


Seok-Oh Jeong, In Heok Lee, and Jay W. Rojewski  
University of Georgia


# The R Workshop

Applying the Integrated Suite of Software  
Facilities for Statistical Computing and Graphics




University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office

January 23-January 24, 2012



January 23-January 24, 2012


# 1. Introduction



University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office

## R is...


- A programming language and an environment for data manipulation, (statistical) computing, and graphical display.
- Powerful, but **FREE!**



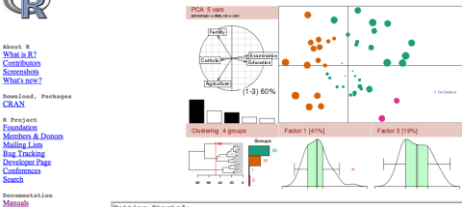
## Introduction

## Download and Installation

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



The R Project for Statistical Computing



Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN [mirror](#).
- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

## Download and Installation

1. Click on “CRAN”.
2. Select a mirror site near you.
3. Click on “Download and install R”.
4. Click on “base”.
5. Download the installation file and run it!



Introduction

## Console

- Run R, then a R-GUI window will appear.
- In R-GUI window, you'll see another window called 'R console'.
- Command prompt:

```
> 3+2
[1] 5
> pi
[1] 3.141693
```



Introduction

## Working Directory

- *Working directory* is the default location for all file input and output.
- Use `getwd()` to report the current working directory, and use `setwd()` to change your working directory.
 

```
> getwd()
> setwd("c:/users/mywork")
```
- Or, from the main menu, select "File" → "Change dir..."



Introduction

## Help

- Need a help for `persp()`? Just type in the command prompt
 

```
> ? persp
or
> help(persp)
```
- Need an extended help? Type
 

```
> ?? log
or
> help.search("log")
```
- Online documentation: Visit R-project website and click on "Manuals".



Introduction

## Package(s)

- All R functions and datasets are stored in packages
- Installation of a package  

```
> install.packages("package name")
```
- Loading a package  

```
> library("package name")
```
- Unloading a package  

```
> detach("package name")
```



Introduction

## R command

- For variable names, we may use alphabets, numbers, period(.), underscore(\_), etc.
- For assignment, <- is used. You may use =, but not preferable
- All names should begin with alphabet or period(.)
- Semicolon(;) separates multiple commands.  

```
> beta.0 <- 3 ; beta.1 <- 2
```
- Comments begin with #  

```
> rnorm(100) # to generates 100 random numbers
```



Introduction

## R command

- Use arrow keys for recalling former commands.
- Type the name of a variable to print its value onto console.

```
> beta.0
[1] 3
> beta.0 + 1
[1] 4
```



Introduction

## R command

- Case-sensitive  

```
> a <- 1
> A <- 2
> a==A
[1] FALSE
```
- The objects are stored in R's database  


```
> ls() # list the objects stored in database
```
- Run the script files.  

```
> source("sample.R")
```




Introduction

January 23-January 24, 2012



## 2. Data Manipulation



University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office


## Data Types

- **Vector**

```
x <- c(1,2,3,4,5)
y <- c("Clarke", "Oconee", "Barrow")
x[3]
```
- **Array**

```
z <- array(1:20, dim=c(4,5))
z[4,5]
```
- **Matrix**

```
z <- matrix(1:20, 4, 5)
A <- matrix(2, 4, 5)
z[3,2]
```



Data Manipulation


## Data Types

- **List**

```
> Jeong <- list(first.name="Seok-Oh",
age=40, citizenship="South Korea")
```
- **Data frame**

```
> X <- c("Jeong", "Lee", "Rojewski")
> Y <- c(40, 35, 53)
> data <- data.frame(last.names=X,
age=Y)
```
- **Factor**

```
> z <- c("LD","BD", "BD", "LD", "Non",
"Non", "Non", "Non")
> z <- factor(z)
> levels(z)
```




Data Manipulation

## Vectors

- **Concatenation**

```
> a <- c(2,2,2,2,2,2)
> a <- c(1, 2, 3); b <- c(5, 6)
> x <- c(a, 4, b) # x <- c(1,2,3,4,5,6)
```
- **Sequence**

```
> x <- seq(from=0, to=1, by=0.1)
> y <- seq(from=0, to=1, length=11)
> z <- 1:10
> rep(1, 10)
```



Data Manipulation

## Vectors

- Arithmetic: Componentwise

```
> x <- 1:3; y <- c(2,2,2)
> x+y
> x-y
> x*y
> x/y
> x^y
> z <- rep(2, 5)
> x+z
> y-3
```



Data Manipulation

## Vectors

- Mathematical functions

```
> x <- 1:10      > log(x)
> exp(x)         > sin(x)  # cos(x), tan(x)
> abs(x)         > sqrt(x)
> sort(x)        > length(x)
> min(x)         > max(x)
> mean(x)        > sum(x)
> prod(x)
```



Data Manipulation

## Vectors

- Logical vectors

```
> x <- 1:10; y <- rep(5, 10)
> z <- x<5      # less than
> sum(z)
> x<=5          # less than or equal to
> x==5          # equal
> x!=5          # not equal
> (x>5)&(y<2)    # and
> (x<5)|(y<2)   # or
```

- Missing values

```
> x <- c(1, 2, 3, NA, 5)
> is.na(x)
```



Data Manipulation

## Vectors

- Index vectors

```
> x <- -10:10
> x[3]
> x[1:3]
> x[c(1,3,5)]
> y <- x[x<0]
> x[x<0] <- -x[x<0]
> x <- c(1, 2, 3, NA, 5)
> x[!is.na(x)]
> x[is.na(x)] <- 4
> fruit <- c(5, 3, 2)
> names(fruit) <- c("apple", "orange", "peach")
> fruit[c("apple", "peach")]
```



Data Manipulation

## Arrays and Matrices

- To generate an array and a matrix

```
> z <- array(1:20, dim=c(4,5))
> A <- matrix(1:20, 4, 5)
> B <- matrix(2, 4, 5)
> z[3,4]
> A[3,4]
> x <- c(1,2,3)
> y <- c(4,5,6)
> cbind(x, y)
> rbind(x, y)
> cbind(B, 1:4)
> C <- cbind(A, B)
```



Data Manipulation

## Arrays and Matrices

- Arithmetic: Componentwise

```
> A <- matrix(1:20, 4, 5)
> B <- matrix(1:20, 4, 5)
> A+B
> A-B
> A*B
> A/B
```

- Arithmetic: Matrix multiplication, inverse

```
> A <- matrix(runif(20), 4, 5)
> B <- A%*%t(A)      # t(): transpose
> solve(B)           # inverse
```



Data Manipulation

## Lists

- A list is an object consisting of a collection of objects called components.

```
> Jeong <- list(first.name="Seok-Oh", age=40,
  married=T, no.children=2, child.ages=c(9, 6))
> Jeong$age
> Jeong[[1]]
> Jeong$child.ages
> Jeong[[5]][1]
```



Data Manipulation

## Factors

- A factor is a vector object used to specify a discrete classification (grouping) of the components of other vectors of the same length.

```
> x <- c(80, 90, 85, 85, 50, 60, 45, 50)
> z <- c("LD", "BD", "BD", "LD", "Non", "Non",
  "Non", "Non")
> z <- factor(z)
> levels(z)
> x.means <- tapply(x, z, mean)
```



Data Manipulation

## Data frames

- A data frame is a list with restrictions:
  - a. The components must be vectors (numeric, character, or logical), factors, numeric matrices, lists, or other data frames.
  - b. Numeric vectors, logicals and factors are included as is, and character vectors are coerced to be factors, whose levels are the unique values appearing in the vector.
  - c. Vector structures appearing as variables of the data frame must all have the same length, and matrix structures must all have the same row size.

Data Manipulation



January 23-January 24, 2012



## 3. Data Import/Export

University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office



## Read Data

- From the console: `scan()`

```
> x <- scan()
1: 1
2: 2
3: 3
4:
Read 3 items
> x
[1] 1 2 3
```



Data Import/Export

## Read Data

- From a file: `scan()`

```
> x <- scan(file="c:/mydata/data_x.txt")
> y <- matrix(scan("c:/mydata/data_y.txt"),
ncol=3, byrow=T)
```
- From a file: `read.table()`

```
> x <- read.table(file="table.txt", header=T,
sep=" ")
```
- From a file: `read.csv()`

```
> x <- read.csv(file="table.csv", header=T)
```



Data Import/Export

## Read Data

- Accessing built-in datasets: `data()`  

```
> library("MatchIt")
> data("lalonde")
```
- Want the list of built-in datasets contained in the currently loaded packages? Just type `data()` .  

```
> data()
```



Data Import/Export

## Export Data

- To the console: `print()`  

```
> x <- scan()
1: 1
2: 2
3: 3
4:
Read 3 items
> print(x)
[1] 1 2 3
```



Data Import/Export

## Export Data

- To a file: `write()`  

```
> x <- seq(from=0, to=1, by=0.1)
> write(x, file="output.txt")
```
- To a file: `write.table()`  

```
> x <- matrix(1:20, 4, 5)
> write.table(x, file="table.txt")
```



Data Import/Export

January 23-January 24, 2012



## 4. Graphics: Visualization of Data

University of Georgia  
 Department of Workforce Education, Leadership, and Social Foundations  
 College of Education Research Office





## Distribution of One-Dimensional Data

- Qualitative data
  - Bar chart: `barplot()`
  - Pie chart: `pie()`
- Quantitative data
  - Stem-and-leaf plot: `stem()`
  - Histogram: `hist()`
  - Boxplot: `boxplot()`



Graphics

### ## Beer Preference Example

```
beer <- c(3, 4, 1, 1, 3, 4, 3, 3, 1, 3, 2, 1, 2, 1, 2, 3, 2, 3, 1,
1, 1, 1, 4, 3, 1)
# (1) Domestic can (2) Domestic bottle,
# (3) Microbrew (4) Import

barplot(table(beer))
barplot(table(beer)/length(beer),
col=c("lightblue", "mistyrose", "lightcyan", "cornsilk"),
names.arg=c("Domestic can", "Domestic bottle", "Microbrew",
"Import"),
ylab="Relative frequency", main="Beer Preference Survey")

beer.counts <- table(beer) # store the table result
pie(beer.counts) # first pie -- kind of dull
names(beer.counts) <- c("Domestic\n can", "Domestic\n bottle",
"Microbrew", "Import") # give names
pie(beer.counts) # prints out names
```



Graphics

```
## Stem-and-leaf
scores <- c(2, 3, 16, 23, 14, 12, 4, 13, 2, 0, 0, 0,
6, 28, 31, 14, 4, 8, 2, 5)
stem(scores)

## histogram
x <- rnorm(1000) # To generate 1,000 random numbers from N(0,1)
hist(x, xlab="data")
hist(x, probability=T, xlab="data")
z <- seq(from=-3, to=3, by=0.01)
lines(z, dnorm(z), col=2)

## Boxplot
growth <- c(75,72,73,61,67,64,62,63) # the size of flies
sugar <- c("C","C","C","F","F","F","S","S") # diet
fly <- list(growth=growth, sugar=sugar)
boxplot(fly$growth)
jpeg(file="flygrowth.jpg", width=480, height=360)
```



Graphics

## Distribution of Multi-Dimensional Data

- Categorical and Quantitative Data
  - Boxplot: `boxplot()`
- Qualitative and Quantitative Data
  - Scatterplot: `plot()`



Graphics

```
## Boxplot
boxplot(growth~sugar, xlab="Sugar Type", ylab="Growth",
        main="Growth against sugar types", data=fly)

## Scatterplot
plot(cars$speed, cars$dist)
# the speed of cars and the distances taken to stop
attach(cars)
plot(speed, dist, col="blue", pch="+",
      ylab="Distance taken to stop", xlab="Speed",
      ylim=c(-20, 140))
lm(dist~speed)
abline(-17.579, 3.932, col="red")
title(main="Scatterplot with best fit line", font.main=4)
```

Graphics



```
## Scatterplot matrix
attach(iris)
pairs(iris[,1:4])
pairs(iris[Species=="virginica", 1:4])

## 2D Histogram
library(hexbin)
plot(hexbin(iris[,3], iris[,4]),
     xlab="Petal Length", ylab="Petal Width")
```

Graphics



January 23-January 24, 2012



## 5. Advanced Programming

University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office



## Conditional Execution

- A conditional statement by 'if-else'
 

```
> if (x<3) print("x<3") else print("x>4")
```
- Commands can be grouped by braces
 

```
> x <- 4
> if ( x < 3 ) {print("x<3"); z <- "M"} else
{print("x>3"); z <- "F"}
```

Advanced Programming



## Iteration, loop

- Loop: A repeatedly executed instruction cycle
- for-loop: loop over all elements in a vector

```
x <- 1:10
n <- length(x)
y <- rep(0, n)
for ( i in 1:n ) {
  y[i] <- x[i]^2
}
z <- x^2
print(cbind(y, z))
```



Advanced Programming

## Iteration, loop

- while-loop: for which we don't know in advance how many iterations there will be

```
n <- 0
sum.so.far <- 0
while ( sum.so.far <= 1000 ) {
  n <- n+1
  sum.so.far <- sum.so.far + n
}
print(c(n, sum.so.far))
sum(1:45)
```

✓ Whenever possible, try to avoid loops!



Advanced Programming

## Applying a function to every row/column

- To apply a function to every row[or column], use `apply()`.

```
> A <- matrix(1:20, 4, 5)
> apply(A, 1, sum) # to every row
> apply(A, 2, sum) # to every column
```



Advanced Programming

## Writing New Functions

```
my.stat <- function(x)
{
  m <- mean(x); s <- sd(x)
  res <- list(x=x, m=m, s=s)


  par(mfrow=c(1,2))
  boxplot(x, main="Boxplot")
  hist(x, prob=T, col="lightgray", main="Histogram",
       xlab="data")
  z <- seq(from=min(x), to=max(x), by=0.01)
  lines(z, dnorm(z, mean=3, sd=1), col=2, lwd=3, lty=2)
  return(res)
}

X <- rnorm(1000, mean=3, sd=1)
my.stat(x=X)
```




Advanced Programming

January 23-January 24, 2012



## 6. Summarizing Data



University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office

0.7160656 -0.3797124 0.7115967 -0.1889281 -0.7032382 -1.1096595 -0.3363823 0.8176170 0.1913517 -0.2098376 [251]  
08228 -1.2692307 1.2485665 -1.0977928 0.6821302 -0.6476787 1.7315770 -1.6743805 0.9240398 0.1716021 [261] 0.8515  
71520 0.8063297 -0.1635424 0.4145850 0.6619074 -0.4082618 1.0023536 0.2962587 -0.0455598 [271] -0.3377601 0.0092  
99061 0.0804276 -1.1327947 -0.4724080 -0.5483509 1.4254622 1.0142226 -0.7548832 [281] -0.4243025 0.2201248 0.0275  
75315 -1.4021574 -1.7079080 1.2708254 0.5252110 0.0944564 -0.1857770 [291] 0.8227716 -0.0343559 0.7184933 -0.3507  
53377 0.3668166 0.7319903 -0.7136734 0.7931802 -1.5419692 [301] -0.3275177 -1.6955021 -0.7664577 2.3103283 -0.537  
10885 0.6676794 0.8812027 -0.8826746 -0.0169117 [311] 0.6240759 -0.0820713 0.1483842 -0.1427901 -0.4799147 -0.458  
58435 0.2093504 0.0814105 1.8690609 [321] 0.4352119 1.9909011 -0.0255430 -0.7666637 -1.4900247 -1.7458375 -1.035  
58887 -0.6450265 -0.6060436 [331] -0.8159328 -0.2177304 0.3557860 -2.3815800 0.4280608 -2.2567250 -1.7111780 1.732  
78034 1.3770028 [341] 0.2271645 0.0190696 -0.6397869 0.4446839 1.2774298 -0.3691963 -0.8395816 -0.3240304 0.1107  
88310 [351] 0.7385059 -0.8239616 -0.6010957 0.6806157 0.0063708 1.0484444 -0.5214221 -0.8688074 -0.6692890 -0.552  
-0.6588511 -0.8981495 -1.0673545 0.7994342 -0.0669976 -1.3183245 0.3702838 -0.4788287 1.6122158 0.9691895 [371]  
25325 0.6018153 -1.0684628 0.7096711 -1.3780736 0.9156478 1.1259662 -0.9285121 0.1200722 1.5613053 [381] -0.426  
59612 1.1097167 -1.3435353 -0.0180177 0.8386996 -1.0564775 -0.9475907 1.8754802 0.1076621 [391] 1.8070982 1.036  
43806 1.0370373 -0.3677119 0.8464400 0.0340448 -0.055512 -0.5124115 -1.7289937 [401] 0.3550545 -0.46579 0.557  
23223 0.4391668 -0.6706706 -0.3411021 -0.0210135 -0.3515567 -0.0000000 -0.0000000 -0.0000000 -0.0000000 -0.0000000  
55672 -1.026016 -0.106022 -0.364053 -0.3442236 -0.3618187 -0.0000000 -0.0000000 -0.0000000 -0.0000000 -0.0000000  
01147 -0.3688195 1.3783246 0.0001032 -0.9809956 [431] -3.369674 0.3306838 0.0666615 -0.2387170 1.0455666 0.1852  
88038 -1.2481095 -1.4009616 -1.3086729 [441] 0.0898067 -0.9238770 -0.7754579 -0.3700475 1.1343146 0.6966690 1.538  
7596 -0.6098369 -0.8944564 [451] 0.5413484 -0.8647763 -0.9467811 0.8160911 0.9284482 1.3716441 -0.4872980 -1.136  
57635 1.4448767 [461] 1.3493079 -1.4228588 0.4786978 0.3040695 -0.4150574 0.0462064 0.6397801 1.2555683 1.6794  
88545 [471] 1.2078043 -1.3175601 -0.4902107 0.4611098 -2.3246298 -0.9655982 -0.8413548 -0.2264772 -0.1159337 -1.111  
-2.1160452 0.9110626 -0.5646935 -0.6506525 0.5676647 -0.2525807 -0.8341889 -1.1063016 -0.8068980 0.7144335 [491]  
24222 0.0070713 0.9334361 0.5080611 -0.5291276 0.3860185 0.2551496 -1.5254860 0.5882182 -0.9391098 [501] -1.5763  
55546 -0.0850655 0.1205662 0.0492091 1.6740339 -1.3920717 1.8455457 0.8182496 1.0671930 [511] -0.0898167 1.0322  
82744 -0.2253102 0.6124574 -0.0553955 1.0423951 0.2987654 0.4683266 1.0143614 [521] 0.3182044 1.4629167 2.3547  
3721 0.9055108 -1.3635307 0.9651887 1.0087722 -0.0001732 0.0485673 [531] 0.4870944 0.1478238 -0.5577689 0.819  
10771 0.6207478 -0.1193080 -0.8003797 0.1568611 -0.2461361 [541] -1.7861941 0.9169431 0.1155164 1.2118375 -0.500  
7990 1.3077573 -0.9011344 2.6798482 0.1874225 [551] -0.9764731 0.0633646 -0.2033034 0.9134958 -1.4379754 0.428  
16124 1.1363646 0.8937146 0.2342962 [561] -0.0825818 -0.0530010 0.0315594 -0.0683545 0.8379897 0.1801809 -0.7983  
73510 0.0617739 1.0826551 [571] 0.9841809 -1.2388596 0.8687167 -1.5211849 0.2750131 -0.4145926 -0.7230157 -1.4566  
02814 -0.5175049 [581] 1.7852190 -0.9848091 0.7949877 0.3953470 -1.2640616 -0.1460294 -0.4098662 0.0798892 -0.427

## What or How to Summarize Your Data

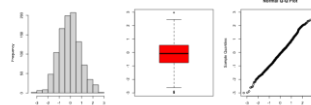
- It depends on your research question(s)
- Do not make blind ventures with your data
- Avoid overcooking
- Use your intelligence and common sense



Summarizing Data

## Distribution!

- Distributional information would be *be-all* and *end-all*
- Statistics such as mean, standard deviation, and regression coefficients stand for some partial (but important) features of the distribution
- Visualization is always the first step of your analysis



Summarizing Data

## Summary Statistics

- **Location** (of the distribution):  
mean, median, trimmed mean...
- **Scale** (of the distribution):  
standard deviation, variance, range, IQR...
- **Shape** (of the distribution):  
skewness, kurtosis...
- **Relative location** (in the distribution):  
quantile, percentile, minimum, maximum...



Summarizing Data

```
library(MatchIt)
data(lalonde)
attach(lalonde)

boxplot(re78~treat)

re78.treat <- re78[treat==1]
re78.contr <- re78[treat==0]

mean(re78); sd(re78)
mean(re78.treat); sd(re78.treat)
mean(re78.contr); sd(re78.contr)

summary(lalonde)
```



Summarizing Data

```
summary.stats <- function(y)
{
  x <- na.omit(y)      # Omit missing values
  m <- mean(x)
  s <- sd(x)
  z <- (x-m)/s         # Standardization
  skew <- mean(z^3)    # Skewness
  kurt <- mean(z^4)    # Kurtosis
  mini <- min(x)       # Minimum
  maxi <- max(x)       # Maximum
  q <- quantile(x, probs=c(.25, .50, .75))
  res <- list(average=m, stdev=s,
              skewness=skew, kurtosis=kurt,
              q1=q[1], q2=q[2], q3=q[3],
              minimum=mini, maximum=maxi)

  return(res)
}

summary.stats(re78.treat)
summary.stats(re78.contr)
```




Summarizing Data

```
table(treat, black)
table(treat, hisp)
table(treat, married)
table(treat, nodegr)
par(mfrow=c(1,2))
boxplot(age~treat, names=c("Control", "Treatment"), ylab="age")
boxplot(educ~treat, names=c("Control", "Treatment"),
        ylab="educ")
```




Summarizing Data

January 23-January 24, 2012




## 7. One- and Two-Sample Tests



University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office

## One-sample t-test


- Given a sample from a population, want to know if the population mean could be a particular value  $\mu_0$ ?
  1. Compute the sample mean.
  2. Compute the normalized difference (t-statistic) between the sample mean and the hypothesized mean  $\mu_0$ .
  3. Compare it with the reference distribution (t-distribution).



One- & Two-Sample Tests

## One-sample t-test

```
# Daily energy intake in kJ for 11 women
daily.intake <- c(5260, 5470, 5640, 6180, 6390, 6515,
  6805, 7515, 7515, 8230, 8770)
# To investigate whether women's intake deviates
# from a recommended value of 7725kJ
mean(daily.intake)
sd(daily.intake)
boxplot(daily.intake); abline(h=7725, col=2, lty=2)
t.test(daily.intake, mu=7725, alternative="less")
# alternative = "greater", "two.sided"
```




One- & Two-Sample Tests

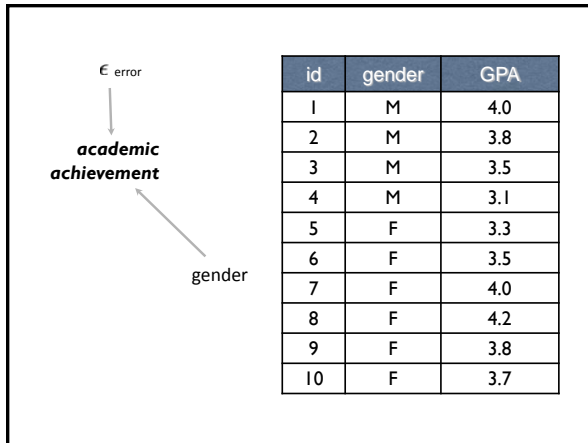
## One-sample Wilcoxon Signed-Rank Test

- Assume only that the data distribution is symmetric around the theoretical mean.
  1. Subtract the theoretical mean and rank the differences.
  2. Ignoring the sign, calculate the sum of positive or negative ranks.

```
wilcox.test(daily.intake, mu=7725)
```



One- & Two- Sample Tests



## Two-sample t-test

- Given one sample each from two population, want to know if the population means could differ each other?
  1. Compute the sample means.
  2. Compute the normalized difference (t-statistic) between the sample.
  3. Compare it with the reference distribution (t-distribution).

One- & Two-Sample Tests



## Two-sample Tests

```
group <- c(rep("M", 4), rep("F", 6))
y <- c(4.0, 3.8, 3.5, 3.1, 3.3, 3.5, 4.0, 4.2, 3.8, 3.7)
boxplot(y~group)

# Two-sample t-test
t.test(y~group)

# Wilcoxon rank-sum test
wilcox.test(y~group)
```



One- & Two-Sample Tests


## Paired Tests

```
library(ISwR)
attach(intake)
intake
t.test(pre, post, paired=T)
t.test(pre, post) # WRONG!!!
wilcox.test(pre, post, paired=T)
# Or...
diff <- post-pre
t.test(diff, mu=0)
wilcox.test(diff, mu=0)
```




One- & Two-Sample Tests

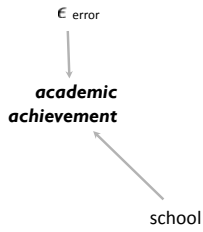
January 23-January 24, 2012




## 8. Analysis of Variance (ANOVA)



University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office

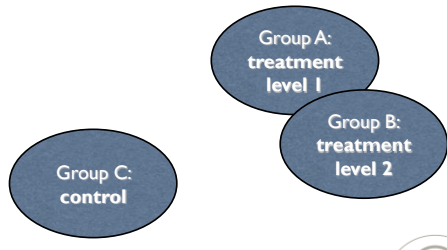



id	school	GPA
1	C	4.0
2	B	3.8
3	B	3.5
4	B	3.1
5	A	3.3
6	A	3.5
7	C	4.0
8	C	4.2
9	A	3.8
10	C	3.7



ANOVA

### Compare Means





ANOVA

One-way ANOVA  
Two-way ANOVA  
Multi-way ANOVA

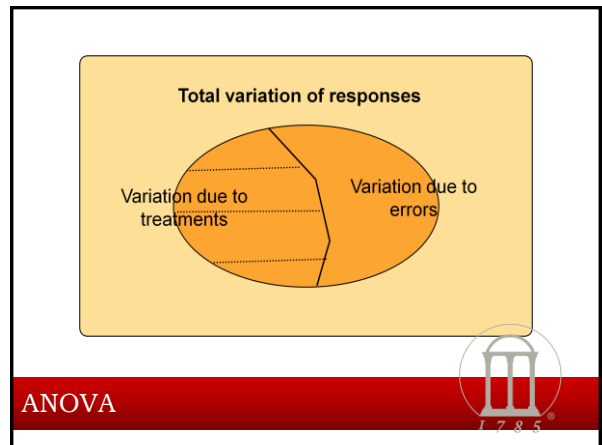
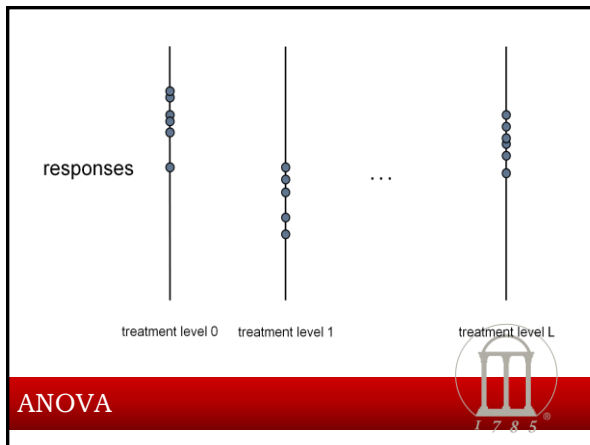
Treatment level

Treatment effect  
main effect  
interaction



ANOVA





```
school <- c(3, 2, 2, 2, 1, 1, 3, 3, 1, 3)
school <- factor(school, labels=c("A", "B", "C"))
y <- c(4.0, 3.8, 3.5, 3.1, 3.3, 3.5, 4.0, 4.2, 3.8, 3.7)
boxplot(y~school)

# ANOVA
res <- lm(y~school)
anova(res)
summary(res)
pairwise.t.test(y, school, p.adj="bonferroni")
res <- aov(y~school)
TukeyHSD(res)

# Kruskal-Wallis test
kruskal.test(y~school)
```

**ANOVA**


```
# Two-way ANOVA
```

```
# Heart rate after administration of enalaprilate
library(ISwR)
attach(heart.rate)
heart.rate
anova(lm(hr~subj+time))
```


```
# Nonparametric counterpart
friedman.test(hr~time|subj)
```

**ANOVA**


January 23-January 24, 2012




## 9. Regression



University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office



id	SAT	GEN	GPA
1	388	F	4.0
2	354	F	3.8
3	361	F	3.5
4	329	F	3.1
5	331	M	3.3
6	364	M	3.5
7	399	M	4.0
8	421	F	4.2
9	398	M	3.8
10	383	M	3.7




Regression

## Correlation

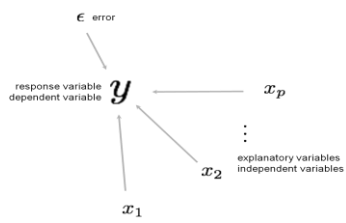
- Pearson's correlation
- Kendall's tau
- Spearman's rank correlation

```
SAT <- c(388, 354, 361, 329, 331, 364, 399, 421, 398, 383)
GPA <- c(4.0, 3.8, 3.5, 3.1, 3.3, 3.5, 4.0, 4.2, 3.8, 3.7)
cor(SAT, GPA, method="pearson")
cor(SAT, GPA, method="kendall")
cor(SAT, GPA, method="spearman")
```




Regression

## Regression Model



$$y = m(x_1, x_2, \dots, x_p) + \epsilon$$

where  $m(x_1, x_2, \dots, x_p) = E(y|x_1, x_2, \dots, x_p)$   
and  $\epsilon \sim N(0, \sigma^2)$



Regression

### Linear model

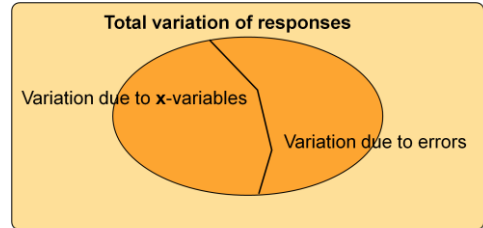
$$m(x_1, x_2, \dots, x_p) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

the expected change in  $y$  per unit change in  $x_1$   
when all the other regressors are held constant  
: *partial regression coefficient*

### Nonlinear model

(e.g)  $m(x) = A \cdot \exp(-\beta x)$

Regression



Null hypothesis.: The regression model is **not** significant

**F-test again!**

Regression



```
SAT <- c(388, 354, 361, 329, 331, 364, 399, 421, 398, 383)
GEN <- c("F", "F", "F", "F", "M", "M", "M", "F", "M", "M")
GPA <- c(4.0, 3.8, 3.5, 3.1, 3.3, 3.5, 4.0, 4.2, 3.8, 3.7)

par(mfrow=c(1,2))
plot(SAT, GPA)
boxplot(GPA~GEN, xlab="GENDER", ylab="GPA")

res <- lsfit(SAT, GPA)
ls.print(res)

res.no <- lsfit(SAT, GPA, intercept=F)
ls.print(res.no)

t.test(GPA~GEN)

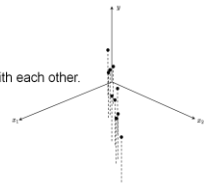
res.lm <- lm(GPA~GEN+SAT)
summary(res.lm)
```

Regression



### Multicollinearity

Regressors may have (nearly) linear dependency with each other.



VIF: variance inflation factor

$$VIF_j > 10$$

Regression



Remedies when multicollinearity is detected:

- ✓ Model re-specification
- ✓ Ridge regression
- ✓ Principal component regression

Regression



## Variable Selection

### • Stepwise regression

```
> data(state)
> statedata <- data.frame(state.x77,
  row.names=state.abb, check.names=T)
> g <- lm(Life.Exp~., data=statedata)
> summary(g)
> step(g)
```

Regression



January 23-January 24, 2012



## 10. Categorical Data Analysis

University of Georgia  
Department of Workforce Education, Leadership, and Social Foundations  
College of Education Research Office



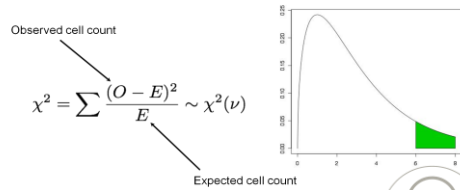
Any evidence of association between **education level** and **income level**? If any, **how strong** is it?

Education	Income		sum
	Low	High	
High school	25	12	37
College	11	14	25
sum	36	26	62

Categorical Data Analysis



To test whether  
*there is **no** association between factors*  
 - or -  
*there **is** an association between factors*



Categorical Data Analysis



```
M <- as.table(rbind(c(25, 12), c(11,14)))
dimnames(M) <- list(education=c("High School", "College"),
                    income=c("Low", "High"))
res <- chisq.test(M)
res$expected
res
```

Categorical Data Analysis

