Batch loading data and list columns

Daniel Anderson

Week 4

Agenda

- Discuss the midterm
    - Canvas quiz (10 points; please don't stress)
    - Take home (40 points)
- Review Lab 2
- `map_dfr` and batch-loading data

- Introduce list columns
- Contrast:
  - `group_by() %>% nest() %>% mutate() %>% map()` with
  - `nest_by() %>% summarize()`
- In–class midterm (last 25 minutes)

Review Lab 2

Learning objectives

- Understand when `map_dfr` can and should be applied
- Better understand file paths, and how `{fs}` can help
- Be able to batch load data of a specific type within a mixed-type directory
- Use filenames to pull data

Learning objectives (cont.)

- Understand list columns and how they relate to `base::split`
- Fluently nest/unnest data frames
- Understand why `tidyr::nest` can be a powerful framework (data frames) and when `tidyr::unnest` can/should be used to move out of nested data frames and into a standard data frame.

Midterm

Questions?

Let's look at the take-home portion

`map_dfr`

- If each iteration returns a data frame, you can use `map_dfr` to automatically bind all the data frames together.

Example

- Create a function that simulates data (please copy the code and follow along)

```
set.seed(8675309)
simulate <- function(n, mean = 0, sd = 1) {
    tibble(sample_id = seq_len(n),
           sample = rnorm(n, mean, sd))
}
simulate(10)
```

```
## # A tibble: 10 × 2
##    sample_id     sample
##        <int>      <dbl>
##  1         1 -0.9965824
##  2         2  0.7218241
##  3         3 -0.6172088
##  4         4  2.029392
##  5         5  1.065416
##  6         6  0.9872197
##  7         7  0.02745393
##  8         8  0.6728723
##  9         9  0.5720665
## 10        10  0.9036777
```

Two more quick examples

```r
simulate(3, 100, 10)
```

```
## # A tibble: 3 × 2
##   sample_id    sample
##       <int>     <dbl>
## 1         1  84.50448
## 2         2 110.2264
## 3         3 101.5008
```

```r
simulate(5, -10, 1.5)
```

```
## # A tibble: 5 × 2
##   sample_id     sample
##       <int>      <dbl>
## 1         1 -10.98995
## 2         2 -11.49188
## 3         3  -7.041312
## 4         4 -10.66270
## 5         5 -11.35096
```

Simulation

- Assume we want to vary the sample size from 10 to 150 by increments of 5
- `mean` stays constant at 100, `sd` is constant at 10

Try with `purrr::map`

02:00

```r
library(tidyverse)
sims <- map(seq(10, 150, 5), simulate, mean = 100, sd = 10)
```

```r
sims[1]
```

```
## [[1]]
## # A tibble: 10 × 2
##    sample_id    sample
##        <int>     <dbl>
##  1         1 103.7618
##  2         2 111.5353
##  3         3 115.7490
##  4         4 105.8853
##  5         5  93.84955
##  6         6  97.71089
##  7         7 100.6392
##  8         8  96.86526
##  9         9  97.51501
## 10        10  98.46205
```

```r
sims[2]
```

```
## [[1]]
## # A tibble: 15 × 2
##    sample_id    sample
##        <int>     <dbl>
##  1         1  93.64743
##  2         2  99.96206
##  3         3 100.4562
##  4         4 106.8407
##  5         5  97.47957
##  6         6  98.48961
##  7         7  91.25069
##  8         8  80.23099
##  9         9 102.3766
## 10        10 100.3609
## 11        11 101.3490
## 12        12 101.1758
## 13        13  91.74411
## 14        14  78.64764
## 15        15 102.1421
```

Swap for `map_dfr`

Try it – what happens?

```
sims_df <- map_dfr(seq(10, 150, 5), simulate, 100, 10)
sims_df
```

```
## # A tibble: 2,320 × 2
##    sample_id    sample
##        <int>     <dbl>
##  1         1  85.64361
##  2         2 103.6789
##  3         3  94.71782
##  4         4 103.1350
##  5         5  99.78701
##  6         6 105.3462
##  7         7 100.0653
##  8         8  94.28314
##  9         9 108.8872
## 10        10 106.0850
## # … with 2,310 more rows
```

01:00

Notice a problem here

```
sims_df[1:15, ]
```

```
## # A tibble: 15 × 2
##    sample_id    sample
##        <int>     <dbl>
##  1         1  85.64361
##  2         2 103.6789
##  3         3  94.71782
##  4         4 103.1350
##  5         5  99.78701
##  6         6 105.3462
##  7         7 100.0653
##  8         8  94.28314
##  9         9 108.8872
## 10        10 106.0850
## 11         1  89.49968
## 12         2  86.99898
## 13         3  85.38054
## 14         4  99.10690
## 15         5 105.0088
```

.id argument

```r
sims_df2 <- map_dfr(seq(10, 150, 5), simulate, 100, 10,
                   .id = "iteration")
sims_df2[1:15, ]
```

```
## # A tibble: 15 × 3
##     iteration sample_id    sample
##     <chr>         <int>     <dbl>
##   1 1                 1 112.1250
##   2 1                 2  88.07056
##   3 1                 3 108.3908
##   4 1                 4 100.8193
##   5 1                 5 102.1545
##   6 1                 6 113.5398
##   7 1                 7 101.4171
##   8 1                 8  99.33668
##   9 1                 9 100.2855
##  10 1                10  90.22043
##  11 2                 1  91.08882
##  12 2                 2 107.3664
##  13 2                 3 101.1745
##  14 2                 4  96.82053
##  15 2                 5 105.5844
```

> `.id`: Either a string or NULL. If a string, the output will contain a variable with that name, storing either the name (if .x is named) or the index (if .x is unnamed) of the input. If NULL, the default, no variable will be created.

– {purrr} documentation

setNames

```
sample_size <- seq(10, 150, 5)
sample_size
```

```
##  [1]  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90
## [22] 115 120 125 130 135 140 145 150
```

```
sample_size <- setNames(sample_size,
                        english::english(seq(10, 150, 5)))
sample_size[1:15]
```

```
##          ten      fifteen       twenty  twenty-five       thirty  thirty
##           10           15           20           25           30
##        forty   forty-five        fifty   fifty-five        sixty   sixty
##           40           45           50           55           60
##      seventy seventy-five       eighty
##           70           75           80
```

Try again

```
sims_df3 <- map_dfr(sample_size, simulate, 100, 10,
                   .id = "n")
sims_df3[1:15, ]
```

```
## # A tibble: 15 × 3
##    n       sample_id    sample
##    <chr>       <int>     <dbl>
##  1 ten             1  98.94914
##  2 ten             2 101.6824
##  3 ten             3  88.16447
##  4 ten             4  90.13604
##  5 ten             5  85.53591
##  6 ten             6  90.69977
##  7 ten             7 105.8858
##  8 ten             8  89.12978
##  9 ten             9 114.4982
## 10 ten            10 111.6440
## 11 fifteen         1 103.2732
## 12 fifteen         2 106.8949
## 13 fifteen         3  88.83591
## 14 fifteen         4 105.5402
## 15 fifteen         5 112.6581
```

Another quick example

`broom::tidy`

- The {broom} package helps us extract model output in a tidy format

```
lm(tvhours ~ age, gss_cat) %>%
    broom::tidy()
```

```
## # A tibble: 2 × 5
##   term          estimate   std.error statistic       p.value
##   <chr>            <dbl>       <dbl>     <dbl>         <dbl>
## 1 (Intercept) 1.992391   0.06980966   28.54033 4.672161e-173
## 2 age         0.02095679 0.001387361  15.10551 4.689310e- 51
```

Fit separate models by year

Again – probs not best statistically

```r
split(gss_cat, gss_cat$year) %>%
 map_dfr(
   ~lm(tvhours ~ age, .x) %>%
     broom::tidy()
)
```

```
## # A tibble: 16 × 5
##    term          estimate   std.error statistic      p.value
##    <chr>            <dbl>       <dbl>     <dbl>        <dbl>
##  1 (Intercept) 2.080163   0.1709061   12.17138  7.995632e-33
##  2 age         0.01948584 0.003485199  5.591027 2.599011e- 8
##  3 (Intercept) 2.078999   0.2176829    9.550583 1.191266e-20
##  4 age         0.01963575 0.004400292  4.462375 9.137366e- 6
##  5 (Intercept) 1.767990   0.2464509    7.173804 1.531756e-12
##  6 age         0.02386070 0.005031548  4.742218 2.459650e- 6
##  7 (Intercept) 2.096054   0.1496431   14.00702  1.419772e-42
##  8 age         0.01781388 0.002977289  5.983256 2.589482e- 9
##  9 (Intercept) 1.855278   0.2156381    8.603668 2.167351e-17
## 10 age         0.02390720 0.004314567  5.541043 3.628675e- 8
## 11 (Intercept) 2.068914   0.2096397    9.868903 2.896085e-22
## 12 age         0.01989505 0.004086638  4.868317 1.251234e- 6
## 13 (Intercept) 1.878070   0.2258400    8.315932 2.280108e-16
```

.id

In cases like the preceding, `.id` becomes invaluable

```
split(gss_cat, gss_cat$year) %>%
 map_dfr(
   ~lm(tvhours ~ age, .x) %>%
     broom::tidy(),
   .id = "year"
)
```

```
## # A tibble: 16 × 6
##    year  term          estimate   std.error statistic      p.value
##    <chr> <chr>            <dbl>       <dbl>     <dbl>        <dbl>
##  1 2000  (Intercept) 2.080163   0.1709061   12.17138  7.995632e-33
##  2 2000  age         0.01948584 0.003485199  5.591027 2.599011e- 8
##  3 2002  (Intercept) 2.078999   0.2176829    9.550583 1.191266e-20
##  4 2002  age         0.01963575 0.004400292  4.462375 9.137366e- 6
##  5 2004  (Intercept) 1.767990   0.2464509    7.173804 1.531756e-12
##  6 2004  age         0.02386070 0.005031548  4.742218 2.459650e- 6
##  7 2006  (Intercept) 2.096054   0.1496431   14.00702  1.419772e-42
##  8 2006  age         0.01781388 0.002977289  5.983256 2.589482e- 9
##  9 2008  (Intercept) 1.855278   0.2156381    8.603668 2.167351e-17
## 10 2008  age         0.02390720 0.004314567  5.541043 3.628675e- 8
## 11 2010  (Intercept) 2.068914   0.2096397    9.868903 2.896085e-22
## 12 2010  age         0.01989505 0.004086638  4.868317 1.251234e-
```

Break

05:00

Batch–loading data

Please follow along

{fs}

Could we apply `map_dfr` here?

```r
# install.packages("fs")
library(fs)
dir_ls(here::here("data"))
```

```
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/i
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
```

```r
dir_ls(here::here("data", "pfiles_sim"))
```

```
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
```

Limit files

- We really only want the `.csv`
    - That happens to be the only thing that's in there but that's regularly not the case

```
dir_ls(here::here("data", "pfiles_sim"), glob = "*.csv")
```

```
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
```

Batch load

- Loop through the directories and `import` or `read_csv`

```r
files <- dir_ls(
  here::here("data", "pfiles_sim"),
  glob = "*.csv"
)
batch <- map_dfr(files, read_csv)
batch
```

```
## # A tibble: 15,945 × 22
##    Entry   Theta Status Count RawScore     SE Infit Infit_Z Outfit Outfi
##    <dbl>   <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>   <dbl>  <dbl>    <d
##  1   123  1.2687      1    36       23 0.3713  0.93   -0.34   0.82    -0
##  2    88  1.5541      1    36       25 0.3852  0.95   -0.37   0.81    -0
##  3   105  3.2773      1    36       33 0.6187  0.9    -0.04   1.63     1
##  4   153  4.4752      1    36       35 1.0234  0.93    0.23   0.35    -0
##  5   437  2.6655      1    36       31 0.5008  0.92   -0.18   0.88    -0
##  6   307  5.7137      0    36       36 1.8371  1       0      1         0
##  7   305  3.7326      1    36       34 0.7408  1.06    0.31   0.86     0
##  8    42  0.609       1    36       18 0.36    1.55    2.56   1.74     3
##  9    59 -2.623       1    36        3 1.0344  0.85    0.06   0.17    -0
## 10   304  5.7137      0    36       36 1.8371  1       0      1         0
## # … with 15,935 more rows, and 11 more variables: PointMeasureCorr <dbl>
## #   ObservMatch <dbl>, ExpectMatch <dbl>, PointMeasureExpected <dbl>, RM
```

Problem

- We've lost a lot of info – no way to identify which file is which

Try to fix it!

02:00

Add id

```
batch2 <- map_dfr(files, read_csv, .id = "file")
batch2
```

```
## # A tibble: 15,945 × 23
## # … with 15,935 more rows, and 23 more variables: file <chr>, Entry <dbl
## #   Status <dbl>, Count <dbl>, RawScore <dbl>, SE <dbl>, Infit <dbl>, In
## #   Outfit <dbl>, Outfit_Z <dbl>, Displacement <dbl>, PointMeasureCorr <
## #   Weight <dbl>, ObservMatch <dbl>, ExpectMatch <dbl>, PointMeasureExpe
## #   RMSR <dbl>, WMLE <dbl>, testeventid <dbl>, ssid <dbl>, asmtprmrydsbl
## #   asmtscndrydsbltycd <dbl>
```

Note – the `file` column contains the full path, which is so long it makes no rows print

```
batch2 %>%
    count(file)
```

```
## # A tibble: 31 × 2
## # … with 21 more rows, and 2 more variables: file <chr>, n <int>
```

- Still not terrifically useful. What can we do?

Step 1

- Remove the `here::here` path from string

```
batch2 <- batch2 %>%
    mutate(
      file = str_replace_all(
        file,
        here::here("data", "pfiles_sim"),
        ""
      )
    )

batch2 %>%
  count(file)
```

```
## # A tibble: 31 × 2
##    file                             n
##    <chr>                        <int>
##  1 /g11ELApfiles18_sim.csv        453
##  2 /g11Mathpfiles18_sim.csv       460
##  3 /g11Rdgpfiles18_sim.csv        453
##  4 /g11Sciencepfiles18_sim.csv    438
##  5 /g11Wripfiles18_sim.csv        453
##  6 /g3ELApfiles18_sim.csv         540
```

Pull out pieces you need

- Regular expressions are most powerful here
  - We haven't talked about them much
- Try RegExplain

Pull grade

- Note – I'm not expecting you to just suddenly be able to do this. This is more for illustration. There's also other ways you could extract the same info

```
batch2 %>%
    mutate(
      grade = str_replace_all(
        file,
        "/g(\\d?\\d).+",
        "\\1"
      )
  ) %>%
    select(file, grade)
```

```
## # A tibble: 15,945 × 2
##    file                    grade
##    <chr>                   <chr>
##  1 /g11ELApfiles18_sim.csv 11
##  2 /g11ELApfiles18_sim.csv 11
##  3 /g11ELApfiles18_sim.csv 11
##  4 /g11ELApfiles18_sim.csv 11
##  5 /g11ELApfiles18_sim.csv 11
```

parse_number

- In this case `parse_number` also works – but note that it would not work to extract the year

```r
batch2 %>%
  mutate(grade = parse_number(file)) %>%
  select(file, grade)
```

```
## # A tibble: 15,945 × 2
##    file                     grade
##    <chr>                    <dbl>
##  1 /g11ELApfiles18_sim.csv     11
##  2 /g11ELApfiles18_sim.csv     11
##  3 /g11ELApfiles18_sim.csv     11
##  4 /g11ELApfiles18_sim.csv     11
##  5 /g11ELApfiles18_sim.csv     11
##  6 /g11ELApfiles18_sim.csv     11
##  7 /g11ELApfiles18_sim.csv     11
##  8 /g11ELApfiles18_sim.csv     11
##  9 /g11ELApfiles18_sim.csv     11
## 10 /g11ELApfiles18_sim.csv     11
## # … with 15,935 more rows
```

Extract year

```
batch2 %>%
    mutate(
      grade = str_replace_all(
        file, "/g(\\d?\\d).+", "\\1"
    ),
   year = str_replace_all(
     file, ".+files(\\d\\d)_sim.+", "\\1"
    )
  ) %>%
    select(file, grade, year)
```

```
## # A tibble: 15,945 × 3
##    file                    grade year 
##    <chr>                   <chr> <chr>
##  1 /g11ELApfiles18_sim.csv 11    18   
##  2 /g11ELApfiles18_sim.csv 11    18   
##  3 /g11ELApfiles18_sim.csv 11    18   
##  4 /g11ELApfiles18_sim.csv 11    18   
##  5 /g11ELApfiles18_sim.csv 11    18   
##  6 /g11ELApfiles18_sim.csv 11    18   
##  7 /g11ELApfiles18_sim.csv 11    18   
##  8 /g11ELApfiles18_sim.csv 11    18   
##  9 /g11ELApfiles18_sim.csv 11    18   
## 10 /g11ELApfiles18_sim.csv 11    18   
```

Extract Content Area

```r
batch2 %>%
    mutate(grade = str_replace_all(file,
                                   "/g(\\d?\\d).+",
                                   "\\1"),
           year = str_replace_all(file,
                                  ".+files(\\d\\d)_sim.+",
                                  "\\1"),
        content = str_replace_all(file,
                                  "/g\\d?\\d(.+)pfiles.+",
                                  "\\1")) %>%
    select(file, grade, year, content)
```

```
## # A tibble: 15,945 × 4
##    file                     grade year  content
##    <chr>                    <chr> <chr> <chr>
##  1 /g11ELApfiles18_sim.csv 11    18    ELA
##  2 /g11ELApfiles18_sim.csv 11    18    ELA
##  3 /g11ELApfiles18_sim.csv 11    18    ELA
##  4 /g11ELApfiles18_sim.csv 11    18    ELA
##  5 /g11ELApfiles18_sim.csv 11    18    ELA
##  6 /g11ELApfiles18_sim.csv 11    18    ELA
##  7 /g11ELApfiles18_sim.csv 11    18    ELA
##  8 /g11ELApfiles18_sim.csv 11    18    ELA
##  9 /g11ELApfiles18_sim.csv 11    18    ELA
```

Double checks: grade

```
batch2 %>%
    mutate(grade = str_replace_all(file,
                                   "/g(\\d?\\d).+",
                                   "\\1"),
           year = str_replace_all(file,
                                  ".+files(\\d\\d)_sim.+",
                                  "\\1"),
           content = str_replace_all(file,
                                     "/g\\d?\\d(.+)pfiles.+",
                                     "\\1")) %>%
    select(file, grade, year, content) %>%
    count(grade)
```

```
## # A tibble: 7 × 2
##   grade     n
##   <chr> <int>
## 1 11     2257
## 2 3      2156
## 3 4      2341
## 4 5      2632
## 5 6      2216
## 6 7      1962
## 7 8      2381
```

Double checks: year

```r
batch2 %>%
    mutate(grade = str_replace_all(file,
                                   "/g(\\d?\\d).+",
                                   "\\1"),
           year = str_replace_all(file,
                                  ".+files(\\d\\d)_sim.+",
                                  "\\1"),
           content = str_replace_all(file,
                                     "/g\\d?\\d(.+)pfiles.+",
                                     "\\1")) %>%
    select(file, grade, year, content) %>%
    count(year)
```

```
## # A tibble: 1 × 2
##   year      n
##   <chr> <int>
## 1 18    15945
```

Double checks: content

```r
batch2 %>%
    mutate(grade = str_replace_all(file,
                                   "/g(\\d?\\d).+",
                                   "\\1"),
           year = str_replace_all(file,
                                  ".+files(\\d\\d)_sim.+",
                                  "\\1"),
           content = str_replace_all(file,
                                     "/g\\d?\\d(.+)pfiles.+",
                                     "\\1")) %>%
    select(file, grade, year, content) %>%
    count(content)
```

```
## # A tibble: 5 × 2
##   content     n
##   <chr>   <int>
## 1 ELA      3627
## 2 Math     3629
## 3 Rdg      3627
## 4 Science  1435
## 5 Wri      3627
```

Finalize

```r
d <- batch2 %>%
    mutate(
      grade = str_replace_all(
        file, "/g(\\d?\\d).+", "\\1"
    ),
      grade = as.integer(grade),
      year = str_replace_all(
        file, ".+files(\\d\\d)_sim.+", "\\1"
    ),
      year = as.integer(grade),
      content = str_replace_all(
        file, "/g\\d?\\d(.+)pfiles.+", "\\1"
    )
  ) %>%
    select(-file) %>%
    select(
      ssid, grade, year, content, testeventid,
      asmtprmrydsbltycd, asmtscndrydsbltycd, Entry:WMLE
  )
```

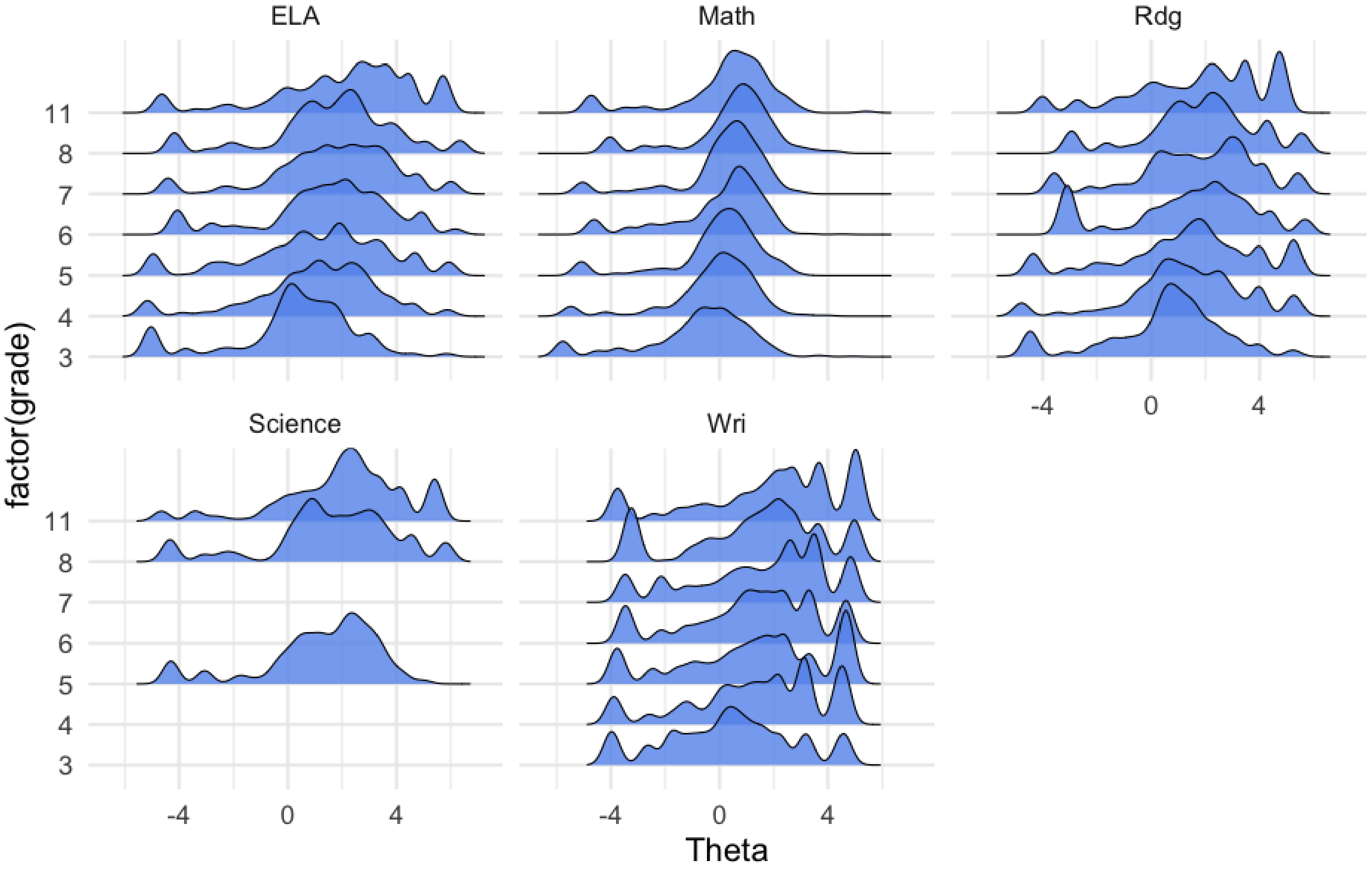
Final product

- In this case, we basically have a tidy data frame already!
- We've reduced our problem from 31 files to a single file

```
d
```

```
## # A tibble: 15,945 × 25
##        ssid grade  year content testeventid asmtprmrydsbltycd asmtscndry
##       <dbl> <int> <int> <chr>         <dbl>             <dbl>
##  1  9466908    11    11 ELA          148933                 0
##  2  7683685    11    11 ELA          147875                10
##  3  9025693    11    11 ELA          143699                40
##  4 10099824    11    11 ELA          143962                82
##  5 18886078    11    11 ELA          150680                10
##  6 10606750    11    11 ELA          144583                80
##  7 10541306    11    11 ELA          145204                50
##  8  7632967    11    11 ELA          148926                10
##  9  7661118    11    11 ELA          148893                50
## 10 10547177    11    11 ELA          144583                82
## # … with 15,935 more rows, and 17 more variables: Theta <dbl>, Status <c
## #   Count <dbl>, RawScore <dbl>, SE <dbl>, Infit <dbl>, Infit_Z <dbl>, O
## #   Outfit_Z <dbl>, Displacement <dbl>, PointMeasureCorr <dbl>, Weight <
## #   ObservMatch <dbl>, ExpectMatch <dbl>, PointMeasureExpected <dbl>, RM
```

Quick look at distributions



Summary stats

```
d %>%
    group_by(grade, content, asmtprmrydsbltycd) %>%
    summarize(mean = mean(Theta)) %>%
    pivot_wider(names_from = content,
                values_from = mean)
```

```
## # A tibble: 77 × 7
## # Groups:   grade [7]
##    grade asmtprmrydsbltycd        ELA        Math         Rdg        Wri
##    <int>             <dbl>      <dbl>       <dbl>      <dbl>      <dbl>
##  1     3                 0 -0.07361   -1.21055     1.010455   1.612308
##  2     3                10  0.3700416 -0.8182091   0.5184354  0.3206475
##  3     3                20 -0.06335   -1.2514      1.52      -0.5775
##  4     3                40 -1.877683  -3.56365    -1.761667  -0.7514286
##  5     3                50  0.9462857 -0.09186957  0.9791176  1.191481
##  6     3                60  0.840775   1.040375    2.181111   1.067
##  7     3                70 -1.104049  -1.517955   -0.8454839 -1.005625
##  8     3                74  0.996      0.0208375   0.6        1.2925
##  9     3                80 -0.144304  -0.5325596   0.6791667  0.2686301
## 10     3                82  0.3708244 -1.080988    0.5676650  0.3440741
## # … with 67 more rows
```

Backing up a bit

- What if we wanted only math files?

```
dir_ls(here::here("data", "pfiles_sim"), regexp = "Math")
```

```
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
## /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/data/p
```

Only Grade 5

You try

```r
g5_paths <- dir_ls(
  here::here("data", "pfiles_sim"),
  regexp = "g5"
)
```

03:00

The rest is the same

```r
g5 <- map_dfr(g5_paths, read_csv, .id = "file") %>%
  mutate(
      file = str_replace_all(
        file,
        here::here("data", "pfiles_sim"),
        ""
      )
    )
g5
```

```
## # A tibble: 2,632 × 23
##    file                   Entry   Theta Status Count RawScore     SE Inf
##    <chr>                  <dbl>   <dbl>  <dbl> <dbl>    <dbl>  <dbl> <db
##  1 /g5ELApfiles18_sim.csv   375  3.154       1    36       32 0.551   1.
##  2 /g5ELApfiles18_sim.csv   305  0.3662      1    36       16 0.3894  0.
##  3 /g5ELApfiles18_sim.csv   163 -4.9547     -1    36        0 1.8495  1
##  4 /g5ELApfiles18_sim.csv   524 -4.9547     -1    36        0 1.8495  1
##  5 /g5ELApfiles18_sim.csv    81  3.154       1    36       32 0.551   0.
##  6 /g5ELApfiles18_sim.csv   325  1.7156      1    36       25 0.3997  1.
##  7 /g5ELApfiles18_sim.csv   163  1.8786      1    36       26 0.4078  0.
##  8 /g5ELApfiles18_sim.csv   116  5.9323      0    36       36 1.8373  1
##  9 /g5ELApfiles18_sim.csv   273  1.4052      1    36       23 0.3891  1.
## 10 /g5ELApfiles18_sim.csv   202  1.8786      1    36       26 0.4078  0.
## # … with 2,622 more rows, and 14 more variables: Outfit <dbl>, Outfit_Z
```

Base equivalents

```r
list.files(here::here("data", "pfiles_sim"))
```

```
##  [1] "g11ELApfiles18_sim.csv"     "g11Mathpfiles18_sim.csv"
##  [3] "g11Rdgpfiles18_sim.csv"     "g11Sciencepfiles18_sim.csv"
##  [5] "g11Wripfiles18_sim.csv"     "g3ELApfiles18_sim.csv"
##  [7] "g3Mathpfiles18_sim.csv"     "g3Rdgpfiles18_sim.csv"
##  [9] "g3Wripfiles18_sim.csv"      "g4ELApfiles18_sim.csv"
## [11] "g4Mathpfiles18_sim.csv"     "g4Rdgpfiles18_sim.csv"
## [13] "g4Wripfiles18_sim.csv"      "g5ELApfiles18_sim.csv"
## [15] "g5Mathpfiles18_sim.csv"     "g5Rdgpfiles18_sim.csv"
## [17] "g5Sciencepfiles18_sim.csv"  "g5Wripfiles18_sim.csv"
## [19] "g6ELApfiles18_sim.csv"      "g6Mathpfiles18_sim.csv"
## [21] "g6Rdgpfiles18_sim.csv"      "g6Wripfiles18_sim.csv"
## [23] "g7ELApfiles18_sim.csv"      "g7Mathpfiles18_sim.csv"
## [25] "g7Rdgpfiles18_sim.csv"      "g7Wripfiles18_sim.csv"
## [27] "g8ELApfiles18_sim.csv"      "g8Mathpfiles18_sim.csv"
## [29] "g8Rdgpfiles18_sim.csv"      "g8Sciencepfiles18_sim.csv"
## [31] "g8Wripfiles18_sim.csv"
```

Full path

```r
list.files(here::here("data", "pfiles_sim"), full.names = TRUE)
```

```
##  [1] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [2] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [3] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [4] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [5] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [6] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [7] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [8] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [9] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [10] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [11] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [12] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [13] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [14] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [15] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [16] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [17] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [18] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [19] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [20] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [21] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [22] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [23] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [24] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
```

Only csvs

```r
list.files(here::here("data", "pfiles_sim"),
           full.names = TRUE,
           pattern = "*.csv")
```

```
##  [1] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [2] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [3] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [4] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [5] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [6] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [7] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [8] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
##  [9] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [10] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [11] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [12] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [13] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [14] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [15] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [16] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [17] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [18] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [19] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [20] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
## [21] "/Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/
```

Why not use base?

- We could, but `{fs}` plays a little nicer with `{purrr}`

```
files <- list.files(
  here::here("data", "pfiles_sim"),
  pattern = "*.csv"
)
batch3 <- map_dfr(files, read_csv, .id = "file")
```

```
## Error: 'g11ELApfiles18_sim.csv' does not exist in current working direct
```

- Need to return full names

```
files
```

```
##  [1] "g11ELApfiles18_sim.csv"     "g11Mathpfiles18_sim.csv"
##  [3] "g11Rdgpfiles18_sim.csv"     "g11Sciencepfiles18_sim.csv"
##  [5] "g11Wripfiles18_sim.csv"     "g3ELApfiles18_sim.csv"
##  [7] "g3Mathpfiles18_sim.csv"     "g3Rdgpfiles18_sim.csv"
##  [9] "g3Wripfiles18_sim.csv"      "g4ELApfiles18_sim.csv"
## [11] "g4Mathpfiles18_sim.csv"     "g4Rdgpfiles18_sim.csv"
## [13] "g4Wripfiles18_sim.csv"      "g5ELApfiles18_sim.csv"
```

Try again

```r
files <- list.files(here::here("data", "pfiles_sim"),
                    pattern = "*.csv",
                    full.names = TRUE)
batch3 <- map_dfr(files, read_csv, .id = "file")
batch3
```

```
## # A tibble: 15,945 × 23
##    file  Entry   Theta Status Count RawScore     SE Infit Infit_Z Outfit
##    <chr> <dbl>   <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>   <dbl>  <dbl>
##  1 1       123  1.2687      1    36       23 0.3713  0.93   -0.34   0.82
##  2 1        88  1.5541      1    36       25 0.3852  0.95   -0.37   0.81
##  3 1       105  3.2773      1    36       33 0.6187  0.9    -0.04   1.63
##  4 1       153  4.4752      1    36       35 1.0234  0.93    0.23   0.35
##  5 1       437  2.6655      1    36       31 0.5008  0.92   -0.18   0.88
##  6 1       307  5.7137      0    36       36 1.8371  1       0      1
##  7 1       305  3.7326      1    36       34 0.7408  1.06    0.31   0.86
##  8 1        42  0.609       1    36       18 0.36    1.55    2.56   1.74
##  9 1        59 -2.623       1    36        3 1.0344  0.85    0.06   0.17
## 10 1       304  5.7137      0    36       36 1.8371  1       0      1
## # … with 15,935 more rows, and 12 more variables: Displacement <dbl>,
## #   PointMeasureCorr <dbl>, Weight <dbl>, ObservMatch <dbl>, ExpectMatch
## #   PointMeasureExpected <dbl>, RMSR <dbl>, WMLE <dbl>, testeventid <dbl
## #   asmtprmrydsbltycd <dbl>, asmtscndrydsbltycd <dbl>
```

indexes

- The prior example gave us indexes, rather than the file path. Why?

No names

```
names(files)
```

```
## NULL
```

- We **need** the file path! An index isn't nearly as useful.

Base method that works

```r
files <- list.files(here::here("data", "pfiles_sim"),
                    pattern = "*.csv",
                    full.names = TRUE)
files <- setNames(files, files)

batch4 <- map_dfr(files, read_csv, .id = "file")
batch4
```

```
## # A tibble: 15,945 × 23
## # … with 15,935 more rows, and 23 more variables: file <chr>, Entry <dbl
## #   Status <dbl>, Count <dbl>, RawScore <dbl>, SE <dbl>, Infit <dbl>, In
## #   Outfit <dbl>, Outfit_Z <dbl>, Displacement <dbl>, PointMeasureCorr <
## #   Weight <dbl>, ObservMatch <dbl>, ExpectMatch <dbl>, PointMeasureExpe
## #   RMSR <dbl>, WMLE <dbl>, testeventid <dbl>, ssid <dbl>, asmtprmrydsbl
## #   asmtscndrydsbltycd <dbl>
```

My recommendation

- If you're working interactively, no reason not to use `{fs}`
- If you are building **functions** that take paths, might be worth considering skipping the dependency

Note

I am **not** saying skip it, but rather that you should **consider** whether it is really needed or not.

List columns

Comparing models

Let's say we wanted to fit/compare a set of models for each content area

1. `lm(Theta ~ asmtprmrydsbltycd)`
2. `lm(Theta ~ asmtprmrydsbltycd + asmtscndrydsbltycd)`
3. `lm(Theta ~ asmtprmrydsbltycd * asmtscndrydsbltycd)`

Data pre-processing

- The disability variables are stored as numbers, we need them as factors
- We'll make the names easier in the process

```
d <- d %>%
    mutate(primary = as.factor(asmtprmrydsbltycd),
           secondary = as.factor(asmtscndrydsbltycd))
```

If you're interested in what the specific codes refer to, see here.

Split the data

The base method we've been using...

```
splt_content <- split(d, d$content)
str(splt_content)
```

```
## List of 5
##  $ ELA    : tibble [3,627 × 27] (S3: tbl_df/tbl/data.frame)
##   ..$ ssid                : num [1:3627] 9466908 7683685 9025693 1009982
##   ..$ grade               : int [1:3627] 11 11 11 11 11 11 11 11 11 11 .
##   ..$ year                : int [1:3627] 11 11 11 11 11 11 11 11 11 11 .
##   ..$ content             : chr [1:3627] "ELA" "ELA" "ELA" "ELA" ...
##   ..$ testeventid         : num [1:3627] 148933 147875 143699 143962 150
##   ..$ asmtprmrydsbltycd   : num [1:3627] 0 10 40 82 10 80 50 10 50 82 ..
##   ..$ asmtscndrydsbltycd  : num [1:3627] 0 0 20 0 0 80 0 0 0 0 ...
##   ..$ Entry               : num [1:3627] 123 88 105 153 437 307 305 42 5
##   ..$ Theta               : num [1:3627] 1.27 1.55 3.28 4.48 2.67 ...
##   ..$ Status              : num [1:3627] 1 1 1 1 1 0 1 1 1 0 ...
##   ..$ Count               : num [1:3627] 36 36 36 36 36 36 36 36 36 36 .
##   ..$ RawScore            : num [1:3627] 23 25 33 35 31 36 34 18 3 36 ..
##   ..$ SE                  : num [1:3627] 0.371 0.385 0.619 1.023 0.501 .
##   ..$ Infit               : num [1:3627] 0.93 0.95 0.9 0.93 0.92 1 1.06
##   ..$ Infit_Z             : num [1:3627] -0.34 -0.37 -0.04 0.23 -0.18 0
##   ..$ Outfit              : num [1:3627] 0.82 0.81 1.63 0.35 0.88 1 0.86
##   ..$ Outfit_Z            : num [1:3627] -0.62 -0.56 1.03 -0.16 -0.12 0
```

We could use this method

```
m1 <- map(
  splt_content,
  ~lm(Theta ~ asmtprmrydsbltycd, data = .x)
)

m2 <- map(
  splt_content,
  ~lm(Theta ~ asmtprmrydsbltycd + asmtscndrydsbltycd,
      data = .x)
)

m3 <- map(
  splt_content,
  ~lm(Theta ~ asmtprmrydsbltycd * asmtscndrydsbltycd,
      data = .x)
)
```

- Then conduct tests to see which model fit better, etc.

Alternative

- Create a data frame with a list column

```
by_content <- d %>%
  group_by(content) %>%
  nest()
by_content
```

```
## # A tibble: 5 × 2
## # Groups:   content [5]
##   content data
##   <chr>   <list>
## 1 ELA     <tibble [3,627 × 26]>
## 2 Math    <tibble [3,629 × 26]>
## 3 Rdg     <tibble [3,627 × 26]>
## 4 Science <tibble [1,435 × 26]>
## 5 Wri     <tibble [3,627 × 26]>
```

What's going on here?

```
str(by_content$data)
```

```
## List of 5
##  $ : tibble [3,627 × 26] (S3: tbl_df/tbl/data.frame)
##   ..$ ssid                : num [1:3627] 9466908 7683685 9025693 1009982
##   ..$ grade               : int [1:3627] 11 11 11 11 11 11 11 11 11 11 .
##   ..$ year                : int [1:3627] 11 11 11 11 11 11 11 11 11 11 .
##   ..$ testeventid         : num [1:3627] 148933 147875 143699 143962 150
##   ..$ asmtprmrydsbltycd   : num [1:3627] 0 10 40 82 10 80 50 10 50 82 ..
##   ..$ asmtscndrydsbltycd  : num [1:3627] 0 0 20 0 0 80 0 0 0 0 ...
##   ..$ Entry               : num [1:3627] 123 88 105 153 437 307 305 42 5
##   ..$ Theta               : num [1:3627] 1.27 1.55 3.28 4.48 2.67 ...
##   ..$ Status              : num [1:3627] 1 1 1 1 1 0 1 1 1 0 ...
##   ..$ Count               : num [1:3627] 36 36 36 36 36 36 36 36 36 36 .
##   ..$ RawScore            : num [1:3627] 23 25 33 35 31 36 34 18 3 36 ..
##   ..$ SE                  : num [1:3627] 0.371 0.385 0.619 1.023 0.501 .
##   ..$ Infit               : num [1:3627] 0.93 0.95 0.9 0.93 0.92 1 1.06
##   ..$ Infit_Z             : num [1:3627] -0.34 -0.37 -0.04 0.23 -0.18 0
##   ..$ Outfit              : num [1:3627] 0.82 0.81 1.63 0.35 0.88 1 0.86
##   ..$ Outfit_Z            : num [1:3627] -0.62 -0.56 1.03 -0.16 -0.12 0
##   ..$ Displacement        : num [1:3627] 0.0018 0.0019 0.0022 0.0023 0.0
##   ..$ PointMeasureCorr    : num [1:3627] 0.42 0.42 0.3 0.27 0.31 0 0.14
##   ..$ Weight              : num [1:3627] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ObservMatch         : num [1:3627] 75 80.6 91.7 97.2 86.1 100 94.4
##   ..$ ExpectMatch         : num [1:3627] 68.3 72 91.7 97.2 86.1 100 94.4
##   ..$ PointMeasureExpected: num [1:3627] 0.35 0.33 0.2 0.12 0.25 0 0.17
```

Explore a bit

```
map_dbl(by_content$data, nrow)
```

```
## [1] 3627 3629 3627 1435 3627
```

```
map_dbl(by_content$data, ncol)
```

```
## [1] 26 26 26 26 26
```

```
map_dbl(by_content$data, ~mean(.x$Theta))
```

```
## [1]  1.28001056 -0.06683086  1.37068376  1.57850321  1.26090709
```

It's a data frame!

We can add these summaries if we want

```
by_content %>%
    mutate(n = map_dbl(data, nrow))
```

```
## # A tibble: 5 × 3
## # Groups:   content [5]
##   content data                      n
##   <chr>   <list>                <dbl>
## 1 ELA     <tibble [3,627 × 26]>  3627
## 2 Math    <tibble [3,629 × 26]>  3629
## 3 Rdg     <tibble [3,627 × 26]>  3627
## 4 Science <tibble [1,435 × 26]>  1435
## 5 Wri     <tibble [3,627 × 26]>  3627
```

map_*

- Note on the previous example we used `map_dbl` and we got a vector in return.

- What would happen if we just used `map`?

```r
by_content %>%
    mutate(n = map(data, nrow))
```

```
## # A tibble: 5 × 3
## # Groups:   content [5]
##   content data                  n
##   <chr>   <list>                <list>
## 1 ELA     <tibble [3,627 × 26]> <int [1]>
## 2 Math    <tibble [3,629 × 26]> <int [1]>
## 3 Rdg     <tibble [3,627 × 26]> <int [1]>
## 4 Science <tibble [1,435 × 26]> <int [1]>
## 5 Wri     <tibble [3,627 × 26]> <int [1]>
```

Let's fit a model!

```r
by_content %>%
    mutate(m1 = map(data, ~lm(Theta ~ primary, data = .x)))
```

```
## # A tibble: 5 × 3
## # Groups:   content [5]
##   content data                    m1
##   <chr>   <list>                  <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>
## 2 Math    <tibble [3,629 × 26]> <lm>
## 3 Rdg     <tibble [3,627 × 26]> <lm>
## 4 Science <tibble [1,435 × 26]> <lm>
## 5 Wri     <tibble [3,627 × 26]> <lm>
```

Extract the coefficients

```
by_content %>%
  mutate(
    m1 = map(data, ~lm(Theta ~ primary, data = .x)),
    coefs = map(m1, coef)
)
```

```
## # A tibble: 5 × 4
## # Groups:   content [5]
##   content data                  m1     coefs
##   <chr>   <list>                <list> <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>   <dbl [11]>
## 2 Math    <tibble [3,629 × 26]> <lm>   <dbl [12]>
## 3 Rdg     <tibble [3,627 × 26]> <lm>   <dbl [11]>
## 4 Science <tibble [1,435 × 26]> <lm>   <dbl [12]>
## 5 Wri     <tibble [3,627 × 26]> <lm>   <dbl [12]>
```

Challenge

- Continue with the above, but output a data frame with three columns: `content`, `intercept`, and `TBI` (which is code 74).

- In other words, output the mean score for students who were coded as not having a disability (code 0), along with students coded as having TBI.

04:00

```
by_content %>%
  mutate(
    m1 = map(data, ~lm(Theta ~ primary, data = .x)),
    coefs = map(m1, coef),
    no_disab = map_dbl(coefs, 1),
    tbi = no_disab + map_dbl(coefs, "primary74")
  ) %>%
  select(content, no_disab, tbi)
```

```
## # A tibble: 5 × 3
## # Groups:   content [5]
##   content   no_disab       tbi
##   <chr>        <dbl>     <dbl>
## 1 ELA      0.9322336 0.1674462
## 2 Math    -0.1587907 0.1910821
## 3 Rdg      1.363101  1.629048
## 4 Science  1.491319  2.790971
## 5 Wri      1.571441  1.167429
```

Note – I wouldn't have neccesarily expected you to add `no_disab` to the TBI coefficient.

Compare models

- Back to our original task – fit all three models

You try first

1. `lm(Theta ~ primary)`
2. `lm(Theta ~ primary + secondary)`
3. `lm(Theta ~ primary + secondary + primary:secondary)`

04:00

Model fits

```
mods <- by_content %>%
  mutate(
    m1 = map(data, ~lm(Theta ~ primary, data = .x)),
    m2 = map(data, ~lm(Theta ~ primary + secondary, data = .x)),
    m3 = map(data, ~lm(Theta ~ primary * secondary, data = .x))
)
mods
```

```
## # A tibble: 5 × 5
## # Groups:   content [5]
##   content data                     m1     m2     m3    
##   <chr>   <list>                   <list> <list> <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>   <lm>   <lm>  
## 2 Math    <tibble [3,629 × 26]> <lm>   <lm>   <lm>  
## 3 Rdg     <tibble [3,627 × 26]> <lm>   <lm>   <lm>  
## 4 Science <tibble [1,435 × 26]> <lm>   <lm>   <lm>  
## 5 Wri     <tibble [3,627 × 26]> <lm>   <lm>   <lm>  
```

Brief foray into parallel iterations

The `stats::anova` function can compare the fit of two models

Pop Quiz

How would we extract just ELA model 1 and 2?

```
mods$m1[[1]]
```

```
##
## Call:
## lm(formula = Theta ~ primary, data = .x)
##
## Coefficients:
## (Intercept)    primary10    primary20    primary40    primary50    prima
##     0.93223      0.38570     -0.03168     -1.84434      1.17372      0.7
```

```
mods$m2[[1]]
```

```
##
## Call:
## lm(formula = Theta ~ primary + second
##
## Coefficients:
## (Intercept)    primary10    primary20
##      1.0043       0.4285       0.28
```

Which fits better?

```r
compare <- anova(mods$m1[[1]], mods$m2[[1]])
compare
```

```
## Analysis of Variance Table
##
## Model 1: Theta ~ primary
## Model 2: Theta ~ primary + secondary
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1   3616 20905
## 2   3605 20100 11    804.26 13.113 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

map2

- Works the same as `map` but iterates over two vectors concurrently

- Let's compare model 1 and 2

```
mods %>%
  mutate(comp12 = map2(m1, m2, anova))
```

```
## # A tibble: 5 × 6
## # Groups:   content [5]
##   content data                   m1     m2     m3     comp12
##   <chr>   <list>                 <list> <list> <list> <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]>
## 2 Math    <tibble [3,629 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]>
## 3 Rdg     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]>
## 4 Science <tibble [1,435 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]>
## 5 Wri     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]>
```

Perhaps not terrifically helpful

Back to our anova object

- Can we pull out useful things?

```
str(compare)
```

```
## Classes 'anova' and 'data.frame':    2 obs. of  6 variables:
##  $ Res.Df    : num  3616 3605
##  $ RSS       : num  20905 20100
##  $ Df        : num  NA 11
##  $ Sum of Sq: num  NA 804
##  $ F         : num  NA 13.1
##  $ Pr(>F)    : num  NA 7.66e-25
##  - attr(*, "heading")= chr [1:2] "Analysis of Variance Table\n" "Model 1
```

Try pulling out the p value

Extract *p* value

- *Note – I'd recommend looking at more than just a p–value, but I do think this is useful for a quick glance*

```
compare$`Pr(>F)`
```

```
## [1]           NA 7.663566e-25
```

```
compare[["Pr(>F)"]]
```

```
## [1]           NA 7.663566e-25
```

```
compare$`Pr(>F)`[2]
```

```
## [1] 7.663566e-25
```

```
compare[["Pr(>F)"]][2]
```

```
## [1] 7.663566e-25
```

All p-values

Note – this is probably the most compact syntax, but that doesn't mean it's the most clear

```
mods %>%
  mutate(comp12 = map2(m1, m2, anova),
         p12 = map_dbl(comp12, list("Pr(>F)", 2)))
```

```
## # A tibble: 5 × 7
## # Groups:   content [5]
##   content data                 m1     m2     m3     comp12
##   <chr>   <list>               <list> <list> <list> <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 7.6
## 2 Math    <tibble [3,629 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 1.7
## 3 Rdg     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 1.5
## 4 Science <tibble [1,435 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 4.6
## 5 Wri     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 5.7
```

Slight alternative

- Write a function that pulls the p–value from model comparison objects

```r
extract_p <- function(anova_ob) {
  anova_ob[["Pr(>F)"]][2]
}
```

- Loop this function through the anova objects

```
mods %>%
  mutate(comp12 = map2(m1, m2, anova),
         p12 = map_dbl(comp12, extract_p))
```

```
## # A tibble: 5 × 7
## # Groups:   content [5]
##   content data                     m1     m2     m3     comp12
##   <chr>   <list>                   <list> <list> <list> <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 7.6
## 2 Math    <tibble [3,629 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 1.7
## 3 Rdg     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 1.5
## 4 Science <tibble [1,435 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 4.6
## 5 Wri     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 5.7
```

Brief sidetrack

We can also create the function using `purrr::compose()`.

Example

Create a centering function (which subtracts the mean from each ob)

```
center <- compose(~.x - mean(.x, na.rm = TRUE))
```

Use `~` and `.x`, just like with the `map()` functions.

Test it out

```r
library(palmerpenguins)

penguins$bill_length_mm %>%
  head()
```

```
## [1] 39.1 39.5 40.3   NA 36.7 39.3
```

```r
penguins$bill_length_mm %>%
  center() %>%
  head()
```

```
## [1] -4.82193 -4.42193 -3.62193       NA -7.22193 -4.62193
```

```r
penguins$bill_length_mm %>%
  center() %>%
  mean(na.rm = TRUE) %>%
  round()
```

```
## [1] 0
```

Compose a p-val extractor

```
p <- compose(~.x[["Pr(>F)"]][2])
```

Use this instead

```
mods %>%
  mutate(
    comp12 = map2(m1, m2, anova),
    p12 = map_dbl(comp12, p)
  )
```

```
## # A tibble: 5 × 7
## # Groups:   content [5]
##   content data                  m1     m2     m3     comp12
##   <chr>   <list>                <list> <list> <list> <list>
## 1 ELA     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 7.6
## 2 Math    <tibble [3,629 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 1.7
## 3 Rdg     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 1.5
## 4 Science <tibble [1,435 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 4.6
## 5 Wri     <tibble [3,627 × 26]> <lm>   <lm>   <lm>   <anova [2 × 6]> 5.7
```

Functions

This was a quick intro – don't worry if it doesn't really make sense yet. We'll talk about them (a lot) more in the coming weeks.

An alternative

Conducting operations by row

Operations by row

The `dplyr::rowwise()` function fundamentally changes the way a `tibble()` behaves

```
df <- tibble(name = c("Me", "You"), x = 1:2, y = 3:4, z = 5:6)
```

```
df %>%
  mutate(m = mean(c(x, y, z)))
```

```
## # A tibble: 2 × 5
##   name      x     y     z     m
##   <chr> <int> <int> <int> <dbl>
## 1 Me        1     3     5   3.5
## 2 You       2     4     6   3.5
```

```
df %>%
  rowwise() %>%
  mutate(m = mean(c(x, y, z)))
```

```
## # A tibble: 2 × 5
## # Rowwise:
##   name      x     y     z     m
##   <chr> <int> <int> <int> <dbl>
## 1 Me        1     3     5     3
## 2 You       2     4     6     4
```

Add a group & summarize

```
df %>%
  rowwise(name) %>%
  summarize(m = mean(c(x, y, z)))
```

```
## # A tibble: 2 × 2
## # Groups:   name [2]
##   name      m
##   <chr> <dbl>
## 1 Me        3
## 2 You       4
```

List columns

If you apply rowwise operation with a list column, you don't have to loop

```
df <- tibble(var = list(1, 2:3, 4:6))
```

```
df %>%
  mutate(
    lngth = map_int(var, lengt
  )
```

```
## # A tibble: 3 × 2
##   var       lngth
##   <list>    <int>
## 1 <dbl [1]>     1
## 2 <int [2]>     2
## 3 <int [3]>     3
```

```
df %>%
  rowwise() %>%
  mutate(lnght = length(var))
```

```
## # A tibble: 3 × 2
## # Rowwise:
##   var       lnght
##   <list>    <int>
## 1 <dbl [1]>     1
## 2 <int [2]>     2
## 3 <int [3]>     3
```

Creating list columns

You can use the `dplyr::nest_by()` function to create a list column for each group, *and* convert it to a rowwise data frame.

```r
d %>%
  nest_by(content)
```

```
## # A tibble: 5 × 2
## # Rowwise:  content
##   content                data
##   <chr>   <list<tibble[,26]>>
## 1 ELA            [3,627 × 26]
## 2 Math           [3,629 × 26]
## 3 Rdg            [3,627 × 26]
## 4 Science        [1,435 × 26]
## 5 Wri            [3,627 × 26]
```

Challenge

Given what we just learned, can you fit a model of the form `Theta ~ primary` to each content area (i.e., *not* using **{purrr}**)?

Wrap it in `list()` (should suggest this in the error reporting if you don't)

```
d %>%
  nest_by(content) %>%
  mutate(m1 = list(lm(Theta ~ primary, data = data)))
```

```
## # A tibble: 5 × 3
## # Rowwise:  content
##   content                   data m1
##   <chr>   <list<tibble[,26]>> <list>
## 1 ELA              [3,627 × 26] <lm>
## 2 Math             [3,629 × 26] <lm>
## 3 Rdg              [3,627 × 26] <lm>
## 4 Science          [1,435 × 26] <lm>
## 5 Wri              [3,627 × 26] <lm>
```

02:00

Challenge 2

Can you extend it further and extract the coefficients with `coef`? What about creating a new column that has the intercept values?

```
d %>%
  nest_by(content) %>%
  mutate(m1 = list(lm(Theta ~ primary, data = data)),
         coefs = list(coef(m1)))
```

```
## # A tibble: 5 × 4
## # Rowwise:  content
##   content                data m1     coefs
##   <chr>   <list<tibble[,26]>> <list> <list>
## 1 ELA             [3,627 × 26] <lm>   <dbl [11]>
## 2 Math            [3,629 × 26] <lm>   <dbl [12]>
## 3 Rdg             [3,627 × 26] <lm>   <dbl [11]>
## 4 Science         [1,435 × 26] <lm>   <dbl [12]>
## 5 Wri             [3,627 × 26] <lm>   <dbl [12]>
```

02:00

Return atomic vectors

```r
d %>%
  nest_by(content) %>%
  mutate(m1 = list(lm(Theta ~ primary, data = data)),
         intercept = coef(m1)[1])
```

```
## # A tibble: 5 × 4
## # Rowwise:  content
##   content                data m1        intercept
##   <chr>   <list<tibble[,26]>> <list>        <dbl>
## 1 ELA            [3,627 × 26] <lm>      0.9322336
## 2 Math           [3,629 × 26] <lm>     -0.1587907
## 3 Rdg            [3,627 × 26] <lm>      1.363101
## 4 Science        [1,435 × 26] <lm>      1.491319
## 5 Wri            [3,627 × 26] <lm>      1.571441
```

Fit all models

The below gets us the same results we got before

```
mods2 <- d %>%
  nest_by(content) %>%
  mutate(
    m1 = list(lm(Theta ~ primary, data = data)),
    m2 = list(lm(Theta ~ primary + secondary, data = data)),
    m3 = list(lm(Theta ~ primary * secondary, data = data))
)
mods2
```

```
## # A tibble: 5 × 5
## # Rowwise:  content
##   content                 data m1     m2     m3    
##   <chr>   <list<tibble[,26]>> <list> <list> <list>
## 1 ELA             [3,627 × 26] <lm>   <lm>   <lm>  
## 2 Math            [3,629 × 26] <lm>   <lm>   <lm>  
## 3 Rdg             [3,627 × 26] <lm>   <lm>   <lm>  
## 4 Science         [1,435 × 26] <lm>   <lm>   <lm>  
## 5 Wri             [3,627 × 26] <lm>   <lm>   <lm>  
```

Look at all R^2

It's a normal data frame!

```
mods %>%
  pivot_longer(
    m1:m3,
    names_to = "model",
    values_to = "output"
)
```

```
## # A tibble: 15 × 4
## # Groups:   content [5]
##    content data                   model output
##    <chr>   <list>                 <chr> <list>
##  1 ELA     <tibble [3,627 × 26]> m1    <lm>
##  2 ELA     <tibble [3,627 × 26]> m2    <lm>
##  3 ELA     <tibble [3,627 × 26]> m3    <lm>
##  4 Math    <tibble [3,629 × 26]> m1    <lm>
##  5 Math    <tibble [3,629 × 26]> m2    <lm>
##  6 Math    <tibble [3,629 × 26]> m3    <lm>
##  7 Rdg     <tibble [3,627 × 26]> m1    <lm>
##  8 Rdg     <tibble [3,627 × 26]> m2    <lm>
##  9 Rdg     <tibble [3,627 × 26]> m3    <lm>
## 10 Science <tibble [1,435 × 26]> m1    <lm>
## 11 Science <tibble [1,435 × 26]> m2    <lm>
```

Extract all R^2

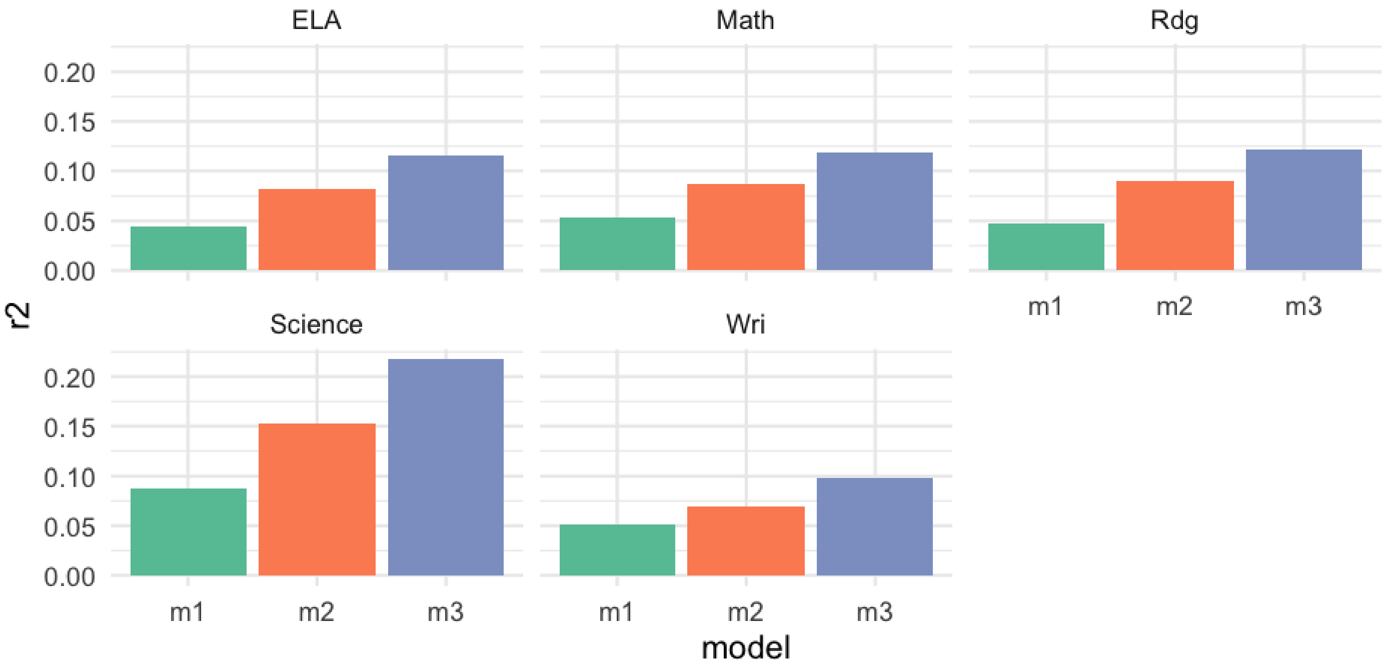
Note – might want to write a function here again

```
r2 <- mods %>%
  pivot_longer(
    m1:m3,
    names_to = "model",
    values_to = "output"
  ) %>%
  mutate(r2 = map_dbl(output, ~summary(.x)$r.squared))
r2
```

```
## # A tibble: 15 × 5
## # Groups:   content [5]
##    content data                  model output      r2
##    <chr>   <list>                <chr> <list>   <dbl>
##  1 ELA     <tibble [3,627 × 26]> m1    <lm>   0.04517421
##  2 ELA     <tibble [3,627 × 26]> m2    <lm>   0.08190917
##  3 ELA     <tibble [3,627 × 26]> m3    <lm>   0.1161187
##  4 Math    <tibble [3,629 × 26]> m1    <lm>   0.05326550
##  5 Math    <tibble [3,629 × 26]> m2    <lm>   0.08675264
##  6 Math    <tibble [3,629 × 26]> m3    <lm>   0.1185931
##  7 Rdg     <tibble [3,627 × 26]> m1    <lm>   0.04805713
##  8 Rdg     <tibble [3,627 × 26]> m2    <lm>   0.08926212
##  9 Rdg     <tibble [3,627 × 26]> m3    <lm>   0.1217497
```

Plot

```
ggplot(r2, aes(model, r2)) +
    geom_col(aes(fill = model)) +
    facet_wrap(~content) +
    guides(fill = "none") +
    scale_fill_brewer(palette = "Set2")
```



Unnesting

- Sometimes you just want to `unnest`
- Imagine we want to plot the coefficients by model... how?
- `broom::tidy()` => `tidyr::unnest()`

Tidy

```r
mods %>%
    pivot_longer(
      m1:m3,
      names_to = "model",
      values_to = "output"
  ) %>%
    mutate(tidied = map(output, broom::tidy))
```

```
## # A tibble: 15 × 5
## # Groups:   content [5]
##    content data                  model output tidied
##    <chr>   <list>                <chr> <list> <list>
##  1 ELA     <tibble [3,627 × 26]> m1    <lm>   <tibble [11 × 5]>
##  2 ELA     <tibble [3,627 × 26]> m2    <lm>   <tibble [22 × 5]>
##  3 ELA     <tibble [3,627 × 26]> m3    <lm>   <tibble [132 × 5]>
##  4 Math    <tibble [3,629 × 26]> m1    <lm>   <tibble [12 × 5]>
##  5 Math    <tibble [3,629 × 26]> m2    <lm>   <tibble [23 × 5]>
##  6 Math    <tibble [3,629 × 26]> m3    <lm>   <tibble [144 × 5]>
##  7 Rdg     <tibble [3,627 × 26]> m1    <lm>   <tibble [11 × 5]>
##  8 Rdg     <tibble [3,627 × 26]> m2    <lm>   <tibble [22 × 5]>
##  9 Rdg     <tibble [3,627 × 26]> m3    <lm>   <tibble [132 × 5]>
## 10 Science <tibble [1,435 × 26]> m1    <lm>   <tibble [12 × 5]>
## 11 Science <tibble [1,435 × 26]> m2    <lm>   <tibble [22 × 5]>
## 12 Science <tibble [1,435 × 26]> m3    <lm>   <tibble [132 × 5]>
## 13 Wri     <tibble [3,627 × 26]> m1    <lm>   <tibble [12 × 5]>
```

Equivalently

```r
mods %>%
    pivot_longer(
      m1:m3,
      names_to = "model",
      values_to = "output"
  ) %>%
  rowwise() %>%
  mutate(tidied = list(broom::tidy(output)))
```

```
## # A tibble: 15 × 5
## # Rowwise:  content
##    content data                  model output tidied
##    <chr>   <list>                <chr> <list> <list>
##  1 ELA     <tibble [3,627 × 26]> m1    <lm>   <tibble [11 × 5]>
##  2 ELA     <tibble [3,627 × 26]> m2    <lm>   <tibble [22 × 5]>
##  3 ELA     <tibble [3,627 × 26]> m3    <lm>   <tibble [132 × 5]>
##  4 Math    <tibble [3,629 × 26]> m1    <lm>   <tibble [12 × 5]>
##  5 Math    <tibble [3,629 × 26]> m2    <lm>   <tibble [23 × 5]>
##  6 Math    <tibble [3,629 × 26]> m3    <lm>   <tibble [144 × 5]>
##  7 Rdg     <tibble [3,627 × 26]> m1    <lm>   <tibble [11 × 5]>
##  8 Rdg     <tibble [3,627 × 26]> m2    <lm>   <tibble [22 × 5]>
##  9 Rdg     <tibble [3,627 × 26]> m3    <lm>   <tibble [132 × 5]>
## 10 Science <tibble [1,435 × 26]> m1    <lm>   <tibble [12 × 5]>
## 11 Science <tibble [1,435 × 26]> m2    <lm>   <tibble [22 × 5]>
## 12 Science <tibble [1,435 × 26]> m3    <lm>   <tibble [132 × 5]>
```

Select and unnest

```
tidied <- mods %>%
    gather(model, output, m1:m3) %>%
    mutate(tidied = map(output, broom::tidy)) %>%
    select(content, model, tidied) %>%
    unnest(tidied)
tidied
```

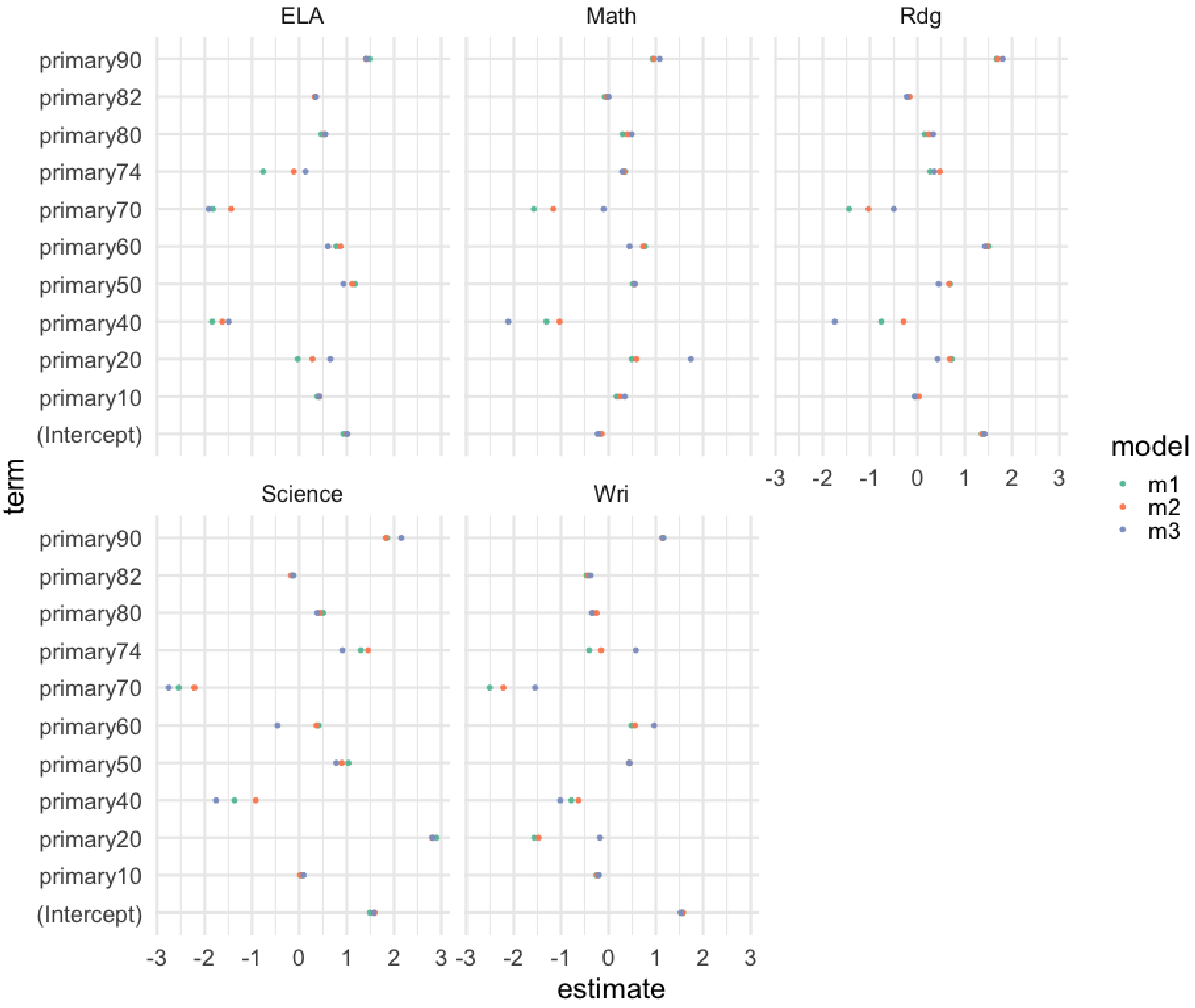
```
## # A tibble: 841 × 7
## # Groups:   content [5]
##    content model term           estimate std.error   statistic     p.val
##    <chr>   <chr> <chr>             <dbl>     <dbl>       <dbl>       <db
##  1 ELA     m1    (Intercept)  0.9322336  0.2150561  4.334839   1.498396e
##  2 ELA     m1    primary10    0.3856986  0.2242965  1.719593   8.559207e
##  3 ELA     m1    primary20   -0.03167527 0.7266436 -0.04359120 9.652327e
##  4 ELA     m1    primary40   -1.844343   0.5559031 -3.317741   9.164595e
##  5 ELA     m1    primary50    1.173722   0.2890447  4.060694   4.996391e
##  6 ELA     m1    primary60    0.7762539  0.3866313  2.007737   4.474555e
##  7 ELA     m1    primary70   -1.830257   0.3086128 -5.930595   3.301860e
##  8 ELA     m1    primary74   -0.7647874  0.5182670 -1.475663   1.401215e
##  9 ELA     m1    primary80    0.4676481  0.2428640  1.925556   5.423822e
## 10 ELA     m1    primary82    0.3382547  0.2267600  1.491686   1.358687e
## # … with 831 more rows
```

Plot

Lets look how the primary coefficients change

```r
to_plot <- names(coef(mods$m1[[1]]))

tidied %>%
  filter(term %in% to_plot) %>%
  ggplot(aes(estimate, term, color = model)) +
  geom_point() +
  scale_color_brewer(palette = "Set2") +
  facet_wrap(~content)
```


ELA
Math
Rdg
Science
Wri
primary90
primary82
primary80
primary74
primary70
primary60
primary50
primary40
primary20
primary10
(Intercept)
term
-3
-2
-1
0
1
2
3
estimate
model
m1
m2
m3

Last bit

- We've kind of been running the wrong models this whole time
- We forgot about grade!
- No problem, just change the grouping factor

By grade

```
by_grade_content <- d %>%
  group_by(content, grade) %>%
    nest()
by_grade_content
```

```
## # A tibble: 31 × 3
## # Groups:   content, grade [31]
##    grade content data              
##    <int> <chr>   <list>            
##  1    11 ELA     <tibble [453 × 25]>
##  2    11 Math    <tibble [460 × 25]>
##  3    11 Rdg     <tibble [453 × 25]>
##  4    11 Science <tibble [438 × 25]>
##  5    11 Wri     <tibble [453 × 25]>
##  6     3 ELA     <tibble [540 × 25]>
##  7     3 Math    <tibble [536 × 25]>
##  8     3 Rdg     <tibble [540 × 25]>
##  9     3 Wri     <tibble [540 × 25]>
## 10     4 ELA     <tibble [585 × 25]>
## # … with 21 more rows
```

Fit models

```
mods_grade <- by_grade_content %>%
  mutate(
    m1 = map(data, ~lm(Theta ~ primary, data = .x)),
    m2 = map(data, ~lm(Theta ~ primary + secondary,
                       data = .x)),
    m3 = map(data, ~lm(Theta ~ primary * secondary,
                       data = .x))
)
mods_grade
```

```
## # A tibble: 31 × 6
## # Groups:   content, grade [31]
##    grade content data                m1     m2     m3    
##    <int> <chr>   <list>              <list> <list> <list>
##  1    11 ELA     <tibble [453 × 25]> <lm>   <lm>   <lm>  
##  2    11 Math    <tibble [460 × 25]> <lm>   <lm>   <lm>  
##  3    11 Rdg     <tibble [453 × 25]> <lm>   <lm>   <lm>  
##  4    11 Science <tibble [438 × 25]> <lm>   <lm>   <lm>  
##  5    11 Wri     <tibble [453 × 25]> <lm>   <lm>   <lm>  
##  6     3 ELA     <tibble [540 × 25]> <lm>   <lm>   <lm>  
##  7     3 Math    <tibble [536 × 25]> <lm>   <lm>   <lm>  
##  8     3 Rdg     <tibble [540 × 25]> <lm>   <lm>   <lm>  
##  9     3 Wri     <tibble [540 × 25]> <lm>   <lm>   <lm>  
## 10     4 ELA     <tibble [585 × 25]> <lm>   <lm>   <lm>  
```

Look at R^2

```r
mods_grade %>%
    pivot_longer(
    m1:m3,
    names_to = "model",
    values_to = "output"
  ) %>%
    mutate(r2 = map_dbl(output, ~summary(.x)$r.squared))
```

```
## # A tibble: 93 × 6
## # Groups:   content, grade [31]
##    grade content data                model output         r2
##    <int> <chr>   <list>              <chr> <list>      <dbl>
##  1    11 ELA     <tibble [453 × 25]> m1    <lm>   0.03353818
##  2    11 ELA     <tibble [453 × 25]> m2    <lm>   0.1084394
##  3    11 ELA     <tibble [453 × 25]> m3    <lm>   0.1536891
##  4    11 Math    <tibble [460 × 25]> m1    <lm>   0.1886003
##  5    11 Math    <tibble [460 × 25]> m2    <lm>   0.3161226
##  6    11 Math    <tibble [460 × 25]> m3    <lm>   0.4046634
##  7    11 Rdg     <tibble [453 × 25]> m1    <lm>   0.02066316
##  8    11 Rdg     <tibble [453 × 25]> m2    <lm>   0.1820512
##  9    11 Rdg     <tibble [453 × 25]> m3    <lm>   0.2337721
## 10    11 Science <tibble [438 × 25]> m1    <lm>   0.1259080
## # … with 83 more rows
```

Plot

```r
mods_grade %>%
  pivot_longer(
    m1:m3,
    names_to = "model",
    values_to = "output"
  ) %>%
  mutate(r2 = map_dbl(output, ~summary(.x)$r.squared)) %>%
  ggplot(aes(model, r2)) +
  geom_col(aes(fill = model)) +
  facet_grid(grade ~ content) +
  guides(fill = "none") +
  scale_fill_brewer(palette = "Set2")
```

Summary

- List columns are really powerful and really flexible

- Also help you stay organized

- You can approach the problem either with **{purrr}** or `dplyr::rowwise()`.

    - **Important**: If you use `rowwise()`, remember to `ungroup()` when you want it to go back to being a normal data frame

    - I'm asking you to learn both – the row–wise approach might be a bit easier but is a little less general (only works with data frames)

In–class Midterm

Next time: Parallel iterations