

Tidying data

Tidying data

- Data rarely come to us as we want to use them.
- Before we can do analysis, typically have organizing to do.
- This is typical of ANOVA-type data, “wide format”:

| pig | feed1 | feed2 | feed3 | feed4 |
|-----|-------|-------|-------|-------|
| 1 | 60.8 | 68.7 | 92.6 | 87.9 |
| 2 | 57.0 | 67.7 | 92.1 | 84.2 |
| 3 | 65.0 | 74.0 | 90.2 | 83.1 |
| 4 | 58.6 | 66.3 | 96.5 | 85.7 |
| 5 | 61.7 | 69.8 | 99.1 | 90.3 |

- 20 pigs randomly allocated to one of four feeds. At end of study, weight of each pig is recorded.
- Are any differences in mean weights among the feeds?
- Problem: want all weights in one column, with 2nd column labelling which feed. Untidy!

Tidy and untidy data (Wickham)

- Data set easier to deal with if:
 - ▶ each observation is one row
 - ▶ each variable is one column
 - ▶ each type of observation unit is one table
- Data arranged this way called “tidy”; otherwise called “untidy”.
- For the pig data:
 - ▶ response variable is weight, but scattered over 4 columns, which are levels of a factor feed.
 - ▶ Want all the weights in one column, with a second column feed saying which feed that weight goes with.
 - ▶ Then we can run aov.

Packages for this section

```
library(tidyverse)
```

Reading in the pig data

```
my_url <- "http://ritsokiguess.site/datafiles/pigs1.txt"
pigs1 <- read_delim(my_url, " ")
pigs1
```

```
# A tibble: 5 x 5
  pig feed1 feed2 feed3 feed4
<dbl> <dbl> <dbl> <dbl> <dbl>
1     1  60.8  68.7  92.6  87.9
2     2   57   67.7  92.1  84.2
3     3   65   74   90.2  83.1
4     4  58.6  66.3  96.5  85.7
5     5  61.7  69.8  99.1  90.3
```

Making it longer

- We wanted all the weights in one column, labelled by which feed they went with.
- This is a very common reorganization, and the magic “verb” is `pivot_longer`:

```
pigs1 %>% pivot_longer(feed1:feed4, names_to="feed",  
                        values_to="weight") -> pigs2
```

The long dataframe pigs2

```
# A tibble: 20 x 3
  pig feed  weight
<dbl> <chr> <dbl>
1     1 feed1  60.8
2     1 feed2  68.7
3     1 feed3  92.6
4     1 feed4  87.9
5     2 feed1   57
6     2 feed2  67.7
7     2 feed3  92.1
8     2 feed4  84.2
9     3 feed1   65
10    3 feed2   74
11    3 feed3  90.2
12    3 feed4  83.1
13    4 feed1  58.6
14    4 feed2  66.3
15    4 feed3  96.5
16    4 feed4  85.7
17    5 feed1  61.7
18    5 feed2  69.8
19    5 feed3  99.1
20    5 feed4  90.3
```

Alternatives

Any way of choosing the columns to pivot longer is good, eg:

```
pigs1 %>% pivot_longer(-pig, names_to="feed",  
                        values_to="weight")
```

```
# A tibble: 20 x 3  
      pig feed  weight  
  <dbl> <chr>  <dbl>  
1     1 feed1   60.8  
2     1 feed2   68.7  
3     1 feed3   92.6  
4     1 feed4   87.9  
5     2 feed1    57  
6     2 feed2   67.7  
7     2 feed3   92.1  
8     2 feed4   84.2  
9     3 feed1    65  
10    3 feed2    74
```


Comments

- `pigs2` now in “long” format, ready for analysis.
- Anatomy of `pivot_longer`:
 - ▶ columns to combine
 - ▶ a name for column that will contain groups (“names”)
 - ▶ a name for column that will contain measurements (“values”)

Identifying the pigs

- Values in `pig` identify pigs *within each group*: pig 1 is four different pigs!
- Create unique pig IDs by gluing pig number onto feed:

```
pigs2 %>% mutate(pig_id=str_c(feed, "_", pig)) -> pigs2
```

The new pigs2

```
# A tibble: 20 x 4
```

| | pig | feed | weight | pig_id |
|----|-------|-------|--------|---------|
| | <dbl> | <chr> | <dbl> | <chr> |
| 1 | 1 | feed1 | 60.8 | feed1_1 |
| 2 | 1 | feed2 | 68.7 | feed2_1 |
| 3 | 1 | feed3 | 92.6 | feed3_1 |
| 4 | 1 | feed4 | 87.9 | feed4_1 |
| 5 | 2 | feed1 | 57 | feed1_2 |
| 6 | 2 | feed2 | 67.7 | feed2_2 |
| 7 | 2 | feed3 | 92.1 | feed3_2 |
| 8 | 2 | feed4 | 84.2 | feed4_2 |
| 9 | 3 | feed1 | 65 | feed1_3 |
| 10 | 3 | feed2 | 74 | feed2_3 |
| 11 | 3 | feed3 | 90.2 | feed3_3 |
| 12 | 3 | feed4 | 83.1 | feed4_3 |
| 13 | 4 | feed1 | 58.6 | feed1_4 |
| 14 | 4 | feed2 | 66.3 | feed2_4 |
| 15 | 4 | feed3 | 96.5 | feed3_4 |
| 16 | 4 | feed4 | 85.7 | feed4_4 |
| 17 | 5 | feed1 | 61.7 | feed1_5 |
| 18 | 5 | feed2 | 69.8 | feed2_5 |
| 19 | 5 | feed3 | 99.1 | feed3_5 |
| 20 | 5 | feed4 | 90.3 | feed4_5 |

...and finally, the analysis

- which is just what we saw before:

```
weight.1 <- aov(weight ~ feed, data = pigs2)
summary(weight.1)
```

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|-----------|----|--------|---------|---------|--------------|
| feed | 3 | 3521 | 1173.5 | 119.1 | 3.72e-11 *** |
| Residuals | 16 | 158 | 9.8 | | |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

- The mean weights of pigs on the different feeds are definitely not all equal.
- So we run Tukey to see which ones differ (over).

Tukey

```
TukeyHSD(weight.1)
```

Tukey multiple comparisons of means
95% family-wise confidence level

```
Fit: aov(formula = weight ~ feed, data = pigs2)
```

```
$feed
```

| | diff | lwr | upr | p adj |
|-------------|-------|------------|-----------|-----------|
| feed2-feed1 | 8.68 | 3.001038 | 14.358962 | 0.0024000 |
| feed3-feed1 | 33.48 | 27.801038 | 39.158962 | 0.0000000 |
| feed4-feed1 | 25.62 | 19.941038 | 31.298962 | 0.0000000 |
| feed3-feed2 | 24.80 | 19.121038 | 30.478962 | 0.0000000 |
| feed4-feed2 | 16.94 | 11.261038 | 22.618962 | 0.0000013 |
| feed4-feed3 | -7.86 | -13.538962 | -2.181038 | 0.0055599 |

All of the feeds differ!

Mean weights by feed

To find the best and worst, get mean weight by feed group. I borrowed an idea from earlier to put the means in descending order:

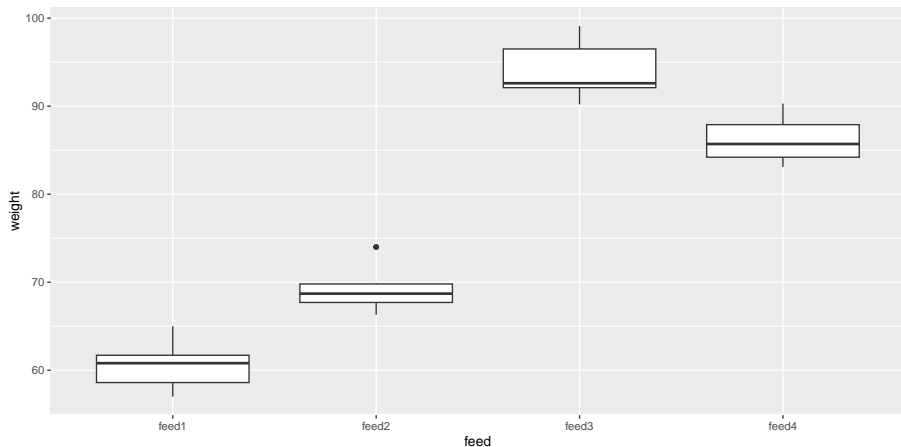
```
pigs2 %>%  
  group_by(feed) %>%  
  summarize(mean_weight = mean(weight))%>%  
  arrange(desc(mean_weight))
```

```
# A tibble: 4 x 2  
  feed mean_weight  
  <chr>      <dbl>  
1 feed3      94.1  
2 feed4      86.2  
3 feed2      69.3  
4 feed1      60.6
```

Feed 3 is best, feed 1 worst.

Should we have any concerns about the ANOVA?

```
ggplot(pigs2, aes(x = feed, y = weight)) + geom_boxplot()
```



Comments

- Feed 2 has an outlier
- But there are only 5 pigs in each group
- The conclusion is so clear that I am OK with this.

Tuberculosis

- The World Health Organization keeps track of number of cases of various diseases, eg. tuberculosis.
- Some data:

```
my_url <- "http://ritsokiguess.site/datafiles/tb.csv"  
tb <- read_csv(my_url)
```

The data (10 randomly chosen rows)

```
tb %>% slice_sample(n = 10)
```

```
# A tibble: 10 x 22
```

| | iso2 | year | m04 | m514 | m014 | m1524 | m2534 | m3544 | m4554 | m5564 | m65 |
|----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| | <chr> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | AI | 2003 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 2 | MR | 2003 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 3 | CR | 1980 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 4 | GA | 1988 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 5 | ET | 1996 | NA | NA | 302 | 1739 | 1609 | 854 | 427 | 201 | 71 |
| 6 | BA | 2007 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | HU | 1982 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 8 | AR | 2008 | 11 | 58 | 69 | 633 | 611 | 390 | 416 | 364 | 295 |
| 9 | TT | 1982 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 10 | LV | 1980 | NA | NA | NA | NA | NA | NA | NA | NA | NA |

```
# i 11 more variables: mu <dbl>, f04 <dbl>, f514 <dbl>, f014 <dbl>,  
#   f1524 <dbl>, f2534 <dbl>, f3544 <dbl>, f4554 <dbl>, f5564 <dbl>,  
#   f65 <dbl>, fu <dbl>
```

Many rows and columns

```
nrow(tb)
```

```
[1] 5769
```

```
ncol(tb)
```

```
[1] 22
```

What we have

- Variables: country (abbreviated), year. Then number of cases for each gender and age group, eg. m1524 is males aged 15–24. Also mu and fu, where age is unknown.
- Lots of missings. Want to get rid of.
- Abbreviations here.

```
tb %>%
```

```
  pivot_longer(m04:fu, names_to = "genage",  
               values_to = "freq", values_drop_na = TRUE) -> t
```

- Code for pivot_longer:
 - ▶ columns to make longer
 - ▶ column to contain the names (categorical)
 - ▶ column to contain the values (quantitative)
 - ▶ drop missings in the values

Results (some)

```
tb2
```

```
# A tibble: 35,750 x 4
  iso2   year genage  freq
<chr> <dbl> <chr>   <dbl>
1 AD     1996 m014      0
2 AD     1996 m1524     0
3 AD     1996 m2534     0
4 AD     1996 m3544     4
5 AD     1996 m4554     1
6 AD     1996 m5564     0
7 AD     1996 m65      0
8 AD     1996 f014     0
9 AD     1996 f1524     1
10 AD    1996 f2534     1
# i 35,740 more rows
```

Separating

- 4 columns, but 5 variables, since `genage` contains both gender and age group. Split that up using `separate`.
- `separate` needs to know:
 - ▶ what to separate (no quotes needed),
 - ▶ how to split, and what to separate into (here you do need quotes):

```
tb2 %>%  
  separate_wider_position(genage,  
                           widths = c("gender" = 1, "age" = 4),  
                           too_few = "align_start") -> tb3
```

Tidied tuberculosis data (some)

```
tb3
```

```
# A tibble: 35,750 x 5
```

| | iso2 | year | gender | age | freq |
|----|-------|-------|--------|-------|-------|
| | <chr> | <dbl> | <chr> | <chr> | <dbl> |
| 1 | AD | 1996 | m | 014 | 0 |
| 2 | AD | 1996 | m | 1524 | 0 |
| 3 | AD | 1996 | m | 2534 | 0 |
| 4 | AD | 1996 | m | 3544 | 4 |
| 5 | AD | 1996 | m | 4554 | 1 |
| 6 | AD | 1996 | m | 5564 | 0 |
| 7 | AD | 1996 | m | 65 | 0 |
| 8 | AD | 1996 | f | 014 | 0 |
| 9 | AD | 1996 | f | 1524 | 1 |
| 10 | AD | 1996 | f | 2534 | 1 |

```
# i 35,740 more rows
```

In practice...

- instead of doing the pipe one step at a time, you *debug* it one step at a time, and when you have each step working, you use that step's output as input to the next step, thus:

```
tb %>%  
  pivot_longer(m04:fu, names_to = "genage",  
               values_to = "freq", values_drop_na = TRUE) %>%  
  separate_wider_position(genage,  
                           widths = c("gender" = 1, "age" = 4),  
                           too_few = "align_start") -> tb_tidy
```

- When you have it working, save the final result (for further work).

Comments

- You can split the R code over as many lines as you like, as long as each line is incomplete, so that R knows more is to come.
- I like to put the pipe symbol on the end of the line.
- Sometimes one function call gets very long, in which case you can separate at commas.

Total tuberculosis cases by year (some of the years)

```
tb_tidy %>%  
  filter(between(year, 1991, 1998)) %>%  
  group_by(year) %>% summarize(total=sum(freq))
```

A tibble: 8 x 2

| | year | total |
|---|-------|--------|
| | <dbl> | <dbl> |
| 1 | 1991 | 544 |
| 2 | 1992 | 512 |
| 3 | 1993 | 492 |
| 4 | 1994 | 750 |
| 5 | 1995 | 513971 |
| 6 | 1996 | 635705 |
| 7 | 1997 | 733204 |
| 8 | 1998 | 840389 |

- Something very interesting happened between 1994 and 1995.

To find out what

- try counting up total cases by country:

```
tb_tidy %>% group_by(iso2) %>%  
  summarize(total=sum(freq)) %>%  
  arrange(desc(total))
```

```
# A tibble: 213 x 2
```

| | iso2 | total |
|----|-------|---------|
| | <chr> | <dbl> |
| 1 | CN | 4065174 |
| 2 | IN | 3966169 |
| 3 | ID | 1129015 |
| 4 | ZA | 900349 |
| 5 | BD | 758008 |
| 6 | VN | 709695 |
| 7 | CD | 603095 |
| 8 | PH | 490040 |
| 9 | BR | 440609 |
| 10 | KE | 431523 |

```
# i 203 more rows
```

What years do I have for China?

China started recording in 1995, which is at least part of the problem:

```
tb_tidy %>% filter(iso2 == "CN") %>%  
  group_by(year) %>%  
  summarize(total = sum(freq))
```

```
# A tibble: 14 x 2
```

| | year | total |
|----|-------|--------|
| | <dbl> | <dbl> |
| 1 | 1995 | 131194 |
| 2 | 1996 | 168270 |
| 3 | 1997 | 195895 |
| 4 | 1998 | 214404 |
| 5 | 1999 | 212258 |
| 6 | 2000 | 213766 |
| 7 | 2001 | 212766 |
| 8 | 2002 | 194972 |
| 9 | 2003 | 267280 |
| 10 | 2004 | 384886 |
| 11 | 2005 | 472719 |

First year of recording by country?

- A lot of countries started recording in about 1995, in fact:

```
tb_tidy %>% group_by(iso2) %>%  
  summarize(first_year=min(year)) %>%  
  count(first_year)
```

```
# A tibble: 14 x 2
```

| | first_year | n |
|----|------------|-------|
| | <dbl> | <int> |
| 1 | 1980 | 2 |
| 2 | 1994 | 2 |
| 3 | 1995 | 130 |
| 4 | 1996 | 31 |
| 5 | 1997 | 17 |
| 6 | 1998 | 6 |
| 7 | 1999 | 10 |
| 8 | 2000 | 4 |
| 9 | 2001 | 1 |
| 10 | 2002 | 3 |
| 11 | 2003 | 2 |

Comment

- So the reason for the big jump in cases is that so many countries started recording then, not that there really were more cases.

Some Toronto weather data

```
my_url <- "http://ritsokiguess.site/STAC32/toronto_weather.csv"
weather <- read_csv(my_url)
weather
```

```
# A tibble: 24 x 35
```

| | station | Year | Month | element | d01 | d02 | d03 | d04 | d05 | d06 | d07 |
|----|---------|-------|-------|---------|-------|-------|-------|-------|-------|-------|-------|
| | <chr> | <dbl> | <chr> | <chr> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | TORONT~ | 2018 | 01 | tmax | -7.9 | -7.1 | -5.3 | -7.7 | -14.7 | -15.4 | -1 |
| 2 | TORONT~ | 2018 | 01 | tmin | -18.6 | -12.5 | -11.2 | -19.7 | -20.6 | -22.3 | -17.5 |
| 3 | TORONT~ | 2018 | 02 | tmax | 5.6 | -8.6 | 0.4 | 1.8 | -6.6 | -3.2 | -4.1 |
| 4 | TORONT~ | 2018 | 02 | tmin | -8.9 | -15 | -9.7 | -8.8 | -12 | -8.2 | -8.7 |
| 5 | TORONT~ | 2018 | 03 | tmax | NA | NA | NA | NA | NA | NA | 3.1 |
| 6 | TORONT~ | 2018 | 03 | tmin | NA | -0.5 | NA | -3.1 | NA | -1.4 | 0.4 |
| 7 | TORONT~ | 2018 | 04 | tmax | 4.5 | 6.5 | 5 | 5.7 | 2.9 | 5.4 | 2 |
| 8 | TORONT~ | 2018 | 04 | tmin | -2.6 | -1.2 | 2.4 | -3.2 | -3.9 | -2.6 | -4.4 |
| 9 | TORONT~ | 2018 | 05 | tmax | 23.5 | 26.3 | 23 | 24 | 24.1 | 17.4 | 15.9 |
| 10 | TORONT~ | 2018 | 05 | tmin | 8.5 | 14.4 | 11.4 | 9.2 | 8.5 | 13.3 | 10.6 |

```
# i 14 more rows
```

```
# i 24 more variables: d08 <dbl>, d09 <dbl>, d10 <dbl>, d11 <dbl>,  
#   d12 <dbl>, d13 <dbl>, d14 <dbl>, d15 <dbl>, d16 <dbl>, d17 <dbl>,  
#   d18 <dbl>, d19 <dbl>, d20 <dbl>, d21 <dbl>, d22 <dbl>, d23 <dbl>,  
#   d24 <dbl>, d25 <dbl>, d26 <dbl>, d27 <dbl>, d28 <dbl>, d29 <dbl>
```

The columns

- Daily weather records for “Toronto City” weather station in 2018:
 - ▶ `station`: identifier for this weather station (always same here)
 - ▶ `Year`, `Month`
 - ▶ `element`: whether temperature given was daily max or daily min
 - ▶ `d01`, `d02`,... `d31`: day of the month from 1st to 31st.

Off we go

Numbers in data frame all temperatures (for different days of the month), so first step is

```
weather %>%  
  pivot_longer(d01:d31, names_to="day",  
               values_to="temperature",  
               values_drop_na = TRUE)
```

A tibble: 703 x 6

| | station | Year | Month | element | day | temperature |
|---|--------------|-------|-------|---------|-------|-------------|
| | <chr> | <dbl> | <chr> | <chr> | <chr> | <dbl> |
| 1 | TORONTO CITY | 2018 | 01 | tmax | d01 | -7.9 |
| 2 | TORONTO CITY | 2018 | 01 | tmax | d02 | -7.1 |
| 3 | TORONTO CITY | 2018 | 01 | tmax | d03 | -5.3 |
| 4 | TORONTO CITY | 2018 | 01 | tmax | d04 | -7.7 |
| 5 | TORONTO CITY | 2018 | 01 | tmax | d05 | -14.7 |
| 6 | TORONTO CITY | 2018 | 01 | tmax | d06 | -15.4 |
| 7 | TORONTO CITY | 2018 | 01 | tmax | d07 | -1 |
| 8 | TORONTO CITY | 2018 | 01 | tmax | d08 | 3 |
| 9 | TORONTO CITY | 2018 | 01 | tmax | d09 | 1.6 |

Element

- Column `element` contains names of two different variables, that should each be in separate column.
- Distinct from eg. `m1524` in tuberculosis data, that contained levels of two different factors, handled by `separate`.
- Untangling names of variables handled by `pivot_wider`.

Handling element

```
weather %>%  
  pivot_longer(d01:d31, names_to="day",  
               values_to="temperature",  
               values_drop_na = TRUE) %>%  
  pivot_wider(names_from=element,  
              values_from=temperature)
```

```
# A tibble: 355 x 6
```

| | station | Year | Month | day | tmax | tmin |
|---|--------------|-------|-------|-------|-------|-------|
| | <chr> | <dbl> | <chr> | <chr> | <dbl> | <dbl> |
| 1 | TORONTO CITY | 2018 | 01 | d01 | -7.9 | -18.6 |
| 2 | TORONTO CITY | 2018 | 01 | d02 | -7.1 | -12.5 |
| 3 | TORONTO CITY | 2018 | 01 | d03 | -5.3 | -11.2 |
| 4 | TORONTO CITY | 2018 | 01 | d04 | -7.7 | -19.7 |
| 5 | TORONTO CITY | 2018 | 01 | d05 | -14.7 | -20.6 |
| 6 | TORONTO CITY | 2018 | 01 | d06 | -15.4 | -22.3 |
| 7 | TORONTO CITY | 2018 | 01 | d07 | -1 | -17.5 |

Further improvements 1/2

- We have tidy data now, but can improve things further.
- `mutate` creates new columns from old (or assign back to change a variable).
- Would like numerical dates. `separate` works, or pull out number as below.
- `select` keeps columns (or drops, with `minus`). Station name has no value to us.

Further improvements 2/2

```
weather %>%  
  pivot_longer(d01:d31, names_to="day",  
               values_to="temperature", values_drop_na = TRUE) %>%  
  pivot_wider(names_from=element, values_from=temperature) %>%  
  mutate(Day = parse_number(day)) %>%  
  select(-station)
```

```
# A tibble: 355 x 6
```

| | Year | Month | day | tmax | tmin | Day |
|----|-------|-------|-------|-------|-------|-------|
| | <dbl> | <chr> | <chr> | <dbl> | <dbl> | <dbl> |
| 1 | 2018 | 01 | d01 | -7.9 | -18.6 | 1 |
| 2 | 2018 | 01 | d02 | -7.1 | -12.5 | 2 |
| 3 | 2018 | 01 | d03 | -5.3 | -11.2 | 3 |
| 4 | 2018 | 01 | d04 | -7.7 | -19.7 | 4 |
| 5 | 2018 | 01 | d05 | -14.7 | -20.6 | 5 |
| 6 | 2018 | 01 | d06 | -15.4 | -22.3 | 6 |
| 7 | 2018 | 01 | d07 | -1 | -17.5 | 7 |
| 8 | 2018 | 01 | d08 | 3 | -1.7 | 8 |
| 9 | 2018 | 01 | d09 | 1.6 | -0.6 | 9 |
| 10 | 2018 | 01 | d10 | 5.9 | -1.3 | 10 |

Final step(s)

- Make year-month-day into proper date.
- Keep only date, tmax, tmin:

```
weather %>%  
  pivot_longer(d01:d31, names_to="day",  
               values_to="temperature", values_drop_na = T) %>%  
  pivot_wider(names_from=element, values_from=temperature) %>%  
  mutate(Day = parse_number(day)) %>%  
  select(-station) %>%  
  unite(datestr, c(Year, Month, Day), sep = "-") %>%  
  mutate(date = as.Date(datestr)) %>%  
  select(c(date, tmax, tmin)) -> weather_tidy
```

Our tidy data frame

```
weather_tidy
```

```
# A tibble: 355 x 3
  date      tmax tmin
<date>    <dbl> <dbl>
1 2018-01-01  -7.9 -18.6
2 2018-01-02  -7.1 -12.5
3 2018-01-03  -5.3 -11.2
4 2018-01-04  -7.7 -19.7
5 2018-01-05 -14.7 -20.6
6 2018-01-06 -15.4 -22.3
7 2018-01-07  -1   -17.5
8 2018-01-08   3    -1.7
9 2018-01-09   1.6  -0.6
10 2018-01-10   5.9  -1.3
# i 345 more rows
```

Plotting the temperatures

- Plot temperature against date joined by lines, but with separate lines for max and min. `ggplot` requires something like

```
ggplot(..., aes(x = date, y = temperature)) + geom_point() +  
  geom_line()
```

only we have two temperatures, one a max and one a min, that we want to keep separate.

- The trick: combine `tmax` and `tmin` together into one column, keeping track of what kind of temp they are. (This actually same format as untidy weather.) Are making `weather_tidy` untidy for purposes of drawing graph only.
- Then can do something like

```
ggplot(d, aes(x = date, y = temperature, colour = maxmin))  
  + geom_point() + geom_line()
```

to distinguish max and min on graph.

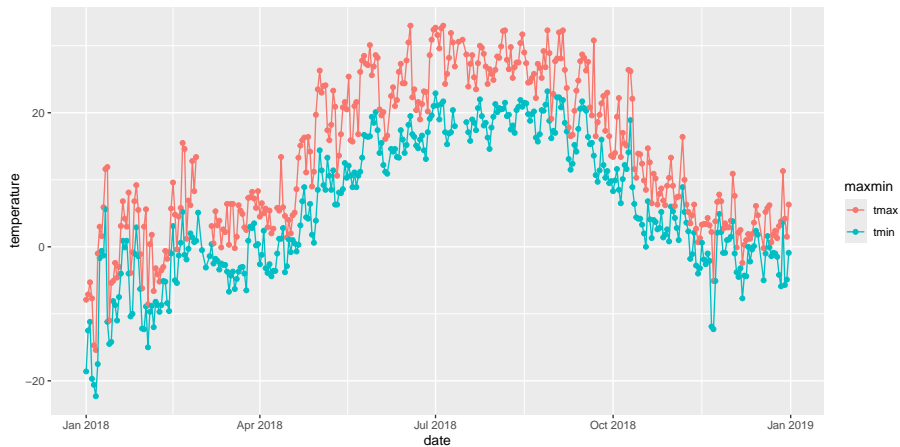
Setting up plot

- Since we only need data frame for plot, we can do the column-creation and plot in a pipeline.
- For a `ggplot` in a pipeline, the initial data frame is omitted, because it is whatever came out of the previous step.
- To make those “one column”s: `pivot_longer`. I save the graph to show overleaf:

```
weather_tidy %>%  
  pivot_longer(tmax:tmin, names_to="maxmin",  
               values_to="temperature") %>%  
  ggplot(aes(x = date, y = temperature, colour = maxmin)) +  
    geom_point() + geom_line() -> g
```

The plot

gg



Summary of tidying “verbs”

| Verb | Purpose |
|---------------------------|--|
| <code>pivot_longer</code> | Combine columns that measure same thing into one |
| <code>pivot_wider</code> | Take column that measures one thing under different conditions and put into multiple columns |
| <code>separate</code> | Turn a column that encodes several variables into several columns |
| <code>unite</code> | Combine several (related) variables into one “combination” variable |

`pivot_longer` and `pivot_wider` are opposites; `separate` and `unite` are opposites.