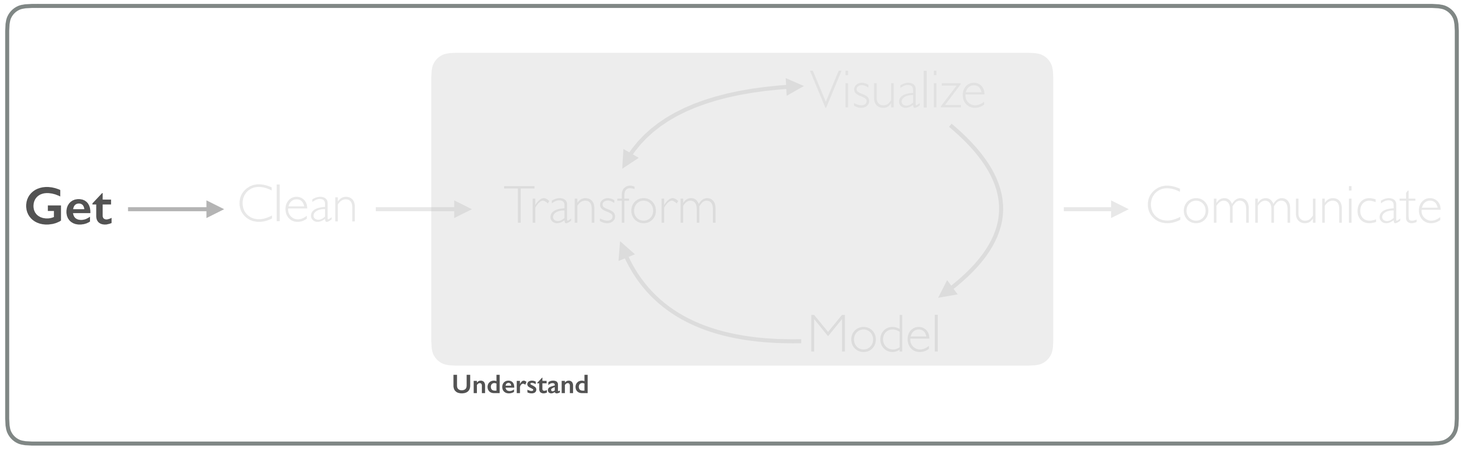
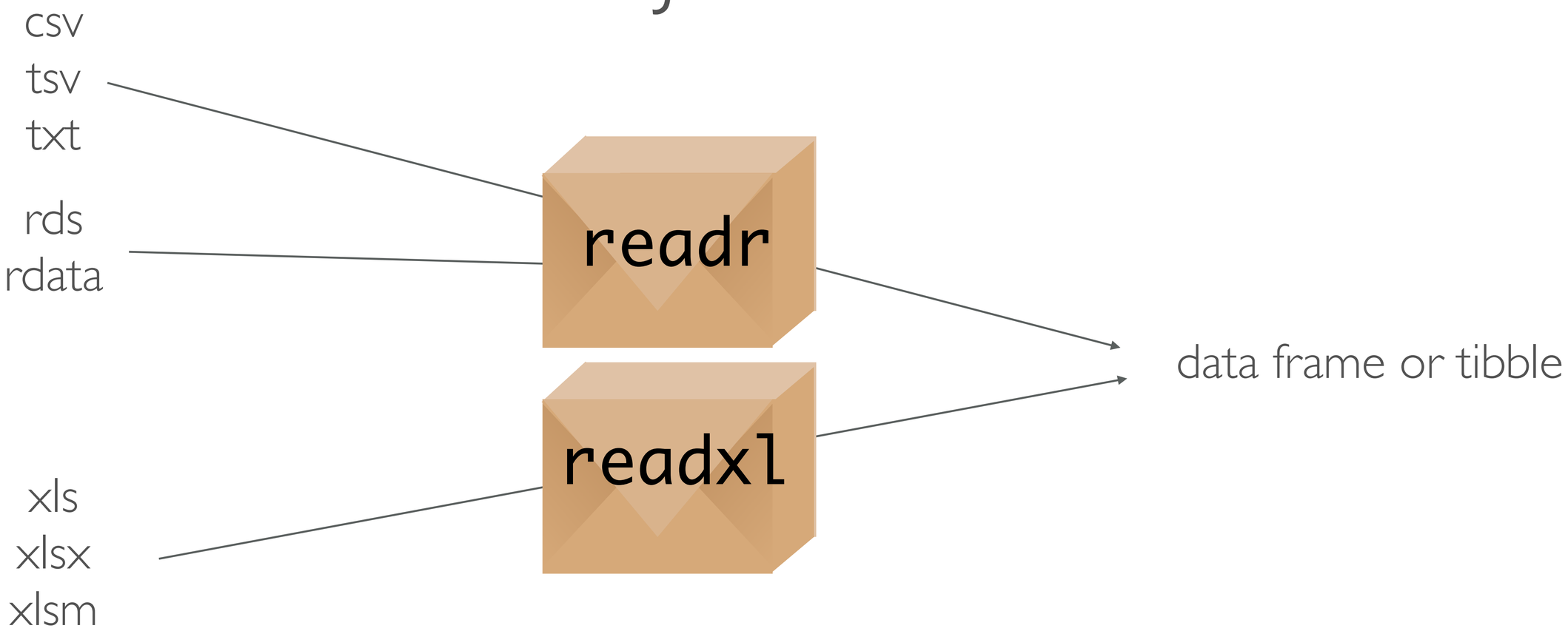
DATA IMPORTING



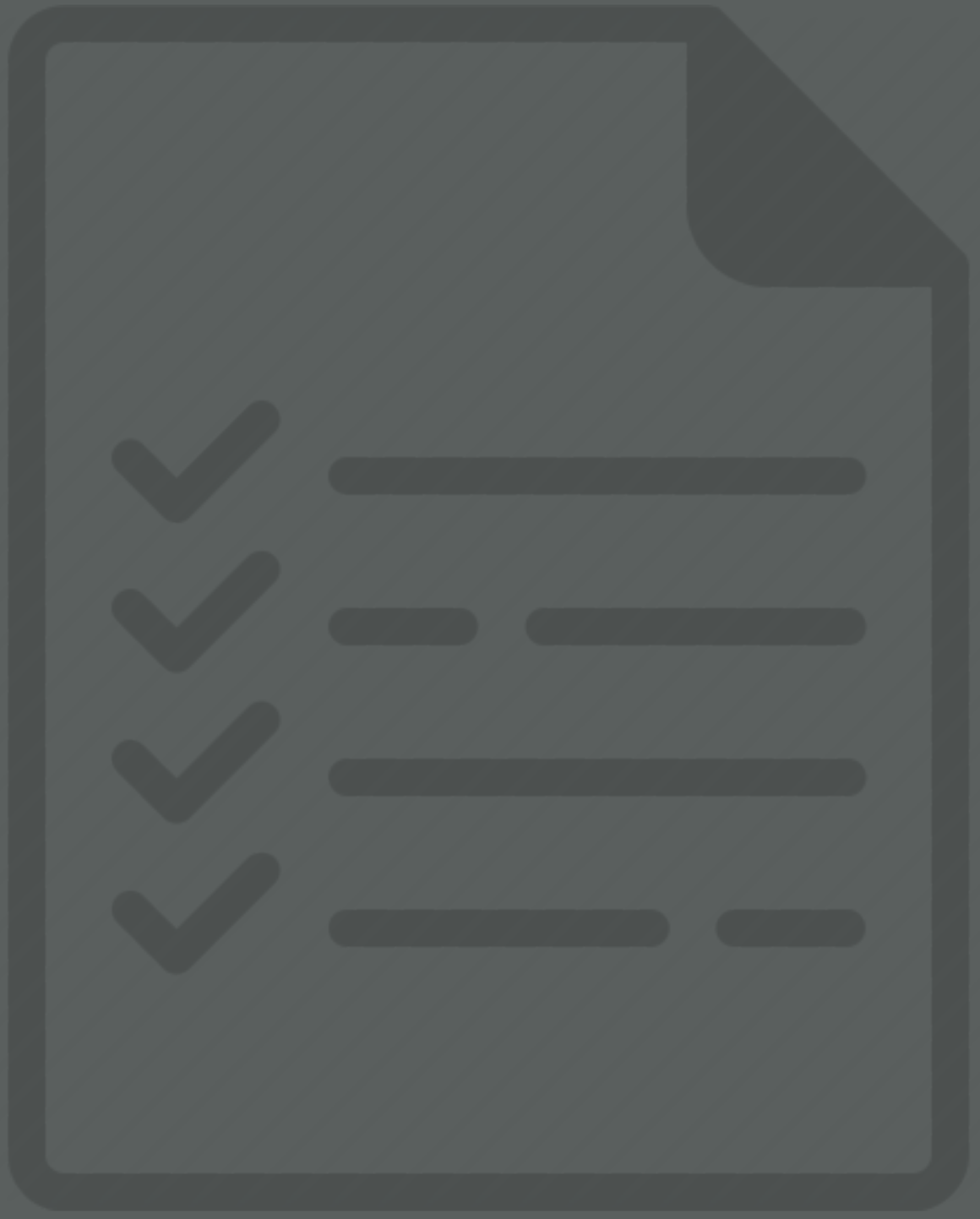
†A modified version of Hadley Wickham's analytic process

OBJECTIVE



Its important to note that base R functions for reading and writing of files; however, their syntax is inconsistent. **`readr`** *and* **`readxl`** *help to standardize these functions.*

PREREQUISITES



PREREQUISITES

- Re-start your R session
  - **Windows:** Ctrl+Shift+F10
  - **Mac:** Command+Shift+F10
- Make sure your working directory is set to the course folder
- We will be using the various "mydata" data sets that are in the data folder

PACKAGE PREREQUISITE

```
library(readxl)
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages ----------------------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

DELIMITED FILES

READING IN A FILE

- Text files are a popular way to hold and exchange tabular data
- Text file formats use **delimiters** to separate the different elements (.csv, .tsv, .txt, etc.)
- .csv most common - use `read_csv()` to read in

```
read_csv("data/mydata.csv")
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

READING IN A FILE

- Text files are a popular way to hold and exchange tabular data
- Text file formats use **delimiters** to separate the different elements (.csv, .tsv, .txt, etc.)
- .csv most common - use `read_csv()` to read in

```
read_csv("data/mydata.csv")
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

- Parsing information
- Resulting data read in

ADDITIONAL SPECIFICATIONS

We can control certain parameters when reading text files in

```
read_csv(“data/mydata.csv”)
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

```
read_csv("data/mydata.csv",
         col_types = "cc_",
         col_names = c("V1", "V2"),
         skip = 2,
         n_max = 1)
# A tibble: 1 × 2
     V1    V2
  <chr> <chr>
1    25  wine
```

ADDITIONAL SPECIFICATIONS

We can control certain parameters when reading text files in

```
read_csv(“data/mydata.csv”)
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

Change column type

```
read_csv("data/mydata.csv",
         col_types = "cc_",
         col_names = c("V1", "V2"),
         skip = 2,
         n_max = 1)
# A tibble: 1 × 2
     V1    V2
  <chr> <chr>
1    25  wine
```

ADDITIONAL SPECIFICATIONS

We can control certain parameters when reading text files in

```
read_csv(“data/mydata.csv”)
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

Change column type

Change column names

```
read_csv("data/mydata.csv",
         col_types = "cc_",
         col_names = c("V1", "V2"),
         skip = 2,
         n_max = 1)
# A tibble: 1 × 2
     V1    V2
  <chr> <chr>
1    25  wine
```

ADDITIONAL SPECIFICATIONS

We can control certain parameters when reading text files in

```
read_csv(“data/mydata.csv”)
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

Change column type

Change column names

Skip *n* number of lines

```
read_csv("data/mydata.csv",
         col_types = "cc_",
         col_names = c("V1", "V2"),
         skip = 2,
         n_max = 1)
# A tibble: 1 × 2
     V1    V2
  <chr> <chr>
1    25  wine
```

ADDITIONAL SPECIFICATIONS

We can control certain parameters when reading text files in

```
read_csv(“data/mydata.csv”)
Parsed with column specification:
cols(
  `variable 1` = col_integer(),
  `variable 2` = col_character(),
  `variable 3` = col_logical()
)
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

Change column type
Change column names
Skip *n* number of lines
Read in *n* number of lines

```
read_csv("data/mydata.csv",
         col_types = "cc_",
         col_names = c("V1", "V2"),
         skip = 2,
         n_max = 1)
# A tibble: 1 × 2
     V1    V2
  <chr> <chr>
1    25  wine
```

WRITING TO A FILE

You can write any data frame, tibble, or matrix to a .csv file

```
library(nycflights13)
write_csv(flights, “data/flights.csv”)
```

WRITING TO A FILE

You can write any data frame, tibble, or matrix to a .csv file

```
library(nycflights13)
write_csv(flights, “data/flights.csv”)
```

Name of R data object

WRITING TO A FILE

You can write any data frame, tibble, or matrix to a .csv file

```
library(nycflights13)
write_csv(flights, “data/flights.csv”)
```

Name of R data object

Path to save the file

WRITING TO A FILE

You can write any data frame, tibble, or matrix to a .csv file

```
library(nycflights13)
write_csv(flights, “data/flights.csv”)
```

Name of R data object

Path to save the file

Name of .csv file - can be anything you choose

ADDITIONAL `readr` FUNCTIONS

Function	Description
`read_delim`	specify the delimiter used in the file with `delim = ??`
`read_tsv`	uses `delim = "/t"`
`read_csv2`	uses `delim = ";"`
`read_fwf`	reads in fixed width files
`read_lines`	read lines from a file
`write_tsv`	write data to a .tsv file
`write_delim`	write data with your own delimiter
Many other functions available	

YOUR TURN!

1. *Write the* `nycflights13::planes` *data set to a .csv file*
2. *Can you figure out how to just read in the first line to see the titles?*
3. *Now read this data back in with the following caveats:*

    i) *read in the first 1000 lines*

    ii) *only read in the first 6 columns*

SOLUTION

```
# write to csv file
write_csv(planes, "data/planes.csv")

# read in column titles
read_lines("data/planes.csv", n_max = 1)

# read in first 1000 rows and first 6 columns
read_csv("data/planes.csv",
         n_max = 1000,
         col_types = "??????___")
```

EXCEL FILES

READING IN A FILE

Excel is still the spreadsheet software of choice

You need to understand both the workbook *and* the sheet that you want to read in

```
# identify the sheet you want
excel_sheets(“data/mydata.xlsx")
[1] "PICK_ME_FIRST!" "Sheet2"          "extra_header"    "functions"
[5] "date_time"      “unique_NA”
```

READING IN A FILE

Excel is still the spreadsheet software of choice

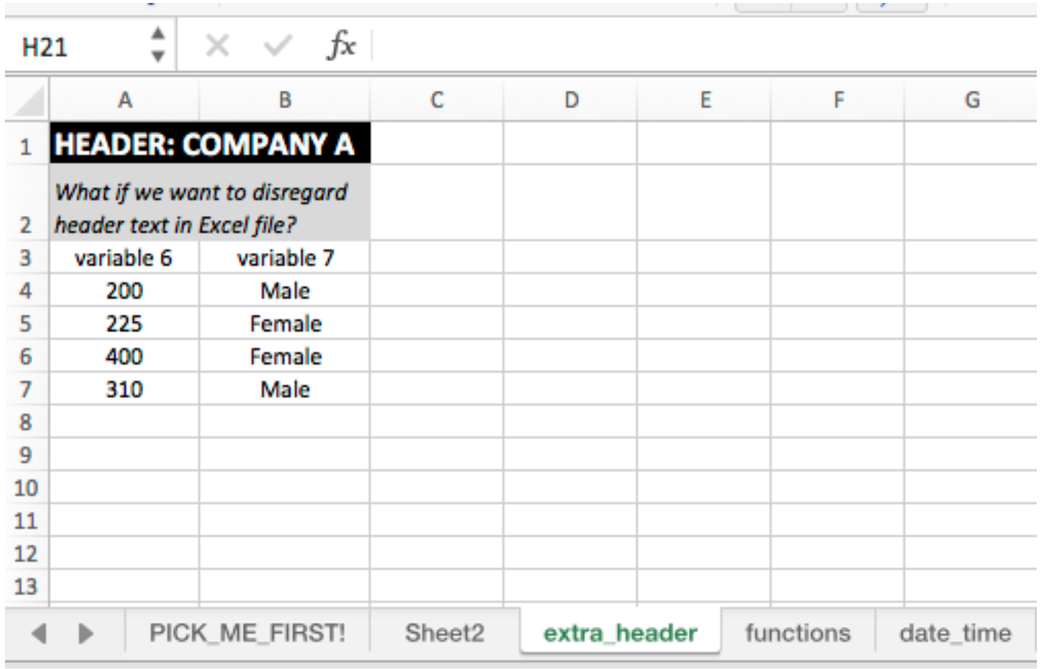
You need to understand both the workbook *and* the sheet that you want to read in

```
# identify the sheet you want
excel_sheets("data/mydata.xlsx")
[1] "PICK_ME_FIRST!" "Sheet2"         "extra_header"   "functions"
[5] "date_time"      "unique_NA"

# now read in the data
read_excel("data/mydata.xlsx", sheet = "PICK_ME_FIRST!")
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <dbl>        <chr>        <dbl>
1           10         beer            1
2           25         wine            1
3            8       cheese            0
```

ADDITIONAL SPECIFICATIONS

Similar specifications exist as we saw with `readr::read_csv`

```
read_excel("data/mydata.xlsx",
           sheet = "extra_header",
           skip = 2)
# A tibble: 4 × 2
  `variable 6` `variable 7`
         <dbl>        <chr>
1          200         Male
2          225       Female
3          400       Female
4          310         Male
```



H21

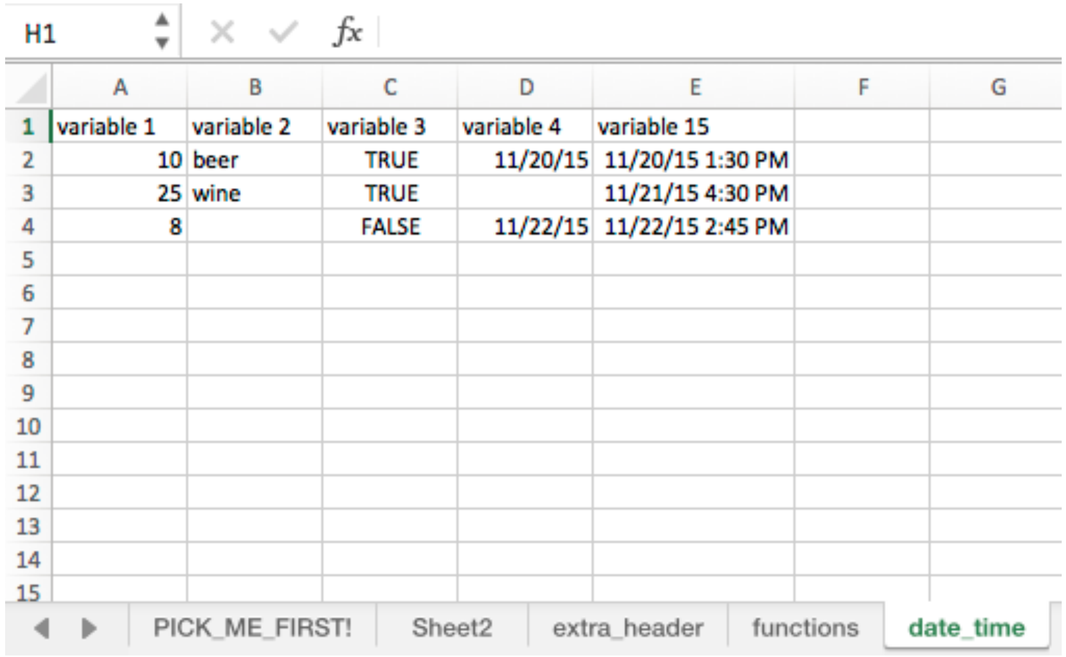
	A	B	C	D	E	F	G
1	**HEADER: COMPANY A**						
2	*What if we want to disregard header text in Excel file?*						
3	variable 6	variable 7					
4	200	Male					
5	225	Female					
6	400	Female					
7	310	Male					
8							
9							
10							
11							
12							
13							

PICK_ME_FIRST! | Sheet2 | extra_header | functions | date_time

ADDITIONAL SPECIFICATIONS

Similar specifications exist as we saw with `readr::read_csv`

```
read_excel("data/mydata.xlsx",
           sheet = "date_time",
           col_types = c("numeric", "blank",
                         "blank", "date",
                         "date"),
           col_names = paste("variable", 1:5))
# A tibble: 3 × 3
  `variable 1` `variable 4`        `variable 5`
         <dbl>       <dttm>              <dttm>
1           10   2015-11-20 2015-11-20 13:30:00
2           25          <NA> 2015-11-21 16:30:00
3            8   2015-11-22 2015-11-22 14:45:00
```



H1

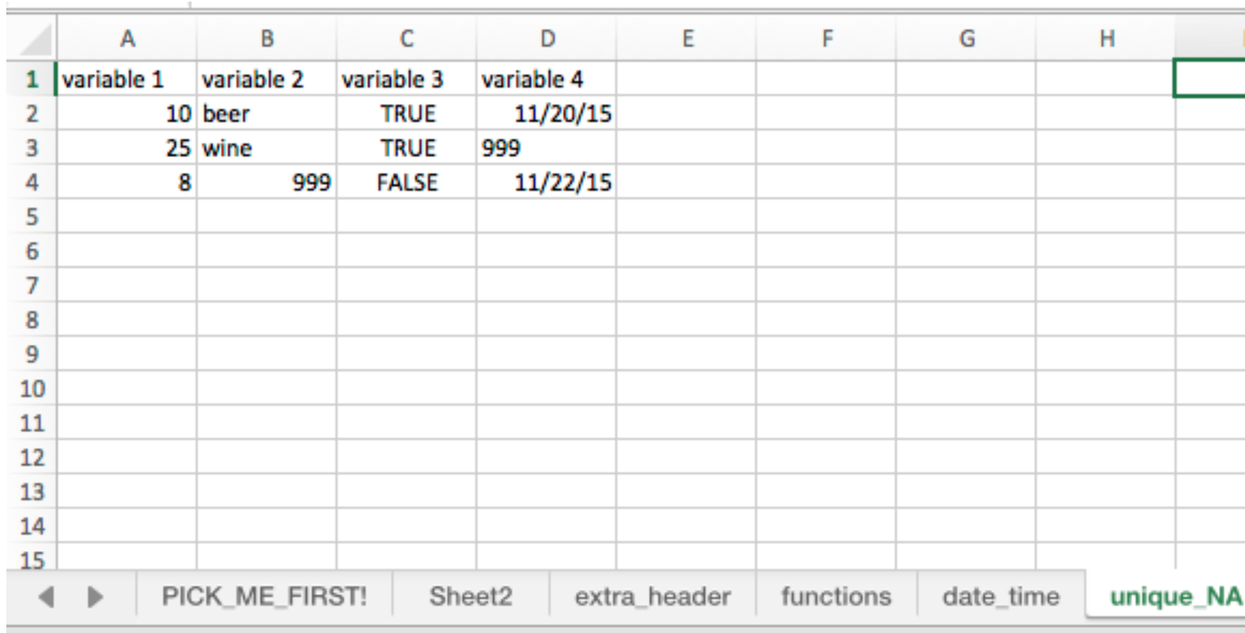
	A	B	C	D	E	F	G
1	variable 1	variable 2	variable 3	variable 4	variable 15		
2	10	beer	TRUE	11/20/15	11/20/15 1:30 PM		
3	25	wine	TRUE		11/21/15 4:30 PM		
4	8		FALSE	11/22/15	11/22/15 2:45 PM		
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							

PICK_ME_FIRST! | Sheet2 | extra_header | functions | date_time

ADDITIONAL SPECIFICATIONS

Similar specifications exist as we saw with `readr::read_csv`

```
read_excel("data/mydata.xlsx",
           sheet = "unique_NA",
           na = "999")
# A tibble: 3 × 4
  `variable 1` `variable 2` `variable 3` `variable 4`
         <dbl>        <chr>        <dbl>        <dbl>
1           10         beer            1        42328
2           25         wine            1           NA
3            8         <NA>            0        42330
```



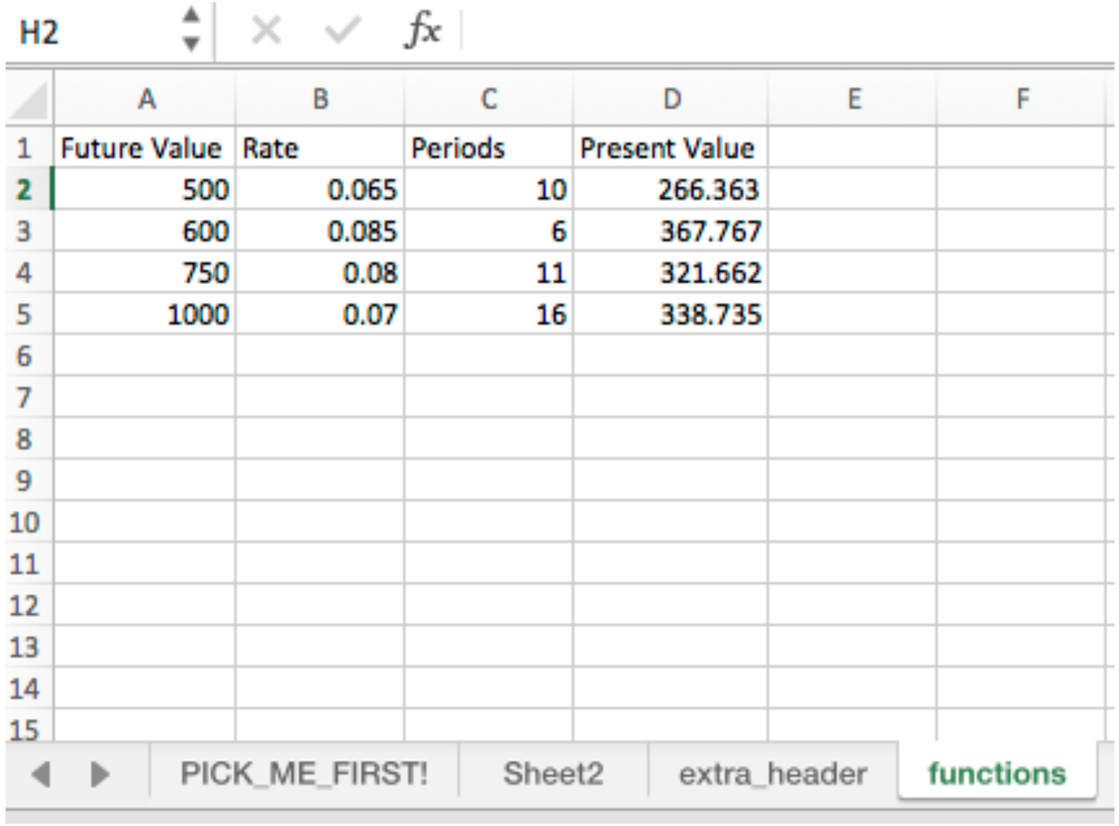
	A	B	C	D	E	F	G	H
1	variable 1	variable 2	variable 3	variable 4				
2	10	beer	TRUE	11/20/15				
3	25	wine	TRUE	999				
4	8	999	FALSE	11/22/15				
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

PICK_ME_FIRST! | Sheet2 | extra_header | functions | date_time | unique_NA

ADDITIONAL SPECIFICATIONS

The **xlsx** package provides a lot of special functionality as well `xlsx::read.xlsx`

```
xlsx::read.xlsx("data/mydata.xlsx",
                sheetName = "functions",
                keepFormulas = TRUE)
  Future.Value  Rate Periods Present.Value
1          500 0.065      10  A2/(1+B2)^C2
2          600 0.085       6  A3/(1+B3)^C3
3          750 0.080      11  A4/(1+B4)^C4
4         1000 0.070      16  A5/(1+B5)^C5
```



WRITING TO A FILE

- There are several packages that allow you to write and format Excel files
- Easiest function to use is `readr::write_excel_csv`
  - includes a UTF-8 Byte order mark which indicates to Excel the csv is UTF-8 encoded

```
library(nycflights13)
write_excel_csv(flights, “data/flights.csv”)
```

YOUR TURN!

1. *What spreadsheets are in the "PEW Middle Class Data.xlsx" file?*

2. *Read in the "3. Median HH income, metro" spreadsheet.*

SOLUTION

```
# identify spreadsheets
excel_sheets("data/PEW Middle Class Data.xlsx")

# read in “3. Median HH income, metro” spreadsheet
read_excel("data/PEW Middle Class Data.xlsx",
           sheet = "3. Median HH income, metro",
           skip =  5)
```

R FILES

READING IN A FILE

- Saving and sharing data as R objects can be more efficient than converting to text or Excel files
- Two primary ways R data can be saved (`.rds, .RData`)

READING IN A FILE

- Saving and sharing data as R objects can be more efficient than converting to text or Excel files
- Two primary ways R data can be saved (`.rds, .RData`)

```
read_rds("data/mydata.rds")
# A tibble: 3 × 3
  `variable 1` `variable 2` `variable 3`
         <int>        <chr>        <lgl>
1           10         beer         TRUE
2           25         wine         TRUE
3            8       cheese        FALSE
```

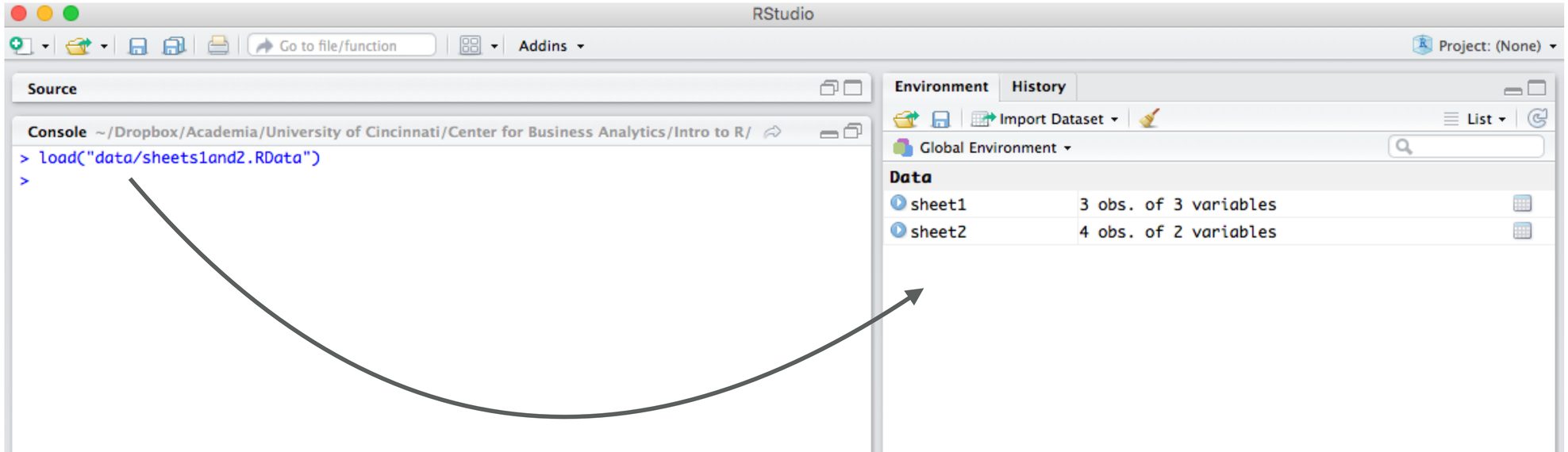
- `.rds`: single R object

READING IN A FILE

- Saving and sharing data as R objects can be more efficient than converting to text or Excel files
- Two primary ways R data can be saved (`.rds, .RData`)

```
load("data/sheets1and2.RData")
```

- `.RData`: multiple R objects



WRITING TO A FILE

We can write to ..`rds` and .`RData` files with the following:

```
# write data to an .rds file
write_rds(mydata, “data/mydata.rds")

# write multiple data objects to a file
save(myvector, mymatrix, mylist, mydata, file = “data/mystuff.RData”)

# write all objects to a file
save.image(file = “data/mystuff.RData”)
```

WRITING TO A FILE

We can write to `.rds` and `.RData` files with the following:

```
# write data to an .rds file
write_rds(mydata, “data/mydata.rds")

# write multiple data objects to a file
save(myvector, mymatrix, mylist, mydata, file = “data/mystuff.RData”)

# write all objects to a file
save.image(file = “data/mystuff.RData”)
```

- You can list as many objects as you’d like

WRITING TO A FILE

We can write to **..rds** and **.RData** files with the following:

```
# write data to an .rds file
write_rds(mydata, “data/mydata.rds")

# write multiple data objects to a file
save(myvector, mymatrix, mylist, mydata, file = “data/mystuff.RData”)

# write all objects to a file
save.image(file = “data/mystuff.RData”)
```

- Short cut to save everything in your global environment

YOUR TURN!

1. *Load any 3 worksheets from the* **`mydata.xlsx`** *workbook and save as 3 separate tibbles*

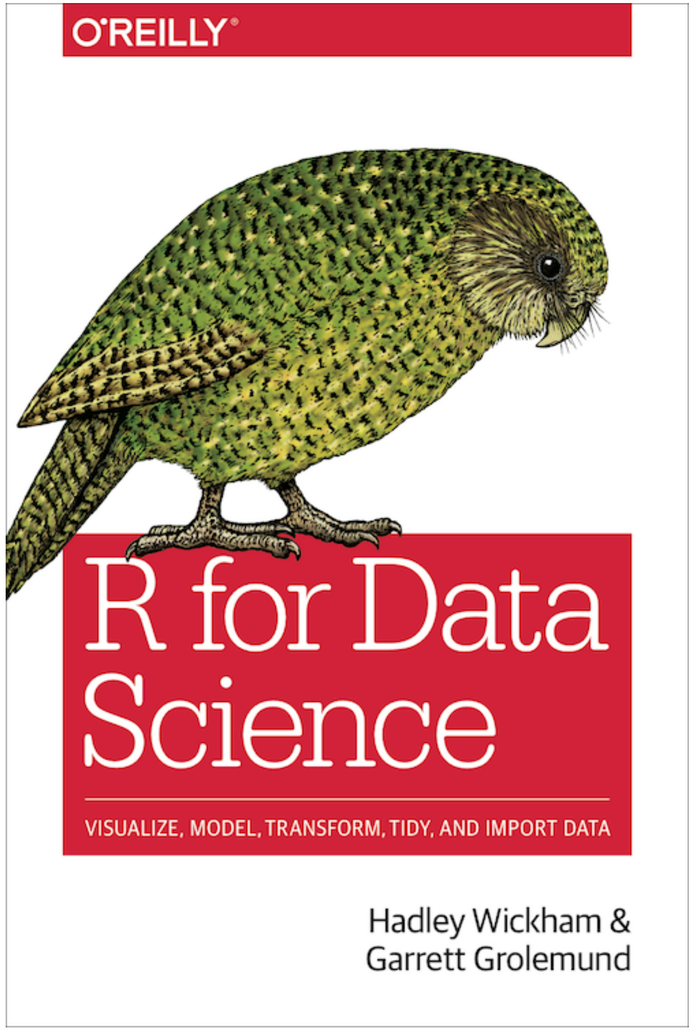
2. *Save all three tibbles together in a* **`.RData`** *file*

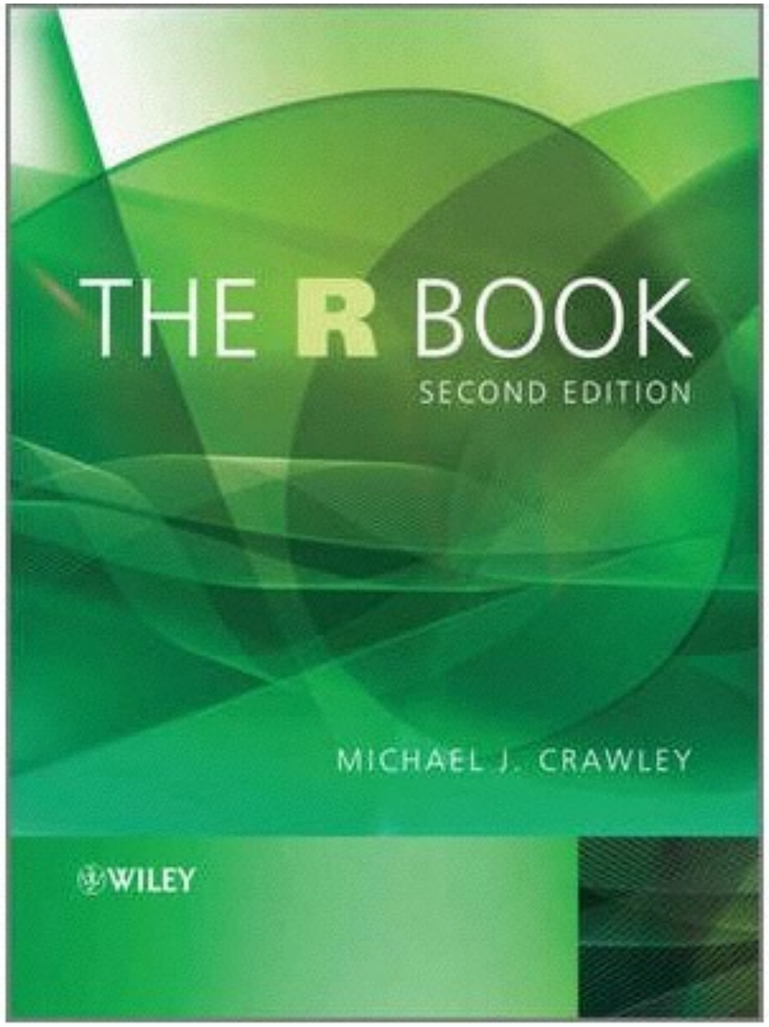
SOLUTION

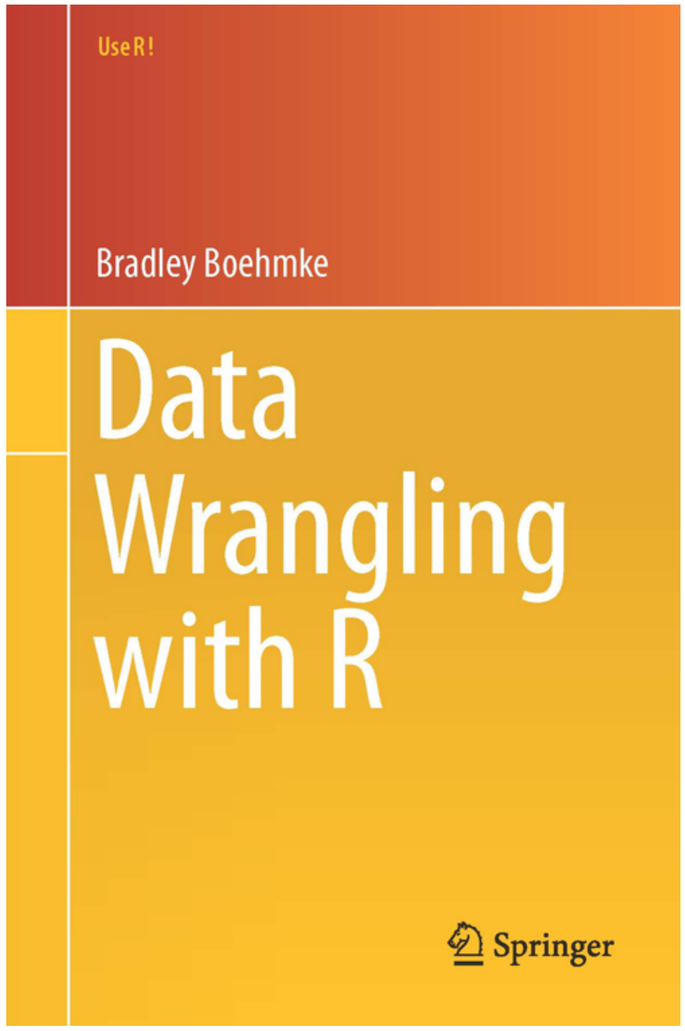
```
# load any three worksheets
sheet1 <- read_excel("data/mydata.xlsx", sheet = 1)
sheet2 <- read_excel("data/mydata.xlsx", sheet = 2)
sheet3 <- read_excel("data/mydata.xlsx", sheet = 3)

# save as all three in a .RData file
save(sheet1, sheet2, sheet3, file = "data/sheets123.RData")
```

SO LITTLE TIME!

LEARN MORE


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham &
Garrett Grolemund


THE R BOOK
SECOND EDITION
MICHAEL J. CRAWLEY
WILEY


Use R!
Bradley Boehmke
Data Wrangling with R
Springer

WHAT TO REMEMBER

FUNCTIONS TO REMEMBER

Operator/Function	Description
`read_**`	read csv, txt, fwf, xlsx, xls, etc. files
`write_**`	write to files
`save, save.image`	saves external representation of R objects

Its important to note that base R functions exist to do many reading and writing of files; however, their syntax is inconsistent. **`readr`** *and* **`readxl`** *help to standardize these functions.*