

Outline

- Workflow assignment
- Tidy data + tutorial
- Go over wrangling homework
- Automation lecture + tutorial
- Break-out groups

Workflow assignment

Pick a project and critique the current workflow

- Something you're working on or something similar to what you want/plan to work on
- Ideally, contains raw data that needs to be cleaned
- If you're not sure about the project you've picked, discuss it with me

Two parts

- Written report (2-4 pages, 2x spaced)
 - Turn in on canvas by 1:30pm on 2/10
- Class presentation (7-8 minutes)
 - In class on 2/10 or 2/17
- Each part will
 - Describe workflow (1/3 of length/time)
 - Critique the efficiency, fidelity, & reproducibility (remaining 2/3 of length/time)

PSYC 259:

Principles of Data Science

Week 4: Part 1

Tidy Data

Why do data need to be wrangled and tidied?

- Data files are often created without the foresight of how they might be used by computers
 - Instead, they are made to look nice for humans
- We need to tidy data to make it convenient for programming
 - Format is dictated by analyses
 - You may need multiple formats

Tidy data (Wickham, 2014)

- Each variable is a column
 - Variable: "All values that measure the same underlying attribute"
- Each observation is a row
 - Observation: Unit for which all variables were measured (person, session, trial, etc.)
- Each type of observational unit is a table
 - e.g., participant questionnaire vs trial-by-trial data

Common types of “un-tidy” data: Column names are values

```
relig_income
#> # A tibble: 18 x 11
#>   religion `<$10k` ` $10-20k` ` $20-30k` ` $30-40k` ` $40-50k` ` $50-75k` ` $75-100k`
#>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
#> 1 Agnostic      27        34        60        81        76       137       122
#> 2 Atheist       12        27        37        52        35        70        73
#> 3 Buddhist      27        21        30        34        33        58        62
#> 4 Catholic    418       617       732       670       638     1116     949
```


But what's the problem?

```
relig_income
#> # A tibble: 18 x 11
#>   religion `<$10k` ` $10-20k` ` $20-30k` ` $30-40k` ` $40-50k` ` $50-75k` ` $75-100k`
#>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
#> 1 Agnostic      27        34        60        81        76       137       122
#> 2 Atheist       12        27        37        52        35        70        73
#> 3 Buddhist      27        21        30        34        33        58        62
#> 4 Catholic    418       617       732       670       638     1116     949
```

But what's the problem?

```
relig_income
#> # A tibble: 18 x 11
#>   religion `<$10k` ` $10-20k` ` $20-30k` ` $30-40k` ` $40-50k` ` $50-75k` ` $75-100k`
#>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
#> 1 Agnostic    27         34         60         81         76        137        122
#> 2 Atheist     12         27         37         52         35         70         73
#> 3 Buddhist    27         21         30         34         33         58         62
#> 4 Catholic  418        617        732        670        638       1116       949
```

- Income level can't be made a factor to use in a model
- Challenging to summarize, each income level needs to be treated as a separate variable
- Need to use *different verbs* to subset observations (*filter* for religion, *select* for income)

Solution: pivot_longer() turns columns into rows

```
relig_income
#> # A tibble: 18 x 11
#>   religion `<$10k` ` $10-20k` ` $20-30k` ` $30-40k` ` $40-50k` ` $50-75k` ` $75-100k`
#>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
#> 1 Agnostic      27        34        60        81        76       137       122
#> 2 Atheist       12        27        37        52        35        70        73
#> 3 Buddhist      27        21        30        34        33        58        62
#> 4 Catholic    418       617       732       670       638     1116     949
```

```
relig_income %>%
  pivot_longer(!religion, names_to = "income", values_to = "count")
#> # A tibble: 180 x 3
#>   religion income      count
#>   <chr>      <chr>      <dbl>
#> 1 Agnostic <$10k           27
#> 2 Agnostic $10-20k         34
#> 3 Agnostic $20-30k         60
#> 4 Agnostic $30-40k         81
#> 5 Agnostic $40-50k         76
```

Common types of “un-tidy” data: Multiple variables in one column

```
> header1  
# A tibble: 7 x 2  
  field      value  
  <chr>    <chr>  
1 Participant 6191  
2 Age        25  
3 Sex        male  
4 Order      1  
5 FirstSpeed 2.4  
6 Block      1  
7 2.4        near
```

But what's the problem?

```
> header1
# A tibble: 7 x 2
  field      value
  <chr>      <chr>
1 Participant 6191
2 Age         25
3 Sex         male
4 Order        1
5 FirstSpeed  2.4
6 Block        1
7 2.4         near
```

But what's the problem?

```
> header1
# A tibble: 7 x 2
  field      value
  <chr>      <chr>
1 Participant 6191
2 Age         25
3 Sex         male
4 Order       1
5 FirstSpeed  2.4
6 Block       1
7 2.4         near
```

- Mixing types in the same column
- No meaningful summary of "value"
- Impossible to use values in a model (i.e., outcome ~ order*sex*block)

Solution: pivot_wider() turns rows into columns

```
> header1
# A tibble: 7 x 2
  field      value
  <chr>      <chr>
1 Participant 6191
2 Age         25
3 Sex         male
4 Order        1
5 FirstSpeed  2.4
6 Block        1
7 2.4         near
```

```
> pivot_wider(header1, names_from = "field", values_from = "value")
# A tibble: 1 x 7
  Participant Age Sex Order FirstSpeed Block `2.4`
  <chr>      <chr> <chr> <chr> <chr>      <chr> <chr>
1 6191      25 male  1     2.4        1 near
```

Common types of "un-tidy" data: Variables glued into one column


country	year	rate		country	year	cases	pop
A	1999	0.7K/19M		A	1999	0.7K	19M
A	2000	2K/20M		A	2000	2K	20M
B	1999	37K/172M		B	1999	37K	172
B	2000	80K/174M		B	2000	80K	174
C	1999	212K/1T		C	1999	212K	1T
C	2000	213K/1T		C	2000	213K	1T

*`separate(table3, rate, sep = "/",
into = c("cases", "pop"))`*

- Solution: use `separate()`

Common types of “un-tidy” data: Variables split across columns

country	century	year
Afghan	19	99
Afghan	20	00
Brazil	19	99
Brazil	20	00
China	19	99
China	20	00



country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

```
unite(table5, century, year,  
      col = "year", sep = "")
```

- Solution: use unite()

Common types of “un-tidy” data: Observations split across tables

DF	A	B	C
x	a	t	1
x	b	u	2
x	c	v	3
z	c	v	3
z	d	w	4

bind_rows(..., .id = NULL)

Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured)

- Solution: use `bind_rows()`
- Solution: use `vroom()` if data are split across files

Common types of “un-tidy” data: Variables split across tables

x			y		
A	B	C	A	B	D
a	t	1	a	t	3
b	u	2	b	u	2
c	v	3	d	w	1

A	B	C	D
a	t	1	3
b	u	2	2
c	v	3	NA

**left_join(x, y, by = NULL,
copy=FALSE, suffix=c(“x”,“y”),...)**
Join matching values from y to x.

- Solution: use left_join() or other joins
- Why not bind_cols?

Data tidying tutorial

Example scripts 3 and 4 in
"259-data-wrangling"