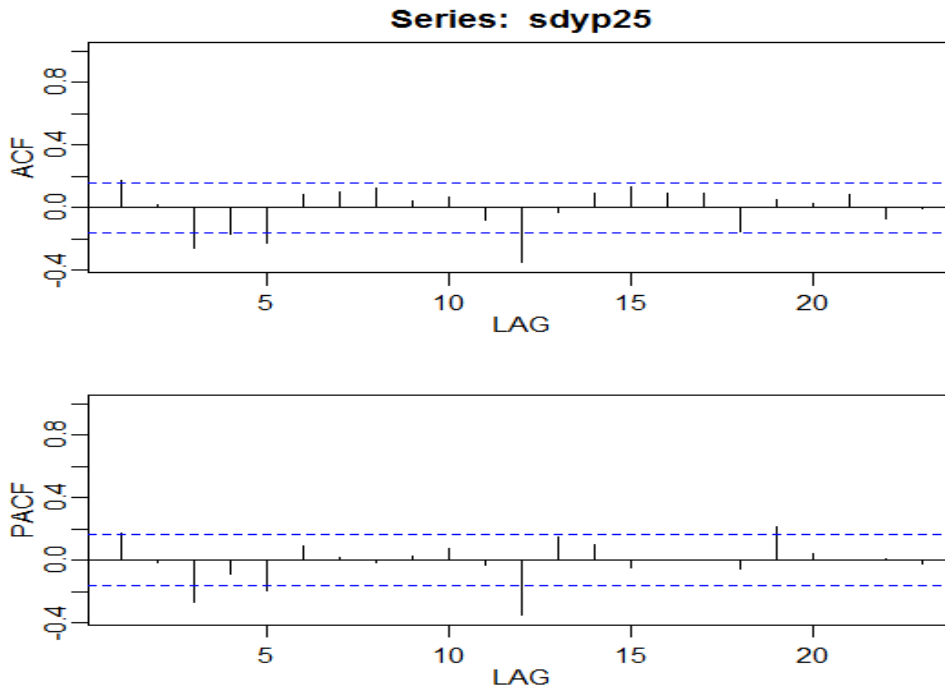


****note:** NOT GOOD! Normality and p values plot both problematic!!! → So let's try a different transformation

```

yp25 <- (room$y)**(1/4)
plot(yp25, type = "l")
sdyp25<-diff(yp25, lag =12)
plot(sdyp25)
plot(sdyp25, type = "l")
acf2(sdyp25)

```



```

sarima(yp25,5,0,0,0,1,1,12)

```

Call:

```

arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q),
period = S),
xreg = constant, optim.control = list(trace = trc, REPORT = 1, reltol =
tol))

```

Coefficients:

	ar1	ar2	ar3	ar4	ar5	sma1	constant
	0.2307	0.108	-0.2375	-0.0074	-0.1373	-0.5467	0.0035
s.e.	0.0831	0.084	0.0809	0.0826	0.0818	0.0841	0.0001

sigma² estimated as 0.0006032: log likelihood = 354.56, aic = -693.12

\$AIC

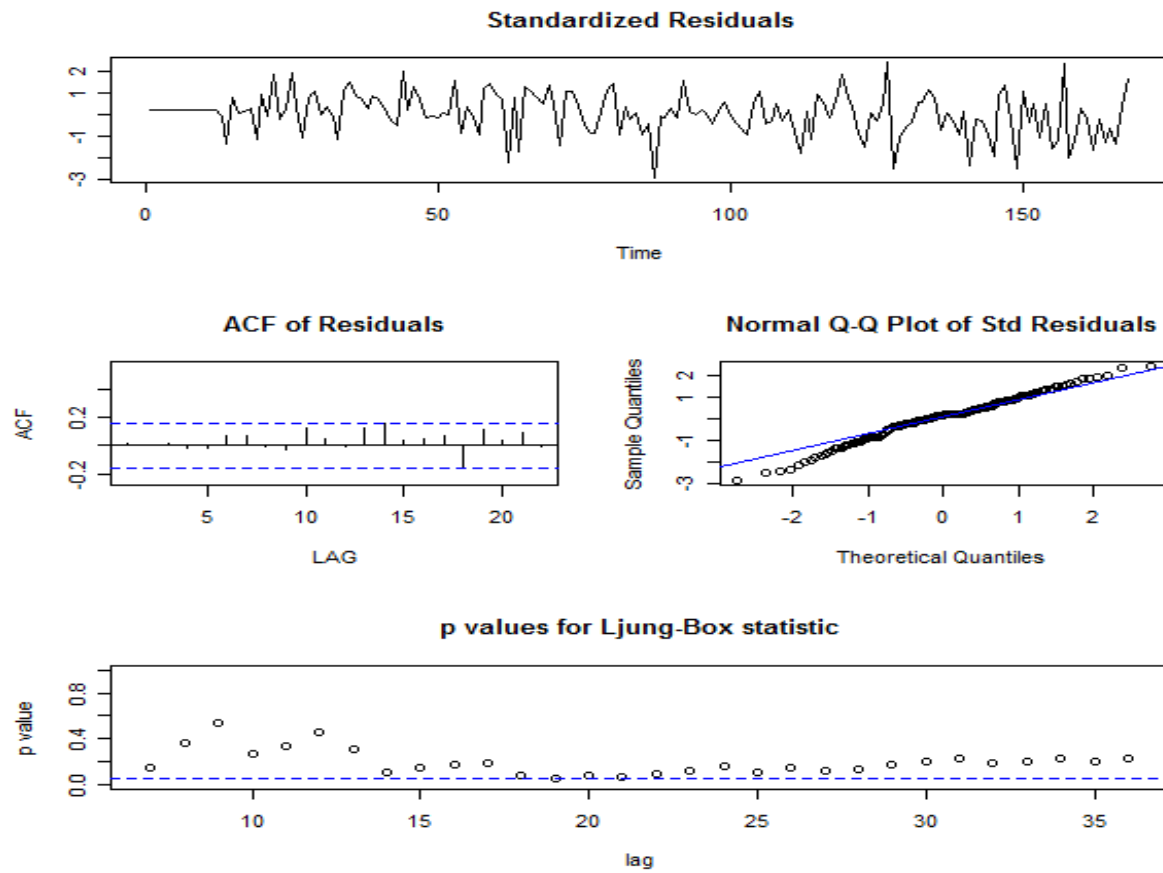
[1] -6.329956

\$AICc

[1] -6.31266

\$BIC

[1] -7.199791



***note: Not perfect, but Better now!*