CENG 3516 STATISTICAL COMPUTING

Lecture 1 - Introduction and Basics

Dr. Eralp DOGU

Feb 17, 2017

In-class exercise: Hello world!

```
"Hello world!"
```

```
## [1] "Hello world!"
```

```
print("Hello world!")
```

```
## [1] "Hello world!"
```

Setting and getting working directory

```
#getwd()
#setwd()
```

Basics: the class in a nutshell

- Everything we'll do comes down to applying **functions** to **data**
- **Data**: things like 7, "seven", 7.000, the matrix $\begin{bmatrix} 7 & 7 & 7 \\ 7 & 7 & 7 \end{bmatrix}$
- **Functions**: things like $\log$, $+$, $<$, $\mod$, `mean`, `sum`, `data`

  A function is a process which turns input objects (**arguments**) into an output object (**return value**)

Data building blocks

You'll encounter different kinds of data types

- **Booleans** Direct binary values: `TRUE` or `FALSE` in R
- **Integers**: whole numbers (positive, negative or zero)
- **Characters** fixed-length blocks of bits, with special coding **strings** = sequences of characters
- **Floating point numbers**: a fraction (with a finite number of bits) times an exponent, like 1.87×10^6
- **Missing or ill-defined values**: `NA`, `NaN`, etc.

Operators (functions)

You can use R as a calculator

Command	Description
`+,-,*,\`	add, subtract, multiply, divide
`^`	raise to the power of
`%%`	remainder after division (ex: `8 %% 3 = 2`)
`( )`	change the order of operations
`log(), exp()`	logarithms and exponents (ex: `log(10) = 2.302`)
`sqrt()`	square root
`round()`	round to the nearest whole number (ex: `round(2.3) = 2`)
`floor(), ceiling()`	round down or round up
`abs()`	absolute value

```
7 + 5 # Addition
```

```
## [1] 12
```

```
7 - 5 # Subtraction
```

```
## [1] 2
```

```
7 * 5 # Multiplication
```

```
## [1] 35
```

```
7 ^ 5 # Exponentiation
```

```
## [1] 16807
```

```
"Hello world."
```

```
## [1] "Hello world."
```

```
exp(1)
```

```
## [1] 2.718282
```

```
cos(3.141593)
```

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

```
log2(1)
```

```
## [1] 0
```

```r
7 / 5 # Division
```

```
## [1] 1.4
```

```r
7 %% 5 # Modulus
```

```
## [1] 2
```

```r
7 %/% 5 # Integer division
```

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

Operators cont'd.

Comparisons are also binary operators; they take two objects, like numbers, and give a Boolean

```r
7 > 5
```

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

```r
7 < 5
```

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

```r
7 >= 7
```

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

```r
7 <= 5
```

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

Operators cont'd.

```r
7 == 5
```

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

```r
7 != 5
```

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

Boolean operators

Basically "and" and "or":

```
(5 > 7) & (6*7 == 42)
```

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

```
(5 > 7) | (6*7 == 42)
```

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

More on types

- `typeof()` function returns the type
- `is.`*foo*`()` functions return Booleans for whether the argument is of type *foo*
- `as.`*foo*`()` (tries to) "cast" its argument to type *foo* — to translate it sensibly into a *foo*-type value

Special case: `as.factor()` will be important later for telling R when numbers are actually encodings and not numeric values. (E.g., 1 = High school grad; 2 = College grad; 3 = Postgrad) ###

```
typeof(7)
```

```
## [1] "double"
```

```
is.numeric(7)
```

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

```
is.na(7)
```

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

```
is.character(7)
```

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

```
is.character("7")
```

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

```
is.character("seven")
```

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

```r
is.na("seven")
```

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

Variables

We can give names to data objects; these give us **variables** A few variables are built in:

```r
pi
```

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

```r
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```r
## months in English
month.abb
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

```r
month.name
```

```
##  [1] "January"   "February"  "March"     "April"     "May"
##  [6] "June"      "July"      "August"    "September" "October"
## [11] "November"  "December"
```

```r
## months in your current locale
format(ISOdate(2000, 1:12, 1), "%B")
```

```
##  [1] "January"   "February"  "March"     "April"     "May"
##  [6] "June"      "July"      "August"    "September" "October"
## [11] "November"  "December"
```

```r
format(ISOdate(2000, 1:12, 1), "%b")
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

Variables can be arguments to functions or operators, just like constants:

```r
pi*10
```

```
## [1] 31.41593
```

```r
cos(pi)
```

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

```r
pi - 4*(4*atan(1/5) - atan(1/239))
```

```
## [1] -4.440892e-16
```

Assignment operator

Most variables are created with the **assignment operator**, `<-` or `=`

```r
x <- 2
x = 2
x
```

```
## [1] 2
```

```r
x ^ x
```

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

```r
x ^ 2
```

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

Assignment operator

Most variables are created with the **assignment operator**, `<-` or `=`

```r
hours <- 12
hours
```

```
## [1] 12
```

```r
days = 2.5
hours * days
```

```
## [1] 30
```

The assignment operator also changes values:

```r
total.hours<- hours * days
total.hours
```

```
## [1] 30
```

```
hourly.wage <- 45
payment<-total.hours*hourly.wage
```

- Using names and variables makes code: easier to design, easier to debug, less prone to bugs, easier to improve, and easier for others to read
- Avoid "magic constants"; use named variables
- Use descriptive variable names and dots between names, variable names are case sensitive
- Good: `num.students <- 29`
- Bad: `ns <- 29`

The workspace

What names have you defined values for?

```
ls()
```

```
## [1] "days"        "hourly.wage" "hours"       "payment"     "total.hours"
## [6] "x"
```

Removing variables:

```
rm("hourly.wage") ###remove variables
ls()
```

```
## [1] "days"        "hours"       "payment"     "total.hours" "x"
```

First data structure: vectors

- Group related data values into one object, a **data structure**
- A **vector** is a sequence of values, all of the same type
- `c()` function returns a vector containing all its arguments in order

```
x <- c(1:10)
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
x ^ x
```

```
##  [1]           1           4          27         256        3125
##  [6]       46656      823543    16777216   387420489 10000000000
```

```r
dim(x) <- c(2,5)
x <- c("Hello","world","!")
x
```

```
## [1] "Hello" "world" "!"
```

```r
x <- c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE)
x
```

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

```r
x <- list("R","12345",FALSE)
x
```

```
## [[1]]
## [1] "R"
##
## [[2]]
## [1] "12345"
##
## [[3]]
## [1] FALSE
```

```r
students <- c("Meltem", "Umut", "Olcay", "Busra", "Hayriye")
midterm <- c(80, 90, 93, 82, 95)
ls()
```

```
## [1] "days"        "hours"       "midterm"     "payment"     "students"
## [6] "total.hours" "x"
```

Using rep() and seq() functions

```r
Example1 <- seq(2:50)
Example1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49
```

```r
Example1 <- seq(1, 9, by = 2)
Example1
```

```
## [1] 1 3 5 7 9
```

```r
Example1 <-seq(1.575, 5.125, by = 0.05)
Example1
```

```
##  [1] 1.575 1.625 1.675 1.725 1.775 1.825 1.875 1.925 1.975 2.025 2.075
## [12] 2.125 2.175 2.225 2.275 2.325 2.375 2.425 2.475 2.525 2.575 2.625
## [23] 2.675 2.725 2.775 2.825 2.875 2.925 2.975 3.025 3.075 3.125 3.175
## [34] 3.225 3.275 3.325 3.375 3.425 3.475 3.525 3.575 3.625 3.675 3.725
## [45] 3.775 3.825 3.875 3.925 3.975 4.025 4.075 4.125 4.175 4.225 4.275
## [56] 4.325 4.375 4.425 4.475 4.525 4.575 4.625 4.675 4.725 4.775 4.825
## [67] 4.875 4.925 4.975 5.025 5.075 5.125
```

```
Example1 <- rep(1:10,2)
Example1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10
```

```
Example1 <- rep(1:2,c(10,15))
Example1
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
Example1 <- rep(1:2,each=10)
Example1
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
```

```
Example1 <- rep(1:2,c(10,10))
Example1
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
```

```
Example2 <- log(Example1)
Example2
```

```
##  [1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##  [8] 0.0000000 0.0000000 0.0000000 0.6931472 0.6931472 0.6931472 0.6931472
## [15] 0.6931472 0.6931472 0.6931472 0.6931472 0.6931472 0.6931472
```

```
ls()
```

```
## [1] "days"        "Example1"    "Example2"    "hours"       "midterm"
## [6] "payment"     "students"    "total.hours" "x"
```

Indexing

- `vec[1]` is the first element, `vec[4]` is the 4th element of `vec`

```
students
```

```
## [1] "Meltem"  "Umut"    "Olcay"   "Busra"   "Hayriye"
```

```
students[4]
```

```
## [1] "Busra"
```

- `vec[-4]` is a vector containing all but the fourth element

```
students[-4]
```

```
## [1] "Meltem"  "Umut"    "Olcay"   "Hayriye"
```

Second data structure: matrices

- A **matrix** is a set of vectors
- `matrix()` function returns a matrix containing all its arguments in order

```
matrix1<-matrix(1:10, 2, 5)
matrix1
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

```
matrix1[2,2]
```

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

```
matrix1[,2]
```

```
## [1] 3 4
```

```
matrix1[2,]
```

```
## [1]  2  4  6  8 10
```

```
matrix2<-matrix(0, 5, 2)
matrix2
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
## [3,]    0    0
## [4,]    0    0
## [5,]    0    0
```

```
matrix3<-matrix(NA, 5, 2)
matrix3
```

```
##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA
## [3,]   NA   NA
## [4,]   NA   NA
## [5,]   NA   NA
```

```
class(matrix3)
```

```
## [1] "matrix"
```

```
attributes(matrix1)
```

```
## $dim
## [1] 2 5
```

```
colnames(matrix1) <- c("A","B","C","D","E")
matrix4<-matrix(data=c(1,2,3,4,5,6,7,8,9,10,11,12),nrow=3,ncol=4)
matrix4
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
matrix5<-matrix(1:12,nrow=3,byrow=T)
matrix5
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

```
######Other ways of creating matrices
matrix6<-cbind(A=1:4,B=5:8,C=9:12)
matrix6
```

```
##      A B  C
## [1,] 1 5  9
## [2,] 2 6 10
## [3,] 3 7 11
## [4,] 4 8 12
```

```
matrix7<-rbind(A=1:4,B=5:8,C=9:12)
matrix7
```

```
##   [,1] [,2] [,3] [,4]
## A    1    2    3    4
## B    5    6    7    8
## C    9   10   11   12
```

Vector arithmetic

Operators apply to vectors "pairwise" or "elementwise":

```
final <- c(88, 84, 95, 82, 91) # Final exam scores
midterm # Midterm exam scores
```

```
## [1] 80 90 93 82 95
```

```
midterm + final # Sum of midterm and final scores
```

```
## [1] 168 174 188 164 186
```

```
((0.5)*midterm + (0.5)*final) # Average exam score
```

```
## [1] 84 87 94 82 93
```

```
course.grades <- (0.3)*midterm + (0.7)*final # Final course grade
course.grades
```

```
## [1] 85.6 85.8 94.4 82.0 92.2
```

Pairwise comparisons

Is the final score higher than the midterm score?

```
midterm
```

```
## [1] 80 90 93 82 95
```

```
final
```

```
## [1] 88 84 95 82 91
```

```
final > midterm
```

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

Boolean operators can be applied elementwise:

```
(final < midterm) & (midterm > 80)
```

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

Functions on vectors

Command	Description
`sum(vec)`	sums up all the elements of `vec`
`mean(vec)`	mean of `vec`
`median(vec)`	median of `vec`
`min(vec), max(vec)`	the largest or smallest element of `vec`
`sd(vec), var(vec)`	the standard deviation and variance of `vec`
`length(vec)`	the number of elements in `vec`
`pmax(vec1, vec2), pmin(vec1, vec2)`	example: `pmax(quiz1, quiz2)` returns the higher of quiz 1 and quiz 2 for each
`sort(vec)`	returns the `vec` in sorted order
`order(vec)`	returns the index that sorts the vector `vec`
`unique(vec)`	lists the unique elements of `vec`
`summary(vec)`	gives a five-number summary
`any(vec), all(vec)`	useful on Boolean vectors

Functions on vectors

```
course.grades
```

```
## [1] 85.6 85.8 94.4 82.0 92.2
```

```
sum(midterm,final)/2
```

```
## [1] 440
```

```
mean(course.grades) # mean grade
```

```
## [1] 88
```

```
median(course.grades)
```

```
## [1] 85.8
```

```
sd(course.grades) # grade standard deviation
```

```
## [1] 5.128353
```

More functions on vectors

```
sort(course.grades)
```

```
## [1] 82.0 85.6 85.8 92.2 94.4
```

```
max(course.grades) # highest course grade
```

```
## [1] 94.4
```

```
min(course.grades) # lowest course grade
```

```
## [1] 82
```

Referencing elements of vectors

```
students
```

```
## [1] "Meltem"  "Umut"    "Olcay"   "Busra"   "Hayriye"
```

Vector of indices:

```
students[c(2,4)]
```

```
## [1] "Umut"  "Busra"
```

Vector of negative indices

```
students[c(-1,-3)]
```

```
## [1] "Umut"    "Busra"   "Hayriye"
```

More referencing

`which()` returns the `TRUE` indexes of a Boolean vector:

```
course.grades
```

```
## [1] 85.6 85.8 94.4 82.0 92.2
```

```
a.threshold <- 90 # A grade = 90% or higher
course.grades >= a.threshold # vector of booleans
```

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

```
a.students <- which(course.grades >= a.threshold) # Applying which()
a.students
```

```
## [1] 3 5
```

```
students[a.students] # Names of A students
```

```
## [1] "Olcay"   "Hayriye"
```

Named components

You can give names to elements or components of vectors

```
students
```

```
## [1] "Meltem"  "Umut"    "Olcay"   "Busra"   "Hayriye"
```

```
names(course.grades) <- students # Assign names to the grades
names(course.grades)
```

```
## [1] "Meltem"  "Umut"    "Olcay"   "Busra"   "Hayriye"
```

```
course.grades[c("Meltem", "Umut", "Olcay")] # Get final grades for 3 students
```

```
## Meltem   Umut  Olcay
##   85.6   85.8   94.4
```

Note the labels in what R prints; these are not actually part of the value

Useful RStudio tips

Keystroke	Description
<tab>	autocompletes commands and filenames, and lists arguments for functions. Highly useful!
<up>	cycle through previous commands in the console prompt
<ctrl-up>	lists history of previous commands matching an unfinished one
<ctrl-enter>	paste current line from source window to console. Good for trying things out ideas from a source file.
<ESC>	as mentioned, abort an unfinished command and get out of the + prompt

CENG 3516 STATISTICAL COMPUTING

Lecture 2 - Introduction to Data

Dr. Eralp DOGU

Feb 24, 2017

Agenda

- Wrapping up vector indexing
- Importing data
- Simple summaries of categorical and continuous data
- Coding style

Importing data

- Start with builtin data

```
library()
library(MASS)
require(MASS)
??MASS
data(islands)
?islands
islands
```

```
##           Africa       Antarctica             Asia        Australia
##            11506             5500            16988             2968
##     Axel Heiberg           Baffin            Banks           Borneo
##               16              184               23              280
##          Britain          Celebes            Celon             Cuba
##               84               73               25               43
##            Devon        Ellesmere           Europe        Greenland
##               21               82             3745              840
##           Hainan       Hispaniola         Hokkaido           Honshu
##               13               30               30               89
##          Iceland          Ireland             Java           Kyushu
##               40               33               49               14
##           Luzon       Madagascar         Melville         Mindanao
##               42              227               16               36
##         Moluccas      New Britain       New Guinea  New Zealand (N)
##               29               15              306               44
##  New Zealand (S)     Newfoundland    North America    Novaya Zemlya
##               58               43             9390               32
##  Prince of Wales         Sakhalin    South America      Southampton
##               13               29             6795               16
##      Spitsbergen          Sumatra           Taiwan         Tasmania
##               15              183               14               26
## Tierra del Fuego            Timor        Vancouver         Victoria
##               19               13               12               82
```

```
Melanoma[1:7,]
```

```
##   time status sex age year thickness ulcer
## 1   10      3   1  76 1972      6.76     1
## 2   30      3   1  56 1968      0.65     0
## 3   35      2   1  41 1977      1.34     0
## 4   99      3   0  71 1968      2.90     0
## 5  185      1   1  52 1965     12.08     1
## 6  204      1   1  28 1971      4.84     1
## 7  210      1   1  77 1972      5.16     1
```

```
Melanoma$status
```

```
##   [1] 3 3 2 3 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 3 1 2 1 2 2 2 1 3 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 1 2
##  [71] 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
## [106] 2 2 2 2 2 1 1 2 3 2 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 2 2 2
## [141] 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 1 2 2 2 2
## [176] 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
attach(Melanoma)
status
```

```
##   [1] 3 3 2 3 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 3 1 2 1 2 2 2 1 3 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 1 2
##  [71] 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
## [106] 2 2 2 2 2 1 1 2 3 2 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 2 2 2
## [141] 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 1 2 2 2 2
## [176] 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
search()
```

```
##  [1] ".GlobalEnv"        "Melanoma"          "package:MASS"
##  [4] "package:stats"     "package:graphics"  "package:grDevices"
##  [7] "package:utils"     "package:datasets"  "package:methods"
## [10] "Autoloads"         "package:base"
```

```
Melanoma2<-subset(Melanoma,year<1970)
Melanoma2[1:7,]
```

```
##    time status sex age year thickness ulcer
## 2    30      3   1  56 1968      0.65     0
## 4    99      3   0  71 1968      2.90     0
## 5   185      1   1  52 1965     12.08     1
## 9   232      1   1  49 1968     12.88     1
## 11  295      1   0  53 1969      4.19     1
## 13  386      1   0  68 1965      3.87     1
## 15  469      1   0  14 1969      2.42     1
```

```
Melanoma3<-transform(Melanoma,thickness=log(thickness))
Melanoma3[1:7,]
```

```
##   time status sex age year  thickness ulcer
## 1   10      3   1  76 1972  1.9110229     1
## 2   30      3   1  56 1968 -0.4307829     0
## 3   35      2   1  41 1977  0.2926696     0
## 4   99      3   0  71 1968  1.0647107     0
## 5  185      1   1  52 1965  2.4915512     1
## 6  204      1   1  28 1971  1.5769147     1
## 7  210      1   1  77 1972  1.6409366     1
```

```
detach(Melanoma)
search()
```

```
##  [1] ".GlobalEnv"        "package:MASS"      "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

- To import tabular data into R, we use the **`read.table()`** command.
- Let's parse this command one component at a time
- The data is in a file called **`irisdata.csv`**, which is an online file
- The file contains a **`header`** as its first row
- The csv format means that the data is comma-separated, so **`sep=","`**

To import tabular data into R, we also use the **`read.csv()`** command with a preset seperation argument.

Exploring the data

- R imports data into a **`data.frame`** object

```
address="https://vincentarelbundock.github.io/Rdatasets/csv/datasets/iris.csv"
irisdata <- read.table(file=url(address),sep = ",", header = T)
irisdata [1:7,]
```

```
##   X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1 1          5.1         3.5          1.4         0.2  setosa
## 2 2          4.9         3.0          1.4         0.2  setosa
## 3 3          4.7         3.2          1.3         0.2  setosa
## 4 4          4.6         3.1          1.5         0.2  setosa
## 5 5          5.0         3.6          1.4         0.2  setosa
## 6 6          5.4         3.9          1.7         0.4  setosa
## 7 7          4.6         3.4          1.4         0.3  setosa
```

```
irisdata <- read.csv(
            paste("https://vincentarelbundock.github.io/Rdatasets/csv/datasets/iris.csv"),
            header=T
                    )
irisdata [1:7,]
```

```
##   X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1 1          5.1         3.5          1.4         0.2  setosa
## 2 2          4.9         3.0          1.4         0.2  setosa
## 3 3          4.7         3.2          1.3         0.2  setosa
## 4 4          4.6         3.1          1.5         0.2  setosa
## 5 5          5.0         3.6          1.4         0.2  setosa
## 6 6          5.4         3.9          1.7         0.4  setosa
## 7 7          4.6         3.4          1.4         0.3  setosa
```

- To view the first few rows of the data, use **`head()`**

```
head(irisdata, 3)
```

```
##   X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1 1          5.1         3.5          1.4         0.2  setosa
## 2 2          4.9         3.0          1.4         0.2  setosa
## 3 3          4.7         3.2          1.3         0.2  setosa
```

- `head(data.frame, n)` returns the first `n` rows of the data frame
- In the Console, you can also use `View(irisdata)` to get a spreadsheet view

Simple summary

- Use the `str()` function to get a simple summary of your data set

```
str(irisdata)
```

```
## 'data.frame':    150 obs. of  6 variables:
##  $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Another simple summary

```
summary(irisdata)
```

```
##        X           Sepal.Length    Sepal.Width     Petal.Length
##  Min.   :  1.00   Min.   :4.300   Min.   :2.000   Min.   :1.000
##  1st Qu.: 38.25   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
##  Median : 75.50   Median :5.800   Median :3.000   Median :4.350
##  Mean   : 75.50   Mean   :5.843   Mean   :3.057   Mean   :3.758
##  3rd Qu.:112.75   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
##  Max.   :150.00   Max.   :7.900   Max.   :4.400   Max.   :6.900
##   Petal.Width          Species
##  Min.   :0.100   setosa    :50
```

```
##  1st Qu.:0.300   versicolor:50
##  Median :1.300   virginica :50
##  Mean   :1.199
##  3rd Qu.:1.800
##  Max.   :2.500
```

Data frame basics

- To see what an R object is made up of, you can use `attributes()`

```
attributes(irisdata)
```

```
## $names
## [1] "X"            "Sepal.Length" "Sepal.Width"  "Petal.Length"
## [5] "Petal.Width"  "Species"
##
## $class
## [1] "data.frame"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
##  [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
##  [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
##  [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
## [103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
## [120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
## [137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150
```

An R **data frame** is a *list* whose columns you can refer to by *name* or *index*

Data frame dimensions

- We can use `nrow()` and `ncol` to determine the number of variables and cases

```
nrow(irisdata) # Number of rows (responses)
```

```
## [1] 150
```

```
ncol(irisdata) # Number of columns (questions)
```

```
## [1] 6
```

Indexing data frames

- There are many different ways of indexing the same piece of a data frame
- Each vector below contains 150 entries. For display purposes, the settings have been adjusted so that only the first 22 are shown below

```r
head(irisdata[["Sepal.Length"]], 22)
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
## [18] 5.1 5.7 5.1 5.4 5.1
```

```r
head(irisdata$Sepal.Length, 22)
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
## [18] 5.1 5.7 5.1 5.4 5.1
```

```r
head(irisdata[,1], 22)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

More indexing

- Note that single brackets and double brackets have different effects

```r
head(irisdata[["Sepal.Length"]], 22)  # Returns the Sepal.Length column as a vector
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
## [18] 5.1 5.7 5.1 5.4 5.1
```

```r
head(irisdata["Sepal.Length"],22) # sub-data-frame containing only "Sepal.Length"
```

```
##    Sepal.Length
## 1           5.1
## 2           4.9
## 3           4.7
## 4           4.6
## 5           5.0
## 6           5.4
## 7           4.6
## 8           5.0
## 9           4.4
## 10          4.9
## 11          5.4
## 12          4.8
## 13          4.8
## 14          4.3
## 15          5.8
## 16          5.7
## 17          5.4
## 18          5.1
## 19          5.7
## 20          5.1
## 21          5.4
## 22          5.1
```

Indexing multiple columns

```
head(irisdata[, c(1,5)]) # Data from 1st and 5th columns
```

```
##   X Petal.Width
## 1 1         0.2
## 2 2         0.2
## 3 3         0.2
## 4 4         0.2
## 5 5         0.2
## 6 6         0.4
```

```
head(irisdata[c("Sepal.Length", "Species")]) # Data from "Sepal.Length" and "Species"
```

```
##   Sepal.Length Species
## 1          5.1  setosa
## 2          4.9  setosa
## 3          4.7  setosa
## 4          4.6  setosa
## 5          5.0  setosa
## 6          5.4  setosa
```

Indexing rows and columns

- Data frames have two dimensions to index across

```
irisdata[6,] # 6th row
```

```
##   X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 6 6          5.4         3.9          1.7         0.4  setosa
```

```
irisdata[6,5] # row 6, column 5
```

```
## [1] 0.4
```

```
irisdata[6, "Species"] # Species of 6th row
```

```
## [1] setosa
## Levels: setosa versicolor virginica
```

```
irisdata[["Species"]][6]  # Does the same thing
```

```
## [1] setosa
## Levels: setosa versicolor virginica
```

More indexing

```r
irisdata[1:3,]  # equivalent to head(irisdata, 3)
```

```
##   X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1 1          5.1         3.5          1.4         0.2  setosa
## 2 2          4.9         3.0          1.4         0.2  setosa
## 3 3          4.7         3.2          1.3         0.2  setosa
```

```r
irisdata[3:5, c(1,5)]
```

```
##   X Petal.Width
## 3 3         0.2
## 4 4         0.2
## 5 5         0.2
```

Subsets of data

- We are often interested in learning something a specific subset of the data

```r
head(irisdata[irisdata$Species=="virginica", ], 22) # Data from virginicas
head(irisdata[which(irisdata$Species=="virginica"), ], 22) # Does the same thing
```

```
##       X Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 101 101          6.3         3.3          6.0         2.5 virginica
## 102 102          5.8         2.7          5.1         1.9 virginica
## 103 103          7.1         3.0          5.9         2.1 virginica
## 104 104          6.3         2.9          5.6         1.8 virginica
## 105 105          6.5         3.0          5.8         2.2 virginica
## 106 106          7.6         3.0          6.6         2.1 virginica
## 107 107          4.9         2.5          4.5         1.7 virginica
## 108 108          7.3         2.9          6.3         1.8 virginica
## 109 109          6.7         2.5          5.8         1.8 virginica
## 110 110          7.2         3.6          6.1         2.5 virginica
## 111 111          6.5         3.2          5.1         2.0 virginica
## 112 112          6.4         2.7          5.3         1.9 virginica
## 113 113          6.8         3.0          5.5         2.1 virginica
## 114 114          5.7         2.5          5.0         2.0 virginica
## 115 115          5.8         2.8          5.1         2.4 virginica
## 116 116          6.4         3.2          5.3         2.3 virginica
## 117 117          6.5         3.0          5.5         1.8 virginica
## 118 118          7.7         3.8          6.7         2.2 virginica
## 119 119          7.7         2.6          6.9         2.3 virginica
## 120 120          6.0         2.2          5.0         1.5 virginica
## 121 121          6.9         3.2          5.7         2.3 virginica
## 122 122          5.6         2.8          4.9         2.0 virginica
```

Cleaner subsetting

- When the subset conditions get long or messy, it is preferable to use the **`subset()`** function
- Here's an example of selecting the OperatingSystem and TVhours responses from all of the students who are either in PPM or Other and who listed their R experience as "Basic competence".

```
head(subset(irisdata, select=c("Species", "Sepal.Length")),22)
```

```
##    Species Sepal.Length
## 1   setosa          5.1
## 2   setosa          4.9
## 3   setosa          4.7
## 4   setosa          4.6
## 5   setosa          5.0
## 6   setosa          5.4
## 7   setosa          4.6
## 8   setosa          5.0
## 9   setosa          4.4
## 10  setosa          4.9
## 11  setosa          5.4
## 12  setosa          4.8
## 13  setosa          4.8
## 14  setosa          4.3
## 15  setosa          5.8
## 16  setosa          5.7
## 17  setosa          5.4
## 18  setosa          5.1
## 19  setosa          5.7
## 20  setosa          5.1
## 21  setosa          5.4
## 22  setosa          5.1
```

Splitting a long function call

- As your function calls get longer and more complicated, you may find it useful to split them over multiple lines
- Here's one way to rewrite the previous line

```
head(
    subset(irisdata,
       select=c("Species", "Sepal.Length"),
       subset = (Species == "virginica" | Species == "versicolor") &
                (Sepal.Length >= 5)
       ),
    20)
```

```
##       Species Sepal.Length
## 51 versicolor          7.0
## 52 versicolor          6.4
## 53 versicolor          6.9
## 54 versicolor          5.5
## 55 versicolor          6.5
## 56 versicolor          5.7
## 57 versicolor          6.3
## 59 versicolor          6.6
## 60 versicolor          5.2
## 61 versicolor          5.0
```

```
## 62 versicolor          5.9
## 63 versicolor          6.0
## 64 versicolor          6.1
## 65 versicolor          5.6
## 66 versicolor          6.7
## 67 versicolor          5.6
## 68 versicolor          5.8
## 69 versicolor          6.2
## 70 versicolor          5.6
## 71 versicolor          5.9
```

Some simple calculations

```
mean(irisdata$Sepal.Length)
```

```
## [1] 5.843333
```

```
mean(irisdata$Sepal.Length[irisdata$Species == "virginica"]) # Average time PPM's spent watching TV
```

```
## [1] 6.588
```

R style recommendations

Enforced style: Assignment operator

Assignment operator. USE <-

```
student.names <- c("Eralp", "Zeynep", "Dogu")  # Good
student.names = c("Eralp", "Zeynep", "Dogu") # Bad
```

- Note: When specifying function arguments, only = is valid

```
sort(irisdata, decreasing=TRUE) # Good
sort(irisdata, decreasing<-TRUE) # Bad!!
```

Enforced style: Spacing

- Binary operators should have spaces around them
- Commas should have a space after, but not before (just like in writing)

```
3 * 4 # Good
3*4 # Bad
which(student.names == "Eralp") # Good
which(student.names=="Eralp") # Bad
```

- For specifying arguments, spacing around = is optional

```r
sort(irisdata, decreasing=TRUE) # Accepted
sort(irisdata, decreasing = FALSE) # Accepted
```

Enforced style: Variable names

- To make code easy to read, debug, and maintain, you should use **concise** but **descriptive** variable names
- Terms in variable names should be separated by `_` or `.`

```r
# Accepted
day_one   day.one   day_1   day.1   day1

# Bad
d1   DayOne   dayone

# Can be made more concise:
first.day.of.the.month
```

- Avoid using variable names that are already pre-defined in R

```r
# EXTREMELY bad:
c   T   pi   sum   mean
```

CENG 3516 STATISTICAL COMPUTING

Lecture 3 - Data summaries and standard graphics

Dr. Eralp DOGU

Mar 3, 2017

Agenda

- Summaries with the aggregate() function
- Standard graphics

Getting started: birthwt data set–Risk Factors Associated with Low Infant Birth Weight

- We're going to start by operating on the `birthwt` dataset from the MASS library
- Description: The birthwt data frame has 189 rows and 10 columns. The data were collected at Baystate Medical Center, Springfield, Mass during 1986.
- Usage: birthwt
- Format: This data frame contains the following columns:

low: indicator of birth weight less than 2.5 kg.

age: mother's age in years.

lwt: mother's weight in pounds at last menstrual period.

race: mother's race (1 = white, 2 = black, 3 = other).

smoke: smoking status during pregnancy.

ptl: number of previous premature labours.

ht: history of hypertension.

ui: presence of uterine irritability.

ftv: number of physician visits during the first trimester.

bwt: birth weight in grams.

```
library(MASS)
str(birthwt)
```

```
## 'data.frame':    189 obs. of  10 variables:
##  $ low  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age  : int  19 33 20 21 18 21 22 17 29 26 ...
##  $ lwt  : int  182 155 105 108 107 124 118 103 123 113 ...
##  $ race : int  2 3 1 1 1 3 1 3 1 1 ...
##  $ smoke: int  0 0 1 1 1 0 0 0 1 1 ...
##  $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
##  $ bwt  : int  2523 2551 2557 2594 2600 2622 2637 2637 2663 2665 ...
```

Renaming the variables

- The dataset doesn't come with very descriptive variable names
- Better names can be used for better understanding

```
colnames(birthwt)
```

```
##  [1] "low"   "age"   "lwt"   "race"  "smoke" "ptl"   "ht"    "ui"
##  [9] "ftv"   "bwt"
```

```
# The default names are not very descriptive

colnames(birthwt) <- c("birthwt.below.2500", "mother.age", "mother.weight",
    "race", "mother.smokes", "previous.prem.labor", "hypertension", "uterine.irr",
    "physician.visits", "birthwt.grams")

# Better names!
```

Renaming the factors

- All the factors are currently represented as integers
- Let's use the `transform()` and `mapvalues()` functions to convert variables to factors and give the factors more meaningful levels

```
library(plyr)
birthwt <- transform(birthwt,
            race = as.factor(mapvalues(race, c(1, 2, 3),
                             c("white","black", "other"))),
            mother.smokes = as.factor(mapvalues(mother.smokes,
                             c(0,1), c("no", "yes"))),
            hypertension = as.factor(mapvalues(hypertension,
                             c(0,1), c("no", "yes"))),
            uterine.irr = as.factor(mapvalues(uterine.irr,
                             c(0,1), c("no", "yes"))),
            birthwt.below.2500 = as.factor(mapvalues(birthwt.below.2500,
                             c(0,1), c("no", "yes")))
            )
```

Summary of the data

- Now that things are coded correctly, we can look at an overall summary

```
summary(birthwt)
```

```
##  birthwt.below.2500   mother.age    mother.weight      race
##  no :130            Min.   :14.00   Min.   : 80.0   black:26
##  yes: 59            1st Qu.:19.00   1st Qu.:110.0   other:67
##                     Median :23.00   Median :121.0   white:96
##                     Mean   :23.24   Mean   :129.8
```

```
##                     3rd Qu.:26.00   3rd Qu.:140.0
##                     Max.   :45.00   Max.   :250.0
##  mother.smokes previous.prem.labor hypertension uterine.irr
##  no :115       Min.   :0.0000      no :177      no :161
##  yes: 74       1st Qu.:0.0000      yes: 12      yes: 28
##                Median :0.0000
##                Mean   :0.1958
##                3rd Qu.:0.0000
##                Max.   :3.0000
##  physician.visits birthwt.grams
##  Min.   :0.0000   Min.   : 709
##  1st Qu.:0.0000   1st Qu.:2414
##  Median :0.0000   Median :2977
##  Mean   :0.7937   Mean   :2945
##  3rd Qu.:1.0000   3rd Qu.:3487
##  Max.   :6.0000   Max.   :4990
```

```
mean(birthwt$birthwt.grams)
```

```
## [1] 2944.587
```

```
# Calculate mean for smokers and nonsmokers
sd(birthwt$birthwt.grams)
```

```
## [1] 729.2143
```

```
# Calculate standard deviation for smokers and nonsmokers
fivenum(birthwt$birthwt.grams)
```

```
## [1]  709 2414 2977 3487 4990
```

```
# Calculate five number summary for smokers and nonsmokers
```

A simple table

- Let's use the `tapply()` function to see what the average birthweight looks like when broken down by race and smoking status

```
with(birthwt, tapply(birthwt.grams, INDEX = list(race, mother.smokes), FUN = mean))
```

```
##             no      yes
## black 2854.500 2504.000
## other 2815.782 2757.167
## white 3428.750 2826.846
```

```
with(birthwt, tapply(birthwt.grams, INDEX = list(race, mother.smokes), FUN = sd))
```

```
##             no      yes
## black 621.2543 637.0568
## other 709.3493 810.0446
## white 710.0989 626.4725
```

```r
with(birthwt, tapply(birthwt.grams, INDEX = list(race, mother.smokes), FUN = median))
```

```
##         no    yes
## black 2920 2381.0
## other 2807 3146.5
## white 3593 2775.5
```

```r
with(birthwt, tapply(birthwt.grams, INDEX = list(race, mother.smokes), FUN = IQR))
```

```
##          no    yes
## black 850.5 627.25
## other 940.0 905.25
## white 811.0 779.50
```

aggregate() function

- Let's first recall what `tapply()` does
- Command: `tapply(X, INDEX, FUN)`
  - Applies `FUN` to `X` grouped by factors in `INDEX`
- **`aggregate()`** performs a similar operation, but presents the results in a form that is at times more convenient
- There are many ways to call the `aggregate()` function
- Analog of `tapply` call: `aggregate(X, by, FUN)`
  - Here, `by` is exactly like `INDEX`

Example: tapply vs aggregate

```r
library(MASS)
with(birthwt, tapply(birthwt.grams, INDEX = list(race, mother.smokes), FUN = mean)) # tapply
```

```
##             no      yes
## black 2854.500 2504.000
## other 2815.782 2757.167
## white 3428.750 2826.846
```

```r
with(birthwt, aggregate(birthwt.grams, by = list(race, mother.smokes), FUN = mean)) # aggregate
```

```
##   Group.1 Group.2        x
## 1   black      no 2854.500
## 2   other      no 2815.782
## 3   white      no 3428.750
## 4   black     yes 2504.000
## 5   other     yes 2757.167
## 6   white     yes 2826.846
```

```
# What the average birthweight looks like when broken down by race and hypertention?
```

Example: different syntax

- Here's a convenient alternative way to call `aggregate`
- It uses the R `formula` syntax, which we'll learn more about when we discuss regression

```
aggregate(birthwt.grams ~ race + mother.smokes, FUN=mean, data=birthwt)
```

```
##    race mother.smokes birthwt.grams
## 1 black            no      2854.500
## 2 other            no      2815.782
## 3 white            no      3428.750
## 4 black           yes      2504.000
## 5 other           yes      2757.167
## 6 white           yes      2826.846
```

A closer look at low birth weight

```
table(birthwt$race,birthwt$mother.smokes)
```

```
##
##         no yes
##   black 16  10
##   other 55  12
##   white 44  52
```

```
weight.smoke.tbl <- with(birthwt, table(birthwt.below.2500, mother.smokes))
weight.smoke.tbl
```

```
##                    mother.smokes
## birthwt.below.2500 no yes
##                no  86  44
##                yes 29  30
```

```
# Create the first crosstab using with() function
```

- Is the mother's age correlated with birth weight?

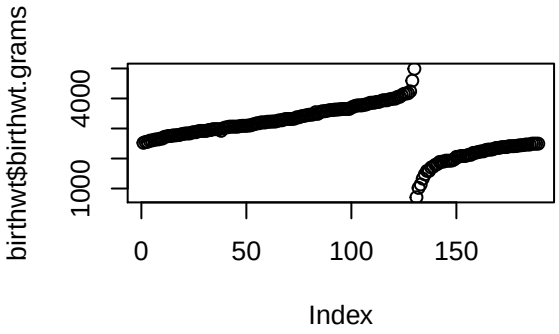
```
par(mfrow = c(2,2))
plot(birthwt$birthwt.grams)
plot(birthwt$birthwt.grams,birthwt$mother.age)
plot(birthwt$birthwt.grams,birthwt$mother.weight)
cor(birthwt$birthwt.grams,birthwt$mother.weight)
```

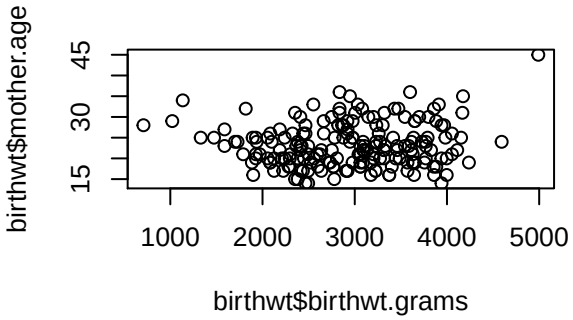
```
## [1] 0.1857333
```

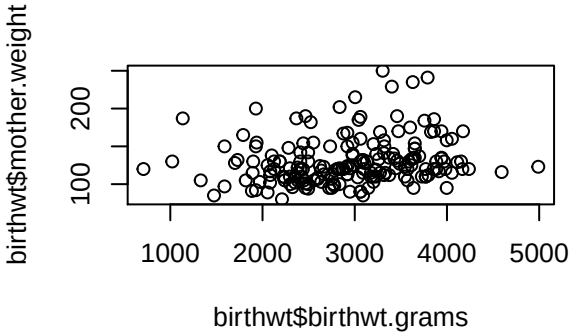
```r
cor(birthwt$birthwt.grams,birthwt$mother.age)  # Calculate correlation
```

```
## [1] 0.09031781
```

```r
# Calculate the correlation coefficient between birthweight and mother's age using with() function
```

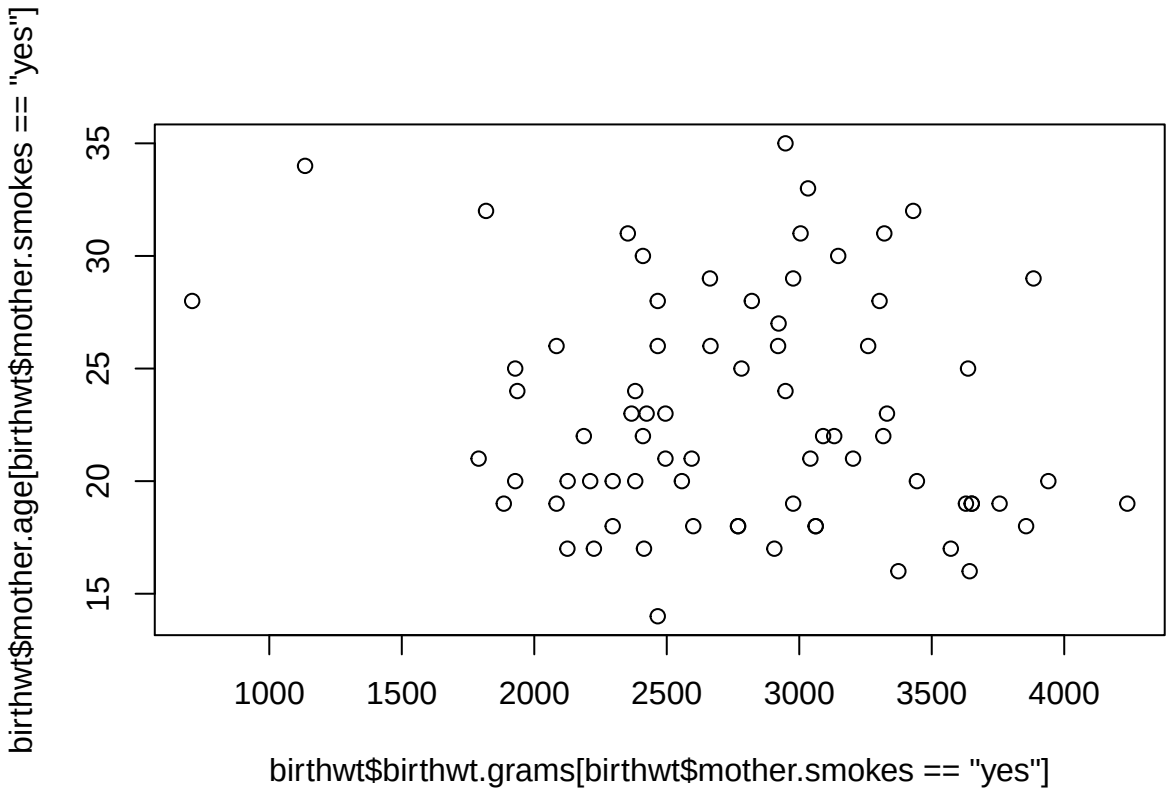






- Does this change when we account for smoking status?

```r
par(mfrow = c(1,1))
plot(birthwt$birthwt.grams[birthwt$mother.smokes == "yes"], birthwt$mother.age[birthwt$mother.smokes ==
```




```
# Create previous plot using with() function
# Calculate the correlation coefficient for the previous plot using with() function
# Compute the summary statistics and correlation for nonsmokers and create a plot for it.
```

Faster way: by() function

- Think of the `by(data, INDICES, FUN)` function as a `tapply()` function that operates on data frames instead of just vectors
- When using `tapply(X, INDEX, FUN)`, `X` is generally a numeric vector
- To calculate correlations, we need to allow `X` to be a data frame or matrix

```
by(data = birthwt,
   INDICES = birthwt[,"mother.smokes"],
   FUN = summary)
```

```
## birthwt[, "mother.smokes"]: no
##  birthwt.below.2500   mother.age    mother.weight      race
##  no :86             Min.   :14.00   Min.   : 85.0   black:16
##  yes:29             1st Qu.:20.00   1st Qu.:112.0   other:55
##                     Median :23.00   Median :124.0   white:44
##                     Mean   :23.43   Mean   :130.9
##                     3rd Qu.:26.00   3rd Qu.:141.5
##                     Max.   :45.00   Max.   :241.0
##  mother.smokes previous.prem.labor hypertension uterine.irr
##  no :115       Min.   :0.0000      no :108      no :100
```

```
##  yes:  0       1st Qu.:0.0000      yes:  7      yes: 15
##                Median :0.0000
##                Mean   :0.1217
##                3rd Qu.:0.0000
##                Max.   :2.0000
##  physician.visits birthwt.grams
##  Min.   :0.0000   Min.   :1021
##  1st Qu.:0.0000   1st Qu.:2509
##  Median :1.0000   Median :3100
##  Mean   :0.8174   Mean   :3056
##  3rd Qu.:1.0000   3rd Qu.:3622
##  Max.   :4.0000   Max.   :4990
## ------------------------------------------------------
## birthwt[, "mother.smokes"]: yes
##  birthwt.below.2500   mother.age    mother.weight      race
##  no :44             Min.   :14.00   Min.   : 80.0   black:10
##  yes:30             1st Qu.:19.00   1st Qu.:107.2   other:12
##                     Median :22.00   Median :120.0   white:52
##                     Mean   :22.95   Mean   :128.1
##                     3rd Qu.:26.00   3rd Qu.:137.2
##                     Max.   :35.00   Max.   :250.0
##  mother.smokes previous.prem.labor hypertension uterine.irr
##  no : 0        Min.   :0.0000      no :69       no :61
##  yes:74        1st Qu.:0.0000      yes: 5       yes:13
##                Median :0.0000
##                Mean   :0.3108
##                3rd Qu.:0.0000
##                Max.   :3.0000
##  physician.visits birthwt.grams
##  Min.   :0.0000   Min.   : 709
##  1st Qu.:0.0000   1st Qu.:2370
##  Median :0.0000   Median :2776
##  Mean   :0.7568   Mean   :2772
##  3rd Qu.:1.0000   3rd Qu.:3246
##  Max.   :6.0000   Max.   :4238
```

```
by(data = birthwt$birthwt.below.2500,
   INDICES = birthwt["mother.smokes"],
   FUN = summary)
```

```
## mother.smokes: no
##  no yes
##  86  29
## ------------------------------------------------------
## mother.smokes: yes
##  no yes
##  44  30
```

```
by(data = birthwt$mother.weight,
   INDICES = birthwt["mother.smokes"],
   FUN = mean)
```

```
## mother.smokes: no
```

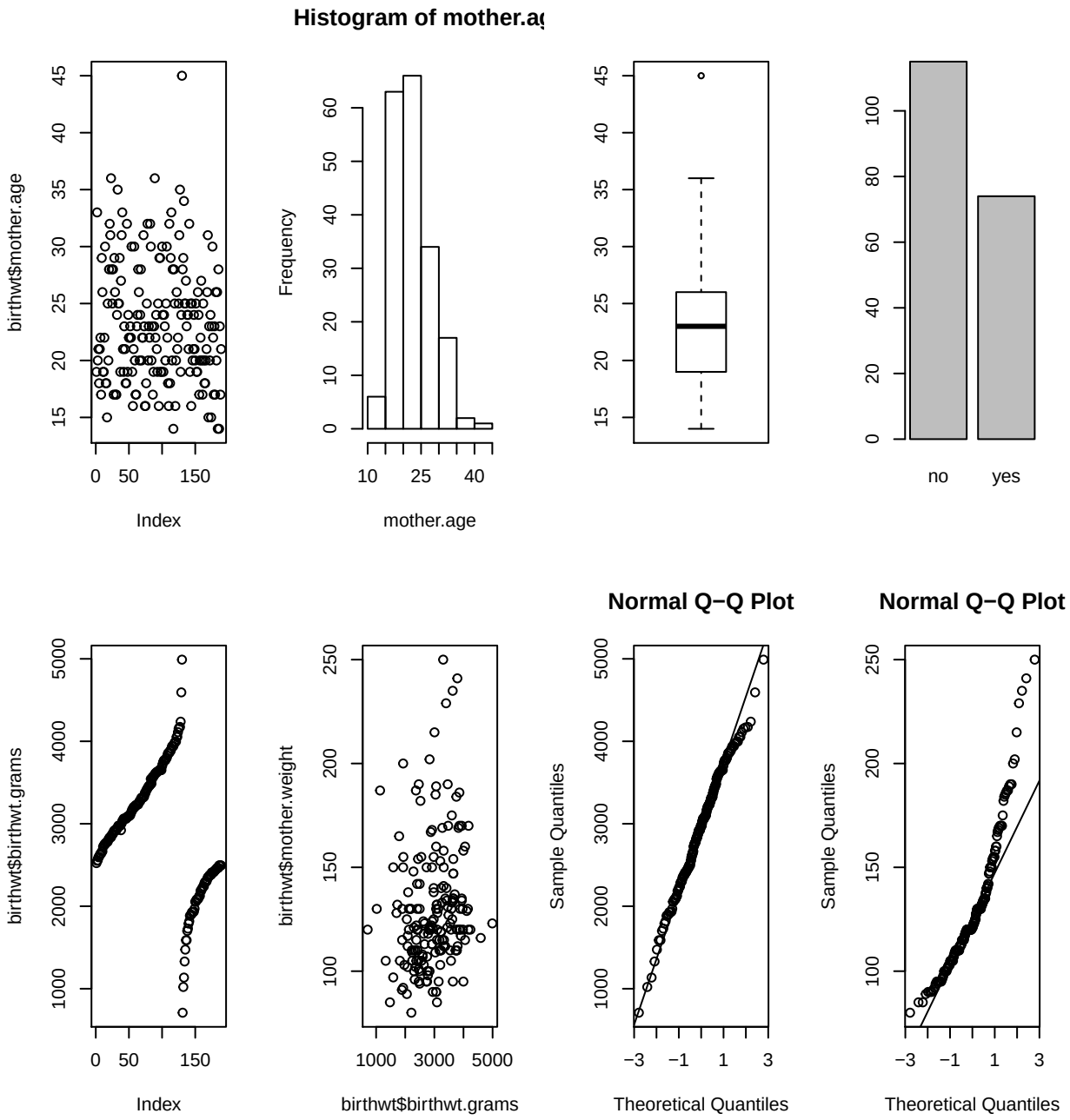
```
## [1] 130.8957
## ------------------------------------------------------------
## mother.smokes: yes
## [1] 128.1351
```

Standard graphics in R

Single-variable plots Let's continue with the `birthwt` data from the `MASS` library.

Here are some basic single-variable plots.

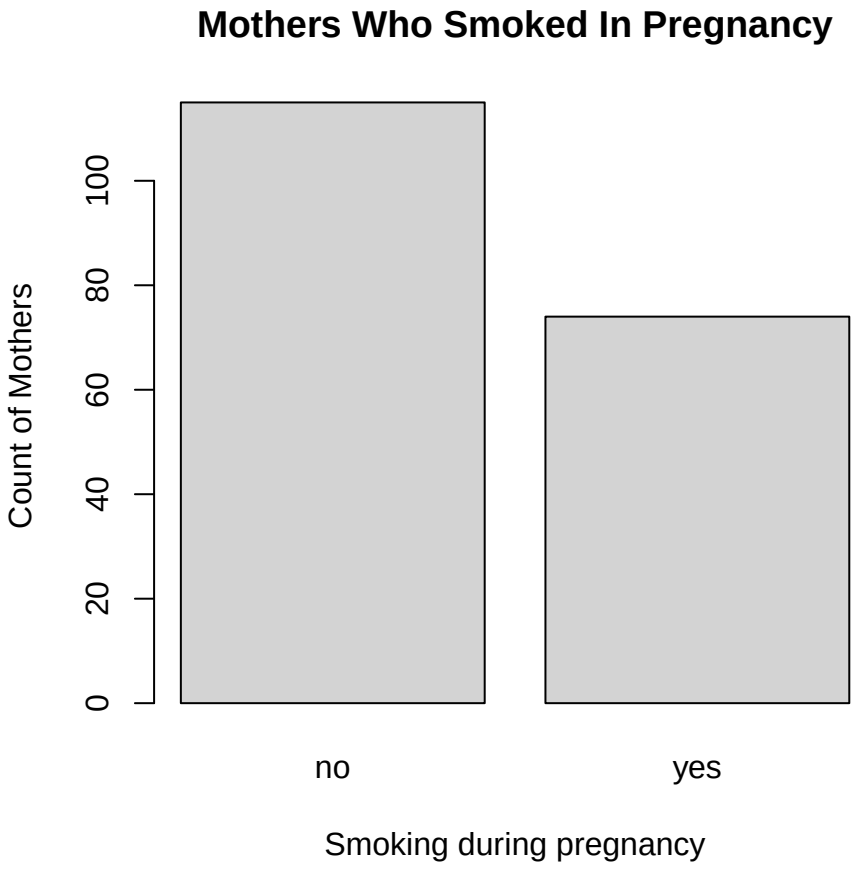
```
par(mfrow = c(2,4)) # Display plots in a single 2 x 2 figure
plot(birthwt$mother.age)
with(birthwt, hist(mother.age))
with(birthwt, boxplot(mother.age))
plot(birthwt$mother.smokes)
plot(birthwt$birthwt.grams)
plot(birthwt$birthwt.grams,birthwt$mother.weight)
qqnorm(birthwt$birthwt.grams)
qqline(birthwt$birthwt.grams)
qqnorm(birthwt$mother.weight)
qqline(birthwt$mother.weight)
```



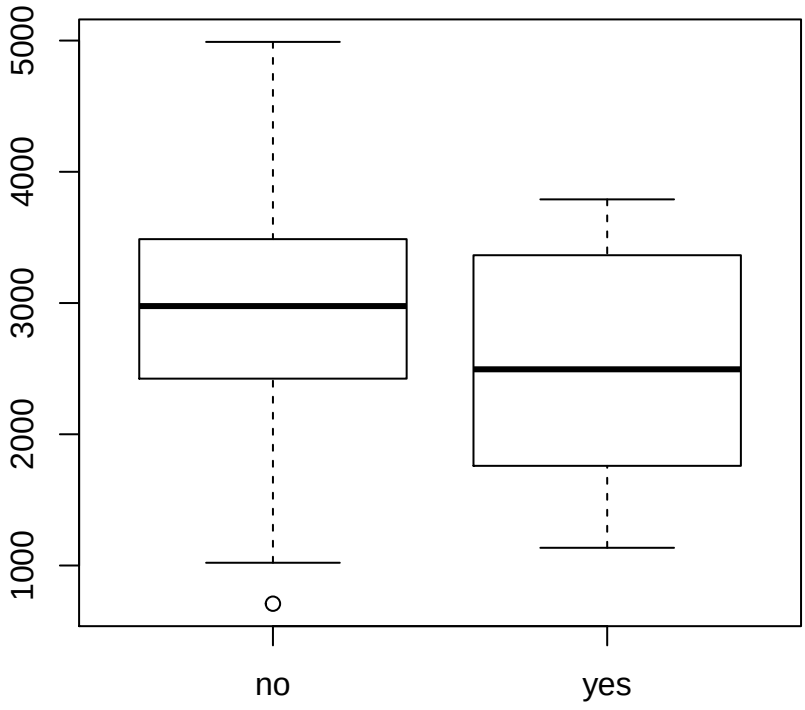
Note that the result of calling `plot(x, ...)` varies depending on what `x` is.
- When `x` is *numeric*, you get a plot showing the value of `x` at every index.
- When `x` is a *factor*, you get a bar plot of counts for every level

Let's add more information to the smoking bar plot, and also change the color by setting the `col` option.

```
par(mfrow = c(1,1))
plot(birthwt$mother.smokes,
     main = "Mothers Who Smoked In Pregnancy",
     xlab = "Smoking during pregnancy",
     ylab = "Count of Mothers",
     col = "lightgrey")
```




```
boxplot(birthwt$birthwt.grams~birthwt$hypertension)
```



```
# Create a boxplot for birthweights conditioned on smoking
# Create a boxplot for birthweights conditioned on physician visits
```

(much) better graphics with ggplot2

Introduction to ggplot2 ggplot2 has a slightly steeper learning curve than the base graphics functions, but it also generally produces far better and more easily customizable graphics.

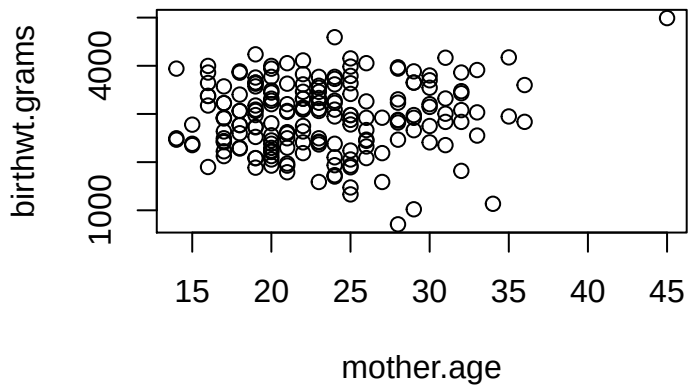
There are two basic calls in ggplot:

- `qplot(x, y, ..., data)`: a "quick-plot" routine, which essentially replaces the base `plot()`
- `ggplot(data, aes(x, y, ...), ...)`: defines a graphics object from which plots can be generated, along with *aesthetic mappings* that specify how variables are mapped to visual properties.

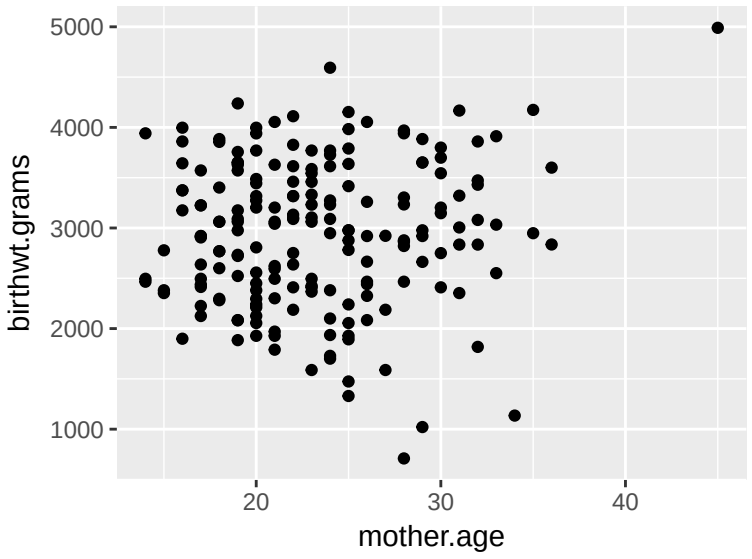
```
library(ggplot2)
```

plot vs qplot Here's how the default scatterplots look in ggplot compared to the base graphics. We'll illustrate things by continuing to use the birthwt data from the `MASS` library.

```
with(birthwt, plot(mother.age, birthwt.grams))  # Base graphics
```

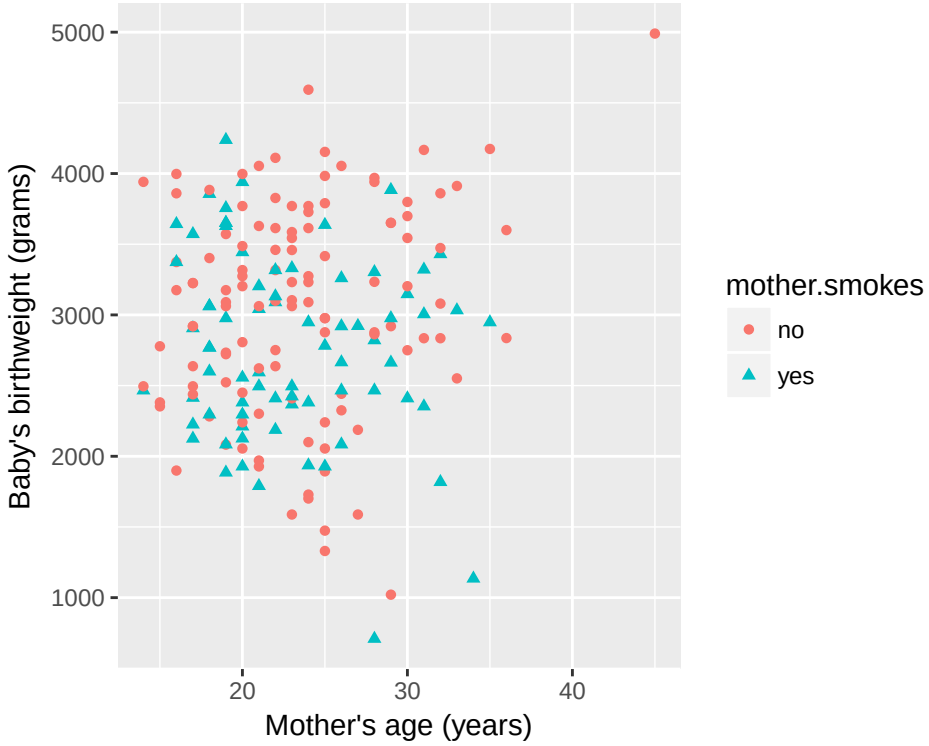



```
qplot(x=mother.age, y=birthwt.grams, data=birthwt)  # using qplot from ggplot2
```



Remember how it took us some effort last time to add color coding, use different plotting characters, and add a legend? Here's the `qplot` call that does it all in one simple line.

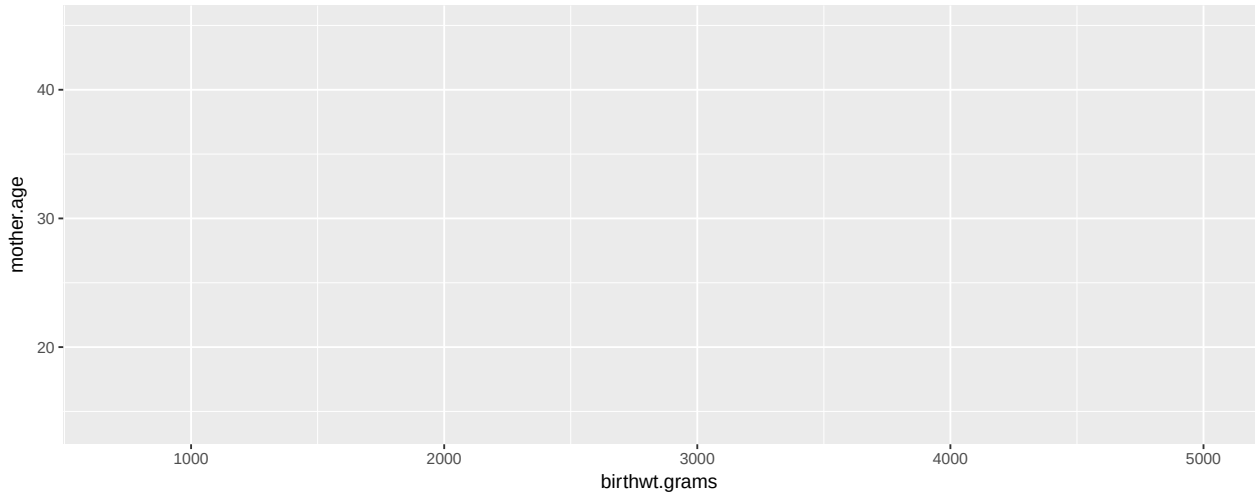
```
qplot(x=mother.age, y=birthwt.grams, data=birthwt,
      color = mother.smokes,
      shape = mother.smokes,
      xlab = "Mother's age (years)",
      ylab = "Baby's birthweight (grams)"
      )
```



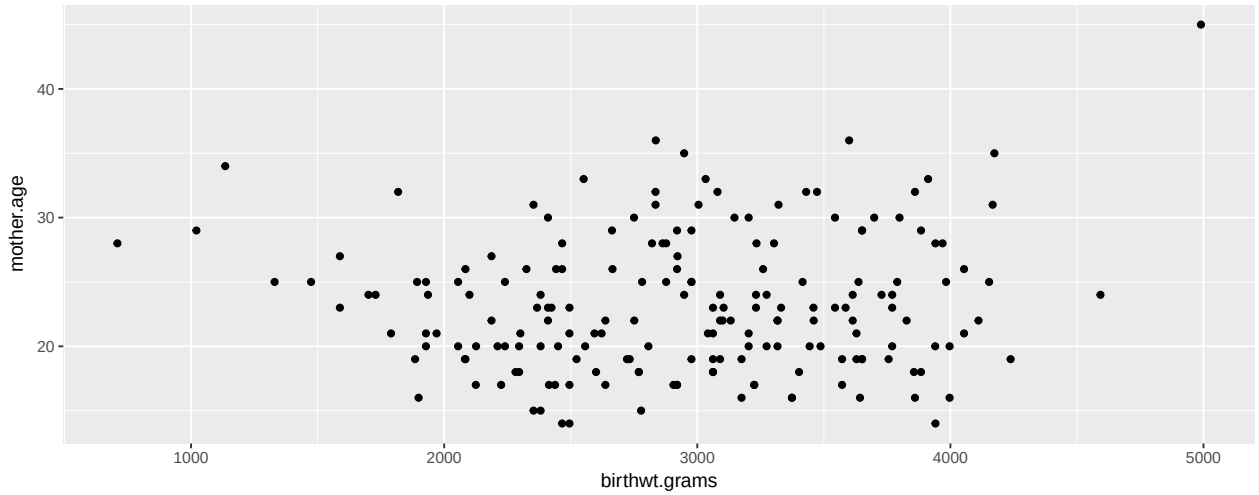
This way you won't run into problems of accidentally producing the wrong legend. The legend is produced based on the `colour` and `shape` argument that you pass in. (Note: `color` and `colour` have the same effect.)

```
ggplot(data=birthwt, aes(x=birthwt.grams, y=mother.age))
```

ggplot function




```
birthwt.plot <- ggplot(data=birthwt, aes(x=birthwt.grams, y=mother.age))
birthwt.plot + geom_point()
```



Let's take a step back and try to understand the ggplot syntax.

1) The first thing we did was to define a graphics object, `birthwt.plot`. This definition told R that we're using the `birthwt` data, and that we want to display `carat` on the x-axis, and `price` on the y-axis.

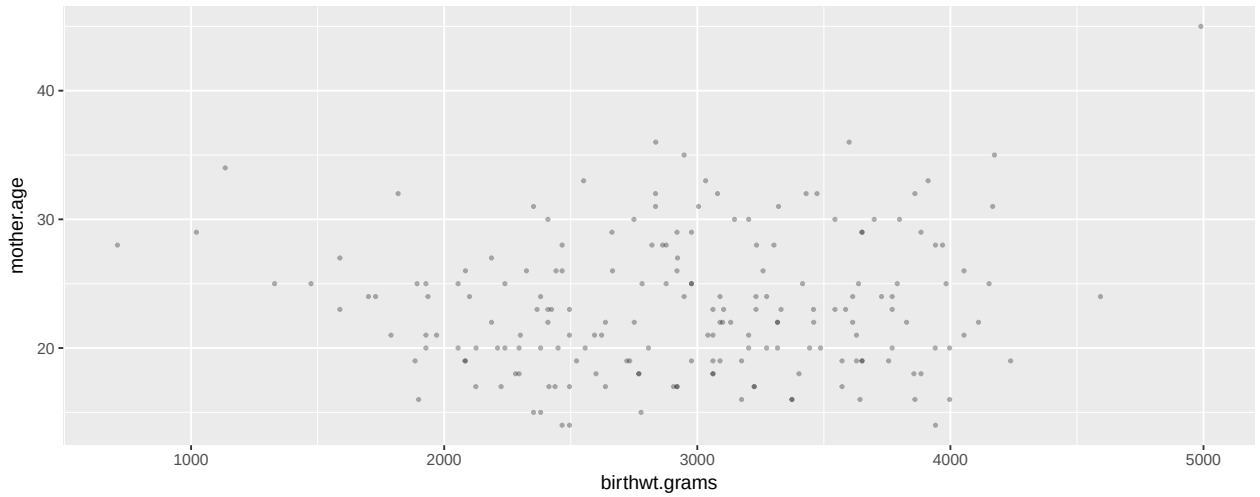
2) We then called `birthwt.plot + geom_point()` to get a scatterplot.

The arguments passed to `aes()` are called **mappings**. Mappings specify what variables are used for what purpose. When you use `geom_point()` in the second line, it pulls `x`, `y`, `colour`, `size`, etc., from the **mappings** specified in the `ggplot()` command.

You can also specify some arguments to `geom_point` directly if you want to specify them for each plot separately instead of pre-specifying a default.

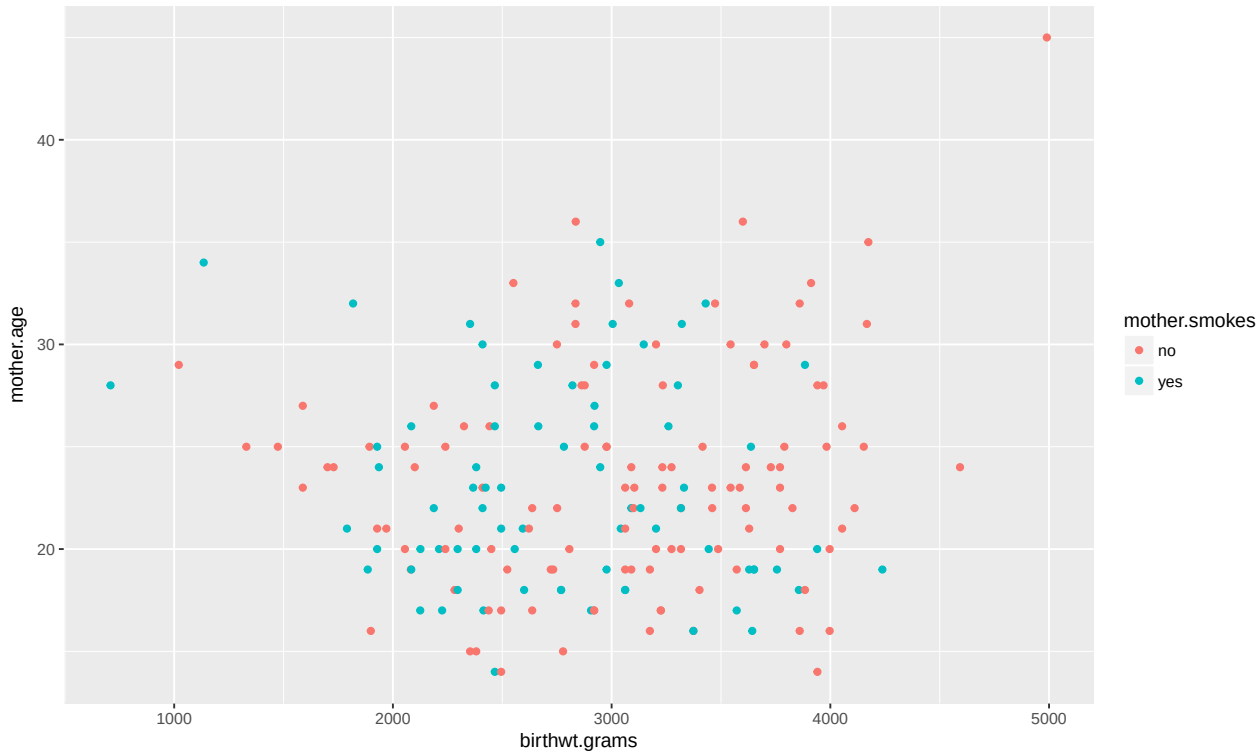
Here we shrink the points to a smaller size, and use the `alpha` argument to make the points transparent.

```
birthwt.plot + geom_point(size = 0.7, alpha = 0.3)
```



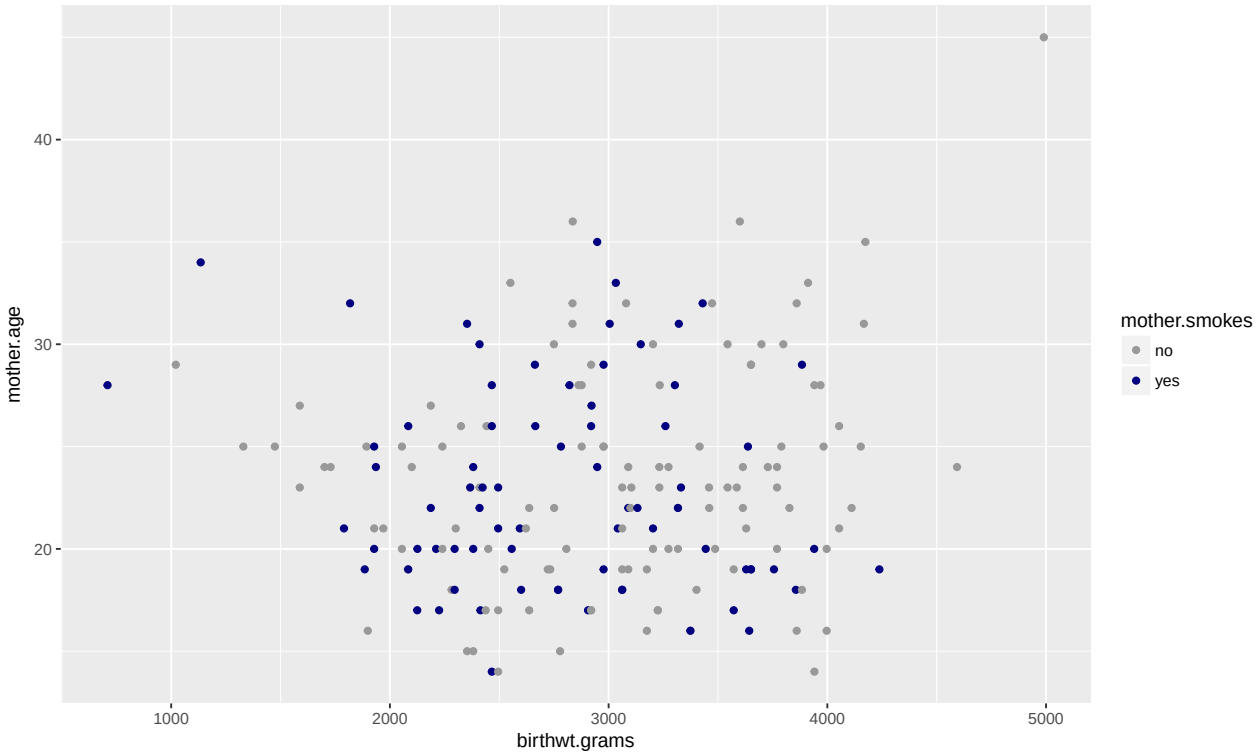
If we wanted to let point color depend on the color indicator of the diamond, we could do so in the following way.

```
birthwt.plot <- ggplot(data=birthwt, aes(x=birthwt.grams, y=mother.age, colour = mother.smokes))
birthwt.plot + geom_point()
```



We can change colors by specifying a different color palette. Here's how we can switch to the `cbPalette` we saw last class.

```
cPalette <- c("#999999", "navyblue")
birthwt.plot <- ggplot(data=birthwt, aes(x=birthwt.grams, y=mother.age, colour = mother.smokes))
birthwt.plot + geom_point() + scale_colour_manual(values=cPalette)
```

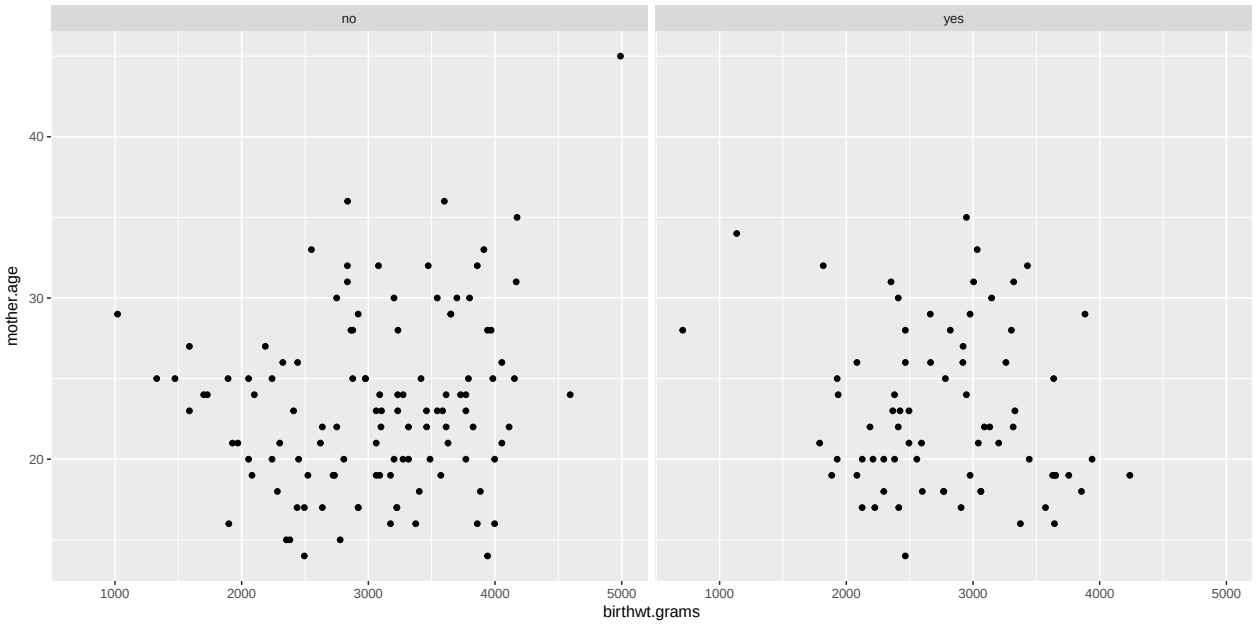


To make the scatterplot look more typical, we can sometimes switch to logarithmic coordinate axis spacing.

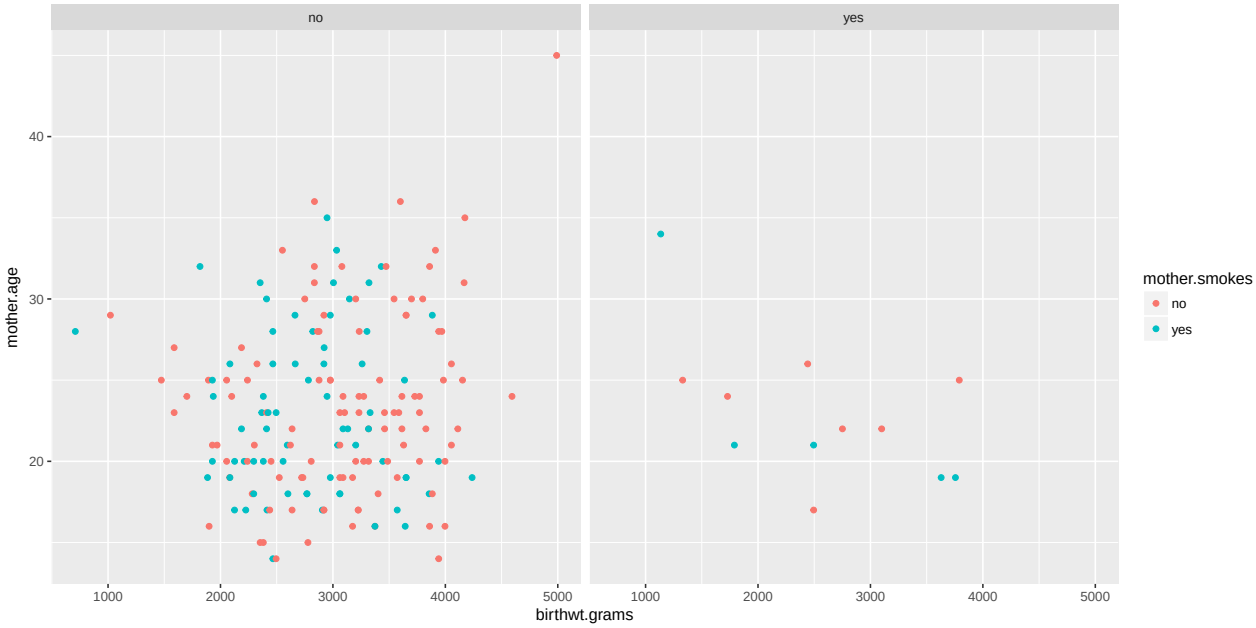
```
birthwt.plot + geom_point() +
  coord_trans(x = "log10", y = "log10")
```

Conditional plots We can create plots showing the relationship between variables across different values of a factor. For instance, here's a scatterplot showing how birth weight varies with hypertension, conditioned on color. It's created using the `facet_wrap(~ factor1 + factor2 + ... + factorn)` command.

```
birthwt.plot <- ggplot(data=birthwt, aes(x=birthwt.grams, y=mother.age))
birthwt.plot + geom_point() + facet_wrap(~ mother.smokes)
```




```
birthwt.plot <- ggplot(data=birthwt, aes(x=birthwt.grams, y=mother.age, colour = mother.smokes))
birthwt.plot + geom_point() + facet_wrap(~ hypertension)
```

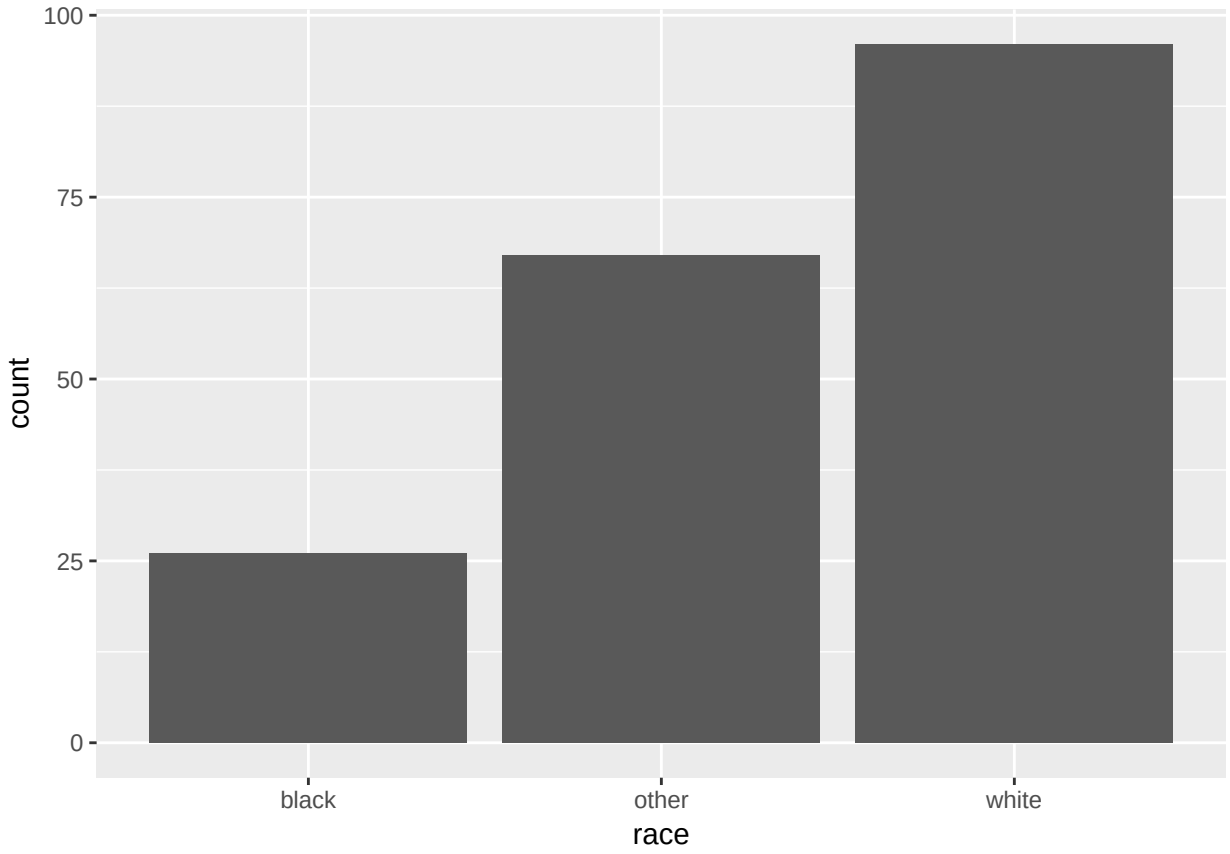


`ggplot` can create a lot of different kinds of plots, just like lattice. Here are some examples.

Function	Description
`geom_point(...)`	Points, i.e., scatterplot
`geom_bar(...)`	Bar chart
`geom_line(...)`	Line chart
`geom_boxplot(...)`	Boxplot
`geom_violin(...)`	Violin plot
`geom_density(...)`	Density plot with one variable
`geom_density2d(...)`	Density plot with two variables

Function	Description
`geom_histogram(...)`	Histogram

```
qplot(x = race, data = birthwt, geom = "bar")
```

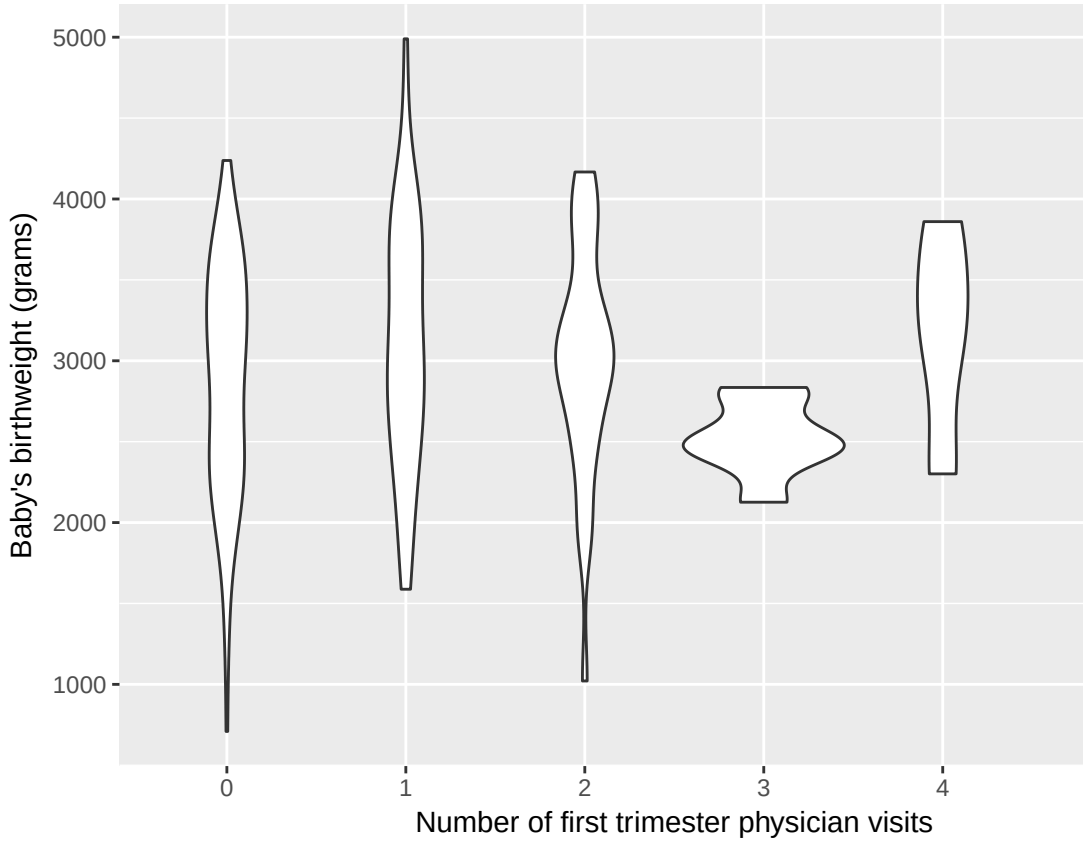


A bar chart

```
base.plot <- ggplot(birthwt, aes(x = mother.age)) +
  xlab("Mother's age")
base.plot + geom_histogram()
base.plot + geom_histogram(aes(fill = race), alpha = 0.5)

base.plot <- ggplot(birthwt, aes(x = mother.age)) +
  xlab("Mother's age")
base.plot + geom_density()
base.plot + geom_density(aes(fill = race), alpha = 0.5)
```

Histograms and density plots

```
base.plot <- ggplot(birthwt, aes(x = as.factor(physician.visits), y = birthwt.grams)) +
  xlab("Number of first trimester physician visits") +
  ylab("Baby's birthweight (grams)")
# Violin plot
base.plot + geom_violin()
```



Box plots and violin plots

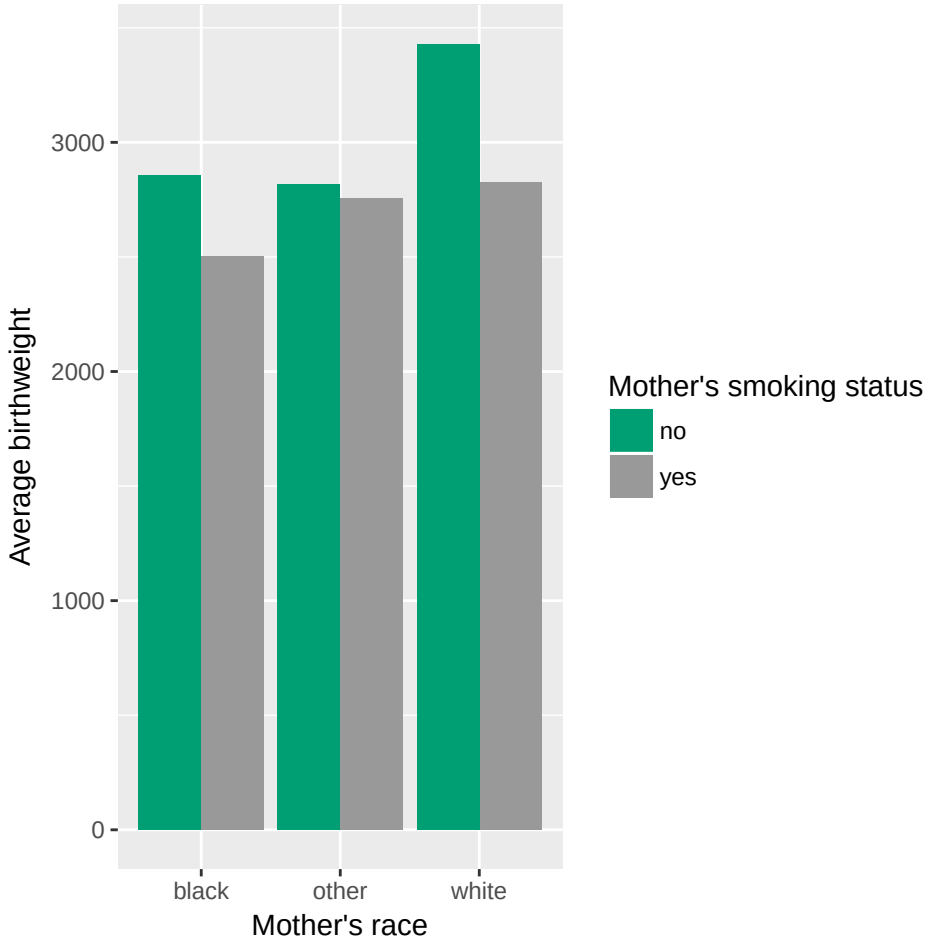
Visualizing means Previously we calculated the following table:

```
bwt.summary <- aggregate(birthwt.grams ~ race + mother.smokes, data = birthwt, FUN = mean) # aggregate
bwt.summary
```

```
##    race mother.smokes birthwt.grams
## 1 black            no      2854.500
## 2 other            no      2815.782
## 3 white            no      3428.750
## 4 black           yes      2504.000
## 5 other           yes      2757.167
## 6 white           yes      2826.846
```

We can plot this table in a nice bar chart as follows:

```r
# Define basic aesthetic parameters
p.bwt <- ggplot(data = bwt.summary, aes(y = birthwt.grams, x = race, fill = mother.smokes))

# Pick colors for the bars
bwt.colors <- c("#009E73", "#999999")

# Display barchart
p.bwt + geom_bar(stat = "identity", position = "dodge") +
  ylab("Average birthweight") +
  xlab("Mother's race") +
  guides(fill = guide_legend(title = "Mother's smoking status")) +
  scale_fill_manual(values=bwt.colors)
```



Does the association between birthweight and mother's age depend on smoking status? We previously ran the following command to calculate the correlation between mother's ages and baby birthweights.

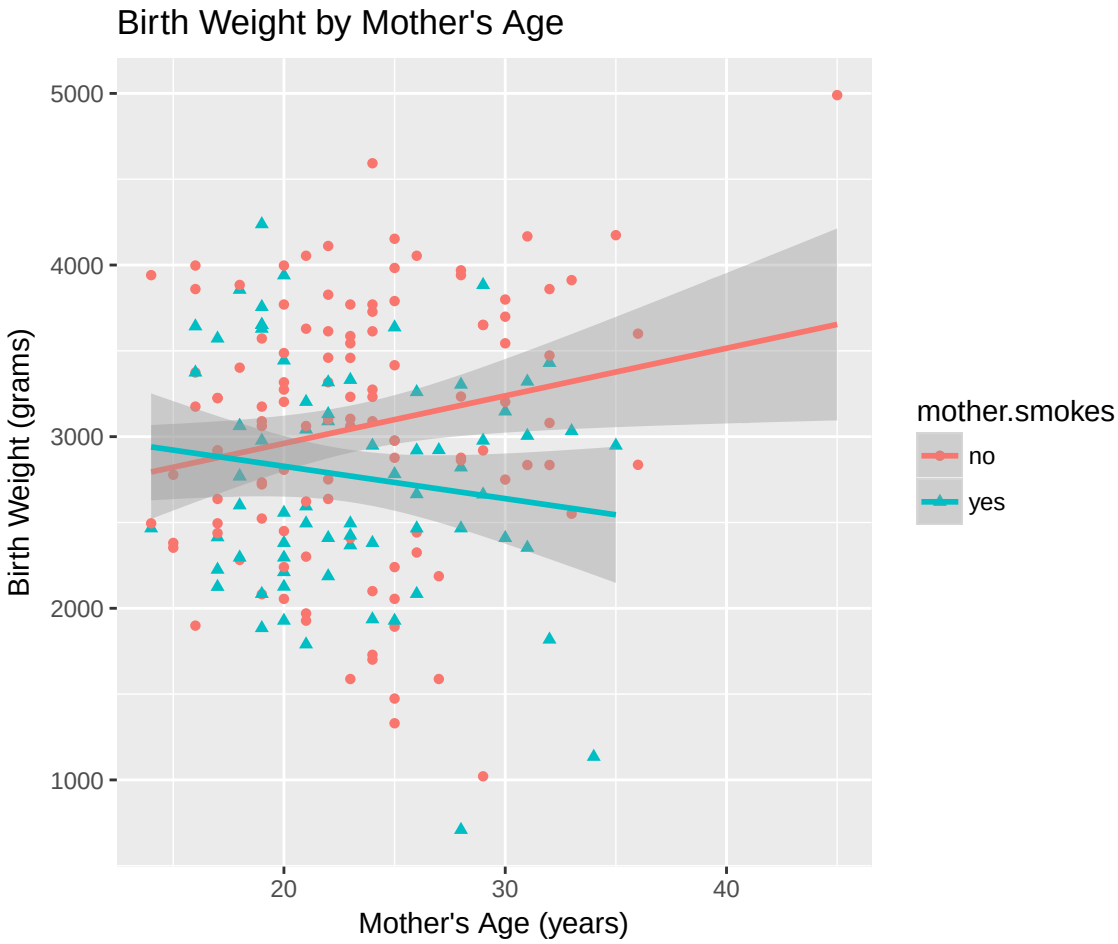
```r
by(data = birthwt[c("birthwt.grams", "mother.age")],
   INDICES = birthwt["mother.smokes"],
   FUN = function(x) {cor(x[,1], x[,2])})
```

```
## mother.smokes: no
```

```
## [1] 0.2014558
## ------------------------------------------------------------
## mother.smokes: yes
## [1] -0.1441649
```

Here's a visualization of our data that allows us to see what's going on.

```
ggplot(birthwt, aes(x=mother.age, y=birthwt.grams, shape=mother.smokes, color=mother.smokes)) +
  geom_point() + # Adds points (scatterplot)
  geom_smooth(method = "lm") + # Adds regression lines
  ylab("Birth Weight (grams)") + # Changes y-axis label
  xlab("Mother's Age (years)") + # Changes x-axis label
  ggtitle("Birth Weight by Mother's Age") # Changes plot title
```



CENG 3516 STATISTICAL COMPUTING

Lecture 4 - Functions

Dr. Eralp DOGU

Mar 10, 2017

Agenda

- Writing functions in R
- If-else statements

Basics of lists

A list is a **data structure** that can be used to store **different kinds** of data

- Recall: a vector is a data structure for storing *similar kinds of data*
- To better understand the difference, consider the following example.

```
my.vector.1 <- c("Ela", 22, FALSE) # (name, weight, is.male)
my.vector.1
```

```
## [1] "Ela"   "22"    "FALSE"
```

```
typeof(my.vector.1)  # All the elements are now character strings!
```

```
## [1] "character"
```

Lists vs. vectors

```
#Info about Lara
my.vector.2 <- c(FALSE, TRUE, 1) # (is.male, is.baby, age)
typeof(my.vector.2)
```

```
## [1] "double"
```

- Vectors expect elements to be all of the same type (e.g., `Boolean`, `numeric`, `character`)
- When data of different types are put into a vector, the R converts everything to a common type

Lists

- To store data of different types in the same object, we use lists
- Simple way to build lists: use `list()` function

```r
my.list <- list("Ela", 22, TRUE)
my.list
```

```
## [[1]]
## [1] "Ela"
##
## [[2]]
## [1] 22
##
## [[3]]
## [1] TRUE
```

```r
sapply(my.list, typeof)
```

```
## [1] "character" "double"    "logical"
```

Named elements

```r
kid.1 <- list(name="Ela", weight=22, is.male=FALSE)
kid.1
```

```
## $name
## [1] "Ela"
##
## $weight
## [1] 22
##
## $is.male
## [1] FALSE
```

Referencing elements of a list (similar to data frames)

```r
kid.1$name # Get "name" element (returns a string)
```

```
## [1] "Ela"
```

```r
kid.1[["name"]] # Get "name" element (returns a string)
```

```
## [1] "Ela"
```

```r
kid.1["name"] # Get "name" slice (returns a sub-list)
```

```
## $name
## [1] "Ela"
```

```r
c(typeof(kid.1$name), typeof(kid.1["name"]))
```

```
## [1] "character" "list"
```

Functions

- We have used a lot of built-in functions: `mean()`, `subset()`, `plot()`, `read.table()`...
- An important part of programming and data analysis is to write custom functions
- Functions help make code **modular**
- Functions make debugging easier
- Remember: this entire class is about applying *functions* to *data*

What is a function?

> A function is a machine that turns **input objects** (arguments) into an **output object** (return value) according to a definite rule.

- Let's look at a really simple function

```r
addOne <- function(x) {
  x + 1
}
```

- `x` is the **argument** or **input**
- The function **output** is the input `x` incremented by 1

```r
addOne(12)
```

```
## [1] 13
```

More interesting example

- Here's a function that returns a % given a numerator, denominator, and desired number of decimal values

```r
calculatePercentage <- function(x, y, d) {
  decimal <- x / y  # Calculate decimal value
  round(100 * decimal, d)  # Convert to % and round to d digits
}

calculatePercentage(27, 80, 1)
```

```
## [1] 33.8
```

- If you're calculating several %'s for your report, you should use this kind of function instead of repeatedly copying and pasting code

Function returning a list

- Here's a function that takes a person's full name (FirstName LastName), weight in lb and height in inches and converts it into a list with the person's first name, person's last name, weight in kg, height in m, and BMI.

```
createPatientRecord <- function(full.name, weight, height) {
  name.list <- strsplit(full.name, split=" ")[[1]]
  first.name <- name.list[1]
  last.name <- name.list[2]
  weight.in.kg <- weight / 2.2
  height.in.m <- height * 0.0254
  bmi <- weight.in.kg / (height.in.m ^ 2)
  list(first.name=first.name, last.name=last.name, weight=weight.in.kg, height=height.in.m,
       bmi=bmi)
}
```

Trying out the function

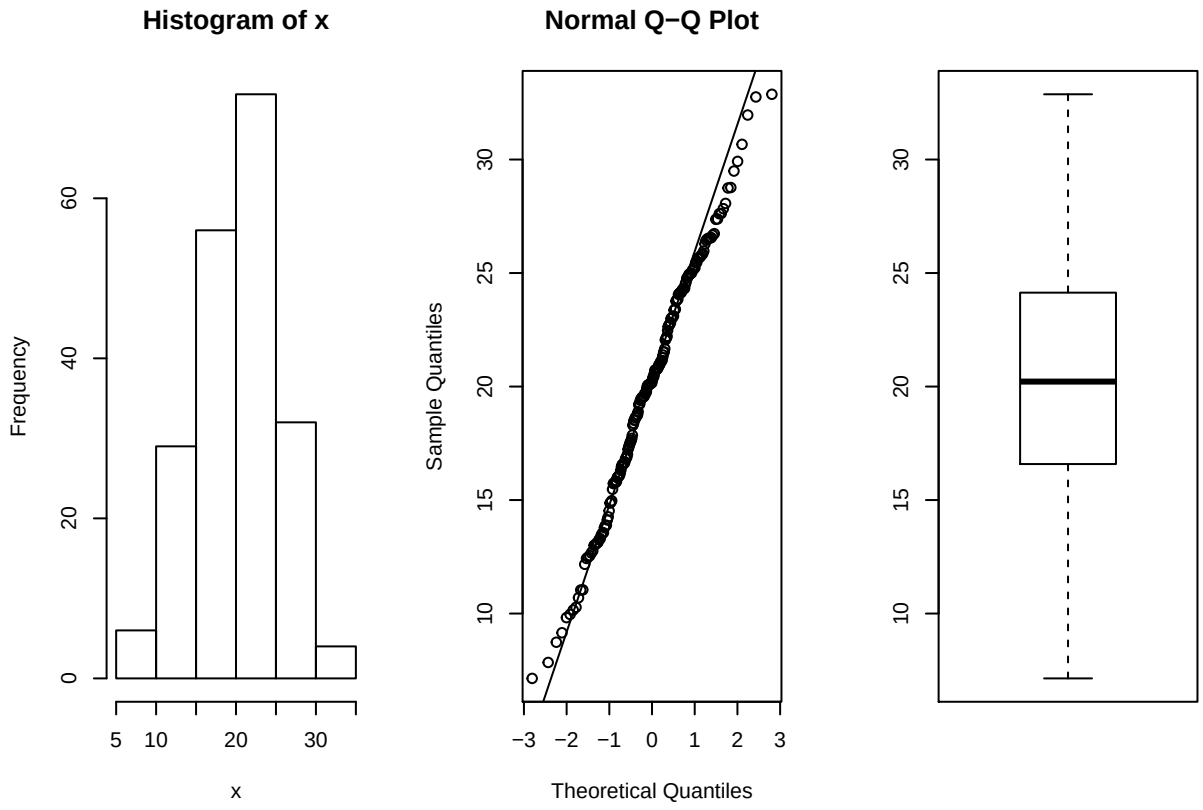
```
createPatientRecord("Ahmet CAN", 185, 12 * 6 + 1)
```

```
## $first.name
## [1] "Ahmet"
##
## $last.name
## [1] "CAN"
##
## $weight
## [1] 84.09091
##
## $height
## [1] 1.8542
##
## $bmi
## [1] 24.45884
```

Another example: 5 number summary

- Calculate mean, 10% trimmed mean, median, geometric and harmonic means of a random variable

```
fiveaverages <- function(x) {
  c(average=mean(x), trimmed=mean(x, trim=0.10), median=median(x), geometricmean= prod(x)^(1/length(x))
}
x <- rnorm(200, mean=20, sd=5) # Vector of 200 kids with weight mean 5 and weight sd 2
par(mfrow=c(1,3))
hist(x)
qqnorm(x)
qqline(x)
boxplot(x)
```



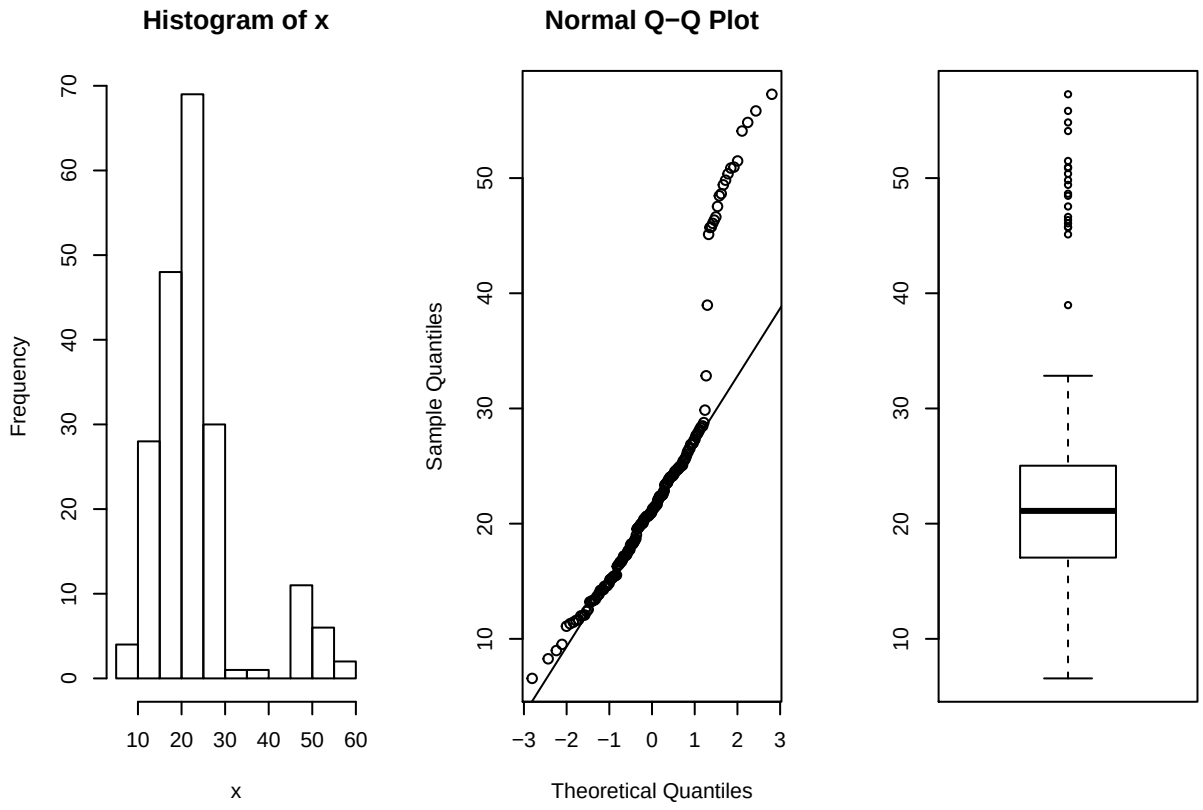

```
fiveaverages(x)
```

```
##       average       trimmed        median geometricmean  harmonicmean
##      20.17758      20.28015      20.21798      19.45330      18.63186
```

Add outliers in the dataset

- Calculate mean, 10% trimmed mean, median, geometric and harmonic means of a random variable

```
x <- c(rnorm(180, mean=20, sd=5),rnorm(20, mean=50, sd=5)) # Vector of 200 kids with outliers
par(mfrow=c(1,3))
hist(x)
qqnorm(x)
qqline(x)
boxplot(x)
```




```
fiveaverages(x)
```

```
##       average       trimmed        median geometricmean  harmonicmean
##      23.05607      21.22100      21.10763      21.35104      19.92103
```

If-else statements

- Oftentimes we want our code to have different effects depending on the features of the input
- Example: Calculating a student's letter grade
- If grade >= 90, assign A
- Otherwise, if grade >= 80, assign B
- Otherwise, if grade >= 70, assign C
- In all other cases, assign F
- To code this up, we use if-else statements

If-else Example: Letter grades

```
calculateLetterGrade <- function(x) {
  if(x >= 90) {
    grade <- "A"
  } else if(x >= 80) {
```

```
    grade <- "B"
  } else if(x >= 70) {
    grade <- "C"
  } else {
    grade <- "F"
  }
  grade
}

course.grades <- c(92, 78, 87, 91, 62)
sapply(course.grades, FUN=calculateLetterGrade)
```

```
## [1] "A" "C" "B" "A" "F"
```

return()

- In the previous examples we specified the output simply by writing the output variable as the last line of the function
- More explicitly, we can use the `return()` function

```
addOne <- function(x) {
  return(x + 1)
}

addOne(12)
```

```
## [1] 13
```

- We will generally avoid the `return()` function, but you can use it if necessary or if it makes writing a particular function easier.

Write your own function for Median Absolute Deviation

```
MADcalculator <- function(x) {

  ...
  ...
  ...


}
```

Write your own function for correlation analysis

- Example:
- Draw a scatter plot for x and y
- Calculate Pearson corelation coefficient
- Automatically evaluate if the relationship is strong, weak, etc...

```
CorrelationAnalyzer <- function(x,y) {

  ...
  ...
  ...


}
```

Write your own function for replacing missing values with the 10% trimmed mean

```
MissingValueReplacer <- function(x,y) {

  ...
  ...
  ...


}
```

CENG 3516 STATISTICAL COMPUTING

Lecture 5 - Probability Distributions in R

Dr. Eralp DOGU

Mar 17, 2017

Agenda

- A probability distribution describes how the values of a random variable is distributed. For example, the collection of all possible outcomes of a sequence of coin tossing is known to follow the binomial distribution. Whereas the means of sufficiently large samples of a data population are known to resemble the normal distribution. Since the characteristics of these theoretical distributions are well understood, they can be used to make statistical inferences on the entire data population as a whole.
- In the following tutorial, we demonstrate how to compute a few well-known probability distributions that occurs frequently in statistical studies.

Getting started:

- Every distribution that R handles has four functions. There is a root name, for example, the root name for the normal distribution is norm. This root is prefixed by one of the letters
- p for "probability", the cumulative distribution function (c.d.f.)
- q for "quantile", the inverse c.d.f.
- d for "density", the density function (p.f. or p.d.f.)
- r for "random", a random variable having the specified distribution
- For the normal distribution, these functions are pnorm, qnorm, dnorm, and rnorm. For the binomial distribution, these functions are **pbinom**, **qbinom**, **dbinom**, and **rbinom**. And so forth. For a continuous distribution (like the normal), the most useful functions for doing problems involving probability calculations are the **p** and **q** functions (c.d.f. and inverse c.d.f.), because the the density (p.d.f.) calculated by the **d** function can only be used to calculate probabilities via integrals and R doesn't do integrals.
- For a discrete distribution (like the binomial), the **d** function calculates the density (p.f.), which in this case is a probability

f(x) = P(X = x) and hence is useful in calculating probabilities. R has functions to handle many probability distributions.

Binomial Distribution

- The binomial distribution is a discrete probability distribution. It describes the outcome of n independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure. If the probability of a successful trial is p, then the probability of having x successful outcomes in an experiment of n independent trials.

- **Problem:** Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of having four or less correct answers if a student attempts to answer every question at random.
- **Solution:** Since only one out of five possible answers is correct, the probability of answering a question correctly by random is 1/5=0.2. We can find the probability of having exactly 4 correct answers by random attempts as follows.

```
dbinom(4, size=12, prob=0.2)
```

```
## [1] 0.1328756
```

$P(X = 4) = 0.1328756$

-To find the probability of having four or less correct answers by random attempts, we apply the function dbinom with x = 0,. . . ,4.

```
dbinom(0, size=12, prob=0.2) +
+ dbinom(1, size=12, prob=0.2) +
+ dbinom(2, size=12, prob=0.2) +
+ dbinom(3, size=12, prob=0.2) +
+ dbinom(4, size=12, prob=0.2)
```

```
## [1] 0.9274445
```

```
# Alternatively, we can use the cumulative probability function for binomial distribution pbinom.

pbinom(4, size=12, prob=0.2)
```

```
## [1] 0.9274445
```

$P(X <= 4) = 0.9274445$

Poisson Distribution

- The Poisson distribution is the probability distribution of independent event occurrences in an interval. If **Lambda** is the mean occurrence per interval, then the probability of having x occurrences within a given interval.
- **Problem:**If there are twelve cars crossing a bridge per minute on average, find the probability of having seventeen or more cars crossing the bridge in a particular minute.
- **Solution:** The probability of having sixteen or less cars crossing the bridge in a particular minute is given by the function ppois.

```
#The probability of having sixteen or less cars crossing the bridge in a particular minute is given by
ppois(16, lambda=12)   # lower tail
```

```
## [1] 0.898709
```

```r
#Hence the probability of having seventeen or more cars crossing the bridge in a minute is in the upper
ppois(16, lambda=12, lower=FALSE)    # upper tail
```

```
## [1] 0.101291
```

Normal Distribution

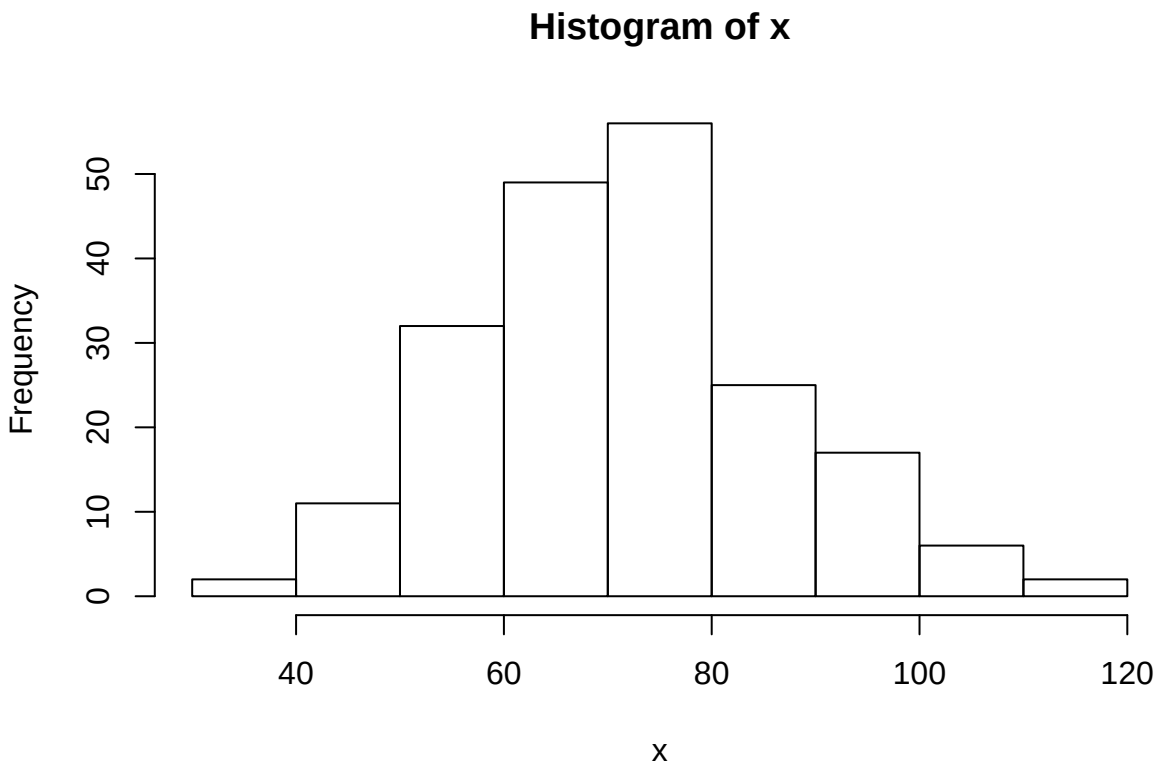
- The normal distribution is defined by the following probability density function, where **mean** is the population mean and **sd** is the standard deviation.
- **Problem:** Assume that the test scores of a college entrance exam fits a normal distribution. Furthermore, the mean test score is 72, and the standard deviation is 15.2. What is the percentage of students scoring 84 or more in the exam?
- **Solution:**We apply the function pnorm of the normal distribution with mean 72 and standard deviation 15.2. Since we are looking for the percentage of students scoring higher than 84, we are interested in the upper tail of the normal distribution.

```r
pnorm(84, mean=72, sd=15.2, lower.tail=FALSE)
```

```
## [1] 0.2149176
```

- **Problem:**Generate n=20 observation with the same parameters?
- **Solution:**We apply the function rnorm of the normal distribution with mean 72 and standard deviation 15.2.

```r
x<-rnorm(200, mean=72, sd=15.2)
hist(x)
```



- **Problem:**Suppose you are interested in the highest 10% of the students. What would be the minimum grade for this group?
- **Solution:**We apply the function qnorm of the normal distribution with mean 72 and standard deviation 15.2.

```
qnorm(0.90, mean=72, sd=15.2)
```

```
## [1] 91.47958
```

Exponential Distribution

- The exponential distribution describes the arrival time of a randomly recurring independent event sequence. If **mean** is the mean waiting time for the next event recurrence.
- **Problem:**Suppose the mean checkout time of a supermarket cashier is three minutes. Find the probability of a customer checkout being completed by the cashier in less than two minutes.
- **Solution:**The checkout processing rate is equals to one divided by the mean checkout completion time. Hence the processing rate is 1/3 checkouts per minute. We then apply the function pexp of the exponential distribution with rate=1/3.

```
pexp(2, rate=1/3)
```

```
## [1] 0.4865829
```

Other resources

- http://www.stat.umn.edu/geyer/old/5101/rlook.html
- http://www.cyclismo.org/tutorial/R/probability.html

CENG 3516 STATISTICAL COMPUTING

Lecture 6 - Confidence Intervals and Hypothesis testing in R

Dr. Eralp DOGU

Mar 17, 2017

Agenda

- Plotting confidence intervals
- Hypothesis testing for one sample mean
- Hypothesis testing for differences in mean between two groups

Let's begin by loading the packages we'll need to get started

```
library(MASS)
library(plyr)
library(ggplot2)
```

Exploring the birthwt data

The first thing we'll do is to get the data into a nicer form.

```
# Rename the columns to have more descriptive names
colnames(birthwt) <- c("birthwt.below.2500", "mother.age", "mother.weight",
    "race", "mother.smokes", "previous.prem.labor", "hypertension", "uterine.irr",
    "physician.visits", "birthwt.grams")

# Transform variables to factors with descriptive levels
birthwt <- transform(birthwt,
            race = as.factor(mapvalues(race, c(1, 2, 3),
                             c("white","black", "other"))),
            mother.smokes = as.factor(mapvalues(mother.smokes,
                             c(0,1), c("no", "yes"))),
            hypertension = as.factor(mapvalues(hypertension,
                             c(0,1), c("no", "yes"))),
            uterine.irr = as.factor(mapvalues(uterine.irr,
                             c(0,1), c("no", "yes")))
            )
```

Over the past lectures we created various tables and graphics to help us better understand the data. Our focus for today is to run confidence intervals and hypothesis tests to assess whether the trends we observed last time are statistically significant.

Let's start with one sample confidence intervals and extend it to multiple sample cases. First we compute mean and standard deviation for birthweight grams and find the confidence intervals for this variable.

```
mean(birthwt$birthwt.grams)
```

```
## [1] 2944.587
```

```
sd(birthwt$birthwt.grams)
```

```
## [1] 729.2143
```

Confidence Intervals for Mean To find confidence intervals, we can create a simple function or we can use the `t.test()` function.

```
#########Standard deviation is known#########################
norm.interval = function(data, variance, conf.level = 0.95) {
   z = qnorm((1 - conf.level)/2, lower.tail = FALSE)
   xbar = mean(data)
   sdx = sqrt(variance/length(data))
   c(xbar - z * sdx, xbar + z * sdx)
}
birthwt.var<-800^2
norm.interval(birthwt$birthwt.grams, birthwt.var)
```

```
## [1] 2830.534 3058.640
```

```
#################################################################
########Standard deviation is unknown#########################
birthwt.CI <- t.test(birthwt$birthwt.grams)$conf.int
birthwt.CI
```

```
## [1] 2839.952 3049.222
## attr(,"conf.level")
## [1] 0.95
```

```
birthwt.CI <- t.test(birthwt$birthwt.grams, conf.level = 0.9)$conf.int
birthwt.CI
```

```
## [1] 2856.908 3032.267
## attr(,"conf.level")
## [1] 0.9
```

After finding the confidence interval we might want to test a hypothesis for the mean.

One sample t-test via t.test() Test whether birthwt.grams mean is different than 2000 grams Ho:mu=2000 Ha:mu is not 2000

or

Ho:mu=2000 Ha:mu $<$ 2000

```r
birthwt.t.test<- t.test(birthwt$birthwt.grams, alternative = c("two.sided"), mu = 2000, conf.level = 0.9
birthwt.t.test
```

```
##
##  One Sample t-test
##
## data:  birthwt$birthwt.grams
## t = 17.808, df = 188, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 2000
## 95 percent confidence interval:
##  2839.952 3049.222
## sample estimates:
## mean of x
##  2944.587
```

```r
birthwt.t.test<- t.test(birthwt$birthwt.grams, alternative = c("less"), mu = 2000, conf.level = 0.95)
birthwt.t.test
```

```
##
##  One Sample t-test
##
## data:  birthwt$birthwt.grams
## t = 17.808, df = 188, p-value = 1
## alternative hypothesis: true mean is less than 2000
## 95 percent confidence interval:
##      -Inf 3032.267
## sample estimates:
## mean of x
##  2944.587
```

```r
names(birthwt.t.test)
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```r
birthwt.t.test$p.value   # p-value
```

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

```r
birthwt.t.test$estimate  # group means
```

```
## mean of x
##  2944.587
```

```r
attr(birthwt.t.test$conf.int, "conf.level")  # confidence level
```

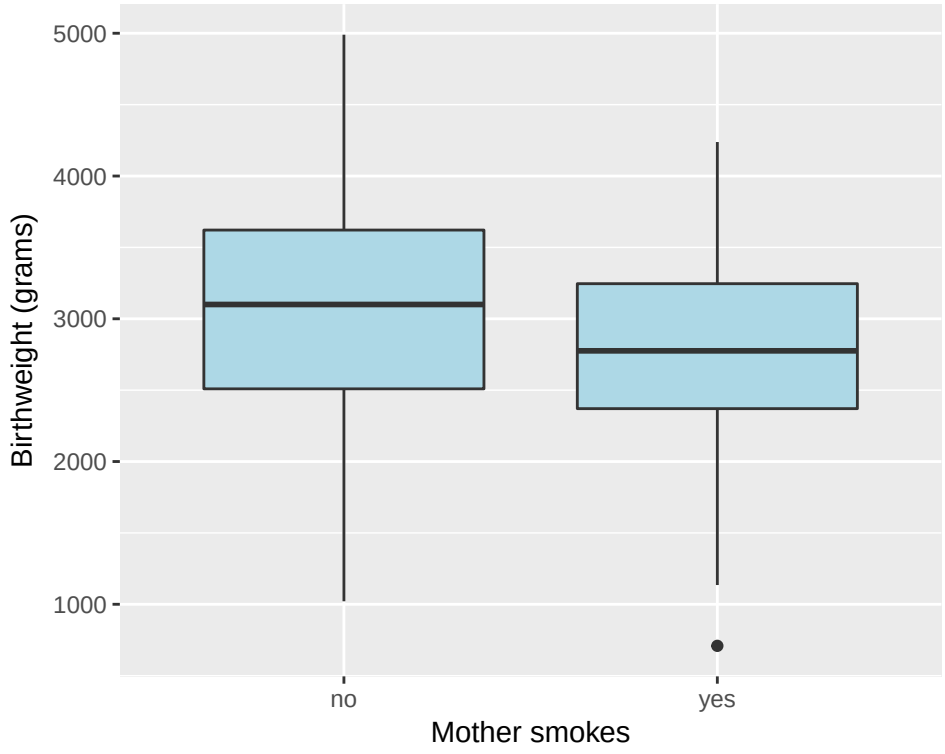
```
## [1] 0.95
```

Testing differences in means

One of the most common statistical tasks is to compare an outcome between two groups. The example here looks at comparing birth weight between smoking and non-smoking mothers.

To start, it always helps to plot things

```
# Create boxplot showing how birthwt.grams varies between
# smoking status
qplot(x = mother.smokes, y = birthwt.grams,
      geom = "boxplot", data = birthwt,
      xlab = "Mother smokes",
      ylab = "Birthweight (grams)",
      fill = I("lightblue"))
```



This plot suggests that smoking is associated with lower birth weight.

How can we assess whether this difference is statistically significant?

Let's compute a summary table

```
aggregate(birthwt.grams ~ mother.smokes,
          data = birthwt,
          FUN = function(x) {c(mean = mean(x), sd = sd(x))})
```

```
##   mother.smokes birthwt.grams.mean birthwt.grams.sd
## 1            no          3055.6957         752.6566
## 2           yes          2771.9189         659.6349
```

Note the way that the `FUN` argument was supplied in this `aggregate()` call. This function is never given a name. It's an example of an **anonymous function**. In addition, the function returns a vector with *named elements*, giving the mean and standard deviation of the vector that's passed to it.

```
aggregate(birthwt.grams ~ mother.smokes, data = birthwt,
          FUN = function(x) {c(mean = mean(x),
                               se = sd(x) / sqrt(length(x)))})
```

Two sample t-test via t.test()

```
##   mother.smokes birthwt.grams.mean birthwt.grams.se
## 1            no         3055.69565         70.18559
## 2           yes         2771.91892         76.68100
```

This difference is looking quite significant. To run a two-sample t-test, we can simple use the `t.test()` function.

```
birthwt.t.test <- t.test(birthwt.grams ~ mother.smokes, data = birthwt)
birthwt.t.test
```

```
##
##  Welch Two Sample t-test
##
## data:  birthwt.grams by mother.smokes
## t = 2.7299, df = 170.1, p-value = 0.007003
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   78.57486 488.97860
## sample estimates:
##  mean in group no mean in group yes
##          3055.696          2771.919
```

We see from this output that the difference is highly significant. The `t.test()` function also outputs a confidence interval for us.

Notice that the function returns a lot of information, and we can access this information element by element

```
names(birthwt.t.test)
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```
birthwt.t.test$p.value   # p-value
```

```
## [1] 0.007002548
```

```
birthwt.t.test$estimate  # group means
```

```
##  mean in group no mean in group yes
##          3055.696          2771.919
```

```
birthwt.t.test$conf.int  # confidence interval for difference
```

```
## [1]  78.57486 488.97860
## attr(,"conf.level")
## [1] 0.95
```

```
attr(birthwt.t.test$conf.int, "conf.level")  # confidence level
```

```
## [1] 0.95
```

`t.test()` accepts input in multiple forms. I like using the formula form whenever it's available, as I find it to be more easily interpretable. Here's another way of specifying the same information.

```
with(birthwt, t.test(x=birthwt.grams[mother.smokes=="no"],
                     y=birthwt.grams[mother.smokes=="yes"]))
```

```
##
##  Welch Two Sample t-test
##
## data:  birthwt.grams[mother.smokes == "no"] and birthwt.grams[mother.smokes == "yes"]
## t = 2.7299, df = 170.1, p-value = 0.007003
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   78.57486 488.97860
## sample estimates:
## mean of x mean of y
##  3055.696  2771.919
```

Specifying `x` and `y` arguments to the `t.test` function runs a t-test to check whether `x` and `y` have the same mean.

What is statistical significance testing doing? Here's a little simulation where we have two groups, a treatment groups and a control group. We're going to simulate observations from both groups. We'll run the simulation two ways.

- First simulation (Null case): the treatment has no effect
- Second simulation (Non-null case): the treatment on average increases outcome

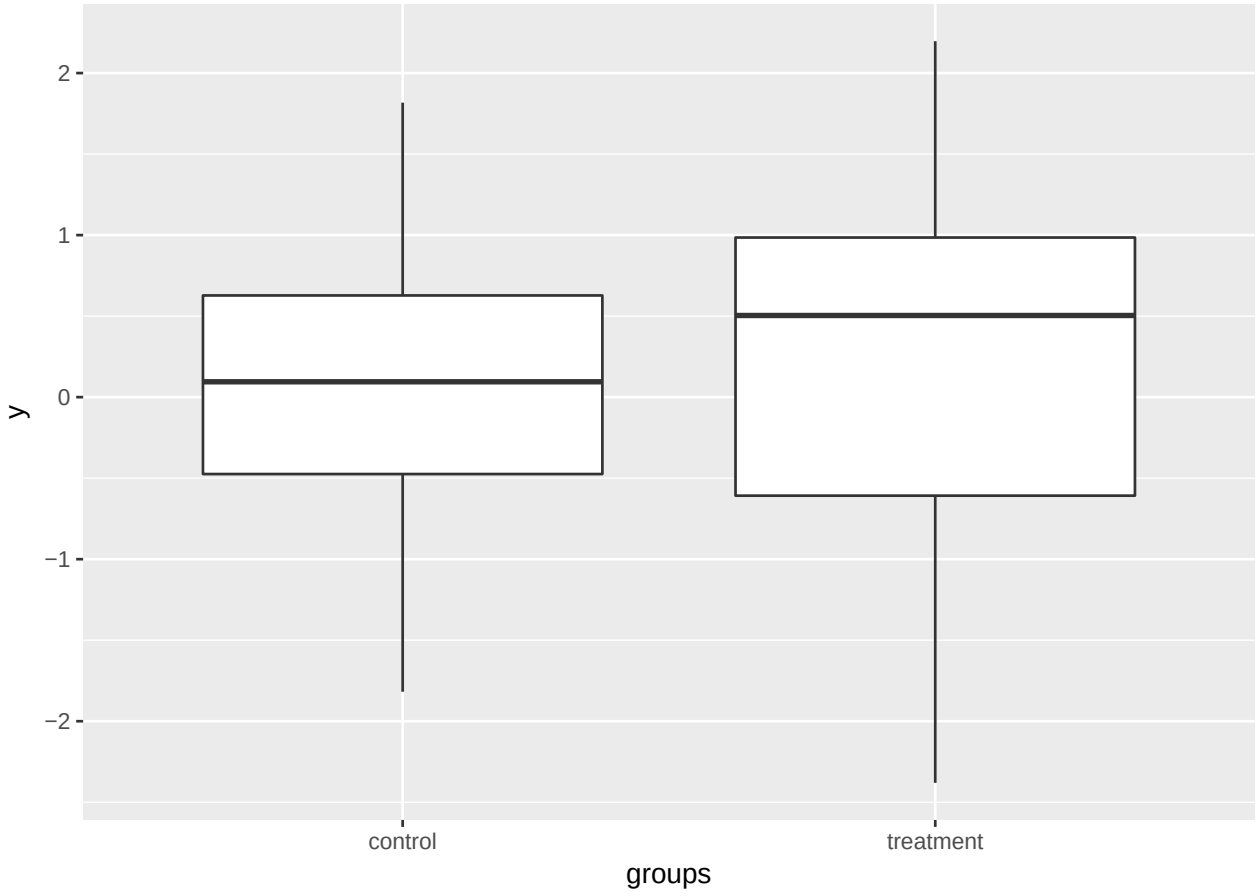
```
set.seed(12345)
# Function to generate data
generateSimulationData <- function(n1, n2, mean.shift) {
  y <- rnorm(n1 + n2) + c(rep(0, n1), rep(mean.shift, n2))
  groups <- c(rep("control", n1), rep("treatment", n2))
  data.frame(y = y, groups = groups)
}
```

Let's look at a single realization in the null setting.

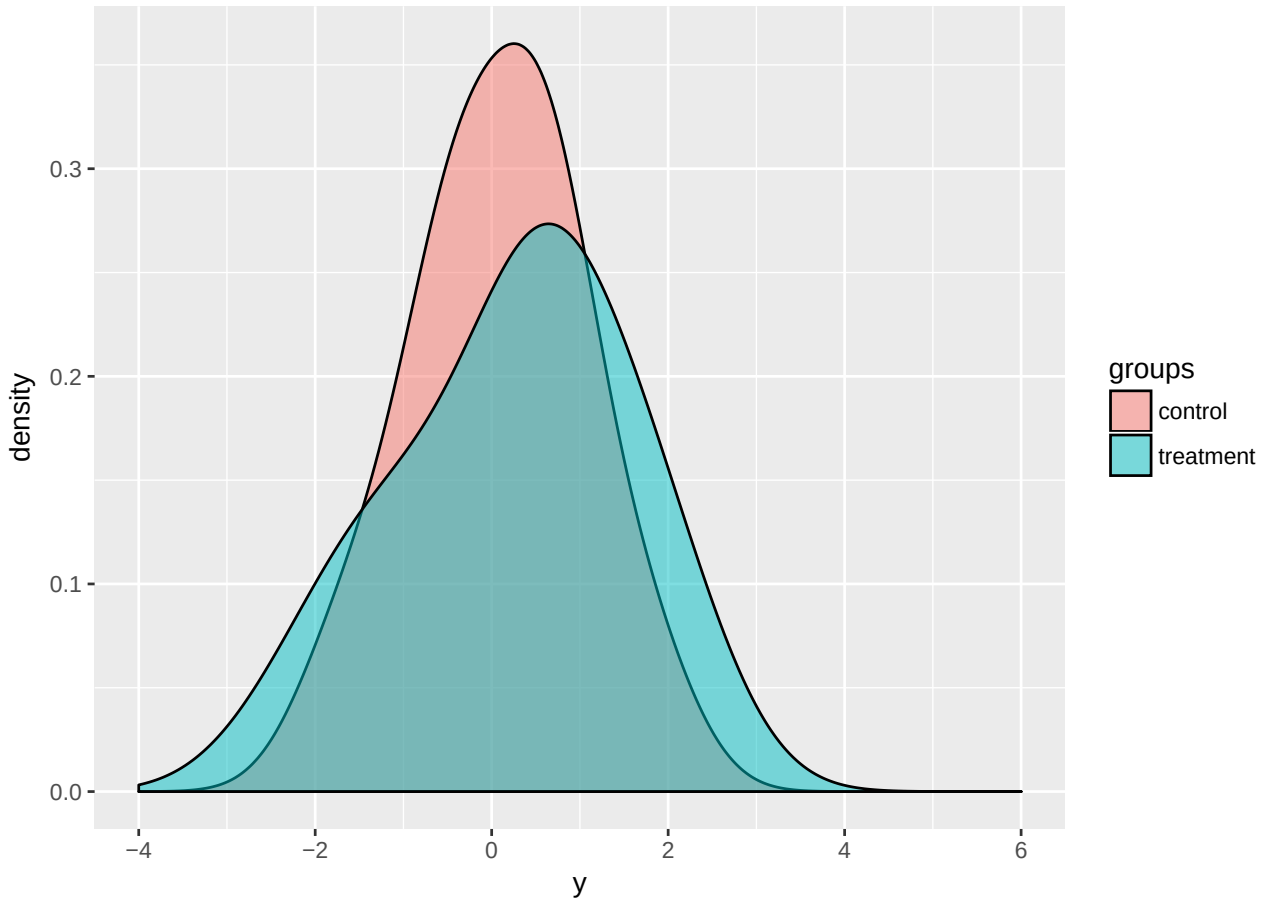
```r
n1 = 30
n2 = 40
# Observation, null case
obs.data <- generateSimulationData(n1 = n1, n2 = n2, mean.shift = 0)
head(obs.data)
```

```
##            y  groups
## 1  0.5855288 control
## 2  0.7094660 control
## 3 -0.1093033 control
## 4 -0.4534972 control
## 5  0.6058875 control
## 6 -1.8179560 control
```

```r
# Box plots
qplot(x = groups, y = y, data = obs.data, geom = "boxplot")
```




```r
# Density plots
qplot(fill = groups, x = y, data = obs.data, geom = "density",
      alpha = I(0.5),
      adjust = 1.5,
      xlim = c(-4, 6))
```




```
# t-test
t.test(y ~ groups, data = obs.data)
```

```
##
##  Welch Two Sample t-test
##
## data:  y by groups
## t = -0.61095, df = 67.998, p-value = 0.5433
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.6856053  0.3641889
## sample estimates:
##   mean in group control mean in group treatment
##              0.07880701              0.23951518
```

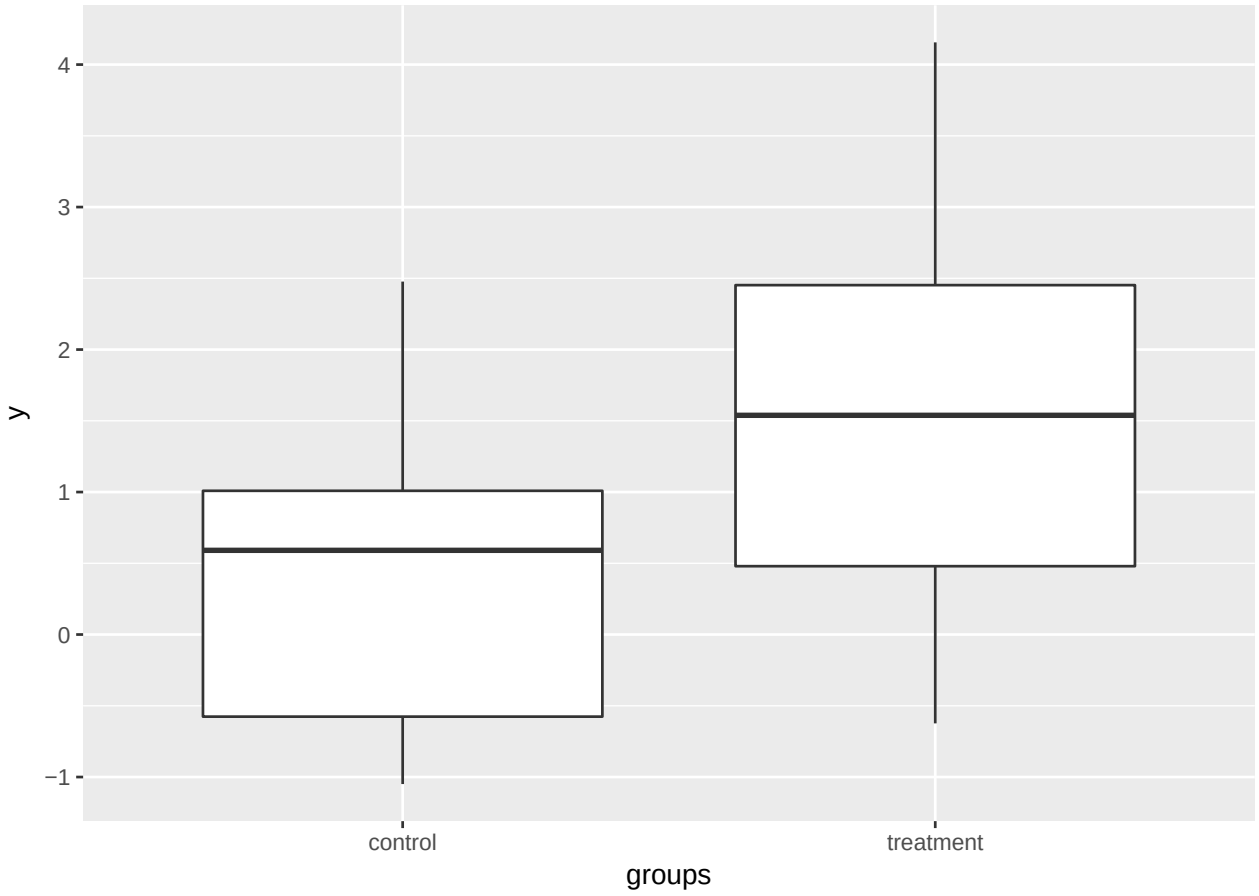
And here's what happens in a random realization in the non-null setting.

```
# Non-null case, very strong treatment effect
# Observation, null case
obs.data <- generateSimulationData(n1 = n1, n2 = n2, mean.shift = 1.5)
head(obs.data)
```

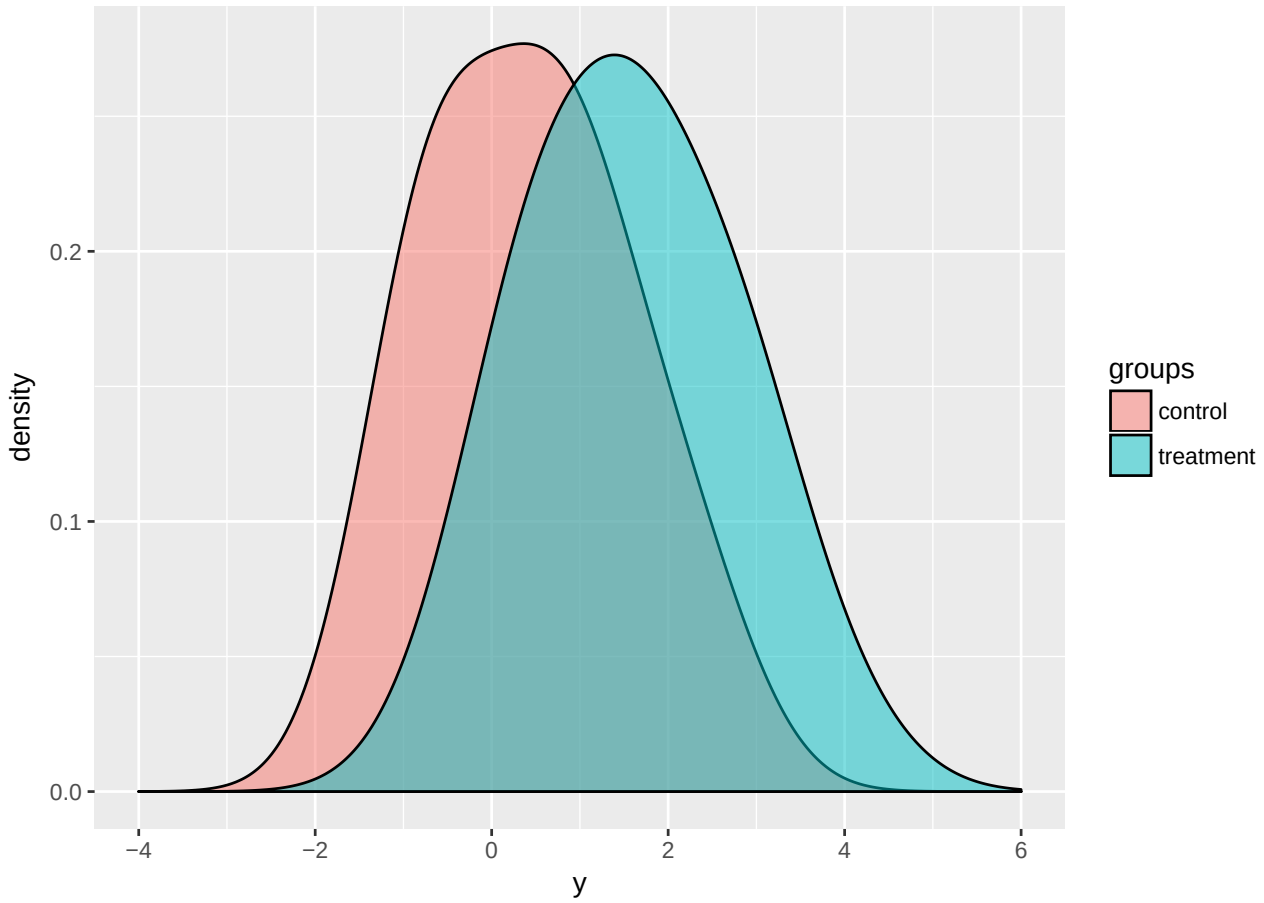
```
##             y  groups
## 1  0.05461558 control
```

```
## 2 -0.78464937 control
## 3 -1.04935282 control
## 4  2.33051196 control
## 5  1.40270538 control
## 6  0.94260085 control
```

```
# Box plots
qplot(x = groups, y = y, data = obs.data, geom = "boxplot")
```




```
# Density plots
qplot(fill = groups, x = y, data = obs.data, geom = "density",
      alpha = I(0.5),
      adjust = 1.5,
      xlim = c(-4, 6))
```




```
# t-test
t.test(y ~ groups, data = obs.data)
```

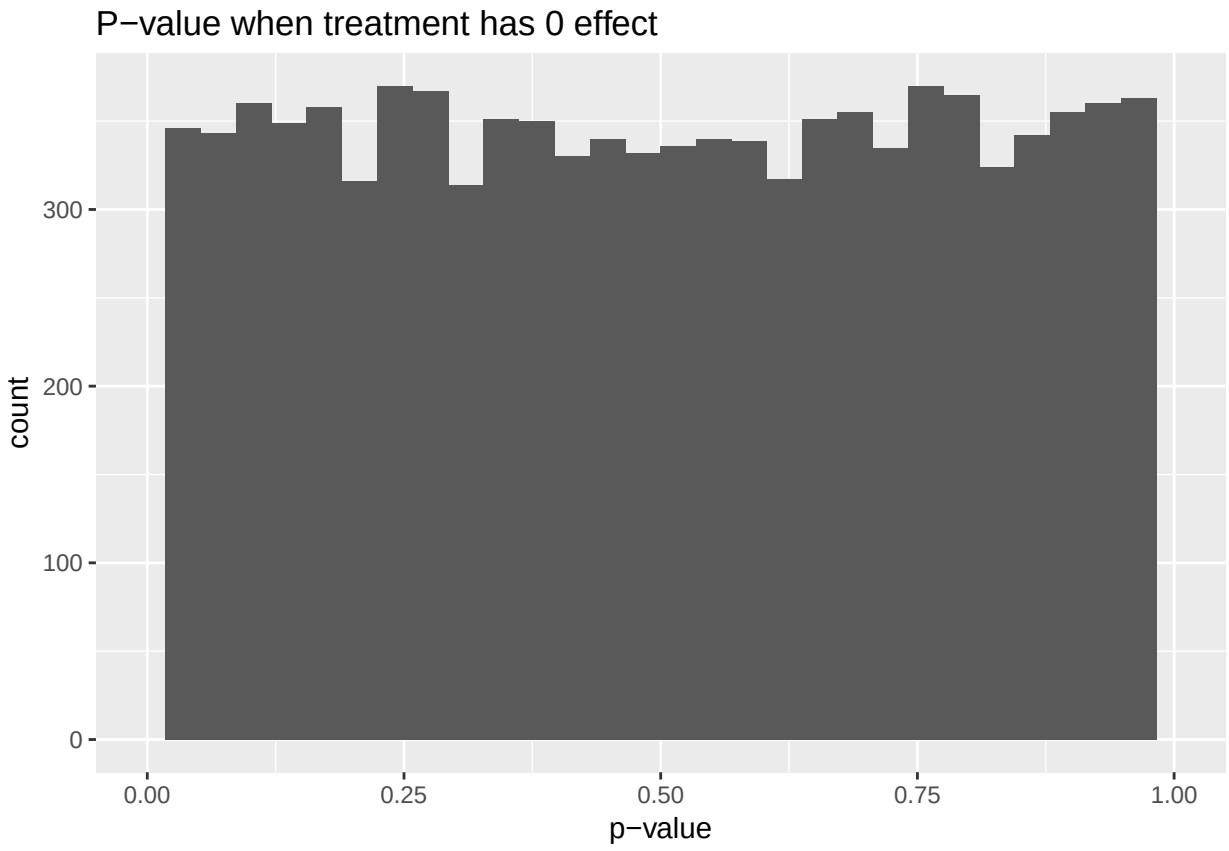
```
##
##  Welch Two Sample t-test
##
## data:  y by groups
## t = -4.3081, df = 64.785, p-value = 5.708e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.6911828 -0.6197985
## sample estimates:
##   mean in group control mean in group treatment
##               0.4191634               1.5746541
```

More interestingly, let's see what happens if we repeat our simulation 10000 times and look at the p-values. We'll use a moderate effect of 0.5 instead of the really strong effect of 1.5 in this simulation.

```
NUM_ITER <- 10000
pvals <- matrix(0, nrow = NUM_ITER, ncol = 2)
for(i in 1:NUM_ITER) {
  # Generate data
  obs.null <- generateSimulationData(n1 = n1, n2 = n2)
  obs.alt <- generateSimulationData(n1 = n1, n2 = n2, mean.shift = 0.5)
```

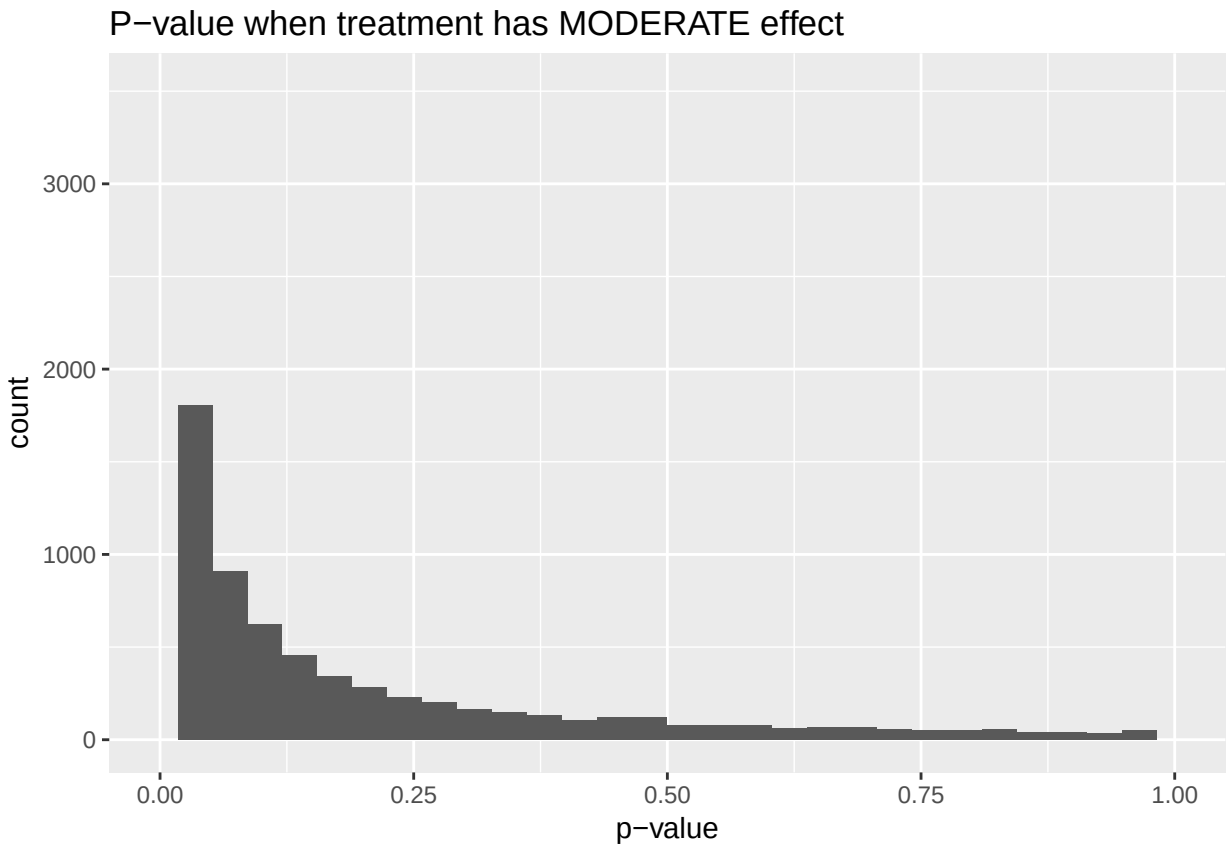
```
  # Record p-values
  pvals[i, 1] <- t.test(y ~ groups, data = obs.null)$p.value
  pvals[i, 2] <- t.test(y ~ groups, data = obs.alt)$p.value
}

pvals <- as.data.frame(pvals)
colnames(pvals) <- c("null", "nonnull")

# Plotting routine
qplot(x = null, data = pvals, xlab = "p-value",
      xlim = c(0, 1),
      main = "P-value when treatment has 0 effect")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qplot(x = nonnull, data = pvals, xlab = "p-value",
      xlim = c(0, 1),
      main = "P-value when treatment has MODERATE effect")
```

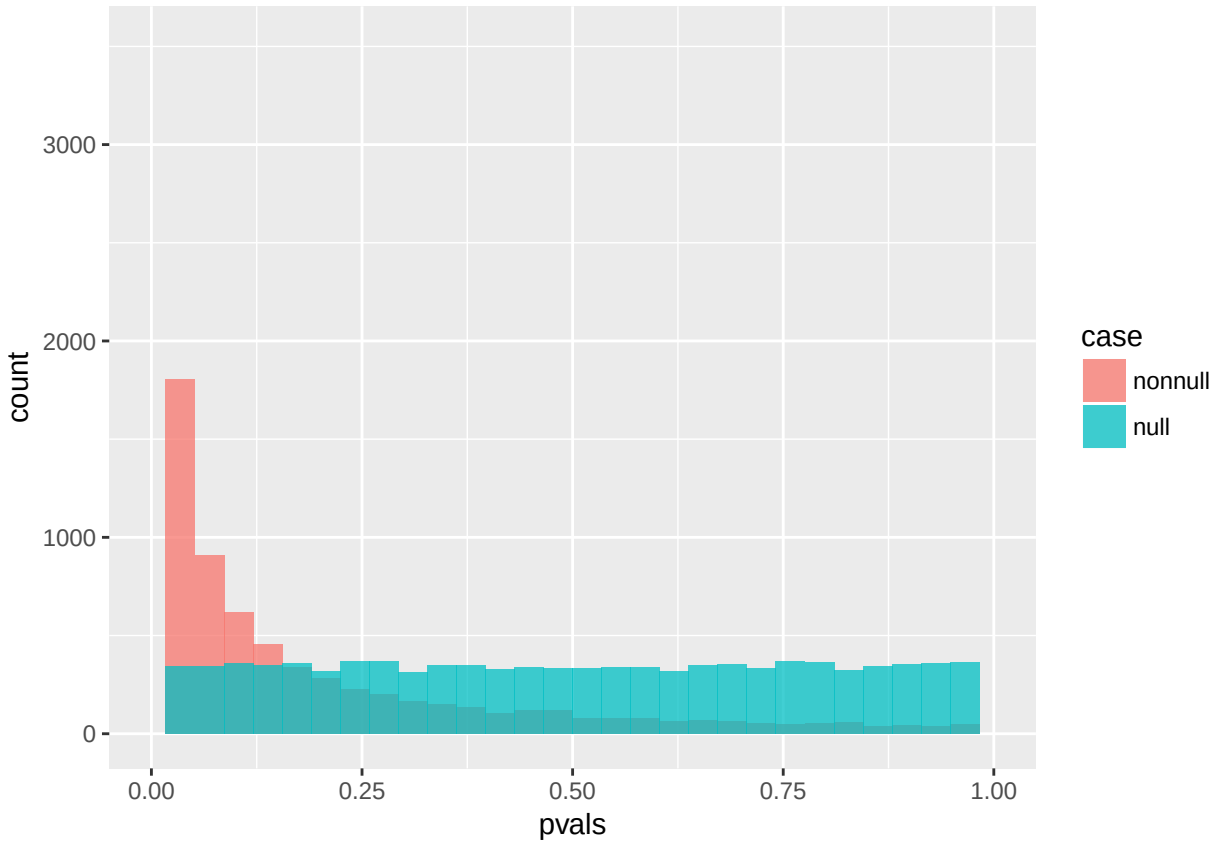
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Let's show both histograms on the same plot.

```
# Let's start by reshaping the data
# This approach isn't the best one, but it works well for this simple case
pvals.df <- data.frame(pvals = c(pvals$null, pvals$nonnull),
                       case = c(rep("null", NUM_ITER), rep("nonnull", NUM_ITER)))

# Plot
ggplot(data = pvals.df, aes(x = pvals, fill = case)) +
  geom_histogram(alpha=0.75, position="identity") +
  xlim(0,1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



What if sample is small and data are non-Gaussian? In your statistics classes you've been taught to approach the t-test with caution. If your data is highly skewed, you would need a very large sample size for the t-statistic to actually be t-distributed.

When it doubt, you can run a non-parametric test. Here's how we run a Mann-Whitney U test (aka Wilcoxon rank-sum test) using the `wilcox.test()` function.

```
# Formula specification
birthwt.wilcox.test <- wilcox.test(birthwt.grams ~ mother.smokes, data=birthwt, conf.int=TRUE)
birthwt.wilcox.test
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  birthwt.grams by mother.smokes
## W = 5249.5, p-value = 0.006768
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##   85.00004 512.00005
## sample estimates:
## difference in location
##               306.1846
```

```
# x,y specification
with(birthwt, wilcox.test(x=birthwt.grams[mother.smokes=="no"],
                          y=birthwt.grams[mother.smokes=="yes"]))
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  birthwt.grams[mother.smokes == "no"] and birthwt.grams[mother.smokes == "yes"]
## W = 5249.5, p-value = 0.006768
## alternative hypothesis: true location shift is not equal to 0
```

In general, hypothesis tests in R return an object of class `htest` which has similar attributes to what we saw in the t-test.

```
class(birthwt.wilcox.test)
```

```
## [1] "htest"
```

Here's a summary of the attributes:

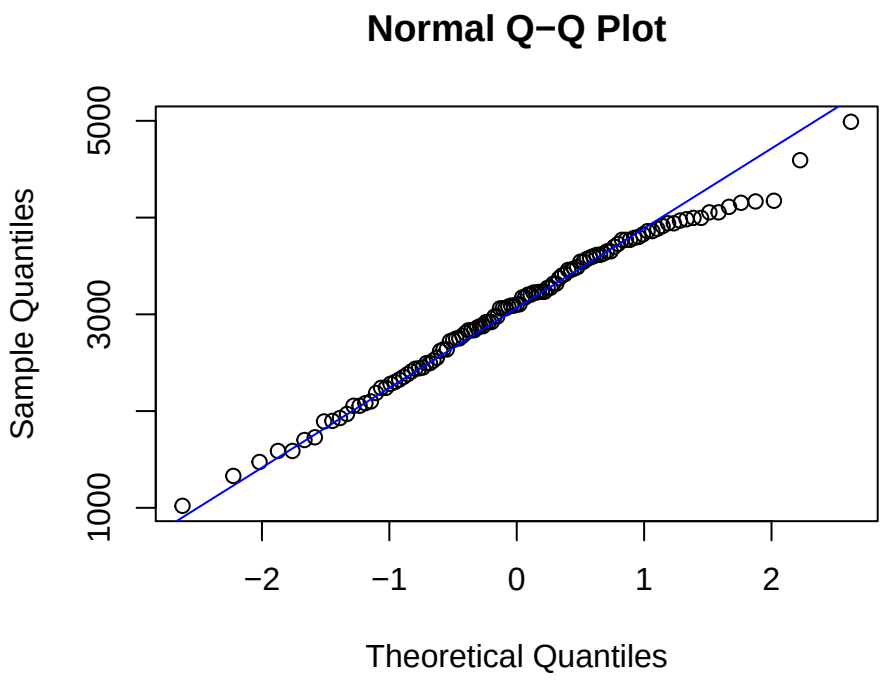
name	description
statistic	the value of the test statistic with a name describing it.
parameter	the parameter(s) for the exact distribution of the test statistic.
p.value	the p-value for the test.
null.value	the location parameter mu.
alternative	a character string describing the alternative hypothesis
method	the type of test applied.
data.name	a character string giving the names of the data.
conf.int	a confidence interval for the location parameter. (Only present if argument conf.int = TRUE.)
estimate	an estimate of the location parameter. (Only present if argument conf.int = TRUE.)

Is the data normal?

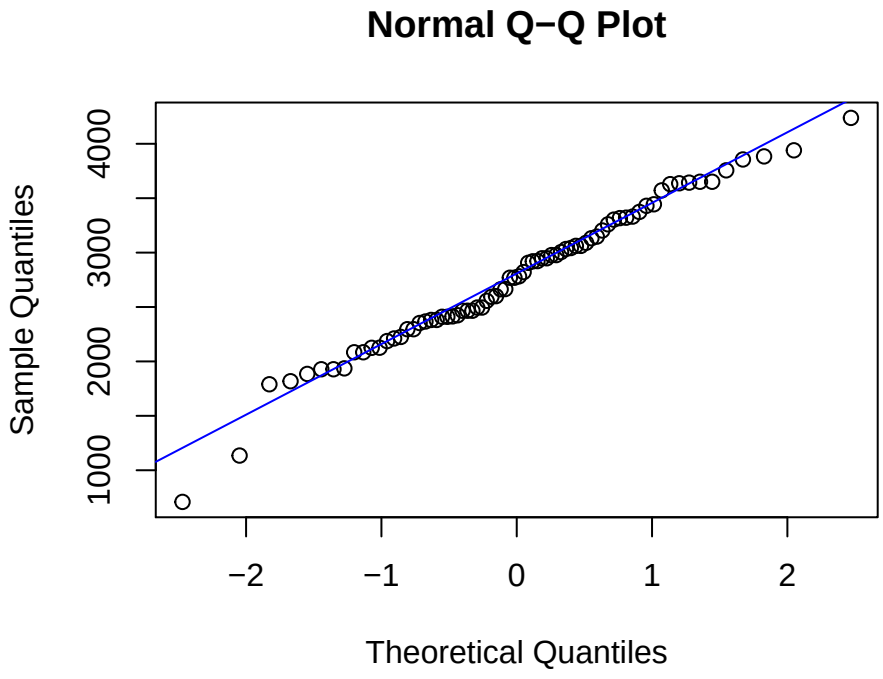
I would recommend using a non-parametric test when the data appears highly non-normal and the sample size is small. If you really want to stick to t-testing, it's good to know how to diagnose non-normality.

qq-plot The simplest thing to look at is a normal qq plot of the data. This is obtained using the `qqnorm()` function.

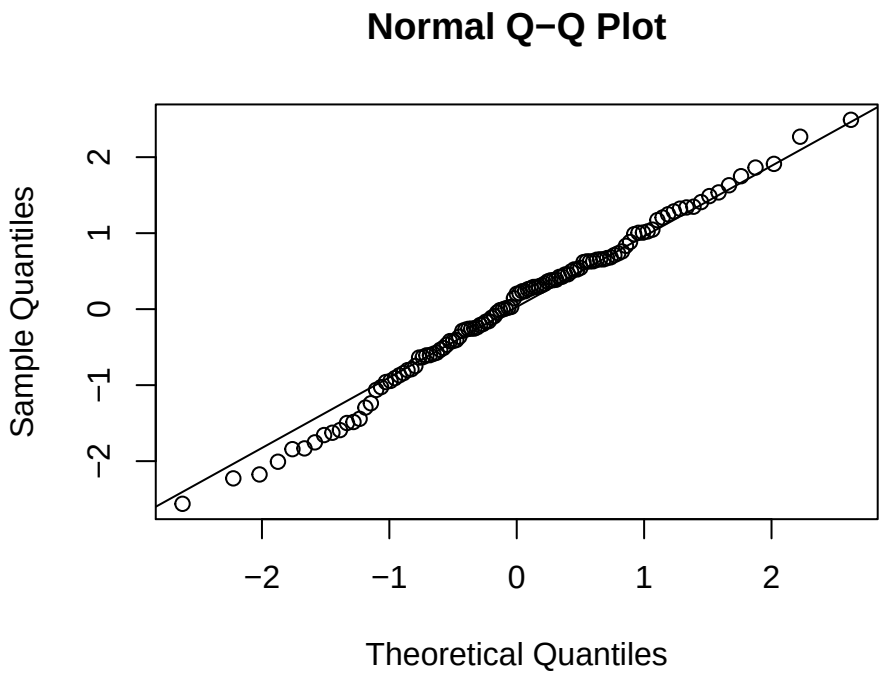
```
# qq plot
with(birthwt, qqnorm(birthwt.grams[mother.smokes=="no"]))
# add reference line
with(birthwt, qqline(birthwt.grams[mother.smokes=="no"], col = "blue"))
```




```
# qq plot
with(birthwt, qqnorm(birthwt.grams[mother.smokes=="yes"]))
# add reference line
with(birthwt, qqline(birthwt.grams[mother.smokes=="yes"], col = "blue"))
```



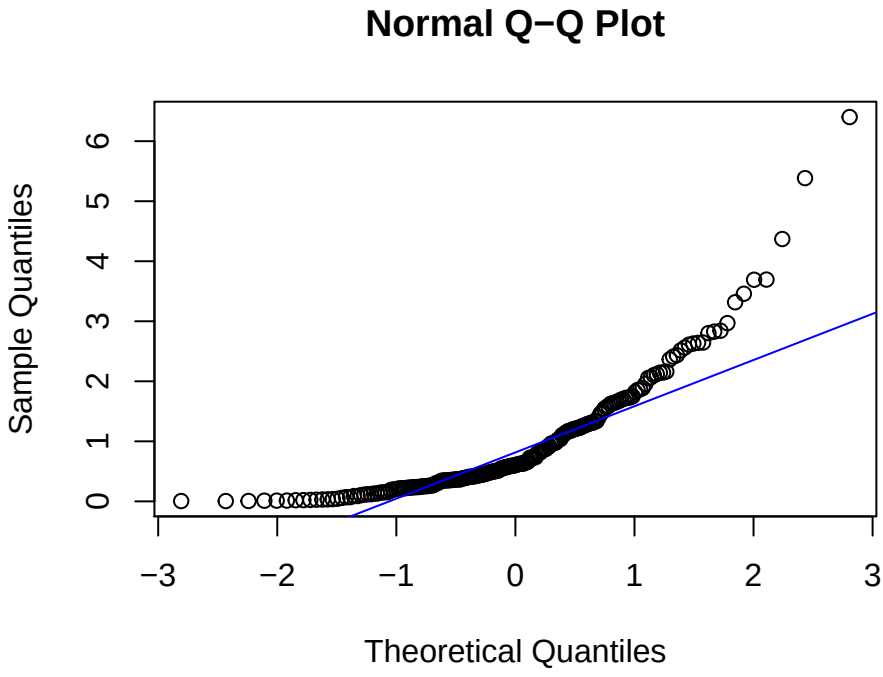

```
# qq plot
x <- rnorm(115); qqnorm(x); qqline(x)
```



If the data are exactly normal, you expect the points to lie on a straight line. The data we have here are pretty close to lying on a line.

Here's what we would see if the data were right-skewed

```
set.seed(12345)
fake.data <- rexp(200)
qqnorm(fake.data)
qqline(fake.data, col = "blue") # add line
```



If you construct a qqplot and it looks like this, you should be carefully, particularly if your sample size is small.

Tests for 2x2 tables

Here's an example of a 2 x 2 table that we might want to run a test on. This one looks at low birthweight broken down by mother's smoking status. You can think of it as another approach to the t-test problem, this time looking at indicators of low birth weight instead of the actual weights.

First, let's build our table using the `table()` function (we did this back in Lecture 5)

```
weight.smoke.tbl <- with(birthwt, table(birthwt.below.2500, mother.smokes))
weight.smoke.tbl
```

```
##                    mother.smokes
## birthwt.below.2500 no yes
##                  0 86  44
##                  1 29  30
```

We also previously calculated the odds ratio for this table, finding that it was approximately 2. This indicated that the odds of low birthweight double when the mother smokes.

To test for significance, we just need to pass our 2 x 2 table into the appropriate function. Here's the result of using fisher's exact test by calling `fisher.test`

```
birthwt.fisher.test <- fisher.test(weight.smoke.tbl)
birthwt.fisher.test
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  weight.smoke.tbl
## p-value = 0.03618
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.028780 3.964904
## sample estimates:
## odds ratio
##   2.014137
```

```
attributes(birthwt.fisher.test)
```

```
## $names
## [1] "p.value"     "conf.int"    "estimate"    "null.value"  "alternative"
## [6] "method"      "data.name"
##
## $class
## [1] "htest"
```

As when using the t-test, we find that there is a significant association between smoking an low birth weight.

You can also use the chi-squared test via the `chisq.test` function. This is the test that you may be more familiar with from your statistics class.

```r
chisq.test(weight.smoke.tbl)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  weight.smoke.tbl
## X-squared = 4.2359, df = 1, p-value = 0.03958
```

You get essentially the same answer by running the chi-squared test, but the output isn't as useful. In particular, you're not getting an estimate or confidence interval for the odds ratio. This is why I prefer `fisher.exact()` for testing 2 x 2 tables.

Tests for j x k tables Here's a small data set on party affiliation broken down by gender.

```r
# Manually enter the data
politics <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(politics) <- list(gender = c("F", "M"),
                    party = c("Democrat","Independent", "Republican"))

politics # display the data
```

```
##       party
## gender Democrat Independent Republican
##      F      762         327        468
##      M      484         239        477
```

We may be interested in asking whether men and women have different party affiliations.

The answer will be easier to guess at if we convert the rows to show proportions instead of counts. Here's one way of doing this.

```r
politics.prop <- apply(politics, 1, FUN = function(x) {x / sum(x)})
politics.prop
```

```
##              gender
## party                 F         M
##   Democrat    0.4894027 0.4033333
##   Independent 0.2100193 0.1991667
##   Republican  0.3005780 0.3975000
```

```r
colSums(politics.prop) # Check that columns sum to 1
```

```
## F M
## 1 1
```

```r
politics.prop <- t(politics.prop) # Transpose the table

# Fix dimnames
dimnames(politics.prop) <- list(gender = c("F", "M"),
                    party = c("Democrat","Independent", "Republican"))

# Output
politics.prop
```

```
##       party
## gender  Democrat Independent Republican
##      F 0.4894027   0.2100193   0.300578
##      M 0.4033333   0.1991667   0.397500
```

By looking at the table we see that Female are more likely to be Democrats and less likely to be Republicans.

We still want to know if this difference is significant. To assess this we can use the chi-squared test (on the counts table, not the proportions table!).

```
chisq.test(politics)
```

```
##
##  Pearson's Chi-squared test
##
## data:  politics
## X-squared = 30.07, df = 2, p-value = 2.954e-07
```

There isn't really a good one-number summary for general j x k tables the way there is for 2 x 2 tables. One thing that we may want to do at this stage is to ignore the Independent category and just look at the 2 x 2 table showing the counts for the Democrat and Republican categories.

```
politics.dem.rep <- politics[,c(1,3)]
politics.dem.rep
```

```
##       party
## gender Democrat Republican
##      F      762        468
##      M      484        477
```

```
# Run Fisher's exact test
fisher.test(politics.dem.rep)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  politics.dem.rep
## p-value = 6.806e-08
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.347341 1.910944
## sample estimates:
## odds ratio
##   1.604345
```

We see that women have significantly higher odds of being Democrat compared to men.

Plotting the table values with confidence

It may be useful to represent the data graphically. Here's one way of doing so with the `ggplot2` package. Note that we plot the **proportions** not the counts.

1. Convert the table into something ggplot2 can process by using `melt()` from the `reshape` package.

```
library(reshape)
```

```
##
## Attaching package: 'reshape'

## The following objects are masked from 'package:plyr':
##
##     rename, round_any
```

```
politics.prop
```

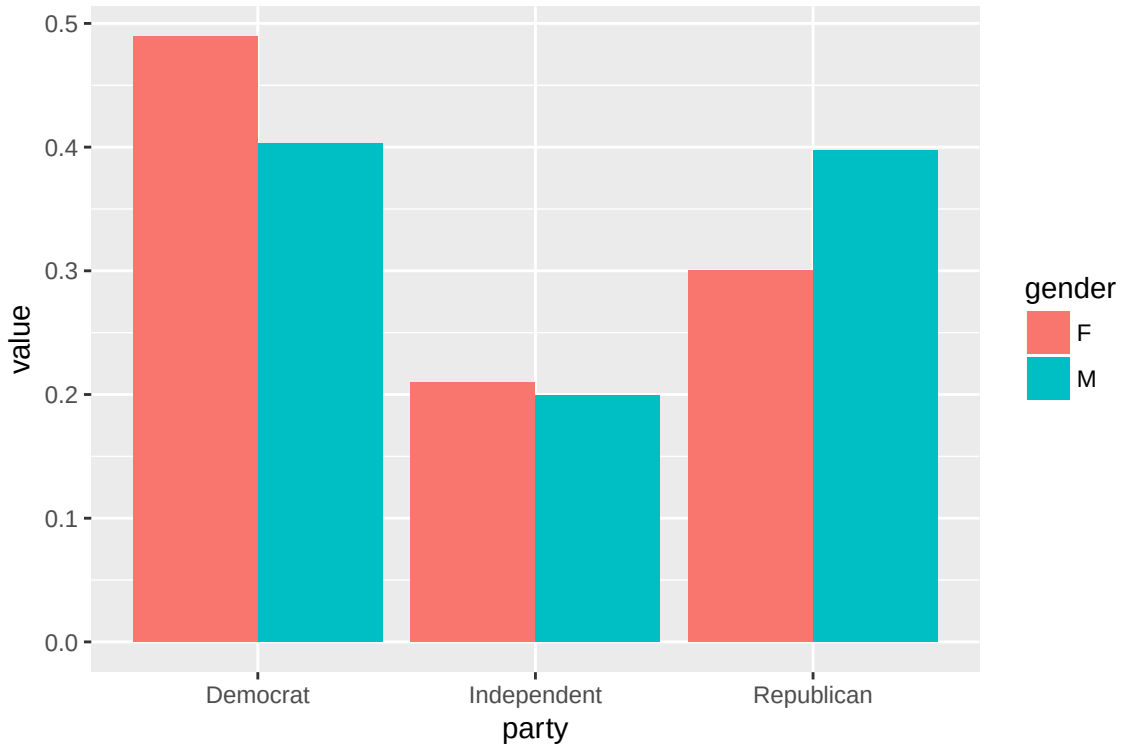
```
##       party
## gender  Democrat Independent Republican
##      F 0.4894027   0.2100193   0.300578
##      M 0.4033333   0.1991667   0.397500
```

```
politics.melt <- melt(politics.prop, id=c("gender","party"))
politics.melt
```

```
##   gender       party     value
## 1      F    Democrat 0.4894027
## 2      M    Democrat 0.4033333
## 3      F Independent 0.2100193
## 4      M Independent 0.1991667
## 5      F  Republican 0.3005780
## 6      M  Republican 0.3975000
```

2. Create a ggplot2 object, and plot with `geom_barplot()`

```
ggplot(politics.melt, aes(x=party, y=value, fill=gender)) + geom_bar(position="dodge", stat="identity")
```



This figure is a nice alternative to displaying a table. One thing we might want to add is a way of gauging the statistical significance of the differences in height. We'll do so by adding error bars.

Adding error bars to bar plots Remember, ggplot wants everything you plot to be sitting nicely in a data frame. Here's some code that will calculate the relevant values and do the plotting.

1. Get the data into a form that's easy to work with.

```
# Form into a long data frame
bi.count.melt <- melt(politics, id=c("gender", "party"))
# print
bi.count.melt
```

```
##   gender       party value
## 1      F    Democrat   762
## 2      M    Democrat   484
## 3      F Independent   327
## 4      M Independent   239
## 5      F  Republican   468
## 6      M  Republican   477
```

```
# Add a column of marginal counts
politics.count.melt <- transform(bi.count.melt, totals = rowSums(politics)[gender])
# print
politics.count.melt
```

```
##   gender       party value totals
## 1      F    Democrat   762   1557
```

```
## 2      M    Democrat   484   1200
## 3      F Independent   327   1557
## 4      M Independent   239   1200
## 5      F  Republican   468   1557
## 6      M  Republican   477   1200
```

2. Calculate confidence intervals.

To calculate confidence intervals for the proportions, we can use `prop.test` or `binom.test`. Essentially you call these functions the numerator and denominator for the proportion.

We'll do this in a for loop.

```
# define list of prop, and lower and upper endpoints
conf.ints <- list(prop = NULL, lower = NULL, upper = NULL)
for(i in 1:nrow(politics.count.melt)) {
  numerator <- politics.count.melt$value[i]
  denominator <- politics.count.melt$totals[i]
  prop.test.out <- prop.test(numerator, denominator)

  # Add estimate of proportion to list
  conf.ints[["prop"]][i] <- prop.test.out$estimate
  # Grab confidence interval
  interval <- prop.test.out$conf.int
  # Add estimate and endpoints to conf.ints list
  conf.ints[["lower"]][i] <- interval[1]
  conf.ints[["upper"]][i] <- interval[2]
}
conf.ints
```

```
## $prop
## [1] 0.4894027 0.4033333 0.2100193 0.1991667 0.3005780 0.3975000
##
## $lower
## [1] 0.4643094 0.3755178 0.1902041 0.1771484 0.2780035 0.3697701
##
## $upper
## [1] 0.5145489 0.4317750 0.2312844 0.2231402 0.3241480 0.4258939
```

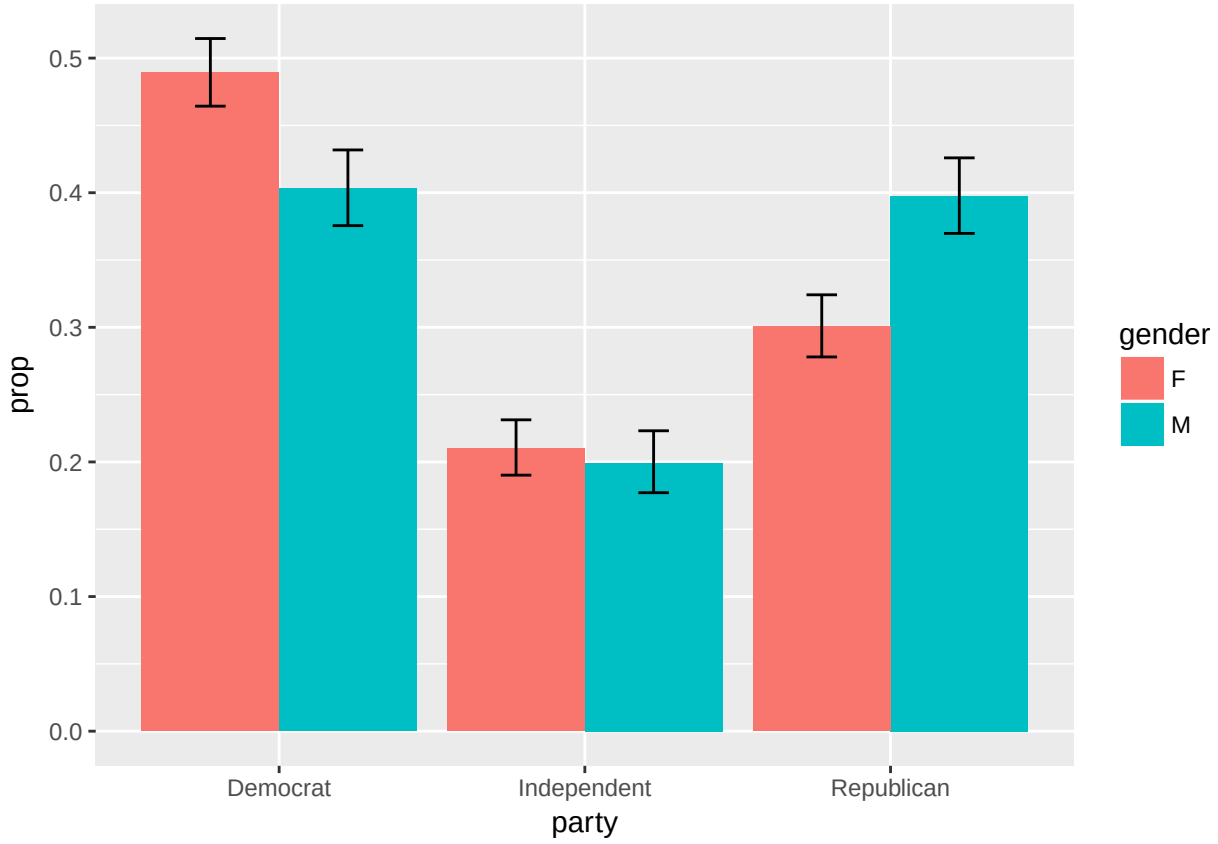
3. Combine the confidence intervals into the data frame

```
politics.toplot <- cbind(politics.count.melt, conf.ints)
politics.toplot
```

```
##   gender       party value totals      prop     lower     upper
## 1      F    Democrat   762   1557 0.4894027 0.4643094 0.5145489
## 2      M    Democrat   484   1200 0.4033333 0.3755178 0.4317750
## 3      F Independent   327   1557 0.2100193 0.1902041 0.2312844
## 4      M Independent   239   1200 0.1991667 0.1771484 0.2231402
## 5      F  Republican   468   1557 0.3005780 0.2780035 0.3241480
## 6      M  Republican   477   1200 0.3975000 0.3697701 0.4258939
```

4. Use `ggplot()`, `geom_bar()` and `geom_errorbar()` to construct the plots

```
ggplot(politics.toplot, aes(x=party, y=prop, fill=gender)) +
  geom_bar(position="dodge", stat="identity") +
  geom_errorbar(aes(ymin=lower, ymax=upper),
                width=.2,                    # Width of the error bars
                position=position_dodge(0.9))
```



ANOVA in R

1-Way ANOVA

We're going to use a data set called InsectSprays. 6 different insect sprays (1 Independent Variable with 6 levels) were tested to see if there was a difference in the number of insects found in the field after each spraying (Dependent Variable).

```
> attach(InsectSprays)
> data(InsectSprays)
> str(InsectSprays)
'data.frame':   72 obs. of  2 variables:
 $ count: num  10 7 20 14 14 12 10 23 17 20 ...
 $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

1. **Descriptive statistics**
    a. Mean, variance, number of elements in each cell
    b. Visualise the data – boxplot; look at distribution, look for outliers

We'll use the tapply() function which is a helpful shortcut in processing data, basically allowing you to specify a response variable, a factor (or factors) and a function that should be applied to each subset of the response variable defined by each level of the factor. I.e. Instead of doing:

```
> mean(count[spray=="A"])   # and the same for B, C, D etc.
```

We use tapply(response,factor,function-name) as follows

- Let's look at the means:

```
> tapply(count, spray, mean)
       A         B         C         D         E         F
14.500000 15.333333  2.083333  4.916667  3.500000 16.666667
```

- The variances:

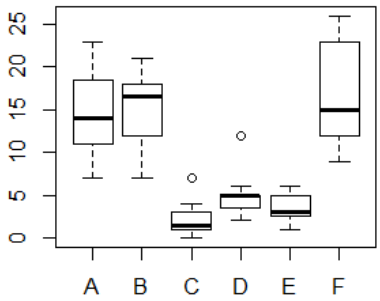
```
> tapply(count, spray, var)
       A         B         C         D         E         F
22.272727 18.242424  3.901515  6.265152  3.000000 38.606061
```

- And sample sizes

```
> tapply(count, spray, length)
 A  B  C  D  E  F
12 12 12 12 12 12
```

- And a boxplot:

```
> boxplot(count ~ spray)
```



    o How does the data look?

A couple of Asides

- Default order is alphabetical. R needs, for example, the control condition to be 1st for treatment contrasts to be easily interpreted.
- If they're not automatically in the correct order – i.e. if they were ordered variables, but came out alphabetically (e.g. "Very.short","Short","Long","Very.long" or "A", "B", "Control"), re-order the variables for ordered IV:

To change to, for example, F < B < C < D < E < A, use:

```
> Photoperiod<-ordered(spray,levels=c("F","B","C","D","E","A"))
```

Check it:

```
> tapply(count,Photoperiod,mean)
        F         B         C         D         E         A
16.666667 15.333333  2.083333  4.916667  3.500000 14.500000
```

- If you want to check that a variable is a factor (especially for variables with numbers as factor levels). We use the `is.factor` directive to find this out

```
is.factor(spray)
[1] TRUE
```

2. **Run 1-way ANOVA**
    a. **Oneway.test ( )**

- Use, for example:

```
> oneway.test(count~spray)
        One-way analysis of means (not assuming equal variances)
data:  count and spray
F = 36.0654, num df = 5.000, denom df = 30.043, p-value = 7.999e-12
```

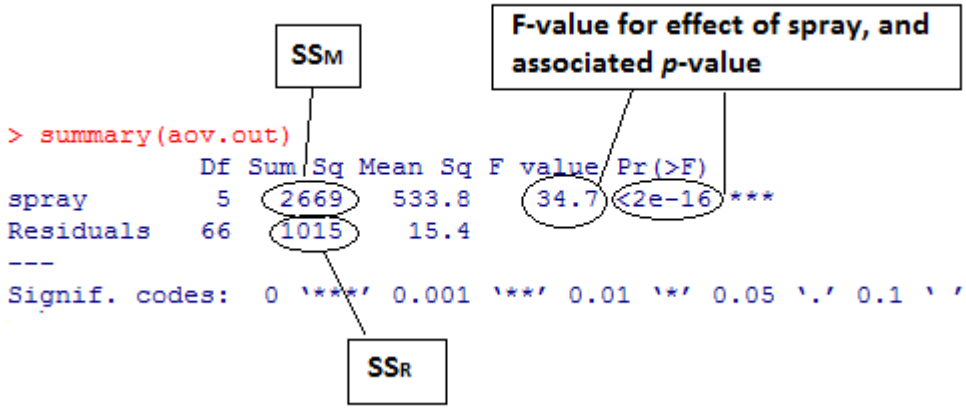
- Default is equal variances (i.e. homogeneity of variance) not assumed – i.e. Welch's correction applied (and this explains why the denom df (which is normally k*(n-1)) is not a whole number in the output)
    - To change this, set "`var.equal=`" option to TRUE

- `Oneway.test( )` corrects for non-homogeneity, but doesn't give much information – i.e. just *F*, *p*-value and *df*s for numerator and denominator – no MS etc.

    b. **Run an ANOVA using aov( )**

- Use this function and store output and use extraction functions to extract what you need.

```
> aov.out = aov(count ~ spray, data=InsectSprays)
> summary(aov.out)
```



∴ ***F*(5,66) = 34.7; *p* < .000**

3. **Post Hoc tests**

- **Tukey HSD(Honestly Significant Difference) is default in R**

```
> TukeyHSD(aov.out)
Tukey multiple comparisons of means
95% family-wise confidence level
Fit: aov(formula = count ~ spray, data = InsectSprays)

$spray
            diff         lwr          upr           p adj
B-A    0.8333333  -3.866075  5.532742 0.9951810
C-A -12.4166667 -17.116075 -7.717258 0.0000000
D-A  -9.5833333 -14.282742 -4.883925 0.0000014
E-A -11.0000000 -15.699409 -6.300591 0.0000000
F-A    2.1666667  -2.532742  6.866075 0.7542147
C-B -13.2500000 -17.949409 -8.550591 0.0000000
D-B -10.4166667 -15.116075 -5.717258 0.0000002
E-B -11.8333333 -16.532742 -7.133925 0.0000000
F-B    1.3333333  -3.366075  6.032742 0.9603075
D-C    2.8333333  -1.866075  7.532742 0.4920707
E-C    1.4166667  -3.282742  6.116075 0.9488669
F-C   14.5833333   9.883925 19.282742 0.0000000
E-D   -1.4166667  -6.116075  3.282742 0.9488669
F-D   11.7500000   7.050591 16.449409 0.0000000
F-E   13.1666667   8.467258 17.866075 0.0000000
>
```

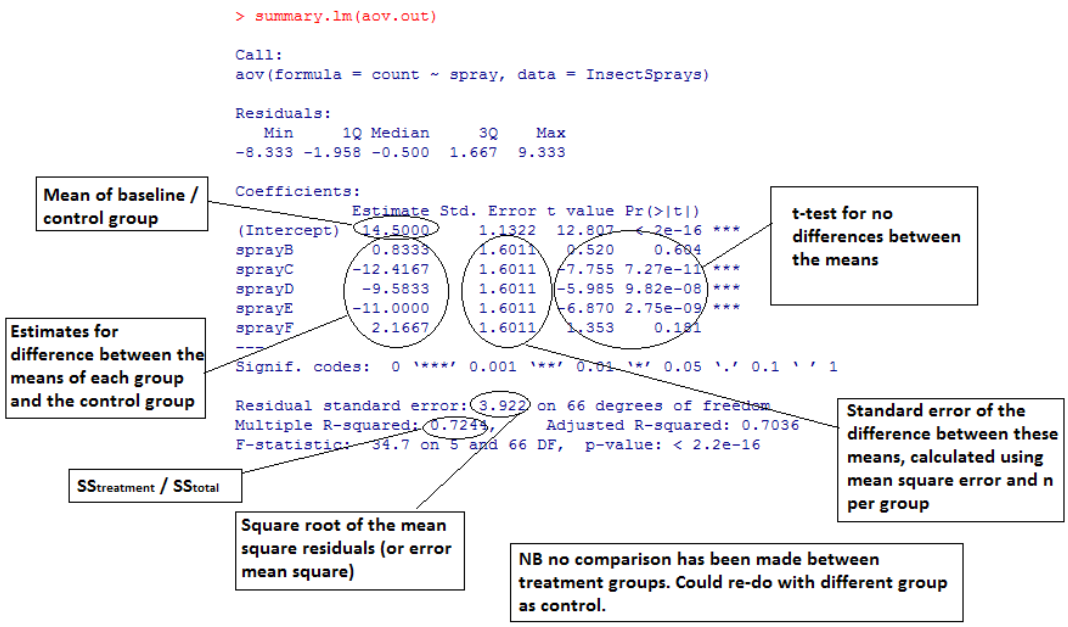
4. **Contrasts**

NB: ANOVA and linear regression are the same thing – more on that tomorrow. For the moment, the main point to note is that you can look at the results from aov() in terms of the linear regression that was carried out, i.e. you can see the parameters that were estimated.

```
> summary.lm(aov.out)
```

Implicitly this can be understood as a set of (non-orthogonal) contrasts of the first group against each of the other groups. R uses these so-called 'Treatment' contrasts as the default, but you can request alternative contrasts (see later)

Interpreting a Treatment Contrasts Output



5. Test assumptions

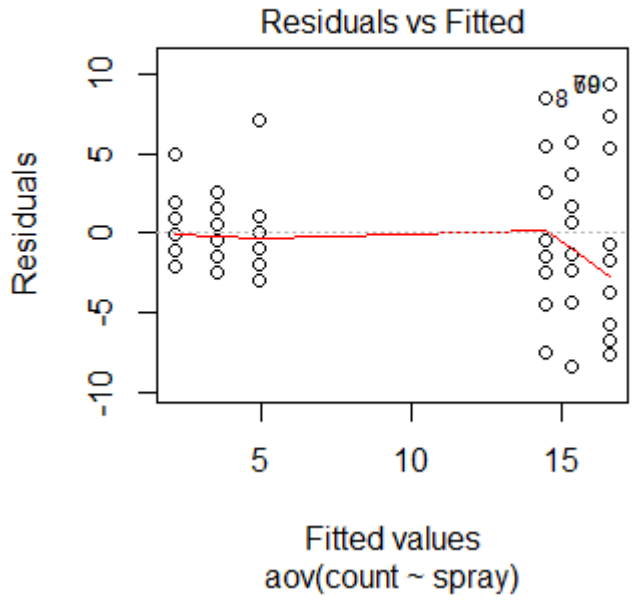
a. Homogeneity of variance

```
bartlett.test(count ~ spray, data=InsectSprays)
        Bartlett test of homogeneity of variances
data:  count by spray
Bartlett's K-squared = 25.9598, df = 5, p-value = 9.085e-05
```

∴ *Significant result, therefore variances cannot be assumed to be equal*

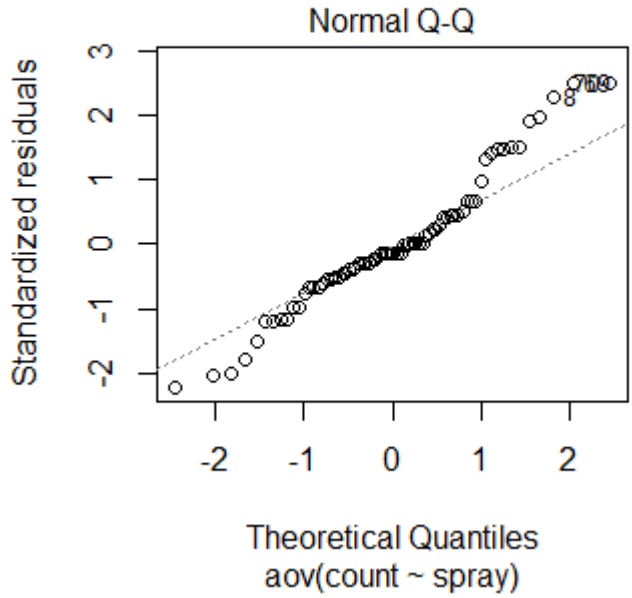
b. Model checking plots

```
> plot(aov.out)       # the aov command prepares the data for these plots
```

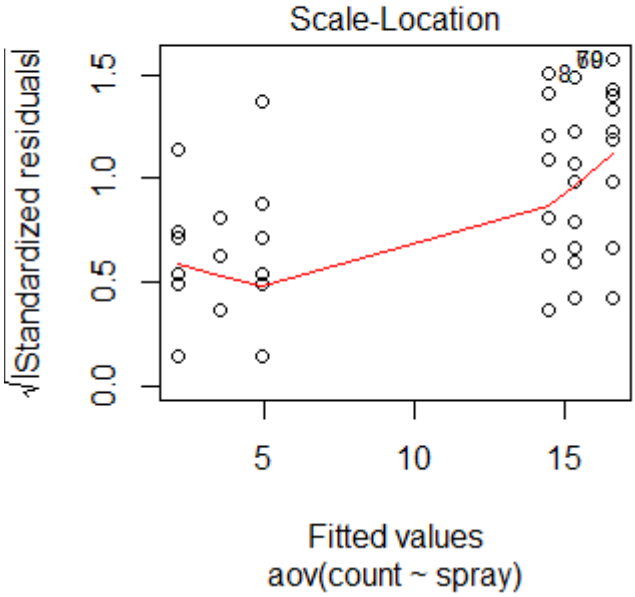


This shows if there is a pattern in the residuals, and ideally should show similar scatter for each condition. Here there is a worrying effect of larger residuals for larger fitted values. This

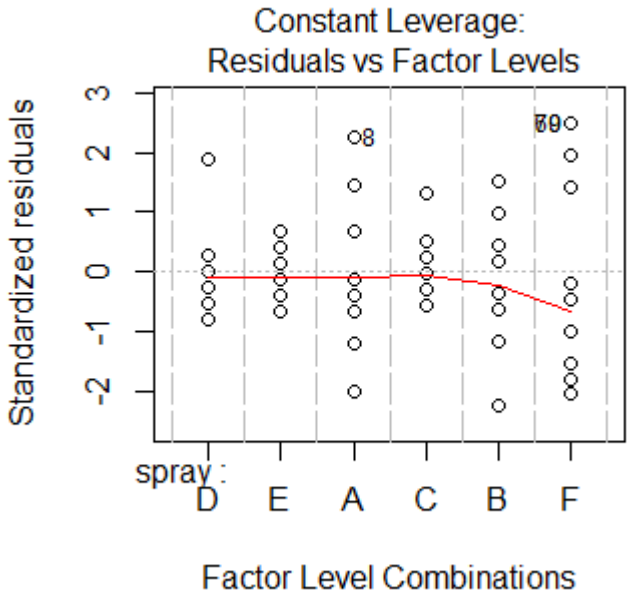
is called ‘heteroscedascity’ meaning that not only is variance in the response not equal across groups, but that the variance has some specific relationship with the size of the response. In fact you could see this in the original boxplots. It contradicts assumptions made when doing an ANOVA.



This looks for normality of the residuals; if they are not normal, the assumptions of ANOVA are potentially violated.



This is like the first plot but now to specifically test if the residuals increase with the fitted values, which they do.



This gives an idea of which levels of the factor are best fitted.

6. Non-parametric alternative to ANOVA:

```
> kruskal.test(count ~ spray, data=InsectSprays)
        Kruskal-Wallis rank sum test
data:  count by spray
Kruskal-Wallis chi-squared = 54.6913, df = 5, p-value = 1.511e-10
```

As for the Wilcoxon test (or Mann-Whitney test) with two samples, this test converts the response values to ranks, and tests whether the ranks are distributed equally across the conditions, as would be expected under the null hypothesis.

7. ANOVA as Linear Regression Analysis

This time, rather than 'attaching' the data frame, we will use the 'with' construct (see session one) to name the data frame and then do operations on variables within it.

```
> summary(PlantGrowth)
     weight       group
 Min.   :3.590   ctrl:10
 1st Qu.:4.550   trt1:10
 Median :5.155   trt2:10
 Mean   :5.073
 3rd Qu.:5.530
 Max.   :6.310
> with(PlantGrowth, tapply(weight, group, mean))
 ctrl  trt1  trt2
5.032 4.661 5.526
> with(PlantGrowth, tapply(weight, group, var))
     ctrl      trt1      trt2
0.3399956 0.6299211 0.1958711
> with(PlantGrowth, bartlett.test(weight ~ group))
        Bartlett test of homogeneity of variances
```

```
data:  weight by group
Bartlett's K-squared = 2.8786, df = 2, p-value = 0.2371
```

Now instead of running an ANOVA with aov(), we will run a linear regression with lm()

```
> lm.out = with(PlantGrowth, lm(weight ~ group))
> summary(lm.out)    # the default summary display will be the linear
                        regression

Call:
lm(formula = weight ~ group)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0710 -0.4180 -0.0060  0.2627  1.3690

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.0320     0.1971  25.527   <2e-16 ***
grouptrt1    -0.3710     0.2788  -1.331   0.1944
grouptrt2     0.4940     0.2788   1.772   0.0877 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6234 on 27 degrees of freedom
Multiple R-squared: 0.2641,   Adjusted R-squared: 0.2096
F-statistic: 4.846 on 2 and 27 DF,  p-value: 0.01591

> summary.aov(lm.out)      # we can ask for the corresponding ANOVA table

            Df Sum Sq Mean Sq F value Pr(>F)
group        2  3.766  1.8832   4.846 0.0159
Residuals   27 10.492  0.3886
```

There is a difference, but where does this difference lie?
Post Hoc test:

```
> TukeyHSD(results)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight ~ group)

$group
             diff        lwr       upr     p adj
trt1-ctrl -0.371 -1.0622161 0.3202161 0.3908711
trt2-ctrl  0.494 -0.1972161 1.1852161 0.1979960
trt2-trt1  0.865  0.1737839 1.5562161 0.0120064
```

One and Two-sample t-tests

The R function t.test() can be used to perform both one and two sample t-tests on vectors of data.

The function contains a variety of options and can be called as follows:

> t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"), mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95)

Here x is a numeric vector of data values and y is an optional numeric vector of data values. If y is excluded, the function performs a one-sample t-test on the data contained in x, if it is included it performs a two-sample t-tests using both x and y.

The option mu provides a number indicating the true value of the mean (or difference in means if you are performing a two sample test) under the null hypothesis. The option alternative is a character string specifying the alternative hypothesis, and must be one of the following: "two.sided" (which is the default), "greater" or "less" depending on whether the alternative hypothesis is that the mean is different than, greater than or less than mu, respectively. For example the following call:

> t.test(x, alternative = "less", mu = 10)

performs a one sample t-test on the data contained in x where the null hypothesis is that $\mu=10$ and the alternative is that $\mu<10$.

The option paired indicates whether or not you want a paired t-test (TRUE = yes and FALSE = no). If you leave this option out it defaults to FALSE.

The option var.equal is a logical variable indicating whether or not to assume the two variances as being equal when performing a two-sample t-test. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used. If you leave this option out it defaults to FALSE.

Finally, the option conf.level determines the confidence level of the reported confidence interval for μ in the one-sample case and μ_1-μ_2 in the two-sample case.

A. One-sample t-tests

Ex. An outbreak of Salmonella-related illness was attributed to ice cream produced at a certain factory. Scientists measured the level of Salmonella in 9 randomly sampled batches of ice cream. The levels (in MPN/g) were:

0.593 0.142 0.329 0.691 0.231 0.793 0.519 0.392 0.418

Is there evidence that the mean level of Salmonella in the ice cream is greater than 0.3 MPN/g?

Let μ be the mean level of Salmonella in all batches of ice cream. Here the hypothesis of interest can be expressed as:

H_0: $\mu = 0.3$
H_a: $\mu > 0.3$

Hence, we will need to include the options alternative="greater", mu=0.3. Below is the relevant R-code:

```
> x = c(0.593, 0.142, 0.329, 0.691, 0.231, 0.793, 0.519, 0.392, 0.418)
> t.test(x, alternative="greater", mu=0.3)

        One Sample t-test

data:  x
t = 2.2051, df = 8, p-value = 0.02927
alternative hypothesis: true mean is greater than 0.3
```

From the output we see that the p-value = 0.029. Hence, there is moderately strong evidence that the mean Salmonella level in the ice cream is above 0.3 MPN/g.

B. Two-sample t-tests

Ex. 6 subjects were given a drug (treatment group) and an additional 6 subjects a placebo (control group). Their reaction time to a stimulus was measured (in *ms*). We want to perform a two-sample t-test for comparing the means of the treatment and control groups.

Let μ_1 be the mean of the population taking medicine and μ_2 the mean of the untreated population. Here the hypothesis of interest can be expressed as:

H_0: $\mu_1-\mu_2=0$
H_a: $\mu_1-\mu_2<0$

Here we will need to include the data for the treatment group in x and the data for the control group in y. We will also need to include the options alternative="less", mu=0. Finally, we need to decide whether or not the standard deviations are the same in both groups.

Below is the relevant R-code when assuming equal standard deviation:

```
> Control = c(91, 87, 99, 77, 88, 91)
> Treat = c(101, 110, 103, 93, 99, 104)
>  t.test(Control,Treat,alternative="less", var.equal=TRUE)

        Two Sample t-test

data:  Control and Treat
t = -3.4456, df = 10, p-value = 0.003136
alternative hypothesis: true difference in means is less than 0
```

Below is the relevant R-code when not assuming equal standard deviation:

```
>  t.test(Control,Treat,alternative="less")

       Welch Two Sample t-test

data:  Control and Treat
t = -3.4456, df = 9.48, p-value = 0.003391
alternative hypothesis: true difference in means is less than 0
```

Here the pooled t-test and the Welsh t-test give roughly the same results (p-value = 0.00313 and 0.00339, respectively).

C. Paired t-tests

There are many experimental settings where each subject in the study is in both the treatment and control group. For example, in a matched pairs design, subjects are matched in pairs and different treatments are given to each subject in the pair. The outcomes are thereafter compared pair-wise. Alternatively, one can measure each subject twice, before and after a treatment. In either of these situations we can't use two-sample t-tests since the independence assumption is not valid. Instead we need to use a paired t-test. This can be done using the option paired =TRUE.

Ex. A study was performed to test whether cars get better mileage on premium gas than on regular gas. Each of 10 cars was first filled with either regular or premium gas, decided by a coin toss, and the mileage for that tank was recorded. The mileage was recorded again for the same cars using the other kind of gasoline. We use a paired t-test to determine whether cars get significantly better mileage with premium gas.

Below is the relevant R-code:

```
> reg = c(16, 20, 21, 22, 23, 22, 27, 25, 27, 28)
> prem = c(19, 22, 24, 24, 25, 25, 26, 26, 28, 32)
> t.test(prem,reg,alternative="greater", paired=TRUE)

        Paired t-test

data:  prem and reg
t = 4.4721, df = 9, p-value = 0.000775
alternative hypothesis: true difference in means is greater than 0
```

The results show that the t-statistic is equal to 4.47 and the p-value is 0.00075. Since the *p*-value is very low, we reject the null hypothesis. There is strong evidence of a mean increase in gas mileage between regular and premium gasoline.

Simple Linear Regression

A simple linear regression model that describes the relationship between two variables x and y can be expressed by the following equation. The numbers α and β are called parameters, and ϵ is the error term.

$$y = \alpha + \beta x + \epsilon$$

For example, in the data set faithful, it contains sample data of two random variables named waiting and eruptions.

The waiting variable denotes the waiting time until the next eruptions, and eruptions denotes the duration. Its linear regression model can be expressed as:

$$Eruption = \alpha + \beta\, Waiting + \epsilon$$

Estimated Simple Regression Equation

If we choose the parameters α and β in the simple linear regression model so as to minimize the sum of squares of the error term ϵ, we will have the so called estimated simple regression equation.

It allows us to compute fitted values of y based on values of x.

$$\hat{y} = a + b\, x$$

Problem

Apply the simple linear regression model for the data set faithful, and estimate the next eruption duration if the waiting time since the last eruption has been 80 minutes.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
```

Then we extract the parameters of the estimated regression equation with the coefficients function.

```
> coeffs = coefficients(eruption.lm); coeffs
(Intercept)     waiting
  -1.874016    0.075628
```

We now fit the eruption duration using the estimated regression equation.

```
> waiting = 80                                  # the waiting time
> duration = coeffs[1] + coeffs[2]*waiting
> duration
(Intercept)
     4.1762
```

Answer

Based on the simple linear regression model, if the waiting time since the last eruption has been 80 minutes, we expect the next one to last 4.1762 minutes.

Alternative Solution

We wrap the waiting parameter value inside a new data frame named newdata.

```
> newdata = data.frame(waiting=80) # wrap the parameter
```

Then we apply the predict function to eruption.lm along with newdata.

```
> predict(eruption.lm, newdata)       # apply predict
     1
4.1762
```

Coefficient of Determination

The coefficient of determination of a linear regression model is the quotient of the variances of the fitted values and observed values of the dependent variable.

If we denote y_i as the observed values of the dependent variable, $\overline{y}$ as its mean, and $\widehat{y}_i$ as the fitted value, then the coefficient of determination is:

$$r^2 = \frac{\sum (\widehat{y}_i - \overline{y})^2}{\sum (y_i - \overline{y})^2}$$

Problem

Find the coefficient of determination for the simple linear regression model of the data set faithful.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
```

Then we extract the coefficient of determination from the r.squared attribute of its summary.

```
> summary(eruption.lm)$r.squared
[1] 0.81146
```

Answer

The coefficient of determination of the simple linear regression model for the data set faithful is 0.81146.

Note

Further detail of the r.squared attribute can be found in the R documentation.

```
> help(summary.lm)
```

Significance Test for Linear Regression

Assume that the error term ϵ in the linear regression model is independent of x, and is normally distributed, with zero mean and constant variance.

We can decide whether there is any significant relationship between x and y by testing the null hypothesis that $\beta = 0$.

Problem

Decide whether there is a significant relationship between the variables in the linear regression model of the data set faithful at .05 significance level.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
```

Then we print out the F-statistics of the significance test with the summary function.

```
> summary(eruption.lm)

Call:
lm(formula = eruptions ~ waiting, data = faithful)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2992 -0.3769  0.0351  0.3491  1.1933

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.87402    0.16014   -11.7   <2e-16 ***
waiting      0.07563    0.00222    34.1   <2e-16 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 0.497 on 270 degrees of freedom
Multiple R-squared: 0.811,      Adjusted R-squared: 0.811
F-statistic: 1.16e+03 on 1 and 270 DF,  p-value: <2e-16
```

Answer

As the p-value is much less than 0.05, we reject the null hypothesis that $\beta = 0$. Hence there is a significant relationship between the variables in the linear regression model of the data set faithful.

Note

Further detail of the summary function for linear regression model can be found in the R documentation.

```
> help(summary.lm)
```

Confidence Interval for Linear Regression

Assume that the error term ϵ in the linear regression model is independent of x, and is normally distributed, with zero mean and constant variance.

For a given value of x, the interval estimate for the mean of the dependent variable, $\bar{y}$, is called the confidence interval.

Problem

In the data set faithful, develop a 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> attach(faithful)     # attach the data frame

> eruption.lm = lm(eruptions ~ waiting)
```

Then we create a new data frame that set the waiting time value.

```
> newdata = data.frame(waiting=80)
```

We now apply the predict function and set the predictor variable in the newdata argument.

We also set the interval type as "confidence", and use the default 0.95 confidence level.

```
> predict(eruption.lm, newdata, interval="confidence")
    fit    lwr    upr
1 4.1762 4.1048 4.2476

> detach(faithful)     # clean up
```

Answer

The 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes is between 4.1048 and 4.2476 minutes.

Note

Further detail of the predict function for linear regression model can be found in the R documentation.

```
> help(predict.lm)
```

Alternative Solution

We create a function

```
> f <- function(x) {
+ newdata = data.frame(waiting=x)
+ predict.lm = predict(eruption.lm, newdata, interval="confidence")
+ return(predict.lm)
+ }
```

Now only give the value to predict

```
> f(10)
       fit       lwr       upr
1 -1.117737 -1.390249 -0.845224
```

Create one seq. and evaluated it

```
> x = 10:100
> f(x)
> predict = f(x)

> plot(eruptions~waiting, data=faithful)
> lines(x,predict[,1],col="red")
> lines(x,predict[,2],col="blue")
> lines(x,predict[,3],col="blue")
```

Prediction Interval for Linear Regression

Assume that the error term ϵ in the simple linear regression model is independent of x, and is normally distributed, with zero mean and constant variance.

For a given value of x, the interval estimate of the dependent variable y is called the prediction interval.

Problem

In the data set faithful, develop a 95% prediction interval of the eruption duration for the waiting time of 80 minutes.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> attach(faithful)     # attach the data frame

> eruption.lm = lm(eruptions ~ waiting)
```

Then we create a new data frame that set the waiting time value.

```
> newdata = data.frame(waiting=80)
```

We now apply the predict function and set the predictor variable in the newdata argument.

We also set the interval type as "predict", and use the default 0.95 confidence level.

```
> predict(eruption.lm, newdata, interval="predict")
    fit    lwr    upr
1 4.1762 3.1961 5.1564

> detach(faithful)     # clean up
```

Answer

The 95% prediction interval of the eruption duration for the waiting time of 80 minutes is between 3.1961 and 5.1564 minutes.

Note

Further detail of the predict function for linear regression model can be found in the R documentation.

```
> help(predict.lm)
```

Alternative Solution

We create a function

```
> f1 <- function(x) {
+ newdata = data.frame(waiting=x)
+ predict.lm = predict(eruption.lm, newdata, interval="predict")
+ return(predict.lm)
+ }
```

Now only give the value to predict

```
> f1(10)
       fit      lwr       upr
1 -1.117737 -2.13254 -0.1029329
```

Create one seq. and evaluated it

```
> x = 10:100
> f1(x)
> predict1 = f1(x)

> plot(eruptions~waiting, data=faithful)
> lines(x,predict1[,1],col="red")
> lines(x,predict1[,2],col="blue")
> lines(x,predict1[,3],col="blue")
```

Residual Plot

The residual data of the simple linear regression model is the difference between the observed data of the dependent variable y and the fitted values $\hat{y}$.

$$Residual = y - \hat{y}$$

Problem

Plot the residual of the simple linear regression model of the data set faithful against the independent variable waiting.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

Then we compute the residual with the resid function.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)

> eruption.res = resid(eruption.lm)
```

We now plot the residual against the observed values of the variable waiting.

```
> plot(faithful$waiting, eruption.res,
+     ylab="Residuals", xlab="Waiting Time",
+     main="Old Faithful Eruptions")

> abline(0, 0)                                    # the horizon
```

Note

Further detail of the resid function can be found in the R documentation.

```
> help(resid)
```

Standardized Residual

The standardized residual is the residual divided by its standard deviation.

$$Standardized\ Residual\ i = \frac{Residual\ i}{Standard\ Deviation\ of\ Residual\ i}$$

Problem

Plot the standardized residual of the simple linear regression model of the data set faithful against the independent variable waiting.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

Then we compute the standardized residual with the rstandard function.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
```

```
> eruption.stdres = rstandard(eruption.lm)
```

We now plot the standardized residual against the observed values of the variable waiting.

```
> plot(faithful$waiting, eruption.stres,
+     ylab="Standardized Residuals",
+     xlab="Waiting Time",
+     main="Old Faithful Eruptions")
> abline(0, 0)                          # the horizon
```

Note

Further detail of the rstandard function can be found in the R documentation.

```
> help(rstandard)
```

Normal Probability Plot of Residuals

The normal probability plot is a graphical tool for comparing a data set with the normal distribution.

We can use it with the standardized residual of the linear regression model and see if the error term ϵ is actually normally distributed.

Problem

Create the normal probability plot for the standardized residual of the data set faithful.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

Then we compute the standardized residual with the rstandard function.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)

> eruption.stdres = rstandard(eruption.lm)
```

We now create the normal probability plot with the qqnorm function, and add the qqline for further comparison.

```
> qqnorm(eruption.stdres,
+     ylab="Standardized Residuals",
+     xlab="Normal Scores",
+     main="Old Faithful Eruptions")

> qqline(eruption.stdres)
```

Note

Further detail of the qqnorm and qqline functions can be found in the R documentation.

```
> help(qqnorm)
```

Exercise

This function computes model II simple linear regression using the following methods: ordinary least squares (OLS), major axis (MA), standard major axis (SMA), and ranged major axis (RMA). The model only accepts one response and one explanatory variable.

```
> install.packages("lmodel2")


> library(lmodel2)


> help(lmodel2)
```

```
## Example 1 (surgical unit data)
> data(mod2ex1)
> Ex1.res <- lmodel2(Predicted_by_model ~ Survival, data=mod2ex1, nperm=99)
> Ex1.res
> plot(Ex1.res)
```

```
## Example 2 (eagle rays and Macomona)
> data(mod2ex2)
> Ex2.res <- lmodel2(Prey ~ Predators, data=mod2ex2, "relative", "relative", 99)
> Ex2.res
> op <- par(mfrow = c(1,2))
> plot(Ex2.res, "SMA")
> plot(Ex2.res, "RMA")
> par(op)
```

```
## Example 3 (cabezon spawning)
> op <- par(mfrow = c(1,2))
> data(mod2ex3)
> Ex3.res <- lmodel2(No_eggs ~ Mass, data=mod2ex3, "relative", "relative", 99 )
> Ex3.res
> plot(Ex3.res, "SMA")
> plot(Ex3.res, "RMA")
> par(op)
```

```
## Example 4 (highly correlated random variables)
> op <- par(mfrow=c(1,2))
> data(mod2ex4)
> Ex4.res <- lmodel2(y ~ x, data=mod2ex4, "interval", "interval", 99)
> Ex4.res
> plot(Ex4.res, "OLS")
> plot(Ex4.res, "MA")
> par(op)



# Example 5 (uncorrelated random variables)
> data(mod2ex5)
> Ex5.res <- lmodel2(random_y ~ random_x, data=mod2ex5, "interval", "interval", 99)
> Ex5.res
> op <- par(mfrow = c(2,2))
> plot(Ex5.res, "OLS")
> plot(Ex5.res, "MA")
> plot(Ex5.res, "SMA")
> plot(Ex5.res, "RMA")
> par(op)



## Example 6 where cor(y,x) = 0 by construct (square grid of points)
> y0 = rep(c(1,2,3,4,5),5)
> x0 = c(rep(1,5),rep(2,5),rep(3,5),rep(4,5),rep(5,5))
> plot(x0, y0)
> Ex6 = as.data.frame(cbind(x0,y0))
> zero.res = lmodel2(y0 ~ x0, data=Ex6, "relative", "relative")
> print(zero.res)
> op <- par(mfrow = c(1,2))
> plot(zero.res, "OLS")
> plot(zero.res, "MA")
> par(op)
```

```
###TESTYOURSELF2

#1.a
firstWeek <- c(8, 10, 9, 11, 8, 7, 9, 10, 9, 9)
secondWeek <- c(5, 7, 5, 6, 7, 5, 4, 6, 5, 6)

t.test(firstWeek, secondWeek, alternative = "greater", paired = TRUE)

#1.d
t.test(firstWeek, secondWeek, alternative = "greater", paired = TRUE, conf.level = 0.99)


#1.e
t.test(firstWeek)
t.test(secondWeek)

#2
notCoffee <- c(8, 10, 9, 11, 8, 7, 9, 10, 9, 9)
coffee <- c(5, 7, 5, 6, 7, 5, 4, 6, 5, 6)

var.test(notCoffee, coffee) #sigma 1 sigma 2 ye e?it mi diye test ettim(ortalama de?il variance a bakt?k)

t.test(notCoffee, coffee, alternative = "greater", var.equal = TRUE) #H0 = reject

t.test(notCoffee, coffee, alternative = "greater", var.equal = TRUE, conf.level = 0.99)

t.test(notCoffee, coffee, alternative = "two.sided", var.equal = TRUE)#kafein t?ketirsek 2.39 la 4.4 aras?nda uyku kayb?na sebep olur

#3
stCoffee <- c(6, 7, 5, 5, 7, 5, 3, 6, 6, 6)

data <- c(notCoffee, coffee, stCoffee) #verileri birle?tirmek i?in
group <- c(rep(1,10), rep(2,10), rep(3,10)) #veri k?melerinin 10 eleman?n? se?mek i?in

result <- aov(data ~ as.factor(group))
summary(result) #Pr(>F) p value'ya e?ittir. at leaast one of the groups is different than others
TukeyHSD(result)

#4
x <- runif(100, 0, 10)
y <- 2 + 3*x + rnorm(100)

cor.test(x,y)

model <- lm(y~x)
summary(model)

#katsay?lar?n sonunda y?ld?z varsa, o katsay?lar modelde bulunmal? demek
```

Statistical Computing
Lab 01

For the first two problems we will use `Cars93` dataset from the MASS library.

We first load the library: `library(MASS)`

1. Manipulating data frames

There are certain situations where we want to transform right-skewed data before analyzing it. Taking the log of right-skewed data often helps to make it more normally distributed.

a. Draw histograms for `MPG.highway` and `MPG.city` and evaluate the shape of the data.

```
p1<- ggplot(Cars93, aes(x = MPG.highway)) + xlab("MPG Highway")
p1+ geom_histogram()
p2 <- ggplot(Cars93, aes(x = MPG.city)) + xlab("MPG City")
p2 + geom_histogram()
```

The shape of highway and city variables are asymmetric and they include potential outliers resulting in heavy tails.

b. Draw density plots for `MPG.highway` and `MPG.city` and check if they support your previous decision.

```
p1 <- ggplot(Cars93, aes(x = MPG.highway)) + xlab("MPG Highway")
p1 + geom_density()
p2 <- ggplot(Cars93, aes(x = MPG.city)) + xlab("MPG City")
p2 + geom_density()
```

c. Check the same variables for existence of any potential outliers.

The tails include some values larger than 40. These observations might be chategorized as potential outliers

```
Potential.outliers.highway<- (subset(Cars93, colnames(data), subset = MPG.highway > 40))
Potential.outliers.city<- (subset(Cars93, colnames(data), subset = MPG.highway > 40))
```

d. Create two data frames for USA and non-USA origin cars

```
USAcars <- (subset(Cars93, colnames(data), subset = Origin == "USA"))
nonUSAcars <- (subset(Cars93, colnames(), subset = Origin == "non-USA"))
```

e. Evaluate “c” again for these datasets.

```
Potential.outliers.highway<- (subset(USA, colnames(data), subset = MPG.highway > 40))
Potential.outliers.city<- (subset(USA, colnames(data), subset = MPG.city > 40))
Potential.outliers.highway<- (subset(nonUSA, colnames(data), subset = MPG.highway > 40))
Potential.outliers.city<- (subset(nonUSA, colnames(data), subset = MPG.city > 40))
```

f. Use the `transform()` and `log()` functions to create a new data frame called `Cars93.log` that has `MPG.highway` and `MPG.city` replaced with `log(MPG.highway)` and `log(MPG.city)`. Run the density plot commands again, this time using your new `Cars93.log` dataset instead of `Cars93`.

```
Cars93.log <- transform(Cars93, MPG.highway = log(MPG.highway), MPG.city = log(MPG.city))
```

g. Do the distributions appear less skewed than before?

```
p1 <- ggplot(Cars93.log, aes(x = MPG.highway)) + xlab("MPG Highway")
p1 + geom_density()
p2 <- ggplot(Cars93.log, aes(x = MPG.city)) + xlab("MPG City")
p2 + geom_density()
```

The distributions appear less skewed than before.

2. Table function

a. Use the `table()` function to tabulate the data `DriveTrain` and `Origin`.

```
table(Cars93$DriveTrain, Cars93$Origin)
```

b. Does it look like foreign car manufacturers had different `Drivetrain` preferences compared to US manufacturers?

```
        USA non-USA
4WD     5      5
Front   34     33
Rear    9      7
```

The table shows that there is no significant difference between USA and nonUSA made cars in terms of DriveTrain.

c. Summarize `Price, MPG.city, MPG.highway, EngineSize, Rear.seat.room` for `Manufacturer, Type.`

```
summary(Cars93)
```

d. Use `with()` and see the differences between choosing `tapply()` and `aggregate()` functions used as an argument.

```
with(Cars93, aggregate(Price, by = list(Origin, Manufacturer), FUN = summary))
with(Cars93, tapply(Price, INDEX = list(Origin, Manufacturer), FUN = summary))
```

e. Explore the differences for `Origin`. Show your findings with summary statistics and support them with plots.

```
summary(USA)
base.plot <- ggplot(USA, aes(x = MPG.highway)) + xlab("USAcars")
base.plot + geom_density()
summary(nonUSA)
base.plot <- ggplot(nonUSA, aes(x = MPG.highway)) + xlab("NONUSAcars")
base.plot + geom_density()
```

f. Check `MPG.highway` and `MPG.city` for existence of any potential outliers conditioning on `Origin` using summary statistics. What would be your choice for central tendency and spread measures?

As mean and median values are pretty close for MPG.highway and MPG.city, we recommend using mean as the measure of central tendency. Thus, providing standard deviation would be the best strategy as a measure of dispersion along with mean.

3. Functions, lists, and if-else practice

a. Write a function called `isPassingGrade` whose input `x` is a number, and which returns `FALSE` if `x` is lower than 50 and `TRUE` otherwise. Try the function on an input value of your choice and confirm that it works correctly.

```
isPassingGrade <- function (x) {
  x>=50
}
Ali <- 60
isPassingGrade(Ali)
```

b. Write a function called `sendMessage` whose input `x` is a number, and which prints text `Congratulations` if `isPassingGrade(x)` is `TRUE` and prints `Oh no!` if `isPassingGrade(x)` is `FALSE`.

```
sendMessage <- function(x){
  if (isPassingGrade(x) == TRUE){
    "Congratulations"
  } else {
    "Oh no!"
  }
```

```
}
sendMessage(Ali)
```

c. Write a function called `gradeSummary` whose input `x` is a number. Your function will return a list with two elements, named `letter.grade` and `passed`. The letter grade will be `"A"` if `x` is at `90`. The letter grade will be `"B"` if `x` is between `80` and `90`. The letter grade will be `"F"` if `x` is lower than `"80"`. If the student's letter grade is an A or B, `passed` should be TRUE; `passed` should be FALSE otherwise.

To check if your function works, try the following cases:

`x = 91` should return

```
list(letter.grade = "A", passed = TRUE)
```

`x = 62` should return

```
list(letter.grade = "F", passed = FALSE)
```

```
gradeSummary <- function(x){
 if (x <= 100 & x >= 90){
  list(letter.grade = "A", passed = TRUE)
 } else if (x < 90 & x >= 80){
  list(letter.grade = "B", passed = TRUE)
 } else if (x>=0 & x < 80){
  list(letter.grade = "F", passed = FALSE)
 }
}
gradeSummary(Ali)
```

d. Create a function for your own summary statistics list. Use your previous MissingValueReplacer() function to clean NAs before computing statistics.

```
MissingValueReplacer <- function(x) {
 ifelse(is.na(x) == FALSE, x, mean(x, na.rm=TRUE))
}
mysummary <- function(x) {
 x<missingvaluereplacer(x)
 c(mean(x), median(x), sd(x).....) }
```

e. Create a function that creates visual summaries for your data using ggplot2.

```
blue.plot <- function(x, y, ...) {
  par(mfrow=c(2,2))
  plot(x, col="blue", ...)
  plot(y, col="blue", ...)
  plot(x, y, col="blue", ...)
}
blue.plot(Cars93$MPG.highway,Cars93$MPG.city)
```

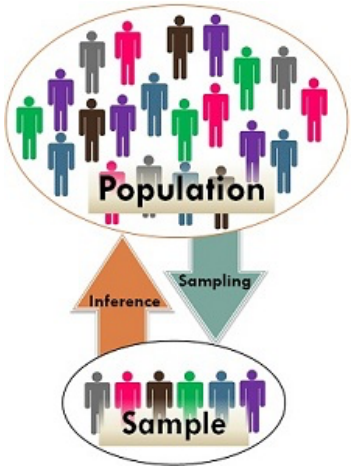
SAMPLING

BACKGROUND

- We have learned ways to **display**, **describe**, and **summarize** data, but have been limited to examining the particular batch of data we have.
- We'd like (and often need) to stretch beyond the data at hand to the world at large.
- Let's investigate three major ideas that will allow us to make this stretch…

IDEA 1: EXAMINE A PART OF THE WHOLE

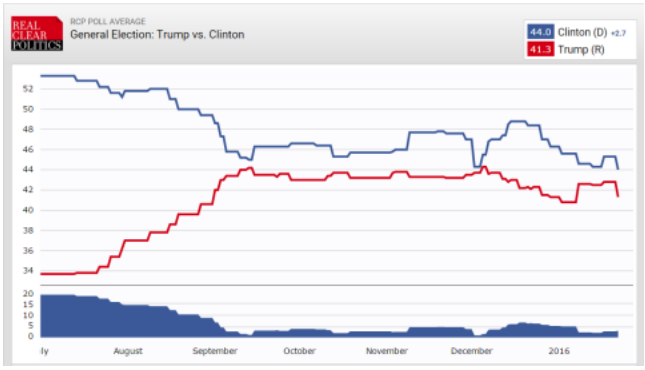
- **Sampling** is a natural thing to do.
- Think about sampling something you are cooking—you taste (examine) a small part of what you're cooking to get an idea about the dish as a whole.



Slide 12-3

IDEA 1: EXAMINE PART OF THE WHOLE

- Opinion polls are examples of sample surveys, designed to ask questions of a small group of people in the hope of learning something about the entire population.
  - Professional pollsters work quite hard to ensure that the sample they take is representative of the population.
  - If not, the sample can give misleading information about the population.



Slide 12-4

IDEA 2: RANDOMIZE

- Randomization can protect you against factors that you know are in the data.
  - It can also help protect against factors you are not even aware of.
- Randomizing protects us from the influences of *all* the features of our population, even ones that we may not have thought about.
  - Randomizing makes sure that *on the average* the sample looks like the rest of the population.

IDEA 3: IT'S THE SAMPLE SIZE

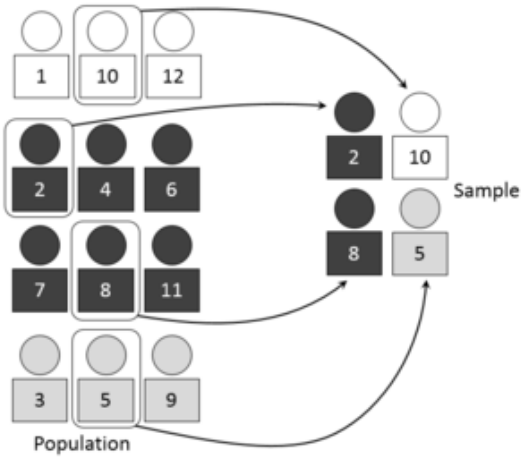
- How large a random sample do we need for the sample to be reasonably representative of the population?
- It's the size of the sample, not the size of the population, that makes the difference in sampling.
  - Exception: If the population is small enough and the sample is more than 10% of the whole population, the population size *can* matter.
- The *fraction* of the population that you've sampled doesn't matter. It's the *sample size* itself that's important.

SIMPLE RANDOM SAMPLES

- We insist that every possible *sample* of the size we plan to draw has an equal chance to be selected.
  - Such samples also guarantee that each individual has an equal chance of being selected.
  - With this method each *combination* of people has an equal chance of being selected as well.
  - A sample drawn in this way is called a Simple Random Sample (SRS).
- An SRS is the standard against which we measure other sampling methods, and the sampling method on which the theory of working with sampled data is based.

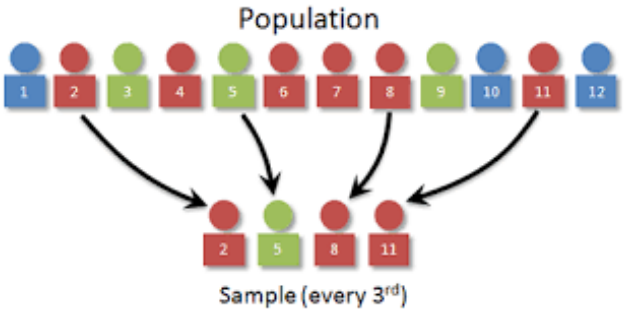
STRATIFIED SAMPLING

- Designs used to sample from large populations are often more complicated than simple random samples.
- Sometimes the population is first sliced into homogeneous groups, called strata, before the sample is selected.
- Then simple random sampling is used within each stratum before the results are combined.
- This common sampling design is called stratified random sampling.

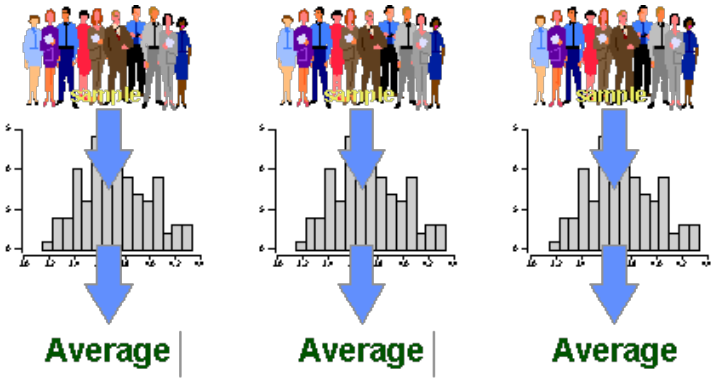


SYSTEMATIC SAMPLES

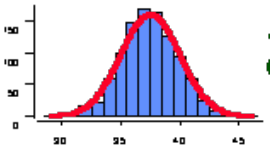
- Sometimes we draw a sample by selecting individuals systematically.
  - For example, you might survey every 3rd person on an alphabetical list of students.
- To make it random, you must still start the systematic selection from a randomly selected individual.



SAMPLING DISTRIBUTION MODELS



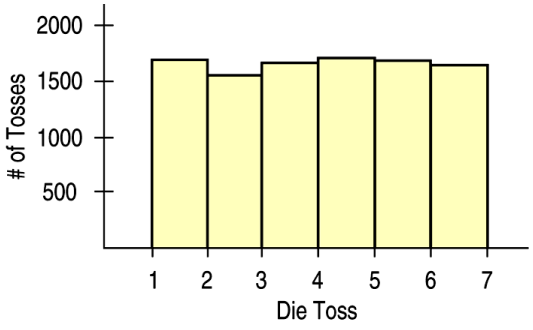
The Sampling Distribution...

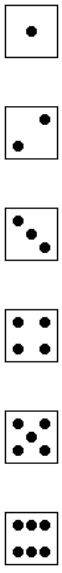


...is the distribution of a statistic across an infinite number of samples

MEANS – THE "AVERAGE" OF ONE DIE

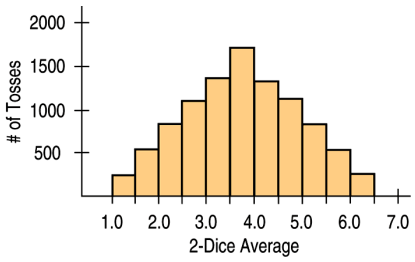
- Let's start with a simulation of 10,000 tosses of a die. A histogram of the results is:



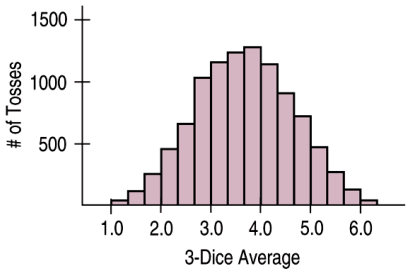


MEANS – THE "AVERAGE" OF ONE DIE

- Looking at the average of two dice after a simulation of 10,000 tosses:

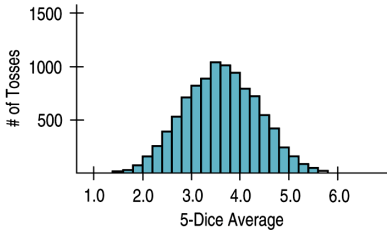


- The average of three dice after a simulation of 10,000 tosses looks like:

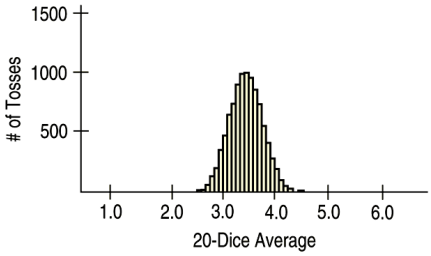


MEANS – THE "AVERAGE" OF ONE DIE

- The average of 5 dice after a simulation of 10,000 tosses looks like:



- The average of 20 dice after a simulation of 10,000 tosses looks like:

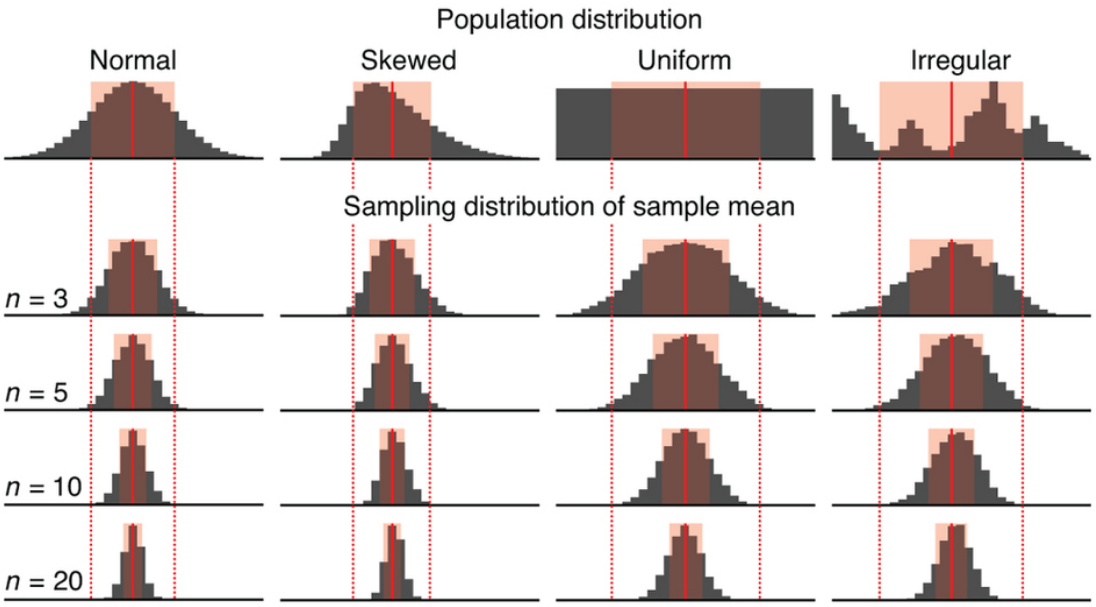


MEANS – WHAT THE SIMULATIONS SHOW

- As the sample size (number of dice) gets larger, each sample average is more likely to be closer to the population mean.
  - So, we see the shape continuing to tighten around 3.5
- And, it probably does not shock you that the sampling distribution of a mean becomes Normal.

THE FUNDAMENTAL THEOREM OF STATISTICS

- The sampling distribution of *any* mean becomes Normal as the sample size grows.
  - All we need is for the observations to be independent and collected with randomization.
  - We don't even care about the shape of the population distribution!
- The Fundamental Theorem of Statistics is called the Central Limit Theorem (CLT).



BUT WHICH NORMAL?

- The CLT says that the sampling distribution of any mean or proportion is approximately Normal.
- But which Normal model?
  - For proportions, the sampling distribution is centered at the population proportion.
  - For means, it's centered at the population mean.
- But what about the standard deviations?

BUT WHICH NORMAL? (CONT.)

- The Normal model for the sampling distribution of the mean has a standard deviation equal to

$$SD(\bar{y}) = \frac{\sigma}{\sqrt{n}}$$

where σ is the population standard deviation.

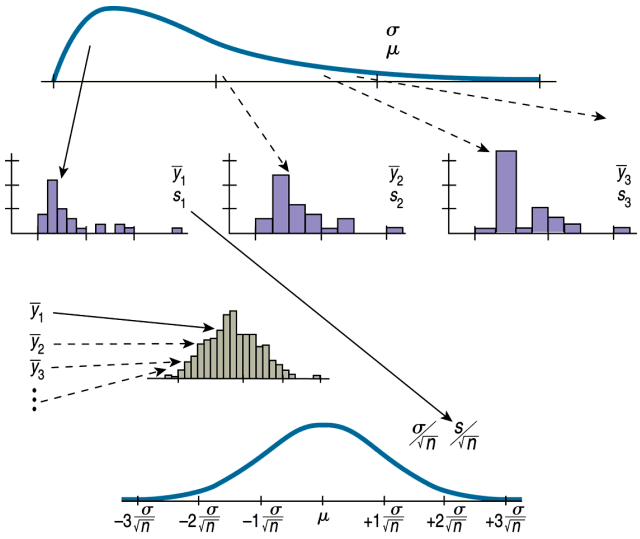
STANDARD ERROR (CONT.)

- When we don't know σ, we're stuck, right?
- We will use sample statistics to estimate these population parameters.
- Whenever we estimate the standard deviation of a sampling distribution, we call it a standard error.

STANDARD ERROR

- Both of the sampling distributions we've looked at are Normal.
  - For means $SE(\bar{y}) = \frac{s}{\sqrt{n}}$

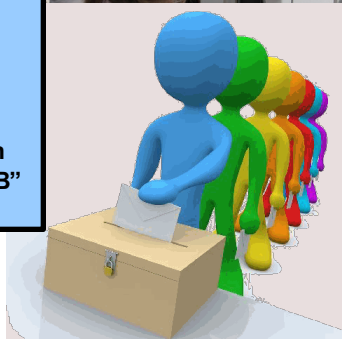
THE PROCESS GOING INTO THE SAMPLING DISTRIBUTION MODEL



CONFIDENCE INTERVAL ESTIMATE

- I am 95% confident that the true average income of U.S. appliance factory workers is between $51,000.00 and $56,000.00 per year.
- I am 90% confident that the true proportion of potential voters supporting candidate "B" is between 35% and 38%.

EXAMPLES



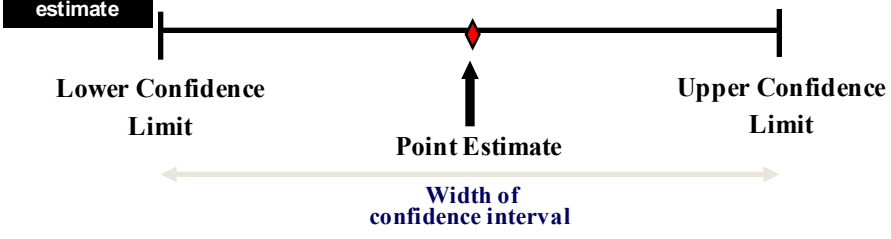
POINT ESTIMATES

The starting point for developing the confidence interval for μ

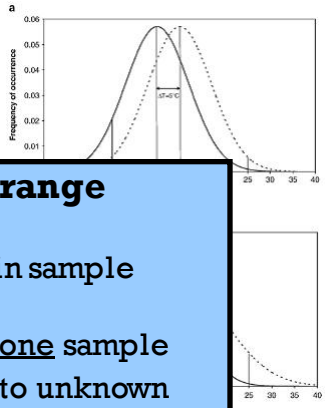
The value of a single sample statistic: mean or proportion

A range of numbers constructed around the point estimate

- A **point estimate** is a single number. For the population mean and population standard deviation, a point estimate is the sample mean and sample standard deviation.
- A **confidence interval** provides additional information about variability



CONFIDENCE INTERVAL ESTIMATES



- **A confidence interval gives a range estimate of values:**
  - Takes into consideration variation in sample statistics from sample to sample
  - Based on all the observations from one sample
  - Gives information about closeness to unknown population parameters
  - Stated in terms of level of confidence
    - Example: **95%** confidence, **99%** confidence
    - Can **never** be **100%** confident

Recall that the sample mean will vary from sample to sample

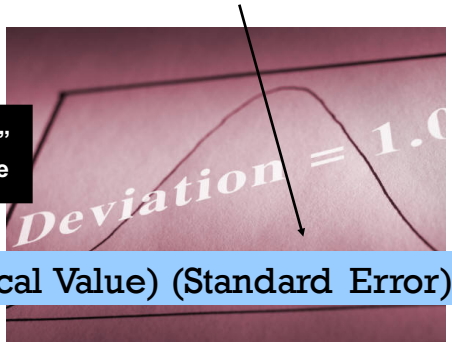
CONFIDENCE INTERVAL ESTIMATES

The general formula for **all** confidence intervals is:

σ / √n or s / √n

Sample Mean or Sample Proportion

The "z" or "t" Critical Value



Point Estimate ± (Critical Value) (Standard Error)

CONFIDENCE LEVEL

Confidence Level

- Confidence in which the interval will contain the unknown population parameter (μ)
- A percentage (less than 100%)

- Suppose confidence level = **95%**
- Also written $(1 - \alpha) = .95$
- A relative frequency interpretation:
  - In the long run, **95%** of all the confidence intervals that **could be** constructed will contain the unknown true parameter (μ)
- A specific interval either will contain or will not contain the true parameter

THE LEVEL OF SIGNIFICANCE

- Because we only select one sample, and **μ** are unknown, we never really know whether the confidence interval includes the true population mean or proportion, or not.

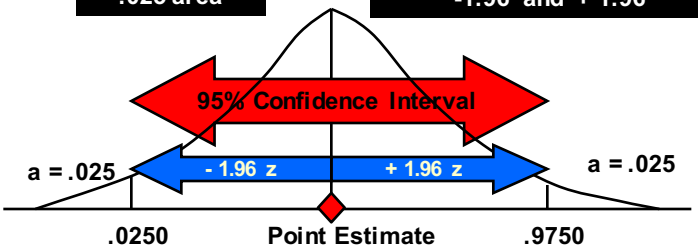
- The ***level of significance***, or "**α**" **risk** is the chance we take that the true population parameter is not contained in the confidence interval.

- Therefore, a 95% confidence interval would have an "α" of 5%

THE LEVEL OF SIGNIFICANCE

If α = .05, then each tail has .025 area

The critical values of "z" that define the "α" areas are -1.96 and + 1.96

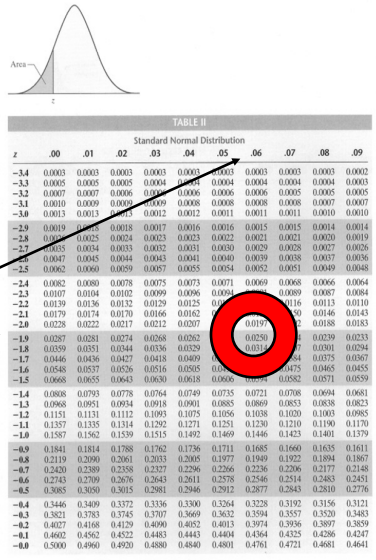


Z	.06
- 1.9	.0250

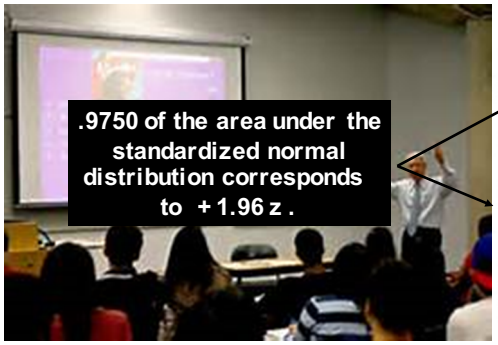
Z	.06
+ 1.9	.9750

" α " is the proportion in the tails of the sampling distribution that is outside the established confidence interval.

The 'z' Table



The 'z' Table



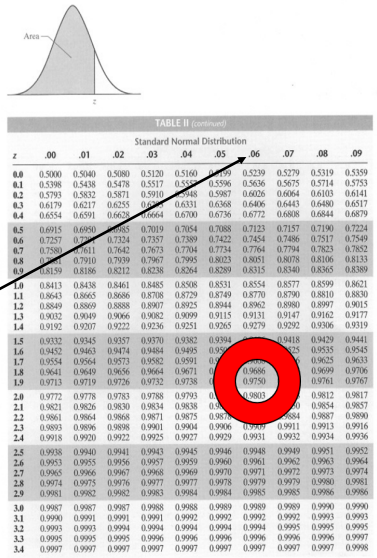


TABLE II (continued)

Standard Normal Distribution

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394			0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495				0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591				0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671		0.9686		0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738		0.9750		0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793		0.9803		0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838				0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875				0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904				0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927				0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945				0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959				0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969				0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977				0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984				0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988				0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992				0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994				0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996				0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997				0.9997	0.9998

CONFIDENCE INTERVAL FOR MEAN

Assumptions

- Population standard deviation σ is **known**
- Population is **normally** distributed
- If population is not normal, use large sample (n > 3)

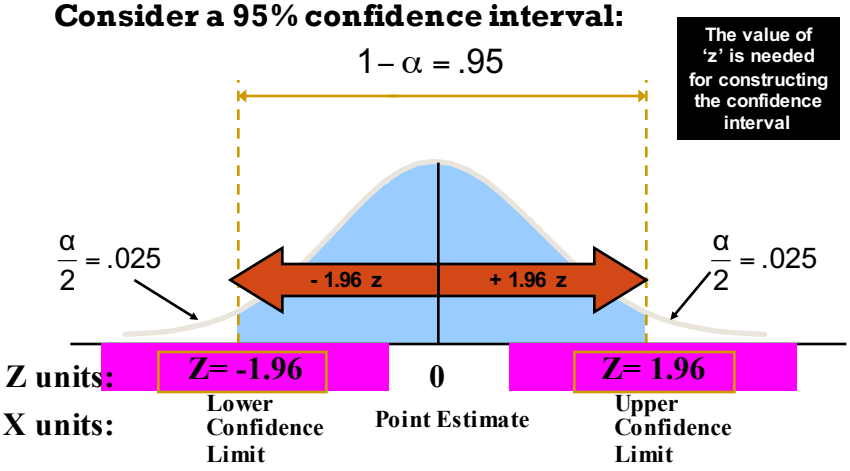
Confidence interval estimate:

$$\bar{X} \pm Z\frac{\sigma}{\sqrt{n}}$$

(where Z is the standardized normal distribution *critical value* for a probability of α/2 in each tail)

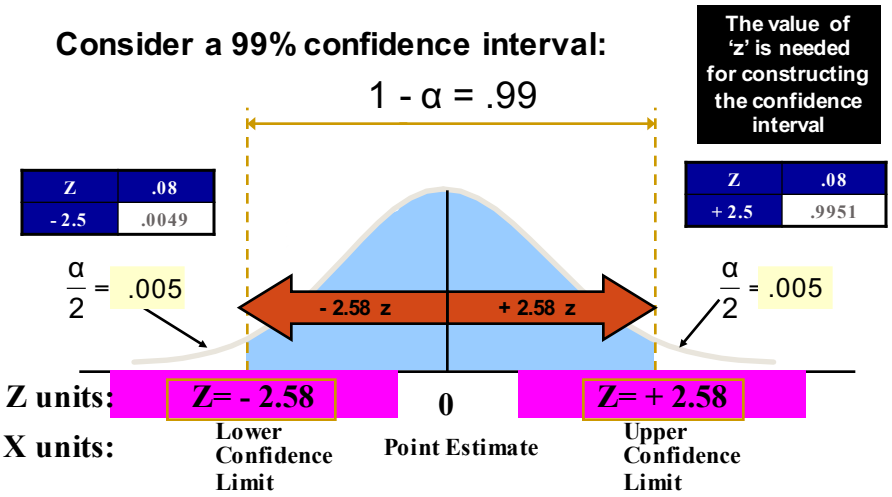
FINDING THE CRITICAL VALUE, Z

Consider a 95% confidence interval:



FINDING THE CRITICAL VALUE, Z

Consider a 99% confidence interval:



FINDING THE CRITICAL VALUE, Z

Commonly used confidence levels are 90%, 95%, and 99%

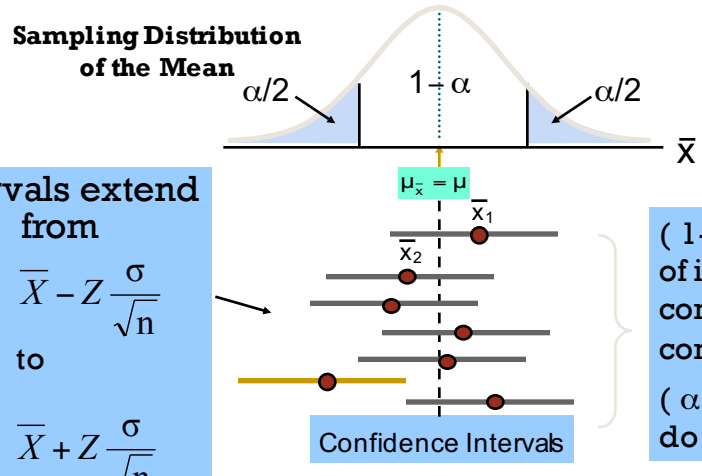
Confidence Level	Confidence Coefficient	'z' Value	$\alpha / 2$ Value
80%	.80	1.28	.1000
90%	.90	1.645	.0500
95%	.95	1.96	.0250
98%	.98	2.33	.0100
99%	.99	2.58	.0050
99.8%	.998	3.08	.0010
99.9%	.999	3.27	.0005

INTERVALS AND LEVEL OF CONFIDENCE

Means vary from sample to sample

Accordingly, confidence intervals of the same percentage will have different lower and upper limits

The true population mean may not be in them at all !



Intervals extend from

$$\overline{X} - Z\frac{\sigma}{\sqrt{n}}$$

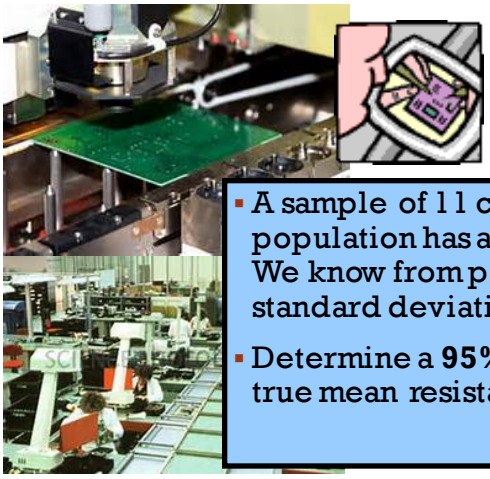
to

$$\overline{X} + Z\frac{\sigma}{\sqrt{n}}$$

(1-α) x 100% of intervals constructed contain μ;

(α) x 100% do not.

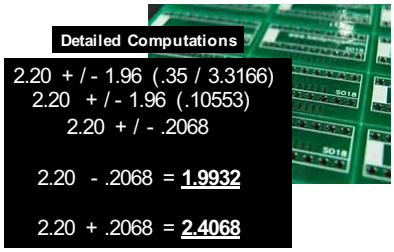
CONFIDENCE INTERVAL FOR MEAN EXAMPLE



- A sample of 11 circuits from a large normal population has a mean resistance of **2.20** ohms. We know from past testing that the population standard deviation is **.35** ohms.
- Determine a **95%** confidence interval for the true mean resistance of the population.

CONFIDENCE INTERVAL FOR MEAN EXAMPLE

$$\overline{X} \pm Z\frac{\sigma}{\sqrt{n}}$$
$$= 2.20 \pm 1.96\,(.35/\sqrt{11})$$
$$= 2.20 \pm .2068$$
$$(1.9932,\ 2.4068)$$



- We are **95%** confident that the true mean resistance is between **1.9932** and **2.4068** ohms
- Although the true mean may or may not be in this interval, **95%** of intervals formed in this manner will contain the true mean

CONFIDENCE INTERVAL FOR MEAN (ST DEV UNKNOWN)

- If the population standard deviation σ is unknown, we can substitute the **sample standard deviation**: (**s**) .
- This introduces extra uncertainty, since '**s**' is variable from sample to sample
- So we use the '***t' distribution*** instead of the normal distribution

"S" is variable from sample to sample, and even more so, in very small samples

CONFIDENCE INTERVAL FOR MEAN (ST DEV UNKNOWN)

Assumptions

- Population standard deviation is unknown
- Population is normally distributed
- If population is not normal, use large sample (n > 30)

Actually, we rarely know the true population standard deviation. We must instead develop a confidence interval using the sample standard deviation

Use **Student's '*t*' Distribution**

The Confidence Interval Estimate:

$$\bar{X} \pm t_{n-1} \frac{S}{\sqrt{n}}$$

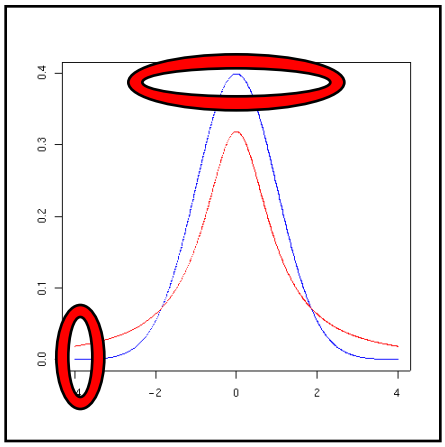
Sample Standard Deviation

Degrees of Freedom

(where t is the critical value of the t distribution with 'n-1' degrees of freedom and an area of $\alpha/2$ in each tail)

STUDENT T DISTRIBUTION

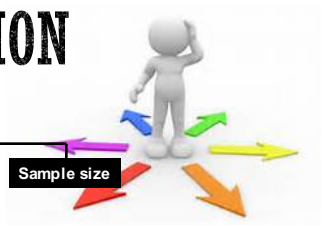
- it is bell-shaped
- has fatter tails
- has flatter center (peak or "kurtosis")
- the sampling distributions of small samples follow the '*t*' distribution
- the smaller the sample size taken, the more variable the '*t*' distribution



STUDENT T DISTRIBUTION

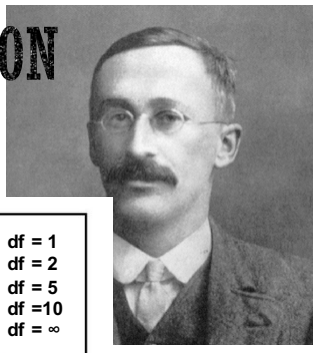
d.f. = n - 1

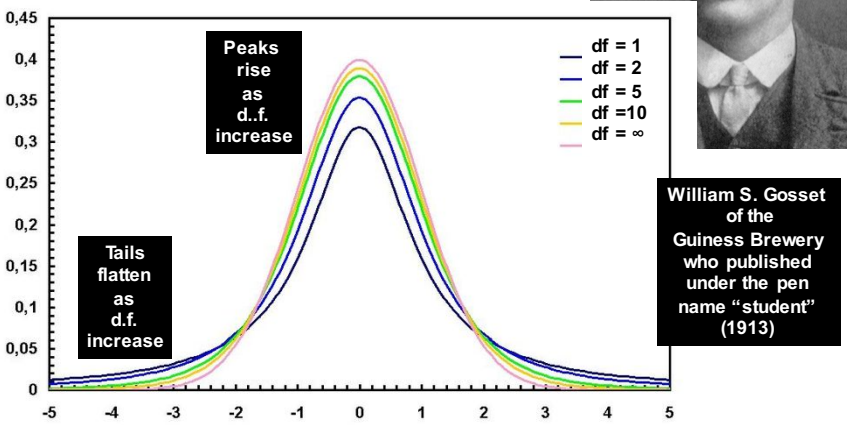
Sample size



- ❑ The '*t*' value depends on *degrees of freedom* (d.f.)
- ❑ The number of observations in the sample that are free to vary after the sample mean has been calculated.
- ❑ The '*t*' distribution changes as the number of degrees of freedom changes.
- ❑ small samples have greater variability and the '*t*' reflects that variability when we are finding areas under the sampling distribution curve.

STUDENT T DISTRIBUTION





William S. Gosset of the Guiness Brewery who published under the pen name "student" (1913)

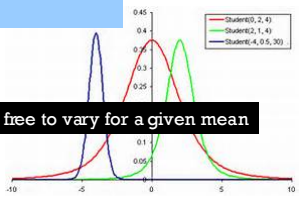
DEGREES OF FREEDOM

Concept: The number of observations that are free to vary after the sample mean has been calculated

Example: Suppose the **mean** of 3 numbers is **8.0**

- **Let X_1 = 7**
- **Let X_2 = 8**
- **What is X_3?**

If the mean of these three values is 8.0, then X_3 must be 9 (i.e., X_3 is not free to vary)

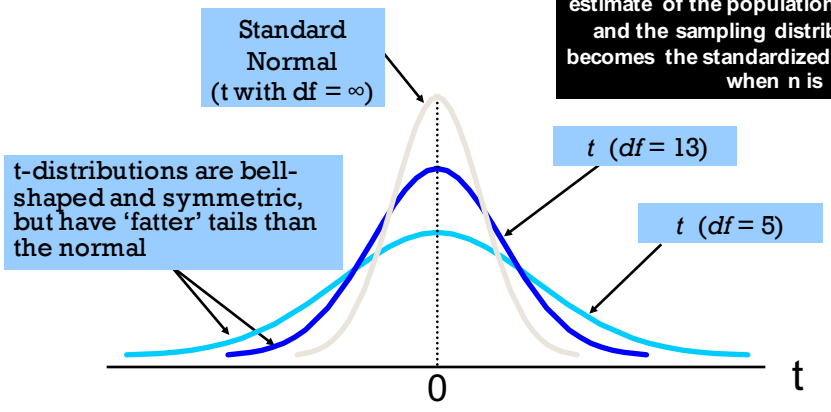


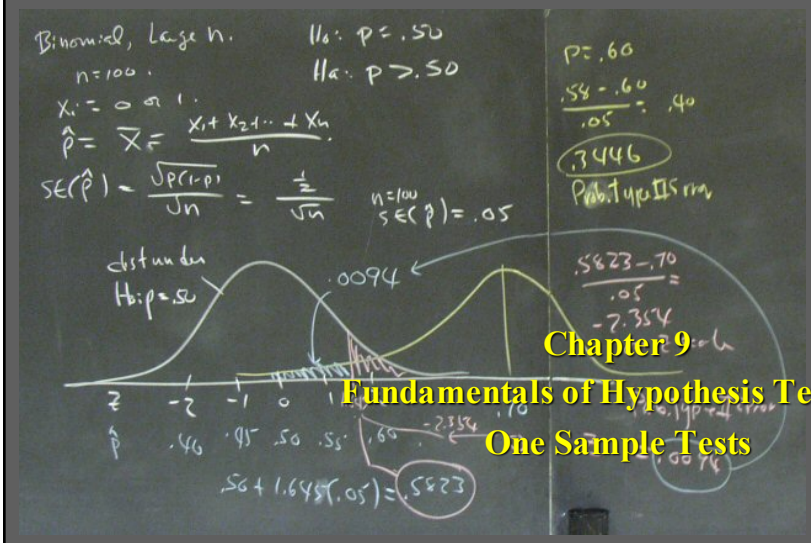
2 values can be any numbers, that is, free to vary, but the third is not free to vary for a given mean

STUDENT'S T DISTRIBUTION

Note: 't' approaches 'Z' as 'n' increases

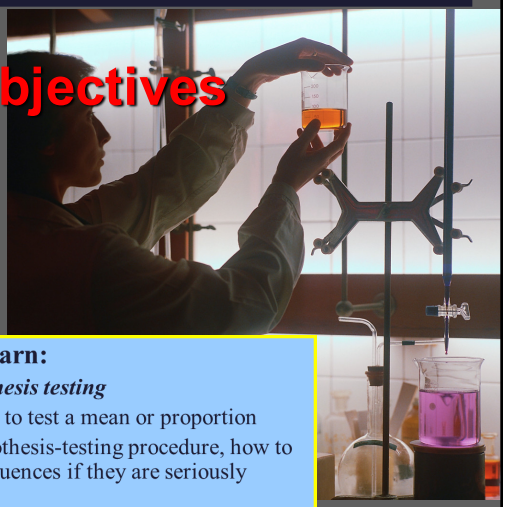
As n and d.f. increase, 's' becomes a better estimate of the population standard deviation, and the sampling distribution eventually becomes the standardized normal distribution when n is large





Chapter 9
Fundamentals of Hypothesis Testing: One Sample Tests

Learning Objectives



In this chapter, you will learn:

- The basic principles of ***hypothesis testing***
- How to use hypothesis testing to test a mean or proportion
- The assumptions of each hypothesis-testing procedure, how to evaluate them, and the consequences if they are seriously violated
- How to avoid the pitfalls involved in hypothesis testing
- Ethical issues involved in hypothesis testing

The Hypothesis

A hypothesis is a claim (assumption) about a population parameter:

population mean (μ)

Example: The mean monthly cell phone bill of this city is $\mu = \$52$

pop. proportion (π)

Example: The proportion of adults in this city with cell phones is $\pi = .68$

The Null Hypothesis, H_0

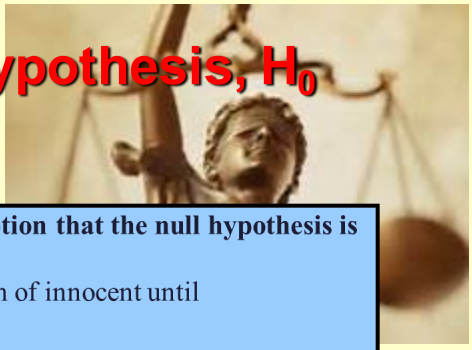
- **States the assumption (numerical) to be tested**
- **Example**: **The mean number of TV sets in U.S. Homes is equal to three.**

$$H_0 : \mu = 3$$

- **Is always about a population parameter, not about a sample statistic.**

it is written in terms of the population

The Null Hypothesis, H_0



- **Begin with the assumption that the null hypothesis is true**
  - Similar to the notion of innocent until proven guilty
- **It refers to the status quo** (THE PREVAILING BELIEF)
- **Always contains " = " , " ≤ " or " ≥ " sign**
- **May or may not be rejected**
- **Ho is never accepted !!!**

The Alternative Hypothesis, H_1

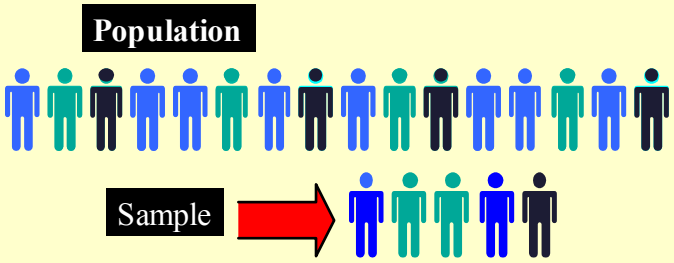
- **Is the opposite of the null hypothesis**
  - e.g., The mean number of TV sets in U.S. homes is ***not*** equal to 3 (H_1: $\mu \neq 3$)
- **Challenges the status quo**
- **Never contains the " = " , " ≤ " or " ≥ " sign**
- **May or may not be proven**
- **Is generally the hypothesis that the researcher is trying to prove**

The Alternative Hypothesis, H_1

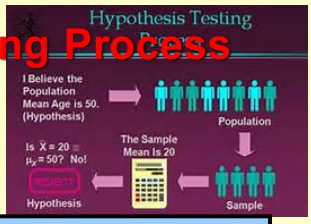
- **Is the opposite of the null hypothesis**
  - e.g., The mean number of TV sets in U.S. homes is not equal to 3 ($H_1: \mu \neq 3$)
- **Always contains " > ", " < ", or " ≠ " sign**

The Hypothesis Testing Process

- **Claim: The population mean age is 50.**
  - **$H_0: \mu = 50$, $H_1: \mu \neq 50$**
- **Sample the population and find sample mean.**

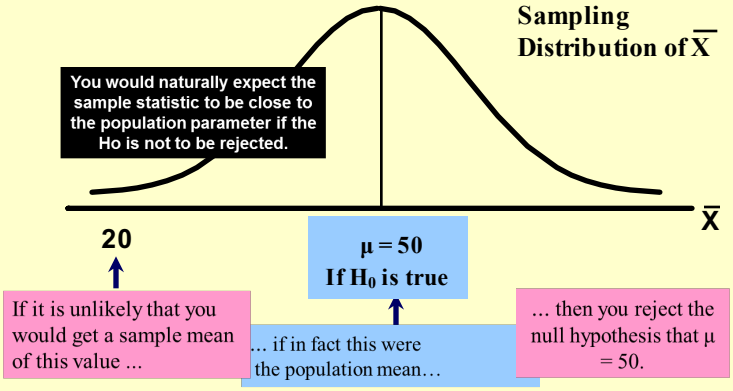


The Hypothesis Testing Process

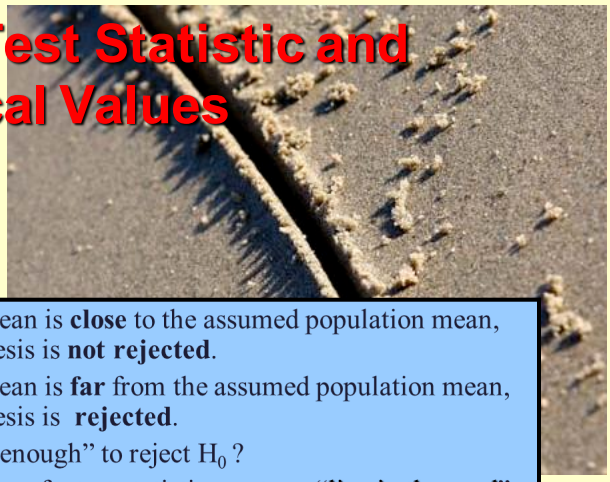


- Suppose the sample mean age was X = 20.
- This is significantly lower than the claimed mean population age of 50.
- If the null hypothesis were true, the probability of getting such a different sample mean would be very small, so you reject the null hypothesis.
- In other words, getting a sample mean of 20 is so unlikely if the population mean was 50, you conclude that the population mean must not be 50.

The Hypothesis Testing Process



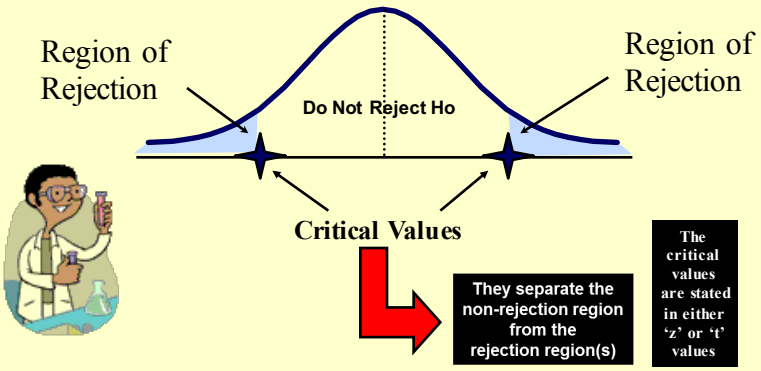
The Test Statistic and Critical Values



- If the sample mean is **close** to the assumed population mean, the null hypothesis is **not rejected**.
- If the sample mean is **far** from the assumed population mean, the null hypothesis is **rejected**.
- How far is "far enough" to reject H_0 ?
- The critical value of a test statistic creates a "**line in the sand**" for decision making.

The Test Statistic and Critical Values

Distribution of the test statistic



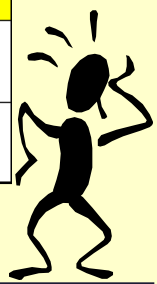
Errors in Decision Making

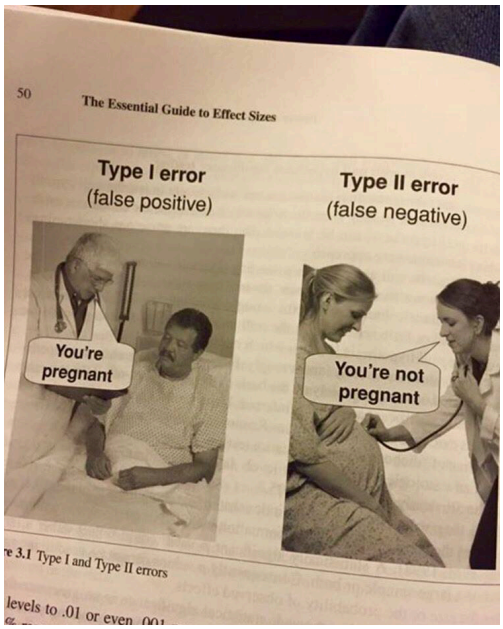
- **Type I Error**
  - **Reject a true null hypothesis**
  - **Considered a serious type of error**
  - **The probability of a Type I Error is 'α'**
  - Called ***level of significance*** of the test
  - Set by researcher in advance:
    within your control or dictated by management
- **Type II Error**
  - **Failure to reject false null hypothesis**
  - **The probability of a Type II Error is 'β'**

Errors in Decision Making

Possible Hypothesis Test Outcomes

	Actual Situation	
Decision	**H_0 True**	**H_0 False**
Do Not Reject H_0	No Error Probability 1 - α	Type II Error Probability β
Reject H_0	Type I Error Probability α	No Error Probability 1 - β





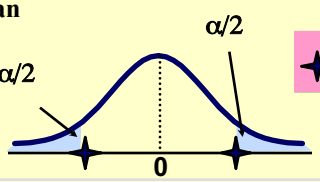
Level of Significance, α

The probability of rejecting Ho when it is true

Claim: The population mean age is 50.

H_0: μ = 50
H_1: μ ≠ 50

Two-tail test

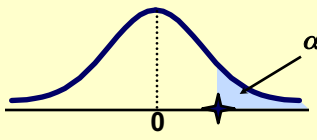


Represents critical value

Rejection region is shaded

H_0: μ ≤ 50
H_1: μ > 50

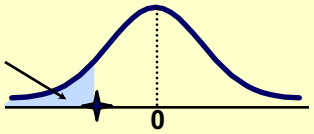
Upper-tail test



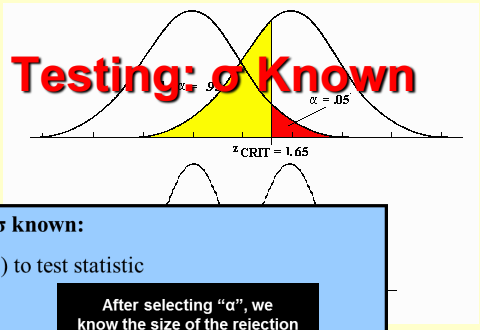
typical values are
α =.01
α =.02
α =.05
α =.10

H_0: μ ≥ 50
H_1: μ < 50

Lower-tail test α



Hypothesis Testing: σ Known



For two tail test for the mean, σ known:

- Convert sample statistic ($\bar{X}$) to test statistic

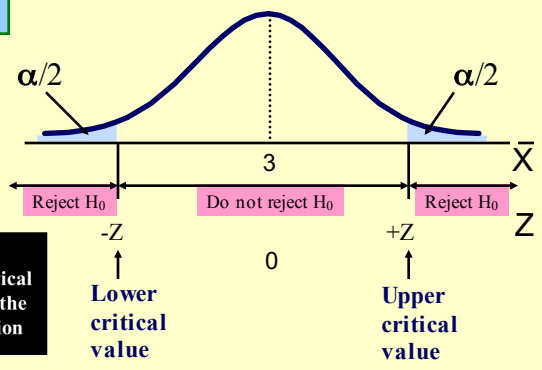
$$Z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

After selecting "α", we know the size of the rejection region, and then we can determine the critical values that define it in terms of the "z" or "t" statistic

- Determine the critical Z values for a specified level of significance α from a table or by using Excel
- Decision Rule: If the test statistic falls in the rejection region, reject H_0 ; otherwise do not reject H_0

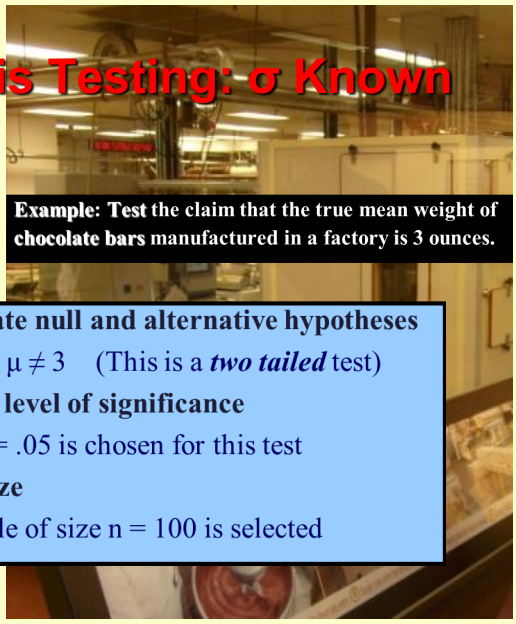
Hypothesis Testing: σ Known

$H_0: \mu = 3$
$H_1: \mu \neq 3$



There are two cutoff values (critical values), defining the regions of rejection

Hypothesis Testing: σ Known



Example: Test the claim that the true mean weight of chocolate bars manufactured in a factory is 3 ounces.

- **State the appropriate null and alternative hypotheses**
  - H_0: $\mu = 3$ H_1: $\mu \neq 3$ (This is a ***two tailed*** test)
- **Specify the desired level of significance**
  - Suppose that $\alpha = .05$ is chosen for this test
- **Choose a sample size**
  - Suppose a sample of size n = 100 is selected

Hypothesis Testing: σ Known

- **Determine the appropriate technique**
  - **σ** is known so this is a **Z** test
- **Set up the critical values**
  - For $\alpha = .05$ the critical Z values are ±1.96
- **Collect the data and compute the test statistic**
  - Suppose the sample results are

n = 100, $\bar{\mathbf{X}} = 2.84$

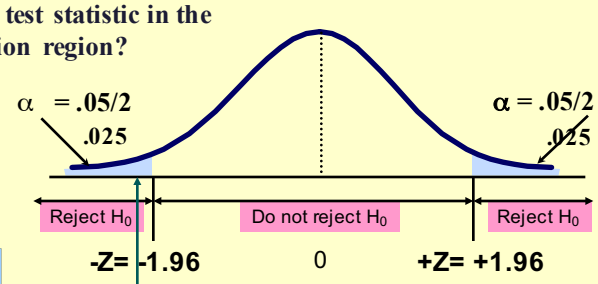
(**σ** = 0.8 is assumed known from past company records)

So the test statistic is:

$$Z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}} = \frac{2.84 - 3}{\frac{0.8}{\sqrt{100}}} = \frac{-.16}{.08} = -2.0$$

Hypothesis Testing: σ Known

Is the test statistic in the rejection region?



a = .05
level of significance

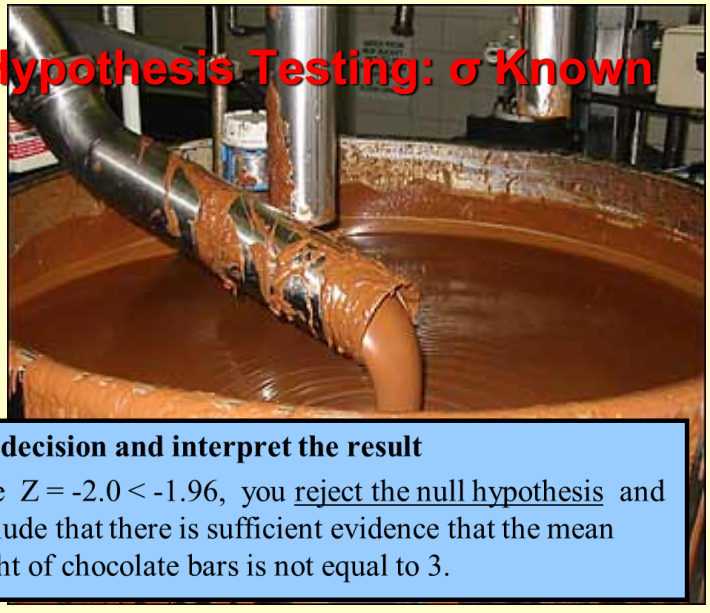
Reject H_0 if $Z < -1.96$ or $Z > 1.96$; otherwise do not reject H_0

Here, $Z = -2.0 < -1.96$, so the test statistic is in the rejection region

Z	0.06
-1.9	.0250

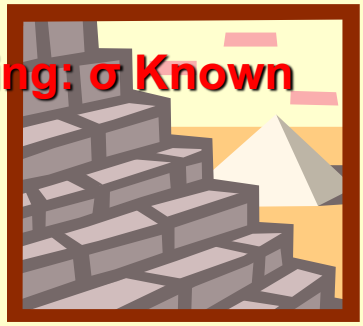
Z	0.06
+1.9	.9750

Hypothesis Testing: σ Known



- **Reach a decision and interpret the result**
  - Since $Z = -2.0 < -1.96$, you reject the null hypothesis and conclude that there is sufficient evidence that the mean weight of chocolate bars is not equal to 3.

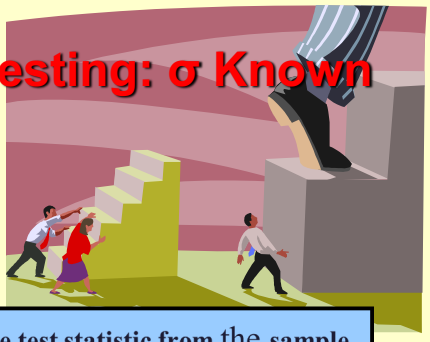
Hypothesis Testing: σ Known



6 Steps of Hypothesis Testing:

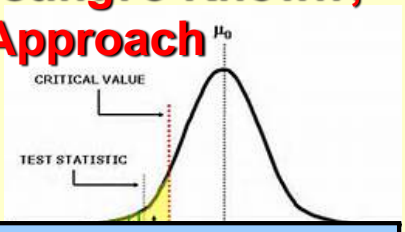
1. State the null hypothesis, H_0 and state the alternative hypotheses, H_1
2. Choose the level of significance, α, and the sample size n.
3. Determine the appropriate statistical technique and the test statistic to use
4. Find the critical values and determine the rejection region(s)

Hypothesis Testing: σ Known



5. Collect data and compute the test statistic from the sample result (either the sample mean or proportion)
6. Compare the test statistic to the critical value to determine whether the test statistic falls in the region of rejection. Make the statistical decision: Reject H_0 if the test statistic falls in the rejection region. Express the decision in the context of the problem

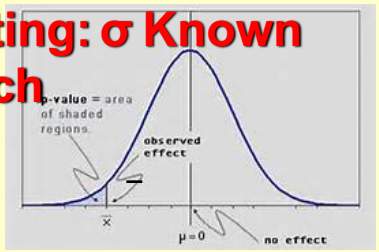
Hypothesis Testing: σ Known , the 'p'- Value Approach



- The p-value is the probability of obtaining a test statistic equal to or more extreme (< or >) than the observed sample value given H_0 is true
  - Also called *observed level of significance**
  - Smallest value of α for which H_0 can be rejected

* Because we can reject or not reject Ho "at a glance" .

Hypothesis Testing: σ Known p-Value Approach



- Convert Sample Statistic (ex. X) to Test Statistic (ex. 'Z' statistic or 't' statistic)
- Obtain the 'p'-value from a table or by using Excel
- Compare the p-value with α
  - If p-value $< \alpha$, reject H_0
  - If p-value $\geq \alpha$, do not reject H_0

For Example:
If 'p' = .03 and 'a' = .05 , reject Ho
If 'p' = .10 and 'a' = .05 , do not reject Ho

Hypothesis Testing: σ Known p-Value Approach

Example: How likely is it to see a sample mean of 2.84 or something lower, or 3.16 or something higher from the mean, if the true mean is $\mu = 3.0$?

Z	0.00
- 2.00	.0228

Z	0.00
+ 2.00	.9772

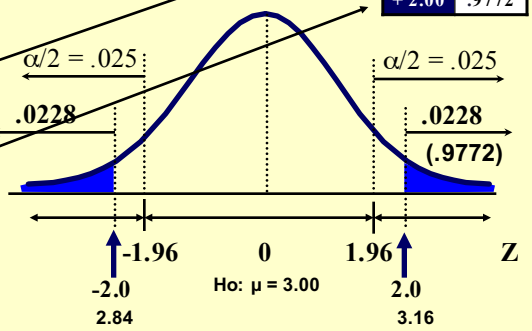
translated to a Z scores :

$$Z = \frac{2.84 - 3.00}{\frac{.8}{\sqrt{100}}} = -\frac{.16}{.08} = -2.00$$

$$Z = \frac{3.16 - 3.00}{\frac{.8}{\sqrt{100}}} = +\frac{.16}{.08} = +2.00$$

p-value

=.0228 + .0228 = .0456



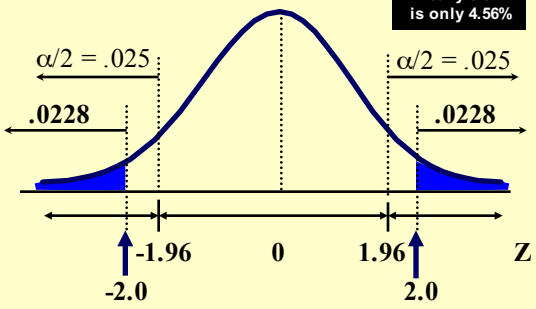
Hypothesis Testing: σ Known p-Value Approach

- Compare the p-value with α
  - If p-value < α, reject H_0
  - If p-value ≥ α, do not reject H_0

The probability of seeing a sample mean of 2.84 or less, or 3.16 or more, if the population mean is really 3.0 is only 4.56%

Here: p-value = .0456
α = .05

Since .0456 < .05, you reject the null hypothesis that μ = 3.0



Hypothesis Testing: σ Known Confidence Interval Connections

Simpler, faster hypothesis test

If a = .05, construct a 95% confidence interval

Can only be used with a two-tail hypothesis test !

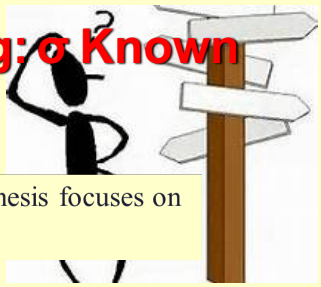
- For $\overline{X} = 2.84$, $\sigma = 0.8$ and $n = 100$, the 95% confidence interval is :

$$2.84 - (1.96)\frac{0.8}{\sqrt{100}} \quad \text{to} \quad 2.84 + (1.96)\frac{0.8}{\sqrt{100}}$$

$$2.6832 \le \mu \le 2.9968$$

Since this interval does not contain the hypothesized mean (3.0), you reject the null hypothesis at $\alpha = .05$

Hypothesis Testing: σ Known One Tail Tests



- In many cases, the alternative hypothesis focuses on a ***particular*** direction

$H_0: \mu \ge 3$
$H_1: \mu < 3$

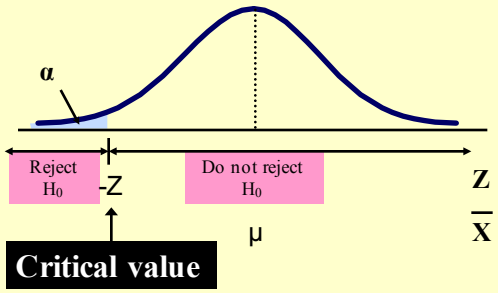
→ **This is a lower-tail test since the alternative hypothesis is focused on the *lower tail* below the mean of 3**

$H_0: \mu \le 3$
$H_1: \mu > 3$

→ **This is an upper-tail test since the alternative hypothesis is focused on the *upper tail* above the mean of 3**

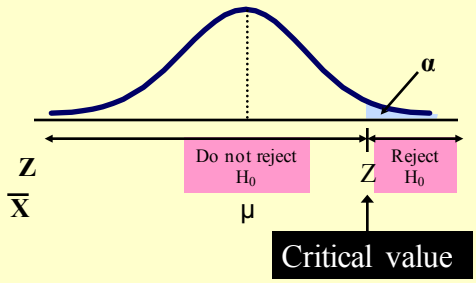
Hypothesis Testing: σ Known Lower Tail Tests

- There is only one critical value, since the rejection area is in only one tail.



Hypothesis Testing: σ Known Upper Tail Tests

- There is only one critical value, since the rejection area is in only one tail.



Hypothesis Testing: σ Known Upper Tail Test Example

A phone industry manager thinks that customer monthly cell phone bills have increased, and now average more than $52 per month. The company wishes to test this claim. Past company records indicate that the standard deviation is about $10.

Form the hypothesis test:

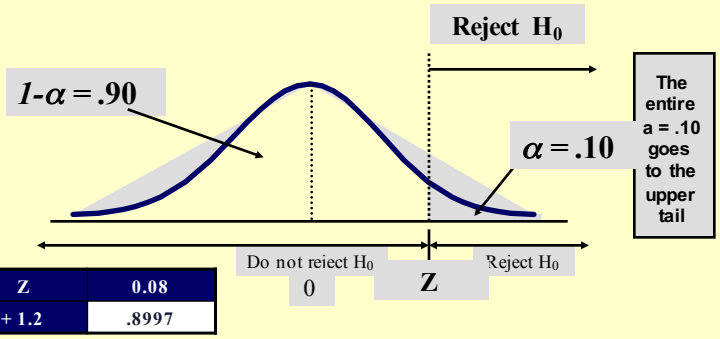
H_0: $\mu \leq 52$ the mean is less than or equal to than $52 per month

H_1: $\mu > 52$ the mean is greater than $52 per month

(i.e., sufficient evidence exists to support the manager's claim)

Hypothesis Testing: σ Known Upper Tail Test Example

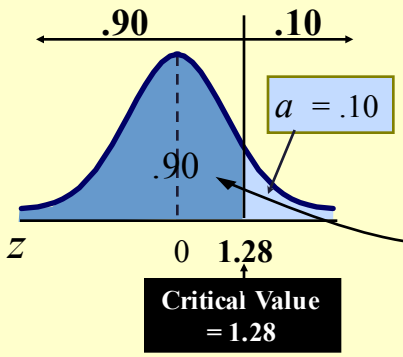
- Suppose that $\alpha = .10$ is chosen for this test
- Find the rejection region:



Z	0.08
+ 1.2	.8997

Hypothesis Testing: σ Known Upper Tail Test Example

What is Z given $a = 0.10$?



Z	.07	.08	.09
1.1	.8790	.8810	.8830
1.2	.8980	.8997	.9015
1.3	.9147	.9162	.9177

Hypothesis Testing: σ Known Upper Tail Test Example

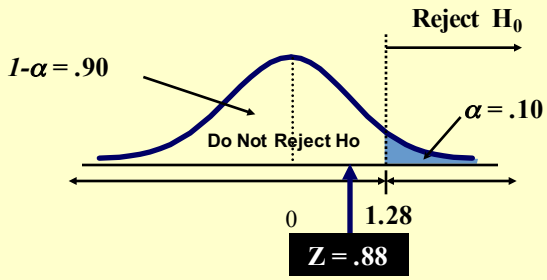
- Obtain sample and compute the test statistic.
- Suppose a sample is taken with the following results:
- n = 64
- $\bar{X} = 53.1$
- (σ = 10 was assumed known from past company records)

Then the test statistic is:

$$Z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}} = \frac{53.1 - 52}{\frac{10}{\sqrt{64}}} = 0.88$$

Hypothesis Testing: σ Known Upper Tail Test Example

- Reach a decision and interpret the result:



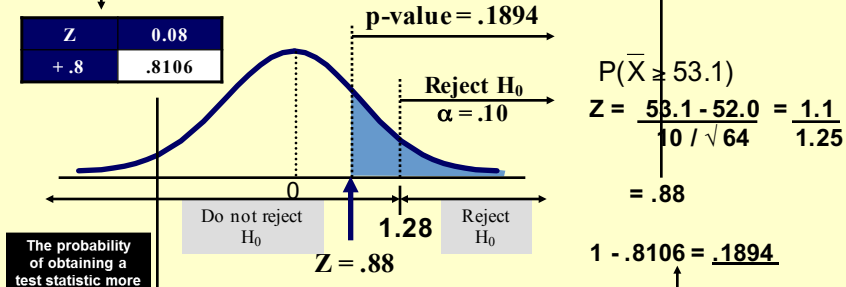
Do not reject H_0 since Z = 0.88 ≤ 1.28

i.e.: there is not sufficient evidence that the mean bill is greater than $52

Hypothesis Testing: σ Known Upper Tail Test Example

- Calculate the p-value and compare to 'α'

Z	0.08
+ .8	.8106



$P(\bar{X} \geq 53.1)$

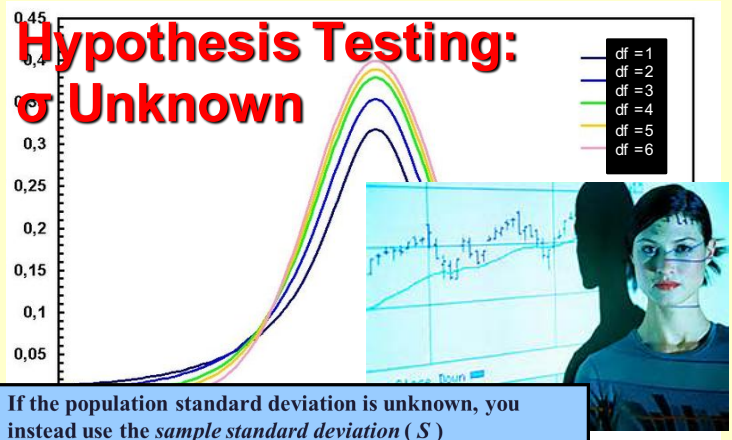
$Z = \frac{53.1 - 52.0}{10 / \sqrt{64}} = \frac{1.1}{1.25} = .88$

1 - .8106 = .1894

The probability of obtaining a test statistic more extreme than the observed sample value, given that the Ho is true, is 18.94%

Do not reject H_0 since p-value = .1894 > α = .10

Hypothesis Testing: σ Unknown



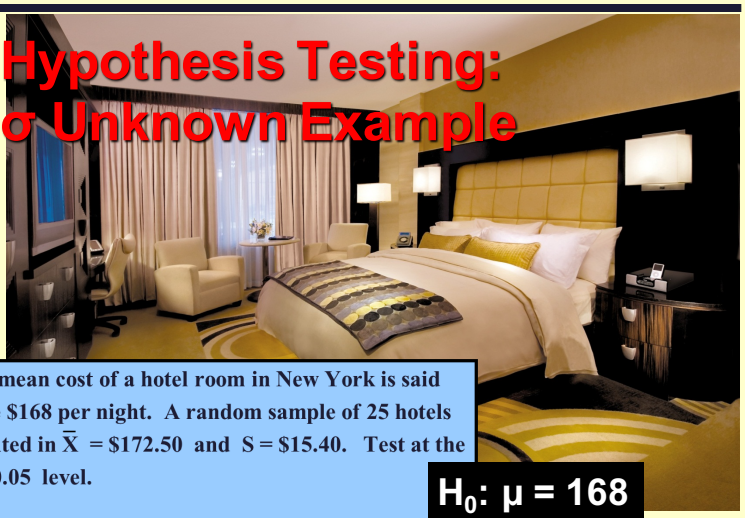
- If the population standard deviation is unknown, you instead use the *sample standard deviation* (*S*)
- Because of this change, you use the *t distribution* instead of the Z distribution to test the null hypothesis about the mean.
- All other steps, concepts, and conclusions are the same.

Hypothesis Testing: σ Unknown

- Recall that the 't' test statistic with n-1 degrees of freedom is:

$$t_{n-1} = \frac{\overline{X} - \mu}{\frac{S}{\sqrt{n}}}$$

Hypothesis Testing: σ Unknown Example

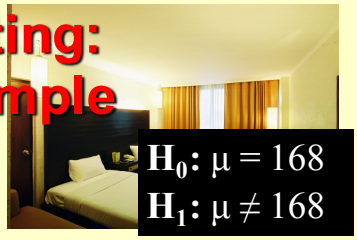


The mean cost of a hotel room in New York is said to be \$168 per night. A random sample of 25 hotels resulted in $\bar{X}$ = \$172.50 and S = \$15.40. Test at the α = 0.05 level.

A stem-and-leaf display and a normal probability plot indicate the data are approximately normally distributed

H_0: μ = 168
H_1: μ ≠ 168

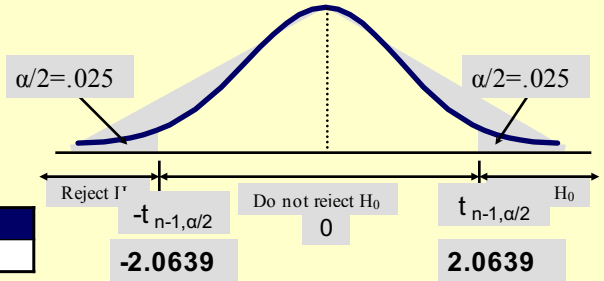
Hypothesis Testing: σ Unknown Example



H_0: μ = 168
H_1: μ ≠ 168

- α = 0.05
- n = 25
- σ is unknown, so use a 't' statistic
- Critical Value: $t_{24} = \pm 2.0639$

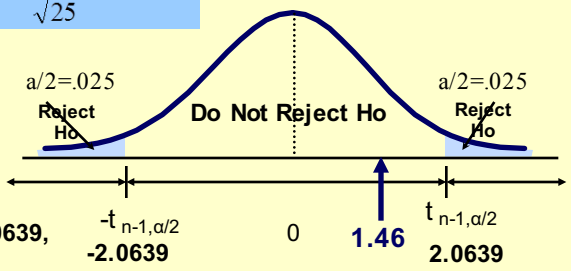
Determine the regions of rejection



t	.025
24 df	2.0639

Hypothesis Testing: σ Unknown Example

$$t_{n-1} = \frac{\bar{X} - \mu}{\frac{S}{\sqrt{n}}} = \frac{172.50 - 168}{\frac{15.40}{\sqrt{25}}} = 1.46$$



Since $t_{df = 24,\ a=.025} = 1.46 < 2.0639$,

Do not reject H_0: not sufficient evidence that true mean cost is different from $168

Hypothesis Testing: Connection to Confidence Intervals

- For $\bar{X} = 172.5$, $S = 15.40$ and $n = 25$, the 95% confidence interval is:

$$172.5 - (2.0639)\frac{15.4}{\sqrt{25}} \quad \text{to} \quad 172.5 + (2.0639)\frac{15.4}{\sqrt{25}}$$

$$166.14 \le \mu \le 178.86$$

Since this interval contains the hypothesized mean (168), you do not reject the null hypothesis at $\alpha = .05$

Hypothesis Testing: σ Unknown

- Recall that you assume that the sample statistic comes from a random sample from a normal distribution.
- If the sample size is small (< 30), you should use a box-and-whisker plot or a normal probability plot to assess whether the assumption of normality is valid.
- If the sample size is large, the central limit theorem applies and the sampling distribution of the mean will be normal.

Hypothesis Testing: Proportions

π

- Involves ***categorical variables*** (yes/no , male/female)
- Two possible outcomes
  - "**Success**" (possesses a certain characteristic)
  - "**Failure**" (does not possesses that characteristic)
- Fraction or proportion of the population in the "success" category is denoted by **π**

Hypothesis Testing: Proportions

- **Sample proportion in the success category is denoted by p**

$$p = \frac{X}{n} = \frac{\text{number of successes in sample}}{\text{sample size}}$$

- When both $n\pi$ and $n(1-\pi)$ are at least 5, p can be approximated by a normal distribution with mean and standard deviation

$$\mu_p = \pi \qquad \sigma_p = \sqrt{\frac{\pi(1-\pi)}{n}}$$

Hypothesis Testing: Proportions

- **The sampling distribution of p is approximately normal, so the test statistic is a 'Z' value:**

$$Z = \frac{p - \pi}{\sqrt{\frac{\pi(1-\pi)}{n}}}$$

Hypothesis Testing: Proportions Example

A marketing company claims that it receives 8% responses from its mailings. To test this claim, a random sample of 500 were surveyed with 30 responses. Test at the $\alpha = .05$ significance level.

First, check:

$n\pi = (500)(.08) = 40$

$n(1-\pi) = (500)(.92) = 460$

Hypothesis Testing: Proportions Example

$H_0: \pi = .08 \quad H_1: \pi \neq .08$

$\alpha = .05$

$n = 500, \quad p = .06$

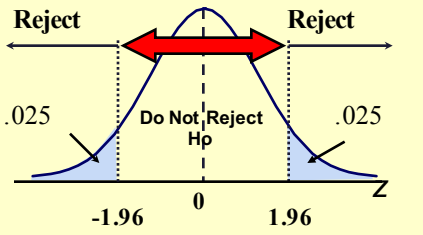
Critical Values: ± 1.96

30 / 500 = .06 = p (sample proportion)

Z	0.06
- 1.9	.0250

Z	0.06
+ 1.9	.9750

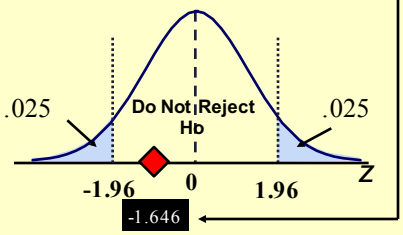
Determine region of rejection



Hypothesis Testing: Proportions Example

Test Statistic:

$$Z = \frac{p - \pi}{\sqrt{\frac{\pi(1-\pi)}{n}}} = \frac{.06 - .08}{\sqrt{\frac{.08(1-.08)}{500}}} = -1.648$$



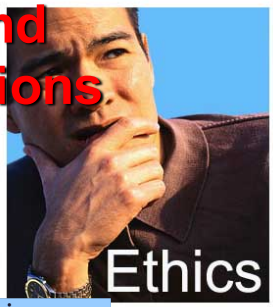
Decision:

Do not reject H_0 at $\alpha = .05$

Conclusion:

There isn't sufficient evidence to reject the company's claim of 8% response rate.

Potential Pitfalls and Ethical Considerations



- Use randomly collected data to reduce selection biases
- Do not use human subjects without informed consent
- Choose the level of significance, *α*, before data collection
- Do not employ "data snooping" to choose between one-tail and two-tail test, or to determine the level of significance
- Do not practice "data cleansing" to hide observations that do not support a stated hypothesis
- Report all pertinent findings

Chapter Summary

In this chapter, we have

- Addressed hypothesis testing methodology
- Performed 'Z' Test for the mean (σ known)
- Discussed ***critical value*** and ***p-value*** approaches to hypothesis testing.
- Performed one-tail and two-tail tests
- Performed 't' test for the mean (σ unknown)
- Performed 'Z' test for the proportion
- Discussed pitfalls and ethical issues

T-TESTS AND ANALYSIS OF VARIANCE

ONE SAMPLE T-TEST

ONE SAMPLE T-TEST

- Used to test whether the population mean is different from a specified value.
- Example: Is the mean height of 12 year old girls greater than 60 inches?

3

STEP 1: FORMULATE THE HYPOTHESES

- The population mean is not equal to a specified value.

  H_0: $\mu = \mu_0$

  H_a: $\mu \neq \mu_0$

- The population mean is greater than a specified value.

  H_0: $\mu = \mu_0$

  H_a: $\mu > \mu_0$

- The population mean is less than a specified value.

  H_0: $\mu = \mu_0$

  H_a: $\mu < \mu_0$

4

STEP 2: CHECK THE ASSUMPTIONS

- The sample is random.
- The population from which the sample is drawn is either normal or the sample size is large.

5

STEPS 3-5

- Step 3: Calculate the test statistic: $t = \dfrac{\bar{y} - \mu_0}{s/\sqrt{n}}$

Where $s = \sqrt{\dfrac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}}$

- Step 4: Calculate the p-value based on the appropriate alternative hypothesis.
- Step 5: Write a conclusion.

6

IRIS EXAMPLE

- A researcher would like to know whether the mean sepal width of a variety of irises is different from 3.5 cm.
- The researcher randomly measures the sepal width of 50 irises.
- Step 1: Hypotheses

  $H_0: \mu = 3.5$ cm

  $H_a: \mu \neq 3.5$ cm

7

TWO SAMPLE T-TEST

TWO SAMPLE T-TEST

- Two sample t-tests are used to determine whether the population mean of one group is equal to, larger than or smaller than the population mean of another group.
- Example: Is the mean cholesterol of people taking drug A lower than the mean cholesterol of people taking drug B?

9

STEP 1: FORMULATE THE HYPOTHESES

- The population means of the two groups are not equal.

  $H_0: \mu_1 = \mu_2$

  $H_a: \mu_1 \neq \mu_2$
- The population mean of group 1 is greater than the population mean of group 2.

  $H_0: \mu_1 = \mu_2$

  $H_a: \mu_1 > \mu_2$
- The population mean of group 1 is less than the population mean of group 2.

  $H_0: \mu_1 = \mu_2$

  $H_a: \mu_1 < \mu_2$

10

STEP 2: CHECK THE ASSUMPTIONS

- The two samples are random and independent.
- The populations from which the samples are drawn are either normal or the sample sizes are large.
- The populations have the same standard deviation.

11

STEPS 3-5

- Step 3: Calculate the test statistic $t = \dfrac{\bar{y}_1 - \bar{y}_2}{s_p\sqrt{\dfrac{1}{n_1} + \dfrac{1}{n_2}}}$

where $s_p = \sqrt{\dfrac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$

- Step 4: Calculate the appropriate p-value.
- Step 5: Write a conclusion.

12

TWO SAMPLE EXAMPLE

- A researcher would like to know whether the mean sepal width of setosa irises is different from the mean sepal width of versicolor irises.

- Step 1 Hypotheses:

  $H_0: \mu_{setosa} = \mu_{versicolor}$

  $H_a: \mu_{setosa} \neq \mu_{versicolor}$

- Step 5 Conclusion: There is strong evidence (p-value < 0.0001) that the mean sepal widths for the two varieties are different.

13

PAIRED T-TEST

PAIRED T-TEST

- The paired t-test is used to compare the means of two dependent samples.

- Example:

  A researcher would like to determine if background noise causes people to take longer to complete math problems. The researcher gives 20 subjects two math tests one with complete silence and one with background noise and records the time each subject takes to complete each test.

15

STEP 1: FORMULATE THE HYPOTHESES

- The population mean difference is not equal to zero.

  H_0: $\mu_{difference} = 0$

  H_a: $\mu_{difference} \neq 0$

- The population mean difference is greater than zero.

  H_0: $\mu_{difference} = 0$

  H_a: $\mu_{difference} > 0$

- The population mean difference is less than a zero.

  H_0: $\mu_{difference} = 0$

  H_a: $\mu_{difference} < 0$

16

STEP 2: CHECK THE ASSUMPTIONS

- The sample is random.
- The data is matched pairs.
- The differences have a normal distribution or the sample size is large.

17

STEPS 3-5

- Step 3: Calculate the test Statistic:

$$t = \frac{\bar{d} - 0}{s_d / \sqrt{n}}$$

Where d bar is the mean of the differences and s_d is the standard deviations of the differences.

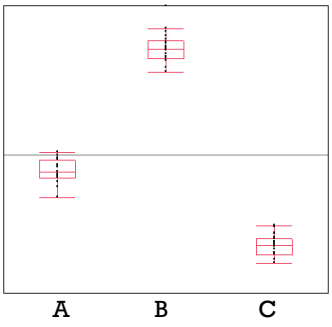
- Step 4: Calculate the p-value.
- Step 5: Write a conclusion.

18

ONE-WAY ANALYSIS OF VARIANCE

ONE-WAY ANOVA

- ANOVA is used to determine whether three or more populations have different distributions.



ANOVA STRATEGY

- The first step is to use the ANOVA F test to determine if there are any significant differences among means.
- If the ANOVA F test shows that the means are not all the same, then follow up tests can be performed to see which pairs of means differ.

21

ONE-WAY ANOVA MODEL

$y_{ij} = \mu_i + \varepsilon_{ij}$

Where

y_{ij} is the response of the jth trial on the ith factor level

μ_i is the mean of the ith group

$\varepsilon_{ij} \sim N(0, \sigma^2)$

$i = 1, \ldots, r$

$j = 1, \ldots, n_i$

In other words, for each group the observed value is the group mean plus some random variation.

22

ONE-WAY ANOVA HYPOTHESIS

- Step 1: We test whether there is a difference in the means.

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_r$$
$$H_a : \text{The } \mu_i \text{ are not all equal.}$$

23

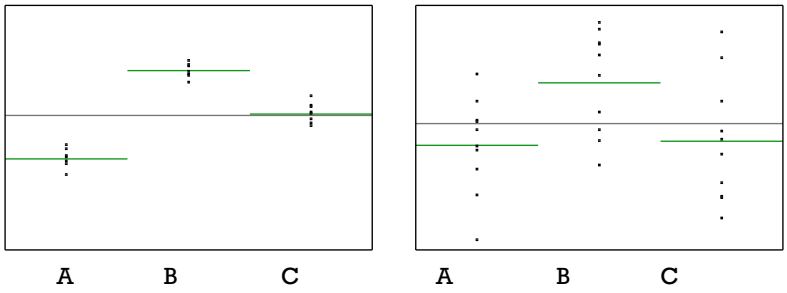
STEP 2: CHECK ANOVA ASSUMPTIONS

- The samples are random and independent of each other.
- The populations are normally distributed.
- The populations all have the same variance.

- The ANOVA F test is robust to the assumptions of normality and equal variances.

24

STEP 3: ANOVA F TEST

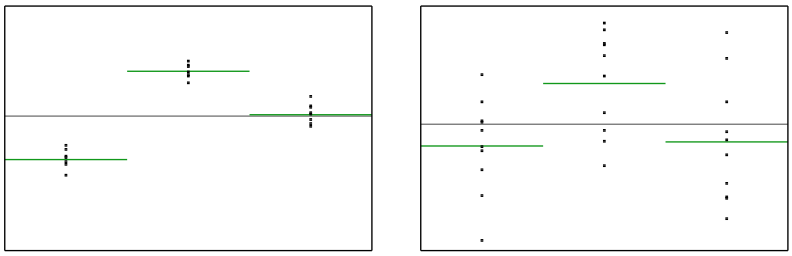


Compare the variation within the samples to the variation between the samples.

25

ANOVA TEST STATISTIC

$$F = \frac{\text{Variation between Groups}}{\text{Variation within Groups}} = \frac{MSG}{MSE}$$



Variation within groups small compared with variation between groups → Large F

Variation within groups large compared with variation between groups → Small F

26

MSG

- The mean square for groups, MSG, measures the variability of the sample averages.
- SSG stands for sums of squares groups.

$$MSG = \frac{SSG}{r-1} = \frac{n_1(\bar{y}_{1\cdot} - \bar{y}_{\cdot\cdot})^2 + n_2(\bar{y}_{2\cdot} - \bar{y}_{\cdot\cdot})^2 + \ldots + n_r(\bar{y}_{1\cdot} - \bar{y}_{\cdot\cdot})^2}{r-1}$$

27

MSE

- Mean square error, MSE, measures the variability within the groups.
- SSE stands for sums of squares error.

$$MSE = \frac{SSE}{n-r} = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2 + \cdots + (n_r-1)s_r^2}{n-r}$$

Where

$$s_i = \sqrt{\frac{\sum_{j=1}^{n_i}(y_{ij} - y_{i\cdot})}{n_i - 1}}$$

28

STEPS 4-5

- Step 4: Calculate the p-value.
- Step 5: Write a conclusion.

29

ANOVA EXAMPLE

- A researcher would like to determine if three drugs provide the same relief from pain.
- 60 patients are randomly assigned to a treatment (20 people in each treatment).
- Step 1: Formulate the Hypotheses

H_0: $\mu_{Drug\ A} = \mu_{Drug\ B} = \mu_{Drug\ C}$

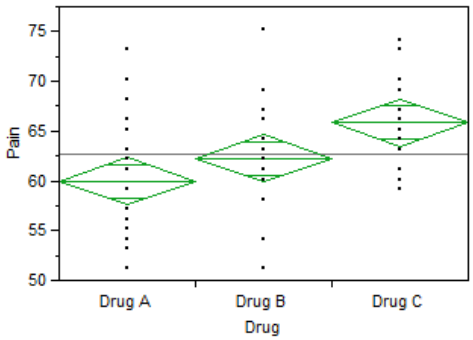
H_a : The μ_i are not all equal.

30

STEPS 2-4

Y, Response: Pain

X, Factor: Drug



31

STEP 5: WRITE A CONCLUSION.

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio	Prob > F
Drug	2	348.3000	174.150	6.1875	0.0037*
Error	57	1604.3000	28.146		
C. Total	59	1952.6000			

- Step 5 Conclusion: There is strong evidence that the drugs are not all the same.

32

FOLLOW-UP TEST

- The p-value of the overall F test indicates that the level of pain is not the same for patients taking drugs A, B and C.
- We would like to know which pairs of treatments are different.
- One method is to use Tukey's HSD (honestly significant differences).

TUKEY TESTS

- Tukey's test simultaneously tests

$$H_0 : \mu_i = \mu_{i'}$$
$$H_a : \mu_i \neq \mu_{i'}$$

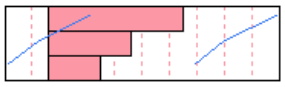
for all pairs of factor levels. Tukey's HSD controls the overall type I error.

TUKEY TESTS

Level		Mean
Drug C	A	65.850000
Drug B	A B	62.250000
Drug A	B	60.000000

Levels not connected by same letter are significantly different.

Level	- Level	Difference	Lower CL	Upper CL
Drug C	Drug A	5.850000	1.81283	9.887174
Drug C	Drug B	3.600000	-0.43717	7.637174
Drug B	Drug A	2.250000	-1.78717	6.287174

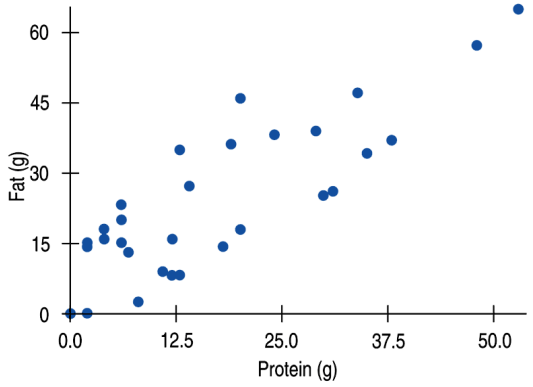


- The output shows that drugs A and C are significantly different.

SIMPLE LINEAR REGRESSION

FAT VERSUS PROTEIN: AN EXAMPLE

- The following is a scatterplot of total *fat* versus *protein* for 30 items on the Burger King menu:



THE LINEAR MODEL

- Correlation says "There seems to be a linear association between these two variables," but it doesn't tell *what that association is*.
- We can say more about the linear relationship between two quantitative variables with a model.
- A model simplifies reality to help us understand underlying patterns and relationships.

THE LINEAR MODEL (CONT.)

- The linear model is just an equation of a straight line through the data.
  - The points in the scatterplot don't all line up, but a straight line can summarize the general pattern.
  - The linear model can help us understand how the values are associated.

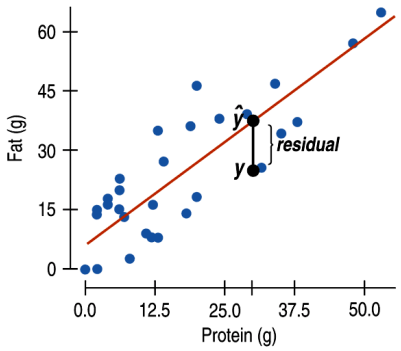
RESIDUALS

- The model won't be perfect, regardless of the line we draw.
- Some points will be above the line and some will be below.
- The estimate made from a model is the predicted value.

$$residual = observed - predicted = y - \hat{y}$$

RESIDUALS (CONT.)

- A negative residual means the predicted value's too big (an overestimate).
- A positive residual means the predicted value's too small (an underestimate).



"BEST FIT" MEANS LEAST SQUARES

- Some residuals are positive, others are negative, and, on average, they cancel each other out.
- So, we can't assess how well the line fits by adding up all the residuals.
- Similar to what we did with deviations, we square the residuals and add the squares.
- The smaller the sum, the better the fit.
- The line of best fit is the line for which the sum of the squared residuals is smallest.

THE LEAST SQUARES LINE

- We write our model as

$$\hat{y} = b_0 + b_1 x$$

- This model says that our *predictions* from our model follow a straight line.
- If the model is a good one, the data values will scatter closely around it.

THE LEAST SQUARES LINE (CONT.)

- In our model, we have a slope (b_1):
  - The slope is built from the correlation and the standard deviations:

$$b_1 = r\frac{s_y}{s_x}$$

  - Our slope is always in units of *y* per unit of *x*.

THE LEAST SQUARES LINE (CONT.)

- In our model, we also have an intercept (b_0).
  - The intercept is built from the means and the slope:

$$b_0 = \bar{y} - b_1\bar{x}$$

- Our intercept is always in units of *y*.

THE LEAST SQUARES LINE (CONT.)

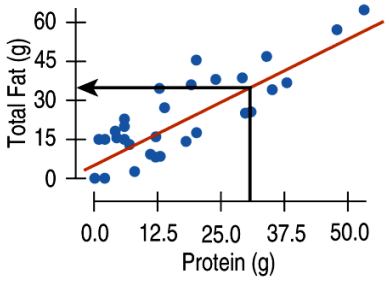
- Since regression and correlation are closely related, we need to check the same conditions for regressions as we did for correlations:
  - Quantitative Variables Condition
  - Straight Enough Condition
  - Outlier Condition

FAT VERSUS PROTEIN: AN EXAMPLE

- The regression line for the Burger King data fits the data well:
  - The equation is

$$\widehat{fat} = 6.8 + 0.97\,protein.$$

  - The *predicted fat* content for a BK Whopper sandwich is 6.8 + 0.97(30) = 35.9 grams of fat.



RESIDUALS REVISITED

- The linear model assumes that the relationship between the two variables is a perfect straight line. The residuals are the part of the data that *hasn't* been modeled.

Data = Model + Residual

or (equivalently)

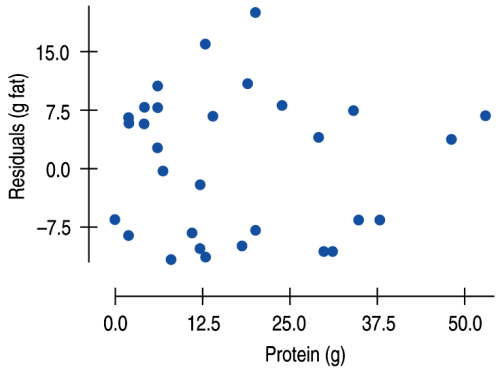
Residual = Data – Model

Or, in symbols,

$$e = y - \hat{y}$$

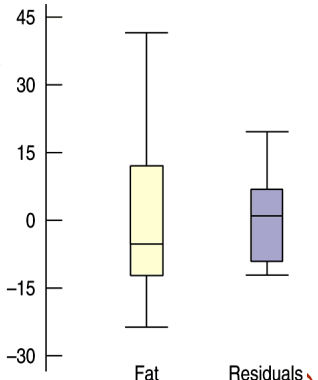
RESIDUALS REVISITED (CONT.)

- The residuals for the BK menu regression look appropriately boring:



R^2—THE VARIATION ACCOUNTED FOR

- The variation in the residuals is the key to assessing how well the model fits.
- In the BK menu items example, total *fat* has a standard deviation of 16.4 grams. The standard deviation of the residuals is 9.2 grams.



R^2—THE VARIATION ACCOUNTED FOR (CONT.)

- If the correlation were 1.0 and the model predicted the *fat* values perfectly, the residuals would all be zero and have no variation.
- As it is, the correlation is 0.83—not perfection.
- However, we did see that the model residuals had less variation than total *fat* alone.
- We can determine how much of the variation is accounted for by the model and how much is left in the residuals.

R^2—THE VARIATION ACCOUNTED FOR (CONT.)

- The squared correlation, r^2, gives the fraction of the data's variance accounted for by the model.
- Thus, $1 - r^2$ is the fraction of the original variance left in the residuals.
- For the BK model, $r^2 = 0.83^2 = 0.69$, so 31% of the variability in total *fat* has been left in the residuals.

R^2—THE VARIATION ACCOUNTED FOR (CONT.)

- All regression analyses include this statistic, although by tradition, it is written R^2 (pronounced "*R*-squared"). An R^2 of 0 means that none of the variance in the data is in the model; all of it is still in the residuals.
- When interpreting a regression model you need to *Tell* what R^2 means.
  - In the BK example, 69% of the variation in total *fat* is accounted for by the model.

ASSUMPTIONS AND CONDITIONS

- Quantitative Variables Condition:
  - Simple linear regression can only be done on two quantitative variables, so make sure to check this condition.
- Straight Enough Condition:
  - The linear model assumes that the relationship between the variables is linear.
  - A scatterplot will let you check that the assumption is reasonable.

ASSUMPTIONS AND CONDITIONS (CONT.)

- It's a good idea to check linearity again *after* computing the regression when we can examine the residuals.
- You should also check for outliers, which could change the regression.
- If the data seem to clump or cluster in the scatterplot, that could be a sign of trouble worth looking into further.

ASSUMPTIONS AND CONDITIONS (CONT.)

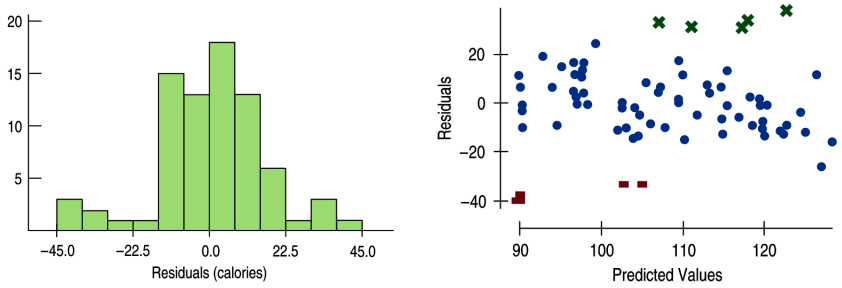
- If the scatterplot is not straight enough, stop here.
  - You can't use a linear model for *any* two variables, even if they are related.
  - They must have a *linear* association or the model won't mean a thing.
- Some nonlinear relationships can be saved by re-expressing the data to make the scatterplot more linear.

ASSUMPTIONS AND CONDITIONS (CONT.)

- Outlier Condition:
  - Watch out for outliers.
  - Outlying points can dramatically change a regression model.
  - Outliers can even change the sign of the slope, misleading us about the underlying relationship between the variables.

SIFTING RESIDUALS FOR GROUPS (CONT.)

- It is a good idea to look at both a histogram of the residuals and a scatterplot of the residuals vs. predicted values:



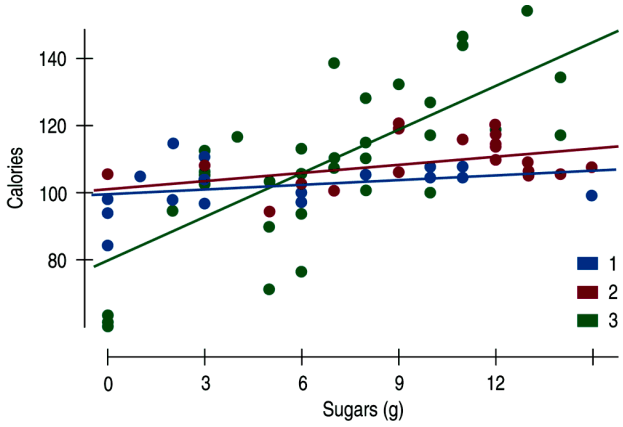
- The small modes in the histogram are marked with different colors and symbols in the residual plot above. What do you see?

SIFTING RESIDUALS FOR GROUPS (CONT.)

- An examination of residuals often leads us to discover groups of observations that are different from the rest.
- When we discover that there is more than one group in a regression, we may decide to analyze the groups separately, using a different model for each group.

SUBSETS

- Below is a graphic of regression lines fit to calories and sugar for each of the three cereal shelves in a supermarket:

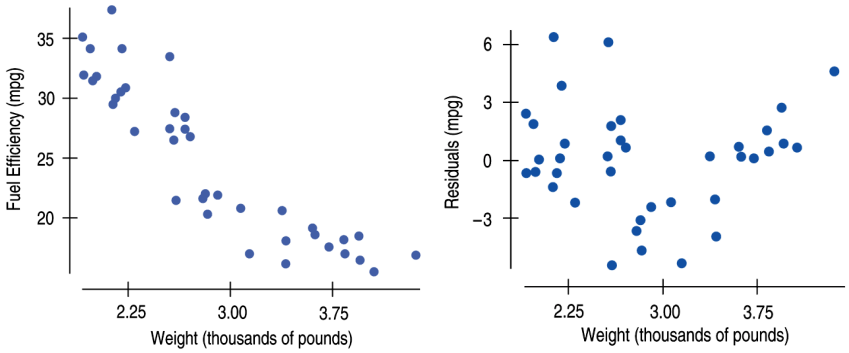


GETTING THE "BENDS"

- Linear regression only works for linear models. (That sounds obvious, but when you fit a regression, you can't take it for granted.)
- A curved relationship between two variables might not be apparent when looking at a scatterplot alone, but will be more obvious in a plot of the residuals.
  - Remember, we want to see "nothing" in a plot of the residuals.

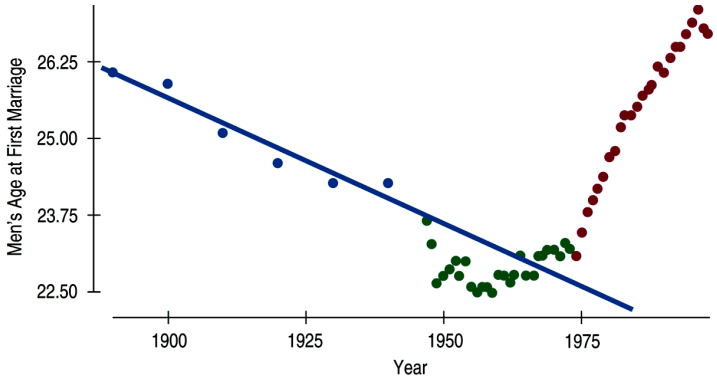
GETTING THE "BENDS" (CONT.)

- The curved relationship between fuel efficiency and weight is more obvious in the plot of the residuals than in the original scatterplot:



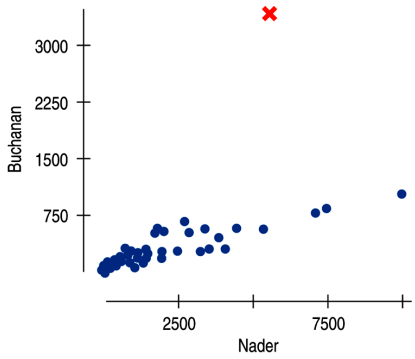
EXTRAPOLATION

- A regression of mean age at first marriage for men vs. year fit to the first 4 decades of the 20th century does not hold for later years:



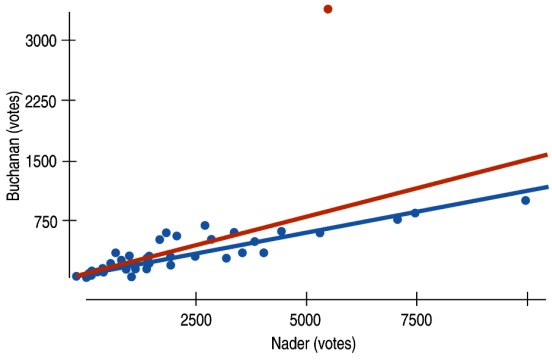
OUTLIERS, LEVERAGE, AND INFLUENCE

- The following scatterplot shows that something was awry in Palm Beach County, Florida, during the 2000 presidential election…



OUTLIERS, LEVERAGE, AND INFLUENCE (CONT.)

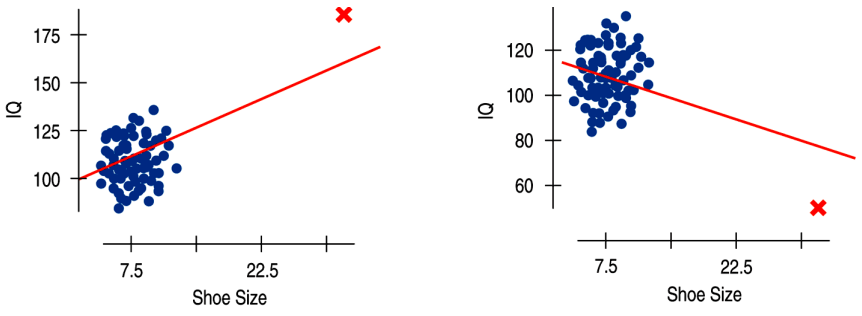
- The red line shows the effects that one unusual point can have on a regression:



OUTLIERS, LEVERAGE, AND INFLUENCE (CONT.)

- A point with high leverage has the potential to change the regression line.
- We say that a point is influential if omitting it from the analysis gives a very different model.

OUTLIERS, LEVERAGE, AND INFLUENCE (CONT.)

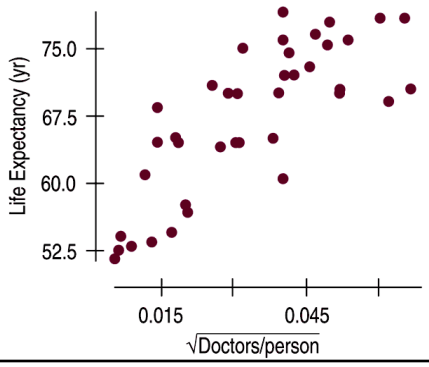


LURKING VARIABLES AND CAUSATION

- No matter how strong the association, no matter how large the R^2 value, no matter how straight the line, there is no way to conclude from a regression alone that one variable *causes* the other.
  - There's always the possibility that some third variable is driving both of the variables you have observed.
- With observational data, as opposed to data from a designed experiment, there is no way to be sure that a lurking variable is not the cause of any apparent association.

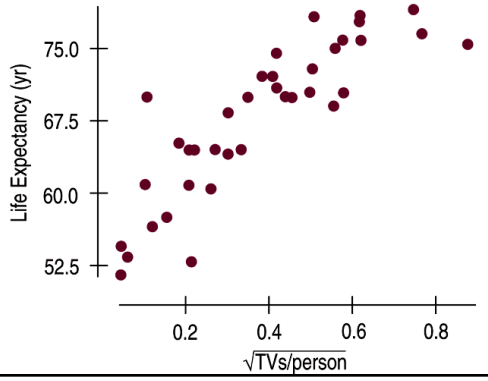
LURKING VARIABLES AND CAUSATION (CONT.)

- The following scatterplot shows that the average *life expectancy* for a country is related to the number of *doctors* per person in that country:



LURKING VARIABLES AND CAUSATION (CONT.)

- This new scatterplot shows that the average *life expectancy* for a country is related to the number of *televisions* per person in that country:



LURKING VARIABLES AND CAUSATION (CONT.)

- Since televisions are cheaper than doctors, send TVs to countries with low life expectancies in order to extend lifetimes. Right?

- How about considering a lurking variable? That makes more sense…
  - Countries with higher standards of living have both longer life expectancies *and* more doctors (and TVs!).
  - If higher living standards *cause* changes in these other variables, improving living standards might be expected to prolong lives and increase the numbers of doctors and TVs.