Introduction to Statistical Learning

Stephen Jun Villejo

School of Statistics

Overview of Statistical Learning

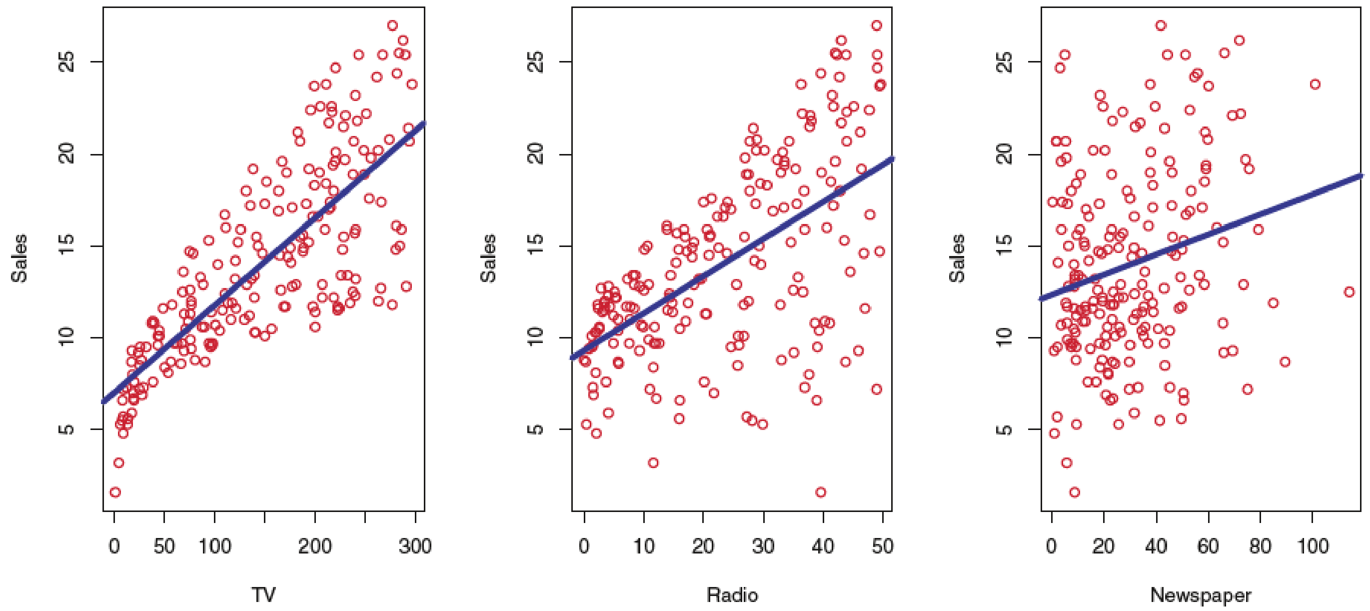
What is *Statistical Learning*?

- Statistical learning refers to a vast set of tools for understanding data.
- These tools can be classified as *supervised* or *unsupervised*.

Overview of Statistical Learning

Example

Suppose we are statistical consultants hired by a client to provide advice on how to improve sales of a particular product. We have dales data of the product in 200 different markets, along with advertising budgets for three different media: TV, radio, and newspaper.



Overview of Statistical Learning

Example

- The advertising budgets are *input variables* while sales is an *output variable*. The input variables are typically denoted using the symbol **X**, a subscript to distinguish them.
- The inputs are also called as *predictors, independent variables, features,* or sometimes just *variables.*
- Output variables are also called *response* or *dependent variable*.

Overview of Statistical Learning

Example

- In a general set-up, we have *p* different predictors, $X_1, X_2, \ldots, X_p$.
- We assume that there is a relationship between Y and $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$, which can be written in a general form

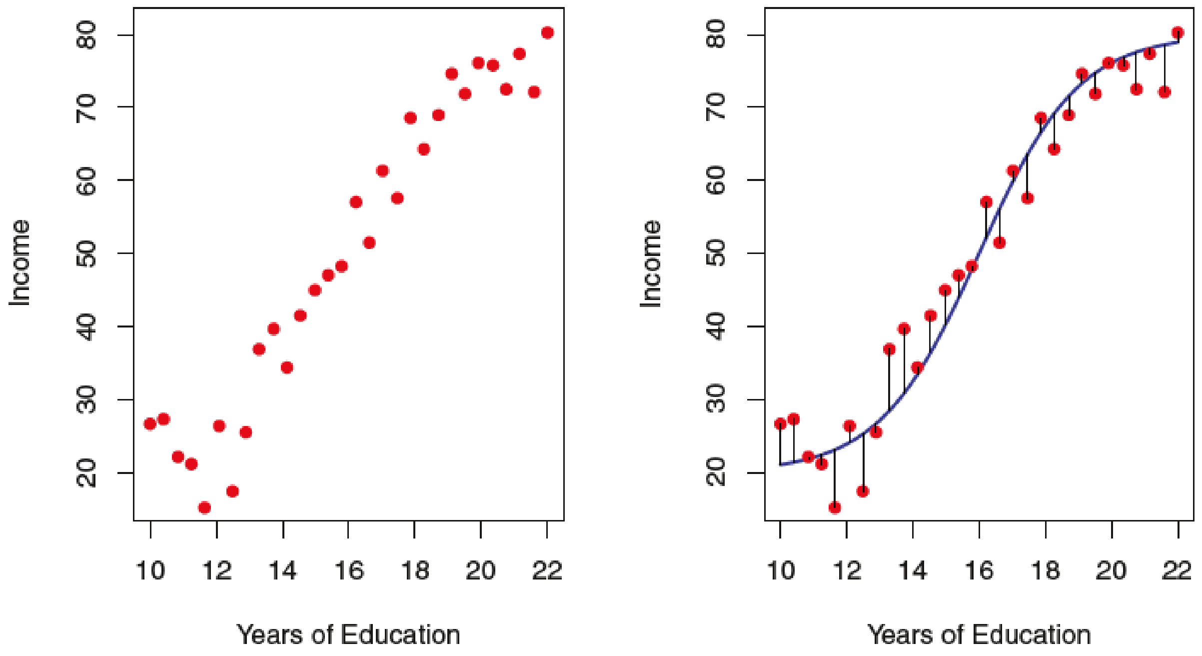
$$Y = f(\boldsymbol{X}) + \varepsilon$$

- $f()$ is some unknown function that represents the *systematic* information that **X** provides about Y.
- ε is a random error term.

Overview of Statistical Learning

Example

- Below, we see the scatter plot between Income and Years od Education.
- We can view f as the systematic part, while the vertical lines are the error terms.



Overview of Statistical Learning

What is *Statistical Learning*?

- In essence, statistical learning refers to a set of approaches for estimating f.

Why estimate f?

- We estimate f for two main reasons: **prediction** and **inference**.

Overview of Statistical Learning

Prediction

- Using our estimate for f which we denote by $\hat{f}$, we obtain the predicted values of Y,

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

- $\hat{f}$ will not be a perfect estimate of f, and this inaccuracy will introduce some error. This error is *reducible* by choosing the most appropriate statistical learning technique.
- Prediction of Y will not be perfect because of the ε which we call the *irreducible error*.

Overview of Statistical Learning

Inference

- Here our goal is not much on predicting Y but on understanding how Y changes as a function of $\boldsymbol{X}$.
- Here, we want to answer the following questions:
  - Which predictors are associated with the response?
  - What is the relationship between the response and each predictor?
  - Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?

Overview of Statistical Learning

What is *Supervised Statistical Learning*?

- Supervised statistical learning involves building a statistical model for predicting, or estimating an output based on one or more inputs.
- Here we wish to fit a model that related the response to the predictors, with the aim of making accurate predictions or understanding relationships between the response and the predictors.
- Examples: linear regression, logistic regression, GAM, boosting, support vector machines

Overview of Statistical Learning

What is *Unsupervised Statistical Learning*?

- With unsupervised statistical learning, there are inputs but no supervising output; nevertheless we can learn relationships and structures from such data.
- Examples: cluster analysis (applied in market segmentation study)

Overview of Statistical Learning

Regression Versus Classification

- We refer to problems with a quantitative response as *regression* problems.
- Problems involving a qualitative response are referred to as *classification* problems.
- We tend to select statistical learning methods on the basis of whether the response is quantitative or qualitative; i.e. we might use linear regression when quantitative and logistic regression when qualitative.

Introduction to R

The R Environment

- **R** is an integrated suite of software facilities for data manipulation, calculation and graphical display.
- It is an environment within which many classical and modern statistical techniques have been implemented.
- A few of these techniques are build into the base R environment, but many are supplied as *packages*.

Introduction to R

The R Environment

- **R** is an *expression language* with a very simple syntax. It is *case sensitive*.
- The entities that R creates and manipulates are known as *objects*. These may be variables, arrays of numbers, character strings, functions, or more general structures built from such components.

Introduction to R

Basic Commands

- **R** uses *functions* to perform operations. To run a function called `funcname`, we type

```
funcname(input1, input2)
```

Example

We use the function `c()` to create a vector of numbers

```
> x <- c(1,3,2,5)
> x
[1] 1 3 2 5
```

Introduction to R

Basic Commands

- Note that the **>** is not part of the command; rather, it is printed by R to indicate that it is ready for another command to be entered.
- We can also save things using = rather than **<-**.

```
> x = c(1,6,2)
> x
[1] 1 6 2
> y = c(1,4,3)
```

Introduction to R

Basic Commands

- Hitting the *up* arrow multiple times will display the previous commands, which then be edited.
- Typing `?funcname` will cause R to open a new help file window with additional information about the function `funcname`.

Introduction to R

Basic Commands

- We can check the length of a vector using the `length()` function.

```
> length(x)
[1] 3
> length(y)
[1] 3
> x+y
[1]  2 10  5
```

Introduction to R

Basic Commands

- The `ls()` function allows us to look at a list of all of the objects, such as data and functions, that we have saved so far. The `rm()` function can be used to delete any that we don't want

```
> ls()
[1] "x" "y"
> rm(x,y)
> ls()
character(0)
```

```
> rm(list=ls())
```

Introduction to R

Basic Commands

- The `ls()` function allows us to look at a list of all of the objects, such as data and functions, that we have saved so far. The `rm()` function can be used to delete any that we don't want

```
> ls()
[1] "x" "y"
> rm(x,y)
> ls()
character(0)
```

```
> rm(list=ls())
```

Introduction to R

Basic Commands

- The `matrix()` can be used to create a matrix of numbers.
- The basic inputs for the function are: the entries, number of rows, and number of columns.

```
> x=matrix(data=c(1,2,3,4), nrow=2, ncol=2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Introduction to R

Basic Commands

- The `byrow=TRUE` option can be used to populate the matrix in order of the rows.

```
> matrix(c(1,2,3,4),2,2,byrow=TRUE)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

- Notice that in the above command, we did not assign the matrix to a value such as `x`. In this case, the matrix is printed to the screen but is not saved for future calculations.

Introduction to R

Basic Commands

- The `sqrt()` function returns the square root of each element of a vector or matrix.
- The command `x^2` raises each element of `x` to the power of 2.

```
> sqrt(x)
     [,1] [,2]
[1,] 1.00 1.73
[2,] 1.41 2.00
> x^2
     [,1] [,2]
[1,]    1    9
[2,]    4   16
```

Introduction to R

Basic Commands

- The `rnorm()` function generates a vector of random standard normal variables, with first argument n the sample size. Each time we call this function, we will get a different answer. We use the set.seed() function to reproduce the exact same set of random numbers.

```
> set.seed(1303)
> rnorm(50)
[1] -1.1440  1.3421  2.1854  0.5364  0.0632  0.5022 -0.0004
. . .
```

- The mean and standard deviation can be altered using the `mean` and `sd` arguments.

Introduction to R

Basic Commands

Example

- Let us create two correlated sets of numbers x and y, and use the `cor()` function to compute the correlation between them.

```
> x=rnorm(50)
> y=x+rnorm(50,mean=50,sd=.1)
> cor(x,y)
[1] 0.995
```

Introduction to R

Basic Commands

- The `mean()` and `var()` functions can be used to compute the mean and variance of a vector of numbers. Applying the `sqrt()` to the out of `var()` will give the standard deviation.

```
> set.seed(3)
> y=rnorm(100)
> mean(y)
[1] 0.0110
> var(y)
[1] 0.7329
> sqrt(var(y))
[1] 0.8561
> sd(y)
[1] 0.8561
```

Introduction to R

Basic Commands

- The `plot()` function is the primary way to plot data in R. For instance `plot(x,y)` produces a scatterplot of the numbers in x versus the numbers in y.

```
> x=rnorm(100)
> y=rnorm(100)
> plot(x,y)
> plot(x,y,xlab="this is the x-axis",ylab="this is the y-axis",
       main="Plot of X vs Y")
```

Introduction to R

Basic Commands

- To save the output of `plot()`, we use `pdf()` to create a pdf and use `jpeg()` to create a jpeg.

```
> pdf("Figure.pdf")
> plot(x,y,col="green")
```

Introduction to R

Basic Commands

- The function `seq()` can be used to create a sequence of numbers. For instance, `seq(a,b)` makes a vector of integers between a and b. `seq(0,1,length=10)` makes a sequence of 10 numbers that are equally spaced between 0 and 1. Typing `3:11` is a shorthand for `seq(3:11)` for integer arguments.

```
> x=seq(1,10)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x=1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x=seq(-pi,pi,length=50)
```

Introduction to R

Basic Commands

- Supposing we have the following data:

```
> A=matrix(1:16,4,4)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

- To get the element in the second row and third column we type

```
> A[2,3]
[1] 10
```

Introduction to R

Basic Commands

- We can also select multiple columns and rows at a time:

```
> A[c(1,3),c(2,4)]
     [,1] [,2]
[1,]    5   13
[2,]    7   15
> A[1:3,2:4]
     [,1] [,2] [,3]
[1,]    5    9   13
[2,]    6   10   14
[3,]    7   11   15
> A[1:2,]
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
```

Introduction to R

Basic Commands

- R treats a single row or column of a matrix as a vector.

```
> A[1,]
[1]  1  5  9 13
```

- The use of a negative sign will keep all rows or columns except those indicated in the index.

```
> A[-c(1,3),]
     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    4    8   12   16
> A[-c(1,3),-c(1,3,4)]
[1] 6 8
```

```
> dim(A)
[1] 4 4
```

Introduction to R

Basic Commands

- To import data into R, the `read.table()` function is one of the primary ways to do this. We use the function `write.table()` to export data.
- An easy way to load an Excel data into R is to save it as a csv file and then using the `read.csv()` function to load it in.

```
> Auto=read.csv("Auto.csv",header=T,na.strings="?")
> fix(Auto)
> dim(Auto)
[1] 397 9
> Auto[1:4,]
```

Introduction to R

Basic Commands

- The **`dim()`** functions tells us the number of observations (rows) and variables (columns) in the data.
- We can use **`names()`** to check the variable names.

```
> names(Auto)
[1] "mpg"          "cylinders"    "displacement" "horsepower"
[5] "weight"       "acceleration" "year"         "origin"
[9] "name"
```

Introduction to R

Basic Commands

- To refer to a variable, we must type the data set and the variable named joined with a $ symbol. For instance, to get the scatterplot between *cylinders* and *mpg*, we have:

```
> plot(Auto$cylinders , Auto$mpg)
```

- Alternatively, we can use the `attach()` function:

```
> attach(Auto)
> plot(cylinders , mpg)
```

Introduction to R

Basic Commands

- The `as.factor()` function converts quantitative variables into qualitative variables.

```
> cylinders=as.factor(cylinders)
```

- If the variable plotted on the x-axis is categorical, then boxplots will automatically be produced by the `plot()` function.

```
> plot(cylinders, mpg)
> plot(cylinders, mpg, col="red")
> plot(cylinders, mpg, col="red", varwidth=T)
> plot(cylinders, mpg, col="red", varwidth=T,horizontal=T)
> plot(cylinders, mpg, col="red", varwidth=T, xlab="cylinders",
    ylab="MPG")
```

Introduction to R

Basic Commands

- The `hist()` function can be used to plot a histogram.

```
> hist(mpg)
> hist(mpg,col=2)
> hist(mpg,col=2,breaks=15)
```

- The `pairs()` function produces a scatterplot matrix.

```
> pairs(Auto)
> pairs(∼ mpg + displacement + horsepower + weight +
    acceleration , Auto)
```

Introduction to R

Basic Commands

- The `summary()` function produces a numerical summary of each variable in a particular data set.
- We use `q()` in order to quit R.
- We use `savehistory()` function to save a record of all the commands that we typed in the most recent session.
- We use `loadhistory()` to load the history of commands.