PSYC 259:
Principles of Data Science

Week 2: Data Workflow

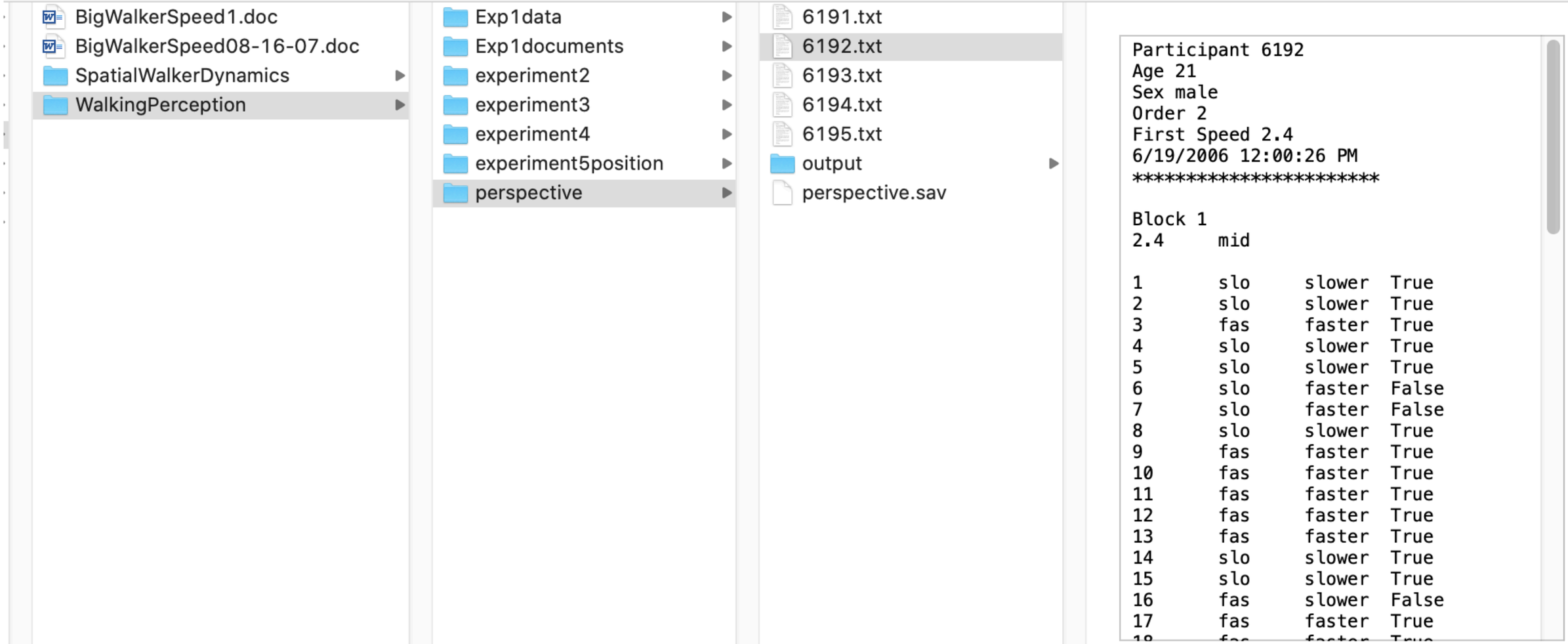
Today

1. Workflow examples
2. “Minimizing mistakes” article discussion
3. Project structure principles
4. File organization + Import tutorial
5. How to get help
6. Homework 1

Workflow examples

Example 1
It was the year 2006…

- A young lab manager was tasked with testing how people walking on a treadmill perceived their own walking speed compared to life-sized avatars walking alongside them on a projection screen
- Software was written
- Results were analyzed
- A poster was presented
- The files sat dormant in storage for 15 years, never seeing the light of day….until now!


BigWalkerSpeed1.doc
BigWalkerSpeed08-16-07.doc
SpatialWalkerDynamics
WalkingPerception
Exp1data
Exp1documents
experiment2
experiment3
experiment4
experiment5position
perspective
6191.txt
6192.txt
6193.txt
6194.txt
6195.txt
output
perspective.sav
Participant 6192
Age 21
Sex male
Order 2
First Speed 2.4
6/19/2006 12:00:26 PM
****
Block 1
2.4 mid
1 slo slower True
2 slo slower True
3 fas faster True
4 slo slower True
5 slo slower True
6 slo faster False
7 slo faster False
8 slo slower True
9 fas faster True
10 fas faster True
11 fas faster True
12 fas faster True
13 fas faster True
14 slo slower True
15 slo slower True
16 fas slower False
17 fas faster True
6192.txt
Plain Text Document - 3 KB
Information
Created Monday, June 19, 2006 at 10:10 AM
Modified Monday, June 19, 2006 at 10:10 AM


```
Block 1
2.4     mid

1       slo     slower  True
2       slo     slower  True
3       fas     faster  True
4       slo     slower  True
5       slo     slower  True
6       slo     faster  False
7       slo     faster  False
8       slo     slower  True
9       fas     faster  True
10      fas     faster  True
11      fas     faster  True
12      fas     faster  True
13      fas     faster  True
14      slo     slower  True
15      slo     slower  True
16      fas     slower  False
17      fas     faster  True
18      fas     faster  True
19      slo     slower  True
20      fas     faster  True

Correct:        17
Slower:         9

Block 2
2.7     near

1       slo     slower  True
2       slo     slower  True
3       slo     slower  True
4       slo     slower  True
```

id	age	sex	order	firstblock	near24	mid24	far24	near27	mid27	far27	near24s	mid24s	far24s	near27s	mid27s	far27s
6191	25	1	1	1	16	15	16	18	17	15	10	7	12	10	11	13
6192	21	1	2	1	18	17	18	18	19	17	8	9	12	8	11	9
6193	21	1	3	1	17	12	18	18	17	19	7	6	10	8	9	9
6194	22	1	1	2	17	17	17	20	19	17	7	7	11	10	9	11
6195	19	2	2	2	15	17	14	17	17	18	7	9	16	7	7	12
6211	19	1	3	2	20	16	13	17	16	18	10	14	13	13	8	10
6221	19	2	1	1	16	12	18	15	16	12	8	6	12	7	10	10
6222	21	2	2	1	20	13	18	19	18	18	10	13	12	9	8	12
6223	18	2	3	1	17	15	15	18	17	17	7	7	15	8	9	9
6224	18	2	1	2	19	15	17	16	19	16	11	9	13	10	11	8
6231	22	2	2	2	15	15	16	20	16	19	9	9	10	10	8	11
6232	21	1	3	2	14	14	11	14	16	15	4	8	7	6	8	13

id	age	sex	order	firstblock	near24	mid24	far24	near27	mid27	far27	near24s	mid24s	far24s	near27s	mid27s	far27s
6191	25	1	1	1	16	[illegible]	16	18	17	15	10	7	12	10	11	13
6192	21	1	2	1	18	17	18	18	19	17	8	9	12	8	11	9
6193	21	1	3	1	17	[illegible]	18	18	17	19	7	6	10	8	9	9
6194	22	1	1	2	17	17	17	20	19	17	7	7	11	10	9	11
6195	19	2	2	2	15	17	14	17	17	18	7	9	16	7	7	12
6211	19	1	3	2	20	16	13	17	16	18	10	14	13	13	8	10
6221	19	2	1	1	16	12	18	15	16	12	8	6	12	7	10	10
6222	21	2	2	1	20	13	18	19	18	18	10	13	12	9	8	12
6223	18	2	3	1	17	15	15	18	17	17	7	7	15	8	9	9
6224	18	2	1	2	19	15	17	16	19	16	11	9	13	10	11	8
6231	22	2	2	2	15	15	16	20	16	19	9	9	10	10	8	11
6232	21	1	3	2	14	14	11	14	16	15	4	8	7	6	8	13

6192.txt
6193.txt
6194.txt
6195.txt
6211.txt
6221.txt
6222.txt
6223.txt
6224.txt
6231.txt
6232.txt

```
Block 1
2.4     mid

1       slo     slower  True
2       slo     slower  True
3       fas     faster  True
4       slo     slower  True
5       slo     slower  True
6       slo     faster  False
7       slo     faster  False
8       slo     slower  True
9       fas     faster  True
10      fas     faster  True
11      fas     faster  True
12      fas     faster  True
13      fas     faster  True
14      slo     slower  True
15      slo     slower  True
16      fas     slower  False
17      fas     faster  True
18      fas     faster  True
19      slo     slower  True
20      fas     faster  True

Correct:        17
Slower:         9
```

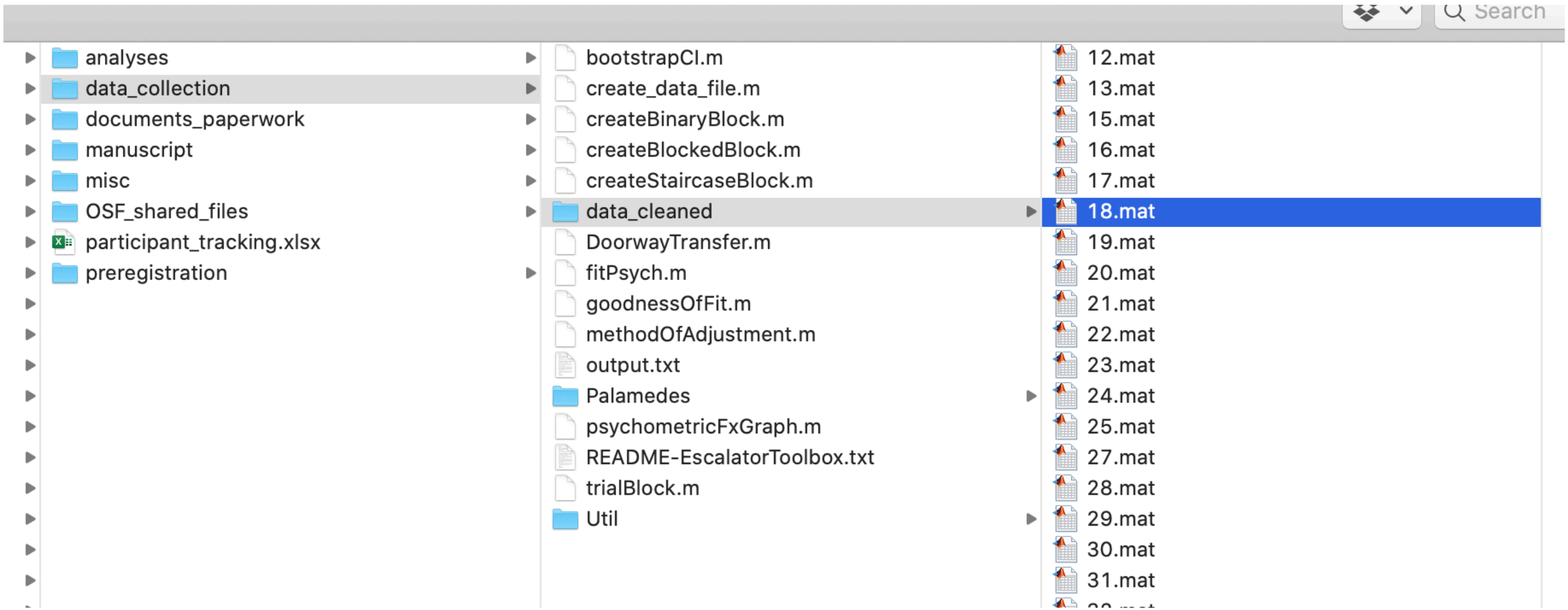
per27n	per27m	per27f	mean24	mean27	data	plspeed	plcond
90.00	85.00	75.00	78.33	83.33	80.00	2.40	near
90.00	95.00	85.00	88.33	90.00	90.00	2.40	near
90.00	85.00	95.00	78.33	90.00	85.00	2.40	near
100.00	95.00	85.00	85.00	93.33	85.00	2.40	near
85.00	85.00	90.00	76.67	86.67	75.00	2.40	near
85.00	80.00	90.00	81.67	85.00	100.00	2.40	near
75.00	80.00	60.00	76.67	71.67	80.00	2.40	near
95.00	90.00	90.00	85.00	91.67	100.00	2.40	near
90.00	85.00	85.00	78.33	86.67	85.00	2.40	near
80.00	95.00	80.00	85.00	85.00	95.00	2.40	near
100.00	80.00	95.00	76.67	91.67	75.00	2.40	near
70.00	80.00	75.00	65.00	75.00	70.00	2.40	near
.	.	.	.	.	75.00	2.40	mid
.	.	.	.	.	85.00	2.40	mid
.	.	.	.	.	60.00	2.40	mid
.	.	.	.	.	85.00	2.40	mid
.	.	.	.	.	85.00	2.40	mid
.	.	.	.	.	80.00	2.40	mid
.	.	.	.	.	60.00	2.40	mid
.	.	.	.	.	65.00	2.40	mid
.	.	.	.	.	75.00	2.40	mid
.	.	.	.	.	75.00	2.40	mid
.	.	.	.	.	75.00	2.40	mid
.	.	.	.	.	70.00	2.40	mid
.	.	.	.	.	80.00	2.40	far
.	.	.	.	.	90.00	2.40	far
.	.	.	.	.	90.00	2.40	far
.	.	.	.	.	85.00	2.40	far

List of crimes against data science

- Files: unorganized, inconsistently named, used unsupported proprietary formats, no version control, cluttered workspace with intermediary/redundant file
- Automation (lack of)
  - Used copy/paste to transfer data rather than automating input
  - Copied data within SPSS from wide to long
  - Made graphs/analyses from drop-down menus
- No metadata, no documentation

Example 2
The year was 2019…

- An assistant professor had a team of 5 undergrad RAs run participants in a study on perception of fitting through doorways
- The RAs entered the participant and condition info into a Matlab program, which ran the trials and wrote output data for each participant
- The RAs logged metadata into a “participant_tracking.xls” document after running each participant, leaving notes about problems with each participant

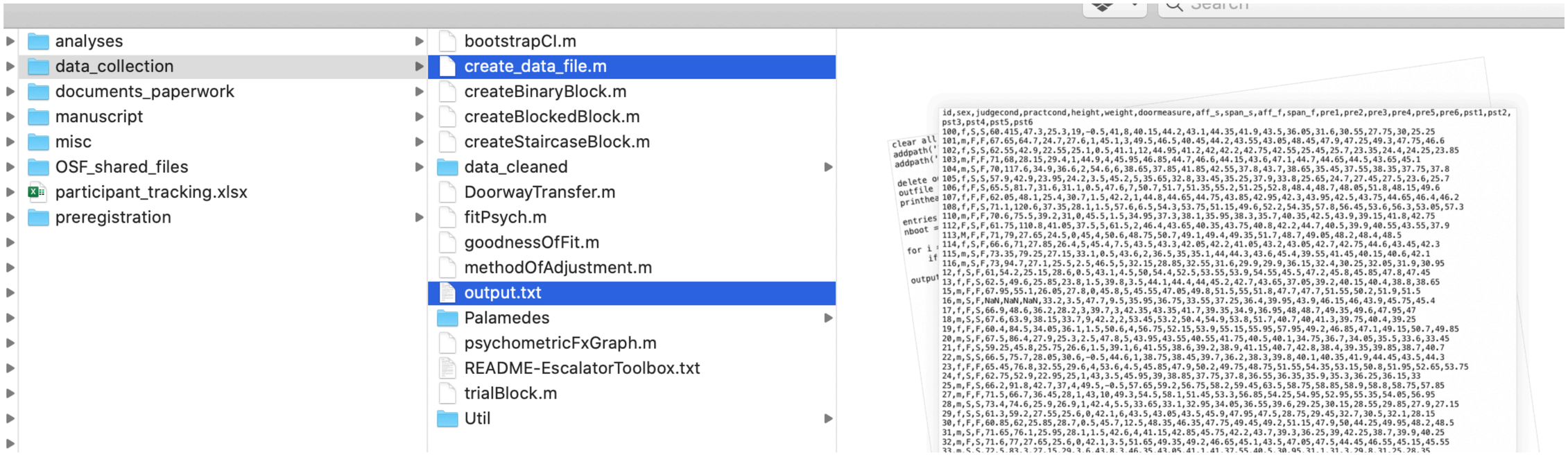

Search
analyses
data_collection
documents_paperwork
manuscript
misc
OSF_shared_files
participant_tracking.xlsx
preregistration
bootstrapCI.m
create_data_file.m
createBinaryBlock.m
createBlockedBlock.m
createStaircaseBlock.m
data_cleaned
DoorwayTransfer.m
fitPsych.m
goodnessOfFit.m
methodOfAdjustment.m
output.txt
Palamedes
psychometricFxGraph.m
README-EscalatorToolbox.txt
trialBlock.m
Util
12.mat
13.mat
15.mat
16.mat
17.mat
18.mat
19.mat
20.mat
21.mat
22.mat
23.mat
24.mat
25.mat
27.mat
28.mat
29.mat
30.mat
31.mat

Command Window

```
>> sex

sex =

  1×1 cell array

    {'m'}

>> id

id =

  1×1 cell array

    {'18'}

>> block1

block1 =

  struct with fields:

          mode: 'staircase'
    start_unit: 32
     down_step: 4
       up_step: 3
    num_trials: 16
   use_for_fit: 1

>> output_aff_f

output_aff_f =

  struct with fields:

              id: '18'
       condition: 'aff_f'
       threshold: 40.8259
           slope: 1.0576
      trial_unit: [1×30 double]
  trial_unit_fit: [1×30 double]
      trial_resp: [1×30 double]
  trial_resp_fit: [1×30 double]
      trial_mode: [1×30 double]
  trial_subblock: [1×30 double]
     stim_levels: [1×107 double]
    elapsed_time: 618.8013
```

Workspace

Name ▲	Value
block1	*1x1 struct*
block2	*1x1 struct*
block_fitting	*2x1 cell*
block_squeezing	*2x1 cell*
door_measure1	*1x1 cell*
door_measure2	*1x1 cell*
height1	*1x1 cell*
height2	*1x1 cell*
id	*1x1 cell*
judgment_cond	*1x1 cell*
measurement_...	*1x1 cell*
out_dir	'C:\Users\labuse
output_aff_f	*1x1 struct*
output_aff_s	*1x1 struct*
output_pre_moa	*1x1 struct*
output_pst_moa	*1x1 struct*
practice_cond	*1x1 cell*
save_figs	1
save_file	'C:\Users\labuse
sex	*1x1 cell*
stim_levels	*1x107 double*
weight1	*1x1 cell*
weight2	*1x1 cell*



- “create_data_file.m” loads each individual file in “data_cleaned” and creates rows of data to write to “output.txt”
- “output.txt” gets copied to “analyses” for R analyses

analyses.R

Source on Save | Run | Source

```
30
31  # READ DATA ----
32
33  #Read data and set factor levels
34  ds <- read_csv("output.txt")
35  ds$id <- factor(ds$id)
36  ds$practice <- factor(ds$practcond, levels =c("S","F"),labels = c("Squeezing","Fitting"))
37  ds$judge <- factor(ds$judgecond, levels =c("S","F"),labels = c("Squeezing","Fitting"))
38  ds$match <- ds$practcond == ds$judgecond
39  ds$match <- factor(ds$match, levels = c(T, F), labels = c("Congruent","Incongruent"))
40
41  # Create error terms ====
42  ds$judge_ref[ds$judge == "Fitting"] <- ds$aff_f[ds$judge == "Fitting"]
43  ds$judge_ref[ds$judge == "Squeezing"] <- ds$aff_s[ds$judge == "Squeezing"]
44
45  ds$pre_err1 <- ds$pre1 - ds$judge_ref
46  ds$pre_err2 <- ds$pre2 - ds$judge_ref
47  ds$pre_err3 <- ds$pre3 - ds$judge_ref
48  ds$pre_err4 <- ds$pre4 - ds$judge_ref
49  ds$pre_err5 <- ds$pre5 - ds$judge_ref
50  ds$pre_err6 <- ds$pre6 - ds$judge_ref
51  ds$pst_err1 <- ds$pst1 - ds$judge_ref
52  ds$pst_err2 <- ds$pst2 - ds$judge_ref
53  ds$pst_err3 <- ds$pst3 - ds$judge_ref
54  ds$pst_err4 <- ds$pst4 - ds$judge_ref
```

40:1 | READ DATA | R Script

Console | Terminal | Jobs

~/Dropbox/past_projects/study_doorwaytransfer/analyses/

```
── Column specification ────────────────────────────────────────────────────────
cols(
  .default = col_double(),
```

Environment | History | Connections

Import Dataset | List

Global Environment

Data	
ds	100 obs. of 26 variables
my_theme	List of 11
pd	Environment
Values	
cbp1	chr [1:8] "#999999" "#E69F00" "#56B4E9" "#009E73" "…

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

Home > Dropbox > past_projects > study_doorwaytransfer > analyses

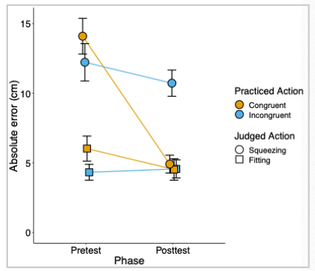
Name	Size	Modified
..		
.RData	136.7 KB	Dec 17, 2019, 3:46 PM
.Rhistory	24.1 KB	Dec 17, 2019, 3:46 PM
age.xlsx	26.1 KB	Jul 23, 2019, 9:46 AM
analyses.R	17.4 KB	Dec 17, 2019, 3:46 PM
analyses.Rproj	204 B	Jan 5, 2021, 1:16 PM
correlations.pdf	9.9 KB	Aug 7, 2019, 3:40 PM
error.pdf	5.4 KB	Aug 7, 2019, 3:42 PM
output.txt	10.4 KB	Jul 17, 2019, 12:22 PM
summarySE.R	1.7 KB	Aug 7, 2018, 4:16 PM

- R reads in output.txt, assigns factors, and calculates DVs
- Creates the analyses and figures for the paper

doorway_transfer_LaTeX_final

Search

apparatus-compressed.pdf
apparatus.pdf
correlations.pdf
DesignFigure.pdf
diff.aux
diff.bbl
diff.blg
diff.log
diff.out
diff.pdf
diff.synctex.gz
diff.tex
doorway_transfer.aux
doorway_transfer.bbl
doorway_transfer.blg
doorway_transfer.log
doorway_transfer.out
doorway_transfer.pdf
doorway_transfer.synctex.gz
doorway_transfer.tex
error.pdf
ExampleTrials.mp4
master.bib
sage_latex_template_4
SageH.bst
sagej.cls
sagej.log
SageV.bst
submission1
submission2
submission3-publication



error.pdf

PDF document - 5 KB

Information Show More

Created Aug 7, 2019 at 3:42 PM
Modified Aug 7, 2019 at 3:42 PM

Tags

Add Tags...

Markup More...

doorway_transfer.tex

Typeset LaTeX Macros Tags Labels Templates

replicability \citep{AsendorpfConner2013, SimmonsNelson2011}, the current study's procedure and analysis plan were pre-registered before data collection began. The pre-registration document was entered on AsPredicted.com ({\tt https://aspredicted.org/s58hb.pdf}). The sample size was determined in advance based on a power analysis: Past work comparing pretest/posttest affordance judgment errors across multiple between-subjects conditions found a large interaction effect size, $\eta^{2} = .152$ \citep{Recal}. However, the transfer effect in the current study might be smaller, so the sample size was conserv
effect size ($\eta^{2} = .0
resulted from a technical
Although postural sway n
accelerometer data were
measurements to trials a
the pre-registration we re
``mismatching"; here we

104
105 \subsection{Participants a
106
107 The final sample included
\textit{SD} = 1.19, 54 fem
conditions in a 2 Judged
(congruent, incongruent)
\textit{squeezing-congrue
male), \textit{squeezing-i
11 male), \textit{fitting-co
male), and \textit{fitting-ir
11 male). Judgment phas
subjects factor. Two addit
replaced for failure to foll
repeated requests from t

doorway_transfer.pdf

Scale 138 Page 12 of 22

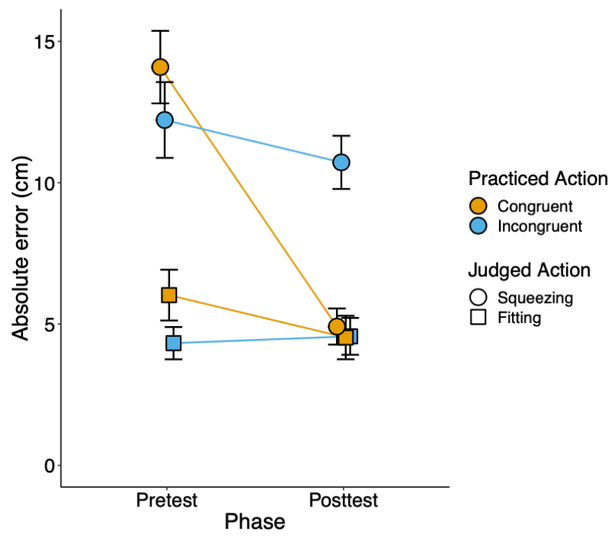


Figure 3. Absolute judgment errors by phase (x-axis: pretest vs posttest), judged action (circles: squeezing; squares: fitting), and practiced action (orange symbols: congruent practice; blue symbols: incongruent practice).

with the pre-registration plan, participants with outlier data (based on inter-quartile range) were excluded and not replaced (3 participants in the FC condition and 1 participant in the FI condition). The influence of outliers on the results is discussed below.

Overall model Table 1 shows results for the overall model predicting absolute judgment errors from judged action, practiced action, and judgment phase (and their interactions) as fixed effects and random intercepts by participant (random slope models failed to converge). As seen in Figure 3, a significant main effect of phase indicates that participants were more accurate in posttest compared with pretest, and a significant main effect of judged action reveals that participants were more accurate when judging fitting compared with squeezing. However, these effects were moderated by significant two-way (judged action×phase, practiced action×phase) and three-way (judged action×practiced action×phase) interactions. Including or excluding outliers did not affect the significance of any of the effects in the

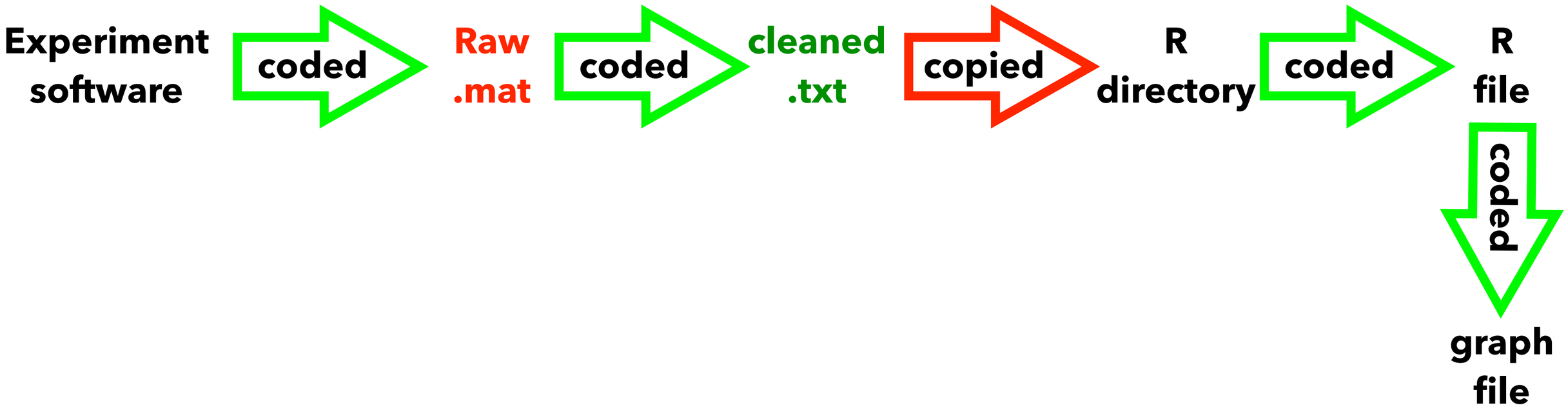
What was good?

- Files: Organized in directories, consistently named, mostly flat text files
- Version control: minimal intermediaries, used GitHub
- Automation: Data collection was automated, cleaning raw data was automated, analyses/figures were automated
- Automation: Minimal copy/paste, minimal user typing
- Metadata saved about each participant to diagnose problems, understand exclusion

What should be better?

- Using matlab scripts/data files makes project harder to share, less future-proof; no way to archive the coding environment
- Copy/pasting data/figures between folders, analyses to paper leave plenty of room for error
- Clear set of processes to regenerate from raw data, but not documented well (requires user to run scripts in correct order which only I know)
- No formal checks of data quality
- Lots of repetitive, single-use Matlab and R code

Comparing the workflows



Minimizing mistakes

article discussion

What principles should guide project structure?

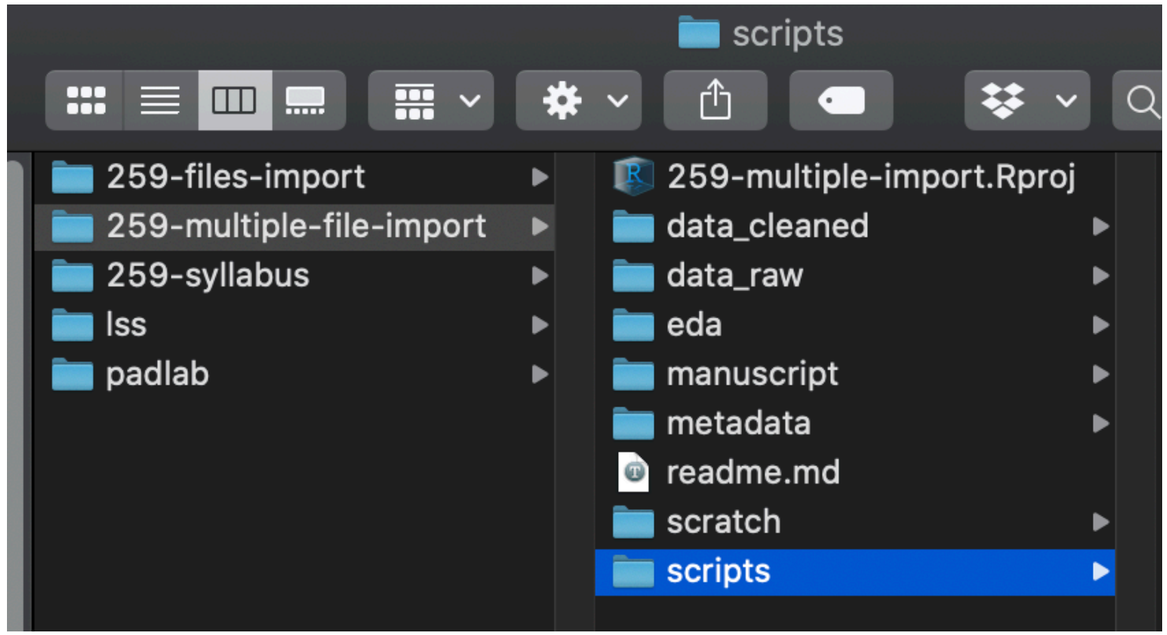
Two principles of project workflow

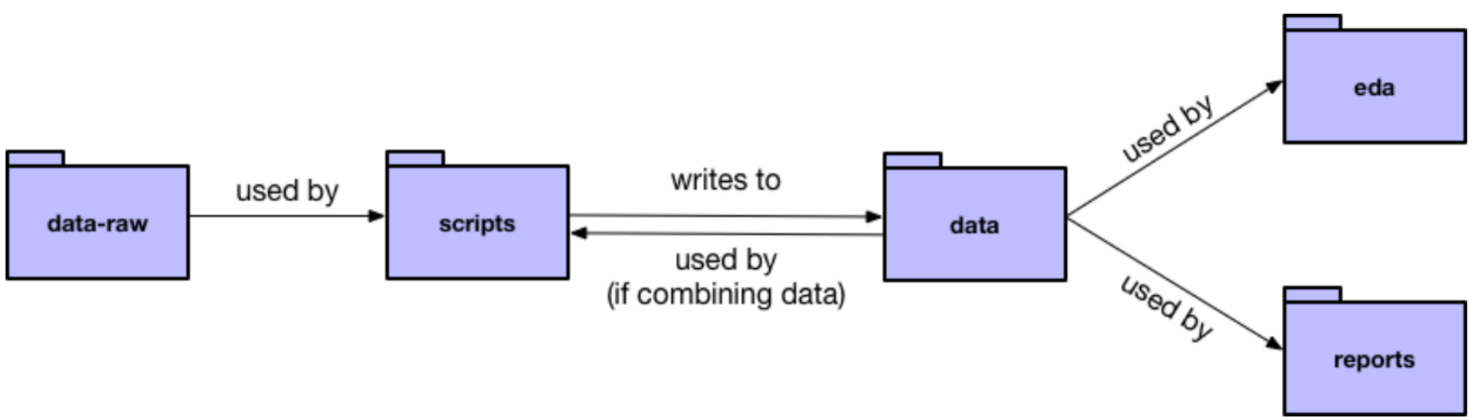
1. Folder organization creates rules and defines a workflow; establishes a location from which to build relative file paths
2. Version control tracks file history without duplication/clutter; allows for collaboration/derivation/experimentation

What principles should guide project structure?

#1 File/folder organization

Folders to organize similar file types (w/in a root project folder)





R Studio Project

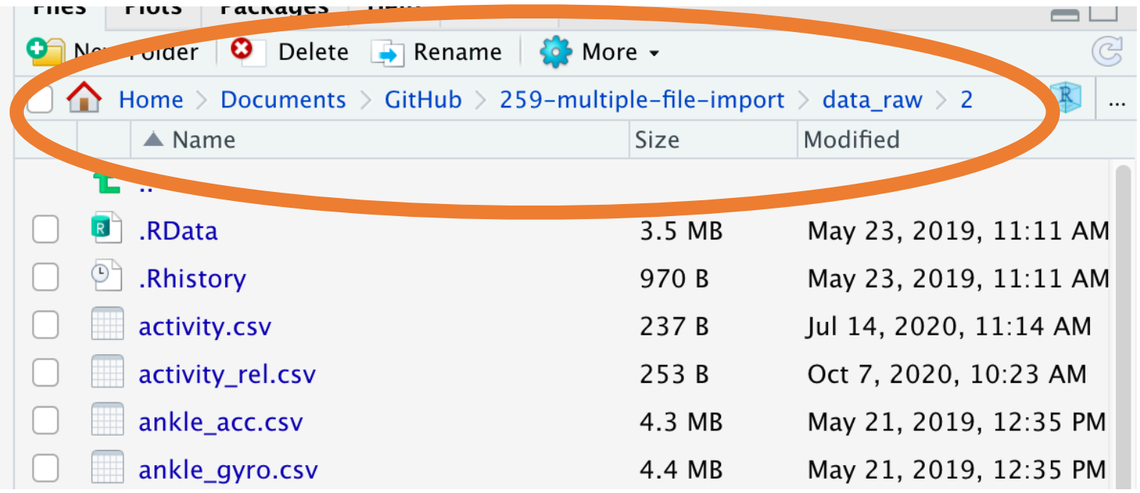
- Essentially, your project folder with some files that keep tabs on your command history
  - Can create a project from an existing folder
  - Open the RProj file to open the project
  - Can save your workspace, but follow R4DS guidelines and decline that feature
- What is real?
  - Not real: objects/data frames in your workspace
  - Real: data files and the scripts used to work with them
  - Intermediaries incur a cost of maintenance

Accessing files from your working directory

- Your RStudio Project folder is your default *working directory*
  - working directory: where R can find files
  - getwd() probably gives you something ugly like `/Users/johnfranchak/Documents/GitHub/259-multiple-file-import`
- *Absolute file paths* like that should be avoided at all costs!
  - Absolute file paths don't transfer between computers with different user names or different operating systems
  - Bad for reproducibility, extensibility, and sharing
- But don't you need them to get into all of those directories you just told me I need to use?

Relative files paths 💪💪💪

- Just tell R what to look for relative to your project (working) directory!



- *here* package detects the project directory and composes filenames from/to any folder
  - `here("folder1","folder2","filename"))`

Other considerations

- Not everything can be automated, but you can try to limit human data entry to a single master table
  - Track notes about sessions, inclusion/exclusion info
    - Keep as part of project metadata, and use it to direct your scripts to pull the ‘right’ data
- Not every project can be contained in a local directory on a single computer (or on github)
  - Large datasets might need other solutions
  - “data_raw” might need to be “data_slightly_cooked”

Tutorial #1 - Data Import/Export

- R Studio Cloud/Github repos
  - 259-files-import
  - 259-multiple-file-import

Assumptions

- Loading packages
- Running chunks of code from a script
- Basic syntax for assigning variables <-
- Objects/data get stored in the environment
- What a dataframe/tibble is

readr package (loads via “tidyverse”)

- read_csv(), read_tsv(), read_delim()
  - Variants to read flat file data w/ different delimiters
  - Reads to a tibble rather than a data frame
  - Doesn’t automatically convert strings to factors
  - Faster than their base r counterparts (e.g., read.csv)
- vroom
  - Reads multiple files to a single dataset
  - Guesses the delimiter

What's a tibble?

- A type of data frame: rectangular data with columns (variables of different types) and rows (usually observations)
- A data frame is a spreadsheet, but you can't edit it!

ppts

Filter

	id	firstjump	age	include
1	2	11.82	7.89	1
2	3	29.22	7.26	1
3	4	14.72	11.60	0
4	5	737.70	8.35	1
5	7	33.86	8.12	1
6	8	13.50	7.76	1
7	12	61.88	18.84	1
8	14	13.94	12.62	0
9	15	19.76	12.98	1

What's a tibble?

- Tibbles are data frames that:
  - Don't do "partial matching"
  - Print a bit cleaner in your console
  - You will rarely notice the difference, and you can easily as_tibble() or as.data.frame() back and forth

```
> ppts_df <- as.data.frame(ppts)
> ppts_tb <- as_tibble(ppts)
> ppts_df$ag
 [1]  7.89  7.26 11.60  8.35  8.12  7.76 18.84 12.62 12.98    NA 17.95 12.03 10.78 14.37 12.23 14.66 11.54
> ppts_tb$ag
NULL
Warning message:
Unknown or uninitialised column: `ag`.
```

Writing data

- write_csv(), write_delim()
  - Writes flat files from a data frame
- save()
  - Saves workspace variables as RData files
- ggsave()
  - Save a ggplot to an image file

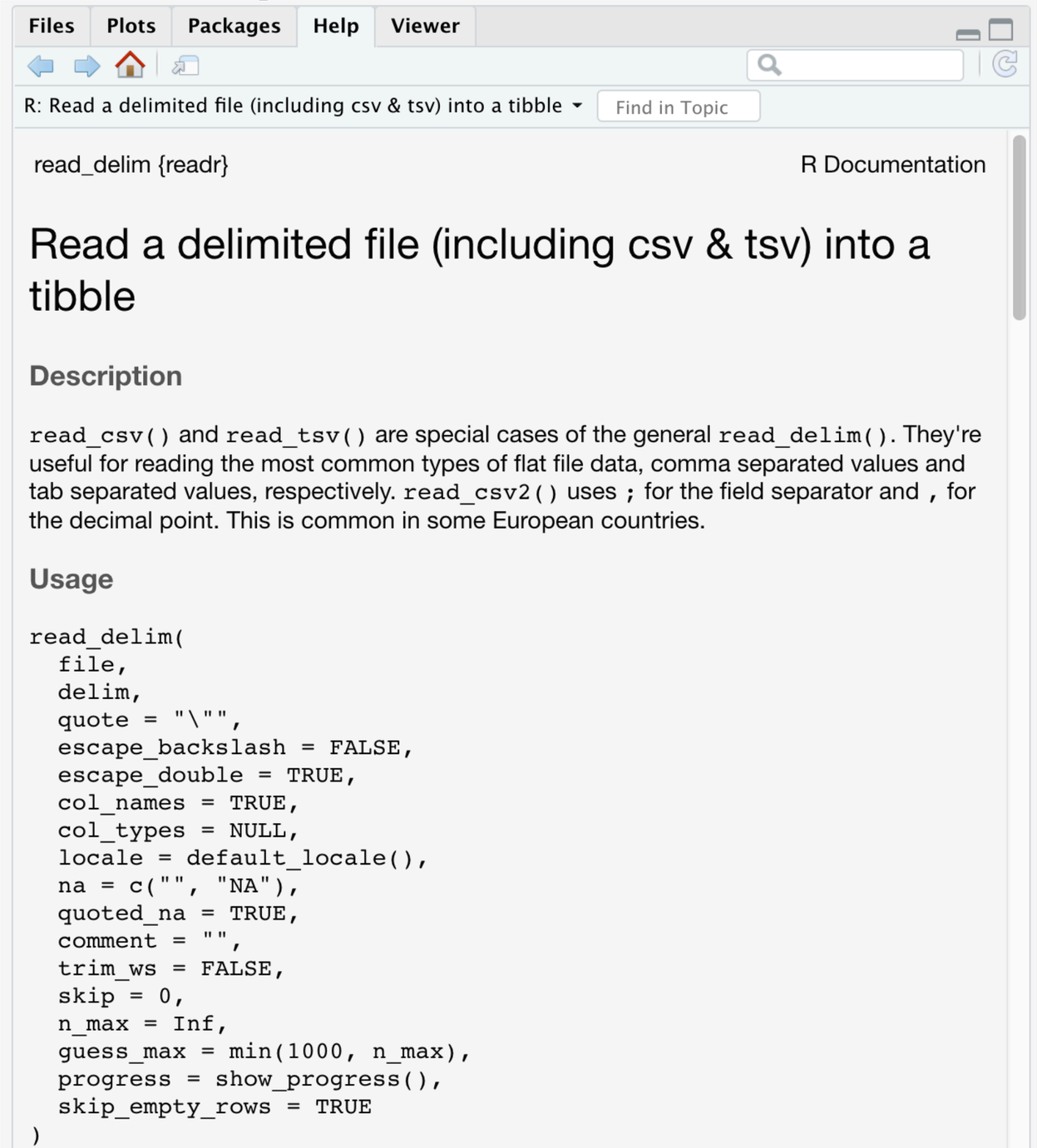
RStudio Cloud

- Find the 259-files-import project and open it
  - It's an "assignment" (but not really) which means you can open it and it creates a copy that you can work in
  - I encourage you to open it up and follow along
  - Add comments/notes to the code (if that helps)
  - Technical problems -> chat with Jake and visit a break-out room

How to get help

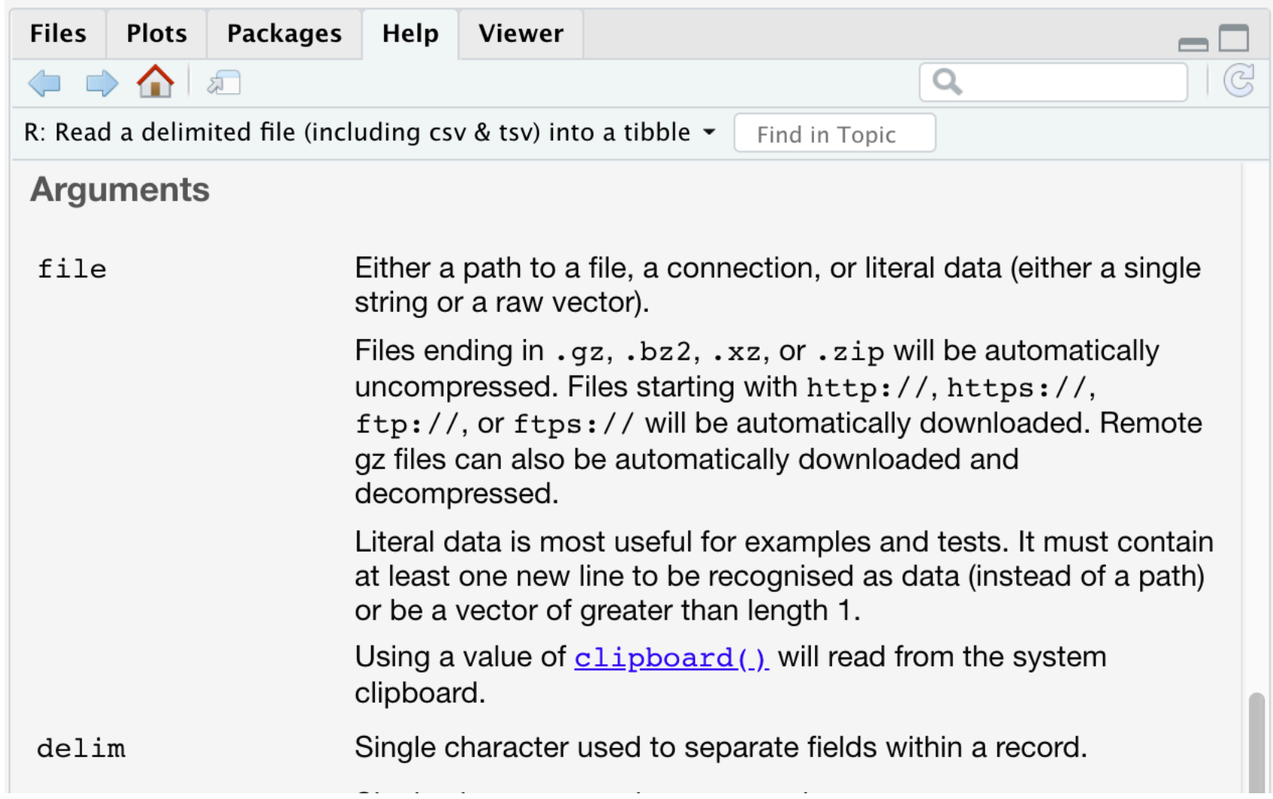
Where to search for help

- R Documentation
  - ?function brings up the documentation for a function (assuming the package is loaded)



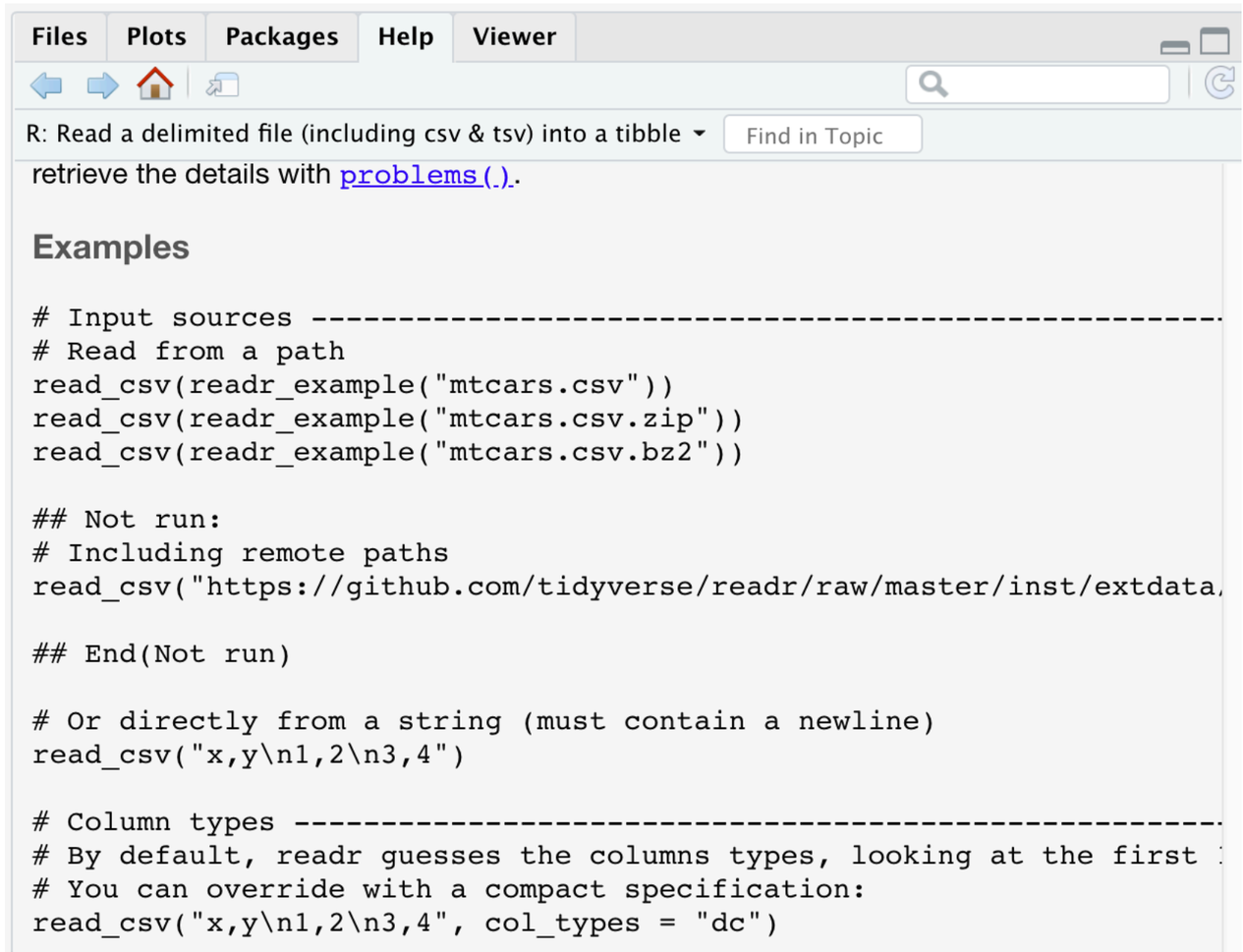
Where to search for help

- R Documentation
  - ?function brings up the documentation for a function (assuming the package is loaded)
  - Scroll down to see the “arguments” definitions



Where to search for help

- R Documentation
  - ?function brings up the documentation for a function (assuming the package is loaded)
- Scroll down to see the “arguments” definitions
- Scroll down more to see examples



Where to search for help

- Package vignettes/blogs
  - https://cran.r-project.org/web/packages/readr/vignettes/readr.html
- Google
  - Blogs/instructional sites
  - StackExchange (especially for error text)
  - Tidyverse community
    - https://community.rstudio.com/c/tidyverse/6
- Take working examples and fiddle!

Tidyverse cheatsheets!

Data Import : : CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.

The front side of this sheet shows how to read text files into R with **readr**.

The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidyr**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save **x**, an R object, to **path**, a file path, as:

Comma delimited file
write_csv(x, path, na = "NA", append = FALSE, col_names = !append**)**

File with arbitrary delimiter
write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = !append**)**

CSV for excel
write_excel_csv(x, path, na = "NA", append = FALSE, col_names = !append**)**

String to file
write_file(x, path, append = FALSE**)**

String vector to file, one element per line
write_lines(x,path, na = "NA", append = FALSE**)**

Object to RDS file
write_rds(x, path, compress = c("none", "gz", "bz2", "xz"), ...**)**

Tab delimited files
write_tsv(x, path, na = "NA", append = FALSE, col_names = !append**)**

Read Tabular Data - These functions share the common arguments:

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive()**)**

a,b,c
1,2,3
4,5,NA

A	B	C
1	2	3
4	5	NA

Comma Delimited Files
read_csv("file.csv"**)**
To make file.csv run:
write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

a;b;c
1;2;3
4;5;NA

A	B	C
1	2	3
4	5	NA

Semi-colon Delimited Files
read_csv2("file2.csv"**)**
write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

a|b|c
1|2|3
4|5|NA

A	B	C
1	2	3
4	5	NA

Files with Any Delimiter
read_delim("file.txt", delim = "|"**)**
write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")

a b c
1 2 3
4 5 NA

A	B	C
1	2	3
4	5	NA

Fixed Width Files
read_fwf("file.fwf", col_positions = c(1, 3, 5)**)**
write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv"**)** Also **read_table().**
write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

USEFUL ARGUMENTS

a,b,c
1,2,3
4,5,NA

Example file
write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

A	B	C
1	2	3
4	5	NA

No header
read_csv(f, **col_names = FALSE**)

x	y	z
A	B	C
1	2	3
4	5	NA

Provide header
read_csv(f, **col_names = c("x", "y", "z")**)

1	2	3
4	5	NA

Skip lines
read_csv(f, **skip = 1**)

A	B	C
1	2	3

Read in a subset
read_csv(f, **n_max = 1**)

A	B	C
NA	2	3
4	5	NA

Missing Values
read_csv(f, **na = c("1", ".")**)

Read Non-Tabular Data

Read a file into a single string
read_file(file, locale = default_locale()**)**

Read each line into its own string
read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive()**)**

Read Apache style log files
read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive()**)**

Read a file into a raw vector
read_file_raw(file**)**

Read each line into a raw vector
read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive()**)**

Data types

readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

A message shows the type of each column in the result.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

age is an integer
sex is a character
earn is a double (numeric)

1. Use **problems()** to diagnose problems.
x <- read_csv("file.csv"); problems(x)

2. Use a col_ function to guide parsing.
- **col_guess()** - the default
- **col_character()**
- **col_double()**, **col_euro_double()**
- **col_datetime(**format = ""**)** Also **col_date(**format = ""**)**, **col_time(**format = ""**)**
- **col_factor(**levels, ordered = FALSE**)**
- **col_integer()**
- **col_logical()**
- **col_number()**, **col_numeric()**
- **col_skip()**

```
x <- read_csv("file.csv", col_types = cols(
  A = col_double(),
  B = col_logical(),
  C = col_factor()))
```

3. Else, read in as character vectors then parse with a parse_ function.
- **parse_guess()**
- **parse_character()**
- **parse_datetime()** Also **parse_date()** and **parse_time()**
- **parse_double()**
- **parse_factor()**
- **parse_integer()**
- **parse_logical()**
- **parse_number()**

```
x$A <- parse_number(x$A)
```

Tidyverse cheatsheets!

Data Transformation with dplyr : : **CHEAT SHEET**

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes

x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

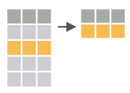
group_by(.data, ..., add = FALSE)
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …)
Returns ungrouped copy of table.
ungroup(g_iris)

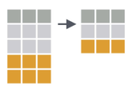
Manipulate Cases

EXTRACT CASES

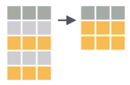
Row functions return a subset of rows as a new table.



filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

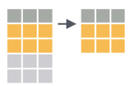


distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*



slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::Logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL)
Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

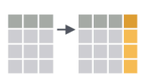
Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)	**num_range**(prefix, range)	**:**, e.g. mpg:cyl
ends_with(match)	**one_of**(…)	**-**, e.g, -Species
matches(match)	**starts_with**(match)	

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

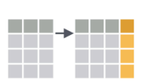


mutate(.data, …)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

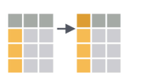
transmute(.data, …)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))



add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*



rename(.data, …) Rename columns.
rename(iris, Length = Sepal.Length)

Homework 1

Homework 1 - Data import (no Github yet)

- Open my RStudio project, which will copy it to your workspace
  - Make all changes in the script file on R Studio Cloud
    - Jake and I will be able to see your progress
  - Complete by next class
  - Questions are increasingly difficult and are meant to challenge you
  - What counts as finished? At least 4 of the questions