An introduction to R
Sponsored by
The Association of Psychological Science
and
Society of Multivariate Experimental Psychology

William Revelle, David M. Condon & Sara Weston
Northwestern University
Evanston, Illinois USA

https://personality-project.org/r/aps/aps-short.pdf
https://personality-project.org/r/aps/aps.Rmd

Partially supported by a grant from the National Science Foundation: SMA-1419324

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

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

Package Dependencies
- Objects act on objects
- Package dependencies

-> Part II

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 is a group project run by a core group of developers (with new releases semiannually). The current version of R is 3.5.0
- 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:

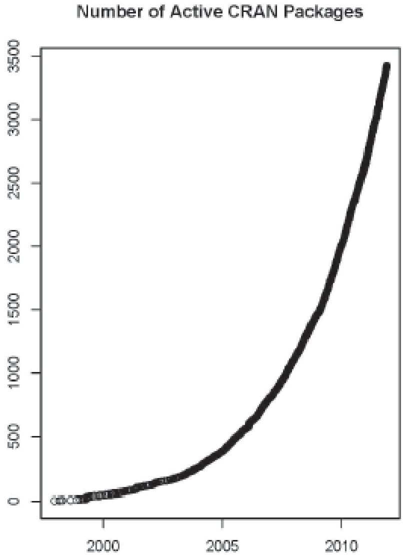
1. an effective data handling and storage facility,
2. a suite of operators for calculations on arrays, in particular matrices,
3. a large, coherent, integrated collection of intermediate tools for data analysis,
4. graphical facilities for data analysis and display either on-screen or on hardcopy, and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

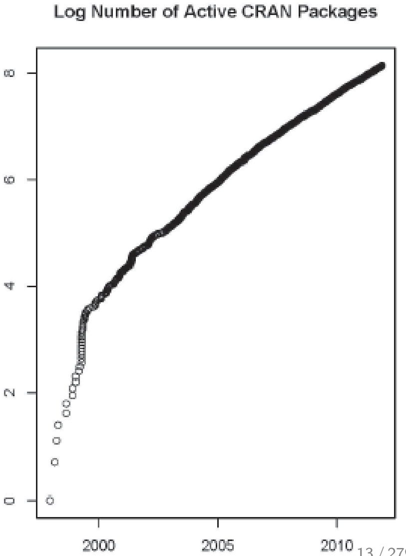
"Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages ... available through the CRAN family of Internet sites covering a very wide range of modern statistics." (Adapted from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

R: A brief history

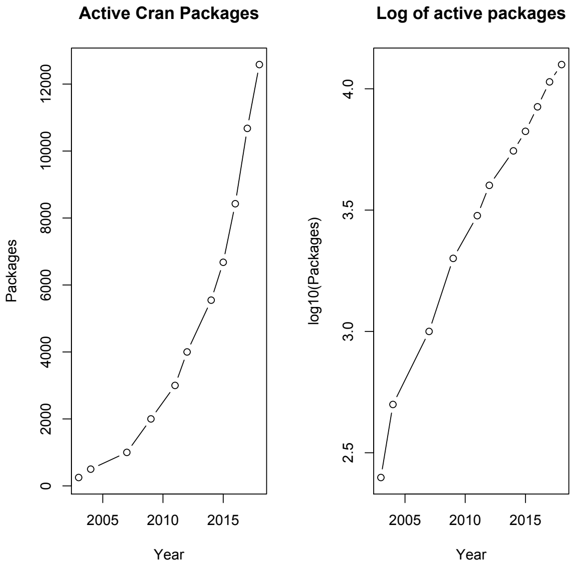
- 1991-93: Ross Dhaka and Robert Gentleman begin work on R project for Macs at U. Auckland (S for Macs).
- 1995: R available by ftp under the General Public License.
- 96-97: mailing list and R core group is formed.
- 2000: John Chambers, designer of S joins the Rcore (wins a prize for best software from ACM for S)
- 2001-2018: 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
  - 2017-05-21 10,677 packages (on CRAN) + 1,383 bioinformatic packages + ?,000s on GitHub
  - 2018-05-20 12,583 packages (on CRAN) + 1,560 bioinformatic packages + ?,000s on GitHub

Rapid and consistent growth in packages contributed to R

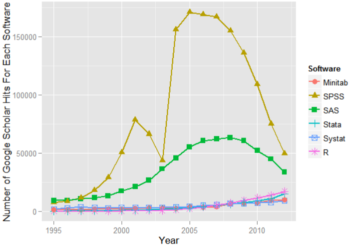


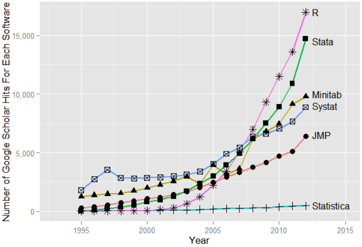


Rapid and consistent growth in packages contributed to R



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 ($> 117, K$ citations, $\approx 32K$ in 2017, 16K 2018)
4. Google Page rank
5. Number of downloads (see http://www.rpackages.io/packages or

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

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

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
- Download R for (Mac) OS X
- 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 (2018-04-23, Joy in Playing) R-3.5.0.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

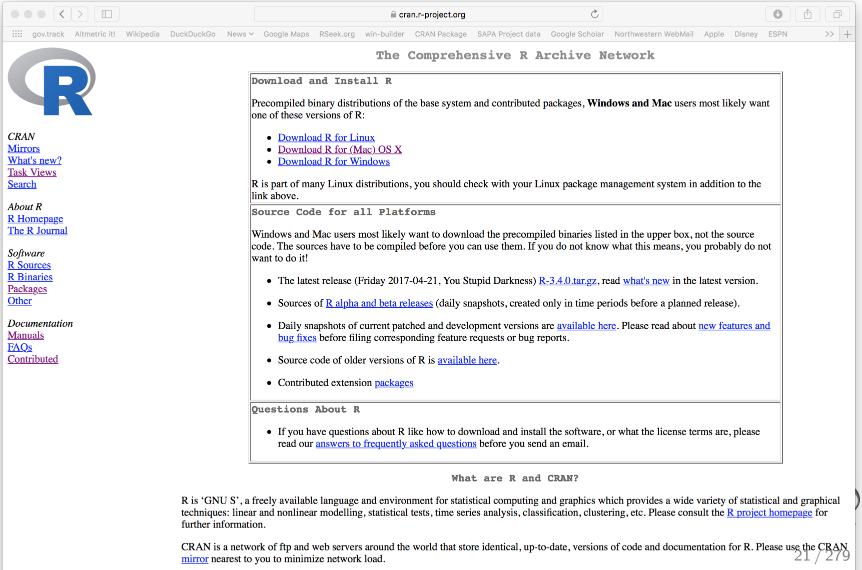
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

Go to the Comprehensive R Archive Network (CRAN)



Download and install the appropriate version – PC

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.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 < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. 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 Uwe Ligges directly in case o 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 – PC

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R-3.5.0 for Windows (32/64 bit)

Download R 3.5.0 for Windows (62 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

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. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

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

Other builds

- 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 availabl 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.htm.

Download and install the appropriate version – Mac

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

As of 2016/03/01 package binaries for R versions older than 2.12.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting accordingly.

R 3.5.0 "Joy in Playing" released on 2018/04/24

Important: since R 3.4.0 release we are now providing binaries for OS X 10.11 (El Capitan) and higher using non-Apple toolkit to provide support for OpenMP and C++17 standard features. To compile packages you may have to download tools from the tools directory and read the corresponding note below.

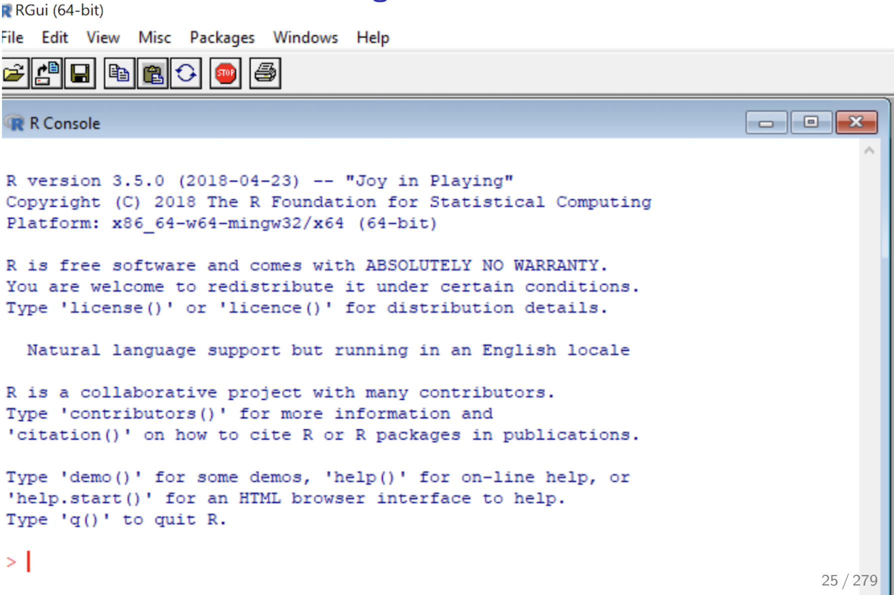
Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`md5 R-3.5.0.pkg`
in the *Terminal* application to print the MD5 checksum for the R-3.5.0.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-3.5.0.pkg`

Lastest release:

R-3.5.0.pkg
MD5-hash: 414029c9c9f706d3d04baa887ccffbc4
SHA1-hash: 6e90d38892bb366630ae30c223a898e8af84dff7
(ca. 74MB)

R 3.5.0 binary for OS X 10.11 (El Capitan) and higher, signed package. Contains R 3.5.0 framework, R.app GUI 1.70 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 5.2. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to

Starting R on a PC



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

```
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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 1.70 (7521) x86_64-apple-darwin15.6.0]
[Workspace restored from /Users/wr/.RData]
[History restored from /Users/wr/.Rapp.history]
Good morning Bill.
Are you ready to have fun?
```

Check the version number for R $\geq$ 3.5.0) and for psych ($\geq$1.8.4

R code

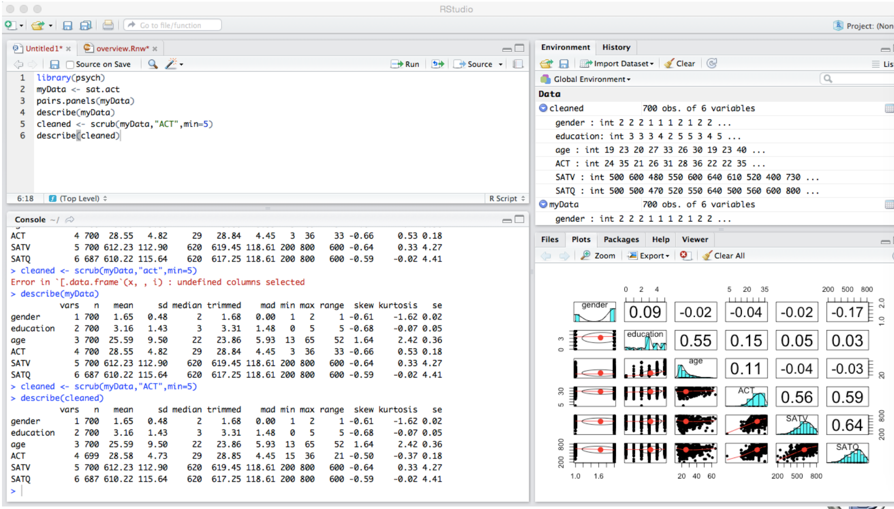
```
sessionInfo()
```

```
R version 3.5.0 (2018-04-23)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS High Sierra 10.13.4

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib

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

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

other attached packages:
[1] psych_1.8.4

loaded via a namespace (and not attached):
[1] compiler_3.5.0  tools_3.5.0     parallel_3.5.0  foreign_0.8-70  nlme_3.1-137    mnormt_1.5
[7] grid_3.5.0      lattice_0.20-35
other attached packages:
```

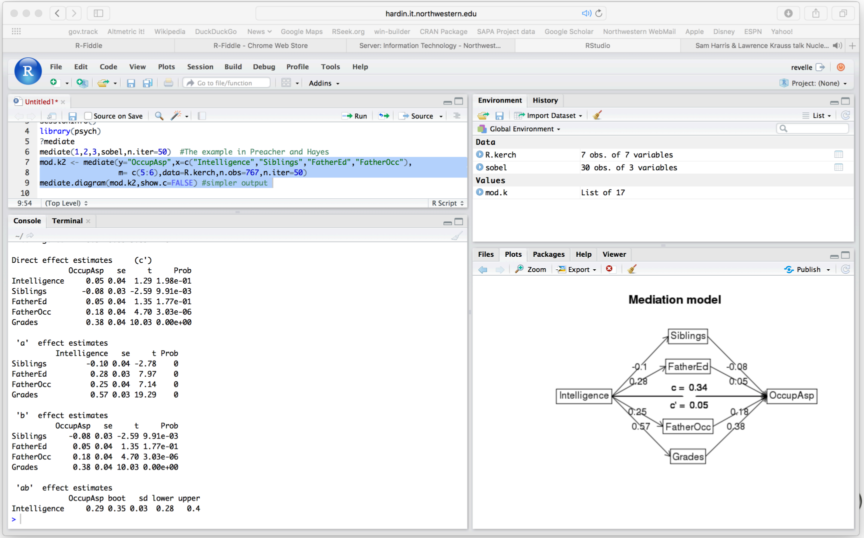
Various ways to run R

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) (recommended by David)
   - allows for multiple cores for parallel processing
4. From the web
   - allows remote R (but R = 3.4 and psych = 1.7.8)
   - Rdocumentation is helpful for package searcjh

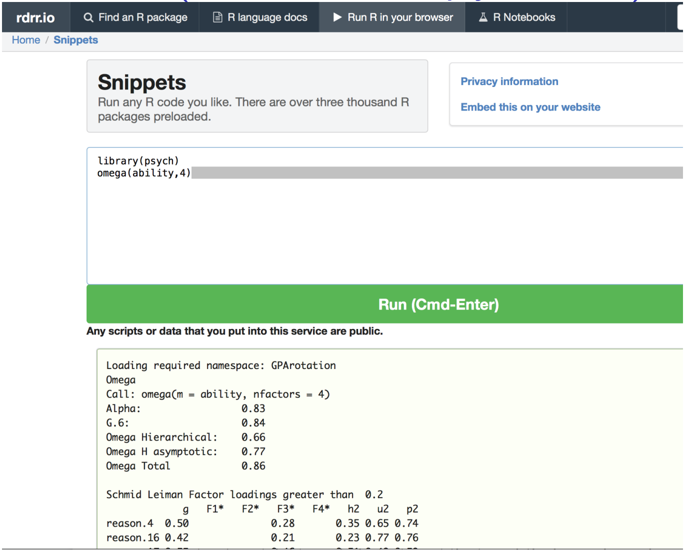
R Studio is a useful “Integrated Development Environment” (IDE)



R Studio may be run on a remote server



https://rdrr.io allows to run on a remote server (but R = 3.4.0 and psych = 1.7.8)



R is extensible: The use of "packages"

1. More than 12,583 packages are available for at CRAN (and growing daily. It was 10,677 last year and 8,427 two 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
  - Zelig
- Special use
  - ltm/eRm/mirt
  - sem
  - lavaan/OpenMx
  - GPArotation
  - mvtnorm
  - $>$ 15,180 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
  - General purpose toolkit
- 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
  - devtools
  - readxl
  - foreign
  - RMySQL
  - readr
  - rio
- Special use
  - plyr & dplyr
  - data.table
  - knitr
  - sweave
  - ggplot2
  - $> 12{,}500$
  - $+$?

- General applications
  - Development tools from GitHub
  - input from excel
  - input from SPSS, , etc. (part of Core)
  - input from MySQL
  - fast input for very large csv files
  - simple to use integrated input/output
- More specialized packages
  - reshape from wide to long etc.
  - faster data handling for large data sets
  - integrate markdown documentation with R
  - integrate LaTeXdocumentation with R
  - powerful grammar of graphics
  - 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 3.5.0) (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 >

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

- Just install one package (e.g., psych) You might have to choose a "mirror" site.

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

- as well as a few suggested packages that add functionality for factor rotation, multivariate normal distributions, etc.

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

- Install the task view installer package.

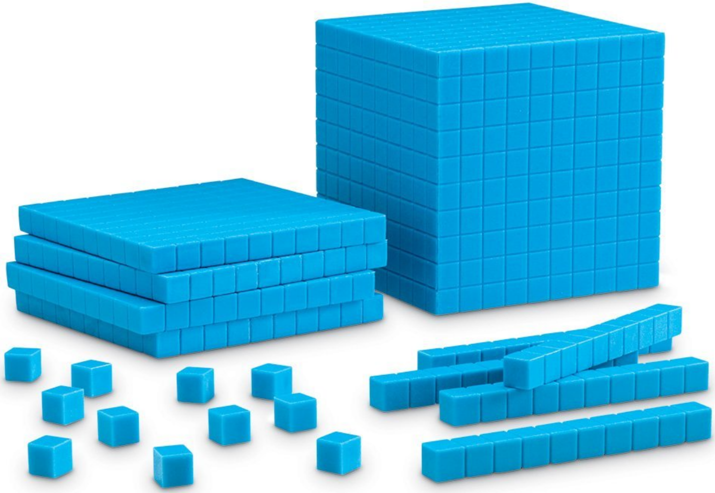
```
> library(ctv)
```

- Make it active

```
> install.views("Psychometrics")
```

- If you want to install all the packages in the "Psychometrics" task view.

Building Blocks



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"
  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"
```

Vectors and character strings

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")
#this uses the c()  function for concatenation,
#and we need to delimitate  each element with " "
#alternatively, use the cs() function which takes C
genders <- cs(male,male,female,male,male,male)
#show this
genders
 "male"   "male"   "female" "male"   "male"   "male
emotions <-  c(4, 7, happy, 7, 3, 8)
```

(We use the c for the *concatenate* operator) or the cs for the *character string* operator.

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

Try it (example 2)

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

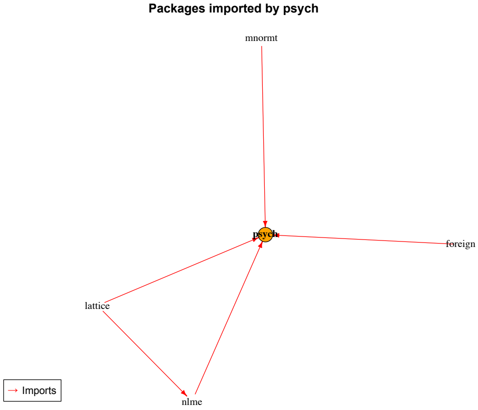
Data: x[c(1:top, (NROW(x) + 1 - bottom):NROW(x)), from:to]

	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							

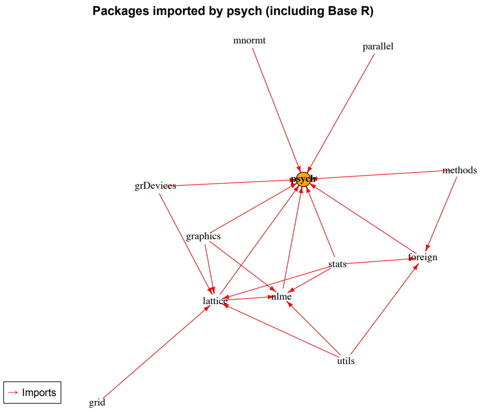
Packages extend the power of R

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 1220 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 12,560 packages that people have contributed
4. Most of these packages depend, in turn, on other packages. They all depend upon core-R.

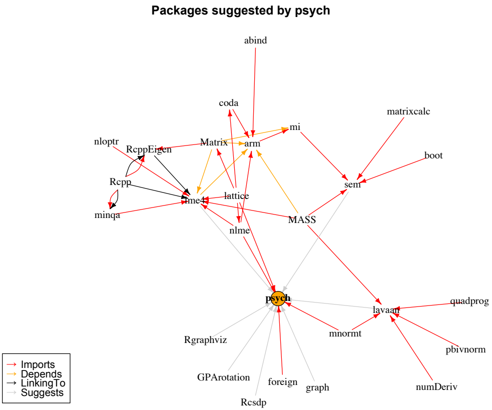
Dependencies of the psych package



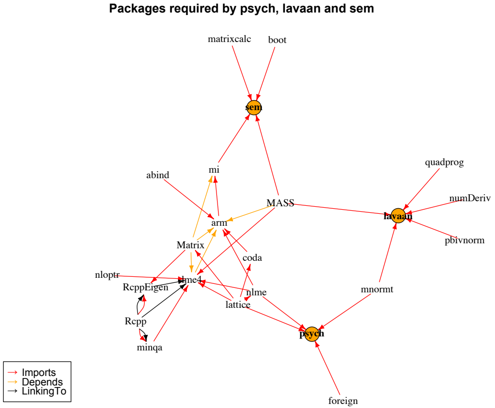
Dependencies of the psych package including base R



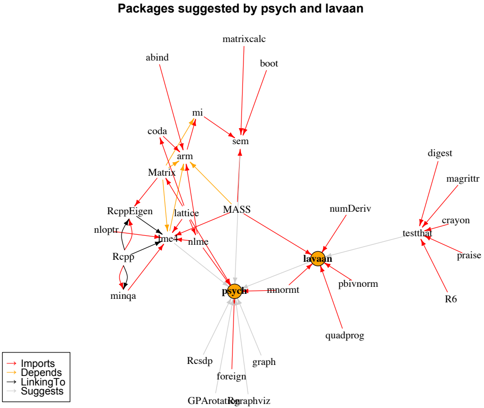
Packages can "suggest" other useful packages which in turn "require" other packages



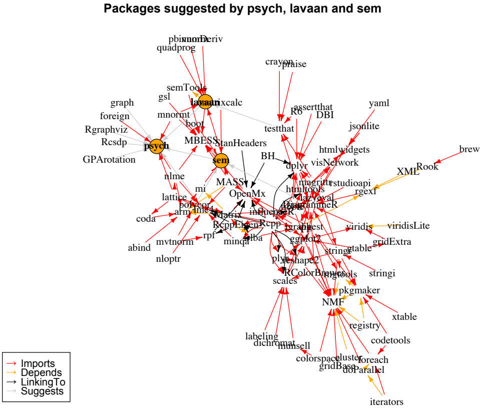
psych, lavaan and sem require other useful packages



psych and lavaan suggest other useful packages

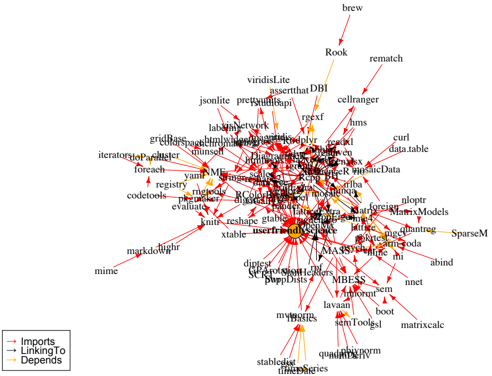


psych, lavaan and sem suggest other useful packages

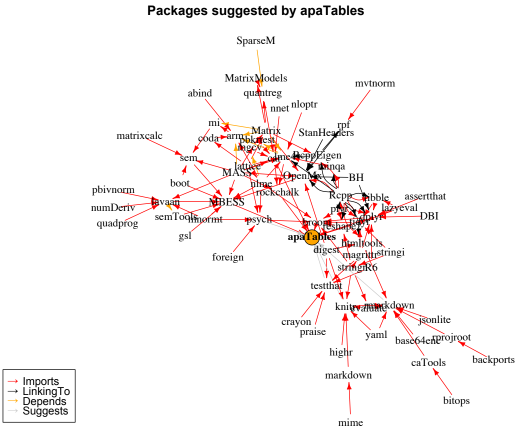


Some packages require many others to be helpful wrapper packages (e.g. userfriendlyscience)

Packages required by userfriendlyscience



apatables require many others to be a helpful wrapper



Questions?

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 II

-> Part I: What is R

Basic R: A brief example
- Basic R capabilities: Calculation, Statistical tables
- Basic Graphics

A brief example of exploratory and confirmatory data analysis
- Data preparation, descriptive statistics, data cleaning, correlation plots: (Examples part ii)
- Inferential statistics

Multiple regression modeling and graphics

-> Part III: Basic statistics and graphics

Basic R commands – remember don't enter the >

R is just a fancy calculator. Add, subtract, sum, products, group

```
> 2 + 2        #sum two numbers
[1] 4      #show the output
> 3^4     #3 raised to the 4th
[1] 81     #that was easy
> sum(1:10)  #find the sum of the first 10 numbers
[1] 55   #the answer
> prod(c(1, 2, 3, 5, 7)) #the product of the concatenated (c) numbers
[1] 210  #Note how we combined product with concatenate
```

It is also a statistics table (the normal distribution, the t, the F, the χ^2 distribution, the xyz distribution)

```
> pnorm(q = 1)  #the probability of a normal with value of 1 sd
[1] 0.8413447  #
> pt(q = 2, df = 20) #what about the probability of a t-test value of
[1] 0.9703672  #this is the upper tail
```

R is a set of distributions. Don't buy a stats book with tables!

Table: To obtain the density, prefix with *d*, probability with *p*, quantiles with *q* and to generate random values with *r*. (e.g., the normal distribution may be chosen by using dnorm, pnorm, qnorm, or rnorm.) Each function can be modified with various parameters.

Distribution	base name	P 1	P 2	P 3	example application
Normal	norm	mean	sigma		Most data
Multivariate normal	mvnorm	mean	r	sigma	Most data
Log Normal	lnorm	log mean	log sigma		income or reaction time
Uniform	unif	min	max		rectangular distributions
Binomial	binom	size	prob		Bernuilli trials (e.g. coin flips)
Student's t	t	df		nc	Finding significance of a t-test
Multivariate t	mvt	df	corr	nc	Multivariate applications
Fisher's F	f	df1	df2	nc	Testing for significance of F test
χ^2	chisq	df		nc	Testing for significance of χ^2
Exponential	exp	rate			Exponential decay
Gamma	gamma	shape	rate	scale	distribution theoryh
Hypergeometric	hyper	m	n	k	
Logistic	logis	location	scale		Item Response Theory
Poisson	pois	lambda			Count data
Weibull	weibull	shape	scale		Reaction time distributions

An example of using r, p, and q for a distributions

R code

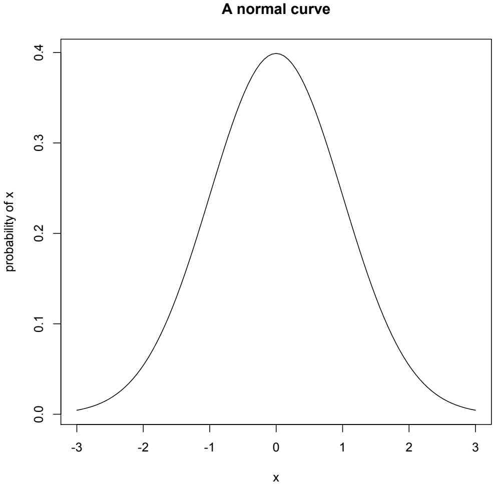
```
set.seed(42) #set the random seed to get the same sequence
x <- rnorm(5) #find 5 randomly distributed normals
round(x,2) #show them, rounded to 2 decimals
round(pnorm(x),2) #show their probabilities to 2 decimals
round(qnorm(pnorm(x)),2)  #find the quantiles of the normal
```

Produces this output

```
> set.seed(42) #set the random seed to get the same sequence
> x <- rnorm(5) #find 5 randomly distributed normals
> round(x,2) #show them, rounded to 2 decimals
[1]  1.37 -0.56  0.36  0.63  0.40
> round(pnorm(x),2) #show their probabilities to 2 decimals
[1] 0.91 0.29 0.64 0.74 0.66
> round(qnorm(pnorm(x)),2)  #find the quantiles of the normal
[1]  1.37 -0.56  0.36  0.63  0.40
```

See (Example 2)

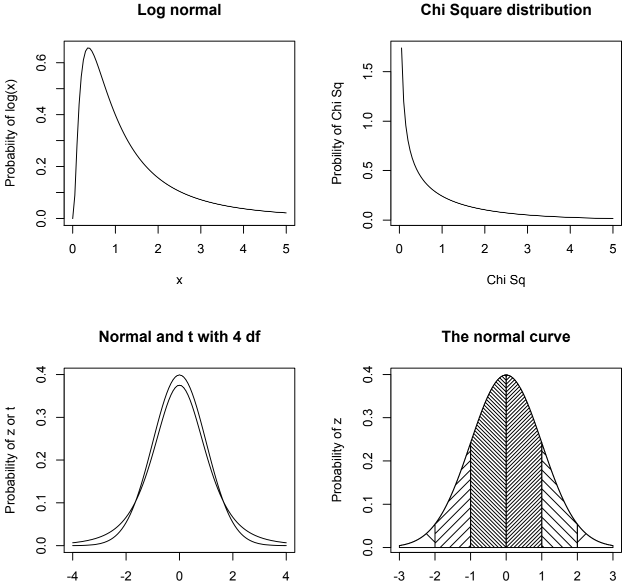
R can draw distributions



We do this by using the `curve` function to which we pass the values of the `dnorm` function.
curve(dnormal(x),-3,3, ylab="probability of x",main="A normal curve")

(Example 3)

R can draw more interesting distributions



R is also a graphics calculator

R code

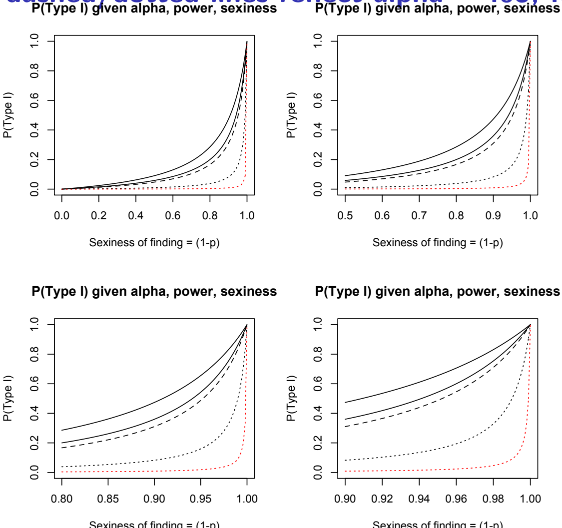
```
op <- par(mfrow=c(2,2))       #set up a 2 x 2 graph
curve(dlnorm(x),0,5,ylab='Probabiity of log(x)',main='Log normal')
curve(dchisq(x,1),0,5,ylab='Probility of Chi Sq',xlab='Chi Sq',main='Chi Square distribution
curve(dnorm(x),-4,4,ylab='Probability of z or t',xlab='z or t',main='Normal and t with 4 df')
curve(dt(x,4),add=TRUE)
#
#somewhat more complicated
#first draw the normal curve
curve(dnorm(x),-3,3,xlab="",ylab="Probability of z") #the range of x
title(main="The normal curve",outer=FALSE) #the title
#add the cross hatching by using polygons
xvals <-  seq(-3,-2,length=100)  #From -3 to 2 with 100 points
dvals <- dnorm(xvals)
polygon(c(xvals,rev(xvals)),c(rep(0,100),rev(dvals)),density=2,angle=-45)
xvals <-  seq(-2,-1,length=100)
dvals <- dnorm(xvals)
polygon(c(xvals,rev(xvals)),c(rep(0,100),rev(dvals)),density=14,angle=45)
xvals <-  seq(-1,-0,length=100)
dvals <- dnorm(xvals)
polygon(c(xvals,rev(xvals)),c(rep(0,100),rev(dvals)),density=34,angle=-45)
xvals <-  seq(2,3,length=100)
dvals <- dnorm(xvals)
polygon(c(xvals,rev(xvals)),c(rep(0,100),rev(dvals)),density=2,angle=45)
xvals <-  seq(1,2,length=100)
dvals <- dnorm(xvals)
polygon(c(xvals,rev(xvals)),c(rep(0,100),rev(dvals)),density=14,angle=-45)
xvals <-  seq(0,1,length=100)
dvals <- dnorm(xvals)
polygon(c(xvals,rev(xvals)),c(rep(0,100),rev(dvals)),density=34,angle=45)
op <- par(mfrow=c(1,1)) #back to a normal 1 x 1 graph
```

R can help teach with 100s of example data sets.

```
> data()

> data(package="psych")
#see the names of the 56 data sets

> data(Titanic)
> ? Titanic

> data(cushny)
> ? cushny

> data(UCBAdmissions)
> ? UCBAdmissions
```

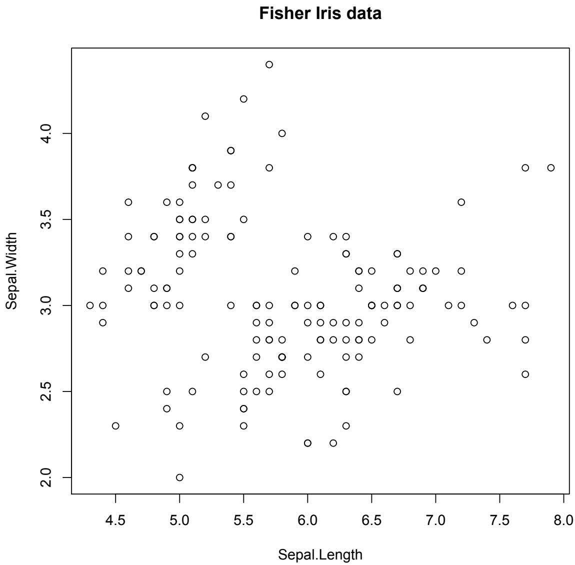
1. This opens up a separate text window and lists all of the data sets in the currently loaded packages.
2. Show the data sets available in a particular package (e.g., *psych*).
3. Gets the particular data set with its help file (e.g., the survival rates on the Titanic cross classified by age, gender and class).
4. Another original data set used by "student" (Gossett) for the t-test.
5. The UC Berkeley example of "sex discrimination" as a Simpson paradox

R can show current statistical concepts: Type I Errors: It is not the power, it is the prior likelihood dashed/dotted lines reflect alpha = .05, .01, .001 with power = 1



1. Extreme claims require extreme probabilities
2. Given that a finding is "significant", what is the likelihood that it is a Type I error?
3. Depends upon the prior likelihood (the 'sexiness') of the claim.

A simple scatter plot using `plot` with Fisher's Iris data set.



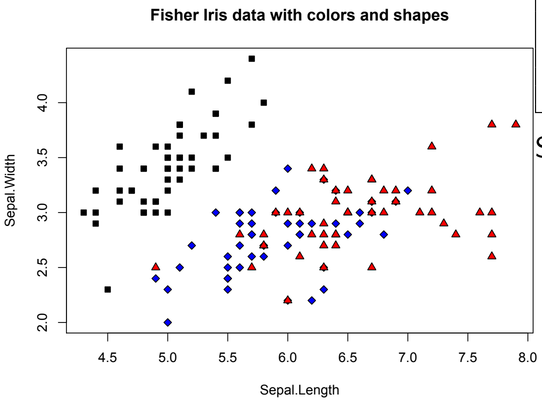
R code

```
plot(iris[1:2],xlab="Sepal.Length",
  ylab="Sepal.Width"
,main="Fisher Iris data")
```

Set parameters

1. xlab for x axis label
2. ylab for y axis label
3. main for title
4. (Example 4)

A simple scatter plot using `plot` with some colors and shapes



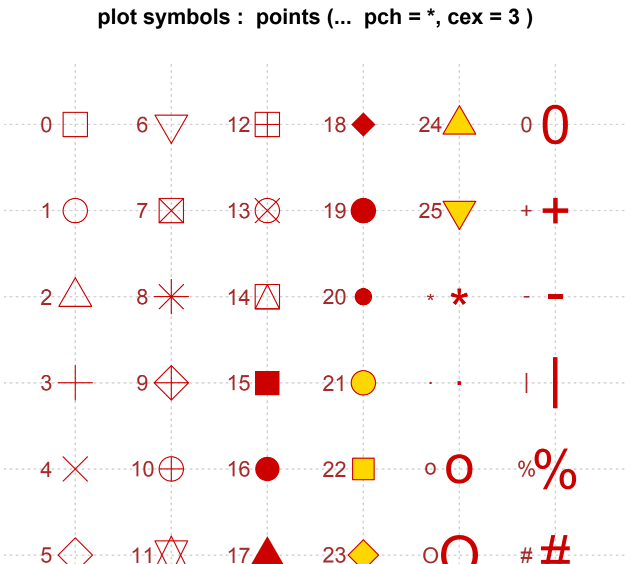
R code

```
plot(iris[1:2],xlab="Sepal.Length",
ylab="Sepal.Width" ,
main="Fisher Iris data with
colors and shapes",
 bg=c("black","blue",
"red")[iris[,"Species"]],
 pch=21+ as.numeric(iris[,5]))
```

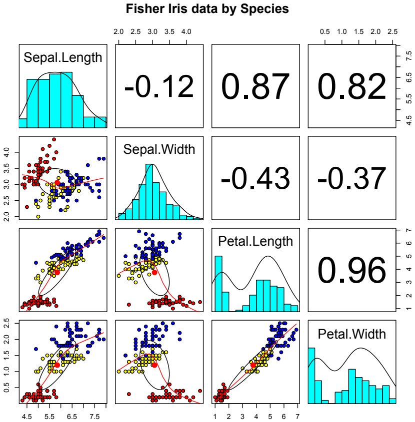
Set parameters

1. bg for background colors of symbols
2. pch chooses the plot character
3. Note how these depend upon iris[,5] which is the species

Show the various graphic options for plot character (pch)



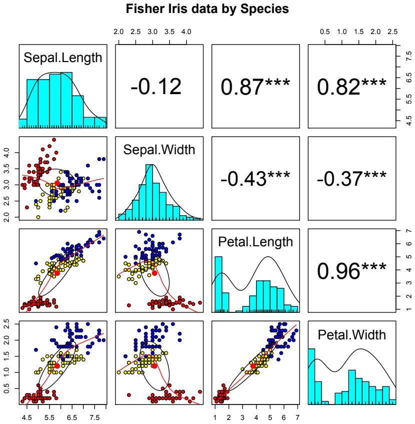
A scatter plot matrix plot with loess regressions using `pairs.panels`



1. Correlations above the diagonal
2. Diagonal shows histograms and densities
3. scatter plots below the diagonal with correlation ellipse
4. locally smoothed (loess) regressions for each pair
5. optional color coding of grouping variables.

```
pairs.panels(iris[1:4],bg=c("red","yellow","blue")
[iris$Species],pch=21,main="Fisher Iris data by
Species")
```

A scatter plot matrix plot with loess regressions using `pairs.panels`



Show "significance" using magic asterisks

```
pairs.panels(iris[1:4],bg=c("red","yellow","blue")
[iris$Species],pch=21,main="Fisher Iris data by
Species",stars=TRUE)
```

A brief example with real data - example 5

1. Get the data
2. Descriptive statistics
   - Graphic
   - Numerical
3. Inferential statistics using the linear model
   - regressions
4. More graphic displays

Get the data and describe it

1. First read the data, either from a built in data set, a local file, a remote file, or from the clipboard.
2. Describe the data using the describe function from *psych*

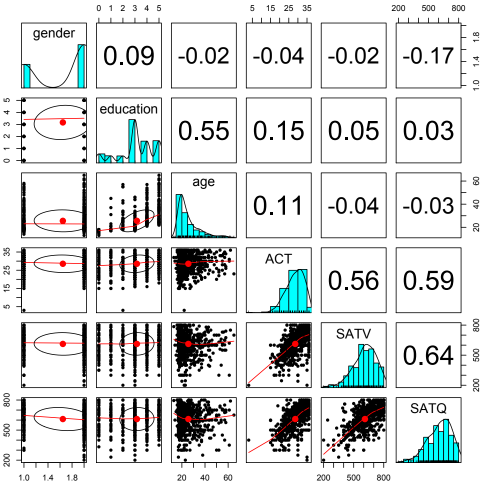
R code

```
my.data <- sat.act  #an example data file that is part of psych
#or
#my.data <-read.file()   #look for it on your hard drive
#or
file.name <-"http://personality-project.org/r/aps/sat.act.txt"
#now read it either locally or remotely
 my.data <- read.file(file.name)
#or if you have copied the data to the clipboard
# my.data <- read.clipboard()  #you can read it from there
describe(my.data) #report basic descriptive statistics
```

```
          var   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    -0.06 0.05
age         3 700  25.59   9.50     22   23.86   5.93  13  65    52  1.64     2.47 0.36
ACT         4 700  28.55   4.82     29   28.84   4.45   3  36    33 -0.66     0.56 0.18
SATV        5 700 612.23 112.90    620  619.45 118.61 200 800   600 -0.64     0.35 4.27
SATQ        6 687 610.22 115.64    620  617.25 118.61 200 800   600 -0.59     0.00 4.41
```

Graphic display of data using `pairs.panels`

pairs.panels(my.data) #Note the outlier for ACT



Clean up the data using `scrub`. Use ?scrub for help on the parameters.

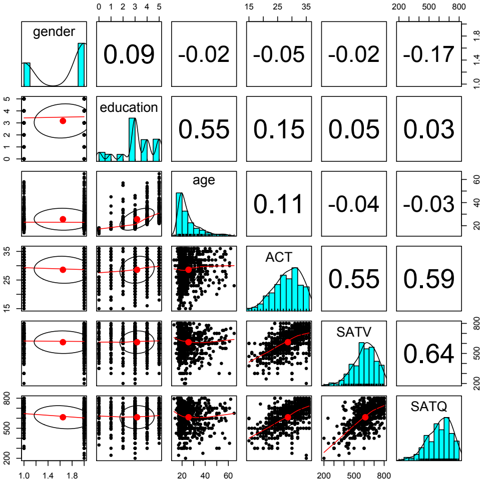
We noticed an outlier in the ACT data in the previous graph (you always graph your data, don't you).
We also noticed that the minimum value for ACT was unlikely (of course, you always describe your data).
So we change any case below 4 on the ACT to be missing (NA).

R code

```
cleaned <- scrub(my.data,"ACT",min=4)  #what data set,
              #which variable, what value to fix
describe(cleaned)   #look at the data again
pairs.panels(cleaned)
```

	var	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	-0.06	0.05
age	3	700	25.59	9.50	22	23.86	5.93	13	65	52	1.64	2.47	0.36
ACT	4	699	28.58	4.73	29	28.85	4.45	15	36	21	-0.50	-0.36	0.18
SATV	5	700	612.23	112.90	620	619.45	118.61	200	800	600	-0.64	0.35	4.27
SATQ	6	687	610.22	115.64	620	617.25	118.61	200	800	600	-0.59	0.00	4.41

Graphic display of cleaned data using `pairs.panels`



Find the pairwise correlations, round to 2 decimals

This also shows how two functions can be nested. We are rounding the output of the cor function.

R code

```
#specify all the parameters being passed
round(cor(x=sat.act,use="pairwise"),digits=2)
#the short way to specify the rounding parameter
round(cor(cleaned,use="pairwise"),2)
```

```
          gender education   age   ACT  SATV  SATQ
gender      1.00      0.09 -0.02 -0.05 -0.02 -0.17
education   0.09      1.00  0.55  0.15  0.05  0.03
age        -0.02      0.55  1.00  0.11 -0.04 -0.03
ACT        -0.05      0.15  0.11  1.00  0.55  0.59
SATV       -0.02      0.05 -0.04  0.55  1.00  0.64
SATQ       -0.17      0.03 -0.03  0.59  0.64  1.00
```

Display it differently using the lowerCor function

Operations that are done a lot may be made into your own functions. Thus, `lowerCor` finds the pairwise correlations, rounds to 2 decimals, displays the lower half of the correlation matrix, and then abbreviates the column labels to make them line up nicely

R code
```
lowerCor(cleaned)
```

```
          gendr edctn age   ACT   SATV  SATQ
gender     1.00
education  0.09  1.00
age       -0.02  0.55  1.00
ACT       -0.05  0.15  0.11  1.00
SATV      -0.02  0.05 -0.04  0.55  1.00
SATQ      -0.17  0.03 -0.03  0.59  0.64  1.00
```

Testing the significance of one correlation using `cor.test`.

R code
```
cor.test(my.data$ACT,my.data$SATQ)
```

```
        Pearson's product-moment correlation

data:  my.data$ACT and my.data$SATQ
t = 18.9822, df = 685, p-value < 2.2e-16
alternative hypothesis: true correlation
                is not equal to 0
95 percent confidence interval:
 0.5358435 0.6340672
sample estimates:
      cor
0.5871122
```

1. Specify the variables to correlate
2. Various statistics associated with the correlation.
3. But what if you want to do many tests? Use `corr.test`

Test many correlations for significance using `corr.test`

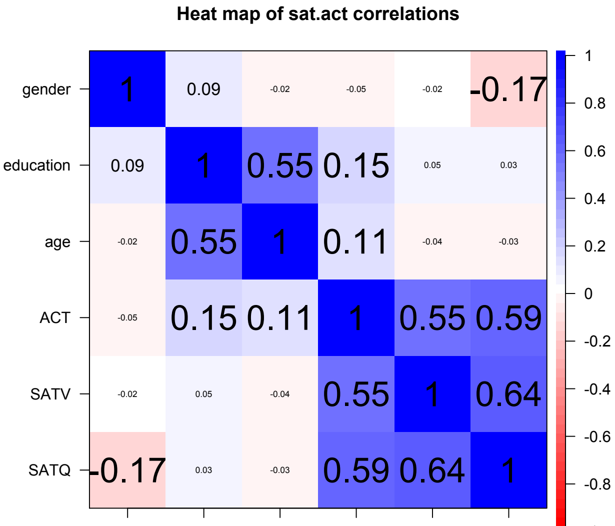
R code

```
corr.test(cleaned)
```

```
all:corr.test(x = cleaned)
Correlation matrix
          gender education   age   ACT  SATV  SATQ
gender      1.00      0.09 -0.02 -0.05 -0.02 -0.17
education   0.09      1.00  0.55  0.15  0.05  0.03
age        -0.02      0.55  1.00  0.11 -0.04 -0.03
ACT        -0.05      0.15  0.11  1.00  0.55  0.59
SATV       -0.02      0.05 -0.04  0.55  1.00  0.64
SATQ       -0.17      0.03 -0.03  0.59  0.64  1.00
Sample Size
          gender education age ACT SATV SATQ
gender       700       700 700 699  700  687
...
SATQ         687       687 687 686  687  687
Probability values (Entries above the diagonal are
                          adjusted for multiple tests.)
          gender education  age  ACT SATV SATQ
gender      0.00      0.17 1.00 1.00    1    0
education   0.02      0.00 0.00 0.00    1    1
age         0.58      0.00 0.00 0.03    1    1
ACT         0.21      0.00 0.00 0.00    0    0
```

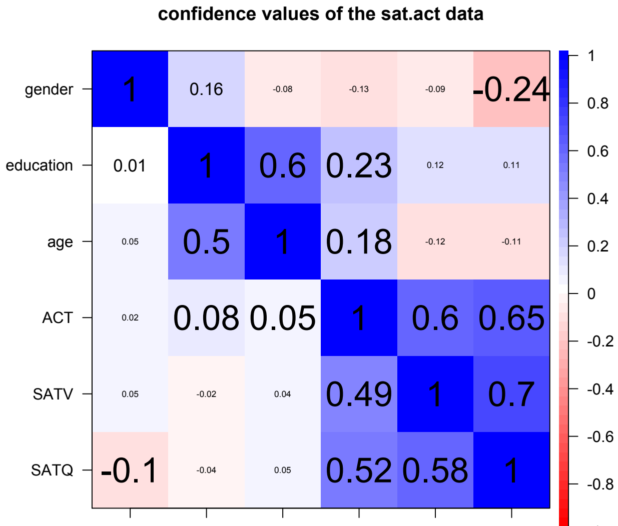
The SAT.ACT correlations. Confidence values from resampling

```
ci <- cor.ci(cleaned,main='Heat map of sat.act')
```



The SAT.ACT bootstrapped confidence intervals of correlation

cor.plot.upperLowerCi(ci,main="Heat map of sat.act")



Are education and gender independent? χ^2 Test of association

```
T <- with(my.data,table(gender,education))
```

```
> T
      education
gender   0   1   2   3   4   5
     1  27  20  23  80  51  46
     2  30  25  21 195  87  95

> chisq.test(T)
        Pearson's Chi-squared test

data:  T
X-squared = 16.0851, df = 5, p-value = 0.006605
```

1. First create a table of associations
   - Do this on our data (my.data)
   - Use the "with" command to specify the data set
2. Show the table
3. Apply χ^2 test

Finding χ^2 from a table of data

- Consider the effect of a treatment on later arrest (From Ashley Kendall, 2016)

Condition	Arrested	Not Arrested
Control	14	21
Treatment	3	23

R code

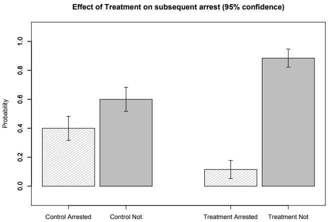
```
ak.df <- data.frame(Control=c(14,21),Treated =c(3,23))
rownames(ak.df) <- c("Arrested","Not Arrested")
ak.df #show the data frame
chisq.test(ak.df)  #Test it using the Yates continuity correction
```

```
> ak.df #show the data frame
             Control Treated
Arrested          14       3
Not Arrested      21      23
> chisq.test(ak.df)  #Test it using the Yates continuity correctio
        Pearson's Chi-squared test with Yates' continuity correcti
data:  ak.df
X-squared = 4.6791, df = 1, p-value = 0.03053
```

Graph the tabled data showing confidence intervals of proportions

R code

```
ak.df <- data.frame(Control=c(14,21),Treated =c(3,23))
ak.p <- t(t(ak.df)/colSums(ak.df)) #convert to probabilities
standard.error  <- sqrt(ak.p[1,] * ak.p[2,]/colSums(ak.df))
stats <- data.frame(mean=as.vector(ak.p),
                        se=rep(standard.error,each=2))
rownames(stats) <- c("Control Arrested","Control Not",
                      "Treatment Arrested","Treatment Not")
error.bars(stats=stats,bars=TRUE,space=c(.1,.1,1,.1),
 density=c(20,-10,20,-10),ylab="Probability",
 xlab="Control vs Treatment",
 main ="Effect of Treatment on subsequent arrest (95% confidence)")
```




```
round(stats,2)
                   mean   se
Control Arrested   0.40 0.08
Control Not        0.60 0.08
Treatment Arrested 0.12 0.06
Treatment Not      0.88 0.06
```

Multiple regression and the general linear model

1. Use the sat.act data example
2. Do the linear model
3. Summarize the results

R code

```
mod1 <- lm(SATV ~ education + gender + SATQ,data=my.data)
summary(mod1,digits=2)
```

```
Call:
lm(formula = SATV ~ education + gender + SATQ, data = my.data)
Residuals:
    Min      1Q  Median      3Q     Max
-372.91  -49.08    2.30   53.68  251.93
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 180.87348   23.41019   7.726 3.96e-14 ***
education     1.24043    2.32361   0.534  0.59363
gender       20.69271    6.99651   2.958  0.00321 **
SATQ          0.64489    0.02891  22.309  < 2e-16 ***
Signif. codes:  0 Ô***Õ 0.001 Ô**Õ 0.01 Ô*Õ 0.05 Ô.Õ 0.1 Ô Õ 1
Residual standard error: 86.24 on 683 degrees of freedom
  (13 observations deleted due to missingness)
Multiple R-squared: 0.4231,      Adjusted R-squared: 0.4205
F-statistic:   167 on 3 and 683 DF,  p-value: < 2.2e-16
```

Zero center the data before examining interactions

In order to examine interactions using multiple regression, we must first "zero center" the data. This may be done using the `scale` function. By default, `scale` will standardize the variables. So to keep the original metric, we make the scaling parameter FALSE.

R code

```
csat <- data.frame(scale(my.data,scale=FALSE))
describe(csat)  #centered not standardized data
```

	vars	n	mean	sd	median	trimmed	mad	min	max
gender	1	700	0	0.48	0.35	0.04	0.00	-0.65	0.35
education	2	700	0	1.43	-0.16	0.14	1.48	-3.16	1.84
age	3	700	0	9.50	-3.59	-1.73	5.93	-12.59	39.41
ACT	4	700	0	4.82	0.45	0.30	4.45	-25.55	7.45
SATV	5	700	0	112.90	7.77	7.22	118.61	-412.23	187.77
SATQ	6	687	0	115.64	9.78	7.04	118.61	-410.22	189.78

Note that we need to take the output of scale (which comes back as a matrix) and make it into a data.frame if we want to use the linear model on it.

Zero center the data before examining interactions

R code

```
csat <- data.frame(scale(my.data,scale=FALSE))
mod2 <-  lm(SATV  ~ education * gender * SATQ,data=csat)
summary(mod2)
```

```
Call:
all:
lm(formula = SATV ~ education * gender * SATQ, data = csat)

Residuals:
    Min      1Q  Median      3Q     Max
-372.53  -48.76    3.33   51.24  238.50

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             0.773576   3.304938   0.234  0.81500
education               2.517314   2.337889   1.077  0.28198
gender                 18.485906   6.964694   2.654  0.00814 **
SATQ                    0.620527   0.028925  21.453  < 2e-16 ***
education:gender        1.249926   4.759374   0.263  0.79292
education:SATQ         -0.101444   0.020100  -5.047 5.77e-07 ***
gender:SATQ             0.007339   0.060850   0.121  0.90404
education:gender:SATQ   0.035822   0.041192   0.870  0.38481
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 84.69 on 679 degrees of freedom
  (13 observations deleted due to missingness)
Multiple R-squared:  0.4469,      Adjusted R-squared:  0.4412
F-statistic: 78.37 on 7 and 679 DF,  p-value: < 2.2e-16
```

Compare model 1 and model 2 using anova

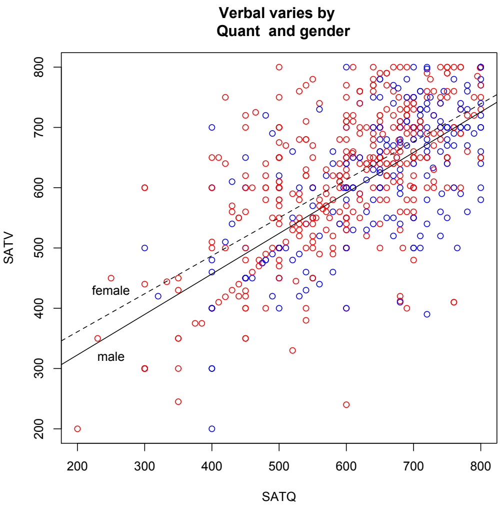
Test the difference between the two linear models

R code

```
anova(mod1,mod2)
```

```
Analysis of Variance Table
Analysis of Variance Table

Model 1: SATV ~ education + gender + SATQ
Model 2: SATV ~ education * gender * SATQ
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1    683 5079984
2    679 4870243  4    209742 7.3104 9.115e-06 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
```

Show the regression lines by gender



First plot all the data.

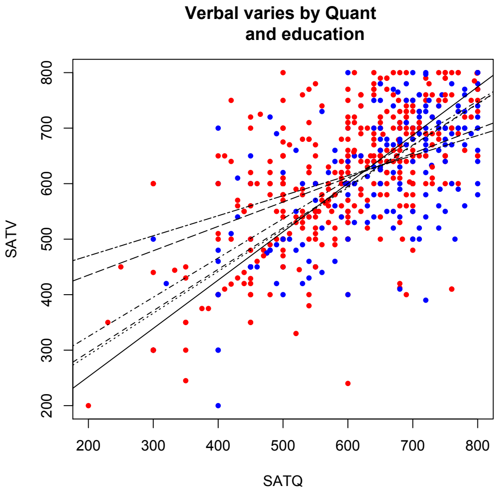
Then add the regression lines.

Then put a title on the whole thing.

R code

```
#first plot the data points
with(my.data,plot(SATV~SATQ,
 col=c("blue","red")[gender]))
#add the regression lines
by(my.data,my.data$gender,
   function(x) abline
  (lm(SATV~SATQ,data=x),
   lty=c("solid","dashed"
        )[x$gender]))
#add a title
title("Verbal varies by
   Quant  and gender")
 #label the lines
text(250,320,"male")
text(250,430,"female")
```

Show the regression lines by education



Do this again, but for levels of education as the moderator.

R code

```
with(my.data,plot(SATV~SATQ,
     col=c("blue","red")[gender],
     pch=20)) #plot character
by(my.data,my.data$education,
   function(x) abline
 (lm(SATV~SATQ,data=x),
 lty=c("solid", "dashed","dotted",
   "dotdash", "longdash",
  "twodash")[(x$education+1)]))

 title("Verbal varies by Quant
         and education")
```

Questions?

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 (including mediation and moderation)
- 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 III

-> Part II: Introduction and Overview
Basic statistics and graphics
    4 steps: read, explore, test, graph
Basic descriptive statistics and graphics
    Graphic displays
    Correlations
Inferential statistics
    The t-test
    ANOVA
Linear Regression, Moderation and Mediation
    Regression from the raw data
    Regression from covariance/correlation matrices
R structure
    Basic R
    Objects and Functions
-> Part IV: Psychometrics

Using R for psychological statistics: Basic statistics

1. Writing syntax
   - For a single line, just type it
   - Mistakes can be redone by using the up arrow key
   - For longer code, use a text editor (built into some GUIs)
2. Data entry
   - Using built in data sets for examples
   - Copying from another program
   - Reading a text or csv file
   - Importing from SPSS or SAS
   - Simulate it (using various simulation routines)
3. Descriptives
   - Graphical displays
   - Descriptive statistics
   - Correlation
4. Inferential
   - the t test
   - the F test
   - the linear model

Data entry overview

1. Using built in data sets for examples
   - `data()` will list > 100 data sets in the `datasets` package as well as all sets in loaded packages.
   - Most packages have associated data sets used as examples
   - *psych* has > 50 example data sets
2. Copying from another program
   - use copy and paste into R using `read.clipboard` and its variations
3. Reading a text or csv file
   - read a local or remote file
4. Importing from SPSS or SAS
   - Use either the *foreign*, *haven* or *rio* packages
5. Simulate it (using various simulation routines)
6. Model it using simulations (e.g., `cta` (Revelle & Condon, 2015))

Examples of built in data sets from the psych package

```
> data(package="psych")
```

ability	16 multiple choice IQ items from the ICAR project (Condon & Revelle, 2014)
Bechtoldt	Seven data sets showing a bifactor solution (Bechtoldt, 1961; Holzinger & Swineford, 1937; Thurstone & Thurstone, 1941).
Dwyer	8 cognitive variables used by Dwyer (1937) for an example.
Reise	Seven data sets showing a bifactor solution (Reise, Morizot & Hays, 2007).
affect	Data sets of affect and arousal scores as a function of personality and movie conditions (Smillie, Cooper, Wilt & Revelle, 2012)
income	US family income from US census 2008
bfi	25 Personality items representing 5 factors (N=2800)
blot	Bond's Logical Operations Test - BLOT (N=150) (Bond, 1995)
burt	11 emotional variables from Burt (1915)
cities	Distances between 11 US cities
epi.bfi	13 scales from the Eysenck Personality Inventory and Big 5 inventory
income	US family income from US census 2008
msq	75 mood items from the Motivational State Questionnaire for N=3896
neo	NEO correlation matrix from the NEOPI-R manual (Costa & McCrae, 1985)
sat.act	3 Measures of ability: SATV, SATQ, ACT (N=700)
Thurstone	Seven data sets showing a bifactor solution.
veg (vegetables)	Paired comparison of preferences for 9 vegetables (Guilford, 1954)

Reading data from another program –using the clipboard

1. Read the data in your favorite spreadsheet or text editor
2. Copy to the clipboard
3. Execute the appropriate `read.clipboard` function with or without various options specified

```
my.data <- read.clipboard() #assumes headers and tab or space deli
my.data <- read.clipboard.csv() #assumes headers and comma delimi
my.data <- read.clipboard.tab() #assumes headers and tab delimite
                                         (e.g., from Excel)
my.data <- read.clipboard.lower()  #read in a matrix given the low
my.data <- read.clipboard.upper() # or upper  off diagonal
my.data <- read.clipboard.fwf()  #read in data using a fixed forma
                                       (see read.fwf for instructi
```

4. `read.clipboard()` has default values for the most common cases and these do not need to be specified. Consult ?read.clipboard for details. In particular, are headers provided for each column of input?

Reading from a local or remote file

1. Perhaps the standard way of reading in data is using the `read` command.
   - First must specify the location of the file
   - Can either type this in directly or use the `file.choose` function. This goes to your normal system file handler.
   - The file name/location can be a remote URL.
2. Two examples of reading data

R code

```
file.name <- file.choose() #this opens a window to allow you find the file
#or
file.name="http://personality-project.org/r/datasets/R.appendix1.data"
my.data <- read.file(file.name)
#or
my.data = read.table(file.name,header=TRUE)   #the conventional way
dim(my.data) #find the dimensionality of our data
describe(my.data) #describe it to check the means, ranges, etc.
```

```
> dim(my.data )  #what are the dimensions of what we read?
[1] 18  2
> describe(my.data ) #do the data look right?
          var  n  mean   sd median trimmed  mad min max range skew kurtosis   se
Dosage*     1 18  1.89 0.76      2    1.88 1.48   1   3     2 0.16    -1.12 0.18
Alertness   2 18 27.67 6.82     27   27.50 8.15  17  41    24 0.25    -0.68 1.61
```

Put it all together: read, show, describe

R code

```
datafilename="http://personality-project.org/r/datasets/R.appendix1.data"
data.ex1<- read.file(datafilename)
dim(data.ex1)  #what are the dimensions of what we read?
data.ex1  #show the data
headTail(data.ex1) #just the top and bottom lines
describe(data.ex1) #descriptive stats
```

```
   Dosage Alertness
1       a        30
2       a        38
...  (rows deleted by hand)
17      c        20
18      c        19

> headTail(data.ex1) #just the top and bottom lines
    Dosage Alertness
1        a        30
2        a        38   'head' rows
3        a        35
4        a        41
...    <NA>      ... (rows automatically deleted)
15       c        17
16       c        21
17       c        20  'tail' rows
18       c        19
> describe(data.ex1) #descriptive stats
          vars  n  mean   sd median trimmed  mad min max range skew kurtosis   se
Dosage*      1 18  1.89 0.76      2    1.88 1.48   1   3     2 0.16    -1.35 0.18
Alertness    2 18 27.67 6.82     27   27.50 8.15  17  41    24 0.25    -1.06 1.61
```

1. Read the data from a remote file
2. Show all the cases (problematic if there are are 100s – 1000s)
3. Just show the first and last (4) lines
4. Find descriptive statistics

However, some might want to Import SAS or SPSS files

The first thing to try is the `read.file` function. For more complicated data sets, there are several different packages that make importing SPSS, SAS, Systat, etc. files possible to do.

- **read.file** Function in psych to read .txt, .csv, .sav, ,xpt, .r, ,.rda, .text (etc.)
- **foreign** Read data stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase. Comes installed with R. Somewhat complicated syntax.
- **haven** Reads/writes SPSS and Stata files. Handles SPSS labels nicely (keeps the item labels, but converts the data to factors).
- **rio** A general purpose package that requires installation of many of the other packages used for data import. Easiest to use, but overkill if just reading in one type of file. Basically a front end to many import/export packages. It determines which package to use based

Read a “foreign” file e.g., an SPSS sav file, using foreign package

`read.spss` Reads a file stored by the SPSS save or export commands. (The defaults lead to problems, make sure to specify that you want use.value.labels = FALSE, to.data.frame = TRUE)

```
read.spss(file, use.value.labels = FALSE, to.data.frame = TRUE,
          max.value.labels = Inf, trim.factor.names = FALSE,
          trim_values = TRUE, reencode = NA, use.missings = to.data.frame)
```

file	Character string: the name of the file or URL to read.
use.value.labels	Convert variables with value labels into R factors with those levels? Should be FALSE
to.data.frame	return a data frame? Defaults to FALSE, probably should be TRUE in most cases.
max.value.labels	Only variables with value labels and at most this many unique values will be converted to factors if use.value.labels = *TRUE*.
trim.factor.names	Logical: trim trailing spaces from factor levels?
trim_values	logical: should values and value labels have trailing spaces ignored when matching for use.value.labels = *TRUE*?
use.missings	logical: should information on user-defined missing values be used to set the corresponding values to NA?

An example of reading from an SPSS file using foreign

```
> library(foreign)

> datafilename <- "http://personality-project.org/r/datasets/finkel.sav"

>  eli <- read.spss(datafilename,to.data.frame=TRUE,
                          use.value.labels=FALSE)
> headTail(eli,2,2)
> describe(eli,skew=FALSE)
```

```
     USER HAPPY SOULMATE ENJOYDEX UPSET
1   "001"     4        7        7     1
2   "003"     6        5        7     0
...  <NA>   ...      ...      ...   ...
68  "076"     7        7        7     0
69  "078"     2        7        7     1
>
          var  n  mean    sd median trimmed   mad min max range   se
USER*       1 69 35.00 20.06     35   35.00 25.20   1  69    68 2.42
HAPPY       2 69  5.71  1.04      6    5.82  0.00   2   7     5 0.13
SOULMATE    3 69  5.09  1.80      5    5.32  1.48   1   7     6 0.22
ENJOYDEX    4 68  6.47  1.01      7    6.70  0.00   2   7     5 0.12
UPSET       5 69  0.41  0.49      0    0.39  0.00   0   1     1 0.06
```

1. Make the *foreign* package active
2. Specify the name (and location) of the file to read
3. Read from a SPSS file
4. Show the top and bottom 2 cases
5. Describe it to make sure it is right

An example of reading from an SPSS file using rio

```
> library(rio)

> datafilename <- "http://personality-project.org/r/datasets/finkel.sav"

>  eli <- import(datafilename)   #note that it figures out what to do
> headTail(eli,2,2) #The first and last 2
> describe(eli,skew=FALSE)

     USER HAPPY SOULMATE ENJOYDEX UPSET
1   "001"     4        7        7     1
2   "003"     6        5        7     0
...  <NA>   ...      ...      ...   ...
68  "076"     7        7        7     0
69  "078"     2        7        7     1
>
         var  n  mean    sd median trimmed   mad min max range   se
USER*      1 69 35.00 20.06     35   35.00 25.20   1  69    68 2.42
HAPPY      2 69  5.71  1.04      6    5.82  0.00   2   7     5 0.13
SOULMATE   3 69  5.09  1.80      5    5.32  1.48   1   7     6 0.22
ENJOYDEX   4 68  6.47  1.01      7    6.70  0.00   2   7     5 0.12
UPSET      5 69  0.41  0.49      0    0.39  0.00   0   1     1 0.06
```

1. Make the *rio* package active
2. Specify the name (and location) of the file to read
3. Import from a SPSS file
4. Show the top and bottom 2 cases
5. Describe it to make sure it is right

An example of reading from an SPSS file using haven

```
> library(haven)


> datafilename <- "http://personality-project.org/r/datasets/finkel.sav"


> eli <- read_spss(datafilename)   #note that it figures out what to d
> headTail(eli,3,2) The first 3 and last 2
> describe(eli,skew=FALSE)
```

```
    USER HAPPY SOULMATE ENJOYDEX UPSET
1   "001"     4        7        7     1
2   "003"     6        5        7     0
3   "004"     6        7        7     0
...  <NA>   ...      ...      ...   ...
68  "076"     7        7        7     0
69  "078"     2        7        7     1>
          var  n  mean    sd median trimmed   mad min max range   se
USER*       1 69 35.00 20.06     35   35.00 25.20   1  69    68 2.42
HAPPY       2 69  5.71  1.04      6    5.82  0.00   2   7     5 0.13
SOULMATE    3 69  5.09  1.80      5    5.32  1.48   1   7     6 0.22
ENJOYDEX    4 68  6.47  1.01      7    6.70  0.00   2   7     5 0.12
UPSET       5 69  0.41  0.49      0    0.39  0.00   0   1     1 0.06
```

1. Make the *haven* package active
2. Specify the name (and location) of the file to read
3. Import from a SPSS file
4. Show the top 3 and bottom 2 cases
5. Describe it to make sure it is right

`read.file` as a convenient solution to reading files

1. Combines `file.choose` and `read.table`
2. Also, based upon the suffix of the data, will choose the most likely way to read a SPSS, csv, text, rds or SAS export file.
3. Not as powerful as *foreign* or *rio* but easier.
4. Automatically reads SPSS .sav files as numeric values but can read them with the item information as well.

R code

```
eli <- read.file().  #goes off and searches for a local file
datafilename <- "http://personality-project.org/r/datasets/finkel.sav"
eli <- read.file(datafilename). #uses that remote address to get it
ashley <- read.file() #a file from Ashley Kendall on my computer
kendall <- read.file(read.file(use.value.labels=TRUE) #keep the labels
ashley[1:3,8:17]
kendall[1:3,8:17]
```

```
ashley[1:3,8:17]
  HighNA LowPA LowNA Active Alert Nervs Frust Worried Irrit Stress
1      8     3     0      1     0     1     2       0     3      2
2      6     1     0      0     2     0     3       0     2      1
3      1     7     0      3     3     1     0       0     0      0
> kendall[1:3,8:17]
  HighNA LowPA LowNA     Active Alert      Nervs      Frust Worried      Irrit     Stress
1      8     3     0   a little     0   a little   somewhat       0  very much   somewhat
2      6     1     0 not at all     2 not at all  very much       0   somewhat   a little
3      1     7     0  very much     3   a little not at all       0 not at all not at all
```

Simulate data (Remember to always call them simulated!)

For many demonstration purposes, it is convenient to generate simulated data with a certain defined structure. The *psych* package has a number of built in simulation functions. Here are a few of them.

1. Simulate various item structures
   - sim.congeneric A one factor congeneric measure model
   - sim.items A two factor structure with either simple structure or a circumplex structure.
   - sim.rasch Generate items for a one parameter IRT model.
   - sim.irt Generate items for a one-four parameter IRT Model
2. Simulate various factor structures
   - sim.simplex Default is a four factor structure with a three time point simplex structure.
   - sim.hierarchical Default is 9 variables with three correlated factors.

Get the data and look at them

Read in some data, look at the first and last few cases (using `headTail`), and then get basic descriptive statistics. For this example, we will use a built in data set.

R code

```
headTail(epi.bfi)
```

	epiE	epiS	epiImp	epilie	epiNeur	bfagree	bfcon	bfext	bfneur	bfop
1	18	10	7	3	9	138	96	141	51	1
2	16	8	5	1	12	101	99	107	116	1
3	6	1	3	2	5	143	118	38	68	
4	12	6	4	3	15	104	106	64	114	1
...	...	...	...	...	...	...	...	...	...	.
228	12	7	4	3	15	155	129	127	88	1
229	19	10	7	2	11	162	152	163	104	1
230	4	1	1	2	10	95	111	75	123	1
231	8	6	3	2	15	85	62	90	131	

epi.bfi has 231 cases from two personality measures.

Now find the descriptive statistics for this data set

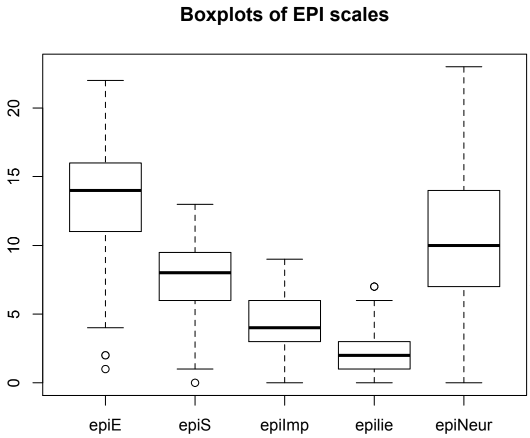
R code

```
describe(epi.bfi)
```

```
         var   n   mean    sd median trimmed   mad min max range  skew kurtosis   se
epiE       1 231  13.33  4.14     14   13.49  4.45   1  22    21 -0.33    -0.01 0.27
epiS       2 231   7.58  2.69      8    7.77  2.97   0  13    13 -0.57     0.04 0.18
epiImp     3 231   4.37  1.88      4    4.36  1.48   0   9     9  0.06    -0.59 0.12
epilie     4 231   2.38  1.50      2    2.27  1.48   0   7     7  0.66     0.30 0.10
epiNeur    5 231  10.41  4.90     10   10.39  4.45   0  23    23  0.06    -0.46 0.32
bfagree    6 231 125.00 18.14    126  125.26 17.79  74 167    93 -0.21    -0.22 1.19
bfcon      7 231 113.25 21.88    114  113.42 22.24  53 178   125 -0.02     0.29 1.44
bfext      8 231 102.18 26.45    104  102.99 22.24   8 168   160 -0.41     0.58 1.74
bfneur     9 231  87.97 23.34     90   87.70 23.72  34 152   118  0.07    -0.51 1.54
bfopen    10 231 123.43 20.51    125  123.78 20.76  73 173   100 -0.16    -0.11 1.35
bdi       11 231   6.78  5.78      6    5.97  4.45   0  27    27  1.29     1.60 0.38
traitanx  12 231  39.01  9.52     38   38.36  8.90  22  71    49  0.67     0.54 0.63
stateanx  13 231  39.85 11.48     38   38.92 10.38  21  79    58  0.72     0.04 0.76
```

Boxplots are a convenient descriptive device

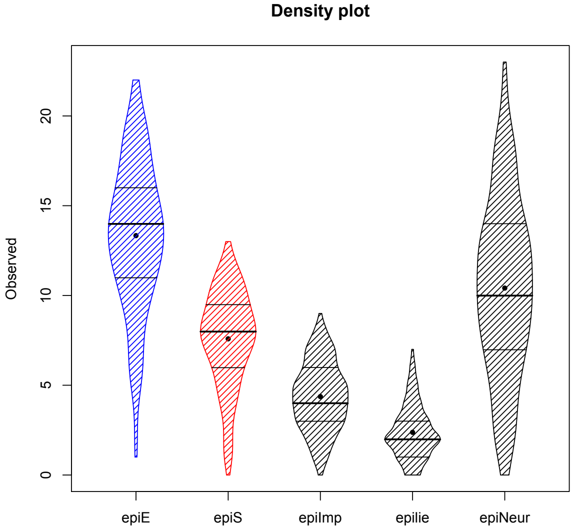
Show the Tukey "boxplot" for the Eysenck Personality Inventory



Use the box plot function and select the first five variables.

```
my.data <- epi.bfi
boxplot(my.data[1:5])
```

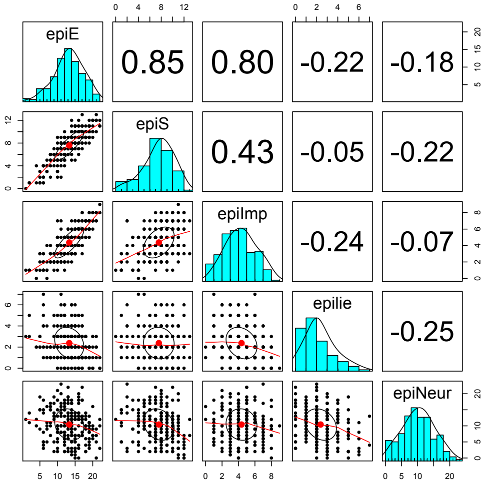
An alternative display is a 'violin' plot (available as `violinBy`)



Use the `violinBy` function from *psych*

```
violinBy(my.data[1:5])
```

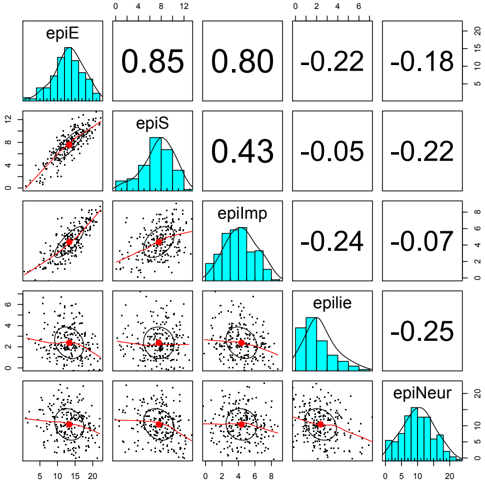
Plot the scatter plot matrix (SPLOM) of the first 5 variables using the `pairs.panels` function. Note that the plotting points overlap because of the polytomous nature of the data.



Use the `pairs.panels` function from *psych*

```
pairs.panels(my.data[1:5])
```

Plot the scatter plot matrix (SPLOM) of the first 5 variables using the `pairs.panels` function but with smaller plot charactet (pch) and jittering the points in order to better show the distributions.



Use the `pairs.panels` function from *psych*

```
pairs.panels(my.data[1:5],pch='.',
      jiggle=TRUE)
```

Find the correlations for this data set, round off to 2 decimal places.

Because we have some missing data, we use "pairwise complete" correlations. For the purists amongst us, it is irritating that the columns are not equally spaced.

R code

```
round(cor(my.data, use = "pairwise"), 2)
```

	epiE	epiS	epiImp	epilie	epiNeur	bfagree	bfcon	bfext	bfneur	bfopen	bdi	traitanx	st
epiE	1.00	0.85	0.80	-0.22	-0.18	0.18	-0.11	0.54	-0.09	0.14	-0.16	-0.23	
epiS	0.85	1.00	0.43	-0.05	-0.22	0.20	0.05	0.58	-0.07	0.15	-0.13	-0.26	
epiImp	0.80	0.43	1.00	-0.24	-0.07	0.08	-0.24	0.35	-0.09	0.07	-0.11	-0.12	
epilie	-0.22	-0.05	-0.24	1.00	-0.25	0.17	0.23	-0.04	-0.22	-0.03	-0.20	-0.23	
epiNeur	-0.18	-0.22	-0.07	-0.25	1.00	-0.08	-0.13	-0.17	0.63	0.09	0.58	0.73	
bfagree	0.18	0.20	0.08	0.17	-0.08	1.00	0.45	0.48	-0.04	0.39	-0.14	-0.31	
bfcon	-0.11	0.05	-0.24	0.23	-0.13	0.45	1.00	0.27	0.04	0.31	-0.18	-0.29	
bfext	0.54	0.58	0.35	-0.04	-0.17	0.48	0.27	1.00	0.04	0.46	-0.14	-0.39	
bfneur	-0.09	-0.07	-0.09	-0.22	0.63	-0.04	0.04	0.04	1.00	0.29	0.47	0.59	
bfopen	0.14	0.15	0.07	-0.03	0.09	0.39	0.31	0.46	0.29	1.00	-0.08	-0.11	
bdi	-0.16	-0.13	-0.11	-0.20	0.58	-0.14	-0.18	-0.14	0.47	-0.08	1.00	0.65	
traitanx	-0.23	-0.26	-0.12	-0.23	0.73	-0.31	-0.29	-0.39	0.59	-0.11	0.65	1.00	
stateanx	-0.13	-0.12	-0.09	-0.15	0.49	-0.19	-0.14	-0.15	0.49	-0.04	0.61	0.57	

Find the correlations for this data set, round off to 2 decimal places using `lowerCor`

This is just a wrapper for round(cor(x,use='pairwise'),2) that has been prettied up with `lowerMat`.

R code

```
lowerCor(my.data)
```

```
         epiE  epiS  epImp epili epiNr bfagr bfcon bfext bfner bfopn bdi   trtnx sttnx
epiE      1.00
epiS      0.85  1.00
epiImp    0.80  0.43  1.00
epilie   -0.22 -0.05 -0.24  1.00
epiNeur  -0.18 -0.22 -0.07 -0.25  1.00
bfagree   0.18  0.20  0.08  0.17 -0.08  1.00
bfcon    -0.11  0.05 -0.24  0.23 -0.13  0.45  1.00
bfext     0.54  0.58  0.35 -0.04 -0.17  0.48  0.27  1.00
bfneur   -0.09 -0.07 -0.09 -0.22  0.63 -0.04  0.04  0.04  1.00
bfopen    0.14  0.15  0.07 -0.03  0.09  0.39  0.31  0.46  0.29  1.00
bdi      -0.16 -0.13 -0.11 -0.20  0.58 -0.14 -0.18 -0.14  0.47 -0.08  1.00
traitanx -0.23 -0.26 -0.12 -0.23  0.73 -0.31 -0.29 -0.39  0.59 -0.11  0.65  1.00
stateanx -0.13 -0.12 -0.09 -0.15  0.49 -0.19 -0.14 -0.15  0.49 -0.04  0.61  0.57  1.00
```

Test the significance and use Holm correction for multiple tests

R code

```
corr.test(my.data)
```

```
Call:corr.test(x = my.data)
Correlation matrix
         epiE  epiS epiImp epilie epiNeur bfagree bfcon bfext bfneur bfopen   bdi traitanx st
epiE     1.00  0.85   0.80  -0.22   -0.18    0.18 -0.11  0.54  -0.09   0.14 -0.16    -0.23
epiS     0.85  1.00   0.43  -0.05   -0.22    0.20  0.05  0.58  -0.07   0.15 -0.13    -0.26
epiImp   0.80  0.43   1.00  -0.24   -0.07    0.08 -0.24  0.35  -0.09   0.07 -0.11    -0.12
..
stateanx -0.13 -0.12  -0.09  -0.15    0.49   -0.19 -0.14 -0.15   0.49  -0.04  0.61     0.57
Sample Size
        epiE epiS epiImp epilie epiNeur bfagree bfcon bfext bfneur bfopen bdi traitanx statea
epiE     231  231    231    231     231     231   231   231    231    231 231      231      2
..
stateanx 231  231    231    231     231     231   231   231    231    231 231      231      2
Probability values (Entries above the diagonal are adjusted for multiple tests.)
        epiE epiS epiImp epilie epiNeur bfagree bfcon bfext bfneur bfopen  bdi traitanx state
epiE    0.00 0.00   0.00   0.03    0.27    0.27  1.00  0.00   1.00   1.00 0.59     0.02
epiS    0.00 0.00   0.00   1.00    0.04    0.08  1.00  0.00   1.00   0.62 1.00     0.00
epiImp  0.00 0.00   0.00   0.01    1.00    1.00  0.01  0.00   1.00   1.00 1.00     1.00
epilie  0.00 0.43   0.00   0.00    0.01    0.32  0.03  1.00   0.03   1.00 0.08     0.02
epiNeur 0.01 0.00   0.26   0.00    0.00    1.00  1.00  0.33   0.00   1.00 0.00     0.00
bfagree 0.01 0.00   0.23   0.01    0.21    0.00  0.00  0.00   1.00   0.00 0.95     0.00
bfcon   0.08 0.48   0.00   0.00    0.04    0.00  0.00  0.00   1.00   0.00 0.25     0.00
bfext   0.00 0.00   0.00   0.50    0.01    0.00  0.00  0.00   1.00   0.00 0.99     0.00
bfneur  0.15 0.30   0.18   0.00    0.00    0.50  0.50  0.57   0.00   0.00 0.00     0.00
bfopen  0.04 0.02   0.30   0.70    0.19    0.00  0.00  0.00   0.00   0.00 1.00     1.00
bdi     0.02 0.04   0.11   0.00    0.00    0.03  0.01  0.03   0.00   0.25 0.00     0.00
traitanx 0.00 0.00  0.07   0.00    0.00    0.00  0.00  0.00   0.00   0.11 0.00     0.00
stateanx 0.05 0.07  0.18   0.02    0.00    0.00  0.04  0.02   0.00   0.52 0.00     0.00
>
```

t.test demonstration with Student's data using `cushny` dataset

William Gossett, publishing under the name Student reported a small sample approximation (t) to the large sample z test. His first example was a data set on the different effect of optical isomers of hyoscyamine hydrobromide reported by Cushny & Peebles (1905). The sleep of 10 patients was measured without any drug and then following administration of D. and L isomers. The data from Cushny are available as the `cushny` data set.

Variable	Cntrl	drug1	drg2L	drg2R	delt1	dlt2L	dlt2R
1	0.60	1.3	2.50	2.10	0.70	1.90	1.50
2	3.00	1.4	3.80	4.40	-1.60	0.80	1.40
3	4.70	4.5	5.80	4.70	-0.20	1.10	0.00
4	5.50	4.3	5.60	4.80	-1.20	0.10	-0.70
5	6.20	6.1	6.10	6.70	-0.10	-0.10	0.50
6	3.20	6.6	7.60	8.30	3.40	4.40	5.10
7	2.50	6.2	8.00	8.20	3.70	5.50	5.70
8	2.80	3.6	4.40	4.30	0.80	1.60	1.50
9	1.10	1.1	5.70	5.80	0.00	4.60	4.70
10	2.90	4.9	6.30	6.40	2.00	3.40	3.50
Mean	3.25	4.0	5.58	5.57	0.75	2.33	2.32
Sd	1.78	2.1	1.66	1.91	1.79	2.00	2.27

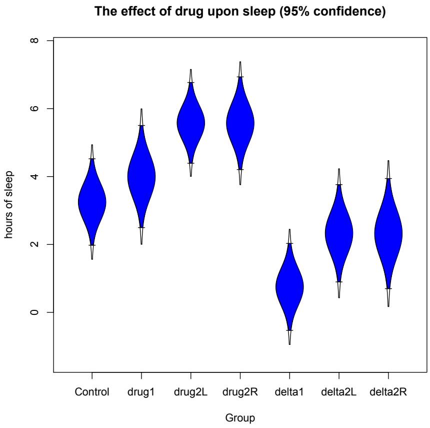
R code

```
error.bars(cushny,xlab="Group",ylab="hours of sleep",
    main="The effect of drug upon sleep (95% confidence)")
```

The `cushny` data set with error bars (Cushny & Peebles, 1905)

R code

```
error.bars(cushny,xlab="Group",ylab="hours of sleep",
      main="The effect of drug upon sleep (95% confidence)")
```



We can show these data graphically using the `error.bars` function. We pass labels to the x and y axis using the xlab and ylab parameters, and then supply an appropriate figure title.
We will use these data to show how to do t-tests as well as the generalization to Analysis of Variance.

Student's t.test: As done by Student

R code

```
with(cushny,t.test(delta1))  #control versus drug 1 difference scores
with(cushny,t.test(delta2L))   #control versus drug2L difference scores
with(cushny,t.test(delta1,delta2L,paired=TRUE)) #difference of differences
```

```
> with(cushny,t.test(delta1))  #control versus drug 1 difference scores
        One Sample t-test
data:  delta1
t = 1.3257, df = 9, p-value = 0.2176
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.5297804  2.0297804
sample estimates:
mean of x
     0.75
with(cushny,t.test(delta2L))   #control versus drug2L difference scores
        One Sample t-test
data:  delta2L
t = 3.6799, df = 9, p-value = 0.005076
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.8976775 3.7623225
sample estimates:
mean of x
     2.33
> with(cushny,t.test(delta1,delta2L,paired=TRUE)) #difference of differences
        Paired t-test
data:  delta1 and delta2L
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.4598858 -0.7001142
```

Two ways of organizing the data: Wide versus long

We can take the wide format of the cushny data set and make it long.

```
cushny[c("delta1","delta2L")]
   delta1 delta2L
1     0.7     1.9
2    -1.6     0.8
3    -0.2     1.1
4    -1.2     0.1
5    -0.1    -0.1
6     3.4     4.4
7     3.7     5.5
8     0.8     1.6
9     0.0     4.6
10    2.0     3.4
```

R code

```
long.sleep <-
  stack(cushny[c("delta1","delta2L")])
long.sleep
```

```
   values     ind
1     0.7  delta1
2    -1.6  delta1
3    -0.2  delta1
4    -1.2  delta1
5    -0.1  delta1
6     3.4  delta1
7     3.7  delta1
8     0.8  delta1
9     0.0  delta1
10    2.0  delta1
11    1.9 delta2L
12    0.8 delta2L
13    1.1 delta2L
14    0.1 delta2L
15   -0.1 delta2L
16    4.4 delta2L
17    5.5 delta2L
18    1.6 delta2L
19    4.6 delta2L
20    3.4 delta2L
```

R code

```
long.sleep <-
+   stack(cushny[c("delta1",
        "delta2L")])
```

```
> long.sleep
   values     ind
1     0.7  delta1
2    -1.6  delta1
3    -0.2  delta1
4    -1.2  delta1
5    -0.1  delta1
6     3.4  delta1
7     3.7  delta1
8     0.8  delta1
9     0.0  delta1
10    2.0  delta1
11    1.9 delta2L
12    0.8 delta2L
13    1.1 delta2L
14    0.1 delta2L
15   -0.1 delta2L
16    4.4 delta2L
17    5.5 delta2L
18    1.6 delta2L
19    4.6 delta2L
20    3.4 delta2L
```

R code

```
t.test(values ~ ind,data=long.sleep)
```

```
        Welch Two Sample t-test
data:  values by ind
t = -1.8608, df = 17.776, p-value = 0.07939
alternative hypothesis: true difference in means is not e
95 percent confidence interval:
 -3.3654832  0.2054832
sample estimates:
 mean in group delta1 mean in group delta2L
                 0.75                  2.33
```

But, the data were paired

R code

```
t.test(values ~ ind,data=long.sleep,
         paired=TRUE)
```

```
data:  values by ind
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true difference in means is not e
95 percent confidence interval:
 -2.4598858 -0.7001142
sample estimates:
mean of the differences
                  -1.58
```

t.test demonstration with Student's data (from the sleep dataset)

Sleep data set is just 2 columns of cushny

R code
```
sleep
```

```
> sleep
   extra group ID
1    0.7     1  1
2   -1.6     1  2
3   -0.2     1  3
4   -1.2     1  4
5   -0.1     1  5
6    3.4     1  6
7    3.7     1  7
8    0.8     1  8
9    0.0     1  9
10   2.0     1 10
11   1.9     2  1
12   0.8     2  2
13   1.1     2  3
14   0.1     2  4
15  -0.1     2  5
16   4.4     2  6
17   5.5     2  7
18   1.6     2  8
19   4.6     2  9
20   3.4     2 10
```

R code
```
with(sleep,t.test(extra~group))
with(sleep,t.test(extra~group,var.equal=TRUE))
```

```
        Welch Two Sample t-test
data:  extra by group
t = -1.8608, df = 17.776, p-value = 0.07939  <-- default value
t = -1.8608, df = 18, p-value = 0.07919. <- equal variances
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.3654832  0.2054832
sample estimates:
mean in group 1 mean in group 2
           0.75            2.33
```

But the data were actually paired. Do it for a paired t-test

R code
```
with(sleep,t.test(extra~group,paired=TRUE))
```

```
        Paired t-test
data:  extra by group
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.4598858 -0.7001142
sample estimates:
mean of the differences
                  -1.58
```

Analysis of Variance as special case of linear model

1. aov provides a wrapper to lm for fitting linear models to balanced or unbalanced experimental designs.
2. The main difference from lm is in the way print, summary and so on handle the fit: this is expressed in the traditional language of the analysis of variance rather than that of linear models.
3. If the formula contains a single Error term, this is used to specify error strata, and appropriate models are fitted within each error stratum.
4. The formula can specify multiple responses.
5. aov is designed for balanced designs, and the results can be hard to interpret without balance: beware that missing values in the response(s) will likely lose the balance.
6. If there are two or more error strata, the methods used are statistically inefficient without balance, and it may be better to use `lme` in package *nlme*.

aov of the sleep data set: compare with the t.test results

R code

```
> sleep
```

```
   extra group ID
1    0.7     1  1
2   -1.6     1  2
3   -0.2     1  3
4   -1.2     1  4
5   -0.1     1  5
6    3.4     1  6
7    3.7     1  7
8    0.8     1  8
9    0.0     1  9
10   2.0     1 10
11   1.9     2  1
12   0.8     2  2
13   1.1     2  3
14   0.1     2  4
15  -0.1     2  5
16   4.4     2  6
17   5.5     2  7
18   1.6     2  8
19   4.6     2  9
20   3.4     2 10
```

R code

```
#independent subjects
summary(aov(extra ~ group, data=sleep))
```

```
> summary(aov(extra ~ group, data=sleep))
            Df Sum Sq Mean Sq F value Pr(>F)
group        1  12.48  12.482   3.463 0.0792 .
Residuals   18  64.89   3.605
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
t = -1.8608, df = 17.776, p-value = 0.07939. <--
t = -1.8608, df = 18, p-value = 0.07919. <- equal variances
```

R code

```
 #correlated subjects
summary(aov(extra~group + Error(ID),data=sleep))
```

```
> summary(aov(extra~group + Error(ID),data=sleep))

Error: ID
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  9  58.08   6.453

Error: Within
          Df Sum Sq Mean Sq F value  Pr(>F)
group      1 12.482  12.482    16.5 0.00283 **
Residuals  9  6.808   0.756
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
t = -4.0621, df = 9, p-value = 0.002833. <---
```

aov: an example of chemicals upon the growth of peas.

R code

```
npk #from Venables
```

```
   block N P K yield
1      1 0 1 1  49.5
2      1 1 1 0  62.8
3      1 0 0 0  46.8
4      1 1 0 1  57.0
5      2 1 0 0  59.8
6      2 1 1 1  58.5
7      2 0 0 1  55.5
8      2 0 1 0  56.0
9      3 0 1 0  62.8
10     3 1 1 1  55.8
11     3 1 0 0  69.5
12     3 0 0 1  55.0
13     4 1 0 0  62.0
14     4 1 1 1  48.8
15     4 0 0 1  45.5
16     4 0 1 0  44.2
17     5 1 1 0  52.0
18     5 0 0 0  51.5
19     5 1 0 1  49.8
20     5 0 1 1  48.8
21     6 1 0 1  57.2
22     6 1 1 0  59.0
23     6 0 1 1  53.2
24     6 0 0 0  56.0
```

Several models

R code

```
mod1 <- aov(yield ~ N,data=npk)
mod2 <- aov(yield ~ N+ P + N*P,data=npk)
mod2a <- aov(yield ~N*P,data=npk)
mod3 <- aov(yield ~ N*P*K,data=npk)
mod4 <- aov(yield ~ block + N*P*K,data=npk)
```

```
> summary(mod1)
            Df Sum Sq Mean Sq F value Pr(>F)
N            1  189.3  189.28   6.061 0.0221 *
Residuals   22  687.1   31.23
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

> summary(mod4)
            Df Sum Sq Mean Sq F value  Pr(>F)
block        5  343.3   68.66   4.447 0.01594 *
N            1  189.3  189.28  12.259 0.00437 **
P            1    8.4    8.40   0.544 0.47490
K            1   95.2   95.20   6.166 0.02880 *
N:P          1   21.3   21.28   1.378 0.26317
N:K          1   33.1   33.13   2.146 0.16865
P:K          1    0.5    0.48   0.031 0.86275
Residuals   12  185.3   15.44
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
```

Analysis of Variance: Another example

aov is designed for balanced designs, and the results can be hard to interpret without balance: beware that missing values in the response(s) will likely lose the balance.

R code

```
datafilename="http://personality-project.org/r/datasets/R.appendix2.data"
data.ex2=read.file(datafilename)   #read the data into a data.frame
data.ex2                                     #show the data
```

```
data.ex2
   Observation Gender Dosage Alertness
1            1      m      a         8
2            2      m      a        12
3            3      m      a        13
4            4      m      a        12
5            5      m      b         6
6            6      m      b         7
7            7      m      b        23
8            8      m      b        14
9            9      f      a        15
10          10      f      a        12
11          11      f      a        22
12          12      f      a        14
13          13      f      b        15
14          14      f      b        12
15          15      f      b        18
16          16      f      b        22
```

R code

```
#do the analysis of variance
aov.ex2 = aov(Alertness~Gender*Dosage,data=data.ex2)
summary(aov.ex2)       #show the summary table
```

```
Call:

 summary(aov.ex2)       #show the summary table
               Df Sum Sq Mean Sq F value Pr(>F)
Gender          1  76.56   76.56   2.952  0.111
Dosage          1   5.06    5.06   0.195  0.666
Gender:Dosage   1   0.06    0.06   0.002  0.962
Residuals      12 311.25   25.94
```

Analysis of Variance

Do the analysis of variances and the show the table of results.

R code

```
#do the analysis of variance
aov.ex2 <- aov(Alertness ~ Gender * Dosage, data=data.ex2)

summary(aov.ex2)        #show the summary table
aov.ex2.  #This shows the coefficients
```

```
>aov.ex2 <- aov(Alertness ~ Gender * Dosage, data=data.ex2)
> summary(aov.ex2)       #show the summary table
              Df Sum Sq Mean Sq F value Pr(>F)
Gender         1  76.56   76.56   2.952  0.111
Dosage         1   5.06    5.06   0.195  0.666
Gender:Dosage  1   0.06    0.06   0.002  0.962
Residuals     12 311.25   25.94

  aov(formula = Alertness ~ Gender * Dosage, data = data.ex2)

Terms:
                 Gender  Dosage Gender:Dosage Residuals
Sum of Squares  76.5625  5.0625        0.0625  311.2500
Deg. of Freedom       1       1             1        12

Residual standard error: 5.092887
Estimated effects may be unbalanced
```

Show the results table

R code

```
print(model.tables(aov.ex2,"means"),digits=3)
```

```
> print(model.tables(aov.ex2,"means"),digits=3)
Tables of means
Grand mean
14.0625

 Gender
Gender
    f     m
16.25 11.88

 Dosage
Dosage
    a     b
13.50 14.62

 Gender:Dosage
      Dosage
Gender a     b
     f 15.75 16.75
     m 11.25 12.50
```

Analysis of Variance: Within subjects

1. Somewhat more complicated because we need to convert "wide" data.frames to "long" or "narrow" data.frames.
2. This can be done by using the `stack` function. Some data sets are already in the long format.
3. A detailed discussion of how to work with repeated measures designs is at http://personality-project.org/r/r.anova.html and at http://personality-project.org/r
4. See also the tutorial by Jason French at http://jason-french.com/tutorials/repeatedmeasures.html
5. Many within subject designs can be treated as multi-level designs. For a discussion of analyzing multilevel data (particularly for personality dynamics), see http://personality-project.org/revelle/publications/rw.paid.17.final.pdf

Analysis of variance within subjects: Getting and showing the data

R code

```
filename="http://personality-project.org/r/datasets/R.appendix5.data"
data.ex5=read.file(filename)   #read the data into a data.frame
headTail(data.ex5,6,12)      #show the data (first 6, last 12)
```

```
    Obs Subject Gender Dosage Task Valence Recall
1     1       A      M      A    F     Neg      8
2     2       A      M      A    F     Neu      9
3     3       A      M      A    F     Pos      5
4     4       A      M      A    C     N eg       7
5     5       A      M      A    C     Neu      9
6     6       A      M      A    C     Pos     10
... ...    <NA>   <NA>   <NA> <NA>    <NA>    ...
97   97       Q      F      C    F     Neg     18
98   98       Q      F      C    F     Neu     17
99   99       Q      F      C    F     Pos     18
100 100       Q      F      C    C     Neg     17
101 101       Q      F      C    C     Neu     19
102 102       Q      F      C    C     Pos     19
103 103       R      F      C    F     Neg     19
104 104       R      F      C    F     Neu     17
105 105       R      F      C    F     Pos     19
106 106       R      F      C    C     Neg     22
107 107       R      F      C    C     Neu     21
108 108       R      F      C    C     Pos     20
```

Describe the data

R code

```
describe(data.ex5)
```

```
         vars   n  mean    sd median trimmed   mad min max range  skew kurtosis   se
Obs         1 108 54.50 31.32   54.5   54.50 40.03   1 108   107  0.00    -1.23 3.01
Subject*    2 108  9.50  5.21    9.5    9.50  6.67   1  18    17  0.00    -1.24 0.50
Gender*     3 108  1.50  0.50    1.5    1.50  0.74   1   2     1  0.00    -2.02 0.05
Dosage*     4 108  2.00  0.82    2.0    2.00  1.48   1   3     2  0.00    -1.53 0.08
Task*       5 108  1.50  0.50    1.5    1.50  0.74   1   2     1  0.00    -2.02 0.05
Valence*    6 108  2.00  0.82    2.0    2.00  1.48   1   3     2  0.00    -1.53 0.08
Recall      7 108 15.63  5.07   15.0   15.74  4.45   4  25    21 -0.13    -0.64 0.49
```

The * signify that the entries are not numerical, but rather categorical or logical.

Analysis of variance within subjects

R code

```
filename="http://personality-project.org/r/datasets/R.appendix5.data"
data.ex5=read.table(filename,header=TRUE) #read the data into a table
#do the anova
aov.ex5 =  aov(Recall~(Task*Valence*Gender*Dosage)+Error(Subject/(Task*Valence))+
 (Gender*Dosage),data.ex5)
 #look at the output
summary(aov.ex5)
```

```
Error: Subject
              Df Sum Sq Mean Sq F value Pr(>F)
Gender         1  542.3   542.3   5.685 0.0345 *
Dosage         2  694.9   347.5   3.643 0.0580 .
Gender:Dosage  2   70.8    35.4   0.371 0.6976
Residuals     12 1144.6    95.4
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Error: Subject:Task
                   Df Sum Sq Mean Sq F value   Pr(>F)
Task                1  96.33   96.33  39.862 3.87e-05 ***
Task:Gender         1   1.33    1.33   0.552    0.472
Task:Dosage         2   8.17    4.08   1.690    0.226
Task:Gender:Dosage  2   3.17    1.58   0.655    0.537
Residuals          12  29.00    2.42
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
+ lots more
```

Analysis of variance within subjects output (continued)

```
Error: Subject:Valence
                       Df Sum Sq Mean Sq F value Pr(>F)
Valence                 2  14.69   7.343   2.998 0.0688 .
Valence:Gender          2   3.91   1.954   0.798 0.4619
Valence:Dosage          4  20.26   5.065   2.068 0.1166
Valence:Gender:Dosage   4   1.04   0.259   0.106 0.9793
Residuals              24  58.78   2.449
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Error: Subject:Task:Valence
                            Df Sum Sq Mean Sq F value Pr(>F)
Task:Valence                 2   5.39  2.6944   1.320  0.286
Task:Valence:Gender          2   2.17  1.0833   0.531  0.595
Task:Valence:Dosage          4   2.78  0.6944   0.340  0.848
Task:Valence:Gender:Dosage   4   2.67  0.6667   0.327  0.857
Residuals                   24  49.00  2.0417
```

Multiple regression

1. Use the Garcia data set from *psych* (protest in Hayes (2013))
2. Do the linear model. (See the Garcia example)
3. Summarize the results

```
mod1 <- lm(respappr ~ prot2 + sexism ,data=Garcia)
 summary(mod1)
Call:
lm(formula = respappr ~ prot2 + sexism, data = Garcia)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5636 -0.8091  0.1281  0.9028  2.3069

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.69100    0.69826   5.286 5.33e-07 ***
prot2        1.43715    0.22273   6.452 2.15e-09 ***
sexism       0.03809    0.13284   0.287    0.775
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 1.177 on 126 degrees of freedom
```

However, zero center the data before examining interactions

```
Garcia.centered<- data.frame(scale(Garcia,scale=FALSE))
mod2 <-  lm( respappr ~ prot2 * sexism + sexism, data=Garcia.centered)
 summary(mod2)


Call:
lm(formula = respappr ~ prot2 * sexism + sexism, data = Garcia.centere

Residuals:
    Min      1Q  Median      3Q     Max
-3.4984 -0.7540  0.0801  0.8301  3.1853

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.01184    0.10085  -0.117  0.90671
prot2         1.45803    0.21670   6.728 5.52e-10 ***
sexism        0.02354    0.12927   0.182  0.85579
prot2:sexism  0.80998    0.28191   2.873  0.00478 **
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 1.144 on 125 degrees of freedom
Multiple R-squared:  0.2962,       Adjusted R-squared:  0.2793
```

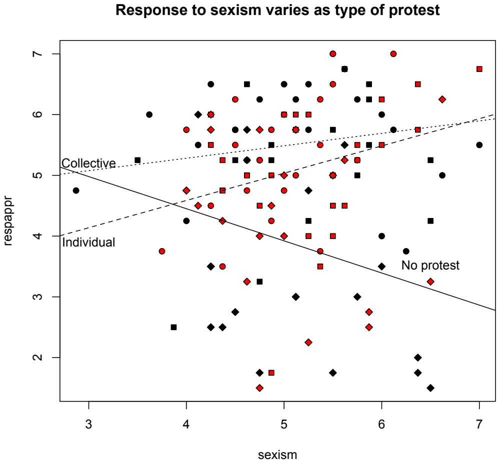
Compare model 1 and model 2

Test the difference between the two linear models

anova(mod1,mod2)

```
Analysis of Variance Table
```
Analysis of Variance Table
```
Model 1: respappr ~ prot2 + sexism
Model 2: respappr ~ prot2 * sexism + sexism
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1    126 174.54
2    125 163.73  1    10.813 8.2551 0.004776 **
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.
```

Show the regression lines by protest




```
plot(respappr ~ sexism, pch = 23-
 data=Garcia, main = "Response to
by(Garcia,Garcia$protest, functio
  data =x),lty=c("solid","dashed",
text(6.5,3.5,"No protest")
 text(3,3.9,"Individual")
  text(3,5.2,"Collective")
```

(See Examples: Garcia)

1. Although most regression examples use the raw data, it is also possible to do this from the correlation/covariance matrices.
2. This is particularly useful for analyzing text book examples or data sets that come from synthetic covariance matrices (SAPA data).
3. Two functions do this
   3.1 `setCor` will find (and draw the paths) between a set of X variables and a set of Y variables from either the raw data or from a correlation matrix.
   3.2 `mediate` will show path diagrams in a way to highlight "mediated" (indirect) and direct effects. The significance of the indirect effect is found by bootstrapped confidence intervals
4. Both of these functions just use the standard matrix equation $\beta xy = \mathbf{R}^{-1} r_{xy}$
5. The two examples are taken from the PMI example in Hayes (2013) which is saved as the `Tal_Or` dataset and used in the `mediate` help file.

setCor finds regressions from covariances

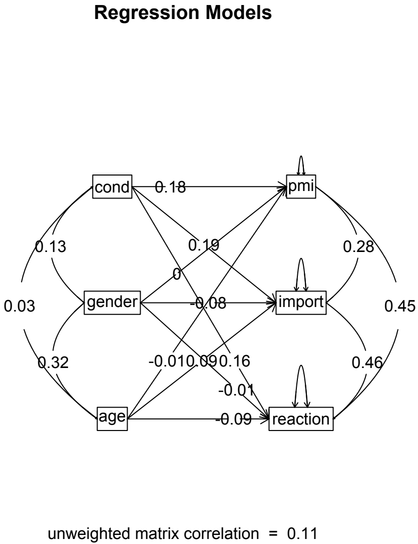
R code
```
C.pmi <- cov(Tal_Or} #just to show how we can do this
lowerMat(C.pmi) #show it
setCor(2:4,c(1,5,6),data=C.pmi)
```

```
> lowerCor(Tal_Or)
         cond  pmi   imprt rectn gendr age
cond      0.25
pmi       0.12  1.75
import    0.16  0.65  3.02
reaction  0.12  0.91  1.25  2.40
gender    0.03  0.01 -0.02 -0.01  0.23
age       0.07 -0.04  0.74 -0.75  0.88 33.65
```

```
Multiple Regression from matrix input

Beta weights
         pmi import reaction
cond    0.18   0.19     0.16
gender  0.00  -0.08    -0.01
age    -0.01   0.09    -0.09

Multiple R
     pmi   import reaction
    0.18    0.21     0.18
multiple R2
     pmi   import reaction
   0.033   0.043    0.033
```

(Specify n.obs if you want the standard errors , t, and probabilities of the estimates.)

Regressions from a covariance matrix



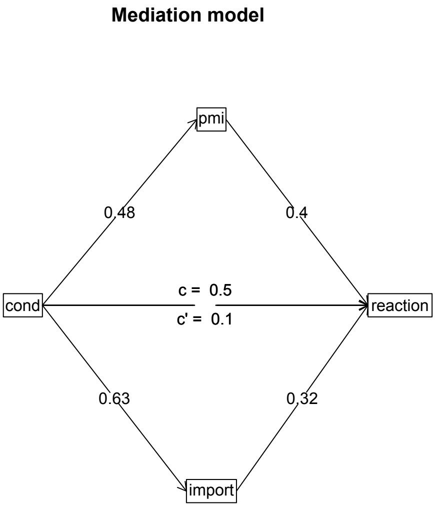
A mediation example from Tal-Or (2010) included in Hayes (2013)

R code

```
mediate(y="reaction",x = "cond",m=c("pmi","import"),data=pmi,n.obs=123,n.iter=5000)
```

```
Call: mediate(y = "reaction", x = "cond", m = c("pmi", "import"), data = C.pmi,
    n.obs = 123, n.iter = 50)
The DV (Y) was  reaction . The IV (X) was  cond . The mediating variable(s) =  pmi import .
Total Direct effect(c) of  cond  on  reaction  =  0.5   S.E. =  0.28  t direct =  1.79   with
Direct effect (c') of  cond  on  reaction  removing  pmi import  =  0.1   S.E. =  0.24  t dire
Indirect effect (ab) of  cond  on  reaction  through  pmi import   =  0.39
Mean bootstrapped indirect effect =  0.4  with standard error =  0.13  Lower CI =  0.19    Upp
R2 of model =  0.33
 To see the longer output, specify short = FALSE in the print statement
 Full output
 Total effect estimates (c)
     reaction   se    t   Prob
cond      0.5 0.28 1.79 0.0766
Direct effect estimates     (c')
       reaction   se    t     Prob
cond       0.10 0.24 0.43 6.66e-01
pmi        0.40 0.09 4.26 4.04e-05
import     0.32 0.07 4.59 1.13e-05
 'a'  effect estimates
       cond   se    t   Prob
pmi    0.48 0.24 2.02 0.0452
import 0.63 0.31 2.02 0.0452
 'b'  effect estimates
       reaction   se    t     Prob
pmi        0.40 0.09 4.26 4.04e-05
import     0.32 0.07 4.59 1.13e-05
 'ab'  effect estimates
     reaction boot   sd lower upper
cond     0.39  0.4 0.13  0.19  0.63
```

A mediation example from Hayes (2013)



A brief technical interlude

1. Data structures
   - The basic: scalers, vectors, matrices
   - More advanced data frames and lists
   - Showing the data
2. Getting the length, dimensions and structure of a data structure
   - length(x), dim(x), str(x)
3. Objects and Functions
   - Functions act upon objects
   - Functions actually are objects themselves
   - Getting help for a function (?function) or ?? function
4. Vignettes for help on the entire package (available either as part of the help file, or as a web page supplement to the package).

The basic types of data structures

1. Scalers (characters, integers, reals, complex)

```
> A <- 1     #Assign the value 1 to the object A
> B <- 2     #Assign the value 2 to the object B
```

2. Vectors (of scalers, all of one type) have `length`

```
> C <- month.name[1:5]  #Assign the names of the first 5 months to
> D <- 12:24   #assign the numbers 12 to 24 to D
> length(D)      #how many numbers are in D?

[1] 13
```

3. Matrices (all of one type) have `dimensions`

```
> E <- matrix(1:20, ncol = 4)
> dim(E)  #number of rows and columns of E

[1] 5 4
```

Show values by entering the variable name

```
> A      #what is the value of A?
[1] 1
> B     #and of B?
[1] 2
> C    #and C
[1] "January"  "February" "March"     "April"     "May"
> D
 [1] 12 13 14 15 16 17 18 19 20 21 22 23 24
> E
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

More complicated (and useful) types: Data frames and Lists

1. Data frames are collections of vectors and may be of different type. They have two dimensions.

```
> E.df <- data.frame(names = C, values = c(31, 28, 31, 30, 31))
> dim(E.df)

[1] 5 2
```

2. Lists are collections of what ever you want. They have length, but do not have dimensions.

```
> F <- list(first = A, a.vector = C, a.matrix = E)
> length(F)

[1] 3
```

Show values by entering the variable name

```
> E.df
     names values
1  January     31
2 February     28
3    March     31
4    April     30
5      May     31
> F
$first
[1] 1

$a.vector
[1] "January"  "February" "March"    "April"    "May"

$a.matrix
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

1. To show the structure of a list, use `str`

```
> str(F)

List of 3
 $ first   : num 1
 $ a.vector: chr [1:5] "January" "February" "March" "April" ...
 $ a.matrix: int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
```

2. To address an element of a list, call it by name or number, to get a row or column of a matrix specify the row, column or both.

```
> F[[2]]

[1] "January"  "February" "March"     "April"

> F[["a.matrix"]][, 2]

[1]  6  7  8  9 10

> F[["a.matrix"]][2, ]

[1]  2  7 12 17
```

Addressing the elements of a data.frame or matrix

Setting row and column names using paste

```
> E <- matrix(1:20, ncol = 4)
> colnames(E) <- paste("C", 1:ncol(E), sep = "")
> rownames(E) <- paste("R", 1:nrow(E), sep = "")
> E
   C1 C2 C3 C4
R1  1  6 11 16
R2  2  7 12 17
R3  3  8 13 18
R4  4  9 14 19
R5  5 10 15 20
> E["R2", ]
C1 C2 C3 C4
 2  7 12 17
> E[, 3:4]
   C3 C4
R1 11 16
R2 12 17
R3 13 18
R4 14 19
R5 15 20
```

Objects and Functions

1. R is a collection of Functions that act upon and return Objects
2. Although most functions can act on an object and return an object (a =f(b)), some are binary operators
   - primitive arithmetic functions +, -, * , /, %*%, ^
   - logical functions <, > ,==, !=
3. Some functions return "invisible" values
   - e.g., p <- `print(x,digits=3)` will print out x to 3 digits but also returns a value to p.
   - Similarly, s <- `summary(some object)` will return the value of the `summary` function.
4. But most useful functions act on an object and return a resulting object
   - This allows for extraordinary power because you can combine functions by making the output of one the input of the next.
   - The number of R functions is very large, for each package has introduced more functions, but for any one task, not many functions need to be learned. Keep a list of the ones you use.

Getting help

1. All functions have a help menu
   - `help(the function)` or just ? the function
   - Most function help pages have examples to show how to use the function
2. Many packages have "vignettes" that give overviews of all the functions in the package and are somewhat more readable than the help for a specific function.
   - The examples are longer, somewhat more readable. (e.g., the vignettea for *psych* is available either from the menu (Mac) or from http://cran.r-project.org/web/packages/psych/vignettes/overview.pdf
3. To find a function in the entire R space, use `findFn` in the *sos* package.
4. Online tutorials (e.g.,http://Rpad.org for a list of important commands, http://personality-project.org/r) for a tutorial for psychologists.
5. Online and hard copy books

A few of the most useful data manipulations functions (adapted from Rpad-refcard). Use ? for details

file.choose	() find a file
file.choose	(new=TRUE) create a new file
read.table	(filename)
read.csv	(filename) reads a comma separated file
read.delim	(filename) reads a tab delimited file
c	(...) combine arguments
from:to	e.g., 4:8
seq	(from,to, by)
rep	(x,times,each) repeat x
gl	(n,k,...) generate factor levels
matrix	(x,nrow=,ncol=) create a matrix
data.frame	(...) create a data frame

dim	(x) dimensions of x
str	(x) Structure of an object
list	(...) create a list
colnames	(x) set or find column names
rownames	(x) set or find row names
ncol(x), nrow(x)	number of row, columns
rbind	(...) combine by rows
cbind	(...) combine by columns
is.na	(x) also is.null(x), is...
na.omit	(x) ignore missing data
table	(x)
merge	(x,y)
apply	(x,rc,FUNCTION)
ls	() show workspace
rm	() remove variables from workspace

More useful statistical functions, Use ? for details

mean	(x)
is.na	(x) also is.null(x), is...
na.omit	(x) ignore missing data
sum	(x)
rowSums	(x) see also colSums(x)
min	(x)
max	(x)
range	(x)
table	(x)
summary	(x) depends upon x
sd	(x) standard deviation
cor	(x) correlation
cov	(x) covariance
solve	(x) inverse of x
lm	`(y~x) linear model`
aov	`(y~x) ANOVA`

Selected functions from *psych* package

describe	(x) descriptive stats
describeBy	(x,y) descriptives by group
pairs.panels	(x) SPLOM
error.bars	(x) means + error bars
error.bars.by	(x) Error bars by groups
fa	(x,n) Factor analysis
principal	(x,n) Principal components
iclust	(x) Item cluster analysis
scoreItems	(x) score multiple scales
score.multiple.choice	(x) score multiple choice scales
alpha	(x) Cronbach's alpha
omega	(x) MacDonald's omega
irt.fa	(x) Item response theory through factor analysis
bestScales	empirical scale construction

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 IV: Psychometrics

-> Part III: Basic Statistics

Classical Test Theory measures of reliability
- Split Half Reliability and α
- Multiple Scales

Multivariate Analysis and Structural Equation Modeling
- Exploratory Factor Analysis
- Confirmatory Factor Analysis and Structural Equation Modeling

Item Response Theory
- Multiple programs
- IRT from factor analysis: the irt.fa function in psych

–> Part V: More help

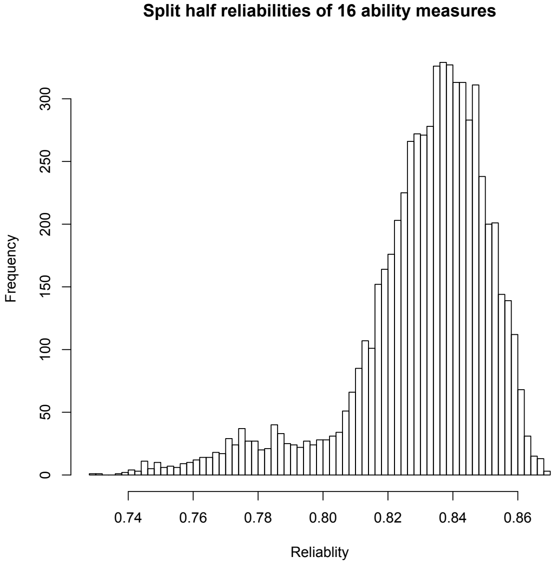
Psychometrics

1. Classical test theory measures of reliability
   - Scoring tests
   - Reliability (alpha, beta, omega)
2. Multivariate Analysis
   - Factor Analysis
   - Components analysis
   - Multidimensional scaling
   - Structural Equation Modeling
3. Item Response Theory
   - One parameter (Rasch) models
   - 2PL and 2PN models

Classical Test Theory estimates of reliability

1. Alternative estimates of reliability
   - `alpha` α reliability of a single scale finds the average split half reliability. (some items may be reversed keyed).
   - `omega` ω_h reliability of a single scale estimates the general factor saturation of the test.
   - `guttman` Find the 6 Guttman reliability estimates
   - `splitHalf` Find the range of split half reliabilities
2. Scoring tests with multiple scales
   - `scoreItems` Score 1 ... n scales using a set of keys and finding the simple sum or average of items. Reversed items are indicated by -1
   - `score.multiple.choice` Score multiple choice items by first converting to 0 or 1 and then proceeding to score the items.

6,435 split half reliabilities of a 16 item ability test



R code

```
sp <- splitHalf(ability,
     raw=TRUE, brute=TRUE)
hist(sp$raw,breaks=50)
```

Finding coefficient α for a scale (see Revelle and Zinbarg, 2009, however, for why you should not)

R code

```
alpha(ability)
```

```
Reliability analysis
Call: alpha(x = ability)

  raw_alpha std.alpha G6(smc) average_r S/N    ase mean   sd
      0.83      0.83    0.84      0.23 4.9 0.0086 0.51 0.25

 lower alpha upper     95% confidence boundaries
0.81 0.83 0.85

 Reliability if an item is dropped:
          raw_alpha std.alpha G6(smc) average_r S/N alpha se
reason.4       0.82      0.82    0.82      0.23 4.5   0.0093
reason.16      0.82      0.82    0.83      0.24 4.7   0.0091
...
rotate.6       0.82      0.82    0.82      0.23 4.5   0.0092
rotate.8       0.82      0.82    0.83      0.24 4.6   0.0091

 Item statistics
             n    r r.cor r.drop mean   sd
reason.4  1442 0.58  0.54   0.50 0.68 0.47
reason.16 1463 0.50  0.44   0.41 0.73 0.45
r...
rotate.4  1460 0.58  0.56   0.48 0.22 0.42
rotate.6  1456 0.56  0.53   0.46 0.31 0.46
rotate.8  1460 0.51  0.47   0.41 0.19 0.39
```

Using scoreItems to score 25 Big 5 items (see bfi example)

R code

```
 keys.list <- list(Agree=c(-1,2:5),Conscientious=c(6:8,-9,-10),
Extraversion=c(-11,-12,13:15), Neuroticism=c(16:20),Openness = c(21,-22,23,24,-25))
keys <- make.keys(bfi,keys.list)
scores <- scoreItems(keys,bfi)
```

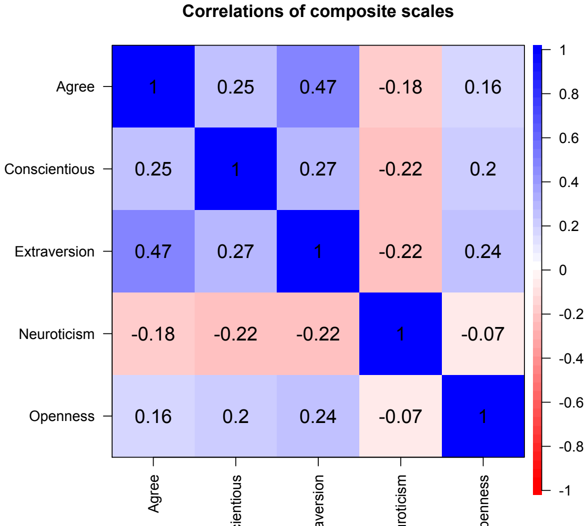
```
Call: score.items(keys = keys, items = bfi)

(Unstandardized) Alpha:
      Agree Conscientious Extraversion Neuroticism Openness
alpha   0.7          0.72         0.76        0.81      0.6

Average item correlation:
          Agree Conscientious Extraversion Neuroticism Openness
average.r  0.32          0.34         0.39        0.46     0.23

 Guttman 6* reliability:
         Agree Conscientious Extraversion Neuroticism Openness
Lambda.6   0.7          0.72         0.76        0.81      0.6

Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
              Agree Conscientious Extraversion Neuroticism Openness
Agree          0.70          0.36         0.63      -0.245     0.23
Conscientious  0.26          0.72         0.35      -0.305     0.30
Extraversion   0.46          0.26         0.76      -0.284     0.32
Neuroticism   -0.18         -0.23        -0.22       0.812    -0.12
Openness       0.15          0.19         0.22      -0.086     0.60
```

score.items output, continued

```
Item by scale correlations:
 corrected for item overlap and scale reliability
          Agree Conscientious Extraversion Neuroticism Openness
A1        -0.40         -0.06        -0.11        0.14    -0.14
A2         0.67          0.23         0.40       -0.07     0.17
A3         0.70          0.22         0.48       -0.11     0.17
A4         0.49          0.29         0.30       -0.14     0.01
A5         0.62          0.23         0.55       -0.23     0.18
C1         0.13          0.53         0.19       -0.08     0.28
C2         0.21          0.61         0.17        0.00     0.20
C3         0.21          0.54         0.14       -0.09     0.08
C4        -0.24         -0.66        -0.23        0.31    -0.23
C5        -0.26         -0.59        -0.29        0.36    -0.10
E1        -0.30         -0.06        -0.59        0.11    -0.16
E2        -0.39         -0.25        -0.70        0.34    -0.15
E3         0.44          0.20         0.60       -0.10     0.37
E4         0.51          0.23         0.68       -0.22     0.04
E5         0.34          0.40         0.55       -0.10     0.31
N1        -0.22         -0.21        -0.11        0.76    -0.12
N2        -0.22         -0.19        -0.12        0.74    -0.06
N3        -0.14         -0.20        -0.14        0.74    -0.03
N4        -0.22         -0.30        -0.39        0.62    -0.02
N5        -0.04         -0.14        -0.19        0.55    -0.18
O1         0.16          0.20         0.31       -0.09     0.52
O2        -0.01         -0.18        -0.07        0.19    -0.45
O3         0.26          0.20         0.42       -0.07     0.61
O4         0.06         -0.02        -0.10        0.21     0.32
O5        -0.09         -0.14        -0.11        0.11    -0.53
gender     0.25          0.11         0.12        0.14    -0.07
education  0.06          0.03         0.01       -0.06     0.13
age        0.22          0.14         0.07       -0.13     0.10
```

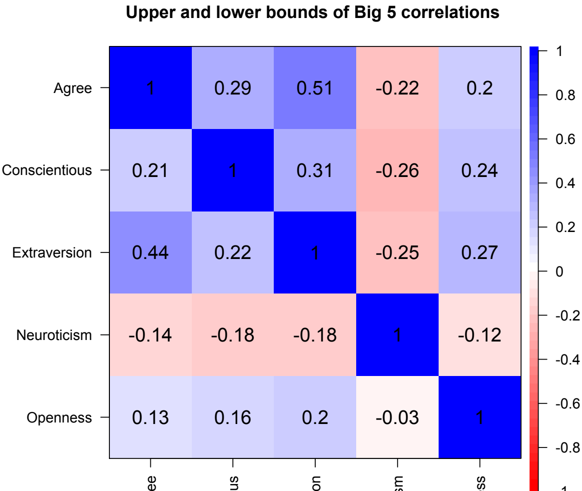
Correlations of composite scores based upon item correlations

```
ci <- cor.ci(bfi,keys=keys,main='Correlations of composite scales')
```



Upper and Lower bounds of Correlations of composite scores based upon item correlations and bootstrap resampling

cor.plot.upperLowerCi(ci,main='Upper and lower bounds of Big 5 correlations')



Factor analysis of Thurstone 9 variable problem

R code

```
f3 <- fa(Thurstone,nfactors=3)  #use this built in dataset
f3  #we keep the output as an object to use later
```

```
Factor Analysis using method =  minres
Call: fa(r = Thurstone, nfactors = 3)
Standardized loadings (pattern matrix) based upon correlation matrix
                 MR1   MR2   MR3   h2   u2 com
Sentences        0.91 -0.04  0.04 0.82 0.18 1.0
Vocabulary       0.89  0.06 -0.03 0.84 0.16 1.0
Sent.Completion  0.83  0.04  0.00 0.73 0.27 1.0
First.Letters    0.00  0.86  0.00 0.73 0.27 1.0
4.Letter.Words  -0.01  0.74  0.10 0.63 0.37 1.0
Suffixes         0.18  0.63 -0.08 0.50 0.50 1.2
Letter.Series    0.03 -0.01  0.84 0.72 0.28 1.0
Pedigrees        0.37 -0.05  0.47 0.50 0.50 1.9
Letter.Group    -0.06  0.21  0.64 0.53 0.47 1.2

                       MR1  MR2  MR3
SS loadings           2.64 1.86 1.50
Proportion Var        0.29 0.21 0.17
Cumulative Var        0.29 0.50 0.67
Proportion Explained  0.44 0.31 0.25
Cumulative Proportion 0.44 0.75 1.00

 With factor correlations of
     MR1  MR2  MR3
MR1 1.00 0.59 0.54
MR2 0.59 1.00 0.52
MR3 0.54 0.52 1.00
```

Factor analysis output, continued

```
 With factor correlations of
     MR1  MR2  MR3
MR1 1.00 0.59 0.54
MR2 0.59 1.00 0.52
MR3 0.54 0.52 1.00

Mean item complexity =  1.2
Test of the hypothesis that 3 factors are sufficient.

The degrees of freedom for the null model are  36  and the objective function was  5.2
The degrees of freedom for the model are 12  and the objective function was  0.01

The root mean square of the residuals (RMSR) is  0.01
The df corrected root mean square of the residuals is  0.01

Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                 MR1  MR2  MR3
Correlation of scores with factors              0.96 0.92 0.90
Multiple R square of scores with factors        0.93 0.85 0.81
Minimum correlation of possible factor scores   0.86 0.71 0.63
```

Bootstrapped confidence intervals

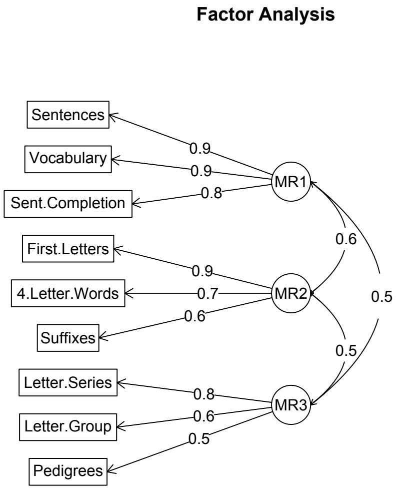
R code

```
fa(Thurstone,3,n.obs=213,n.iter=20)    #to do bootstrapping
```

```
...
Coefficients and bootstrapped confidence intervals
                  low   MR1 upper   low   MR2 upper   low   MR3 upp
Sentences         0.83  0.91  0.97 -0.10 -0.04  0.06 -0.02  0.04  0.
Vocabulary        0.80  0.89  0.98  0.00  0.06  0.15 -0.12 -0.03  0.
Sent.Completion   0.75  0.83  0.90 -0.05  0.04  0.11 -0.08  0.00  0.
First.Letters    -0.08  0.00  0.09  0.68  0.86  0.97 -0.09  0.00  0.
4.Letter.Words   -0.13 -0.01  0.12  0.57  0.74  0.90 -0.01  0.10  0.
Suffixes          0.07  0.18  0.26  0.50  0.63  0.76 -0.23 -0.08  0.
Letter.Series    -0.09  0.03  0.13 -0.06 -0.01  0.08  0.68  0.84  0.
Pedigrees         0.27  0.37  0.52 -0.17 -0.05  0.04  0.33  0.47  0.
Letter.Group     -0.16 -0.06  0.08  0.12  0.21  0.29  0.41  0.64  0.

 Interfactor correlations and bootstrapped confidence intervals
        lower estimate upper
MR1-MR2  0.47     0.59  0.68
MR1-MR3  0.39     0.54  0.61
MR2-MR3  0.30     0.52  0.64
```

The simple factor structure

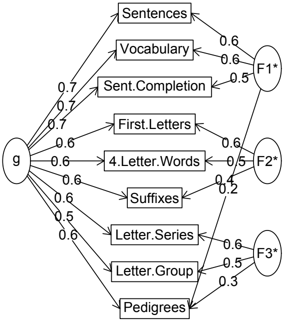
factor.diagram(f3) # show the diagram



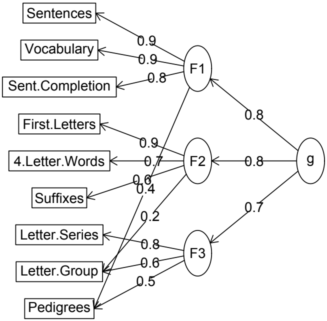
Two ways of viewing the higher order structure

om <- omega(Thurstone) omega.diagram(om,sl=FALSE)

Omega

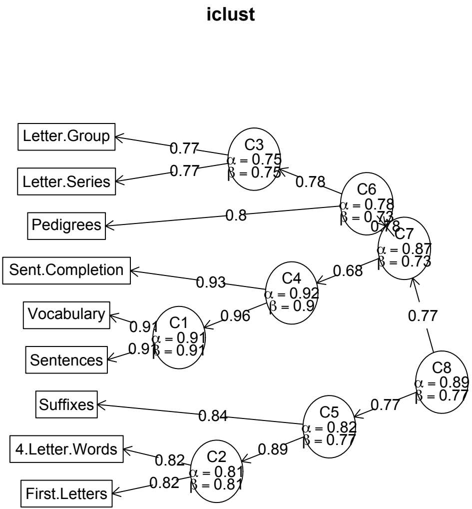


Hierarchical (multilevel) Structure



A hierarchical cluster structure found by `iclust`

iclust(Thurstone)



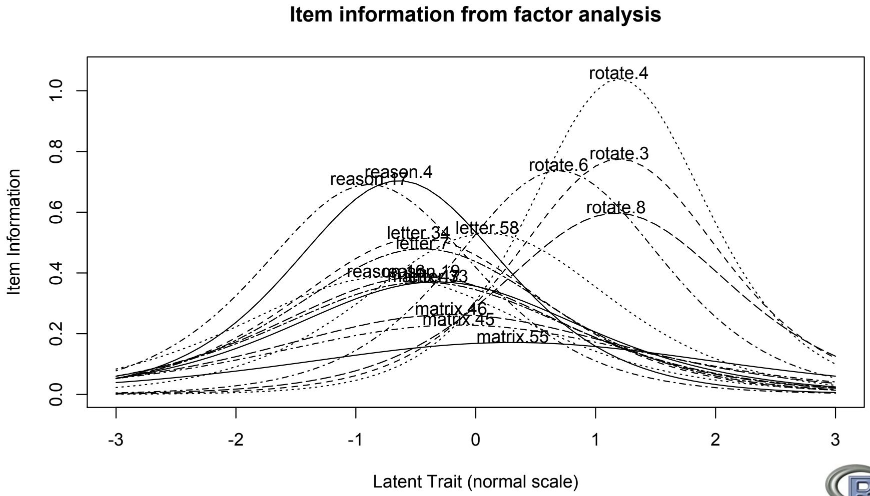
Structural Equation modeling packages

1. sem (Fox, Nie & Byrnes, 2013)
   - uses RAM notation
2. lavaan (Rosseel, 2012)
   - Mimics as much as possible MPLUS output
   - Allows for multiple groups
   - Easy syntax
3. OpenMx (Neale, Hunter, Pritikin, Zahery, Brick, Kickpatrick, Estabrook, Bates, Maes & Boker, 2016)
   - Open source and R version of Mx
   - Allows for multiple groups (and almost anything else)
   - Complicated syntax

Mutiple packages to do Item Response Theory analysis

1. *psych* uses a factor analytic procedure to estimate item discriminations and locations
   - `irt.fa` finds either tetrachoric or polychoric correlation matrices
     - converts factor loadings to discriminations
   - `plot.irt` plots item information and item characteristic functions
   - look at examples for `irt.fa`
   - two example data sets: `ability` and `bfi`
2. Other packages to do more conventional IRT include *ltm*, *eRm*, *mirt*, + others

Item Response Information curves for 16 ability items from ICAR



Questions?

A few of the most useful data manipulations functions (adapted from Rpad-refcard). Use ? for details

file.choose	() find a file
file.choose	(new=TRUE) create a new file
read.table	(filename)
read.csv	(filename) reads a comma separated file
read.delim	(filename) reads a tab delimited file
c	(...) combine arguments
from:to	e.g., 4:8
seq	(from,to, by)
rep	(x,times,each) repeat x
gl	(n,k,...) generate factor levels
matrix	(x,nrow=,ncol=) create a matrix
data.frame	(...) create a data frame

dim	(x) dimensions of x
str	(x) Structure of an object
list	(...) create a list
colnames	(x) set or find column names
rownames	(x) set or find row names
ncol(x), nrow(z)	number of row, columns
rbind	(...) combine by rows
cbind	(...) combine by columns
is.na	(x) also is.null(x), is...
na.omit	(x) ignore missing data
table	(x)
merge	(x,y)
apply	(x,rc,FUNCTION)
ls	() show workspace
rm	() remove variables from workspace

More useful statistical functions, Use ? for details

mean	(x,na.rm=TRUE) *
is.na	(x) also is.null(x), is...
na.omit	(x) ignore missing data
sum	(x)
rowSums	(x) see also colSums(x)
colSums	(x) see also rowSums(x)
min	(x,na.rm=TRUE)*
max	(x) *ignores NA values
range	(x)
table	(x)
summary	(x) depends upon x
sd	(x) standard deviation
cor	(x,use="pairwise") correlation
cov	(x) covariance
solve	(x) inverse of x
lm	`(y~x) linear model`
aov	`(y~x) ANOVA`

Selected functions from *psych* package

describe	(x) descriptive stats
describeBy	(x,y) descriptives by group
pairs.panels	(x) SPLOM
error.bars	(x) means + error bars
error.bars.by	(x) Error bars by groups
fa	(x,n) Factor analysis
principal	(x,n) Principal components
iclust	(x) Item cluster analysis
scoreItems	(x) score multiple scales
score.multiple.choice	(x) score multiple choice scales
alpha	(x) Cronbach's alpha
omega	(x) MacDonald's omega
irt.fa	(x) Item response theory through factor analysis
mediate	(y,x,m,data) Mediation/moderation

More help

1. An introduction to R as HTML, PDF or EPUB from `http://cran.r-project.org/manuals.html` (many different links on this page
2. FAQ General and then Mac and PC specific
3. R reference card `http://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf`
4. Various "cheat sheets" from RStudio `http://www.rstudio.com/resources/cheatsheets/`
5. Using R for psychology `http://personality-project.org/r/`
6. Package vignettes (e.g., `http://personality-project.org/r/psych/vignettes/overview.pdf`)
7. R listserve, StackOverflow, your students and colleagues

An introduction to the psych package

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

The psych package

1. Developed at NU over the past 12 years to make using R easier for psychologists
2. Basically does the kind of statistics that my students and I find useful for personality, motivation and cognitive psychology
3. Available at CRAN for PCs and Macs
4. Development version (for Macs) is always available at the http://personality-project.org/r repository.
5. Bugs are fixed and new versions with new toys (functions) are released about every 4-6 months.
6. Version number reflects the year and month of release (1.8.4)
7. Has several vignettes to describe what it does:
   - http://personality-project.org/r/psych/vignettes/intro.pdf An introduction
   - http://personality-project.org/r/psych/vignettes/overview.pdf An overview
   - http://personality-project.org/r/psych/vignettes/psych_for_sem.pdf as a front end to doing sem

Show all the functions in the psych package

objects("package:psych")

```
objects("package:psych")
  [1] "%+%"                     "ability"                 "affect"                  "all.income"
  [5] "alpha"                   "anova.psych"             "autoR"                   "Bechtoldt"
  ...
[49] "cohen.kappa"            "comorbidity"            "con2cat"                "congeneric.s
 [53] "cor.ci"                  "cor.plot"                "cor.plot.upperLowerCi"   "cor.smooth"
 ...
 [81] "cushny"                  "d2r"                     "densityBy"               "describe"
...
[109] "epi.dictionary"          "equamax"                 "error.bars"              "error.bars.l
...
[177] "ICC2latex"               "iclust"                  "ICLUST"                  "ICLUST.clust
...
[201] "irt.fa"                  "irt.item.diff.rasch"     "irt.person.rasch"        "irt.response
...
[241] "mixed.cor"               "mixedCor"                "mlArrange"               "mlPlot"
...
[253] "omega"                   "omega.diagram"           "omega.graph"             "omega2latex'
...
[309] "read.clipboard.upper"    "read.file"               "read.file.csv"           "read.https"
...
[329] "score.alpha"             "score.irt"               "score.irt.2"             "score.irt.po
[333] "score.items"             "score.multiple.choice"   "scoreFast"               "scoreIrt"
...
[405] "Thurstone"               "Thurstone.33"            "topBottom"               "tr"
[409] "Tucker"                  "unidim"                  "varimin"                 "veg"
...
```

Get your data: using `read.file` or `read.clipboard`

From a website: define the file name

R code
```
fn <- "http://personality-project.org/r/datasets/Maps.mixx.msq1.epi.bf.txt"
fn  #show it to check
[1] "http://personality-project.org/r/datasets/Maps.mixx.msq1.epi.bf.txt"
mydata <- read.file(fn,header=TRUE)
```

From a local file: find fhe file using `read.file`

R code
```
> my.data <- read.file() #will open a search window, read the file
#depending upon the suffix, will read .sav, .csv, .txt, .rds,
         .rDa, etc.
```

From the clipboard: (first, go to the remote site, copy to the clipboard and then use the `read.clipboard` function).

R code
```
mydata <- read.clipboard()     #or
mydata <- read.clipboard.tab()  #if an excel file
my.data <- read.clipboard.csv() #if a tab delimited file
```

Checking the data using `describe`

```
> dim(mydata)
[1] 231  86
> describe(mydata)
             vars   n  mean    sd median trimmed   mad min max range  skew kurtosis   se
id              1 231 66.82 45.13     58   64.14 50.41   1 160   159  0.45    -0.96 2.97
delighted       2 231  0.82  1.05      1    0.68  1.48   0   9     9  2.46    14.15 0.07
sociable        3 231  1.32  0.96      1    1.28  1.48   0   3     3  0.06    -1.03 0.06
jittery         4 231  0.55  0.78      0    0.39  0.00   0   3     3  1.37     1.23 0.05
hostile         5 231  0.35  0.85      0    0.17  0.00   0   9     9  5.34    45.21 0.06
sluggish        6 231  1.21  0.96      1    1.14  1.48   0   3     3  0.47    -0.70 0.06
depressed       7 231  0.56  0.83      0    0.39  0.00   0   3     3  1.45     1.37 0.05
...
ashamed        71 231  0.32  1.15      0    0.06  0.00   0   9     9  5.92    40.25 0.08
anxious        72 231  0.75  1.26      0    0.53  0.00   0   9     9  3.85    21.39 0.08
idle           73 231  0.98  1.15      1    0.83  1.48   0   9     9  3.11    18.20 0.08
epiE           74 231 13.33  4.14     14   13.49  4.45   1  22    21 -0.33    -0.06 0.27
epiS           75 231  7.58  2.69      8    7.77  2.97   0  13    13 -0.57    -0.02 0.18
epiImp         76 231  4.37  1.88      4    4.36  1.48   0   9     9  0.06    -0.62 0.12
epilie         77 231  2.38  1.50      2    2.27  1.48   0   7     7  0.66     0.24 0.10
...
traitanx       85 231 39.01  9.52     38   38.36  8.90  22  71    49  0.67     0.47 0.63
stateanx       86 231 39.85 11.48     38   38.92 10.38  21  79    58  0.72    -0.01 0.76
```

Cleaning the data using `scrub`

We want to change 9s in variables 2 - 73 into NA

```
> cleaned <- scrub(mydata,where=2:73,isvalue=9,newvalue=NA)
> describe(cleaned)
              vars   n  mean    sd median trimmed   mad min max range  skew kurtosis   se
id               1 231 66.82 45.13   58.0   64.14 50.41   1 160   159  0.45    -0.96 2.97
delighted        2 230  0.78  0.90    1.0    0.67  1.48   0   3     3  0.79    -0.52 0.06
sociable         3 231  1.32  0.96    1.0    1.28  1.48   0   3     3  0.06    -1.03 0.06
jittery          4 231  0.55  0.78    0.0    0.39  0.00   0   3     3  1.37     1.23 0.05
hostile          5 230  0.31  0.63    0.0    0.17  0.00   0   3     3  2.12     4.19 0.04
sluggish         6 231  1.21  0.96    1.0    1.14  1.48   0   3     3  0.47    -0.70 0.06
depressed        7 231  0.56  0.83    0.0    0.39  0.00   0   3     3  1.45     1.37 0.05
...
ashamed         71 228  0.21  0.57    0.0    0.05  0.00   0   3     3  3.00     8.95 0.04
anxious         72 228  0.64  0.84    0.0    0.51  0.00   0   3     3  1.11     0.32 0.06
idle            73 229  0.91  0.88    1.0    0.82  1.48   0   3     3  0.64    -0.42 0.06
epiE            74 231 13.33  4.14   14.0   13.49  4.45   1  22    21 -0.33    -0.06 0.27
epiS            75 231  7.58  2.69    8.0    7.77  2.97   0  13    13 -0.57    -0.02 0.18
epiImp          76 231  4.37  1.88    4.0    4.36  1.48   0   9     9  0.06    -0.62 0.12
epilie          77 231  2.38  1.50    2.0    2.27  1.48   0   7     7  0.66     0.24 0.10
...
traitanx        85 231 39.01  9.52   38.0   38.36  8.90  22  71    49  0.67     0.47 0.63
stateanx        86 231 39.85 11.48   38.0   38.92 10.38  21  79    58  0.72    -0.01 0.76
```

Multiple ways to graphically display data

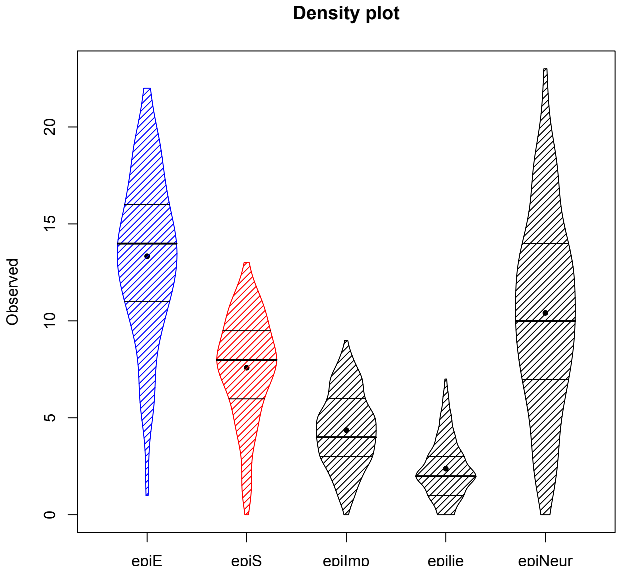
1. box.plots (Core R)
2. Violin plots (`violinBy` in *psych*)
3. Scatter Plot Matrix (SPLOM) plots (`pairs.panels` in *psych*)

First, lets just make a smaller data.frame and then issue two different graphic commands.

R code

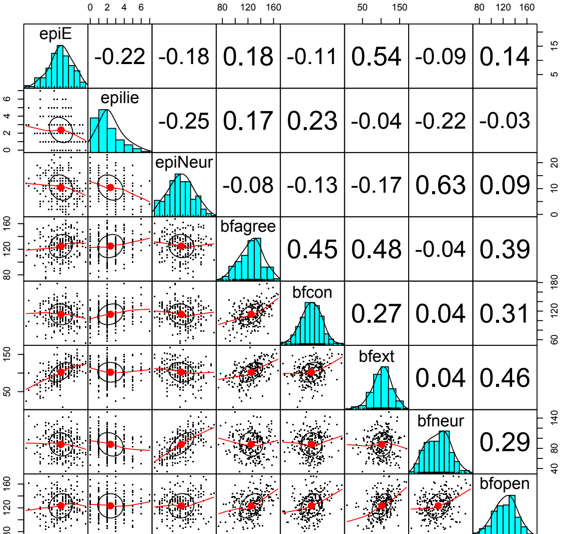
```
my.scales <- cleaned[74:86]
violinBy(my.scales[1:5])
pairs.panels(my.scales[c(1,4,5:10)],gap=0,pch=".")
```

Violin Plot `violinBy(my.scales[1:5])`



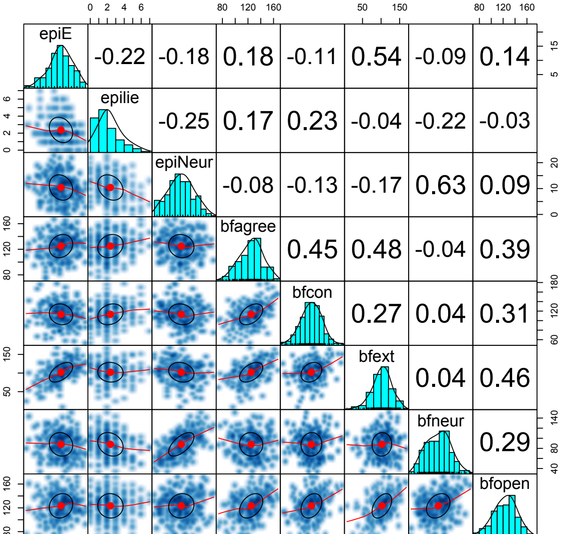
Scatter Plot of Matrices (SPLOM) of select variables

```
pairs.panels(my.scales[c(1,4,5:10)],gap=0,pch=".")
```



Scatter Plot of Matrices (SPLOM) of select variables

```
pairs.panels(my.scales[c(1,4,5:10)],gap=0,pch=".",smoother=TRU
```



Show a table of correlations

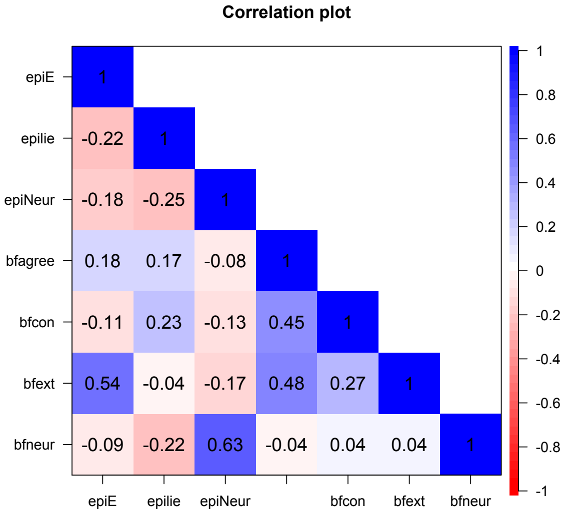
R code

```
R <- lowerCor(my.scales[c(1,4,5:10)])
cor.plot(R,numbers=TRUE)
```

```
         epiE  epili epiNr bfagr bfcon bfext bfner bfopn
epiE      1.00
epilie   -0.22  1.00
epiNeur  -0.18 -0.25  1.00
bfagree   0.18  0.17 -0.08  1.00
bfcon    -0.11  0.23 -0.13  0.45  1.00
bfext     0.54 -0.04 -0.17  0.48  0.27  1.00
bfneur   -0.09 -0.22  0.63 -0.04  0.04  0.04  1.00
bfopen    0.14 -0.03  0.09  0.39  0.31  0.46  0.29  1.00
```

Automatically calls the `cor` and `round` functions with default parameters and then does a pretty print out using `lowerMat`. Invisibly returns the full (square) matrix of unrounded values.

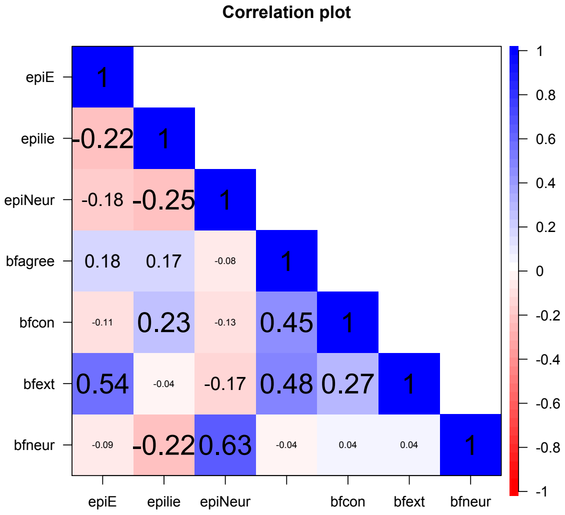
A simple heat map using `cor.plot`



R code

```
cor.plot(epi.bfi[c(1,4:9)],
numbers=TRUE,upper=FALSE,
scale=FALSE)
```

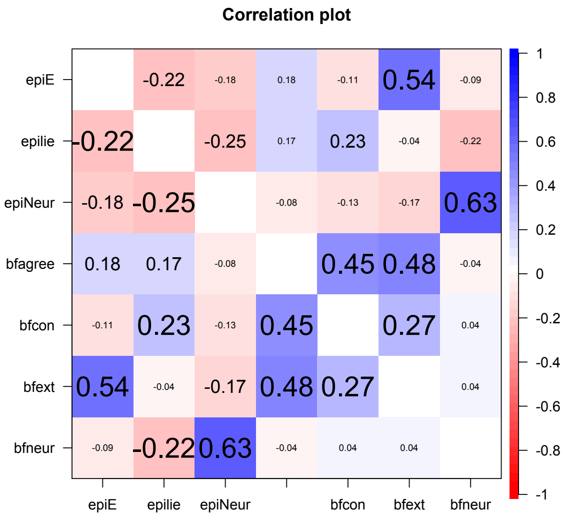
Scale the correlations using `cor.plot`



R code

```
cor.plot(epi.bfi[c(1,4:9)],
numbers=TRUE,upper=FALSE)
```

Show the whole matrix `cor.plot`



R code
```
cor.plot(epi.bfi[c(1,4:9)],
numbers=TRUE, diag=FALSE)
```

Testing and displaying the "significance" of a set of correlations

1. Normal theory `corr.test`
   - raw probabilities as well as with a Holm adjusted for multiple correlations
2. Display these with `cor.plot`
3. Boot strapped confidence intervals based significance using `cor.ci`
   - Graphic displays correlations scaled by "significance"
   - Graphic displays of probability of correlation using `plot.cor.upperLowerCi`

Code for the next slides

R code

```
my.scales <- epi.bfi[c(1,4:10)]
corr.test(my.scales)
ci <- cor.ci(my.scales)
cor.plot.upperLowerCi(ci)
```

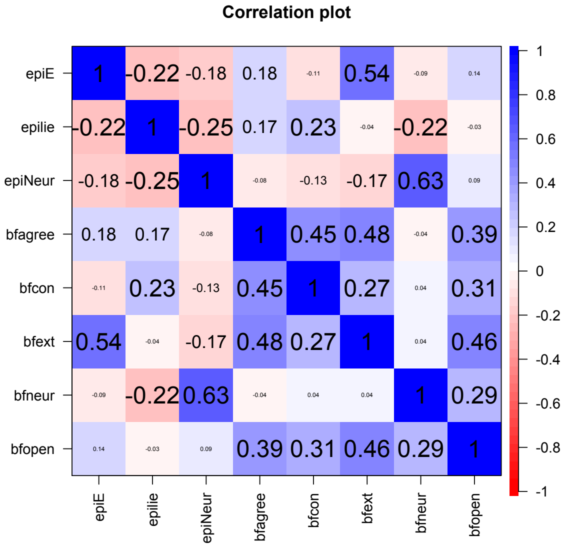
Normal theory test of correlations using `corr.test`

R code

```
corr.test(my.scales)
```

```
> corr.test(my.scales)
Call:corr.test(x = my.scales)
Correlation matrix
         epiE epilie epiNeur bfagree bfcon bfext bfneur bfopen
epiE     1.00  -0.22   -0.18    0.18 -0.11  0.54  -0.09   0.14
epilie  -0.22   1.00   -0.25    0.17  0.23 -0.04  -0.22  -0.03
epiNeur -0.18  -0.25    1.00   -0.08 -0.13 -0.17   0.63   0.09
bfagree  0.18   0.17   -0.08    1.00  0.45  0.48  -0.04   0.39
bfcon   -0.11   0.23   -0.13    0.45  1.00  0.27   0.04   0.31
bfext    0.54  -0.04   -0.17    0.48  0.27  1.00   0.04   0.46
bfneur  -0.09  -0.22    0.63   -0.04  0.04  0.04   1.00   0.29
bfopen   0.14  -0.03    0.09    0.39  0.31  0.46   0.29   1.00
Sample Size
[1] 231
Probability values (Entries above the diagonal are adjusted for multiple tests.)
        epiE epilie epiNeur bfagree bfcon bfext bfneur bfopen
epiE    0.00   0.01    0.11    0.11  0.75  0.00   1.00    0.4
epilie  0.00   0.00    0.00    0.12  0.01  1.00   0.01    1.0
epiNeur 0.01   0.00    0.00    1.00  0.43  0.12   0.00    1.0
bfagree 0.01   0.01    0.21    0.00  0.00  0.00   1.00    0.0
bfcon   0.08   0.00    0.04    0.00  0.00  0.00   1.00    0.0
bfext   0.00   0.50    0.01    0.00  0.00  0.00   1.00    0.0
bfneur  0.15   0.00    0.00    0.50  0.50  0.57   0.00    0.0
bfopen  0.04   0.70    0.19    0.00  0.00  0.00   0.00    0.0

 To see confidence intervals of the correlations, print with the short=FALSE option
```

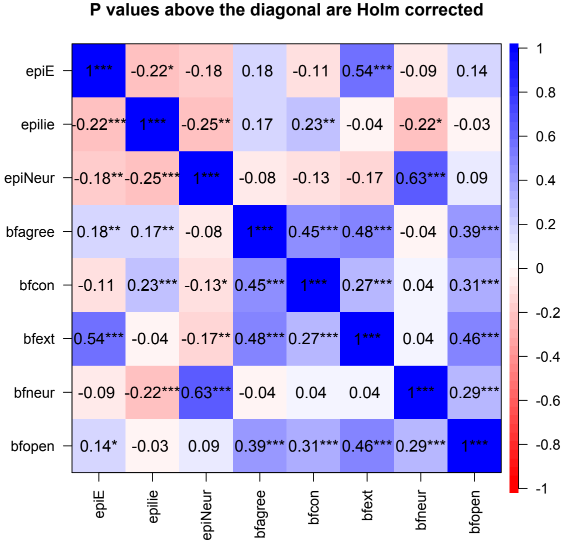
Heat map scaled by "significance" using `cor.ci`



R code

```
corPlot(ci,numbers=TRUE)
```

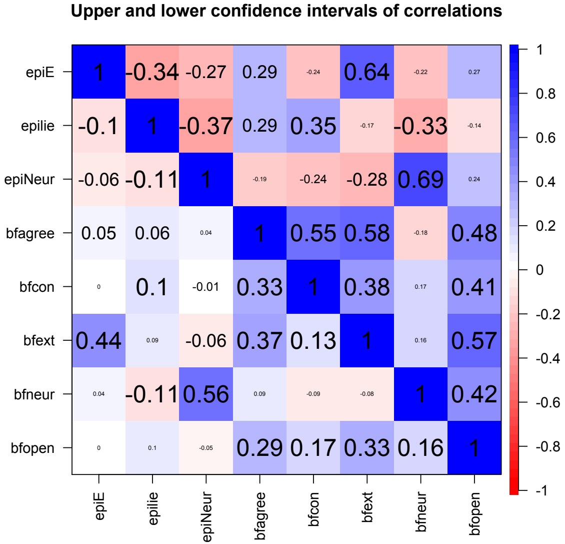
Heat map scaled by "significance" and showing magic asterisks



R code

```
corPlot(my.scales,stars=TRUE,
numbers=TRUE,xlas=3,
main="P values above the
 diagonal are Holm corrected")
```

Heat map scaled by "significance" and showing confidence intervals



R code

```
cor.plot.upperLowerCi(ci)
```

Multiple types of reliability

1. Internal consistency estimates
   - α, λ_6 , use the `alpha` function
   - $\omega_{hierarchical}$ and ω_{total} use the `omega` function
2. IntraClass coefficients
   - ICC
3. Rater agreement use `wkappa` function (finds Cohen's kappa and weighted kappa)
4. Multilevel reliability and generalizability, use `mlr` or `multilevel.reliability`

For the next examples we will use a built in data set

1. `bfi` consists of 25 personality items measuring 5 factors as well as some demographics.
2. The data were collected as part of the SAPA project and have 2,800 subjects.
3. For help on this data set, ?bfi
4. To see all of the *psych* data sets: `data(package="psych")`

First, we intentionally misspecify the data

R code

```
alpha(bfi[1:5]) #score the first five items
```

```
Some items ( A1 ) were negatively correlated with the total scale and
probably should be reversed.
To do this, run the function again with the 'check.keys=TRUE' option
Reliability analysis
Call: alpha(x = bfi[1:5])

  raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd median_r
      0.43      0.46    0.53      0.15 0.85 0.016  4.2 0.74     0.32

 lower alpha upper     95% confidence boundaries
0.4 0.43 0.46

 Reliability if an item is dropped:
   raw_alpha std.alpha G6(smc) average_r  S/N alpha se  var.r med.r
A1      0.72      0.73    0.67     0.398 2.64   0.0087 0.0065 0.376
A2      0.28      0.30    0.39     0.097 0.43   0.0219 0.1098 0.081
A3      0.18      0.21    0.31     0.061 0.26   0.0249 0.1015 0.081
A4      0.25      0.31    0.44     0.099 0.44   0.0229 0.1607 0.105
A5      0.21      0.24    0.36     0.072 0.31   0.0238 0.1311 0.095

 Item statistics

      n raw.r std.r r.cor r.drop mean  sd
A1 2784 0.066 0.024 -0.39  -0.31  2.4 1.4     <-- need to rekey this item
A2 2773 0.630 0.666  0.58   0.37  4.8 1.2
A3 2774 0.724 0.742  0.72   0.48  4.6 1.3
A4 2781 0.686 0.661  0.50   0.37  4.7 1.5
A5 2784 0.700 0.719  0.64   0.45  4.6 1.3
```

Try it again. Turn on automatic reversals. Get the scores

R code

```
scores <- alpha(bfi[1:5], check.keys =TRUE)
```

```
Reliability analysis
Call: alpha(x = bfi[1:5], check.keys = TRUE)

  raw_alpha std.alpha G6(smc) average_r S/N   ase mean  sd median_r
       0.7      0.71    0.68      0.33 2.5 0.009  4.7 0.9     0.34

 lower alpha upper     95% confidence boundaries
0.69 0.7 0.72

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r S/N alpha se  var.r med.r
A1-      0.72      0.73    0.67      0.40 2.6   0.0087 0.0065  0.38
A2       0.62      0.63    0.58      0.29 1.7   0.0119 0.0169  0.29
A3       0.60      0.61    0.56      0.28 1.6   0.0124 0.0094  0.32
A4       0.69      0.69    0.65      0.36 2.3   0.0098 0.0159  0.37
A5       0.64      0.66    0.61      0.32 1.9   0.0111 0.0126  0.34

 Item statistics
       n raw.r std.r r.cor r.drop mean  sd
A1- 2784  0.58  0.57  0.38   0.31  4.6 1.4
A2  2773  0.73  0.75  0.67   0.56  4.8 1.2
A3  2774  0.76  0.77  0.71   0.59  4.6 1.3
A4  2781  0.65  0.63  0.47   0.39  4.7 1.5
A5  2784  0.69  0.70  0.60   0.49  4.6 1.3
...

Warning message:
In alpha(bfi[1:5], check.keys = TRUE) :
```

R functions will return objects without necessarily telling you

1. The basic logic of R is that you can do lots of calculations, but you might not want all the output.
2. The output is there, to be processed by other functions if you want, but you probably don't want to see all of it unless you ask.
3. Thus, `alpha` returns the scores based upon the scales you asked for, but doesn't show them, because they are so many,
4. The `str` command tells you the structure of an object. The `names` will just list the names of the objects.

names and str of alpha output

R code

```
names(scores)
str(scores)
```

```
names(scores)
 [1] "total"          "alpha.drop"     "item.stats"     "response.freq" "keys"
  "scores"        "nvar"           "boot.ci"
 [9] "boot"           "Unidim"         "Fit"            "call"           "title"

     $ total          :'data.frame':        1 obs. of  8 variables:
  ..$ raw_alpha: num 0.703
  ..$ std.alpha: num 0.713
  ..$ G6(smc)  : num 0.683
  ..$ average_r: num 0.332
  ..$ S/N      : num 2.48
  ..$ ase      : num 0.00895
  ..$ mean     : num 4.65
  ..$ sd       : num 0.898
 $ alpha.drop   :'data.frame':        5 obs. of  6 variables:
  ..$ raw_alpha: num [1:5] 0.719 0.617 0.6 0.686 0.643
  ..$ std.alpha: num [1:5] 0.726 0.626 0.613 0.694 0.656
  ..$ G6(smc)  : num [1:5] 0.673 0.579 0.558 0.65 0.605
  ..$ average_r: num [1:5] 0.398 0.295 0.284 0.361 0.322
  ..$ S/N      : num [1:5] 2.64 1.67 1.58 2.26 1.9
  ..$ alpha se : num [1:5] 0.00873 0.0119 0.01244 0.00983 0.01115
 $ item.stats   :'data.frame':        5 obs. of  7 variables:
  ..$ n     : num [1:5] 2784 2773 2774 2781 2784
  ..$ raw.r : num [1:5] 0.581 0.728 0.76 0.654 0.687
  ..$ std.r : num [1:5] 0.566 0.748 0.767 0.631 0.699
  ..$ r.cor : num [1:5] 0.376 0.667 0.709 0.471 0.596
  ..$ r.drop: num [1:5] 0.308 0.564 0.587 0.394 0.489
```

One of the objects of `alpha` is the scores object

R code
```
describe(scores$scores)
```

But, since there are scores for all subjects, but just one score, this is not very interesting.

```
 describe(scores$scores)
   vars    n mean  sd median trimmed  mad min max range  skew kurto
X1    1 2800 4.65 0.9    4.8    4.73 0.89   1   6     5 -0.76
>
```

Note that alpha has the option of doing cumulative scores (adding up items, or scoring in the unit of the items (the default).

R code
```
scores <- alpha(bfi[1:5],check.keys=TRUE,cumulative=TRUE)
#set the cumulative option to be true
describe(scores$scores)
```

```
describe(scores$scores)
   vars    n  mean   sd median trimmed  mad min max range  skew ku
X1    1 2800 23.08 4.54     24   23.43 4.45   5  30    25 -0.73
```

Perhaps a more useful case: scoring multiple scales using `scoreItems`

First, define the scoring keys, and then call `scoreitems` Use the `msq` data set

R code
```
keys <- make.keys(msq[1:75],list(
 EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake", "full.of.pep",
       "lively", "-sleepy", "-tired", "-drowsy"),
TA =c("intense", "jittery", "fearful", "tense", "clutched.up", "-quiet", "-still",
      "-placid", "-calm", "-at.rest") ,
PA =c("active", "excited", "strong", "inspired", "determined", "attentive",
          "interested", "enthusiastic", "proud", "alert"),
 NAf =c("jittery", "nervous", "scared", "afraid", "guilty", "ashamed", "distressed",
      "upset", "hostile", "irritable" ))
)
msq.scores <- scoreItems(keys,msq[1:75])
 msq.scores
```

```
Call: scoreItems(keys = keys, items = msq[1:75])

(Unstandardized) Alpha:
        EA   TA   PA  NAf
alpha 0.93 0.75 0.92 0.83

Standard errors of unstandardized Alpha:
         EA     TA     PA    NAf
ASE   0.004 0.0082 0.0044 0.0064

Average item correlation:
            EA   TA   PA  NAf
average.r 0.58 0.23 0.52 0.33
```

Score multiple scales (continued)

```
Standard errors of unstandardized Alpha:
         EA     TA     PA     NAf
ASE   0.004 0.0082 0.0044 0.0064

Average item correlation:
            EA   TA   PA  NAf
average.r 0.58 0.23 0.52 0.33

Median item correlation:
  EA   TA   PA  NAf
0.59 0.24 0.52 0.40

 Guttman 6* reliability:
           EA  TA   PA  NAf
Lambda.6 0.95 0.8 0.93 0.87

Signal/Noise based upon av.r :
             EA TA PA NAf
Signal/Noise 14  3 11 4.9

Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
       EA   TA    PA    NAf
EA    0.932 0.29 0.870 -0.069
TA    0.238 0.75 0.226  0.710
PA    0.804 0.19 0.915  0.044
NAf  -0.061 0.56 0.039  0.831
```

More detailed item statistics

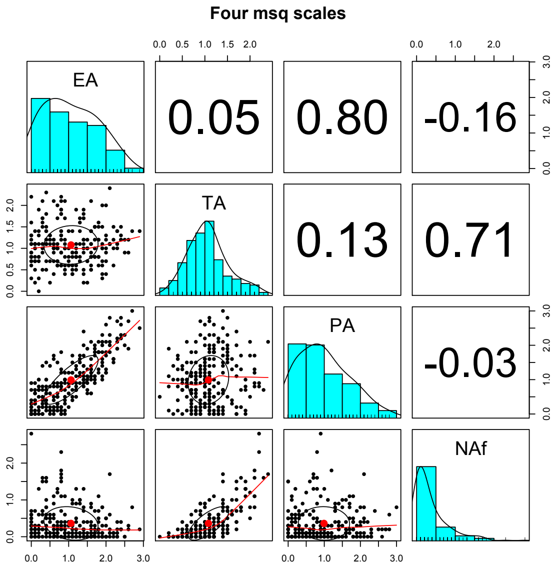
R code

```
print(msq.scores,short=FALSE)
```

```
Item by scale correlations:
 corrected for item overlap and scale reliability
                EA    TA    PA   NAf
delighted     0.56  0.04  0.67 -0.17
sociable      0.57  0.06  0.64 -0.15
jittery       0.15  0.52  0.23  0.46
hostile      -0.21  0.37 -0.13  0.58
sluggish     -0.65 -0.02 -0.40  0.21
depressed    -0.30  0.44 -0.26  0.67
satisfied     0.54 -0.23  0.62 -0.35
relaxed       0.35 -0.56  0.40 -0.51
warmhearted   0.47 -0.09  0.66 -0.19
blue         -0.23  0.40 -0.19  0.64
intense       0.25  0.42  0.44  0.47
strong        0.55  0.00  0.69 -0.03
scared       -0.05  0.61  0.08  0.75
enthusiastic  0.69  0.16  0.83 -0.07
proud         0.57  0.05  0.73 -0.11
sad          -0.22  0.46 -0.18  0.73
active        0.71  0.18  0.83 -0.05
full.of.pep   0.80  0.05  0.80 -0.15
unhappy      -0.30  0.44 -0.26  0.70
lively        0.78  0.04  0.81 -0.13
```

```
Non missing response frequency for each item
                0    1    2    3 miss
delighted    0.50 0.27 0.19 0.04 0.00
sociable     0.24 0.30 0.35 0.11 0.00
jittery      0.60 0.28 0.09 0.03 0.00
hostile      0.77 0.17 0.05 0.01 0.00
sluggish     0.24 0.44 0.19 0.13 0.00
depressed    0.61 0.26 0.08 0.05 0.00
satisfied    0.22 0.29 0.38 0.12 0.00
relaxed      0.13 0.20 0.43 0.24 0.00
warmhearted  0.17 0.23 0.37 0.22 0.00
blue         0.60 0.30 0.08 0.02 0.00
intense      0.54 0.28 0.15 0.03 0.00
strong       0.31 0.26 0.32 0.11 0.00
scared       0.80 0.15 0.04 0.01 0.00
enthusiastic 0.43 0.29 0.19 0.08 0.00
proud        0.41 0.24 0.26 0.10 0.00
sad          0.66 0.22 0.09 0.03 0.00
active       0.38 0.32 0.22 0.08 0.00
full.of.pep  0.54 0.21 0.20 0.05 0.00
unhappy      0.65 0.24 0.07 0.04 0.00
lively       0.44 0.28 0.22 0.06 0.00
```

Show the SPLOM of the msq scales using `pairs.panels`



But what if we have overlapping scales?

1. Sometimes we are interested in how higher order scales relate to lower order scales.
2. The problem is, the items overlap.
3. Some people solve this problem by dropping the overlapping items. But this changes the meaning of the scales.
4. A fairly straightfoward procedure is estimate the overlapping variances with the best estimate of shared (common) variance, similar to what is done when finding coefficient α.
5. Need to do this on the correlation matrix of the items, not the raw data.
6. See ?scoreOverlap

Correcting for item overlap using `scoreOverlap`

R code

```
small.msq <- msq[ c("active", "energetic", "vigorous", "wakeful",
 "wide.awake", "full.of.pep", "lively", "sleepy", "tired", "drowsy",
 "intense", "jittery", "fearful","tense", "clutched.up", "quiet",
 "still",    "placid", "calm", "at.rest") ]
small.R <- cor(small.msq,use="pairwise")
keys.list <- list(
EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake",
"full.of.pep", "lively", "-sleepy", "-tired", "-drowsy"),
TA =c("intense", "jittery", "fearful", "tense", "clutched.up",
"-quiet", "-still",   "-placid", "-calm", "-at.rest") ,
high.EA = c("active", "energetic", "vigorous", "wakeful",
 "wide.awake", "full.of.pep",  "lively"),
low.EA =c("sleepy", "tired", "drowsy"),
lowTA= c("quiet", "still", "placid", "calm", "at.rest"),
highTA = c("intense", "jittery", "fearful", "tense", "clutched.up")
   )

keys <- make.keys(small.R,keys.list)

adjusted.scales <- scoreOverlap(keys.list,small.R)
```

Correcting for item overlap using `scoreOverlap`. (continued)

```
Call: scoreOverlap(keys = keys.list, r = small.R)

(Standardized) Alpha:
     EA      TA high.EA  low.EA   lowTA  highTA
   0.93    0.75    0.94    0.93    0.73    0.76

(Standardized) G6*:
     EA      TA high.EA  low.EA   lowTA  highTA
   0.88    0.68    0.94    0.90    0.73    0.75

Average item correlation:
     EA      TA high.EA  low.EA   lowTA  highTA
   0.59    0.23    0.68    0.81    0.35    0.38

Number of items:
     EA      TA high.EA  low.EA   lowTA  highTA
     10      10       7       3       5       5
Signal to Noise ratio based upon average r and n
     EA      TA high.EA  low.EA   lowTA  highTA
   14.1     3.0    14.8    12.9     2.7     3.1

Scale intercorrelations corrected for item overlap and attenuation
 adjusted for overlap correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
           EA    TA high.EA low.EA lowTA highTA
EA       0.93  0.27   0.965 -0.803 -0.18  0.253
TA       0.23  0.75   0.282 -0.167 -0.81  0.821
high.EA  0.90  0.24   0.937 -0.620 -0.12  0.324
low.EA  -0.75 -0.14  -0.579  0.928  0.25 -0.023
lowTA   -0.15 -0.60  -0.098  0.204  0.73 -0.335
```

Compare scoreOverlap with non-adjusted

R code

```
adjusted.scales <- scoreOverlap(keys.list,small.R)
raw <- scoreItems(keys.list,small.R)
```

```
Scale intercorrelations corrected for item overlap and attenuation
 adjusted for overlap correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
           EA    TA high.EA low.EA lowTA highTA
EA        0.93  0.27   0.965 -0.803 -0.18  0.253
TA        0.23  0.75   0.282 -0.167 -0.81  0.821
high.EA   0.90  0.24   0.937 -0.620 -0.12  0.324
low.EA   -0.75 -0.14  -0.579  0.928  0.25 -0.023
lowTA    -0.15 -0.60  -0.098  0.204  0.73 -0.335
highTA    0.21  0.62   0.273 -0.019 -0.25  0.757
```

```
Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
           EA    TA high.EA low.EA lowTA highTA
EA        0.93  0.27   1.024 -0.848 -0.18  0.253
TA        0.23  0.75   0.282 -0.167 -1.06  1.056
high.EA   0.96  0.24   0.937 -0.620 -0.12  0.324
low.EA   -0.79 -0.14  -0.579  0.928  0.25 -0.023
lowTA    -0.15 -0.78  -0.098  0.204  0.73 -0.335
highTA    0.21  0.80   0.273 -0.019 -0.25  0.757
```

α, $\omega_{hierarchical}$ and β as alternative measures of internal consistency

1. α as the mean split half reliability
   - `alpha` to find α
   - `splitHalf` to find all (if $n \leq 16$) or 10,000 random possible split half reliabilities ($n > 16$)
2. $\omega_{hierarchical}$ and ω_{total} as factor based reliabilities
   - $\omega_{hierarchical}$ estimates general factor saturation
   - Found using `omega` and `omegaSem`
3. β as worst split half reliability as an alternative estimate of the general factor saturation.
   - Found using a hierarchical clustering algorithm (`iclust`).
   - `iclust` is also useful for scale construction.

α from alpha and all split halves found using splitHalf

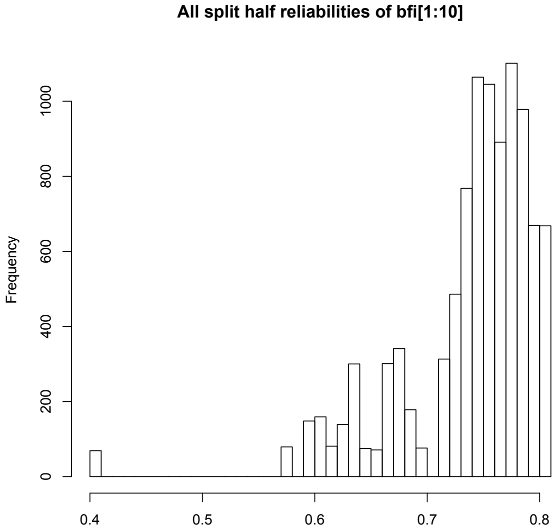
Find α and all split half reliabilities of 5 Agreeableness items and 5 Conscientiousness items from ~~the bfi~~ data set included in *psych*.

R code

```
alpha(bfi[1:10] ,check.keys=TRUE) #find alpha, let it automatically re
splitHalf(bfi[1:10],key=c(1,9,10)) #reverse 3 items
```

```
Reliability analysis
Call: alpha(x = bfi[1:10])
  raw_alpha std.alpha G6(smc) average_r S/N  ase mean   sd
      0.73      0.74    0.76      0.22 2.8 0.01  4.5 0.73
 lower alpha upper     95% confidence boundaries
0.71 0.73 0.75

Split half reliabilities
Call: splitHalf(r = bfi[1:10], key = c(1, 9, 10))

Maximum split half reliability (lambda 4) =  0.81
Guttman lambda 6                          =  0.76
Average split half reliability            =  0.73
Guttman lambda 3 (alpha)                  =  0.74
Minimum split half reliability  (beta)    =  0.41
Average interitem r =  0.22  with median =  0.17
```

All possible spit halves of 5 agreeableness and 5 conscientiousness items. Note the one worst one! This is not one construct.



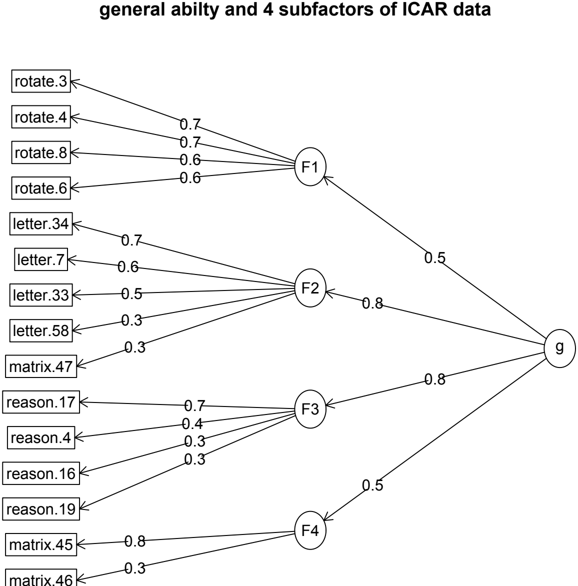
Using the omega function

R code

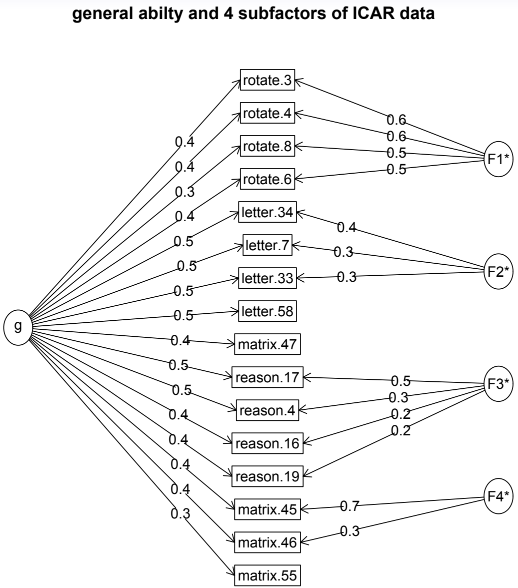
```
omega(ability,4)
```

```
Omega
Call: omega(m = ability, nfactors = 4)
Alpha:                 0.83
G.6:                   0.84
Omega Hierarchical:    0.65
Omega H asymptotic:    0.76
Omega Total            0.86

Schmid Leiman Factor loadings greater than  0.2
             g   F1*   F2*   F3*   F4*   h2   u2   p2
reason.4  0.50              0.27        0.34 0.66 0.73
reason.16 0.42              0.21        0.23 0.77 0.76
reason.17 0.55              0.47        0.52 0.48 0.57
reason.19 0.44              0.21        0.25 0.75 0.77
letter.7  0.52        0.35              0.39 0.61 0.69
letter.33 0.46        0.30              0.31 0.69 0.70
letter.34 0.54        0.38              0.43 0.57 0.67
letter.58 0.47        0.20              0.28 0.72 0.78
matrix.45 0.40                    0.66 0.59 0.41 0.27
matrix.46 0.40                    0.26 0.24 0.76 0.65
matrix.47 0.42                         0.23 0.77 0.79
matrix.55 0.28                         0.12 0.88 0.65
rotate.3  0.36  0.61                   0.50 0.50 0.26
rotate.4  0.41  0.61                   0.54 0.46 0.31
rotate.6  0.40  0.49                   0.41 0.59 0.39
rotate.8  0.32  0.53                   0.40 0.60 0.26

With eigenvalues of:
   g  F1*  F2*  F3*  F4*
```

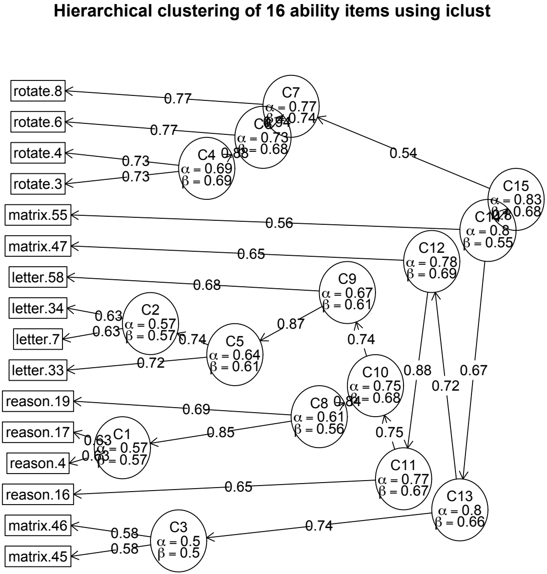
16 ability items from the International Cognitive Ability Resource



Estimating $\omega_{hierarchical}$ and ω_{total} using `omega`



Hierarchical clustering of 16 ICAR ability items: `iclust(ability)`



Exploratory Factor Analysis

1. How many factors: an unsolved problem
   - Parallel analysis, MAPS, VSS, BIC, RMSEA, etc. available in `nfactors` and `fa.parallel`
2. Factor extraction algorithms available in the `fa` function
   - maximum likelihood, minimum residual, principal factor, ...
3. Factor rotation procedures are done using *GPArotation* package
   - orthogonal: varimax, quartimax, bifactor, ...
   - oblique: oblimin, geomin, biquartimin, ...
4. Displaying the solutions using `fa.plot`

Note, that EFA is not the same as Principal Components Analysis and the two should not be confused.

1. PCA done using `principal`

The number of factors problem is easy and hard

No best rule, one worst rule
"Solving the number of factors problem is easy, I do it everyday before breakfast. But knowing the right solution is harder." (Henry Kaiser)

1. Parallel analysis (Extract factors until the eigen values are less than those of a random matrix).
   - Although a good rule for 100-500 subjects, this will not do as well with many (>1000) subjects.
2. Velicer's Mininum Average Partial (MAP) is pretty good
3. For items, the Very Simple Structure (VSS) criterion is pretty good.
4. Multiple statistical tests, many have problems with sample size.
   - If you want few factors, run few subjects
   - If you want many factors, run many subjects
5. One worst rule is the eigen value of 1.0 rule.

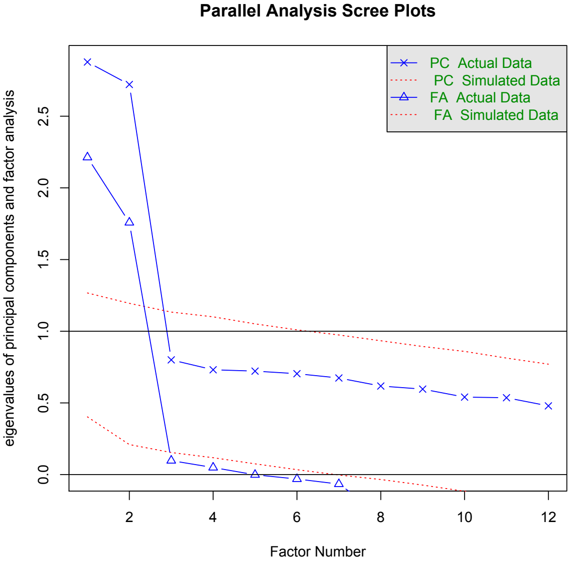
What about parallel analysis? Pearson R or polychoric ρ?

Of the raw (Pearson) correlations compared to the polychoric correlations

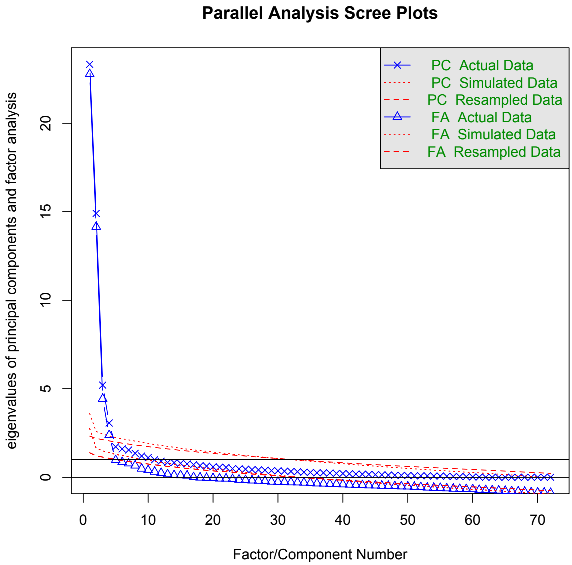
R code

```
 fa.parallel(cleaned[2:73])
Parallel analysis suggests that the number of factors =  5
                   and the number of components =  5


 #use polychoric correlations
fa.parallel(cleaned[2:73],cor="poly")

> fa.parallel(cleaned[2:73],cor="poly")
some warnings are issued
The items do not have an equal number of response
             alternatives, global set to FALSE
Parallel analysis suggests that the number of factors =  4
                 and the number of components =  4
Warning message:
In cor.smooth(mat) : Matrix was not positive definite,
             smoothing was done
>
```

Parallel analysis with Pearson correlations



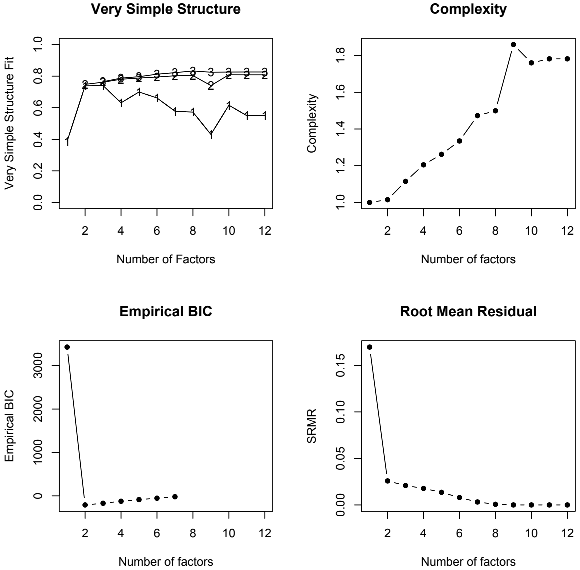
Parallel analysis with polychoric correlations (takes somewhat longer)



How many factors: what does `nfactors` tell us?

```
> nfactors(cleaned[2:73])

Number of factors
Call: vss(x = x, n = n, rotate = rotate, diagonal = diagonal, fm = fm,
    n.obs = n.obs, plot = FALSE, title = title, use = use, cor = cor)
VSS complexity 1 achieves a maximimum of 0.74  with  2  factors
VSS complexity 2 achieves a maximimum of 0.91  with  3  factors
The Velicer MAP achieves a minimum of 0.01  with  9  factors
Empirical BIC achieves a minimum of  -10081.25  with  6  factors
Sample Size adjusted BIC achieves a minimum of  -1843.62  with  11  factors

Statistics by number of factors
   vss1 vss2    map  dof chisq     prob sqresid  fit RMSEA   BIC SABIC complex eChisq  SRMR e(
1  0.68 0.00 0.0552 2484  8907  0.0e+00   192.3 0.68 0.115 -4612  3261     1.0  35253 0.173 0
2  0.74 0.90 0.0179 2413  5827 2.9e-282    58.9 0.90 0.087 -7305   343     1.2   7619 0.080 0
3  0.69 0.91 0.0145 2343  4928 8.6e-186    40.3 0.93 0.077 -7823  -397     1.4   4442 0.061 0
4  0.66 0.86 0.0110 2274  4095 1.7e-107    29.0 0.95 0.067 -8281 -1074     1.5   2532 0.046 0
5  0.62 0.85 0.0101 2206  3587  1.9e-69    24.9 0.96 0.061 -8419 -1427     1.7   1933 0.040 0
6  0.63 0.84 0.0094 2139  3274  4.1e-51    22.0 0.96 0.057 -8367 -1588     1.8   1560 0.036 0
7  0.63 0.83 0.0094 2073  3047  2.1e-40    19.7 0.97 0.055 -8235 -1665     1.9   1314 0.033 0
8  0.63 0.82 0.0091 2008  2810  7.0e-30    17.9 0.97 0.052 -8118 -1754     2.0   1100 0.031 0
...
18 0.53 0.78 0.0105 1413  1467  1.5e-01     9.2 0.98 0.032 -6223 -1745     2.6    324 0.017 0
19 0.52 0.78 0.0108 1359  1371  4.0e-01     8.7 0.99 0.030 -6025 -1718     2.6    291 0.016 0
20 0.52 0.77 0.0111 1306  1284  6.6e-01     8.2 0.99 0.028 -5824 -1685     2.7    258 0.015 0
```

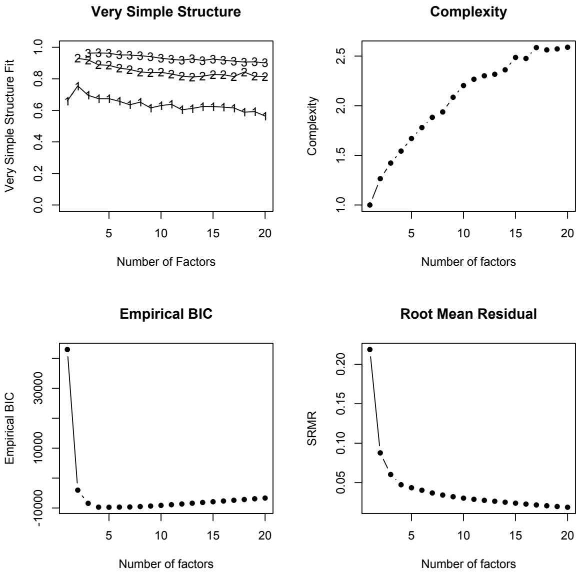
The number of factors from `nfactors`



What if we use polychoric correlations

```
> nfactors(cleaned[2:73],cor="poly")
The items do not have an equal number of response alternatives, global set to FALSE

Number of factors
Call: vss(x = x, n = n, rotate = rotate, diagonal = diagonal, fm = fm,
    n.obs = n.obs, plot = FALSE, title = title, use = use, cor = cor)
VSS complexity 1 achieves a maximimum of 0.76  with  2  factors
VSS complexity 2 achieves a maximimum of 0.93  with  2  factors
The Velicer MAP achieves a minimum of 0.02  with  9  factors
Empirical BIC achieves a minimum of  -9764.2  with  5  factors
Sample Size adjusted BIC achieves a minimum of  39801.49  with  20  factors

Statistics by number of factors
   vss1 vss2   map  dof chisq prob sqresid  fit RMSEA   BIC SABIC complex eChisq  SRMR eCRMS
1  0.66 0.00 0.099 2484 56503    0   280.2 0.66  0.33 42984 50857     1.0  56472 0.219 0.222
2  0.76 0.93 0.027 2413 51500    0    58.1 0.93  0.32 38368 46016     1.3   9088 0.088 0.090
3  0.70 0.92 0.023 2343 50169    0    31.0 0.96  0.32 37417 44843     1.4   4295 0.060 0.063
4  0.67 0.89 0.018 2274 48889    0    21.6 0.97  0.32 36513 43721     1.5   2649 0.047 0.050
5  0.67 0.89 0.018 2206 48230    0    18.6 0.98  0.32 36224 43216     1.7   2242 0.044 0.047
6  0.66 0.87 0.018 2139 47695    0    16.0 0.98  0.33 36053 42833     1.8   1930 0.040 0.044
7  0.64 0.86 0.017 2073 47098    0    13.7 0.98  0.33 35816 42386     1.9   1612 0.037 0.041
8  0.65 0.84 0.017 2008 46595    0    11.8 0.99  0.33 35666 42031     1.9   1391 0.034 0.039
...
19 0.59 0.82 0.022 1359 43055    0     3.5 1.00  0.40 35659 39966     2.6    462 0.020 0.027
20 0.56 0.82 0.022 1306 42770    0     3.1 1.00  0.41 35662 39801     2.6    419 0.019 0.026
Warning message:
In cor.smooth(mat) : Matrix was not positive definite, smoothing was done
```

The number of factors from `nfactors`



EFA of the Motivational State Questionnaire

R code

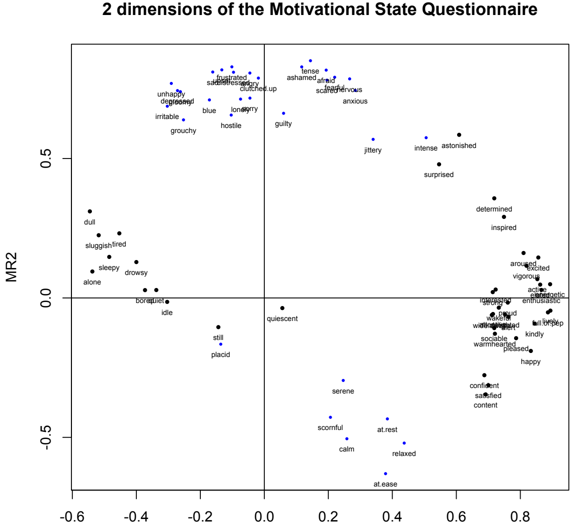
```
f2 <- fa(msq[1:72],2)
summary(f2)
```

```
ummary(f2)

Factor analysis with Call: fa(r = msq[1:72], nfactors = 2)

Test of the hypothesis that 2 factors are sufficient.
The degrees of freedom for the model is 2413  and the objective fu
The number of observations was  3896  with Chi Square =  67730.13

The root mean square of the residuals (RMSA) is  0.09
The df corrected root mean square of the residuals is  0.09

Tucker Lewis Index of factoring reliability =  0.637
RMSEA index =  0.083  and the 90 % confidence intervals are  0.083
BIC =  47780.16
 With factor correlations of
      MR1   MR2
MR1  1.00 -0.13
MR2 -0.13  1.00
```

Show the factors, sorted by factor loadings

```
> print(f2,sort=TRUE)
Factor Analysis using method =  minres
Call: fa(r = msq[1:72], nfactors = 2, cor = "poly")
Standardized loadings (pattern matrix) based upon correlation matrix
             item   MR1   MR2    h2   u2 com
lively         20  0.89 -0.05 0.811 0.19 1.0
energetic      55  0.89  0.05 0.789 0.21 1.0
full.of.pep    18  0.89 -0.05 0.800 0.20 1.0
...
sluggish        5 -0.52  0.22 0.348 0.65 1.4
sleepy         59 -0.48  0.15 0.274 0.73 1.2
tired          28 -0.45  0.23 0.285 0.71 1.5
drowsy         51 -0.40  0.13 0.189 0.81 1.2
...
tense          69  0.14  0.85 0.714 0.29 1.1
frustrated     65 -0.10  0.83 0.718 0.28 1.0
ashamed        70  0.12  0.83 0.676 0.32 1.0
upset          48 -0.13  0.82 0.714 0.29 1.1
...
relaxed         8  0.44 -0.52 0.519 0.48 1.9
calm           50  0.26 -0.50 0.354 0.65 1.5
at.rest        26  0.38 -0.43 0.378 0.62 2.0
...

                        MR1   MR2
SS loadings           21.07 17.15
Proportion Var         0.29  0.24
Cumulative Var         0.29  0.53
Proportion Explained   0.55  0.45
Cumulative Proportion  0.55  1.00
```

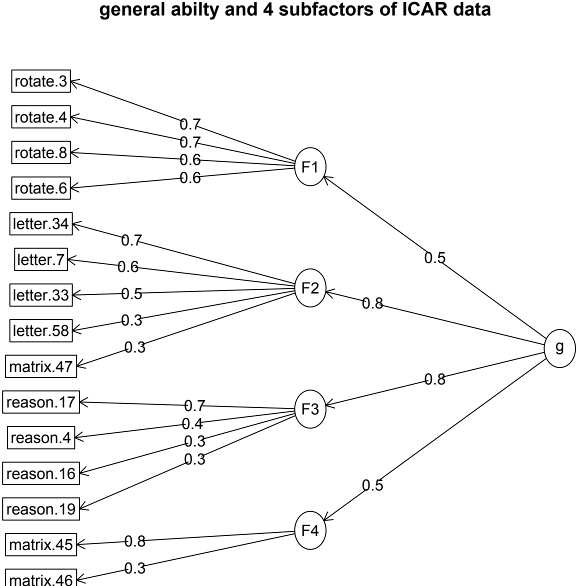
fa.plot(f2,labels=colnames(msq[1:72]),cex=.5,title="2 dimensions of the Motivational State Questionnaire")



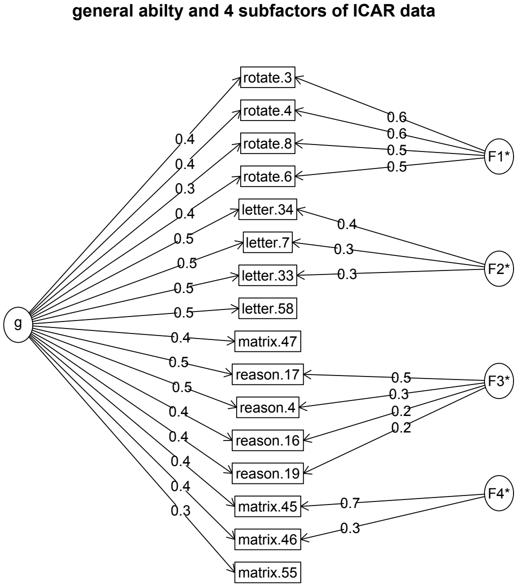
Convert to and sort polar coordinates `round(polar(f2),2)`

```
              Var theta21 vecl21
strong         12    1.68   0.51
enthusiastic   14    1.89   0.75
...
anxious        71   69.09   0.63
nervous        45   71.26   0.69
...
angry          44   93.20   0.65
sorry          58   93.59   0.52
...
sad            16  101.25   0.68
blue           10  103.59   0.53
...
drowsy         51  162.17   0.18
sleepy         59  163.11   0.26
...
relaxed         8  310.03   0.46
at.rest        26  311.56   0.34
..
happy          61  347.14   0.73
pleased        60  349.61   0.64
...
alert          52  354.89   0.59
```

16 ability items from the International Cognitive Ability Resource



Schmid Leiman transformation of 16 ability items from ICAR



More types of reliability

1. α is a hodgepodge ratio of general factor and group factor reliability
2. ω_h (omega hierarchical) is an estimate of the general factor variance of a test
3. ω_t (omega total) is an estimate of the total reliable variance of a test
4. When do we use these?
   - When estimating how much of a test measures one thing. $omega_h$
   - When estimating what is the total reliable variance in a test (when adjusting for test reliability in an SEM context)

ω_h and ω_t reliabilities

R code
```
om <- omega(ability, nfactors=4)
```

```
om
Omega
Call: omega(m = ability, nfactors = 4)
Alpha:                  0.83
G.6:                    0.84
Omega Hierarchical:     0.65
Omega H asymptotic:     0.76
Omega Total             0.86

Schmid Leiman Factor loadings greater than  0.2
             g   F1*  F2*  F3*  F4*  h2   u2   p2
reason.4  0.50           0.27      0.34 0.66 0.73
reason.16 0.42           0.21      0.23 0.77 0.76
reason.17 0.55           0.47      0.52 0.48 0.57
reason.19 0.44           0.21      0.25 0.75 0.77
letter.7  0.52      0.35           0.39 0.61 0.69
letter.33 0.46      0.30           0.31 0.69 0.70
letter.34 0.54      0.38           0.43 0.57 0.66
letter.58 0.47      0.20           0.28 0.72 0.78
matrix.45 0.40                0.66 0.59 0.41 0.27
matrix.46 0.40                0.26 0.24 0.76 0.65
matrix.47 0.42                     0.23 0.77 0.79
matrix.55 0.28                     0.12 0.88 0.65
rotate.3  0.36 0.61                0.50 0.50 0.26
rotate.4  0.41 0.61                0.54 0.46 0.31
rotate.6  0.40 0.49                0.41 0.59 0.39
rotate.8  0.32 0.53                0.40 0.60 0.26
```

ω continued

```
With eigenvalues of:
   g  F1*  F2*  F3*  F4*
3.04 1.32 0.46 0.42 0.55

general/max  2.3   max/min =   3.17
mean percent general =  0.58    with sd =  0.2 and cv of  0.35
Explained Common Variance of the general factor =  0.53

The degrees of freedom are 62  and the fit is  0.05
The number of observations was  1525  with Chi Square =  70.19  with prob <  0.22
The root mean square of the residuals is  0.01
The df corrected root mean square of the residuals is  0.02
RMSEA index =  0.009  and the 90 % confidence intervals are  0 0.014
BIC =  -384.25

Compare this with the adequacy of just a general factor and no group factors
The degrees of freedom for just the general factor are 104  and the fit is  0.78
The number of observations was  1525  with Chi Square =  1186.18  with prob <  5e-183
The root mean square of the residuals is  0.09
The df corrected root mean square of the residuals is  0.09

RMSEA index =  0.083  and the 90 % confidence intervals are  0.078 0.085
BIC =  423.88

Measures of factor score adequacy
                                                g  F1*   F2*   F3*  F4*
Correlation of scores with factors           0.83 0.80  0.53  0.56 0.71
Multiple R square of scores with factors     0.69 0.64  0.28  0.32 0.50
Minimum correlation of factor score estimates 0.37 0.28 -0.44 -0.37 0.00

 Total, General and Subset omega for each subset
```

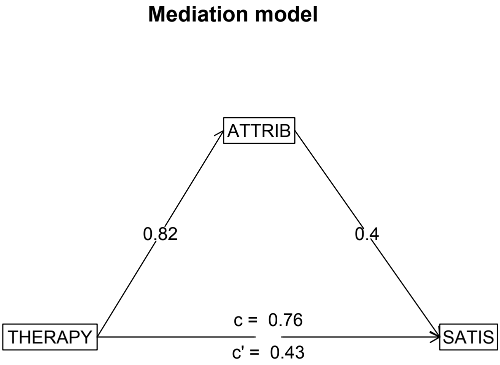
Data set from Preacher & Hayes (2004)

R code

```
# from Preacher and Hayes (2004)
sobel <- structure(list(SATIS = c(-0.59, 1.3, 0.02, 0.01, 0.79, -0.35,
-0.03, 1.75, -0.8, -1.2, -1.27, 0.7, -1.59, 0.68, -0.39, 1.33,
...
"Therapy", "Attributional Positivity"), .Names = c("SATIS", "THERAPY",
"ATTRIB")))
 #n.iter set to 50 (instead of default of 5000) for speed of example
mediate(1,2,3,sobel,n.iter=50)  #The example in Preacher and Hayes
```

```
The DV (Y) was  SATIS . The IV (X) was  THERAPY . The mediating variable(s) =  ATTRIB .

Total Direct effect(c) of  THERAPY  on  SATIS  =  0.76   S.E. =  0.31  t direct =  2.5
Direct effect (c') of  THERAPY  on  SATIS  removing  ATTRIB  =  0.43   S.E. =  0.32  t di
Indirect effect (ab) of  THERAPY  on  SATIS  through  ATTRIB   =  0.33
Mean bootstrapped indirect effect =  0.31  with standard error =  0.16  Lower CI =  0.07
R2 of model =  0.31
 To see the longer output, specify short = FALSE in the print statement

 Full output

 Total effect estimates (c)
        SATIS   se   t   Prob
THERAPY  0.76 0.31 2.5 0.0186

Direct effect estimates     (c')
        SATIS   se    t  Prob
THERAPY  0.43 0.32 1.35 0.190
ATTRIB   0.40 0.18 2.23 0.034

 'a'  effect estimates
         THERAPY  se    t   Prob
```

The Preacher mediation example



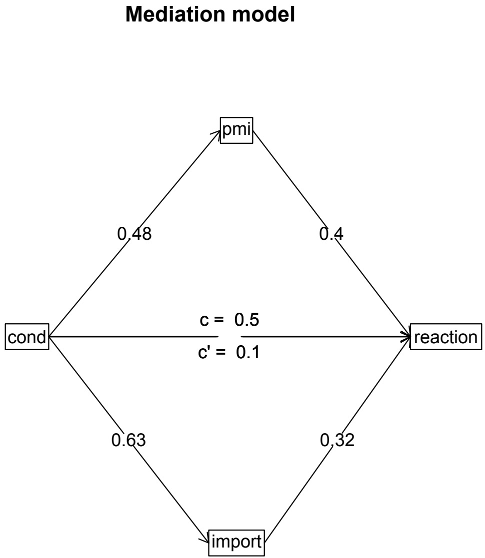
Take the data example from Hayes (2013)

R code

```
C.pmi <- cov(Tal_Or)
 #n.iter set to 50 (instead of default of 5000) for speed of example
mediate(y="reaction",x = "cond",m=c("pmi","import"),data=C.pmi,n.obs=123,n.iter=50)
```

```
Call: mediate(y = "reaction", x = "cond", m = c("pmi", "import"), data = C.pmi,
    n.obs = 123, n.iter = 50)

The DV (Y) was  reaction . The IV (X) was  cond . The mediating variable(s) =  pmi import

Total Direct effect(c) of  cond  on  reaction  =  0.5   S.E. =  0.28  t direct =  1.79
Direct effect (c') of  cond  on  reaction  removing  pmi import  =  0.1   S.E. =  0.24  t
Indirect effect (ab) of  cond  on  reaction  through  pmi import   =  0.39
Mean bootstrapped indirect effect =  0.7  with standard error =  0.17  Lower CI =  0.39
R2 of model =  0.33
 To see the longer output, specify short = FALSE in the print statement

 Full output

 Total effect estimates (c)
      reaction   se    t   Prob
cond       0.5 0.28 1.79 0.0766

Direct effect estimates     (c')
       reaction   se    t     Prob
cond       0.10 0.24 0.43 6.66e-01
pmi        0.40 0.09 4.26 4.04e-05
import     0.32 0.07 4.59 1.13e-05

 'a'  effect estimates
       cond   se    t   Prob
```

The Hayes (2013) example mediation



The "New" Psychometrics

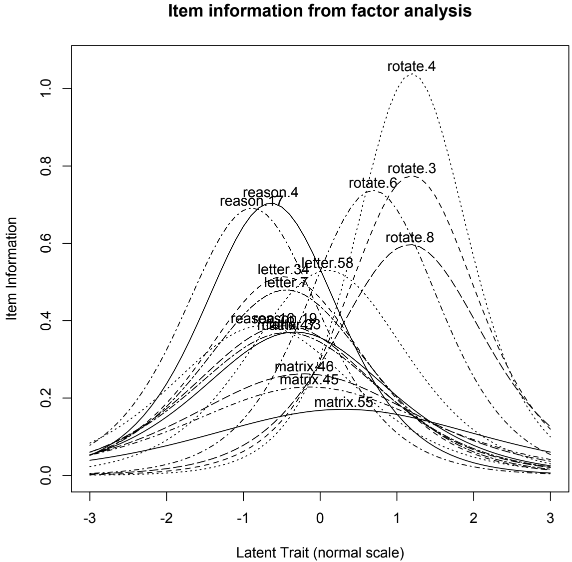
1. Classical Test theory examines responses assuming items are equivalent, or at least congeneric equivalent
2. Item Response Theory models item difficulty as well as item discrimination
3. Although seemingly very different models, factor analysis of categorical items (using tetrachoric or polychoric correlations) is equivalent to IRT 2 PL models.
4. Rasch model is just a 1 PL model where items differ in difficulty, but not discrimination.
5. 2PL has difficulty and discrimination estimated from factor analysis of tetrachoric/polychoric items.

R code

```
f1 <- irt.fa(ability)
```

```
 f1 <- irt.fa(ability)

> f1
Item Response Analysis using Factor Analysis

Call: irt.fa(x = ability)
Item Response Analysis using Factor Analysis

 Summary information by factor and item
 Factor =  1
               -3   -2   -1    0    1    2     3
reason.4     0.05 0.24 0.64 0.53 0.16 0.03  0.01
reason.16    0.08 0.22 0.38 0.31 0.14 0.05  0.01
reason.17    0.08 0.33 0.69 0.42 0.11 0.02  0.00
reason.19    0.06 0.17 0.35 0.36 0.19 0.07  0.02
letter.7     0.05 0.18 0.41 0.44 0.20 0.06  0.02
letter.33    0.05 0.15 0.31 0.36 0.20 0.08  0.02
letter.34    0.05 0.19 0.45 0.46 0.20 0.06  0.01
letter.58    0.02 0.09 0.30 0.53 0.35 0.12  0.03
matrix.45    0.05 0.11 0.19 0.23 0.17 0.09  0.04
...
Test Info    0.67 2.11 4.73 5.83 5.28 2.55  0.69
```

FA solution with tetrachoric correlations



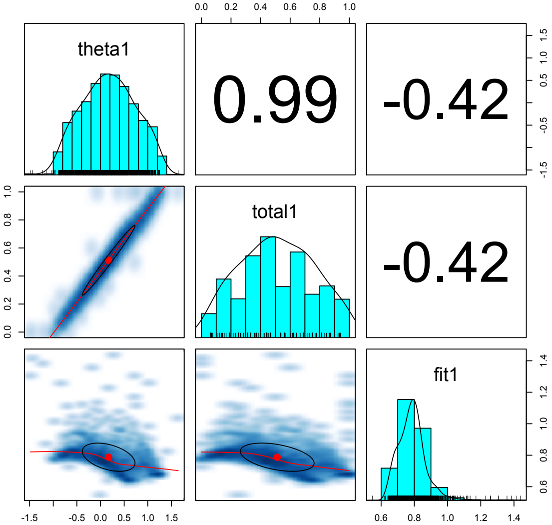
IRT based scoring and Classical Test Theory based scoring

1. CTT and IRT based scores correlate almost perfectly without missing data
2. With lots of missing data, and different items for different people, IRT based scores provide more subtle distinctions.
3. `scoreIrt.2pl` and `scoreIrt.1pl` will do IRT based scores.
4. By default, will find the irt based parameters and then do the scoring.

R code

```
ability.irt <- irt.fa(ability)
ability.scores <- scoreIrt(ability.irt,ability)
 pairs.panels(ability.scores,smoother=TRUE)
```

CTT and IRT based scores are almost identical



Multilevel reliability

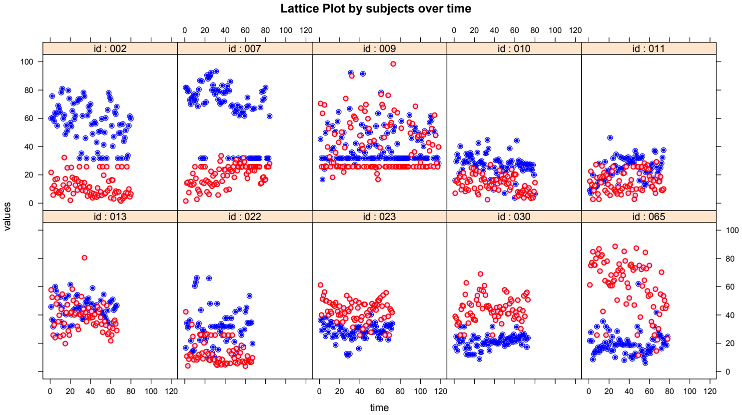
1. Classic reliability measures assess the variance of between person differences compared to error of the measurement.
$$\rho_{xx} = \frac{1 - \sigma_e^2}{\sigma_x^2} \tag{1}$$
2. Multilevel reliability is a series of generallizability coeffiicients, generalizing over items, over time, time x items,
$$R_{kF} = \frac{\sigma_{id}^2 + (\sigma_{idxitems}^2/m)}{\sigma_{id}^2 + (\sigma_{idxitems}^2/m) + \sigma_{error}^2/(km)} \tag{2}$$
From Equation 6 (Shrout & Lane, 2012, p 310). See Shrout & Lane (2012) for five other generalizability formula.
3. Implemented in *psych* as `mlr` or `multilevel.reliability`
4. Also simulations using `sim.multi`
5. I show the data from Fisher (2015) who reports 10 subjects measured over 60 (or more) days on 28 affect items.
6. (Download the R data files, minor rearrangement and

Table: The `multilevel.reliablity` function estimates of the generalizability coefficients for the positively and negatively valenced items from Fisher (2015). RkF is the reliability of average of all ratings across all items and times (Fixed time effects), R1R is the generalizability of a single time point across all items (Random time effects), RkR is the generalizability of average time points across all items (Random time effects), Rc is the generalizability of change (fixed time points, fixed items), RkRn is the generalizability of between person differences averaged over time (time nested within people) and Rcn is the generalizability of within person variations averaged over items (time nested within people).

Multilevel reliability estimates

Coefficient	Positive items	Negative items
RkF	1.00	1.00
R1R	0.80	0.77
RkR	1.00	1.00
Rc	0.72	0.71
RkRn	1.00	1.00
Rcn	0.64	0.59

Assessing reliability of within subject differences in affect. Data from Fisher (2015)



psych includes some very old ideas

1. Schmid-Leiman (Schmid & Leiman, 1957) transformations from correlated factor stuctures to higher order structures.
2. Dwyer extension (Dwyer, 1937; Mosier, 1938; Horn, 1973) to extend a factor solution to more variables.
3. This can be used to extend other variables into a factor space, or to relate two domains to each other.

Extend a data set into another

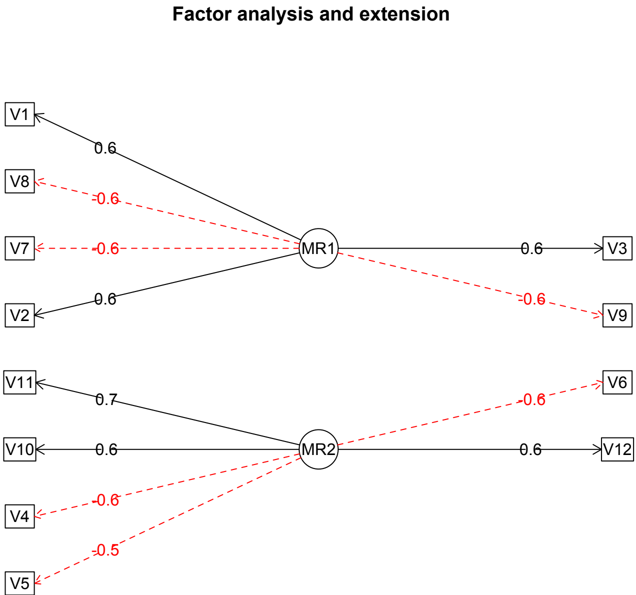
First, create the data set

R code

```
set.seed(42)
 d <- sim.item(12)     #two orthogonal factors
R <- cor(d)
Ro <- R[c(1,2,4,5,7,8,10,11),c(1,2,4,5,7,8,10,11)]
 Roe <- R[c(1,2,4,5,7,8,10,11),c(3,6,9,12)]
 fo <- fa(Ro,2)
 fe <- fa.extension(Roe,fo)
 fa.diagram(fo,fe=fe)
```

```
fe

Call: fa.extension(Roe = Roe, fo = fo)
Standardized loadings (pattern matrix) based upon correlation matrix
      MR1   MR2   h2   u2
V3   0.63 -0.02 0.39 0.61
V6   0.04 -0.61 0.37 0.63
V9  -0.61  0.01 0.38 0.62
V12 -0.06  0.58 0.33 0.67

                       MR1  MR2
SS loadings           0.77 0.69
Proportion Var        0.19 0.17
Cumulative Var        0.19 0.37
Proportion Explained  0.53 0.47
Cumulative Proportion 0.53 1.00
```

Factor extension

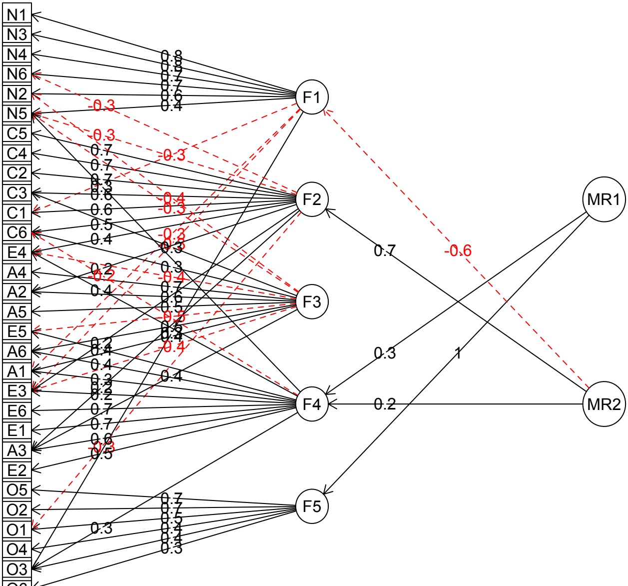


Hiearchical factor analysis

R code

```
neo52 <- fa.multi(neo,5,2)
fa.multi.diagram(neo52)
```

Hierarchical (multilevel) Structure



ESEM can be thought of as factor extension from A to B and B to A

1. If we have two sets of variables that show factor structures within each set
2. And then link the factor structures.
3. Tjhis can be done in SEM, but here show how to do exploratory SEM
4. We make up a toy data set

ESEM

R code

```
#make up a sem like problem using sim.structure
fx <-matrix(c( .9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
 fy <- matrix(c(.6,.5,.4),ncol=1)
 rownames(fx) <- c("V","Q","A","nach","Anx")
 rownames(fy)<- c("gpa","Pre","MA")
Phi <-matrix( c(1,0,.7,.0,1,.7,.7,.7,1),ncol=3)
 gre.gpa <- sim.structural(fx,Phi,fy)
print(gre.gpa)
```

```
Call: sim.structural(fx = fx, Phi = Phi, fy = fy)

 $model (Population correlation matrix)
         V     Q     A  nach   Anx   gpa   Pre    MA
V     1.00  0.72  0.54  0.00  0.00  0.38  0.32  0.25
Q     0.72  1.00  0.48  0.00  0.00  0.34  0.28  0.22
A     0.54  0.48  1.00  0.48 -0.42  0.50  0.42  0.34
nach  0.00  0.00  0.48  1.00 -0.56  0.34  0.28  0.22
Anx   0.00  0.00 -0.42 -0.56  1.00 -0.29 -0.24 -0.20
gpa   0.38  0.34  0.50  0.34 -0.29  1.00  0.30  0.24
Pre   0.32  0.28  0.42  0.28 -0.24  0.30  1.00  0.20
MA    0.25  0.22  0.34  0.22 -0.20  0.24  0.20  1.00

$reliability (population reliability)
   V     Q     A nach  Anx  gpa  Pre   MA
```

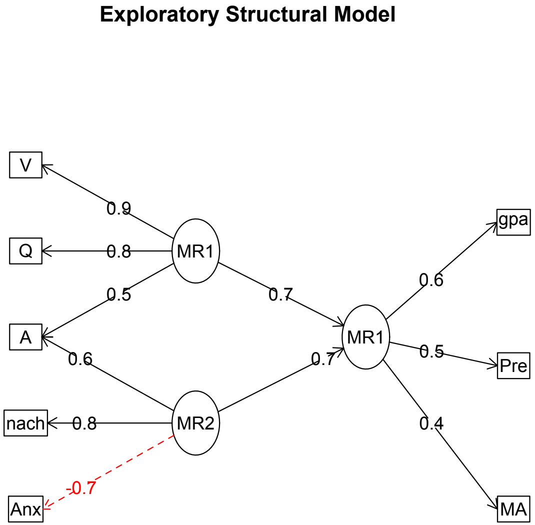
Exploratory Structural Equation Modling

R code

```
example <- esem(gre.gpa$model,varsX=1:5,varsY=6:8,nfX=2,nfY=1,
n.obs=1000,plot=FALSE)
```

```
> example
Exploratory Structural Equation Modeling  Analysis using method =  minres
Call: esem(r = gre.gpa$model, varsX = 1:5, varsY = 6:8, nfX = 2, nfY = 1,
    n.obs = 1000, plot = FALSE)

For the 'X' set:
       MR1   MR2
V      0.91 -0.06
Q      0.81 -0.05
A      0.53  0.57
nach -0.10  0.81
Anx   0.08 -0.71

For the 'Y' set:
    MR1
gpa 0.6
Pre 0.5
MA  0.4

Correlations between the X and Y sets.
     X1   X2   Y1
X1 1.00 0.19 0.68
X2 0.19 1.00 0.67
Y1 0.68 0.67 1.00

The degrees of freedom for the null model are  56  and the empirical chi square function
The degrees of freedom for the model are 7  and the empirical chi square function was 21
```

ESEM of our toy problem



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

Bechtoldt, H. (1961). An empirical study of the factor analysis stability hypothesis. *Psychometrika*, *26*(4), 405–432.

Bond, T. G. (1995). *BLOT:Bond's Logical Operations Test*. Townsville, Australia: James Cook Univer- sity. (Original work published 1976).

Burt, C. (1915). General and specific factors underlying the primary emotions. In *Reports of the British Association for the Advancement of Science (85th Meeting)*, (pp. 694–696)., London (retrieved from the web at http://www.biodiversitylibrary.org/item/95822#790)). John Murray.

Condon, D. M. & Revelle, W. (2014). The International Cognitive Ability Resource: Development and initial validation of a public-domain measure. *Intelligence*, *43*, 52–64.

Costa, P. T. & McCrae, R. R. (1985). *NEO PI professional manual*. Odessa, FL: Psychological Assessment Resources, Inc.

Cushny, A. R. & Peebles, A. R. (1905). The action of optical isomer. ii. hyoscines. *The Journal of Physiology*, *32*(501-510).

Dwyer, P. S. (1937). The determination of the factor loadings of a given test from the known factor loadings of other tests. *Psychometrika*, *2*(3), 173–178.

Fisher, A. J. (2015). Toward a dynamic model of psychological assessment: Implications for personalized care. *Journal of Consulting and Clinical Psychology*, *83*(4), 825 – 836.

Fox, J., Nie, Z., & Byrnes, J. (2013). *sem: Structural Equation Models*. R package version 3.1-3.

Guilford, J. P. (1954). *Psychometric Methods* (2nd ed.). New York: McGraw-Hill.

Hayes, A. F. (2013). *Introduction to mediation, moderation, and conditional process analysis: A regression-based approach*. New York: Guilford Press.

Holzinger, K. & Swineford, F. (1937). The bi-factor method. *Psychometrika*, *2*(1), 41–54.

Horn, J. L. (1973). On extension analysis and its relation to correlations between variables and factor scores. *Multivariate Behavioral Research*, *8*(4), 477 – 489.

Mosier, C. (1938). A note on Dwyer: The determination of the factor loadings of a given test. *Psychometrika*, *3*(4), 297–299.

Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kickpatrick, R. M., Estabrook, R., Bates, T. C., Maes, H. H., & Boker, S. M. (2016). OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika*.

Preacher, K. J. & Hayes, A. F. (2004). SPSS and SAS procedures for estimating indirect effects in simple mediation models. *Behavior Research Methods, Instruments, & Computers*, *36*(4), 717–731.

Reise, S., Morizot, J., & Hays, R. (2007). The role of the bifactor model in resolving dimensionality issues in health outcomes measures. *Quality of Life Research*, *16*(0), 19–31.

Revelle, W. & Condon, D. M. (2015). A model for personality at three levels. *Journal of Research in Personality*, *56*, 70–81.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software*, *48*(2), 1–36.

Schmid, J. J. & Leiman, J. M. (1957). The development of hierarchical factor solutions. *Psychometrika*, *22*(1), 83–90.

Shrout, P. E. & Lane, S. P. (2012). Psychometrics. In *Handbook of research methods for studying daily life*. Guilford Press.

Smillie, L. D., Cooper, A., Wilt, J., & Revelle, W. (2012). Do extraverts get more bang for the buck? refining the affective-reactivity hypothesis of extraversion. *Journal of Personality and Social Psychology*, *103*(2), 306–326.

Tal-Or, N., Cohen, J., Tsfati, Y., & Gunther, A. C. (2010). Testing causal direction in the influence of presumed media influence. *Communication Research*, *37*(6), 801–824.

Thurstone, L. L. & Thurstone, T. G. (1941). *Factorial studies of intelligence*. Chicago, Ill.: The University of Chicago press.

Venables, W. N. & Ripley, B. D. (2002). *Modern Applied Statistics with S* (Fourth ed.). New York: Springer. ISBN 0-387-95457-0.