Barry Grant
bjgrant@umich.edu
http://thegrantlab.org
What is R?

R is a freely distributed and widely used programming language and environment for statistical computing, data analysis, and graphics.

R provides an unparalleled interactive environment for data analysis.

It is script-based (i.e. driven by computer code) and not GUI-based (point and click with menus).
What is R?

R is a freely distributed and widely used programming language and environment for statistical computing, data analysis and graphics. R provides an unparalleled interactive environment for data analysis. It is script-based (i.e. driven by computer code) and not GUI-based (point and click with menus).
What is R?

R is a freely distributed and widely used programing language and environment for statistical computing, data analysis and graphics.

R provides an unparalleled interactive environment for data analysis. It is script-based (i.e. driven by computer code) and not GUI-based (point and click with menus).

Type "R" in your terminal

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (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.
What is R?

R is a freely distributed and widely used programming language and environment for statistical computing, data analysis, and graphics. R provides an unparalleled interactive environment for data analysis. It is script-based (i.e. driven by computer code) and not GUI-based (point and click with menus).

Type "R" in your terminal

This is the R prompt
What is R?

R is a freely distributed and widely used programing language and environment for statistical computing, data analysis and graphics.

R provides an unparalleled interactive environment for data analysis. It is script-based (i.e. driven by computer code) and not GUI-based (point and click with menus).

Type "R" in your terminal

This is the R prompt: Type `q()` to quit!
What R is NOT

A performance optimized software library for incorporation into your own C/C++ etc. programs.

A molecular graphics program with a slick GUI.

Backed by a commercial guarantee or license.

Microsoft Excel!
What about Excel?

- Data manipulation is easy
- Can see what is happening
- **But**: graphics are poor
- Looping is hard
- Limited statistical capabilities
- Inflexible and irreproducible
- **There are many many things Excel just cannot do!**
Rule of thumb: Every analysis you do on a dataset will have to be redone 10–15 times before publication. Plan accordingly!
Why use R?

- Productivity
- Flexibility
- Designed for data analysis
<table>
<thead>
<tr>
<th>Language Rank</th>
<th>Types</th>
<th>Spectrum Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. C</td>
<td>📑rolleyboard, 🖥computer, 🌐internet</td>
<td>100.0</td>
</tr>
<tr>
<td>2. Java</td>
<td>🌐internet, 📑rolleyboard, 🖥computer</td>
<td>98.1</td>
</tr>
<tr>
<td>3. Python</td>
<td>🌐internet, 🖥computer</td>
<td>98.0</td>
</tr>
<tr>
<td>4. C++</td>
<td>📑rolleyboard, 🖥computer</td>
<td>95.9</td>
</tr>
<tr>
<td>5. R</td>
<td>📑rolleyboard, 🖥computer</td>
<td>87.9</td>
</tr>
<tr>
<td>6. C#</td>
<td>🌐internet, 📑rolleyboard, 🖥computer</td>
<td>86.7</td>
</tr>
<tr>
<td>7. PHP</td>
<td>🌐internet</td>
<td>82.8</td>
</tr>
<tr>
<td>8. JavaScript</td>
<td>🌐internet, 📑rolleyboard</td>
<td>82.2</td>
</tr>
<tr>
<td>9. Ruby</td>
<td>🌐internet, 🖥computer</td>
<td>74.5</td>
</tr>
<tr>
<td>10. Go</td>
<td>🌐internet, 🖥computer</td>
<td>71.9</td>
</tr>
</tbody>
</table>

http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages
R and Python: The Numbers

Popularity Rankings

R and Python's popularity between 2013 and February 2015 (Tiobe Index)

Jobs And Salary?

2014 Dice Tech Salary Survey: Average Salary For High Paying Skills and Experience

Python

R

$ 115,531

$94,139
• R is the “lingua franca” of data science in industry and academia.

• Large user and developer community.
  • As of Aug 1st 2016 there are 8811 add on R packages on CRAN and 1211 on Bioconductor - more on these later!

• Virtually every statistical technique is either already built into R, or available as a free package.

• Unparalleled exploratory data analysis environment.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity</td>
<td>Core R functions are modular and work well with others</td>
</tr>
<tr>
<td>Interactivity</td>
<td>R offers an unparalleled exploratory data analysis environment</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Access to existing tools and cutting-edge statistical and graphical methods</td>
</tr>
<tr>
<td>Support</td>
<td>Extensive documentation and tutorials available online for R</td>
</tr>
<tr>
<td>R Philosophy</td>
<td>Encourages open standards and reproducibility</td>
</tr>
<tr>
<td><strong>Modularity</strong></td>
<td>Core R functions are modular and work well with others</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Interactivity</strong></td>
<td>R offers an unparalleled exploratory data analysis environment</td>
</tr>
<tr>
<td><strong>Infrastructure</strong></td>
<td>Access to existing tools and cutting-edge statistical and graphical methods</td>
</tr>
<tr>
<td><strong>Support</strong></td>
<td>Extensive documentation and tutorials available online for R</td>
</tr>
<tr>
<td><strong>R Philosophy</strong></td>
<td>Encourages open standards and reproducibility</td>
</tr>
</tbody>
</table>
Modularity

R was designed to allow users to interactively build complex workflows by interfacing smaller ‘modular’ functions together.

An alternative approach is to write a single complex program that takes raw data as input, and after hours of data processing, outputs publication figures and a final table of results.

All-in-one custom ‘Monster’ program
Another common approach to bioinformatics data analysis is to write individual scripts in Perl/Python/Awk/C etc. to carry out each subsequent step of an analysis.

This can offer many advantages but can be challenging to make robustly modular and interactive.
Interactivity & exploratory data analysis

Learning R will give you the freedom to explore and experiment with your data.

“Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions”. [J. W. Tukey]
Interactivity & exploratory data analysis

Learning R will give you the freedom to explore and experiment with your data.

“Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions”. [J. W. Tukey]

Bioinformatics data is intrinsically high dimensional and frequently ‘messy’ requiring exploratory data analysis to find patterns - both those that indicate interesting biological signals or suggest potential problems.
R Features = \texttt{functions()}

\begin{itemize}
\item Input sequence or structure
  \texttt{aa = get.seq("lrz2.A")}
\item BLAST or HMMER search
  \texttt{hits = blast(aa)}
\item Download related structures
  \texttt{files = get.pdb(hits)}
\item Align structures
  \texttt{pdsbs = pdbaln(files)}
\item Input user specific PDBs
\item Input simulation trajectory
\item Ensemble NMA
  \texttt{modes = nma(pdsbs)}
\item Ensemble PCA & additonal sequence/structure analysis
  \texttt{pc = pca(pdsbs)}
  \texttt{# conserv() rmsd()}
  \texttt{# rmsf() dccm() etc.}
\item Fluctuation analysis
  \texttt{plot(modes)}
\item Correlation networks
  \texttt{cna(ci)}
\item X-ray & MD PCA
  \texttt{plot(pc)}
\item Clustering
  \texttt{heatmap(modes)}
\item PCA vs NMA
  \texttt{rmsip(pc,modes)}
\end{itemize}
How do we use R?
Two main ways to use R

1. Terminal
2. RStudio

1- Code Editor
2- R Console
3- Workspace and History
4- Plots and files
We will use **RStudio** today.
Let's get started...

1. Code Editor
2. R Console
3. Workspace and History
4. Plots and files
Some simple R commands

1. ```> 2+2
   [1] 4```  
   R prompt!

2. ```> 3^2
   [1] 9```  

3. ```> sqrt(25)
   [1] 5```  

4. ```> 2*(1+1)
   [1] 4```  

5. ```> 2*1+1
   [1] 3```  
   Order of precedence

6. ```> exp(1)
   [1] 2.718282```  

7. ```> log(2.718282)
   [1] 1```  

8. ```> log(10, base=10)
   [1] 1```  
   Optional argument

9. ```> log(10 + , base = 10)
   [1] 1```  
   Incomplete command

10. ```> x=1:50
    > plot(x, sin(x))```  
    Do it Yourself!
Learning a new language is hard!
Error Messages

Sometimes the commands you enter will generate errors. Common beginner examples include:

• Incomplete brackets or quotes e.g.
  
  ```
  ((4+8)*20  <enter>
  +
  ```
  This returns a + here, which means you need to enter the remaining bracket - R is waiting for you to finish your input.
  Press `<ESC>` to abandon this line if you don't want to fix it.

• Not separating arguments by commas e.g.
  ```
  plot(1:10 col="red")
  ```

• Typos including miss-spelling functions and using wrong type of brackets e.g.
  ```
  exp{4}
  ```
Your turn!

http://tinyurl.com/bioboot-R1
Topics Covered:

Calling Functions
Getting help in R
Vectors and vectorization
Workspace and working directory
RStudio projects
Side-note: Use the code editor for R scripts
R scripts

• A simple text file with your R commands (e.g. day4.r) that contains your R code for one complete analysis

• **Scientific method:** complete record of your analysis

• **Reproducible:** rerunning your code is easy for you or someone else

• In RStudio, select code and type `<ctrl+enter>` to run the code in the R console

• **Key point:** [Save your R script!](#)
Side-note: RStudio shortcuts

Sends current line or selection to console (faster to type: command/ctrl+enter)

Sends entire file to console

Re-send the lines of code you last ran to the console (useful after edits)

Other RStudio shortcuts!
Up/Down arrows (recall cmds)
Ctrl + 2 (move cursor to console)
Ctrl +1 (move cursor to editor)
Rscript: Third way to use R

1. Terminal

2. RStudio

3. Rscript

From the command line!

> Rscript --vanilla my_analysis.R

# or within R: source(my_analysis.R)
Side-Note: R workspaces

- When you close RStudio, **SAVE YOUR .R SCRIPT**

- You can also save data and variables in an R workspace, but this is generally not recommended

- Exception: working with an enormous dataset

- Better to start with a clean, empty workspace so that past analyses don’t interfere with current analyses

- **rm(list = ls())** clears out your workspace

- You should be able to reproduce everything from your R script, so **save your R script, don’t save your workspace!**
Help from within R

- Getting help for a function
  
  ```
  > help("log")
  > ??log
  ```

- Searching across packages
  
  ```
  > help.search("logarithm")
  ```

- Finding all functions of a particular type
  
  ```
  > apropos("log")
  [7]  "SSlogis"  "as.data.frame.logical"  "as.logical"
     "as.logical.factor"  "dlogis"  "is.logical"
  [13]  "log"  "log10"  "log1p"  "log2"  "logLik"  "logb"
  [19]  "logical"  "loglin"  "plogis"  "print.logLik"  "qlogis"
     "rlogis"
What the function does in general terms

Description

log computes logarithms, by default natural logarithms. \( \log_{10} \) computes common (i.e., base 10) logarithms, and \( \log_2 \) computes binary (i.e., base 2) logarithms. The general form \( \log(x, \text{base}) \) computes logarithms with base base.

\[ \log(x) \text{ computes } \log(1+x) \text{ accurately also for } |x| < 1 \text{ (and less accurately when } x \text{ is approximately } -1). \]

exp computes the exponential function.

\[ \exp(x) - 1 \text{ accurately also for } |x| < 1. \]

How to use the function

Usage

\[
\begin{align*}
\log(x) & \quad \text{for } x \text{ a numeric or complex vector.} \\
\log_{10}(x) & \quad \text{for log to base 10.} \\
\log_2(x) & \quad \text{for log to base 2.} \\
\log(1+x) & \quad \text{for log of value x plus one.} \\
\exp(x) & \quad \text{for exp(x).} \\
\expml(x) & \quad \text{for expm1(x).} \\
\end{align*}
\]

What does the function need

Arguments

- \( x \): a numeric or complex vector.
- \( \text{base} \): a positive or complex number: the base with respect to which logarithms are computed. Defaults to \( = \exp(1) \).

What does the function return

Value

A vector of the same length as \( x \) containing the transformed values. \( \log(0) \) gives -Inf, and \( \log(x) \) for negative values of \( x \) is NaN. \( \exp(-\text{ Inf}) = 0 \).

For complex inputs to the log functions, the value is a complex number with imaginary part in the range \( \{\pi, -\pi\} \) which end of the range is used might be platform-specific.

S4 methods

\( \exp, \expml, \log, \log_{10}, \log_2 \) and \( \log_{10p} \) are S4 generic and are members of the \textbf{Math} group generic.

Note that this means that the S4 generic for \( \log \) has a signature with only one argument, \( x \), but that base can be passed to methods (but will not be used for method selection). On the other hand, if you only set a method for the \textbf{Math} group generic then base argument of \( \log \) will be ignored for your class.

Source

\( \log_{10}p \) and \( \expml \) may be taken from the operating system, but if not available there are based on the Fortran subroutine \texttt{dlnrel} by W. Fullerton of Los Alamos Scientific Laboratory (see \texttt{http://www.netlib.org/slatec/fnlib/dlnrel}) and (for small \( x \)) a single Newton step for the solution of \( \log_{10p}(y) = x \) respectively.

References


See Also

\textbf{Trig, sqrt, Arithmetic}

Examples

\[
\begin{align*}
\log(\exp(3)) & \quad \text{for } \log(\exp(3)). \\
\log_{10}(1e7) & \quad \text{# } 7 \\
x & \quad 10^{-(1\times2+1:9)} \\
cbind(x, \log(1+x), \log_{10p}(x), \exp(x) - 1, \expml(x)) & \quad \text{for } \log_{10p}(x). \\
\end{align*}
\]

[Package base version 3.0.1 Index]
Optional Exercise

Use R to do the following. Create a new script to save your work and code up the following four equations:

\[
\begin{align*}
1 + 2(3 + 4) \\
\ln(4^3 + 3^{2+1}) \\
\sqrt{(4+3)(2+1)} \\
\left(\frac{1+2}{3+4}\right)^2
\end{align*}
\]
Learning Resources

- **TryR**. An excellent interactive online R tutorial for beginners.  
  <http://tryr.codeschool.com/>

- **RStudio**. A well designed reference card for RStudio.  
  <https://help.github.com/categories/bootcamp/>

- **DataCamp**. Online tutorials using R in your browser.  
  <https://www.datacamp.com/>

  <http://r4ds.had.co.nz/>