In_course_assignment.

Vatsala Tewari

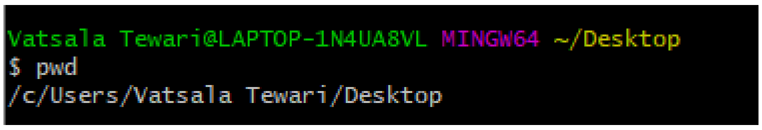
Unix

Exercise U.1

Write the path of your current working directory? What command do you use?

Answer U.1

The path to my current working directors is `/c/Users/Vatsala Tewari/Desktop`
I used the **pwd** command to show this path.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ pwd
/c/Users/Vatsala Tewari/Desktop
```

Exercise U.2

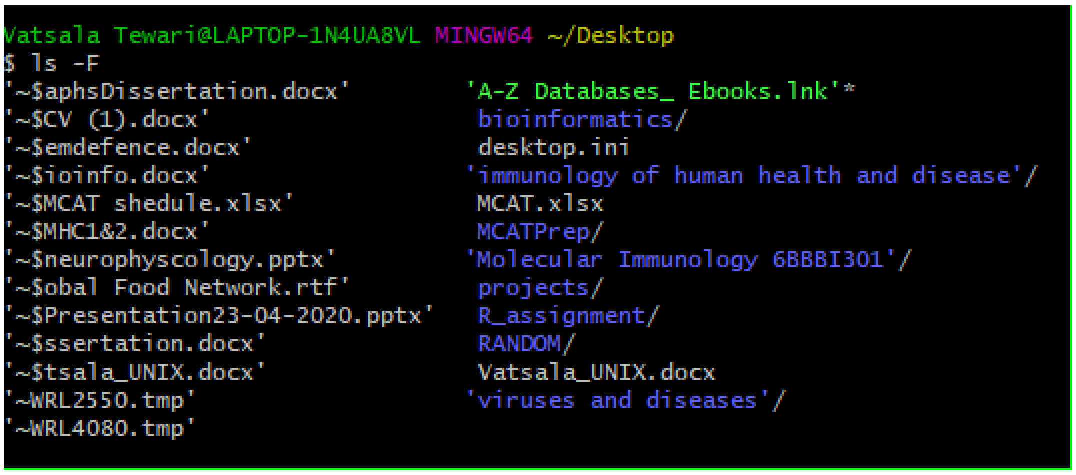
How many files and folder are present in your working directory? What's the flag that helps distinguish between files and directories?

Answer U.2

The function here to use is ls-F, this will list all the files and give a summary of type of data stored , / for folders , * for executables, @ for symbolic link etc.

The functions distinguish these with characters . Files are distinguished by no character, and directories with a /

Here I have used ls -F,

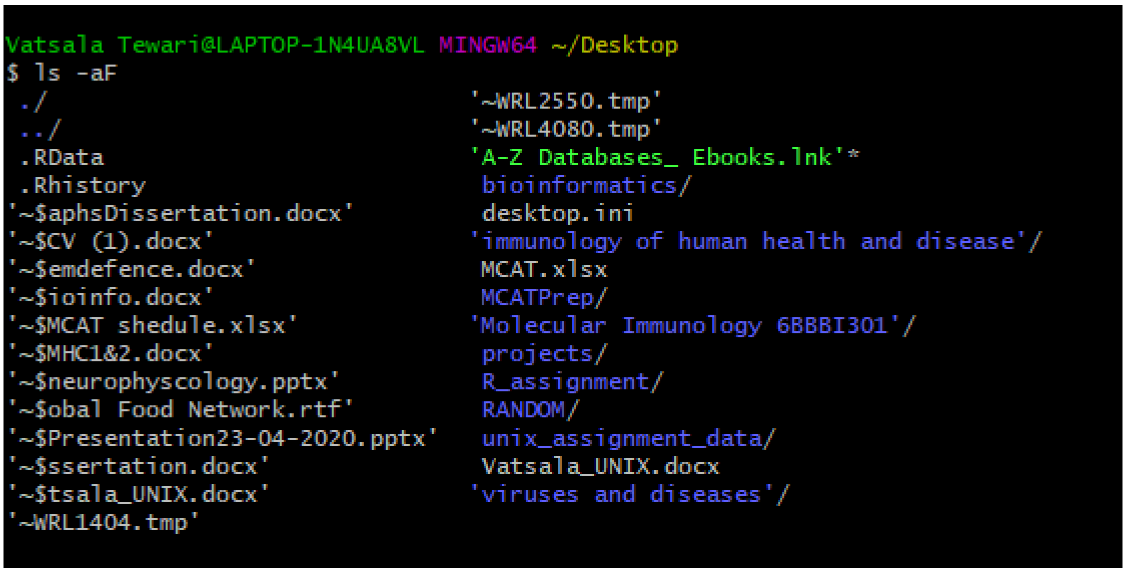


```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ ls -F
'~$aphsDissertation.docx'         'A-Z Databases_ Ebooks.lnk'*
'~$CV (1).docx'                    bioinformatics/
'~$emdefence.docx'                 desktop.ini
'~$ioinfo.docx'                   'immunology of human health and disease'/
'~$MCAT shedule.xlsx'              MCAT.xlsx
'~$MHC1&2.docx'                    MCATPrep/
'~$neurophyscology.pptx'          'Molecular Immunology 6BBBI301'/
'~$obal Food Network.rtf'          projects/
'~$Presentation23-04-2020.pptx'    R_assignment/
'~$ssertation.docx'                RANDOM/
'~$tsala_UNIX.docx'                Vatsala_UNIX.docx
'~WRL2550.tmp'                    'viruses and diseases'/
'~WRL4080.tmp'
```

But these are not all the files. Yes! , we all have hidden or ghost files on our computers that store important information, for e.g. .profile is a hidden file. They don't store confidential information, they are just out of the view in the normal user interface as they store very important data , that doesn't need to be edited by the user on the application and has information that is prevented to be accidentally deleted.

In bash ls-aF

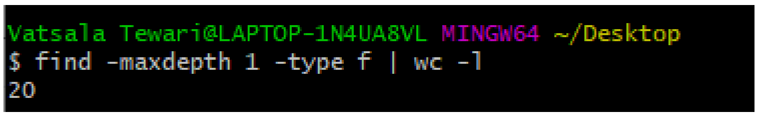
Will list ALL FILES and also give us the information about the type.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ ls -aF
 ./                                 '~WRL2550.tmp'
 ../                                '~WRL4080.tmp'
 .RData                             'A-Z Databases_ Ebooks.lnk'*
 .Rhistory                           bioinformatics/
'~$aphsDissertation.docx'            desktop.ini
'~$CV (1).docx'                     'immunology of human health and disease'/
'~$emdefence.docx'                   MCAT.xlsx
'~$ioinfo.docx'                      MCATPrep/
'~$MCAT shedule.xlsx'               'Molecular Immunology 6BBBI301'/
'~$MHC1&2.docx'                      projects/
'~$neurophyscology.pptx'             R_assignment/
'~$obal Food Network.rtf'            RANDOM/
'~$Presentation23-04-2020.pptx'      unix_assignment_data/
'~$ssertation.docx'                  Vatsala_UNIX.docx
'~$tsala_UNIX.docx'                 'viruses and diseases'/
'~WRL1404.tmp'
```

Although counting here is easy, sometimes there are large no. of files and directories hence we can use functions present in bash.

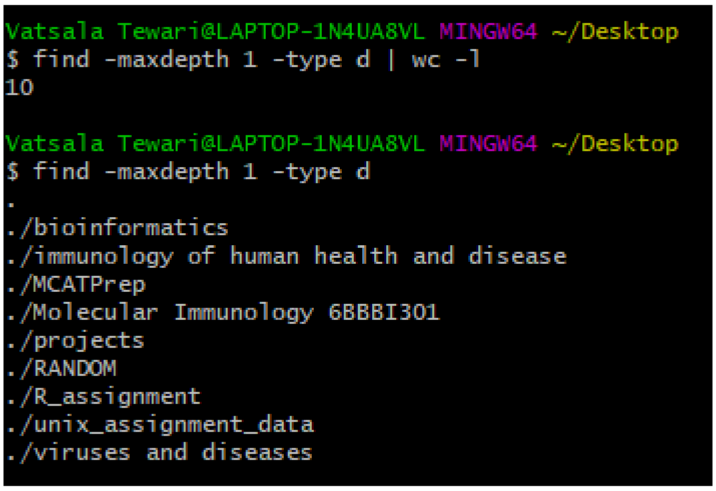
$ find function will find -type f (type of file) in desktop as defined by maxdepth 1 function. Piping this further to a wc -l function to count number of lines in the list, will help us understand without counting how many files are present.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ find -maxdepth 1 -type f | wc -l
20
```

Hence, there are 20 such files in my desktop, counting the hidden files.

10 folders or directories are there in my desktop counting the hidden directory (.)



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ find -maxdepth 1 -type d | wc -l
10

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ find -maxdepth 1 -type d
.
./bioinformatics
./immunology of human health and disease
./MCATPrep
./Molecular Immunology 6BBBI301
./projects
./RANDOM
./R_assignment
./unix_assignment_data
./viruses and diseases
```

You can subtract number of hidden files from the number we found above. Hidden files can be found by listing all files and counting those which start with (.)

Exercise U.3

Sort the files by size and select the biggest one

Answer U.3

Ls-a list all the files,

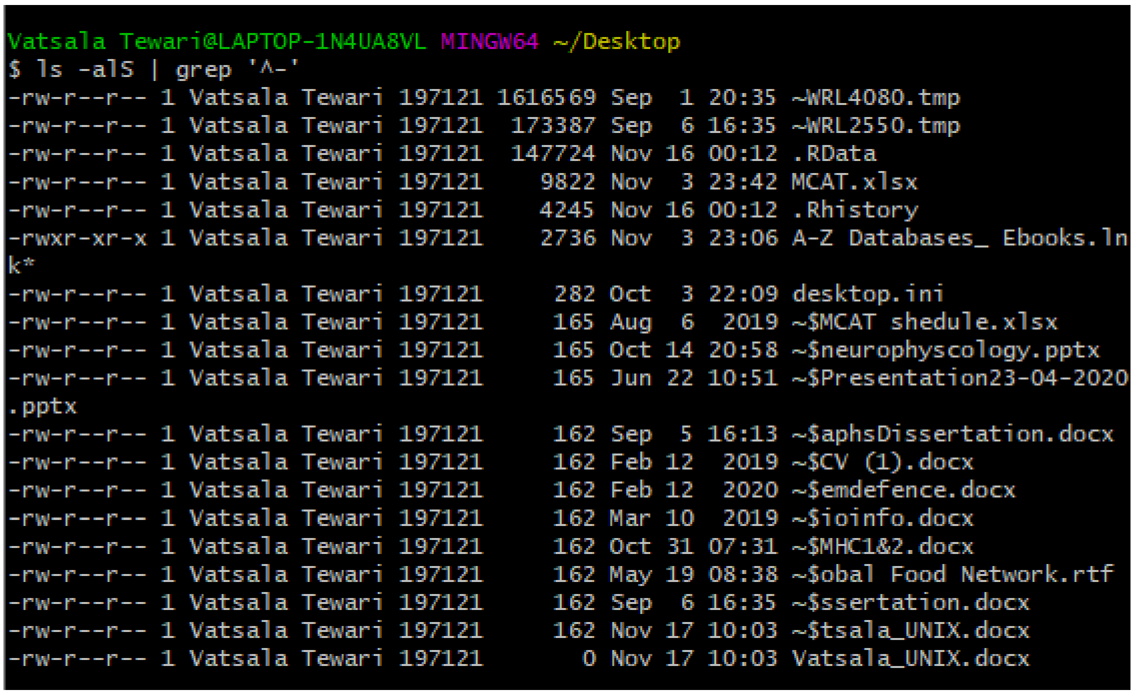
Ls-l lists longer listing of files

ls-S lists size and sorts them by size.

The | pipes this together to the next function

Grep searches if a single – is given for files.

Hence,The biggest file is WRL4080.tmo



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ ls -alS | grep '^-'
-rw-r--r-- 1 Vatsala Tewari 197121 1616569 Sep  1 20:35 ~WRL4080.tmp
-rw-r--r-- 1 Vatsala Tewari 197121  173387 Sep  6 16:35 ~WRL2550.tmp
-rw-r--r-- 1 Vatsala Tewari 197121  147724 Nov 16 00:12 .RData
-rw-r--r-- 1 Vatsala Tewari 197121    9822 Nov  3 23:42 MCAT.xlsx
-rw-r--r-- 1 Vatsala Tewari 197121    4245 Nov 16 00:12 .Rhistory
-rwxr-xr-x 1 Vatsala Tewari 197121    2736 Nov  3 23:06 A-Z Databases_ Ebooks.ln
k*
-rw-r--r-- 1 Vatsala Tewari 197121     282 Oct  3 22:09 desktop.ini
-rw-r--r-- 1 Vatsala Tewari 197121     165 Aug  6  2019 ~$MCAT shedule.xlsx
-rw-r--r-- 1 Vatsala Tewari 197121     165 Oct 14 20:58 ~$neurophyscology.pptx
-rw-r--r-- 1 Vatsala Tewari 197121     165 Jun 22 10:51 ~$Presentation23-04-2020
.pptx
-rw-r--r-- 1 Vatsala Tewari 197121     162 Sep  5 16:13 ~$aphsDissertation.docx
-rw-r--r-- 1 Vatsala Tewari 197121     162 Feb 12  2019 ~$CV (1).docx
-rw-r--r-- 1 Vatsala Tewari 197121     162 Feb 12  2020 ~$emdefence.docx
-rw-r--r-- 1 Vatsala Tewari 197121     162 Mar 10  2019 ~$ioinfo.docx
-rw-r--r-- 1 Vatsala Tewari 197121     162 Oct 31 07:31 ~$MHC1&2.docx
-rw-r--r-- 1 Vatsala Tewari 197121     162 May 19 08:38 ~$obal Food Network.rtf
-rw-r--r-- 1 Vatsala Tewari 197121     162 Sep  6 16:35 ~$ssertation.docx
-rw-r--r-- 1 Vatsala Tewari 197121     162 Nov 17 10:03 ~$tsala_UNIX.docx
-rw-r--r-- 1 Vatsala Tewari 197121       0 Nov 17 10:03 Vatsala_UNIX.docx
```

Exercise U.4

Download a folder at the following link: https://www.dropbox.com/sh/mozhvvhku3dg7m4/AAC15bDj1jf6qFkzm618k6bfa?

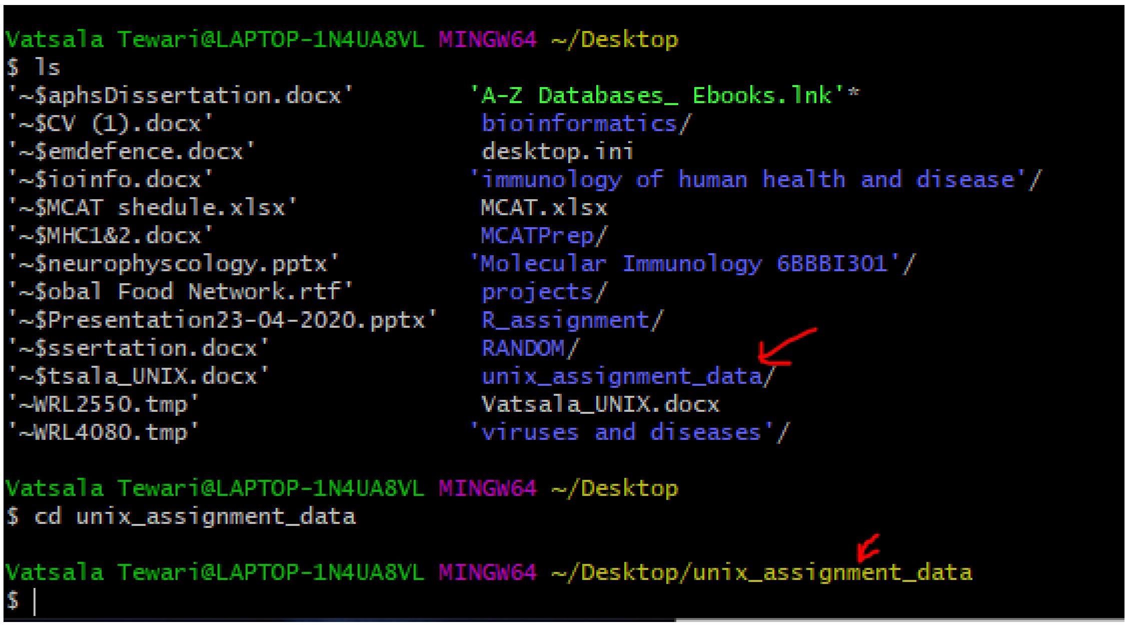
dl=0>

Move from your current working directory into the ‘unix_assignment_data’ folder just downloaded. Make

sure you are in the correct directory and list the files in the folder.

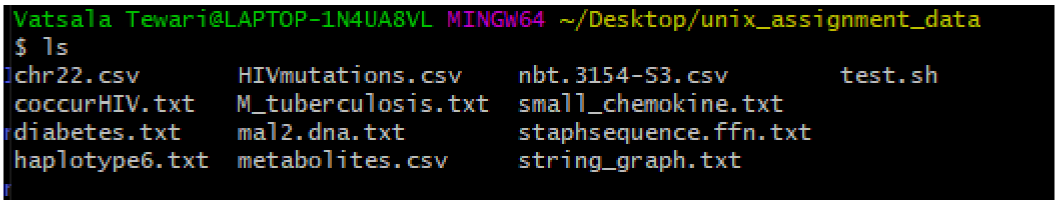
Answer U.4

Using the cd function we can jump to a directory.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ ls
'~$aphsDissertation.docx'         'A-Z Databases_ Ebooks.lnk'*
'~$CV (1).docx'                    bioinformatics/
'~$emdefence.docx'                 desktop.ini
'~$ioinfo.docx'                   'immunology of human health and disease'/
'~$MCAT shedule.xlsx'              MCAT.xlsx
'~$MHC1&2.docx'                    MCATPrep/
'~$neurophyscology.pptx'          'Molecular Immunology 6BBBI301'/
'~$obal Food Network.rtf'          projects/
'~$Presentation23-04-2020.pptx'    R_assignment/
'~$ssertation.docx'                RANDOM/
'~$tsala_UNIX.docx'                unix_assignment_data/
'~WRL2550.tmp'                     Vatsala_UNIX.docx
'~WRL4080.tmp'                    'viruses and diseases'/

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop
$ cd unix_assignment_data

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ |
```

To list files we use the ls function



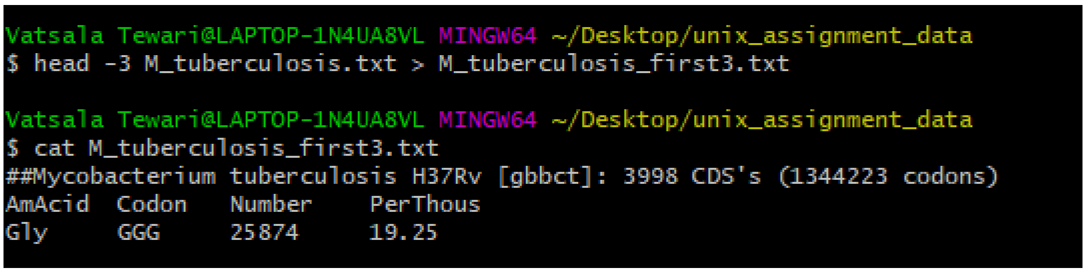
```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ ls
chr22.csv       HIVmutations.csv    nbt.3154-S3.csv        test.sh
coccurHIV.txt   M_tuberculosis.txt  small_chemokine.txt
diabetes.txt    mal2.dna.txt        staphsequence.ffn.txt
haplotype6.txt  metabolites.csv     string_graph.txt
```

Exercise U.5

1. Create a new file called 'M_tuberculosis_first3.txt' containing the first 3 rows of the 'M_tuberculosis.txt'file.

Answer U.5 : Head -3 lists the first three lines of the M_tuberculosis.txt function to the new file using the > function.

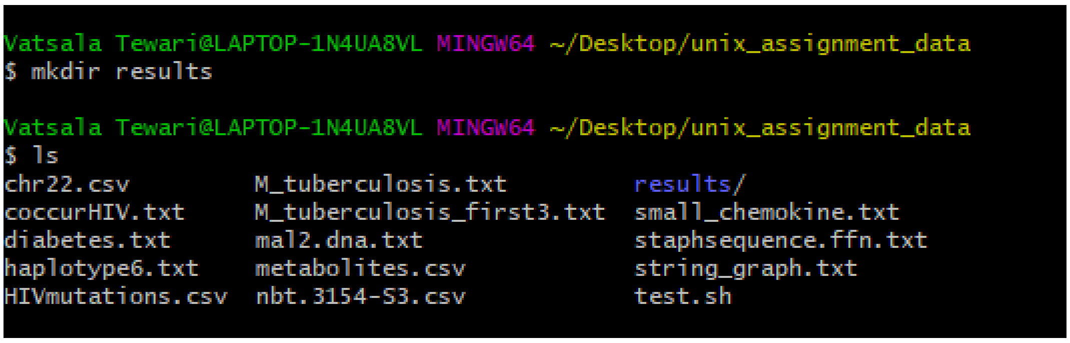
We can see this using the cat function, to verify the result of our code.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ head -3 M_tuberculosis.txt > M_tuberculosis_first3.txt

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ cat M_tuberculosis_first3.txt
##Mycobacterium tuberculosis H37Rv [gbbct]: 3998 CDS's (1344223 codons)
AmAcid  Codon   Number   PerThous
Gly     GGG     25874    19.25
```

2. Make a new folder called 'results'

mkdir creates a new directory which is highlighted like this **results/**

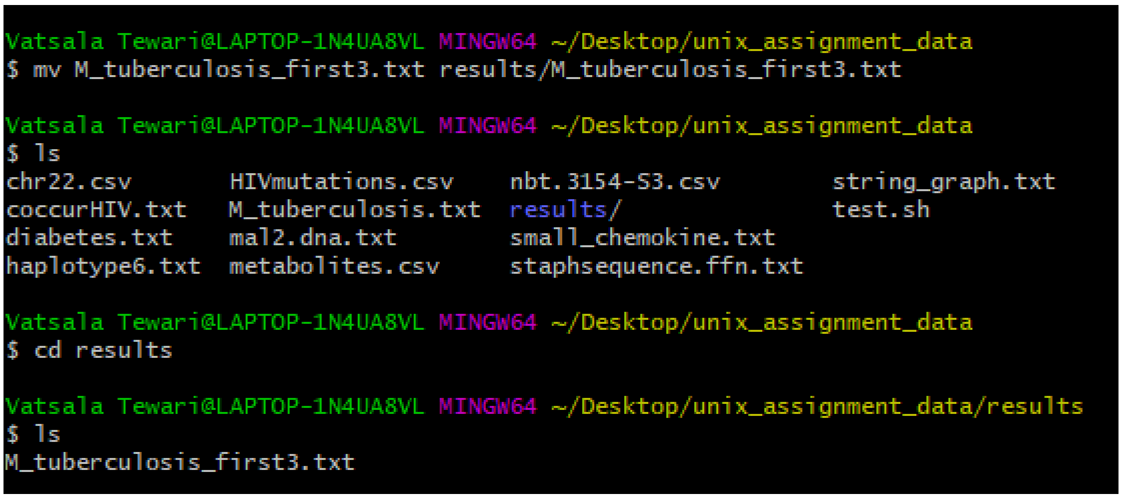


```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ mkdir results

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ ls
chr22.csv         M_tuberculosis.txt         results/
coccurHIV.txt     M_tuberculosis_first3.txt  small_chemokine.txt
diabetes.txt      mal2.dna.txt               staphsequence.ffn.txt
haplotype6.txt    metabolites.csv            string_graph.txt
HIVmutations.csv  nbt.3154-S3.csv            test.sh
```

3. Move the 'M_tuberculosis_first3.txt' file into the 'results' folder

To move the file to the folder we can use the rename function by the code,below.

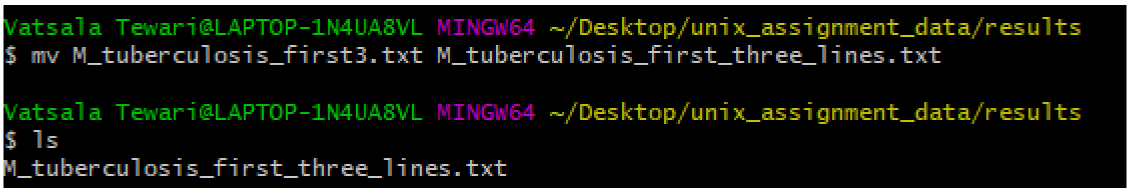
Where mv is for move, then we list the original file then we add results/ before the name of the file to change its path to the results directory.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ mv M_tuberculosis_first3.txt results/M_tuberculosis_first3.txt

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ ls
chr22.csv       HIVmutations.csv    nbt.3154-S3.csv        string_graph.txt
coccurHIV.txt   M_tuberculosis.txt  results/               test.sh
diabetes.txt    mal2.dna.txt        small_chemokine.txt
haplotype6.txt  metabolites.csv     staphsequence.ffn.txt

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ cd results

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data/results
$ ls
M_tuberculosis_first3.txt
```

4. Rename the file to 'M_tuberculosis_first_three_lines.txt'

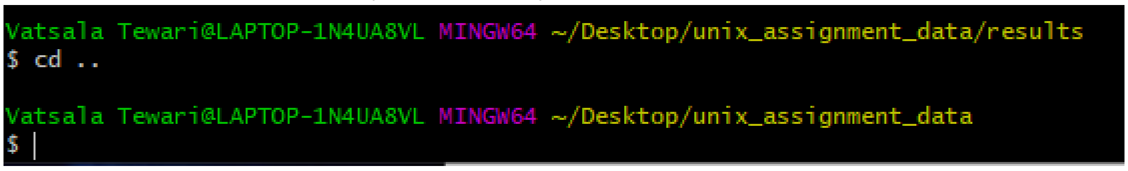
Using the similar logic above rather than changing the path we can just change the name of the file, like done below.



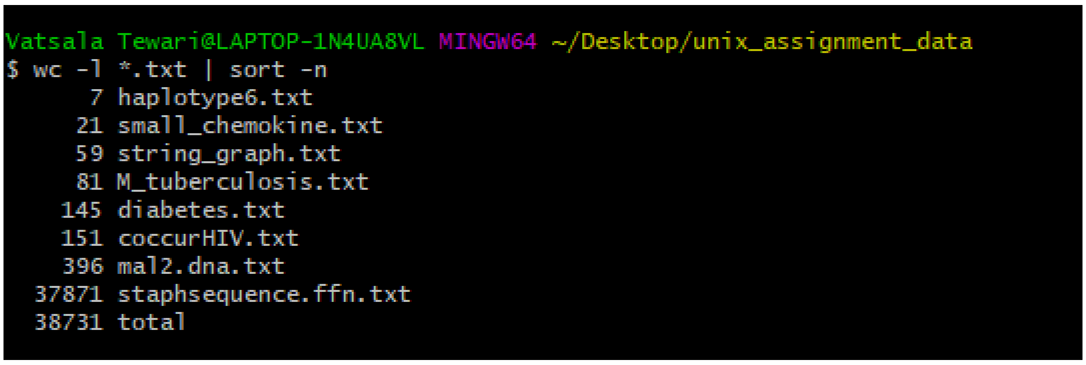
```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data/results
$ mv M_tuberculosis_first3.txt M_tuberculosis_first_three_lines.txt

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data/results
$ ls
M_tuberculosis_first_three_lines.txt
```

Exercise U.6

1. Go back to the 'unix_assignment_data' folder and count the number of lines, words, and characters in the txt files.
   Answer. Wc command will help us to identify the lines, words and characters.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data/results
$ cd ..

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ |
```



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ wc -l *.txt | sort -n
      7 haplotype6.txt
     21 small_chemokine.txt
     59 string_graph.txt
     81 M_tuberculosis.txt
    145 diabetes.txt
    151 coccurHIV.txt
    396 mal2.dna.txt
  37871 staphsequence.ffn.txt
  38731 total
```

You can understand the code this way

wc – l is to count the number of lines for every file ending with a txt and then pipe them to sort them by number

2. Which of these files contains the fewest lines?

Haplotype6.txt contains the fewest lines

Exercise U.7

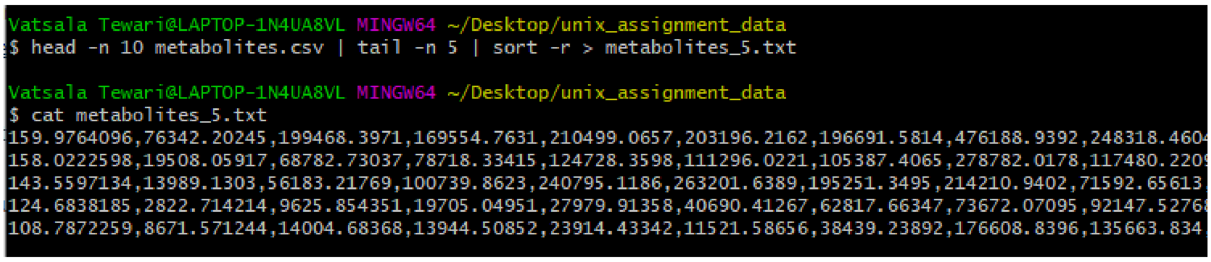
Pipe these commands together:

1.Extract the first 10 lines from 'metabolites.txt' 2.Extract the last 5 lines from the previous 10 3.Sort these

5 lines in reverse order 4.Redirect the final file to an output called 'metabolites_5.txt'

Answer

Head -n 10 lists the first 10 files, in metabolite.csv then pipes it to the next command which then lists the last five lines further piping it to the next command that sorts the results in the reverse order and then saves the input in a file called metabolite.txt



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ head -n 10 metabolites.csv | tail -n 5 | sort -r > metabolites_5.txt

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ cat metabolites_5.txt
159.9764096,76342.20245,199468.3971,169554.7631,210499.0657,203196.2162,196691.5814,476188.9392,248318.460
158.0222598,19508.05917,68782.73037,78718.33415,124728.3598,111296.0221,105387.4065,278782.0178,117480.220
143.5597134,13989.1303,56183.21769,100739.8623,240795.1186,263201.6389,195251.3495,214210.9402,71592.65613
124.6838185,2822.714214,9625.854351,19705.04951,27979.91358,40690.41267,62817.66347,73672.07095,92147.5276
108.7872259,8671.571244,14004.68368,13944.50852,23914.43342,11521.58656,38439.23892,176608.8396,135663.834
```

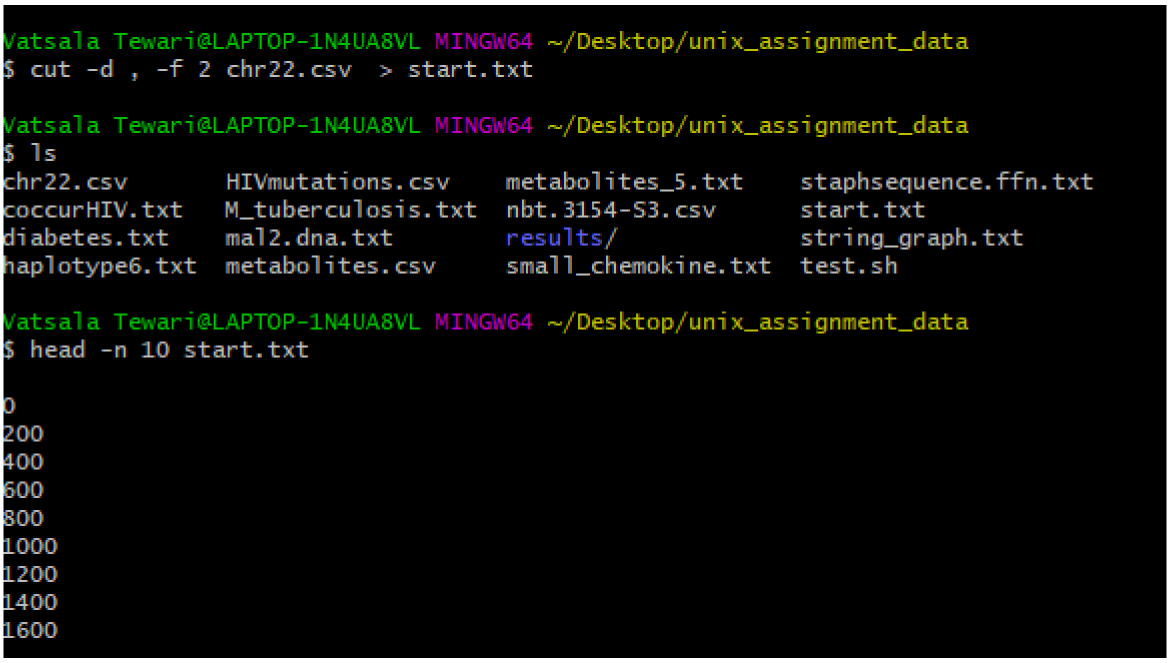
Exercise U.8

Extract the second column of the ‘chr22.csv’ file and redirect the output into a new file called ‘start.txt’

Answer U.8

The cut command helps to display select columns, but first we need to decide the delimiter, then -f 2 defines that we need to show only the second column from the chr22.csv and save it to using > , a file name start.txt

The further commands were to check.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ cut -d , -f 2 chr22.csv  > start.txt

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ ls
chr22.csv       HIVmutations.csv    metabolites_5.txt    staphsequence.ffn.txt
coccurHIV.txt   M_tuberculosis.txt  nbt.3154-S3.csv      start.txt
diabetes.txt    mal2.dna.txt        results/             string_graph.txt
haplotype6.txt  metabolites.csv     small_chemokine.txt  test.sh

Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ head -n 10 start.txt

0
200
400
600
800
1000
1200
1400
1600
```

Exercise U.9

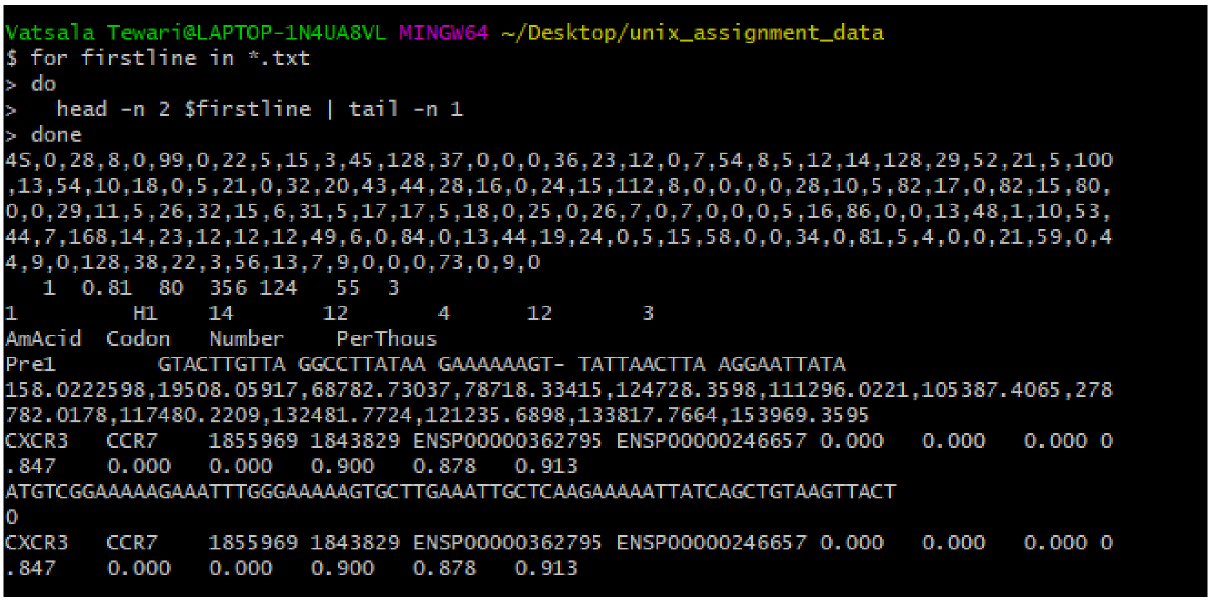
Make a loop that takes each text file and first extracts the first two lines and pipe the result to tail to extract the first of the two lines

Answer :

for firstline in^.txt code establishes that bash need to repeat the variable firstline for every txt file in the directory unix_assignment_data.

Then the file will run the head command taking the first two lines and piping it to the next function that takes the second last line.

This way the first line of every group can be seen below.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ for firstline in *.txt
> do
>   head -n 2 $firstline | tail -n 1
> done
45,0,28,8,0,99,0,22,5,15,3,45,128,37,0,0,0,36,23,12,0,7,54,8,5,12,14,128,29,52,21,5,100
,13,54,10,18,0,5,21,0,32,20,43,44,28,16,0,24,15,112,8,0,0,0,0,28,10,5,82,17,0,82,15,80,
0,0,29,11,5,26,32,15,6,31,5,17,17,5,18,0,25,0,26,7,0,7,0,0,0,5,16,86,0,0,13,48,1,10,53,
44,7,168,14,23,12,12,12,49,6,0,84,0,13,44,19,24,0,5,15,58,0,0,34,0,81,5,4,0,0,21,59,0,4
4,9,0,128,38,22,3,56,13,7,9,0,0,0,73,0,9,0
   1  0.81   80  356 124   55   3
1          H1    14        12        4       12        3
AmAcid  Codon   Number    PerThous
Pre1        GTACTTGTTA GGCCTTATAA GAAAAAAGT- TATTAACTTA AGGAATTATA
158.0222598,19508.05917,68782.73037,78718.33415,124728.3598,111296.0221,105387.4065,278
782.0178,117480.2209,132481.7724,121235.6898,133817.7664,153969.3595
CXCR3   CCR7    1855969 1843829 ENSP00000362795 ENSP00000246657 0.000   0.000   0.000 0
.847    0.000   0.000   0.900   0.878   0.913
ATGTCGGAAAAAGAAATTTGGGAAAAAGTGCTTGAAATTGCTCAAGAAAAATTATCAGCTGTAAGTTACT
0
CXCR3   CCR7    1855969 1843829 ENSP00000362795 ENSP00000246657 0.000   0.000   0.000 0
.847    0.000   0.000   0.900   0.878   0.913
```

Exercise U.10

Create a shell script with the same command of the previous exercise and call it 'extraction.sh' (try to make it efficient and use the special variables "$1" etc. . . for the number of lines to be passed and for the filename)

Answer U.10

Sometimes we want to get a different output by changing variables via similar process, for example what if we wanted the 6-18 rows from the diabetes.txt we would have to make a new loop with a new variable name.

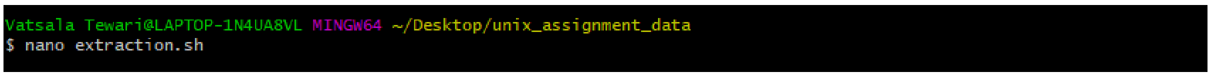
Not really, we can create a shell script.

It's like a text editor but saves the 'formulae' with 'blanks' in the form of $1,$2,$3.. that we can fill later.

Save it and then call it whenever we need to use the function.

This is a more effective and versatile way.

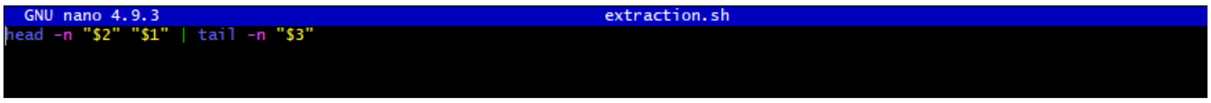
Below you can see the use of $nano to make a new shell called extraction.sh .



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ nano extraction.sh
```

We open the text editor and save the formulae. $2 saves the number of lines extracted from the top, $1 saves name of the file the extraction will take place, and $3 saves the no. of lines from bottom.

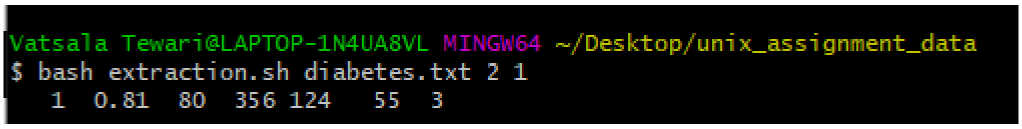
There is a reason why it is saved so, so that when we call the command later from the bash shell it makes sense.



```
  GNU nano 4.9.3                                  extraction.sh
head -n "$2" "$1" | tail -n "$3"
```

It makes sense, because this is what the following code means,

Bash should use extraction.sh 's saved formular and run it on diabetes to extract the first two lines and then the second last line.



```
Vatsala Tewari@LAPTOP-1N4UA8VL MINGW64 ~/Desktop/unix_assignment_data
$ bash extraction.sh diabetes.txt 2 1
   1   0.81   80   356 124    55   3
```

r assignment

Vatsala Tewari

18/11/2020

R

Exercise R.1

Define the vector x <- c(5,9,2,3,4,6,7,0,8,12,2,9) Decide what the result will be of the following: (a) x[2] (b) x[2:4] (c) x[c(2,3,6)] (d) x[c(1:5,10:12)] (e) x[-(10:12)] Use R to check your answers

Answer R.1

Variables store values is R. Vectors mentioned in the question are a list of variables. Long vectors are created by **concatenations**. Here the defining of the vectors is done by explicit decleration of the values.

a.[1] 9 b.[1] 9 2 3 c.[1] 9 2 6 d.[1] 5 9 2 3 4 12 2 9 e.[1] 5 9 2 3 4 6 7 0 8

```
 x <- c(5,9,2,3,4,6,7,0,8,12,2,9)
```

```
x[2]
x[2:4]
x[c(2,3,6)]
x[c(1:5,10:12)]
x[-(10:12)]
```

Exercise R.2

Run the following code in the R console:

```
n <- 1000
x <- seq(1,n)
sum(x)
```

Based on the result, what do you think the functions seq and sum do? You can use the help system. Explain your answer.

a)sum creates a list of numbers and seq adds them up.

b)seq creates a list of numbers and sum adds them up.

c)seq computes the difference between two arguments and sum computes the sum of 1 through 1000.

d)sum always returns the same number

*Hint. Also inspect the object X

Answer R.2

ANS. B)seq creates a list of numbers and sum adds them up. Sequence in the english language means “a particular order in which things follow each other”, since no other mathematical function is identified in the line x<-seq(1,n),**seq** will sequence numbers from 1,2,3. . . 1000, as n is defined as 1000. The sequence will be stored in the vector x.

hence, here n is defined as 1000, in the first line. x <- seq(1,n), uses the seq function to store first 1000 positive intergers to a vector x. then, sum(x) will sum up all the variables in vector x.

the mathematical formulae used for such calculations is 1000(1000+1)/2.

I used the help function and the formulae above to confirm my chain of thought.

```
help(seq)
```

```
## starting httpd help server ... done
```

```
help(sum)
```

Exercise R.3

Make a vector called `expression` containing the expression values in RPKM of 6 genes. The values are:0, 20,3,5,10,0.5

Answer R.3

The use of <- and c(concatenate) helps to define the vector expression. if this is done correctly can be seen in the global environment.

```
expression <- c(0,20,3,5,10,0.5)
```

Exercise R.3

The expression values we stored in `expression` are from the genes BMP2, NOTCH1, CDH11, FABP4, CLDN5, CLDN11.

> Create a vector with these gene names and call the object `genes` Now associate the two vectors by using the name function as below.

```
names(expression) <- genes
```

> Describe what the names function did

Answer R.3

The names function i believe gave the numbers in expression vector, which had the similar lenght of the vectors, the cprresponding characters names saved in the genes vector. Since, in the global environment we can see the change from **num:** to **named num:**

```r
genes <- c("BMP2" , "NOTCH1" , "CDH11" , "FABP4" , "CLDN5" , "CLDN11")
```

```r
names(expression) <- genes
help(names)
```

Exercise R.4

Use the [operator to access the expression of > the first three genes in the list, which are already stored in expression > the expression of the genes BMP2 and CDH11

Answer R.4

Using the Following code, we can access **just** the expression of the first three genes.Using the *unname* function, the names of the genes can be removed.I have used the [] operator to define the series of the numbers for which the expression is required.

if both expression and name are required then the unname function can be removed.

```r
unname(expression[1:3])
```

```
## [1]  0 20  3
```

```r
unname(expression[c("BMP2","CDH11")])
```

```
## [1] 0 3
```

Exercise R.5

Create a dataframe called `gene_expression` using genes and relative expression. Hint use the data.frame() function.

Answer R.5

The code helps us form a data frame, and to check if the code works the **str** function is used.

```r
gene_expression <- data.frame("gene" = genes , "expression" = unname(expression))
str(gene_expression)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ gene      : chr  "BMP2" "NOTCH1" "CDH11" "FABP4" ...
##  $ expression: num  0 20 3 5 10 0.5
```

Exercise R.6

Download a csv file from the following link:

https://emckclac-my.sharepoint.com/:x:/g/personal/k1642876_kcl_ac_uk/ETDCdwOlqMpBpL8o0DlFTV0BtcGm_2YRofvwEeTzK6d-lg?e=f1j0a7

Create a folder `R_assignment` and set this as your working directory.

Create a R object called `cell_features` containing this file using the read.csv function

Answer R.6

I will create a folder for me to to download the data to using the following

```
dir.create("R_assignment")
```

```
## Warning in dir.create("R_assignment"): 'R_assignment' already exists
```

Now, i will update my working directory, in the set up chunk with the following code, **knitr::opts_knit$set(root.dir = "C:/Users/Vatsala Tewari/Desktop/R_assignment")**

I will download the data from the link given using the download.file(url,destination) function. Further i will save a copy in raw data folder further in the R_assignment folder.

I tried to use the following code and i tried to troubleshoot using the import dataset and by updating libraries, but after runing the following command my data_r.csv always gets saved with the link as the html and not with data.

The code isn't wrong as when i try to import the data via the given url using the import dataset function in the global environment again the same problem occurs.

I tried to load other links and they loaded fine.

Hence, below I show that I know the fundamentals to load the data via link. I have instead used to load the data via the normal path to download the data via user interface.

download.file(url = "https://emckclac-my.sharepoint.com/:x:/g/personal/k1642876_kcl_ac_uk/ETDCdwOlqMpBpL8o0DlF
2YRofvwEeTzK6d-lg?e=f1j0a7", destfile = "data_r.csv") is the code that should have worked.

```
#the demand is commented, so that it doesn't run.
#download.file(url = "https://emckclac-my.sharepoint.com/:x:/g/
#personal/k1642876_kcl_ac_uk/ETDCdwOlqMpBpL8o0DlFTV0BtcGm_2YRofv
#wEeTzK6d-lg?e=f1j0a7", destfile = "data_r.csv")
```

To load the data into R we use the read.csv function.

```
cell_features <- read.csv("data_r.csv")
```

Exercise R.6.1

Explore the dataset and give the following info

a. Number of columns and rows
b. The class of the values in each column

Answer R.6.1

1. number of columns and rows can be inspected using the dim function. Hence, Rows are 2963 and number of columns are 11
2. In the 11 columns the class of each column will be the following. a)ï..cell_line_id **INT** b)short_name **CHR** c)concentration **INT** d)batch **CHR** e)num_cells **INT** f)cell_area_mean **NUM** g)roundness_mean **NUM** h)nucleus_area_mean **NUM** i)nucleus_roundness_mean **NUM** j)proliferation_mean **NUM** k)clumps **NUM**

   The use of the following codes helped me arrive at this conclusion

```
dim(cell_features)
```

```
## [1] 2963   11
```

```
str(cell_features)
```

```
## 'data.frame':    2963 obs. of  11 variables:
##  $ ï..cell_line_id       : int  1396 1396 1396 1396 1396 1396 1396 609 609 609 ...
##  $ short_name            : chr  "BOB" "BOB" "BOB" "BOB" ...
##  $ concentration         : int  5 25 5 25 1 5 25 1 5 25 ...
##  $ batch                 : chr  "Batch KA" "Batch KA" "Batch KA" "Batch KA" ...
##  $ num_cells             : int  492 843 521 735 127 475 580 232 309 327 ...
##  $ cell_area_mean        : num  2.98 3.14 3.01 3.12 2.74 ...
##  $ roundness_mean        : num  0.487 0.471 0.489 0.483 0.577 ...
##  $ nucleus_area_mean     : num  2.22 2.33 2.24 2.33 2.05 ...
##  $ nucleus_roundness_mean: num  0.518 0.59 0.517 0.605 0.46 ...
##  $ proliferation_mean    : num  3.32 3.23 3.31 3.24 3.42 ...
##  $ clumps                : num  -0.0709 -0.0505 -0.08 -0.0362 -0.1305 ...
```

Exercise R.6.2

Use the accessor $ to extract the short name of the cell lines and assign them to the object `cell_lines`. What is the class of this object?

Answer R.6.2

```
The below code forms a structure called cell_lines and stores short names in it.
```

the structure function helps define class of the function as a character

```
cell_lines <- cell_features$short_name
str(cell_lines)
```

```
##  chr [1:2963] "BOB" "BOB" "BOB" "BOB" "BOB" "BOB" "BOB" "iasn_3" "iasn_3" ...
```

Exercise R.6.3

The `cell_features` dataframe contains a column called concentration which refers to different fibronectin concentration used as extracellular matrix for the cell lines. Using the str function you can see that the concentration column is numerical. Do you think this is correct or it should be changed to factors?

```
> If yes, how do you perform the conversion?
```

```
> If you converted the concentration column to factors, check the number of levels.
```

Answer R.6.3

Yes, I believe the concentration column should be saved as a factor, since the numbers can be better represented as levels. i used the following code to convert it to factors, **cell_features**$concentration < -factor(cell_{f}eatures$**concentration)** the $ represents what to look for in the cell_features table.

using the summary function the number of levels are 4. Namely, *87-0s*, *915-1s* , *985-5s* , *976-25s*. Hence the levels are 0,1,5,25. We can also use the nlevels function to find no.of levels

```
cell_features$concentration <- factor(cell_features$concentration)
summary(cell_features$concentration)
```

```
##   0   1   5  25
##  87 915 985 976
```

```
nlevels(cell_features$concentration)
```

```
## [1] 4
```

Exercise R.6.4

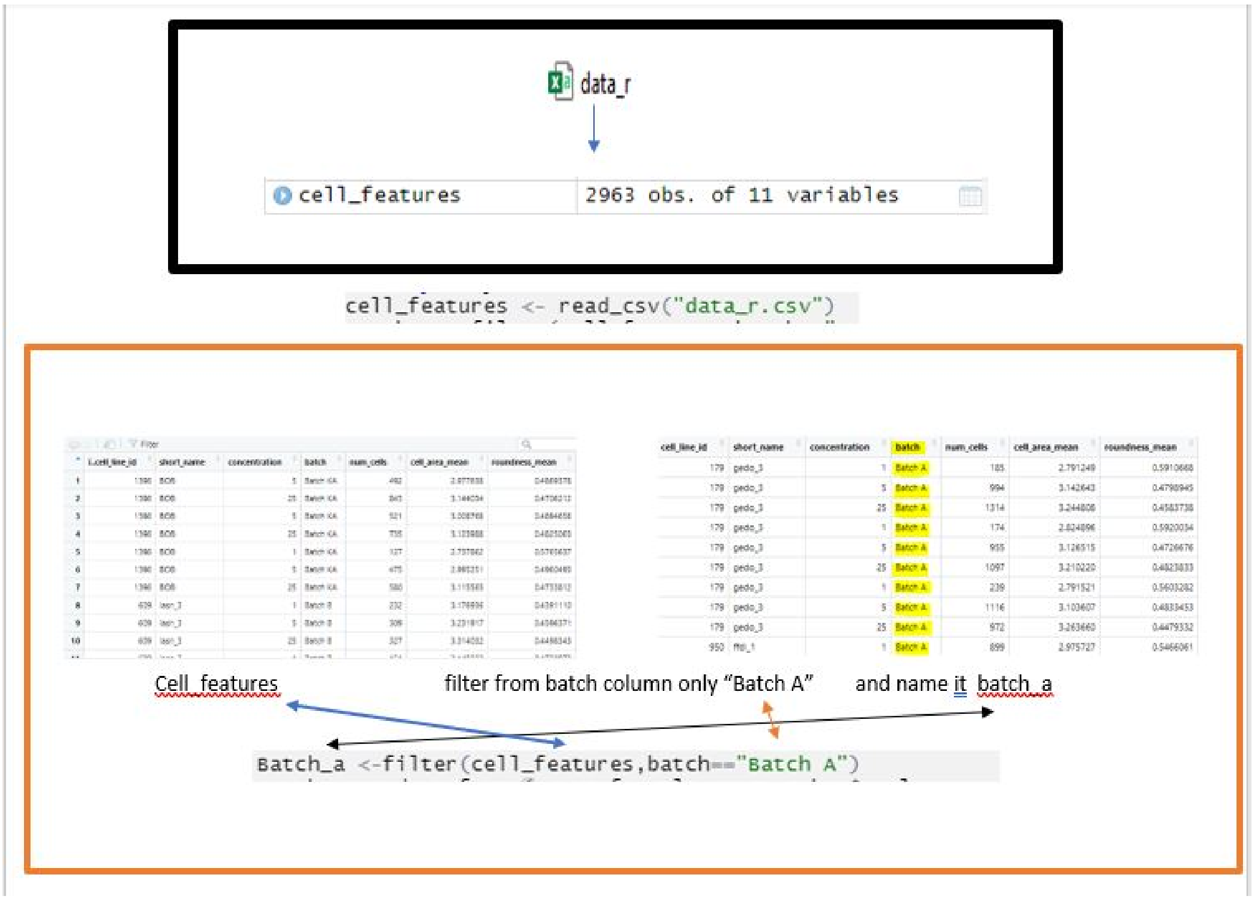
Create a new dataframe called `Batch_A` containing only rows with info about Batch A and with the columns describing nucleus measurements.

Answer R.6.4

To use some functions we need to download the tidyverse package. then, we read the table using the read_csv function, to load from file to R memmory. Storing it to cell_features, further we will use the filter function to filter from the cell_features table, the batch column only where the data is equal to (==) the string "Batch A", this is then stored in the batch_a file, then we create a data.frame, using the data from all those row's stored in batch_a, meaning all those rows where the batch was Batch A.

Using the data.frame we create a data frame containing the nucleus area mean and nucleus roundness from the batch_a file.

The following image summarises the codes.



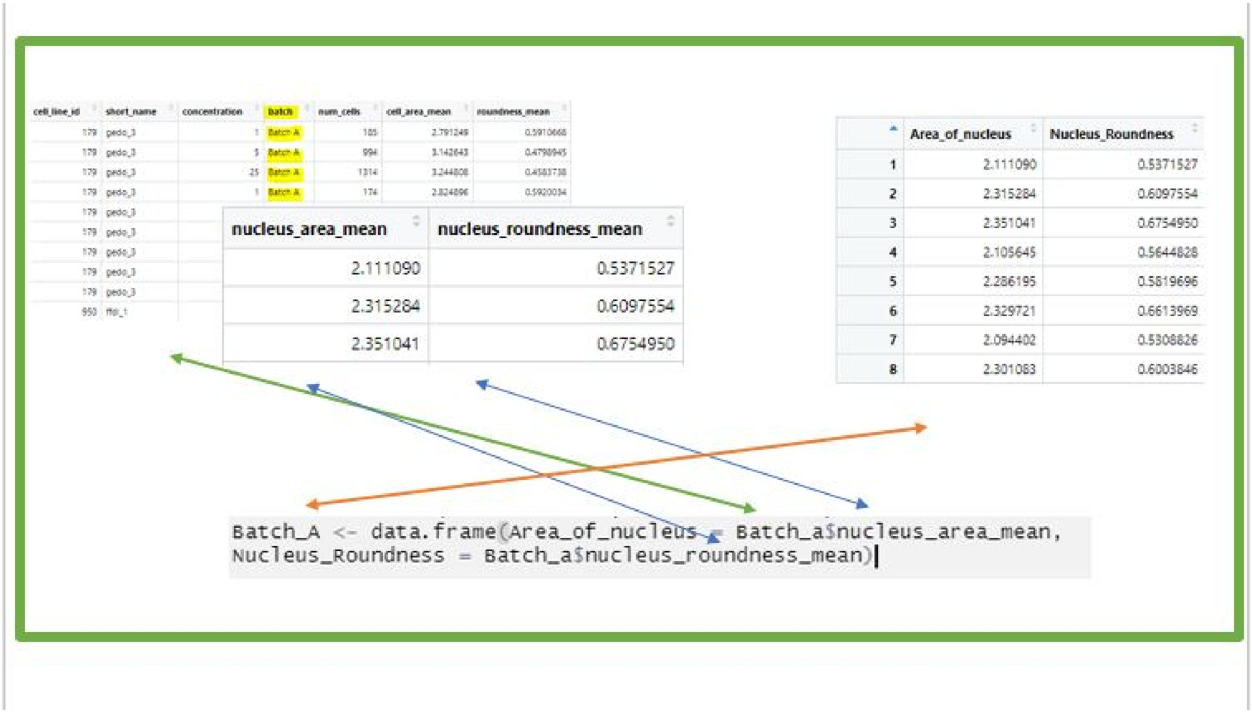
```
cell_features    2963 obs. of 11 variables
```

```r
cell_features <- read_csv("data_r.csv")
```

	i..cell_line_id	short_name	concentration	batch	num_cells	cell_area_mean	roundness_mean
1	1390	BOB	5	Batch KA	492	2.977838	0.4869378
2	1390	BOB	25	Batch KA	843	3.144034	0.4708212
3	1390	BOB	5	Batch KA	521	3.008768	0.4884658
4	1390	BOB	25	Batch KA	735	3.123888	0.4823065
5	1390	BOB	1	Batch KA	127	2.737862	0.5763637
6	1390	BOB	5	Batch KA	475	2.985251	0.4960485
7	1390	BOB	25	Batch KA	580	3.115585	0.4733812
8	639	lasn_3	1	Batch B	232	3.178936	0.4391110
9	639	lasn_3	5	Batch B	309	3.231817	0.4396371
10	639	lasn_3	25	Batch B	327	3.314032	0.4488343

cell_line_id	short_name	concentration	batch	num_cells	cell_area_mean	roundness_mean
179	gedo_3	1	Batch A	185	2.791249	0.5910668
179	gedo_3	5	Batch A	994	3.142643	0.4798945
179	gedo_3	25	Batch A	1314	3.244808	0.4583738
179	gedo_3	1	Batch A	174	2.824896	0.5920034
179	gedo_3	5	Batch A	955	3.126515	0.4726676
179	gedo_3	25	Batch A	1097	3.210220	0.4823833
179	gedo_3	1	Batch A	239	2.791521	0.5603282
179	gedo_3	5	Batch A	1116	3.103607	0.4833453
179	gedo_3	25	Batch A	972	3.263660	0.4479332
950	ffd_1	1	Batch A	899	2.975727	0.5466061

```r
Batch_a <-filter(cell_features,batch=="Batch A")
```



cell_line_id	short_name	concentration	batch	num_cells	cell_area_mean	roundness_mean
179	gedo_3	1	Batch A	185	2.791249	0.5910668
179	gedo_3	5	Batch A	994	3.142643	0.4798945
179	gedo_3	25	Batch A	1314	3.244808	0.4583738
179	gedo_3	1	Batch A	174	2.824896	0.5920034
179	gedo_3					
179	gedo_3					
179	gedo_3					
179	gedo_3					
179	gedo_3					
950	ffd_1					

nucleus_area_mean	nucleus_roundness_mean
2.111090	0.5371527
2.315284	0.6097554
2.351041	0.6754950

	Area_of_nucleus	Nucleus_Roundness
1	2.111090	0.5371527
2	2.315284	0.6097554
3	2.351041	0.6754950
4	2.105645	0.5644828
5	2.286195	0.5819696
6	2.329721	0.6613969
7	2.094402	0.5308826
8	2.301083	0.6003846

```r
Batch_A <- data.frame(Area_of_nucleus = Batch_a$nucleus_area_mean,
Nucleus_Roundness = Batch_a$nucleus_roundness_mean)
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
cell_features <- read_csv("data_r.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   cell_line_id = col_double(),
##   short_name = col_character(),
##   concentration = col_double(),
##   batch = col_character(),
##   num_cells = col_double(),
##   cell_area_mean = col_double(),
##   roundness_mean = col_double(),
##   nucleus_area_mean = col_double(),
##   nucleus_roundness_mean = col_double(),
##   proliferation_mean = col_double(),
##   clumps = col_double()
## )
```

```
Batch_a <-filter(cell_features,batch=="Batch A")
Batch_A <- data.frame(Area_of_nucleus = Batch_a$nucleus_area_mean, Nucleus_Roundness = Batch_a$nucleus_
```

Exercise R.6.5

Create a new dataframe called `high_cells` where the number of cells is higher than 500 in fibronectin concentration 1.

> How many unique cell lines have more than 500 cells in how many batches?

Answer R.6.5

1. Firstly we will store the information of those rows where the column of num_cells, has <500. Hence we use the code, in the 1st line in the chunk below, what this does is it creates a data frame - like a table, containing only the num_cells column from former cell_feature data frame, also due to the (<500) condition only those data where num_cells have more than 500 are taken. The '$dollar' sign represents "to go into" or "to extract".
2. The result for this will display all rows in cell_features only to show logicals, TRUE and FALSE. Although we only require the TRUE rows.
3. Hence, we not make a new data frame, asked in the question, high_cells, where we pipe the cells data to further filter only that data that has the string TRUE printed in the column.

4. Answer 1688 To count these values we can use nrow to find number, or open the high_cells and see that the last value is no. 1668, also present in the global environment.

```r
cells <- data.frame(cell_features$num_cells<500)
high_cells<- data.frame(cells %>%
  filter(cell_features.num_cells...500 == "TRUE"))
nrow(high_cells)
```

```
## [1] 1668
```

Exercise R.7

Load the libraries `readr` and `tidyverse` Starting from the same csv file `dataset_1.csv` create another R object using the `read_csv` function and call it `cell_features_tbl`. Tbl stands for tibble.

> What's the difference between a dataframe and a tibble?

Answer R.7

The library and the packages can be installed in the console, to not load the chunks again. We can do so by typing. library(readr) library(tidyverse) in the console. I have commented this command as I have loaded it in the console.

Here the dataset_1.csv is called data_r.csv. i read the data_r.csv to the cell_features_tbl function.

```r
#library(readr)
#library(tidyverse)
cell_features_tbl <- read_csv("data_r.csv")
```

data frame and tibble differences: 1. tibble's data frame is formed via data_frame()/read_csv, whereas in data.frame()/read.csv. 2. tibble is simpler to work with, as it doesnt change the inputs (strings to factors), allows name of the tibble to start with a number. for e.g. 500fibroniectincells could be a tibble, but would be a error in data frame generation. 3.when you print the table or str(), it shows you the rows and columns that fit on your screen, with type of column e.g col_double for numeric data, it does this by guessing the data type after forming a analysis of the first 1000 rows.

Exercise R.7.1

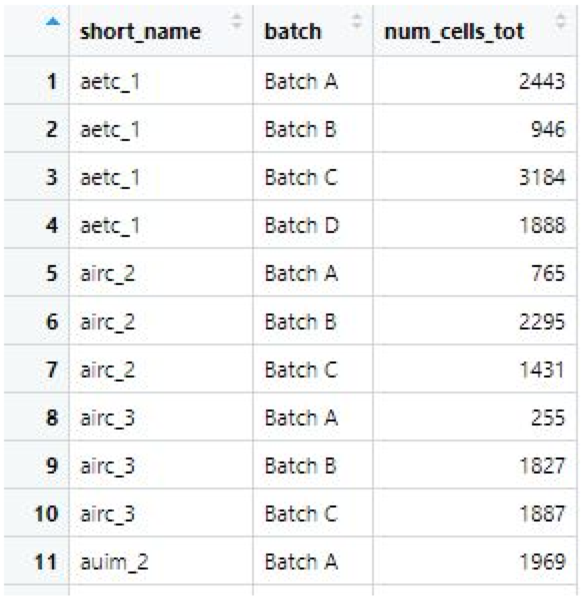
Using the output from the first exercise create a new object called `cell_features25` using the following commands and using pipes %>%

1. filter only the rows containing info about fibronectin 25 Hint. remember the `==` sign
2. select only the columns short_name, batch, num_cells, cell_area_mean
3. group by short_name and batch
4. create a new column called num_cells_tot containing the sum of the number of cells for each group of cell lines and batches

Answer R.7.1

We again us the filter command to identify the concentration as 25 in cell__features__tbl, pipe this to next function which selects the short__name, batch , num__cells , cell__area__mean. we further use this commands results as the input for the next command using the pipe function. Here, we group the data by short__name and batch, further piping the concised results to the summarise function, which creates a column name as num__cells__tot to sum the values in thier num__cells column.

the output is shown as such after the command chunk below is run



	short_name	batch	num_cells_tot
1	aetc_1	Batch A	2443
2	aetc_1	Batch B	946
3	aetc_1	Batch C	3184
4	aetc_1	Batch D	1888
5	airc_2	Batch A	765
6	airc_2	Batch B	2295
7	airc_2	Batch C	1431
8	airc_3	Batch A	255
9	airc_3	Batch B	1827
10	airc_3	Batch C	1887
11	auim_2	Batch A	1969

```
Cell_features25 <- filter( cell_features_tbl,concentration=="25") %>%
    select(short_name, batch, num_cells,cell_area_mean)%>%
   group_by(short_name, batch)%>%
  summarise(num_cells_tot = sum(num_cells))
```

Exercise R.7.2

Do a similar exercise as before starting from the original `cell_features` table where you ask a question yourself.

Answer R.7.2

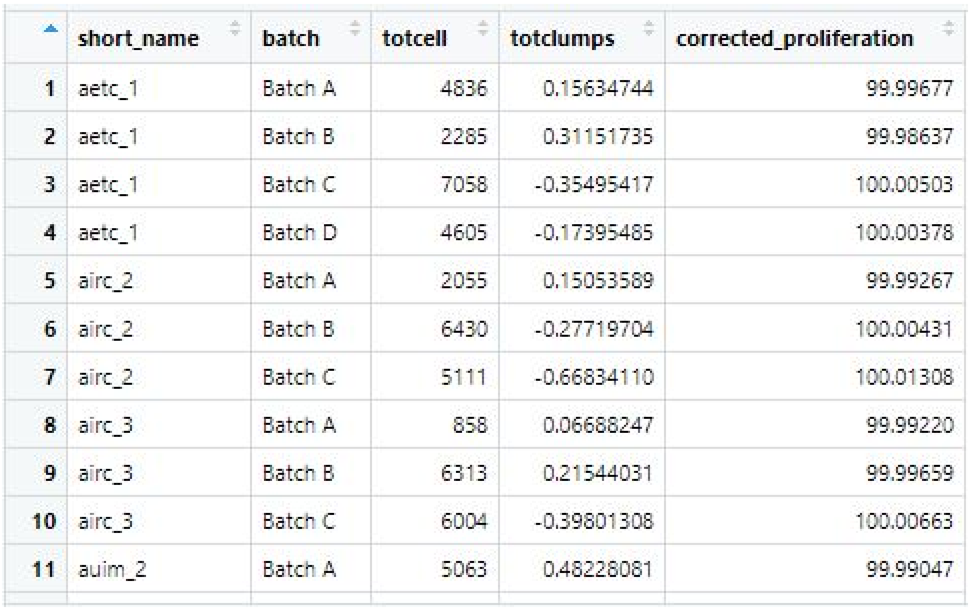
Here I asked the question, what is the actual proliferation of all cells?

1. Here i had taken an imaginary pre-requisite of a threshold, of 2. Hence my first line filters all those rows where the proliferation_mean is > "2". In biological samples we always take a threshold, here the threshold i set was 2.
2. Then we select using the next code line in the chunk below, only those which we need first to compress the distributed data, hence short__name and batch, and the inputs for calculation, here for my imaginary hypothetical formulae, I need the input, num__cell and clumps.
3. I summarise and collapse the distribution using the sum function to columns totcell and totclumps.

4. I then mutate, or create a new column called corrected proliferation using the input to the formulae which is completely imaginary! ((total cell - totalclumps)/total cell)*100

```r
Cell_feature_new <- filter(cell_features,proliferation_mean >"2")%>%
  select(short_name,batch,num_cells,clumps)%>%
  group_by(short_name,batch)%>%
  summarise(totcell = sum(num_cells), totclumps = sum(clumps)) %>%
  mutate(corrected_proliferation = (( ((totcell - totclumps)/totcell)*100 )))
```

```
## 'summarise()' regrouping output by 'short_name' (override with '.groups' argument)
```



	short_name	batch	totcell	totclumps	corrected_proliferation
1	aetc_1	Batch A	4836	0.15634744	99.99677
2	aetc_1	Batch B	2285	0.31151735	99.98637
3	aetc_1	Batch C	7058	-0.35495417	100.00503
4	aetc_1	Batch D	4605	-0.17395485	100.00378
5	airc_2	Batch A	2055	0.15053589	99.99267
6	airc_2	Batch B	6430	-0.27719704	100.00431
7	airc_2	Batch C	5111	-0.66834110	100.01308
8	airc_3	Batch A	858	0.06688247	99.99220
9	airc_3	Batch B	6313	0.21544031	99.99659
10	airc_3	Batch C	6004	-0.39801308	100.00663
11	auim_2	Batch A	5063	0.48228081	99.99047

Exercise R.7.3

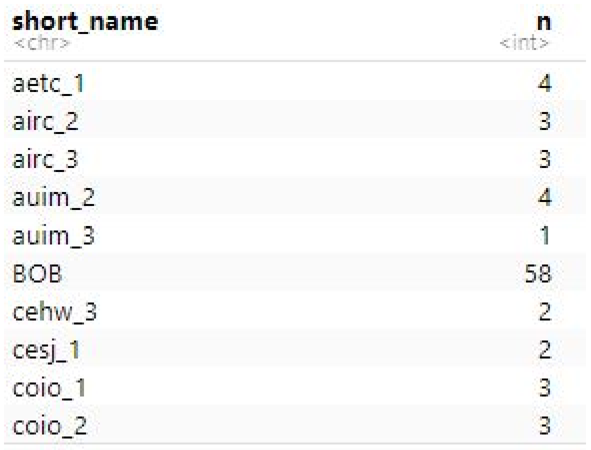
Use the count function to count how many rows you have for each short_name and batch.

Answer R.7.3

The following code counts no. of rows that collapsed into one for each short_name.

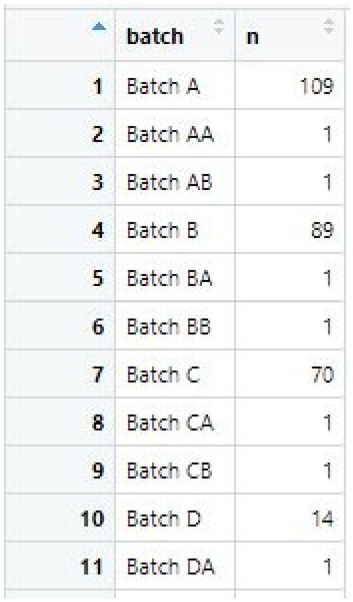
```r
library(tidyverse)
Cell_feature_new %>%
  count(short_name)
```

```
## # A tibble: 110 x 2
## # Groups:   short_name [110]
##    short_name     n
##    <chr>      <int>
##  1 aetc_1         4
##  2 airc_2         3
##  3 airc_3         3
##   4 auim_2         4
##   5 auim_3         1
##   6 BOB           58
##   7 cehw_3         2
##   8 cesj_1         2
##   9 coio_1         3
##  10 coio_2         3
## # ... with 100 more rows
```



short_name <chr>	n <int>
aetc_1	4
airc_2	3
airc_3	3
auim_2	4
auim_3	1
BOB	58
cehw_3	2
cesj_1	2
coio_1	3
coio_2	3

```r
count_batch <- Cell_feature_new %>%
  group_by(batch)%>%
  count(batch)
```



	batch	n
1	Batch A	109
2	Batch AA	1
3	Batch AB	1
4	Batch B	89
5	Batch BA	1
6	Batch BB	1
7	Batch C	70
8	Batch CA	1
9	Batch CB	1
10	Batch D	14
11	Batch DA	1