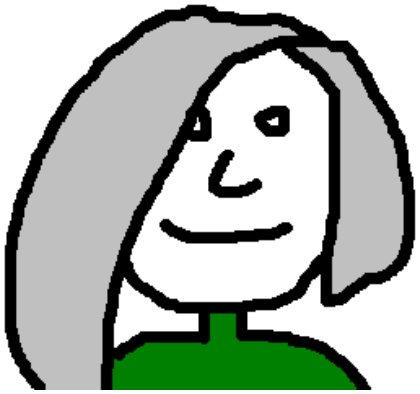
Introduction to Statistics and R

LSA Summer Institute 2009

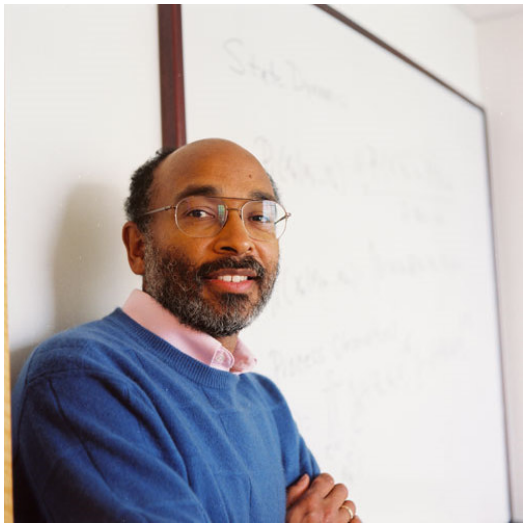
Peter Graff

Acknowledgements



Ellen Gurman Bard

T. Florian Jaeger



Emory Brown

All errors are my own!

PC/Mac/Linux

- This guide is PC based.
- R code is operating system independent
- The R environment will marginally differ depending on your operating system.
- You should be able to generalize all operations onto your respective operating system.
- All “environment” operations can also be done via R-code.

The R Language
Setup

- Download
  - to download R go to http://cran.cnr.berkeley.edu/ and click R-2.8.1.tar.gz
- Installing Packages
  - to install packages in R, enter the following code into the R console

```
> install.packages(package,repos="http://cran.r-project.org")
> install.packages(c(package1, package2, ..., packageN),repos="http://cran.r-project.org")
```

- Packages you will need include
  - `rpart, chron, Hmisc, Design, Matrix, lme4, coda, e1071, zipfR, ape, languageR, MASS`
  - So to install the above packages you would enter:

```
> install.packages(c("rpart","chron","Hmisc","Design","Matrix","lme4","coda","e1071","zipfR","ape","languageR", "MASS"),repos="http://cran.r-project.org")
```

The R Language
Environment

- Type R-code into the R-console...
  - hit ENTER to accept
  - hit UP to access previously entered code
- Use the menu to...
  - Folder icon: Open saved script
  - Computer with blue paper icon: Load saved image
  - Diskette: Save image
  - File > Change dir: Set working directory
  - File > New Script: Open New Script
  - Edit > GUI preferences: Set font size
  - Packages > Install package(s): Lazy package install

The R Language
Scripting

- Even though it seems tedious at first it is good practice to write everything you do in R into a script. This is a normal text file generated in the R environment and saved with the extension ".R"
- To open new script click: File > New Script
- Enter your code into the script as if you were typing it into the console.
- Comment your code! In are you can comment out text in your script if you precede it with the "#" sign. You should always force yourself to comment your code in detail. It will make your life a lot easier.

```
> # This line will not be executed.
```

- To run part of your script select the text and press CTRL+R
- To run the whole script select Edit > Run all or press CTRL+A followed by CTRL+R

The R Language
Important Basic Functions

- library(*package*)
  - Loads a given package
- read.csv(file="*filename.csv*", header=T)
  - Imports a CSV file with a header into R
- read.table(file="*filename.csv*", header=T)
  - Imports a tab-delimited text file with a header into R
- help(*function*) and ?*function*
  - Opens the help window for a given function
- objects()
  - Displays all declared variables
- save.image(file="*yourfilename*.RData")
  - Saves all functions and variables you defined to an R-Image. Or use the menu if you're as lazy as me.

The R Language
Variables

- The basic unit of computation in R is the vector. Even a single number is stored as a single number vector and can be manipulated as such.
- Use so called assignment operators to declare variables.
  - "="
    `x = 5`
    assigns number 5 the name x
  - R-specific assignment operators "->" and "<-"
    `x <- 5` and `5 -> x` are both equivalent to `x = 5`
- Multi-member vectors can be made with the c() and the cbind() functions. Separate multiple arguments with commas.
  - Concatenate vertically
    ```
    > c(1,2)
    [1] 1 2
    ```
  - Concatenate horizontally
    ```
    > cbind(1,2)
         [,1] [,2]
    [1,]    1    2
    ```
- And of course you can also make a little matrix like that
  - ```
    > cbind(c(1,2),c(1,2))
         [,1] [,2]
    [1,]    1    1
    [2,]    2    2
    ```
    ```

The R Language
Dataframes

- Now it's only one more step to another important R-entity, the dataframe.
  - A dataframe is a matrix with column names
  - When you import data with a header into R with one of the read.csv/read.table functions, it is imported as a data frame.
  - Dataframes can also be declared as variables
    - `data = read.csv(file="filename.csv", header=T)`
- There are two important operators for dataframes
  - The column operator "$"
    Returns a single column of a dataframe as a vector
    - *`dataframe`*`$`*`columnname`*
  - The index operator "[,]"
    Returns a certain cell or part of a dataframe (also works for matrices)
    - *`dataframe`*`[`*`row`*`,`*`column`*`]`

The R Language
Dataframes

- Example from Bresnan (2007; load *languageR* and type in "`verbs`" to call this dataframe)
- `head(verbs,2)`

  Displays the header and the first 2 lines of `verbs.` Default is 6.

```
>     Real Verb      AnimOfRec     AnimOfThem      LengthOfThem
   1    NP  feed       animate       inanimate       2.639057
   2    NP  give       animate       inanimate       1.098612
```

The R Language
Boolean Operators

- And “&”
- Or “|”
- Equals “==”
- Greater than “>”
- Smaller than “<”
- Greater or equal “>=”
- Smaller or equal “<=”
- Not equal “!=”

- Using Boolean operators with the dataframe index function can be very useful.
  - > *dframe*[,*dframe*$*col2*==“X”]
    Returns the rows of the dataframe, where col2 equals “X”
  - > dframe[*dframe*$*col1*,*dframe*$*col2*==“X”]
    Returns the cells of col1 of the dataframe, where col2 equals “X”
- R also has Boolean variables “TRUE”/“T” and “FALSE”/“F”, which you will mostly encounter as values for arguments of functions.

The R Language
Mathematical Operators

- Addition “+”

```
> 5+7
[1] 12
```

- Subtraction “-”

```
> 5-7
[1] -2
```

- Multiplication “*”

```
> 5*7
[1] 35
```

- Division “/”

```
> 5/7
[1] 0.7142857
```

- Power “^”

```
> 5^7
[1] 78125
```

The R Language
Functions

- A function is another basic unit of computation.
- Functions take a predetermined set of arguments and perform certain operations on them.
- You can easily define your own functions in R

```
> multiply.by.2 <- function(input) return(2*input)
> multiply.by.2(2)
[1] 4
```

- For functions with multiple arguments separate individual arguments with ","

```
> multiply.this <- function(n1,n2) return(n1*n2)
> multiply.this(2,4)
[1] 8
```

- This can be very useful if you tend to perform similar operations on different datasets.

The R Language
Functions you should know

- `attach(`*`dataframe`*`)`
  Attaches a particular dataframe. You no longer need to use the column operator "$" to refer to particular colums in your data. (e.g. `ReactionTimes` is not equivalent to `MyData$ReactionTimes`)
- `detach(`*`dataframe`*`)`
  Unattaches previously attached dataframe.

The R Language
The Aggregate function in R

- `aggregate(dataframe$vector, list(dataframe$vector1,..., dataframe$vectorN,), function)`
- Applies function to cells determined by list.

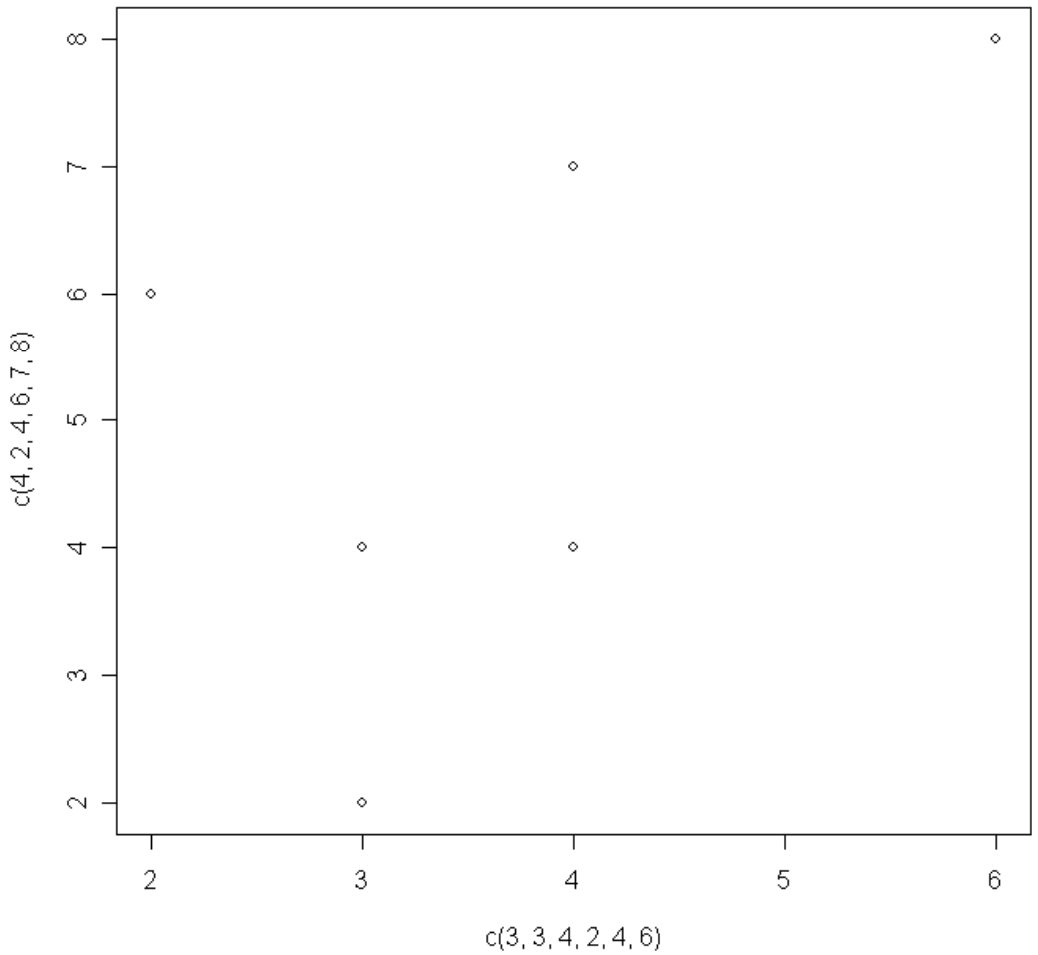
The R Language
Plotting

- R is great for plotting!
- Most plotting functions take a vector or a dataframe as arguments. Today we will only deal with plotting functions that take vectors as a crucial concept, the formula, has not yet been introduced.
- All plotting functions will thus be of the format `function(x)` or `function(x,y)` where `x` and `y` are vectors.

The R Language
Plotting

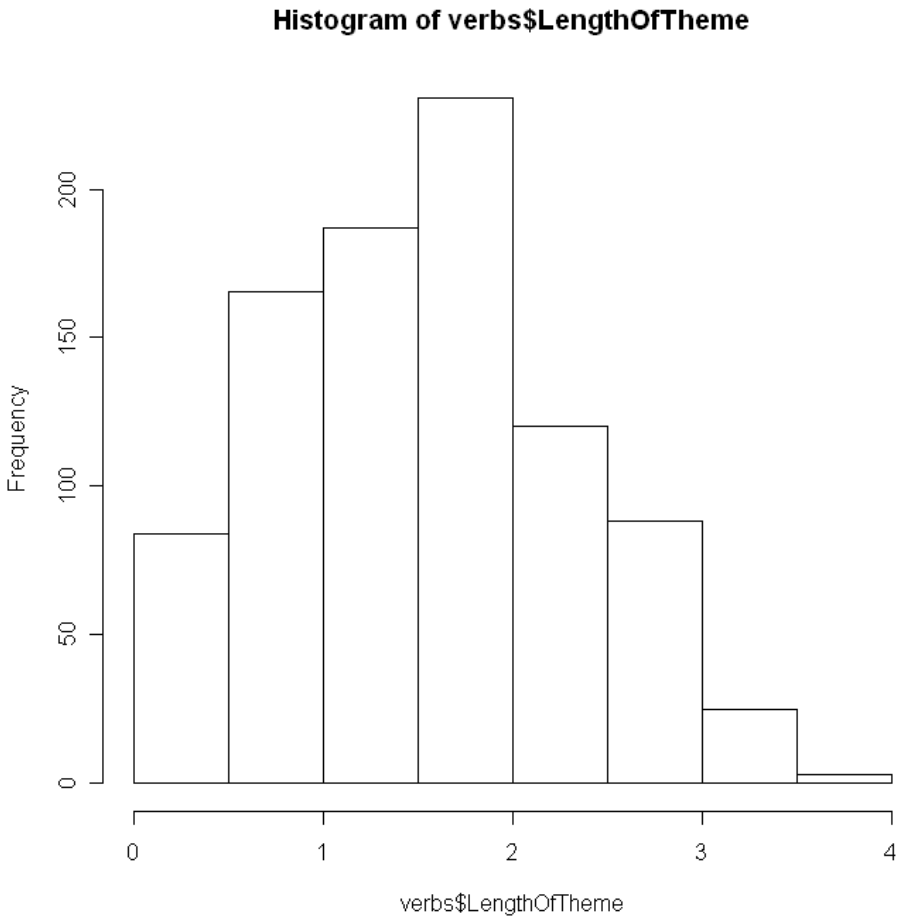
- Graphic parameters are additional arguments fed to the plotting function.
- R lets you specify pretty much anything when plotting. The help() function applied to a given plot tells you what additional parameters you can specify specific to that plot function
- help(par) tells you the possible graphic parameter settings that apply to almost all plotting function
  - `col = "color"`
    Sets default plotting color, consult internet for color codes.
  - `main = "title"`
    Sets plot title to *title*
  - `xlab = "title"`
    Sets x-axis title to *title*
  - `ylab = "title"`
  - `xlim = c(min,max)`
    Sets x-axis limits between *min* and *max*
  - `ylim = c(min,max)`
- Remember additional arguments are separated with comma ",".

XY-Plot



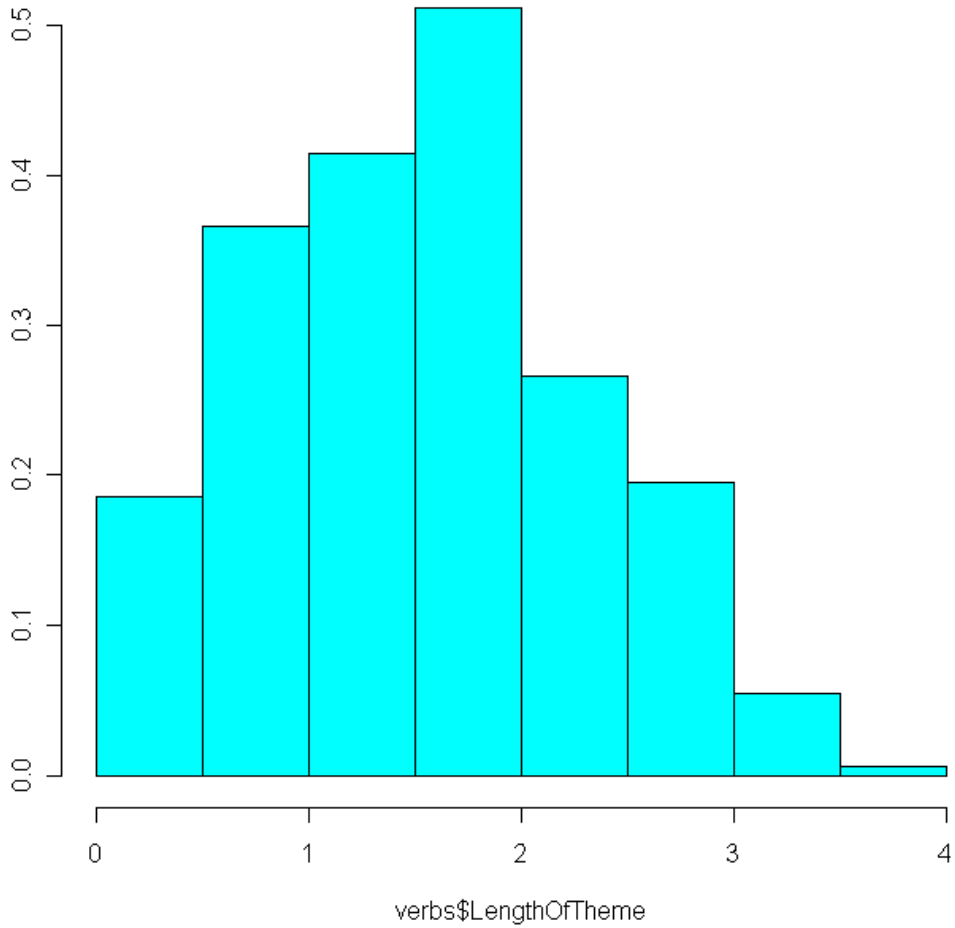
- XY-Plot
  plot(x, y)
  Generates a coordinate system with dots at the points with coordinates (x[1],y[1])... (x[n],y[n]), x and y need to be of equal length.
- Lines next time!

The R Language
Histograms



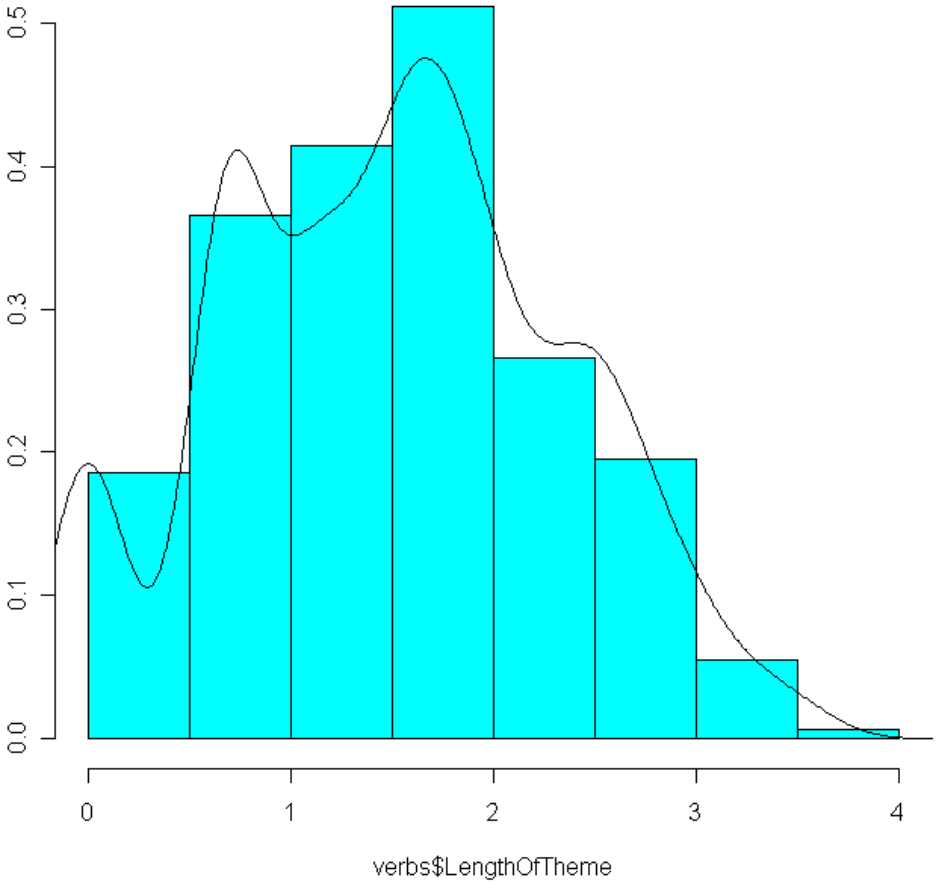
- hist(x,*numberofbins*) makes a histogram of the vector x with *numberofbins* bins.

The R Language
Histograms



- truehist(x) makes a standardized histogram of the vector x

The R Language
Histograms



- To overlay a standardized histogram with a smoothed density curve enter lines(density(x)) straight after generating the truehist(x)