Essential Research Toolkit for the Humanities

Week 5: Data manipulation and error handling

Anna Pryslopska

May 6, 2024

Psycholinguistics and Cognitive Modeling Lab

Homework

✔ Good naming!

✔ (Very) fancy ways of doing the selection of items, fine as long as you made sure to look through the responses

❝ **Don't understand, have problems, idk** ❞ isn't good enough (only once): ask in forum, describe better, contact me, best honest attempt

✖ Task 4: forgot printing press

✖ Task 6: forgot the empty set, first item, or a duck, or EVERYTHING

🛈 Only need to load packages once

🛈 Surname is enough in file name or in file

ℹ️ All assignments can be done with code from class.

😐 Used %>% instead of |> (native, newer, more readable, consistent with other languages)

❗ You don't need to use my way to do things, you can show off your skills! But make sure your code works (no assignment = no variable, syntax errors, math operators + characters = ✖), include packages.

```
64 noisy.rt <- noisy |> rename(, ID = MD5.hash.of.participant.s.IP.address,
65                             SENTENCE = Sentence..or.sentence.MD5.) |>
66                        filter(, Label == "experiment",
67                               PennElementType != "Scale",
68                               PennElementType != "TextInput",
69                               Reading.time != "NULL") |>
70                        mutate(, RT.numeric = as.numeric(Reading.time)) |>
71                        filter(, Reading.time %in% 80:2000) |>
72                        select(, c(ID, ITEM, CONDITION, SENTENCE, Reading.time,
73                               Parameter)) |>
74                        na.omit()
```

1. What do the following evaluate to and why? Operator coercion

`0`	`FALSE + 0L`	coerces logical → numeric: `FALSE` to `0`
`1`	`1 - FALSE`	same reason
`1`	`FALSE + 1`	same reason
✖	`!TRUE`	not `TRUE`
✔	`5 & TRUE`	coerces numeric → logical: `5` to `TRUE`
✖	`0 & TRUE`	same as `FALSE & TRUE`
✔	`1 | FALSE`	tautology: same as `FALSE | TRUE`
`NA`	`FALSE | NA`	`NA` is special

2. Why do the following functions fail?

```
Summary(moses)
read_cvs(moses.csv)
tail(moses, n==10)
describe(Moses)
filter(moses, Condition == 102)
arragne(moses, ID)
mutate(moses, Items = as.character("ITEM"))
```

3. Clean up the Moses illusion data using pipes.

```
moses |>
    rename(ID=MD5.hash.of.participant.s.IP.address,
           ANSWER=Value)|>
    select(ITEM, CONDITION, ANSWER, ID, Label,
           Parameter) |>
    na.omit() |>
    filter(Parameter == "Final",
           Label != "instructions",
           CONDITION %in% 1:2) |>
    arrange (ITEM, CONDITION, desc(ANSWER)) |>
    mutate( ITEM = as.numeric( ITEM ) ) -> moses.clean
```

4. Make two new variables: “printing” and “dont.know”.

```
dont.know <- c("Don't Know", "Don't know", "don't knoe",
"don't know", "don`t know", "dont know", "don´t know",
"i don't know", "Not sure", "no idea", "forgotten",
"I do not know", "I don't know")
```

```
moses |> select(RESPONSE) |> arrange(RESPONSE) |>
unique()
```

```
printing <- c("Print", "printer", "the book printer",
"Printing books", "printing press", "press", "Press",
"letter press", "letterpress", "Letterpressing",
"inventing printing", "book printing", "bookpress",
"inventing the book press/his bibles", "Buchdruck",
"finding a way to print books", "Book print", "Drucka",
"for inventing the pressing machine", "Printing")
```

```
moses |> filter(ITEM == 108) |> select(RESPONSE) |>
unique()
```

5. Preprocess noisy channel data.

```
noisy.aj <-
   noisy |>
   filter(Label == "experiment",
          PennElementType == "Scale") |>
   mutate(RATING = as.numeric(Value)) |>
   rename(ID = "MD5.hash.of.participant.s.IP.address") |>
   select(ID, ITEM, CONDITION, RATING) |>
   na.omit()
```

5. Preprocess noisy channel data.

```
noisy.rt <-
   noisy |>
   rename(ID = "MD5.hash.of.participant.s.IP.address",
          SENTENCE = "Sentence..or.sentence.MD5.") |>
   mutate(RT = as.numeric(Reading.time)) |>
   filter(Label == "experiment",
          PennElementType != "Scale",
          PennElementType != "TextInput",
          Reading.time != "NULL",
          RT > 80 & RT < 2000) |>
   select(ID, ITEM, CONDITION, SENTENCE, RT,
          Parameter) |>
   na.omit()
```

✔ `filter(PennElementType != "Scale",`
`PennElementType != "TextInput")`

✔ `filter(PennElementType != "Scale" &`
`PennElementType != "TextInput")`

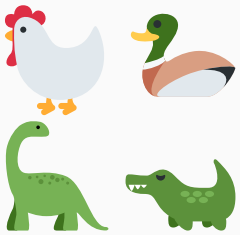
★ `filter(!PennElementType %in% c("Scale", "TextInput"))`

`filter(!(PennElementType %in% c("Scale", "TextInput")))`

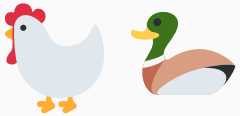
✖ `filter(PennElementType != "Scale" |`
`PennElementType != "TextInput")` returns everything

✖ `filter(PennElementType != "Scale|TextInput")` 1 character

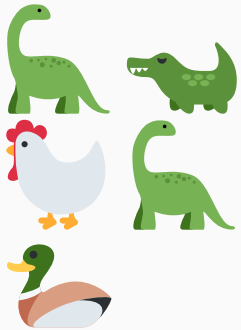
6. Solve the logic exercise (`bird`, `can swim`, `|`, `&`, `!`)



`bird`:



`can swim`:



∅

`!bird`

`!can swim`

`bird & can swim`

`!bird & can swim`

`!bird & !can swim`

`bird & !can swim`

`bird | can swim`

`!bird | can swim`

`bird | !bird`

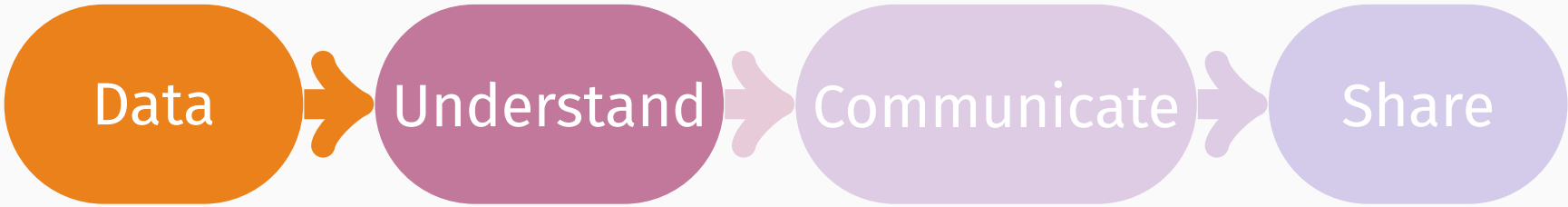
`bird & !bird`

Questions?

Table of contents

Where are we this week?

Recap



R & RStudio, packages, data types, formats, encoding

import from workspace, assign values, operations, clean, filter, arrange, select, merge, group, summarize, export, visualize

document, create clean and beautiful reports

connect, collaborate, backup

Tidy code

A rose by any other name

Names uniquely identify variables and functions

Capitalization matters `DATA != Data != data`

Everyone has an opinion/preference

- ✔ `for_snake_case`
- ✔ `somePreferCamelCase`
- ✔ `others.use.periods`
- ✔ `Some_people.just.WANT_To_Watch.theWorldBURN`
- ✔ `Num8er5` are fine
- ✖ `Sp aces` are not
- ✖ `Speci@l+character$&operator<s>!` are a illegal

R has no official style guide, but preference for alphanumeric & _

Naming variables and functions

R doesn’t care but you do

- ✔ `ueOd2FNRGAP0dRopq4OqU`
- ✔ `ueOd2FNRGAPOdRopq4OqU`
- ✔ `ueOb2FNRGAPOdRopq4OqU`
- ✔ `He_just.kept_talking.in.one.long_incredibly.unbroken.`

What do I call you?

Use descriptive names and be consistent!

GOOD `raw_data, corpus.cleaned, items`

OK `dat, data, model, ratings, corpus`

BAD `d, df, var, aaa, foo, temp`

- ➔ reduce effort
- ➔ stay comprehensible and meaningful (good science is reproducible!)
- ➔ easy to remember and self-explanatory
- ➔ length doesn't matter (much): use TAB to autocomplete

Joining data frames

Add correct answers

Download preprocessed Moses illusion data and questions & answers from ILIAS and save them as `moses` and `questions`.

Put the two together using `merge()` (base R) or `*_join()` (`dplyr`)

```
merge(x=DATA1, y=DATA2, by=COLUMNS)
```

```
*_join(x=DATA1, y=DATA2, by=COLUMNS)
```

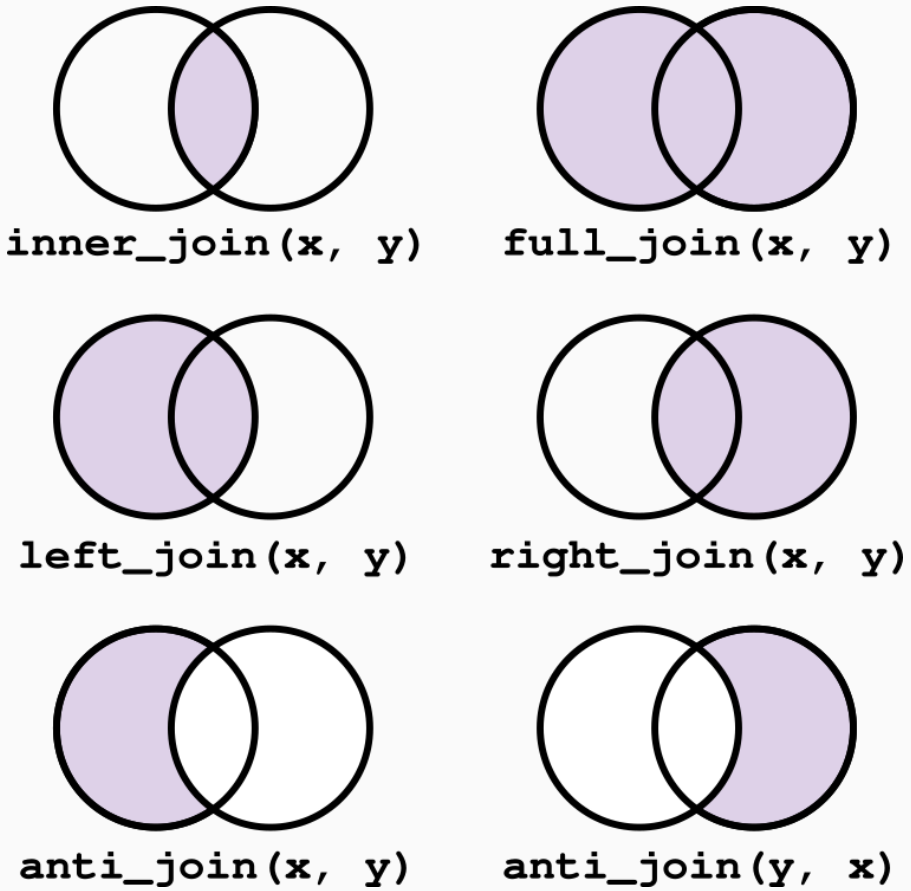
Mutating joins

`left_join()`
keep all observations in x.

`right_join()`
keep all observations in y.

`full_join()`
keep all observations in x and y.

`inner_join()`
keep all observations from x that have a matching key in y



Hugo Tavares GitHub

Task 1: Merge Moses illusion data with questions and answers

Using pipes, merge `moses` and `questions` by the columns `"ITEM", "CONDITION", "LIST"`. Then select the columns `ITEM, CONDITION, ID, ANSWER, CORRECT_ANSWER, LIST`.

```
merge(x = DATA1, y = DATA2, by = c(COLUMNS))
inner_join(x = DATA1, y = DATA2, by = c(COLUMNS))
full_join(x = DATA1, y = DATA2, by = c(COLUMNS))
```

If-else statements

Calculate accuracy: What if?

Was the answer correct or incorrect?

```
moses |>
  mutate(ACCURATE = CORRECT_ANSWER == ANSWER)          logical
```

```
ifelse(TEST, DO WHEN TRUE, DO WHEN FALSE)
```

```
moses |>
  mutate(ACCURATE = ifelse(test = CORRECT_ANSWER ==
ANSWER, yes = TRUE, no = FALSE))                       logical
```

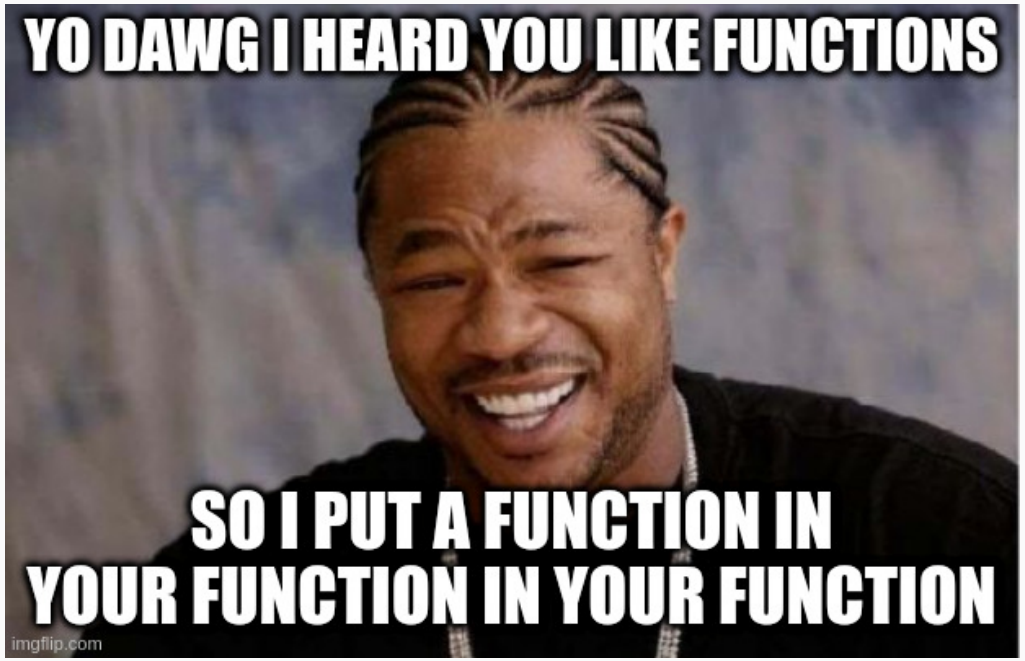
```
moses |>
  mutate(ACCURATE = ifelse(CORRECT_ANSWER == ANSWER,
"correct", "incorrect"))                               strings
```

Task 2: Did you fall for the illusion?

Was the answer correct or incorrect? Make a new column which compares the correct answer with the answer given.

```
moses |>
    inner_join(questions, by=c("ITEM", "CONDITION",
              "LIST")) |>
    select(ITEM, CONDITION, ID, ANSWER, CORRECT_ANSWER,
           LIST) |>
    mutate(ACCURATE = ANSWER == CORRECT_ANSWER)
```

Accuracy: correct or incorrect or don't know?




```
moses |>
    mutate(ACCURATE = ifelse(CORRECT_ANSWER == ANSWER,
        yes = "correct",
        no = ifelse(ANSWER == "dont_know",
                    yes = "dont_know",
                    no = "incorrect")))
```

Task 3: Did you fall for the illusion or know the answer?

Was the answer correct or incorrect or "don't know"? Make a new column which compares the correct answer with the answer given.

```
moses |>
    inner_join(…) |>
    select(…) |>
    mutate(ACCURATE = ifelse(TRUE CONDITION,
           yes = …, no = ifelse(…)))
```

Goal of experiment: Trick, no treat

Groups/conditions in experiment

1 **Moses illusion**
"What is the name of the first man to walk on the *sun*?"

2 **Well-formed question**
"What is the name of the first man to walk on the *moon*?"

100 Well-formed control "In which country is Florida located?"

101 Bad control "Which Nordic country are coconut trees native to?"

Predictions

1	No correct answers but you will try to answer anyway
2	Correct answer is predefined (e.g. Armstrong)
100	Correct answer is predefined (e.g. USA)
101	No correct answers and you will notice this

Condition numbers are meaningless!

```
mutate(WHERE, NEW=case_when(TRUE CONDITION ~ NEW VALUE))
```

```
moses |>
    mutate(CONDITION = case_when(
           CONDITION == '1' ~ 'illusion',
           CONDITION == '2' ~ 'no illusion',
           CONDITION == '100' ~ 'good filler',
           CONDITION == '101' ~ 'bad filler')
          )
```

Task 4: Change conditions to strings

Change the `CONDITION` column or make a new one where the conditions have descriptive names/strings (e.g. “illusion”) instead of numbers.

Nr	Description
1	Illusion (no correct answers)
2	Control (correct answer is predefined)
100	Good filler (correct answer is predefined)
101	Bad filler (no correct answers)

```
mutate(WHERE, NEW=case_when(TRUE CONDITION ~ NEW VALUE))
```

Grouping and summarizing

Grouping

`group_by(WHERE, BY WHAT)`
changes the unit of analysis from the complete dataset to particular groups.

`ungroup(WHERE)`
undos grouping.

Invisible but useful for summaries: How did the groups compare?

Summarizing

`summarise(WHERE, NEW=FUNCTION(VALUE))`
calculates values.

```
summarise(my.awesome.data,
  Count = n(),                    count cases
  Mean = mean(RT),                average reading time
  SD = sd(RT),                    how spread out is the data
  Min = min(Rating),              minimal value
  Sum = sum(Value))               sum of values
```

`mutate()`	changes an existing column or adds a new one.
`summarise()`	calculates a single value (per group).

Task 5: Did you get got?

Summarize the results from the Moses illusion experiment. Count the answer types per condition (correct, incorrect, don't know).

```
group_by(WHERE, WHAT)
summarize(WHERE, NEW = FUNCTION(VALUE))
```

```
moses |>
    …
    group_by(CONDITION, ACCURATE) |>
    summarise(Count = n())
```

Getting help

A little stuck: In R

- 🤔 `help("NAME")`
- 😟 `?NAME`
- 😰 `??NAME`
- RStudio help window
- Set the correct working directory `setwd(DIRECTORY)`
- Load the right packages `library(PACKAGE)`
- Unload conflicting packages `detach(PACKAGE)`
- Disambiguate functions `dplyr::filter()` or `stats::filter()`
- Clean environment and restart

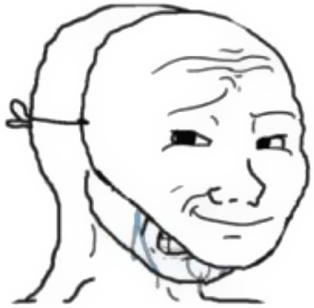
A little stuck: Online

- Google the exact error message:
  “Error in esquisser() : could not find function ”esquisser””
- RDocumentation `www.rdocumentation.org`
- Cheatsheets `www.rstudio.com/resources/cheatsheets`
- Tidyverse `www.tidyverse.org`
- Stackoverflow `stackoverflow.com`
- Reddit `reddit.com/r/rstats`

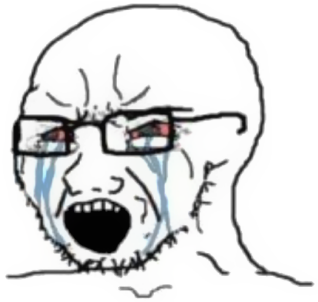
Create a *minimal reproducible example*
`gist.github.com/hadley/270442`

Everyone does it

DESIGNERS



Look, we have similar ideas.



No! You stole my idea.

PROGRAMMERS

Man, I stole your code.

It's not my code.

Wrap-up

Summary

- ✔ tidy code
- ✔ power of names
- ✔ joining data frames
- ✔ if...else
- ✔ grouping
- ✔ summarizing
- ✔ getting help
- ⏩ Data visualization
- 💡 Recommended reading for troubleshooting: `www.r4wrds.com/intro/m_troubleshooting.html`

Homework assignment due May 10th 15:30

Submit 1 R script.

- Read `https://r.qcbs.ca/workshop03/pres-en/workshop03-pres-en.html`
- Complete assignment 4 ($\rightarrow$ ILIAS)