Learning R
tldR

Basics & Foundations

Class Updates

- Office hours starting this week
  - James: Tu 11:30AM - 12:30PM
  - Karl: Th 11:30AM - 12:30PM
- Lab Discussion Period @ 4:00PM - 5:00PM
  - Holdsworth 302 (same as class room)
- Join the class Discord for tech support & questions
- Class Topics Chosen!
  - Species Distribution Modeling
  - Advanced Statistics in R

Part 1: Background

Quick History of GIS

- GIS was historically command-line driven software supported by government agencies, and later by tech companies and industry.
- 1969: The Minnesota Land Management Information System (MLMIS) was one of the earliest government mainframe GIS systems from 1969 which operated through a command-line interface.
- 1986: ESRI's 1986 release of ARC/INFO GIS continued the command-line driven GIS legacy and used its own proprietary language AML.
- 1999: ESRI releases ArcMap.
- 2004: ESRI transitions from ArcMap to ArcGIS in version 9.0 update.

Why would you want to use R for GIS?

The tool you use shapes what you make with it!

- GUI-GIS trades functionality for approachability
  - General GIS operations (ex: clip, buffer, intersect etc)
  - Visual spatial data creation
  - Machine Learning
- CL-GIS trades approachability for functionality
  - Complicated, custom analytics
  - Repetitive tasks
  - Niche analytics

How did R get it’s GIS functionality?

- 1972: C coding language is developed by Dennis Ritchie at Bell Laboratories.
- Foundational GIS software is then developed using C/C++
  - GDAL/OGR enables reading and writing raster/vector data
  - GEOS (Geometry Engine - Open Source) enables spatial operations like intersection, union, difference etc.
  - PROJ enables cartographic projections and coordinate transformations.
  - PostGIS enables advanced spatial operations and querying
- 2000: R is released as a statistically oriented implementation of C
- >2000: Foundational GIS software is integrated with R (slowly)

Part 2: R Fundamentals

R Concepts - Arithmetic

- + = Addition
  - > 1 + 1
  - > 2
- - = Subtraction
- * = Multiplication
- / = Division
- ^ = Exponent
- %% = Modulus (remainder from division)
- %/% = Integer Division

R Concepts - Logicals

- > Great than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to
- == Equal to
- != Not equal to
- Note: These typically return TRUE/FALSE values if run alone, but can also be used to filter your data along with other functionality.

R Concepts - Running Code

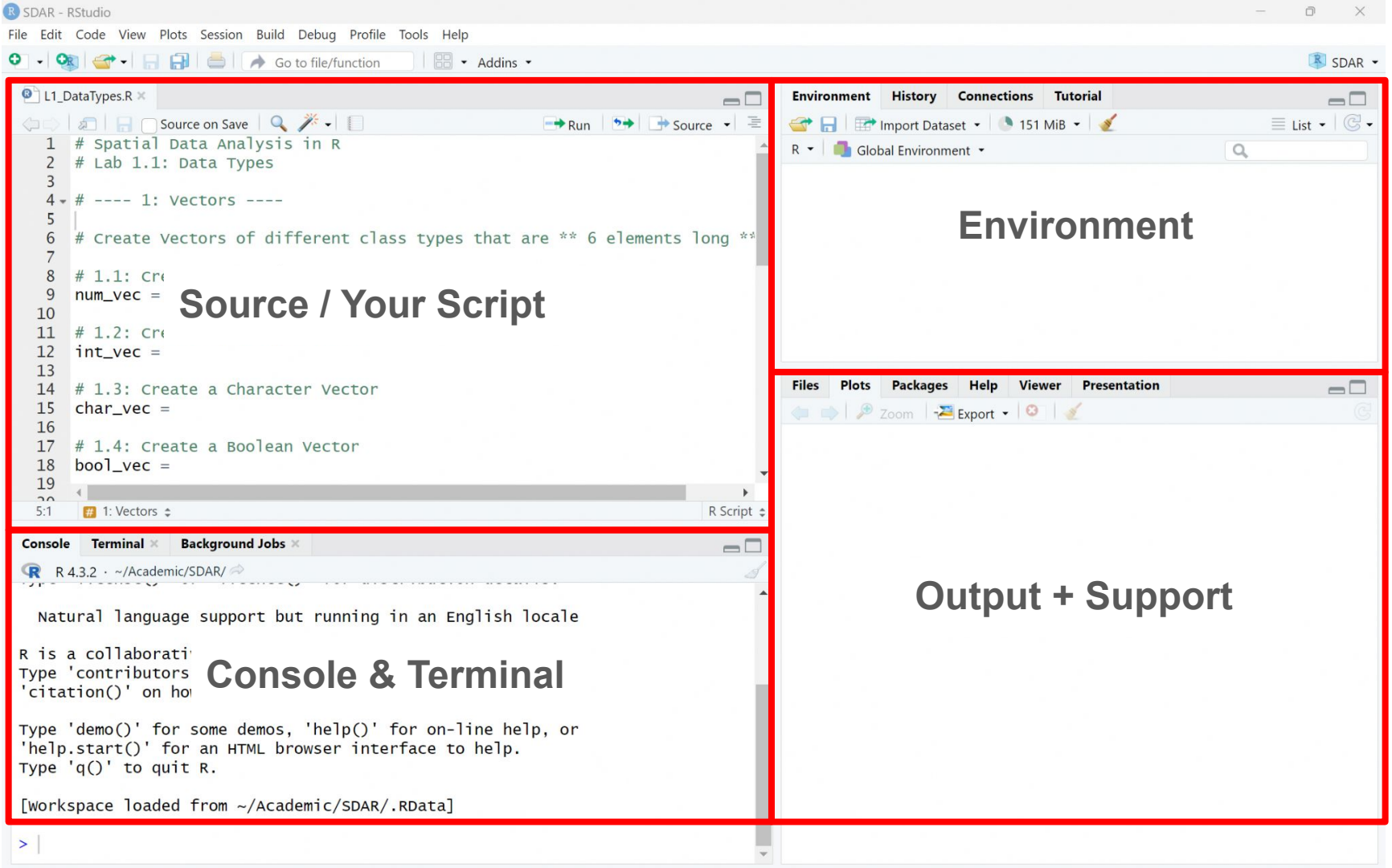
- 

will run your entire script from top to bottom
- **Ctrl+Enter** will run the current line of code that your cursor is on
  - Cmd on Mac
  - Ctrl on Windows/Linux
- Highlight sections of code + Ctrl+Enter to run that selection

R Concepts - Running Code

- You have a few options of places to run code in RStudio

1. **Source**: Your saved R Script
2. **Console**: Direct, unsaved interface to R itself
3. **Terminal**: Access to your system shell within the RStudio interface.
    - Depending on your operating system (Windows, macOS, or Linux), this could be Command Prompt, PowerShell, or Bash etc.


SDAR - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
SDAR
L1_DataTypes.R
Source on Save
Run
Source
1 # Spatial Data Analysis in R
2 # Lab 1.1: Data Types
3
4 # ---- 1: Vectors ----
5
6 # Create Vectors of different class types that are ** 6 elements long **
7
8 # 1.1: Cr
9 num_vec =
10
11 # 1.2: Cr
12 int_vec =
13
14 # 1.3: Create a Character Vector
15 char_vec =
16
17 # 1.4: Create a Boolean Vector
18 bool_vec =
19
5:1
1: Vectors
R Script
Source / Your Script
Environment
History
Connections
Tutorial
Import Dataset
151 MiB
List
R
Global Environment
Environment
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Output + Support
Console
Terminal
Background Jobs
R 4.3.2 · ~/Academic/SDAR/
Natural language support but running in an English locale
R is a collaborati
Type 'contributors
'citation()' on ho
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 ~/Academic/SDAR/.RData]
Console & Terminal
>

R Concepts - Variables

- A variable is a symbol or name that stands for a value.
- Variables are used to store data values
- **=** or **<-** both mean ‘equals’ when you are assigning a new variable
  - x = 10
  - y <- 20
- **->** can be used to assign a value to a variable to the right
  - 30 -> z
- Example: Calling a variable
  - > x
  - > 10

R Concept - Environment & Variables

- In R, variables in the environment are overwritten if you run the code again with the same name!
- Example:
  - 1 > x = 10
  - 2 > x = 25
  - 3 > x
  - Output: 25

R Concepts - Quality of Life Syntax

- **#** Comment on your code with a hashtag
- **# - - - -** Create Mapped Sections w/ This **- - - -**
- You can use a semicolon **;** to run multiple commands on the same line
  - x = 15 ; 25 -> y
- You can clear your environment with this code:
  - **rm(list=ls())**

R Concepts - Functions

- **Function** Definition: A set of instructions or code that performs a specific task, which usually takes some inputs (called **arguments** or **parameters**), processes them, and returns a result.
  - Example: mean() returns the mean of the data provided to it.
    - > mean(data)
    - > 55.13 # The mean of data is 55.13

function(parameters)

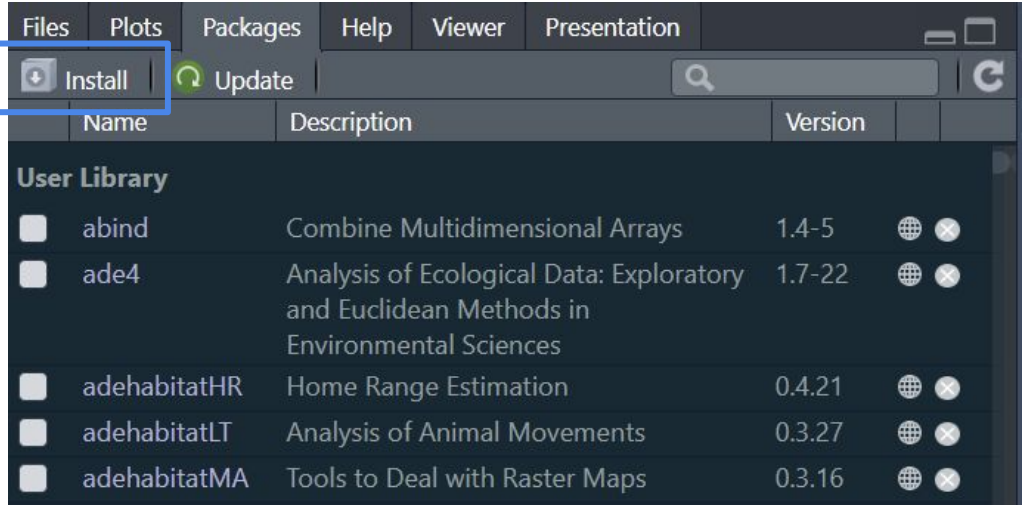
R Concepts - Base R Functions

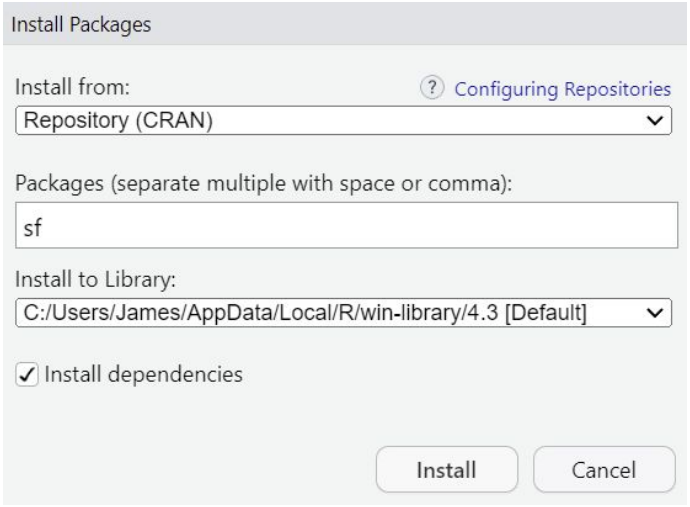
- mean(x) : Mean of x.
- median(x) : Median of x.
- sd(x) : Standard deviation of x.
- var(x) : Variance of x.
- quantile(x, probs) : Quantiles of x at specified probabilities probs.
- abs(x) : Absolute value of x.
- sqrt(x) : Square root of x.
- round(x, digits) : Round values in x to specified digits.

R Concepts - Packages

- **Package** Definition: A collection of functions, data, or compiled code. Packages extend the capability of R by providing additional functionality.
  - Example: sf package gives GIS functionality to R!
    - > **install.packages**(sf) # Install sf package
    - > **require**(sf) # Load package into R
    - > x = st_read(“shapefile.shp”)

R Concepts - Installing Packages





Packages - Expanding R Functionality

Loading Data

- **SF**
- **Terra**
- lidR

Plotting

- **ggplot2**
- rasterVis
- leaflet / mapboxapi

Analytics

- **spatstat**
- **gstat**
- landscapemetrics
- adehabitat
- landsat
- **dismo**
- geoR
- spatialreg

Data Manipulation

- **dplyr**
- **tidyr**
- purrr

Application Interfacing

- RQGIS
- rgrass
- rsatscan

R Workflow

1. **Load Packages (Optional)**
2. Load Data
3. Prepare Data for Analysis
4. Conduct Analysis
5. Visualize Output

R Workflow

1. Load Packages (Optional)
2. **Load Data**
3. Prepare Data for Analysis
4. Conduct Analysis
5. Visualize Output

R Concepts - Data Class Types

- **Vector** - 1D same class type storage
- **Matrix** - 2D same class type storage
- **Data Frame** - 2D multi-class-type storage ***
  - Most commonly used, will be what your .csv file loads in as
- **List** - Unrestricted, flexible storage

R Concepts - Data Types

- Most common data types:
  - **Characters** = Letters
  - **Decimal** Values = Numerics
  - **Natural** Numbers = Integers
    - Integers are also Numerics
  - **Logical** = Boolean Values (TRUE/FALSE)

Vectors

- Definition: A vector is a **1D** data structure consisting of one or more elements.
  - Created with **c() function**!
- In a vector, all of the elements must be of the same class type.

```
> vector = c("Vectors", "are all the same", "class type!")
> print(vector)
[1] "Vectors"          "are all the same" "class type!"
```

Matrices

- A Matrix is a **2D** collection of elements of the **SAME class type** arranged into a fixed number of rows and columns.
  - Created with **matrix() function**!

```
int_matrix = matrix(c(1:5, 11:15, 21:25, 31:35, 41:45, 51:55),
                    byrow = TRUE, # filled by rows
                    nrow = 6, # 6 rows
                    ncol = 5) # 5 columns
int_matrix
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]   11   12   13   14   15
## [3,]   21   22   23   24   25
## [4,]   31   32   33   34   35
## [5,]   41   42   43   44   45
## [6,]   51   52   53   54   55
```

Matrix Math!

```
int_matrix + 5
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    6    7    8    9   10
## [2,]   16   17   18   19   20
## [3,]   26   27   28   29   30
## [4,]   36   37   38   39   40
## [5,]   46   47   48   49   50
## [6,]   56   57   58   59   60
```

```
int_matrix * 5
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    5   10   15   20   25
## [2,]   55   60   65   70   75
## [3,]  105  110  115  120  125
## [4,]  155  160  165  170  175
## [5,]  205  210  215  220  225
## [6,]  255  260  265  270  275
```

```
int_matrix - 5
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   -4   -3   -2   -1    0
## [2,]    6    7    8    9   10
## [3,]   16   17   18   19   20
## [4,]   26   27   28   29   30
## [5,]   36   37   38   39   40
## [6,]   46   47   48   49   50
```

```
int_matrix / 5
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.2  0.4  0.6  0.8    1
## [2,]  2.2  2.4  2.6  2.8    3
## [3,]  4.2  4.4  4.6  4.8    5
## [4,]  6.2  6.4  6.6  6.8    7
## [5,]  8.2  8.4  8.6  8.8    9
## [6,] 10.2 10.4 10.6 10.8   11
```

Data Frames

- Data Frames are a versatile **2D** form of data storage that have the variables of a data set as columns, and the observations as rows.
  - Created with **data.frame() function**!
- Data Frames allows you to compile vectors of **DIFFERENT data types** into a singular data frame without any data type conflict!
- Data frames are a type of list, but they have a few restrictions:
  - All elements of a data frame have an equal length
  - You can’t use the same name for two different columns

Data Frames Example

- We can combine a numeric vector, integer vector, character vector, and a boolean vector into a dataframe table!

```
dat_vec = data.frame(num_vec, int_vec, char_vec, bool_vec)
dat_vec
```

```
##   num_vec int_vec char_vec bool_vec
## 1     1.5       1      how     TRUE
## 2     2.4       2      are     TRUE
## 3     3.3       3      you     TRUE
## 4     4.2       4    doing    FALSE
## 5     5.1       5 Michael?    FALSE
```

Lists

- Lists are a collection of elements without any restriction on the class, length or structure of each element.
  - Created with **list() function**!
- Lists are like a hub of different elements, including other lists if you so desire! But because of this, you can’t access lists in the same way that you would with a 2D matrix or data frame. (more on this later)

```
list_vec = list(num_vec, int_vec, char_vec, bool_vec)
```

Converting Between Data Types

- as.vector()
- as.matrix()
- as.data.frame()
- as.list()

Part 3: Project & Data Management

Directories

- **Root Directory**: The top-most directory in the filesystem from which all other directories branch off. Aka the starting point of the filesystem hierarchy.
  - Example: C:/…
  - Navigation Example: C:/Users/UserName/Documents/Project/Data/data.csv
- **Current Working Directory** (CWD): The folder in which a user or program is working at a given time. Essentially, it's the default directory that the system uses for all relative file and directory operations.
  - The CWD can be modified
  - RStudio typically sets the CWD to the directory where the last loaded script was located
    - Unless you’re using an R Project!

Absolute vs. Relative Paths

- **Absolute Paths**: Specifies a file or folders location in relation to the root directory of the filesystem. It contains the complete directory list required to locate the file.
  - Example: C:/Users/UserName/Documents/Project/Data/data.csv
  - Constant, they do not change regardless of the working directory.
  - Bad portability: Often requires updating the absolute paths as they are unique to your computer.
- **Relative Paths**: Specifies a file or folders location in relation to the current working directory. It is more flexible and portable than an absolute path.
  - Changes based on the current directory, making them more adaptable.
  - Great portability: Paths don't need to be changed as long as the directory structure remains intact.

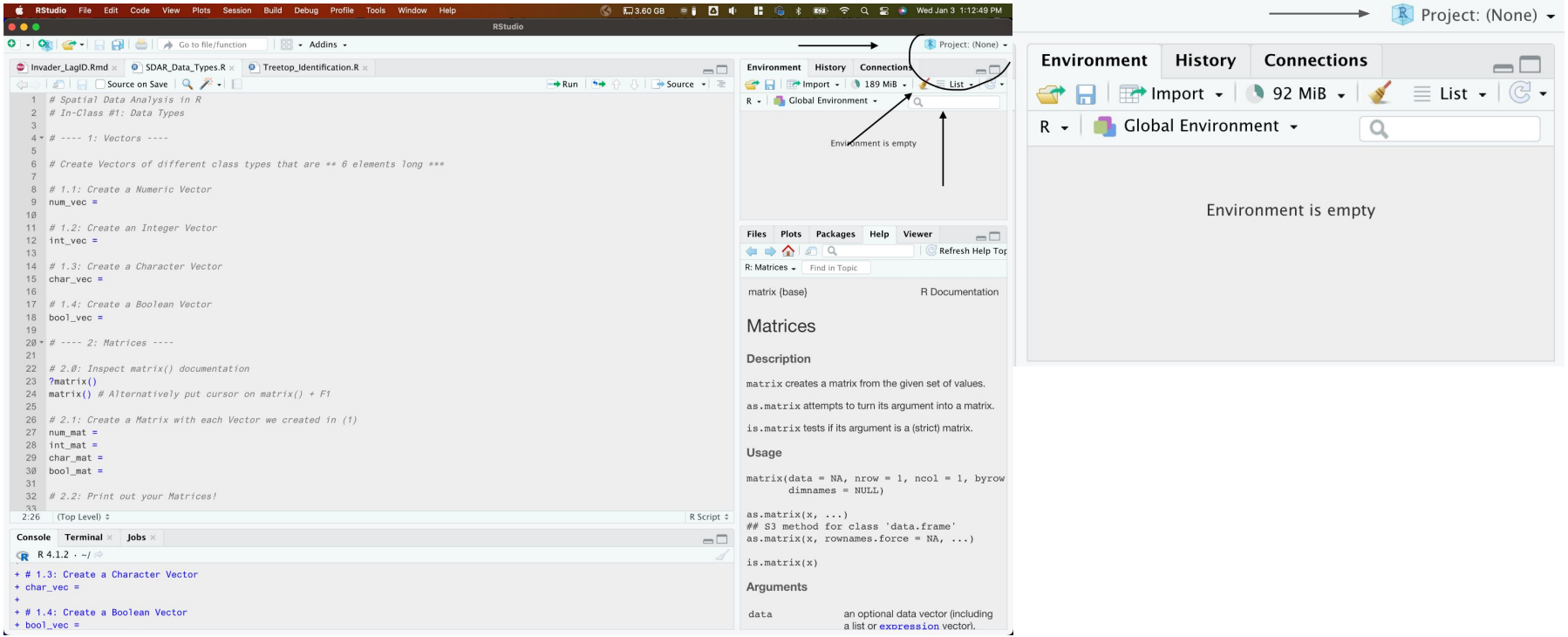
R Applications - Loading in Data

- read.csv(file_location)
- Example
  - > hps_taxa = read.csv('C:\Users\James\Documents\Invader_Dispersal\data\hps_taxa.csv')
  - > head(hps_taxa)

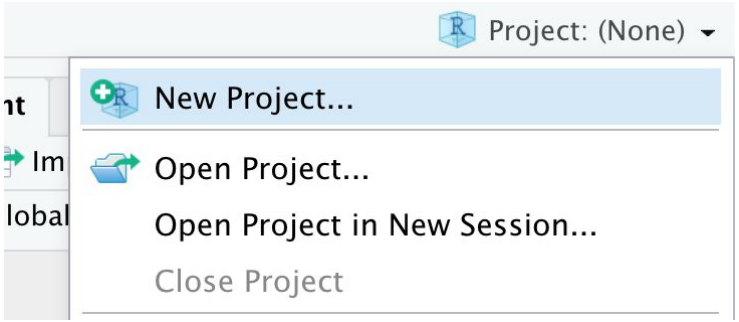
USDAcode	AcceptedName	USDAstatus	invasive_status	Regulated	Habit	n_records
IRSI	Iris sibirica	I	NA	No	Forb/herb	1325
AGHO3	Ageratum houstonianum	I	invasive	No	Forb/herb	1807
ARMA7	Aristolochia macrophylla	N	NA	No	Vine	5052
ARPY8	Pleioblastus fortunei	unknown	NA	No	Graminoid	56
MELI4	Mentzelia lindleyi	N	NA	No	Forb/herb	2450

R Concepts - RStudio Projects

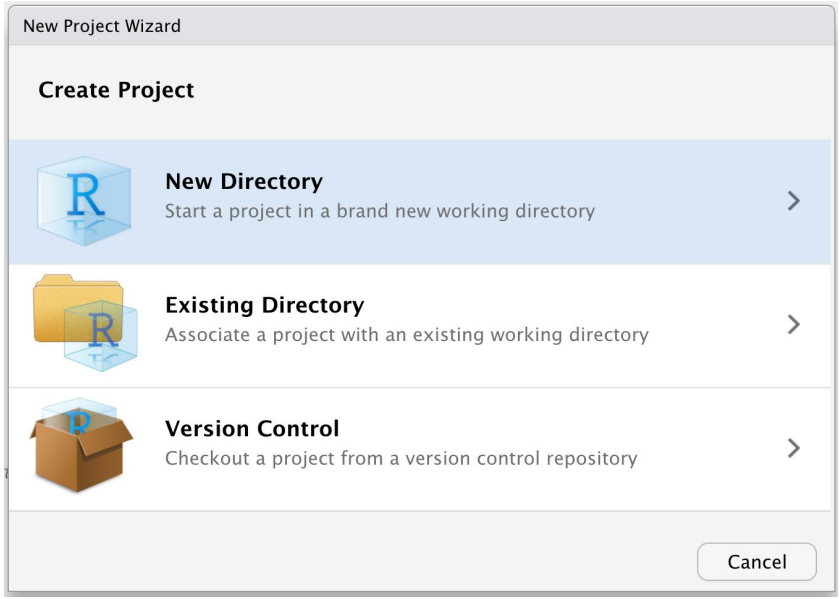
- Projects organize your workflow and help save environment data (so you don't have to rerun your code every time you boot up R-Studio!).



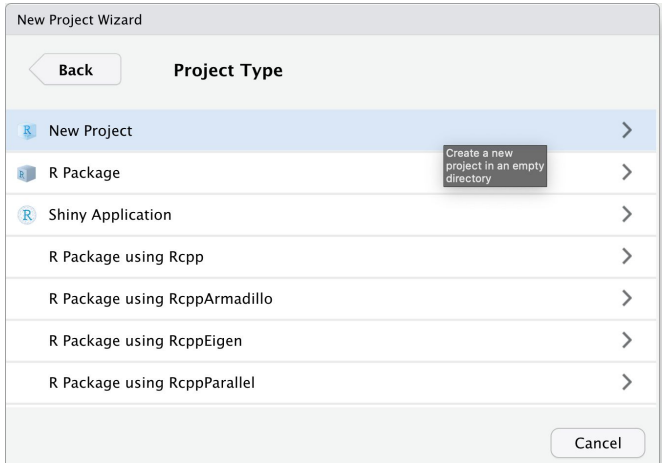
1:


Project: (None)
New Project...
Open Project...
Open Project in New Session...
Close Project

2:

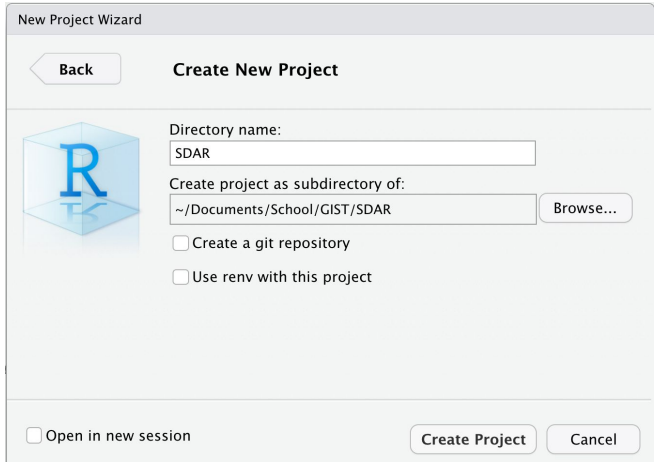

New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

3:



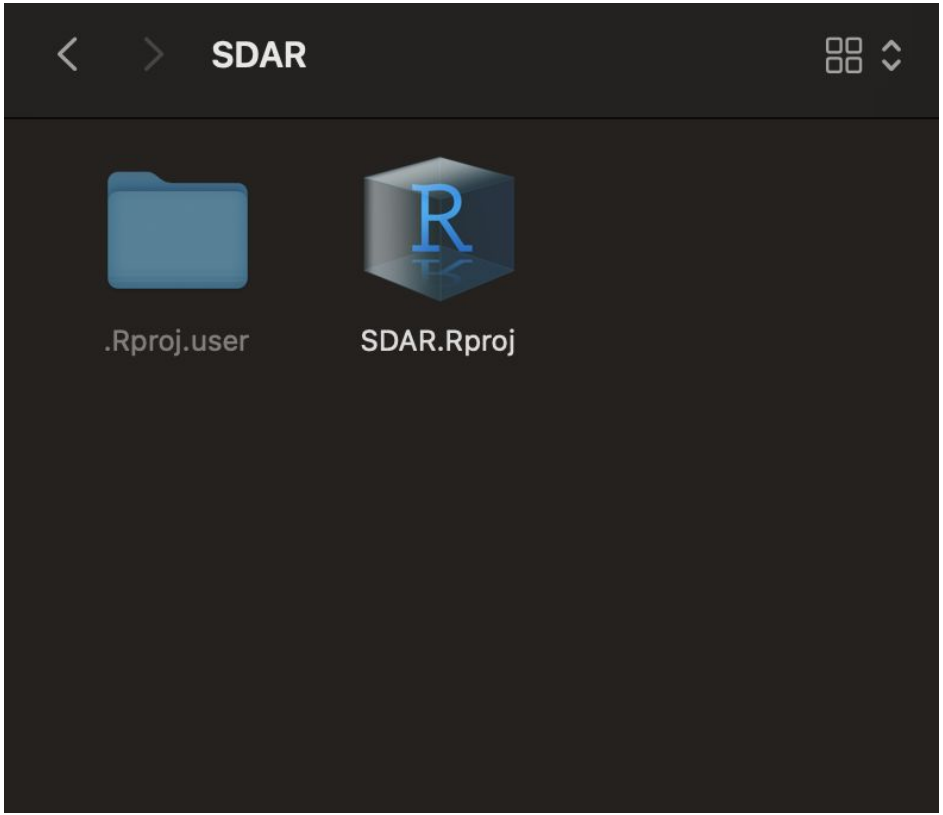
- Create a new Project

4:



- Name directory to be created.
- Choose location of where the directory will be created (Browse…)

You’ve created a project!



R Applications - Better Loading in Data

- here Package: Allows for easier project directory based file location.
- Example
  - > install.packages(here)
  - > require(here)
  - > hps_taxa = read.csv(**here**(‘data’, ‘hps_taxa.csv’))
- Package::Function() Method
  - > hps_taxa = read.csv(**here::here**(‘data’, ‘hps_taxa.csv’))
- Compare to earlier method
  - > hps_taxa = read.csv(‘C:\Users\James\Documents\Invader_Dispersal\data\hps_taxa.csv’)

R Applications - Alternative Loading in Data

- file.choose() & choose.dir()
  - Good for: Making code accessible.
  - Bad for: General code usage, you have to select the file every run!

R Workflow

1. Load Packages (Optional)
2. Load Data
3. **Prepare Data for Analysis**
4. Conduct Analysis
5. Visualize Output

What is data preparation?

- This is highly analysis dependent, but is generally some combination of:
  - **Subsetting Data**: Some functions only want specific columns in their arguments, so you have to make sure that is all you're giving it.
  - **Data Cleaning**: Many functions don't mix well with NA values, so removing them is important. Also removing any data that may skew your analysis is helpful.
  - **Reformatting Data**: A function may only accept a matrix as an input, so the dataframe you got from loading in your .csv file will need to be re-formatted using as.matrix().
    - Example
      - > df = read.csv(here::here('data', 'table.csv'))
      - > df_matrix = as.matrix(df)

R Syntax - Subsetting

- data[row , column]
  - **data[2,]** : subsets only the second row from data
  - **data[1:20,]** : subsets rows 1 through 20 from data
  - **data[, 3] or data[3]** : subsets column 3 from data
    - data[,3] is a matrix/dataframe
    - data[3] is a vector
  - **data[, “column1”]** : subsets just column1
  - **data[, c(“column1”, “column2”, “column3”)]** : subsets column1-3
- $
  - data$column1 : subsets just column1
    - Same as data[, “column1”]

List Subsetting is Different!

```
# Lets create a list
list_vec = list(num_vec, int_vec, char_vec, bool_vec)

# A single bracket [] will not return the element in its original data type, but as a list
list_vec[2]
```

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

```
# This is shown if you use the class() function we used earlier!
  # This is good to note if you are trying to input a list element into a data structure that requires
homogeneity! ***
class(list_vec[2])
```

```
## [1] "list"
```

```
# To select an entire element within a list, and retain it's class type - use [[]]
list_vec[[2]] # if you open the list, it is the second element within the list = [[2]]
```

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

R Workflow

1. Load Packages (Optional)
2. Load Data
3. Prepare Data for Analysis
4. **Conduct Analysis - Later in Course!**
5. Visualize Output

R Workflow

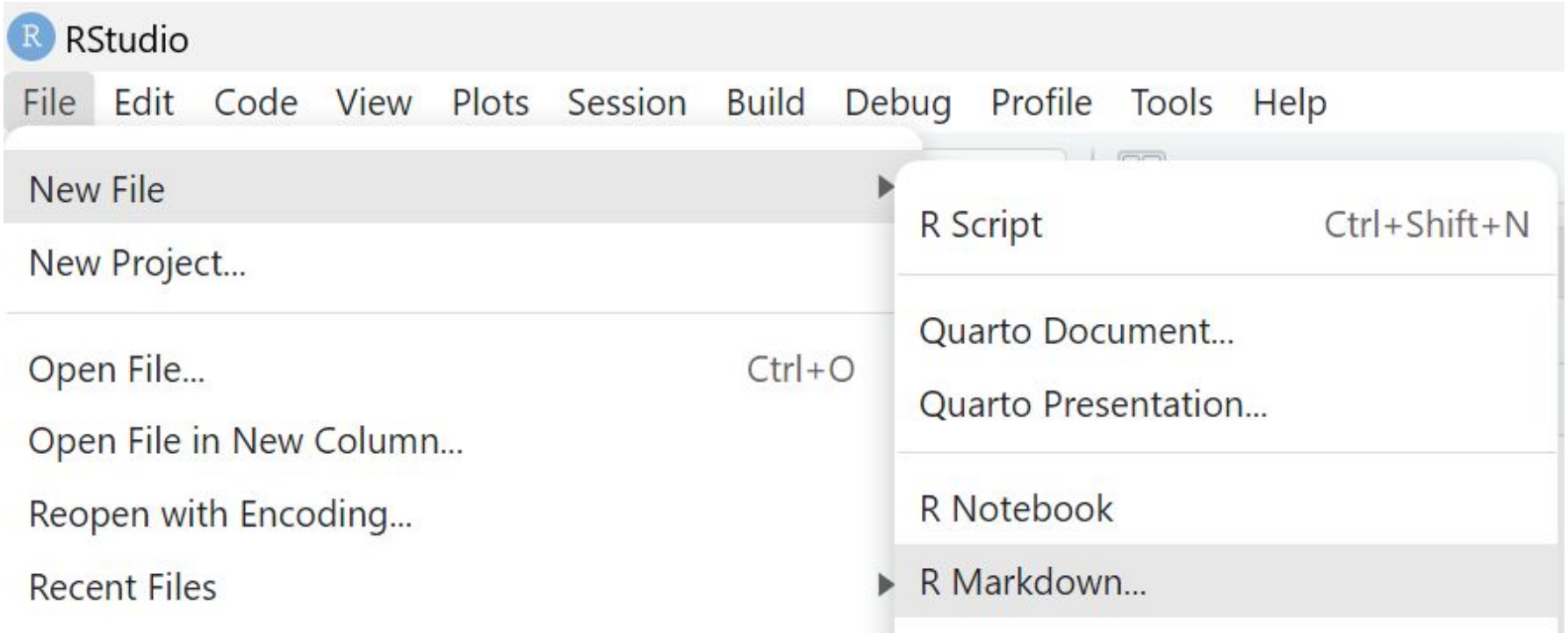
1. Load Packages (Optional)
2. Load Data
3. Prepare Data for Analysis
4. Conduct Analysis - Later in Course!
5. **Visualize Output**

R Markdown

R-Markdown

- A syntax for writing code chunk-wise
- Allows for stitching into PDFs and HTML documents
- Especially useful for data analysis reporting
- Pros:
  - Tidy way of organizing your code
  - Makes documenting and presenting your code easier
- Cons:
  - Requires some setup
  - Learning curve

Creating an R Markdown File (Rmd)



Rmd Syntax TLDR

- # Denotes order of text size
  - # is larger than ##
- You can freely type anywhere in the document
- Italic: Use *text* or _text_.
- Bold: Use **text** or __text__.
- To use R functionality, you need to create an R chunk with ```{r} ```

Rmd Syntax - Example

```
# Header
## Subheader
### Sub-Subheader
This is a summary of the cars dataset!
```{r}
summary(cars)
```
```

R Code Chunk

Header

Subheader

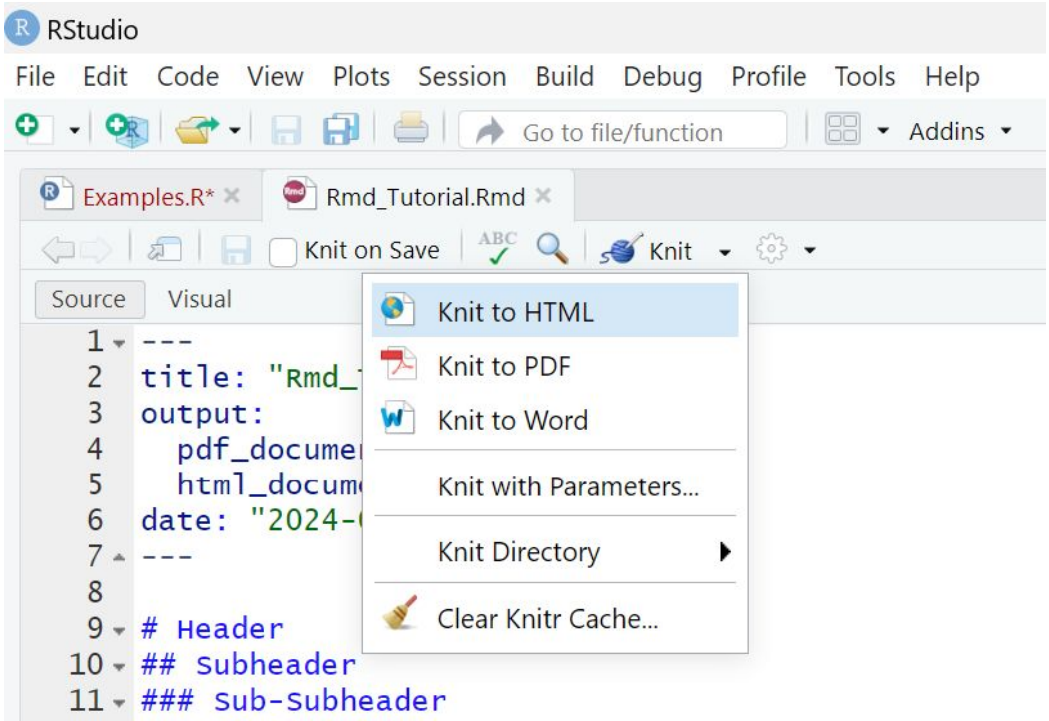
Sub-Subheader

This is a summary of the cars dataset!

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Rmd Knitting



- You can knit to different file types, which creates a document of your code!
- Knitting to html doesn't require setup, but knitting to a PDF does.

Rmd Knitting - Setup LaTeX for PDF Knitting

```
No LaTeX installation detected (LaTeX is required to create PDF output). You should
install a LaTeX distribution for your platform: https://www.latex-project.org/get/

  If you are not sure, you may install TinyTeX in R: tinytex::install_tinytex()

  Otherwise consider MiKTeX on Windows - http://miktex.org

  MacTeX on macOS - https://tug.org/mactex/
  (NOTE: Download with Safari rather than Chrome _strongly_ recommended)
```

- > install.packages(tinytex)
- > tinytex::install_tinytex()
- > # Now you should be able to knit PDFs!

Congrats!
With this you’re 60% of the way there