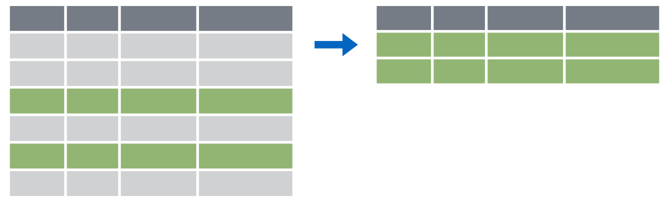
ANT 6973: DATA VISUALIZATION AND EXPLORATION

DATA MANIPULATION, PART 2

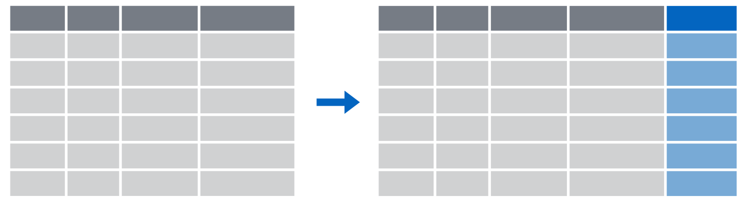
SINGLE TABLE VERBS

Extract variables with `select()`

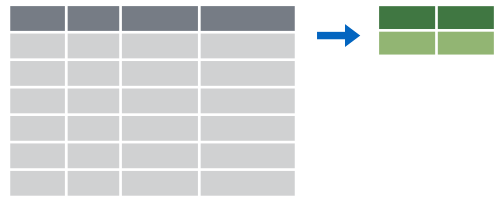


Extract cases with `filter()`

Arrange cases with `arrange()`



Make new variables with `mutate()`



Make tables of summaries with `summarise()`
along with `group_by()`

`summarise()`

A NOTE ON SPELLING…

Hadley Wickham
@hadleywickham

The holy grail: "For consistency, aim to use British (rather than American) spelling." #rstats developer.r-project.org/Rds.html. Colour is right!

10 2:38 PM - Nov 27, 2013

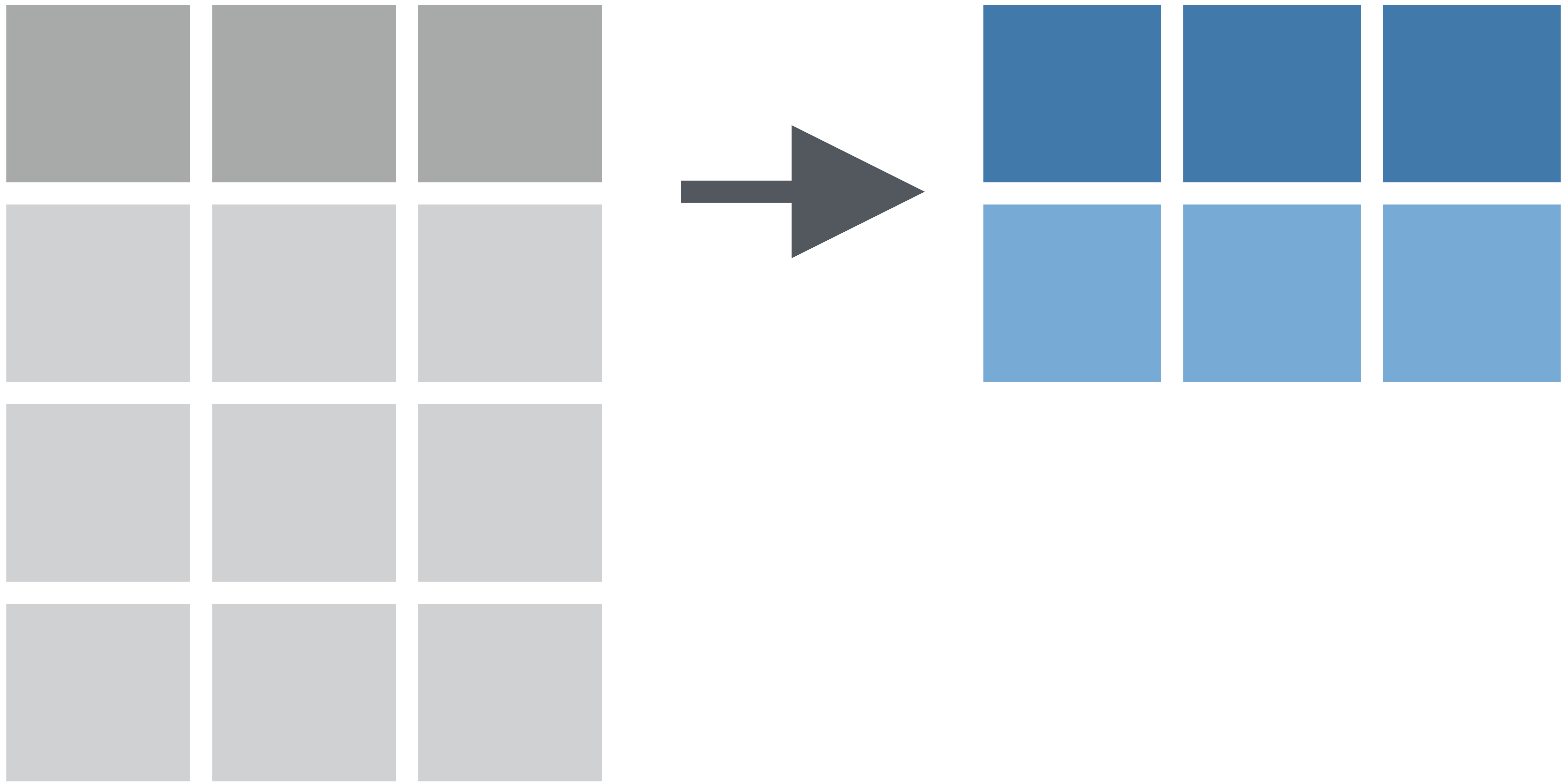
19 people are talking about this

But both the British and American spellings work:

```
summarize() = summarise()
color() = colour()
```

SUMMARISE()

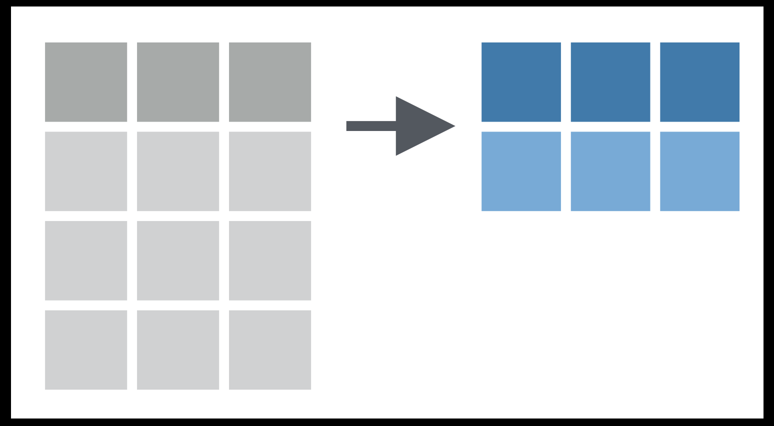
Compute table of summaries.



A *summary function* returns a *single value* that summarizes many values in a column

SUMMARISE()

Compute table of summaries.



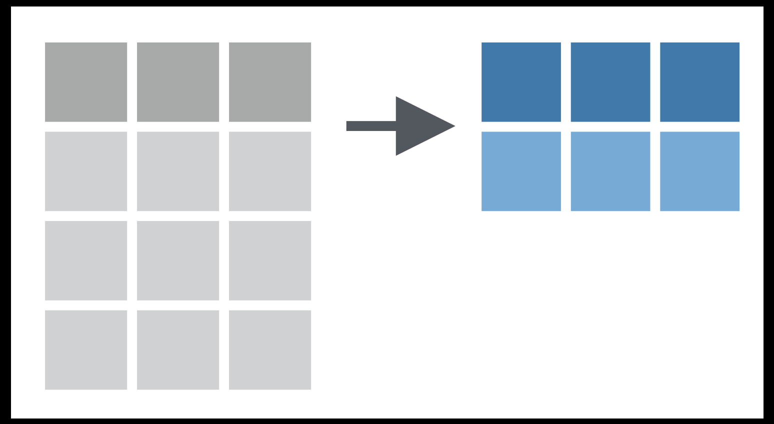
```
summarise(.data, …)
```

data frame to summarize

Name-value pairs of summary functions

SUMMARISE()

Compute table of summaries.



```
summarise(babynames, max_prop = max(prop))
```

data frame to summarize

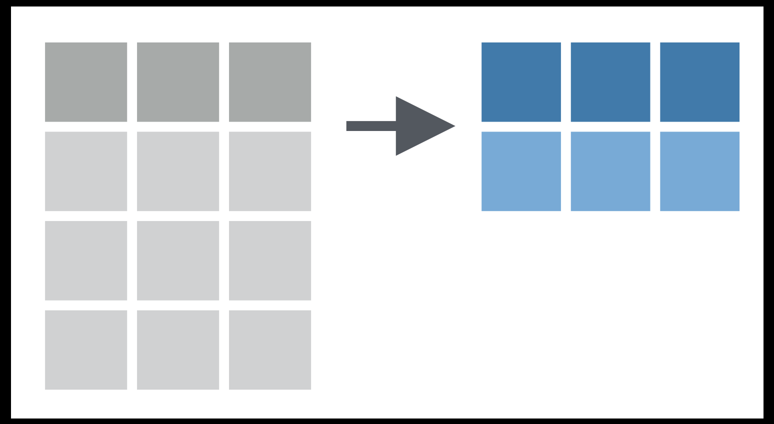
Name of new variable

Summary function

Column of data to summarize

SUMMARISE()

Compute table of summaries.



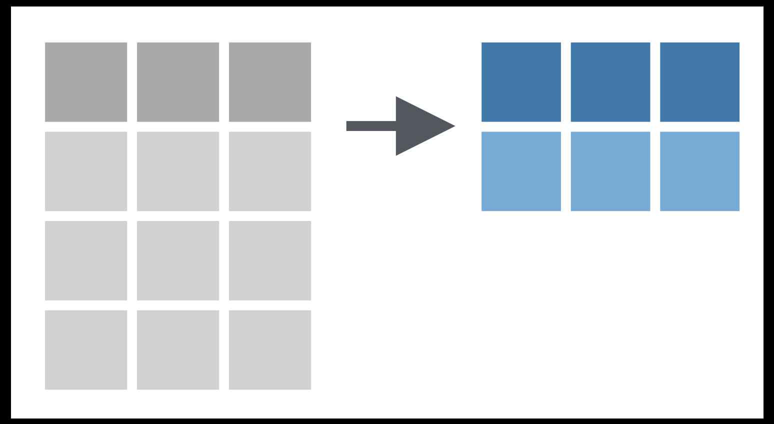
```
summarise(babynames, max_prop = max(prop))
```

year	sex	name	n	prop
1880	F	Mary	7065	0.07238
1880	F	Anna	2604	0.02667
1880	F	Emma	2003	0.02052
1880	F	Elizabet	1939	0.01986
1880	F	Minnie	1746	0.01788
1880	F	Margare	1578	0.01616

→

max_prop
0.0815

SUMMARISE()



Compute table of summaries.

```
babynames %>%
  summarise(max_n = max(n),
            min_n = min(n))
```

Multiple summaries separated by commas

year	sex	name	n	prop
1880	F	Mary	7065	0.07238
1880	F	Anna	2604	0.02667
1880	F	Emma	2003	0.02052
1880	F	Elizabet	1939	0.01986
1880	F	Minnie	1746	0.01788
1880	F	Margare	1578	0.01616

max_n	min_n
99686	5

USEFUL SUMMARY FUNCTIONS

- **Center:** `mean()`, `median()`
- **Spread:** `sd()`, `var()`, `IQR()`, `mad()`
- **Range:** `min()`, `max()`, `quantile()`
- **Logical:** `any()`, `all()`
- **Position:** `first()`, `last()`, `nth()`
- **Count:** `n()`, `n_distinct()`

USEFUL SUMMARY FUNCTIONS

- **Center:** `mean()`, `median()`
- **Spread:** `sd()`, `var()`, `IQR()`, `mad()`
- **Range:** `min()`, `max()`, `quantile()`
- **Logical:** `any()`, `all()`
- **Position:** `first()`, `last()`, `nth()`
- **Count:** `n()`, `n_distinct()`

Base R functions

USEFUL SUMMARY FUNCTIONS

- **Center:** `mean()`, `median()`
- **Spread:** `sd()`, `var()`, `IQR()`, `mad()`
- **Range:** `min()`, `max()`, `quantile()`
- **Logical:** `any()`, `all()`
- **Position:** `first()`, `last()`, `nth()`
- **Count:** `n()`, `n_distinct()`

dplyr functions

ACTIVITY 1

- Use `summarise()` to compute three statistics about babynames:
  - The smallest (minimum) year in the dataset
  - The largest (maximum) year in the dataset
  - The total number of children represented in the data

```
babynames %>%
  summarise(first_yr = min(year),
            last_yr = max(year),
            total_n = sum(n))
first_yr last_yr   total_n
     <dbl>   <dbl>      <int>
1     1880    2017 348120517
```

ACTIVITY 2

- Extract the rows where name is "Khaleesi". Then use `summarise()` to find:
    - The total number of children named Khaleesi
    - The first year Khaleesi appeared in the data

```
babynames %>%
  filter(name == "Khaleesi") %>%
  summarise(total = sum(n),
            first_year = min(year))
  total first_year
  <int> <dbl>
1  1964  2011
```

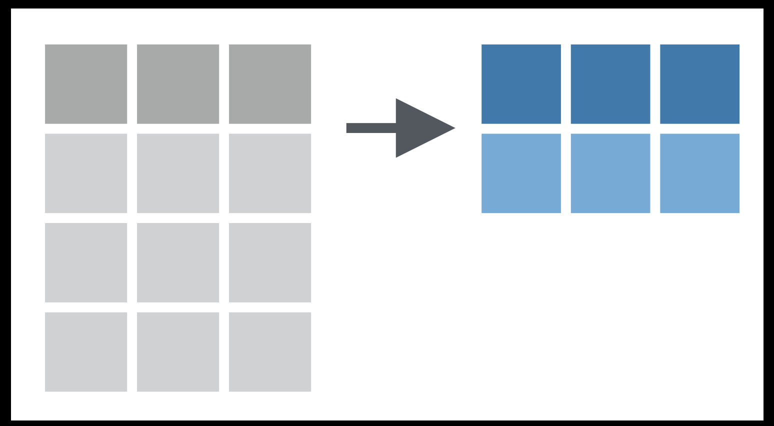
USEFUL SUMMARY FUNCTIONS

- **Center:** `mean()`, `median()`
- **Spread:** `sd()`, `var()`, `IQR()`, `mad()`
- **Range:** `min()`, `max()`, `quantile()`
- **Logical:** `any()`, `all()`
- **Position:** `first()`, `last()`, `nth()`
- **Count:** `n()`, `n_distinct()`

dplyr functions

n()

The number of rows in a dataset/group



```
babynames %>%
  summarise(n_rows = n())
```

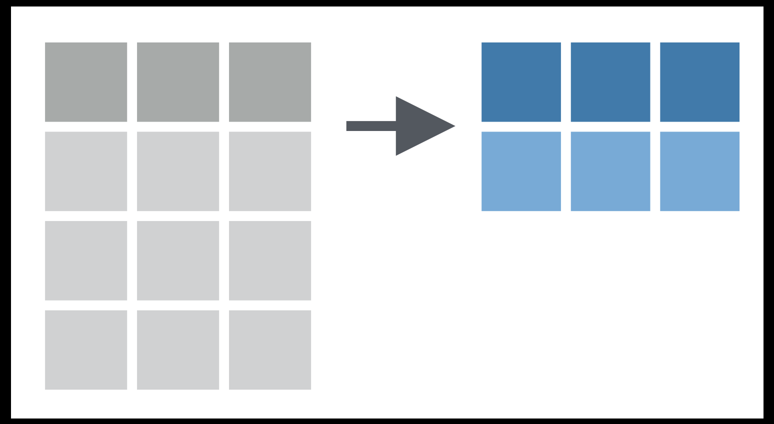
year	sex	name	n	prop
1880	F	Mary	7065	0.07238
1880	F	Anna	2604	0.02667
1880	F	Emma	2003	0.02052
1880	F	Elizabet	1939	0.01986
1880	F	Minnie	1746	0.01788
1880	F	Margare	1578	0.01616

→

n_rows
1924665

n_distinct()

The number of distinct values in a column/group.



```
babynames %>%
  summarise(n_rows = n(),
            n_names = n_distinct(name))
```

year	sex	name	n	prop
1880	F	Mary	7065	0.07238
1880	F	Anna	2604	0.02667
1880	F	Emma	2003	0.02052
1880	F	Elizabet	1939	0.01986
1880	F	Minnie	1746	0.01788
1880	F	Margare	1578	0.01616

n_rows	n_names
1924665	97310

group_by()

Grouping cases

GROUP_BY()

- `group_by()` changes the unit of analysis to *groups* in the data
- Any `dplyr` verbs used on a *grouped tibble* will be applied "by group"
- Especially useful when paired with `summarise()`
- To remove the grouping, either use `ungroup()` or use the optional argument `.groups` in the `summarise()` function for finer control

GROUP_BY()

Groups cases by common values of one or more columns.

```
babynames %>%
  group_by(sex)
```

```
Source: local data frame [1,825,433 x 5]
Groups: sex [2]

   year   sex      name     n       prop
  <dbl> <chr>     <chr> <int>      <dbl>
1  1880     F      Mary  7065 0.07238359
```

GROUP_BY()

Groups cases by common values of one or more columns.

```
babynames %>%
  group_by(sex) %>%
  summarise(total = sum(n))
```

Grouping variable

sex	total
F	172371079
M	175749438

New summary variable

Note that all other columns in original data are not in summary

PRACTICE DATA

~/OneDrive - University of Texas at San Antonio/Teaching/Data Visualization/activities/ant6973-activities - RStudio Source Editor

reshape.Rmd*

Knit · Insert · Run

```
---
title: "Tidy Data"
output: html_document
editor_options:
  chunk_output_type: console
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

library("gapminder")
library("tidyverse")
library("knitr")
```


```{r}
cases ← tibble(country = c("FR", "DE", "US"),
               `2011` = c(7000, 5800, 15000),
               `2012` = c(6900, 6000, 14000),
               `2013` = c(7000, 6200, 13000))
```


```{r}
pollution ← tibble(city = c("New York", "New Yor
                   size = c("large", "small", "l
                   amount = c(23, 14, 22, 16, 12
```

```

18:1 Chunk 2

```
pollution <- tibble(city = ...,
                    size = ...,
                    amount = ...)
```

```
pollution %>%
 summarise(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
18.5	37	2
19.0	38	2
88.5	177	2

```
group_by() + summarise()
```

city	particle size	amount (μg/m^3)
New York	large	23
New York	small	14

London	large	22
London	small	16

Beijing	large	121
Beijing	small	56

mean	sum	n
18.5	37	2
19.0	38	2
88.5	177	2

```
group_by() + summarise()
```

city	particle size	amount (µg/m3)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (µg/m3)
New York	large	23
New York	small	14

London	large	22
London	small	16

Beijing	large	121
Beijing	small	56

city	mean	sum	n
New York	18.5	37	2
London	19.0	38	2
Beijing	88.5	177	2

```
pollution %>%
  group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	mean	sum	n
New York	large	23	23	1
New York	small	14	14	1
London	large	22	22	1
London	small	16	16	1
Beijing	large	121	121	1
Beijing	small	56	56	1

```
pollution %>%
  group_by(city, size) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

ACTIVITY 3

- Use `group_by()` and `summarise()` to calculate the number of male and female babies born in each year.

```
babynames %>%
  group_by(year, sex) %>%
  summarise(n_babies = sum(n))
```

```
`summarise()` has grouped output by 'year'. You can override
using the `.groups` argument.
# A tibble: 276 x 3
# Groups:   year [138]
   year sex   n_babies
  <dbl> <chr>    <int>
1  1880 F        90993
2  1880 M       110491
3  1881 F        91953
4  1881 M       100743
5  1882 F       107847
6  1882 M       113686
7  1883 F       112319
8  1883 M       104627
9  1884 F       129020
# … with 267 more rows
```

Note that each call to `summarise()` typically removes a layer of grouping.

If output retains some grouping, dplyr notifies you

```r
babynames %>%
  group_by(year, sex) %>%
  summarise(n_babies = sum(n), .groups = "drop")
# A tibble: 276 x 3
    year sex   n_babies
   <dbl> <chr>    <int>
 1  1880 F        90993
 2  1880 M       110491
 3  1881 F        91953
 4  1881 M       100743
 5  1882 F       107847
 6  1882 M       113686
 7  1883 F       112319
 8  1883 M       104627
 9  1884 F       129020
10  1884 M       114442
# … with 266 more rows
```

Change this behavior with the optional **`.groups`** argument.

```
babynames %>%
  group_by(year, sex) %>%
  summarise(n_babies = sum(n), .groups = "keep")
# A tibble: 276 x 3
# Groups:   year, sex [276]
   year sex   n_babies
  <dbl> <chr>    <int>
1  1880 F        90993
2  1880 M       110491
3  1881 F        91953
4  1881 M       100743
5  1882 F       107847
6  1882 M       113686
7  1883 F       112319
8  1883 M       104627
9  1884 F       129020
# … with 267 more rows
```

Change this behavior with the optional `.groups` argument.

ACTIVITY 4

- On the storms data set, calculate the maximum wind speed and minimum pressure for each **hurricane** in each year, and arrange the summary in descending order of wind speed (hint: filter first).

name	year	month	day	hour	lat	long	status	category	wind	pressure	ts_diameter	hu_diameter
Amy	1975	6	27	0	27.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	27	6	28.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	27	12	29.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	27	18	30.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	28	0	31.5	-78.8	tropical depression	-1	25	1012	NA	NA
Amy	1975	6	28	6	32.4	-78.7	tropical depression	-1	25	1012	NA	NA

```r
storms %>%
  filter(status == "hurricane") %>%
  group_by(year, name) %>%
  summarise(wind_max = max(wind),
            pressure_min = min(pressure)) %>%
  arrange(desc(wind_max))
```

```
# Groups:   year [41]
   year name    wind_max pressure_min
  <dbl> <chr>      <dbl>        <dbl>
1  1988 Gilbert      160          888
2  2005 Wilma        160          882
3  1998 Mitch        155          905
4  2005 Rita         155          895
5  1977 Anita        150          926
6  1979 David        150          924
7  1992 Andrew       150          922
8  2005 Katrina      150          902
# … with 200 more rows
```

ACTIVITY 5

- Building on the previous code, calculate the average hurricane wind_max for each year. Which year had the most intense hurricanes, on average?

```
storms %>%
  filter(status == "hurricane") %>%
  group_by(year, name) %>%
  summarise(wind_max = max(wind)) %>%
  summarise(avg_wind_max = mean(wind_max)) %>%
  arrange(desc(avg_wind_max))
   year avg_wind_max
  <dbl>        <dbl>
1  1999        133.
2  1988        117.
3  1992        113.
4  2008        108.
5  2004        104.
6  2005        103.
7  2009        103.
8  2002         98.8
# … with 33 more rows
```

Second call to `summarise()` uses the year grouping only.

You might want to make the groupings explicit for readability.

```r
storms %>%
  filter(status == "hurricane") %>%
  group_by(year, name) %>%
  summarise(wind_max = max(wind), .groups = "drop_last") %>%
  summarise(avg_wind_max = mean(wind_max)) %>%
  arrange(desc(avg_wind_max))
   year avg_wind_max
  <dbl>        <dbl>
 1  1999        133.
 2  1988        117.
 3  1992        113.
 4  2008        108.
 7  2009        103.
 8  2002         98.8
# …
```

```
storms %>%
  filter(status == "hurricane") %>%
  group_by(year, name) %>%
  summarise(wind_max = max(wind)) %>%
  ungroup() %>%
  group_by(year) %>%
  summarise(avg_wind_max = mean(wind_max)) %>%
  arrange(desc(avg_wind_max))
   year avg_wind_max
  <dbl>        <dbl>
 1  1999        133.
 2  1988        117.
 3  1992        113.
 4  2008        108.
 7  2009        103.
 8  2002         98.8
# …
```

THINGS TO WATCH OUT FOR

- Many summary functions will return NA if there are any missing values
- Fortunately, many summary functions have an `na.rm = TRUE` argument to avoid this problem.

THINGS TO WATCH OUT FOR

- Once grouped, the tibble will remain that way unless grouping layers are removed. Be careful carrying out futher operations and summaries!
- It's good practice to use `ungroup()` after finishing your grouped operations.

ACTIVITY 6

- What is the average diameter of the area experiencing hurricane strength winds (`hu_diameter`) for each category of hurricane (`category`)?

```
storms %>%
  group_by(category) %>%
  summarise(mean_diameter = mean(hu_diameter))
  category mean_diameter
  <ord>            <dbl>
1 -1                  NA
2 0                   NA
3 1                   NA
4 2                   NA
5 3                   NA
6 4                   NA
7 5                   NA
```

Many of these values are missing

This causes the means to be NA

```
storms %>%
  group_by(category) %>%
  summarise(mean_diameter = mean(hu_diameter, na.rm = TRUE))
  category mean_diameter
  <ord>            <dbl>
1 -1                 0
2 0                  0
3 1                 57.3
4 2                 78.8
5 3                 91.4
6 4                102.
7 5                120.
```

Remove missing values before calculating means

COLUMN-WISE SUMMARIES

- Often you may need to perform the same summary or other operation across *multiple columns*.

- This can be accomplished by writing each column operation, but a more efficient and less error-prone approach is provided by the `across()` function.

Limited

```
df %>%
  group_by(g1, g2) %>%
  summarise(a = mean(a), b =
```

Better

```
df %>%
  group_by(g1, g2) %>%
  summarise(across(a:d, mean))
```

Inside of `across()`, the first argument is a `select()`-style expression of columns to summarize.

COLUMN-WISE SUMMARIES

- Often you may need to perform the same summary or other operation across *multiple columns*.

- This can be accomplished by writing each column operation, but a more efficient and less error-prone approach is provided by the `across()` function.

Limited

```
df %>%
  group_by(g1, g2) %>%
  summarise(a = mean(a), b = m
```

Better

```
df %>%
  group_by(g1, g2) %>%
  summarise(across(a:d, mean))
```

Second argument is the summary function (or a list of multiple functions)

COLUMN-WISE SUMMARIES

- Often you may need to perform the same summary or other operation across *multiple columns*.

- This can be accomplished by writing each column operation, but a more efficient and less error-prone approach is provided by the `across()` function.

Limited

```
df %>%
  group_by(g1, g2) %>%
  summarise(a = mean(a), b = m
```

Better

```
df %>%
  group_by(g1, g2) %>%
  summarise(across(a:d, mean, na.rm = TRUE))
```

Additional arguments for the summary function can be provided after a comma.

msleep (lots of missing values)

name	genus	vore	order	conservation	sleep_total	sleep_rem	sleep_cycle	awake	brainwt	bodywt
Cheetah	Acinonyx	carni	Carnivora	lc	12.1	NA	NA	11.9	NA	50.000
Owl monkey	Aotus	omni	Primates	NA	17.0	1.8	NA	7.0	0.01550	0.480
Mountain beaver	Aplodontia	herbi	Rodentia	nt	14.4	2.4	NA	9.6	NA	1.350
Greater short-tailed shrew	Blarina	omni	Soricomorpha	lc	14.9	2.3	0.1333333	9.1	0.00029	0.019
Cow	Bos	herbi	Artiodactyla	domesticated	4.0	0.7	0.6666667	20.0	0.42300	600.000
Three-toed sloth	Bradypus	herbi	Pilosa	NA	14.4	2.2	0.7666667	9.6	NA	3.850
Northern fur seal	Callorhinus	carni	Carnivora	vu	8.7	1.4	0.3833333	15.3	NA	20.490
Vesper mouse	Calomys	NA	Rodentia	NA	7.0	NA	NA	17.0	NA	0.045
Dog	Canis	carni	Carnivora	domesticated	10.1	2.9	0.3333333	13.9	0.07000	14.000
Roe deer	Capreolus	herbi	Artiodactyla	lc	3.0	NA	NA	21.0	0.09820	14.800

ACTIVITY 7A

- For each vore, calculate the average of *only the columns that start with "sleep"* in the msleep data set.
  - Hint: use `na.rm = TRUE`

```
msleep %>%
  group_by(vore) %>%
  summarise(across(starts_with("sleep"), mean, na.rm = TRUE))
  vore    sleep_total sleep_rem sleep_cycle
  <chr>         <dbl>     <dbl>       <dbl>
1 NA            10.2       1.88       0.183
2 carni         10.4       2.29       0.373
3 herbi          9.51      1.37       0.418
4 insecti       14.9       3.52       0.161
5 omni          10.9       1.96       0.592
```

Equivalent to…

```
msleep %>%
  group_by(vore) %>%
  summarise(sleep_total = mean(sleep_total, na.rm = TRUE),
            sleep_rem = mean(sleep_rem, na.rm = TRUE),
            sleep_cycle = mean(sleep_cycle, na.rm = TRUE))
  vore    sleep_total sleep_rem sleep_cycle
  <chr>         <dbl>     <dbl>       <dbl>
1 NA            10.2       1.88       0.183
2 carni         10.4       2.29       0.373
3 herbi          9.51      1.37       0.418
4 insecti       14.9       3.52       0.161
5 omni          10.9       1.96       0.592
```

ACTIVITY 7B

- For each vore, calculate the average of *only the numeric columns* in the msleep data set.
  - Hint: use `na.rm = TRUE`

```
msleep %>%
  group_by(vore) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE))
  vore    sleep_total sleep_rem sleep_cycle awake brainwt  bodywt
  <chr>         <dbl>     <dbl>       <dbl> <dbl>   <dbl>   <dbl>
1 NA            10.2       1.88       0.183 13.8  0.00763   0.858
2 carni         10.4       2.29       0.373 13.6  0.0793   90.8
3 herbi          9.51      1.37       0.418 14.5  0.622   367.
4 insecti       14.9       3.52       0.161  9.06 0.0216   12.9
5 omni          10.9       1.96       0.592 13.1  0.146    12.7
```

Grouping variable isn't numeric, but it still appears in the summary.

ACTIVITY 7C

- For each vore, calculate the average of *all* columns in the msleep data set.
  - Hint: use `na.rm = TRUE`

```
msleep %>%
  group_by(vore) %>%
  summarise(across(everything(), mean, na.rm = TRUE))
```

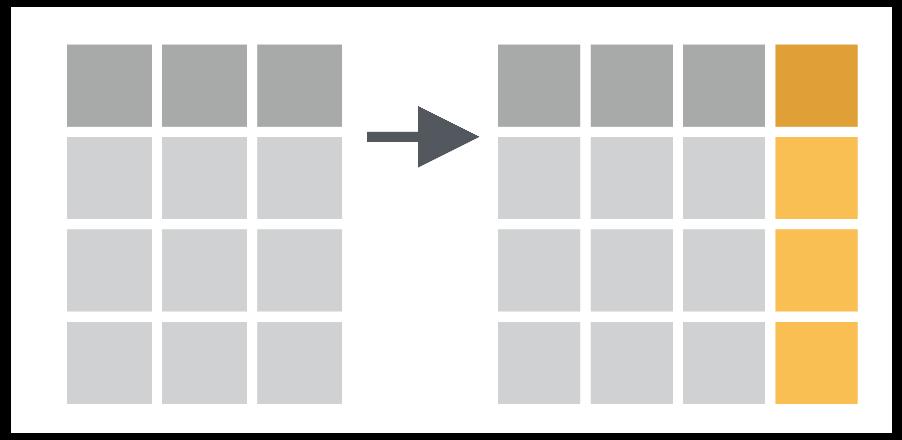
	vore	name	genus	order	conservation	sleep_total	sleep_rem	sleep_cycle	awake	brainwt	bodywt
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	carni	NA	NA	NA	NA	10.4	2.29	0.373	13.6	0.0793	90.8
2	herbi	NA	NA	NA	NA	9.51	1.37	0.418	14.5	0.622	367.
3	insecti	NA	NA	NA	NA	14.9	3.52	0.161	9.06	0.0216	12.9
4	omni	NA	NA	NA	NA	10.9	1.96	0.592	13.1	0.146	12.7
5	NA	NA	NA	NA	NA	10.2	1.88	0.183	13.8	0.00763	0.858

Note that mean() doesn't make sense for the text variables.

OTHER GROUPED OPERATIONS

- `group_by()` can also be used with `mutate()` and `filter()` to do some interesting things.

- Reminder: `mutate()` creates a new variable of the same length as the original data.



ACTIVITY 8

- In babynames, try to recreate the `prop` column using a grouped mutate (call it "`new_prop`"). Specifically, divide each row's n by the total number of n for that sex and year.

- Why are the values slightly different?

```r
babynames %>%
  group_by(sex, year) %>%
  mutate(grp_sum = sum(n),
         new_prop = n  / grp_sum)
```

Current *group's* total n

Current row's new_prop

```
# Groups:   sex, year [276]
    year sex   name          n   prop grp_sum new_prop
   <dbl> <chr> <chr>     <int>  <dbl>   <int>    <dbl>
 1  1880 F     Mary       7065 0.0724   90993   0.0776
 2  1880 F     Anna       2604 0.0267   90993   0.0286
 3  1880 F     Emma       2003 0.0205   90993   0.0220
 4  1880 F     Elizabeth  1939 0.0199   90993   0.0213
 5  1880 F     Minnie     1746 0.0179   90993   0.0192
 6  1880 F     Margaret   1578 0.0162   90993   0.0173
 7  1880 F     Ida        1472 0.0151   90993   0.0162
 8  1880 F     Alice      1414 0.0145   90993   0.0155
 9  1880 F     Bertha     1320 0.0135   90993   0.0145
10  1880 F     Sarah      1288 0.0132   90993   0.0142
# … with 1,924,655 more rows
```

Note that the result has as many rows as the original data (it's not a summary)

```
babynames %>%
  group_by(sex, year) %>%
  mutate(new_prop = n / sum(n))
# Groups:   sex, year [276]
    year sex   name          n   prop new_prop
   <dbl> <chr> <chr>     <int>  <dbl>    <dbl>
 1  1880 F     Mary       7065 0.0724   0.0776
 2  1880 F     Anna       2604 0.0267   0.0286
 3  1880 F     Emma       2003 0.0205   0.0220
 4  1880 F     Elizabeth  1939 0.0199   0.0213
 5  1880 F     Minnie     1746 0.0179   0.0192
 6  1880 F     Margaret   1578 0.0162   0.0173
 7  1880 F     Ida        1472 0.0151   0.0162
 8  1880 F     Alice      1414 0.0145   0.0155
 9  1880 F     Bertha     1320 0.0135   0.0145
10  1880 F     Sarah      1288 0.0132   0.0142
# … with 1,924,655 more rows
```

More concisely...

Group denominator calculated for each for each row

MORE ON GROUPED MUTATES

- Grouped mutates are useful for calculating deviations, ranks, and other row-level values within groups.

ACTIVITY 9A

- Using msleep, determine how much each species' log body weight differs from the average log body weight for its order.

name	genus	vore	order	conservation	sleep_total	sleep_rem	sleep_cycle	awake	brainwt	bodywt
Cheetah	Acinonyx	carni	Carnivora	lc	12.1	NA	NA	11.9	NA	50.000
Owl monkey	Aotus	omni	Primates	NA	17.0	1.8	NA	7.0	0.01550	0.480
Mountain beaver	Aplodontia	herbi	Rodentia	nt	14.4	2.4	NA	9.6	NA	1.350
Greater short-tailed shrew	Blarina	omni	Soricomorpha	lc	14.9	2.3	0.1333333	9.1	0.00029	0.019
Cow	Bos	herbi	Artiodactyla	domesticated	4.0	0.7	0.6666667	20.0	0.42300	600.000
Three-toed sloth	Bradypus	herbi	Pilosa	NA	14.4	2.2	0.7666667	9.6	NA	3.850
Northern fur seal	Callorhinus	carni	Carnivora	vu	8.7	1.4	0.3833333	15.3	NA	20.490

```r
msleep %>%
  group_by(order) %>%
  mutate(log_bodywt = log(bodywt),
         order_mean = mean(log_bodywt, na.rm = TRUE),
         bodywt_dev = log_bodywt - order_mean)
```

Each row's log bodywt

Current order's mean bodywt

Current row's deviation from mean of current order

```
# Groups:   order [19]
   name             genus    vore  order    conservation sleep_total sleep_rem sleep_cycle awake  brainwt  bodywt log_bodywt order_m
   <chr>            <chr>    <chr> <chr>    <chr>              <dbl>     <dbl>       <dbl> <dbl>    <dbl>   <dbl>      <dbl>    <d
 1 Cheetah          Acinon… carni Carniv… lc                  12.1      NA          NA      11.9 NA        50          3.91       3
 2 Owl monkey       Aotus    omni  Primat… NA                  17         1.8        NA       7    0.0155    0.48      -0.734      1.
 3 Mountain beav… Aplodo… herbi Rodent… nt                  14.4       2.4        NA       9.6 NA         1.35       0.300     -1.98      2.28
 4 Greater short… Blarina omni  Sorico… lc                  14.9       2.3         0.133   9.1  0.00029   0.019     -3.96      -3.54     -0.423
 5 Cow              Bos      herbi Artiod… domesticated         4         0.7         0.667  20    0.423   600          6.40       4.65      1.75
 6 Three-toed sl… Bradyp… herbi Pilosa   NA                  14.4       2.2         0.767   9.6 NA         3.85       1.35       1.35      0
 7 Northern fur … Callor… carni Carniv… vu                   8.7       1.4         0.383  15.3 NA        20.5        3.02       3.15     -0.130
 8 Vesper mouse     Calomys NA    Rodent… NA                   7        NA          NA      17    NA         0.045     -3.10      -1.98     -1.12
 9 Dog              Canis    carni Carniv… domesticated        10.1       2.9         0.333  13.9  0.07     14          2.64       3.15     -0.511
10 Roe deer         Capreo… herbi Artiod… lc                   3        NA          NA      21     0.0982   14.8        2.69       4.65     -1.95
```

ACTIVITY 9B

- Using baby names, add a rank column to each name for each year and sex. What were the top 10 ranked boys names in 2015, and what were their ranks?

```r
babynames %>%
  group_by(year, sex) %>%
  mutate(rank = min_rank(desc(prop))) %>%
  filter(year == 2015 & sex == "M" & rank <= 10)
# Groups:   year, sex [1]
    year sex   name           n    prop  rank
   <dbl> <chr> <chr>      <int>   <dbl> <int>
 1  2015 M     Noah       19613 0.00962     1
 2  2015 M     Liam       18355 0.00900     2
 3  2015 M     Mason      16610 0.00815     3
 4  2015 M     Jacob      15938 0.00782     4
 5  2015 M     William    15889 0.00780     5
 6  2015 M     Ethan      15069 0.00739     6
 7  2015 M     James      14799 0.00726     7
 8  2015 M     Alexander  14531 0.00713     8
 9  2015 M     Michael    14413 0.00707     9
10  2015 M     Benjamin   13692 0.00672    10
```

Each row's rank within year and sex

Get top 10 for 2015 male babies.

ACTIVITY 10A

- When building statistical models, it's often useful to standardize numeric variables by converting to z-scores:

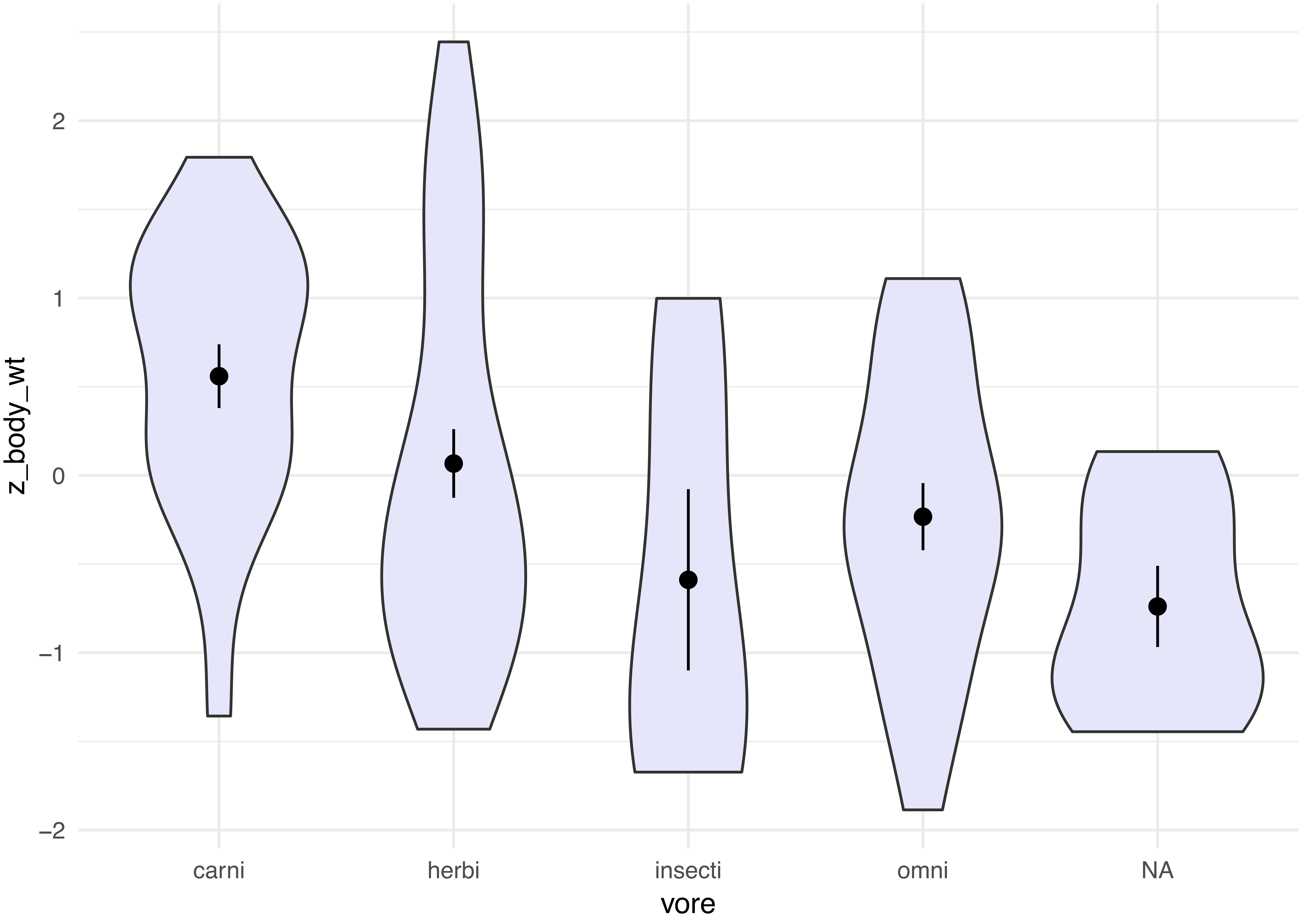
    - z = (observed value - mean) / standard deviation

- Using msleep, use `mutate()` to calculate the z-score for log(`bodywt`) for each row in the data set.

- Plot the data using a violin plot and `stat_summary()`

```
ggplot(msleep, aes(x = vore, y = z_log_bodywt)) +
  geom_violin() +
  stat_summary(fun.data = "mean_se")
```

```r
msleep %>%
  mutate(log_bodywt = log(bodywt),
         z_log_bodywt = (log_bodywt - mean(log_bodywt, na.rm = TRUE)) /
                        sd(log_bodywt, na.rm = TRUE)) %>%
  ggplot(aes(x = vore, y = z_log_bodywt)) +
  geom_violin() +
  stat_summary(fun.data = "mean_se")
```



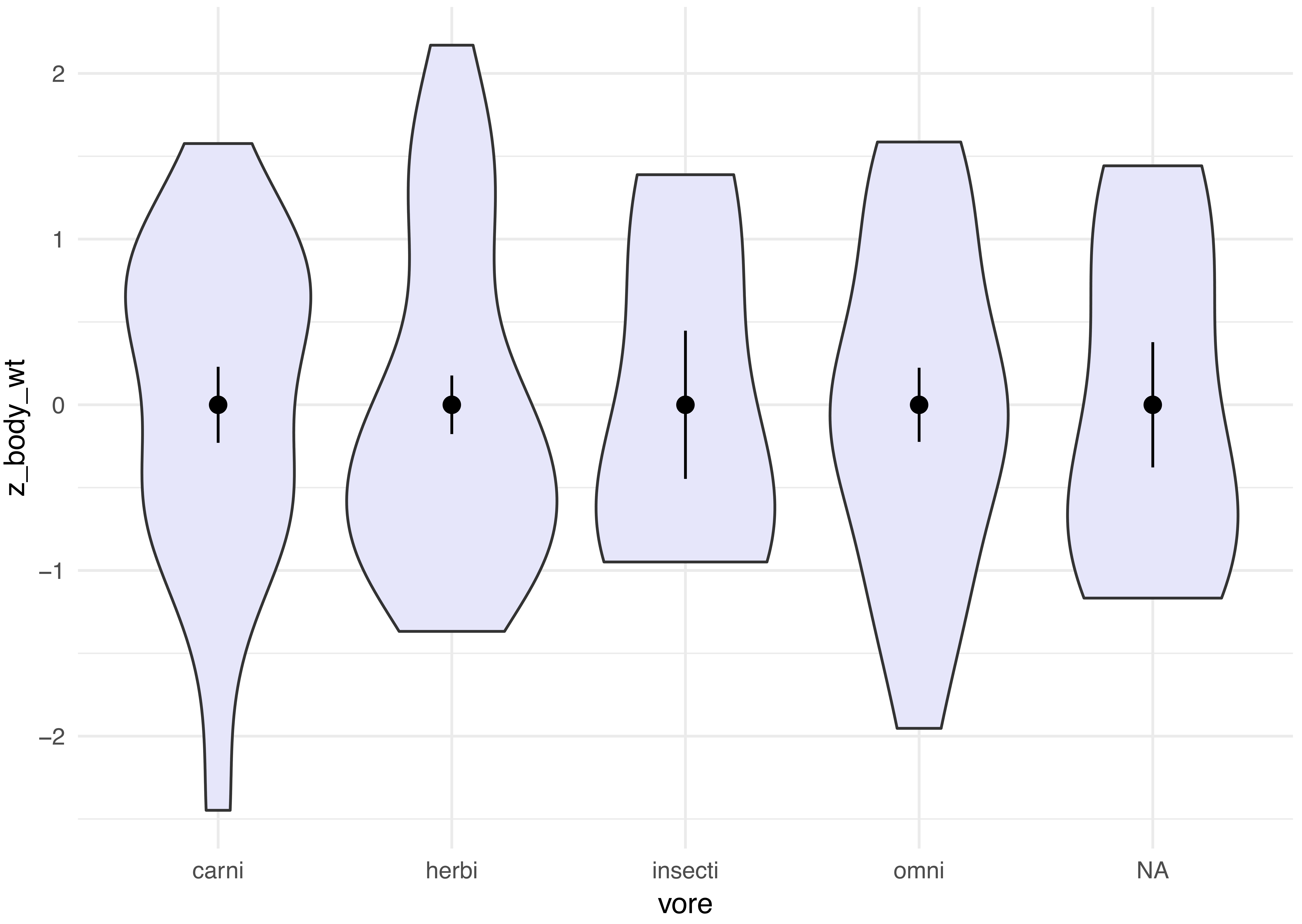
z_body_wt
2
1
0
−1
−2
carni
herbi
insecti
omni
NA
vore

ACTIVITY 10B

- Now do the same, but standardize *within groups* by using `group_by(vore)`.

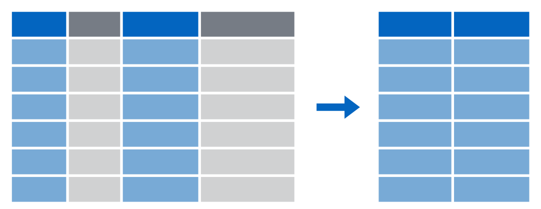
```
ggplot(msleep, aes(x = vore, y = z_log_bodywt)) +
  geom_violin() +
  stat_summary(fun.data = "mean_se")
```

```
msleep %>%
  mutate(log_bodywt = log(bodywt)) %>%
  group_by(vore) %>%
  mutate(z_log_bodywt = (log_bodywt - mean(log_bodywt, na.rm = TRUE)) /
                        sd(log_bodywt, na.rm = TRUE)) %>%
  ggplot(aes(x = vore, y = z_log_bodywt)) +
  geom_violin() +
  stat_summary(fun.data = "mean_se")
```

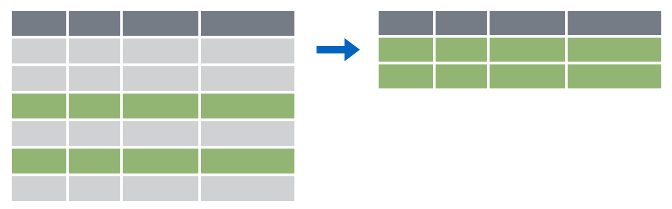


2
1
0
−1
−2
z_body_wt
carni
herbi
insecti
omni
NA
vore

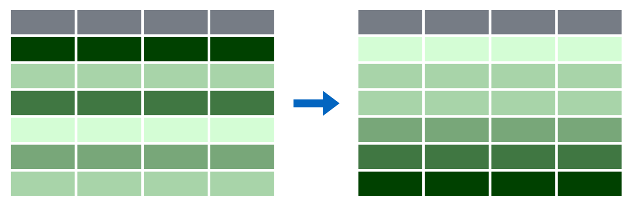
SINGLE TABLE VERBS



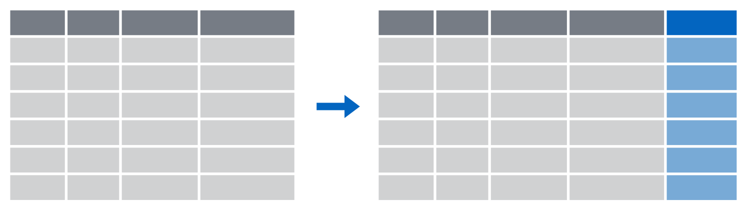
Extract variables with `select()`



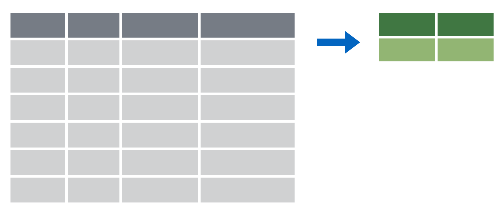
Extract cases with `filter()`



Arrange cases with `arrange()`



Make new variables with `mutate()`



Make tables of summaries with `summarise()`
along with `group_by()`