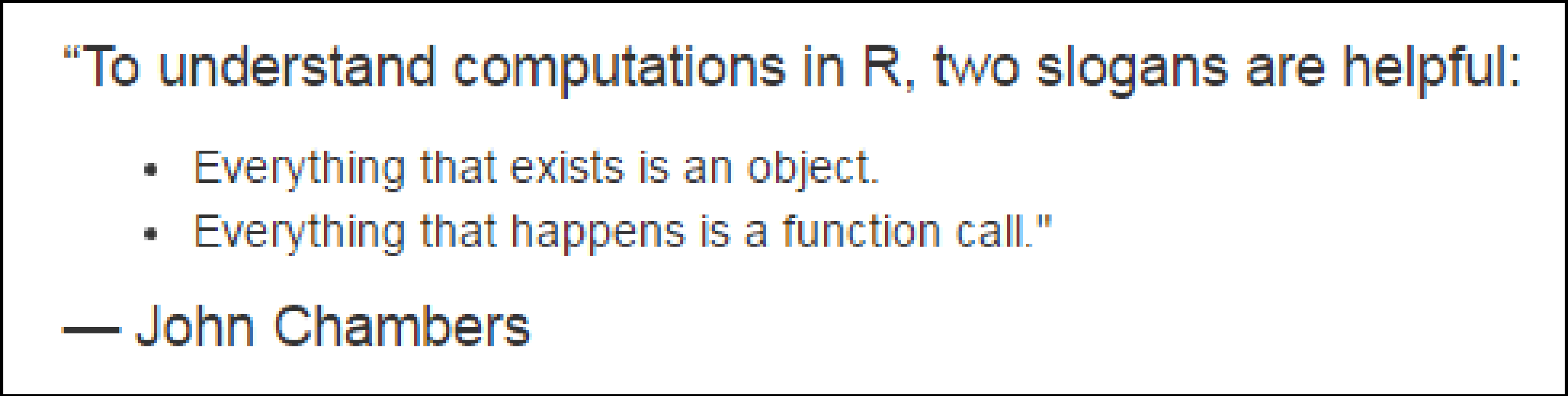
R Data Structures

WMG Training
Sept - Oct 2016

National Environmental Standards and Regulations Enforcement Agency (NESREA)

Facilitator: Victor Ordu

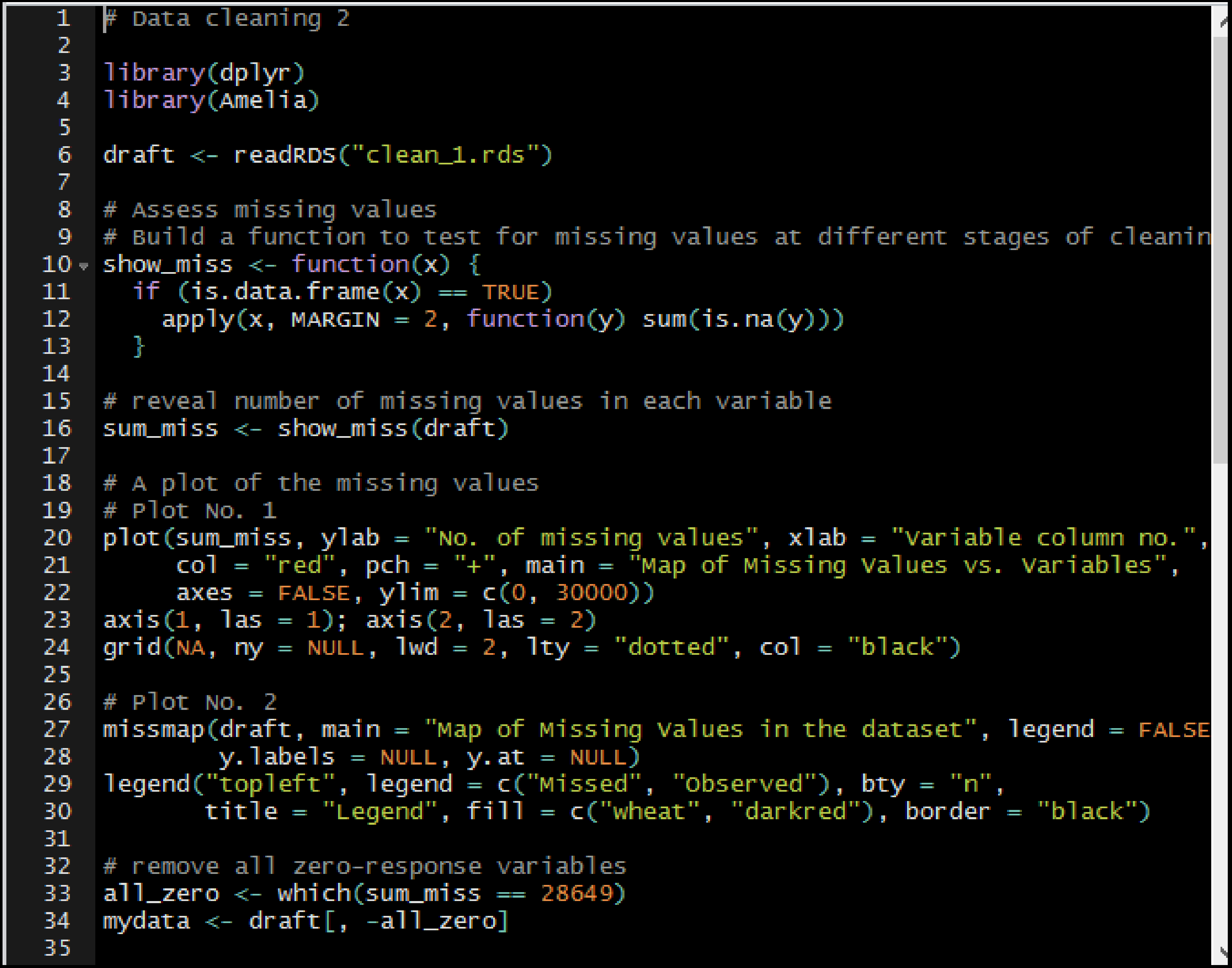

"To understand computations in R, two slogans are helpful:

- Everything that exists is an object.
- Everything that happens is a function call."

— John Chambers

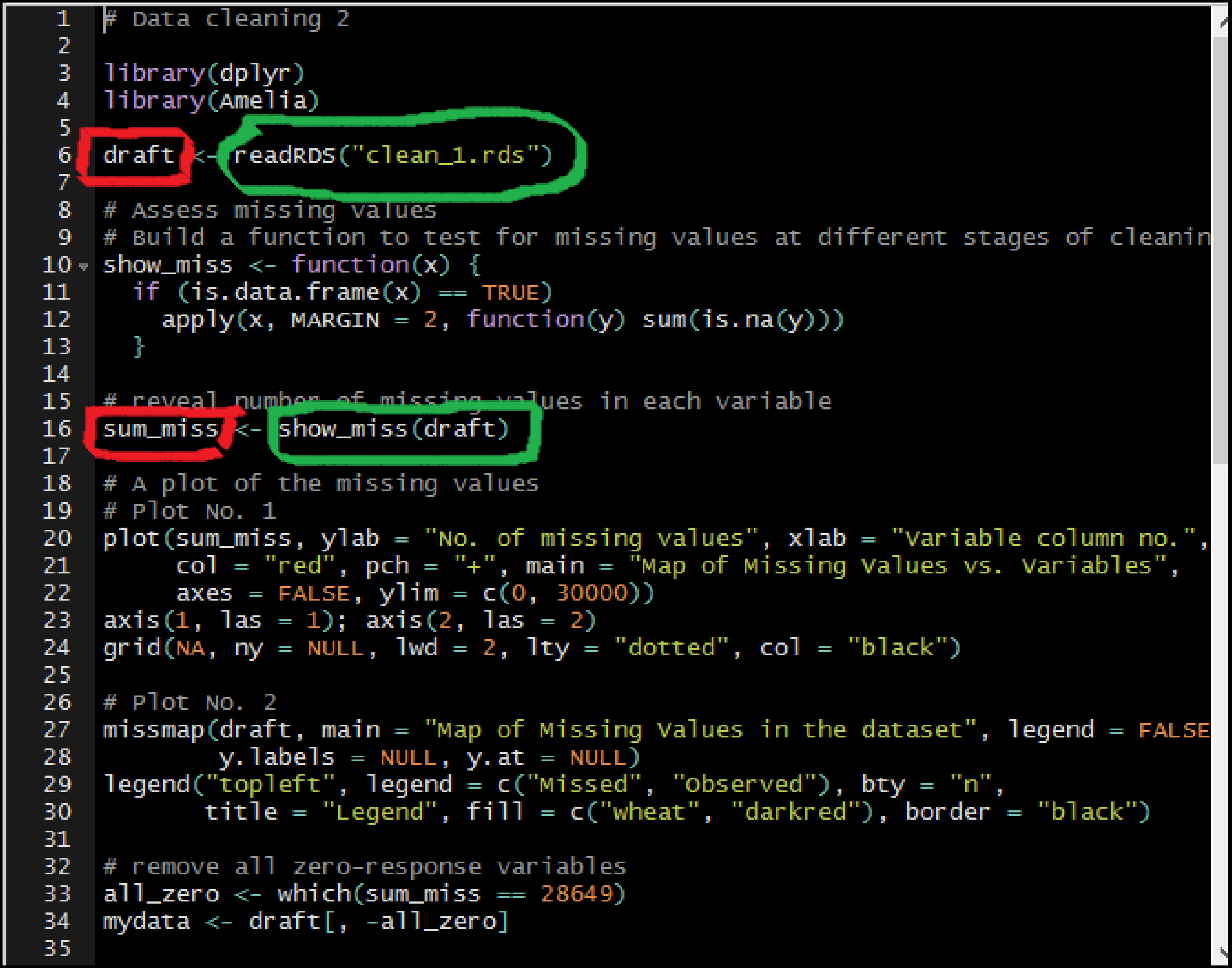
Source: H Wickham (2014). *Advanced R.* Chapman & Hall, Boca Raton. http://adv-r.had.co.nz/

Recap



```r
# Data cleaning 2

library(dplyr)
library(Amelia)

draft <- readRDS("clean_1.rds")

# Assess missing values
# Build a function to test for missing values at different stages of cleanin
show_miss <- function(x) {
  if (is.data.frame(x) == TRUE)
    apply(x, MARGIN = 2, function(y) sum(is.na(y)))
  }

# reveal number of missing values in each variable
sum_miss <- show_miss(draft)

# A plot of the missing values
# Plot No. 1
plot(sum_miss, ylab = "No. of missing values", xlab = "Variable column no.",
     col = "red", pch = "+", main = "Map of Missing Values vs. Variables",
     axes = FALSE, ylim = c(0, 30000))
axis(1, las = 1); axis(2, las = 2)
grid(NA, ny = NULL, lwd = 2, lty = "dotted", col = "black")

# Plot No. 2
missmap(draft, main = "Map of Missing Values in the dataset", legend = FALSE
        y.labels = NULL, y.at = NULL)
legend("topleft", legend = c("Missed", "Observed"), bty = "n",
       title = "Legend", fill = c("wheat", "darkred"), border = "black")

# remove all zero-response variables
all_zero <- which(sum_miss == 28649)
mydata <- draft[, -all_zero]

```

Recap



```r
# Data cleaning 2

library(dplyr)
library(Amelia)

draft <- readRDS("clean_1.rds")

# Assess missing values
# Build a function to test for missing values at different stages of cleanin
show_miss <- function(x) {
  if (is.data.frame(x) == TRUE)
    apply(x, MARGIN = 2, function(y) sum(is.na(y)))
  }

# reveal number of missing values in each variable
sum_miss <- show_miss(draft)

# A plot of the missing values
# Plot No. 1
plot(sum_miss, ylab = "No. of missing values", xlab = "Variable column no.",
     col = "red", pch = "+", main = "Map of Missing Values vs. Variables",
     axes = FALSE, ylim = c(0, 30000))
axis(1, las = 1); axis(2, las = 2)
grid(NA, ny = NULL, lwd = 2, lty = "dotted", col = "black")

# Plot No. 2
missmap(draft, main = "Map of Missing Values in the dataset", legend = FALSE
        y.labels = NULL, y.at = NULL)
legend("topleft", legend = c("Missed", "Observed"), bty = "n",
       title = "Legend", fill = c("wheat", "darkred"), border = "black")

# remove all zero-response variables
all_zero <- which(sum_miss == 28649)
mydata <- draft[, -all_zero]

```

- *Caveat*
  - Not everything I will tell you will be 100% correct

  - Why?
    - Stupid mistakes
    - Updates
    - New knowledge
    - Multiple approaches, some better than others
    - Continuous learning

  - *Moral*: *commit to personal growth*

Notation

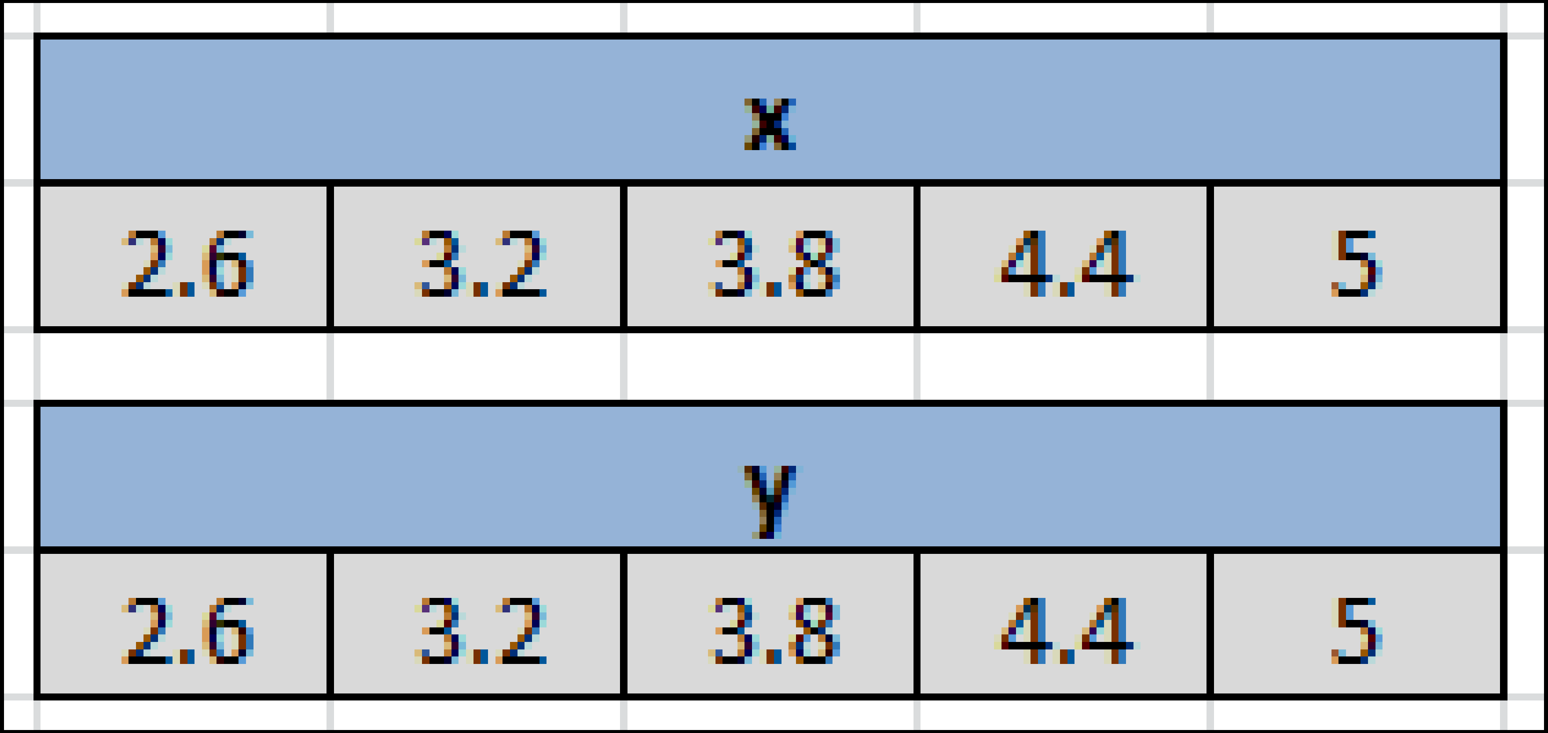
- Regular text with look like this…
- Highlighted items will look like this…
- `this <- R_code(“will look like”)`

Types of Data Structures

1. Vectors
2. Data frames
3. Matrices
4. Lists
5. Arrays

Types of Data Structures

1. Vectors
2. Data frames
3. Matrices
4. Lists
5. Arrays



x				
2.6	3.2	3.8	4.4	5

y				
2.6	3.2	3.8	4.4	5

Kinds of Vectors

- Six (6) kinds:
  1. Character
  2. Integer
  3. Double (or numeric)
  4. Logical

Vectors

- Common characteristics
  - All elements are of a particular <u>data **type**</u> (in lay language, "type" would be numbers, words, etc.)
  - One-dimensional
  - The lowest vector is of length 1
  - The largest … well, depends on the `.Machine`

Making vectors

- By assignment
  - The concatenate function
    - Latin: *con – **caten** – atus* (chain)
    - Some call it “combine” function
    - Indispensible in creating vectors
- You can ‘grow’ a vector
  - You may ask how, much, much, much, much later

- Character vectors
  - Strings are always placed in quotation marks when coding i.e. `“boy”`, `“NESREA”`, `“R is easy to learn”`, `“A string can be a whole sentence!”`, `“9”`.
  - Some character vectors are inbuilt into R e.g. `letters`, `LETTERS`, `month.abb`, `month.name`
  - Remember use quotation marks: `“ ”` or `‘ ’`.
  - We can create empty vectors with specific lengths e.g. `character(length = 10)` or `character(10)`
  - Limit approx. 2^{31} (about 2 billion) characters!

– Exercise
  - Start a clean slate with `rm(list = ls())`
  - Make a character vector `Name` containing full names (both Surname and Given Name) of 10 adults
  - Make a second vector `Facility` of names of 10 facilities (imaginary, please!)
  - Use `typeof()` to check what type of vector `Name` is
  - Confirm the type of `Facility` using `is.character()`
  - Note: We can use `as.character()` to convert another vector to a character vector.

- Integer vectors
  - 1L, 2L, 3L
    - Why the ‘L’?
  - Not numerical per se
  - Wide range – max up to 2,147,483,647

  - Exercise
    - Make an integer vector Age of 10 adult subjects
    - Make an integer vector `StaffStrength` for 10 facilities

- Numeric (double) vectors
  - These are real numbers
  - Story of the term double
  - Some numeric vectors are inbuilt – mathematical constants e.g. `pi`, `exp(1)`,

- Logical vectors
  - TRUE/FALSE (not true/false); T/F
  - Zero is FALSE; **any** non-zero is TRUE

  - Exercise
    - Make a logical vector `PermitSighted` for 10 facilities.
    - Make another one `usingPPE` for 10 individuals.
    - Use `str()`, `typeof()`, `is.logical()`, to explore them.

Stats brief

- Types of variables
  - Quantitative
  - Qualitative
- Levels of measurement
  - Nominal
  - Ordinal
  - Interval
  - Ratio

Factors

- Integer values that are mapped to "strings"
- Used to represent categorical data
- Each category is called a level
- One of the most powerful uses of R

  – Exercise
    - Make a vector `industryType` using 3 categories – small, medium, large – for 10 facilities only.
    - Make a factor `industryCategory` by calling the function `factor()` on `industryType`.
    - Now use `typeof()`, `is.factor`, `is.character`, `is.integer()` to review these 2 objects.

Things to note…

- Legal names
- Coercion
- Limits
- Common mistakes
  - Confusing factors with characters

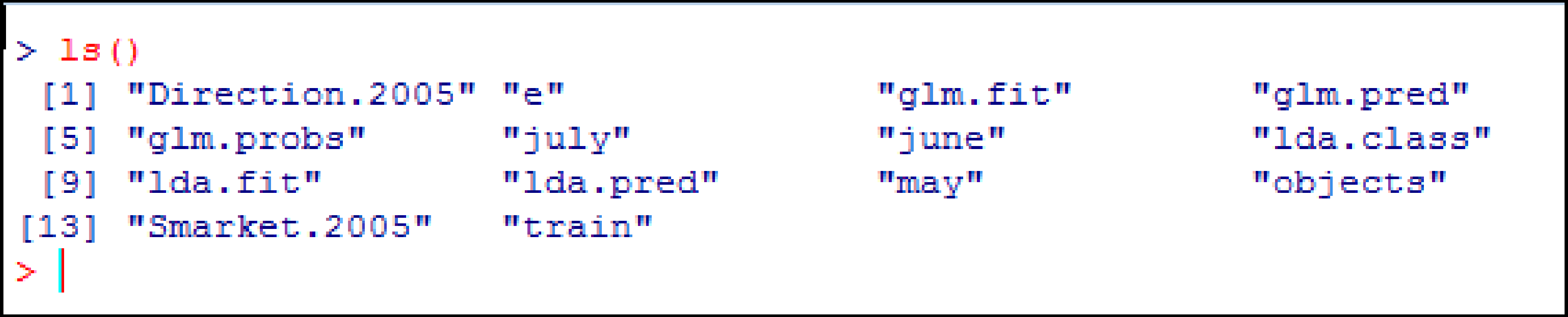
	Homogenous	Heterogeneous
1-dimension	*Atomic vectors*	*Lists*
2-dimensions	*Matrices*	*Data frames*
N-dimensions	*Arrays*	

> R toolbox

- `help()` or `?`
- `getwd(); setwd()`
- `ls()`
- `rm()`
- `save(); load()`

> R toolbox

- An example – `ls()`

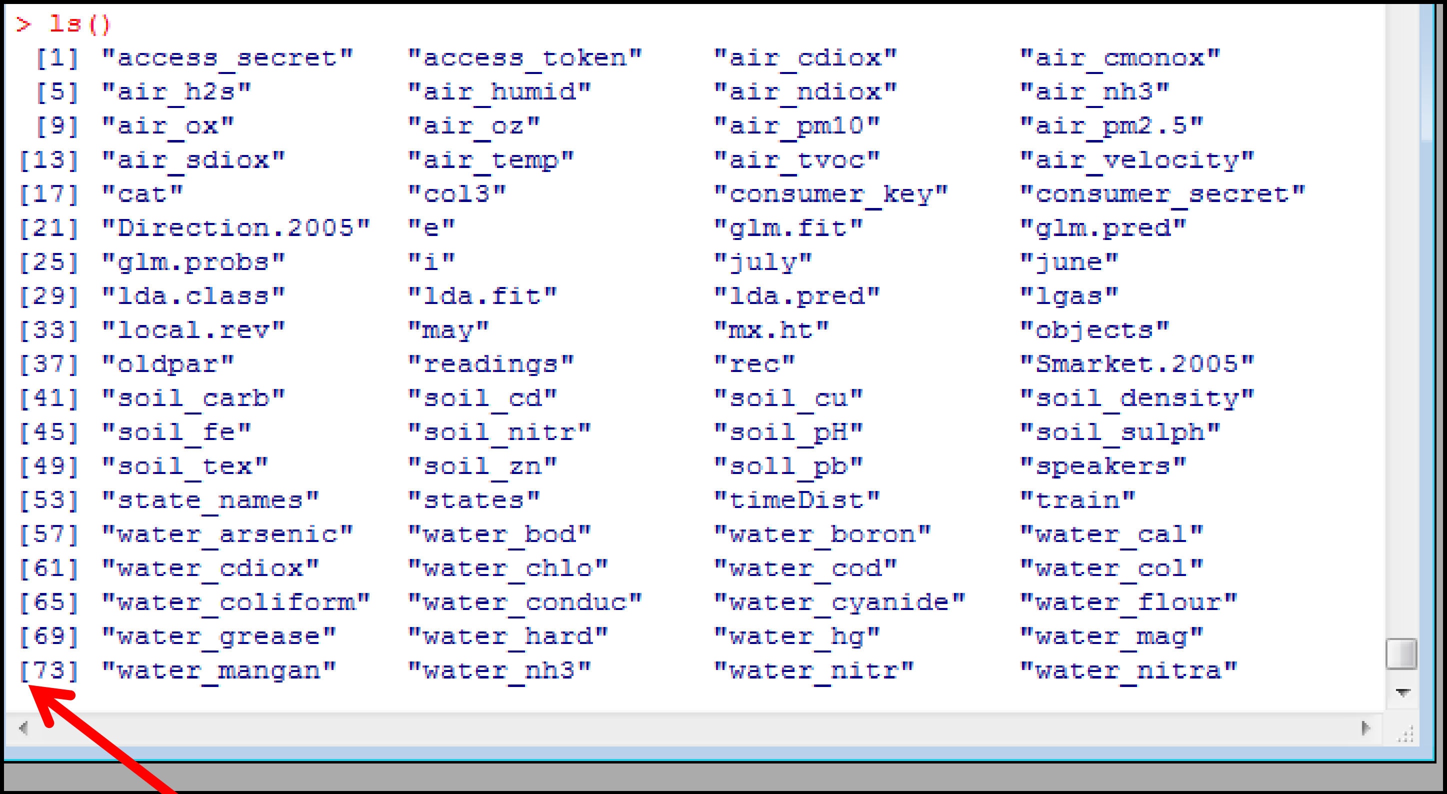


```
> ls()
 [1] "Direction.2005" "e"              "glm.fit"        "glm.pred"
 [5] "glm.probs"      "july"           "june"           "lda.class"
 [9] "lda.fit"        "lda.pred"       "may"            "objects"
[13] "Smarket.2005"   "train"
> 
```

  - It’s relatively easy to see all the objects at a glance
  - Note that this function is called without any arguments

> R toolbox

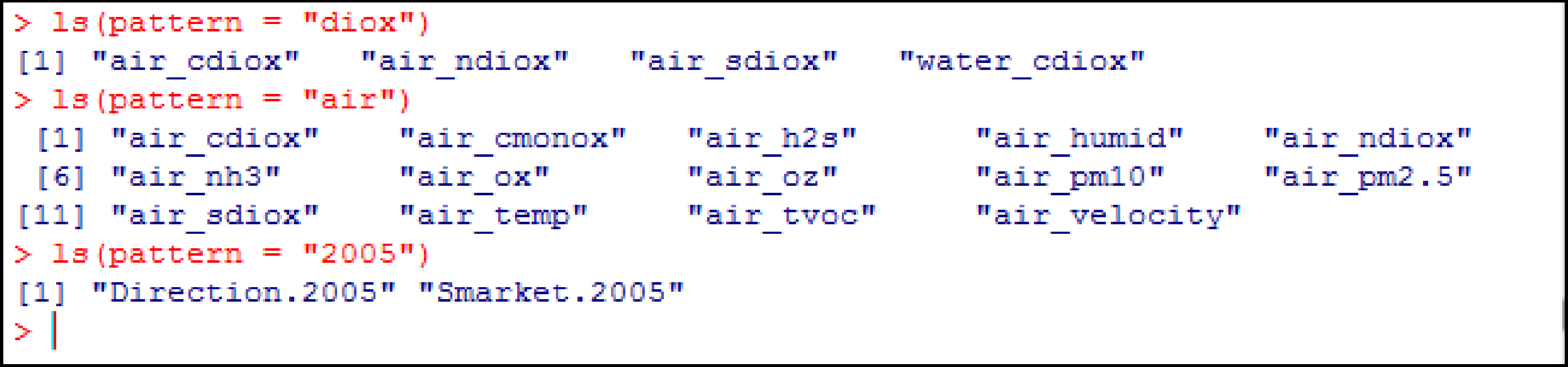
- But how do you deal with this?



```
> ls()
 [1] "access_secret"   "access_token"    "air_cdiox"       "air_cmonox"
 [5] "air_h2s"         "air_humid"       "air_ndiox"       "air_nh3"
 [9] "air_ox"          "air_oz"          "air_pm10"        "air_pm2.5"
[13] "air_sdiox"       "air_temp"        "air_tvoc"        "air_velocity"
[17] "cat"             "col3"            "consumer_key"    "consumer_secret"
[21] "Direction.2005"  "e"               "glm.fit"         "glm.pred"
[25] "glm.probs"       "i"               "july"            "june"
[29] "lda.class"       "lda.fit"         "lda.pred"        "lgas"
[33] "local.rev"       "may"             "mx.ht"           "objects"
[37] "oldpar"          "readings"        "rec"             "Smarket.2005"
[41] "soil_carb"       "soil_cd"         "soil_cu"         "soil_density"
[45] "soil_fe"         "soil_nitr"       "soil_pH"         "soil_sulph"
[49] "soil_tex"        "soil_zn"         "soll_pb"         "speakers"
[53] "state_names"     "states"          "timeDist"        "train"
[57] "water_arsenic"   "water_bod"       "water_boron"     "water_cal"
[61] "water_cdiox"     "water_chlo"      "water_cod"       "water_col"
[65] "water_coliform"  "water_conduc"    "water_cyanide"   "water_flour"
[69] "water_grease"    "water_hard"      "water_hg"        "water_mag"
[73] "water_mangan"    "water_nh3"       "water_nitr"      "water_nitra"
```

Long list beyond screen

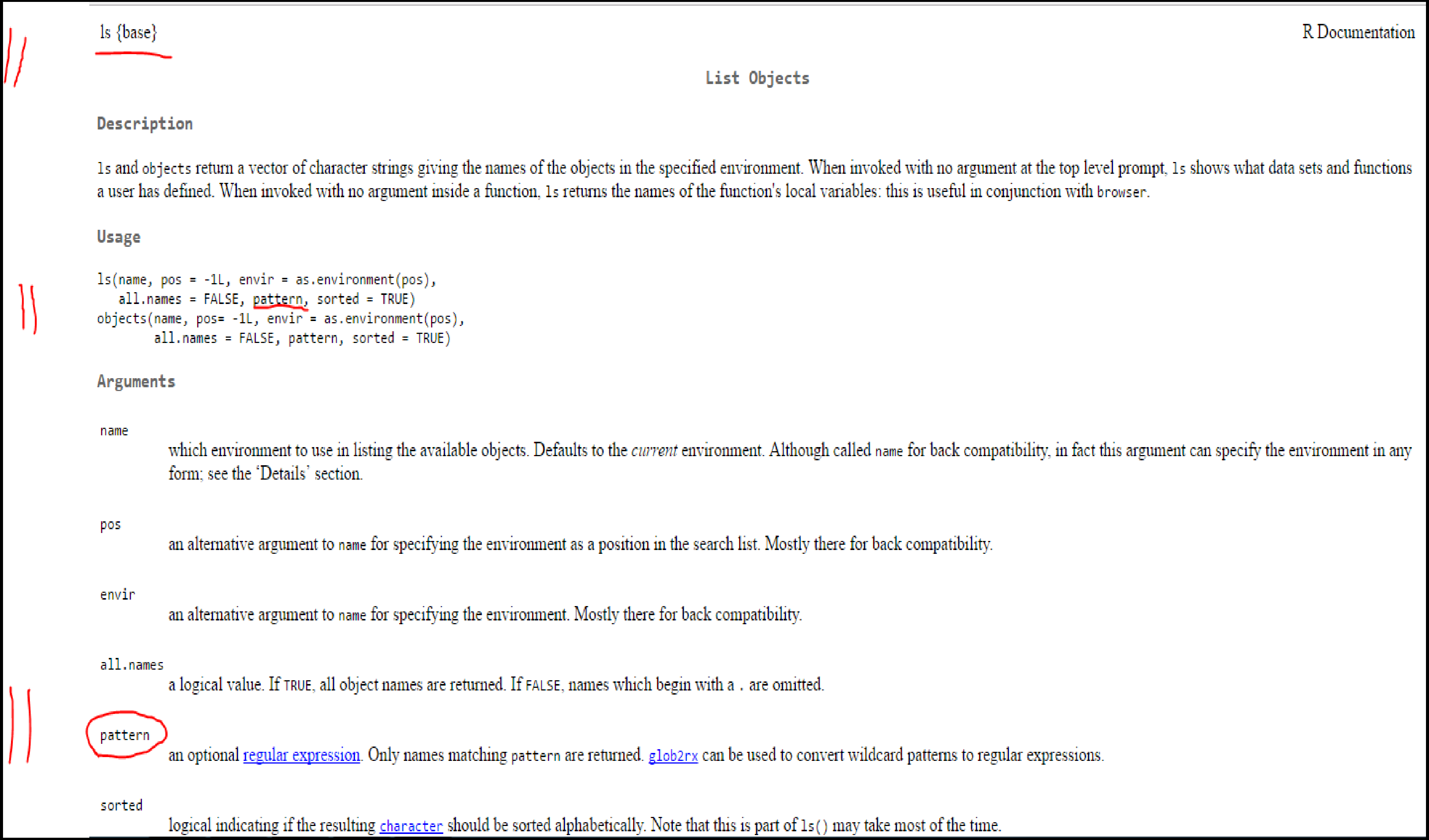
> R toolbox



```
> ls(pattern = "diox")
[1] "air_cdiox"   "air_ndiox"   "air_sdiox"   "water_cdiox"
> ls(pattern = "air")
 [1] "air_cdiox"    "air_cmonox"   "air_h2s"      "air_humid"    "air_ndiox"
 [6] "air_nh3"      "air_ox"       "air_oz"       "air_pm10"     "air_pm2.5"
[11] "air_sdiox"    "air_temp"     "air_tvoc"     "air_velocity"
> ls(pattern = "2005")
[1] "Direction.2005" "Smarket.2005"
> 
```

– Extend use of functions by defining other parameters (optional)
– Use ? to learn about a function's uses & arguments
– When more familiar, use others like `args()`

> R toolbox



ls {base} R Documentation

List Objects

Description

ls and objects return a vector of character strings giving the names of the objects in the specified environment. When invoked with no argument at the top level prompt, ls shows what data sets and functions a user has defined. When invoked with no argument inside a function, ls returns the names of the function's local variables: this is useful in conjunction with browser.

Usage

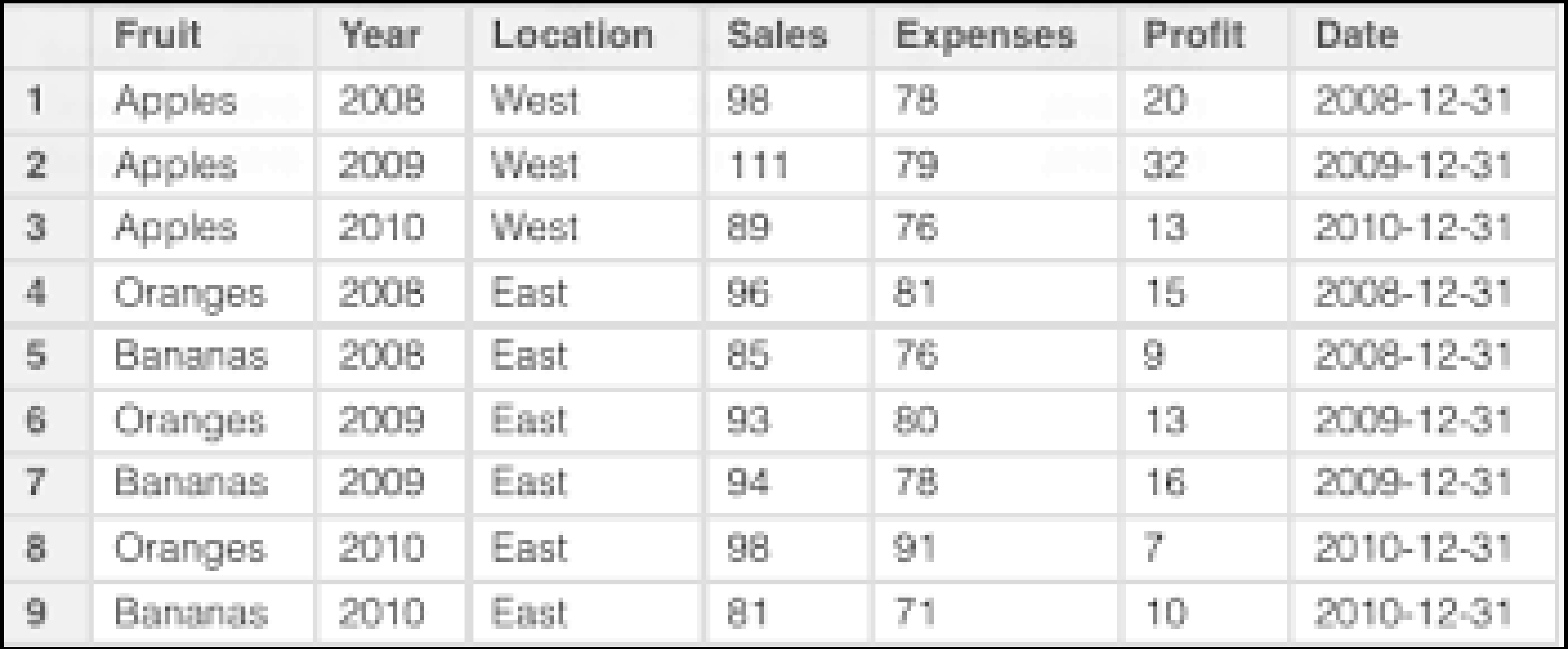
```
ls(name, pos = -1L, envir = as.environment(pos),
   all.names = FALSE, pattern, sorted = TRUE)
objects(name, pos= -1L, envir = as.environment(pos),
        all.names = FALSE, pattern, sorted = TRUE)
```

Arguments

name	which environment to use in listing the available objects. Defaults to the *current* environment. Although called name for back compatibility, in fact this argument can specify the environment in any form; see the 'Details' section.
pos	an alternative argument to name for specifying the environment as a position in the search list. Mostly there for back compatibility.
envir	an alternative argument to name for specifying the environment. Mostly there for back compatibility.
all.names	a logical value. If TRUE, all object names are returned. If FALSE, names which begin with a . are omitted.
pattern	an optional regular expression. Only names matching pattern are returned. glob2rx can be used to convert wildcard patterns to regular expressions.
sorted	logical indicating if the resulting character should be sorted alphabetically. Note that this is part of ls() may take most of the time.

Types of Data Structures

1. Vectors
2. Data frames
3. Matrices
4. Lists
5. Arrays



	Fruit	Year	Location	Sales	Expenses	Profit	Date
1	Apples	2008	West	98	78	20	2008-12-31
2	Apples	2009	West	111	79	32	2009-12-31
3	Apples	2010	West	89	76	13	2010-12-31
4	Oranges	2008	East	96	81	15	2008-12-31
5	Bananas	2008	East	85	76	9	2008-12-31
6	Oranges	2009	East	93	80	13	2009-12-31
7	Bananas	2009	East	94	78	16	2009-12-31
8	Oranges	2010	East	98	91	7	2010-12-31
9	Bananas	2010	East	81	71	10	2010-12-31

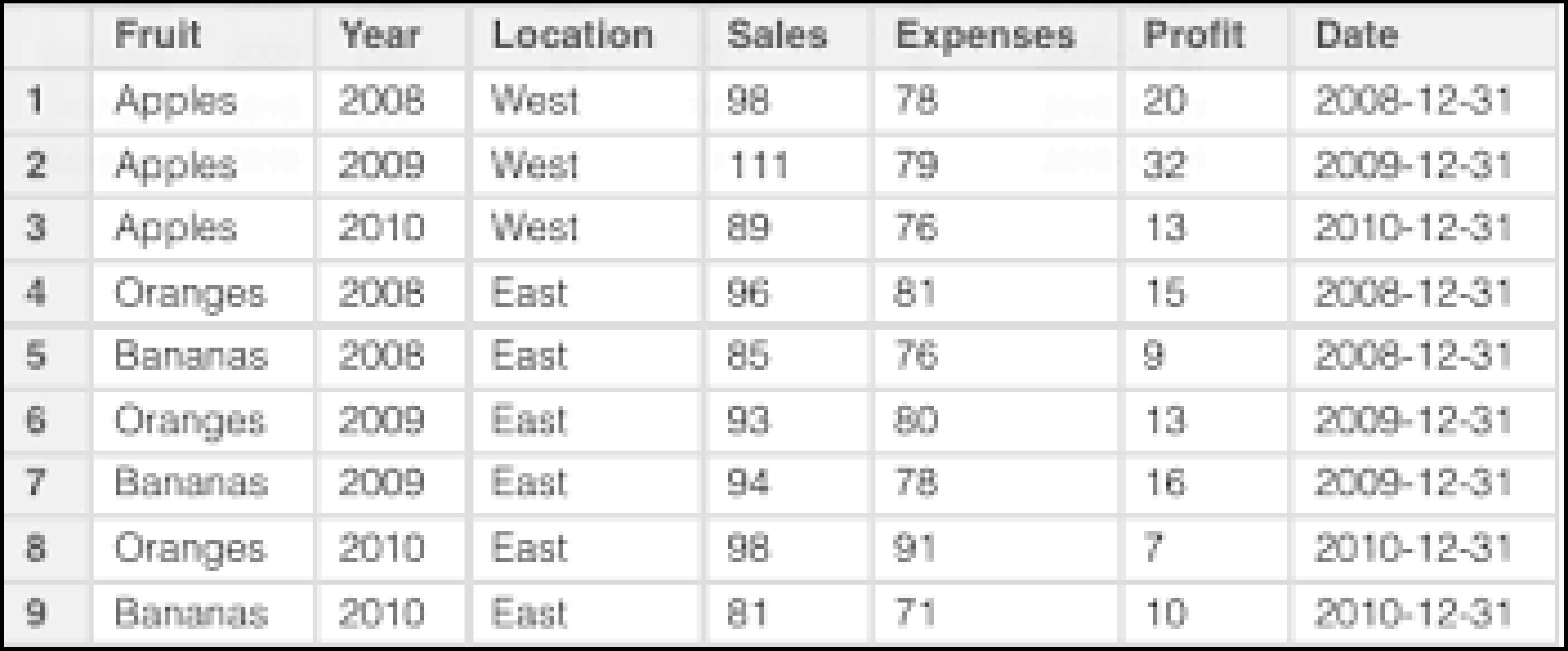
Data frames

- Commonly used R object
- You will either:
  1. Get a data frame
  2. Make your own data frame
- Have columns and rows. Usually represent
  - Columns $\longrightarrow$ variables
  - Rows $\longrightarrow$ observations
- Each column is actually a vector!

header

row

column



	Fruit	Year	Location	Sales	Expenses	Profit	Date
1	Apples	2008	West	98	78	20	2008-12-31
2	Apples	2009	West	111	79	32	2009-12-31
3	Apples	2010	West	89	76	13	2010-12-31
4	Oranges	2008	East	96	81	15	2008-12-31
5	Bananas	2008	East	85	76	9	2008-12-31
6	Oranges	2009	East	93	80	13	2009-12-31
7	Bananas	2009	East	94	78	16	2009-12-31
8	Oranges	2010	East	98	91	7	2010-12-31
9	Bananas	2010	East	81	71	10	2010-12-31

Making a data frame

- A data frame can be built from scratch inside R using the function `data.frame()`
- Combines vectors with same length
- Vectors with different lengths are built using `expand.grid()` – used for simulations

Importing data (simplified)

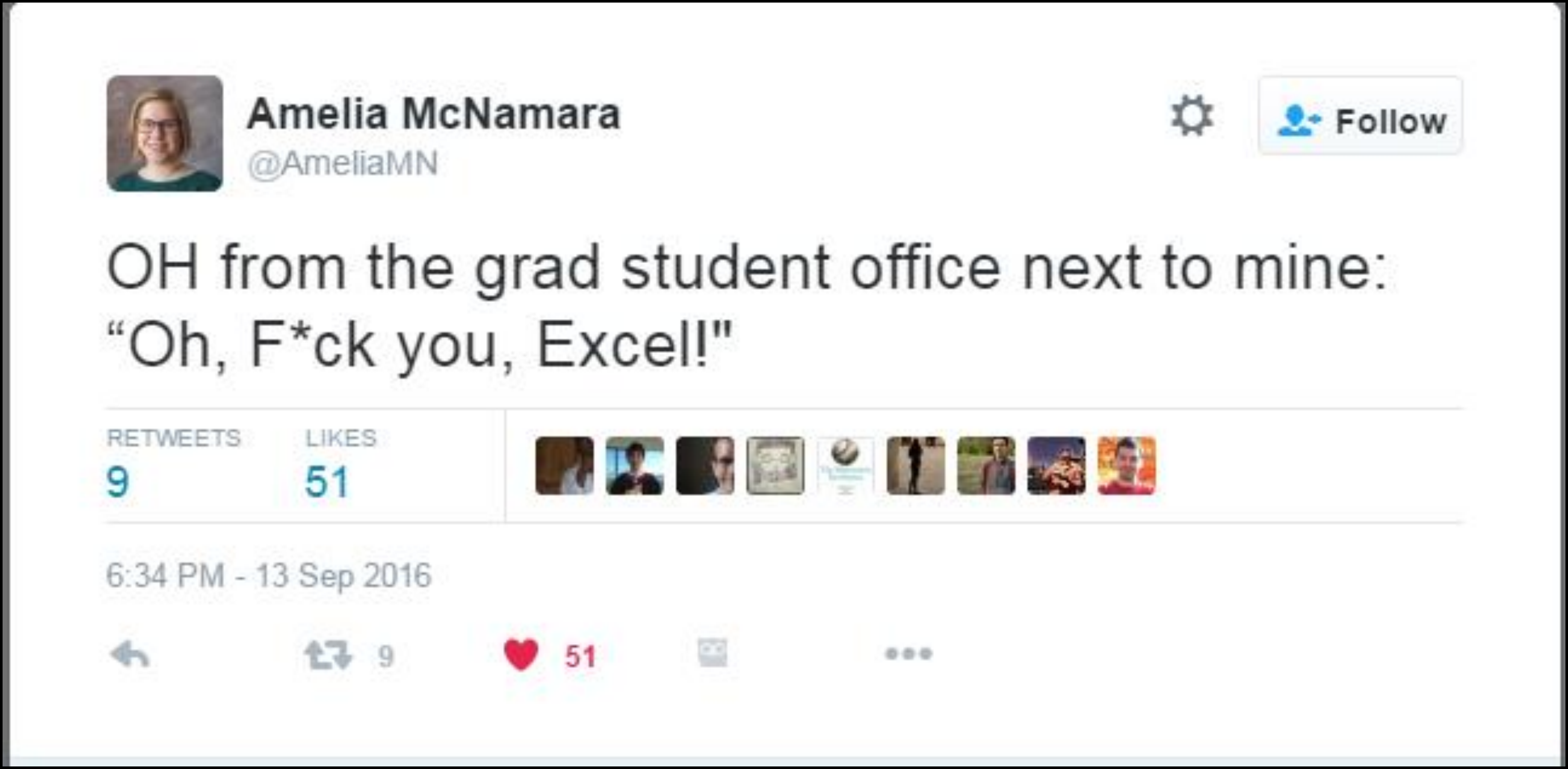
- An important skill
- Many datasets are imported as data frames
- A useful file format is .csv
- Key function is `read.table()` & its variants
- Data can also be imported from other formats e.g. .xls/.xlsx, .spv, .tab,...
- Easiest way to start –> save Excel files as CSV
- Many useful R packages for diff. data formats

Exploring data frames

- A few functions
  - Check dimensions – `dim()`
  - View in spreadsheet format – `View()`
  - Examine structure – `str()`
  - See first/last records – `head()`/`tail()`
  - Summarise variables – `summary()`
  - See/set names of variable – `colnames()`
  - Check if – `is.data.frame()`
  - Change to – `as.data.frame()`
  - Make changes to data using `edit()`

Stats brief

- Grouping data
  - tabulation
- Measures of central tendency
  - Mean
  - Median
  - Mode
- Measures of dispersion
  - Range, IQR
  - Variance, Standard deviation



Amelia McNamara
@AmeliaMN

OH from the grad student office next to mine:
“Oh, F*ck you, Excel!"

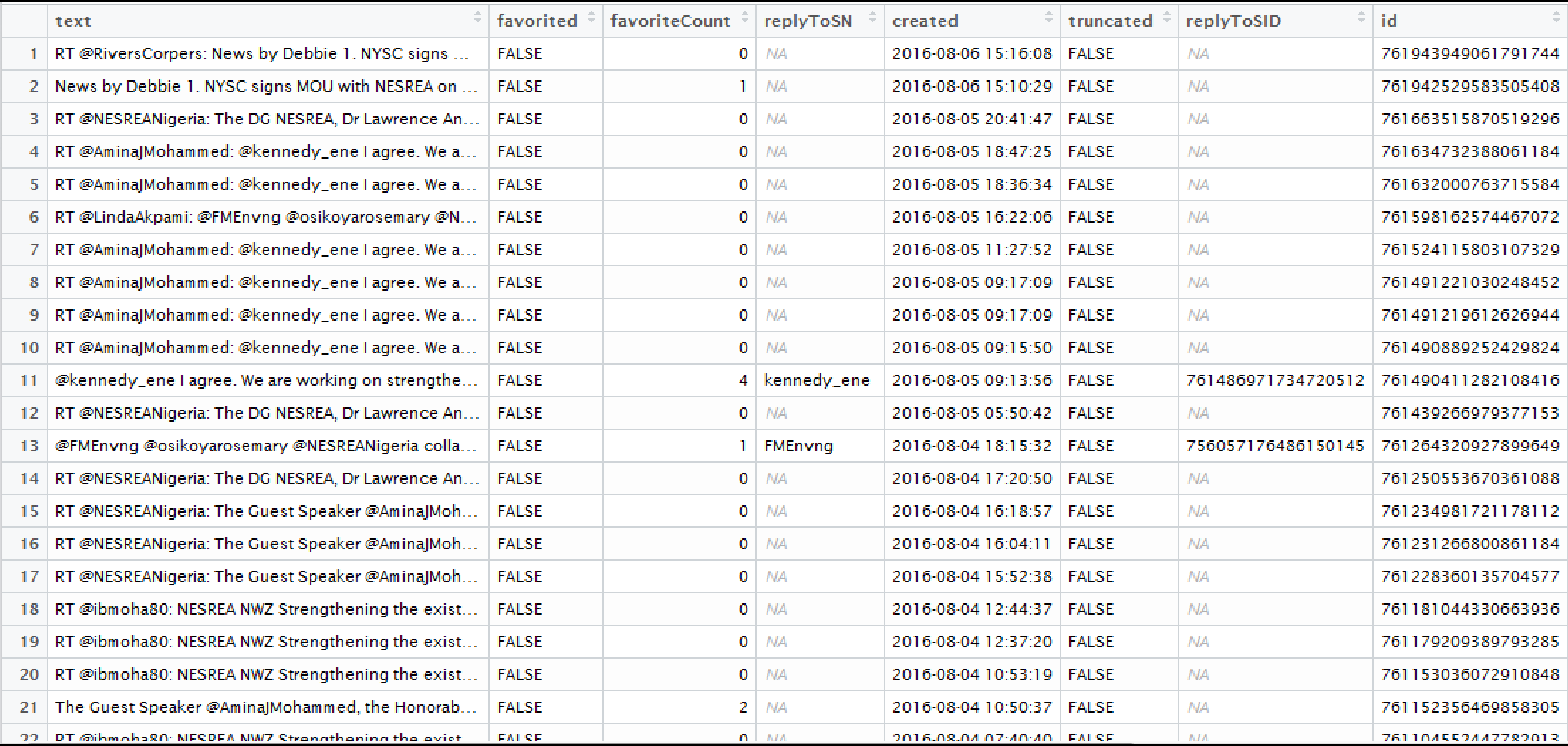
RETWEETS 9 LIKES 51

6:34 PM - 13 Sep 2016

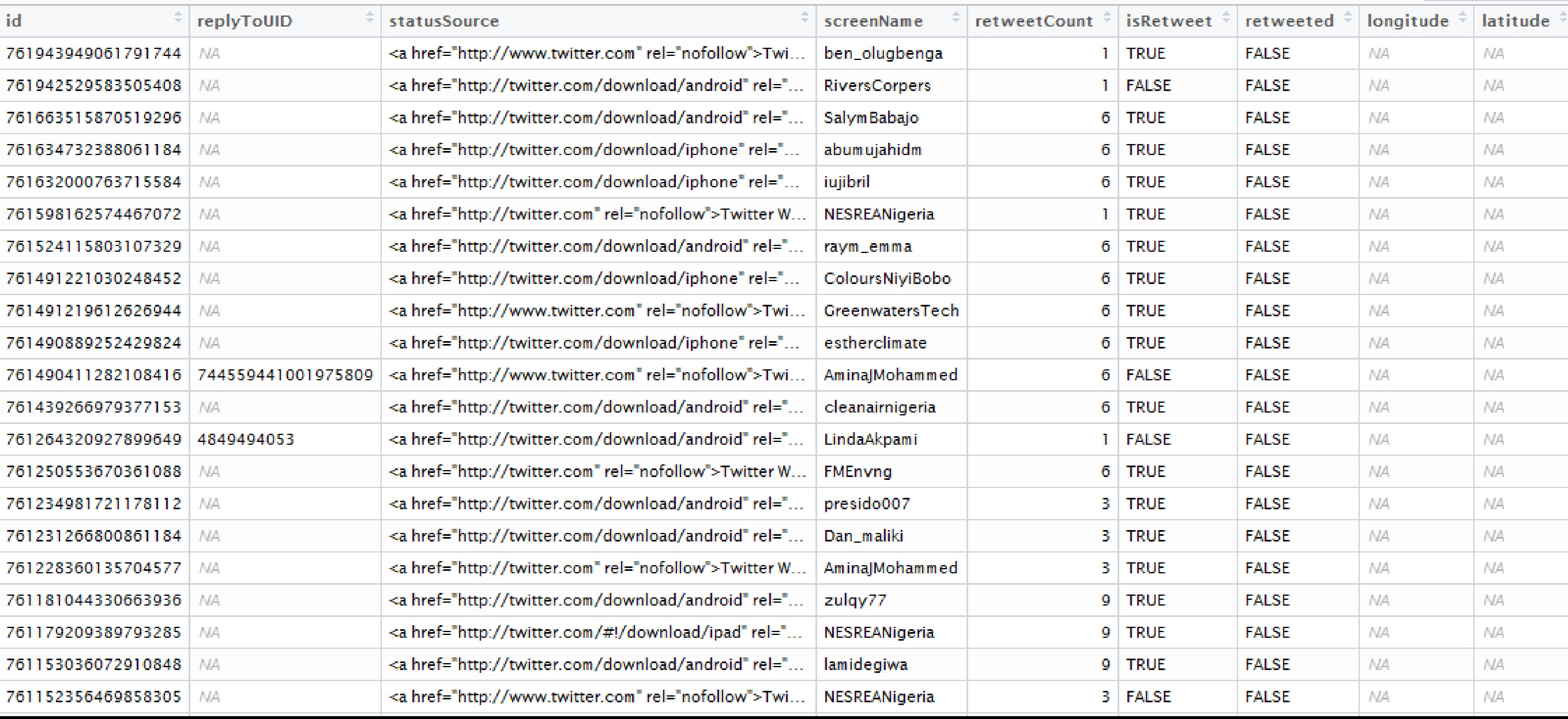
Example with Twitter data

- Preliminaries
  - R package called `twitteR`; access Twitter data
  - Call `install.packages("twitter")`
  - Use `library(help = "twitteR")` to access package's DESCRIPTION file
  - Documentation: `help(package = "twitteR")`
  - Many packages have `vignette()`s – show you how to use package

- More preliminaries
  - Authenticate i.e. log on through Twitter OAuth
  - Our details are found in authentication.R
  - Establish connection with `setup_twitter_oauth()`
    - This gives us access to the Twitter data
  - Download the data with `searchTwitter()`
  - Convert to data frame with `twListToDF()`



	text	favorited	favoriteCount	replyToSN	created	truncated	replyToSID	id
1	RT @RiversCorpers: News by Debbie 1. NYSC signs …	FALSE	0	NA	2016-08-06 15:16:08	FALSE	NA	761943949061791744
2	News by Debbie 1. NYSC signs MOU with NESREA on …	FALSE	1	NA	2016-08-06 15:10:29	FALSE	NA	761942529583505408
3	RT @NESREANigeria: The DG NESREA, Dr Lawrence An…	FALSE	0	NA	2016-08-05 20:41:47	FALSE	NA	761663515870519296
4	RT @AminaJMohammed: @kennedy_ene I agree. We a…	FALSE	0	NA	2016-08-05 18:47:25	FALSE	NA	761634732388061184
5	RT @AminaJMohammed: @kennedy_ene I agree. We a…	FALSE	0	NA	2016-08-05 18:36:34	FALSE	NA	761632000763715584
6	RT @LindaAkpami: @FMEnvng @osikoyarosemary @N…	FALSE	0	NA	2016-08-05 16:22:06	FALSE	NA	761598162574467072
7	RT @AminaJMohammed: @kennedy_ene I agree. We a…	FALSE	0	NA	2016-08-05 11:27:52	FALSE	NA	761524115803107329
8	RT @AminaJMohammed: @kennedy_ene I agree. We a…	FALSE	0	NA	2016-08-05 09:17:09	FALSE	NA	761491221030248452
9	RT @AminaJMohammed: @kennedy_ene I agree. We a…	FALSE	0	NA	2016-08-05 09:17:09	FALSE	NA	761491219612626944
10	RT @AminaJMohammed: @kennedy_ene I agree. We a…	FALSE	0	NA	2016-08-05 09:15:50	FALSE	NA	761490889252429824
11	@kennedy_ene I agree. We are working on strengthe…	FALSE	4	kennedy_ene	2016-08-05 09:13:56	FALSE	761486971734720512	761490411282108416
12	RT @NESREANigeria: The DG NESREA, Dr Lawrence An…	FALSE	0	NA	2016-08-05 05:50:42	FALSE	NA	761439266979377153
13	@FMEnvng @osikoyarosemary @NESREANigeria colla…	FALSE	1	FMEnvng	2016-08-04 18:15:32	FALSE	756057176486150145	761264320927899649
14	RT @NESREANigeria: The DG NESREA, Dr Lawrence An…	FALSE	0	NA	2016-08-04 17:20:50	FALSE	NA	761250553670361088
15	RT @NESREANigeria: The Guest Speaker @AminaJMoh…	FALSE	0	NA	2016-08-04 16:18:57	FALSE	NA	761234981721178112
16	RT @NESREANigeria: The Guest Speaker @AminaJMoh…	FALSE	0	NA	2016-08-04 16:04:11	FALSE	NA	761231266800861184
17	RT @NESREANigeria: The Guest Speaker @AminaJMoh…	FALSE	0	NA	2016-08-04 15:52:38	FALSE	NA	761228360135704577
18	RT @ibmoha80: NESREA NWZ Strengthening the exist…	FALSE	0	NA	2016-08-04 12:44:37	FALSE	NA	761181044330663936
19	RT @ibmoha80: NESREA NWZ Strengthening the exist…	FALSE	0	NA	2016-08-04 12:37:20	FALSE	NA	761179209389793285
20	RT @ibmoha80: NESREA NWZ Strengthening the exist…	FALSE	0	NA	2016-08-04 10:53:19	FALSE	NA	761153036072910848
21	The Guest Speaker @AminaJMohammed, the Honorab…	FALSE	2	NA	2016-08-04 10:50:37	FALSE	NA	761152356469858305
22	RT @ibmoha80: NESREA NWZ Strengthening the exist…	FALSE	0	NA	2016-08-04 07:40:40	FALSE	NA	761104552447782013



id	replyToUID	statusSource	screenName	retweetCount	isRetweet	retweeted	longitude	latitude
761943949061791744	NA	<a href="http://www.twitter.com" rel="nofollow">Twi...	ben_olugbenga	1	TRUE	FALSE	NA	NA
761942529583505408	NA	<a href="http://twitter.com/download/android" rel="...	RiversCorpers	1	FALSE	FALSE	NA	NA
761663515870519296	NA	<a href="http://twitter.com/download/android" rel="...	SalymBabajo	6	TRUE	FALSE	NA	NA
761634732388061184	NA	<a href="http://twitter.com/download/iphone" rel="...	abumujahidm	6	TRUE	FALSE	NA	NA
761632000763715584	NA	<a href="http://twitter.com/download/iphone" rel="...	iujibril	6	TRUE	FALSE	NA	NA
761598162574467072	NA	<a href="http://twitter.com" rel="nofollow">Twitter W...	NESREANigeria	1	TRUE	FALSE	NA	NA
761524115803107329	NA	<a href="http://twitter.com/download/android" rel="...	raym_emma	6	TRUE	FALSE	NA	NA
761491221030248452	NA	<a href="http://twitter.com/download/iphone" rel="...	ColoursNiyiBobo	6	TRUE	FALSE	NA	NA
761491219612626944	NA	<a href="http://www.twitter.com" rel="nofollow">Twi...	GreenwatersTech	6	TRUE	FALSE	NA	NA
761490889252429824	NA	<a href="http://twitter.com/download/iphone" rel="...	estherclimate	6	TRUE	FALSE	NA	NA
761490411282108416	744559441001975809	<a href="http://www.twitter.com" rel="nofollow">Twi...	AminaJMohammed	6	FALSE	FALSE	NA	NA
761439266979377153	NA	<a href="http://twitter.com/download/android" rel="...	cleanairnigeria	6	TRUE	FALSE	NA	NA
761264320927899649	4849494053	<a href="http://twitter.com/download/android" rel="...	LindaAkpami	1	FALSE	FALSE	NA	NA
761250553670361088	NA	<a href="http://twitter.com" rel="nofollow">Twitter W...	FMEnvng	6	TRUE	FALSE	NA	NA
761234981721178112	NA	<a href="http://twitter.com/download/android" rel="...	presido007	3	TRUE	FALSE	NA	NA
761231266800861184	NA	<a href="http://twitter.com/download/android" rel="...	Dan_maliki	3	TRUE	FALSE	NA	NA
761228360135704577	NA	<a href="http://twitter.com" rel="nofollow">Twitter W...	AminaJMohammed	3	TRUE	FALSE	NA	NA
761181044330663936	NA	<a href="http://twitter.com/download/android" rel="...	zulqy77	9	TRUE	FALSE	NA	NA
761179209389793285	NA	<a href="http://twitter.com/#!/download/ipad" rel="...	NESREANigeria	9	TRUE	FALSE	NA	NA
761153036072910848	NA	<a href="http://twitter.com/download/android" rel="...	lamidegiwa	9	TRUE	FALSE	NA	NA
761152356469858305	NA	<a href="http://www.twitter.com" rel="nofollow">Twi...	NESREANigeria	3	FALSE	FALSE	NA	NA

Subsetting

- The `$` operator
  - Look at `str()` again
  - Note the `$` in the output
  - You can pick out individual columns with syntax `dataframe$columnname`
  - Same thing as a vector
  - Get to get or set variables

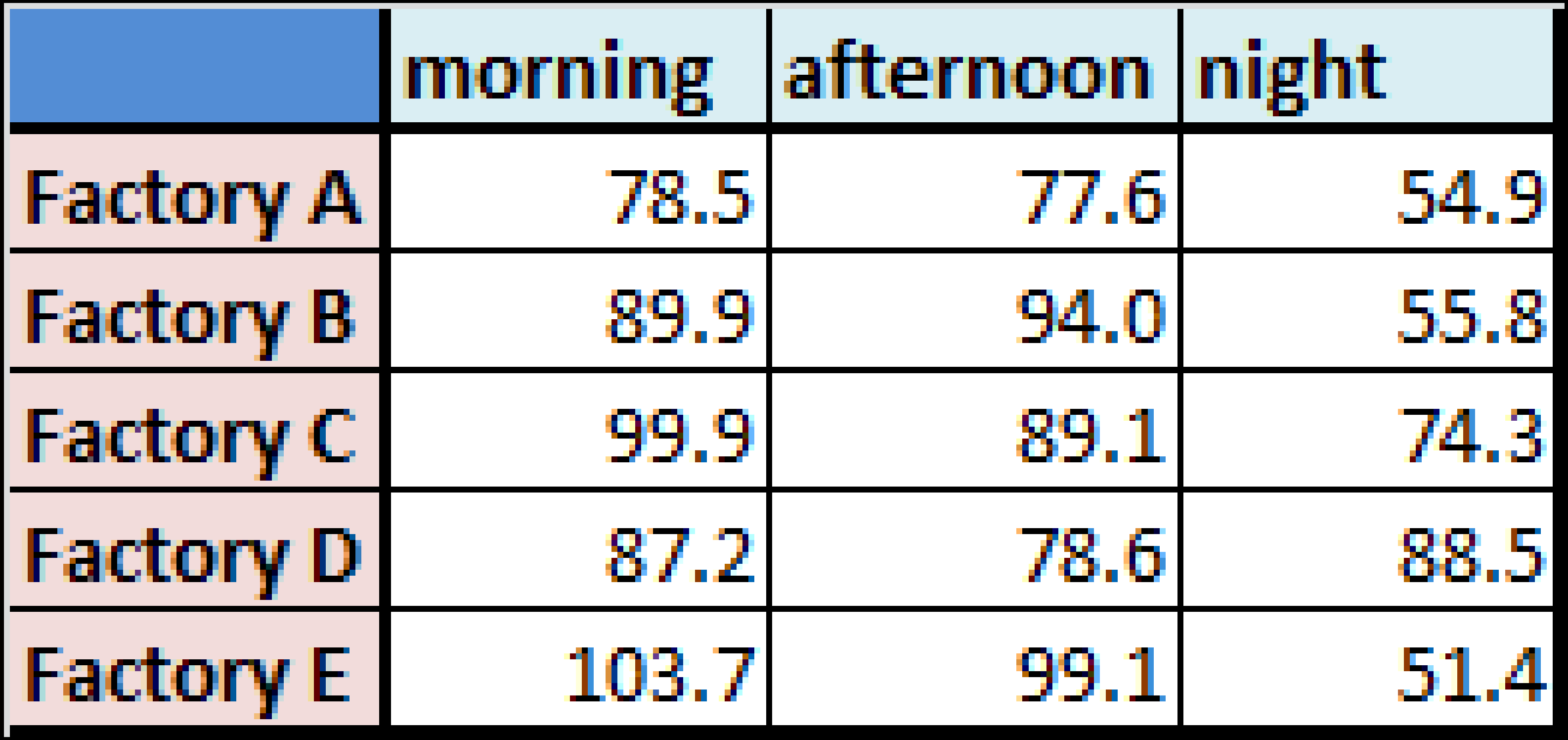
> R toolbox

```
> search()
> detach()
> library()
> require()
> saveRDS()
> readRDS()
> file.edit()
```

	Homogenous	Heterogeneous
1-dimension	*Atomic vectors*	*Lists*
2-dimensions	*Matrices*	*Data frames*
N-dimensions	*Arrays*	

Types of Data Structures

1. Vectors
2. Data frames
3. Matrices
4. Lists
5. Arrays



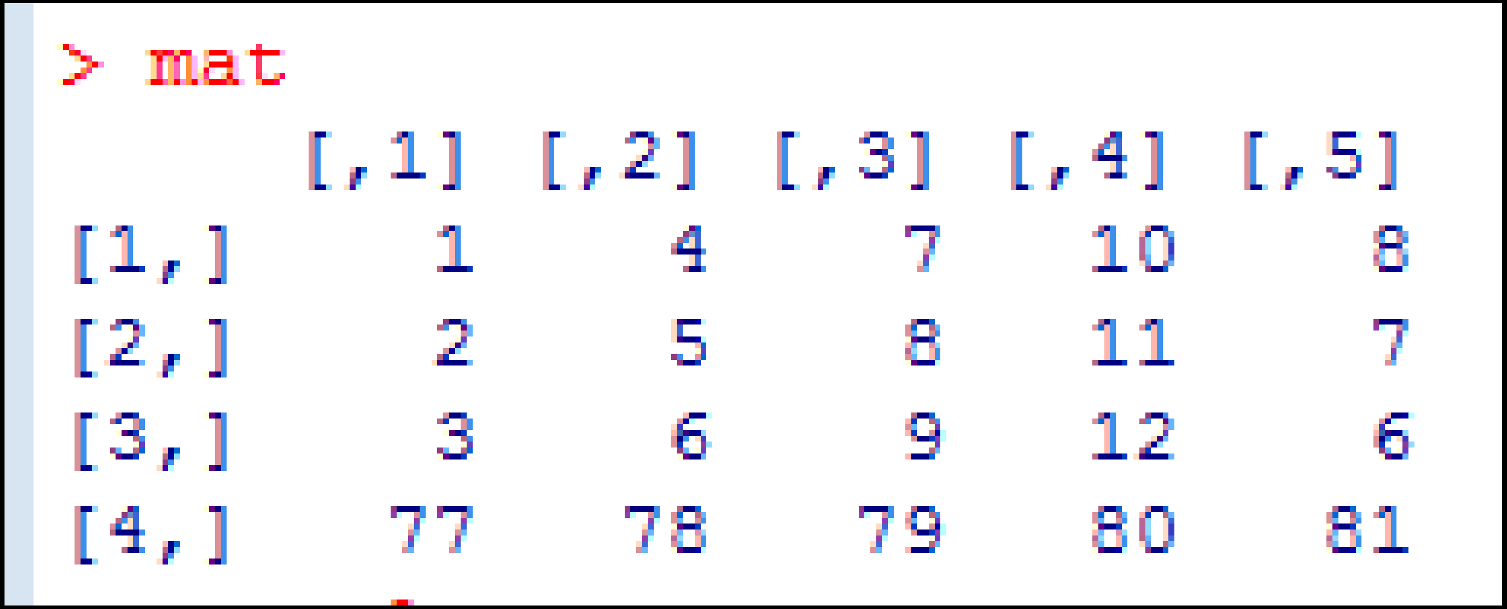
	morning	afternoon	night
Factory A	78.5	77.6	54.9
Factory B	89.9	94.0	55.8
Factory C	99.9	89.1	74.3
Factory D	87.2	78.6	88.5
Factory E	103.7	99.1	51.4

Matrices

- Like data frame are 2-dimensional
  - rows & columns
- Like vectors, only of particular type
  - integer, character, numeric, logical, etc.
- Can be built using `matrix()`
- Others include `rbind()`, `cbind()`
- Matrix arithmetic possible in R but not our focus

- Matrix creation is also useful in other operations such as drawing multiple plots
- Exploring a matrix is somewhat similar to that of a data frame
  - `dim()`
  - `typeof()`
  - `class()`

- Run the script create-matrix.R to create a matrix, mat



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- bracket operator is also used for data frames

Indexing

- This is a crucial aspect of R programming
- Enable you to get or set elements of a data structure
- Bracket operator "[ ]" is used for indexing
  - For vectors: [ ]
  - For matrices & data frames: [,]
  - For lists: [ ] and [[ ]]

- The format for indexing is `[row, column]`
  - `[1, 2]` means “first row, second column”

- This notation is like mapping the location of an element in a 2-dimensional data structure.

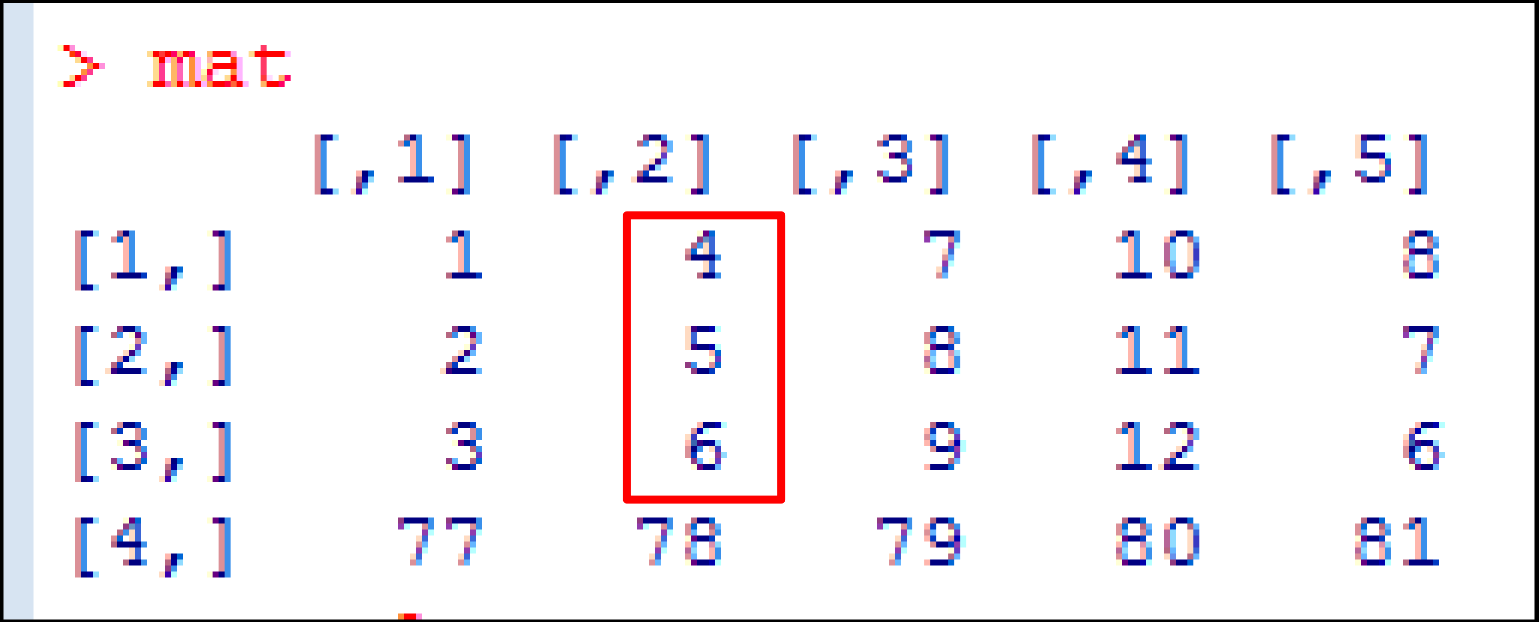
- If you want to **get** a matrix value on the 4th row and the 3rd column, you run

```
matrix[4, 3]
```

- If you want to **change** a matrix value on the 4th row and the 3rd column, you run

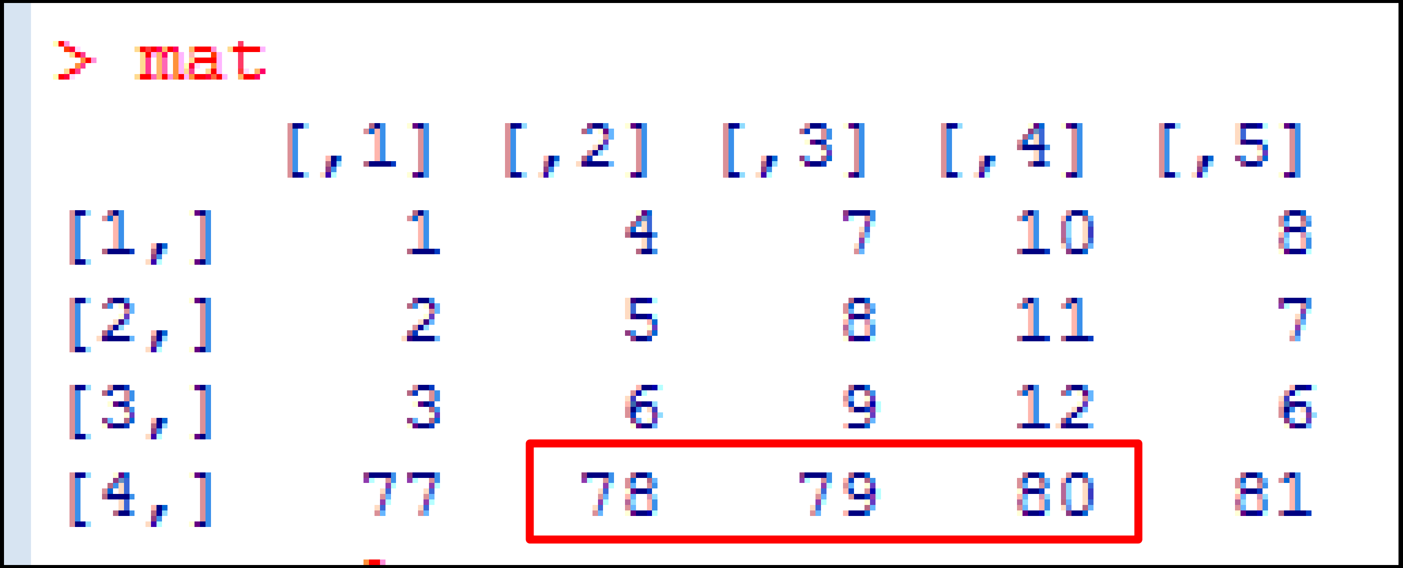
```
matrix[4, 3] <- <new value>
```

- Ranges also work well with indexing
  - First 3 rows of column 2 is written as `[1:3, 2]`



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

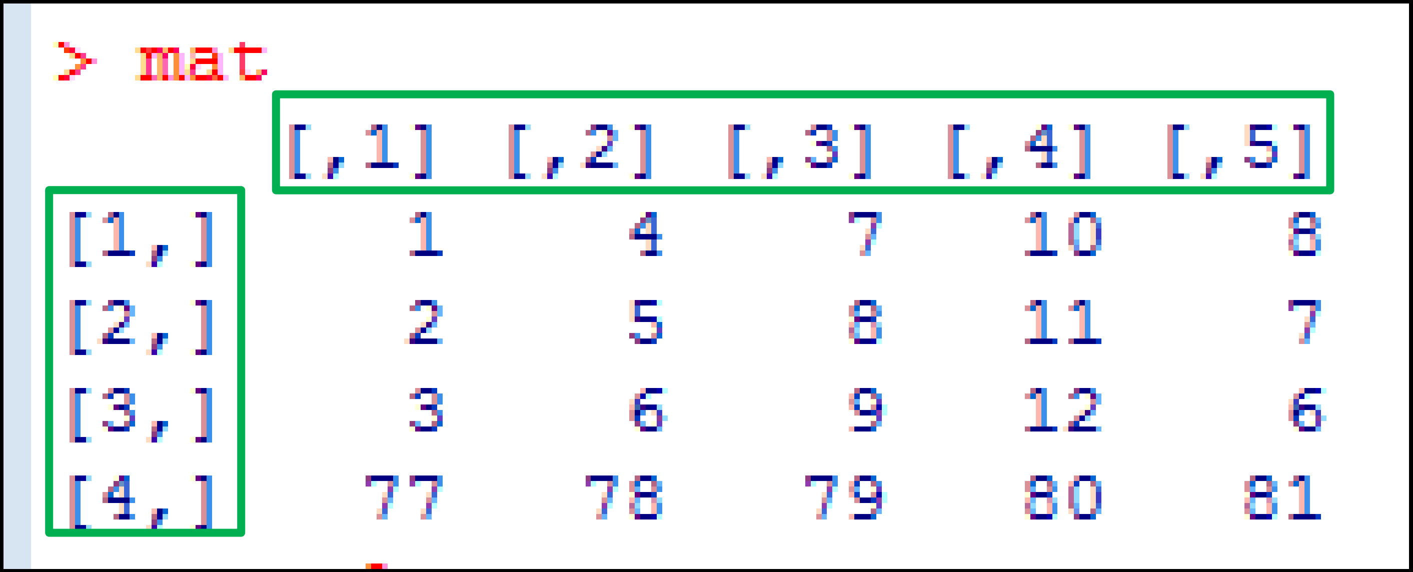
– Row 4 of columns 2 to 4 is coded as `[4, 2:4]`



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- Note the indices (or is it indexes?) of the matrix along the margins of the output

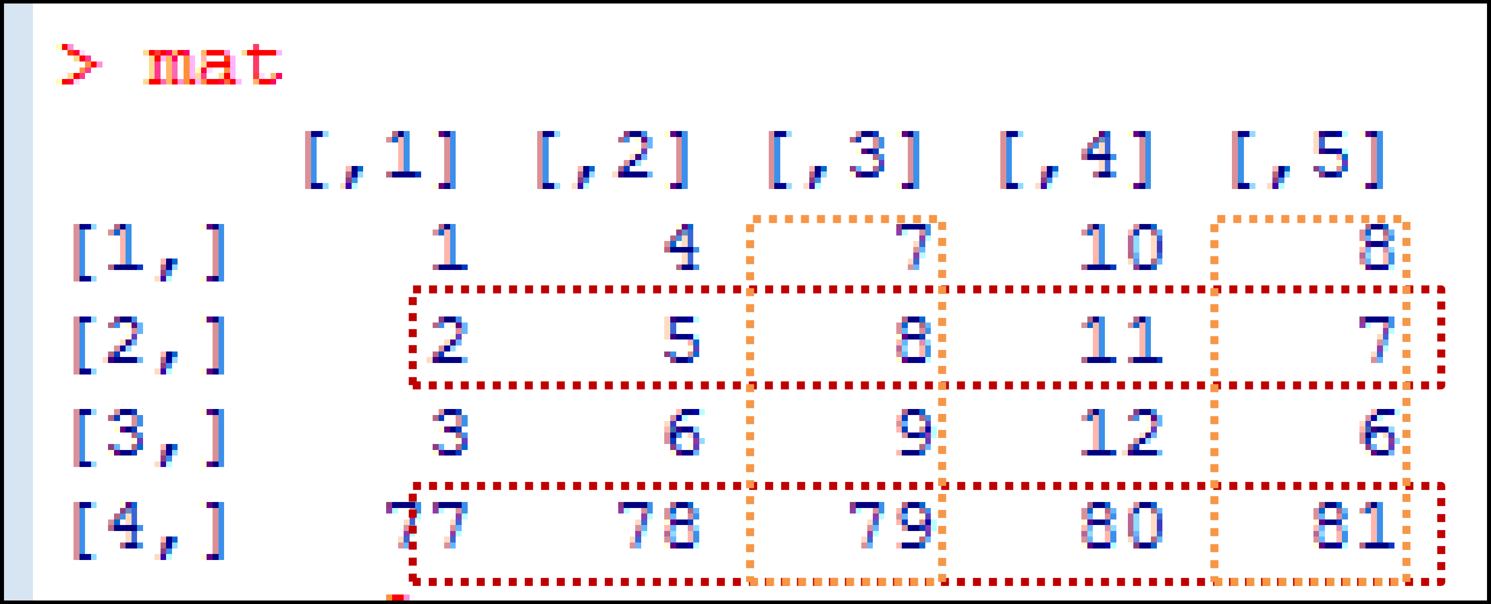
(also, learn that there is no trivial or frivolous output in R. Studying it can teach you a lot!)



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

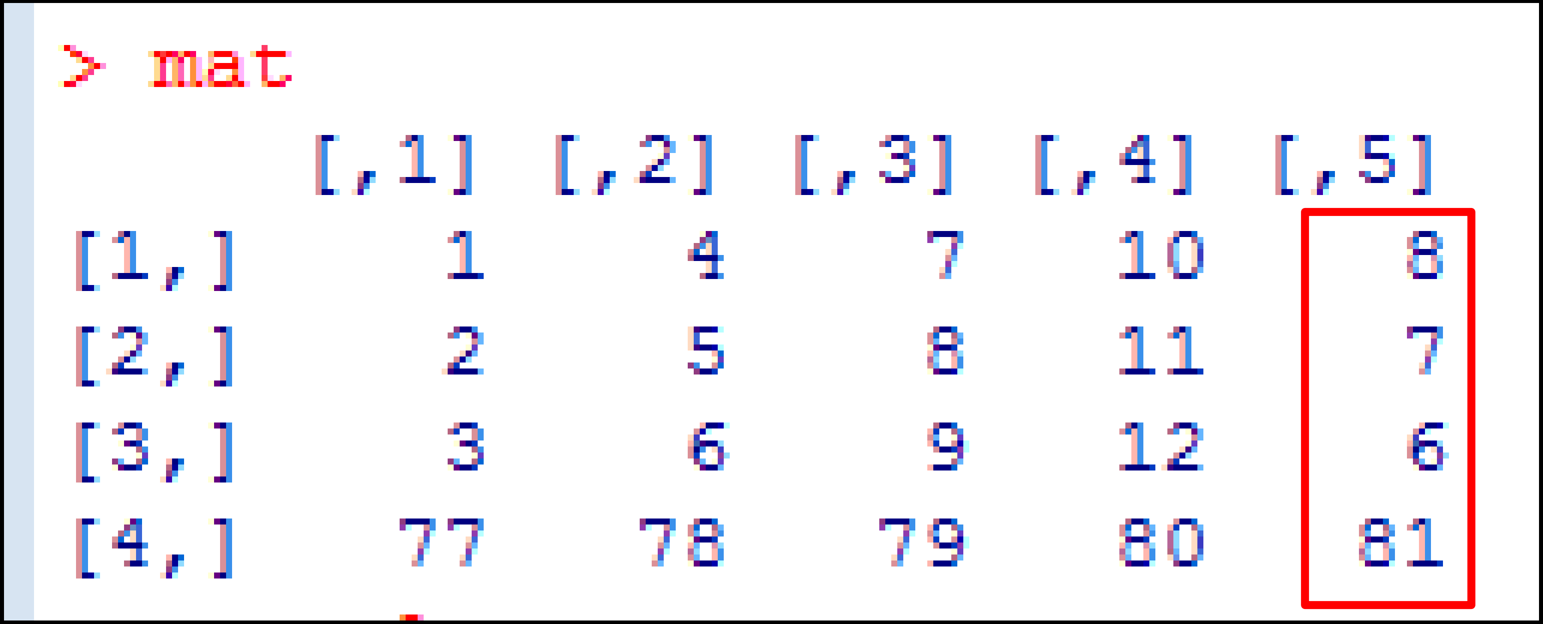
- Nit-picking is also possible
  - To select rows 2 & 4 of columns 3 & 5 write

```
[c(2, 4), c(3, 5)] # concatenate function
```



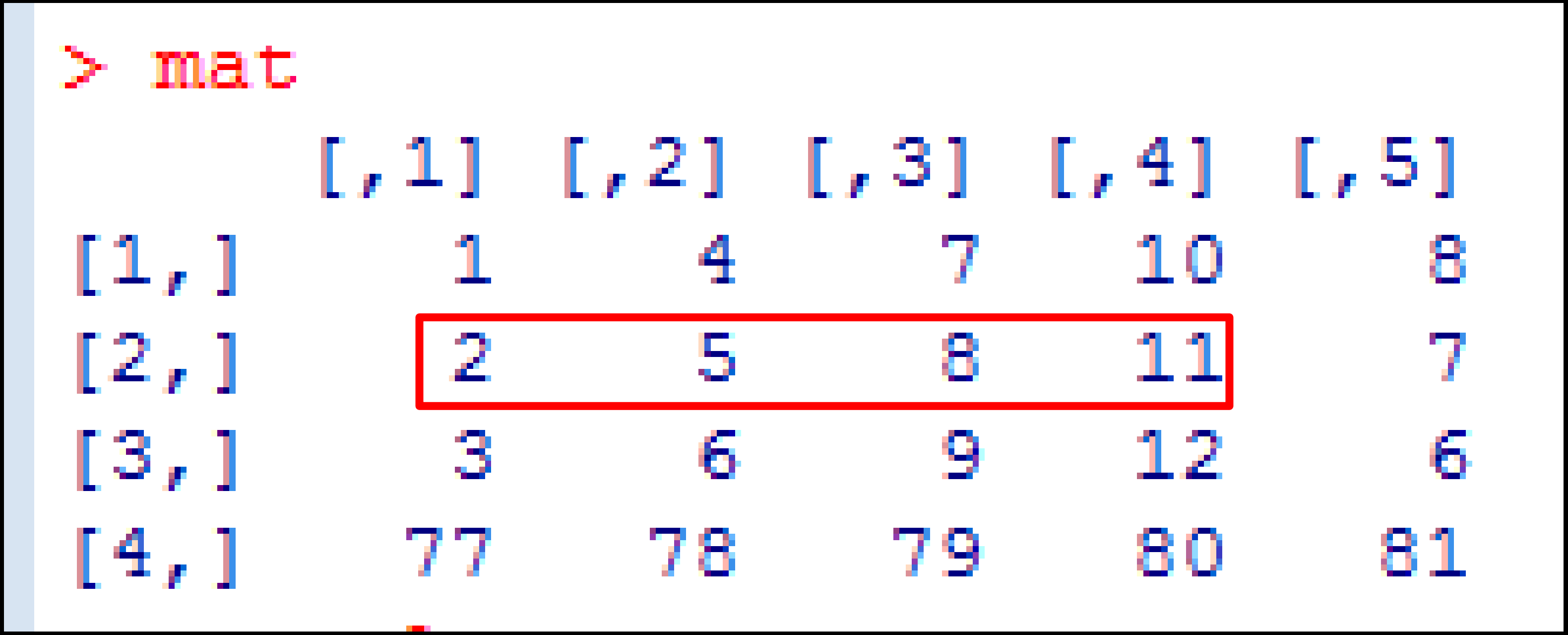
```
> mat
      [,1] [,2] [,3] [,4] [,5]
[1,]     1    4    7   10    8
[2,]     2    5    8   11    7
[3,]     3    6    9   12    6
[4,]    77   78   79   80   81
```

- An empty value means ‘ALL’
  - E.g. `[, 5]` (all rows in column 5)



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- Negative indexing is there too
  - Row 2 except column 5 is written as `[2, -5]`



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

Indexing works for vectors, too!

- Bracket notation applies to vectors
- Because they have 1 dimension, there will be no comma inside the brackets

  i.e. `[   ]` and not `[  ,  ]`
- Try it out!

	Homogenous	**Heterogeneous**
1-dimension	*Atomic vectors*	*Lists*
2-dimensions	*Matrices*	*Data frames*
N-dimensions	*Arrays*	

Lists

- R lists are unique data structures – very versatile
- Heterogenous – can hold any kind of R object in memory, singly or in combination
- A list can also contain lists

Making a list

- The function `list()` is used to form a list
- Download the sample script create-list.R to see how lists can be formed
- Go run each line of code and carefully study the output.
- Observe that the code constructs list(s) of all the R objects studied thus far (and more!)

Quiz

1. How many list elements are there in `mumbo_jumbo`?
2. List the different types of R objects you can find among the elements.

Indexing lists

- The bracket operator `[ ]` is also used to extract elements of a list
  - Single `[ ]`
    - Will give you a list
  - Double `[[ ]]`
    - Will give you the element

  - Looking at `str()`, you may notice that `$` also identifies each element of the list.
    - To access them the elements must be named – easily done with the function names()

Exercises

- Run `mumbo_jumbo[1]`, and then `mumbo_jumbo[[1]]`
- Now call `typeof()` on each of them e.g. `typeof(mumbo_jumbo[1])`
- Discuss the output.

- A well-known expert recently shared a photo on Twitter to help understanding of R list indexing. Click here to view it. Discuss.

Data frames are lists!

- Recall that `$` is used to mark **named** columns of a data frame
- Also recall that each column in a data frame can stand alone as a vector
- A data frame is essentially a list with these characteristics
  - Made up of vectors
  - All the vectors are of equal length
  - Run `typeof()` on any data frame to check.

Challenge

- Name the elements of `mumbo_jumbo` with the `names()` function as follows:
  1. Calcium
  2. EAR_done
  3. Bin.code
  4. Bin.code_expt
  5. filesOnFlash
  6. Circle
  7. table_def
  8. MS_simul
  9. NESREA_hex
  10. NESREA_bits
- Use the name and `$` to extract any element from the list.
- What is the type of the extracted element? What does that tell you about the nature of indexing done with `$`.

Conclusion

- The aim of this training session is to equip you with skills and tools to begin basic analysis of social media data.
- Social media sites offer an API for easy access to organised data.
- Facebook & Twitter data are downloaded as objects that can be handled in R i.e. lists, data frames, etc.
- To make use of these data, we will use the R packages called `twitteR` and `Rfacebook`, developed by Jeff Gentry & Pablo Barbera, respectively.
- We will start with Twitter first. To prepare for the next session:
  - Run `install.packages(“twitteR”)`, to get the package (requires internet connection)
  - If curious, you can look through the documentation by running `help(package = “twitteR”)`

- Thanks for your kind attention.