CGIwithR: Facilities for processing web forms using *R*

David Firth
Nuffield College, Oxford, United Kingdom

http://www.stats.ox.ac.uk/~firth

Version 0.50 (5 May 2003)

Abstract

CGIwithR is a package for use with the *R* statistical computing environment, to facilitate processing of information from web-based forms, and reporting of results in the Hypertext Markup Language (HTML), through the Common Gateway Interface (CGI). *CGIwithR* permits the straightforward use of *R* as a CGI scripting language. This paper serves as an extended user manual for *CGIwithR*, supplementary to the *R* help pages installed with the package.

Key words: online calculator.

1 Introduction

Information entered on web-based forms is often processed by a *CGI script*, a program called by the web server which takes the data from the form as its input and typically returns its output encoded in HTML for display on the client browser. CGI scripts can be written in any suitable programming language; *Perl* is especially popular. The purpose of *CGIwithR* is to enable use of the *S* language, as implemented in the open-source *R* project (Ihaka and Gentleman, 1996), to write CGI scripts. This is useful for at least two reasons: (i) to make easily available the extensive statistical facilities of *R* in CGI scripts, and (ii) to allow statisticians and others familiar with the *S* language to write form-handlers without having first to learn another language such as *Perl*.

The *CGIwithR* package has two main components:

- a set of *R* functions for decoding the form data and for report-writing in HTML;
- a 'wrapper' shell script, `R.cgi`, which is called by the web server, and which in turn calls *R* to process the form data and compose the resulting HTML page.

The `R.cgi` wrapper should work on most Unix-like operating systems. It ought to be possible to adapt it for use also in conjunction with web servers running under Windows NT and its successors, but the author has no plans for that. The *R* functions for handling CGI data and for writing output in HTML are platform-independent.

At least two other existing projects, *Rcgi* and *Rweb*, make *R* available via a web interface; see 'R Web Interfaces' in *R*'s *Frequently Asked Questions* document for more information. The emphasis of *CGIwithR* is on making *R* available as a scripting language, rather than on providing a user interface to *R*: see section 5.3 below for some further related remarks.

The *CGIwithR* software is provided free under the terms of the Gnu Public License, version 2 or later. There is absolutely no warranty: *CGIwithR* is not fit for any purpose, and any use made of it is entirely at the risk of the user.

2 Configuration and installation

The *CGIwithR* package is available on the Comprehensive R Archive Network (CRAN), and should be installed locally in the same way as other *R* packages. Various methods are available for this; see *R*'s "Installation and Administration" documentation for up-to-date information.

The `inst` subdirectory of the *CGIwithR* package contains the further subdirectories

`cgi-bin`

`doc`

`examples`

To make CGI scripts written in *R* available to the local web server, the files `cgi-bin/.Rprofile` and `cgi-bin/R.cgi` must be moved or copied to a directory in which the web server is permitted to look for CGI executables (such directories are often named `cgi-bin`); this may need the help of a system administrator. The first few lines of `R.cgi` should be edited to set certain configuration options, in particular to locate *R* on the local system, and to locate the local installation of *Ghostscript* if graphical output is to be allowed. Full configuration instructions are contained in `R.cgi` itself. File permissions should be set such that `.Rprofile` is readable by the web server, and `R.cgi` is both readable and executable by the web server. For example, if those two files have been placed in directory `/var/www/cgi-bin`, the necessary permissions can be set (by the system administrator) as follows:

```
chmod a+r /var/www/cgi-bin/.Rprofile
chmod a+rx /var/www/cgi-bin/R.cgi
```

3 Using CGIwithR

3.1 How forms call scripts

Suppose that the URL for the directory permitted to contain CGI scripts is `/cgi-bin`, say. (There may be more than one such directory available at any given installation; this depends on the local configuration of the web server.) Then any web page containing form elements can pass its data to a script written in *R* (let us call such a script `myscript.R`) by using a `<FORM>` tag like the following:

```
<FORM action="/cgi-bin/R.cgi/myscript.R" method="POST">
```

This passes the data first to the `R.cgi` script, which in turn passes it to *R* for processing by `myscript.R`. Note that `R.cgi` here is *not* a directory. In general, the parts of a URL between the slashes are not directory names, but symbols to tell the HTTP server where to find a particular document. In most cases, the server will look for a directory of the same name as the symbol in question, and if there is one, go into that directory and proceed to the next symbol. But if the server sees that a particular symbol matches the name of an executable *program* (here `R.cgi`), it will run that program and pass the remainder of the URL (which here is simply `myscript.R`) to the program as an argument.

The form data are always made available in *R* as an object named `formData`, a list whose components have the names of data items collected in the form; see section 5 below for some examples. In most applications, each data component has a unique name which is specified in the HTML code for the form (using `NAME=`). An exception to this is the use of a 'multiple select' box in the HTML form, for example

```
<SELECT NAME="ABCpicker" SIZE=4 MULTIPLE>
  <OPTION>A
  <OPTION>B
  <OPTION>C
</SELECT>
```

where more than one of A,B,C may be chosen. In this case, formData may end up having more than one component named "ABCpicker". Using formData$ABCpicker as normal would extract only the *first* such component. To get *all* of the options selected in a 'multiple select' box, one could for example use unlist(formData[names(formData)=="ABCpicker"]) to make a character vector containing all of the selections made.

A CGI script called by R.cgi can contain any valid *R* expressions, which are evaluated and the results passed back directly to the web server.

The instance of *R* that is called to do the processing is either

1. a path specified on the first line of the CGI script, in the form

   ```
   #! /usr/local/bin/R
   ```

2. if no such path is specified (that is, the first line of the script does not begin with "#!"), the *R* specified as configuration variable R_DEFAULT in the wrapper script R.cgi.

3.2 Some scripting tools

CGIwithR defines the following functions for use in accessing form entries and for writing output in HTML. Their use is demonstrated in some examples below.

- scanText: a function for converting a free text entry into a list of words or numbers.
- tag, untag, br, mailto, linkto, img: these are all functions which output HTML tags.
- indentPrint: a print method which indents the display of an *R* object by a specified number of characters.
- lf, comments: these allow some basic formatting and commenting of HTML code if desired.
- webPNG: a graphics device wrapper for sending plots to a specified location on the web server for display.

4 Security

CGI scripts are potentially very insecure. Only trusted users of a system should be allowed to write and maintain CGI scripts that run on it; here 'trusted' implies not only lack of malevolence, but the ability to identify potential security problems and eliminate them. CGI scripts written in *R* are no less risky than those written in other languages such as *Perl*. An example of a script which would almost certainly open up the host system to attack, and which therefore must not be used, is described in section 5.3 below.

The configuration variables R_NICE and MAX_DATA_LENGTH can be set in R.cgi to provide some, albeit very limited, control over the system resources that may be demanded by CGI scripts written in *R*.

5 Examples

The example files referred to below are all in the inst/examples directory of the *CGIwithR* package.

5.1 A simple example

The screenshot in Figure 1 is a browser's view of trivial.html, which is a form whose entries are processed by a script called trivial.R. This script demonstrates the basic features of *CGIwithR*. The output is shown in Figure 2.

A handler script such as trivial.R constructs the HTML response to a CGI request. General HTML tags can be generated readably in *R* by using tag and untag, as for example

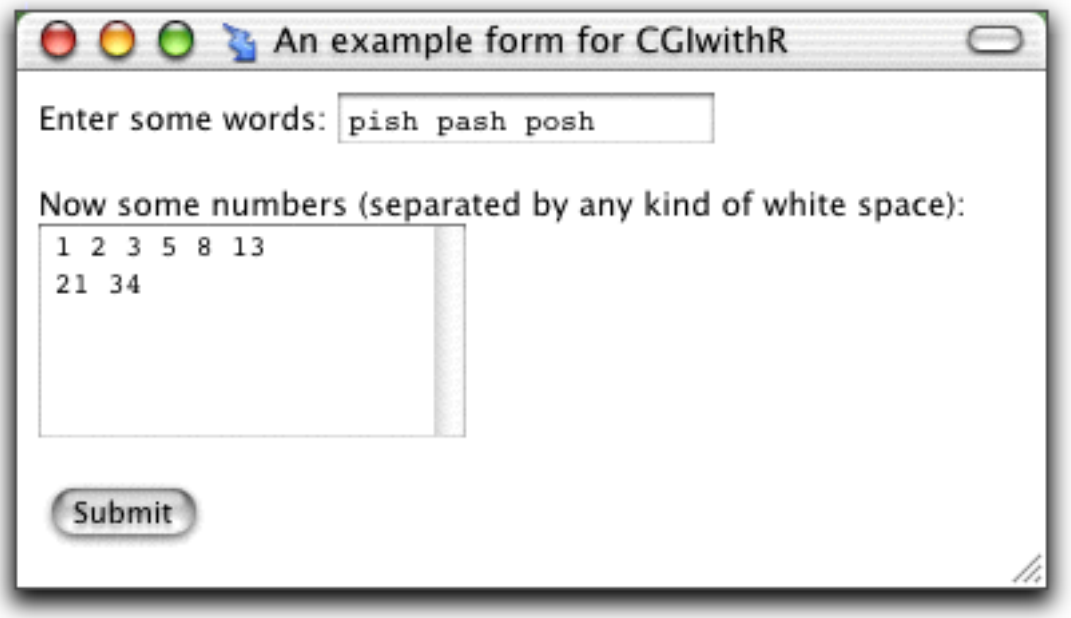

Figure 1: The form created by `trivial.html`, with some data entered.

```
tag(BODY, bgcolor = "yellow")
    lf(2)
    tag(h3)
        cat("A large heading")
    untag(h3)
```

Note that optional arguments to HTML tags, such as the `bgcolor` argument here, are specified in an obvious way.

The form data are made available through a list in *R* called `formData`. In the example the supplied data are just sent back with minimal processing:

```
    tag(p)
        cat("Your words in italics:")
        tag(i)
            cat(formData$words)
        untag(i)
    untag(p)
    lf(2)
    tag(p)
        cat("Your numbers:")
        tag(pre)
            numbers <- as.numeric(scanText(formData$numbers))
            print(numbers)
        untag(pre)
    untag(p)
```

The use of `scanText` here converts the form's text field into a vector. The expression `lf(2)` simply inserts two line feeds into the resultant HTML for readability.

Graphics can be included using the function `img`. In the example, a plot in PNG format is constructed and displayed by

```
    cat("Here is a graph:") ; br()
```

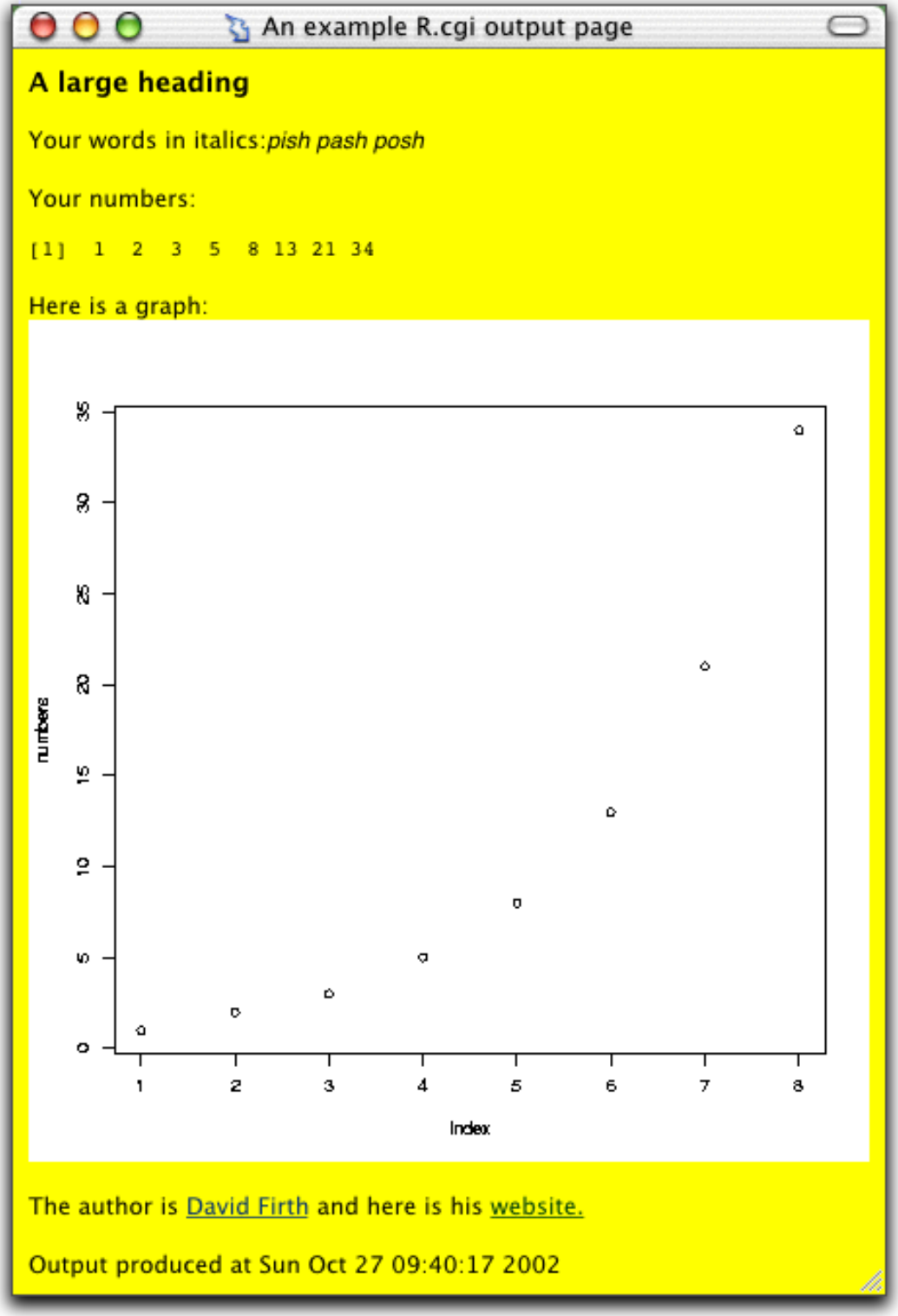

Figure 2: The results from `trivial.html`

```
graphDir <- "/Users/david/Sites/graphs/"
graphURLroot <- "/~david/graphs/"
webPNG("temp.png")
plot(numbers)
img(src = "temp.png") ; br(2)
```

where the `br` function is used to insert HTML linebreaks. The graphics device wrapper `webPNG` specifies a file to which graphs are sent. In order for this to work, the web server (which owns the *R* process) must be given permission to write to a web-served directory specified as `graphDir`. For the example above, this could be set up by doing

```
mkdir /Users/david/Sites/graphs
chmod a+wx /Users/david/Sites/graphs
```

The additional variable `graphURLroot` gives the URL for HTTP access to that directory.

Two special functions `linkto` and `mailto` are provided to make hyperlinks easy. In the example they are used as follows:

```
cat("The author is ")
mailto("David Firth", "david.firth@nuffield.ox.ac.uk")
cat(" and here is his ")
linkto("website.", "http://www.stats.ox.ac.uk/~firth/") ; br()
```

5.2 The online 'QV calculator'

The motivation for writing *CGIwithR* came from a need to re-implement in *R* the *QV Calculator* (Firth, 2000), which was written originally in *Xlisp-Stat*. The *QV Calculator* takes as its data the estimated variance-covariance matrix for a set of parameter estimates in a statistical model, and reports a set of 'quasi standard errors' which in many instances are more informative than 'conventional' standard errors which relate to a specific model paramaterization. For details of the methodology see Firth (2000). The *QV Calculator* currently uses *R*, suitably equipped with the contributed package `qvcalc` (also at CRAN), for its computation.

The *QV calculator* input form can be found online at

http://www.stats.ox.ac.uk/~firth/qvcalc

The underlying CGI script `qvweb.R` is provided, for reference purposes, in the `inst/examples` directory of the *CGIwithR* package. One general feature worth noting is the use of `indentPrint` to return pre-formatted output to the browser with a specified number of leading spaces added. For example, in `qvweb.R`, the main table of results, which is computed as a dataframe `qv$qvframe` in *R*, is output as

```
tag(pre)
    indentPrint(qv$qvframe, indent = 6, digits = 3)
untag(pre)
```

so that it appears nicely placed on the output page.

5.3 A foolish and dangerous example

It is not difficult to write CGI scripts in *R* that open up the server to attack: *R* is very general, and includes facilities for access to the host operating system. It would be unwise to allow unauthorised use of such facilities.

As an example WHICH SHOULD NOT BE USED, the pair of files `dangerous.html` and `dangerous.R` show how to implement a web-based terminal interface to *R* using *CGIwithR*. Figures 3 and 4 show sample input and output pages. This example is provided only to show how such an interface *can* be constructed. IT SHOULD NOT BE USED. Making all of the facilities of *R* available in this way would most likely allow outsiders to access and/or damage parts of the host system.

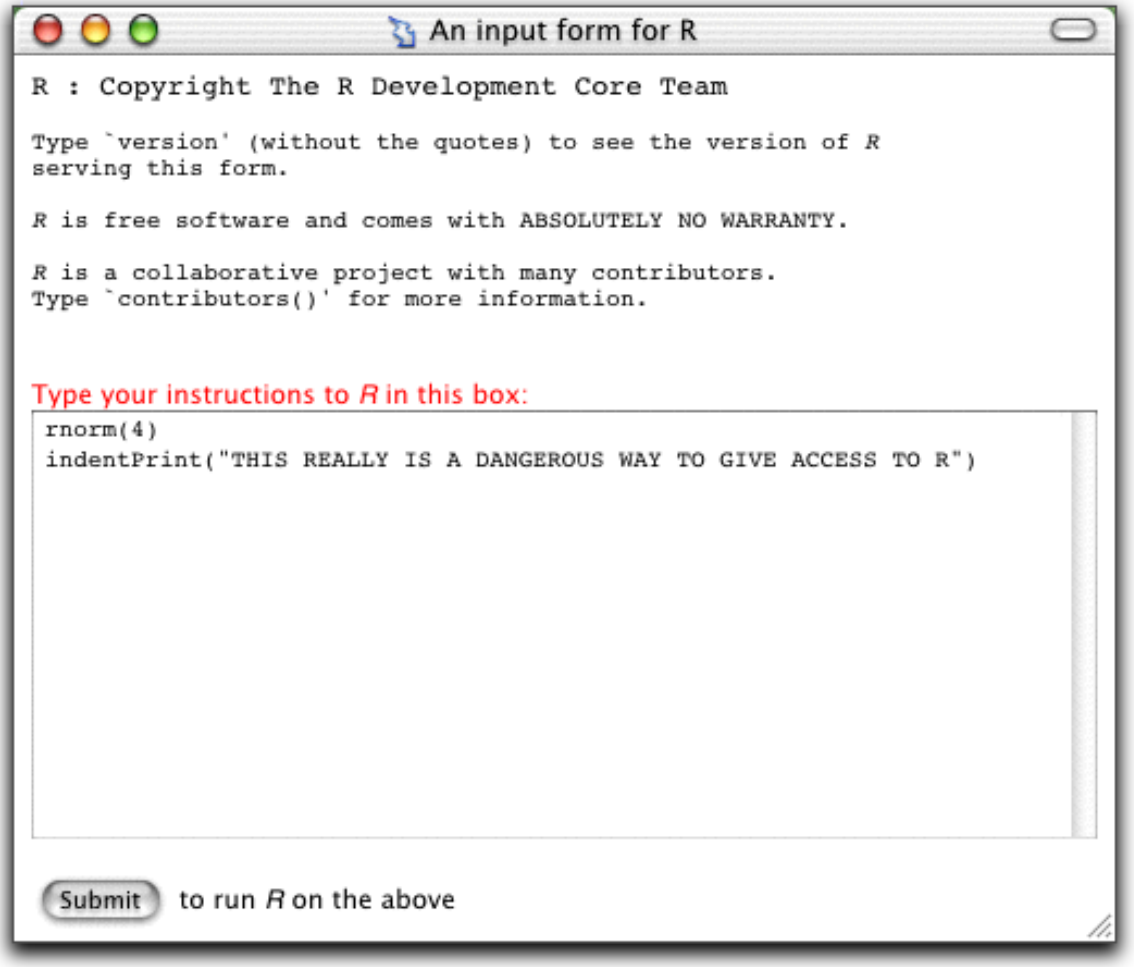

Figure 3: The *R* input form made by `dangerous.html`

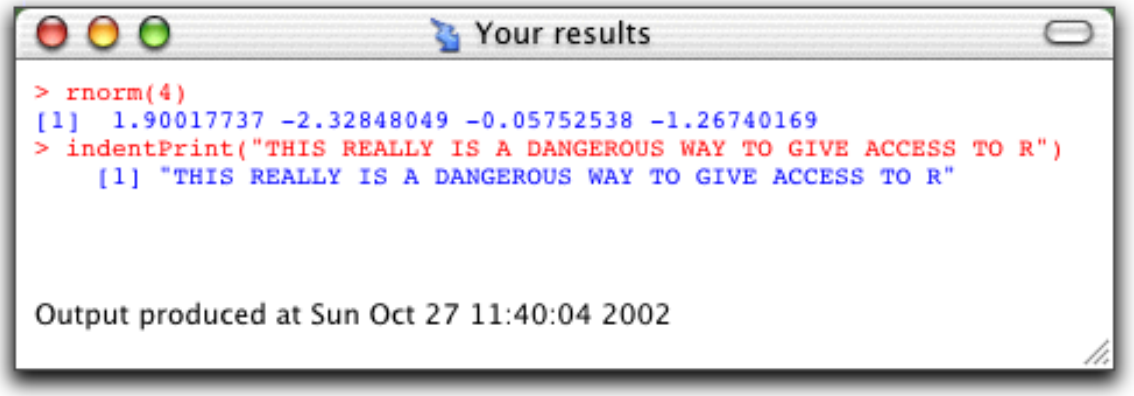

Figure 4: Output as delivered by `dangerous.R`

For ways of making *R* available via the web see instead the *Rcgi* and *Rweb* projects referred to in the 'R Web Interfaces' section of *R*'s *Frequently Asked Questions* document.

6 Some technical details

6.1 System compatibility

The 'wrapper' shell script `R.cgi` should work with most Bourne-family shells (including `sh, bash, ksh, zsh`) on Unix-like systems. The author is aware of success in using *CGIwithR* on Mac OS X, Linux and Solaris systems with the Apache web server, and would appreciate reports of success or failure on other systems.

6.2 Decoding form data

The data entered on HTML forms is passed in an encoded string which is not suitable for direct manipulation in *R*. The `formData` variable made available to all CGI scripts written in *R* is a decoded version created in the `.First.lib` function of the *CGIwithR* package. An alternative and widely used decoding device is the `uncgi` program by Steven Grimm. The design of *CGIwithR* is such that it is compatible with `uncgi`: if `uncgi` is used, no further decoding is done by *CGIwithR*.

6.3 The .Rprofile file

As distributed in the `inst/cgi-bin` directory, the file `.Rprofile` contains a single *R* command,

```
library(CGIwithR, warn.conflicts=FALSE)
```

which makes the *CGIwithR* package available in the instance of *R* started by the web server when responding to a CGI request. For this to work, the `.Rprofile` file needs to be placed in the same location on the web server as the shell script `R.cgi`.

Acknowledgement

The author is grateful to pod at Oxford University Computing Services for some very helpful insights on the workings of shell scripts, to Henrik Bengtsson (via Jean-Pierre Müller) for an ASCII/hexadecimal translation table, and to two anonymous *Journal of Statistical Software* reviewers for helpful comments. Thanks also to Steven Grimm for making available `uncgi`, which gave some important clues.

References

Firth, D. (2000). Quasi-variances in Xlisp-Stat and on the web. *Journal of Statistical Software* **5.4**, 1–13.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics* **5**, 299–314.