Lab 04: BMI 5/625

Working with Tidy Data

Alison Hill (with modifications by Steven Bedrick)

Let's review

Data wrangling to date!

From `dplyr`:

- `filter`
- `arrange`
- `mutate`
- `group_by`
- `summarize`
- `glimpse`
- `distinct`
- `count`
- `tally`
- `pull`
- `top_n`
- `case_when`

Let's add from `dplyr`:

- `select`

From `tidyr`:

- `pivot_longer`
- `pivot_wider`

Plus 1 other package:

- `skimr::skim`

The Great British Baking Data Set

```
Rows: 1,772
Columns: 5
$ series    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
$ episode   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
$ baker     <chr> "Annetha", "David", "Edd", "Jasminder", "Jonathan", "Loui
$ challenge <chr> "signature", "signature", "signature", "signature", "sign
$ cake      <chr> "cake", "cake", "cake", "cake", "cake", "cake", "cake", "
```

Un-tidy cakes

```
# A tibble: 2 × 4
  series challenge    cake pie_tart
  <fct>  <chr>       <dbl>    <dbl>
1 1      showstopper     5        5
2 1      signature      12        4
```

```
# A tibble: 2 × 4
  series challenge    cake pie_tart
  <fct>  <chr>       <dbl>    <dbl>
1 3      showstopper    12       17
2 3      signature      24       12
```

```
# A tibble: 2 × 4
  series challenge    cake pie_tart
  <fct>  <chr>       <dbl>    <dbl>
1 2      showstopper     8       17
2 2      signature      21        7
```

```
# A tibble: 2 × 4
  series challenge    cake pie_tart
  <fct>  <chr>       <dbl>    <dbl>
1 4      showstopper    27        9
2 4      signature      11       15
```

Four seasons, four datasets...

Each row: a challenge type ("signature" vs. "showstopper") and a count of entries by type

Still un-tidy cakes

```
cakes_untidy %>%
  bind_rows()
```

At least now it's a single dataframe...

```
# A tibble: 16 × 4
   series challenge    cake pie_tart
   <fct>  <chr>       <dbl>    <dbl>
 1 1      showstopper     5        5
 2 1      signature      12        4
 3 2      showstopper     8       17
 4 2      signature      21        7
 5 3      showstopper    12       17
 6 3      signature      24       12
 7 4      showstopper    27        9
 8 4      signature      11       15
 9 5      showstopper    20        6
10 5      signature       4        7
11 6      showstopper    12        0
12 6      signature      20       17
13 7      showstopper    19        3
14 7      signature      11       10
15 8      showstopper    26       12
```

Finally tidy cakes

```
cakes_tidy <- cakes_untidy %>%
  pivot_longer(cake:pie_tart,
               names_to="bake_type",
               values_to="num_bakes") %>%
  arrange(series)
cakes_tidy
```

```
# A tibble: 32 × 4
   series challenge   bake_type num_bakes
   <fct>  <chr>       <chr>         <dbl>
 1 1      showstopper cake              5
 2 1      showstopper pie_tart          5
 3 1      signature   cake             12
 4 1      signature   pie_tart          4
 5 2      showstopper cake              8
 6 2      showstopper pie_tart         17
 7 2      signature   cake             21
 8 2      signature   pie_tart          7
 9 3      showstopper cake             12
10 3      showstopper pie_tart         17
# ℹ 22 more rows
```

What about changing types?

```
cakes_tidy <- cakes_untidy %>%
  pivot_longer(cake:pie_tart,
               names_to="bake_type",
               names_transform = list(bake_type=as.factor),
               values_to="num_bakes") %>%
  arrange(series)
cakes_tidy
```

```
# A tibble: 32 × 4
   series challenge   bake_type num_bakes
   <fct>  <chr>       <fct>         <dbl>
 1 1      showstopper cake              5
 2 1      showstopper pie_tart          5
 3 1      signature   cake             12
 4 1      signature   pie_tart          4
 5 2      showstopper cake              8
 6 2      showstopper pie_tart         17
 7 2      signature   cake             21
 8 2      signature   pie_tart          7
 9 3      showstopper cake             12
10 3      showstopper pie_tart         17
# ℹ 22 more rows
```

Know Your Tidy Data

```r
glimpse(cakes_tidy)
```

```
Rows: 32
Columns: 4
$ series    <fct> 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5,
$ challenge <chr> "showstopper", "showstopper", "signature", "signature", "
$ bake_type <fct> cake, pie_tart, cake, pie_tart, cake, pie_tart, cake, pie
$ num_bakes <dbl> 5, 5, 12, 4, 8, 17, 21, 7, 12, 17, 24, 12, 27, 9, 11, 15,
```

```r
library(skimr)
skim(cakes_tidy)
```

Name	cakes_tidy
Number of rows	32
Number of columns	4
_	
Column type frequency:	
character	1
factor	2
numeric	1
__	
Group variables	None

```r
library(skimr)
skim(cakes_tidy)
```

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
challenge	0	1	9	11	0	2	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
series	0	1	FALSE	8	1: 4, 2: 4, 3: 4, 4: 4
bake_type	0	1	FALSE	2	cak: 16, pie: 16

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
num_bakes	0	1	12.56	7.1	0	7	12	17.5	27	

```
skim(cakes_tidy) %>%
  summary()
```

Table: Data summary

Name	cakes_tidy
Number of rows	32
Number of columns	4
_	
Column type frequency:	
character	1
factor	2
numeric	1
__	
Group variables	None

Benefits of Tidy Data

```r
cakes_tidy %>%
  count(challenge, bake_type, wt = num_bakes, sort = TRUE)
```

```
# A tibble: 4 × 3
  challenge   bake_type     n
  <chr>       <fct>     <dbl>
1 showstopper cake        129
2 signature   cake        124
3 signature   pie_tart     80
4 showstopper pie_tart     69
```

```r
cakes_tidy %>%
  count(series, bake_type, wt = num_bakes)
```

```
# A tibble: 16 × 3
   series bake_type     n
   <fct>  <fct>     <dbl>
 1 1      cake         17
 2 1      pie_tart      9
 3 2      cake         29
 4 2      pie_tart     24
 5 3      cake         36
 6 3      pie_tart     29
 7 4      cake         38
 8 4      pie_tart     24
 9 5      cake         24
10 5      pie_tart     13
11 6      cake         32
12 6      pie_tart     17
13 7      cake         30
14 7      pie_tart     13
15 8      cake         47
16 8      pie_tart     20
```

```r
library(skimr)

cakes_tidy %>%
  group_by(bake_type) %>%
  select_if(is.numeric) %>%
  skim() %>% summary
```

Table: Data summary

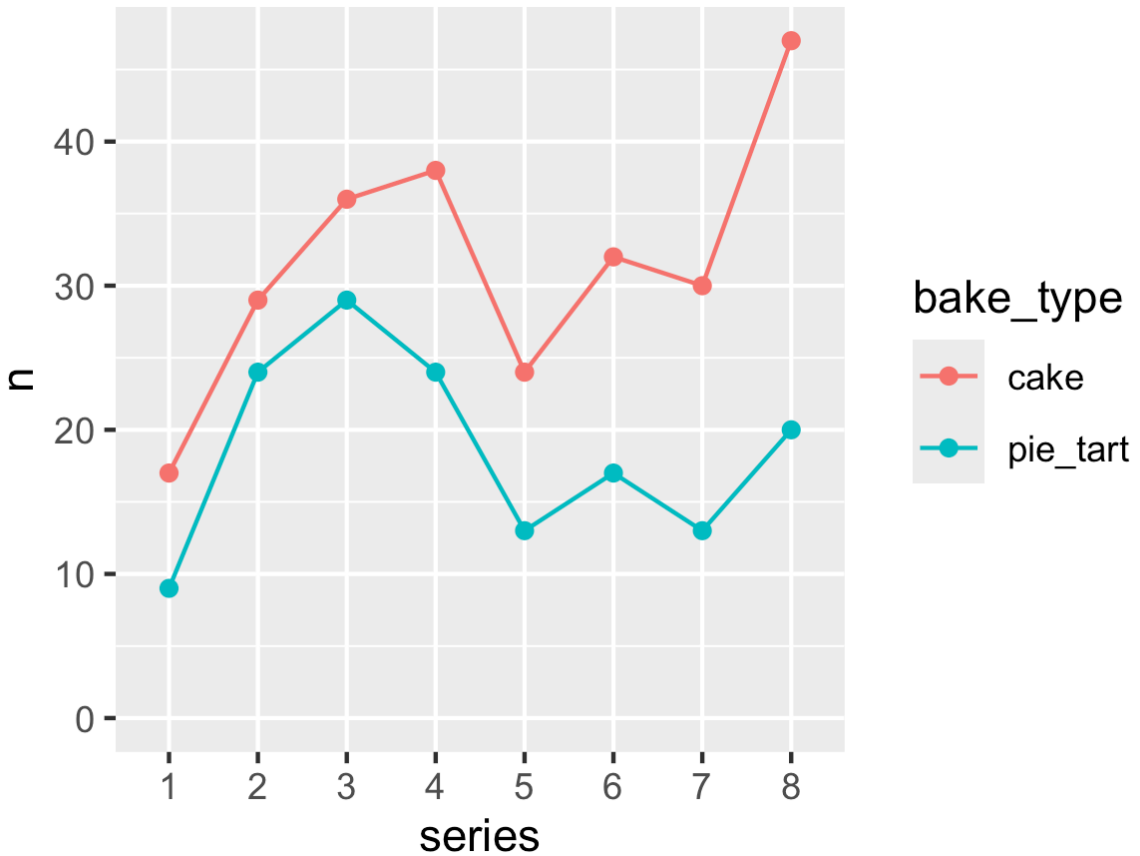
Name	Piped data
Number of rows	32
Number of columns	2
_	
Column type frequency:	
numeric	1
__	
Group variables	bake_type

See: https://suzanbaert.netlify.com/2018/01/dplyr-tutorial-1/

```r
cakes_by_series <- cakes_tidy %>%
  count(series, bake_type, wt = num_bakes)
cakes_by_series
```

```
# A tibble: 16 × 3
   series bake_type     n
   <fct>  <fct>     <dbl>
 1 1      cake         17
 2 1      pie_tart      9
 3 2      cake         29
 4 2      pie_tart     24
 5 3      cake         36
 6 3      pie_tart     29
 7 4      cake         38
 8 4      pie_tart     24
 9 5      cake         24
10 5      pie_tart     13
11 6      cake         32
12 6      pie_tart     17
13 7      cake         30
14 7      pie_tart     13
15 8      cake         47
16 8      pie_tart     20
```

```r
ggplot(cakes_by_series, aes(x = series, y = n,
                            color = bake_type,
                            group = bake_type)) +
  geom_point() +
  geom_line() +
  expand_limits(y = 0)
```



`select()`: Your new best friend

Selection Helpers

`dplyr` gives us helpful syntax for selecting columns:

```
cakes_raw %>% head(4)
```

```
# A tibble: 4 × 5
  series episode baker     challenge cake
   <dbl>   <dbl> <chr>     <chr>     <chr>
1      1       1 Annetha   signature cake
2      1       1 David     signature cake
3      1       1 Edd       signature cake
4      1       1 Jasminder signature cake
```

What if we only want *some* of the columns?

`dplyr::select()` to the rescue

```
cakes_raw %>% select(cake)
```

```
# A tibble: 1,772 × 1
   cake
   <chr>
 1 cake
 2 cake
 3 cake
 4 cake
 5 cake
 6 cake
 7 cake
 8 cake
 9 cake
10 <NA>
# ℹ 1,762 more rows
```

`dplyr::select()` to the rescue

```
cakes_raw %>% select(cake, baker) %>% head(4)
```

```
# A tibble: 4 × 2
  cake  baker
  <chr> <chr>
1 cake  Annetha
2 cake  David
3 cake  Edd
4 cake  Jasminder
```

But this is only the beginning!

... All columns *other* than `cake`

```
cakes_raw %>% select(!cake) %>% head(4)
```

```
# A tibble: 4 × 4
  series episode baker     challenge
   <dbl>   <dbl> <chr>     <chr>
1      1       1 Annetha   signature
2      1       1 David     signature
3      1       1 Edd       signature
4      1       1 Jasminder signature
```

Columns that *start* with a string?

```
cakes_raw %>% select(starts_with("c"))
```

```
# A tibble: 1,772 × 2
   challenge cake
   <chr>     <chr>
 1 signature cake
 2 signature cake
 3 signature cake
 4 signature cake
 5 signature cake
 6 signature cake
 7 signature cake
 8 signature cake
 9 signature cake
10 signature <NA>
# ℹ 1,762 more rows
```

The last column...

```
cakes_raw %>% select(last_col()) %>% head(4)
```

```
# A tibble: 4 × 1
  cake
  <chr>
1 cake
2 cake
3 cake
4 cake
```

A *range* of contiguous columns

```
cakes_raw %>% select(baker:cake) %>% head(4)
```

```
# A tibble: 4 × 3
  baker     challenge cake
  <chr>     <chr>     <chr>
1 Annetha   signature cake
2 David     signature cake
3 Edd       signature cake
4 Jasminder signature cake
```

There are many other helpers:

Matching columns by name:

- `starts_with()`/`ends_with()`
- `contains()`
- `num_range()` (for matching numerical ranges: think columns named for years, quarters, etc.)

See the `select` help page for more examples...

Many Tidyverse functions work with select helpers

```
billboard %>% glimpse
```

```
Rows: 317
Columns: 79
$ artist       <chr> "2 Pac", "2Ge+her", "3 Doors Down", "3 Doors Down", "5
$ track        <chr> "Baby Don't Cry (Keep ... ", "The Hardest Part Of  ... ",
$ date.entered <date> 2000-02-26, 2000-09-02, 2000-04-08, 2000-10-21, 2000-
$ wk1          <dbl> 87, 91, 81, 76, 57, 51, 97, 84, 59, 76, 84, 57, 50, 71
$ wk2          <dbl> 82, 87, 70, 76, 34, 39, 97, 62, 53, 76, 84, 47, 39, 51
$ wk3          <dbl> 72, 92, 68, 72, 25, 34, 96, 51, 38, 74, 75, 45, 30, 28
$ wk4          <dbl> 77, NA, 67, 69, 17, 26, 95, 41, 28, 69, 73, 29, 28, 18
$ wk5          <dbl> 87, NA, 66, 67, 17, 26, 100, 38, 21, 68, 73, 23, 21, 1
$ wk6          <dbl> 94, NA, 57, 65, 31, 19, NA, 35, 18, 67, 69, 18, 19, 13
$ wk7          <dbl> 99, NA, 54, 55, 36, 2, NA, 35, 16, 61, 68, 11, 20, 11,
$ wk8          <dbl> NA, NA, 53, 59, 49, 2, NA, 38, 14, 58, 65, 9, 17, 1, 2
$ wk9          <dbl> NA, NA, 51, 62, 53, 3, NA, 38, 12, 57, 73, 9, 17, 1, 2
$ wk10         <dbl> NA, NA, 51, 61, 57, 6, NA, 36, 10, 59, 83, 11, 17, 2,
$ wk11         <dbl> NA, NA, 51, 61, 64, 7, NA, 37, 9, 66, 92, 1, 17, 2, 36
$ wk12         <dbl> NA, NA, 51, 59, 70, 22, NA, 37, 8, 68, NA, 1, 3, 3, 37
```

Many Tidyverse functions work with select helpers

```
billboard %>%
  pivot_longer(cols=starts_with("wk"),
               names_to = "week",
               names_prefix = "wk",
               values_to = "rank"
  ) %>% head(10)
```

```
# A tibble: 10 × 5
   artist track                   date.entered week   rank
   <chr>  <chr>                   <date>       <chr> <dbl>
 1 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   1        87
 2 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   2        82
 3 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   3        72
 4 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   4        77
 5 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   5        87
 6 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   6        94
 7 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   7        99
 8 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   8        NA
 9 2 Pac  Baby Don't Cry (Keep ... 2000-02-26   9        NA
```

`janitor`: Your *other* new best friend

Often, data comes to us in... a less than pristine state:

```
glimpse(gapmnd)
```

```
Rows: 213
Columns: 15
$ `Fixed broadband Internet subscribers (per 100 people)` <chr> "Afghanista
$ `1998`                                                  <dbl> NA, NA, NA,
$ `1999`                                                  <dbl> NA, NA, NA,
$ `2000`                                                  <dbl> NA, NA, NA,
$ `2001`                                                  <dbl> 0.000000000
$ `2002`                                                  <dbl> 0.000000000
$ `2003`                                                  <dbl> 0.000000e+0
$ `2004`                                                  <dbl> 6.880265e-0
$ `2005`                                                  <dbl> 7.356639e-0
$ `2006`                                                  <dbl> 0.001625928
$ `2007`                                                  <dbl> 0.001581161
$ `2008`                                                  <dbl> 0.001537626
$ `2009`                                                  <dbl> 0.00299058,
$ `2010`                                                  <dbl> 0.004362367
$ `2011`                                                  <lgl> NA, NA, NA,
```

Note the very inconvenient column names...

The `janitor` package is here to help!

```
gapmnd %>% janitor::clean_names() %>% glimpse()
```

```
Rows: 213
Columns: 15
$ fixed_broadband_internet_subscribers_per_100_people <chr> "Afghanistan",
$ x1998                                                <dbl> NA, NA, NA, NA,
$ x1999                                                <dbl> NA, NA, NA, NA,
$ x2000                                                <dbl> NA, NA, NA, NA,
$ x2001                                                <dbl> 0.0000000000, 0
$ x2002                                                <dbl> 0.0000000000, 0
$ x2003                                                <dbl> 0.000000e+00, 0
$ x2004                                                <dbl> 6.880265e-04, 0
$ x2005                                                <dbl> 7.356639e-04, 8
$ x2006                                                <dbl> 0.001625928, NA
$ x2007                                                <dbl> 0.001581161, 0.
$ x2008                                                <dbl> 0.001537626, 2.
$ x2009                                                <dbl> 0.00299058, 2.8
$ x2010                                                <dbl> 0.004362367, 3.
$ x2011                                                <lgl> NA, NA, NA, NA,
```

`janitor` has many other capabilities...

- Transforming columns
- Removing empty rows/columns
- Collapsing sets of values values to `NA`, as needed

It also has a very nice cross-tabulation syntax (`tabyl()`)!

You have 2 *challenges* today!

Described here

Also see a reference walkthrough here

Tidy Data:

http://r4ds.had.co.nz/tidy-data.html

http://moderndive.com/4-tidy.html

http://vita.had.co.nz/papers/tidy-data.html

https://github.com/jennybc/lotr-tidy#readme