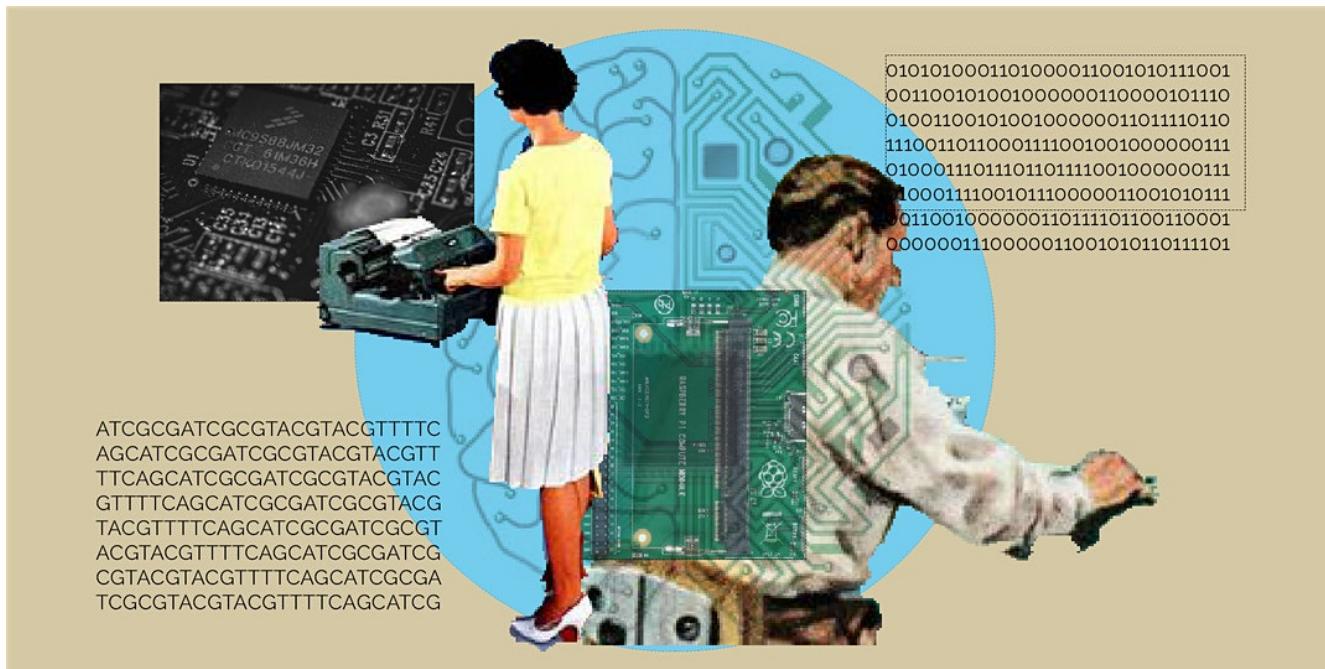


Mining big data and demystifying the genome: learn R

April 18-19, 2020



My background

I got a bachelor of arts in Biology

I got a Ph.D. in Cell and Molecular Biology

I don't have a formal computer science background

I had one bioinformatics course in grad school (we learned Ruby)

I taught myself R

Now, I'm a postdoc at NYU School of Medicine

Now, I use R everyday to analyze RNA sequencing data

We're trying to understand why some ovarian cancer cells can become resistant to chemotherapeutic drugs

Computational Biology/Bioinformatics is an interesting field because a lot of us are largely self taught (this is my impression at least)

RStudio should look something like this:

What are these different panels?

The screenshot shows the RStudio interface with the following panels:

- Console**: The main workspace where R code is entered and executed.
- Global Environment**: A pane showing the current objects in memory. It displays the message: "Global Environment" and "Environment is empty". Below this, it says "(As you save objects to memory, they will show up here)".
- Files**: A file browser showing the project directory structure: C:/Users/benja/Dropbox/Genspace/Project2. The contents of the folder are listed below, including "Project2.Rproj".

Console
(This is the interactive Input/Output interface with R)

Global Environment
(As you save objects to memory, they will show up here)

Files

Name	Size	Modified
Project2.Rproj	218 B	Dec 8, 2019, 1:22 PM

There are several interesting things here!

- Files
- Plots
- Packages
- Help

(All of these will be very useful)

Jargon! Oh lord

The screenshot shows the RStudio interface with the following components:

- Top Bar:** Shows the path "C:/Users/benja/Dropbox/Genspace/Project2 - RStudio" and standard menu items: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Console Tab:** Active tab, showing the path "C:/Users/benja/Dropbox/Genspace/Project2/".
- Global Environment Tab:** Shows the heading "Global Environment" and the subtext "(As you save objects to memory, they will show up here)".
- Files Tab:** Shows a file named "Project2.Rproj" with a size of 218 B and a modified date of Dec 8, 2019, 1:22 PM.

Console Section:

Console

(This is the interactive Input/Output interface with R)

Global Environment Section:

Global Environment
(As you save objects to memory, they will show up here)

Files Section:

Name	Size	Modified
Project2.Rproj	218 B	Dec 8, 2019, 1:22 PM

Text Overlay:

There are several interesting things here!

- Files
- Plots (All of these will be very useful)
- Packages
- Help

- There is a specific vocabulary used by computer scientists
- Stop me when I use jargon! Let me explain!
- It is helpful to learn some of this jargon so we can communicate with others

Input/Output and the Console Window

- Back to RStudio

Input/Output and the Console Window

The screenshot shows the RStudio interface with the following components visible:

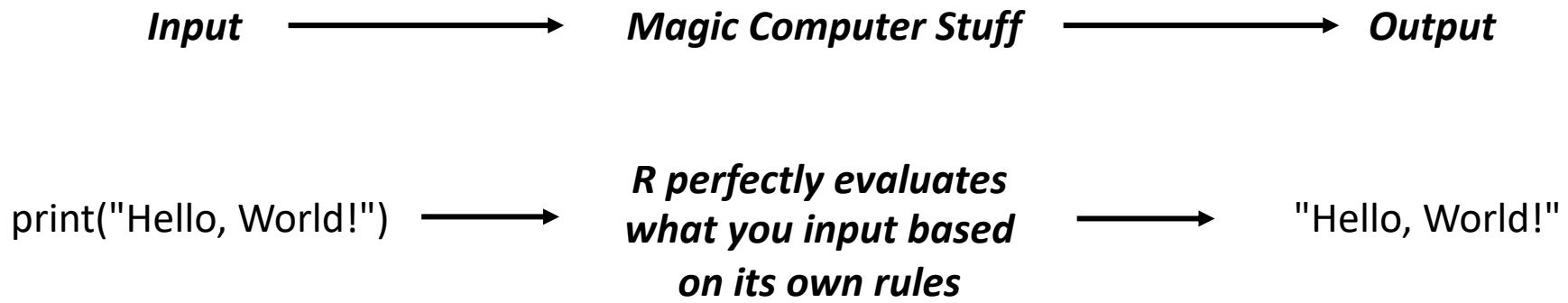
- Top Bar:** Shows the path "C:/Users/benja/Dropbox/Genspace/R_Course_Working - RStudio" and the standard RStudio menu bar: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** An "Untitled1" script editor window with a toolbar above it. The status bar indicates "1:1 (Top Level) R Script".
- Console Window:** The main workspace where R code is run. It contains the following text:

```
C:/Users/benja/Dropbox/Genspace/R_Course_Working/ 
> print("Hello, World!")
[1] "Hello, World!"
```

A black arrow points from the word "Input" to the command `print("Hello, World!")`. Another black arrow points from the word "Output" to the resulting output `[1] "Hello, World!"`.
- Environment Tab:** Shows the Global Environment tab with the message "Environment is empty".
- File Explorer:** Shows the file structure under "C:/":

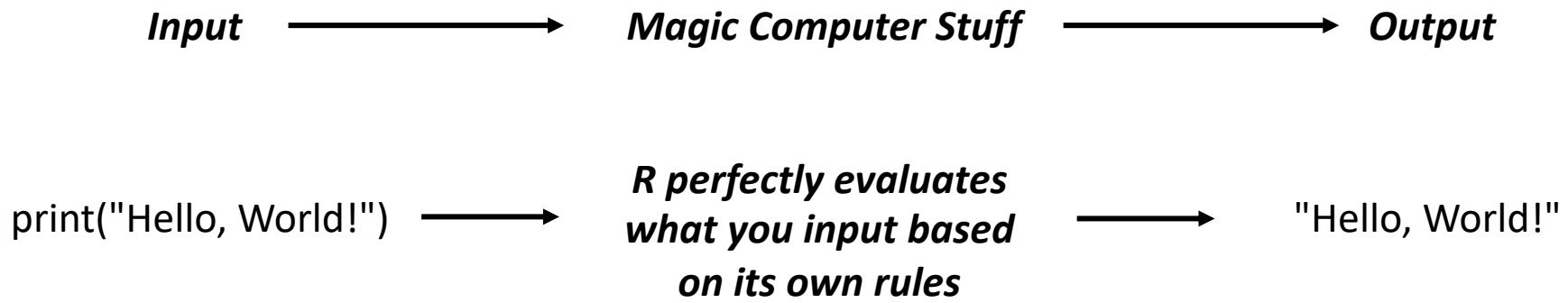
Name	Size	Modified
R_Course_Working.Rproj	218 B	Dec 7, 2019, 12:40 PM

What is happening here?



The rules that R (or any coding language, for that matter) uses to "read" the code you have given it, is called syntax

What is happening here?



The rules that R (or any coding language, for that matter) uses to "read" the **code** you have given it, is called **syntax**

code is like a recipe that the computer reads.

Talk like a computer scientist! Jargon interlude:

The rules that R (or any coding language, for that matter) uses to "read" the **code** you have given it, is called **syntax**

code is like a recipe that you (the user) have written. The computer reads this recipe. It follows each step in the code until it reaches the end

code is synonymous with **script**

You could tell your boss/professor/teacher, "Hey, look at the R code that I wrote! This script lets me look for mutations in the genomes of cancer cells"

If you want R to do what you want, you need to know R's **syntax!**

- Syntax in computer science has the same meaning as syntax in our everyday life:
- What is syntax?
- How do you "decode" the phrases:

"Eats, shoots & leaves"

versus

"Eats shoots & leaves"

"You don't need to be a grammar nerd to enjoy this one....Who knew grammar could be so much fun?"—*Newsweek*

The #1 New York Times Bestseller

Eats, Shoots & Leaves



The Zero Tolerance
Approach to Punctuation

! LYNNE TRUSS

With a Foreword by Frank McCourt,
author of *Angela's Ashes*

Title: Syntax is important

Subtitle: Computers are not humans

- Back to RStudio!
- In the **console** (bottom left), try typing the following lines of code

First,

print("Hello, World!") {press enter}

Then,

print "Hello, World!" {press enter}

Then,

"print Hello, World!" {press enter}

Then,

print('Hello, World!') {press enter}

Then,

print(Hello, World!) {press enter}

Last,

HelloWorld = "Hello, World!" {press enter}
print(HelloWorld) {press enter}

What happened? [Report back to the class!]

```
print("Hello, World!") [1] "Hello, World!"
```

```
print "Hello, World!" Error: unexpected string constant in "print "Hello, World!""
```

```
"print Hello, World!" [1] "print Hello, World!"
```

```
print('Hello, World!') [1] "Hello, World!"
```

```
print(Hello, World!) Error: unexpected '!' in "print(Hello, World!"
```

```
HelloWorld = "Hello, World!" [1] "Hello, World!"  
print(HelloWorld)
```

Talk to your partner(s): Explain this behavior (2 minutes)

```
print("Hello, World!") [1] "Hello, World!"
```

```
print "Hello, World!" Error: unexpected string constant in "print "Hello, World!""
```

```
"print Hello, World!" [1] "print Hello, World!"
```

```
print('Hello, World!') [1] "Hello, World!"
```

```
print(Hello, World!) Error: unexpected '!' in "print(Hello, World!)"
```

```
HelloWorld = "Hello, World!" [1] "Hello, World!"  
print(HelloWorld)
```

There are different types of **objects** in R

- When you write code, everything you write is an **object**
- Every type of **object** is special
- The syntax of your code tells R what type of **object** it is
- R treats different **objects** in different ways

Based on your brainstorming with your partner(s),

To make this code work correctly, what are the important things R is looking for? (Type into Group Chat)

In this example,
R knows what to do because of

- Quotation marks " " or ‘ ’
- Parentheses ()
- Equals signs =

How does R interpret each of these?

You got it!

"Hello, World!"

Here, the quotation marks are telling R that the stuff inside is a **String**

print(__)

Here, the parentheses are telling R that print is a **Function**

HelloWorld

The lack of quotation marks tells R that this is a **Variable**

HelloWorld = "Hello, World!"

The equals sign is telling R to set this **variable** to equal this **string**.
Cool thing here, this **variable** is stored in the computer's memory.
You can access this **string** again by telling R to retrieve that **variable**

*** this **variable** is now stored in R's **Global Environment**. You can now see this in the "Global Environment" panel in the upper right of Rstudio

R is flexible.
There are many ways to do something right.

```
print("Hello, World!") → [1] "Hello, World!"
```

```
HelloWorld = "Hello, World!" → [1] "Hello, World!"  
print(HelloWorld)
```

But syntax is important, and there are many ways
something can go wrong

```
print "Hello, World!" → Error: unexpected string constant in "print "Hello, World!""
```

```
"print Hello, World!" → [1] "print Hello, World!"
```

```
print(Hello, World!) → Error: unexpected '!' in "print(Hello, World!)"
```

When you code, most of your time will be spent **debugging**

OMG, my code has a **Bug**:



`print "Hello, World!" → Error: unexpected string constant in "print "Hello, World!""`

You run some code. And it gives an error!

`"print Hello, World!" → [1] "print Hello, World!"`

You run some code. And the output isn't what you want!

When you code, most of your time will be spent **debugging**

`print "Hello, World!" → Error: unexpected string constant in "print "Hello, World!""`

OR

`"print Hello, World!" → [1] "print Hello, World!"`

This is **debugging**

- The process of figuring out why your code isn't doing what you think it should
- And changing it so that it does

OMG, my code has a **bug**:

- Am I stupid?

Lol, No. The vast majority of everyone's time is spent debugging

`print("Hello, World!") → [1] "Hello, World!"`

Some important types of objects

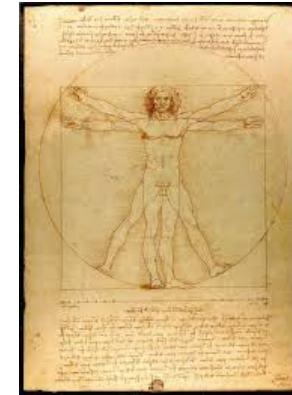
Strings

Examples:

"cat"

"female"

"Homo sapiens"



Numbers

Examples:

0

1

0.25

0.333333333333

1000

10^{32}

10^{-18}

-3

1	2	1	0	1	7	4	5	8	1
4	3	1	9	0	3	2	4	5	8
2	9	8	4	1	9	1	4	1	4
5	2	9	8	4	5	8	0	8	5
2	4	3	6	2	7	3	9	0	2
9	2	6	2	7	3	9	0	2	1
6	1	5	0	8	2	1	7	8	0
2	7	9	5	0	2	1	3	9	3
1	6	7	5	0	1	3	9	7	1
8	4	1	3	1	0	1	4	8	7
9	5	9	8	7	2	8	1	9	8
8	4	1	3	1	0	1	4	8	7
5	9	8	7	2	8	1	4	8	7
3	2	1	4	8	1	4	8	1	4
6	8	4	1	3	1	0	1	4	8

Character and Numeric Classes

- You can ask R what kind of object something is
(and you will often want to do this as your code gets more complicated)
- Back to RStudio
- Enter some (or all) of the following into the console of Rstudio:

```
class("cat")
class("print")
class(print)
class(1)
class(0.3)
class(-50)
class("1")
```

class() is a ***function***:

- A ***function*** always has parentheses
- The stuff inside its parentheses are called ***arguments***
- A ***function*** is this little black box. It grabs the ***arguments*** that you give it, it does something behind the scenes, and gives you something else
- We'll talk more about this, so don't worry about it now
- class() is looking at the object you give it, determining what kind of an object it is, and then it tells you what kind object it is

Character and Numeric Classes

- You can ask R what kind of object something is
(and you will often want to do this as your code gets more complicated)
- Back to RStudio
- Enter some (or all) of the following into the console of Rstudio:

```
class("cat")
class("print")
class(print)
class(1)
class(0.3)
class(-50)
class("1")
```

What outputs did you get?

[1] "numeric"

[1] "function"

[1] "character"

Anything unexpected?

```
> class("1")
[1] "character"
```

What can you do with Numeric and Character Classes?

- Back to RStudio
- Try typing some of the following into the console of Rstudio
(the numbers and words are arbitrary, use your own examples too)

```
1 + 1
1 - 1
2 / 4
2 / 3
2 * 0.5
2^2
4^(1/2)
```

```
"cat" + "dog"
"cat" * 3
"cat" / "dog"
"1" + "2"
"300" / "10"
"5" + 3
```

- What behavior did you observe?
- How do you explain this?

Character and Numeric Classes

- You can do math on objects that are Numeric
- You can't really do math on objects that are Character
- If you tell R to work with "1", it will make no assumptions. It will not know it is a number.
- $1 \neq "1"$

Booleans: Another cool class of object!

- Does anyone know what a ***Boolean*** is?

TRUE or FALSE

(These are the only two options!)

- RStudio – try this code (and/or swap in your own objects):

```
1 == 1  
1 == 1.0  
1 == 1.1  
"cat" == "cat"  
"cat" == "CAT"
```

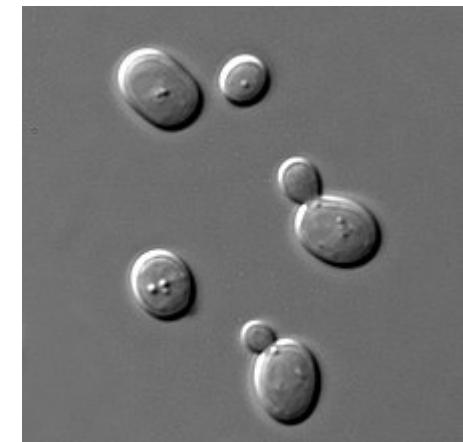
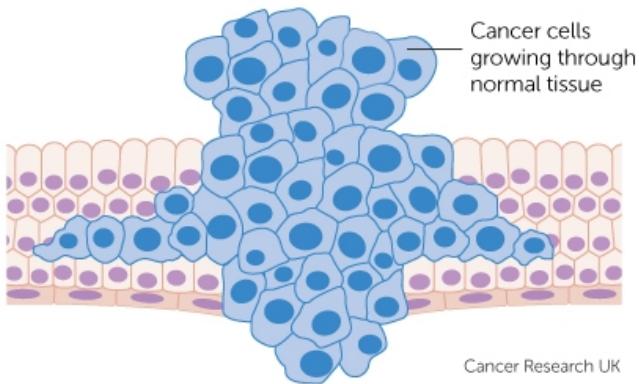
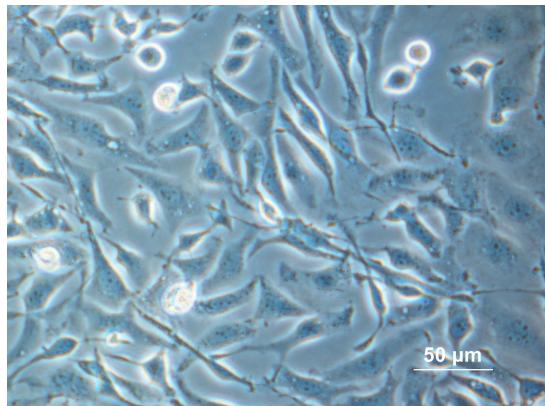
What do you think `==` does?

`==` is what is called a ***logical operator***
(There are other logical operators too)

`==` is a way to ask if the first thing is the same as the second thing

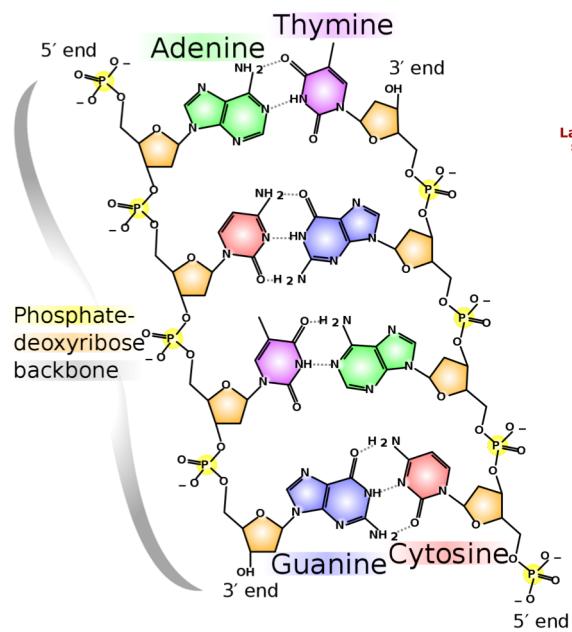
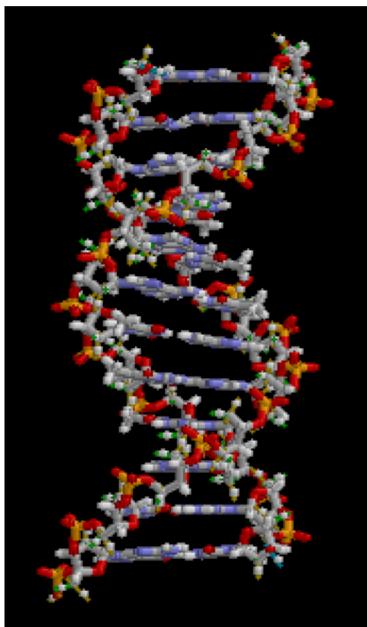
DNA sequencing!

- What organism are you interested in?
- How might you get DNA?



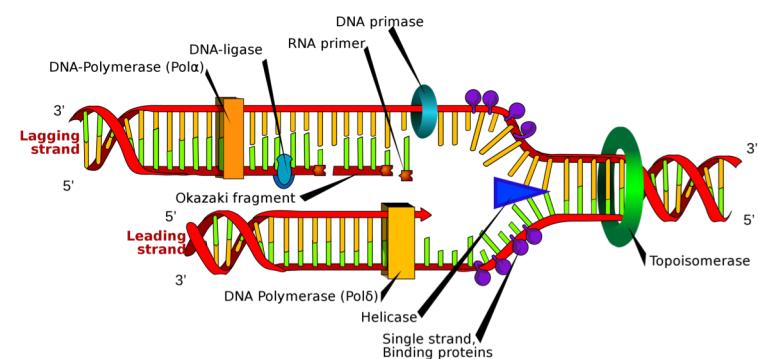
DNA sequencing!

- You have to get that DNA ready for sequencing
- We call this sample prep
- The final processed DNA that is ready for sequencing is called your **library**



By brian0918™ - Own work,
Public Domain,
<https://commons.wikimedia.org/w/index.php?curid=404735>

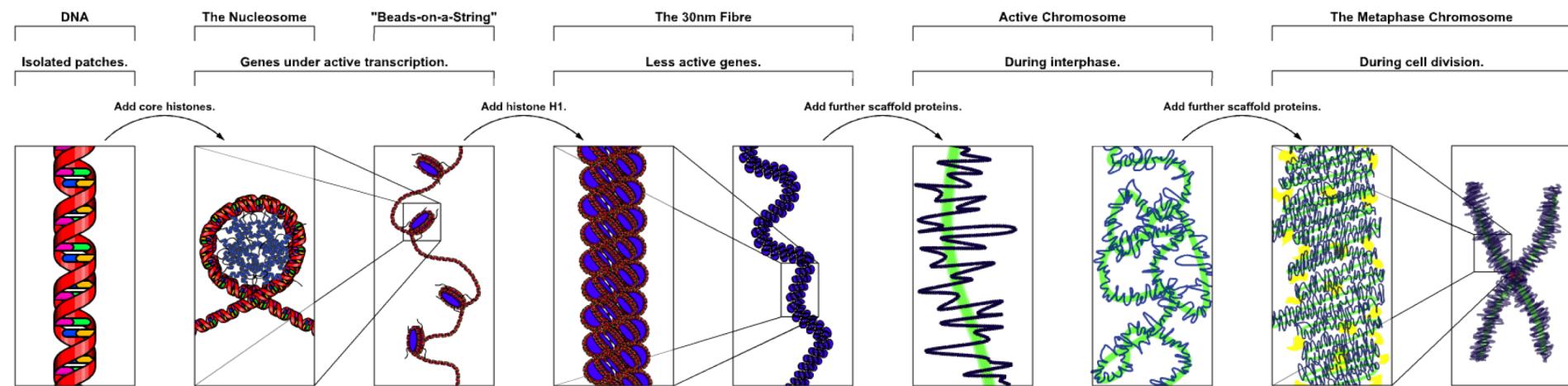
By Madprime (talk · contribs) - Own workThe source code of this SVG is valid.This vector image was created with Inkscape., CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=1848174>



By LadyofHats Mariana Ruiz - Own work
(Original text: Own work. Image renamed from File:DNA replication.svg), Public Domain,
<https://commons.wikimedia.org/w/index.php?curid=6916971>

DNA sequencing!

- You have to get that DNA ready for sequencing
- We call this sample prep
- The final processed DNA that is ready for sequencing is called your **library**

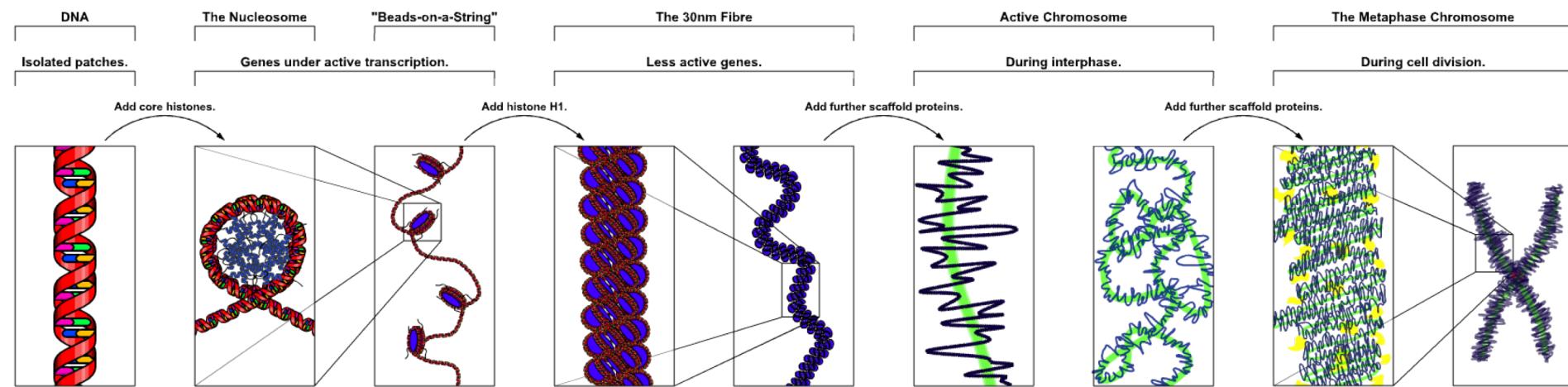


By Original uploader was Richard Wheeler at en.wikipedia - Transferred from en.wikipedia to Commons by sevela.p., CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=4017531>

Illumina is a company that makes the most popular sequencing platform (at this moment in time)

Basic principles underlying Illumina sequencing are no different than other molecular biology techniques you are probably familiar with

- Polymerase Chain Reaction (PCR)



By Original uploader was Richard Wheeler at en.wikipedia - Transferred from en.wikipedia to Commons by sevela.p., CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=4017531>

DNA sequencing!

- How do you sequence
- https://www.youtube.com/watch?v=fCd6B5HRaZ8&feature=emb_logo

The Human Genome is pretty big

- One diploid human genome is 6 billion base pairs in length
- Illumina sequencing can only sequence pieces of DNA that are 100-150 bp in length
- You would need 40 million 150 bp fragments if you want to capture the whole human genome
- Illumina NovaSeq 6000 stats here:
S2 Flow Cell (one of the possible options)
8 human genomes per flow cell (at 30x coverage)
\$16,000 Cost per flow cell
\$2,000 Cost per genome

Chromosome #	size (base pairs)
1	248,956,422
2	242,193,529
3	198,295,559
4	190,214,555
5	181,538,259
6	170,805,979
7	159,345,973
8	145,138,636
9	138,394,717
10	133,797,422
11	135,086,622
12	133,275,309
13	114,364,328
14	107,043,718
15	101,991,189
16	90,338,345
17	83,257,441
18	80,373,285
19	58,617,616
20	64,444,167
21	46,709,983
22	50,818,468
X	156,040,895
Y	57,227,415
Haploid Genome	3,088,269,832
Diploid Genome	6,176,539,664

DNA sequencing!

- What do you do it next?
- Brainstorm questions that you could answer with Whole Genome DNA sequencing data

You can load DNA sequences into R!

- Back to RStudio
- Try typing some of the following into the console of Rstudio
(Use some of your own examples too)

```
dna_1 = "ATGCAT"  
dna_2 = "ATGCCT"  
dna_3 = "ATGCAT"  
print(dna_1)
```

What kind of an object is dna_1?

Character!

Maybe you want to check to see if there is a mutation in your DNA sequence?

- How might you do this in R?
- In RStudio:

```
dna_1 = "ATGCAT"  
dna_2 = "ATGCCT"  
dna_3 = "ATGCAT"
```

```
dna_ref = "ATGCAT"
```

```
dna_1 == dna_ref  
dna_2 == dna_ref  
dna_3 == dna_ref
```

Which object(s) has a mutation?

How do you know?

What type of object is the output?

This is overkill! Why do I need to use R? I could see that dna_2 was different just by looking at it

- In the Google Drive folder there are 4 Beta Hemoglobin coding sequences
- One sequence is labeled HBB_ref.fasta
- This is the reference sequence deposited at NCBI
- (NCBI – National Center for Biotechnology Information)
- NCBI is an invaluable resource where you can find lots of sequences that have been generated by researchers all around the world
- Here: "reference sequence" is what the research community has decided is the standard human sequence of the HBB gene



NCBI Resources How To Sign in to NCBI

Nucleotide Nucleotide Advanced Help

GenBank Send to: Change region shown

Customize view

Analyze this sequence Run BLAST

Pick Primers

Highlight Sequence Features

Find in this Sequence

Show in Genome Data Viewer

Homo sapiens hemoglobin subunit beta (HBB), mRNA

NCBI Reference Sequence: NM_000518.5

FASTA Graphics

Go to:

LOCUS NM_000518 628 bp mRNA linear PRI 04-DEC-2019

DEFINITION Homo sapiens hemoglobin subunit beta (HBB), mRNA.

ACCESSION NM_000518

VERSION NM_000518.5

KEYWORDS RefSeq; MANE Select.

SOURCE Homo sapiens (human)

ORGANISM Homo sapiens

Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominoidea; Homo.

REFERENCE 1 (bases 1 to 628)

AUTHORS Ma Q, An L, Tian H, Liu J, Zhang L, Li X, Wei C, Xie C, Ding H, Qin W and Su Y.

TITLE Interactions between human hemoglobin subunits and peroxiredoxin 2

JOURNAL Environ Mol Cell Biol. 2010 Dec; 29(12):1005-1006. (2010)

Articles about the HBB gene

Characterization of the IVS-II-821 (A>G)C (<i>HBB</i>: c.316-30A [Hemoglobin. 2019])

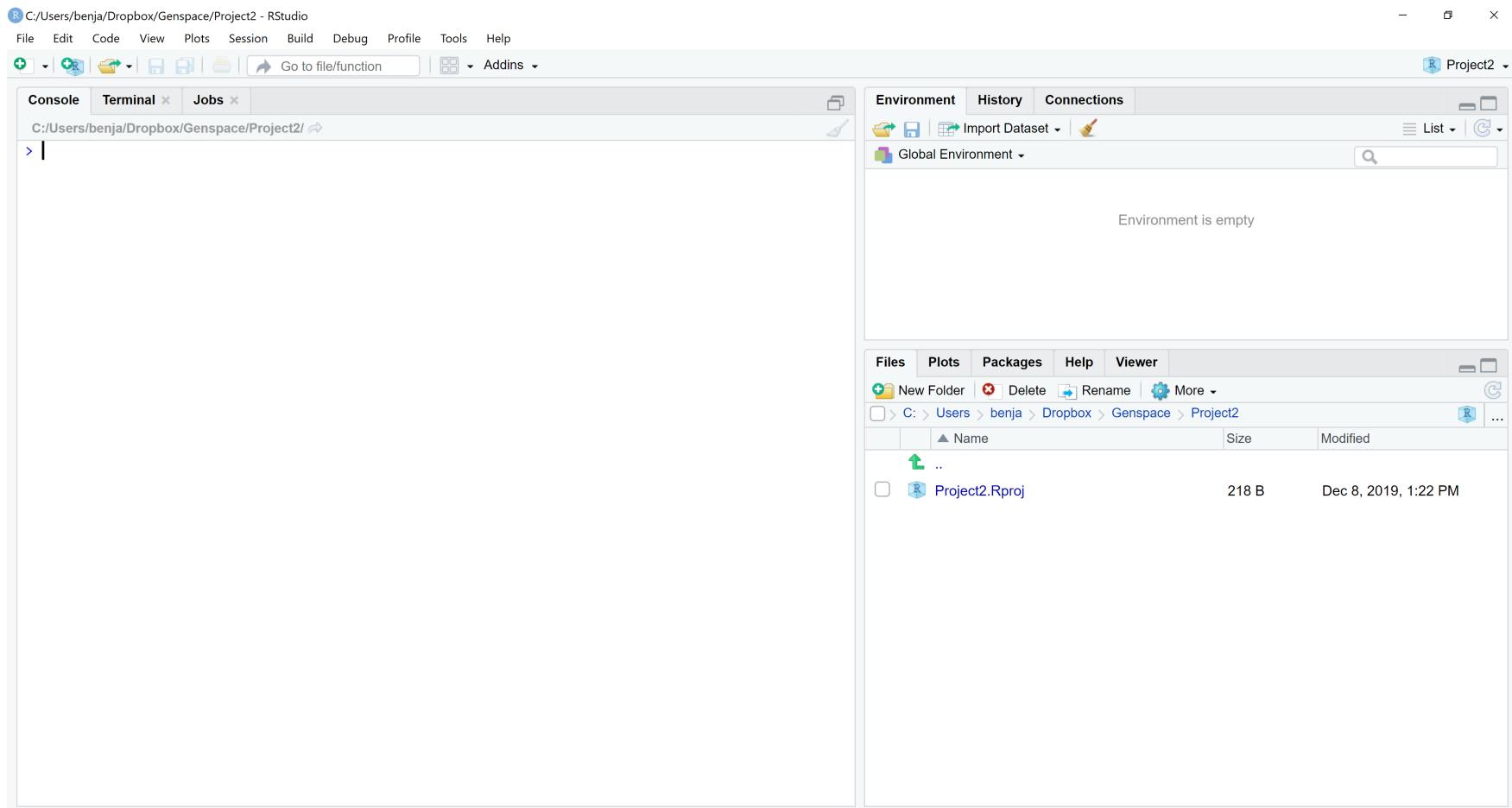
Severe Thalassemia Caused by Hb Zunyi (B147(HC3)Stop) (J Clin Hematol. 2010)

This is overkill! Why do I need to use R?
I could see that dna_2 was different just by looking at it

- In the Google Drive folder there are 4 Beta Hemoglobin coding sequences
- One sequence is labeled HBB_ref.fasta
- There are 3 additional HBB variant files saved in this folder
- Open these in your text editor
- Is there a mutation?
- Let's use R to figure that out

It is kind of tedious to always write one line of code at a time. Right?

- Let's write a script!
- In RStudio, "File" → "New File" → "R Script"



Now you have a new panel in RStudio

The screenshot shows the RStudio interface with several panels:

- Editor:** A large panel on the left containing code. It has a title bar "Untitled1" and a status bar at the bottom. The text inside says: "Editor" and "(You can write your *scripts* here)".
- Global Environment:** A panel on the right showing the global environment. It has tabs for "Environment", "History", and "Connections". Under "Environment", it says "Environment is empty". The text inside says: "Global Environment" and "(As you save objects to memory, they will show up here)".
- Console:** A panel at the bottom left showing the R console. It has tabs for "Console", "Terminal", and "Jobs". The text inside says: "Console" and "(This is the interactive Input/Output interface with R)".
- Files:** A panel at the bottom right showing the file structure. It has tabs for "Files", "Plots", "Packages", "Help", and "Viewer". It shows a single file "R_Course_Working.Rproj" in the "C:\Users\benja\Dropbox\Genspace\R_Course_Working" directory. The text inside says: "There are several interesting things here!" followed by a bulleted list:
 - Files
 - Plots
 - Packages
 - Help(All of these will be very useful)

Write some code in the editor panel in RStudio (Write whatever you want)

The screenshot shows the RStudio interface with the following components:

- Top Bar:** Shows the path "C:/Users/benja/Dropbox/Genspace/Project2 - RStudio".
- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for New Project, Open, Save, Run, Source, and Addins.
- Editor Panel:** An untitled R script named "Untitled1.R" containing the code:

```
1 print("R is actually pretty fun")
2
```
- Environment Panel:** Shows the Global Environment which is currently empty.
- Files Panel:** Shows the project structure under "C:/Users/benja/Dropbox/Genspace/Project2". A file named "Project2.Rproj" is listed with a size of 218 B and a modified date of Dec 8, 2019, 1:22 PM.
- Console Panel:** Shows the current working directory as "C:/Users/benja/Dropbox/Genspace/Project2".

Now to run that code (You have several options)

The screenshot shows the RStudio interface with the following components:

- Code Editor:** An "Untitled1" script window containing the R code:

```
1 print("R is actually pretty fun")
2
```

A red circle highlights the "Run" button in the toolbar above the code editor.
- Environment Pane:** Shows the "Global Environment" tab with the message "Environment is empty".
- File Browser:** A "Files" tab showing the project directory structure:

Name	Size	Modified
Project2.Rproj	218 B	Dec 8, 2019, 1:22 PM
- Console:** Shows the current working directory as "C:/Users/benja/Dropbox/Genspace/Project2/" and a prompt ">".

Listed Steps:

1. Make sure your cursor is on the line of code you want to run
2. Click Run

Now to run that code (You have several options)

OR

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The title bar indicates the current project is "C:/Users/benja/Dropbox/Genspace/Project2 - RStudio". The main window has several panes:

- Code Editor:** An "Untitled1" script contains the following R code:

```
1 print("R is actually pretty fun")
2
```
- Environment Pane:** Shows the Global Environment tab with the message "Environment is empty".
- File Browser:** Shows the project directory structure:

	Name	Size	Modified
..	..		
C:\Users\benja\Dropbox\Genspace\Project2\	Project2.Rproj	218 B	Dec 8, 2019, 1:22 PM
- Console:** The status bar at the bottom left shows "1:1 (Top Level)".
- Terminal:** The status bar at the bottom right shows "R Script".

And the output shows up in the console panel

The screenshot displays the RStudio interface with the following components:

- Top Bar:** Shows the path "C:/Users/benja/Dropbox/Genspace/Project2 - RStudio".
- File Menu:** Includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help.
- Toolbar:** Includes icons for New Project, Open Project, Save, Run, Source, and Addins.
- Code Editor:** An untitled R script with the following code:

```
1 print("R is actually pretty fun")
2
```
- Environment Panel:** Shows the Global Environment tab with the message "Environment is empty".
- Files Panel:** Shows the project directory structure:

```
C: > Users > benja > Dropbox > Genspace > Project2
```

with files "Project2.Rproj" (218 B, Dec 8, 2019, 1:22 PM).
- Console Panel:** Shows the command and its output:

```
> print("R is actually pretty fun")
[1] "R is actually pretty fun"
```

An arrow points to the output line "[1] "R is actually pretty fun"".

Compare the HBB gene sequences again. But this time use editor panel to write a little script (Take a few minutes)

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** C:/Users/benja/Dropbox/Genspace/Project2 - RStudio
- Editor Panel:** Untitled1* (R Script)
1 print("R is actually pretty fun")
2
- Environment Panel:** Environment tab selected. Global Environment: Environment is empty.
- Files Panel:** Files tab selected. Project2.Rproj (218 B, Dec 8, 2019, 1:22 PM)
- Console Panel:** Console tab selected. Output:
> print("R is actually pretty fun")
[1] "R is actually pretty fun" ←

Let's make things more complicated!

- So far the objects we have created contain a single thing
- For example,

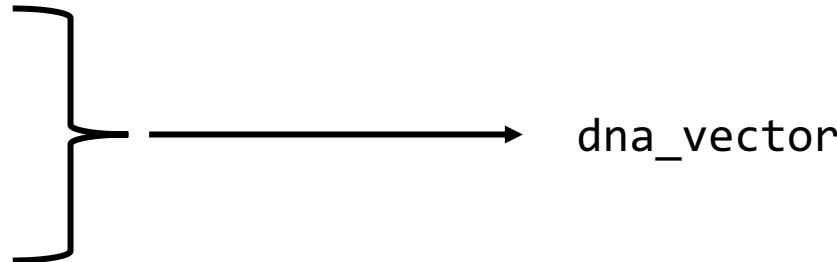
```
dna_ref = "ATGCAT"  
dna_1 = "ATGCAT"  
dna_2 = "ATGCCT"  
dna_3 = "ATGCAT"
```

- But sometimes/often, data will be complex
- If you had the sequence of the hemoglobin beta gene from 20,000 people, would you want to create 20,000 variables to hold each sequence on its own?
- Probably not:
 - it would take a long time
 - it could be confusing
 - it would be easy to make a mistake

You can save multiple items within a single **Vector**

- So you can take these four DNA sequences,

```
dna_ref = "ATGCAT"  
dna_1 = "ATGCAT"  
dna_2 = "ATGCCT"  
dna_3 = "ATGCAT"
```



- And put them all into a single object
- And this new object will look like this:

```
[1] "ATGCAT" "ATGCAT" "ATGCCT" "ATGCAT"
```

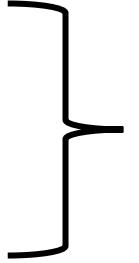
- How do you think you might do that?
- (Hint, in R, what do you call an object that performs a task for you?)
- A **function!**

The art of Googling for answers to your R problems

- As you work more in R, you will get better and better
(practice is helpful)
- But no one ever remembers everything
- there is no reason you should have to
- There are a lot of online resources to help you effectively use R
- There is a VERY large community of R users in the world
- I guarantee, when you experience a problem with R:
 - someone else has had the exact same problem,
 - they have posted that question to an internet forum (often on stackoverflow.com)
 - And many people have posted their helpful responses to help solve that problem
(Sometimes their answers are hard to follow. This will get easier over time)
- **So back to our problem:**

How do you do this?

```
 dna_ref = "ATGCAT"  
 dna_1 = "ATGCAT"  
 dna_2 = "ATGCCT"  
 dna_3 = "ATGCAT"
```

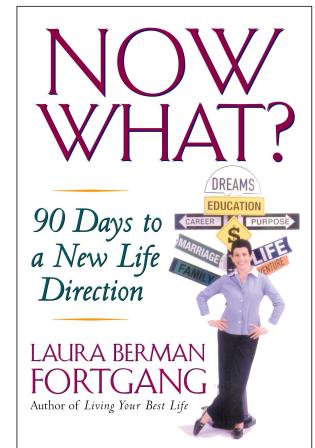


The diagram shows four lines of code: dna_ref, dna_1, dna_2, and dna_3. A curly brace groups dna_ref, dna_1, and dna_2 together, pointing to the variable dna_vector on the right.

- How would you phrase a Google search to give you your answer?
 - (Take a minute, talk with your partner, Google it)
-
- What is the function? **c()**
 - What were some of your Google search phrases?
 - What were the important words to include?
 - Anything weird or unexpected? Any trouble finding the specific info you needed?

So you have found the right function to use, Now What?

- You need to know how the function works!
- This info is called **documentation**
- All functions should have documentation
- You can find this info in a couple of different places:
 - Probably in one of the Google results you just opened
 - Help within RStudio



RStudio's Help Panel

The screenshot shows the RStudio interface with several panels:

- Top Bar:** Shows the path "C:/Users/benja/Dropbox/Genspace/Project2 - RStudio".
- File Menu:** Includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help.
- Toolbar:** Includes icons for New, Open, Save, Run, Source, and Addins.
- Code Editor:** An untitled R script with the following code:

```
1 print("R is actually pretty fun")
2
```
- Console:** Displays the output of the printed statement: "R is actually pretty fun".
- Environment:** Shows the Global Environment, which is currently empty.
- Bottom Right Panel:** A file browser with tabs for Files, Plots, Packages, Help, and Viewer. The Help tab is circled in red. The viewer panel shows a single file: "Project2.Rproj" located at "C:/Users/benja/Dropbox/Genspace/Project2".
- Text Overlay:** The text "Click the 'Help' tab in the bottom right panel" is overlaid on the image, pointing to the circled tab.

RStudio's Help Panel

C:/Users/benja/Dropbox/Genspace/Project2 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1* Untitled2*

Source on Save | Import Dataset | Global Environment

Run | Source | Addins

1 print("R is actually pretty fun")
2

Environment History Connections

Environment is empty

Type the name of the function you want to learn about here

2:1 (Top Level) R Script

Console Terminal Jobs

C:/Users/benja/Dropbox/Genspace/Project2/

> print("R is actually pretty fun")
[1] "R is actually pretty fun"
>

Files Plots Packages Help Viewer

Home Find in Topic

R Resources

- Learning R Online
- CRAN Task Views
- R on StackOverflow
- Getting Help with R

RStudio

- RStudio IDE Support
- RStudio Community Forum
- RStudio Cheat Sheets
- RStudio Tip of the Day
- RStudio Packages
- RStudio Products

Manuals

RStudio's Help Panel

The screenshot shows the RStudio interface with the following components:

- Top Bar:** C:/Users/benja/Dropbox/Genspace/Project2 - RStudio. Includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help menus.
- Toolbar:** Includes icons for New, Open, Save, Run, Source, and Addins.
- Code Editor:** Untitled1* and Untitled2*. Untitled1* contains the code:

```
1 print("R is actually pretty fun")
```
- Environment Tab:** Shows the Global Environment which is currently empty.
- Console Tab:** (Top Level) shows the output of the command:

```
> print("R is actually pretty fun")  
[1] "R is actually pretty fun"  
>
```
- Help Panel:** The documentation for the `c()` function is displayed. A red oval highlights the panel area.

The documentation content includes:

- Title:** `c {base}`
- Description:** This is a generic function which combines its arguments.
- Usage:** The default method combines its arguments to form a vector. All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.

Alternative way to get help with a function:

- In the console, type **?{function_name}**

The screenshot shows the RStudio interface with the following components:

- Console:** Displays the command `> ?c` and its output [1] "c". An arrow points from the text "For example, ?c" to the question mark command in the console.
- Environment:** Shows the message "Environment is empty".
- File Browser:** Shows a directory structure under "C:/Users/benja/Dropbox/Genspace/Project2".

Name	Size	Modified
..		
.Rhistory	203 B	Dec 9, 2019, 1:05 PM
Project2.Rproj	218 B	Dec 9, 2019, 1:07 PM

Alternative way to get help with a function:

- In the console, type **?{function_name}**

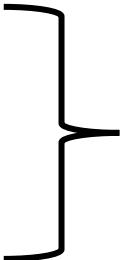
The screenshot shows the RStudio interface with the following details:

- Console:** Displays the command `> ?c` and its output, which is empty.
- Help Viewer:** Shows the documentation for the `c` function. The title is "Combine Values into a Vector or List".
 - Description:** States it's a generic function combining arguments into a vector.
 - Usage:** Describes the default method of combining arguments into a vector.
- Environment:** Shows that the environment is empty.

A large red oval highlights the Help Viewer area, specifically the documentation for the `c` function.

Do it (take a few minutes)

```
dna_ref = "ATGCAT"  
dna_1 = "ATGCAT"  
dna_2 = "ATGCCT"  
dna_3 = "ATGCAT"
```



The diagram shows four DNA sequences: dna_ref, dna_1, dna_2, and dna_3. A brace groups dna_ref, dna_1, and dna_2, which then points to a variable named dna_vector.

- Read the documentation for the c() function
- (There is always more info than you need, look for what you need, ignore the rest)
- Make a vector combining these 4 DNA sequences and set them equal to a new variable you create called **dna_vector**

Hint 1:

- A **function** always has parentheses
- The stuff inside its parentheses are called **arguments**
- A **function** is a little machine. It grabs the **arguments** that you give it, it does something behind the scenes, and gives you (**returns**) something else

Hint 2:

- So far, we have used the = sign to set a value to a variable
- There are actually two ways to do this.
- The two ways are (more or less) identical
- The other way is to use <- (the less than sign followed by a dash). Think of it like a little arrow funneling the thing on the right into the variable on the left

So these two expressions are identical:

```
dna_ref = "ATGCAT"  
dna_ref <- "ATGCAT"
```

Did you have something like this?

```
dna_ref = "ATGCAT"
```

```
dna_1 = "ATGCAT"
```

```
dna_2 = "ATGCCT"
```

```
dna_3 = "ATGCAT"
```

```
dna_vector = c(dna_ref,dna_1,dna_2,dna_3)
```

Any other solutions?

Other things to know about vectors in R

- Every item stored in a vector is called an ***element***
- The position of an ***element*** within a vector is called its ***index***
- In RStudio make this new vector

```
animal_vector = c("Human", "Mouse", "Fly", "Worm")
```

- In the new object you just made, **animal_vector**, what is the ***index*** of "Human"?
- 1
- It seems obvious, but in many other programming languages, the index of the first item in a vector (or vector like object) is arbitrarily called 0
- You can add elements to an existing vector
- Some functions work on vectors

Activity (Break out groups):

Figure out how to solve the following problems using Google: (search "working with vectors in r" the first hit is very helpful)

- Write a line of R code that will return the 4th element in animal_vector
- Write a line of R code that will return the first 3 elements of animal_vector
- Write a line of R codes that will return elements 2 and 4 of animal_vector

- Uh oh, You realized you forgot that "Panda", and "Maple" should be in your vector animal_vector
- Add them
- Calculate (using a function) how many species are in your vector
- Uh oh, you realize that the last element is a plant. Please remove it
- How many elements are in your vector now?
- Please make two new vectors one called vertebrate_vector containing the vertebrate species and one called invertebrate_vector containing the invertebrate species (** Accomplish this using what you have learned about indices)