Psychology 350: An introduction to R for Psychological Research

William Revelle
Northwestern University
Evanston, Illinois USA

https://personality-project.org/courses/350

Spring, 2024

Outline

Part I: What is R, where did it come from, why use it

- Installing R and adding packages: the building blocks of R

Part II: A brief introduction – an overview

- R is just a fancy (very fancy) calculator
- Descriptive data analysis
- Some inferential analysis

Part III R is a powerful statistical system

- Data entry (detail and practice)
- Descriptive (again)
- Inferential (t and F with more practice)
- Regression
- Basic R commands

Part IV: Psychometrics

- Reliability and its discontents
- EFA, CFA, SEM

Part V: Help and More Help

- List of useful commands

Part VI: The psych package and more practice

Outline of Part I

- A brief history of computers and computing
- What is R?
    - Where did it come from, why use it?
    - Misconceptions
- Installing R on your computer and adding packages
- Installing for your operating system
    - R-Applications
- Installing and using packages
    - What are packages
    - Installing packages
- Building Blocks
    - Objects
- R is a language
    - Objects act on objects
- Package dependencies
- -> Part II

Ada Lovelace: The first programmer

1. The Babbage Analytical Engine (1838) was a design for a punch card based calculating machine (generalized from an automatic loom) but was never actually built. (see Bromley, 1982).
2. Ada Lovelace wrote a "memoir" on how to program it (Lovelace, 1842). She was perhaps the first programmer.
3. The U.S. Department of Defense (1980) initiated an operating system/language in her honor (ADA) to get around the problem of too many different programming systems.

The von Neuman machine and programable analysis

1. The "Van Neuman" machine instigated programs that could act depending on the state of the machine. (See Isaacson, 2014)
   - Developed to calculate flight paths for artillery and rockets.
   - Turing had developed computers for breaking codes, but this work was classified and not known about until much later.
2. Large scale computing was for rocket design, military applications, the census, meterology.
3. "Computers wore skirts" at NASA (Johnson, nd: see Katherine Johnson's 90th birthday
4. The "computers" could do in a morning what the engineers would take all day to do NASA history 1935-1970

Computers go to college/university

1. The use of computers in physics, oceanography, and psychology exploded in the 1950s-1960s, as it did in business
2. Originally using electronic tubes
   - Univac (1950), IBM 704 (1954), 7090 (1959)
3. Control Data Corporation and the "super computer"
   - CDC1604 (designed by Seymour Cray) was the first commercially successful computer with transistors (1960)
   - CDC 3600 (1963)
   - CDC 6400 and 6600 (1964)
4. IBM 360 series (vapor ware?)
5. These were all large systems that eventually allowed remote time-sharing.
6. Commands were submitted using punched cards (Hollerith aka IBM cards)
7. Turn around time ranged from minutes (late at night) to 8-12 hours (during the day).

Computer languages (a brief summary

1. Machine language (written in Boolean at the bit level), organized as a very limited set of instructions
2. Assembly languages would take somewhat higher level instructions and generate the machine language translation
3. Compiled languages (e.g. FORTRAN, LISP, C) were more readable, but needed to be "compiled" into Assembly which then in turn would run machine level commands.
4. Operating systems (OS) provide higher level functionality, but tend to be written in Assembly.
5. Large sets of programs would be organized for specific tasks; e.g., for statistics, forecasting, modeling.

Statisical computing languages for mainframes

1. BMD and BMDP for biomedical research (UCLA)
2. SAS for agricultural research (North Carolina State)
3. SPSS (Statistical Package for the Social Sciences) (Stanford)
   - SPSS was written in Fortran for the IBM family of computers
   - Northwestern University was a major source of translation of SPSS into CDC Fortran
4. Although originally developed at universities, these three statistical systems eventually became stand alone and proprietary companies.

From mainframes to minis to micros to personal computers

1. The three main statistical systems were written in Fortran for mainframe computers.
2. Even with the introduction of remote terminals, the main frame syntax remains in these languages.
3. With the introduction of personal computers, statistics could now be done interactively.
4. Languages for personal computers included SysStat (by Leland Wilkinson) as well as S (developed at Bell Labs) and S+
5. R owes its history to S (originally conceived as S for the Mac).

R: Statistics for all us

1. What is it?
2. Why use it?
3. Common (mis)perceptions of R
4. Examples for psychologists
   - graphical displays
   - basic statistics
   - advanced statistics
5. List of major commands and packages
6. Some basic programming concepts in R
7. An overview of the *psych* package
8. Extended practice on your data sets

R: What is it?

1. R: An international collaboration
2. R: The open source - public domain version of S+
3. R: Written by statisticians (and some of us) for statisticians (and the rest of us)
4. R: Not just a statistics system, also an extensible language.
    - This means that as new statistics are developed they tend to appear in R far sooner than elsewhere.
    - R facilitates asking questions that have not already been asked.

Statistical Programs for Psychologists

- General purpose programs
  - R
  - S+
  - SAS
  - SPSS
  - STATA
  - Systat
- Specialized programs
  - Mx
  - EQS
  - AMOS
  - LISREL
  - MPlus
  - Your favorite program

Statistical Programs for Psychologists

- General purpose programs
    - R
    - $+
    - $A$
    - $P$$
    - $TATA
    - $y$tat
- Specialized programs
    - Mx (OpenMx is part of R)
    - EQ$
    - AMO$
    - LI$REL
    - MPlu$
    - Your favorite program

R: A way of thinking

- “R is the lingua franca of statistical research. Work in all other languages should be discouraged.”
- “This is R. There is no if. Only how.”
- “Overall, SAS is about 11 years behind R and S-Plus in statistical capabilities (last year it was about 10 years behind) in my estimation.”
- Q: My institute has been heavily dependent on SAS for the past while, and SAS is starting to charge us a very deep amount for license renewal.... The team is [considering] switching to R, ... I am talking about the entire institute with considerable number of analysts using SAS their entire career. ... What kind of problems and challenges have you faced?
  A: “One of your challenges will be that with the increased productivity of the team you will have time for more intellectually challenging problems. That frustrates some people.”

R is open source, how can you trust it?

- Q: “When you use it [R], since it is written by so many authors, how do you know that the results are trustable?”
- A: “The R engine [...] is pretty well uniformly excellent code but you have to take my word for that. Actually, you don’t. The whole engine is open source so, if you wish, you can check every line of it. If people were out to push dodgy software, this is not the way they’d go about it.”
- Q: Are R packages bug free?
- A: No. But bugs are fixed rapidly when identified.
- Q: How does function x work? May I adapt it for my functions.
- A: Look at the code. Borrow what you need.

What is R?: Technically

- R is an open source implementation of S (The statistical language developed at Bell Labs). (S-Plus is a commercial implementation)
- R is a language and environment for statistical computing and graphics. R is available under GNU Copy-left
- R version 1.0.0 was released on February 29, 2000
- R is a group project run by a core group of developers (with new releases semiannually). The current version of R is 4.3.3
- R is an integrated suite of software facilities for data manipulation, calculation and graphical display.

(Adapted from Robert Gentleman and the r-project.org web page)

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It is:

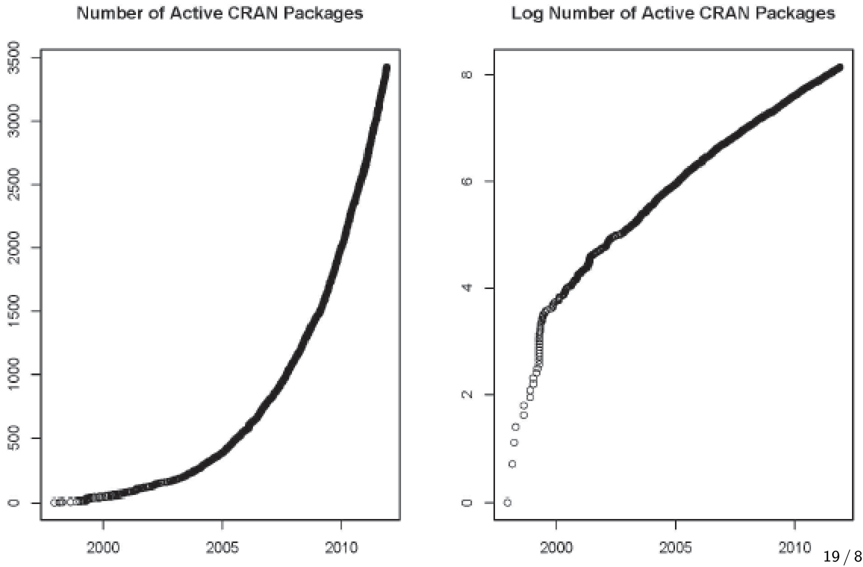
1. an effective data handling and storage facility,
2. a suite of operators for calculations on arrays, in particular matrices,
3. a large, coherent, integrated collection of intermediate tools for data analysis,
4. graphical facilities for data analysis and display either on-screen or on hardcopy, and
5. a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

“Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages ... available through the CRAN family of Internet sites covering a very wide range of modern statistics.” (Adapted from r-project.org web page)

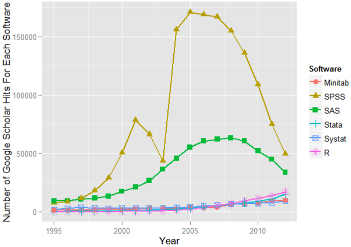
R: A brief history

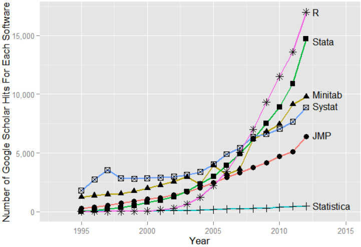
- 1991-93: Ross Dhaka and Robert Gentleman begin work on R project for Macs at U. Auckland (S for Macs).
- 1995: R available by ftp under the General Public License.
- 96-97: mailing list and R core group is formed.
- 2000: John Chambers, designer of S joins the Rcore (wins a prize for best software from ACM for S)
- 2001-2017: Core team continues to improve base package with a new release every 6 months (now more like yearly).
- Many others contribute "packages" to supplement the functionality for particular problems.
    - 2003-04-01: 250 packages
    - 2004-10-01: 500 packages
    - 2007-04-12: 1,000 packages
    - 2009-10-04: 2,000 packages
    - 2011-05-12: 3,000 packages
    - 2014-05-16: 5,547 packages (on CRAN) + 824 bioinformatic packages on BioConductor
    - 2015-05-20 6,678 packages (on CRAN) + 1024 bioinformatic packages + ?,000s on GitHub
    - 2016-03-31 8,427 packages (on CRAN) + 1,104 bioinformatic packages + ?,000s on GitHub/R-Forge
    - 2017-05-21 10,677 packages (on CRAN) + 1,383 bioinformatic packages + ?,000s on GitHub
    - 2018-09-30 13,127packages (on CRAN) + 1,560 bioinformatic packages + ?,000s on GitHub

Rapid and consistent growth in packages contributed to R



Popularity compared to other statistical packages





http://r4stats.com/articles/popularity/ considers various measures of popularity

1. discussion groups
2. blogs
3. Google Scholar citations ($> 140K$ citations, $\approx 35K/year$)
4. Google Page rank
5. Number of downloads (see http://www.rpackages.io/packages)

R as a way of facilitating replicable science

1. R is not just for statisticians, it is for all research oriented psychologists.
2. R scripts are published in psychology journals to show new methods:
   - *Psychological Methods*
   - *Psychological Science*
   - *Journal of Research in Personality*
3. R based data sets are now accompanying journal articles:
   - The *Journal of Research in Personality* now accepts R code and data sets.
   - JRP special issue in R,
4. By sharing our code and data the field can increase the possibility of doing replicable science.

Reproducible Research: Sweave and KnitR

> *Sweave is a tool that allows to embed the R code for complete data analyses in LATEXdocuments. The purpose is to create dynamic reports, which can be updated automatically if data or analysis change. Instead of inserting a prefabricated graph or table into the report, the master document contains the R code necessary to obtain it. When run through R, all data analysis output (tables, graphs, etc.) is created on the fly and inserted into a final LATEXdocument. The report can be automatically updated if data or analysis change, which allows for truly reproducible research.*

Friedrich Leisch (2002). Sweave: Dynamic generation of statistical reports using literate data analysis. I

Supplementary material for journals can be written in Sweave/KnitR/ RMarkdown

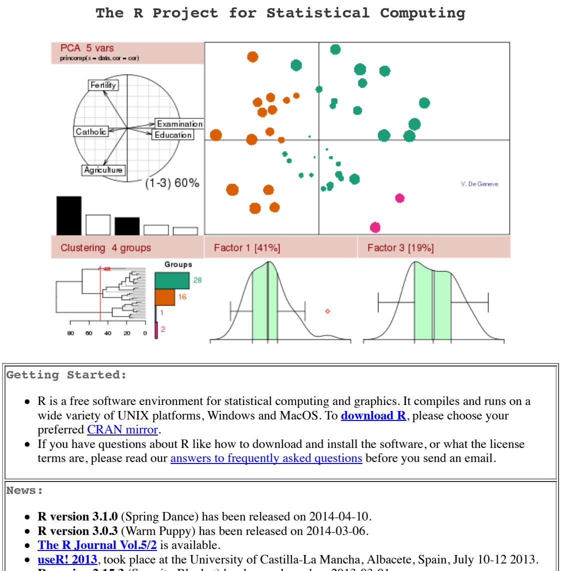
Misconception: R is hard to use

1. R doesn't have a GUI (Graphical User Interface)
   - Partly true, many use syntax.
   - Partly not true, GUIs exist (e.g., R Commander, R-Studio).
   - Quasi GUIs for Mac and PCs make syntax writing easier.
2. R syntax is hard to use
   - Not really, unless you think an iPhone is hard to use.
   - Easier to give instructions of 1-4 lines of syntax rather than pictures of menu after menu to pull down.
   - Keep a copy of your syntax, modify it for the next analysis.
3. R is not user friendly: A personological description of R
   - R is Introverted: it will tell you what you want to know if you ask, but not if you don't ask.
   - R is Conscientious: it wants commands to be correct.
   - R is not Agreeable: its error messages are at best cryptic.
   - R is Stable: it does not break down under stress.
   - R is Open: new ideas about statistics are easily developed.

Misconceptions: R is hard to learn – some interesting facts

1. With a brief web based tutorial http://personality-project.org/r, 2nd and 3rd year undergraduates in psychological methods and personality research courses are using R for descriptive and inferential statistics and producing publication quality graphics.
2. More and more psychology departments are using it for graduate and undergraduate instruction.
3. R is easy to learn, hard to master
    - R-help newsgroup is very supportive (usually)
    - Multiple web based and pdf tutorials see (e.g., http://www.r-project.org/)
    - Short courses using R for many applications. (Look at APS program). Go to March, 2017 APS Observer article by Sara Weston and Debbie Yee.
4. Books and websites for SPSS and SAS users trying to learn R (e.g., http://r4stats.com/) by Bob Muenchen (look for link to free version).

Go to the R.project.org



The R Project for Statistical Computing

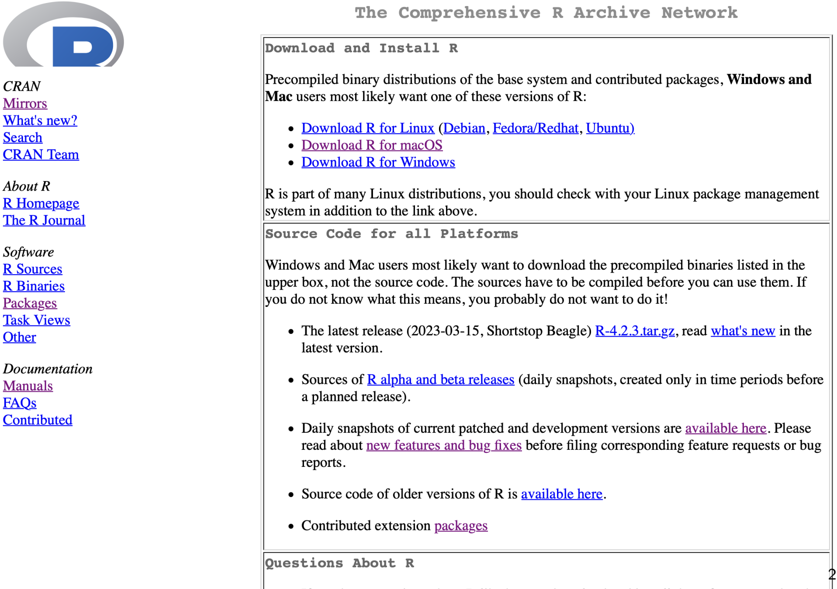
Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News:

- **R version 3.1.0** (Spring Dance) has been released on 2014-04-10.
- **R version 3.0.3** (Warm Puppy) has been released on 2014-03-06.
- **The R Journal Vol.5/2** is available.
- **useR! 2013**, took place at the University of Castilla-La Mancha, Albacete, Spain, July 10-12 2013.
- **R version 2.15.3** (Security Blanket) has been released on 2013-03-01.

Go to the Comprehensive R Archive Network (CRAN)



The Comprehensive R Archive Network

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

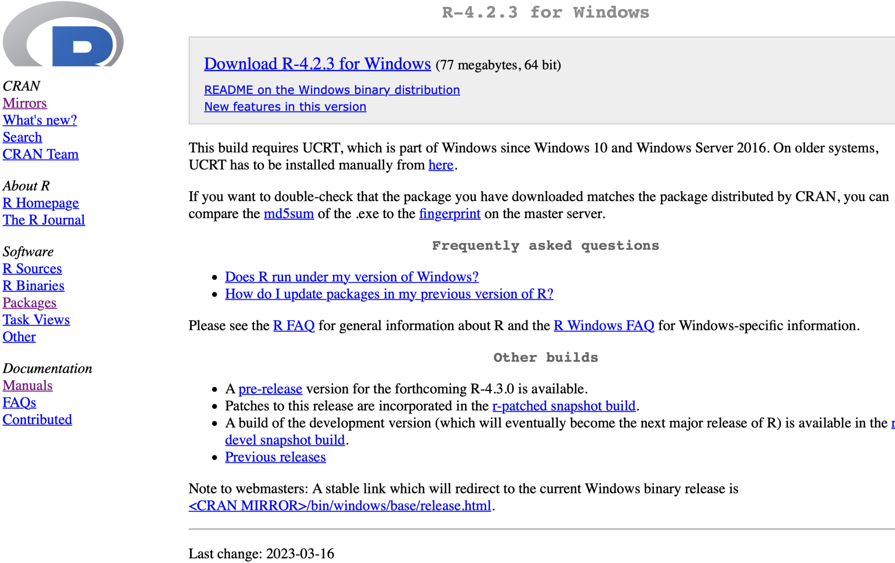
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2023-03-15, Shortstop Beagle) R-4.2.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

Download and install the appropriate version – PC



R-4.2.3 for Windows

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Download R-4.2.3 for Windows (77 megabytes, 64 bit)
README on the Windows binary distribution
New features in this version

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

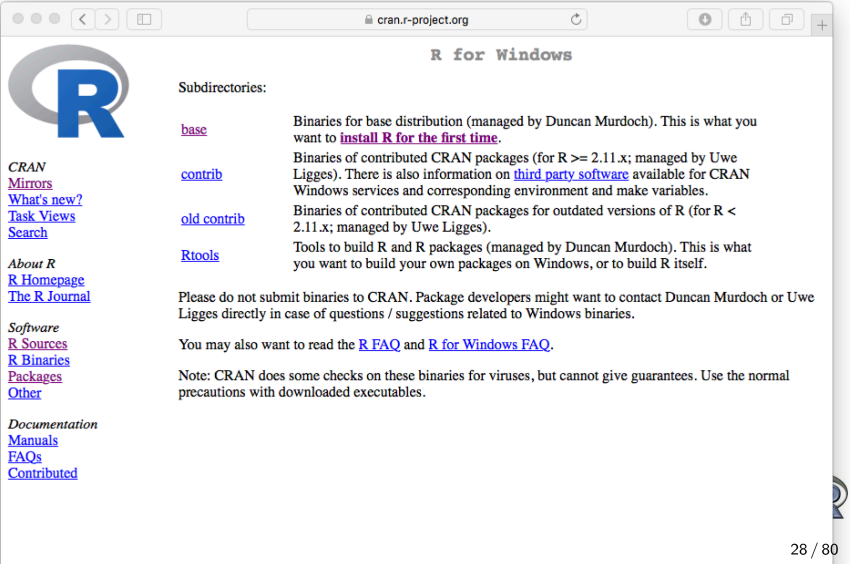
Other builds

- A pre-release version for the forthcoming R-4.3.0 is available.
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r
devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2023-03-16

Download and install the appropriate version – PC



R for Windows

Subdirectories:

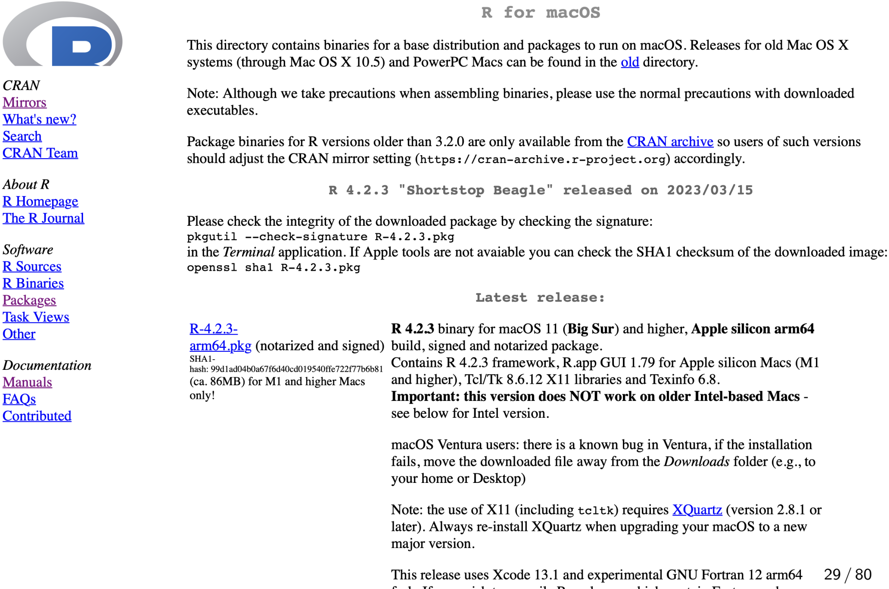
base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.11.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.11.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Download and install the appropriate version – Mac



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

R for macOS

This directory contains binaries for a base distribution and packages to run on macOS. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Package binaries for R versions older than 3.2.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting (`https://cran-archive.r-project.org`) accordingly.

R 4.2.3 "Shortstop Beagle" released on 2023/03/15

Please check the integrity of the downloaded package by checking the signature:
`pkgutil --check-signature R-4.2.3.pkg`
in the *Terminal* application. If Apple tools are not avaiable you can check the SHA1 checksum of the downloaded image:
`openssl sha1 R-4.2.3.pkg`

Latest release:

R-4.2.3-arm64.pkg (notarized and signed)
SHA1-hash: 99d1ad04b0a67f6d40cd019540ffe722f77b6b81
(ca. 86MB) for M1 and higher Macs only!

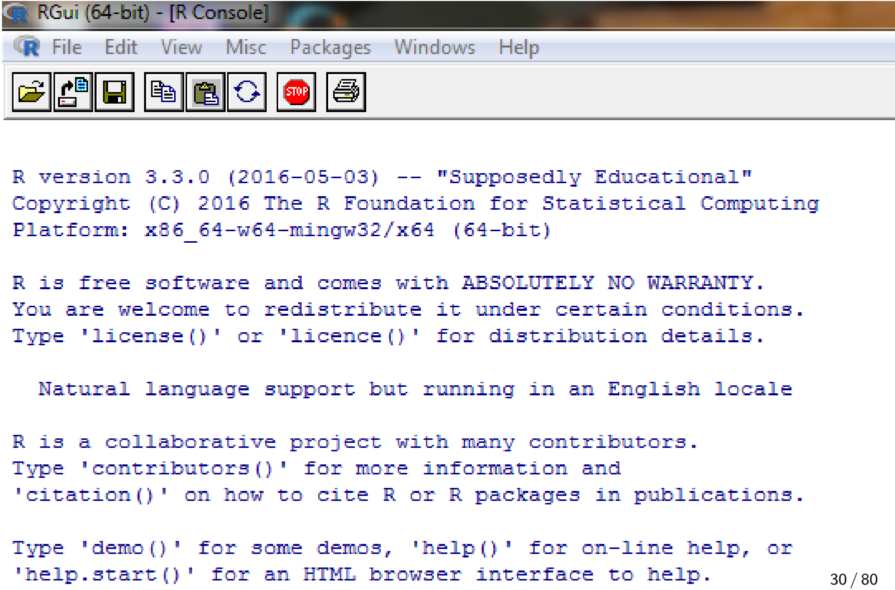
R 4.2.3 binary for macOS 11 (**Big Sur**) and higher, **Apple silicon arm64** build, signed and notarized package.
Contains R 4.2.3 framework, R.app GUI 1.79 for Apple silicon Macs (M1 and higher), Tcl/Tk 8.6.12 X11 libraries and Texinfo 6.8.
Important: this version does NOT work on older Intel-based Macs - see below for Intel version.

macOS Ventura users: there is a known bug in Ventura, if the installation fails, move the downloaded file away from the *Downloads* folder (e.g., to your home or Desktop)

Note: the use of X11 (including `tcltk`) requires XQuartz (version 2.8.1 or later). Always re-install XQuartz when upgrading your macOS to a new major version.

This release uses Xcode 13.1 and experimental GNU Fortran 12 arm64

Starting R on a PC



```
R version 3.3.0 (2016-05-03) -- "Supposedly Educational"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.
```

Start up R and get ready to play (most recent Mac version)

```
R Under development (unstable) (2023-03-17 r83997) -- "Unsuffered C
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (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.

[R.app GUI 1.79 (8196) aarch64-apple-darwin20]

[Workspace restored from /Users/WR/.RData]
[History restored from /Users/WR/.Rapp.history]
```

Check the version number for R ≥ 4.3.3) and for psych (≥2.4.3)

R code

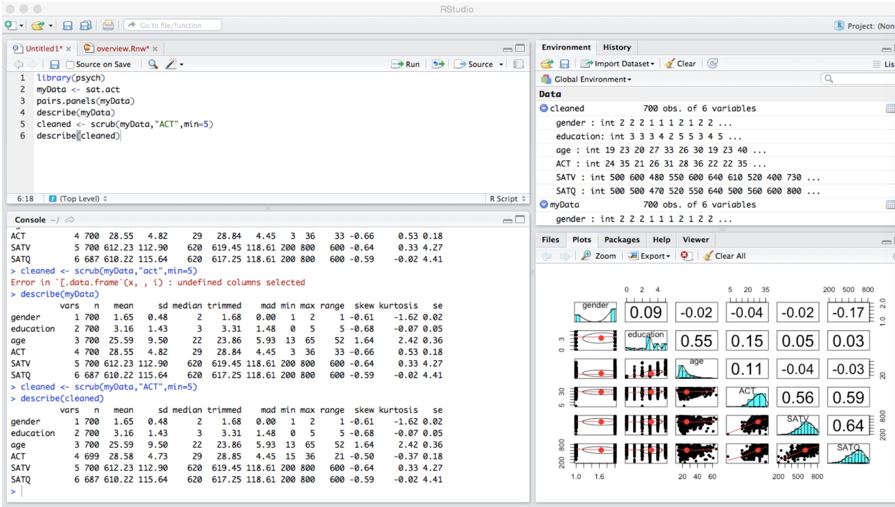
```
> library(psych) #make the psych package active (if installed)
> sessionInfo()  #what packages are active?  (need to installed first
```

```
R Under development (unstable) (2023-03-17 r83997)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib


Random number generation:
 RNG:     Mersenne-Twister
 Normal:  Inversion
 Sample:  Rounding

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Chicago
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] psychTools_2.3.3 psych_2.3.3

loaded via a namespace (and not attached):
[1] compiler_4.3.0  tools_4.3.0     parallel_4.3.0  foreign_0.8-84  nlme_3.1-162    mnormt_2.
[8] lattice_0.20-45
```

Various ways to run R

1. UNIX (and *NIX like) environments
   - Can be scripted for use on remote servers
   - Particularly fast if on remote processors with many cores
   - RStudio Server as "Integrated Development Environment" (IDE)
2. PC
   - quasi GUI + text editor of choice
   - RStudio as "Integrated Development Environment" (IDE) (recommended by Sara)
3. Mac
   - R.app + text editor of choice (preferred by Bill)
   - RStudio as "Integrated Development Environment" (IDE) (preferred by David)
   - allows for multiple cores for parallel processing
4. From the web
   - R Fiddle (but R = 3.1.2 and psych = 1.3.12)

R Studio is a useful "Integrated Development Environment" (IDE)

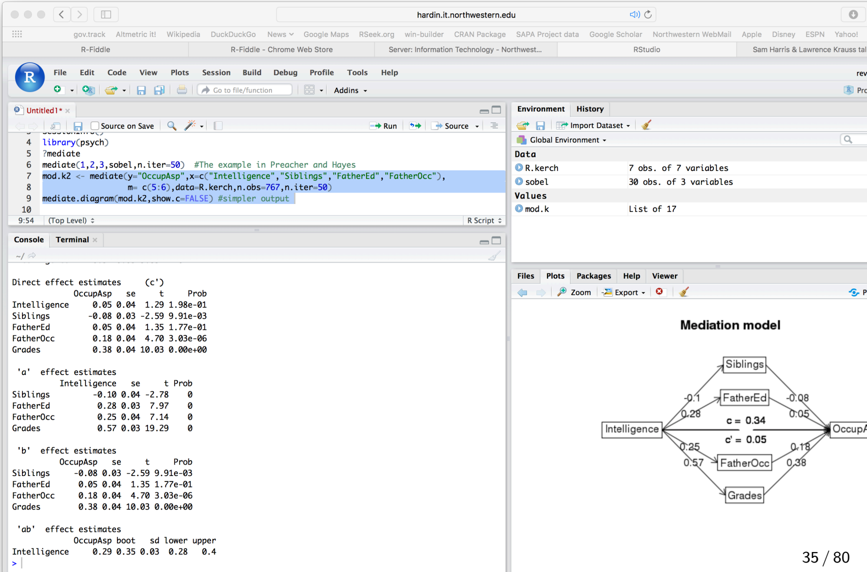


```
library(psych)
myData <- sat.act
pairs.panels(myData)
describe(myData)
cleaned <- scrub(myData,"ACT",min=5)
describe(cleaned)
```

```
Data
cleaned              700 obs. of 6 variables
   gender : int 2 2 2 1 1 1 2 1 2 2 ...
   education: int 3 3 3 4 2 5 5 3 4 5 ...
   age : int 19 23 20 27 33 26 30 19 23 40 ...
   ACT : int 24 35 21 26 31 28 36 22 22 35 ...
   SATV : int 500 600 480 550 600 640 610 520 400 730 ...
   SATQ : int 500 500 470 520 550 640 500 560 600 800 ...
myData               700 obs. of 6 variables
   gender : int 2 2 2 1 1 1 2 1 2 2 ...
```

```
ACT          4 700  28.55   4.82     29   28.84   4.45   3  36    33 -0.66     0.53 0.18
SATV         5 700 612.23 112.90    620  619.45 118.61 200 800   600 -0.64     0.33 4.27
SATQ         6 687 610.22 115.64    620  617.25 118.61 200 800   600 -0.59    -0.02 4.41
> cleaned <- scrub(myData,"act",min=5)
Error in `[.data.frame`(x, , i) : undefined columns selected
> describe(myData)
          vars   n   mean     sd median trimmed    mad min max range  skew kurtosis   se
gender       1 700   1.65   0.48      2    1.68   0.00   1   2     1 -0.61    -1.62 0.02
education    2 700   3.16   1.43      3    3.31   1.48   0   5     5 -0.68    -0.07 0.05
age          3 700  25.59   9.50     22   23.86   5.93  13  65    52  1.64     2.42 0.36
ACT          4 700  28.55   4.82     29   28.84   4.45   3  36    33 -0.66     0.53 0.18
SATV         5 700 612.23 112.90    620  619.45 118.61 200 800   600 -0.64     0.33 4.27
SATQ         6 687 610.22 115.64    620  617.25 118.61 200 800   600 -0.59    -0.02 4.41
> cleaned <- scrub(myData,"ACT",min=5)
> describe(cleaned)
          vars   n   mean     sd median trimmed    mad min max range  skew kurtosis   se
gender       1 700   1.65   0.48      2    1.68   0.00   1   2     1 -0.61    -1.62 0.02
education    2 700   3.16   1.43      3    3.31   1.48   0   5     5 -0.68    -0.07 0.05
age          3 700  25.59   9.50     22   23.86   5.93  13  65    52  1.64     2.42 0.36
ACT          4 699  28.58   4.73     29   28.85   4.45  15  36    21 -0.50    -0.37 0.18
SATV         5 700 612.23 112.90    620  619.45 118.61 200 800   600 -0.64     0.33 4.27
SATQ         6 687 610.22 115.64    620  617.25 118.61 200 800   600 -0.59    -0.02 4.41
>
```

R Studio may be run on a remote server



```r
library(psych)
?mediate
mediate(1,2,3,sobel,n.iter=50)  #The example in Preacher and Hayes
mod.k2 <- mediate(y="OccupAsp",x=c("Intelligence","Siblings","FatherEd","FatherOcc"),
                  m= c(5:6),data=R.kerch,n.obs=767,n.iter=50)
mediate.diagram(mod.k2,show.c=FALSE) #simpler output
```

Data

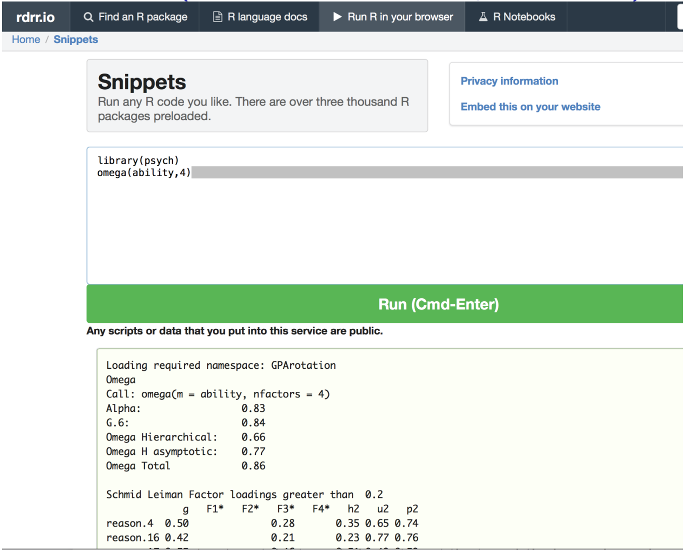
R.kerch	7 obs. of 7 variables
sobel	30 obs. of 3 variables

Values

mod.k	List of 17

```
Direct effect estimates     (c')
             OccupAsp   se     t     Prob
Intelligence     0.05 0.04  1.29 1.98e-01
Siblings        -0.08 0.03 -2.59 9.91e-03
FatherEd         0.05 0.04  1.35 1.77e-01
FatherOcc        0.18 0.04  4.70 3.03e-06
Grades           0.38 0.04 10.03 0.00e+00

 'a'  effect estimates
            Intelligence   se     t Prob
Siblings           -0.10 0.04 -2.78    0
FatherEd            0.28 0.03  7.97    0
FatherOcc           0.25 0.04  7.14    0
Grades              0.57 0.03 19.29    0

 'b'  effect estimates
           OccupAsp   se     t     Prob
Siblings      -0.08 0.03 -2.59 9.91e-03
FatherEd       0.05 0.04  1.35 1.77e-01
FatherOcc      0.18 0.04  4.70 3.03e-06
Grades         0.38 0.04 10.03 0.00e+00

 'ab'  effect estimates
             OccupAsp boot   sd lower upper
Intelligence     0.29 0.35 0.03  0.28   0.4
>
```

`https://rdrr.io` allows to run on a remote server (but R = 4.0.3 and psych = 2.0.9)



Snippets

Run any R code you like. There are over three thousand R packages preloaded.

```
library(psych)
omega(ability,4)
```

Any scripts or data that you put into this service are public.

```
Loading required namespace: GPArotation
Omega
Call: omega(m = ability, nfactors = 4)
Alpha:                 0.83
G.6:                   0.84
Omega Hierarchical:    0.66
Omega H asymptotic:    0.77
Omega Total            0.86

Schmid Leiman Factor loadings greater than  0.2
              g   F1*   F2*   F3*   F4*   h2   u2   p2
reason.4  0.50              0.28        0.35 0.65 0.74
reason.16 0.42              0.21        0.23 0.77 0.76
```

R is extensible: The use of “packages”

1. More than 19,366 packages are available for at CRAN (and growing daily. It was 16,222 in 2022 and 10,677 in 6 years ago).
2. Can search all packages that do a particular operation by using the sos package (probably disappearing soon).
   - `install.packages("sos")` #if you haven't already
   - library(sos) # make it active once you have it
     - `findFn`(”X”) #will search a web data base for all packages/functions that have ”X”
     - `findFn`(”principal components”) #will return 2,318 matches from 180 packages and reports the top 400
     - `findFn`(”Item Response Theory”) # will return 394 matches in 93 packages
     - `findFn`(”INDSCAL ”) # will return 18 matches in 6 packages.
3. install.packages(”X”) will install a particular package (add it to your R library (you need to do this just once)
4. library(X) #will make the package X available to use if it has been installed (and thus in your library)

A small subset of very useful packages

- General use
  - core R
  - MASS
  - lattice
  - lme4 (core)
  - psych
- Special use
  - ltm/eRm/mirt
  - sem
  - lavaan/OpenMx
  - GPArotation
  - mnormt
  - $>$ 19,366 known
  - $+$?

- General applications
  - most descriptive and inferential stats
  - Modern Applied Statistics with S
  - Lattice or Trellis graphics
  - Linear mixed-effects models
  - Personality/psychometrics/general purpose
- More specialized packages
  - Latent Trait Model (IRT)
  - SEM and CFA (RAM path notation)
  - SEM and CFA (multiple groups)
  - Jennrich rotations
  - Multivariate distributions
  - Thousands of more packages on CRAN
  - Code on GitHub/ webpages/journal articles

Even more very useful packages (see also Computer World list)

General use	General applications
devtools	Development tools from GitHub
readxl	input from excel
foreign	input from SPSS, , etc. (part of Core)
RMySQL	input from MySQL
readr	fast input for very large csv files
rio	simple to use integrated input/output
Special use	**More specialized packages**
plyr & dplyr	reshape from wide to long etc.
data.table	faster data handling for large data sets
knitr	integrate markdown documentation with R
sweave	integrate LaTeXdocumentation with R
ggplot2	powerful grammar of graphics
$>$ 19,336	Thousands of more packages on CRAN
+ ?	Code on webpages/journal articles

Ok, how do I get it: Getting started with R

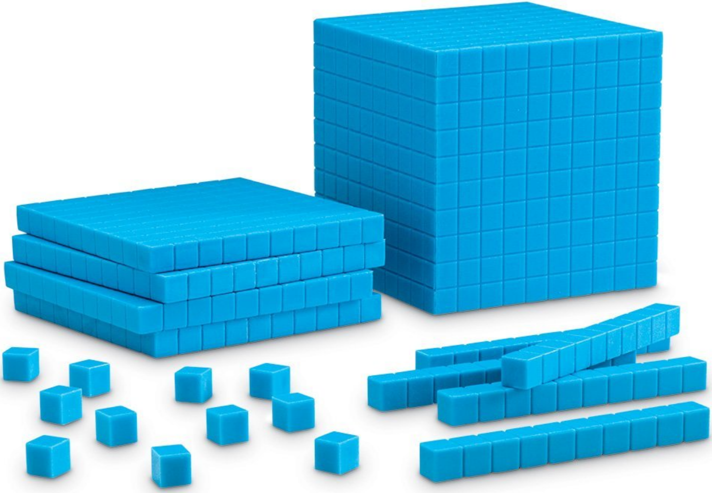
- Download from R Cran (http://cran.r-project.org/)
  - Choose appropriate operating system and download compiled R
- Install R (current version is 4.3.3) (See a tutorial on how to install R and various packages at http://personality-project.org/r/psych)
- Start R
- Add useful packages (just need to do this once)
  - install.packages(”ctv”) #this downloads the task view package
  - library(ctv) #this activates the ctv package
  - install.views(”Psychometrics”) #among others
  - Take a 5 minute break
- Activate the package(s) you want to use today (e.g., *psych*)
  - library(psych) #necessary for most of today's examples
- Use R

Annotated installation guide: don’t type the >

```
> install.packages("ctv")


> library(ctv)

> install.views("Psychometrics")




#or just install a few packages
> install.packages("psych",
           dependencies=TRUE)
#which installs psych and its
          required packages


> install.packages("GPArotation")
> install.packages("mnormt")
```

- Install the task view installer package. You might have to choose a “mirror” site.
- Make it active
- Install all the packages in the “Psychometrics” task view. This will take a few minutes.
- Or, just install one package (e.g., psych)
- as well as a few suggested packages that add functionality for factor rotation, multivariate normal distributions, etc.

Building Blocks (from Sara Weston)



R Basics

R is an object-oriented programming language.

R is a language

- Think of R like having a conversation with a specific person.
- They (R) have their own language, and you need to learn how to speak it.
- R is not very forgiving of mistakes, so pay attention to grammar and punctuation.

R is an object-oriented programming language.
What is an object? // Everything!

Single-value objects

- The most basic object contains a single value.
  4
- Objects can be numbers, strings, or logical values.
  ```
   4
  "female"x
  TRUE
  ```
- We can save objects to our environment by assigning them to names.
- Note, although better style is to use the “get” command, you can also use the = (which means replace) command.
  ```
  happy <- 4     #read as happy gets 4, or
              happy is given the value of 4
  gender = "female"
      #read as gender is given the value of 4
  ```
- The *only way* to create or change an object is to assign it to a name.

Single-value objects (aka in some languages as scalers)

You can call objects using their name. Writing the name of an object will print its value to your console.

```
 happy
[1] 4
```

You can also use the name of an object as a substitute for its value.

```
 happy + 8
[1] 12
```

Vectors

A vector is an ordered set of values. Some of us would call this an ordered n-tuplet.

```
genders <-  c("male","male","female","male",
                      "male","male")
emotions <-  c(4, 7, happy, 7, 3, 8)
```

(We use the c for the *concatenate* operator).

Important rules:

- Order matters
- Each element included in the vector is of the same class (numerical, logical, character) which will be the class of the object

```
 class(emotions)
[1] "numeric"
 class(genders)
[1] "character"
```

Order matters

Values in a vector are given a specific position and they will always be printed in that position.
(Hence the term ordered n-tuplet.)

```
emotions
[1] 4 7 4 7 3 8
```

Same class

You cannot mix numbers and strings and logical values in a single vector.

```
bad.vector = c(7, 9, "2")
#by typing the name, we are asking for its contents

bad.vector
the numerical values have become characters!
[1] "7" "9" "2"
```

Indexing vectors

Indexing is when you want to refer to specific parts or values of a vector.

Usually we index with square brackets.

You can refer to the positions of the values by their number.

```
> emotions[1:3]
[1] 4 7 4
emotions[c(1,5)]   #concatenate  1 and 5
[1] 4 3
```

Indexing vectors

Indexing is when you want to refer to specific parts or values of a vector.

Usually we index with square brackets.

You can refer to the names of the values by their number, if they have names.

```
 names(emotions) <- c("Bill", "David", "Sara",
                       "Dan", "Josh", "Pat")
 emotions["Sara"]
Sara
   4
 emotions[c("Bill", "David")]
 Bill David
    4     7
```

Indexing vectors

Indexing is when you want to refer to specific parts or values of a vector.

Usually we index with square brackets.

You can use logical statements to select values that meet certain criteria.

```
 emotions[emotions > 6]
David   Dan   Pat
    7     7     8
```

Data frames

Data frames are lists of vectors which are related to one another (Think “spreadsheets”)

Features:

- Data frames have two dimension: rows and columns.
- (Usually) Columns represent variables.
- Every value in a column is the same class (numeric, character, etc)
- (Usually) Rows represent observations (people, mice, time points, etc).
- Values in rows can be different classes.
- The length of each vector must be the same.

Data frames

Because data frames are simply collections of vectors, you can create a data frame using vectors.

```
 data.example = data.frame(GENDER = genders,
+                           EMOTIONS = emotions)
 data.example
      GENDER EMOTIONS
Bill    male        4
David   male        7
Sara  female        4
Dan     male        7
Josh    male        3
Pat     male        8
```

Indexing data frames

We can use the same methods to select specific parts of data frames. The trick is data frames have two dimensions, not one. So we have to separate selecting rows from selecting columns.

Using numbers

Indexing a vector

```
 emotions[1:3]
[1] 4 7 4
```

Indexing a data frame

```
data.example[1:3, 1:2]
  GENDER EMOTIONS
1   male        4
2   male        7
3 female        4
```

Indexing data frames

We can use the same methods to select specific parts of data frames. The trick is data frames have two dimensions, not one. So we have to separate selecting rows from selecting columns. But, we can specify that we want all of either a row or column by leaving it blank

Indexing a data frame

```
> data.example[,1]    #give me the entire first column (as a vector)
[1] male   male   female male   male   male
Levels: female male

> data.example[,1,drop=FALSE] #give me the entire first column neatly
      GENDER
Bill    male
David   male
Sara  female
Dan     male
Josh    male
Pat     male
```

Indexing data frames

We can use the same methods to select specific parts of data frames. The trick is data frames have two dimensions, not one. So we have to separate selecting rows from selecting columns.

Using names

Indexing a vector

```
emotions[c("Josh","Pat")]
Josh  Pat
   3    8
```

Indexing a data frame

```
data.example[ , "GENDER"]   #refer  to the column by name
[1] male   male   female male   male   male
Levels: female male
```

Indexing data frames

We can use the same methods to select specific parts of data frames. The trick is data frames have two dimensions, not one. So we have to separate selecting rows from selecting columns.

Using logical statements

Indexing a vector

```
 emotions[emotions < 7]
Bill Sara Josh
   4    4    3
```

Indexing a data frame

```
 data.example[data.example$GENDER == "female", ]
    GENDER EMOTIONS
Sara female        4
```

We looked for equality by using the == operator (read as equals)

Indexing data frames

Data frames can also be indexed using the dollar sign $.

```
 data.example$EMOTIONS
[1] 4 7 4 7 3 8
```

This is read as “from the data frame called data.example, give me the variable called EMOTIONS.”

Other kinds of objects

Lists

- Like vectors, but each element can be *anything* (value, vector, data frame, another list)
- Output of analysis functions
- Can index using $
- Can index by name
- or, can index by [ ] for the name and content of the vector or [[ ]] for the contents

Matrices

- Like data frames but every value has to be the same class (character, numeric, logical)
- Useful for matrix algebra (i.e., lots of correlation and regression analyses)
- Operations are faster on matrices than data frames (for large data sets)

R is a language

R is an object-oriented programming language.

- Think of R like having a conversation with a specific person.
- They (R) have their own language, and you need to learn how to speak it.
- R is not very forgiving of mistakes, so pay attention to grammar and punctuation.

Translating R

```
catch(x = ball)
```

Nouns

Subject: R is the subject of every sentence.

Object: Objects are objects of the sentence!

Verbs

- Functions are the verbs of sentences.

```
catch(x = ball)
```

- Functions are always followed by parentheses.

```
mean(data.example$EMOTIONS)
[1] 5.5
```

- Functions can be nested. This is like a run-on sentence.

```
 round(mean(emotions))
[1] 6
```

Find the mean of the values in emotion, then round that number.

Adverbs and other modifiers

To be more specific or change the default way of doing something, specify arguments. These are like adverbs or clauses.

```
catch(x = ball, how = "smoothly",
      where = "beach",
      with = friends)
```

Arguments might be character values, numbers, more data, anything. The documentation (help) for a function will tell you what arguments are available to be changed and what values they can or should take.

```
help(t.test)  # or
?t.test

t.test(x = groupA, y = groupB, paired = T, mu = 5)
```

Punctuation

- Spaces – you can put as many spaces as you want between words and symbols, but not within them.

```
mean(        data) #ok
me      an( data) #not ok
```

- Parentheses – It's easy to forget one or put one in the wrong place when nesting.

```
round(x = mean(data, digits = 3) #this is wrong
round(x = mean(data), digits = 3) #this is ok
```

- Captialization – MATTERS

```
data != DATA != Data
```

The power of R: Objects can act upon objects

1. Every function returns an object.
   - This object can contain objects.
   - To see what is in an object use the str command to see the **str**ucture of an object.
2. Other functions can then act upon those objects to create objects
   - mean(), sd(), median() each return objects as values
   - describe() then packages those objects to return a general set of useful statistics.
3. It is this ability to use the output object from one function as the input to the next function that makes R so powerful.

Functions act upon the output of other functions

R code

```
m <- mean(sat.act$SATV,na.rm=TRUE)
s <- sd(sat.act$SATQ,na.rm=TRUE)
md <- median(sat.act[,3],na.rm=TRUE)
describe(sat.act)  #combines these prior three and more
```

```
describe(sat.act)  #combines these prior three functions and more
          vars   n   mean     sd median trimmed    mad min max range  skew kurtosis   se
gender       1 700   1.65   0.48      2    1.68   0.00   1   2     1 -0.61    -1.62 0.02
education    2 700   3.16   1.43      3    3.31   1.48   0   5     5 -0.68    -0.07 0.05
age          3 700  25.59   9.50     22   23.86   5.93  13  65    52  1.64     2.42 0.36
ACT          4 700  28.55   4.82     29   28.84   4.45   3  36    33 -0.66     0.53 0.18
SATV         5 700 612.23 112.90    620  619.45 118.61 200 800   600 -0.64     0.33 4.27
SATQ         6 687 610.22 115.64    620  617.25 118.61 200 800   600 -0.59    -0.02 4.41
```

Use str to see the structure of an object

R code

```
d <- describe(sat.act)  #form a new object
names(d)   #just the names of the objects
str(d)  #the detailed structure of those objects
d     #the objects organized in a pretty way for display
```

```
 d <- describe(sat.act)  #form a new object
> names(d)   #just the names of the objects
 [1] "vars"     "n"        "mean"     "sd"       "median"   "trimmed"  "mad"      "min"      
[10] "range"    "skew"     "kurtosis" "se"      
> str(d)  #the detailed structure of those objects
Classes ?psych?, ?describe? and 'data.frame':      6 obs. of  13 variables:
 $ vars    : int  1 2 3 4 5 6
 $ n       : num  700 700 700 700 700 687
 $ mean    : num  1.65 3.16 25.59 28.55 612.23 ...
 $ sd      : num  0.478 1.425 9.499 4.824 112.903 ...
 $ median  : num  2 3 22 29 620 620
 $ trimmed : num  1.68 3.31 23.86 28.84 619.45 ...
 $ mad     : num  0 1.48 5.93 4.45 118.61 ...
 $ min     : num  1 0 13 3 200 200
 $ max     : num  2 5 65 36 800 800
 $ range   : num  1 5 52 33 600 600
 $ skew    : num  -0.615 -0.681 1.643 -0.656 -0.644 ...
 $ kurtosis: num  -1.6247 -0.0749 2.4243 0.535 0.3252 ...
 $ se      : num  0.0181 0.0539 0.359 0.1823 4.2673 ...
> d     #the objects organized in a pretty way for display
          vars   n   mean     sd median trimmed    mad min max range  skew kurtosis   se
gender       1 700   1.65   0.48      2    1.68   0.00   1   2     1 -0.61    -1.62 0.02
education    2 700   3.16   1.43      3    3.31   1.48   0   5     5 -0.68    -0.07 0.05
age          3 700  25.59   9.50     22   23.86   5.93  13  65    52  1.64     2.42 0.36
ACT          4 700  28.55   4.82     29   28.84   4.45   3  36    33 -0.66     0.53 0.18
```

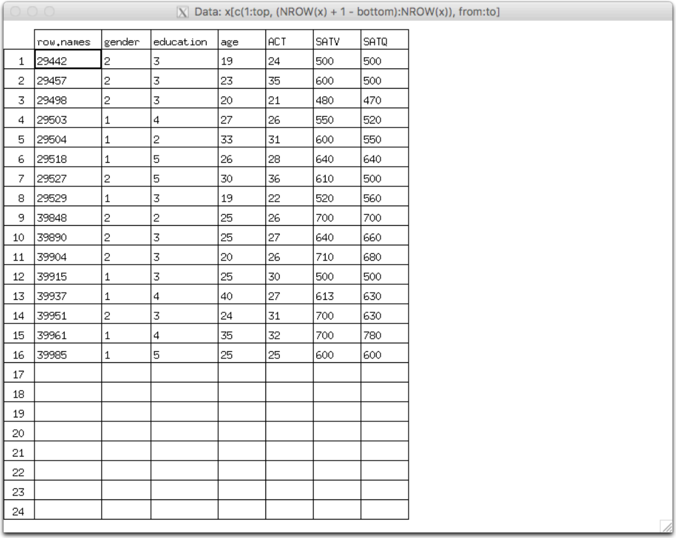
Several ways to see the contents of an object

R code

```
headTail(sat.act) #shows the first and last
                   n rows of the data frame
quickView(sat.act) #opens a window showing the first and last n rows
                  of the data frame (scrollable)
view(sat.act)  #opens a window to show all the rows and
                   columns of the data frame (scrollable)
```

```
      gender education age ACT SATV SATQ
29442      2         3  19  24  500  500
29457      2         3  23  35  600  500
29498      2         3  20  21  480  470
29503      1         4  27  26  550  520
...      ...       ... ... ...  ...  ...
39937      1         4  40  27  613  630
39951      2         3  24  31  700  630
39961      1         4  35  32  700  780
39985      1         5  25  25  600  600
```

Objects act on objects



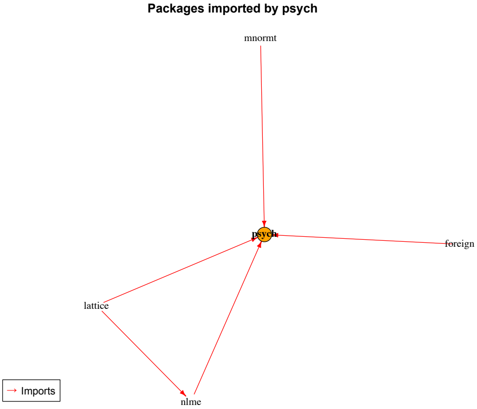
Data: x[c(1:top, (NROW(x) + 1 - bottom):NROW(x)), from:to]

	row.names	gender	education	age	ACT	SATV	SATQ
1	29442	2	3	19	24	500	500
2	29457	2	3	23	35	600	500
3	29498	2	3	20	21	480	470
4	29503	1	4	27	26	550	520
5	29504	1	2	33	31	600	550
6	29518	1	5	26	28	640	640
7	29527	2	5	30	36	610	500
8	29529	1	3	19	22	520	560
9	39848	2	2	25	26	700	700
10	39890	2	3	25	27	640	660
11	39904	2	3	20	26	710	680
12	39915	1	3	25	30	500	500
13	39937	1	4	40	27	613	630
14	39951	2	3	24	31	700	630
15	39961	1	4	35	32	700	780
16	39985	1	5	25	25	600	600
17							
18							
19							
20							
21							
22							
23							
24							

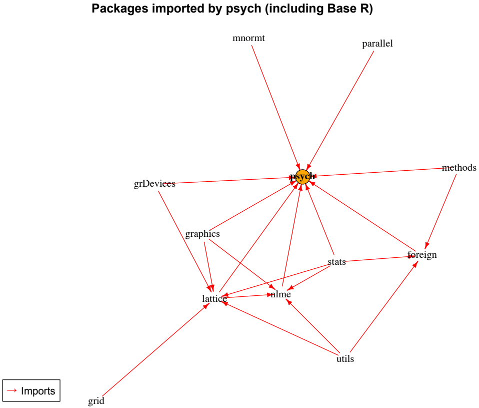
Packages extend the power of R

1. Just as functions can take the output from another function, so can packages build upon other packages.
2. Core packages come with the R installation
   - *base*-R includes 1218 different functions and then also loads in 5-8 other core packages:
   - e.g., *stats* includes 447 functions (commands) that do most of those basic statistics not done by base;
   - *foreign* handles different input and output formats from "foreign" languages (e.g., SPSS)
3. The Comprehensive R Archive Network (CRAN) is the repository for the other 10,700 packages that people have contributed
4. Most of these packages depend, in turn, on other packages. They all depend upon core-R.

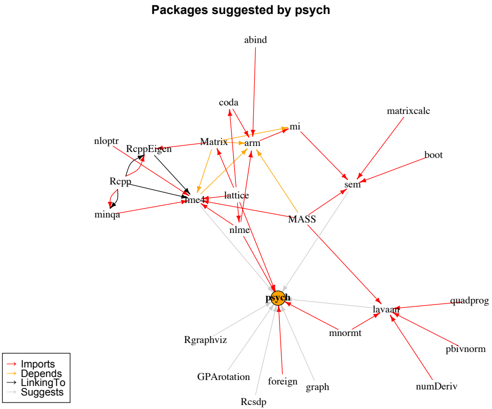
Dependencies of the psych package



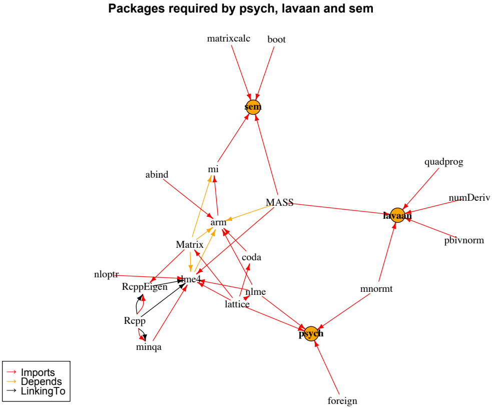
Dependencies of the psych package including base R



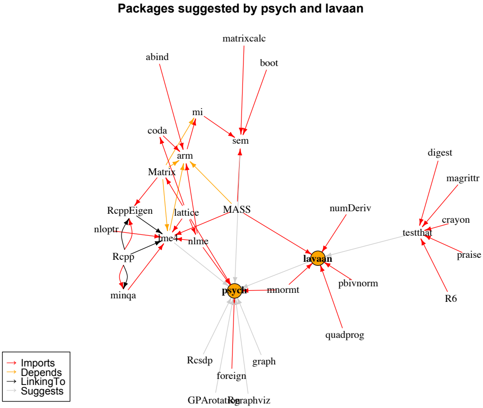
Packages can “suggest” other useful packages which in turn “require” other packages



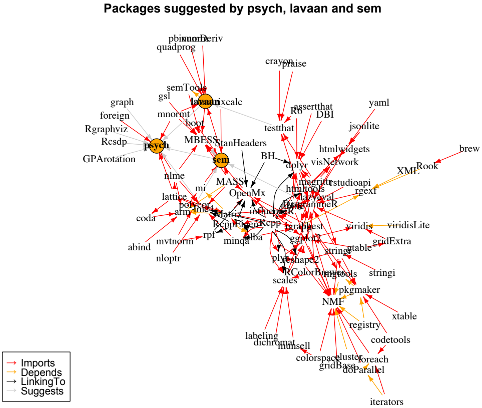
psych, lavaan and sem require other useful packages



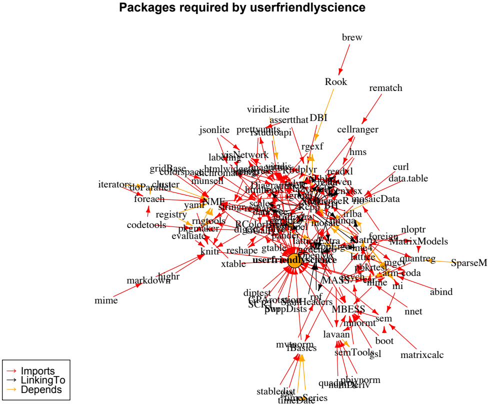
psych and lavaan suggest other useful packages



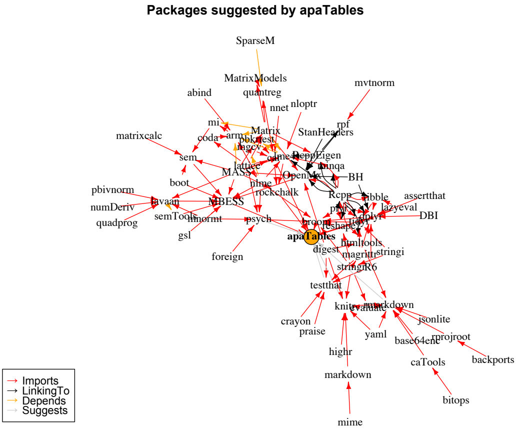
psych, lavaan and sem suggest other useful packages



Some packages require many others to be helpful wrapper packages (e.g. userfriendlyscience)



apatables require many others to be a helpful wrapper



Bromley, A. G. (1982). Charles Babbage's Analytical Engine, 1838. *IEEE annals of the history of computing*, *4*(3), 196–217.

Isaacson, W. (2014). *The Innovators: How a Group of Inventors, Hackers, Geniuses and Geeks Created the Digital Revolution*. Simon and Schuster.

Lovelace, A. A. (1842). Sketch of the analytical engine invented by Charles Babbage, by LF Menabrea, officer of the military engineers, with notes upon the memoir by the translator. *Taylor's Scientific Memoirs*, *3*, 666–731.