Practical R: Data Ingestion

Abhijit Dasgupta

BIOF 339

A quick refresh

- We talked about various data structures in R
- The primacy of the `data.frame`
  - Extracting individual variables from a data frame
  - `breast_cancer$ER.Status`, `breast_cancer[,'ER.Status']`, `breast_cancer[['ER.Status']]`
  - Extracting rows of a `data.frame`
- Identifying data classes using the `class` function
- Recognizing different classes: `numeric`, `character`, `factor`, `Date`, ..
  - testing for a class: `is.numeric`
  - converting to a class: `as.numeric`

RMarkdown tip of the day

You can add options to each R chunk to add or suppress output

Option	Property
echo=TRUE/FALSE	Does the document show the R code
eval=TRUE/FALSE	Does the chunk get evaluated by R
message=TRUE/FALSE	Do messages get printed
warning=TRUE/FALSE	Do warnings get printed

You can also set these globally in a RMD file by putting the following in the first R chunk:

```
knitr::opts_chunk$set(echo=T, eval=T, message=F, warning=F)
```

See here for the full gory details

Note that the correct way to write `TRUE` and `FALSE` is all caps. They can be shortened to `T` and `F` respectively, but it's better to get used to the full word.

Package tip of the semester

Use

```
library(tidyverse)
```

or

```
pacman::p_load('tidyverse')
```

for pretty much every R script and R Markdown file (put this at the top of a script file, but after the header in a R Markdown)

Data ingestion

Data ingestion

Unlike Excel, you have to pull data into R for R to operate on it

Typically your data is in some sort of file (Excel, csv, sas7bdat, dta, txt)

You need to find a way to pull it into R

The GUI you've used is one way, but not very programmatic

Data ingestion

Type	Function	Package	Notes
csv	read_csv	readr	Takes care of formatting
csv	read.csv	base	Built in
csv	fread	data.table	Fastest
Excel	read_excel	readxl	
sas7bdat	read_sas	haven	SAS format
sav	read_spss	haven	SPSS format
dta	read_dta	haven	Stata format

Data ingestion

We will use this csv data and this Excel data for the following:

```
brca_clinical <- readr::read_csv('../data/BreastCancer_Clinical.csv')
brca_clinical2 <- data.table::fread('../data/BreastCancer_Clinical.csv')
```

```
str(brca_clinical)
```

```
spec_tbl_df [77 × 30] (S3: spec_tbl_df/tbl_df/tbl/da
 $ Complete TCGA ID                  : chr [1:77] '
 $ Gender                            : chr [1:77] '
 $ Age at Initial Pathologic Diagnosis: num [1:77] 4
 $ ER Status                         : chr [1:77] '
 $ PR Status                         : chr [1:77] '
 $ HER2 Final Status                 : chr [1:77] '
 $ Tumor                             : chr [1:77] '
 $ Tumor--T1 Coded                   : chr [1:77] '
 $ Node                              : chr [1:77] '
 $ Node-Coded                        : chr [1:77] '
 $ Metastasis                        : chr [1:77] '
 $ Metastasis-Coded                  : chr [1:77] '
 $ AJCC Stage                        : chr [1:77] '
 $ Converted Stage                   : chr [1:77] '
 $ Survival Data Form                : chr [1:77] '
 $ Vital Status                      : chr [1:77] '
 $ Days to Date of Last Contact      : num [1:77] 7
 $ Days to date of Death             : num [1:77] 7
```

```
str(brca_clinical2)
```

```
Classes 'data.table' and 'data.frame':    77 obs. of
 $ Complete TCGA ID                  : chr  "TCGA-A
 $ Gender                            : chr  "FEMALE
 $ Age at Initial Pathologic Diagnosis: int  40 56 5
 $ ER Status                         : chr  "Negati
 $ PR Status                         : chr  "Negati
 $ HER2 Final Status                 : chr  "Negati
 $ Tumor                             : chr  "T2" "T
 $ Tumor--T1 Coded                   : chr  "T_Othe
 $ Node                              : chr  "N0" "N
 $ Node-Coded                        : chr  "Negati
 $ Metastasis                        : chr  "M0" "M
 $ Metastasis-Coded                  : chr  "Negati
 $ AJCC Stage                        : chr  "Stage
 $ Converted Stage                   : chr  "Stage
 $ Survival Data Form                : chr  "follow
 $ Vital Status                      : chr  "DECEAS
 $ Days to Date of Last Contact      : int  754 169
 $ Days to date of Death             : num  754 169
```

A note on two "super"-data.frame objects

A `tibble`

```
# A tibble: 6 × 30
  `Complete TCGA ID` Gender `Age at Initial Patholog
  <chr>              <chr>                        <
1 TCGA-A2-A0CM       FEMALE
2 TCGA-BH-A18Q       FEMALE
3 TCGA-A7-A0CE       FEMALE
4 TCGA-D8-A142       FEMALE
5 TCGA-AO-A0J6       FEMALE
6 TCGA-A2-A0YM       FEMALE
# … with 25 more variables: HER2 Final Status <chr>,
#   Tumor--T1 Coded <chr>, Node <chr>, Node-Coded <c
#   Metastasis-Coded <chr>, AJCC Stage <chr>, Conver
#   Survival Data Form <chr>, Vital Status <chr>,
#   Days to Date of Last Contact <dbl>, Days to date
#   OS event <dbl>, OS Time <dbl>, PAM50 mRNA <chr>,
#   SigClust Unsupervised mRNA <dbl>, SigClust Intri
```

A `data.table`

```
   Complete TCGA ID Gender Age at Initial Pathologic
1:     TCGA-A2-A0CM FEMALE
2:     TCGA-BH-A18Q FEMALE
3:     TCGA-A7-A0CE FEMALE
4:     TCGA-D8-A142 FEMALE
5:     TCGA-AO-A0J6 FEMALE
6:     TCGA-A2-A0YM FEMALE
   PR Status HER2 Final Status Tumor Tumor--T1 Codec
1:  Negative          Negative    T2         T_Other
2:  Negative          Negative    T2         T_Other
3:  Negative          Negative    T2         T_Other
4:  Negative          Negative    T3         T_Other
5:  Negative          Negative    T2         T_Other
6:  Negative          Negative    T2         T_Other
   Metastasis-Coded AJCC Stage Converted Stage Survi
1:         Negative  Stage IIA       Stage IIA
2:         Negative  Stage IIB   No_Conversion
3:         Negative  Stage IIA       Stage IIA
4:         Negative  Stage IIB       Stage IIB
5:         Negative  Stage IIA       Stage IIA
6:         Negative  Stage IIA       Stage IIA
   Days to Date of Last Contact Days to date of Deat
1:                          754                   75
2:                         1692                  169
3:                          309                    N
4:                          425                    N
5:                          775                    N
```

A note on two "super"-data.frame objects

- A `tibble` works pretty much like any `data.frame`, but the printing is a little saner
- A `data.table` is faster, has more inherent functionality, but has a very different syntax

We'll work almost entirely with `tibble`'s and not `data.table`

Suggested modifications:

- If using `fread`, convert the resulting object to a `data.frame` or `tibble` using `as_data_frame()` or `as_tibble()`
- Convert the column names to not have spaces using, for example,

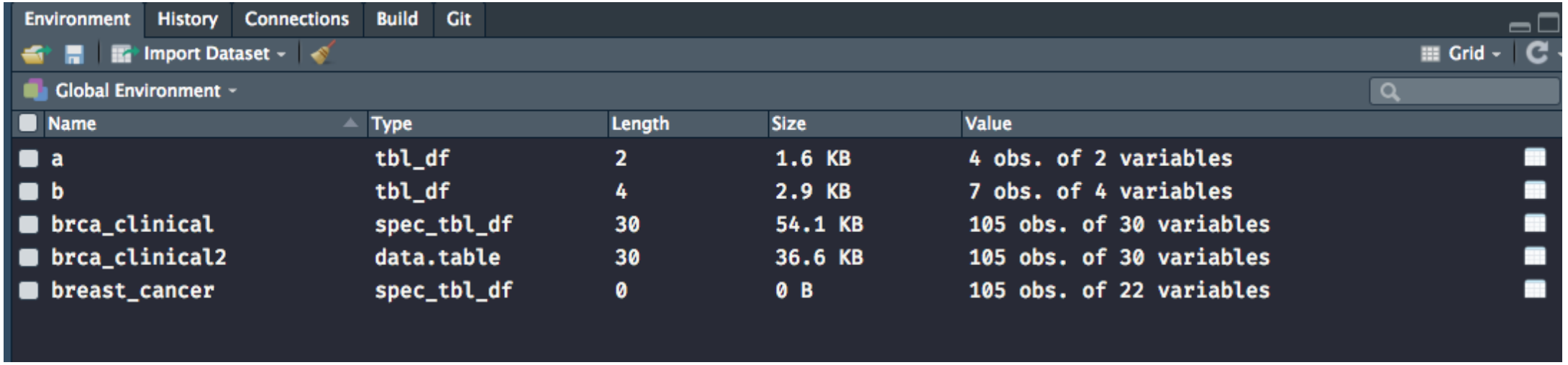
```
brca_clinical <- janitor::clean_names(brca_clinical)
```


janitor::clean_names
convert all column names to _*_ case!
(not so awesome column names)
FUN(units)!
Sample!
Such.Mess!
Ooo.OOF-.
MESS
Mm.,OOPS
clean_names()
(your data frame + case choice here)
CHOOSE:
snake
lower_camel
big_camel
screaming_snake
+ more!
WAY MORE DEALWITHABLE COLUMN NAMES!
so_much_nicer
_yay_ya_
hope_rest
_wo
ooo
_cool
_than
_good_feel_
_nic
teddy
_a_
snake
HORST 2019

Data ingestion

Note that you **have** to give a name to what you're importing using `read_*` or whatever you're using, otherwise it won't stay in R

```
brca_clinical <- readr::read_csv('../data/BreastCancer_Clinical.csv')
```



> See what happens if you don't give a name to a dataset you ingest.

Reading Excel

You can find the names of the sheets in an Excel file:

```
readxl::excel_sheets('../data/BreastCancer.xlsx')
```

```
[1] "Cllinical"  "Expression"
```

So you can ingest a particular sheet from an Excel file using

```
brca_expression <- readxl::read_excel('../data/BreastCancer.xlsx', sheet='Expression')
```

Data export

Data export

Type	Function	Package	Notes
csv	write_csv	readr	Takes care of formatting
csv	write.csv	base	Built in
csv	fwrite	data.table	Fastest
Excel	write.xlsx	openxlsx	
sas7bdat	write_sas	haven	SAS format
sav	write_spss	haven	SPSS format
dta	write_dta	haven	Stata format

We'll often save tabular results using these functions

These can also be useful for exporting results, but the R Markdown related packages are better for that

Simplifying import/export

We'll be using a package that makes this easier.

It's called rio and it has two basic functions: `import` and `export`.

The `rio` package uses the different packages mentioned earlier but unifies it into a single syntax

For example:

```
rio::import('data/clinical_data_breast_cancer_modified.csv')
```

rio reads the end of the file being imported or exported and decides which functions from which package should be used for the job.

rio accesses different packages that are right for each job, so you don't have to.

Simplifying import/export

You can also import multiple sheets from Excel, or multiple objects from .RData files, into a list of data frames

```
dat <- rio::import_list('data/BreastCancer.xlsx')
```

```
class(dat)
```

```
[1] "list"
```

```
names(dat)
```

```
[1] "Cllinical"  "Expression"
```

```
map_chr(dat, class)
```

```
   Cllinical   Expression
"data.frame" "data.frame"
```

Saving your work

You would often like to store intermediate datasets, and final datasets, so that you can access them quickly.

There are several ways of saving even large datasets so that they can be quickly accessed.

Function	Package	Example	Retrieving the stored data
saveRDS	base	`saveRDS(weather, file = 'weather.rds')`	`weather <- readRDS('weather.rds')`
write_fst	fst	`write_fst(weather, file='weather.fst')`	`weather <- read_fst('weather.fst')`

These methods are meant for storing single objects

Saving your work

If you want to store all of your objects into a single file, you can store them in a .RData file.

```
save.image(file="<filename>.RData")
```

To keep multiple specified objects in a .RData file,

```
save(<obj1>, <obj2>, <obj3>, file = "<filename>.RData")
```

Retrieving your work

You can retrieve the objects in a .RData file using the function `load`.

```
load(file = "<filename>.RData")
```

This will store each object in its original name in your R environment.