Parallel Looping & Variants

Daniel Anderson

Week 5

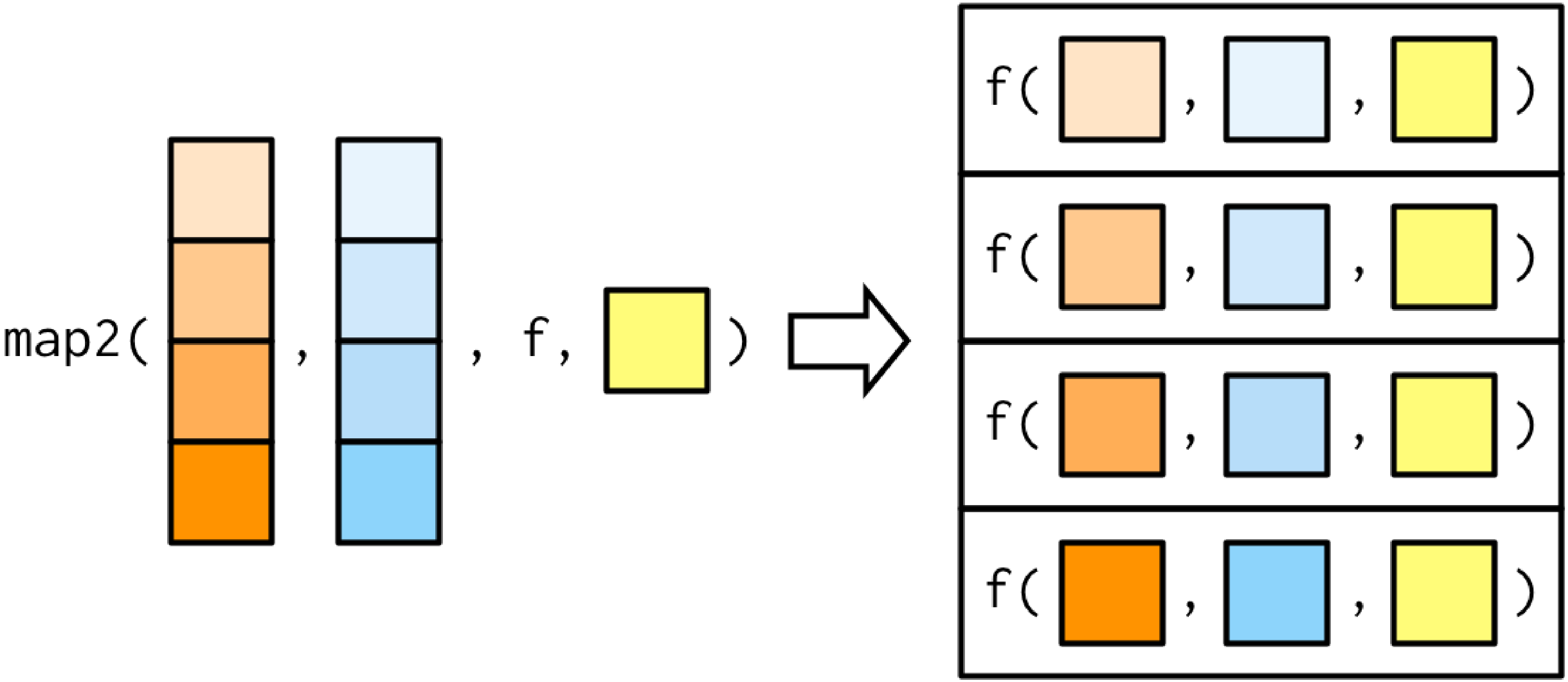
Agenda

- Discuss `map2_*` and `pmap_*` (parallel iterations)
- `walk()` and friends
- `modify()`
- `safely()`
- `reduce()`

Learning objectives

- Understand the differences between `map`, `map2`, and `pmap`
- Know when to apply `walk` instead of `map`, and why it may be useful
- Understand the similarities and differences between `map` and `modify`
- Diagnose errors with `safely` and understand other situations where it may be helpful
- Collapsing/reducing lists with `purrr::reduce()` or `base::Reduce()`

map2



A few Examples

Basic simulations – iterating over two vectors

Plots by month, changing the title

Simulation

- Simulate data from a normal distribution
  - Vary n from 5 to 150 by increments of 5
  - For each n, vary μ from -2 to 2 by increments of 0.25

How do we get all combinations

`expand.grid`

Example `expand.grid`

Bonus: It turns it into a data frame!

```
ints <- 1:3
lets <- c("a", "b", "c")
expand.grid(ints, lets)
```

```
##   Var1 Var2
## 1    1    a
## 2    2    a
## 3    3    a
## 4    1    b
## 5    2    b
## 6    3    b
## 7    1    c
## 8    2    c
## 9    3    c
```

Set conditions

Please follow along

```r
conditions <- expand.grid(
  n = seq(5, 150, 5),
  mu = seq(-2, 2, 0.25)
)
```

```r
head(conditions)
```

```
##    n mu
## 1  5 -2
## 2 10 -2
## 3 15 -2
## 4 20 -2
## 5 25 -2
## 6 30 -2
```

```r
tail(conditions)
```

```
##       n mu
## 505 125  2
## 506 130  2
## 507 135  2
## 508 140  2
## 509 145  2
## 510 150  2
```

Simulate!

```
sim1 <- map2(conditions$n, conditions$mu, ~{
    rnorm(n = .x, mean = .y, sd = 10)
})
str(sim1)
```

```
## List of 510
##  $ : num [1:5] 9.47 11.21 -7.08 -9.52 14.75
##  $ : num [1:10] 5.53 -6.65 11.84 5.97 10.29 ...
##  $ : num [1:15] 5.36 -12.25 -4.77 -9.96 -24.12 ...
##  $ : num [1:20] -13.776 0.971 -4.511 3.479 -3.309 ...
##  $ : num [1:25] -10.3897 0.0782 5.7215 8.8343 6.5607 ...
##  $ : num [1:30] -12.49 7.42 -5.95 -9.56 -22.72 ...
##  $ : num [1:35] 7.098 -0.351 -4.839 0.802 11.27 ...
##  $ : num [1:40] 0.238 -0.162 9.217 -10.91 -9.475 ...
##  $ : num [1:45] -15.36 -4.17 -1.1 -5.25 9.51 ...
##  $ : num [1:50] 2.647 -9.012 -16.856 -0.585 -2.493 ...
##  $ : num [1:55] 2.05 -11.7 1.02 6.81 -12.34 ...
##  $ : num [1:60] 0.841 4.746 -15.114 2.418 -25.436 ...
##  $ : num [1:65] 18.11 -5.65 13.83 -4.19 6.15 ...
##  $ : num [1:70] 4.92 6.8 -20.73 4.92 4.92 ...
##  $ : num [1:75] 1.208 -8.579 -13.281 -13.267 0.666 ...
##  $ : num [1:80] -1.02 -8.1 -7.58 -18.32 -1.62 ...
##  $ : num [1:85] -6.12 -1.57 -5.89 -19.54 -9.57 ...
##  $ : num [1:90] 9.4 -0.217 1.298 2.209 4.611 ...
##  $ : num [1:95] 0.366 2.115 4.806 3.638 19.459 ...
```

More powerful

Add it as a list column!

```
sim2 <- conditions %>%
  as_tibble() %>% # Not required, but definitely helpful
  mutate(sim = map2(n, mu, ~rnorm(n = .x, mean = .y, sd = 10)))
sim2
```

```
## # A tibble: 510 × 3
##         n    mu sim
##     <dbl> <dbl> <list>
##  1      5    -2 <dbl [5]>
##  2     10    -2 <dbl [10]>
##  3     15    -2 <dbl [15]>
##  4     20    -2 <dbl [20]>
##  5     25    -2 <dbl [25]>
##  6     30    -2 <dbl [30]>
##  7     35    -2 <dbl [35]>
##  8     40    -2 <dbl [40]>
##  9     45    -2 <dbl [45]>
## 10     50    -2 <dbl [50]>
## # … with 500 more rows
```

Unnest

```
conditions %>%
  as_tibble() %>%
  mutate(sim = map2(n, mu, ~rnorm(.x, .y, sd = 10))) %>%
  unnest(sim)
```

```
## # A tibble: 39,525 × 3
##        n    mu        sim
##    <dbl> <dbl>      <dbl>
##  1     5    -2  11.23609
##  2     5    -2 -19.33569
##  3     5    -2  20.63748
##  4     5    -2  -4.236432
##  5     5    -2  -1.626640
##  6    10    -2 -14.53035
##  7    10    -2  16.91659
##  8    10    -2   0.7720395
##  9    10    -2   5.808367
## 10    10    -2 -16.02990
## # … with 39,515 more rows
```

Challenge

Can you replicate what we just did, but using a `rowwise()` approach?

```
conditions %>%
  rowwise() %>%
  mutate(sim = list(rnorm(n, mu, sd = 10))) %>%
  unnest(sim)
```

```
## # A tibble: 39,525 × 3
##          n    mu       sim
##      <dbl> <dbl>     <dbl>
##  1       5    -2 -6.617047
##  2       5    -2  8.366971
##  3       5    -2  7.079862
##  4       5    -2  5.905861
##  5       5    -2  6.570213
##  6      10    -2 -17.77516
##  7      10    -2 -6.816531
##  8      10    -2 -10.39636
##  9      10    -2  11.71602
## 10      10    -2  3.212861
## # … with 39,515 more rows
```

03:00

Vary the sd too?

pmap

Which we'll get to soon

Varying the title of a plot

The data

Please follow along

```r
library(fivethirtyeight)
pulitzer
```

```
## # A tibble: 50 × 7
##    newspaper                circ2004 circ2013 pctchg_circ num_finals1990_20
##    <chr>                       <dbl>    <dbl>       <int>                <ir
##  1 USA Today                 2192098  1674306         -24
##  2 Wall Street Journal       2101017  2378827          13
##  3 New York Times            1119027  1865318          67
##  4 Los Angeles Times          983727   653868         -34
##  5 Washington Post            760034   474767         -38
##  6 New York Daily News        712671   516165         -28
##  7 New York Post              642844   500521         -22
##  8 Chicago Tribune            603315   414930         -31
##  9 San Jose Mercury News      558874   583998           4
## 10 Newsday                    553117   377744         -32
## # … with 40 more rows
```

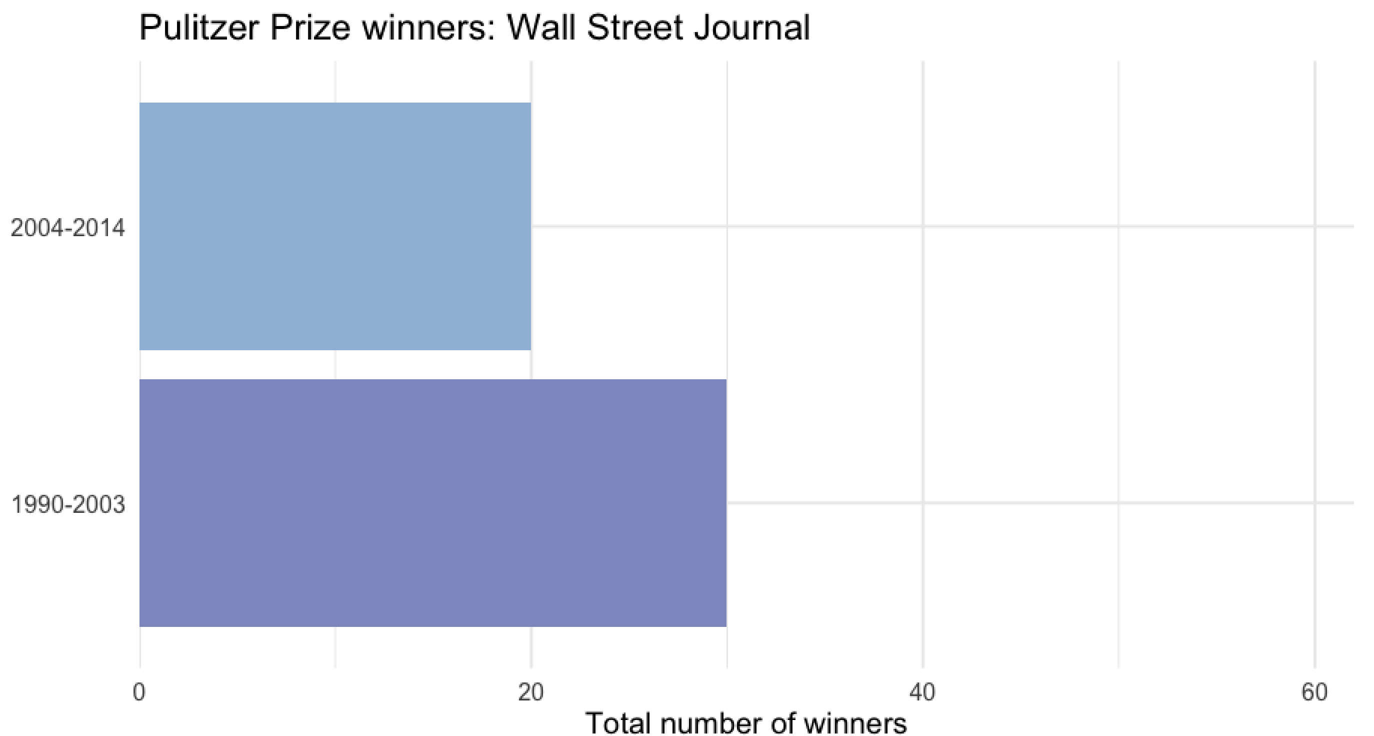
Prep data

```
pulitzer<- pulitzer %>%
  select(newspaper, starts_with("num")) %>%
  pivot_longer(
    -newspaper,
    names_to = "year_range",
    values_to = "n",
    names_prefix = "num_finals"
  ) %>%
  mutate(year_range = str_replace_all(year_range, "_", "-")) %>%
  filter(year_range != "1990-2014")

head(pulitzer)
```

```
## # A tibble: 6 × 3
##   newspaper           year_range     n
##   <chr>               <chr>      <int>
## 1 USA Today           1990-2003      1
## 2 USA Today           2004-2014      1
## 3 Wall Street Journal 1990-2003     30
## 4 Wall Street Journal 2004-2014     20
## 5 New York Times      1990-2003     55
## 6 New York Times      2004-2014     62
```

One plot

```r
wsj <- pulitzer %>%
    filter(newspaper == "Wall Street Journal")

ggplot(wsj, aes(n, year_range)) +
  geom_col(aes(fill = n)) +
  scale_fill_distiller(
    type = "seq",
    limits = c(0, max(pulitzer$n)),
    palette = "BuPu",
    direction = 1
  ) +
  scale_x_continuous(
    limits = c(0, max(pulitzer$n)),
    expand = c(0, 0)
  ) +
  guides(fill = "none") +
  labs(
    title = "Pulitzer Prize winners: Wall Street Journal",
    x = "Total number of winners",
    y = ""
  )
```


Pulitzer Prize winners: Wall Street Journal
2004-2014
1990-2003
0
20
40
60
Total number of winners

Nest data

```
by_newspaper <- pulitzer %>%
    group_by(newspaper) %>%
    nest()

by_newspaper
```

```
## # A tibble: 50 × 2
## # Groups:   newspaper [50]
##    newspaper               data
##    <chr>                   <list>
##  1 USA Today               <tibble [2 × 2]>
##  2 Wall Street Journal     <tibble [2 × 2]>
##  3 New York Times          <tibble [2 × 2]>
##  4 Los Angeles Times       <tibble [2 × 2]>
##  5 Washington Post         <tibble [2 × 2]>
##  6 New York Daily News     <tibble [2 × 2]>
##  7 New York Post           <tibble [2 × 2]>
##  8 Chicago Tribune         <tibble [2 × 2]>
##  9 San Jose Mercury News <tibble [2 × 2]>
## 10 Newsday                 <tibble [2 × 2]>
## # … with 40 more rows
```

Produce all plots

You try first!

Don't worry about the correct title yet, if you don't want

03:00

```r
by_newspaper %>%
    mutate(
      plot = map(
        data, ~{
          ggplot(aes(n, year_range)) +
            geom_col(aes(fill = n)) +
          scale_fill_distiller(
            type = "seq",
            limits = c(0, max(pulitzer$n)),
            palette = "BuPu",
            direction = 1
          ) +
          scale_x_continuous(
            limits = c(0, max(pulitzer$n)),
            expand = c(0, 0)
          ) +
        guides(fill = "none") +
        labs(
          title = "Pulitzer Prize winners",
          x = "Total number of winners",
          y = ""
        )
      }
    )
  )
```

Add title

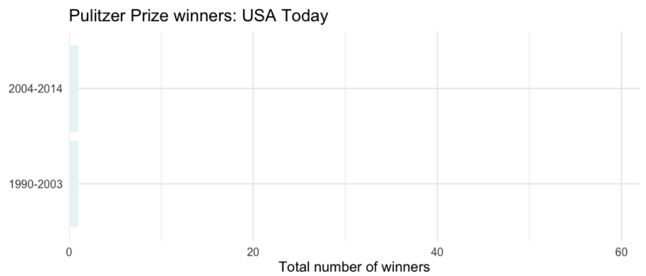
```
library(glue)

p <- by_newspaper %>%
    mutate(
      plot = map2(
     data, newspaper, ~{
         ggplot(.x, aes(n, year_range)) +
           geom_col(aes(fill = n)) +
         scale_fill_distiller(
           type = "seq",
           limits = c(0, max(pulitzer$n)),
           palette = "BuPu",
           direction = 1
         ) +
         scale_x_continuous(
           limits = c(0, max(pulitzer$n)),
           expand = c(0, 0)
         ) +
         guides(fill = "none") +
         labs(
           title = glue("Pulitzer Prize winners: {.y}"),
           x = "Total number of winners",
           y = ""
```

```
p
```

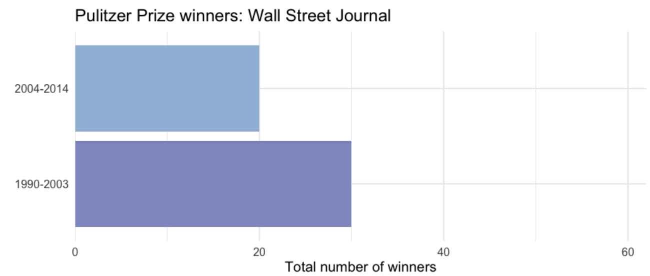
```
## # A tibble: 50 × 3
## # Groups:   newspaper [50]
##    newspaper                 data             plot
##    <chr>                     <list>           <list>
##  1 USA Today                 <tibble [2 × 2]> <gg>
##  2 Wall Street Journal       <tibble [2 × 2]> <gg>
##  3 New York Times            <tibble [2 × 2]> <gg>
##  4 Los Angeles Times         <tibble [2 × 2]> <gg>
##  5 Washington Post           <tibble [2 × 2]> <gg>
##  6 New York Daily News       <tibble [2 × 2]> <gg>
##  7 New York Post             <tibble [2 × 2]> <gg>
##  8 Chicago Tribune           <tibble [2 × 2]> <gg>
##  9 San Jose Mercury News     <tibble [2 × 2]> <gg>
## 10 Newsday                   <tibble [2 × 2]> <gg>
## # … with 40 more rows
```

Look at a couple plots

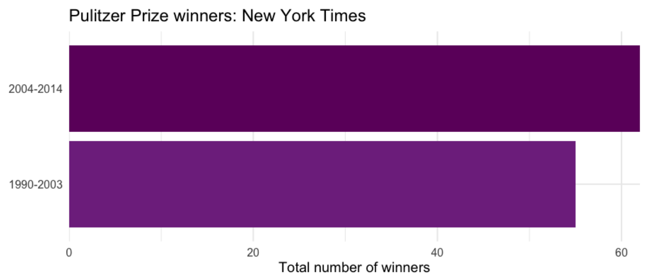
```r
p$plot[[1]]
```



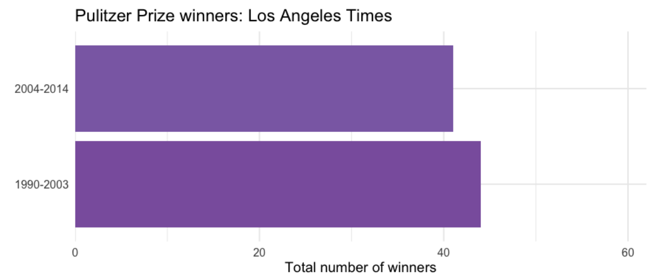

```r
p$plot[[2]]
```




```r
p$plot[[3]]
```




```r
p$plot[[4]]
```



Challenge

(You can probably guess where this is going)

Can you reproduce the prior plots using a `rowwise()` approach?

03:00

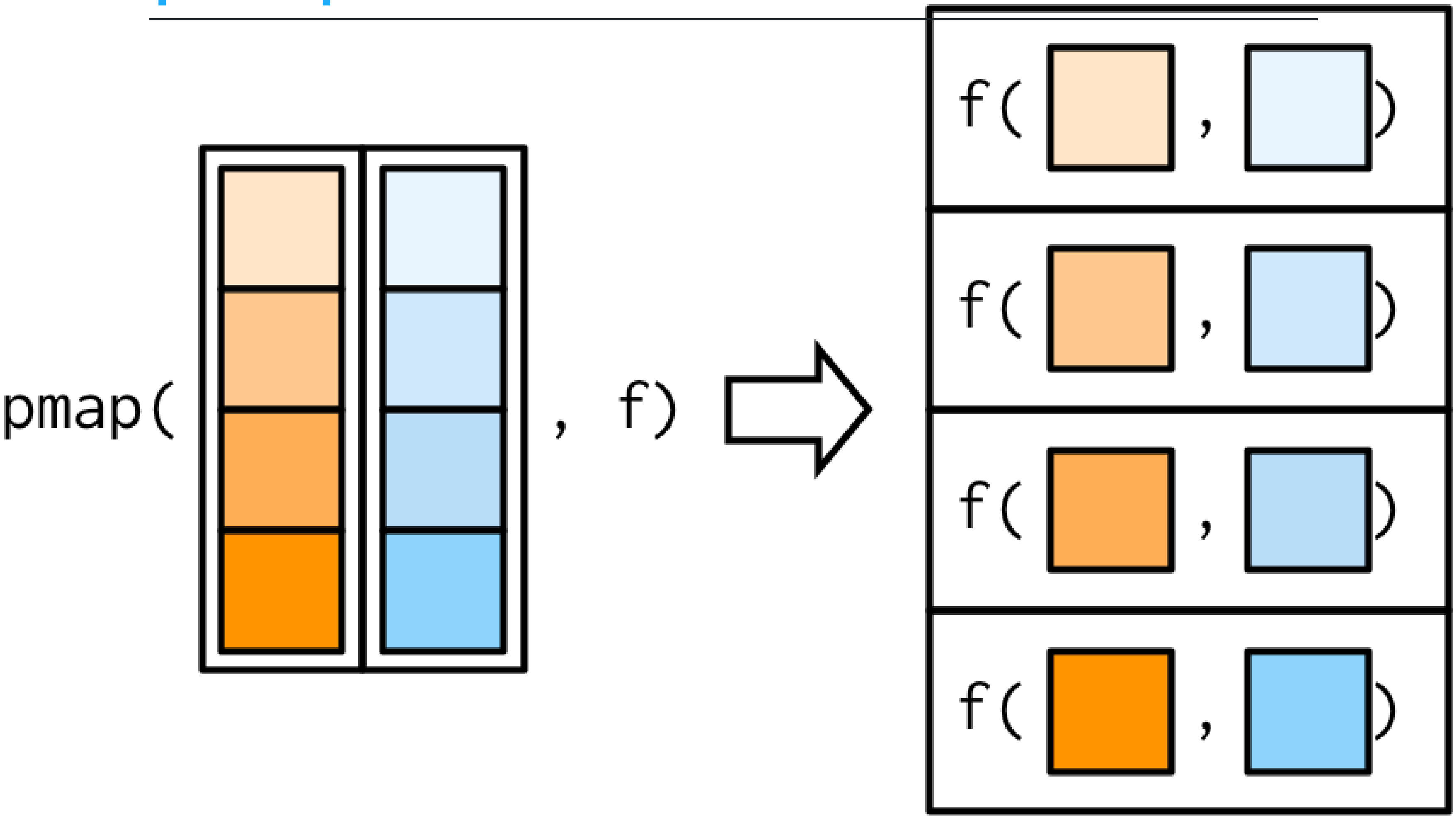
```r
pulitzer %>%
nest_by(newspaper) %>%
    mutate(
      plot = list(
     ggplot(data, aes(n, year_range)) +
        geom_col(aes(fill = n)) +
        scale_fill_distiller(
          type = "seq",
          limits = c(0, max(pulitzer$n)),
          palette = "BuPu",
          direction = 1
        ) +
        scale_x_continuous(
          limits = c(0, max(pulitzer$n)),
          expand = c(0, 0)
        ) +
        guides(fill = "none") +
        labs(
          title = glue("Pulitzer Prize winners: {newspaper}"),
          x = "Total number of winners",
          y = ""
        )
      )
  )
```

```
## # A tibble: 50 × 3
```

Iterating over n vectors

pmap

pmap



Simulation

- Simulate data from a normal distribution
    - Vary n from 5 to 150 by increments of 5
    - For each n, vary μ from -2 to 2 by increments of 0.25
    - For each σ from 1 to 3 by increments of 0.1

```r
full_conditions <- expand.grid(
  n = seq(5, 150, 5),
  mu = seq(-2, 2, 0.25),
  sd = seq(1, 3, .1)
)
```

```r
head(full_conditions)
```

```
##    n mu sd
## 1  5 -2  1
## 2 10 -2  1
## 3 15 -2  1
## 4 20 -2  1
## 5 25 -2  1
## 6 30 -2  1
```

```r
tail(full_conditions)
```

```
##         n mu sd
## 10705 125  2  3
## 10706 130  2  3
## 10707 135  2  3
## 10708 140  2  3
## 10709 145  2  3
## 10710 150  2  3
```

Full Simulation

```
fsim <- pmap(
  list(
    number = full_conditions$n,
    average = full_conditions$mu,
    stdev = full_conditions$sd
  ),
  function(number, average, stdev) {
    rnorm(n = number, mean = average, sd = stdev)
  }
)
str(fsim)
```

```
## List of 10710
##  $ : num [1:5] -2.69 -1.13 -2.56 -2.81 -3.09
##  $ : num [1:10] -3.867 -1.429 -1.567 -0.964 -1.355 ...
##  $ : num [1:15] -1.913 -2.158 -3.85 0.209 -3.776 ...
##  $ : num [1:20] -2.96 -1.44 -2.49 -3.51 -3.02 ...
##  $ : num [1:25] -3.232 0.512 -1.843 -1.531 -1.217 ...
##  $ : num [1:30] -0.84 -2.23 -2.12 -1.15 -1.59 ...
##  $ : num [1:35] -2.484 -0.561 -0.962 -0.574 -2.11 ...
##  $ : num [1:40] -3.769 -2.752 -0.915 -3.348 -2.75 ...
##  $ : num [1:45] -0.915 -2.217 -2.843 -2.104 -2.282 ...
##  $ : num [1:50] -0.796 -2.114 -3.433 -2.958 -2.386 ...
##  $ : num [1:55] -3.141 -3.462 -2.242 -1.983 -0.963 ...
```

Alternative spec

```r
fsim <- pmap(
  list(
    full_conditions$n,
    full_conditions$mu,
    full_conditions$sd
  ),
  ~rnorm(n = ..1, mean = ..2, sd = ..3)
)
str(fsim)
```

```
## List of 10710
##  $ : num [1:5] -1.042 -2.196 -0.969 -2.763 -1.576
##  $ : num [1:10] -2 -1.81 -1.86 -4.61 -2.96 ...
##  $ : num [1:15] -0.492 -2.355 -2.651 -2.468 -0.951 ...
##  $ : num [1:20] -1.92 -2.512 -1.415 -2.339 -0.969 ...
##  $ : num [1:25] -2.43 -1.7 -4.26 -1.14 -3.97 ...
##  $ : num [1:30] -3.554 -1.969 -1.428 -1.46 -0.276 ...
##  $ : num [1:35] -4.07 -1.72697 -3.03746 -1.54112 -0.00792 ...
##  $ : num [1:40] -1.98 -2.74 -2.38 -1.54 -1.17 ...
##  $ : num [1:45] -2.802 -2.193 0.462 -0.726 -1.645 ...
##  $ : num [1:50] -0.9602 -0.0473 -1.8664 -1.6328 -1.5447 ...
##  $ : num [1:55] -1.54 -1.72 -2.35 -2.31 -2.83 ...
##  $ : num [1:60] -0.927 -3.526 -2.883 -1.166 -1.988 ...
##  $ : num [1:65] -2.88 -1.22 -1.21 -3.33 -2.06 ...
```

Simpler

Maybe a little too clever

- A data frame is a list so...

```
fsim <- pmap(
  full_conditions,
  ~rnorm(n = ..1, mean = ..2, sd = ..3)
)
str(fsim)
```

```
## List of 10710
##  $ : num [1:5] -3.705 -0.907 -3.069 -2.746 -1.356
##  $ : num [1:10] -0.984 -2.316 -2.465 -1.896 -1.651 ...
##  $ : num [1:15] -1.9 -3.23 -1.47 -1.18 -1.44 ...
##  $ : num [1:20] -2.73 -1.73 -2.775 -0.321 -1.173 ...
##  $ : num [1:25] -2.864 -2.244 -0.483 -0.422 -2.011 ...
##  $ : num [1:30] -1.288 -4.209 -1.85 -0.922 0.17 ...
##  $ : num [1:35] -0.772 -1.375 -1.177 -2.744 -2.904 ...
##  $ : num [1:40] -1.74 -2.48 -1.59 -2.4 -0.32 ...
##  $ : num [1:45] -2.53 -1.35 -3.34 -3.43 -1.87 ...
##  $ : num [1:50] -1.36 -1.42 -3.89 -1.84 -2.14 ...
##  $ : num [1:55] -0.96 -2.814 -2.812 -2.169 -0.218 ...
##  $ : num [1:60] -3.67 -1.49 -2.34 -2.31 -2.08 ...
```

List column version

```
full_conditions %>%
    as_tibble() %>%
    mutate(sim = pmap(list(n, mu, sd), ~rnorm(..1, ..2, ..3)))
```

```
## # A tibble: 10,710 × 4
##        n    mu    sd sim
##    <dbl> <dbl> <dbl> <list>
##  1     5    -2     1 <dbl [5]>
##  2    10    -2     1 <dbl [10]>
##  3    15    -2     1 <dbl [15]>
##  4    20    -2     1 <dbl [20]>
##  5    25    -2     1 <dbl [25]>
##  6    30    -2     1 <dbl [30]>
##  7    35    -2     1 <dbl [35]>
##  8    40    -2     1 <dbl [40]>
##  9    45    -2     1 <dbl [45]>
## 10    50    -2     1 <dbl [50]>
## # … with 10,700 more rows
```

Unnest

```
full_conditions %>%
    as_tibble() %>%
    mutate(sim = pmap(
        list(n, mu, sd), ~rnorm(..1, ..2, ..3)
      )
  ) %>%
    unnest(sim)
```

```
## # A tibble: 830,025 × 4
##        n    mu    sd        sim
##    <dbl> <dbl> <dbl>      <dbl>
##  1     5    -2     1 -3.279599
##  2     5    -2     1 -3.463857
##  3     5    -2     1 -1.351400
##  4     5    -2     1 -1.890146
##  5     5    -2     1 -1.665575
##  6    10    -2     1 -2.070767
##  7    10    -2     1 -1.621322
##  8    10    -2     1 -2.847094
##  9    10    -2     1 -0.5013484
## 10    10    -2     1 -2.259621
## # … with 830,015 more rows
```

Replicate with `nest_by()`

You try first

```
full_conditions %>%
  rowwise() %>%
  mutate(sim = list(rnorm(n, mu, sd))) %>%
  unnest(sim)
```

```
## # A tibble: 830,025 × 4
##          n     mu     sd        sim
##      <dbl>  <dbl>  <dbl>      <dbl>
##  1       5     -2      1 -0.7758622
##  2       5     -2      1 -2.208930
##  3       5     -2      1 -2.674586
##  4       5     -2      1 -3.309357
##  5       5     -2      1 -2.722848
##  6      10     -2      1 -1.191633
##  7      10     -2      1 -3.199831
##  8      10     -2      1 -3.458367
##  9      10     -2      1 -1.735262
## 10      10     -2      1 -2.885026
## # … with 830,015 more rows
```

03:00

Plot

Add a caption stating the total number of Pulitzer prize winners across years

Add column for total

```
pulitzer <- pulitzer %>%
    group_by(newspaper) %>%
    mutate(tot = sum(n))
pulitzer
```

```
## # A tibble: 100 × 4
## # Groups:   newspaper [50]
##    newspaper           year_range     n   tot
##    <chr>               <chr>      <int> <int>
##  1 USA Today           1990-2003      1     2
##  2 USA Today           2004-2014      1     2
##  3 Wall Street Journal 1990-2003     30    50
##  4 Wall Street Journal 2004-2014     20    50
##  5 New York Times      1990-2003     55   117
##  6 New York Times      2004-2014     62   117
##  7 Los Angeles Times   1990-2003     44    85
##  8 Los Angeles Times   2004-2014     41    85
##  9 Washington Post     1990-2003     52   100
## 10 Washington Post     2004-2014     48   100
## # … with 90 more rows
```

Easiest way (imo)

Create a column to represent exactly the label you want.

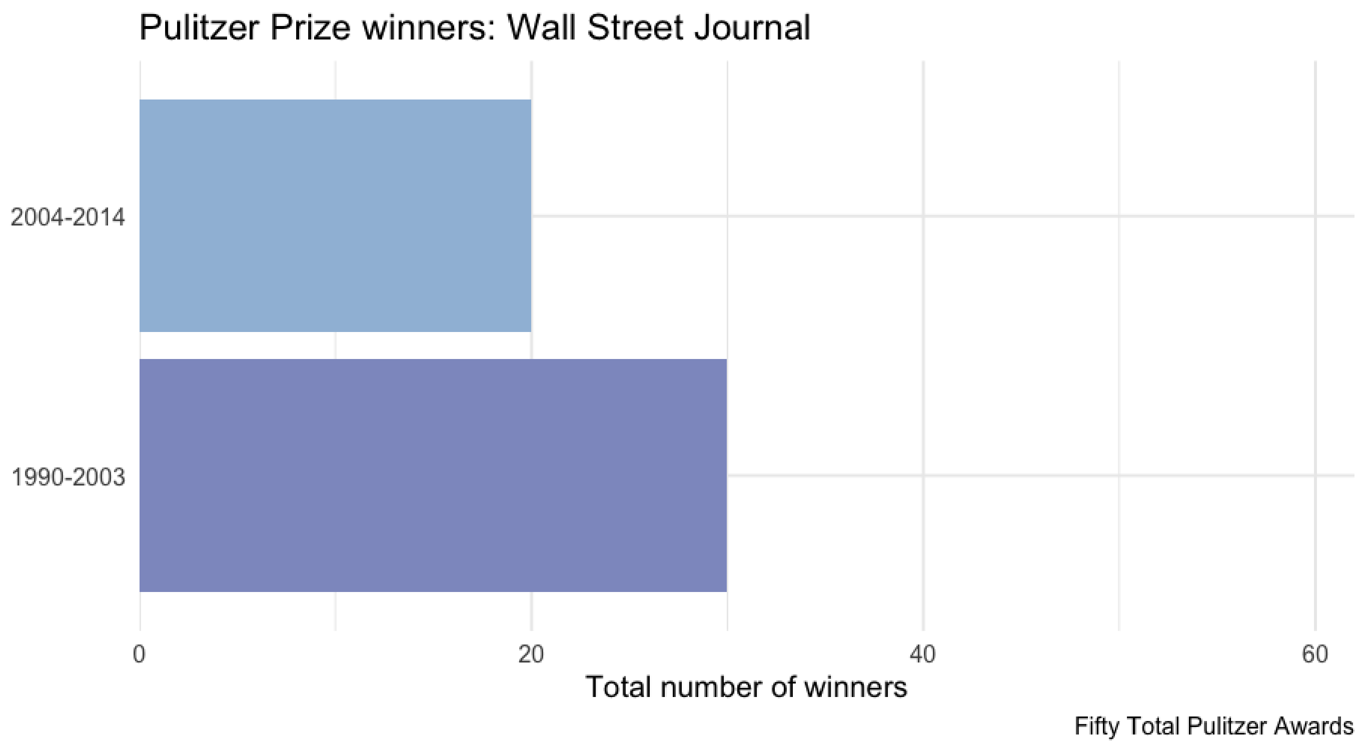
```r
#install.packages("english")
library(english)
pulitzer <- pulitzer %>%
    mutate(
      label = glue(
        "{str_to_title(as.english(tot))} Total Pulitzer Awards"
    )
)
```

```r
select(pulitzer, newspaper, label)
```

```
## # A tibble: 100 × 2
## # Groups:   newspaper [50]
##    newspaper           label
##    <chr>               <glue>
##  1 USA Today           Two Total Pulitzer Awards
##  2 USA Today           Two Total Pulitzer Awards
##  3 Wall Street Journal Fifty Total Pulitzer Awards
##  4 Wall Street Journal Fifty Total Pulitzer Awards
##  5 New York Times      One Hundred Seventeen Total Pulitzer Awards
##  6 New York Times      One Hundred Seventeen Total Pulitzer Awards
##  7 Los Angeles Times   Eighty-Five Total Pulitzer Awards
##  8 Los Angeles Times   Eighty-Five Total Pulitzer Awards
##  9 Washington Post     One Hundred Total Pulitzer Awards
## 10 Washington Post     One Hundred Total Pulitzer Awards
## # … with 90 more rows
```

Produce one plot

```
wsj2 <- pulitzer %>%
    filter(newspaper == "Wall Street Journal")

ggplot(wsj2, aes(n, year_range)) +
  geom_col(aes(fill = n)) +
  scale_fill_distiller(
    type = "seq",
    limits = c(0, max(pulitzer$n)),
    palette = "BuPu",
    direction = 1
  ) +
  scale_x_continuous(
    limits = c(0, max(pulitzer$n)),
    expand = c(0, 0)
  ) +
  guides(fill = "none") +
  labs(
    title = glue("Pulitzer Prize winners: Wall Street Journal"),
    x = "Total number of winners",
    y = "",
    caption = unique(wsj2$label)
  )
```


Pulitzer Prize winners: Wall Street Journal
2004-2014
1990-2003
0
20
40
60
Total number of winners
Fifty Total Pulitzer Awards

Produce all plots

Nest first

```
by_newspaper_label <- pulitzer %>%
    group_by(newspaper, label) %>%
    nest()

by_newspaper_label
```

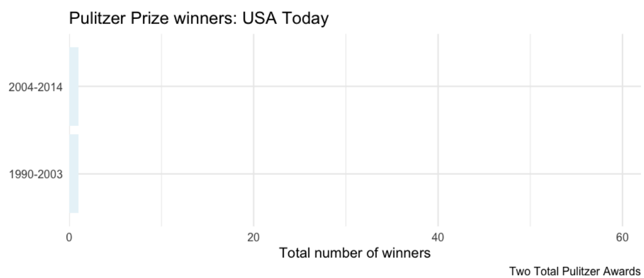
```
## # A tibble: 50 × 3
## # Groups:   newspaper, label [50]
##    newspaper                 label                                         dat
##    <chr>                     <glue>                                        <li
##  1 USA Today                 Two Total Pulitzer Awards                     <ti
##  2 Wall Street Journal       Fifty Total Pulitzer Awards                   <ti
##  3 New York Times            One Hundred Seventeen Total Pulitzer Awards <ti
##  4 Los Angeles Times         Eighty-Five Total Pulitzer Awards             <ti
##  5 Washington Post           One Hundred Total Pulitzer Awards             <ti
##  6 New York Daily News       Six Total Pulitzer Awards                     <ti
##  7 New York Post             Zero Total Pulitzer Awards                    <ti
##  8 Chicago Tribune           Thirty-Eight Total Pulitzer Awards            <ti
##  9 San Jose Mercury News Six Total Pulitzer Awards                         <ti
## 10 Newsday                   Eighteen Total Pulitzer Awards                <ti
## # … with 40 more rows
```

Produce plots

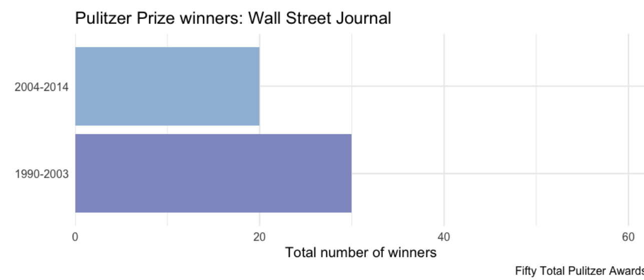
```
final_plots <- by_newspaper_label %>%
    mutate(plots = pmap(list(newspaper, label, data), ~{
    ggplot(..3, aes(n, year_range)) +
      geom_col(aes(fill = n)) +
      scale_fill_distiller(
        type = "seq",
        limits = c(0, max(pulitzer$n)),
        palette = "BuPu",
        direction = 1
      ) +
        scale_x_continuous(
          limits = c(0, max(pulitzer$n)),
          expand = c(0, 0)
      ) +
        guides(fill = "none") +
        labs(
       title = glue("Pulitzer Prize winners: {..1}"),
          x = "Total number of winners",
          y = "",
       caption = ..2
        )
      })
)
```

Look at a couple plots

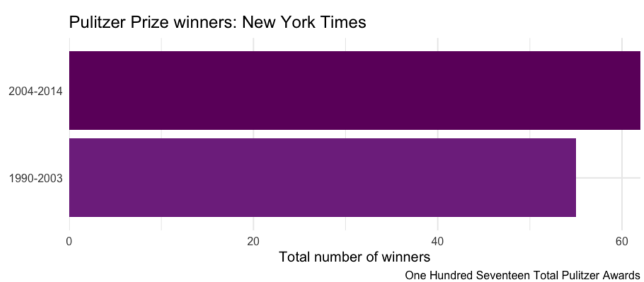
```
final_plots$plots[[1]]
```



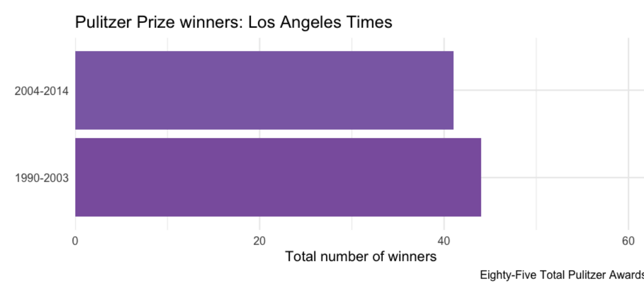

```
final_plots$plots[[2]]
```




```
final_plots$plots[[3]]
```




```
final_plots$plots[[4]]
```



Replicate with `nest_by()`

You try first

03:00

```r
final_plots2 <- pulitzer %>%
  ungroup() %>%
  nest_by(newspaper, label) %>%
    mutate(
      plots = list(
        ggplot(data, aes(n, year_range)) +
          geom_col(aes(fill = n)) +
        scale_fill_distiller(
          type = "seq",
          limits = c(0, max(pulitzer$n)),
          palette = "BuPu",
          direction = 1
        ) +
          scale_x_continuous(
            limits = c(0, max(pulitzer$n)),
            expand = c(0, 0)
        ) +
          guides(fill = "none") +
          labs(
            title = glue("Pulitzer Prize winners: {newspaper}"),
            x = "Total number of winners",
            y = "",
            caption = label
        )
      )
)
```

```r
final_plots2
```

```
## # A tibble: 50 × 4
## # Rowwise:  newspaper, label
##    newspaper                          label
##    <chr>                              <glue>                                   <list
##  1 Arizona Republic                   Seven Total Pulitzer Awards
##  2 Atlanta Journal Constitution Six Total Pulitzer Awards
##  3 Baltimore Sun                      Thirteen Total Pulitzer Awards
##  4 Boston Globe                       Forty-One Total Pulitzer Awards
##  5 Boston Herald                      Zero Total Pulitzer Awards
##  6 Charlotte Observer                 Four Total Pulitzer Awards
##  7 Chicago Sun-Times                  Two Total Pulitzer Awards
##  8 Chicago Tribune                    Thirty-Eight Total Pulitzer Awards
##  9 Cleveland Plain Dealer             Eleven Total Pulitzer Awards
## 10 Columbus Dispatch                  One Total Pulitzer Awards
## # … with 40 more rows
```

Save all plots

We'll have to iterate across at least two things: (a) file path/names, and (b) the plots themselves

We can do this with the `map()` family, but instead we'll use a different function, which we'll talk about more next week.

As an aside, what are the **steps** we would need to take to do this?

Could we use a `nest_by()` solution?

Try with `nest_by()`

You try first:

- Create a vector of file paths
- "loop" through the file paths and the plots to save them

04:00

Example

Create a directory

```
fs::dir_create(here::here("plots", "pulitzers"))
```

Create file paths

```
files <- str_replace_all(
  tolower(final_plots$newspaper),
  " ",
  "_"
)
paths <- here::here("plots", "pulitzers", glue("{files}.png"))
paths
```

```
##  [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/
```

Add paths to data frame

```
final_plots %>%
  ungroup() %>%
  mutate(path = paths) %>%
  select(plots, path)
```

```
## # A tibble: 50 × 2
##    plots  path
##    <list> <chr>
##  1 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  2 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  3 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  4 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  5 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  6 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  7 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  8 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
##  9 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
## 10 <gg>   /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2
## # … with 40 more rows
```

Save

```r
final_plots %>%
  ungroup() %>%
  mutate(path = paths) %>%
  rowwise() %>%
  summarize(
    ggsave(
      path,
      plots,
      width = 9.5,
      height = 6.5,
      dpi = 500
    )
  )
```

```
## # A tibble: 50 × 1
##    `ggsave(path, plots, width = 9.5, height = 6.5, dpi = 500)`
##    <chr>
##  1 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
##  2 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
##  3 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
##  4 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
##  5 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
##  6 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
##  7 /Users/daniel/Teaching/data_sci_specialization/2021-22/c3-fp-2022/plo
```

Wrap-up

- Parallel iterations greatly increase the things you can do – iterating through at least two things simultaneously is pretty common

- The `nest_by()` approach can regularly get you the same result as `group_by() %>% nest() %>% mutate() %>% map()`

    - Caveat – must be in a data frame, which means working with list columns
    - My view – it's still worth learning both. Looping with **{purrr}** is super flexible and often safer than base versions (type safe). Doesn't have to be used within a data frame.

Break

Looping variants

Agenda

- `walk()` and friends
- `modify()`
- `safely()`
- `reduce()`

Reminder

Learning Objectives (for this part)

- Know when to apply `walk` instead of `map`, and why it may be useful
- Understand the parallels and differences between `map` and `modify`
- Diagnose errors with `safely` and understand other situations where it may be helpful
- Collapsing/reducing lists with `purrr::reduce()` or `base::Reduce()`

Setup

Let's go back to our plotting example:

Saving

- We saw last time that we could use `nest_by()`
  - Required a bit of awkwardness with adding the paths to the data frame
  - Instead, we'll do it again but with the `walk()` family

Why `walk()`?

> Walk is an alternative to map that you use when you want to call a function for its side effects, rather than for its return value. You typically do this because you want to render output to the screen or save files to disk – the important thing is the action, not the return value.

–r4ds

More practical

If you use `walk()`, nothing will get printed to the screen.
This is particularly helpful for RMarkdown files.

Example

Please do the following

- Create a new RMarkdown document
- Paste the code you have for creating the plots in a code chunk there (along with the library loading, data cleaning, etc.)

03:00

Create a directory

We already did this, but in case we hadn't...

```
fs::dir_create(here::here("plots", "pulitzers"))
```

Create file paths

```
newspapers <- str_replace_all(
  tolower(final_plots$newspaper),
  " ",
  "_"
)
paths <- here::here(
  "plots",
  "pulitzers",
  glue("{newspapers}.png")
)
```

Challenge

- Use a `map()` family function to loop through `paths` and `final_plots$plots` to save all plots.

- Render (knit) your file. What do you notice?

03:00

`walk()`

Just like `map()`, we have parallel variants of `walk()`, including, `walk2()`, and `pwalk()`

These work just like `map()` but don't print to the screen

Try replacing your prior code with a `walk()` version.

How does the rendered output change?

02:00

Save plots

```
walk2(paths, final_plots$plots, ggsave,
      width = 9.5,
      height = 6.5,
      dpi = 500)
```

modify

> Unlike `map()` and its variants which always return a fixed object type (list for `map()`, integer vector for `map_int()`, etc), the `modify()` family always returns the same type as the input object.

`map` vs `modify`

map

```
map(mtcars, ~as.numeric(scale(.x)))
```

```
## $mpg
##  [1]  0.15088482  0.15088482  0.44954345  0.21725341 -0.23073453 -0.3302
## [10] -0.14777380 -0.38006384 -0.61235388 -0.46302456 -0.81145962 -1.6078
## [19]  1.71054652  2.29127162  0.23384555 -0.76168319 -0.81145962 -1.1267
## [28]  1.71054652 -0.71190675 -0.06481307 -0.84464392  0.21725341
##
## $cyl
##  [1] -0.1049878 -0.1049878 -1.2248578 -0.1049878  1.0148821 -0.1049878
## [11] -0.1049878  1.0148821  1.0148821  1.0148821  1.0148821  1.0148821
## [21] -1.2248578  1.0148821  1.0148821  1.0148821  1.0148821 -1.2248578 -
## [31]  1.0148821 -1.2248578
##
## $disp
##  [1] -0.57061982 -0.57061982 -0.99018209  0.22009369  1.04308123 -0.0461
## [10] -0.50929918 -0.50929918  0.36371309  0.36371309  0.36371309  1.9467
## [19] -1.25079481 -1.28790993 -0.89255318  0.70420401  0.59124494  0.9623
## [28] -1.09426581  0.97046468 -0.69164740  0.56703942 -0.88529152
##
## $hp
##  [1] -0.53509284 -0.53509284 -0.78304046 -0.53509284  0.41294217 -0.6080
```

modify

```
modify(mtcars, ~as.numeric(scale(.x)))
```

```
##                             mpg        cyl        disp          hp
## Mazda RX4            0.15088482 -0.1049878 -0.57061982 -0.53509284  0.56
## Mazda RX4 Wag        0.15088482 -0.1049878 -0.57061982 -0.53509284  0.56
## Datsun 710           0.44954345 -1.2248578 -0.99018209 -0.78304046  0.47
## Hornet 4 Drive       0.21725341 -0.1049878  0.22009369 -0.53509284 -0.96
## Hornet Sportabout   -0.23073453  1.0148821  1.04308123  0.41294217 -0.83
## Valiant             -0.33028740 -0.1049878 -0.04616698 -0.60801861 -1.56
## Duster 360          -0.96078893  1.0148821  1.04308123  1.43390296 -0.72
## Merc 240D            0.71501778 -1.2248578 -0.67793094 -1.23518023  0.17
## Merc 230             0.44954345 -1.2248578 -0.72553512 -0.75387015  0.60
## Merc 280            -0.14777380 -0.1049878 -0.50929918 -0.34548584  0.60
## Merc 280C           -0.38006384 -0.1049878 -0.50929918 -0.34548584  0.60
## Merc 450SE          -0.61235388  1.0148821  0.36371309  0.48586794 -0.98
## Merc 450SL          -0.46302456  1.0148821  0.36371309  0.48586794 -0.98
## Merc 450SLC         -0.81145962  1.0148821  0.36371309  0.48586794 -0.98
## Cadillac Fleetwood  -1.60788262  1.0148821  1.94675381  0.85049680 -1.24
## Lincoln Continental -1.60788262  1.0148821  1.84993175  0.99634834 -1.11
## Chrysler Imperial   -0.89442035  1.0148821  1.68856165  1.21512565 -0.68
## Fiat 128             2.04238943 -1.2248578 -1.22658929 -1.17683962  0.90
## Honda Civic          1.71054652 -1.2248578 -1.25079481 -1.38103178  2.49
## Toyota Corolla       2.29127162 -1.2248578 -1.28790993 -1.19142477  1.16
## Toyota Corona        0.23384555 -1.2248578 -0.89255318 -0.72469984  0.19
## Dodge Challenger    -0.76168319  1.0148821  0.70420401  0.04831332 -1.56
## AMC Javelin         -0.81145962  1.0148821  0.59124494  0.04831332 -0.83
## Camaro Z28          -1.12671039  1.0148821  0.96239618  1.43390296  0.24
```

```r
map2(LETTERS[1:3], letters[1:3], paste0)
```

```
## [[1]]
## [1] "Aa"
##
## [[2]]
## [1] "Bb"
##
## [[3]]
## [1] "Cc"
```

```r
modify2(LETTERS[1:3], letters[1:3], paste0)
```

```
## [1] "Aa" "Bb" "Cc"
```

safely

Errors during iterations

Sometimes a loop will work for most cases, but return an error on a few

Often, you want to return the output you can

Alternatively, you might want to diagnose *where* the error is occurring

`purrr::safely()`

Example

```
by_cyl <- mpg %>%
  group_by(cyl) %>%
  nest()
by_cyl
```

```
## # A tibble: 4 × 2
## # Groups:   cyl [4]
##     cyl data
##   <int> <list>
## 1     4 <tibble [81 × 10]>
## 2     6 <tibble [79 × 10]>
## 3     8 <tibble [70 × 10]>
## 4     5 <tibble [4 × 10]>
```

Please run the code above

01:00

Try to fit a model

(please follow along)

Notice the error message is *super* helpful! (this is new)

```
by_cyl %>%
  mutate(mod = map(data, ~lm(hwy ~ displ + drv, data = .x)))
```

```
## Error in `mutate()`:
## ! Problem while computing `mod = map(data, ~lm(hwy ~ displ + drv, data =
## ℹ The error occurred in group 2: cyl = 5.
## Caused by error in `contrasts<-`:
## ! contrasts can be applied only to factors with 2 or more levels
```

Safe return

- First, define safe function – note that this will work for any function

```
safe_lm <- safely(lm)
```

- Next, loop the safe function, instead of the standard function

```
safe_models <- by_cyl %>%
  mutate(safe_mod = map(data, ~safe_lm(hwy ~ displ + drv, data =
safe_models
```

```
## # A tibble: 4 × 3
## # Groups:   cyl [4]
##     cyl data               safe_mod
##   <int> <list>             <list>
## 1     4 <tibble [81 × 10]> <named list [2]>
## 2     6 <tibble [79 × 10]> <named list [2]>
## 3     8 <tibble [70 × 10]> <named list [2]>
## 4     5 <tibble [4 × 10]>  <named list [2]>
```

What's returned?

```r
safe_models$safe_mod[[1]]
```

```
## $result
##
## Call:
## .f(formula = ..1, data = ..2)
##
## Coefficients:
## (Intercept)        displ         drvf
##      37.370       -5.289        3.882
##
##
## $error
## NULL
```

```r
safe_models$safe_mod[[4]]
```

```
## $result
## NULL
##
## $error
## <simpleError in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]])
```

Inspecting

I often use `safely()` to help me de–bug. Why is it failing *there (but note the new error messages help with this too).

First – create a new variable to filter for results with errors

```
safe_models %>%
  mutate(error = map_lgl(safe_mod, ~!is.null(.x$error)))
```

```
## # A tibble: 4 × 4
## # Groups:   cyl [4]
##     cyl data                  safe_mod         error
##   <int> <list>                <list>           <lgl>
## 1     4 <tibble [81 × 10]> <named list [2]> FALSE
## 2     6 <tibble [79 × 10]> <named list [2]> FALSE
## 3     8 <tibble [70 × 10]> <named list [2]> FALSE
## 4     5 <tibble [4 × 10]>  <named list [2]> TRUE
```

Inspecting the data

```
safe_models %>%
  mutate(error = map_lgl(safe_mod, ~!is.null(.x$error))) %>%
  filter(isTRUE(error)) %>%
  select(cyl, data) %>%
  unnest(data)
```

```
## # A tibble: 4 × 11
## # Groups:   cyl [1]
##     cyl manufacturer model       displ  year trans      drv     cty   hwy
##   <int> <chr>        <chr>       <dbl> <int> <chr>      <chr> <int> <int>
## 1     5 volkswagen   jetta         2.5  2008 auto(s6)   f        21    29
## 2     5 volkswagen   jetta         2.5  2008 manual(m5) f        21    29
## 3     5 volkswagen   new beetle    2.5  2008 manual(m5) f        20    28
## 4     5 volkswagen   new beetle    2.5  2008 auto(s6)   f        20    29
```

The `displ` and `drv` variables are constant, so no relation can be estimated.

Pull results that worked

```
safe_models %>%
  mutate(results = map(safe_mod, "result"))
```

```
## # A tibble: 4 × 4
## # Groups:   cyl [4]
##     cyl data                safe_mod           results
##   <int> <list>              <list>             <list>
## 1     4 <tibble [81 × 10]> <named list [2]> <lm>
## 2     6 <tibble [79 × 10]> <named list [2]> <lm>
## 3     8 <tibble [70 × 10]> <named list [2]> <lm>
## 4     5 <tibble [4 × 10]>  <named list [2]> <NULL>
```

Now we can `broom::tidy()` or whatevs

Notice that there is no `cyl == 5`.

```
safe_models %>%
  mutate(results = map(safe_mod, "result"),
         tidied = map(results, broom::tidy)) %>%
  select(cyl, tidied) %>%
  unnest(tidied)
```

```
## # A tibble: 11 × 6
## # Groups:   cyl [3]
##      cyl term        estimate std.error  statistic      p.value
##    <int> <chr>          <dbl>     <dbl>      <dbl>        <dbl>
##  1     4 (Intercept) 37.37023  3.537572  10.56381  1.052943e-16
##  2     4 displ       -5.288562 1.436068  -3.682668 4.235795e- 4
##  3     4 drvf         3.882134 0.9971876  3.893083 2.073699e- 4
##  4     6 (Intercept) 27.96536  2.347630  11.91217  5.718039e-19
##  5     6 displ       -2.333261 0.6373304 -3.660991 4.651570e- 4
##  6     6 drvf         4.570840 0.6012367  7.602397 6.789988e-11
##  7     6 drvr         6.384355 1.229277   5.193585 1.713129e- 6
##  8     8 (Intercept) 14.82265  2.887289   5.133759 2.708515e- 6
##  9     8 displ        0.3060487 0.5719058  0.5351383 5.943528e- 1
## 10     8 drvf         8.555294 2.679129   3.193311 2.156229e- 3
## 11     8 drvr         3.709336 0.7319048  5.068058 3.473594e- 6
```

When else might we use this?

Any sort of web scraping – pages change and URLs don't always work

Example

```r
library(rvest)
links <- list(
  "https://en.wikipedia.org/wiki/FC_Barcelona",
  "https://nosuchpage",
  "https://en.wikipedia.org/wiki/Rome"
)
pages <- map(links, ~{
  Sys.sleep(0.1)
  read_html(.x)
})
```

```
## Error in open.connection(x, "rb"): Could not resolve host: nosuchpage
```

The problem

I can't connect to https://nosuchpage because it doesn't exist

BUT

That also means I can't get *any* of my links because *one* page errored (imagine it was 1 in 1,000 instead of 1 in 3)

`safely()` to the rescue

Safe version

```
safe_read_html <- safely(read_html)
pages <- map(links, ~{
  Sys.sleep(0.1)
  safe_read_html(.x)
})
str(pages)
```

```
## List of 3
##  $ :List of 2
##   ..$ result:List of 2
##   .. ..$ node:<externalptr>
##   .. ..$ doc :<externalptr>
##   .. ..- attr(*, "class")= chr [1:2] "xml_document" "xml_node"
##   ..$ error : NULL
##  $ :List of 2
##   ..$ result: NULL
##   ..$ error :List of 2
##   .. ..$ message: chr "Could not resolve host: nosuchpage"
##   .. ..$ call   : language open.connection(x, "rb")
##   .. ..- attr(*, "class")= chr [1:3] "simpleError" "error" "condition"
##  $ :List of 2
##   ..$ result:List of 2
##   .. ..$ node:<externalptr>
##   .. ..$ doc :<externalptr>
##   .. ..- attr(*, "class")= chr [1:2] "xml_document" "xml_node"
```

Non-results

In a real example, we'd probably want to double-check the pages where we got no results

```
errors <- map_lgl(pages, ~!is.null(.x$error))
links[errors]
```

```
## [[1]]
## [1] "https://nosuchpage"
```

reduce

Reducing a list

The `map()` family of functions will always return a vector the same length as the input

`reduce()` will collapse or reduce the list to a single element

Example

```r
l <- list(
  c(1, 3),
  c(1, 5, 7, 9),
  3,
  c(4, 8, 12, 2)
)

reduce(l, sum)
```

```
## [1] 55
```

What's going on?

The code `reduce(l, sum)` is the same as

```
sum(l[[4]], sum(l[[3]], sum(l[[1]], l[[2]])))
```

```
## [1] 55
```

Or slidghlty differently

```
first_sum <- sum(l[[1]], l[[2]])
second_sum <- sum(first_sum, l[[3]])
final_sum <- sum(second_sum, l[[4]])
final_sum
```

```
## [1] 55
```

Why might you use this?

What if you had a list of data frames like this

```
l_df <- list(
  tibble(id = 1:3, score = rnorm(3)),
  tibble(id = 1:5, treatment = rbinom(5, 1, .5)),
  tibble(id = c(1, 3, 5, 7), other_thing = rnorm(4))
)
```

We can join these all together with a single loop – we want the output to be of length 1!

```r
reduce(l_df, full_join)
```

```
## # A tibble: 6 × 4
##      id     score treatment other_thing
##   <dbl>     <dbl>     <int>       <dbl>
## 1     1  0.4766972         0  -1.639860
## 2     2  0.1374334         1  NA
## 3     3 -1.199778          1  -1.084139
## 4     4 NA                 0  NA
## 5     5 NA                 1   0.1617546
## 6     7 NA                NA  -1.353905
```

Note – you have to be careful on directionality

```
reduce(l_df, left_join)
```

```
## # A tibble: 3 × 4
##      id      score treatment other_thing
##   <dbl>      <dbl>     <int>       <dbl>
## 1     1  0.4766972         0   -1.639860
## 2     2  0.1374334         1   NA
## 3     3 -1.199778          1   -1.084139
```

```
reduce(l_df, right_join)
```

```
## # A tibble: 4 × 4
##      id      score treatment other_thing
##   <dbl>      <dbl>     <int>       <dbl>
## 1     1  0.4766972         0  -1.639860
## 2     3 -1.199778          1  -1.084139
## 3     5 NA                 1   0.1617546
## 4     7 NA                NA  -1.353905
```

Another example

You probably just want to `bind_rows()`

```
l_df2 <- list(
  tibble(id = 1:3, scid = 1, score = rnorm(3)),
  tibble(id = 1:5, scid = 2, score = rnorm(5)),
  tibble(id = c(1, 3, 5, 7), scid = 3, score = rnorm(4))
)
reduce(l_df2, bind_rows)
```

```
## # A tibble: 12 × 3
##       id  scid      score
##    <dbl> <dbl>      <dbl>
##  1     1     1  0.5484699
##  2     2     1 -0.9978538
##  3     3     1 -0.7118563
##  4     1     2  0.6336533
##  5     2     2 -0.1875615
##  6     3     2 -0.2686937
##  7     4     2  0.4248864
##  8     5     2  0.8668824
##  9     1     3 -0.1675307
## 10     3     3  1.418294
## 11     5     3 -0.8494587
## 12     7     3 -1.575979
```

Non–loop version

Luckily, the prior slide has become obsolete, because `bind_rows()` will do the list reduction for us.

```
bind_rows(l_df2)
```

```
## # A tibble: 12 × 3
##        id  scid      score
##     <dbl> <dbl>      <dbl>
##  1      1     1  0.5484699
##  2      2     1 -0.9978538
##  3      3     1 -0.7118563
##  4      1     2  0.6336533
##  5      2     2 -0.1875615
##  6      3     2 -0.2686937
##  7      4     2  0.4248864
##  8      5     2  0.8668824
##  9      1     3 -0.1675307
## 10      3     3  1.418294
## 11      5     3 -0.8494587
## 12      7     3 -1.575979
```

Another example

This is a poor example, but there are use cases like this

```
library(palmerpenguins)
map(penguins, as.character) %>%
  reduce(paste)
```

```
##   [1] "Adelie Torgersen 39.1 18.7 181 3750 male 2007"   "Adelie Torgerse
##   [3] "Adelie Torgersen 40.3 18 195 3250 female 2007"   "Adelie Torgerse
##   [5] "Adelie Torgersen 36.7 19.3 193 3450 female 2007" "Adelie Torgerse
##   [7] "Adelie Torgersen 38.9 17.8 181 3625 female 2007" "Adelie Torgerse
##   [9] "Adelie Torgersen 34.1 18.1 193 3475 NA 2007"     "Adelie Torgerse
##  [11] "Adelie Torgersen 37.8 17.1 186 3300 NA 2007"     "Adelie Torgerse
##  [13] "Adelie Torgersen 41.1 17.6 182 3200 female 2007" "Adelie Torgerse
##  [15] "Adelie Torgersen 34.6 21.1 198 4400 male 2007"   "Adelie Torgerse
##  [17] "Adelie Torgersen 38.7 19 195 3450 female 2007"   "Adelie Torgerse
##  [19] "Adelie Torgersen 34.4 18.4 184 3325 female 2007" "Adelie Torgerse
##  [21] "Adelie Biscoe 37.8 18.3 174 3400 female 2007"    "Adelie Biscoe 3
##  [23] "Adelie Biscoe 35.9 19.2 189 3800 female 2007"    "Adelie Biscoe 3
##  [25] "Adelie Biscoe 38.8 17.2 180 3800 male 2007"      "Adelie Biscoe 3
##  [27] "Adelie Biscoe 40.6 18.6 183 3550 male 2007"      "Adelie Biscoe 4
##  [29] "Adelie Biscoe 37.9 18.6 172 3150 female 2007"    "Adelie Biscoe 4
##  [31] "Adelie Dream 39.5 16.7 178 3250 female 2007"     "Adelie Dream 37
##  [33] "Adelie Dream 39.5 17.8 188 3300 female 2007"     "Adelie Dream 40
##  [35] "Adelie Dream 36.4 17 195 3325 female 2007"       "Adelie Dream 39
```

Why use `reduce()`

This is one that I use a fair bit, but have a hard time coming up with good examples for.

The tidyverse makes it less needed, generally.

Still a good "tool" to have

Wrap up

- Lots more to `{purrr}` but we've covered a lot

- Functional programming can *really* help your efficiency, and even if it slows you down initially, I'd recommend always striving toward it, because it will ultimately be a huge help.

Questions?

If we have any time left – let's work on the homework

Next time

Functions

Beginning next class, the focus of the course will shift