List columns

Daniel Anderson

Week 4, Class 2

Agenda

- Review Lab 2
- Introduce list columns
- Contrast:
  - `group_by() %>% nest() %>% mutate() %>% map()` with
  - `nest_by() %>% summarize()`
- In–class midterm (last 20 minutes)

Learning objectives

- 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.

Review Lab 2

Setup

Please follow along

First import the data

```
library(tidyverse)
library(fs)
files <- dir_ls(here::here("data", "pfiles_sim"),
                glob = "*.csv")
d <- files %>%
    map_df(read_csv, .id = "file")
```

Parse file data

```
d <- d %>%
  mutate(
    file = str_replace_all(
      file,
      here::here("data", "pfiles_sim"),
      ""
    ),
    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(year),
    content = str_replace_all(
      file,
      "/g\\d?\\d(.+)pfiles.+",
      "\\1"
    )
)
```

Select variables

```
d <- d %>%
    select(ssid, grade, year, content, testeventid,
           asmtprmrydsbltycd, asmtscndrydsbltycd, Entry:WMLE)
```

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[,27] [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] 18 18 18 18 18 18 18 18 18 18 .
##   ..$ 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
##   ..$ Displacement        : num [1:3627] 0.0018 0.0019 0.0022 0.0023 0.0
```

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

What's going on here?

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

```
## List of 5
##  $ : tibble[,26] [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] 18 18 18 18 18 18 18 18 18 18 .
##   ..$ 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 x 3
## # Groups:   content [5]
##   content data                                  n
##   <chr>   <list>                            <dbl>
## 1 ELA     <tibble[,26] [3,627 × 26]>  3627
## 2 Math    <tibble[,26] [3,629 × 26]>  3629
## 3 Rdg     <tibble[,26] [3,627 × 26]>  3627
## 4 Science <tibble[,26] [1,435 × 26]>  1435
## 5 Wri     <tibble[,26] [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`?

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

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

Let's fit a model!

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

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

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

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

```
##
## Call:
## lm(formula = Theta ~ primary, data = .x)
##
## Coefficients:
## (Intercept)    primary10    primary20    primary40    primary50    prima
##     0.93223      0.38570     -0.03168     -0.84434      0.17372      0.7
```

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

```
##
## Call:
## lm(formula = Theta ~ primary + second
##
## Coefficients:
## (Intercept)    primary10    primary20
##     1.00143      0.42185      0.280
```

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

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

Slight alternative

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

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

- Loop this function through the anova objects

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

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

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 x 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 x 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 x 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(x = list(1, 2:3, 4:6))
```

```
df %>%
  mutate(
    l = map_int(x, length)
  )
```

```
## # A tibble: 3 x 2
##   x              l
##   <list>     <int>
## 1 <dbl [1]>      1
## 2 <int [2]>      2
## 3 <int [3]>      3
```

```
df %>%
  rowwise() %>%
  mutate(l = length(x))
```

```
## # A tibble: 3 x 2
## # Rowwise:
##   x              l
##   <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.

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

```
## # A tibble: 5 x 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 x 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 x 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

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

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

Extract all R^2

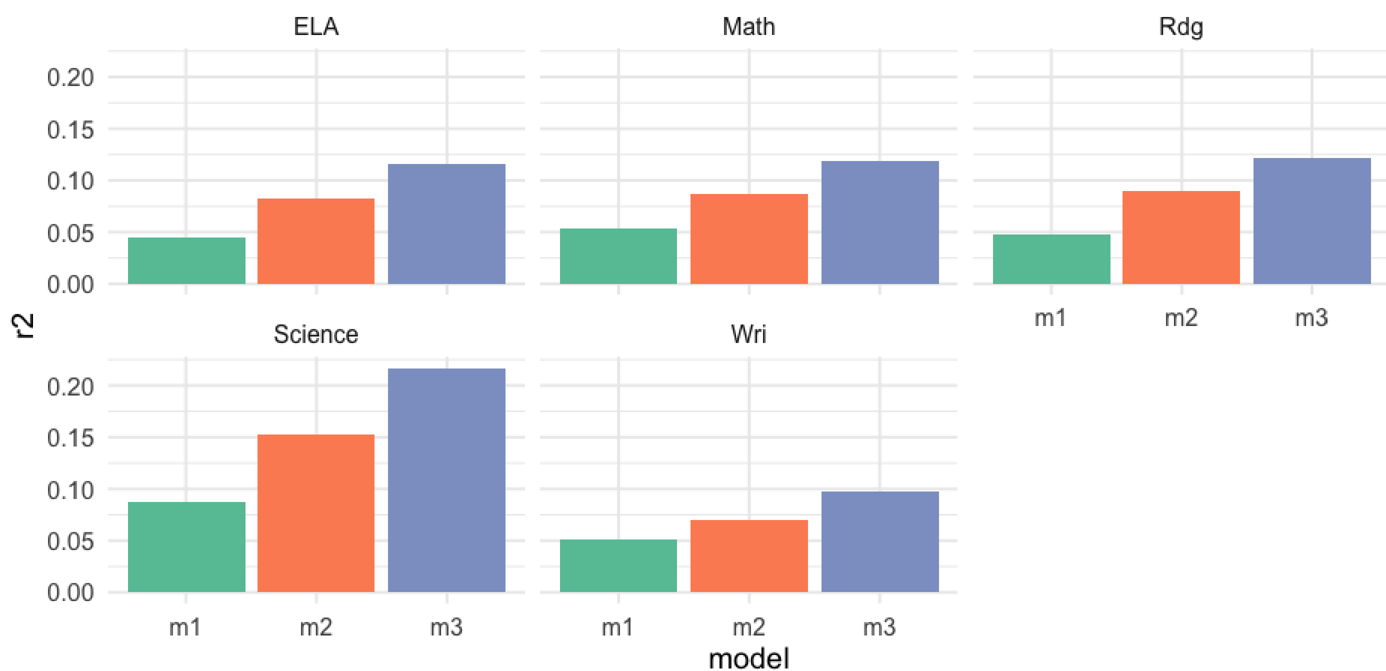
Note – might want to write a function here again

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

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

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

Equivalently

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

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

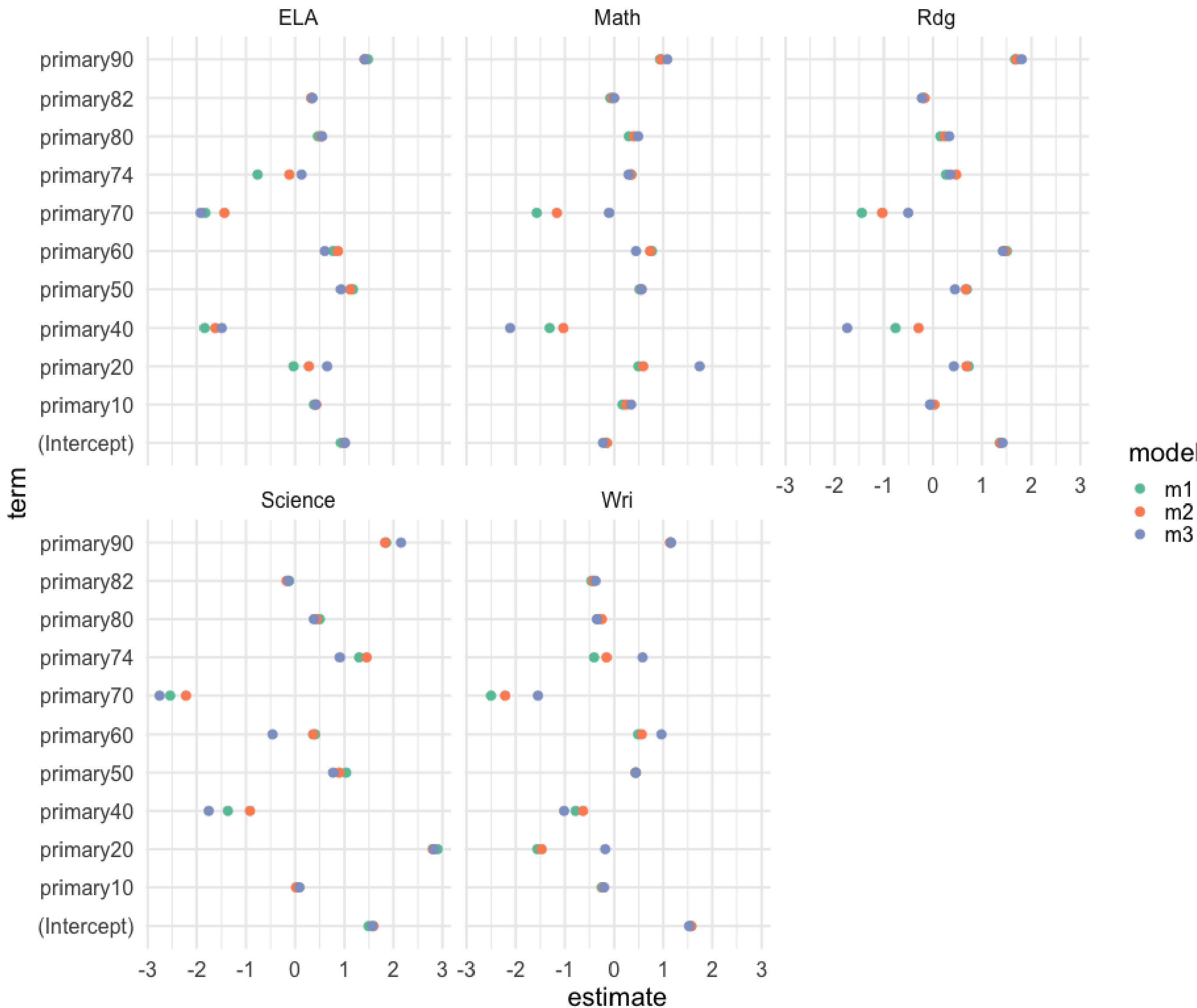
```
## # A tibble: 841 x 7
## # Groups:   content [5]
##    content model term          estimate std.error   statistic     p.val
##    <chr>   <chr> <chr>            <dbl>     <dbl>       <dbl>      <db
##  1 ELA     m1    (Intercept)  0.9322336 0.2150561  4.334839   1.498396e
##  2 ELA     m1    primary10    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

```
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
estimate
-3 -2 -1 0 1 2 3
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 x 3
## # Groups:   grade, content [31]
##    grade content data
##    <int> <chr>   <list>
##  1    11 ELA     <tibble[,25] [453 × 25]>
##  2    11 Math    <tibble[,25] [460 × 25]>
##  3    11 Rdg     <tibble[,25] [453 × 25]>
##  4    11 Science <tibble[,25] [438 × 25]>
##  5    11 Wri     <tibble[,25] [453 × 25]>
##  6     3 ELA     <tibble[,25] [540 × 25]>
##  7     3 Math    <tibble[,25] [536 × 25]>
##  8     3 Rdg     <tibble[,25] [540 × 25]>
##  9     3 Wri     <tibble[,25] [540 × 25]>
## 10     4 ELA     <tibble[,25] [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 x 6
## # Groups:   grade, content [31]
##    grade content data                        m1     m2     m3    
##    <int> <chr>   <list>                      <list> <list> <list>
##  1    11 ELA     <tibble[,25] [453 × 25]> <lm>   <lm>   <lm>  
##  2    11 Math    <tibble[,25] [460 × 25]> <lm>   <lm>   <lm>  
##  3    11 Rdg     <tibble[,25] [453 × 25]> <lm>   <lm>   <lm>  
##  4    11 Science <tibble[,25] [438 × 25]> <lm>   <lm>   <lm>  
##  5    11 Wri     <tibble[,25] [453 × 25]> <lm>   <lm>   <lm>  
##  6     3 ELA     <tibble[,25] [540 × 25]> <lm>   <lm>   <lm>  
##  7     3 Math    <tibble[,25] [536 × 25]> <lm>   <lm>   <lm>  
##  8     3 Rdg     <tibble[,25] [540 × 25]> <lm>   <lm>   <lm>  
##  9     3 Wri     <tibble[,25] [540 × 25]> <lm>   <lm>   <lm>  
## 10     4 ELA     <tibble[,25] [585 × 25]> <lm>   <lm>   <lm>  
## # … with 21 more rows
```

Look at R^2

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

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

Plot

```
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)

Questions?

In-class Midterm

Next time: Parallel iterations