Introduction to Data Science using R

Topics

- Using R for Statistical Modeling and Exploratory Data Analysis

    - Data Science Overview, Regression and Classification Techniques, R and R, vectors, data frames, reading data from text files and web. Using summary, mean, std, median, min, max, quartile, box plot, scatter plot, and hist functions .

    - Data transformation and subsetting methods for model builds, using Apply function, assessing accuracy of models, Plotting results of models for interpretation and tuning purposes.

    - Build different kinds of graphs (scatterplot, stacked bar chart, lineplot, sunflowerplot, pie chart, heatmap, histogram, density plots, word clouds). Building dynamic Dashboards with Shiny, and Using R Markdown.

    - Basics of building predictive models using R, Linear regression, Logistic Regression, Bayes Models, Trees, Clustering, and Random Forest.

Labs & Reading

- We will use R / RStudio for labs in class
- Suggested Reading
  - Introduction to Statistical Learning http://www-bcf.usc.edu/~gareth/ISL/

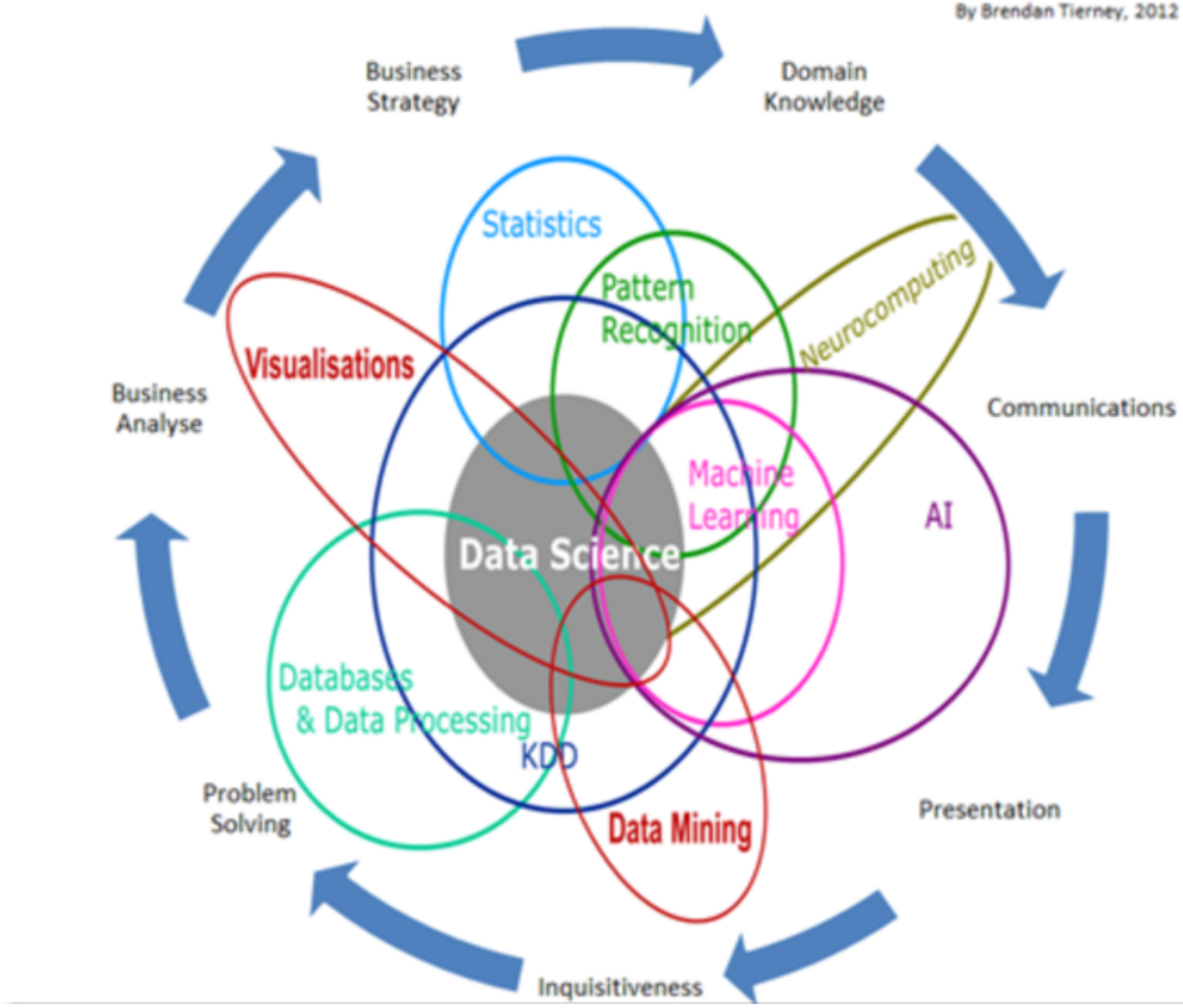
What is Data Science & Data Mining?

Data science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured, which is a continuation of some of the data analysis fields such as statistics, data mining, and predictive analytics

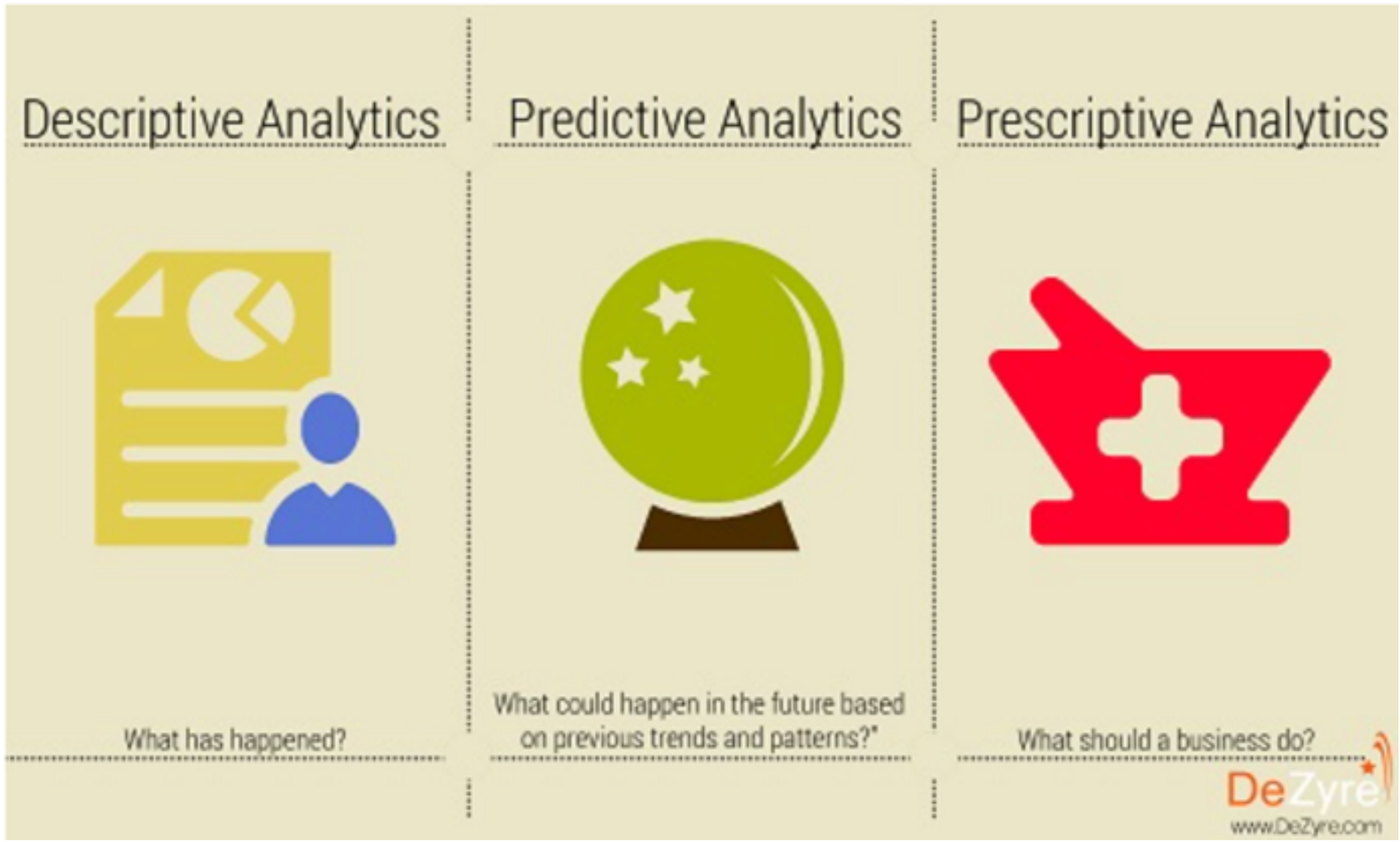
Source: Wikipedia

Data mining is basically the process of discovering patterns in large data sets

Data Science Is Multidisciplinary



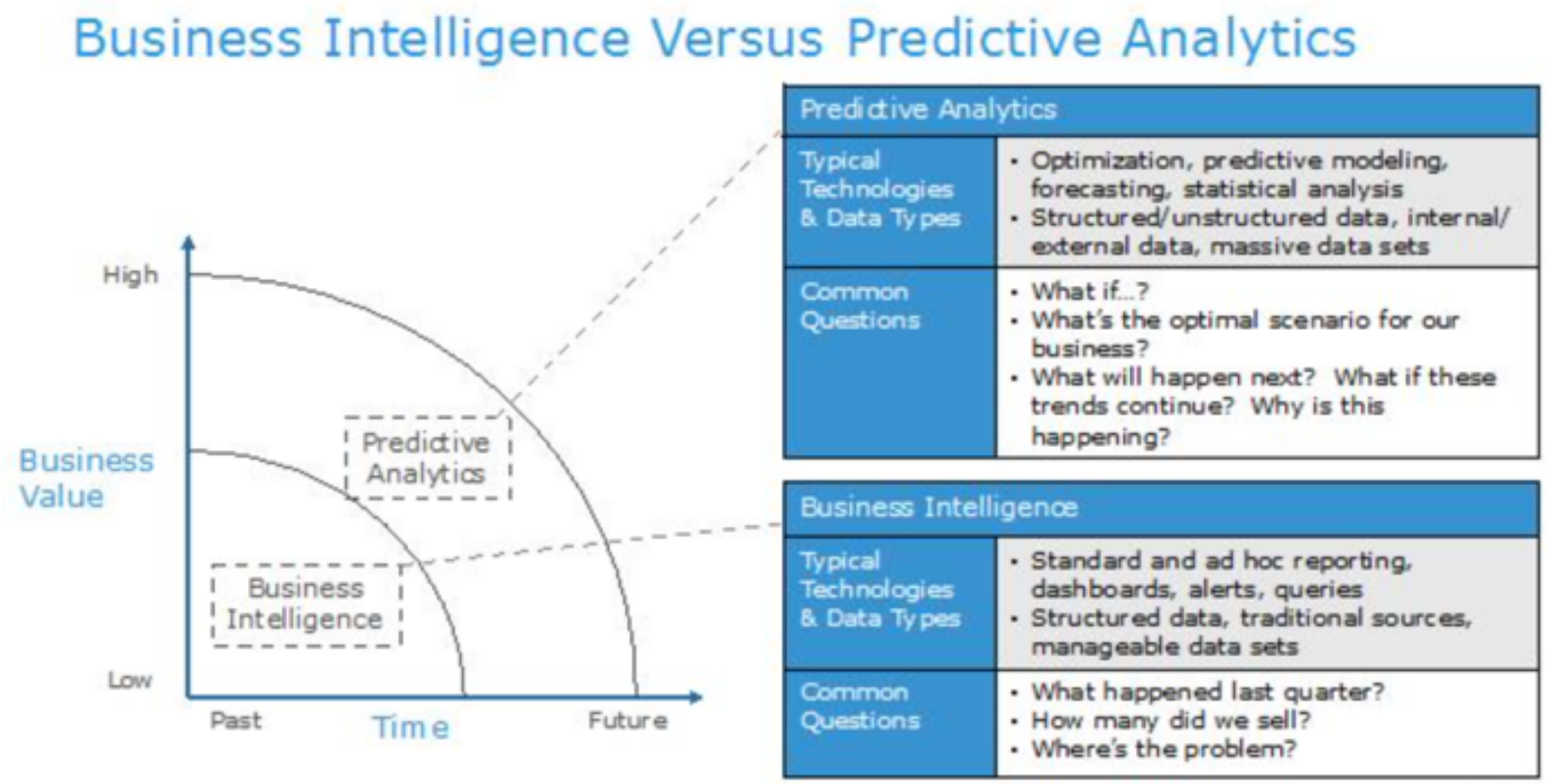
Types of Analytics: descriptive, predictive, prescriptive analytics



Prescriptive Analytics Provides Advice Based on Predictions

Source: https://www.dezyre.com

Types of Analytics: descriptive, predictive, prescriptive analytics



Source: EMC²

Prescriptive Analytics Provides Advice Based on Predictions

- Contains a) actionable data and b) a feedback system that tracks the outcome produced by the action taken

Use Cases (Customer/ Sales) - Sample

Recommendation Engine	Upsell, Position products
Customer Churn	Predict churn probability and prevent
Customer Segmentation	Upsell, Product Positioning and Profit Maximization
Fraud Detection	Predict and prevent
Sentiment Analysis	Feedback from social network and other avenues

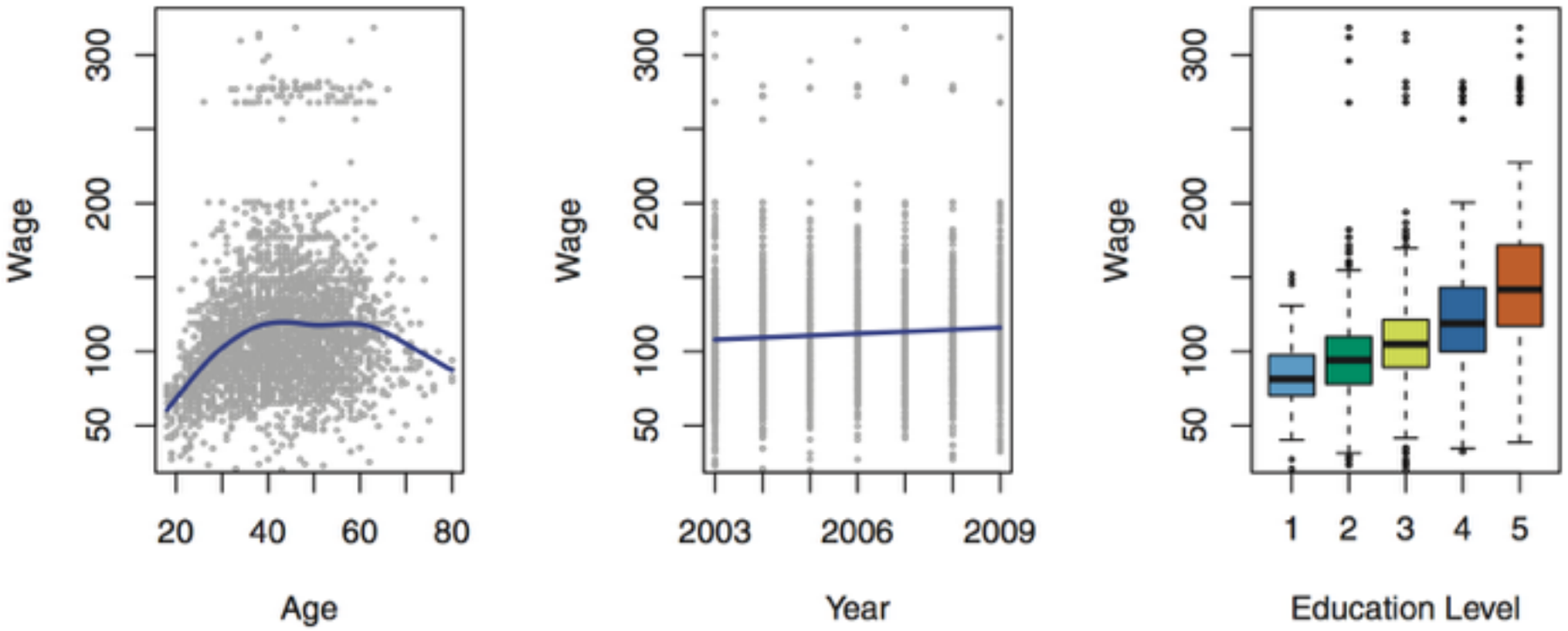
Quotes - Topic: What is going to be the biggest data trend of 2016?

Brian Hopkins, VP and Principal Strategest at Forrester Research
“Machine learning will reduce the insight killer — time. Machine learning will replace manual data wrangling and data governance dirty work. The freeing up of time will accelerate data strategies.”

Kirk Borne, Principal Data Scientist at Booz Allen Hamilton and founder of RocketDataScience.org
“In 2016, the world of big data will focus more on smart data, regardless of size. Smart data are wide data (high variety), not necessarily deep data (high volume). Data are “smart” when they consist of feature-rich content and context (time, location, associations, links, interdependencies, etc.) that enable intelligent and even autonomous data-driven processes, discoveries, decisions, and applications.”

Paul Zikopoulos, VP of Analytics at IBM
“I would say Data science for the masses is one and the other is more disruption with open source technologies to the point that no one knows what Hadoop means anymore, and more projects we never heard of trying to flatten the time to data science.”

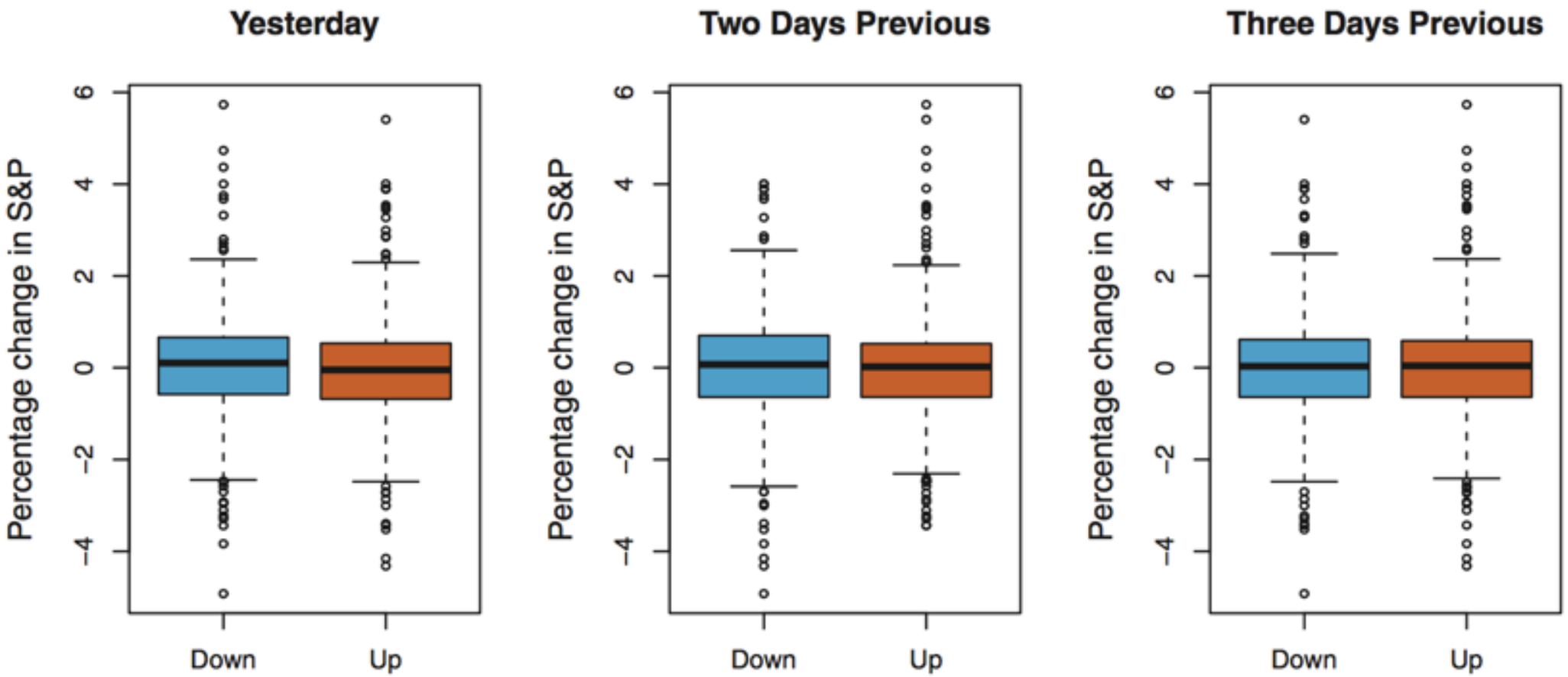
Wage Data



How to predict wage?

Wage = f(Age, Year, Education Level)

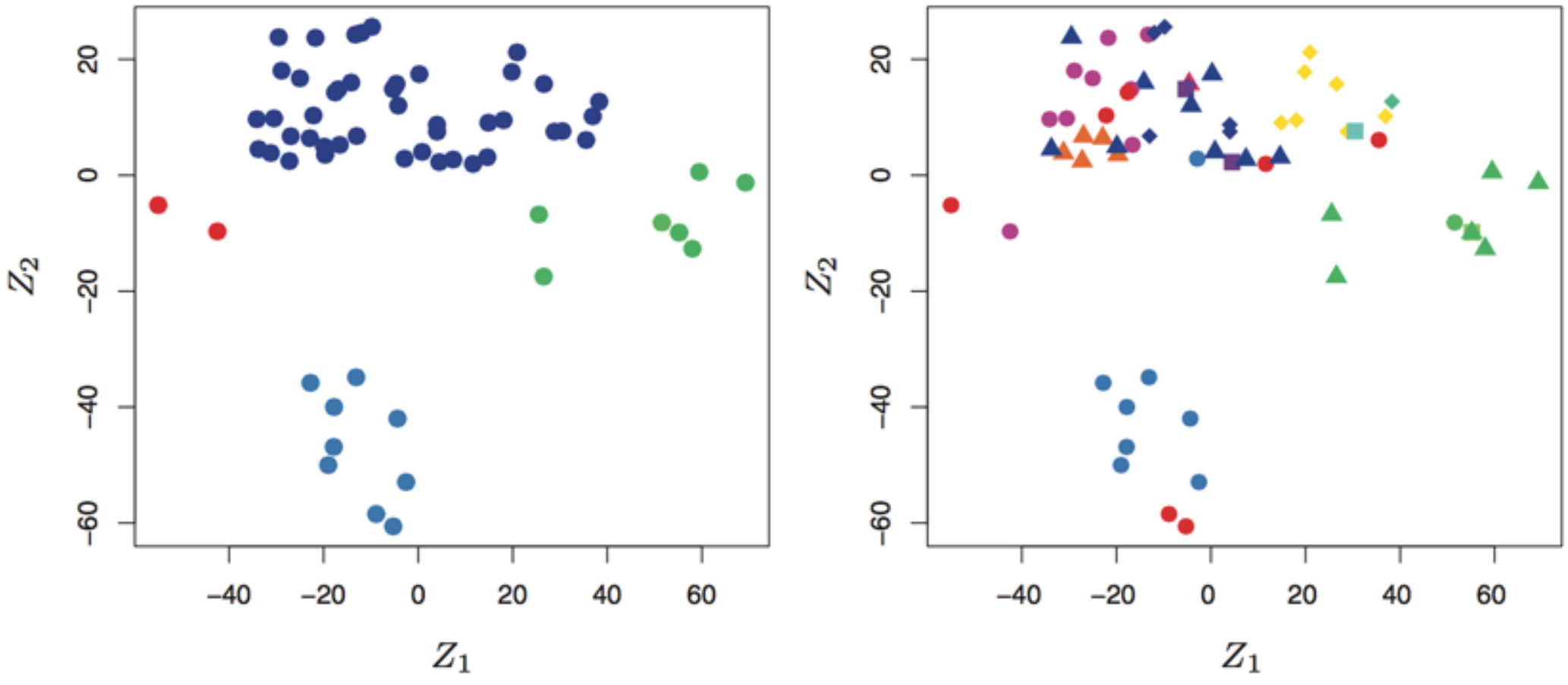
Stock Market Data



How to predict today's direction?

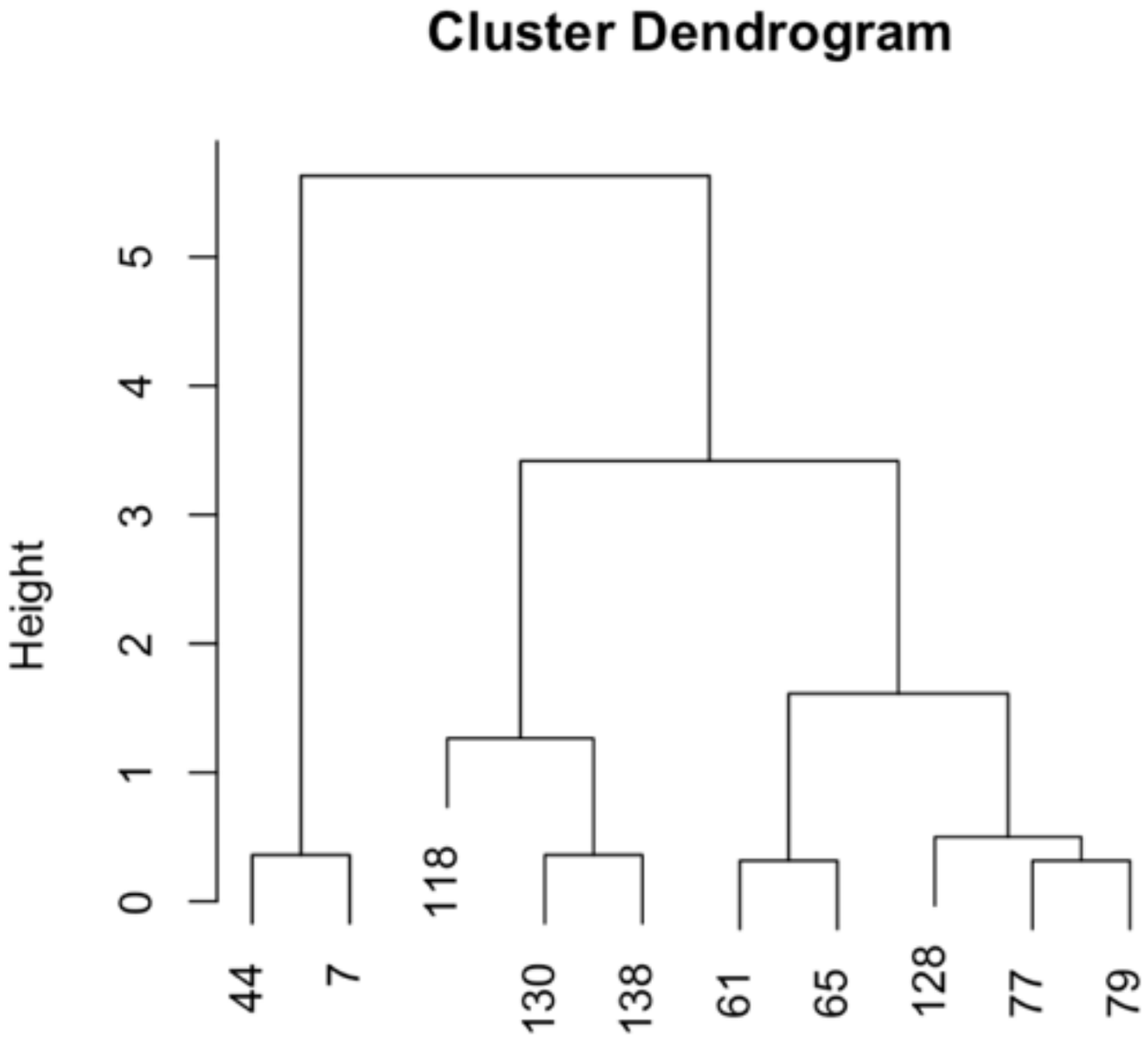
Direction_0 = f(Direction_1, Direction_2, Direction_3)

Gene Expression Data

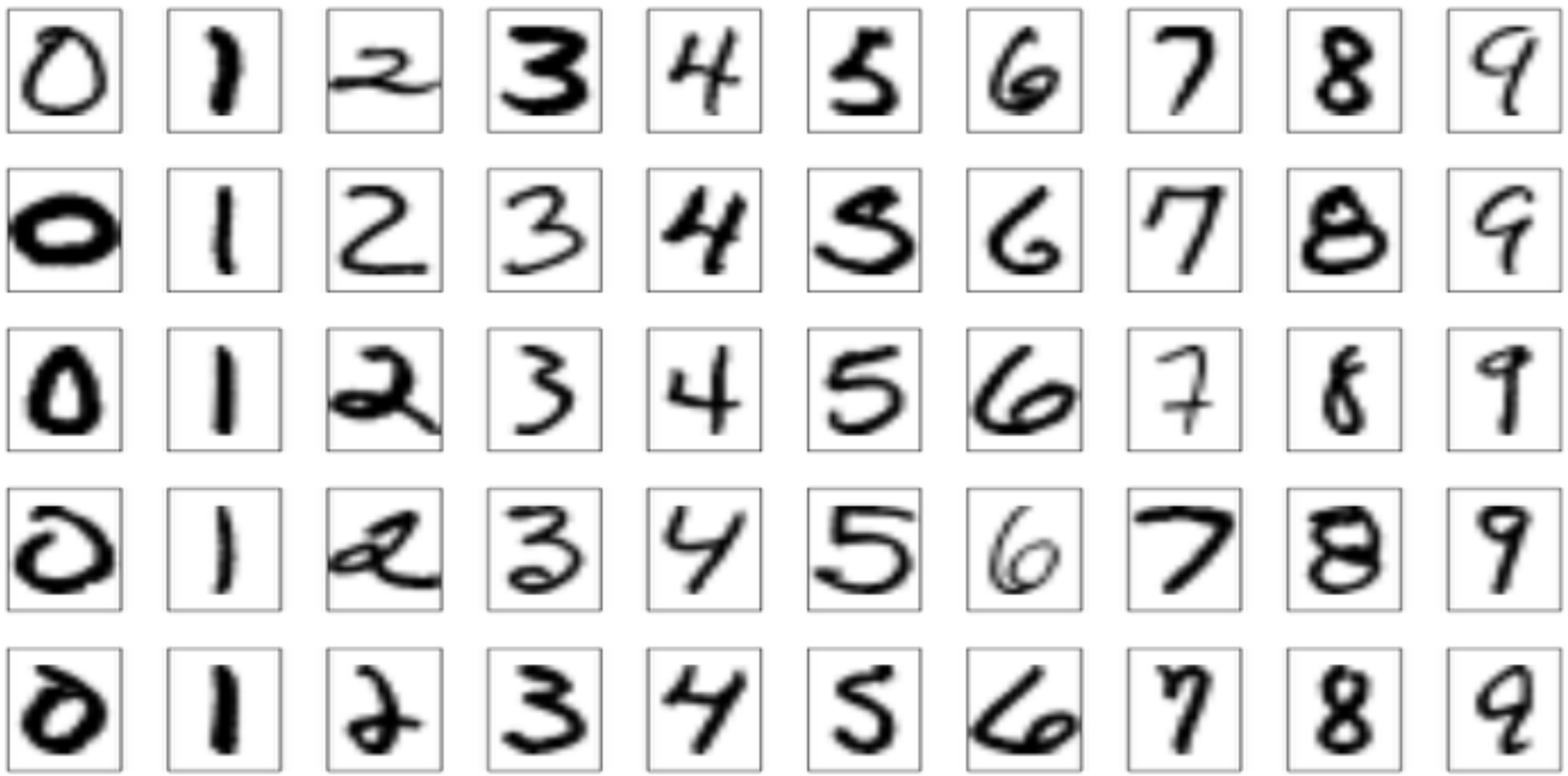


Clustering

Hierarchical Clustering



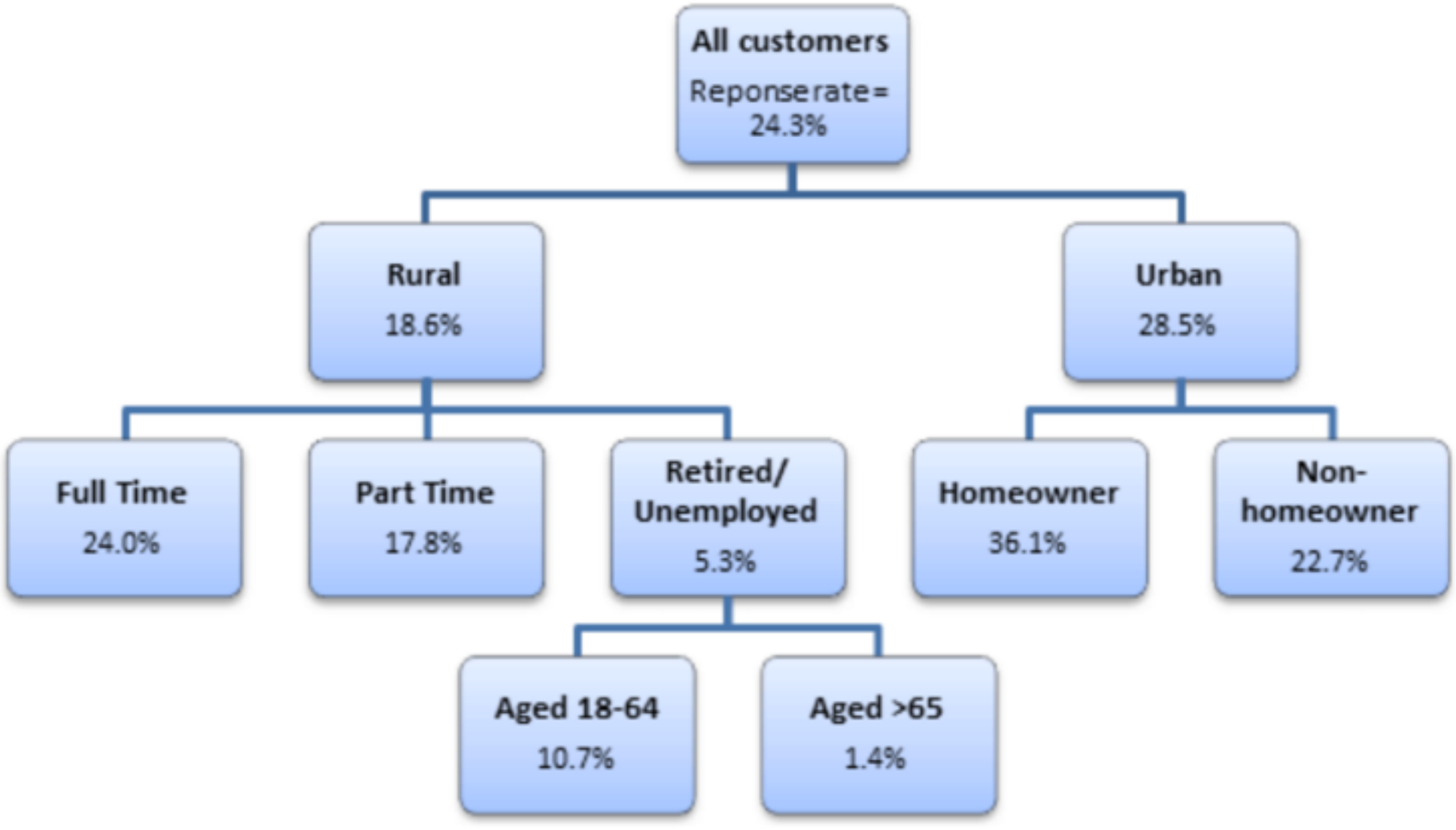
Recognize digit



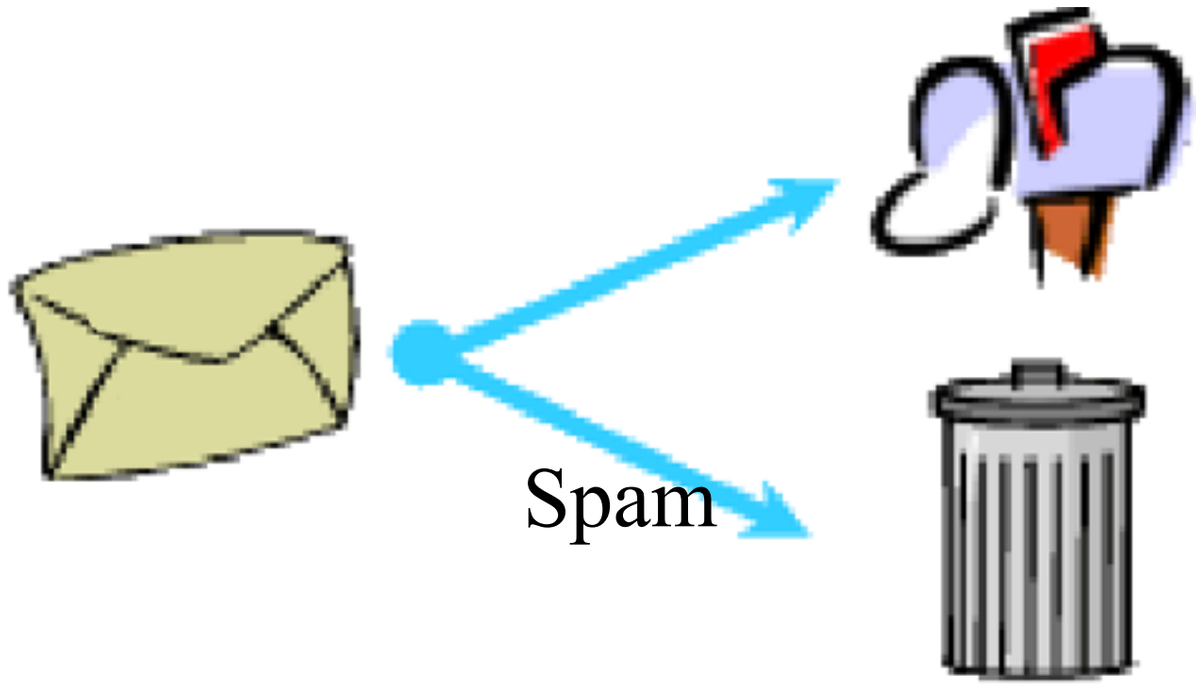
Each image is a 16 × 16 8-bit grayscale representation of a handwritten digit

Customer segmentation

Customer segmentation



Spam Detection



Spam Detection

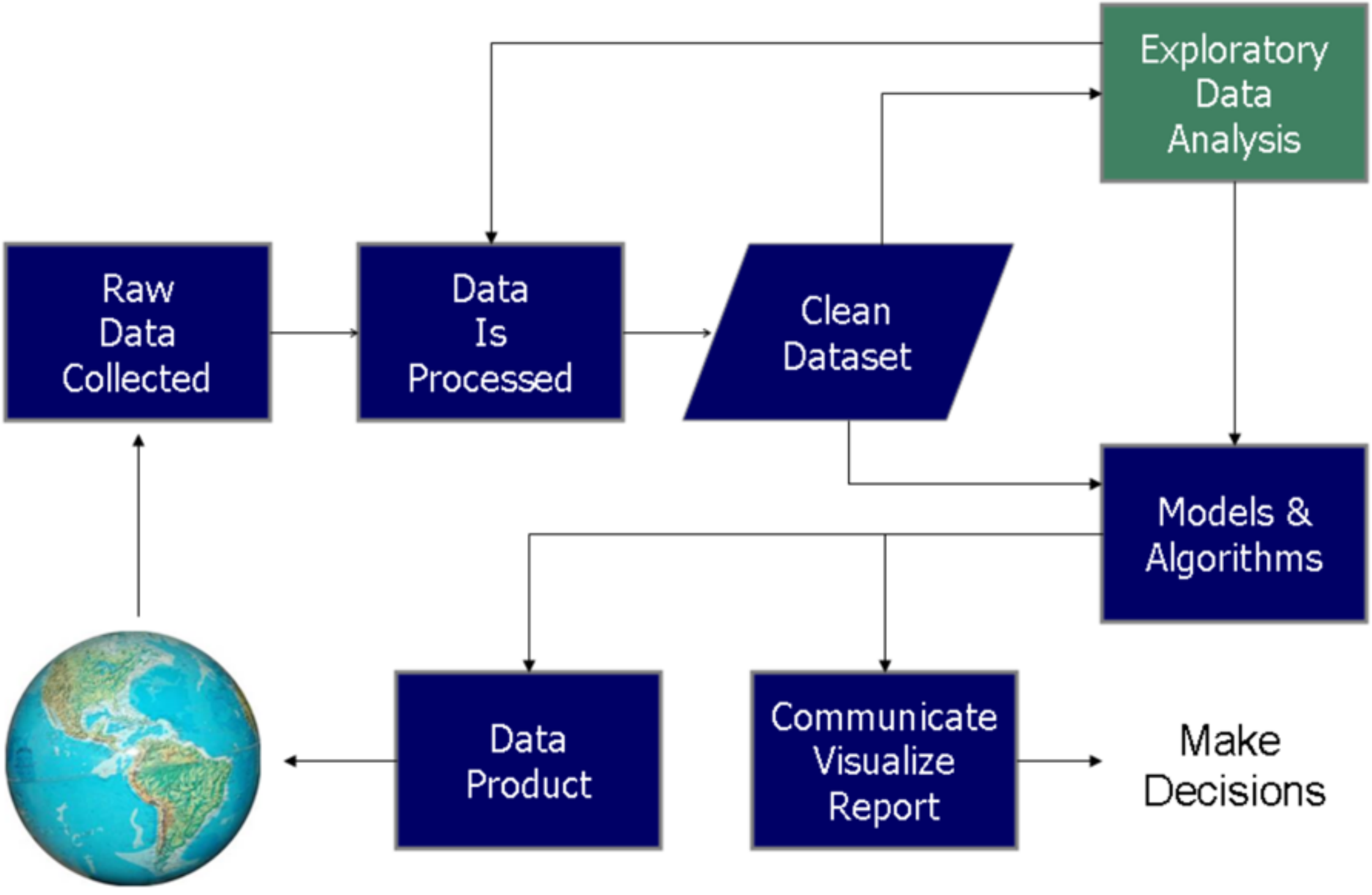
Spam Detection

- data from 4601 emails sent to an individual (named George, at HP labs, before 2000). Each is labeled as *spam* or *email*.
- goal: build a customized spam filter.
- input features: relative frequencies of 57 of the most commonly occurring words and punctuation marks in these email messages.

	george	you	hp	free	!	edu	remove
spam	0.00	2.26	0.02	0.52	0.51	0.01	0.28
email	1.27	1.27	0.90	0.07	0.11	0.29	0.01

Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between spam *and* email.

Data Science Process



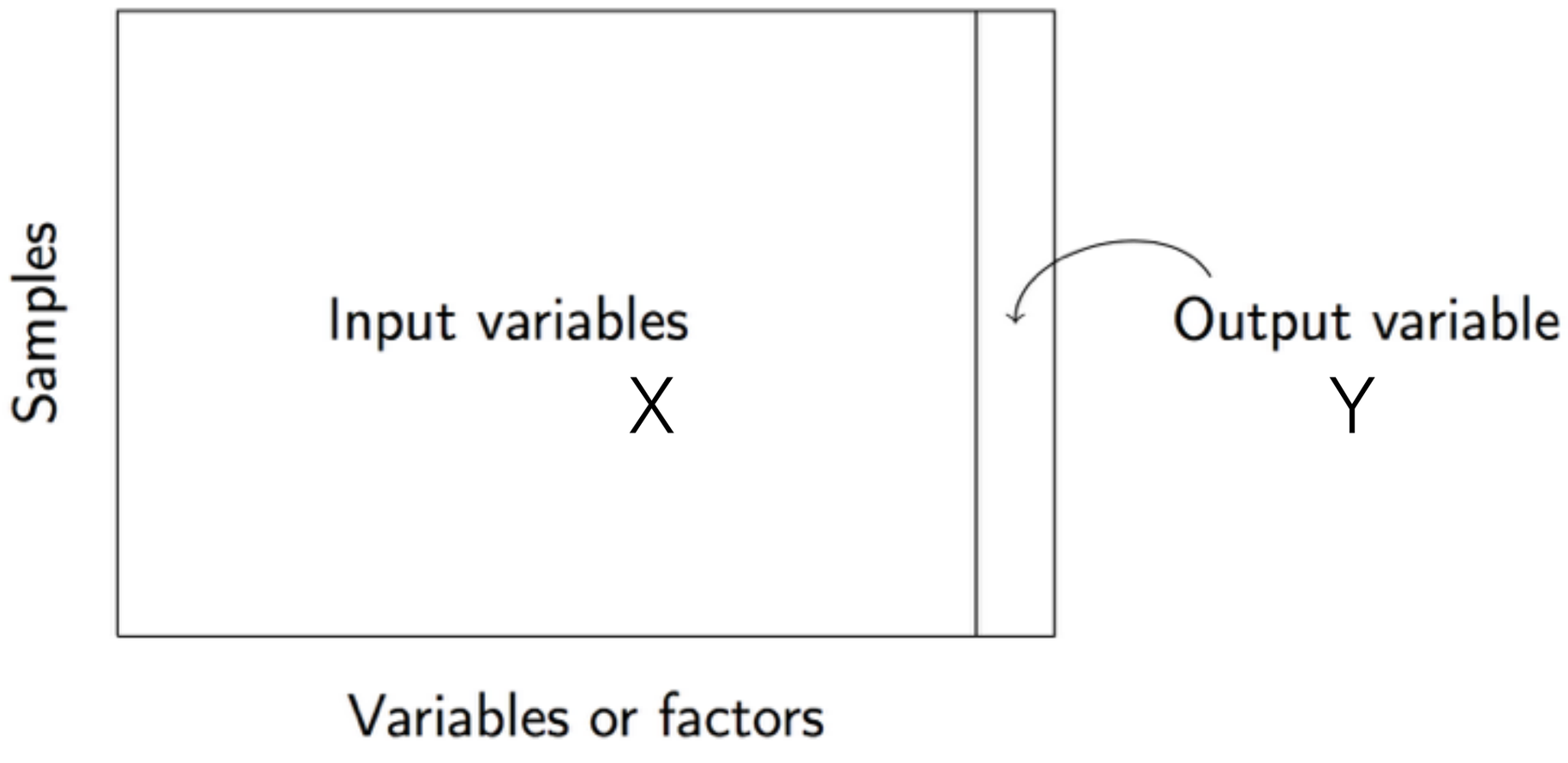
Source: Wikipedia

ML Intro

- Supervised vs Unsupervised Learning
- Regression vs Classification

Supervised vs Unsupervised Learning

In **supervised learning**, there are *input* variables, and *output* variables:



Supervised vs Unsupervised

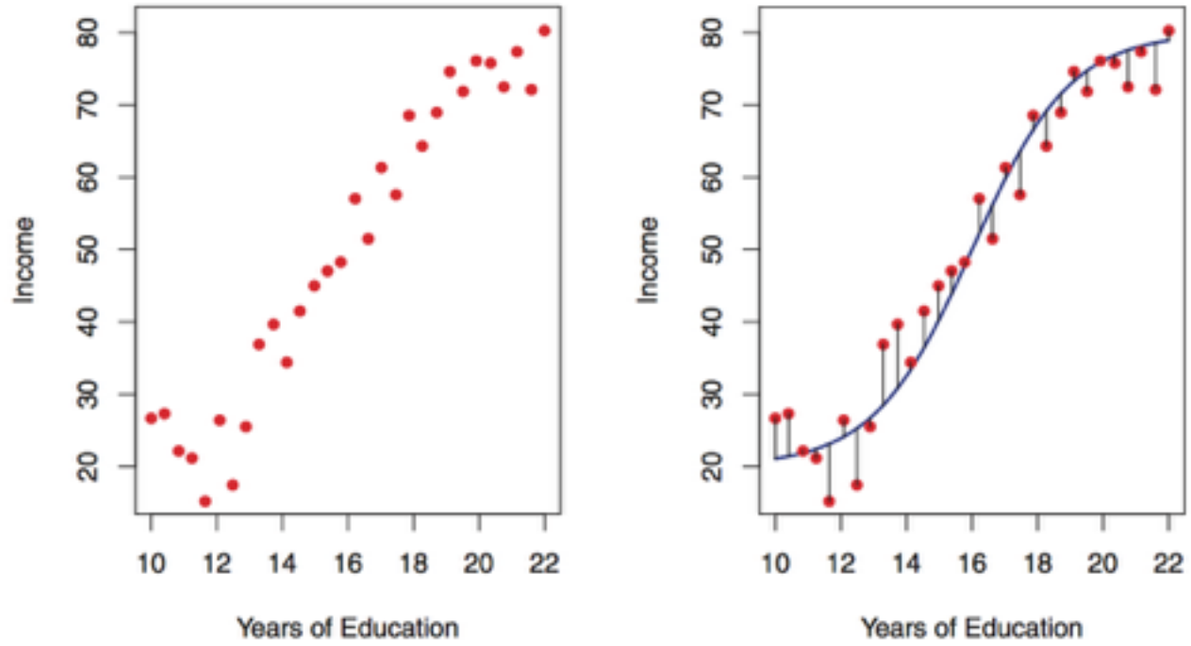
- Supervised
    - There is an outcome measurement Y (dependent variable, response)
    - A set of predictors (independent variables, features)
    - Purpose: Predict unknown response with known input variables; Understand the relationship between variables
- Unsupervised
    - No outcome variable; instead a set or grouping of data
    - Purpose:

Regression vs Classification

- Regression
    - Typically quantitative responses
    - Examples: Person's weight, Stock Price, Car Price
- Classification
    - Typically qualitative responses
    - Examples: Person's gender, Stock Price direction, Disease diagnosis

What Is Statistical Learning?

- response = f(predictors)



- $Y = f(X) + \varepsilon$

Estimate f

- $\hat{Y} = \hat{f}(X)$

Models - How to determine f(X)?

- $Y = f(X) + \varepsilon$
- Example: Income = f(Years of Education) + ε
- How to determine f(Years of Education)?
    - All we have is: sample dataset

Models

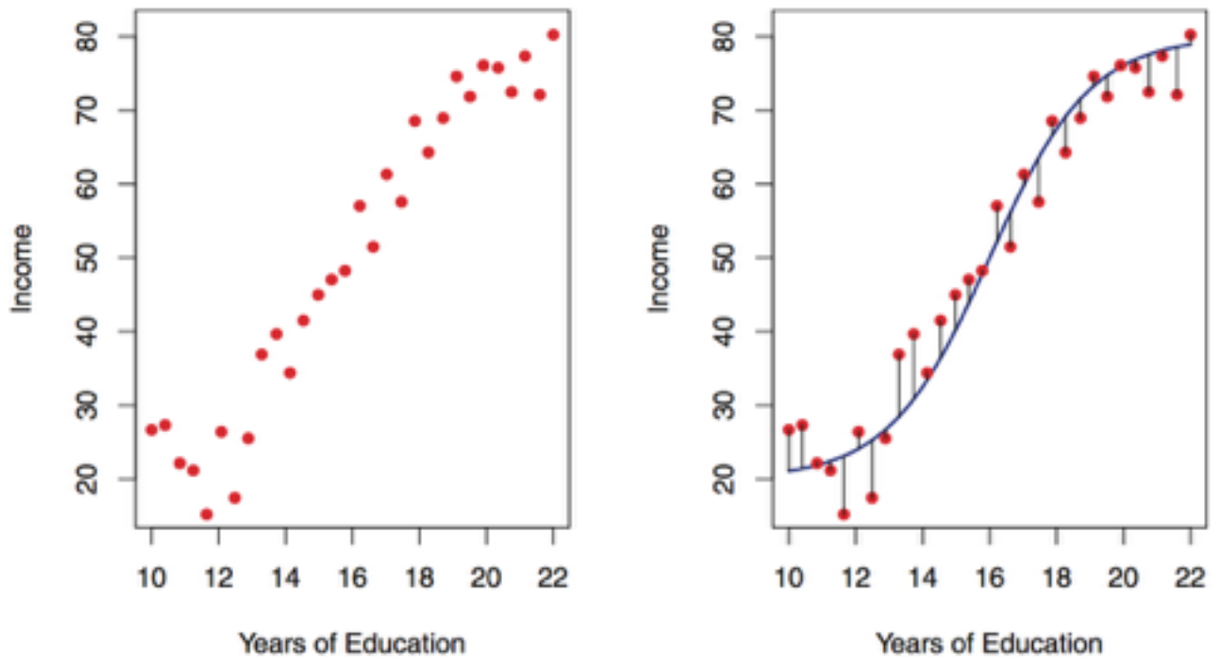
1. Choose the function f
2. Decide how to score the function
3. Estimate parameters - Search

Choose function f

- Class of functions
  - Linear functions: $\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$
  - Nearest Neighbor Model $\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$
  - Additive functions
  - SVM , Boosting, Trees, neural networks and more
- No Method is universally better than any other (reasonable one)
  - David Wolpert: No free lunch theorem

Score Function

- Judges how well the models fits
- We use sample data to estimate the Loss function
- Ordinary Linear Regression
    - $Income_{data}$ - $Income_{predicted}$
    - Most commonly used
        - $RSS = \sum(y_i - (\beta_0 + \beta_1 x_i))^2$
        - Assumed function in this case is $y = \beta_0 + \beta_1 x$
- Next step
    - How to estimate the parameters? (β_0, β_1)



Estimate Parameters - Search

- Minimize the Loss function using sample data: find the best combination of β_0 and β_1 for which the Loss function L($Income_{data}$, $Income_{predicted}$) is minimum

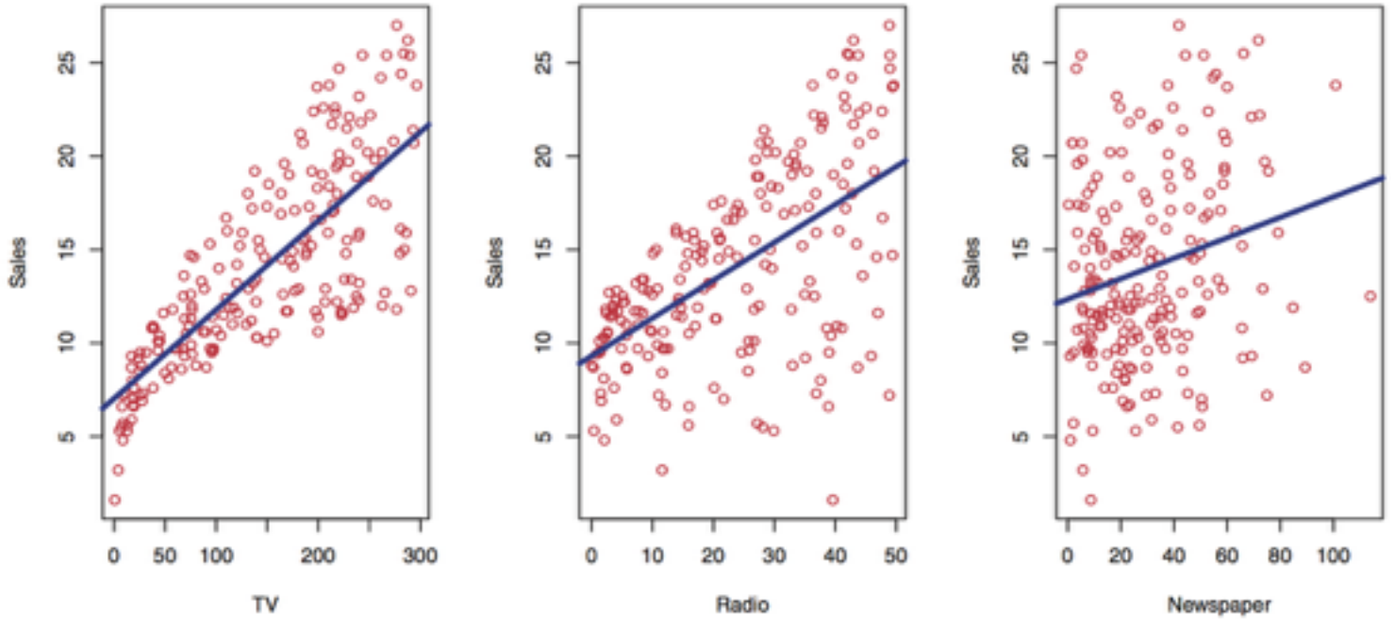
- Minimization method depends on Predictive functions chosen and Loss functions

Now we have a model

- Example

- Income = 15 + 3.5*(No of Years of Education)

What Is Statistical Learning?



- Response: Sales
- Predictors: TV, Radio, Newspaper
- $Y = f(X) + \varepsilon$
- Sales = f(TV, Radio, Newspaper) + ε

Linear Model

- Sales = f(TV, Radio, Newspaper) + ε
- Sales ≈ $\beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{Radio} + \beta_3 \times \text{Newspaper}$
- Here the function f is: $\beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{Radio} + \beta_3 \times \text{Newspaper}$
- Estimate $\beta_0, \beta_1, \beta_2, \beta_3$ ==> Estimate for f
- Use training data to estimate f
- Estimate Sales using est. f(TV, Radio, Newspaper)

Sample Problems

Forecast sales using store, promotion, and competitor data

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

In their first Kaggle competition, Rossmann is challenging you to predict 6 weeks of daily sales for 1,115 stores located across Germany. Reliable sales forecasts enable store managers to create effective staff schedules that increase productivity and motivation. By helping Rossmann create a robust prediction model, you will help store managers stay focused on what's most important to them: their customers and their teams!

Source: https://www.kaggle.com/c/rossmann-store-sales

Forecast use of a city bikeshare system

Get started on this competition through Kaggle Scripts

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.

Source: https://www.kaggle.com/c/bike-sharing-demand

Can computer vision spot distracted drivers?

We've all been there: a light turns green and the car in front of you doesn't budge. Or, a previously unremarkable vehicle suddenly slows and starts swerving from side-to-side.

When you pass the offending driver, what do you expect to see? You certainly aren't surprised when you spot a driver who is texting, seemingly enraptured by social media, or in a lively hand-held conversation on their phone.

According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. Given a dataset of 2D dashboard camera images, State Farm is challenging Kagglers to classify each driver's behavior. Are they driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat?

Source: https://www.kaggle.com/c/state-farm-distracted-driver-detection

Classify products into the correct category

Get started on this competition through Kaggle Scripts

The Otto Group is one of the world's biggest e-commerce companies, with subsidiaries in more than 20 countries, including Crate & Barrel (USA), Otto.de (Germany) and 3 Suisses (France). We are selling millions of products worldwide every day, with several thousand products being added to our product line.

A consistent analysis of the performance of our products is crucial. However, due to our diverse global infrastructure, many identical products get classified differently. Therefore, the quality of our product analysis depends heavily on the ability to accurately cluster similar products. The better the classification, the more insights we can generate about our product range.

Source: https://www.kaggle.com/c/otto-group-product-classification-challenge

R Intro and Practice

The R Project

- Environment for statistical computing and graphics
  - Free software
- Associated with simple programming language
  - Similar to S and S-plus
- www.r-project.org

The R Project

- Versions of R exist of Windows, MacOS, Linux and various other Unix flavors

- R was originally written by Ross Ihaka and Robert Gentleman, at the University of Auckland

- It is an implementation of the S language, which was principally developed by John Chambers

1984

Compiled C vs Interpreted R

- C requires a complete program to run
  - Program is translated into machine code
  - Can then be executed repeatedly
- R can run interactively
  - Statements converted to machine instructions as they are encountered
  - This is much more flexible, but also slower

R Function Libraries

- Implement many common statistical procedures
- Provide excellent graphics functionality
- A convenient starting point for many data analysis projects

Interactive R

- R defaults to an interactive mode

- A prompt “>” is presented to users

- Each input expression is evaluated…
- … and a result returned

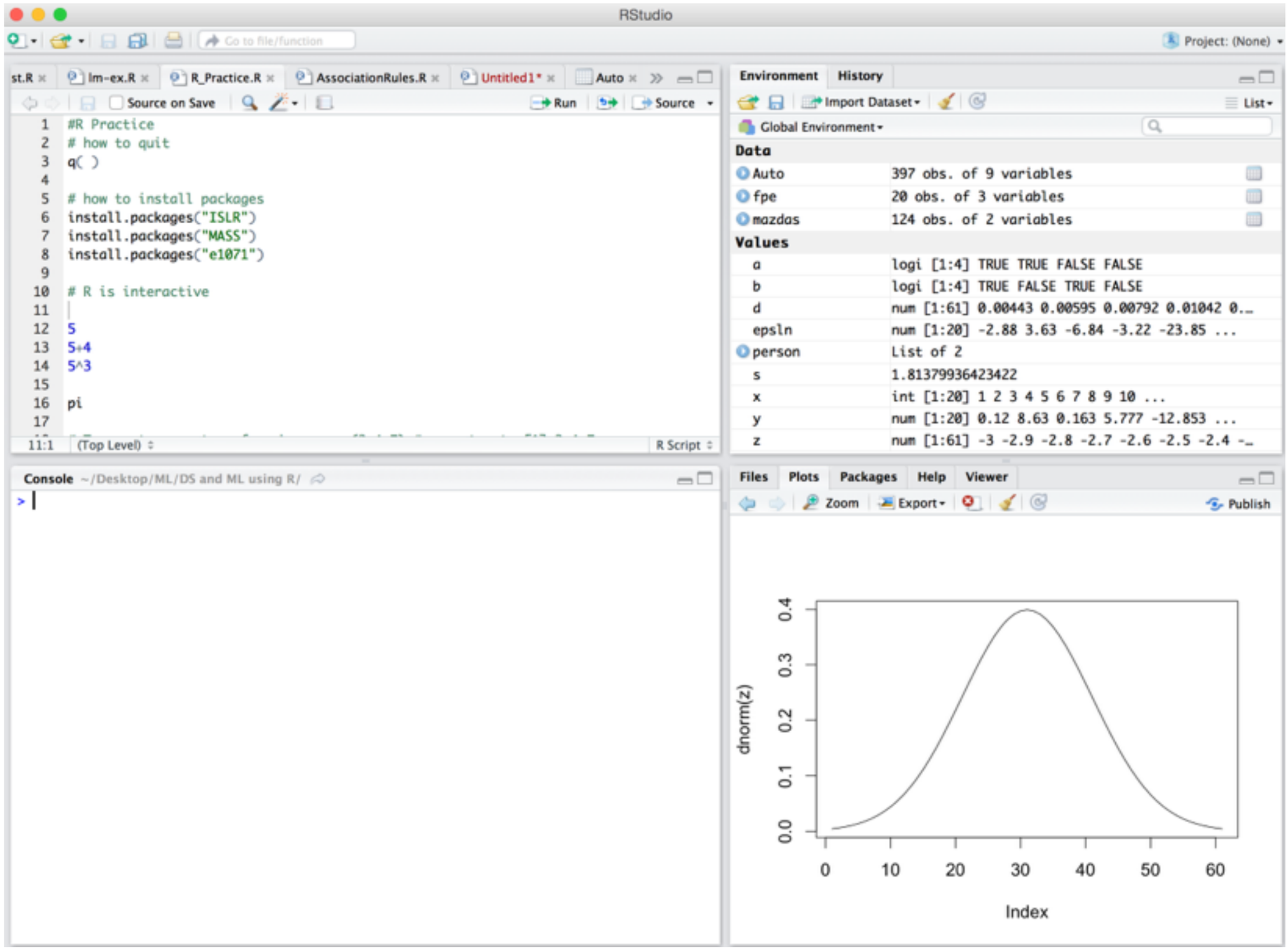
R Practice

- R Console
- Expressions & Assignments
- Vectors
- Lists and Data Frames
- Reading and analyzing data
- Simple Graphs and Plotting Data

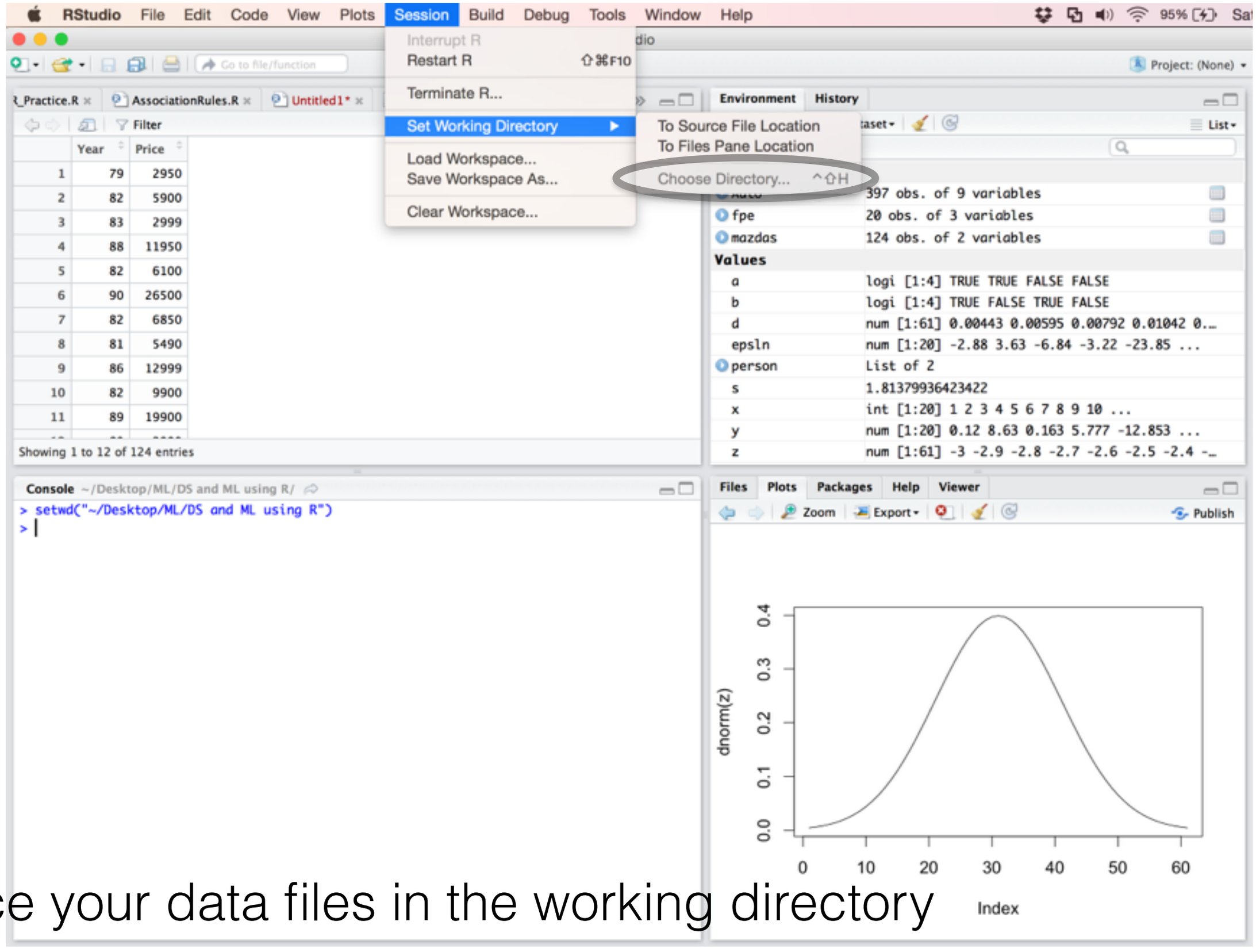
Installation

- R is available at https://cran.r-project.org
    - Install R
- RStudio allows the user to run R in a more user-friendly environment. It is open-source (i.e. free) and available at http://www.rstudio.com/
    - Install RStudio
    - https://www.rstudio.com/products/rstudio/#Desktop

RStudio Screen

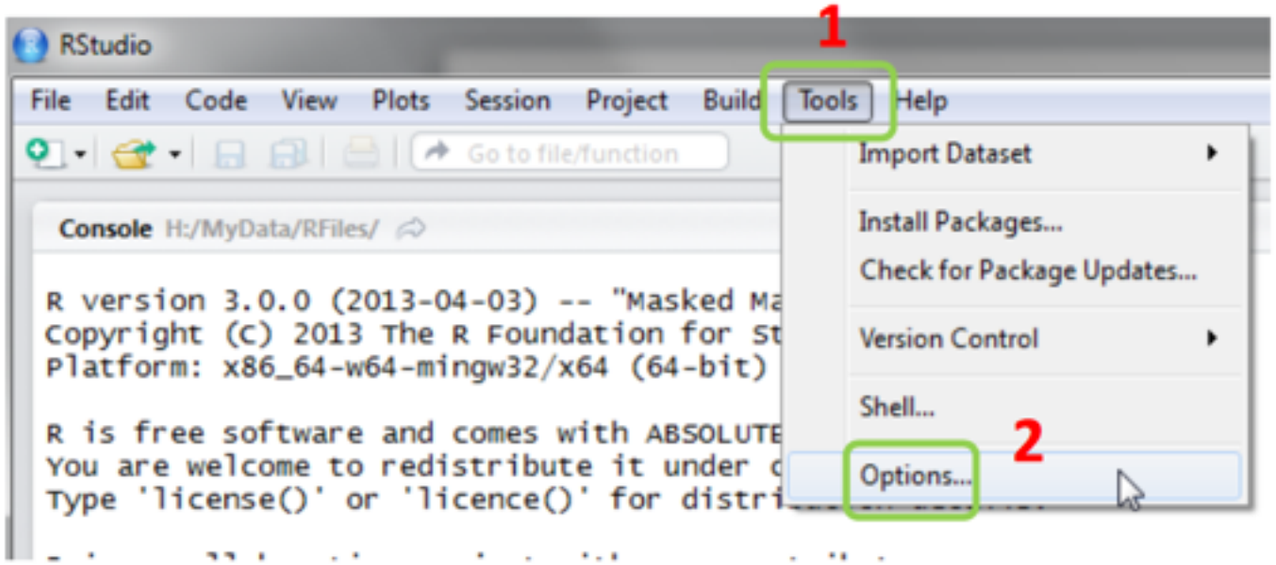


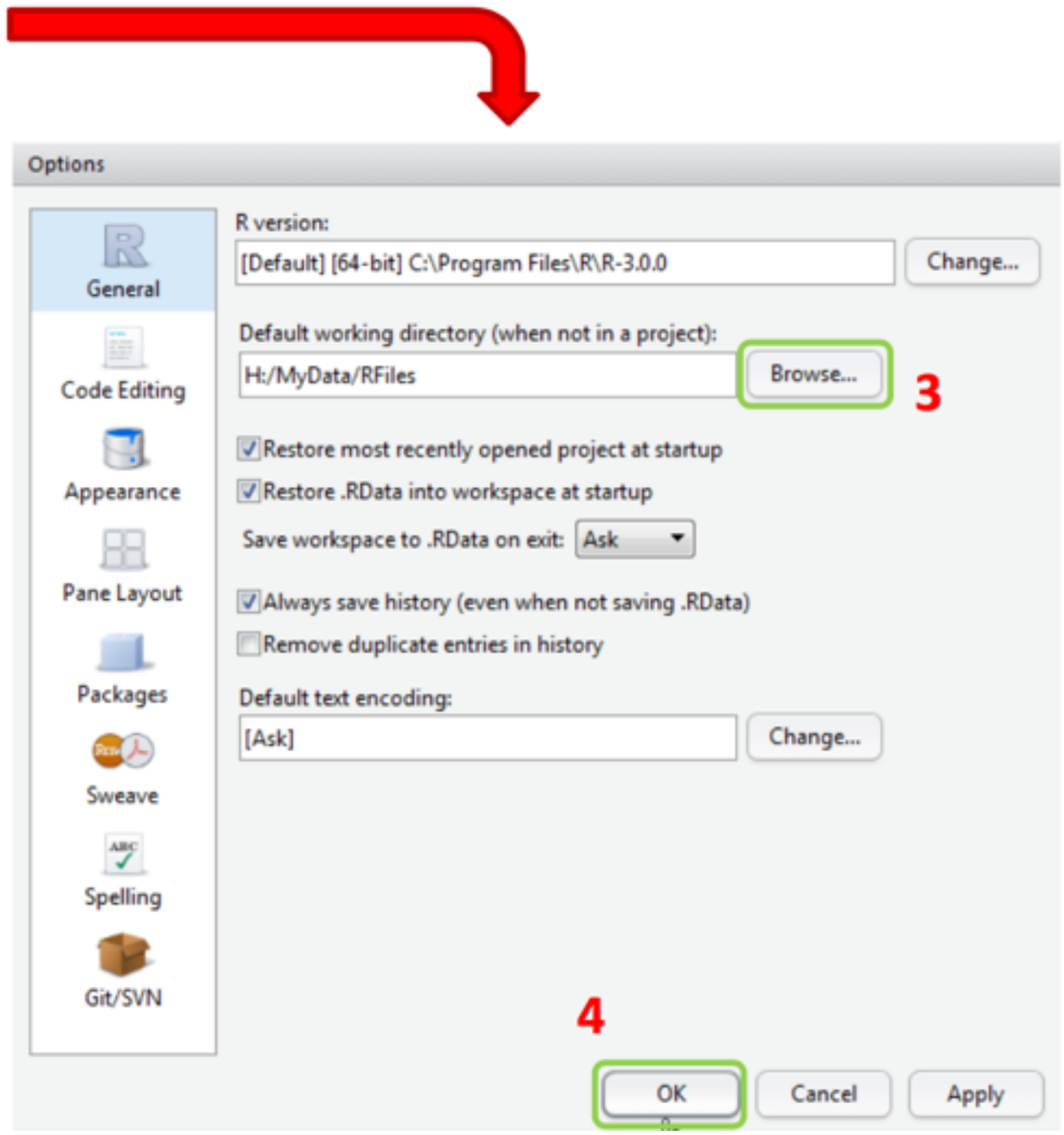
Setting working directory



Place your data files in the working directory

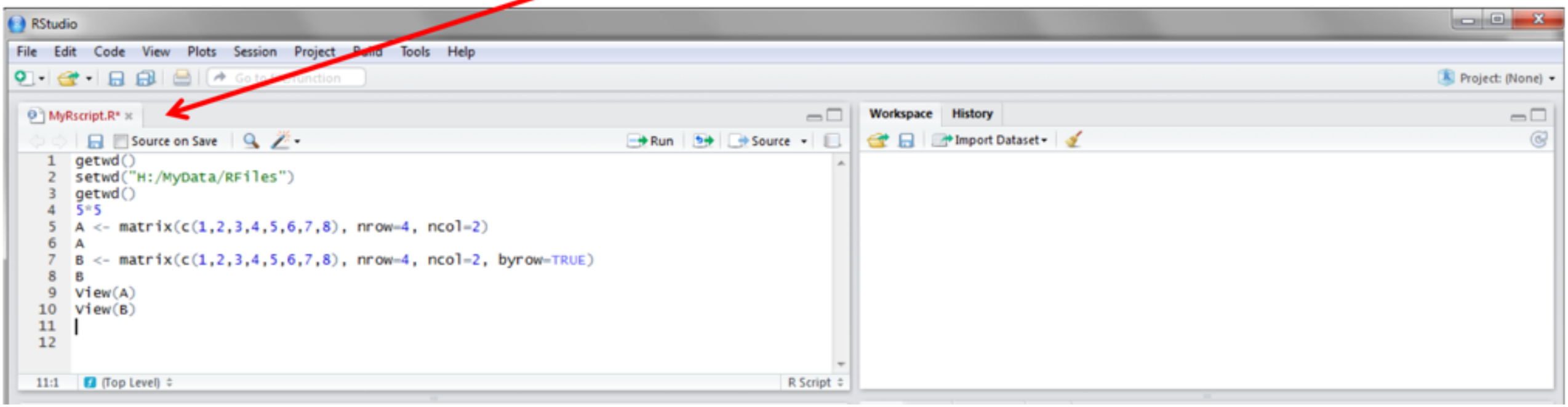
Setting default directory





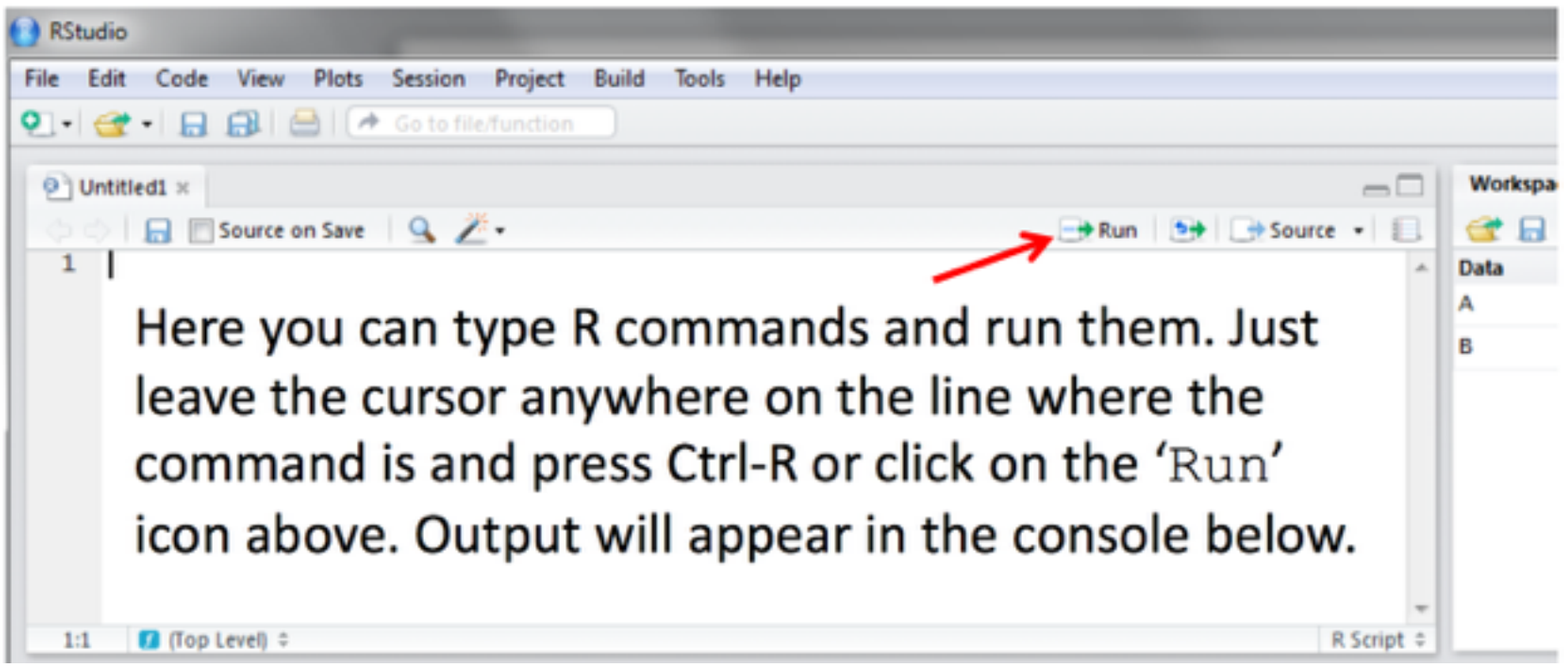
Every time you open RStudio, it goes to a default directory. You can change the default to a folder where you have your datafiles so you do not have to do it every time. In the menu go to Tools->Options

Using scripts



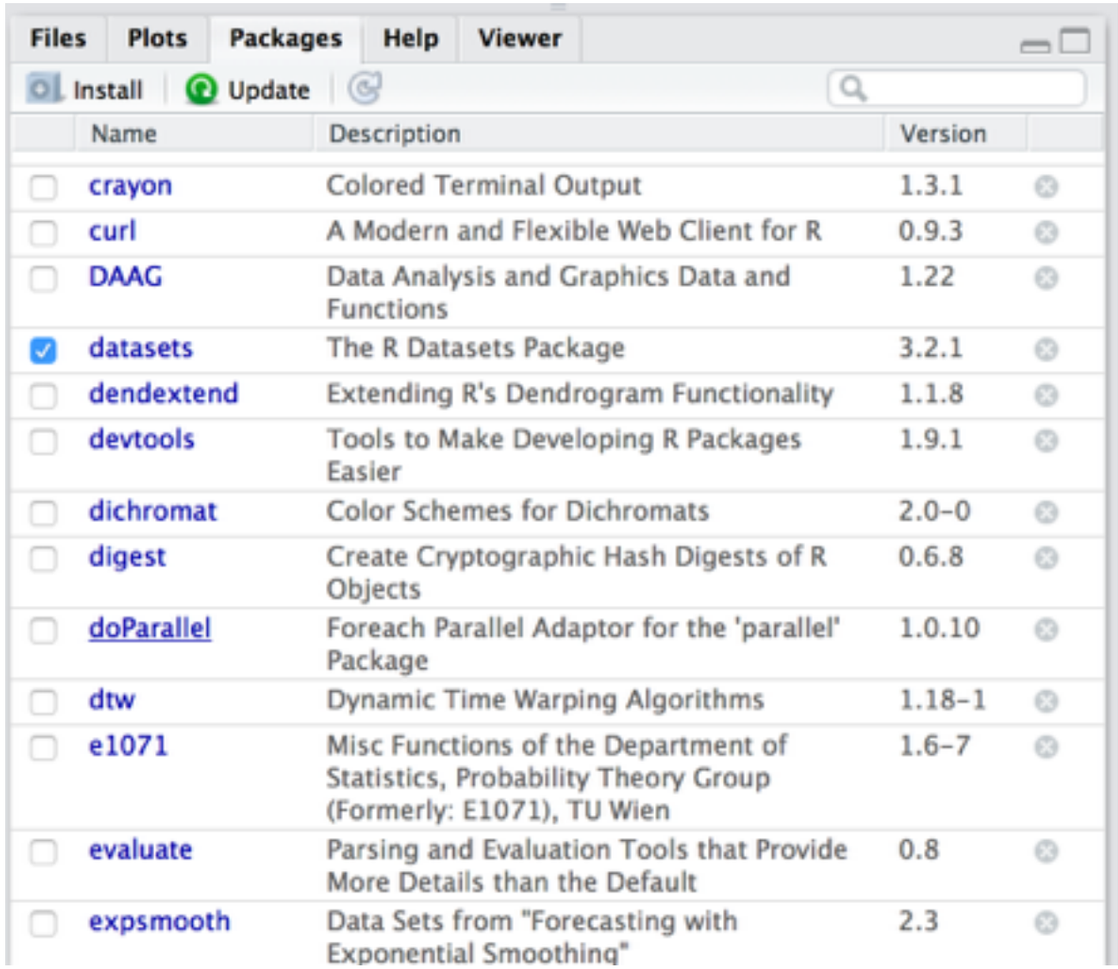
Using scripts

To create a new R script you can either go to `File -> New -> R Script`, or click on the icon with the "+" sign and select "R Script", or simply press Ctrl+Shift+N. Make sure to save the script.

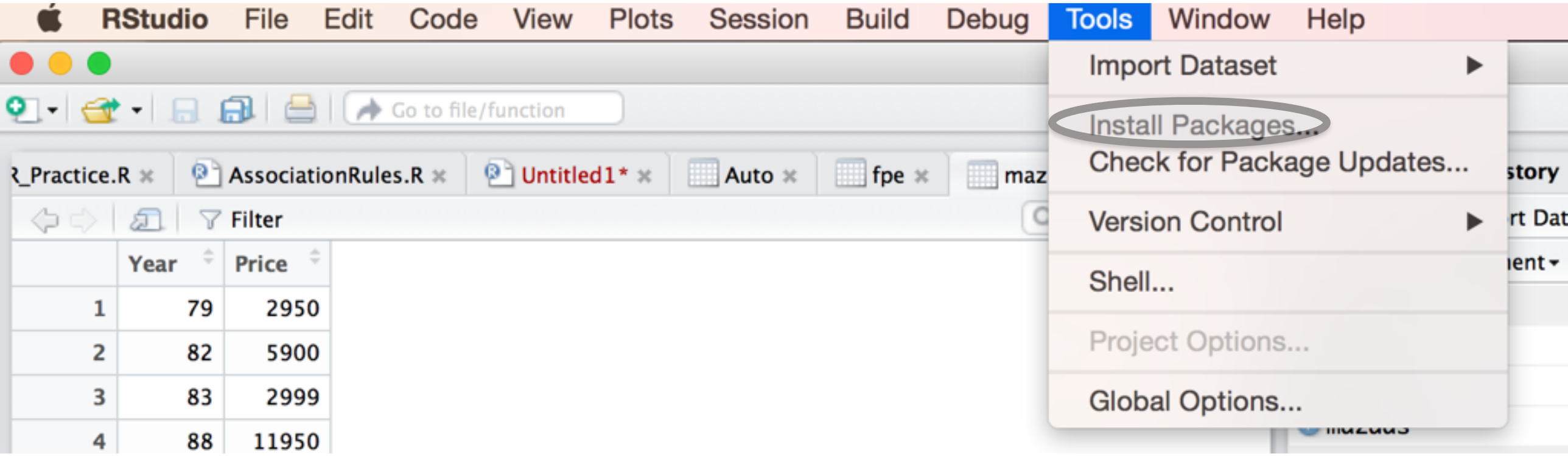


Packages

The package tab shows the list of add-ons included in the installation of RStudio. If checked, the package is loaded into R, if not, any command related to that package won't work, you will need select it. You can also install other add-ons by clicking on the 'Install Packages' icon. Another way to activate a package is by typing, for example, library(e1071). This will automatically check the package.



Installing packages



Install packages ISLR, MASS, e1071

Plots

RStudio

File Edit Code View Plots Session Project Build Tools Help

Go to file/function — Project: (None)

HousePets.R × | MyRscript.R × | house.pets × | A × | B × | Graphs.R* ×

Source on Save | Run | Source

```
1
2  library(car)  # By John Fox and Sanford Weisberg
3  library(rgl)  # By Daniel Adler and Duncan Murdoch
4
5  # Scatterplot per group
6
7  scatterplot(prestige ~ income|type, boxplots=FALSE, span=0.75, data=Prestige)
8
9  # Scatterplots in matrix form
10
11 scatterplotMatrix(~ prestige + income + education, span=0.7,data=Prestige)
12
13 # 3D graph, scatter3d is from the --car package. It will open in a separate window.
14
15 scatter3d(prestige ~ income + education, id.n=3, data=Duncan)
```

11:7 (Top Level) R Script

Workspace | History

Import Dataset

Data	
A	4x2 double matrix
B	4x2 double matrix
house.pets	3 obs. of 4 variables
Values	
feed	character[3]
pets	character[3]
run	numeric[3]
weight	numeric[3]

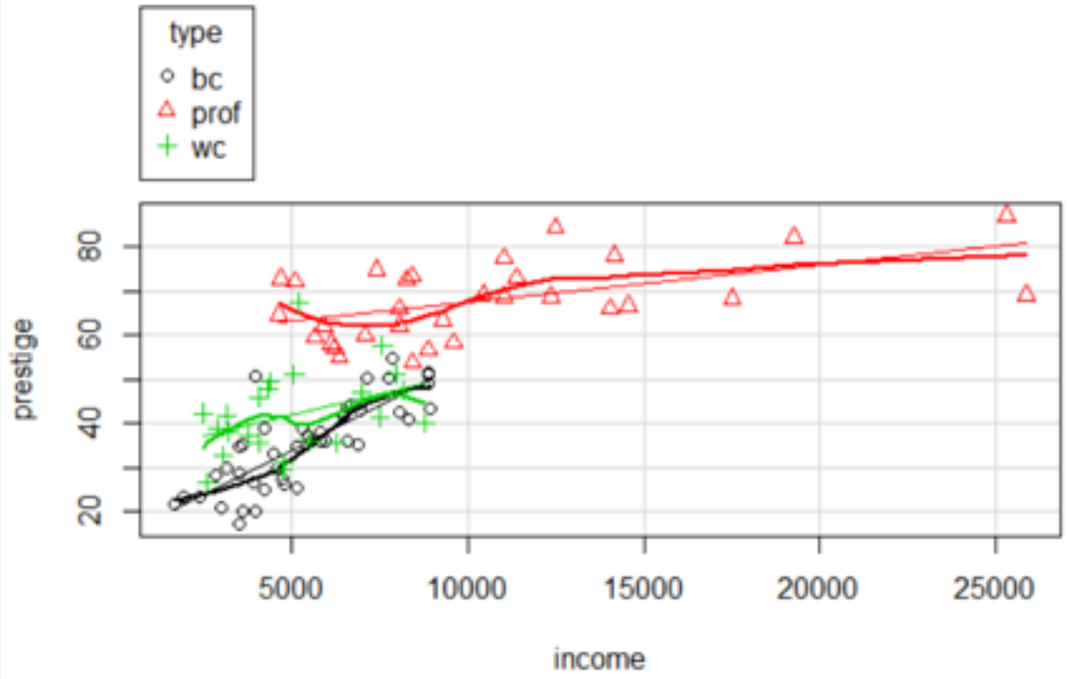
Files | Plots | Packages | Help

Zoom | Export | Clear All

Console H:/MyData/RFiles/

```
> scatterplot(prestige~income|type, boxplots=FALSE, span=0.75, data=Prestige)
>
```

The **plots** tab will display the graphs.
The one shown here is created by the command on line 7 in the script above.
See next slide to see what happens when you have more than one graph



Plots

RStudio

File Edit Code View Plots Session Project Build Tools Help

Go to file/function

Project: (None)

HousePets.R | MyRscript.R | house.pets | A | B | Graphs.R*

Source on Save | Run | Source

```
1
2  library(car)  # By John Fox and Sanford Weisberg
3  library(rgl)  # By Daniel Adler and Duncan Murdoch
4
5  # Scatterplot per group
6
7  scatterplot(prestige ~ income|type, boxplots=FALSE, span=0.75, data=Prestige)
8
9  # Scatterplots in matrix form
10
11 scatterplotMatrix(~ prestige + income + education, span=0.7,data=Prestige)
12
13 # 3D graph, scatter3d is from the --car package. It will open in a separate window.
14
15 scatter3d(prestige ~ income + education, id.n=3, data=Duncan)
```

13:1 (Top Level) R Script

Console H:/MyData/RFiles/

```
> scatterplot(prestige~income|type, boxplots=FALSE, span=0.75, data=Prestige)
> scatterplotMatrix(~ prestige + income + education, span=0.7,data=Prestige)
>
```

Here there is a second graph (see line 11 above). If you want to see the first one, click on the left-arrow icon.

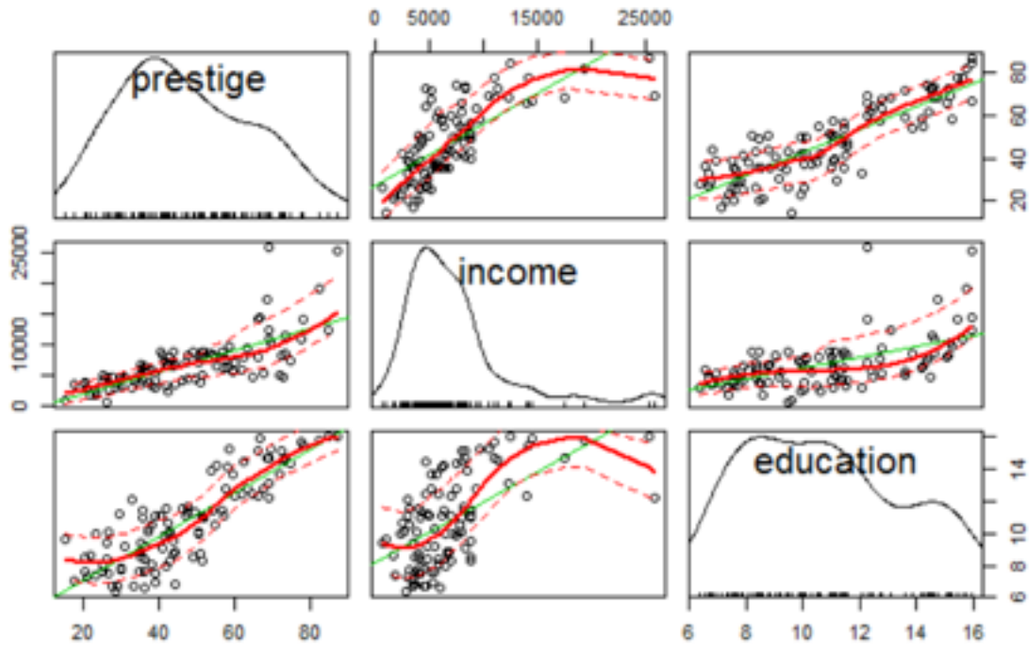
Workspace | History

Import Dataset

Data	
A	4x2 double matrix
B	4x2 double matrix
house.pets	3 obs. of 4 variables
Values	
feed	character[3]
pets	character[3]
run	numeric[3]
weight	numeric[3]

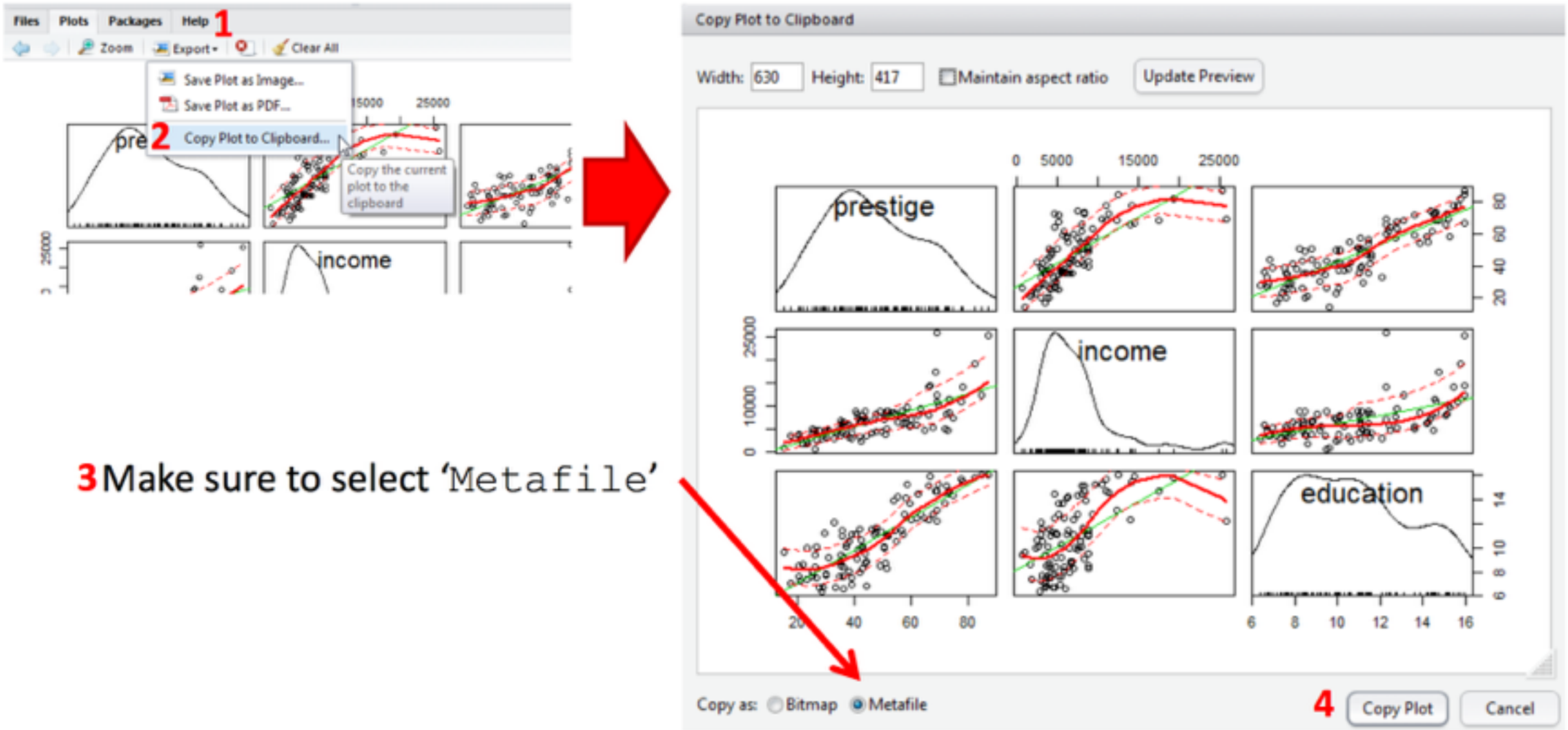
Files | Plots | Packages | Help

Zoom | Export | Clear All



Plots

To extract the graph, click on "`Export`" where you can save the file as an image (PNG, JPG, etc.) or as PDF, these options are useful when you only want to share the graph or use it in a LaTeX document. Probably, the easiest way to export a graph is by copying it to the clipboard and then paste it directly into your Word document.



3 Make sure to select '`Metafile`'

Arithmetic with R

```
#Try the following commands in R Console

# An addition
5 + 5

# A subtraction
5 - 3

# A multiplication
3 * 5

# A division
(5 + 5) / 2

# Exponentiation
2^4

# Modulo
35%%4
```

Basic data types in R

1. Logical
2. Numeric
3. Integer
4. Complex
5. Character

```
my_logical <- FALSE
class(my_logical)

my_numeric <- 42
class(my_numeric)

my_integer <- 3L
class(my_integer)

my_complex <- 2+3i
class(my_complex)

my_character <- "universe"
class(my_character)
```

Expressions and Assignments

The standard arithmetic operators are +, -, *, and / for add, subtract, multiply and divide, and ^ for exponentiation, so 2^5=32. These operators have the standard precedence, with exponentiation highest and addition/subtraction lowest, but you can always control the order of evaluation with parentheses.

You can use mathematical functions, such as sqrt, exp, and log.

Note: R is a case sensitive language. mydata, MyData, and myData are three different objects!

Expressions and Assignments

```
####----Variable Assignment---######
#method 1
x <- 45

#method 2
x = 45

#To view the variable
x

#Example code
my_quarters = 4
my_dimes <- 3
my_coins = my_quarters + my_dimes
my_coins

pi
log(0.3/(1-0.3))
s <- pi/sqrt(3)
```

Expressions and Assignments

```
####----Variable Assignment---######
#method 1
x <- 45

#method 2
x = 45

#To view the variable
x

#Example code
my_quarters = 4
my_dimes <- 3
my_coins = my_quarters + my_dimes
my_coins

pi
log(0.3/(1-0.3))
s <- pi/sqrt(3)
```

Common types of objects

Commonly used R Objects are:

- **Vectors** : has 1 dimension and one type of data
- **Lists** : is a collection of different objects.
- **Matrices** : has 2 dimensions and one type of data
- **Factors** : helps to classify distinct values of the elements in the vector as labels/ levels.
- **Data Frames** : set of data with different data types.

Vectors

Vector Creation

```
####---creating Vector---###
numeric_vector <- c(1, 2, 3)
numeric_vector
character_vector <- c("a", "b", "c")
character_vector
boolean_vector <- c(T,F,T)
boolean_vector

# outputs a vector with sequence of numbers
seq_num <- 1:10
seq_num
#output the vector with sequence of number using start, end and increment parameters
seq(0,1,0.1)

seq(1,3,length=5)
```

Vectors

Naming a Vector

```
#method 1
max_temp <- c(74,71,73,75,67,71,69)
max_temp
days <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday" ,"Saturday", "Sunday")
names(max_temp) <- days
max_temp

#method 2
min_temp <- c("Monday" = 53, "Tuesday" = 52, "Wednesday" = 53, "Thursday" = 61, "Friday"
= 57, "Saturday"=59,"Sunday"=56)
min_temp
```

Vectors

Vector Selection

```
#select using index
max_temp[3]
max_temp[1:3]
max_temp[c(1,3)]
max_temp[-c(1,4)] #removing index 1 and 4

#select using name
max_temp["Wednesday"]
max_temp[c("Tuesday","Friday")]

#select using condition check
#select all the days where temperature less than 70
max_temp[max_temp<70]
```

Vectors

Vector Continued…

```
x=1:10

#select elements < 5
x[x<5]

# Some Useful Features and Functions
x <- c(3,4,7)
x[-3]
x[x >=5]
x+3

z <- c(1.5, 1/6,1/3)

# View only the first two decimal places of z:
round(z,2)

sort(z)
length(z)
max(z)
mean(z)
```

Vectors

Vector Continued…

```
x=1:100

# To randomly sample from an existing vector:
sample(x,10,replace=T)

# Or to randomly sample from a sequence of numbers from 1 to 500:
sample(1:500,10,replace=F)
```

```
# Missing values are represented by NA:
w = c(3,4,NA,5)

# Some functions ignore NA's:
sort(w)

# Other function, by default, do not:
mean(w)

# Two ways to determine the mean of the non-missing values are:
mean(w,na.rm=T)

w=na.omit(w)
mean(w)
```

Lists

Creating Lists

```
# Numeric vector: 1 up to 10
my_vector1 <- 1:10

#character vector
my_vector2 <- c('a','b','c','d','e','f')

#matrix
my_matrix <- matrix(1:9, ncol = 3)

#creating list

my_list <- list(my_vector1,my_vector2,my_matrix)
my_list
str(my_list)
```

Lists

Naming Lists

```
#naming list

#method 1
list.info <-c("num.vector","char.vector","matrix")
names(my_list) <- list.info
str(my_list)



#method 2
my_list2 <- list("num.vector"=my_vector1,"char.vector"=my_vector2, "matrix" =
my_matrix)
str(my_list2)
```

Lists

Indexing Lists

```
#list indexing
#selecting 1st element
my_list[1]
my_list["num.vector"]
my_list$num.vector

#selecting 1st and 3rd element
my_list[c(1,3)] #vector approach
my_list[c("num.vector","matrix")] #name approach
my_list[c(T,F,T)] # logical approach

#selcting third char from char.vector
#method 1 - using index
my_list[[2]][[3]]
#method 2 - using dollar sign
my_list$char.vector[3]
#method 3 - using name
my_list[["char.vector"]][3]
```

Matrix

Creating a Matrix

```
matrix(1:15) #column matrix
matrix(1:15,1) #row matrix
matrix(1:15,nrow = 5)
matrix(1:15,nrow = 5,byrow = T)
matrix(sample(1:50,40),nrow = 4)
matrix(c(1,2,3,4,5)) #only one col created



max_temp <- c(74,71,73,75,67,71,69)
days <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday" ,"Saturday", "Sunday")
rain <- c(F,F,F,F,T,F,T)
weather.Col <- cbind(days,max_temp,rain)
weather.Col
weather.Row <- rbind(days,max_temp,rain)
weather.Row
```

Matrix

Naming a Matrix

```
max_temp <- c(74,71,73,75,67,71,69)
days <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday" ,"Saturday", "Sunday")
rain <- c(F,F,F,F,T,F,T)

#Matrix naming - method 1
weather <- cbind("col1"=days,"col2"=max_temp,"col3"=rain)
weather
#Matrix naming - method 2#
rownames(weather) <- c( "row1","row2","row3","row4","row5","row6","row7")
colnames(weather)  <- c("days","Temperature","Rainfall")
weather

# Matrix naming - method 3
matrix(sample(1:100,12),byrow=T,nrow=4,ncol=3,
dimnames=list(c("R1","R2","R3","R4"),c("C1","C2","C3")))
```

Matrix

Matrix Manipulations

```
#Adding elements to matrix
#adding column
snow <- c(F,F,F,F,F,F,F)
weather.Snow <- cbind(weather,"Snowfall"=snow)
weather.Snow

#adding rows
row8 <- c(DAYS="holiday",TEMPERATURE=65,RAINFALL = F, SnowFall = T)
weather.holiday <- rbind(weather.Snow,row8)
weather.holiday
```

Matrix Subsetting

```
weather.holiday[13] #selecting 13th entry
weather.holiday[3,c(2,4)]  #row3 + col 2 and 4
weather.holiday[c(1,5),2]  #col2 + row 1 and 5
weather.holiday[c(1,5),c(2,4)]   #submatrix
weather.holiday[2:5,1:3]
```

Factor

Creating Factor

```
#create a vector of speed details
speed.vector <- c("High", "Slow", "High", "Extreme","Slow", "Medium")
#display the vector
str(speed.vector)
summary(speed.vector)
#converts the vector to factor levels

#method 1 - R automatically assigns the levels without any order

speed.factor <- factor(speed.vector)
str(speed.factor)
summary(speed.factor)
#method 2 - converting into factors specifying the order
speed.factor.order <- factor(speed.vector,ordered = TRUE, levels =
c("Slow","Medium","High","Extreme"))
str(speed.factor.order)
summary(speed.factor.order)
```

Factor

Naming a Factor levels using labels

```
#a vector with direction details
direction.vector <- c("R", "L", "L", "R", "R")
#conversion to factor without giving names
direction.factor.unnamed <- factor(direction.vector)
str(direction.factor.unnamed)
summary(direction.factor.unnamed)


#naming the factor levels
direction.factor <- factor(direction.vector, levels = c("R","L"), labels = c("Right","Left"))
direction.factor
str(direction.factor)
summary(direction.factor)
```

Dataframe

Creating a Dataframe

```
continents <- c("Asia", "Africa", "North America", "South America", "Antarctica",
"Europe","Australia")
area <- c(16920000,11730000,9460000,6890000,5300000,3930000,3478200)
size <- c("Large","Large","Large","Medium","Medium","Small","Small")
belowEq <- c(F,T,F,T,T,F,T)

#creating dataframe
continents.df <- data.frame(continents,area,size,belowEq)
continents.df
str(continents.df)
```

Naming a Dataframe

```
#naming a dataframe
names(continents.df) = c("Name","Area(mi-
sq)","Size","BelowEquator")
continents.df
```

Dataframe

Dataframe subsetting

```
#selects all row, col 1
continents.df[1]

#selects row 3 with all columns
continents.df[3,]

#selects row 1 and col 3
continents.df[1,3]

#selecting multiple rows with all columns
continents.df[c(1,6),]

#selecting multiple columns with all rows
continents.df[,c(1,3)]

#selecting multiple rows and columns
continents.df[c(1,6),c(1,3)]
```

Dataframe

Dataframe subsetting

```
#select columns using names

#Select the column  “continents” from the dataframe
continents.df$Name

#select the 3rd item from the column “continents”
continents.df$Name[3]

#select using condition check
#select details of  the large continents
continents.df[continents.df$Size=="Large",]

#display the  name of the continents which has land below equator
continents.df[continents.df$BelowEquator=="TRUE","Name"]
```

Subsets of Dataframes

Exercise 1

(a) Use head() to check the names of the columns, and the first few rows of data, in the data frame rainforest (DAAG).
(b) Use table(rainforest$species) to check the names and numbers of each species that are present in the data. The following extracts the rows for the species Acmena smithii

```
> library(DAAG)
> Acmena <- subset(rainforest, species=="Acmena smithii")
```

The following extracts the rows for the species Acacia mabellae and Acmena smithii

```
> AcSpecies <- subset(rainforest, species %in% c("Acacia mabellae", "Acmena smithii"))
```

(c) Extract the rows for all species except C. fraseri.

```
>
```

Exercise 2

Use head() to check the names of the columns, and the first few rows of data, in the data frame ais (DAAG).
Use summary() to get an idea about data

Extract the following subsets from the data frame ais (DAAG):
(a) Extract the data for the rowers.
(b) Extract the data for the rowers, the netballers and the tennis players.
(c) Extract the data for the female basketabllers and rowers.
(Hint: col1 == "x" & col2 == " y")

Reading Data

	setting	effort	change
Bolivia	46	0	1
Brazil	74	0	10
Chile	89	16	29
Colombia	77	16	25
CostaRica	84	21	29
Cuba	89	15	40
DominicanRep	68	14	21
Ecuador	70	6	0
ElSalvador	60	13	13
Guatemala	55	9	4
Haiti	35	3	0
Honduras	51	7	7
Jamaica	87	23	21
Mexico	83	4	9
Nicaragua	68	0	7
Panama	84	19	22
Paraguay	74	3	6
Peru	73	0	2
TrinidadTobago	84	15	29
Venezuela	91	7	11

This small dataset includes an index of social setting, an index of family planning effort, and the percent decline in the crude birth rate between 1965 and 1975. The data are available on the web at http://data.princeton.edu/wws509/datasets/ in a file called `effort.dat` which includes a header with the variable names.

Reading Data

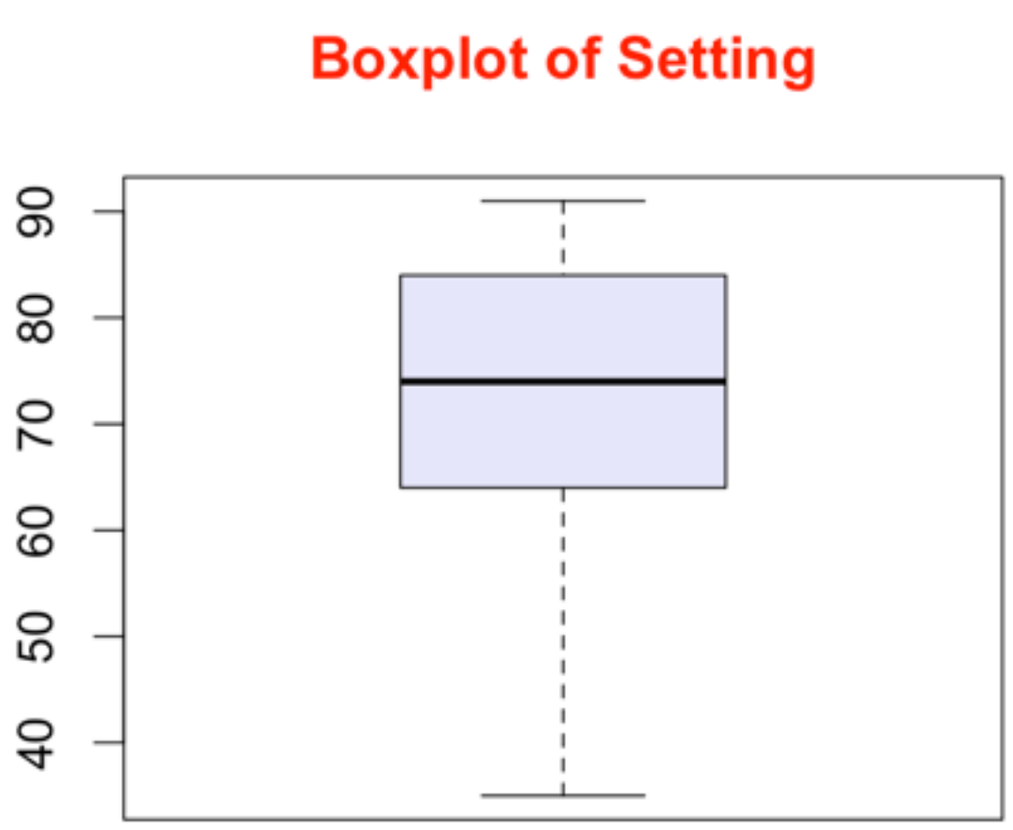
```
#Read data from web url
fpe <- read.table(“http://data.princeton.edu/wws509/datasets/effort.dat")

#view dataset
fpe

#view column change
fpe$change

#view summary of data
summary(fpe)

mean(fpe$effort)

#List countries with effort = 0
fpe[fpe$effort == 0,]

#List the record for the country "Chile"
fpe["Chile",]
```

- Exercise: List the countries where social setting is high (say above 80) but effort is low (say below 10)? Hint: logical operator & can be used for “and” operation.

Plotting

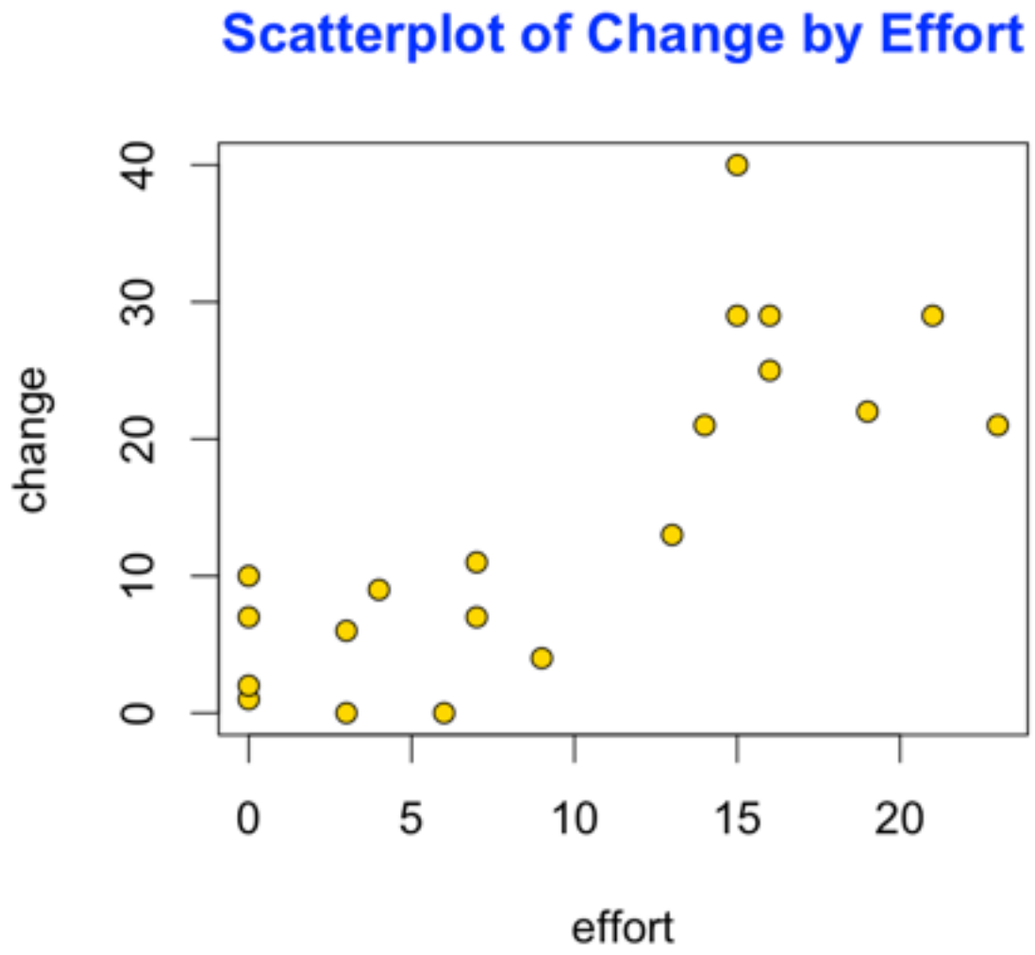
boxplot(setting, col="lavender")

title("Boxplot of Setting", col.main="red")



Plotting

```
plot(effort, change, pch=21, bg="gold")

title("Scatterplot of Change by Effort", col.main="blue")
```



Exercise

Using fpe dataset

Draw box plot for change, setting, and effort

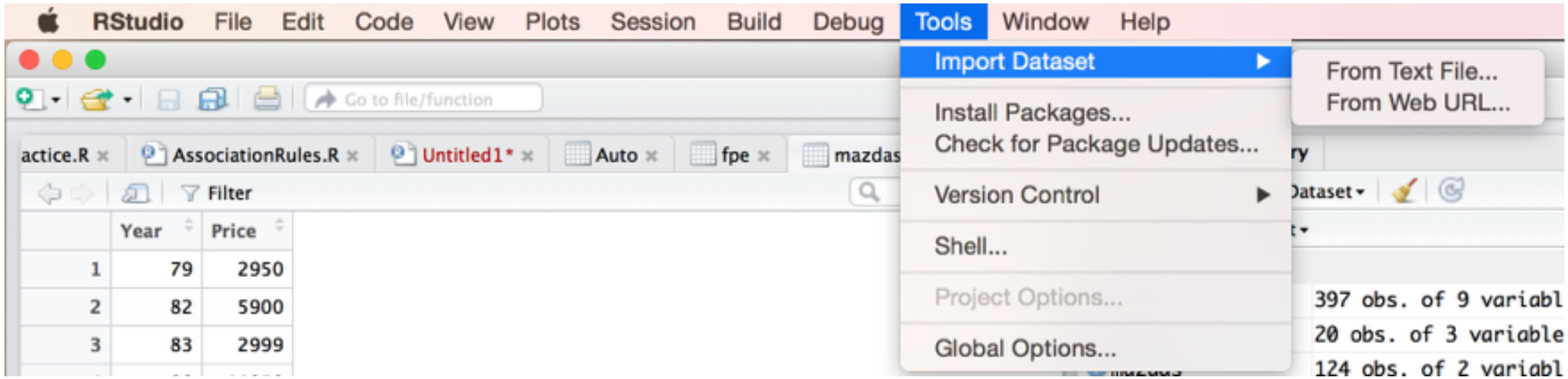
Draw scatter plot between setting and effort

Output Graph

Function	Output to
pdf("mygraph.pdf")	pdf file
win.metafile("mygraph.wmf")	windows metafile
png("mygraph.png")	png file
jpeg("mygraph.jpg")	jpeg file
bmp("mygraph.bmp")	bmp file
postscript("mygraph.ps")	postscript file

```
# output graph to pdf file

myvector <- seq(.5, 36, .5) +  rnorm(72, mean=0, sd=3)

pdf("testplot.pdf")

plot(myvector)

dev.off()
```

How to import data from web URL



Mazda dataset

- Exercise:
- Import mazdas dataset from web URL http://www.statsci.org/data/oz/mazdas.txt
- attach dataset using attach(mazdas) command
- Draw scatter plot between year and price
- Draw boxplots for year and price

Old Faithful dataset

- **Lab**

  Check out faithful dataset

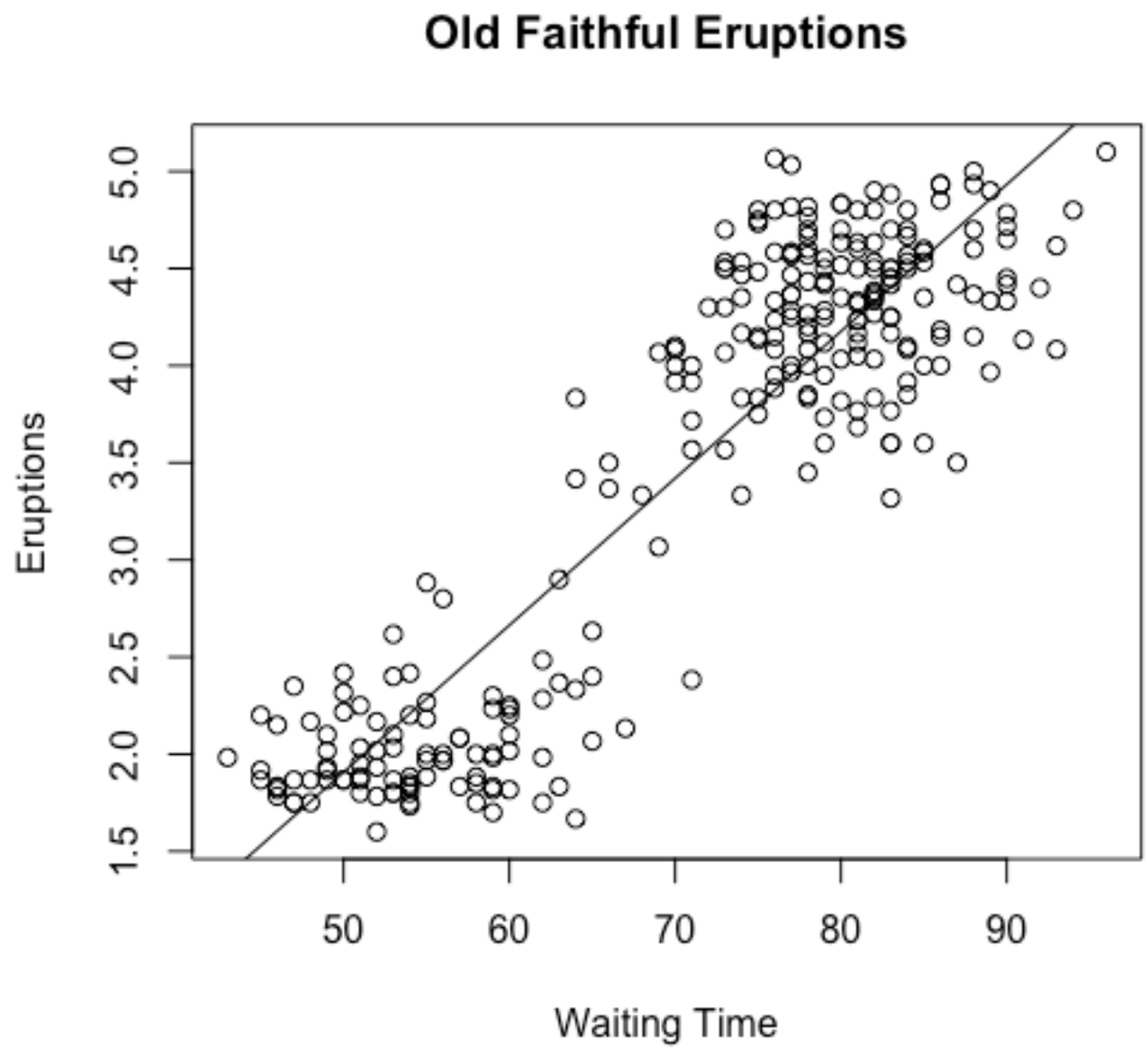
  Draw scatter plot between eruptions and waiting

  Use lm to create a simple linear model; Note down the coefficients

  Draw the fitted line

```
#old faithful
summary(faithful)

plot(faithful$waiting, faithful$eruptions,
          ylab="Eruptions", xlab="Waiting Time", main="Old Faithful Eruptions"
     )

lm.fit = lm(eruptions ~ waiting, data=faithful)

abline(lm.fit)
```

Old Faithful dataset



Hubble Telescope Dataset

- **Lab**

Install the package gamair (from CRAN) and examine the help page for the data frame hubble. Type data(hubble) to bring the data into the workspace. (This is necessary because the gamair package, unlike most other packages, does not use the lazy loading mechanism for data.)

(a) Plot y (Velocity in km sec−1) versus x (Distance in Mega-parsec)

Note: 1 Mega-parsec = 3.09e19 km

(b) Fit a line, omitting the constant term; for this the lm() function call is

lm(y~ -1 + x, data=hubble)

(c) The inverse of the slope (convert distance from Mega-parsec to km) is then the age of the universe in sec. Convert the age to years.

[The answer should be around 13 billion years.]

hubble *Hubble Space Telescope Data*

Description

Data on distances and velocities of 24 galaxies containing Cepheid stars, from the Hubble space telescope key project to measure the Hubble constant.

Usage

```
data(hubble)
```

Format

A data frame with 3 columns and 24 rows. The columns are:

- Galaxy A (factor) label identifying the galaxy.
- y The galaxy's relative velocity in kilometres per second.
- x The galaxy's distance in Mega parsecs. 1 parsec is 3.09e13 km.

Details

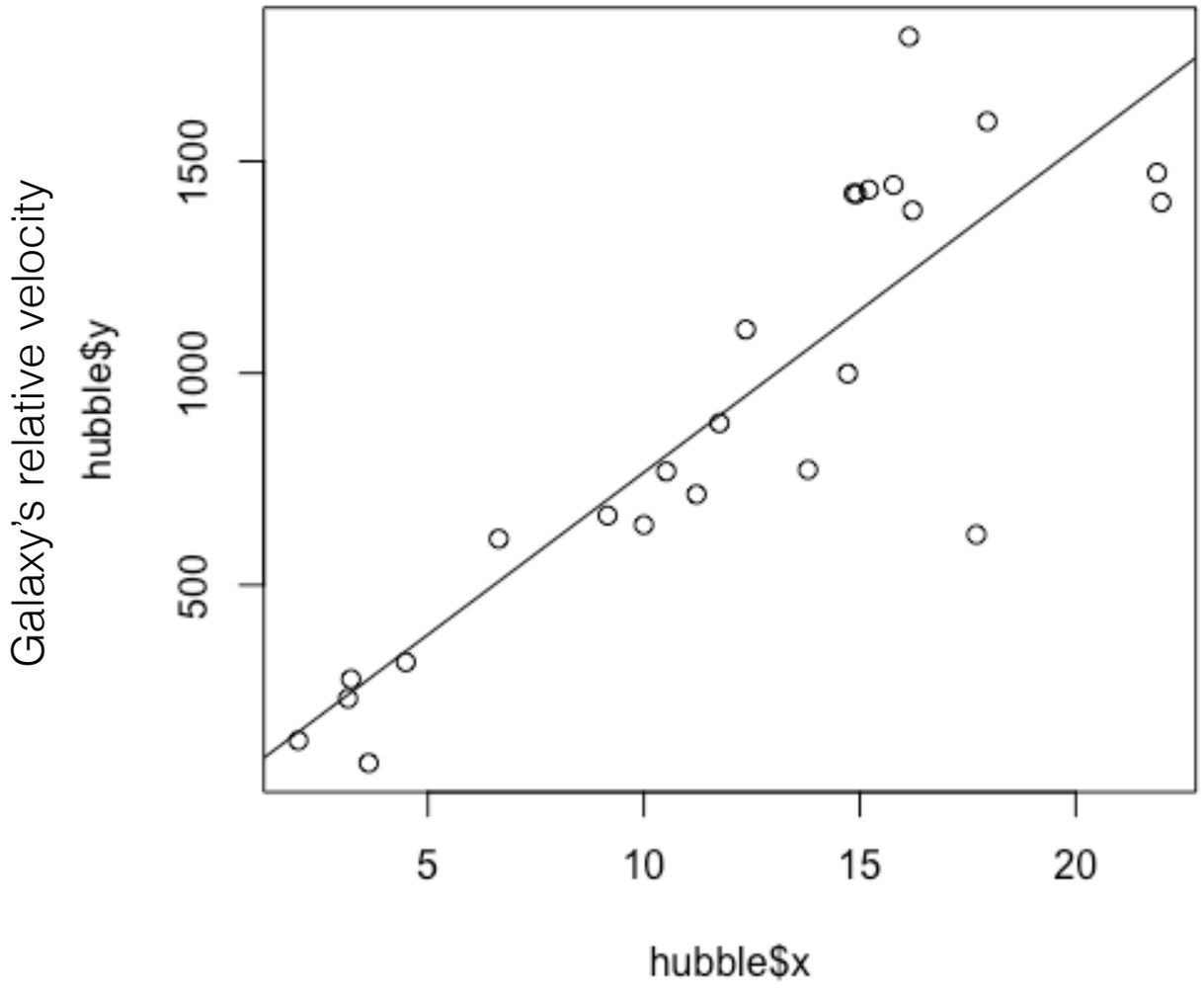
Cepheids are variable stars which have a known relationship between brightness and period. Hence the distance to galaxies containing these stars can be estimated from the observed brightness of the Cepheid, relative to its absolute brightness as predicted by its period. The velocity of the galaxy can be estimated from its mean red-shift.

The data can be used to get a reasonably good idea of the age of the universe A data free alternative estimate of 6000 years is given in the reference (not the source!).

Source

Tables 4 and 5 of Freedman et al. 2001. The Astrophysical Journal 553:47-72

Hubble Telescope Dataset



Galaxy's distance

	Galaxy	y	x
1	NGC0300	133	2.00
2	NGC0925	664	9.16
3	NGC1326A	1794	16.14
4	NGC1365	1594	17.95
5	NGC1425	1473	21.88
6	NGC2403	278	3.22
7	NGC2541	714	11.22
8	NGC2090	882	11.75

Estimate of age of Universe = x / y = 1 / (slope of line)

Hubble Telescope Dataset

```
install.packages('gamair')

library('gamair')
data(hubble)
View(hubble)

lm.model <- lm(y ~ -1 + x, data=hubble)

plot(hubble$x, hubble$y)

slope <- lm.model$coefficients['x']
slope_kmsec <- slope/3.09e19

# apply Mega-parsec to km & sec to year conversations
UniverseAge <- (1/slope_kmsec ) / (60^2 * 24 * 365)

UniverseAge

#try ggplot
require(ggplot2)

ggplot(hubble, aes(x,y)) + geom_point() + geom_text(aes(label=Galaxy))
```

Hubble Telescope Dataset

- **Exercise**

(a) Use lm.model$residuals to analyze the residuals

(b) Use sort(abs(lm.model$residuals), decreasing=TRUE) to list observations with decreasing absolute residuals

(c) Remove the top 2 observations with the high residuals

(Hint: hubble[-c(8,2),] will remove rows 8 and 2 from the dataset)

(d) Redo the lab with the new dataset and estimate the age of universe

If else statement(s)

Expression

```
if(condition) {
expr
}
```

Expression

```
if(condition) {
expr1
} else {
expr2
}
```

Expression

```
if(condition1) {
expr1
} else if(condition2) {
expr2
} else {
expr3
}
```

Implementation

```
x <- -5
if(x < 0) {
print("x is a negative number")
}
```

Implementation

```
x <- -5
if(x < 0) {
print("x is a negative number")
} else {
print("x is either a positive number
or zero")
}
```

Implementation

```
x <- 5
if(x < 0) {
print("x is a negative number")
} else if(x == 0) {
print("x is zero")
} else {
print("x is a positive number")
}
```

ifelse statement

Vector equivalent form of the **if...else** statement in R

Expression

```
ifelse(test_expression,x,y)
```

Implementation

```
x = 1:10
ifelse(x < 6,”low","high")
```

```
> x = 1:10
> ifelse(x < 6,"low","high")
 [1] "low"  "low"  "low"  "low"  "low"  "high" "high" "high" "high"
[10] "high"
```

ifelse statement

- **Lab**

(a) Load Boston dataset

```
library(MASS)
Views(Boston)
?Boston
```

(b) Convert the chas variable to Y and N (from 1 and 0) using ifelse construct

```
Boston$chas = ifelse(Boston$chas == 1, "Y", “N")
Views(Boston)
```

(c) Convert the crim variable to High, Medium, and Low based on

High: >3

Medium: .1 to 3

Low: < .1

Hint: you can nest ifelse statements — ifelse(test_expr, a, ifelse(test_expr1, b,c))

(d) How many records have High crime rate?

```
sum(ifelse(Boston$crim==‘High',1,0))
nrow(Boston[Boston$crim=='High',])
```

while loop

Expression

```
while(condition) {
expr
}
```

Implementation

```
y <- 0
while(y <= 7) {
  print(paste("y is set to", y))
  y <- y + 1
}
```

for loop

Expression

```
for(items in sequence) {
expr
}
```

Implementation

```
continents <- c("Asia", "Africa", "North America", "South America", "Antarctica",
"Europe","Australia")

#method 1
for (items in continents) {
  print(items)
}

#method 2
for (items in 1:length(continents)){
  print(continents[items])
}
```

for loop

- **Lab**

Create 4 time series data and plot them

```
par(mfrow=c(1,1))

colors.list =
c('blue','green','brown','red','black','cyan','yellow','pink','orange','gold')


x = 1:20

plot(NULL, xlim=c(0,20), ylim=c(0,20), ylab="y", xlab="x")

for(d in 1:4){

  y = rnorm(20, 10, 5-1*d)

  lines(x,y,lwd=2,col=colors.list[d])

}
```

Functions

Expression

```
my_fun <- function(arg1, arg2) {
  body
}
```

Implementation

```
tenfold <- function(x){
  y = 10*x
  return(y)
}
tenfold(5)
```

Functions

Expression

```
my_fun <- function(arg1, arg2) {
  body
}
```

Implementation

```
tenfold <- function(x){
  y = 10*x
  return(y)
}
tenfold(5)
```

Functions

sapply()

- Simplify apply
- Used with homogeneous r objects like ‘Vectors’
- Returns the results as vectors
- Automatically names the vector

```
continents <- c("Asia", "Africa", "North America", "South
America", "Antarctica", "Europe","Australia")

# returns as vector with names
sapply(continents,nchar)

#returns as vector without names
sapply(continents,nchar, USE.NAMES = FALSE)
```

More slides

Mazda dataset

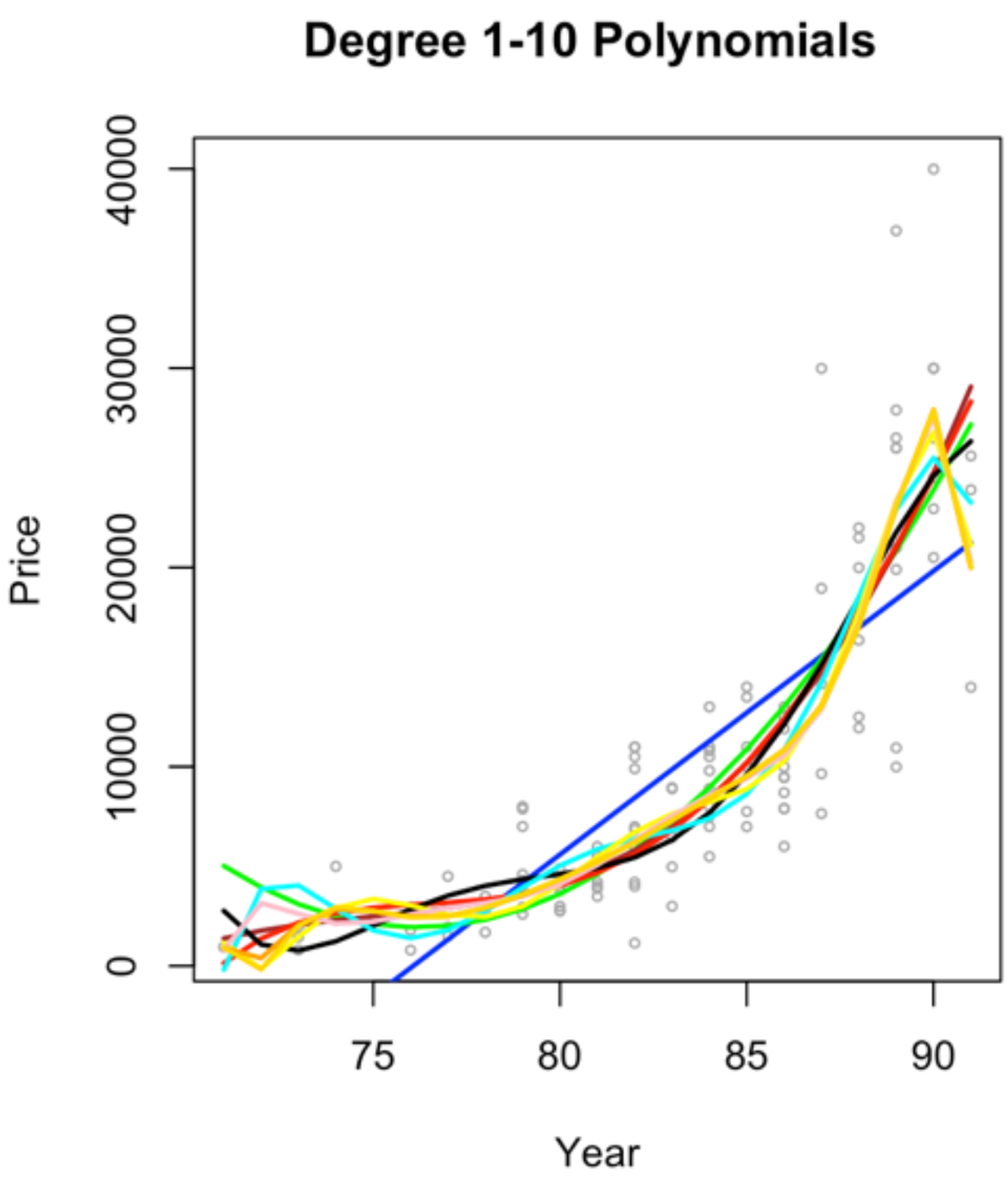
```
summary(mazdas)

plot(mazdas)

attach(mazdas)

# Create two datasets.
# Training dataset:  used to build the model (80% of of the data)
# Test dataset: used to test the model
set.seed(2016)
train=sample(nrow(mazdas), nrow(mazdas)*.8)

mazdas[-train,]

#
# check the results of train and -train
#
# Build linear model
# Response: Price
# Predictor(s): Year
# Price = a0 + a1*Year
# mazdas[train]

lm.fit = lm(Price~Year,data=mazdas, subset=train)
```

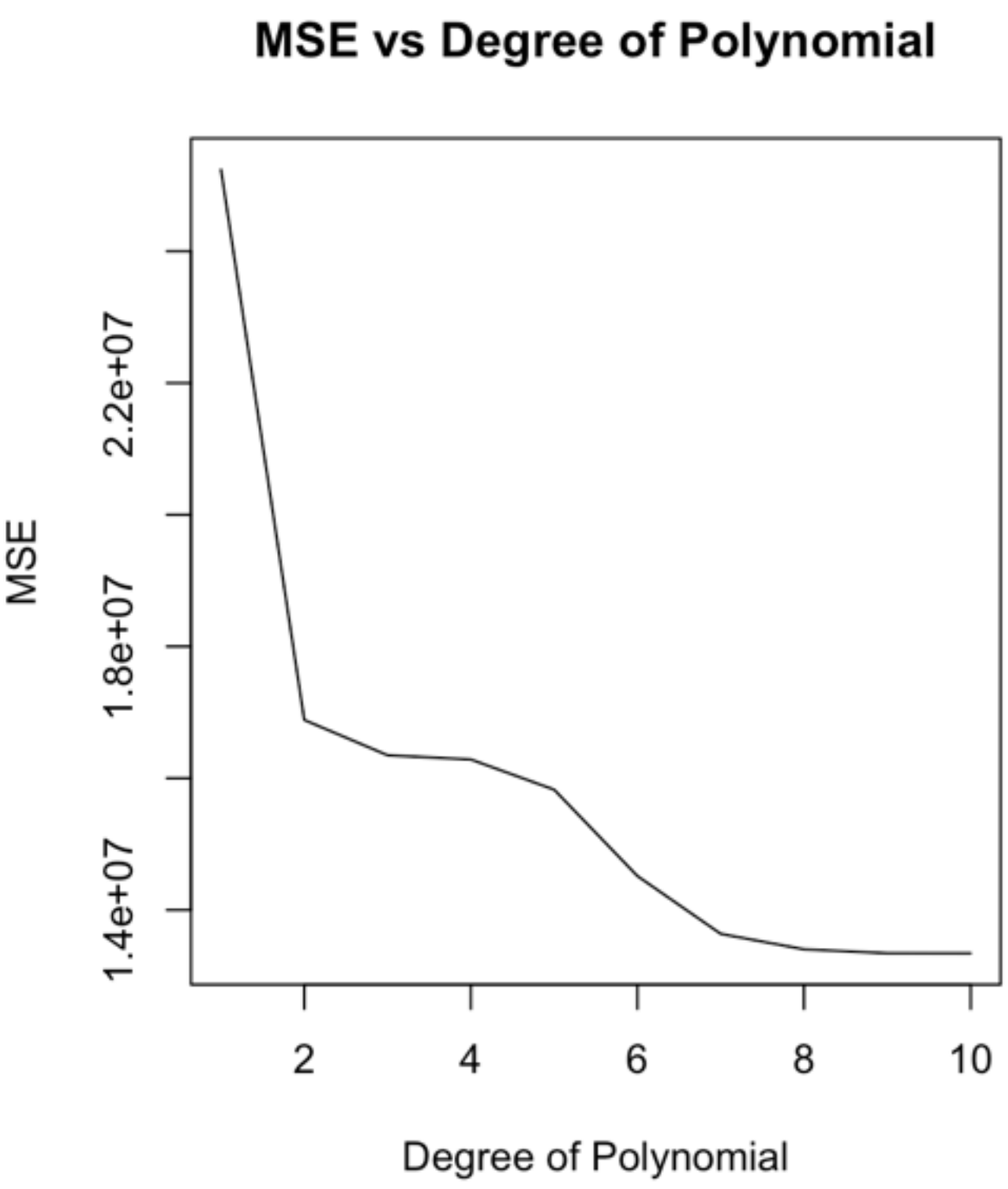
Mazda dataset

```
#
# Check model summary
#

summary(lm.fit)

# Price = -108238.1 + 1422.9*Year

plot(mazdas$Year, mazdas$Price, xlab="Year", ylab="Price", cex=.5, col="darkgrey")

# Predict Sales for the year range (of the data)
seq(range(mazdas$Year)[1], range(mazdas$Year)[2])

Year.range = seq(range(mazdas$Year)[1], range(mazdas$Year)[2])

preds=predict(lm.fit,   newdata=data.frame(Year = Year.range))

# Plot prediction

lines(Year.range,preds,lwd=2,col="blue")

# Check accuracy of the model using test data

ytest = mazdas$Price[-train]

mazdas$Year[-train]

ytestp = predict(lm.fit,newdata=data.frame(Year = mazdas$Year[-train]))

# Mean squared error
mse = rep(0.0, 4)
mse[1] <- mean((ytest - ytestp)^2)
```

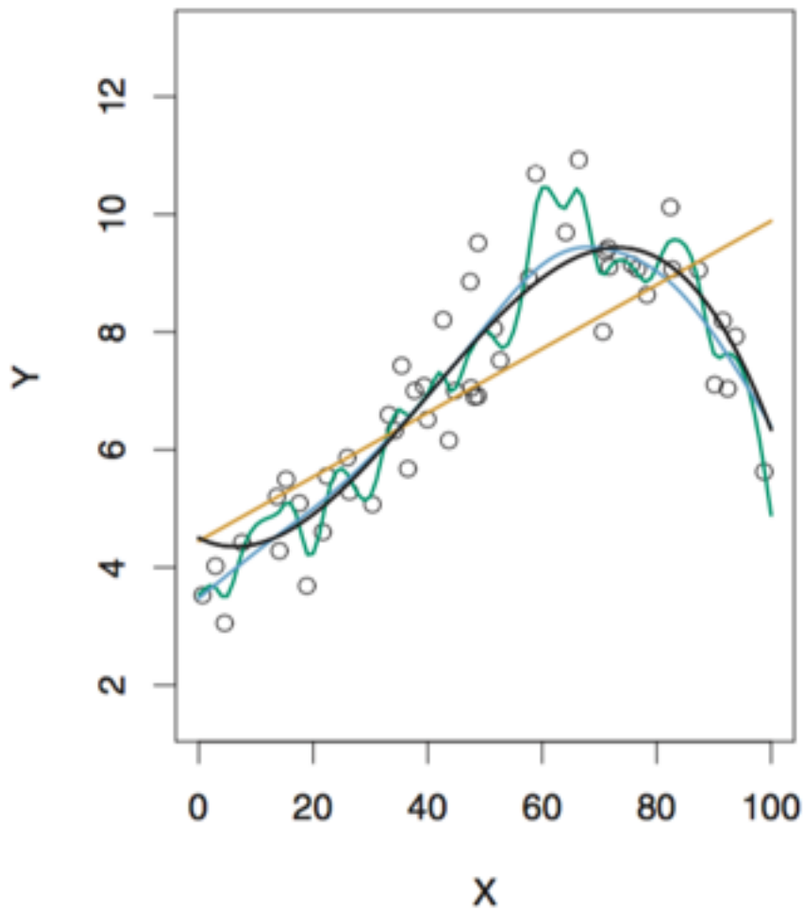
Mazda dataset

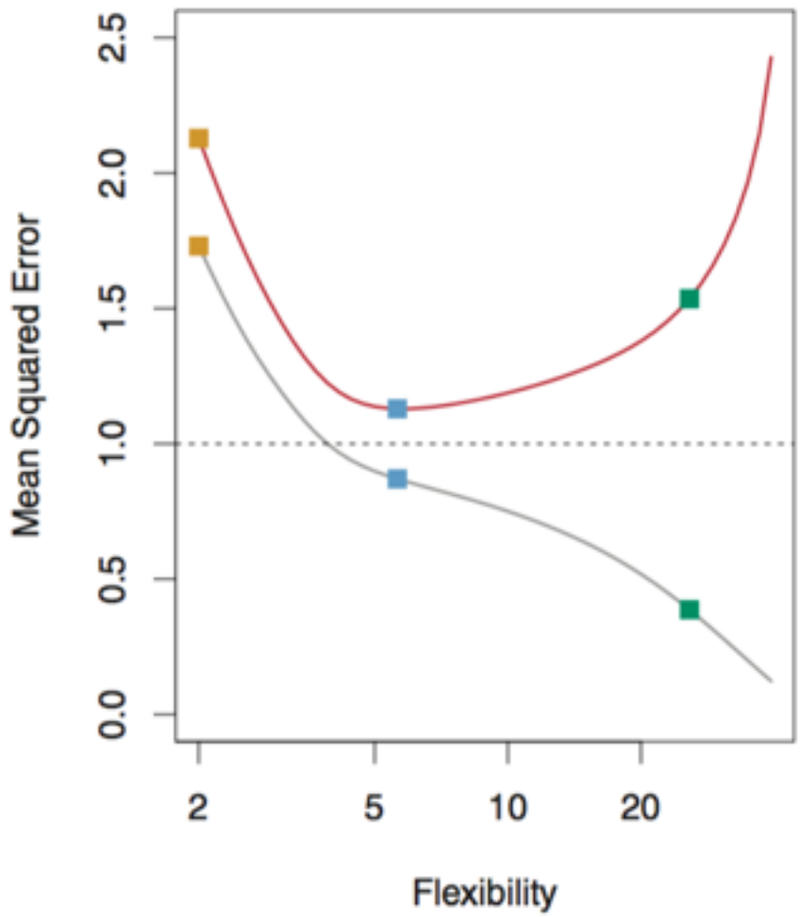
```
par(mfrow=c(2,1))

Year.range = seq(range(mazdas$Year)[1], range(mazdas$Year)[2])

plot(mazdas$Year, mazdas$Price, xlab="Year", ylab="Price", cex=.5, col="darkgrey")

colors.list = c('blue','green','brown','red','black','cyan','yellow','pink','orange','gold')

title("Degree 1-10 Polynomials ")
degree=1:10
mse = rep(0.0, length(degree))
for(d in degree){
  # FIT A MODEL
  lm.fit = lm(mazdas$Price~poly(Year,d),data=mazdas, subset=train)
  # PREDIT USING TEST DATA
  preds=predict(lm.fit, newdata=data.frame(Year = Year.range))
  lines(Year.range,preds,lwd=2,col=colors.list[d])
  # Calculate mean squared error
  mse[d] = mean(lm.fit$residuals^2)
}

mse

plot(mse, xlab="Degree of Polynomial", ylab="MSE", type="l")
title("MSE vs Degree of Polynomial")
```


Degree 1-10 Polynomials
Price
0
10000
20000
30000
40000
75
80
85
90
Year

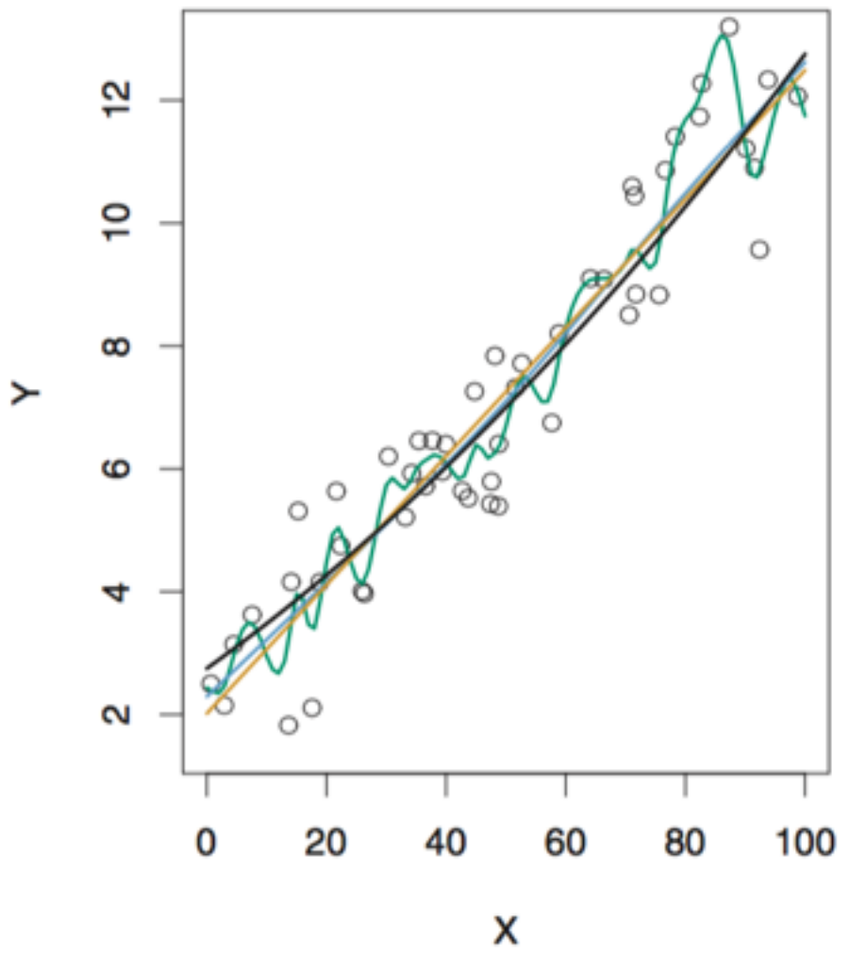

MSE vs Degree of Polynomial
MSE
1.4e+07
1.8e+07
2.2e+07
2
4
6
8
10
Degree of Polynomial

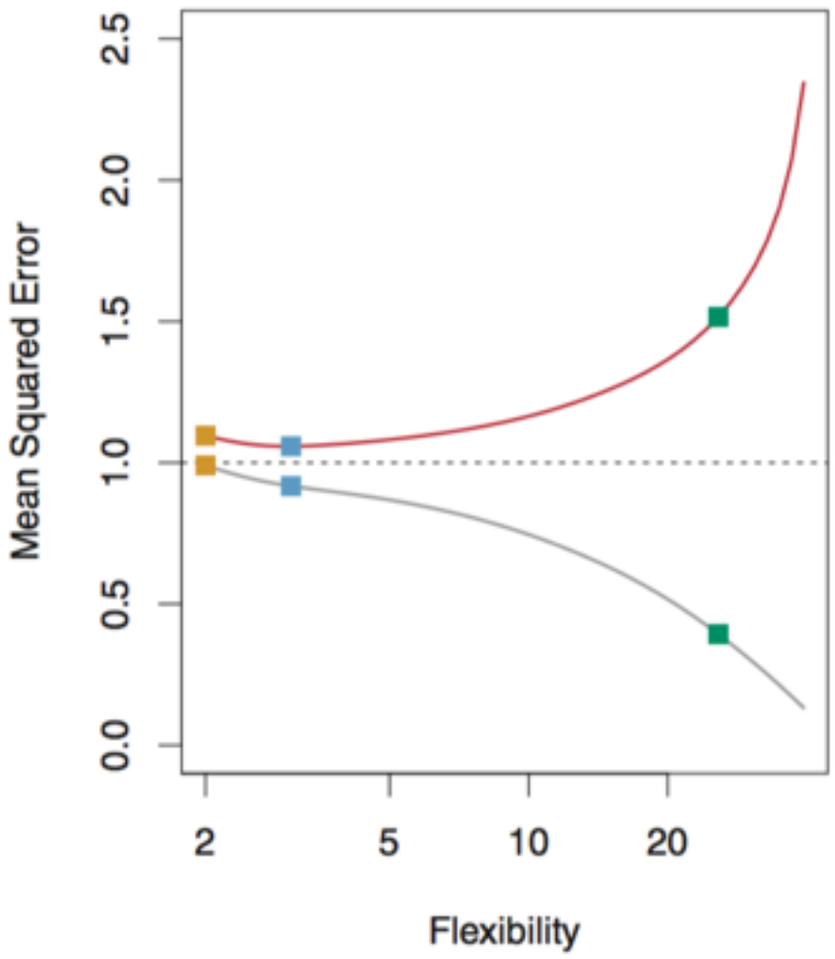
Assessing Model Accuracy



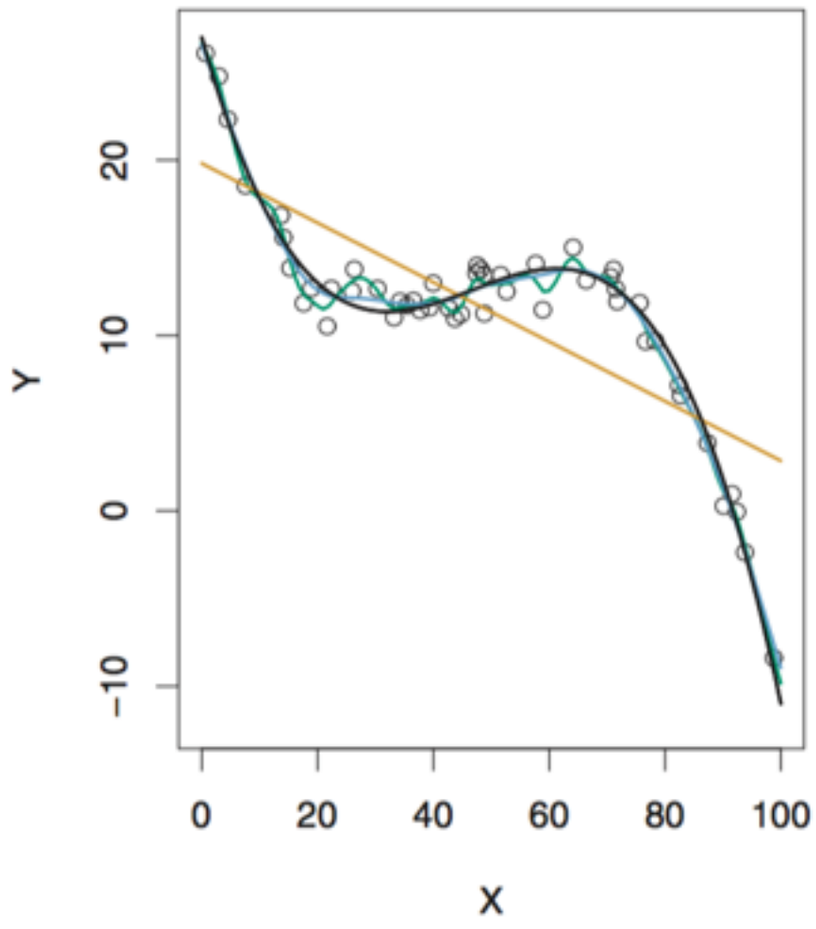


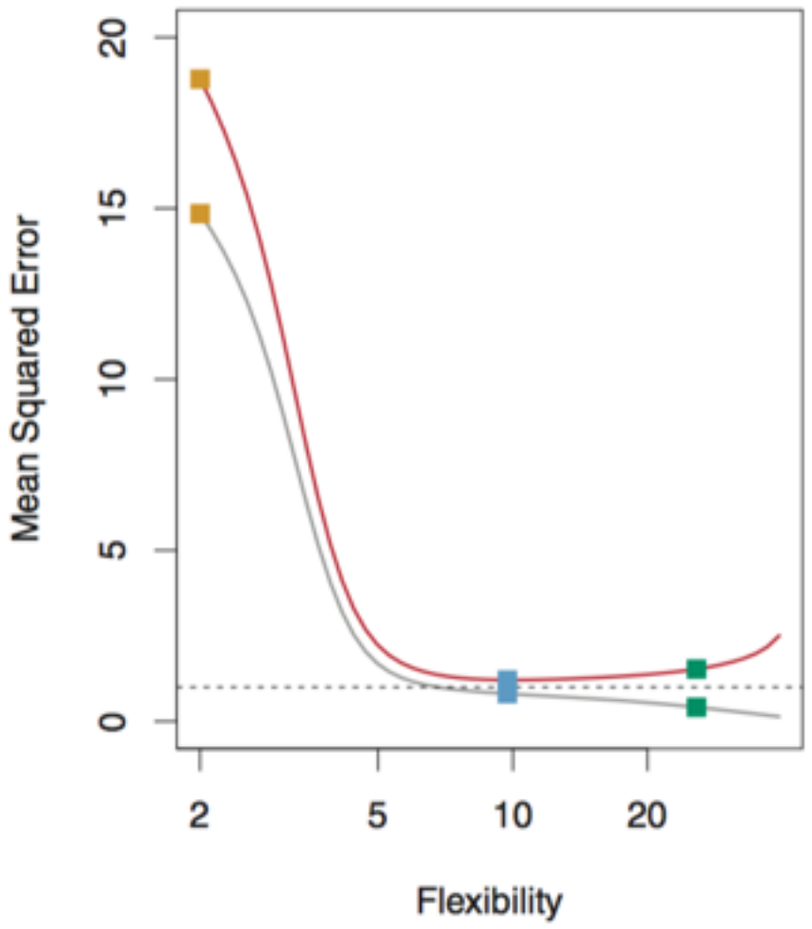
Assessing Model Accuracy



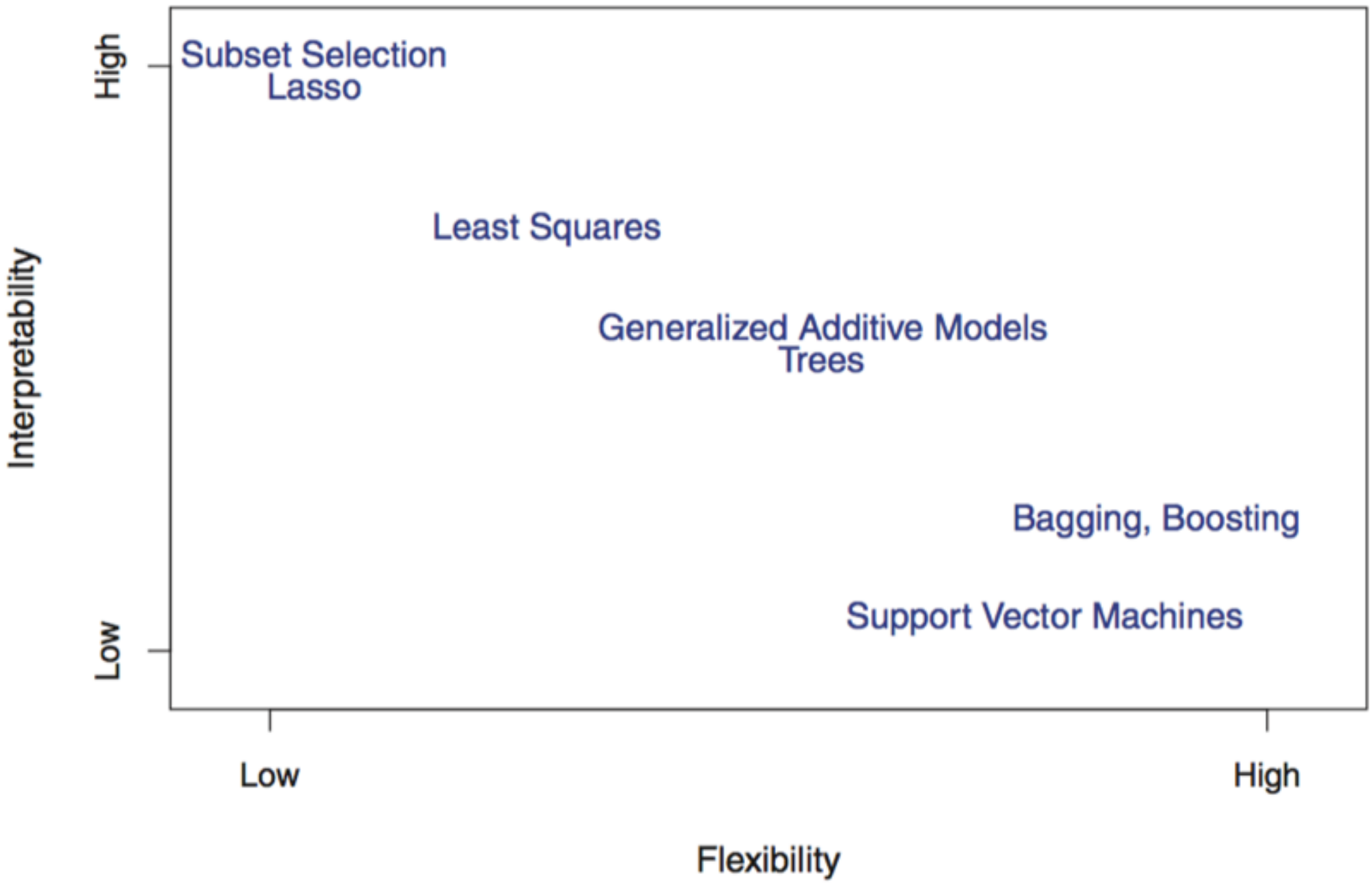


Assessing Model Accuracy





Interpretability vs Flexibility



References

1. An Introduction to Statistical Learning
with Applications in R

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani