Northeastern University

INFO 6105
Data Sci Eng Mth & Tools
Lecture 2
Formulas & Modeling in R

14 January 2019

· Very important! Need to master it to solve quadratic equations

When you have $ax^2+bx+c=0$

the solutions are

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

Example solve

$$x^2-9x=-8$$

First write in standard form

$$1x^2-9x+8=0$$

$a=1$ $b=-9$ $c=8$

plug in values into formula

$$x=\frac{-(-9)\pm\sqrt{(-9)^2-4(1)(8)}}{2(1)}$$

$$x=\frac{9\pm\sqrt{81-32}}{2}$$

$$x=\frac{9\pm\sqrt{49}}{2}$$

$x=\frac{9+7}{2}$ $x=\frac{9-7}{2}$

$x=\frac{16}{2}$ $x=\frac{2}{2}$

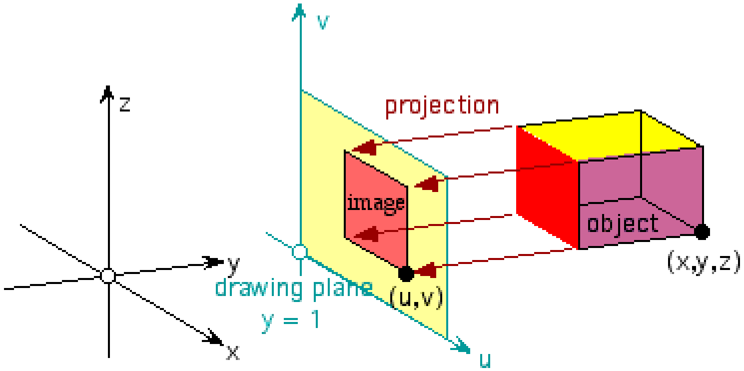
solutions → $x=8$ $x=1$

Part 2

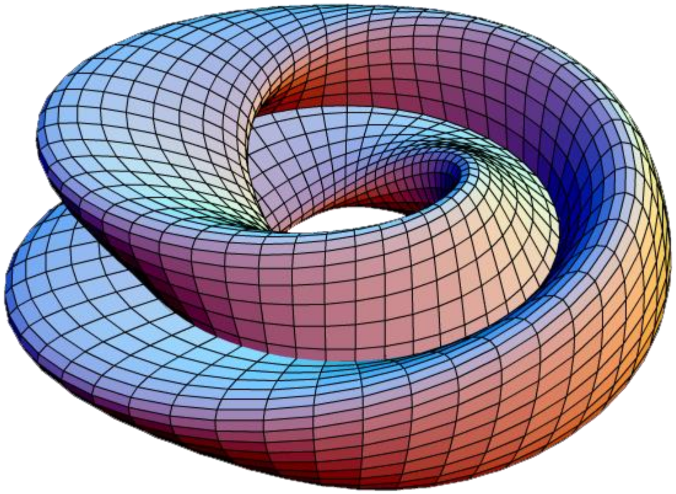
MODELING & FORMULAS IN R

BigData Processing

- *Projection* operations on distributed datasets (a.k.a. disk methods, a.k.a mapreduce)



- Building *models* and throwing away the data (a.k.a. ML)
  - One datum at a time



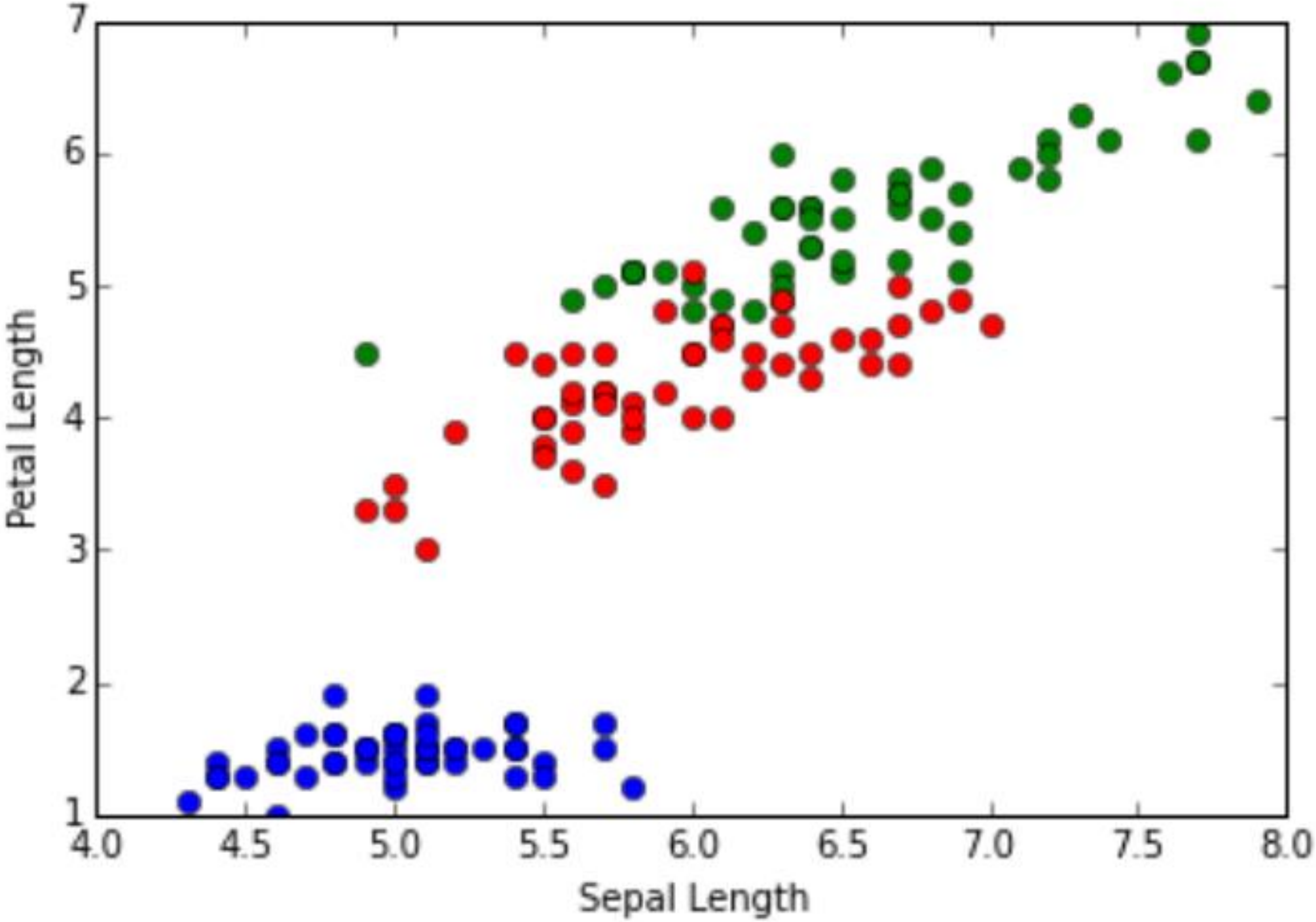
Formulas in R

- Formulas are used inside function calls to generate "special behavior“
  - Allow you to *capture the values of variables without evaluating them so that they can be interpreted by the function*
  - Use these R objects to express a relationship between variables
  - Example from language lab: **`barchart(key ~ freq, data = head(stats, 20), col = "magenta", main = "Keywords-simple noun phrases", xlab = "Frequency")`**
- The variable on the left-hand side of a tilde (**~**) is called the dependent (label) variable, while the variables on the right-hand side are called the independent (predictor) variables and are joined by plus signs (**+**)
  - **`f ← y ~ x + b`** #y is a function of x and b
  - **`f <- as.formula("y ~ x + b")  #try typeof(f)`**
  - **`Sepal.Width ~ Petal.Width + log(Petal.Length) + Species`**
  - **`Sepal.Width ~ Petal.Width | Species  #sepal width is a function of petal width, conditioned on species`**

List of Formulas (proofs when formulas..)

- ```
  i <- y ~ x
  j <- y ~ x + x1
  k <- y ~ x + x1 + x2
  ```
- ```
  # Concatenate
  formulae <-
  list(as.formula(i),as.formula(j),as.formula(k))
  ```
- ```
  # List
  formulae[[1]]
  ```
- ```
  # Update
  update(y ~ x1 + x2, ~. + x3)    # y ~ x1 + x2 + x3
  ```
- ```
  $ Check
  library(plyr)
  is.formula(f)
  ```
  ```

Where formulas are used: Statistical Modeling

- **Simplified, mathematically-formalized way to approximate reality and optionally to make predictions from this approximation**
- **A statistical model represents the data generating process in an idealized form**
- **Modeling functions in R are where you need:**
  - **A `formula` object as an argument**
  - **Data as an argument, which allows you to specify a data frame that you want to attach for the duration of the model**
  - **Tools like `subset` to select the data that you want to use**

Linear modeling

- You use `lm()` to fit linear models
- You can use it to perform regression, analysis of variance and analysis of covariance
- # iris dataset
  - https://en.wikipedia.org/wiki/Iris_flower_data_set
  - https://archive.ics.uci.edu/ml/datasets/iris
- ```
  data(iris)
  head(iris)
  ```
- `setosa = iris[iris$Species == 'setosa']`
- `versicolor = iris[iris$Species == 'versicolor']`
- `virginica = iris[iris$Species == 'virginica’]`
- `s = plot(setosa$Sepal.Length, setosa$Petal.Length)`
- `vi = plot(virginica$Sepal.Length, virginica$Petal.Length)`
- `ve = plot(versicolor$Sepal.Length, versicolor$Petal.Length)`
  ```

Setosa, versicolor, virginica



Linear modeling (continued)

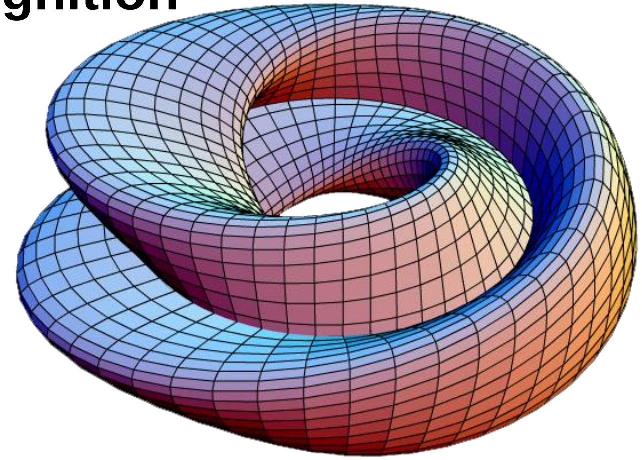
- `x = iris$Petal.Width + log(iris$Petal.Length)`
- `y = iris$Sepal.Width`
- `plot(x,y)`
- `x = iris$Petal.Width[iris$Species=="setosa"] + log(iris$Petal.Length[iris$Species=="setosa"])`
- `y = iris$Sepal.Width[iris$Species=="setosa"]`
- `plot(x,y)`
- `lm.m <- lm(Sepal.Width ~ Petal.Width + log(Petal.Length) + Species, data = iris, subset = Sepal.Length > 4.6)`
- `print(lm.m)`
- `abline(lm.m)`

Non-linear modeling

- Let's do an exponential with noise, and see if we can recover the general tendency as a model so we can use it to predict..
- Use `nls()` to fit non-linear models
- `x <- seq(0,50,1)`
- `y <- ((runif(1,10,20)*x)/(runif(1,0,10)+x))`
- `plot(x,y)`
- `y <- ((runif(1,10,20)*x)/(runif(1,0,10)+x))+rnorm(51,0,1)`
- `plot(x,y)`
- `nls.m <- nls(y ~ a*x/(b+x),`
- `start=c(a=4, b=1))`
- `lines(x, predict(nls.m), col = "red")`

Time-out

- Building a model is *exactly* what Artificial Neural Networks do
- It's just that because there's so much data, our typical math and statistical packages cannot handle the data crunching..
- ANNs are designed to do exactly that, real well
- All they are curve builders in high-dimensional space
- Also, that's what your brain is all about
- It draws geometry to allow you to navigate best possible paths in life in accordance to your experience
- Intelligence = geometry & pattern recognition



Where formulas are used: `Graphics`

- `library (graphics)`
- `#`https://www.rdocumentation.org/packages/graphics/versions/3.4.0/topics/plot.formula
- ```
  data(airquality)
  plot(Ozone ~ Wind, data = airquality, pch =
  as.character(Month))
  ```
  ```

Where formulas are used: `lattice`

- `library (lattice)`
- `data(airquality)`
- # https://cran.r-project.org/web/packages/lattice/lattice.pdf
- ```
  histogram(~ Ozone | factor(Month),
       data = airquality,
       layout = c(2, 3),
       xlab = "Ozone (ppb)")
  ```
  ```

Where formulas are used: `ggplot2`

- `library (ggplot2)`
- `data(mpg)`
- `# use formulas in ggplot2 function geom_smooth(), to specify the formula to use in the smoothing function; This will influence the form of the fit`
  - http://ggplot2.tidyverse.org/reference/geom_smooth.html
  - https://stat.ethz.ch/R-manual/R-devel/library/splines/html/bs.html
- ggplot code:

```
ggplot(mpg, aes(displacement, hwy) +
      geom_point() +
      geom_smooth
      (
            method = "lm",
            formula = y ~ splines::bs(x, 3),
            se = FALSE
      )
```


CLASS
DISMISSED
iStockphoto