

Data Science using R



Veerasak Kritsanapraphan - IMC

About myself?

- Graduated from San Francisco State University in Master of Science in Computer Information System, 1997
- PhD. Candidate at Chulalongkorn University, research focus on Data Science, Big Data, Mobile Computing and Internet of Thing (IOT)
- Head of Technology Innovation at True Digital Group
- Instructor for IMC and Software Park in Data Science

Introduce yourself?

- What is your name/position/role in your organization?
- Tell you a bit about yourself?
- What is your passion?
- What is your expectation for this class?



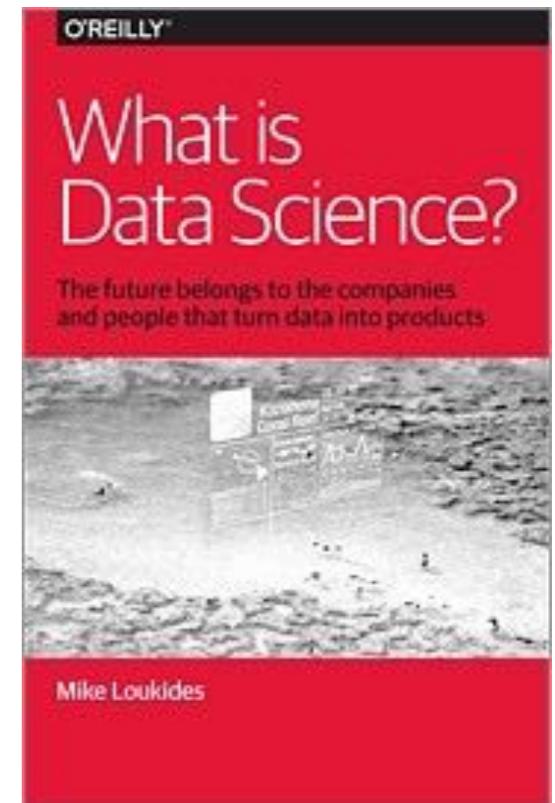
Slide and Sample Data

<https://github.com/vkrit/datascienceusingr>



What is Data Science?

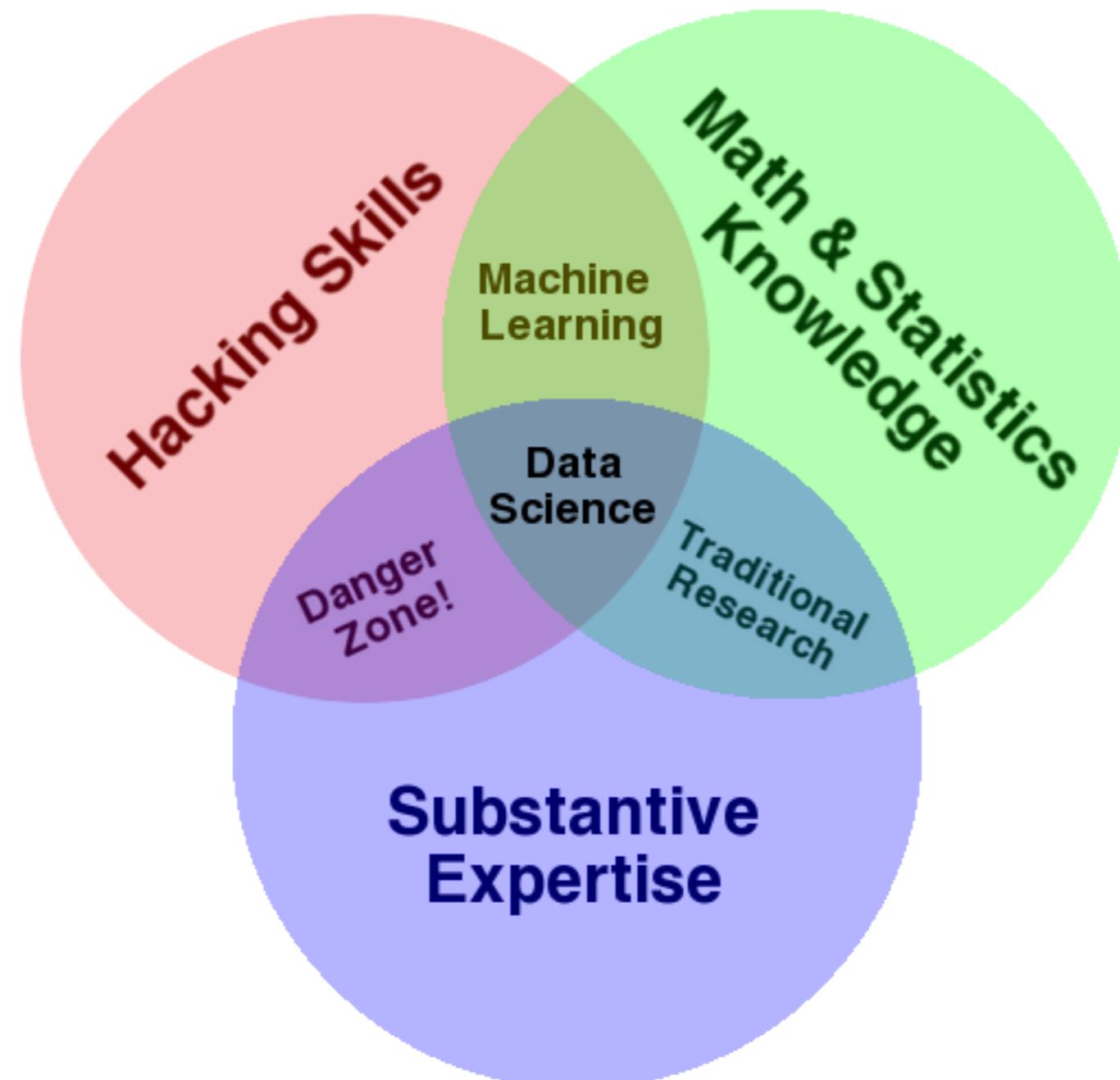
- Data Science aims to derive **knowledge** from **big data, efficiently and intelligently**
- Data Science encompasses the **set of activities, tools, and methods** that enable **data-driven activities** in science, business, medicine, and government



<http://www.oreilly.com/data/free/what-is-data-science.csp>

Goals of Data Science: Turn **data** into **data products**.

Data Science - Drew Convey's Definition



What is R?

- R is a system for statistical computation and graphics.
- It is heavily influenced by the **S** language
- **R** was initially written by **Ross Ihaka** and **Robert Gentleman** at the Department of Statistics of the University of Auckland in Auckland, New Zealand.
- The “**R Core Team**” maintain the source code for the software and release regular updates

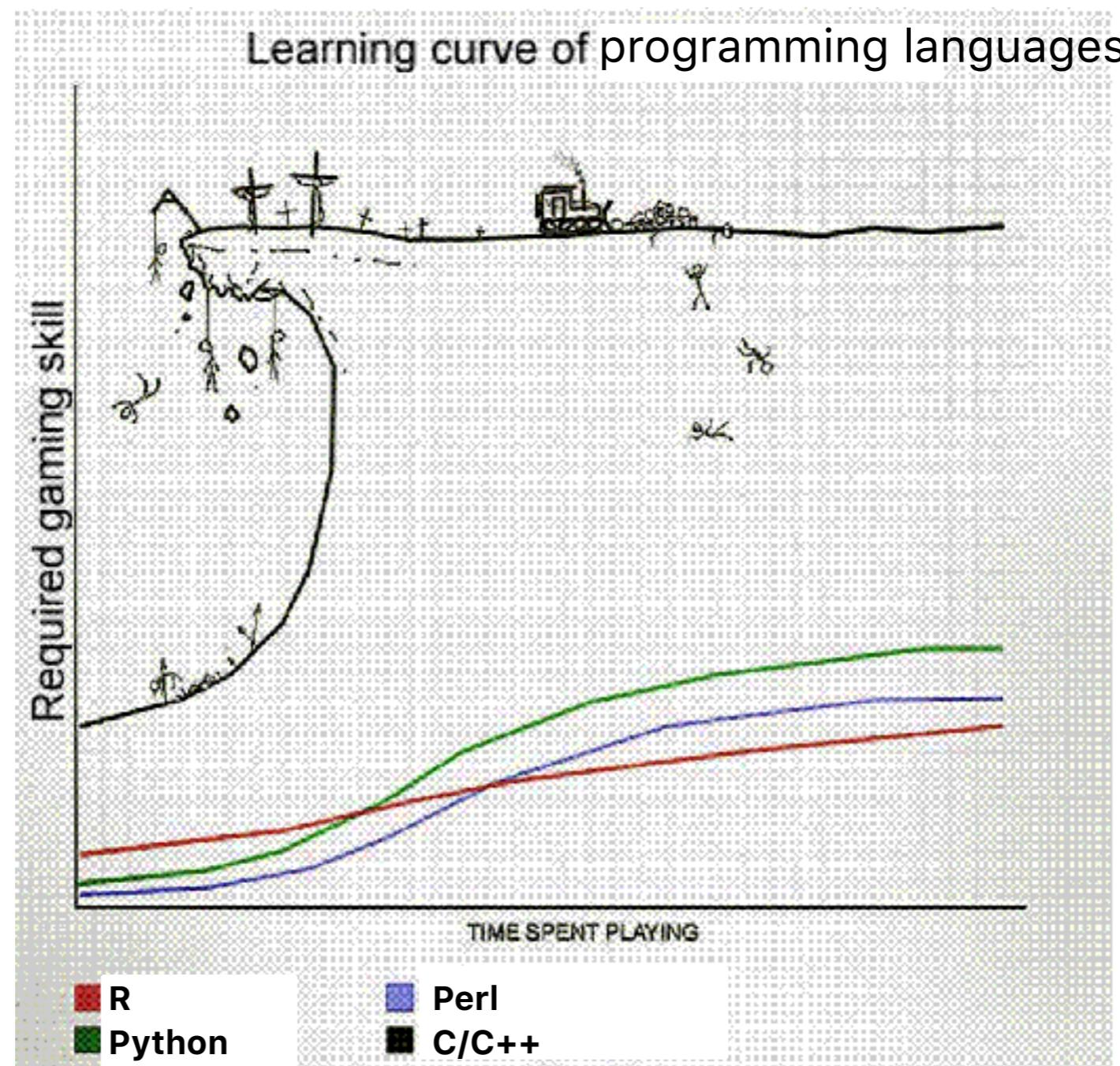
What is R?

- In addition, the R project is added to by many of its users, who write source code for many different types of analytical procedures
- Everything from analytical chemistry to epidemiology to linguistics
- Currently 10,000+ different user--written libraries available (<http://cran.r-project.org>)

Why use R?

- R is Open-Source Software
- R has been used for Statistical Computing for more than 20 years
- Many built-in functions and installable packages that will cover nearly every possible need
- R is an interpreted language
- Code doesn't have to be compiled
- Interactive console makes testing and debugging easy
- Cons to using R
 - Slower than compiled languages
 - Can have runtime errors

Why use R?



Data Science Programming Language

Platform	2019 % share	2018 % share	% change
Python	65.8%	65.6%	0.2%
R Language	46.6%	48.5%	-4.0%
SQL Language	32.8%	39.6%	-17.2%
Java	12.4%	15.1%	-17.7%
Unix shell/awk	7.9%	9.2%	-13.4%
C/C++	7.1%	6.8%	3.7%
Javascript	6.8%	na	na
Other programming and data languages	5.7%	6.9%	-17.1%
Scala	3.5%	5.9%	-41.0%
Julia	1.7%	0.7%	150.4%
Perl	1.3%	1.0%	25.2%
Lisp	0.4%	0.3%	46.1%

Tools



R
www.r-project.org

The engine*



RStudio
www.rstudio.org

The pretty face**

* Many alternatives exist. Smallest learning curve.

** A few alternatives exist. This happens to be the easiest at the moment.



Installing R and R-Studio

Download R

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



[Home]

[Download](#)

[CRAN](#)

[R Project](#)

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Reporting Bugs](#)

[Conferences](#)

[Search](#)

[Get Involved: Mailing Lists](#)

[Developer Pages](#)

[R Blog](#)

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News

- [R version 3.6.1 \(Action of the Toes\)](#) has been released on 2019-07-05.
- useR! 2020 will take place in St. Louis, Missouri, USA.
- [R version 3.5.3 \(Great Truth\)](#) has been released on 2019-03-11.
- The R Foundation Conference Committee has released a [call for proposals](#) to host useR! 2020 in North America.

Download R Studio

<https://rstudio.com/products/rstudio/download/#download>

RStudio Desktop	RStudio Desktop	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
Free	\$995/year	Free	\$4,975/year (5 Named Users)
DOWNLOAD <small>Learn more</small>	BUY <small>Learn more</small>	DOWNLOAD <small>Learn more</small>	BUY <small>Evaluation Learn more</small>
Integrated Tools for R	✓	✓	✓
Priority Support		✓	✓
Access via Web Browser		✓	✓
Enterprise Security			✓
Project Sharing			✓
Manage Multiple R Sessions & Versions			✓
Admin Dashboard			✓
Load Balancing			✓
Auditing and Monitoring			✓
Data Connectivity			✓
Launcher			✓
Tutorial API			✓
License	AGPL	Commercial	AGPL
			Commercial

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.5019 - Ubuntu 18/Debian 10 (64-bit)	106.04 MB	2019-11-01	a6c9af3d8b1621eb155d23c879c1a75a
RStudio 1.2.5019 - Debian 9 (64-bit)	106.39 MB	2019-11-01	bc7b0b25b41e39fb6f1aefaf74163a133
RStudio 1.2.5019 - Fedora 28/Red Hat 8 (64-bit)	120.89 MB	2019-11-01	2291b1befb02622b3aa02c43638ee5c2
RStudio 1.2.5019 - macOS 10.12+ (64-bit)	126.88 MB	2019-11-01	55738355277e8ec660e628acaf2a401b
RStudio 1.2.5019 - SLES/OpenSUSE 12 (64-bit)	99.04 MB	2019-11-01	3bcbf47f40944cc4a5cf4f6fb42319c1
RStudio 1.2.5019 - OpenSUSE 15 (64-bit)	107.09 MB	2019-11-01	29d07b198b7aac92356f8487911efbfa
RStudio 1.2.5019 - Fedora 19/Red Hat 7 (64-bit)	120.26 MB	2019-11-01	dab1cb5f0ed39f5bcf0c795e2938fa94
RStudio 1.2.5019 - Ubuntu 14/Debian 8 (64-bit)	96.93 MB	2019-11-01	f86811fce50b48850fed259d6ce7ef13
RStudio 1.2.5019 - Windows 10/8/7 (64-bit)	149.82 MB	2019-11-01	4d6521a9b89d70c3bf50414c8b6708f2
RStudio 1.2.5019 - Ubuntu 16 (64-bit)	104.91 MB	2019-11-01	67d5a2c255f2bc1a171c7e417853102c

Workshop 1 - Software Installation

- 1) Install R
- 2) Install R-Studio

RStudio Features

- Code completion
- Command history search
- Command history to R script / file
- Function extraction from Rscript
- Sweave and Knitr support

Introducing R-Studio

The screenshot displays the R-Studio interface with the following components:

- Script Editor:** Shows the R script `diamondPricing.R` containing code to load ggplot2, source a plotting function, view and summary the diamonds dataset, calculate average size, and create a qplot for diamond pricing.
- Workspace:** Lists the `diamonds` dataset (53940 obs. of 10 variables), the calculated `aveSize` (0.7979), the `clarity` factor levels, and the `format.plot` function.
- Console:** Displays the results of running the R script, including summary statistics for the diamonds dataset and the generated plot command.
- Plots:** A scatter plot titled "Diamond Pricing" showing Price vs. Carat. The plot uses color to represent diamond clarity levels, with a legend on the right mapping colors to clarity grades: II, SI2, SI1, VVS2, VSI, VVS1, and I.

RStudio Panes

RStudio has 4 main windows ('panes'):

- **The Source pane:** create a file that you can save and run later
- **The Console pane:** type or paste in commands to get output from R
- **The Workspace/History pane:** see a list of variables or previous commands
- **The Files/Plots/Packages/Help pane:** see plots, help pages, and other items in this window.

Source and Console Panes

Source window: Create a file here, so that you can save and run it later (or turn in as homework)

Any non-command line
should start with a #

Console: Type or paste commands in here to get results from R

If you are loading a data file, you will need to be in the correct directory

> denotes that R is waiting for a command

+ denotes that R is waiting for you to finish the previous command (not shown here)

The screenshot shows the RStudio interface with two main panes: the Source pane and the Console pane.

Source pane: The title bar says "hw1.R x". The code content is as follows:

```
1 #  
2 # David Choi  
3 # HW 1  
4 #  
5  
6 # Here are the steps to accomplish problem 1:  
7 x = sqrt(5)  
8 y = sqrt(2)  
9 z = x + y  
10 z  
11
```

Console pane: The title bar says "Console ~/Dropbox/teaching/DataVis mini 4/homework 1". The history shows:

```
[Workspace loaded from ~/.RData]  
> setwd("~/Dropbox/teaching/DataVis mini 4/homework 1")  
> x = sqrt(5)  
> y = sqrt(2)  
> z = x + y  
> z  
[1] 3.650282  
>
```

Console Panes

Variables

- Save the results of a command in memory by **giving it a name**:

`z` uses `x` and `y`: →

Values are not linked;
updating `x` doesn't →
change `z`

Redefine `z` using the new →
value for `x`

Use “ “ or ‘ ’ to distinguish →
between variable names
and regular text

Note:

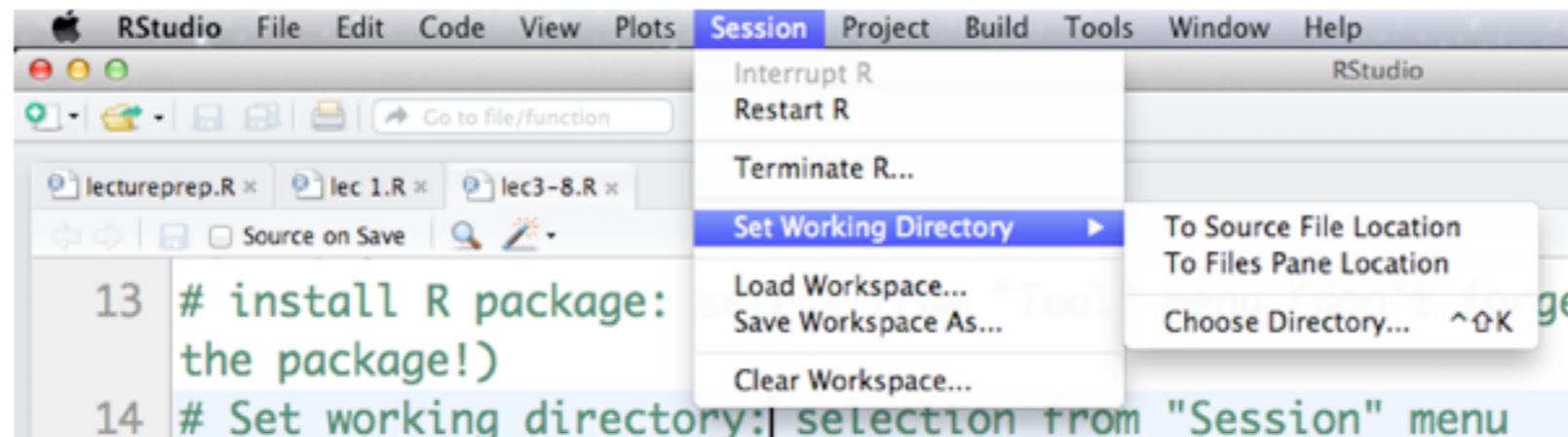
`x <- 3`: same as `x=3`

```
Console ~ / ↗
> x = 3
> y = 2
> x           ← Create x and y
[1] 3
> y
[1] 2
> z = sqrt(x^2+y^2)
> z
[1] 3.605551
> x = 5
> z
[1] 3.605551
> z = sqrt(x^2+y^2)
> z
[1] 5.385165
> str = 'Hi there'
> str
[1] "Hi there"
> str2 = "It's cold outside"
> str2
[1] "It's cold outside"
```

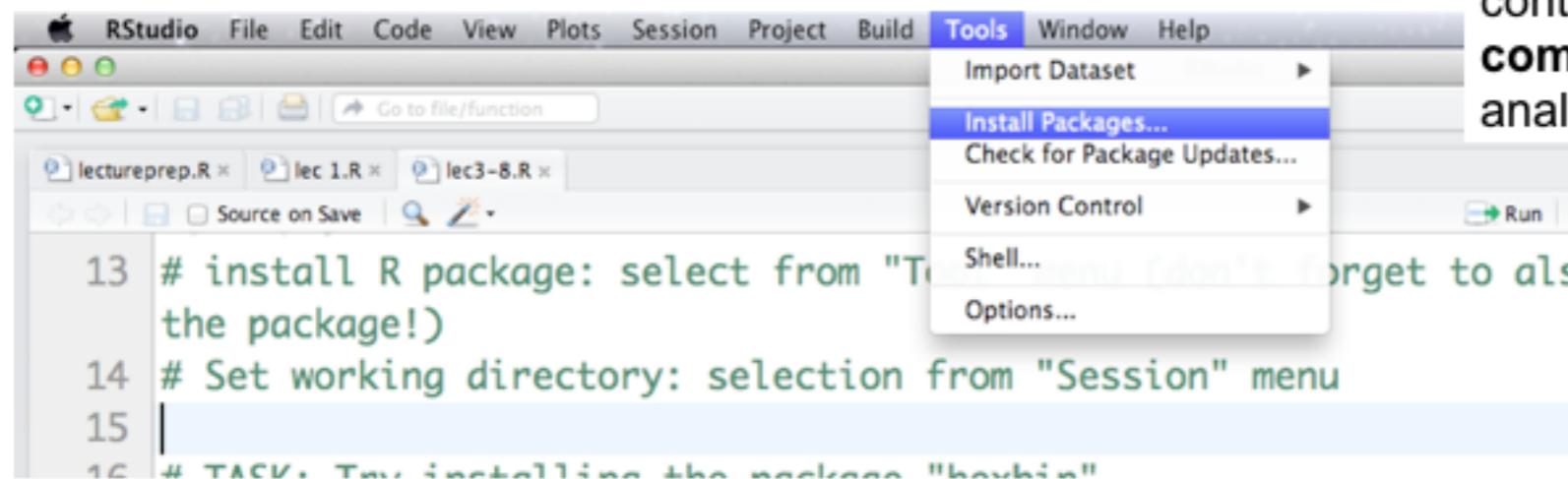
RStudio Menu

Two helpful menu items in Rstudio

- Set the current directory:



- Install a package



Packages are extensions to R, containing **new commands** for analysis or graphics

R Markdown

The image shows the RStudio interface with two main panes. The left pane displays the R Markdown source code in a file named 'example.Rmd'. The right pane shows the rendered HTML output. Both panes have a blue background with a network graph watermark.

Left Pane (Code View):

```
1 Header 1
2 -----
3 This is an R Markdown document. Markdown is a
4 simple formatting syntax for authoring web pages.
5 Use an asterisk mark, to provide emphasis such as
6 *italics* and **bold**.
7 Create lists with a dash:
8 - Item 1
9 - Item 2
10 - Item 3
11
12 You can write `in-line` code with a back-tick.
13 ...
14
15 Code blocks display
16 with fixed-width font
17 ...
18
19 > Blockquotes are offset
20
```

Right Pane (Preview HTML):

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark, to provide emphasis such as *italics* and **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

You can write `in-line` code with a back-tick.

Code blocks display
with fixed-width font

Blockquotes are offset

R Markdown

chunks.Rmd x

ABC MD Knit HTML Chunks

```
1 R Code Chunks
2 -----
3
4 With R Markdown, you can insert R code
5 chunks including plots:
6
7 ````{r qplot, fig.width=4, fig.height=3,
8 | message=FALSE}
9 # quick summary and plot
10 library(ggplot2)
11 summary(cars)
12 qplot(speed, dist, data=cars) +
13   geom_smooth()
```

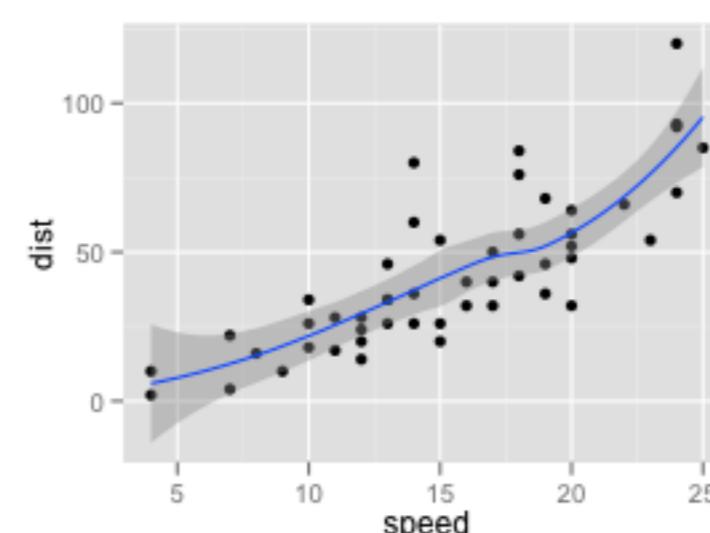
R Code Chunks

With R Markdown, you can insert R code chunks including plots:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

```
##       speed          dist
##  Min.   : 4.0   Min.   :  2
##  1st Qu.:12.0   1st Qu.: 26
##  Median :15.0   Median : 36
##  Mean   :15.4   Mean   : 43
##  3rd Qu.:19.0   3rd Qu.: 56
##  Max.   :25.0   Max.   :120
```

```
qplot(speed, dist, data = cars) + geom_smooth()
```



Workshop 2: Hello World

Open RStudio on your machine

- File > New File > R Markdown ...
- Change **summary(cars)** in the first code block to **print("Hello world!")**
- Click Knit HTML to produce an HTML file.
- Save your Rmd file as helloworld.Rmd

Data Types

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)

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

Operators

```
7 + 5 # Addition
```

```
[1] 12
```

```
7 - 5 # Subtraction
```

```
[1] 2
```

```
7 * 5 # Multiplication
```

```
[1] 35
```

```
7 ^ 5 # Exponentiation
```

```
[1] 16807
```

```
7 / 5 # Division
```

```
[1] 1.4
```

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

```
[1] 2
```

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

```
[1] 1
```

Operators

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

```
7 > 5
```

```
[1] TRUE
```

```
7 < 5
```

```
[1] FALSE
```

```
7 >= 7
```

```
[1] TRUE
```

```
7 <= 5
```

```
[1] FALSE
```

```
7 == 5
```

```
[1] FALSE
```

```
7 != 5
```

```
[1] TRUE
```

Boolean Operators

Basically “and” and “or”:

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

```
[1] FALSE
```

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

```
[1] TRUE
```

Basic Math / Basic Logic

- + addition
- subtraction
- * multiplication
- / division
- % modulus (remainder)
- ^ to the power

?Arithmetic

- ! NOT
- & bitwise AND
- | bitwise OR
- && short circuit AND
- || short circuit OR
- == equality
- != NOT equality

?Logic

Also try ?Syntax, ?Comparison

More 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

```
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
```

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

```
[1] FALSE
```

Variables

We can give names to data objects; these give us **variables**

A few variables are built in:

```
pi
```

```
[1] 3.141593
```

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

```
pi*10
```

```
[1] 31.41593
```

```
cos(pi)
```

```
[1] -1
```

Assignment Operator

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

```
approx.pi <- 22 / 7  
approx.pi
```

```
[1] 3.142857
```

```
diameter.in.cubits = 10  
approx.pi*diameter.in.cubits
```

```
[1] 31.42857
```

Variables

- 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
 - Good: num.students <- 35
 - Bad: ns <- 35

Workspace

What names have you defined values for?

```
ls()
```

```
[1] "approx.pi"           "circumference.in.cubits"  
[3] "diameter.in.cubits"
```

Getting rid of variables:

```
rm("circumference.in.cubits")  
ls()
```

```
[1] "approx.pi"           "diameter.in.cubits"
```

Data Structure : Vector

- 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

```
students <- c("Sean", "Louisa", "Frank", "Farhad", "Li")
midterm <- c(80, 90, 93, 82, 95)
```

- Typing the variable name at the prompt causes it to display

```
students
```

```
[1] "Sean"    "Louisa"  "Frank"   "Farhad" "Li"
```

Indexing Vector

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

```
students
```

```
[1] "Sean"    "Louisa"  "Frank"   "Farhad"  "Li"
```

```
students[4]
```

```
[1] "Farhad"
```

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

```
students[-4]
```

```
[1] "Sean"    "Louisa"  "Frank"   "Li"
```

Vector Arithmetic

Operators apply to vectors “pairwise” or “elementwise”:

```
final <- c(78, 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] 158 174 188 164 186
```

```
(midterm + final)/2 # Average exam score
```

```
[1] 79 87 94 82 93
```

```
course.grades <- 0.4*midterm + 0.6*final # Final course grade  
course.grades
```

```
[1] 78.8 86.4 94.2 82.0 92.6
```

Pairwise Comparison

Is the final score higher than the midterm score?

```
midterm
```

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

```
final
```

```
[1] 78 84 95 82 91
```

```
final > midterm
```

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

Function on Vectors

```
course.grades
```

```
[1] 78.8 86.4 94.2 82.0 92.6
```

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

```
[1] 86.8
```

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

```
[1] 86.4
```

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

```
[1] 6.625708
```

More on Functions

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

```
[1] 78.8 82.0 86.4 92.6 94.2
```

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

```
[1] 94.2
```

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

```
[1] 78.8
```

Referencing elements of Vectors

```
students
```

```
[1] "Sean"    "Louisa"  "Frank"   "Farhad"  "Li"
```

Vector of indices:

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

```
[1] "Louisa" "Farhad"
```

Vector of negative indices

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

```
[1] "Louisa" "Farhad" "Li"
```

More on Referencing

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

```
course.grades
```

```
[1] 78.8 86.4 94.2 82.0 92.6
```

```
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] "Frank" "Li"
```

Workshop 3

Using the same Rmarkdown file from Workshop 1, do the following:-

- 1) Creating sequences

Colon operator:

```
1:10 # Numbers 1 to 10
```

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

```
127:132 # Numbers 127 to 132
```

```
## [1] 127 128 129 130 131 132
```

Workshop 3

seq function: `seq(from, to, by)`

```
seq(1,10) # Numbers 1 to 10
```

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

```
seq(1,10,2) # Odd numbers from 1 to 10
```

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

```
seq(2,10,2) # Even numbers from 2 to 10
```

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

Workshop 3 : Use RStudio to answer questions below

- (a) Use : to output the sequence of numbers from 3 to 12
- (b) Use seq() to output the sequence of numbers from 3 to 30 in increments of 3
- (c) Save the sequence from (a) as a variable x, and the sequence from (b) as a variable y. Output their product $x*y$

Difference between = and <-

- The operators <- and = assign into the environment in which they are evaluated.
- The operator <- can be used anywhere, whereas the operator = is only allowed at the top level (e.g., in the complete expression typed at the command prompt) or as one of the subexpressions in a braced list of expressions.

```
matrix(1,nrow=2)
```

```
matrix(1,nrow<-2)
```

Reading and Getting Data into R

Combine Command

```
c(1, 2, 3, 4)  
c(item1, item2, item3, item4)  
c("item1", "item2", "item3")
```

Scan Command

```
our.data = scan()
```

```
scan(what = 'character')  
data5 = scan(sep = ';', what = 'char')  
data6 = scan(file = 'data.txt')
```

Working Directory

```
getwd()
```

Reading Bigger Data Files

```
read.csv()  
read.csv(file, sep = ';', header = TRUE, row.names)  
fw = read.csv(file.choose())  
  
my.csv = read.table(file.choose(), header = TRUE)  
my.tsv = read.delim(file.choose())  
my.tsv = read.csv(file.choose(), sep = '\t')  
my.tsv = read.table(file.choose(), header = TRUE, sep = '\t')
```

Importing Data

- To import tabular data into R, we use the `read.table()` command

```
1 survey <- read.table("survey_data.csv", header=TRUE, sep=",")  
2  
3
```

- Let's parse this command one component at a time
 - The data is in a file called `survey_data.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=","`
- Could've also used `read.csv()`, which is just `read.table()` with the preset `sep=","`

Exploring the Data

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

```
class(survey)
```

```
[1] "data.frame"
```

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

```
head(survey, 3)
```

	Program	PriorExp	Rexperience	OperatingSystem	TVhours
1	MISM	Some experience	Never used	Windows	1
2	Other	Some experience	Basic competence	Windows	8
3	MISM	Extensive experience Editor	Basic competence	Windows	4
1	Microsoft Word				
2	Microsoft Word				
3	Microsoft Word				

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

Simple Summary

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

```
str(survey)
```

```
'data.frame': 31 obs. of 6 variables:  
 $ Program      : Factor w/ 3 levels "MISM","Other",...: 1 2 1 3 2 3 2 1 3 3 ...  
 $ PriorExp     : Factor w/ 3 levels "Extensive experience",...: 3 3 1 2 2 3 3 3  
 2 1 ...  
 $ Rexperience  : Factor w/ 3 levels "Basic competence",...: 3 1 1 2 3 3 3 3 3 3  
 ...  
 $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 2 2 2 1 1 2 1 2  
 ...  
 $ TVhours       : int 1 8 4 5 3 10 0 10 15 4 ...  
 $ Editor        : Factor w/ 1 level "Microsoft Word": 1 1 1 1 1 1 1 1 1 1 ...
```

- This says that `TVhours` is a numeric variable, while all the rest are factors (categorical)

Another Summary

```
summary(survey)
```

Program	PriorExp	Rexperience
MISM :10	Extensive experience : 5	Basic competence : 2
Other: 6	Never programmed before:12	Installed on machine: 7
PPM :15	Some experience :14	Never used :22

OperatingSystem	TVhours	Editor
Mac OS X:14	Min. : 0.000	Microsoft Word:31
Windows :17	1st Qu.: 1.000	
	Median : 4.000	
	Mean : 5.742	
	3rd Qu.: 9.000	
	Max. :40.000	

Data Frames

- 2 Dimensional Objects, it has rows and columns. R treats the columns as separate samples or variables, rows represent the replicates or observations.
- To see what an R object is made up of, you can use `attributes()`

```
attributes(survey)
```

```
$names
[1] "Program"          "PriorExp"        "Rexperience"    "OperatingSystem"
[5] "TVhours"          "Editor"          "Experience"      "OperatingSystem"

$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 19 20 21 22 23
[24] 24 25 26 27 28 29 30 31
```

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 survey responses and the number of survey questions

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

```
[1] 31
```

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

```
[1] 6
```

- When writing reports, you will often want to say how large your sample size was
- To do this *inline*, use the syntax:

```
`r nrow(survey)`
```

- This allows us to write “31 students responded to the survey”, and have the number displayed automatically change when `nrow(survey)` changes.

Indexing Data Frame

- There are many different ways of indexing the same piece of a data frame

```
survey[["Program"]] # "Program" element
```

```
[1] MISM Other MISM PPM  Other PPM  Other MISM PPM  PPM  MISM  
[12] PPM   PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM  PPM  
[23] MISM Other PPM  PPM  MISM  PPM  Other Other MISM  
Levels: MISM Other PPM
```

```
survey$Program # "Program" element
```

```
[1] MISM Other MISM PPM  Other PPM  Other MISM PPM  PPM  MISM  
[12] PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM  PPM  
[23] MISM Other PPM  PPM  MISM  PPM  Other Other MISM  
Levels: MISM Other PPM
```

```
survey[,1] # Data from 1st column
```

```
[1] MISM Other MISM PPM  Other PPM  Other MISM PPM  PPM  MISM  
[12] PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM  PPM  
[23] MISM Other PPM  PPM  MISM  PPM  Other Other MISM  
Levels: MISM Other PPM
```

More Indexing

- Note that single brackets and double brackets have different effects

```
survey[["Program"]]
```

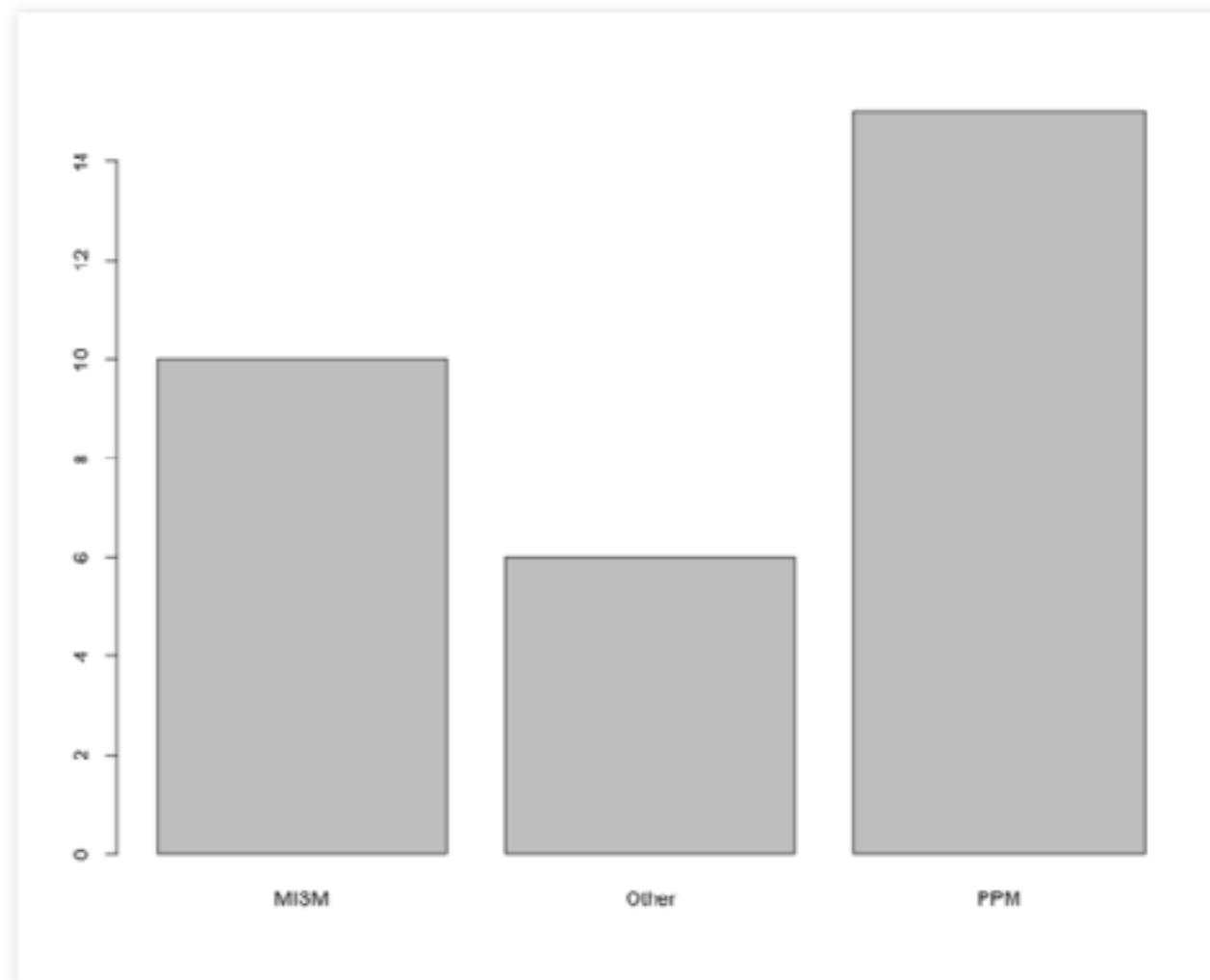
```
[1] MISM Other MISM PPM Other PPM Other MISM PPM PPM MISM  
[12] PPM PPM PPM PPM PPM PPM MISM MISM MISM PPM PPM  
[23] MISM Other PPM PPM MISM PPM Other Other MISM  
Levels: MISM Other PPM
```

```
survey["Program"] # sub-data frame containing only "Program"
```

```
Program  
1      MISM  
2      Other  
3      MISM  
4      PPM  
5      Other  
6      PPM  
7      Other  
8      MISM  
9      PPM  
10     PPM  
11     MISM  
12     PPM  
13     PPM  
14     PPM
```

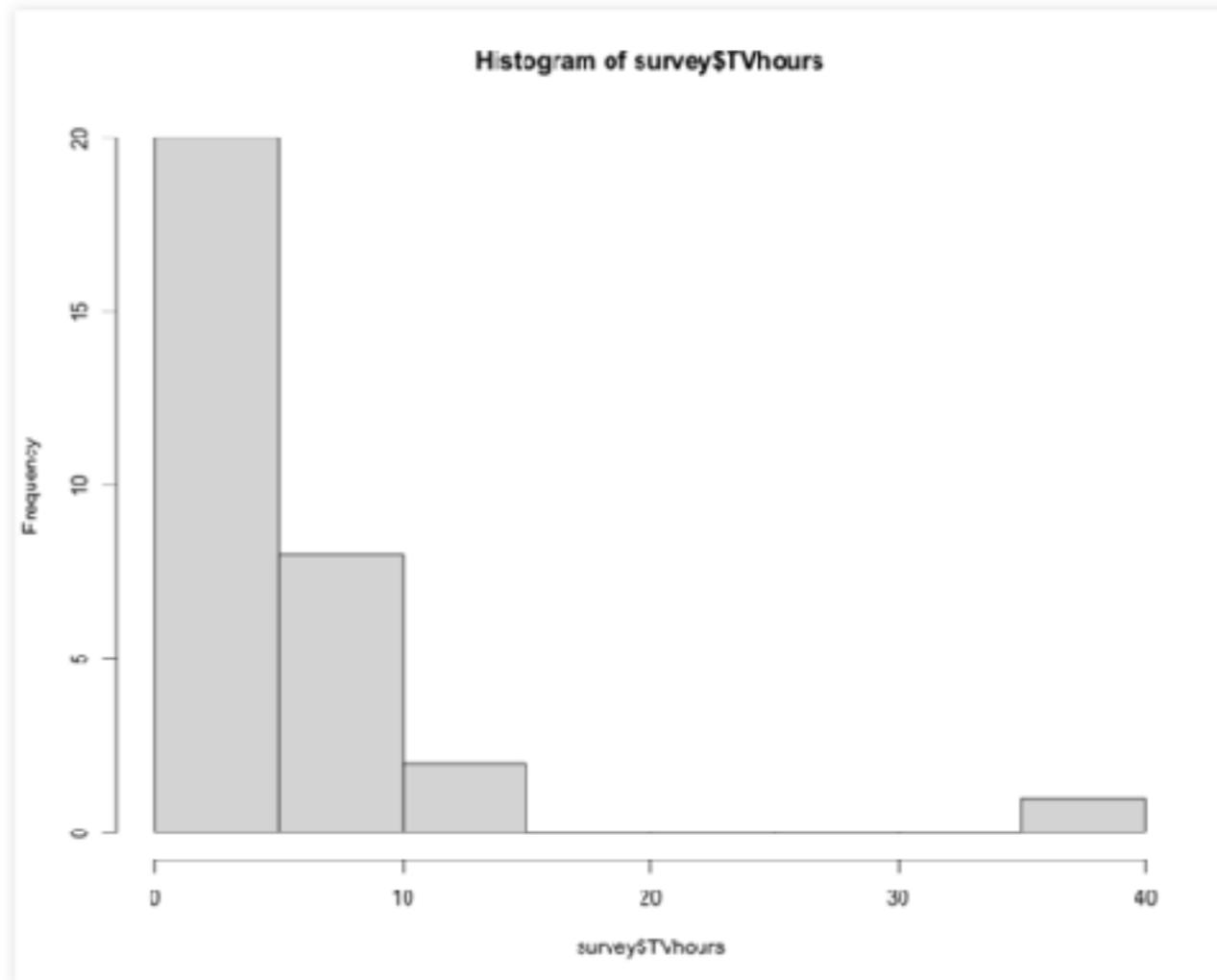
Bar Plot (Categorical Data)

```
plot(survey[["Program"]])
```



Histogram (Continuous Data)

```
hist(survey$TVhours, col="lightgray")
```



Indexing Multiple Columns

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

```
Program TVhours
1 MISM 1
2 Other 8
3 MISM 4
4 PPM 5
5 Other 3
6 PPM 10
```

```
head(survey[c("Program", "Editor")]) # Data from "Program" and "Editor"
```

```
Program Editor
1 MISM Microsoft Word
2 Other Microsoft Word
3 MISM Microsoft Word
4 PPM Microsoft Word
5 Other Microsoft Word
6 PPM Microsoft Word
```

Indexing Row and Column

- Data frames have two dimensions to index across

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

```
Program      PriorExp Rexperience OperatingSystem TVhours
6      PPM Some experience    Never used        Mac OS X      10
          Editor
6 Microsoft Word
```

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

```
[1] 10
```

```
survey[6, "Program"] # Program of 6th survey respondent
```

```
[1] PPM
Levels: MISM Other PPM
```

```
survey[["Program"]][6] # Program of 6th survey respondent
```

```
[1] PPM
Levels: MISM Other PPM
```

More Indexing

- We can use this operator for indexing

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

	Program	PriorExp	Rexperience	OperatingSystem	TVhours
1	MISM	Some experience	Never used	Windows	1
2	Other	Some experience	Basic competence	Windows	8
3	MISM	Extensive experience	Basic competence Editor	Windows	4
1	Microsoft Word				
2	Microsoft Word				
3	Microsoft Word				

```
survey[3:5, c(1,5)]
```

	Program	TVhours
3	MISM	4
4	PPM	5
5	Other	3

Subsets of Data

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

```
survey[survey$Program=="MISM",] # Data from the MISM students
```

	Program	PriorExp	Rexperience	OperatingSystem
1	MISM	Some experience	Never used	Windows
3	MISM	Extensive experience	Basic competence	Windows
8	MISM	Some experience	Never used	Windows
11	MISM	Extensive experience	Installed on machine	Mac OS X
18	MISM	Extensive experience	Never used	Mac OS X
19	MISM	Extensive experience	Never used	Windows
20	MISM	Some experience	Never used	Windows
23	MISM	Some experience	Never used	Windows
27	MISM	Some experience	Never used	Mac OS X
31	MISM	Some experience	Installed on machine	Windows
	TVhours	Editor		
1		1 Microsoft Word		
3		4 Microsoft Word		
8		10 Microsoft Word		
11		0 Microsoft Word		
18		3 Microsoft Word		
19		0 Microsoft Word		
20		1 Microsoft Word		
23		3 Microsoft Word		
27		0 Microsoft Word		
31		0 Microsoft Word		

More subsets example

- Let's pull all of the PPM students who have never used R before

```
survey[survey$Program=="PPM" & survey$Rexperience=="Never used", ]
```

	Program	PriorExp	Rexperience	OperatingSystem	TVhours
6	PPM	Some experience	Never used	Mac OS X	10
9	PPM	Never programmed before	Never used	Mac OS X	15
10	PPM	Extensive experience	Never used	Windows	4
12	PPM	Never programmed before	Never used	Windows	0
13	PPM	Some experience	Never used	Mac OS X	10
14	PPM	Never programmed before	Never used	Mac OS X	4
15	PPM	Some experience	Never used	Windows	10
16	PPM	Some experience	Never used	Mac OS X	2
22	PPM	Never programmed before	Never used	Windows	7
25	PPM	Never programmed before	Never used	Mac OS X	6
		Editor			
6	Microsoft Word				
9	Microsoft Word				
10	Microsoft Word				
12	Microsoft Word				
13	Microsoft Word				
14	Microsoft Word				
15	Microsoft Word				
16	Microsoft Word				
22	Microsoft Word				
25	Microsoft Word				

Subset

- 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".

```
subset(survey, select=c("OperatingSystem", "TVhours"), subset=(Program == "PPM"  
| Program == "Other") & Rexperience == "Basic competence")
```

	OperatingSystem	TVhours
2	Windows	8

Calculation from subset

```
mean(survey$TVhours[survey$Program == "PPM"]) # Average time PPM's spent watching TV
```

```
[1] 8.8
```

```
mean(survey$TVhours[survey$Program == "MISM"]) # Average time MISM's spent watching TV
```

```
[1] 2.2
```

```
mean(survey$TVhours[survey$Program == "Other"]) # Average time "Others" spent watching TV
```

```
[1] 4
```

Coding Style

- Coding style (and code commenting) will become increasingly more important as we get into more advanced and involved programming tasks
- A few R “style guides” exist:
 - <http://r-pkgs.had.co.nz/style.html>
 - Google’s R Style Guide (<https://google.github.io/styleguide/Rguide.xml>)

Example Style Guide

Assignment operator. USE <-

```
student.names <- c("Eric", "Hao", "Jennifer") # Good  
student.names = c("Eric", "Hao", "Jennifer") # Bad
```

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

```
sort(tv.hours, decreasing=TRUE) # Good  
sort(tv.hours, decreasing<-TRUE) # Bad!!
```

Style Guide : Variable Name

- 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 `.`

```
# 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

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

Workshop 4: Importing and Indexing data

Data practice

```
survey <- read.csv("survey_data.csv", header=TRUE);
```

Use command to answer these questions?

- (a) How many survey respondents are from MISM or Other?
- (b) What % of survey respondents are from PPM?

Index practice

- (a) Use \$ notation to pull the OperatingSystem column from the survey data
- (b) Do the same thing with [,] notation, referring to OperatingSystem by name

More on Data Frame

```
library(MASS)  
head(Cars93, 3)
```

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city
1	Acura	Integra	Small	12.9	15.9	18.8	25
2	Acura	Legend	Midsize	29.2	33.9	38.7	18
3	Audi	90	Compact	25.9	29.1	32.3	20
	MPG.highway		AirBags		DriveTrain	Cylinders	EngineSize
1	31		None	Front	4	1.8	
2	25	Driver & Passenger		Front	6	3.2	
3	26	Driver only		Front	6	2.8	
	Horsepower	RPM	Rev.per.mile	Man.trans.avail	Fuel.tank.capacity		
1	140	6300	2890	Yes		13.2	
2	200	5500	2335	Yes		18.0	
3	172	5500	2280	Yes		16.9	
	Passengers	Length	Wheelbase	Width	Turn.circle	Rear.seat.room	
1	5	177	102	68	37	26.5	
2	5	195	115	71	38	30.0	
3	5	180	102	67	37	28.0	
	Luggage.room	Weight	Origin		Make		
1	11	2705	non-USA	Acura	Integra		
2	15	3560	non-USA	Acura	Legend		
3	14	3375	non-USA		Audi	90	

Add column using “Transform”

- `transform()` returns a new data frame with columns modified or added as specified by the function call

```
Cars93.metric <- transform(Cars93,
                           KMPL.city = 0.425 * MPG.city,
                           KMPL.highway = 0.425 * MPG.highway)
tail(names(Cars93.metric))
```

```
[1] "Luggage.room"   "Weight"        "Origin"       "Make"
[5] "KMPL.city"      "KMPL.highway"
```

- Our data frame has two new columns, giving the fuel consumption in km/l

Another Approach

```
KMPL.city.2 <- 0.425 * Cars93$MPG.city  
# Add a new column called KMPL.city.2  
Cars93.metric$KMPL.city.2 <- KMPL.city.2  
tail(names(Cars93.metric))
```

```
[1] "Weight"          "Origin"        "Make"          "KMPL.city"  
[5] "KMPL.highway"   "KMPL.city.2"
```

- Let's check that both approaches did the same thing

```
identical(Cars93.metric$KMPL.city, Cars93.metric$KMPL.city.2)
```

```
[1] TRUE
```

Changing level of a factor

```
manufacturer <- Cars93$Manufacturer  
head(manufacturer, 10)
```

```
[1] Acura      Acura      Audi       Audi       BMW        Buick      Buick  
[8] Buick      Buick      Cadillac  
32 Levels: Acura Audi BMW Buick Cadillac Chevrolet Chrysler ... Volvo
```

We'll use the `mapvalues(x, from, to)` function from the `plyr` library.

```
library(plyr)  
  
# Map Chevrolet, Pontiac and Buick to GM  
manufacturer.combined <- mapvalues(manufacturer,  
                                    from = c("Chevrolet", "Pontiac", "Buick"),  
                                    to = rep("GM", 3))  
  
head(manufacturer.combined, 10)
```

```
[1] Acura      Acura      Audi       Audi       BMW        GM        GM  
[8] GM         GM         Cadillac  
30 Levels: Acura Audi BMW GM Cadillac Chrysler Dodge ... Volvo
```

Another example

- A lot of data comes with integer encodings of levels
- You may want to convert the integers to more meaningful values for the purpose of your analysis
- Let's pretend that in the class survey 'Program' was coded as an integer with 1 = MISM, 2 = Other, 3 = PPM
- Here's how we would get back the program codings using the `transform()`, `as.factor()` and `mapvalues()` functions

```
survey <- transform(survey, Program = as.factor(mapvalues(Program, c(1, 2, 3),
c("MISM", "Other", "PPM"))))  
head(survey)
```

	Program	PriorExp	Rexperience	OperatingSystem
1	MISM	Some experience	Never used	Windows
2	Other	Some experience	Basic competence	Windows
3	MISM	Extensive experience	Basic competence	Windows
4	PPM	Never programmed before	Installed on machine	Windows
5	Other	Never programmed before	Never used	Windows
6	PPM	Some experience	Never used	Mac OS X
	TVhours	Editor		
1	1	Microsoft Word		
2	8	Microsoft Word		
3	4	Microsoft Word		
4	5	Microsoft Word		
5	3	Microsoft Word		
6	10	Microsoft Word		

table() function

- The `table()` function builds **contingency tables** showing counts at each combination of factor levels

```
table(Cars93$AirBags)
```

Driver & Passenger	16	Driver only	43	None	34
--------------------	----	-------------	----	------	----

```
table(Cars93$Origin)
```

USA	non-USA
48	45

```
table(Cars93$AirBags, Cars93$Origin)
```

	USA	non-USA
Driver & Passenger	9	7
Driver only	23	20
None	16	18

- Looks like US and non-US cars had about the same distribution of AirBag types

more table()

- When `table()` is supplied a data frame, it produces contingency tables for all combinations of factors

```
head(Cars93[c("AirBags", "Origin")], 3)
```

```
      AirBags   Origin  
1       None non-USA  
2 Driver & Passenger non-USA  
3     Driver only non-USA
```

```
table(Cars93[c("AirBags", "Origin")])
```

AirBags	Origin	
	USA	non-USA
Driver & Passenger	9	7
Driver only	23	20
None	16	18

Basic 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("Michael", 165, TRUE) # (name, weight, is.male)  
my.vector.1
```

```
[1] "Michael" "165"      "TRUE"
```

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

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

List vs Vector

```
my.vector.2 <- c(FALSE, TRUE, 27) # (is.male, is.citizen, 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

```
my.list <- list("Michael", 165, TRUE)  
my.list
```

```
[[1]]  
[1] "Michael"  
  
[[2]]  
[1] 165  
  
[[3]]  
[1] TRUE
```

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

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

Named Elements

```
patient.1 <- list(name="Michael", weight=165, is.male=TRUE)  
patient.1
```

```
$name  
[1] "Michael"
```

```
$weight  
[1] 165
```

```
$is.male  
[1] TRUE
```

Referencing element inside list

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

```
[1] "Michael"
```

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

```
[1] "Michael"
```

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

```
$name  
[1] "Michael"
```

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

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

Function

- 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

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

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

```
addOne(12)
```

```
[1] 13
```

Another example

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

```
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 return 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)  
}
```

Try out function

```
createPatientRecord("Michael Smith", 185, 12 * 6 + 1)
```

```
$first.name  
[1] "Michael"
```

```
$last.name  
[1] "Smith"
```

```
$weight  
[1] 84.09091
```

```
$height  
[1] 1.8542
```

```
$bmi  
[1] 24.45884
```

Another example

- Calculate mean, median and standard deviation

```
threeNumberSummary <- function(x) {  
  c(mean=mean(x), median=median(x), sd=sd(x))  
}  
x <- rnorm(100, mean=5, sd=2) # Vector of 100 normals with mean 5 and sd 2  
threeNumberSummary(x)
```

mean	median	sd
4.926875	5.050984	1.953703

If-else statement

- 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

```
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.

Workshop 5 : Transform

For the first two problems we'll use the Cars93 data set from the MASS library.

```
library(MASS)
```

1. Manipulating data frames

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).

2. Functions, lists, and if-else

- (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.
- (b) Write a function called **sendMessage** whose input x is a number, and which prints **Congratulations** if **isPassingGrade(x)** is TRUE and prints **Oh no!** if **isPassingGrade(x)** is FALSE.



Data Wrangling

Common Problem

- One of the most common problems you'll encounter when importing manually-entered data is inconsistent data types within columns
- For a simple example, let's look at TVhours column in a messy version of the survey data

```
survey.messy <- read.csv("survey_messy.csv", header=TRUE)  
survey.messy$TVhours
```

```
[1] 1           8h          4           5  
[5] 3           ~10         0           10  
[9] 15 (incl movies) 4           0           0  
[13] 10          4           10          2hours  
[17] 5           3           0           1  
[21] 2           7           3           10  
[25] 6.5          40          0           12  
[29] adfjalkj     3           0           ...  
16 Levels: ~10 0 1 10 12 15 (incl movies) 2 2hours 3 4 40 5 6.5 7 ... adfjalkj
```

What are the problems?

```
str(survey.messy)
```

```
'data.frame': 31 obs. of 6 variables:  
 $ Program      : Factor w/ 3 levels "MISM","Other",...: 1 2 1 3 2 3 2 1 3 3 ...  
 $ PriorExp     : Factor w/ 3 levels "Extensive experience",...: 3 3 1 2 2 3 3 3  
 2 1 ...  
 $ Rexperience   : Factor w/ 3 levels "Basic competence",...: 3 1 1 2 3 3 3 3 3 3  
 ...  
 $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 2 2 2 1 1 2 1 2  
 ...  
 $ TVhours       : Factor w/ 16 levels "-10","0","1",...: 3 15 10 12 9 1 2 4 6 10  
 ...  
 $ Editor        : Factor w/ 1 level "Microsoft Word": 1 1 1 1 1 1 1 1 1 1 ...
```

- Several of the entries have non-numeric values in them (they contain strings)
- As a result, TVhours is being imported as factor

Fix the type

```
as.character(tv.hours.messy)[1:30]
```

```
[1] "1"          "8h"        "4"  
[4] "5"          "3"          "~10"  
[7] "0"          "10"         "15 (incl movies)"  
[10] "4"          "0"          "0"  
[13] "10"         "4"          "10"  
[16] "2hours"     "5"          "3"  
[19] "0"          "1"          "2"  
[22] "7"          "3"          "10"  
[25] "6.5"        "40"         "0"  
[28] "12"         "adfjalkj"   "3"
```

```
as.numeric(as.character(tv.hours.messy))[1:30]
```

```
[1] 1.0    NA    4.0   5.0   3.0   NA    0.0  10.0   NA   4.0   0.0  0.0  10.0  4.0  
[15] 10.0   NA    5.0   3.0   0.0   1.0   2.0  7.0    3.0  10.0  6.5  40.0  0.0  12.0  
[29]    NA    3.0
```

```
typeof(as.numeric(as.character(tv.hours.messy))) # Success!! (Almost...)
```

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

A small improvement

- All the corrupted cells now appear as NA, which is R's missing indicator
- We can do a little better by cleaning up the vector once we get it to character form

```
tv.hours.strings <- as.character(tv.hours.messy)  
tv.hours.strings
```

```
[1] "1"                 "8h"                "4"  
[4] "5"                 "3"                  "~10"  
[7] "0"                 "10"                "15 (incl movies)"  
[10] "4"                "0"                  "0"  
[13] "10"               "4"                  "10"  
[16] "2hours"           "5"                  "3"  
[19] "0"                 "1"                  "2"  
[22] "7"                 "3"                  "10"  
[25] "6.5"              "40"                "0"  
[28] "12"               "adfjalkj"          "3"  
[31] "0"
```

Deleting non-numeric (or .) characters

```
tv.hours.strings
```

```
[1] "1"                 "8h"                "4"  
[4] "5"                 "3"                  "~10"  
[7] "0"                 "10"               "15 (incl movies)"  
[10] "4"                "0"                  "0"  
[13] "10"               "4"                  "10"  
[16] "2hours"            "5"                  "3"  
[19] "0"                 "1"                  "2"  
[22] "7"                 "3"                  "10"  
[25] "6.5"               "40"                 "0"  
[28] "12"                "adfjalkj"           "3"  
[31] "0"
```

```
# Use gsub() to replace everything except digits and '.' with a blank ""  
gsub("[^0-9.]", "", tv.hours.strings)
```

```
[1] "1"    "8"    "4"    "5"    "3"    "10"   "0"    "10"   "15"   "4"    "0"  
[12] "0"   "10"   "4"    "10"   "2"    "5"    "3"    "0"    "1"    "2"    "7"  
[23] "3"   "10"   "6.5"  "40"   "0"    "12"   ""     "3"    "0"
```

Redo

```
tv.hours.messy
```

```
[1] 1          8h          4          5
[5] 3          ~10         0          10
[9] 15 (incl movies) 4          0          0
[13] 10         4          10         2hours
[17] 5          3          0          1
[21] 2          7          3          10
[25] 6.5        40         0          12
[29] adfjalkj   3          0          ...
16 Levels: ~10 0 1 10 12 15 (incl movies) 2 2hours 3 4 40 5 6.5 7 ... adfjalkj
```

```
tv.hours.clean <- as.numeric(gsub("[^0-9.]", "", tv.hours.strings))
tv.hours.clean
```

```
[1] 1.0 8.0 4.0 5.0 3.0 10.0 0.0 10.0 15.0 4.0 0.0 0.0 10.0 4.0
[15] 10.0 2.0 5.0 3.0 0.0 1.0 2.0 7.0 3.0 10.0 6.5 40.0 0.0 12.0
[29] NA   3.0 0.0
```

Another approach

- We can also handle this problem by setting `stringsAsFactors = FALSE` when importing our data.

```
survey.messy <- read.csv("survey_messy.csv", header=TRUE, stringsAsFactors=FALSE)
str(survey.messy)
```

```
'data.frame': 31 obs. of 6 variables:
 $ Program      : chr  "MISM" "Other" "MISM" "PPM" ...
 $ PriorExp     : chr  "Some experience" "Some experience" "Extensive
experience" "Never programmed before" ...
 $ Rexperience   : chr  "Never used" "Basic competence" "Basic competence"
"Installed on machine" ...
 $ OperatingSystem: chr  "Windows" "Windows" "Windows" "Windows" ...
 $ TVhours       : chr  "1" "8h" "4" "5" ...
 $ Editor        : chr  "Microsoft Word" "Microsoft Word" "Microsoft Word"
"Microsoft Word" ...
```

- Now everything is a character instead of a factor

One-line Cleanup

- Let's clean up the `TVhours` column and cast it to numeric all in one command

```
survey <- transform(survey.messy, TVhours = as.numeric(gsub("[^0-9.]", "", TVhours)))
str(survey)
```

```
'data.frame': 31 obs. of 6 variables:
 $ Program      : chr  "MISM" "Other" "MISM" "PPM" ...
 $ PriorExp     : chr  "Some experience" "Some experience" "Extensive
experience" "Never programmed before" ...
 $ Rexperience   : chr  "Never used" "Basic competence" "Basic competence"
"Installed on machine" ...
 $ OperatingSystem: chr  "Windows" "Windows" "Windows" "Windows" ...
 $ TVhours       : num  1 8 4 5 3 10 0 10 15 4 ...
 $ Editor        : chr  "Microsoft Word" "Microsoft Word" "Microsoft Word"
"Microsoft Word" ...
```

What about other character column?

```
table(survey[["Program"]])
```

MISM	Other	PPM
10	6	15

```
table(as.factor(survey[["Program"]]))
```

MISM	Other	PPM
10	6	15

- Having factors coded as characters may be OK for many parts of our analysis

Let's fix it

```
# Figure out which columns are coded as characters  
chr.indexes <- sapply(survey, FUN = is.character)  
chr.indexes
```

Program	PriorExp	Rexperience	OperatingSystem
TRUE	TRUE	TRUE	TRUE
TVhours	Editor		
FALSE	TRUE		

```
# Re-code all of the character columns to factors  
survey[chr.indexes] <- lapply(survey[chr.indexes], FUN = as.factor)
```

Here is the outcome

```
str(survey)
```

```
'data.frame': 31 obs. of 6 variables:  
 $ Program      : Factor w/ 3 levels "MISM","Other",...: 1 2 1 3 2 3 2 1 3 3 ...  
 $ PriorExp     : Factor w/ 3 levels "Extensive experience",...: 3 3 1 2 2 3 3 3  
 2 1 ...  
 $ Rexperience   : Factor w/ 3 levels "Basic competence",...: 3 1 1 2 3 3 3 3 3 3  
 ...  
 $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 2 2 2 1 1 2 1 2  
 ...  
 $ TVhours       : num  1 8 4 5 3 10 0 10 15 4 ...  
 $ Editor        : Factor w/ 1 level "Microsoft Word": 1 1 1 1 1 1 1 1 1 1 ...
```

- Success!

Loop

```
for(i in 1:4) {  
  print(i)  
}
```

```
[1] 1  
[1] 2  
[1] 3  
[1] 4
```

```
phrase <- "Good Night, "  
for(word in c("and", "Good", "Luck")) {  
  phrase <- paste(phrase, word)  
  print(phrase)  
}
```

```
[1] "Good Night, and"  
[1] "Good Night, and Good"  
[1] "Good Night, and Good Luck"
```

Loop Syntax

A **for loop** executes a chunk of code for every value of an **index variable** in an **index set**

- The basic syntax takes the form

```
for(index.variable in index.set) {  
    code to be repeated at every value of index.variable  
}
```

- The index set is often a vector of integers, but can be more general

Example

```
index.set <- rnorm(4) # 4 random standard normal variables  
index.set
```

```
[1] -1.9912818 0.7119512 0.3999575 0.5872188
```

```
for(i in index.set) {  
  print(abs(i))  
}
```

```
[1] 1.991282  
[1] 0.7119512  
[1] 0.3999575  
[1] 0.5872188
```

Example

```
index.set <- list(name="Michael", weight=185, is.male=TRUE) # a list
for(i in index.set) {
  print(c(i, typeof(i)))
}
```

```
[1] "Michael"    "character"
[1] "185"        "double"
[1] "TRUE"       "logical"
```

Example

```
fake.data <- matrix(rnorm(500), ncol=5) # create fake 100 x 5 data set  
head(fake.data,2) # print first two rows
```

```
 [,1]      [,2]      [,3]      [,4]      [,5]  
[1,] 0.9551196 -0.3985364 -0.2338078 -1.018674 -0.684313  
[2,] 0.6483192 -0.5487889  0.2091660  2.016935 -1.030668
```

```
col.sums <- numeric(ncol(fake.data)) # variable to store running column sums  
for(i in 1:nrow(fake.data)) {  
  col.sums <- col.sums + fake.data[i,] # add ith observation to the sum  
}  
print(col.sums)
```

```
[1] -12.808804 -11.552563   3.211727   4.049877 -9.901475
```

```
colSums(fake.data) # A better approach (see also colMeans())
```

```
[1] -12.808804 -11.552563   3.211727   4.049877 -9.901475
```

While loop

- **while loops** repeat a chunk of code while the specified condition remains true

```
day <- 1
num.days <- 365
while(day <= num.days) {
  day <- day + 1
}
```

- We won't really be using while loops in this class

Various apply() function

Command	Description
<code>apply(x, MARGIN, FUN)</code>	Obtain a vector/array/list by applying <code>FUN</code> along the specified <code>MARGIN</code> of an array or matrix <code>x</code>
<code>lapply(x, FUN)</code>	Obtain a list by applying <code>FUN</code> to the elements of a list <code>x</code>
<code>sapply(x, FUN)</code>	Simplified version of <code>lapply</code> . Returns a vector/array instead of list.
<code>tapply(x, INDEX, FUN)</code>	Obtain a table by applying <code>FUN</code> to each combination of the factors given in <code>INDEX</code>

- These functions are (good!) alternatives to loops
- They are typically *more efficient* than loops (often run considerably faster on large data sets)
- Take practice to get used to, but make analysis easier to debug and less prone to error when used effectively

Example: apply()

```
colMeans(fake.data)
```

```
[1] -0.12808804 -0.11552563  0.03211727  0.04049877 -0.09901475
```

```
apply(fake.data, MARGIN=2, FUN=mean) # MARGIN = 1 for rows, 2 for columns
```

```
[1] -0.12808804 -0.11552563  0.03211727  0.04049877 -0.09901475
```

```
# Function that calculates proportion of vector indexes that are > 0
propPositive <- function(x) mean(x > 0)
apply(fake.data, MARGIN=2, FUN=propPositive)
```

```
[1] 0.51 0.47 0.51 0.51 0.46
```

Example: lapply(), sapply()

```
lapply(survey, is.factor) # Returns a list
```

```
$Program  
[1] TRUE  
  
$PriorExp  
[1] TRUE  
  
$Rexperience  
[1] TRUE  
  
$OperatingSystem  
[1] TRUE  
  
$TVhours  
[1] FALSE  
  
$Editor  
[1] TRUE
```

```
sapply(survey, FUN = is.factor) # Returns a vector with named elements
```

Program	PriorExp	Rexperience	OperatingSystem
TRUE	TRUE	TRUE	TRUE
TVhours	Editor		
FALSE	TRUE		

Example: apply, lapply, sapply

```
apply(cars, 2, FUN=mean) # Data frames are arrays
```

```
speed dist  
15.40 42.98
```

```
lapply(cars, FUN=mean) # Data frames are also lists
```

```
$speed  
[1] 15.4
```

```
$dist  
[1] 42.98
```

```
sapply(cars, FUN=mean) # sapply() is just simplified lapply()
```

```
speed dist  
15.40 42.98
```

tapply()

- Think of tapply() as a generalized form of the table() function

```
library(MASS)
# Get a count table, data broken down by Origin and DriveTrain
table(Cars93$Origin, Cars93$DriveTrain)
```

	4WD	Front	Rear
USA	5	34	9
non-USA	5	33	7

```
# Calculate average MPG.City, broken down by Origin and Drivetrain
tapply(Cars93$MPG.city, INDEX = Cars93[c("Origin", "DriveTrain")], FUN=mean)
```

Origin	DriveTrain		
	4WD	Front	Rear
USA	17.6	22.14706	18.33333
non-USA	23.4	24.93939	19.14286

Example: tapply()

- Let's get the average horsepower by car Origin and Type

```
tapply(Cars93[["Horsepower"]], INDEX = Cars93[c("Origin", "Type")], FUN=mean)
```

Origin	Type					
	Compact	Large	Midsize	Small	Sporty	Van
USA	117.4286	179.4545	153.5000	89.42857	166.5000	158.40
non-USA	141.5556	NA	189.4167	91.78571	151.6667	138.25

- What's that NA doing there?

```
any(Cars93$Origin == "non-USA" & Cars93>Type == "Large")
```

```
[1] FALSE
```

- None of the non-USA manufacturers produced Large cars!

with()

- Thus far we've repeatedly typed out the data frame name when referencing its columns
- This is because the data variables don't exist in our working environment
- Using **with**(*data*, *expr*) lets us specify that the code in *expr* should be evaluated in an environment that contains the elements of *data* as variables

```
with(Cars93, table(Origin, Type))
```

Origin	Type					
	Compact	Large	Midsize	Small	Sporty	Van
USA	7	11	10	7	8	5
non-USA	9	0	12	14	6	4

Example: with()

```
any(Cars93$Origin == "non-USA" & Cars93>Type == "Large")
```

```
[1] FALSE
```

```
with(Cars93, any(Origin == "non-USA" & Type == "Large")) # Same effect!
```

```
[1] FALSE
```

```
with(Cars93, tapply(Horsepower, INDEX = list(Origin, Type), FUN=mean))
```

	Compact	Large	Midsize	Small	Sporty	Van
USA	117.4286	179.4545	153.5000	89.42857	166.5000	158.40
non-USA	141.5556	NA	189.4167	91.78571	151.6667	138.25

- Using `with()` makes code simpler, easier to read, and easier to debug

Workshop 6: Loop

Loop practice

- (a) Write a function called `calculateRowMeans` that uses a `for` loop to calculate the row means of a matrix `x`.
- (b) Try out your function on the random matrix `fake.data` defined below.
- (b) Use the `apply()` function to calculate the row means of the matrix `fake.data`
- (c) Compare this to the output of the `rowMeans()` function to check that your calculation is correct.

Statistics Analysis

Statistical Analysis

- Descriptive Statistics
- Inferential Statistics

Descriptive Statistics

Descriptive statistics are broken down into two categories. Measures of central tendency and measures of variability (spread).

- **Measure of Central Tendency**

Central tendency refers to the idea that there is one number that best summarizes the entire set of measurements, a number that is in some way “central” to the set.

- MEAN
- Median
- Mode

Descriptive Statistics

- **Measure of Spread / Dispersion**

Measure of Spread refers to the idea of variability within your data.

- **Standard Deviation**

Standard deviation is the measurement of average distance between each quantity and mean. That is, how data is spread out from mean. A low standard deviation indicates that the data points tend to be close to the mean of the data set, while a high standard deviation indicates that the data points are spread out over a wider range of values.

$$\text{S.D.} = \sqrt{\frac{1}{n-1} \sum_{i=0}^n (x - \bar{x})^2}$$

$$\text{S.D.} = \sqrt{\frac{1}{n} \sum_{i=0}^n (x - \mu)^2}$$



- **Mean Deviation / Mean Absolute Deviation**

It is an average of absolute differences between each value in a set of values, and the average of all values of that set

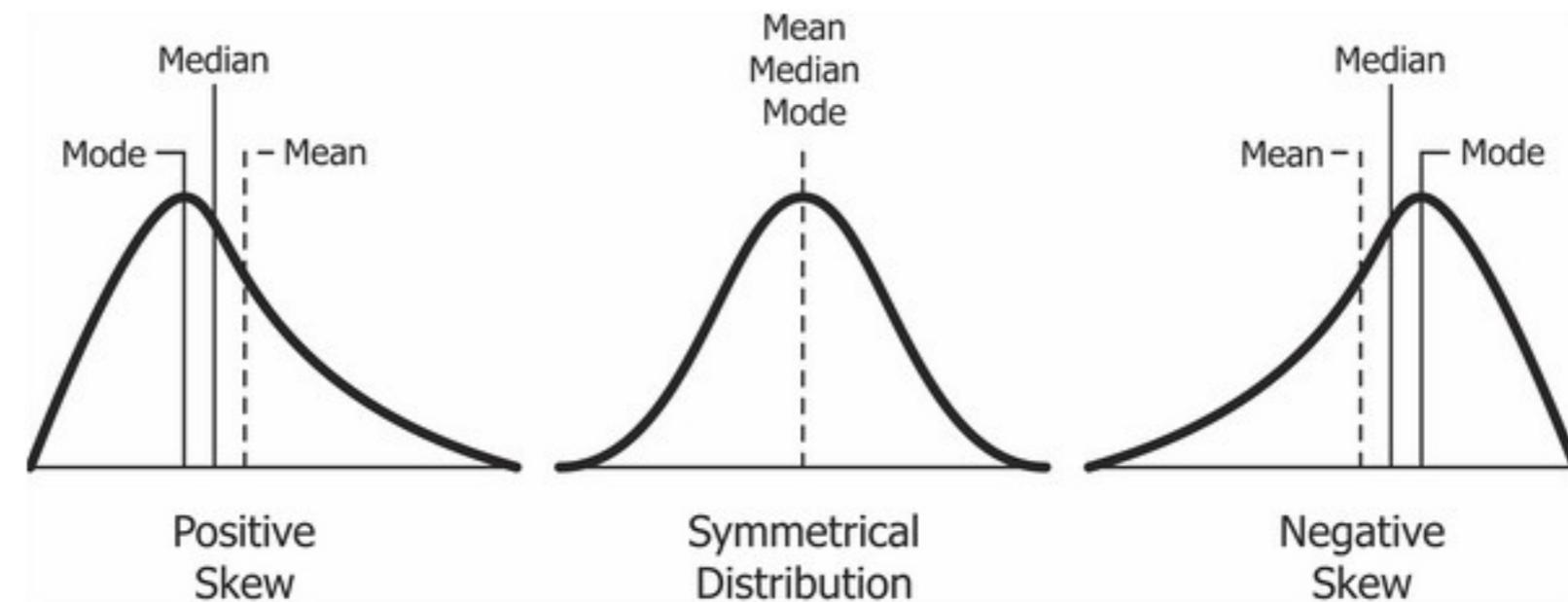
$$\text{M.D.} = \frac{1}{n} \sum_{i=0}^n |x_i - \bar{x}|$$

- **Variance** **Variance** = $(S.D.)^2$
- **Range**
- **Percentile**
- **Quartiles**

Descriptive Statistics

- **Skewness**

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or undefined.



Descriptive Statistics

- Pearson First Coefficient of Skewness (Mode skewness)

$$\frac{\text{Mean} - \text{Mode}}{\text{Standard Deviation}}$$

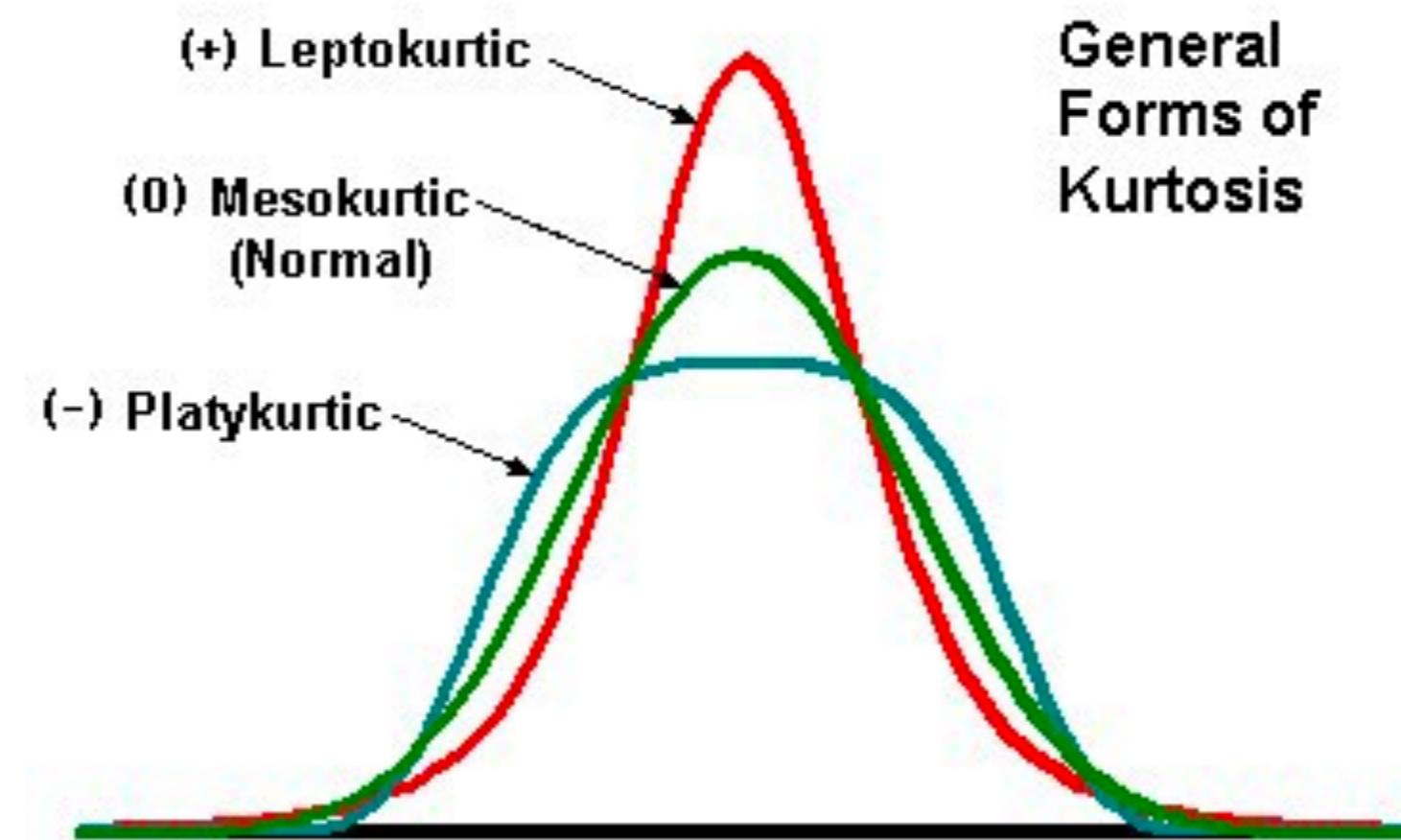
- Pearson Second Coefficient of Skewness (Median skewness)

$$\frac{3(\text{Mean} - \text{Median})}{\text{Standard Deviation}}$$

Descriptive Statistics

- **Kurtosis**

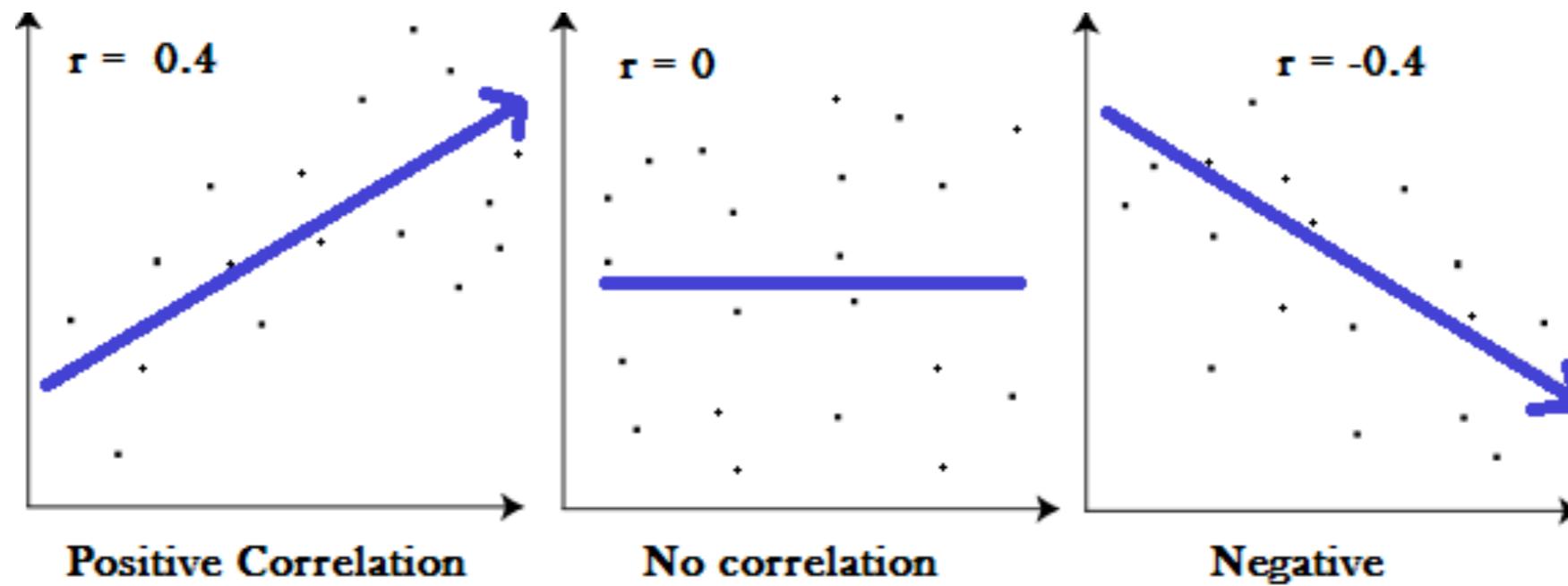
Kurtosis is a measure of whether the data are heavy-tailed (profusion of outliers) or light-tailed (lack of outliers) relative to a normal distribution.



Descriptive Statistics

- **Correlation**

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related.



Descriptive Statistics in R

Let's get started

- We're going to start by operating on the `birthwt` dataset from the MASS library
- Let's get it loaded and see what we're working with

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

```
'data.frame': 189 obs. of 10 variables:
 $ low   : int  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 ...
 $ ht    : int  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
- Let's get better column names (use `help(birthwt)` to understand the variables and come up with better names)

```
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
```

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

Output table

- Let's use the header `{r, results='asis'}`, along with the `kable()` function from the `knitr` library

```
library(knitr)
kable(with(birthwt, tapply(birthwt.grams, INDEX = list(race, mother.smokes), FUN
= mean)), format = "markdown")
```

	no	yes
black	2854.500	2504.000
other	2815.782	2757.167
white	3428.750	2826.846

- `kable()` outputs the table in a way that Markdown can read and nicely display
- Note: changing the CSS changes the table appearance

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

```
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

```
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

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

- We'll see later that aggregate output can be more convenient for plotting

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

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

- The odds of low bwt among non-smoking mothers is

```
or.smoke.bwt <- (weight.smoke.tbl[2,2] / weight.smoke.tbl[1,2]) /  
(weight.smoke.tbl[2,1] / weight.smoke.tbl[1,1])  
or.smoke.bwt
```

```
[1] 2.021944
```

- So the odds of low birth weight are 2 times higher when the mother smokes

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

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

```
[1] 0.09031781
```

- Does this change when we account for smoking status?

```
with(birthwt, cor(birthwt.grams[mother.smokes == "yes"],  
mother.age[mother.smokes == "yes"]))
```

```
[1] -0.1441649
```

```
with(birthwt, cor(birthwt.grams[mother.smokes == "no"], mother.age[mother.smokes  
== "no"]))
```

```
[1] 0.2014558
```

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[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
```

Skewness and Kurtosis

```
> library(moments)
> skewness(birthwt$mother.age)
[1] 0.7164391
> kurtosis(birthwt$mother.age)
[1] 3.568442
>
```

psych package

```
> library(psych)
> describe(birthwt)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	189	0.31	0.46	0	0.27	0.00	0	1	1	0.80	-1.36	0.03
2	2	189	23.24	5.30	23	22.90	5.93	14	45	31	0.71	0.53	0.39
3	3	189	129.81	30.58	121	126.07	20.76	80	250	170	1.38	2.25	2.22
4	4	189	1.85	0.92	1	1.81	0.00	1	3	2	0.31	-1.75	0.07
5	5	189	0.39	0.49	0	0.37	0.00	0	1	1	0.44	-1.82	0.04
6	6	189	0.20	0.49	0	0.08	0.00	0	3	3	2.76	8.17	0.04
7	7	189	0.06	0.24	0	0.00	0.00	0	1	1	3.55	10.67	0.02
8	8	189	0.15	0.36	0	0.07	0.00	0	1	1	1.97	1.87	0.03
9	9	189	0.79	1.06	0	0.62	0.00	0	6	6	1.56	3.00	0.08
10	10	189	2944.59	729.21	2977	2961.76	834.70	709	4990	4281	-0.21	-0.14	53.04

```
>
```



Exploratory Data Analysis

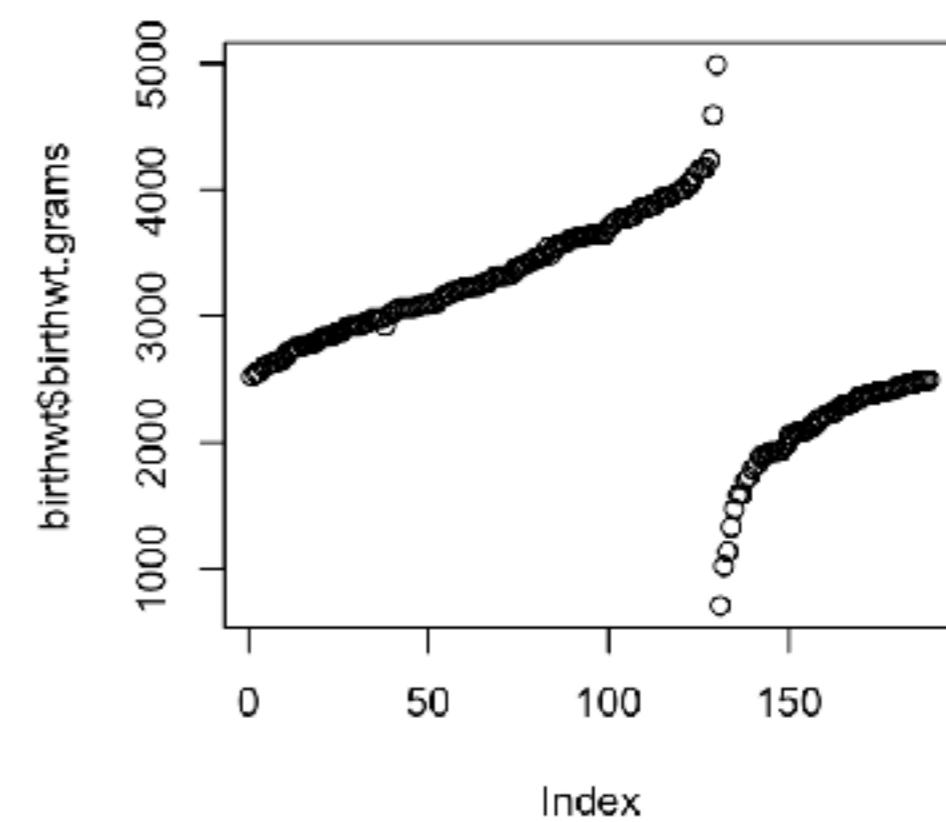
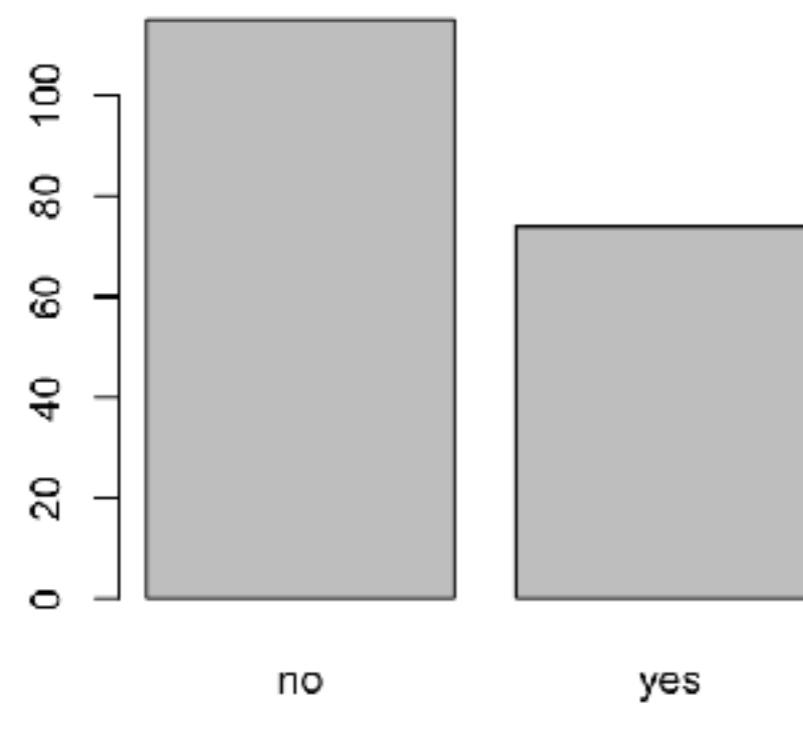
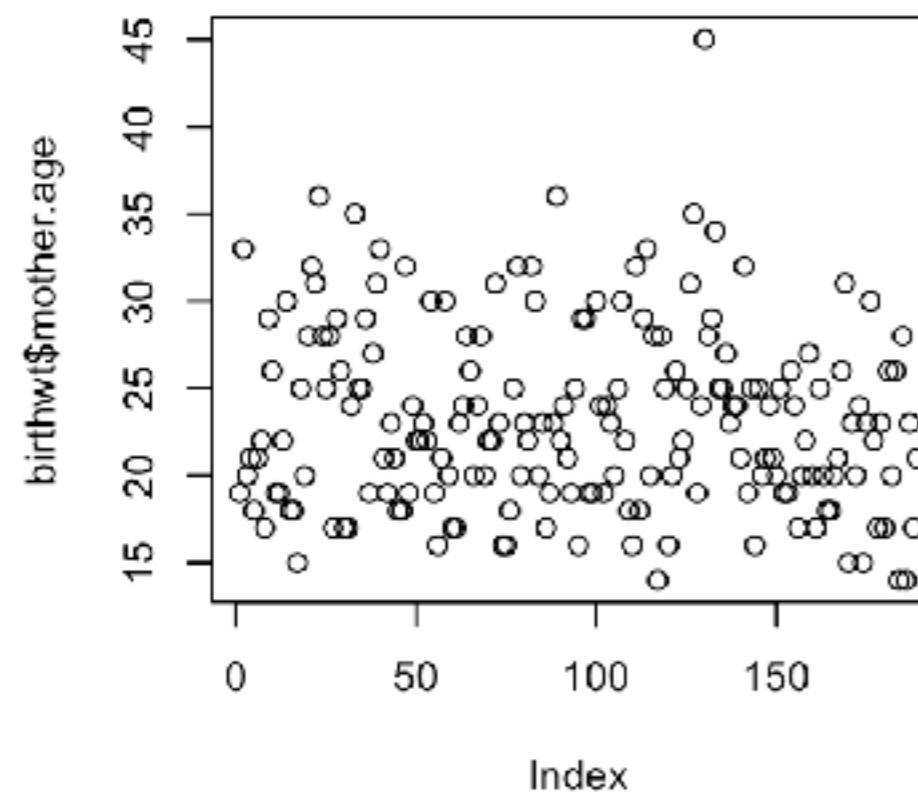
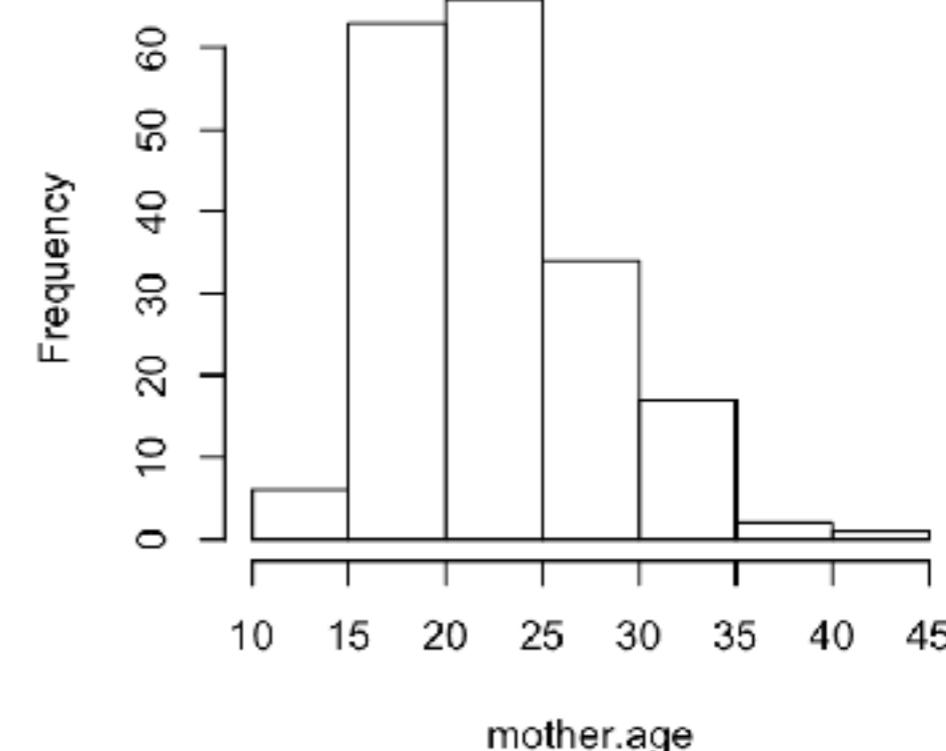
Basic single plot

Let's continue with the `birthwt` data from the `MASS` library.

Here are some basic single-variable plots.

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

Histogram of mother.age

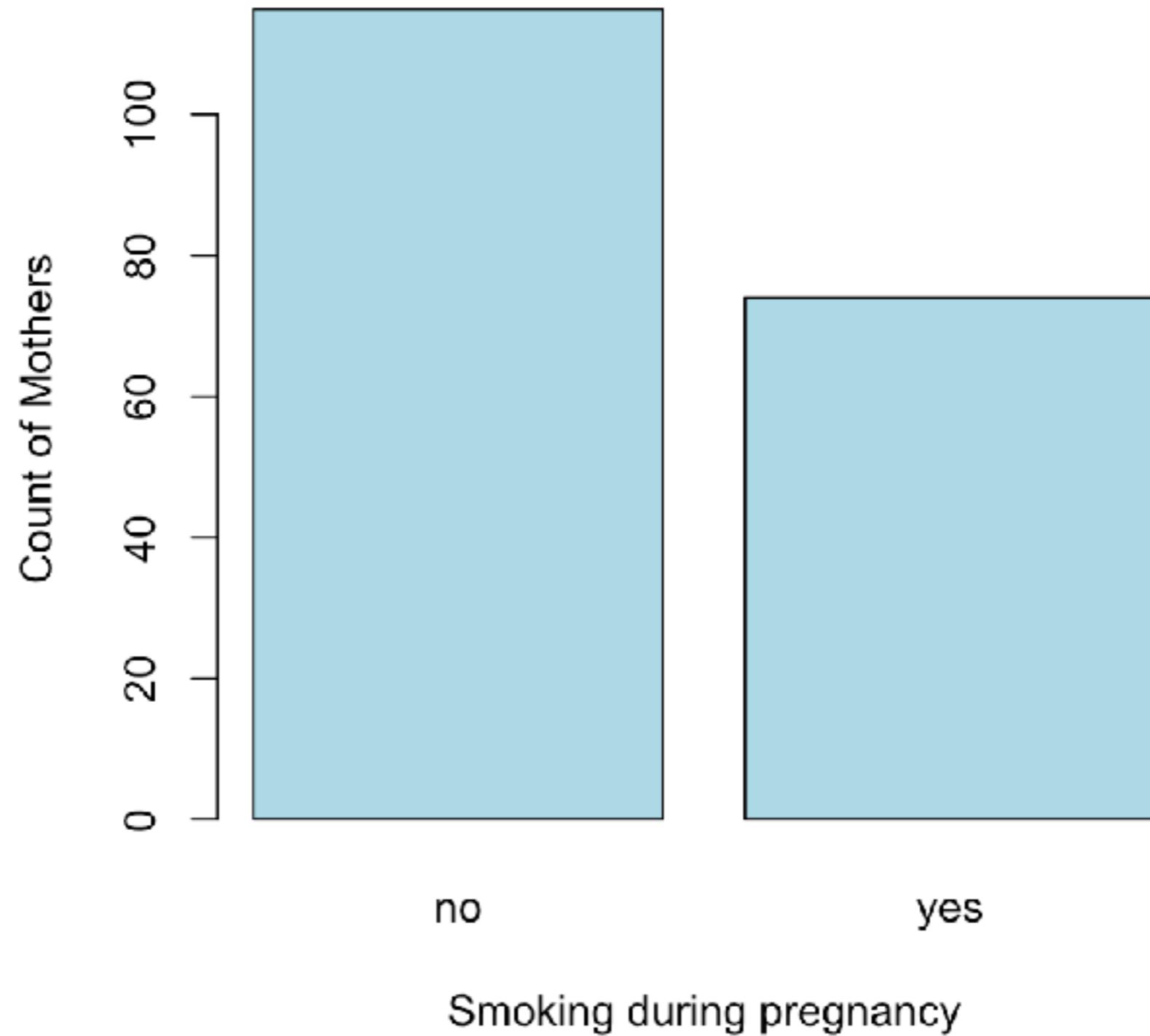


Another example

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

```
plot(birthwt$mother.smokes,  
      main = "Mothers Who Smoked In Pregnancy",  
      xlab = "Smoking during pregnancy",  
      ylab = "Count of Mothers",  
      col = 'lightblue')
```

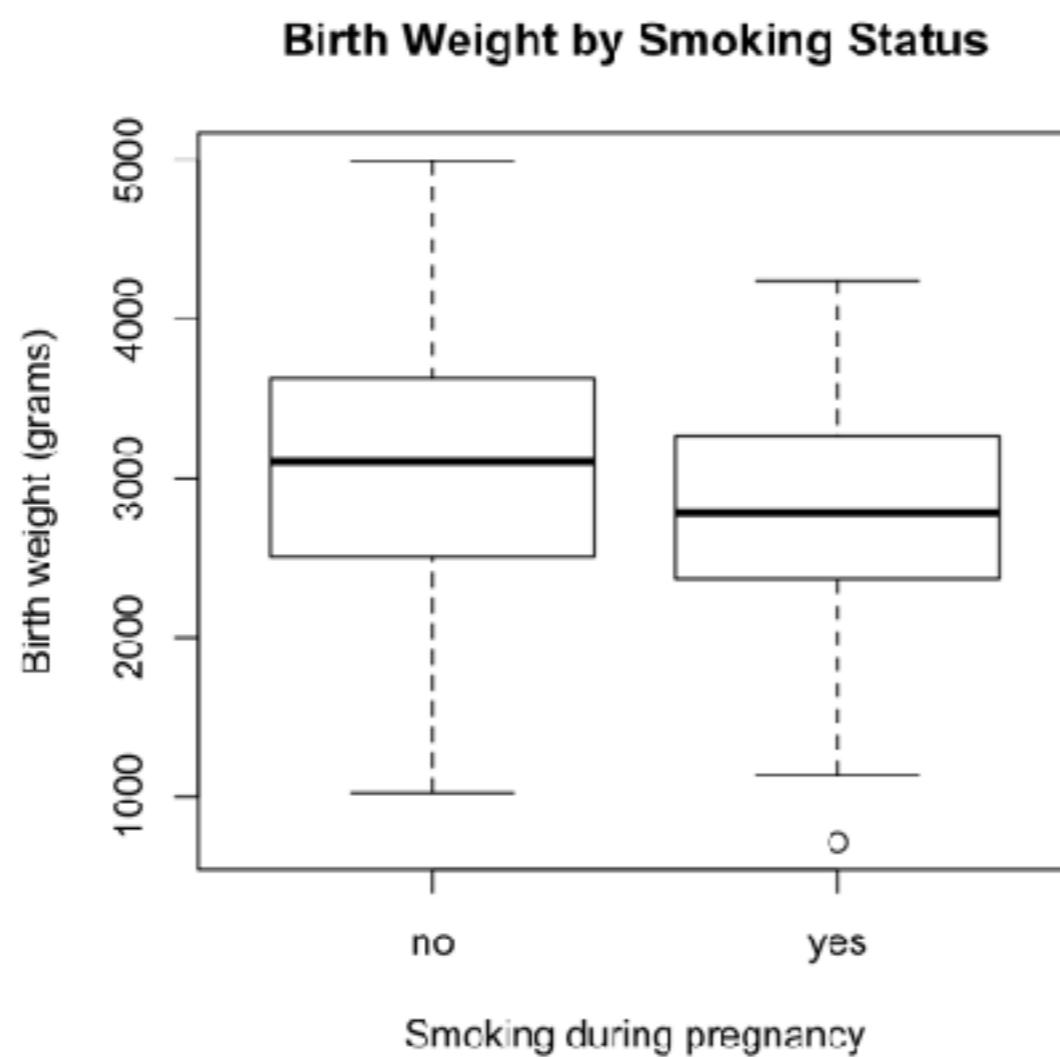
Mothers Who Smoked In Pregnancy



Plots with several variables

If we call `plot(x, y, ...)` with `x` a factor and `y` numeric, R will produce boxplots of `y` at every level of `x`.

```
with(birthwt, plot(mother.smokes, birthwt.grams,  
                    main = "Birth Weight by Smoking Status",  
                    xlab = "Smoking during pregnancy",  
                    ylab = "Birth weight (grams)"))
```



Plotting using ggplot2

What is ggplot2?

- An implementation of the Grammar of Graphics by Leland Wilkinson
- Written by Hadley Wickham (while he was a graduate student at Iowa State)
- A “third” graphics system for R (along with base and lattice)
- Available from CRAN via `install.packages()`
- Web site: <http://ggplot2.org> (better documentation)

What is ggplot2?

- Grammar of graphics represents and abstraction of graphics ideas/objects
- Think “verb”, “noun”, “adjective” for graphics
- Allows for a “theory” of graphics on which to build new graphics and graphics objects
- “Shorten the distance from mind to page”

Grammar of Graphics

- “In brief, the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system”
- from ggplot2 book

The Basics: qplot()

- Works much like the plot function in base graphics system
- Looks for data in a data frame, similar to lattice, or in the parent environment
- Plots are made up of aesthetics (size, shape, color) and geoms (points, lines)

The Basics: qplot()

Factors are important for indicating subsets of the data (if they are to have different properties); they should be labeled

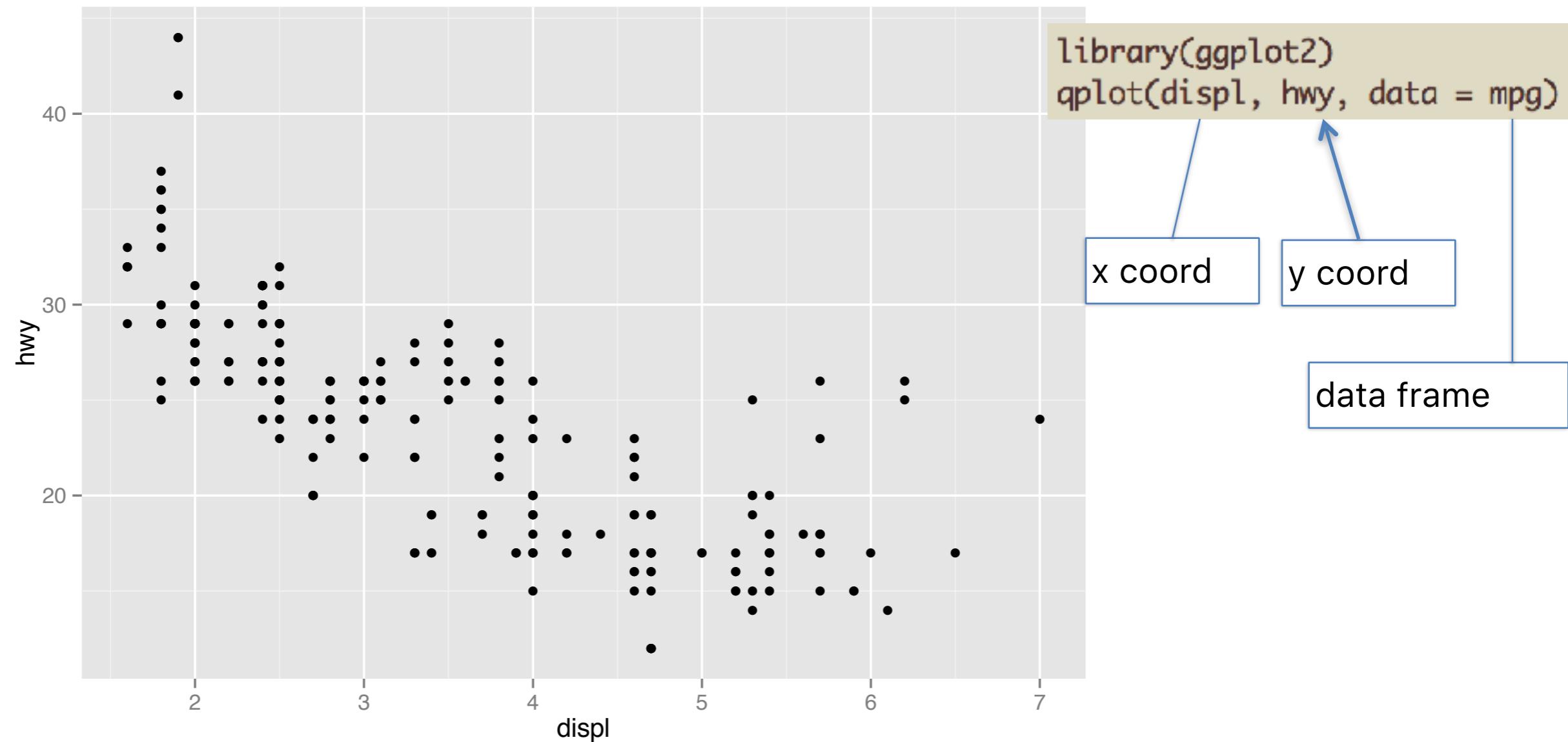
The qplot() hides what goes on underneath, which is okay for most operations

ggplot() is the core function and very flexible for doing things qplot() cannot do

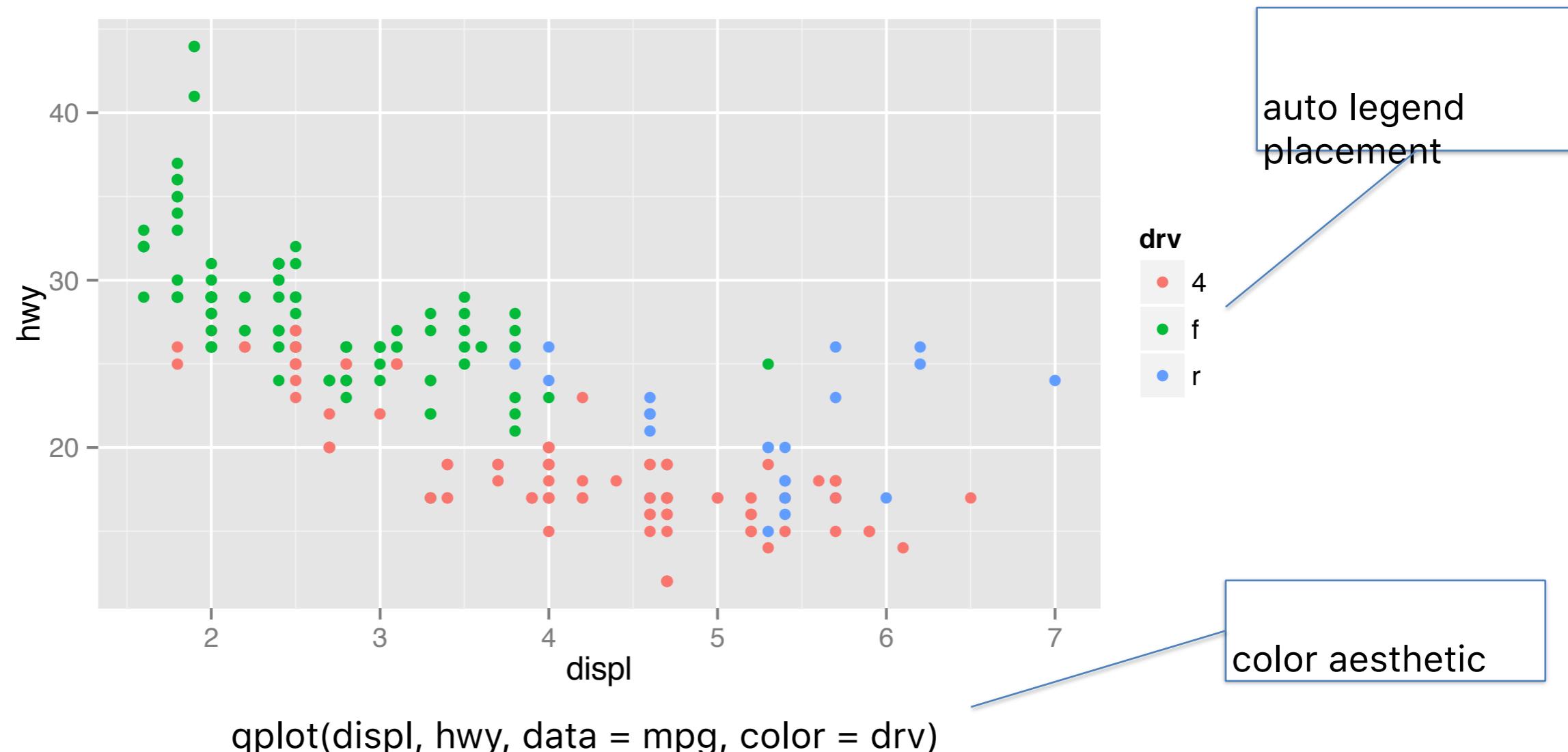
Example Dataset

```
> library(ggplot2)
> str(mpg)
'data.frame': 234 obs. of 11 variables:
 $ manufacturer: Factor w/ 15 levels "audi","chevrolet",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ model       : Factor w/ 38 levels "4runner 4wd",...: 2 2 2 2 2 2 2 3 3 3 ...
 $ displ        : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year         : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl          : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans        : Factor w/ 10 levels "auto(av)","auto(l3)",...: 4 9 10 1 4 9 1 9 4 10 ...
 ...
 $ drv          : Factor w/ 3 levels "4","f","r": 2 2 2 2 2 2 2 1 1 1 ...
 $ cty          : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy          : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl           : Factor w/ 5 levels "c","d","e","p",...: 4 4 4 4 4 4 4 4 4 4 ...
 $ class        : Factor w/ 7 levels "2seater","compact",...: 2 2 2 2 2 2 2 2 2 2 ...
```

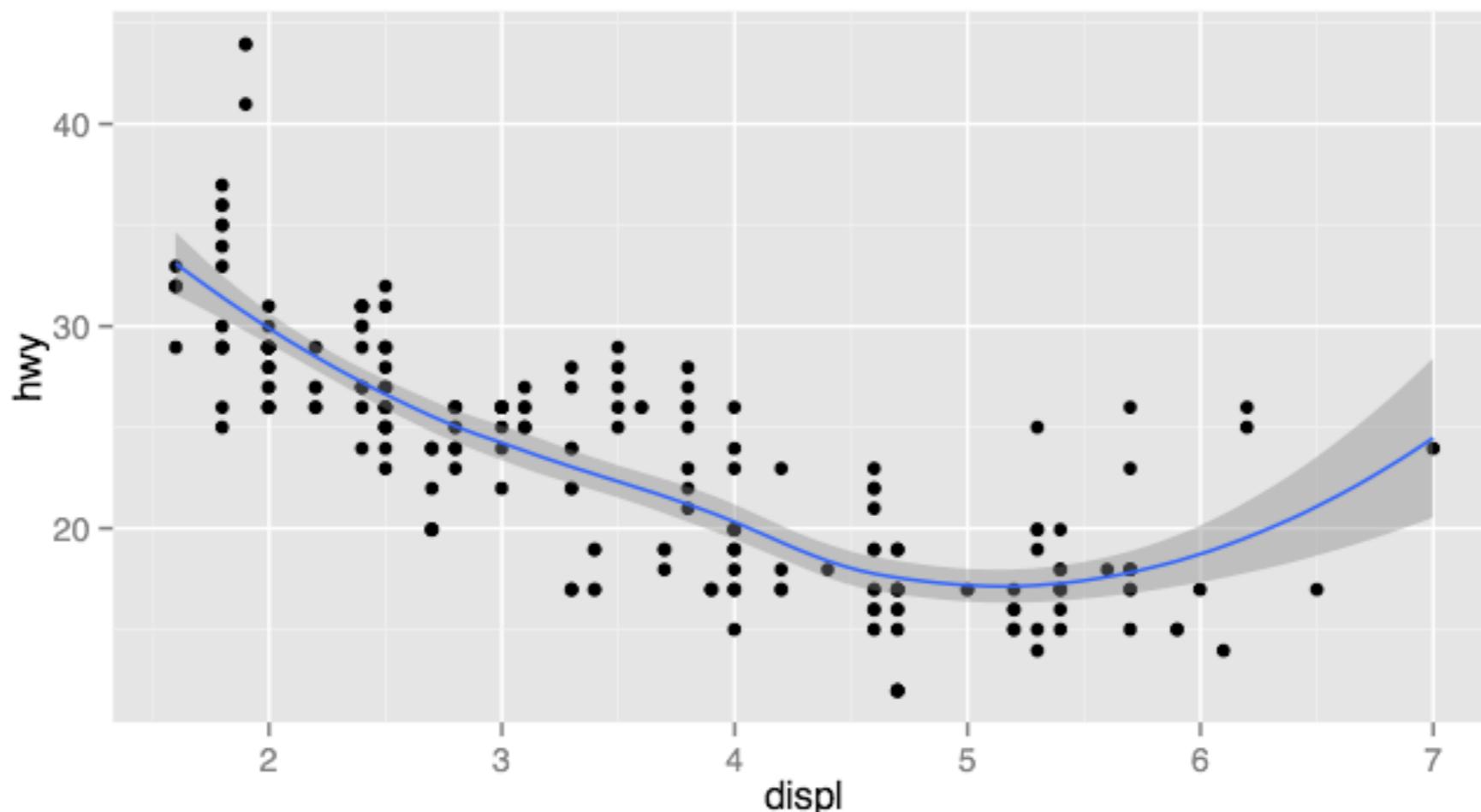
ggplot2 “Hello, world!”



Modifying aesthetics

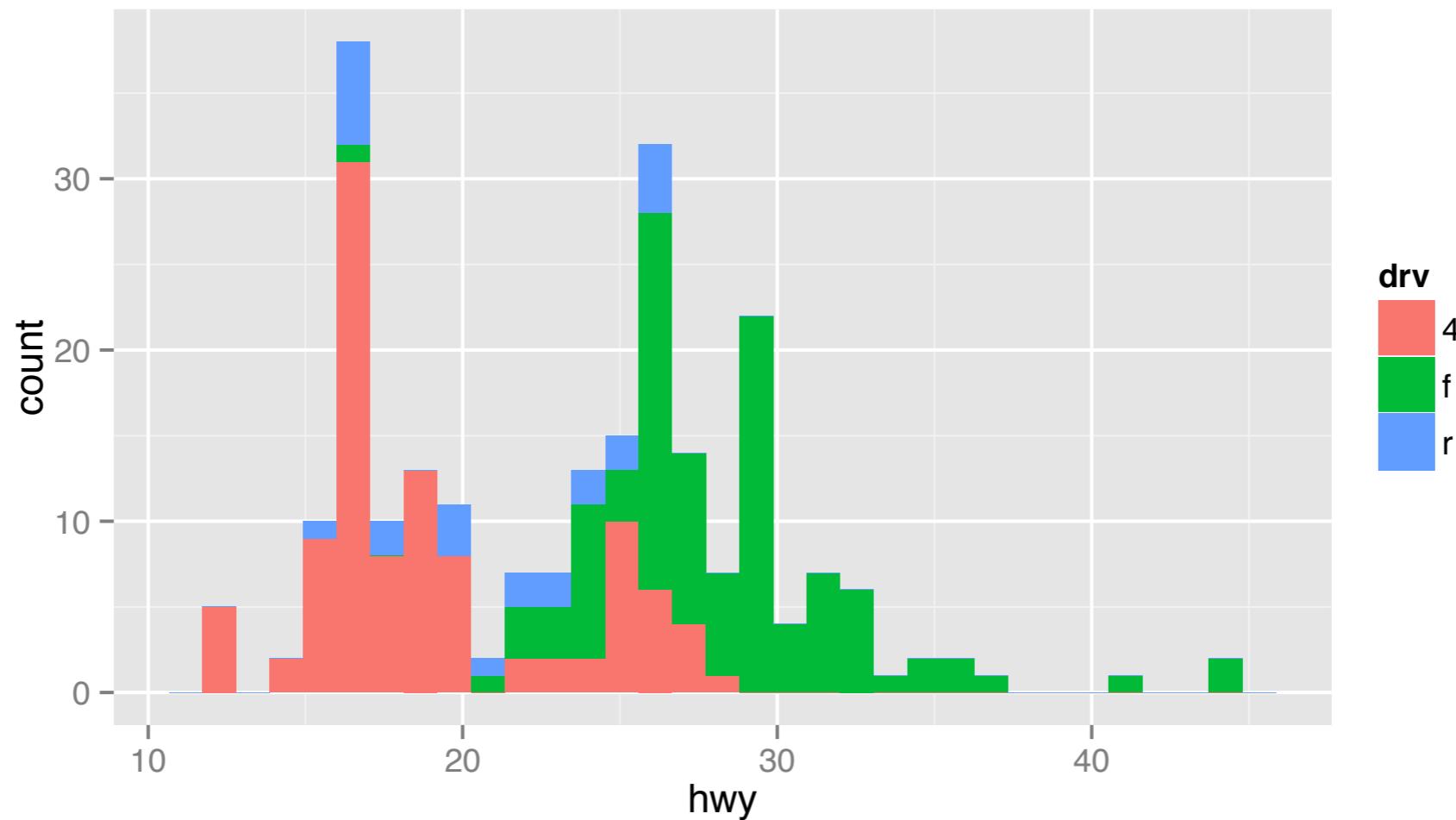


Adding a geom



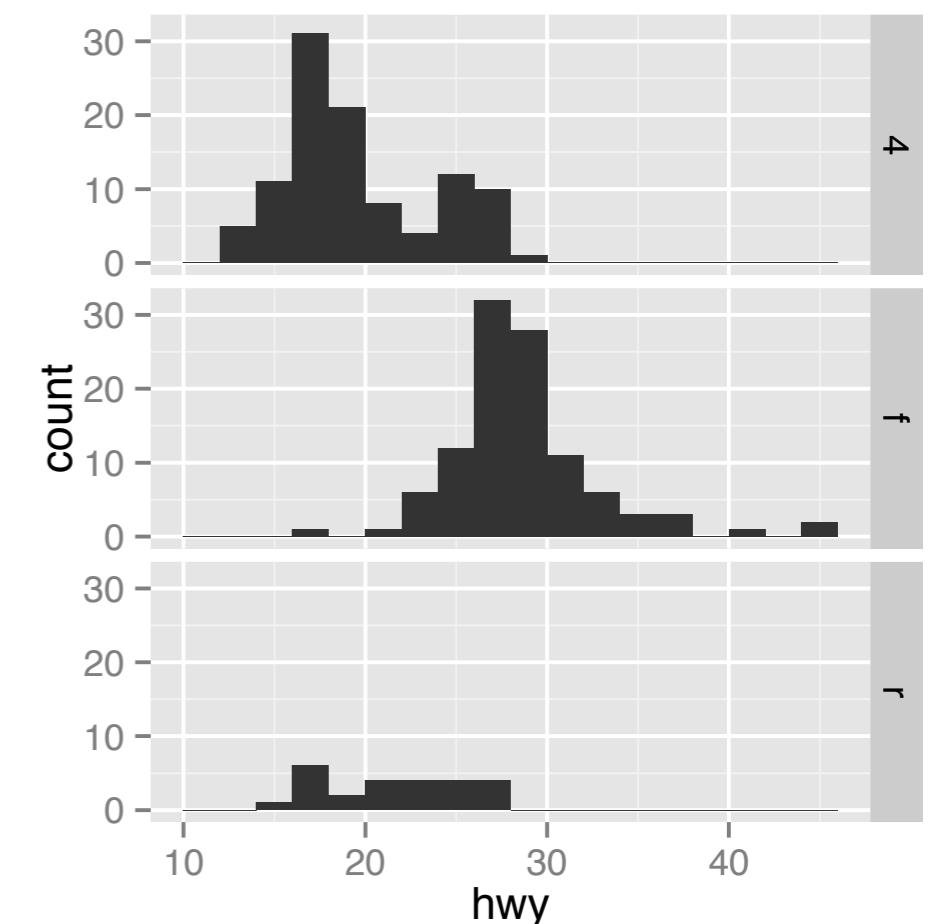
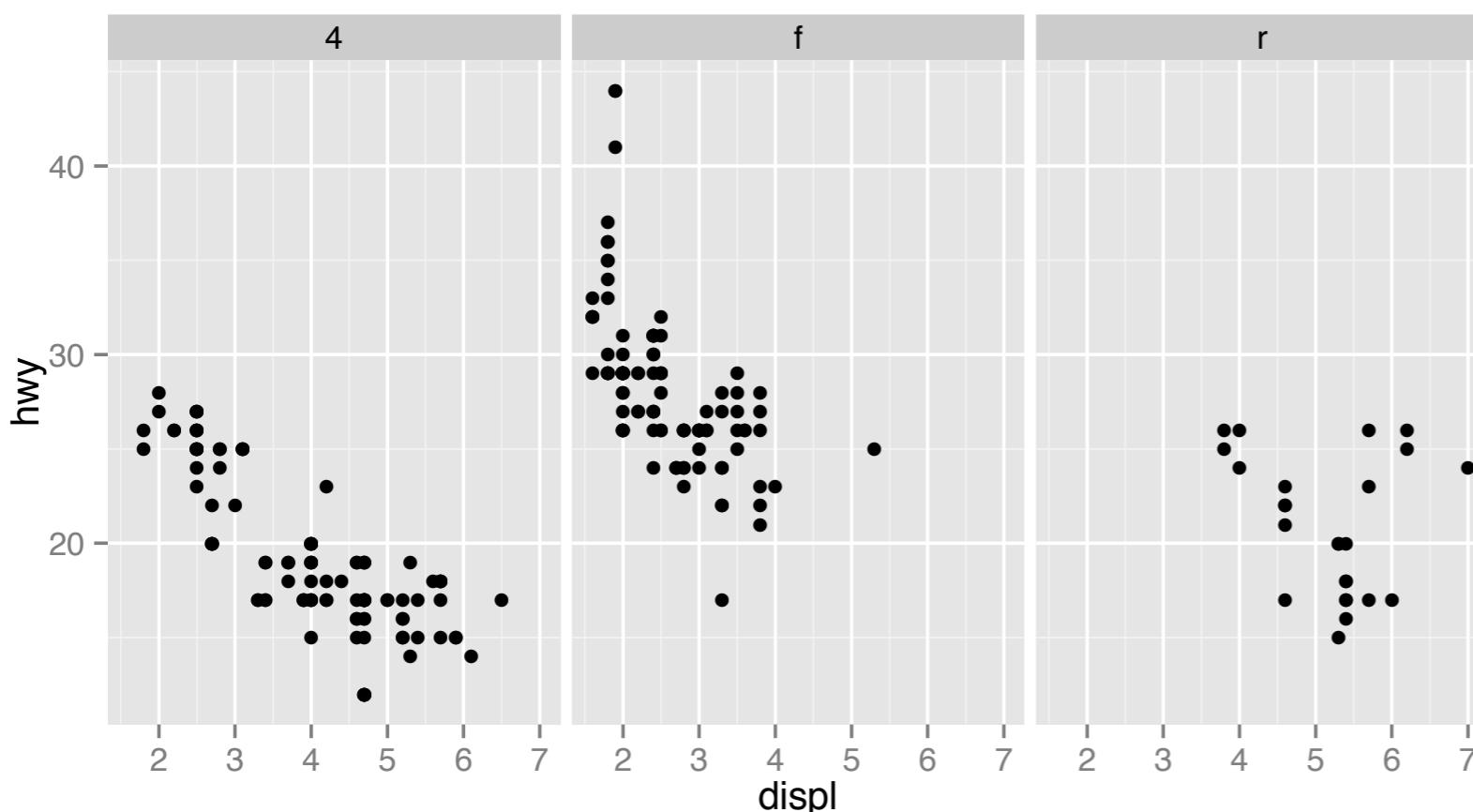
```
qplot(displ, hwy, data = mpg, geom = c("point", "smooth"))
```

Histograms



Facets

```
qplot(displ, hwy, data = mpg, facets = . ~ drv)
```



```
qplot(hwy, data = mpg, facets = drv ~ ., binwidth = 2)
```

MAACS Cohort

- Mouse Allergen and Asthma Cohort Study
- Baltimore children (aged 5—17)
- Persistent asthma, exacerbation in past year
- Study indoor environment and its relationship with asthma morbidity
- Recent publication: <http://goo.gl/WqE9j8>

Example: MAACS

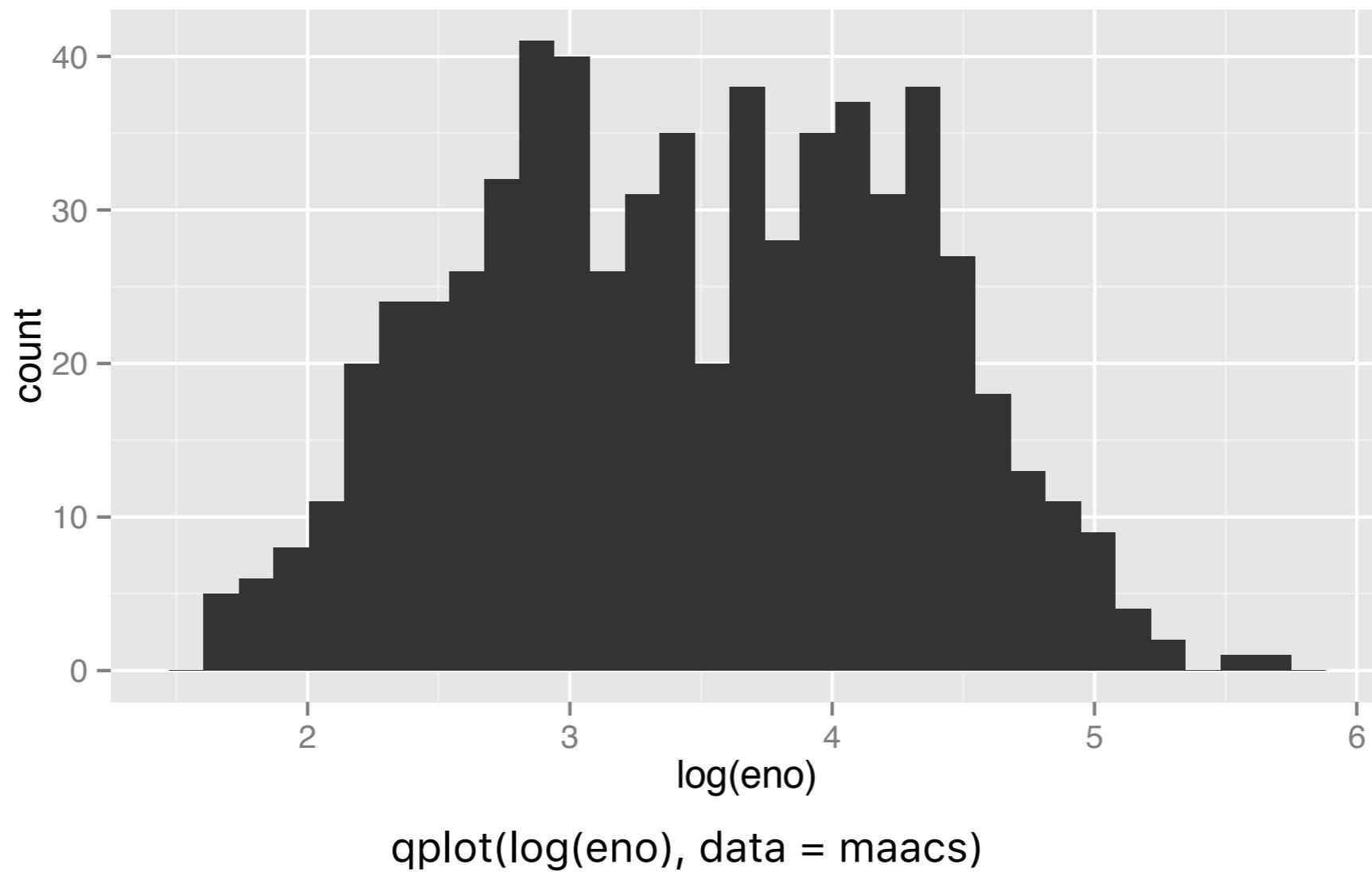
Exhaled nitric oxide

```
> str(maacs)
'data.frame': 750 obs. of 5 variables:
 $ id       : int 1 2 3 4 5 6 7 8 9 10 ...
 $ eno      : num 141 124 126 164 99 68 41 50 12 30 ...
 $ duBedMusM: num 2423 2793 3055 775 1634 ...
 $ pm25     : num 15.6 34.4 39 33.2 27.1 ...
 $ _mopos   : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
```

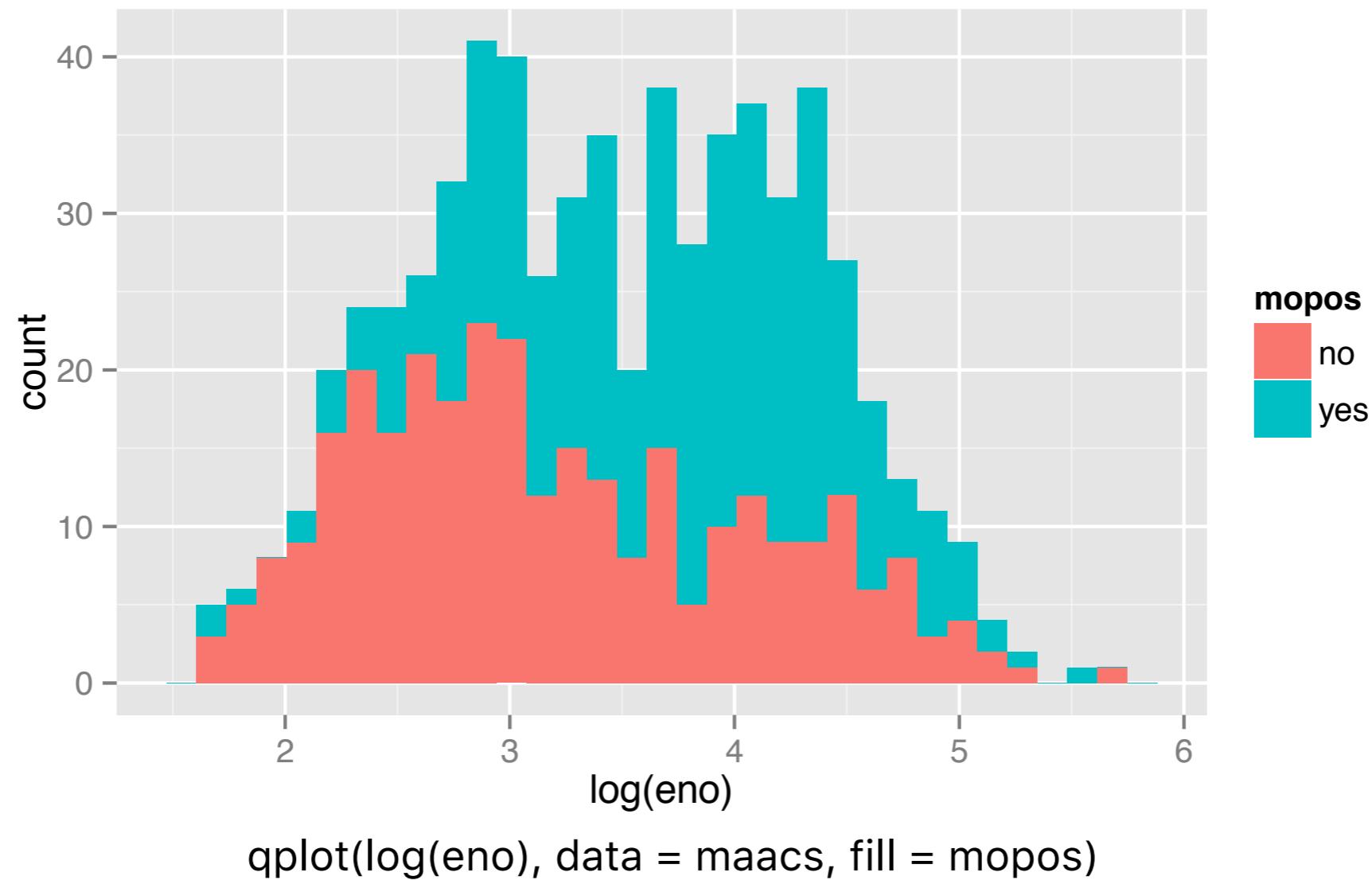
Fine particulate matter

Sensitized to mouse allergen

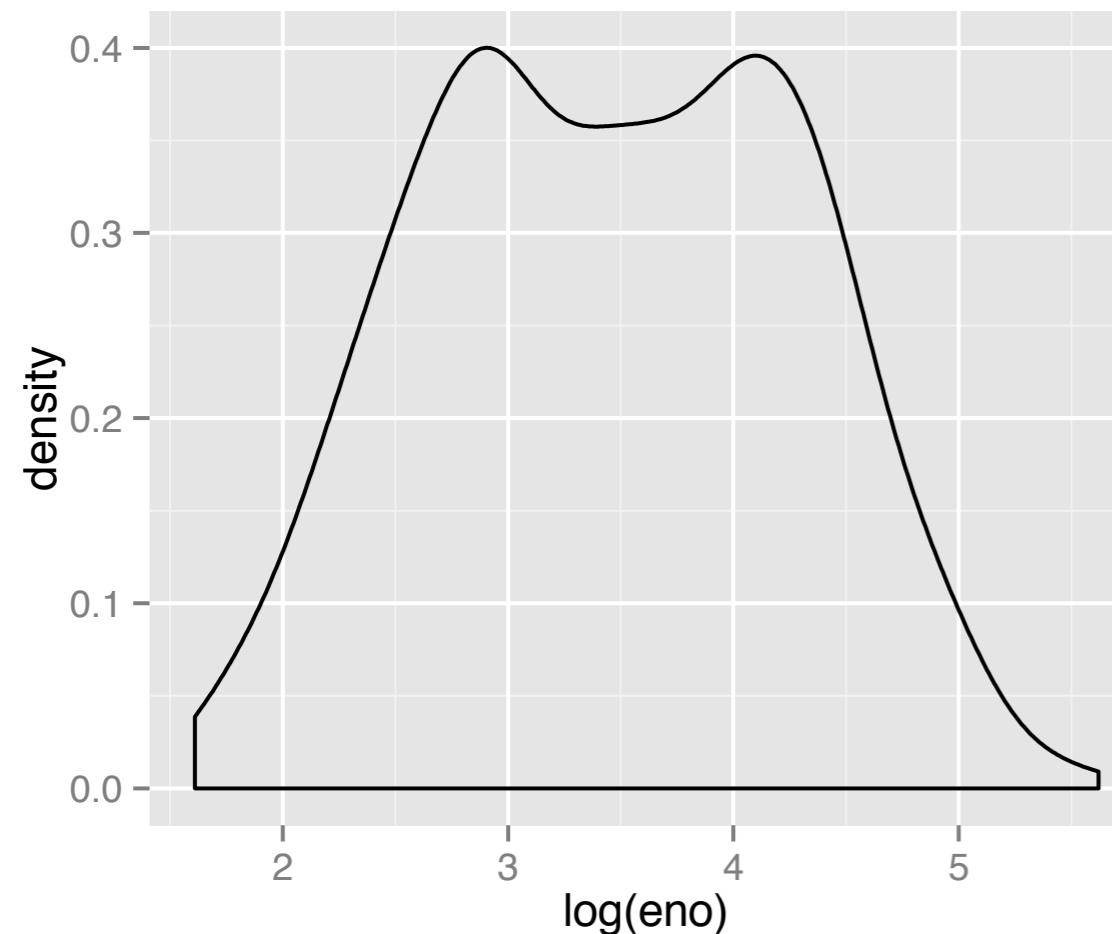
Histogram of eNO



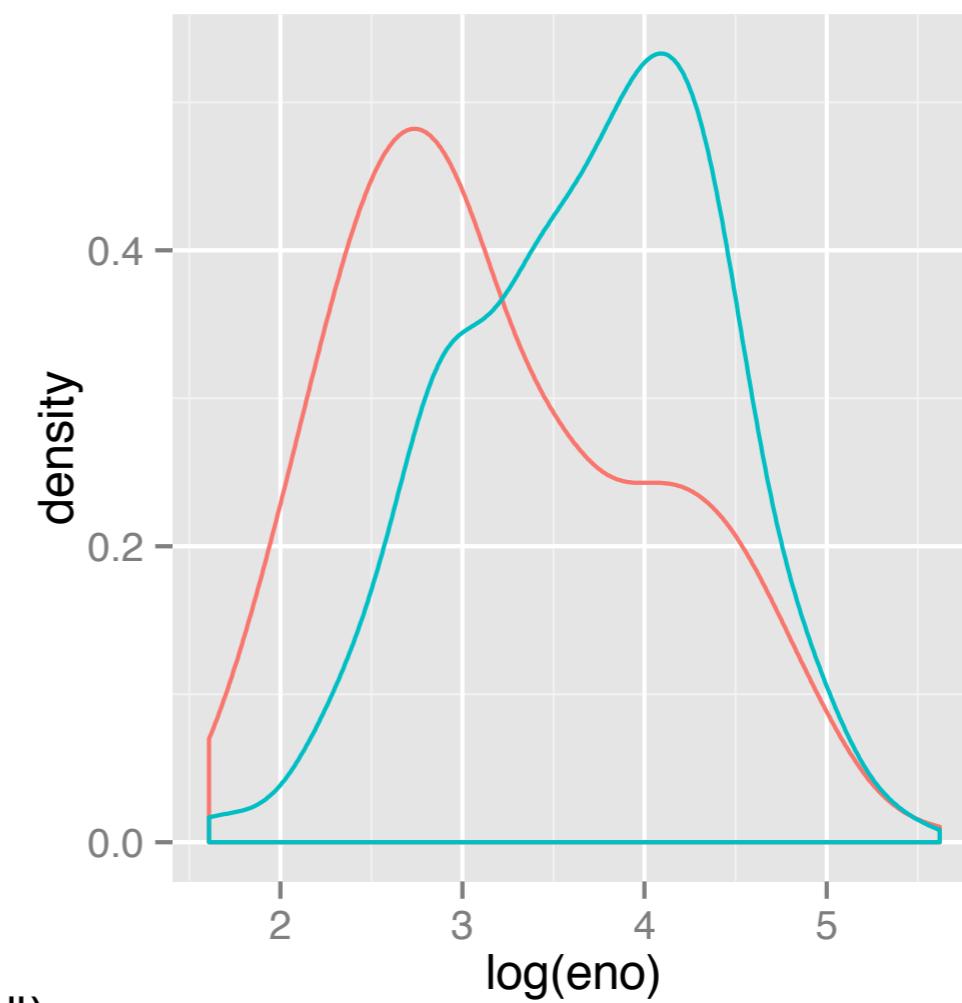
Histogram by Group



Density Smooth

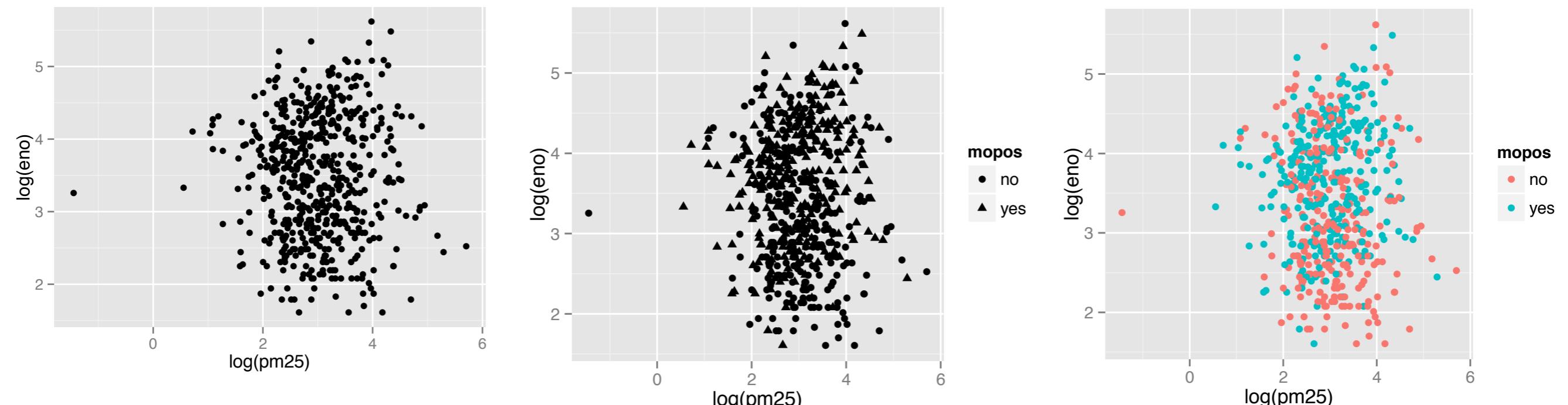


```
qplot(log(eno), data = maacs, geom = "density")
```



```
qplot(log(eno), data = maacs, geom = "density", color = r
```

Scatterplots: eNO vs. PM2.5

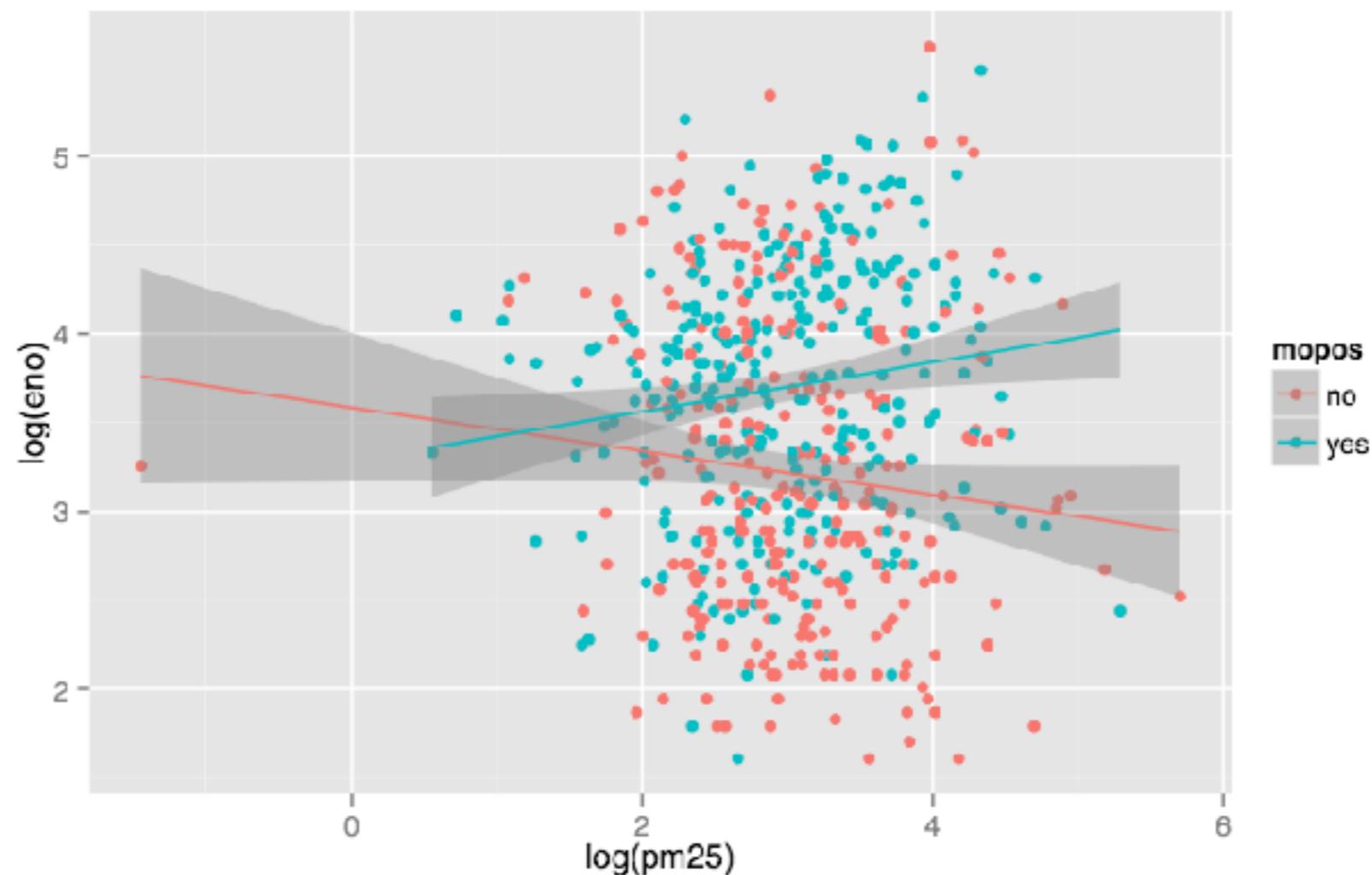


```
qplot(log(pm25), log(eno),  
      data = maacs)
```

```
qplot(log(pm25), log(eno), data  
      = maacs, shape = mopos)
```

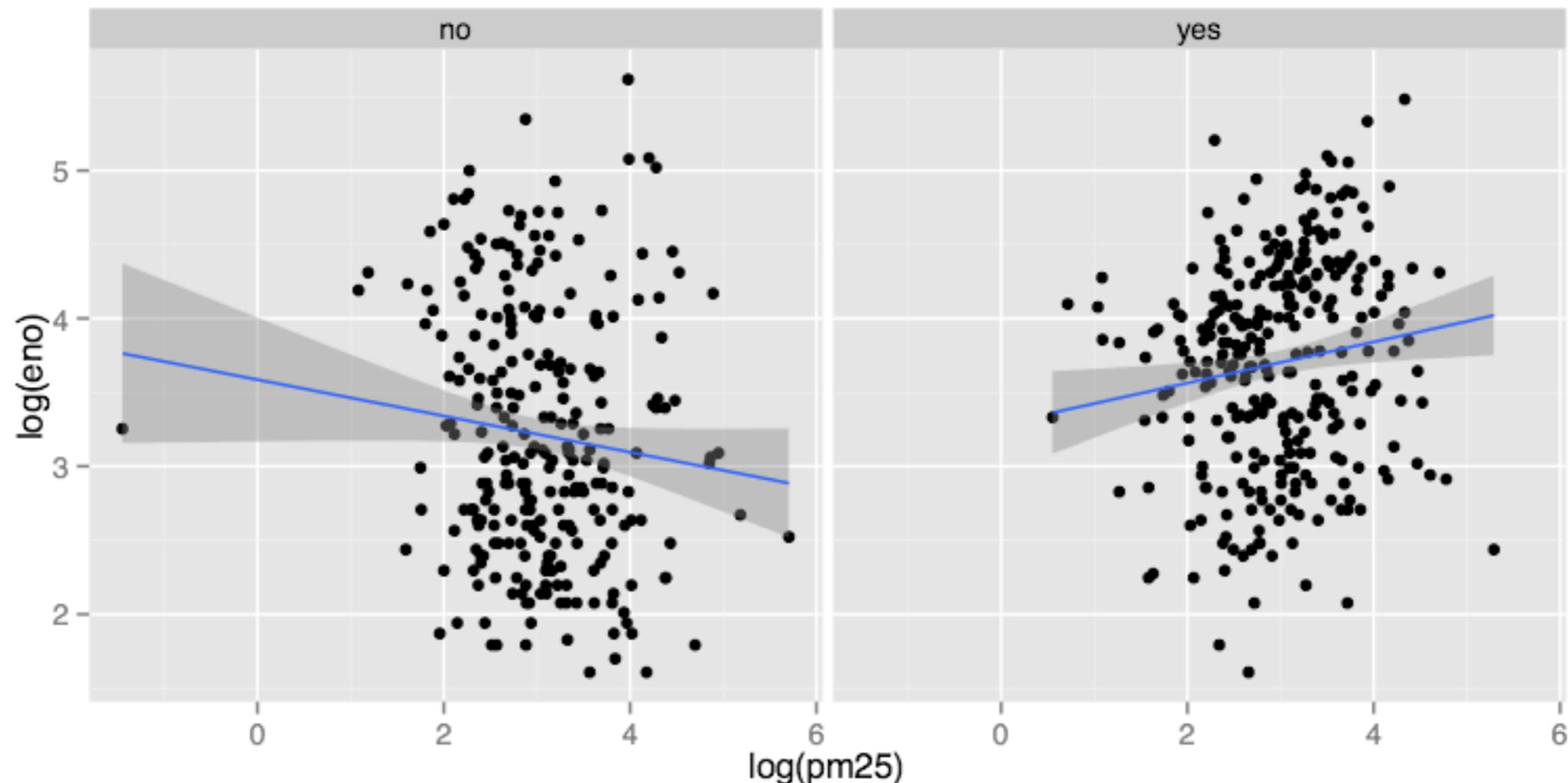
```
qplot(log(pm25), log(eno),  
      data = maacs, color = mopos)
```

Scatterplots: eNO vs. PM2.5



```
qplot(log(pm25), log(eno), data = maacs, color = mopos) + geom_smooth(method = "lm")
```

Scatterplots: eNO vs. PM2.5



```
qplot(log(pm25), log(eno), data = maacs, facets = . ~ mupos) + geom_smooth(method = "lm")
```

Summary of qplot()

- The qplot() function is the analog to plot() but with many built-in features
- Syntax somewhere in between base/lattice
- Produces very nice graphics, essentially publication ready (if you like the design)
- Difficult to go against the grain/customize (don't bother; use full ggplot2 power in that case)

Resources

- The ggplot2 book by Hadley Wickham
- The R Graphics Cookbook by Winston Chang (examples in base plots and in ggplot2)
- ggplot2 web site (<http://ggplot2.org>)
- ggplot2 mailing list (<http://goo.gl/OdW3uB>), primarily for developers

What is ggplot2?

- An implementation of the Grammar of Graphics by Leland Wilkinson
- Grammar of graphics represents and abstraction of graphics ideas/objects
- Think “verb”, “noun”, “adjective” for graphics
- Allows for a “theory” of graphics on which to build new graphics and graphics objects

Basic Components of a ggplot2 Plot

- A data frame
- aesthetic mappings: how data are mapped to color, size
- geoms: geometric objects like points, lines, shapes.
- facets: for conditional plots.
- stats: statistical transformations like binning, quantiles, smoothing.
- scales: what scale an aesthetic map uses (example: male = red, female = blue).
- coordinate system

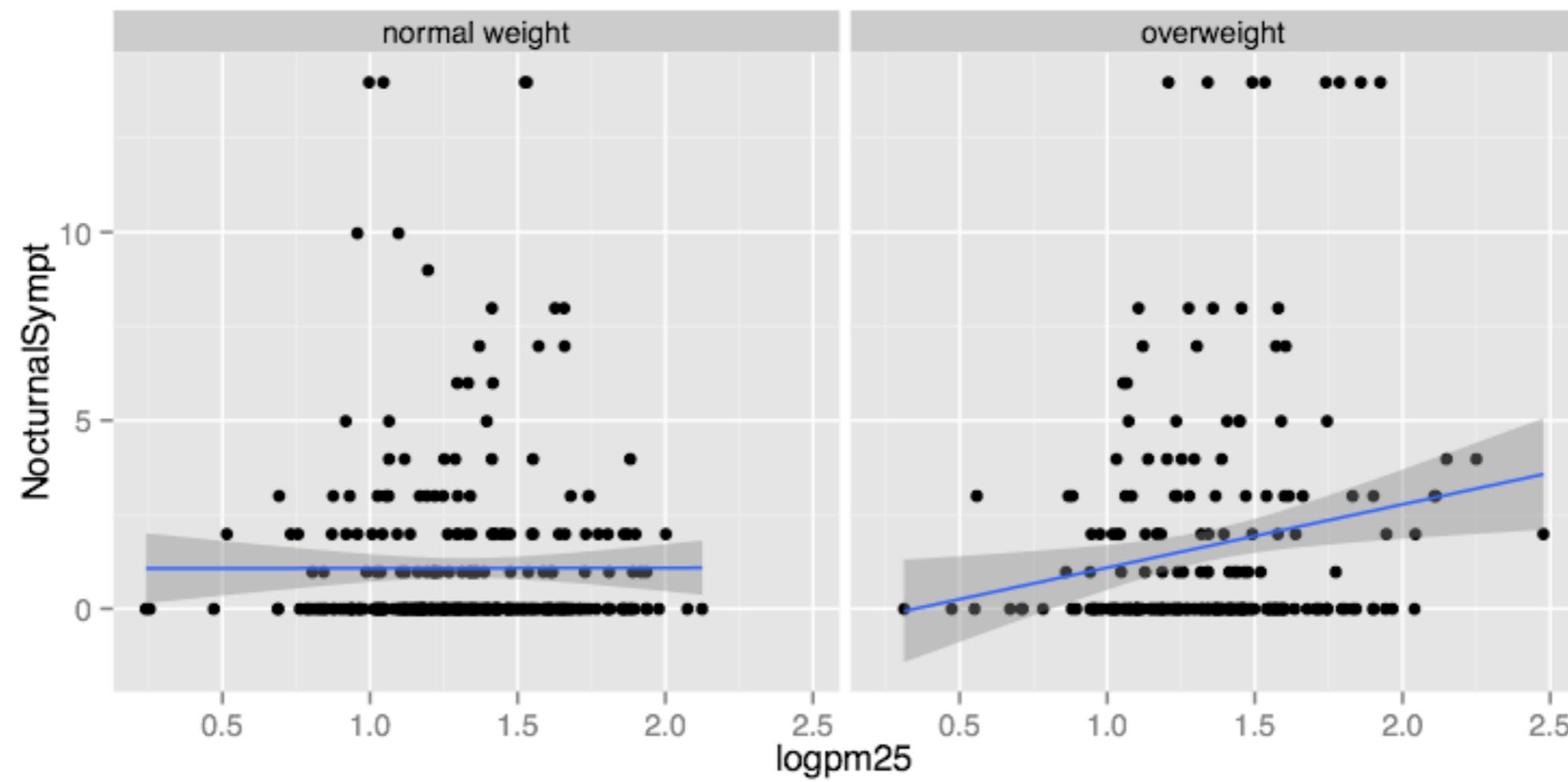
Building Plots with ggplot2

- When building plots in ggplot2 (rather than using qplot) the “artist’s palette” model may be the closest analogy
- Plots are built up in layers
 - Plot the data
 - Overlay a summary
 - Metadata and annotation

Example: BMI, PM2.5, Asthma

- Mouse Allergen and Asthma Cohort Study
- Baltimore children (age 5-17)
- Persistent asthma, exacerbation in past year
- Does BMI (normal vs. overweight) modify the relationship between PM2.5 and asthma symptoms?

Basic Plot



```
qplot(logpm25, NocturnalSympt, data = maacs, facets = . ~ bmicat, geom = c("point",  
"smooth"), method = "lm")
```

Building Up in Layers

```
> head(maacs)
  logpm25    bmicat NocturnalSympt
2 1.5361795 normal weight      1
3 1.5905409 normal weight      0
4 1.5217786 normal weight      0
5 1.4323277 normal weight      0
6 1.2762320  overweight       8
8 0.7139103  overweight       0
```

```
> g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
```

```
> summary(g)
data: logpm25, bmicat, NocturnalSympt [554x3]
mapping: x = logpm25, y = NocturnalSympt
faceting: facet_null()
```

Data Frame

Aesthetics

Initial call to
ggplot

Summary of
ggplot object

No Plot Yet!

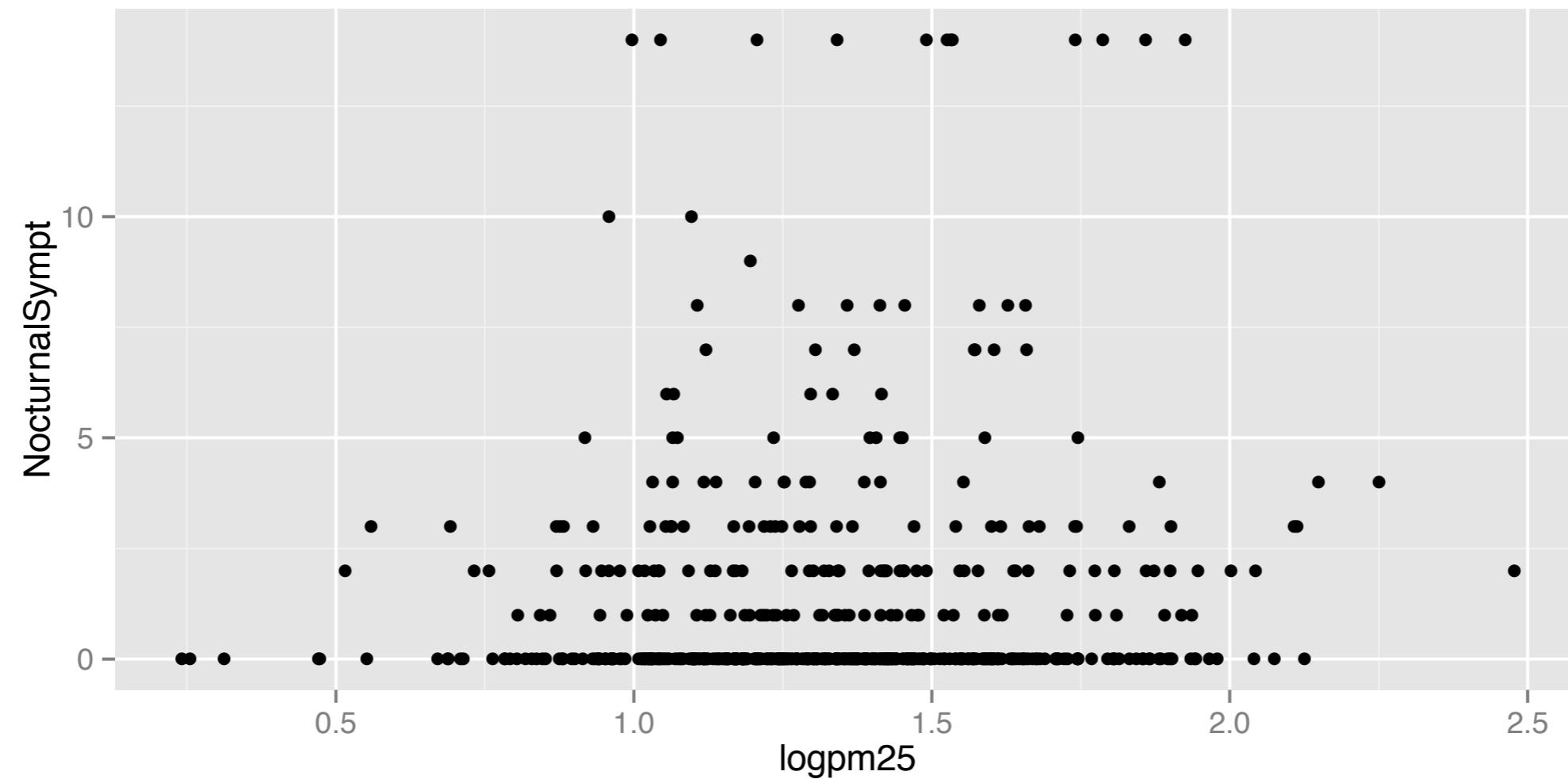
```
> g <- ggplot(maacs, aes(logpm25, NocturnalSympt))  
> print(g)  
Error: No layers in plot
```

```
> p <- g + geom_point()  
> print(p)  
  
> g + geom_point()
```

Explicitly save and print
ggplot object

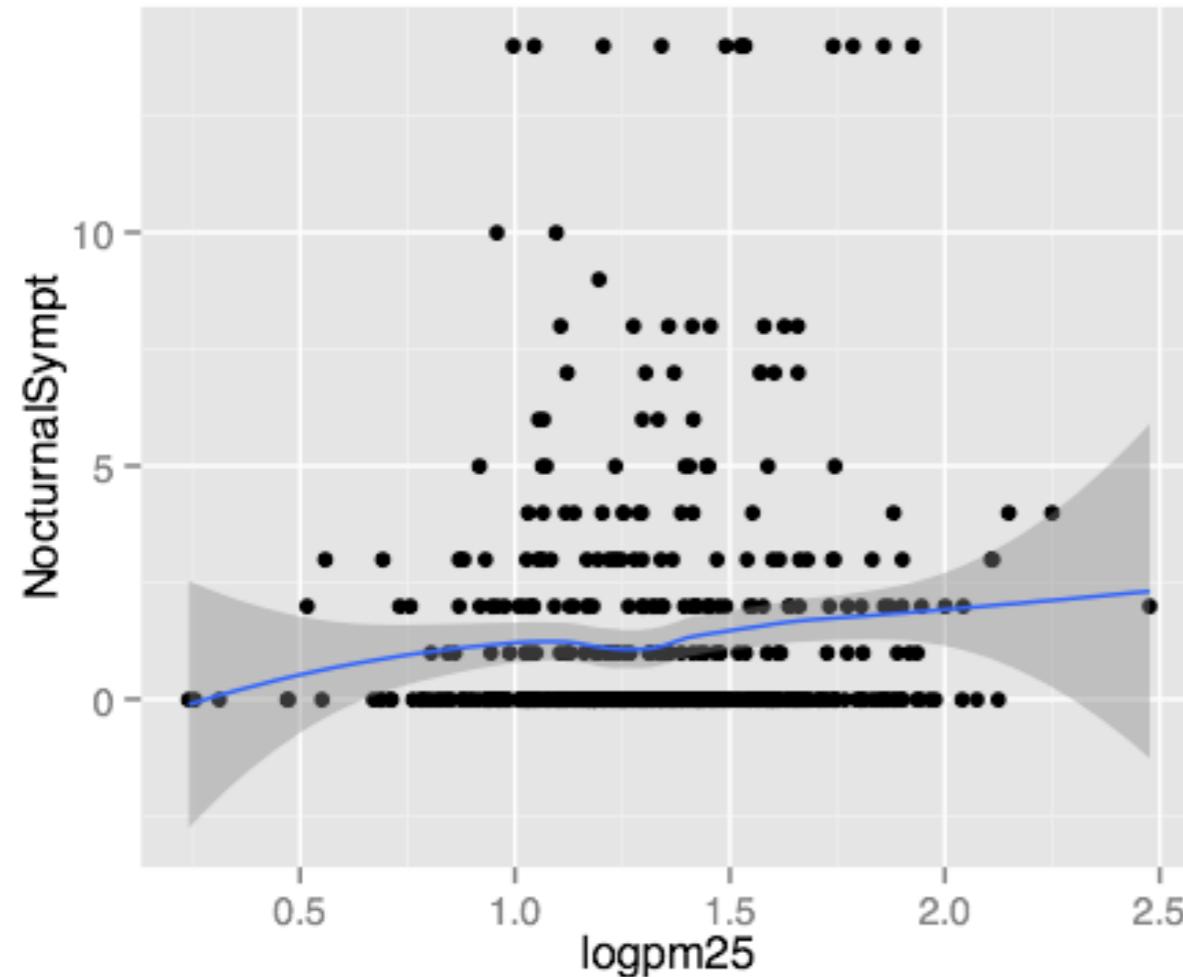
Auto-print plot object
without saving

First Plot with Point Layer

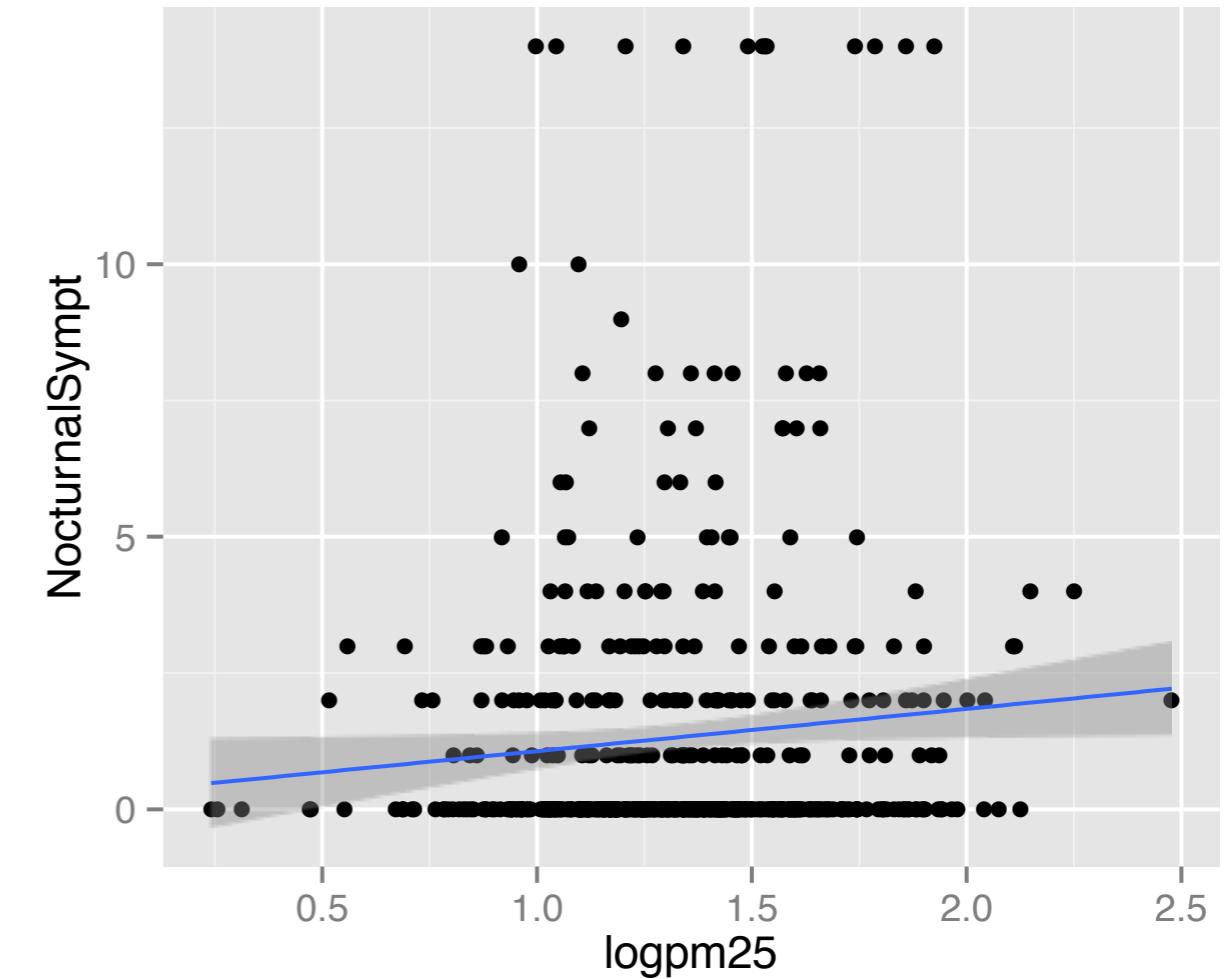


```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))  
g + geom_point()
```

Adding More Layers: Smooth

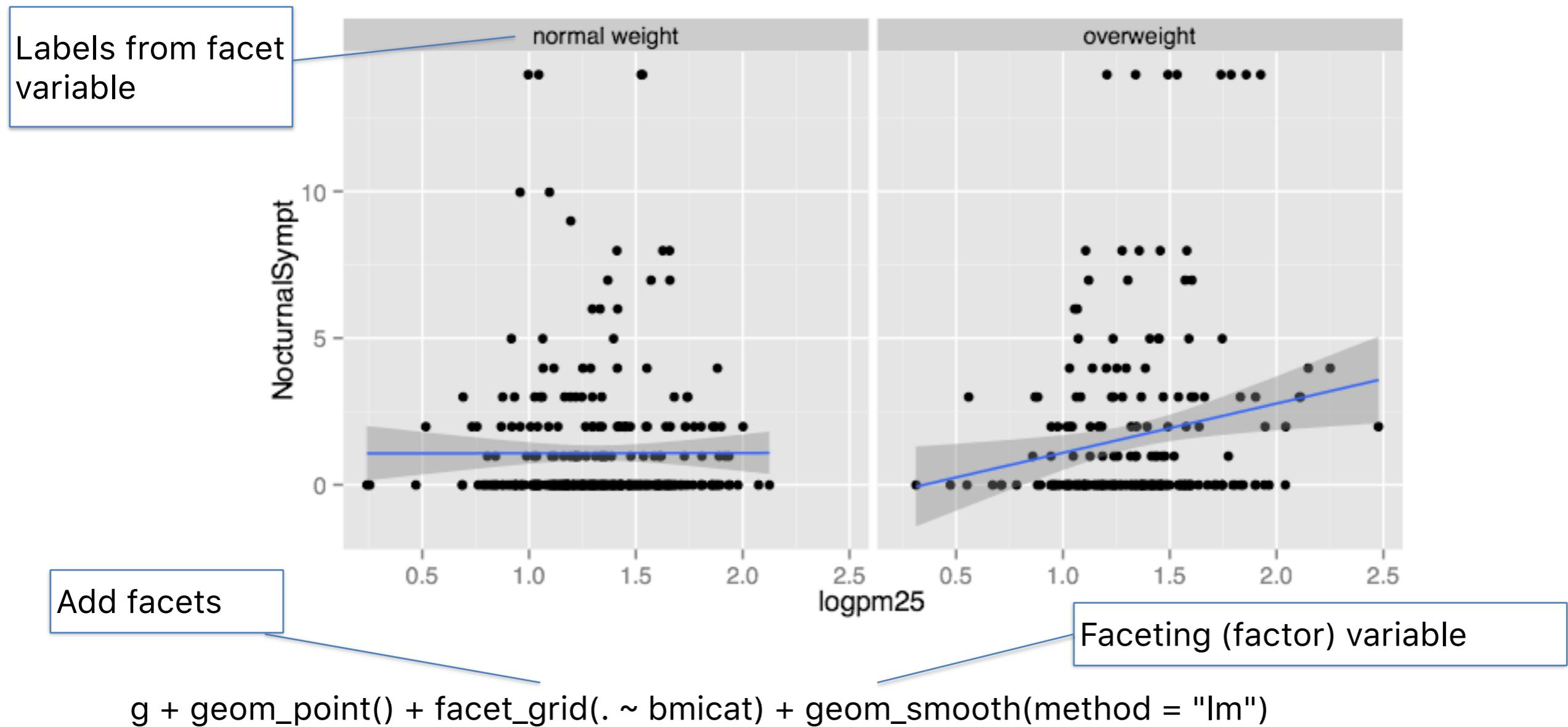


`g + geom_point() + geom_smooth()`



`g + geom_point() + geom_smooth(method = "lm")`

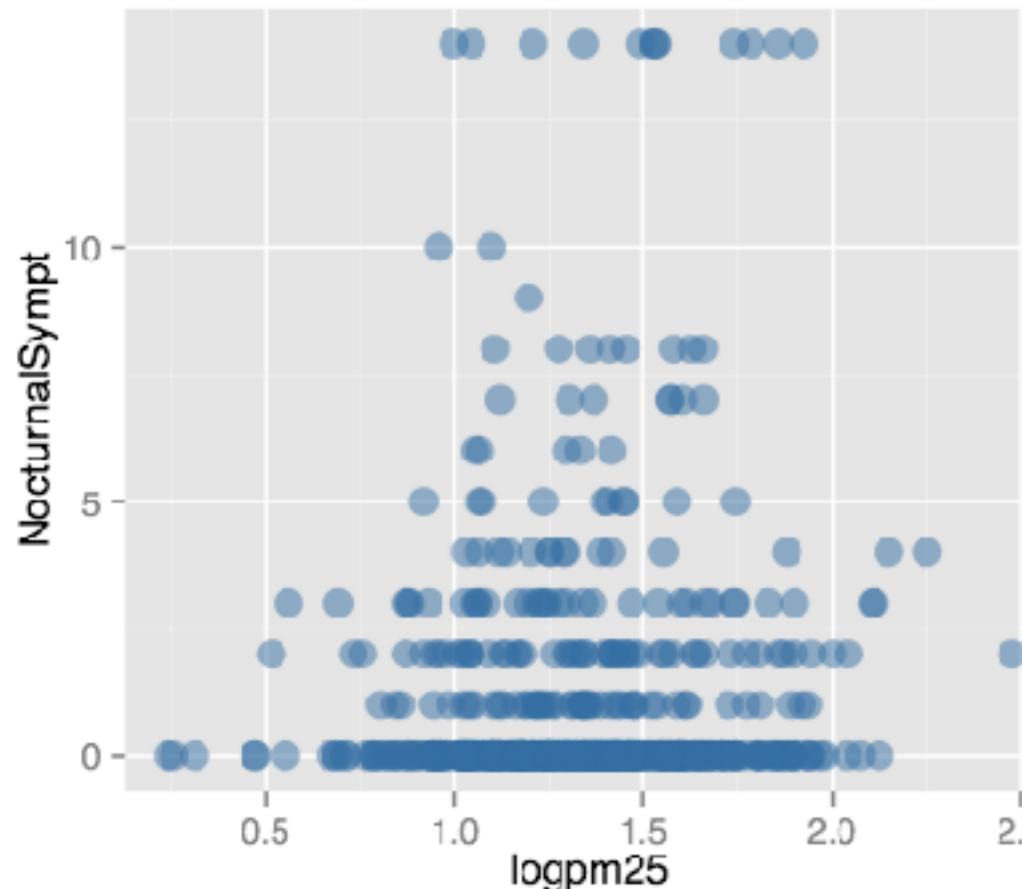
Adding More Layers: Facets



Annotation

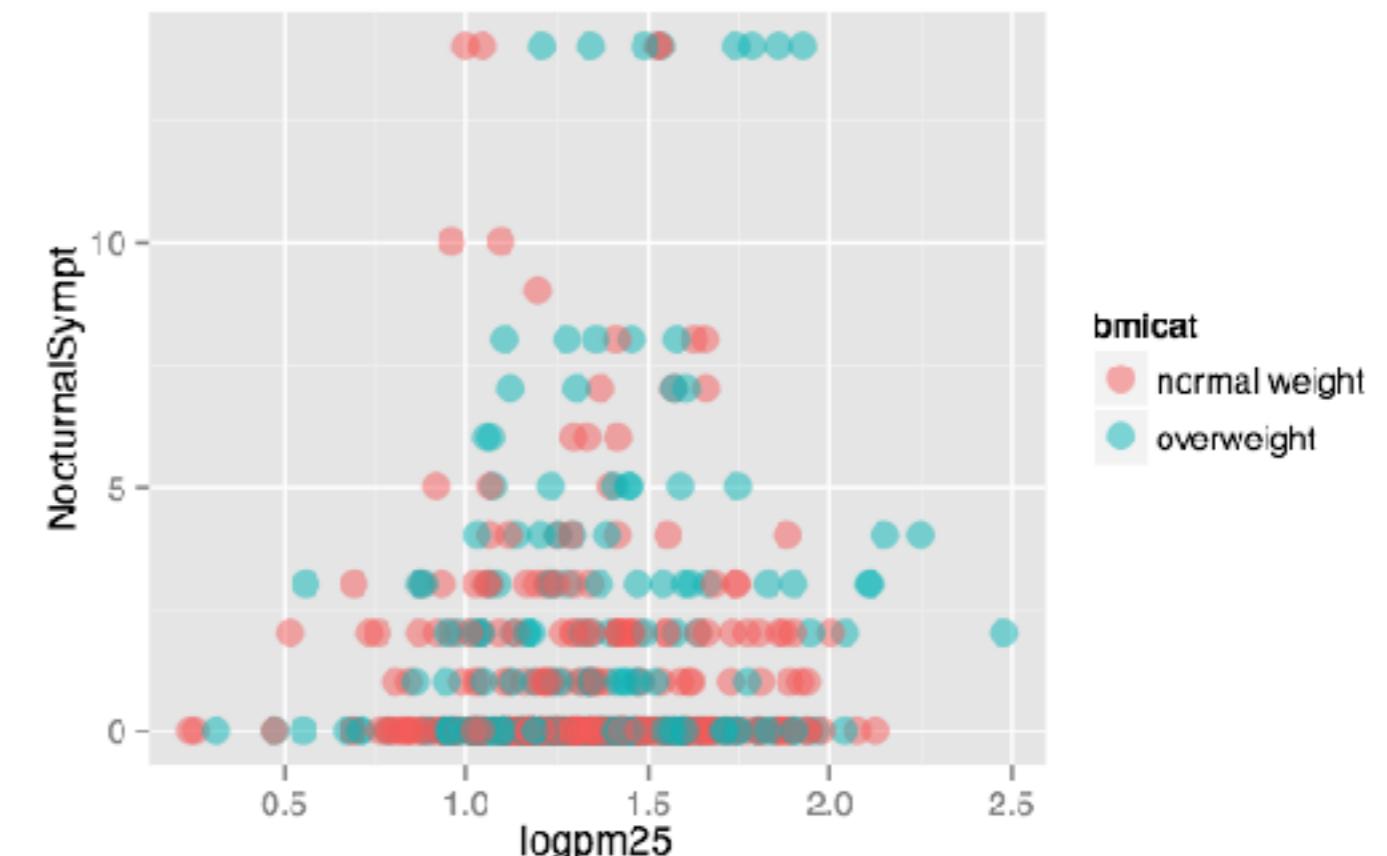
- Labels: `xlab()`, `ylab()`, `labs()`, `ggtitle()`
- Each of the “geom” functions has options to modify
- For things that only make sense globally, use `theme()`
 - Example: `theme(legend.position = "none")`
- Two standard appearance themes are included
 - `theme_gray()`: The default theme (gray background)
 - `theme_bw()`: More stark/plain

Modifying Aesthetics



```
g + geom_point(color = "steelblue", size = 4,  
alpha = 1/2)
```

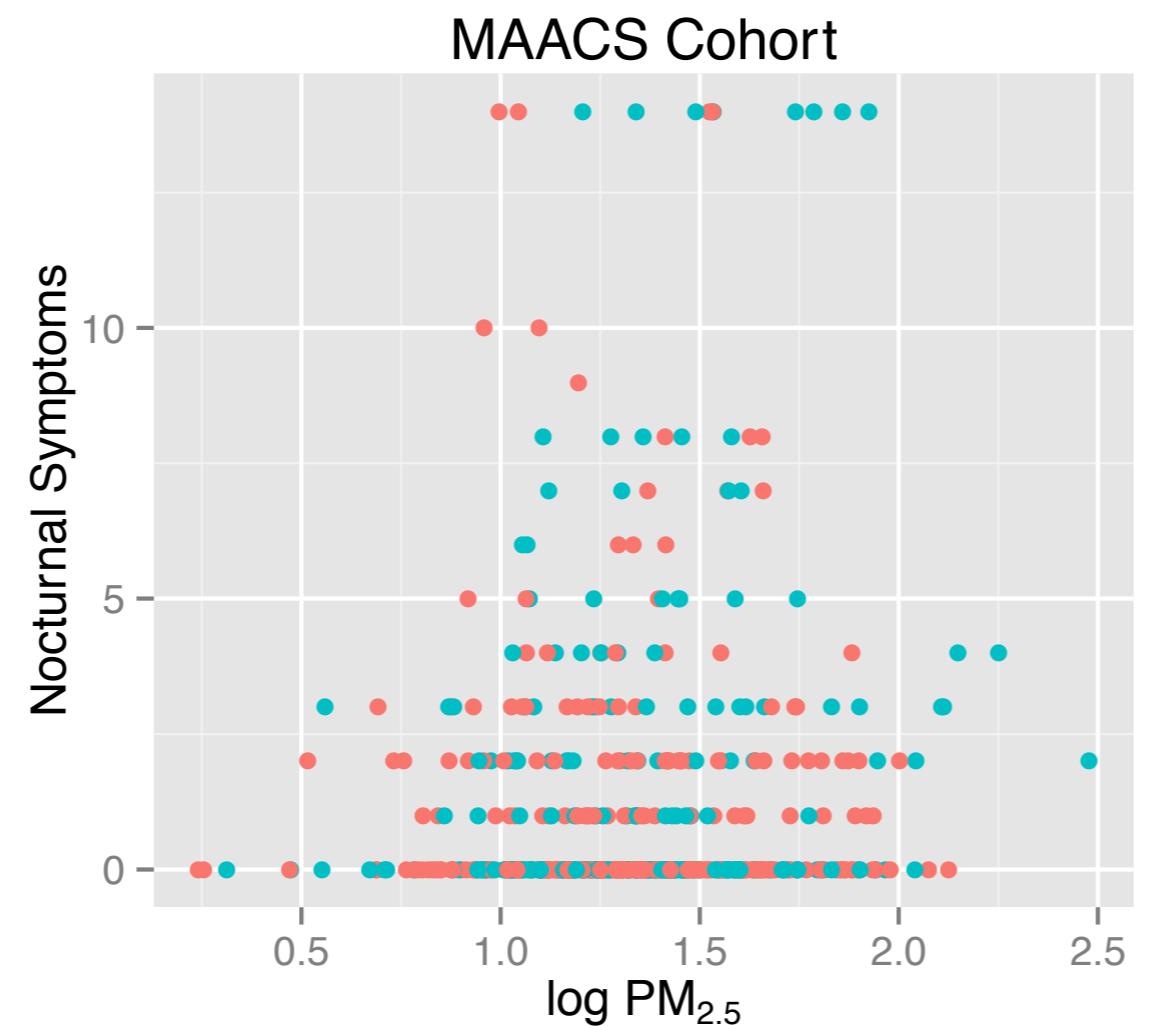
Constant values



```
g + geom_point(aes(color = bmicat), size = 4,  
alpha = 1/2)
```

Data variable

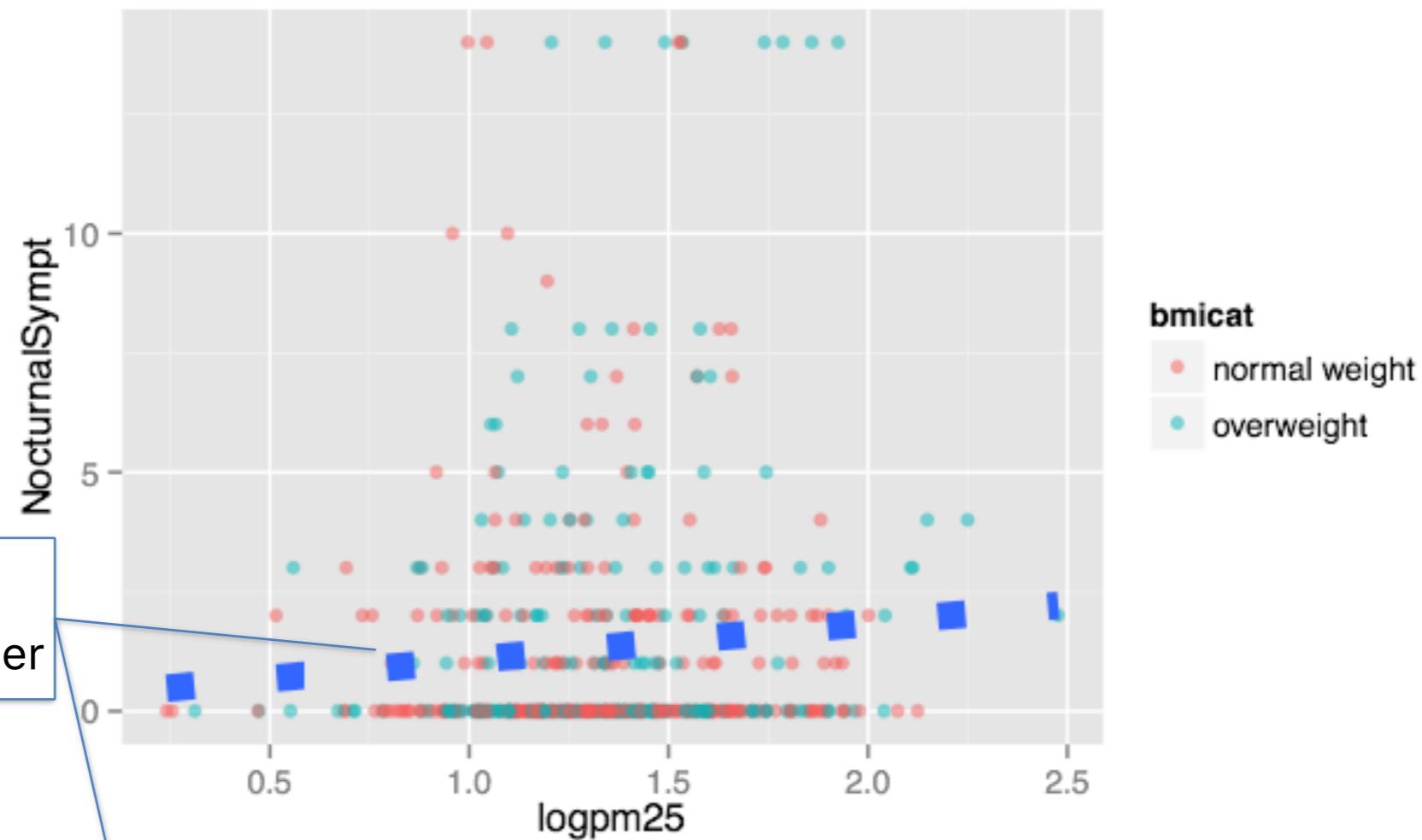
Modifying Labels



labs() function for
modifying titles and x-, y-
axis labels

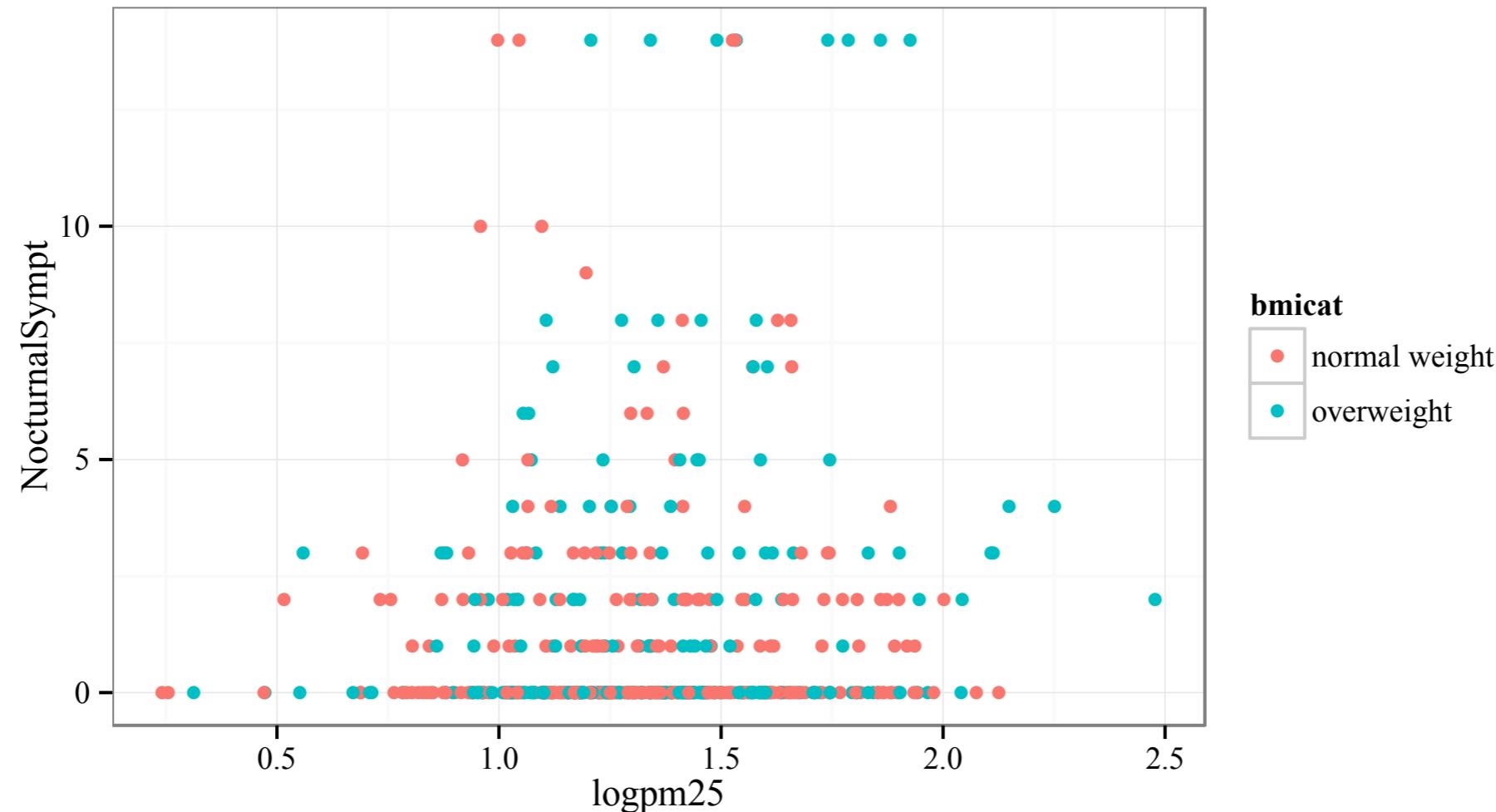
```
g + geom_point(aes(color = bmicat)) + labs(title = "MAACS Cohort") + labs(x = expression("log " * PM[2.5]), y = "Nocturnal Symptoms")
```

Customizing the Smooth



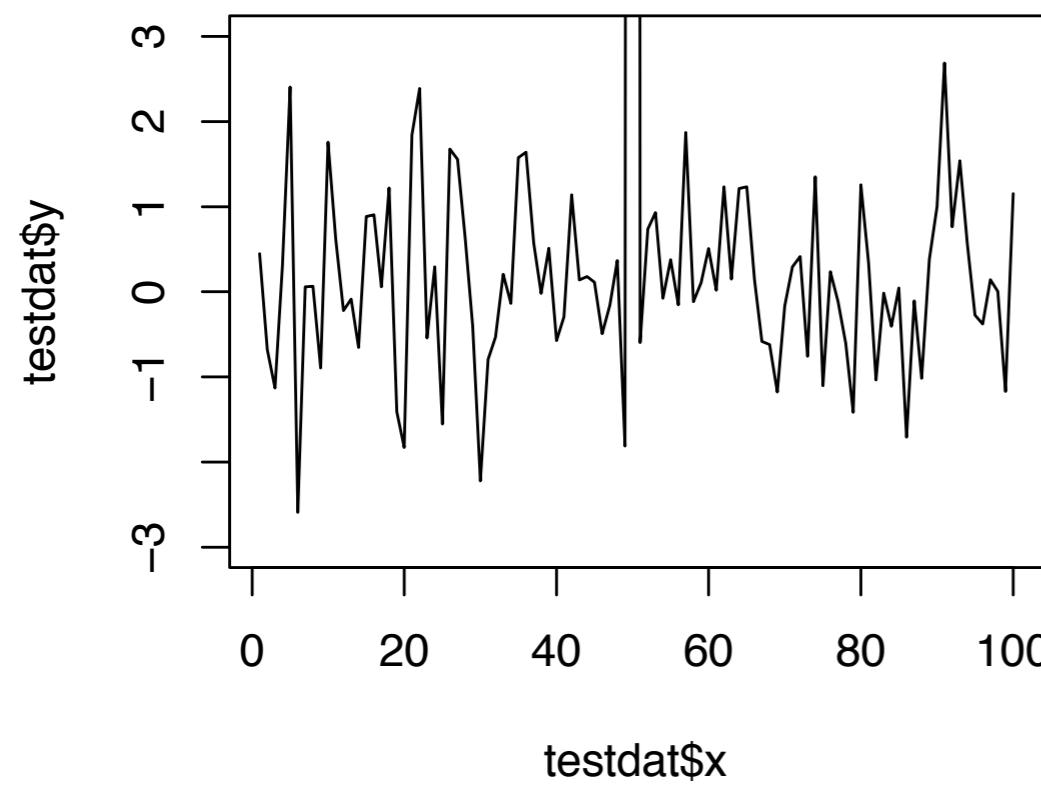
```
g + geom_point(aes(color = bmicat), size = 2, alpha = 1/2) + geom_smooth(size = 4,  
linetype = 3, method = "lm", se = FALSE)
```

Changing the Theme

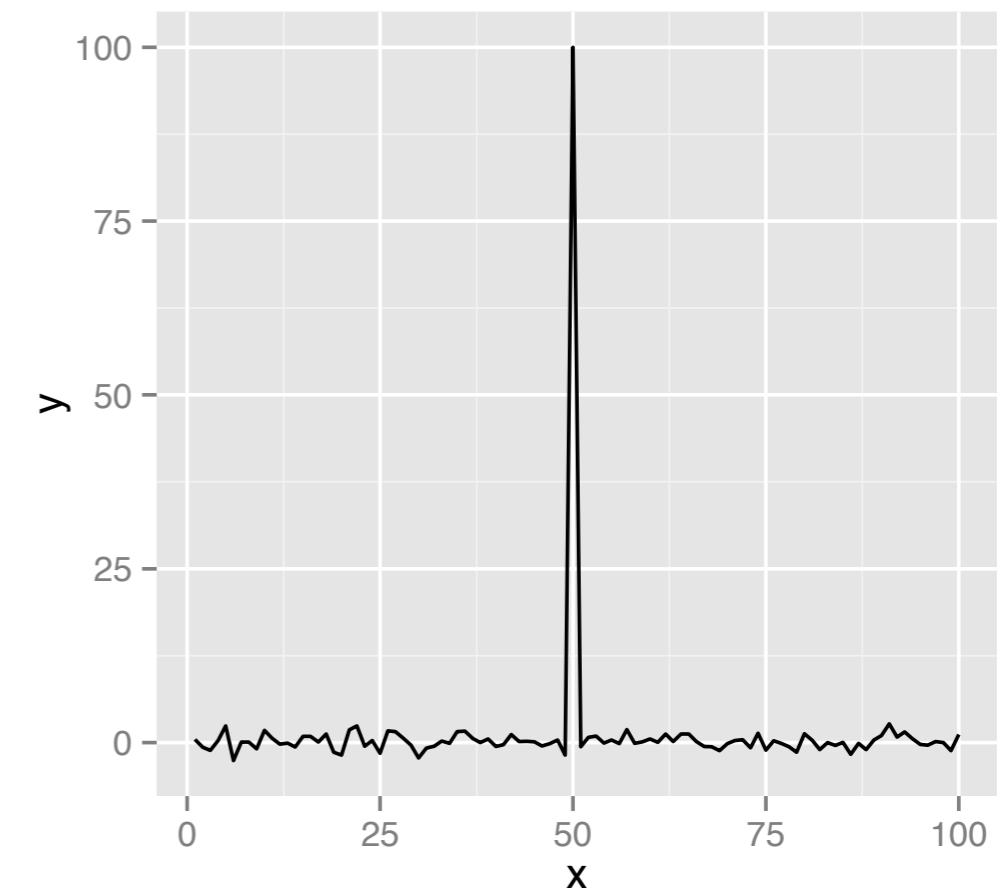


```
g + geom_point(aes(color = bmicat)) + theme_bw(base_family = "Times")
```

A Notes about Axis Limits



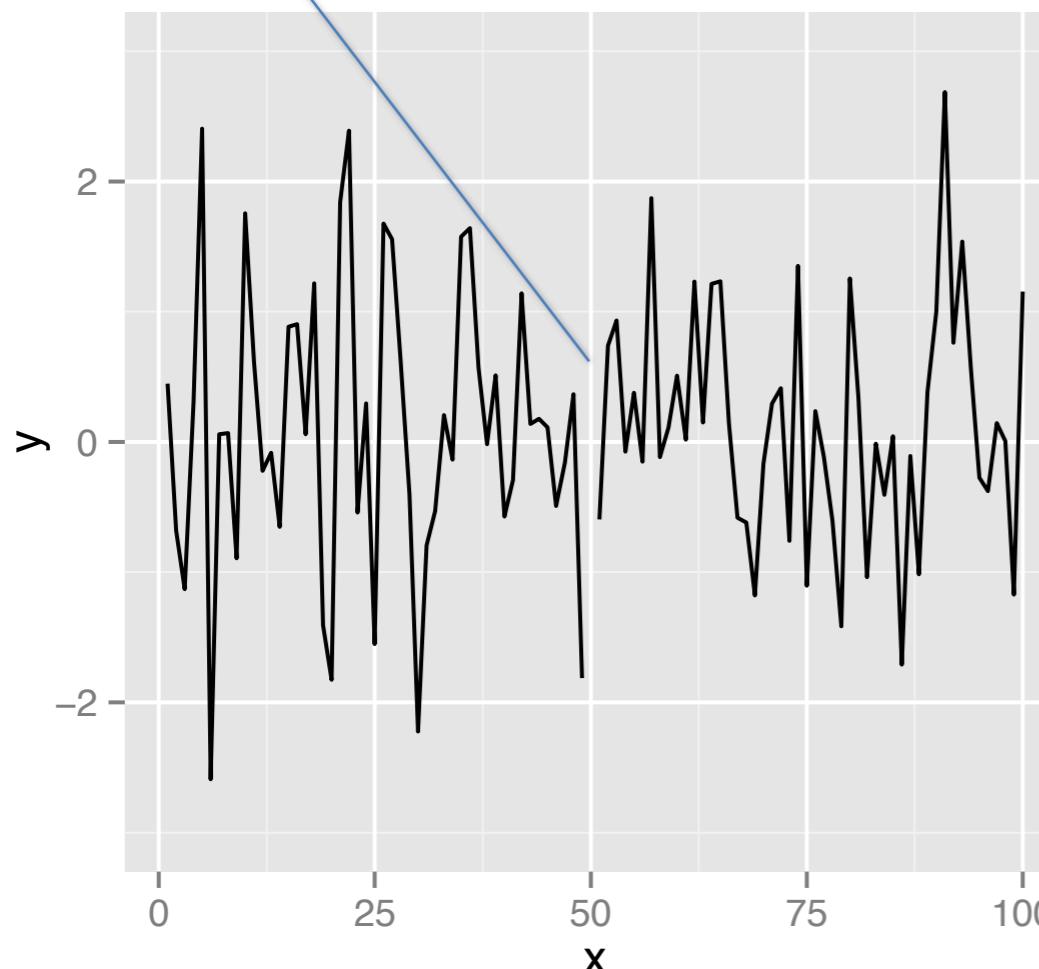
```
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50,2] <- 100 ## Outlier!
plot(testdat$x, testdat$y, type = "l", ylim = c(-3,3))
```



```
g <- ggplot(testdat, aes(x = x, y = y))
g + geom_line()
```

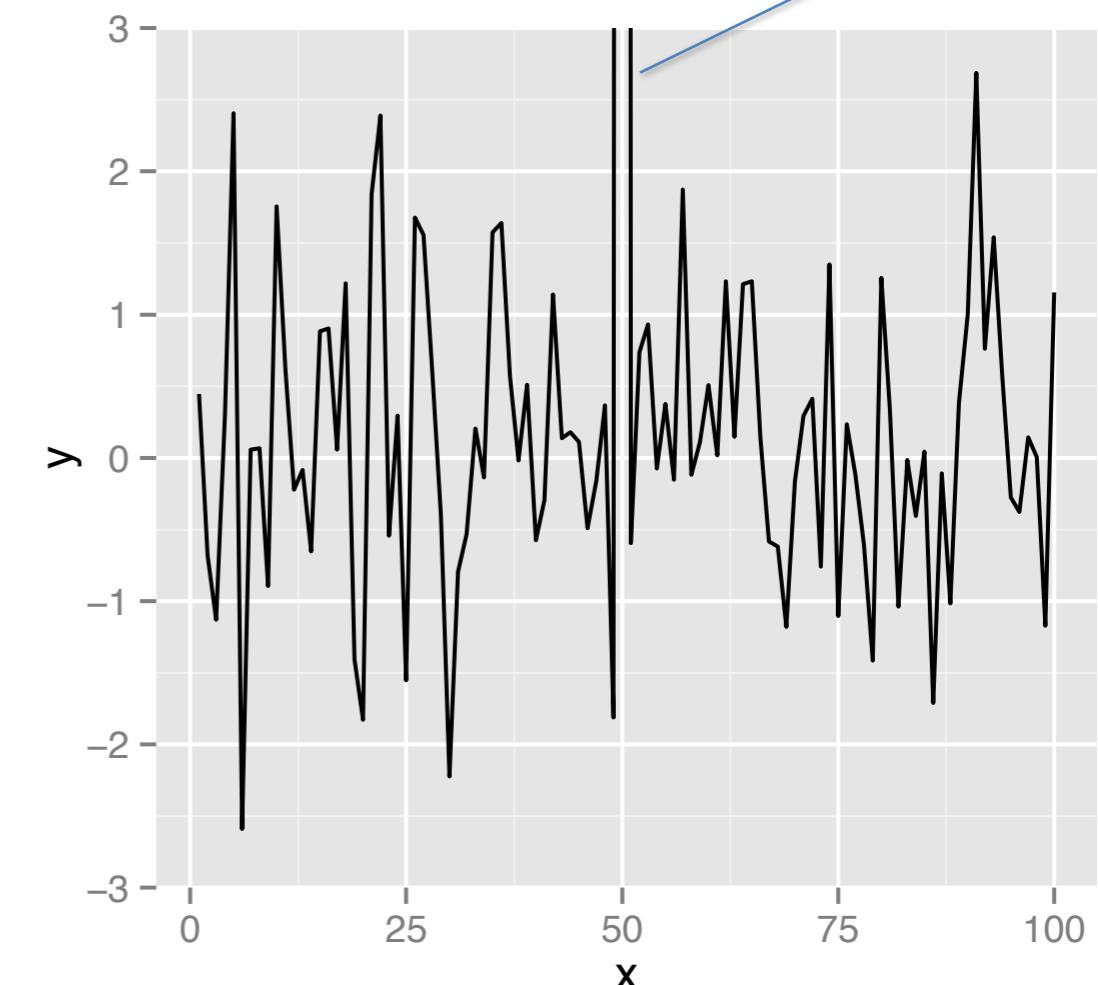
Axis Limits

Outlier missing



```
g + geom_line() + ylim(-3, 3)
```

Outlier included



```
g + geom_line() + coord_cartesian(ylim = c(-3, 3))
```

Workshop 7: Plot data

- Plotting the Cars93 data
- (a) Use **qplot** to create a scatterplot with **Price** on the **y-axis** and **EngineSize** on the **x-axis**.
- (b) Repeat part (a) using the **ggplot** function and **geom_point()** layer.
- (c) Repeat part (b), but this time specifying that the **color** mapping should depend on **Type** and the **shape** mapping should depend on **DriveTrain**.



Other Packages

To load data

RODBC, RMySQL, RPostgreSQL, RSQLite - If you'd like to read in data from a database, these packages are a good place to start. Choose the package that fits your type of database.

XLConnect, xlsx - These packages help you read and write Microsoft Excel files from R. You can also just export your spreadsheets from Excel as .csv's.

foreign - Want to read a SAS data set into R? Or an SPSS data set? Foreign provides functions that help you load data files from other programs into R.

R can handle plain text files – no package required. Just use the functions `read.csv`, `read.table`, and `read.fwf..`

xlsx

```
df <- read.xlsx("<name and extension of your file>",
                 sheetIndex = 1)
```

Note that it is necessary to add a sheet name or a sheet index to this function. In the example above, the first sheet of the Excel file was assigned. If you have a bigger data set, you might get better performance when using the `read.xlsx2()` function:

```
df <- read.xlsx2("<name and extension of your file>",
                  sheetIndex = 1,
                  startRow=2,
                  colIndex = 2)
```

```
write.xlsx(df,  
          "df.xlsx",  
          sheetName="Data Frame")
```

The function requires you first to specify what data frame you want to export. In the second argument, you specify the name of the file that you are outputting. If, however, you want to write the data frame to a file that already exists, you can execute the following command:

```
write.xlsx(df,  
          "<name and extension of your existing file>",  
          sheetName="Data Frame"  
          append=TRUE)
```

To manipulate data

dplyr - Essential shortcuts for subsetting, summarizing, rearranging, and joining together data sets. dplyr is our go to package for fast data manipulation.

tidyr - Tools for changing the layout of your data sets. Use the gather and spread functions to convert your data into the tidy format, the layout R likes best.

stringr - Easy to learn tools for regular expressions and character strings.

lubridate - Tools that make working with dates and times easier.

dplyr

```
install.packages("dplyr")  
library(dplyr)
```

dplyr verbs

	Description
<code>select()</code>	select columns
<code>filter()</code>	filter rows
<code>arrange()</code>	re-order or arrange rows
<code>mutate()</code>	create new columns
<code>summarise()</code>	summarise values
<code>group_by()</code>	allows for group operations in the “split-apply-combine” concept

select

Select a set of columns: the name and the sleep_total columns.

```
sleepData <- select(msleep, name, sleep_total)  
head(sleepData)
```

```
##                                     name sleep_total  
## 1                               Cheetah      12.1  
## 2             Owl monkey        17.0  
## 3       Mountain beaver      14.4  
## 4 Greater short-tailed shrew   14.9  
## 5                           Cow        4.0  
## 6      Three-toed sloth      14.4
```

Subtraction (-)

To select all the columns except a specific column, use the “-“ (subtraction) operator (also known as negative indexing)

```
head(select(msleep, -name))
```

```
##          genus vore      order conservation sleep_total sleep_rem
## 1    Acinonyx carni   Carnivora           lc      12.1       NA
## 2      Aotus omni    Primates        <NA>     17.0       1.8
## 3  Aplodontia herbi   Rodentia           nt      14.4       2.4
## 4    Blarina omni Soricomorpha           lc      14.9       2.3
## 5      Bos herbi Artiodactyla domesticated     4.0       0.7
## 6 Bradypus herbi      Pilosa        <NA>
##   sleep_cycle awake brainwt bodywt
## 1         NA   11.9      NA 50.000
## 2         NA    7.0 0.01550   0.480
## 3         NA    9.6      NA  1.350
## 4  0.1333333   9.1 0.00029   0.019
## 5  0.6666667  20.0 0.42300 600.000
## 6  0.7666667   9.6      NA   3.850
```

colon (:)

To select a range of columns by name, use the ":" (colon) operator

```
head(select(msleep, name:order))
```

```
##          name   genus  vore      order
## 1     Cheetah Acinonyx carni  Carnivora
## 2 Owl monkey     Aotus  omni   Primates
## 3 Mountain beaver Aplodontia herbi Rodentia
## 4 Greater short-tailed shrew    Blarina  omni Soricomorpha
## 5             Cow       Bos herbi Artiodactyla
## 6 Three-toed sloth   Bradypus herbi Pilosa
```

To select all columns that start with the character string “sl”, use the function `starts_with()`

```
head(select(msleep, starts_with("sl")))
```

```
##   sleep_total sleep_rem sleep_cycle
## 1      12.1        NA         NA
## 2      17.0        1.8         NA
## 3      14.4        2.4         NA
## 4      14.9        2.3  0.1333333
## 5       4.0        0.7  0.6666667
## 6      14.4        2.2  0.7666667
```

Some additional options to select columns based on a specific criteria include

1. `ends_with()` = Select columns that end with a character string
2. `contains()` = Select columns that contain a character string
3. `matches()` = Select columns that match a regular expression
4. `one_of()` = Select columns names that are from a group of names

filter

Filter the rows for mammals that sleep a total of more than 16 hours.

```
filter(msleep, sleep_total >= 16)
```

```
##          name      genus   vore      order conservation
## 1    Owl monkey     Aotus   omni Primates       <NA>
## 2 Long-nosed armadillo Dasypus carni Cingulata        lc
## 3 North American Opossum Didelphis   omni Didelphimorphia        lc
## 4    Big brown bat Eptesicus insecti Chiroptera        lc
## 5 Thick-tailed opossum Lutreolina carni Didelphimorphia        lc
## 6    Little brown bat      Myotis insecti Chiroptera       <NA>
## 7    Giant armadillo Priodontes insecti Cingulata        en
## 8 Arctic ground squirrel Spermophilus   herbi Rodentia        lc
##   sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1      17.0       1.8           NA     7.0  0.01550   0.480
## 2      17.4       3.1  0.3833333     6.6  0.01080   3.500
## 3      18.0       4.9  0.3333333     6.0  0.00630   1.700
## 4      19.7       3.9  0.1166667     4.3  0.00030   0.023
## 5      19.4       6.6           NA     4.6       NA   0.370
## 6      19.9       2.0  0.2000000     4.1  0.00025   0.010
## 7      18.1       6.1           NA     5.9  0.08100  60.000
## 8      16.6       NA           NA     7.4  0.00570   0.920
```

filter with and

Filter the rows for mammals that sleep a total of more than 16 hours *and* have a body weight of greater than 1 kilogram.

```
filter(msleep, sleep_total >= 16, bodywt >= 1)
```

```
##          name      genus     vore      order conservation
## 1 Long-nosed armadillo   Dasypus    carni  Cingulata        lc
## 2 North American Opossum Didelphis    omni Didelphimorphia    lc
## 3 Giant armadillo Priodontes insecti    carni  Cingulata        en
##   sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1       17.4       3.1     0.3833333   6.6   0.0108     3.5
## 2       18.0       4.9     0.3333333   6.0   0.0063     1.7
## 3       18.1       6.1        NA     5.9   0.0810    60.0
```

Pipe (%>%)

```
head(select(msleep, name, sleep_total))
```

```
##                                     name sleep_total
## 1                               Cheetah      12.1
## 2             Owl monkey       17.0
## 3     Mountain beaver       14.4
## 4 Greater short-tailed shrew   14.9
## 5                      Cow        4.0
## 6 Three-toed sloth       14.4
```

```
msleep %>%  
  select(name, sleep_total) %>%  
  head
```

```
##                                     name sleep_total  
## 1                               Cheetah      12.1  
## 2             Owl monkey        17.0  
## 3     Mountain beaver       14.4  
## 4 Greater short-tailed shrew    14.9  
## 5                           Cow        4.0  
## 6 Three-toed sloth       14.4
```

sqldf

```
library(sqldf)
```

```
setwd()
crashes <- read.csv("crashes.csv")
roads <- read.csv("roads.csv")
head(crashes)
```

```
##   Year      Road N_Crashes Volume
## 1 1991 Interstate 65      25 40000
## 2 1992 Interstate 65      37 41000
## 3 1993 Interstate 65      45 45000
## 4 1994 Interstate 65      46 45600
## 5 1995 Interstate 65      46 49000
## 6 1996 Interstate 65      59 51000
```

```
tail(crashes)
```

```
##   Year      Road N_Crashes Volume
## 105 2007 Interstate 275      32 21900
## 106 2008 Interstate 275      21 21850
## 107 2009 Interstate 275      25 22100
## 108 2010 Interstate 275      24 21500
## 109 2011 Interstate 275      23 20300
## 110 2012 Interstate 275      22 21200
```

```
print(roads)
```

```
##          Road      District Length
## 1 Interstate 65     Greenfield     262
## 2 Interstate 70     Vincennes    156
## 3          US-36 Crawfordsville 139
## 4          US-40     Greenfield   150
## 5          US-52 Crawfordsville 172
```

Outer join

```
join_string <- "select
  crashes.*
, roads.District
, roads.Length
from crashes
  left join roads
    on crashes.Road = roads.Road"
```

```
crashes_join_roads <- sqldf(join_string,stringsAsFactors = FALSE)
```

```
## Loading required package: tcltk
```

```
head(crashes_join_roads)
```

```
##   Year      Road N_Crashes Volume District Length
## 1 1991 Interstate 65        25 40000 Greenfield  262
## 2 1992 Interstate 65        37 41000 Greenfield  262
## 3 1993 Interstate 65        45 45000 Greenfield  262
## 4 1994 Interstate 65        46 45600 Greenfield  262
## 5 1995 Interstate 65        46 49000 Greenfield  262
## 6 1996 Interstate 65        59 51000 Greenfield  262
```

Inner join

By using an inner join, only matching rows will be kept.

```
join_string2 <- "select
  crashes.*
, roads.District
, roads.Length
from crashes
inner join roads
on crashes.Road = roads.Road"
```

```
crashes_join_roads2 <- sqldf(join_string2, stringsAsFactors = FALSE)
head(crashes_join_roads2)
```

##	Year	Road	N_Crashes	Volume	District	Length
## 1	1991	Interstate 65	25	40000	Greenfield	262
## 2	1992	Interstate 65	37	41000	Greenfield	262
## 3	1993	Interstate 65	45	45000	Greenfield	262
## 4	1994	Interstate 65	46	45600	Greenfield	262
## 5	1995	Interstate 65	46	49000	Greenfield	262
## 6	1996	Interstate 65	59	51000	Greenfield	262

```
tail(crashes_join_roads2)
```

	##	Year	Road	N_Crashes	Volume	District	Length
##	83	2007	US-36	49	24000	Crawfordsville	139
##	84	2008	US-36	52	24500	Crawfordsville	139
##	85	2009	US-36	55	24700	Crawfordsville	139
##	86	2010	US-36	35	23000	Crawfordsville	139
##	87	2011	US-36	33	21000	Crawfordsville	139
##	88	2012	US-36	31	20500	Crawfordsville	139

Where

```
join_string2 <- "select
  crashes.*
, roads.District
, roads.Length
  from crashes
    inner join roads
      on crashes.Road = roads.Road
    where crashes.Road = 'US-40'"
crashes_join_roads4 <- sqldf(join_string2,stringsAsFactors = FALSE)
head(crashes_join_roads4)
```

```
##   Year Road N_Crashes Volume District Length
## 1 1991 US-40        46 21000 Greenfield    150
## 2 1992 US-40       101 21500 Greenfield    150
## 3 1993 US-40        76 23000 Greenfield    150
## 4 1994 US-40        72 21000 Greenfield    150
## 5 1995 US-40        75 24000 Greenfield    150
## 6 1996 US-40       136 23500 Greenfield    150
```

```
tail(crashes_join_roads4)
```

```
##   Year Road N_Crashes Volume District Length
## 17 2007 US-40        45 59500 Greenfield    150
## 18 2008 US-40        23 61000 Greenfield    150
## 19 2009 US-40        67 65000 Greenfield    150
## 20 2010 US-40       102 67000 Greenfield    150
## 21 2011 US-40        87 67500 Greenfield    150
## 22 2012 US-40        32 67500 Greenfield    150
```

To visualize data

ggplot2 - R's famous package for making beautiful graphics. ggplot2 lets you use the grammar of graphics to build layered, customizable plots.

ggvis - Interactive, web based graphics built with the grammar of graphics.

rgl - Interactive 3D visualizations with R



htmlwidgets - A fast way to build interactive (javascript based) visualizations with R. Packages that implement htmlwidgets include:

leaflet (maps)

dygraphs (time series)

DT (tables)

diagrammeR (diagrams)

network3D (network graphs)

threeJS (3D scatterplots and globes).



googleVis - Let's you use Google Chart tools to visualize data in R.

Google Chart tools used to be called Gapminder, the graphing software

Hans Rosling made famous in his TED talk.

To model data

car - car's Anova function is popular for making type II and type III Anova tables.

mgcv - Generalized Additive Models

lme4/nlme - Linear and Non-linear mixed effects models

randomForest - Random forest methods from machine learning

multcomp - Tools for multiple comparison testing

vcd - Visualization tools and tests for categorical data

glmnet - Lasso and elastic-net regression methods with cross validation

survival - Tools for survival analysis

To report results

shiny - Easily make interactive, web apps with R. A perfect way to explore data and share findings with non-programmers.

R Markdown - The perfect workflow for reproducible reporting. Write R code in your markdown reports. When you run render, R Markdown will replace the code with its results and then export your report as an HTML, pdf, or MS Word document, or a HTML or pdf slideshow. The result? Automated reporting. R Markdown is integrated straight into RStudio.

xtable - The xtable function takes an R object (like a data frame) and

Introduction to Shiny

- Open Sourced by RStudio November 2012
- Default widgets and settings make it easy to generate apps
- Don't need to know HTML, CSS and javascript to get started
- Twitter Bootstrap for default UI - looks good
- Web sockets for communication between client and server
- Reactive Programming model
- Works on Windows, Mac, Linux

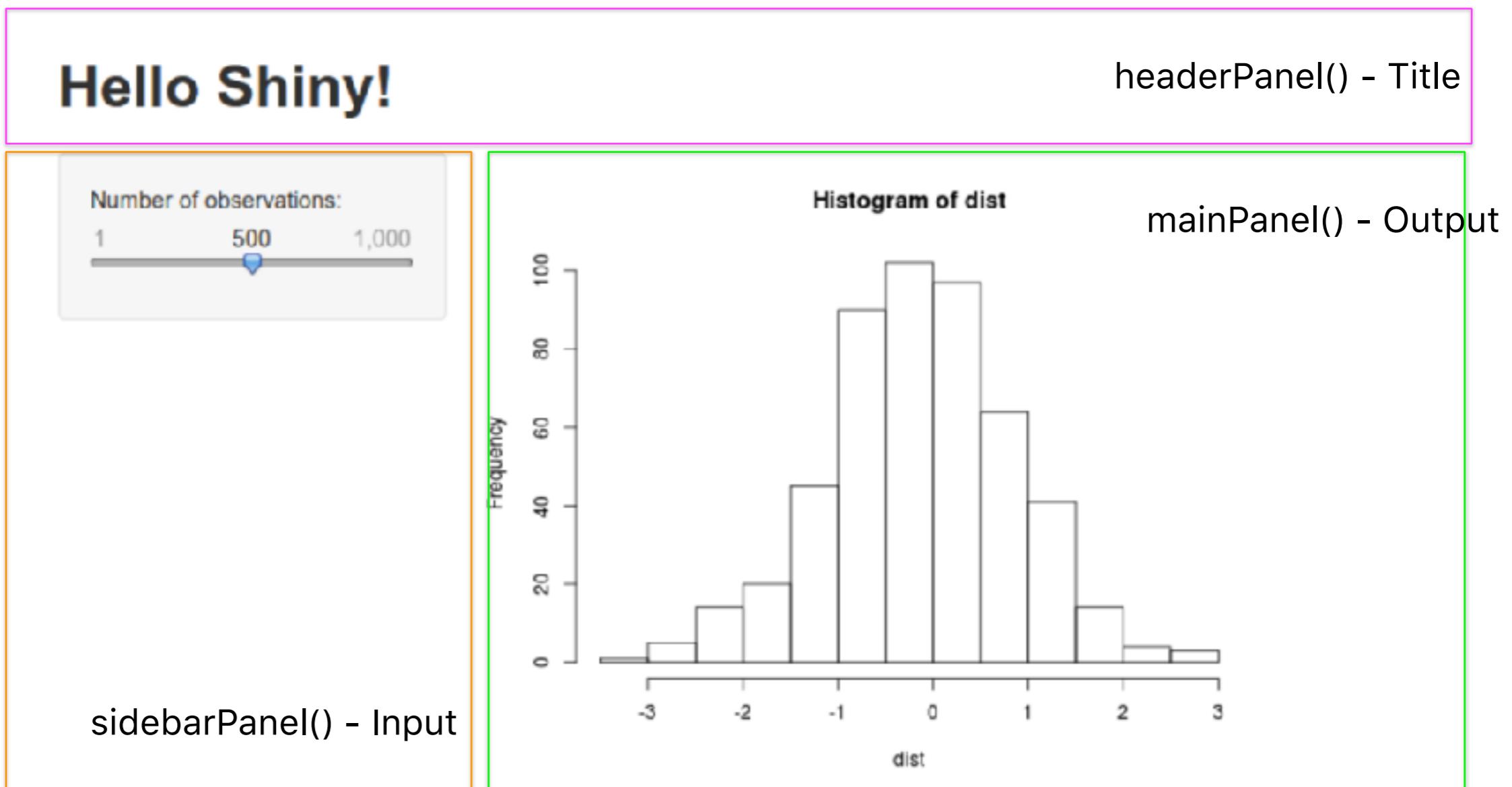
Introduction to Shiny

Ready to use shiny

- Install R from CRAN
- Useful to have Chrome, Firefox, Safari...
- Install Shiny using R command: `install.packages("shiny")`

Simple Example

- library(shiny)
- runExample("01_hello")



ui.R::: Controls the look of the App

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

server.R : Specifies what R is doing

```
library(shiny)

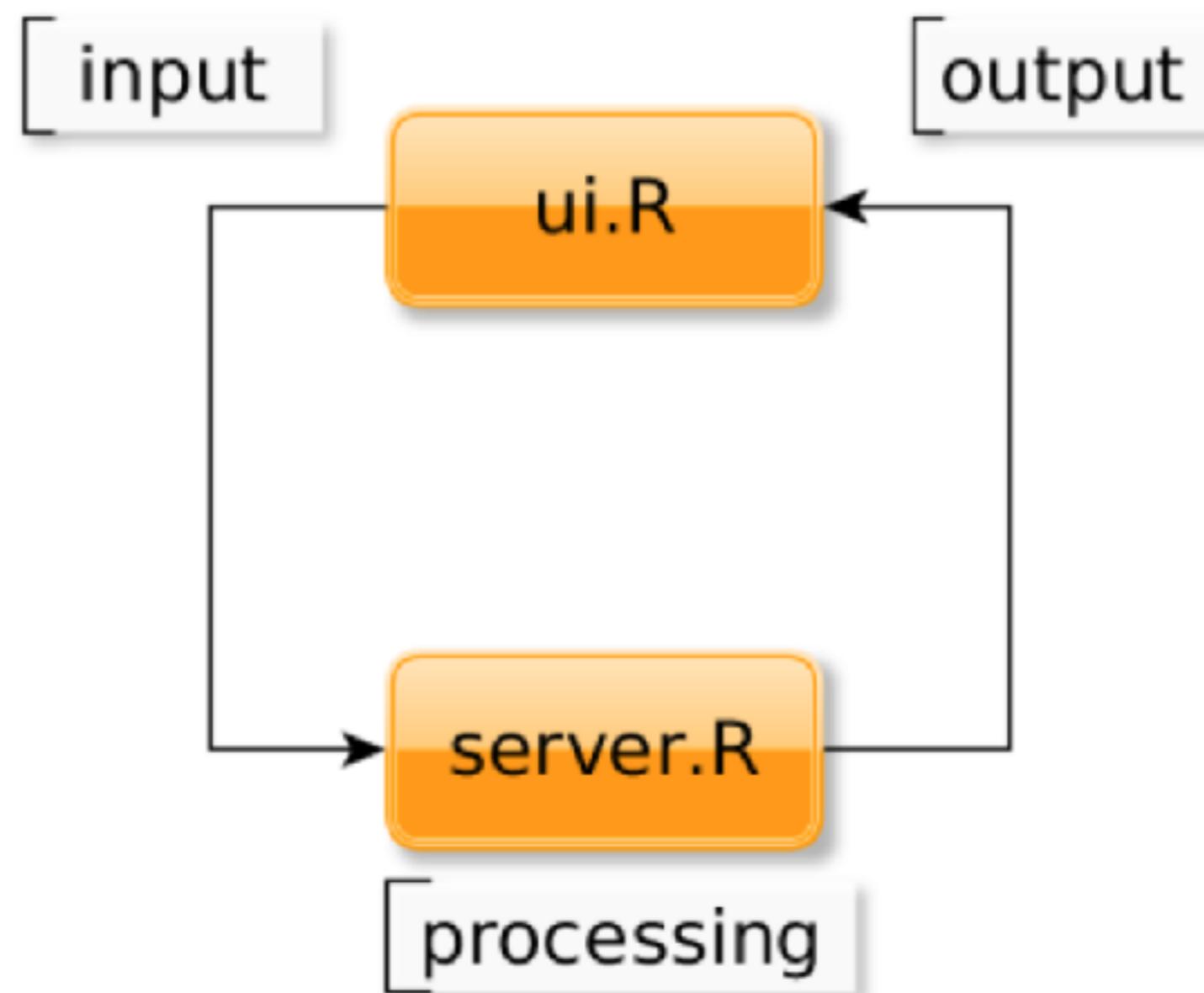
# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  # 1) It is "reactive" and therefore should be automatically
  #    re-executed when inputs change
  # 2) Its output type is a plot

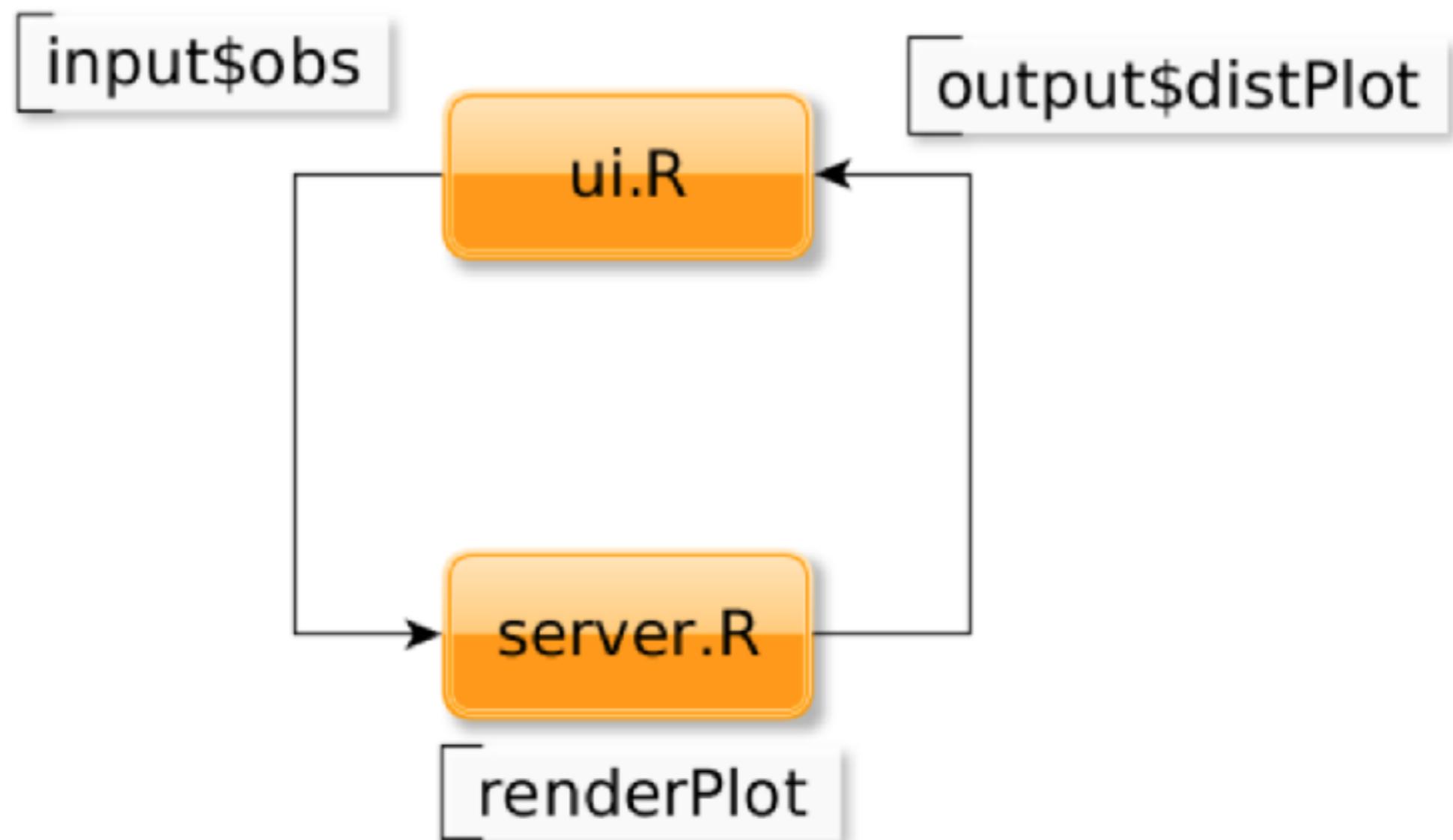
  output$distPlot <- renderPlot({
    x      <- faithful[, 2] # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Relationship of ui.R and server.R



Relationship of ui.R and server.R



For Spatial data

sp, maptools - Tools for loading and using spatial data including shapefiles.

maps - Easy to use map polygons for plots.

ggmap - Download street maps straight from Google maps and use them as a background in your ggplots.

To write your own R packages

devtools - An essential suite of tools for turning your code into an R package.

testthat - testthat provides an easy way to write unit tests for your code projects.

roxygen2 - A quick way to document your R packages. roxygen2 turns inline code comments into documentation pages and builds a package namespace.



Build R Package

Why write an R package?

- To keep track of the miscellaneous R functions that you write and reuse.
- To distribute the data and software that accompany a paper.

Creating the package skeleton

- 1.New Project -> New Directory -> R Package
- 2.Pick the name for your package (only letters, numbers, and .), and where it'll live locally.
- 3.Check the “Create git repository” because why not.
- 4.Click on the DESCRIPTION file, and edit it.
- 5.Put your code files in the R/ subdirectory.
- 6.Load the package (“Build” tab, More -> Load all; or ⌘-⇧-L)

DESCRIPTION file

Package: veepack

Version: 0.1

Date: 2016-11-27

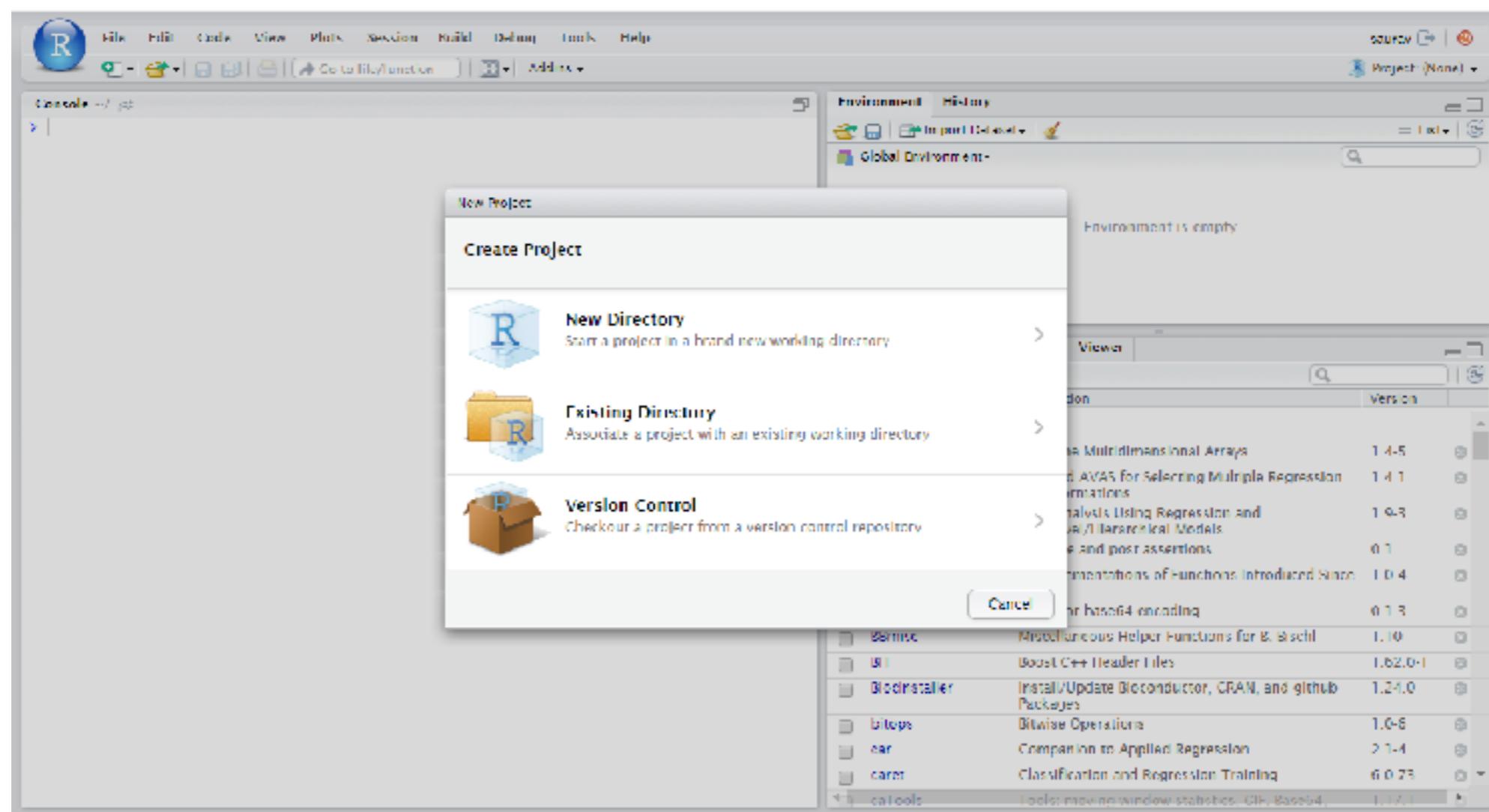
Title: Some functions

Description: Some description

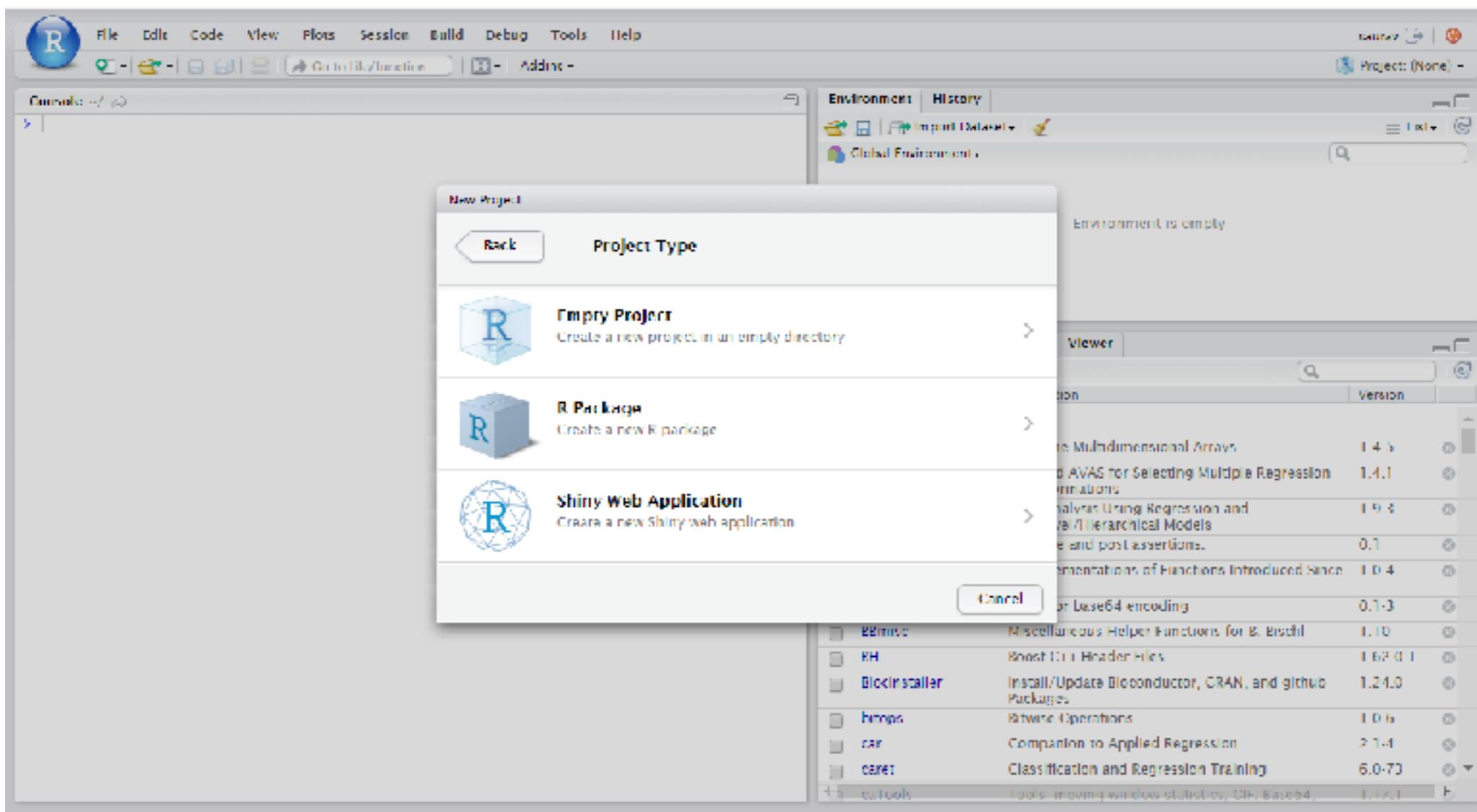
Author: Veerasak Kritsanaphan<veerasak.kritsanaphan@gmail.com>

Maintainer: Veerasak Kritsanaphan<veerasak.kritsanaphan@gmail.com>

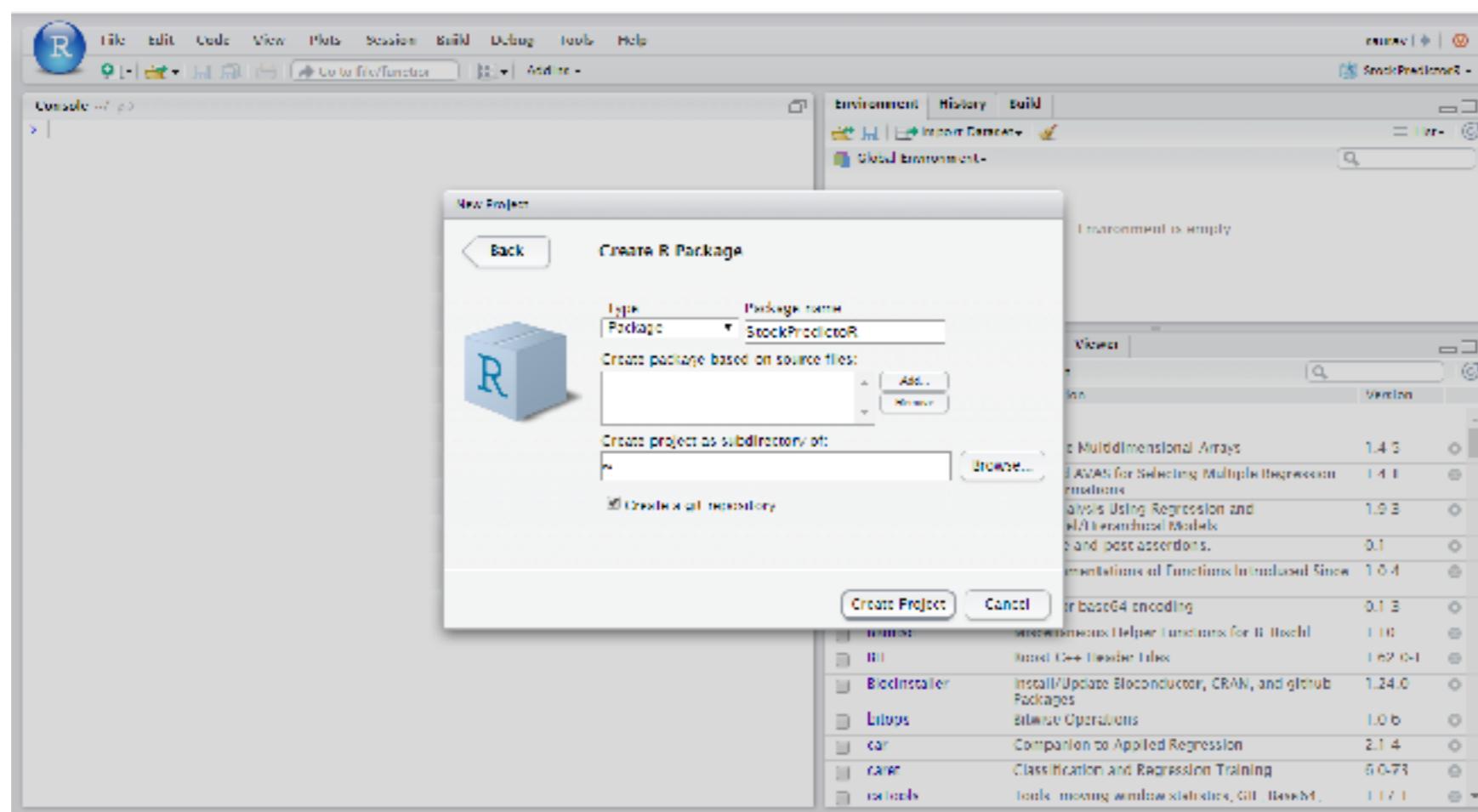
1.Create a new project by going to File > New Project.



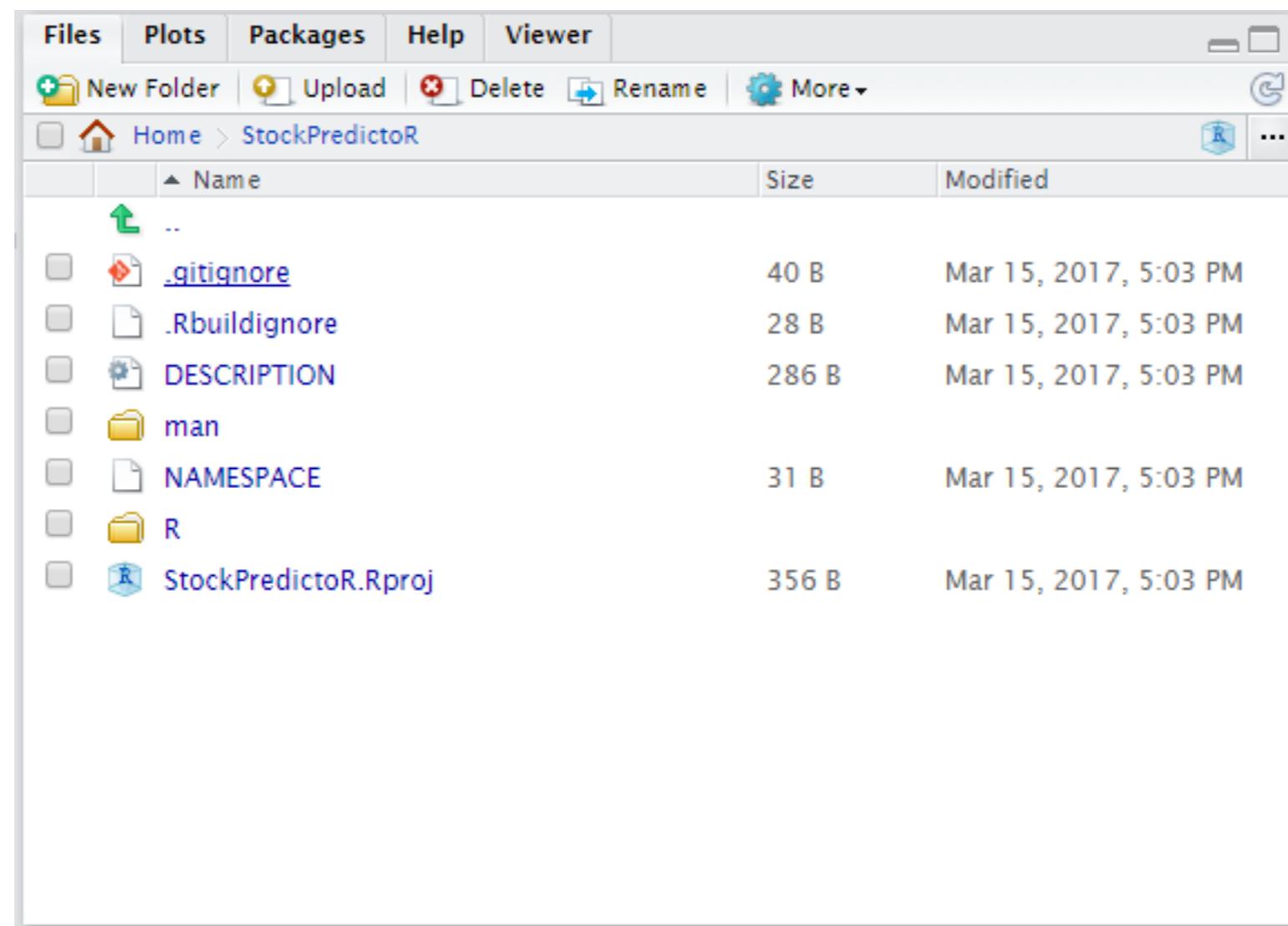
2. You can choose an existing directory or choose to create a new one. Then choose the project type as R package.



3. This is the time you choose an appropriate name for your package. I'm going to name it StockPredictoR. For naming your packages, you can use periods (like predictoR) or camel case as we are using here. I'll advice you to not use underscores while naming your packages. Also, choose the appropriate subdirectory that you want to store this project in.



4. This will create the following files in the directory. All the code will be stored in R folder while the manual and supporting documents will be stored in man folder.



5. Build package

The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. The Build menu is open, showing options: Build and Reload (Ctrl+Shift+B), Clean and Rebuild, Test Package (Ctrl+Alt+F7), Check Package (Ctrl+Shift+E), **Build Source Package** (highlighted in blue), Build Binary Package, Document (Ctrl+Shift+D), Stop Build, and Configure Build Tools... The main workspace shows an R script named DESCRIPTION.R with code for a package named StockPredictoR. The code includes a function stock_predict that takes a symbol and returns a probability. Below the script is a Console window showing the package being built and tested. The right panel displays the package structure in the Environment tab, showing files like .gitignore, Rbuildignore, Rhistory, DESCRIPTION, man, NAMESPACE, R, and StockPredictoR.Rproj. The Build tab shows the build process, and the Files tab lists the package contents.

```
R DESCRIPTION.R
#> #' @title Predicts Stock Price Movement
#> #' @description This package predicts tomorrow's market
#> #' @param symbol
#> #' @return NULL
#> #' @examples stock_predict("AAPL")
#> #' @export
#>
#> stock_predict<-function(symbol)
#> {
#>   #Importing price data for the given symbol
#>   data<-data.frame(as.xts(getSymbols(symbol)))
#>   #
```

```
Console -/StockPredictoR/ <-->
Restarting R session...

> library(StockPredictoR)
> example("stock_predict")

Stock p> stock_predict("AAPL")
[1] "Probability of Stock price going up tomorrow:"
[2] 0.448176
> stock_predict("GOOGL")
[1] "Probability of Stock price going up tomorrow:"
[1] 0.4718388

Restarting R session...

> library(StockPredictoR)
>
```

Build and Reload Ctrl+Shift+B
Clean and Rebuild
Test Package Ctrl+Alt+F7
Check Package Ctrl+Shift+E
Build Source Package
Build Binary Package
Document Ctrl+Shift+D
Stop Build
Configure Build Tools...

Environment History Build Git

3.2.2
* installing 'source' package 'StockPredictoR' ...
** R
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (StockPredictoR)

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Home > StockPredictoR

Name	Size	Modified
.gitignore	10 B	Mar 15, 2017, 5:03 PM
Rbuildignore	28 B	Mar 15, 2017, 5:03 PM
Rhistory	8 B	Mar 15, 2017, 7:19 PM
DESCRIPTION	424 B	Mar 16, 2017, 3:16 PM
man		
NAMESPACE	31 B	Mar 15, 2017, 5:03 PM
R		
StockPredictoR.Rproj	433 B	Mar 16, 2017, 2:54 PM



Inferential Statistics

Why inferential statistics?

Suppose you want to know the average salary of Data Science professionals in Thailand. Which of the following methods can be used to calculate it?

1. Meet every Data Science professional in Thailand. Note down their salaries and then calculate the total average?

2. Or hand pick a number of professionals in a city like Bangkok.

Note down their salaries and use it to calculate the Thailand average.

Enter Inferential Statistics

- In simple language, **Inferential Statistics is used to draw inferences beyond the immediate data available.**
- With the help of inferential statistics, we can answer the following questions:
 1. Making **inferences** about the **population** from the **sample**.
 2. Concluding whether a **sample** is **significantly** different from the **population**.

For example, let's say you collected the salary details of Data Science professionals in Chiangmai. And you observed that the average salary of Chiangmai's data scientists is more than the average salary across Bangkok. Now, we can conclude if the difference is statistically significant.

Enter Inferential Statistics

3. If **adding or removing a feature** from a **model** will really help to **improve** the model.
4. If **one model** is significantly **better** than **the other?**
5. **Hypothesis testing** in general.

Central Limit Theorem

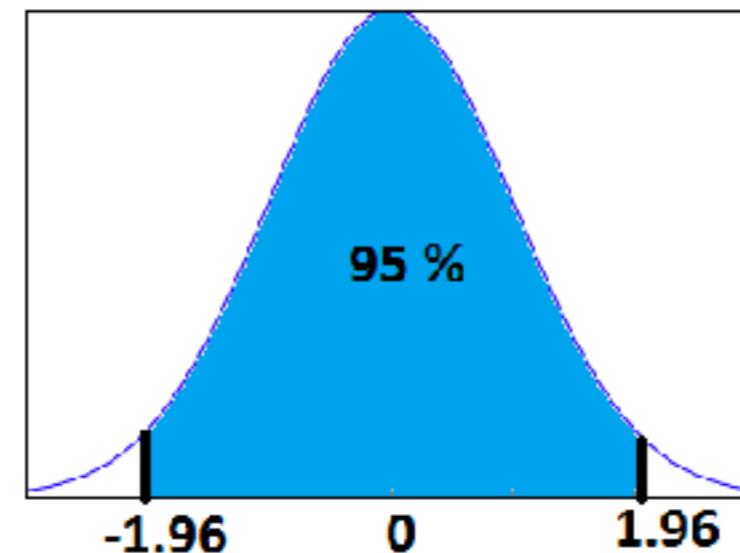
- It states that when plotting a sampling distribution of means, the mean of sample means will be equal to the population mean. And the sampling distribution will approach a **normal distribution** with **variance equal to σ/\sqrt{n}** where σ is the standard deviation of population and n is the sample size.

Points to note:

- Central Limit Theorem **holds true irrespective of the type of distribution of the population.**
- Now, we have a way to **estimate the population mean** by just making repeated observations of samples of a fixed size.
- **Greater the sample size, lower the standard error and greater the accuracy** in determining the population mean from the sample mean.

Confidence Interval

The confidence interval is a type of interval estimate from the sampling distribution which gives a range of values in which the population statistic may lie. Let us understand this with the help of an example.



We know that 95% of the values lie within 2 (1.96 to be more accurate) standard deviation of a normal distribution curve. So, for the above curve, the blue shaded portion represents the confidence interval for a sample mean of 0.

Hypothesis Testing

Fundamental of Hypothesis Testing

There two types of **statistical inferences**, **Estimation** and **Hypothesis Testing**

Hypothesis Testing: A hypothesis is a claim (assumption) about one or more population parameters.

- Average price of a six-pack in the U.S. is $\mu = \$4.90$
- The population mean monthly cell phone bill of this city is:
 $\mu = \$42$
- The average number of TV sets in U.S. Homes is equal to three; $\mu = 3$



It Is always about a population parameter, not about a sample statistic

Sample evidence is used to assess the probability that the claim about the population parameter is true

A. **It starts with Null Hypothesis, H_0**

$$H_0: \underline{\quad} = \underline{\quad}$$

1. We begin with the assumption that H_0 is true and any difference between the sample statistic and true population parameter is due to chance and not a real (systematic) difference.
2. Similar to the notion of “innocent until proven guilty”
3. That is, “innocence” is a null hypothesis.

Null Hypo, Continued

- 4. Refers to the status quo
- 5. Always contains “=”, “≤” or “≥” sign
- 6. May or may not be **rejected**

B. Next we state the Alternative Hypothesis, H_1

- 1. Is the opposite of the null hypothesis
 - 1. e.g., The average number of TV sets in U.S. homes is not equal to 3 ($H_1: \mu \neq 3$)
- 2. Challenges the status quo
- 3. Never contains the “=”, “≤” or “≥” sign
- 4. May or may not be **proven**
- 5. Is generally the hypothesis that the researcher is trying to prove.
Evidence is always examined with respect to H_1 , never with respect to H_0 .
- 6. We never “accept” H_0 , we either “reject” or “not reject” it

Summary:

- In the process of hypothesis testing, the null hypothesis initially is assumed to be true
- Data are gathered and examined to determine whether the evidence is strong enough with respect to the alternative hypothesis to reject the assumption.
- In other words, the burden is placed on the researcher to show, using sample information, that the null hypothesis is false.
- If the sample information is sufficient enough in favor of the alternative hypothesis, then the null hypothesis is rejected. This is the same as saying if the persecutor has enough evidence of guilt, the “innocence is rejected.”
- Of course, erroneous conclusions are possible, type I and type II errors.

Reason for Rejecting H_0

Illustration: Let say, we **assume** that average age in the US is 50 years ($H_0=50$). If in fact this is the true (unknown) population mean, it is unlikely that we get a sample mean of 20. So, if we have a sample that produces an average of 20, then we **reject** that the null hypothesis that average age is 50. (note that we are rejecting our assumption or claim). (would we get 20 if the true population mean was 50? NO. That is why we reject 50)

How Is the Test done?

We use the distribution of a Test Statistic, such as Z or t as the criteria.

A. Rejection Region Method:

Divide the distribution into rejection and non-rejection regions

Defines the unlikely values of the sample statistic if the null hypothesis is true, the critical value(s)

Defines **rejection region** of the sampling distribution

Rejection region(s) is designated by α , (level of significance)

Typical values are .01, .05, or .10

α is selected by the researcher at the beginning

α provides the critical value(s) of the test

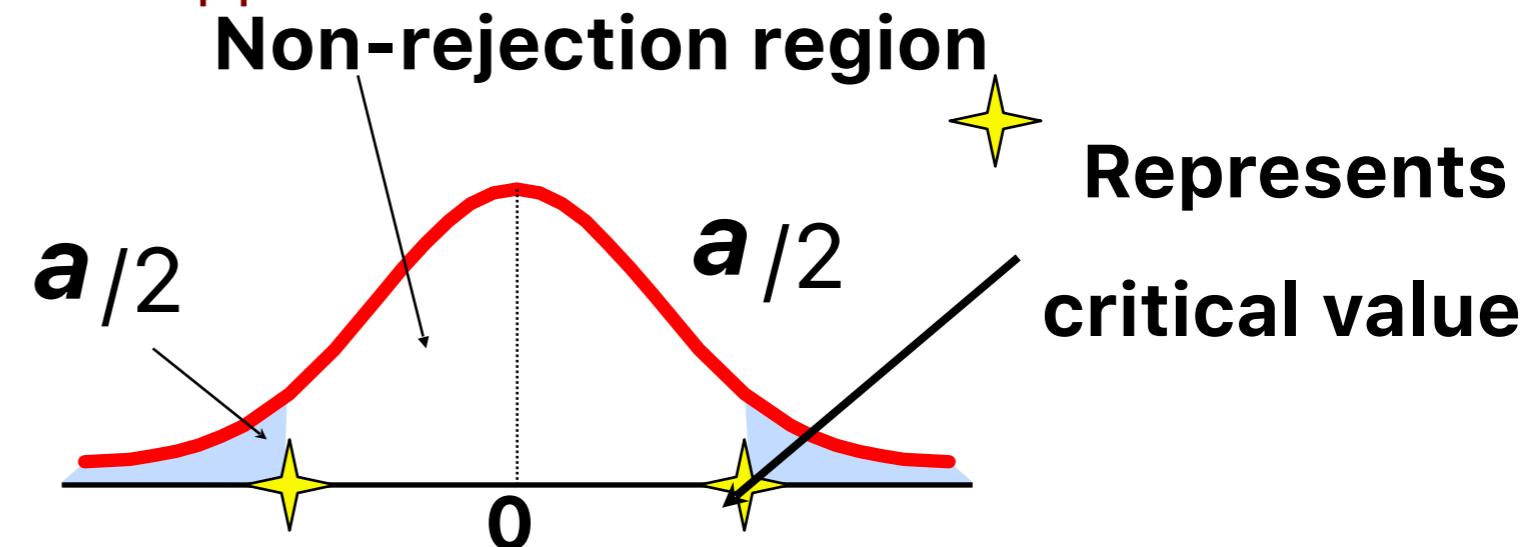
Rejection Region or Critical Value Approach:

Level of significance = α

$$H_0: \mu = 12$$

$$H_1: \mu \neq 12$$

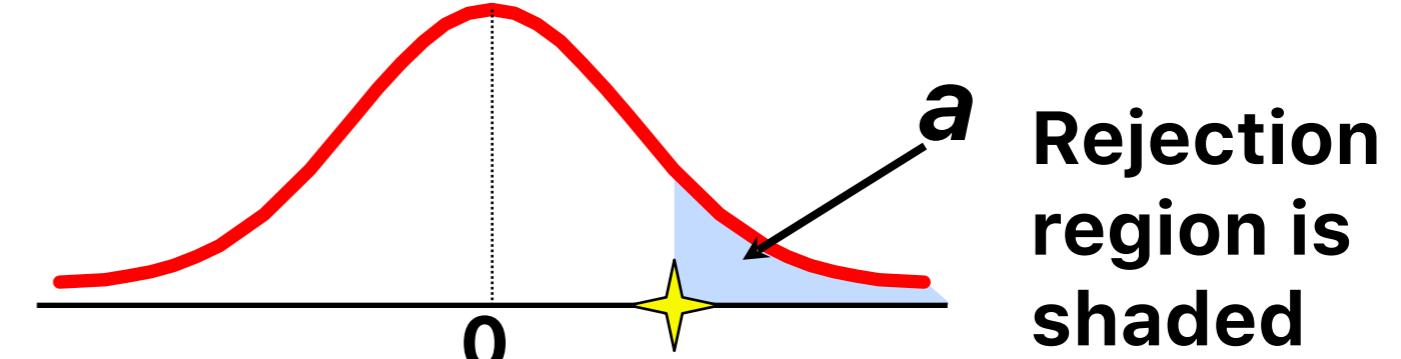
Two-tail test



$$H_0: \mu \leq 12$$

$$H_1: \mu > 12$$

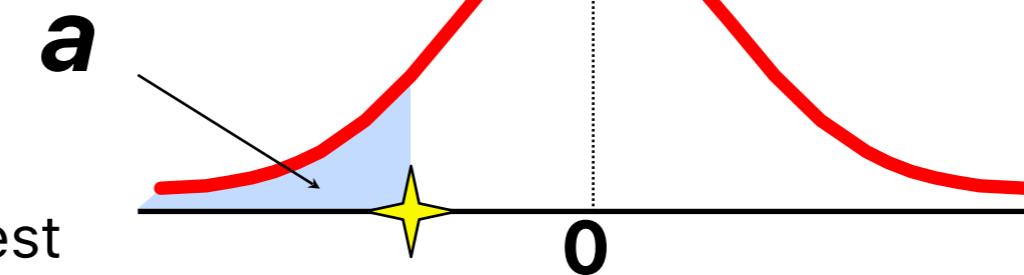
Upper-tail test



$$H_0: \mu \geq 12$$

$$H_1: \mu < 12$$

Lower-tail test



P-Value Approach –

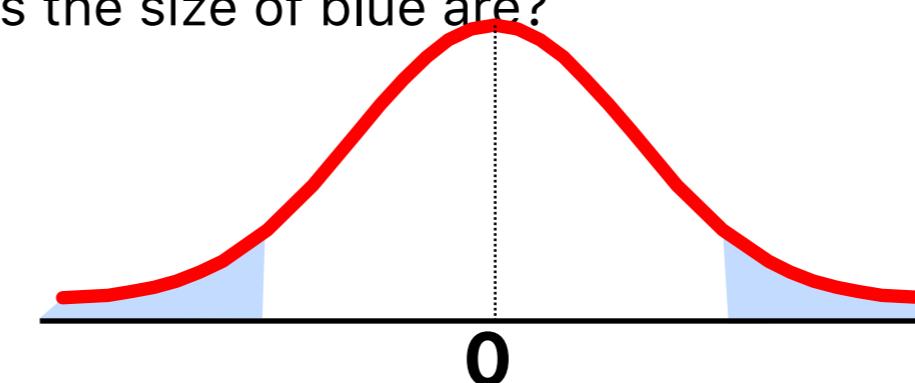
P-value=Max. Probability of (Type I Error), calculated from the sample.

Given the sample information what is the size of blue area?

$$H_0: \mu = 12$$

$$H_1: \mu \neq 12$$

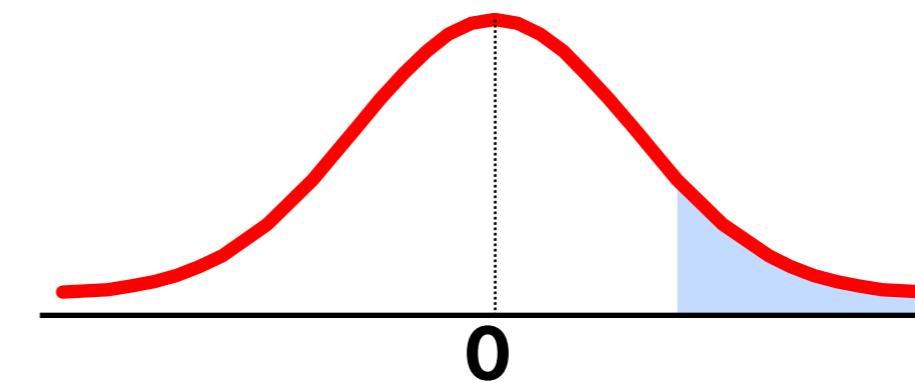
Two-tail test



$$H_0: \mu \leq 12$$

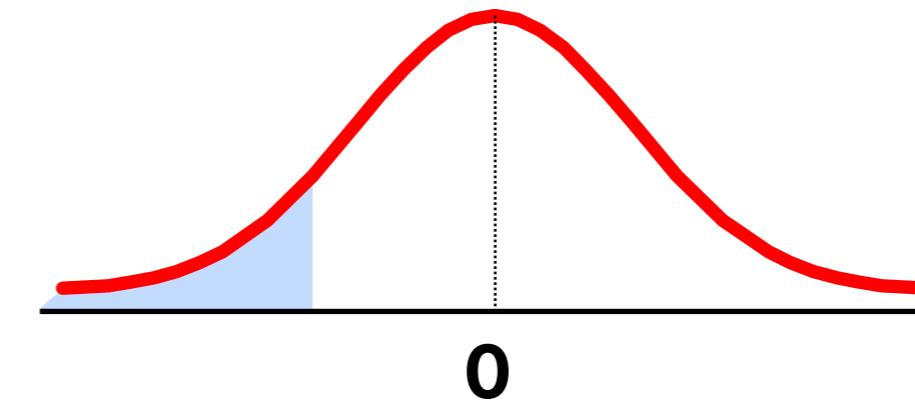
$$H_1: \mu > 12$$

Upper-tail test



$$H_0: \mu \geq 12$$

$$H_1: \mu < 12$$



Type I and II Errors:

The size of α , the rejection region, affects the risk of making different types of incorrect decisions.

Type I Error

Rejecting a **true null hypothesis** when it should **NOT** be rejected

Considered a serious type of error

The probability of Type I Error is α

It is also called **level of significance** of the test

Type II Error

Fail to reject a **false null hypothesis** that should have been rejected

The probability of Type II Error is β

Decision		Actual Situation			
		Hypothesis Testing		Legal System	
		H0 True	H0 False	Innocence	Not innocence
Do Not Reject H_0	No Error $(1 - \alpha)$	Type II Error (β)		No Error (not guilty, found not guilty) $(1 - \alpha)$	Type II Error (guilty, found not guilty) (β)
	Type I Error (α)	No Error $(1 - \beta)$		Type I Error (Not guilty, found guilty) (α)	No Error (guilty, found guilty) $(1 - \beta)$



Type I and Type II errors cannot happen at the same time

1. Type I error can only occur if H_0 is **true**
2. Type II error can only occur if H_0 is **false**
3. There is a tradeoff between type I and II errors. If the probability of type I error (α) increased, then the probability of type II error (β) declines.
4. When the difference between the hypothesized parameter and the actual true value is small, the probability of type two error (the non-rejection region) is larger.
5. Increasing the sample size, n , for a given level of α , reduces β



B. P-Value approach to Hypothesis Testing:

1. The rejection region approach allows you to examine evidence but restrict you to not more than a certain probability (say $\alpha = 5\%$) of rejecting a true H_0 by mistake.
2. The P-value approach allows you to use the information from the sample and then calculate the **maximum probability** of rejecting a true H_0 by mistake.
3. Another way of looking at P-value is the probability of observing a sample information of "A=11.5" when the true population parameter is "12=B". The P-value is the **maximum probability** of such mistake taking

- 
4. That is to say that P-value is the smallest value of α for which H_0 can be rejected based on the sample information
 5. Convert Sample Statistic (e.g., sample mean) to Test Statistic (e.g., Z statistic)
 6. Obtain the p-value from a table or computer
 7. Compare the p-value with α

If p-value < α , reject H_0

If p-value $\geq \alpha$, do not reject H_0

Test of Hypothesis for the Mean

σ known

σ Unknown

The test statistic is:

$$z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

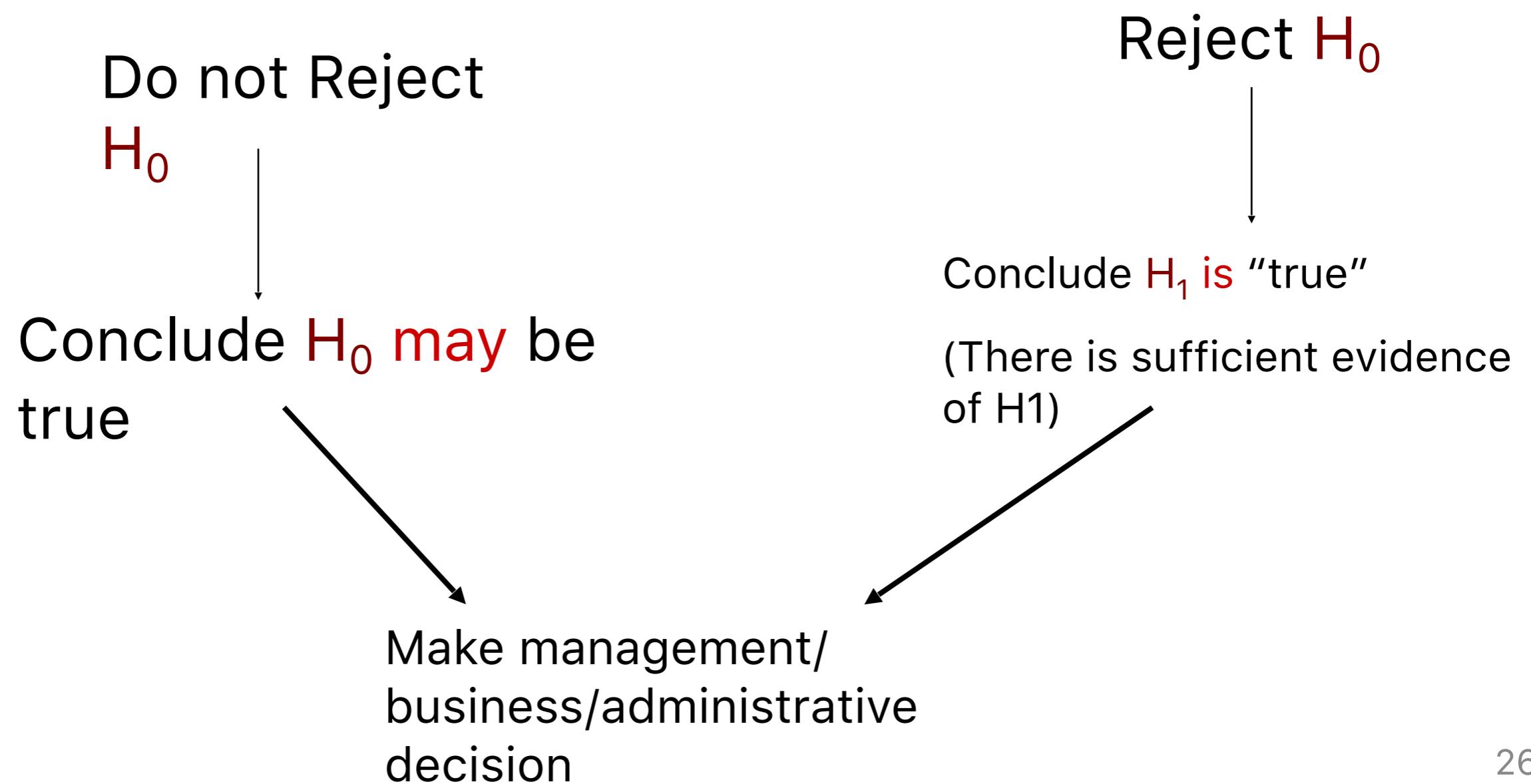
The test statistic is:

$$t_{n-1} = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}}$$

Steps to Hypothesis Testing

1. State the H_0 and H_1 clearly
2. Identify the test statistic (two-tail, one-tail, and Z or t distribution)
3. Depending on the type of risk you are willing to take, specify the level of significance,
4. Find the decision rule, critical values, and rejection regions. If –
 $CV < \text{actual value (sample statistic)} < +CV$, then **do not reject the H_0**

Make statistical decision



When do we use a two-tail test?

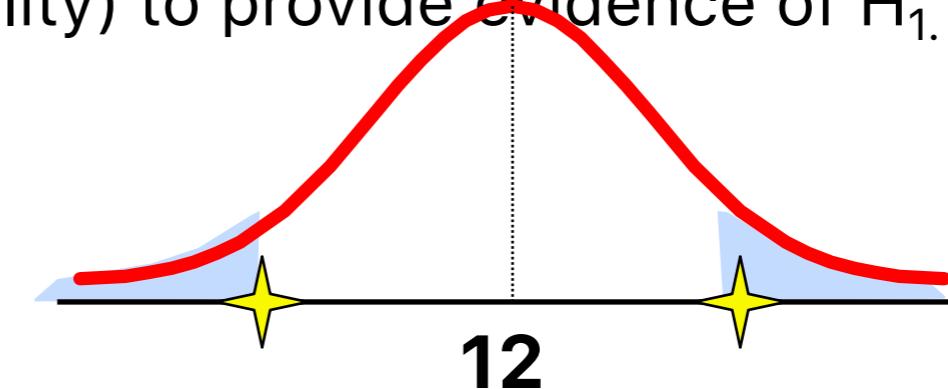
The answer depends on the question you are trying to answer.

A two-tail is used when the researcher has no idea which direction the study will go, interested in both direction. (example: testing a new technique, a new product, a new theory and we don't know the direction)

A new machine is producing 12 fluid once can of soft drink. The quality control manager is concern with cans containing too much or too little. Then, the test is a two-tailed test. That is the two rejection regions in tails is most likely (higher probability) to provide evidence of H_1 .

$$H_0 : \mu = 12 \text{ oz}$$

$$H_1 : \mu \neq 12 \text{ oz}$$



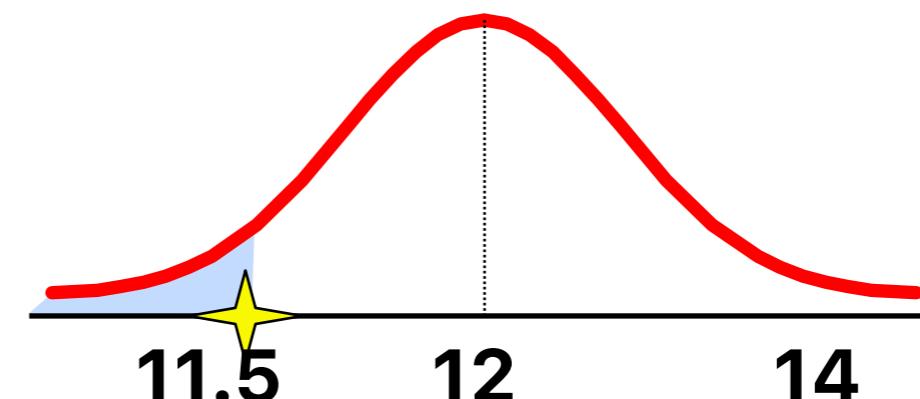
One-tail test is used when the researcher is interested in the direction.

Example: The soft-drink company puts a label on cans claiming they contain 12 oz. A consumer advocate desires to test this statement. She would assume that each can contains **at least** 12 oz and tries to find evidence to the contrary. That is, she examines the evidence for less than 12 oz.

What tail of the distribution is the most logical (higher probability) to find that evidence? The only way to reject the claim is to get evidence of less than 12 oz, left tail.

$$H_0 : \mu \geq 12 \text{ oz}$$

$$H_1 : \mu < 12 \text{ oz}$$



Review of Hypo. Testing

What is HT?

Probability of making erroneous conclusions

Type I – only when Null Hypo is true

Type II – only when Null Hypo is false

Two Approaches

The Rejection or Critical Value Approach

The P-value Approach (we calculate the observed level of significance)

Test Statistics

Z- distribution if Population Std. Dev. is Known

t-distribution if the Population Std. Dev. is unknown

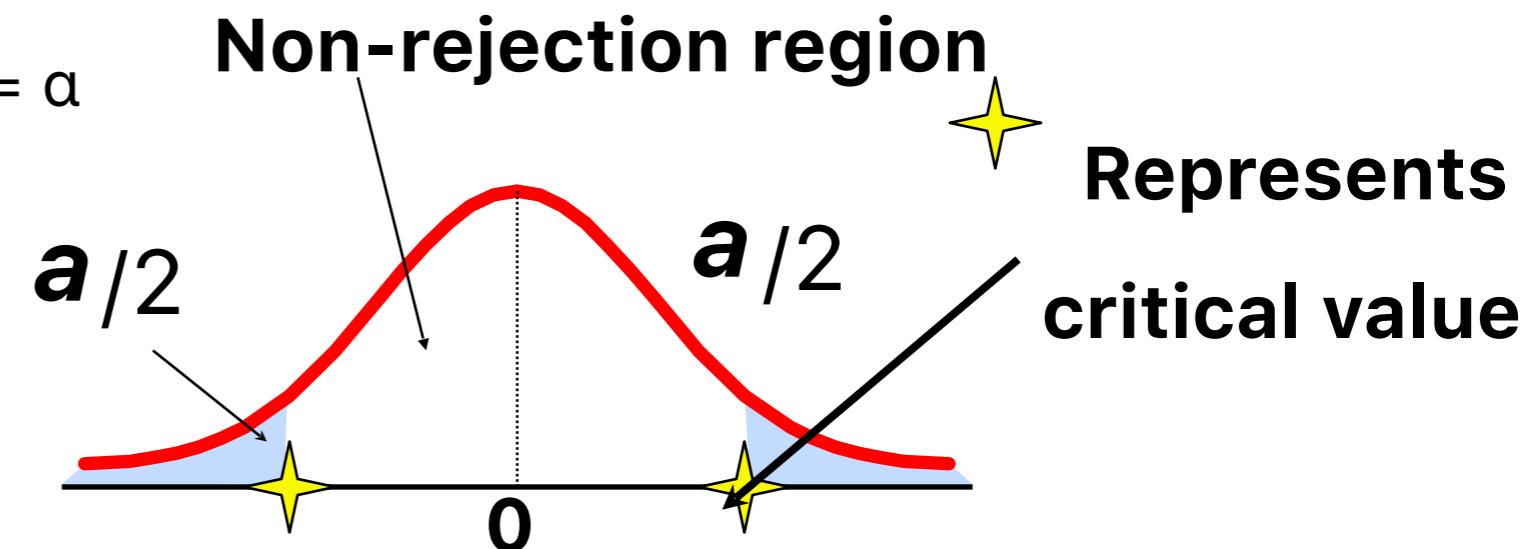
Rejection Region or Critical Value Approach:

The given level of significance = α

$$H_0: \mu = 12$$

$$H_1: \mu \neq 12$$

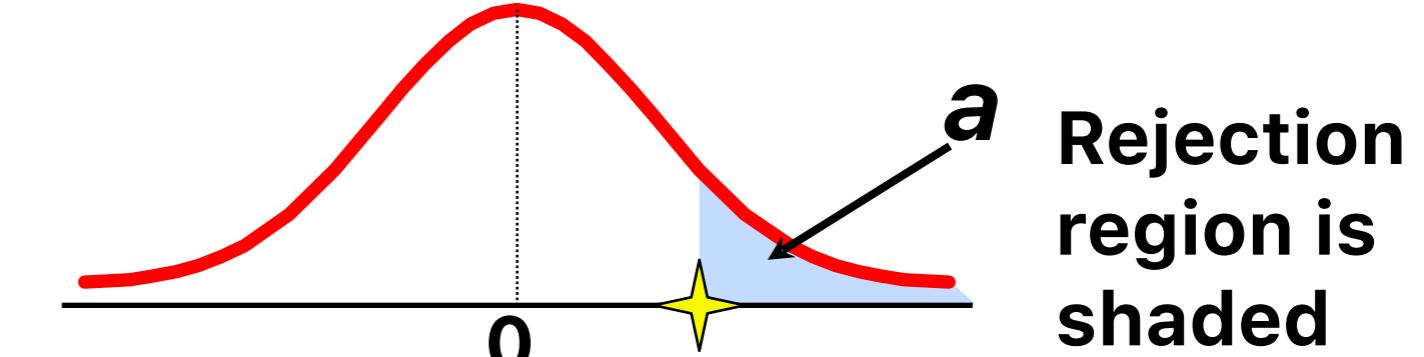
Two-tail test



$$H_0: \mu \leq 12$$

$$H_1: \mu > 12$$

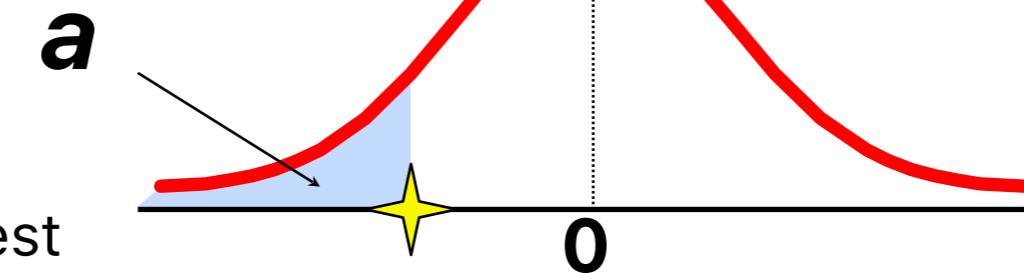
Upper-tail test



$$H_0: \mu \geq 12$$

$$H_1: \mu < 12$$

Lower-tail test



P-Value Approach –

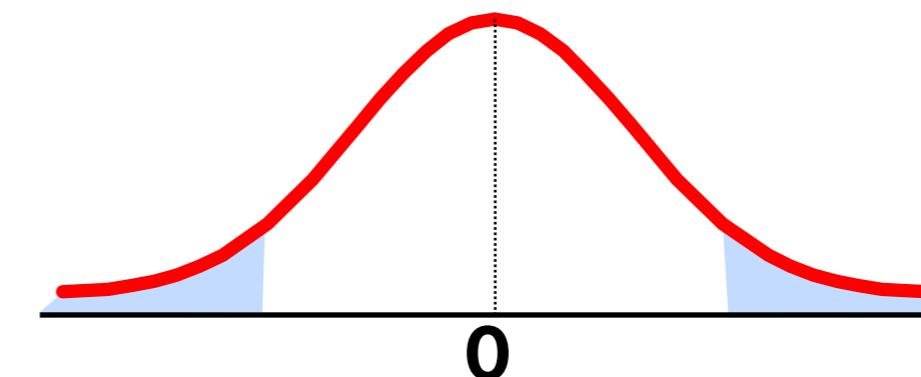
P-value=Max. Probability of (Type I Error), calculated from the sample.

Given the sample information what is the size of the blue areas? (The observed level of significance)

$$H_0: \mu = 12$$

$$H_1: \mu \neq 12$$

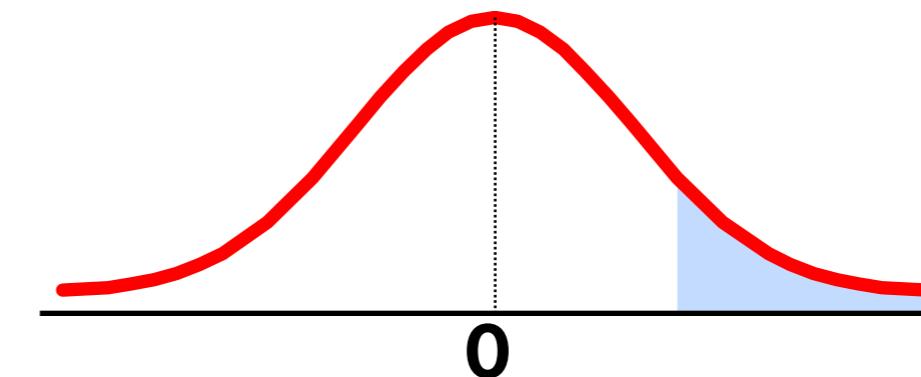
Two-tail test



$$H_0: \mu \leq 12$$

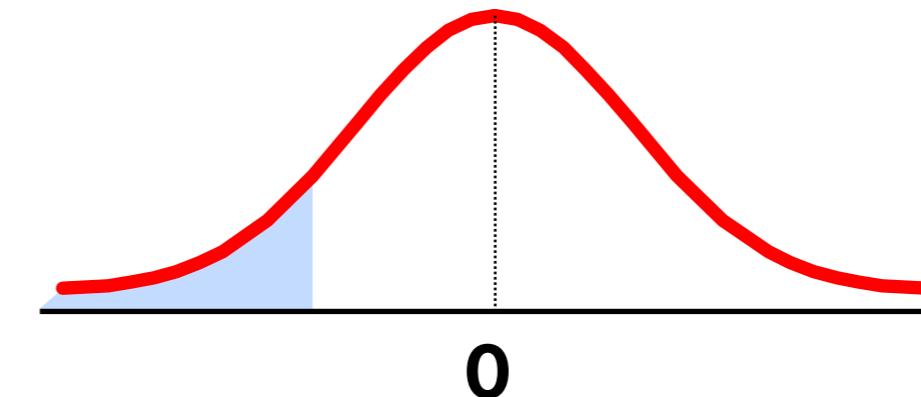
$$H_1: \mu > 12$$

Upper-tail test



$$H_0: \mu \geq 12$$

$$H_1: \mu < 12$$





Example 1:

Let's assume a sample of 25 cans produced a sample mean of 11.5 0z and the population std dev=1 0z.

Question 1:

At a 5% level of significance (that is allowing for a maximum of 5% prob. of rejecting a true null hypo), is there evidence that the population mean is different from 12 oz?

Null Hypo is:?

Alternative Hypo is?

Can both approaches be used to answer this question?

- 
- A: **Rejection region approach:** calculate the actual test statistics and compare it with the critical values
 - B: **P-value approach:** calculate the actual probability of type I error given the sample information. Then compare it with 1%, 5%, or 10% level of significance.

Interpretation of Critical Value/Rejection Region Approach:

Interpretation of P-value Approach:



Question 2:

At a 5% level of significance (that is allowing for a maximum of 5% prob. of rejecting a true null hypo), is the evidence that the population mean is **less than 12 oz**?

Null Hypo is:?

Alternative Hypo is?

Can both approaches be used to answer this question?

Interpretation of Critical Value Approach:

Interpretation of P-value Approach:

Question 3:

If in fact the pop. mean is 12 oz, what is the probability of obtaining a sample mean of 11.5 or less oz (sample size 25)? Null

Null Hypo is:?

Alternative Hypo is?

Question 4:

If in fact the pop. mean is 12 oz, and the sample mean is 11.5 (or less), what is the probability of erroneously rejecting the null hypo that the pop. mean is 12 oz?

Null Hypo is:?

Alternative Hypo is?

Can both approaches be used to answer these question?

Connection to Confidence Intervals

While the confidence interval estimation and hypothesis testing serve different purposes, they are based on same concept and conclusions reached by two methods are consistent for a two-tail test.

In CI method we estimate an interval for the population mean with a degree of confidence. If the estimated interval **contains** the hypothesized value under the hypothesis testing, then this is equivalent of **not rejecting** the null hypothesis. For example: for the beer sample with mean 5.20, the confidence interval is:

$$P(4.61 \leq \mu \leq 5.78) = 95\%$$

Since this interval contains the Hypothesized mean (\$4.90), we do not (did not) reject the null hypothesis at $\alpha = .05$

Did not reject and within the interval, thus consistent results.



t-test using R

Testing differences in mean between two groups

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

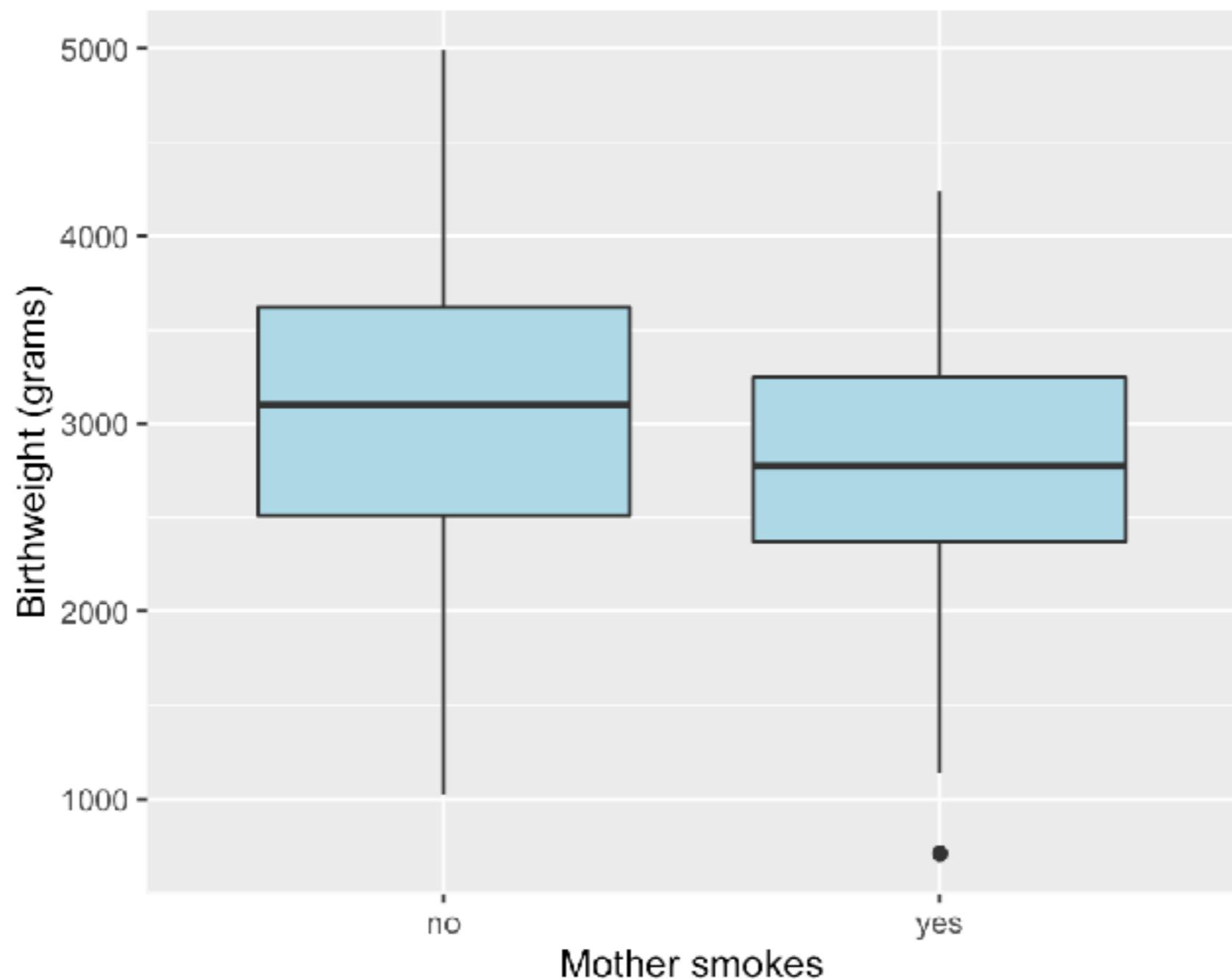
```
library(MASS)
library(plyr)
library(ggplot2)
```

```
# 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"))))
)
```

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
```



The standard deviation is good to have, but to assess statistical significance we really want to have the standard error (which the standard deviation adjusted by the group size).

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

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

t-test via t.test()

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
```

p-value

```
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
```

Workshop 8: t-test

Testing means between two groups

(a) Using the Cars93 data and the `t.test()` function, run a t-test to see if average MPG.highway is different between US and non-US vehicles.

Try doing this both using the formula style input and the x, y style input.

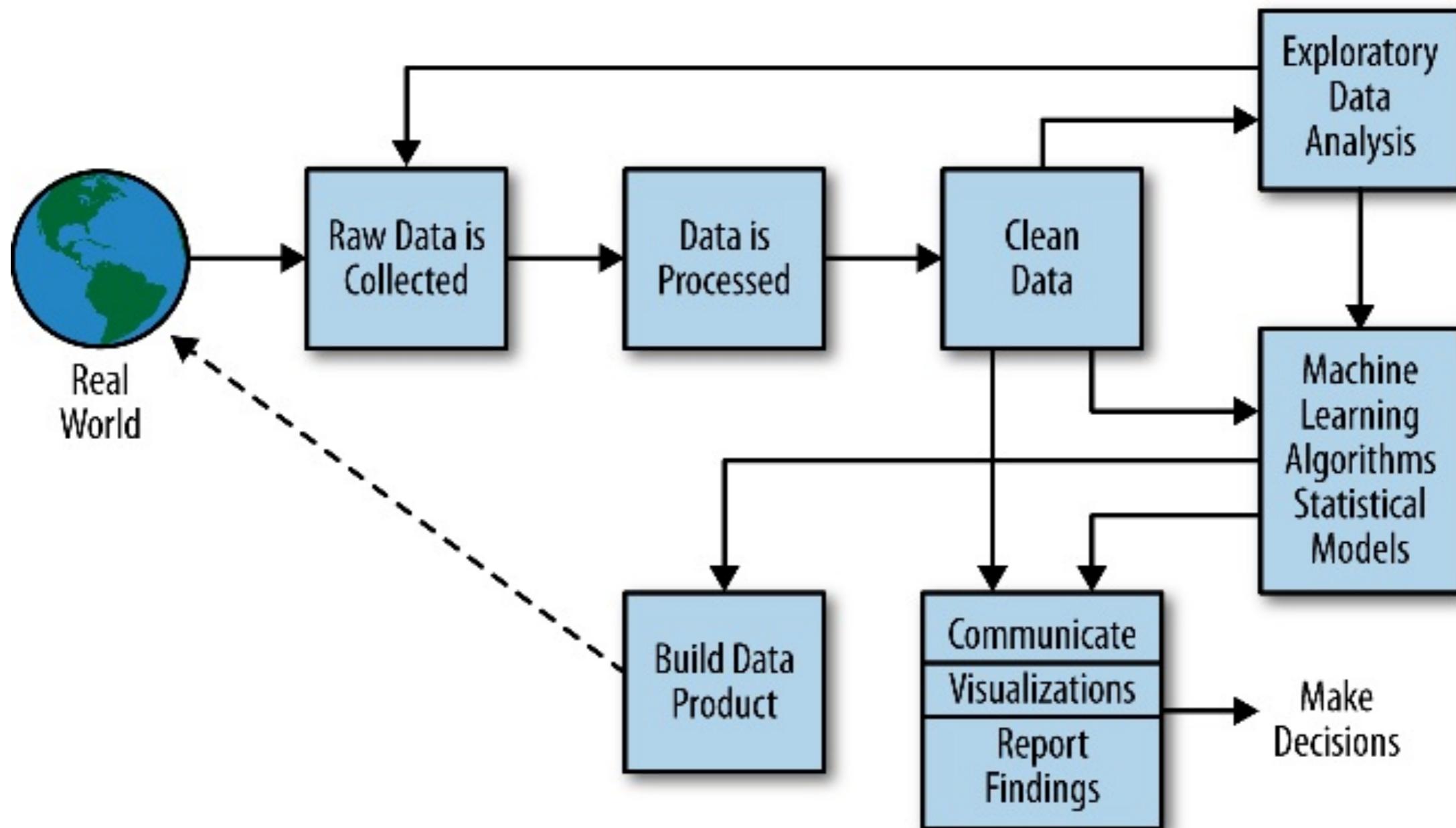
(b) What is the confidence interval for the difference?



Data Science Life Cycle

Fundamental Data Science for Data Scientist

Data Science Process





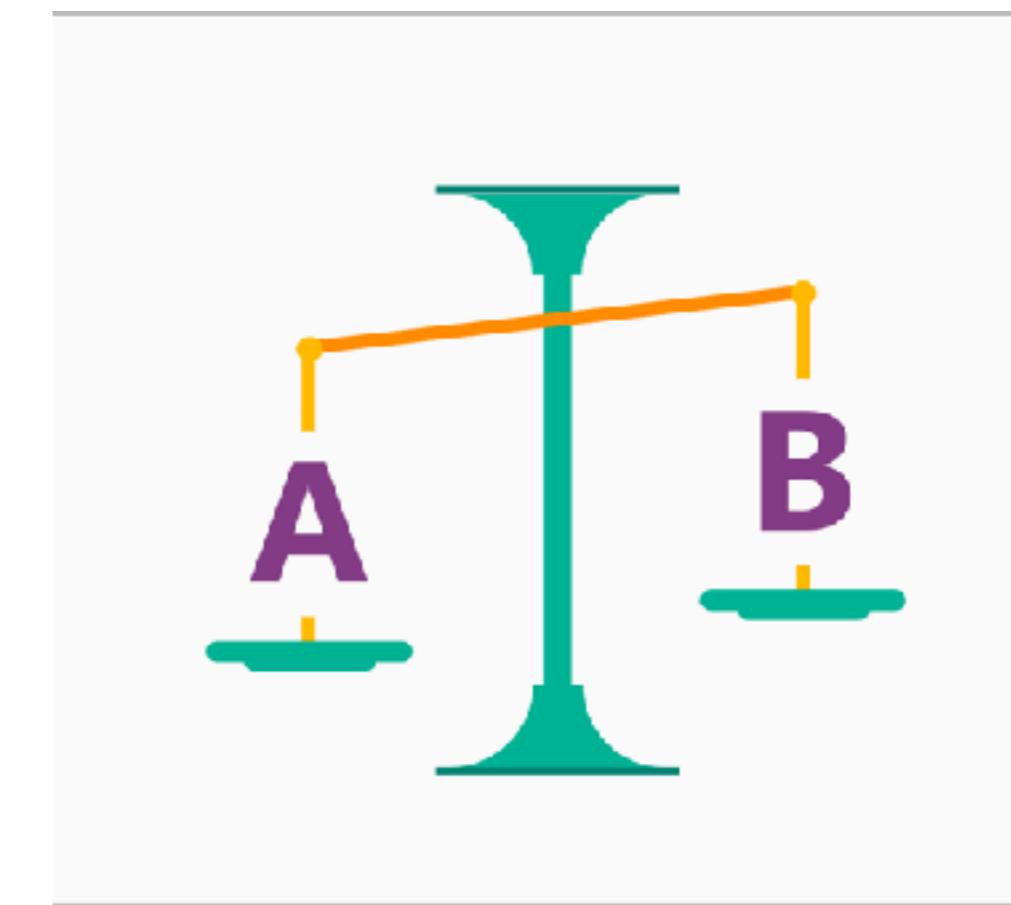
Machine Learning Concept

What **questions** can data science **answer**?

1. Is this **A** or **B**?
2. Is this **weird**?
3. How **much** or how **many**?
4. How is this **organized**?
5. What should I **do next**?

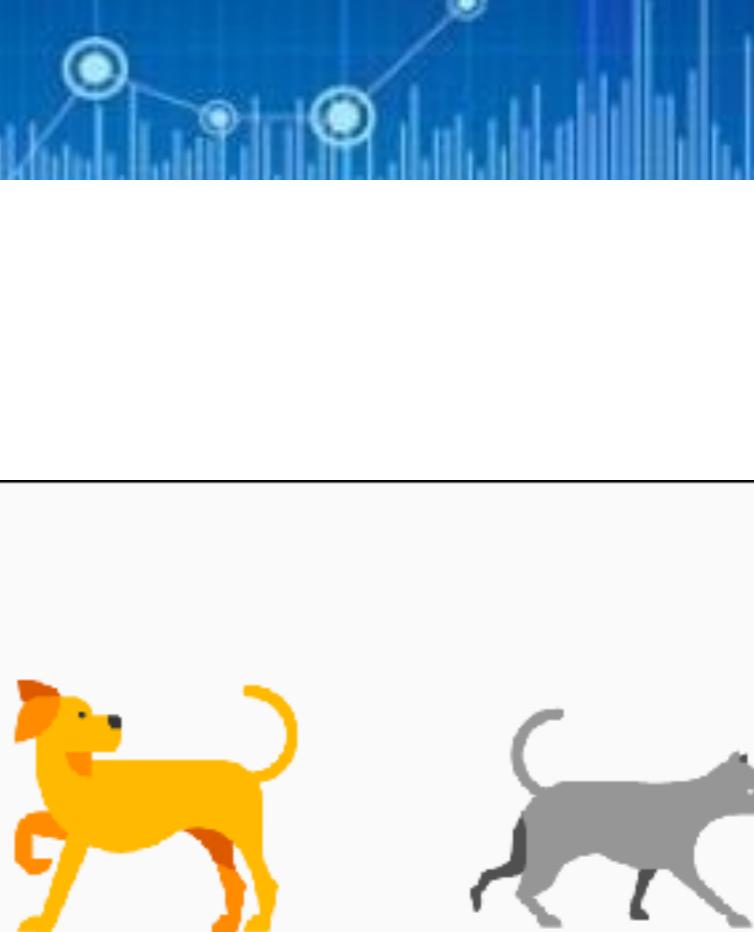
Is this A or B?

Classification algorithm



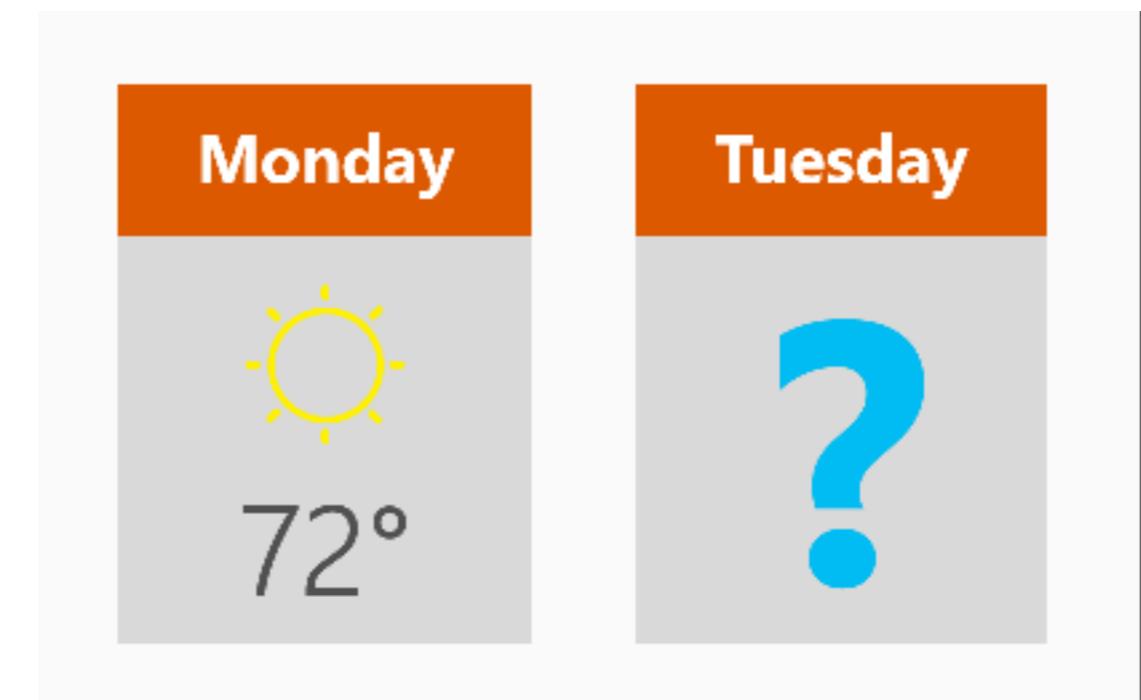
Is this weird?

Anomaly detection algorithm



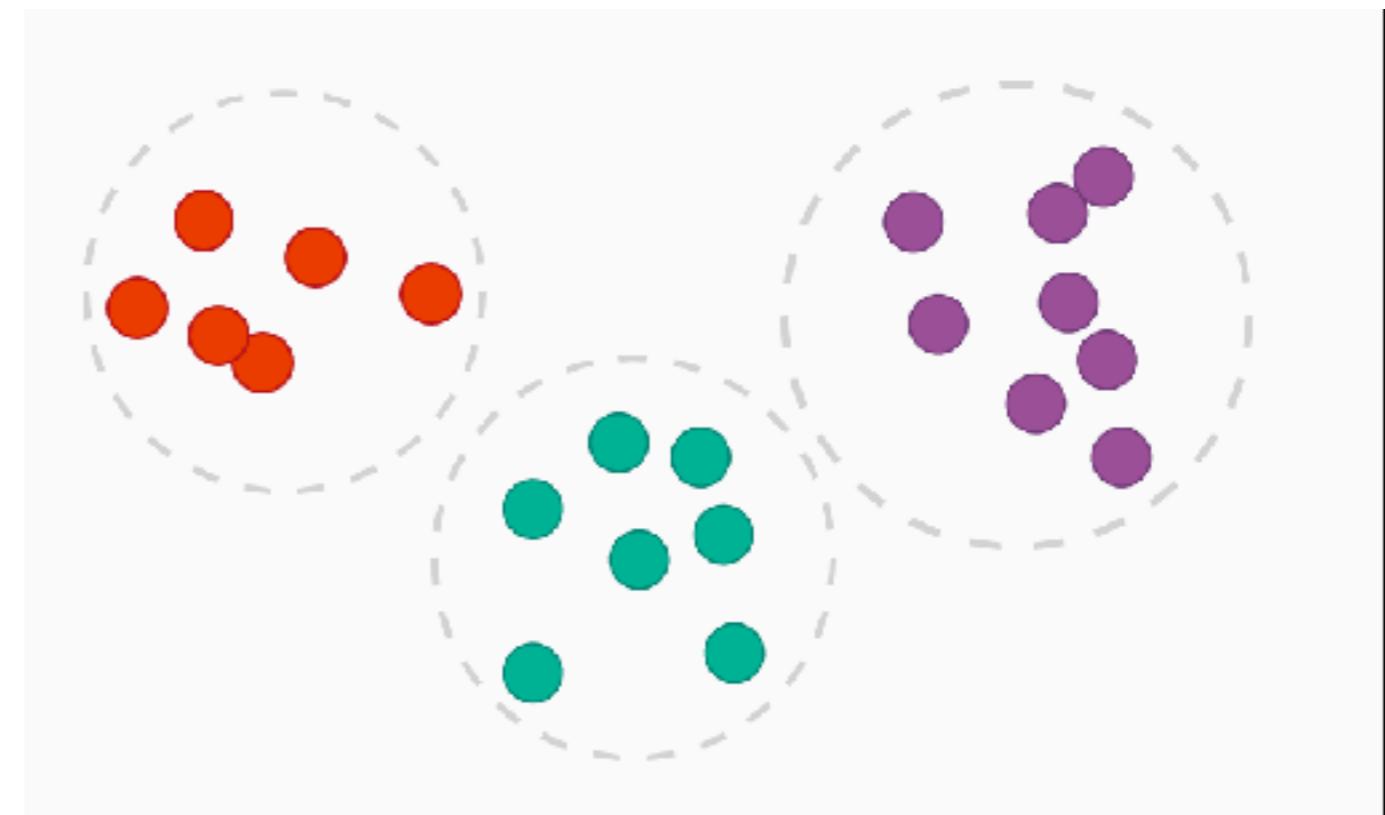
How much and How many?

Regression algorithm



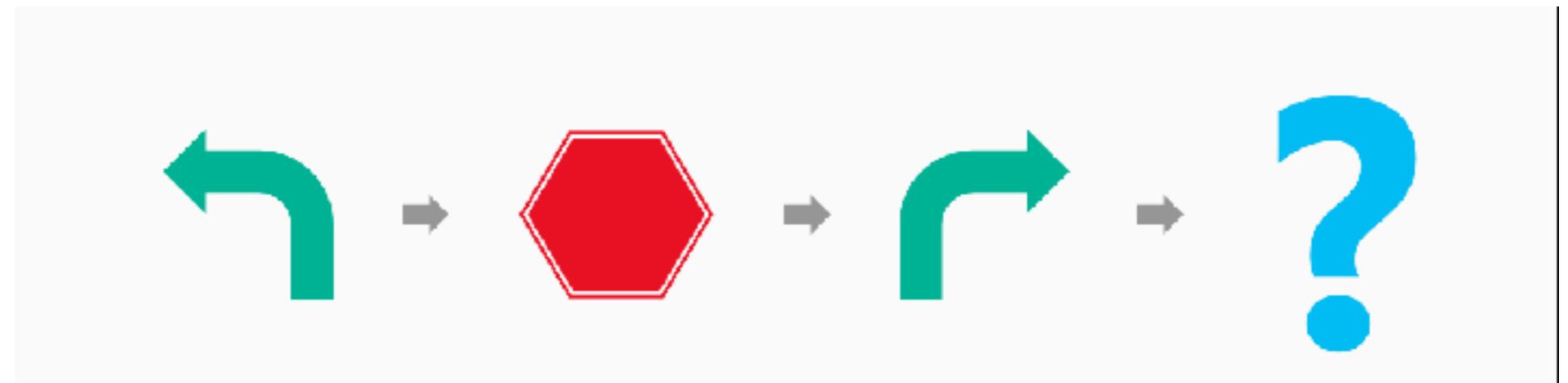
How is this organized?

Clustering algorithm

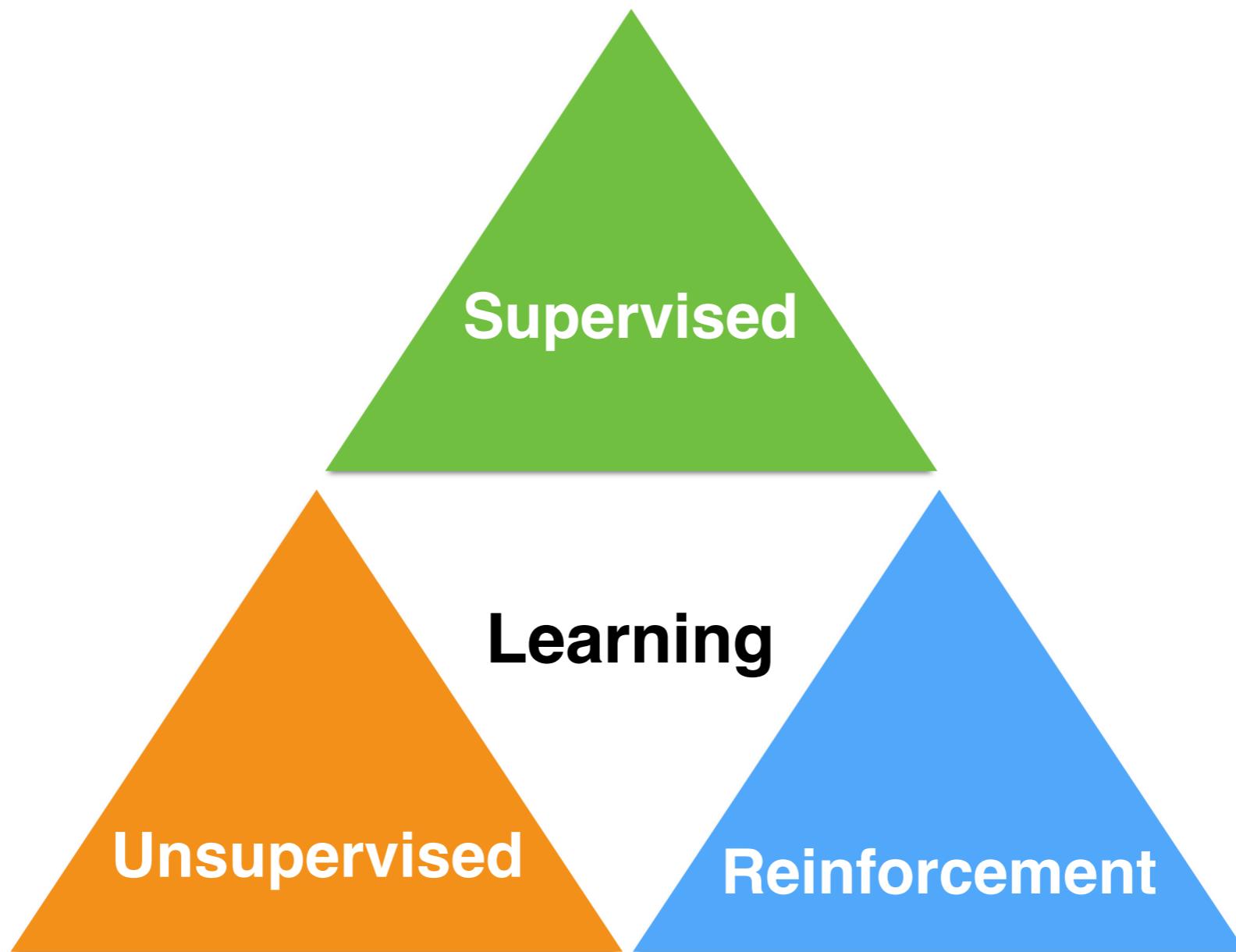


What should I do next?

Reinforcement Learning algorithm



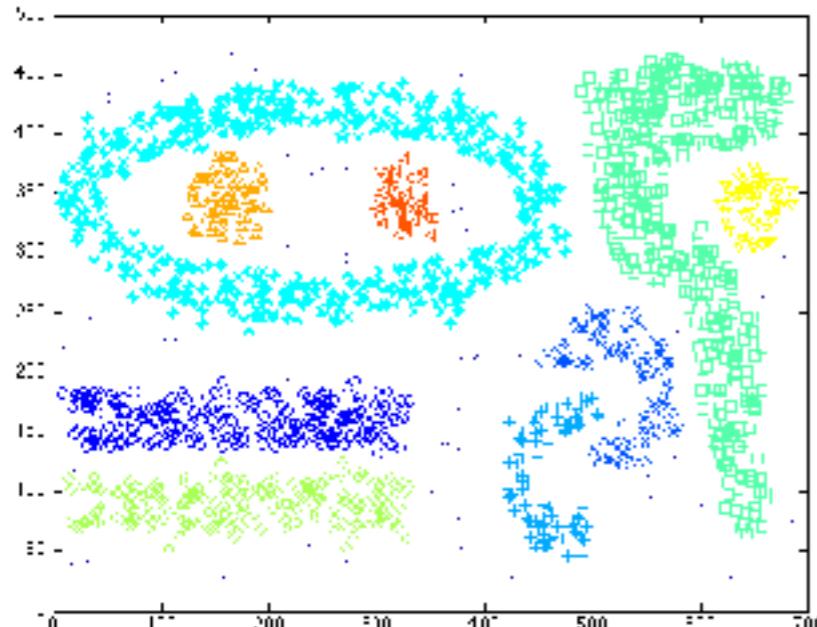
- Labeled data
- Direct feedback
- Predict outcome/future



- No labels
- No feedback
- “Find hidden structure”

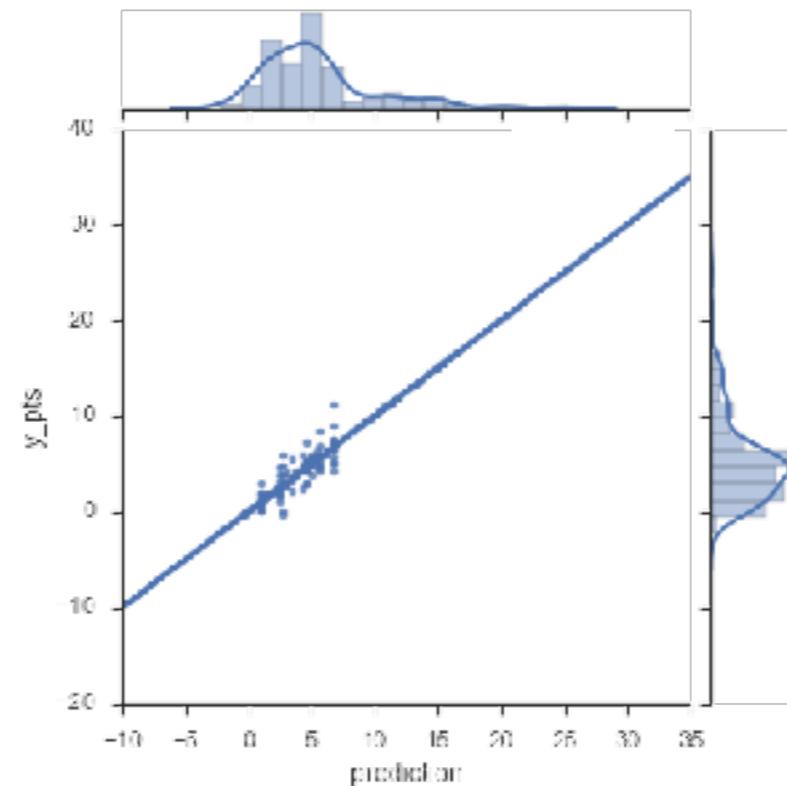
- Decision process
- Reward system
- Learn series of actions

Unsupervised Learning

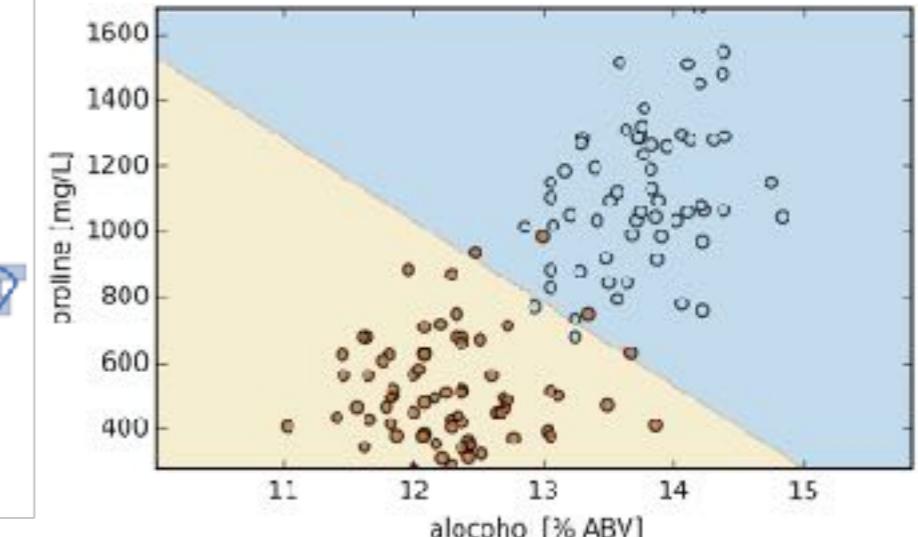


Clustering:
[DBSCAN on a toy dataset]

Supervised Learning



Regression:
[Soccer Fantasy Score prediction]



Classification:
[SVM on 2 classes of the Wine dataset]

Regression

Introduction to Regression Analysis

Regression analysis is used to:

1. Predict values of a dependent variable, Y , based on its relationship with values of at least one independent variable, X .
2. Explain the impact of changes in an independent variable on the dependent variable by estimating the **numerical value** of the relationship

Dependent variable: the variable we wish to explain

Independent variable: the variable used to explain the dependent variable

Simple Linear Regression Model

Only **one** independent variable (thus, simple), X

Relationship between X and Y is described by a linear function

Changes in Y are assumed to be caused by changes in X, that is,

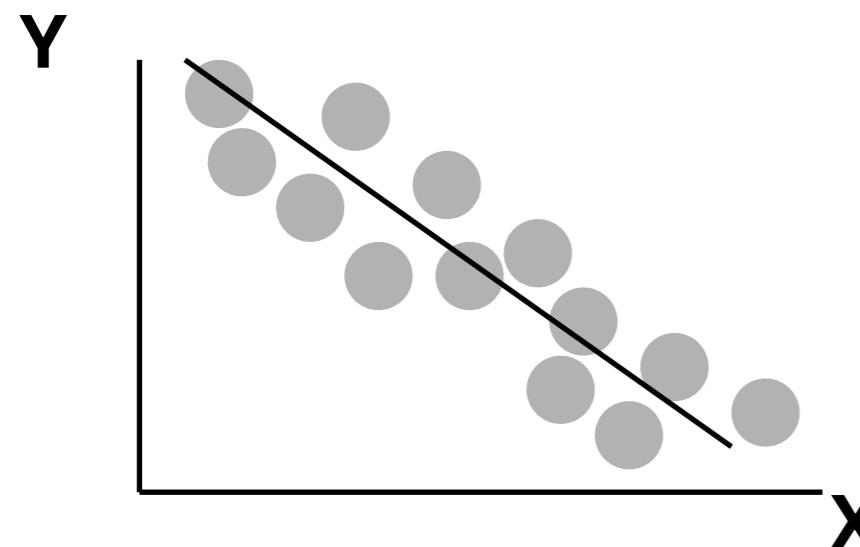
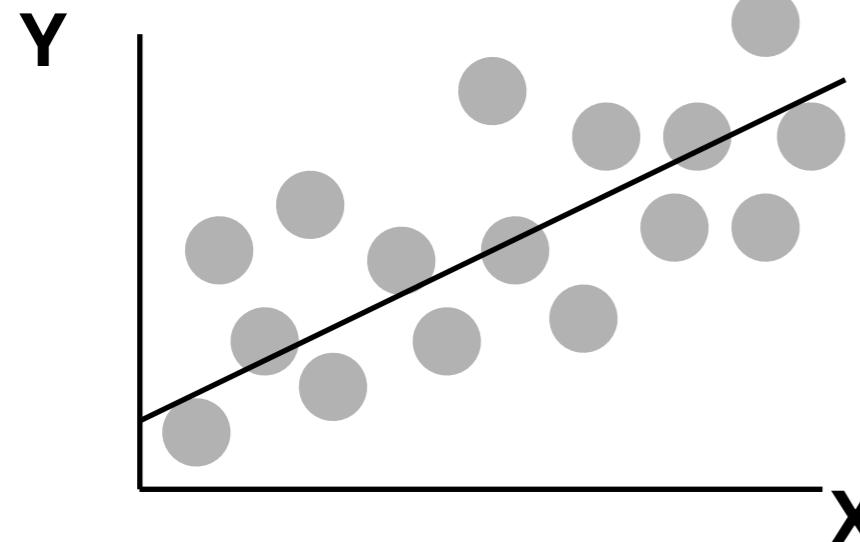
Change In X —————→ **Causes** Change in Y

Important points before we start a regression analysis:

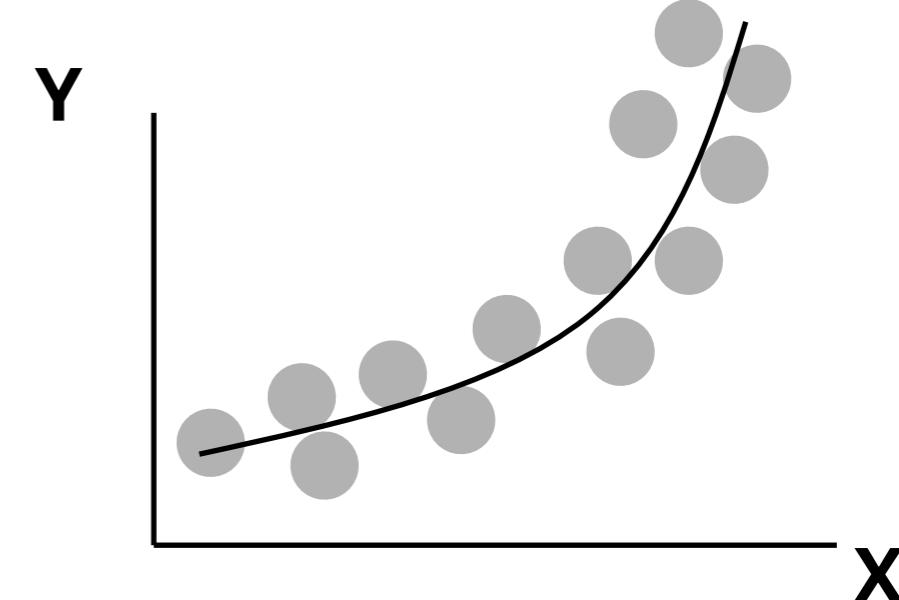
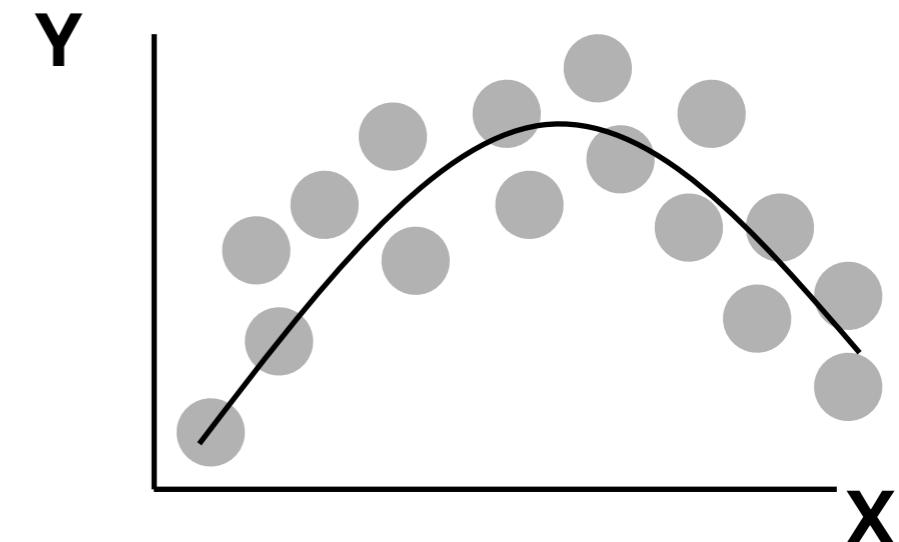
- The most important thing in deciding whether or not there is a relationship between X and Y is to have a systematic model that is based on **logical reasons**.
- Investigate the nature of the relationship between X and Y (use scatter diagram, covariance, correlation coefficient)
- Remember that regression is not an exact or deterministic mathematical equation. It is a **behavioral relationship** that is subject to randomness.
- Remember that X is not the only thing that explains the behavior of Y. There are other factor that you may not have information about.
- All you are trying to do is to have an estimate of the relationship using the **best linear fit** possible

Types of Relationships

Linear relationships



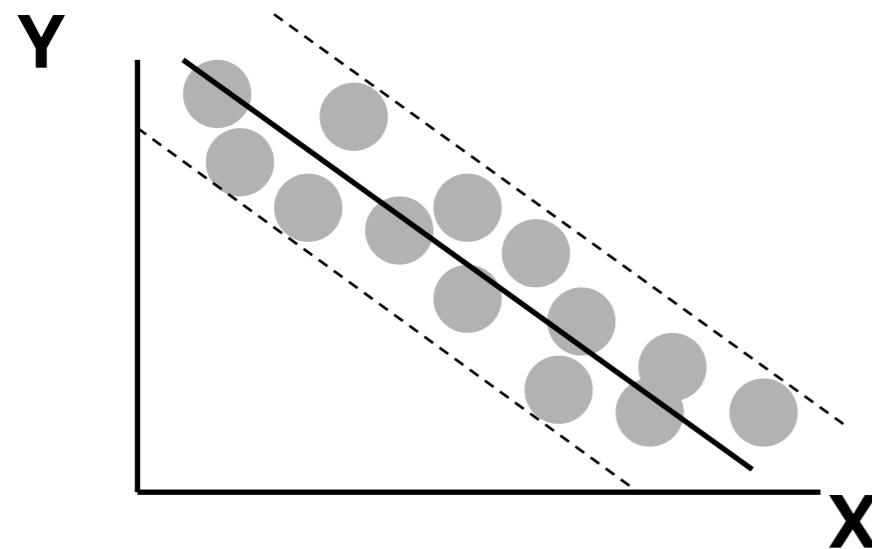
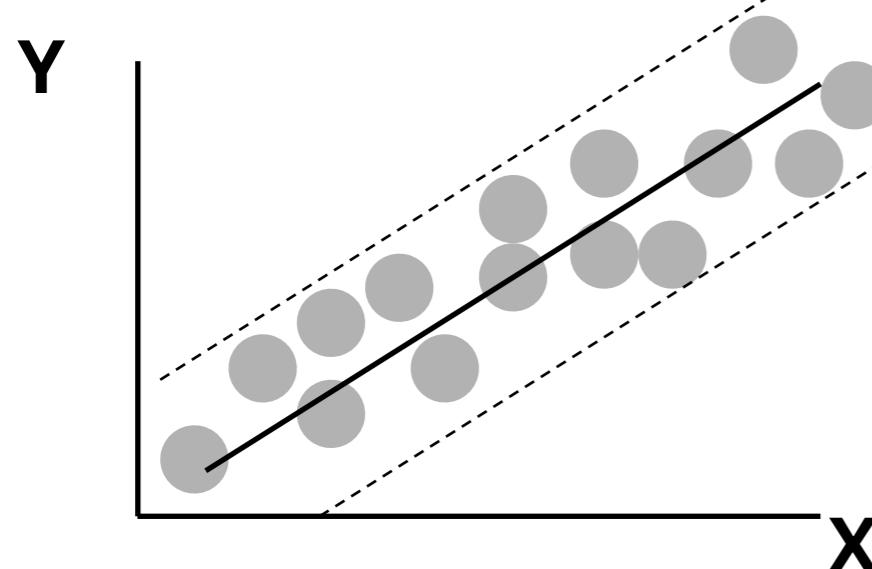
Curvilinear relationships



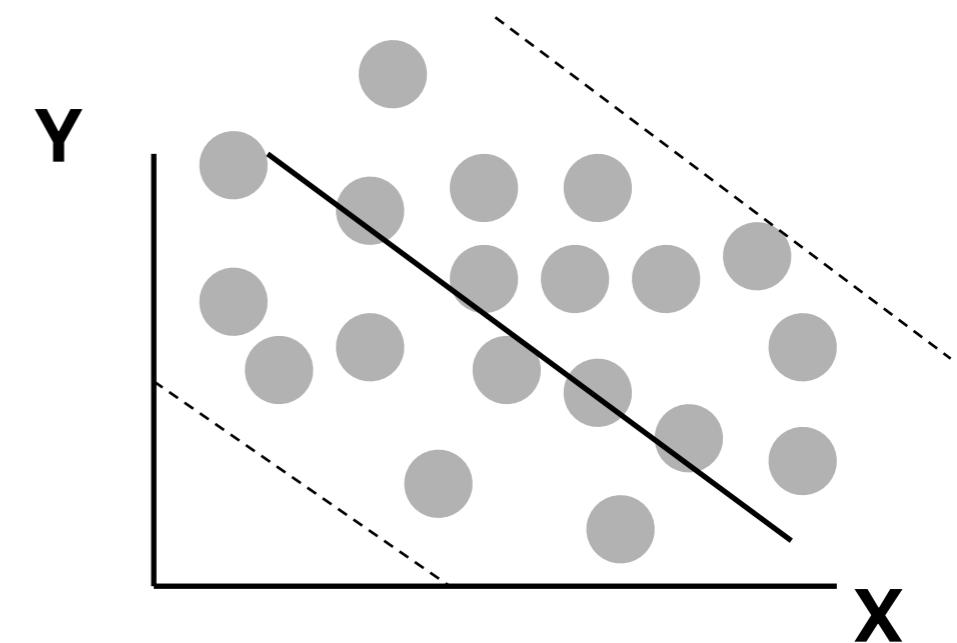
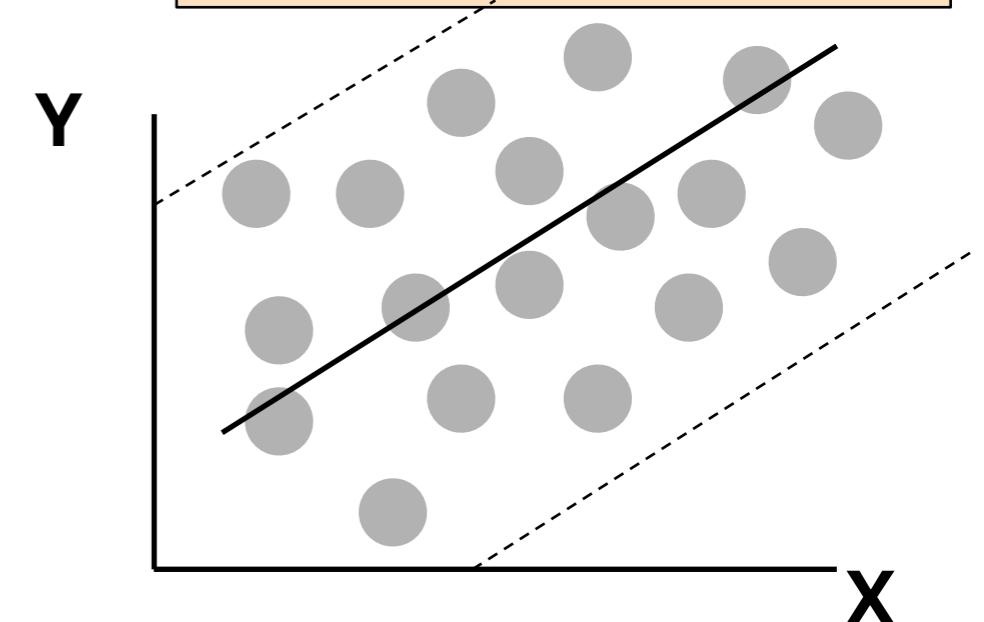
Types of Relationships

(continued)

Strong relationships



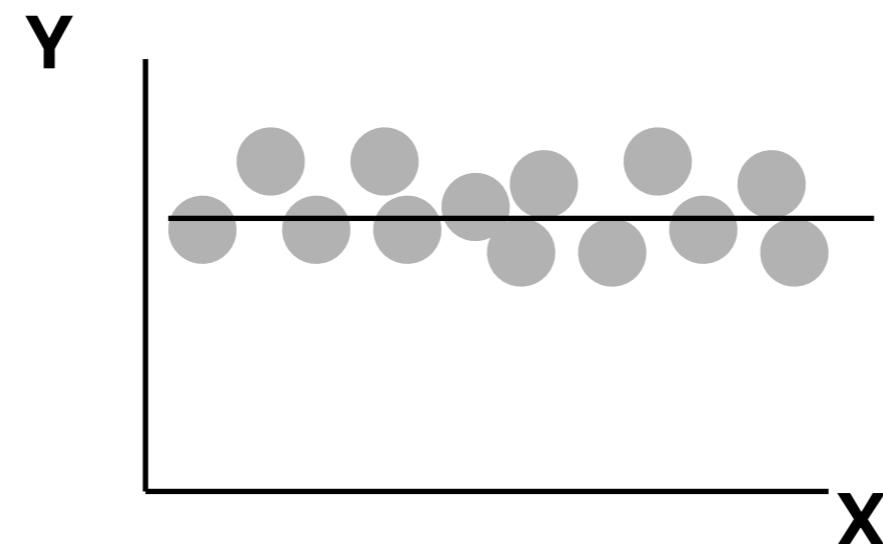
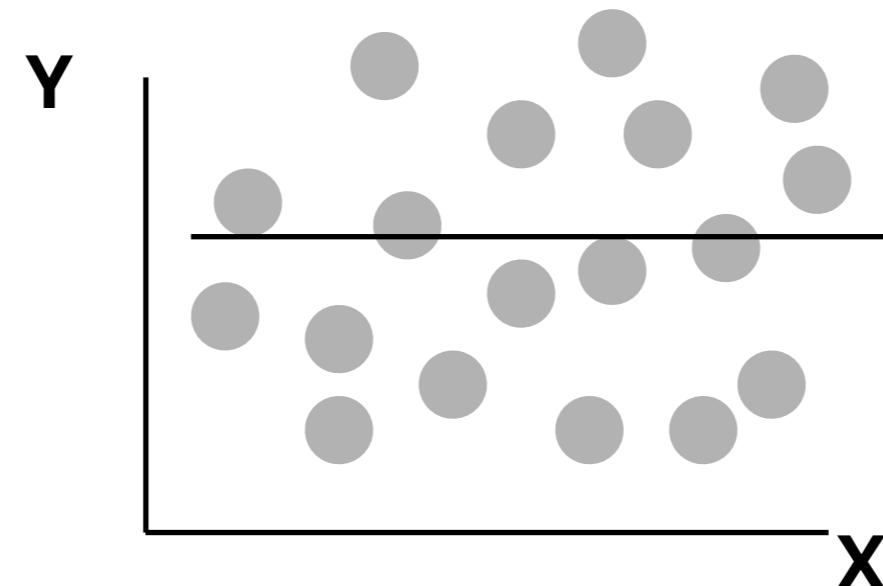
Weak relationships



Types of Relationships

(continued)

No relationship



Simple Linear Regression Conceptual Model

The population regression model: This is a conceptual model, a hypothesis, or a postulation

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

Diagram illustrating the components of the population regression model:

- Dependent Variable: Y_i
- Population Y intercept: β_0
- Population Slope Coefficient: β_1
- Independent Variable: X_i
- Random Error term: ϵ_i

The equation is divided into two main components:

- Linear component:** $\beta_0 + \beta_1 X_i$
- Random Error component:** ϵ_i

Interpretation of the slope and the intercept

$$\beta_0 = E(Y | X = 0) ; \quad \beta_1 = \Delta E(Y|X)/\Delta (X);$$

b_0 is the estimated average value of Y when the value of X is b_0 zero

b_1 is the estimated change in the average value of Y as a result of a one-unit change in X

Units of measurement of X and Y are very important for the correct interpretation of the slope and the intercept

Example: $\widehat{App\ Val} = 165.03 + 6.93 (Lot\ size)$

Predict the app. Value of a house with 10,000 s.f. lot size

$$\widehat{App\ Val} = 165.03 + 6.93 (10) = \$234,330$$

How Good is this prediction?

How Good is the Model's prediction Power?

Total variation is made up of two parts:

$$\mathbf{SST} = \mathbf{SSR} + \mathbf{SSE}$$

Total Sum
of Squares

Regression
Sum of
Squares

Error Sum of
Squares

$$SST = \sum (Y_i - \bar{Y})^2 \quad SSR = \sum (\hat{Y}_i - \bar{Y})^2 \quad SSE = \sum (Y_i - \hat{Y}_i)^2$$

where:

\bar{Y} = Average value of the dependent variable

Y_i = Observed values of the dependent variable

\hat{Y}_i = Predicted value of Y for the given X_i value



SST = total sum of squares

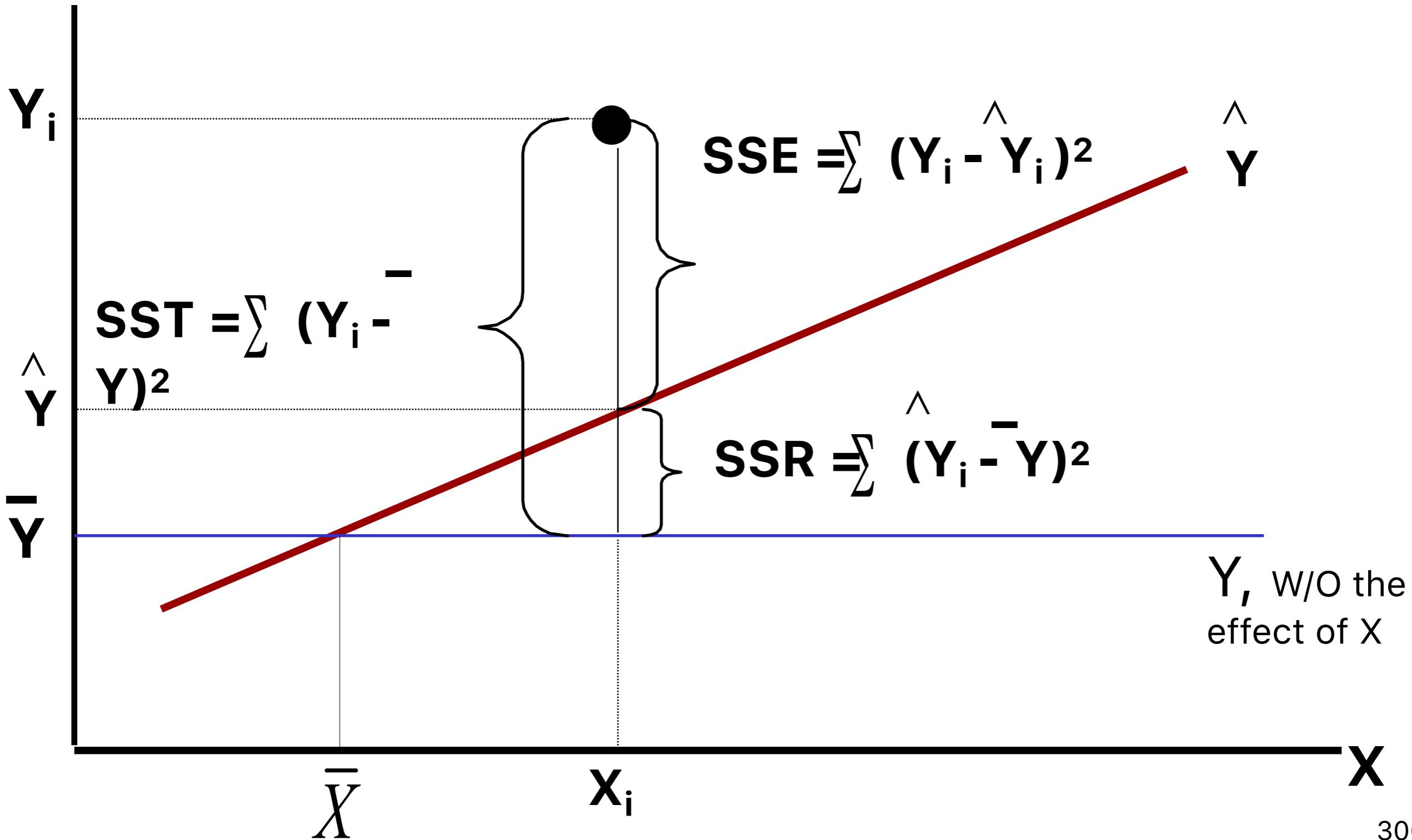
Measures total variation of the Y_i values around their mean

SSR = regression sum of squares (Explained)

Explained portion of total variation attributed to Y's relationship with X

SSE = error sum of squares (Unexplained)

Variation of Y values attributable to other factors than its relationship with X



How Good is the Model's prediction Power?

The **coefficient of determination** is the portion of the total variation in the dependent variable, Y, that is explained by variation in the independent variable, X

The coefficient of determination is also called **r-squared** and is denoted as **r^2**

$$r^2 = \frac{SSR}{SST} = \frac{\text{regression sum of squares}}{\text{total sum of squares}}$$

$$0 \leq r^2 \leq 1$$

Standard Error of Estimate

The standard deviation of the variation of observations
around the regression line is estimated by

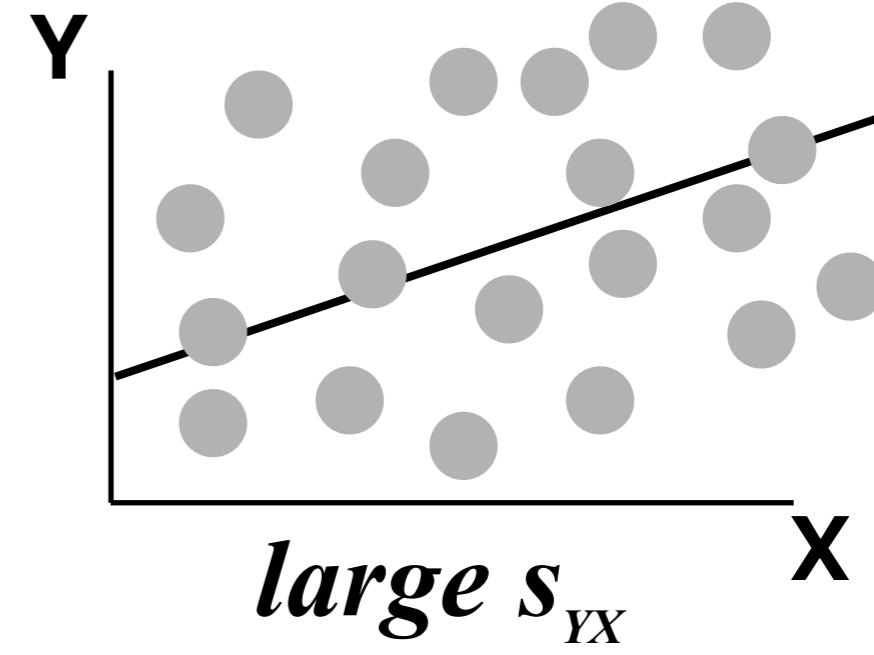
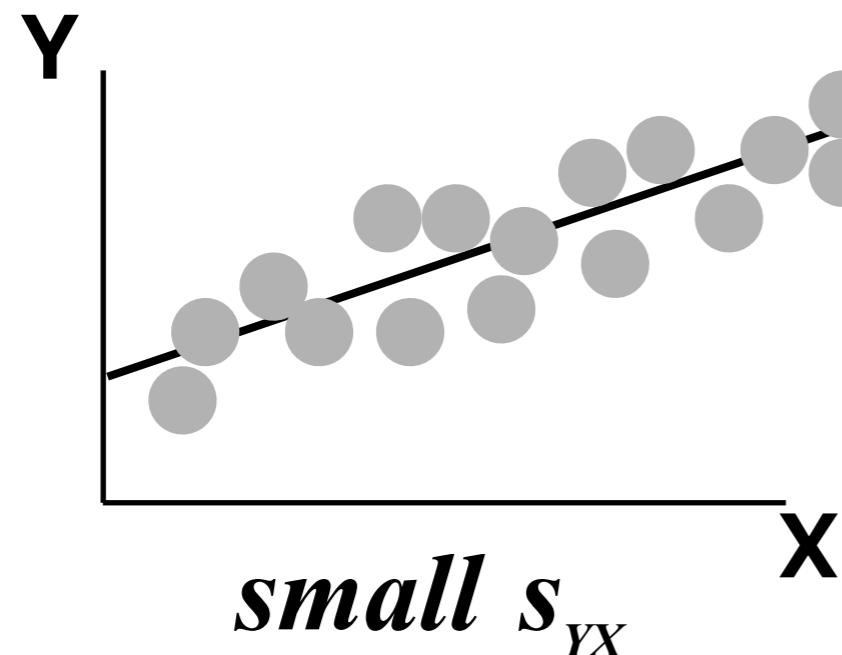
Where SSE = error sum of squares; n = sample size

$$S_{yx} = \sqrt{\frac{SSE}{n - 2}} = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n - 2}} = \sqrt{MSE}$$

The concept is the same as the standard deviation
(average difference) around the mean of a univariate

Comparing Standard Errors

S_{YX} is a measure of the variation of observed Y values from the regression line



The magnitude of S_{YX} should always be judged relative to the size of the Y values in the sample data

i.e., $S_{YX} = \$36.34K$ is moderately small relative to house prices in the \\$200 - \\$300K range (average 215K)

Assumptions of Regression

Normality of Error

Error values (ε) are normally distributed for any given value of X

Homoscedasticity

The probability distribution of the errors has constant variance

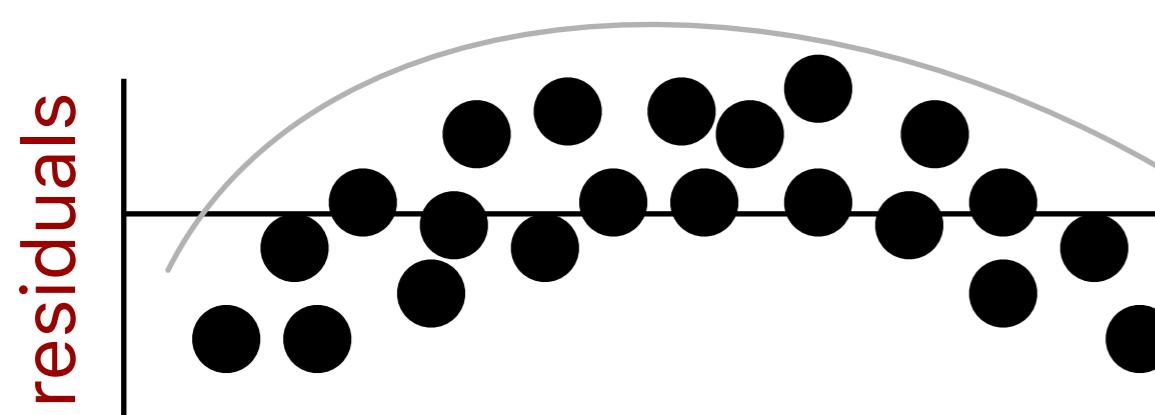
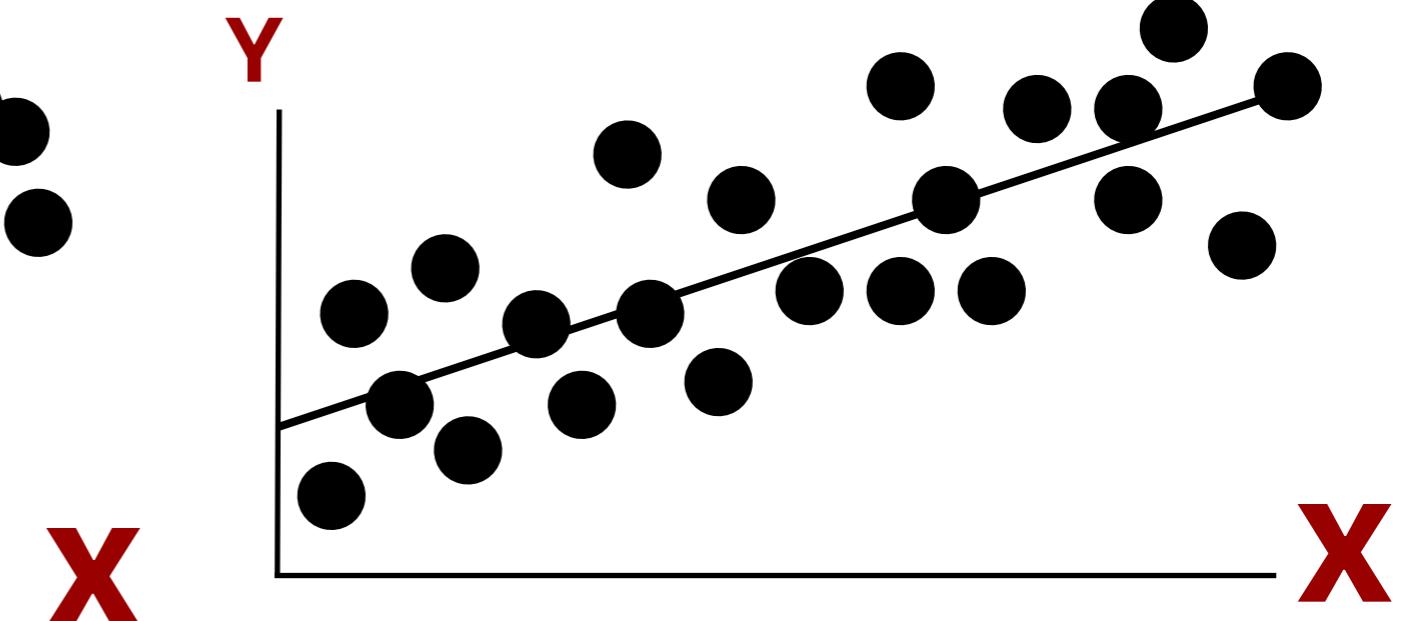
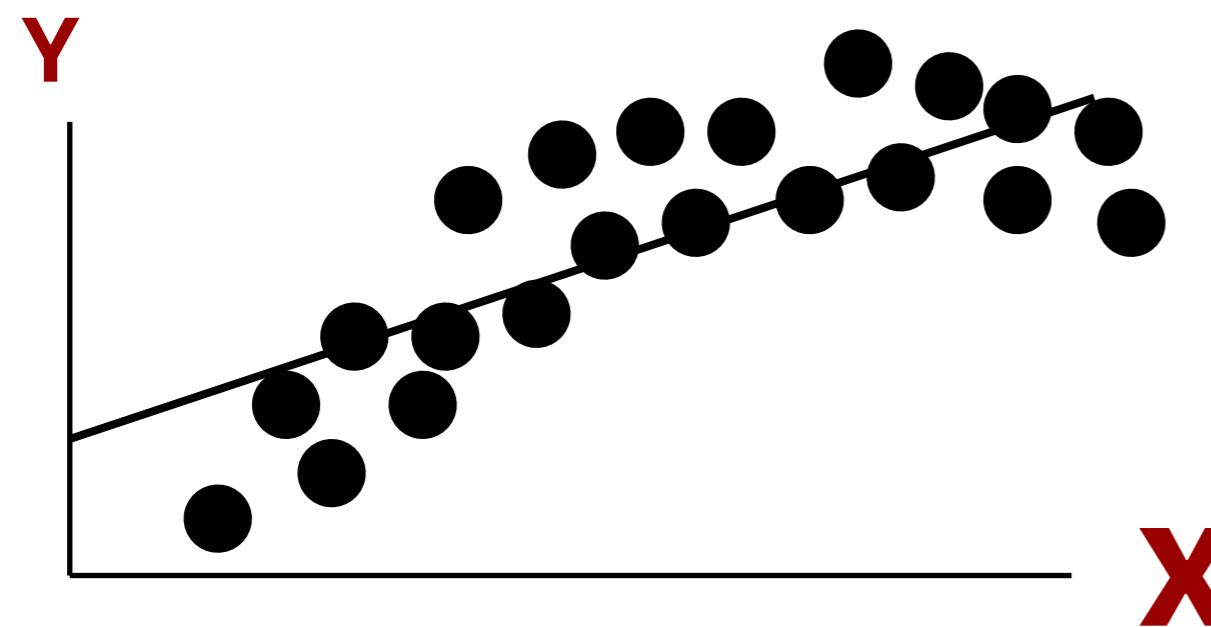
Independence of Errors

Error values are statistically independent

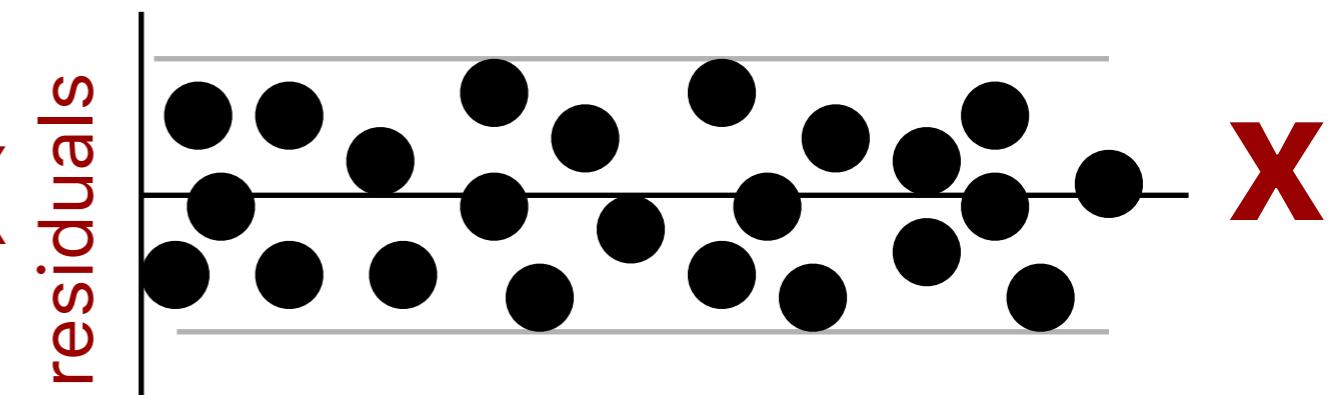
How to investigate the appropriateness of the fitted model

- The residual for observation i , e_i , is the difference between its observed and predicted value;
- Check the assumptions of regression by examining the residuals
$$e_i = Y_i - \hat{Y}_i$$
 - Examine for linearity assumption
 - Examine for constant variance for all levels of X (homoscedasticity)
 - Evaluate normal distribution assumption
 - Evaluate independence assumption
- Graphical Analysis of Residuals
Can plot residuals vs. X

Residual Analysis for Linearity

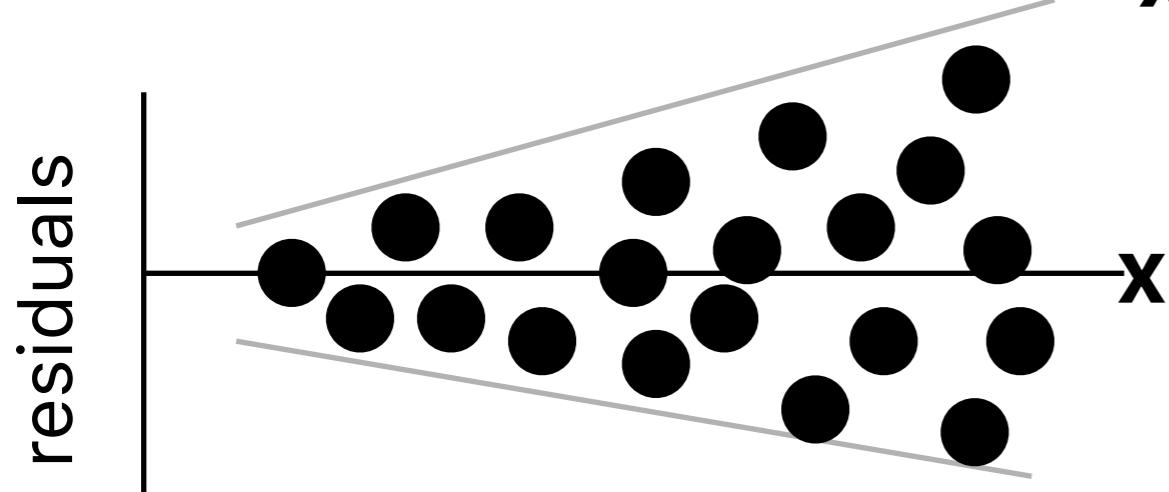
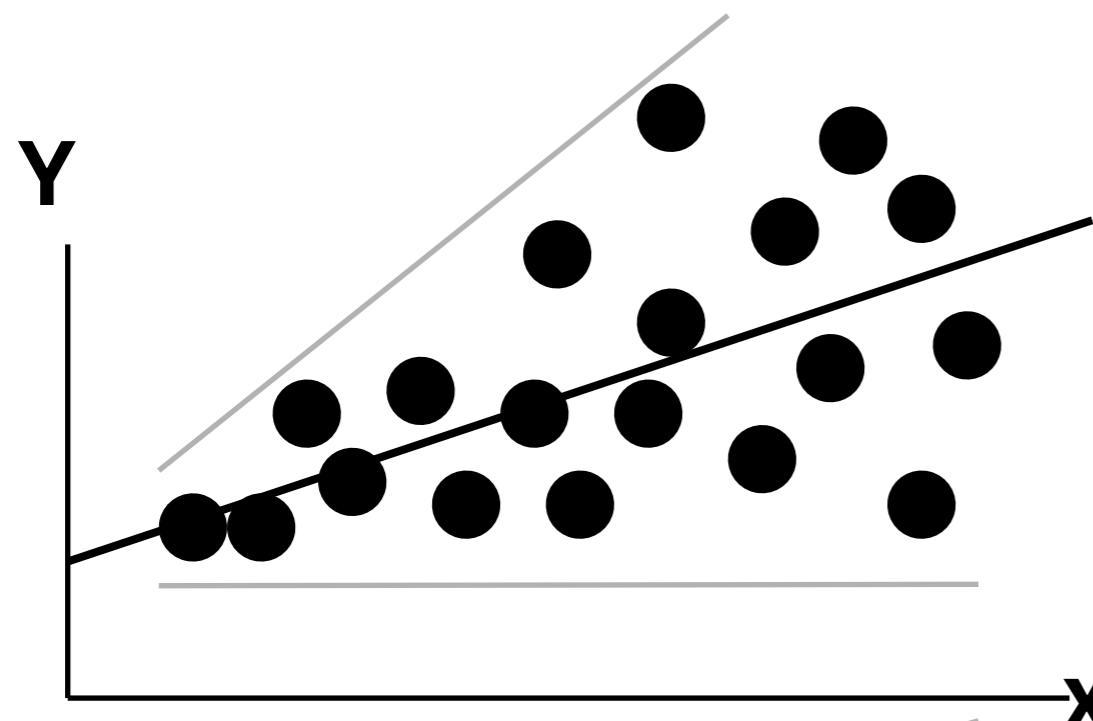


Not Linear

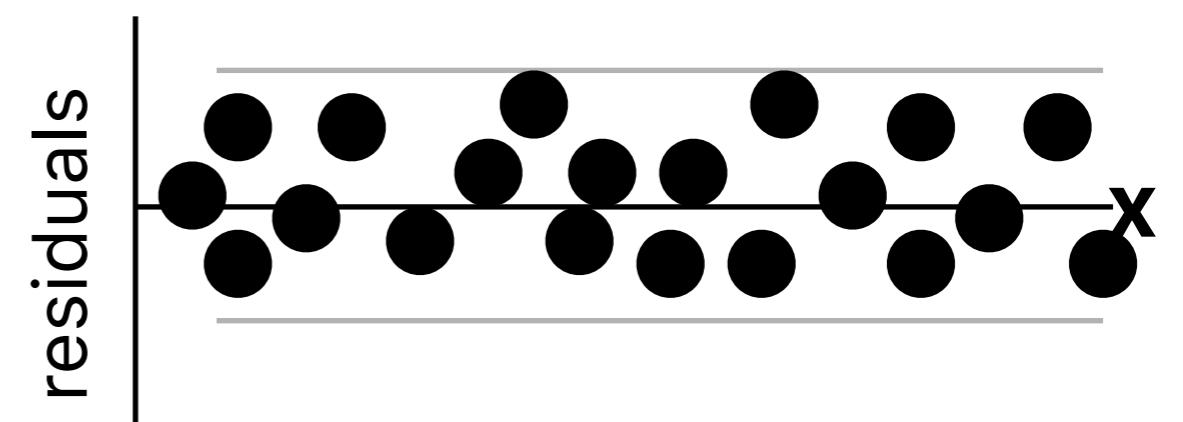
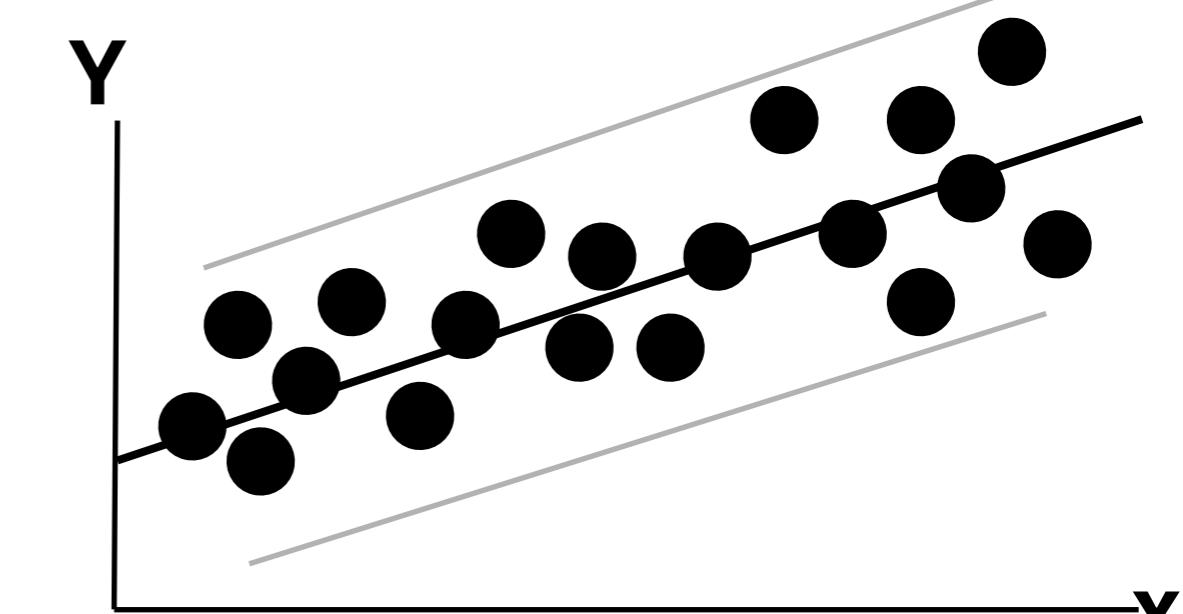


Linear

Residual Analysis for Homoscedasticity

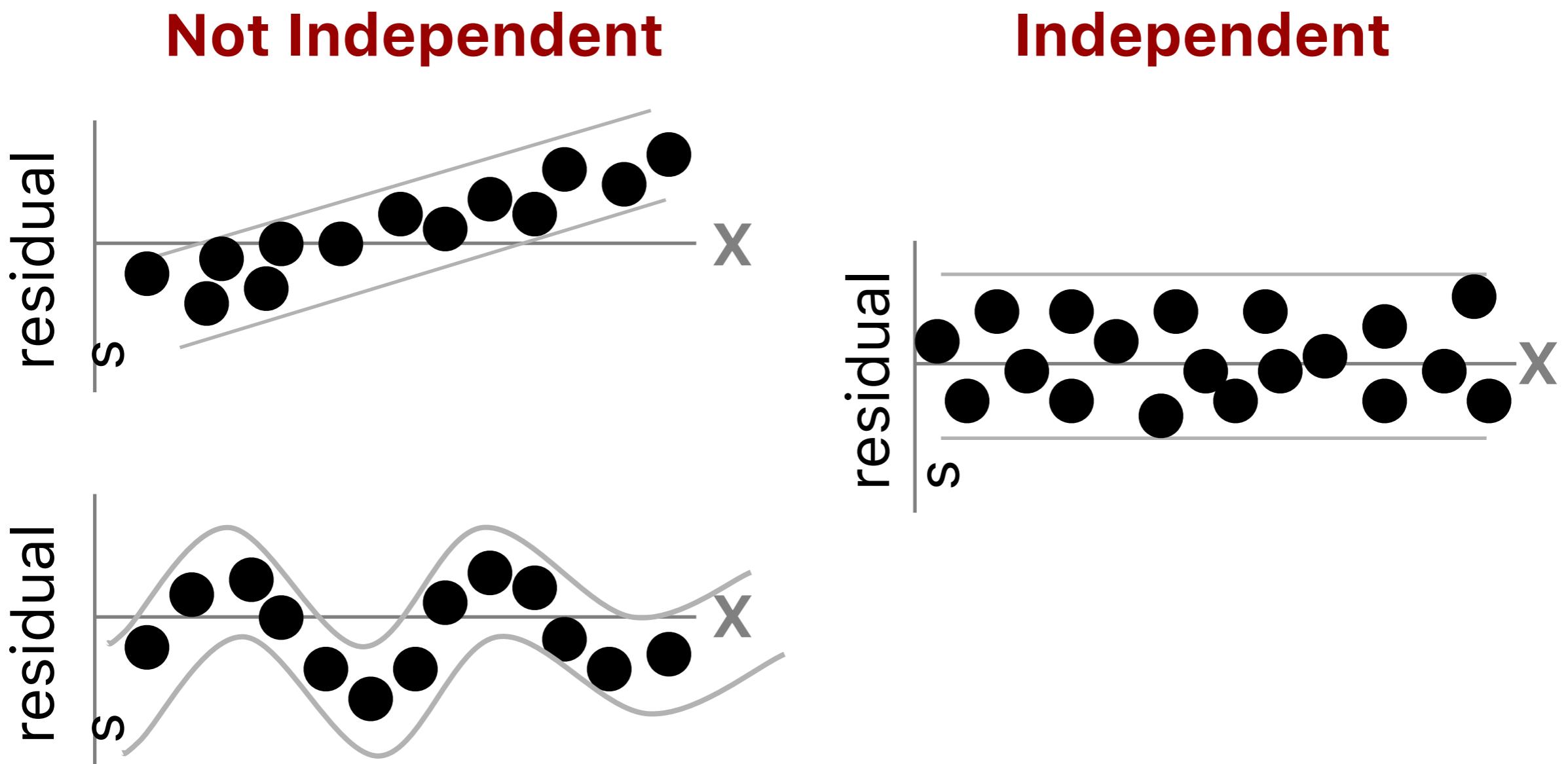


Non-constant
variance



Constant variance

Residual Analysis for Independence



Regression

Linear Regression

Let's first import the data set and see what we're working with.

```
# Import data set  
crime <- read.table("crime_simple.txt", sep = "\t", header = TRUE)
```

The variable names that this data set comes with are very confusing, and even misleading.

R: Crime rate: # of offenses reported to police per million population

Age: The number of males of age 14-24 per 1000 population

S: Indicator variable for Southern states (0 = No, 1 = Yes)

Ed: Mean # of years of schooling x 10 for persons of age 25 or older

Ex0: 1960 per capita expenditure on police by state and local government

Ex1: 1959 per capita expenditure on police by state and local government

LF: Labor force participation rate per 1000 civilian urban males age 14-24

M: The number of males per 1000 females

N: State population size in hundred thousands

NW: The number of non-whites per 1000 population

U1: Unemployment rate of urban males per 1000 of age 14-24

U2: Unemployment rate of urban males per 1000 of age 35-39

W: Median value of transferable goods and assets or family income in tens of \$

X: The number of families per 1000 earning below 1/2 the median income



We really need to give these variables better names

```
# Assign more meaningful variable names
colnames(crime) <- c("crime.per.million", "young.males", "is.south", "average.ed",
                      "exp.per.cap.1960", "exp.per.cap.1959", "labour.part",
                      "male.per.fem", "population", "nonwhite",
                      "unemp.youth", "unemp.adult", "median.assets", "num.low.salary")

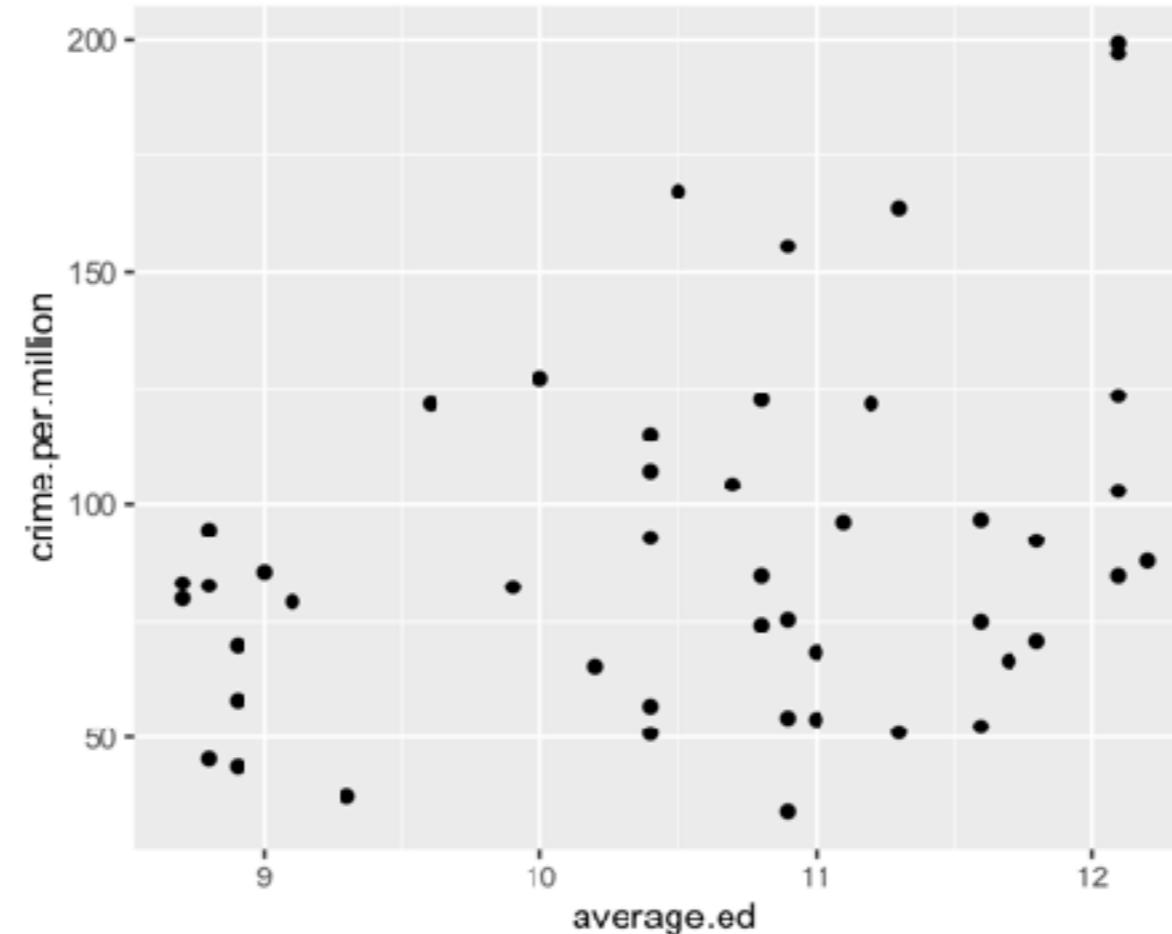
# Convert is.south to a factor
# Divide average.ed by 10 so that the variable is actually average education
# Convert median assets to 1000's of dollars instead of 10's
crime <- transform(crime, is.south = as.factor(is.south),
                     average.ed = average.ed / 10,
                     median.assets = median.assets / 100)

# print summary of the data
summary(crime)
```

```
## crime.per.million young.males is.south average.ed
## Min. : 34.20 Min. :119.0 0:31 Min. : 8.70
## 1st Qu.: 65.85 1st Qu.:130.0 1:16 1st Qu.: 9.75
## Median : 83.10 Median :136.0 Median :10.80
## Mean : 90.51 Mean :138.6 Mean :10.56
## 3rd Qu.:105.75 3rd Qu.:146.0 3rd Qu.:11.45
## Max. :199.30 Max. :177.0 Max. :12.20
## exp.per.cap.1960 exp.per.cap.1959 labour.part male.per.fem
## Min. : 45.0 Min. : 41.00 Min. :480.0 Min. : 934.0
## 1st Qu.: 62.5 1st Qu.: 58.50 1st Qu.:530.5 1st Qu.: 964.5
## Median : 78.0 Median : 73.00 Median :560.0 Median : 977.0
## Mean : 85.0 Mean : 80.23 Mean :561.2 Mean : 983.0
## 3rd Qu.:104.5 3rd Qu.: 97.00 3rd Qu.:593.0 3rd Qu.: 992.0
## Max. :166.0 Max. :157.00 Max. :641.0 Max. :1071.0
## population nonwhite unemp.youth unemp.adult
## Min. : 3.00 Min. : 2.0 Min. : 70.00 Min. : 20.00
## 1st Qu.: 10.00 1st Qu.: 24.0 1st Qu.: 80.50 1st Qu.: 27.50
## Median : 25.00 Median : 76.0 Median : 92.00 Median : 34.00
## Mean : 36.62 Mean :101.1 Mean : 95.47 Mean : 33.98
## 3rd Qu.: 41.50 3rd Qu.:132.5 3rd Qu.:104.00 3rd Qu.: 38.50
## Max. :168.00 Max. :423.0 Max. :142.00 Max. : 58.00
## median.assets num.low.salary
## Min. :2.880 Min. :126.0
## 1st Qu.:4.595 1st Qu.:165.5
## Median :5.370 Median :176.0
## Mean :5.254 Mean :194.0
## 3rd Qu.:5.915 3rd Qu.:227.5
## Max. :6.890 Max. :276.0
```

First step: some plotting and summary statistics

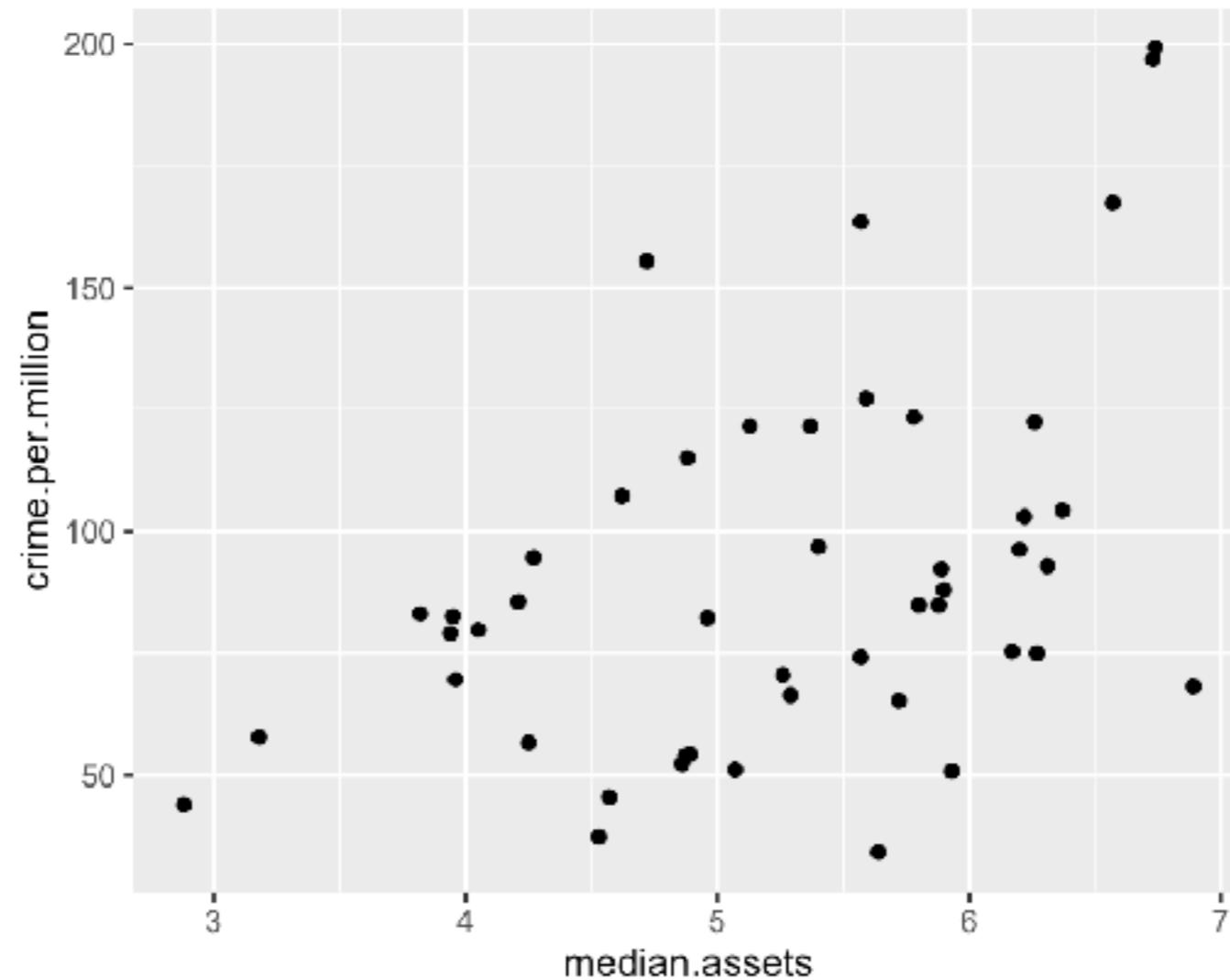
```
# Scatter plot of outcome (crime.per.million) against average.ed  
qplot(average.ed, crime.per.million, data = crime)
```



```
# correlation between education and crime  
with(crime, cor(average.ed, crime.per.million))
```

```
## [1] 0.3228349
```

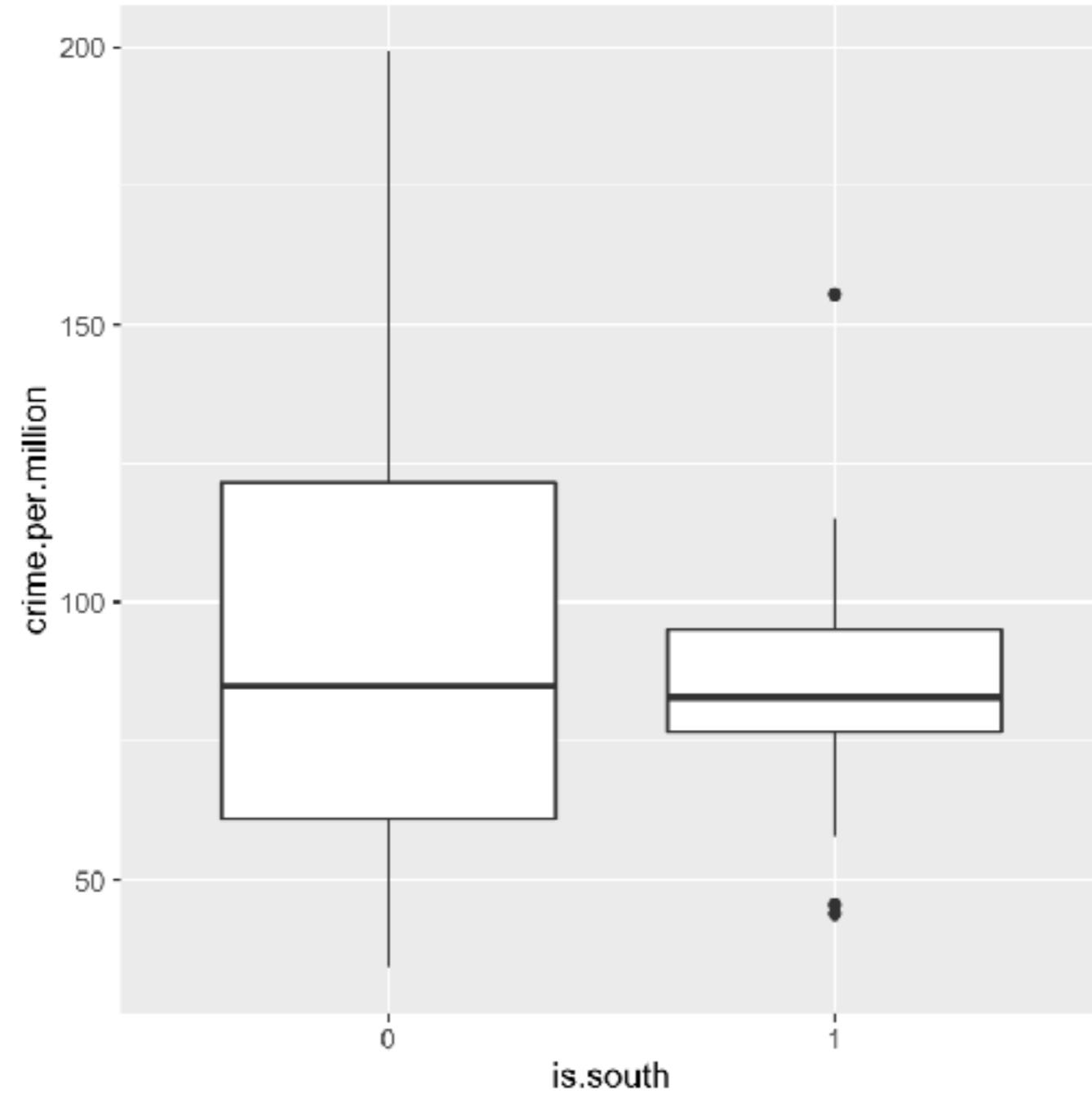
```
# Scatter plot of outcome (crime.per.million) against median.assets  
qplot(median.assets, crime.per.million, data = crime)
```



```
# correlation between education and crime  
with(crime, cor(median.assets, crime.per.million))
```

```
## [1] 0.4413199
```

```
# Boxplots showing crime rate broken down by southern vs non-southern state  
qplot(is.south, crime.per.million, geom = "boxplot", data = crime)
```



Constructing a regression model

```
crime.lm <- lm(crime.per.million ~ ., data = crime)
# Summary of the linear regression model
crime.lm
```

```
##
## Call:
## lm(formula = crime.per.million ~ ., data = crime)
##
## Coefficients:
## (Intercept)      young.males      is.southl      average.ed
## -6.918e+02       1.040e+00      -8.308e+00      1.802e+01
## exp.per.cap.1960 exp.per.cap.1959      labour.part      male.per.fem
## 1.608e+00        -6.673e-01      -4.103e-02      1.648e-01
## population       nonwhite       unemp.youth      unemp.adult
## -4.128e-02       7.175e-03      -6.017e-01      1.792e+00
## median.assets    num.low.salary
## 1.374e+01        7.929e-01
```

```
summary(crime.lm)

##
## Call:
## lm(formula = crime.per.million ~ ., data = crime)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -34.884 -11.923 -1.135 13.495 50.560 
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -6.918e+02  1.559e+02 -4.438 9.56e-05 ***
## young.males  1.040e+00  4.227e-01  2.460  0.01931 *  
## is.south1    -8.308e+00  1.491e+01 -0.557  0.58117    
## average.ed   1.802e+01  6.497e+00  2.773  0.00906 ** 
## exp.per.cap.1960 1.608e+00  1.059e+00  1.519  0.13836    
## exp.per.cap.1959 -6.673e-01  1.149e+00 -0.581  0.56529    
## labour.part   -4.103e-02  1.535e-01 -0.267  0.79087    
## male.per.fem  1.648e-01  2.099e-01  0.785  0.43806    
## population   -4.128e-02  1.295e-01 -0.319  0.75196    
## nonwhite      7.175e-03  6.387e-02  0.112  0.91124    
## unemp.youth   -6.017e-01  4.372e-01 -1.376  0.17798    
## unemp.adult   1.792e+00  8.561e-01  2.093  0.04407 *  
## median.assets 1.374e+01  1.058e+01  1.298  0.20332    
## num.low.salary 7.929e-01  2.351e-01  3.373  0.00191 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.94 on 33 degrees of freedom
## Multiple R-squared:  0.7692, Adjusted R-squared:  0.6783 
## F-statistic: 8.462 on 13 and 33 DF,  p-value: 3.686e-07
```

```
options(scipen=4) # Set scipen = 0 to get back to default

summary(crime.lm)

## 
## Call:
## lm(formula = crime.per.million ~ ., data = crime)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -34.884 -11.923  -1.135  13.495  50.560 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -691.837588 155.887918 -4.438 0.0000956 ***
## young.males    1.039810  0.422708  2.460  0.01931 *  
## is.south1     -8.308313  14.911588 -0.557  0.58117    
## average.ed     18.016011  6.496504  2.773  0.00906 ** 
## exp.per.cap.1960  1.607818  1.058667  1.519  0.13836    
## exp.per.cap.1959 -0.667258  1.148773 -0.581  0.56529    
## labour.part    -0.041031  0.153477 -0.267  0.79087    
## male.per.fem    0.164795  0.209932  0.785  0.43806    
## population     -0.041277  0.129516 -0.319  0.75196    
## nonwhite       0.007175  0.063867  0.112  0.91124    
## unemp.youth     -0.601675  0.437154 -1.376  0.17798    
## unemp.adult      1.792263  0.856111  2.093  0.04407 *  
## median.assets   13.735847 10.583028  1.298  0.20332    
## num.low.salary   0.792933  0.235085  3.373  0.00191 ** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 21.94 on 33 degrees of freedom
## Multiple R-squared:  0.7692, Adjusted R-squared:  0.6783 
## F-statistic: 8.462 on 13 and 33 DF,  p-value: 0.0000003686
```

Exploring the lm object

What kind of output do we get when we run a linear model (lm) in R?

```
# List all attributes of the linear model  
attributes(crime.lm)
```

```
## $names  
## [1] "coefficients"    "residuals"        "effects"          "rank"  
## [5] "fitted.values"   "assign"           "qr"               "df.residual"  
## [9] "contrasts"       "xlevels"          "call"              "terms"  
## [13] "model"  
##  
## $class  
## [1] "lm"
```

```
# coefficients  
crime.lm$coef
```

```
##          (Intercept)      young.males      is.southl      average.ed  
## -691.837587905  1.039809653 -8.308312889  18.016010601  
## exp.per.cap.1960 exp.per.cap.1959      labour.part      male.per.fem  
##  1.607818377   -0.667258285 -0.041031047  0.164794968  
## population      nonwhite      unemp.youth      unemp.adult  
## -0.041276887    0.007174688 -0.601675298  1.792262901  
## median.assets  num.low.salary  
## 13.735847285   0.792932786
```

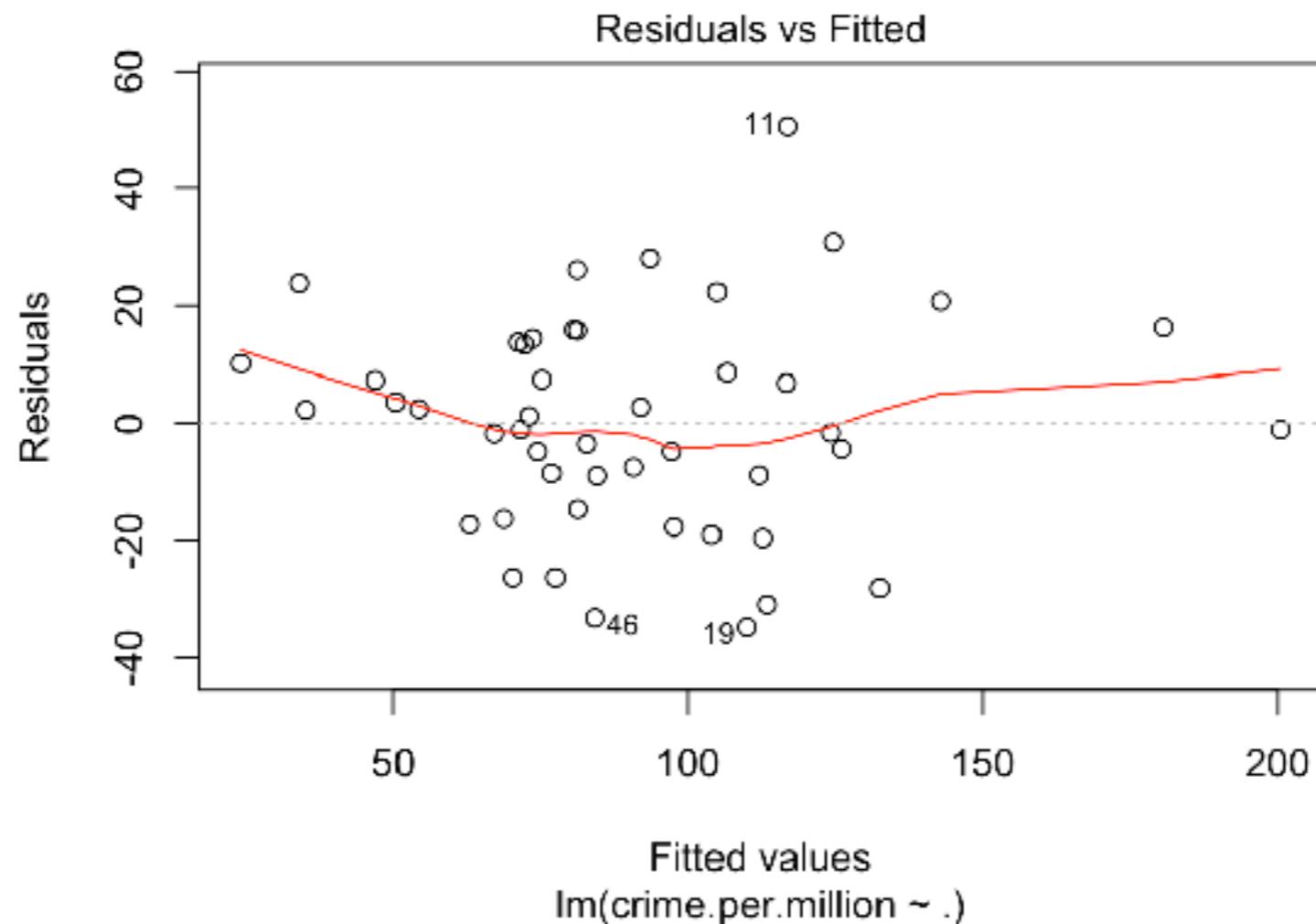
None of the attributes seem to give you p-values. Here's what you can do to get a table that allows you to extract p-values.

```
# Pull coefficients element from summary(lm) object  
round(summary(crime.lm)$coef, 3)
```

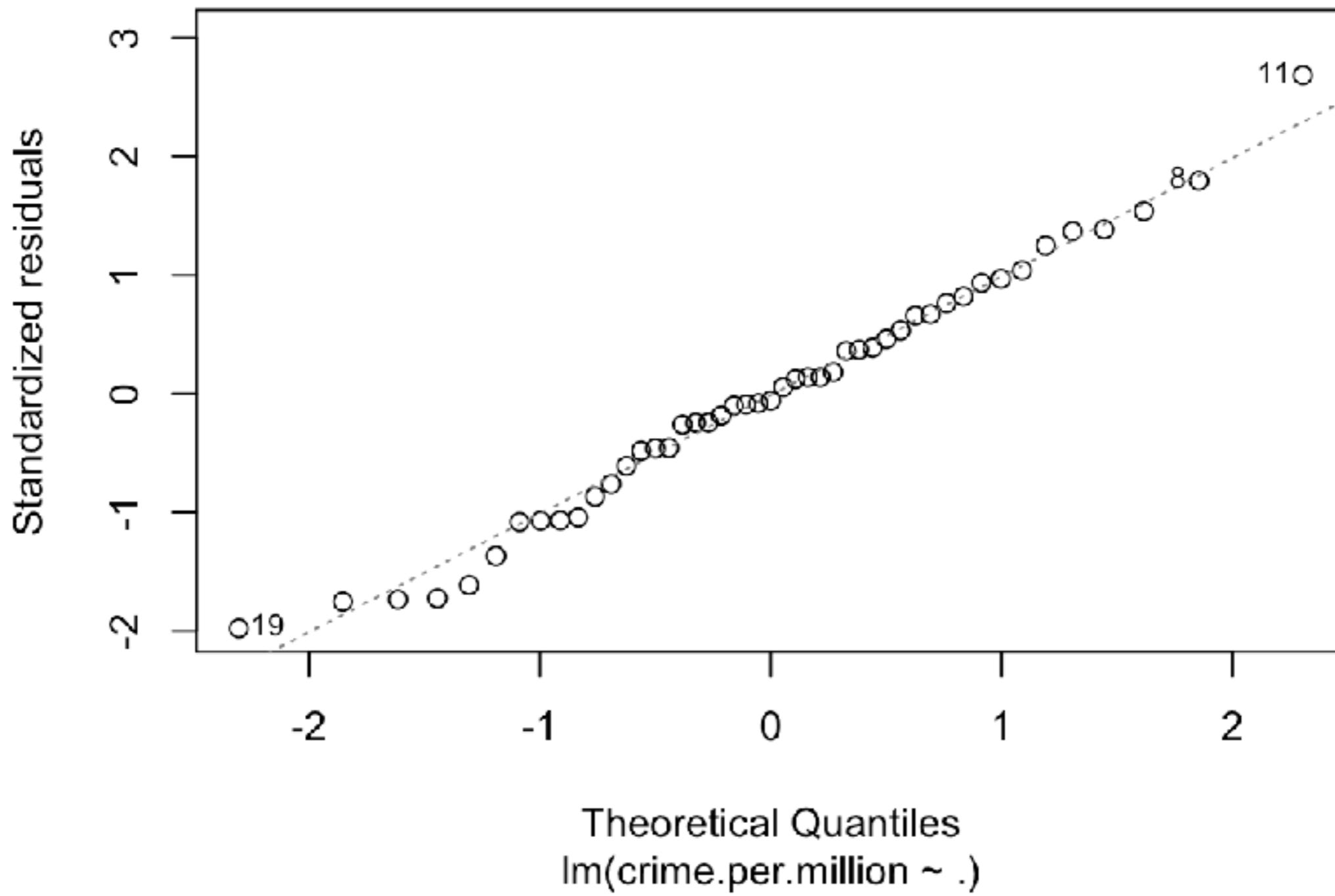
	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	-691.838	155.888	-4.438	0.000
## young.males	1.040	0.423	2.460	0.019
## is.south1	-8.308	14.912	-0.557	0.581
## average.ed	18.016	6.497	2.773	0.009
## exp.per.cap.1960	1.608	1.059	1.519	0.138
## exp.per.cap.1959	-0.667	1.149	-0.581	0.565
## labour.part	-0.041	0.153	-0.267	0.791
## male.per.fem	0.165	0.210	0.785	0.438
## population	-0.041	0.130	-0.319	0.752
## nonwhite	0.007	0.064	0.112	0.911
## unemp.youth	-0.602	0.437	-1.376	0.178
## unemp.adult	1.792	0.856	2.093	0.044
## median.assets	13.736	10.583	1.298	0.203
## num.low.salary	0.793	0.235	3.373	0.002

Plotting the lm object

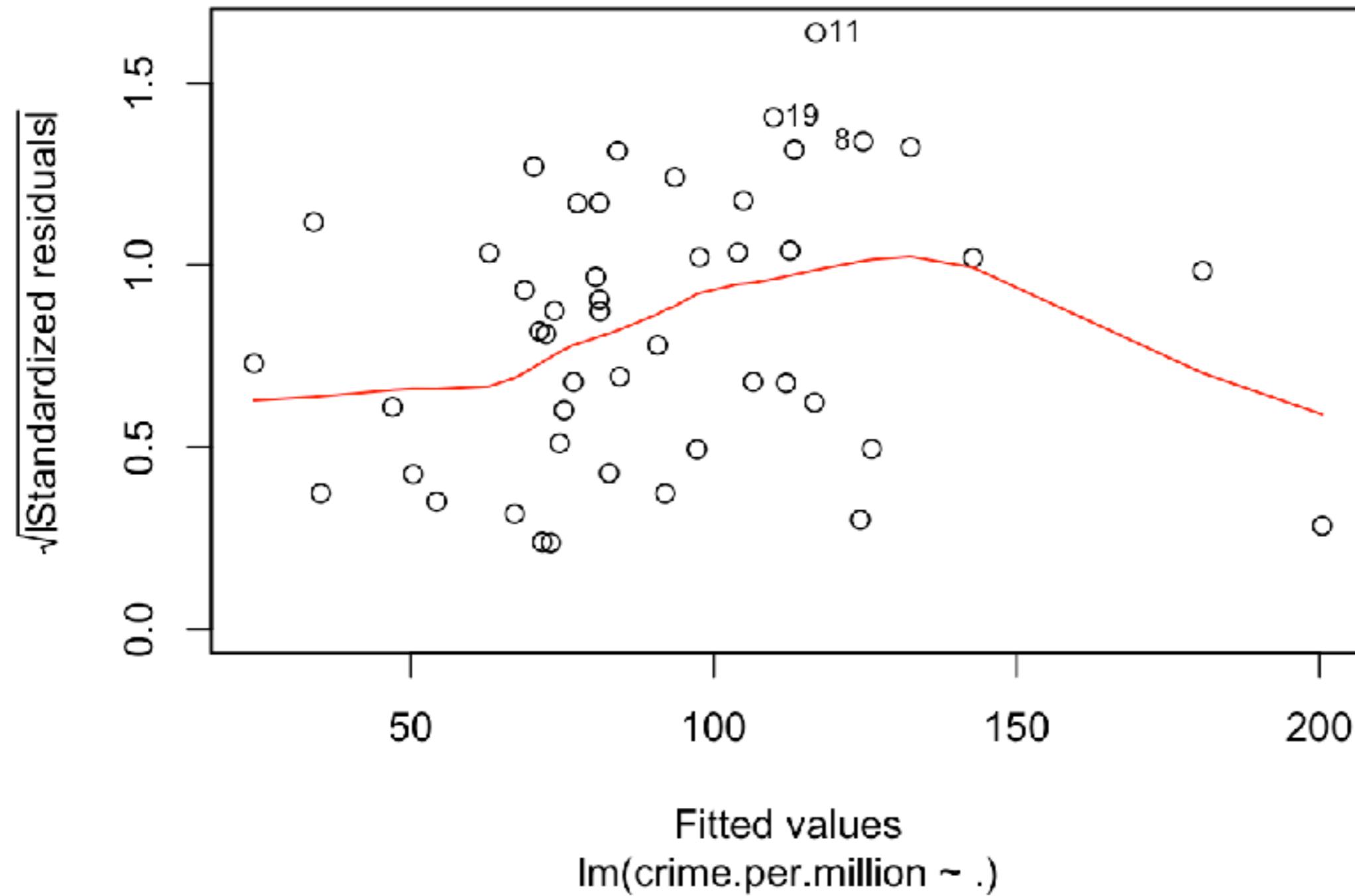
```
plot(crime.lm)
```



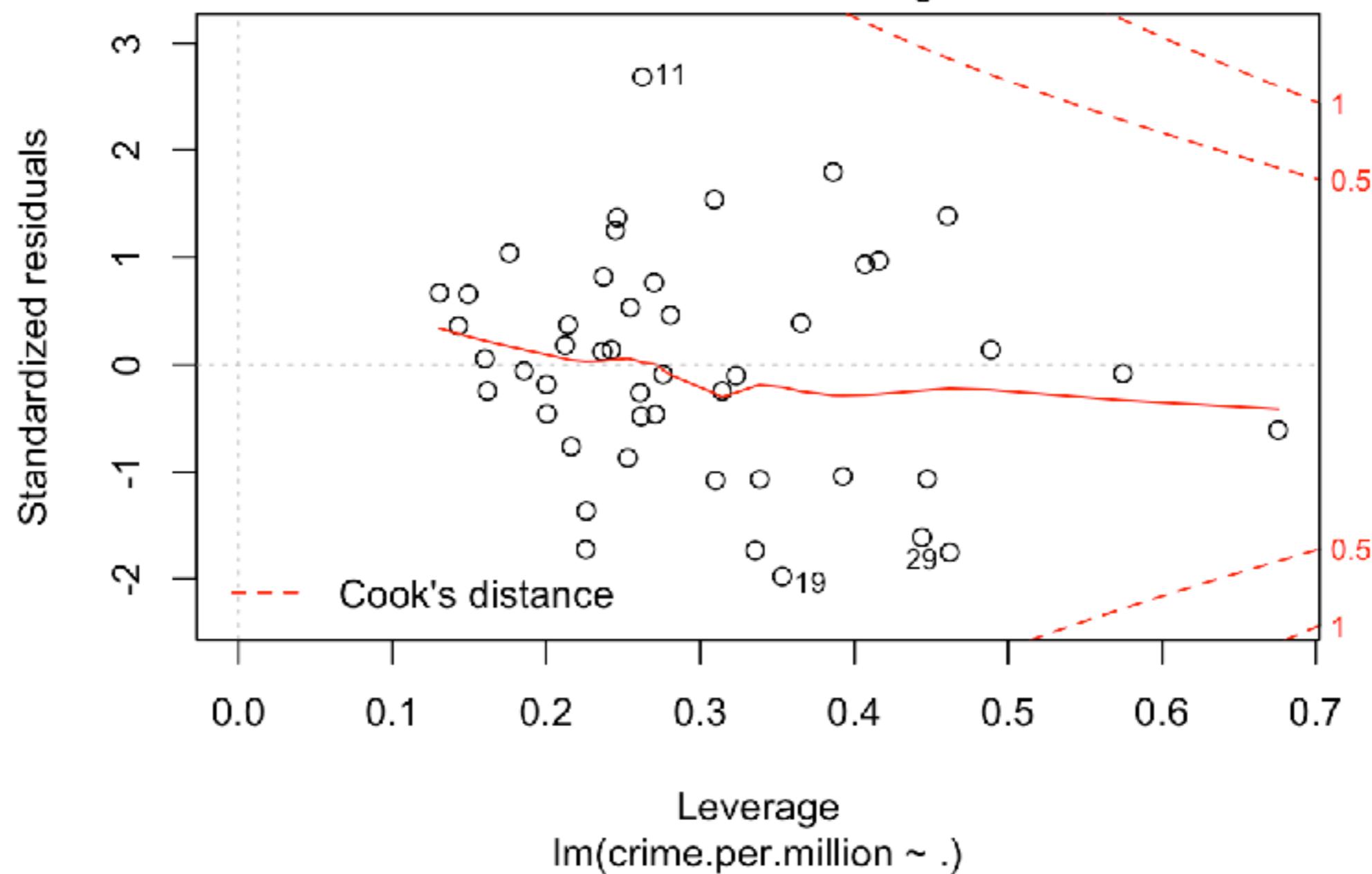
Normal Q-Q



Scale-Location



Residuals vs Leverage

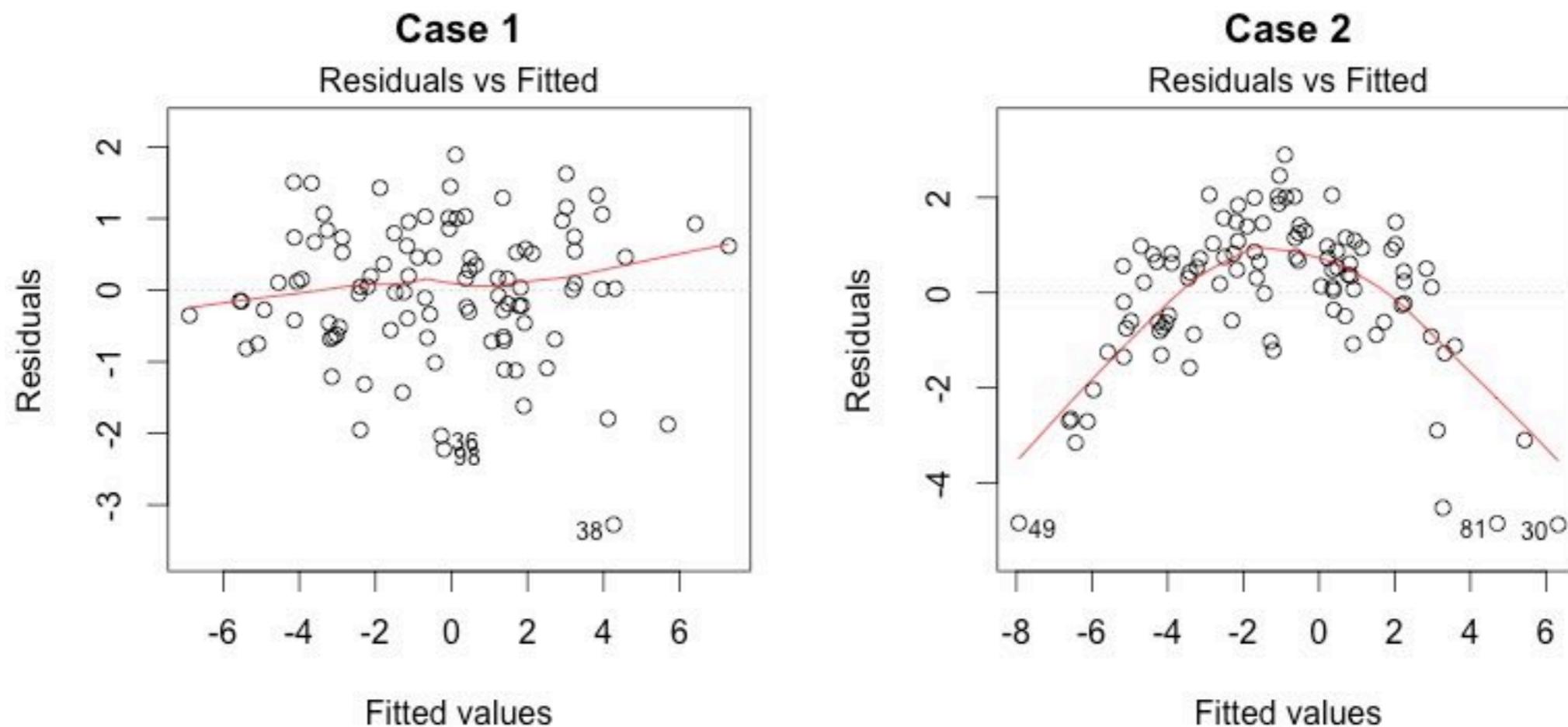


Residuals vs Fitted Plot

Residuals vs. Fitted When a linear model is appropriate, we expect

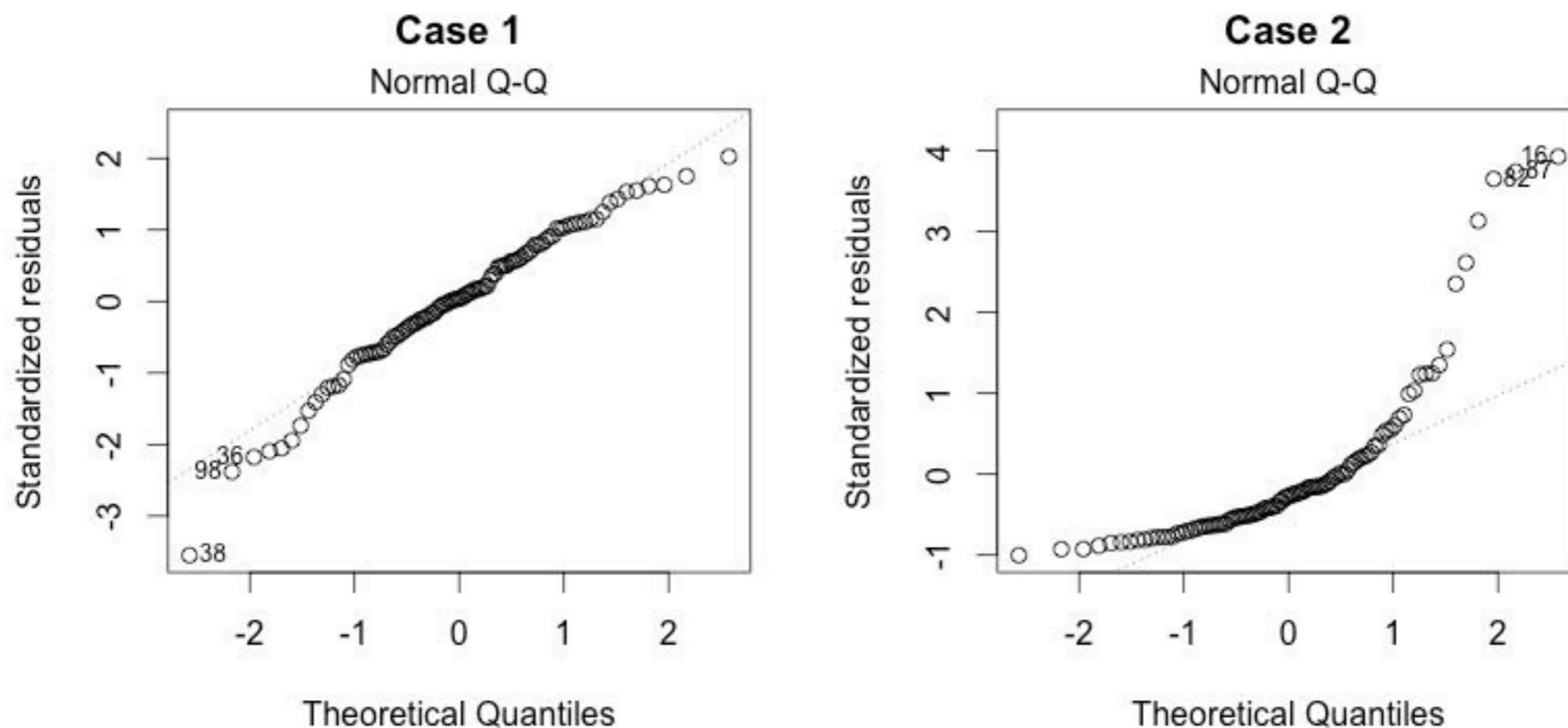
- 1 the residuals will have constant variance when plotted against fitted values; and
- 2 the residuals and fitted values will be uncorrelated.

If there are clear trends in the residual plot, or the plot looks like a funnel, these are clear indicators that the given linear model is inappropriate.



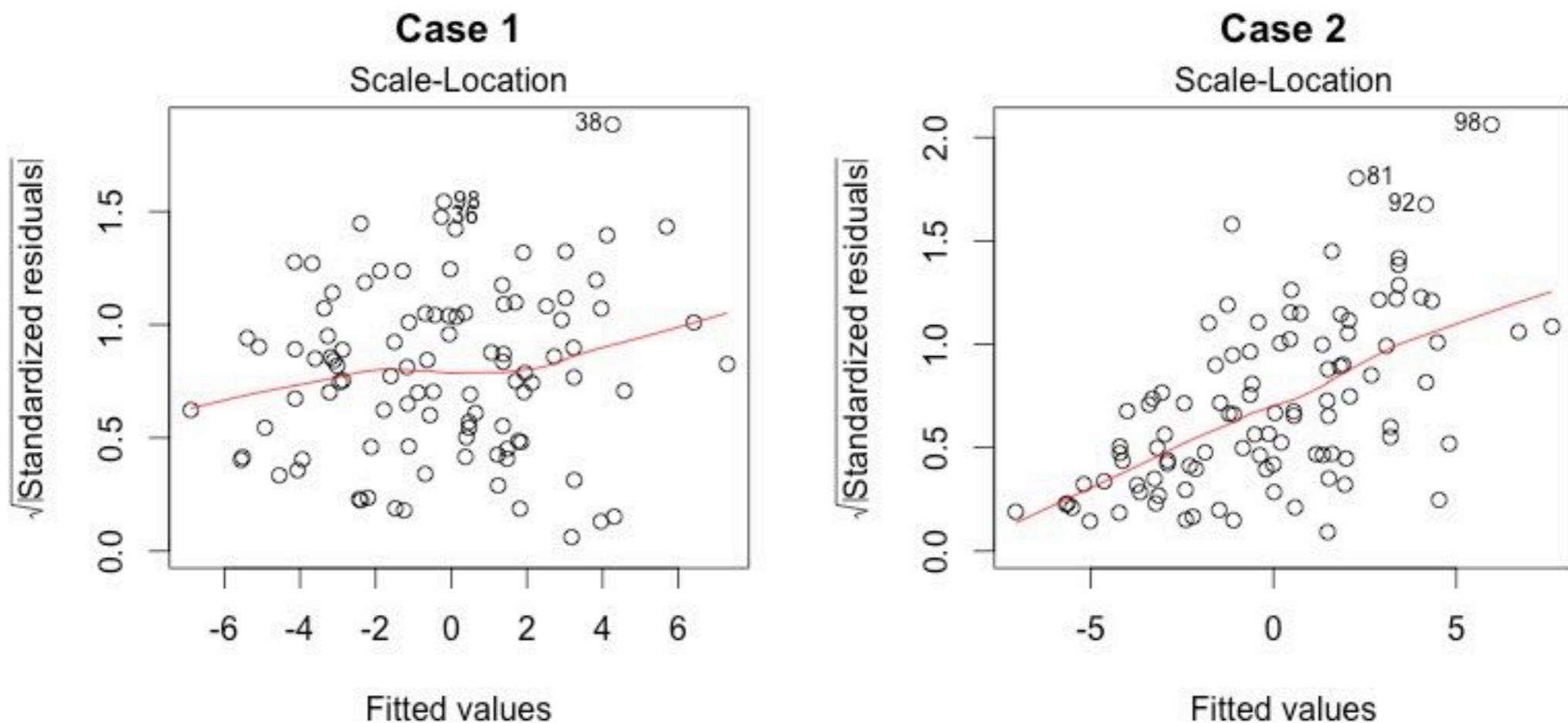
Normal QQ Plot

Normal QQ plot You can use a linear model for prediction even if the underlying normality assumptions don't hold. However, in order for the p-values to be believable, the residuals from the regression must look approximately normally distributed.



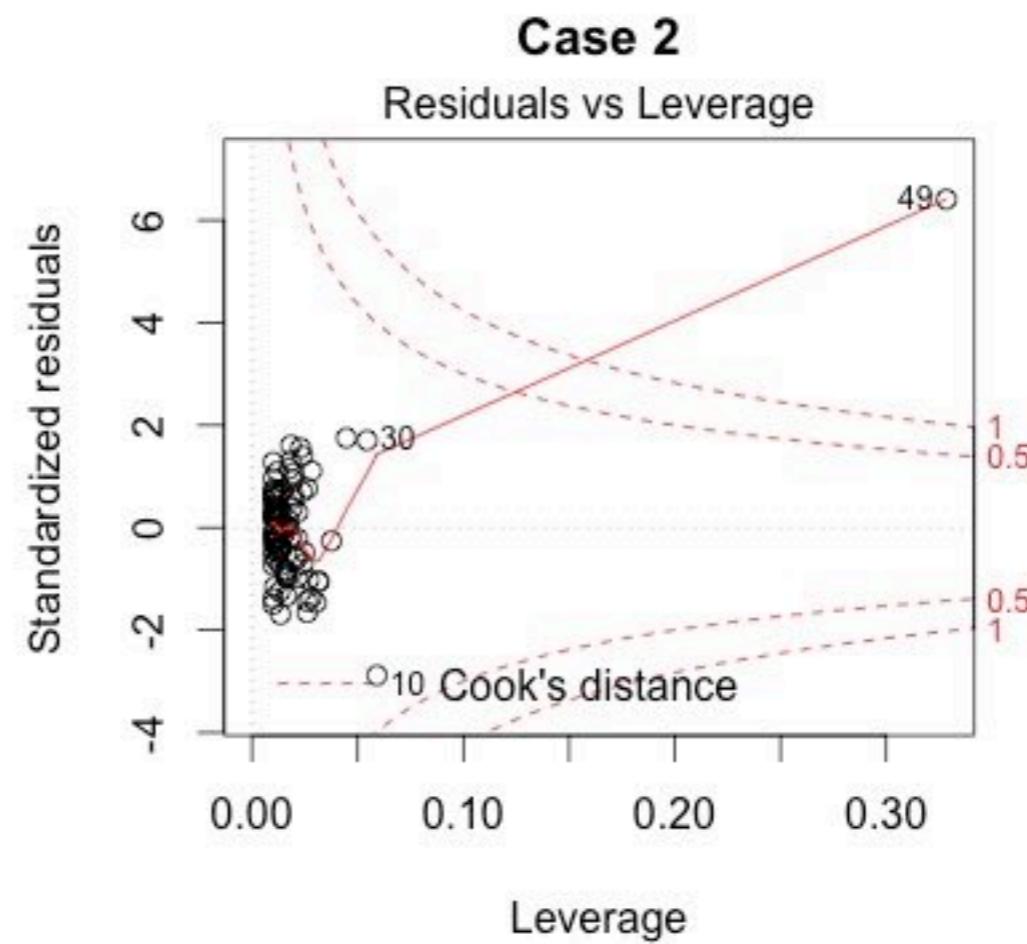
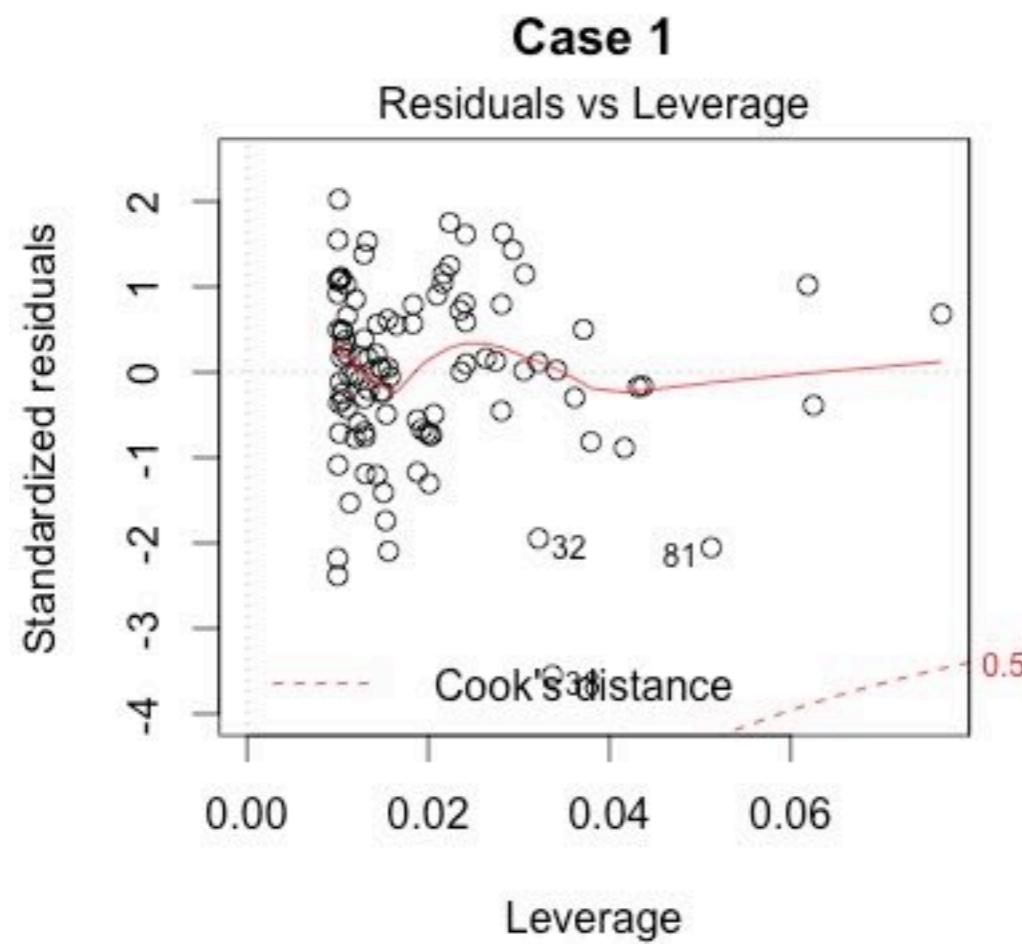
Scale-Location Plot

Scale-location plot This is another version of the residuals vs fitted plot. There should be no discernible trends in this plot.



Residuals vs Leverage

Residuals vs Leverage. Leverage is a measure of how much an observation influenced the model fit. It's a one-number summary of how different the model fit would be if the given observation was excluded, compared to the model fit where the observation is included. Points with high residual (poorly described by the model) and high leverage (high influence on model fit) are outliers. They're skewing the model fit away from the rest of the data, and don't really seem to fit with the rest of the data.



Workshop 9: Regression

We'll begin by loading some packages.

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

Interaction terms in regression

```
# Building up the familiar birthwt data...  
  
# 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"))))  
)
```

Workshop 9 : Regression

- (a) Run a linear regression to better understand how birthweight varies with the mother's age and smoking status (do not include interaction terms).
- (b) What is the coefficient of mother.age in your regression? How do you interpret this coefficient?
- (c) How many coefficients are estimated for the mother's smoking status variable? How do you interpret these coefficients?

Classification

Data

- Before one can present and interpret information, there has to be a process of gathering and sorting data.
- Definition of data is "**facts or figures from which conclusions can be drawn**".
- Usually we collect many measurement on a person or object. Each measurement we call "**Variable**" and each person or object we call "**Observation**".

Types of Data

1. Categorical Data (Qualitative)

I. Nominal

II. Ordinal

2. Numerical Data

1. Interval/Ratio (Scale, Parametric, Quantitative)

1. Discrete : Whole Number

2. Continuous : Fraction

2. Ratio : True Zero

3. Interval : False Zero

Getting data in R

R can import data from practically everywhere

- CSV
- excel
- SPSS
- SAS
- Stata
- SQL
- XML
- JSON

Recap import data in R

```
# first row contains variable names, comma is separator  
# assign the variable id to row names  
# note the / instead of \ on mswindows systems  
  
mydata <- read.table("c:/mydata.csv", header=TRUE,  
                      sep=",", row.names="id")
```

Data Frame

Function

Keep first line
as a header

Field
Separation

Filename to
import

Getting Data

Iris Data Set from UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Iris>)

Attribute Information:

1. Sepal Length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm
5. Classes:
 - Iris Setosa
 - Iris Versicolour
 - Iris Virginica



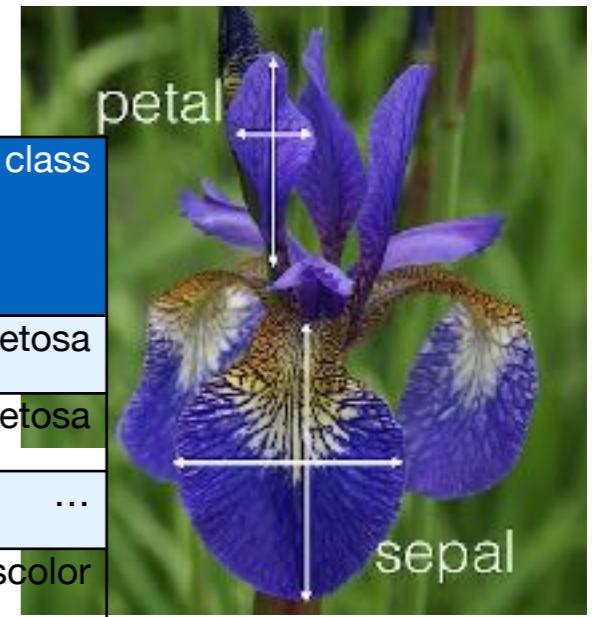
Nomenclature

IRIS

<https://archive.ics.uci.edu/ml/datasets/Iris>

Instances (samples, observations)

	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
...
50	6.4	3.2	4.5	1.5	veriscolor
...
150	5.9	3.0	5.1	1.8	virginica



Features (attributes, dimensions, variables)

Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\mathbf{y} = [y_1, y_2, y_3, \dots, y_N]$$

Iris setosa



Iris virginica



Iris versicolor



Getting Data

```
> iris <- read.csv("iris.data.csv", header=TRUE)

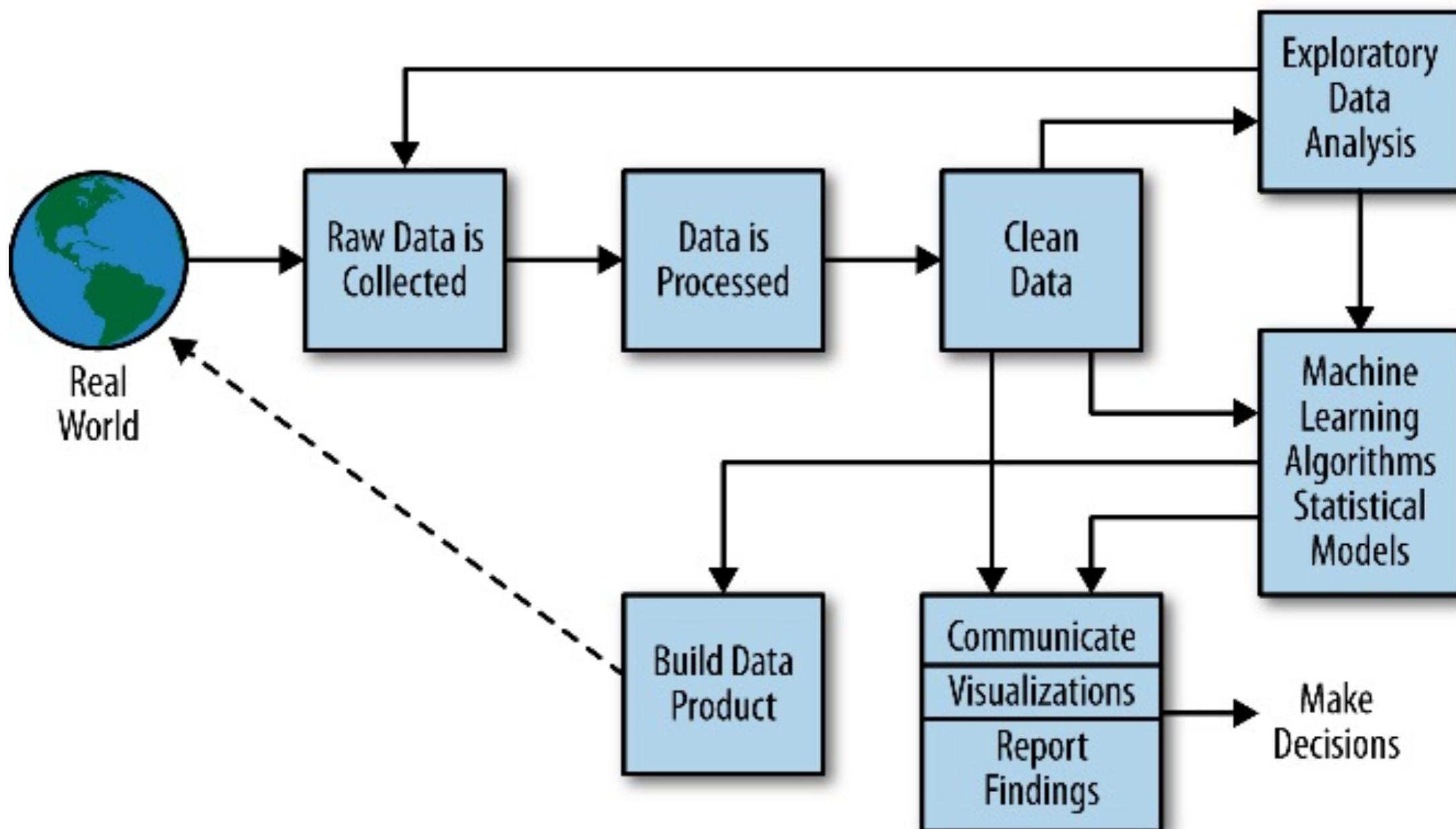
> library(datasets)

> iris

> colnames(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length",
  "Petal.Width", "Species")
```

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1        3.5         1.4        0.2   setosa
2          4.9        3.0         1.4        0.2   setosa
3          4.7        3.2         1.3        0.2   setosa
4          4.6        3.1         1.5        0.2   setosa
5          5.0        3.6         1.4        0.2   setosa
6          5.4        3.9         1.7        0.4   setosa
> nrow(iris)
[1] 150
> table(iris$Species)

setosa versicolor virginica
      50        50        50
>
```



Exploratory Data Analysis (EDA)

- The goal during EDA is to develop an understanding of your data.
- The easiest way to do this is to use questions as tools to guide your investigation.
- When you ask a question, the question focuses your attention on a specific part of your dataset and helps you decide which graphs, models, or transformations to make.
- Two types of questions will always be useful for making discoveries within your data.
 - 1.What type of variation occurs within my variables?
 - 2.What type of covariation occurs between my variables?

Variation

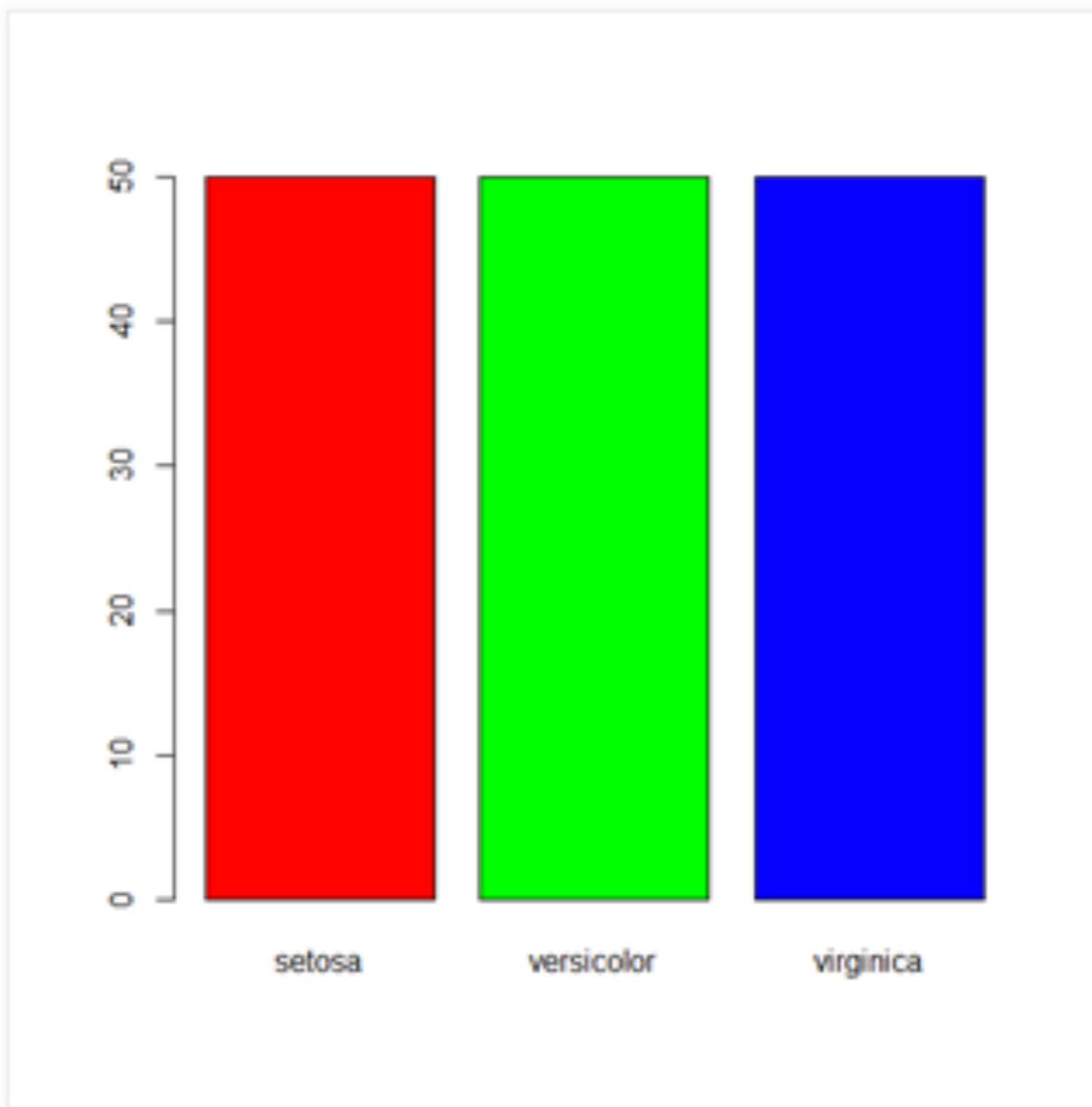
- The tendency of the values of a variable to change from measurement to measurement. You can see variation easily in real life.
- If you measure any continuous variable twice, you will get two different results. This is true even if you measure quantities that are constant, like the speed of light. Categorical variables can also vary if you measure across different subjects or different times.
- Every variable has its own pattern of variation, which can reveal interesting information. The best way to understand that pattern is to visualise the distribution of the variable's values.

Visualizing distributions

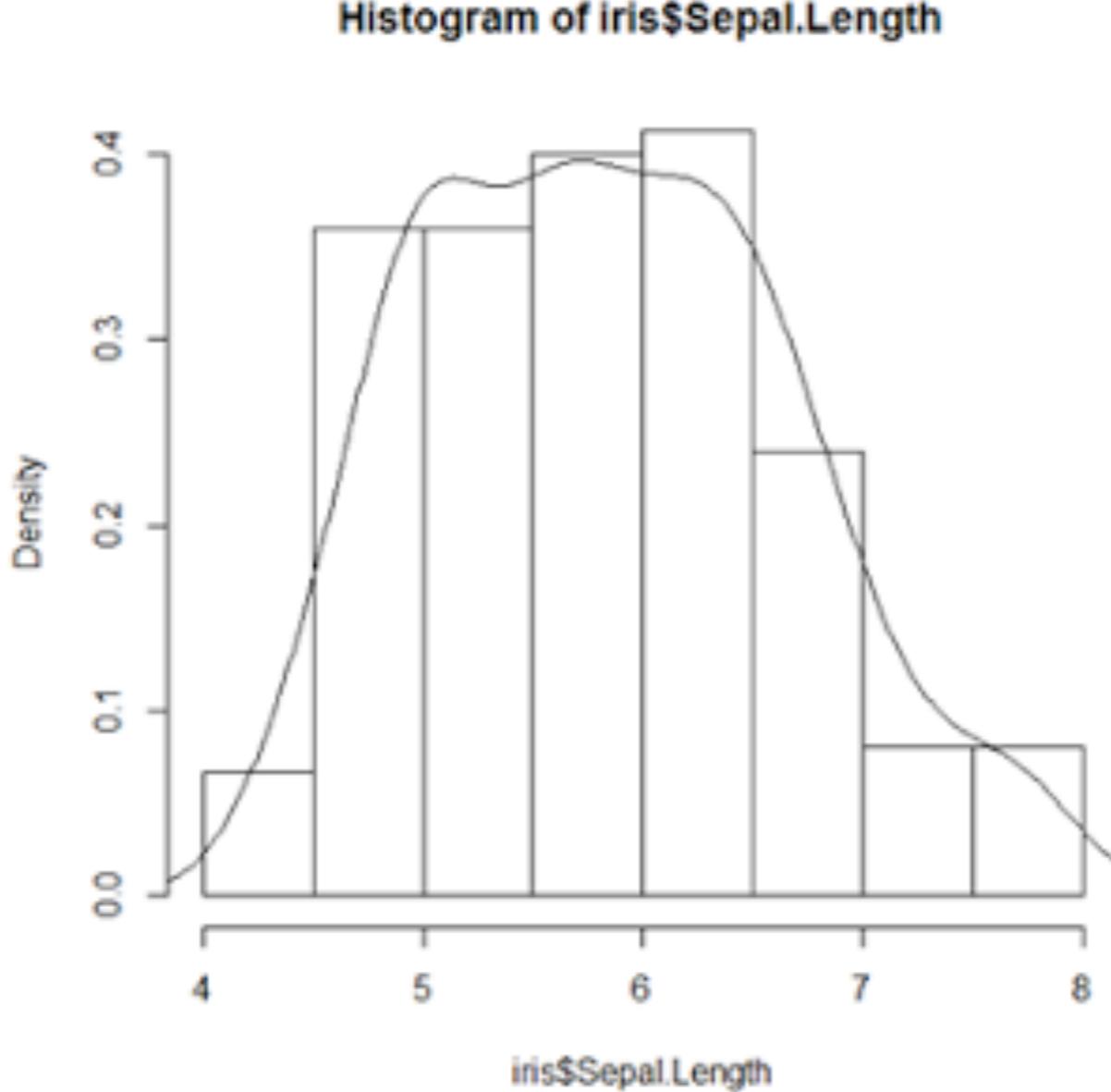
How you visualize the distribution of a variable will depend on whether the variable is categorical or continuous.

- A variable is **categorical** if it can only take one of a small set of values. In R, **categorical variables** are usually saved as factors or character vectors. To examine the distribution of a categorical variable, **use a bar chart**.
- A variable is **continuous** if it can take any of an infinite set of ordered values. Numbers and date-times are two examples of continuous variables. To examine the distribution of a **continuous** variable, **use a histogram**:

```
> categories <- table(iris$Species)
> barplot(categories, col=c('red', 'green', 'blue'))
>
```

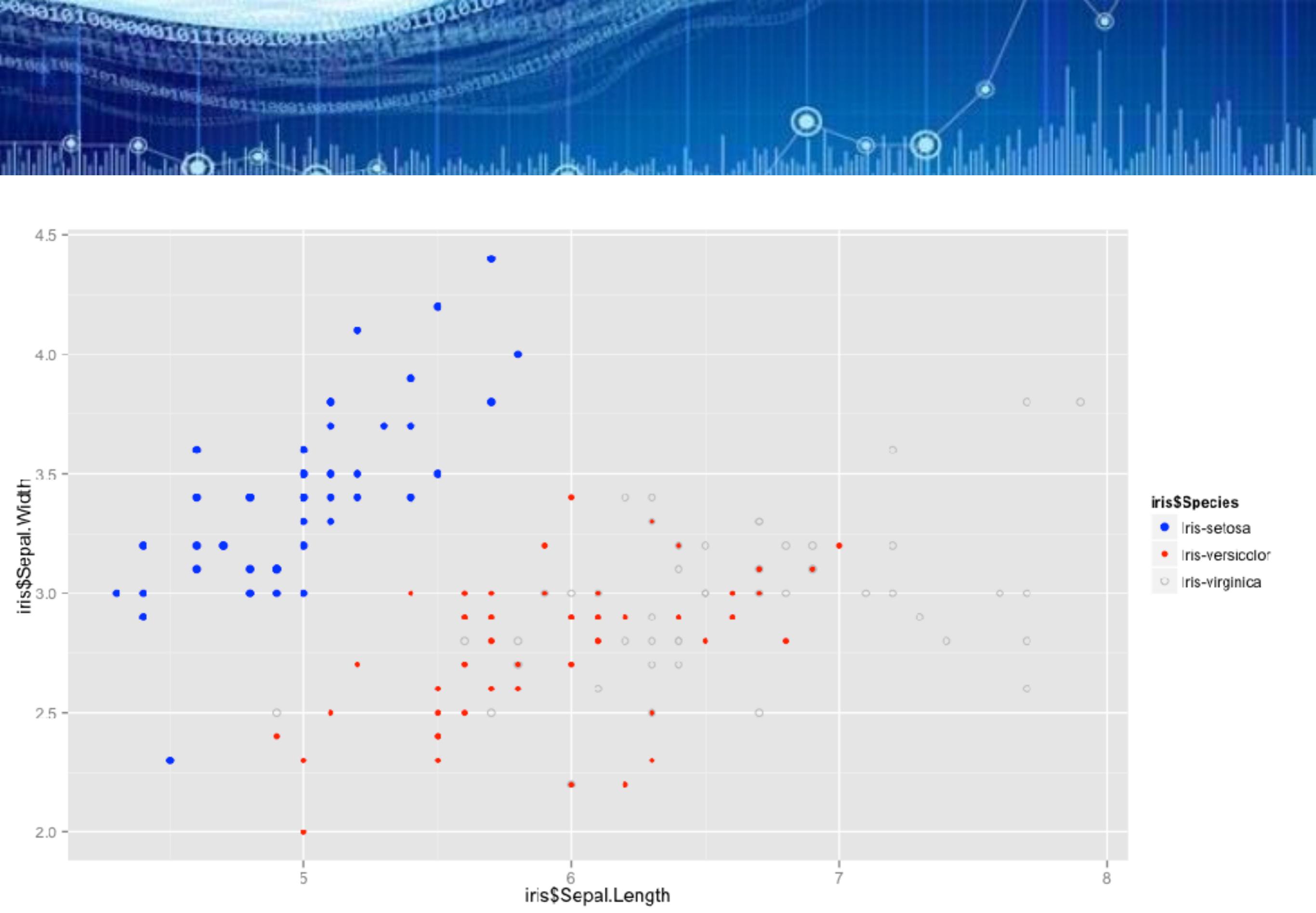


```
> # Plot the histogram  
> hist(iris$Sepal.Length, breaks=10, prob=T)  
> # Plot the density curve  
> lines(density(iris$Sepal.Length))  
>
```





```
ggplot(iris, aes(x=iris$SepalLength, y=iris$SepalWidth,  
group=iris$Species, shape=iris$Species)) +  
  geom_point(aes(colour=iris$Species)) +  
  scale_shape_manual(values=c(19,20,21))+  
  scale_colour_manual(values=c("blue", "red","gray"))
```

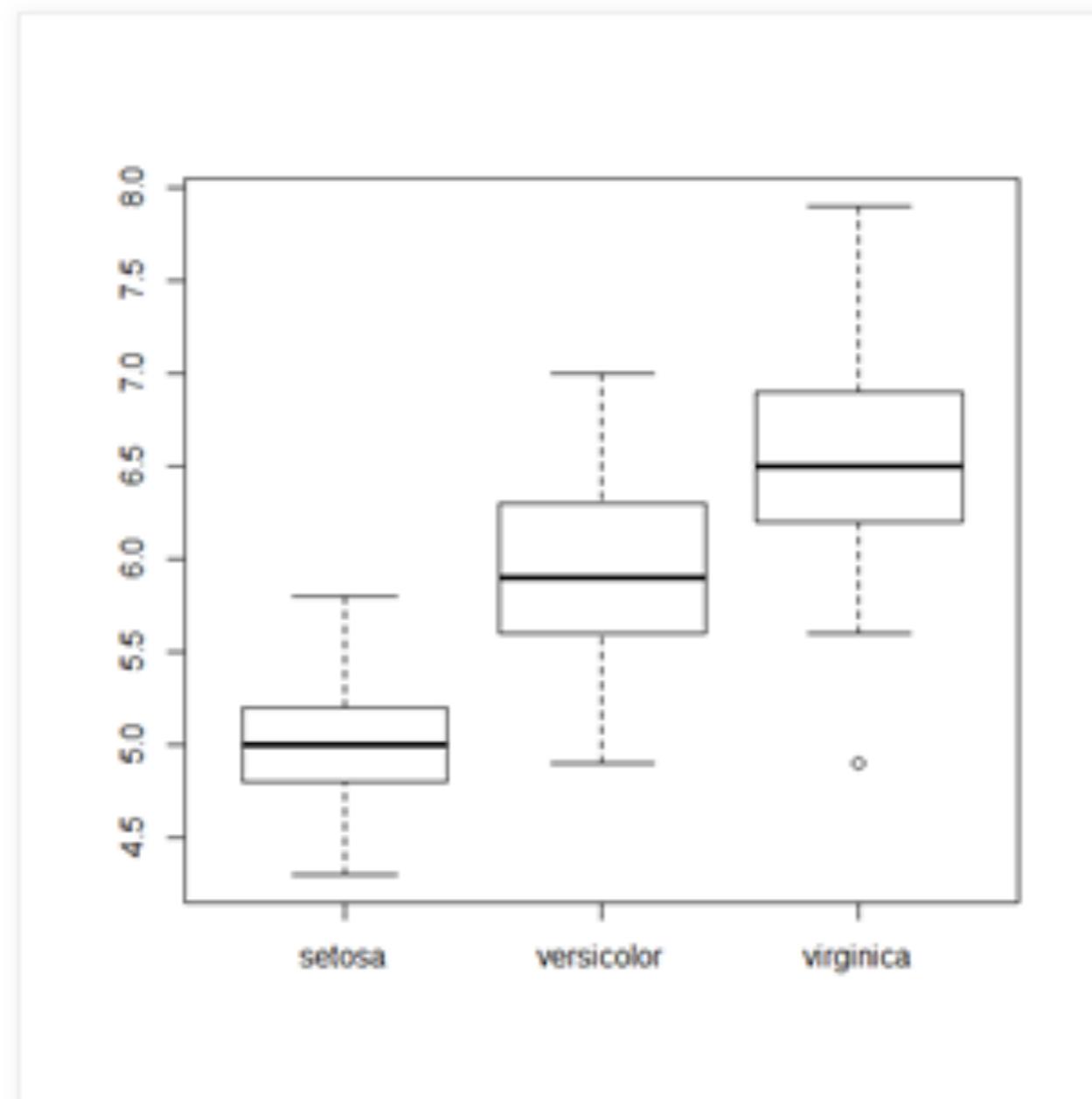


Covariation

If variation describes the behavior within a variable, **covariation describes the behavior between variables. Covariation is the tendency for the values of two or more variables to vary together in a related way.** The best way to spot covariation is to visualize the relationship between two or more variables. How you do that should again depend on the type of variables involved.

1) **A categorical and continuous variable :** We want to explore the distribution of a continuous variable broken down by a categorical variable. Boxplot can help us

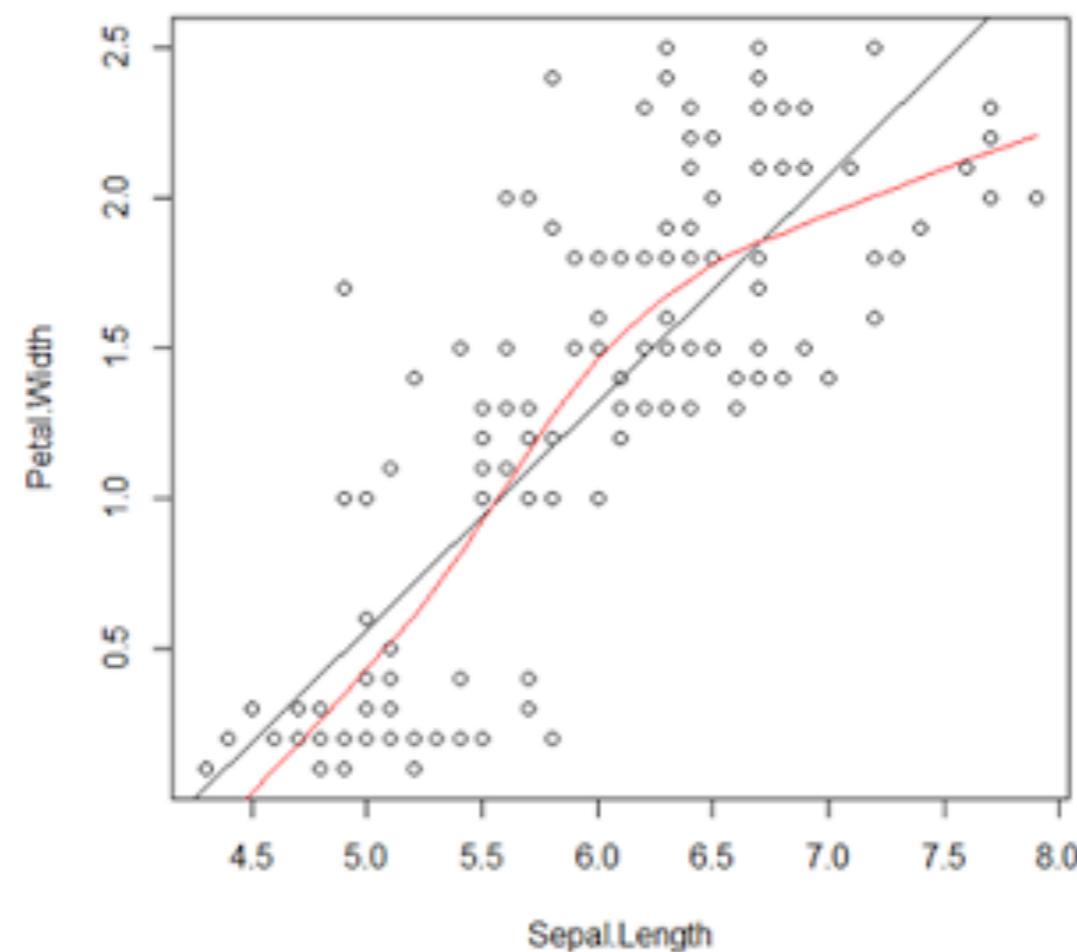
```
> boxplot(Sepal.Length~Species, data=iris)
>
```





2) Two continuous variables : One great way to visualize the covariation between two continuous variables: draw a **scatterplot**.

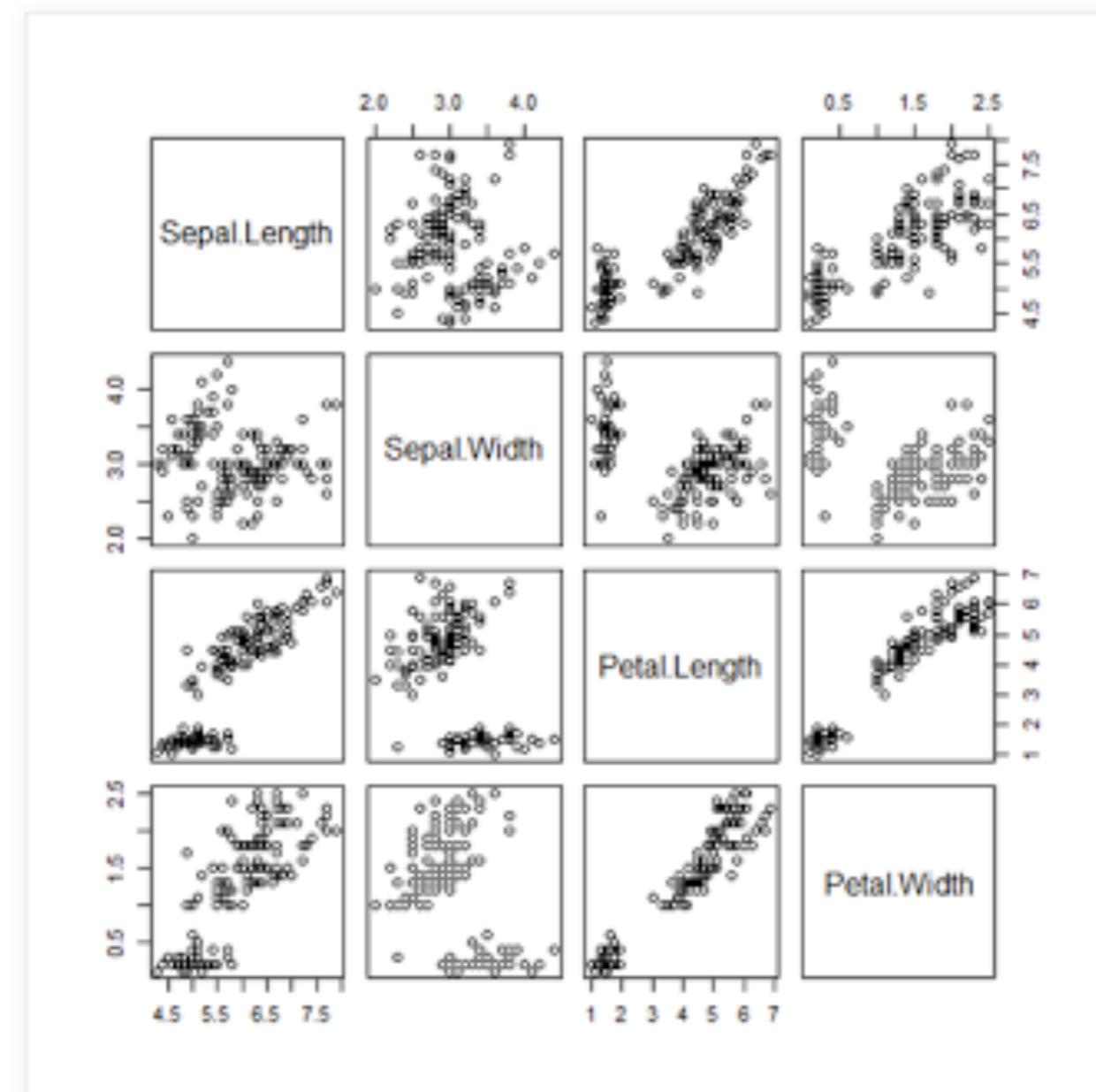
```
> # First plot the 2 variables  
> plot(Petal.Width~Sepal.Length, data=iris)  
> # Learn the regression model  
> model <- lm(Petal.Width~Sepal.Length, data=iris)  
> # Plot the regression line  
> abline(model)  
> # Now learn the local linear model  
> model2 <- lowess(iris$Petal.Width~iris$Sepal.Length)  
> lines(model2, col="red")  
>
```



```

> # Scatter plot for all pairs
> pairs(iris[,c(1,2,3,4)])
> # Compute the correlation matrix
> cor(iris[,c(1,2,3,4)])
      Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length     1.0000000 -0.1170695   0.8716902   0.8179410
Sepal.Width      -0.1170695  1.0000000  -0.4284401  -0.3661259
Petal.Length      0.8716902 -0.4284401   1.0000000   0.9628654
Petal.Width       0.8179410 -0.3661259   0.9628654   1.0000000
>

```





Data Preparation

Preparing Training Data

At this step, the purpose is to transform the raw data in a form that can fit into the data mining model.

- Data sampling
- Data validation and handle missing data
- Normalize numeric value into a uniform range
- Compute aggregated value (a special case is to compute frequency counts)
- Expand categorical field to binary fields
- Discretize numeric value into categories
- Create derived fields from existing fields
- Reduce dimensionality
- Power and Log transformation

Data Sampling

```
> # select 10 records out from iris with replacement
> index <- sample(1:nrow(iris), 10, replace=T)
> index
[1] 133 36 107 140 66 67 36 3 97 37
> irissample <- iris[index,]
> irissample
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
133          6.4        2.8         5.6        2.2 virginica
36           5.0        3.2         1.2        0.2    setosa
107          4.9        2.5         4.5        1.7 virginica
140          6.9        3.1         5.4        2.1 virginica
66           6.7        3.1         4.4        1.4 versicolor
67           5.6        3.0         4.5        1.5 versicolor
36.1         5.0        3.2         1.2        0.2    setosa
3            4.7        3.2         1.3        0.2    setosa
97           5.7        2.9         4.2        1.3 versicolor
37           5.5        3.5         1.3        0.2    setosa
>
```

Impute missing data

- Discard the whole record
- Infer missing value based on the data of other record. Approach is to fill the missing data with the average or the median.

```
> # Create some missing data
> irissample[10, 1] <- NA
> irissample
   Sepal.Length Sepal.Width Petal.Length Petal.Width     Species
133          6.4         2.8          5.6          2.2  virginica
 36          5.0         3.2          1.2          0.2    setosa
107          4.9         2.5          4.5          1.7  virginica
140          6.9         3.1          5.4          2.1  virginica
 66          6.7         3.1          4.4          1.4 versicolor
 67          5.6         3.0          4.5          1.5 versicolor
36.1          5.0         3.2          1.2          0.2    setosa
 3          4.7         3.2          1.3          0.2    setosa
 97          5.7         2.9          4.2          1.3 versicolor
 37           NA         3.5          1.3          0.2    setosa
...          ...         ...          ...          ...       ...
```

```
> library(e1071)
Loading required package: class
Warning message:
package 'e1071' was built under R version 2.14.2
> fixIris1 <- impute(irissample[,1:4], what='mean')
> fixIris1
  Sepal.Length Sepal.Width Petal.Length Petal.Width
133      6.400000     2.8        5.6        2.2
36       5.000000     3.2        1.2        0.2
107      4.900000     2.5        4.5        1.7
140      6.900000     3.1        5.4        2.1
66       6.700000     3.1        4.4        1.4
67       5.600000     3.0        4.5        1.5
36.1     5.000000     3.2        1.2        0.2
3        4.700000     3.2        1.3        0.2
97       5.700000     2.9        4.2        1.3
37      5.655556     3.5        1.3        0.2
> fixIris2 <- impute(irissample[,1:4], what='median')
> fixIris2
  Sepal.Length Sepal.Width Petal.Length Petal.Width
133      6.4        2.8        5.6        2.2
36       5.0        3.2        1.2        0.2
107      4.9        2.5        4.5        1.7
140      6.9        3.1        5.4        2.1
66       6.7        3.1        4.4        1.4
67       5.6        3.0        4.5        1.5
36.1     5.0        3.2        1.2        0.2
3        4.7        3.2        1.3        0.2
97       5.7        2.9        4.2        1.3
37      5.6        3.5        1.3        0.2
>
```

Normalize numeric value

```
> # scale the columns  
> # x-mean(x)/standard deviation  
> scaleiris <- scale(iris[, 1:4])  
> head(scaleiris)  
  Sepal.Length Sepal.Width Petal.Length Petal.Width  
[1,] -0.8976739  1.01560199 -1.335752 -1.311052  
[2,] -1.1392005 -0.13153881 -1.335752 -1.311052  
[3,] -1.3807271  0.32731751 -1.392399 -1.311052  
[4,] -1.5014904  0.09788935 -1.279104 -1.311052  
[5,] -1.0184372  1.24503015 -1.335752 -1.311052  
[6,] -0.5353840  1.93331463 -1.165809 -1.048667  
>
```

Reduce dimensionality

There are two ways to reduce the number of input attributes.

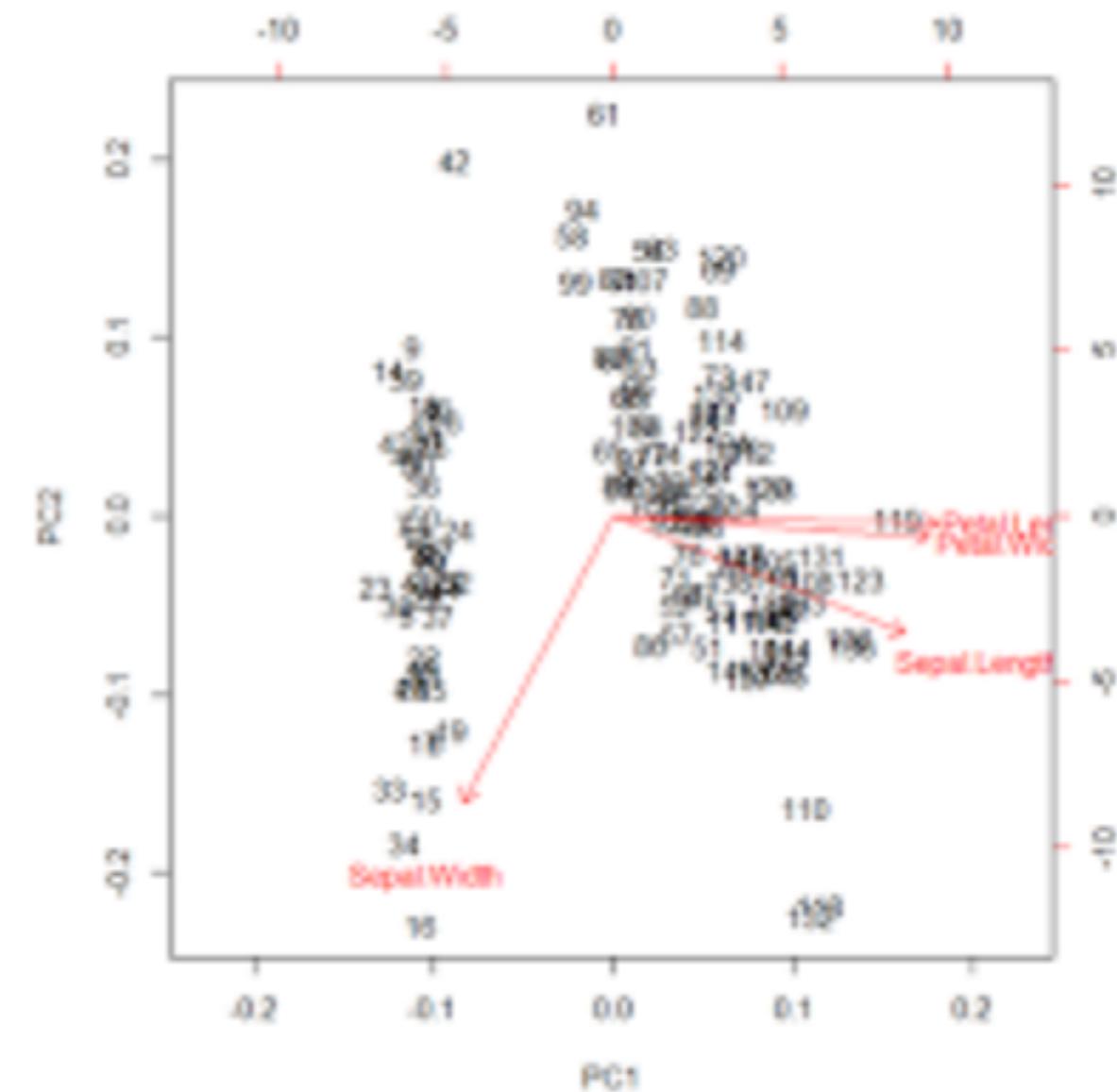
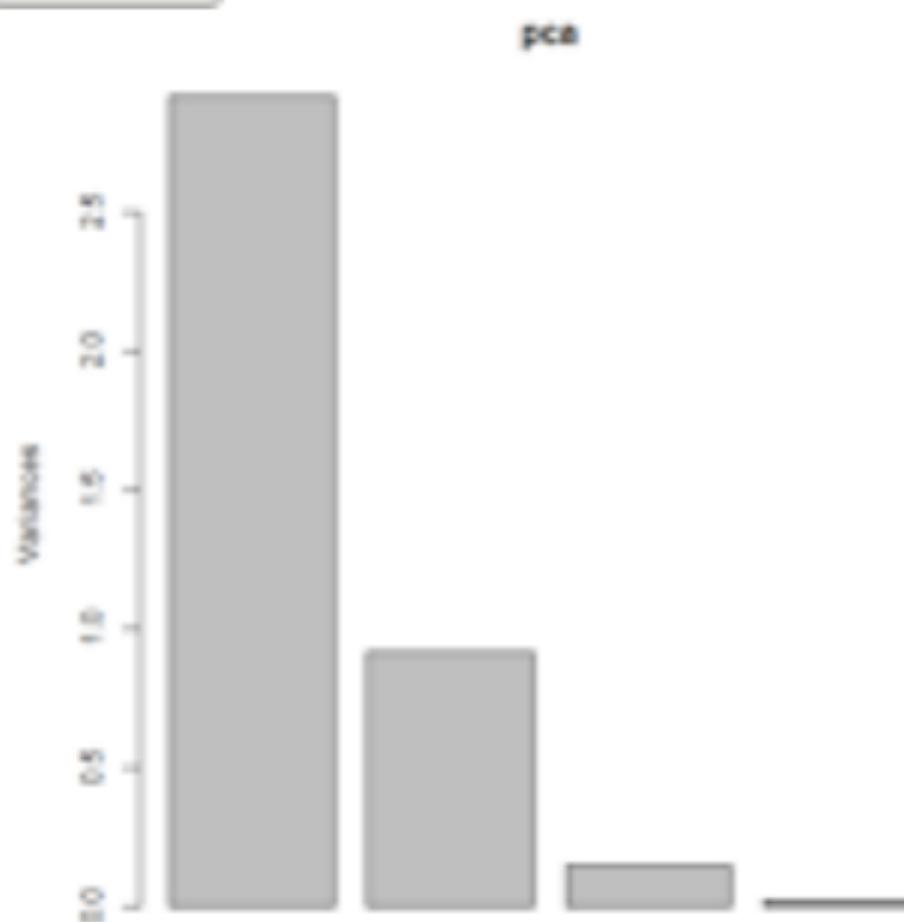
1. Removing irrelevant input variables.
2. Removing redundant input variables.

```

> # Use iris data set
> cor(iris[, -5])
      Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length 1.0000000000 -0.1175697841 0.8717537759 0.8179411263
Sepal.Width -0.1175697841 1.0000000000 -0.4284401043 -0.3661259325
Petal.Length 0.8717537759 -0.4284401043 1.0000000000 0.9628654314
Petal.Width 0.8179411263 -0.3661259325 0.9628654314 1.0000000000
> # Some attributes shows high correlation, compute PCA
> pca <- prcomp(iris[,-5], scale=T)
> summary(pca)
Importance of components:
              PC1        PC2        PC3        PC4
Standard deviation     1.708361 0.9560494 0.3830886 0.1439265
Proportion of Variance 0.729620 0.2285100 0.0366900 0.0051800
Cumulative Proportion  0.729620 0.9581300 0.9948200 1.0000000
> # Notice PC1 and PC2 covers most variation
> plot(pca)
> pca$rotation
             PC1        PC2        PC3        PC4
Sepal.Length 0.5210659147 -0.37741761556 0.7195663527 0.2612862800
Sepal.Width -0.2693474425 -0.92329565954 -0.2443817795 -0.1235096196
Petal.Length 0.5804130958 -0.02449160909 -0.1421263693 -0.8014492463
Petal.Width  0.5648565358 -0.06694198697 -0.6342727371 0.5235971346
> # Project first 2 records in PCA direction
> predict(pca)[1:2,]
             PC1        PC2        PC3        PC4
[1,] -2.257141176 -0.4784238321 0.1272796237 0.02408750846
[2,] -2.074013015  0.6718826870 0.2338255167 0.10266284468
> # plot all points in top 2 PCA direction
> biplot(pca)

```

Pin it



Add derived attributes

```
> iris2 <- transform(iris, ratio=round(Sepal.Length/Sepal.Width, 2))
> head(iris2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species ratio
1          5.1        3.5         1.4       0.2   setosa 1.46
2          4.9        3.0         1.4       0.2   setosa 1.63
3          4.7        3.2         1.3       0.2   setosa 1.47
4          4.6        3.1         1.5       0.2   setosa 1.48
5          5.0        3.6         1.4       0.2   setosa 1.39
6          5.4        3.9         1.7       0.4   setosa 1.38
```

Discretize numeric value into categories

```
> # Equal width cuts
> segments <- 10
> maxL <- max(iris$Petal.Length)
> minL <- min(iris$Petal.Length)
> theBreaks <- seq(minL, maxL,
+                     by=(maxL-minL)/segments)
> cutPetalLength <- cut(iris$Petal.Length,
+                         breaks=theBreaks,
+                         include.lowest=T)
> newdata <- data.frame(orig.Petal.Len=iris$Petal.Length,
+                         cut.Petal.Len=cutPetalLength)
> head(newdata)
  orig.Petal.Len cut.Petal.Len
1      1.4      [1,1.59]
2      1.4      [1,1.59]
3      1.3      [1,1.59]
4      1.5      [1,1.59]
5      1.4      [1,1.59]
6      1.7  (1.59,2.18]
>
> # Constant frequency / height
> myBreaks <- quantile(iris$Petal.Length,
+                        probs=seq(0,1,1/segments))
> cutPetalLength2 <- cut(iris$Petal.Length,
+                         breaks=myBreaks,
+                         include.lowest=T)
> newdata2 <- data.frame(orig.Petal.Len=iris$Petal.Length,
+                         cut.Petal.Len=cutPetalLength2)
> head(newdata2)
  orig.Petal.Len cut.Petal.Len
1      1.4      [1,1.4]
2      1.4      [1,1.4]
3      1.3      [1,1.4]
4      1.5  (1.4,1.5]
5      1.4      [1,1.4]
6      1.7  (1.7,3.9]
```

Binarize categorical attributes

```
> cat <- levels(iris$Species)
> cat
[1] "setosa"      "versicolor"   "virginica"
> binarize <- function(x) {return(iris$Species == x)}
> newcols <- sapply(cat, binarize)
> colnames(newcols) <- cat
> data <- cbind(iris[,c('Species')], newcols)
> data[45:55,]
      setosa versicolor virginica
[1, ] 1       1       0       0
[2, ] 1       1       0       0
[3, ] 1       1       0       0
[4, ] 1       1       0       0
[5, ] 1       1       0       0
[6, ] 1       1       0       0
[7, ] 2       0       1       0
[8, ] 2       0       1       0
[9, ] 2       0       1       0
[10,] 2       0       1       0
[11,] 2       0       1       0
```

Workshop 10 - Getting Data

- Choose data from dataset below.
 - Sales Win or Loss Dataset
 - HR Employee Attrition Dataset
 - Telco Customer Churn Dataset
 - Titanic Survival Dataset
- Import data
- Practice data visualization
- Practice data preparation



Machine Learning Techniques

Topic

- **Basic concepts**

- Decision tree induction
- Evaluation of classifiers
- Naïve Bayesian classification
- K-nearest neighbor
- Support Vector Machine
- Neural Net
- Ensemble methods: Bagging and Boosting
- Summary

An example application

An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.

A decision is needed: whether to put a new patient in an intensive-care unit.

Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.

Problem: to predict **high-risk patients** and discriminate them from **low-risk patients**.

Another application

A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,

- age
 - Marital status
 - annual salary
 - outstanding debts
 - credit rating
- etc.

Problem: to decide whether an application should approved, or to classify applications into two categories, **approved** and **not approved**.

Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- **A computer system learns from data**, which represent some “past experiences” of an application domain.
- **Our focus:** learn **a target function** that can be used to predict the values of a discrete class attribute, e.g., **approve** or **not-approved**, and **high-risk** or **low risk**.
- The task is commonly called: **Supervised learning, classification**, or **inductive learning**.

The data and the goal

Data: A set of data records (also called examples, instances or cases) described by

k attributes: A_1, A_2, \dots, A_k .

a class: Each example is labelled with a pre-defined class.

Goal: To learn a **classification model** from the data that can be used to predict the classes of new (future, or test) cases/instances.

An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

An example: the learning task

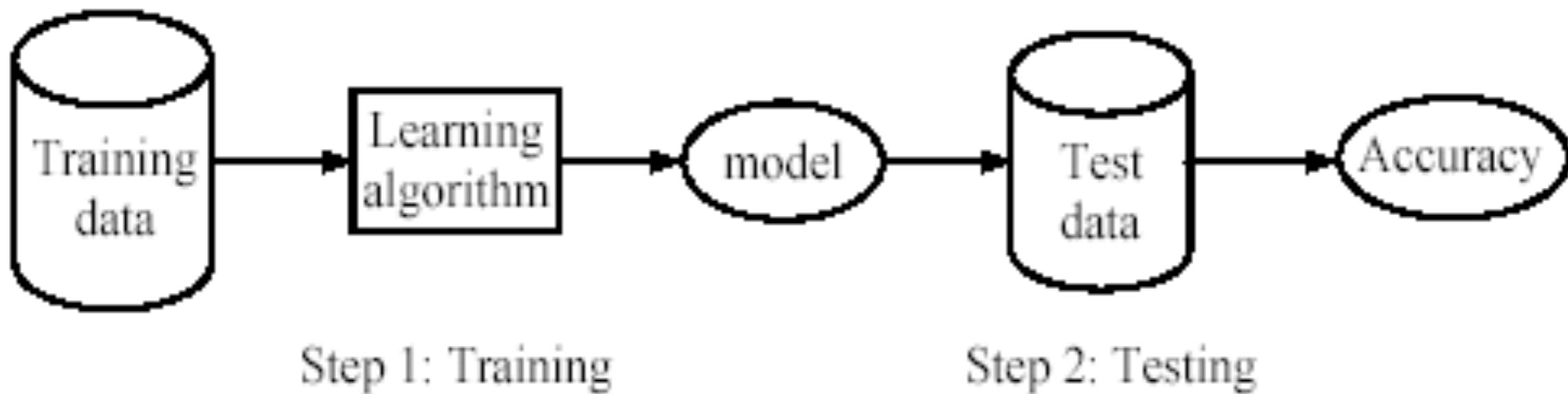
- Learn a classification model from the data
- Use the model to classify future loan applications into
 - Yes (approved) and
 - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

Supervised learning process: two steps

- **Learning (training)**: Learn a model using the training data
- **Testing**: Test the model using **unseen test data** to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



What do we mean by learning?

Given

a data set D ,

a task T , and

a performance measure M ,

a computer system is said to **learn** from D to perform the task T if after learning the system's performance on T improves as measured by M .

In other words, the learned model helps the system to perform T better as compared to no learning.

An example

Data: Loan application data

Task: Predict whether a loan should be approved or not.

Performance measure: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., **Yes**):

$$\text{Accuracy} = 9/15 = 60\%.$$

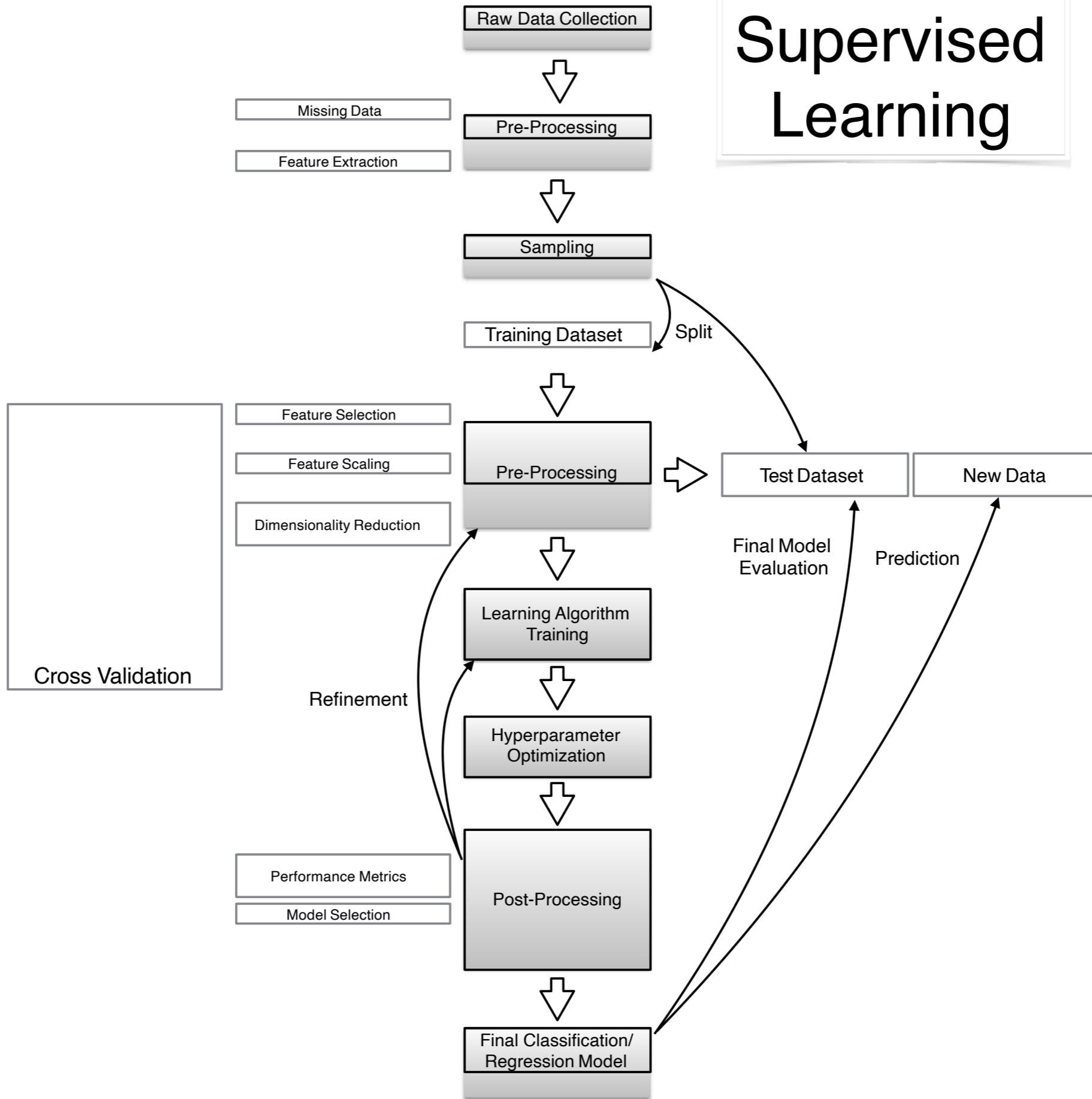
We can do better than 60% with learning.

Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

Supervised Learning



Topic

- Basic concepts
- **Decision tree induction**
- Evaluation of classifiers
- Naïve Bayesian classification
- K-nearest neighbor
- Support Vector Machine
- Neural Net
- Ensemble methods: Bagging and Boosting
- Summary

Introduction

- Decision tree learning is one of the most widely used techniques for classification.
 - Its classification accuracy is competitive with other methods, and
 - it is very efficient.
- The classification model is a tree, called **decision tree**.
- **C4.5** by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

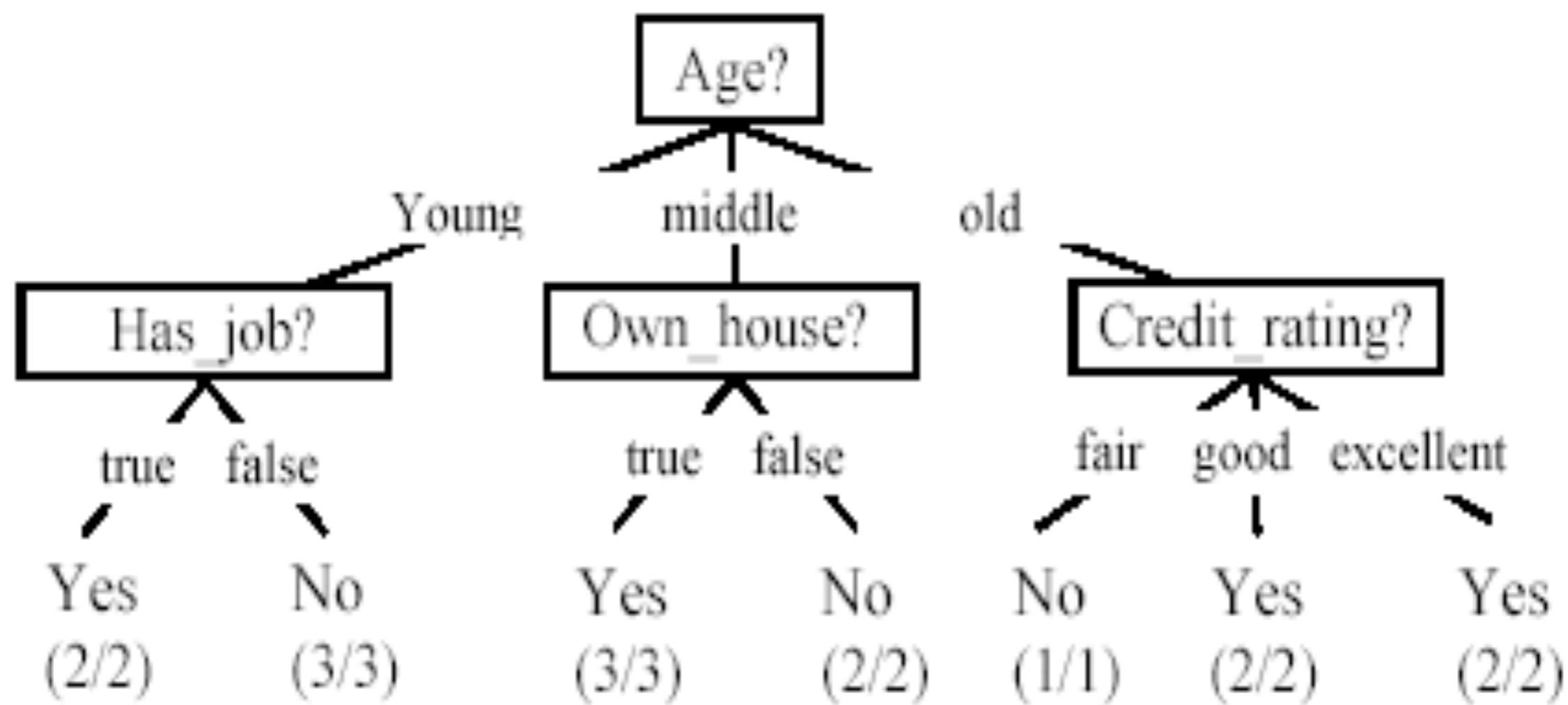
The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

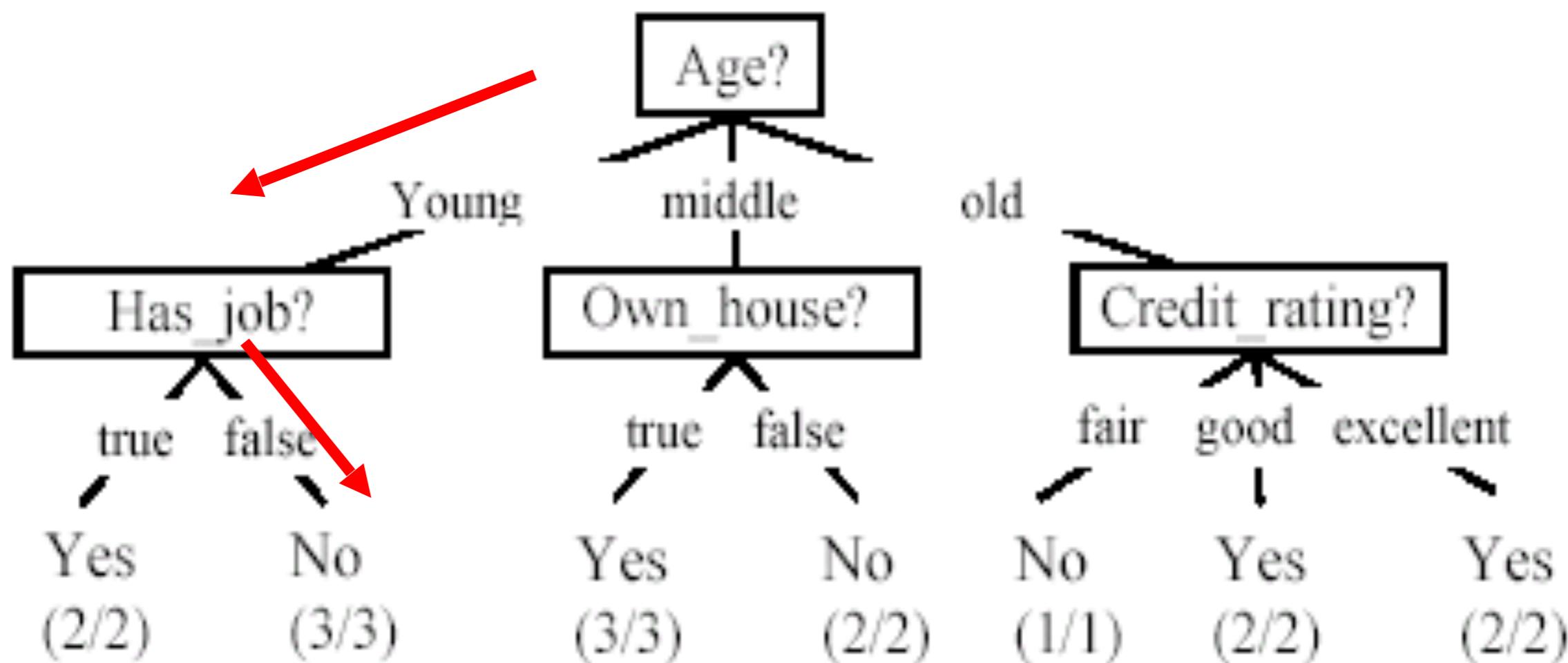
A decision tree from the loan data

- Decision nodes and leaf nodes (classes)



Use the decision tree

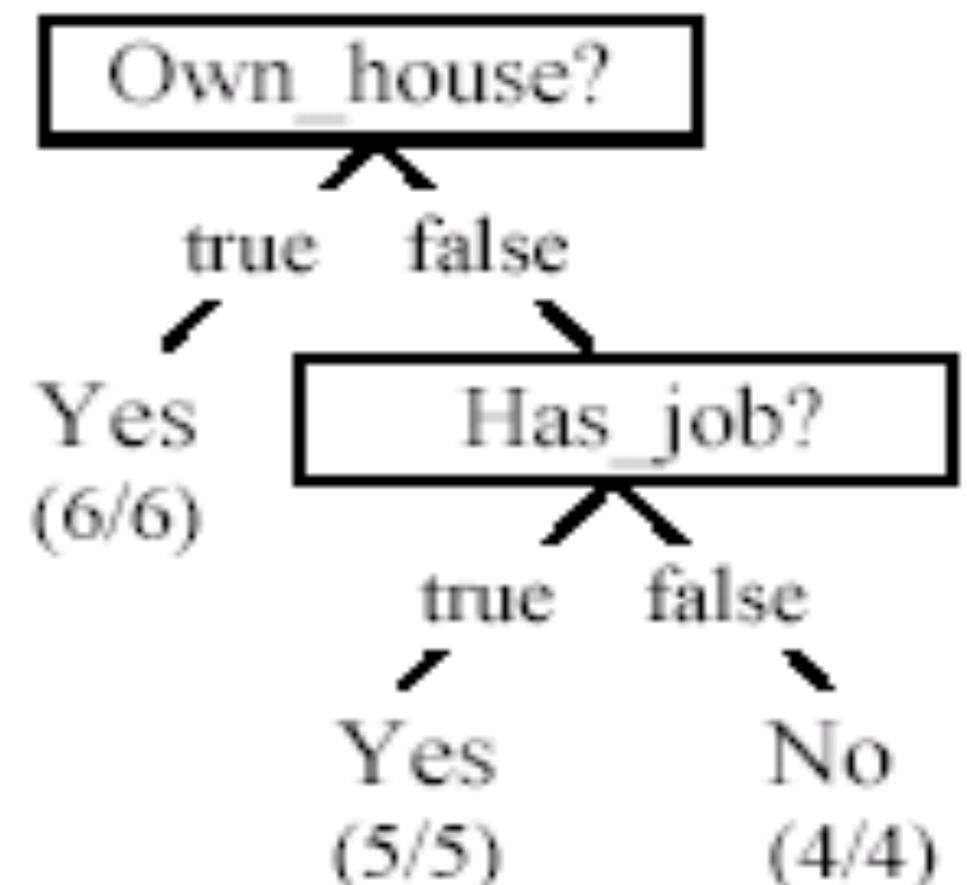
Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	No



Is the decision tree unique?

- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
 - Easy to understand and perform better.

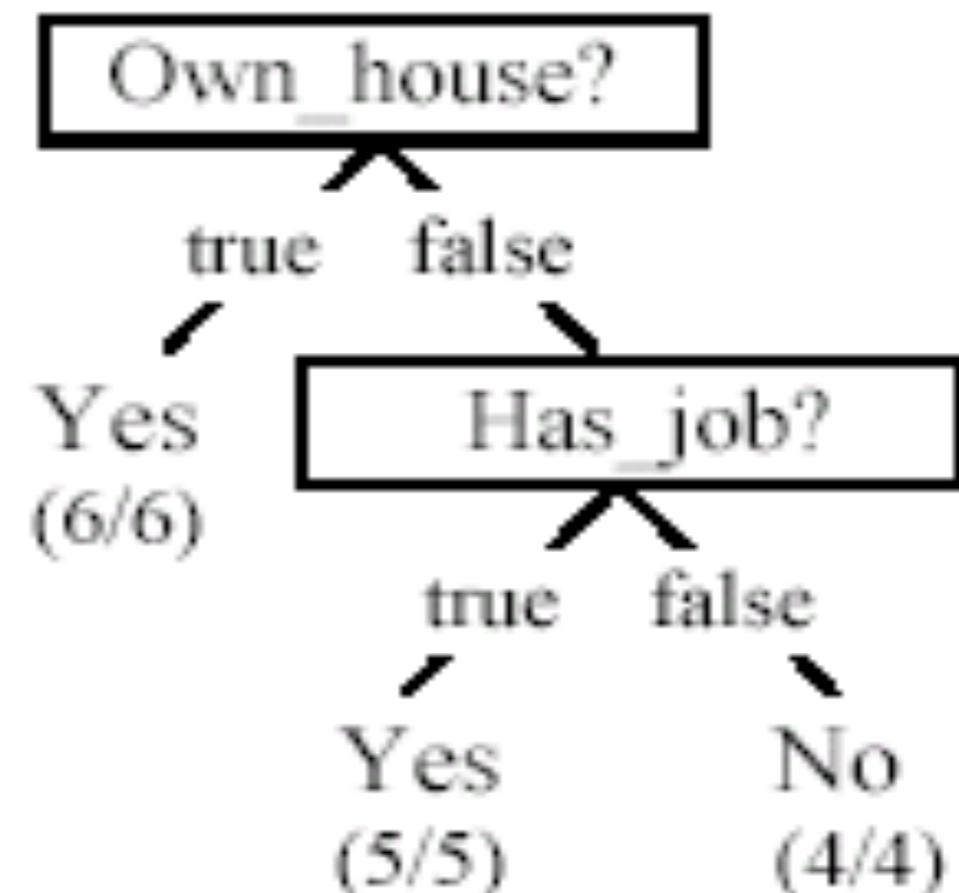
- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



NP-hard (Non-deterministic Polynomial-time hard), in computational complexity theory, is a class of problems that are, informally, "at least as hard as the hardest problems in NP"

From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.



Own_house = true → Class = Yes [sup=6/15, conf=6/6]

Own_house = false, Has_job = true → Class = Yes [sup=5/15, conf=5/5]

Own_house = false, Has_job = false → Class = No [sup=4/15, conf=4/4]

Choose an attribute to partition data

- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.

A subset of data is **pure** if all instances belong to the same class.

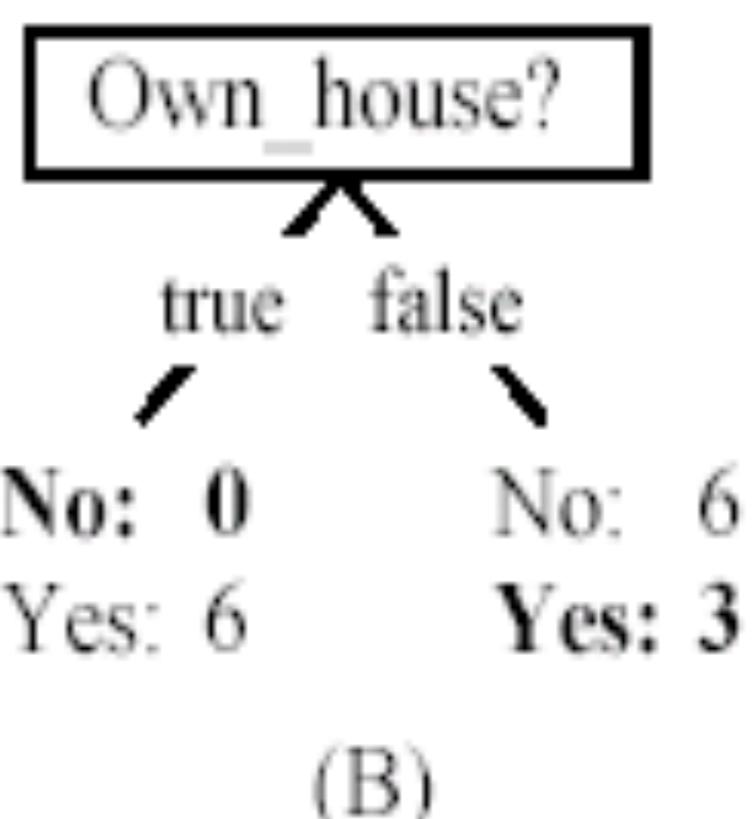
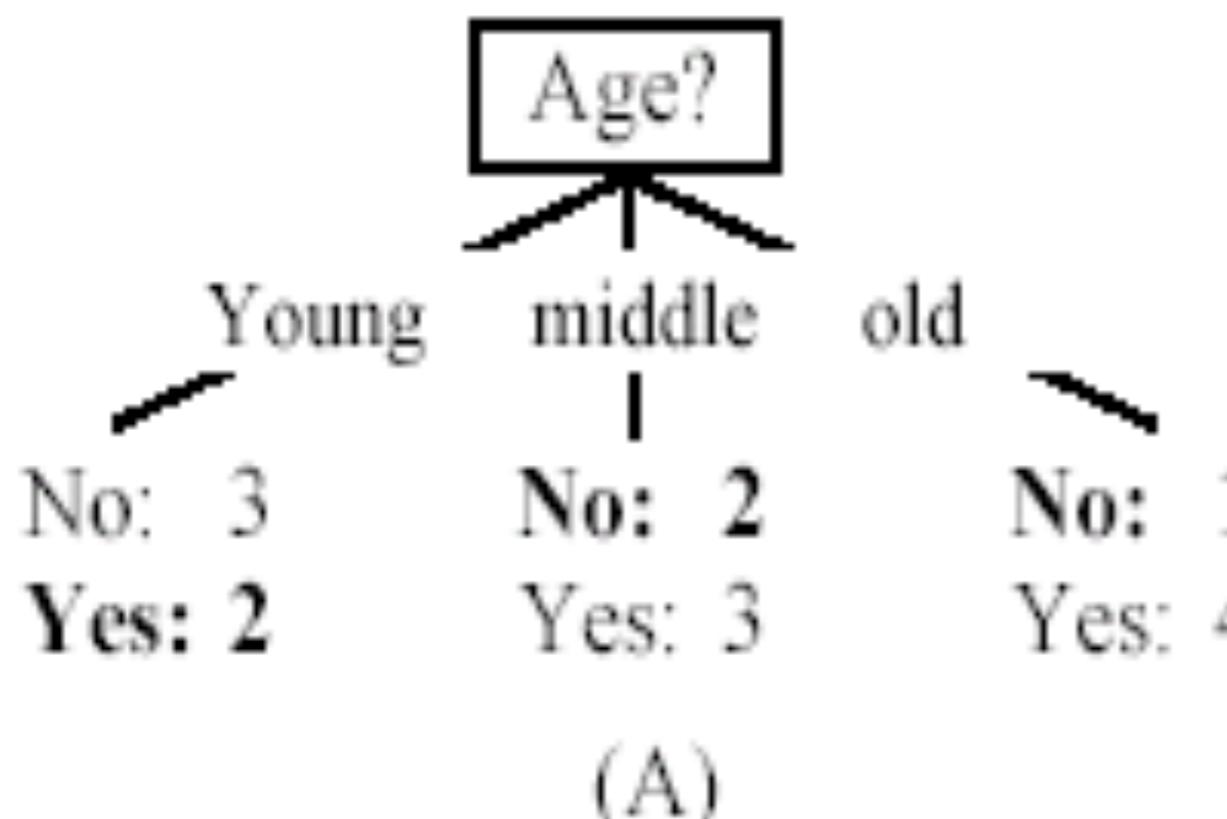
- The *heuristic* in C4.5 is to choose the attribute with the maximum **Information Gain** or **Gain Ratio** based on information theory.

The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Two possible roots, which is better?



- Fig. (B) seems to be better.

Entropy measure: let us get a feeling

1. The data set D has 50% positive examples ($\Pr(\text{positive}) = 0.5$) and 50% negative examples ($\Pr(\text{negative}) = 0.5$).

$$\text{entropy}(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(\text{positive}) = 0.2$) and 80% negative examples ($\Pr(\text{negative}) = 0.8$).

$$\text{entropy}(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(\text{positive}) = 1$) and no negative examples, ($\Pr(\text{negative}) = 0$).

$$\text{entropy}(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

Information gain

Given a set of examples D , we first compute its entropy:

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

If we make attribute A_i , with v values, the root of the current tree,
this will partition D into v subsets $D_1, D_2 \dots, D_v$. The expected entropy
if A_i is used as the current root:

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

An example

$$\text{entropy}(D) = \frac{6}{15} \times \log_2 \frac{6}{15} + \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

$$\begin{aligned}\text{entropy}_{\text{Own_house}}(D) &= \frac{6}{15} \times \text{entropy}(D_1) + \frac{9}{15} \times \text{entropy}(D_2) \\ &= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918 \\ &= 0.551\end{aligned}$$

$$\begin{aligned}\text{entropy}_{\text{Age}}(D) &= \frac{5}{15} \times \text{entropy}(D_1) + \frac{5}{15} \times \text{entropy}(D_2) + \frac{5}{15} \times \text{entropy}(D_3) \\ &= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722 \\ &= 0.888\end{aligned}$$

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Age	Yes	No	entropy(Di)
young	2	3	0.971
middle	3	2	0.971
old	4	1	0.722

$$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$$

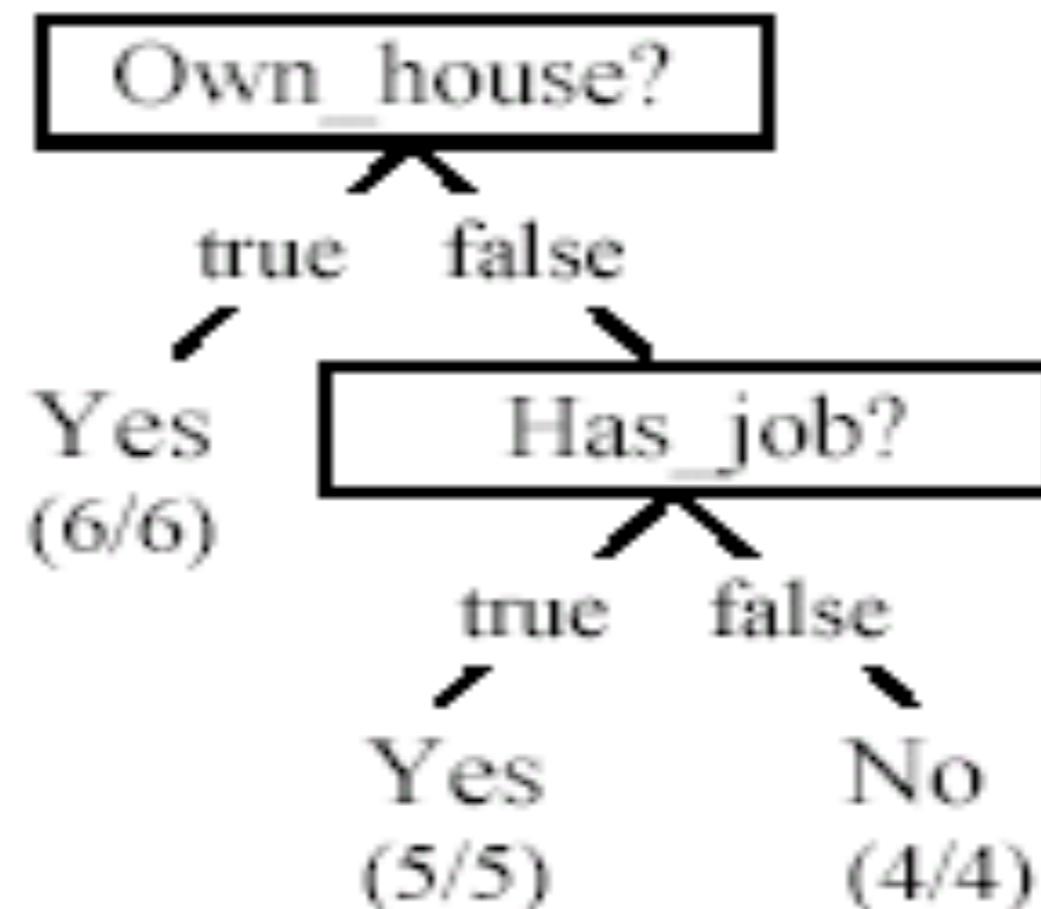
$$gain(D, \text{Own_house}) = 0.971 - 0.551 = 0.420$$

$$gain(D, \text{Has_Job}) = 0.971 - 0.647 = 0.324$$

$$gain(D, \text{Credit_Rating}) = 0.971 - 0.608 = 0.363$$

- Own_house is the best choice for the root.

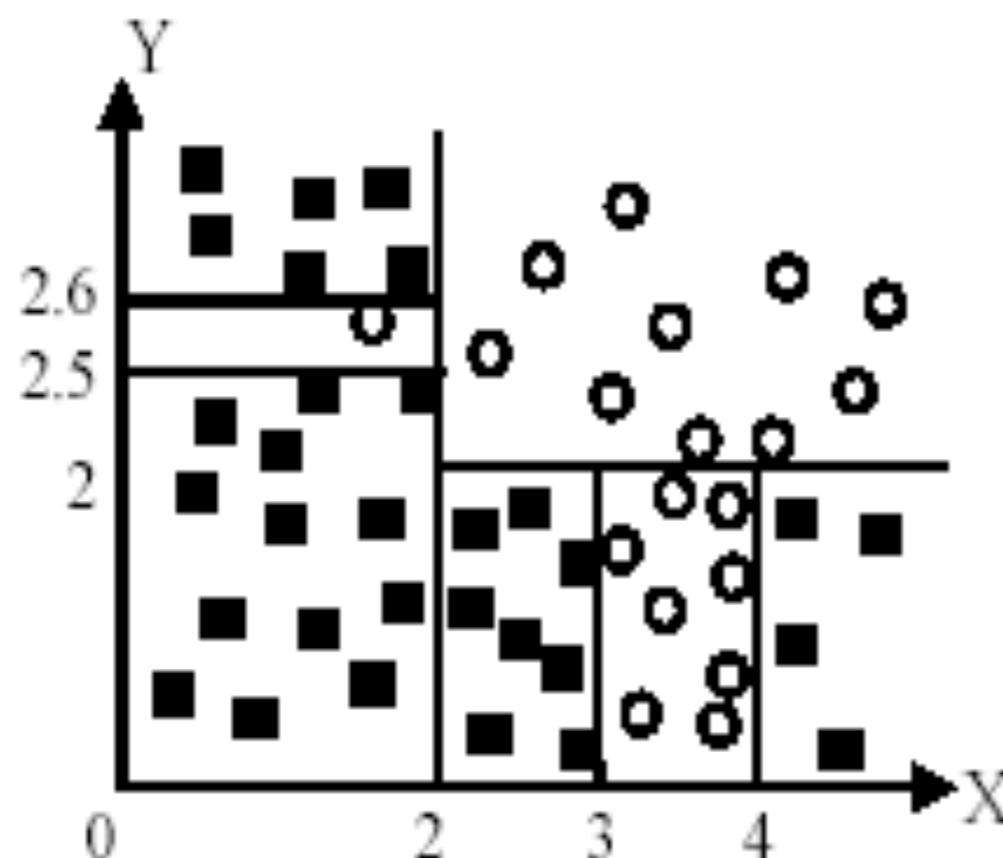
We build the final tree



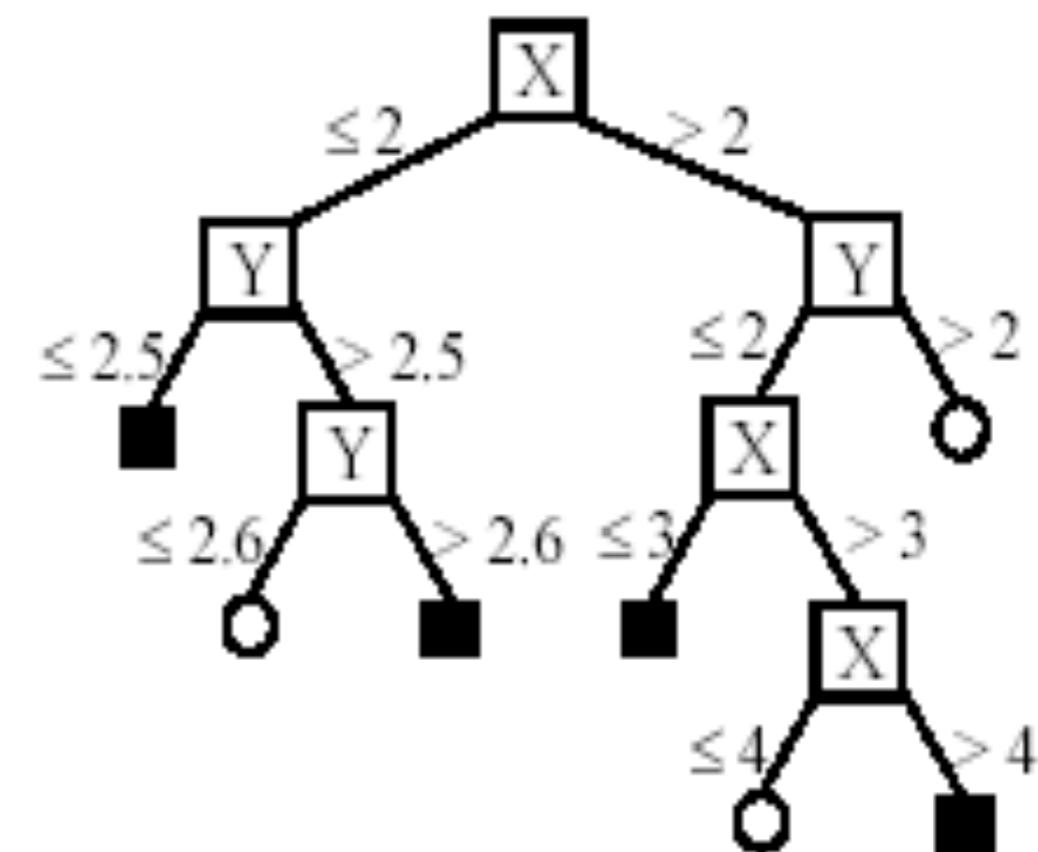
Handling continuous attributes

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
 - Use information gain or gain ratio again
 - Sort all the values of an continuous attribute in increasing order $\{v_1, v_2, \dots, v_r\}$,
 - One possible threshold between two adjacent values v_i and v_{i+1} . Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

An example in a continuous space



(A) A partition of the data space



(B). The decision tree

Avoid overfitting in classification

Overfitting: A tree may overfit the training data

Good accuracy on training data but poor on test data

Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers

Two approaches to avoid overfitting

Pre-pruning: Halt tree construction early

Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.

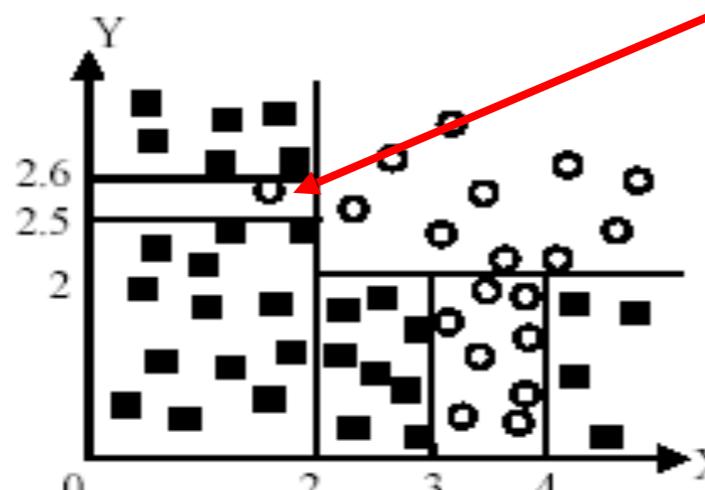
Post-pruning: Remove branches or sub-trees from a “fully grown” tree.

This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.

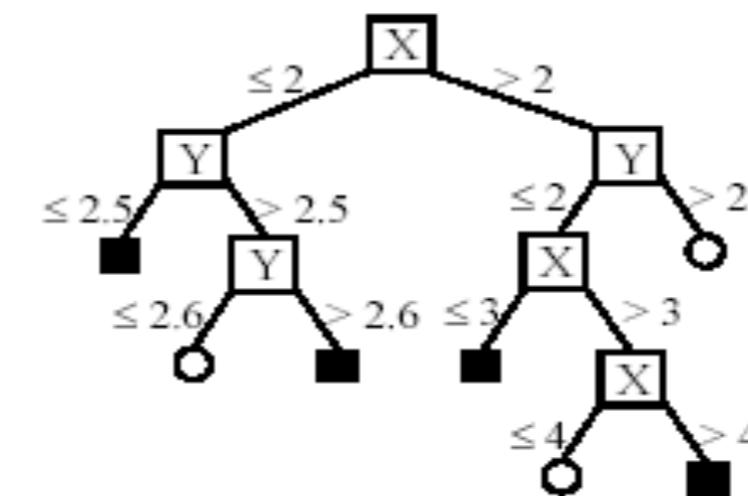
A validation set may be used for pruning as well.

An example

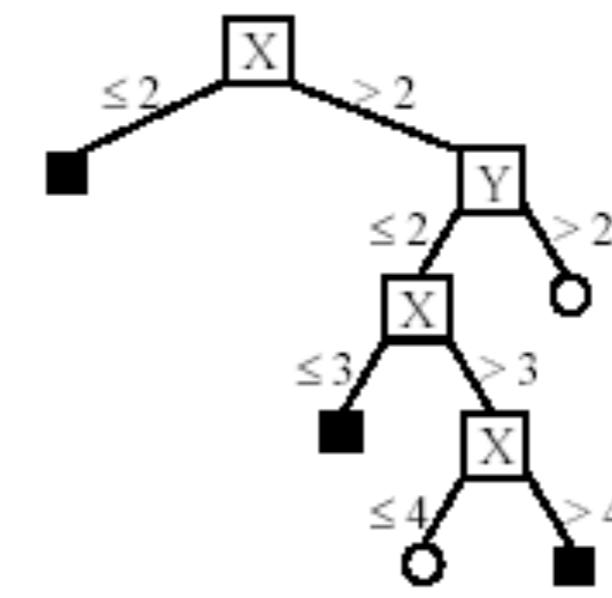
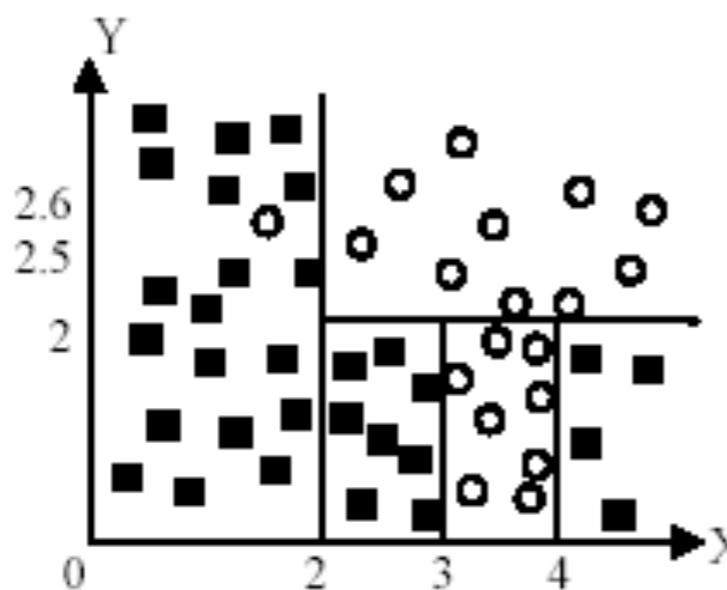
Likely to overfit the data



(A) A partition of the data space



(B). The decision tree



Other issues in decision tree learning

- From tree to rules, and rule pruning
- Handling of miss values
- Handing skewed distributions
- Handling attributes and classes with different costs.
- Attribute construction
- Etc.

Decision Tree in R

Iris Data Preparation

```
> set.seed(1234)  
  
> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))  
  
> trainData <- iris[ind==1,]  
  
> testData <- iris[ind==2,]
```

Decision Tree

Conditional Inference Trees

Description

Recursive partitioning for continuous, censored, ordered, nominal and multivariate response variables in a conditional inference framework.

Usage

```
ctree(formula, data, subset = NULL, weights = NULL,  
      controls = ctree_control(), xtrafo = ptratio, ytrafo = ptratio,  
      scores = NULL)
```

Arguments

- formula** a symbolic description of the model to be fit. Note that symbols like : and – will not work and the tree will make use of all variables listed on the rhs of **formula**.
- data** a data frame containing the variables in the model.
- subset** an optional vector specifying a subset of observations to be used in the fitting process.
- weights** an optional vector of weights to be used in the fitting process. Only non-negative integer valued weights are allowed.
- controls** an object of class [TreeControl](#), which can be obtained using [ctree_control](#).
- xtrafo** a function to be applied to all input variables. By default, the [ptratio](#) function is applied.
- ytrafo** a function to be applied to all response variables. By default, the [ptratio](#) function is applied.
- scores** an optional named list of scores to be attached to ordered factors.

Decision Tree - Create Model

```
> library(party)
> myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
> iris_ctree <- ctree(myFormula, data=trainData)
> # check the prediction
> table(predict(iris_ctree), trainData$Species)
```

	setosa	versicolor	virginica
setosa	40	0	0
versicolor	0	37	3
virginica	0	1	31

```
> print(iris_ctree)
```

Conditional inference tree with 4 terminal nodes

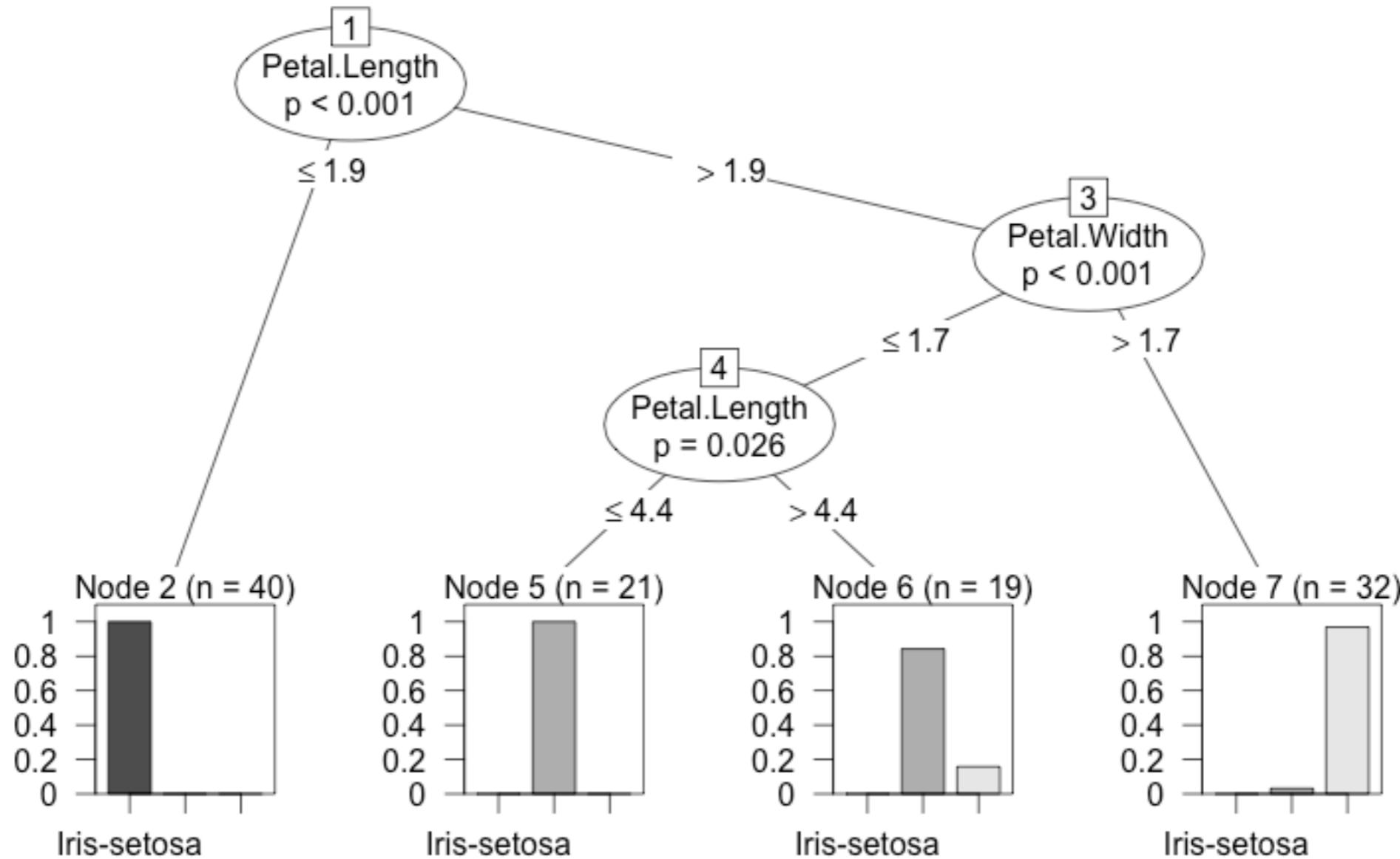
Response: Species

Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width

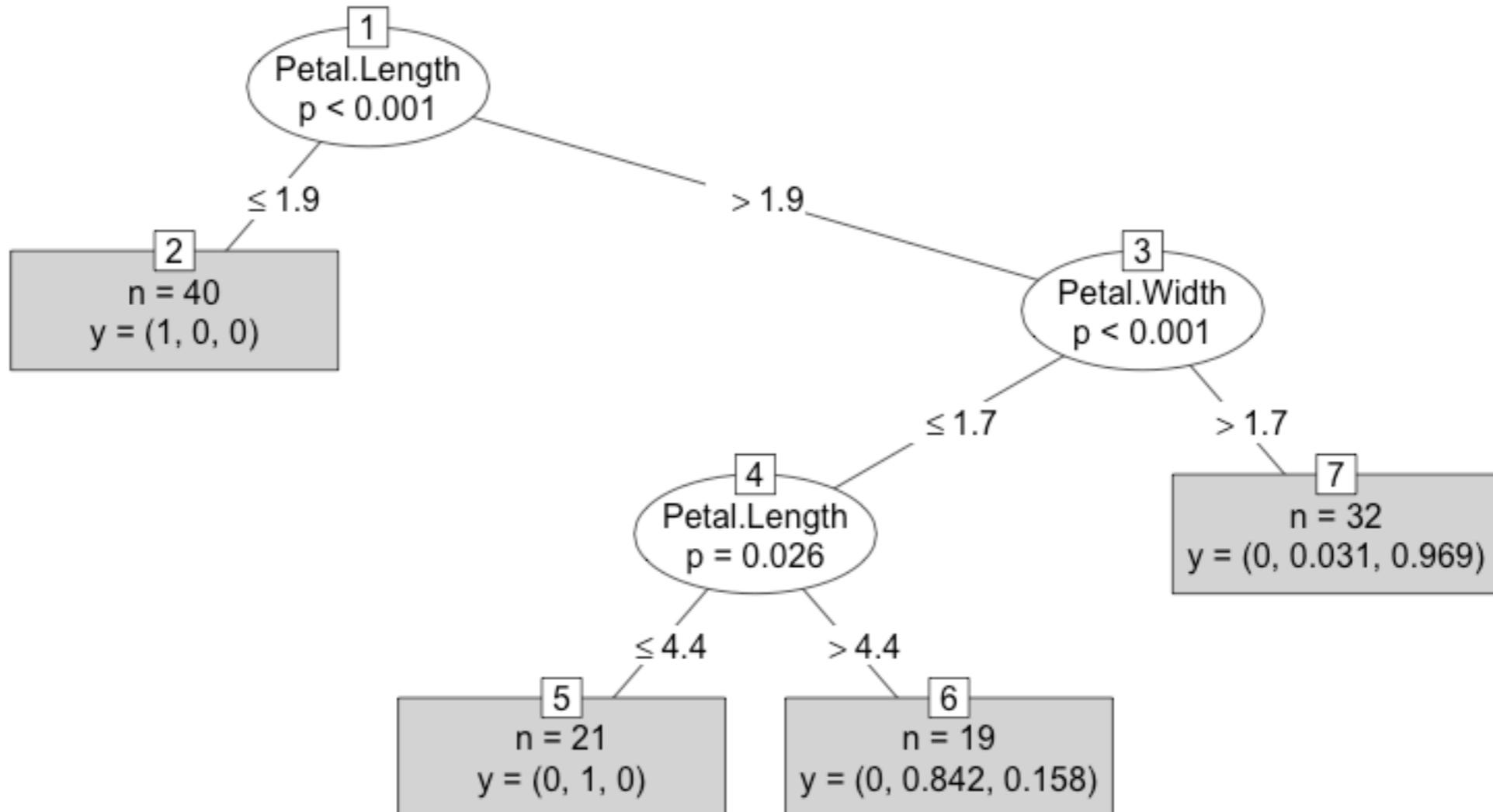
Number of observations: 112

- 1) Petal.Length <= 1.9; criterion = 1, statistic = 104.643
 - 2)* weights = 40
- 1) Petal.Length > 1.9
 - 3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
 - 4) Petal.Length <= 4.4; criterion = 0.974, statistic = 7.397
 - 5)* weights = 21
 - 4) Petal.Length > 4.4
 - 6)* weights = 19
 - 3) Petal.Width > 1.7
 - 7)* weights = 32

```
> plot(iris_ctree)
```



```
> plot(iris_ctree, type="simple")
```



Decision Tree - Prediction

```
> # predict on test data  
> testPred <- predict(iris_ctree, newdata = testData)  
> table(testPred, testData$Species)
```

testPred	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	12	2
virginica	0	0	14

Workshop 11 - Decision Tree

1. From the data that you import
2. Perform Decision Tree Algorithm to predict target Y

Evaluating classification methods

Predictive accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$

Efficiency

time to construct the model

time to use the model

Robustness: handling noise and missing values

Scalability: efficiency in disk-resident databases

Interpretability:

understandable and insight provided by the model

Compactness of the model: size of the tree, or the number of rules.

Evaluation methods

- **Holdout set:** The available data set D is divided into two disjoint subsets,
 - the *training set* D_{train} (for learning a model)
 - the *test set* D_{test} (for testing the model)
- **Important:** training set should not be used in testing and the test set should not be used in learning.
 - Unseen test set provides a unbiased estimate of accuracy.
 - The test set is also called the **holdout set**. (the examples in the original data set D are all labeled with classes.)
 - This method is mainly used when the data set D is large.

Evaluation methods (cont...)

- **n-fold cross-validation**: The available data is partitioned into n equal-size disjoint subsets.
- Use each subset as the test set and combine the rest $n-1$ subsets as the training set to learn a classifier.
- The procedure is run n times, which give n accuracies.
- The final estimated accuracy of learning is the average of the n accuracies.
- 10-fold and 5-fold cross-validations are commonly used.
- This method is used when the available data is not large.

Evaluation methods (cont...)

- **Leave-one-out cross-validation**: This method is used when the data set is very small.
- It is a special case of cross-validation
- Each fold of the cross validation has only **a single test example** and all the rest of the data is used in training.
- If the original data has m examples, this is **m -fold cross-validation**

Evaluation methods (cont...)

- **Validation set:** the available data is divided into three subsets,
 - a training set,
 - a validation set and
 - a test set.
- A validation set is used frequently for estimating parameters in learning algorithms.
- In such cases, the values that give the best accuracy on the validation set are used as the final parameter values.
- Cross-validation can be used for parameter estimating as well.

Classification measures

- Accuracy is only one measure ($\text{error} = 1 - \text{accuracy}$).
- **Accuracy is not suitable in some applications.**
- In text mining, we may only be interested in the documents of a particular topic, which are only a small portion of a big document collection.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, **we are interested only in the minority class.**
 - High accuracy does not mean any intrusion is detected.
 - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes**.

Precision and recall measures

- Used in information retrieval and text classification.
- We use a confusion matrix to introduce them.

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

where

TP: the number of correct classifications of the positive examples (**true positive**),

FN: the number of incorrect classifications of positive examples (**false negative**),

FP: the number of incorrect classifications of negative examples (**false positive**), and

TN: the number of correct classifications of negative examples (**true negative**).

Precision and recall measures (cont...)

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

$$p = \frac{TP}{TP + FP}.$$

$$r = \frac{TP}{TP + FN}.$$

- **Precision p** is the number of **correctly classified positive examples** divided by the total number of examples that are classified as positive.
- **Recall r** is the number of **correctly classified positive examples** divided by the total number of actual positive examples in the test set.

An example

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

This confusion matrix gives

precision $p = 100\%$ and

recall $r = 1\%$

because we only classified one positive example correctly and no negative examples wrongly.

Note: precision and recall only measure classification on the positive class.

F_1 -value (also called F_1 -score)

It is hard to compare two classifiers using two measures. F_1 score combines precision and recall into one measure

$$F_1 = \frac{2pr}{p+r}$$

F_1 -score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two.

For F_1 -value to be large, both p and r must be large.

Receive operating characteristics curve

It is commonly called the **ROC curve**.

It is a plot of the **true positive rate (TPR)** against the **false positive rate (FPR)**.

True positive rate:

$$TPR = \frac{TP}{TP + FN}$$

False positive rate:

$$FPR = \frac{FP}{TN + FP}$$

Sensitivity and Specificity

In statistics, there are two other evaluation measures:

Sensitivity: Same as TPR

Specificity: Also called True Negative Rate (TNR)

Then we have

$$TNR = \frac{TN}{TN + FP}$$

$$FPR = 1 - specificity$$

Example ROC curves

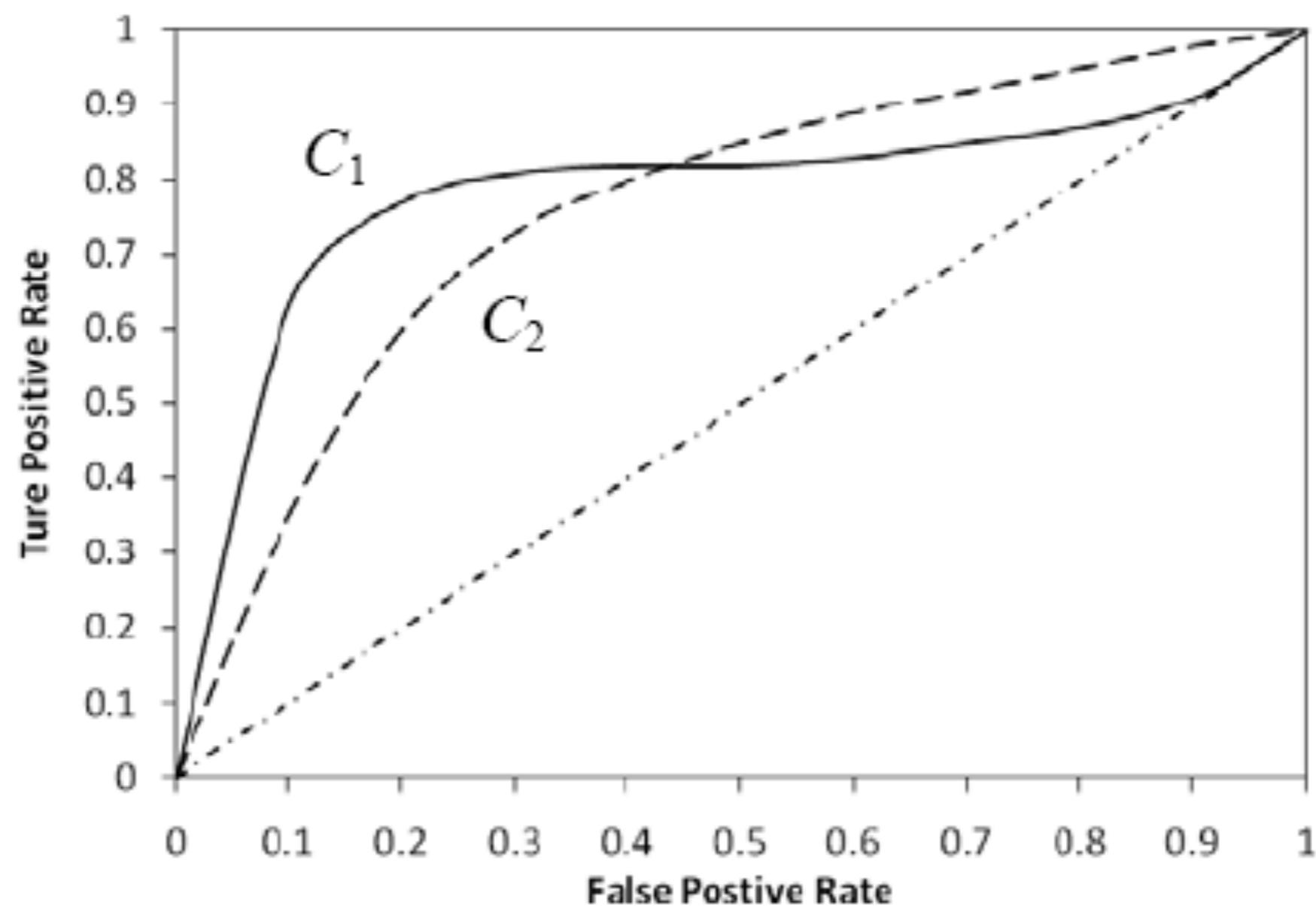


Fig. 3.8. ROC curves for two classifiers (C_1 and C_2) on the same data

Area under the curve (AUC)

Which classifier is better, C_1 or C_2 ?

It depends on which region you talk about.

Can we have one measure?

Yes, we compute the area under the curve (AUC)

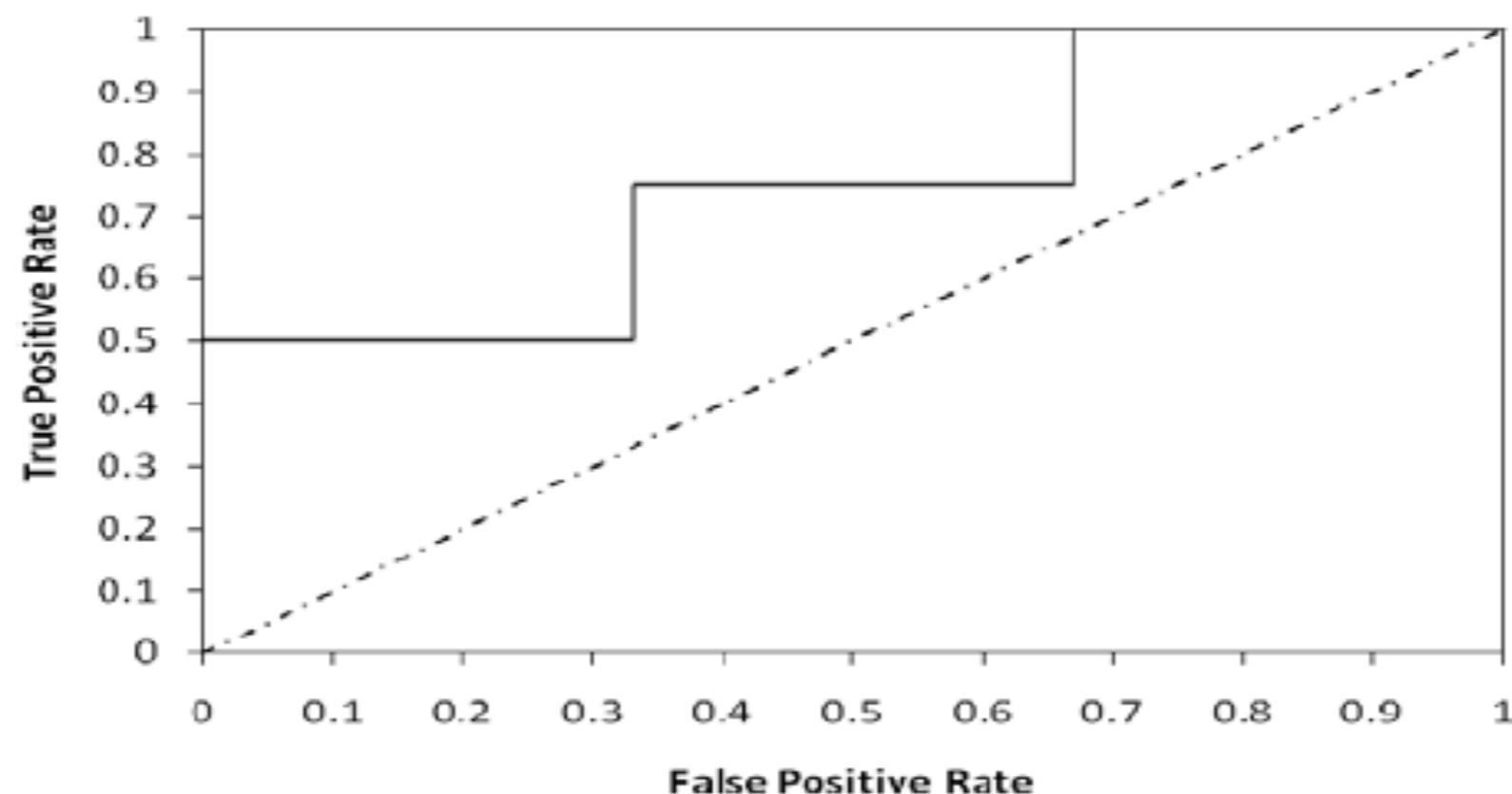
If AUC for C_i is greater than that of C_j , it is said that C_i is better than C_j .

If a classifier is perfect, its AUC value is 1

If a classifier makes all random guesses, its AUC value is 0.5.

Drawing an ROC curve

Rank	1	2	3	4	5	6	7	8	9	10
Actual class	+	+	-	-	+	-	-	+	-	-
TP	1	2	2	2	3	3	3	4	4	4
FP	0	0	0	1	2	2	3	4	4	5
TN	6	6	6	5	4	4	3	2	1	0
FN	4	3	2	2	2	1	1	0	0	0
TPR	0.25	0.5	0.5	0.5	0.75	0.75	0.75	1	1	1
FPR	0	0	0	0.17	0.33	0.33	0.50	0.67	0.67	0.83



Evaluation of Classifier in R

Confusion Matrix

```
install.packages("caret")  
library(caret)  
confusionMatrix(testPred, testdata$Species)
```

Confusion Matrix and Statistics			
Prediction	Reference		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	15	0	0
Iris-versicolor	0	13	2
Iris-virginica	0	0	13
Overall Statistics			
Accuracy : 0.9535			
95% CI : (0.8419, 0.9943)			
No Information Rate : 0.3488			
P-Value [Acc > NIR] : < 2.2e-16			
Kappa : 0.9303			
McNemar's Test P-Value : NA			

Statistics by Class:

	Class: Iris-setosa	Class: Iris-versicolor	Class: Iris-virginica
Sensitivity	1.0000	1.0000	0.8667
Specificity	1.0000	0.9333	1.0000
Pos Pred Value	1.0000	0.8667	1.0000
Neg Pred Value	1.0000	1.0000	0.9333
Prevalence	0.3488	0.3023	0.3488
Detection Rate	0.3488	0.3023	0.3023
Detection Prevalence	0.3488	0.3488	0.3023
Balanced Accuracy	1.0000	0.9667	0.9333

Topic

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- **Naïve Bayesian classification**
- K-nearest neighbor
- Support Vector Machine
- Neural Net
- Ensemble methods: Bagging and Boosting
- Summary

Bayesian classification

Probabilistic view: Supervised learning can naturally be studied from a probabilistic point of view.

Let A_1 through A_k be attributes with discrete values. The class is C .

Given a test example d with observed attribute values a_1 through a_k .

Classification is basically to compute the following posteriori probability.

The prediction is the class c_j such that

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$

is maximal

Apply Bayes' Rule

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$
$$= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) \Pr(C = c_j)}{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|})}$$
$$= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) \Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_r) \Pr(C = c_r)}$$

- $\Pr(C=c_j)$ is the class prior probability: easy to estimate from the training data.

Computing probabilities

The denominator $P(A_1=a_1, \dots, A_k=a_k)$ is irrelevant for decision making since it is the same for every class.

We only need $P(A_1=a_1, \dots, A_k=a_k | C=c_i)$, which can be written as

$$Pr(A_1=a_1 | A_2=a_2, \dots, A_k=a_k, C=c_j) * Pr(A_2=a_2, \dots, A_k=a_k | C=c_j)$$

Recursively, the second factor above can be written in the same way, and so on.

Now an assumption is needed.

Conditional independence assumption

All attributes are conditionally independent given the class $C = c_j$.

Formally, we assume,

$$\Pr(A_1=a_1 \mid A_2=a_2, \dots, A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$$

and so on for A_2 through $A_{|A|}$. I.e.,

$$\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_i) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

Final naïve Bayesian classifier

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|}) \\ = \frac{\Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)}$$

Classify a test instance

If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class. Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg \max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

An example

- Compute all probabilities required for classification

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

$$\Pr(C = t) = 1/2,$$

$$\Pr(C = f) = 1/2$$

$$\Pr(A=m \mid C=t) = 2/5$$

$$\Pr(A=g \mid C=t) = 2/5$$

$$\Pr(A=h \mid C=t) = 1/5$$

$$\Pr(A=m \mid C=f) = 1/5$$

$$\Pr(A=g \mid C=f) = 2/5$$

$$\Pr(A=h \mid C=f) = 2/5$$

$$\Pr(B=b \mid C=t) = 1/5$$

$$\Pr(B=s \mid C=t) = 2/5$$

$$\Pr(B=q \mid C=t) = 2/5$$

$$\Pr(B=b \mid C=f) = 2/5$$

$$\Pr(B=s \mid C=f) = 1/5$$

$$\Pr(B=q \mid C=f) = 2/5$$

Now we have a test example:

$$A = m \quad B = q \quad C = ?$$

An Example (cont ...)

For $C = t$, we have

$$\Pr(C = t) \prod_{j=1}^2 \Pr(A_j = a_j | C = t) = \frac{1}{2} \times \frac{2}{5} \times \frac{2}{5} = \frac{2}{25}$$

For class $C = f$, we have

$$\Pr(C = f) \prod_{j=1}^2 \Pr(A_j = a_j | C = f) = \frac{1}{2} \times \frac{1}{5} \times \frac{2}{5} = \frac{1}{25}$$

$C = t$ is more probable. t is the final class.

Additional issues

Numeric attributes: Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.

Zero counts: An particular attribute value never occurs together with a class in the training set. We need smoothing.

Missing values: Ignored

$$\Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda n_i}$$

On naïve Bayesian classifier

Advantages:

Easy to implement

Very efficient

Good results obtained in many applications

Disadvantages

Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

naïve Bayesian classifier in R

Naive Bayes

Naive Bayes Classifier

Description

Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

Usage

```
## S3 method for class 'formula'  
naiveBayes(formula, data, laplace = 0, ..., subset, na.action = na.pass)  
## Default S3 method:  
naiveBayes(x, y, laplace = 0, ...)  
  
## S3 method for class 'naiveBayes'  
predict(object, newdata,  
       type = c("class", "raw"), threshold = 0.001, eps = 0, ...)
```

Naive Bayes

Arguments

- x** A numeric matrix, or a data frame of categorical and/or numeric variables.
- y** Class vector.
- formula** A formula of the form `class ~ x1 + x2 + ...`. Interactions are not allowed.
- data** Either a data frame of predictors (categorical and/or numeric) or a contingency table.
- laplace** positive double controlling Laplace smoothing. The default (0) disables Laplace smoothing.
- ...** Currently not used.
- subset** For data given in a data frame, an index vector specifying the cases to be used in the training sample.
(NOTE: If given, this argument must be named.)
- na.action** A function to specify the action to be taken if NAs are found. The default action is not to count them for the computation of the probability factors. An alternative is `na.omit`, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)
- object** An object of class "naiveBayes".
- newdata** A data frame with new predictors (with possibly fewer columns than the training data). Note that the column names of `newdata` are matched against the training data ones.
- type** If "raw", the conditional a-posterior probabilities for each class are returned, and the class with maximal probability else.
- threshold** Value replacing cells with probabilities within `eps` range.
- eps** double for specifying an epsilon-range to apply laplace smoothing (to replace zero or close-zero probabilities by `threshold`.)

Naive Bayes

```
> library(e1071)
> # Can handle both categorical and numeric input,
> # but output must be categorical
> model <- naiveBayes(Species~., data=iristrain)
> prediction <- predict(model, iristest[,-5])
> table(prediction, iristest[,5])
```

prediction	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	10	2
virginica	0	0	8

```
> confusionMatrix(prediction, testdata$Species)
```

Confusion Matrix and Statistics

		Reference		
		Iris-setosa	Iris-versicolor	Iris-virginica
Prediction				
Iris-setosa		15	0	0
Iris-versicolor		0	13	2
Iris-virginica		0	0	13

Overall Statistics

Accuracy : 0.9535

95% CI : (0.8419, 0.9943)

No Information Rate : 0.3488

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9303

McNemar's Test P-Value : NA

Statistics by Class:

	Class: Iris-setosa	Class: Iris-versicolor	Class: Iris-virginica
--	--------------------	------------------------	-----------------------

Sensitivity	1.0000	1.0000	0.8667
Specificity	1.0000	0.9333	1.0000
Pos Pred Value	1.0000	0.8667	1.0000
Neg Pred Value	1.0000	1.0000	0.9333
Prevalence	0.3488	0.3023	0.3488
Detection Rate	0.3488	0.3023	0.3023
Detection Prevalence	0.3488	0.3488	0.3023
Balanced Accuracy	1.0000	0.9667	0.9333

Workshop 12 - Naive Bayes Classifier

1. From the data that you import
2. Perform Naive Bayes Classifier Algorithm to predict target Y
3. Perform performance management by creating confusion matrix

Topic

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Naïve Bayesian classification
- **K-nearest neighbor**
- Support Vector Machine
- Neural Net
- Ensemble methods: Bagging and Boosting
- Summary

k-Nearest Neighbor Classification (kNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance d , define k -neighborhood P as k nearest neighbors of d
- Count number n of training instances in P that belong to class c_j
- Estimate $\Pr(c_j|d)$ as n/k
- No training is needed. Classification time is linear in training set size for each test case.

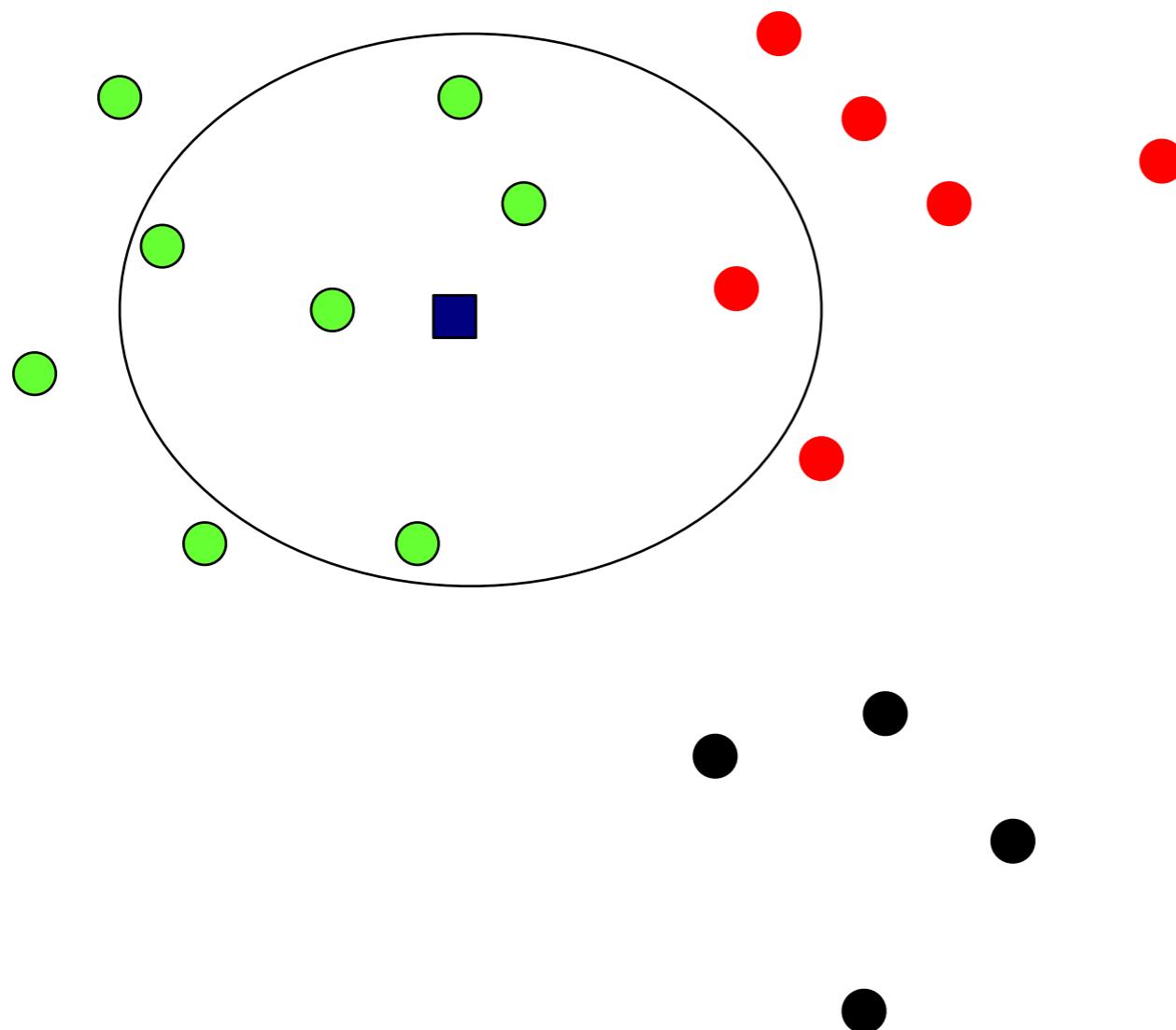
kNNAlgorithm

Algorithm kNN(D, d, k)

- 1 Compute the distance between d and every example in D ;
- 2 Choose the k examples in D that are nearest to d , denote the set by P ($\subseteq D$);
- 3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- *Distance function* is crucial, but depends on applications.

Example: k=6 (6NN)



- Government
- Science
- Arts

A new point ■
 $\Pr(\text{science} | \text{■})?$

Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model



k-NN in R

Fundamental Data Science for Data Scientist

K-NN

k-Nearest Neighbour Classification

Description

k-nearest neighbour classification for test set from training set. For each row of the test set, the `k` nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the `k`th nearest vector, all candidates are included in the vote.

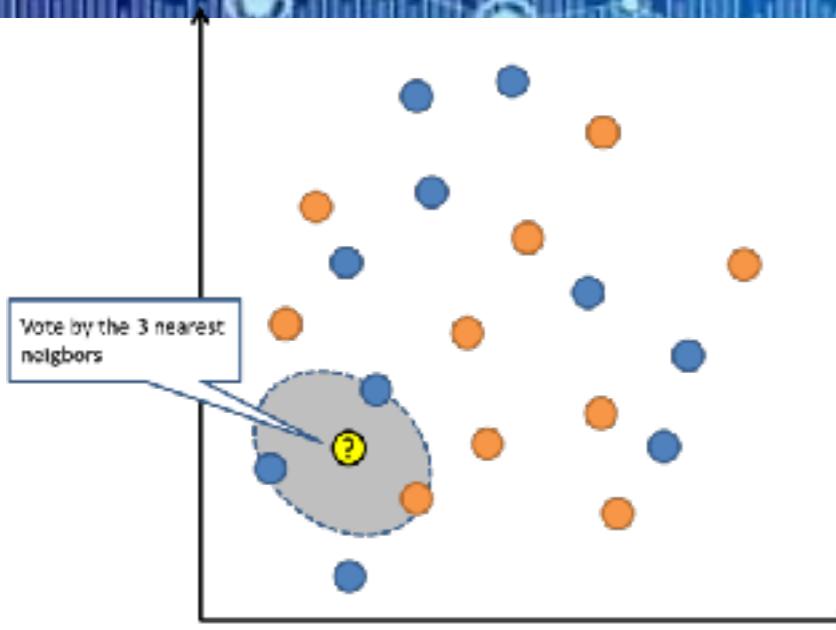
Usage

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

Arguments

- `train` matrix or data frame of training set cases.
- `test` matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
- `cl` factor of true classifications of training set
- `k` number of neighbours considered.
- `l` minimum vote for definite decision, otherwise doubt. (More precisely, less than `k-1` dissenting votes are allowed, even if `k` is increased by ties.)
- `prob` If this is true, the proportion of the votes for the winning class are returned as attribute `prob`.
- `use.all` controls handling of ties. If true, all distances equal to the `k`th largest are included. If false, a random selection of distances equal to the `k`th is chosen to use exactly `k` neighbours.

K-NN



```
> library(class)
> train_input <- as.matrix(iristrain[,-5])
> train_output <- as.vector(iristrain[,5])
> test_input <- as.matrix(iristest[,-5])
> prediction <- knn(train_input, test_input,
+                      train_output, k=5)
> table(prediction, iristest$Species)

prediction   setosa versicolor virginica
  setosa        10          0          0
  versicolor     0         10          1
  virginica      0          0          9
>
```

```
> confusionMatrix(prediction, testdata$Species)
```

Confusion Matrix and Statistics

Prediction	Reference		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	15	0	0
Iris-versicolor	0	12	0
Iris-virginica	0	1	15

Overall Statistics

Accuracy : 0.9767
95% CI : (0.8771, 0.9994)

No Information Rate : 0.3488

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.965

McNemar's Test P-Value : NA

Statistics by Class:

	myFormula		
	Class: Iris-setosa	Class: Iris-versicolor	Class: Iris-virginica
Sensitivity	1.0000	0.9231	1.0000
Specificity	1.0000	1.0000	0.9643
Pos Pred Value	1.0000	1.0000	0.9375
Neg Pred Value	1.0000	0.9677	1.0000
Prevalence	0.3488	0.3023	0.3488
Detection Rate	0.3488	0.2791	0.3488
Detection Prevalence	0.3488	0.2791	0.3721
Balanced Accuracy	1.0000	0.9615	0.9821

Workshop 13 - k-NN

1. From the data that you import
2. Perform k-NN Algorithm to predict target
3. Perform performance management by creating confusion matrix

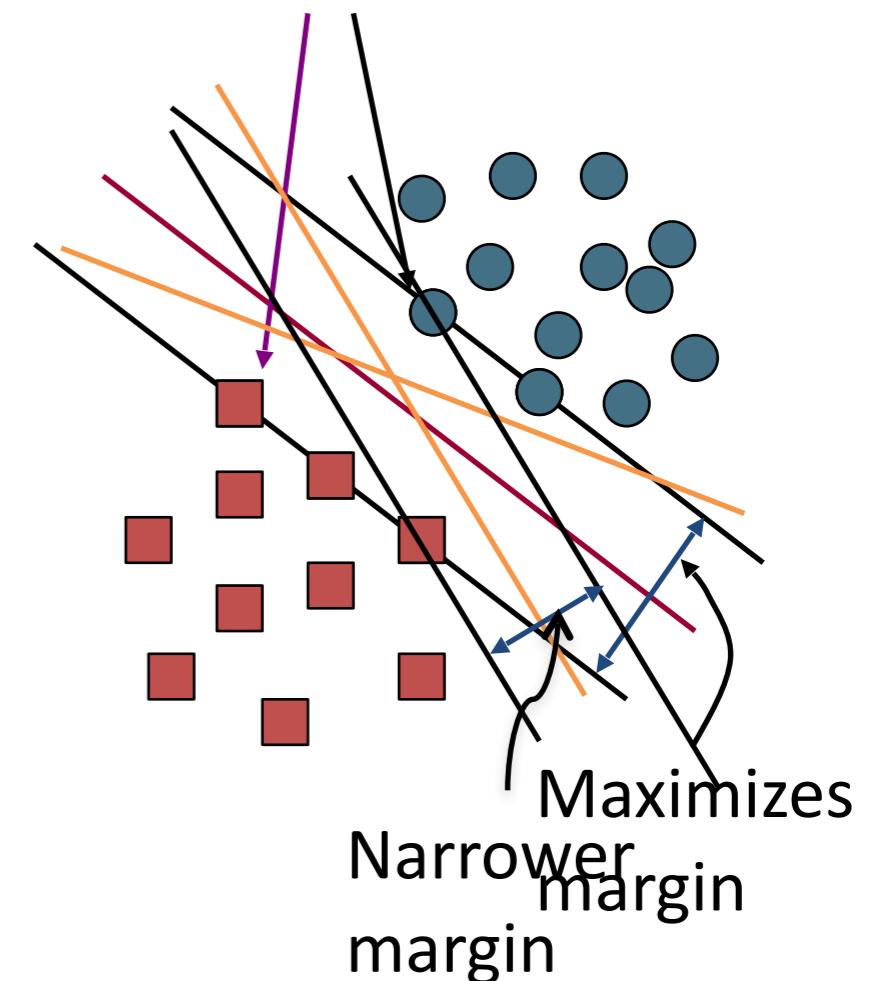
Topic

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Naïve Bayesian classification
- K-nearest neighbor
- **Support Vector Machine**
- Neural Net
- Ensemble methods: Bagging and Boosting
- Summary

Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
 - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*

Support vectors



Maximum Margin: Formalization

\mathbf{w} : decision hyperplane normal vector

\mathbf{x}_i : data point i

y_i : class of data point i (+1 or -1) NB: Not 1/0

Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$

Functional margin of \mathbf{x}_i is: $y_i (\mathbf{w}^\top \mathbf{x}_i + b)$

But note that we can increase this margin simply by scaling \mathbf{w}, \mathbf{b}

Functional margin of dataset is twice the minimum functional margin for any point

The factor of 2 comes from measuring the whole width of the margin

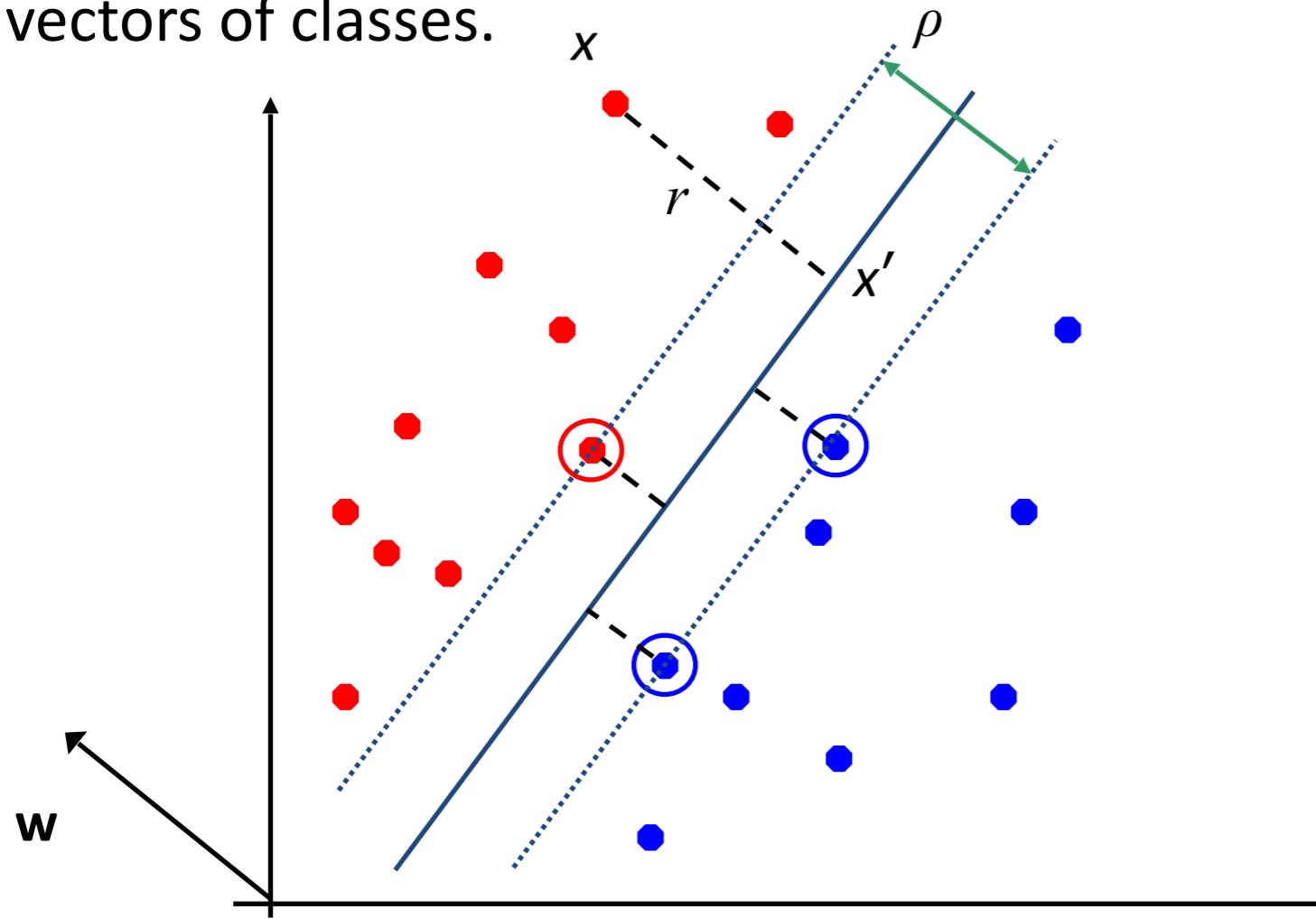
Geometric Margin

Distance from example to the separator is

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

Examples closest to the hyperplane are **support vectors**.

Margin ρ of the separator is the width of separation between support vectors of classes.



Derivation of finding r :

Dotted line $\mathbf{x}' - \mathbf{x}$ is perpendicular to decision boundary so parallel to \mathbf{w} . Unit vector is $\mathbf{w}/|\mathbf{w}|$, so line is $r\mathbf{w}/|\mathbf{w}|$.

$$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|.$$

$$\mathbf{x}' \text{ satisfies } \mathbf{w}^T \mathbf{x}' + b = 0.$$

$$\text{So } \mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$$

$$\text{Recall that } |\mathbf{w}| = \sqrt{\mathbf{w}^T \mathbf{w}}.$$

$$\text{So } \mathbf{w}^T \mathbf{x} - yr|\mathbf{w}| + b = 0$$

So, solving for r gives:

$$r = y(\mathbf{w}^T \mathbf{x} + b)/|\mathbf{w}|$$

Linear SVM Mathematically

The linearly separable case

Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x}_i, y_i)\}$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

For support vectors, the inequality becomes an equality

Then, since each example's distance from the hyperplane is

$$r = y \frac{\|\mathbf{w}\|}{\|\mathbf{w}^T \mathbf{x} + b\|}$$

The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

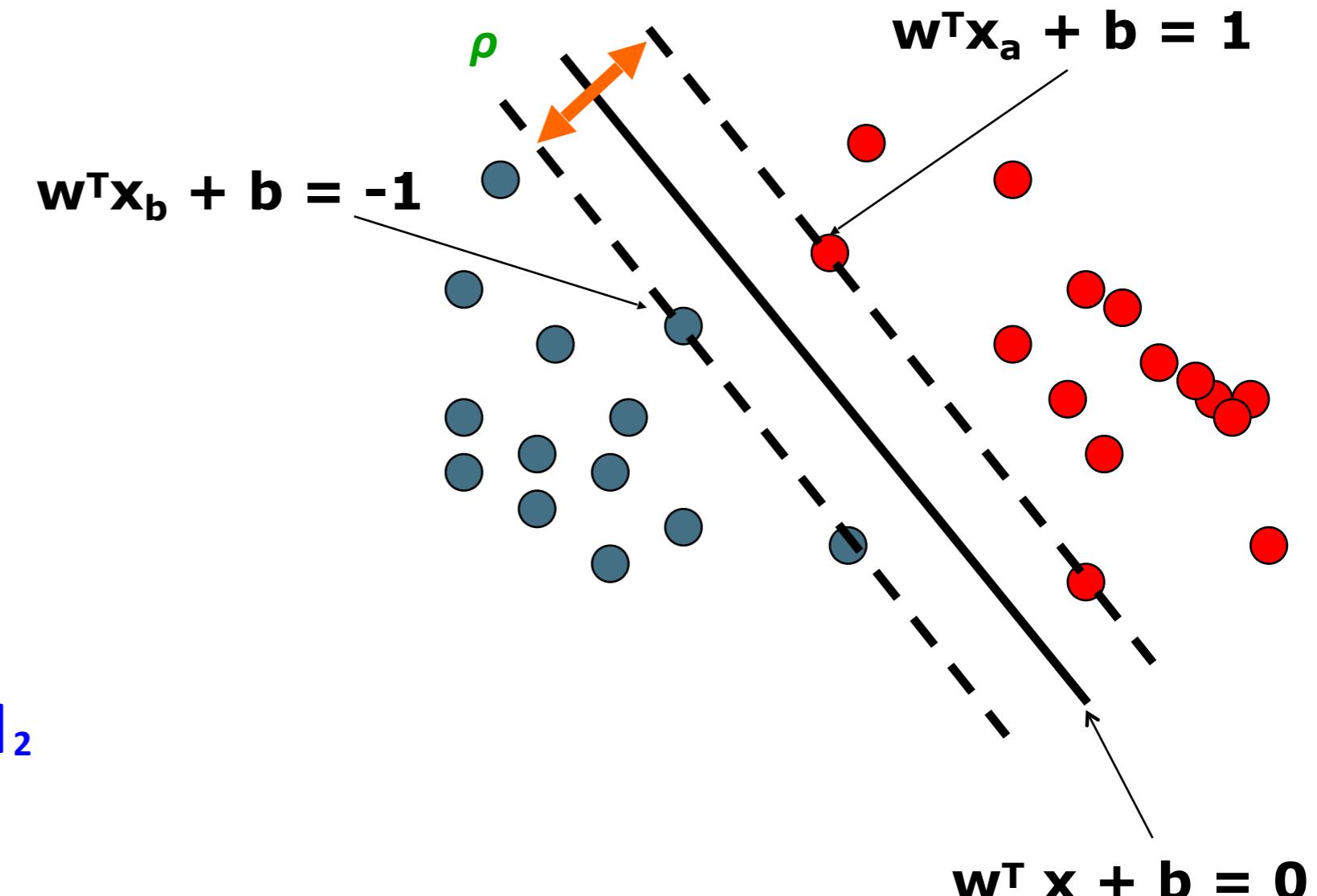
Hyperplane
 $\mathbf{w}^T \mathbf{x} + b = 0$

Extra scale constraint:
 $\min_{i=1,\dots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$

This implies:

$$\mathbf{w}^T (\mathbf{x}_a - \mathbf{x}_b) = 2$$

$$\rho = \|\mathbf{x}_a - \mathbf{x}_b\|_2 = 2/\|\mathbf{w}\|_2$$





Support Vector Machine in R

Support Vector Machine

svm {e1071}

R Documentation

Support Vector Machines

Description

`svm` is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

Usage

```
## S3 method for class 'formula'  
svm(formula, data = NULL, ..., subset, na.action =  
na.omit, scale = TRUE)  
## Default S3 method:  
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =  
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),  
coef0 = 0, cost = 1, nu = 0.5,  
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,  
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,  
..., subset, na.action = na.omit)
```

Arguments

formula

a symbolic description of the model to be fit.

data

an optional data frame containing the variables in the model. By default the variables are taken from the environment which 'svm' is called from.

x

a data matrix, a vector, or a sparse matrix (object of class [Matrix](#) provided by the [Matrix](#) package, or of class [matrix.csr](#) provided by the [SparseM](#) package, or of class [simple_triplet_matrix](#) provided by the [slam](#) package).

y

a response vector with one label for each row/component of `x`. Can be either a factor (for classification tasks) or a numeric vector (for regression).

Support Vector Machine

```
> library(e1071)
> tune <- tune.svm(Species~., data=iristrain, gamma=10^(-6:-1), cost=10^(1:4))
> summary(tune)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
  gamma  cost
 0.001 10000
- best performance: 0.03333333
> model <- svm(Species~., data=iristrain, method="C-classification",
kernel="radial", probability=T, gamma=0.001, cost=10000)
> prediction <- predict(model, iristest, probability=T)
> table(iristest$Species, prediction)

  prediction
            setosa versicolor virginica
setosa          10         0         0
versicolor        0         10         0
virginica         0          3         7
>
```

Workshop 14 - SVM

1. From the data that you import
2. Perform SVM Algorithm to predict target Y
3. Perform performance management by creating confusion matrix

Topic

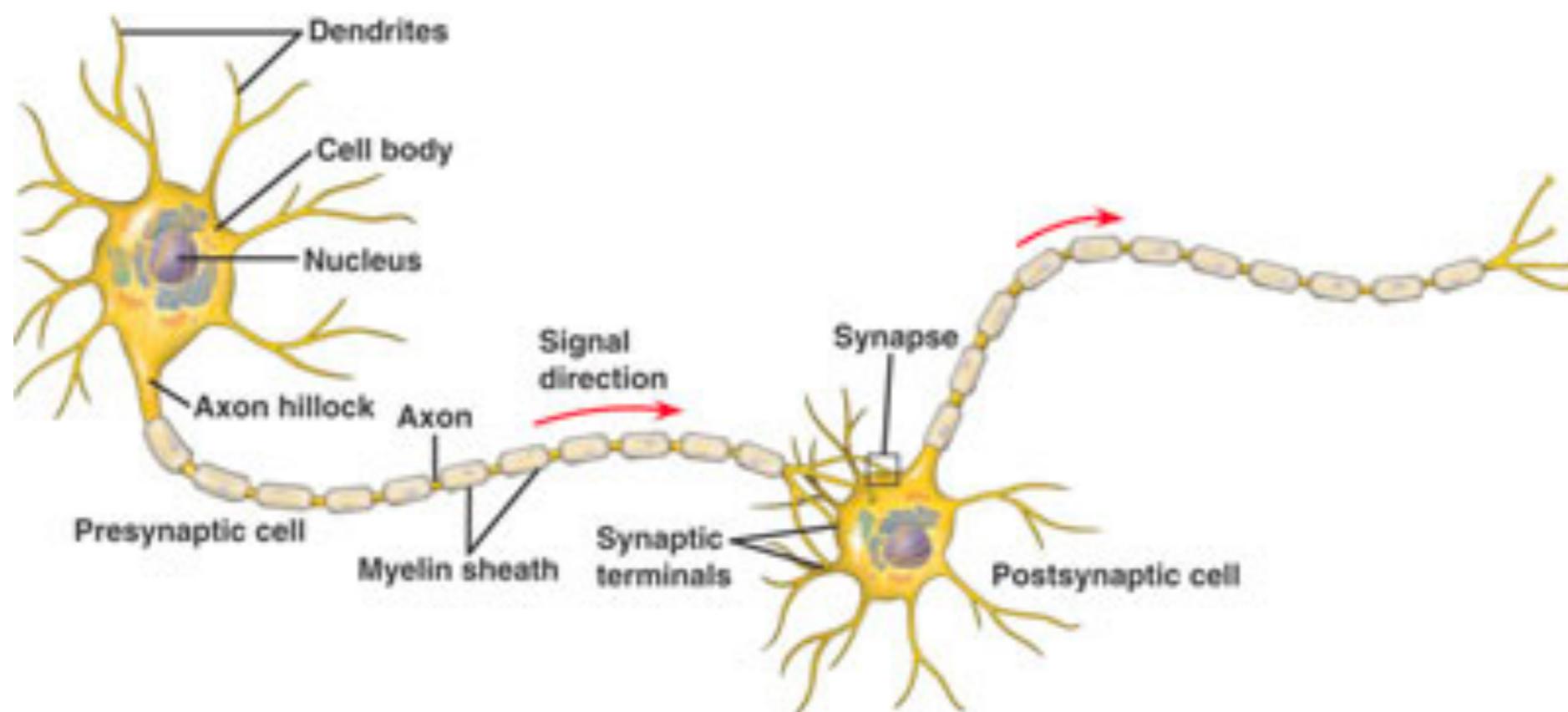
- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Naïve Bayesian classification
- K-nearest neighbor
- Support Vector Machine
- **Neural Net**
- Ensemble methods: Bagging and Boosting
- Summary

Neural Network

- Neural networks learning methods provide a robust approach to approximating real-valued, discrete valued and vector valued target functions.
- Often used in problems such as handwriting, voice, or image recognition
- **Feed-Forward Neural Networks, Convolutional Neural Networks, Deep Learning**

Biological Motivation

- Inspired by biological learning systems that are built from very complex webs of interconnected neurons

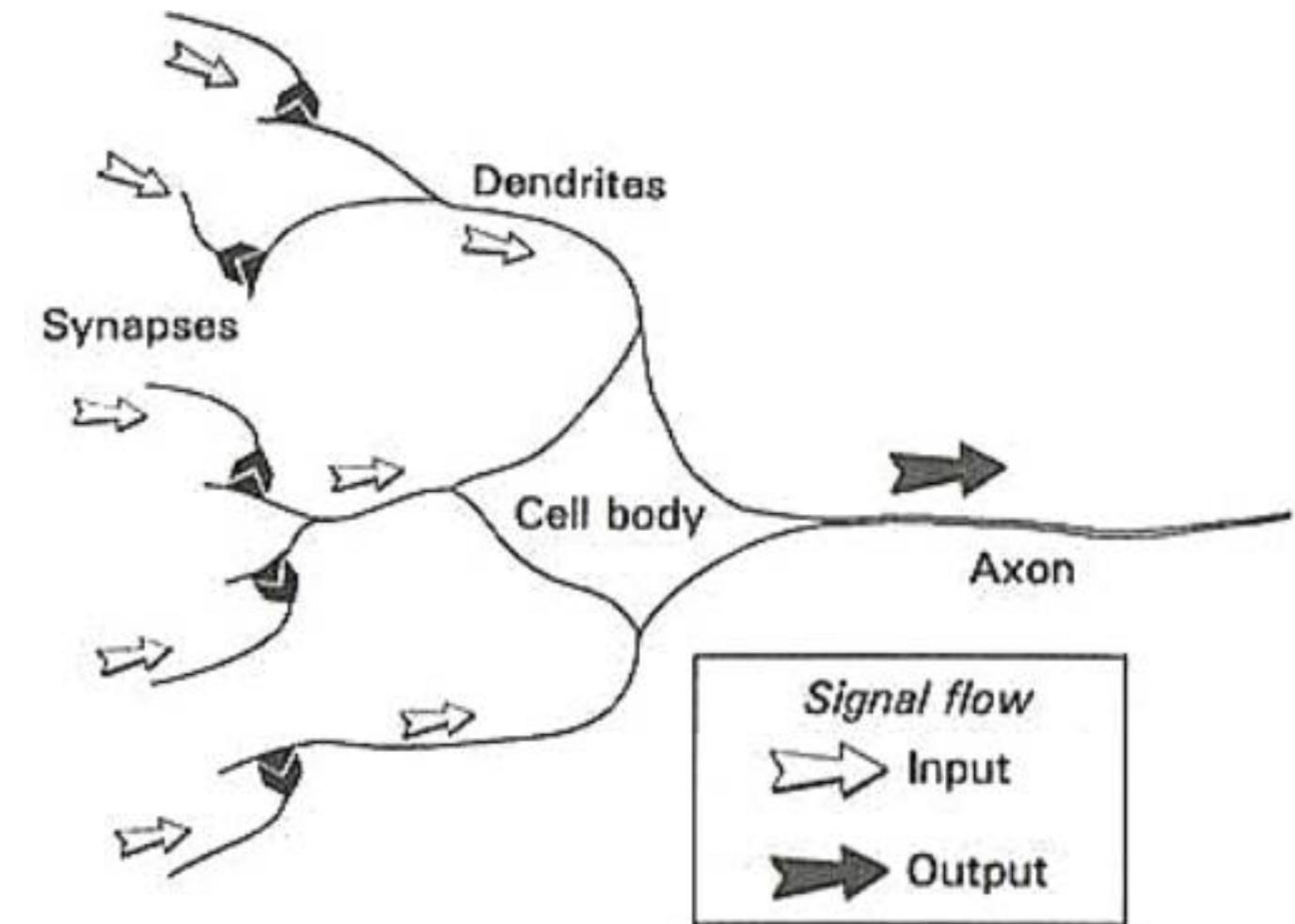


Facts About Neuro-biology

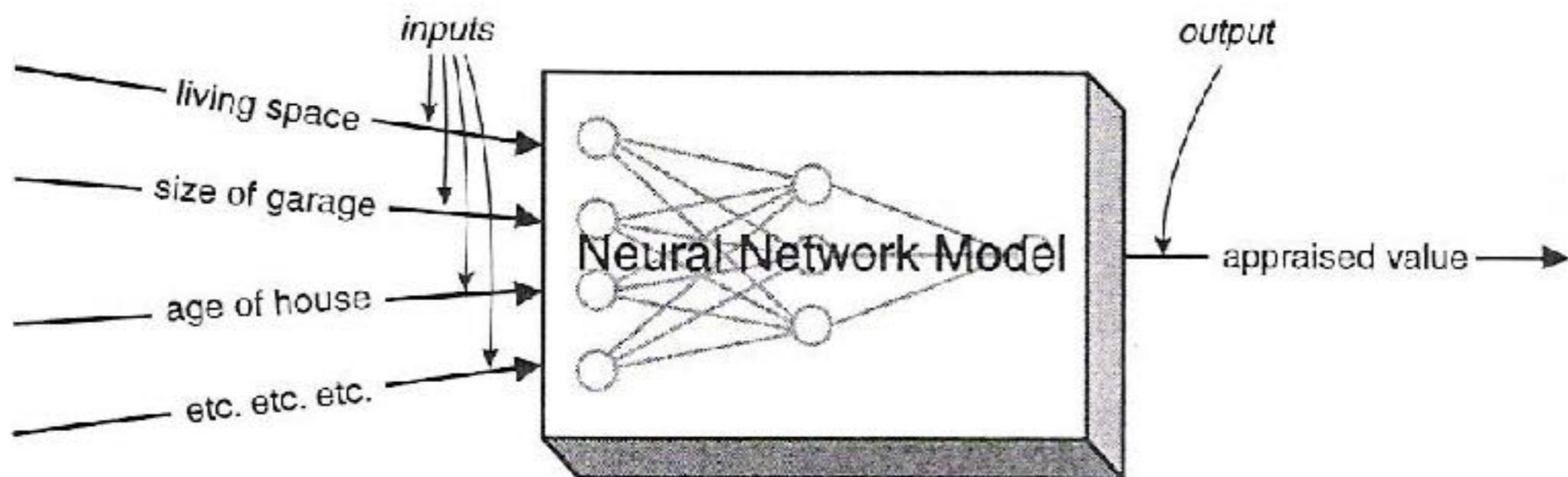
- Human brain is estimated to contain a densely interconnected network of approximately 10^{11} neurons.
- Each connected, on average, to 10^4 other neurons.
- Neuron activity is typically excited through connections to other neurons.
- The fastest neuron switching times are in the order of 10^{-3} seconds.
- It requires 10^{-1} seconds to visibly recognize your mother.

Analogy of Biological Learning Systems

- Artificial neural networks are built out of a interconnected set of simple units, where each unit takes a number of real-valued inputs (can be the outputs of other units) and produces a single real-valued output (which may become the input of other units).



Example

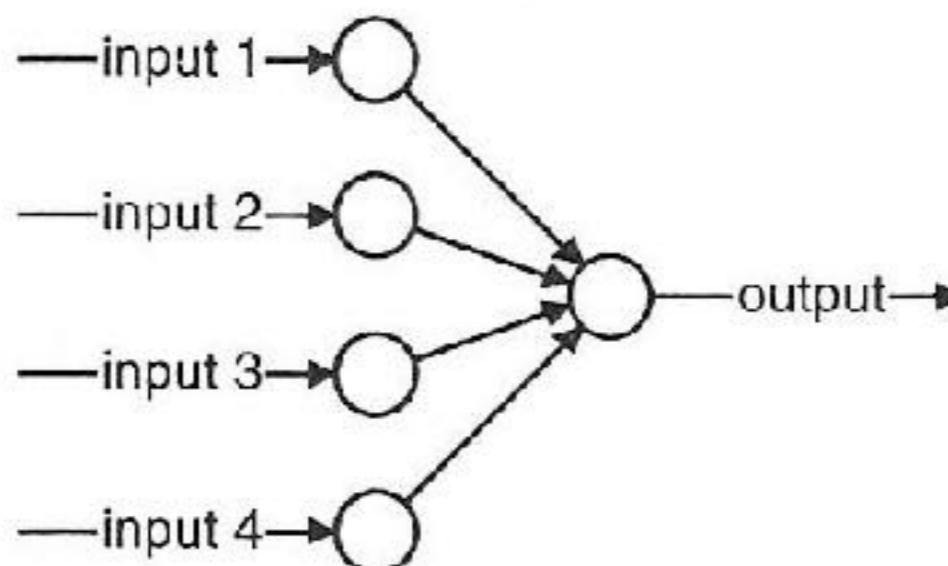


A neural network is like a black box that know how to process input to create an output. The calculation is quite complex and difficult to understand.

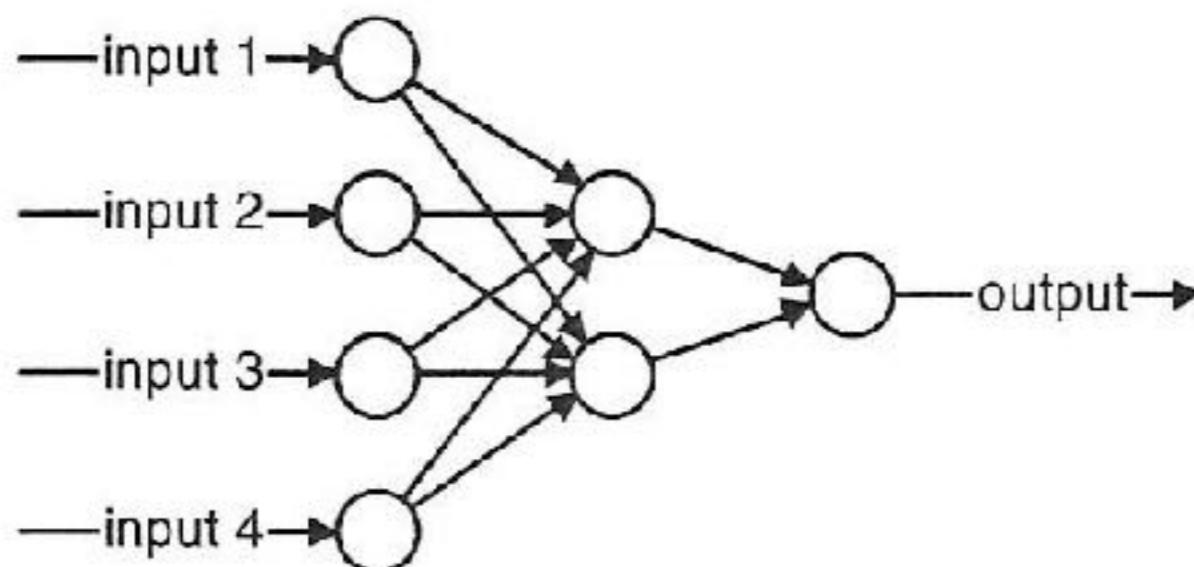
Neural Networks Process

1. Identify the input and output features
2. Transform the input and output so that they are in a small range
(-1 to 1)
3. Set up a network with an appropriate topology
4. Train the network on a representative set of training examples
5. Use the validation set to choose the set of weights that minimizes the error
6. Evaluate the network using the test set to see how well it performs
7. Apply the model generated by the network to predict outcomes for unknown input

What is a Neural Network? (1)

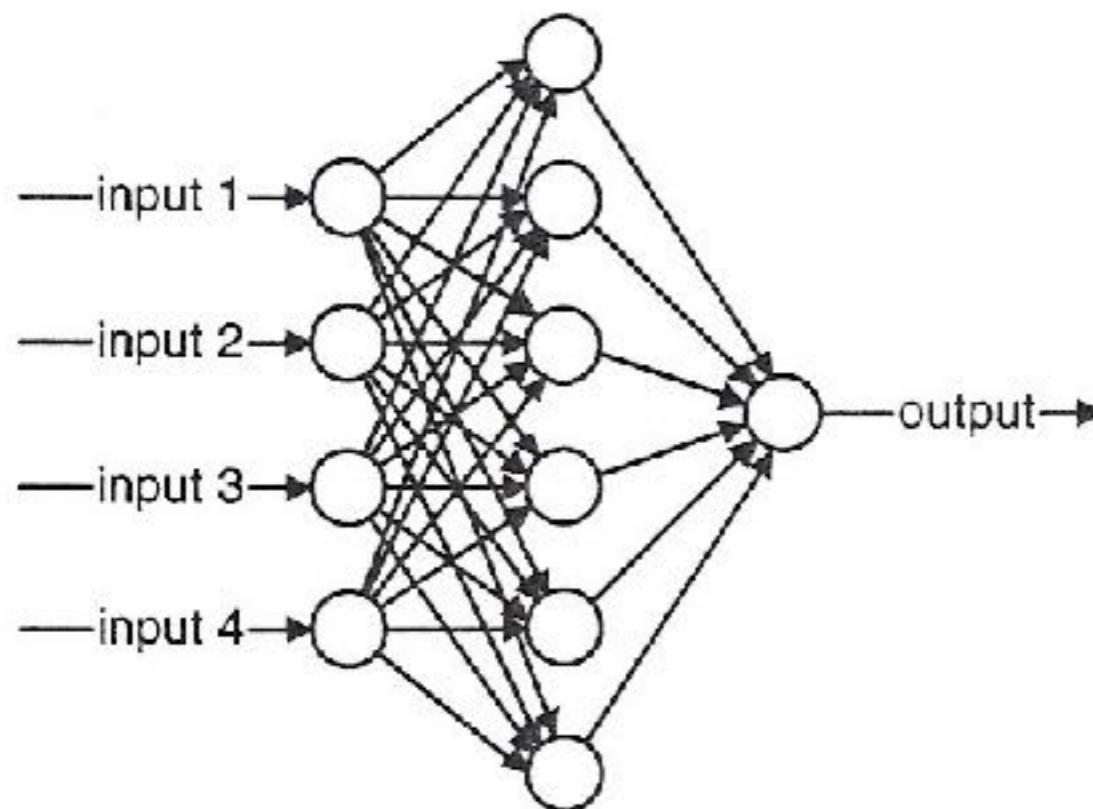


This simple neural network takes four inputs and produces an output. This result of training this network is equivalent to the statistical technique called logistic regression.

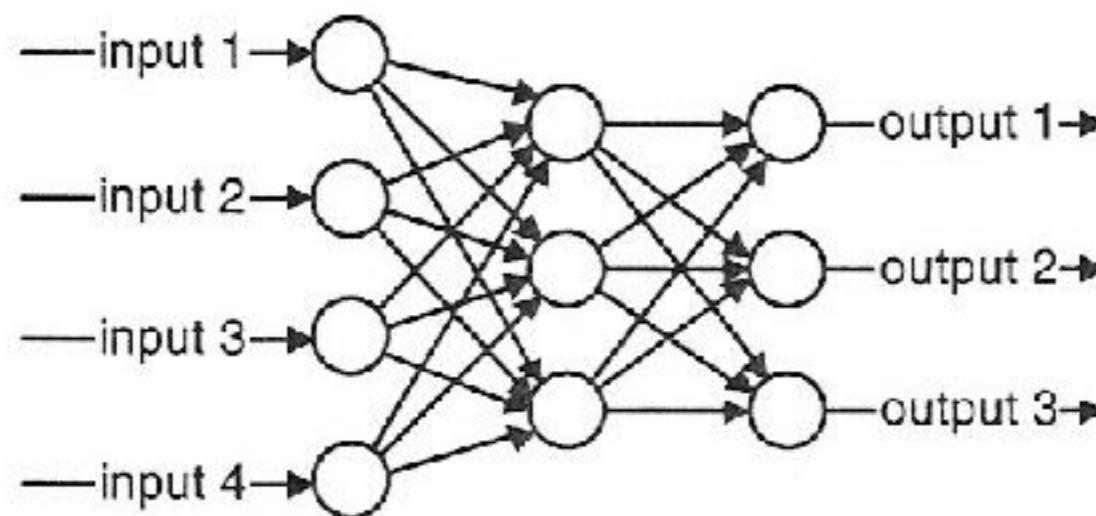


This network has a middle layer called the *hidden layer*, which makes the network more powerful by enabling it to recognize more patterns.

What is a Neural Network? (2)



Increasing the size of the hidden layer makes the network more powerful but introduces the risk of overfitting. Usually, only one hidden layer is needed.



A neural network can produce multiple output values.

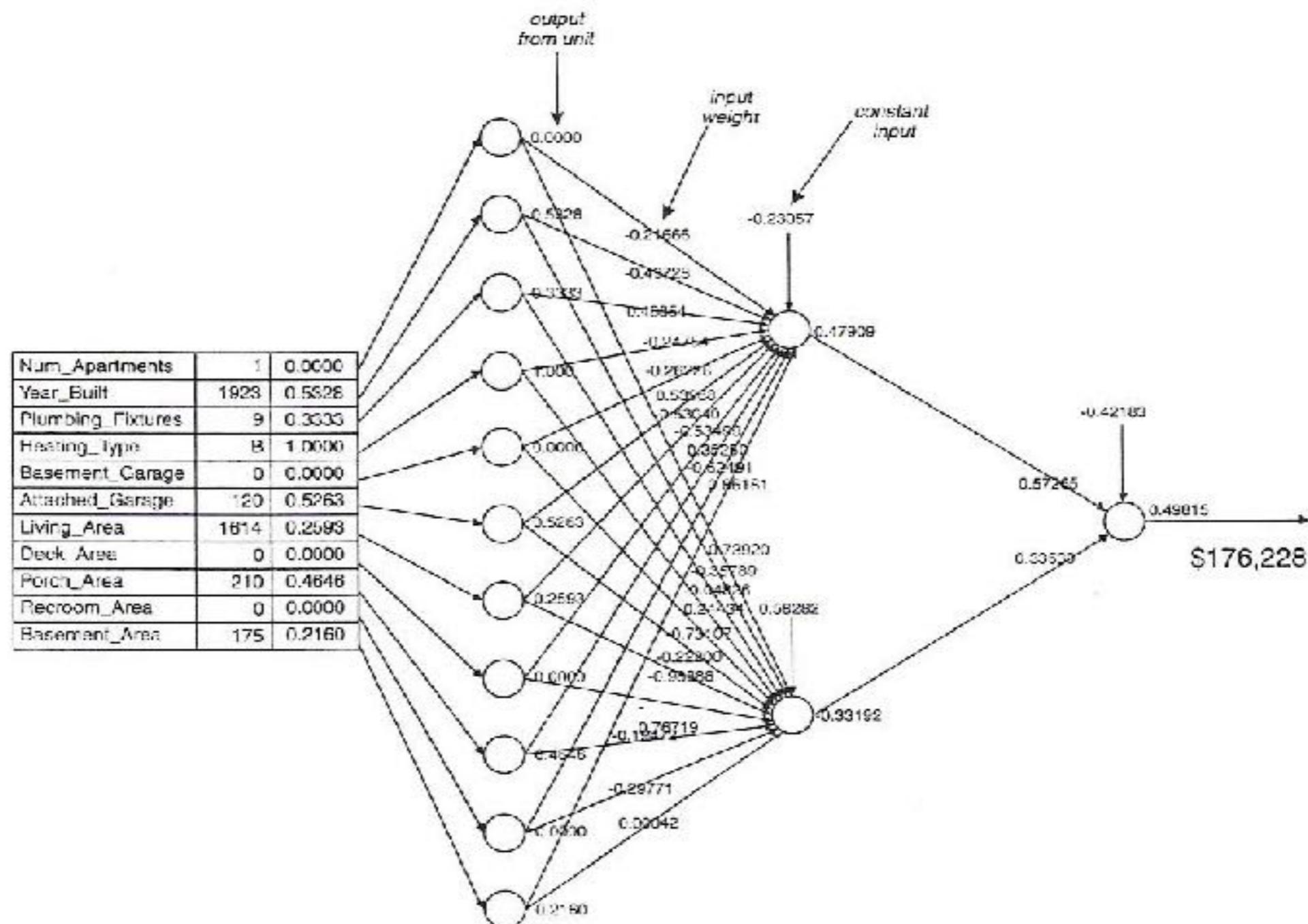
Three Main Components of a Neural Network

- Activation function
- Network topology
- How the network is trained?
 - Feed forward
 - Back propagation

Activation Function (1)

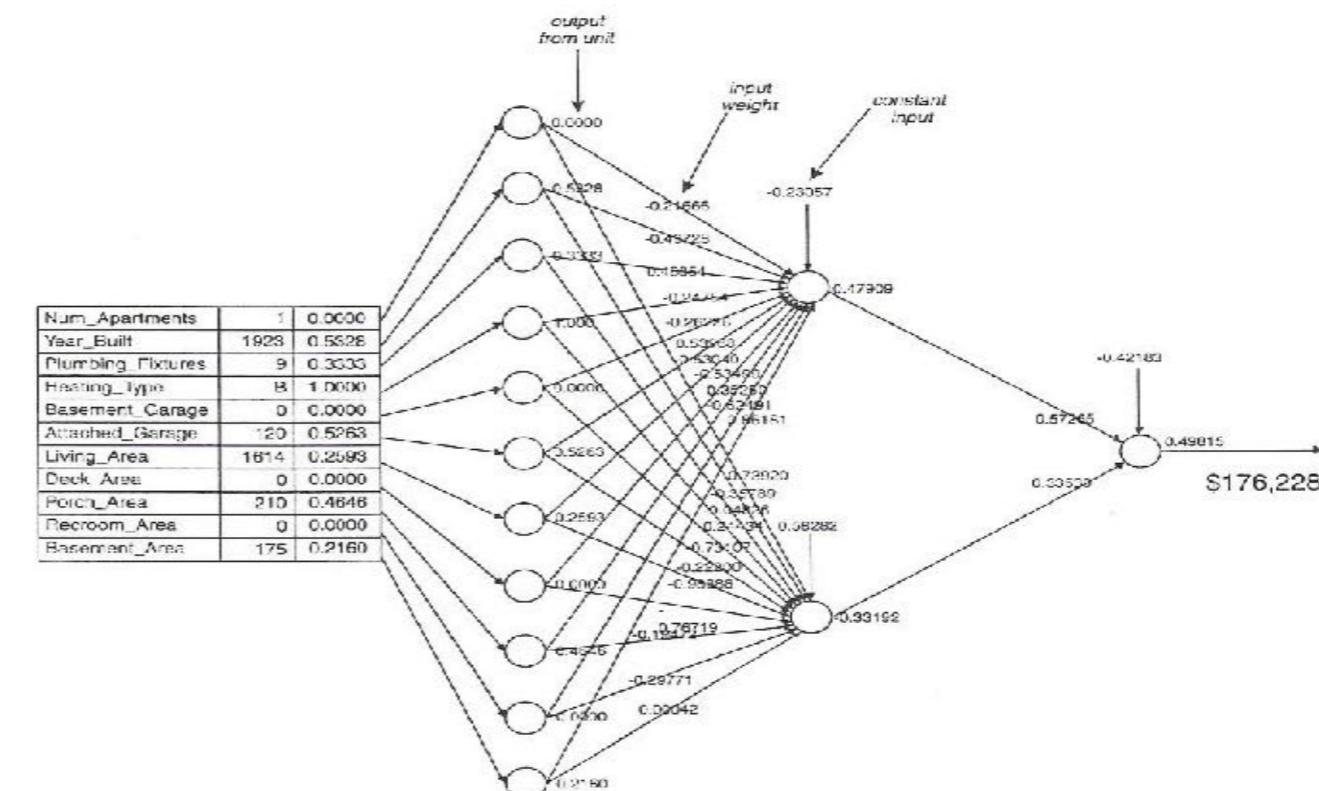
- **Combination function**
 - The function that merges all the input into a single value.
 - The most common combination function is the weighted sum.
- **Transfer function**
 - The function that transfers the value of the combination function to the output of the unit.

Feed-Forward Neural Networks (1)



Feed-Forward Neural Networks: Input Layer

- **Input layer**
 - Each unit in the input layer is connected to exactly one source field.



Feed-Forward Neural Networks: Hidden layer (1)

- **Hidden layer**
 - Each unit in the hidden layer is typically connected to all the units in the input layer.
 - The units in the hidden layer calculate their output by multiplying the value of each input by its corresponding weight, adding them up, and applying the transfer function.

Feed-Forward Neural Networks: Hidden layer (2)

- **Hidden layer**
 - A neural network can have any number of hidden layers, but in general **one** hidden layer is sufficient.
 - The wider the layer (the more units it contains), the greater the capacity of the network to recognize patterns.

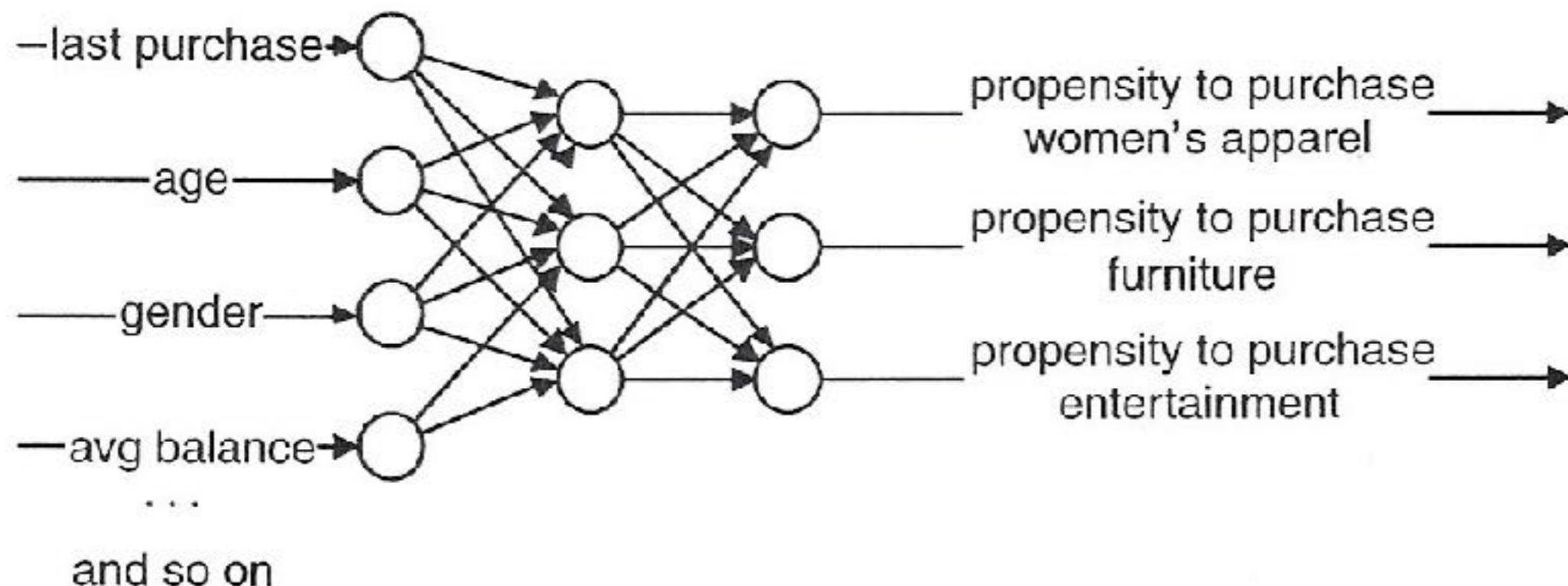
Feed-Forward Neural Networks : Output Layer (1)

- **Output layer**
 - It is fully connected to all the units in the hidden layer.
 - Most of the time, the neural network is being used to calculate a single value, so there is only one unit in the output layer and the value.
 - **We must map this value back to understand the output.**

Feed-Forward Neural Networks: Output Layer (2)

- **Output layer**
 - It is possible for the output layer to have more than one unit.
 - Eg. A department store chain wants to predict the likelihood that customers will be purchasing products from various departments, such as women's apparel, furniture, and entertainments.

Feed-Forward Neural Networks: Output Layer (3)



- After feeding the inputs for a customer into the network, the network calculates three values.

Back-Propagation Algorithm

- Assume the network is a fixed structure that corresponds to a directed graph.
- Learning corresponds to choosing a weight value for each edge in the graph.
- It attempts to minimize the squared error between the network output values and the target values for these outputs.
- It is the ANN learning technique that is most commonly used.

Back Propagation Neural Networks: Steps (1)

1. The network gets a training example and, using the existing weights in the network, it calculates the output.
2. Back propagation calculates the error by taking the difference between the calculated result and the expected (actual result).

Back Propagation Neural Networks: Steps (2)

3. The error is fed back through the network and the weights are adjusted to minimize the error.
4. After being shown enough training examples, the weights on the network no longer change significantly and the error no longer decreases. This is the point where training stops.

Back-Propagation Algorithm: Input

- Training example is a pair of the form $\langle x, t \rangle$, where x is the vector of network input values and t is the vector of target network output values
 - n is the learning rate
 - n_{in} is the number of network inputs
 - n_{hidden} is the number of units in the hidden layer
 - n_{out} is the number of output units
- The input from unit i into unit j is denoted x_{ji}
- The weight from unit i to unit j is denoted w_{ji}

Back-Propagation Algorithm: process (1)

- Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units
- Initialize all network weight to small random numbers
- Until the termination condition is met, Do repeat the steps

Back-Propagation Algorithm: process (2)

Propagate the input forward through the network:

1. Input the instance x to the network and compute the output o_u of every unit u in the network

Propagate the errors backward through the network:

2. For each network output unit k , calculate its error term e_k

$$e_k \leftarrow o_k (1 - o_k) (t_k - o_k)$$

Back-Propagation Algorithm: process

3. For each hidden unit h , calculate its error term e_h

$$e_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} e_k$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = n e_j x_{ji}$$

Termination conditions

- Stop after a fixed number of iterations
- Stop when the error on the training examples falls below some threshold
- Stop when the error on a separate validation set of examples meets some criterion

Hidden Layer Representations

- Training examples constrain the network inputs and outputs.
- The weight tuning procedure is free to set weights that define whatever hidden unit representation is most effective in minimizing the squared error.
- The ability of multilayer networks to automatically discover useful representation **at the hidden layers** is a key feature of ANN learning.

Heuristics for Using Feed-Forward & Back Propagation Networks

- One important decision is the **number of units in the hidden layer**. The more unit, the more patterns the network can recognize.
- We generally do not use hidden layers larger than the number of inputs.
- A good place to start for many problems is to experiment with one, two, and three nodes in the hidden layer.

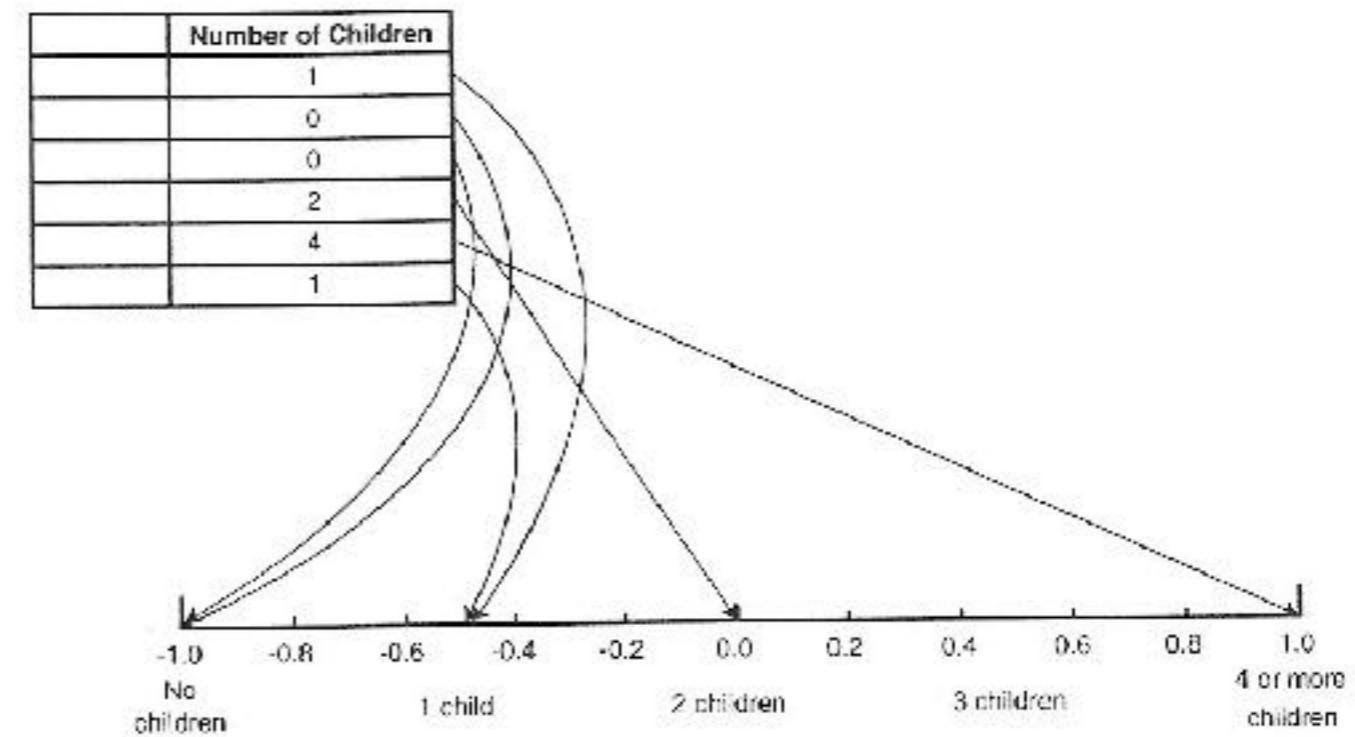
Preparing the Data: Features with Continuous Values

- The value should be scaled to be in a reasonable range.
- Plan for a larger range
- Reject out-of-range values
- Peg values lower than the minimum to the minimum and higher than the maximum to the maximum
- Map the minimum value to -0.9 and the maximum value to 0.9 instead of -1 and 1

Preparing the Data: Features with Ordered Discrete Values (1)

- First, count the number of different values and assign each a proportional fraction into some range.
- eg. If there are five distinct values, these get mapped to 0, 0.25, 0.50, 0.75, and 1.0

$$\begin{array}{rcl} 0 & \rightarrow & 0\ 0\ 0\ 0 = 0/16 = 0.0000 \\ 1 & \rightarrow & 1\ 0\ 0\ 0 = 8/16 = 0.5000 \\ 2 & \rightarrow & 1\ 1\ 0\ 0 = 12/16 = 0.7500 \\ 3 & \rightarrow & 1\ 1\ 1\ 0 = 14/16 = 0.8750 \end{array}$$



Preparing the Data: Features with Ordered Discrete Values (2)

- It is also possible to break a range into unequal parts.
- One example is called thermometer codes

$$\textcircled{m} \ 0 \rightarrow 0\ 0\ 0 = 0 / 16 = 0.0000$$

$$\textcircled{m} \ 1 \rightarrow 1\ 0\ 0 = 8 / 16 = 0.5000$$

$$\textcircled{m} \ 2 \rightarrow 1\ 1\ 0 = 12 / 16 = 0.7500$$

$$\textcircled{m} \ 3 \rightarrow 1\ 1\ 1 = 14 / 16 = 0.8750$$

It shows that the difference on one end of the scale is less significant than differences on the other end.

Preparing the Data: Features with Categorical Values (1)

- When working with categorical variables in neural networks, the mapping of variables into numbers **introduces an ordering** of the variables, which the neural network takes into account.
- The second way of handling categorical features is to break the categories into flags, one for each value.

Preparing the Data: Features with Categorical Values (2)

Gender	N coding			N-1 coding	
	Gender Male Flag	Gender Female Flag	Gender Unknown Flag	Gender Male Flag	Gender Female Flag
Male	+1.0	-1.0	-1.0	+1.0	-1.0
Female	-1.0	+1.0	-1.0	-1.0	+1.0
Unknown	-1.0	-1.0	+1.0	-1.0	-1.0

Interpreting the Results (1)

- When estimating a continuous value, often the output needs to be scaled back to the correct range.
- For binary or categorical output variables, the approach is still to take the inverse of the translation used for training the network.
- eg. If “churn” is given the value of 1 and “no churn” a value of -1 , the values near 1 represent churn and those near -1 represent no churn.

Interpreting the Results (2)

- When there are two outcomes, the meaning of the output depends on the training set used to train the network.
- The average value produced by the network during training is usually going to be close to the average value in the training set.

Interpreting the Results (3)

- One way to handle is, eg.
- If the training set had 50% churn and 50% no churn, the average value the network will produce from the training examples is going to be close to 0.0.
- Values higher than 0.0 are more like churn and those less than 0.0, less like churn.
- If the original training set had 10% churn, then the cutoff would more reasonable be –
- 0.8 rather than 0.0 (0.8 is 10% of the way from -1 to 1).

Interpreting the Results (4)

- For binary values, it is also possible to create a network that produces two output, one for each value.
- In this case, each output represents the strength of evidence that that category is the correct one. The chosen category would then be the one with the higher value.

How to Know What is Going on Inside a Neural Network

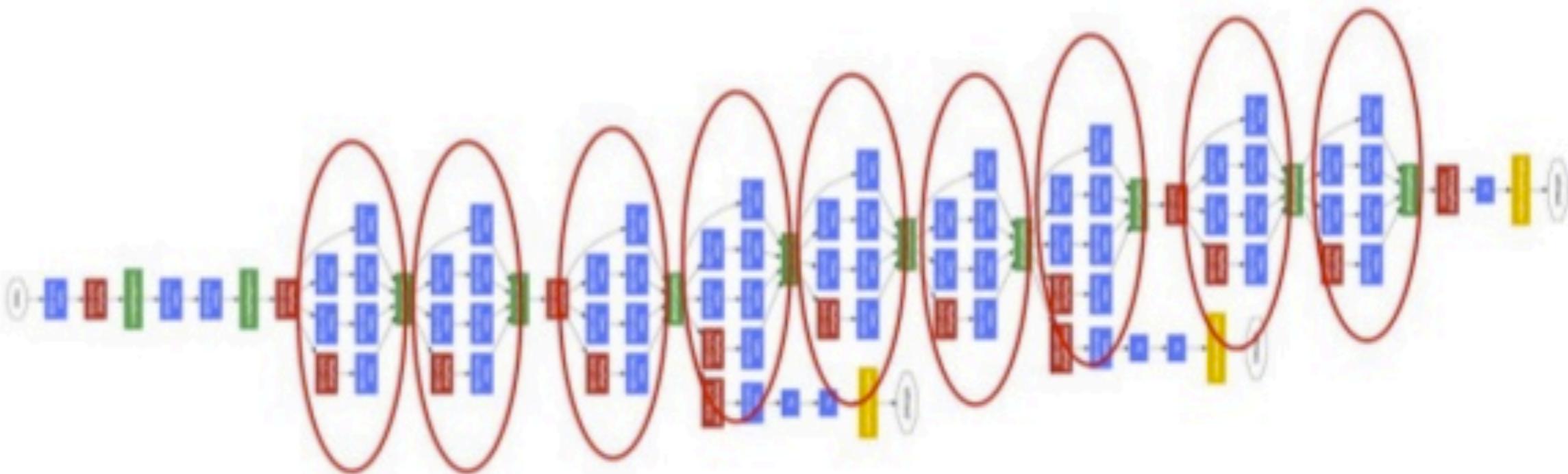
Sensitivity analysis uses the test set to determine how sensitive the output of the network is to each input :

1. Find the average value for each input.
2. Measure the output of the network when all inputs are at their average value.
3. Measure the output of the network when each input is modified, one at a time, to be at **its minimum and maximum values**.

Summary

- Neural networks can be used for both categorical and continuous inputs.
- Neural networks learn best when input fields have been mapped to the range between -1 and +1.
- Neural networks work best when there are only a few variables.
- When training a network, there is no guarantee that the resulting set of weights is optimal.
- A neural network cannot explain what it is doing.

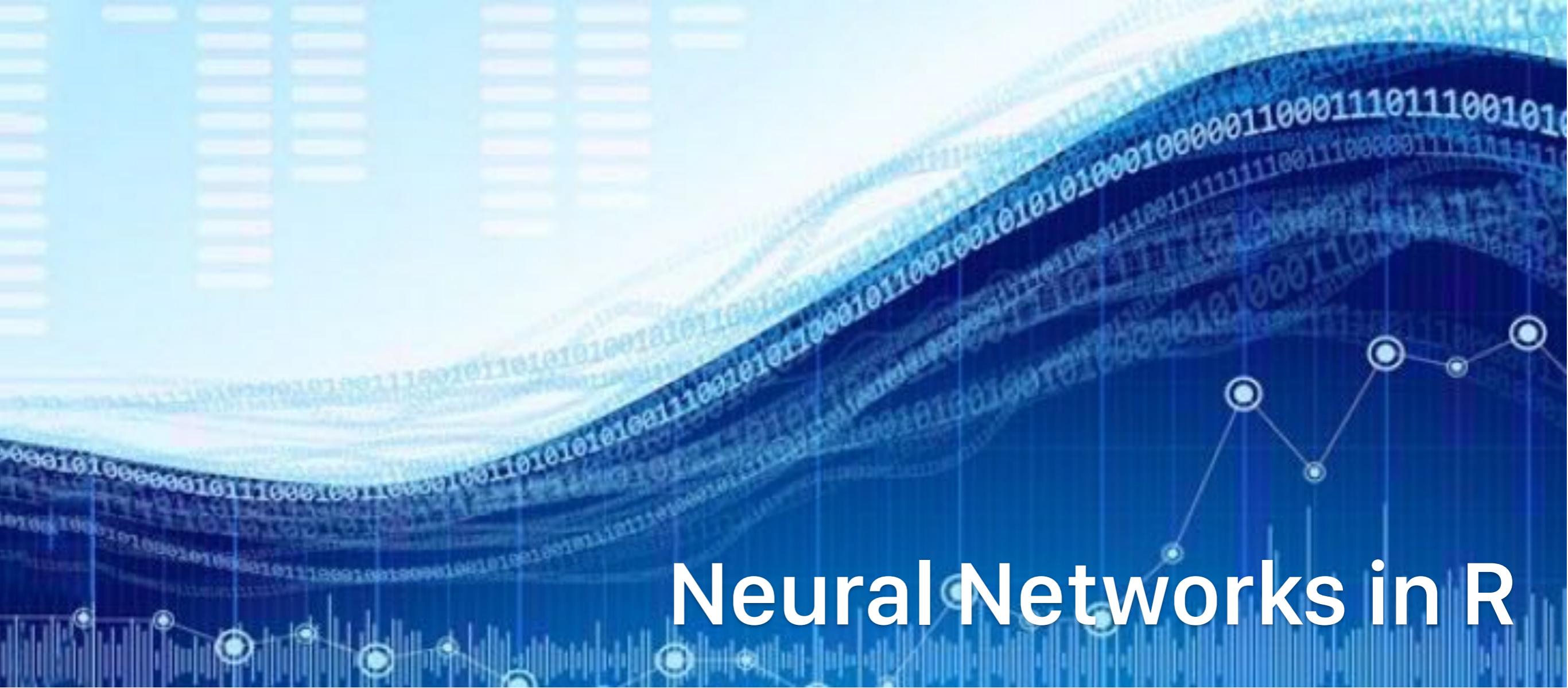
GoogLeNet (Inception)



9 Inception modules

Network in a network in a network...

Convolution
Pooling
Softmax
Other



Neural Networks in R

Fundamental Data Science for Data Scientist

Neural Network

Training of neural networks

Description

`neuralnet` is used to train neural networks using backpropagation, resilient backpropagation (RPROP) with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version (GRPROP) by Anastasiadis et al. (2005). The function allows flexible settings through custom-choice of error and activation function. Furthermore the calculation of generalized weights (Intrator O. and Intrator N., 1993) is implemented.

Usage

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,  
          stepmax = 1e+05, rep = 1, startweights = NULL,  
          learningrate.limit = NULL,  
          learningrate.factor = list(minus = 0.5, plus = 1.2),  
          learningrate=NULL, lifesign = "none",  
          lifesign.step = 1000, algorithm = "rprop+",  
          err.fct = "sse", act.fct = "logistic",  
          linear.output = TRUE, exclude = NULL,  
          constant.weights = NULL, likelihood = FALSE)
```

Neural Network

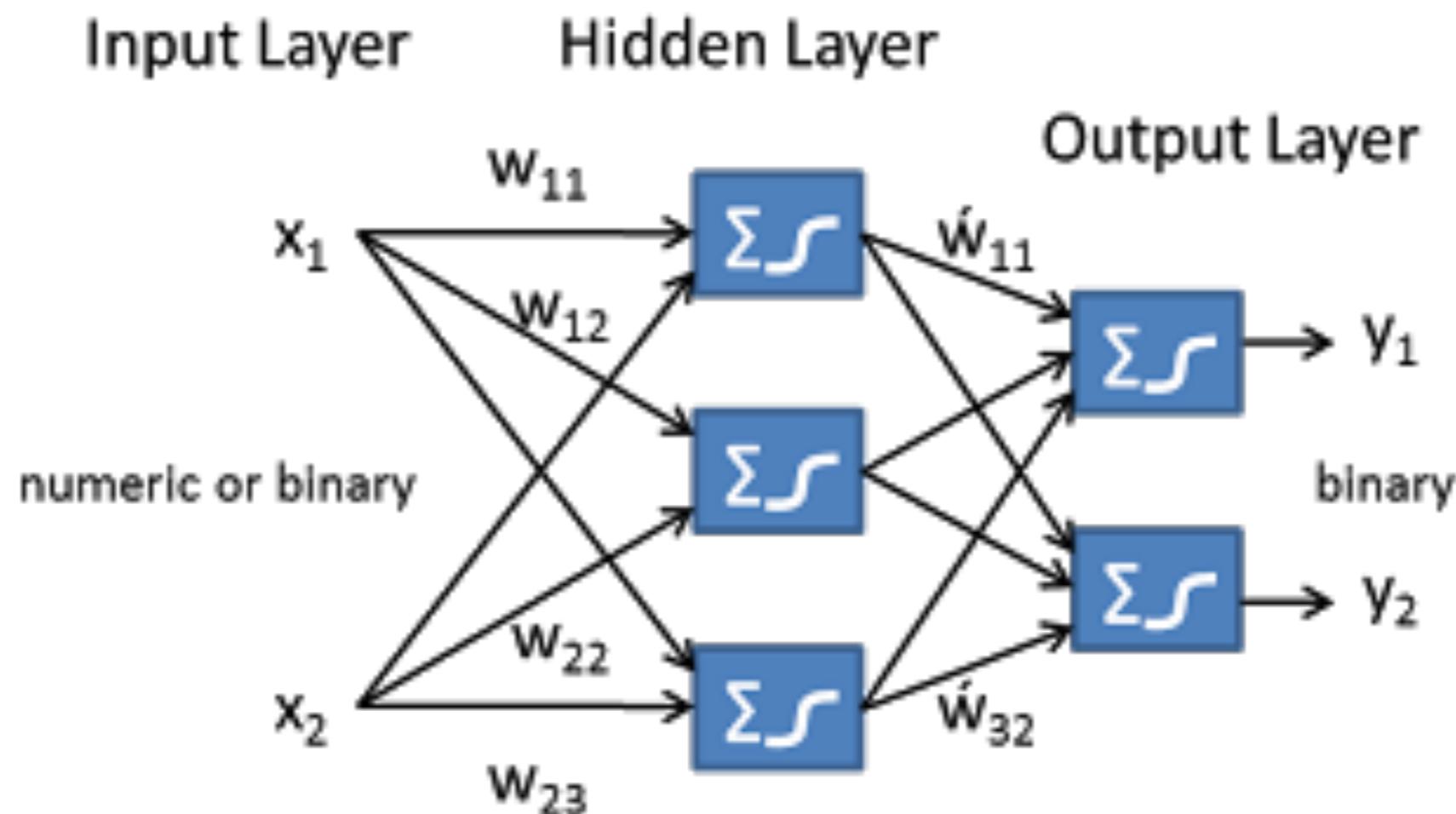
Arguments

<code>formula</code>	a symbolic description of the model to be fitted.
<code>data</code>	a data frame containing the variables specified in <code>formula</code> .
<code>hidden</code>	a vector of integers specifying the number of hidden neurons (vertices) in each layer.
<code>threshold</code>	a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria.
<code>stepmax</code>	the maximum steps for the training of the neural network. Reaching this maximum leads to a stop of the neural network's training process.
<code>rep</code>	the number of repetitions for the neural network's training.
<code>startweights</code>	a vector containing starting values for the weights. The weights will not be randomly initialized.
<code>learningrate.limit</code>	a vector or a list containing the lowest and highest limit for the learning rate. Used only for RPROP and GRPROP.
<code>learningrate.factor</code>	a vector or a list containing the multiplication factors for the upper and lower learning rate. Used only for RPROP and GRPROP.
<code>learningrate</code>	a numeric value specifying the learning rate used by traditional backpropagation. Used only for traditional backpropagation.
<code>lifesign</code>	a string specifying how much the function will print during the calculation of the neural network. 'none', 'minimal' or 'full'.

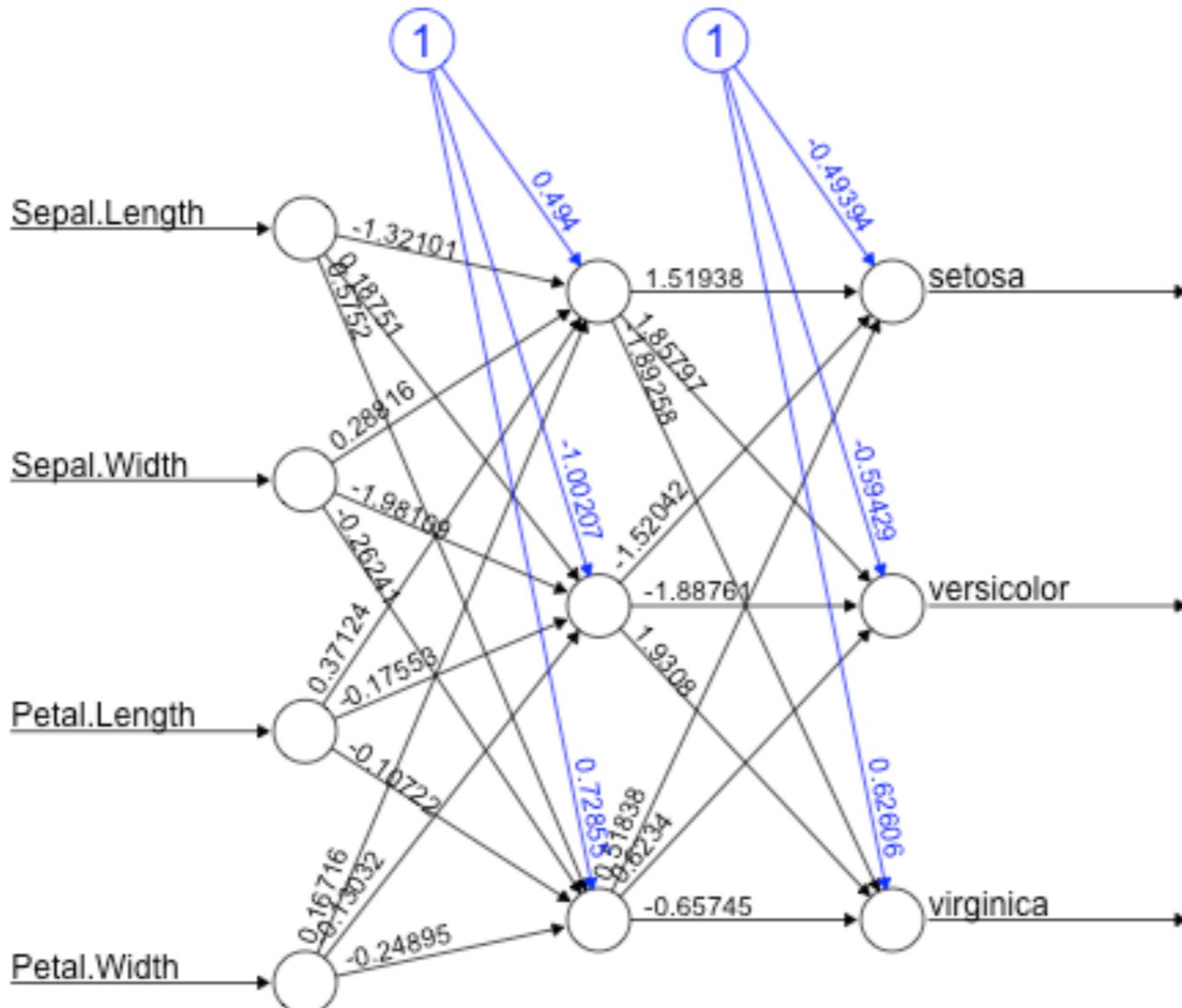
Neural Network

<code>lifesign.step</code>	an integer specifying the stepsize to print the minimal threshold in full lifesign mode.
<code>algorithm</code>	a string containing the algorithm type to calculate the neural network. The following types are possible: 'backprop', 'rprop+', 'rprop-', 'sag', or 'slr'. 'backprop' refers to backpropagation, 'rprop+' and 'rprop-' refer to the resilient backpropagation with and without weight backtracking, while 'sag' and 'slr' induce the usage of the modified globally convergent algorithm (grprop). See Details for more information.
<code>err.fct</code>	a differentiable function that is used for the calculation of the error. Alternatively, the strings 'sse' and 'ce' which stand for the sum of squared errors and the cross-entropy can be used.
<code>act.fct</code>	a differentiable function that is used for smoothing the result of the cross product of the covariate or neurons and the weights. Additionally the strings, 'logistic' and 'tanh' are possible for the logistic function and tangent hyperbolicus.
<code>linear.output</code>	logical. If <code>act.fct</code> should not be applied to the output neurons set <code>linear.output</code> to TRUE, otherwise to FALSE.
<code>exclude</code>	a vector or a matrix specifying the weights, that are excluded from the calculation. If given as a vector, the exact positions of the weights must be known. A matrix with n-rows and 3 columns will exclude n weights, where the first column stands for the layer, the second column for the input neuron and the third column for the output neuron of the weight.
<code>constant.weights</code>	a vector specifying the values of the weights that are excluded from the training process and treated as fix.
<code>likelihood</code>	logical. If the error function is equal to the negative log-likelihood function, the information criteria AIC and BIC will be calculated. Furthermore the usage of <code>confidence.interval</code> is meaningful.

Neural Network



```
2
3 library(neuralnet)
4 iris <- read.csv("iris.data.csv", header=TRUE)
5 # Prepare iris
6 set.seed(567)
7 ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
8 trainData <- iris[ind==1,]
9 testData <- iris[ind==2,]
10 nnet_iristrain <- trainData
11
12 #Binarize the categorical output
13 nnet_iristrain <- cbind(nnet_iristrain, trainData$Species == 'Iris-setosa')
14 nnet_iristrain <- cbind(nnet_iristrain, trainData$Species == 'Iris-versicolor')
15 nnet_iristrain <- cbind(nnet_iristrain, trainData$Species == 'Iris-virginica')
16 names(nnet_iristrain)[6] <- 'Setosa'
17 names(nnet_iristrain)[7] <- 'Versicolor'
18 names(nnet_iristrain)[8] <- 'Virginica'
19
20 nn <- neuralnet(Setosa+Versicolor+Virginica ~ Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,
21                   data=nnet_iristrain, hidden=c(3))
22
23 plot(nn)
24 mypredict <- compute(nn, testData[-5])$net.result
25 # Put multiple binary output to categorical output
26 maxidx <- function(arr) {
27   return(which(arr == max(arr)))
28 }
29 idx <- apply(mypredict, c(1), maxidx)
30 prediction <- c('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')[idx]
31 table(prediction, testData$Species)
32
33
```



Error: 0.001277 Steps: 281

```
> table(prediction, testData$Species)
```

prediction	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	15	0	0
Iris-versicolor	0	13	0
Iris-virginica	0	0	15

Accuracy

Confusion Matrix and Statistics

Prediction	Reference		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	15	0	0
Iris-versicolor	0	12	0
Iris-virginica	0	1	15

Overall Statistics

Accuracy : 0.9767442

95% CI : (0.8771095, 0.9994114)

No Information Rate : 0.3488372

P-Value [Acc > NIR] : < 0.0000000000000022204

Kappa : 0.9649837

McNemar's Test P-Value : NA



Statistics by Class:

	Class: Iris-setosa	Class: Iris-versicolor	Class: Iris-virginica
Sensitivity	1.0000000	0.9230769	1.0000000
Specificity	1.0000000	1.0000000	0.9642857
Pos Pred Value	1.0000000	1.0000000	0.9375000
Neg Pred Value	1.0000000	0.9677419	1.0000000
Prevalence	0.3488372	0.3023256	0.3488372
Detection Rate	0.3488372	0.2790698	0.3488372
Detection Prevalence	0.3488372	0.2790698	0.3720930
Balanced Accuracy	1.0000000	0.9615385	0.9821429

Workshop 15 - Neural Networks

1. From the data that you import
2. Build Neural Networks Model to predict target
3. Perform performance management by creating confusion matrix

Topic

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Naïve Bayesian classification
- K-nearest neighbor
- Support Vector Machine
- Neural Net
- **Ensemble methods: Bagging and Boosting**
- Summary

Combining classifiers

- So far, we have only discussed individual classifiers, i.e., how to build them and use them.
- Can we combine multiple classifiers to produce a better classifier?
- Yes, sometimes
- We discuss two main algorithms:
 - **Bagging**
 - **Boosting**

Bagging

- Breiman, 1996
- Bootstrap Aggregating = Bagging
 - Application of bootstrap sampling
 - Given: set D containing m training examples
 - Create a sample $S[i]$ of D by drawing m examples at random with replacement from D
 - $S[i]$ of size m: expected to leave out 0.37 of examples from D

Bagging (cont...)

- Training
 - Create k bootstrap samples $S[1], S[2], \dots, S[k]$
 - Build a distinct classifier on each $S[i]$ to produce k classifiers, using the same learning algorithm.
- Testing
 - Classify each new instance by voting of the k classifiers (equal weights)

Bagging Example

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

Bagging (cont ...)

When does it help?

When learner is unstable

Small change to training set causes large change in the output classifier

True for decision trees, neural networks; not true for k -nearest neighbor, naïve Bayesian, class association rules

Experimentally, bagging can help substantially for unstable learners, may somewhat degrade results for stable learners

Boosting

- A family of methods:
 - We only study **AdaBoost** (Freund & Schapire, 1996)
- Training
 - Produce a sequence of classifiers (the same base learner)
 - Each classifier is dependent on the previous one, and focuses on the previous one's errors
 - Examples that are incorrectly predicted in previous classifiers are given higher weights
- Testing
 - For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

AdaBoost

Weighted training set

(x_1, y_1, w_1)

(x_2, y_2, w_2)

...

(x_n, y_n, w_n)

Non-negative weights
sum to 1



called a weaker classifier

- Build a classifier h_t whose accuracy on training set $> \frac{1}{2}$ (better than random)

Change weights

AdaBoost algorithm

Algorithm AdaBoost.M1

Input: sequence of m examples $\langle(x_1, y_1), \dots, (x_m, y_m)\rangle$
with labels $y_i \in Y = \{1, \dots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i .

Do for $t = 1, 2, \dots, T$:

1. Call **WeakLearn**, providing it with the distribution D_t .
2. Get back a hypothesis $h_t : X \rightarrow Y$.

3. Calculate the error of h_t : $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$.

If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.

4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$h_{fin}(x) = \arg \max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}.$$

Does AdaBoost always work?

- The actual performance of boosting depends on the data and the base learner.
 - It requires the base learner to be unstable as bagging.
- Boosting seems to be susceptible to noise.
 - When the number of outliers is very large, the emphasis placed on the hard examples can hurt the performance.

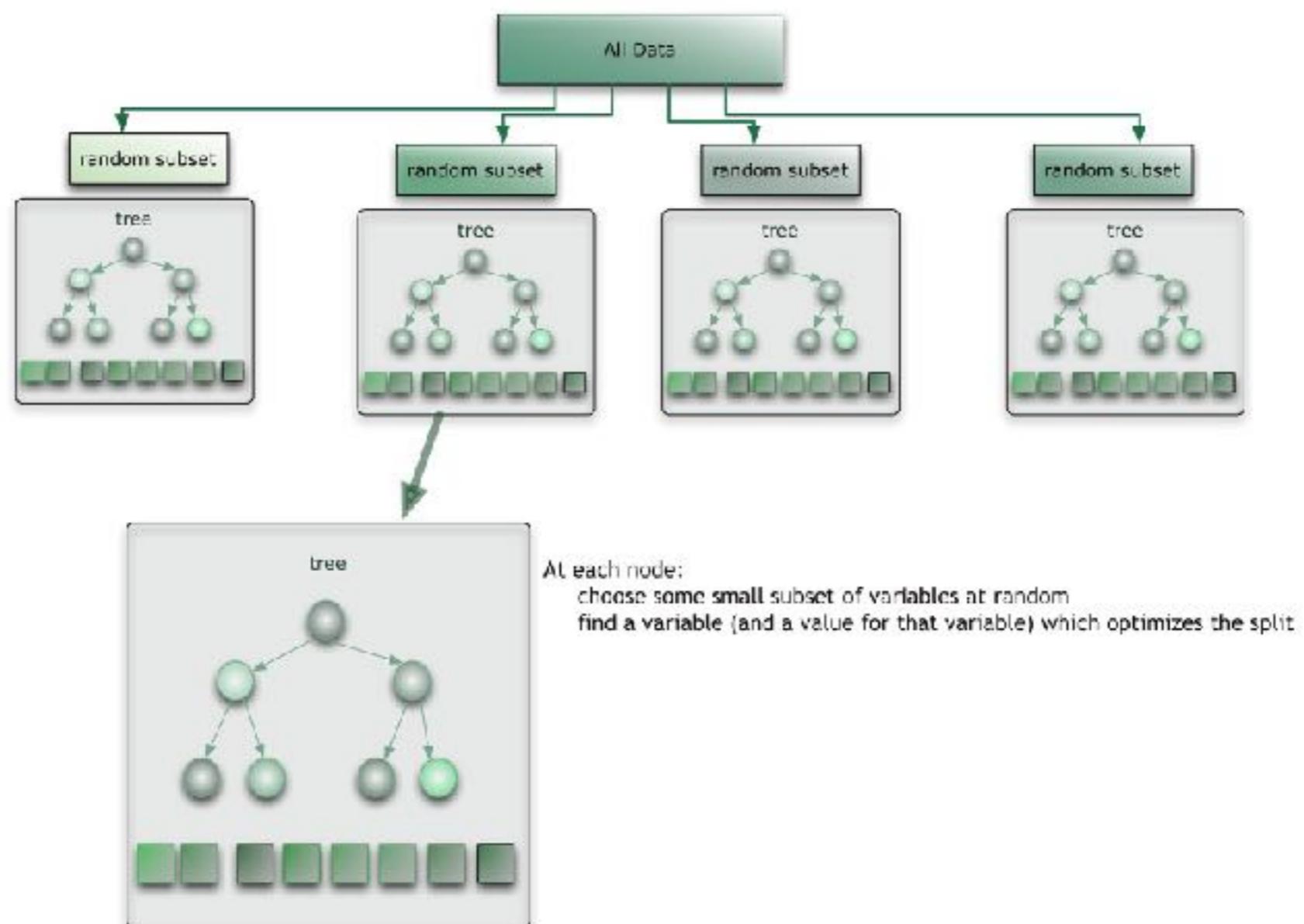


Ensemble in R

Fundamental Data Science for Data Scientist

Ensemble : Bagging

Random Forest



Random Forest

Here is how such a system is trained; for some number of trees T :

- 1) Sample N cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.
- 2) At each node:
 - a) For some number m (see below), m predictor variables are selected at random from all the predictor variables.
 - b) The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.
 - c) At the next node, choose another m variables at random from all predictor variables and do the same.

Bagging

```
> library(randomForest)
#Train 100 trees, random selected attributes
> model <- randomForest(Species~, data=iristrain, nTree=500)
#Predict using the forest
> prediction <- predict(model, newdata=iristest, type='class')
> table(prediction, iristest$Species)
> importance(model)
      MeanDecreaseGini
Sepal.Length      7.807602
Sepal.Width       1.677239
Petal.Length     31.145822
Petal.Width      38.617223
```

Confusion Matrix and Statistics

		Reference		
		Iris-setosa	Iris-versicolor	Iris-virginica
Prediction	Iris-setosa	15	0	0
	Iris-versicolor	0	13	2
		Iris-virginica	0	13

Overall Statistics

Accuracy : 0.9535

95% CI : (0.8419, 0.9943)

No Information Rate : 0.3488

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9303

McNemar's Test P-Value : NA

Statistics by Class:

	Class: Iris-setosa	Class: Iris-versicolor
Sensitivity	1.0000	1.0000
Specificity	1.0000	0.9333
Pos Pred Value	1.0000	0.8667
Neg Pred Value	1.0000	1.0000
Prevalence	0.3488	0.3023
Detection Rate	0.3488	0.3023
Detection Prevalence	0.3488	0.3488
Balanced Accuracy	1.0000	0.9667
	Class: Iris-virginica	
Sensitivity	0.8667	
Specificity	1.0000	
Pos Pred Value	1.0000	
Neg Pred Value	0.9333	
Prevalence	0.3488	
Detection Rate	0.3023	
Detection Prevalence	0.3023	
Balanced Accuracy	0.9333	

Boosting

```
> library(adabag)  
  
> iris.adaboost <- boosting(Species~, data=trainData, boost=TRUE, mfinal=5)  
  
> iris.adaboost$class  
  
> table(iris.adaboost$class, trainData$Species)  
  
> prediction <- predict(iris.adaboost, newdata=testData)  
  
> table(prediction$class, testData$Species)  
  
> confusionMatrix(prediction$class, testData$Species)
```

Confusion Matrix and Statistics

Prediction	Reference		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	15	0	0
Iris-versicolor	0	13	1
Iris-virginica	0	0	14

Overall Statistics

Accuracy : 0.9767

95% CI : (0.8771, 0.9994)

No Information Rate : 0.3488

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9651

McNemar's Test P-Value : NA

Statistics by Class:

Class: Iris-setosa Class: Iris-versicolor

Sensitivity	1.0000	1.0000
Specificity	1.0000	0.9667
Pos Pred Value	1.0000	0.9286
Neg Pred Value	1.0000	1.0000
Prevalence	0.3488	0.3023
Detection Rate	0.3488	0.3023
Detection Prevalence	0.3488	0.3256
Balanced Accuracy	1.0000	0.9833

Class: Iris-virginica

Sensitivity	0.9333
Specificity	1.0000
Pos Pred Value	1.0000
Neg Pred Value	0.9655
Prevalence	0.3488
Detection Rate	0.3256
Detection Prevalence	0.3256
Balanced Accuracy	0.9667

Workshop 16 - Ensemble Method

1. Train your model using either Bagging (Random Forest) or Boosting (AdaBoost)
2. Perform performance management by creating confusion matrix

Machine Learning Summary

- Applications of supervised learning are in almost any field or domain.
- We studied 8 classification techniques.
- There are still many other methods, e.g.,
 - Bayesian networks
 - Neural networks
 - Genetic algorithms
 - Fuzzy classification
 - This large number of methods also show the importance of classification and its wide applicability.
- It remains to be an active research area.



Unsupervised Learning

Supervised learning vs. unsupervised learning

Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.

These patterns are then utilized to predict the values of the target attribute in future data instances.

Unsupervised learning: The data have no target attribute.

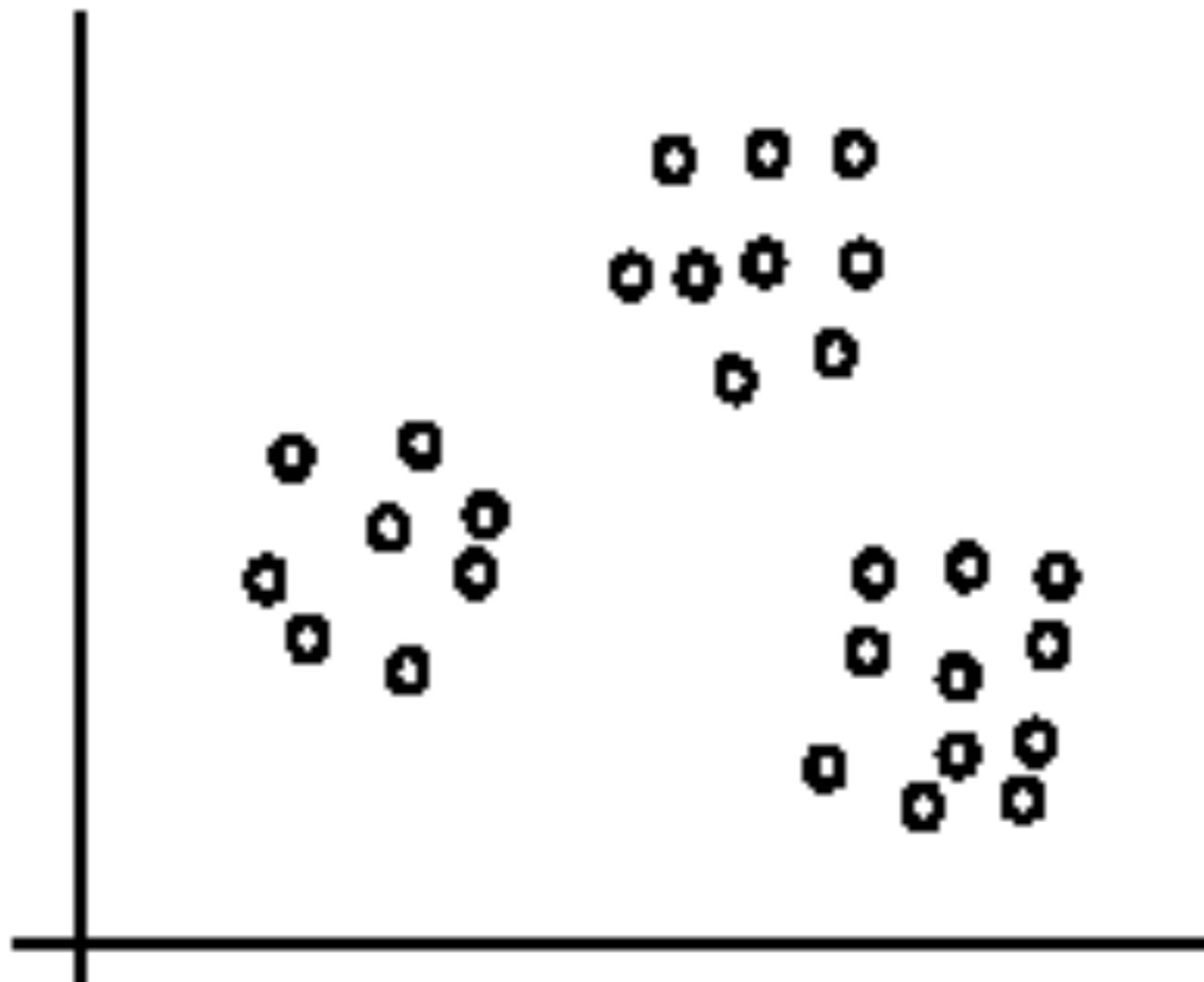
We want to explore the data to find some intrinsic structures in them.

Clustering

- Clustering is a technique for finding **similarity groups** in data, called **clusters**. i.e.,
 - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
 - In fact, association rule mining is also unsupervised
- This chapter focuses on clustering.

An illustration

The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
 - Tailor-made for each person: too expensive
 - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
 - To do targeted marketing.

What is clustering for? (cont...)

Example 3: Given a collection of text documents, we want to organize them according to their content similarities,

To produce a topic hierarchy

In fact, clustering is one of the most utilized data mining techniques.

It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.

In recent years, due to the rapid increase of online documents, text clustering becomes important.

Aspects of clustering

- A clustering algorithm
 - Partitional clustering
 - Hierarchical clustering
 - ...
- A distance (similarity, or dissimilarity) function
- Clustering quality
 - Inter-clusters distance \Rightarrow maximized
 - Intra-clusters distance \Rightarrow minimized
- The **quality** of a clustering result depends on the algorithm, the

K-means clustering

K-means is a **partitional clustering** algorithm

Let the set of data points (or instances) D be

$$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\},$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ir})$ is a **vector** in a real-valued space $X \subseteq \mathbb{R}^r$, and r

is the number of attributes (dimensions) in the data.

The k-means algorithm partitions the given data into k clusters.

Each cluster has a cluster **center**, called **centroid**.

k is specified by the user

K-means algorithm

Given k , the k-means algorithm works as follows:

- 1) Randomly choose k data points (**seeds**) to be the initial **centroids**, cluster centers
- 2) Assign each data point to the closest **centroid**
- 3) Re-compute the **centroids** using the current cluster memberships.
- 4) If a convergence criterion is not met, go to 2).

K-means algorithm – (cont ...)

```
Algorithm k-means( $k$ ,  $D$ )
1   Choose  $k$  data points as the initial centroids (cluster centers)
2   repeat
3       for each data point  $\mathbf{x} \in D$  do
4           compute the distance from  $\mathbf{x}$  to each centroid;
5           assign  $\mathbf{x}$  to the closest centroid      // a centroid represents a cluster
6       endfor
7       re-compute the centroids using the current cluster memberships
8   until the stopping criterion is met
```

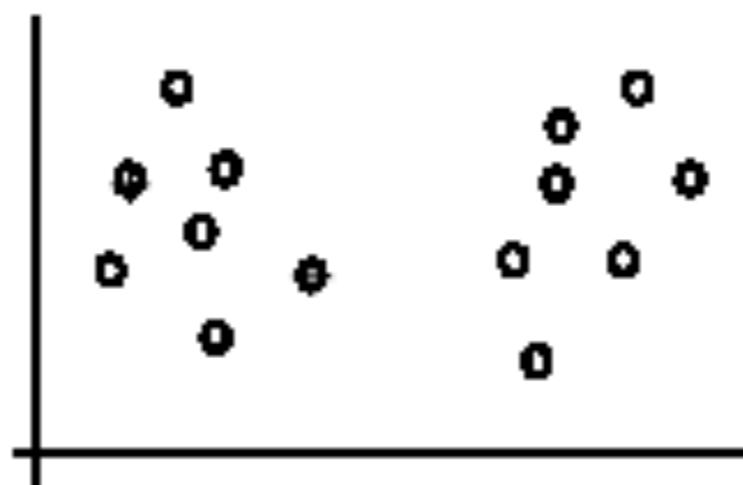
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

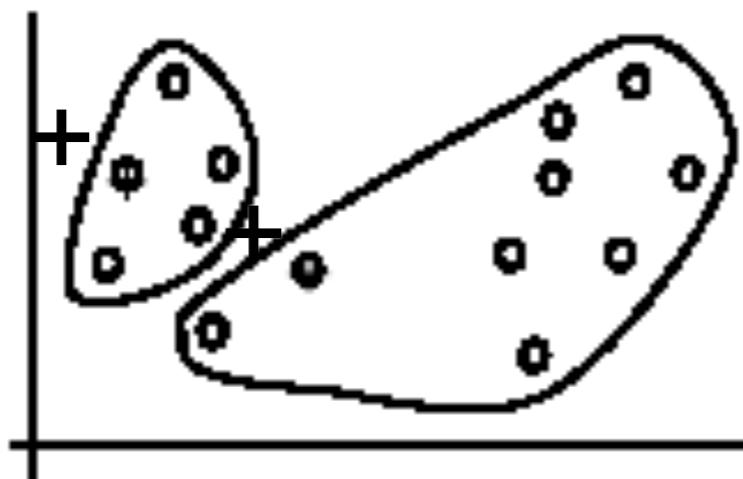
C_i is the j th cluster, \mathbf{m}_j is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point \mathbf{x} and centroid \mathbf{m}_j .

$$SSE = \sum_{j=1}^k \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \quad (1)$$

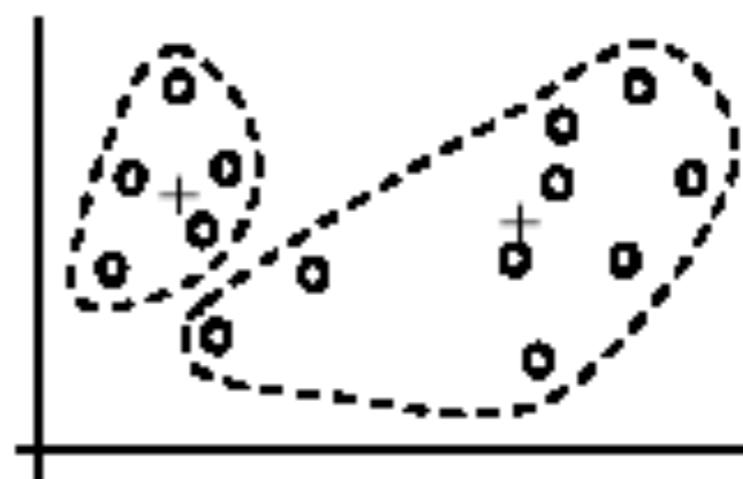
An example



(A). Random selection of k centers

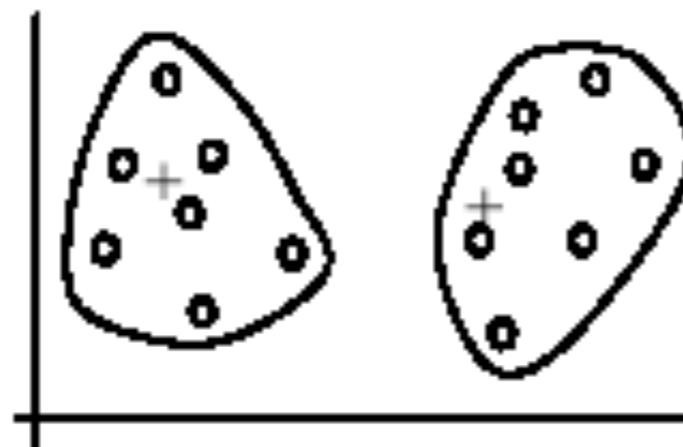


Iteration 1: (B). Cluster assignment

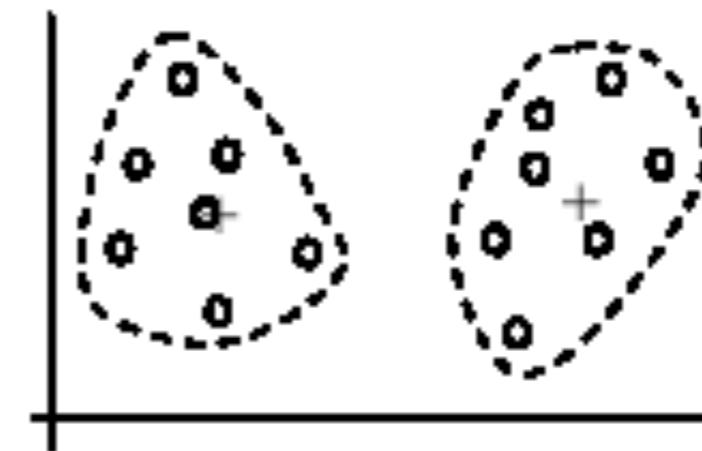


(C). Re-compute centroids

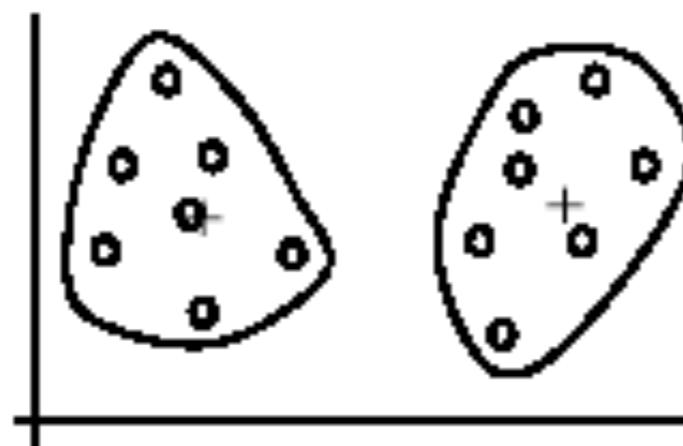
An example (cont ...)



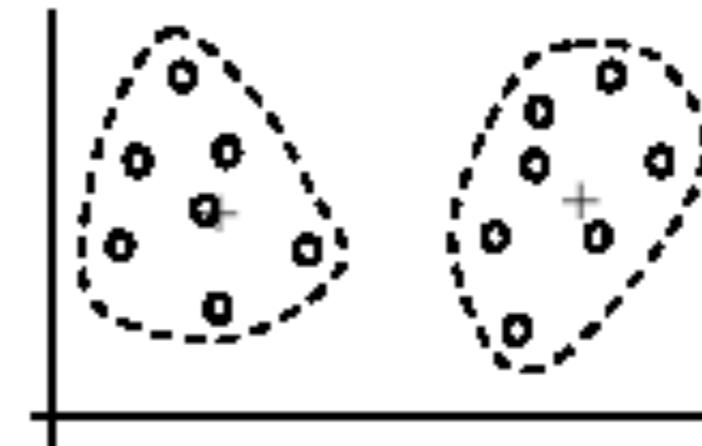
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

An example distance function

The k -means algorithm can be used for any application data set where the **mean** can be defined and computed. In the **Euclidean space**, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \quad (2)$$

where $|C_j|$ is the number of data points in cluster C_j . The distance from one data point \mathbf{x}_i to a mean (centroid) \mathbf{m}_j is computed with

$$\begin{aligned} dist(\mathbf{x}_i, \mathbf{m}_j) &= \| \mathbf{x}_i - \mathbf{m}_j \| \\ &= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \dots + (x_{ir} - m_{jr})^2} \end{aligned} \quad (3)$$

A disk version of k -means

- K-means can be implemented with data on disk
 - In each iteration, it scans the data once.
 - as the centroids can be computed incrementally
- It can be used to cluster large datasets that do not fit in main memory
- We need to control the number of iterations
 - In practice, a limit is set (< 50).
- Not the best method. There are other scale-up algorithms, e.g., BIRCH (balanced iterative reducing and clustering using hierarchies).

A disk version of k-means (cont...)

Algorithm disk- k -means(k, D)

- 1 Choose k data points as the initial centroids $\mathbf{m}_j, j = 1, \dots, k,$
- 2 **repeat**
- 3 initialize $\mathbf{s}_j = \mathbf{0}, j = 1, \dots, k,$ // $\mathbf{0}$ is a vector with all 0's
- 4 initialize $n_j = 0, j = 1, \dots, k;$ // n_j is the number points in cluster j
- 5 **for** each data point $\mathbf{x} \in D$ **do**
- 6 $j = \arg \min_j \text{dist}(\mathbf{x}, \mathbf{m}_j);$
- 7 assign \mathbf{x} to the cluster $j;$
- 8 $\mathbf{s}_j = \mathbf{s}_j + \mathbf{x};$
- 9 $n_j = n_j + 1;$
- 10 **endfor**
- 11 $\mathbf{m}_i = \mathbf{s}_j / n_j, i = 1, \dots, k,$
- 12 **until** the stopping criterion is met

Strengths of k-means

Strengths:

Simple: easy to understand and to implement

Efficient: Time complexity: $O(tkn)$,

where n is the number of data points,

k is the number of clusters, and

t is the number of iterations.

Since both k and t are small, k-means is considered a linear algorithm.

K-means is the most popular clustering algorithm.

Note that: it terminates at a **local optimum** if SSE is used. The **global optimum** is hard to find due to complexity.

Weaknesses of k-means

The algorithm is only applicable if the **mean** is defined.

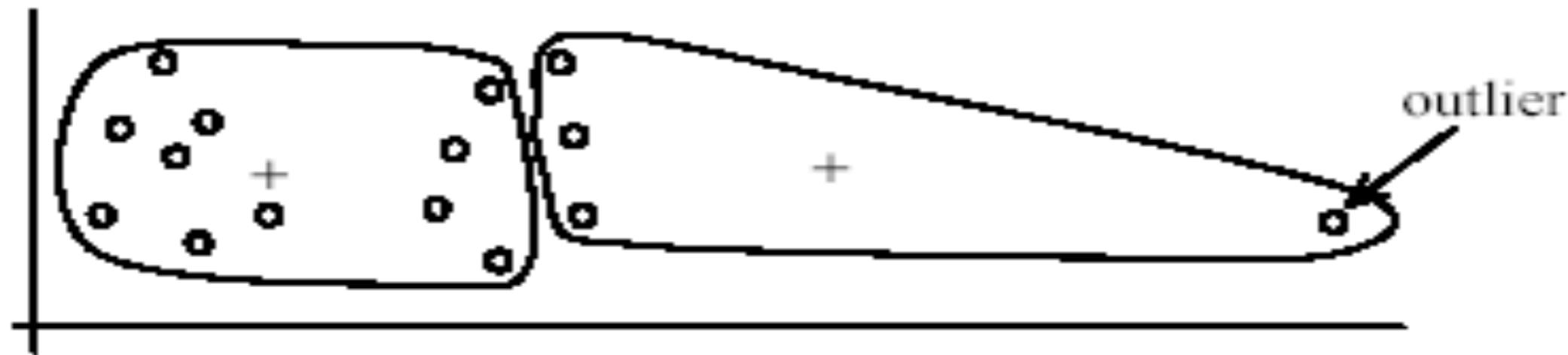
For categorical data, **k-mode** - the centroid is represented by most frequent values.

The user needs to specify **k**.

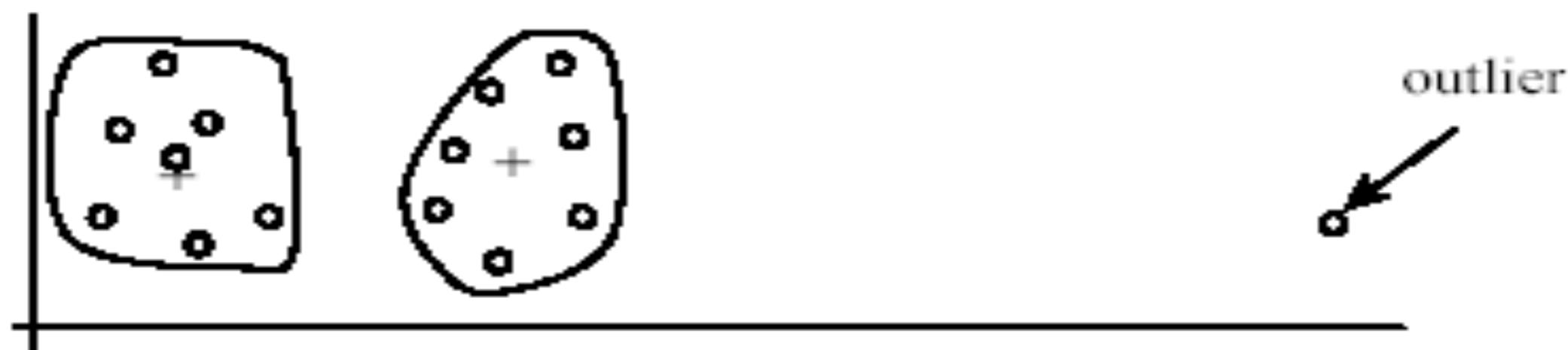
The algorithm is sensitive to **outliers**

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

One method is to **remove some data points** in the clustering process that are much further away from the centroids than other data points.

To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

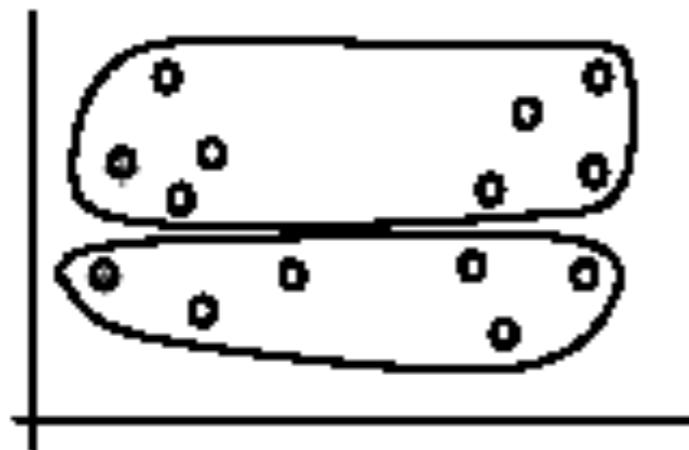
Another method is to **perform random sampling**. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.

- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

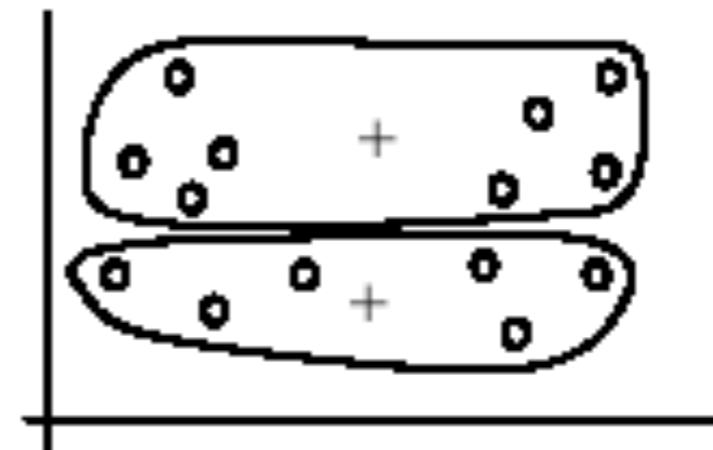
Weaknesses of k-means (cont ...)



(A). Random selection of seeds (centroids)

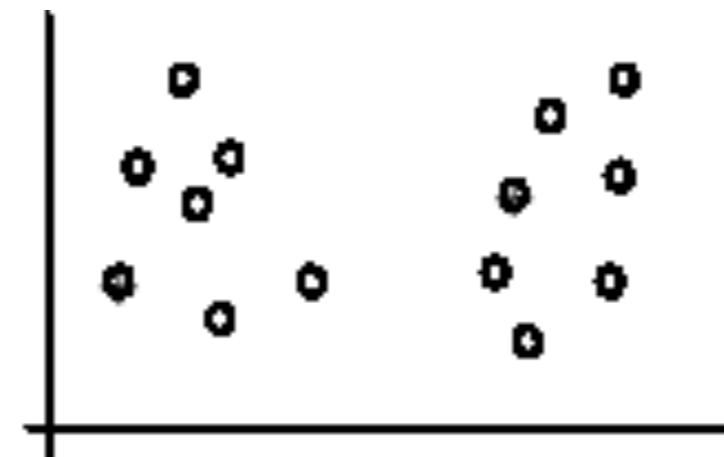


(B). Iteration 1



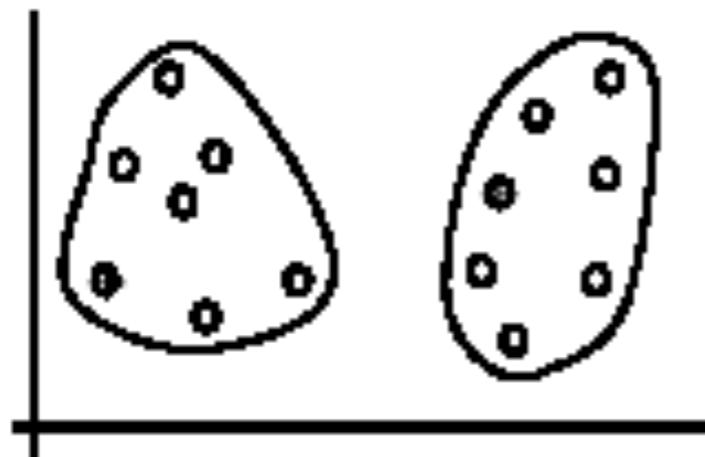
(C). Iteration 2

Weaknesses of k-means (cont ...)

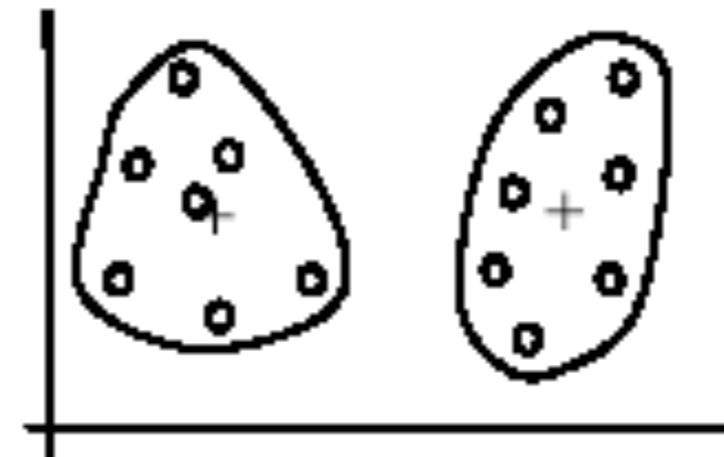


- There are some methods to help choose good seeds

(A). Random selection of k seeds (centroids)



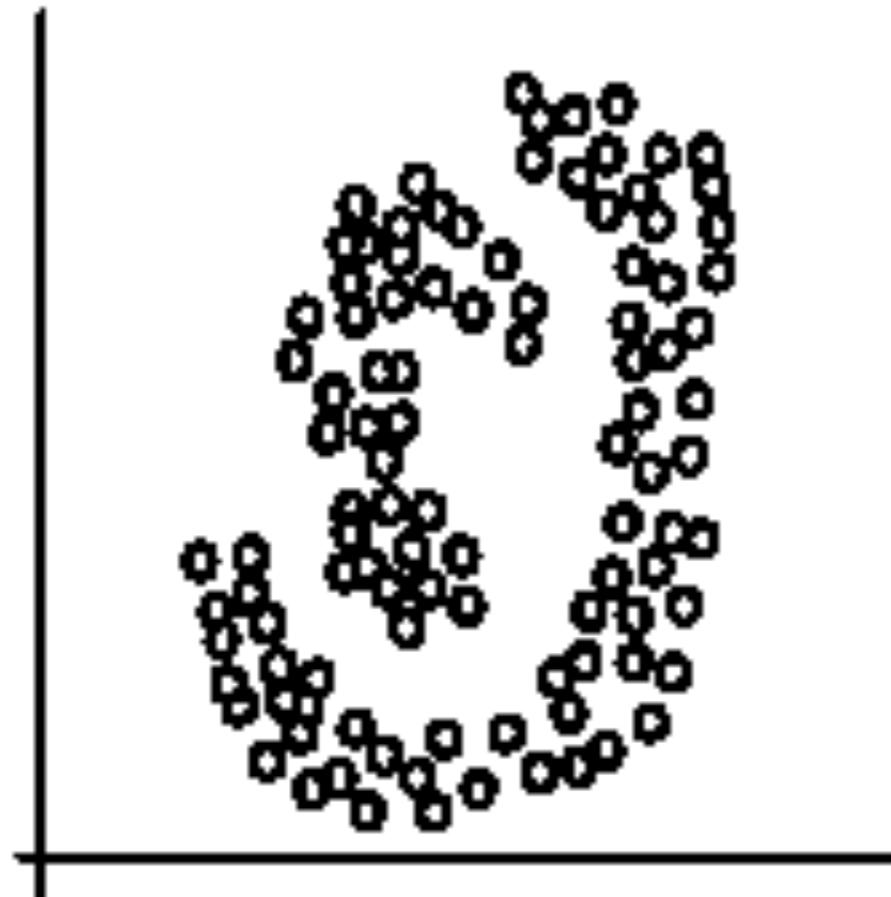
(B). Iteration 1



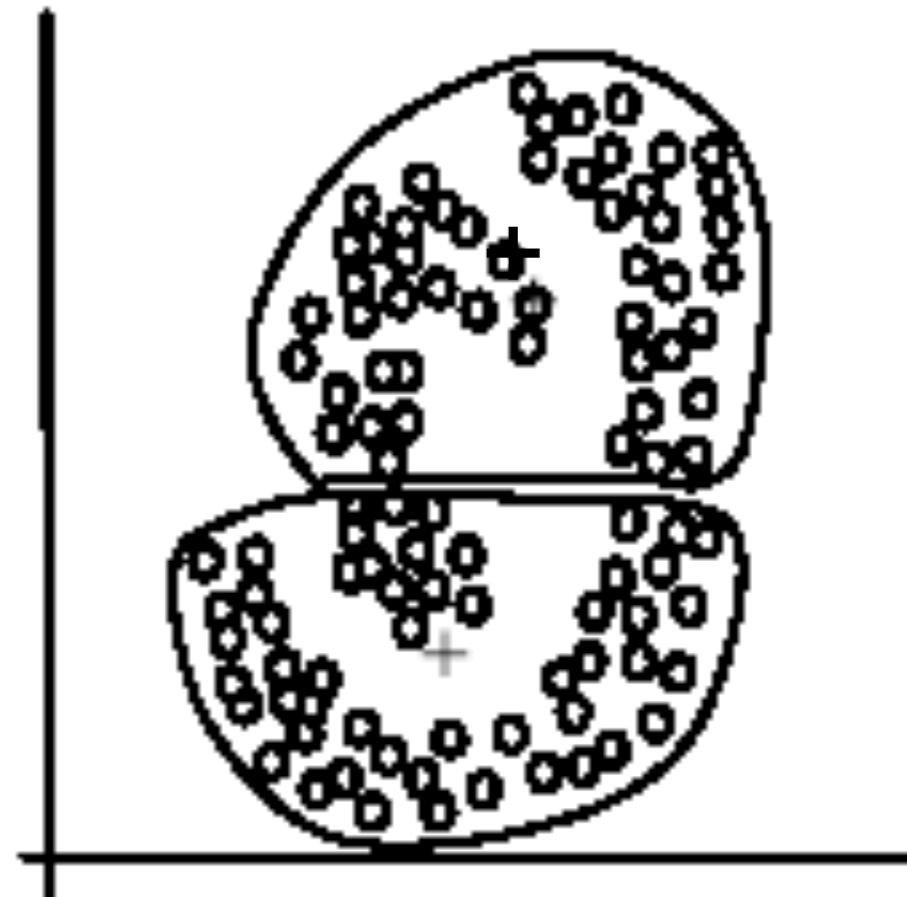
(C). Iteration 2

Weaknesses of k-means (cont ...)

The k -means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): k -means clusters

K-means summary

- Despite weaknesses, k-means is still the most popular algorithm due to its simplicity, efficiency and
 - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
 - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Common ways to represent clusters

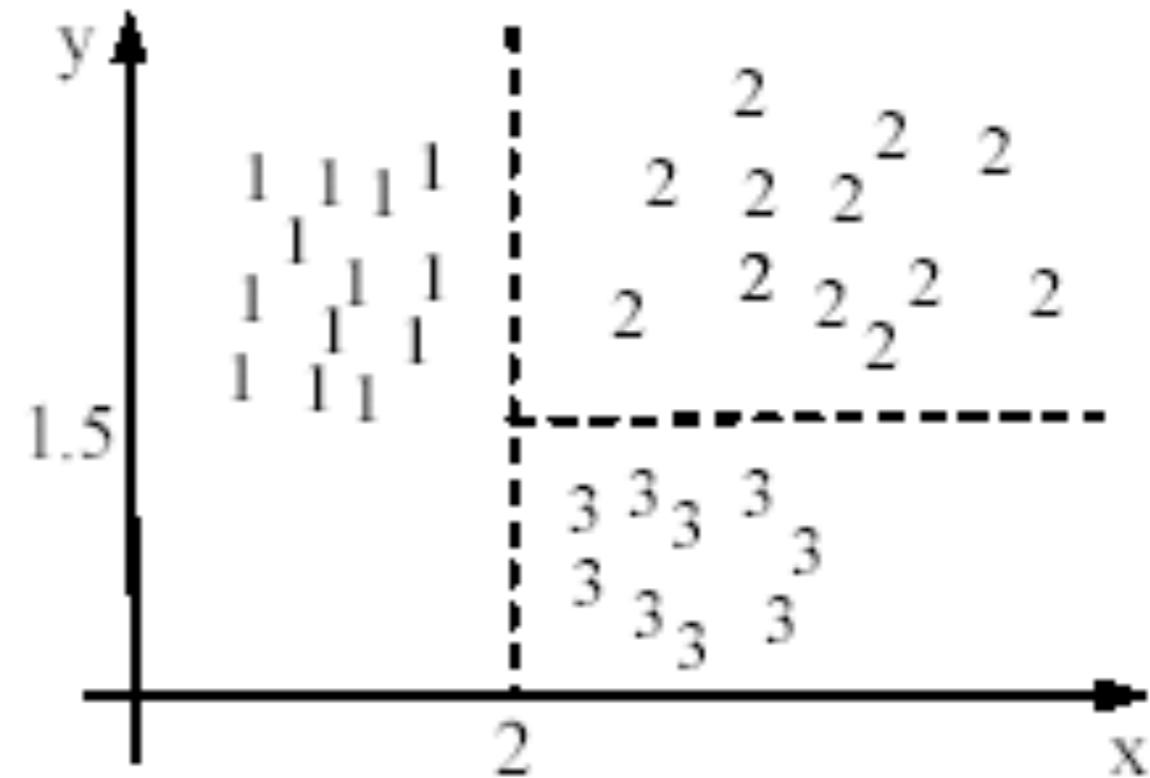
Use the centroid of each cluster to represent the cluster.

- compute the radius and
- standard deviation of the cluster to determine its spread in each dimension
- The centroid representation alone works well if the clusters are of the hyper-spherical shape.
- If clusters are elongated or are of other shapes, centroids are not sufficient

Using classification model

All the data points in a cluster are regarded to have the same class label, e.g., the cluster ID.

run a supervised learning algorithm on the data to find a classification model.



$x \leq 2 \rightarrow$ cluster 1

$x > 2, y > 1.5 \rightarrow$ cluster 2

$x > 2, y \leq 1.5 \rightarrow$ cluster 3

Use frequent values to represent cluster

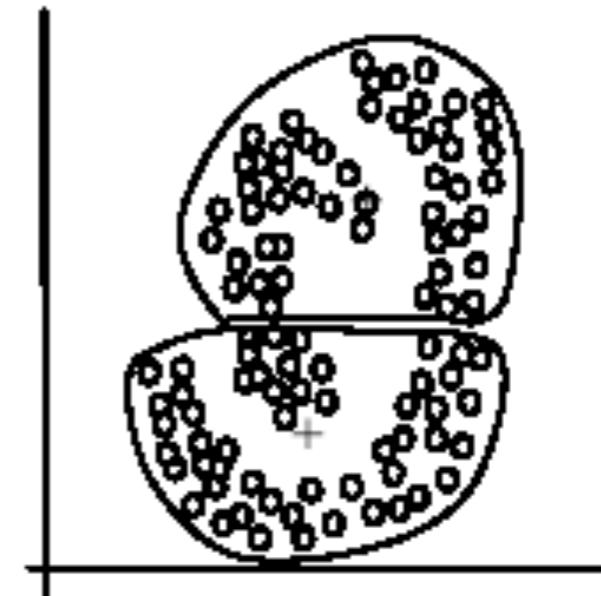
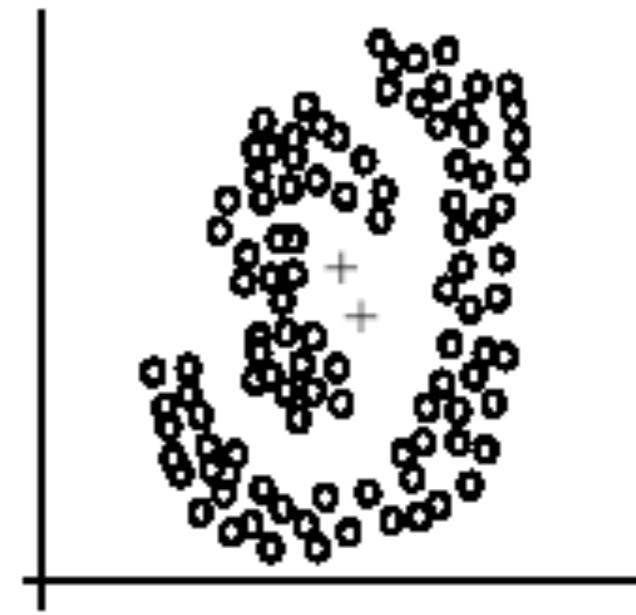
- This method is mainly for clustering of categorical data (e.g., k -modes clustering).
- Main method used in text clustering, where a small set of frequent words in each cluster is selected to represent the cluster.

Clusters of arbitrary shapes

Hyper-elliptical and hyper-spherical clusters are usually easy to represent, using their centroid together with spreads.

Irregular shape clusters are hard to represent. They may not be useful in some applications.

- Using centroids are not suitable (upper figure) in general
- K-means clusters may be more useful (lower figure), e.g., for making 2 size T-shirts.





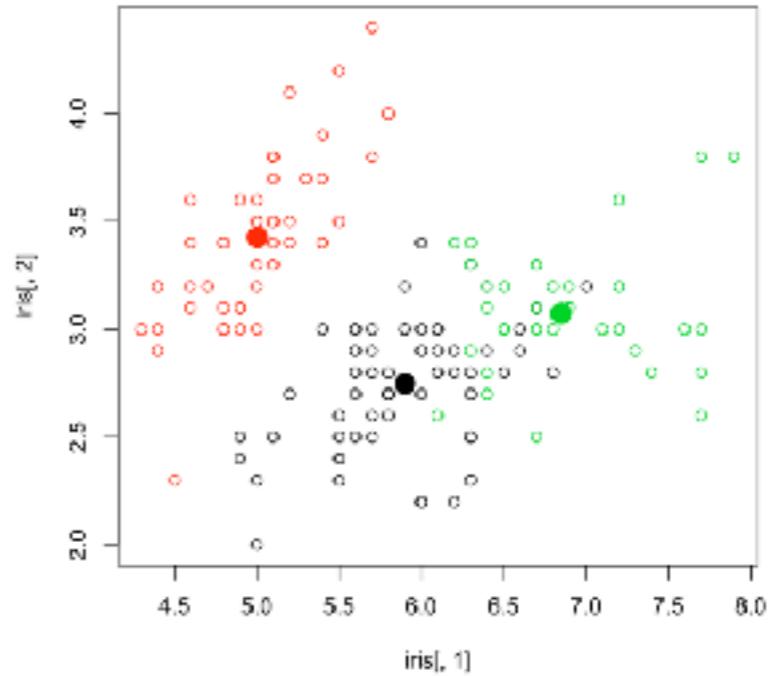
k-Mean in R

Fundamental Data Science for Data Scientist

K-Means Clustering

1. Pick an initial set of K centroids (this can be random or any other means)
2. For each data point, assign it to the member of the closest centroid according to the given distance function
3. Adjust the centroid position as the mean of all its assigned member data points. Go back to (2) until the membership isn't change and centroid position is stable.
4. Output the centroids.

```
library(stats)  
set.seed(101)  
km <- kmeans(iris[,1:4], 3)  
plot(iris[,1], iris[,2], col=km$cluster)  
points(km$centers[,c(1,2)], col=1:3, pch=19, cex=2)
```



```
table(km$cluster, iris$Species)
```

	setosa	versicolor	virginica
1	0	48	14
2	50	0	0
3	0	2	36

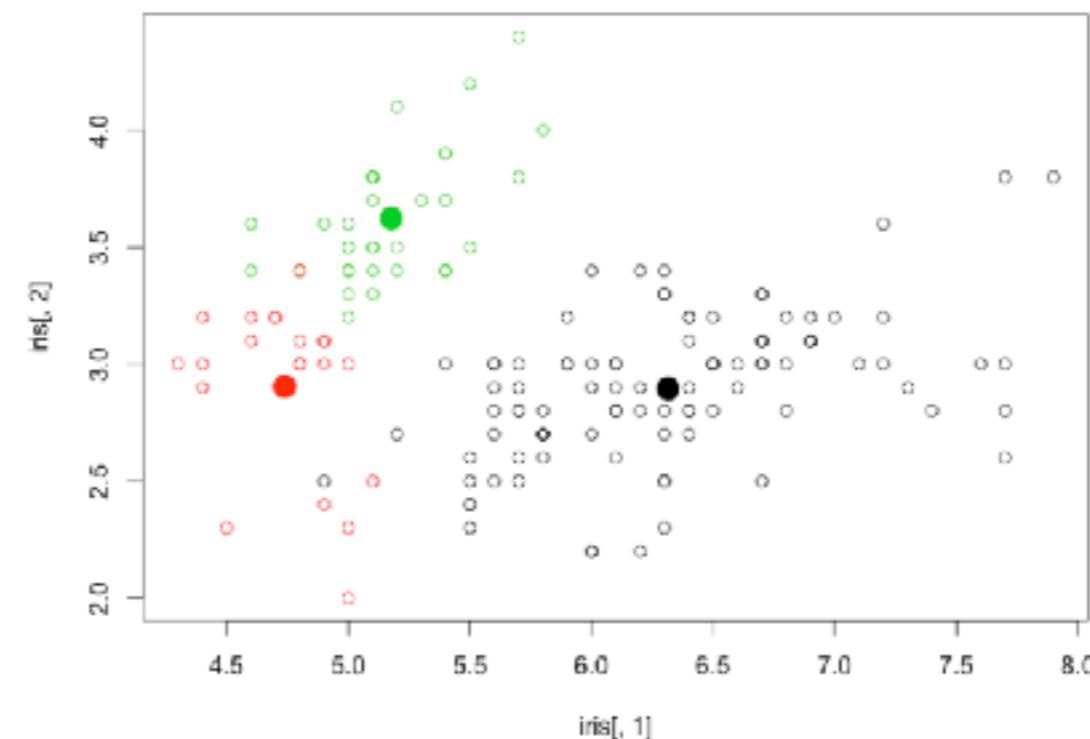
Another round

```
set.seed(900)
```

```
km <- kmeans(iris[,1:4], 3)
```

```
plot(iris[,1], iris[,2], col=km$cluster)
```

```
points(km$centers[,c(1,2)], col=1:3, pch=19, cex=2)
```



```
table(km$cluster, iris$Species)
```

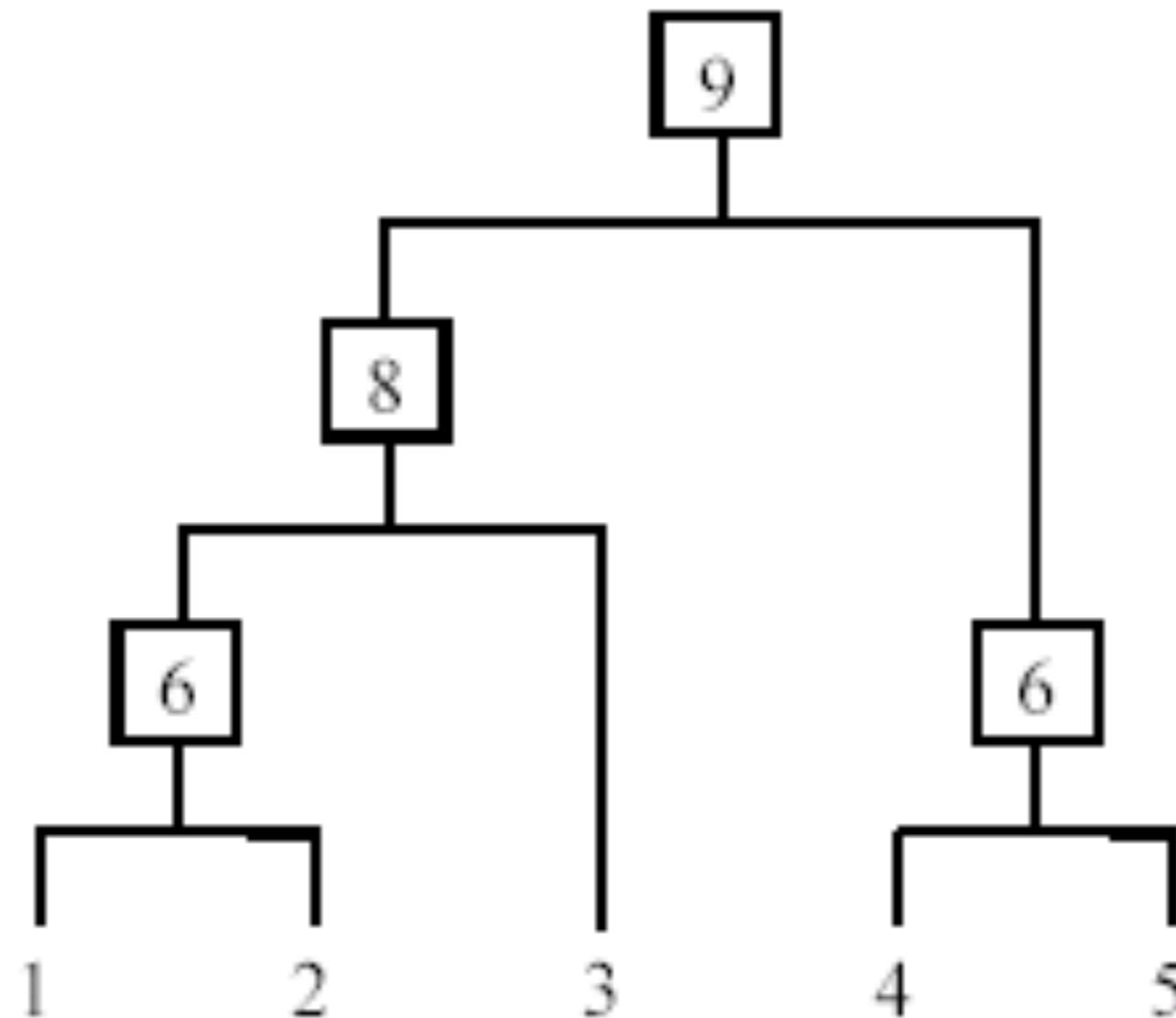
	setosa	versicolor	virginica
1	0	46	50
2	17	4	0
3	33	0	0
>			

Workshop 17 - k-Mean Clustering

1. Data that you import
2. Perform k-Mean clustering to divide data into K groups

Hierarchical Clustering

Produce a nested sequence of clusters, a **tree**, also called **Dendrogram**.



Types of hierarchical clustering

Agglomerative (bottom up) clustering: It builds the dendrogram (tree) from the bottom level, and

- merges the most similar (or nearest) pair of clusters
- stops when all the data points are merged into a single cluster (i.e., the root cluster).

Divisive (top down) clustering: It starts with all data points in one cluster, the root.

- Splits the root into a set of child clusters. Each child cluster is recursively divided further
- stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

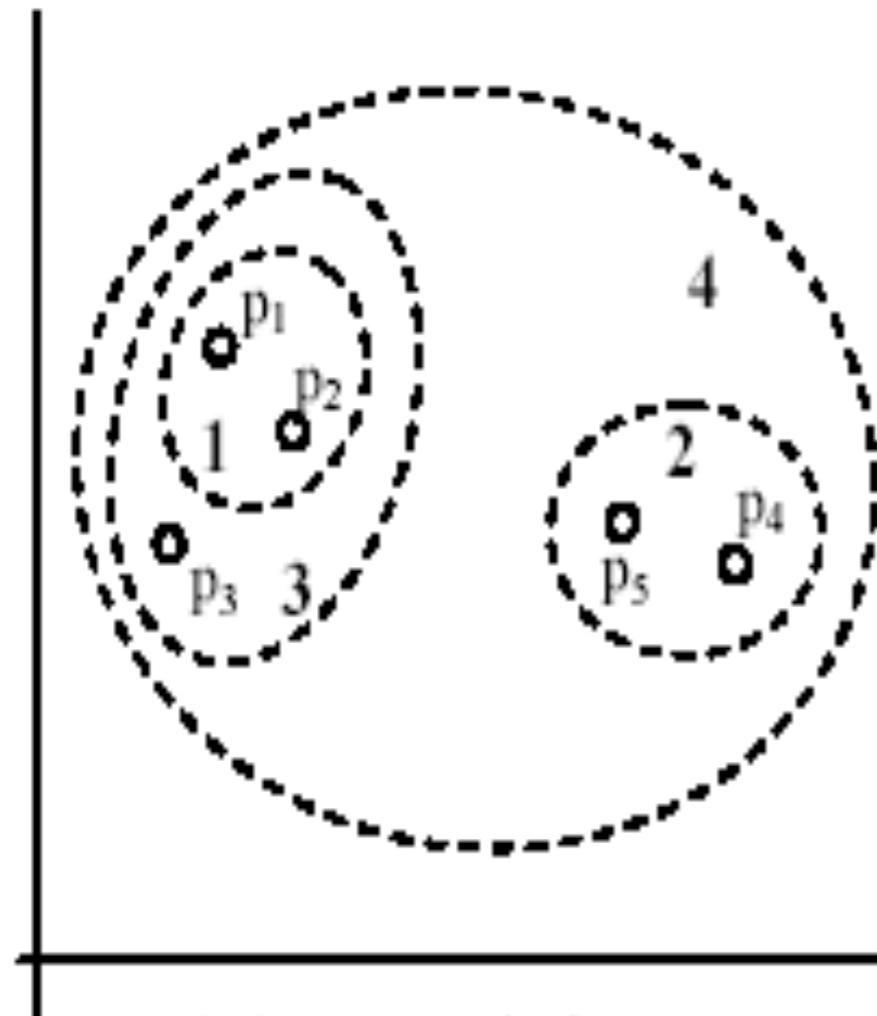
Agglomerative clustering

- It is more popular than divisive methods.
- At the beginning, each data point forms a cluster (also called a node).
- Merge nodes/clusters that have the least distance.
- Go on merging
- Eventually all nodes belong to one cluster

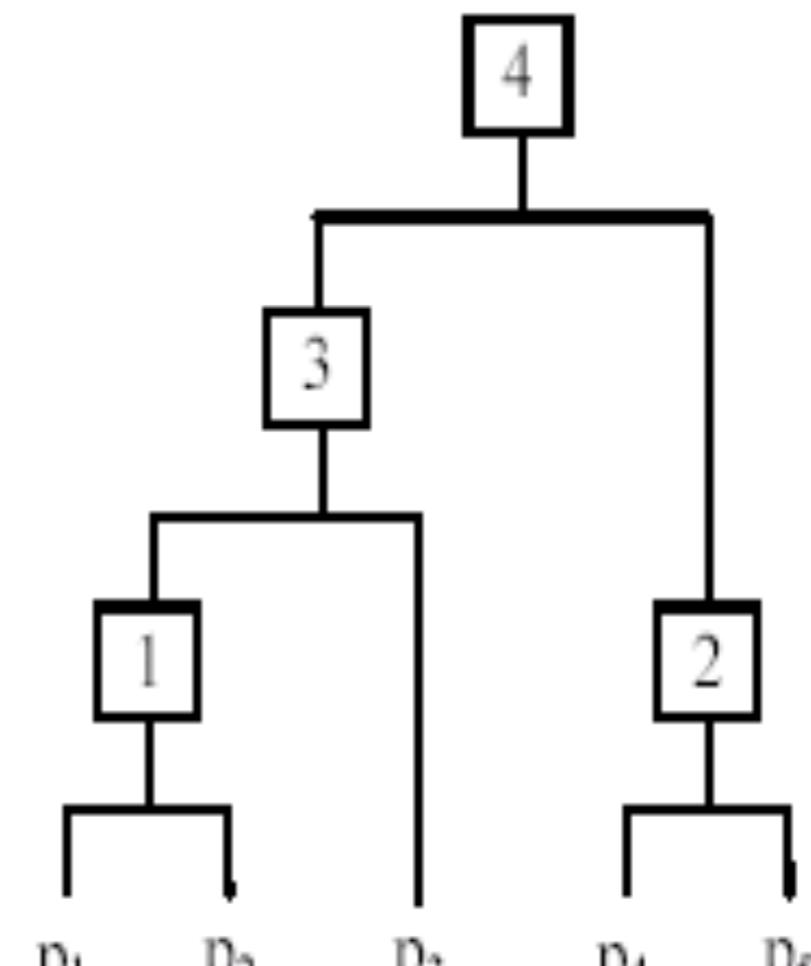
Agglomerative clustering algorithm

```
Algorithm Agglomerative( $D$ )
1   Make each data point in the data set  $D$  a cluster,
2   Compute all pair-wise distances of  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in D$ ;
2   repeat
3       find two clusters that are nearest to each other;
4       merge the two clusters form a new cluster  $c$ ;
5       compute the distance from  $c$  to all other clusters;
12  until there is only one cluster left
```

An example: working of the algorithm



(A). Nested clusters



(B) Dendrogram

Measuring the distance of two clusters

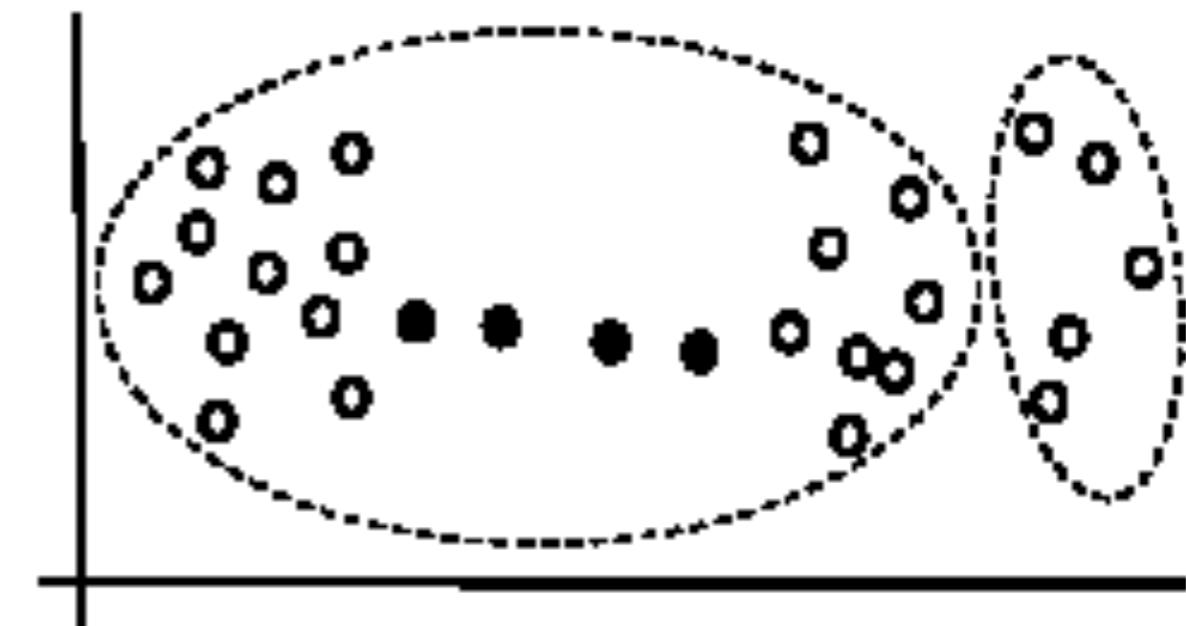
- A few ways to measure distances of two clusters.
- Results in different variations of the algorithm.
 - Single link
 - Complete link
 - Average link
 - Centroids
 - ...

Single link method

The distance between two clusters is the distance between two **closest data points** in the two clusters, one data point from each cluster.

It can find arbitrarily shaped clusters, but

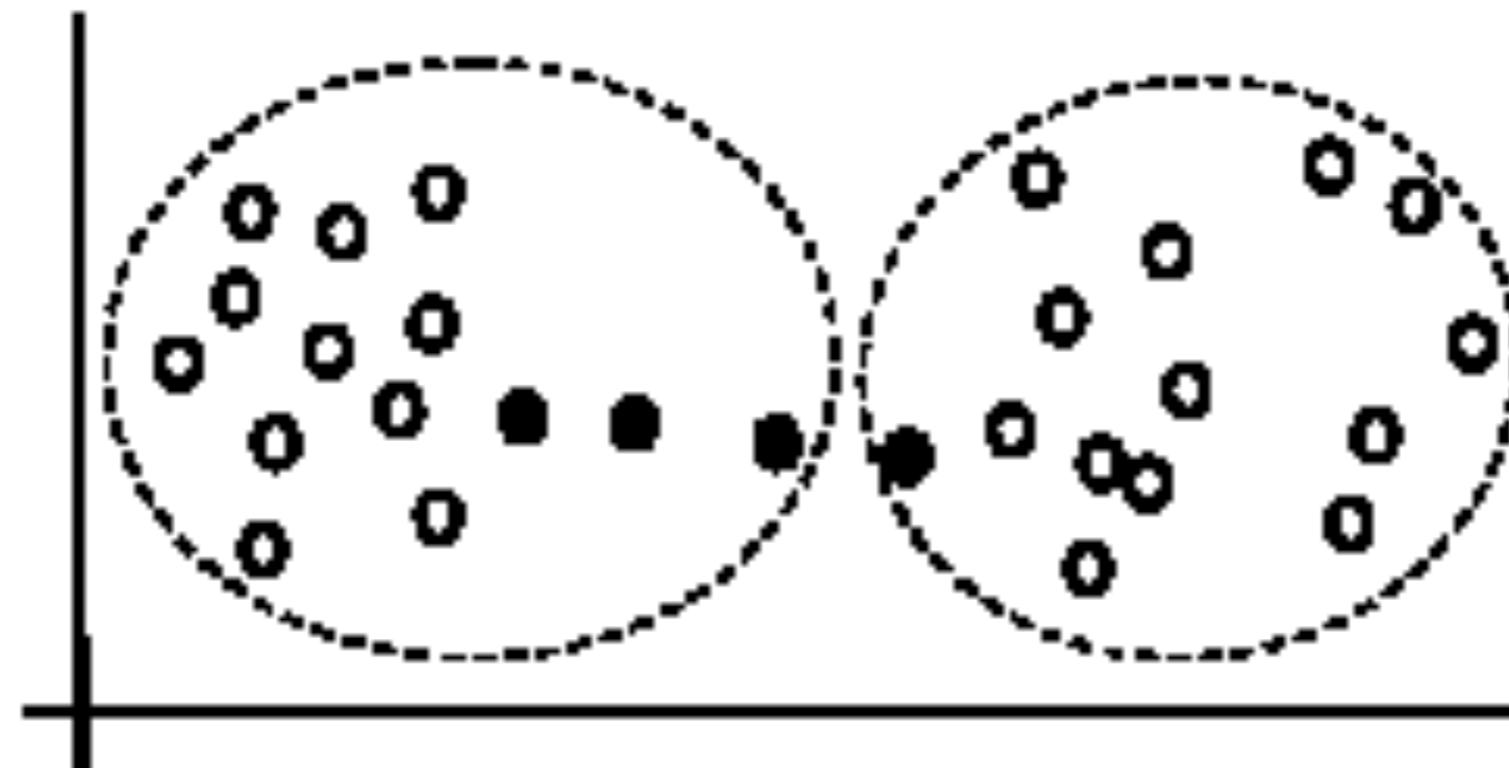
It may cause the undesirable “**chain effect**” by noisy points



Two natural clusters are split into two

Complete link method

- The distance between two clusters is the distance of two **furthest** data points in the two clusters.
- It is sensitive to outliers because they are far away



Average link and centroid methods

Average link: A compromise between

- the sensitivity of complete-link clustering to outliers and
- the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects.

In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.

Centroid method: In this method, the distance between two clusters is the distance between their centroids

Distance functions

- Key to clustering. “**similarity**” and “**dissimilarity**” can also commonly used terms.
- There are numerous distance functions for
 - Different types of data
 - Numeric data
 - Nominal data
 - Different specific applications

Distance functions for numeric attributes

Most commonly used functions are

Euclidean distance and

Manhattan (city block) distance

We denote distance with: $dist(\mathbf{x}_i, \mathbf{x}_j)$, where \mathbf{x}_i and \mathbf{x}_j are data points
(vectors)

They are special cases of **Minkowski distance**. h is positive integer.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \left((x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + \dots + (x_{ir} - x_{jr})^h \right)^{\frac{1}{h}}$$

Euclidean distance and Manhattan distance

If $h = 2$, it is the **Euclidean distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ir} - x_{jr})^2}$$

If $h = 1$, it is the **Manhattan distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ir} - x_{jr}|$$

Weighted Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \dots + w_r(x_{ir} - x_{jr})^2}$$

Squared distance and Chebychev distance

Squared Euclidean distance: to place progressively greater weight on data points that are further apart.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ir} - x_{jr})^2$$

Chebychev distance: one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \dots, |x_{ir} - x_{jr}|)$$

Distance function for text documents

- A text document consists of a sequence of sentences and each sentence consists of a sequence of words.
- To simplify: a document is usually considered a “bag” of words in document clustering.
 - Sequence and position of words are ignored.
- A document is represented with a vector just like a normal data point.
- It is common to use similarity to compare two documents rather than distance.
 - The most commonly used similarity function is the **cosine similarity**.

Data standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- Consider the following pair of data points
 - $\mathbf{x}_i: (0.1, 20)$ and $\mathbf{x}_j: (0.9, 720)$.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9 - 0.1)^2 + (720 - 20)^2} = 700.000457,$$

- The distance is almost completely dominated by $(720-20) = 700$.
- **Standardize attributes:** to force the attributes to have a common value range

Hierarchical Clustering in R

Fundamental Data Science for Data Scientist

Hierarchical Clustering

Compute distance between every pairs of point/cluster.

- (a) Distance between point is just using the distance function.
- (b) Compute distance between pointA to clusterB may involve many choices (such as the min/max/avg distance between the pointA and points in the clusterB).
- (c) Compute distance between clusterA to clusterB may first compute distance of all points pairs (one from clusterA and the other from clusterB) and then pick either min/max/avg of these pairs.

Combine the two closest point/cluster into a cluster. Go back to (1) until only one big cluster remains

```
set.seed(101)

sampleiris <- iris[sample(1:150, 40),] # get samples from iris dataset

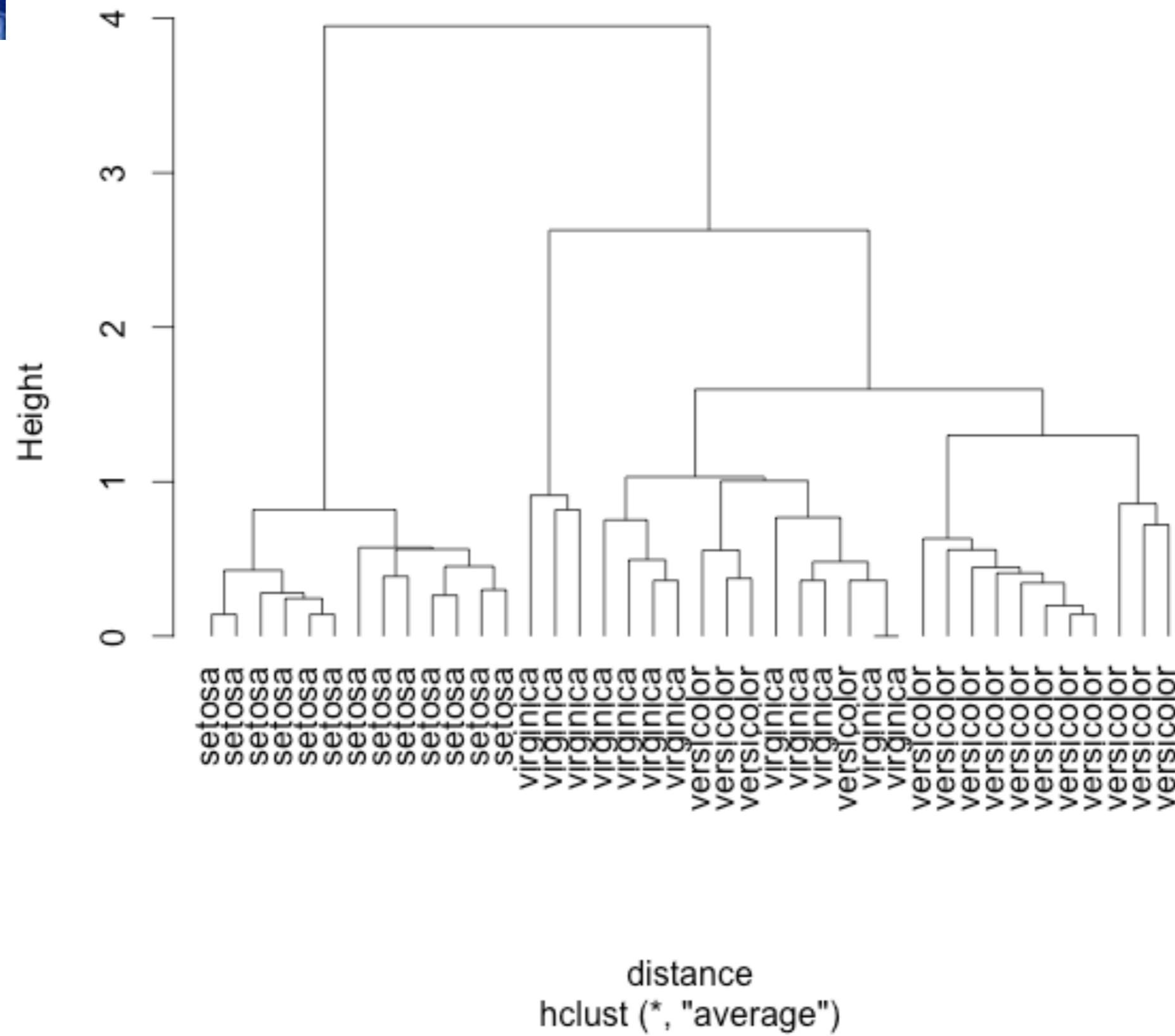
# each observation has 4 variables, ie, they are interpreted as 4-D points

distance <- dist(sampleiris[,-5], method="euclidean")

cluster <- hclust(distance, method="average")

plot(cluster, hang=-1, label=sampleiris$Species)
```

Cluster Dendrogram



It's possible to prune the result tree.

```
par(mfrow=c(1,2))

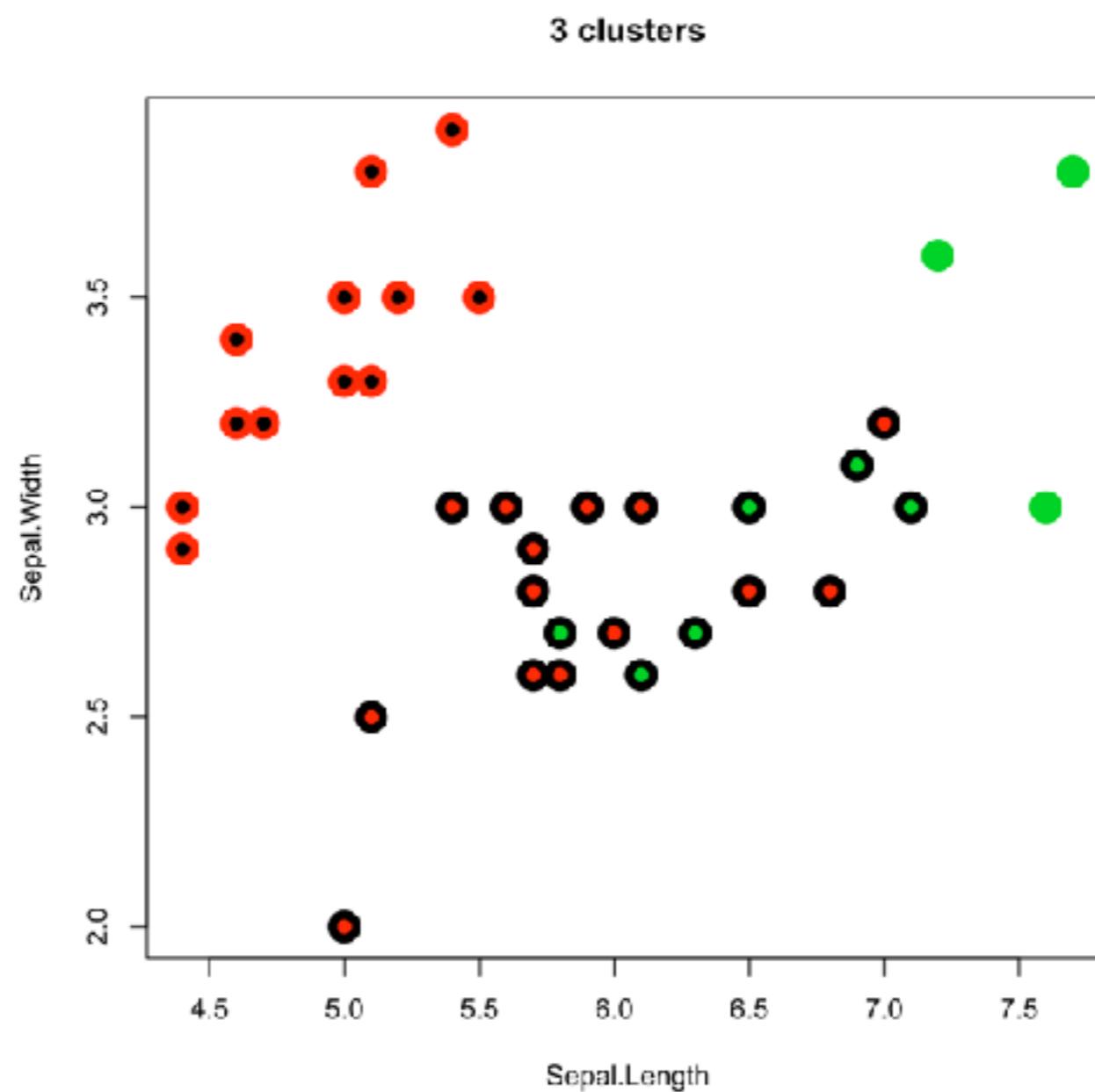
group.3 <- cutree(cluster, k = 3) # prune the tree by 3 clusters

table(group.3, sampleiris$Species) # compare with known classes
```

group.3	setosa	versicolor	virginica
1	0	15	9
2	13	0	0
3	0	0	3

> |

```
plot(sampleiris[,c(1,2)], col=group.3, pch=19, cex=2.5, main="3 clusters")
points(sampleiris[,c(1,2)], col=sampleiris$Species, pch=19, cex=1)
```



Workshop 18 - Hierarchical Clustering

1. From Data that you import
2. Perform Hierarchical clustering to divide data into 4 groups

Recommendation System

- Concept
- Building Recommendation System



Recommendation System Topics

- Basic concepts
- Apriori algorithm
- Summary

Association rule mining

- Proposed by **Agrawal et al in 1993.**
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for **Market Basket Analysis** to find how items purchased by customers are related.

Bread → Milk [sup = 5%, conf = 100%]

The model: data

$I = \{i_1, i_2, \dots, i_m\}$: a set of *items*.

Transaction t :

t a set of items, and $t \subseteq I$.

Transaction Database T : a set of transactions $T = \{t_1, t_2, \dots, t_n\}$.

Transaction data: supermarket data

Market basket transactions:

t1: {bread, cheese, milk}

t2: {apple, eggs, salt, yogurt}

...

...

tn: {biscuit, eggs, milk}

Concepts:

An *item*: an item/article in a basket

I: the set of all items sold in the store

A *transaction*: items purchased in a basket; it may have TID (transaction ID)

A *transactional dataset*: A set of transactions

Transaction data: a set of documents

A text document data set. Each document is treated as a “bag” of keywords

doc1: Student, Teach, School

doc2: Student, School

doc3: Teach, School, City, Game

doc4: Baseball, Basketball

doc5: Basketball, Player, Spectator

doc6: Baseball, Coach, Game, Team

doc7: Basketball, Team, City, Game

The model: rules

A transaction t contains X , a set of items (itemset) in I , if $X \subseteq t$.

An association rule is an implication of the form:

$X \rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \emptyset$

An itemset is a set of items.

E.g., $X = \{\text{milk, bread, cereal}\}$ is an itemset.

A k -itemset is an itemset with k items.

E.g., $\{\text{milk, bread, cereal}\}$ is a 3-itemset

Rule strength measures

Support: The rule holds with support sup in T (the transaction data set) if $sup\%$ of transactions contain $X \cup Y$.

$$sup = \Pr(X \cup Y).$$

Confidence: The rule holds in T with confidence $conf$ if $conf\%$ of transactions that contain X also contain Y .

$$conf = \Pr(Y | X)$$

An association rule is a pattern that states when X occurs, Y occurs with certain probability.

Support and Confidence

Support count: The support count of an itemset X , denoted by $X.count$, in a data set T is the number of transactions in T that contain X . Assume T has n transactions.

Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Goal and key features

Goal: Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).

Key Features

Completeness: find all rules.

No target item(s) on the right-hand-side

Mining with data on **hard disk** (not in memory)

An example

Transaction data

Assume:

$\text{minsup} = 30\%$

$\text{minconf} = 80\%$

An example **frequent itemset**:

{Chicken, Clothes, Milk} [sup = 3/7]

Association rules from the itemset:

Clothes \rightarrow Milk, Chicken [sup = 3/7, conf = 3/3]

...

...

Clothes, Chicken \rightarrow Milk, [sup = 3/7, conf = 3/3]



- t1: Beef, Chicken, Milk
- t2: Beef, Cheese
- t3: Cheese, Boots
- t4: Beef, Chicken, Cheese
- t5: Beef, Chicken, Clothes, Cheese, Milk
- t6: Chicken, Clothes, Milk
- t7: Chicken, Milk, Clothes

Transaction data representation

A simplistic view of shopping baskets,

Some important information not considered. E.g,

the quantity of each item purchased and

the price paid.

Many mining algorithms

There are a large number of them!!

They use different strategies and data structures.

Their resulting sets of rules are all the same.

Given a transaction data set T , and a minimum support and a minimum confident, the set of association rules existing in T is uniquely determined.

Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.

We study only one: **the Apriori Algorithm**

Recommendation System Topics

- Basic concepts
- Apriori algorithm
- Summary

The Apriori algorithm

Probably the best known algorithm

Two steps:

Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).

Use frequent itemsets to *generate rules*.

E.g., a frequent itemset

{Chicken, Clothes, Milk} [sup = 3/7]

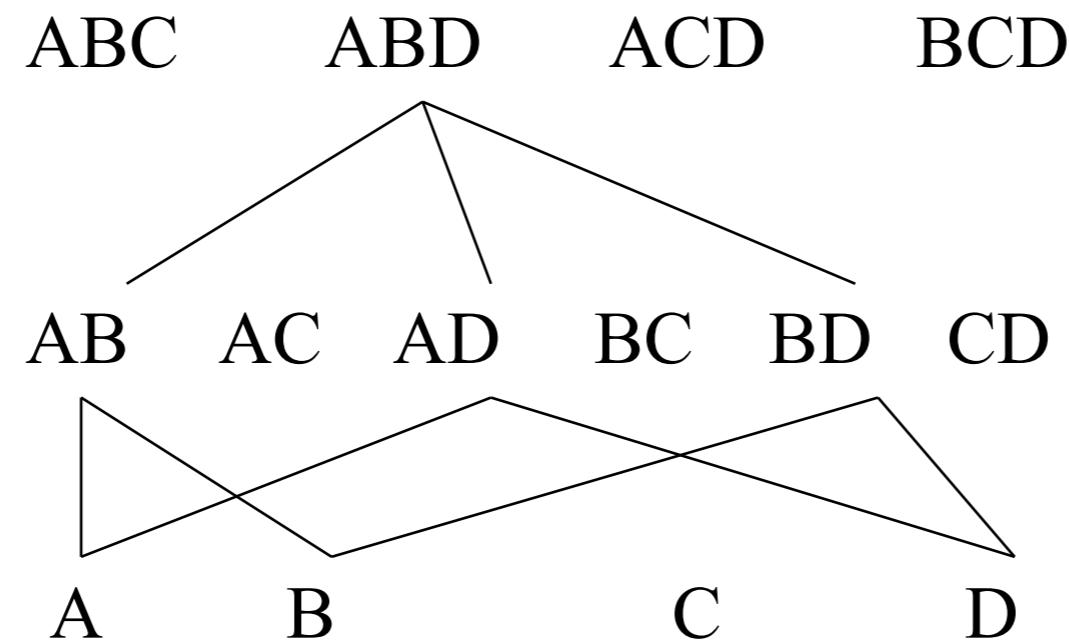
and one rule from the frequent itemset

Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

A **frequent itemset** is an itemset whose support is $\geq \text{minsup}$.

Key idea: The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



The Algorithm

Iterative algo. (also called **level-wise search**): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.

In each iteration k , only consider itemsets that contain some $k-1$ frequent itemset.

Find frequent itemsets of size 1: F_1

From $k = 2$

C_k = candidates of size k : those itemsets of size k that could be frequent, given F_{k-1}

F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once).

Example – Finding frequent itemsets

Dataset T
minsup=0.5

TID	Items
T100	1, 3, 4
T200	2, 3, 5
T300	1, 2, 3, 5
T400	2, 5

itemset:count

1. scan T → C₁: {1}:2, {2}:3, {3}:3, {4}:1, {5}:3

→ F₁: {1}:2, {2}:3, {3}:3, {5}:3

→ C₂: {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}

2. scan T → C₂: {1,2}:1, {1,3}:2, {1,5}:1, {2,3}:2, {2,5}:3, {3,5}:2

→ F₂: {1,3}:2, {2,3}:2, {2,5}:3, {3,5}:2

→ C₃: {2, 3, 5}

3. scan T → C₃: {2, 3, 5}:2 → F₃: {2, 3, 5}

Details: ordering of items

The items in I are sorted in **lexicographic order** (which is a total order).

The order is used throughout the algorithm in each itemset.

$\{w[1], w[2], \dots, w[k]\}$ represents a k -itemset w consisting of items $w[1], w[2], \dots, w[k]$, where $w[1] < w[2] < \dots < w[k]$ according to the total order.

Apriori candidate generation

The **candidate-gen** function takes F_{k-1} and returns a **superset** (called the candidates) of the set of all **frequent k -itemsets**. It has two steps

join step: Generate all possible candidate

itemsets C_k of length k

prune step: Remove those candidates in C_k that

cannot be frequent.

An example

$$\begin{aligned}F_3 = & \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \\& \{1, 3, 5\}, \{2, 3, 4\}\}\end{aligned}$$

After join

$$C_4 = \{\{1, 2, 3, 4\}, \{1, 3, 4, 5\}\}$$

After pruning:

$$C_4 = \{\{1, 2, 3, 4\}\}$$

because $\{1, 4, 5\}$ is not in F_3 ($\{1, 3, 4, 5\}$ is removed)

Step 2: Generating rules from frequent itemsets

Frequent itemsets \neq association rules

One more step is needed to generate association rules

For each frequent itemset X ,

For each proper nonempty subset A of X ,

Let $B = X - A$

$A \rightarrow B$ is an association rule if

$\text{Confidence}(A \rightarrow B) \geq \text{minconf}$,

$\text{support}(A \rightarrow B) = \text{support}(A \cup B) = \text{support}(X)$

$\text{confidence}(A \rightarrow B) = \text{support}(A \cup B) / \text{support}(A)$

Generating rules: an example

Suppose $\{2,3,4\}$ is frequent, with sup=50%

Proper nonempty subsets: $\{2,3\}$, $\{2,4\}$, $\{3,4\}$, $\{2\}$, $\{3\}$, $\{4\}$, with sup=50%, 50%, 75%, 75%, 75% respectively

These generate these association rules:

$2,3 \rightarrow 4$, confidence=100%

$2,4 \rightarrow 3$, confidence=100%

$3,4 \rightarrow 2$, confidence=67%

$2 \rightarrow 3,4$, confidence=67%

$3 \rightarrow 2,4$, confidence=67%

$4 \rightarrow 2,3$, confidence=67%

All rules have support = 50%

Generating rules: summary

To recap, in order to obtain $A \rightarrow B$, we need to have $\text{support}(A \cup B)$ and $\text{support}(A)$

All the required information for confidence computation has already been recorded in itemset generation. No need to see the data T any more.

This step is not as time-consuming as frequent itemsets generation.

On Apriori Algorithm

Seems to be very expensive

Level-wise search

K = the size of the largest itemset

It makes at most K passes over data

In practice, K is bounded (10).

The algorithm is very fast. Under some conditions, all rules can be found in **linear time**.

Scale up to large data sets

More on association rule mining

- Clearly the space of all association rules is **exponential**, $O(2^m)$, where m is the number of items in I .
- The mining exploits **sparseness of data**, and **high minimum support** and **high minimum confidence** values.
- Still, it always produces a **huge number of rules**, thousands, tens of thousands, millions, ...

Building Recommendation System in R



Support, Confidence and Lift

There are several measures used to understand various aspects of associated products.

Let's understand the measures with the help of an example.

- In a store, there are 1000 transactions overall.
- Item A appears in 80 transactions and
- Item B occurs in 100 transactions.
- Items A and B appear in 20 transactions together.



Support is the ratio of number of times two or more items occur together to the total number of transactions.

- **Support of A** = $\Pr(A) = 80/1000 = 8\%$ and
- **Support of B** = $\Pr(B) = 100/1000 = 10\%$.

Confidence is a conditional probability that a randomly selected transaction will include Item A given Item B.

- **Confidence of A** = $\Pr(A/B) = 20/100 = 20\%$.

Lift can be expressed as the ratio of the probability of Items A and B occurring together to the multiple of the two individual probabilities for Item A and Item B.

- **Lift** = $\Pr(A,B) / \Pr(A).\Pr(B) = (20/1000)/((80/1000)\times(100/1000)) = 2.5$.

How would you use Support, Confidence and Lift?

Support of a product or product bundle indicates the popularity of the product or product bundle in the transaction set. Higher the support, more popular is the product or product bundle. This measure can help in identifying driver of traffic to the store. Hence, if Barbie dolls have a higher support then they can be attractively priced to attract traffic to a store.

Confidence can be used for product placement strategy and increasing profitability. Place high-margin items with associated high selling (driver) items. If Market Basket Analysis indicates that customers who bought high selling Barbie dolls also bought high-margin candies, then candies should be placed near Barbie dolls.

Lift indicates the strength of an association rule over the random co-occurrence of Item A and Item B, given their individual support. Lift provides information about the change in probability of Item A in presence of Item B. Lift values greater than 1.0 indicate that transactions containing Item B tend to contain Item A more often than transactions that do not contain Item B.

Apriori Algorithm

?apriori

Usage

```
apriori(data, parameter = NULL, appearance = NULL, control = NULL)
```

Arguments

`data`

object of class [transactions](#) or any data structure which can be coerced into [transactions](#) (e.g., a binary matrix or data.frame).

`parameter`

object of class [APparameter](#) or named list. The default behavior is to mine rules with support 0.1, confidence 0.8, and maxlen 10.

`appearance`

object of class [APappearance](#) or named list. With this argument item appearance can be restricted. By default all items can appear unrestricted.

`control`

object of class [APcontrol](#) or named list. Controls the performance of the mining algorithm (item sorting, etc.)

Apriori Algorithm

So lets get started by loading up our libraries and data set.

```
# Load the libraries
```

```
library(arules)
```

```
library(arulesViz)
```

```
library(datasets)
```

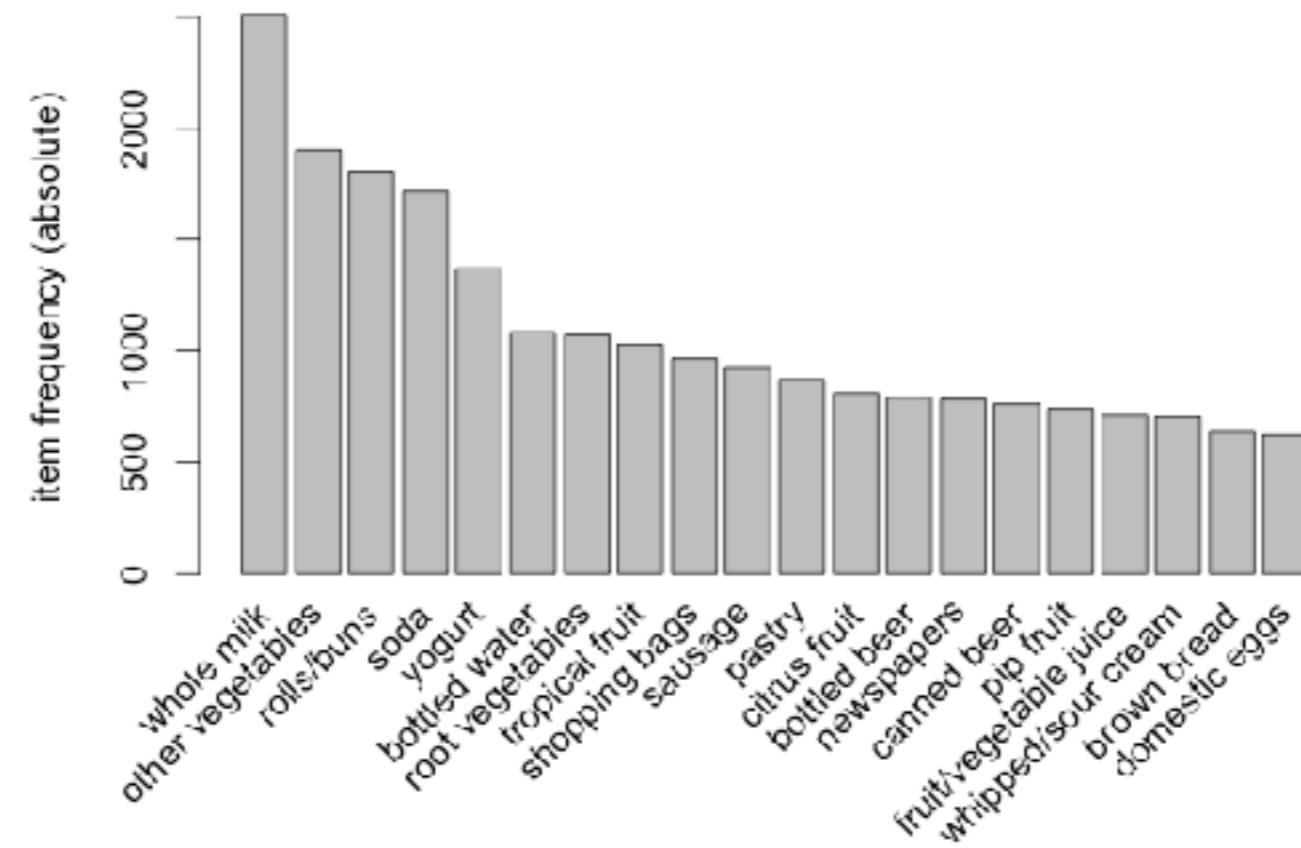
```
# Load the data set
```

```
data(Groceries)
```

Explore Data

Create an item frequency plot for the top 20 items

```
itemFrequencyPlot(Groceries, topN=20,type="absolute")
```



```
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))
```

```
# Show the top 5 rules, but only 2 digits
```

```
options(digits=2)
```

```
inspect(rules[1:5])
```

lhs	rhs	support	confidence	lift
1 {liquor, red/blush wine} => {bottled beer}	0.0019	0.90	11.2	
2 {curd, cereals} => {whole milk}	0.0010	0.91	3.6	
3 {yogurt, cereals} => {whole milk}	0.0017	0.81	3.2	
4 {butter, jam} => {whole milk}	0.0010	0.83	3.3	
5 { soups, bottled beer} => {whole milk}	0.0011	0.92	3.6	
>				

```
summary(rules)
```

set of 410 rules

rule length distribution (lhs + rhs):sizes

3	4	5	6
29	229	140	12

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
3.0	4.0	4.0	4.3	5.0	6.0

summary of quality measures:

support	confidence	lift
Min. :0.00102	Min. :0.80	Min. : 3.1
1st Qu.:0.00102	1st Qu.:0.83	1st Qu.: 3.3
Median :0.00122	Median :0.85	Median : 3.6
Mean :0.00125	Mean :0.87	Mean : 4.0
3rd Qu.:0.00132	3rd Qu.:0.91	3rd Qu.: 4.3
Max. :0.00315	Max. :1.00	Max. :11.2

mining info:

data	ntransactions	support	confidence
Groceries	9835	0.001	0.8

Sort Rules

```
rules<-sort(rules, by="confidence", decreasing=TRUE)
```

lhs	rhs	support	confidence	lift
1 {rice, sugar}	=> {whole milk}	0.0012	1	3.9
2 {canned fish, hygiene articles}	=> {whole milk}	0.0011	1	3.9
3 {root vegetables, butter, rice}	=> {whole milk}	0.0010	1	3.9
4 {root vegetables, whipped/sour cream, flour}	=> {whole milk}	0.0017	1	3.9
5 {butter, soft cheese, domestic eggs}	=> {whole milk}	0.0010	1	3.9

Change to have limit association in one

```
# change to have maximum of 3  
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf =  
0.8,maxlen=3))  
inspect(rules[1:5])
```

	lhs	rhs	support	confidence	lift
1	{liquor, red/blush wine}	=> {bottled beer}	0.0019	0.90	11.2
2	{curd, cereals}	=> {whole milk}	0.0010	0.91	3.6
3	{yogurt, cereals}	=> {whole milk}	0.0017	0.81	3.2
4	{butter, jam}	=> {whole milk}	0.0010	0.83	3.3
5	{ soups, bottled beer}	=> {whole milk}	0.0011	0.92	3.6

Rules pruned

```
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned
summary(rules)
```

```
set of 330 rules

rule length distribution (lhs + rhs):sizes
 3   4   5   6
 29  216  84   1

      Min. 1st Qu. Median  Mean 3rd Qu. Max.
      3.0    4.0    4.0   4.2   5.0    6.0

summary of quality measures:
      support      confidence       lift
      Min. :0.00102  Min. :0.80  Min. : 3.1
      1st Qu.:0.00102  1st Qu.:0.82  1st Qu.: 3.3
      Median :0.00122  Median :0.85  Median : 3.6
      Mean   :0.00127  Mean   :0.86  Mean   : 3.8
      3rd Qu.:0.00132  3rd Qu.:0.91  3rd Qu.: 4.3
      Max.  :0.00315  Max.  :1.00  Max.  :11.2

mining info:
      data ntransactions support confidence
      Groceries          9835     0.001         0.8
```

Targeting Items

What are customers likely to buy before buying whole milk?

What are customers likely to buy if they purchase whole milk?

This essentially means we want to set either the Left Hand Side and Right Hand Side. This is not difficult to do with R!

Find whole milk's antecedents

```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf =  
0.08), appearance = list(default="lhs",rhs="whole milk"), control =  
list(verbose=F))
```

```
rules<-sort(rules, decreasing=TRUE,by="confidence")
```

```
inspect(rules[1:5])
```

	lhs	rhs	support	confidence	lift
1	{rice, sugar}	=> {whole milk}	0.0012	1	3.9
2	{canned fish, hygiene articles}	=> {whole milk}	0.0011	1	3.9
3	{root vegetables, butter, rice}	=> {whole milk}	0.0010	1	3.9
4	{root vegetables, whipped/sour cream, flour}	=> {whole milk}	0.0017	1	3.9
5	{butter, soft cheese, domestic eggs}	=> {whole milk}	0.0010	1	3.9

Likely to buy after buy whole milk

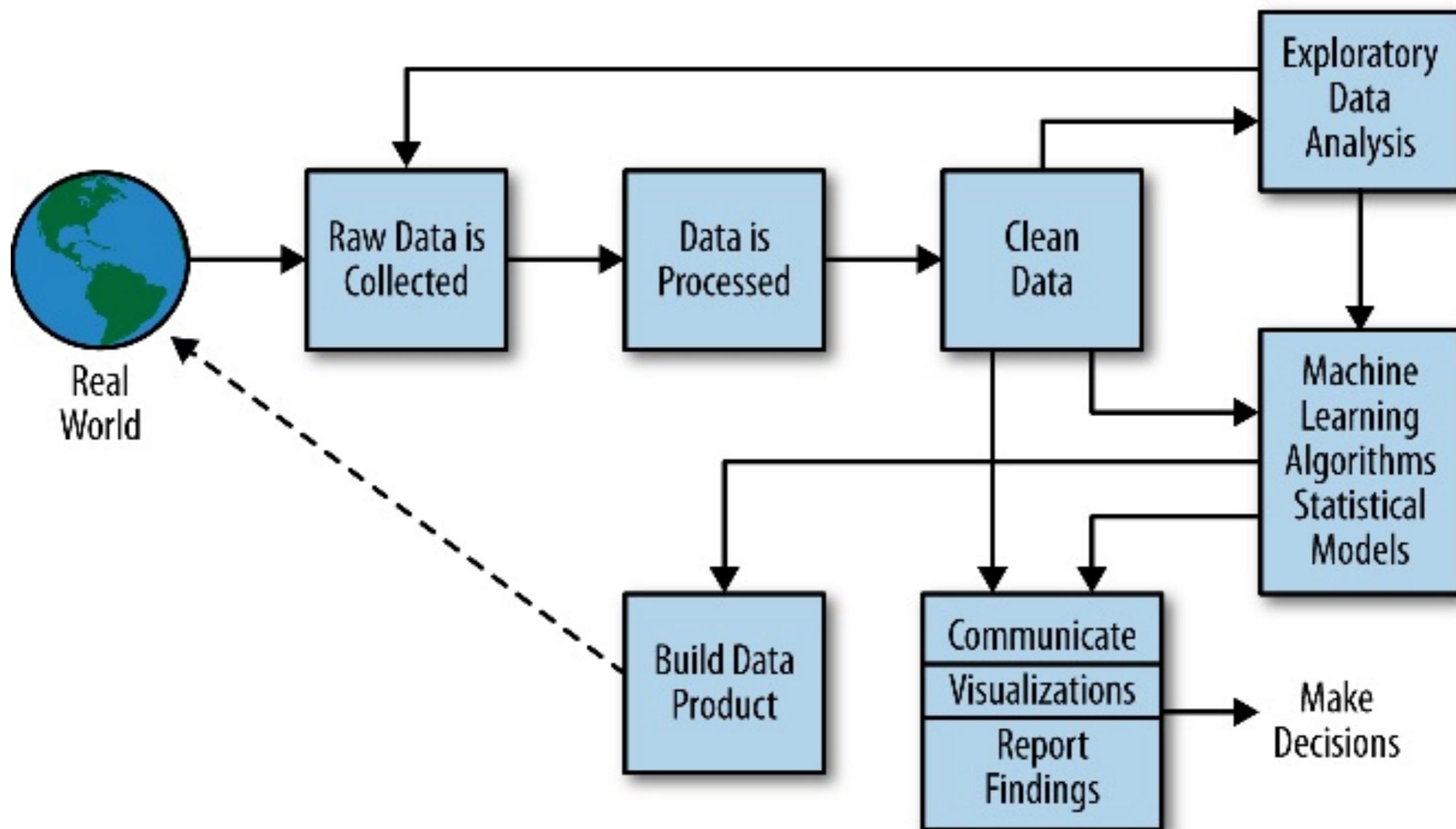
```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),  
appearance = list(default="rhs",lhs="whole milk"), control = list(verbose=F))  
rules<-sort(rules, decreasing=TRUE,by="confidence")  
inspect(rules[1:5])
```

	lhs	rhs	support	confidence	lift
1	{whole milk} =>	{other vegetables}	0.075	0.29	1.5
2	{whole milk} =>	{rolls/buns}	0.057	0.22	1.2
3	{whole milk} =>	{yogurt}	0.056	0.22	1.6
4	{whole milk} =>	{root vegetables}	0.049	0.19	1.8
5	{whole milk} =>	{tropical fruit}	0.042	0.17	1.6
>					

Workshop 19- Build Recommendation System

1. Use Titanic Data download from [http://www.rdatamining.com/data/
titanic.raw.rdata?attredirects=0&d=1](http://www.rdatamining.com/data/titanic.raw.rdata?attredirects=0&d=1)
2. Generate Association Rules
3. set rhs=c("Survived=No", "Survived=Yes") in appearance to make sure that
only "Survived=No" and "Survived=Yes" will appear in the rhs of rules

Conclusion





veerasak.kritsanaphan@gmail.com