Statistical Computing and Data Visualization in R

Lecture 2

Lists

- A list is similar to a vector, but it can contain different types of elements (including vectors and other lists!)

```
> x <- list("California", c(1,3,5,1), 92, TRUE)
> x
[[1]]
[1] "California"

[[2]]
[1] 1 3 5 1

[[3]]
[1] 92

[[4]]
[1] TRUE
```

Lists

- You can also create an empty list with the `vector` function:

```
> x <- vector("list", length=2)
> x
[[1]]
[1] NULL

[[2]]
[1] NULL
```

- The sub-setting operator for lists is `[[]]`, and it can be used recursively with `[]`

```
> x <- list("California", c(1,3,5,1), 92, TRUE)
> x[[1]]
[1] "California"
> x[[2]][4]
[1] 1
```

Lists

- Be careful! If you use `[]` with a list you recover a list with a single element, rather than the element itself:

```
> x[1]
[[1]]
[1] "California”
```

- Unfortunately, the `[[]]` operator does not work like `[]`, e.g., multiple indexes are interpreted recursively

```
> x[[1:2]]          #Same as x[[1]][2]
Error in x[[1:2]] : subscript out of bounds
> x[[c(2,4)]]       #Same as x[[2]][4]
[1] 1
```

Lists

- If you want to recover a subset of elements of the list you can use the `[]`, but remember that you will get another list!

```
> x <- list("California", c(1,3,5,1),
92, TRUE)
> x[c(1,3)]
[[1]]
[1] "California"


[[2]]
[1] 92
```

Lists

- You can assign names to the elements of a list when it is created, which gives you other ways to access them:

```
> x <- list(state="CA",co=c(1,3,5,1),loc=92)
> x
$state
[1] "CA"

$co
[1] 1 3 5 1

$loc
[1] 92

> x$state
[1] "CA"
> x[["state"]]
[1] "CA"
> x["state"]
$state
[1] "CA"
```

Data Frames

- Data frames are similar to matrices, but different columns can contain data of different classes (but all entries in each column must be of the same class).

```
> x = data.frame(var1 = 1:4, var2 = c(T, T, F,
F), var3=factor(c(1,1,2,1)))
> x
  var1  var2 var3
1    1  TRUE    1
2    2  TRUE    1
3    3 FALSE    2
4    4 FALSE    1
```

- Data frames are usually constructed when loading data from file. The are infrequently constructed “by hand”.

Data Frames

- You can access elements of a data frame in the same way you access elements of a matrix.

```
> x = data.frame(var1 = 1:4, var2 = c(T, T, F, F), var3=factor(c(1,1,2,1)))
> x[2,3]
[1] 1
Levels: 1 2
> x[4,1]
[1] 4
```

- You can access whole columns in a couple of different ways

```
> x[,2]
[1]  TRUE  TRUE FALSE FALSE
> x$var2
[1]  TRUE  TRUE FALSE FALSE
> x[,"var2"]
[1]  TRUE  TRUE FALSE FALSE
```

Data Frames

- The function `subset` provides an alternative mechanism to select out certain rows and columns of the data frame.

```
> data(airquality)
> subset(airquality, Temp>80, select=
c(Ozone,Temp))
    Ozone Temp
29     45   81
35     NA   84
36     NA   85
```

- The main advantage of `subset` is that it allows you to use the name of the columns directly (easier to read)

```
> airquality[airquality$Temp>80, c("Ozone",
"Temp")]
    Ozone Temp
29     45   81
35     NA   84
36     NA   85
```

Data Frames

- Other examples of the use of `subset`

```
> subset(airquality, Day==1, select=-Temp)
    Ozone Solar.R Wind Month Day
1      41     190  7.4     5   1
32     NA     286  8.6     6   1
62    135     269  4.1     7   1
93     39      83  6.9     8   1
124    96     167  6.9     9   1
> subset(airquality, select=Ozone:Wind)
    Ozone Solar.R Wind
1      41     190  7.4
2      36     118  8.0
3      12     149 12.6
4      18     313 11.5
5      NA      NA 14.3
6      28      NA 14.9
7      23     299  8.6
```

Data Frames

- Sometimes it is easier to “attach” the data frames

```
> x = data.frame(var1 = 1:4, var2 = c(T, T, F, F),
var3=factor(c(1,1,2,1)))
> var1
Error: object 'var1' not found
> attach(x)
> var1
[1] 1 2 3 4
```

- Do not forget to detach the data frame when you are done. The columns in the data frame will mask other variables in your environment that have the same name.

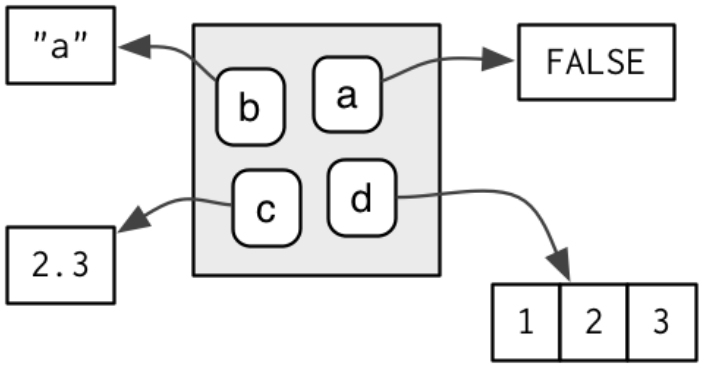
```
> detach(x)
> var1
Error: object 'var1' not found
```

- Changes made to the “attached” values are not stored!!!

Environments

- To understand what attach does it is useful to know about environments.
- An environment associates (binds) a set of names to a set of values:

```
> e <- new.env()
> e$a <- FALSE
> e$b <- "a"
> e$c <- 2.3
> e$d <- 1:3
```



- Similar to lists, but not quite the same.

Environments

- Each environment includes a pointer to a parent.
  - Environments form a tree.
  - The root of the tree is the empty environment (only environment without a parent).
  - The base environment is the child of the empty environment (contains the basic functions of the S language)
  - Environments associated with packages are added in the order in which they were loaded.
  - The global environment is the interactive environment in which you operate and contains your objects.

Environments

- When you type the name of an object, R searches first in the current environment. If it does not find it there, then goes to the parent environment and searches there. That process is iterated until you either find the object of reach the empty environment.
- You can see the environments in the search path by using the function `search()`.

Environments

- Objects in different environments might have the same name. If so, the one further up the tree (closer to the current environment) “masks” the others.
- `attach()` creates a new environment and makes copies of the columns of the data frame being attached.
  - The new environment, which has the same name as the original data frame, is added to the search tree so that it becomes the parent of the current environment.
- We will talk more about environments when we discuss scoping.

Transformations of variables

- The `transform` function can be used to easily transform columns of a data frame. By default, the transformation might replace the old values:

```
> data(airquality)
> airquality
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
3      12     149 12.6   74     5   3
4      18     313 11.5   62     5   4
5      NA      NA 14.3   56     5   5

> transform(airquality, Ozone=-Ozone)
    Ozone Solar.R Wind Temp Month Day
1     -41     190  7.4   67     5   1
2     -36     118  8.0   72     5   2
3     -12     149 12.6   74     5   3
4     -18     313 11.5   62     5   4
5      NA      NA 14.3   56     5   5
```

- Remember to save the transformed dataset!

Transformations of variables

- The function `transform` can also be used to add columns to the dataset:

```
> data(airquality)
> airquality
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
3      12     149 12.6   74     5   3
4      18     313 11.5   62     5   4
5      NA      NA 14.3   56     5   5

> transform(airquality, Temp.celcius=(Temp-32)/1.8)
    Ozone Solar.R Wind Temp Month Day Temp.celcius
1      41     190  7.4   67     5   1     19.44444
2      36     118  8.0   72     5   2     22.22222
3      12     149 12.6   74     5   3     23.33333
4      18     313 11.5   62     5   4     16.66667
5      NA      NA 14.3   56     5   5     13.33333
```

Aggregations

- Often we need to split the data in a data frame into subsets and compute summary statistics for each column on this subset.
- The function aggregate() simplifies this task!

```
> aggreage(state.x77, list(Region = state.region), mean)
          Region Population   Income Illiteracy  ...
1     Northeast   5495.111 4570.222   1.000000  ...
2         South   4208.125 4011.938   1.737500  ...
3 North Central   4803.000 4611.083   0.700000  ...
4          West   2915.308 4702.615   1.023077  ...
```

- You can get similar results (one column at a time) using `tapply()` (a variante of the `apply()` function we mentioned in lecture 1).

```
> tapply(state.x77[,1], state.region, mean)
    Northeast         South North Central          West
     5495.111      4208.125      4803.000      2915.308
```

- The aggregate() function can also be used with formulas! (We will study formulas later on in the course.

Reformatting your data

- When categorical predictors are present there are multiple ways in which it can be stored.
  - As a series of vectors, one for each combination of categorical predictors.
  - As a series vectors, one containing all the stacked data and the rest containing factors corresponding to each of the predictors.
  - As matrices or higher dimensional arrays, with each dimension corresponding to one categorical predictor.
- The second format is the most useful for working with R!

Reformatting your data

- You can move between the second and third data formats relatively easily:

```
> x = data.frame(matrix(seq(1, 120), ncol=2))
> x
   X1  X2
1   1  61
2   2  62
3   3  63
....

> z = stack(x)
> z
    values ind
1        1  X1
2        2  X1
3        3  X1
....

> w = unstack(z)
> w
   X1  X2
1   1  61
2   2  62
3   3  63
....
```

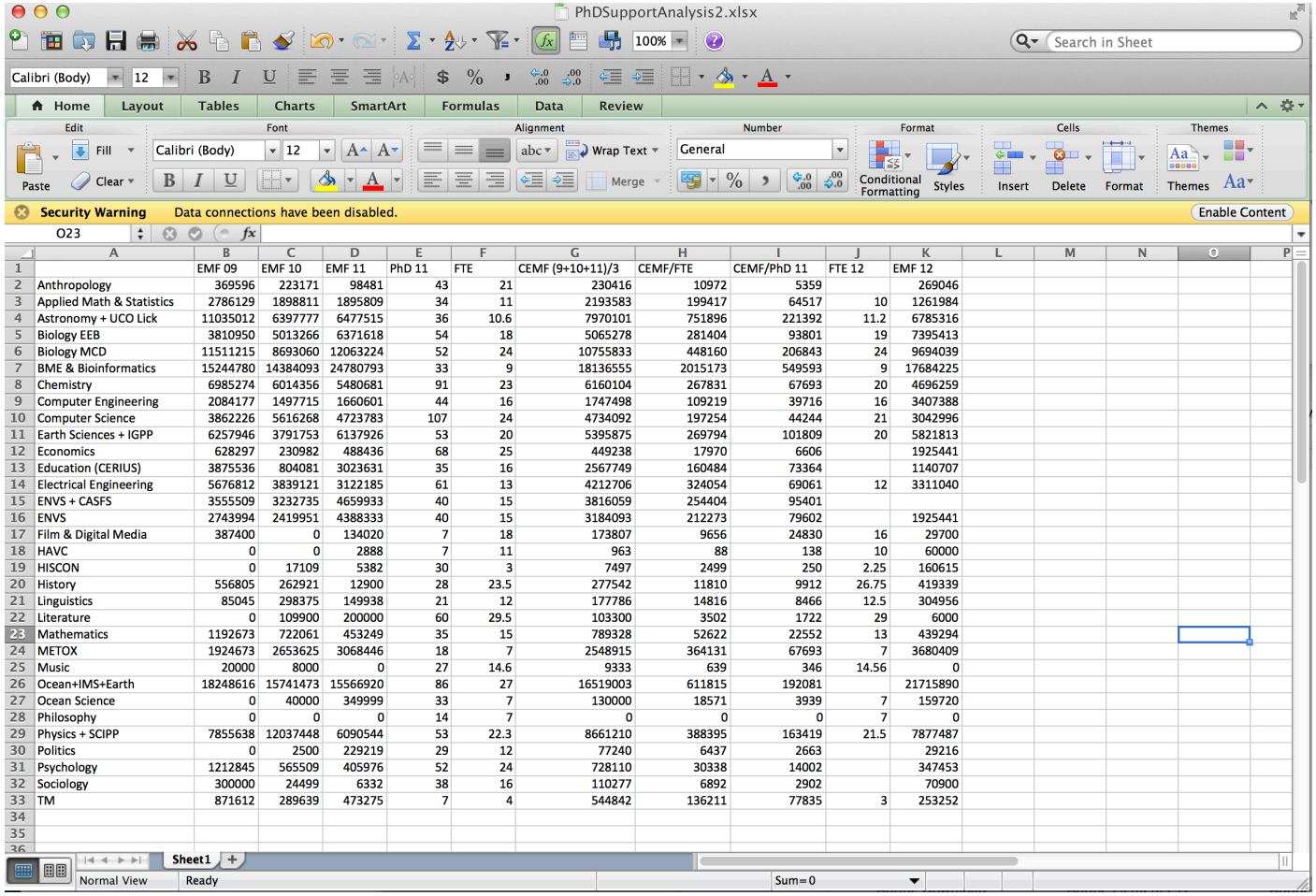
Working with external data

- In most circumstances you will not be typing you data directly into R, but you will be reading it from a file:
  - The most common formats are comma-delimited (csv) and tab-delimited text (ASCII) files. These are "universal" and be opened in any text processor. Usually these will be local to your own computer, but R can also read files that have been posted on the internet.
  - There are packages that allow you to read special formats such as Excel and SPSS.
  - There is functionality for accessing relational databases and perform queries.

Working with External Data: Preparing your text files for R

- Make sure that you have the same number of columns (sometime called "fields") on each row.
  - Add column names as appropriate.
  - If this a tab- or space-delimited file, empty cells (representing missing data) need to filled with the letters NA.
- Eliminate any special characters (such as accents, punctuation symbols and quotation marks).
- For column headers, eliminate blank spaces between words, or replace them with some other symbol (avoid "." and "_").
  - White spaces are ok in other places, as long as you use .csv files (more on this later).

Preparing your text files for R

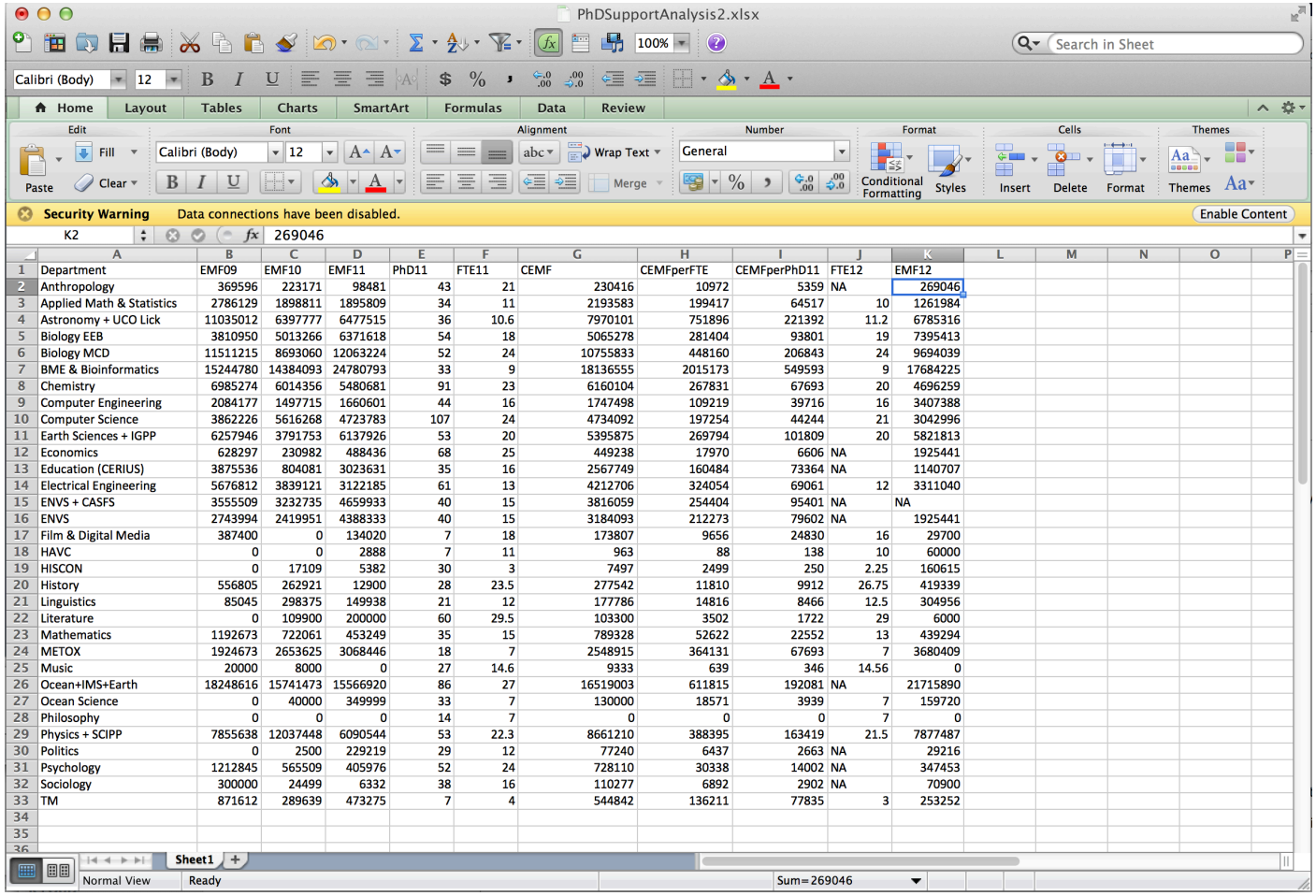


	A	B	C	D	E	F	G	H	I	J	K
1		EMF 09	EMF 10	EMF 11	PhD 11	FTE	CEMF (9+10+11)/3	CEMF/FTE	CEMF/PhD 11	FTE 12	EMF 12
2	Anthropology	369596	223171	98481	43	21	230416	10972	5359		269046
3	Applied Math & Statistics	2786129	1898811	1895809	34	11	2193583	199417	64517	10	1261984
4	Astronomy + UCO Lick	11035012	6397777	6477515	36	10.6	7970101	751896	221392	11.2	6785316
5	Biology EEB	3810950	5013266	6371618	54	18	5065278	281404	93801	19	7395413
6	Biology MCD	11511215	8693060	12063224	52	24	10755833	448160	206843	24	9694039
7	BME & Bioinformatics	15244780	14384093	24780793	33	9	18136555	2015173	549593	9	17684225
8	Chemistry	6985274	6014356	5480681	91	23	6160104	267831	67693	20	4696259
9	Computer Engineering	2084177	1497715	1660601	44	16	1747498	109219	39716	16	3407388
10	Computer Science	3862226	5616268	4723783	107	24	4734092	197254	44244	21	3042996
11	Earth Sciences + IGPP	6257946	3791753	6137926	53	20	5395875	269794	101809	20	5821813
12	Economics	628297	230982	488436	68	25	449238	17970	6606		1925441
13	Education (CERIUS)	3875536	804081	3023631	35	16	2567749	160484	73364		1140707
14	Electrical Engineering	5676812	3839121	3122185	61	13	4212706	324054	69061	12	3311040
15	ENVS + CASFS	3555509	3232735	4659933	40	15	3816059	254404	95401		
16	ENVS	2743994	2419951	4388333	40	15	3184093	212273	79602		1925441
17	Film & Digital Media	387400	0	134020	7	18	173807	9656	24830	16	29700
18	HAVC	0	0	2888	7	11	963	88	138	10	60000
19	HISCON	0	17109	5382	30	3	7497	2499	250	2.25	160615
20	History	556805	262921	12900	28	23.5	277542	11810	9912	26.75	419339
21	Linguistics	85045	298375	149938	21	12	177786	14816	8466	12.5	304956
22	Literature	0	109900	200000	60	29.5	103300	3502	1722	29	6000
23	Mathematics	1192673	722061	453249	35	15	789328	52622	22552	13	439294
24	METOX	1924673	2653625	3068446	18	7	2548915	364131	67693	7	3680409
25	Music	20000	8000	0	27	14.6	9333	639	346	14.56	0
26	Ocean+IMS+Earth	18248616	15741473	15566920	86	27	16519003	611815	192081		21715890
27	Ocean Science	0	40000	349999	33	7	130000	18571	3939	7	159720
28	Philosophy	0	0	0	14	7	0	0	0	7	0
29	Physics + SCIPP	7855638	12037448	6090544	53	22.3	8661210	388395	163419	21.5	7877487
30	Politics	0	2500	229219	29	12	77240	6437	2663		29216
31	Psychology	1212845	565509	405976	52	24	728110	30338	14002		347453
32	Sociology	300000	24499	6332	38	16	110277	6892	2902		70900
33	TM	871612	289639	473275	7	4	544842	136211	77835	3	253252
34											
35											
36											

Preparing your text files for R

- You need to add a title to the first column.
- The empty cells on the last two columns needs to be filled with “NA”.
- Blank spaces and symbols such as “+”, “/”, “(“ and “)” in the column names need to be eliminated.

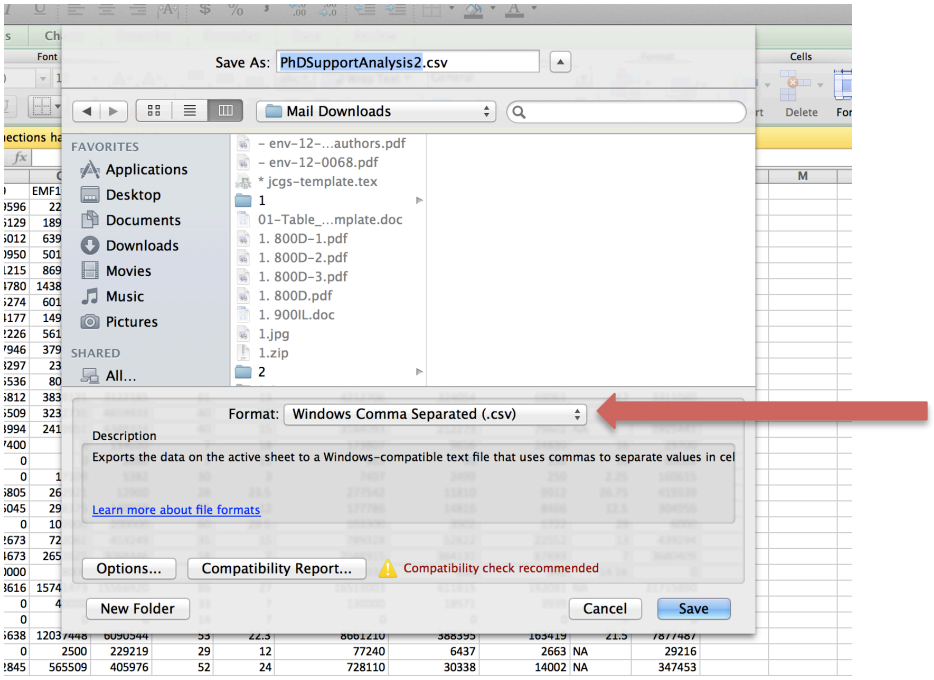
Preparing your text files for R



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Department	EMF09	EMF10	EMF11	PhD11	FTE11	CEMF	CEMFperFTE	CEMFperPhD11	FTE12	EMF12					
2	Anthropology	369596	223171	98481	43	21	230416	10972	5359	NA	269046					
3	Applied Math & Statistics	2786129	1898811	1895809	34	11	2193583	199417	64517	10	1261984					
4	Astronomy + UCO Lick	11035012	6397777	6477515	36	10.6	7970101	751896	221392	11.2	6785316					
5	Biology EEB	3810950	5013266	6371618	54	18	5065278	281404	93801	19	7395413					
6	Biology MCD	11511215	8693060	12063224	52	24	10755833	448160	206843	24	9694039					
7	BME & Bioinformatics	15244780	14384093	24780793	33	9	18136555	2015173	549593	9	17684225					
8	Chemistry	6985274	6014356	5480681	91	23	6160104	267831	67693	20	4696259					
9	Computer Engineering	2084177	1497715	1660601	44	16	1747498	109219	39716	16	3407388					
10	Computer Science	3862226	5616268	4723783	107	24	4734092	197254	44244	21	3042996					
11	Earth Sciences + IGPP	6257946	3791753	6137926	53	20	5395875	269794	101809	20	5821813					
12	Economics	628297	230982	488436	68	25	449238	17970	6606	NA	1925441					
13	Education (CERIUS)	3875536	804081	3023631	35	16	2567749	160484	73364	NA	1140707					
14	Electrical Engineering	5676812	3839121	3122185	61	13	4212706	324054	69061	12	3311040					
15	ENVS + CASFS	3555509	3232735	4659933	40	15	3816059	254404	95401	NA	NA					
16	ENVS	2743994	2419951	4388333	40	15	3184093	212273	79602	NA	1925441					
17	Film & Digital Media	387400	0	134020	7	18	173807	9656	24830	16	29700					
18	HAVC	0	0	2888	7	11	963	88	138	10	60000					
19	HISCON	0	17109	5382	30	3	7497	2499	250	2.25	160615					
20	History	556805	262921	12900	28	23.5	277542	11810	9912	26.75	419339					
21	Linguistics	85045	298375	149938	21	12	177786	14816	8466	12.5	304956					
22	Literature	0	109900	200000	60	29.5	103300	3502	1722	29	6000					
23	Mathematics	1192673	722061	453249	35	15	789328	52622	22552	13	439294					
24	METOX	1924673	2653625	3068446	18	7	2548915	364131	67693	7	3680409					
25	Music	20000	8000	0	27	14.6	9333	639	346	14.56	0					
26	Ocean+IMS+Earth	18248616	15741473	15566920	86	27	16519003	611815	192081	NA	21715890					
27	Ocean Science	0	40000	349999	33	7	130000	18571	3939	7	159720					
28	Philosophy	0	0	0	14	7	0	0	0	7	0					
29	Physics + SCIPP	7855638	12037448	6090544	53	22.3	8661210	388395	163419	21.5	7877487					
30	Politics	0	2500	229219	29	12	77240	6437	2663	NA	29216					
31	Psychology	1212845	565509	405976	52	24	728110	30338	14002	NA	347453					
32	Sociology	300000	24499	6332	38	16	110277	6892	2902	NA	70900					
33	TM	871612	289639	473275	7	4	544842	136211	77835	3	253252					
34																
35																
36																

Preparing your text files for R

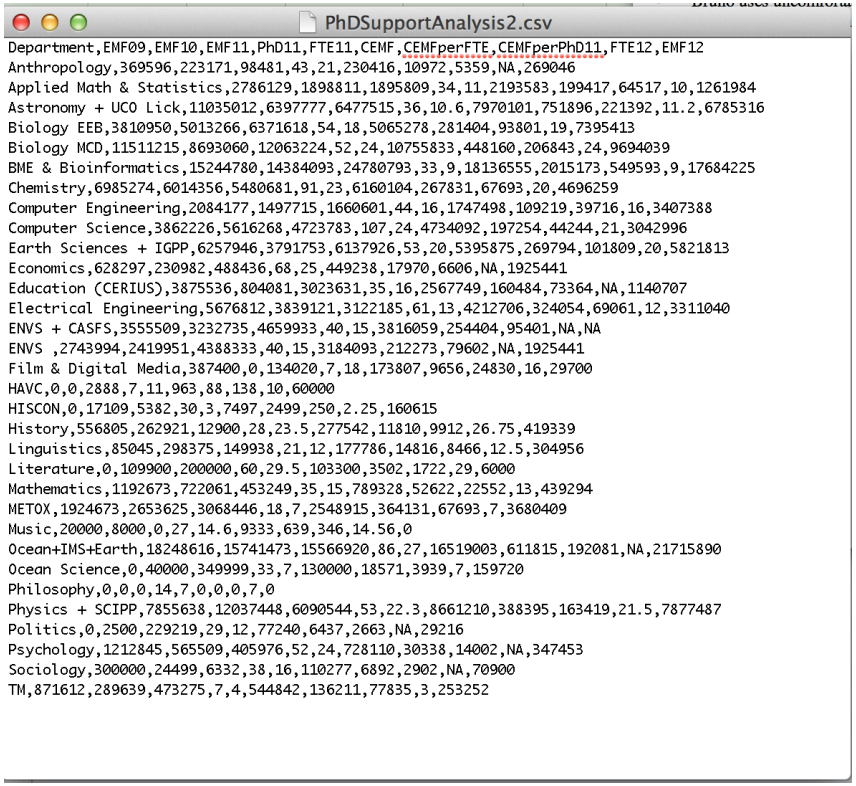
- Once your file is ready, make sure you save it in a format that R can read:
  - R will not read Excel files (.xls or .xlsx) or any other specialized format without a special library.
  - Instead use either tab separated text files (.txt) or **comma separated files (.csv)**.
- Use the option “Save as” in your “File” menu.



This is how it looks on a Mac!

Preparing your files for R

- Note that Excel does not show you the “real” internal structure of the .csv files.
- To see how they look on the inside open them with a plain text editor (*textedit* in Mac, *notepad* in Windows).
- Sometimes you need to edit the file manually and add quotation marks to the text to avoid errors.


PhDSupportAnalysis2.csv

```
Department,EMF09,EMF10,EMF11,PhD11,FTE11,CEMF,CEMFperFTE,CEMFperPhD11,FTE12,EMF12
Anthropology,369596,223171,98481,43,21,230416,10972,5359,NA,269046
Applied Math & Statistics,2786129,1898811,1895809,34,11,2193583,199417,64517,10,1261984
Astronomy + UCO Lick,11035012,6397777,6477515,36,10.6,7970101,751896,221392,11.2,6785316
Biology EEB,3810950,5013266,6371618,54,18,5065278,281404,93801,19,7395413
Biology MCD,11511215,8693060,12063224,52,24,10755833,448160,206843,24,9694039
BME & Bioinformatics,15244780,14384093,24780793,33,9,18136555,2015173,549593,9,17684225
Chemistry,6985274,6014356,5480681,91,23,6160104,267831,67693,20,4696259
Computer Engineering,2084177,1497715,1660601,44,16,1747498,109219,39716,16,3407388
Computer Science,3862226,5616268,4723783,107,24,4734092,197254,44244,21,3042996
Earth Sciences + IGPP,6257946,3791753,6137926,53,20,5395875,269794,101809,20,5821813
Economics,628297,230982,488436,68,25,449238,17970,6606,NA,1925441
Education (CERIUS),3875536,804081,3023631,35,16,2567749,160484,73364,NA,1140707
Electrical Engineering,5676812,3839121,3122185,61,13,4212706,324054,69061,12,3311040
ENVS + CASFS,3555509,3232735,4659933,40,15,3816059,254404,95401,NA,NA
ENVS ,2743994,2419951,4388333,40,15,3184093,212273,79602,NA,1925441
Film & Digital Media,387400,0,134020,7,18,173807,9656,24830,16,29700
HAVC,0,0,2888,7,11,963,88,138,10,60000
HISCON,0,17109,5382,30,3,7497,2499,250,2.25,160615
History,556805,262921,12900,28,23.5,277542,11810,9912,26.75,419339
Linguistics,85045,298375,149938,21,12,177786,14816,8466,12.5,304956
Literature,0,109900,200000,60,29.5,103300,3502,1722,29,6000
Mathematics,1192673,722061,453249,35,15,789328,52622,22552,13,439294
METOX,1924673,2653625,3068446,18,7,2548915,364131,67693,7,3680409
Music,20000,8000,0,27,14.6,9333,639,346,14.56,0
Ocean+IMS+Earth,18248616,15741473,15566920,86,27,16519003,611815,192081,NA,21715890
Ocean Science,0,40000,349999,33,7,130000,18571,3939,7,159720
Philosophy,0,0,0,14,7,0,0,0,7,0
Physics + SCIPP,7855638,12037448,6090544,53,22.3,8661210,388395,163419,21.5,7877487
Politics,0,2500,229219,29,12,77240,6437,2663,NA,29216
Psychology,1212845,565509,405976,52,24,728110,30338,14002,NA,347453
Sociology,300000,24499,6332,38,16,110277,6892,2902,NA,70900
TM,871612,289639,473275,7,4,544842,136211,77835,3,253252
```

Reading files: the working directory

- The working directory is the default path where R searches for files and stores any that you crea.
- To figure out you current working directory

```
> getwd()
[1] "/Users/abel"
```

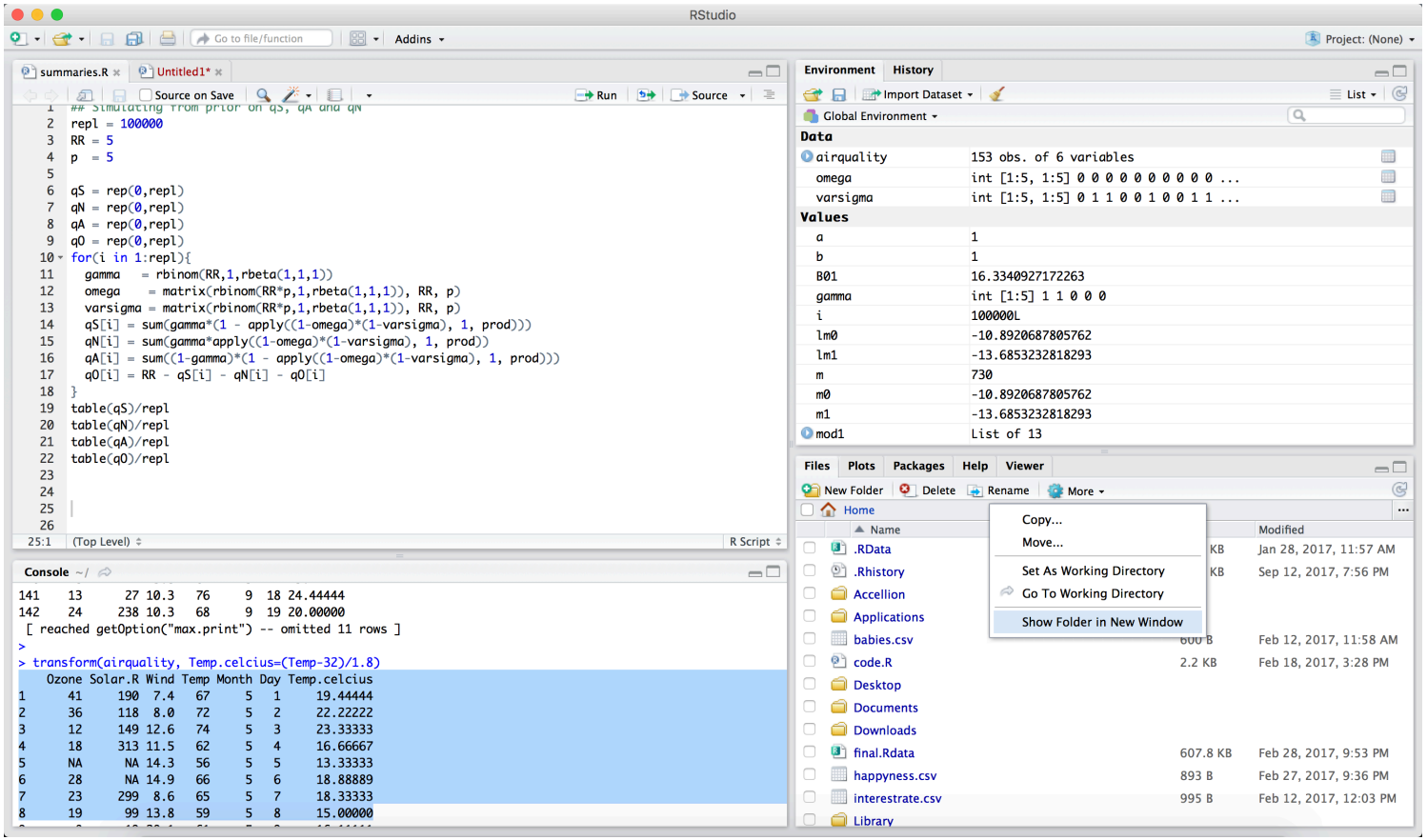
- You can change your working using the `setwd()` function:

```
> setwd("/Users/abel/Documents")
```

- If you are working on a PC make sure you replace the forward slashes in the directory paths for backlashes!

Changing directories

- In RStudio it easy to change directories from the Files menu.



Reading your file into R

- You can read a file using the `read.table` command. The output of `read.table` is a data frame!

```
> cacounties = read.table(file="california_counties.csv", sep=",",
header=F)
```

No column names

Comma delimited

- As mentioned before, R will look for the file in the working directory. If it cannot find it there, it will throw an error.
- You can read files from a different directory, but you will need to add the full path to the file name.

Reading your file into R

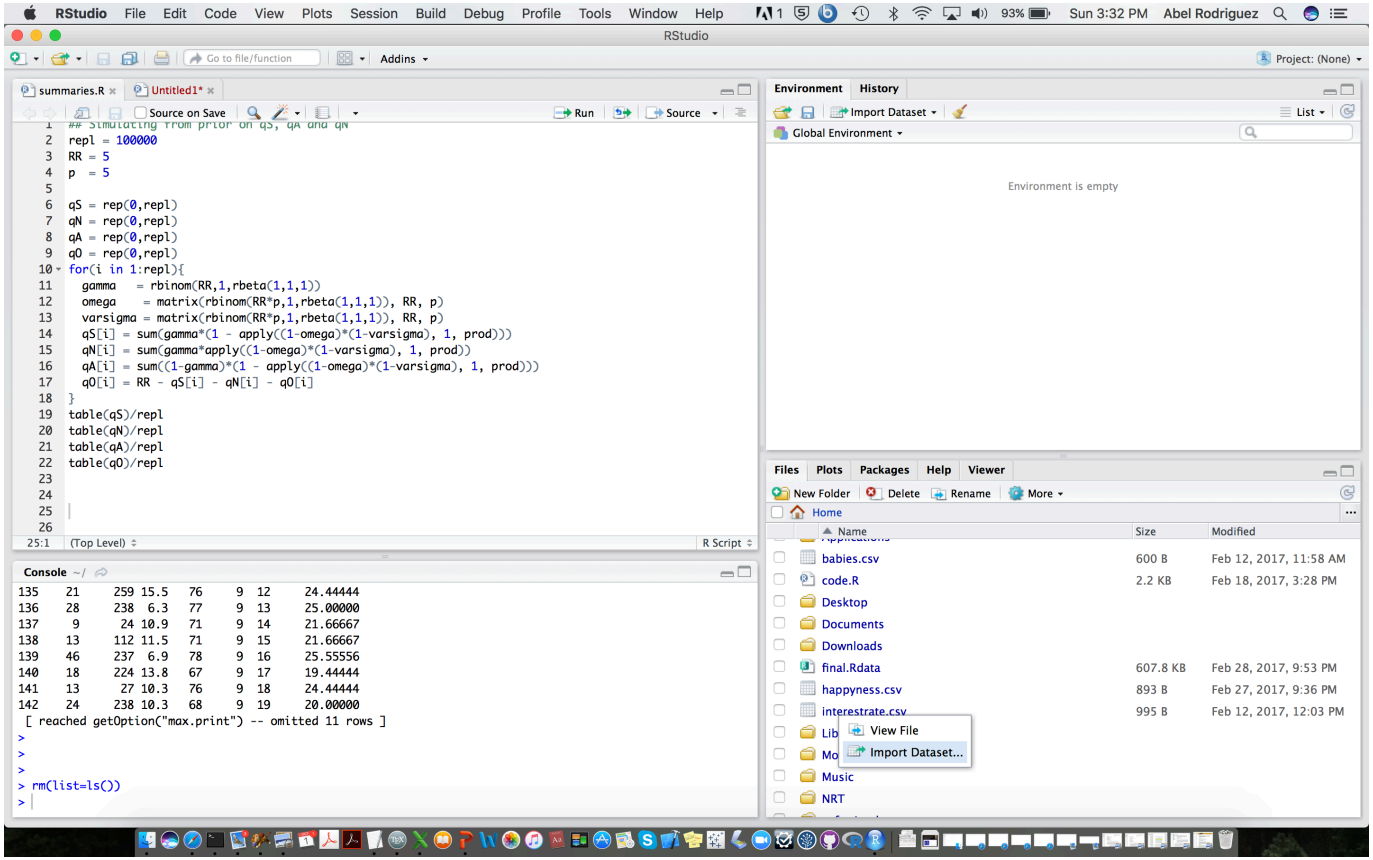
- Note that now a new object called "cacounties" appears on the window located on the top right corner (as expected, the object has 58 rows and 3 columns).
- If you now type

```
> cacounties
```

In the command window you can see the content of the object. Because the file had no header, the columns are named V1, V2 and V3.

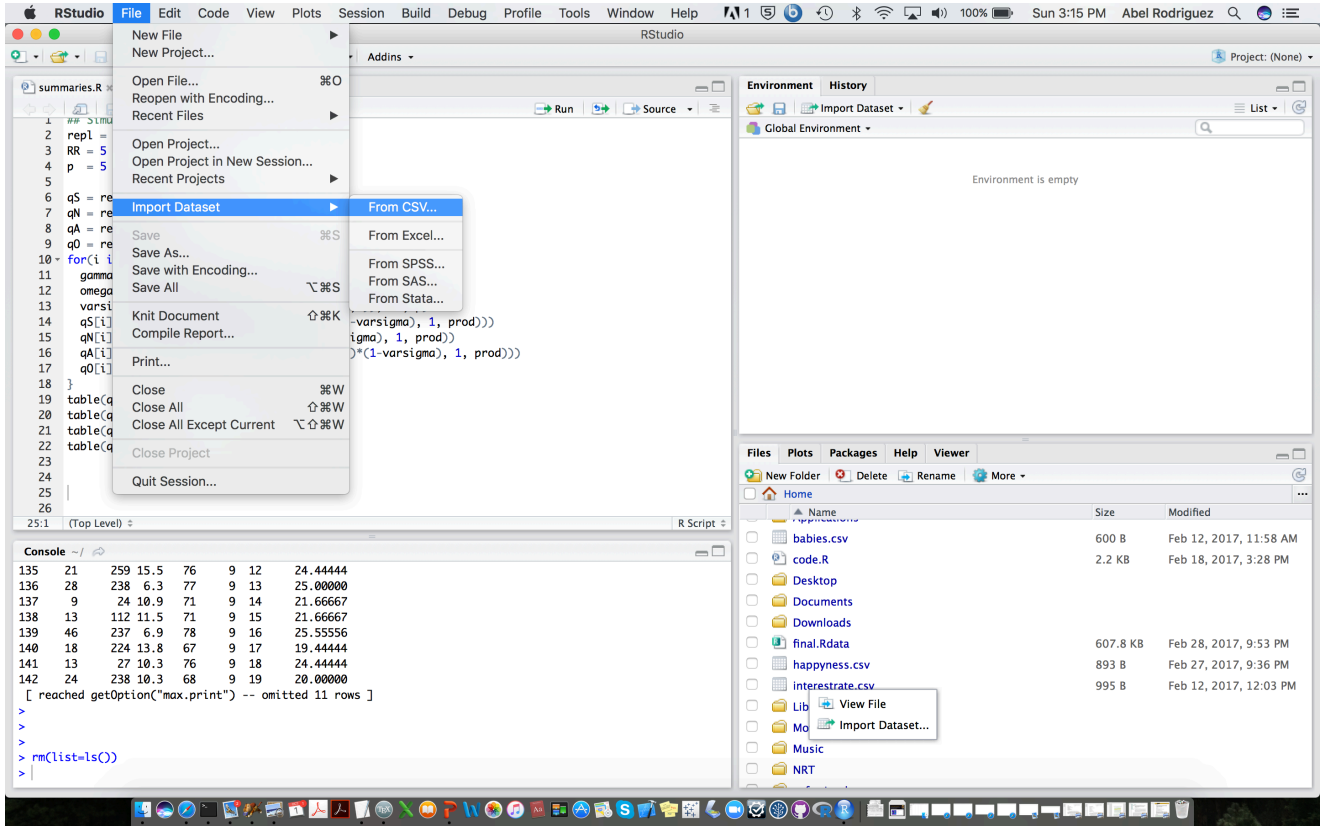
Reading your file into RStudio

- In Rstudio you can also edit or load a file using the Files tab of the bottom right window



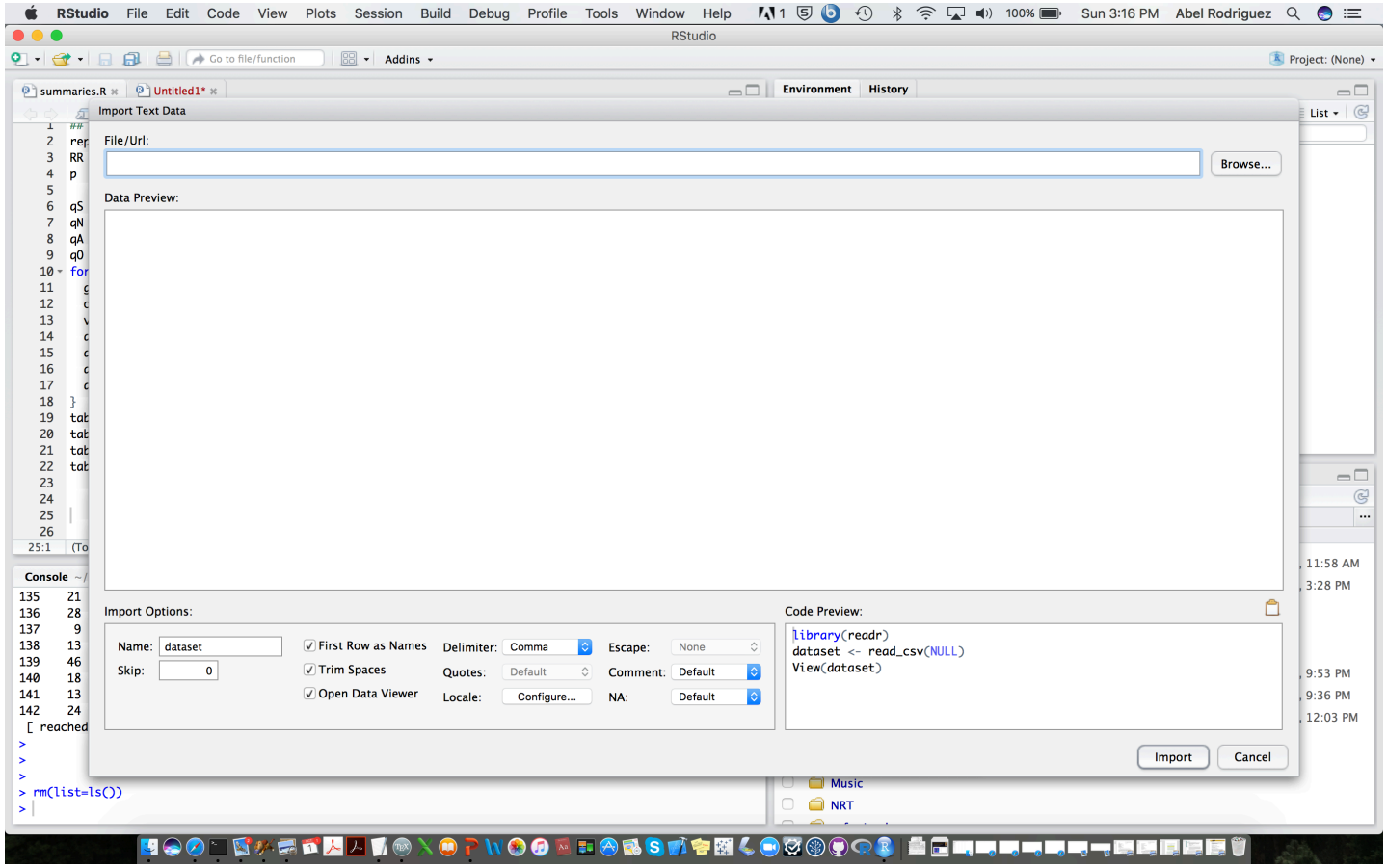
Reading your file into RStudio

- Yet another way is to use the Import Dataset option in the File menu



Reading your file into RStudio

- The menu option uses the readr library and makes it easy to load files from a url!



Reading your file into RStudio

- If you are working interactively, the menu options for changing working directories/reading files are more convenient. But if you are automating your code, you want to know what the right commands are!!!
- Note that the File -> Import Dataset menu does show you the underlying R code being used.
- The `View()` function opens the file in Rstudio for editing!!! Be careful what you do …

Reading your file into R

- Another option to read .tex files is the scan command.
- Scan is quite fast, but it useful only if all data in the file is of the same type (numeric, character, etc). No column names allowed, and values are read into a long vector.

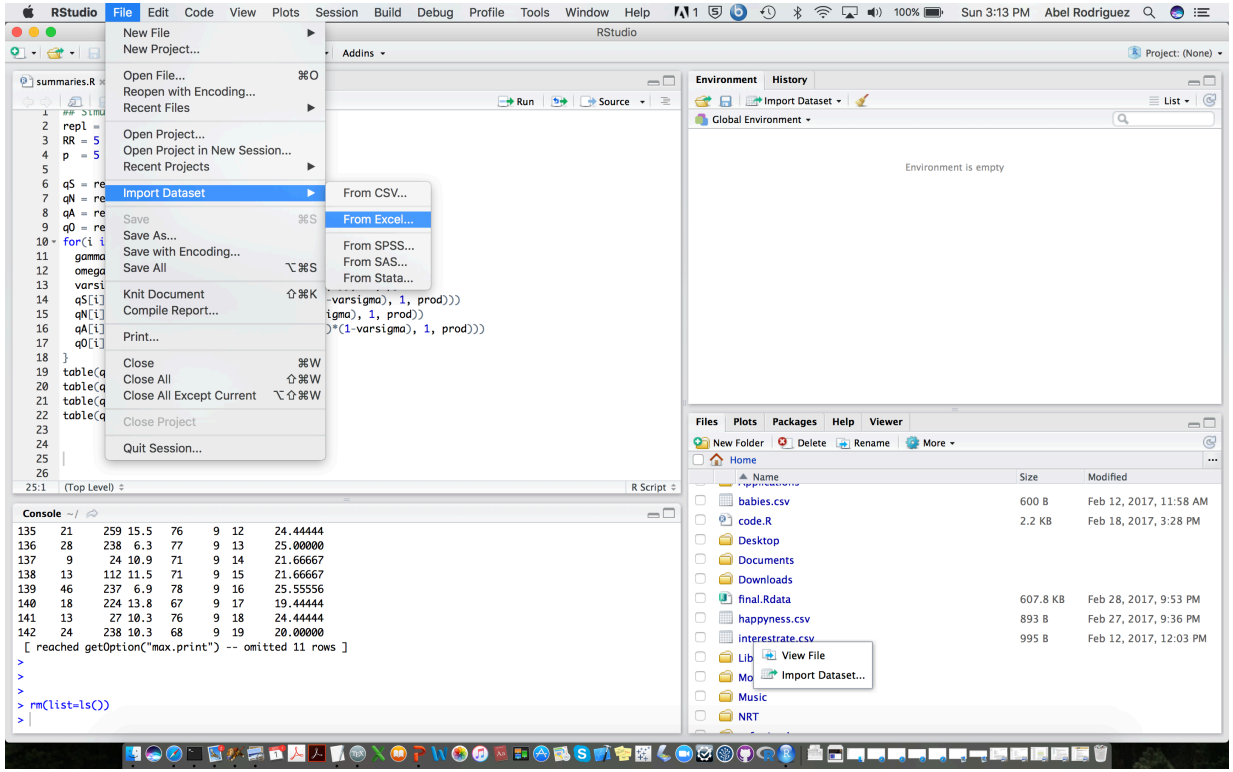
```
> x = scan("temp.txt")
```

- Another useful command is readLines, which allows you to read .tex files one line at a time (useful if you need to process text/search for regular expressions).

```
> x = readLines("cancer.csv")
```

Reading Excel files

- The package readxl allows you read Excel files in R. It is easier to use from the File -> Import Dataset menu.



Database integration

- The R implementation from the R Core Team is not very good at managing large datasets because it must hold everything in memory.
- When working with very large dataset it might be convenient to store the full data in a fast database and just pull the bits that you need from there in a dynamic way.
- Similarly, when the data is more reasonably stored in the form of a relational database, you might want to run your queries in the database server and pull just the information that you need, rather than trying to copy the full database into R first.

Database integration

- The package RODBC allows you connect to databases that use the ODBC (Open Database Connectivity) standard (nowadays, most database systems).
- The package includes functions for connecting to the database and for submitting SQL queries to it.
- The details of how to use this package are highly system-specific, and setting it up usually requires collaboration with your system/database manager.

Saving your results

- To save the whole session (workspace)

```
> save.image("myworkspace.Rdata")
```

- To save specific objects (as a binary object)

```
> x = seq(1,8)
> y = c("CA", "NE")
> save(x, y, file="myobjects.R")
```

- To save a matrix as a text file

```
> x = matrix(1:10, ncol=5, nrow=2)
> write(t(x),
file="myfile.txt",ncolumns=5)
```

Loading previously saved results

- Objects that were previously saved with `save` or sessions saved using `save.image` can be recovered using the `load` command

```
> load("myworkspace.Rdata")
> save(x, y, file="myobjects.R")
```

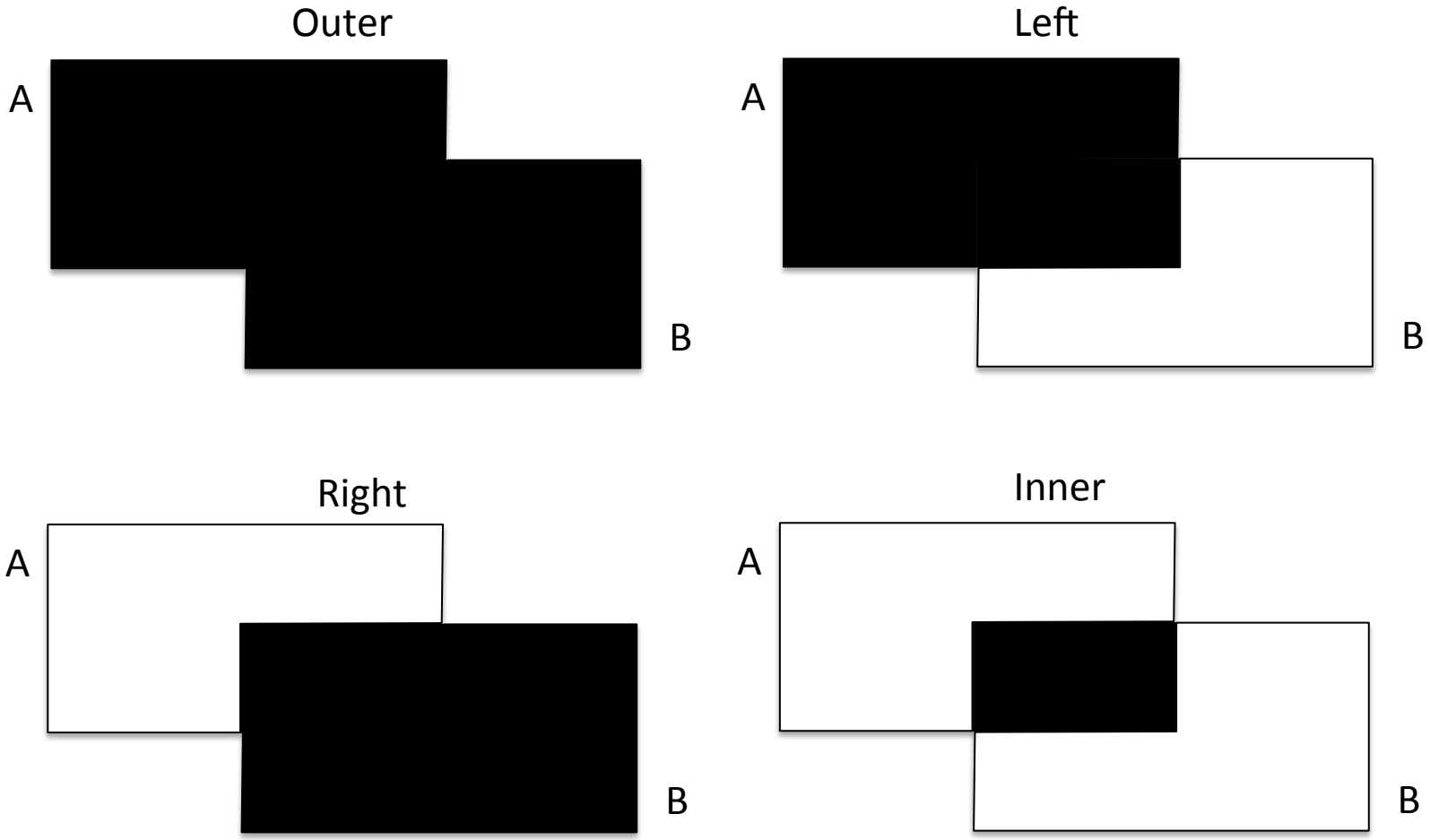
- Files save with the write command can be recovered using the scan function

```
> x = matrix(scan(file = "myfile.txt",
ncol=5, byrow=T)
```

Merging data frames

- Sometimes you might have two datasets that need to be combined together.
  - Author, book name, publication date, year …
  - Author, place of birth, nationality, etc …
- The merge function allows you to combine them without having to write a complex `for` loop.
  - Similar to a *joint* query in SQL.

Merging data frames



Merging data frames

- For example:

```
> x = read.table(”fecundity.csv”)
> head(x)
[1]
> y = read.table(”fish.csv”)
> head(y)
[1]
> merge(x, y, by=”ID”, all.x=TRUE)
```