Introduction to R

Marcus W. Beck[1] Todd D. O'Brien[2]

[1]ORISE, USEPA NHEERL Gulf Ecology Division
Email: beck.marcus@epa.gov

[2]NOAA/NMFS Copepod Project
Email: todd.obrien@noaa.gov

What you'll learn in this intro

- Course expectations and pre-worskhop materials
- What is R?
- What's possible with R?
- R basics
  - Installation
  - Command-line interface
  - Coding basics
  - Functions and objects
  - Data import and manipulation
- Help!

Course expectations

The November 17^{th} workshop will provide you with a set of tools for evaluating time series data from SWMP

Preparation for the workshop:

- Review pre-workshop toolkit materials
- Bring a computer with R version 3.0.0 or later
- Optionally, install RStudio in addition to R
- Have a basic proficiency using R, what this means:
  - You do not have to be an expert!
  - Understand the basics of a command-line interface
  - Know how to open R, load a script, execute functions
  - Know how to save your work
- Install the 'SWMPr' package after installing R

Course expectations

This is not an R training workshop...

...but we will be using R exclusively to handle SWMP data

You will receive an overview of the theory behind time series analysis

We will use an R package developed to work with SWMP data

A package is a collection of functions written by others that can be installed within R

This package can automatically handle common problems working with SWMP data and time series, it is designed to make your life easier!

Course expectations

The pre-workshop toolkit includes:

- R installation instructions - *R_install_guide.pdf*
- Intro to R (current document) - *intro_to_R.pdf*
- Basics of data analysis with R - *r_for_data_analysis.pdf*
- Installing and working with the SWMPr package - *intro_to_swmpr.pdf*

The final item is optional as we will cover the content in the workshop, but you should have the SWMPr package installed prior to the 17th

Please note that webinar attendees will have limited interaction with the instructors, although we will have moderators handling questions.

Course expectations

Copies of all instructional materials will be made available on the course website: copepod.org/nerrs-swmp-workshop/

Physical attendees will also receive a flashdrive with the course materials

The instructors can also be contacted with questions prior to the workshop:

- For R questions including toolkit and installation - Marcus Beck, beck.marcus@epa.gov
- For time series analysis questions - Todd O'Brien, todd.obrien@noaa.gov

Course expectations

Finally, the presentation materials combine content with R code and R output

R code that can be executed will look like this:

```
# here's some R code
rnorm(10)
```

The output will look similar to this in R (without '##'):

```
##  [1]  0.5  1.6 -0.4 -0.9  0.5  0.8  1.5  0.7 -0.4  1.3
```

When applicable, R scripts are provided that contain only the executable code within each training module. You can use these in R as you read each presentation.

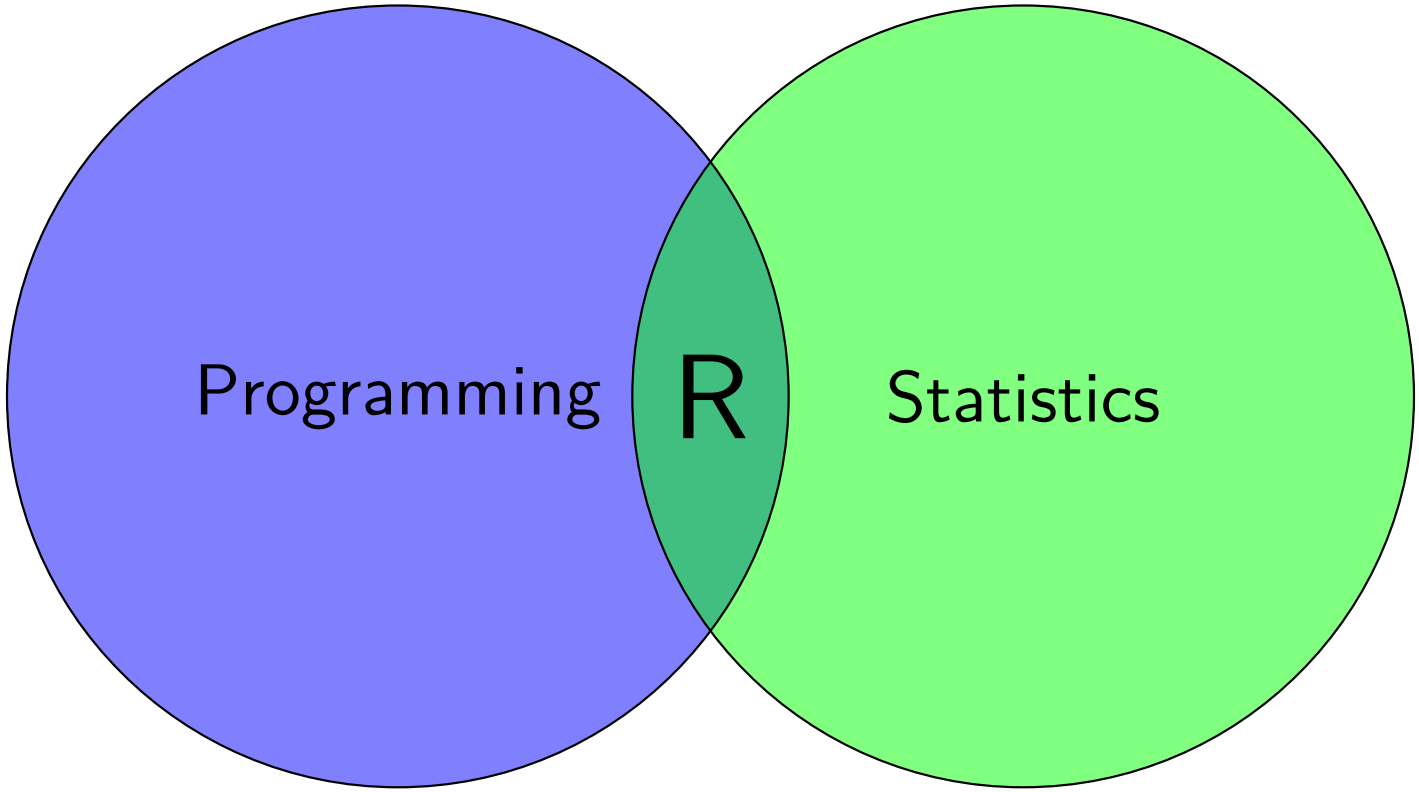
What is  ?

R is a computer language that allows the user to program algorithms and use tools that have been programmed by others

Different from other statistics software because it provides both tools for analysis and it is also a programming language...

You do not have to be a progammer to use R!

What is R?

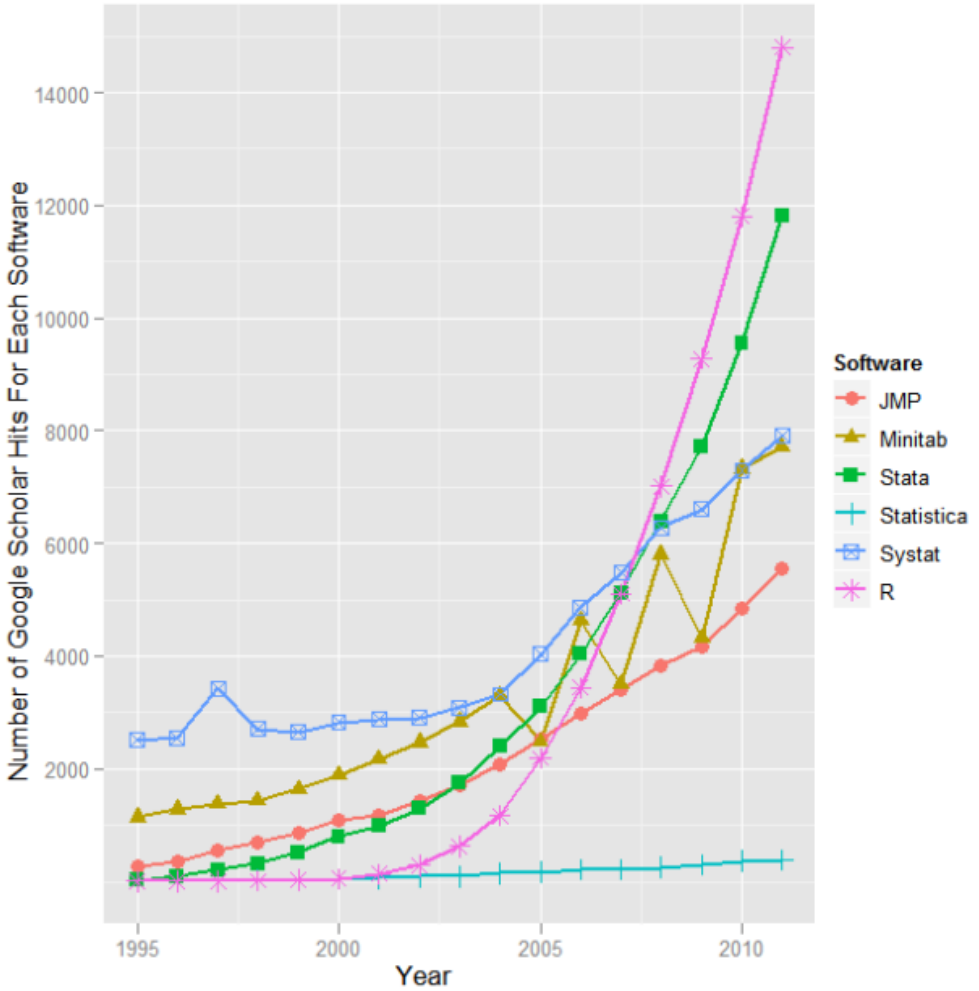


R is both... this creates a steep learning curve.

What is R ?

R is becoming the statistical software of choice

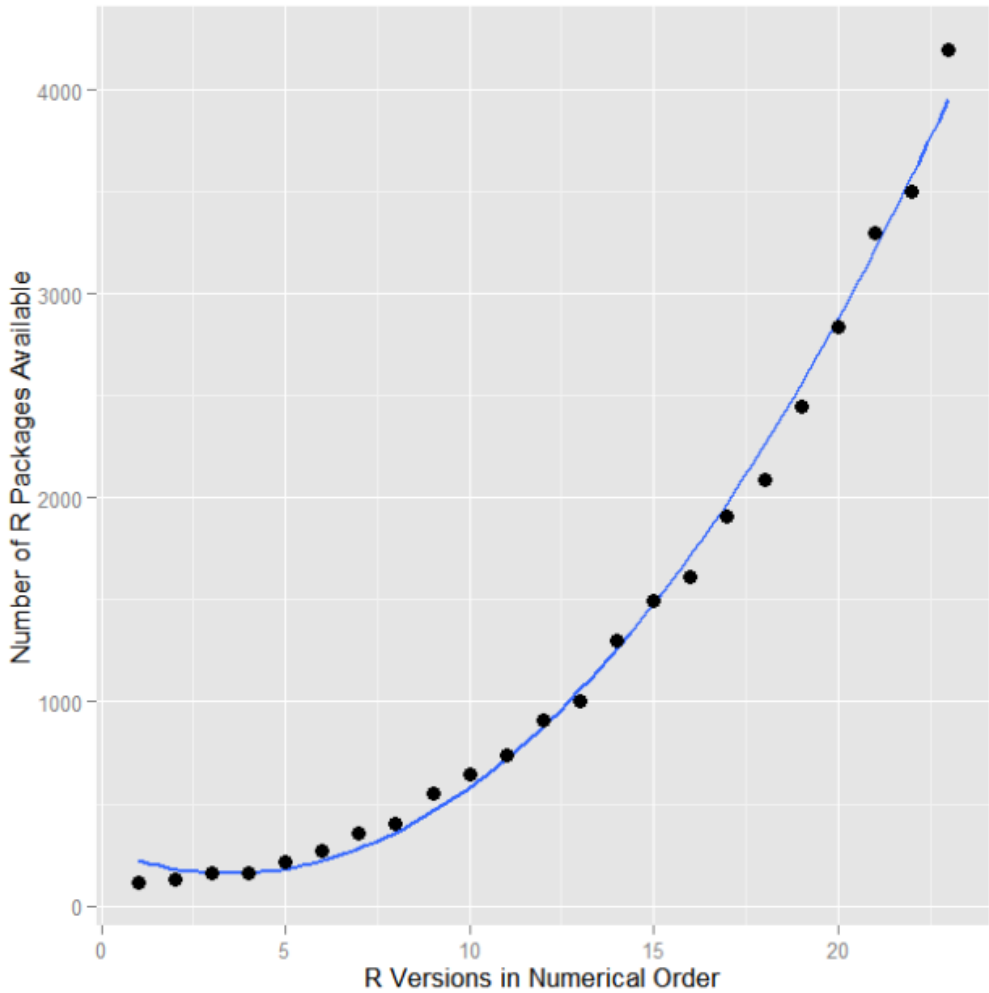
Plot of Google scholar hits over time for different software packages [r4stats.com]



What is R ?

R is becoming the statistical software of choice

Exponential growth in number of contributed packages [r4stats.com]



What's possible with  ?

R is incredibly flexible, if you want something done, someone else has written the code...

R is open-source software, which mean it's free and is supported by a large network of contributors - the Comprehensive R Network [CRAN]

CRAN is a collection of sites which carry identical material, consisting of the R distribution(s), the contributed extensions, documentation for R, and binaries [R FAQ]

Basically a repository of R utilities that others have written as well as the source files for R - the CRAN task views contain descriptions of contributed packages by category

What's possible with R?

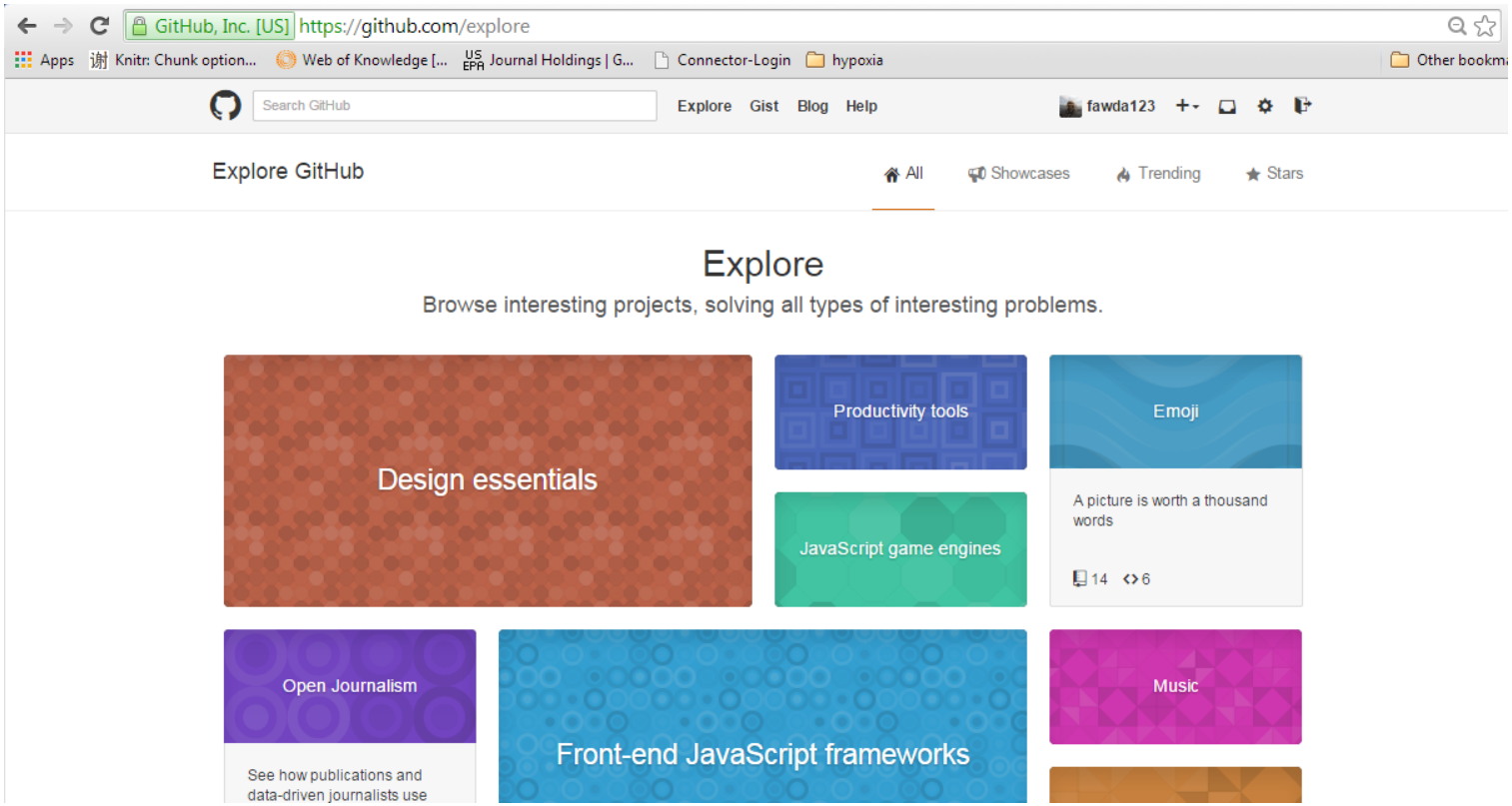
CRAN task views

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics

What's possible with R?

Github has also been increasingly used to store packages online - like an informal CRAN:



What's possible with R?

R comes with a base package that is included in installation, others are downloaded as needed, e.g.,

```
# download from CRAN
install.packages('ggplot2')
```

The base package will be sufficient for most of your needs - includes arithmetic, input/output, basic programming support, graphics, etc.

Once a package is installed, it must be loaded to use its functions:

```
# load ggplot2
library(ggplot2)
```

What's possible with  ?

Or you can download packages from Github, which requires using the devtools package:

```
# download devtools from CRAN
install.package('devtools')

# load devtools
library(devtools)

# install SWMPr from Github, load
install_github('fawda123/SWMPr')
library(SWMPr)
```

What's possible with  ?

Each package may also come with a demonstration

This provides a neat way to see what an R package has to offer

To see a list of packages with demonstrations, run this code:

```
# view packages with demos
demo(package = .packages(all.available = TRUE))
```

To view a demonstration of basic graphic capabilities in R:

```
# view a demo for the graphics package
demo(graphics)
```

What's possible with R?

Each package comes with extensive help documentation

Help files for a package:

```
# view help files for a package
help(package = 'ggplot2')
```

Or for an individual function in a package:

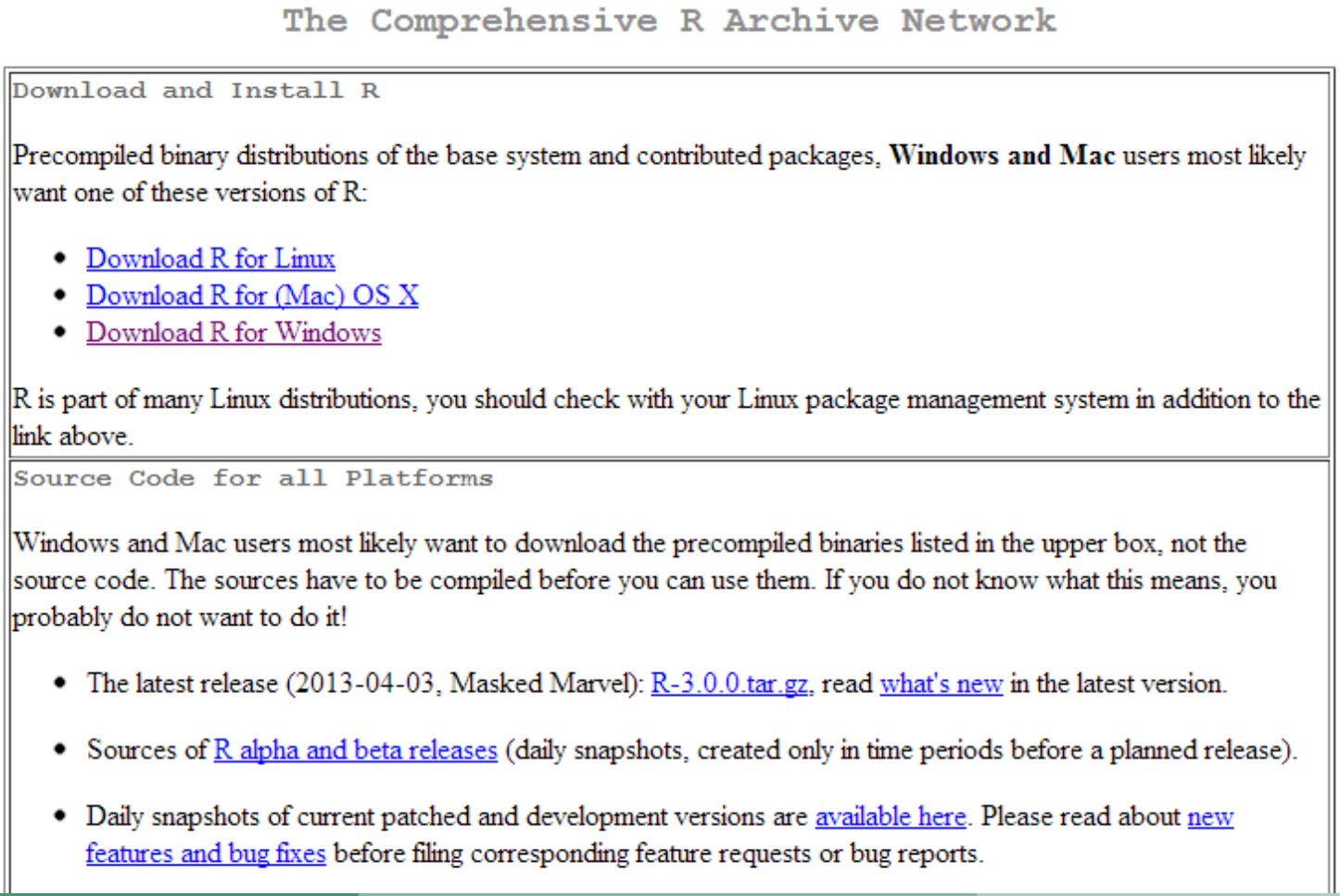
```
# get help file
help(mean, package = 'base')

# or do this
?mean
```

basics

Following the instructions in our installation guide for step-by-step directions

Or visit r-project.org and follow directions



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.

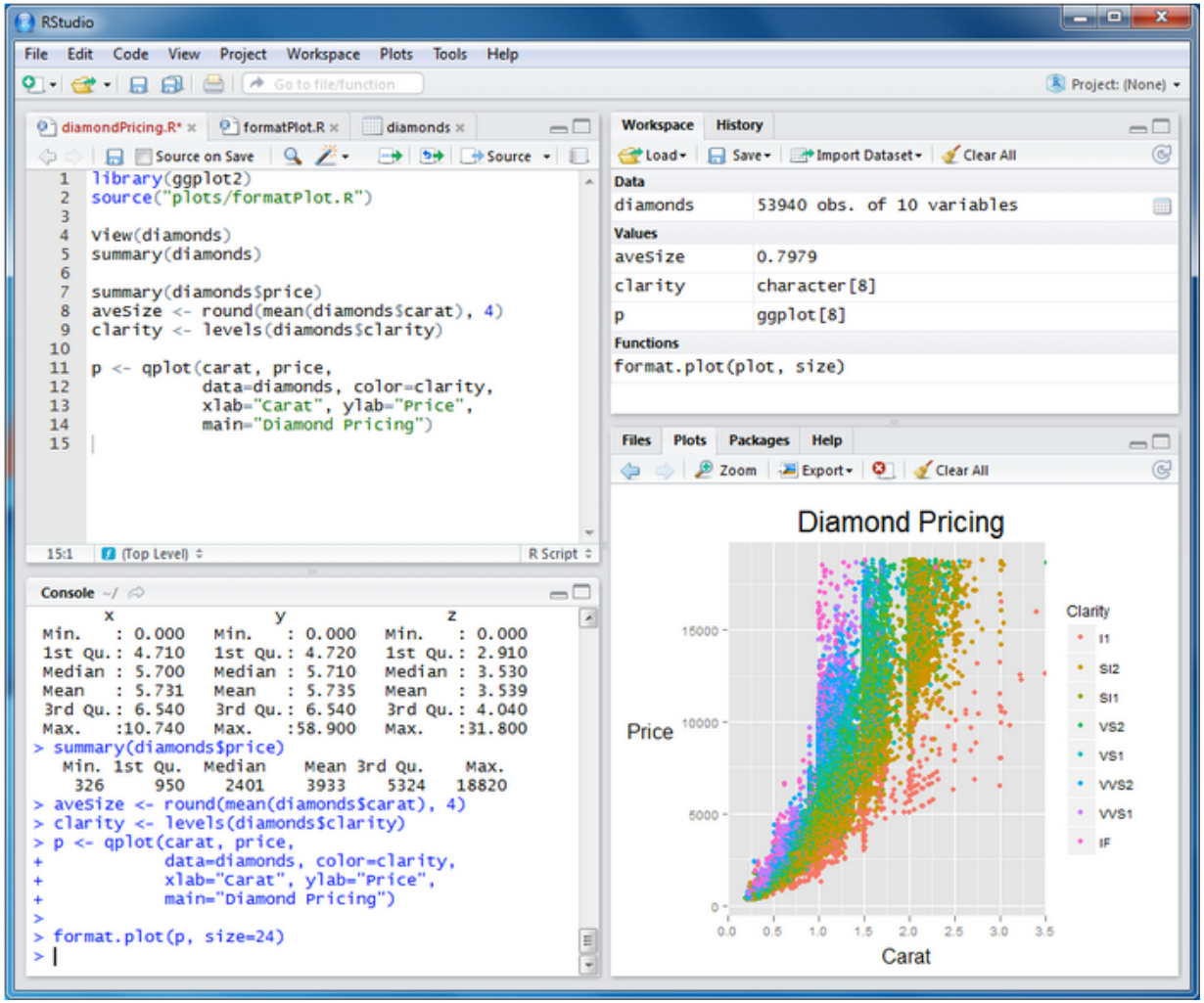
Source Code for all Platforms

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 (2013-04-03, Masked Marvel): R-3.0.0.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.

basics

RStudio is also recommended but not required



basics

How is R different from Excel? R is a command-line interface

```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-w64-mingw32/x64 (64-bit)

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

  Natural language support but running in an English locale

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

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

> |
```

What next??

basics

Lines of code are executed by R at the prompt (>)

Enter the code and press enter, the output is returned

```
print('hello world!')

## [1] "hello world!"

2 + 2

## [1] 4

(2 + 2) / 4

## [1] 1
```

basics

A disadvantage of code is that everything entered must be 100 % correct

```
2 + a

## Error:  object 'a' not found

a

## Error:  object 'a' not found
```

But this provides an explicit documentation of your workflow...

...your code is a living document of your analyses.

basics

Assigning data to R objects is critical for analysis

Assignment is possible using <- or =

```
a <- 1
2 + a

## [1] 3

a = 1
2 + a

## [1] 3
```

basics

More complex assignments are possible

```
a <- c(1, 2, 3 , 4)
a

## [1] 1 2 3 4

a <- seq(1, 4)
a

## [1] 1 2 3 4

a <- c('a', 'b', 'c')
a

## [1] "a" "b" "c"
```

basics

Anatomy of a function - functions perform tasks for you, much like in Excel

function(arguments)

```
c(1, 2) # concatenate function to combine value
```

```
## [1] 1 2
```

```
mean(c(1, 2)) # mean function
```

```
## [1] 2
```

```
seq(1, 4) # create a sequence of values
```

```
## [1] 1 2 3 4
```

basics

Understanding classes of R objects is necessary for analysis

An object is a variable of interest that will always have a class

Most common are 'numeric' or 'character' classes

```
class(1)

## [1] "numeric"

class('1')

## [1] "character"
```

'Factors' are also common, define categorical variables

basics

Different types of methods are assigned to different class types

The class also imposes limits on how it interacts with other class

For example, we cannot add add two objects with different classes:

```
# this does not work
'1' + 1

## Error:  non-numeric argument to binary operator
```

There is no 'addition' method for character objects...

basics

Objects (and their classes) are stored in the computer's memory in different ways - aka the workspace for your R session

The most common structures are 'vectors' and 'data.frames'

Vectors are a collection of objects of the same class, whereas a data frame is analogous to a table with rows and columns (e.g., collection of vectors)

```
a <- c(1,2) # numeric vector
a

## [1] 1 2

b <- c('a', 'b') # character
b

## [1] "a" "b"
```

```
# data frame with both
c <- data.frame(a,b)
c

##   a b
## 1 1 a
## 2 2 b
```

basics

How are data imported into R?

R needs to know where the data are located on your computer:

```
# the location where you want to work
setwd('C:/Projects/my_data/')
```

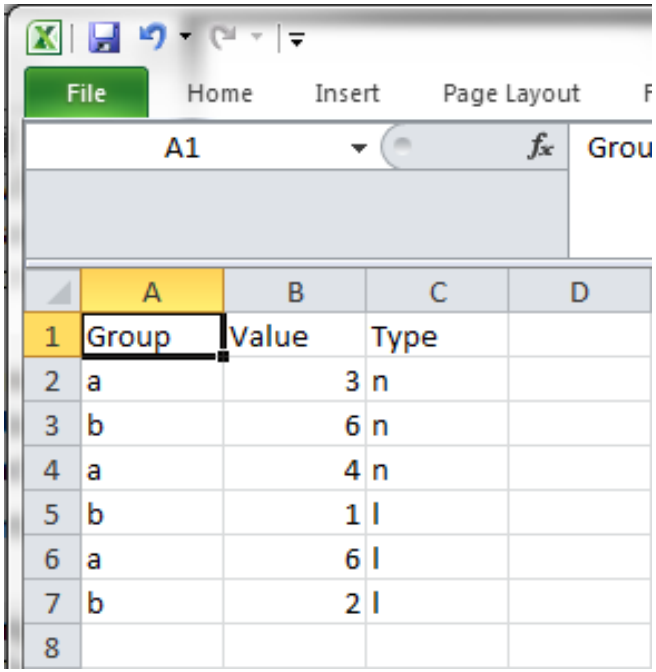
This establishes a 'working directory' for data import/export

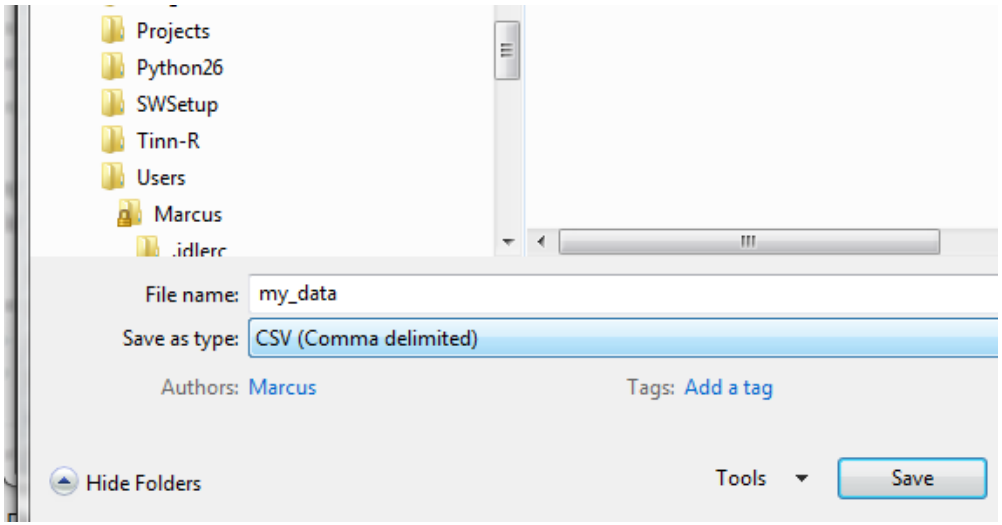
R can import almost any type of data but 'spreadsheet' or text-based files are most common

basics

How are data imported into R?

The easiest approach is to save an Excel file as .csv or .txt file





basics

How are data imported into R?

Use the read.table or read.csv functions to import the data, must be in your working directory

```
# use read.csv if .csv
dat <- read.csv('my_data.csv', header = T)

# use read.table if .txt
dat <- read.table('my_data.txt', sep = ',', header = T)
```

basics

Imported data can be viewed several ways, view the whole object or parts

Rows or columns can be obtained by indexing with brackets separated by a comma: data[row, column]

```
dat

##   Group Value Type
## 1     a     3    n
## 2     b     6    n
## 3     a     4    n
## 4     b     1    l
## 5     a     6    l
## 6     b     2    l
```

```
dat[1, ] # row 1

##   Group Value Type
## 1     a     3    n

dat[, 2] # column 2

## [1] 3 6 4 1 6 2

dat[4, 1] # row 4, column 1

## [1] b
## Levels: a b
```

basics

Imported data can be viewed several ways, view the whole object or parts

Access using column names or the attach function

```
dat$Value
```

```
## [1] 3 6 4 1 6 2
```

```
dat[, 'Value']
```

```
## [1] 3 6 4 1 6 2
```

```
attach(dat)
Value
```

```
## [1] 3 6 4 1 6 2
```

basics

Where to go for help?

- A user-friendly intro to R
- Several good introductory texts are available - Zuur et al. 2009. A Beginner's Guide to R. Springer.
- R cheatsheet
- Google is your friend
- Help files for each function using '?function' - may or may not be helpful
- An intro to R - very detailed
- Ask us!