

Basic Genomic Data Wrangling with R

Quantitative Biomedical Research Center (QBRC)
email: qbrc@hsph.harvard.edu

Quantitative Biomedical Research Center (qBRC)

Research service center established (<http://www.hsph.harvard.edu/qbrc>) to provide high-throughput data analysis services:

- **Cloud Application Development**
- **Big data analysis, data mining, and grant support**
- **Training**

Email us at: qbrc@hsph.harvard.edu for project inquiry

Workshop Aims

- Allow participants with limited programming background to start learning R
- Using common transcriptomic and genomic data files
- Provide basic background on the R language
- Demonstrate simple data manipulation and filtering techniques

Section 1: Introduction

Why learn programming?

Data size and scope is constantly growing.

Saving time- automating analysis

Saving face- reproducible analysis and findings

- (Hopefully!) less human error

Most things have tradeoffs. Questions to ask yourself:

- Is this something I will have to do often?
- Is it tedious?
- Do I already know how to do this?
 - For small tasks, probably easier/quicker to use what you are comfortable with

Writing the actual code

Do NOT use a word processing program (e.g. MS Word) to write your code

- Introduces special/unseen characters which break things!
- Also be careful of copy/paste from PDFs or from the web for similar reasons

Use a “code aware” editor-- offers highlighting and coloring which can be helpful

- Notepad++ for Windows
- TextMate, etc. for Mac

Even better-- use a **I**ntegrated **D**evelopment **E**nvironment (IDE)

- Offers improved syntax highlighting, auto-completion, debugging, etc.
- RStudio is very popular for R development (<https://www.rstudio.com/products/rstudio2/>)

Install R and RStudio

Install R

Mac:

<https://cran.r-project.org/bin/macosx/>

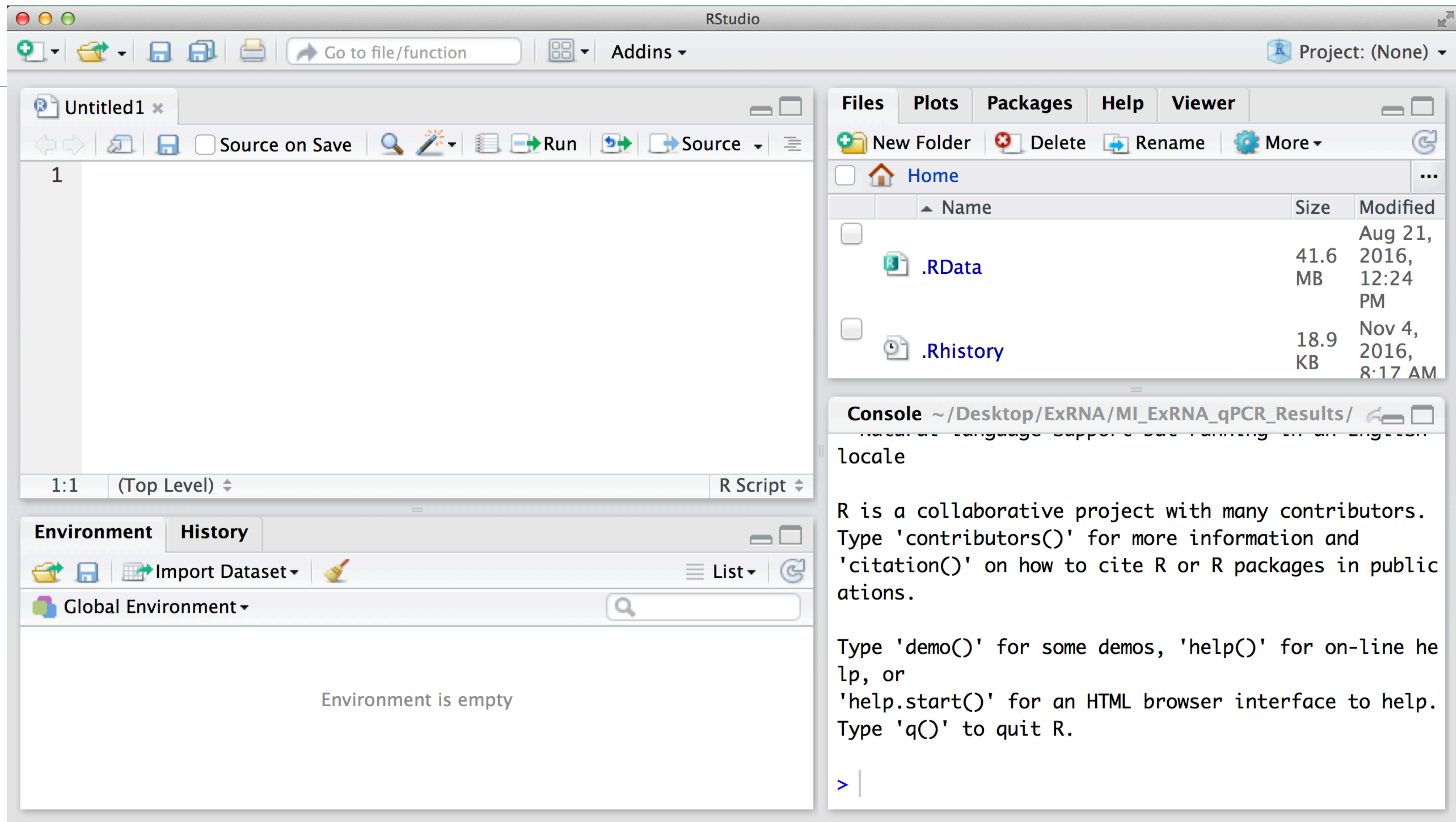
Windows:

<https://cran.r-project.org/bin/windows/base/>

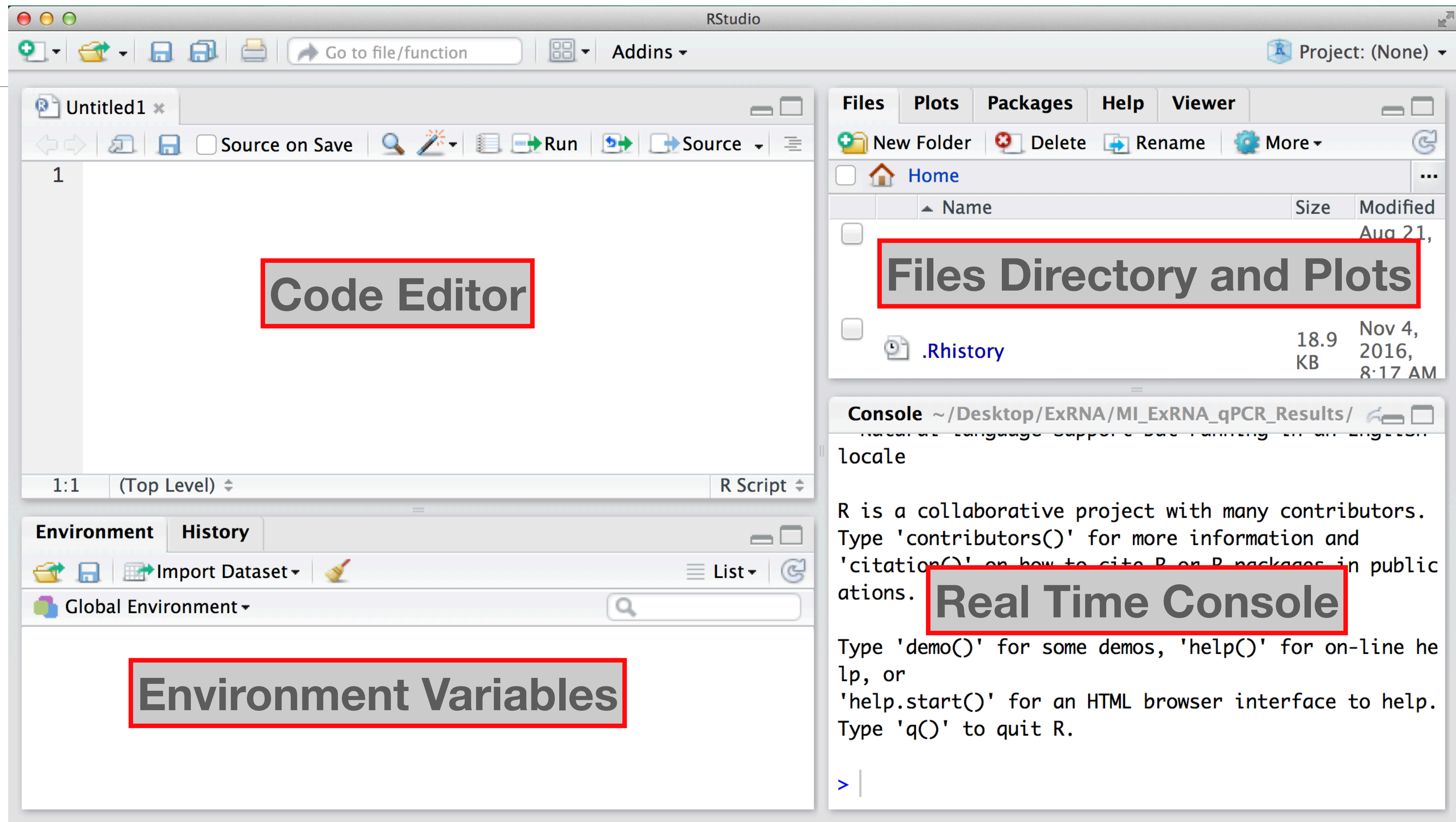
Install RStudio

<https://rstudio.com/products/rstudio/download/#download>

Getting familiar with your R installation and Rstudio



Getting familiar with your R installation and Rstudio



Coding- the importance of being exact

```
> f <-2  
> F + 1  
Error: object 'F' not found
```

Case matters!

Try not to use spaces

- Sometimes you simply cannot: variable names cannot contain spaces
- Even if it is valid (such as in file names) it just makes things more difficult

```
> FA <-2  
> F A + 1  
Error: unexpected symbol in "F A"
```

Data Structure - Atomic Vectors

Atomic Vectors - The basic variable unit of R represented in vector/array or matrix

```
> x=c(1,2,3,4)
> x
[1] 1 2 3 4
> dim(x)=c(2,2)
> x
  [,1] [,2]
[1,]  1  3
[2,]  2  4
> matrix(c(5,6,7,8), nrow=2, ncol=2)
  [,1] [,2]
[1,]  5  7
[2,]  6  8
> y=c("a", "b", "c", "d")
> y
[1] "a" "b" "c" "d"
```

Fundamental Operators

Arithmetic

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x % y	modulus (x mod y) 5%%2 is 1
x %/ y	integer division 5%/2 is 2

Logic

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

```
> x
  [,1] [,2]
[1,]  1  3
[2,]  2  4
> y
  [,1] [,2]
[1,]  5  7
[2,]  6  8
> x+y
  [,1] [,2]
[1,]  6 10
[2,]  8 12
> x[1,1]>y[1,1]
[1] FALSE
```

Data Structures - List

Allow for a single variable to store mixing data type

```
> y = list(a=1, 17, b=4:5, c="a")
```

```
> y
```

```
$a
```

```
[1] 1
```

```
[[2]]
```

```
[1] 17
```

```
$b
```

```
[1] 4 5
```

```
$c
```

```
[1] "a"
```

```
> y$a
```

```
[1] 1
```

```
> y[[1]]
```

```
[1] 1
```

```
> y[[2]]
```

```
[1] 17
```

```
> names(y)
```

```
[1] "a" "" "b" "c"
```

Data Structures - Data Frames

Special kind of list to store rectangular data sets. Think of it as excel sheet of R

```
> df.y$c
[1] a a
Levels: a
> y
$a
[1] 1
[[2]]
[1] 17
$b
[1] 4 5
$c
[1] "a"
```

```
> df.y=data.frame(y)
> df.y
  a X17 b c
1 1  17 4 a
2 1  17 5 a
> df.y$a
[1] 1 1
> df.y$a[1]
[1] 1
> df.y$c[1]
[1] a
Levels: a
> df.y$c
[1] a a
Levels: a
```

Data Structures - Factors

A data type unique to R and other statistical language, it condenses variables into unique categorical variables and store them as levels. Makes computation more efficient

```
> x=sample(letters[1:5], 10, replace = TRUE)
> x
[1] "e" "b" "b" "e" "c" "d" "b" "d" "a" "b"
> y=factor(x)
> y
[1] e b b e c d b d a b
Levels: a b c d e
> levels(y)
[1] "a" "b" "c" "d" "e"
> as.numeric(y)
[1] 5 2 2 5 3 4 2 4 1 2
```


R relies on its rich libraries



CRAN (The Comprehensive R Archived Networks)

- The primary R library archive
- Large number of statistical and machine learning packages available



Bioconductor

- R library archive specifically for Open Source Bioinformatics Software
- Largest collection of any Bioinformatics Library

Note: you will see the term ‘package’ and ‘library’ when talk about R. Package is the collection of function and library is where packages are archived.

A typical question we might ask from a Genomic Data Set

What is the expression of genes significantly upregulated in pathway of our interest that also contain Single Nucleotide Polymorphisms (SNPs)?

Rough steps:

1. Read data files
2. Filter for genes in a pathway
3. Filter for upregulated genes
4. Get the coordinates of those genes
5. Filter to keep only genes with SNPs
6. Get the expression for those genes across all samples
7. Write results to a file

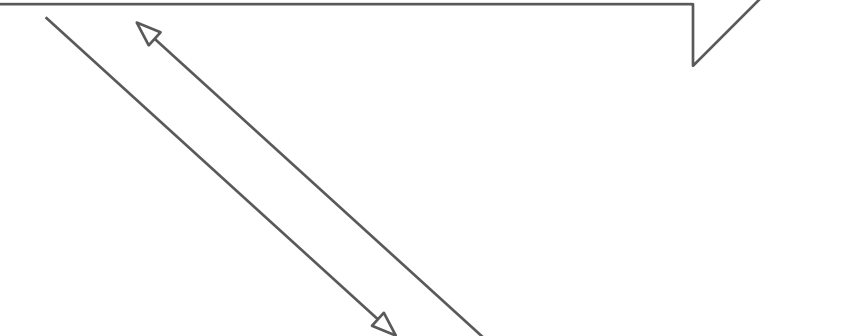
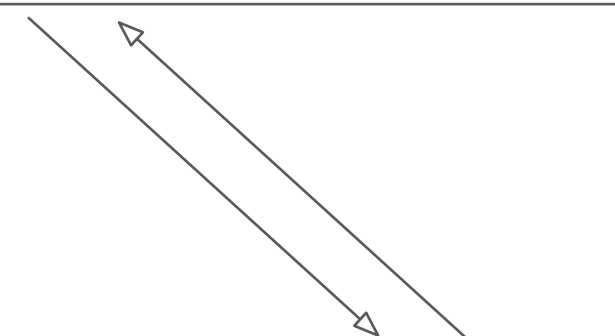
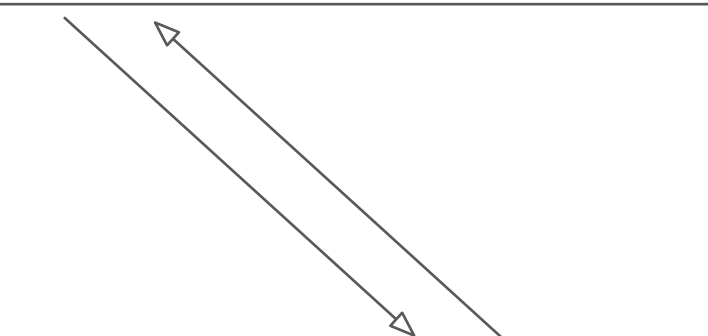
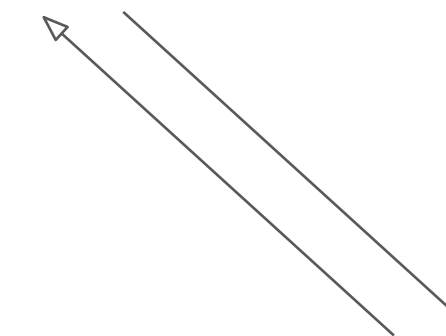
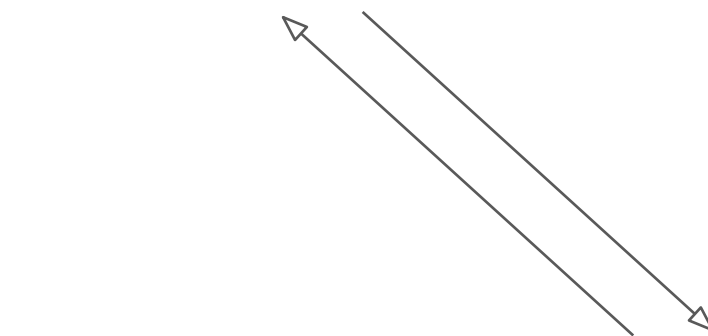
Section 2: Read in and filter data

Our map

1. Read data files
2. Filter for genes in a pathway
3. Filter for upregulated genes
4. Get the coordinates of those genes
5. Filter to keep only genes with SNPs
6. Get the expression for those genes across all samples
7. Write results to a file

Filtering

For loops, if statements



Reading files

Merging

Writing files

Raw data

Useful data

Reading files

Before we can work on the data, we need to load it in R...

```
read.table  
read.delim  
read.csv  
read.csv2  
read.delim2
```

These are built-in “functions” from R

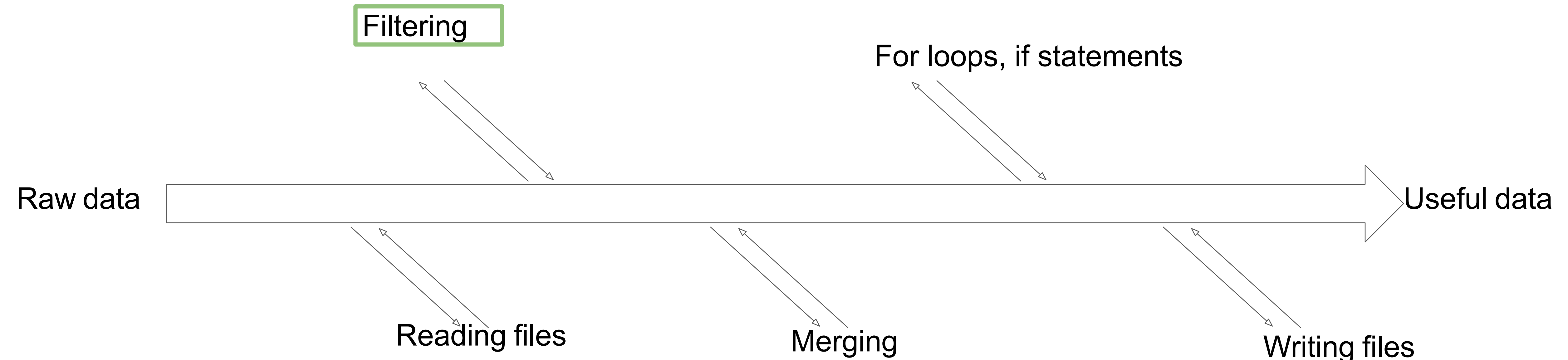
- encapsulates functionality, in this case opening, reading, closing a file

All read files, but with slightly different default behavior.

```
dge_results <- read.table('differential_results.csv',  
                          sep=',', header=T,  
                          stringsAsFactors=F)  
  
expressions = read.table('gene_expression.tsv',  
                          header=T, stringsAsFactors=F)  
  
annotations = read.table('gene_annotations.tsv',  
                          sep='\t', header=T,  
                          stringsAsFactors=F)  
  
pathways = read.table('my_pathway_genes.txt',  
                      sep='\t',  
                      col.names=c('gene_name', 'p_way'),  
                      stringsAsFactors=F)  
  
mutations = read.table('mutations.tsv',  
                       sep='\t', header=T,  
                       stringsAsFactors=F)
```

Data frames and filtering

1. ~~Read data files~~
2. Filter for genes in a pathway
3. Filter for upregulated genes
4. Get the coordinates of those genes
5. Filter to keep only genes with SNPs
6. Get the expression for those genes across all samples
7. Write results to a file



Reading files with read.table (or equivalents) automatically creates a dataframe for you.

```
expressions <- read.table('gene_expression.tsv', sep='\t',
stringsAsFactors=F)
```

	gene	SW1_Control	SW2_Control	...
1	KRAS	23	32	...
2	TP53	450	302	...
3	CDK9	102	99	...
...				

What is a DataFrame

R's version of a Excel spreadsheet- contains different data (of different types!) in the columns and “observations” in the rows.

patient_id	age	weight	smoker	...
532	42	67	no	...
745	35	102	yes	...

gene	S1	S2	S3	...
NRAS	102	100	300	...
CDK9	231	200	223	...

Slicing data

To refer to a cell in Excel, you're used to A1 (first row, first column), C2 (third row, second column). Similar with a DataFrame:

	gene	SW1_Control	SW2_Control	...
1	KRAS	23	32	...
2	TP53	450	302	...
3	CDK9	102	99	...
...				

Select the expression of TP53 in sample SW1_Control

```
expressions[2, 'SW1_Control']  
] expressions[2, 2]
```

Same results, which seems
“better” or less prone to
confusion?

Slicing data (cont'd)

To select entire rows or columns:

	gene	SW1_Control	SW2_Control	...
1	KRAS	23	32	...
2	TP53	450	302	...
3	CDK9	102	99	...
...				

Select first column:

```
expressions[1]  
expressions['gene']  
expressions[,1]  
expressions$gene
```

Select first row:

```
expressions[1,]
```

Note the result and what it looks like. The first two and last two give you different items

	gene	SW1_Control	SW2_Control	...
1	KRAS	23	32	...
2	TP53	450	302	...
3	CDK9	102	99	...
...

```
expressions[1]  
expressions[ 'gene'  
            ]
```

```
expressions[,1]  
expressions$gene
```

A “mini” one-column dataframe

	Gene
1	KRAS
2	TP53
3	CDK9
...	...

KRAS
TP53
CDK9
...

Just a list of the values

To select multiple rows or columns:

	gene	SW1_Control	SW2_Control	...
1	KRAS	23	32	...
2	TP53	450	302	...
3	CDK9	102	99	...
...				

Select first three columns (Gene, SW1_Control, SW2_Control):

```
expressions[1:3]
```

Select all columns except first:

```
expressions[2:ncol(expressions)]
```

or

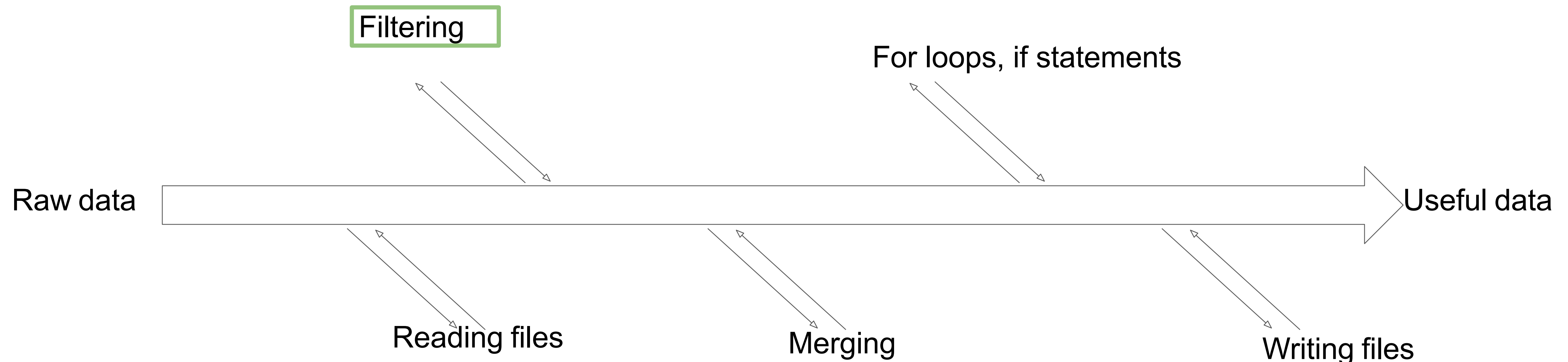
```
expressions[-1]
```

`ncol(expressions)` gets the number of columns in the `expressions` dataframe.

This way you do not need to know the size/shape of your dataframe ahead of time-- more flexible!

Data frames and filtering

1. ~~Read data files~~
2. Filter for genes in a pathway
3. Filter for upregulated genes
4. Get the coordinates of those genes
5. Filter to keep only genes with SNPs
6. Get the expression for those genes across all samples
7. Write results to a file



Select genes located on chromosome 7 from small gene annotation file 'demo_annotations.tsv'

How to run in excel

```
#select gene annotations that are on chromosome 7  
Open "demo_annotations.tsv" in excel  
Insert table  
Select 'chrom' column  
Set 'chrom' column equal 'chr7'
```

How to run in R

```
#select gene annotations that are on chromosome 7  
df=read.table("demo_annotations.tsv", header=T)  
chroms=df$chrom  
is_chr7 = chroms == 'chr7'  
is_chr7  
df[is_chr7,]
```

Commands Explained

demo_annotations.tsv (stored as df)

	chrom	start	end	strand	name
1	chr1	100	150	+	KRAS
2	chr7	200	275	-	TP53
3	chr8	300	350	-	CDK9
4	chr7	500	600	+	CD44

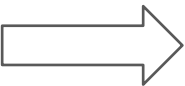
df[is_chr7,]

F
T
F
T

	chrom	start	end	strand	name
1	chr1	100	150	+	KRAS
2	chr7	200	275	-	TP53
3	chr8	300	350	-	CDK9
4	chr7	500	600	+	CD44

is_chr7 = df\$chrom == 'chr7'

F	=	chr1	==	chr7
T		chr7		chr7
F		chr8		chr7
T		chr7		chr7



	chrom	start	end	strand	name
2	chr7	200	275	-	TP53
4	chr7	500	600	+	CD44

Select genes located on chromosome 3 from gene annotations

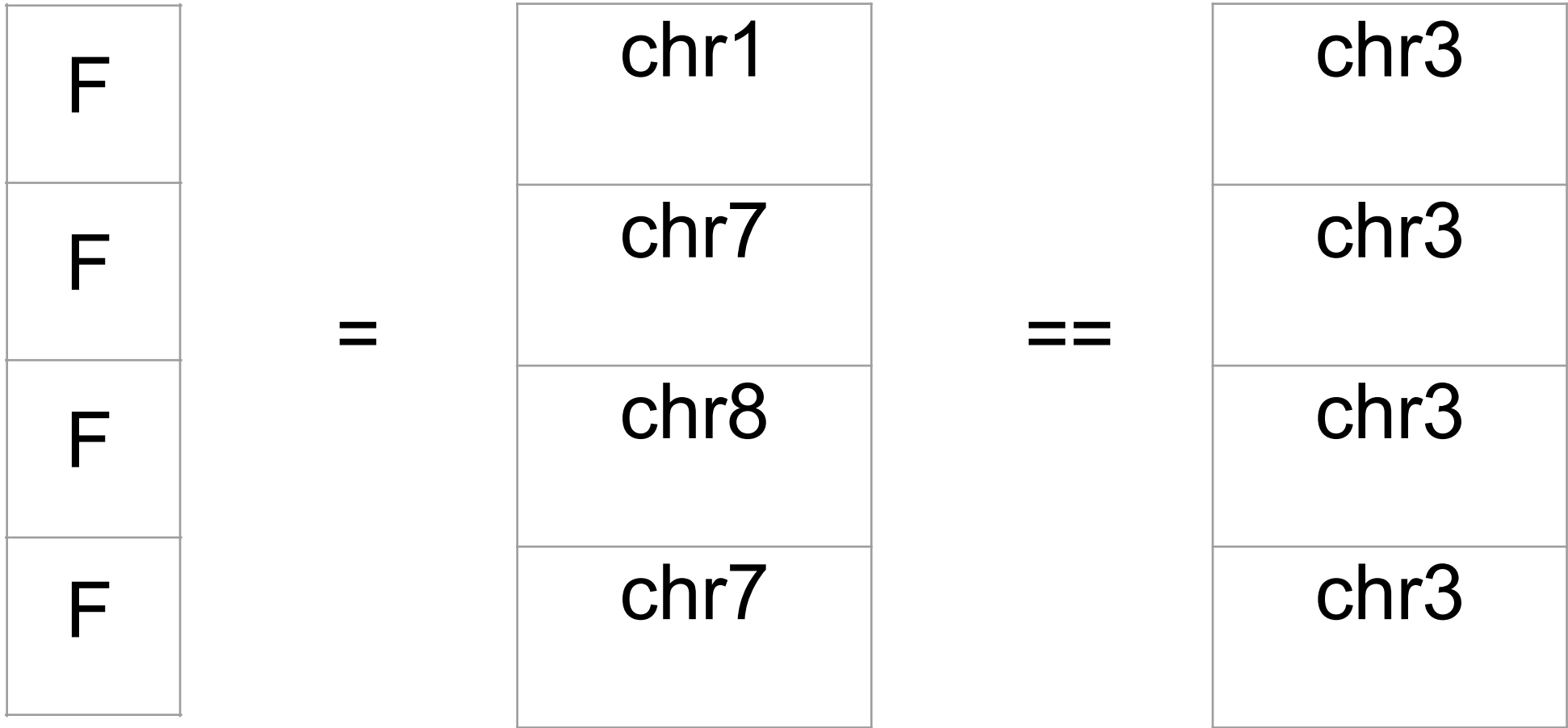
How to run in excel

```
#select gene annotations that are on chromosome 3  
Open "demo_annotations.tsv" in excel  
Insert table  
Select 'chrom' column  
Set 'chrom' column equal 'chr3'
```

How to run in R

```
#select gene annotations that are on chromosome 3  
# 'df' and 'chroms' variables are already assigned  
df=read.table("demo_annotations.tsv", header=T)  
is_chr3 <- chroms == 'chr3'  
df[is_chr3,]
```

```
is_chr3 = df$chrom == 'chr3'
```



```
df[is_chr3, ]
```

		chrom	start	end	strand	name
F	1	chr1	100	150	+	KRAS
F	2	chr7	200	275	-	TP53
F	3	chr8	300	350	-	CDK9
F	4	chr7	500	600	+	CD44

```
demo_annotations.tsv
```

	chrom	start	end	strand	name
1	chr1	100	150	+	KRAS
2	chr7	200	275	-	TP53
3	chr8	300	350	-	CDK9
4	chr7	500	600	+	CD44

➡ Nothing!

Select genes from gene annotations that are in your oncogene list ('KRAS', 'TP53')

How to run in excel

```
#select gene annotations for ('KRAS', 'TP53')  
Go to 'name' column  
Find 'KRAS'  
Click on 'KRAS'  
Find 'TP53'  
Click on 'TP53'
```

How to run in R

```
#select gene that are annotated as oncogenes use %in%  
oncogenes = c('KRAS', 'TP53')  
is_oncogene = df$name %in% oncogenes  
df[is_oncogene,]
```

Select genes from gene annotations that are in your oncogene list ('KRAS', 'TP53') and on chrom 7

How to run in excel

```
#select gene annotations for ('KRAS', 'TP53') and chrom 7  
Go to 'name' column  
Find 'KRAS'  
Click on 'KRAS'  
Find 'TP53'  
Click on 'TP53'  
Set 'chrom' column equal 'chr7'
```

How to run in R

```
#select gene annotations for ('KRAS', 'TP53') and chrom 7  
selection_criteria = is_oncogene & is_chr7  
data_subset = df[selection_criteria, ]  
data_subset
```

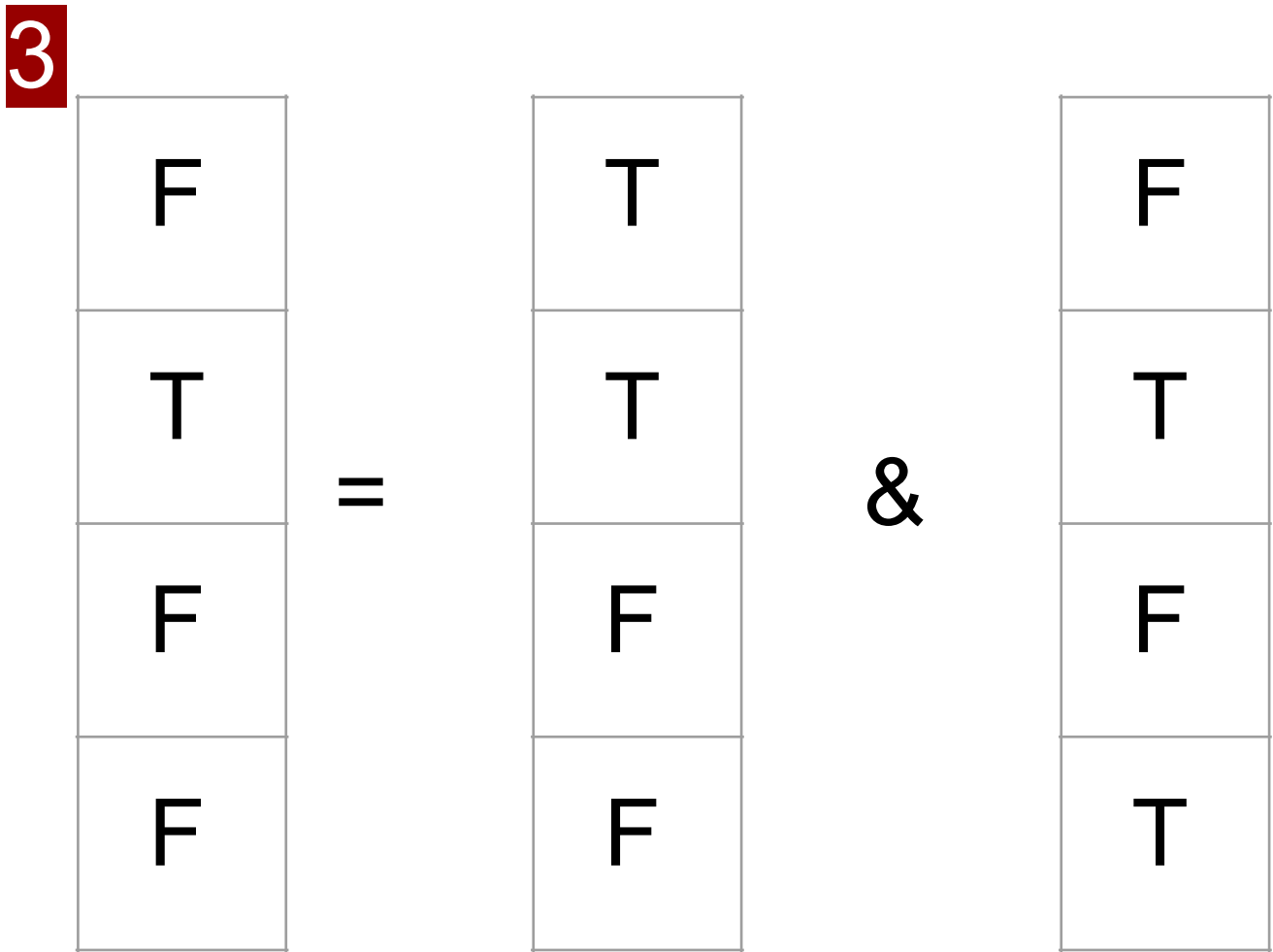
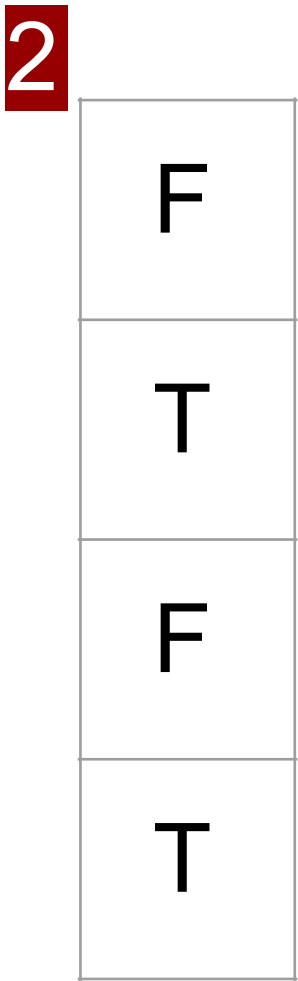
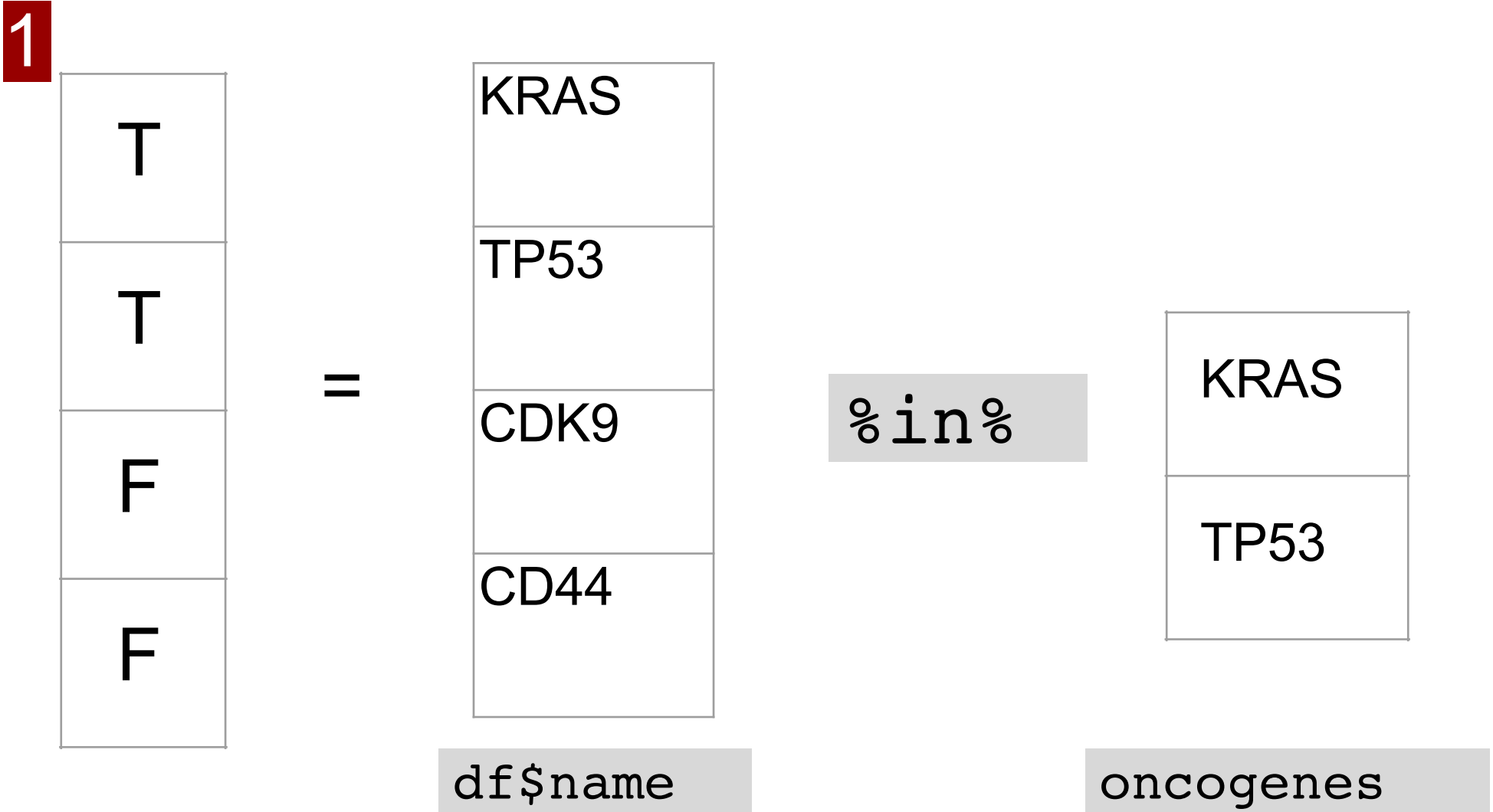
Filtering

```
oncogenes = c('KRAS', 'TP53')
is_oncogene = df$name %in%
oncogenes
is_chr7 = df$chrom == 'chr7'
```

```
selection_criteria = is_oncogene &
is_chr7
data_subset =
df[selection_criteria, ]
```

demo_annotations.ts

v	chrom	start	end	strand	name
1	chr1	100	150	+	KRAS
2	chr7	200	275	-	TP53
3	chr8	300	350	-	CDK9
4	chr7	500	600	+	CD44



Section 3: Wrangle a real data set

Find ras signaling genes that are significantly up-regulated from our result files

Steps to find these genes:

1. Get all the gene names that are in the pathway of interest (ras_signaling) from 'my_pathway_genes.tsv'
1. Find differential gene expression(DGE) results for those RAS genes from 'differential_results.csv'
2. Find all the differentially expressed genes an adjusted $p < 0.05$ threshold from 'differential_results.csv'
3. Find all the differentially expressed genes that are up-regulated with log fold change ($\log_2\text{FoldChange}$) > 0 from 'differential_results.csv'
4. Keep only genes that pass all 3 "tests" (are True for all 3 conditions)
5. Just in case, remove missing data

Find ras signaling genes that are significantly up-regulated from our result files

How to run in excel

Files too large
Too many steps
Too many genes to select

How to run in R

```
ras_genes = pathways[pathways['pathway'] == 'ras_signaling', 'gene_name']
is_ras_gene = dge_results$gene %in% ras_genes
is_significant = dge_results$padj < 0.05
is_upregulated = dge_results$log2FoldChange > 0
selected_rows = dge_results[is_ras_gene & is_significant & is_upregulated, ]
selected_rows = na.omit(selected_rows)

# For simplicity, let's keep only a subset of the columns from 'selected_rows'
upregulated_ras_genes=selected_rows[c('gene', 'baseMean', 'log2FoldChange', 'padj'
')] ]
```

Find ras signaling genes that are significantly up-regulated with gene coordinates from our result files

Steps to find these genes:

1. Get all the gene names that are in the pathway of interest (ras_signaling) from 'my_pathway_genes.tsv'
2. Find differential gene expression(DGE) results for those RAS genes from 'differential_results.csv'
3. Find all the differentially expressed genes an adjusted $p < 0.05$ threshold from 'differential_results.csv'
4. Find all the differentially expressed genes that are up-regulated with log fold change ($\log_2\text{FoldChange}$) > 0 from 'differential_results.csv'
5. Keep only genes that pass all 3 "tests" (are True for all 3 conditions)
6. Just in case, remove missing data
7. Merge result with 'gene_annotations.tsv' on the selected genes

Find ras signaling genes that are significantly up-regulated with gene coordinates and gene expression from our result files

Steps to find these genes:

1. Get all the gene names that are in the pathway of interest (ras_signaling) from 'my_pathway_genes.tsv'
2. Find differential gene expression(DGE) results for those RAS genes from 'differential_results.csv'
3. Find all the differentially expressed genes an adjusted $p < 0.05$ threshold from 'differential_results.csv'
4. Find all the differentially expressed genes that are up-regulated with log fold change ($\log_2\text{FoldChange}$) > 0 from 'differential_results.csv'
5. Keep only genes that pass all 3 "tests" (are True for all 3 conditions)
6. Just in case, remove missing data
7. Merge result with 'gene_annotations.tsv' on the selected genes
8. Merge result with 'gene_expression.csv' on the selected genes

Find ras signaling genes that are significantly up-regulated with gene coordinates and gene expression from our result files

How to run in excel

Files too large

Too many steps

Too many genes to select

Merge my result with annotations table?

– You can spend your day go through this Microsoft Power Query [tutorial](#)

Loading gene expression data in Excel will crush your computer

How to run in R

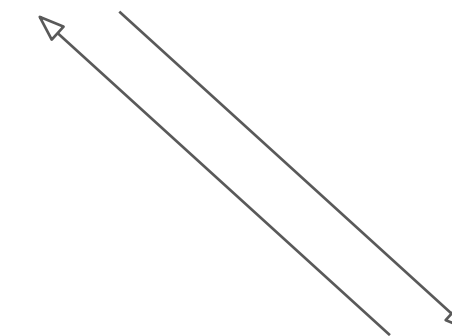
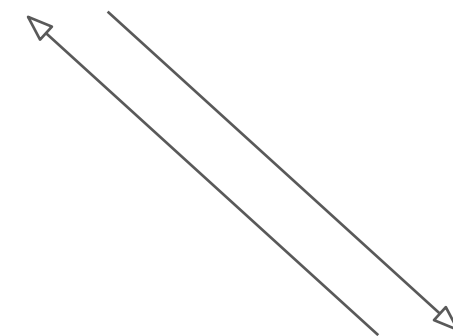
```
# Continue from Previous commands
ras_up_genes_w_coords_expression = merge(ras_up_genes_w_coords,
                                          expressions,
                                          by='gene' )
```

Section 4: For, if, and write results

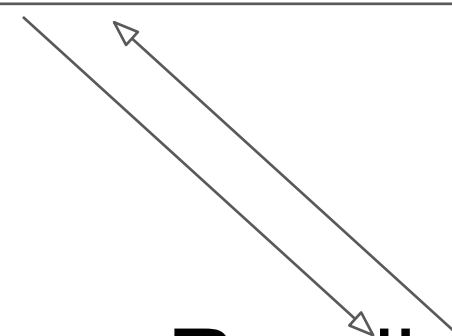
- ~~1. Read data files~~
- ~~2. Filter for genes in a pathway~~
- ~~3. Filter for upregulated genes~~
- ~~4. Get the coordinates of those genes~~
- ~~5.~~
6. Filter to keep only genes with SNPs
Get the expression for those genes across all samples
7. Write results to a file

For loops, if statements

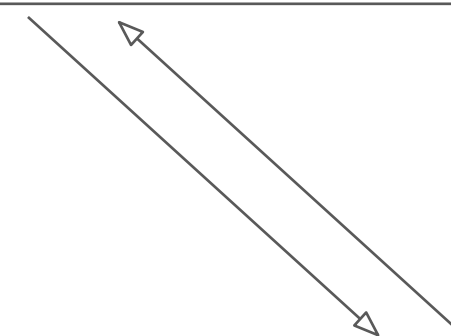
Filtering



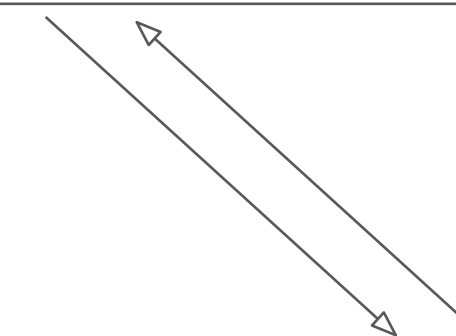
Reading files



Merging



Writing files



Raw data



Useful data

Integrating the SNP data with Expression data

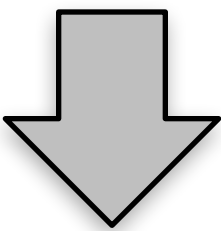
Problem: The chromosome notations are different and can not be merged

ras_up_genes_w_coords:

gene	baseMean	log2FoldChange	padj	chrom	start	end	strand
ANGPT1	1111.06165	0.5136897	2.78E-04	chr8	107249482	107498055	-
BCL2L1	10000.05911	0.3472426	8.74E-08	chr20	31664452	31723989	-
CALM2	73953.50812	0.27904	5.37E-06	chr2	47160082	47176601	-
...

mutations:

chrom	pos	ref	alt
5	20578198	C	T
2	4642922	T	A
...



Add 'chr' to each chrom value

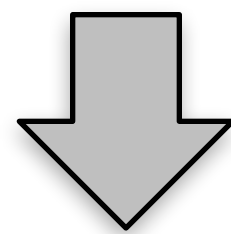
chrom	pos	ref	alt
chr5	20578198	C	T
chr2	4642922	T	A
...

Integrating the SNP data with Expression data

Problem: The chromosome notations are different and can not be merged

mutations:

chrom	pos	ref	alt
5	20578198	C	T
2	4642922	T	A
...



Add 'chr' to
each chrom
value

chrom	pos	ref	alt
chr5	20578198	C	T
chr2	4642922	T	A
...

Add 'chr' to '5' using **paste()**

```
i=1
chrom = mutations[i,'chrom']
chrom_w_prefix = paste('chr', chrom, sep="")
mutations[i,'chrom'] = chrom_w_prefix
```

Then we change each row one-by-one for 7479 times?
There is a better way to do this...

Note on paste():

`paste('x', 'y')` makes 'x y' (note the space), `paste('x', 'y', sep='_')` makes 'x_y', etc. The `sep=''` (empty single or double quotes) just sticks them together with nothing between

Use flow control - the “for loop”

```
my_vector <-c(10,11,12,13)
for (item in my_vector){
  print(item)

  #do actual operations here
}
```

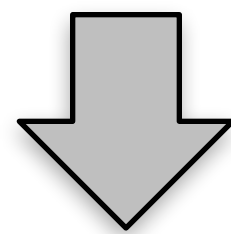
Notes:

- Everything between { and } is repeated
 - Called a “block” of code
 - Note indentation within the block is not necessary, but helps with reading
- The loop variable item is arbitrary (could call it whatever you like)
 - First time, item=10
 - Second time item=11
 - etc....
 -

“Fixing” the SNPs Data using for loop

mutations:

chrom	pos	ref	alt
5	20578198	C	T
2	4642922	T	A
...



Add 'chr' to
each chrom
value

chrom	pos	ref	alt
chr5	20578198	C	T
chr2	4642922	T	A
...

Changing all chromosome names in mutation with **nrow()**

```
for (i in 1:nrow(mutations)){  
    chrom =  
    mutations[i,'chrom']  
    chrom_w_prefix = paste('chr', chrom, sep='')  
    mutations[i,'chrom'] = chrom_w_prefix  
}
```

Filtering for polymorphic genes

ras_up_genes_w_coords:

gene	log2FoldChange	padj	chrom	start	end	strand
KRAS	2.3	0.0021	chr1	2000	5000	+
TP53	1.2	0.01	chr1	6500	7500	-

Idea:
For each row (a gene), see if any of the mutations are “inside” that gene e.g. Here, the first mutation is on the KRAS gene

How do we code that?

mutations:

chrom	pos	ref	alt
chr1	2200	A	G
chr1	6000	C	T

Final Result:

gene	log2FoldChange	padj	chrom	start	end	strand
KRAS	2.3	0.0021	chr1	2000	5000	+

Filtering for polymorphic genes

Idea:

For each row (a gene), see if any of the mutations are “inside” that gene?

```
print('Filter to keep only genes that are mutated.')

# Again, this a slow, but clear way to do this.
ras_up_mutated_genes = data.frame()                                # define an empty result variable
for ( i in 1:nrow(ras_up_genes_w_coords) )
{
  gene_info = ras_up_genes_w_coords[i,]                            # get row i and save as gene_info
  same_chrom = gene_info$chrom == mutations$chrom                  # get mutation row with same chrom
  past_start = mutations$pos > gene_info$start                      # get position > then gene_info start
  before_end = mutations$pos < gene_info$end                       # get position < then gene_info start
  overlap = same_chrom & past_start & before_end                   # get mutations fit all 3 conditions
  if ( any(overlap) )                                              # if there is any mutation quality
  {
    ras_up_mutated_genes =
      rbind(ras_up_mutated_genes, gene_info) # use rbind() to add to result variable
  }
}

print('Merge to add in the expression data')
# note no by.x or by.y since the 'gene' column is in each dataframe
final_data = merge(ras_up_mutated_genes, expressions)
```

Writing files

Now, we want to save our clean results to a file. Look at the help from R's built-in `write.table` function:

`write.table`
Data Output

package:utils

R Documentation

Description:

‘write.table’ prints its required argument ‘x’ (after converting it to a data frame if it is not one nor a matrix) to a file or connection.

Usage:

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = "", eol = "\n", na = "NA",
            dec = ".", row.names = TRUE, col.names = TRUE, qmethod = c("escape",
            "double"), fileEncoding = "")
write.csv2(...)
```


Writing files

```
write.table(final_data, "final_results.tsv", sep='\t', quote=F)
```

Add quotes? (Usually looks better without)



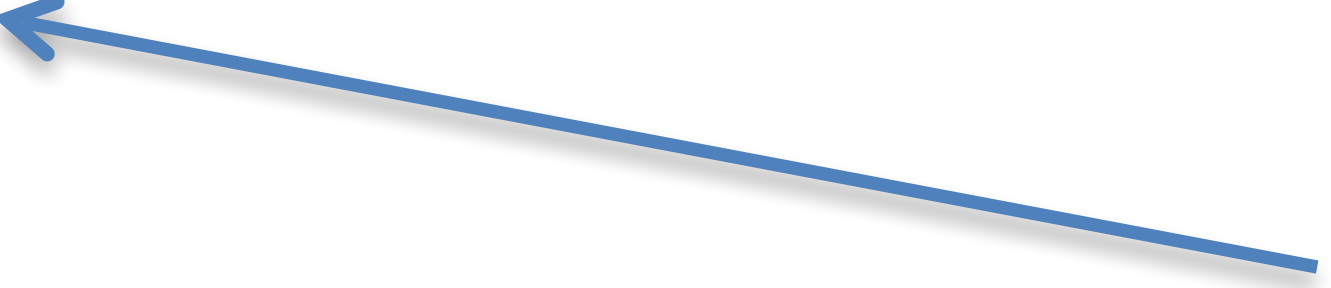
The dataframe we want to save



The location of the output file



How each column should be separated.
Commonly a comma (",") or tab ("\t")



Writing Excel File

- Requires installing package, use `install.packages` function

```
if(!require(readxl)) install.packages("readxl")
if(!require(writexl)) install.packages("writexl")

library(readxl)      # load library to write excel file
library(writexl)     # load library to write excel file
```

Writing Excel File

Write final_data into an excel file:

```
write_xlsx(final_data, "Outputs/final_results.xlsx", col_names=TRUE)
```



Create a new file

Write ras_up_genes_w_coords into the same file as a sheet:

```
write_xlsx(ras_up_genes_w_coords, "Outputs/ras_up_genes_w_coords.xlsx",  
col_names=TRUE)
```

In this session we learned

- How to navigate through RStudio
- Fundamental R data structures and operators
- Read in and write csv and excel data files
- Slice, filter, and merge data
- Loop through operations using 'for' loop

Final Write File script

```
if(!require(readxl)) install.packages("readxl")
if(!require(writexl)) install.packages("writexl")

library(readxl) # load library to write excel file
library(writexl) # load library to write excel file

# get current working directory.
getwd()

# create an output directory if it doesn't exist
if(!dir.exists('Outputs')){
  dir.create('Outputs')
}

# write final data out as a tab-seperated-value file (.tsv)
print('Write the final data to file.')
write.table(final_data, 'Outputs/final_data.tsv', sep='\t', quote=F)
print('Write the final data to Excel file.')

# start a new excel file and write final results
write_xlsx(final_data, "Outputs/final_results.xlsx", col_names=TRUE)

# add a new sheet with new data onto the file we just created
write_xlsx(ras_up_genes_w_coords, "Outputs/ras_up_genes_w_coords.xlsx", col_names=TRUE)
```