When pivot-wider goes wrong

Some long data that should be wide

obs	time	y
1	pre	19
2	post	18
3	pre	17
4	post	16
5	pre	15
6	post	14

- Six observations of variable `y`, but three measured before some treatment and three measured after.
- Really matched pairs, so want column of `y`-values for `pre` and for `post`.
- `pivot_wider`.

What happens here?

```
d %>% pivot_wider(names_from = time, values_from = y)
```

obs	pre	post
1	19	NA
2	NA	18
3	17	NA
4	NA	16
5	15	NA
6	NA	14

- Should be *three* `pre` values and *three* `post`.
- Why did this happen?
- `pivot_wider` needs to know which *row* to put each observation in.
- Uses combo of columns *not* named in `pivot_wider`, here `obs` (only).

The problem

```
d %>% pivot_wider(names_from = time, values_from = y)
```

obs	pre	post
1	19	NA
2	NA	18
3	17	NA
4	NA	16
5	15	NA
6	NA	14

- There are 6 different `obs` values, so 6 different rows.
- No data for `obs` 2 and `pre`, so that cell missing (`NA`).
- Not enough data (6 obs) to fill 12 $(= 2 \times 6)$ cells.
- `obs` needs to say which subject provided which *2* observations.

Fixing it up

subject	time	y
1	pre	19
1	post	18
2	pre	17
2	post	16
3	pre	15
3	post	14

- column `subject` shows which subject provided each `pre` and `post`.
- when we do `pivot_wider`, now only *3* rows, one per subject.

Coming out right

```
d2 %>% pivot_wider(names_from = time, values_from = y)
```

subject	pre	post
1	19	18
2	17	16
3	15	14

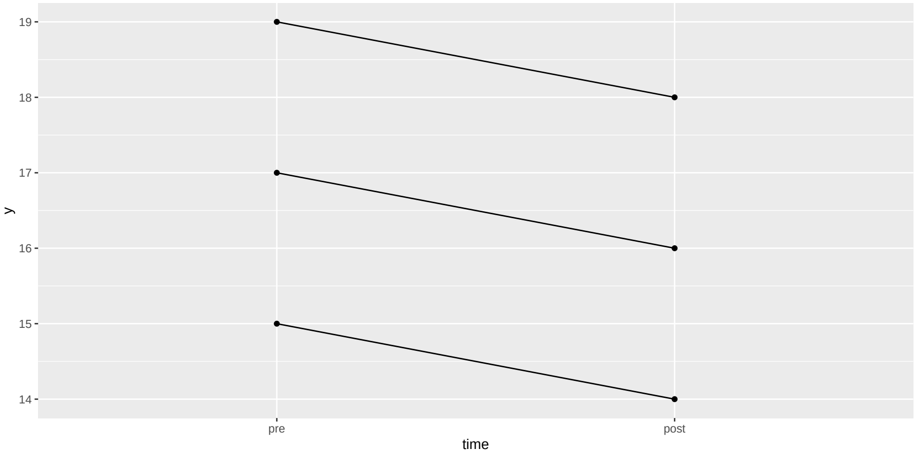
- row each observation goes to determined by other column `subject`, and now a `pre` and `post` for each `subject`.
- right layout for matched pairs t or to make differences for sign test or normal quantile plot.
- “spaghetti plot” needs data longer, as `d2`.

Spaghetti plot

```
d2 %>% mutate(time = fct_inorder(time)) %>%
  ggplot(aes(x = time, y = y, group = subject)) +
    geom_point() + geom_line() -> g
```

The spaghetti plot

```
g
```



- each subject's y decreases over time, with subject 1 highest overall.

Another example

- Two independent samples this time

group	y
control	8
control	11
control	13
control	14
treatment	12
treatment	15
treatment	16
treatment	17

- These should be arranged like this
- but what if we make them wider?

wider

```
d3 %>% pivot_wider(names_from = group, values_from = y)
```

```
## Warning: Values are not uniquely identified; output will co
cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates
## * Use `values_fn = {summary_fun}` to summarise duplicates
```

control	treatment
8, 11, 13, 14	12, 15, 16, 17

- row determined by what not used for `pivot_wider`: nothing!
- everything smooshed into *one* row!
- this time, too *much* data for the layout.
- Four data values squeezed into each of the two cells: “list-columns”.

Get the data out

- To expand list-columns out into the data values they contain, can use `unnest`:

```
d3 %>% pivot_wider(names_from = group, values_from = y) %>%
  unnest(c(control, treatment))
```

```
## Warning: Values are not uniquely identified; output will co
cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates
## * Use `values_fn = {summary_fun}` to summarise duplicates
```

control	treatment
8	12
11	15
13	16
14	17

A proper use of list-columns

```
d3 %>% nest_by(group) %>%
  summarize(n = nrow(data),
            mean_y = mean(data$y),
            sd_y = sd(data$y))
```

```
## `summarise()` has grouped output by 'group'. You can overri
```

group	n	mean_y	sd_y
control	4	11.5	2.645751
treatment	4	15.0	2.160247

- another way to do `group_by` and `summarize` to find stats by group.
- run this one piece at a time to see what it does.