Data structures in R

Abhijit Dasgupta

BIOF 339

A quick refresh

- R is a scripting language for data analysis and statistics
- R Markdown is a way of combining textual information and R code to produce reproducible documents
- RStudio is an integrated environment that makes it easier to work with R

You type commands (*code*) for R to run.

- objects like data (*nouns*)
- functions that do something to R objects (*verbs*)

Examples

```
airquality
diamonds

summary(airquality)
```

Objects in R

Let's start with the `airquality` data.

- It is an object
- of class `class(airquality)` = data.frame

How about each column?

Let's look at the Ozone and Wind columns

- We can access them using `airquality$Ozone` and `airquality$Wind`
  - `class(airquality$Ozone)` = integer
  - `class(airquality$Wind)` = numeric

	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
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10
11	7	NA	6.9	74	5	11
12	16	256	9.7	69	5	12
13	11	290	9.2	66	5	13
14	14	274	10.9	68	5	14
15	18	65	13.2	58	5	15
16	14	334	11.5	64	5	16
17	34	307	12.0	66	5	17
18	6	78	18.4	57	5	18
19	30	322	11.5	68	5	19
20	11	44	9.7	62	5	20
21	1	8	9.7	59	5	21
22	11	320	16.6	73	5	22
23	4	25	9.7	61	5	23
24	32	92	12.0	61	5	24
25	NA	66	16.6	57	5	25

Objects in R

```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
str(iris)
```

```
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Now we see another type of object, a `factor`

Objects in R

```
library(ggplot2)
str(midwest)
```

```
tibble [437 × 28] (S3: tbl_df/tbl/data.frame)
 $ PID                 : int [1:437] 561 562 563 564 565 566 567 568 569 570 ...
 $ county              : chr [1:437] "ADAMS" "ALEXANDER" "BOND" "BOONE" ...
 $ state               : chr [1:437] "IL" "IL" "IL" "IL" ...
 $ area                : num [1:437] 0.052 0.014 0.022 0.017 0.018 0.05 0.017 0.027 0.024 0.058 ...
 $ poptotal            : int [1:437] 66090 10626 14991 30806 5836 35688 5322 16805 13437 173025 ...
 $ popdensity          : num [1:437] 1271 759 681 1812 324 ...
 $ popwhite            : int [1:437] 63917 7054 14477 29344 5264 35157 5298 16519 13384 146506 ...
 $ popblack            : int [1:437] 1702 3496 429 127 547 50 1 111 16 16559 ...
 $ popamerindian       : int [1:437] 98 19 35 46 14 65 8 30 8 331 ...
 $ popasian            : int [1:437] 249 48 16 150 5 195 15 61 23 8033 ...
 $ popother            : int [1:437] 124 9 34 1139 6 221 0 84 6 1596 ...
 $ percwhite           : num [1:437] 96.7 66.4 96.6 95.3 90.2 ...
 $ percblack           : num [1:437] 2.575 32.9 2.862 0.412 9.373 ...
 $ percamerindan       : num [1:437] 0.148 0.179 0.233 0.149 0.24 ...
 $ percasian           : num [1:437] 0.3768 0.4517 0.1067 0.4869 0.0857 ...
 $ percother           : num [1:437] 0.1876 0.0847 0.2268 3.6973 0.1028 ...
 $ popadults           : int [1:437] 43298 6724 9669 19272 3979 23444 3583 11323 8825 95971 ...
 $ perchsd             : num [1:437] 75.1 59.7 69.3 75.5 68.9 ...
 $ percollege          : num [1:437] 19.6 11.2 17 17.3 14.5 ...
 $ percprof            : num [1:437] 4.36 2.87 4.49 4.2 3.37 ...
 $ poppovertyknown     : int [1:437] 63628 10529 14235 30337 4815 35107 5241 16455 13081 154934 ...
 $ percpovertyknown    : num [1:437] 96.3 99.1 95 98.5 82.5 ...
 $ percbelowpoverty    : num [1:437] 13.15 32.24 12.07 7.21 13.52 ...
 $ percchildbelowpovert: num [1:437] 18 45.8 14 11.2 13 ...
```

Objects in R

The most common types of data we see are `numeric`, `character`, `factor`. You can also see `Date` and `logical`

You can test to see if data is of a particular type, or convert from one data type to another

Data type	Test	Convert
numeric	is.numeric	as.numeric
character	is.character	as.character
factor	is.factor	as.factor

This conversion is important. Why?

05:00

Factors

Factors are uniquely an R thing.

They are meant to represent categorical data (gender, race, state, phenotype)

They look like character vectors, but internally act like integers, so you have to be a bit careful with them

Whenever you're in doubt, convert them to characters using `as.character`.

We'll see the utility of factors when we do data munging, summaries and modeling

Every object in R has a name

You give an object a name using the syntax `name <- object`

Naming conventions:

1. Snake_case or pothole_case
2. CamelCase
3. Some.people.use.periods

I'm partial to `snake_case`.

The point here is to make expressive names using English so you know what is stored in the name.

A silly exercise

From the iris dataset, save each column into a new object, giving it a name. Then see what kind of data that object contains.

05:00

Bigger objects

Scalar -> vector (array) -> matrix (2-d array)

- A scalar is a single number or word
- A vector is a bunch of scalars arranged in a row or column
- A matrix is a bunch of scalars arranged in rows and columns

Each of these must be of the same data type

Examples

```
2
```

```
[1] 2
```

```
c(2,3,4,5,6)
```

```
[1] 2 3 4 5 6
```

> `c()` is the concatenate function

```
matrix(c(1,2,3,4), byrow = T, nrow = 2)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Data comes in many flavors

Heterogeneous data

From Excel, we are familiar with keeping different kinds of data together in a spreadsheet

- Expression levels (numeric)
- Gene names (character)
- Date of experiment (Date)

In R, the objects that can hold heterogeneous data are `data.frame` and `list`

Data sets

Typical data structure

- Data is typically in a rectangular format
    - spreadsheet, database table
    - CSV (comma-separated values) or TSV (tab-separated values) files
- Characteristic
    - Rows are observations
    - Columns are variables
    - Each column has the same number of observations

> **Tidy data** is a particularly amenable format for data analysis.

The `data.frame`

Dataframes are the primary mode of storing datasets in R

They were revolutionary in that they kept heterogeneous data together

They share properties of both a **matrix** and a **list**

```
class(airquality)
```

```
[1] "data.frame"
```

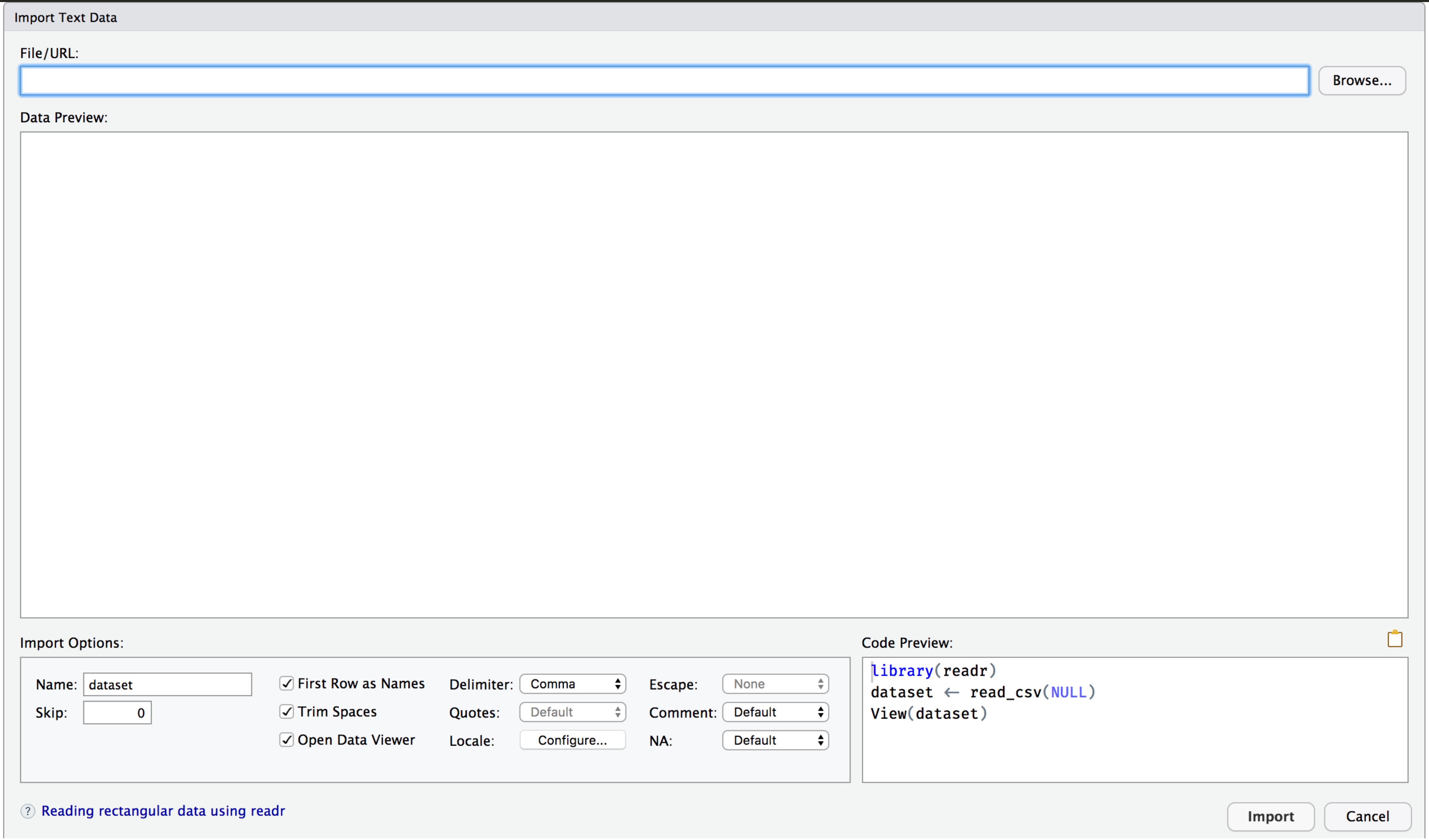
> Technically, a data.frame is a list of vectors (or objects, generally) of the same length

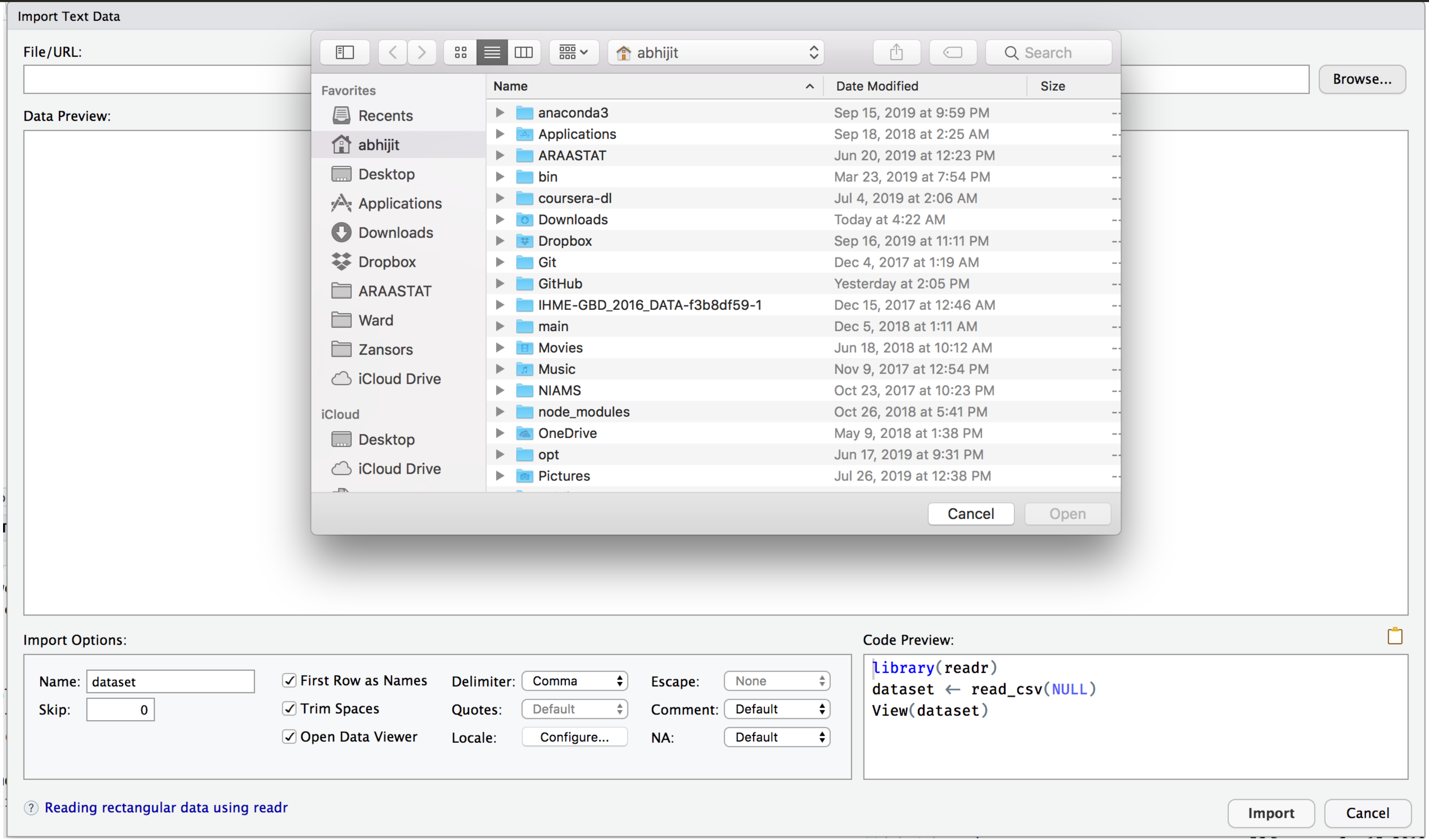
Load some data

We'll load the `spine` dataset into R.

To do this, download the data from the web, and store it in the main folder in your project.

Then, in the Environment pane, import it using the **Import Dataset** button. You will use the `From text (readr)` option


Import Text Data
File/URL:
Browse...
Data Preview:
Import Options:
Name: dataset
Skip: 0
First Row as Names
Trim Spaces
Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default
Code Preview:
library(readr)
dataset <- read_csv(NULL)
View(dataset)
Reading rectangular data using readr
Import
Cancel


Import Text Data
File/URL:
Browse...
Data Preview:
Favorites
Recents
abhijit
Desktop
Applications
Downloads
Dropbox
ARAASTAT
Ward
Zansors
iCloud Drive
iCloud
Desktop
iCloud Drive
abhijit
Search
Name
Date Modified
Size
anaconda3 Sep 15, 2019 at 9:59 PM
Applications Sep 18, 2018 at 2:25 AM
ARAASTAT Jun 20, 2019 at 12:23 PM
bin Mar 23, 2019 at 7:54 PM
coursera-dl Jul 4, 2019 at 2:06 AM
Downloads Today at 4:22 AM
Dropbox Sep 16, 2019 at 11:11 PM
Git Dec 4, 2017 at 1:19 AM
GitHub Yesterday at 2:05 PM
IHME-GBD_2016_DATA-f3b8df59-1 Dec 15, 2017 at 12:46 AM
main Dec 5, 2018 at 1:11 AM
Movies Jun 18, 2018 at 10:12 AM
Music Nov 9, 2017 at 12:54 PM
NIAMS Oct 23, 2017 at 10:23 PM
node_modules Oct 26, 2018 at 5:41 PM
OneDrive May 9, 2018 at 1:38 PM
opt Jun 17, 2019 at 9:31 PM
Pictures Jul 26, 2019 at 12:38 PM
Cancel
Open
Import Options:
Name: dataset
Skip: 0
First Row as Names
Trim Spaces
Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default
Code Preview:
library(readr)
dataset <- read_csv(NULL)
View(dataset)
Reading rectangular data using readr
Import
Cancel

Import Text Data

File/URL:

~/ARAASTAT/Teaching/BIOF339/slides/lectures/data/Dataset_spine.csv

Browse...

Data Preview:

Pelvic incidence (double)	Pelvic tilt (double)	Lumbar lordosis angle (double)	Sacral slope (double)	Pelvic radius (double)	Degree spondylolisthesis (double)	Pelvic slope (double)	Direct tilt (double)	Thoracic slope (double)	Cervical tilt (double)	Sacrum angle (double)	Scoliosis slope (dou
63.02782	22.552586	39.60912	40.47523	98.67292	-0.2544000	0.74450346	12.5661	14.5386	15.30468	-28.658501	43.5123
39.05695	10.060991	25.01538	28.99596	114.40543	4.5642586	0.41518568	12.8874	17.5323	16.78486	-25.530607	16.1102
68.83202	22.218482	50.09219	46.61354	105.98514	-3.5303173	0.47488916	26.8343	17.4861	16.65897	-29.031888	19.2221
69.29701	24.652878	44.31124	44.64413	101.86850	11.2115234	0.36934526	23.5603	12.7074	11.42447	-30.470246	18.8329
49.71286	9.652075	28.31741	40.06078	108.16872	7.9185006	0.54336047	35.4940	15.9546	8.87237	-16.378376	24.9171
40.25020	13.921907	25.12495	26.32829	130.32787	2.2306517	0.78999286	29.3230	12.0036	10.40462	-1.512209	9.6548
53.43293	15.864336	37.16593	37.56859	120.56752	5.9885507	0.19891957	13.8514	10.7146	11.37832	-20.510434	25.9477
45.36675	10.755611	29.03835	34.61114	117.27007	-10.6758708	0.13197255	28.8165	7.7676	7.60961	-25.111459	26.3543
43.79019	13.533753	42.69081	30.25644	125.00289	13.2890182	0.19040763	22.7085	11.4234	10.59188	-20.020075	40.0276
36.68635	5.010884	41.94875	31.67547	84.24142	0.6644371	0.36770014	26.2011	8.7380	14.91416	-1.702097	21.4320
49.70661	13.040974	31.33450	36.66564	108.64827	-7.8259858	0.68800950	31.3502	16.5097	15.17645	-0.502127	18.3437
31.23239	17.715819	15.50000	13.51657	120.05540	0.4997514	0.60834276	21.4356	9.2589	14.76412	-21.724559	36.4449
48.91555	19.964556	40.26379	28.95100	119.32136	8.0288946	0.13947816	32.7916	7.2049	8.61882	-1.215542	27.3713
53.57217	20.460828	33.10000	33.11134	110.96670	7.0448029	0.08193099	15.0580	12.8127	12.00109	-1.734117	15.6205

Previewing first 50 entries.

Import Options:

Name: Dataset_spine

Skip: 0

☑ First Row as Names

☑ Trim Spaces

☑ Open Data Viewer

Delimiter: Comma

Quotes: Default

Locale: Configure...

Escape: None

Comment: Default

NA: Default

Code Preview:

```
library(readr)
Dataset_spine <- read_csv("ARAASTAT/Teaching/BIOF339
/slides/lectures/data/Dataset_spine.csv")
View(Dataset_spine)
```

? Reading rectangular data using readr

Import Cancel

A digression: Lists and Matrices

Matrices

A **matrix** is a rectangular array of data *of the same type*

```
matrix(0, nrow=2, ncol=4)
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
```

```
matrix(letters, nrow=2)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,] "a"  "c"  "e"  "g"  "i"  "k"  "m"  "o"  "q"  "s"   "u"   "w"   "y"  
[2,] "b"  "d"  "f"  "h"  "j"  "l"  "n"  "p"  "r"  "t"   "v"   "x"   "z"  
```

```
matrix(letters, nrow=2, byrow=T)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"  
[2,] "n"  "o"  "p"  "q"  "r"  "s"  "t"  "u"  "v"  "w"   "x"   "y"   "z"  
```

Matrices

You can create a matrix from a set of *vectors* of the same length

```
x <- c(1,2,3,4)
y <- c(10,20,30,40)
```

Put columns together

```
cbind(c(1,2,3,4), c(10,20,30,40)) ## Column bind
```

```
     [,1] [,2]
[1,]    1   10
[2,]    2   20
[3,]    3   30
[4,]    4   40
```

Matrices

You can create a matrix from a set of *vectors* of the same length

```
x <- c(1,2,3,4)
y <- c(10,20,30,40)
```

Put rows together

```
example_matrix <- rbind(c(1,2,3,4), c(10,20,30,40)) ## Row bind
example_matrix
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
```

Extracting elements

```
example_matrix
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
```

```
example_matrix[1,] ## Extracts 1st row
```

```
[1] 1 2 3 4
```

```
example_matrix[,2:3] ## extracts 2nd & 3rd columns
```

```
     [,1] [,2]
[1,]    2    3
[2,]   20   30
```

```
example_matrix[1,4]
```

```
[1] 4
```

Matrix properties

```
example_matrix
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
```

```
nrow(example_matrix) ## Number of rows
```

```
[1] 2
```

```
ncol(example_matrix) ## Number of columns
```

```
[1] 4
```

```
dim(example_matrix) ## shortcut for above
```

```
[1] 2 4
```

Matrix arithmetic

```
example_matrix
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
```

```
example_matrix + 5 ## Add 5 to each element
```

```
     [,1] [,2] [,3] [,4]
[1,]    6    7    8    9
[2,]   15   25   35   45
```

```
example_matrix * 2 ## Multiply each element by 2
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    4    6    8
[2,]   20   40   60   80
```

Two matrices

```
example_matrix
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
```

```
example_matrix2 <- rbind(3:6, 9:12)
example_matrix2
```

```
     [,1] [,2] [,3] [,4]
[1,]    3    4    5    6
[2,]    9   10   11   12
```

```
example_matrix + example_matrix2
```

```
     [,1] [,2] [,3] [,4]
[1,]    4    6    8   10
[2,]   19   30   41   52
```

Two matrices

```
example_matrix
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
```

```
example_matrix2
```

```
     [,1] [,2] [,3] [,4]
[1,]    3    4    5    6
[2,]    9   10   11   12
```

```
example_matrix * example_matrix2 ## Not matrix multiplication, but element-wise multiplication
```

```
     [,1] [,2] [,3] [,4]
[1,]    3    8   15   24
[2,]   90  200  330  480
```

Two matrices

```
rbind(example_matrix, example_matrix2)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   10   20   30   40
[3,]    3    4    5    6
[4,]    9   10   11   12
```

```
cbind(example_matrix, example_matrix2)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    2    3    4    3    4    5    6
[2,]   10   20   30   40    9   10   11   12
```

Two matrices

```
dim(example_matrix2)
```

```
[1] 2 4
```

```
t(example_matrix2) ## Transpose of a matrix
```

```
     [,1] [,2]
[1,]    3    9
[2,]    4   10
[3,]    5   11
[4,]    6   12
```

```
example_matrix %*% t(example_matrix2) ## Matrix multiplication
```

```
     [,1] [,2]
[1,]   50  110
[2,]  500 1100
```

Lists

Lists are collections of arbitrary objects in R

```
example_list <- list(c('Andy','Brian','Harry'),
                     c(12, 16, 16),
                     c(TRUE, TRUE, FALSE),
                     matrix(1, nrow=2, ncol=3))
example_list
```

```
[[1]]
[1] "Andy"  "Brian" "Harry"

[[2]]
[1] 12 16 16

[[3]]
[1]  TRUE  TRUE FALSE

[[4]]
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```

Extracting elements from lists

```
example_list[[3]]
```

```
[1]  TRUE  TRUE FALSE
```

```
example_list[1:2]
```

```
[[1]]
[1] "Andy"  "Brian" "Harry"

[[2]]
[1] 12 16 16
```

Extracting elements from lists

```
example_list[[4]]
```

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```

```
class(example_list[[4]])
```

```
[1] "matrix" "array"
```

```
example_list[[4]][1,]
```

```
[1] 1 1 1
```

Named lists

```
example_named_list <- list('Names' = c('Andy','Brian','Harry'),
                     "YearsOfEducation" = c(12, 16, 16),
                     "Married" = c(TRUE, TRUE, FALSE),
                     'something' = matrix(1, nrow=2, ncol=3))

example_named_list[['Names']]
```

```
[1] "Andy"  "Brian" "Harry"
```

```
example_named_list$Names
```

```
[1] "Andy"  "Brian" "Harry"
```

```
example_named_list$Names[3]
```

```
[1] "Harry"
```

Back to a Data Frame

Data frames

A data.frame object is a **named list** where each element is of the same length

You can use both *matrix* and *list* functions to operate on data.frame objects!!

Data Frames

```
head(data_spine)
```

```
  Pelvic.incidence Pelvic.tilt Lumbar.lordosis.angle Sacral.slope Pelvic.radius
1         63.02782   22.552586              39.60912     40.47523      98.67292
2         39.05695   10.060991              25.01538     28.99596     114.40543
3         68.83202   22.218482              50.09219     46.61354     105.98514
4         69.29701   24.652878              44.31124     44.64413     101.86850
5         49.71286    9.652075              28.31741     40.06078     108.16872
6         40.25020   13.921907              25.12495     26.32829     130.32787
  Degree.spondylolisthesis Pelvic.slope Direct.tilt Thoracic.slope
1                -0.254400    0.7445035     12.5661        14.5386
2                 4.564259    0.4151857     12.8874        17.5323
3                -3.530317    0.4748892     26.8343        17.4861
4                11.211523    0.3693453     23.5603        12.7074
5                 7.918501    0.5433605     35.4940        15.9546
6                 2.230652    0.7899929     29.3230        12.0036
  Cervical.tilt Sacrum.angle Scoliosis.slope Class.attribute
1      15.30468   -28.658501         43.5123        Abnormal
2      16.78486   -25.530607         16.1102        Abnormal
3      16.65897   -29.031888         19.2221        Abnormal
4      11.42447   -30.470246         18.8329        Abnormal
5       8.87237   -16.378376         24.9171        Abnormal
6      10.40462    -1.512209          9.6548        Abnormal
```

Data Frames

```
dim(data_spine)
```

```
[1] 310  13
```

```
nrow(data_spine)
```

```
[1] 310
```

```
data_spine_small <- data_spine[1:4,] ## Matrix operation
```

Data Frames

```
data_spine_small[,2] ## Matrix extraction by position
```

```
[1] 22.55259 10.06099 22.21848 24.65288
```

```
data_spine_small[[2]] ## List extraction by position
```

```
[1] 22.55259 10.06099 22.21848 24.65288
```

Data Frames

```
data_spine_small[['Pelvic.tilt']] ## Named list extraction
```

```
[1] 22.55259 10.06099 22.21848 24.65288
```

```
data_spine_small[,'Pelvic.tilt'] ## Data frame named column extraction
```

```
[1] 22.55259 10.06099 22.21848 24.65288
```

```
data_spine_small$Pelvic.tilt ## Dollar sign extraction
```

```
[1] 22.55259 10.06099 22.21848 24.65288
```

Data Frames

My preference is for

1. *data frame named column extraction* `data_spine_small[,'Pelvic.tilt']`,
2. *named list extraction* `data_spine_small[['Pelvic.tilt']]`
3. *Dollar-based extraction* `data_spine_small$Pelvic.tilt`

Data Frames

```
names(data_spine_small)
```

```
 [1] "Pelvic.incidence"         "Pelvic.tilt"
 [3] "Lumbar.lordosis.angle"    "Sacral.slope"
 [5] "Pelvic.radius"            "Degree.spondylolisthesis"
 [7] "Pelvic.slope"             "Direct.tilt"
 [9] "Thoracic.slope"           "Cervical.tilt"
[11] "Sacrum.angle"             "Scoliosis.slope"
[13] "Class.attribute"
```

```
data_spine_small[,c('Pelvic.tilt', 'Pelvic.slope','Class.attribute')]
```

```
  Pelvic.tilt Pelvic.slope Class.attribute
1    22.55259    0.7445035        Abnormal
2    10.06099    0.4151857        Abnormal
3    22.21848    0.4748892        Abnormal
4    24.65288    0.3693453        Abnormal
```