Introduction to R

Session 1 – Introduction

Statistical Consulting Centre

consulting@stat.auckland.ac.nz
The Department of Statistics
The University of Auckland

19 July, 2017

SCIENCE
DEPARTMENT OF STATISTICS

Wednesday

Each session comprises two parts: lecture and practice.

Session	Time	Session
1	09:00am - 10:30am	Introduction
	10:30am - 11:00am	Break
2	11:00am - 12:30pm	Data subsetting
	12:30pm - 01:30pm	Lunch break
3	01:30pm - 03:00pm	Data manipulation
	03:00pm - 03:30pm	Break
4	03:30pm - 05:00pm	Data exploration

Thursday

Each session comprises two parts: lecture and practice.

Session	Time	Session
1	09:00am - 10:30am	Graphics
	10:30am - 11:00am	Break
2	11:00am - 12:30pm	Advanced Graphics (ggplot2)
	12:30pm - 01:30pm	Lunch break
3	01:30pm - 03:00pm	Simple analysis
	03:00pm - 03:30pm	Break
4	03:30pm - 05:00pm	Advanced analysis

R and UoA's Department of Statistics

- R was initially written by Robert Gentleman and Ross Ihaka *R & R* of the **Department of Statistics, University of Auckland**.
- Three members of the *R Development Core Team* are in UoA's Department of Statistics.

R and UoA's Department of Statistics

Ross Ihaka and Robert Gentleman

R and UoA's Department of Statistics

Paul Murrell and Thomas Lumley

R and UoA's Department of Statistics

What does this mean?

If you want to learn R, you are talking to the right people!

Chris Triggs
Director Consulting Services
Phone: +64 9 373 7599 ext 88856
Email: triggs@stat.auckland.ac.nz

For more information, please see Chris's profile.

Yannan Jiang
Senior Research Fellow
Phone: +64 9 373 7599 ext 84725
Email: y.jiang@auckland.ac.nz

For more information, please see Yannan's profile.

Kathy Ruggiero
Senior Lecturer
Phone: +64 9 373 7599 ext 89456
Email: k.ruggiero@auckland.ac.nz

For more information, please see Kathy's profile.

Jessica McLay
Research Fellow
Phone: +64 9 373 7599 ext 73678 or 85313
Email: jessica.mclay@auckland.ac.nz

For more information, please see Jessica's profile.

Rachel Chen
Research Fellow
Phone: +64 9 373 7599 ext 89384
Email: rachel.chen@auckland.ac.nz

For more information, please see Rachel's profile.

Avinesh Pillai
Research Fellow
Phone: +64 9 373 7599 ext 82368 (Mon-Wed) or ext 81169 (Thurs & Fri)
Email: a.pillai@auckland.ac.nz

For more information, please see Avinesh's profile.

Statistical Consulting Centre

The School of Biological Sciences (SBS) has a contract with the Statistical Consulting Centre (SCC) to provide statistical support to staff and postgraduate students of SBS.

`https://www.stat.auckland.ac.nz/consulting/meet-us/any1_uoa/appointment_scheduler_kevin`

What is 'R'?

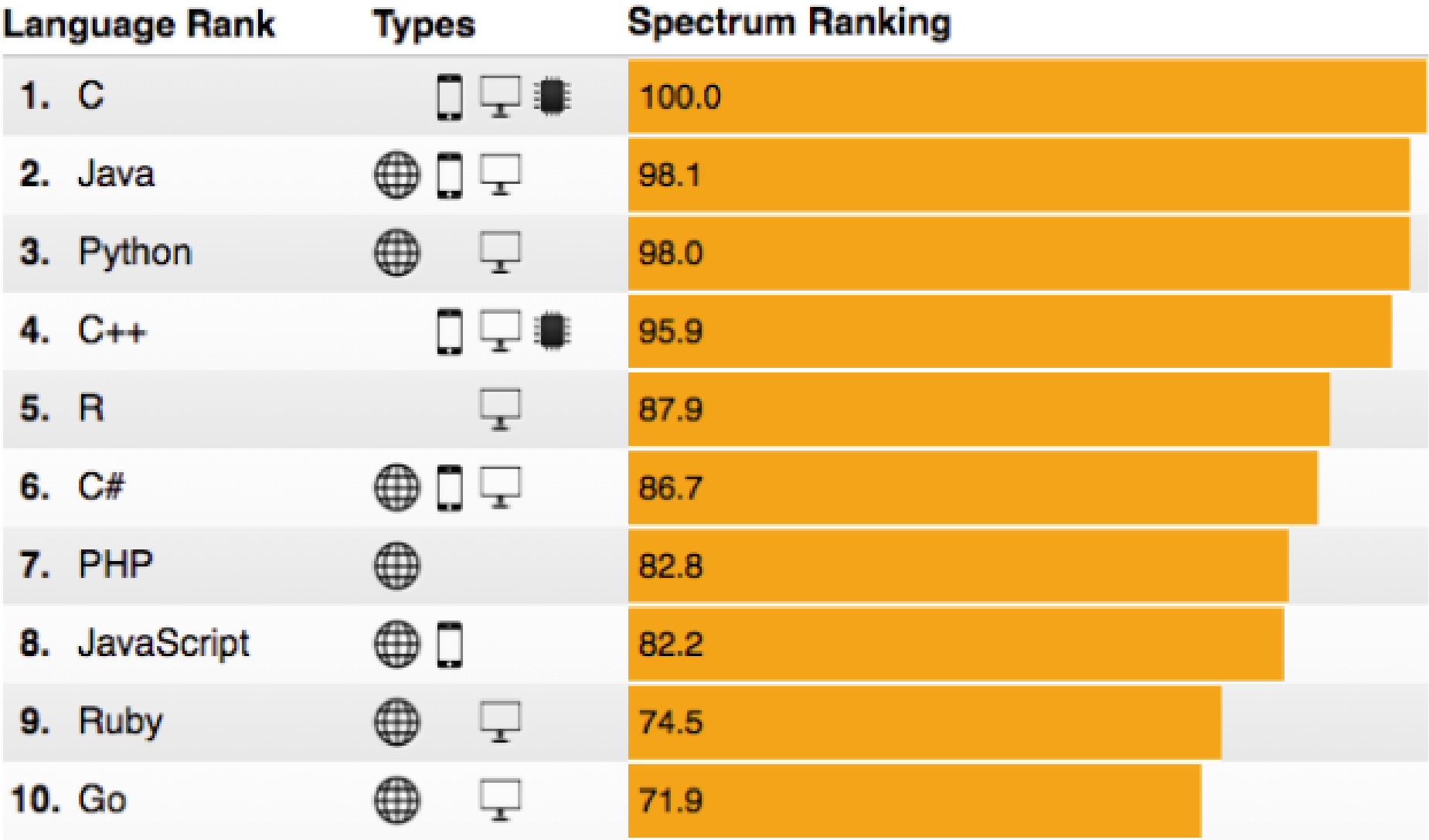
What does this mean?

R is a free software environment for statistical computing and graphics"

Key words:

- FREE!
- Statistical computing
- Graphics (much more flexible than SAS, SPSS, JMP, etc.)
- Support from communities of different fields, i.e. R packages. https://cran.r-project.org/web/views/.
- Even Microsoft is in it: Microsoft R Open. https://mran.microsoft.com/open/.
- https://www.slideshare.net/RevolutionAnalytics/r-then-and-now

What is R? (IEEE Spectrum's ranking 2016)



What is 'R'?

What does this mean?

R is a free software environment for statistical computing and graphics"

R and the biological sciences:

- Many *applications of statistical methods to biological datasets are implemented in R*
- These R *packages* are publically available on the web for immediate download and use.
- Bioconductor) https://www.bioconductor.org/.
- E.g. Next Generation Sequencing, Genomics.

How to download and install R

1. Go to the CRAN (Comprehensive R Archive Network) `cran.stat.auckland.ac.nz`.
2. Download the relevant version for Linux/Mac/Windows.
   - We will only look at R in the Windows environment today.
3. Install it on your computer (for Windows only):
   - Choose "Yes (customized startup)" in Startup options.
   - Choose "SDI (separate windows) "in Display mode.
   - Choose "HTML help" in Help .

Using the R editor

- The R GUI is not menu driven.
- Commands can be typed at the console.
  - OK for simple calculations requiring few lines of code
  - Painful for anything more!
- We *strongly* recommend using an R editor
  - Great for reproducible analyses and research.
  - Best editor for you depends on whether you are a(n)...
    1. Beginner: Built-in R editor,
    2. Advanced user: Rstudio, Tinn-R, Notepad++, and many others.
    3. R geek: Emacs

Rstudio

- integrated development environment, or IDE, for R programming.
- Download and install it from `http://www.rstudio.com/download`.

Reasons to use it

- Writing better R code.
- Producing reports (R markdown).
- Producing interactive reports/tools (Shiny).
- Developing R packages.

Using R as a calculator

```
1 + 2
```

```
## [1] 3
```

```
1 + 3^2
```

```
## [1] 10
```

```
log(15) - sqrt(3.4)
```

```
## [1] 0.8641413
```

```
pnorm(1.96)
```

```
## [1] 0.9750021
```

Variable assignment

- <- is the "assign to" operator, made up of < and - without a space.
- E.g., `x <- 2` is read as "The value 2 is assigned to the object `x`".

```
x <- 2
y <- 3
x^2 - 3 * y + 5
```

```
## [1] 0
```

- <- has a direction, from right to left, `x <- 2` means assigning 2 to `x`,

Variable assignment

- -> operates from left to right, assigning x to 2.
  - 2 is a real value so you can not do that.

```
2 <- x
```

```
## Error in 2 <- x: invalid (do_set) left-hand side to assignm
```

- = has no direction and can be confusing sometimes.
- It is good programming practice to use <-.
- The most important thing is to keep consistent.

Getting help

- Google!!!!
  e.g. How to calculate the mean in `R`? The search results tell you that the function `mean()` would be helpful.
- Quick-R: `http://www.statmethods.net/`
- R-bloggers: `https://www.r-bloggers.com/`

Getting help

- `?`
  e.g. `?mean` brings up the help file for this function. It will tell you (almost) everything you need to know to use `mean()`.
- `??`
  e.g. `??mean` searches for everything related to mean in your computer.
- `RSiteSearch(" ")`
  Searches everything on CRAN as well as your computer.

Data, files, statisticians and R

- Statisticians prefer (read: ***want***) rectangular data files
  - Each case in its own row
  - Data collected on each variable in its own column
  - Variable names in the first row of each column
  - No blanks, e.g. fill with NA, *, 99999, anything but a blank!
- R likes (read: ***needs***) this too!
- R prefers to read data files in Comma Separated Value (CSV) format.
- This does not mean R only reads files stored in csv format.

Getting data into R

Try your best to save your data in a `csv` or `txt` format.

- Most datasets are saved in an Excel spreadsheet.
- Do as much data cleaning as you can in Excel. No comments, no formatting, no colours, no fancy fonts.
- Convert it into `csv` by clicking on `Save As`. Change the `Save as type` from `xlsx` or `xls` into `CSV (Comma Delimited)`.
- `CSV` can have one worksheet only. If you have multiple worksheets, it saves the active worksheet.

Read and Check

- Always set a working directory using `setwd()`, this can be a directory where you store the data and/or outputing the results.
- Use `read.csv` to read a CSV file into `R`.
- `dim()`: Returns the number of observations (rows) and variables (columns).
- `head()`/`tail()`: Returns the first/last few rows of a data set.
- `str()`: Returns the structure of the dataset, e.g., dimension, column names, type of data object, first few values of each variable.
- `names()`: Returns the names of the variables contained in a dataset.

Patient.df

Seven variables:

- `ID`: Identifcation Number.
- `Age`: in years
- `Sex`: 0 = Female, 1 = Male
- `Race`: 1 = Caucasian, 2 = African, 3 = Other
- `Weight`: in pounds
- `Height`: in inches
- `Smoke`: 1 = Yes, 2 = No

Reading data into R

```
setwd("your working directory")
Patient.df <- read.csv("Patient.csv")
head(Patient.df)
```

```
##   Patient.ID Age    Sex Race Weight Height Smoke
## 1          3  21   Male    1  179.5   70.4    NA
## 2          4  32 Female    1     NA   63.9    NA
## 3          9  48 Female    1  149.7   61.8     2
## 4         10  35   Male    1  203.5   69.8    NA
## 5         11  48   Male    1  155.3     NA     2
## 6         19  44   Male    2  189.6   70.2     1
```

names(Patient.df)

```
# Names of the variables
names(Patient.df)
```

```
## [1] "Patient.ID" "Age"        "Sex"
## [4] "Race"       "Weight"     "Height"
## [7] "Smoke"
```

- Anything following the # symbol is treated as a comment and ignored by R.
- Writing comments is a very good habit to develop!

dim() and str()

```
dim(Patient.df)
```

```
## [1] 17030     7
```

```
str(Patient.df)
```

```
## 'data.frame':    17030 obs. of  7 variables:
##  $ Patient.ID: int  3 4 9 10 11 19 34 44 45 48 ...
##  $ Age       : int  21 32 48 35 48 44 42 24 67 56 ...
##  $ Sex       : Factor w/ 2 levels "Female","Male": 2 1 1 2
##  $ Race      : int  1 1 1 1 1 2 2 1 2 1 ...
##  $ Weight    : num  180 NA 150 204 155 ...
##  $ Height    : num  70.4 63.9 61.8 69.8 NA 70.2 62.6 64.4 6
##  $ Smoke     : int  NA NA 2 NA 2 1 1 1 NA 2 ...
```

Note that **character** vector, Sex, is automatically converted to **factor**.

Reading data into R

```
Patient.df <- read.csv("Patient.csv",
                       stringsAsFactors = FALSE)
str(Patient.df)
```

```
## 'data.frame':    17030 obs. of  7 variables:
##  $ Patient.ID: int  3 4 9 10 11 19 34 44 45 48 ...
##  $ Age       : int  21 32 48 35 48 44 42 24 67 56 ...
##  $ Sex       : chr  "Male" "Female" "Female" "Male" ...
##  $ Race      : int  1 1 1 1 1 2 2 1 2 1 ...
##  $ Weight    : num  180 NA 150 204 155 ...
##  $ Height    : num  70.4 63.9 61.8 69.8 NA 70.2 62.6 64.4 6
##  $ Smoke     : int  NA NA 2 NA 2 1 1 1 NA 2 ...
```

stringsAsFactors

stringsAsFactors argument is set to FALSE, so **character** vectors are not converted to **factor**s.

Data Type

Everything in R is a vector (but some have only one element).

1. Numeric (same as double), or integer. E.g. `Patient.ID`, `Age`, `Race`, `Weight`, `Height` and `Smoke`
2. String (same as character). E.g. `Sex`
3. Logical: `TRUE` or `FALSE`, e.g.

```
1 == 1
```

```
## [1] TRUE
```

```
2 <= 0
```

```
## [1] FALSE
```

```
3 != 2
```

```
## [1] TRUE
```

Descriptive statistics

Calculate the mean of `Height`:

```
mean(Height)
```

```
## Error in mean(Height): object 'Height' not found
```

You must tell `R` that `Height` is a variable (column) *within* `Patient.df`, i.e.

```
mean(Patient.df$Height)
```

```
## [1] NA
```

You must also tell `R` how to deal with missing values: remove them before calculating the mean, i.e.

```
mean(Patient.df$Height, na.rm = TRUE)
```

```
## [1] 65.43787
```

table of counts

```r
# One-way table of counts

table(Patient.df$Sex)
```

```
##
## Female   Male
##   9077   7953
```

table of proportions

```
# Total count
total <- sum(table(Patient.df$Sex))
total
```

```
## [1] 17030
```

```
# Proportions of total
table(Patient.df$Sex)/total
```

```
##
##    Female      Male
## 0.5330006 0.4669994
```

One-way tables `with` less typing

Tired of typing `Patient.df$` over and over again? Use the `with` function.

```
Sex.table <- with(Patient.df, table(Sex))
Sex.table
```

```
## Sex
## Female   Male
##   9077   7953
```

```
total <- sum(Sex.table)
Sex.table/total
```

```
## Sex
##    Female      Male
## 0.5330006 0.4669994
```

One-way tables with less typing

```
# Convert to percentages
Sex.pct <- 100 * Sex.table/total
Sex.pct
```

```
## Sex
##   Female     Male
## 53.30006 46.69994
```

```
# Round to 1 decimal place
round(Sex.pct, 1)
```

```
## Sex
## Female   Male
##   53.3   46.7
```

Two-way frequency tables

```
Sex.Race.tab <- with(Patient.df, table(Sex, Race))
Sex.Race.tab
```

```
##          Race
## Sex          1    2    3
##   Female 6114 2687  274
##   Male   5498 2173  279
```

Two-way frequency tables

```
# Calculate proportion with respect to 'margin'
# total margin = 1 (row total) or 2 (column total)
perc.Sex.Race <- prop.table(Sex.Race.tab, margin = 2)
perc.Sex.Race
```

```
##          Race
## Sex             1         2         3
##   Female 0.5265243 0.5528807 0.4954792
##   Male   0.4734757 0.4471193 0.5045208
```

Two-way frequency tables

```
# Tabulate as percentages
round(100 * perc.Sex.Race, 1)
```

```
##          Race
## Sex          1    2    3
##   Female 52.7 55.3 49.5
##   Male   47.3 44.7 50.5
```

Summary

- Quick introduction to R
- Getting data into R
- Frequency tables