

Introduction to R for Biologists

Day 1 – Intro to R and the Tidyverse Ecosystem

*Several slides have been contributed by or modified from slides by Dr. Claus Wilke

Day 1 Outline

1. How to get set up using R
2. How and why to use RStudio & R Markdown (.Rmd)
3. Basics of programming
 - Data types
 - Functions
 - Troubleshooting
4. Intro to the Tidyverse
 - Tidy vs untidy data
 - Tidyverse-specific functions

R: The premier data analysis and visualization platform

<https://cran.r-project.org/>



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.

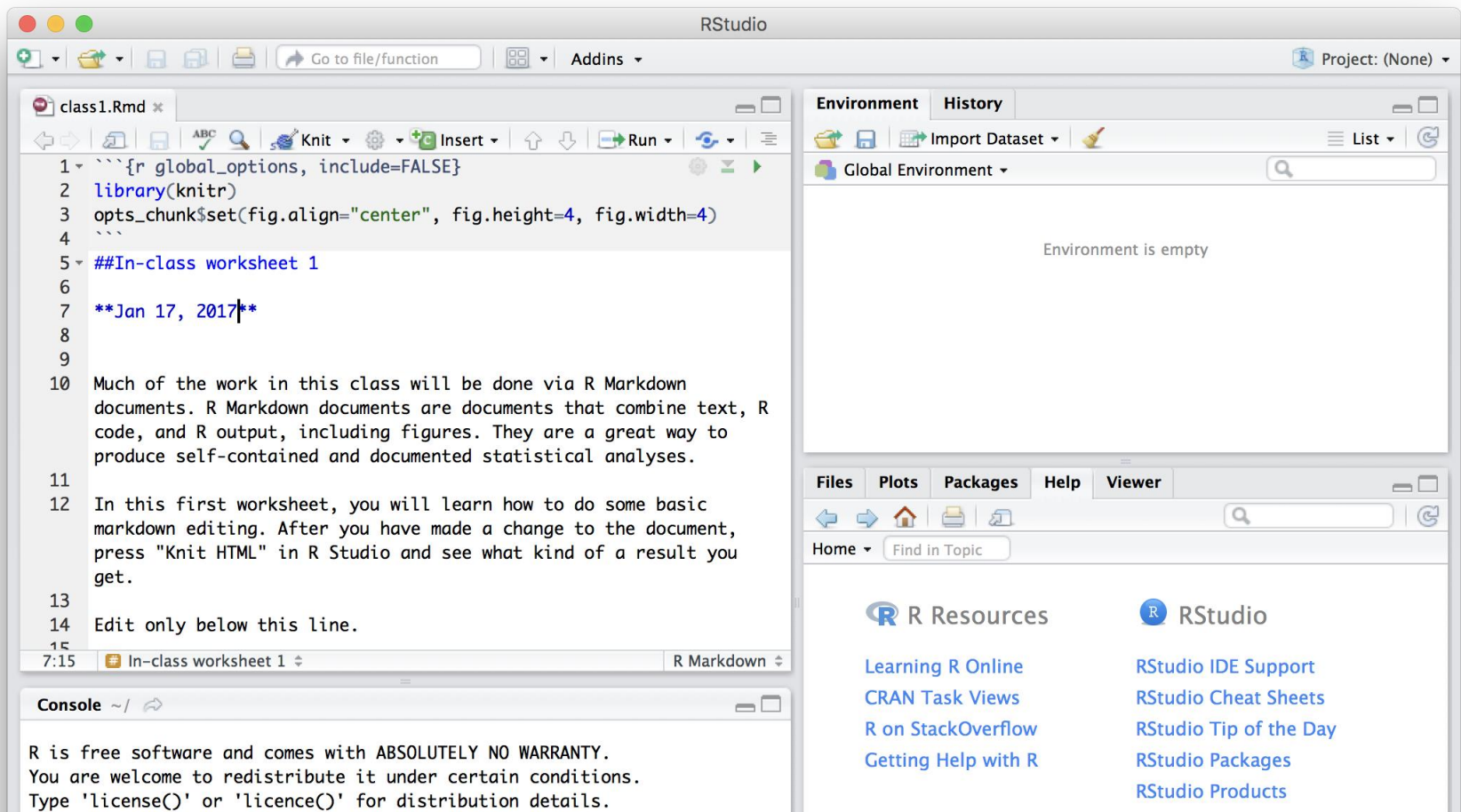
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper

R Studio:

A nice user interface for R

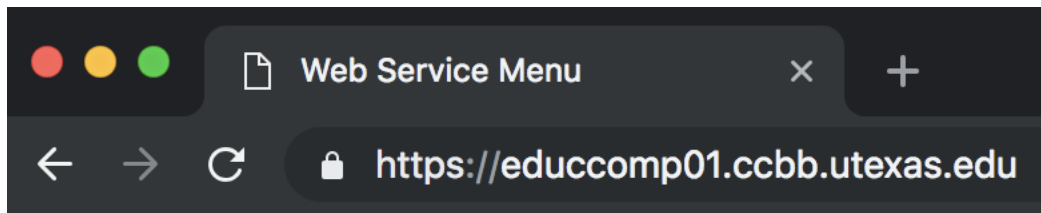
<https://www.rstudio.com/products/rstudio/download/>



Access R Studio through your web browser

1. <https://gsafcomp01.ccb.b.utexas.edu/>
2. <https://gsafcomp02.ccb.b.utexas.edu/>

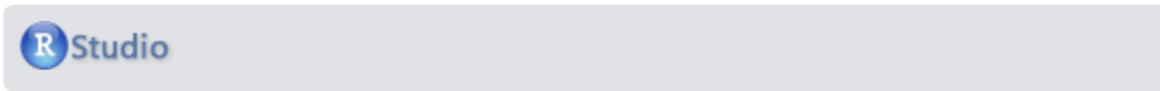
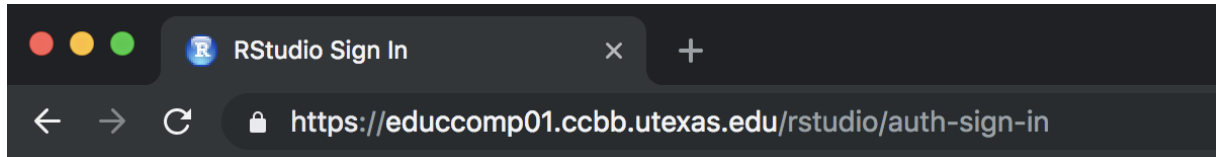
Select RStudio



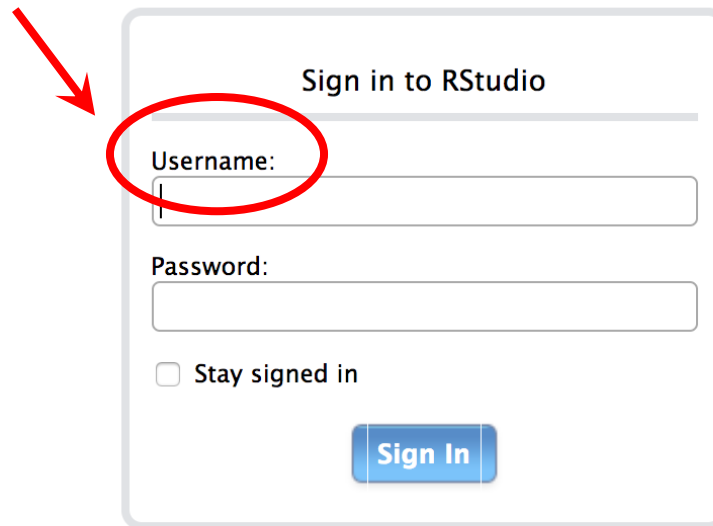
Please choose one of the following applications:

- [RStudio](#) ←
- [Jupyterhub](#)

Sign in with your student# and password

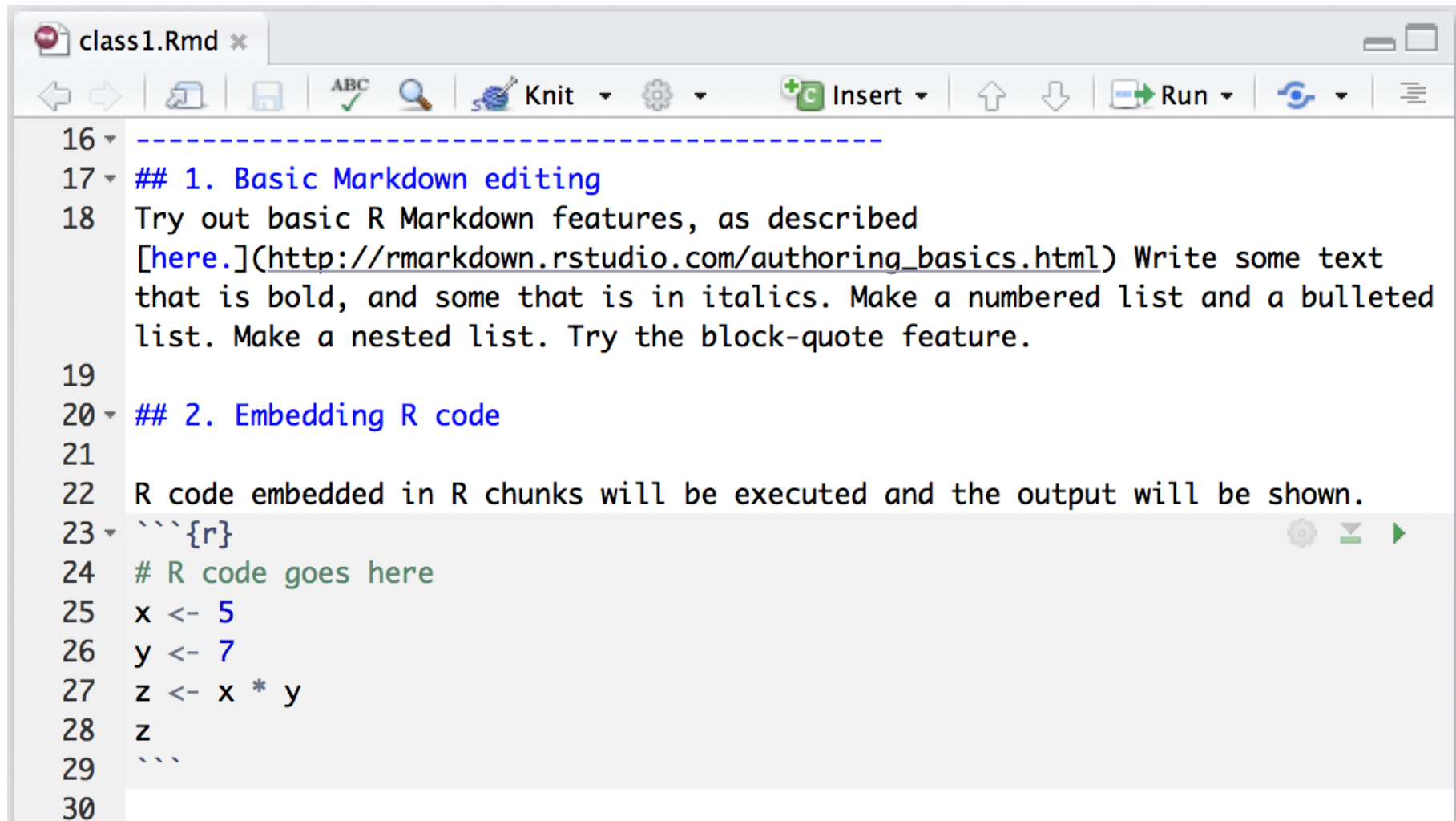


Refer to class email for your individual username

A screenshot of the 'Sign in to RStudio' form. The form is titled 'Sign in to RStudio' and contains two input fields: 'Username:' and 'Password:'. A red arrow points to the 'Username:' field, which is also circled in red. Below the password field is a checkbox labeled 'Stay signed in'. At the bottom of the form is a blue button labeled 'Sign In'.

R Markdown

R Markdown: Writing documents with embedded R code



The screenshot shows the RStudio editor interface with a file named 'class1.Rmd' open. The editor displays R Markdown code with line numbers 16 through 30. The code includes a section header, a paragraph of text with a link, another section header, and an R code chunk. The R code chunk is highlighted with a light blue background and contains three lines of code: `x <- 5`, `y <- 7`, and `z <- x * y`. The output of the R code is not visible in this view.

```
16 -----
17 ## 1. Basic Markdown editing
18 Try out basic R Markdown features, as described
   [here.](http://rmarkdown.rstudio.com/authoring_basics.html) Write some text
   that is bold, and some that is in italics. Make a numbered list and a bulleted
   list. Make a nested list. Try the block-quote feature.
19
20 ## 2. Embedding R code
21
22 R code embedded in R chunks will be executed and the output will be shown.
23 ```{r}
24 # R code goes here
25 x <- 5
26 y <- 7
27 z <- x * y
28 z
29 ```
30
```

R Markdown:

Writing documents with embedded R code

1. Basic Markdown editing

Try out basic R Markdown features, as described [here](#). Write some text that is bold, and some that is in italics. Make a numbered list and a bulleted list. Make a nested list. Try the block-quote feature.

2. Embedding R code

R code embedded in R chunks will be executed and the output will be shown.

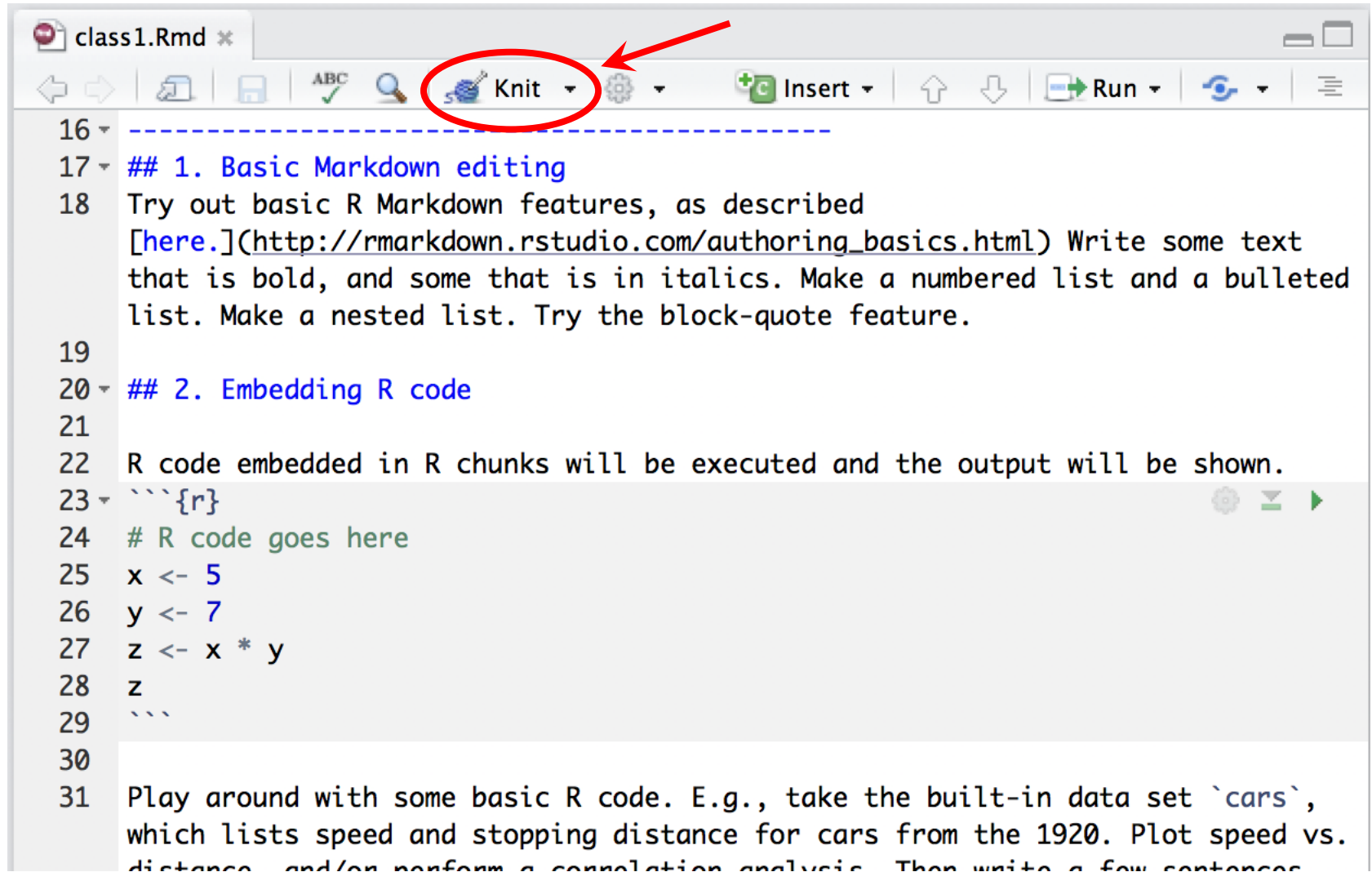
```
# R code goes here
```

```
x <- 5  
y <- 7  
z <- x * y  
z
```

```
## [1] 35
```

Play around with some basic R code. E.g., take the built-in data set `cars`, which lists speed and stopping distance for cars from the 1920. Plot speed vs. distance, and/or perform a correlation analysis. Then write a few sentences describing what you see.

We convert R Markdown to HTML by “knitting” the Markdown file



Markdown basics

http://rmarkdown.rstudio.com/authoring_basics.html

normal text

italics

****bold****

Header 1

Header 2

List:

1. Item 1
2. Item 2
3. Item 3



normal text

italics

bold

Header 1

Header 2

List:

1. Item 1
2. Item 2
3. Item 3

Markdown basics

Embedded R code will be evaluated and printed

```
```{r}  
head(cars)
plot(cars$speed, cars$dist)
```
```

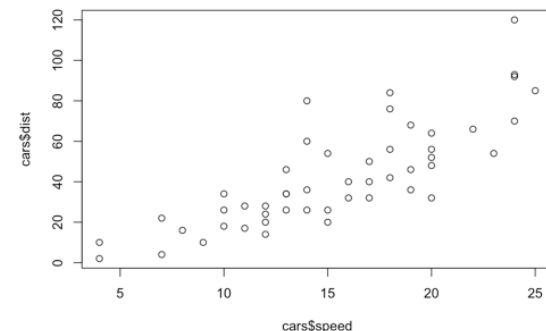


Embedded R code will be evaluated and printed

```
head(cars)
```

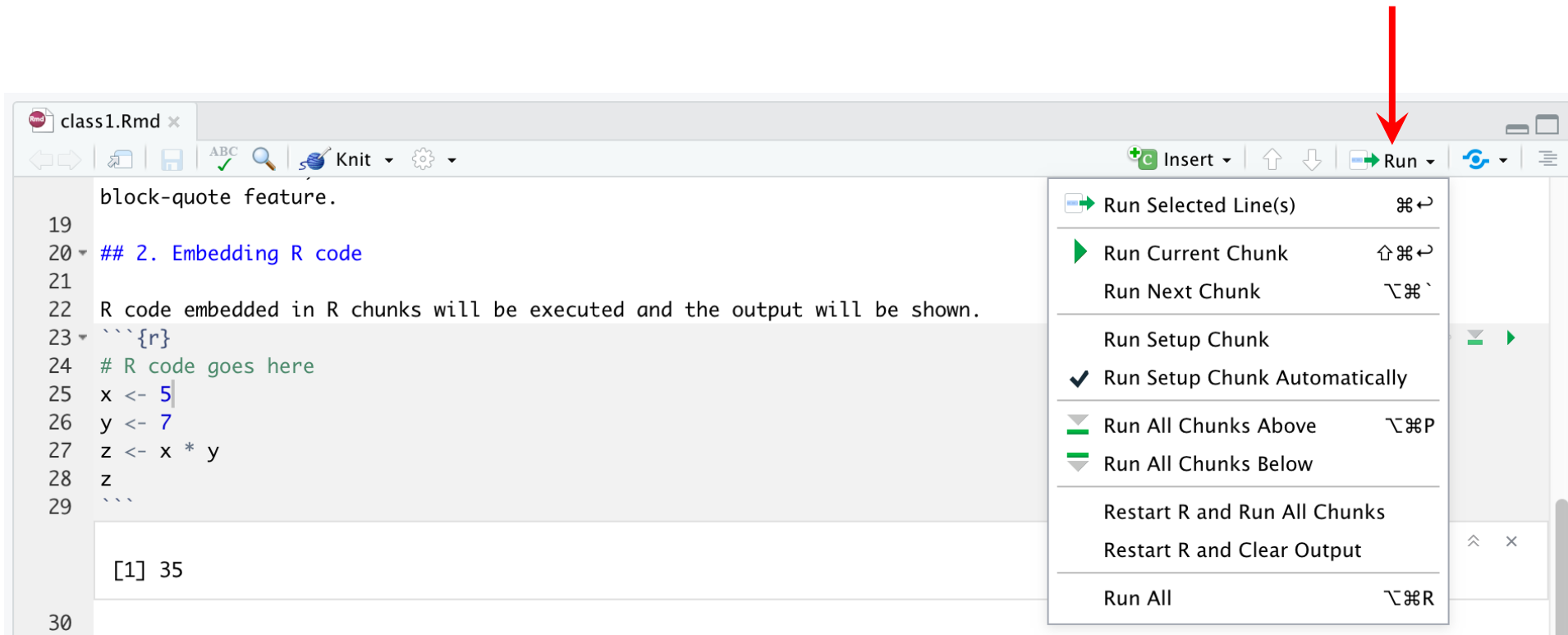
```
##  speed dist  
## 1     4    2  
## 2     4   10  
## 3     7    4  
## 4     7   22  
## 5     8   16  
## 6     9   10
```

```
plot(cars$speed, cars$dist)
```



Different ways to execute code in RStudio

Press the “Run” button



The screenshot shows the RStudio interface with a file named `class1.Rmd` open. The code in the editor is as follows:

```
19 block-quote feature.  
20 ## 2. Embedding R code  
21  
22 R code embedded in R chunks will be executed and the output will be shown.  
23 ```{r}  
24 # R code goes here  
25 x <- 5  
26 y <- 7  
27 z <- x * y  
28 z  
29 ```
```

The output of the code chunk is displayed below the editor:

```
[1] 35
```

A red arrow points to the **Run** button in the toolbar. The dropdown menu for the **Run** button is open, showing the following options:

- Run Selected Line(s)
- Run Current Chunk
- Run Next Chunk
- Run Setup Chunk
- Run Setup Chunk Automatically
- Run All Chunks Above
- Run All Chunks Below
- Restart R and Run All Chunks
- Restart R and Clear Output
- Run All

Highlight code you want to execute and press ctrl+Enter (cmd+Enter on Macs)

R code embedded in R chunks will be executed and the output will be shown.

```
``{r}  
# R code goes here  
x <- 5  
y <- 7  
z <- x * y  
z  
``
```



Console

Terminal x

Jobs x

~/Desktop/projects/ ↗

```
> x <- 5  
> y <- 7  
> z <- x * y  
> z  
[1] 35  
> |
```


Place pointer on line of code you want to execute, press ctrl+Enter (cmd+Enter on Macs)

R code embedded in R chunks will be executed and the output will be shown.

```
`` `{r}  
# R code goes here  
x <- 5  
y <- 7  
z <- x * y  
z  
`` `
```



Console

Terminal x

Jobs x

~/Desktop/projects/ ↗

```
> z <- x * y  
> |
```

Use ctrl+Shift+Enter (cmd+Shift+Enter on Macs) to execute an entire code chunk

R code embedded in R chunks will be executed and the output will be shown.

```
`` `{r}`  
# R code goes here  
x <- 5  
y <- 7  
z <- x * y  
z  
`` `
```



Console

Terminal x

Jobs x

~/Desktop/projects/ ↩

```
> x <- 5  
> y <- 7  
> z <- x * y  
> z  
[1] 35  
> |
```

Shortcuts for coding

- **Ctrl+Shift+M** (Cmd+Shift+M on Macs) produces a pipe operator `%>%` (will be used with the tidyverse)
- **Ctrl+Shift+C** (Cmd+Shift+C on Macs) will comment/uncomment a line or multiple lines
- **Tab** and **Shift+Tab** will indent and un-indent lines, respectively

R Programming Basics

Assignments, numbers, vectors

```
> x <- 5
```

Assign number 5 to variable x

```
> x
```

```
[1] 5
```

```
> 5*x^2+7
```

Calculate $5x^2+7$

```
[1] 132
```

```
> y <- c(1, 2, 3, 4, 5)
```

Create vector, assign
to variable y

```
> y
```

```
[1] 1 2 3 4 5
```

```
> x*y
```

Multiply each element
in y with the number in x

```
[1] 5 10 15 20 25
```

Strings

A string contains text:

```
> name <- "Rachael Cox"
> name
[1] "Rachael Cox"
```

A vector of strings:

```
> animals <- c("cat", "mouse", "mouse",
"cat", "rabbit")
> animals
[1] "cat"      "mouse"    "mouse"    "cat"
"rabbit"
```

Factors

Factors keep track of distinct categories (levels) in a vector:

```
> animals  
[1] "cat"      "mouse"    "mouse"    "cat"  
"rabbit"
```

```
> factor(animals)  
[1] cat      mouse    mouse    cat      rabbit  
Levels: cat mouse rabbit
```

Data frames

We use data frames to store data sets with multiple variables:

```
> pets <- data.frame(  
  family = c(1, 2, 3, 4, 5),  
  pet = animals  
)
```

```
> pets
```

| | family | pet |
|---|--------|-------|
| 1 | 1 | cat |
| 2 | 2 | mouse |
| 3 | 3 | mouse |
| 4 | 4 | cat |

Data frames

We access individual columns in a data frame with \$ + the column name:

```
> pets$family  
[1] 1 2 3 4 5
```

```
> pets$pet  
[1] cat      mouse    mouse    cat      rabbit  
Levels: cat mouse rabbit
```

Data frames

R has many built-in data frames:

```
> cars
```

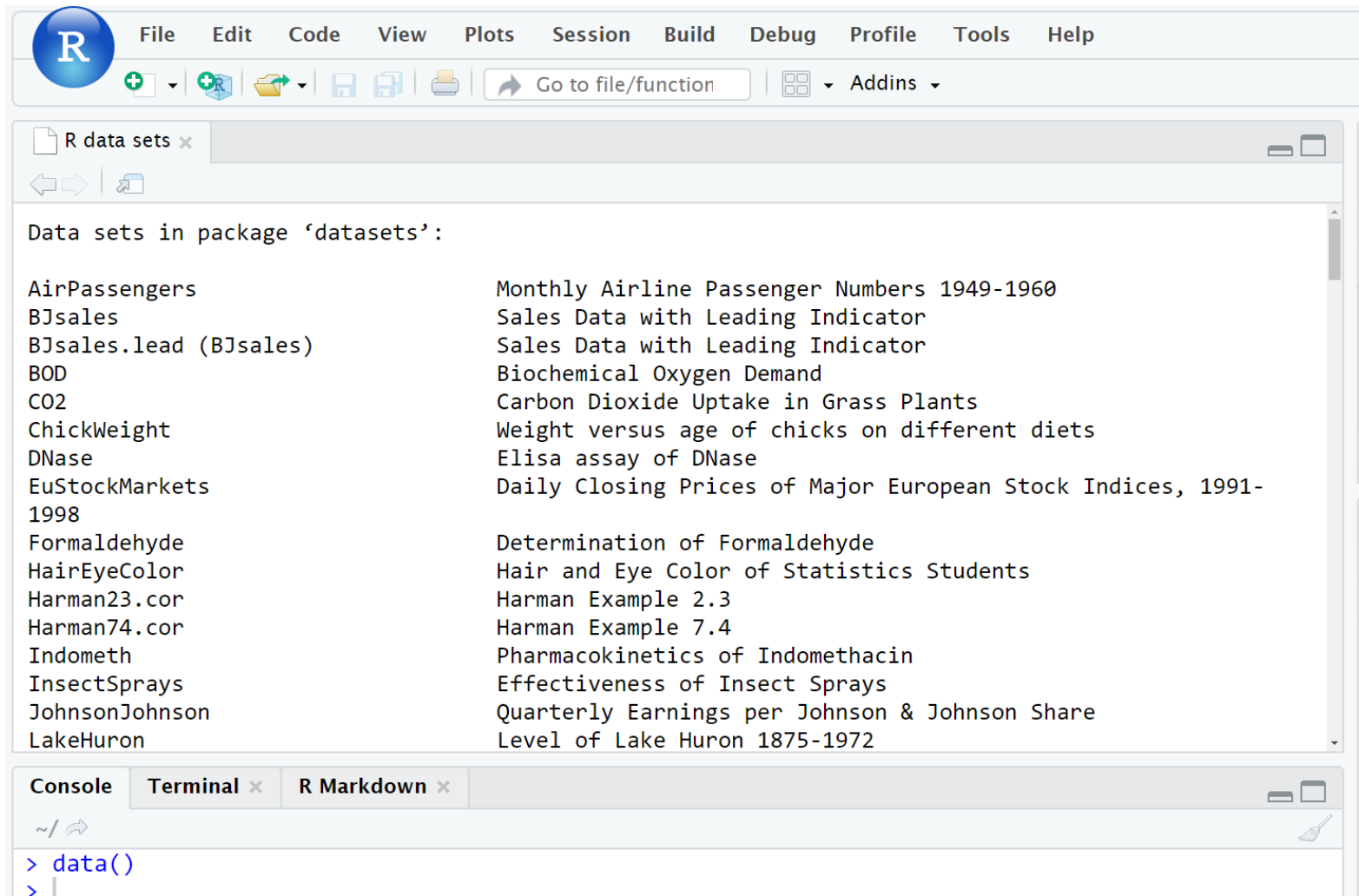
| | speed | dist |
|---|-------|------|
| 1 | 4 | 2 |
| 2 | 4 | 10 |
| 3 | 7 | 4 |
| 4 | 7 | 22 |
| 5 | 8 | 16 |
| 6 | 9 | 10 |
| 7 | 10 | 18 |
| 8 | 10 | 26 |
| 9 | 10 | 34 |

```
> chickwts
```

| | weight | feed |
|-----|--------|-----------|
| 1 | 179 | horsebean |
| 2 | 160 | horsebean |
| 3 | 136 | horsebean |
| 4 | 227 | horsebean |
| ... | ... | ... |
| 11 | 309 | linseed |
| 12 | 229 | linseed |
| 13 | 181 | linseed |
| 14 | 141 | linseed |

Data frames

Available built-in datasets can be accessed with `data()`



Data frames

Data set information can be accessed with `?dataset`

The screenshot shows the RStudio interface. In the left pane, the 'R data sets' list has 'cars' highlighted with a red circle. A red arrow points from this circle to the 'Speed and Stopping Distances of Cars' help page in the right pane. The 'Help' tab in the top navigation bar is also circled in red. The console at the bottom shows the command `?cars` being entered.

Environment **History** **Connections**

Files **Plots** **Packages** **Help** **Viewer**

R: Speed and Stopping Distances of Cars

`cars {datasets}`

Speed and Stopping Distances of Cars

Description

The data give the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s.

Usage

```
cars
```

Format

A data frame with 50 observations on 2 variables.

```
[,1] speed numeric Speed (mph)
[,2] dist  numeric Stopping distance (ft)
```

Source

Ezekiel, M. (1930) *Methods of Correlation Analysis*. Wiley.

Console **Terminal** **R Markdown**

```
> data()
> ?data
> ?cars
> |
```

Data frames

The `head()` function shows the first few lines of a data frame:

```
> head(cars)
  speed  dist
1     4     2
2     4    10
3     7     4
4     7    22
5     8    16
6     9    10
>
```

Functions

Functions are called in the format `function(argument)`

```
> head(cars)
```



Function name



First argument

Functions

Functions can have any number of required arguments or optional arguments

> head(cars, 8)

Function name

First argument (required)

Second argument (optional; default = 6)

Functions

`head(cars, 8)` will show the first 8 lines of the data frame instead of the default 6:

```
> head(cars, 8)
```

```
  speed dist
```

```
1      4     2
```

```
2      4    10
```

```
3      7     4
```

```
4      7    22
```

```
5      8    16
```

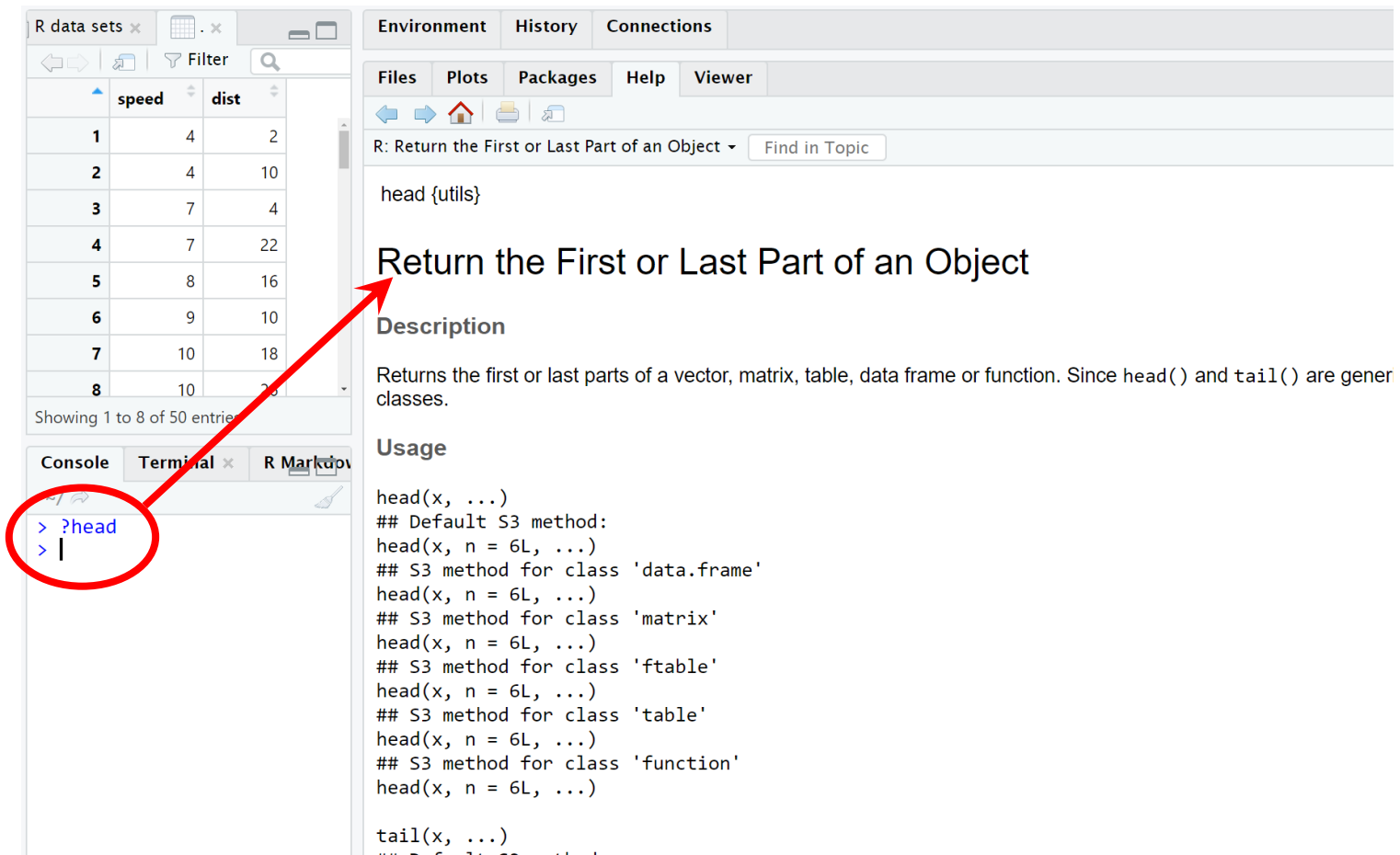
```
6      9    10
```

```
7     10    18
```

```
8     10    26
```


Functions

More information about what a function does and/or requires can be accessed with `?function`



The screenshot shows the RStudio interface. On the left, a data frame with columns 'speed' and 'dist' is displayed. The 'Console' tab is active, showing the command `> ?head` entered, with a red circle around it. A red arrow points from this command to the help page for `head()` on the right. The help page includes the title 'Return the First or Last Part of an Object', a description, and usage examples.

| | speed | dist |
|---|-------|------|
| 1 | 4 | 2 |
| 2 | 4 | 10 |
| 3 | 7 | 4 |
| 4 | 7 | 22 |
| 5 | 8 | 16 |
| 6 | 9 | 10 |
| 7 | 10 | 18 |
| 8 | 10 | 20 |

Showing 1 to 8 of 50 entries

Environment **History** **Connections**

Files **Plots** **Packages** **Help** **Viewer**

R: Return the First or Last Part of an Object

head {utils}

Return the First or Last Part of an Object

Description

Returns the first or last parts of a vector, matrix, table, data frame or function. Since `head()` and `tail()` are generic functions.

Usage

```
head(x, ...)  
## Default S3 method:  
head(x, n = 6L, ...)  
## S3 method for class 'data.frame':  
head(x, n = 6L, ...)  
## S3 method for class 'matrix':  
head(x, n = 6L, ...)  
## S3 method for class 'ftable':  
head(x, n = 6L, ...)  
## S3 method for class 'table':  
head(x, n = 6L, ...)  
## S3 method for class 'function':  
head(x, n = 6L, ...)  
  
tail(x, ...)
```

Functions

?function has argument information

The screenshot shows the RStudio interface. On the left, a data frame with columns 'speed' and 'dist' is visible. The console at the bottom left shows the command `> ?head` entered. The main pane displays the help page for the `tail` function. The 'Arguments' section is circled in red, and a red arrow points from the console input to it.

Environment **History** **Connections**

Files **Plots** **Packages** **Help** **Viewer**

R: Return the First or Last Part of an Object

```
## S3 method for class 'ftable'
tail(x, n = 6L, addrownums = FALSE, ...)
## S3 method for class 'table'
tail(x, n = 6L, addrownums = TRUE, ...)
## S3 method for class 'function'
tail(x, n = 6L, ...)
```

Arguments

| | |
|------------|--|
| x | an object |
| n | a single integer. If positive, size for the resulting object: number of elements for a vector (including lists), rows for a matrix or data frame or lines for a function. If negative, all but the n last/first number of elements of x. |
| addrownums | if there are no row names, create them from the row numbers. |
| ... | arguments to be passed to or from other methods. |

Showing 1 to 8 of 50 entries

Console **Terminal**

```
> ?head
> |
```

Functions

We can implicitly or explicitly pass arguments

```
> head(cars, 8)
```

```
  speed dist
```

```
1      4     2
```

```
2      4    10
```

```
3      7     4
```

```
4      7    22
```

```
5      8    16
```

```
6      9    10
```

```
7     10    18
```

```
8     10    26
```

```
>
```

```
> head(x=cars, n=8)
```

```
  speed dist
```

```
1      4     2
```

```
2      4    10
```

```
3      7     4
```

```
4      7    22
```

```
5      8    16
```

```
6      9    10
```

```
7     10    18
```

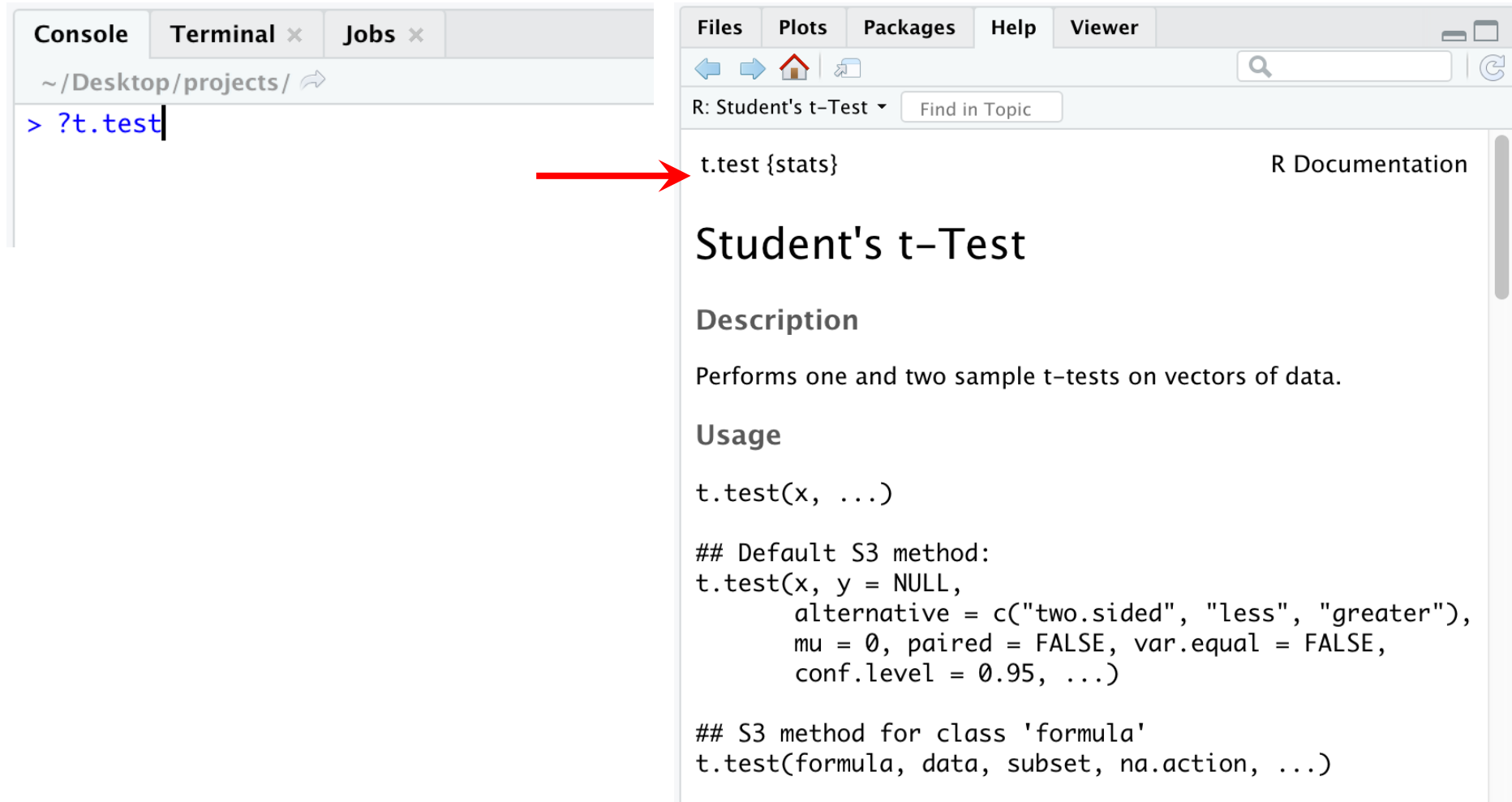
```
8     10    26
```

```
>
```

Troubleshooting

Ask RStudio for help

Type `?function` into console



The screenshot shows the RStudio interface. On the left, the Console pane displays the command `> ?t.test`. A red arrow points from this command to the right pane. The right pane shows the help documentation for the `t.test` function, titled "Student's t-Test". The documentation includes a description, usage, and default arguments.

Console:

```
~/Desktop/projects/
> ?t.test
```

Help Pane: Student's t-Test

Description

Performs one and two sample t-tests on vectors of data.

Usage

```
t.test(x, ...)
```

Default S3 method:

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
```

S3 method for class 'formula'

```
t.test(formula, data, subset, na.action, ...)
```

Ask Google for help

how do i run a t test in r



All

Videos

Images

News

Shopping

More

Settings

Tools

About 2,780,000,000 results (0.64 seconds)

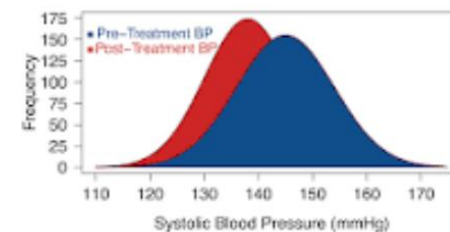
How to Perform **T-tests in R**. To conduct a one-sample **t-test in R**, we use the syntax `t.test(y, mu = 0)` where `x` is the name of our variable of interest and `mu` is set equal to the mean specified by the null hypothesis.


Aug 17, 2015


[datascienceplus.com](#) > t-tests ▾

[How to Perform T-tests in R | DataScience+](#)


Systolic Blood Pressure Before and After Treatment




 Error in `t.test.default(x, y)` : not enough 'x' observations

 Error in `t.test.default(x, y)` : not enough 'x' observations - Google Search

Ask StackOverflow for help

 **stackoverflow**

Products




Log in

Sign up

Home

PUBLIC

 **Stack Overflow**

Tags

Users

Jobs

TEAMS


What's this?

Rotating and spacing axis labels in ggplot2


Ask Question

Asked 10 years, 10 months ago Active 18 days ago Viewed 768k times

680

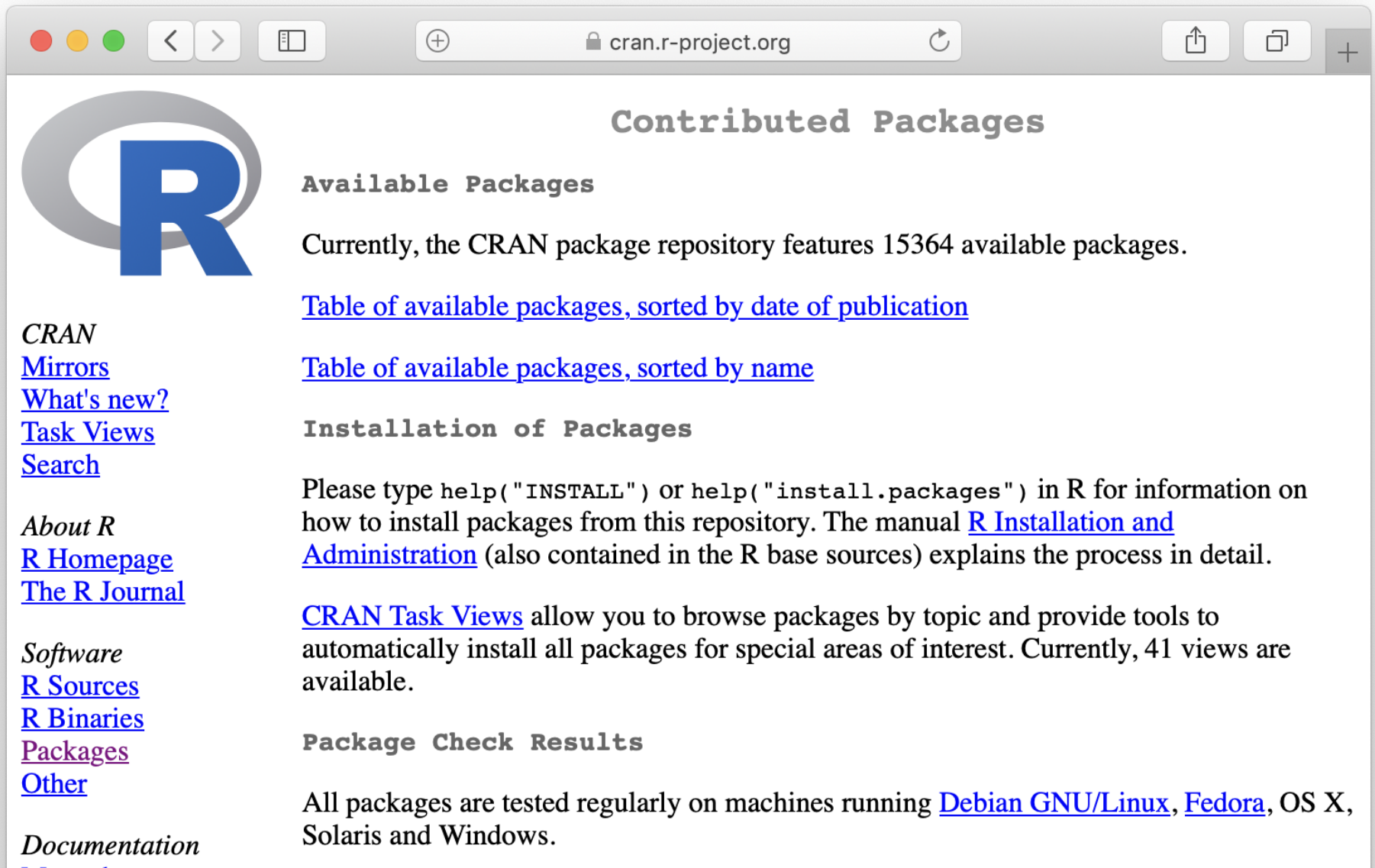


I have a plot where the x-axis is a factor whose labels are long. While probably not an ideal visualization, for now I'd like to simply rotate these labels to be vertical. I've figured this part out with the code below, but as you can see, the labels aren't totally visible.



Extending R through packages:
There's a package for everything

R packages are available on CRAN (Comprehensive R Archive Network)

A screenshot of a web browser displaying the CRAN (Comprehensive R Archive Network) website. The browser's address bar shows 'cran.r-project.org'. The page features the CRAN logo on the left, which consists of a large blue 'R' inside a grey circle. To the right of the logo, the heading 'Contributed Packages' is displayed. Below this, there are sections for 'Available Packages' and 'Installation of Packages'. The 'Available Packages' section states that there are 15364 available packages and provides two links: 'Table of available packages, sorted by date of publication' and 'Table of available packages, sorted by name'. The 'Installation of Packages' section explains how to install packages using the 'help()' function in R and provides a link to the 'R Installation and Administration' manual. It also mentions 'CRAN Task Views' which allow browsing packages by topic. At the bottom, there is a 'Package Check Results' section stating that all packages are tested regularly on various operating systems like Debian GNU/Linux, Fedora, OS X, Solaris, and Windows. On the left side of the page, there is a vertical list of links: 'CRAN', 'Mirrors', 'What's new?', 'Task Views', 'Search', 'About R', 'R Homepage', 'The R Journal', 'Software', 'R Sources', 'R Binaries', 'Packages', 'Other', and 'Documentation'.

Contributed Packages

Available Packages

Currently, the CRAN package repository features 15364 available packages.

[Table of available packages, sorted by date of publication](#)

[Table of available packages, sorted by name](#)

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual [R Installation and Administration](#) (also contained in the R base sources) explains the process in detail.

[CRAN Task Views](#) allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 41 views are available.

Package Check Results

All packages are tested regularly on machines running [Debian GNU/Linux](#), [Fedora](#), OS X, Solaris and Windows.

[CRAN](#)

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

[About R](#)

[R Homepage](#)

[The R Journal](#)

[Software](#)

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

[Documentation](#)

Bio-specific R packages are available on Bioconductor



Search:

[Home](#)

[Install](#)

[Help](#)

[Developers](#)

[About](#)

About Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data.

Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, and an active user community. Bioconductor is also available as an [AMI](#) (Amazon Machine Image) and [Docker](#) images.

News

- See our [google calendar](#) for events, conferences, meetings, forums, etc. Add your event with email to events at [bioconductor.org](#).
- Bioconductor **3.11** is available.
- Nominate an outstanding community member for a Bioconductor Award! See the [support site](#) for more information.
- Registration open for [BioC2020](#).
- Core team **job opportunities** available, contact Martin.Morgan at [RoswellPark.org](#)
- Bioconductor [F1000 Research Channel](#) is

BioC 2020

Get the latest updates on the [BioC 2020 Conference!](#)

- BioC 2020 is going virtual July 27 - July 31. Please see the [Registration Page](#) for more information.
- Nominate an outstanding Bioconductor community member for a Bioconductor Award! See [posting](#) for more information.
- Call for birds-of-feather, hack-a-thon, and how-to sections. Please see [posting](#) for more information.
- Registration is now open. [Register today.](#)

Install »

- Discover [1903 software packages](#) available in Bioconductor release 3.11.

Get started with Bioconductor

- [Install Bioconductor](#)
- [Get support](#)
- [Latest newsletter](#)
- [Follow us on twitter](#)
- [Install R](#)

Learn »

Master Bioconductor tools

- [Courses](#)
- [Support site](#)
- [Package vignettes](#)
- [Literature citations](#)
- [Common work flows](#)
- [FAQ](#)
- [Community resources](#)
- [Videos](#)