Tidying and organizing 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)
library(readxl)
```

Reading in the pig data

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

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

Gathering up the columns

- 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
```

- `pigs2` is now in "long" format, ready for analysis. See next page.
- Anatomy of `pivot_longer`:
  - columns to combine
  - a name for column that will contain groups
  - a name for column that will contain measurements

Long format pigs

```
pigs2
```

pig	feed	weight
1	feed1	60.8
1	feed2	68.7
1	feed3	92.6
1	feed4	87.9
2	feed1	57.0
2	feed2	67.7
2	feed3	92.1
2	feed4	84.2
3	feed1	65.0
3	feed2	74.0
3	feed3	90.2
3	feed4	83.1
4	feed1	58.6

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
pigs2 %>% slice_sample(n=7)
```

pig	feed	weight	pig_id
2	feed2	67.7	feed2_2
5	feed3	99.1	feed3_5
3	feed3	90.2	feed3_3
1	feed4	87.9	feed4_1
2	feed1	57.0	feed1_2
5	feed1	61.7	feed1_5
4	feed1	58.6	feed1_4

...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:

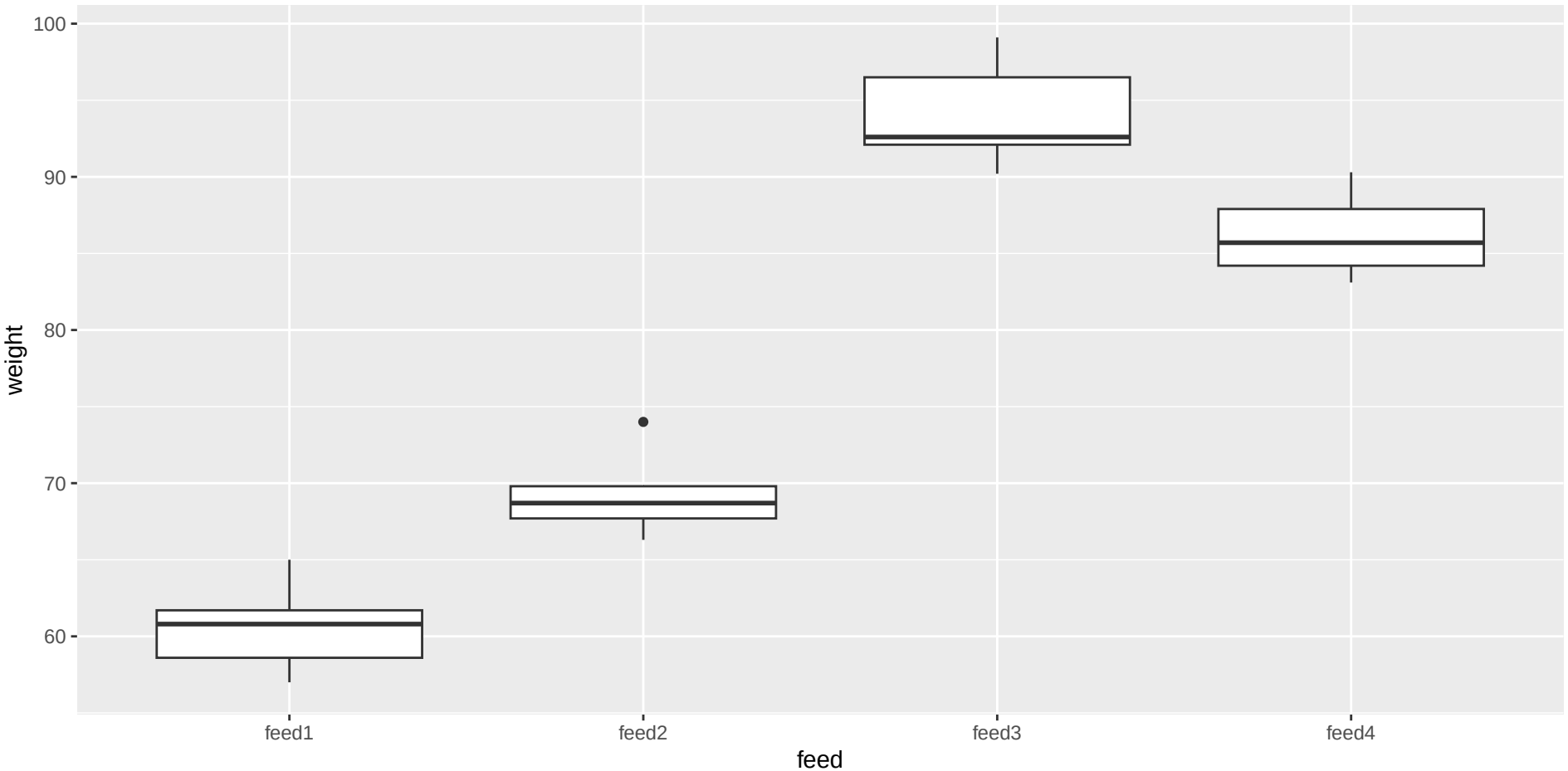
```
pigs2 %>%
  group_by(feed) %>%
  summarize(mean_weight = mean(weight))%>%
  arrange(desc(mean_weight))
```

feed	mean_weight
feed3	94.10
feed4	86.24
feed2	69.30
feed1	60.62

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)
```

```
## Rows: 5769 Columns: 22
## -- Column specification --------------------------
## Delimiter: ","
## chr  (1): iso2
## dbl (21): year, m04, m514, m014, m1524, m2534, m...
##
## i Use `spec()` to retrieve the full column specification fo
## i Specify the column types or set `show_col_types = FALSE`
```

The data (messed up)

```
tb
```

iso2	year	m04	m514	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f04	f514	f014	f1524	f2534	f3544	f4554	f5564	f65	fu
AD	1989	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1990	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1991	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1992	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1993	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1994	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1995	NA	NA	0	0	0	4	1	0	0	NA	NA	NA	0	1	1	0	0	1	0	NA
AD	1996	NA	NA	0	0	1	2	2	1	6	NA	NA	NA	0	1	2	3	0	0	1	NA
AD	1998	NA	NA	0	0	0	1	0	0	0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	1999	NA	NA	0	0	0	1	1	0	0	NA	NA	NA	0	0	0	1	0	0	0	NA
AD	2000	NA	NA	0	0	1	0	0	0	0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	2001	NA	NA	0	NA	NA	2	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AD	2002	NA	NA	0	0	0	1	0	0	0	NA	NA	NA	0	1	0	0	0	0	0	NA

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
```

- columns to make longer
- column to contain the names
- column to contain the values
- (optional) drop missings in the values

Results (some)

```
tb2
```

iso2	year	genage	freq
AD	1996	m014	0
AD	1996	m1524	0
AD	1996	m2534	0
AD	1996	m3544	4
AD	1996	m4554	1
AD	1996	m5564	0
AD	1996	m65	0
AD	1996	f014	0
AD	1996	f1524	1
AD	1996	f2534	1
AD	1996	f3544	0
AD	1996	f4554	0
AD	1996	f5564	1

Separating

- 4 columns, but 5 variables, since `genage` contains both gender and age group. Split that up using separate.
- `separate` needs 3 things:
  - what to separate (no quotes needed),
  - what to separate into (here you do need quotes),
  - how to split.
- For "how to split", here "after first character":

```
tb2 %>% separate(genage, c("gender", "age"), 1) -> tb3
```

Tidied tuberculosis data (some)

```
tb3
```

iso2	year	gender	age	freq
AD	1996	m	014	0
AD	1996	m	1524	0
AD	1996	m	2534	0
AD	1996	m	3544	4
AD	1996	m	4554	1
AD	1996	m	5564	0
AD	1996	m	65	0
AD	1996	f	014	0
AD	1996	f	1524	1
AD	1996	f	2534	1
AD	1996	f	3544	0
AD	1996	f	4554	0
AD	1996	f	5564	1

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 = T) %>%
  separate(genage, c("gender", "age"), 1)
```

iso2	year	gender	age	freq
AD	1996	m	014	0
AD	1996	m	1524	0
AD	1996	m	2534	0
AD	1996	m	3544	4
AD	1996	m	4554	1
AD	1996	m	5564	0
AD	1996	m	65	0

Total tuberculosis cases by year (some of the years)

```
tb3 %>%
  filter(between(year, 1991, 1998)) %>%
  group_by(year) %>% summarize(total=sum(freq))
```

year	total
1991	544
1992	512
1993	492
1994	750
1995	513971
1996	635705
1997	733204
1998	840389

- Something very interesting happened between 1994 and 1995.

To find out what

- try counting up total cases by country:

```
tb3 %>% group_by(iso2) %>%
  summarize(total=sum(freq)) %>%
  arrange(desc(total))
```

iso2	total
CN	4065174
IN	3966169
ID	1129015
ZA	900349
BD	758008
VN	709695
CD	603095
PH	490040
BR	440609
KE	421523

what years do I have for China?

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

```
tb3 %>% filter(iso2=="CN") %>%
  group_by(year) %>%
  summarize(total=sum(freq))
```

year	total
1995	131194
1996	168270
1997	195895
1998	214404
1999	212258
2000	213766
2001	212766
2002	194972
2003	267280
2004	284826

first year of recording for each country?

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

```
tb3 %>% group_by(iso2) %>%
  summarize(first_year=min(year)) %>%
  count(first_year)
```

first_year	n
1980	2
1994	2
1995	130
1996	31
1997	17
1998	6
1999	10
2000	4
2001	1
2002	3
2003	2

Some Toronto weather data

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

```
## Rows: 24 Columns: 35
## -- Column specification --------------------------
## Delimiter: ","
## chr  (3): station, Month, element
## dbl (32): Year, d01, d02, d03, d04, d05, d06, d0...
##
## i Use `spec()` to retrieve the full column specification fo
## i Specify the column types or set `show_col_types = FALSE`
```

The data (some, messed up)

```
weather
```

station	Year	Month	element	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25
TO000008	2008	1	tmax	-	-	-	-	-	-	-	3.0	1.6	5.9	11.5	6.1	9	-	-	-	-	-	-	3.1	6.8	4.2	3.0	8.1	-
CITY				7.9	7.1	5.3	7.7	1.4	1.7	5.1	4.0				11.5	0.4	5.1	2.4	4.6	3.0						3.9	0.8	
TO000008	2008	1	tmin	-	-	-	-	-	-	-	-	-	-	5.6	-	-	-	-	-	-	-	-	0.9	-	0.9	-	-	-
CITY				1.8	6.2	5.1	1.2	9.2	7.0	2.6	2.1	3.7	5.5	7.0	6.1	3.3	1.1	2.4	5.5	4.8	2.1	8.7	1.1	7.0	5.4	4.0	0.1	4.0	1.0	1.4	0.1
TO000008	2008	2	tmax	6	0.4	1.8	-	-	-	-	-	-	-	-	5.7	9.6	4.8	-	4.5	5.8	1	5.1	5.4	1.6	2.6	9.6	2.1	2.8
CITY				8.6		6.6	3.2	4.1	5.2	3.5	3.0	0.6	1.8	0.8			0.4											
TO000008	2008	2	tmin	-	-	-	-	-	-	-	-	-	-	-	-	-	3.1	-	-	-	0.6	5.2	-	-	-	2.0	1.4	0
CITY				8.9	1.5	9.0	7.8	8.8	1.2	8.0	2.8	8.7	9.7	8.7	5.1	5.2	8.4	9.6	1.0		5.0	5.4	1.3		1.2	1.9	0.3	
TO000008	2008	3	tmax	NA	NA	NA	NA	NA	3.1	0.5	5.3	3.2	3.9	-	2.6	2.2	6.4	0.3	6.4	6.4	-	1.5	6.2	5.4	4.9	2.9	2.5	7
CITY														0.1						0.2								
TO000008	2008	3	tmin	NA		NA		NA		0.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

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.
- 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 = T) -> d
```

So far

```
d
```

station	Year	Month	element	day	temperature
TORONTO CITY	2018	01	tmax	d01	-7.9
TORONTO CITY	2018	01	tmax	d02	-7.1
TORONTO CITY	2018	01	tmax	d03	-5.3
TORONTO CITY	2018	01	tmax	d04	-7.7
TORONTO CITY	2018	01	tmax	d05	-14.7
TORONTO CITY	2018	01	tmax	d06	-15.4
TORONTO CITY	2018	01	tmax	d07	-1.0
TORONTO CITY	2018	01	tmax	d08	3.0
TORONTO CITY	2018	01	tmax	d09	1.6
TORONTO CITY	2018	01	tmax	d10	5.9
TORONTO CITY	2018	01	tmax	d11	11.6
TORONTO CITY	2018	01	tmax	d12	11.9
TORONTO CITY	2018	01	tmax	d13	-11.0

The days

- 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`:

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

So far

```
d
```

station	Year	Month	day	tmax	tmin
TORONTO CITY	2018	01	d01	-7.9	-18.6
TORONTO CITY	2018	01	d02	-7.1	-12.5
TORONTO CITY	2018	01	d03	-5.3	-11.2
TORONTO CITY	2018	01	d04	-7.7	-19.7
TORONTO CITY	2018	01	d05	-14.7	-20.6
TORONTO CITY	2018	01	d06	-15.4	-22.3
TORONTO CITY	2018	01	d07	-1.0	-17.5
TORONTO CITY	2018	01	d08	3.0	-1.7
TORONTO CITY	2018	01	d09	1.6	-0.6
TORONTO CITY	2018	01	d10	5.9	-1.3
TORONTO CITY	2018	01	d11	11.6	5.6
TORONTO CITY	2018	01	d12	11.9	-11.2
TORONTO CITY	2018	01	d13	-11.0	-14.5

Further improvements

- 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:

```
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) -> d
```

So far

```
d
```

Year	Month	day	tmax	tmin	Day
2018	01	d01	-7.9	-18.6	1
2018	01	d02	-7.1	-12.5	2
2018	01	d03	-5.3	-11.2	3
2018	01	d04	-7.7	-19.7	4
2018	01	d05	-14.7	-20.6	5
2018	01	d06	-15.4	-22.3	6
2018	01	d07	-1.0	-17.5	7
2018	01	d08	3.0	-1.7	8
2018	01	d09	1.6	-0.6	9
2018	01	d10	5.9	-1.3	10
2018	01	d11	11.6	5.6	11
2018	01	d12	11.9	-11.2	12
2018	01	d13	-11.0	-14.5	13

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
```

date	tmax	tmin
2018-01-01	-7.9	-18.6
2018-01-02	-7.1	-12.5
2018-01-03	-5.3	-11.2
2018-01-04	-7.7	-19.7
2018-01-05	-14.7	-20.6
2018-01-06	-15.4	-22.3
2018-01-07	-1.0	-17.5
2018-01-08	3.0	-1.7
2018-01-09	1.6	-0.6
2018-01-10	5.9	-1.3
2018-01-11	11.6	5.6
2018-01-12	11.9	-11.2
2018-01-13	-11.0	-14.5

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.

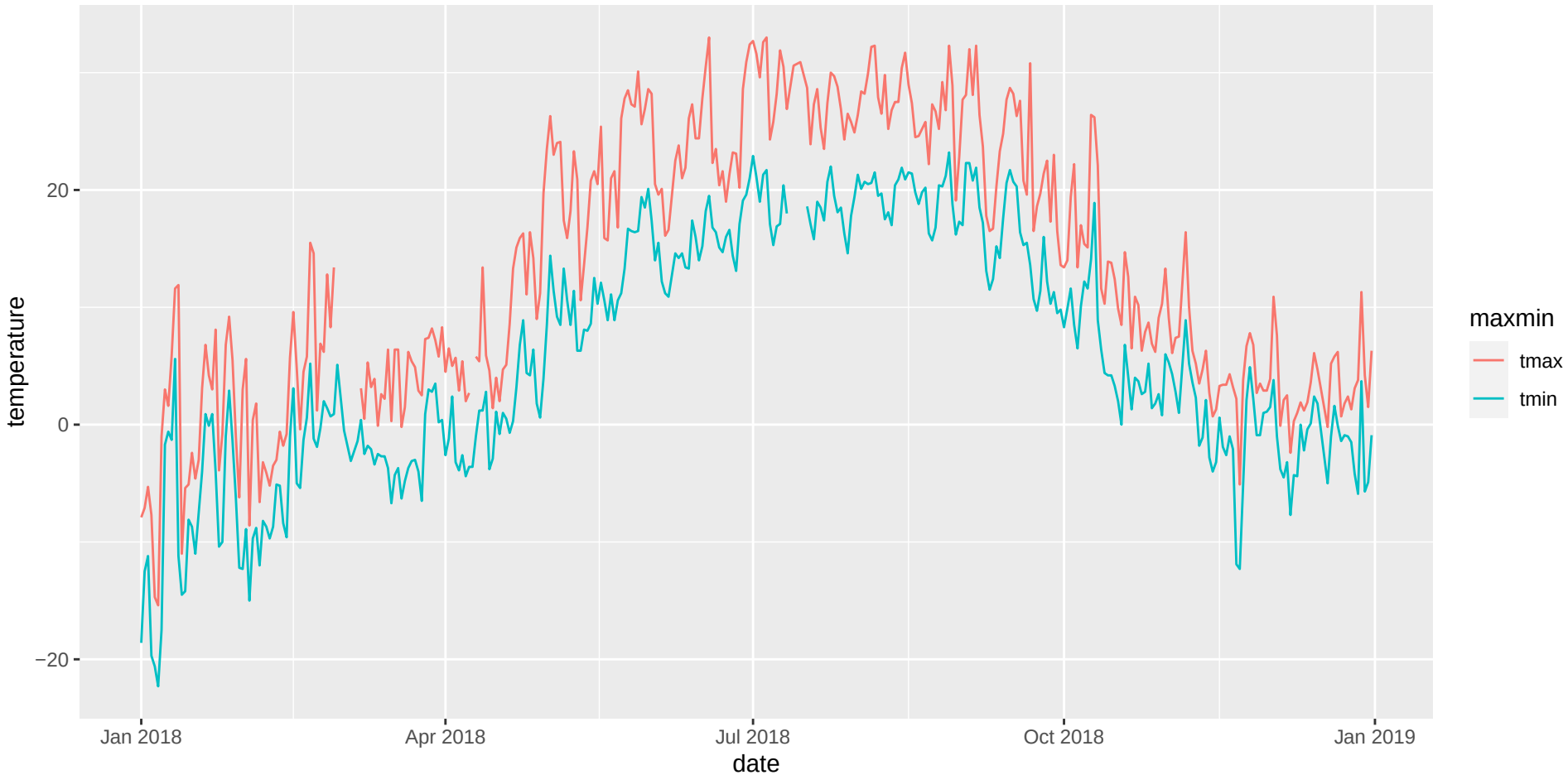
Setting up plot

- 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_line() -> g
```

The plot

```
g
```


20
0
−20
temperature
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
date
maxmin
tmax
tmin

Summary of tidying "verbs"

Verb	Purpose
`pivot_longer`	Combine columns that measure same thing into one
`pivot_wider`	Take column that measures one thing under different conditions and put into multiple columns
`separate`	Turn a column that encodes several variables into several columns
`unite`	Combine several (related) variables into one "combination" variable

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