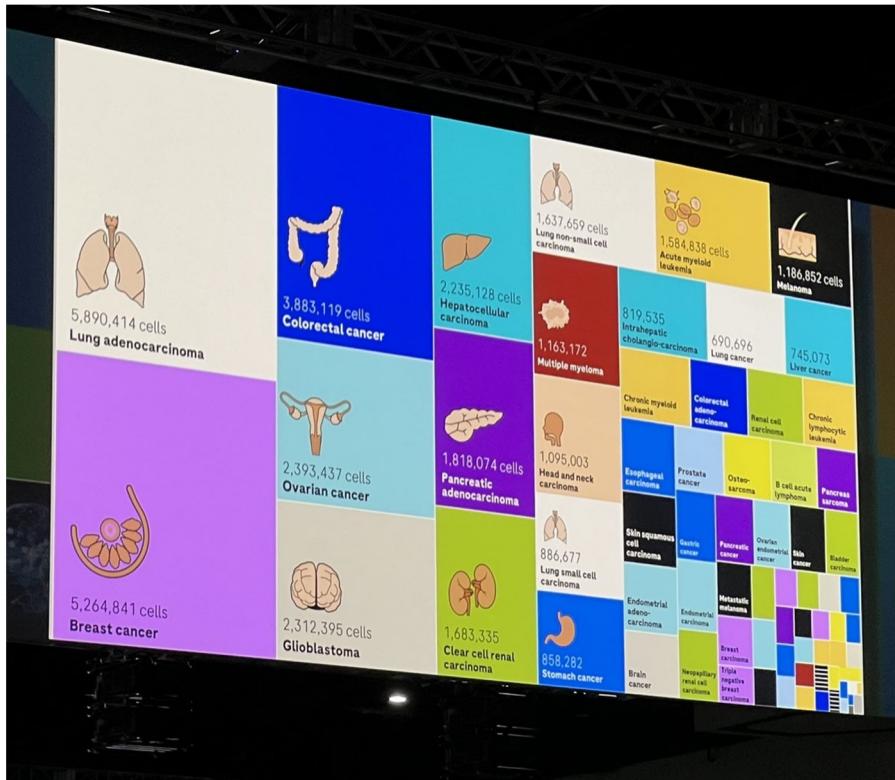


CRI Bioinformatics Bootcamp

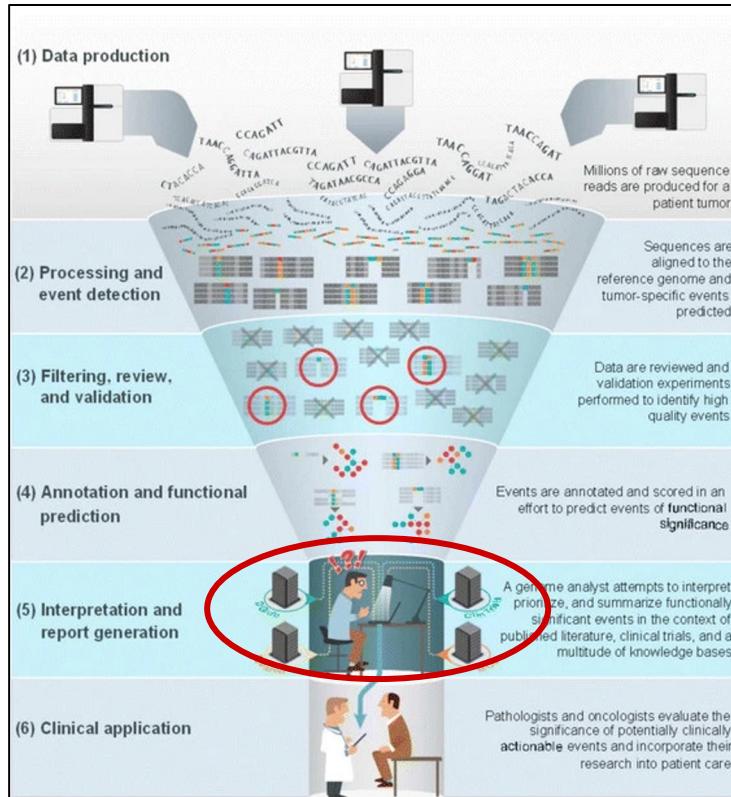
R Workshop

Why should we learn to code?

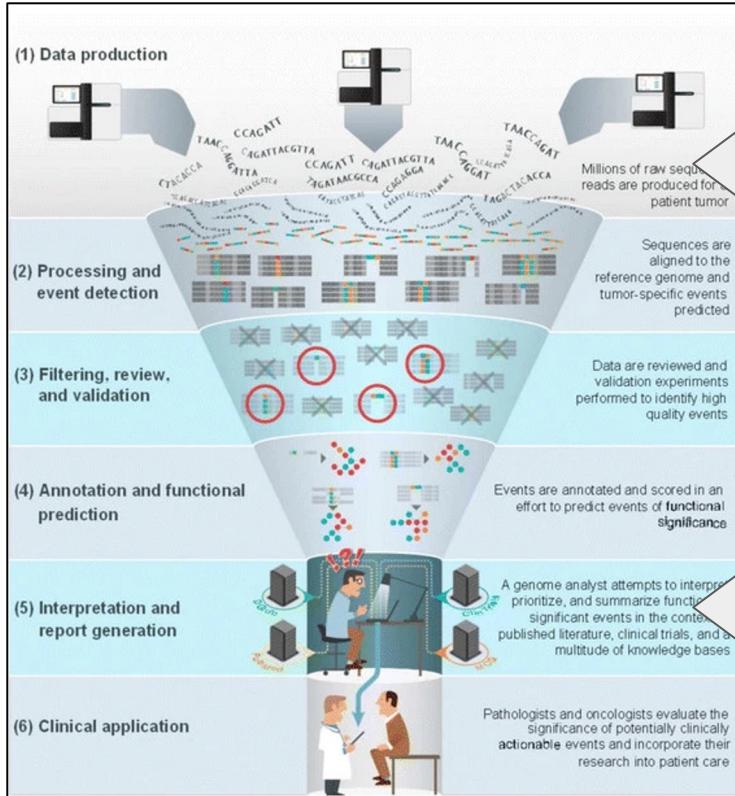


- Scale and accessibility of high-dimensional technologies
 - Value in understanding how to generate and explore datasets
 - Hypothesis-driven research through data generation

How can we make sense of the data?



What is the question?



What kind of data is or should be generated?

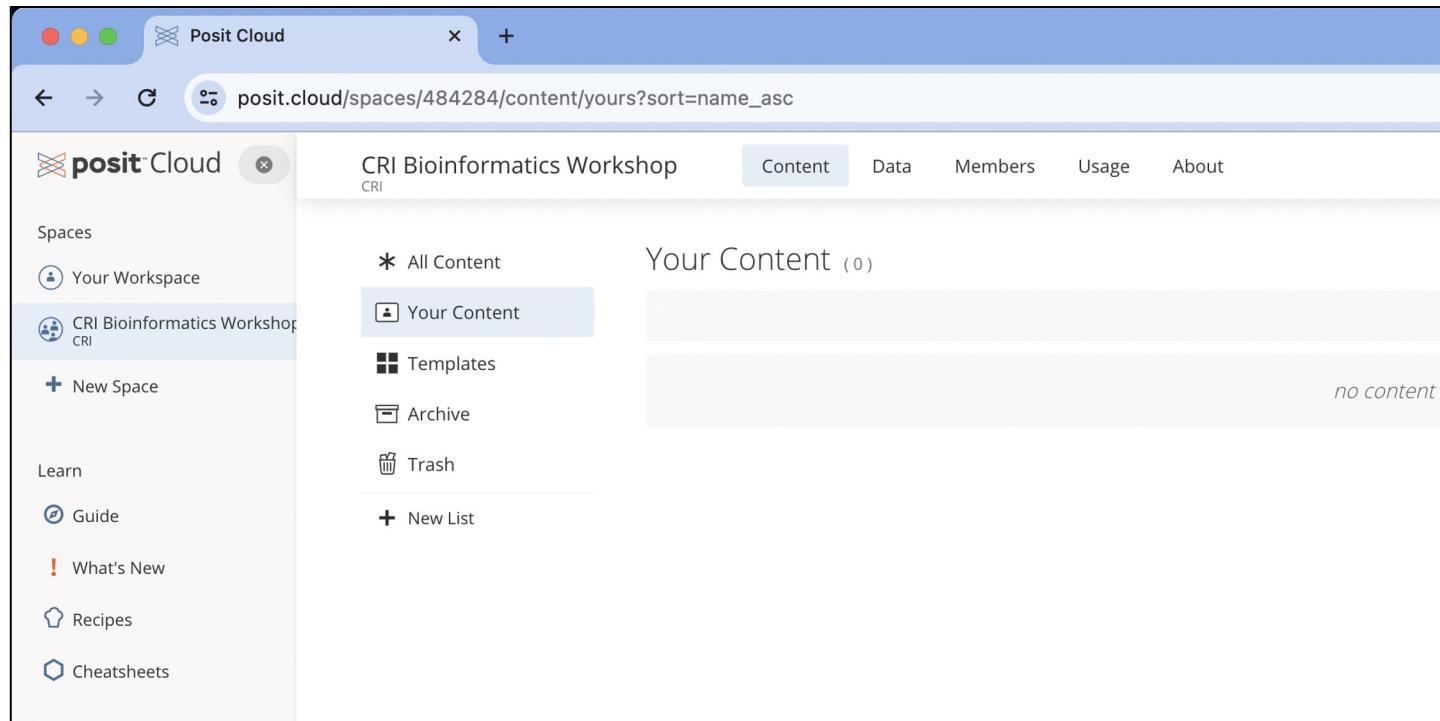
How do we alleviate the bottleneck?

What is the hypothesis?

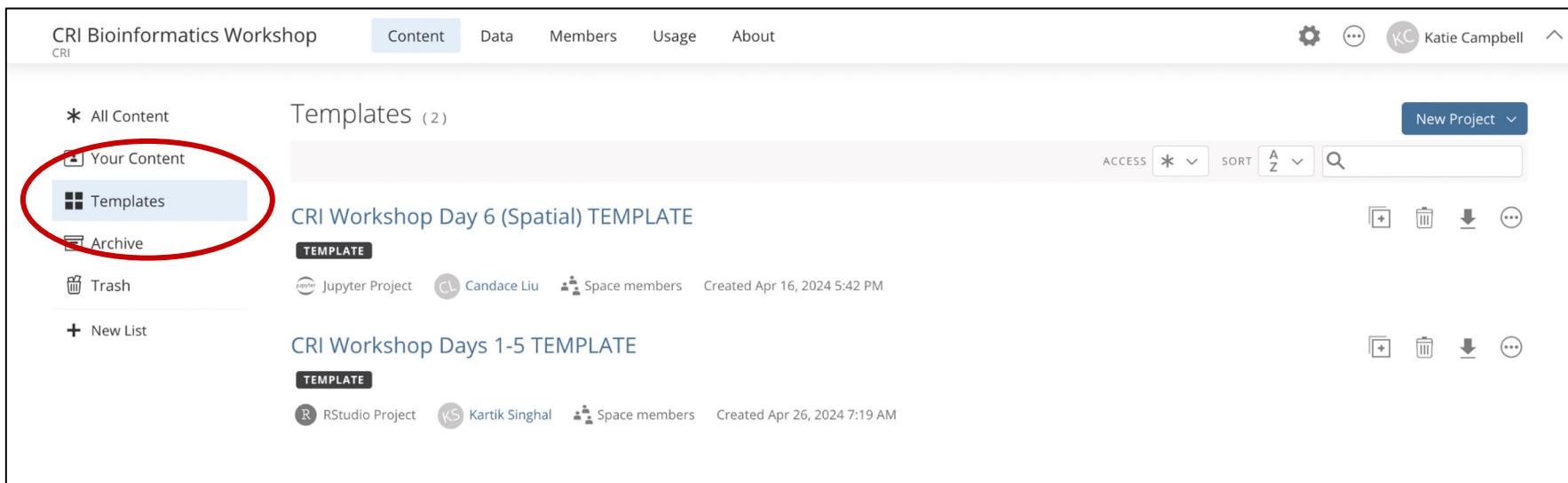
Goals for the R Workshop (Days 1-2)

- Learn enough to be a little dangerous
- Feel comfortable reading and running commands
- Understand what resources are available for when you leave

Using RStudio through Posit Cloud



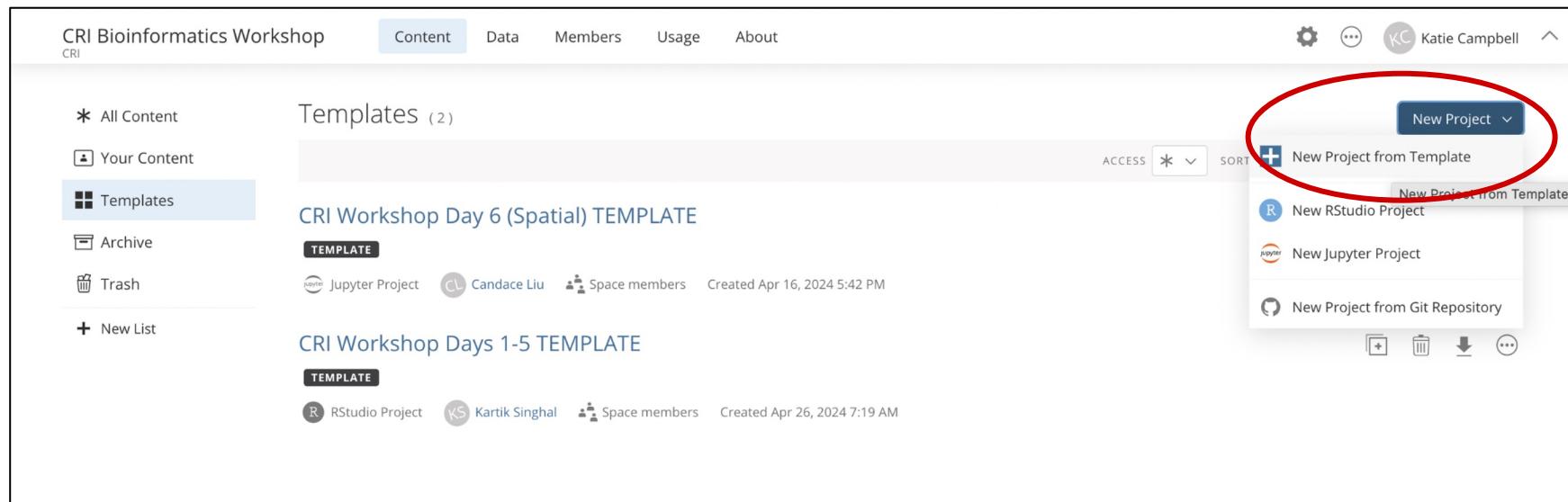
Getting started: Initiating your R project for this course



The screenshot shows the 'Content' tab selected in the navigation bar of the CRI Bioinformatics Workshop. The main area displays a list of 'Templates' (2). A red circle highlights the 'Templates' link in the sidebar, which is currently selected. The first template listed is 'CRI Workshop Day 6 (Spatial) TEMPLATE' (Jupyter Project, created by Candace Liu on April 16, 2024). The second template listed is 'CRI Workshop Days 1-5 TEMPLATE' (RStudio Project, created by Kartik Singhal on April 26, 2024). The sidebar also includes links for 'All Content', 'Your Content', 'Archive', 'Trash', and '+ New List'.

Step 1. Navigate to the **Templates** page

Getting started: Initiating your R project for this course



CRI Bioinformatics Workshop

Content Data Members Usage About

Katie Campbell

All Content

Your Content

Templates (2)

ACCESS SORT

New Project

- New Project from Template**
- New RStudio Project
- New Jupyter Project
- New Project from Git Repository

Templates

CRI Workshop Day 6 (Spatial) TEMPLATE

TEMPLATE

Jupyter Project Candace Liu Space members Created Apr 16, 2024 5:42 PM

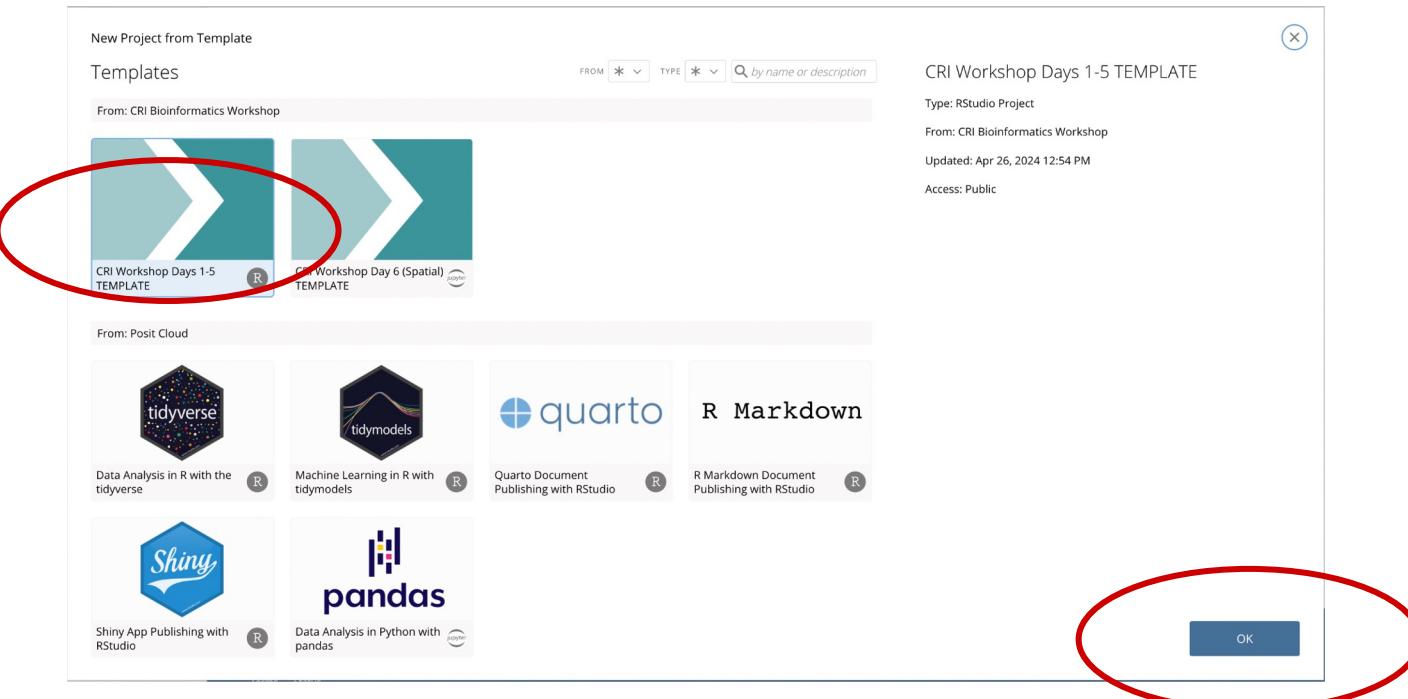
CRI Workshop Days 1-5 TEMPLATE

TEMPLATE

RStudio Project Kartik Singhal Space members Created Apr 26, 2024 7:19 AM

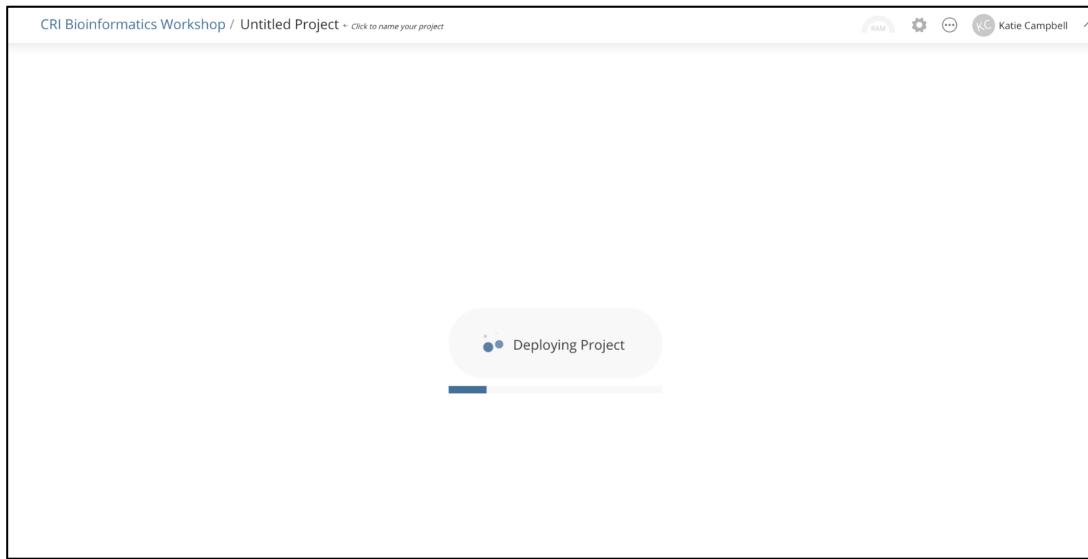
Step 2. Select **New Project > New Project from Template**

Getting started: Initiating your R project for this course



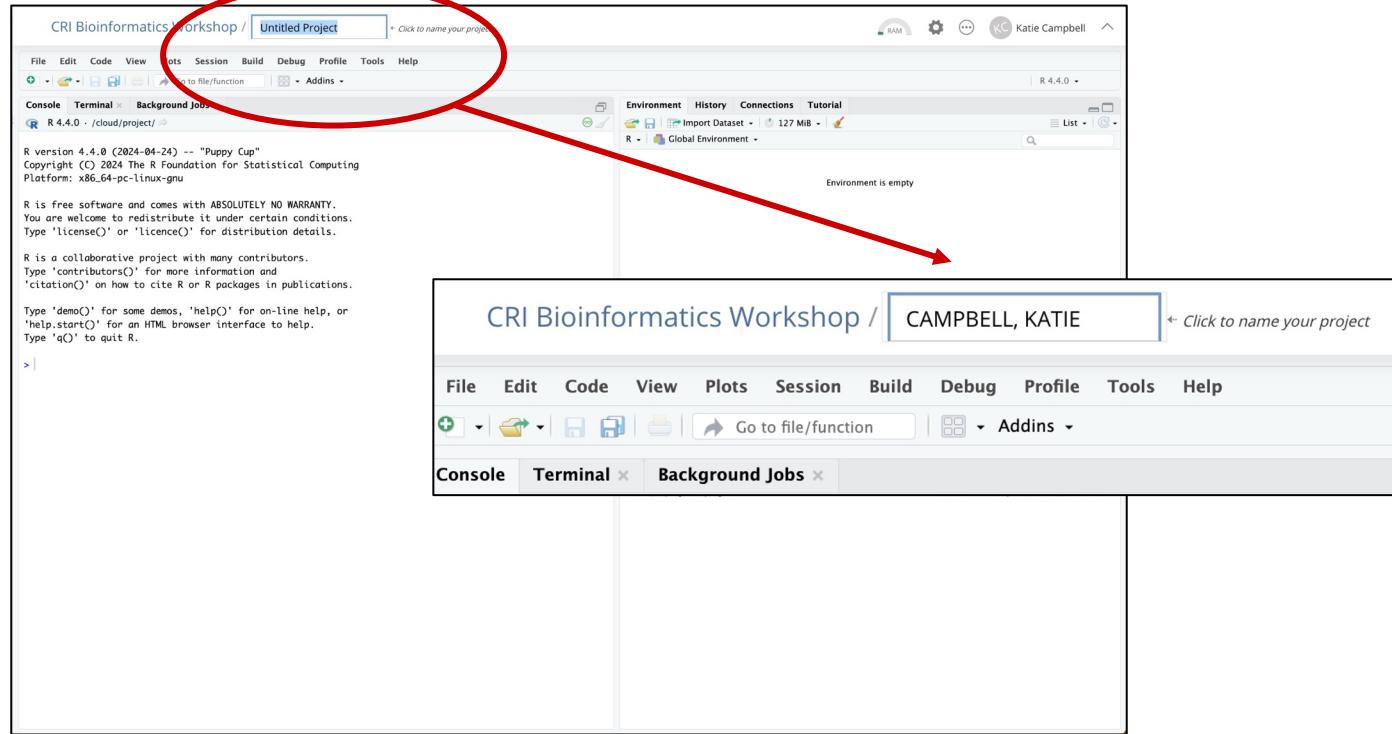
Step 3. Select **CRI Workshop Days 1-5 TEMPLATE > OK**

Getting started: Initiating your R project for this course



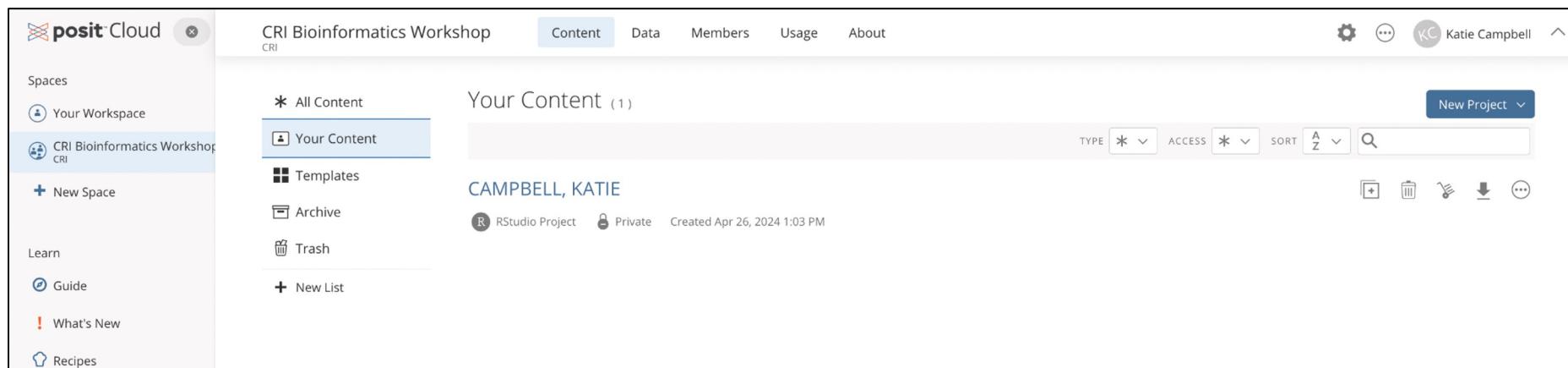
Step 4. Wait for the deployment of your project (currently called **Untitled Project**)

Getting started: Initiating your R project for this course



Step 5. Rename your project to your **LAST NAME, FIRST NAME**

Only use *your* project for each day of this course

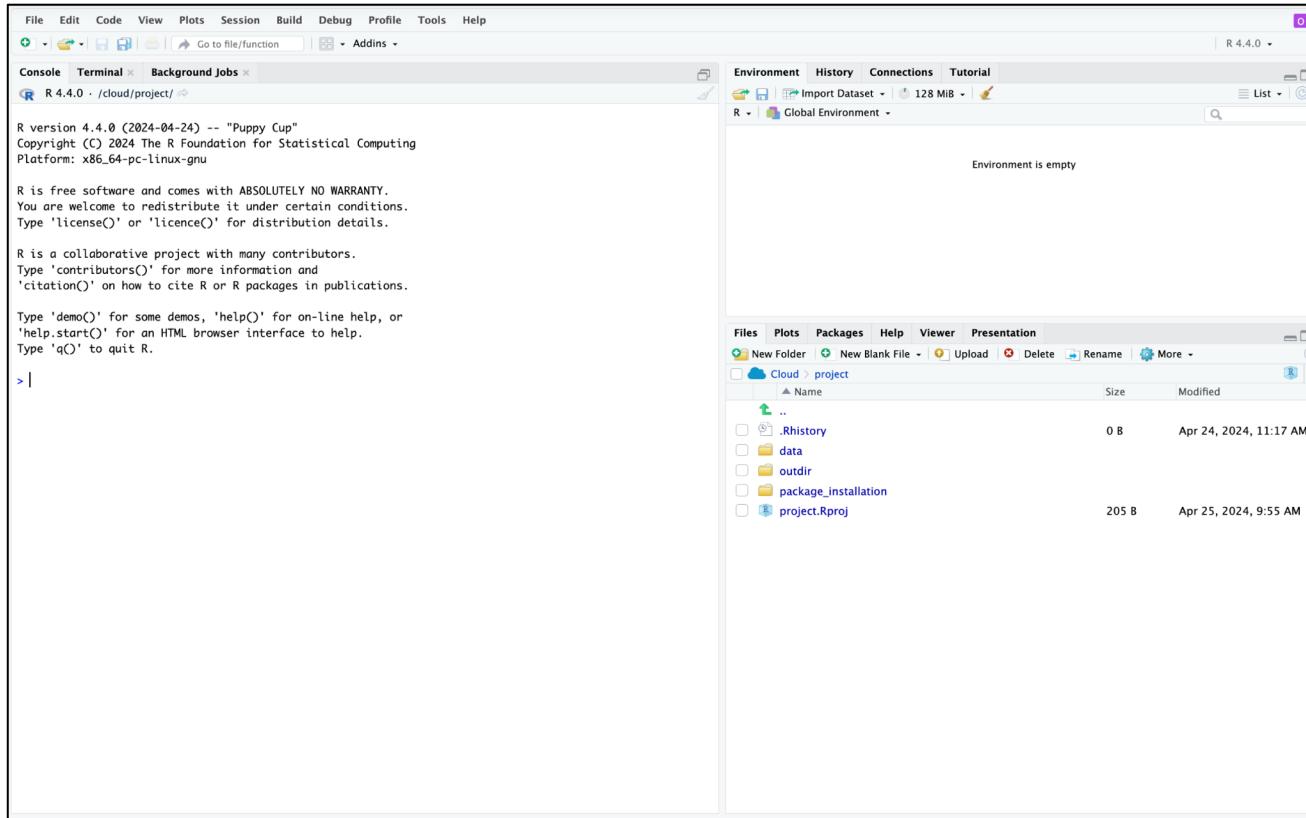


The screenshot shows the posit Cloud interface. The left sidebar has sections for 'Spaces' (Your Workspace, CRI Bioinformatics Workshop, New Space), 'Learn' (Guide, What's New, Recipes), and a search bar. The main area shows 'CRI Bioinformatics Workshop' with tabs for Content (selected), Data, Members, Usage, and About. The Content tab shows 'Your Content (1)' with a list item 'CAMPBELL, KATIE' (RStudio Project, Private, Created Apr 26, 2024 1:03 PM). There are filters for TYPE, ACCESS, SORT, and a search bar, along with a 'New Project' button and a toolbar with icons for add, delete, edit, download, and more.

This project will be yours to save your work over the course of the next 5 days. Note that this is a private project for your access, but all TAs and instructors are also able to access your project.

Introduction to R and RStudio

Orienting yourself with the RStudio interface



Creating a new R script

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The File menu is open, showing options like New File, Open File..., and Create a new R script. The main workspace shows an empty environment with the message "Environment is empty". The bottom pane displays a file browser for a project named "project" in the Cloud. The browser lists files and folders: .Rhistory (0 B, Apr 24, 2024, 11:17 AM), data, outdir, package_installation, and project.Rproj (205 B, Apr 25, 2024, 9:55 AM). A status bar at the bottom right indicates R 4.4.0.

File Edit Code View Plots Session Build Debug Profile Tools Help

New File

Open File... Open File in New Column... Recent Files Share Project... Import Dataset Save Save As... Save All Print... Close Close All Close All Except Current

Quarto Doc Create a new R script Quarto Presentation... R Notebook R Markdown... Shiny Web App... Plumber API... C File C++ File Header File Markdown File HTML File CSS File JavaScript File D3 Script Python Script Shell Script SQL Script Stan File Text File R Sweave R HTML R Documentation...

Environment History Connections Tutorial

Import Dataset 128 MB

Environment is empty

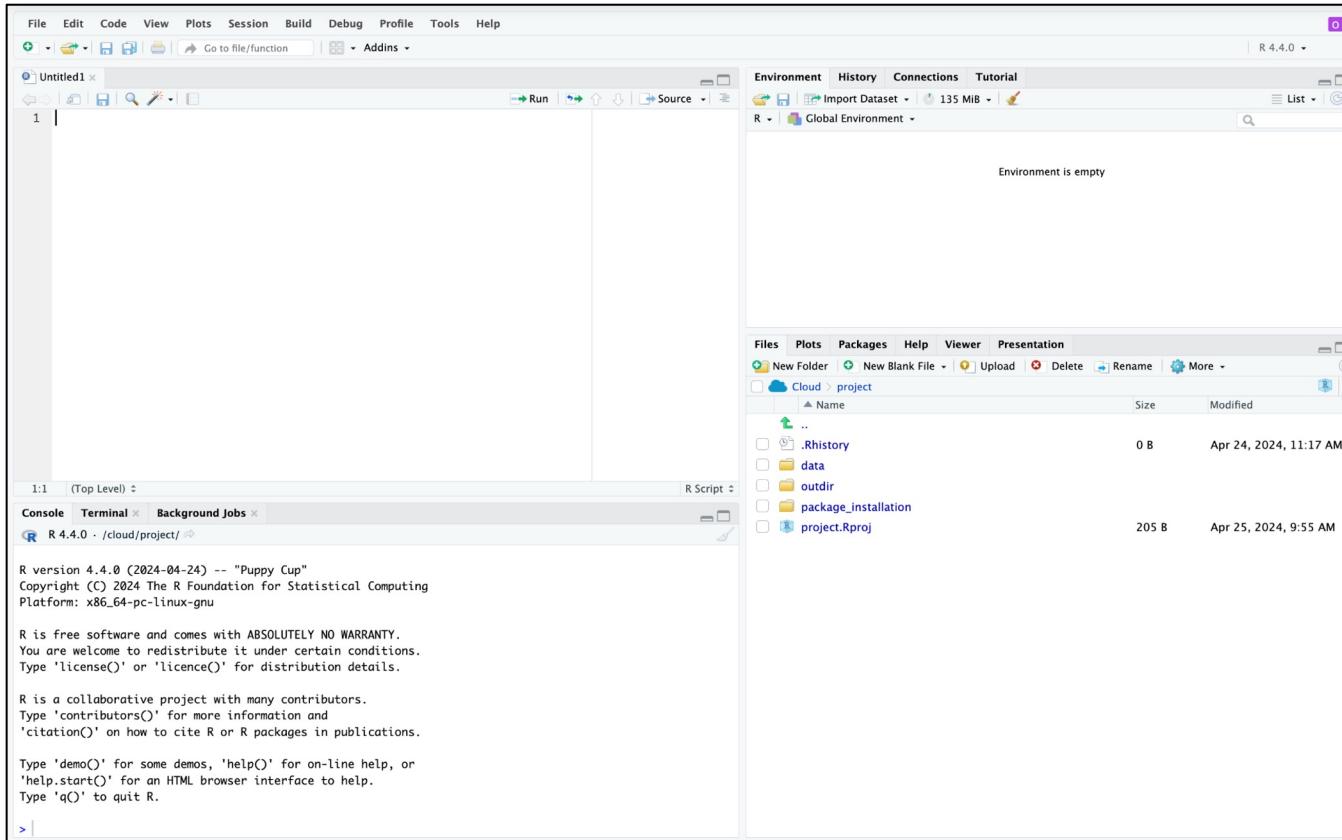
Files Plots Packages Help Viewer Presentation

New Folder New Blank File Upload Delete Rename More

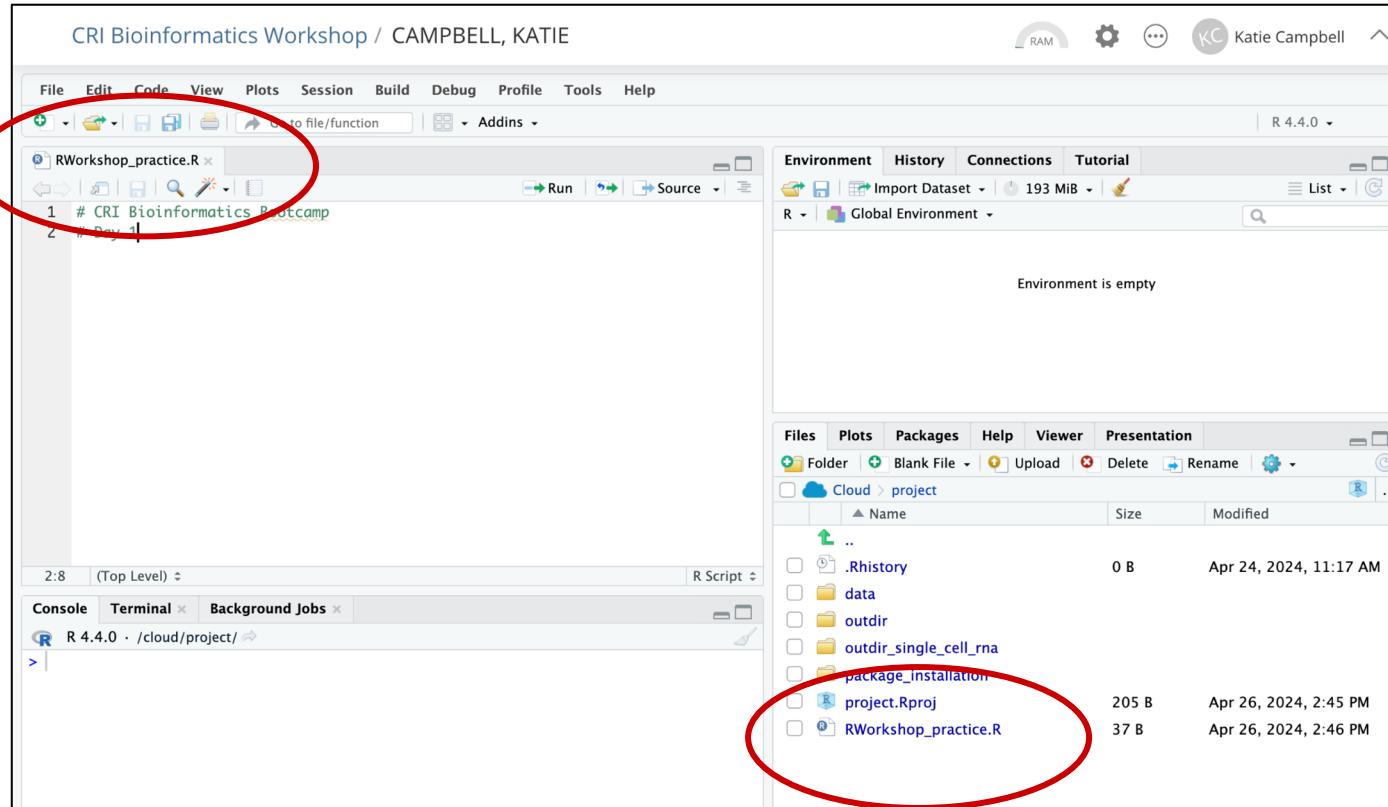
Cloud > project

Name	Size	Modified
..		
.Rhistory	0 B	Apr 24, 2024, 11:17 AM
data		
outdir		
package_installation		
project.Rproj	205 B	Apr 25, 2024, 9:55 AM

(Re)orienting yourself with the RStudio interface



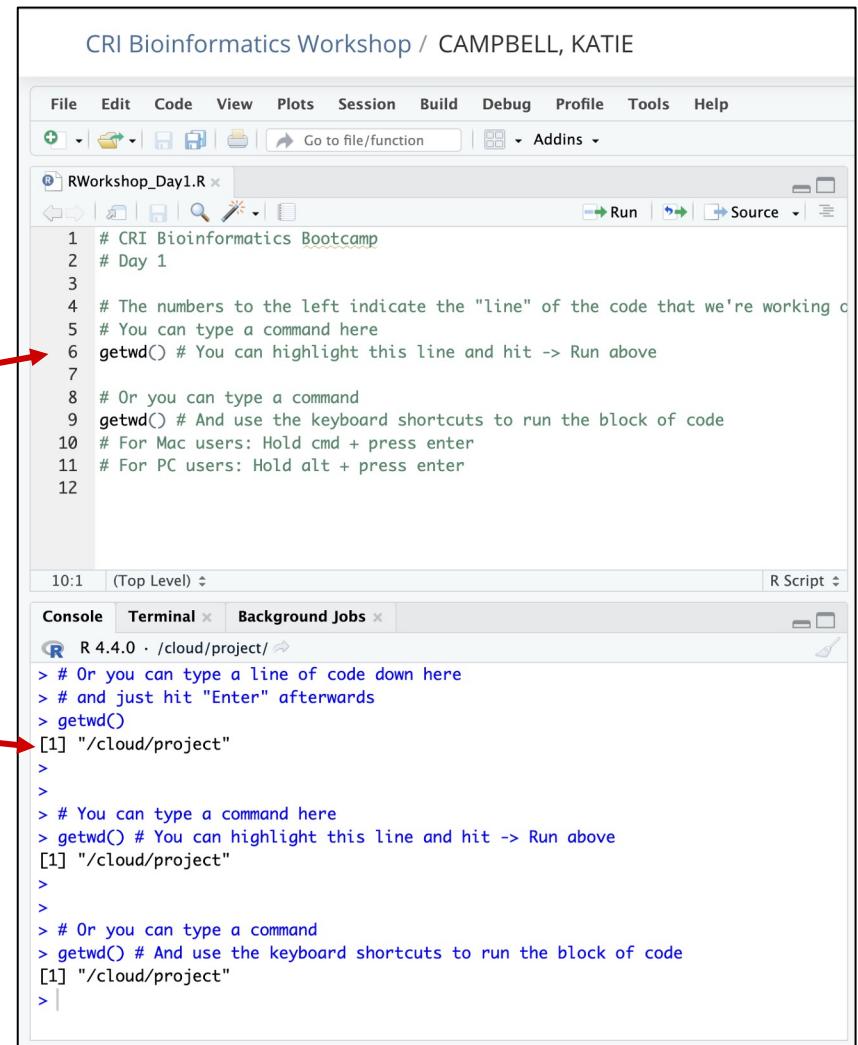
Saving your first Rscript in your project



Running your code

You can type commands directly into your script to save

Or type commands into the Console



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar below has icons for file operations and a search bar labeled 'Go to file/function'. A dropdown menu shows 'Addins'. The left sidebar has icons for file, folder, and search. The main workspace shows a script named 'RWorkshop_Day1.R' with the following content:

```
1 # CRI Bioinformatics Bootcamp
2 # Day 1
3
4 # The numbers to the left indicate the "line" of the code that we're working on
5 # You can type a command here
6 getwd() # You can highlight this line and hit -> Run above
7
8 # Or you can type a command
9 getwd() # And use the keyboard shortcuts to run the block of code
10 # For Mac users: Hold cmd + press enter
11 # For PC users: Hold alt + press enter
12
```

The status bar at the bottom indicates '10:1 (Top Level) R Script'. The bottom panel shows the 'Console' tab active, displaying R session output:

```
R 4.4.0 · /cloud/project/ ↵
> # Or you can type a line of code down here
> # and just hit "Enter" afterwards
> getwd()
[1] "/cloud/project"
>
>
> # You can type a command here
> getwd() # You can highlight this line and hit -> Run above
[1] "/cloud/project"
>
>
> # Or you can type a command
> getwd() # And use the keyboard shortcuts to run the block of code
[1] "/cloud/project"
>
```

The working directory: Orienting the filesystem of your computer

```
# Get the current working directory
current_dir <- getwd()
print(paste("Current directory:", current_dir))

# List files in the current directory
files <- list.files()
print("Files in the current directory:")
print(files)

# Change the current directory
new_dir <- "path/to/new/directory"
setwd(new_dir)
print(paste("Changed directory to:", new_dir))
```

The working directory: Orienting the filesystem of your computer

```
# Get the current working directory
current_dir <- getwd()
print(paste("Current directory:", current_dir))

# List files in the current directory
files <- list.files()
print("Files in the current directory:")
print(files)

# Change the current directory
new_dir <- "path/to/new/directory"
setwd(new_dir)
print(paste("Changed directory to:", new_dir))
```

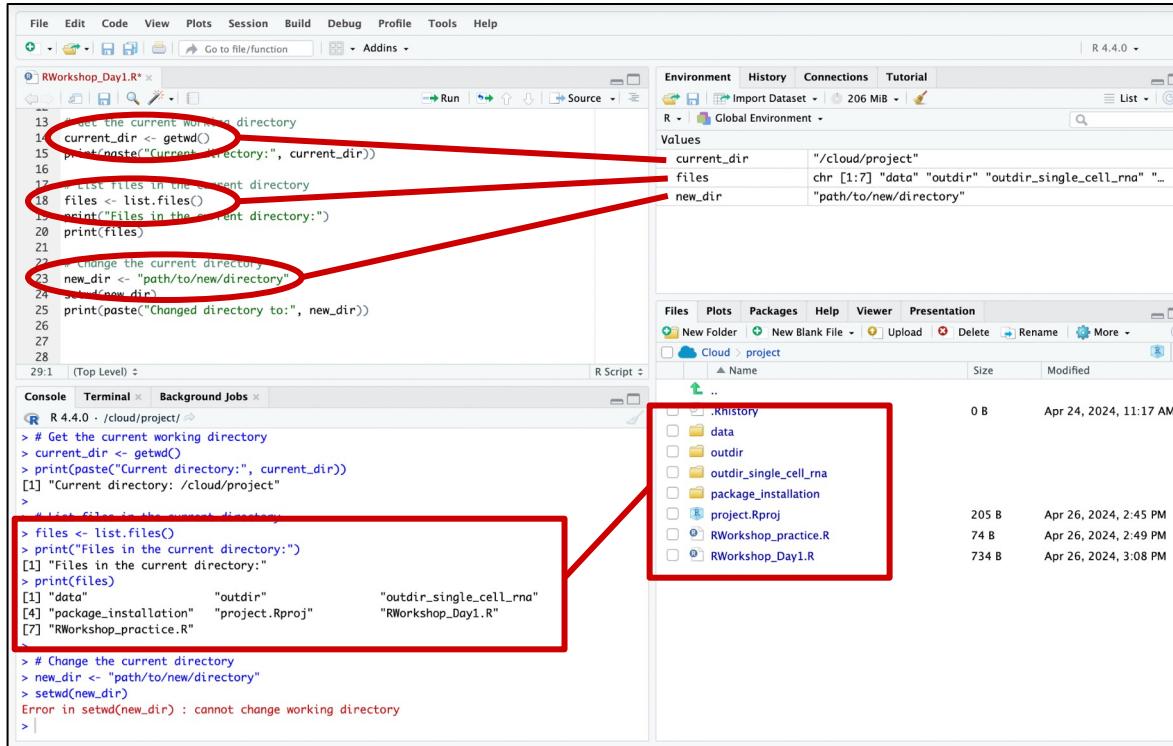
Functions are indicated by `function_name(parameters)`

`paste()` can be used to concatenate strings together

`print()` will print out what's inside the parentheses into the console

`<-` assigns the content on the right (the string) to the variable name on the left

Assignment of variables saves them within your environment



The screenshot illustrates the assignment of variables within an R environment. In the R Script pane, a script named `RWorkshop_Day1.R` is run. The code sets the current working directory, lists files in the current directory, and then changes the current directory to a new path. The `Environment` pane shows the variables `current_dir`, `files`, and `new_dir` are assigned values. The `Console` pane shows the execution of the script, including the assignment of `new_dir` and the error message when attempting to change the working directory. The `Files` pane shows the directory structure, including the newly created directory `new_dir`.

```
File Edit Code View Plots Session Build Debug Profile Tools Help
RWorkshop_Day1.R* Go to file/function Run Source Environment History Connections Tutorial R 4.4.0
13 # Set the current working directory
14 current_dir <- getwd()
15 print(paste("Current directory:", current_dir))
16
17 # List files in the current directory
18 files <- list.files()
19 print("Files in the current directory:")
20 print(files)
21
22 # Change the current directory
23 new_dir <- "path/to/new/directory"
24 setwd(new_dir)
25 print(paste("Changed directory to:", new_dir))

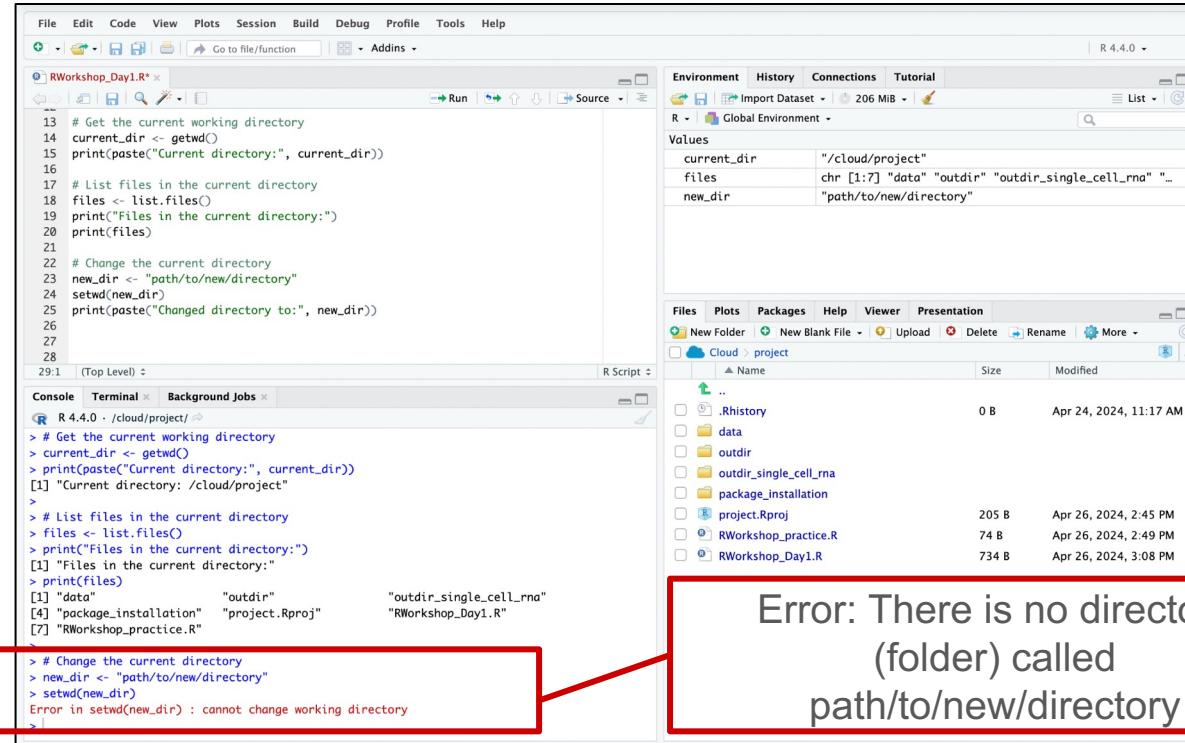
29.1 (Top Level) R Script
Console Terminal Background Jobs
R 4.4.0 - /cloud/project/ ~
> # Get the current working directory
> current_dir <- getwd()
> print(paste("Current directory:", current_dir))
[1] "Current directory: /cloud/project"
>
> # List files in the current directory
> files <- list.files()
> print("Files in the current directory:")
[1] "Files in the current directory:"
> print(files)
[1] "data"           "outdir"          "outdir_single_cell_rna"
[4] "package_installation" "project.Rproj" "RWorkshop_Day1.R"
[7] "RWorkshop_practice.R"

> # Change the current directory
> new_dir <- "path/to/new/directory"
> setwd(new_dir)
Error in setwd(new_dir) : cannot change working directory
> |
```

Values	current_dir	"/cloud/project"	files	chr [1:7] "data" "outdir" "outdir_single_cell_rna" ...	new_dir	"path/to/new/directory"
--------	-------------	------------------	-------	--	---------	-------------------------

Files	Plots	Packages	Help	Viewer	Presentation	..
New Folder	New Blank File	Upload	Delete	Rename	More	..
Cloud	project					
Name	Size	Modified				
.Rhistory	0 B	Apr 24, 2024, 11:17 AM				
data						
outdir						
outdir_single_cell_rna						
package_installation						
project.Rproj	205 B	Apr 26, 2024, 2:45 PM				
RWorkshop_practice.R	74 B	Apr 26, 2024, 2:49 PM				
RWorkshop_Day1.R	734 B	Apr 26, 2024, 3:08 PM				

Be prepared for Errors...



The screenshot shows the RStudio interface with the following components:

- Code Editor:** An R script titled "RWorkshop_Day1.R" is open. The code attempts to get the current working directory, list files in the current directory, change the current directory to "path/to/new/directory", and then print the new directory. The code is as follows:

```
13 # Get the current working directory
14 current_dir <- getwd()
15 print(paste("Current directory:", current_dir))
16
17 # List files in the current directory
18 files <- list.files()
19 print("Files in the current directory:")
20 print(files)
21
22 # Change the current directory
23 new_dir <- "path/to/new/directory"
24 setwd(new_dir)
25 print(paste("Changed directory to:", new_dir))
26
27
28
29.1 (Top Level) R Script
```

- Console:** The output of the R script is shown. It prints the current directory as "/cloud/project", lists files in the directory, and then attempts to change the working directory to "path/to/new/directory". The final line shows an error message: "Error in setwd(new_dir) : cannot change working directory".

```
R 4.4.0 - /cloud/project/ 
> # Get the current working directory
> current_dir <- getwd()
> print(paste("Current directory:", current_dir))
[1] "Current directory: /cloud/project"
>
> # List files in the current directory
> files <- list.files()
> print("Files in the current directory:")
[1] "Files in the current directory:"
> print(files)
[1] "data"                      "outdir"                    "outdir_single_cell_rna"
[4] "package_installation"      "project.Rproj"             "RWorkshop_Day1.R"
[7] "RWorkshop_practice.R"
>
> # Change the current directory
> new_dir <- "path/to/new/directory"
> setwd(new_dir)
Error in setwd(new_dir) : cannot change working directory
>
```

- Environment:** The environment pane shows the variables defined in the script: current_dir, files, and new_dir.
- File Explorer:** The file explorer pane shows the project structure in the "Cloud" workspace, including files like .Rhistory, data, outdir, and RWorkshop_Day1.R.

A red box highlights the error message in the console, and a red callout box points to it with the text "Error: There is no directory (folder) called path/to/new/directory".

Primitive Variables in R

- **Most basic data types**
 - Cannot be decomposed
- **Numeric (Float)**
 - Default variable for numbers
 - Includes floating point and integer
- **Integer**
- **Logical (Boolean)**
 - **TRUE** or **FALSE**
 - Conditional Statements
 - And/Or/Not
- **Character (String)**
 - Text data
 - "Hello world!"
- **Other types**
 - Raw (Byte Representation)
 - Complex
- **Data structures**
 - Composite of primitive

Numeric

```
# Numeric variable
num_var <- 10
print(num_var) # Output: 10

# Arithmetic operations
result <- num_var * 2
print(result) # Output: 20
```

Integer

```
# Integer variable
int_var <- 20L # The 'L' suffix indicates an integer
print(int_var) # Output: 20

# Integer arithmetic
result <- int_var / 5
print(result) # Output: 4
```

Character

```
# Character variable
char_var <- "Hello, World!"
print(char_var) # Output: Hello, World!

# Concatenation
new_string <- paste(char_var, "This is R programming.")
print(new_string) # Output: Hello, World! This is R programming.
```

Vector

Vector and List

- **Vector**

- 1 dimensional
- Single data type
 - Numeric, Character, Logical
- Efficient (pre-allocation)
- Building Matrices

- **List**

- Heterogeneous data
- Dynamic
- Creating Dataframes

- **Factor**

- Categorical data
- Levels
- Memory efficient
 - Fixed set of values can be mapped with an integer

```
# Creating a numeric vector
numeric_vector <- c(1, 2, 5.3, 6, -2, 4)

# Creating a character vector
character_vector <- c("apple", "banana", "cherry")

# Creating a logical vector
logical_vector <- c(TRUE, FALSE, TRUE, FALSE)

#Pre-allocate
my_vector <- numeric(10)
```

List

```
# Creating a list with mixed types
my_list <- list(
  barcode = "AAAGCAAGTTACGCGC",
  umis = 25,
  seq = c("CASSLVLPGYTEAFF", "CASVGQGEYEQYF"),
  cell = TRUE,
)

# Accessing elements
print(my_list$barcode)      # Outputs "AAAGCAAGTTACGCGC"
print(my_list$seq)          # Outputs c("CASSLVLPGYTEAFF", "CASVGQGEYEQYF")
```

Matrix

- 2 dimensional
- Single data type
- *matrix()*
 - data
 - nrow
 - ncol
 - byrow
 - Whether to fill by rows or columns
 - dimnames'
- Access elements by row and column
- Matrix operations

Creating a Matrix:

```
# Create a matrix with random values
mat <- matrix(rnorm(20), nrow = 4, ncol = 5)
```

Matrix operations

```
element <- mat[1, 2]

# Calculate row sums
row_sums <- rowSums(mat)

# Calculate column sums
col_sums <- colSums(mat)

# Create another matrix
mat2 <- matrix(rnorm(20), nrow = 5, ncol = 4)

# Perform matrix multiplication
mat_mult <- mat %*% mat2

# Transpose the matrix
mat_transpose <- t(mat)

# Select the first two rows
first_two_rows <- mat[1:2, ]

# Select the first three columns
first_three_cols <- mat[, 1:3]
```

Dataframe

- **What is a dataframe?**

- “Excel”-like structure
- Heterogeneous data
- Labeled columns and rows
- `data.frame()`

- **Using the dataframe**

- Accessing columns by name
 - `df$column_name`
- Access by row and column
 - `df[i,j]`
- Filtering
 - `subset(df, column_name > 25)`
- Add columns
 - `df$col2 <- c(1,2,3)`
- Add rows
 - `df <- rbind(df, c("w",2,TRUE))`
- Modify values
 - `df[i,j] <- 25`

```
# Create a data.frame
df <- data.frame(
  barcode = c("AACCTGAGAAGGCCT-1", "AACCTGCAAGTTGTC-1", "AACCTGCACCGATAT-1"),
  phenotype = c("TEMRA", "Naive", "Dysfunctional"),
  treatment = c("Pre", "Post", "Pre"),
  response = c("R","NR","R")
)

result <- df$phenotype
print(result) # prints "TEMRA","Naive","Dysfunctional"

result <- df[1,2]
print(result) # prints "TEMRA"

result <- subset(df,treatment=="Post")
print(result) # prints AACCTGCAAGTTGTC-1 Naive Post NR

df$site <- c("Tumor","Tumor","Blood")

df <- rbind(df,c("AACCTGCACCGATAT-1","Naive","Post","R","Tumoxr"))

df[4,5] <- "Tumor"
```

```
> df
      barcode phenotype treatment response site
1 AACCTGAGAAGGCCT-1      TEMRA      Pre        R Tumor
2 AACCTGCAAGTTGTC-1      Naive      Post       NR Tumor
3 AACCTGCACCGATAT-1 Dysfunctional      Pre        R Blood
4 AACCTGCACCGATAT-1      Naive      Post       R Tumor
```

String Manipulation

- Strings vs substrings
 - Substring is a sequence of text within a string.
 - “Hello” is a substring of “Hello world!”
- Case conversion
 - Uppercase and lowercase matter, R is case-sensitive
- Find and replace
 - Find a substring and replace it with another substring
 - Examples: ID matching, motifs, etc.
- Length calculation
 - Sequence length
- Stringr package (tidyverse)
 - str_length
 - str_to_upper
 - str_locate
 - str_sub
 - str_replace and str_replace_all
- Base: grep, grepl, gsub
- Regular expressions
 - Search pattern
 - “hell*olrd!” matches “hello world!”

Length Calculation

Determining the length of sequences.

```
sequence_length <- str_length(sequence)
print(sequence_length)
```

Case Conversion

Converting uppercase to lowercase, or vice versa.

```
sequence_upper <- str_to_upper(sequence)
print(sequence_upper)
```

Substring Extraction

Extracting parts of sequences, such as cutting out genes or regions of interest.

```
extracted_sequence <- str_sub(sequence, 3, 8)
print(extracted_sequence)
```

Replacing Substrings

Modifying sequences by replacing specific nucleotides or amino acids.

```
dna_sequence <- "ATGCGTACGTTGACT"
rna_sequence <- str_replace_all(dna_sequence, "T", "U")
print(rna_sequence)
```

Logical operations

- Evaluate to True or False
 - logical data type
- Use case
 - Conditional statements
 - if / else
 - Filtering data
 - `subset(df,treatment="post")`
- Comparisons
 - Equal: `==`
 - Not equal: `!=`
 - Greater Than: `>`
 - Less Than `<`
 - `%in%`
 - Test inclusion of one vector in another
- Boolean statements
 - and: both are true `&`
 - or: atleast one is true `|`
 - not: evaluates to false `!"`

```
# Greater than
result_gt <- x > y
print(result_gt) # Output: FALSE
```

```
# Less than
result_lt <- x < y
print(result_lt) # Output: TRUE
```

```
# Equal to
result_eq <- x == y
print(result_eq) # Output: FALSE
```

```
# Not equal to
result_neq <- x != y
print(result_neq) # Output: TRUE
```

```
# Logical operations
a <- TRUE
b <- FALSE

# AND operation
result_and <- a & b
print(result_and) # Output: FALSE

# OR operation
result_or <- a | b
print(result_or) # Output: TRUE
```

```
# NOT operation
result_not <- !a
print(result_not) # Output: FALSE
```

```
x <- 3
if (x > 5) {
  print("x is greater than 5")
} else if (x == 5) {
  print("x is equal to 5")
} else {
  print("x is less than 5")
}
```

Writing your own functions

- Encapsulate reusable block of code for a specific task
- Elements of a function
 - Name
 - Input (Arguments)
 - Body
 - Output (Optional)
 - Writing to a file
 - Plotting
- Why write functions?
 - Breaking down larger problems
 - Reusability
 - Minimizing errors
 - Maintainability

```
function_name <- function(argument1, argument2, ...) {  
  # Function body  
  # Perform operations  
  # Return a value (optional)  
}
```

```
# Define a function to calculate the square of a number  
square <- function(x) {  
  return(x^2)  
}  
  
# Call the function  
result <- square(5)  
print(result) # Output: 25
```

Manipulating data frames in tidyverse

What “Tidy” means:

- Each variable is a column
- Each observation is a row
- A table is a consistent set of observations for a known set of variables.

```
library(tidyverse)

# Create a tibble (tidyverse version of data.frame)
df <- tibble(
  barcode = c("AAACCTGAGAAGGCCT-1", "AAACCTGCAAGTTGTC-1", "AAACCTGCACCGATAT-1"),
  phenotype = c("TEMRA", "Naive", "Dysfunctional"),
  treatment = c("Pre", "Post", "Pre"),
  response = c("R", "NR", "R")
)
```

Tidyverse Packages for Data Manipulation:

- **dplyr**: Provides a grammar of data manipulation for data frames, enabling easy filtering, selecting, mutating, summarizing, and arranging of data.
- **tidyverse**: Offers tools for reshaping and tidying messy datasets, such as `gather()` and `spread()` functions for converting between wide and long formats.
- **ggplot2**: Allows for the creation of sophisticated data visualizations using a layered grammar of graphics.

Nice features:

- Printing
- Variable information
- Extracting a column for a tibble yields another tibble
 - data.frame will yield a vector
- Row index not added by default
 - Explicit id column

Tidyverse Operations

Filter

- Conditional subsetting
- Multi-condition

Mutate

- Add new columns
- Modify columns
- vectorized
 - Applied to column

Pipe: “%>%”

- Chaining commands
- Readability

```
# Filter rows where treatment is "Post"
result <- df %>%
  filter(treatment == "Post")
print(result) # prints the row with treatment "Post"

# Add a new column 'site' and update its values
df <- df %>%
  mutate(site = c("Tumor", "Tumor", "Blood"))

# Add a row to the dataframe
df <- df %>%
  add_row(
    barcode = "AAACCTGCACCGATAT-1",
    phenotype = "Naive",
    treatment = "Post",
    response = "R",
    site = "Tumor"
  )
```

Reading in your own files

CSV/TSV

- `read.csv` or `read_csv` (“tidyverse”)
- Comma separated or tab separated

Excel

- `readxl()`
- sheet and cell selection

Understanding the structure of your data

- `str()`
 - Shows human readable structure of any R object (data.frame, tibble, etc.)
- `summary()`
 - Adaptive to type of object
 - Summarizes mean, median, etc. for numeric data
 - Shows model diagnostics

```
# Read a CSV file into a data frame
data <- read_csv("path/to/your/file.csv")

# View the structure of the data frame
print("Structure of the data frame:")
print(str(data))

# View the first few rows of the data frame
print("First few rows of the data frame:")
print(head(data))
```

```
# Read a specific sheet by name
data_sheet2 <- read_excel("path/to/your/file.xlsx", sheet = "SecondSheet")

# Read a specific sheet by index
data_sheet3 <- read_excel("path/to/your/file.xlsx", sheet = 3)

# Read specific cells, e.g., A1:C10 from the first sheet
data_range <- read_excel("path/to/your/file.xlsx", range = "Sheet1!A1:C10")
```

```
# Create a numeric vector
numeric_vector <- c(23, 45, 67, 23, 89, 45, 12, 54, NA, 78)

# Use summary to get a statistical overview of the vector
summary_result <- summary(numeric_vector)
print(summary_result)
```

```
> print(summary_result)
   Min. 1st Qu. Median      Mean 3rd Qu.      Max.    NA's
12.00   23.00   45.00   48.44   67.00   89.00        1
```

Hands-on: Reading, writing, and interpreting data structures

- These slides are available on the course website under **Resources: During the course: Lecture slides**
 - Set up your project through Posit
- Start coding
 - Start your first Rscript
 - Work through the blocks of code under the **Course: Introduction to R** page
 - Review the details on how the code works in the Lecture slides for assistance
 - Put a post-it on your laptop if you get stuck, indicating for a TA to come up to you
 - Work through the blocks of code on this page, practicing in both your Rscript and the terminal
 - Taking the next step
 - There are a list of **Additional exercises** at the bottom of the page for you to try on your own

Goal: Start building your R vocabulary