

Tidyverse Workshop

Data wrangling with **dplyr** & **tidyr**



Presented by Emi Tanaka

School of Mathematics and Statistics



dr.emi.tanaka@gmail.com



@statsgen

1st Dec 2019 @ Biometrics by the Botanic Gardens | Adelaide, Australia



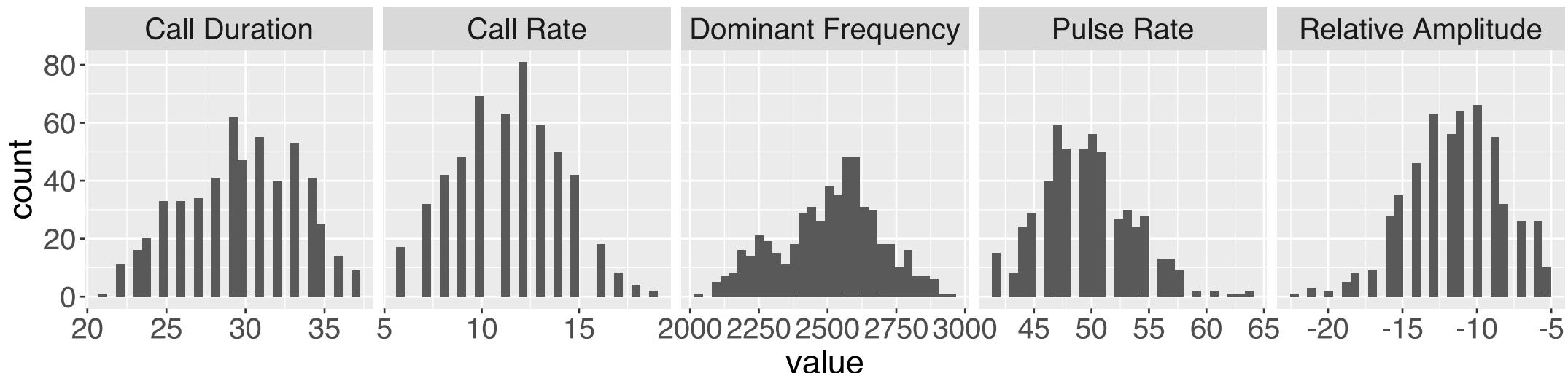
Aim: using this data, plot below graph

```
glimpse(dat)
```

Observations: 535

Variables: 5

```
$ CD <dbl> 28, 27, 31, 33, 26, 31, 33, 28, 29, 34, 27, 29, 25, 33, 24, 30, 23, 34, 28, 2  
$ CR <dbl> 12, 15, 13, 15, 11, 6, 15, 12, 14, 12, 9, 8, 10, 7, 11, 10, 13, 14, 7, 16, 14  
$ DF <dbl> 2315, 2304, 2646, 2281, 2789, 2719, 2233, 2667, 2527, 2574, 2360, 2380, 2535,  
$ RA <dbl> -10, -8, -12, -7, -8, -6, -12, -16, -12, -10, -12, -12, -12, -6, -15, -12, -10, -1  
$ PR <dbl> 46, 49, 43, 50, 57, 51, 46, 55, 56, 45, 47, 50, 57, 57, 52, 46, 51, 59, 44, 4
```





Mapping variable to aesthetic

```
glimpse(dat)
```

Observations: 535

Variables: 5

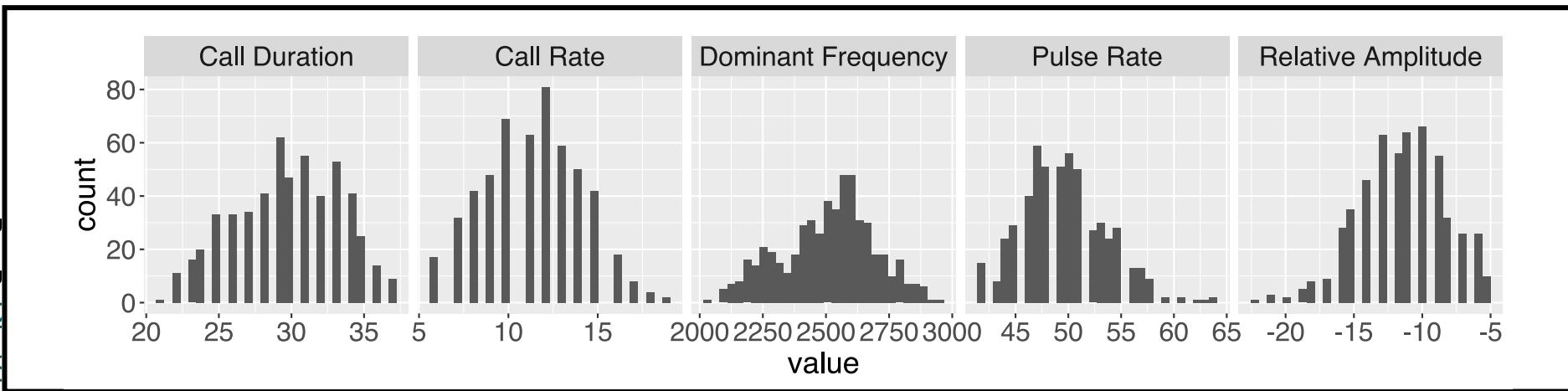
\$ CD <dbl> 28, 27,

\$ CR <dbl> 12, 15,

\$ DF <dbl> 2315, 2

\$ RA <dbl> -10, -8

\$ PR <dbl> 46, 49, 43, 50, 57, 51, 46, 55, 56, 45, 47, 50, 57, 57, 52, 46, 51, 59, 44, 44



```
ggplot(<DATA>, aes(x = <VAR>)) +  
  geom_histogram() +  
  facet_wrap(~ <VAR>,  
            scales = "free_x",  
            nrow = 1)
```

 Data wrangling

Raw data is hardly ever in a format ready for downstream analysis

The data we *have*

CD	CR	DF	RA	PR
1	1	1	1	1
2	2	2	2	2

The data we *need*

```
ggplot(<DATA>, aes(x = value)) +  
  geom_histogram() +  
  facet_wrap(~ name,  
            scales = "free_x",  
            nrow = 1)
```



Data wrangling: `tidyverse::pivot_longer`

The following commands all produce the same output on the right:

```
<div style="border: 1px solid black; padding: 10px; border-bottom: none; background-color: #f0f0ff; margin-bottom: 10px;">pivot_longer(dat, cols = c("CD", "CR", "DF", "RA", "PR))  
pivot_longer(dat, cols = c(CD, CR, DF, RA, PR))  
pivot_longer(dat, cols = CD:PR)  
pivot_longer(dat, cols = everything())
```

Yup, that's all to take your data from `wider` form to `longer` form!

# A tibble: 2,665 × 2		
	name	value
	<chr>	<dbl>
1	CD	28
2	CR	12
3	DF	2315
4	RA	-10
5	PR	46
6	CD	27
7	CR	15
8	DF	2304
9	RA	-8
10	PR	49
# ... with 2,665 more rows		

Tidy evalution

- tidyverse packages all employ **tidy evalution**, which includes **non-standard evaluation**, through **rlang** package
- It's the reason below are the same

```
<div style="border: 1px solid black; padding: 10px; border-radius: 10px; background-color: #f9f9f9;">pivot_longer(dat,  
  cols = c("CD", "CR", "DF", "RA", "PR"))  
  
pivot_longer(dat,  
  cols = c(CD, CR, DF, RA, PR))
```

- You've actually been using tidy evalution in ggplot!

What would happen below?

```
vars <- "CD", "CR", "DF", "RA", "PR")  
pivot_longer(dat, cols = vars)
```

What if the object name is in the data?

```
CD <- c("CD", "CR", "DF", "RA", "PR")  
pivot_longer(dat, cols = CD)
```

To "unquote", you need to use !!
(pronounced bang-bang)

```
CD <- c("CD", "CR", "DF", "RA", "PR")  
pivot_longer(dat, cols = !!!CD)
```

tidyselect

Packages in tidyverse generally use `tidyselect`

```
pivot_longer(dat, cols = CD:PR)
```

Selects all variables starting from CD to PR.

CD	CR	DF	RA	PR
1	1	1	1	1
2	2	2	2	2

```
pivot_longer(dat, cols = everything())
```

signal_dat

Frog ID	CD	CR	DF	RA	PR	Standard1	Standard2	Standard3	Alternative1	Alternative2
13196	28	12	2315	-10	46	47	46	42	28	
13197	27	15	2304	-8	49	69	44	36	33	
13198	31	13	2646	-12	43	139	102	85	227	1
13206	33	15	2281	-7	50	112	112	117	101	
13207	26	11	2789	-8	57	101	101	80	126	
13208	31	6	2719	-6	51	90	68	73	143	1

Previous

1

2

3

4

5

...

90

Next

Note: the header is slightly modified from original.

Non-syntactic variable names

- Syntactic names consist of letters, digits, . and _ only and begin with letters or . AND also cannot be in reserved words list (?Reserved)
- You need to surround non-syntactic names with backticks if you wish to refer to them
- E.g. in signal_dat, "Frog ID" is a variable with non-syntactic name because it has a space in it
- To select "Frog ID", we use

```
signal_dat$`Frog ID`
```

```
[1] 13196 13197 13198 13206 13207 13208 13211 13275 1
```

Rename variable names with dplyr::rename

- Working with non-syntactic names is often a pain!
- You can rename a variable using `dplyr::rename`

```
signal_cdat <- rename(signal_dat,  
  frog_id           = `Frog ID`,  
  two_choice_latency = `Two Choice Latency`,  
  two_choice        = `Two Choice`,  
  phonotaxis_score  = `Phonotaxis Score`,  
  speaker_position   = `Speaker Position`,  
  first_presented    = `First Presented`)
```

Clean variable names

- But it's still a pain to rename one-by-one
- The `janitor` package is fantastic way to clean at once

```
signal_cdat <- janitor::clean_names(signal_dat)
```

```
glimpse(signal_cdat)
```

Observations: 535

Variables: 18

\$ frog_id

<dbl> 13196, 13197, 13198, 13206, 13207, 13208, 13211, 13275, 13276,

\$ cd

<dbl> 28, 27, 31, 33, 26, 31, 33, 28, 29, 34, 27, 29, 25, 33, 24, 30,

\$ cr

<dbl> 12, 15, 13, 15, 11, 6, 15, 12, 14, 12, 9, 8, 10, 7, 11, 10, 13,

\$ df

<dbl> 2315, 2304, 2646, 2281, 2789, 2719, 2233, 2667, 2527, 2574, 236,

\$ ra

<dbl> -10, -8, -12, -7, -8, -6, -12, -16, -12, -10, -12, -12, -6, -15,

\$ pr

<dbl> 46 49 43 50 57 51 46 55 56 45 47 50 57 57 52 46

Note `janitor` is *not* part of `tidyverse` but it's one of my favorites!

Quick skim of the data

```
skimr::skim(signal_cdat)
```

Skim summary statistics

n obs: 535

n variables: 18

— Variable type:character —

	variable	missing	complete	n	min	max	empty	n_unique
first_presented	0	535	535	8	11	0	0	2
speaker_position	161	374	535	4	5	0	0	2
two_choice	1	534	535	8	11	0	0	2

— Variable type:numeric —

	variable	missing	complete	n	mean	sd	p0	p25	p50	100%
alternative1	0	535	535	83.77	43.91	20	52	76	100	100
alternative2	1	534	535	78.01	40.21	17	51	67	93	100

Note `skimr` is *not* part of `tidyverse` but it's also one of my favorites!

Make new variables with dplyr::mutate

Aim: create a new variable of phonotaxis score



```
signal_cdat2 <- mutate(signal_cdat,
  # rounding needed to get the same result as paper
  xbara = round((alternative1 + alternative2 + alternative3) / 3),
  xbars = round((standard1 + standard2 + standard3) / 3),
  # defintion of phonotaxis score
  score = (xbars - xbara) / (xbars + xbara))
```

Think `mutate` as in for biology when a string of DNA is modified by mutation

Select variables with dplyr::select



```
select(signal_cdat2,  
      # below is using tidyselect  
      c(starts_with("standard"), starts_with("alt"),  
        starts_with("xbar"), score, starts_with("pho")))
```

A tibble: 535 x 10

	standard1	standard2	standard3	alternative1	alternative2	alternative3	xbara	xbarb	xbarc
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	47	46	42	28	48	43	40	38	35
2	69	44	36	33	27	37	32	29	26
3	139	102	85	227	111	126	155	141	133
4	112	112	117	101	81	61	81	78	75

Subsetting data with dplyr::filter



```
signal_cdatm <- filter(signal_cdat2, is.na(speaker_position))  
glimpse(select(signal_cdatm, speaker_position))
```

Observations: 161

Variables: 1

\$ speaker_position <chr> NA, NA

- Note: `dplyr::filter` conflicts with `stats::filter`
- Usually loading `tidyverse` displays this conflict

```
tidyverse_conflicts()
```

```
— Conflicts — tidyverse_conflicts() —  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag()    masks stats::lag()
```

Summarising data with dplyr::summarise usually coupled with dplyr::group_by



```
signal_cdatg <- group_by(signal_cdatm, two_choice)
signal_cdats <- summarise(signalf_cdatg,
                           avg_score = mean(score, na.rm = TRUE),
                           sd_score = sd(score, na.rm = TRUE),
                           nmissing = sum(is.na(score)),
                           n = n())
signal_cdats
```

```
# A tibble: 2 x 5
  two_choice   avg_score   sd_score nmissing      n
  <chr>        <dbl>       <dbl>     <int> <int>
1 Alternative  0.0407     0.122      0      60
2 Standard    -0.0386     0.152      1     101
```

The full code to get to the table summary

```
signal_cdat <- janitor::clean_names(signal_dat)
signal_cdat2 <- mutate(signal_cdat,
  xbara = round(alternative1 + alternative2 + alternative3) / 3,
  xbars = round(standard1 + standard2 + standard3) / 3,
  score = (xbars - xbara) / (xbars + xbara))
signal_cdatm <- filter(signal_cdat2, is.na(speaker_position))
signal_cdatg <- group_by(signal_cdatm, two_choice)
signal_cdats <- summarise(signal_cdatg,
  avg_score = mean(score, na.rm = TRUE),
  sd_score = sd(score, na.rm = TRUE),
  nmissing = sum(is.na(score)),
  n = n())
```

😩 The *pain point* - you have to think of a new variable name each time OR risk using the same and accidentally overwrite when unknowingly missed a sequence

The pipe operator from magrittr

`%>%`

`<data> %>% <function>(<argA>, <argB>)`

is the same as

`<function>(<data>, <argA>, <argB>)`

E.g. `filter(signal_cdat2, is.na(speaker_position))`

is the same as

`signal_cdat2 %>% filter(is.na(speaker_position))`

Chain of pipes

```
signal_dat %>%  
  janitor::clean_names() %>%  
  mutate(xbara = round(alternative1 + alternative2 + alternative3) / 3,  
         xbars = round(standard1 + standard2 + standard3) / 3,  
         score = (xbars - xbara) / (xbars + xbara)) %>%  
  filter(is.na(speaker_position)) %>%  
  group_by(two_choice) %>%  
  summarise(avg_score = mean(score, na.rm = TRUE),  
            sd_score = sd(score, na.rm = TRUE),  
            nmissing = sum(is.na(score)),  
            n = n())  
  
# A tibble: 2 x 5  
  two_choice avg_score sd_score nmissing     n  
  <chr>          <dbl>      <dbl>     <int> <int>  
1 Alternative    0.0407     0.122        0     60
```

Main dplyr verbs

- `mutate()` - create new or overwrite existing variables based on existing variables
- `select()` - select and rename variables (reduces column)
- `filter()` - subset data (reduces rows)
- `summarise()` - reduce data to a summary statistics
- `arrange()` - ???
- `group_by()` coupled with `ungroup()` - group operations

What does `arrange()` do?

Special extensions

*_if(), *_at() and *_all()

mutate_if()	summarise_if()
mutate_at()	summarise_at()
mutate_all()	summarise_all()
select_if()	group_by_if()
select_at()	group_by_at()
select_all()	group_by_all()
filter_if()	arrange_if()
filter_at()	arrange_at()
filter_all()	arrange_all()

*_if()



```
signal_cdat %>%  
  mutate_if(is.character, as.factor) %>%  
  select_if(is.factor)
```

A tibble: 535 x 3

	two_choice	speaker_position	first_presented
	<fct>	<fct>	<fct>
1	Standard	left	Alternative
2	Alternative	left	Alternative
3	Alternative	left	Alternative
4	Standard	left	Standard
5	Alternative	left	Standard
6	Standard	left	Standard
7	Alternative	left	Alternative
8	Alternative	left	Standard

*_at()



```
scale2 <- function(x, na.rm = FALSE)
  (x - mean(x, na.rm = na.rm)) / sd(x, na.rm = na.rm)
signal_cdat2 %>%
  mutate_at("score", scale2)
```

To use tidyselect, the variables need to be wrapped with `vars()` function:



```
signal_cdat2 %>%
  mutate_at(vars(cd:pr), scale2)
```

*_all()

Applies function to all variables

```
</> signal_cdat2 %>%  
  mutate_all(I)
```

Open and go through:
`challenge-04-wrangling.Rmd`

For answers go to (but don't look until trying!):
`challenge-04-wrangling-solution.Rmd`

Session Information

```
devtools::session_info()
```

– Session info

setting value

version R version 3.6.0 (2019-04-26)

os macOS Mojave 10.14.6

system x86_64, darwin15.6.0

ui X11

language (EN)

collate en_AU.UTF-8

These slides are licensed under