INTRODUCTION TO R

Introduction to Data Science DSC 105 Fall 2024

R Installation and First Steps

September 11, 2024

Contents

Figure 1: RStudio Ball Logo (Source: rstudio.com)

Figure 2: Bridge and Waterfall at Pontoise (Cezanne, 1881)

1 Overview

- Why are we using R?
- Getting in/out of R
- Installing R on Windows and Mac
- R Packages and libraries

  Inspiration and ideas especially from Davies(2016) and other places gratefully received (see references).

2 Why we are using R

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	28	-	-
Python	3	5	6	7	23	16	-	-
C++	4	3	3	3	2	2	2	8
C#	5	4	5	6	9	-	-	-
JavaScript	6	7	9	9	6	30	-	-
PHP	7	6	4	4	20	-	-	-
R	8	14	35	-	-	-	-	-
SQL	9	-	-	-	-	-	-	-
Go	10	56	15	-	-	-	-	-
Perl	14	8	7	5	4	3	-	-
Lisp	32	23	12	13	16	7	3	2
Ada	34	22	20	15	15	5	9	3

- One of the 'big three' (Python, R, SQL)
- FOSS and especially open to non-programmers
- Strong on analysis and visualization

Image Source: TIOBE.com/index - Check some of these languages out! Image is from 2021. Update 2024: R fell back to position 19.

If you don't want to leave Emacs, you can also use the `eww` browser.

When it comes to data analysis, three languages are mentioned most often: R, Python and SQL. All three have their relative merits and issues.

I chose R as the programming language for this introductory course. The choice is partly **personal** and partly **professional**. *Personal:* I like it and it's new for me (I've only taught it since early 2020), so I am still excited about it. It's good if your instructor is excited about the material! *Professional:* as business professionals, you don't want to have to be programmers. At the same time, you need to be able to speak with experts and do and extend your own analyses (not be restricted for example by dashboards).

On a *practical* note, R has a very large, diverse user and developer community. Unlike Python, many of the users do not have a technology background. This means that the "world of R" is more easily accessible if digital technologies and programming aren't your main interests. The SQL community is probably even larger and even more diverse (databases being a more general interest than even statistical analysis), but the language SQL itself is hardly extensible, very focused on querying and less on visualization.

In reality, as a data scientist, or even as a business practitioner with serious, systematic data analytics interests, you need to know all of these - R, SQL, and Python. Here, we'll start with R.

For a direct comparison of Python and R for data cleaning and exploratory analysis with examples, see e.g. Radecic (2020), Uprety (2020) and Shotwell (2020). To see how R outperforms Python, see Grogan (2020). To see some equivalents of SQL in R, check ODSC (2018). And for an overview of data science tools beyond Python, R, and SQL, see Gallatin (2018). And here's a neat infographic from datacamp comparing both for data analysis.

There are downsides to using R as well, of course, and it has been called "hard to learn", too (Muenchen 2017), partly and paradoxically because the language is so flexible and extensible. Also, some innovations, like the Tidyverse, aren't necessarily good for beginners (Matloff 2019).

Of course, there's also always an index - in this case the "TIOBE" index of programming language popularity (based on the languages people search for), see figure 2. As you can see, R improved its position in one year from 20th to 8th. That's by far the strongest improvement of any language among the top 10. Still, Python is three times more search-successful. Neither Python nor SQL have changed their position compared to one year ago. The popularity of R quite likely rides on the popularity of statistics due to the interest in COVID-19 data analysis.

3 Matloff's 10 reasons

```
1. Public domain implementation of S
2. De facto standard among professional statisticians
3. Superior to comparable commercial products
4. Available for Windows, MacOS, and Linux
5. Extensible through library packaging
6. Has OOP and functional programming features
7. Saves data and command history between sessions
8. Has a large and helpful user community
9. Allows for interactive data exploration via command-line
10. Superior graphics capabilities
```

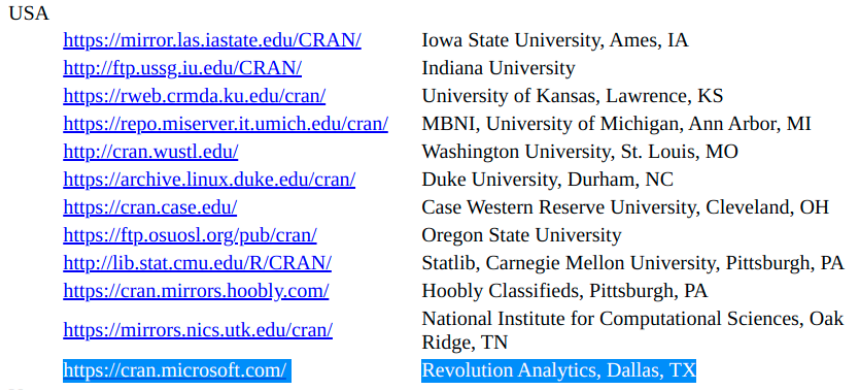
Source: The Art of R Programming (2011)
Things you should know the definition of (roughly):

- Public domain
- OOP and functional programming

- Command history
- Command-line

4 Obtaining and installing R from cran

URL: `https://cran.r-project.org/mirrors.html`



USA

https://mirror.las.iastate.edu/CRAN/	Iowa State University, Ames, IA
http://ftp.ussg.iu.edu/CRAN/	Indiana University
https://rweb.crmda.ku.edu/cran/	University of Kansas, Lawrence, KS
https://repo.miserver.it.umich.edu/cran/	MBNI, University of Michigan, Ann Arbor, MI
http://cran.wustl.edu/	Washington University, St. Louis, MO
https://archive.linux.duke.edu/cran/	Duke University, Durham, NC
https://cran.case.edu/	Case Western Reserve University, Cleveland, OH
https://ftp.osuosl.org/pub/cran/	Oregon State University
http://lib.stat.cmu.edu/R/CRAN/	Statlib, Carnegie Mellon University, Pittsburgh, PA
https://cran.mirrors.hoobly.com/	Hoobly Classifieds, Pittsburgh, PA
https://mirrors.nics.utk.edu/cran/	National Institute for Computational Sciences, Oak Ridge, TN
https://cran.microsoft.com/	Revolution Analytics, Dallas, TX

- CRAN = "Comprehensive R Archive Network" at r-project.org
- Use *mirror sites* (**what's that?**) for download (open browser)
- Practice: on the CLI, check for updates of everything:

```
sudo apt update -y && sudo apt upgrade -y
```

You can download the installer for your operating system from your local CRAN ("**Comprehensive R Archive Network**") mirror here: `https://cran.r-project.org/mirrors.html`.

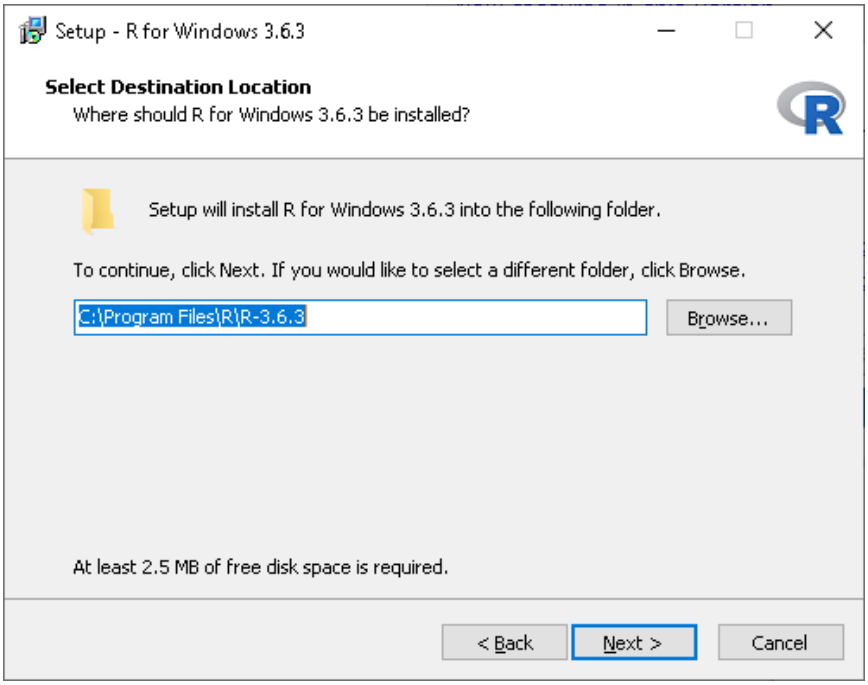
For example, if you are in Berlin, the Nürnberg server is closest: `https://ftp.fau.de/cran/`.

Challenge: Which server would you use if you were in Russia? Does the download page for that server look any different? Check it out! (Hint)

USA: notice that the TX server is at "revolutionanalytics.com", which used to be another R IDE bought by Microsoft. Microsoft embraced R so fiercely that they even started their own subset of it, Microsoft R Open, which you can get from MRAN (Microsoft R Application Network). **Can you discern the strategy here?** You can get it by reading this series of news flashes from Microsoft.

Which other open source related platforms are now Microsoft? Answer: GitHub

4.1 How this looks under windows



I tried this on Lenovo and Dell laptops running Windows 10 and it worked:

1. After opening the `R..win.exe` file, a popup asks you if you will let this pogram modify your hard disk. Say "yes" (why is this necessary?[1])

2. In the installation dialog, accept all settings and check the options for establishing a desktop shortcut and a quick launch icon.

3. The location of your R program files will be `C:\Program Files\R`. Once the installation is finished, you should have an icon on your desktop named `Rx64 4.0.2` (or whatever your version is).

4. Double click it to open the R console for the first time. At the `>` prompt, type `1+1` and `RETURN` to see if R can compute. Then type `demo(graphics)` and hit `RETURN` (`"Enter"`) repeatedly to see a few R plots.

[1] To open the R console, and direct plots to the correct device, the R program needs to be "plugged into" your operating system, as it were. You could still run it otherwise but e.g. you'd have to always type the exact program path.

5. I also switched from my integrated (default) graphics card to a "High Performance NVIDIA" graphics card (which I did not know I had!).

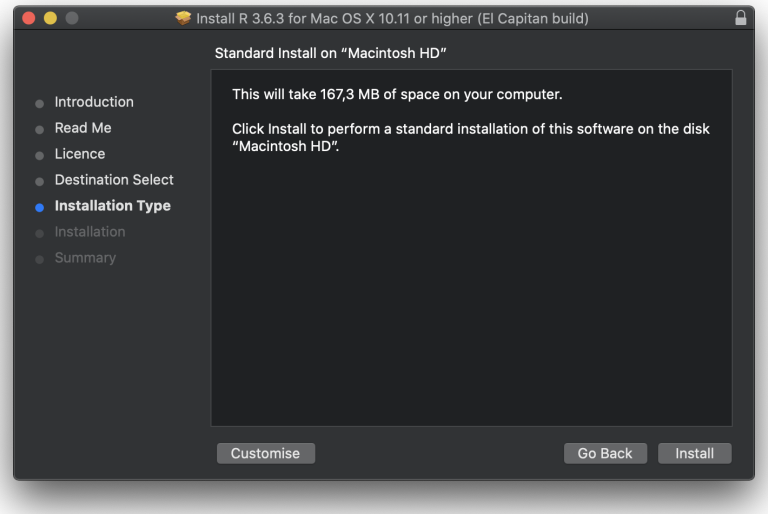
6. To leave, type `q()` at the prompt or leave with the `File > Exit` graphical menu. When asked if you wish to save the workspace, say "no".

7. When installing a program, a dialog was opened offering me to install packages in a local folder (accept this with "yes").

   See this datacamp blog post (March 11, 2020) for installation instruction for Windows, MacOS X and Ubuntu (Linux).

(If you have other troubles with R + MacOS, let me know. I have a Mac available and may be able to figure something out.)

4.2 How this looks on a mac



New installation & reconfiguration (2020)

I did this on a MacMini (2014) running MacOS 10.13.6 without too many problems (see below). Essentially the only problem occurred when trying to install packages (discussed later) and I could fix it easily by changing a system setting.

1. To download and install R for MacOS, go to `r-project.org`, and click on `CRAN` right below the `Download` headline. The CRAN mirror page opens. Scroll down to find a German mirror site and click to download the `.DMG` installer file, which will install the program.

2. There were system-level error messages though the program installed alright. But I could not install CRAN packages because of this error: `tar: Failed to set default locale`. This refers to a problem with the `tar` unzip program. I checked `stackoverflow.com` and found a fix that in turn directed me back to a CRAN helpfile with lots (too much, really) information for Mac users.

3. To fix the problem, close R, open a terminal and type: `defaults write org.R-project.R force.LANG en_US.UTF-8`. Then restart R and the problem should have disappeared (it did for me and never came back).

   See also this datacamp blog post (March 11, 2020) for installation instruction for Windows, MacOS X and Ubuntu (Linux).

(If you have other troubles with R + MacOS, (don't) let me know. I have a Mac available and may be able to figure something out.)

5 Installing R on your PC at home

- See FAQ on GitHub (birkenkrahe/org)
- Linux: `sudo apt install r-base && sudo apt install emacs`

- Windows or Mac: You need Emacs from emacs-modified.gitlab.io
- You need my `.emacs` file from tinyurl.com/lyon-emacs
- Come to my office hours for help (tinyurl.com/fall24-office-hours)

6 Practice: Find R on your machine & run R scripts

Open a terminal to execute the following shell commands. Do either:

- Open a "dumb" terminal outside of Emacs
- Open a "smart" terminal inside Emacs (`M-x shell`)
- Create an Org-mode file with `bash` code blocks (like here)

1. Check where the R executable is located:

   ```
   which R
   ```

   ```
   /usr/bin/R
   ```

   The location of applications is stored in the `$PATH` (environmental variable):

   ```
   echo $PATH
   ```

2. View the top of the file:

   ```
   cat /usr/bin/R | head -10
   ```

   ```
   #!/bin/bash
   # Shell wrapper for R executable.

   R_HOME_DIR=/usr/lib/R
   if test "${R_HOME_DIR}" = "/usr/lib/R"; then
      case "linux-gnu" in
      linux*)
   run_arch=‘uname -m‘
   case "$run_arch" in
      x86_64|mips64|ppc64|powerpc64|sparc64|s390x)
   ```

3. The R files are contained in `$R_HOME_DIR`, which is `/usr/lib/R`

4. Now look for the `Rscript` program:

```
which Rscript
```

```
/usr/bin/Rscript
```

5. Create an R test file `test.R` on the shell list and view it:

```
echo "str(mtcars)" > test.R
ls -l test.R
cat test.R
```

```
-rw-rw-r-- 1 aletheia aletheia 12 Sep 11 10:31 test.R
str(mtcars)
```

6. Run the file on the command line as a script:

```
Rscript test.R
```

```
'data.frame': 32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

7. Run the file as a batch job (in the background):

```
R CMD BATCH test.R
```

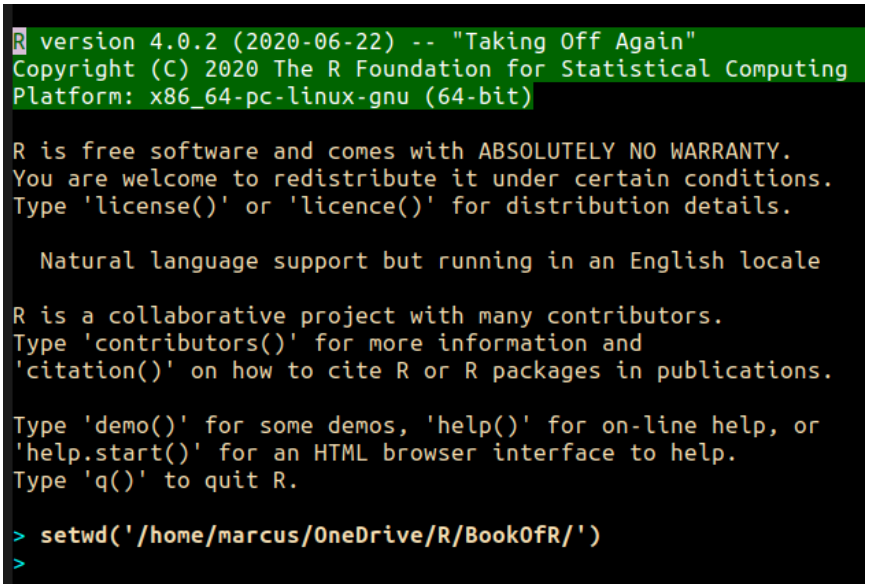
8. The results are stored in a file: `testR.out`:

```
cat test.Rout
```

```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

*** Loaded .Rprofile ***
[Previously saved workspace restored]

> str(mtcars)
'data.frame': 32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
>
> proc.time()
   user  system elapsed
  0.058   0.014   0.068
```

7 NEXT R shell: Version and platform



- What type of bit-architecture do you have?

  ```
  uname -m
  ```

  ```
  x86_64
  ```

- The `uname` command prints system information. In Emacs, run `M-x man RET uname` to access the manual page.
- To find out more about your system, enter

  ```
  cat /etc/os-release
  ```

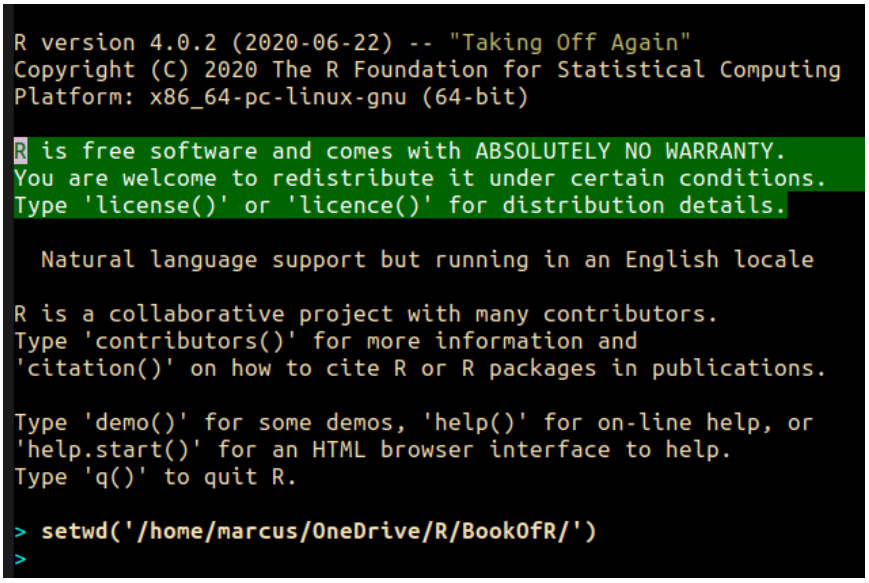
- You can also look at CPU information:

```
cat /proc/cpuinfo
```

This is the first screen you see (figure 7) after starting R on the command-line. The highlighted section shows the current (June 2020) version of Base-R, as the core R program is officially called. Versions get their own names, like operating systems (my Ubuntu Linux operating system e.g. has the version number `18.04-LTS` and the name "Bionic Beaver"). `R 4.0.2` is also called "Taking Off Again". Lastly, the platform of the operating system on which the R program runs, is shown - a 64-bit version of Linux using the x86 computer architecture.

Challenge: what type of computer architecture does your computer have (most importantly: 64-bit)? (Linux: `cat /etc/cpuinfo`)

8 R shell: Distribution license



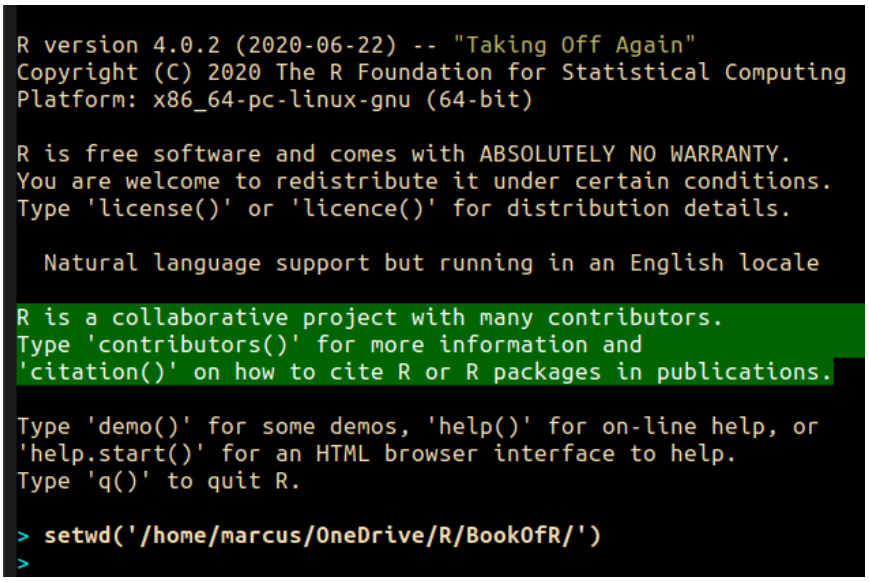
- Open the R console, either in a dumb or in a smart terminal (`M-x shell`), or in a dedicated R buffer (`*R*`), or in Emacs (`M-x R`).
- Type `license()`. What is "GNU"?

As you'll find out when following the instructions in figure 8 by entering `license()` at the prompt, the R software is distributed "under the terms of the GNU General Public License" (GPL). Popular software also distributed

under the GPL include the Linux "kernel" (the core of the operating system), and the GNU compiler collection. You may have heard of the term "open source", which essentially means the same thing, though one may quibble (and people do, a lot). What's important to remember: use of the GPL (= making R "free software") has contributed enormously to the success of this language.

Challenge: what is "GNU software" exactly? Which programs belong to it? Are there any programs that you have used before? (Hint)

9 R shell: The R project



On the R console:

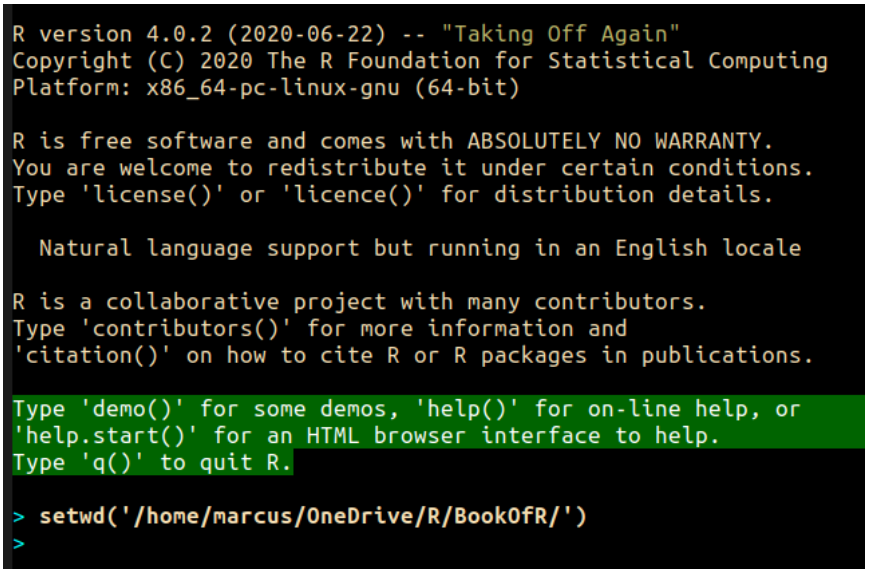
- Enter `citation()`. Why cite software?
- Enter `contributors()`. Who can contribute?

  Behind R is a large project of volunteers (figure 9. At it centre is the "R Core Group" of developers. Because R is part of the "GNU suite" of programs, and because its predecessor was called S, it is also sometimes called "GNU S". Becker (2004) has written an interesting historical account of S. When using R for analysis in a thesis, a paper, an essay or a blog post, one should cite it as a source. This is what the code `citation()` is for. Same goes for specific packages (more

on this later) like "data.table" that are not part of Base-R. The citation alternatives may also prompt you to check out LaTeX and BibTeX, which are quasi-standards for the professional (and beautiful!) formatting of scientific papers.

Challenge: is there any connection between R and LaTeX? Or more general between the programming language R und markup languages (like HTML or LaTeX)? (Hint)

10 R shell: Demo and help



Do this on an R console in a dumb or smart shell:

1. Enter `demo(graphics)` for some graphics examples.
2. Enter `help.start()` - where is this page?

```
help.start()


If the browser launched by 'xdg-open' is already running, it is *not* restarted,
    and you must switch to its window.
Otherwise, be patient ...
```

3. Calling `help` or `?` on Linux opens the manual page for the item (don't do this in a code block but in the R console):

```
?Nile
help(mtcars)
```

The section higlighted in figure 10 suggests a few commands that you ought to try for yourself:

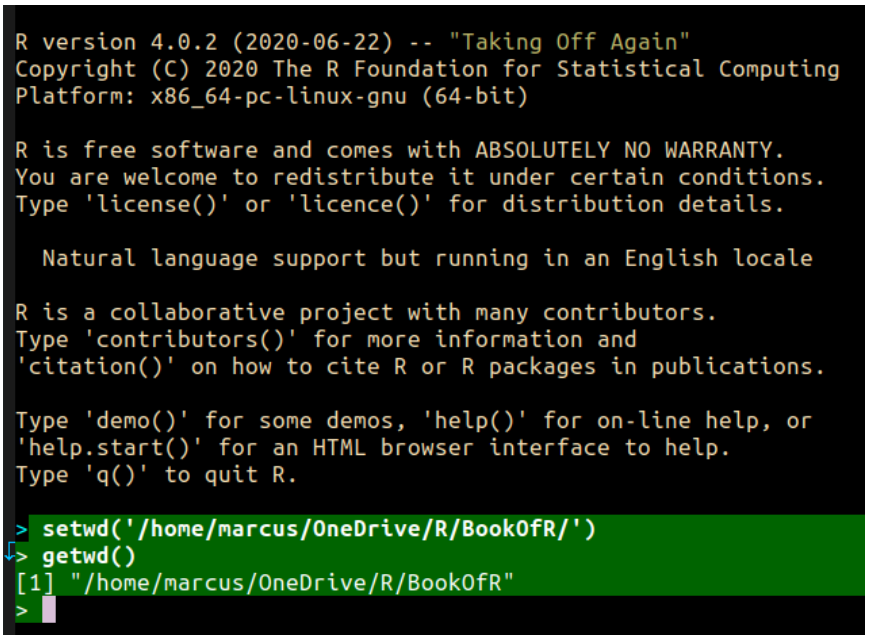
`help()` is a function to get help for whatever you put in between the brackets. A quick win is `help(help)`, or help about the help function. The format of the help pages is borrowed from the Unix man[ual] pages. An alternative to `help()` is `?` followed by the term you need help with, e.g. `?help`, which is the same as `help(help)` but much shorter. Lastly, `help.start()` opens a browser window with help in HTML format. Very useful access to a wealth of systematic information. If you don't know the exact name, you can also search across all documentation using `help.search()` or the shortcut `??`. Try entering `??cars` if you are looking for datasets on cars. You'll find that there are four known datasets with cars in different packages.

Via the dataset search, you can also find out that functions like `help()` or `demo()` are part of the `utils` package - respective functions are listed as `utils::[function]`. It contains all sorts of functions for housekeeping and administration.

The R help system is however not written for beginners. Personally, I more often go to textbooks or, preferably, to stackoverflow.com if I have a question or need to remind myself of a command or a way of doing things.

There are a few interactive demo programs available, too. You should try `demo(graphics)` and marvel at the various possibilities of R to create plots with your data. Notice how few lines of code are sufficient to create great effects! The window that opens when you execute the demo commands is the standard graphics output when using R in command-line mode.

11 R environment: working directory



In the R console:

1. Enter `getwd()` ("get working dir")

   ```
   getwd()
   ```

   ```
   [1] "/home/aletheia/GitHub/ds1"
   ```

2. Use `setwd()` to change directory to your user home directory (`$HOME`):

   - Using a relative path address: from the current location (`.`)

     ```
     setwd("../../../")
     getwd()
     ```

   - Using an absolute path address: from the root directory (`/`)

     ```
     setwd("/home/aletheia")
     getwd()
     ```

3. Use `system` to run `bash` shell commands from inside R:

```
system("pwd")  # present working directory
```

```
/home/aletheia/GitHub/ds1
```

A file listing command:

```
system("ls") # list files
```

```
data  img  ipynb  LICENSE  org pdf  README.md
```

A shell pipe with a file listing and a counting command combined:

```
system("ls -la | wc -l") # count number of lines in file listing
```

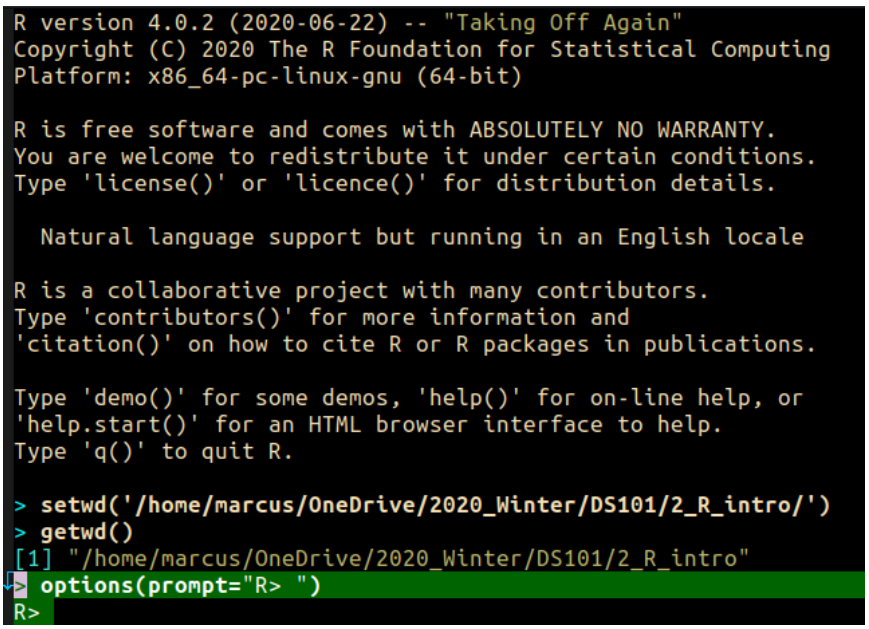
```
13
```

When you start R, you may be asked, which working directory you wish to use. This is where all files created (e.g. plots) will be put and where R will look first to load scripts with R commands for execution.

The setwd() command in figure 11 allows you to set any directory as working directory. To check which one is used right now, you can use getwd().

How you specify the path to the current working directory depends on your operating system, e.g. `/home/marcus` for my home directory on MacOS/Linux, or `C:\Users\Marcus` under Windows. Especially as a Windows user, you should look at your file organisation - this will pay off as soon as you use the terminal or command-line. The Bash shell that I use on my Linux computer (and that most MacOS users will use) is also available within Windows 10 (Posey 2018).

12 R display `options`



1. The function `options` controls all glocal options for R:

   ```
   help(options)
   ```

2. `options` is a list:

   ```
   class(options())
   ```

   ```
   [1] "list"
   ```

3. You can look at it:

   ```
   options()
   ```

   ```
   $add.smooth
   [1] TRUE

   $bitmapType
   ```

```
[1] "cairo"

$browser
[1] "xdg-open"

$browserNLdisabled
[1] FALSE

$CBoundsCheck
[1] FALSE

$check.bounds
[1] FALSE

$citation.bibtex.max
[1] 1

$continue
[1] "+ "

$contrasts
   unordered            ordered
"contr.treatment"      "contr.poly"

$crayon.enabled
[1] FALSE

$datatable.alloccol
[1] 1024

$datatable.allow.cartesian
[1] FALSE

$datatable.auto.index
[1] TRUE

$datatable.dfdispatchwarn
[1] TRUE

$datatable.optimize
```

```
[1] Inf

$datatable.print.class
[1] TRUE

$datatable.print.colnames
[1] "auto"

$datatable.print.keys
[1] TRUE

$datatable.print.nrows
[1] 100

$datatable.print.rownames
[1] TRUE

$datatable.print.topn
[1] 5

$datatable.print.trunc.cols
[1] FALSE

$datatable.show.indices
[1] FALSE

$datatable.use.index
[1] TRUE

$datatable.verbose
[1] FALSE

$datatable.warnredundantby
[1] TRUE

$defaultPackages
[1] "datasets"  "utils"     "grDevices" "graphics"  "stats"     "methods"

$demo.ask
[1] "default"
```

```
$deparse.cutoff
[1] 60

$device
function (display = "", width, height, pointsize, gamma, bg,
    canvas, fonts, family, xpos, ypos, title, type, antialias,
    symbolfamily)
{
    if (display != "XImage") {
   check <- Sys.getenv("_R_CHECK_SCREEN_DEVICE_", "")
   msg <- "screen devices should not be used in examples etc"
   if (identical(check, "stop"))
       stop(msg, domain = NA)
   else if (identical(check, "warn"))
       warning(msg, immediate. = TRUE, noBreaks. = TRUE,
   domain = NA)
    }
    if (display == "" && .Platform$GUI == "AQUA" && is.na(Sys.getenv("DISPLAY",
   NA)))
   Sys.setenv(DISPLAY = ":0")
    new <- list()
    if (!missing(display))
   new$display <- display
    if (!missing(width))
   new$width <- width
    if (!missing(height))
   new$height <- height
    if (!missing(gamma))
   new$gamma <- gamma
    if (!missing(pointsize))
   new$pointsize <- pointsize
    if (!missing(bg))
   new$bg <- bg
    if (!missing(canvas))
   new$canvas <- canvas
    if (!missing(xpos))
   new$xpos <- xpos
    if (!missing(ypos))
   new$ypos <- ypos
```

```
    if (!missing(title))
    new$title <- title
    if (!checkIntFormat(new$title))
    stop("invalid 'title'")
    if (!missing(type)) {
    new$type <- match.arg(type, c("Xlib", "cairo", "nbcairo",
        "dbcairo"))
    if (!capabilities("cairo") && type != "Xlib")
        warning("cairo-based types are not supported on this build - using \"Xlib\"
    }
    if (!missing(family))
    new$family <- family
    if (!missing(fonts))
    new$fonts <- fonts
    if (!missing(antialias) && type != "Xlib")
    new$antialias <- match.arg(antialias, aa.cairo)
    if (!missing(symbolfamily))
    new$symbolfamily <- symbolfamily
    d <- check.options(new, name.opt = ".X11.Options", envir = .X11env)
    if (d$type == "Xlib" && !missing(family)) {
    fns <- X11Fonts()
    if (!family %in% names(fns))
        stop("unknown family for X11(type = \"XLib\")")
    d$fonts[1] <- fns[[family]]
    }
    type <- if (capabilities("cairo"))
    switch(d$type, cairo = 1L, nbcairo = 2L, dbcairo = 3L,
        0L)
    else 0L
    if (display == "XImage")
    type <- 0L
    antialias <- match(d$antialias, aa.cairo)
    if (grepl("darwin", R.version$os))
    check_for_XQuartz()
    .External2(C_X11, d$display, d$width, d$height, d$pointsize,
    d$gamma, d$colortype, d$maxcubesize, d$bg, d$canvas,
    d$fonts, NA_integer_, d$xpos, d$ypos, d$title, type,
    antialias, d$family, optionSymbolFont(d$symbolfamily))
    invisible()
}
```

```
<bytecode: 0x5de5a0e20ba0>
<environment: namespace:grDevices>

$device.ask.default
[1] FALSE

$digits
[1] 7

$dvipscmd
[1] "dvips"

$echo
[1] TRUE

$editor
[1] "emacsclient"

$encoding
[1] "native.enc"

$example.ask
[1] "default"

$expressions
[1] 5000

$help.search.types
[1] "vignette" "demo"     "help"

$help.try.all.packages
[1] FALSE

$HTTPUserAgent
[1] "R (4.1.2 x86_64-pc-linux-gnu x86_64 linux-gnu)"

$internet.info
[1] 2

$keep.parse.data
```

```
[1] TRUE

$keep.parse.data.pkgs
[1] FALSE

$keep.source
[1] TRUE

$keep.source.pkgs
[1] FALSE

$length
[1] 99999

$locatorBell
[1] TRUE

$mailer
[1] "mailto"

$matprod
[1] "default"

$max.print
[1] 99999

$menu.graphics
[1] TRUE

$na.action
[1] "na.omit"

$nwarnings
[1] 50

$OutDec
[1] "."

$pager
[1] "cat"
```

```
$papersize
[1] "a4"

$PCRE_limit_recursion
[1] NA

$PCRE_study
[1] FALSE

$PCRE_use_JIT
[1] TRUE

$pdfviewer
[1] "/usr/bin/xdg-open"

$pkgType
[1] "source"

$printcmd
[1] "/usr/bin/lpr"

$prompt
[1] "R> "

$repos
[1] "https://mirrors.nics.utk.edu/cran/"

$rl_word_breaks
[1] " \t\n\"\\'`><=%;,|&{()}"

$scipen
[1] 0

$show.coef.Pvalues
[1] TRUE

$show.error.locations
[1] TRUE
```

```
$show.error.messages
[1] TRUE

$show.signif.stars
[1] TRUE

$STERM
[1] "iESS"

$str
$str$strict.width
[1] "no"

$str$digits.d
[1] 3

$str$vec.len
[1] 4

$str$list.len
[1] 99

$str$deparse.lines
NULL

$str$drop.deparse.attr
[1] TRUE

$str$formatNum
function (x, ...)
format(x, trim = TRUE, drop0trailing = TRUE, ...)
<environment: 0x5de5a182cc18>


$str.dendrogram.last
[1] "'"

$stringsAsFactors
[1] FALSE
```

```
$texi2dvi
[1] "/usr/bin/texi2dvi"

$timeout
[1] 60

$ts.eps
[1] 1e-05

$ts.S.compat
[1] FALSE

$unzip
[1] "/usr/bin/unzip"

$useFancyQuotes
[1] TRUE

$verbose
[1] FALSE

$warn
[1] 0

$warning.length
[1] 1000

$width
[1] 94
```

4. You can extract display options with `$`, e.g. for the R console prompt:

```
options()$prompt
```

```
[1] "R> "
```

5. Another important option setting is for the repository that R uses to download packages: Set to the default CRAN repo in my `.Rprofile`

```
options()$repos
```

```
[1] "https://mirrors.nics.utk.edu/cran/"
```

6. Change the shell prompt to `R>`:

```
options(prompt="R> ")
```

7. The change affects only your current R session. Change to the `*R*` console buffer to check this:

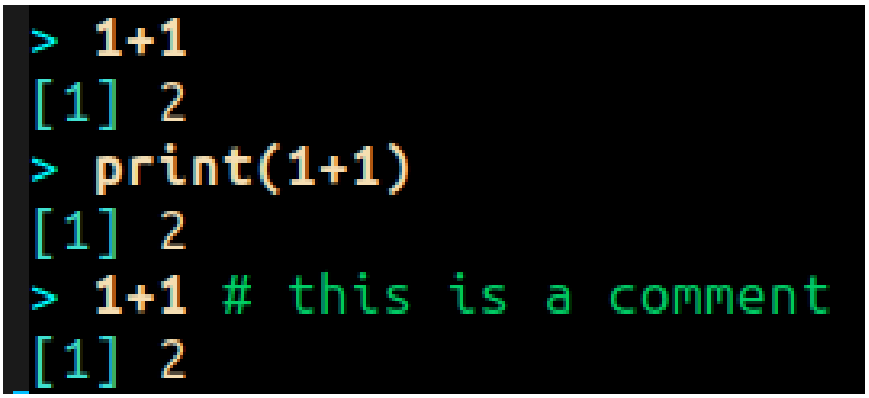
```
> setwd('/home/aletheia/GitHub/ds1/org/')
> options(prompt="R> ")
'org_babel_R_eoe'
R> [1] "org_babel_R_eoe"
R>
```

8. On the R console, change the prompt back to what it was.

```
R>
R> options(prompt="> ")
>
> options()$prompt
[1] "> "
>
```

Figure 12 shows a new utility command, `options()`, that you can use to change the identifying prompt at the beginning of the command line. You don't have to do this but it's nice to know that and how you can do it. One of the advantages of working on the command-line is that you experience how you can adapt your working environment to your personal needs - something that most graphical environments do not allow you do to (at least not without a lot more effort). Freedom of extensibility is the name of the command-line game.

13 R computing and commenting



1. In the R console compute 2 + 2 (code block, `*R*` buffer or terminal)
2. Pass the operation `2+2` as an argument to the `print` function
3. Run both operations again but with an inline comment
4. Put the code into an R script `print.R` (C-x C-f)
5. Open a shell with `M-x shell` and run the script there.
6. Run the script again but as a background "batch" job.

One of the advantages of the interactive command-line is the ability to perform arithmetic operations. In figure 13 we begin with a simple addition. We'll do a lot more of this in the next section. When you type the command and click `ENTER`, R responds by printing out the result without the need to explicit instruct it using a `print` command (though as you can see, this works as well). You also see here that `#` is the R sign for a comment (which is ignored upon execution). The ominous `[1]` at the beginning of each output line indicates the number of columns printed. R does this because it is strongest when manipulating tabular data - data ordered in columns and rows.

14 R packages

- Packages contain functions and data sets
- Most packages must be installed and loaded first

- Default data sets are pre-loaded: `?datasets`

15 Install R packages

```
> install.packages("MASS")
Installing package into '/home/marcus/R/x86_64-pc-linux-gnu-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://ftp.fau.de/cran/src/contrib/MASS_7.3-54.tar.gz'
Content type 'application/x-gzip' length 506246 bytes (494 KB)
==================================================
downloaded 494 KB
```

- To install package "`MASS`": enter `install.packages("MASS")`
- Installation includes identifying location on your computer: you may have to do it on the R console and confirm creation of a local repo
- Installation downloads compressed *tarball* from a CRAN mirror site
- `md5sum` is a GNU utility program that checks correct file transfer
- Package version and R version may be out of synch

16 Installing older versions of packages for older version of R

For example for the `MASS` package: check your R `version` and then pick an earlier package version using the CRAN archive.

For example, if you have R version 4.0.4 (2021-02-15), then version 7.3.54 from 2021-05-03 is a safe bet:

```
install.packages("remotes")
require(remotes)
install_version("MASS", version="7.3.54")
library(MASS)
search()  # MASS appears in environment list
```

17 Miscellaneous package commands

- For a list of currently loaded packages: `search()`
- To load a package into current R session only: `library("...")`

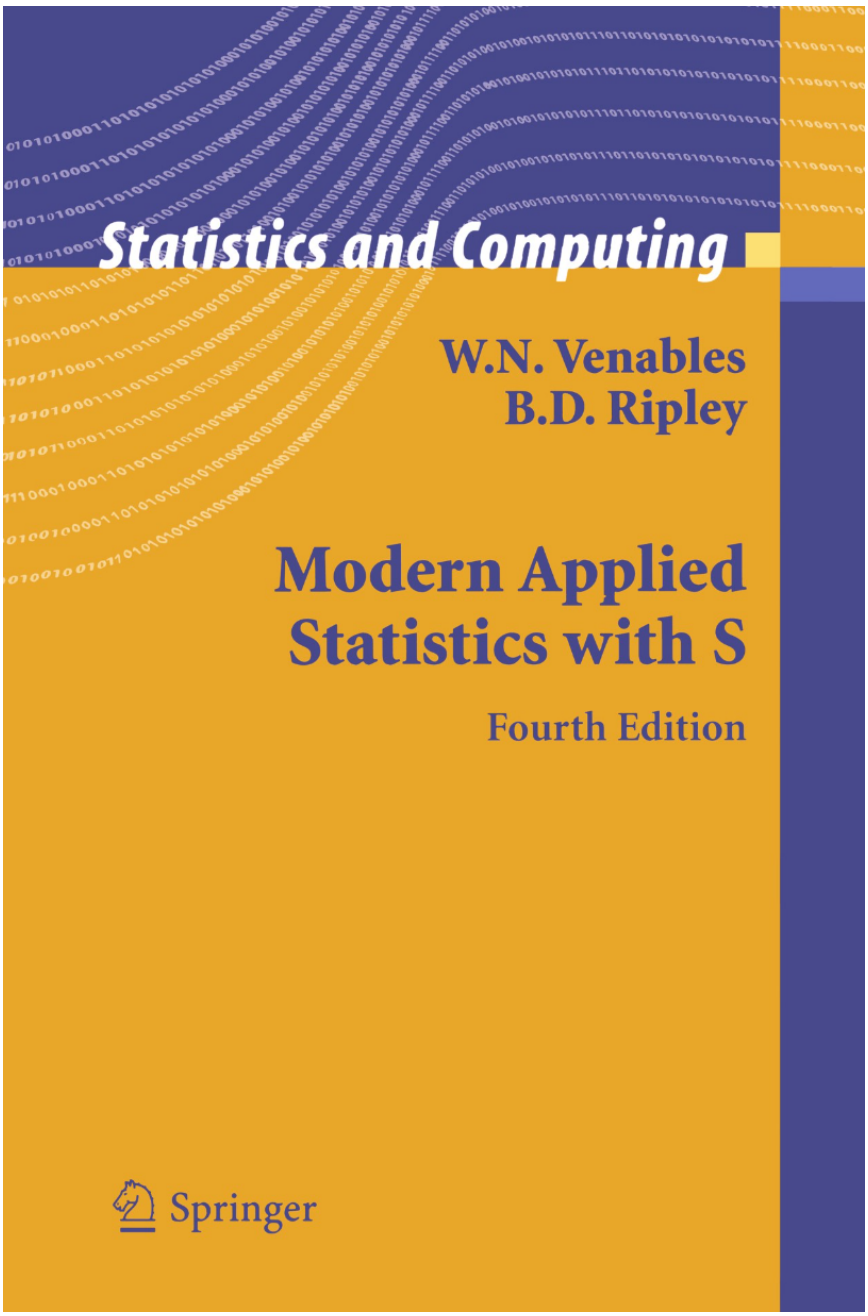


Figure 3: MASS is from the book by Venables/Ripley (2002)

- `data()` will list all datasets for all installed packages
- To uninstall a package, use `remove.packages("[pkgname]")`: for example, install `dyplr`, load it, and then remove it again.
- Close your R console (where `dplyr` is still loaded), open a new one, and try to load it.
- To see all installed packages: `installed.packages()`
- That's a lot of packages. To look only at the top/bottom of the list, pipe (`|>`) the command into `head()` and `tail()`:

```
installed.packages() |> head()  # top of the list
installed.packages() |> tail()  # bottom of the list
```

- To update packages: `update.packages()` (this can take a while and you'll have to confirm updates - run it and `cancel`.)
- For a short package description: `packageDescription("...")`. Get the description for the `base` package:

```
packageDescription("base")
```

```
Package: base
Version: 4.1.2
Priority: base
Title: The R Base Package
Author: R Core Team and contributors worldwide
Maintainer: R Core Team <do-use-Contact-address@r-project.org>
Contact: R-help mailing list <r-help@r-project.org>
Description: Base R functions.
License: Part of R 4.1.2
Suggests: methods
Built: R 4.1.2; ; 2022-02-09 05:09:20 UTC; unix

-- File: /usr/lib/R/library/base/Meta/package.rds
```

- To see all datasets in a package: `data(package="...")`. List all datasets in the base R datasets collection `datasets`:

```
data(package="datasets")


Data sets in package 'datasets':

AirPassengers               Monthly Airline Passenger Numbers 1949-1960
BJsales                     Sales Data with Leading Indicator
BJsales.lead (BJsales)      Sales Data with Leading Indicator
BOD                         Biochemical Oxygen Demand
CO2                         Carbon Dioxide Uptake in Grass Plants
ChickWeight                 Weight versus age of chicks on different diets
DNase                       Elisa assay of DNase
EuStockMarkets              Daily Closing Prices of Major European Stock Indices,
      1991-1998
Formaldehyde                Determination of Formaldehyde
HairEyeColor                Hair and Eye Color of Statistics Students
Harman23.cor                Harman Example 2.3
Harman74.cor                Harman Example 7.4
Indometh                    Pharmacokinetics of Indomethacin
InsectSprays                Effectiveness of Insect Sprays
JohnsonJohnson              Quarterly Earnings per Johnson & Johnson Share
LakeHuron                   Level of Lake Huron 1875-1972
LifeCycleSavings            Intercountry Life-Cycle Savings Data
Loblolly                    Growth of Loblolly pine trees
Nile                        Flow of the River Nile
Orange                      Growth of Orange Trees
OrchardSprays               Potency of Orchard Sprays
PlantGrowth                 Results from an Experiment on Plant Growth
Puromycin                   Reaction Velocity of an Enzymatic Reaction
Seatbelts                   Road Casualties in Great Britain 1969-84
Theoph                      Pharmacokinetics of Theophylline
Titanic                     Survival of passengers on the Titanic
ToothGrowth                 The Effect of Vitamin C on Tooth Growth in Guinea Pigs
UCBAdmissions               Student Admissions at UC Berkeley
UKDriverDeaths              Road Casualties in Great Britain 1969-84
UKgas                       UK Quarterly Gas Consumption
USAccDeaths                 Accidental Deaths in the US 1973-1978
USArrests                   Violent Crime Rates by US State
USJudgeRatings              Lawyers' Ratings of State Judges in the US Superior
      Court
```

USPersonalExpenditure	Personal Expenditure Data
UScitiesD	Distances Between European Cities and Between US Citie
VADeaths	Death Rates in Virginia (1940)
WWWusage	Internet Usage per Minute
WorldPhones	The World's Telephones
ability.cov	Ability and Intelligence Tests
airmiles	Passenger Miles on Commercial US Airlines, 1937-1960
airquality	New York Air Quality Measurements
anscombe	Anscombe's Quartet of 'Identical' Simple Linear Regressions
attenu	The Joyner-Boore Attenuation Data
attitude	The Chatterjee-Price Attitude Data
austres	Quarterly Time Series of the Number of Australian Residents
beaver1 (beavers)	Body Temperature Series of Two Beavers
beaver2 (beavers)	Body Temperature Series of Two Beavers
cars	Speed and Stopping Distances of Cars
chickwts	Chicken Weights by Feed Type
co2	Mauna Loa Atmospheric CO2 Concentration
crimtab	Student's 3000 Criminals Data
discoveries	Yearly Numbers of Important Discoveries
esoph	Smoking, Alcohol and (O)esophageal Cancer
euro	Conversion Rates of Euro Currencies
euro.cross (euro)	Conversion Rates of Euro Currencies
eurodist	Distances Between European Cities and Between US Citie
faithful	Old Faithful Geyser Data
fdeaths (UKLungDeaths)	Monthly Deaths from Lung Diseases in the UK
freeny	Freeny's Revenue Data
freeny.x (freeny)	Freeny's Revenue Data
freeny.y (freeny)	Freeny's Revenue Data
infert	Infertility after Spontaneous and Induced Abortion
iris	Edgar Anderson's Iris Data
iris3	Edgar Anderson's Iris Data
islands	Areas of the World's Major Landmasses
ldeaths (UKLungDeaths)	Monthly Deaths from Lung Diseases in the UK
lh	Luteinizing Hormone in Blood Samples
longley	Longley's Economic Regression Data
lynx	Annual Canadian Lynx trappings 1821-1934
mdeaths (UKLungDeaths)	Monthly Deaths from Lung Diseases in the UK
morley	Michelson Speed of Light Data

```
mtcars                      Motor Trend Car Road Tests
nhtemp                      Average Yearly Temperatures in New Haven
nottem                      Average Monthly Temperatures at Nottingham, 1920-1939
npk                         Classical N, P, K Factorial Experiment
occupationalStatus          Occupational Status of Fathers and their Sons
precip                      Annual Precipitation in US Cities
presidents                  Quarterly Approval Ratings of US Presidents
pressure                    Vapor Pressure of Mercury as a Function of Temperature
quakes                      Locations of Earthquakes off Fiji
randu                       Random Numbers from Congruential Generator RANDU
rivers                      Lengths of Major North American Rivers
rock                        Measurements on Petroleum Rock Samples
sleep                       Student's Sleep Data
stack.loss (stackloss)      Brownlee's Stack Loss Plant Data
stack.x (stackloss)         Brownlee's Stack Loss Plant Data
stackloss                   Brownlee's Stack Loss Plant Data
state.abb (state)           US State Facts and Figures
state.area (state)          US State Facts and Figures
state.center (state)        US State Facts and Figures
state.division (state)      US State Facts and Figures
state.name (state)          US State Facts and Figures
state.region (state)        US State Facts and Figures
state.x77 (state)           US State Facts and Figures
sunspot.month               Monthly Sunspot Data, from 1749 to "Present"
sunspot.year                Yearly Sunspot Data, 1700-1988
sunspots                    Monthly Sunspot Numbers, 1749-1983
swiss                       Swiss Fertility and Socioeconomic Indicators (1888)
      Data
treering                    Yearly Treering Data, -6000-1979
trees                       Diameter, Height and Volume for Black Cherry Trees
uspop                       Populations Recorded by the US Census
volcano                     Topographic Information on Auckland's Maunga Whau
      Volcano
warpbreaks                  The Number of Breaks in Yarn during Weaving
women                       Average Heights and Weights for American Women
```

- For a list of search paths (to find pkgs): `searchpaths()`

```
searchpaths()
```

```
 [1] ".GlobalEnv"
 [2] "/home/aletheia/R/x86_64-pc-linux-gnu-library/4.1/data.table"
 [3] "ESSR"
 [4] "/usr/lib/R/library/stats"
 [5] "/usr/lib/R/library/graphics"
 [6] "/usr/lib/R/library/grDevices"
 [7] "/usr/lib/R/library/utils"
 [8] "/usr/lib/R/library/datasets"
 [9] "/usr/lib/R/library/methods"
[10] "Autoloads"
[11] "/usr/lib/R/library/base"
```

- To list functions in a package, use `lsf.str` for lots of detail, or `ls` for an overview - you must load the package first:

```
library(MASS)
ls("package:MASS")
lsf.str("package:MASS")
```

```
[1] "abbey"             "accdeaths"         "addterm"           "Aids2"
  [5] "Animals"           "anorexia"          "area"              "as.fractions"
  [9] "bacteria"          "bandwidth.nrd"     "bcv"               "beav1"
 [13] "beav2"             "biopsy"            "birthwt"           "Boston"
 [17] "boxcox"            "cabbages"          "caith"             "Cars93"
 [21] "cats"              "cement"            "chem"              "con2tr"
 [25] "contr.sdif"        "coop"              "corresp"           "cov.mcd"
 [29] "cov.mve"           "cov.rob"           "cov.trob"          "cpus"
 [33] "crabs"             "Cushings"          "DDT"               "deaths"
 [37] "denumerate"        "dose.p"            "drivers"           "dropterm"
 [41] "eagles"            "enlist"            "epil"              "eqscplot"
 [45] "farms"             "fbeta"             "fgl"               "fitdistr"
 [49] "forbes"            "fractions"         "frequency.polygon" "GAGurine"
 [53] "galaxies"          "gamma.dispersion"  "gamma.shape"       "gehan"
 [57] "genotype"          "geyser"            "gilgais"           "ginv"
 [61] "glm.convert"       "glm.nb"            "glmmPQL"           "hills"
 [65] "hist.FD"           "hist.scott"        "housing"           "huber"
 [69] "hubers"            "immer"             "Insurance"         "is.fractions"
 [73] "isoMDS"            "kde2d"             "lda"               "ldahist"
 [77] "leuk"              "lm.gls"            "lm.ridge"          "lmsreg"
```

```
 [81] "lmwork"           "loglm"            "loglm1"           "logtrans"
 [85] "lqs"              "lqs.formula"      "ltsreg"           "mammals"
 [89] "mca"              "mcycle"           "Melanoma"         "menarche"
 [93] "michelson"        "minn38"           "motors"           "muscle"
 [97] "mvrnorm"          "nclass.freq"      "neg.bin"          "negative.binomi
[101] "negexp.SSival"    "newcomb"          "nlschools"        "npk"
[105] "npr1"             "Null"             "oats"             "OME"
[109] "painters"         "parcoord"         "petrol"           "phones"
[113] "Pima.te"          "Pima.tr"          "Pima.tr2"         "polr"
[117] "psi.bisquare"     "psi.hampel"       "psi.huber"        "qda"
[121] "quine"            "Rabbit"           "rational"         "renumerate"
[125] "rlm"              "rms.curv"         "rnegbin"          "road"
[129] "rotifer"          "Rubber"           "sammon"           "select"
[133] "Shepard"          "ships"            "shoes"            "shrimp"
[137] "shuttle"          "Sitka"            "Sitka89"          "Skye"
[141] "snails"           "SP500"            "stdres"           "steam"
[145] "stepAIC"          "stormer"          "studres"          "survey"
[149] "synth.te"         "synth.tr"         "theta.md"         "theta.ml"
[153] "theta.mm"         "topo"             "Traffic"          "truehist"
[157] "ucv"              "UScereal"         "UScrime"          "VA"
[161] "waders"           "whiteside"        "width.SJ"         "write.matrix"
[165] "wtloss"
addterm : function (object, ...)
area : function (f, a, b, ..., fa = f(a, ...), fb = f(b, ...), limit = 10, eps = 1
as.fractions : function (x)
bandwidth.nrd : function (x)
bcv : function (x, nb = 1000, lower = 0.1 * hmax, upper = hmax)
boxcox : function (object, ...)
con2tr : function (obj)
contr.sdif : function (n, contrasts = TRUE, sparse = FALSE)
corresp : function (x, ...)
cov.mcd : function (...)
cov.mve : function (...)
cov.rob : function (x, cor = FALSE, quantile.used = floor((n + p + 1)/2), method =
    "classical"), nsamp = "best", seed)
cov.trob : function (x, wt = rep(1, n), cor = FALSE, center = TRUE, nu = 5, maxit
denumerate : function (x)
dose.p : function (obj, cf = 1:2, p = 0.5)
dropterm : function (object, ...)
enlist : function (vec)
```

```
eqscplot : function (x, y, ratio = 1, tol = 0.04, uin, ...)
fbeta : function (x, alpha, beta)
fitdistr : function (x, densfun, start, ...)
fractions : function (x, cycles = 10, max.denominator = 2000, ...)
frequency.polygon : function (x, nclass = nclass.freq(x), xlab = "", ylab = "", .
gamma.dispersion : function (object, ...)
gamma.shape : function (object, ...)
ginv : function (X, tol = sqrt(.Machine$double.eps))
glm.convert : function (object)
glm.nb : function (formula, data, weights, subset, na.action, start = NULL, etasta
    control = glm.control(...), method = "glm.fit", model = TRUE, x = FALSE, y = T
    contrasts = NULL, ..., init.theta, link = log)
glmmPQL : function (fixed, random, family, data, correlation, weights, control, ni
    ...)
hist.FD : function (x, prob = TRUE, xlab = deparse(substitute(x)), ...)
hist.scott : function (x, prob = TRUE, xlab = deparse(substitute(x)), ...)
huber : function (y, k = 1.5, tol = 1e-06)
hubers : function (y, k = 1.5, mu, s, initmu = median(y), tol = 1e-06)
is.fractions : function (f)
isoMDS : function (d, y = cmdscale(d, k), k = 2, maxit = 50, trace = TRUE, tol = (
kde2d : function (x, y, h, n = 25, lims = c(range(x), range(y)))
lda : function (x, ...)
ldahist : function (data, g, nbins = 25, h, x0 = -h/1000, breaks, xlim = range(bre
    width, type = c("histogram", "density", "both"), sep = (type != "density"), co
    xlab = deparse(substitute(data)), bty = "n", ...)
lm.gls : function (formula, data, W, subset, na.action, inverse = FALSE, method =
    x = FALSE, y = FALSE, contrasts = NULL, ...)
lm.ridge : function (formula, data, subset, na.action, lambda = 0, model = FALSE,
    contrasts = NULL, ...)
lmsreg : function (...)
lmwork : function (object)
loglm : function (formula, data, subset, na.action, ...)
loglm1 : function (formula, data, ...)
logtrans : function (object, ...)
lqs : function (x, ...)
lqs.formula : function (formula, data, ..., method = c("lts", "lqs", "lms", "S", "
    na.action, model = TRUE, x.ret = FALSE, y.ret = FALSE, contrasts = NULL)
ltsreg : function (...)
mca : function (df, nf = 2, abbrev = FALSE)
mvrnorm : function (n = 1, mu, Sigma, tol = 1e-06, empirical = FALSE, EISPACK = FA
```

```
nclass.freq : function (x)
neg.bin : function (theta = stop("'theta' must be given"))
negative.binomial : function (theta = stop("'theta' must be specified"), link = "l
negexp.SSival : function (mCall, data, LHS)
Null : function (M)
parcoord : function (x, col = 1, lty = 1, var.label = FALSE, ...)
polr : function (formula, data, weights, start, ..., subset, na.action, contrasts
    model = TRUE, method = c("logistic", "probit", "loglog", "cloglog", "cauchit")
psi.bisquare : function (u, c = 4.685, deriv = 0)
psi.hampel : function (u, a = 2, b = 4, c = 8, deriv = 0)
psi.huber : function (u, k = 1.345, deriv = 0)
qda : function (x, ...)
rational : function (x, cycles = 10, max.denominator = 2000, ...)
renumerate : function (x)
rlm : function (x, ...)
rms.curv : function (obj)
rnegbin : function (n, mu = n, theta = stop("'theta' must be specified"))
sammon : function (d, y = cmdscale(d, k), k = 2, niter = 100, trace = TRUE, magic
select : function (obj)
Shepard : function (d, x, p = 2)
stdres : function (object)
stepAIC : function (object, scope, scale = 0, direction = c("both", "backward", "f
    keep = NULL, steps = 1000, use.start = FALSE, k = 2, ...)
studres : function (object)
theta.md : function (y, mu, dfr, weights, limit = 20, eps = .Machine$double.eps^0
theta.ml : function (y, mu, n = sum(weights), weights, limit = 10, eps = .Machine$
    trace = FALSE)
theta.mm : function (y, mu, dfr, weights, limit = 10, eps = .Machine$double.eps^0
truehist : function (data, nbins = "Scott", h, x0 = -h/1000, breaks, prob = TRUE,
    ymax = max(est), col = "cyan", xlab = deparse(substitute(data)), bty = "n", .
ucv : function (x, nb = 1000, lower = 0.1 * hmax, upper = hmax)
width.SJ : function (x, nb = 1000, lower = 0.1 * hmax, upper = hmax, method = c("s
write.matrix : function (x, file = "", sep = " ", blocksize)
```

18 Load datasets

- After loading a package that contains data sets, the data sets are not loaded (they may be very large).
- To load a data set contained in package, use `data([name])`.

- You can (often) get help on datasets with `?` or `help([name])` [2]
- Example: `phones` data set in the `MASS` package - add and remove it

```
ls()  # user-defined data that are loaded in the current session
library(MASS) # load MASS package
ls()
rm(list=ls())
ls()
```

```
character(0)
character(0)
character(0)
```

- Why is the printout of the empty listing `character(0)`?

```
ls()
class(ls())  # ls() is a ‘character‘ vector
```

```
character(0)
[1] "character"
```

19 Explore data

- When you've loaded a data set, you should take a look at it.
- Most useful: `str` to see the data structure, `head` and `tail` to see the first and last few rows.

  Structure:

```
str(ToothGrowth)  # structure of built-in ToothGrowth dataset
```

```
'data.frame': 60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

[2] Strictly speaking, the availability of help depends on the package design - well written packages and data sets are well documented and are accompanied by short and detailed descriptions, or even papers (so-called "vignettes"). An example is the `Rcpp` package that interfaces R and C++.

Head: To display `m` rows only, add the parameter `n = m`.

```
head(ToothGrowth, n = 3)
```

```
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
```

Tail:

```
tail(ToothGrowth, n = 3)
```

```
    len supp dose
58 27.3   OJ    2
59 29.4   OJ    2
60 23.0   OJ    2
```

- These functions have many different attributes (check the help).

20 Practice: R package commands

1. Open an R shell with `M-x R`
2. [Install the `MASS` package with `install.packages` IF NOT DONE YET]

   ```
   install.packages("MASS")
   ```

3. List all data sets in `MASS` with `data(package="MASS")`

   ```
   data(package="MASS")
   ```

   ```
   Data sets in package ‘MASS’:

   abbey                       Determinations of Nickel Content
   accdeaths                   Accidental Deaths in the US 1973-1978
   Aids2                       Australian AIDS Survival Data
   ```

Animals	Brain and Body Weights for 28 Species
anorexia	Anorexia Data on Weight Change
bacteria	Presence of Bacteria after Drug Treatments
beav1	Body Temperature Series of Beaver 1
beav2	Body Temperature Series of Beaver 2
biopsy	Biopsy Data on Breast Cancer Patients
birthwt	Risk Factors Associated with Low Infant Birth Weight
Boston	Housing Values in Suburbs of Boston
cabbages	Data from a cabbage field trial
caith	Colours of Eyes and Hair of People in Caithness
Cars93	Data from 93 Cars on Sale in the USA in 1993
cats	Anatomical Data from Domestic Cats
cement	Heat Evolved by Setting Cements
chem	Copper in Wholemeal Flour
coop	Co-operative Trial in Analytical Chemistry
cpus	Performance of Computer CPUs
crabs	Morphological Measurements on Leptograpsus Crabs
Cushings	Diagnostic Tests on Patients with Cushing's Syndrome
DDT	DDT in Kale
deaths	Monthly Deaths from Lung Diseases in the UK
drivers	Deaths of Car Drivers in Great Britain 1969-84
eagles	Foraging Ecology of Bald Eagles
epil	Seizure Counts for Epileptics
farms	Ecological Factors in Farm Management
fgl	Measurements of Forensic Glass Fragments
forbes	Forbes' Data on Boiling Points in the Alps
GAGurine	Level of GAG in Urine of Children
galaxies	Velocities for 82 Galaxies
gehan	Remission Times of Leukaemia Patients
genotype	Rat Genotype Data
geyser	Old Faithful Geyser Data
gilgais	Line Transect of Soil in Gilgai Territory
hills	Record Times in Scottish Hill Races
housing	Frequency Table from a Copenhagen Housing Conditions Survey
immer	Yields from a Barley Field Trial
Insurance	Numbers of Car Insurance claims
leuk	Survival Times and White Blood Counts for Leukaemia Patients
mammals	Brain and Body Weights for 62 Species of Land Mammals

mcycle	Data from a Simulated Motorcycle Accident
Melanoma	Survival from Malignant Melanoma
menarche	Age of Menarche in Warsaw
michelson	Michelson's Speed of Light Data
minn38	Minnesota High School Graduates of 1938
motors	Accelerated Life Testing of Motorettes
muscle	Effect of Calcium Chloride on Muscle Contraction in Ra
Hearts	
newcomb	Newcomb's Measurements of the Passage Time of Light
nlschools	Eighth-Grade Pupils in the Netherlands
npk	Classical N, P, K Factorial Experiment
npr1	US Naval Petroleum Reserve No. 1 data
oats	Data from an Oats Field Trial
OME	Tests of Auditory Perception in Children with OME
painters	The Painter's Data of de Piles
petrol	N. L. Prater's Petrol Refinery Data
phones	Belgium Phone Calls 1950-1973
Pima.te	Diabetes in Pima Indian Women
Pima.tr	Diabetes in Pima Indian Women
Pima.tr2	Diabetes in Pima Indian Women
quine	Absenteeism from School in Rural New South Wales
Rabbit	Blood Pressure in Rabbits
road	Road Accident Deaths in US States
rotifer	Numbers of Rotifers by Fluid Density
Rubber	Accelerated Testing of Tyre Rubber
ships	Ships Damage Data
shoes	Shoe wear data of Box, Hunter and Hunter
shrimp	Percentage of Shrimp in Shrimp Cocktail
shuttle	Space Shuttle Autolander Problem
Sitka	Growth Curves for Sitka Spruce Trees in 1988
Sitka89	Growth Curves for Sitka Spruce Trees in 1989
Skye	AFM Compositions of Aphyric Skye Lavas
snails	Snail Mortality Data
SP500	Returns of the Standard and Poors 500
steam	The Saturated Steam Pressure Data
stormer	The Stormer Viscometer Data
survey	Student Survey Data
synth.te	Synthetic Classification Problem
synth.tr	Synthetic Classification Problem
topo	Spatial Topographic Data

```
Traffic                    Effect of Swedish Speed Limits on Accidents
UScereal                   Nutritional and Marketing Information on US Cereals
UScrime                    The Effect of Punishment Regimes on Crime Rates
VA                         Veteran's Administration Lung Cancer Trial
waders                     Counts of Waders at 15 Sites in South Africa
whiteside                  House Insulation: Whiteside's Data
wtloss                     Weight Loss Data from an Obese Patient
```

4. Open the `help` for the data set `Boston` in `MASS` - how many rows (observations) and columns (variables) does it have?

```
help(Boston)
```

5. Load the `MASS` package into your current R session

```
library(MASS)
```

6. Load the data set `Boston` into your current R session

```
data(Boston)
```

7. List all loaded packages with `search()`

```
search()
```

```
[1] ".GlobalEnv"         "package:MASS"       "package:data.table" "ESSR"
[5] "package:stats"      "package:graphics"   "package:grDevices"  "package:utils"
[9] "package:datasets"   "package:methods"    "Autoloads"          "package:base"
```

8. List all loaded objects with `ls()` (which output do you expect?)

```
ls()
```

```
[1] "Boston"
```

9. Display the structure of `Boston` with `str(Boston)`

```
str(Boston)
```

```
'data.frame': 506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

10. Display the first **three** rows of `Boston` with `head`

```
head(Boston,3)
```

```
     crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat medv
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7
```

11. Check loaded object list with `ls()`, then remove all loaded objects with `rm(list=ls())`

```
ls()
rm(list=ls())
ls()
```

```
[1] "Boston"
character(0)
```

12. List loaded packages with `search()`. Then detach the `MASS` package with `detach("package:MASS")` and list the loaded packages again.

```
search()
detach("package:MASS")
search()
```

```
 [1] ".GlobalEnv"         "package:MASS"       "package:data.table" "ESSR"
 [5] "package:stats"      "package:graphics"   "package:grDevices"  "package:utils
 [9] "package:datasets"   "package:methods"    "Autoloads"          "package:base"
[1] ".GlobalEnv"         "package:data.table" "ESSR"               "package:stats"
 [5] "package:graphics"   "package:grDevices"  "package:utils"      "package:datas
 [9] "package:methods"    "Autoloads"          "package:base"
```

21 Saving your workspace

- When you quit an R session with `q()` or `quit()`, you're asked if you want to save the *workspace image*.
- The workspace image includes all objects that were defined in the session, like loaded libraries, datasets, variables etc.
- In the current directory, R saves your command history (in a readable text file `.Rhistory`), and all data (in a machine-readable file `.RData`).
- Quit a current R session with `y` and check those files out (open a `Dired` buffer with `C-x C-d` or find them with `C-x C-f`.

n* Customizing at startup

- When you install packages, you do not need administrative rights, even if R is installed in a read-only portion of your computer. The OS will offer you to install packages in a user directory.
- When downloading the package as part of the installation or updating process, Windows forces you to pick a mirror. You can disable this by creating your own `~/.Rprofile` file and specifying a download mirror.
  - Saved R commands: `.Rhistory`
  - Saved R variables: `.RData`
  - R profile settings: `.Rprofile`
- See also: "Fun with .Rprofile and customizing R startup" (Fischetti, 2014)

22 Practice: Customizing at startup

1. Check where the R home is:

```
R.home(component="home")
```

```
[1] "/usr/lib/R"
```

2. Check if there's a system-wide `.Rprofile` configuration file:

```
system("cd /usr/lib/R; ls -la .Rprofile")  # must use R's Home directory
```

```
ls: cannot access '.Rprofile': No such file or directory
```

3. Find out which directory Emacs (and R) consider to be your `$HOME`:

```
system("echo $HOME") # $HOME is the the same as ~/
```

```
/home/aletheia
```

4. Create a file `.Rprofile` in your Emacs `$HOME` directory and put the following lines into it [3]:

```
options(repos=c("https://mirrors.nics.utk.edu/cran/"))
options(crayon.enabled = FALSE)
message("*** Loaded .Rprofile ***")
```

5. Open a new R shell and display the value of `options()$repos` that you just reset. Every time a new R shell is started, `.Rprofile` is read. Make sure that the `message` is displayed.

```
options()$repos
```

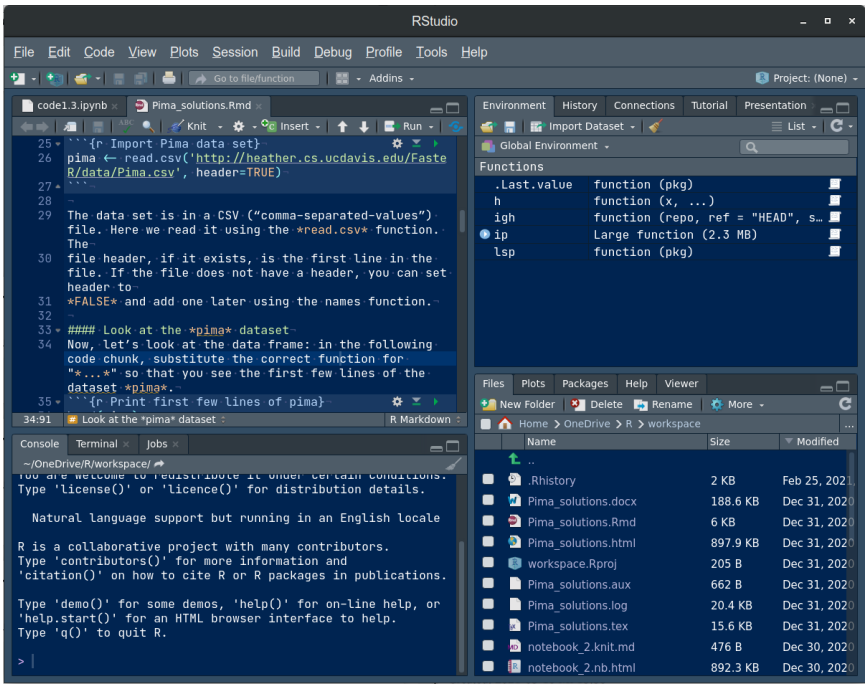
```
[1] "https://mirrors.nics.utk.edu/cran/"
```

6. Install the `remotes` package from the new location.

```
install.packages("remotes")
```

[3] You can also re-set this home directory - this FAQ explains how.

23 The RStudio IDE



- RStudio is a popular (FOSS) IDE for R with literate programming capabilities (it supports interactive R Notebooks)
- We're not using RStudio (why) but Emacs + ESS + Org-mode instead
- You can download RStudio from here - perhaps you learn to like it[4]

24 Concept Summary

- R is an easy to **learn** language to quickly and interactively analyse datasets. R is especially strong on visualization.
- R can be downloaded from `r-project.org` and installed on your computer.
- There is plenty of **help** on R available from within the program, or on the Internet using the wider community of practitioners.

[4] I don't like it because I think it's way too complicated but some developers swear by it. It contains a script editor, an R console, an environment buffer and a graphics buffer. It is fairly customizable, but nowhere near as flexible as Emacs + ESS + Orgmode.

- When you open R, you establish a working **environment**, which includes packages, functions and variables.

25 Code summary

TERM	MEANING
`license()`, `licence()`	License info
`help()`, `?help`	get help
`??[name]`	check occurrences
`demo()`	R demos
`getwd()`, `setwd()`	get/set working dir
`options(prompt=)`	set prompt
`options(repos=)`	set download repo
`options()$prompt`	display prompt
`options()$repos`	display download repo
`print(1+1)`	result of `1+1`
`quit()`, `q()`	leave R
`# ...`	comment
`library("MASS")`	load
`detach("package:[name]")`	unload package
`install.packages("MASS")`	install
`installed.packages()`	list all packages
`update.packages()`	update
`packageDescription("MASS")`	describe
`help(package="MASS")`	show
`data()`	built-in datasets
`search()`	list loaded pkgs
`searchpaths()`	list pkg search paths
`ls()`	list loaded objects
`rm(list=ls())`	unload objects

26 What next?

See also: HAL 9000: "I'm sorry Dave, I'm afraid I can't do that."

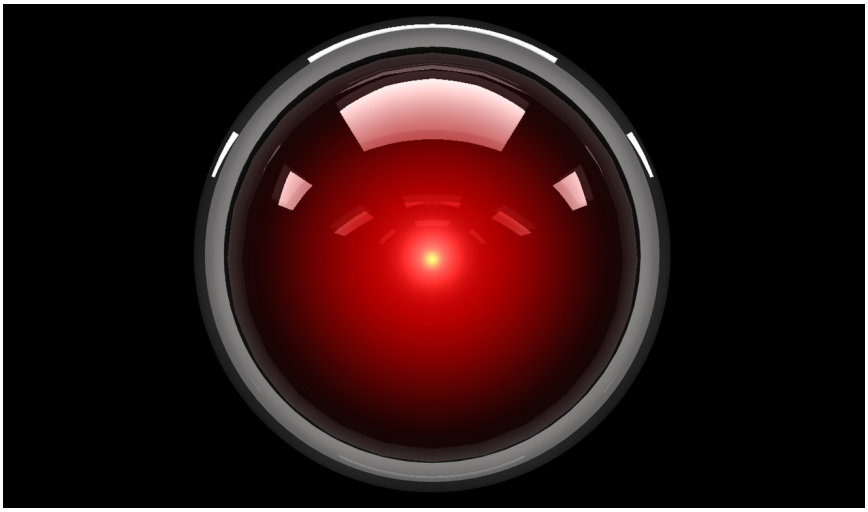


Figure 4: HAL 9000 interface (Kubrick's 2001 Space Odyssey)

27 What now? read!

- Read frequently and widely
- Go both deep and stay shallow: You've seen that I don't just cite peer-reviewed papers but blog posts, too. The truth is that I have

personally learnt a lot more from them than from scientific papers. However, this is partly a function of my experience and skill. Without these, it might be hard to distinguish what's good and bad - just like when you google any topic you don't know anything about yet. But even if you're a bloody beginner, I recommend reading widely and both deeply (with a lot of focus, e.g. when looking up terms, repeating analyses and retyping code) and shallowly (skimming articles, reading comments), because you build an associative network of terms, arguments and practices. I follow a bunch of data science experts on Twitter for the same reason. If you do this for any topic that is being discussed on a factual (rather than an overly political or emotional) basis, you'll learn more faster[5].

- For example: take a look at "R Weekly" for a weekly, curated collection of articles from the R community. This will give you an idea of the spread of information.

[5]Data science is a mixed affair when it comes to this last tip: because of the importance of statistics and models for COVID-19, public discussions e.g. on X/Twitter are often instantly politicized and emotionally charged. However, to be able to navigate these waters and still extract the common good, is an important ability that is, for me, also part of "data literacy". Learning how to read and discern different views, focus on facts and problem-solving, while not ignoring the wider problem setting, is my working definition of the scientific method.

28 What now? play!

Read: Data Scientists Should Learn Through Play

To understand why you should play (see figure 28), check the article by an active blogger and professional in the R-blogosphere, Keith McNulty, who leads data science at the global strategy consulting firm McKinsey & Co. He argues that "learning through playing around" with the software is a good way to learn (McNulty 2020) - I agree. Though I am often distracted by having to create teaching material for you, playing around on or off the command-line, looking at interesting data and combing through them using the analytical tools R offers, or checking other people's plots or inferences, is the most fun way of learning R. There's nothing wrong with reading or working through a course, watching teaching videos, of course, either. #+end$_{\text{notes}}$

29 What's the next topic?

`../img/3_maths.gif`

Arithmetic with R

30 References

- Adolfo Alvarez (25 Mar 2019). R Packages: A Beginner's Guide. Online: datacamp.com.
- Robert Becker (2004). A Brief History of S. Online: sas.waterloo.ca.
- Tilman M. Davies (2016). The Book of R. No Starch Press.
- Tony Fischetti (September 17, 2014). Fun with .Rprofile and customizing R startup. URL: R-bloggers.com.
- Kyle Gallatin (1 Nov 2018). Some Important Data Science Tools that aren't Python, R, SQL or Math. Online: towardsdatascience.com.
- Michael Grogan (23 Jul 2020). How R Still Excels Compared To Python. Online: towardsdatascience.com.
- Knuth D (1992). Literate Programming. Stanford, Center for the Study of Language and Information Lecture Notes 27.
- Norman Matloff (2019). TidyverseSceptic. Online: github.com.
- Keith McNulty (23 Jun 2020). Data Scientists Should Learn Through Play. Online: drkeithmcnulty.com.
- Robert A. Muenchen (2017). Why R is Hard to Learn. Online: r4stats.com.
- Brien Posey (5 Feb 2018). How To Navigate the File System in Windows 10's Bash Shell. Online: redmondmag.com.
- Dario Radecic (10 Sept 2020). Trying R for the First Time. Online: towardsdatascience.com.
- Gordon Shotwell (30 Dec 2019). Why I use R. Online: blog.shotwell.ca.
- Sagar Uprety (23 Jul 2020). Data Cleaning and Exploratory Analysis in Python and R. Online: towardsdatascience.com.
- Venables/Ripley (2002). Modern Applied Statistics with S. Springer. Online: researchgate.net.
- Yuleng Zeng (28 Aug 2018). An Introduction to R and LaTeX. Online: bookdown.org.

- R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

31 Hints

31.1 Download from CRAN

Mirror sites are called that way because they are actual identical copies of the original site. The quality of the cloned page is monitored. The result looks interesting (to me). You can see how well maintained a particular mirror site is.

31.2 Opening R for the first time

The projects listed here (by no means a complete list!) are divided in applications and infrastructure projects. **Applications** of R include bioinformatics (e.g. in the medical sciences or in genomics), geospatial statistics (anything related to maps), and finance (R is strong with this one!). **Infrastructure** includes incorporation of R in Wikis (like Wikipedia) - for example to generate plots on the fly - and ESS ("Emacs Speaks Statistics"), which is the interface to the extensible text editor that I'm using (e.g. to create all documentation for this course - essentially from one text file). An alternative to ESS is the highly popular IDE (Integrated Development Environment) RStudio. We will not be using it in this course but I encourage you to check it out, try it and see if you like it, especially if my teaching tempo is too slow for you!

31.3 Distribution license

Go to GNU Software to see a list of all programs distributed under the GPL. These programs constitute the GNU system of free software. Looking through the list, I noticed the following programs that I have used: Chess (chess game implementation), Emacs (extensible text editor that I am using in this very moment), Gimp (image manipulation), Gnome (desktop for my operating system, Ubuntu Linux), and so on... 425 programs are listed here alone (29 Aug 2020).

31.4 The R Project

There is no special connection between LaTeX and R, except that both are free software programs, one for formatting (especially when mathematical formulas need to be presented), the other one for statistical calculations and visualisation. However, to communicate data analysis results and to make the analysis process itself reproducible, a combination between these two goals (formatting/programming) is desirable. This is exactly what "literate programming" (Knuth 1984) does. There is also a program called "R Markdown" to create documents that enables you e.g. to created HTML, PDF, ePUB and Kindle books with only one source. You can find examples at bookdown.org. See also Zeng (2018) for a brief introduction to both R and LateX - sufficient to get started - written apparently as a minimal example for bookdown. For LaTeX there are also cloud editors like overleaf.com.

31.5 R Packages

You can directly search for this dataset - I usually take the search string "`r doc [name]`, in this case `r doc MASS boston`, which gets me straight to this page. At the top, you can read that "The `Boston` data frame has 506 rows and 14 columns". There's also an R Notebook, which shows various aspects of this dataset.

Another way to find the answer is by using the command `str()` that you already know: `str(Boston` contains the answer in the first line - as long as `MASS` has been loaded. (Check out what happens if not by closing the R session with `q()` (don't save the workspace) and reopening it again.

The simplest way is to type `help(Boston)` (again, only after loading the `MASS` package).