

Basic operations with R.*

Greg Kochanski
<http://kochanski.org/gpk>

2010/11/03 10:28:48 UTC

1 Introduction

In this handout, we will discuss basic operations with the R statistical package.

R will operate as a stand-alone package on Windows, Linux or Macs. That means R runs as an application; you start R, then read in data files, do your statistics, and plot your results. You can use a separate application (like Wordpad) to create your data files or you can enter them in R's data editor.

2 Getting Help

Typing

```
> help()
```

gives you a dense description of the help system. To get help on a command, type

```
> help(command)
```

Help pages start with a brief description of the command. Then, they show all the possible information you can feed into the command, in the “Usage” section. The “arguments” section explains the command's options in more detail, option by option. And, near the end, there are often some examples of how you might use it.

3 Entering Data with R's editor

R has a data editor that shows you a nice spreadsheet view of your data. For instance, we can look at one of the built-in data sets, like “faithful” (which is data on the duration and intervals between eruptions of the Old Faithful geyser in Yellowstone National Park in the US).

Note that R produces the “>” or “+” characters at the beginning of lines. Don't type them; they are there to tell you that R is ready for input. “>” means R is ready for a new command, and “+” means that you're in the midst of a command.

```
> fix(faithful)
```

This command¹ produces an editing window that looks like Figure 1.

*This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA. This work is available under <http://kochanski.org/gpk/teaching/06010xford>.

¹ `fix()` calls `edit()` which (when you are editing data) calls `data.entry()`, which calls `dataentry()`. You may well want to call `help(dataentry)` to find out what keys to press.

Editing Data in R: Editing data in R suffers from two flaws. First, there isn't a nice graphical way to create a data set from scratch. Second, there is no way to delete data points in the editor. In a real data set, one generally doesn't want to delete a datum, as it opens up the temptation to delete the data that one doesn't like. (And doing that would invalidate all your statistical results.) In R, you can create more data points easily enough by clicking and typing outside the bounds of the data arrays; just don't accidentally create too many. Worst comes to worst, you can delete data by leaving the editor and taking a slice via `x <- x[1123]`; or similar.

| | Subject's Name | Height | Weight | Dialect | Job Satisfaction | Chocolate Use |
|---|----------------|--------|--------|---------|------------------|---------------|
| 1 | George | 1.8 m | 90 Kg | Luton | 2 | 18 g |
| 2 | Fred | 1.72 m | 78 Kg | London | 1 | 28 g |
| 3 | Tom | 1.77 m | 83 Kg | Lincoln | 3 | 21 g |

Table 1: Data entry for a survey, as you would enter it into R or SPSS. Each subject is a statistical “trial”, and occupies a row. Each different measurement on the subject is a column. (The sequence numbers on the left are required by R when reading the table in from a file. They will be automatically generated by the R data editor. Their purpose is to provide a way of referring to a particular line.)

Each “trial” is a row – in this case, each eruption is a row, and the two columns correspond to the duration of an eruption and the interval between eruptions. If you had an experiment with many people, you'd have one row per person; if you had several different types of data on the people, you'd have several columns (e.g. height, weight, native dialect, ...)². So, if you asked 5 different questions of each person in a survey, you'd have 5 columns because they are different types of measurements (one might “measure” job satisfaction, and another might “measure” chocolate consumption). See Table 1 for an example.

If you make the same measurement several times on each person, as in Table 2, you'd have several rows per person. This would let you see how consistent your subjects were, and perhaps average away some of the randomness of their responses.

So far, we have just modified existing data, since the “faithful” data set is built into R. Unfortunately, R does not provide a single command to create a data set from scratch. This is reasonable in real research (you often have a separate program to collect the data and there is often too much data to easily type in), and anyway R was written by statisticians, and statisticians wait for someone else to collect that data. So, if you want to enter data in R, what does one do? The answer is to create an empty data set with the right columns, then fill it up in the editor. It's not too hard:

```
> d = data.frame(name=character(0), height=numeric(0),
+               mass=numeric(0), dialect=character(0),
+               satisfaction=character(0),
+               chocolate=numeric(0)
+               )
> fix(d)
Warning messages:
1: added factor levels in 'name' in: edit.data.frame(get(subx, ...
2: added factor levels in 'dialect' in: edit.data.frame(get(subx, ...
3: added factor levels in 'satisfaction'...
```

Don't be bothered by the warning messages: they are just telling you that you have more distinct names than you did before the editing process.

² This is the same format you'd use for SPSS.

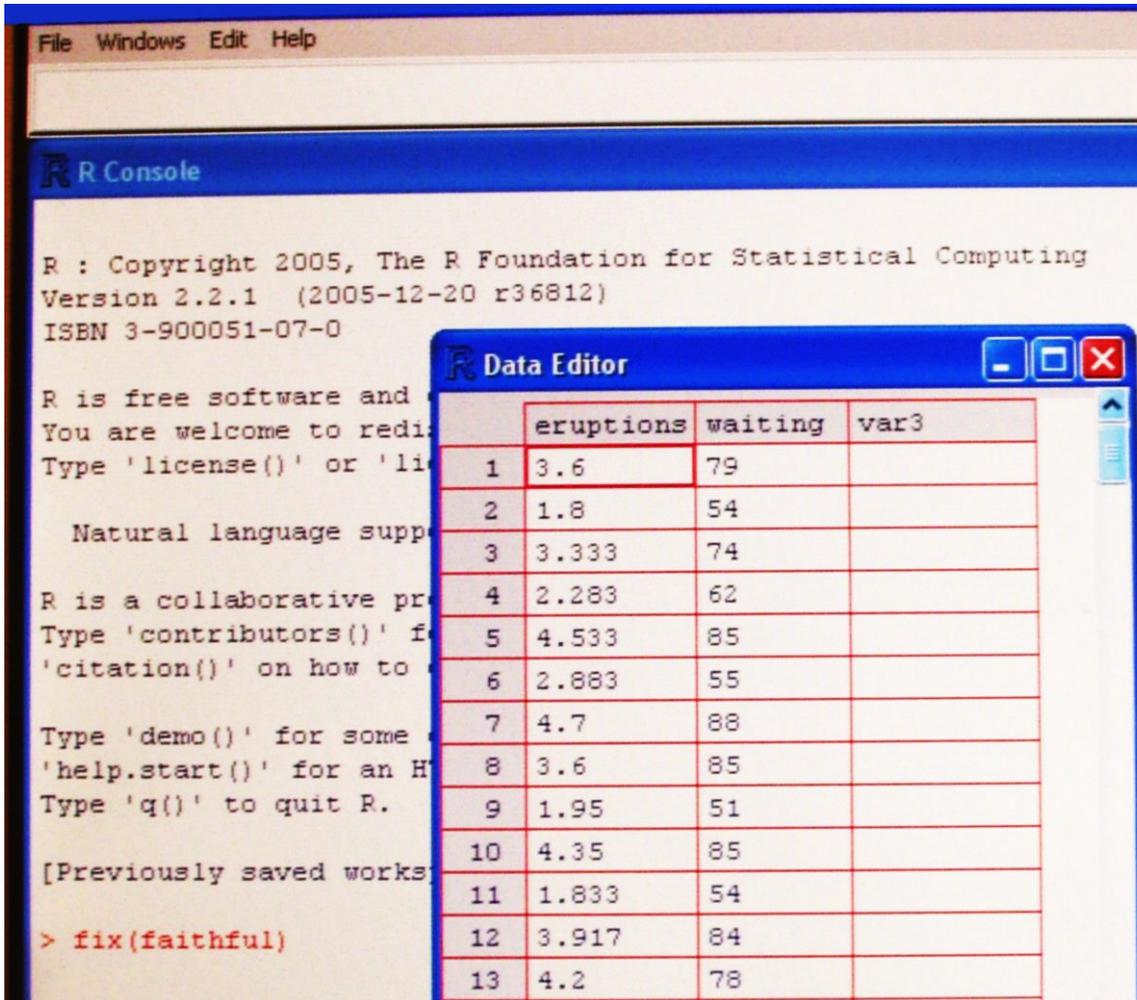


Figure 1: Editing the “faithful” data set. The “Data Editor” window appears when you type `fix(faithful)`.

| | Subject's Name | Trial | Height | Weight | Dialect | Job Satisfaction | Chocolate Use |
|-----|----------------|-------|--------|--------|---------|------------------|---------------|
| 1 | George | 1 | 1.8 m | 90 Kg | Luton | 2 | 18 g |
| 2 | George | 2 | 1.8 m | 90 Kg | Luton | 2 | 12 g |
| 3 | George | 3 | 1.8 m | 90 Kg | Luton | 1 | 11 g |
| 4 | Fred | 1 | 1.72 m | 78 Kg | London | 1 | 28 g |
| 5 | Fred | 2 | 1.72 m | 78 Kg | London | 2 | 22 g |
| 6 | Fred | 3 | 1.71 m | 79 Kg | London | 1 | 18 g |
| 7 | Tom | 1 | 1.77 m | 83 Kg | Lincoln | 3 | 21 g |
| ... | | | | | | | |

Table 2: Sample data entry for an experiment where you test each subject several times. The column labelled “Trial” keeps track of whether a given line was the first, second, or third for that subject.

In the above example, we created a data frame with 6 columns. Three are character strings: “name”, “dialect”, and “satisfaction”; they will be treated statistically as discrete variables. The other three are numbers, treated as continuous (gradient) quantities

Since one can change the types and names of columns in the editor, and add columns, it’s actually possible to simplify this process still further. Just create a completely empty data frame and edit it:

```
> d = data.frame()
NULL data frame with 0 rows
> fix(d)
```

Best of all, you get fewer warning messages, other than the obvious one saying that you were creating a completely empty (NULL) data frame. You can change the names of the columns by clicking on the topmost cell (the one with the column name in it). That will let you change the name and also specify whether the column is numeric (continuous) or character (discrete).

4 Reading Data from Files

See the `read.table` function below in the example. A more difficult and complicated function is `scan` which can be useful in some specialised circumstances.

You should also look at the R Documentation, specifically the R Data Import/Export guide at <http://cran.r-project.org/doc/manuals/R-data.html>. Note that it is possible to read data produced by SPSS (via the `read.spss` function in the `foreign` package) and write data that SPSS can read via the `write.spss` function in the `foreign` package).

5 How to arrange the data going into R

While it is generally true that for R and SPSS each line should correspond to a trial and each column should correspond to a distinct measurement, that rule doesn’t always give a unique answer.

| | Name | Stimulus | Response |
|-----|--------|---|----------|
| 1 | George | Do dogs eat bananas? | Yes |
| 2 | George | Frogs deserve legal protection of their webbing. | Yes |
| 3 | George | He is very banana-like. | Yes |
| 4 | George | Frogs deserve legal protection for their webbing. | Yes |
| 5 | George | I am the apple of my eye. | Yes |
| 6 | Fred | Do dogs eat bananas? | Yes |
| ... | ... | ... | ... |

Table 3: Representing grammaticality judgements as a stimulus and response.

| | Name | Do dogs eat bananas? | Frogs deserve legal protection of their webbing | He is very banana-like | Frogs deserve legal protection for their webbing | I am the apple of my eye |
|-----|--------|----------------------|---|------------------------|--|--------------------------|
| 1 | George | Yes | Yes | Yes | Yes | Yes |
| 2 | Fred | Yes | Yes | Yes | No | Yes |
| ... | ... | ... | ... | ... | ... | ... |

Table 4: Representing grammaticality judgements as a survey.

For instance, consider an experiment where you put a question up on a screen, and the subject answers yes or no. Let's say that each subject gets 5 questions, and let's say that the task is to judge grammaticality of the sentences.

One way of looking at the experiment is as a stimulus and response. You give the subject a stimulus, and he or she produces a response. This might be best represented as in Table 3.

However, another entirely reasonable way of looking at the experiment is as a survey that has five questions. This would be entered as shown in Table 4.

6 Display of Pairs of Categorical Data

This example involves intonational phonology of English. A corpus of data was marked with the IViE [Grabe, 2001] intonational phonological labels.

```
Tone1 Tone2
1 BOF L*H
2 L*H L*H
```

```

3 L*H %
4 % EOF
5 BOF H*
6 H* !H*
7 !H* L*H
8 L*H L*H

```

Intonational Phonology: The ideas behind intonational phonology are

- Certain syllables in English (and other languages) are special. These are called “accented”, and they are marked “prosodically.” This means that there are some acoustically measurable changes to those syllables that do not change the identity of the phoneme. Normally, this is interpreted to mean changes to the fundamental frequency (f_0) of speech and the duration of the phonemes, but accented syllables are also consistently louder than non-accented syllables Kochanski et al. [2005].

- These accented syllables are believed to naturally fall into several categories, roughly analogous to phonemes. That is, they differ “phonologically”. (The term “phonology” actually is not particularly well defined. It is best defined in terms of the minimal acoustic differences between words: any change that converts one word (e.g. “pat”) into another word (e.g. “pit”) is termed phonological.

By analogy, we are talking here about changes in prosody that change the meaning or emotional content of a sentence from one distinct choice to another.³) From a practical point of view, these categories are simply the distinctions that linguists make; they probably amount to distinctions that people can easily make.

- These accents are believed to be combinable with some sort of grammar, so that some combinations are possible and others not. This grammar presumably depends upon the dialect of the speaker [Grabe, 2004, Grabe et al., 2005].
- The accents are expressed in terms of high and low tones, which correlate roughly with high and low pitch. So, an accent might be labelled as “H*L” to indicate that it starts high and goes low (the “*” indicates the position of the accented syllable). Other symbols are “%” to indicate the end of a phrase, “#” to indicate a speech error or restart, and “?” to indicate an un-classifiable accent. “BOF” and “EOF” marks also appear to indicate the beginning and end of a data file.

Figure 2 shows an interesting way of plotting pairs of categorical data.

The script to produce this plot follows (with annotations in italics):

The easiest kind of file that we can read into an R data frame has a one-line header containing the column labels (separated by whitespace). The remainder of the file is the data. The data lines start with a row counter, then show the data for each column, separated by spaces. (Thus, the data rows have one more column than the number of labels.) You can create this file with Notepad (or even with Microsoft Word, but be sure to save as plain text and watch out for line wrap). Watch out for the default comment character='#'. To read in the file sentences_bitones.txt, do this:

```
> f = read.table("data/sentences_bitones.txt", comment.char="")
```

Note that we had to add an extra argument, comment.char="". That's because by default, R interprets everything after a hash mark (#) as a comment rather than data. Normally, that's OK³ but here, some of our labels are hash marks and should not be interpreted as the beginnings of comments. (A “#” label means a speech error or re-start of a sentence.)

³It's OK, but pretty useless. If you have comments, you might as well put them into a comment column. If the comments indicate unusual operation of the experiment (e.g. “sample dropped on the floor”), it might be valuable to be able to select commented data

```
> # Alternatively, you can get the data direct from my web server:
> f = read.table("http://kochanski.org/gpk/teaching/06010xford/02sentences_bitones.txt",
+               comment.char="")
```

The summary function does different things for different kinds of data, with the intent of giving you a brief textual description of your data.

```
> summary(f)
Tone1      Tone2
BOF      :924  EOF      :924
%        :815  %        :815
H*L      :805  H*L      :805
H*       :626  H*       :626
L*H      :404  L*H      :404
!H*L     :185  !H*L     :185
(Other) :229  (Other) :229
```

So, we have an object, `f`, containing the data. What can we do with it? We can pull out one column at a time by using a dollar sign: `f$Tone1` means “take the `Tone1` column of the data frame called ‘`f`’.” The output turns out to be rather long and we only show a bit of it.

```
> f$Tone1
...
[3809] H*  H*L %   BOF H*  H*  L*H %   BOF H*  H*L H%  BOF L*
[3823] L*H %   BOF H*  L*H %   BOF H*  L*H %   BOF H*  H*  H%
...
Levels: !H* !H*L # % %H *? BOF H% H* H*L L% L* L*H
```

The numbers in square brackets are added by R and they tell you where you are in a long vector. This particular vector has 3988 entries.

```
> attach(f)
> Tone1
...
[3809] H*  H*L %   BOF H*  H*  L*H %   BOF H*  H*L H%  BOF L*
[3823] L*H %   BOF H*  L*H %   BOF H*  L*H %   BOF H*  H*  H%
...
Levels: !H* !H*L # % %H *? BOF H% H* H*L L% L* L*H
```

We can look at just a few of the tones by selecting a range (here, we’re selecting from the first element of `Tone1` along to the 10th element.

```
> Tone1[1:10]
[1] BOF L*H L*H %   BOF H*  !H* L*H L*H %
Levels: !H* !H*L # % %H *? BOF H% H* H*L L% L* L*H
>
> Tone1[1:30]
[1] BOF L*H L*H %   BOF H*  !H* L*H L*H %   BOF H*L L*H H*L %   BOF H*  L*H L*H
[20] %   BOF L*  L*H L*H %   BOF H*  L*H L*H %
Levels: !H* !H*L # % %H *? BOF H% H* H*L L% L* L*H
```

The [20] tells us that the 20th entry in the `Tone 1` vector is at the beginning of the second line of output, and it is “%”.

if only to see if it differs from the uncommented data. On the other hand, if the comments don’t indicate anything relevant (e.g. “Get eggs, milk, dry cleaning.”) why put them in the file at all? The only problem with a comment column is that comments often contain spaces, and (by default) R separates columns by any kind of white space. To get around this, you can put quotes around your comments if they contain spaces.

Now, we can look for pairings of tones using this strange but logical syntax:

```
> Tone2[Tone1=="L*H"]
...
[136] % % % % % % % % % % L*H % % L*H
[151] % % % % L* % % % % % L*H % L*H % %
[166] L*H H% % % L*H % L*H % % H*L H*L !H*L L*H L*H %
...
[391] % % H*L % % % H*L % % % L*H !H*L % H*L
Levels: !H* !H*L # % %H *? EOF H% H* H*L L% L* L*H
```

That tells us that “L*H” is often followed by “%”, and the last line (“Levels:”) tells us the complete set of possibilities that can follow a L*H accent.

We can count the length of that vector:

```
> length(Tone2[Tone1=="L*H"])
[1] 404
```

But, watch out! You might guess that `length(Tone1=="L*H")` is also 404, but not so. The expression `Tone1=="L*H"` is a vector of true/false values that is as long as the `Tone1` vector: 3988 entries.

```
> length(Tone1=="L*H")
[1] 3988
> Tone1=="L*H"
...
[3961] TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE
[3973] TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE
[3985] FALSE FALSE FALSE FALSE
```

Another easy thing to do is find out what fraction of the data is a particular tone like L*H:

```
> mean(Tone1=="L*H")
[1] 0.1013039
```

So, 10% of the tones are L*H. This works because TRUE counts as 1 and FALSE is also zero.

We can get more complicated here, looking at what tone follows a L*H, like this:

```
> mean(Tone2[Tone1=="L*H"]=="%")
[1] 0.5816832
```

So, 58% of the time, a L*H is followed by a “%” (end of phrase without a major pitch motion) symbol.

Create a table of frequencies:

```
> x = table(f)
> x
Tone2
Tone1 !H* !H*L # % %H *? EOF H% H* H*L L% L* L*H
!H* 3 22 0 2 0 0 0 0 6 0 2 15
!H*L 0 2 0 157 0 0 0 15 2 9 0 0 0
# 0 0 1 0 0 0 0 0 2 7 0 0 3
% 0 0 0 0 0 0 806 0 0 6 0 0 3
%H 0 0 0 0 0 0 0 0 10 3 0 0 1
*? 0 0 0 0 0 0 0 0 1 0 0 0 0
BOF 0 0 0 0 14 1 0 0 535 302 0 7 65
H% 0 3 0 0 0 0 109 0 0 8 0 0 0
```

| | | | | | | | | | | | | | |
|-----|----|-----|---|-----|---|---|---|----|----|-----|---|----|-----|
| H* | 46 | 116 | 7 | 5 | 0 | 0 | 0 | 16 | 46 | 203 | 0 | 10 | 177 |
| H*L | 1 | 39 | 4 | 416 | 0 | 0 | 0 | 43 | 26 | 207 | 0 | 2 | 67 |
| L% | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |
| L* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 1 | 3 | 0 | 0 | 6 |
| L*H | 0 | 3 | 1 | 235 | 0 | 0 | 0 | 34 | 3 | 51 | 9 | 1 | 67 |

Now, we can plot that frequency table several ways:

```
> mosaicplot(x, shade=TRUE, off=70)
```

A window now appears with Figure 2. The colors mark how big the entries are, relative to a simple statistical model where the Tone1 and Tone2 values are independent. In this example, they are not independent; while we won't go into the details of the model yet, there are strong correlations between neighbouring tones (as one might expect).

Alternatively, if you believe that the probabilities should be independent and you wanted to see whether that assumption was wrong and how it was broken, you could use `assocplot()`. That shows deviations from statistical independence. There are also reputed to be useful functions in the “vcd” package.

References

- Esther Grabe. IViE labelling guide. Manual, Oxford University Phonetics Laboratory, Oxford, UK, 2001. URL <http://www.phon.ox.ac.uk/~esther/ivyweb/guide.html>. Version 3; available at <http://www.phon.ox.ac.uk/~esther/ivyweb/guide.html>.
- Esther Grabe. Intonational variation in urban dialects of english spoken in the british isles. In P. Gilles and J. Peters, editors, *Regional Variation in Intonation*, Linguistische Arbeiten, pages 9–31. Niemeyer, Tuebingen, 2004.
- Esther Grabe, Greg Kochanski, and John Coleman. The intonation of native accent varieties in the british isles – potential for miscommunication? In Katarzyna Dziubalska-Kolaczyk and Joanna Przedlacka, editors, *English pronunciation models: a changing scene.*, Linguistic Insights Series. Peter Lang, 2005.
- G. Kochanski, E. Grabe, J. Coleman, and B. Rosner. Loudness predicts prominence: Fundamental frequency lends little. *J. Acoust. Soc. of America*, 118(2):1038–1054, August 2005. URL <http://dx.doi.org/10.1121/1.1923349>.

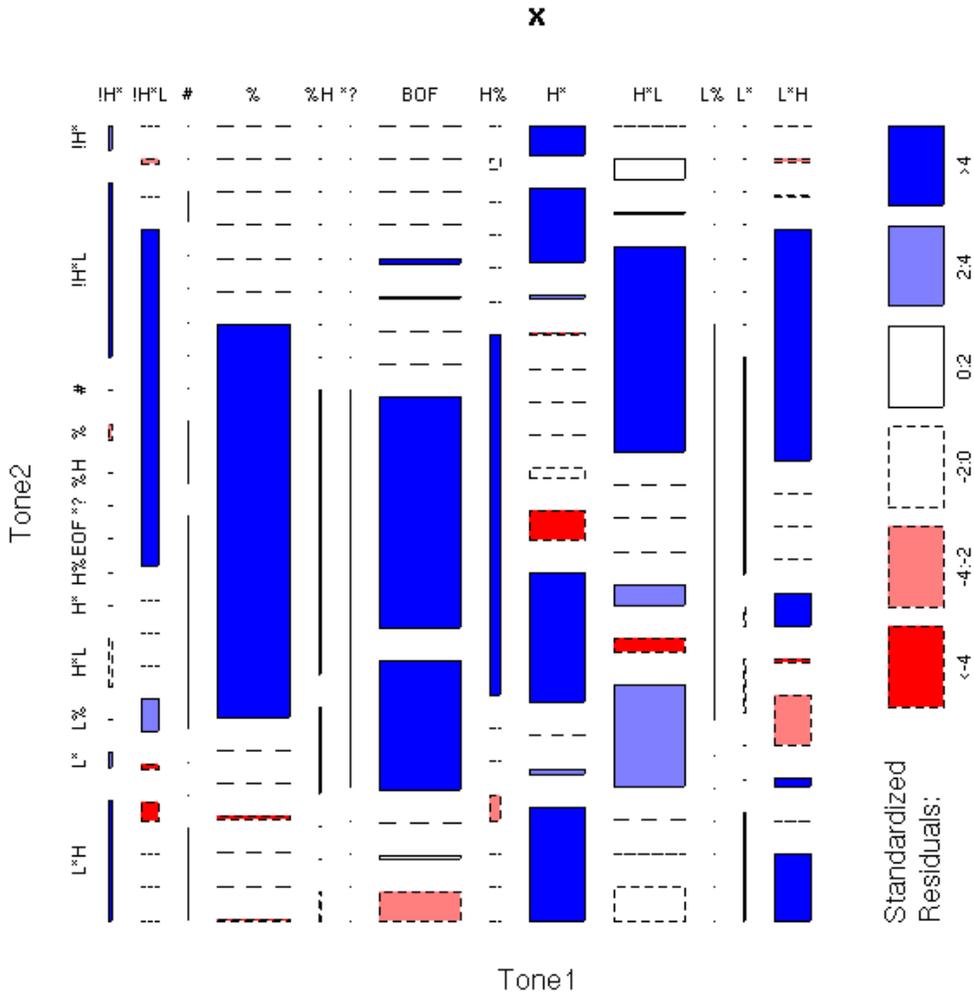


Figure 2: This is a mosaic plot of neighbouring pairs of accents in speech. There are 13 types of symbols in the data, and this plot shows how often different accents appear next to each other. The box areas are proportional to the probability of observing the pair of accents, and the coloring shows how the probabilities differ from what one would expect if the neighbouring accents were independent of each other. Red boxes are substantially less probable than one might expect and blue boxes are substantially more probable than one might expect.