Introduction to R Language

HI 743

Department of Health Informatics and Administration
Zilber College of Public Health
University of Wisconsin - Milwaukee

February 6, 2025

R & RStudio

R was created during the 1990's as a derivative of the S programming language. Recently it has risen to popularity for reasons including:

- R is open source and freely available.
- R is available for Windows, Mac, and Linux.
- R has an extensive set of tools for statistical analysis.
- R has an extensive, flexible graphical facility capable of producing publication quality figures.
- R has an expanding set of freely available 'packages' to extend R's capabilities.
- R has an extensive support network with numerous online resources.

Comprehensive R Archive Network (CRAN)

The “Comprehensive R Archive Network” (CRAN) is a collection of sites which carry identical material, consisting of the R distribution(s), the contributed extensions, documentation for R, and binaries.

https://CRAN.R-project.org/

From CRAN, you can obtain the latest official release of R, daily snapshots of R (copies of the current source trees), as gzipped and bzipped tar files, a wealth of additional contributed code, as well as prebuilt binaries for various operating systems (Linux, Mac OS Classic, macOS, and MS Windows). CRAN also provides access to documentation on R, existing mailing lists and the R Bug Tracking system.

Why use R? Open Source.

R is released under the GNU General Public License (GPL), version 2 or version 3. The GPL is a free software license that guarantees users the freedom to run, study, modify, and distribute software. It explicitly encourages using this software in all contexts with freedom from proprietary restrictions.

It is the opinion of the R Core Team that one can use R for commercial purposes (e.g., in business or in consulting). The GPL, like all Open Source licenses, permits all and any use of the package. Most add-on packages, including all recommended ones, also explicitly allow commercial use in this way.

Why use R? Packages.

R packages are extensions to the R statistical programming language. R packages contain code, data, and documentation in a standardized collection format that can be installed by users of R, typically via a centralized software repository. The two most popular software repositories are:

- **CRAN**: is R's central software repository. As of November 2020, more than 16,000 packages are available. The master site is located at the Vienna University of Economics and Business and is mirrored on servers around the world.
- **Bioconductor**: project provides R packages for the analysis of genomic data. This includes object-oriented data-handling and analysis tools for data from Affymetrix, cDNA microarray, and next-generation high-throughput sequencing methods.

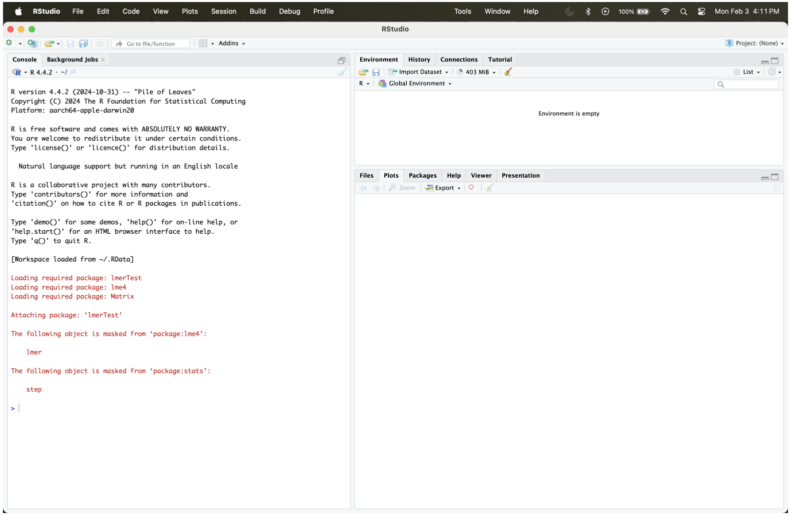
RStudio Orientation

RStudio is an **Integrated Development Environment** (IDE). RStudio can be thought of as an add-on to R which provides a more user-friendly interface, incorporating the R Console, a script editor and other useful functionality (like R markdown and GitHub integration).

RStudio includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

RStudio Orientation

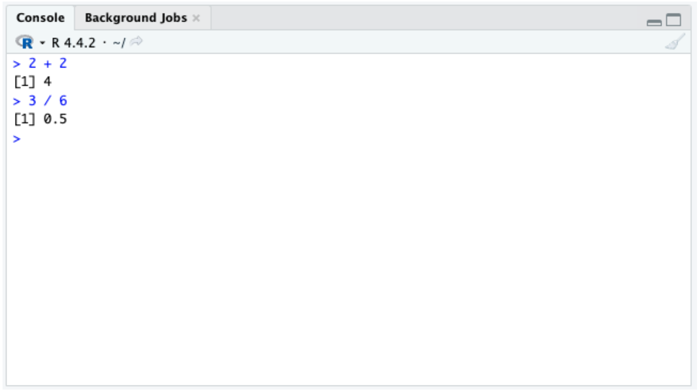
When you open RStudio for the first time, you will see the following layout.



```
R version 4.4.2 (2024-10-31) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20

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.

[Workspace loaded from ~/.RData]

Loading required package: lmerTest
Loading required package: lme4
Loading required package: Matrix

Attaching package: ‘lmerTest’

The following object is masked from ‘package:lme4’:

    lmer

The following object is masked from ‘package:stats’:

    step

> 
```

RStudio Orientation: Console

The Console is the workhorse of R. This is where R evaluates all the code you write. You can type R code directly into the Console at the command line prompt, $>$. For example, if you type $2 + 2$ into the Console you should obtain the answer 4.

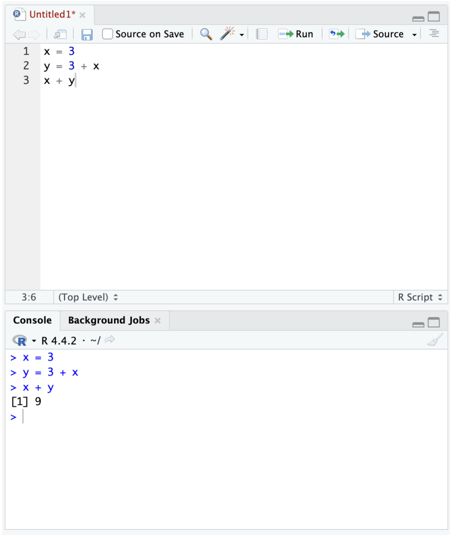


```
> 2 + 2
[1] 4
> 3 / 6
[1] 0.5
>
```

RStudio Orientation: Console

Instead of typing R code directly into the Console a better approach is to create an R script. An R script is just a plain text file with a .R file extension which contains your lines of R code. These lines of code are then sourced into the R Console line by line.

The [→Run] button in the top-right will run the entire script in the source pane. Otherwise, 'Command+Enter' (Ctrl+Enter on Windows) will run the current line, or the highlighted block of text.

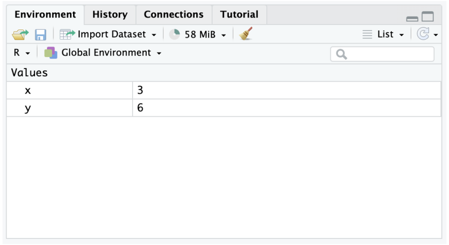


```
x = 3
y = 3 + x
x + y
```

```
> x = 3
> y = 3 + x
> x + y
[1] 9
>
```

RStudio Orientation: Top-Right Panel

- The **Environment** tab displays all the objects you have created in the current (global) environment.
- The **History** tab contains a list of all the commands you have entered into the R Console.
- The **Connections** tab allows you to connect to various data sources such as external databases.

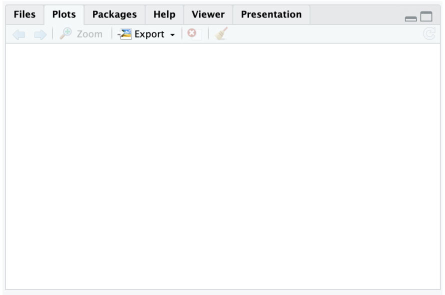


Values

x	3
y	6

RStudio Orientation: Bottom-Right Panel

- The **Files** tab lists all external files and directories in the current working directory on your computer.
- The **Plots** tab is where all the plots you create in R are displayed (unless you tell R otherwise).
- The **Packages** tab lists all of the packages that you have installed on your computer.
- The **Help** tab displays the R help documentation for any function.
- The **Viewer** tab displays local web content such as web graphics generated by some packages.



Packages: CRAN

To install a package from CRAN you can use:

Install Packages Function

```
install.packages()
```

For example if you want to install the remotes package enter the following code into the Console window of RStudio

```
install.packages("ISLR", dependencies = TRUE)
```

You can also use the following function to update existing packages.

Update Packages Function

```
update.packages()
```

Packages: Bioconductor

To install packages from Bioconductor, you first need to install the `BiocManager` package. Once the BiocManager package has been installed you can either install all of the 'core' Bioconductor packages with:

Bioconductor Packages

```
BiocManager::install()
```

```
BiocManager::install(c("GenomicRanges", "edgeR"))
```

Note: The '::' notation specifies which package specifically to call a function from.

Packages: GitHub

The most efficient method for installed directly from GitHub is to use the `install_github()` function from the `remotes` package.

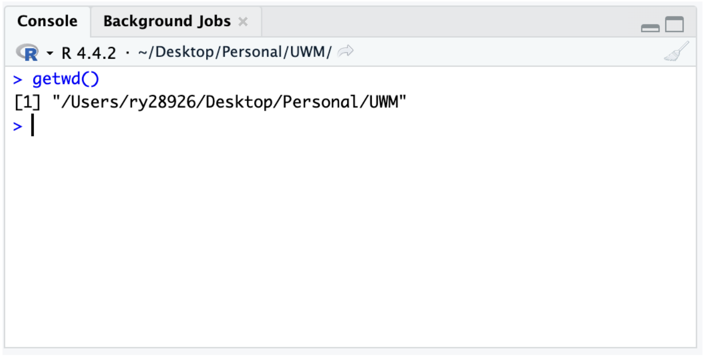
GitHub Packages

```
remotes::install_github('tidyverse/dplyr')
```

Typically, one would install from GitHub if they're looking to use a development version of an R Package. It's important to note that installing from GitHub could add security risks if the repository is not properly vetted.

Working Directories

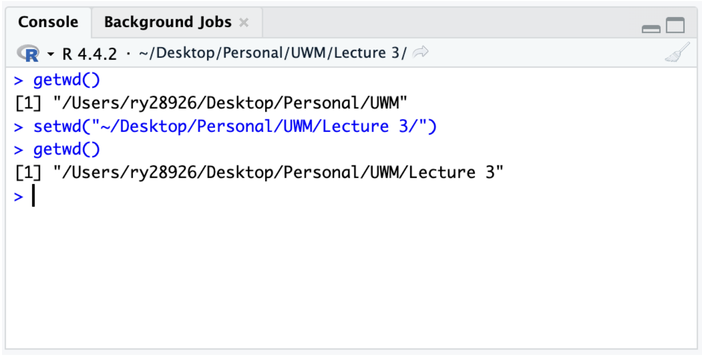
The working directory is the default location where R will look for files you want to load and where it will put any files you save. You can check the file path of your working directory by looking at bar at the top of the Console pane. You can also use the `getwd()` function in the Console which returns the file path of the current working directory.



```
> getwd()
[1] "/Users/ry28926/Desktop/Personal/UWM"
> 
```

Working Directories

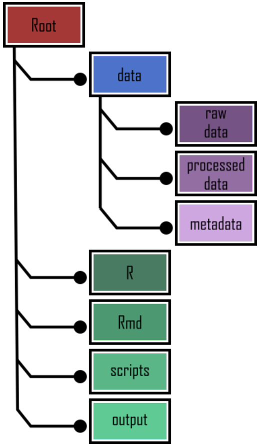
The `setwd()` function will allow you to change your working directory. This can help reduce the amount of typing needed to locate files that exist together. It will also be where files save-to by default.



```
> getwd()
[1] "/Users/ry28926/Desktop/Personal/UWM"
> setwd("~/Desktop/Personal/UWM/Lecture 3/")
> getwd()
[1] "/Users/ry28926/Desktop/Personal/UWM/Lecture 3"
> 
```

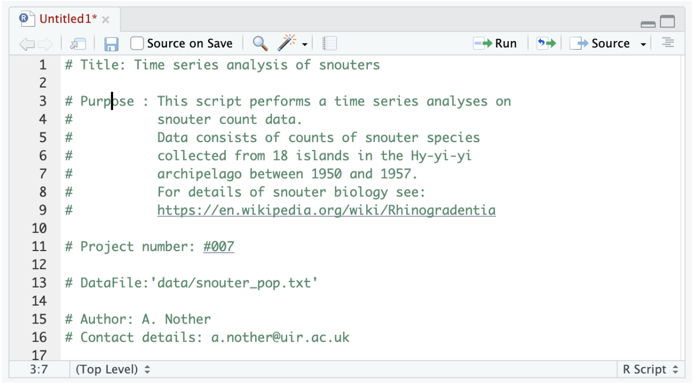
Recommended Project Directory Structure

It's good practice to structure your working directory in a consistent and logical way to help both you and your collaborators. Shown is a recommended directory structure.



Project Documentation

It's important include in your R script all the information required to make your work reproducible (author names, dates, sampling design etc). Comments can be made with the # symbol.



```
# Title: Time series analysis of snouters

# Purpose : This script performs a time series analyses on
#           snouter count data.
#           Data consists of counts of snouter species
#           collected from 18 islands in the Hy-yi-yi
#           archipelago between 1950 and 1957.
#           For details of snouter biology see:
#           https://en.wikipedia.org/wiki/Rhinogradentia

# Project number: #007

# DataFile:'data/snouter_pop.txt'

# Author: A. Nother
# Contact details: a.nother@uir.ac.uk
```

Objects in R

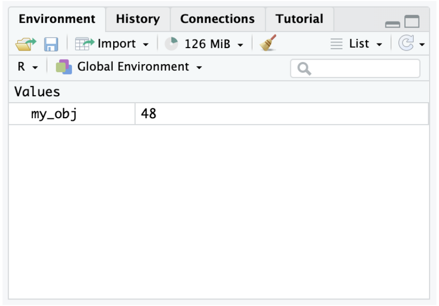
A main concept in R is that everything is an **object**.

These objects can be almost anything, from a single number or character string (like a word) to highly complex structures like the output of a plot, a summary of your statistical analysis or a set of R commands that perform a specific task.

```
> my_obj = 48
> my_obj
## [1] 48
```

Objects in R

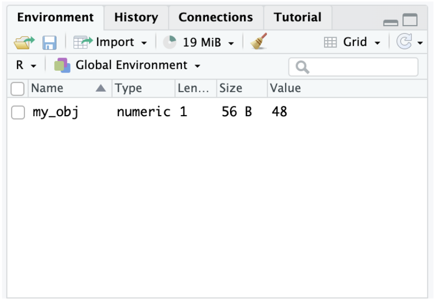
Now that we've created this object, R knows all about it and will keep track of it during this current R session. All of the objects you create will be stored in the current workspace and you can view all the objects in your workspace in RStudio by clicking on the 'Environment' tab in the top right hand pane.



```
Values
  my_obj     48
```

Objects in R

If you click on the down arrow on the 'List' icon in the same pane and change to 'Grid' view RStudio will show you a summary of the objects including the type (numeric - it's a number), the length (only one value in this object), its 'physical' size and its value (48 in this case).

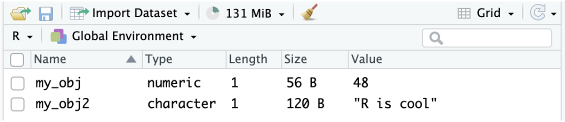


Name	Type	Len...	Size	Value
my_obj	numeric	1	56 B	48

Objects in R

There are many *types* of objects that we can assign values to. For instance, we can create a *character* type variable.

```
> my_obj2 = "R is cool"
> my_obj2
## [1] "R is cool"
```



```
Name      Type       Length  Size   Value
my_obj    numeric    1       56 B   48
my_obj2   character  1       120 B  "R is cool"
```

Notice that we have enclosed the string in quotes. If you forget to use the quotes you will receive an error message.

Bibliography

1. https://cran.r-project.org/
2. https://intro2r.com/