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 are randomly allocated to one of four feeds. At the end of the study, the weight of each pig is recorded, and we want to know whether there are any differences in mean weights among the feeds.
- Problem: want the weights all in one column, with 2nd column labelling which feed each weight was from. 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

```r
library(tidyverse)
library(readxl)
```

Reading in the pig data

```
my_url <- "http://www.utsc.utoronto.ca/~butler/c32/pigs1.txt"
pigs1 <- read_delim(my_url, " ")
pigs1
```

feed1	feed2	feed3	feed4
60.8	68.7	92.6	87.9
57.0	67.7	92.1	84.2
65.0	74.0	90.2	83.1
58.6	66.3	96.5	85.7
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 `gather`: what makes the columns different (different feeds), what makes them the same (all weights), which columns to combine.

Long format pigs

```
pigs2
```

feed	weight
feed1	60.8
feed2	68.7
feed3	92.6
feed4	87.9
feed1	57.0
feed2	67.7
feed3	92.1
feed4	84.2
feed1	65.0
feed2	74.0
feed3	90.2
feed4	83.1
feed1	58.6

…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 later to put the means in descending order:

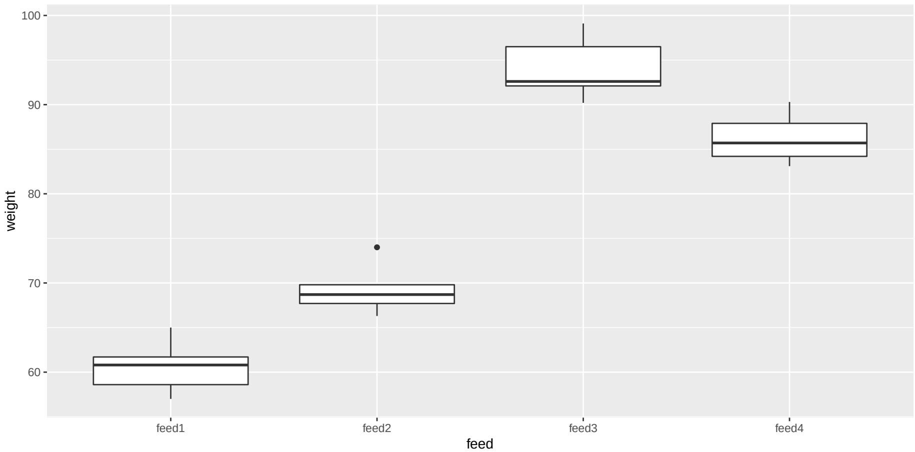
```
pigs2 %>%
  group_by(feed) %>%
  summarize(mean_weight = mean(weight))%>%
  arrange(desc(mean_weight))
```

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://www.utsc.utoronto.ca/~butler/c32/tb.csv"
tb <- read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   iso2 = col_character()
## )

## See spec(...) for full column specifications.
```

The data

```
glimpse(tb)
```

```
## Rows: 5,769
## Columns: 22
## $ iso2  <chr> "AD", "AD", "AD", "AD", "AD", "AD...
## $ year  <dbl> 1989, 1990, 1991, 1992, 1993, 199...
## $ m04   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ m514  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ m014  <dbl> NA, NA, NA, NA, NA, NA, 0, 0, 0, ...
## $ m1524 <dbl> NA, NA, NA, NA, NA, NA, 0, 0, 0, ...
## $ m2534 <dbl> NA, NA, NA, NA, NA, NA, 0, 1, 0, ...
## $ m3544 <dbl> NA, NA, NA, NA, NA, NA, 4, 2, 1, ...
## $ m4554 <dbl> NA, NA, NA, NA, NA, NA, 1, 2, 0, ...
## $ m5564 <dbl> NA, NA, NA, NA, NA, NA, 0, 1, 0, ...
## $ m65   <dbl> NA, NA, NA, NA, NA, NA, 0, 6, 0, ...
## $ mu    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ f04   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, N...
```

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.

All frequencies in one column

```
tb %>%
  pivot_longer(m04:fu, names_to = "genage",
               values_to = "freq", values_drop_na = T) -> tb2
```

- 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)) %>%
  count(year, wt=freq)
```

year	n
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 %>%
  count(iso2, wt=freq) %>%
  arrange(desc(n))
```

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

what years do I have for China?

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

```
tb3 %>% filter(iso2=="CN") %>%
  count(year, wt=freq)
```

year	n
1995	131194
1996	168270
1997	195895
1998	214404
1999	212258
2000	213766
2001	212766
2002	194972
2003	267280
2004	384886
2005	47[illegible]71[illegible]

first year of recording for each country?

- A lot of countries started recording in about 1995:

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

iso2	first_year
CA	1980
CK	1980
FJ	1994
MN	1994
AL	1995
AM	1995
AO	1995
AT	1995
AZ	1995

Some Toronto weather data

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

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   station = col_character(),
##   Month = col_character(),
##   element = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

The data (some)

```
weather
```

station	Year	Month	element	d01	d02	d03	d04
TORONTO CITY	2018	01	tmax	-7.9	-7.1	-5.3	-7.7
TORONTO CITY	2018	01	tmin	-18.6	-12.5	-11.2	-19.7
TORONTO CITY	2018	02	tmax	5.6	-8.6	0.4	1.8
TORONTO CITY	2018	02	tmin	-8.9	-15.0	-9.7	-8.8
TORONTO CITY	2018	03	tmax	NA	NA	NA	NA
TORONTO CITY	2018	03	tmin	NA	-0.5	NA	-3.1
TORONTO CITY	2018	04	tmax	4.5	6.5	5.0	5.7
TORONTO CITY	2018	04	tmin	-2.6	-1.2	2.4	-3.2
TORONTO CITY	2018	05	tmax	23.5	26.3	23.0	24.0
TORONTO CITY	2018	05	tmin	8.5	14.4	11.4	9.2
TORONTO CITY	2018	06	tmax	28.2	20.5	19.6	20.1
TORONTO CITY	2018	06	tmin	17.4	14.0	15.5	12.2
TORONTO CITY	2018	07	tmax	32.7	31.6	29.6	32.6

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.

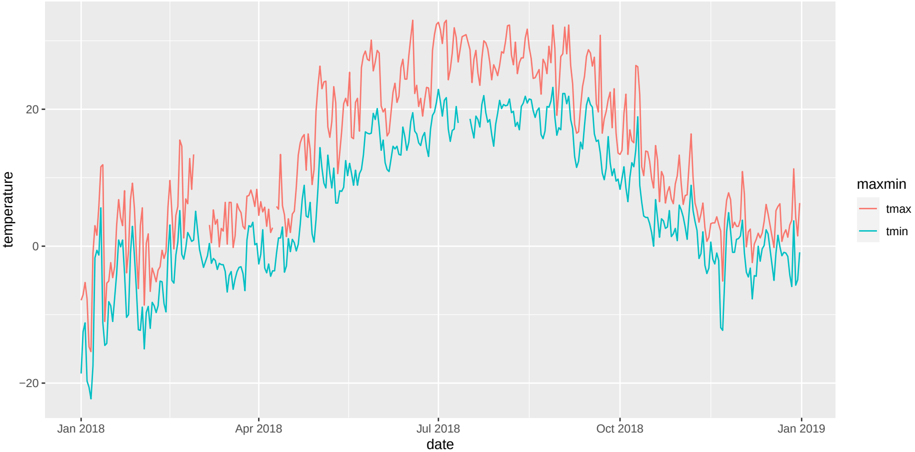
Setting up plot

- 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
```



temperature
20
0
−20
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.

Doing things with data frames

Let's go back to our Australian athletes:

```
## Parsed with column specification:
## cols(
##   Sex = col_character(),
##   Sport = col_character(),
##   RCC = col_double(),
##   WCC = col_double(),
##   Hc = col_double(),
##   Hg = col_double(),
##   Ferr = col_double(),
##   BMI = col_double(),
##   SSF = col_double(),
##   `%Bfat` = col_double(),
##   LBM = col_double(),
##   Ht = col_double(),
##   Wt = col_double()
```

Choosing a column

```
athletes %>% select(Sport)
```

Sport
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball

Choosing several columns

```
athletes %>% select(Sport, Hg, BMI)
```

Sport	Hg	BMI
Netball	13.6	19.16
Netball	12.7	21.15
Netball	12.3	21.40
Netball	12.3	21.03
Netball	12.8	21.77
Netball	11.8	21.38
Netball	12.7	21.47
Netball	12.4	24.45
Netball	12.4	22.63
Netball	14.1	22.80
Netball	12.5	23.58
Netball	12.1	20.06
Netball	12.7	23.01

Choosing consecutive columns

```
athletes %>% select(Sex:WCC)
```

Sex	Sport	RCC	WCC
female	Netball	4.56	13.30
female	Netball	4.15	6.00
female	Netball	4.16	7.60
female	Netball	4.32	6.40
female	Netball	4.06	5.80
female	Netball	4.12	6.10
female	Netball	4.17	5.00
female	Netball	3.80	6.60
female	Netball	3.96	5.50
female	Netball	4.44	9.70
female	Netball	4.27	10.60
female	Netball	3.90	6.30
female	Netball	4.02	9.10

Choosing all-but some columns

```
athletes %>% select(-(RCC:LBM))
```

Sex	Sport	Ht	Wt
female	Netball	176.8	59.90
female	Netball	172.6	63.00
female	Netball	176.0	66.30
female	Netball	169.9	60.70
female	Netball	183.0	72.90
female	Netball	178.2	67.90
female	Netball	177.3	67.50
female	Netball	174.1	74.10
female	Netball	173.6	68.20
female	Netball	173.7	68.80
female	Netball	178.7	75.30
female	Netball	183.3	67.40
female	Netball	174.4	70.00

Select-helpers

Other ways to select columns: those whose name:

- `starts_with` something
- `ends_with` something
- `contains` something
- `matches` a “regular expression”
- `num_range` like x1 to x3
- `everything()` all the columns

Columns whose names begin with S

```
athletes %>% select(starts_with("S"))
```

Sex	Sport	SSF
female	Netball	49.0
female	Netball	110.2
female	Netball	89.0
female	Netball	98.3
female	Netball	122.1
female	Netball	90.4
female	Netball	106.9
female	Netball	156.6
female	Netball	101.1
female	Netball	126.4
female	Netball	114.0
female	Netball	70.0
female	Netball	77.0

Columns whose names end with C

either uppercase or lowercase:

```
athletes %>% select(ends_with("c"))
```

RCC	WCC	Hc
4.56	13.30	42.2
4.15	6.00	38.0
4.16	7.60	37.5
4.32	6.40	37.7
4.06	5.80	38.7
4.12	6.10	36.6
4.17	5.00	37.4
3.80	6.60	36.5
3.96	5.50	36.3
4.44	9.70	41.4
4.27	10.60	37.7

Case-sensitive

```
athletes %>% select(ends_with("C", ignore.case=F))
```

RCC	WCC
4.56	13.30
4.15	6.00
4.16	7.60
4.32	6.40
4.06	5.80
4.12	6.10
4.17	5.00
3.80	6.60
3.96	5.50
4.44	9.70
4.27	10.60
3.90	6.30
4.02	9.10

Column names containing letter R

```
athletes %>% select(contains("r"))
```

Sport	RCC	Ferr
Netball	4.56	20
Netball	4.15	59
Netball	4.16	22
Netball	4.32	30
Netball	4.06	78
Netball	4.12	21
Netball	4.17	109
Netball	3.80	102
Netball	3.96	71
Netball	4.44	64
Netball	4.27	68
Netball	3.90	78
Netball	4.02	107

Exactly two characters, ending with T

In regular expression terms, this is `^.t$`:

- `^` means "start of text"
- `.` means "exactly one character, but could be anything"
- `$` means "end of text".

```
athletes %>% select(matches("^.t$"))
```

Ht	Wt
176.8	59.90
172.6	63.00
176.0	66.30
169.9	60.70
183.0	72.90
178.2	67.90
177.3	67.50
174.1	74.10

Choosing rows by number

```r
athletes %>% slice(16:25)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.25	10.7	39.5	13.2	127	24.47	156.6	26.50
female	Netball	4.46	10.9	39.7	13.7	102	23.99	115.9	23.01
female	Netball	4.40	9.3	40.4	13.6	86	26.24	181.7	30.10
female	Netball	4.83	8.4	41.8	13.4	40	20.04	71.6	13.93
female	Netball	4.23	6.9	38.3	12.6	50	25.72	143.5	26.65
female	Netball	4.24	8.4	37.6	12.5	58	25.64	200.8	35.52
female	Netball	3.95	6.6	38.4	12.8	33	19.87	68.9	15.59
female	Netball	4.03	8.5	37.7	13.0	51	23.35	103.6	19.61
female	BBall	3.96	7.5	37.5	12.3	60	20.56	109.1	19.75
female	BBall	4.41	8.3	38.2	12.7	68	20.67	102.8	21.30

Non-consecutive rows

```r
athletes %>%
  slice(10,13,17,42)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.44	9.7	41.4	14.1	64	22.80	126.4	24.97
female	Netball	4.02	9.1	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.46	10.9	39.7	13.7	102	23.99	115.9	23.01
female	Row	4.37	8.1	41.8	14.3	53	23.47	98.0	21.79

A random sample of rows

```
athletes %>% slice_sample(n=8)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
male	Row	4.40	5.3	42.5	14.5	109	24.06	46.5	9.03
male	T400m	4.86	3.9	44.9	15.4	73	22.83	34.5	6.56
female	Swim	4.38	5.8	42.0	14.0	27	21.28	55.6	13.61
male	T400m	5.03	6.6	44.7	15.9	191	19.85	30.9	6.53
female	Netball	4.15	6.0	38.0	12.7	59	21.15	110.2	25.26
male	Swim	5.33	5.2	47.8	16.1	176	21.38	52.0	8.44
male	WPolo	4.86	8.9	46.9	15.8	65	23.58	57.7	10.25

Rows for which something is true

```
athletes %>% filter(Sport == "Tennis")
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat	LBM
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86	56.58
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64	56.01
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07	46.52
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31	51.75
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07	42.15
female	Tennis	5.16	7.2	44.3	14.5	88	18.29	61.9	12.92	48.76
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45	41.93
male	Tennis	5.66	8.3	50.2	17.7	38	23.76	56.5	10.05	72.00
male	Tennis	5.03	6.4	42.7	14.3	122	22.01	47.6	8.51	68.00
male	Tennis	4.97	8.8	43.0	14.9	233	22.34	60.4	11.50	63.00
male	Tennis	5.38	6.3	46.0	15.7	32	21.07	34.9	6.26	72.00

More complicated selections

```
athletes %>% filter(Sport == "Tennis", RCC < 5)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45
male	Tennis	4.97	8.8	43.0	14.9	233	22.34	60.4	11.50

Another way to do “and”

```
athletes %>% filter(Sport == "Tennis") %>%
  filter(RCC < 5)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45
male	Tennis	4.97	8.8	43.0	14.9	233	22.34	60.4	11.50

Either/Or

```
athletes %>% filter(Sport == "Tennis" | RCC > 5)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Row	5.02	6.4	44.8	15.2	48	19.76	91.0	19.20
female	T400m	5.31	9.5	47.1	15.9	29	21.35	57.9	11.07
female	Field	5.33	9.3	47.0	15.0	62	25.27	102.8	19.51
female	TSprnt	5.16	8.2	45.3	14.7	34	20.30	46.1	10.15
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07
female	Tennis	5.16	7.2	44.3	14.5	88	18.29	61.9	12.92
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45
male	Swim	5.13	7.1	46.8	15.9	34	22.46	44.5	8.47
male	Swim	5.09	4.7	46.6	15.9	55	23.68	33.7	6.16

Sorting into order

```r
athletes %>% arrange(RCC)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	3.80	6.60	36.5	12.4	102	24.45	156.6	26.57
female	Netball	3.90	6.30	35.9	12.1	78	20.06	70.0	15.01
female	T400m	3.90	6.00	38.9	13.5	16	19.37	48.4	10.48
female	Row	3.91	7.30	37.6	12.9	43	22.27	125.9	25.16
female	Netball	3.95	6.60	38.4	12.8	33	19.87	68.9	15.59
female	Row	3.95	3.30	36.9	12.5	40	24.54	74.9	16.38
female	Netball	3.96	5.50	36.3	12.4	71	22.63	101.1	17.93
female	BBall	3.96	7.50	37.5	12.3	60	20.56	109.1	19.75
female	Tennis	4.00	4.20	36.6	12.0	57	25.36	109.0	20.86
female	Netball	4.02	9.10	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.03	8.50	37.7	13.0	51	23.35	103.6	19.61
female	Netball	4.06	5.80	38.7	12.8	78	21.77	122.1	23.11
female	Swim	4.07	5.90	39.5	13.3	25	20.42	54.6	11.47

Breaking ties by another variable

```
athletes %>% arrange(RCC, BMI)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	3.80	6.60	36.5	12.4	102	24.45	156.6	26.57
female	T400m	3.90	6.00	38.9	13.5	16	19.37	48.4	10.48
female	Netball	3.90	6.30	35.9	12.1	78	20.06	70.0	15.01
female	Row	3.91	7.30	37.6	12.9	43	22.27	125.9	25.16
female	Netball	3.95	6.60	38.4	12.8	33	19.87	68.9	15.59
female	Row	3.95	3.30	36.9	12.5	40	24.54	74.9	16.38
female	BBall	3.96	7.50	37.5	12.3	60	20.56	109.1	19.75
female	Netball	3.96	5.50	36.3	12.4	71	22.63	101.1	17.93
female	Tennis	4.00	4.20	36.6	12.0	57	25.36	109.0	20.86
female	Netball	4.02	9.10	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.03	8.50	37.7	13.0	51	23.35	103.6	19.61
female	Netball	4.06	5.80	38.7	12.8	78	21.77	122.1	23.11
female	Swim	4.07	5.90	39.5	13.3	25	20.42	54.6	11.47

Descending order

```
athletes %>% arrange(desc(BMI))
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
male	Field	5.48	6.20	48.2	16.3	94	34.42	82.7	13.91
male	Field	4.96	8.30	45.3	15.7	141	33.73	113.5	17.41
male	Field	5.48	4.60	49.4	18.0	132	32.52	55.7	8.51
female	Field	4.75	7.50	43.8	15.2	90	31.93	131.9	23.01
male	Field	5.01	8.90	46.0	15.9	212	30.18	112.5	19.94
male	Field	5.01	8.90	46.0	15.9	212	30.18	96.9	18.08
male	Field	5.09	8.90	46.3	15.4	44	29.97	71.1	13.97
female	Field	4.58	5.80	42.1	14.7	164	28.57	109.6	21.30
female	Field	4.51	9.00	39.7	14.3	36	28.13	136.3	24.88
male	WPolo	5.34	6.20	49.8	17.2	143	27.79	75.7	13.49
male	WPolo	4.90	7.60	45.6	16.0	90	27.56	67.2	11.79
male	Field	5.11	9.60	48.2	16.7	103	27.39	65.9	11.66
female	Field	4.81	6.80	42.7	15.3	50	26.95	98.5	20.10

"The top ones"

```
athletes %>%
  arrange(desc(Wt)) %>%
  slice(1:7) %>%
  select(Sport, Wt)
```

Sport	Wt
Field	123.2
BBall	113.7
Field	111.3
Field	108.2
Field	102.7
WPolo	101.0
BBall	100.2

Another way

```
athletes %>%
  slice_max(order_by = Wt, n=7) %>%
  select(Sport, Wt)
```

Sport	Wt
Field	123.2
BBall	113.7
Field	111.3
Field	108.2
Field	102.7
WPolo	101.0
BBall	100.2

Create new variables from old ones

```
athletes %>%
  mutate(wt_lb = Wt * 2.2) %>%
  select(Sport, Sex, Wt, wt_lb) %>%
  arrange(Wt)
```

Sport	Sex	Wt	wt_lb
Gym	female	37.80	83.16
Gym	female	43.80	96.36
Gym	female	45.10	99.22
Tennis	female	45.80	100.76
Tennis	female	47.40	104.28
Gym	female	47.80	105.16
T400m	female	49.20	108.24
Row	female	49.80	109.56
T400m	female	50.90	111.98
Netball	female	51.90	114.18

Turning the result into a number

Output is always data frame unless you explicitly turn it into something else, eg. the weight of the heaviest athlete, as a number:

```
athletes %>% arrange(desc(Wt)) %>% pluck("Wt", 1)
```

```
## [1] 123.2
```

Or the 20 heaviest weights in descending order:

```
athletes %>%
  arrange(desc(Wt)) %>%
  slice(1:20) %>%
  pluck("Wt")
```

```
##  [1] 123.20 113.70 111.30 108.20 102.70 101.00
##  [7] 100.20  98.00  97.90  97.90  97.00  96.90
## [13]  96.30  94.80  94.80  94.70  94.70  94.60
## [19]  94.25  94.20
```

Another way to do the last one

```
athletes %>%
  arrange(desc(Wt)) %>%
  slice(1:20) %>%
  pull("Wt")
```

```
##  [1] 123.20 113.70 111.30 108.20 102.70 101.00
##  [7] 100.20  98.00  97.90  97.90  97.00  96.90
## [13]  96.30  94.80  94.80  94.70  94.70  94.60
## [19]  94.25  94.20
```

`pull` grabs the column you name *as a vector* (of whatever it contains).

To find the mean height of the women athletes

Two ways:

```
athletes %>% group_by(Sex) %>% summarize(m = mean(Ht))
```

Sex	m
female	174.5940
male	185.5059

```
athletes %>%
  filter(Sex == "female") %>%
  summarize(m = mean(Ht))
```

m
174.594

Summary of data selection/arrangement “verbs”

Verb	Purpose
`select`	Choose columns
`print`	Display non-default # of rows/columns
`slice`	Choose rows by number
`sample_n`	Choose random rows
`filter`	Choose rows satisfying conditions
`arrange`	Sort in order by column(s)
`mutate`	Create new variables
`group_by`	Create groups to summarize by
`summarize`	Calculate summary statistics (by groups if defined)
`pluck`	Extract items from data frame
`pull`	Extract a single column from a data frame as a vector

Looking things up in another data frame

Recall the tuberculosis data set, tidied:

```
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

Actual country names

Found actual country names to go with those abbreviations, in spreadsheet:

```
my_url <-
  "http://www.utsc.utoronto.ca/~butler/c32/ISOCountryCodes081507.xlsx"
```

Note trick for reading in `.xlsx` from URL:

```
f <- tempfile()
download.file(my_url, f)
country_names <- read_excel(f)
```

- set up temporary file
- download spreadsheet to there
- read it from temporary file (which is “local”)

The country names

```
country_names
```

Code	Code_UC	Country
ad	AD	Andorra
ae	AE	United Arab Emirates
af	AF	Afghanistan
ag	AG	Antigua and Barbuda
ai	AI	Anguilla
al	AL	Albania
am	AM	Armenia
an	AN	Netherlands Antilles
ao	AO	Angola
aq	AQ	Antarctica
ar	AR	Argentina
arpa	ARPA	Old style Arpanet
as	AS	American Samoa

Looking up country codes

Matching a variable in one data frame to one in another is called a **join** (database terminology):

```
tb3 %>% left_join(country_names, by = c("iso2" = "Code_UC"))
```

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

Total cases by country

```
options(dplyr.summarise.inform=FALSE)
```

```
tb3 %>%
  group_by(iso2) %>%
  summarize(cases = sum(freq)) %>%
  left_join(country_names, by = c("iso2" = "Code_UC")) %>%
  select(Country, cases)
```

Country	cases
Andorra	64
United Arab Emirates	487
Afghanistan	80005
Antigua and Barbuda	21
Anguilla	1
Albania	2467
Armenia	6757

or even sorted in order

```
tb3 %>%
  group_by(iso2) %>%
  summarize(cases = sum(freq)) %>%
  left_join(country_names, by = c("iso2" = "Code_UC")) %>%
  select(Country, cases) %>%
  arrange(desc(cases))
```

Country	cases
China	4065174
India	3966169
Indonesia	1129015
South Africa	900349
Bangladesh	758008
Vietnam	709695
NA	603095
Philippines	490040

Comments

- This is probably not quite right because of:
  - the 1994-1995 thing
  - there is at least one country in `tb3` that was not in `country_names` (the NA above). Which?

```
tb3 %>%
  anti_join(country_names, by = c("iso2" = "Code_UC")) %>%
  distinct(iso2)
```

iso2
CD
ME
NA
PS
RS
TL