Introduction to Bioinformatics

Introduction to Basic Programming with R

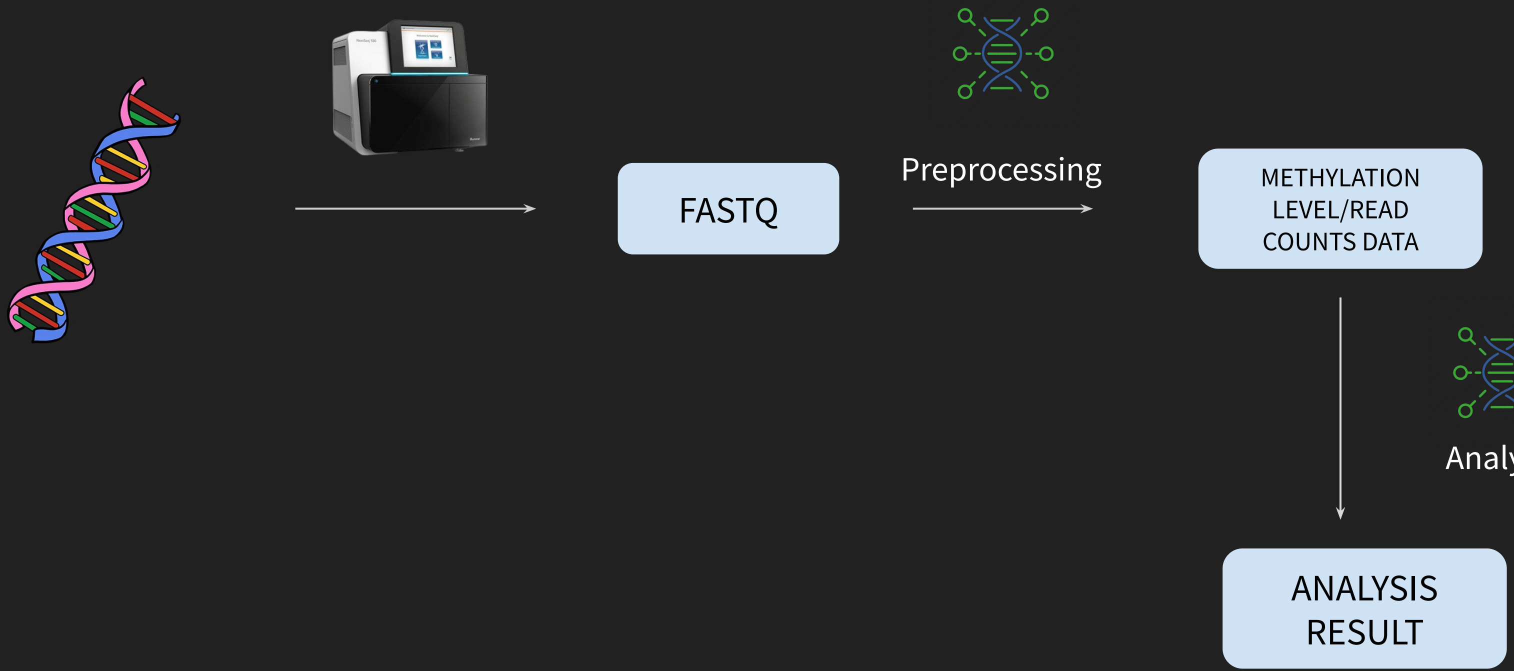
Haries Ramdhani. 2021.

Today’s Outline

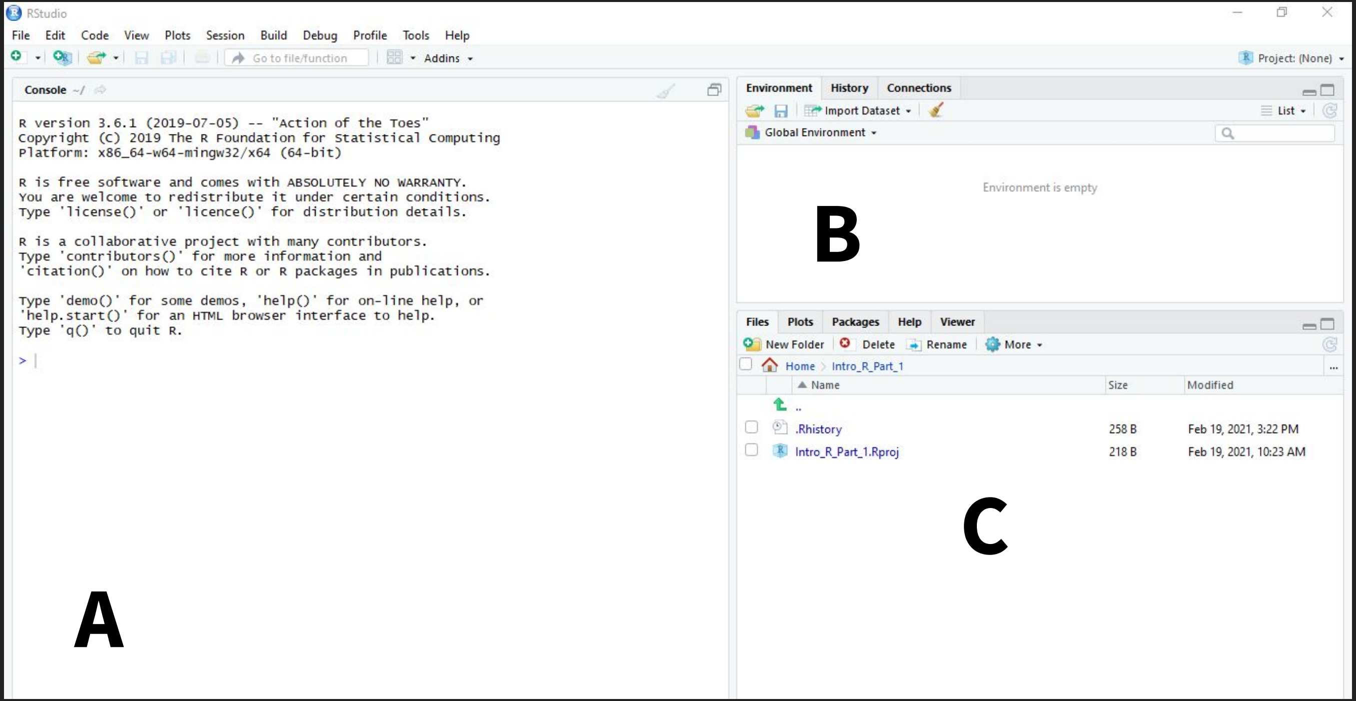
1) Quick Overview of RStudio
2) Programming Fundamentals in R:
    a) R Data Types
    b) R Operations
3) Data Frames and Factors
4) Data Manipulation

A Quick Glimpse

NGS (Bisulfite Sequencing)



Quick Introduction to R Studio

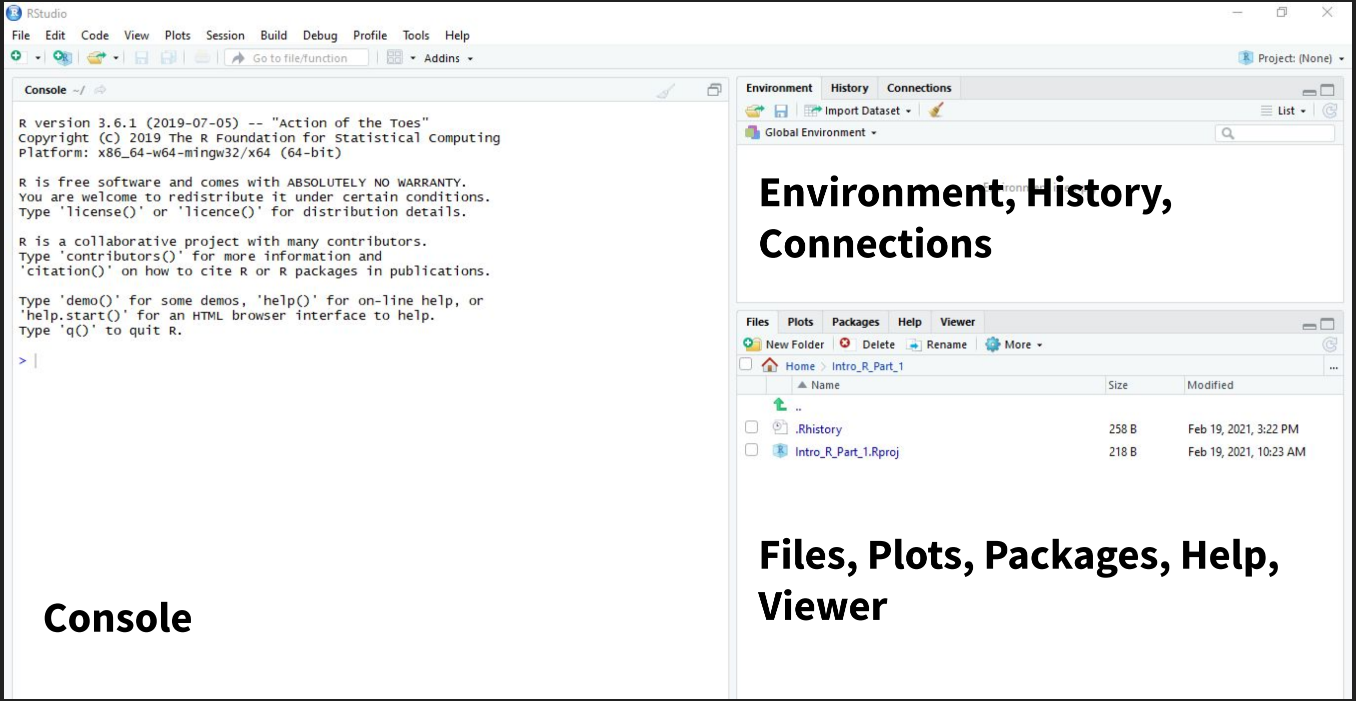

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console ~/
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
B
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home › Intro_R_Part_1
A
C

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

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.

> |
```

Name	Size	Modified
..		
.Rhistory	258 B	Feb 19, 2021, 3:22 PM
Intro_R_Part_1.Rproj	218 B	Feb 19, 2021, 10:23 AM

Figure 1 Display of RStudio

Quick Introduction to R Studio



```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

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.

> 
```

Name	Size	Modified
..		
.Rhistory	258 B	Feb 19, 2021, 3:22 PM
Intro_R_Part_1.Rproj	218 B	Feb 19, 2021, 10:23 AM

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console ~/
Environment History Connections
Import Dataset
List
Global Environment
Environment, History, Connections
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home > Intro_R_Part_1
Console
Files, Plots, Packages, Help, Viewer

Figure 1 Display of RStudio

Quick Introduction to RStudio

A typical RStudio layout contains the following:

- **Console** : to type our code and see the output directly

- **Environment** : Provide detailed list of every functions or variables that are defined
- **History**: Provide list of the history of the used commands

- **Files** : Store all the files that are currently available to user
- **Plots** : Where the charts and plots will appear
- **Packages** : List information about the packages installed
- **Help** : Allow to search the help directly from RStudio

More information: R Studio Layout - St. Olaf College

Creating Objects/Variables in R

Objects stores the value of your choice so you can reuse it over and over again

To create an object you need:

1) a name (e.g. ‘number_of_genes’)
2) a value (e.g. ‘25’)
3) the assignment operator (‘<-‘)

```
> number_of_genes <- 25
> number_of_genes
[1] 25
```

Naming Objects/Variables in R

Objects stores the value of your choice so you can reuse it over and over again

1) **Avoid spaces and special characters**

```
> number_of_genes <- 25 ✅
> number of genes <- 25 ❌
> n@mber_of_g#n#s <- 25 ❌
```

Naming Objects/Variables in R

Objects stores the value of your choice so you can reuse it over and over again

1) **Avoid spaces and special characters**
2) **Use short, easy-to-understand names**
3) **Avoid commonly used names**

```
> number_of_genes <- 25 ✅
> number of genes <- 25 ❌
> n@mber_of_g#n#s <- 25 ❌
> mean <- 25 ❌
```

Data Types in R

Similar to other programming languages R has the following data types

Basic Data Types

- **LOGICAL : TRUE/ FALSE**
- **NUMERIC : 4.0, 5, 1018**
- **CHARACTER : ‘ACGTGAC’, ‘TP53’, ‘This is a text’**

Data Types in R

Similar to other programming languages R has the following data types

Basic Data Types

- **LOGICAL : TRUE/ FALSE**
- **NUMERIC : 4.0, 5, 1018**
- **CHARACTER : ‘ACGTGAC’, ‘TP53’, ‘This is a text’**

Operations in R

We can perform mathematical operations in R

```
> 1 + 1
[1] 2
```

Addition

```
> 190 - 18
[1] 172
```

Subtraction

```
> 5 * 3
[1] 15
```

Multiplication

```
> 8 / 4
[1] 2
```

Division

Operations in R

We can perform mathematical operations in R

`> 1 + 1`
`[1] 2`
	Addition	`> 5 ^ 2`
		`[1] 25`
			Exponent

`> 190 - 18`
`[1] 172`
	Subtraction	`> 20 %% 2`
		`[1] 0`
			Modulus (Remainder of division)

`> 5 * 3`
`[1] 15`
	Multiplication	`> 7 %/% 5`
		`[1] 1`
			Integer Division

`> 8 / 4`
`[1] 2`
	Division		

Operations in R

We can perform logical operations too in R

`> 1 == 1`	**Exactly equal**
`[1] TRUE`	
`> 190 < 18`	**Less than**
`[1] FALSE`	
`> 5 > 3`	**Greater than**
`[1] TRUE`	
`> 8 != 4`	**Not equal to**
`[1] TRUE`	

Operations in R

We can perform logical operations too in R

```
> 1 == 1          Exactly equal
[1] TRUE
> 190 < 18        Less than
[1] FALSE
> 5 > 3           Greater than
[1] TRUE
> 8 != 4          Not equal to
[1] TRUE
```

```
> 1 <= 1                 Less than equal
[1] TRUE
> 190 >= 18              Greater than
[1] TRUE
> (5 > 3) | (3 > 5)      OR
[1] TRUE
> (2 > 1) & (4 < 3)      AND
[1] FALSE
```

Data Types in R

Similar to other programming languages R has the following data types

More Data Types

- **VECTORS**
- **LISTS**
- **MATRICES**
- **ARRAYS**
- **FACTORS**
- **DATA FRAMES**

Data Types in R - VECTORS

Vector is one of the simplest data types in R, everything that is not defined as other data type is vector

```
> c(‘Apple’, ‘Orange’, ‘Mango’)
> c(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Data Types in R - LISTS

List can contain many different types of elements inside it

```
> list(c(2,5,3),21.3,sin)

> list(1, ‘apple’, 2.9099090)
```

Data Types in R - MATRICES

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
> matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
     [,1]  [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
```

Data Types in R - FACTORS

When matrix can only take two dimensions, arrays can be any number of dimension.

```
> factor(c('green','green','yellow','red','red','red','green'))
[1] green  green  yellow red    red    red    green
Levels: green red yellow
```

Data Types in R - DATA FRAMES

Data frames are tabular data objects. It’s different from matrix in a way that each column can contain different modes of data.

```
> data.frame(
      gender = c("Male", "Male","Female"),
      height = c(152, 171.5, 165),
      weight = c(81,93, 78),
      Age = c(42,38,26)
  )
```

```
    gender   height  weight  Age
1   Male     152.0     81     42
2   Male     171.5     93     38
3   Female   165.0     78     26
```

Data Frames

Data frames:

- are tabular data objects
- can contain different types of data inside it
- contain vector of equal length

Data Frames

Data frames:

- are tabular data objects
- can contain different types of data inside it
- contain vector of equal length

```
> data.frame(
      gender = c("Male", "Male","Female"),
      height = c(152, 171.5, 165),
      weight = c(81,93, 78),
      Age = c(42,38,26)
  )
```

```
     gender   height  weight  Age
1    Male     152.0    81     42
2    Male     171.5    93     38
3    Female   165.0    78     26
```

Data Frames

Most of the times we will most likely import our data instead of creating it by ourselves

Data Frames

Most of the times we will most likely import our data instead of creating it by ourselves

Reading data from your table is as simple as the following

```
data_frame_variable <- read.csv(“directory/filename”)
```

Data Frames

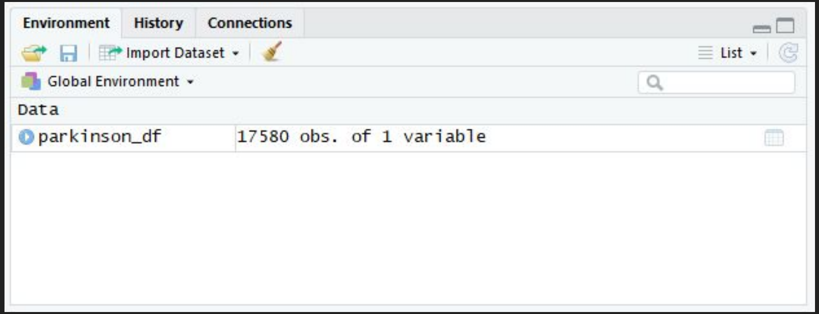
Most of the times we will most likely import our data instead of creating it by ourselves

Reading data from your table is as simple as the following
data_frame_variable <- **read.csv(**“directory/filename”**)**

For example to read the shared data we can perform the following:
parkinson_df <- **readcsv(**“parkinson_de_data.tsv”**)**

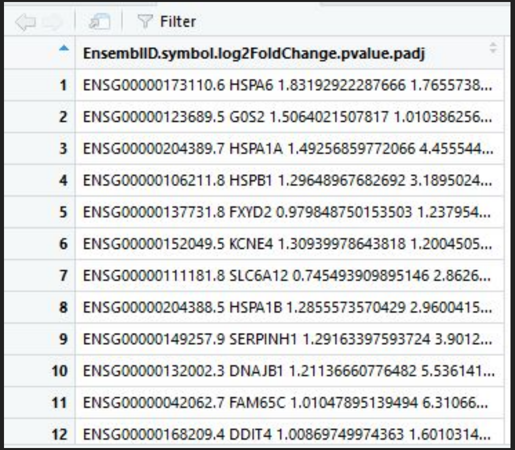
Data Frames

If the import is successful, we can see the variable listed in the environment



```
Data
parkinson_df          17580 obs. of 1 variable
```

Clicking twice we can see our imported data frame



	EnsemblID.symbol.log2FoldChange.pvalue.padj
1	ENSG00000173110.6 HSPA6 1.8319292287666 1.7655738...
2	ENSG00000123689.5 G0S2 1.5064021507817 1.010386256...
3	ENSG00000204389.7 HSPA1A 1.49256859772066 4.455544...
4	ENSG00000106211.8 HSPB1 1.29648967682692 3.1895024...
5	ENSG00000137731.8 FXYD2 0.979848750153503 1.237954...
6	ENSG00000152049.5 KCNE4 1.30939978643818 1.2004505...
7	ENSG00000111181.8 SLC6A12 0.745493909895146 2.8626...
8	ENSG00000204388.5 HSPA1B 1.2855573570429 2.9600415...
9	ENSG00000149257.9 SERPINH1 1.29163397593724 3.9012...
10	ENSG00000132002.3 DNAJB1 1.21136660776482 5.536141...
11	ENSG00000042062.7 FAM65C 1.01047895139494 6.31066...
12	ENSG00000168209.4 DDIT4 1.00869749974363 1.6010314...

But this isn’t what we wanted :(

Data Frames

read.csv can take multiple arguments, the most common ones are:

- ***header*** : whether to use the first line as the header or not
- ***sep*** : what character is being used to separate the columns in the data
- ***dec*** : what character is being used as the decimal point
- ***nrows*** : maximum number of rows to read

Data Frames

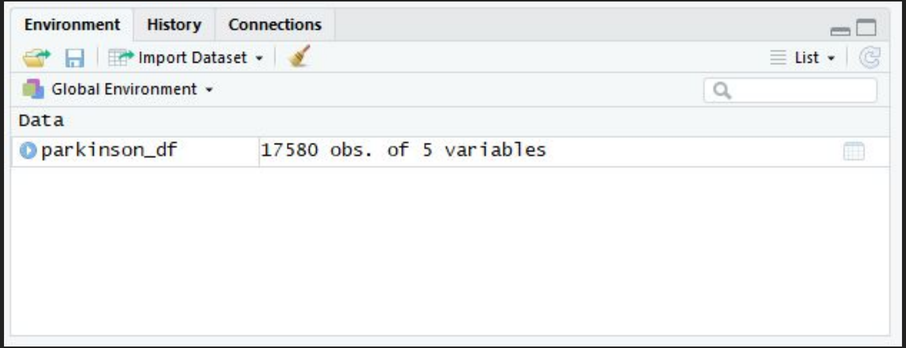
read.csv can take multiple arguments, the most common ones are:

- ***header*** : whether to use the first line as the header or not
- ***sep*** : what character is being used to separate the columns in the data
- ***dec*** : what character is being used as the decimal point
- ***nrows*** : maximum number of rows to read

More information regarding the function can be accessed like usual, just put question mark in front of the function name in RStudio (?**read.csv**)

Data Frames

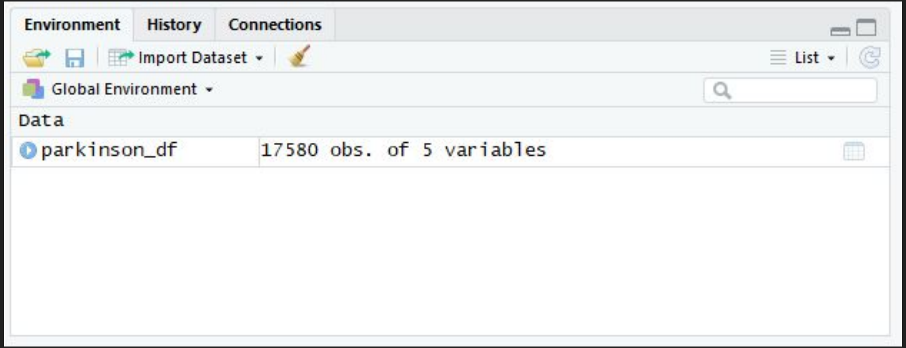
If we read the previous data correctly, the following should appear in our screen:



```
Data
parkinson_df    17580 obs. of 5 variables
```

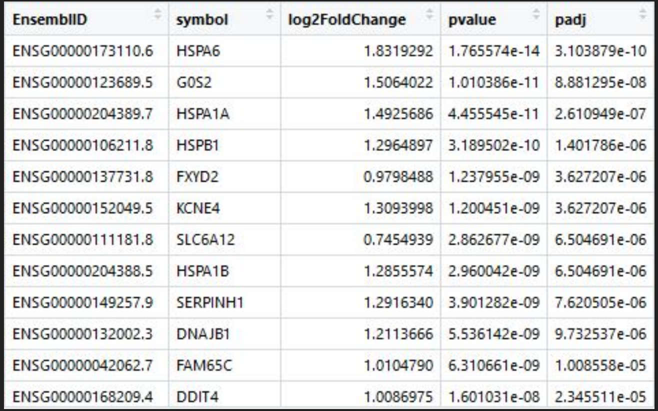
Data Frames

If we read the previous data correctly, the following should appear in our screen:



```
Data
parkinson_df    17580 obs. of 5 variables
```

Clicking the variable twice



EnsemblID	symbol	log2FoldChange	pvalue	padj
ENSG00000173110.6	HSPA6	1.8319292	1.765574e-14	3.103879e-10
ENSG00000123689.5	G0S2	1.5064022	1.010386e-11	8.881295e-08
ENSG00000204389.7	HSPA1A	1.4925686	4.455545e-11	2.610949e-07
ENSG00000106211.8	HSPB1	1.2964897	3.189502e-10	1.401786e-06
ENSG00000137731.8	FXYD2	0.9798488	1.237955e-09	3.627207e-06
ENSG00000152049.5	KCNE4	1.3093998	1.200451e-09	3.627207e-06
ENSG00000111181.8	SLC6A12	0.7454939	2.862677e-09	6.504691e-06
ENSG00000204388.5	HSPA1B	1.2855574	2.960042e-09	6.504691e-06
ENSG00000149257.9	SERPINH1	1.2916340	3.901282e-09	7.620505e-06
ENSG00000132002.3	DNAJB1	1.2113666	5.536142e-09	9.732537e-06
ENSG00000042062.7	FAM65C	1.0104790	6.310661e-09	1.008558e-05
ENSG00000168209.4	DDIT4	1.0086975	1.601031e-08	2.345511e-05

Data Frames

We can examine our data frames using the following functions:

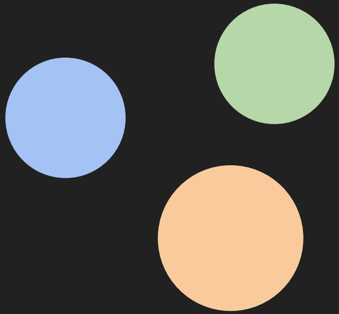
- dim(): shows the dimensions of the data frame by row and column
- str(): shows the structure of the data frame
- summary(): provides summary statistics on the columns of the data frame
- colnames(): shows the name of each column in the data frame
- head(): shows the first 6 rows of the data frame
- tail(): shows the last 6 rows of the data frame
- View(): shows a spreadsheet-like display of the entire data frame

More information : Examine Data Frames in R with These Basic Functions

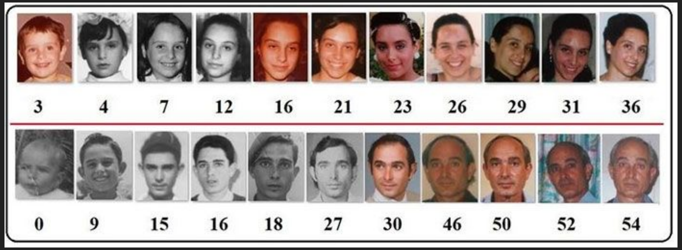
Factors

Factors:

- are usually used to store categorical variables
- are very important when doing statistical modelling
- very efficient when storing character data



Color is categorical



Age is continuous

Subsetting Data Frames

Previously we already talked on how we can perform subsetting in 1 dimensional data (vector) but how do we do it in dataframe?

[Move to practice]

More Function for Data Frames

There are also tremendous amounts of data frame function that we haven't explored, here are the followings:

- Aggregate functions [max(), min(), avg(), sum()]
- Ordering functions [order()]
- Rename columns

Saving data frames:

write.csv(variable, file=filename**)**

DPLYR

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges

Previous subsets [ ] $\longrightarrow$

- select()
- filter()
- group_by()
- summarize()
- mutate()

Installing and using DPLYR (and other packages)

Installing is simple, you just have to type the following
install.packages("name_of_the_packages")

Installing and using DPLYR (and other packages)

Installing is simple, you just have to type the following

```
install.packages("name_of_the_packages")
```

Installing packages from BioConductor is a bit different

```
if (!requireNamespace('BiocManager', quietly = TRUE))
        install.packages('BiocManager')

BiocManager::install("minfi")
```

Installing and using DPLYR (and other packages)

Installing is simple, you just have to type the following

install.packages("name_of_the_packages")

Installing packages from BioConductor is a bit different

```
if (!requireNamespace('BiocManager', quietly = TRUE))
    install.packages('BiocManager')

BiocManager::install("minfi")
```

To load the installed library we can use

library("package_name")

DPLYR

Common functions in DPLYR

- select() : select our intended column
- filter() : filter the data by some conditions
- group_by() : group the data by column
- summarize() : summarize the results of grouping
- mutate() : mutate a column, create new column with new logic