



Elements of R for Genetics and Bioinformatics

**Thomas Lumley
Ken Rice**

Universities of Washington and Auckland

Seattle, July 2015

Introduction: Course Aims

- Under the hood of R;
 - R essentials, and programming skills
 - Examples implement common genetic analyses, efficiently
- Using R for more sophisticated analyses;
 - Using tools, packages produced by others
 - Bioconductor-based bioinformatics

While no R knowledge is assumed, we will move quickly – and knowing *either* some programming or some genetics will help a lot – also trying our coding ideas on your data, later.

If you are new(er) to R and need help with e.g. reading in data, manipulating data, understanding help files, *please ask* Thomas, Ken, or someone else in class, during hands-on periods.

Introduction: About Prof Lumley



- Professor, Univ of Auckland
- R Core developer
- Genetic/Genomic research in Cardiovascular Epidemiology
- Sings bass

Introduction: About Prof Rice



- Associate Prof, UW Biostat
- Not an author, but a user (and a teacher)
- Genetic/Genomic research in Cardiovascular Epidemiology
- Sings bass!

... and you?

(who are you, what area of genetics, what are you looking for from the course)

Introduction: Course structure

10 sessions over 2.5 days

- Day 0.5; Review of R, graphics
- Day 1; More graphics, data-wrangling, programming (loops),
- Day 2; Big data, bioconductor, more bioconductor, interfacing with other code

Web page: <http://faculty.washington.edu/kenrice/sig/>

Introduction: Session structure

- 45 mins teaching (questions welcome – please interrupt!)
- 30 mins hands-on
- 15 mins summary, discussion

Keys are posted after each session – though there is more than one correct answer



1. Review of R

Ken Rice
Thomas Lumley

Universities of Washington and Auckland

Seattle, July 2015

What is R?

R is a 'programming environment for statistics and graphics'

The base R has fewer prepackaged statistics procedure than SPSS or SAS, but it is much easier to extend with new procedures.

There are over 6000 published extension packages for R, many aimed at genetics and genomics research.

Using R

R is a free implementation of S, for which John Chambers won the ACM Software Systems award.

For the S system, which has forever altered how people analyze, visualize, and manipulate data.

The downside is that using R effectively may require changing how *you* analyze, visualize, and manipulate data.

R is a command-line system, not a point-and-click system.

A calculator

```
> 2+2
[1] 4
> 1536/317000
[1] 0.004845426
> exp(pi)-pi
[1] 19.9991
> x <- 3
> y <- 2
> x+y
[1] 5
> ls()
[1] "x" "y"
> round(pi, 6)
[1] 3.141593
> round(pi,
+ 6)
[1] 3.141593
```

Scripts

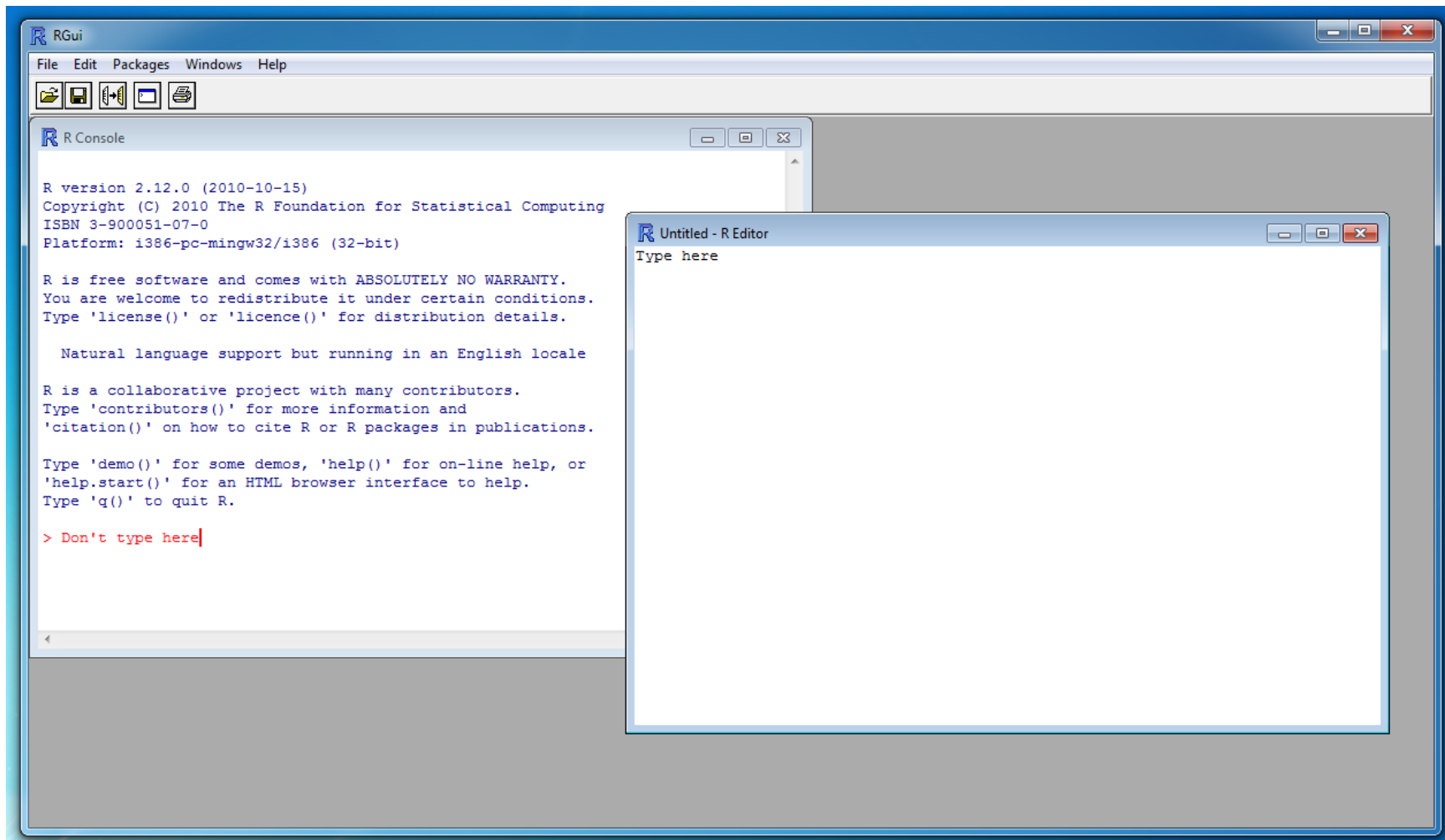
For longer analyses (and for this course), it's better to type code into a script and then run it. In base R;

- Windows: *File | New script*, CTRL-R to run lines or regions
- Mac: *File | New Document*, command-return to run
- Some other text editors offer this: Emacs, Tinn, WinEDT, JGR, Eclipse. RStudio has a script editor *and* data viewer

Please note we *strongly* recommend doing the exercises in pairs

Scripts

Default interface (and how to use it);



Reading data

Ability to read in data is assumed. But to illustrate commands, and show some less-standard approaches, we'll review reading in;

- Text files
- Other statistics packages datasets
- Web pages

Much more information is in the [Data Import/Export](#) manual.

Reading text data

The easiest format has variable names in the first row

case	id	gender	deg	yrdeg	field	startyr	year	rank	admin
1	1	F	Other	92	Other	95	95	Assist	0
2	2	M	Other	91	Other	94	94	Assist	0
3	2	M	Other	91	Other	94	95	Assist	0
4	4	M	PhD	96	Other	95	95	Assist	0

and fields separated by spaces. In R, use

```
salary <- read.table("salary.txt", header=TRUE)
```

to read the data from the file `salary.txt` in the current working directory into the data frame `salary`.

Syntax notes

- Spaces in commands don't matter (except for readability), but Capitalisation Does Matter.
- `TRUE` (and `FALSE`) are logical constants
- Unlike many systems, R does not distinguish between commands that do something and commands that compute a value. Everything is a function: i.e. it returns a value.
- Arguments to functions can be named (`header=TRUE`) or unnamed (`"salary.txt"`)
- A whole data set (called a `data frame` is stored in a variable (`salary`), so more than one dataset can be available at the same time.

Did it work?

The `head()` function shows the first few lines of the data frame

```
> head(salary)
```

```
  case id gender  deg yrdeg field startyr year  rank admin salary
1    1  1     F Other   92 Other    95   95 Assist    0   6684
2    2  2     M Other   91 Other    94   94 Assist    0   4743
3    3  2     M Other   91 Other    94   95 Assist    0   4881
4    4  4     M  PhD   96 Other    95   95 Assist    0   4231
5    5  6     M  PhD   66 Other    91   91  Full    1  11182
6    6  6     M  PhD   66 Other    91   92  Full    1  11507
```

It *should* look like this!

The `View()` command similarly shows your data frame, in a manageable way.

Where's my file?

- Find out your current directory with `getwd()`
- Change directory with the menus
- or with `setwd("I:/like/this/directory/better")`
- `file.choose()` pops up a dialog for choosing a file and returns the name, so

```
salary <- read.table(file.choose(), header=TRUE)
```

Reading text data

Sometimes columns are separated by commas (or tabs)

```
Ozone,Solar.R,Wind,Temp,Month,Day
41,190,7.4,67,5,1
36,118,8,72,5,2
12,149,12.6,74,5,3
18,313,11.5,62,5,4
NA,NA,14.3,56,5,5
```

Use

```
ozone <- read.table("ozone.csv", header=TRUE, sep=",")
```

or

```
ozone <- read.csv("ozone.csv")
```

Syntax notes

- Forgetting `header=TRUE` in `read.table()` is bad (try it!)
- Functions can have optional arguments (`sep` wasn't used the first time). Use `help("read.table")` or `?read.table` for a complete description of the function and all its arguments.
- There's more than one way to do it.
- `NA` is the code for missing data. Think of it as “Don't Know”. R handles it sensibly in computations: eg `1+NA`, `NA & FALSE`, `NA & TRUE`. You cannot test `temp==NA` (Is temperature equal to some number I don't know?), so there is a function `is.na()`.

Reading text data

Sometime the variable names aren't included

1	0.2	115	90	1	3	68	42	yes
2	0.7	193	90	3	1	61	48	yes
3	0.2	58	90	1	3	63	40	yes
4	0.2	5	80	2	3	65	75	yes
5	0.2	8.5	90	1	2	64	30	yes

and you have to supply them

```
psa <- read.table("psa.txt", col.names=c("ptid","nadirpsa",  
    "pretxpsa", "ps","bss","grade","age",  
    "obstime","inrem"))
```

or

```
psa <- read.table("psa.txt")  
names(psa) <- c("ptid","nadirpsa","pretxpsa", "ps",  
    "bss","grade","age","obstime","inrem")
```

Syntax notes

- Assigning a single vector (or anything else) to a variable uses the same syntax as assigning a whole data frame.
- `c()` is a function that makes a single vector from its arguments.
- `names()` is a function that accesses the variable names of a data frame
- Some functions, such as `names()`, can be used on the LHS of an assignment.

Other statistical packages

```
library("foreign")
stata <- read.dta("salary.dta")
spss <- read.spss("salary.sav", to.data.frame=TRUE)
sasxport <- read.xport("salary.xpt")
epiinfo <- read.epiinfo("salary.rec")
```

Notes:

- Many functions in R live in optional **packages**. The `library()` function lists packages, shows help, or loads packages from the package library.
- The **foreign** package is in the standard distribution. It handles import and export of data. Thousands of extra packages are available at <http://cran.r-project.org>.

The web

Files for `read.table()` can live on the web

```
f12000<-read.table("http://faculty.washington.edu/tlumley/  
data/FLvote.dat", header=TRUE)
```

It's also possible to read from more complex web services (such as the genome databases)

Operating on data

We assume you know how to use the \$ sign, to indicate columns of interest in a dataset, e.g. `antibiotics$duration` means the variable `duration` in the data frame `antibiotics`.

```
## This is a comment
## Convert temperature to real degrees
antibiotics$tempC <- (antibiotics$temp-32)*5/9
## display mean, quartiles of all variables
summary(antibiotics)
```


Subsets

Everything in R is a vector (but some have only one element).
There are several ways to use `[]` to extract subsets

```
## First element
antibiotics$temp[1]
## All but first element
antibiotics$temp[-1]
## Elements 5 through 10
antibiotics$temp[5:10]
## Elements 5 and 7
antibiotics$temp[c(5,7)]
## People who received antibiotics (note ==)
antibiotics$temp[ antibiotics$antib==1 ]
## or
with(antibiotics, temp[antib==1])
```

Notes

- Positive indices select elements, negative indices drop elements
- `5:10` is the sequence from 5 to 10
- You need `==` to test equality, not just `=`
- `with()` temporarily sets up a data frame as the default place to look up variables.

More subsets

For data frames you need two indices – naming rows and columns can also be useful;

```
## First row
antibiotics[1,]
## Second column
antibiotics[,2]
## Some rows and columns
antibiotics[3:7, 2:4]
## Columns by name
antibiotics[, c("id","temp","wbc")]
## People who received antibiotics
antibiotics[antibiotics$antib==1, ]
## Put this subset into a new data frame
yes <- antibiotics[antibiotics$antib==1,]
```

Computations

```
mean(antibiotics$temp)
median(antibiotics$temp)
var(antibiotics$temp)
sd(antibiotics$temp)
mean(yes$temp)
mean(antibiotics$temp[antibiotics$antib==1])
with(antibiotics, mean(temp[sex==2]))
toohot <- with(antibiotics, temp>99)
mean(toohot)
```

Factors

Factors represent categorical variables. You can't do mathematical operations on them (except for `==`)

```
> table(salary$rank,salary$field)
```

```
          Arts Other Prof
Assist   668 2626  754
Assoc   1229 4229 1071
Full     942 6285 1984
```

```
> antibiotics$antib<-factor(antibiotics$antib,
                             labels=c("Yes","No"))
```

```
> antibiotics$agegp<-cut(antibiotics$age, c(0,18,65,100))
```

```
> table(antibiotics$agegp)
(0,18]  (18,65]  (65,100]
      2         19         4
```

Help!

- `?fn` or `help(fn)` for help on `fn`
- `help.search("topic")` for help pages related to `"topic"`
- `apropos("tab")` for functions whose names contain `"tab"`
- `RSiteSearch("FDR")` to search the R Project website (requires internet access)