GETTING STARTED

With R Progamming

Patricia C. Muskus

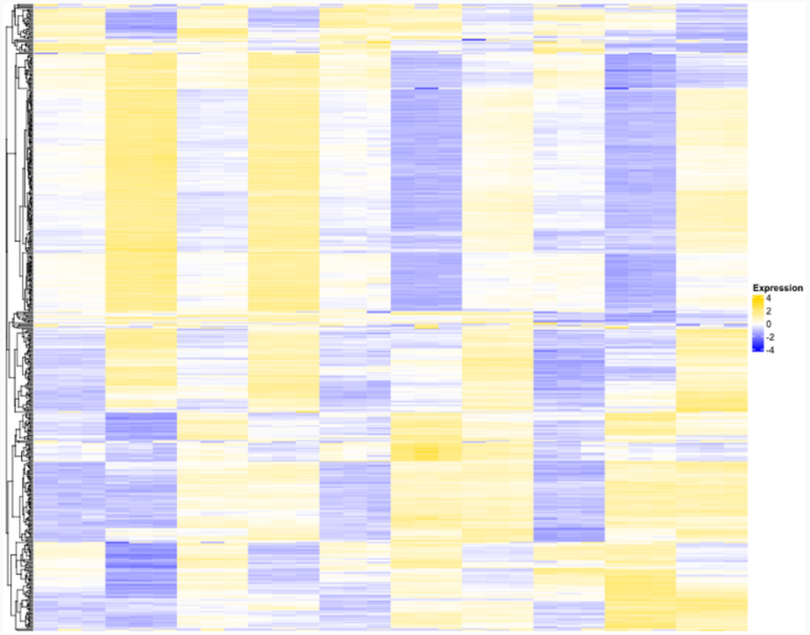
OIST OKINAWA INSTITUTE OF SCIENCE AND TECHNOLOGY

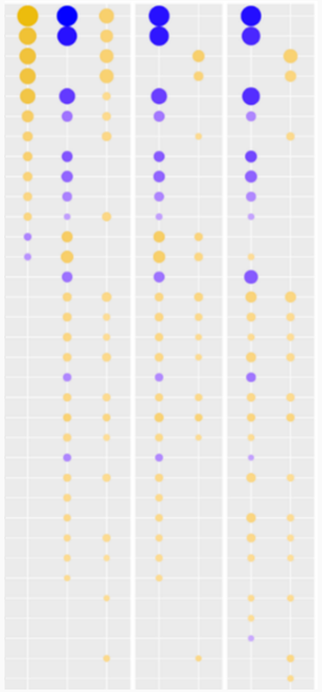
What to Expect?

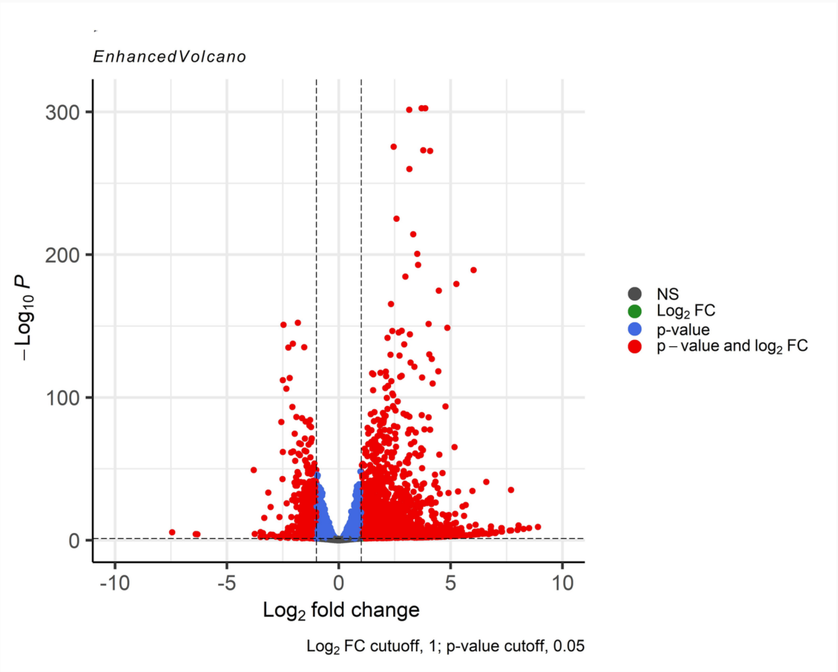
This workshop IS:	• A quick survery on R programming • An introduction to RStudio (local and cluster) • Organization, workflows, available resources
This workshop is NOT:	• A deep dive into R programming • Analysis & interpreting results
Before we begin, do you have...	• R installed locally • RStudio (optional) • Access to Deigo (optional) • Familiarity with terminal basics

What is R?

- R is a statistical open-source programming language and computing environment.
  - Matlab & GraphPad Prism
- Enables diverse analyses: modeling, hypothesis testing, time-series, machine learning, clustering, Shiny applications, & more.
- Generates publication-quality figures
- Large community
- More than 2,300 packages for genomics data analysis (http://bioconductor.org/)







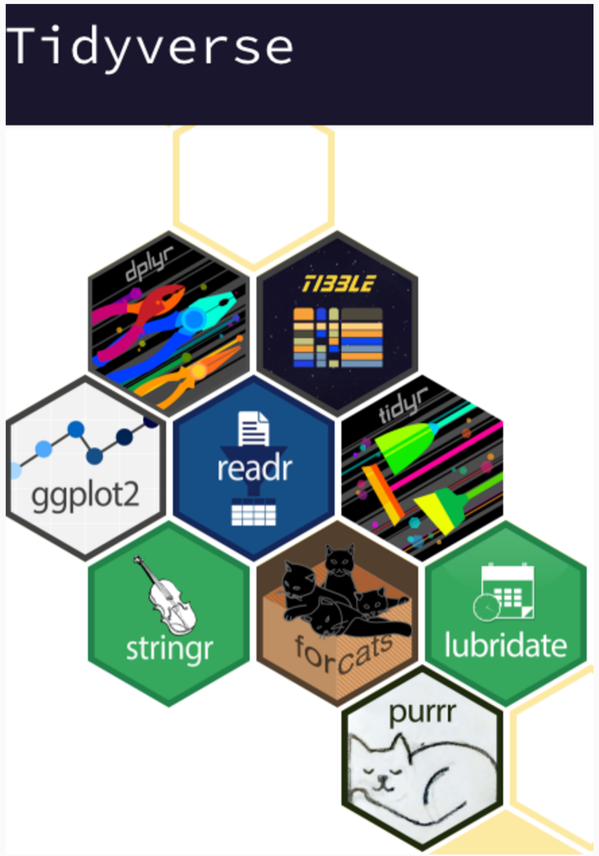
R Documentation & Packages

R Documentation

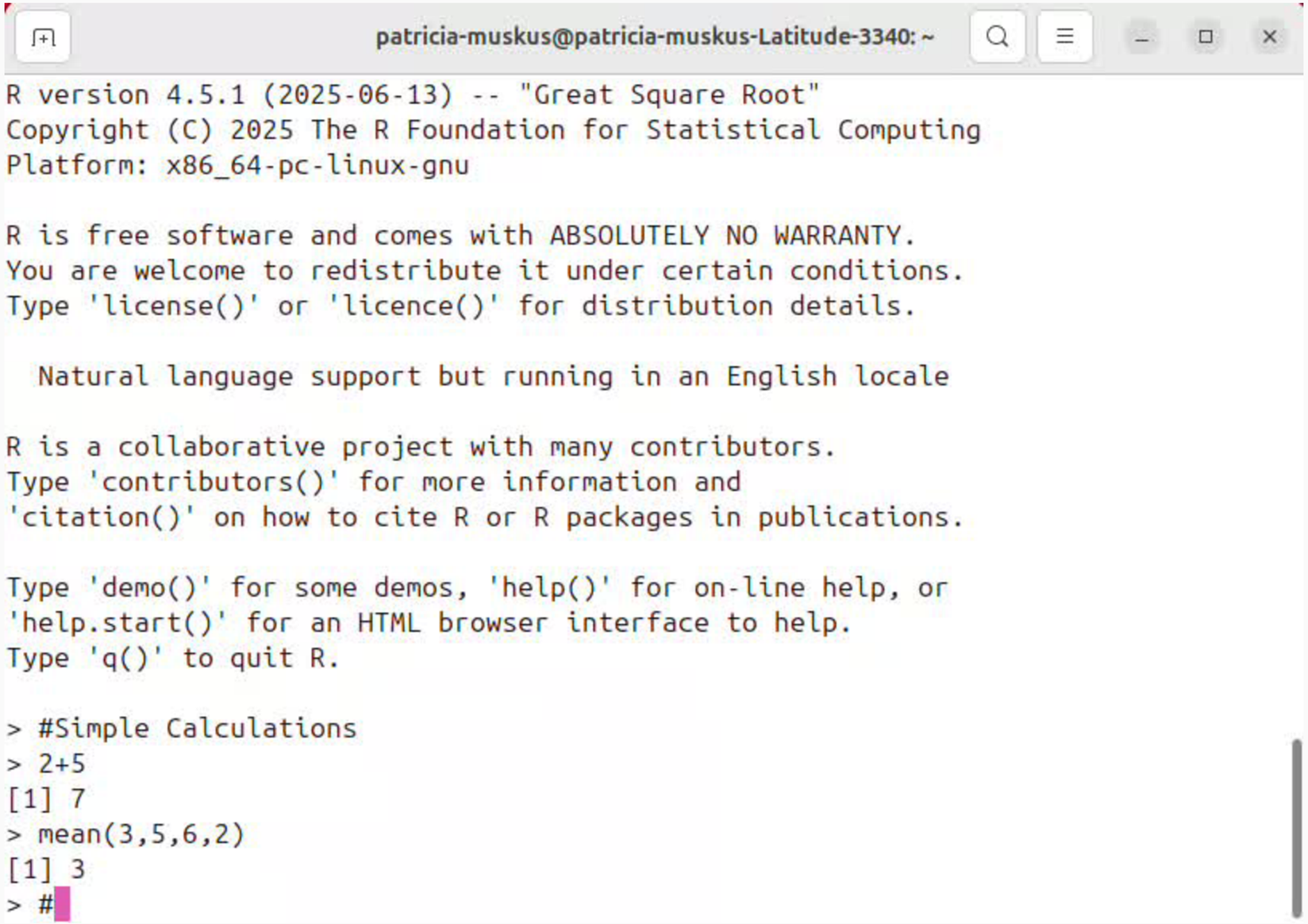
R Packages

dplyr

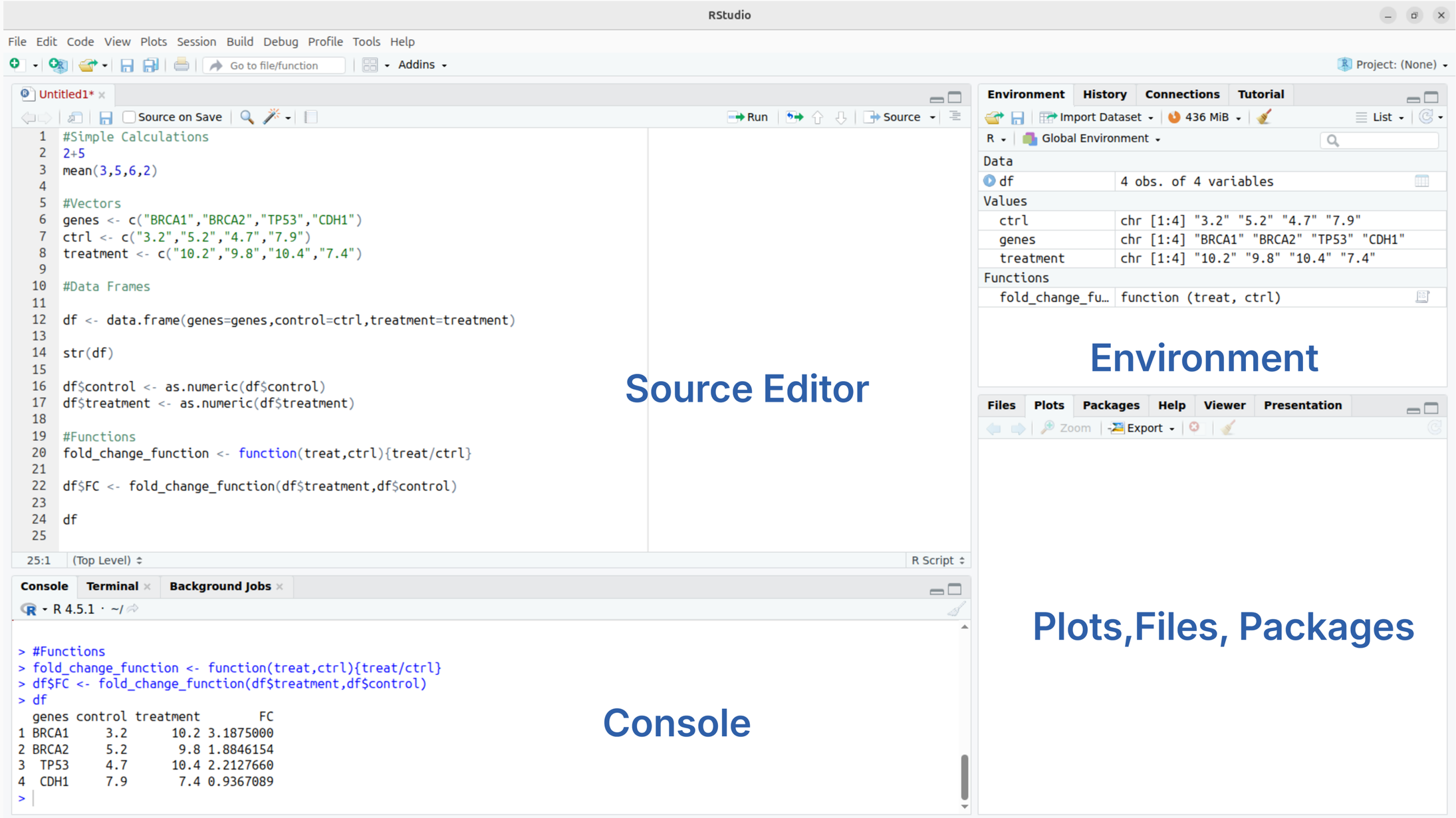
ggplot2



R on the Command Line



R GUI: R-Studio



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

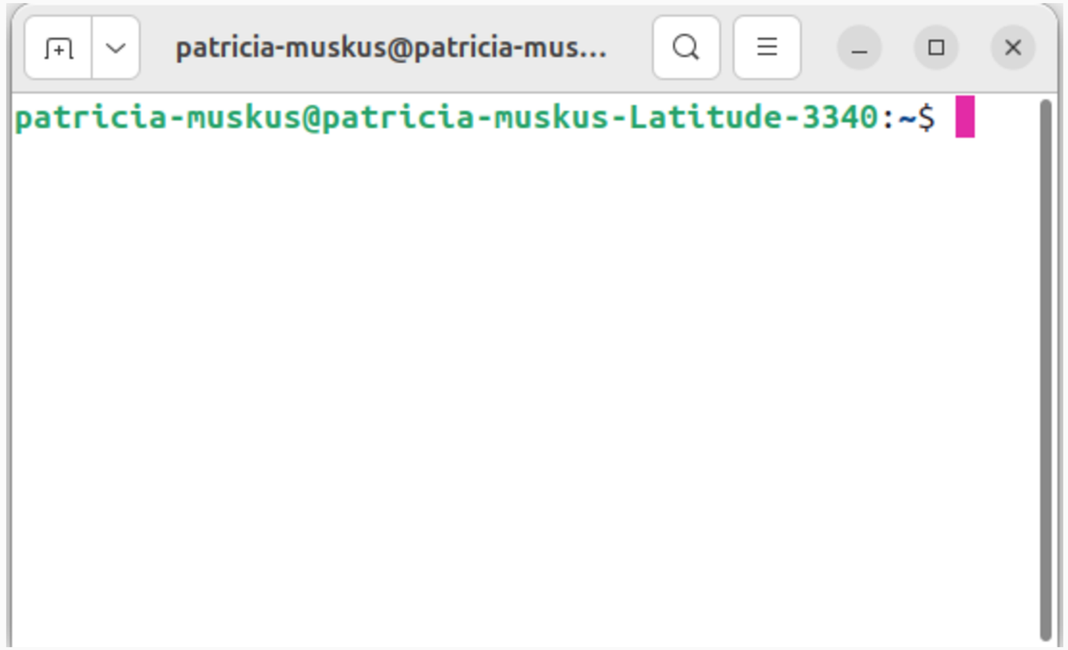
R defaults to single-threaded execution - parallelism can be possible with certain packages

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

Open R

Linux **Ctrl** + **Alt** + **T**

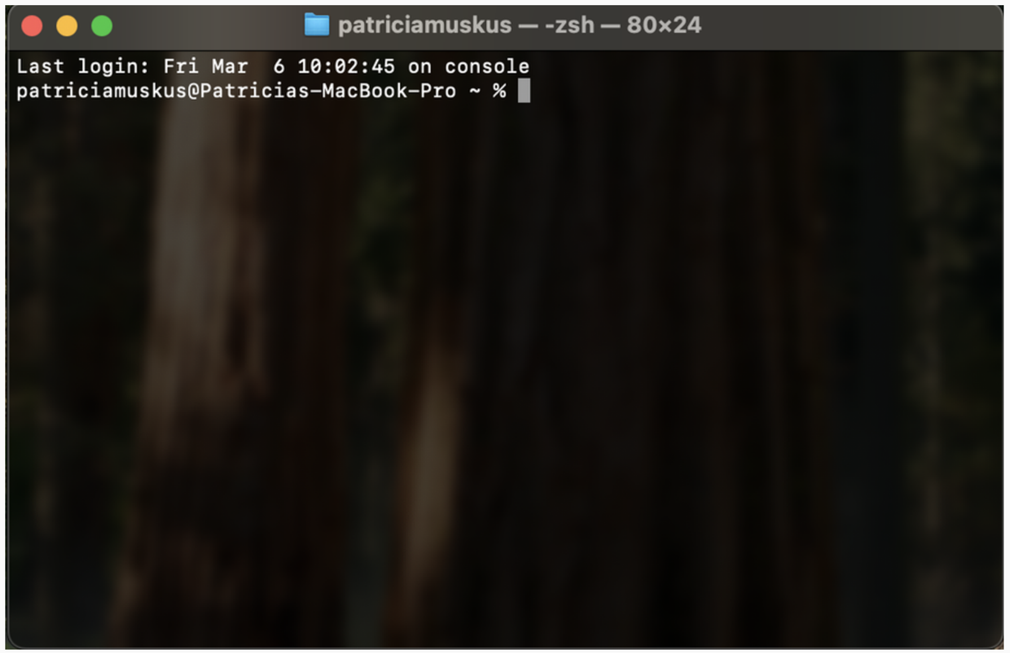
Type “**R**” → Enter



macOS **⌘** + **Space**

Type “**Terminal**” → Enter

Type “**R**” → Enter



R's Core Syntax - Base R

Basic Arithmetic

Base R

```
r$> 2 + 5
[1] 7

r$> 3 * 6
[1] 18

r$> 25 / 5
[1] 5

r$> sqrt(25)
[1] 5

r$> log10(1)
[1] 0

r$> 4 ^ 2
[1] 16
```

Objects & Assignments

```
r$> x <- 9

r$> y <- 10

r$> z <- x + y

r$> x
[1] 9

r$> y
[1] 10

r$> 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

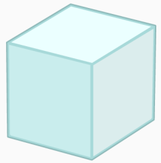
```
r$> class(expresion_level)
[1] "numeric"

r$> class(gene)
[1] "character"

r$> class(brca1_variant)
[1] "logical"

r$> class(brca2_variant)
[1] "logical"
```

Change between data types

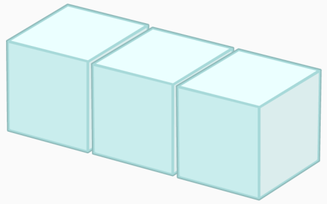
```
r$> as.character(expresion_level)
[1] "23.4"

r$> class(expresion_level)
[1] "numeric"
```


How is data stored in R?

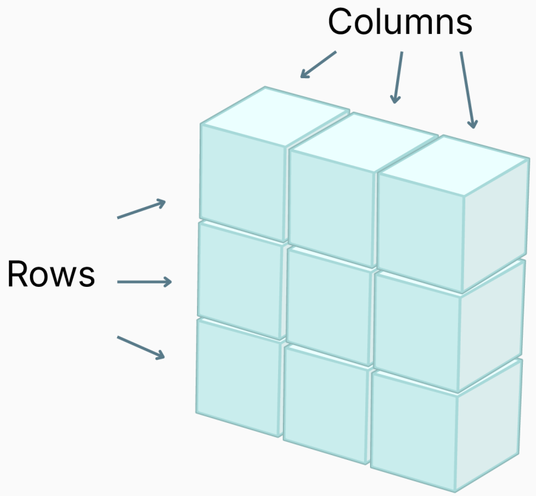
Scalar



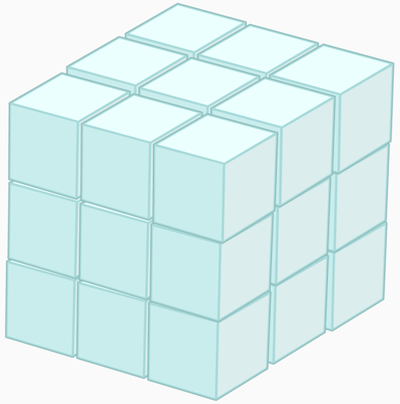
Vector



Matrix

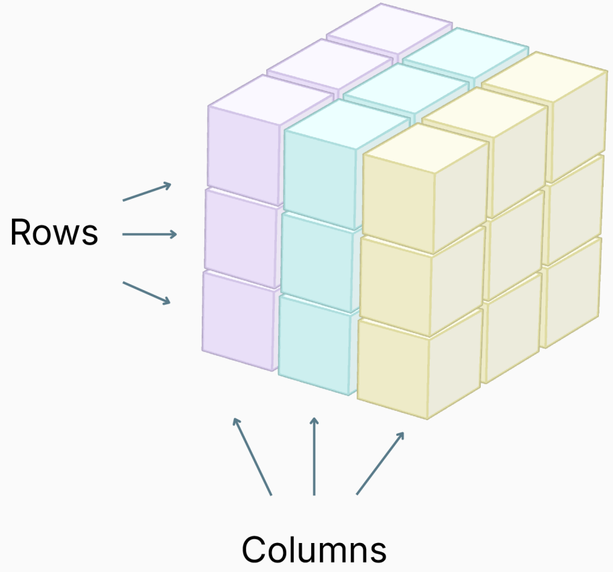


Array



SAME DATA TYPE

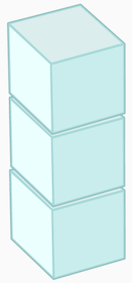
Data Frame



MIXED DATA TYPES

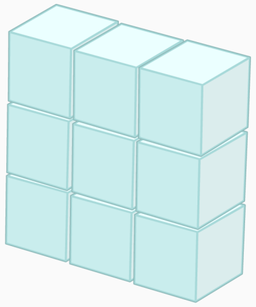
How is data stored in R?

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

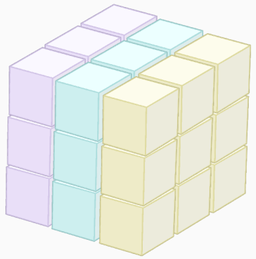
Data: count_matrix

	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
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



```
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,"O
    rganism"= organism,"Sample" = sample)

r$> View(metadata)
```

Example

Rownames

Colnames

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				

```
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","C
    ox5a","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.fra
    me("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

```
r$> deg_results <- data.frame(
    "Genes"=genes,"p.val"=p.va
    l,"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
```

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

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"
```

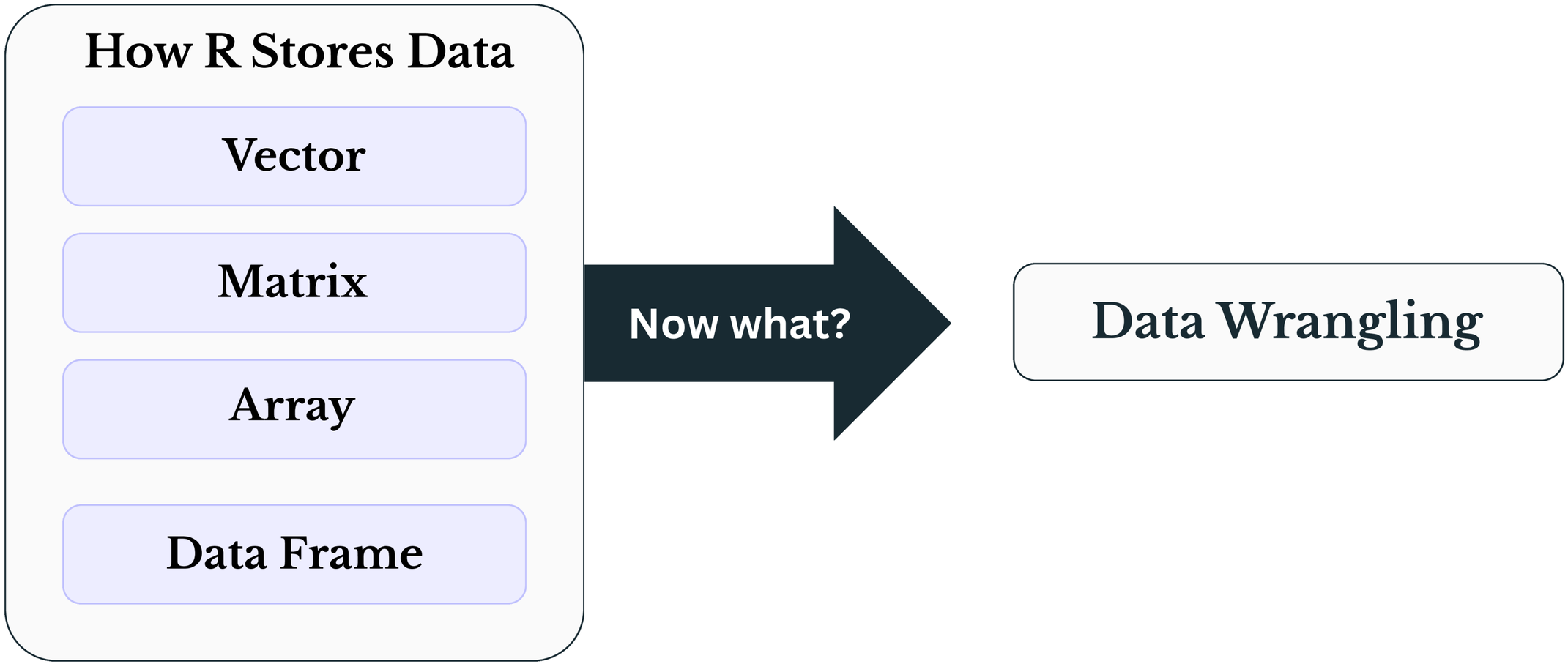
Data: count_matrix

	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

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

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[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!

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

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

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

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

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

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

```
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
```

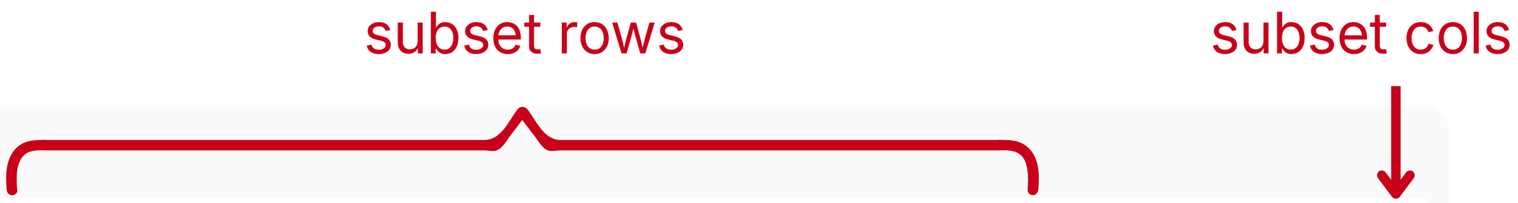

R Programming Concepts

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!




```
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"
```

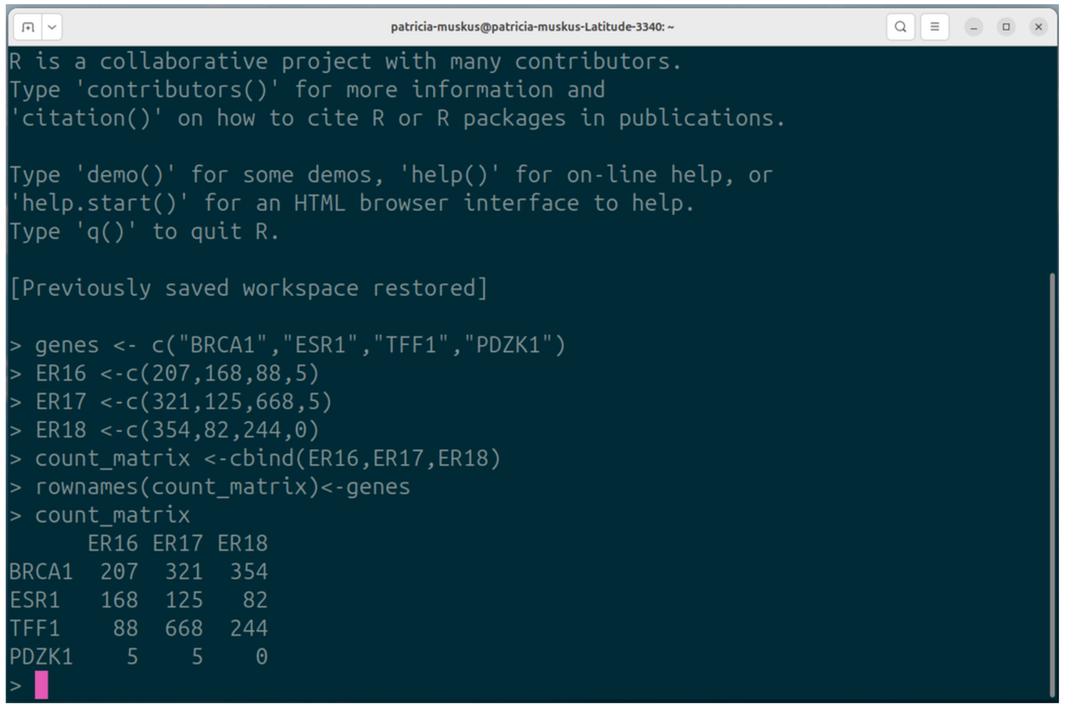
More commands...

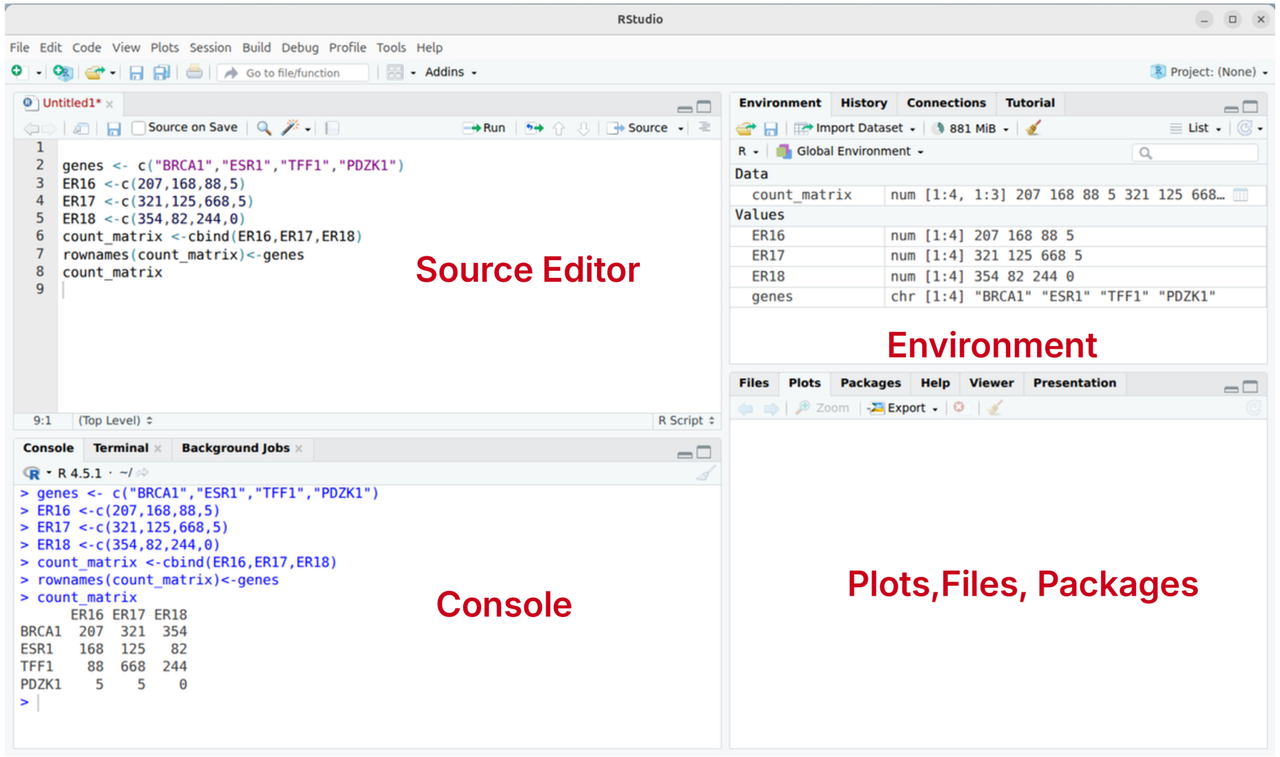
ls() lists all objects in your workspace
rm(object1) will remove the object specified

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



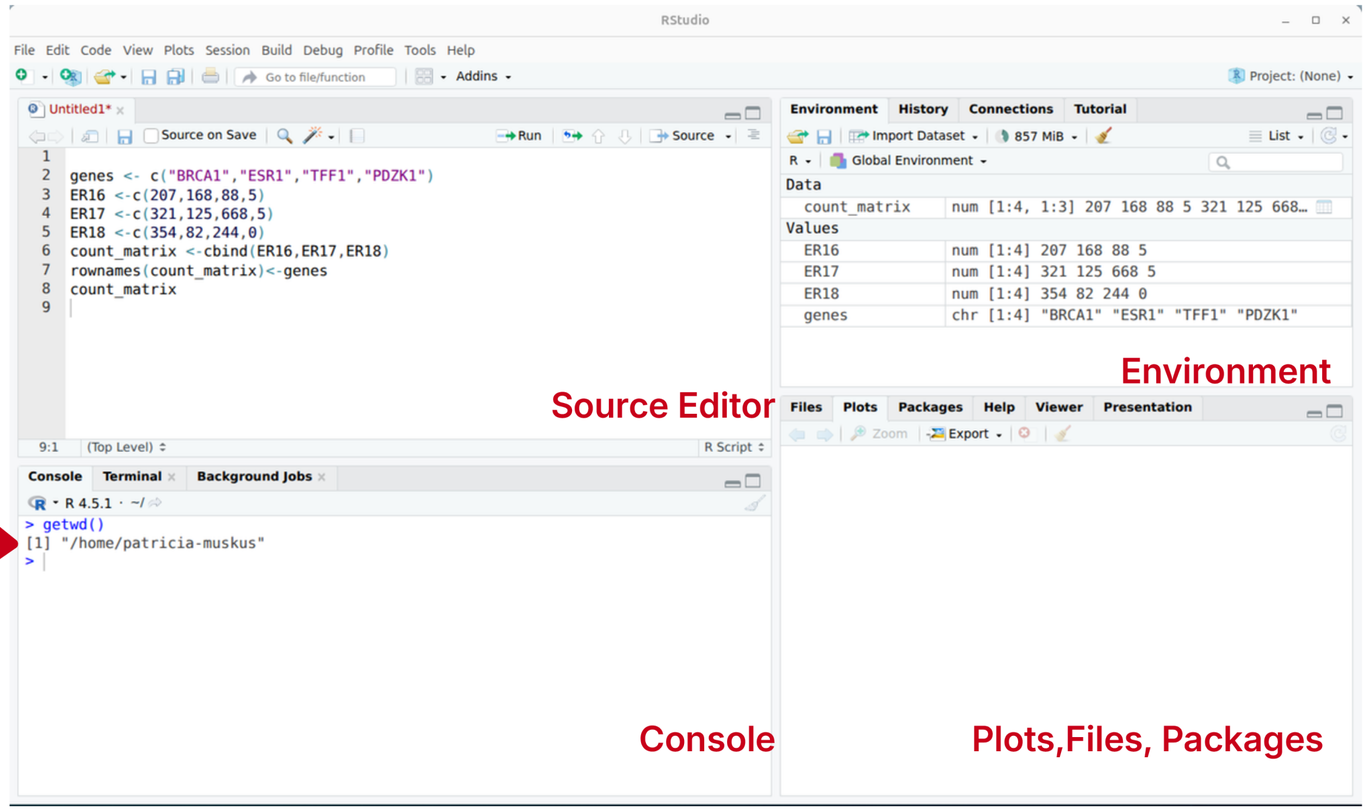


RStudio Working Directory

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

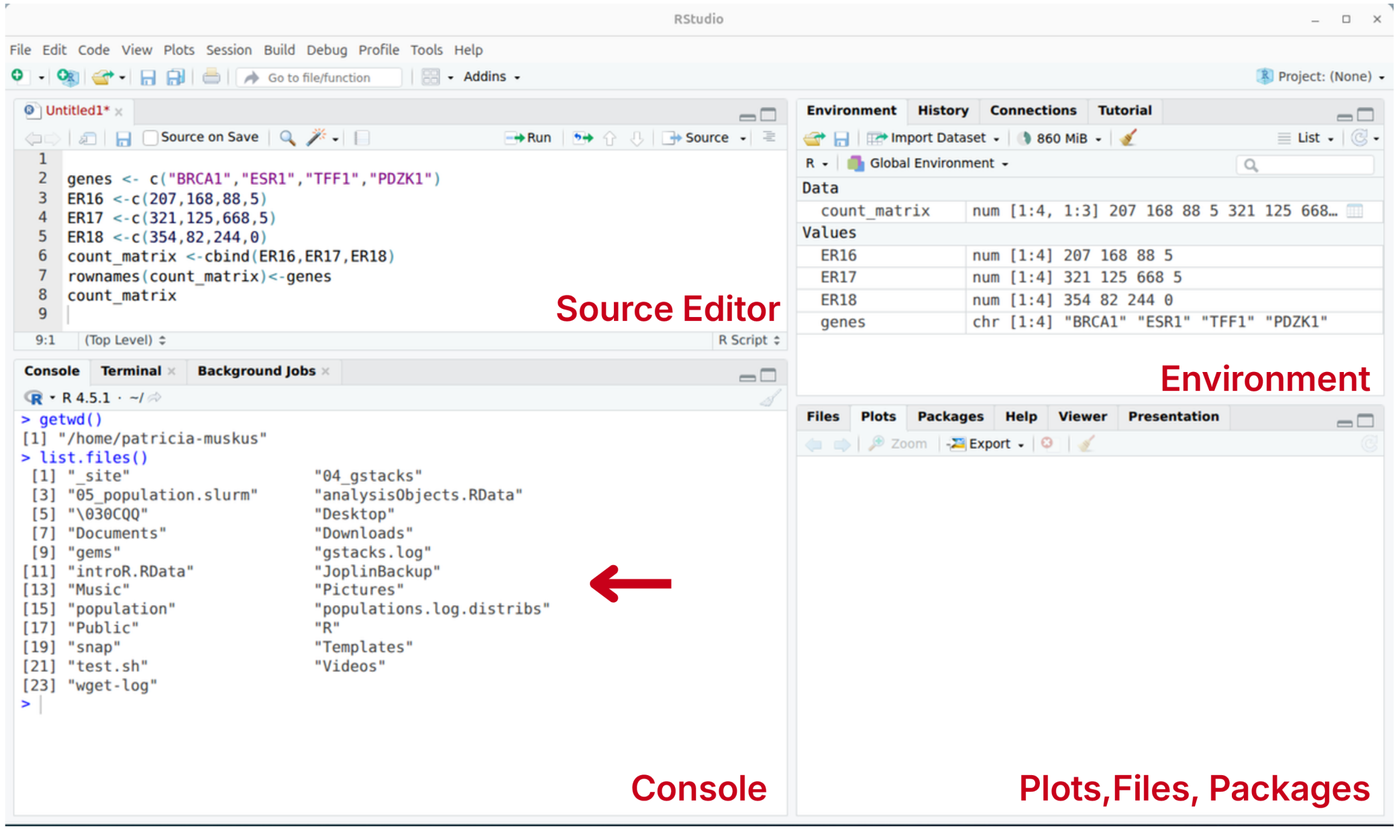
If not specified, this tends to be your "**/home**" directory

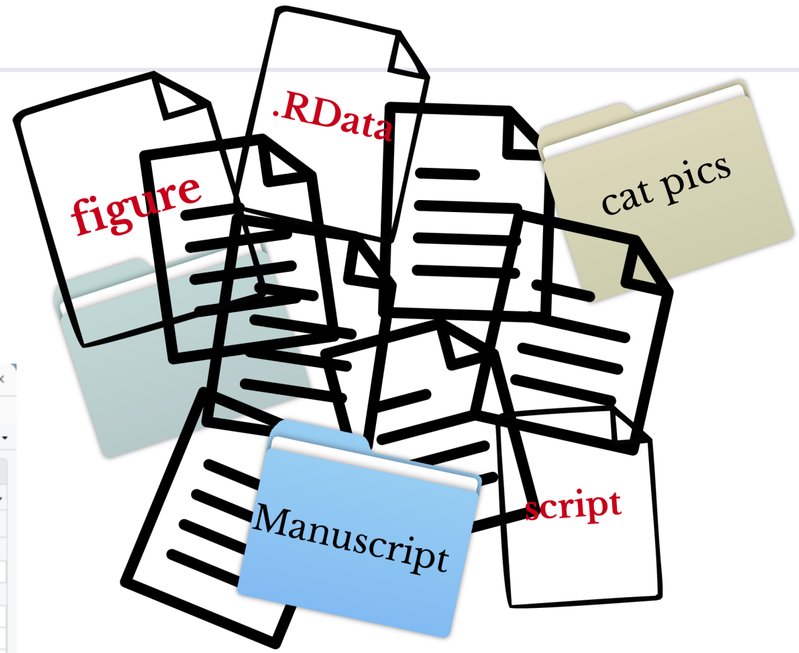


RStudio Working Directory

Why is this a problem???

Let's take a look at the current working dir

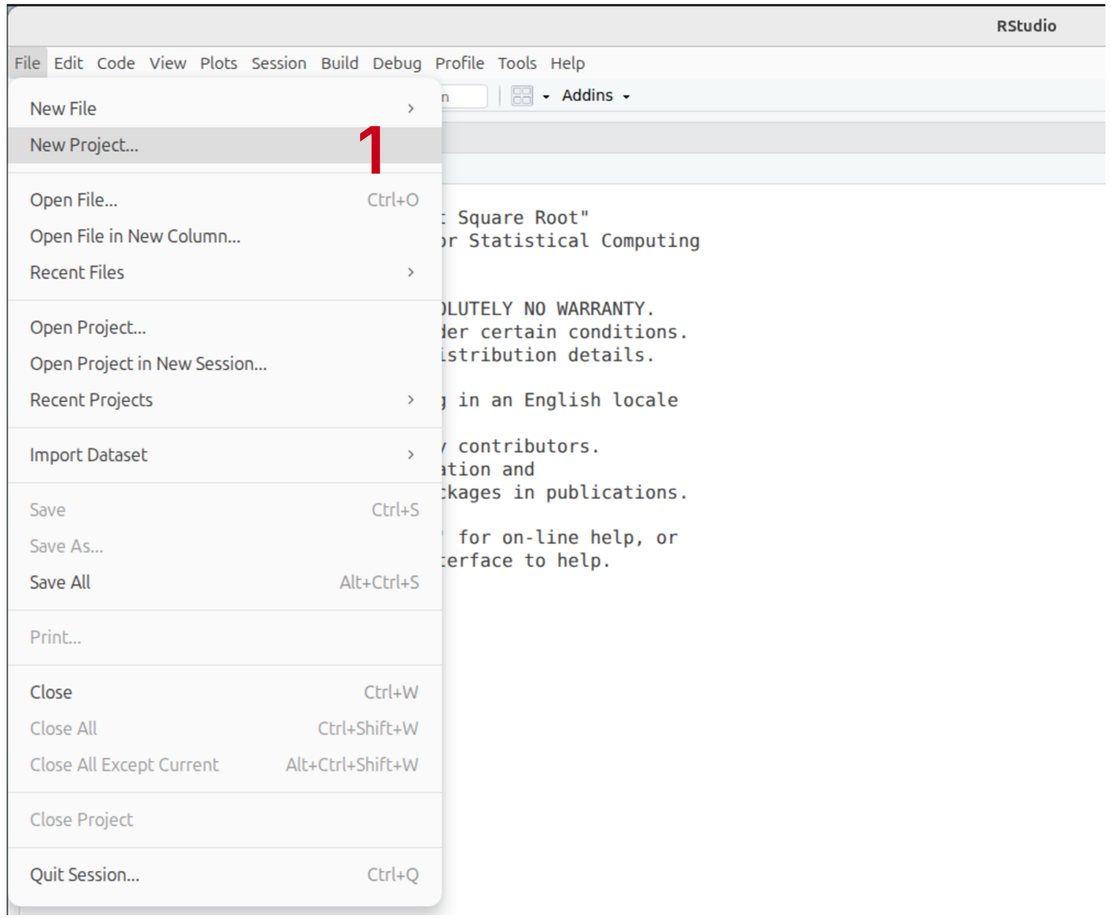


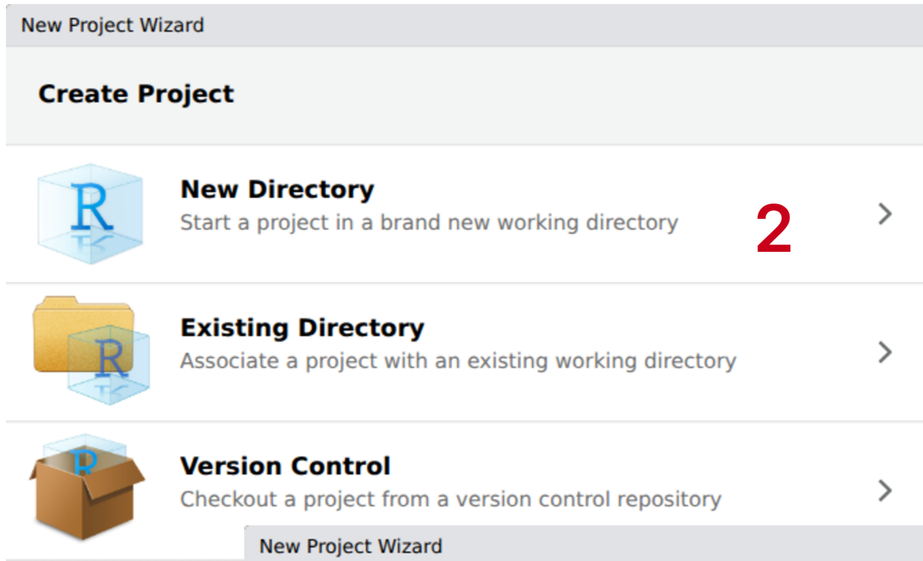


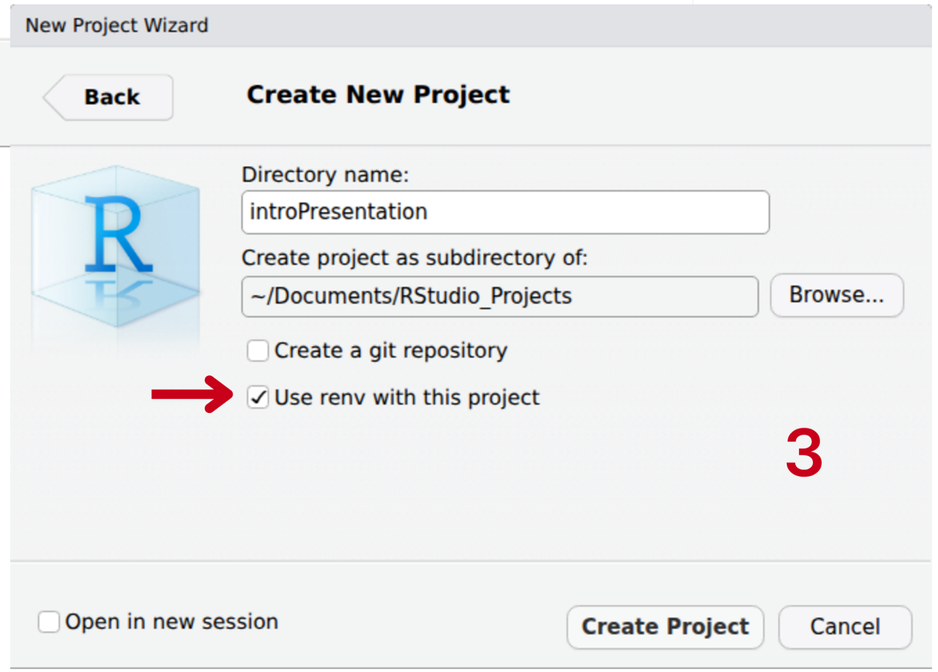

```
R Project/
├── data/
│   ├── raw_data/
│   └── processed_data/
├── scripts/
├── figures/
│   ├── raw_figures/
│   └── publication_figures/
└── tables/
```

Advise? Keep a project for each experiment

RStudio Project



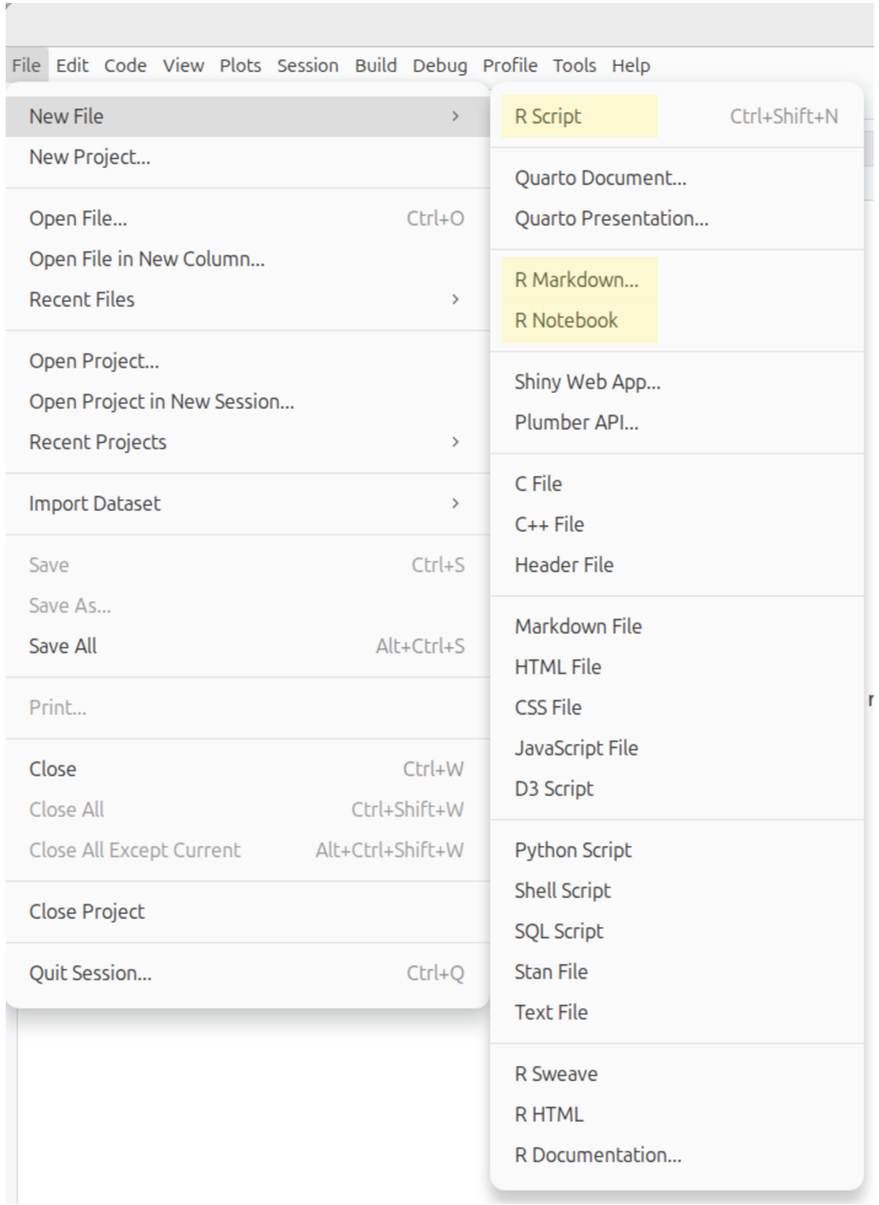




Renv - R package which saves all packages and libraries used in the current project.

learn more here:
https://rstudio.github.io/renv/articles/renv.html

RStudio Files

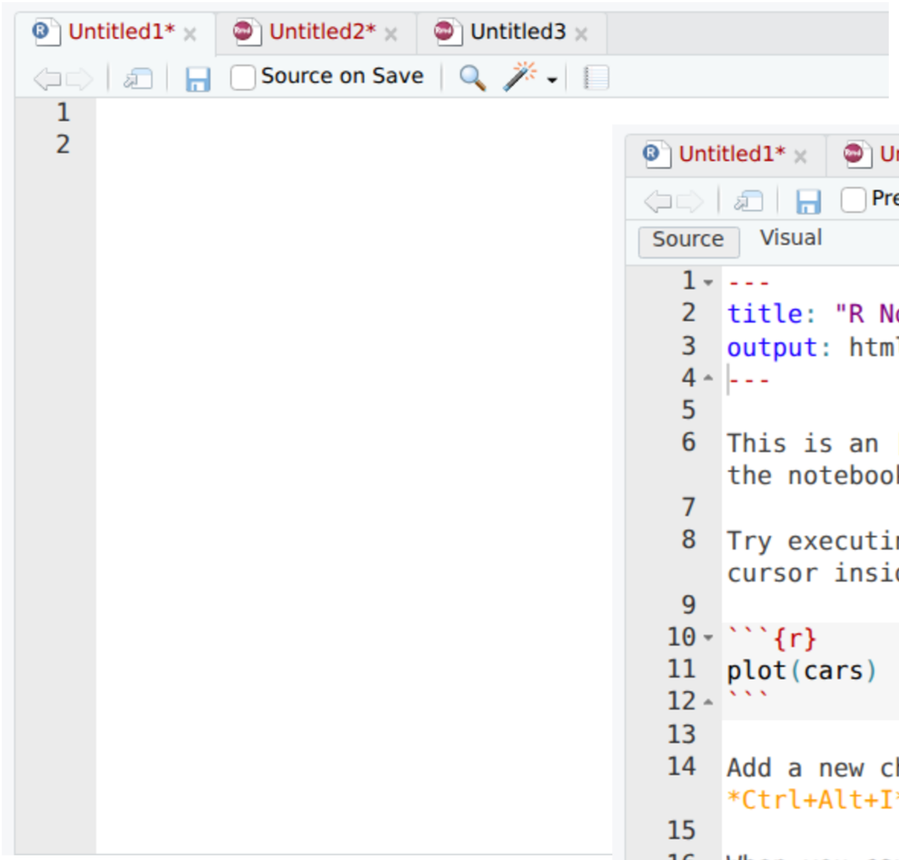


R Notebook- execute code chunks interactively, sending **one line at a time** to the console, and automatically generating a preview when saved.
Good for interactive development and exploratory analysis.

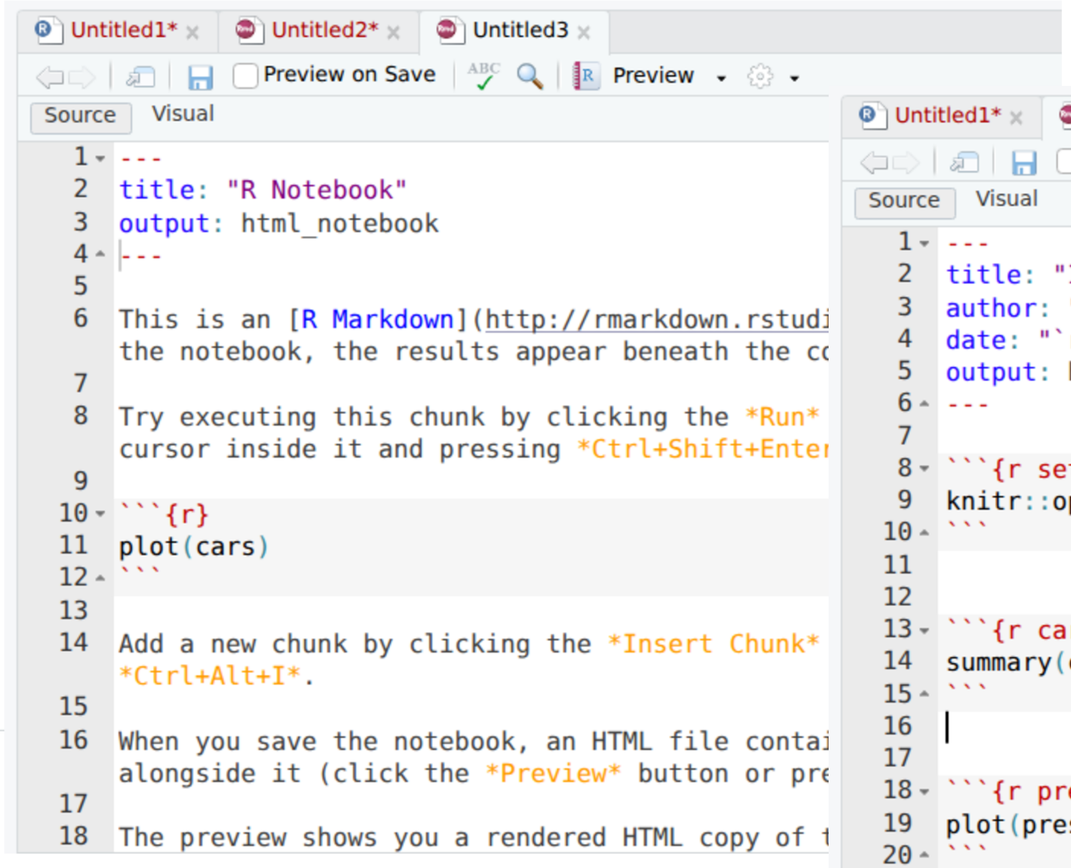
R Markdown- send all code to the console at once when executed.
More flexibility for "knit" or "rendering" options such as PDF, HTML, and Word documents.

Good for creating documentation style render from your analysis

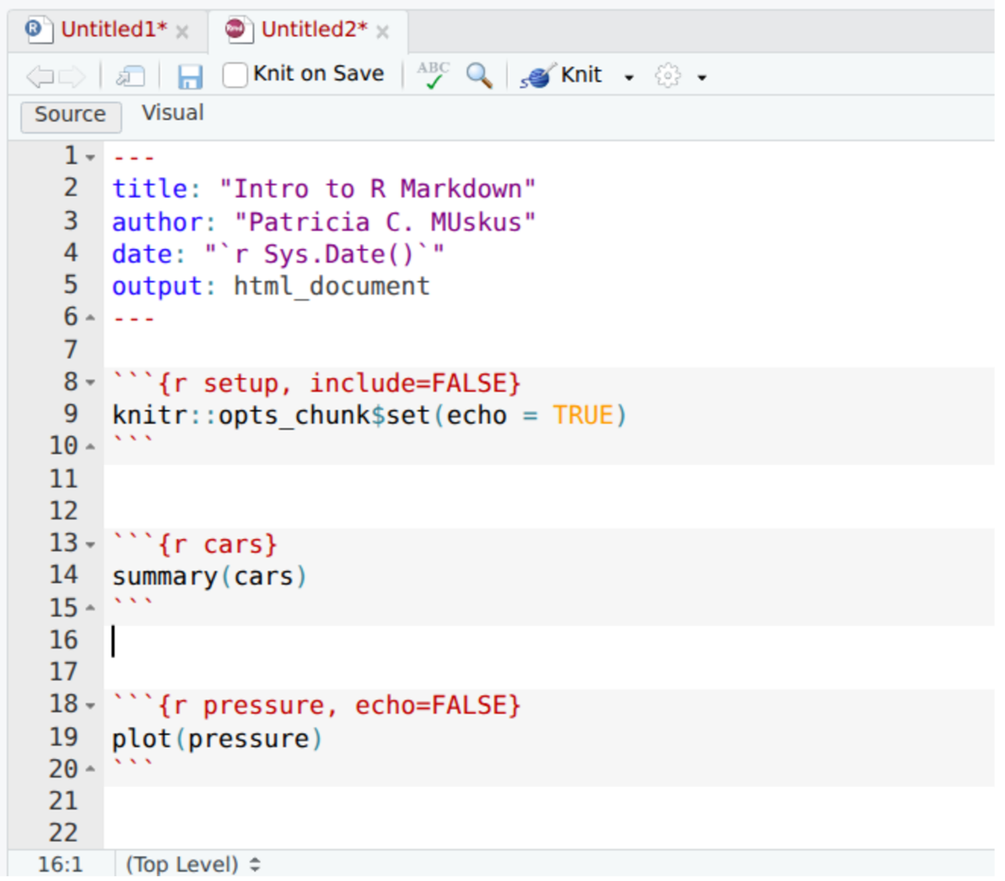
RStudio Files



Script



Notebook



Markdown

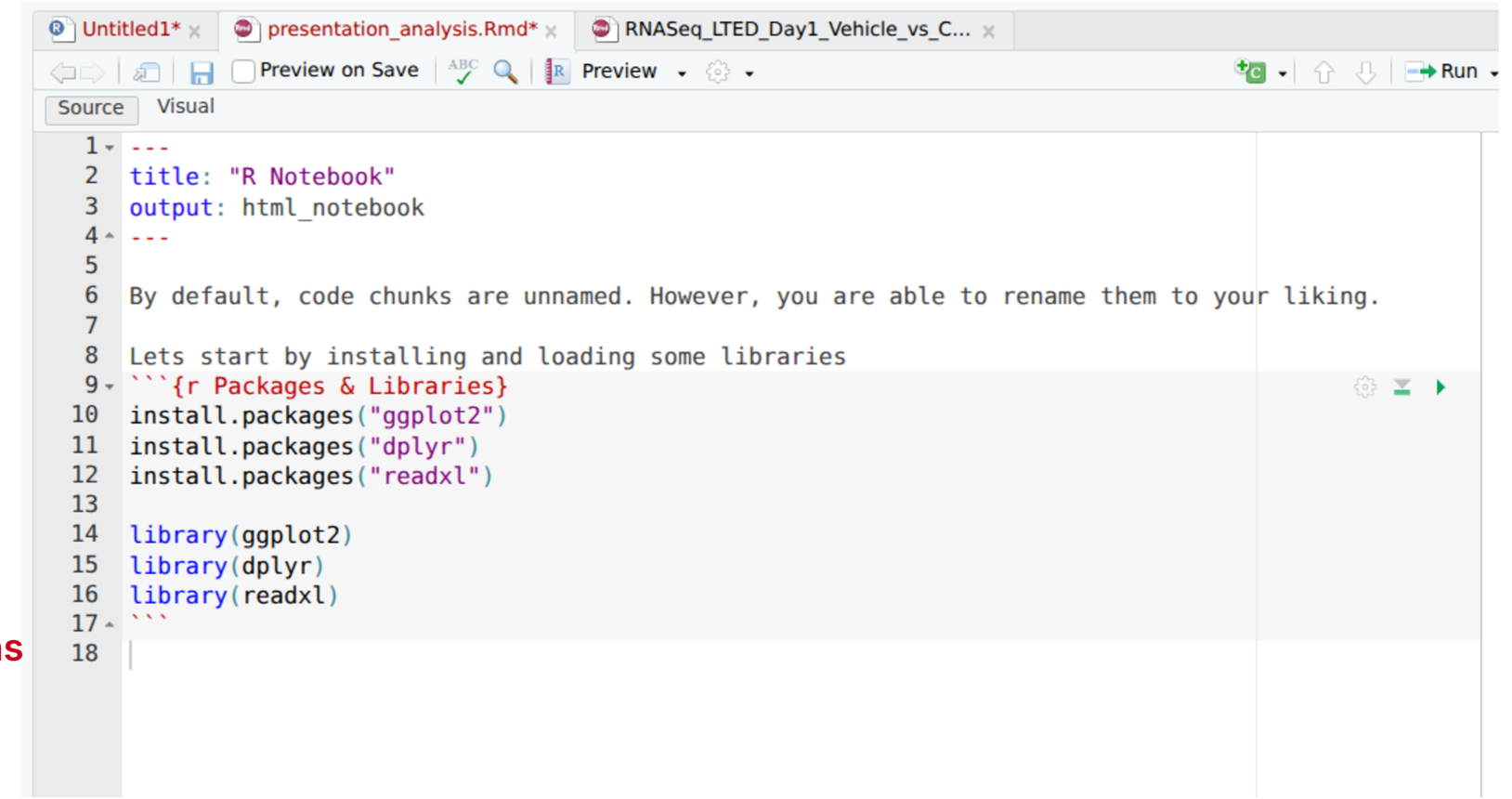
R Packages & Libraries

To install packages

install.packages(“package_name”)

To load packages

library(package_name) ← No quotations



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

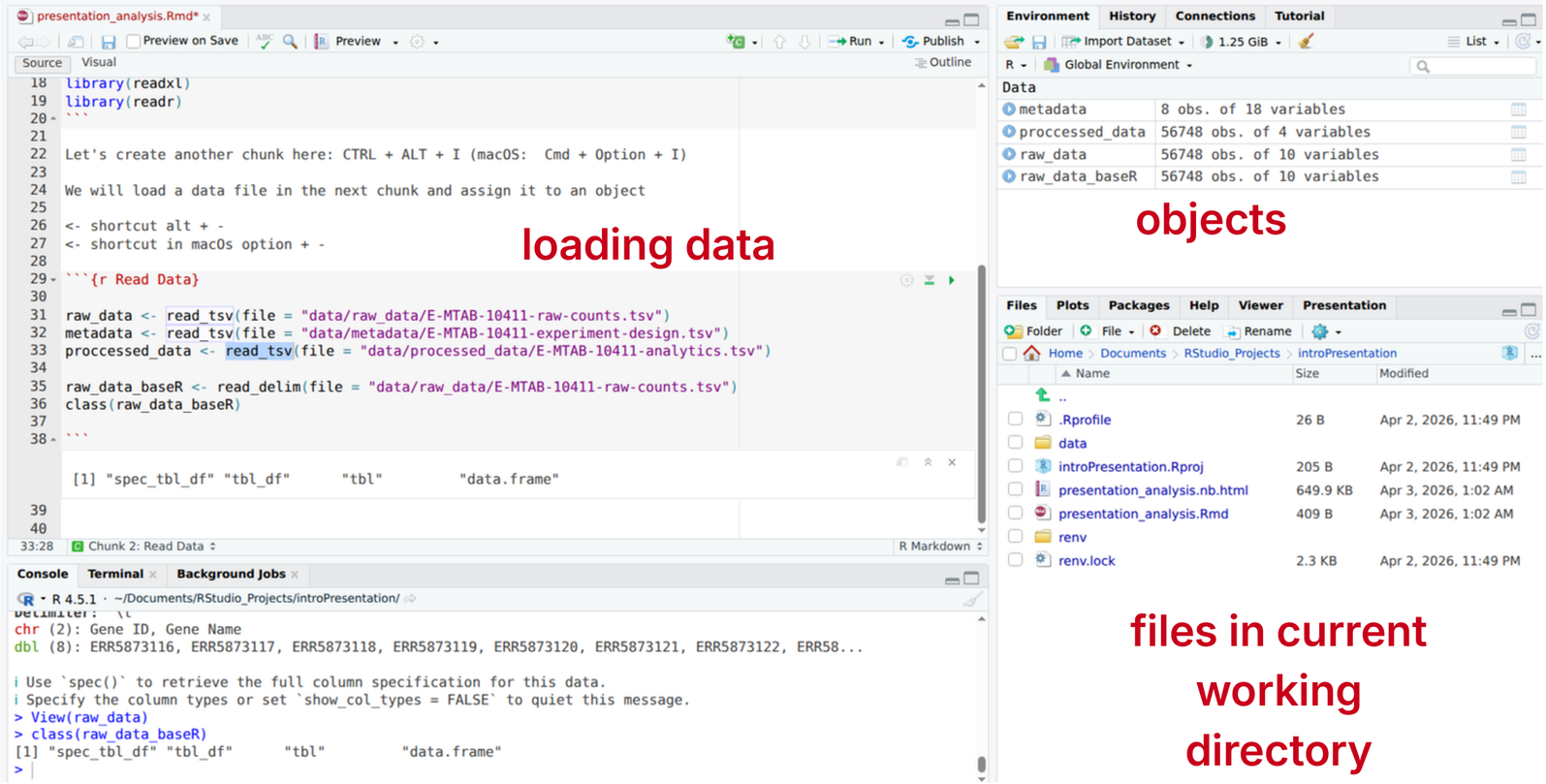
Let's load/read a tsv type data

Using baseR...

read_delim(file = "data/raw_data/data.tsv")

Using readr package...

read_tsv(file = "data/raw_data/data.tsv")



Try it yourself!

presentation_analysis.Rmd* | sra_metadata

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!

presentation_analysis.Rmd* × sra_metadata ×

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

```r
#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")

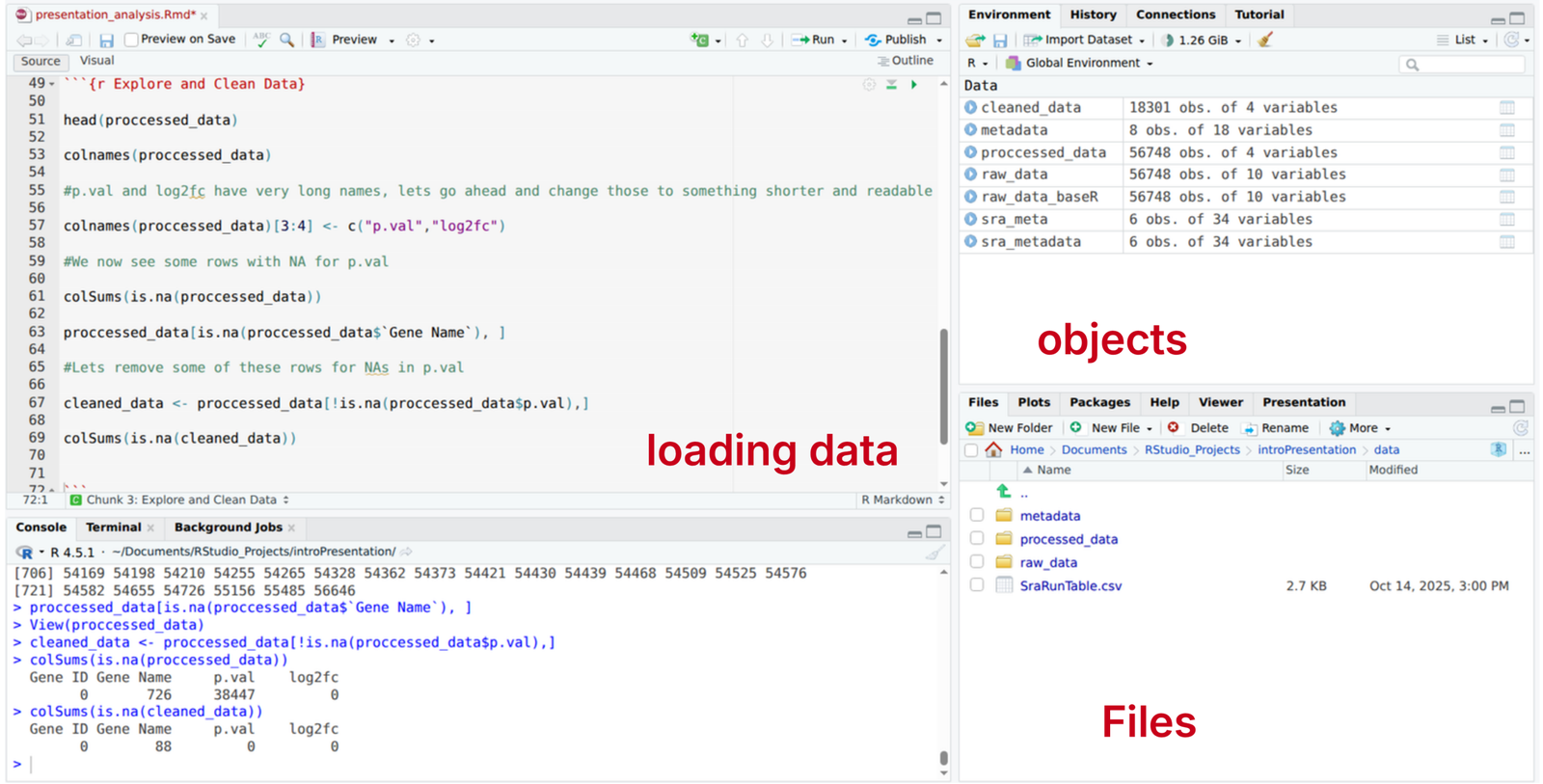
Do we have NAs? Where?

colSums(is.na(dataObject))

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

Let's remove NAs

new_dataObject <- dataObject[!is.na(proccessed_data$p.val),]




```
> colSums(is.na(proccessed_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)
```

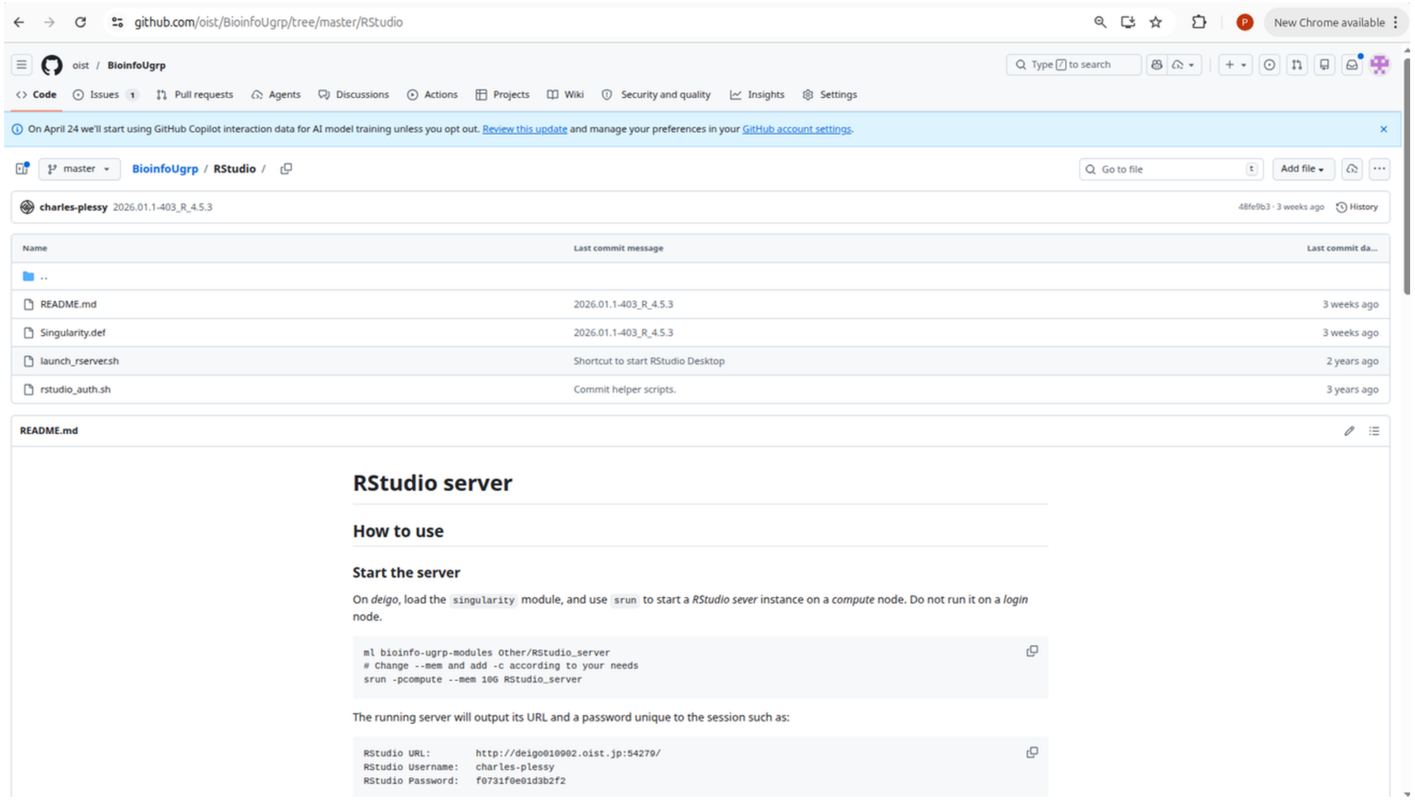
Function	baseR	dplyr
Change colnames	colnames(object)[index] ← c("new1","new2")	rename(object,old_name= new_name)
remove NA's	object[!is.na(object$colu mn,]	object %>% filter(!is.na(columnName))
Subset a row or column	object[row,col]	select(colname1,colname 2)

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/



Intro to R Programming Feedback

ENGLISH


Scientific Computing and Data Analysis
OIST | Core Facilities
OKINAWA INSTITUTE OF SCIENCE AND TECHNOLOGY
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

日本語