

# R for Linguists

Week 1: Day 2

Wassink

Spring 2018

University of Washington

# Today

- Table groups
- Two ways to run R: CRAN vs. R Studio
- Data structures:
  1. What is a corpus?
  2. What kind of data do I have?
  3. How to save a textfile for use by R

# Rdg Qs

- no questions posted to Canvas

**Files to have ready for today:**

YZ40NF2E\_RP.txt

count-freqs-shell.R

# The command line: complaints

- What is scary about the command line?
  - hard to learn the commands/instructions
  - you can do something wrong and not know what it is or how to fix it
  - I do simple stuff. Why should I use it to make a simple table?
  - ...

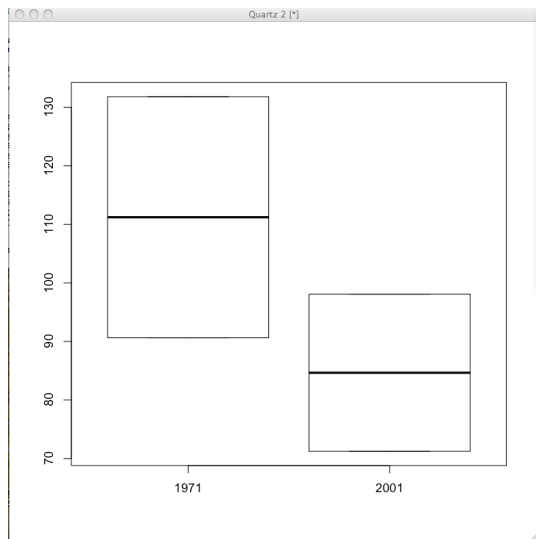
# Counterarguments

- scripts are reusable
- versatility: scripts can do just what you want
- responsibility: do we really know what Excel or AntConc did with our data?
- flexibility: work better with some databases
- speed
- better understanding of how analyses are done
- and of course, there's always `help()` or `?read.xlsx`

# Now...Installing R: Method I

1. [.http://cran.at.r-project.org](http://cran.at.r-project.org)
2. Install the base program in the location recommended for your operating system
3. Install all of R into the suggested directory
4. Launch R...you will see this:

# The R Interface



Quartz Window Command history Editor

```
Commands
x<-c(1:10)
sample(x,size=5,replace=T,prob=NULL)
sample(x,5,T,NULL)
sample(x,5)
sample(x)
sample(x,10,F)
(a.name<-"James")
class(a.name)
class(x)
length(a.name)
(names<-c("James","Jonathan","Jean-Luc'
numbers1<-c(1,2,3); numbers2<-c(4,5,6
numbers1.and.numbers2<-append(num
numbers1.and.numbers2
numbers2+numbers2

Delete entry Clear History
Load History Save History

39 106 2001 t
40 54 2001 t
41 49 2001 t
42 56 2001 t
43 58 2001 t
44 97 2001 t
> colnames(vot)
[1] "VOT" "year" "Consonant"
> summary(vot)
> summary(vot)
VOT year Consonant
Min. : 45.00 Min. :1971 k:21
1st Qu.: 71.50 1st Qu.:1971 t:23
Median : 81.50 Median :2001
Mean : 96.45 Mean :1989
3rd Qu.:120.25 3rd Qu.:2001
Max. :193.00 Max. :2001
> vot.Consonant.mean <- tapply(vot$VOT,vot$Consonant,mean)
> place.laryngeal <-tapply(vot$VOT,list(vot$Consonant,vot$year),mean)
> boxplot(place.laryngeal)
>
```

R Data Editor

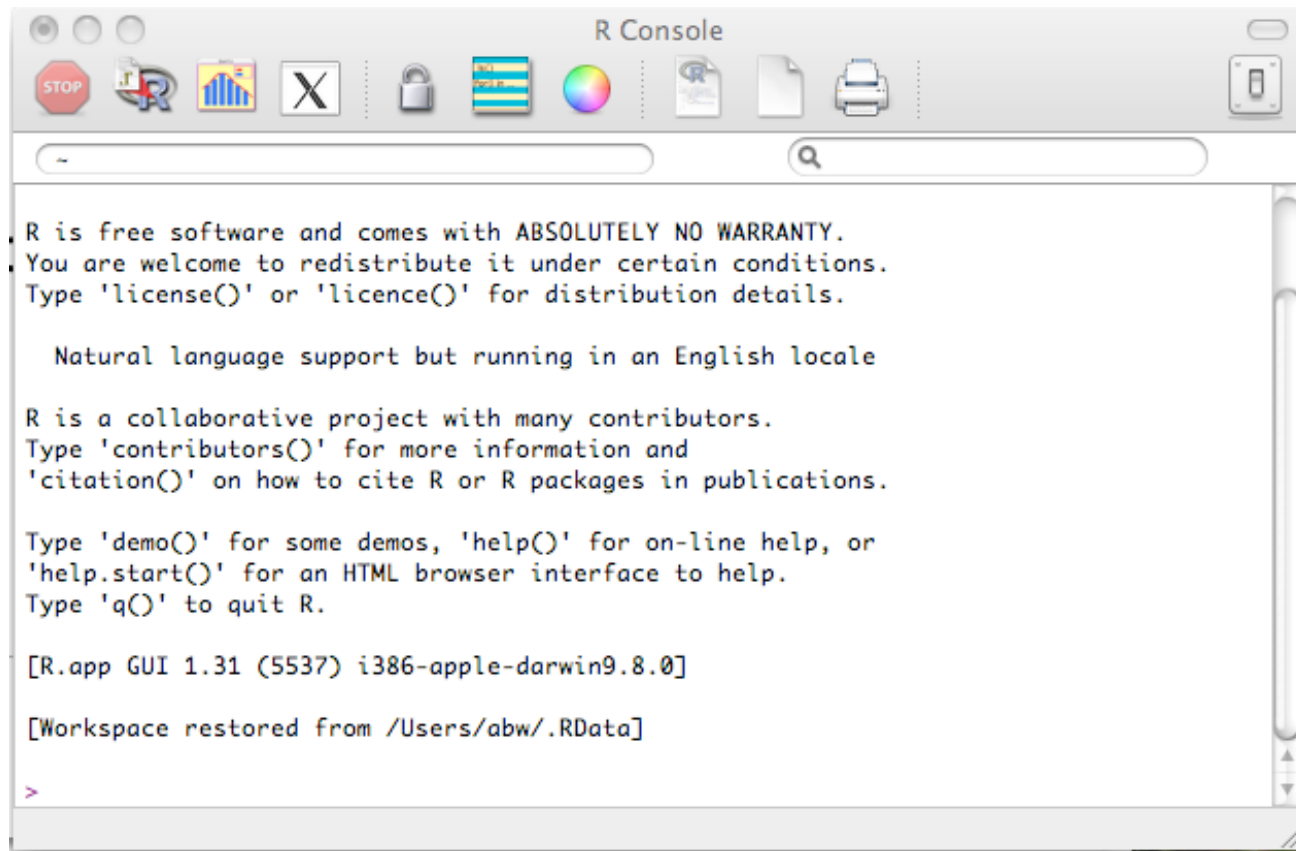
| VOT  | year | Consonant |
|------|------|-----------|
| 67   | 1... | k         |
| 1... | 1... | k         |
| 79   | 1... | k         |
| 1... | 1... | k         |
| 53   | 1... | k         |
| 65   | 1... | k         |
| 75   | 1... | k         |
| 1... | 1... | k         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 1... | 1... | t         |
| 84   | 2... | k         |
| 82   | 2... | k         |
| 72   | 2... | k         |
| 1... | 2... | k         |
| 1... | 2... | k         |

The quartz device window  
(for tables and graphics)

The R Console (for  
instructions), with or without  
the The R command history

R Data Editor  
Window (for  
Excel-type view  
of working  
dataset)

\*note: can type at the  
command line to access this:  
> edit(name)

The image shows a screenshot of the R Console window on a Mac. The window title is "R Console". The toolbar contains icons for a red stop sign, the R logo, a bar chart, a window with an 'X', a lock, a flag, a color wheel, a document with the R logo, a document, and a printer. The main text area contains the following text:

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
[R.app GUI 1.31 (5537) i386-apple-darwin9.8.0]  
  
[Workspace restored from /Users/abw/.RData]  
  
>
```

**5** • to get other packages (sets of functions) R needs, enter:

```
> install.packages("ggplot2")
```

**6** • install these packages: `ggplot2`, `gcookbook`, `plyr`, `phonR`, `gsubfn`, `cluster`, `ape`, `vowels.R`, `languageR`

**7** • next, download examples and exercises from the course companion website:  
<http://www.linguistics.ucsb.edu/faculty/stgries/research/qclwr/qclwr.html>

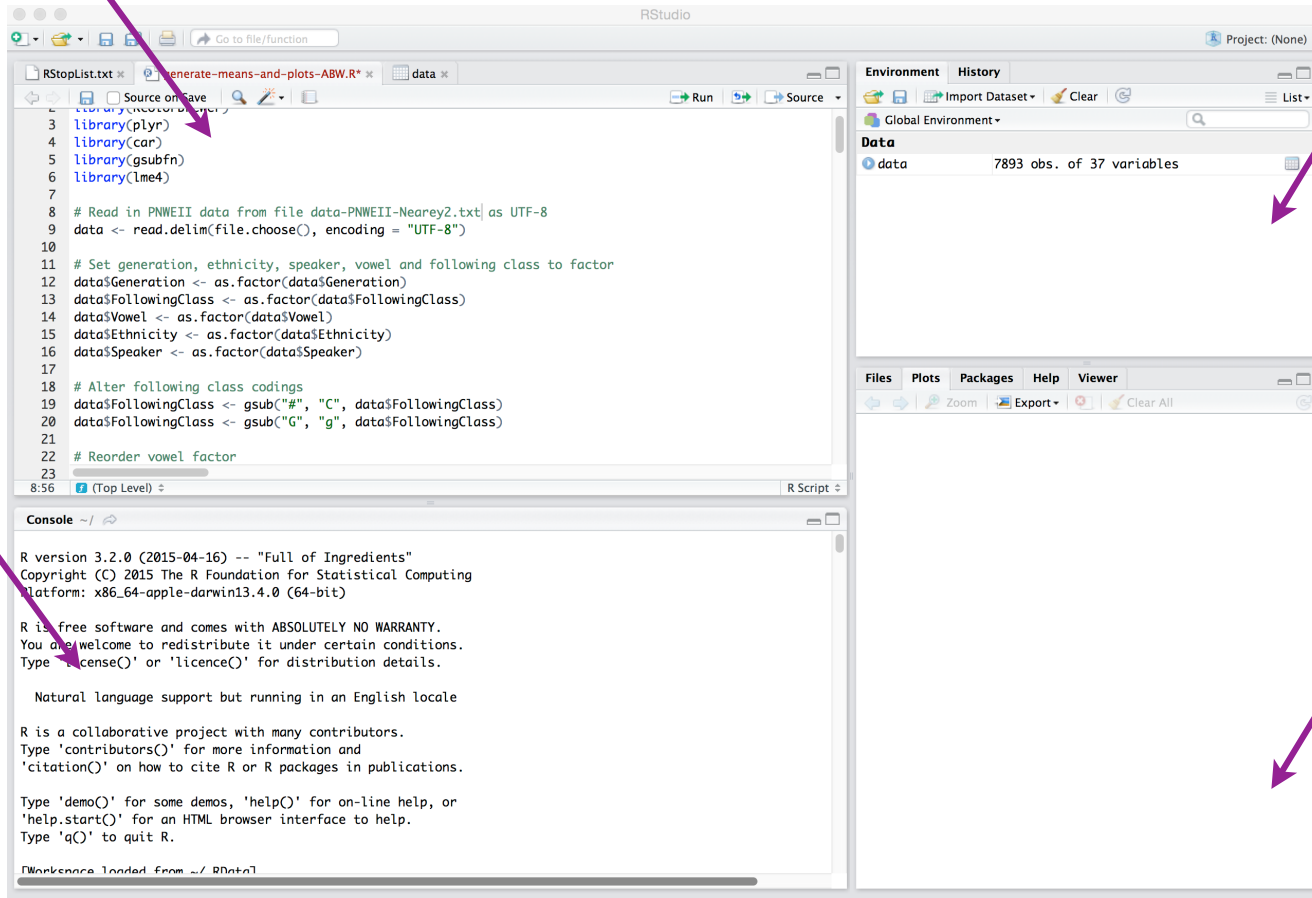


# Method 2: R Studio

- A very widely-used, free graphical user interface
- to install R studio, navigate to: [https://  
www.rstudio.com/](https://www.rstudio.com/)

Source Editor Pane

Environment Pane



Console Pane

Output Pane

# R Studio Interface

# Packages

- “packages in R are collections of functions and/or data that are bundled up for easy distribution” p. 1 RGC
- extend functionality of R
- shared over CRAN (Comprehensive R Archive Network), Bioconductor, Omegahat
- once installed, you need to “load” packages each new R session using `library()`

```
> library(ggplot2)
```

Note quotations usage:  
`install.packages("ggplot2")`,  
but  
`library(ggplot2)`

# Command Line 101

- Do this: find out what your working directory is:

- `getwd()`

```
> getwd()  
[1] "/Users/abw"
```

- Set your working directory to location of your qclwr directory:

- `setwd()`

- quotations must surround your pathname

- Mac trick: use CMD+I “Get info” to copy/paste pathname

```
> setwd("/Users/abw/Documents/work/teaching/  
LING580RinLinguisticAnalysis/qclwr/_scripts/  
exact_matches.r")
```

# Help!

- getting help
- Windows: *Help:HTML help*
- Mac: *Help:R for MAC OSX FAQs*
- either:
  - `> help.start()` to get to R's online help website
  - `> ?help` if you don't know the function you want
  - `> ?sqrt` if you do.

# Intro to Gries book

- Why are we using a book on “corpus” linguistics?
- corpus lx is a way of doing linguistic analysis (*how* is as important as *what*)
  - data retrieval (digging into the data)
  - data evaluation (saving data for work elsewhere)
  - R as “all-purpose tool” (calculator, graphics, statistics, programming)

McEnery and Wilson, 1996: "the study of language based on examples of real life language use"

# Anatomy of an R session

- Clear workspace & working memory
- Set working directory
- Do something
  - write source code (script)
  - input
  - manipulate data
  - annotate your source code
  - save your work
- Clear working memory

Script: “A bit of R code that does something, or gives instructions to R to do something” (Gries)

# Tips and Tricks

- `Ctrl+1` — Move focus to the Source Editor
- `Ctrl+2` — Move focus to the Console
- `Ctrl+L` — Clear the Console
- `Ctrl+Enter` — Execute single line from source pane
- `SHIFT + CTRL + C` — comment sequence of lines
- `Esc` — Interrupt R
- `↑` to access history (last commands executed)
- `TAB`— (source and console panes) autocomplete for items active in your workspace



# R primer

- R is an “Interpreted language:” a programming language in which commands are 'indirectly' executed ("interpreted") by an interpreter program, which executes source code step-by-step, rather than translating it into machine code. This can be contrasted with a compiled language which is converted into machine code and then 'directly' executed by the host central processing unit. (Python, PERL, MATLAB are interpreted.)

# R primer

- Commands typically consist of:
  - functions (instruction to do something)
  - arguments (target of the operation, how to apply itself)
  - assignment operator `<-`
  - user-assigned variable or data structure
- ex. `x<-c(1:10) # this is an inline comment`
- ex. `q()`



Null argument

# cont.

- Variables: temporary “holders” that store values, allow for abstraction
  - Variable names can be any combination of numbers, letters, and `_`
- Functions: Instructions that take zero or more arguments, do some computation, and return a value and/or have other side-effects
- Examples: `rm()`, `print()`, `length()`, `sample()`

# rm()

- Look up this function in your Output pane
- Use this at the beginning and end of an R workflow:

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

# Functions

e.g., `length()`

- Arguments: Element a function needs to work: (1) target of the instruction, (2) how to apply method
- Some functions may take null arguments, others cannot (e.g., `c()`)
- Some may take (upto) a particular number of arguments
- Some require arguments to be labeled
- if no labels ok, strict order must be observed
- if use labels, order can be mixed up

# file.choose()

```
Mac: > file.choose()
```

```
Mac: > file.list<-select.list(dir(scan(nmax=1,  
what="char")),multiple=T)
```

```
Windows: > choose.files()
```

- Open pre-existing file
- Useful because essentially removes need to set working directory for each R session
- Do This: What does each of these do?

```
dir()
```

```
scan()
```

```
select.list()
```

# A Few Fave Functions

- `head()` # Return first 6 lines of data structure
- `tail()` # Return last 6 lines
- `my.variable <- c()` # assignment occurs this way
- `str()` # examine the file structure
- `length()`
- `nchar()`
- `getwd()` # tell me what my working directory is
- `quit()` # sometimes arg can be null
- `sqrt(6);sqrt(16)` # semicolons allow two functions to be  
# combined in one single line
- `sample(100)` # nice way to get a randomized list of  
# integers b/w 1 and 100 w/o replacement
- `table(i)` # yields frequency table of vector elements

- `scan(file="", what=double(0), sep="")` # loads the contents of text files into a vector
- `⋮` # double(0) if input is to be vector of numbers
- `what` # "char" if input is to be vector of characters
- `⋮` # "" any whitespace character can separate entries
- `⋮` # \t indicates that tabs separate your entries
- `sep` # \n indicates that newlines (return) separate them
- `scan(file="clipboard")` # load the contents of the clipboard

Don't forget to assign the output of scan to a variable!

- `apply()` #family of functions. This one works on arrays.

A helpful link on the apply() functions is this:

<http://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/>



# Corpora

- Gries comes at this as a corpus linguist: corpus ≠ text archive
- Corpus: machine-readable collection of texts
- Text: spoken or written
  1. produced in natural communicative settings
  2. representative: “authentic”
  3. balanced: all parts of the genre or variety are sampled, reflect real-world proportions

# Types of Corpora

- General vs Specific:
  - general corpora are representative and balanced for language as a whole
  - specific corpora are restricted to a particular variety, register, genre, speaker, etc.
- Raw vs Annotated:
  - raw corpora are corpus materials only
  - annotated corpora are encoded ... (recall)

# Recall

- Annotation: the act of adding, to primary linguistic data, information representing analyses or models of aspects of the data.
  - interactional events: overlap or turn points
  - lemmas: dictionary form (“loved” lemma: “love”)
  - tokenization (segmentation)
  - interlinear glosses (morphological) or part of speech tags
  - parsing of intonational units or syllables
  - syntactic constituents

# Annotation Graphs

- \* The elements of a linguistic analysis that can be annotated and for which annotation conventions can be codified separately are of at least three types: (1) tokenization/segmentation, (2) syntagmatic structure, (3) paradigmatic content of the events/tokens and structure.

- \* tokenization: the identification of instances or things (segmentation)
- \* syntagmatic structure: the identification of (often linear) relations among things
- \* paradigmatic content: the identification of classes of things or relational functions

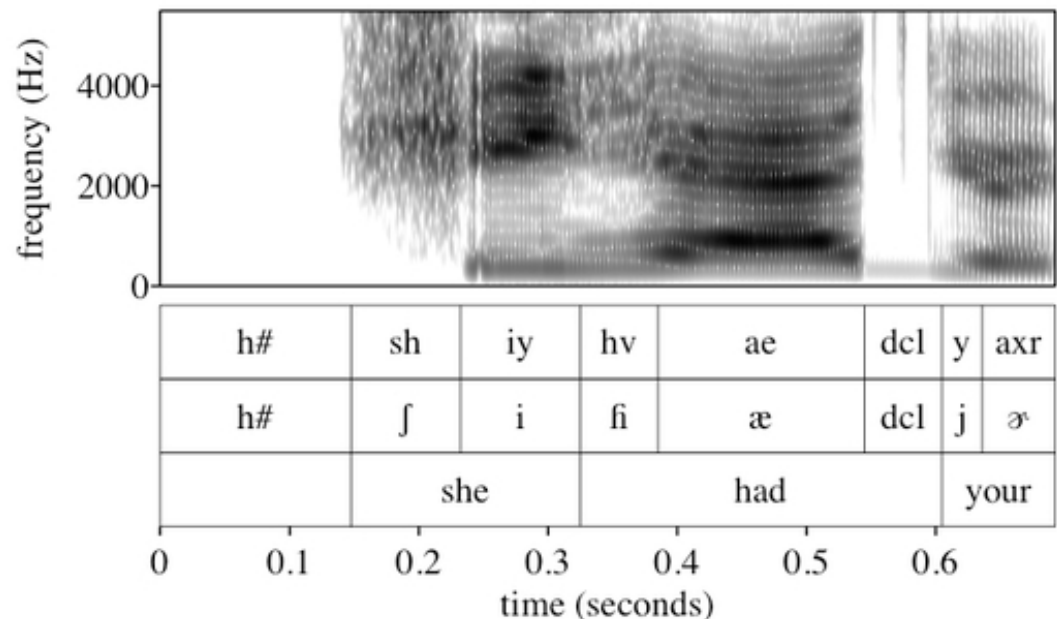
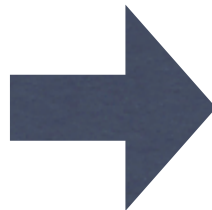
## This is an Annotation Graph (Bird & Liberman 2001):

An annotation graph is a directed acyclic graph, in which each annotation token is (minimally) a triple consisting of two nodes that point to the positions in the string of labels on any annotation tier, and the label for the arc connecting these points.

A triple  
= set of 3 ordered elements  
= 2 coordinates + 1 label

Here's the triple for the word tier item "she"...

2360 5200 she



# Recall

- Metadata: structured information about data.  
Metadata is descriptive information about an object or resource whether it be physical or electronic.
- \* Dublin Core Metadata Initiative (<http://dublincore.org/documents/dces/>)
- \* Linguistic Data Consortium (<http://www ldc.upenn.edu/Creating/documentation.shtml>)
- \* Text Encoding Initiative (<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/CC.html>)
- \* Corpus Encoding Standard (<https://www.cs.vassar.edu/CES/>)

# Types

- Diachronic vs Synchronic (Monitor relevant here):
  - diachronic corpora represent a variety over time (COHA; 200 years)
  - synchronic corpora are contemporary snapshots (GLOWBE)
- Monolingual vs Parallel:
  - monolingual corpora represent one language
  - parallel corpora contain the same text from different languages (translations)

# Metadata

## Recording Metadata (often in a readme.txt document in main project archive)

- \* Project name
- \* Project website (url)
- \* Name of sound or video file
- \* Format of sound or video file
- \* Name of database file(s) (transcriptions, text tiers, annotation files, etc)
- \* Name of the data administrator or investigator
- \* How to contact data administrator
- \* IRB approval number
- \* Date recording made
- \* Location of recording
- \* Speakers on recording
- \* Publications associated with the project (rules for citation)
- \* Register (Formality)
- \* Type of recorded data (unscripted conversation, dyadic or small group interview, individual interview, reading passage, wordlist, minimal pair list, words in isolation, self-commutation test, map task, attitude or subjective reaction test)
- \* Names of elicitation instruments used to elicit data (with filename, as appropriate)
- \* Version history available for datafiles?
- \* Translations available of transcription files?

## Speaker-level tags:

- \* Name
- \* Sex
- \* Age
- \* Age cohort
- \* Known speech impediments or disorders
- \* Ethnicity
- \* Socioeconomic class
- \* Highest educational level attained
- \* Occupation
- \* Place of birth
- \* Residence history (places lived for more than 6 months)
- \* Regionality
- \* Social Network information available (yes/no)? If yes, name of datafile:
- \* Neighborhood
- \* Bi/Multilingual (yes/no)
- \* Language background (all language varieties [dialect region/language name] spoken)
- \* Languages spoken natively
- \* Languages of high fluency
- \* Languages of low fluency
- \* Writing system used or preferred by speaker
- \* Level of literacy

## Group-level tags:

- \* Language
- \* Language modality (signed, spoken)
- \* Dialect
- \* Task (wordlist, reading passage, casual conversation, etc.)
- \* Bi/Multilingual (yes/no)

## Token-level tags\*:

- \* Vowel (IPA category)
- \* Word
- \* Preceding phone
- \* Following phone
- \* Place
- \* Manner
- \* Voicing
- \* Phonation type
- \* Normalized (y/n)
- \* Stress (primary/secondary/unstressed)
- \* Tone level
  
- \* Other phonetic tags
- \* Window length
- \* Sampling Rate

\* — “annotations” if in-line.

# Data Structures

- an organized form of information, such as an array list or string, in which connected data items are held in a computer
- some types: vector, array, dictionary, graph, hash, heap, list, frequency list, linked list, matrix, object, queue, ring, stack, tree.



# c()

- perhaps the most common R function??
- “combine”
- type into your console and tell us what it means

Combines arguments to form a vector

# Variables

- a value that may change depending on the scope of a particular operation, problem, or set of operations
  - ex. `x` in:  
`x<-c(1:10)`  
`x<-c(10:1)`
- `x` may be referred to independently of the function, can be a name,
  - ex. `agerange<-c(1:10)`

# Vectors

- most basic data structure in R.
- one-dimensional, sequentially ordered sequences of elements (numbers or character strings, or variables). Many other types of data structures can be understood in terms of vectors.
- ex. [1], [1,2,3], [a,b,c], sqrt(5), “myfile.rtf”

```
[1] 57 50 2 71 15
> sample(100)
 [1] 85 22 35 90 73 5 36 100 92 65 71 61 97 81 86 46 17 41 16
 [20] 45 26 72 78 80 70 8 60 31 89 74 33 23 18 77 57 13 25 55
 [39] 56 47 40 6 94 63 52 20 3 59 27 82 66 93 53 76 44 30 48
 [58] 96 28 42 75 14 21 38 34 19 12 91 69 87 10 43 99 62 88 49
 [77] 11 24 32 79 83 98 37 58 4 39 9 50 84 67 7 64 15 2 68
 [96] 29 51 95 1 54
>
```

# Data types

- integers: 7, 2, -5
- real numbers: 13.22354234, -15.124234
- strings: “hello”, “bom dia”, “7”
- lists: [-7, 2, 5], [“hello”, “aloha”]
- in R, all elements of a vector must be of a single (the same) data type
- dataframes: similar to tables, have columns and rows

# Factors

- Similar to vectors, but ... have levels
- How do we think of them in Linguistics? From Statistics, a grouping variable. Each observation in a dataset has as an attribute one of a closed set of values representing defined levels of a *categorical* independent or dependent variable.

e.g., Gender - 2 levels - M/F

- vectors may be factorized:

```
> gender <- c("male", "female", "male",  
"male", "female")
```

```
> (f <- factor(gender))
```

# Importing data

- Use `read.table()` to import data from an external file into a dataframe
- Use `scan()` to import data as a string
- Dataframes have row and column labels; vectors don't
- We still have to name it something to manipulate it in R. Let's name it "df":

```
> df <- scan(file=file.choose(),  
encoding="UTF-8", sep="\n", what="char")
```

- Do this: use `scan()` to read file `YZ40NF2E_RP.txt`

# Demo()

- practice exploring the demonstration materials available for our installed packages
- practice printing demo output to pdf
- `demo(package = "ggplot2")`

\*use some other package. This one was (deliberately) chosen because it doesn't have any

# To do now...

- ...(by midnight Friday) upload a sample demonstration plot to Canvas (assignments area)
- Install your text editor
- Install R Studio and packages