Lecture 14: Reshaping data

So far

- `select`: choose certain columns
- `filter`: choose certain rows
- `summarize`: calculate summary statistics
- `group_by`: group rows together
- `mutate`: create new columns
- `count`: count the number of rows
- `arrange`: re-order the rows
- `across`: apply functions across columns

~~want~~

Back to the dog data

```r
sc_data <- cleaned_data |>
  select(RID, GroupAssignment, sc_pre, sc_post)

sc_data
```

```
   RID GroupAssignment  sc_pre  sc_post
1    1         Control 3.900000 3.800000
2    2          Direct 5.150000 5.263158
3    3        Indirect 4.100000 4.150000
4    4         Control 4.650000 5.100000
5    5          Direct 3.650000 3.600000
6    6        Indirect 4.350000 4.650000
7    7         Control 4.750000 4.400000
8    8          Direct 4.600000 4.650000
9    9        Indirect 4.200000 4.150000
10  10         Control 5.800000 5.750000
11  11          Direct 4.400000 4.800000
12  12        Indirect 4.100000 4.250000
13  13         Control 5.400000 5.600000
```

lm(sc_post ~ GroupAssignment)

↑ only looks at post-test scores

lm(sc_pre ~ GroupAssignment)

↑ only looks at pre-test scores

Question: What if we want to fit a model with this data?

$$SC_i = \beta_0 + \beta_1 Direct_i + \beta_2 Indirect_i + \beta_3 Post_i + \varepsilon_i$$

(Social connectedness as a function of Group Assignment and pre/post stage in the data)

Fitting a model

Want code that looks like this:

```
lm(score ~ GroupAssignment + stage, data = sc_data)
```

Problem: We don't have a column for stage! Or a column for score!

SC score

(control,
Direct,
Indirect)

(pre, post)

want:

	Group Assignment	Stage	SC score
1	Control	pre	3.9
1	control	post	3.8
2	Direct	pre	5.15
2	Direct	post	5.26

pivot_longer

take sc_pre , sc_post columns

names

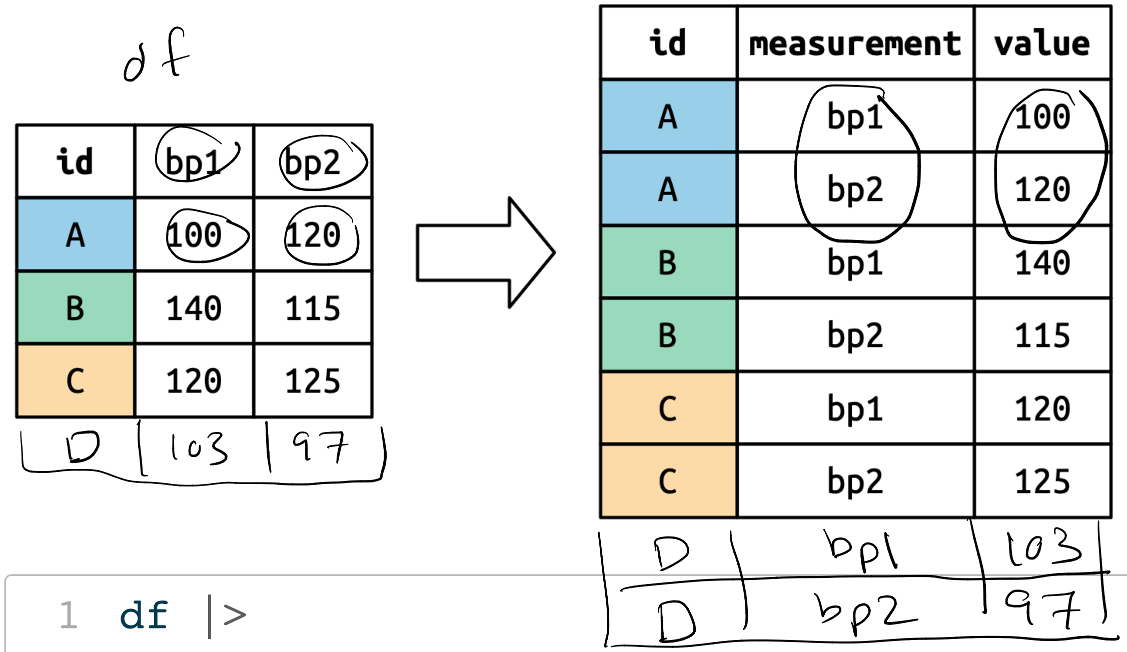
```r
sc_data |>
  pivot_longer(cols = c(sc_pre, sc_post),
               names_to = "stage",
               values_to = "score")
```

```
# A tibble: 568 × 4
     RID GroupAssignment stage   score
   <int> <chr>           <chr>   <dbl>
 1     1 Control         sc_pre   3.9
 2     1 Control         sc_post  3.8
 3     2 Direct          sc_pre   5.15
 4     2 Direct          sc_post  5.26
 5     3 Indirect        sc_pre   4.1
 6     3 Indirect        sc_post  4.15
 7     4 Control         sc_pre   4.65
 8     4 Control         sc_post  5.1
 9     5 Direct          sc_pre   3.65
10     5 Direct          sc_post  3.6
# ℹ 558 more rows
```

new column contains names of old columns

values in sc_pre, sc_post columns become a <u>new</u> column, called "score"

pivot_longer



df

id	bp1	bp2
A	100	120
B	140	115
C	120	125
D	103	97

id	measurement	value
A	bp1	100
A	bp2	120
B	bp1	140
B	bp2	115
C	bp1	120
C	bp2	125
D	bp1	103
D	bp2	97

```r
df |>
  pivot_longer(
    cols = bp1:bp2,
    names_to = "measurement",
    values_to = "value"
  )
```

(Image from *R for Data Science*)

pivot_longer

Another example:

```
# A tibble: 260 × 38
   Adult (15+) literacy rate …¹ `1975` `1976` `1977` `1978` `1979`
`1980` `1981`
   <chr>                          <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>  <dbl>
 1 Afghanistan                       NA     NA     NA     NA   4.99    NA
NA
 2 Albania                           NA     NA     NA     NA  NA       NA
NA
 3 Algeria                           NA     NA     NA     NA  NA       NA
NA
 4 Andorra                           NA     NA     NA     NA  NA       NA
NA
 5 Angola                            NA     NA     NA     NA  NA       NA
```

How might we want to restructure this data?

country	year	literacy_rate
Afghanistan	1975	NA
Afghanistan	1976	NA
Afghanistan	1977	:
	:	
	1979	4.99
		:

pivot_longer

```
# A tibble: 260 × 38
   Adult (15+) literacy rate …¹ `1975` `1976` `1977` `1978` `1979`
`1980` `1981`
   <chr>                         <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>  <dbl>
 1 Afghanistan                      NA     NA     NA     NA   4.99     NA
NA
 2 Albania                          NA     NA     NA     NA  NA        NA
NA
 3 Algeria                          NA     NA     NA     NA  NA        NA
NA
 4 Andorra                          NA     NA     NA     NA  NA        NA
NA
 5 Angola                           NA     NA     NA     NA  NA        NA
```

```r
litF |>
  rename(country = starts_with("Adult")) |>
  pivot_longer(
    cols = -country,   <- pivot all columns except country
    names_to = ...,    "year"
    values_to = ...    "literacy_rate"
  )
```

pivot_longer

```r
litF |>
  rename(country = starts_with("Adult")) |>
  pivot_longer(
    cols = -country,
    names_to = "year",
    values_to = "literacy_rate"
  ) |>
  drop_na(literacy_rate)
```

```
# A tibble: 571 × 3
   country     year  literacy_rate
   <chr>       <chr>         <dbl>
 1 Afghanistan 1979           4.99
 2 Afghanistan 2011          13   
 3 Albania     2001          98.3 
 4 Albania     2008          94.7 
 5 Albania     2011          95.7 
 6 Algeria     1987          35.8 
 7 Algeria     2002          60.1 
 8 Algeria     2006          63.9 
 9 Angola      2001          54.2 
10 Angola      2011          58.6 
# ℹ 561 more rows
```

pivot_longer

```r
litF |>
  rename(country = starts_with("Adult")) |>
  pivot_longer(
    cols = -country,
    names_to = "year",
    values_to = "literacy_rate",
    values_drop_na = T
  )
```

drop NAs in the new values column (here is literacy_rate)

```
# A tibble: 571 × 3
   country     year  literacy_rate
   <chr>       <chr>         <dbl>
 1 Afghanistan 1979           4.99
 2 Afghanistan 2011          13
 3 Albania     2001          98.3
 4 Albania     2008          94.7
 5 Albania     2011          95.7
 6 Algeria     1987          35.8
 7 Algeria     2002          60.1
 8 Algeria     2006          63.9
 9 Angola      2001          54.2
10 Angola      2011          58.6
# ℹ 561 more rows
```

Back to the dog data

```r
sc_data |>
  pivot_longer(cols = c(sc_pre, sc_post),
               names_to = "stage",
               values_to = "score")
```

```
# A tibble: 568 × 4
     RID GroupAssignment stage   score
   <int> <chr>           <chr>   <dbl>
 1     1 Control         sc_pre   3.9
 2     1 Control         sc_post  3.8
 3     2 Direct          sc_pre   5.15
 4     2 Direct          sc_post  5.26
 5     3 Indirect        sc_pre   4.1
 6     3 Indirect        sc_post  4.15
 7     4 Control         sc_pre   4.65
 8     4 Control         sc_post  5.1
 9     5 Direct          sc_pre   3.65
10     5 Direct          sc_post  3.6
# ℹ 558 more rows
```

Idea:

GroupAssignment	Type	Stage	Score
	SC	pre	3.9
	SC	post	38

stage

type of measurement

Does the `stage` column only contain information about stage?

Back to the dog data

sc pre sc post

```
sc_data |>
  pivot_longer(cols = c(sc_pre, sc_post),
               names_to = c("measurement", "stage"),
               names_sep = "_",
               values_to = "score")
```

← separate names of original columns by _

```
# A tibble: 568 × 5
     RID GroupAssignment measurement stage score
   <int> <chr>           <chr>       <chr> <dbl>
 1     1 Control         sc          pre    3.9
 2     1 Control         sc          post   3.8
 3     2 Direct          sc          pre    5.15
 4     2 Direct          sc          post   5.26
 5     3 Indirect        sc          pre    4.1
 6     3 Indirect        sc          post   4.15
 7     4 Control         sc          pre    4.65
 8     4 Control         sc          post   5.1
 9     5 Direct          sc          pre    3.65
10     5 Direct          sc          post   3.6
# ℹ 558 more rows
```

Working with all the measurements

```r
cleaned_data |>
  pivot_longer(cols = -c(RID, GroupAssignment),
               names_to = c("measurement", "stage"),
               names_sep = "_",
               values_to = "score")
```

← pivot all columns
~~except~~ RID & GroupAssign

← separating names by _

```
# A tibble: 4,544 × 5
     RID GroupAssignment measurement stage score
   <int> <chr>           <chr>       <chr> <dbl>
 1     1 Control         pa          pre    3.2
 2     1 Control         pa          post   3.8
 3     1 Control         happiness   pre    2.33
 4     1 Control         happiness   post   3.33
 5     1 Control         sc          pre    3.9
 6     1 Control         sc          post   3.8
 7     1 Control         fs          pre    6.12
 8     1 Control         fs          post   6
 9     1 Control         stress      pre    2
10     1 Control         stress      post   2
# ℹ 4,534 more rows
```

Fitting a model

```r
long_data <- cleaned_data |>
  pivot_longer(cols = -c(RID, GroupAssignment),
               names_to = c("measurement", "stage"),
               names_sep = "_",
               values_to = "score")

lm(score ~ GroupAssignment + stage, data = long_data)
```

```
Call:
lm(formula = score ~ GroupAssignment + stage, data = long_data)

Coefficients:
            (Intercept)    GroupAssignmentDirect  
GroupAssignmentIndirect  
                3.16307                 -0.10118  
-0.04836  
               stagepre  
                0.13805  
```

includes all measurements (pa, Stress, lonely, sc, etc.)

lm(pa ~ Group + Stage)

But what if I want to fit a *separate* model for each well-being/ill-being measurement?

idea:

Group Stage pa happiness sc ...

pivot_longer

```
# A tibble: 4,544 × 5
     RID GroupAssignment measurement stage score
   <int> <chr>           <chr>       <chr> <dbl>
 1     1 Control         pa          pre    3.2 
 2     1 Control         pa          post   3.8 
 3     1 Control         happiness   pre    2.33
 4     1 Control         happiness   post   3.33
 5     1 Control         sc          pre    3.9 
 6     1 Control         sc          post   3.8 
 7     1 Control         fs          pre    6.12
 8     1 Control         fs          post   6   
 9     1 Control         stress      pre    2   
10     1 Control         stress      post   2   
# ℹ 4,534 more rows
```

Perhaps we want to have a column for stage, and a column for each measurement?

pivot_longer

create a separate column for each of the first parts of the original column names

```
cleaned_data |>
  pivot_longer(cols = -c(RID, GroupAssignment),
               names_to = c(".value", "stage"),
               names_sep = "_")
```

pivot all the columns except for RID & Group

separate names by _

make a new column for stage contains the second part of the original column names

```
# A tibble: 568 × 11
      RID GroupAssignment stage    pa happiness    sc    fs stress
homesick
    <int> <chr>           <chr> <dbl>     <dbl> <dbl> <dbl>  <int>
<int>
 1      1 Control         pre     3.2      2.33  3.9   6.12      2
3
 2      1 Control         post    3.8      3.33  3.8   6         2
3
 3      2 Direct          pre     3        3.33  5.15  5.25      2
4
 4      2 Direct          post    3.2      4     5.26  6         1
2
 5      3 Indirect        pre     2.8      2.67  4.1   5.38      4
```

pa_pre pa_post sc_pre sc_post ...

↳ pa sc ... stage

pivot_wider

```r
long_data
```

```
# A tibble: 4,544 × 5
     RID GroupAssignment measurement stage score
   <int> <chr>           <chr>       <chr> <dbl>
 1     1 Control         pa          pre    3.2 
 2     1 Control         pa          post   3.8 
 3     1 Control         happiness   pre    2.33
 4     1 Control         happiness   post   3.33
 5     1 Control         sc          pre    3.9 
 6     1 Control         sc          post   3.8 
 7     1 Control         fs          pre    6.12
 8     1 Control         fs          post   6   
 9     1 Control         stress      pre    2   
10     1 Control         stress      post   2   
# ℹ 4,534 more rows
```

id columns

=> one row for each combo of id columns

1 Control pre

1 Control post

pivot_wider

```r
long_data |>
  pivot_wider(id_cols = c(RID, GroupAssignment, stage),
              names_from = measurement,
              values_from = score)
```

← columns that we don't pivot

← new columns from "measurement" column

↖ entries from "score" column

```
# A tibble: 568 × 11
     RID GroupAssignment stage    pa happiness    sc    fs  stress
homesick
   <int> <chr>           <chr> <dbl>     <dbl> <dbl> <dbl>   <dbl>
<dbl>
 1     1 Control         pre     3.2      2.33  3.9   6.12       2
3
 2     1 Control         post    3.8      3.33  3.8   6          2
3
 3     2 Direct          pre     3        3.33  5.15  5.25       2
4
 4     2 Direct          post    3.2      4     5.26  6          1
2
 5     3 Indirect        pre     2.8      2.67  4.1   5.38       4
```

Class activity

https://sta279-f23.github.io/class_activities/ca_lecture_14.html