Administrative

Welcome to Statistics and Linguistic Applications offered under the course number LIN875. The goal of this course is to give linguistics graduate students statistical literacy. This is increasingly important as the profession in general becomes more receptive to numbers. The course is not a computational linguistics course, but we will be learning how to use a software tool called ‘R’ to do our calculations for us.

instructor John Hale is going to do his best to help you with your project, but he is not primarily a statistician. Free professional help is available from [http://www.cstat.msu.edu](http://www.cstat.msu.edu)

books The only required texts are two free book drafts available in PDF from the 875 website, [http://www.msu.edu/course/lin/875](http://www.msu.edu/course/lin/875)


optional books If you would like to have a physical book, consider

  - Introductory Statistics with R. Peter Dalgaard. 2002. isbn 0387954759
    - Concise; the appendices are like quick reference cards collating common R commands.
    - A longer work that walks the reader with more time to spend through the basics more gently.

software The course uses the R system for statistical computing and graphics. It is a free version of [Becker et al. 1988](http://www.r-project.org) ‘S’ language that offers “an integrated suite of software facilities for data manipulation, calculation and graphical display.” It was originally created by Ross Ihaka and Robert Gentleman at the University of Auckland Statistics Department. It works on many computer operating systems including Unix, Windows and Mac OS. It is extendable in that it incorporates a functional programming language influenced by Lisp.

  [http://www.r-project.org](http://www.r-project.org)

Please go to the R website and install it on your computer this week. Read the FAQ for installation tips. Take a look at the Introduction to R [Venables et al. 2006](http://www.r-project.org) posted on the distribution site under ‘Manuals’. You don’t need to master this material, just come in next week with a comfort level about what R actually is.

valuation Students in this class are graded on three things

  - participation 10 %
  - homework 40 %
  - final project 50 %

Participation measures students’ intellectual engagement: was the student in class? Did he or she contribute questions about material not initially understood? Did he or she seek help at office hours or by email?

Homeworks are meant to apply ideas introduced in class and to strengthen students’ analytical and computer skills. Distributed semi-weekly, they are graded either as pass/fail or on a 7-point scale.

The final project encourages reflection about the use of probability or statistics in a theoretical model or empirical study of the student’s own choosing. This project could be, for instance:

  - empirical study detail and justify the analysis of an experiment to be run or observations to be collected
  - theoretical model demonstrate or disprove a property of some stochastic theory in linguistics.
A taste of R

R works like a calculator

One invokes R by double-clicking its icon, or by typing R at your operating system’s command line prompt. From there on in, you type R expressions at the R prompt, a greater-than sign >. Expressions can be separated by either a newline or a semicolon. When you are done, issue the command q(). This gets you out of R and back to what you were doing before.

Expressions can use operators like + and - for addition, ^ for exponentiation. There are a variety of built-in facilities, like the constant pi, and functions such as sqrt, exp and log. To call a function, surround its argument with parentheses. Invoke the online help with help(name) or by prefixing the name of interest with a question mark.

> sqrt(2)
[1] 1.414214

The bracketed 1 included in R’s response indicates that there was only one row of outputs needed to display the square root function’s result. By contrast, if one had asked for fifteen random numbers between 0 and 1.0 the result would have taken up more space.

> runif(15)
[1] 0.07786083 0.72144393 0.47507902 0.11465331 0.69637248 0.87815102
[7] 0.27674971 0.89522324 0.08347364 0.31153150 0.79394031 0.07108006
[13] 0.28062156 0.81051126 0.05235952

Some functions take additional named arguments that are more like attributes.

> log(100, base = 10)
[1] 2

R has assignment

R is like those calculators that have nameable memory registers. If you want to assign a name to a particular value, use the left-pointing arrow, i.e. name <- expression. The arrow is composed of two keyboard characters, the less-than symbol and the hyphen that together name one operator. You can make up whatever variable names you want as long as they start with a letter; case matters. Also, be careful not to choose names already in use by the R system like q, the quit command.

The example below assigns the numerical value of the square root of two to a symbolic name sq2

> sq2 <- sqrt(2)

No value is printed after an assignment; R assumes you will access the variable by name if you need the value. All the variables that have been created reside in a ‘workspace’. To see which variables are defined within a workspace, use ls(). You can erase variables with rm().

R uses vectors

Much of R’s functionality is automatically threaded across components of a structured object. For instance, consider a vector of textual strings created using the concatenate function c.

> mystuff <- c("linguists", "love", "statistics")
> length(mystuff)
[1] 3
> toupper(mystuff)
[1] "LINGUISTS" "LOVE" "STATISTICS"
Just as the built-in function `toupper` converts all three components of `mystuff` into uppercase, the division and exponentiation operators apply to corresponding elements to compute the body mass index of some made-up people (example from Dalgaard page 4).

```r
> weight <- c(60, 72, 57, 90, 95, 72)
> height <- c(1.75, 1.8, 1.65, 1.9, 1.74, 1.91)
> bmi <- weight/height^2
> bmi
```

There are special logical values `True` and `False` which are often used as intermediate steps in wrangling data. For instance, we might want to know which patients have a body mass index greater than 25kg/m².

```r
> bmi > 25
[1] FALSE FALSE FALSE FALSE TRUE FALSE
```

Vectors having regular structure can be created using `seq` which is abbreviated to just the colon.

```r
> seq(4, 9)
[1] 4 5 6 7 8 9
> seq(0, 100, 10)
[1] 0 10 20 30 40 50 60 70 80 90 100
```

```r
> uptoFive <- 1:5
> uptoFive
[1] 1 2 3 4 5
```

```r
> downfromFive <- 5:1
> downfromFive
[1] 5 4 3 2 1
```

It’s worth looking at section 2.3 of Venables et al. 2006 to gain proficiency with commands that create vectors with systematic, repetitious structure.

**R has graphics**

The course webpage depicts Zipf’s law, a controversial assertion whose truth you can easily check yourself. The claim is that the frequency of a word is inversely proportional to its rank. As Mandelbrot 1965 points out, it is definitional that rank and number of attestations vary inverse directions. Zipf’s law encompasses that stronger claim that these two quantities in fact vary in inverse proportion.

To make the picture on the webpage, I used the Brown corpus Kučera and Francis 1967 a balanced collection of English texts available through the MSU library. One form of the Brown corpus comes annotated with part-of-speech information, separated from the actual English word by a slash.

A simple Perl script strips off these part of speech tags to extract a sorted list of words each paired with their number of attestations in the corpus. The results immediately show that the definite article is the most common word.
Thus the most frequent word will occur approximately twice as often as the second most frequent word, which occurs twice as often as the fourth most frequent word, etc. For example, in the Brown Corpus “the” is the most frequently occurring word, and all by itself accounts for nearly 7% of all word occurrences. True to Zipf’s Law, the second-place word “of” accounts for slightly over 3.5% of words, followed by “and”. Only 135 vocabulary items are needed to account for half the Brown Corpus.

This distributional information can be loaded into R with the `read.table` command.

```r
> brown <- read.table("/tmp/results", header = F)
> names(brown) <- c("attestations", "word")
> head(brown)

attestations  word
1      62481  the
2      58126   ,
3      49125   .
4      35951  of
5      27850  and
6      25650   to
```

```r
> plot(1:100, brown$attestations[1:100], main = "Zipf's Law in the Brown Corpus", +     xlab = "rank", ylab = "attestations", log = "xy")
```

---

Measurement in linguistics

There is a distinction between an experiment and an observational study.

**observational study** A corpus study is typically observational; the linguist doesn’t decide what people say, but merely records what is said. With this data, sometimes an association can be established e.g. repetition disfluencies tend to precede heavier constituents [Clark and Wasow 1998].

**(controlled) experiment** The experimenter (randomly) assigns subjects to levels of some ‘treatment’ e.g. auditory or written stimuli. Then he or she measures some response — the ‘dependent’ variable. Typical dependent variables include number of mistakes on a test, reaction times, acceptability judgments etc.

Different kinds of data support different kinds of inference.

**nominal** Named properties which have no meaningful order on a scale of any type are called ‘nominal’. Examples: What language is being observed? What dialect? Which word or variant, i.e. *going to* or *gonna*? What is the gender of the person being observed? There is no average fruit.

**ordinal** Orderable properties are called ‘ordinal’. They aren’t observed on a measurable scale, but this kind of property is transitive so that if \( a < b \) and \( b < c \) then also \( a < c \). Example: Zipf’s rank frequency of word-attestations, rating scales like Strongly Agree, Agree, Disagree, Strongly Disagree. The order of finishers in a cross-country race is ordinal. It doesn’t matter how many seconds ahead of the next runner over the finish line you were — they are still one rank behind you.

**interval** Properties measured on a scale that does not have a true zero value are called ‘interval’. On an interval scale, the magnitude of differences of adjacent observations can be determined (unlike the adjacent items on an ordinal scale), but because the zero value on the scale is arbitrary the scale cannot be interpreted in any absolute sense. Examples: temperature (Fahrenheit or Centigrade scales), magnitude estimates such as “This soup is ten times hotter than that other soup.”

**ratio** When measuring on a scale that does have an absolute zero value, the data are called ‘ratio’. They get this name because ratios of these measurements are meaningful. For instance, a vowel that is 100 msec long is twice as long as a 50 msec vowel, and 200 msec is twice 100 msec. Contrast this with temperature where 80 degrees Fahrenheit is not twice as hot as 40 degrees. Examples: Acoustic measures like sound frequency, durations such as reaction time, distances e.g. centimeters.

Parametric tests assume that dependent variables are interval or ratio-scored. Non-parametric tests work with frequencies and rank-ordered scales.

**Homework**

1. Get R working on your computer.

2. Read chapter 1 of Baayen. Take a look at Joan Bresnan’s dative alternation data using the commands that Baayen shows you. Use your Internet connection to automatically `install.packages` before issuing `library(languageR)`.

3. Also read chapter 1 of Vasishth (barely 5 pages). Can you imagine adapting his grading example in your own TA’ing? Ask yourself:
   
   (a) How would you ask R for just the cases where Bresnan observed an animate recipient in Switchboard?
   
   (b) What is the relationship between the summary statistics Vasishth introduces and the boxplot?

Don’t hesitate to use the R online help. The next class meeting will be a lab day to diagnose practical issues using R with linguistic data; bring your laptop if you have one.
References


