Hewlett Packard Enterprise

Data Analytics

CFA Finance

March 2019

Course Objectives

- Define what you want to know
- Define what data you need to answer the question(s)
- Gather the data
- Learn basic R concepts
- Clean the data
- Explore the data
- Use data to answer to the question(s)
  - Case Study: Linear Regression
  - Case Study: Machine Learning
- Document the analysis

Define what you want to know

Define what you want to know

Stop and reflect

- The last time you wanted to know something…
  - What was it you wanted to know?
  - How did you go about getting an answer?

- We perform analysis often, even for simple day-to-day tasks
  - I have to catch a flight tomorrow; what time should I wake up?
  - Do I need to pick up milk at the grocery store?
  - What should I get my spouse for our anniversary?

- We don't perform analysis without wanting to learn something
  - SO…don't start your analysis until you know what the question is!

Define what you want to know

Types of questions

- Descriptive
  - Explains an outcome
  - Focuses on the past
  - Helps to understand why something happened
  - Usually presented in scorecards and dashboards
  - The easiest type of analysis
- Predictive
  - Estimates the likelihood of future outcomes
  - No algorithm can be certain
- Prescriptive
  - Intended to describe actions or decisions to arrive at a probable (and desired) outcome
  - Requires actionable data and a feedback loop
  - The most complex type of analysis

Define what you want to know

Tree of questions

- Typically, answering a question may lead to more questions
  - What if I categorize or stratify my data?
  - What if I filter my data to meet certain conditions (date ranges, regions)

- We should anticipate answers creating more questions
  - The questions may progress from descriptive to predictive, and predictive to prescriptive

- Document the path of thinking
  - What was the initial question?
  - What questions branched off from that answer?

Define what data you need

Hewlett Packard Enterprise

Define what data you need

Stop and reflect

What time do I need to wake up to catch my flight?	Do I need to pick up milk at the grocery store?	What should I get my spouse as an anniversary gift?
• What time is the flight? • What time do I need to arrive at the airport? • How long does it take to get to the airport? • How long will it take me to pack? • Do I eat breakfast at home or at the airport? • Where will I be parking my car?	• How much milk do we have left? • How quickly have we been using milk? • Is the milk expired?	• Is this a milestone anniversary? • What did I get my spouse last year? • What did my spouse give me last year? • Did I forgot my anniversary?

Define what data you need

Types of data

What time do I need to wake up to catch my flight?	Do I need to pick up milk at the grocery store?	What should I get my spouse as an anniversary gift?
• What time is the flight? • What time do I need to arrive at the airport? • How long does it take to get to the airport? • How long will it take me to pack? • Do I eat breakfast at home or at the airport? • Where will I be parking my car?	• How much milk do we have left? • How quickly have we been using milk? • Is the milk expired?	• Is this a milestone anniversary? • What did I get my spouse last year? • What did my spouse give me last year? • Did I forgot my anniversary?

Some data are simply facts (they have a single, undisputable value)

Define what data you need

Types of data

What time do I need to wake up to catch my flight?	Do I need to pick up milk at the grocery store?	What should I get my spouse as an anniversary gift?
• What time is the flight? • What time do I need to arrive at the airport? • How long does it take to get to the airport? • How long will it take me to pack? • Do I eat breakfast at home or at the airport? • Where will I be parking my car?	• How much milk do we have left? • How quickly have we been using milk? • Is the milk expired?	• Is this a milestone anniversary? • What did I get my spouse last year? • What did my spouse give me last year? • Did I forgot my anniversary?

Some data are simply facts (they have a single, undisputable value)

Some data can be described by their characteristics (count, average, median, variance)

Hewlett Packard Enterprise

Define what data you need

Types of data

What time do I need to wake up to catch my flight?	Do I need to pick up milk at the grocery store?	What should I get my spouse as an anniversary gift?
• What time is the flight? • What time do I need to arrive at the airport? • How long does it take to get to the airport? • How long will it take me to pack? • Do I eat breakfast at home or at the airport? • Where will I be parking my car?	• How much milk do we have left? • How quickly have we been using milk? • Is the milk expired?	• Is this a milestone anniversary? • What did I get my spouse last year? • What did my spouse give me last year? • Did I forgot my anniversary?

Some data are simply facts (they have a single, undisputable value)

Some data can be described by their characteristics (count, average, median, variance)

Some data can have conditional values (the answer can vary based on the condition)

Hewlett Packard Enterprise

Define what data you need

Types of data

What time do I need to wake up to catch my flight?	Do I need to pick up milk at the grocery store?	What should I get my spouse as an anniversary gift?
• What time is the flight? • What time do I need to arrive at the airport? • How long does it take to get to the airport? • How long will it take me to pack? • Do I eat breakfast at home or at the airport? • Where will I be parking my car?	• How much milk do we have left? • How quickly have we been using milk? • Is the milk expired?	• Is this a milestone anniversary? • What did I get my spouse last year? • What did my spouse give me last year? • Did I forgot my anniversary?

Some data are simply facts (they have a single, undisputable value)

Some data can be described by their characteristics (count, average, median, variance)

Some data can have conditional values (the answer can vary based on the condition)

Some data may require a feedback loop (did previous instances create positive or negative results)

Define what data you need

Be thorough

– Make sure to identify every piece of data needed
  – Do not limit yourself to data that is readily available
  – Research may be required to get some data
  – Note if data is proprietary or has other protections to consider

– Just like with questions, you may discover as you define data needs, it will create branches with more data needs

Gather the data

Hewlett Packard Enterprise

Gather the data

Does the data exist?

- If the data does not exist, develop a plan for creating it
  - What is the name of the data?
  - Is the data discrete or continuous?
  - What is the operational definition of the data?
  - What is the frequency of the data?
  - What are the units?

- The data collection plan should be detailed enough that anyone could follow the plan and generate the same data set as anyone else

- If the data is to be used in an operational setting, then a more thorough plan should be developed so that team members are accountable and responsible for gathering and storing the data

Gather the data

Can you access the data?

- Treat data as property; there may be instances where you do not have sufficient rights to access it
- If the data belongs to the company, follow any necessary steps to get access rights
  - And remember the first bullet point
- If the data belongs to someone else, determine what steps are needed to gain access
  - It may require a registration, acceptance of terms
  - If costs are involved, determine if it is a one-time or recurring cost and whether or not funding is available
  - And remember the first bullet point

- In all cases, state the name of the source and the date the data was accessed in your analysis

Gather the data

Is there enough data?

– Creating or removing subgroups could generate more data points

– If there are enough observations but data is missing, use imputing methods to fill in the blanks
  – Random imputation (select a value between a known range)
  – Last Observation Carried Forward or Next Observation Carried Backward (useful for time-series data)
  – Positional imputation (use the mean, median or mode as a replacement)
  – Linear regression (create a regression model to find an acceptable replacement)
  – K Nearest Neighbors (KNN) (determine the most frequent discrete value or mean among the ‘k’ nearest neighbors)

– If there are not enough observations, consider bootstrapping methods
  – Bootstrapping is a method of drawing and replacing values out of a sample to generate more observations
  – Other methods include bagging (creating observations based on the descriptive values of smaller samples)

Gather the data

Points to remember

– Because your analysis should be reproducible, make sure to note the source and when the data was gathered

– Because the data almost always has an owner, get any necessary approvals prior to using the data, and make sure to acknowledge ownership in the analysis

Intro to R

Hewlett Packard
Enterprise

Intro to R

Concepts and benefits

– A language and an environment for statistical computing and graphics

– A highly extensible Open Source tool

– Compiles and runs on various platforms including UNIX, Linux, Windows, MacOS and others

– An interpreted language (it does not need to be compiled prior to execution)

Intro to R

Concepts and benefits

– Not only highly extensible, but also has a very active user community continually developing and refining installable packages

– Runs entirely in memory (instead of using a hard drive/memory hybrid) for faster processing

– Can either execute commands at a prompt or write and load functions to be used

– Supports matrix arithmetic

Intro to R

Concepts and benefits

Pros	Cons
Vast package ecosystem – if a statistical technique exists, odds are there's a package for it	Memory management not a primary design
Freeware – source code and everything about it are free to look at	Security capabilities are not built into it
Several popular machine learning algorithms are implemented in R, such as generalized additive model, nearest neighbors, linear discriminant analysis, neural network, and random forest	Code cannot be embedded into a web browser, although certain packages (like Shiny) allow for creating web interfaces
Powerful graphics and charting capabilities	Not a lot of interactivity in the language
Computer science background isn't required to get started	

Intro to R

Getting Started

Throughout the presentation, actual lines of code are presented using the following format:

```
>x <- 2
>y <- 3
>z <- x * y
>print(z)
```

As you go through the slides, practice using the R console by typing in the commands yourself. The output to the console screen should match what is shown on the slide

```
[1]  6
```

Hewlett Packard Enterprise

Intro to R

Creating a vector

– Instead of creating a ‘loop’ to perform an operation on a set of variables, R stores the set as a single variable (an ordered vector) and any operation on that vector happens to every element

– Use the ‘c’ command (which means *combine*) to create an ordered vector, then perform an operation on it:

```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>print(y)
[1]  1  4  9 16 25 36
```

x	1	2	3	4	5	6
y	1	4	9	16	25	36

Note that while it is standard practice to assign values with “<-” R accepts “=” as well

Hewlett Packard Enterprise

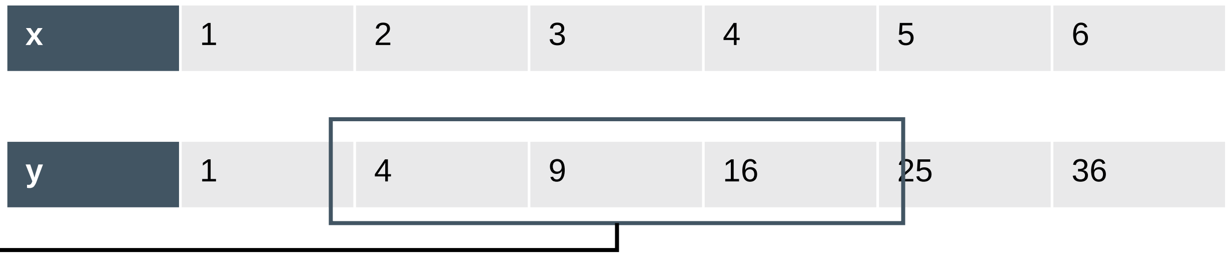
Intro to R

Referencing elements of a vector

– Any element of the ordered vector can be referenced using brackets

– Any number of elements can be returned, even non-contiguous values

```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>print(y[2:4])
[1]  4  9 16
```



Hewlett Packard Enterprise

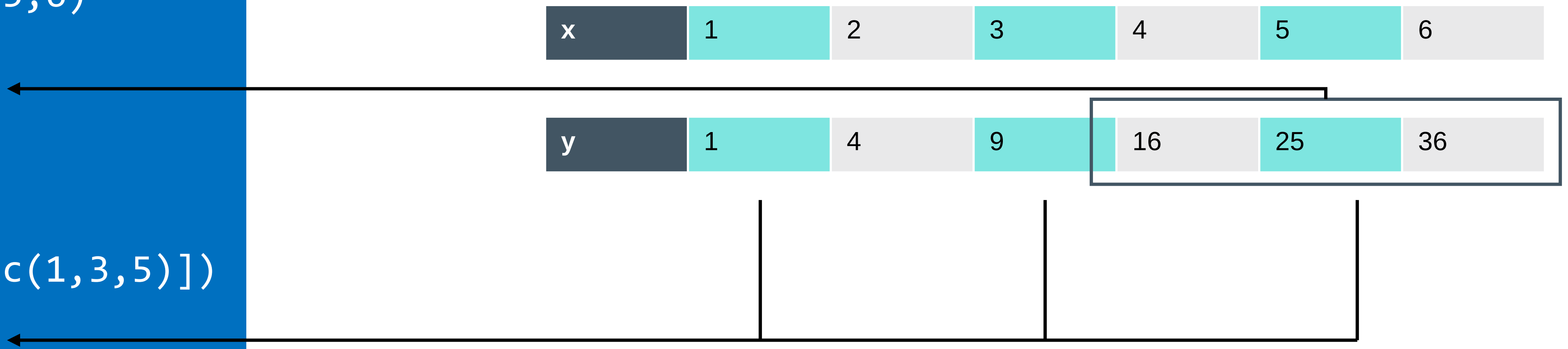
Intro to R

Referencing elements of a vector

– Logical conditions can also be applied as a reference

– As long as the vector lengths are the same, ‘masks’ of other vectors can also be used

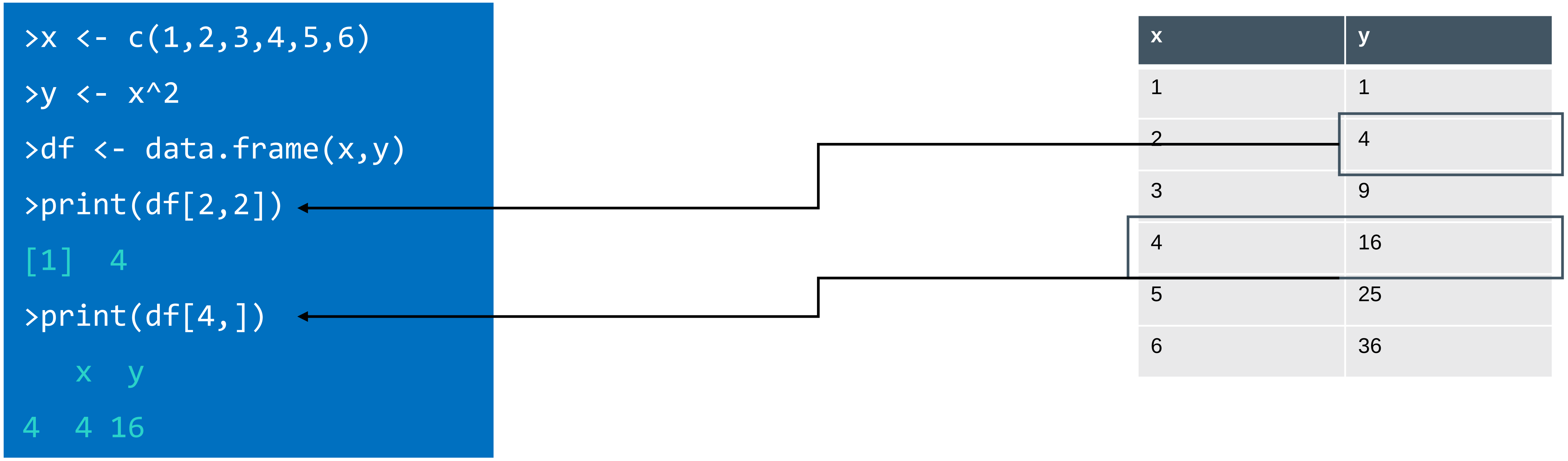
```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>print(y[y > 9])
[1] 16 25 36
>print(y[x %in% c(1,3,5)])
[1]  1  9 25
```



Intro to R

Referencing elements of a data frame

- Think of data frames (and matrices) as a collection of vectors
  - All vector lengths must be the same, but can have varying types
- The brackets must contain two dimensions
  - If a row or column index isn't specified, it will consider every element and return a vector



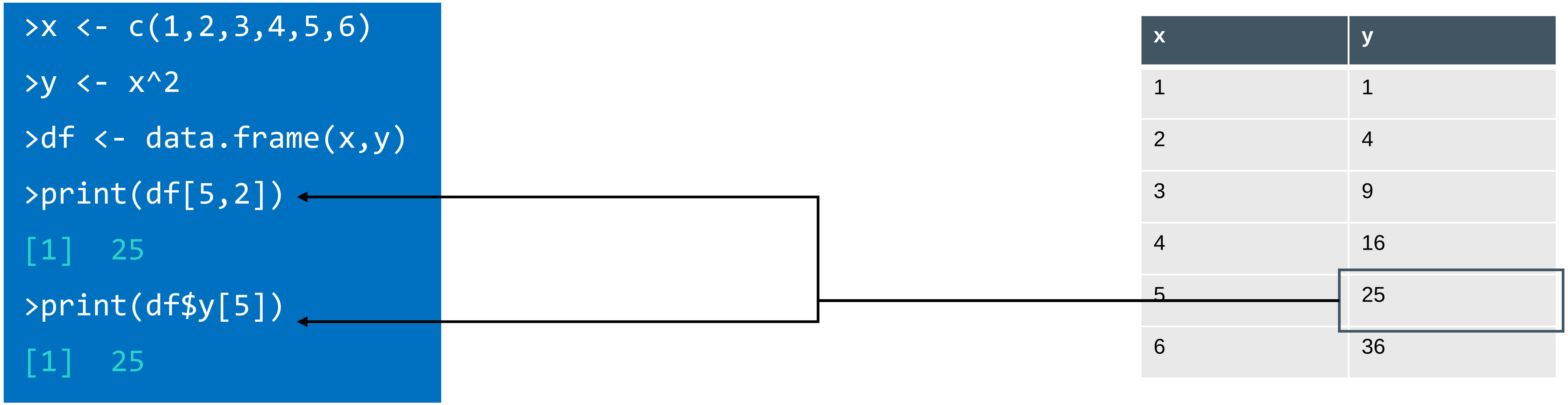
```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>df <- data.frame(x,y)
>print(df[2,2])
[1]  4
>print(df[4,])
   x  y
4  4 16
```

x	y
1	1
2	4
3	9
4	16
5	25
6	36

Intro to R

Referencing elements of a data frame

- There are two ways to specify a variable (column) in a data frame
  - Specify a value for the second argument in brackets
  - Specify the variable name with a ‘$’ preceding it
  - When using the $ method, note that only one argument is needed in brackets since it is a reference to a vector



```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>df <- data.frame(x,y)
>print(df[5,2])
[1]  25
>print(df$y[5])
[1]  25
```

x	y
1	1
2	4
3	9
4	16
5	25
6	36

Hewlett Packard Enterprise

Intro to R

Referencing elements of a data frame

– Just like earlier, logical statements can be used within the brackets

```
>x <- c(1,2,3,4,5,6)

>y <- x^2

>df <- data.frame(x,y)

>print(df[df$x<5,])

   x  y

1  1  1

2  2  4

3  3  9

4  4 16
```

x	y
1	1
2	4
3	9
4	16
5	25
6	36

Intro to R

Referencing elements of a data frame

– Finally, you can use a separate variable to store a ‘logical mask’ and use it within the brackets

```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>df <- data.frame(x,y)
>mask <- df$x %in% c(1,3,6)
>print(mask)
[1] TRUE FALSE TRUE FALSE FALSE TRUE
>print(df$y[mask])
[1]  1  9 36
```

x	y
1	1
2	4
3	9
4	16
5	25
6	36

Hewlett Packard
Enterprise

Intro to R

Properties of a data frame

– The `str()` function (structure) describes the number of observations (rows), variables (columns), the data type, and the first few values for each

```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>df <- data.frame(x,y)
>str(df)
'data.frame': 6 obs. of 2 variables:
$ x: num  1 2 3 4 5 6
$ y: num  1 4 9 16 25 36
```

Each column is considered a variable

Each row is considered an observation

x	y
1	1
2	4
3	9
4	16
5	25
6	36

Hewlett Packard Enterprise

Intro to R

Summary of a data table

– The `summary()` function provides some basic statistics for each variable

  – If the data type is numeric, it will return the minimum, first quartile, median, mean, third quartile, and maximum

  – Otherwise, it will return frequency counts for each value

```
>x <- c("A", "B", "A", "C", "C", "A")
>y <- c(1, 2, 3, 4, 5, 6)
>df <- data.frame(x,y)
>summary(df)
```

```
 x             y
 A:3     Min.   :1.00
 B:1     1st Qu.:2.25
 C:2     Median :3.50
         Mean   :3.50
         3rd Qu.:4.75
         Max.   :6.00
```

Hewlett Packard
Enterprise

Intro to R

Attributes of a data table (and other objects)

– The `attributes()` function, which can be used on objects other than data tables, describes additional attributes, many of which can take assignments

```
>x <- c(1,2,3,4,5,6)
>y <- x^2
>df <- data.frame(x,y)
>attributes(df)
$names
[1]  "x" "y"
$row.names
[1]  1 2 3 4 5 6
$class
[1] "data.frame"
```

```
>print(names(df))
[1]  "x" "y"
>names(df)[1] <- "NewX"
>str(df)
'data.frame':   6 obs. of 2 variables:
$ NewX: Factor w/ 3 levels "A" "B" "C"
$ y   : num  1 2 3 4 5 6
```

Remember that the names attribute is a vector

Intro to R

Attributes of a data table (and other objects)

– The `aggregate()` function performs basic summary functions based on grouped data

```
>x <- c("A","B","A","C","C","A")
>y <- c(1,2,3,4,5,6)
>df <- data.frame(x,y)
>aggregate(y ~ x, df, sum)
  x  y
1 A 10
2 B  2
3 C  9
```

Helpful hint: if you want to calculate n for each group within x, use the 'length' function

The 'formula' format (y ~ x) basically says to perform something on the first argument 'y' (left of the tilde) by the second argument 'x' (right of the tilde). If multiple arguments are needed on the left side, use a cbind() function to list multiple columns. If multiple arguments are needed on the right side, simply use a '+' between column names.

Intro to R

Exercise 1

You want to do some analysis for consultants working on a project. Each consultant recorded planned and actual hours for this month. Bob planned 40 hours and worked 42 hours. Inez planned 50 hours and worked 54 hours. Lee planned 40 hours and worked 36 hours. Satish planned 20 hours and worked 30 hours.

1. Create vectors for the names, planned hours, actual hours, and % worked
2. Create a data frame with the vectors
3. Display the names of consultants who worked more hours than planned
4. Display how many total hours were worked

Intro to R

Exercise 1 – Step 1 Answer

You want to do some analysis for consultants working on a project. Each consultant recorded planned and actual hours for this month. Bob planned 40 hours and worked 42 hours. Inez planned 50 hours and worked 54 hours. Lee planned 40 hours and worked 36 hours. Satish planned 20 hours and worked 30 hours.

1. Create vectors for the names, planned hours, actual hours, and % worked
2. Create a data frame with the vectors
3. Display the names of consultants who worked mor
4. Display how many total hours were worked

```
>names <- c(“Bob”,”Inez”,”Lee”,”Satish”)
>plannedHours <- c(40,50,40,20)
>actualHours <- c(42,54,36,30)
>pctWorked <- actualHours / plannedHours
>print(pctWorked)
[1] 1.05 1.08 0.90 1.50
```

Intro to R

Exercise 1 – Step 2 Answer

You want to do some analysis for consultants working on a project. Each consultant recorded planned and actual hours for this month. Bob planned 40 hours and worked 42 hours. Inez planned 50 hours and worked 54 hours. Lee planned 40 hours and worked 36 hours. Satish planned 20 hours and worked 30 hours.

1. Create vectors for the names, planned hours, actual hours, and % worked
2. Create a data frame with the vectors
3. Display the names of consultants who
4. Display how many total hours were w

```
>df <- data.frame(names,plannedHours,
    actualHours,pctWorked)
>str(df)
'data.frame':  4 obs. of  4 variables:
$ names       : Factor w/ 4 levels "Bob","Inez", ...
$ plannedHours: num  40 50 40 20
$ actualHours : num  42 54 36 30
$ pctWorked   : num  1.05 1.08 0.9 1.5
```

A 'factor' data type stores each unique value and then uses an index number instead of storing the full value. This economizes memory usage.

Hewlett Packard Enterprise

Intro to R

Exercise 1 – Step 3 Answer

You want to do some analysis for consultants working on a project. Each consultant recorded planned and actual hours for this month. Bob planned 40 hours and worked 42 hours. Inez planned 50 hours and worked 54 hours. Lee planned 40 hours and worked 36 hours. Satish planned 20 hours and worked 30 hours.

1. Create vectors for the names, planned hours, actual hours, and % worked
2. Create a data frame with the vectors
3. Display the names of consultants who worked more hours than planned
4. Display how many total hours were worked

Since ‘names’ (and ‘hinames’) is a factor data type, R also reminds the user of all the values (levels) available for that factor.

```
>hinames <- df$names[df$pctWorked > 1]
>print(hinames)
[1] Bob    Inez   Satish
Levels: Bob Inez Lee Satish
```

Intro to R

Exercise 1 – Step 1 Answer

You want to do some analysis for consultants working on a project. Each consultant recorded planned and actual hours for this month. Bob planned 40 hours and worked 42 hours. Inez planned 50 hours and worked 54 hours. Lee planned 40 hours and worked 36 hours. Satish planned 20 hours and worked 30 hours.

1. Create vectors for the names, planned hours, actual hours, and % worked
2. Create a data frame with the vectors
3. Display the names of consultants who worked mor
4. Display how many total hours were worked

```
>actTotal <- sum(df$actualHours)
>print(actTotal)
[1] 162
```

Intro to R

User functions

– While it is useful to enter commands at the prompt, there is also a need to create, store, retrieve and execute a series of commands

– The `function()` method creates an object that will accept arguments and execute several lines of code

– Saved functions can be read and stored in memory using the `source()` command

Intro to R

Creating a function

– Here is a small function that will return a vector of ASCII values for each character in a string

```
str2byte <- function(szInput)
{
	bytes <- integer()
	for (i in 1:nchar(szInput))
	{
		asciival <- charToRaw(substr(szInput,i,i))
		bytes <- c(bytes,asciival)
	}
	return(bytes)
}

>str2byte(“Hello World!”)
 [1]  72 101 108 108 111  32  87 111 114 108 100  33
```

  – Note: the charToRaw() function works on strings with any length, so this is a redundant function

– Functions should be saved in .R files so they can then be shared or loaded later. For example, if we saved this function in a file called “strings.R” then the command `source(“strings.R”)` will load all functions found in that file into memory.

– An .R file can contain more than one function

Hewlett Packard Enterprise

Intro to R

Exercise 2

An easy way to check and see if a number is even or odd is to see whether or not there is a remainder when dividing by 2. In R, %/% calculates the modulo while %% calculates the remainder.

Armed with this information, write a small function that takes a vector of numbers as an argument and returns a vector of only the odd numbers in that list.

Intro to R

Exercise 2 – Answer

An easy way to check and see if a number is even or odd is to see whether or not there is a remainder when dividing by 2. In R, %/% calculates the modulo while %% calculates the remainder.

Armed with this information, write a small function that takes a vector of numbers as an argument and returns a vector of only the odd numbers in that list.

```
onlyOdds <- function(nums)
{

        remainders <- nums %% 2


        isodd <- remainders==1


        return(nums[isodd])
}
```

Create a vector of the remainder of every number in the vector divided by 2

Create a vector of TRUE/FALSE values if the remainder is 1

Apply the TRUE/FALSE values as a mask for the original list of numbers

Hewlett Packard Enterprise

Intro to R
Some helpful pointers

- Usually a Google search starting with “r programming” and your question will result in several links to official R documentation, blogs, and educational sites
  - Someone at some point has probably tried to do what you’re trying to do now
- Installing and launching the “swirl” library will give you interactive instruction
- When developing functions, test your syntax at the command line; once it returns the result you expected, add it to the function being developed in the script screen
- Try not using loops when a data transformation function will do the same work; they are much more efficient

Clean the data

Clean the data

A quick alignment on terms

– Depending on the tool, rows and columns have different names

– For this presentation, we’ll simply use ‘rows’ and columns’

Tool	Row	Column
Excel	Row	Column
Database	Record	Field
R	Observation	Variable
RapidMiner	Example	Attribute
Python	Row	Column

Hewlett Packard Enterprise

Clean the data

What does it mean to ‘clean’ data?

- Several different terms are used (cleansing, blending, transforming, etc.) but the concept is to make your data usable for exploring, analyzing and modeling

- Common activities that may happen in this stage (not all are required every time)
  - Filtering (selecting rows and/or columns to be included or excluded)
  - Grouping (defining groups or families for data values or ranges)
  - Fixing (updating values that violate properties or rules)
  - Enhancing (adding rows or columns from other sources or formulas)
  - Aggregating (adding columns that summarize attributes based on a grouping)
  - Pivoting (Using values of rows within a column to create columns)
  - Melting (Changing a set of similar columns into one new column and a descriptor)

Clean the data

Filtering data

– Performed to reduce the size of the dataset

– Performed to apply certain criteria to values within the dataset

  – This usually applies to rows

– Only filter a column if you know that you don’t need its contents for any reason

  – Even if you don’t want to display the data in a column, you may still need it to perform other actions (like a join or an aggregation)

Date	Invoice	Product	Cost	Quantity
1/21/2019	INV20143-X	Widget-Small	$12.00	1
1/21/2019	INV20144-X	Widget-Large	$25.00	4
2/4/2019	INV20157-XA	Widget-Small	$12.00	20
2/4/2019	INV20162-X	Widget-Medium	$18.00	5

We can apply a conditional filter to remove rows (for example, we can apply a filter to keep rows where the date is in February 2019)

We can apply a filter to remove columns (for example, we can filter out quantity if we know we don’t need it for the analysis)

Hewlett Packard Enterprise

Clean the data

Grouping data

– Performed to create a column that would be useful in analysis
  – Grouping should be based on a set of rules so that, if additional rows are added, the grouping can consistently be extended to those rows
– Very useful when trying to analyze free-form text

Date	Invoice	Product	Cost	Quantity
1/21/2019	INV20143-X	Widget-Small	$12.00	1
1/21/2019	INV20144-X	Widget-Large	$25.00	4
2/4/2019	INV20157-XA	Widget-Small	$12.00	20
2/4/2019	INV20162-X	Widget-Medium	$18.00	5

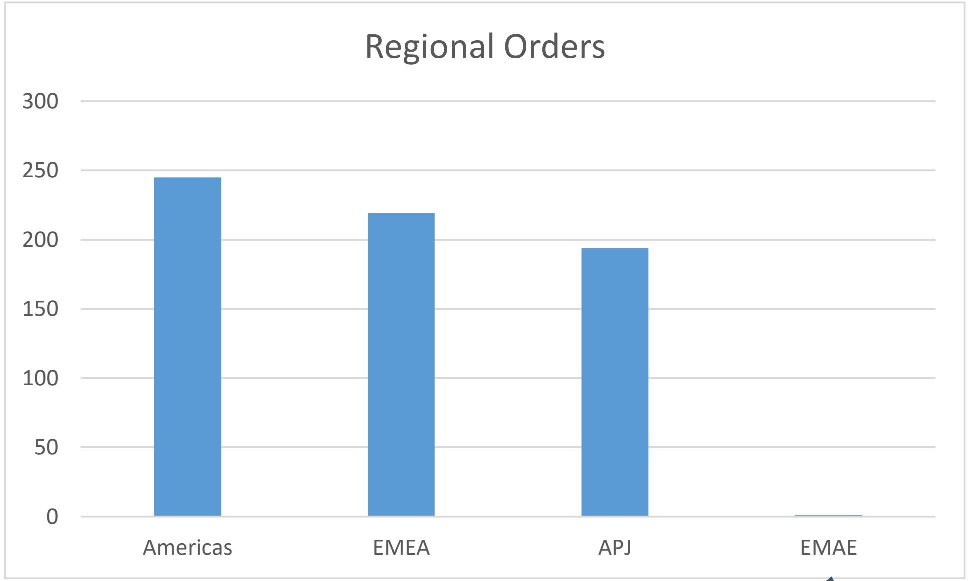
A new column called "Order Size" could give us a new grouping (small, medium, large) with rules (small = 1, medium = 2 to 10, large > 10). This is useful when we start aggregating or modeling the data.

Hewlett Packard Enterprise

Clean the data

Fixing data

- Fixing the actual value of the data
  - Usually requires investigation and manual correction, although data anomalies can sometimes be flagged using histograms (for categorical data) or identifying outliers (for numerical data)
- Fixing the format of the data
  - Text
    - Leading and trailing spaces may occur
    - Upper or lower case consistency
    - Can cause joins to fail
  - Numeric
    - Decimal places
    - Commas and decimals
  - Date formats



Visual indicator that we have an error in the data

Clean the data

Enhancing data

– Enhancing data doesn’t always require data from other sources

– Text substrings and concatenations

  – Substrings are useful if larger text values are concatenated from smaller pieces of information (like data embedded within a serial number)

  – Concatenations are useful if smaller text values can be stored in a single, larger text value (like an ID)

– Numeric

  – Converting from one base to another (like from hexadecimal to decimal)

  – Converting from one unit to another (like from miles to kilometers)

– Dates

  – Day of the week

  – Working Day (for financial processes)

  – Week of the year

  – Month of the quarter

  – Weekday or weekend

ISBN 978-3-16-148410-0

9 783161 484100

The International Standard Book Number can yield several pieces of information:

- When the number was assigned (10 digits prior to 2007, 13 digits after 2007)
- Country group
- Publisher
- Title code

While some of the pieces of information still need to be joined to a reference table to make sense of the code, splitting the ISBN into smaller pieces makes the data ready to join

Hewlett Packard Enterprise

Clean the data

Enhancing data using joins to other sources

– Very common cleaning method

– Datasets that have a common column (or set of columns) can be joined together to bring the unique columns from both sources into one larger dataset

– The type of join used is critical

– Be careful not to accidentally filter rows if the wrong type of join is used

  – An inner join has the most aggressive filtering since it only allows rows where both data sets have a value

DeptID	DeptName
1	Finance
2	Operations
3	Supply Chain
4	Marketing
5	IT

DeptID	Location
2	Wroclaw
3	Houston
5	Bangalore
6	Guadalajara
8	Dalian

Type of join	Description	Result set
Inner	Only include rows where the DeptID is found for both data sets	2 Operations Wroclaw 3 Supply Chain Houston 5 IT Bangalore
Left	Include all rows from the left data set	1 Finance (NULL) 2 Operations Wroclaw 3 Supply Chain Houston 4 Marketing (NULL) 5 IT Bangalore
Right	Include all rows from the right data set	2 Wroclaw Operations 3 Houston Supply Chain 5 Bangalore IT 6 Guadalajara (NULL) 8 Dalian (NULL)
Outer	Include all rows from both data sets	1 Finance (NULL) 2 Operations Wroclaw 3 Supply Chain Houston 4 Marketing (NULL) 5 IT Bangalore 6 (NULL) Guadalajara 8 (NULL) Dalian

Hewlett Packard Enterprise

Clean the data

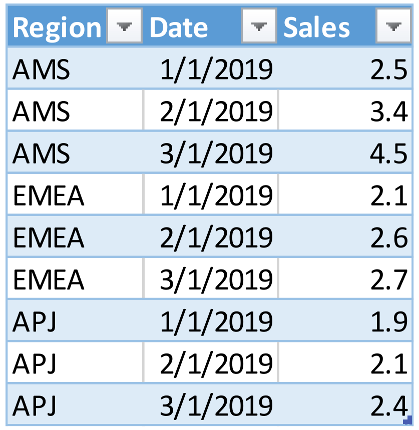
Aggregating data

- Apply common statistical functions to groups of data
  - Count() returns the number of rows
  - Sum(x) returns the sum of the values in the x column
  - Min(x) returns the smallest value in the x column
  - Max(x) returns the largest value in the x column
  - Mean(x) or Avg(x) returns the average of the values in the x column
  - Median(x) returns the median of the values in the x column
  - Var(x) returns the variance of the values in the x column
  - StDev(x) returns the standard deviation of the values in the x column
- The Count() function doesn’t require a column to be specified
- Aggregations will naturally return one value for each unique value in the grouped columns

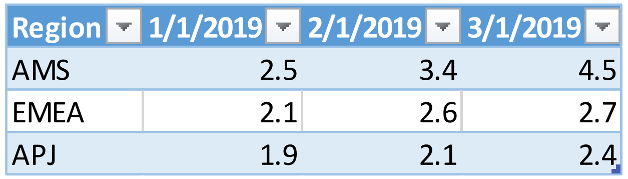
Clean the data

Pivoting data

- Transposes data by converting all of the unique values within a column so that each value gets a new column
- Useful for formatting data, but more difficult for some analysis methods
- Usually wait until all other cleaning functions have been executed before pivoting the data



Region	Date	Sales
AMS	1/1/2019	2.5
AMS	2/1/2019	3.4
AMS	3/1/2019	4.5
EMEA	1/1/2019	2.1
EMEA	2/1/2019	2.6
EMEA	3/1/2019	2.7
APJ	1/1/2019	1.9
APJ	2/1/2019	2.1
APJ	3/1/2019	2.4



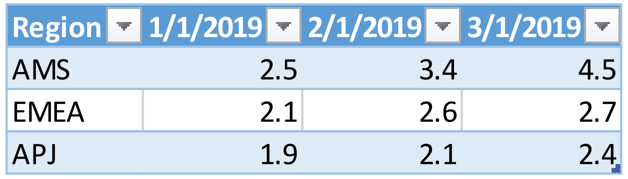
Region	1/1/2019	2/1/2019	3/1/2019
AMS	2.5	3.4	4.5
EMEA	2.1	2.6	2.7
APJ	1.9	2.1	2.4

Pivoting the Sales column using the Date column

Clean the data

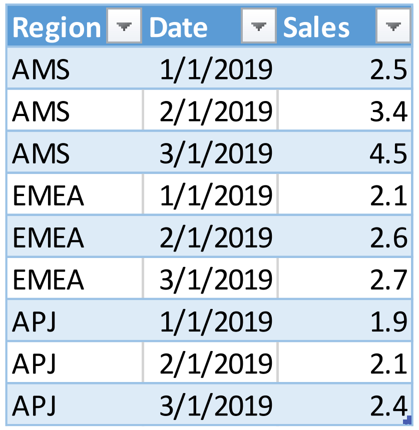
Melting data

- Think of this as de-pivoting a data set
- Transposes data by taking a set of associated columns and moving their values into a new column
- This helps make some types of analysis methods easier to process



Region	1/1/2019	2/1/2019	3/1/2019
AMS	2.5	3.4	4.5
EMEA	2.1	2.6	2.7
APJ	1.9	2.1	2.4

Melting columns 2, 3 and 4 (since they are associated) into a new column called 'Date'



Region	Date	Sales
AMS	1/1/2019	2.5
AMS	2/1/2019	3.4
AMS	3/1/2019	4.5
EMEA	1/1/2019	2.1
EMEA	2/1/2019	2.6
EMEA	3/1/2019	2.7
APJ	1/1/2019	1.9
APJ	2/1/2019	2.1
APJ	3/1/2019	2.4

Hewlett Packard Enterprise

Explore the data

Explore the data

Anscombe's quartet

– Observe these four groups of data

– What conclusions can you draw from their descriptive statistics?

– Are the four groups similar?

Group 1		Group 2		Group 3		Group 4	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

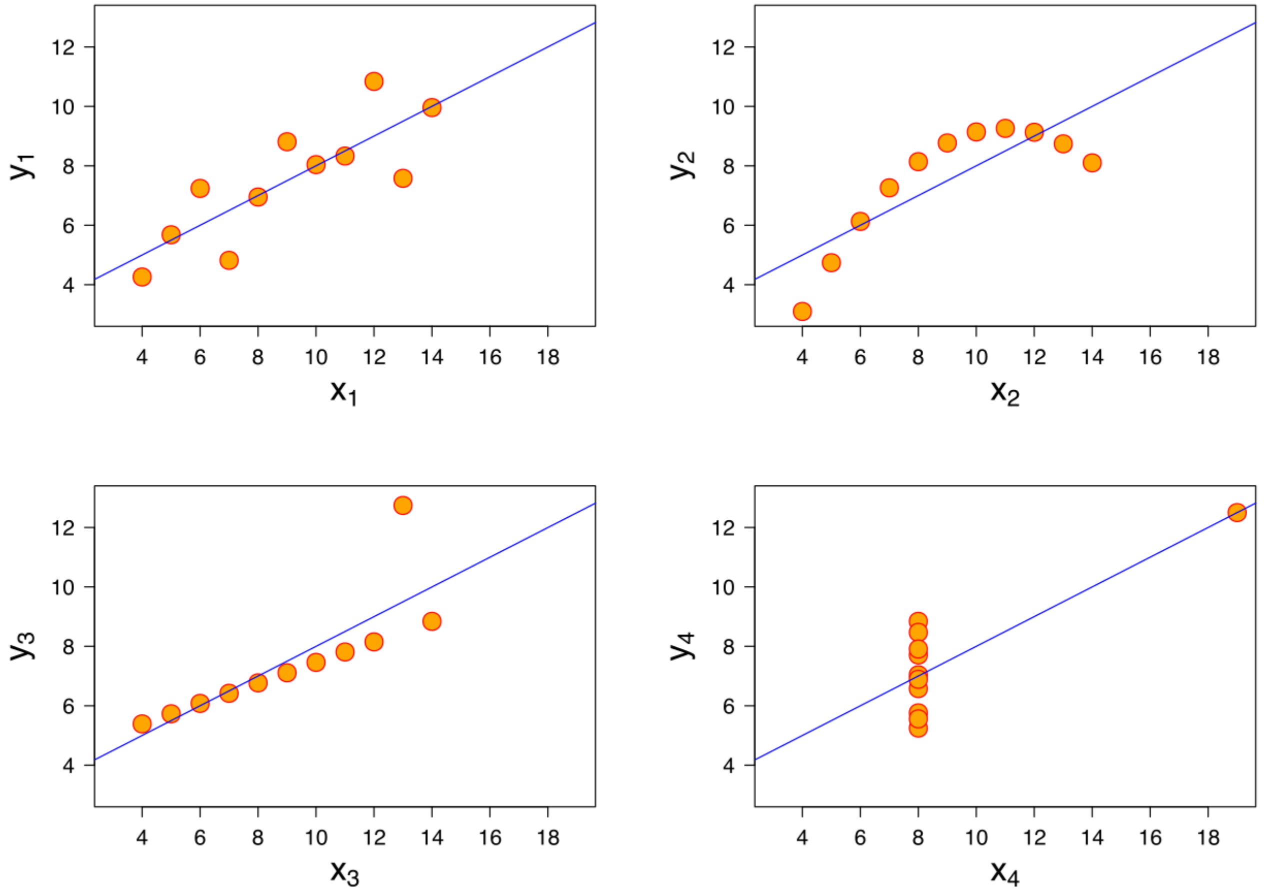
Mean of x	9.00	9.00	9.00	9.00
Sample variance of x	11.00	11.00	11.00	11.00
Mean of y	7.50	7.50	7.50	7.50
Sample variance of y	4.125	4.125	4.125	4.125
Correlation between x and y	0.816	0.816	0.816	0.816
Linear regression line	y = 3 + 0.5x	y = 3 + 0.5x	y = 3 + 0.5x	y = 3 + 0.5x

Hewlett Packard Enterprise

Explore the data

Anscombe’s quartet – in pictures

- Francis Anscombe created this quartet to demonstrate two things:
  - We should visualize the data before we begin our analysis
  - We should understand how outliers can affect the descriptive properties of the data
- While descriptive statistics are useful, it isn’t enough to just use them in exploration
  - Otherwise, we would have assumed the four groups were identical!



Hewlett Packard Enterprise

Explore the data

Common descriptive statistics

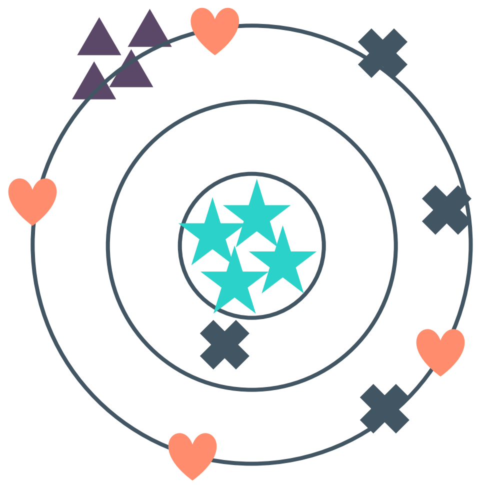
– Common descriptive statistics look at two aspects of the data: the location and the dispersion

– The location of the data describes the tendency of the data (where it tends to be located)

– The dispersion of the data describes the spread of data with relation to its location (or tendency)

– Imagine darts on a dartboard:

✖	Location is right of center, dispersion is wide
▲	Location is left of center, dispersion is narrow
♥	Location is at the center, dispersion is wide
★	Location is at the center, dispersion is narrow



Hewlett Packard Enterprise

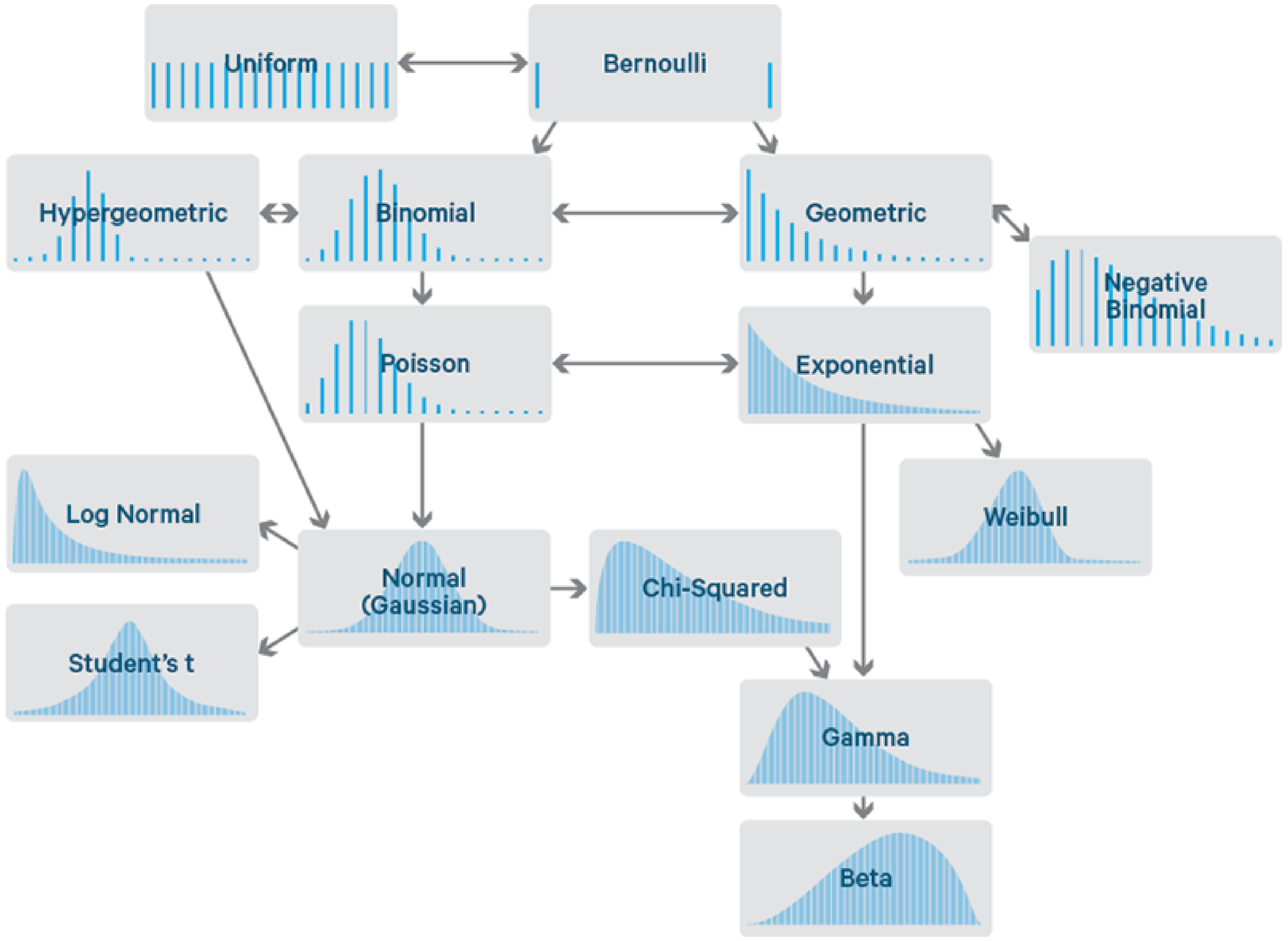
Explore the data

Common descriptive statistics

Measure	Describes	Notes
Mean	Location	• Easy to calculate • Commonly understood • Sensitive to outliers • Preferred for symmetric distribution
Median	Location	• Not as easy to calculate • Less sensitive to outliers • Preferred for skewed (asymmetric) distributions
Standard Deviation	Dispersion	• Not as easy to calculate • Relatively understood • Sensitive to outliers • Preferred for symmetric distribution
IQR (Interquartile Range)	Dispersion	• Easy to calculate • Not as easily understood • Not as sensitive to outliers • Preferred for skewed (asymmetric) distributions

Explore the data

A map of different probability distributions



Regardless of the shape of the distribution, the probabilities always add up to 1.

Hewlett Packard Enterprise

Explore the data

Common distributions and respective R functions

Distribution	Output	Example	R Density Function	R Distribution Function	R Quantile Function	R Random Generator
Uniform	All outcomes have an equal chance of occurring	Rolling a (fair) six-sided die	`dunif()`	`punif()`	`qunif()`	`runif()`
Binomial	Only one of two possible outcomes	Flipping a coin	`dbinom()`	`pbinom()`	`qbinom()`	`rbinom()`
Poisson	A discrete count of events within a given unit	Typos on a page	`dpois()`	`ppois()`	`qpois()`	`rpois()`
Exponential	The amount of time between events	Failure rates	`dexp()`	`pexp()`	`qexp()`	`rexp()`
Normal	Outcomes are balanced under a bell-shaped curve	Any independent, well-behaved distribution	`dnorm()`	`pnorm()`	`qnorm()`	`rnorm()`

Distributions

Uniform Distribution

1. Generating random numbers
   - runif(n, min, max)
   - n = number of observations
   - min = smallest value
   - max = largest value
2. Density function
   - dunif(x, min, max)

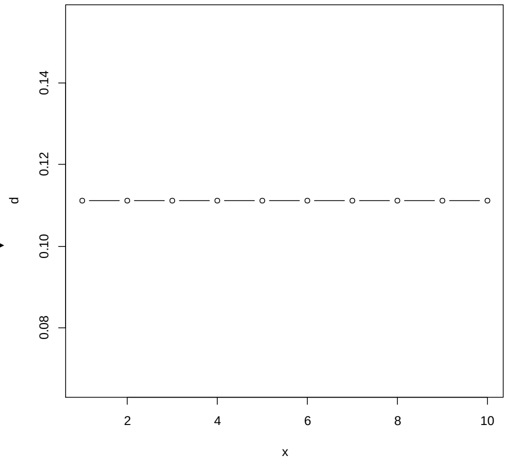
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>d <- dunif(x,1,10)
>plot(x,d,type="b")
```

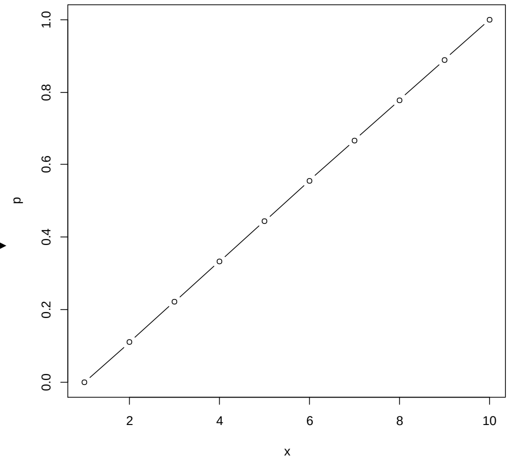
3. Distribution (cumulative density) function
   - punif(x, min, max)

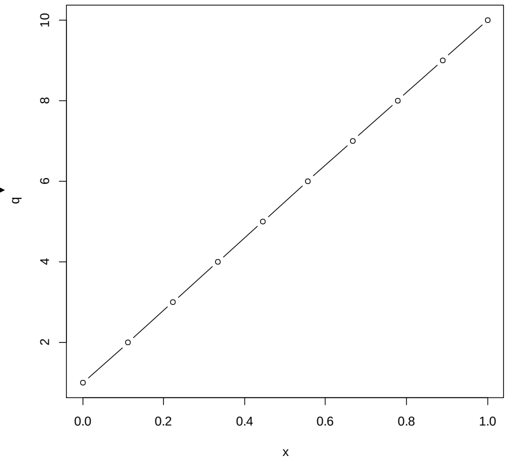
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>p <- punif(x,1,10)
>plot(x,p,type="b")
```

4. Quantile function
   - qunif(x, min, max)

```
>x <- p
>q <- qunif(x,1,10)
>plot(x,q,type="b")
```







Hewlett Packard Enterprise

Distributions

Binomial Distribution

1. Generating random numbers
   - `rbinom(n, size, prob)`
   - `n = number of observations`
   - `size = trials per observation`
   - `prob = probability of success`
2. Density function
   - `dbinom(x, size, prob, log=FALSE)`

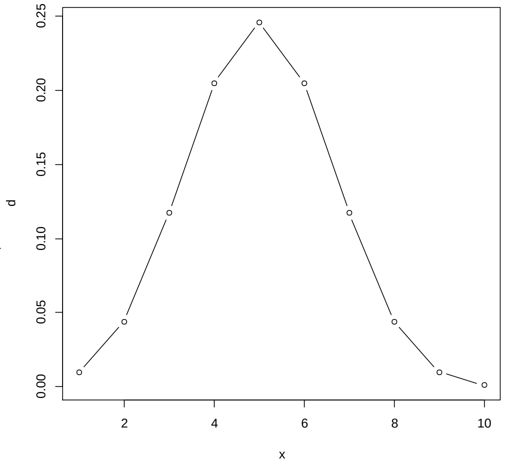
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>d <- dbinom(x,10,0.5)
>plot(x,d,type=“b”)
```

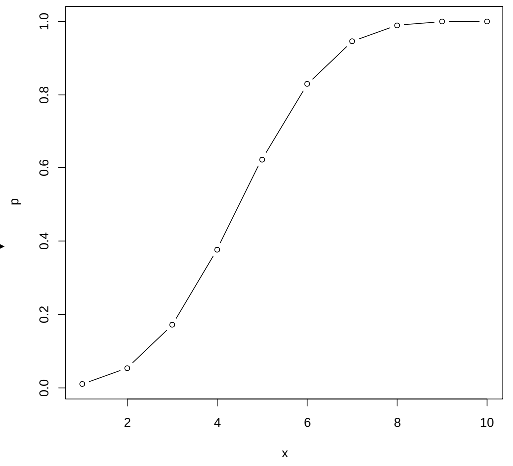
3. Distribution (cumulative density) function
   - `pbinom(x, size, prob, lower.tail=TRUE)`

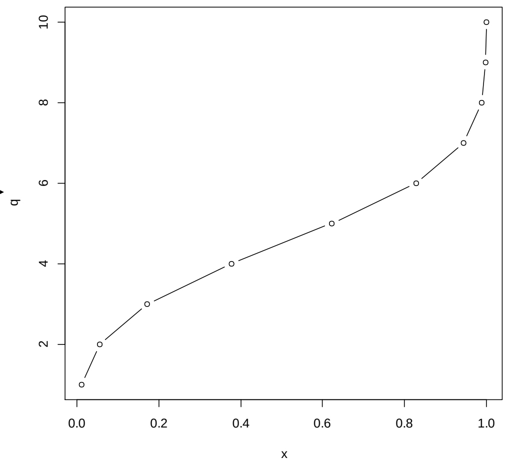
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>p <- pbinom(x,10,0.5)
>plot(x,p,type=“b”)
```

4. Quantile function
   - `qbinom(x, size, prob, lower.tail=TRUE)`

```
>x <- p
>q <- qbinom(x,10,0.5)
>plot(x,q,type=“b”)
```







Hewlett Packard Enterprise

Distributions

Poisson Distribution

1. Generating random numbers
   - `rpois(n, lambda)`
   - `n = number of observations`
   - `lambda = mean and variance`

2. Density function
   - `dpois(x, lambda, log=FALSE)`

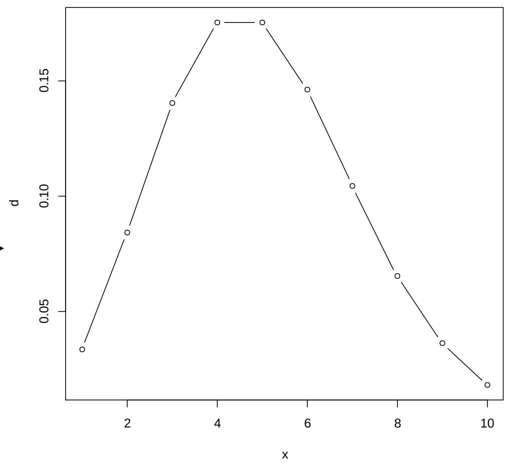
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>d <- dpois(x,5)
>plot(x,d,type=“b”)
```

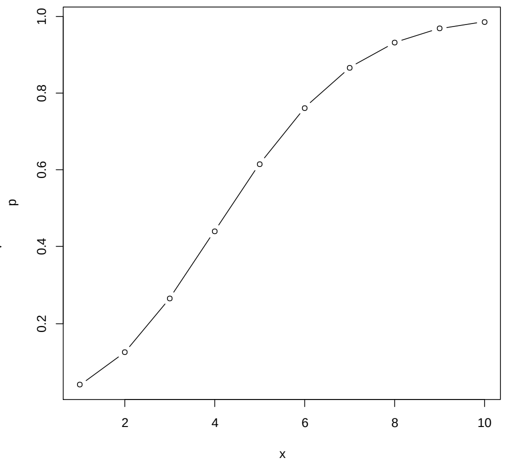
3. Distribution (cumulative density) function
   - `ppois(x, lambda, lower.tail=TRUE)`

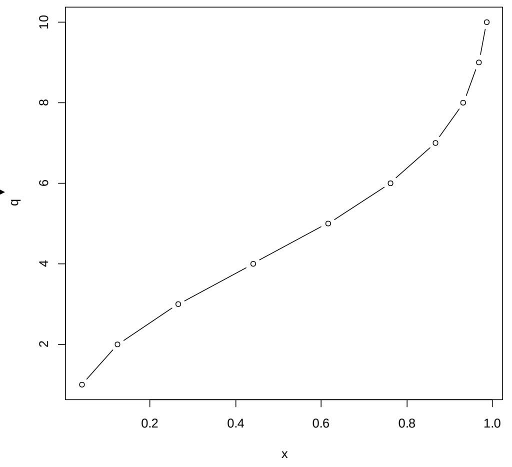
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>p <- ppois(x,5)
>plot(x,p,type=“b”)
```

4. Quantile function
   - `qpois(x, lambda, lower.tail=TRUE)`

```
>x <- p
>q <- qpois(x,5)
>plot(x,q,type=“b”)
```







Hewlett Packard Enterprise

Distributions

Exponential Distribution

1. Generating random numbers
   - `rexp(n, rate)`
   - `n = number of observations`
   - `rate = 1 / mean`

2. Density function
   - `dexp(x, rate, log=FALSE)`

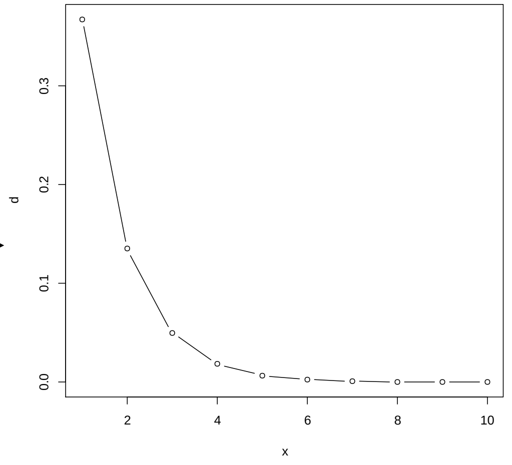
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>d <- dexp(x,1)
>plot(x,d,type=“b”)
```

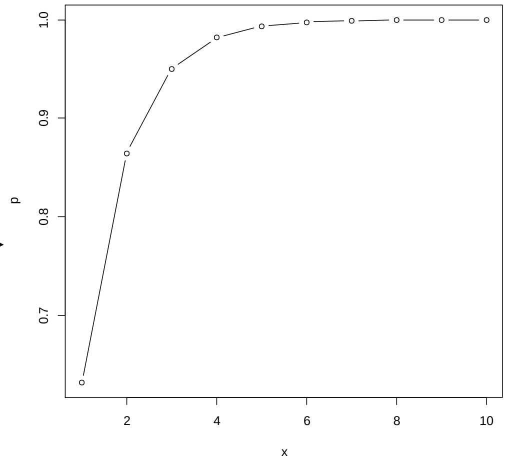
3. Distribution (cumulative density) function
   - `pexp(x, rate, lower.tail=TRUE)`

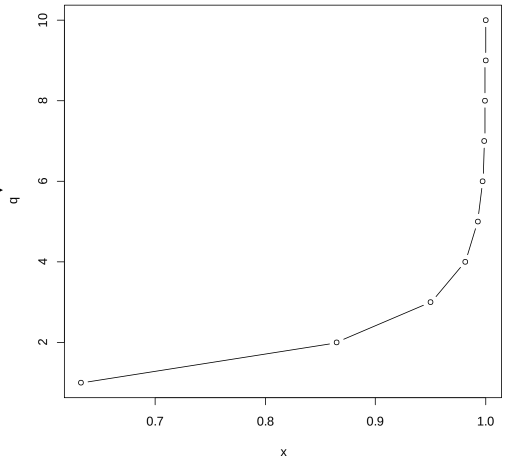
```
>x <- c(1,2,3,4,5,6,7,8,9,10)
>p <- pexp(x,1)
>plot(x,p,type=“b”)
```

4. Quantile function
   - `qexp(x, rate, lower.tail=TRUE)`

```
>x <- p
>q <- qexp(x,1)
>plot(x,q,type=“b”)
```





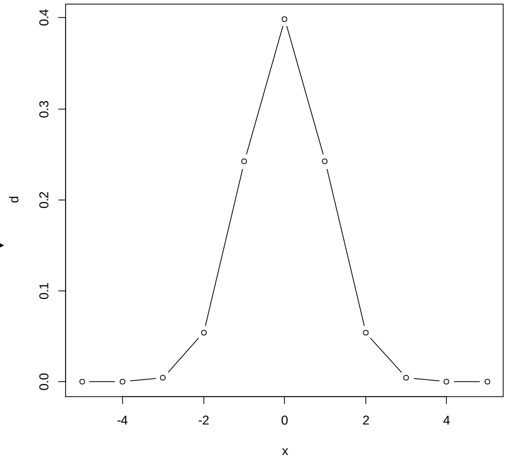


Distributions

Normal Distribution

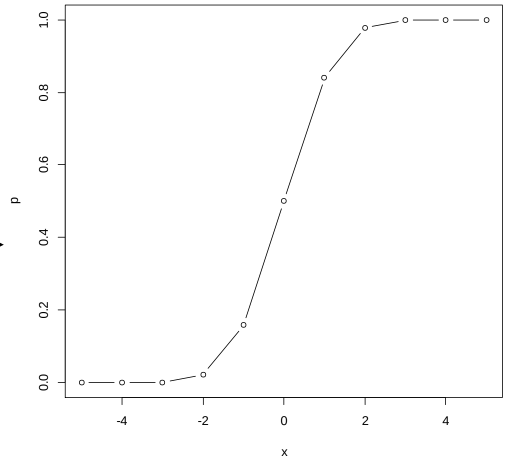
1. Generating random numbers
   - `rnorm(n, mean, sd)`
   - `n = number of observations`
   - `mean = mean`
   - `sd = standard deviation`
2. Density function
   - `dnorm(x, mean, sd, log=FALSE)`

```
>x <- c(-5,-4,-3,-2,-1,0,1,2,3,4,5)
>d <- dnorm(x,0)
>plot(x,d,type="b")
```



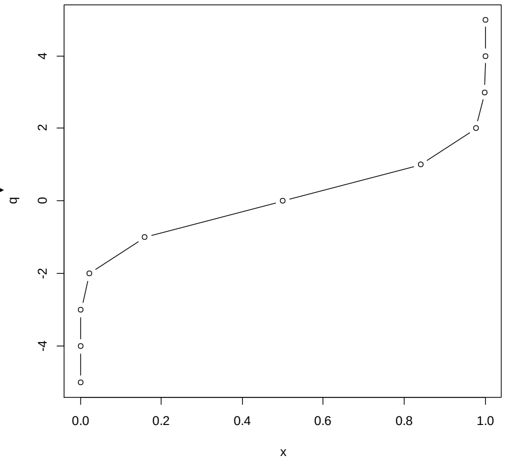
3. Distribution (cumulative density) function
   - `pnorm(x, mean, sd, lower.tail=TRUE)`

```
>x <- c(-5,-4,-3,-2,-1,0,1,2,3,4,5)
>p <- pnorm(x,0)
>plot(x,p,type="b")
```



4. Quantile function
   - `qnorm(x, mean, sd, lower.tail=TRUE)`

```
>x <- p
>q <- qnorm(x,0)
>plot(x,q,type="b")
```



Hewlett Packard Enterprise

Distributions

Exercise 1

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers and only one of them is correct. What is the probability of getting four or less correct answers if a student attempts to answer every question at random?

1. What type of probability distribution is this?
2. What is the answer?

Distributions

Exercise 1 – Step 1 Answer

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers and only one of them is correct. What is the probability of getting four or less correct answers if a student attempts to answer every question at random?

1. What type of probability distribution is this?
2. What is the answer?

Because each answer is either correct or incorrect, it is a binomial distribution

Hewlett Packard Enterprise

Distributions

Exercise 1 – Step 2 Answer

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers and only one of them is correct. What is the probability of getting four or less correct answers if a student attempts to answer every question at random?

1. What type of probability distribution is this?
2. What is the answer?

```
>pbinom(4,size=12,prob=0.2)
[1] 0.92744
```

- Use `pbinom()` since we're looking for a cumulative probability (probability of 0, 1, 2, 3, or 4 correct answers). If we just wanted to know the probability for 4 and only 4 correct answers, use `dbinom()`
- Since we want to know the probability for 4 or fewer correct answers, set the first argument to 4
- Since there are a total of 12 questions (i.e. 12 observations), set the `size` argument to 12
- Since the student is choosing each answer at random, there is a 1 in 5 probability that the correct answer will be chosen, so set the `prob` argument to 1/5, or 0.2

- The answer is that the probability of four or less questions answered correctly by random in a twelve-question multiple choice quiz is 92.7%

Hewlett Packard Enterprise

Distributions

Exercise 2

If there are twelve cars crossing a bridge per minute on average, what is the probability of having seventeen or more cars crossing the bridge in a particular minute?

1. What type of probability distribution is this?
2. What is the answer?

Distributions

Exercise 2 – Step 1 Answer

If there are twelve cars crossing a bridge per minute on average, what is the probability of having seventeen or more cars crossing the bridge in a particular minute?

1. What type of probability distribution is this?
2. What is the answer?

Because we're looking at a count, it is a Poisson distribution

Distributions

Exercise 2 – Step 2 Answer

If there are twelve cars crossing a bridge per minute on average, what is the probability of having seventeen or more cars crossing the bridge in a particular minute?

1. What type of probability distribution is this?
2. What is the answer?

```
>ppois(16,lambda=12,lower=FALSE)
[1] 0.10129
```

- Use `ppois()` since we're looking for a cumulative probability (probability of 17 or more cars). If we just wanted to know the probability for 17 and only 17 cars, use `dpois()`
- Since we want to know the probability of 17 or more cars, we want the right (upper) tail of the distribution. Set the `lower` argument to FALSE
- However, the value of x is always in the lower tail (the lower tail has all values less than or equal to x). Therefore, we set the first argument to 16 (to represent the range 0 to 16) and then take the upper tail (to represent the range 17 and above)
- Since there is an average of 12 cars per minute, set the `lambda` argument to 12

- The answer is that the probability of having 17 or more cars crossing the bridge in a particular minute is 10.1%

Distributions

Exercise 3

Suppose the mean checkout time of a supermarket cashier is three minutes. What is the probability of a customer checkout time being completed by the cashier in two minutes or less?

1. What type of probability distribution is this?
2. What is the answer?

Distributions

Exercise 3 – Step 1 Answer

Suppose the mean checkout time of a supermarket cashier is three minutes. What is the probability of a customer checkout time being completed by the cashier in two minutes or less?

1. What type of probability distribution is this?
2. What is the answer?

Because we’re looking at a rate over time, it is an exponential distribution

Distributions

Exercise 3 – Step 2 Answer

Suppose the mean checkout time of a supermarket cashier is three minutes. What is the probability of a customer checkout time being completed by the cashier in two minutes or less?

1. What type of probability distribution is this?
2. What is the answer?

```
>pexp(2,rate=1/3)
[1] 0.48658
```

- Use `pexp()` since we're looking for a cumulative probability (probability of two minutes or less). If we just wanted to know the probability right at 2 minutes (which isn't really useful in this case), use `dexp()`
- Since we want to know the rate for two minutes or less, set the first argument to 2
- Since there is an average of 1 customer every 3 minutes, set the `rate` argument to 1/3 (the rate of checkouts completed per minute)

- The answer is that the probability of finishing a checkout by the cashier in two minutes or less is 48.7%

Hewlett Packard Enterprise

Explore the data

Common visualization tools

Measure	Describes	Variations
Histogram	Distribution of the data	• A Pareto chart sorts the categories from largest to smallest frequency (assuming the categories have no predefined order)
Run chart	Movement of the data (either time-series or some other category)	• A control chart visually adds the location and dispersion of the data, and also flags anomalies (special causes)
Scatterplot	Correlation of two variables	• A linear regression line can be added to the scatterplot to show predicted values
Boxplot	Location and dispersion of the data	• Whiskers and outliers can be added to further visualize the dispersion of the data

Explore the data

Sample data in R

For these examples, we'll take advantage of some built-in data sets in R

```
>df <- mtcars
>str(df)
'data.frame':   32 obs. of 11 variables:
$ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
$ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
$ disp: num  160 160 108 258 360 ...
$ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
$ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
$ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
$ qsec: num  16.5 17 18.6 19.4 17 ...
$ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
$ am  : num  1 1 1 0 0 0 0 0 0 0 ...
$ gear: num  4 4 4 3 3 3 3 4 4 4 ...
$ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

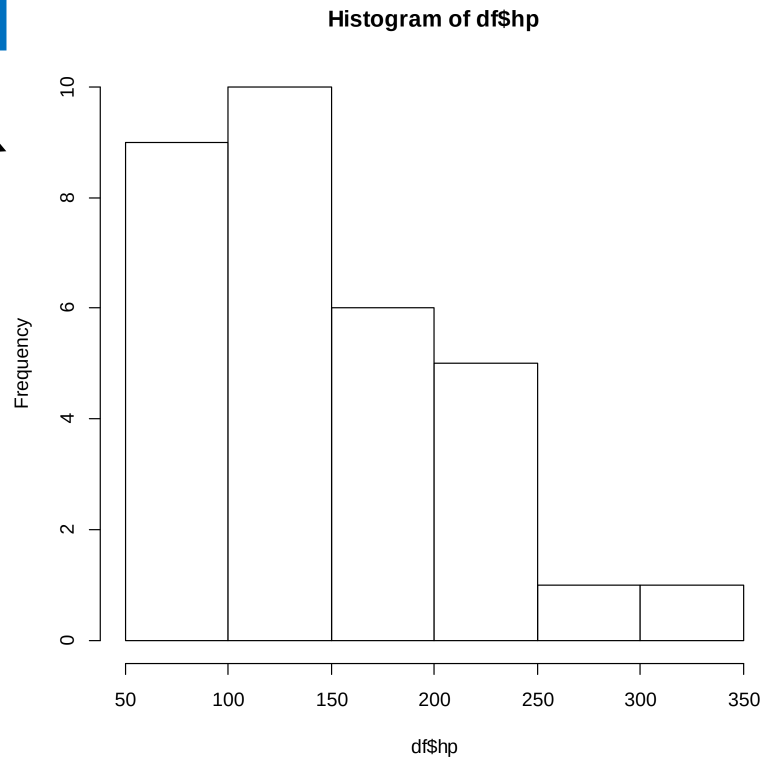
But where are the names of the cars? They're contained in the row names.

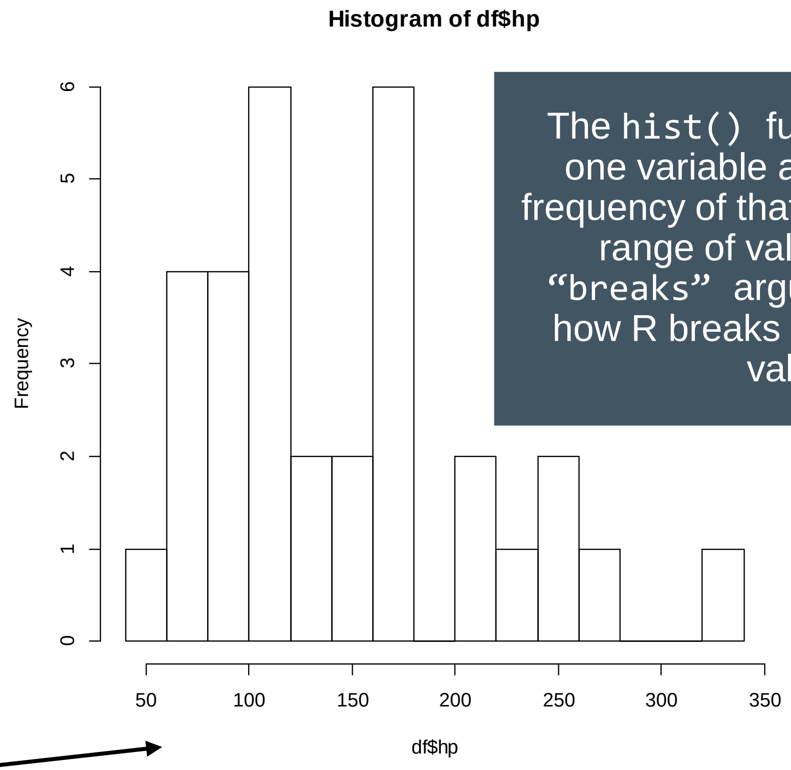
```
>row.names(df)
 [1] "Mazda RX4"           "Mazda RX4 Wag"      "Datsun 710"
 [4] "Hornet 4 Drive"      "Hornet Sportabout"  "Valiant"
 [7] "Duster 360"          "Merc 240D"          "Merc 230"
[10] "Merc 280"            "Merc 280C"          "Merc 450SE"
[13] "Merc 450SL"          "Merc 450SLC"        "Cadillac Fleetwood"
[16] "Lincoln Continental" "Chrysler Imperial"  "Fiat 128"
[19] "Honda Civic"         "Toyota Corolla"     "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"        "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"          "Porsche 914-2"
[28] "Lotus Europa"        "Fort Pantera L"     "Ferrari Dino"
[31] "Maserati Bora"       "Volvo 142E"
```

Explore the data

Histogram

```
>hist(df$hp)
```





The `hist()` function only uses one variable and displays the frequency of that variable across a range of values. Use the "`breaks`" argument to change how R breaks up the groups of values.

```
>hist(df$hp,breaks=20)
```

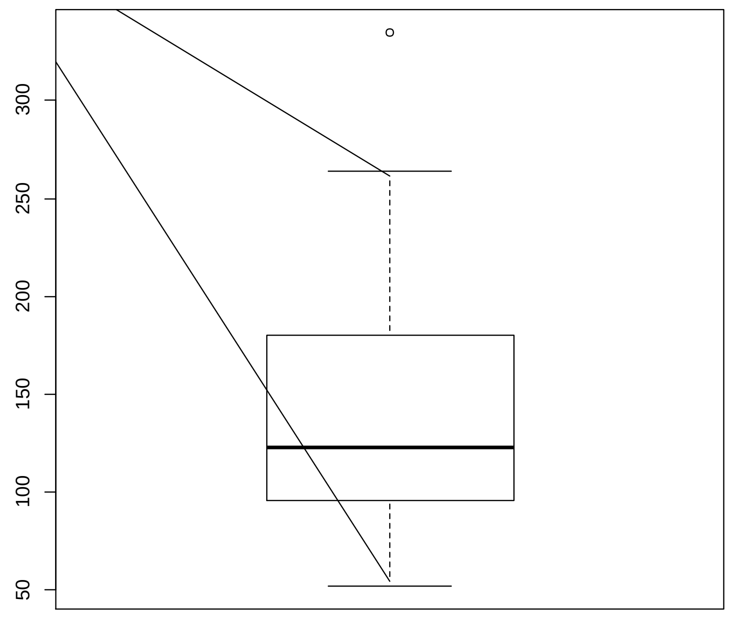
The horsepower values for the 32 cars in the dataset appear to be skewed right

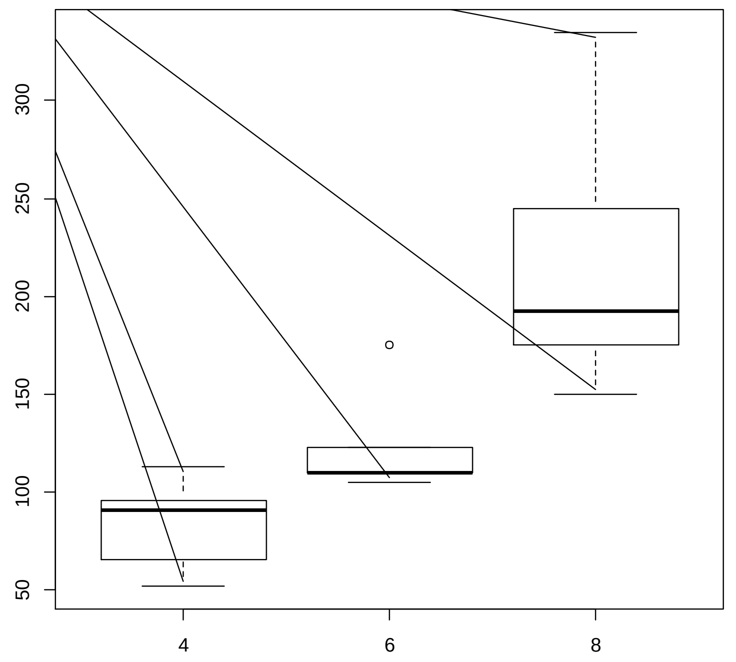
Explore the data

Boxplot

```
>boxplot(df$hp)
```

When considering the entire dataset, the Maserati Bora's horsepower is an outlier…





…but when grouped by number of cylinders, it is no longer an outlier

However, the Ferrari Dino's horsepower is an outlier when just looking at other 6-cylinder cars

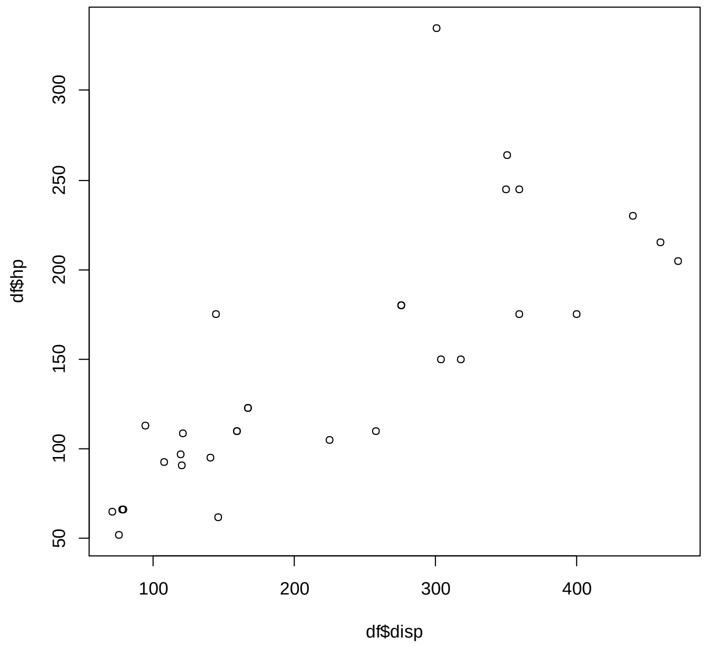
```
>boxplot(df$hp ~ df$cyl)
```

Hewlett Packard Enterprise

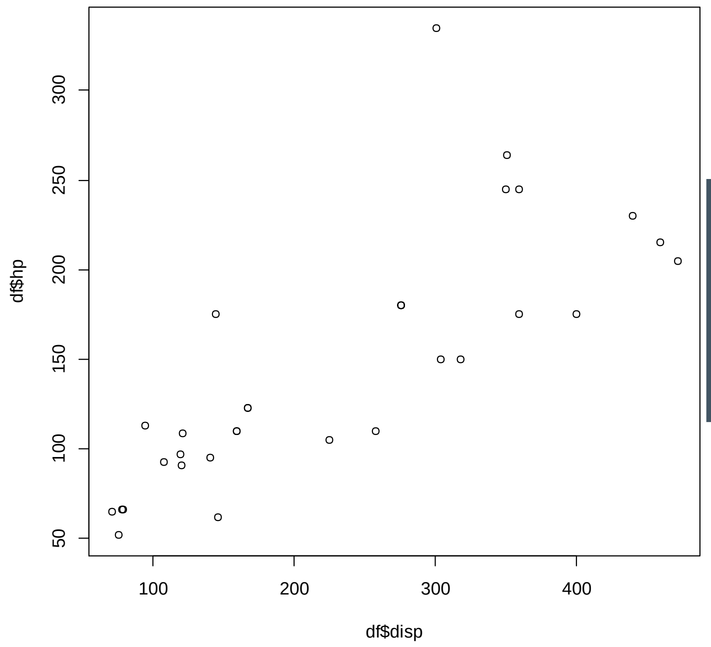
Explore the data

Scatterplot

```
>plot(df$disp,df$hp)
```



The syntax for the boxplot function allows specifying x and y separately OR using a formula…



…and the result is the same.

```
>plot(df$hp ~ df$disp)
```

If the data is ordered, the argument “`type=b`” will draw both points and lines

Explore the data

Exercise 1

Load the ‘iris’ dataset into a variable using the command df <- iris. The df data frame should have 150 observations of 5 variables.

1. Create a histogram to look at the sepal length of the “setosa” species. Does the distribution look symmetrical or skewed?
2. Create a boxplot to look at the petal lengths of the three different varieties. Which species appears to have the shortest petal length? Which species have outliers?
3. Create a scatterplot to compare the petal lengths and widths of the “versicolor” species. Does it appear that the two may be correlated?

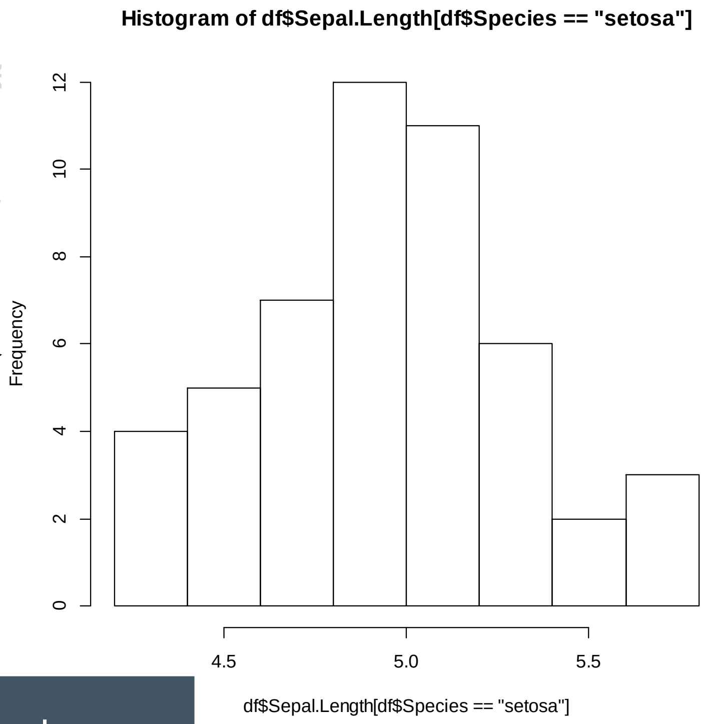
Explore the data

Exercise 1 – Step 1 Answer

Load the ‘iris’ dataset into a variable using the command df <- iris. The df data frame should have 150 observations of 5 variables.

1. Create a histogram to look at the sepal length of the “setosa” species. Does the distribution look symmetrical or skewed?
2. Create a boxplot to look at the petal lengths of the three different varie
   have the shortest petal length? Which species have outliers?
3. Create a scatterplot to compare the petal lengths and widths of the “v
   that the two may be correlated?

```
>hist(df$Sepal.Length[df$Species==“setosa”])
```



It does appear roughly symmetrical

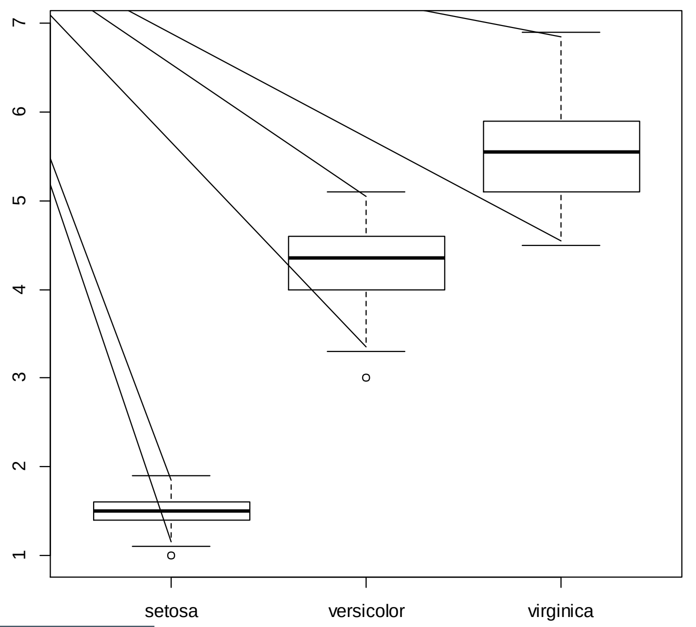
Explore the data

Exercise 1 – Step 2 Answer

Load the 'iris' dataset into a variable using the command df <- iris. The df data frame should have 150 observations of 5 variables.

1. Create a histogram to look at the sepal length of the "setosa" species. Does the distribution look symmetrical or skewed?
2. Create a boxplot to look at the petal lengths of the three different varie have the shortest petal length? Which species have outliers?
3. Create a scatterplot to compare the petal lengths and widths of the "v that the two may be correlated?

```
>boxplot(df$Petal.Length ~ df$Species)
```



The "setosa" species has the shortest petal length. Both the "versicolor" and "setosa" species have outliers in their data

Explore the data

Exercise 1 – Step 3 Answer

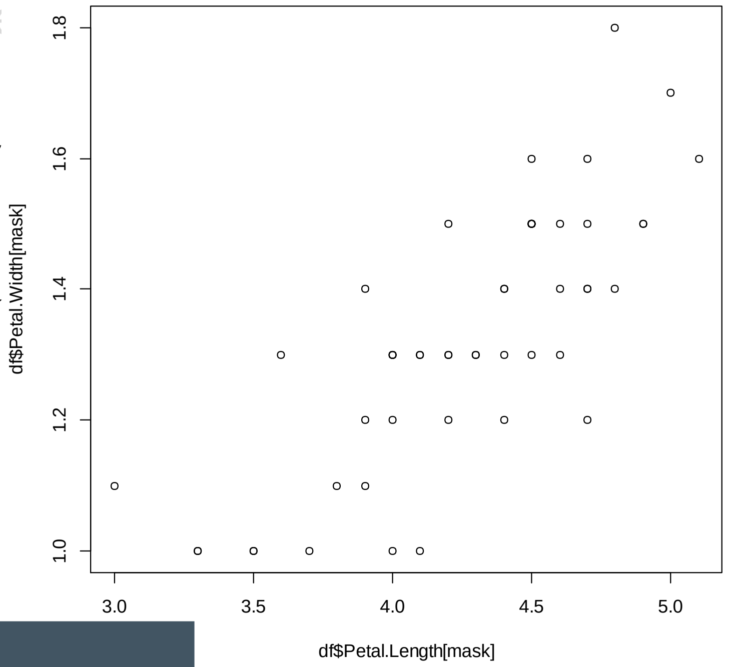
Load the ‘iris’ dataset into a variable using the command df <- iris. The df data frame should have 150 observations of 5 variables.

1. Create a histogram to look at the sepal length of the “setosa” species. Does the distribution look symmetrical or skewed?

2. Create a boxplot to look at the petal lengths of the three different varie have the shortest petal length? Which species have outliers?

3. Create a scatterplot to compare the petal lengths and widths of the “v that the two may be correlated?

```
>mask <- df$Species==“versicolor”
>plot(df$Petal.Length[mask],df$Petal.Width[mask])
```



It does appear that the two are correlated.

Explore the data

Be inquisitive

- What happens if you create subgroups in the data?
  - Create logical subgroups with similar properties (e.g. geographical location)
  - Create subgroups based on where you would predict to see differences (e.g. weekday versus weekend data)
- If outliers exist, what can you learn from it?
- If special causes exist on a control chart, what can you learn from it?
- Does the data match your predictions or expectations? If not, why not?

Use data to answer the question

Hewlett Packard Enterprise

Use data to answer the question

Statistical inference

– Statistical inference is the process of using data analysis to deduce properties of an underlying probability distribution (Oxford Dictionary of Statistics, 2008)
  – Because we can’t measure every outcome, we have to use statistical inference to describe the properties
  – We make inferences often: failure rates, satisfaction scores

– Types of statistical inference include
  – Point estimate: a value that approximates a parameter of interest
  – Interval estimate: an interval in which repeated sampling would be contained, within a certain level of confidence
  – Hypothesis testing: evidence that rejects (or fails to reject) assumed parameters
  – Clustering: classifying data points into groups

– Entire books have been written on this topic; this content will only scratch the surface

Use data to answer the question
Hypothesis testing overview

– Hypothesis testing in statistics is a way to help make inferences about our data; this is where the term ‘statistical inference’ comes from

– The null hypothesis, usually written as H_0 in formulas, asserts that the samples (or the sample with respect to its population) are statistically the same

  – It doesn’t assert that they’re exactly the same, because variation is always expected, but the variation is within acceptable limits

– The alternative hypothesis, usually written as H_A in formulas, asserts that the samples (or the sample with respect to its population) differ in some significant way

  – The difference can be that one is greater than, less than, or simply not equal to the other

– The p-value indicates which hypothesis is favored

  – If the p-value is lower than alpha (typically set to 0.05), then the null hypothesis “is rejected in favor of the alternative hypothesis”

  – If the p-value is not lower than alpha, then the null hypothesis “fails to be rejected” and is the favored hypothesis; in other words, there isn’t enough evidence to throw out the null hypothesis but it doesn’t mean that the null hypothesis is unconditionally accepted either

Use data to answer the question

Types of hypothesis testing

Data Type	Number of Samples	Test With	Sample Size	R Function

Discrete	One sample	One Proportion Z-Test	Small (n <= 30)
			Large (n > 30)
				`binom.test()`
				`prop.test()`

	Two samples	Two Proportion Z-Test	Any (but be careful to draw conclusions with n < 5)	`prop.test()`

	More than two samples	Binomial Analysis of Means	Any	`ANOM()`
				`(requires a separate package)`

Continuous	One sample	One Sample T-Test	Any (but check for normality for n < 30)	`t.test()`
	Two samples (independent)	Two Sample Unpaired T-Test	Any (but check for normality for n < 30)	`t.test()`
	Two samples (dependent)	Two Sample Paired T-Test	Any (but check for normality for n < 30)	`t.test()`
	More than two samples	Analysis of Variance (ANOVA)		`aov()`

Hewlett Packard Enterprise

Hypothesis Testing

One Proportion Z-Test

- Use when you only have one sample of discrete data
- If the sample size is small (n <= 30) then use the binom.test() function
- If the sample size is large (n > 30) then use the prop.test() function
- In both cases:
  - x is the number of successes
  - n is the number of trials
  - p is the probability to test against
  - alternative describes the alternative hypothesis (less, greater or two.sided)

```
>binom.test(x, n, p, alternative)
>prop.test(x, n, p, alternative)
```

Hewlett Packard Enterprise

Hypothesis Testing

Two Proportion Z-Test

– Use when you have two samples of discrete data

– In the function:

  – x is a vector of the number of successes

  – n is a vector of the number of trials

  – p is NULL (or simply doesn’t need to be specified)

  – alternative describes the alternative hypothesis (less, greater or two.sided)

```
>prop.test(x, n, p=NULL, alternative)
```

Hewlett Packard Enterprise

Hypothesis Testing

One Sample T-Test

– Use when you want to compare the mean of one sample with a known (or theoretical) mean

– If the sample size is < 30, a check should be performed to see if the data have a normal distribution

  – If it fails the normality test, a one-sample Wilcoxon test is preferred

– In the function:

  – x is a vector of the sample values

  – mu is the known (or theoretical) mean

  – alternative describes the alternative hypothesis (less, greater or two.sided)

```
>t.test(x, mu, alternative)
```

Hypothesis Testing

Two Sample Unpaired T-Test

- Use when you want to compare the means of two independent samples
- If the sample size is < 30, a check should be performed to see if both sets of data have a normal distribution
  - If it fails the normality test, a two-sample Wilcoxon test is preferred
- In the function:
  - x and y are vectors of the sample values for the two groups
  - alternative describes the alternative hypothesis (less, greater or two.sided)
  - var.equal is a TRUE/FALSE value to indicate if the samples have equal variance (default is FALSE)

```
>t.test(x, y, alternative, var.equal)
```

Hypothesis Testing

Two Sample Paired T-Test

- Use when you want to compare the means of two dependent samples, usually a ‘before’ and ‘after’ examination of the same group
- If the sample size is < 30, a check should be performed to see if both sets of data have a normal distribution
  - If it fails the normality test, a two-sample Wilcoxon test is preferred
- In the function:
  - x and y are vectors of the ‘before’ and ‘after’ sample values for the group
  - alternative describes the alternative hypothesis (less, greater or two.sided)
  - paired is set to TRUE

```
>t.test(x, y, alternative, paired=TRUE)
```

Hypothesis Testing

Analysis of Variance (ANOVA)

– Use when you want to compare the means of more than two independent samples

– Data should pass the following criteria:

  – Samples are obtained randomly
  – Data for each sample has a normal distribution
  – Samples have a common variance

– In the function:

  – formula describes the variables in the data frame to use (e.g. values ~ group)
  – data is a data frame with at least one variable with sample values and one variable describing the groups

```
>aov(formula, data)
```

Hewlett Packard Enterprise

Hypothesis Testing

Exercise 1

There is a population of mice containing half male and half female ($p = 0.5$). Some of these mice ($n= 160$) have developed a spontaneous cancer, 95 of which are male and 65 are female. Does the cancer affect more male than female mice?

1. What type of test should be used?
2. What is the answer?

Hypothesis Testing

Exercise 1 – Step 1 Answer

There is a population of mice containing half male and half female (p = 0.5). Some of these mice (n= 160) have developed a spontaneous cancer, 95 of which are male and 65 are female. Does the cancer affect more male than female mice?

1. What type of test should be used?
2. What is the answer?

The data is discrete and we only have one sample with size n = 160. A one proportion z-test should be used.

Hewlett Packard Enterprise

Hypothesis Testing

Exercise 1 – Step 1 Answer

There is a population of mice containing half male and half female (p = 0.5). Some of these mice (n= 160) have developed a spontaneous cancer, 95 of which are male and 65 are female. Does the cancer affect more male than female mice?

1. What type of test should be used?

2. What is the answer?

```
>result <- prop.test(x=95,
           n=160,p=0.5,
           alternative="two.sided")
>print(result)
X-squared = 5.625, df = 1, p-value = 0.01771
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.5163169 0.6667870
sample estimates:
      p
0.59375
```

- Use `prop.test()` since we have discrete data and our sample size > 30
- The null hypothesis (status quo) is that the cancer affects both male and female mice at the same rate
- Since 95 male mice had the cancer, use that number for x (as the count of successes)
- Set n equal to 160, the number of mice that were observed with the cancer
- Set p equal to 0.5, which is the probability to test against (also known as the expected probability)

- The p-value (0.01771) is less than 0.05, so we reject the null hypothesis (that it affects male and female mice at the same rate) in favor of the alternative hypothesis. The cancer does appear to affect more male than female mice.

Hewlett Packard Enterprise

Hypothesis Testing

Exercise 2

Group A has the following weights: 38.9, 61.2, 73.3, 21.8, 63.4, 64.6, 48.4, 48.8 and 48.5. Group B has the following weights: 67.8, 60.0, 63.4, 76, 89.4, 73.3, 67.3, 61.3 and 62.4. Does the average weight of group A differ significantly from group B?

1. What type of test should be used?
2. What is the answer?

Hypothesis Testing

Exercise 2 – Step 1 Answer

Group A has the following weights: 38.9, 61.2, 73.3, 21.8, 63.4, 64.6, 48.4, 48.8 and 48.5. Group B has the following weights: 67.8, 60.0, 63.4, 76, 89.4, 73.3, 67.3, 61.3 and 62.4. Does the average weight of group A differ significantly from group B?

1. What type of test should be used?
2. What is the answer?

The data is continuous and we have two samples that are independent. A two sample unpaired t-test should be used.

Hypothesis Testing

Exercise 2 – Step 1 Answer

Group A has the following weights: 38.9, 61.2, 73.3, 21.8, 63.4, 64.6, 48.4, 48.8 and 48.5. Group B has the following weights: 67.8, 60.0, 63.4, 76, 89.4, 73.3, 67.3, 61.3 and 62.4. Does the average weight of group A differ significantly from group B?

1. What type of test should be used?

2. What is the answer?

```
>a <- c(38.9,61.2,73.3,21.8,63.4,64.6,48.4,48.8,48.5)
>b <- c(67.8,60.0,63.4,76.0,89.4,73.3,67.3,61.3,62.4)
>result <- t.test(a,b,alternative="two.sided",
           var.equal=FALSE)
>print(result)
data:  a and b
t = -2.7842, df = 13.114, p-value = 0.01538
alternative hypothesis: true difference in means
95 percent confidence interval:
 -29.981920  -3.795858
sample estimates:
mean of x mean of y
 52.10000  68.98889
```

- Use `t.test()` since we have two samples of continuous data
- The null hypothesis (status quo) is that the average weight for both groups is the same
- Combine the weights of group A into vector ‘a’ and assign it to x
- Combine the weights of group B into vector ‘b’ and assign it to y
- The alternative hypothesis is that the average weights of both groups are not equal
- We don’t know if the variances are equal or not, so we’ll accept the default and set it to FALSE

- The p-value (0.01538) is less than 0.05, so we reject the null hypothesis (that the average weight is the same for both groups) in favor of the alternative hypothesis. The average weights of the two groups are statistically different.

Hewlett Packard Enterprise

Hypothesis Testing

Exercise 3

Create a data frame based on the instructions below. After constructing the data frame, run ANOVA analysis on the data to see if there are any significant differences between the average values of the three groups.

1. Create the data frame
2. Are there any significant differences between the average values?

The data frame should have two variables called “group” and “x”. There should be three different groups of ten observations, so that the total number of observations in the data frame is 30.

– Group A should have 10 normal random variables with mean 5 and standard deviation 2

– Group B should have 10 normal random variables with mean 6 and standard deviation 3

– Group C should have 10 normal random variables with mean 20 and standard deviation 4

Hypothesis Testing

Exercise 3 – Step 1 Answer

Create a data frame based on the instructions below. After constructing the data frame, run ANOVA analysis on the data to see if there are any significant differences between the average values of the three groups.

1. Create the data frame
2. Are there any significant differences between the average values?

```
>x <- c(rnorm(10,5,2),rnorm(10,6,3),rnorm(10,20,4))
>lab <- c(rep(“Group A”,10),rep(“Group B”,10),rep(“Group C”,10))
>df <- data.frame(lab,x)
>str(df)
‘data.frame’:  30 obs. of  2 variables:
 $ lab: Factor w/ 3 levels “Group A”, “Group B”,..: 1 1 1 1 1 1 1 1 1 ...
 $ x  : num  8.3 8.18 6.26 3.02 4.58 ...
```

The `rep()` function repeats a value by a specified number of times.

Since these are random, the values you get will differ.

Hypothesis Testing

Exercise 3 – Step 2 Answer

Create a data frame based on the instructions below. After constructing the data frame, run ANOVA analysis on the data to see if there are any significant differences between the average values of the three groups.

1. Create the data frame
2. Are there any significant differences between the average values?

```
>result <- aov(x ~ lab,df)
>summary(result)
            Df Sum Sq Mean Sq F value   Pr(>F)
lab          2 1374.6   687.3   55.29 2.84e-10 ***
Residuals   27  335.6    12.4
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
```

- Use `aov()` since we have more than two samples of continuous data
- The null hypothesis (status quo) is that the average for all groups is the same
- The formula `x ~ lab` means to evaluate the values in x by the labels in lab
- Simply printing the result doesn't give the detail, which is why the `summary()` function is used here

- The p-value (2.84e-10) for the variable `lab` is less than 0.05, so we reject the null hypothesis (that the average is the same for all groups) in favor of the alternative hypothesis. The average is not the same for all groups, although we need to analyze further to know which group(s) are different.

Hewlett Packard Enterprise

Use data to answer the question

Machine learning overview

- Machine learning algorithms are described as learning a target function (f) that best maps input variables (X) to an output variable (Y): (Y) = f(X)
- For the machine to 'learn,' existing data needs to be set aside
  - Training data is used to build the model
  - Testing data is used to check the accuracy of the model
  - A rough estimate is to use 70 to 80 percent of the data for training, and the remainder for testing
  - Keep the training and testing data separate (they should have no common rows)
- Choose the best learning model based on the type of target output
  - Some models are designed to predict continuous values
  - Others are designed to predict categorical values
- Only use columns that will be useful in the model
  - Using columns that have similar data (like a date value and a month value) can create confounding
  - Columns that don't have useful information can create unwanted noise
- The 'response' is the variable we want to predict
- "All models are wrong, but some are useful." – George Box

Use data to answer the question

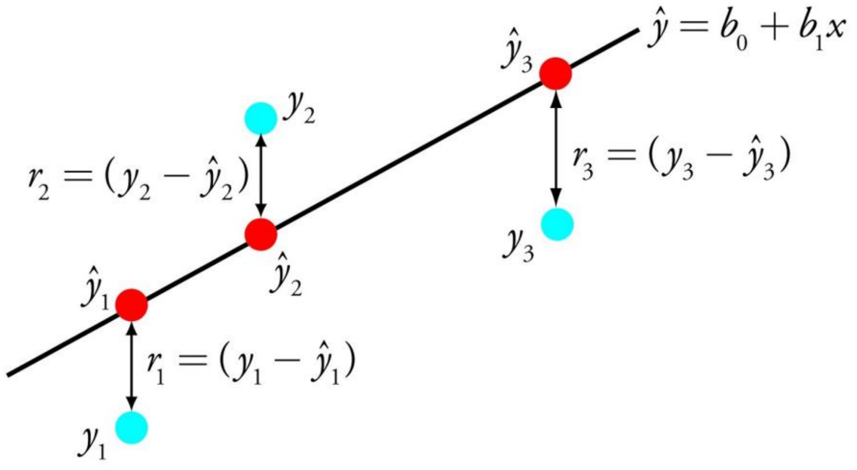
Machine learning models: linear regression

– Intended to predict continuous values

– The model is represented by an equation that best fits the input (x) and output (y) variables by finding specific weights (coefficients) for the input variables

– The best fit is determined by the amounts of the residuals – the distance between the predicted and actual values

– One of the most common models for predicting continuous values


$\hat{y} = b_0 + b_1 x$
$\hat{y}_3$
y_2
$r_2 = (y_2 - \hat{y}_2)$
$r_3 = (y_3 - \hat{y}_3)$
y_3
$\hat{y}_2$
$\hat{y}_1$
$r_1 = (y_1 - \hat{y}_1)$
y_1

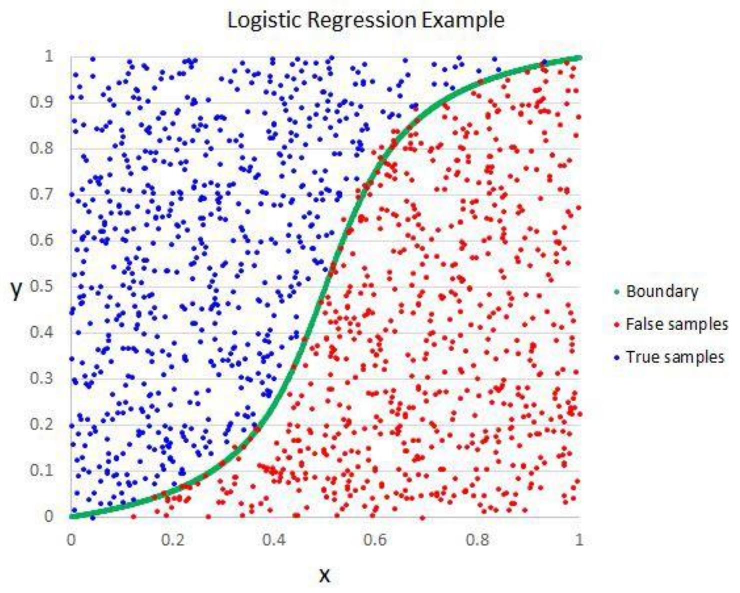
Use data to answer the question

Machine learning models: logistic regression

– Intended to predict binary classification values

– The model works similar to the linear regression model

– The difference is that the model applies an additional function to transform the output



Use data to answer the question

Machine learning models: Naïve Bayes

– Intended to predict classification values

– The model uses Bayes’ Theorem to calculate a series of probabilities:

  – The probabilities of the individual input (x) values

  – The probabilities of the input (x) values with respect to a known output (y)

– The term ‘naïve’ in the title is because it assumes that each column is independent of the other columns

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

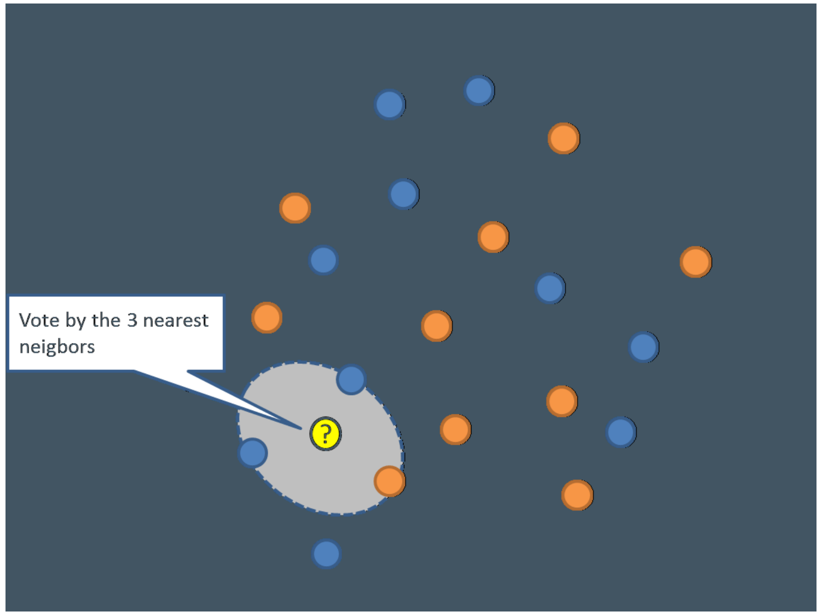
Use data to answer the question

Machine learning models: K-nearest neighbors (KNN)

– This model can be used for either continuous or categorical data, although the underlying function is based on classification

– Simply, the model predicts the output for a new input by looking at the values of its nearest neighbors

  – The number of neighbors can be adjusted to yield more accurate results

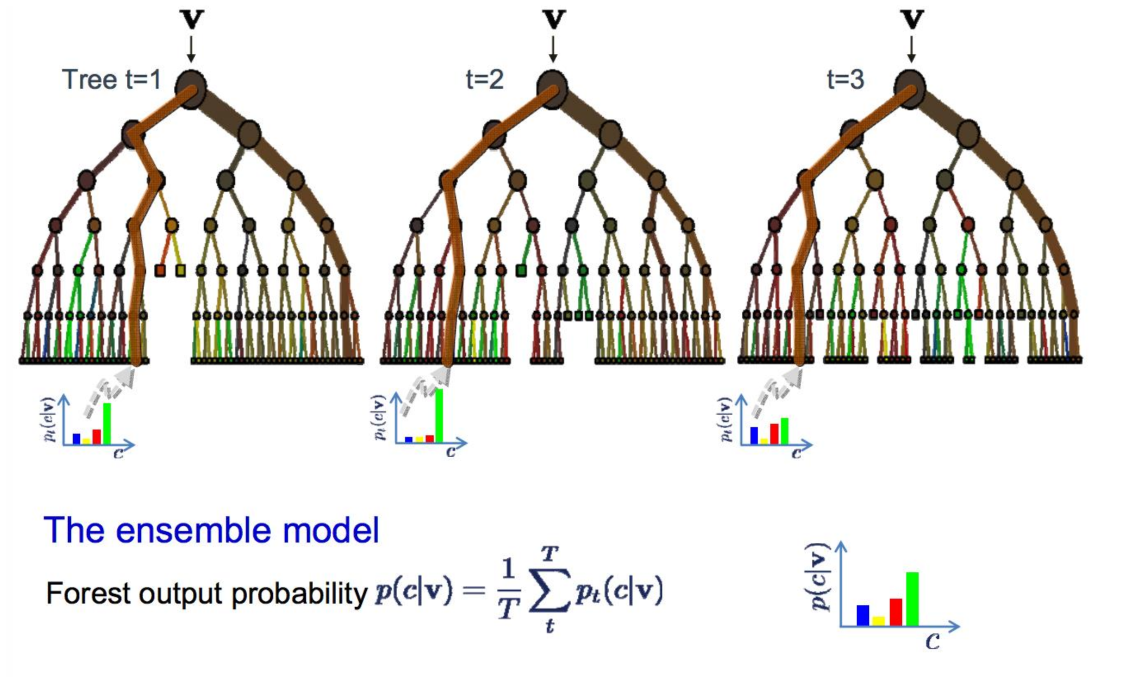


Use data to answer the question

Machine learning models: random forest

– Intended to predict categorical values

– This is a recursive model, passing through the training data several times to identify optimal branching points along with the weights that should be used at each branch

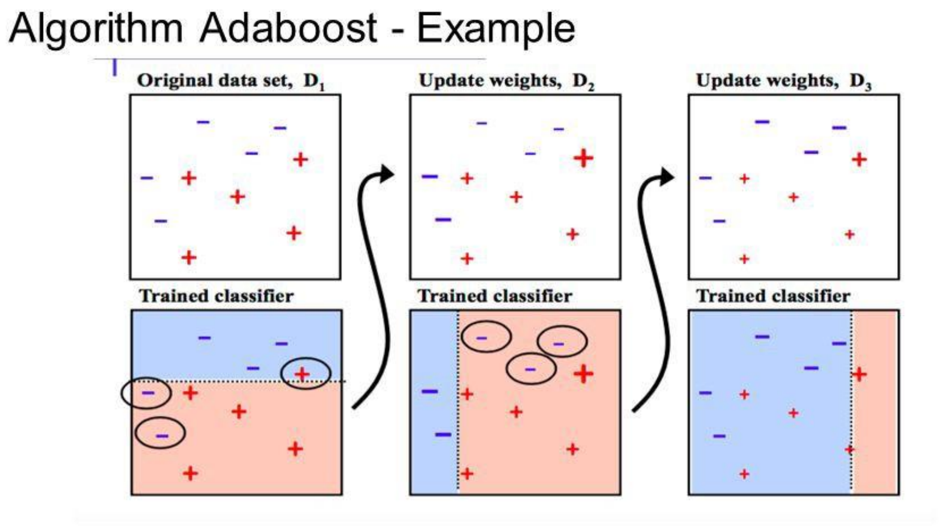

v
Tree t=1
t=2
t=3
$p_t(c|\mathbf{v})$
c
The ensemble model
Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_t^T p_t(c|\mathbf{v})$
$p(c|\mathbf{v})$

Use data to answer the question

Machine learning models: boosting

– Intended to predict categorical values

– This is also a recursive model, but in this case, the recursion is the model trying to correct errors from the previous model



courtesy to Alexander Ihler http://sli.ics.uci.edu/Classes/2012F-273a?action=download&upname=10-ensembles.pdf

Hewlett Packard Enterprise

Case Study – Linear Regression

Hewlett Packard Enterprise

Linear Regression

Overview

– Know that there are several machine learning models available in R; linear regression is simply one of the easiest ones to use

– Linear regression can also help in quickly identifying variables in data which are useful predictive factors, which in turn can aid in experimental design

Hewlett Packard Enterprise

Linear Regression

Sample data

Load up the Motor Trend cars data…

```
>df <- mtcars
>str(df)
‘data.frame’:   32 obs. of 11 variables:
$ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
$ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
$ disp: num  160 160 108 258 360 ...
$ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
$ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
$ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
$ qsec: num  16.5 17 18.6 19.4 17 ...
$ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
$ am  : num  1 1 1 0 0 0 0 0 0 0 ...
$ gear: num  4 4 4 3 3 3 3 4 4 4 ...
$ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Which factors are most useful in predicting miles per gallon (MPG)?

Linear Regression

Finding Significant Predictors – Selecting Factors

– For this example, “mpg” is the response

– Other response variables should be removed to prevent confounding. Variables ‘hp’ (horsepower) and ‘qsec’ (quarter mile time) will be excluded.

– Two variables, ‘vs’ (v-shape versus straight cylinder configuration) and ‘am’ (automatic or manual transmission) are represented with 0/1 values and could be mistaken for those being actual numerical values. The factor() function will be used on those variables to ensure they are handled properly.

Linear Regression

Finding Significant Predictors – First Pass

Run the linear model function

```
>model <- lm(mpg~cyl+disp+drat+wt+factor(vs)+factor(am)+gear+carb-1,df)
>summary(model)
Residuals:
    Min      1Q  Median      3Q     Max 
-4.4758 -1.1444 -0.2247  1.5135  5.1971 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)  
cyl         -0.684204   0.990157  -0.691   0.4965  
disp        -0.003527   0.014052  -0.251   0.8041  
drat         0.872621   1.646385   0.530   0.6012  
wt          -2.008462   1.549497  -1.296   0.2078  
factor(vs)0 28.898465  12.046643   2.399   0.0249 *
factor(vs)1 29.530255  11.572659   2.552   0.0178 *
factor(am)1  1.709254   1.993083   0.858   0.4000  
gear         0.355059   1.497221   0.237   0.8146  
carb        -0.966204   0.657199  -1.470   0.1551  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.679 on 23 degrees of freedom
Multiple R-squared:  0.9882,    Adjusted R-squared:  0.9836 
F-statistic: 214.8 on 9 and 23 DF,  p-value: < 2.2e-16
```

Each factor has an estimated coefficient, standard error, and t value. However, we’re interested in the Pr values. Remember, we want low scores here.

The variable ‘gear’ has the highest Pr value, so we’ll remove it from our list of factors and run it again…

Linear Regression

Finding Significant Predictors – Second Pass

Run the linear model function without the ‘gear’ variable

```
>model <- lm(mpg~cyl+disp+drat+wt+factor(vs)+factor(am)+carb-1,df)
>summary(model)
Residuals:
    Min      1Q  Median      3Q     Max 
-4.4930 -1.1300 -0.2971  1.4318  5.1388 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
cyl         -0.762283   0.915270  -0.833  0.41314
disp        -0.002849   0.013486  -0.211  0.83445
drat         0.900138   1.609677   0.559  0.58120
wt          -2.123744   1.442048  -1.473  0.15382
factor(vs)0 30.446406   9.923886   3.068  0.00528 **
factor(vs)1 31.097083   9.312552   3.339  0.00274 **
factor(am)1  1.880067   1.821471   1.032  0.31228
carb        -0.865774   0.492573  -1.758  0.09156 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.626 on 24 degrees of freedom
Multiple R-squared:  0.9882,    Adjusted R-squared:  0.9843
F-statistic: 251.5 on 8 and 24 DF,  p-value: < 2.2e-16
```

We’ll continue to remove the highest Pr score from the list until all variables have significantly low Pr values.

The variable ‘disp’ has the highest Pr value, so we’ll remove it from our list of factors and run it again…

Hewlett Packard Enterprise

Linear Regression

Finding Significant Predictors – Third Pass

Run the linear model function without the ‘disp’ variable

```
>model <- lm(mpg~cyl+drat+wt+factor(vs)+factor(am)+carb-1,df)
>summary(model)
Residuals:
    Min      1Q  Median      3Q     Max
-4.4763 -1.1063 -0.2238  1.4940  5.1702

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
cyl          -0.8815     0.7067  -1.247 0.223801
drat          0.8594     1.5673   0.548 0.588317
wt           -2.3414     0.9896  -2.366 0.026036 *
factor(vs)0  31.2594     8.9711   3.484 0.001836 **
factor(vs)1  31.9492     8.2321   3.881 0.000672 ***
factor(am)1   1.7956     1.7428   1.030 0.312730
carb         -0.8190     0.4316  -1.898 0.069362 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.575 on 25 degrees of freedom
Multiple R-squared:  0.9882,    Adjusted R-squared:  0.9849
F-statistic: 298.9 on 7 and 25 DF,  p-value: < 2.2e-16
```

We’ll continue to remove the highest Pr score from the list until all variables have significantly low Pr values.

The variable ‘drat’ has the highest Pr value, so we’ll remove it from our list of factors and run it again…

Hewlett Packard Enterprise

Linear Regression

Finding Significant Predictors – Fourth Pass

Run the linear model function without the ‘drat’ variable

```
>model <- lm(mpg~cyl+wt+factor(vs)+factor(am)+carb-1,df)
>summary(model)
Residuals:
    Min      1Q  Median      3Q     Max 
-4.6505 -1.1766 -0.3141  1.5317  5.2570 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
cyl          -1.0174     0.6528  -1.559   0.1312    
wt           -2.4587     0.9531  -2.580   0.0159 *  
factor(vs)0  35.2625     5.1434   6.856 2.81e-07 ***
factor(vs)1  35.9398     3.7957   9.468 6.53e-10 ***
factor(am)1   2.0262     1.6684   1.214   0.2355    
carb         -0.7416     0.4024  -1.843   0.0767 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.54 on 26 degrees of freedom
Multiple R-squared:  0.9881,    Adjusted R-squared:  0.9853 
F-statistic: 358.3 on 6 and 26 DF,  p-value: < 2.2e-16
```

We’ll continue to remove the highest Pr score from the list until all variables have significantly low Pr values.

The variable ‘am’ has the highest Pr value, so we’ll remove it from our list of factors and run it again…

Hewlett Packard Enterprise

Linear Regression

Finding Significant Predictors – Fifth Pass

Run the linear model function without the ‘am’ variable

```
>model <- lm(mpg~cyl+wt+factor(vs)+carb-1,df)
>summary(model)
Residuals:
   Min     1Q Median     3Q    Max
-4.670 -1.558 -0.457  1.246  5.752

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
cyl          -1.3357     0.6032  -2.214 0.035428 *
wt           -3.1390     0.7779  -4.035 0.000403 ***
factor(vs)0  39.9347     3.4437  11.596 5.41e-12 ***
factor(vs)1  39.7482     2.1574  18.424  < 2e-16 ***
carb         -0.4975     0.3516  -1.415 0.168522
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.563 on 27 degrees of freedom
Multiple R-squared:  0.9874,    Adjusted R-squared:  0.985
F-statistic: 422.2 on 5 and 27 DF,  p-value: < 2.2e-16
```

We’ll continue to remove the highest Pr score from the list until all variables have significantly low Pr values.

The variable ‘carb’ has the highest Pr value, so we’ll remove it from our list of factors and run it again…

Hewlett Packard Enterprise

Linear Regression

Finding Significant Predictors – Sixth Pass

Run the linear model function without the ‘carb’ variable

```
>model <- lm(mpg~cyl+wt+factor(vs)-1,df)
>summary(model)
Residuals:
    Min      1Q  Median      3Q     Max
-4.3115 -1.7450 -0.3759  1.5174  6.0433

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
cyl          -1.3641     0.6135  -2.223 0.034433 *
wt           -3.2464     0.7879  -4.120 0.000304 ***
factor(vs)0  38.7461     3.3989  11.399 4.95e-12 ***
factor(vs)1  39.2702     2.1686  18.109  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.608 on 28 degrees of freedom
Multiple R-squared:  0.9864,    Adjusted R-squared:  0.9845
F-statistic: 509.1 on 4 and 28 DF,  p-value: < 2.2e-16
```

We now have the list of significant factors for our predictive model.

To verify, check the results by creating some exploratory charts.

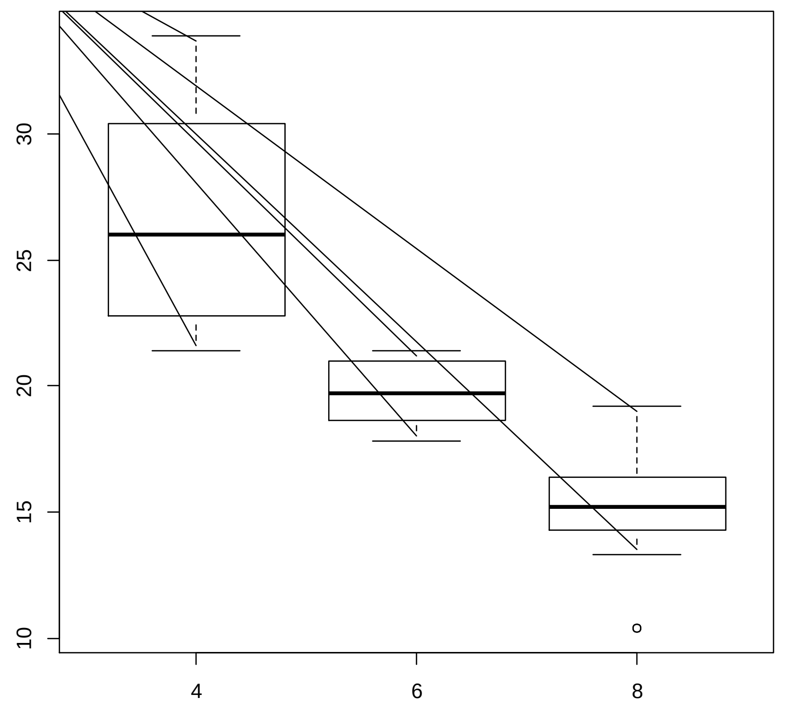
Hewlett Packard Enterprise

Linear Regression

Finding Significant Predictors – Checking the ‘cyl’ variable

How do the MPG values look when grouped by number of cylinders?

```
>boxplot(df$mpg ~ df$cyl)
```

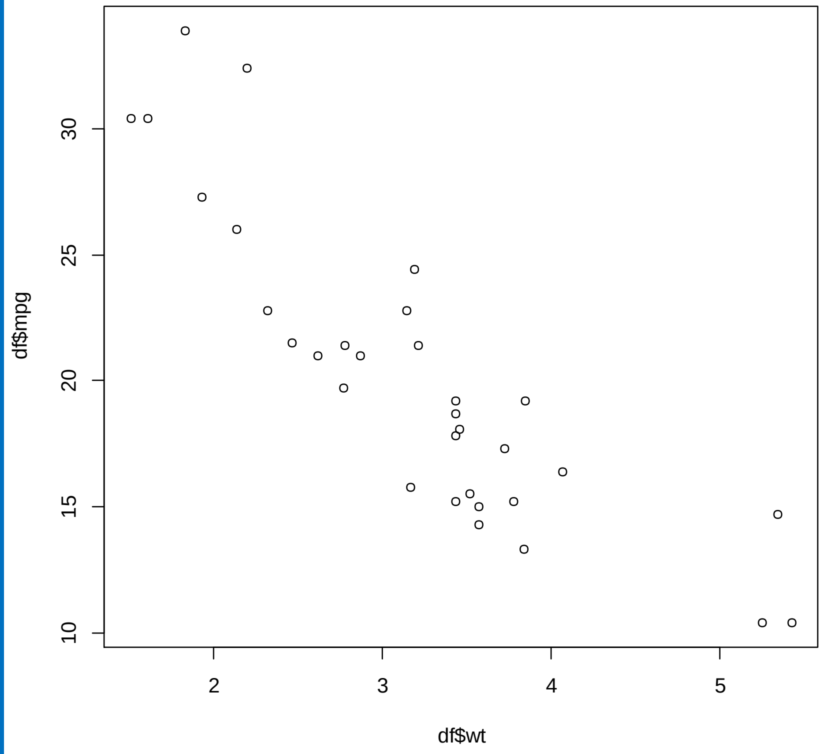


Linear Regression

Finding Significant Predictors – Checking the ‘wt’ variable

How do the MPG values look when compared to the weight?

```
>plot(df$wt,df$mpg)
```

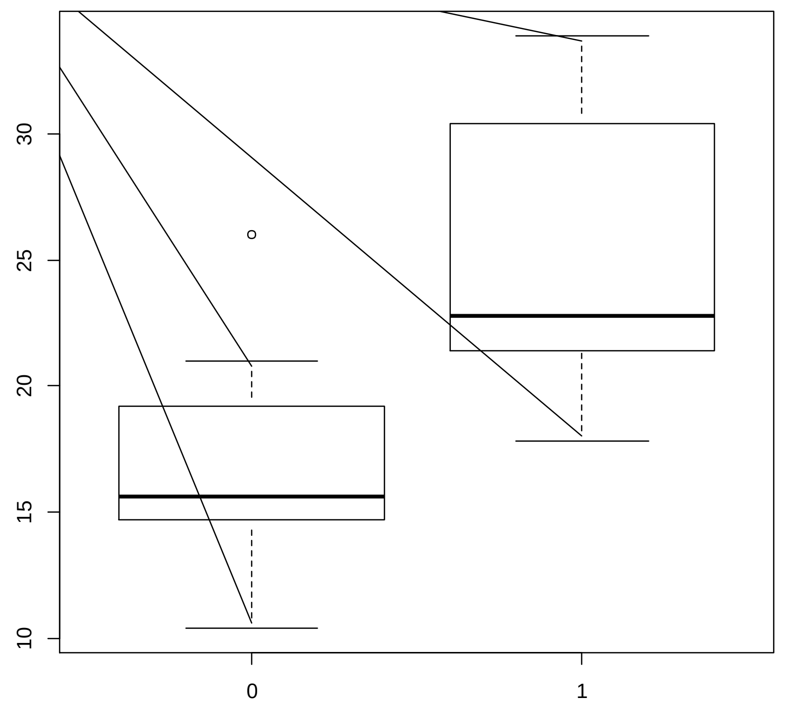


Linear Regression

Finding Significant Predictors – Checking the ‘vs’ variable

How do the MPG values look when compared to the cylinder configuration (v-shaped versus straight)?

```
>boxplot(df$mpg ~ df$vs)
```



Linear Regression

Finding Significant Predictors – Summary

- Three variables in the dataset are best at predicting miles per gallon: the weight of the car, the number of cylinders, and the cylinder configuration.
  - Cars with fewer cylinders have a better MPG rating than cars with more cylinders
  - Cars with a lower weight have a better MPG rating than cars with a higher weight
  - Cars with a straight cylinder configuration (in the data, 1 represented straight) have a better MPG rating than cars with a V-shaped configuration
- R saves the output of the `lm()` function as a predictive model, which can then be used on future data sets using the `predict()` function with the model and the new data as arguments

Case Study – Machine Learning

Hewlett Packard Enterprise

Machine Learning

Overview

– On April 14, 1912, disaster struck the RMS Titanic when it hit an iceberg; within a matter of hours the entire ship sank

– Data of the ship’s manifest for that maiden voyage exists

– Can we use the data from the manifest to predict which passengers survived the disaster?

Machine Learning

Remember the steps of data analytics

Analytic Step	Response
Define what we want to know	What factors, if any, from the passenger manifest are useful in predicting whether or not someone survived?
Define what data we need	Since we are limiting our analysis to the passenger manifest, the manifest is all we need
Gather the data	Data is available in Titanic.csv

Machine Learning

The ‘caret’ package

– Before we can create the model, ensure that the package ‘caret’ has been installed and loaded

– Installing a package only has to be done once, but loading a package has to be done every time an R session is loaded

```
>install.packages(“caret”)

>library(caret)
```

- The `caret` package has several machine learning models available; the convenience of using this package is that the primary functions (train, predict) and their arguments are standardized across several models

Hewlett Packard Enterprise

Machine Learning

Structure of the Titanic dataset

– Load the data into R and make some general observations

```
>dfTitanic <- read.csv("Titanic.csv",as.is=TRUE)
>dim(dfTitanic)
[1] 1310   12
>str(dfTitanic)
'data.frame':   1310 obs. of  12 variables:
$ Passenger.Class                : chr  "First" "First" "First" ...
$ Name                           : chr  "Allen, Miss. Elisabet" ...
$ Sex                            : chr  "Female" "Male" "Femal" ...
$ Age                            : num  29 0.917 2 30 25 ...
$ No.of.Siblings.or.Spouses.on.B : int  0 1 1 1 1 0 1 0 2 0 ...
$ No.of.Parents.or.Children.on.B : int  0 2 2 2 2 0 0 0 0 0 ...
$ Ticket.Number                  : chr  "24160" "113781" "1137" ...
$ Passenger.Fare                 : num  211 152 152 152 152 ...
$ Cabin                          : chr  "B5" "C22 C26" "C22 C2" ...
$ Port.of.Embarkation            : chr  "Southampton" "Southam" ...
$ Life.Boat                      : chr  "2" "11" "" ""  ...
$ Survived                       : chr  "Yes" "Yes" "No" "No" ...
```

- The `as.is=TRUE` argument prevents R from converting character values to factors; while factoring is useful from a memory management standpoint (i.e. data compression) it can sometimes cause problems with predictive models if the model thinks the factor value is meaningful

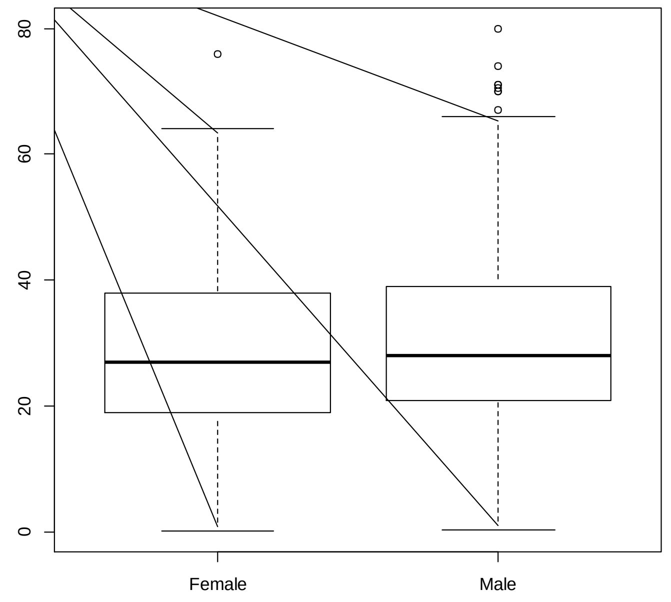
Hewlett Packard Enterprise

Machine Learning

Visualizing the Titanic data

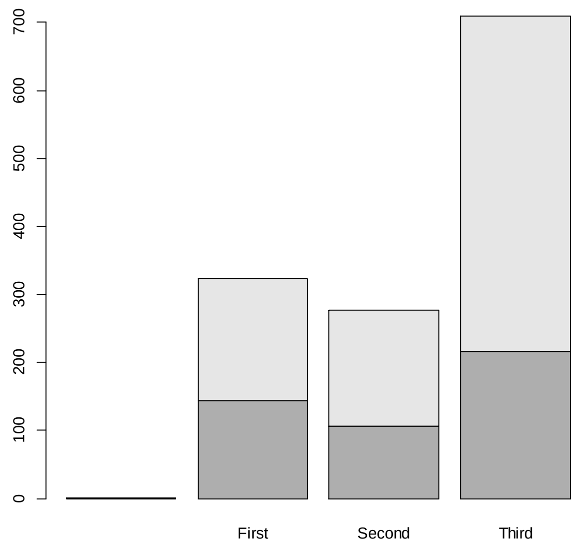
– Load the data into R and make some general observations

```
>boxplot(dfTitanic$Age ~ dfTitanic$Sex)
```



- The median and IQR for the ages look reasonable; there are a lot of outliers for male passengers

```
>barplot(table(dfTitanic$Sex,dfTitanic$Passenger.Class))
```



- First and second class seem to have reasonable proportions; there are many more male than female passengers in third class. Families were probably more likely to travel in first or second class
- It looks like there is a record that doesn't have class defined; we'll need to filter it out

Machine Learning

Continuing down the data analytics path

Analytic Step	Response
Define what we want to know	What factors, if any, from the passenger manifest are useful in predicting whether or not someone survived?
Define what data we need	Since we are limiting our analysis to the passenger manifest, the manifest is all we need
Gather the data	Data is available in Titanic.csv
Clean the data	• Remove the ‘lifeboat’ column since it could confound the ‘survived’ column • Remove the ‘cabin’ column since it could confound the ‘passenger class’ column • Remove columns that are unique identifiers (names, ticket numbers) • Filter the records that don’t have passenger class or point of embarkation information

Machine Learning

Cleaning the data

– Primarily, we need to use filtering functions to clean the data

```
>dfTitanic <- dfTitanic[dfTitanic$Passenger.Class!="",]

>dfTitanic <- dfTitanic[dfTitanic$Port.of.Embarkation!="",]

>dfTitanic$Life.Boat <- NULL

>dfTitanic$Cabin <- NULL

>dfTitanic$Name <- NULL

>dfTitanic$Ticket.Number <- NULL

>dim(dfTitanic)
[1] 1307    8
```

- We won't use the `is.na()` function because the missing passenger class and point of embarkation values don't have N/A values, they just have blank values
- Setting a column to NULL is the equivalent of removing it from the data frame. There are other techniques that can be used to do the same thing; this technique is the simplest

Hewlett Packard Enterprise

Machine Learning

Handling N/A values during cleaning

– In exploring the data, there is another problem: there are several rows where the age has an N/A value, and one row where the passenger fare is N/A

  – These need to either be filtered or imputed to be used in the model

  – For simplicity, just replace missing values with the median values for age and passenger fare, but note that there are several, better methods available for imputing values

```
>median(dfTitanic$Age)
[1] NA

>dfTitanic$Age[is.na(dfTitanic$Age)] <- median(dfTitanic$Age,na.rm=TRUE)
>median(dfTitanic$Age)
[1] 28

>median(dfTitanic$Passenger.Fare)
[1] NA

>dfTitanic$Passenger.Fare[is.na(dfTitanic$Passenger.Fare)] <-
median(dfTitanic$Passenger.Fare,na.rm=TRUE)
>median(dfTitanic$Pasenger.Fare)
[1] 14.4542
```

Include `na.rm=TRUE` or else we'll be in a cycle of replacing NA values with NA values! How? Functions like `median()` will return an NA value if any of the vector values are NA.

Hewlett Packard Enterprise

Machine Learning

Training and testing datasets

– Since we explored and cleaned data in parallel, it is time to start building the predictive model

– The first step is to create two datasets: a training dataset (used to create the mode) and a test dataset (used to test the accuracy of the model)

```
>set.seed(1138)

>train <- sample(nrow(dfTitanic),0.7*nrow(dfTitanic),replace=FALSE)

>head(train)
[1] 358 217 355 779 872 624

>dfTrain <- dfTitanic[train,]
>dfTest <- dfTitanic[-train,]
```

- A quick overview of random number generators: they're not truly random, but a seemingly random sequence based on a seed value. If the same seed is used, the same 'random' numbers will appear. For others to replicate work, seeds should be set to generate the same set of 'random' numbers

- The `sample()` function takes three arguments
  - The vector of data from which to gather samples
  - The size of the sample
  - Whether or not samples should be put back into the sampling pool or not

- The `train` vector is simply holding row numbers

- It's a standard practice to not use the same data that created the model to test the results, so we want to break our dataset into two subsets – one to train the model and one to test the model

Hewlett Packard Enterprise

Machine Learning

Building the model

– Build the model and review the results

```
>modTree <- train(Survived ~ .,data=dfTrain,method=“rf”)

>modTree
Random Forest

914 samples
  7 predictor
  2 classes: ‘No’, ‘Yes’

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 914, 914, 914, 914, 914, 914, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.7928679  0.5326842
  5     0.7545996  0.5106645
  9     0.7634578  0.4899738

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

- The train() function has three basic arguments
  - The formula for the model; remember that the tilde represents the word “by” so in this case, the formula is the Survived variable (the response) by all other variables (the predictors)
  - The dataset (in this case, our training dataset)
  - The training method to use (there are a lot of different values for this function; “rf” stands for random forest)

Machine Learning

Investigating the model

– Understand a bit more about this model by using `varImp` (importance of the variables)

```
>varImp(modTree)
rf variable importance

Overall
SexMale                            100.000
Passenger.Fare                      58.173
Age                                 38.297
Passenger.ClassThird                23.889
No.of.Parents.or.Children.on.Board  13.221
No.of.Siblings.or.Spouses.on.Board  12.960
Port.of.EmbarkationSouthampton       5.745
Passenger.ClassSecond                5.052
Port.of.EmbarkationQueenstown        0.000
```

- The importance of the variables shows us which variables were most important in building the tree
- Port of embarkation doesn't seem to have much importance; we can rerun the model without that variable to see if accuracy improves
- Also, remember that we had a large amount of ages that were missing, so it would also be interesting to see the results without including the age variable
- Knowing what we know about disaster procedures (like rescuing women and children first), the variables most important to making the prediction shouldn't be too much of a surprise

Hewlett Packard
Enterprise

Machine Learning

Testing the model

– Now that we have a predictive model, we can send data into it and check the results

– In this case, we'll send the data we set aside (the test set) and then compare predicted and actual values

  – Note that we only need to specify the test data frame; the model already knows which variables are predictors and responses

  – Use a confusion matrix to compare two values: the predicted values and the actual values

- Sensitivity is the number of correct positive predictions divided by the total number of positive values, also known as the true positive rate (TPR)
- Specificity is the number of correct negative predictions divided by the total number of negative values, also known as the true negative rate (TNR)
- A "false positive" is when the predicted value was positive but the actual value was negative (the model falsely identified it as positive)
- A "false negative" is when the predicted value was negative but the actual value was positive (the model falsely identified it as negative)

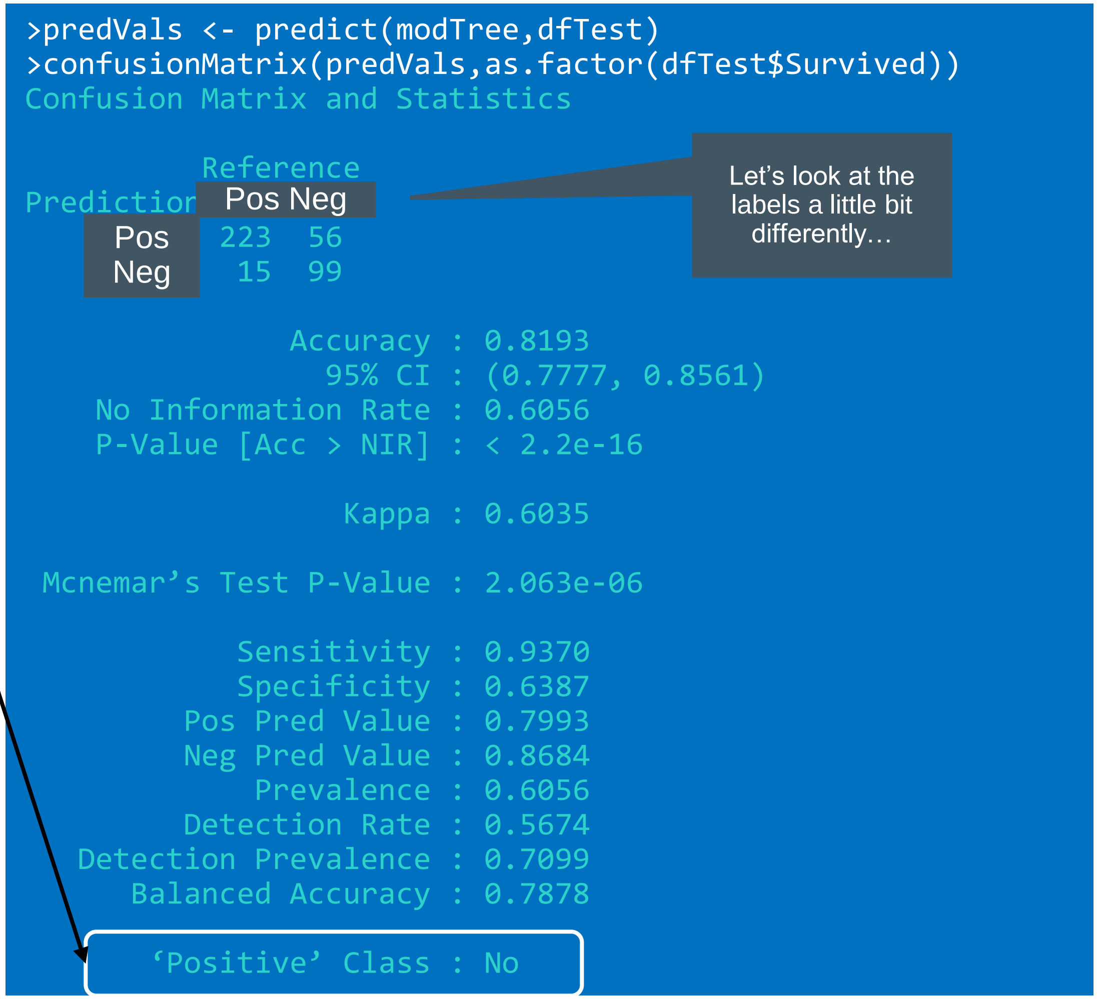
```
>predVals <- predict(modTree,dfTest)
>confusionMatrix(predVals,as.factor(dfTest$Survived))
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  223  56
       Yes  15  99

               Accuracy : 0.8193
                 95% CI : (0.7777, 0.8561)
    No Information Rate : 0.6056
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6035

 Mcnemar's Test P-Value : 2.063e-06

            Sensitivity : 0.9370
            Specificity : 0.6387
         Pos Pred Value : 0.7993
         Neg Pred Value : 0.8684
             Prevalence : 0.6056
         Detection Rate : 0.5674
   Detection Prevalence : 0.7099
      Balanced Accuracy : 0.7878

       'Positive' Class : No
```

Note that the predicted values are stored as factors, so convert the values in the test dataset to factors to make sure it is an apples-to-apples comparison

Machine Learning

Wait…I'm not getting the same results…

- Because the model has specified that "No" is a positive value, it gets a bit confusing (but that's not why it's called a Confusion Matrix…)
- Sensitivity
  - True positives (223) / all positive values (223+15)
  - 223+15 is the True Positive Rate (TPR)
  - 223 / (223+15) = 0.9370
- Specificity
  - True negatives (99) / all negative values (56+99)
  - 56+99 is the True Negative Rate (TNR)
  - 99 / (56+99) = 0.6387
- Pos Pred Value
  - True positives (223) / all predictive positive values (223+56)
  - 223 / (223+56) = 0.7993
- Neg Pred Value
  - True negatives (99) / all predicted negative values (15+99)
  - 99 / (15+99) = 0.8684
- Always check what the stated 'Positive' class is to make sure the calculations are done correctly



```
>predVals <- predict(modTree,dfTest)
>confusionMatrix(predVals,as.factor(dfTest$Survived))
Confusion Matrix and Statistics

          Reference
Prediction Pos Neg
       Pos 223  56
       Neg  15  99

               Accuracy : 0.8193
                 95% CI : (0.7777, 0.8561)
    No Information Rate : 0.6056
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6035

 Mcnemar's Test P-Value : 2.063e-06

            Sensitivity : 0.9370
            Specificity : 0.6387
         Pos Pred Value : 0.7993
         Neg Pred Value : 0.8684
             Prevalence : 0.6056
         Detection Rate : 0.5674
   Detection Prevalence : 0.7099
      Balanced Accuracy : 0.7878

       'Positive' Class : No
```

Document the analysis

Hewlett Packard Enterprise

Document the analysis

Stop and reflect

- On March 23, 1989, Martin Fleischmann and Stanley Pons claimed that a table-top apparatus had produced anomalous heat (understood as "excess" heat) of a magnitude they asserted would defy explanation except in terms of nuclear processes, which later came to be referred to as "cold fusion." In addition to the results from calorimetry, they further reported measuring small amounts of nuclear reaction byproducts, including neutrons and tritium. The reported results received wide media attention and raised hopes of a cheap and abundant source of energy.
- Many scientists tried to replicate the experiment with the few details available. Hopes faded due to the large number of negative replications, the withdrawal of many reported positive replications, the discovery of flaws and sources of experimental error in the original experiment, and finally the discovery that Fleischmann and Pons had not actually detected nuclear reaction byproducts. By late 1989, most scientists considered cold fusion claims dead.
- An article from BBC News published in 2017 states, “Science is facing a ‘reproducibility crisis’ where more than two-thirds of researchers have tried and failed to reproduce another scientist’s experiments, research suggests.”
- Data science is not (and should not be) exempt from requiring reproducible research

Document the analysis

Make your analysis reproducible

- In your documentation, describe every step of your analysis
  - What were your original sources of data? When were they collected?
  - If you generated random numbers in your analysis, what seed value did you use?
  - What cleaning steps did you perform?
- Explain the rationale behind your conclusions with easy-to-understand statements
- Ask others to replicate your analysis when possible
- Use embedded formulas and charts (as opposed to static values and pictures) in your documentation so that the underlying data can be seen
- Be prepared to defend your analysis

Principal Component Analysis

Hewlett Packard Enterprise

Principal Component Analysis

Overview

- PCA is an unsupervised model, which means we don’t need to specify a response variable
- PCA helps narrow the number of features to be evaluated
- Useful in studying “wide” data sets (sets with many variables)
  - Having too many variables makes it difficult for exploratory data analysis
- Method analyzes Principal Components
  - Principal Components describe the direction and magnitude of largest variance among the variables
  - Each principal component explains a certain percentage of the overall variance
  - The first principal component will have the most substantial variance, the second one will have the second most substantial, and so on
- PCA performs linear transformations to create the principal components
  - The result is that each principal component has both a direction (eigenvector) and a magnitude (eigenvalue)
- The number of principal components will match the number of data dimensions
- Useful in data sets that have large numbers of features, some of the features have a lot of variability or ‘noise’ in them, and/or not all of the features are independent
  - Facial recognition is a common application of PCA

Principal Component Analysis

PCA with the Iris data set

– Load the Iris data set

– Create the PCA model for the data set, removing categorical variables

```
>dfData <- as.data.frame(iris)
>pcaData <- prcomp(dfData[,-5],center=TRUE,scale=TRUE)
>pcaData
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971

>summary(pcaData)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

- Remember that the -5 as the second argument in the subset of dfData means to keep all variables except for the fifth variable, which is the categorical variable “Species”
- Also remember that we don’t want to alter the original data set
- Displaying the model itself will show the variance of each principal component with respect to each variable
- Displaying the summary of the model will show the overall principal component values. As you can see, the principal components are in descending order of their Proportion of Variance; PC1 will have the largest, PC2 will have the second largest, etc.
- PC1 and PC2 account for 95.81% of the variance, so both of them should be sufficient to create a training model

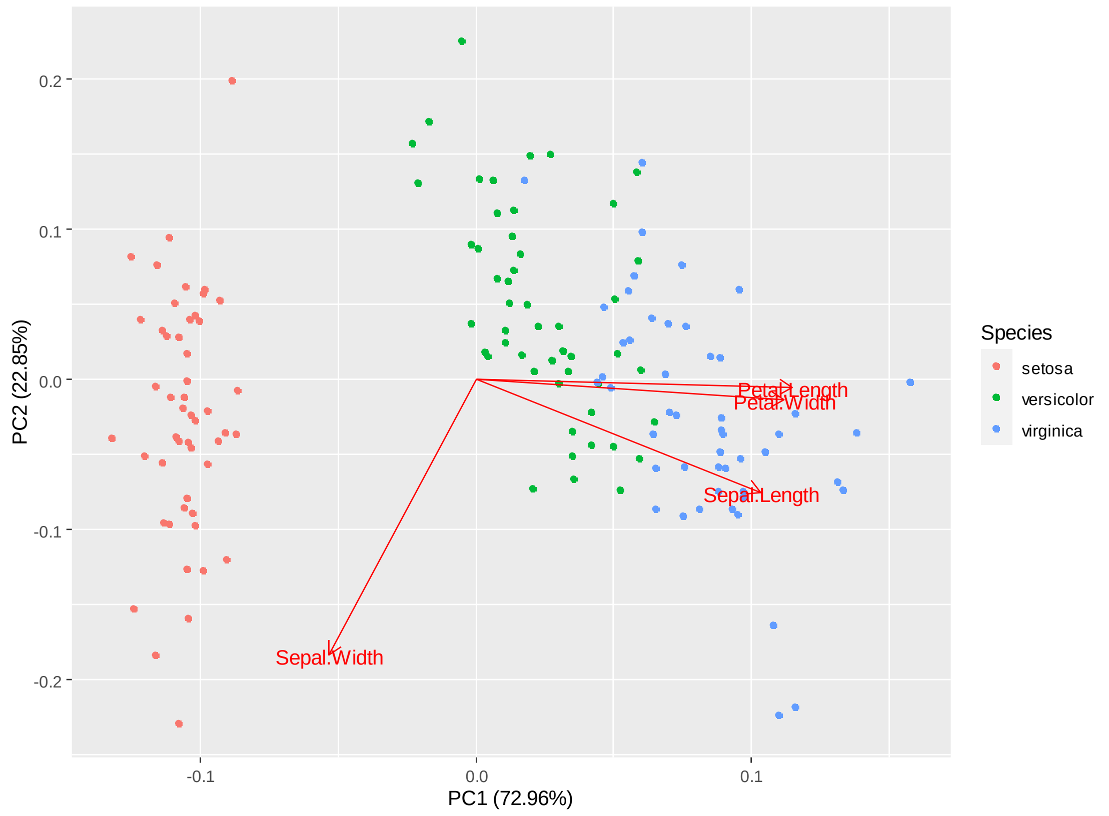
Hewlett Packard Enterprise

Principal Component Analysis

Displaying principal components on top of the initial data set

– Note: the autoplot() function comes from the package “ggfortify”

```
>autoplot(pcaData,loadings=TRUE,loadings.label=TRUE,data=dfData,colour=“Species”)
```



- The variables are now displayed as vectors, showing their respective variance to PC1 and PC2
- Just focusing on the x values, Petal.Length, Petal.Width and Sepal.Length have a strong positive correlation to PC1, while Sepal.Width has a negative correlation
- Just focusing on the y values, Sepal.Width has a very strong negative correlation to PC2, while the other variables have a very weak negative correlation
- Most interesting, though, is that while the species values were used for the color of the markers, they aren’t part of the data set being plotted! These are the PC1 and PC2 values of each observation. (As an unsupervised model, it’s done a fairly good job of grouping the data)

Hewlett Packard Enterprise

Principal Component Analysis

Iris data set using a decision tree ML model

– Train the model with a training data set

```
>set.seed(1138)
>nTrain <- sample(nrow(dfData),nrow(dfData)*0.7,replace=FALSE)
>dfTrain <- dfData[nTrain,]
>dfTest <- dfData[-nTrain,]
>modTree <- train(Species~.,data=dfTrain,method="rpart")
>modTree
CART

105 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.0000000  0.9443338  0.9144603
  0.4545455  0.6705055  0.5368619
  0.5000000  0.5923632  0.4221453

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.
```

- Multiplying the number of rows by 0.7 sets 70% of the observations to the training data set
- For the model, all variables (except the response variable) are used

Principal Component Analysis

Iris data set using a random forest ML model (continued)

– Predict values for the training data set and display the confusion matrix

```
>szPredict <- predict(modTree,dfTest)
>confusionMatrix(szPredict,dfTest$Species)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         11          0         0
  versicolor      0         16         2
  virginica       0          1        15

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.8173, 0.986)
    No Information Rate : 0.3778
    P-Value [Acc > NIR] : 6.255e-15

                  Kappa : 0.8982

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9412           0.8824
Specificity                 1.0000            0.9286           0.9643
Pos Pred Value              1.0000            0.8889           0.9375
Neg Pred Value              1.0000            0.9630           0.9310
Prevalence                  0.2444            0.3778           0.3778
Detection Rate              0.2444            0.3556           0.3333
Detection Prevalence        0.2444            0.4000           0.3556
Balanced Accuracy           1.0000            0.9349           0.9233
```

- The results of the model are excellent
- One versicolor plant was incorrectly predicted as a virginica plant
- Two virginica plants were incorrectly predicted as versicolor plants

Hewlett Packard Enterprise

Principal Component Analysis

Iris data set using a PCA model

– Train the model with a training data set

```
>set.seed(1138)
>nTrain <- sample(nrow(dfData),nrow(dfData)*0.7,replace=FALSE)
>dfTrain <- dfData[nTrain,]
>dfTest <- dfData[-nTrain,]
>modPCA <- prcomp(dfTrain[,-5],center=TRUE,scale=TRUE)
>modPCA
Standard deviations (1, .., p=4):
[1] 1.7286319 0.9251886 0.3706444 0.1359426

Rotation (n x k) = (4 x 4):
                    PC1        PC2        PC3        PC4
Sepal.Length  0.5129154 0.40633386  0.7219553  0.2249249
Sepal.Width  -0.3082396 0.90871524 -0.2560184 -0.1169596
Petal.Length  0.5733471 0.04061818 -0.1816125 -0.7978974
Petal.Width   0.5596242 0.08648451 -0.6166459  0.5468903

>summary(modPCA)
Importance of components:
                         PC1    PC2     PC3     PC4
Standard deviation     1.729 0.9252 0.37064 0.13594
Proportion of Variance 0.747 0.2140 0.03434 0.00462
Cumulative Proportion  0.747 0.9610 0.99538 1.00000
```

- Results are very similar to our first exercise, but in order to demonstrate how well a model performs, we still need to create a model using training data and then comparing it to predictions from the test data set

Hewlett Packard Enterprise

Principal Component Analysis

Iris data set using a PCA model (continued)

– Create transformed data sets and then use a decision tree model

```
>pcaTrain <- predict(modPCA,dfTrain)
>pcaTrain <- as.data.frame(pcaTrain)
>pcaTrain$Species <- dfTrain$Species
>pcaTest <- predict(modPCA,dfTest)
>pcaTest <- as.data.frame(pcaTest)
>pcaTest$Species <- dfTest$Species

>modTreePCA <- train(Species~PC1+PC2,data=pcaTrain,method=“rpart”)
>modTreePCA
CART

105 samples
  2 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.0000000  0.9093610  0.8617955
  0.3939394  0.7023216  0.5675153
  0.5000000  0.6253264  0.4478846

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.
```

- Using predict() will give us a data frame with the same number of observations as the training and test data sets and with the new principal components
- Since the output is a matrix array, change it to a data frame so that it can accept the “Species” values
- The goal will be to create a new model just using the most useful components (PC1 and PC2)
- The initial performance shows an accuracy of 90.93%
  - Remember that the initial model, with all four observations, was 94.43%

Hewlett Packard Enterprise

Principal Component Analysis

Iris data set using a PCA model (continued)

– Check the confusion matrix with the predictions of the test data

```
>szPredictPCA <- predict(modTreePCA,pcaTest)
>confusionMatrix(szPredictPCA,pcaTest$Species)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         11          0         0
  versicolor      0         16         4
  virginica       0          1        13

Overall Statistics

               Accuracy : 0.8889
                 95% CI : (0.7595, 0.9629)
    No Information Rate : 0.3778
    P-Value [Acc > NIR] : 1.51e-12

                  Kappa : 0.8303

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9412           0.7647
Specificity                 1.0000            0.8571           0.9643
Pos Pred Value              1.0000            0.8000           0.9286
Neg Pred Value              1.0000            0.9600           0.8710
Prevalence                  0.2444            0.3778           0.3778
Detection Rate              0.2444            0.3556           0.2889
Detection Prevalence        0.2444            0.4444           0.3111
Balanced Accuracy           1.0000            0.8992           0.8645
```

- The results of this model are excellent
- One versicolor plant was incorrectly predicted as a virginica plant
- Four virginica plants were incorrectly predicted as versicolor plants
- This was a simple example; imagine if your data set had dozens of features
- Two things to remember:
  - The second model only used two features instead of four
  - The PCA model is unsupervised, so if we didn’t have response data, the PCA model would’ve created logical groupings

Hewlett Packard Enterprise

Key Concepts

Hewlett Packard Enterprise

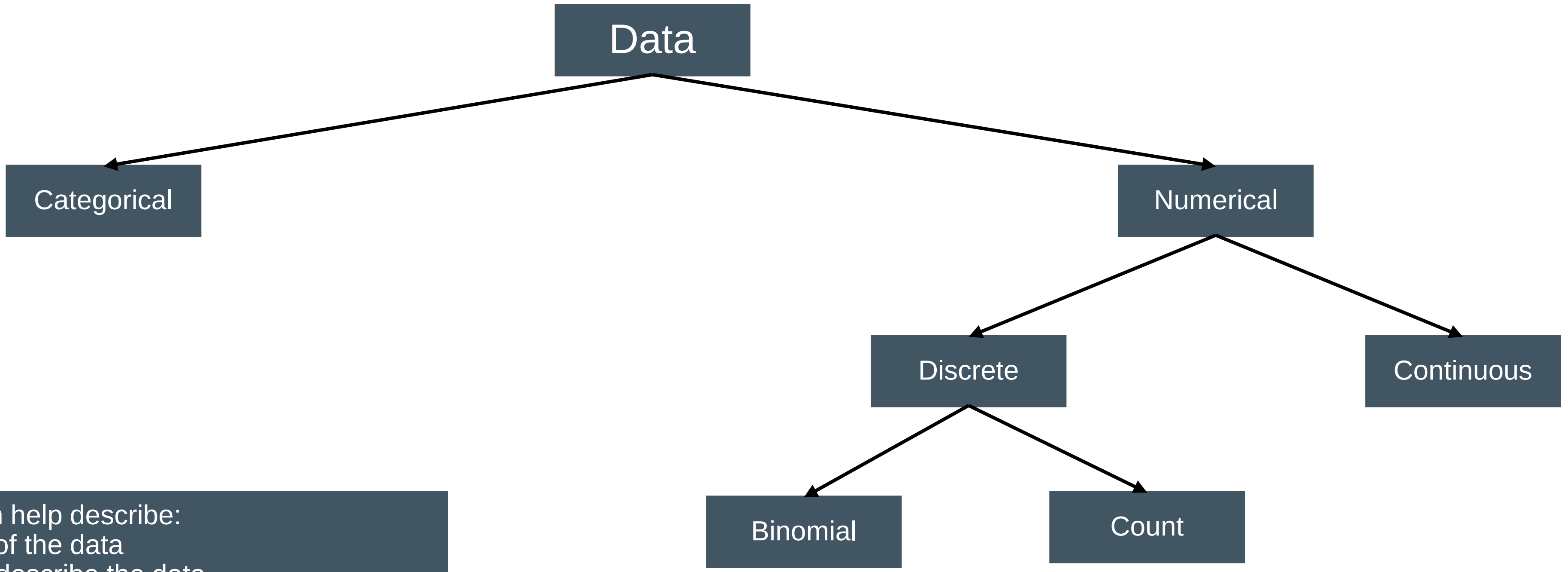
Key Concepts

Business Analytics

- Refers to the skills, technologies and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning
- Three levels of application
  - Descriptive analytics: gain insight from historical data with reporting and scorecards
  - Predictive analytics: employs predictive modeling using statistical and machine learning techniques
  - Prescriptive analytics: recommends decisions through optimization and simulation
- As the levels increase (descriptive > predictive > prescriptive), the value to management increases, as well as the difficulty to implement
  - Descriptive analytics show where the business has been (no foresight)
  - Predictive analytics show what the business will be (foresight in performance)
  - Prescriptive analytics show how to get better performance (foresight in actions)

Key Concepts

Data Types



Knowing the data type can help describe:

- Expected distribution of the data
- Statistics available to describe the data

Hewlett Packard Enterprise

Key Concepts

Probability

- Probability is the branch of mathematics concerning numerical descriptions of how likely an event is to occur, or how likely it is that a proposition is true
- The probability of an event is a number between 0 and 1
  - Roughly speaking, 0 indicates impossibility of the event and 1 indicates certainty
- For discrete events, there are several formulas available to calculate probability based on the number of ways a particular event can occur
- In statistics, there are several formulas available to calculate probability based on the location, dispersion and distribution types of data
  - This is the p-value that is reported from statistical tests
  - Hypothesis testing is based on whether or not a probability (p-value) is above or below an alpha level (usually 0.05)

Key Concepts

Confusion Matrix

- Tool used to describe actual versus predicted results for categorical data
  - Typically used for TRUE/FALSE designations of some sort of test, but it is not limited to a 2x2 table
- Example: Using a model to predict if 100 students will pass a test or not

	Actual Pass	Actual Fail
Predicted Pass	68	5
Predicted Fail	6	21

68 students passed and the model predicted they would pass (TP = true positive = 68)
5 students failed but the model predicted they would pass (FP = false positive = 5)
6 students passed but the model predicted they would fail (FN = false negative = 6)
21 students failed and the model predicted they would fail (TN = true negative = 21)
74 students passed regardless of the prediction (TPR = true positive rate = 68 + 6 = 74)
26 students failed regardless of the prediction (TNR = true negative rate = 5 + 21 = 26)

Sensitivity is the true positives / all positives (TP / TPR = 68 / 74 = 0.9189)
Specificity is the true negatives / all negatives (TN / TNR = 21 / 26 = 0.8077)
Accuracy is the correct values / all values (TP + TN / TPR + TNR = 68 + 21 / 74 + 26 = 89 / 100 = 0.89)

Key Concepts

Supervised and Unsupervised Learning

- In supervised learning, the data set includes the response variable (the value to predict) for all observations
  - The data set is split into a training set and a test set (in some cases, a third set called the cross-validation set is set aside as well)
    - The training set is used to 'train' the machine; the response variable is used to create the model (usually 70% of the data)
    - The test set is used to 'test' the accuracy of the model; the response variable is not used to create predictions
    - The test set can have the response variable present when making predictions; the model will only use features that were used in the creation of the model (in other words, the model is going to ignore the response variable during prediction if they exist)
    - A comparison of the response values and predicted values from the test set determine the accuracy of the model
    - If a cross-validation set is used, it will typically have different parameter values for the model (e.g. number of trees in a forest) but some applications can include validation techniques (such as bootstrapping) during the training process
- In unsupervised learning, the data set does not include a response variable
  - For categorical data, the model can define compelling groups for the data; the user will need to determine the meaning and usefulness of the groups (groups of people watching Netflix programs may group them into a definable set of preferences)
  - For numerical data, the model can define thresholds or patterns (credit card transactions that appear fraudulent based on frequency, amount, and other features)

Key Concepts

Data Stratification

– Recall the concept of ‘dispersion’ in data – the ‘spread’ of the observations with respect to its location
  – Variation, standard deviation, and quartiles are all examples of dispersion statistics

– Also, recall the concept that there can be several sources of variation within the data
  – Common-cause variation is natural and relatively predictable; special-cause variation is unpredictable

– ‘Homogeneous’ data typically have low variability; ‘heterogeneous’ data typically have higher variability
  – A group of 20-year old college students in a Physics 101 class would be a homogeneous sample
  – A group of 18-21 year old college students in Physics 101, History 112 and Chemistry 211 would be a heterogeneous sample

– Sources of variation in heterogeneous data sets could be in secondary factors
  – In order to remove some of those secondary factors, we use a technique called ‘stratification’

Key Concepts

Data Stratification Example

- Imagine a dataset describing the length of time to deploy a bot solution for a group of projects
  - What would the average be?
  - What would the variance (standard deviation) be?
  - What would be some of the factors contributing to the variance?
    - Complexity of the project
    - Skill/experience of the developer
    - Stability of the process being automated
- Imagine stratifying the data based on one of those attributes
  - What would the averages be?
  - What would the variances (standard deviations) be?
  - Would the dispersion of the data for those stratified data sets be smaller or larger than the whole?