



# GETTING STARTED

## With R Programming

Patricia C. Muskus



OIST OKINAWA INSTITUTE  
OF SCIENCE AND TECHNOLOGY



# What to Expect?

---

## **This workshop IS:**

- A quick survey on R programming
- An introduction to RStudio (local and cluster)
- Organization, workflows, available resources

---

## **This workshop is NOT:**

- A deep dive into R programming
- Analysis & interpreting results

---

## **Before we begin, do you have...**

- R installed locally
- RStudio (optional)
- Access to Deigo (optional)
- Familiarity with terminal basics

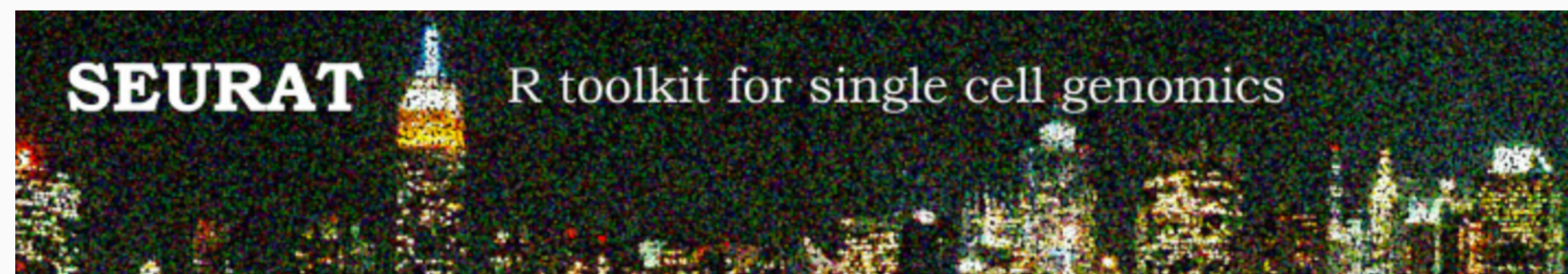
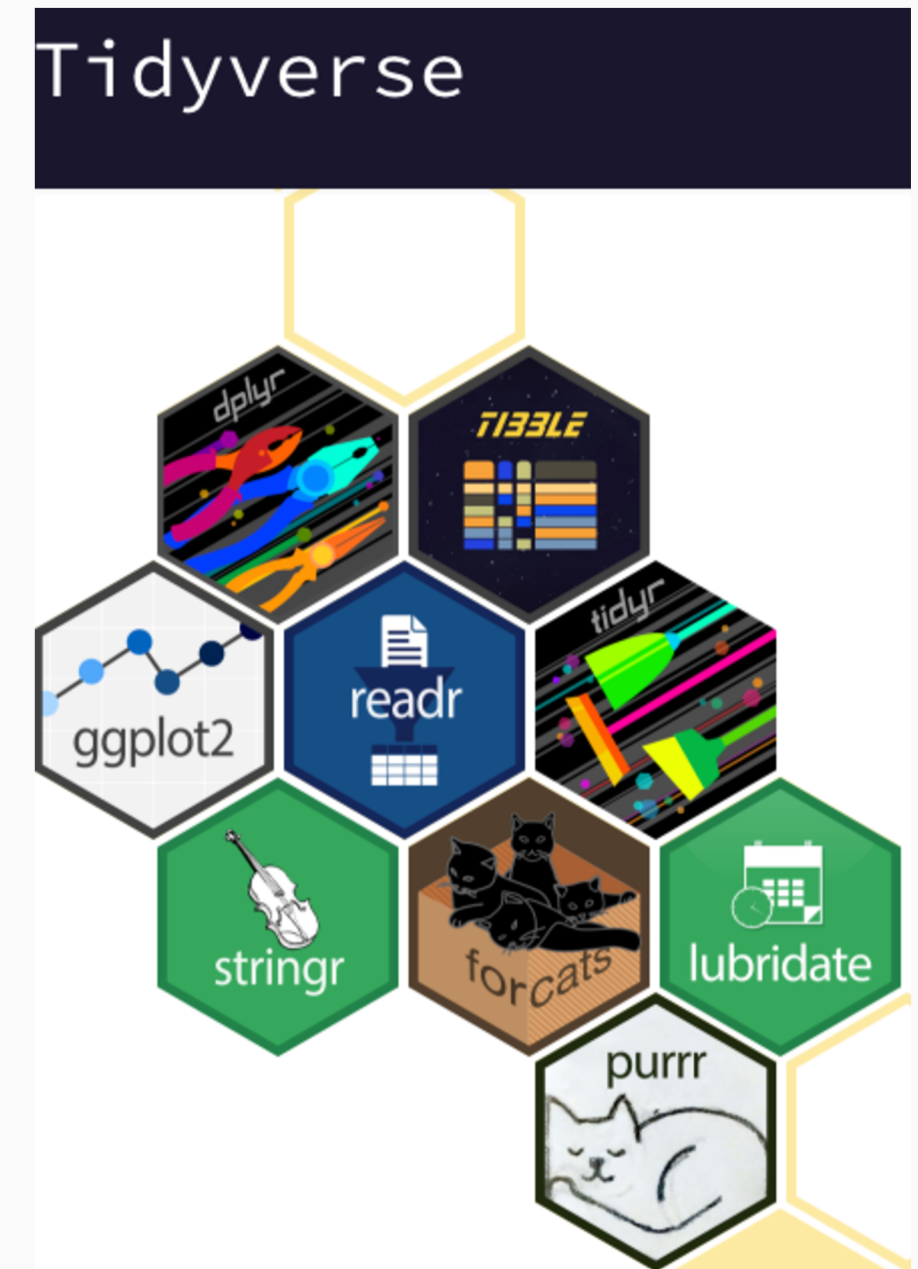




# R Documentation & Packages

R Documentation

R Packages



# R on the Command Line

```
patricia-muskus@patricia-muskus-Latitude-3340: ~  
R version 4.5.1 (2025-06-13) -- "Great Square Root"  
Copyright (C) 2025 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu  
  
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.  
  
> #Simple Calculations  
> 2+5  
[1] 7  
> mean(3,5,6,2)  
[1] 3  
> #
```



# R GUI: R-Studio

The screenshot shows the RStudio interface with several components highlighted by blue text labels:

- Source Editor:** The central pane containing R code for simple calculations, vectors, data frames, and functions.
- Environment:** The right-hand pane showing the current environment with variables like 'df' and 'fold\_change\_fu...'. It includes a table of values for the 'df' data frame.
- Console:** The bottom-left pane showing the execution of the code from the source editor, resulting in a printed data frame.
- Plots, Files, Packages:** The right-hand pane below the environment, which is currently empty.

```
1 #Simple Calculations
2 2+5
3 mean(3,5,6,2)
4
5 #Vectors
6 genes <- c("BRCA1","BRCA2","TP53","CDH1")
7 ctrl <- c("3.2","5.2","4.7","7.9")
8 treatment <- c("10.2","9.8","10.4","7.4")
9
10 #Data Frames
11
12 df <- data.frame(genes=genes,control=ctrl,treatment=treatment)
13
14 str(df)
15
16 df$control <- as.numeric(df$control)
17 df$treatment <- as.numeric(df$treatment)
18
19 #Functions
20 fold_change_function <- function(treat,ctrl){treat/ctrl}
21
22 df$FC <- fold_change_function(df$treatment,df$control)
23
24 df
25
```

ctrl	chr [1:4]	"3.2"	"5.2"	"4.7"	"7.9"
genes	chr [1:4]	"BRCA1"	"BRCA2"	"TP53"	"CDH1"
treatment	chr [1:4]	"10.2"	"9.8"	"10.4"	"7.4"

```
> #Functions
> fold_change_function <- function(treat,ctrl){treat/ctrl}
> df$FC <- fold_change_function(df$treatment,df$control)
> df
  genes control treatment      FC
1 BRCA1    3.2    10.2 3.1875000
2 BRCA2    5.2     9.8 1.8846154
3 TP53    4.7    10.4 2.2127660
4 CDH1    7.9     7.4 0.9367089
```



# Quick Notes on R

---

**R's core syntax Base R**- the language, data structures, statistical functions, and graphics system are all available immediately after installing R.

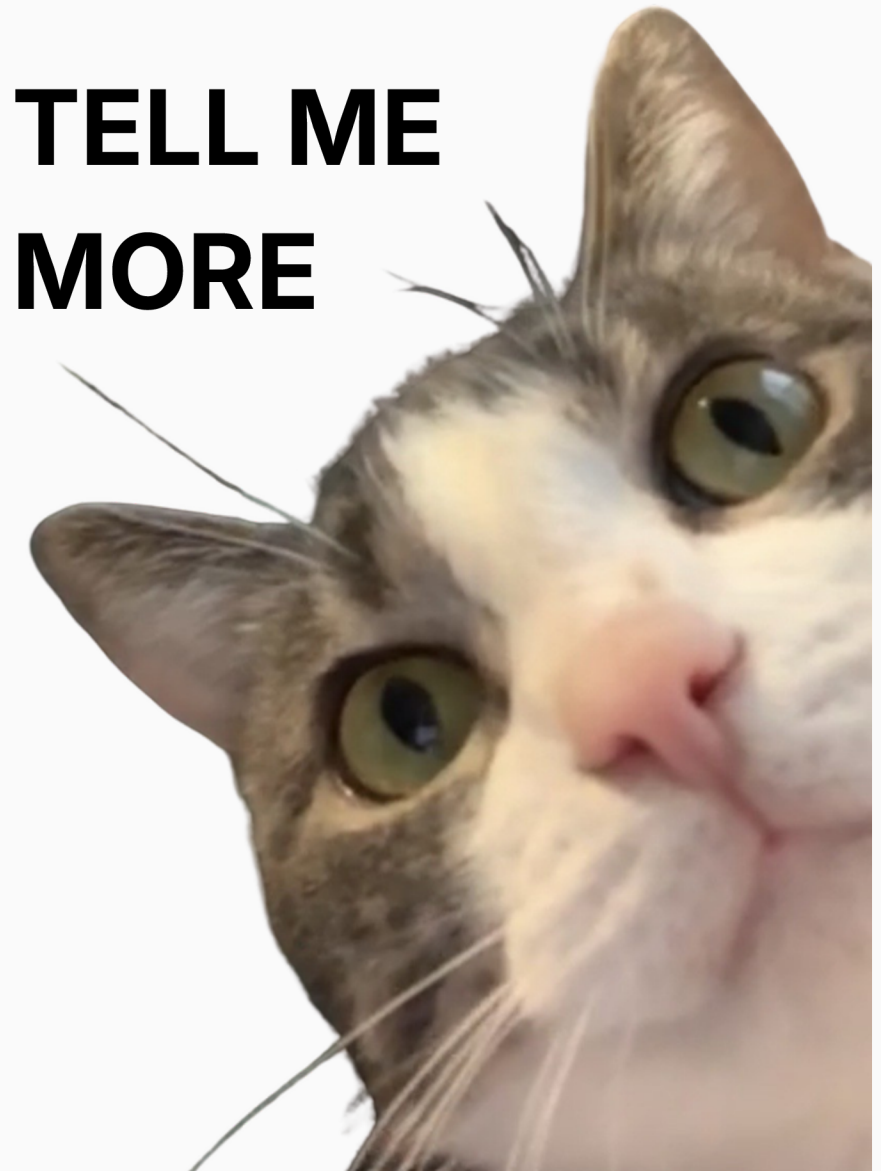
**R is vectorised by design** - operations apply to entire vectors at once. Functions will operate on all elements of a vector without the needing to loop through them

**R defaults to single-threaded execution** - parallelism can be possible with certain packages

**Indexing starts at 1, not 0** -  $x[1]$  is the first element. Different from most programming languages

**How interesting...**

**TELL ME  
MORE**



# Open R

---

Linux

Ctrl

+

Alt

+

T

Type “R” → Enter

macOS

⌘

+

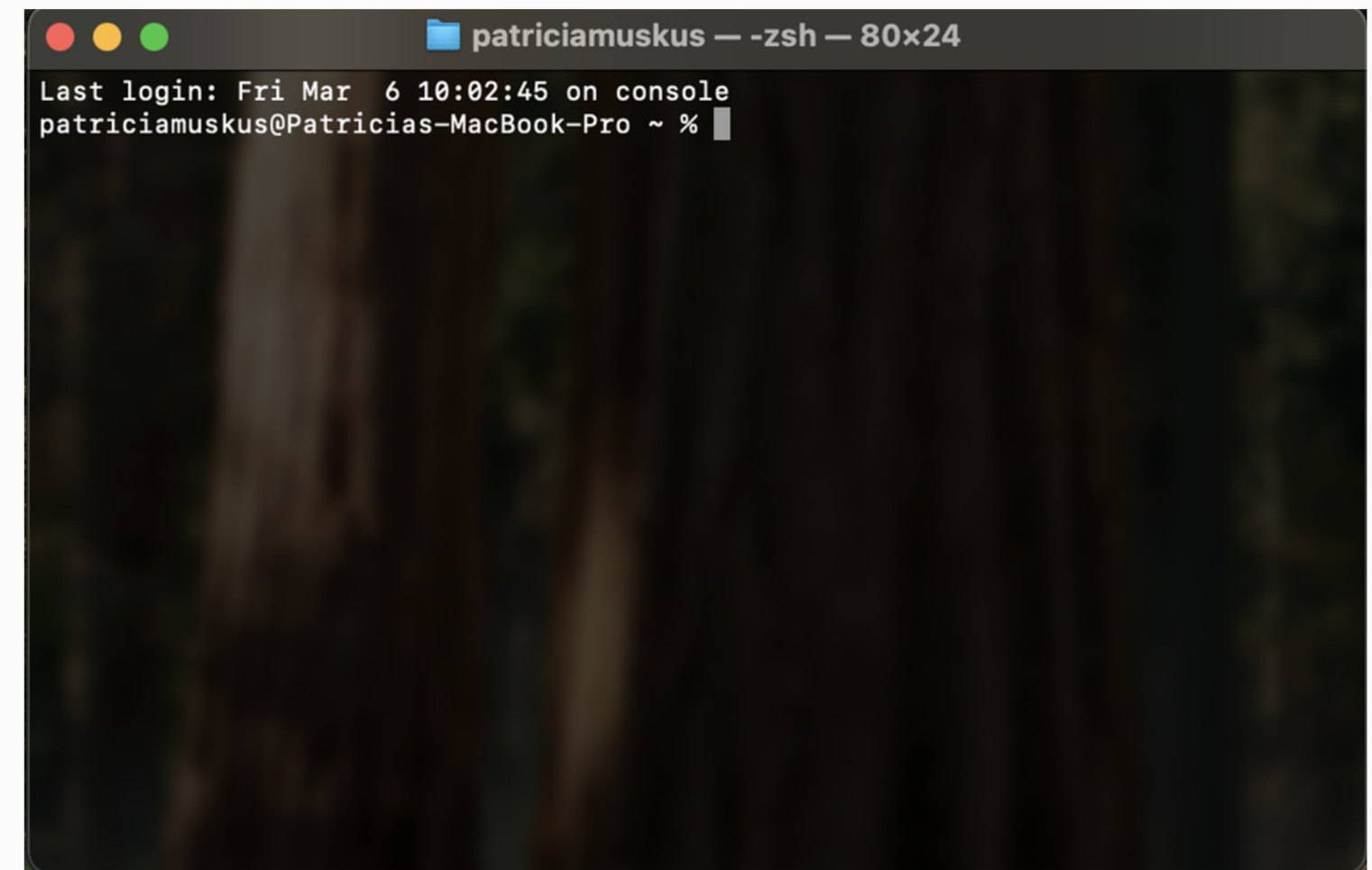
Space

Type “Terminal” → Enter

Type “R” → Enter



A terminal window on a Linux system. The title bar shows the user 'patricia-muskus' and the host 'patricia-mus...'. The terminal prompt is 'patricia-muskus@patricia-muskus-Latitude-3340:~\$' with a pink cursor. The window includes standard Linux window controls (minimize, maximize, close) and a search icon.



A terminal window on a macOS system. The title bar shows the user 'patriciamuskus' and the window title '-zsh - 80x24'. The terminal output shows the last login time: 'Last login: Fri Mar 6 10:02:45 on console' and the prompt 'patriciamuskus@Patricias-MacBook-Pro ~ %'. The window includes standard macOS window controls (red, yellow, green buttons).



# R's Core Syntax - Base R

## Basic Arithmetic

### Base R

```
⌈$> 2 + 5
[1] 7

⌈$> 3 * 6
[1] 18

⌈$> 25 / 5
[1] 5

⌈$> sqrt(25)
[1] 5

⌈$> log10(1)
[1] 0

⌈$> 4 ^ 2
[1] 16
```

## Objects & Assignments

```
⌈$> x <- 9

⌈$> y <- 10

⌈$> z <- x + y

⌈$> x
[1] 9

⌈$> y
[1] 10

⌈$> z
[1] 19
```

Prefer `<-` over `=` for assignment. While `X = 9` is valid, `=` is most seen for passing named arguments to functions. Mixing the two creates confusion



# Data Types

## Numeric

```
r$> expression_level <- 23.4
```

## Character

```
r$> gene <- "BRCA1"
```

## Logical

```
r$> brca1_variant <- TRUE  
r$> brca2_variant <- FALSE
```

## Checking data types

```
r$> class(expression_level)  
[1] "numeric"  
  
r$> class(gene)  
[1] "character"  
  
r$> class(brca1_variant)  
[1] "logical"  
  
r$> class(brca2_variant)  
[1] "logical"
```

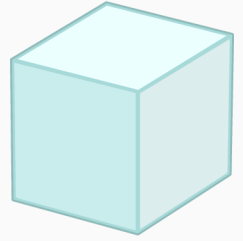
## Change between data types

```
r$> as.character(expression_level)  
[1] "23.4"  
  
r$> class(expression_level)  
[1] "numeric"
```

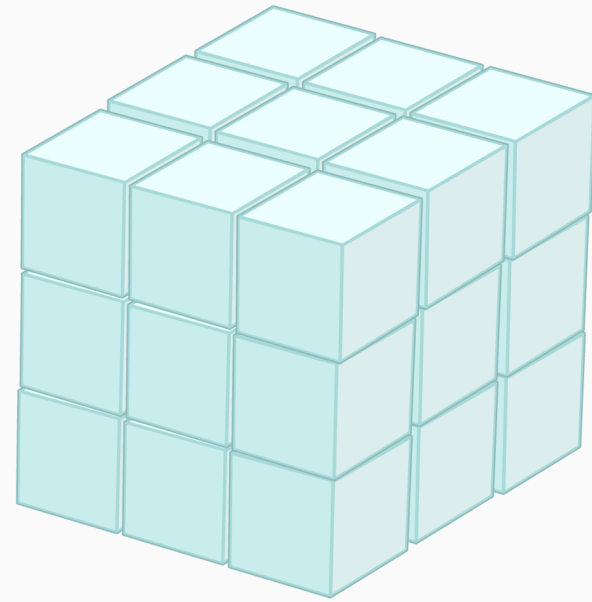


# How is data stored in R?

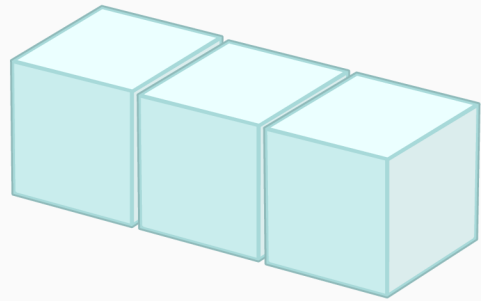
Scalar



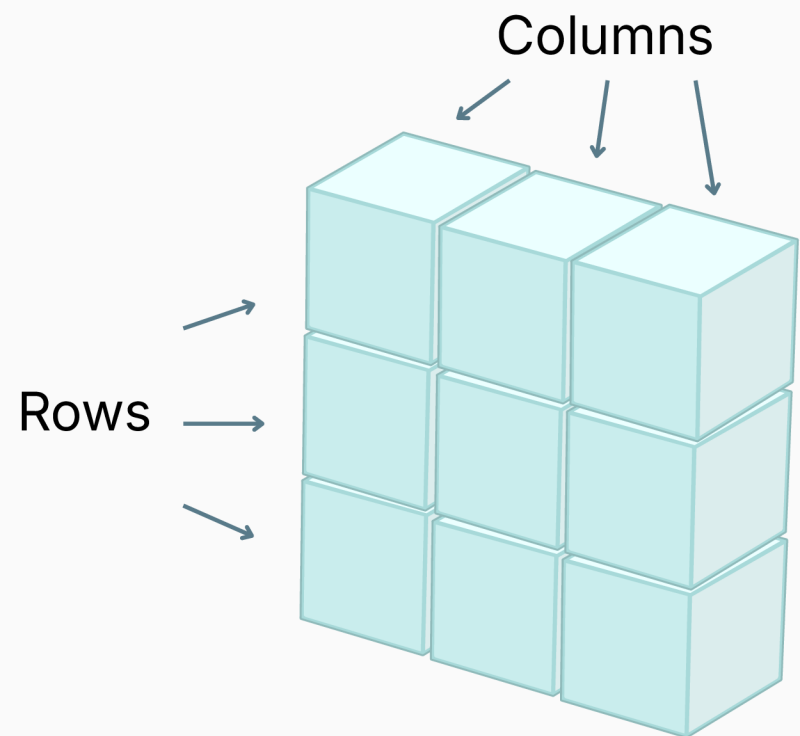
Array



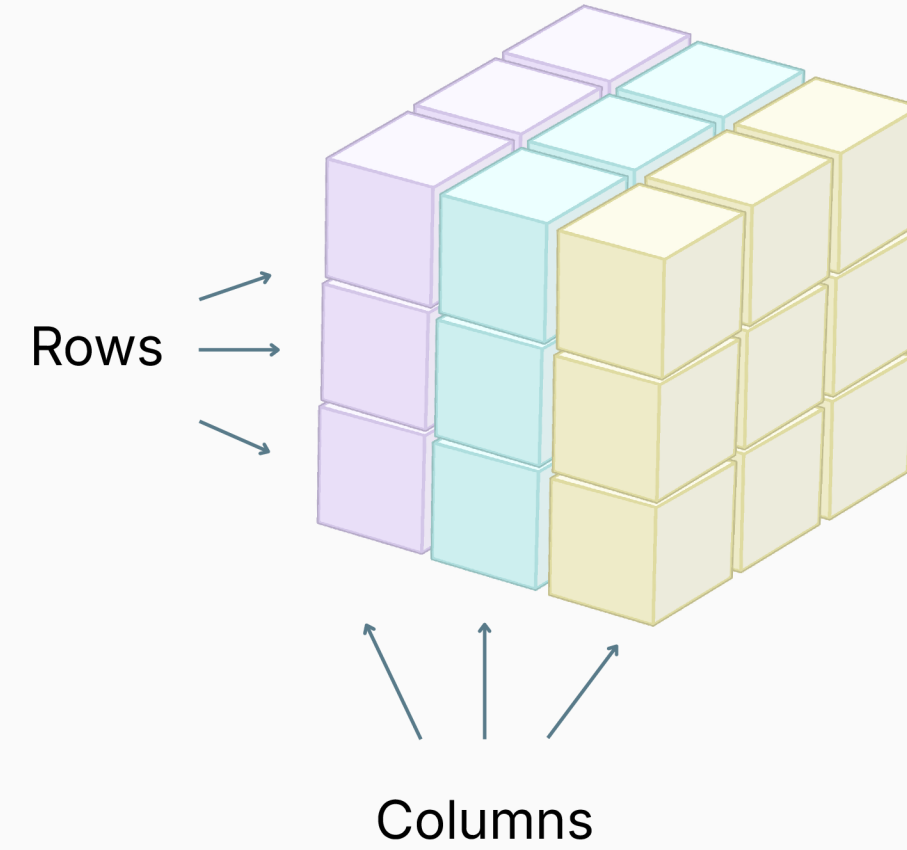
Vector



Matrix



Data Frame



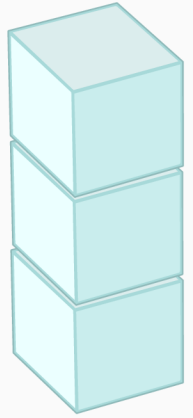
**MIXED DATA TYPES**

**SAME DATA TYPE**



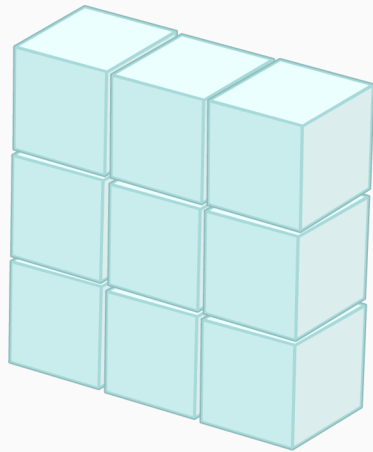
# How is data stored in R?

## Vector



```
r$> genes <- c("BRCA1", "ESR1", "TFF1", "PDZK1")
```

## Matrix



```
r$> genes <- c("BRCA1", "ESR1", "TFF1", "PDZK1")
```

```
r$> ERR5873116 <- c(207, 168, 88, 5)
```

```
r$> ERR5873117 <- c(321, 125, 668, 5)
```

```
r$> ERR5873118 <- c(354, 82, 244, 0)
```

## Example

```
r$> genes <- c("BRCA1", "ESR1", "TFF1", "PDZK1")
```

```
r$> ERR5873116 <- c(207, 168, 88, 5)
```

```
r$> ERR5873117 <- c(321, 125, 668, 5)
```

```
r$> ERR5873118 <- c(354, 82, 244, 0)
```

```
r$> count_matrix <- cbind(ERR5873116, ERR5873117,  
ERR5873118)
```

```
r$> rownames(count_matrix) <- genes
```

```
r$> count_matrix
```

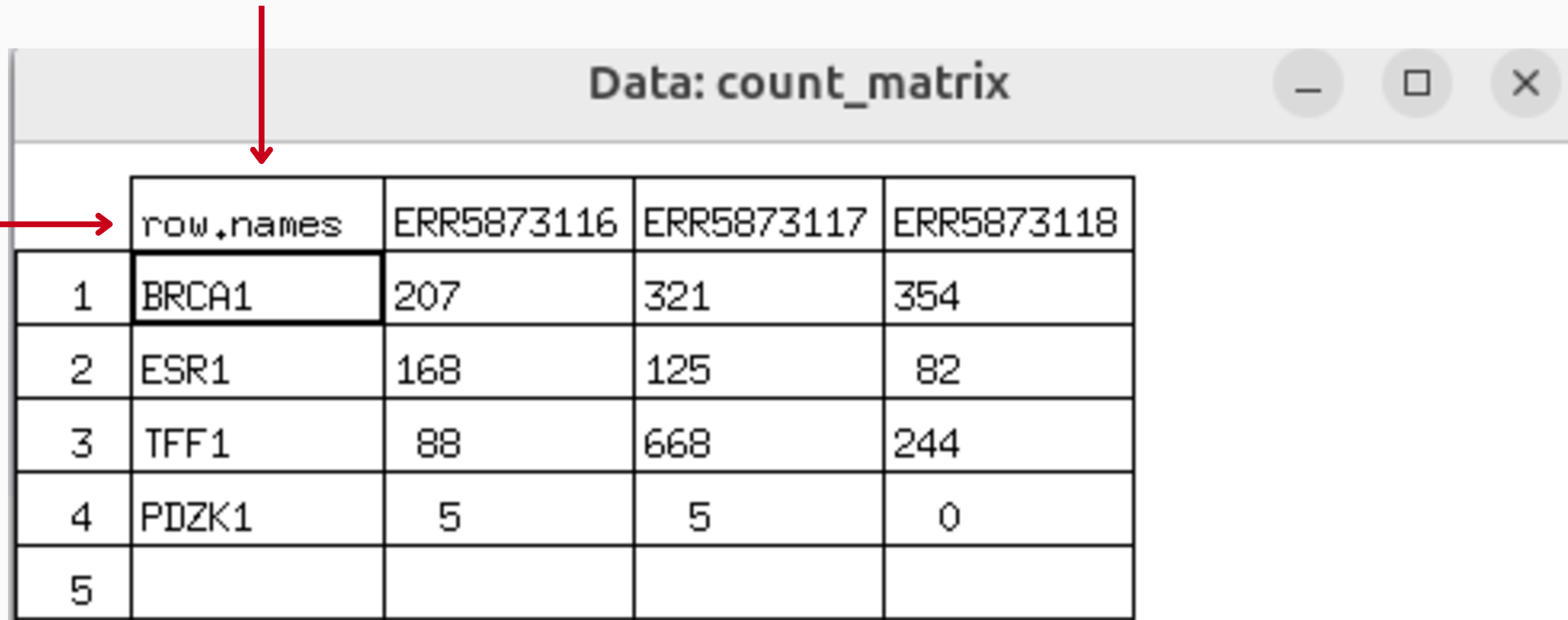
	ERR5873116	ERR5873117	ERR5873118
BRCA1	207	321	354
ESR1	168	125	82
TFF1	88	668	244
PDZK1	5	5	0



# How is data stored in R?

**Rownames**

**Colnames**



	row.names	ERR5873116	ERR5873117	ERR5873118
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0
5				

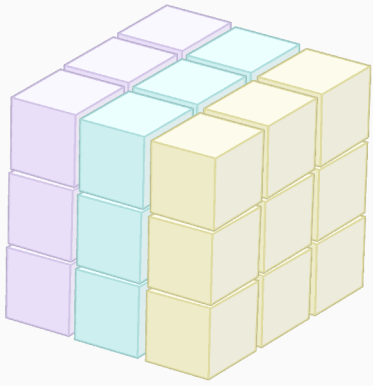
```
r$> rownames(count_matrix)
[1] "BRCA1" "ESR1" "TFF1" "PDZK1"

r$> colnames(count_matrix)
[1] "ERR5873116" "ERR5873117" "ERR5873118"
```



# How is data stored in R?

## Data Frame



## Example

**Rownames**

**Colnames**

	Run	Condition	Organism	Sample
1	ERR5873116	WT	mus	pancreas
2	ERR5873117	WT	mus	pancreas
3	ERR5873120	KO	mus	pancreas
4	ERR5873121	KO	mus	pancreas
5				

```
r$> run <- c("ERR5873116", "ERR5873117", "ERR5873120", "ERR5873121")
r$> condition <- c("WT", "WT", "KO", "KO")
r$> organism <- c("mus", "mus", "mus", "mus")
r$> sample <- c("pancreas", "pancreas", "pancreas", "pancreas")
r$> metadata <- data.frame("Run" = run, "Condition" = condition, "Organism" = organism, "Sample" = sample)
r$> View(metadata)
```

```
r$> rownames(metadata)
[1] "1" "2" "3" "4"

r$> colnames(metadata)
[1] "Run" "Condition" "Organism" "Sample"
```

```
r$> class(metadata)
[1] "data.frame"
```



# Try it yourself!

---

	Genes	p.val	Log2FC
1	Gnai	0.1066	0.4
2	H19	0.1967	1.3
3	Cox5a	0.0120	4.7
4	Ngrf	0.0098	-1.4
5	Axin2	0.0000	-0.7

1. Create “deg\_results” data frame with columns “Genes”, “p.val”, and “log2fc”
2. Confirm data structure
3. Confirm row & column names



# Try it yourself!

	Genes	p.val	Log2FC
1	Gnai	0.1066	0.4
2	H19	0.1967	1.3
3	Cox5a	0.0120	4.7
4	Ngrf	0.0098	-1.4
5	Axin2	0.0000	-0.7

What's wrong here?



```
r$> genes <- c("Gnai", "H19", "Cox5a", "Ngrf", "Axin2")
r$> p.val <- c(0.1066, 0.1967, 0.0120, 0.0098, 0.0000)
r$> log2fc <- c(0.4, 1.3, 4.7, -1.4, -0.7)
r$> deg_results <- as.data.frame("Genes"=genes, "p.val"=p.val, "Log2FC"=log2fc)
Error in as.data.frame(Genes = genes, p.val = p.val, Log2FC = log2fc) :
  argument "x" is missing, with no default
```



# Try it yourself!

	Genes	p.val	Log2FC
1	Gnai	0.1066	0.4
2	H19	0.1967	1.3
3	Cox5a	0.0120	4.7
4	Ngrf	0.0098	-1.4
5	Axin2	0.0000	-0.7

`as.data.frame` is used to convert between data structures

```
r$> deg_results <- data.frame(  
  "Genes"=genes, "p.val"=p.val,  
  "Log2FC"=log2fc)
```

```
r$> deg_results  
  Genes  p.val  Log2FC  
1  Gnai  0.1066   0.4  
2   H19  0.1967   1.3  
3 Cox5a  0.0120   4.7  
4  Ngrf  0.0098  -1.4  
5 Axin2  0.0000  -0.7
```



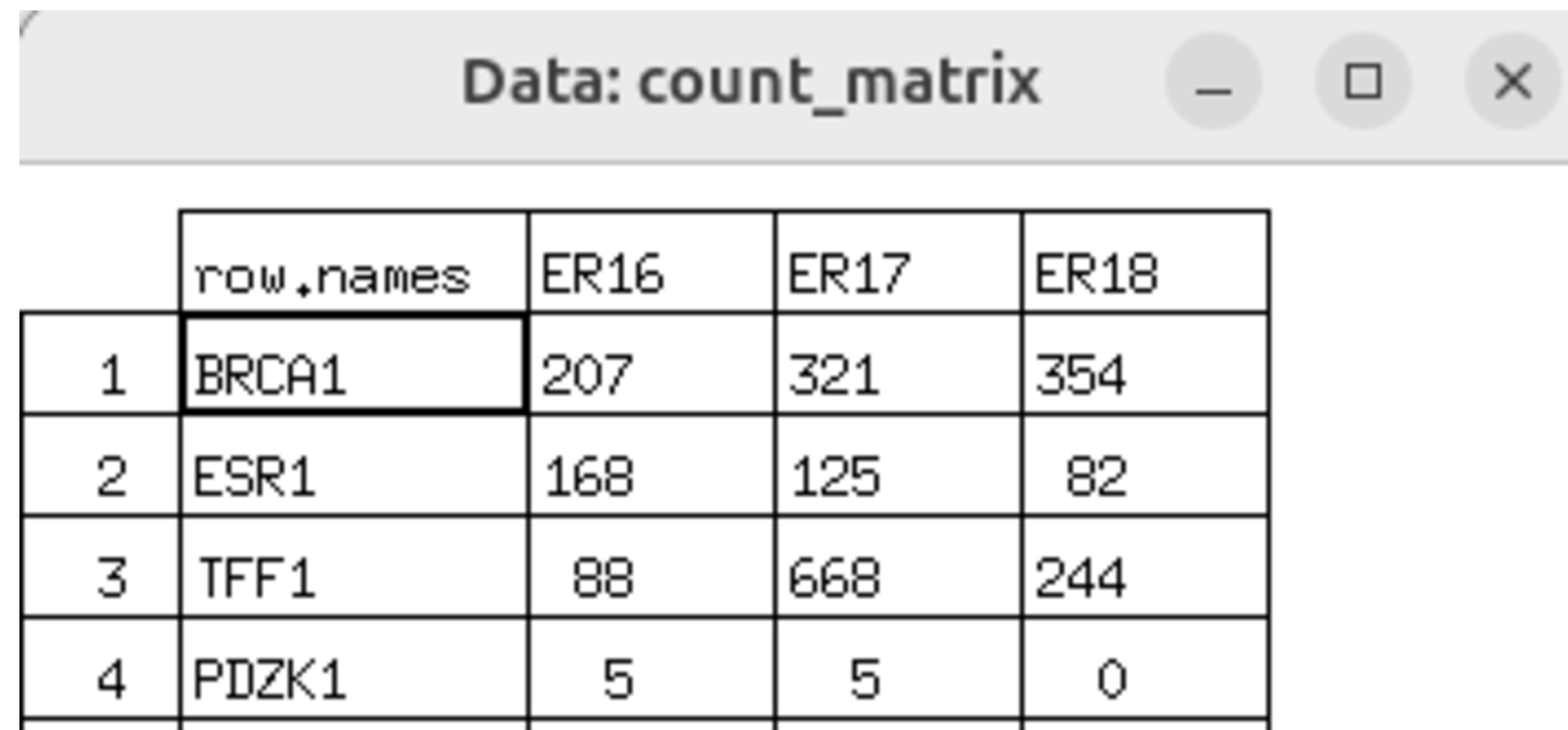
# Converting Between Data Structures

```
r$> class(count_matrix)
[1] "matrix" "array"

r$> class(metadata)
[1] "data.frame"

r$> count_matrix <- as.data.frame(count_matrix)

r$> class(count_matrix)
[1] "data.frame"
```



The screenshot shows a window titled "Data: count\_matrix" with a table of data. The table has four rows and four columns. The first column contains row numbers 1 through 4. The second column contains row names: BRCA1, ESR1, TFF1, and PDZK1. The third, fourth, and fifth columns contain numerical values for ER16, ER17, and ER18 respectively.

	row.names	ER16	ER17	ER18
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0

Because matrices and data frames support “rownames”, R will preserve the row names during the transition of matrix to data frame.



**Quick Break**



**Back in 10**

## How R Stores Data

Vector

Matrix

Array

Data Frame

Now what?

Data Wrangling



# R Programming Concepts

Data: metadata

	Run	Condition	Organism	Sample
1	ERR5873116	WT	mus	pancreas
2	ERR5873117	WT	mus	pancreas
3	ERR5873120	KO	mus	pancreas
4	ERR5873121	KO	mus	pancreas
5				

## data.frame\$column

```
r$> class(metadata)
[1] "data.frame"

r$> metadata$Condition
[1] "WT" "WT" "KO" "KO"

r$> metadata$Run
[1] "ERR5873116" "ERR5873117" "ERR5873120"
[4] "ERR5873121"
```

## data.frame[row,column]

```
r$> metadata[, "Condition"]
[1] "WT" "WT" "KO" "KO"

r$> metadata[, 3]
[1] "mus" "mus" "mus" "mus"
```

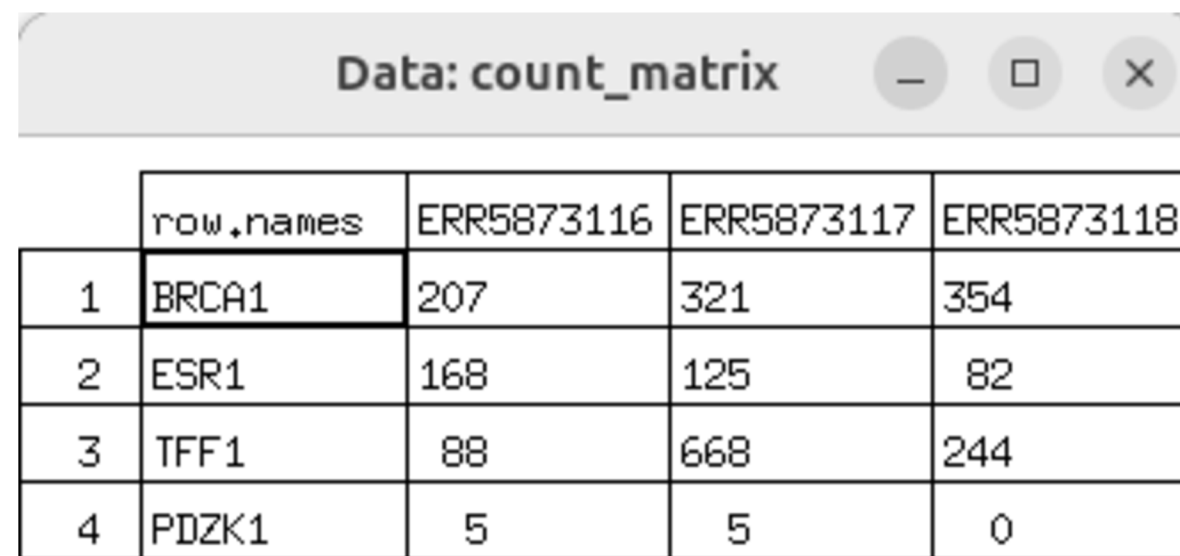
## data.frame[c(rows),c(cols)]

```
r$> metadata[c(1,3), c("Run", "Condition")]
      Run Condition
1 ERR5873116      WT
3 ERR5873120      KO
```



# R Programming Concepts

**R uses 1 based indexing, meaning the first element is always at position 1, not 0**



	row.names	ERR5873116	ERR5873117	ERR5873118
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0

**data.frame[row:index,col:index]**

```
r$> count_matrix[1:3,2:3]
      ERR5873117 ERR5873118
BRCA1         321         354
ESR1          125          82
TFF1          668         244

r$> count_matrix[2,1:3]
      ERR5873116 ERR5873117 ERR5873118
ESR1          168          125          82
```

**Elements can be omitted by using negative indexing**

```
r$> count_matrix[-c(2,3),]
      ERR5873116 ERR5873117 ERR5873118
BRCA1         207         321         354
PDZK1           5           5           0
```



# R Programming Concepts

Data: count\_matrix

	row.names	ERR5873116	ERR5873117	ERR5873118
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0

**subset(data.frame, select=condition)**

```
r$> subset(count_matrix,select = ERR5873116)
      ERR5873116
BRCA1         207
ESR1          168
TFF1           88
PDZK1           5
```

**Better readability  
but often less common**



```
r$> subset(count_matrix,subset = rownames(count_matrix)
  == "BRCA1",select = ERR5873116)
      ERR5873116
BRCA1         207
```

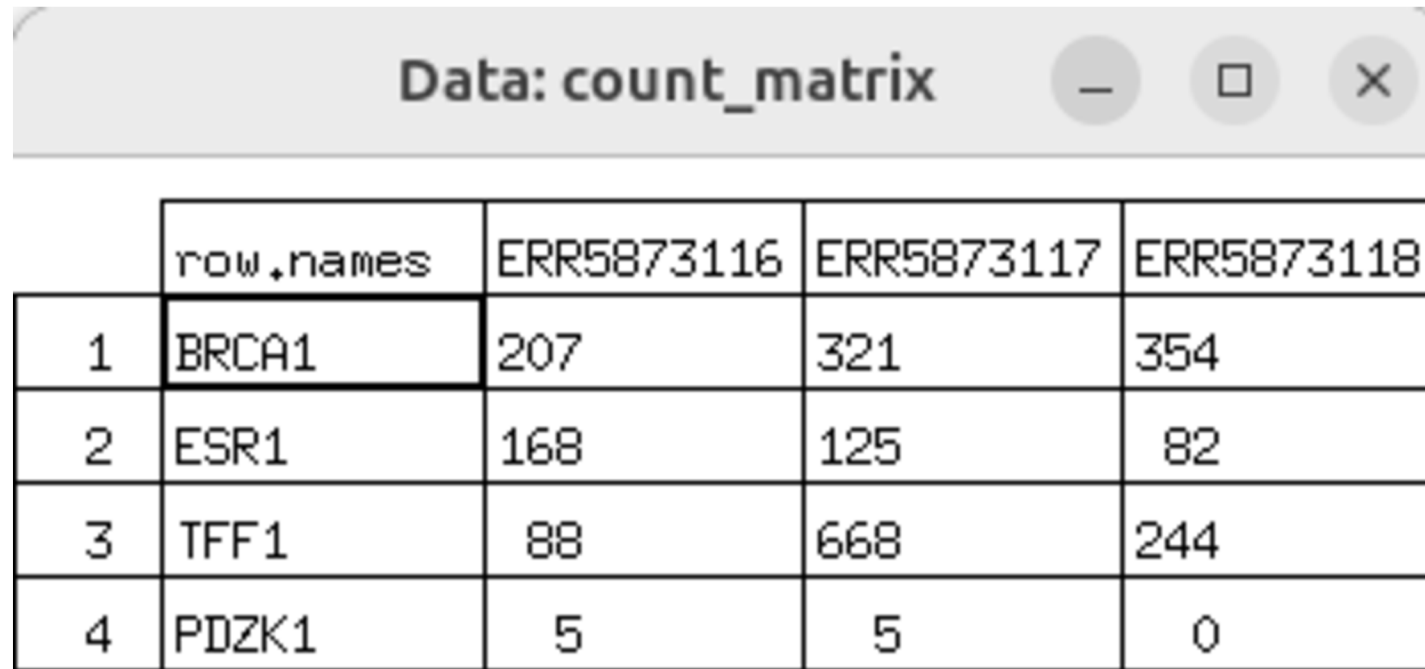
**Bracket indexing is more  
common, but often less  
readable**



```
r$> count_matrix[1,1,drop = FALSE]
      ERR5873116
BRCA1         207
```



# Try it yourself!



The screenshot shows a R console window titled "Data: count\_matrix". It displays a data frame with 4 rows and 4 columns. The first column contains row indices (1-4), the second column contains gene names (BRCA1, ESR1, TFF1, PDZK1), and the next three columns contain counts for three different samples (ERR5873116, ERR5873117, ERR5873118).

	row.names	ERR5873116	ERR5873117	ERR5873118
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0

Because matrices and data frames support “rownames”, R will preserve the row names during the transition of matrix to data frame.

1. Add a new column “genes” to data frame by copying row.names to a column
2. Clean the data frame by removing row.names from data frame

**Hint:** “\$” is used to access and create columns in data frames



# Try it yourself!

Data: count_matrix				
	row.names	ERR5873116	ERR5873117	ERR5873118
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0

Because matrices and data frames support “rownames”, R will preserve the row names during the transition of matrix to data frame.

```
r$> count_matrix$genes <- row  
names(count_matrix)
```

```
r$> count_matrix  
      ER16 ER17 ER18 genes  
BRCA1  207  321  354 BRCA1  
ESR1   168  125   82  ESR1  
TFF1   88  668  244  TFF1  
PDZK1   5   5    0  PDZK1
```

```
r$> rownames(count_matrix) <-  
NULL
```

```
r$> count_matrix  
      ER16 ER17 ER18 genes  
1  207  321  354 BRCA1  
2  168  125   82  ESR1  
3   88  668  244  TFF1  
4    5    5    0  PDZK1
```



# Changing Column Order

[1] [2] [3] [4]

Data: count_matrix				
	row.names	ERR5873116	ERR5873117	ERR5873118
1	BRCA1	207	321	354
2	ESR1	168	125	82
3	TFF1	88	668	244
4	PDZK1	5	5	0

```
r$> count_matrix
  ER16 ER17 ER18 genes
1  207  321  354 BRCA1
2  168  125   82  ESR1
3   88  668  244  TFF1
4    5    5    0 PDZK1
```

`data.frame[row, c(index)]`

"genes"  
at index 1  
now

```
r$> count_matrix <- count_matrix[,c(4,1,2,3)]
r$> count_matrix
  genes ER16 ER17 ER18
1 BRCA1  207  321  354
2  ESR1  168  125   82
3  TFF1   88  668  244
4 PDZK1    5    5    0
```

# R Programming Concepts

---

## Subsetting with Logical Operators

**>** greater than

**<** less than

**>=** greater and equal to

**<=** less and equal to

**==** exactly equal to

**!=** not equal to

```
r$> ctrl_group <- metadata[metadata$Condition=="WT",]  
  
r$> ctrl_group  
      Run Condition Organism  Sample  
1 ERR5873116      WT      mus pancreas  
2 ERR5873117      WT      mus pancreas
```



# R Programming Concepts

## Subsetting with Logical Operators - More Examples!

	Genes	p.val	Log2FC
1	Gnai	0.1066	0.4
2	H19	0.1967	1.3
3	Cox5a	0.0120	4.7
4	Ngrf	0.0098	-1.4
5	Axin2	0.0000	-0.7

```
r$> deg_sig <- deg_results[deg_results$p.val <= 0.05,]
```

```
r$> deg_sig
  Genes p.val Log2FC
3 Cox5a 0.0120  4.7
4  Ngrf 0.0098 -1.4
5 Axin2 0.0000 -0.7
```

```
r$> deg_sig <- deg_results[deg_results$p.val <= 0.05 & deg_results$Log2FC >1,]
```

```
r$> deg_sig
  Genes p.val Log2FC
3 Cox5a 0.012  4.7
```



# Subsetting with Logical Operators - More Examples!

subset rows

subset cols

```
ctrl_group<-metadata[metadata$Condition=="control"& metadata$Reads_byMillion > 20,]  
ctrl_group  
Sample_ID Condition Batch Reads_byMillion  
CTRL1 control a 24.5
```

```
> significant <- metadata[metadata$Condition == "control" & metadata$p.val <= 0.05,]  
> significant  
Sample_ID Condition Batch Reads_byMillion p.val  
2 CTRL2 control a 19.8 0.01  
1 CTRL1 control a 24.5 0.03
```



# What to Expect?

## Day 2

- RStudio Orientation
- R's file system organization
  - Folder structure & naming conventions
- Packages & libraries
- Load & read different file types
- Quick introduction to R and RStudio in Deigo



# Quick Note on .RData

---

**.RData files-** These files are specific to R, they contain any objects created during the current session. If not named, this file is a **hidden** file

On this note... “workspace image” is interpreted as save “everything” on the current session under a “.RData” file.

To save one or more specific objects

**save(object 1, object 2, object 5,  
file= “myObjects.RData”)**

To save all objects (“workspace image”)

**save.image( file= “introImage.RData”)**

```
> q()  
Save workspace image? [y/n/c]:
```

```
> count_matrix  
      ER16 ER17 ER18  
BRCA1  207  321  354  
ESR1   168  125   82  
TFF1   88   668  244  
PDZK1   5    5    0  
> save.image("introR.RData")  
> q()
```



# Quick Note on .RData

To load the “.RData” file

**load(“introImage.RData”)**



```
[Previously saved workspace restored]
> load("introR.RData")
> ls()
 [1] "brca1_variant"      "brca2_variant"      "condition"           "count_matrix"
 [5] "deg_results"       "ER16"                "ER17"                "ER18"
 [9] "expression_level"  "gene"                "genes"               "log2fc"
[13] "metadata"          "organism"            "p.val"                "run"
[17] "sample"            "x"                   "y"                   "z"
```

More commands...

**ls()** lists all objects in your workspace  
**rm(object1)** will remove the object specified

By default, R will load the previously saved workspace image.

```
> rm(brca1_variant)
> ls()
 [1] "brca2_variant"      "condition"           "count_matrix"       "deg_results"
 [5] "ER16"               "ER17"                "ER18"                "expression_level"
 [9] "gene"               "genes"               "log2fc"              "metadata"
[13] "organism"           "p.val"                "run"                  "sample"
[17] "x"                   "y"                   "z"
```



# From Terminal to RStudio

```
patricia-muskus@patricia-muskus-Latitude-3340: ~  
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.  
  
[Previously saved workspace restored]  
  
> genes <- c("BRCA1", "ESR1", "TFF1", "PDZK1")  
> ER16 <-c(207,168,88,5)  
> ER17 <-c(321,125,668,5)  
> ER18 <-c(354,82,244,0)  
> count_matrix <-cbind(ER16,ER17,ER18)  
> rownames(count_matrix)<-genes  
> count_matrix  
      ER16 ER17 ER18  
BRCA1 207  321  354  
ESR1   168  125   82  
TFF1   88  668  244  
PDZK1   5    5    0  
>
```



The RStudio interface is shown with the following components:

- Source Editor:** Contains the R script code from the terminal window.
- Environment:** Displays the objects created in the script, including the `count_matrix` data frame and the `genes` character vector.
- Console:** Shows the execution output of the script, including the printed `count_matrix` data frame.

Object	Class	Dimensions	Values
count_matrix	num	[1:4, 1:3]	207 168 88 5 321 125 668...
ER16	num	[1:4]	207 168 88 5
ER17	num	[1:4]	321 125 668 5
ER18	num	[1:4]	354 82 244 0
genes	chr	[1:4]	"BRCA1" "ESR1" "TFF1" "PDZK1"

Object	Class	Dimensions	Values
count_matrix	num	[1:4, 1:3]	ER16 ER17 ER18 BRCA1 207 321 354 ESR1 168 125 82 TFF1 88 668 244 PDZK1 5 5 0



# RStudio Working Directory

**Working Directory** - A path in your computer that specifies the location where R will find and save all of your files.

When R loads the previously saved work image, it is loading whichever “.RData” file it finds under the current working directory

If not specified, this tends to be your “/home” directory →

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for creating a count matrix from gene names and expression values.
- Environment:** Shows the current R environment with variables like 'count\_matrix' and 'genes'.
- Console:** Shows the output of the `getwd()` command, indicating the current working directory is `"/home/patricia-muskus"`.
- Plots, Files, Packages:** The bottom right pane, currently empty.

```
1
2 genes <- c("BRCA1", "ESR1", "TFF1", "PDZK1")
3 ER16 <- c(207, 168, 88, 5)
4 ER17 <- c(321, 125, 668, 5)
5 ER18 <- c(354, 82, 244, 0)
6 count_matrix <- cbind(ER16, ER17, ER18)
7 rownames(count_matrix) <- genes
8 count_matrix
9
```

Data	
count_matrix	num [1:4, 1:3] 207 168 88 5 321 125 668...
Values	
ER16	num [1:4] 207 168 88 5
ER17	num [1:4] 321 125 668 5
ER18	num [1:4] 354 82 244 0
genes	chr [1:4] "BRCA1" "ESR1" "TFF1" "PDZK1"

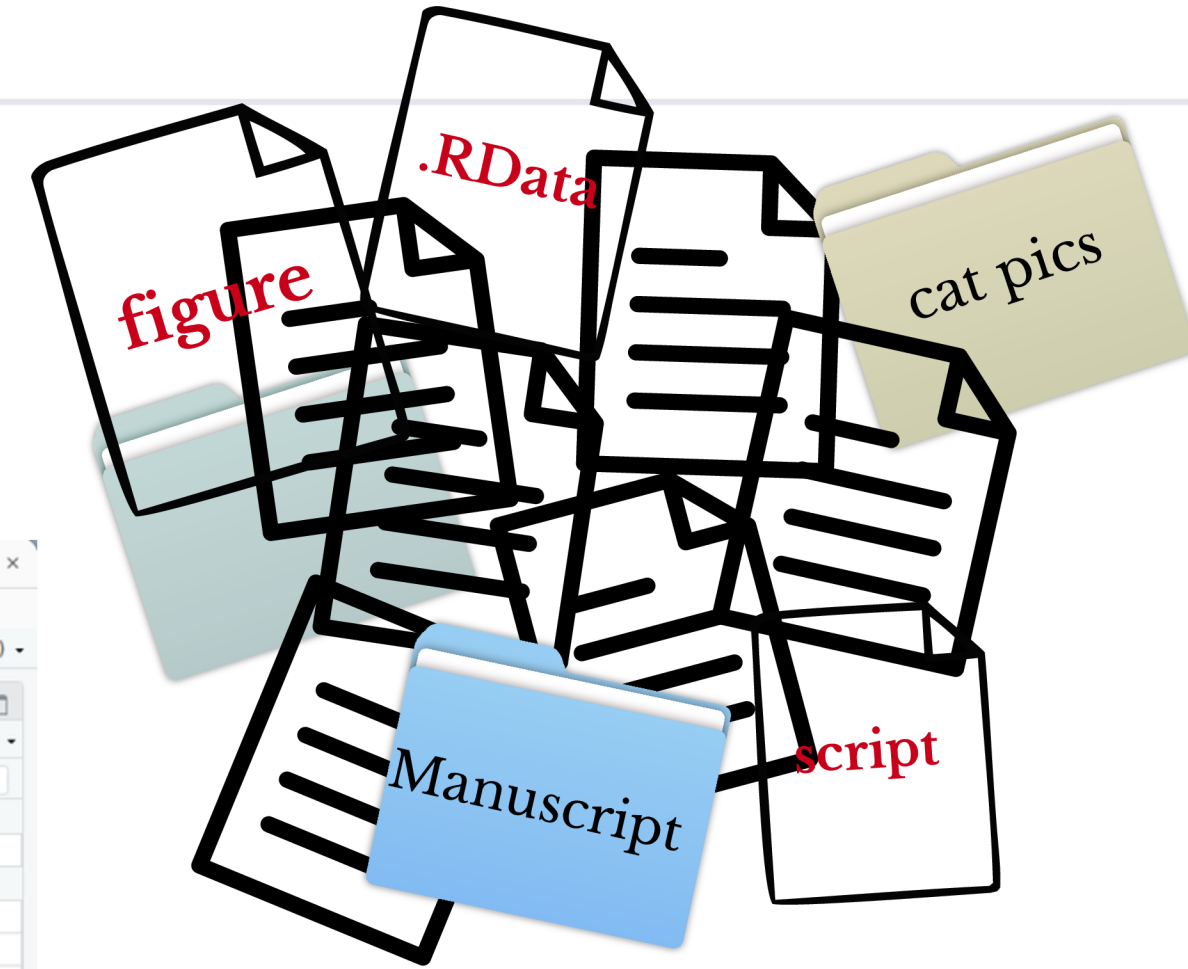
```
> getwd()
[1] "/home/patricia-muskus"
>
```



# RStudio Working Directory

Why is this a problem???

Let's take a look at the current working dir



The screenshot shows the RStudio interface with several panels. The Source Editor (top left) contains R code for creating a count matrix. The Environment (top right) shows the current R session with a 'Global Environment' and a 'Data' table. The Console (bottom left) shows the output of `getwd()` and `list.files()`. The bottom right panel shows the Files, Plots, Packages, Help, Viewer, and Presentation tabs. A red arrow points from the Console output to the 'R Project' folder in the diagram to the right.

**Source Editor**

```
1 genes <- c("BRCA1", "ESR1", "TFF1", "PDZK1")
2 ER16 <- c(207, 168, 88, 5)
3 ER17 <- c(321, 125, 668, 5)
4 ER18 <- c(354, 82, 244, 0)
5 count_matrix <- cbind(ER16, ER17, ER18)
6 rownames(count_matrix) <- genes
7 count_matrix
```

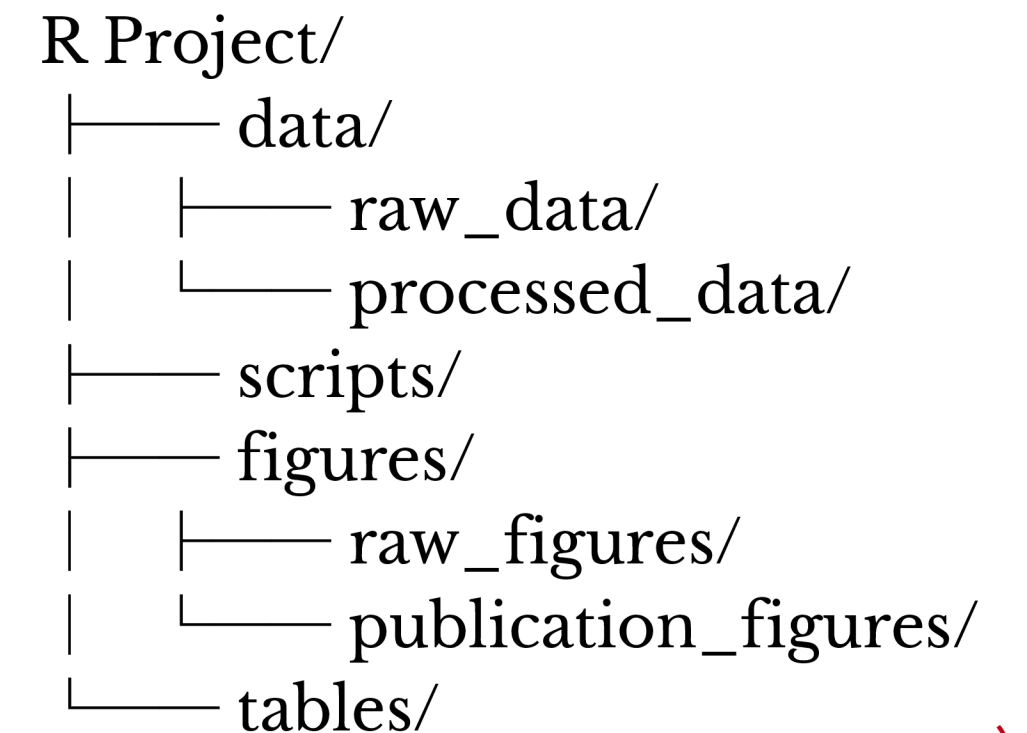
**Environment**

Data	count_matrix	num [1:4, 1:3]	207	168	88	5	321	125	668...
Values									
ER16	num [1:4]	207	168	88	5				
ER17	num [1:4]	321	125	668	5				
ER18	num [1:4]	354	82	244	0				
genes	chr [1:4]	"BRCA1"	"ESR1"	"TFF1"	"PDZK1"				

**Console**

```
> getwd()
[1] "/home/patricia-muskus"
> list.files()
 [1] "_site"
 [3] "05_population.slurm"
 [5] "\030CQQ"
 [7] "Documents"
 [9] "gems"
[11] "introR.RData"
[13] "Music"
[15] "population"
[17] "Public"
[19] "snap"
[21] "test.sh"
[23] "wget-log"
      "04_gstacks"
      "analysisObjects.RData"
      "Desktop"
      "Downloads"
      "gstacks.log"
      "JoplinBackup"
      "Pictures"
      "populations.log.distribs"
      "R"
      "Templates"
      "Videos"
```

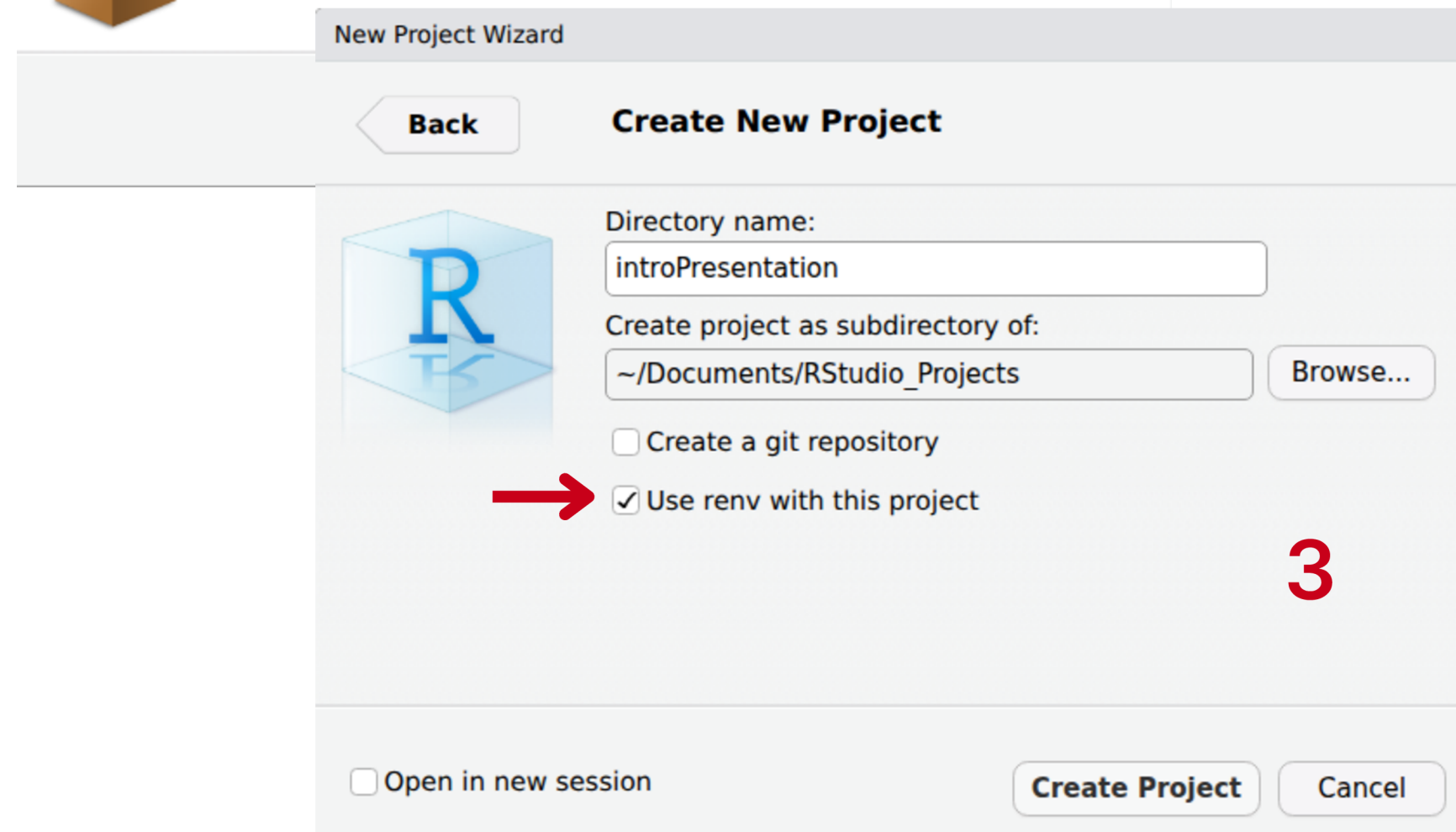
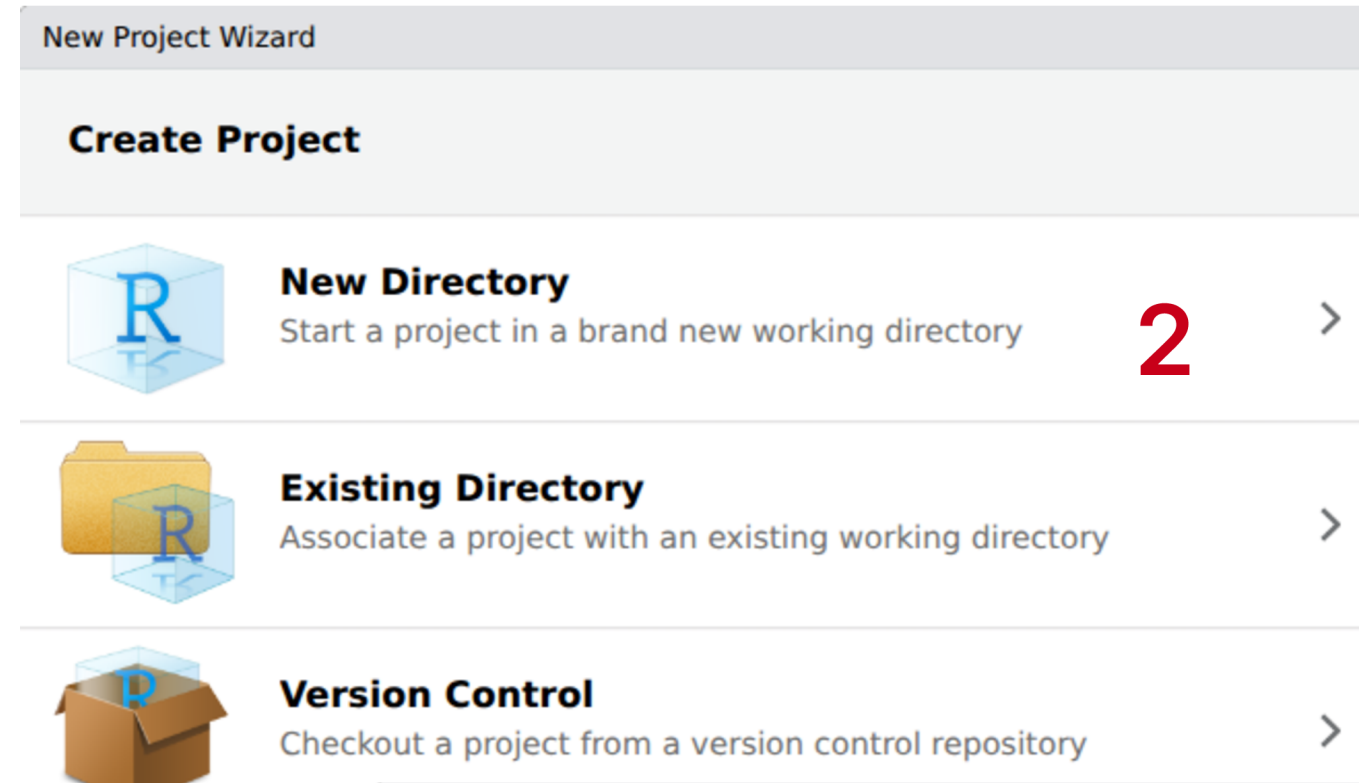
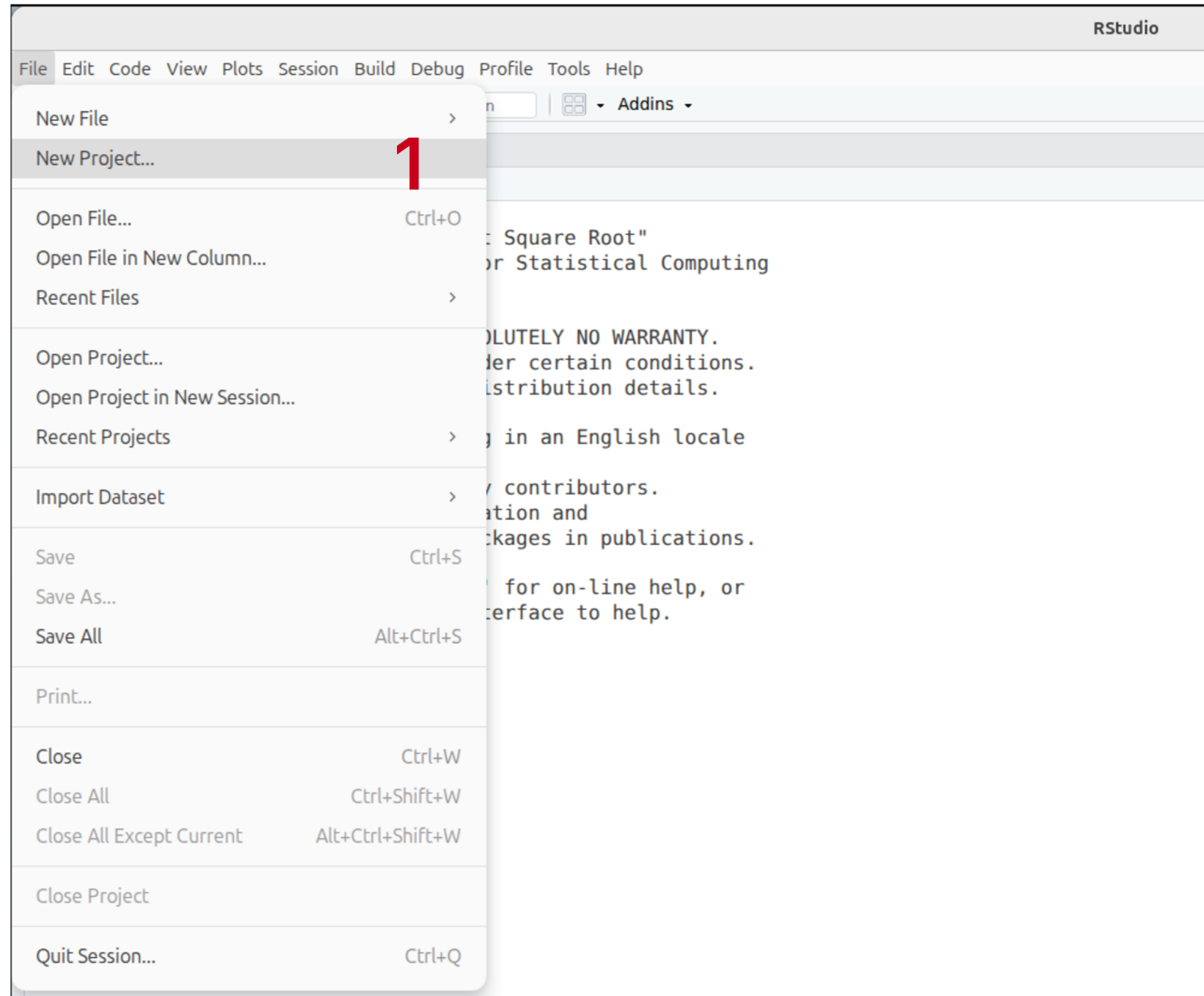
**Plots, Files, Packages**



Advise? Keep a project for each experiment



# RStudio Project



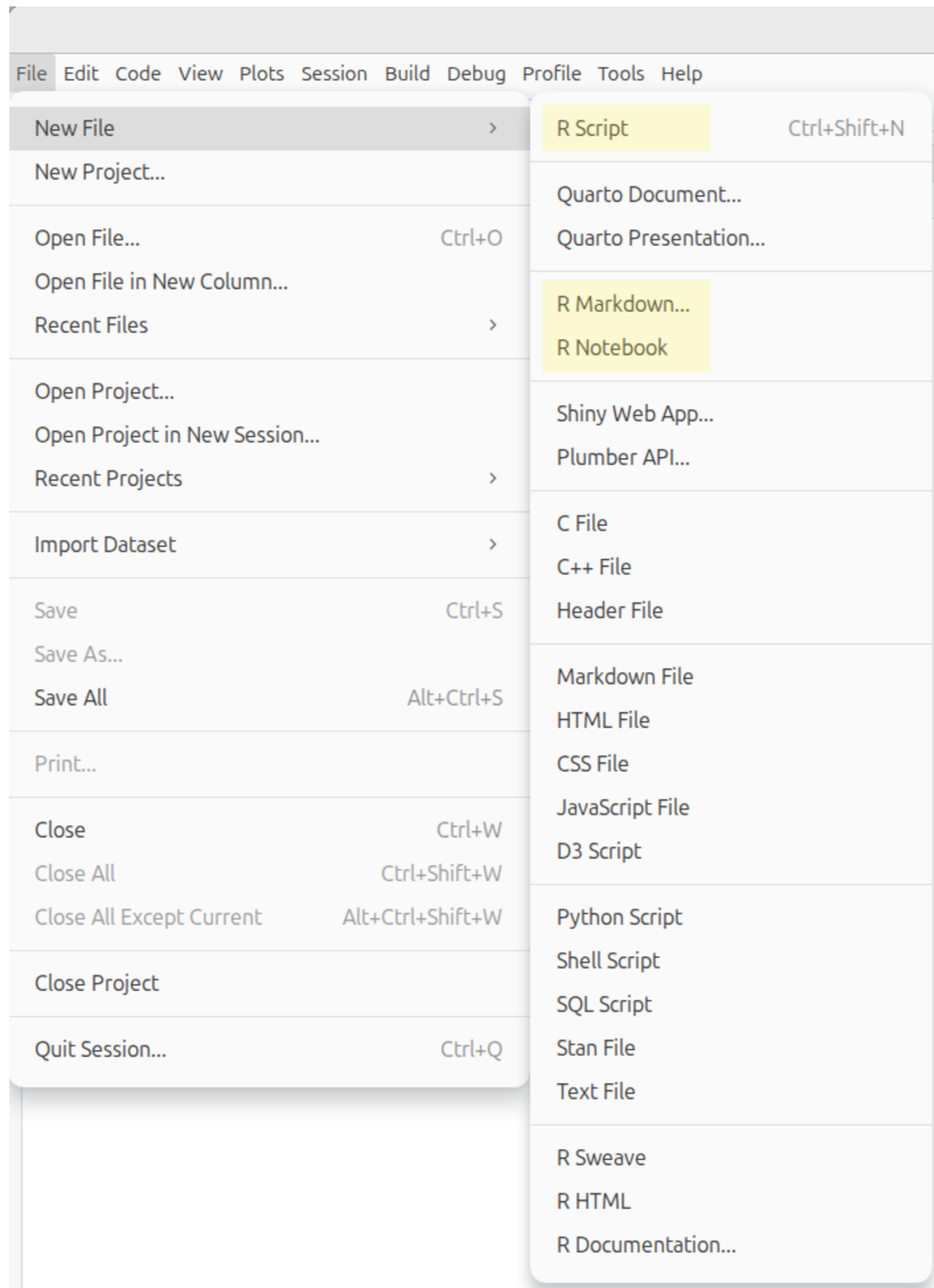
**Renv** - R package which saves all packages and libraries used in the current project.

learn more here:

<https://rstudio.github.io/renv/articles/renv.html>



# RStudio Files



**R Notebook-** execute code chunks interactively, sending **one line at a time** to the console, and automatically generating a preview when saved.

**Good for interactive development and exploratory analysis.**

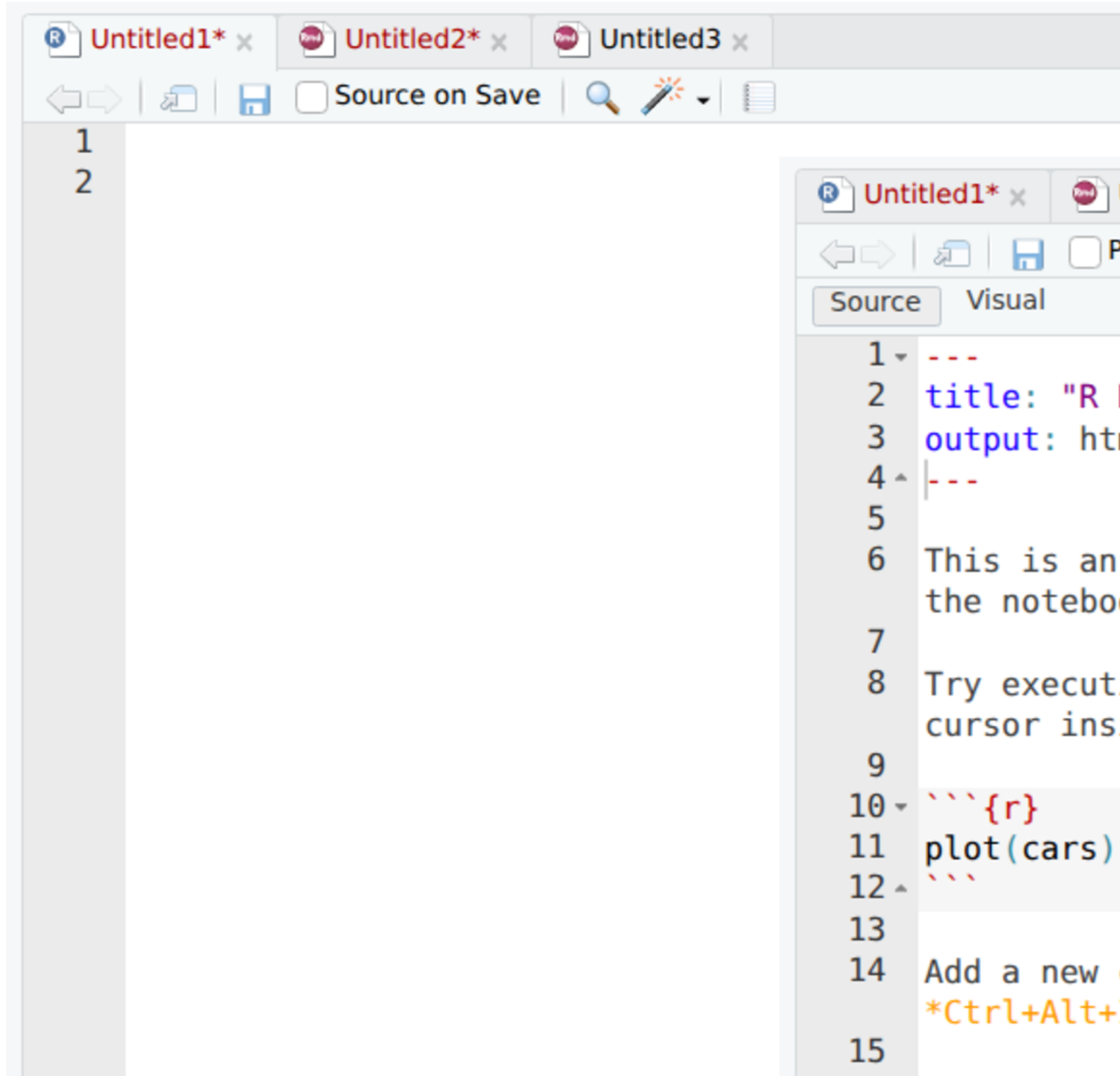
**R Markdown-** send all code to the console at once when executed.

More flexibility for "knit" or "rendering" options such as PDF, HTML, and Word documents.

**Good for creating documentation style render from your analysis**

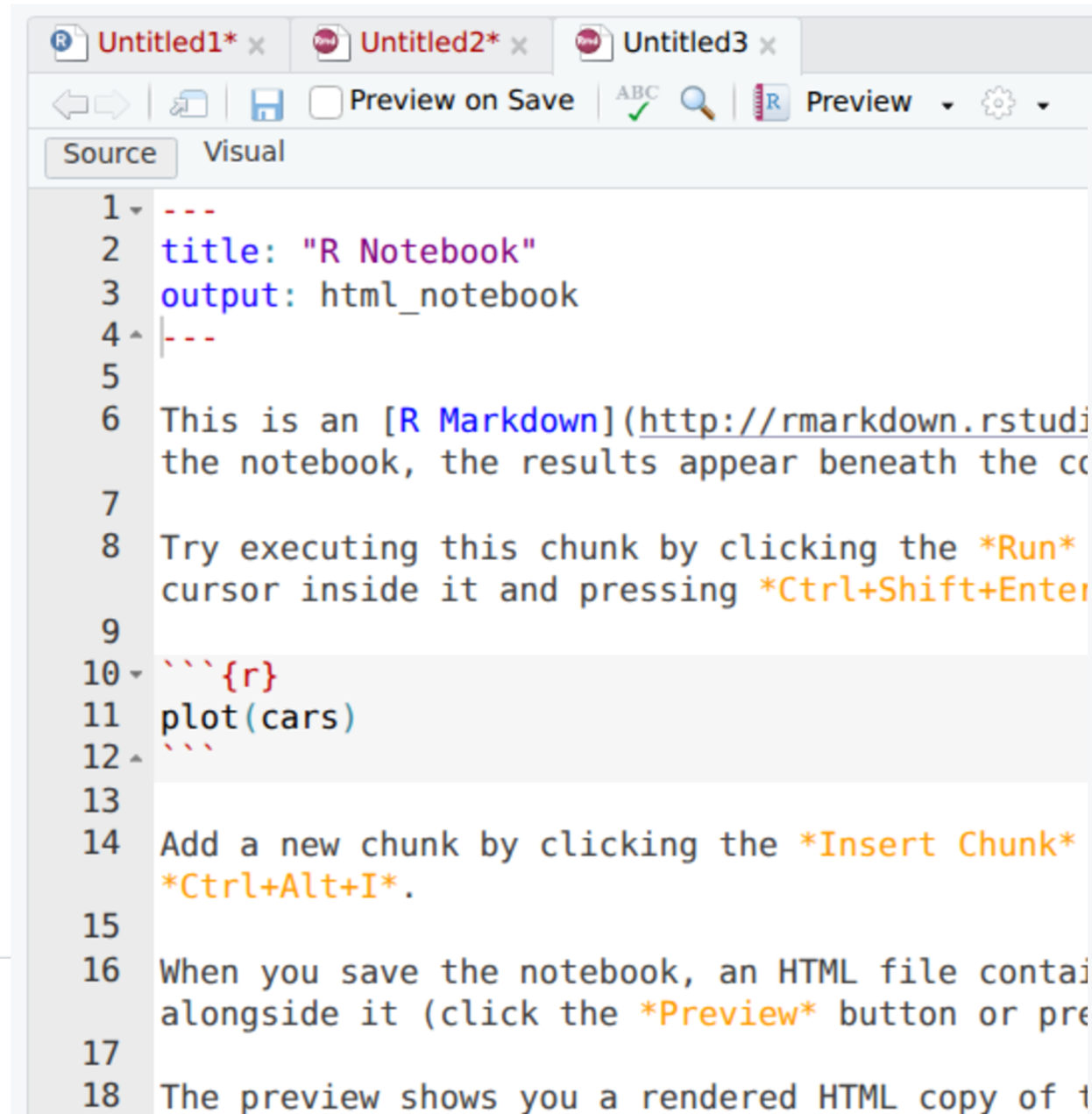


# RStudio Files



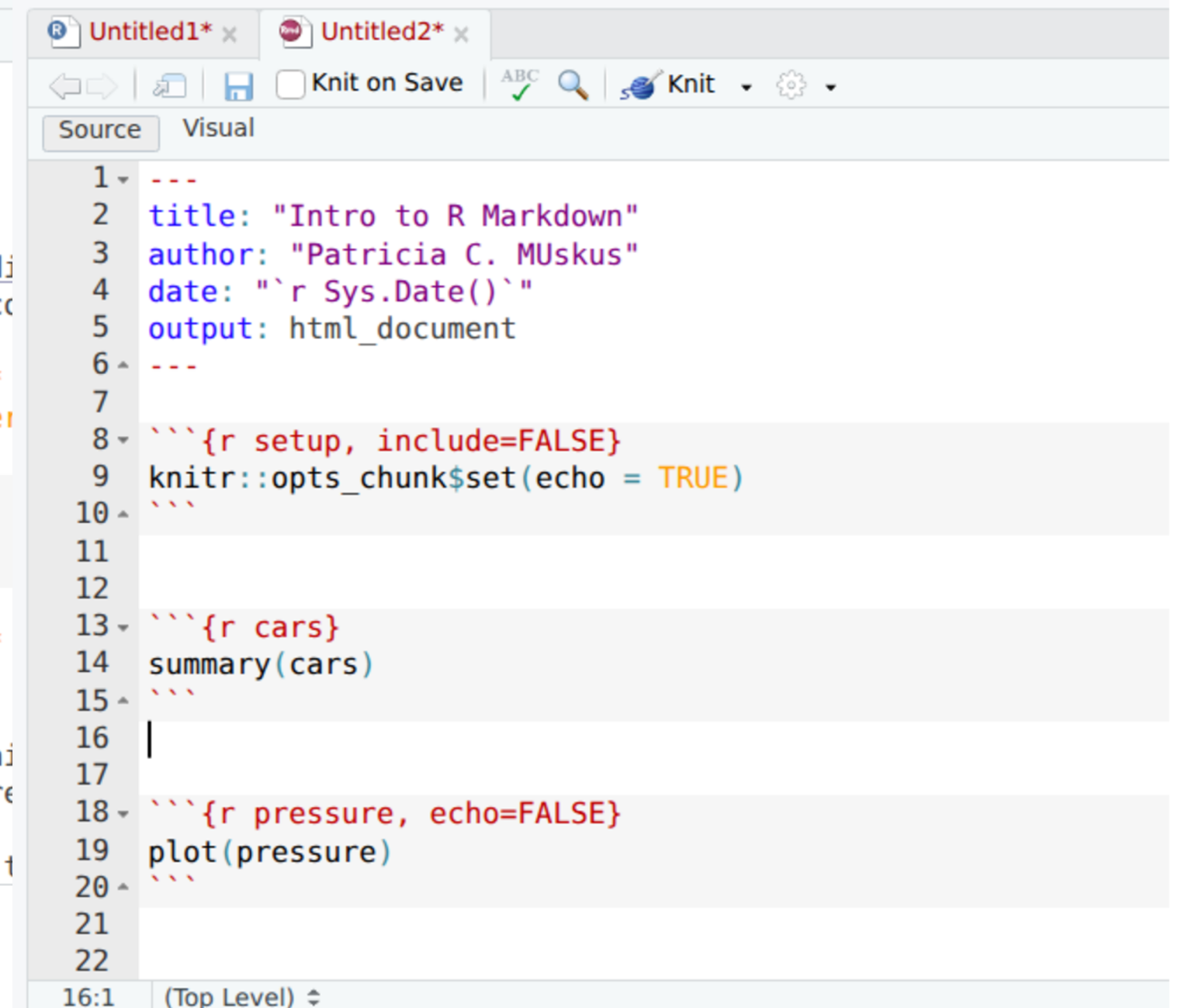
1  
2

Script



```
1 ---  
2 title: "R Notebook"  
3 output: html_notebook  
4 ---  
5  
6 This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute the code in a  
7 the notebook, the results appear beneath the code.  
8 Try executing this chunk by clicking the *Run* button or pressing *Ctrl+Shift+Enter* on your  
9  
10 ```{r}  
11 plot(cars)  
12 ```  
13  
14 Add a new chunk by clicking the *Insert Chunk* button or pressing *Ctrl+Alt+I* on your  
15  
16 When you save the notebook, an HTML file containing the rendered version of the notebook will be  
17 alongside it (click the *Preview* button or press *Ctrl+Shift+V* on your keyboard). For more  
18 The preview shows you a rendered HTML copy of the notebook in the viewer pane.
```

Notebook



```
1 ---  
2 title: "Intro to R Markdown"  
3 author: "Patricia C. MUskus"  
4 date: "`r Sys.Date()`"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12  
13 ```{r cars}  
14 summary(cars)  
15 ```  
16  
17  
18 ```{r pressure, echo=FALSE}  
19 plot(pressure)  
20 ```  
21  
22  
16:1 (Top Level) ↕
```

Markdown



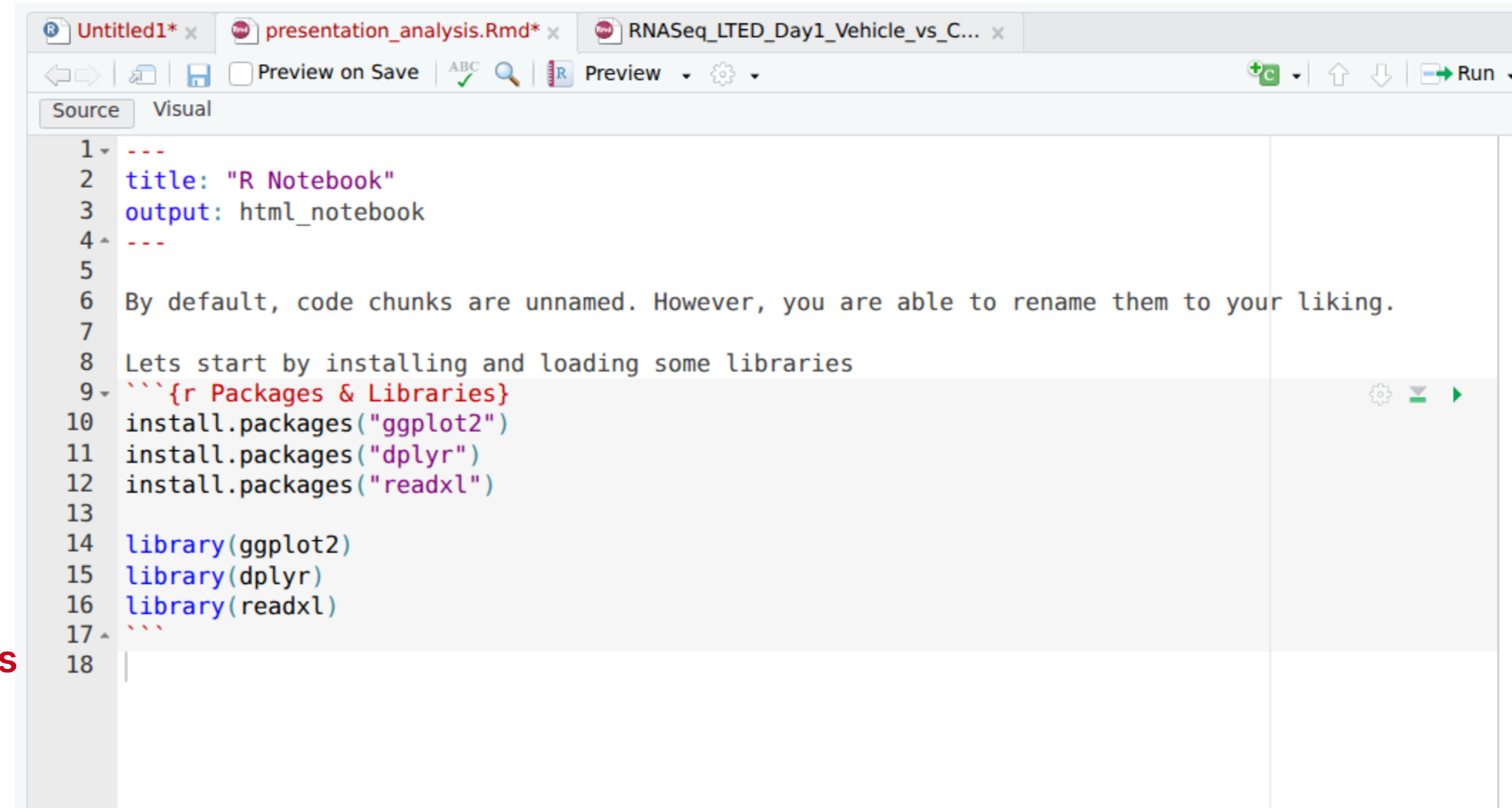
# R Packages & Libraries

To install packages

**`install.packages("package_name")`**

To load packages

**`library(package_name)`** ← No quotations



```
1 ---
2 title: "R Notebook"
3 output: html_notebook
4 ---
5
6 By default, code chunks are unnamed. However, you are able to rename them to your liking.
7
8 Lets start by installing and loading some libraries
9 ```{r Packages & Libraries}
10 install.packages("ggplot2")
11 install.packages("dplyr")
12 install.packages("readxl")
13
14 library(ggplot2)
15 library(dplyr)
16 library(readxl)
17 ```
18 |
```



# R Loading and Reading Data Files

Most data extensions can be read using baseR functions.

However, some data file extensions will need the use of particular packages to be installed and loaded

Let's load/read a tsv type data

Using baseR...

**read\_delim**(file = "data/raw\_data/data.tsv")

Using readr package...

**read\_tsv**(file = "data/raw\_data/data.tsv")

The screenshot displays the RStudio interface. The Source pane shows R code for loading data files. The Environment pane on the right lists objects: metadata (8 obs. of 18 variables), processed\_data (56748 obs. of 4 variables), raw\_data (56748 obs. of 10 variables), and raw\_data\_baseR (56748 obs. of 10 variables). The Files pane shows the current working directory containing files like .Rprofile, data, introPresentation.Rproj, presentation\_analysis.nb.html, presentation\_analysis.Rmd, renv, and renv.lock.

**loading data**

```
18 library(readxl)
19 library(readr)
20 ...
21
22 Let's create another chunk here: CTRL + ALT + I (macOS: Cmd + Option + I)
23
24 We will load a data file in the next chunk and assign it to an object
25
26 <- shortcut alt + -
27 <- shortcut in macOs option + -
28
29 {r Read Data}
30
31 raw_data <- read_tsv(file = "data/raw_data/E-MTAB-10411-raw-counts.tsv")
32 metadata <- read_tsv(file = "data/metadata/E-MTAB-10411-experiment-design.tsv")
33 processed_data <- read_tsv(file = "data/processed_data/E-MTAB-10411-analytics.tsv")
34
35 raw_data_baseR <- read_delim(file = "data/raw_data/E-MTAB-10411-raw-counts.tsv")
36 class(raw_data_baseR)
37 ...
38
39 [1] "spec_tbl_df" "tbl_df"      "tbl"        "data.frame"
40
33:28 Chunk 2: Read Data
```

**objects**

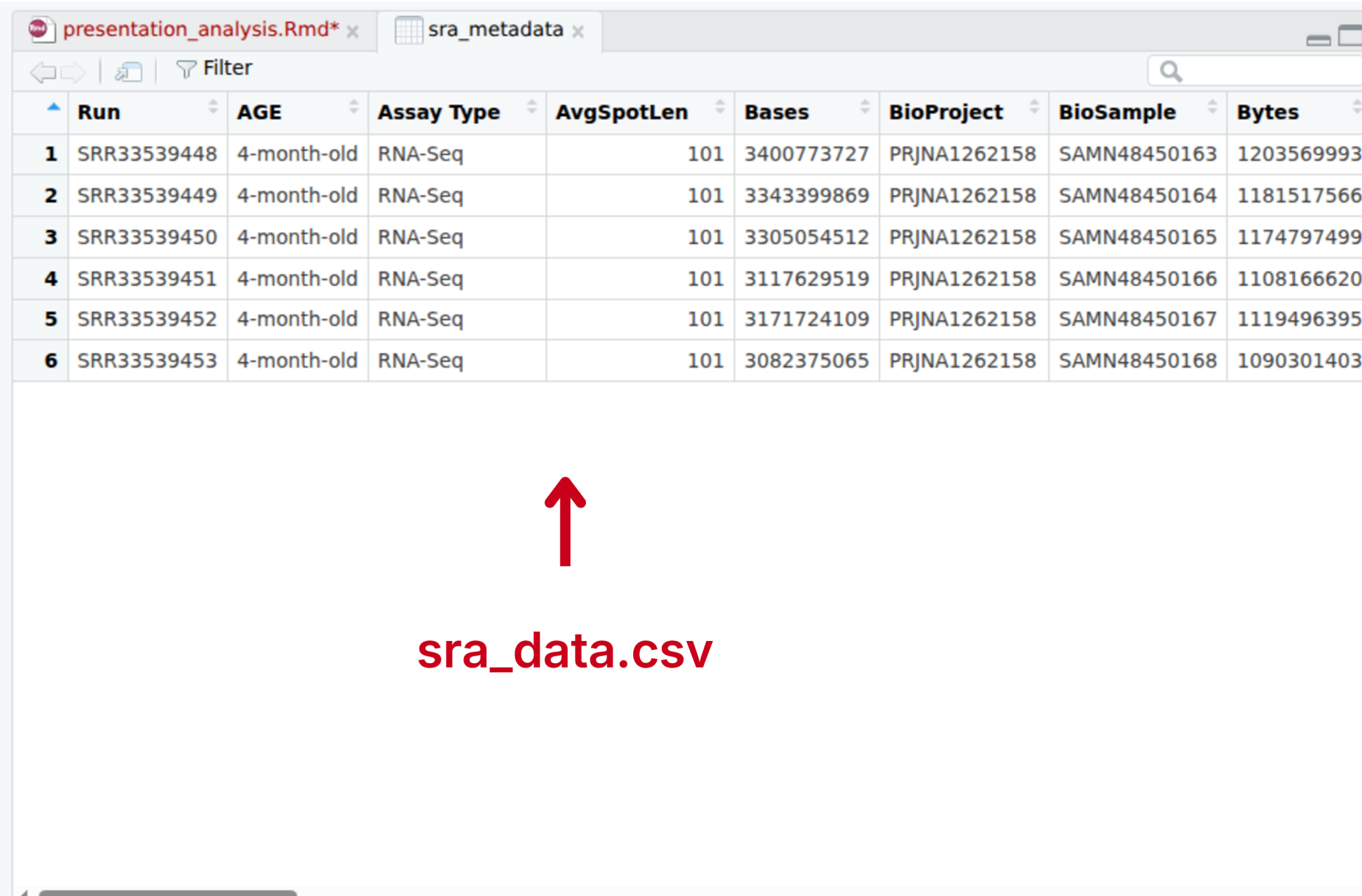
Object	Observations	Variables
metadata	8	18
processed_data	56748	4
raw_data	56748	10
raw_data_baseR	56748	10

**files in current working directory**

Name	Size	Modified
..		
.Rprofile	26 B	Apr 2, 2026, 11:49 PM
data		
introPresentation.Rproj	205 B	Apr 2, 2026, 11:49 PM
presentation_analysis.nb.html	649.9 KB	Apr 3, 2026, 1:02 AM
presentation_analysis.Rmd	409 B	Apr 3, 2026, 1:02 AM
renv		
renv.lock	2.3 KB	Apr 2, 2026, 11:49 PM



# Try it yourself!



presentation\_analysis.Rmd\* x sra\_metadata x

Filter

	Run	AGE	Assay Type	AvgSpotLen	Bases	BioProject	BioSample	Bytes
1	SRR33539448	4-month-old	RNA-Seq	101	3400773727	PRJNA1262158	SAMN48450163	1203569993
2	SRR33539449	4-month-old	RNA-Seq	101	3343399869	PRJNA1262158	SAMN48450164	1181517566
3	SRR33539450	4-month-old	RNA-Seq	101	3305054512	PRJNA1262158	SAMN48450165	1174797499
4	SRR33539451	4-month-old	RNA-Seq	101	3117629519	PRJNA1262158	SAMN48450166	1108166620
5	SRR33539452	4-month-old	RNA-Seq	101	3171724109	PRJNA1262158	SAMN48450167	1119496395
6	SRR33539453	4-month-old	RNA-Seq	101	3082375065	PRJNA1262158	SAMN48450168	1090301403

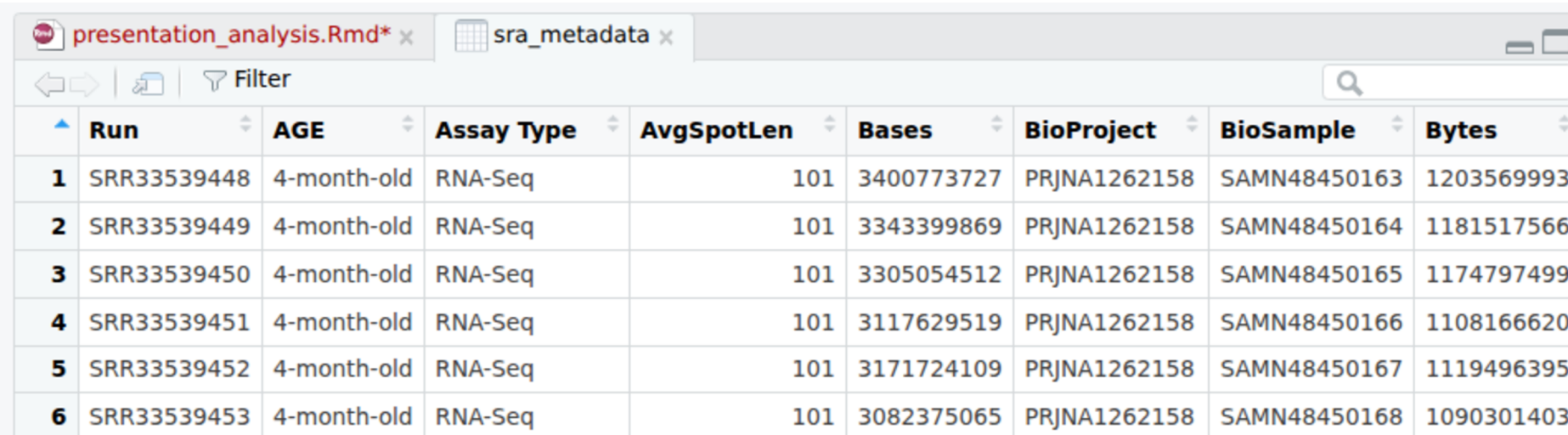
sra\_data.csv

1. Use baseR functions or installed/loaded packages to read a file from your local library ( csv, xls, txt, tsv)
2. Assign the file to a new object
3. Check the file class & explore!

Use “?function” to explore the function’s documentation and learn what else it can do



# Try it yourself!



	Run	AGE	Assay Type	AvgSpotLen	Bases	BioProject	BioSample	Bytes
1	SRR33539448	4-month-old	RNA-Seq	101	3400773727	PRJNA1262158	SAMN48450163	1203569993
2	SRR33539449	4-month-old	RNA-Seq	101	3343399869	PRJNA1262158	SAMN48450164	1181517566
3	SRR33539450	4-month-old	RNA-Seq	101	3305054512	PRJNA1262158	SAMN48450165	1174797499
4	SRR33539451	4-month-old	RNA-Seq	101	3117629519	PRJNA1262158	SAMN48450166	1108166620
5	SRR33539452	4-month-old	RNA-Seq	101	3171724109	PRJNA1262158	SAMN48450167	1119496395
6	SRR33539453	4-month-old	RNA-Seq	101	3082375065	PRJNA1262158	SAMN48450168	1090301403

sra\_data.csv



```
#Reading CSV files with Readr package  
sra_metadata <- read_csv(file = "data/SraRunTable.csv")  
class(sra_metadata)
```

```
#Reading CSV files with baseR  
sra_meta <- read.csv(file = "data/SraRunTable.csv")  
class(sra_meta)
```

```
> class(sra_metadata)  
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"  
>  
> #Reading CSV files with baseR  
> sra_meta <- read.csv(file = "data/SraRunTable.csv")  
> class(sra_meta)  
[1] "data.frame"
```



**Quick Break**



**Back in 10**

# Explore & Clean Data

Let's explore our processed data file

For quick view of first lines

**head(dataObject)**

Let's change the names of our columns

**colnames(dataObject)[3:4] <-  
c("p.val", "log2fc")**

Do we have NAs? Where?

**colSums(is.na(dataObject))**

**dataObject[is.na(dataObject\$`col name`), ]**

Let's remove NAs

**new\_dataObject <- dataObject[!is.na(processed\_data\$p.val), ]**

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for exploring and cleaning data. The code includes comments and functions like `head()`, `colnames()`, `colSums(is.na())`, and `cleaned_data <- processed_data[!is.na(processed_data$p.val), ]`.
- Environment Panel:** Lists data objects: `cleaned_data` (18301 obs. of 4 variables), `metadata` (8 obs. of 18 variables), `processed_data` (56748 obs. of 4 variables), `raw_data` (56748 obs. of 10 variables), `raw_data_baseR` (56748 obs. of 10 variables), `sra_meta` (6 obs. of 34 variables), and `sra_metadata` (6 obs. of 34 variables).
- Files Panel:** Shows a file explorer view of the `data` directory, containing `metadata`, `processed_data`, `raw_data`, and `SraRunTable.csv` (2.7 KB, Oct 14, 2025, 3:00 PM).
- Console:** Shows the execution of the R code, including the output of `colSums(is.na(processed_data))` and `colSums(is.na(cleaned_data))`.

Annotations in red text are placed over the screenshot:

- loading data:** Points to the `cleaned_data` object in the Environment panel.
- objects:** Points to the list of objects in the Environment panel.
- Files:** Points to the file explorer view in the Files panel.

```
> colSums(is.na(processed_data))
Gene ID Gene Name      p.val      log2fc
      0         726    38447         0

> colSums(is.na(cleaned_data))
Gene ID Gene Name      p.val      log2fc
      0         88         0         0
```



# Challenge!

---

Previously we saw how we can manipulate data frames and matrices using baseR

Try to do some of the following manipulations using a package, such as dplyr

```
install.packages("dplyr")  
library(dplyr)
```

Function	baseR	dplyr
Change colnames		
remove NA's		
Subset a row or column		



# Challenge!

Previously we saw how we can manipulate data frames and matrices using baseR

Try to do some of the following manipulations using a package, such as dplyr

```
install.packages("dplyr")  
library(dplyr)
```

Function	baseR	dplyr
Change colnames	<code>colnames(object)[index] ← c("new1","new2")</code>	<code>rename(object,old_name=new_name)</code>
remove NA's	<code>object[!is.na(object\$column, )]</code>	<code>object %&gt;% filter(!is.na(columnName))</code>
Subset a row or column	<code>object[row,col]</code>	<code>select(colname1,colname2)</code>



# R & RStudio on the Cluster

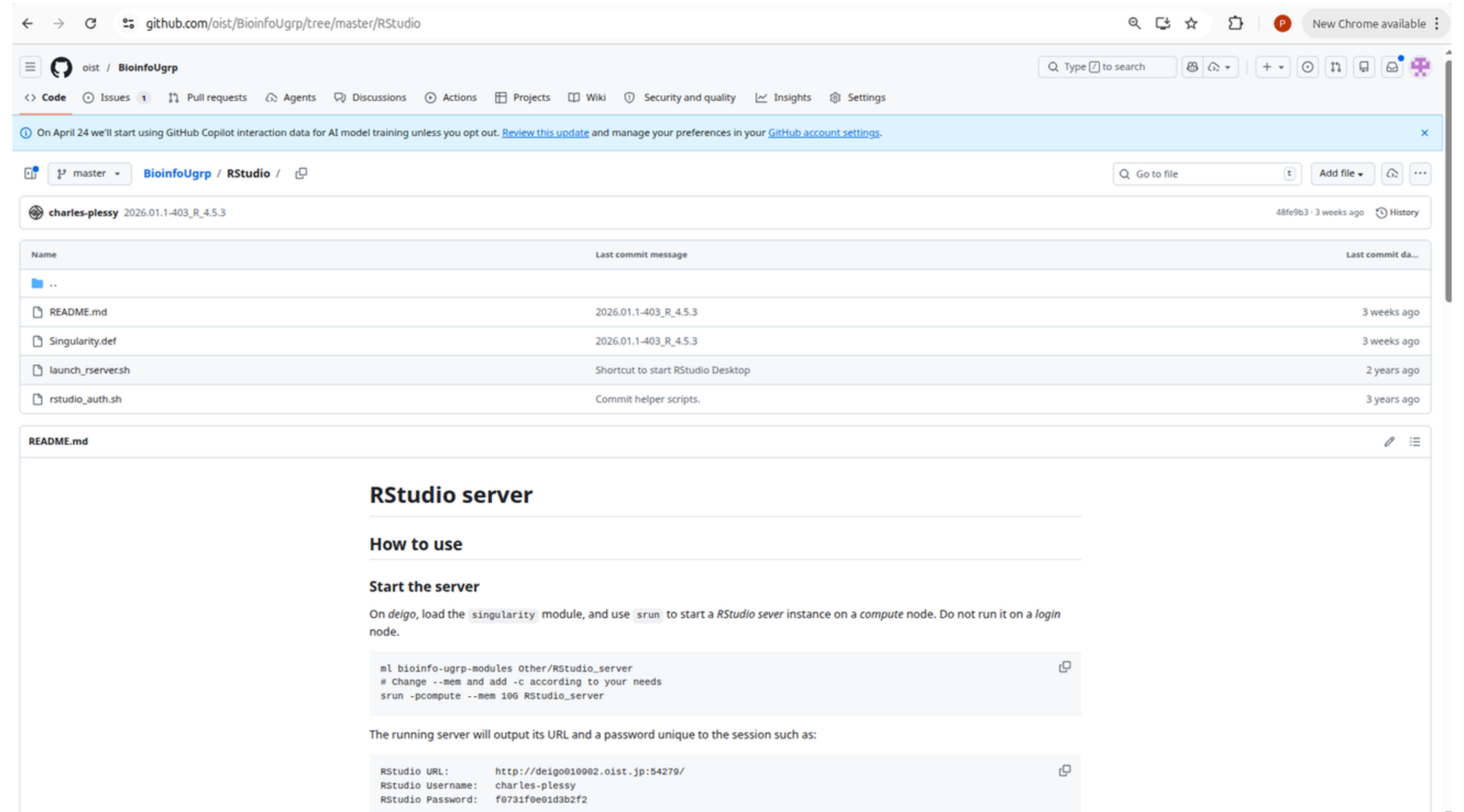
```
$ ssh -X oist-id@deigo.oist.jp
```

```
patricia-muskus@patricia-muskus-Latitude-3340:~$ ssh deigo
patricia-muskus@deigo's password:

*****
*
* Unauthorized access to this resource is prohibited.
* Okinawa Institute of Science and Technology.
*
*****

Last login: Wed Mar 25 13:33:18 2026 from 192.168.252.249
```

## https://github.com/oist/BioinfoUgrp/



github.com/oist/BioinfoUgrp/tree/master/RStudio

oist / BioinfoUgrp

Code Issues Pull requests Agents Discussions Actions Projects Wiki Security and quality Insights Settings

On April 24 we'll start using GitHub Copilot interaction data for AI model training unless you opt out. [Review this update](#) and manage your preferences in your [GitHub account settings](#).

master BioinfoUgrp / RStudio

charles-plessy 2026.01.1-403\_R\_4.5.3 48fe9b3 · 3 weeks ago History

Name	Last commit message	Last commit da...
..		
README.md	2026.01.1-403_R_4.5.3	3 weeks ago
Singularity.def	2026.01.1-403_R_4.5.3	3 weeks ago
launch_rserver.sh	Shortcut to start RStudio Desktop	2 years ago
rstudio_auth.sh	Commit helper scripts.	3 years ago

README.md

### RStudio server

#### How to use

#### Start the server

On *deigo*, load the `singularity` module, and use `srunc` to start a *RStudio sever* instance on a *compute* node. Do not run it on a *login* node.

```
m1 bioinfo-ugrp-modules Other/RStudio_server
# Change --mem and add -c according to your needs
srunc -pcompute --mem 100 RStudio_server
```

The running server will output its URL and a password unique to the session such as:

```
RStudio URL:      http://deigo010902.oist.jp:54279/
RStudio Username: charles-plessy
RStudio Password: f0731f0e01d3b2f2
```



# Intro to R Programming Feedback





*A New ~~Hope~~ Cluster*

*Needs YOU*

*Name our new HPC cluster*

*NOW*

*Until April 17. Do not delay!*

English: <https://scda-guide.oist.jp/name-new-cluster-en>

Japanese: <https://scda-guide.oist.jp/name-new-cluster-ja>



**ENGLISH**



**日本語**