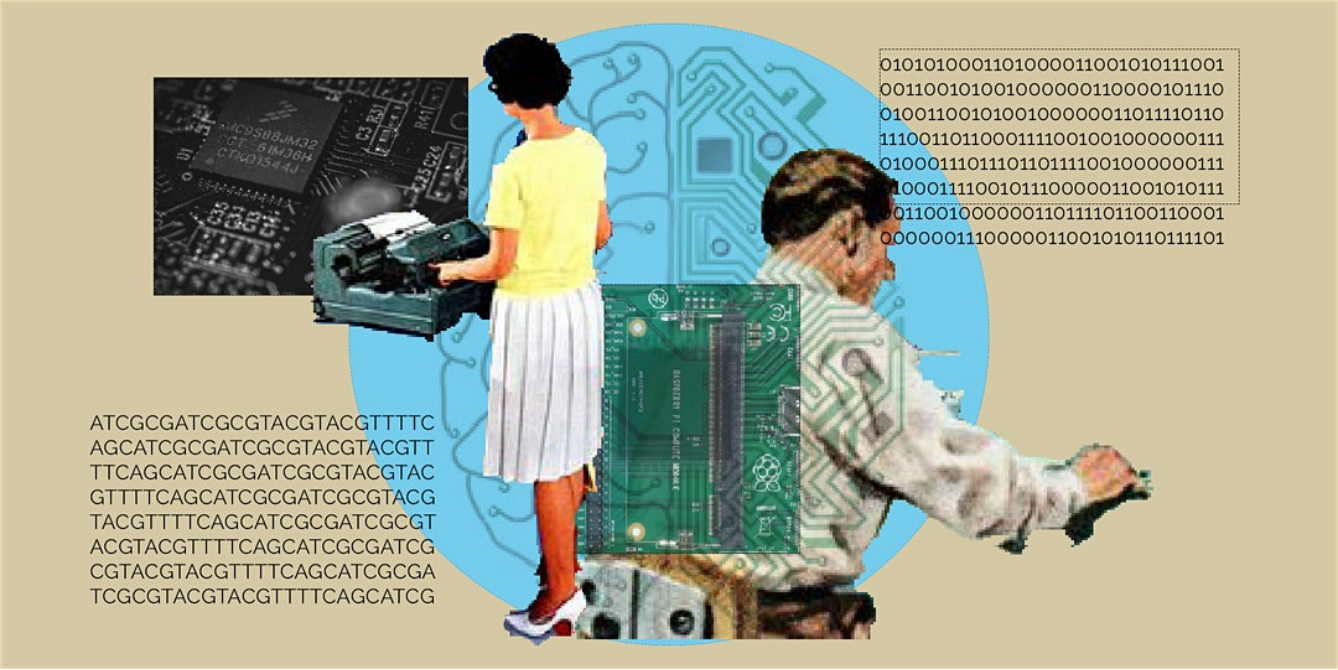
Genspace

Mining big data and demystifying the genome: learn R

April 18-19, 2020



Doing something more fun!

- Returning to our Hemoglobin Example!
- What question did you answer when you explored those hemoglobin beta gene sequences?
- As a curious bioinformatician, is this answer satisfying to you?
- If not, what is a more interesting question?
- Where is the mutation? (At what basepair?)
- What is the mutation? (What should the base be? What is it in the mutant sequence?)
- Is there a consequence of the mutation? (Molecular biology refresher: how do mRNAs get decoded into proteins?)

Make a plan before starting

- Before we get started on a more complicated task, it can help to write the steps you will have to accomplish to get a task done.
- This is called pseudocoding
- So for example, pseudo code of your script where you played around with the object species_vector might look like this

Create a vector called animal_vector. Add the names of a few species to it

Add 2 more items to the animal_vector object

Calculate how many elements are in the animal_vector object

Remove the last element in animal_vector

Calculate how many elements are in the animal_vector object

Make a vector with the elements of animal_vector that are vertebrates

Make a vector with the elements of animal_vector that are invertebrates

It is okay if I don't know how I am going to accomplish a particular step.

By breaking them down into small chunks, it will be easy to work one step at a time

Make some pseudocode in Rstudio that will help you answer these questions:

- Where is the mutation? (At what basepair?)
- What is the mutation? (What should the base be? What is it in the mutant sequence?)
- Is there a consequence of the mutation? (Molecular biology refresher: how do mRNAs get decoded into proteins?) [** This one is a little more complicated, let's ignore for now]

Tips:

Open a new .R file in RStudio

File → New File → R Script

If you start a line with a hashtag #, you are telling R "Don't run this line"

We call these line ***comments*** this is to help you remember what your code does.

If you are working with other people, comments are really helpful for them to understand how your code works

Write your pseudocode in your editor panel in RStudio and comment each line out by adding a # to the beginning of every line

Let's come to a consensus on our pseudo code

Now let's tackle it:

- In one of our steps we are asking a question:
- Is this base from the variant sequence the same as the base in the ref sequence
- And we want to different things depending on the answer
- We need a ***conditional*** statement
- Here is the syntax of an ***if*** statement in R:

```
if (expression_that_generates_a_boolean){
   ****Do a task****
}
```

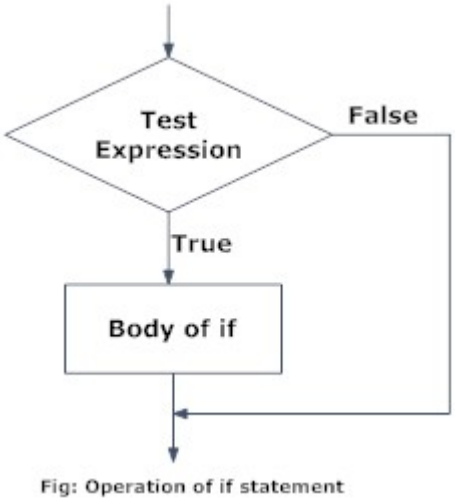


Fig: Operation of if statement

- What is one way you know to generate a Boolean?
- The logical operator ==
- You may want to use !=

Use an if statement to check if base 1 of the variant is not equal to base 1 of the ref sequence

- Currently, we have saved our sequences as strings (so we can't just use the indexing rules that we learned for vectors)
- I did this google search "return part of a string in R"
- The first hit shows me that I can use this function:

substr(x, start, stop)

The documentation tells me that the argument x should be a string

Start and stop are like indices in a vector. (positions of the part of the string you want to return)

Try:

substr(hbb_ref,1,2)

substr(hbb_ref,2,2)

So, what **expression_that_generates_a_Boolean** could you use in this if statement to answer the question of whether these two bases are the same?

```
if (expression_that_generates_a_boolean){
   ****Do a task****
}
```

Use an if statement to check if base 1 of the variant is not equal to base 1 of the ref sequence

```
if (substr(hbb_var1,1,1) != substr(hbb_ref,1,1){
   print("Mismatch")
}
```

And we need to do that for every single base:

- We have a repetitive task
- We can use a ***loop***
- Do we know exactly how many times we need to do this task?
- Yes
- We need a ***for*** loop
- a ***For*** loop repeats a task a set number of times
- Here is the syntax of a for loop in R:

```
for(element in vector){
   ****Do a task****
}
```

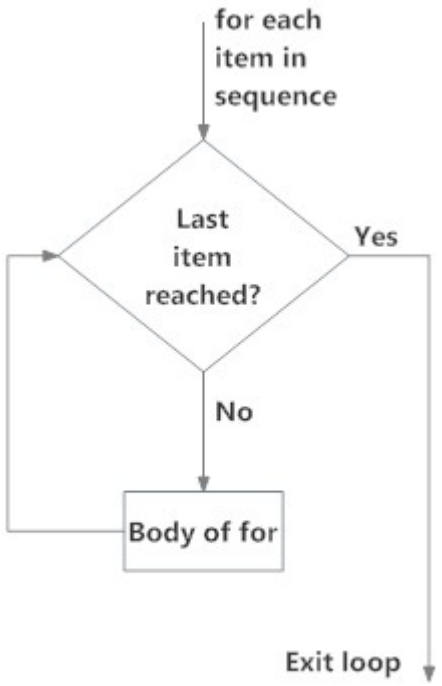


Fig: operation of for loop

Try this code:

```
for(animal in animal_vector){
  print(animal)
}
print("Done with the loop")
```

- A for loop deconstructs a vector
- It takes the first element out of the vector and sets it equal to a variable (in this example you name this variable **animal)**
- The for loop does something (in this case, it prints **animal**)
- Then it takes the next element from the vector and sets it equal to the variable **animal**
- Then it does something again

{this repeats until there are no more elements in the vector}

- At this point the for loop is completed and R reads the next line of code below

How could you construct a for loop that gives the bases of each?

- Hint:

```
> substr(hbb_ref,1,1)
[1] "A"
> substr(hbb_ref,2,2)
[1] "T"
```

- We want a for loop where the element that is returned to us in each round is the number/position of the base we want to look at:

```
for(index in vector_that_has_every_index){
  print(substr(hbb_1,index,index))
  print(substr(hbb_ref,index,index))
}
```

- How do we get **`vector_that_has_every_index`**?
- Try these pieces of code in RStudio:

```
1:10
num = 5
1:num
nchar(hbb_ref)
```

How could you construct a for loop that gives the bases of each?

- Explain what is happening with these lines of code

```
1:10
num = 5
1:num
nchar(hbb_ref)
```

- How do we get **`vector_that_has_every_index`**?

```
num_bases = nchar(hbb_ref)
1:num_bases
or
1:nchar(hbb_ref)
```

- The for loop will look like this

```
for(index in 1:nchar(hbb_ref)){
  print(substr(hbb_1,index,index))
  print(substr(hbb_ref,index,index))
}
```

Now we need to integrate our if statement with our for loop

Objects – Of increasing complexity!

- `x  = "cat"` (zero dimensions)
- `x = c("dog", "cat", "mouse", "platypus")` → this is a vector, (1 dimension)
  - What do I mean by 1 dimension?
  - If you typed this:

    ```
    x[2]
    "cat"
    ```

  - By providing this one index value, you are able to find where the "cat" is within the vector

- But often an object with two dimensions is really useful
- What would two dimensional object look like?

R is complicated!

- There are many ways to accomplish a single task in R
- There are multiple different types of objects that serve this purpose
  - Matrix
  - Data frame
  - Tibbles
- And the data is more or less in a similar format
- 2 dimensional object.
- How do you think you would grab the piece of data stored in a specific cell?

	A	B	C	D	E
1	Patient_ID	subtype	A1BG	A1CF	A2BP1
2	TCGA-02-0047-01A-01R-1849-01	Proneural	125.0069	0	244.6295
3	TCGA-02-0055-01A-01R-1849-01	Mesenchymal	391.8038	0	137.3511
4	TCGA-02-2483-01A-01R-1849-01	Proneural	271.8522	0	111.029
5	TCGA-02-2485-01A-01R-1849-01	Classical	83.9429	0	257.1429
6	TCGA-02-2486-01A-01R-1849-01	Mesenchymal	108.2561	0	4.2683
7	TCGA-06-0125-01A-01R-1849-01	Classical	201.6663	0	52.5565
8	TCGA-06-0125-02A-11R-2005-01	Classical	168.104	0	318.8557
9	TCGA-06-0129-01A-01R-1849-01	Proneural	133.7468	0.412	18.1301
10	TCGA-06-0130-01A-01R-1849-01	Mesenchymal	152.8932	0	90.2527
11	TCGA-06-0132-01A-02R-1849-01	Neural	129.7286	0	973.462
12	TCGA-06-0138-01A-02R-1849-01	Neural	250.8639	0	125.7775
13	TCGA-06-0141-01A-01R-1849-01	Mesenchymal	427.3409	0	66.2218
14	TCGA-06-0152-02A-01R-2005-01	Mesenchymal	30.7533	0	214.4458
15	TCGA-06-0156-01A-03R-1849-01	Proneural	392.9522	0	25.676
16	TCGA-06-0157-01A-01R-1849-01	Classical	49.3148	0	51.7221

data_frame[row#,column#]

Let's look at Data Frames

- When you download R, you download some example data sets that you can play around with
- A couple useful ones are called:
- iris
- mtcars

You can access these objects by typing **mtcars** or **iris** and hitting enter

What class of object are these? (Test this)

Let's look at Data Frames

- An important thing to know about a data frame is how big it is: How many columns, how many rows

- To find this out you could Google: "How many columns and rows in a data frame in R"
- ncol(), nrow(), and dim()

Let's look at Data Frames

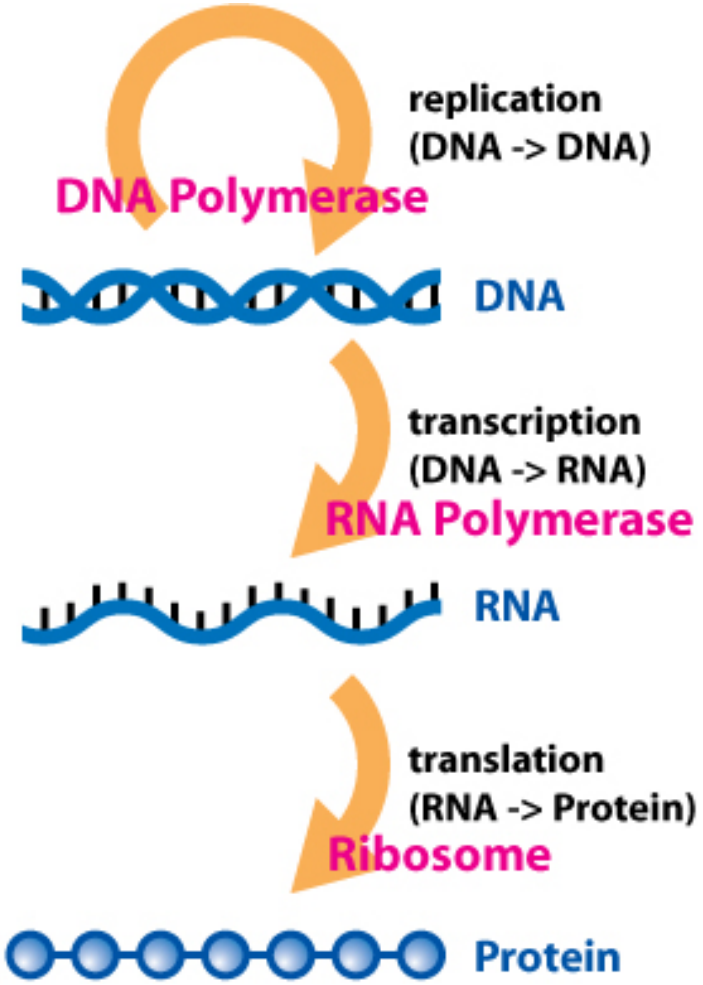
- How would you access the 1st column of the data frame iris?
- Remember: dataframe[row#,column#]
- iris[,1]
- There is another way to do this too!
- If the column has a name, you can access it like this:
- data_frame$column_name
- How do you know what the names of the columns are?
- Grab the 3rd column in the data frame iris and set it equal to a variable called petal_len
- What class of object is petal_len?
- Does this look familiar?
- Back to vectors!

Let's look at Data Frames

- Imagine you are publishing a paper about iris petals
- What is something you might want to do with this raw data?
- Calculate the mean length?
- mean() - search for this function in
- What was the function?
- What arguments did it take? What class of object did the arguments need to be?
- What else might you want to calculate?
- Standard Deviation, Median, Max?

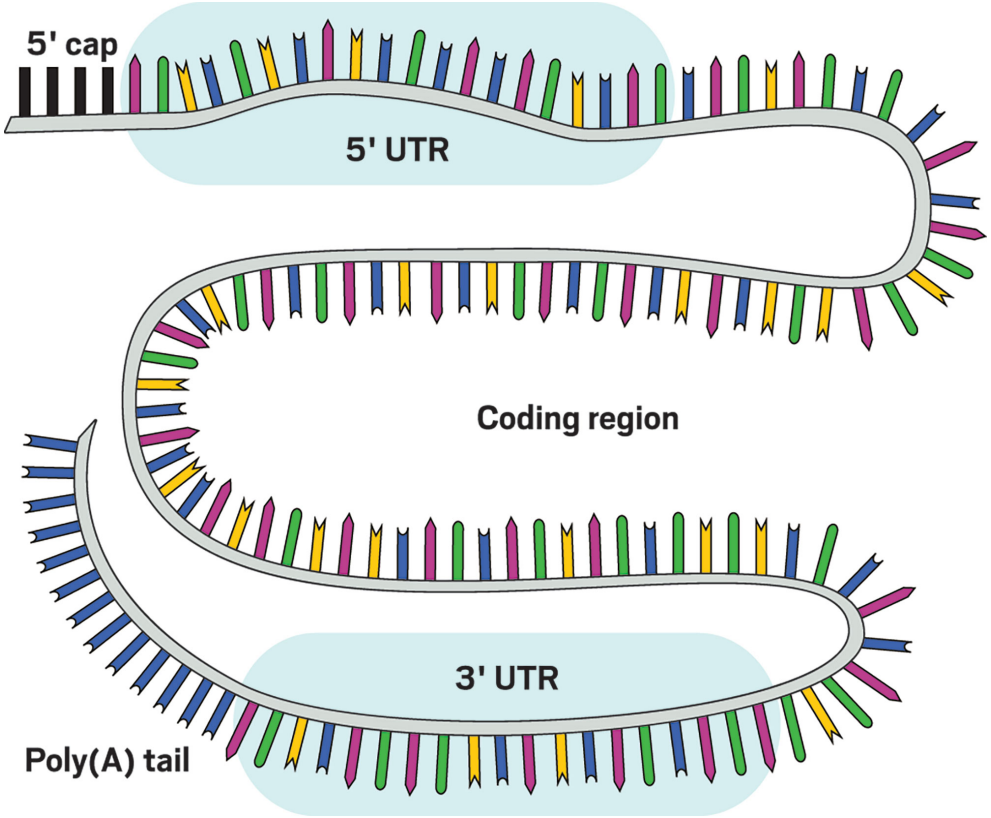
RNA seq interlude

- Central dogma of molecular biology



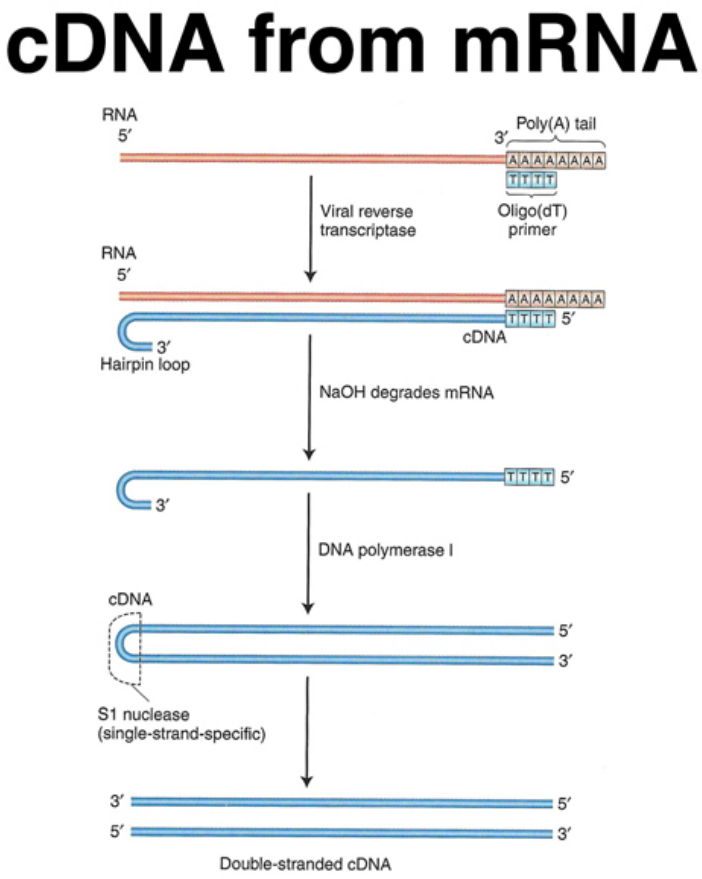
mRNAs (from Eukaryotes at least) are usually polyadenylated

- You can take advantage of this feature to selectively grab mRNAs out of a mixture of nucleic acid



You can make cDNA libraries from your mRNAs

- cDNA = complementary DNA
- These are DNA now, and you can sequence using an Illumina sequencer



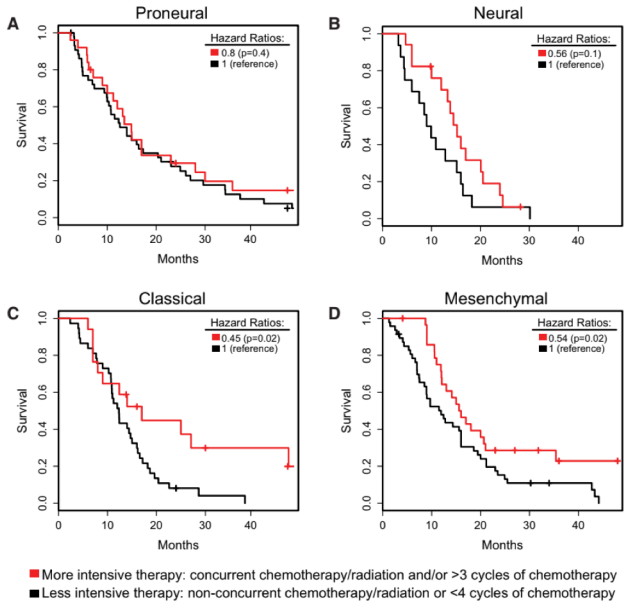
Talk about the paper

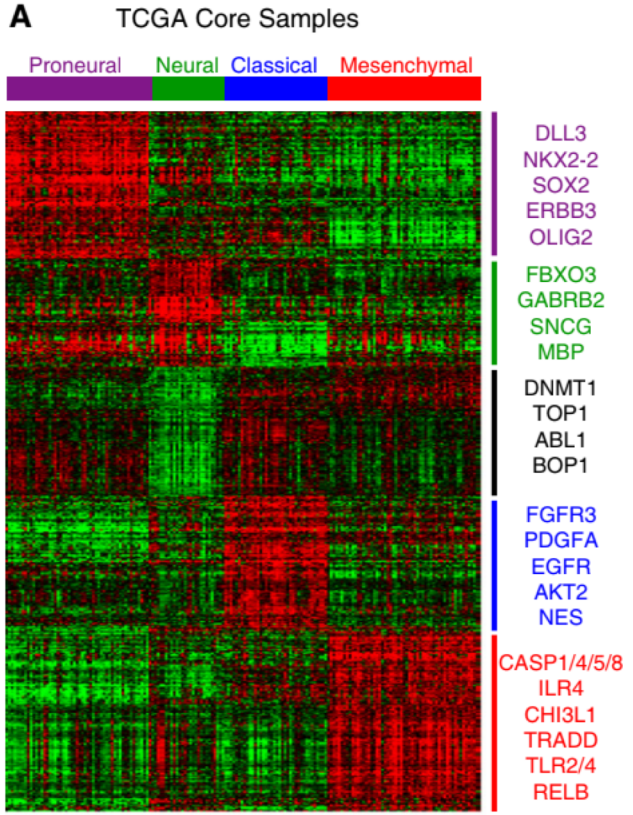
Cell PRESS

Cancer Cell
Article

Integrated Genomic Analysis Identifies Clinically Relevant Subtypes of Glioblastoma Characterized by Abnormalities in *PDGFRA*, *IDH1*, *EGFR*, and *NF1*

Roel G.W. Verhaak,[1,2,17] Katherine A. Hoadley,[3,4,17] Elizabeth Purdom,[7] Victoria Wang,[8] Yuan Qi,[4,5] Matthew D. Wilkerson,[4,5] C. Ryan Miller,[4,6] Li Ding,[9] Todd Golub,[1,10] Jill P. Mesirov,[1] Gabriele Alexe,[1] Michael Lawrence,[1,2] Michael O'Kelly,[1,2] Pablo Tamayo,[1] Barbara A. Weir,[1,2] Stacey Gabriel,[1] Wendy Winckler,[1,2] Supriya Gupta,[1] Lakshmi Jakkula,[11] Heidi S. Feiler,[11] J. Graeme Hodgson,[12] C. David James,[12] Jann N. Sarkaria,[13] Cameron Brennan,[14] Ari Kahn,[15] Paul T. Spellman,[11] Richard K. Wilson,[9] Terence P. Speed,[7,16] Joe W. Gray,[11] Matthew Meyerson,[1,2] Gad Getz,[1] Charles M. Perou,[3,4,8] D. Neil Hayes,[4,5,*] and The Cancer Genome Atlas Research Network





Packages

- R comes with many built in useful functions
- Things like print(), length(), class(), mean(), max(), ncol(), etc.
- But sometimes you will have a very specific task and you may need to write your own new function
- You can do this!
- Also, because the R community is so large, there are many users who have written functions that may be useful people who do similar types of work as them
- They put these functions they have written in a thing called a package, which you can download from the internet, and load into R

Probably the most popular package in R is called **ggplot**

- We're going to download it
- Load it in R
- And use it to make graphics from the Glioblastoma dataset from the Verhaak et al 2010 paper

Try typing this into the RStudio console:

```
ggplot()
```

```
install.packages("ggplot2")
library("ggplot2")
```

Now try typing this into the RStudio console again:

```
ggplot()
```

Probably the most popular package in R is called **ggplot**

- Find "R Resources" in the google docs folder
- Open the link that says "r-graphics.org"
- This is a great resource
- It will tell you exactly how to make almost any kind of plot that you would want to make using ggplot in R

To look at the data from the paper

- We need to upload the data file in R
- It is called "glioblastoma_rna_seq.csv"
- (Open it in your text editor)
- It's too big to copy and paste
- How do we import data into R?
- Functions
- Google "How to import a csv file in R"
- read.csv()
- When you read the documentation you will see that the syntax should be:

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
dec = ".", fill = TRUE, comment.char = "", ...)
```

```
read.csv(file)
```

Where file is the path to our file

The path is the address of where something is living in the hard drive of your computer

To look at the data from the paper

- To make things easy. Save the "glioblastoma_rna_seq.csv" file in the same folder where your R project is saved

```
read.csv("glioblastoma_rna_seq.csv")
```

- This should work
- But you can open data that is saved elsewhere, but you will need the absolute path (the full address of the file)

```
read.csv("C:\\Users\\benja\\Documents\\Genspace_PC\\R_Course_Working\\glioblastoma_rna_seq.csv")
```

(Something like this on a PC)

```
read.csv("/Users/kingb08/Dropbox/Genspace/20200418_Workshop/glioblastoma_rna_seq.csv")
```

(Something like this on a Mac)

** Make sure to save it to a variable so that the data is now saved in memory

```
rna = read.csv("glioblastoma_rna_seq.csv")
```