Data Types

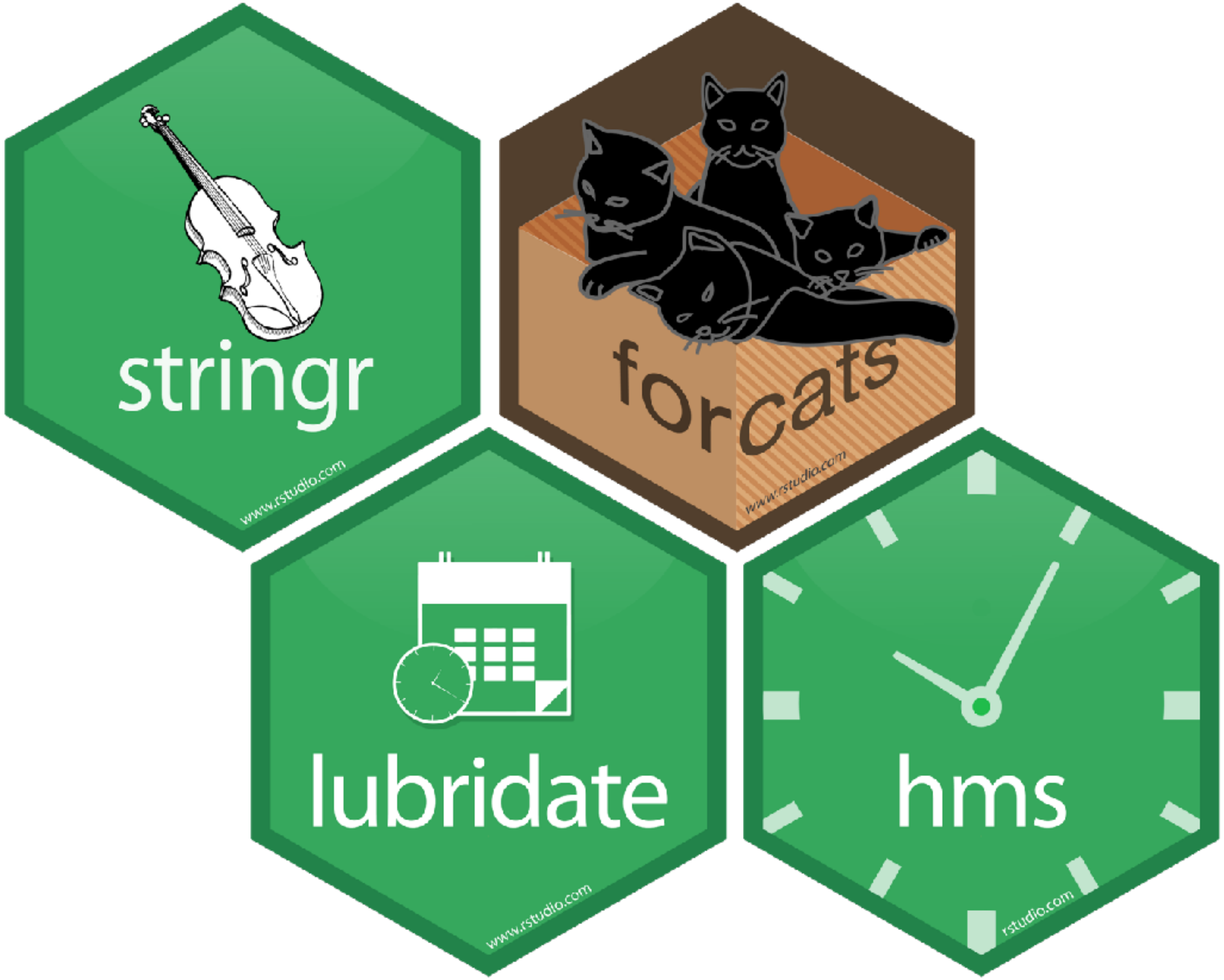
Jake Thompson

wjakethompson.com

/ @wjakethompson

ATLAS

Accessible Teaching, Learning, and Assessment Systems


stringr
www.rstudio.com
forcats
www.rstudio.com
lubridate
www.rstudio.com
hms
rstudio.com

ATLAS

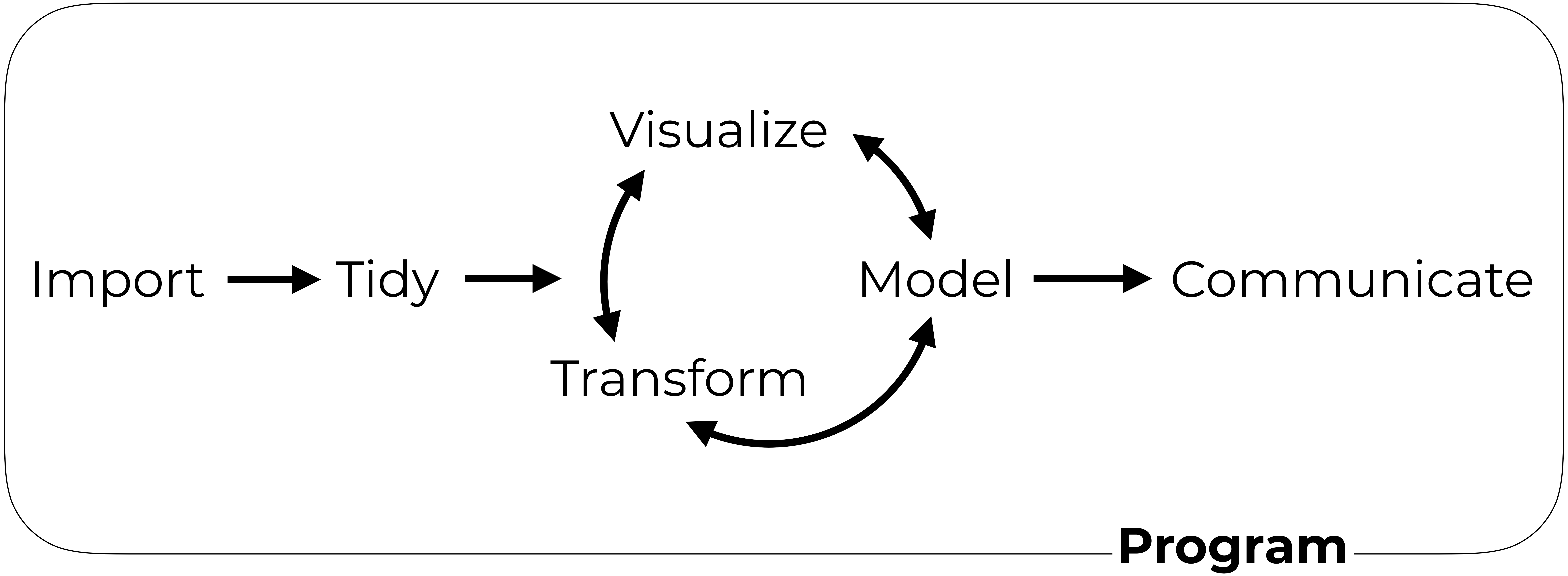
Your Turn 0

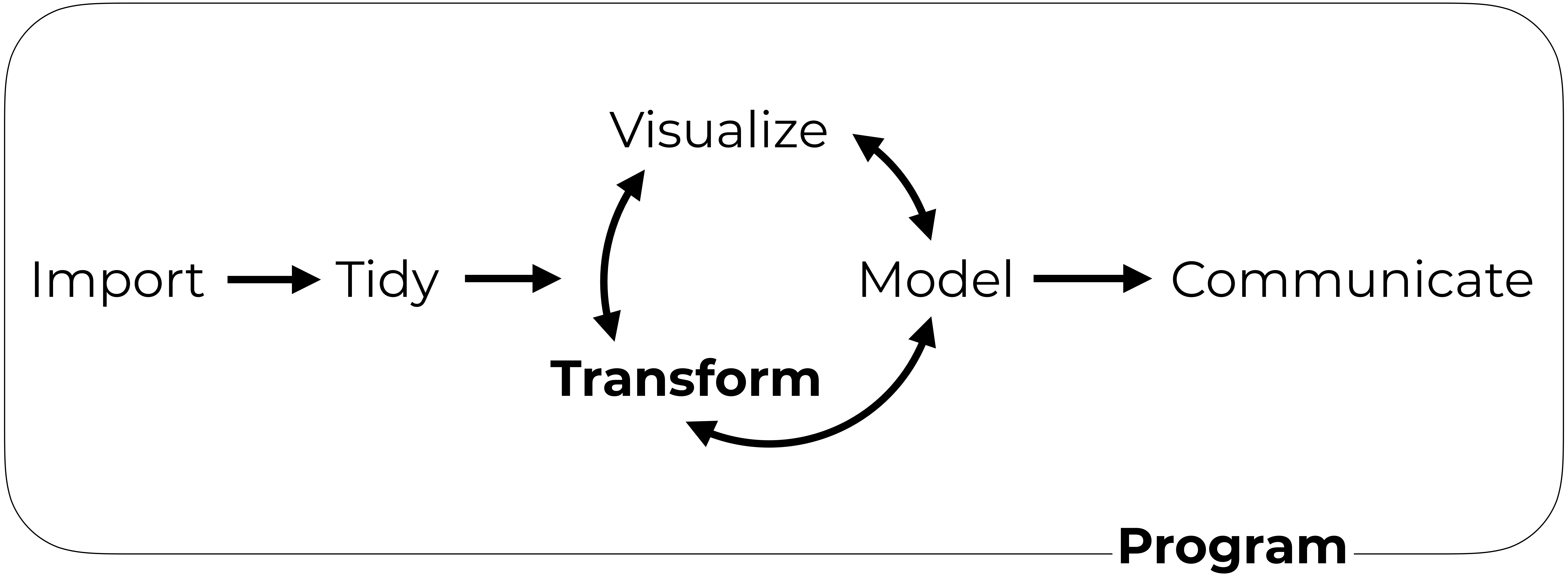
- Open **03-Data-Types.Rmd**
- Run the setup chunk

Quiz

What types of data are in this data set?

	time_hour	name	air_time	distance	day	delayed
1	2013-01-01 05:00:00	United Air Lines Inc.	13620s (~3.78 hours)	1400	Tuesday	TRUE
2	2013-01-01 05:00:00	United Air Lines Inc.	13620s (~3.78 hours)	1416	Tuesday	TRUE
3	2013-01-01 05:00:00	American Airlines Inc.	9600s (~2.67 hours)	1089	Tuesday	TRUE
4	2013-01-01 05:00:00	JetBlue Airways	10980s (~3.05 hours)	1576	Tuesday	FALSE
5	2013-01-01 06:00:00	Delta Air Lines Inc.	6960s (~1.93 hours)	762	Tuesday	FALSE
6	2013-01-01 05:00:00	United Air Lines Inc.	9000s (~2.5 hours)	719	Tuesday	TRUE
7	2013-01-01 06:00:00	JetBlue Airways	9480s (~2.63 hours)	1065	Tuesday	TRUE
8	2013-01-01 06:00:00	ExpressJet Airlines Inc.	3180s (~53 minutes)	229	Tuesday	FALSE
9	2013-01-01 06:00:00	JetBlue Airways	8400s (~2.33 hours)	944	Tuesday	FALSE
10	2013-01-01 06:00:00	American Airlines Inc.	8280s (~2.3 hours)	733	Tuesday	TRUE


Visualize
Import
Tidy
Model
Communicate
Transform
Program


Visualize
Import
Tidy
Model
Communicate
Transform
Program

Logicals

Logicals

R's data type for boolean values (i.e., TRUE and FALSE)

```
typeof(TRUE)
# "logical"

typeof(FALSE)
# "logical"

typeof(c(TRUE, TRUE, FALSE))
# "logical"
```

```
flights %>%
  mutate(delayed = arr_delay > 0) %>%
  select(arr_delay, delayed)
# A tibble: 336,776 x 2
#    arr_delay delayed
#        <dbl> <lgl>
#  1        11 TRUE
#  2        20 TRUE
#  3        33 TRUE
#  4       -18 FALSE
#  5       -25 FALSE
#  6        12 TRUE
#  7        19 TRUE
#  8       -14 FALSE
#  9        -8 FALSE
# 10         8 TRUE
# ... with 336,766 more rows
```

Can we compute the proportion of flights that arrived late?

ATLAS

Most useful skills

- Math with logicals
  - When you do math with logicals **TRUE becomes 1** and **FALSE becomes 0**.
- The **sum** of a logical vector is the **count of TRUEs**

```
sum(c(TRUE, FALSE, TRUE, TRUE))
# 3
```

- The **mean** of a logical vector is the **proportion of TRUEs**

```
mean(c(1, 2, 3, 4) < 4)
# 0.75
```

ATLAS

Your Turn 1

Use flights to create **delayed**, a variable that displays whether a flight was delayed (**arr_delay > 0**).

Then, remove all rows that contain an NA in **delayed**.

Finally, create a summary table that shows:

1. How many flights were delayed?
2. What proportion of flights were delayed?

04 : 00

ATLAS

```
flights %>%
  mutate(delayed = arr_delay > 0) %>%
  drop_na(delayed) %>%
  summarize(total = sum(delayed), prop = mean(delayed))
# A tibble: 1 x 2
#    total  prop
#    <int> <dbl>
# 1 133004 0.406
```

Strings

(character) **strings**

Anything surrounded by quotes(") or single quotes(')

```
> "one"
> "1"
> "one's"
> '"Hello World"'
> "foo
+
+
+ oops. I'm stuck in a string."
```

ATLAS

Warm Up

Decide in your group:

Are boys names or girls names more likely too end in a vowel?

01:00

ATLAS

```
babynames
# A tibble: 1,924,665 x 5
#       year sex   name           n    prop
#      <dbl> <chr> <chr>      <int>   <dbl>
#   1   1880 F     Mary        7065 0.07
#   2   1880 F     Anna        2604 0.0
#   3   1880 F     Emma        2003 0.0
#   4   1880 F     Elizabeth   1939 0.0
#   5   1880 F     Minnie      1746 0.0
#   6   1880 F     Margaret    1578 0.0
#   7   1880 F     Ida         1472 0.01
#   8   1880 F     Alice       1414 0.0145
#   9   1880 F     Bertha      1320 0.0135
#  10   1880 F     Sarah       1288 0.0132
# … with 1,924,655 more rows
```

How can we build the proportion of boys and girls whose name ends in a vowel?

ATLAS

Most useful skills

1. How to extract / replace substrings
2. How to find matches for patterns
3. Regular expressions

stringr


In R4DS

[Strings]

ATLAS

str_sub()

Extract or replace portions of a string with **str_sub()**

```
str_sub(string, start = 1, end = -1)
```

string(s) to manipulate

position of first character to extract within each string

position of last character to extract within each string

Quiz

What will this return?

```
str_sub("Garrett", 1, 2)
```

```
"Ga"
```

Quiz

What will this return?

```
str_sub("Garrett", 1, 1)
```

```
"G"
```

Quiz

What will this return?

```
str_sub("Garrett", 2)
```

```
"arrett"
```

ATLAS

stringr

Quiz

What will this return?

```
str_sub("Garrett", -3)
```

```
"ett"
```

Quiz

What will this return?

```
g <- "Garrett"
str_sub(g, -3) <- "eth"
g
```

```
"Garreth"
```

Your Turn 2

Fill in the blanks to:

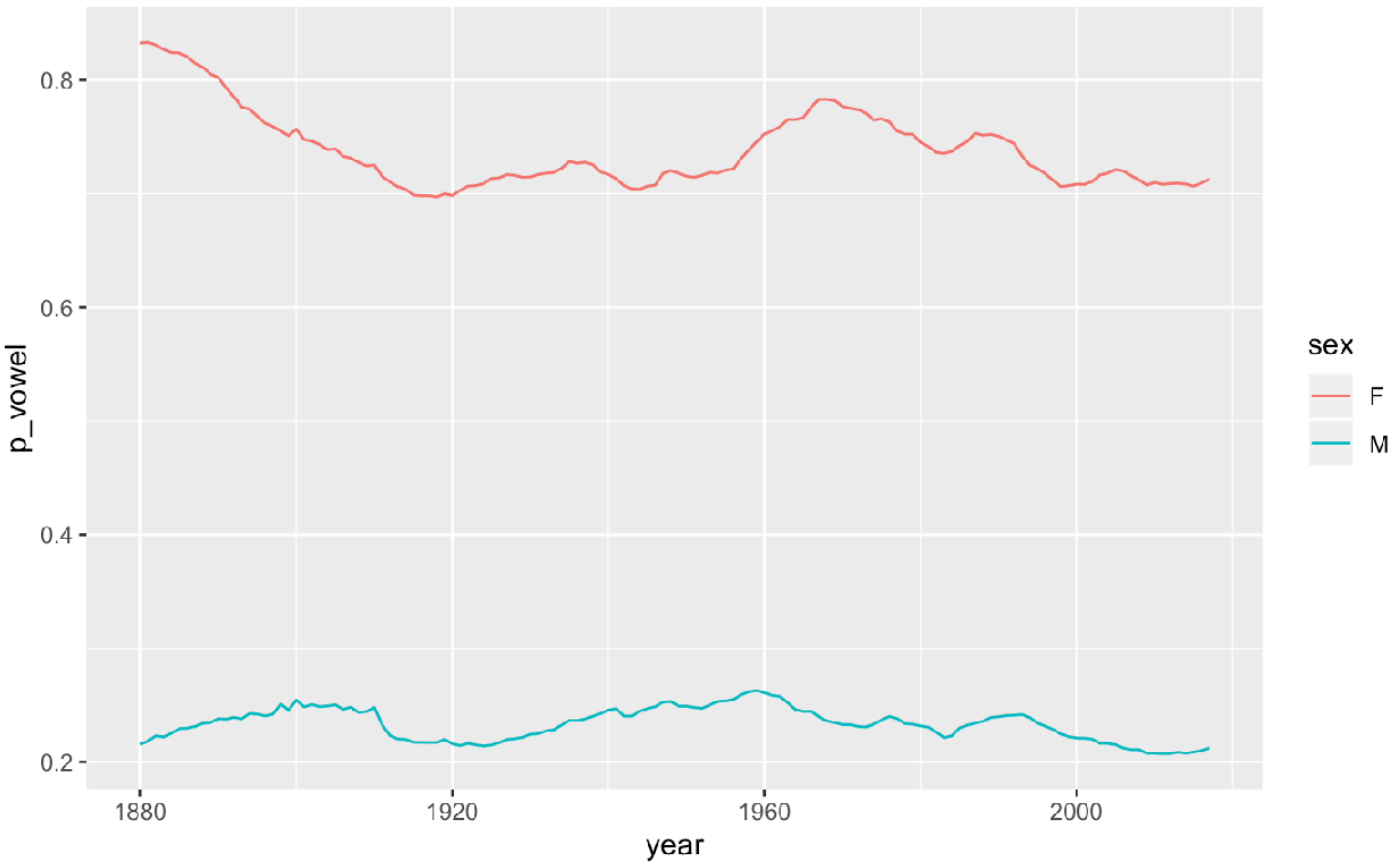
1. Isolate the last letter of every name
2. and create a logical variable that displays whether the last letter is one of "a", "e", "i", "o", "u", or "y".
3. Use a weighted mean to calculate the proportion of children whose name ends in a vowel (by year and sex)
4. and then display the results as a line plot.

05:00

ATLAS

stringr

```
babynames %>%
  mutate(last = str_sub(name, -1),
         vowel = last %in% c("a", "e", "i", "o", "u", "y")) %>%
  group_by(year, sex) %>%
  summarize(p_vowel = weighted.mean(vowel, n)) %>%
  ggplot() +
    geom_line(mapping = aes(year, p_vowel, color = sex))
```



0.8
0.6
0.4
0.2
p_vowel
1880
1920
1960
2000
year
sex
F
M

Factors

Factors

R's representation of categorical data. Consists of:

1. A set of **values**
2. An ordered set of **valid levels**

```
eyes <- factor(x = c("blue", "green", "green"),
               levels = c("blue", "brown", "green"))
eyes
# [1] blue  green green
# Levels: blue brown green
```

Factors

Stored as an integer vector with a levels attribute

```
unclass(eyes)
# [1] 1 3 3
# attr(,"levels")
# [1] "blue"  "brown" "green"
```

ATLAS

In R4DS

Factors

ATLAS

Warm Up

Decide in your group:

Do married people watch more or less TV than single people?

01:00

gss_cat

A sample of data from the General Social Survey, a long-running US survey conducted by NORC at the University of Chicago.

	tvhours	marital	age	race	partyid	relig
1	12	Never married	26	White	Ind,near rep	Protestant
2	NA	Divorced	48	White	Not str republican	Protestant
3	2	Widowed	67	White	Independent	Protestant
4	4	Never married	39	White	Ind,near rep	Orthodox-christian
5	1	Divorced	25	White	Not str democrat	None
6	NA	Married	25	White	Strong democrat	Protestant
7	3	Never married	36	White	Not str republican	Christian
8	NA	Divorced	44	White	Ind,near dem	Protestant

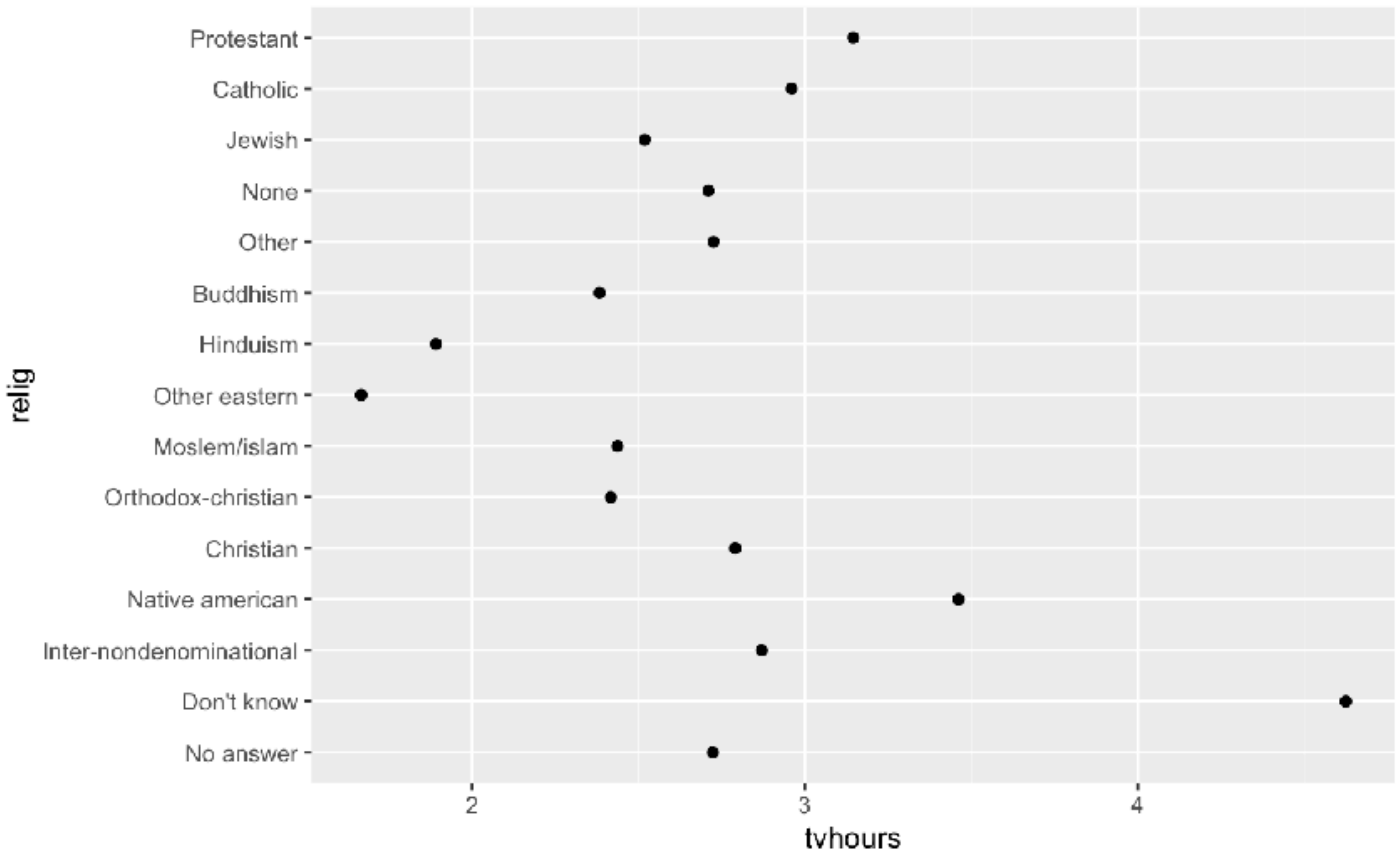
ATLAS

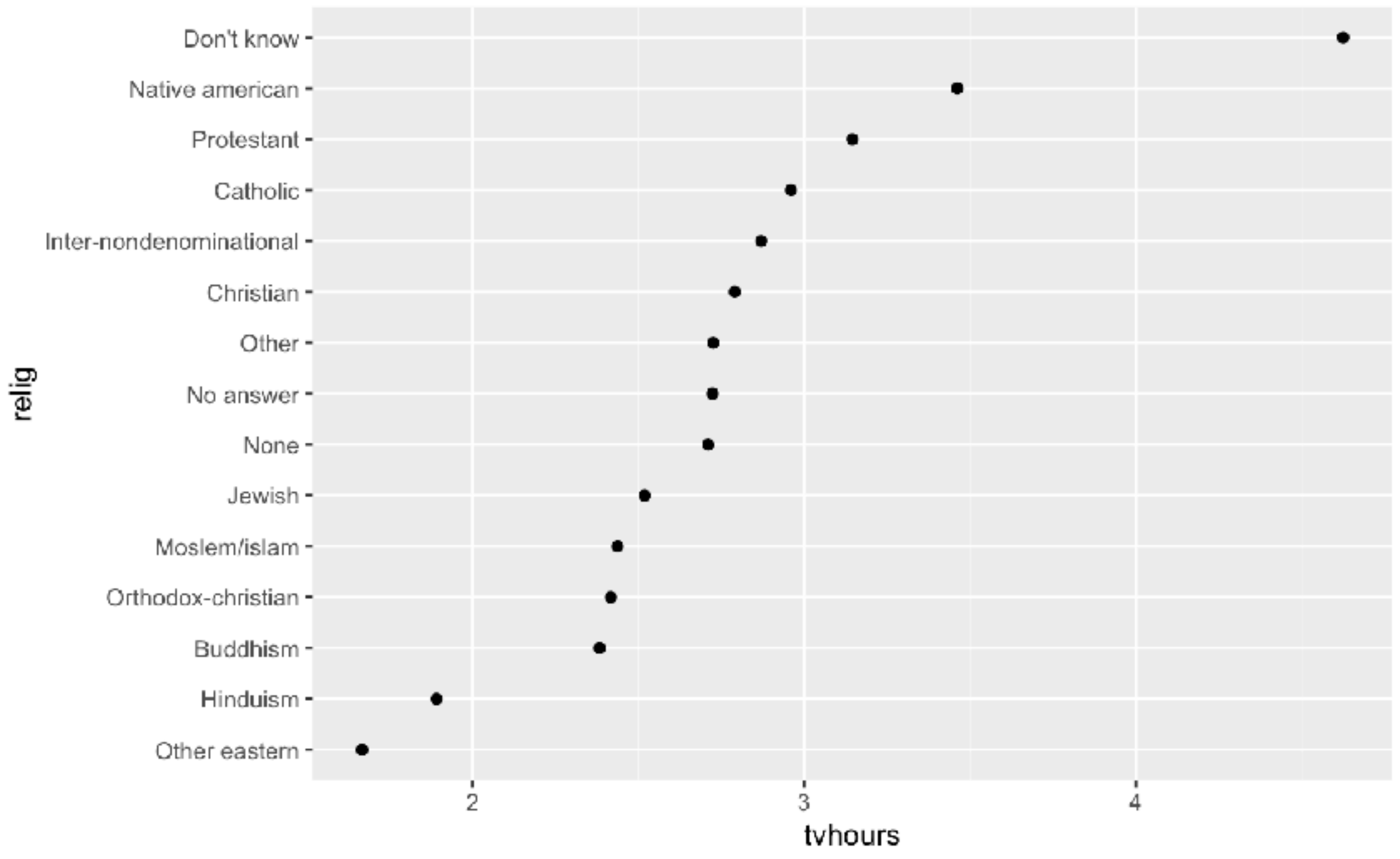
Which religions watch the least TV?

```
gss_cat %>%
  drop_na(tvhours) %>%
  group_by(relig) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(mapping = aes(x = tvhours, y = relig)) +
    geom_point()
```

ATLAS

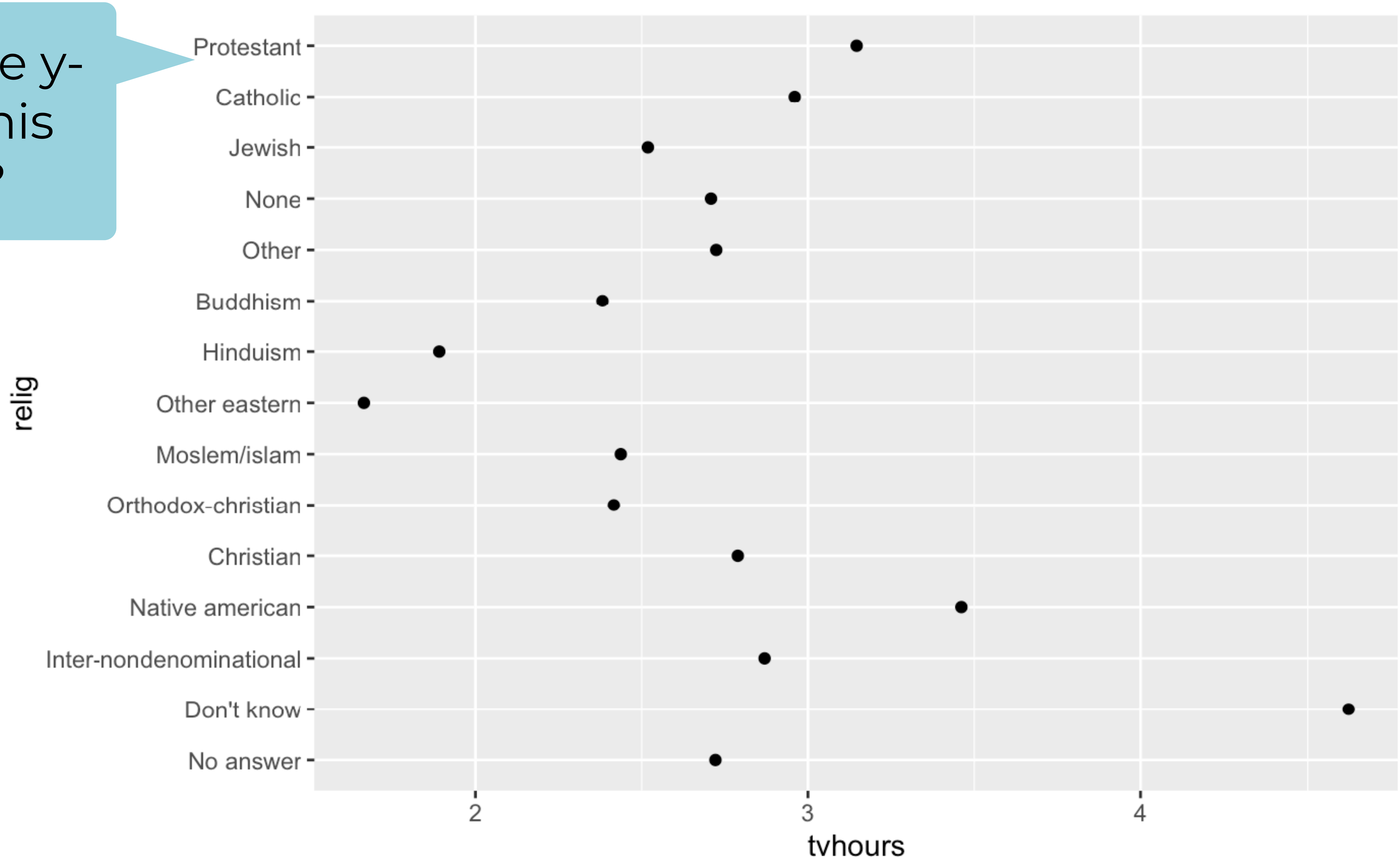
Which do you prefer?





ATLAS

Why is the y-axis in this order?



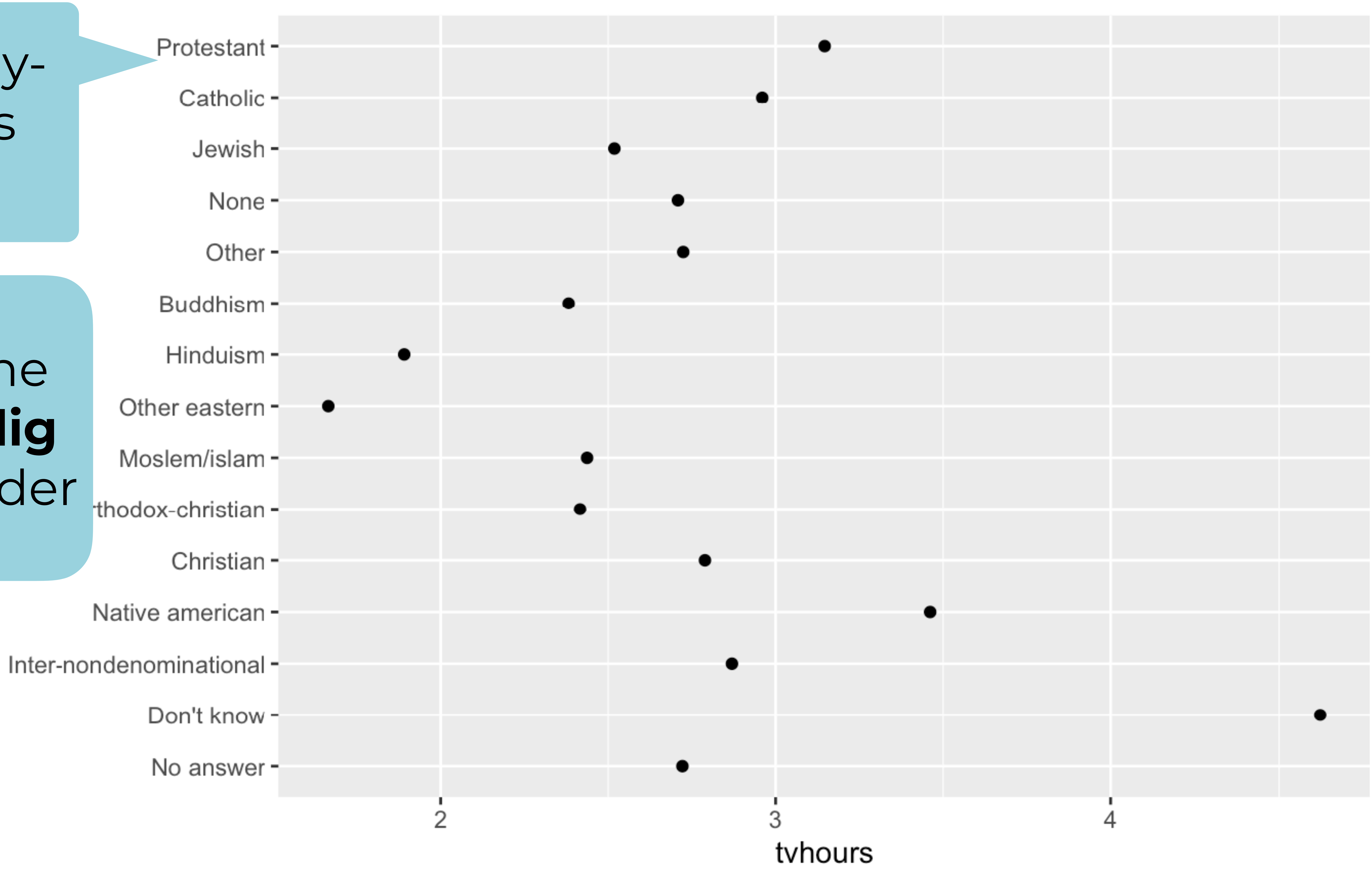
levels()

Use **levels()** to access a factor's levels

```
levels(gss_cat$relig)
#   [1] "No answer"               "Don't know"
#   [3] "Inter-nondenominational" "Native american"
#   [5] "Christian"               "Orthodox-christian"
#   [7] "Moslem/islam"            "Other eastern"
#   [9] "Hinduism"                "Buddhism"
#  [11] "Other"                   "None"
#  [13] "Jewish"                  "Catholic"
#  [15] "Protestant"              "Not applicable"
```

Why is the y-axis in this order?

Because the levels of **relig** have this order



Most useful skills

1. Reorder the levels
2. Recode the levels
3. Collapse levels

Reordering levels

fct_reorder()

Reorders the levels of a factor based on the result of **fun(x)** applied to each group of cases (grouped by level).

```
fct_reorder(f, x, fun = median, ..., .desc = FALSE)
```

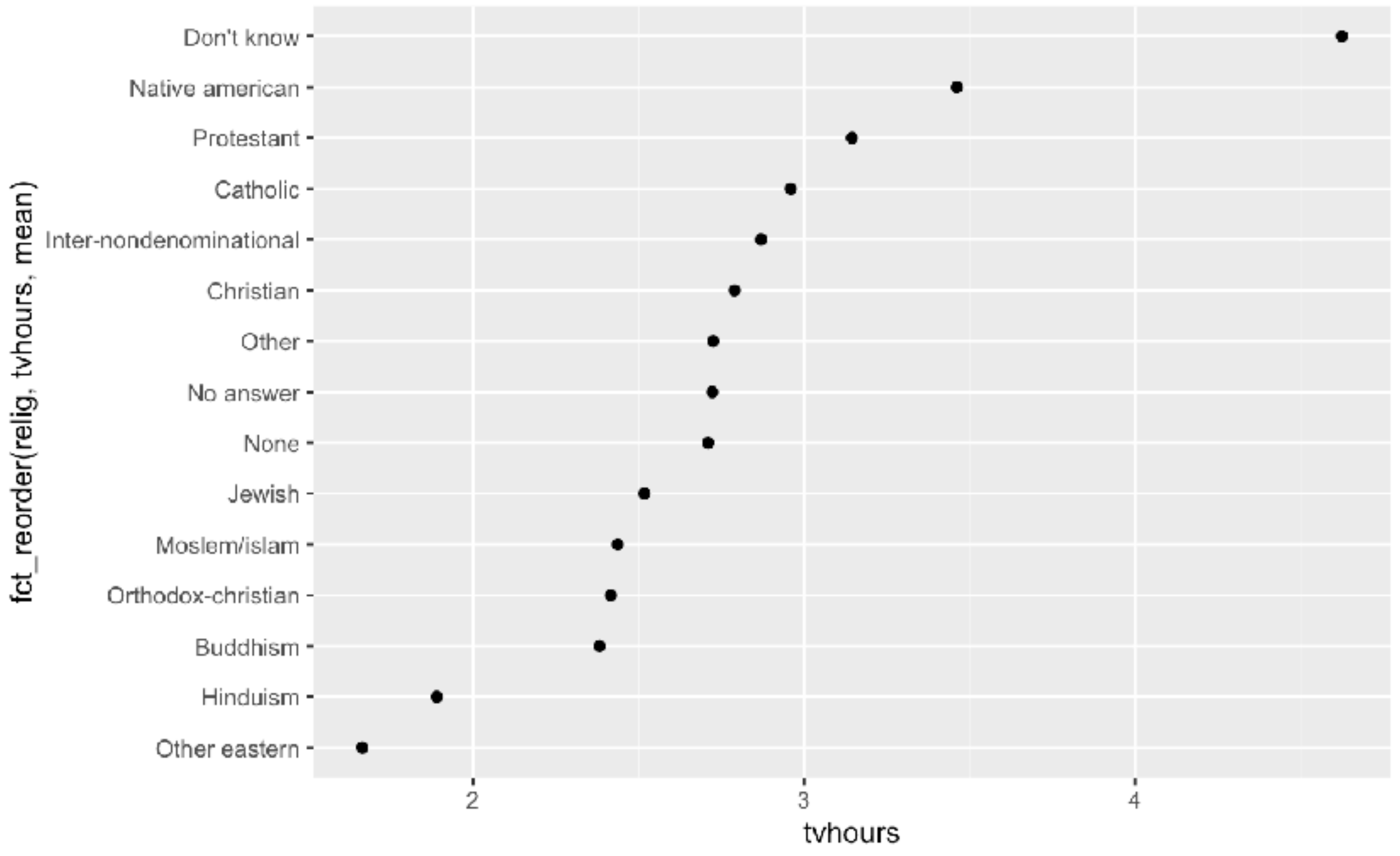
factor to reorder

variable to reorder by (in conjunction with fun)

function to reorder by (in conjunction with x)

put in descending order?

```r
gss_cat %>%
  drop_na(tvhours) %>%
  group_by(relig) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(mapping = aes(x = tvhours,y = fct_reorder(relig, tvhours, mean))) +
    geom_point()
```

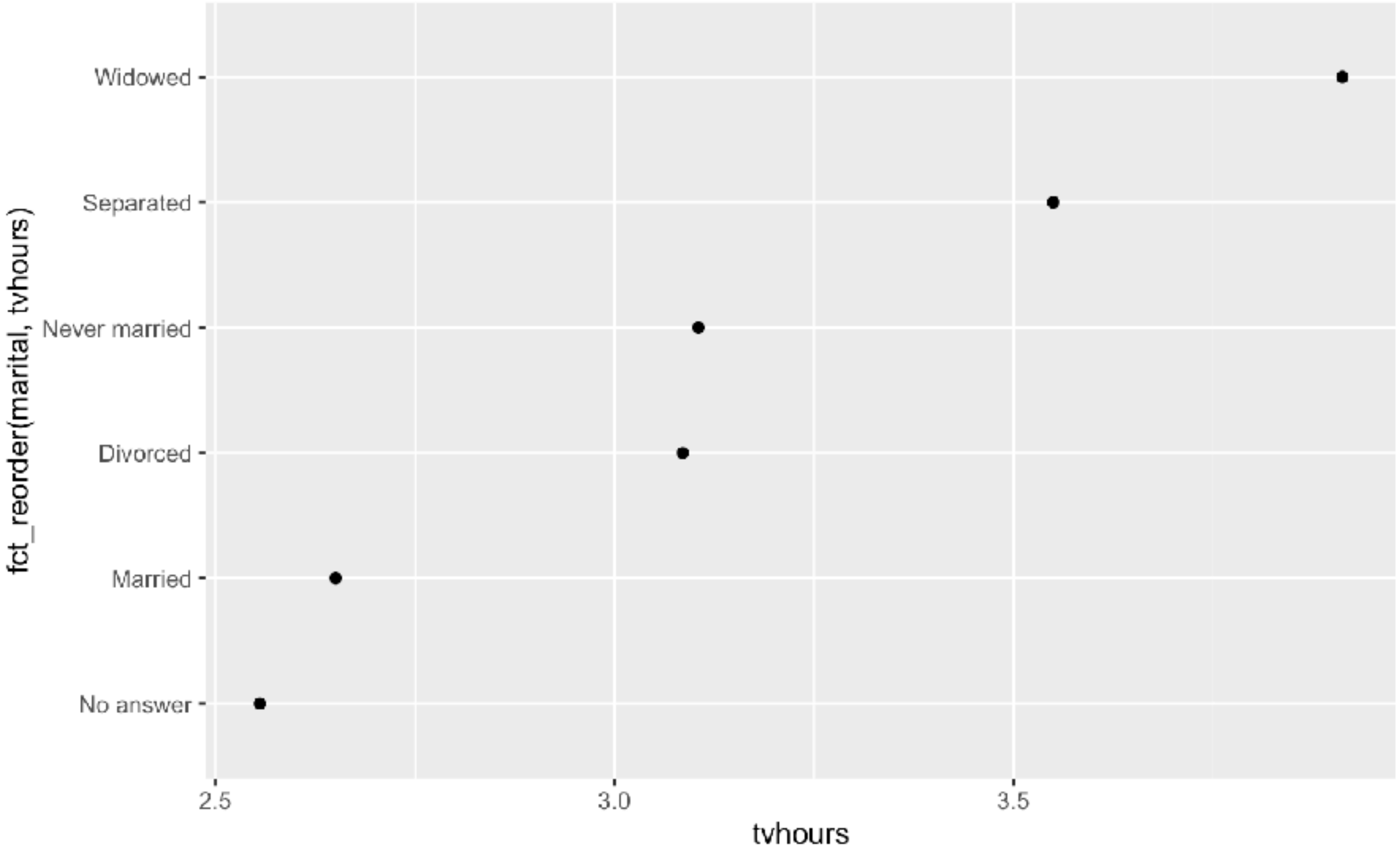


Your Turn 3

Repeat the previous exercise, some of whose code is in your notebook, to make a sensible graph of average TV consumption by marital status.

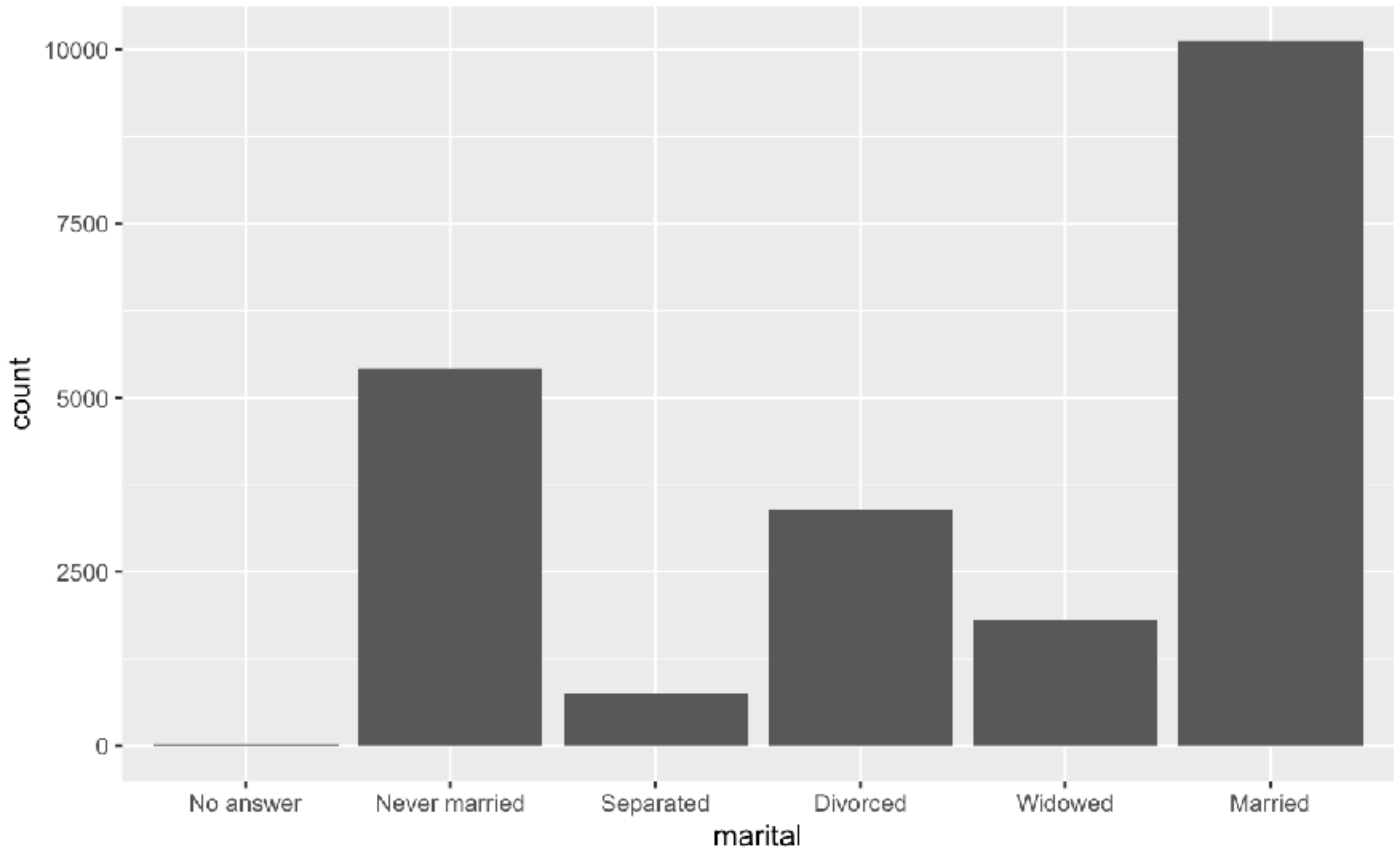
05:00

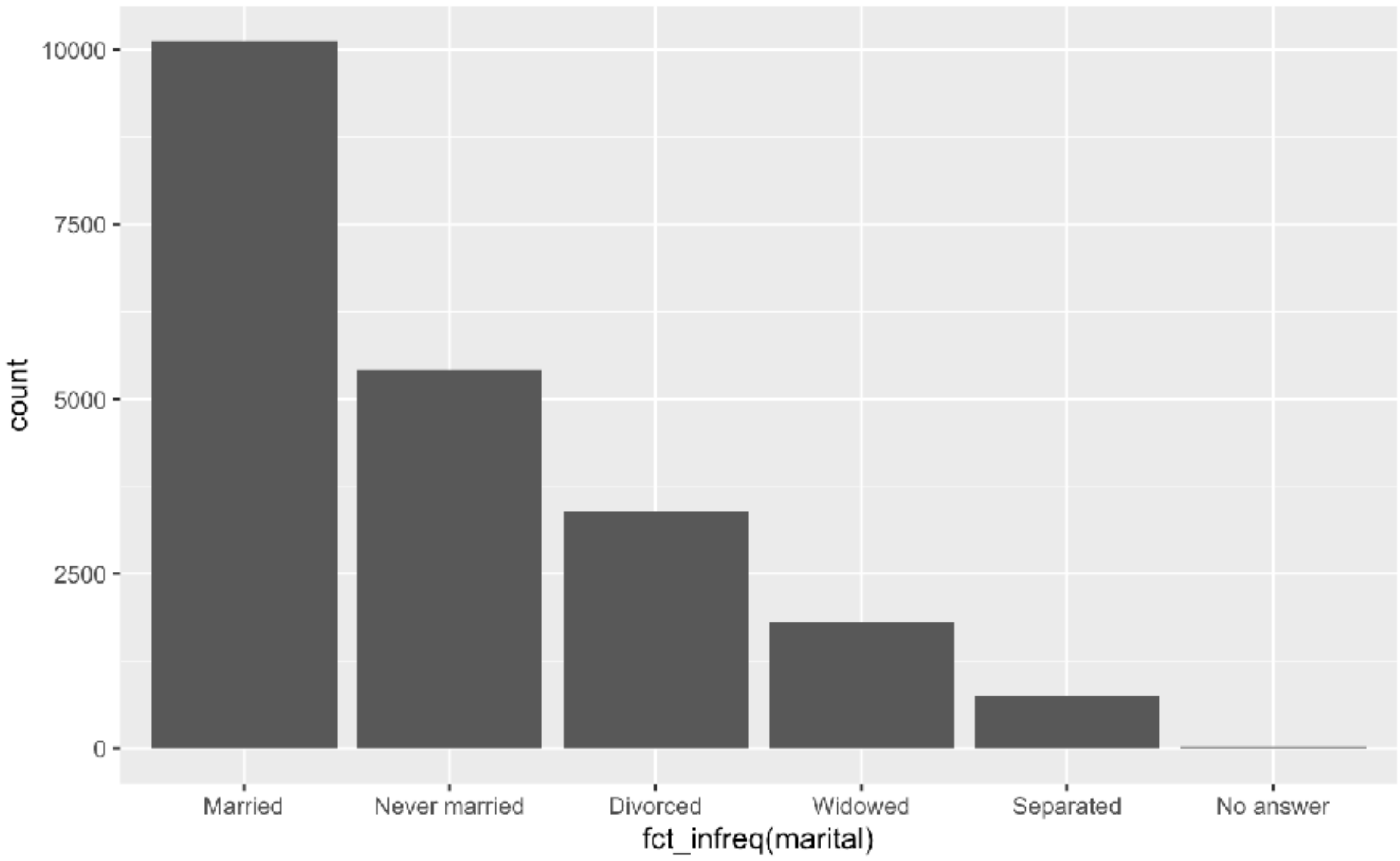
```
gss_cat %>%
  drop_na(tvhours) %>%
  group_by(marital) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(mapping = aes(x = tvhours,y = fct_reorder(marital, tvhours, mean))) +
    geom_point()
```



fct_infreq()

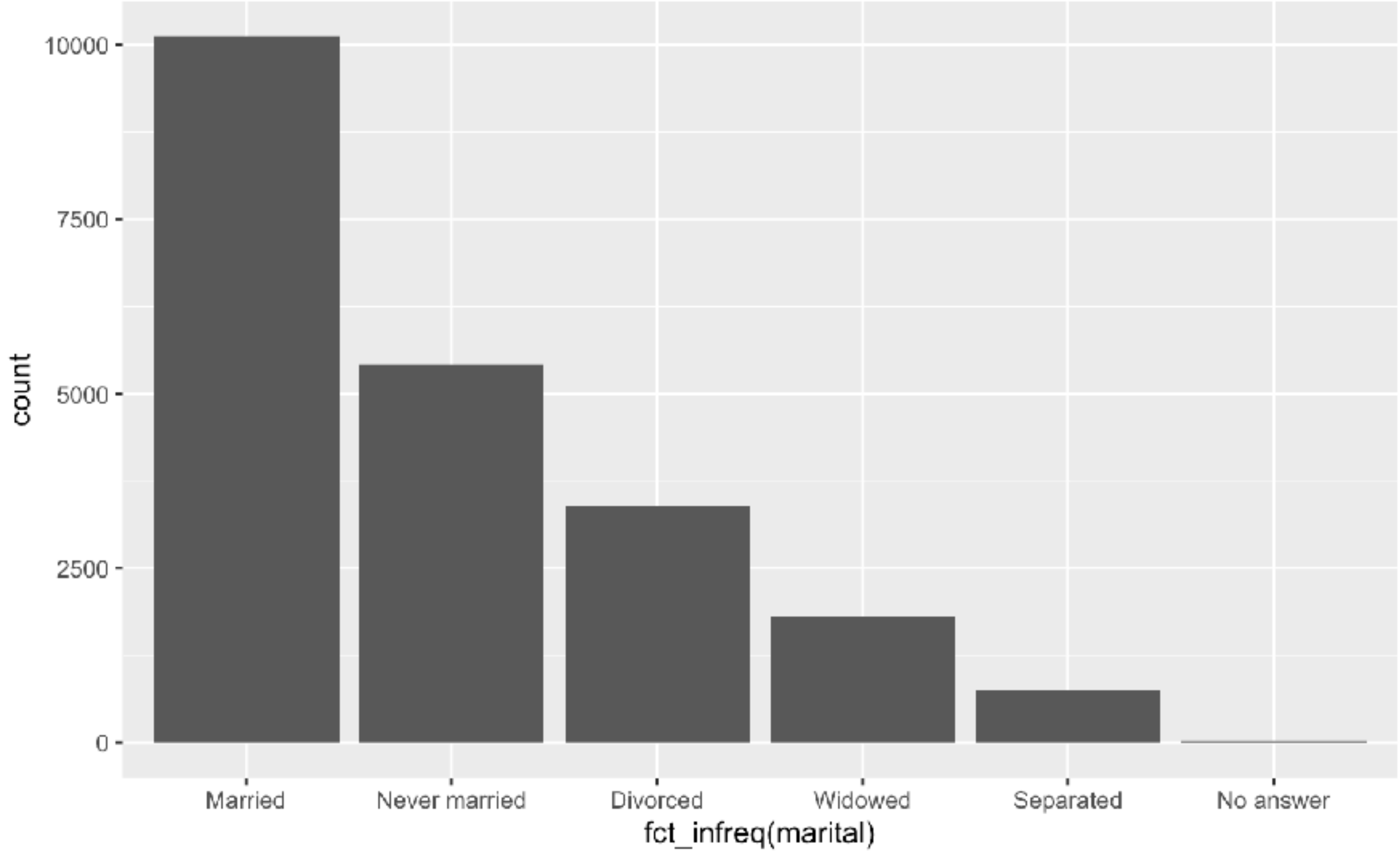
```
gss_cat %>%
  ggplot(aes(fct_infreq(marital)) + geom_bar()
```

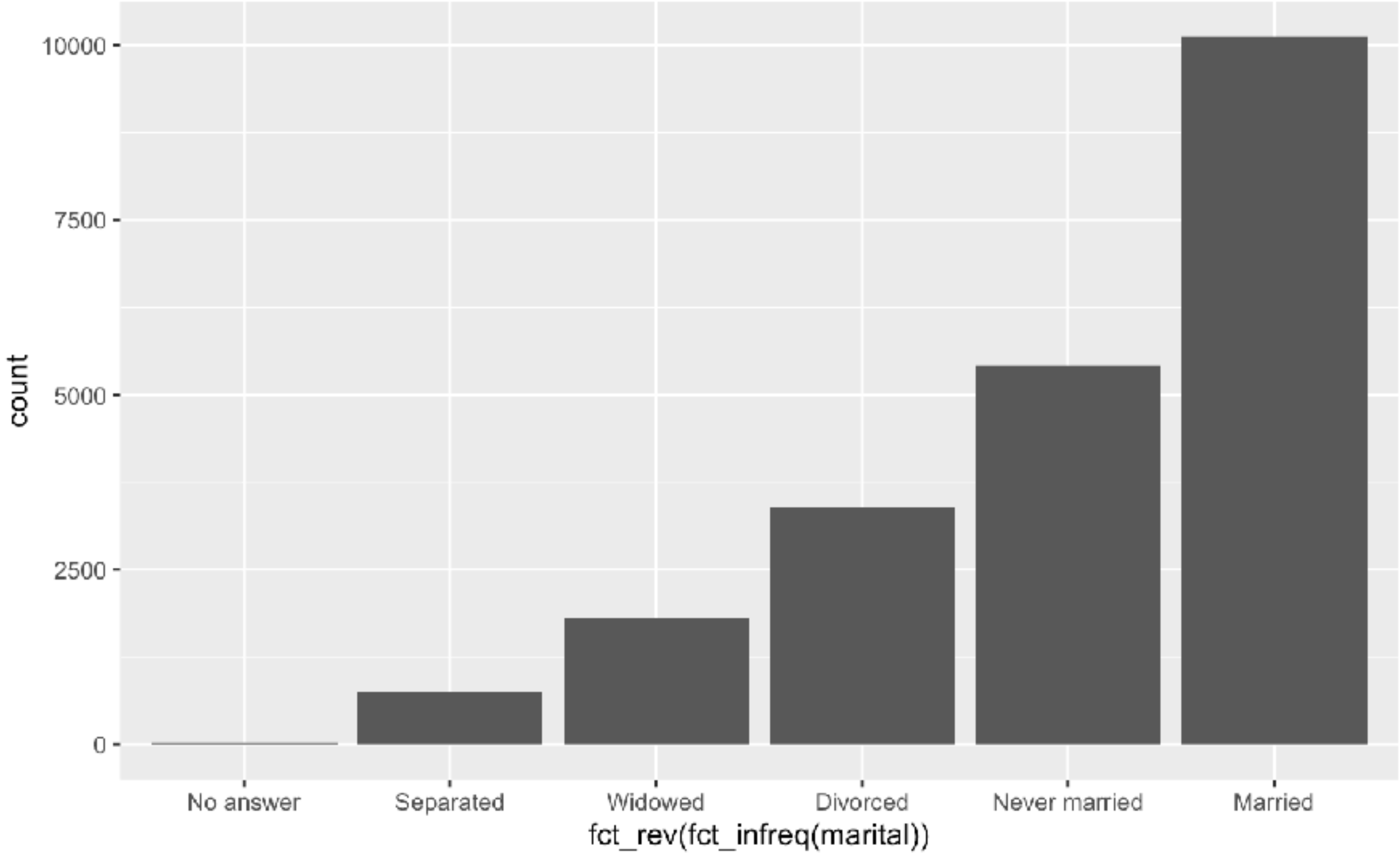




fct_rev()

```
gss_cat %>%
  ggplot(aes(fct_rev(fct_infreq(marital)))) + geom_bar()
```





ATLAS

Changing level values

Your Turn 4

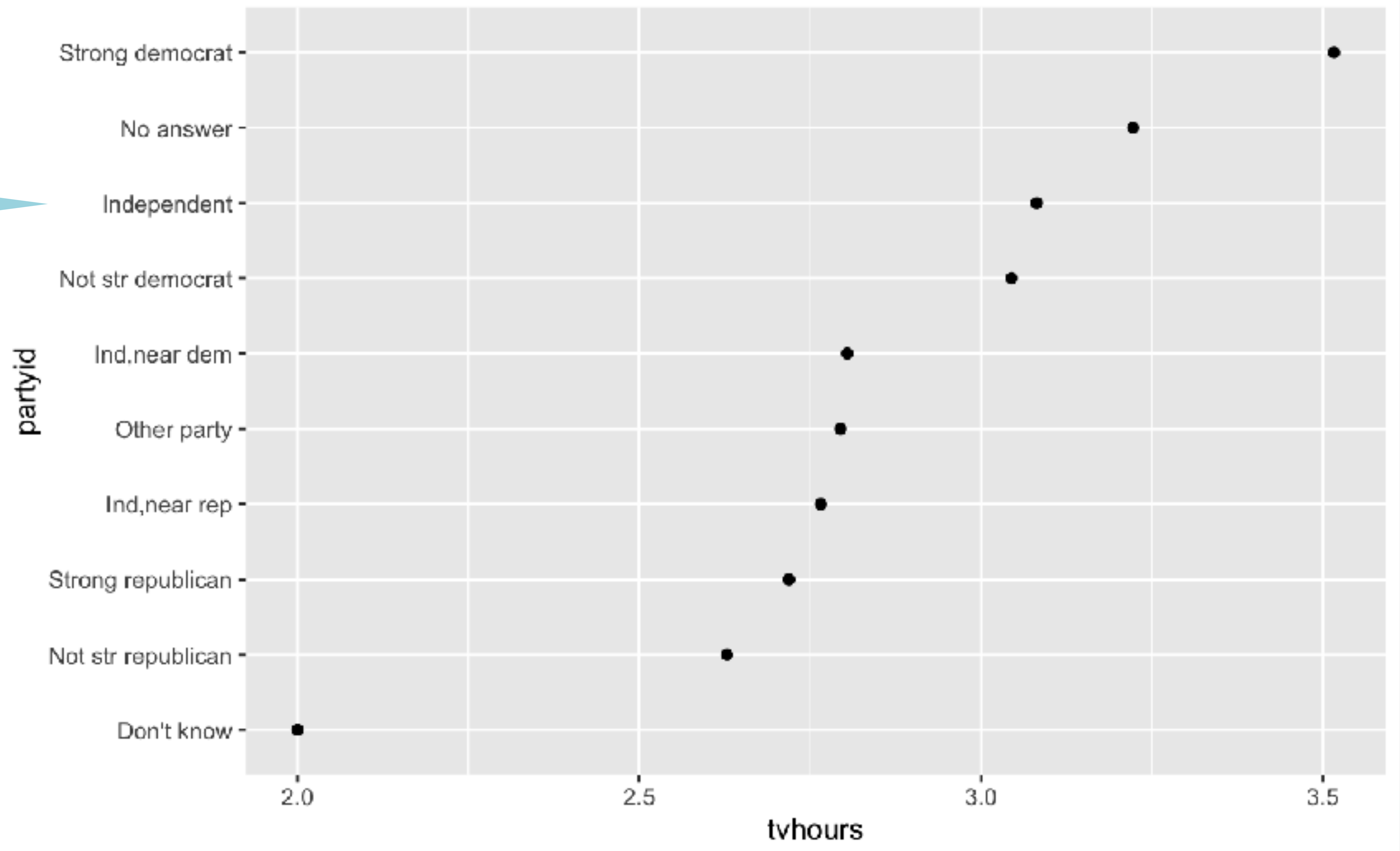
Do you think liberals or conservatives watch more TV?

Compute the average TV hours by party ID and then plot the results.

04 : 00

```r
gss_cat %>%
  drop_na(tvhours) %>%
  group_by(partyid) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(mapping = aes(x = tvhours,y = fct_reorder(marital, tvhours, mean))) +
    geom_point() +
    labs(y = "partyid")
```

1. How can we improve these labels?



fct_recode()

Change values of levels

```
fct_recode(f, ...)
```

factor with levels

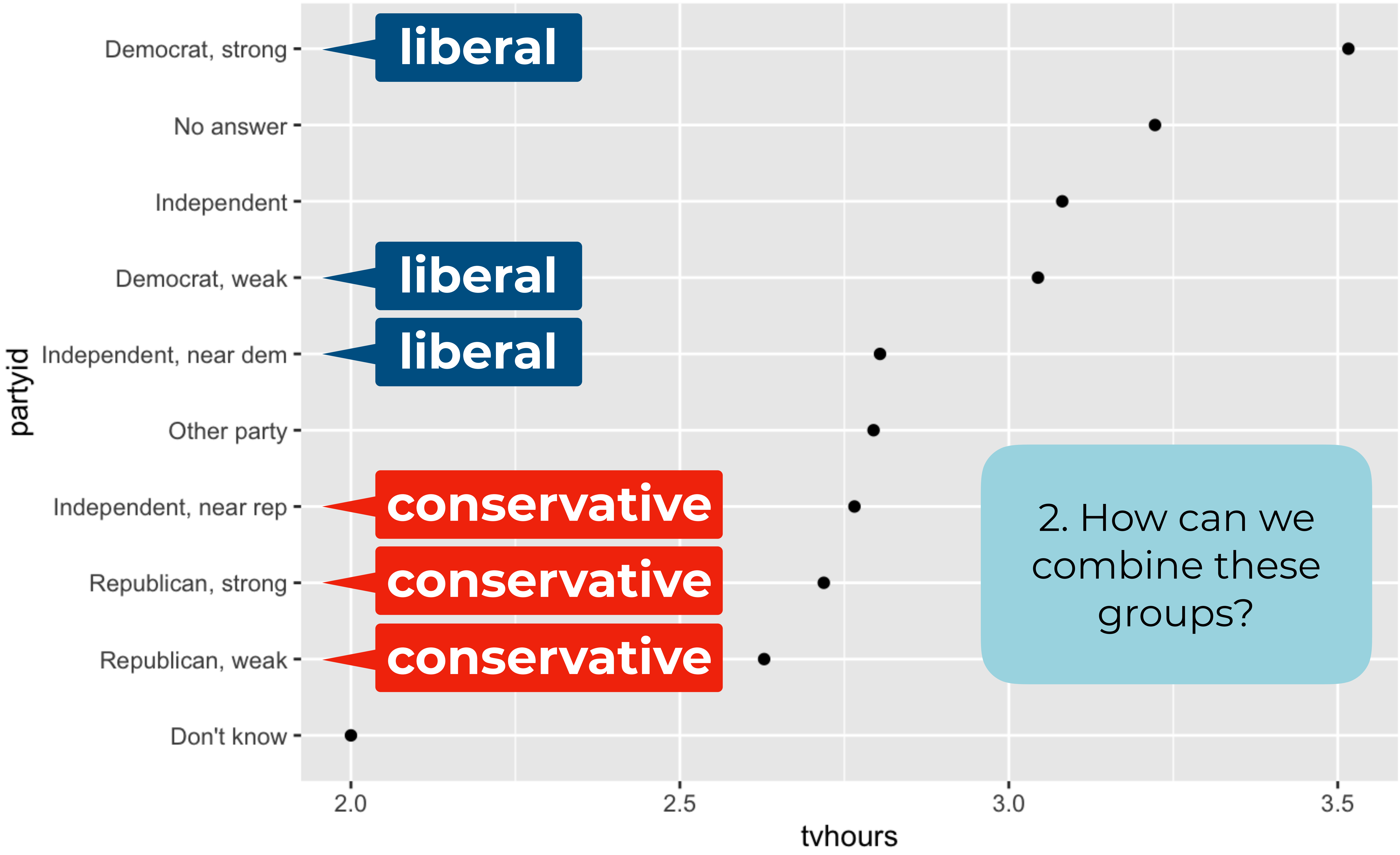
new level = old level pairs (as a named character vector)

ATLAS

```r
gss_cat %>%
  drop_na(tvhours) %>%
  mutate(partyid = fct_recode(partyid,
    "Republican, strong"    = "Strong republican",
    "Republican, weak"      = "Not str republican",
    "Independent, near rep" = "Ind,near rep",
    "Independent, near dem" = "Ind,near dem",
    "Democrat, weak"        = "Not str democrat",
    "Democrat, strong"      = "Strong democrat")) %>%
  group_by(partyid) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(mapping = aes(x = tvhours,y = fct_reorder(partyid, tvhours, mean))) +
    geom_point() +
    labs(y = "partyid")
```

ATLAS

forcats


Democrat, strong
liberal
No answer
Independent
Democrat, weak
liberal
Independent, near dem
liberal
Other party
Independent, near rep
conservative
Republican, strong
conservative
Republican, weak
conservative
Don't know
partyid
2.0
2.5
3.0
3.5
tvhours
2. How can we combine these groups?

ATLAS

Collapsing levels

fct_collapse()

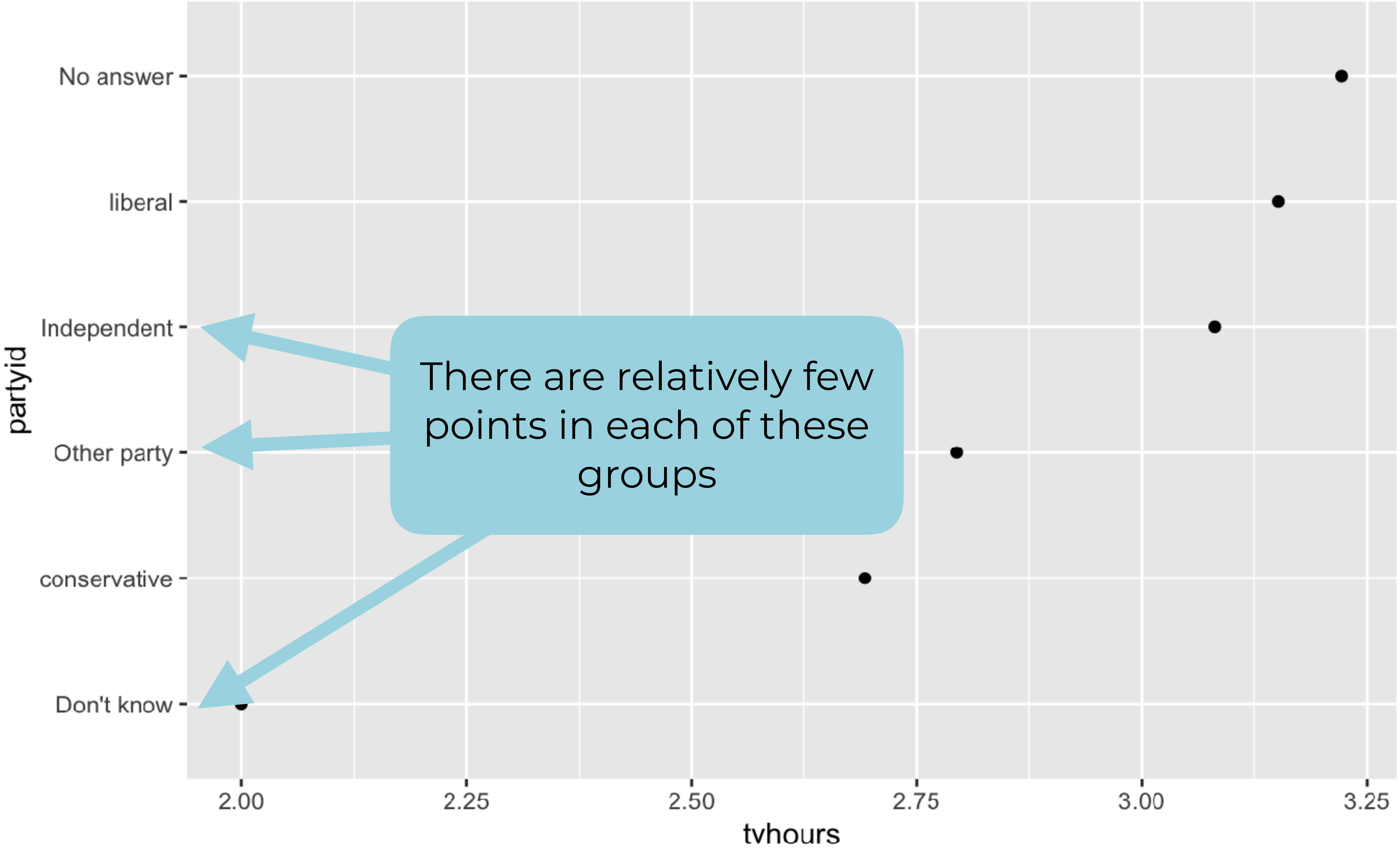
Changes multiple levels into single level

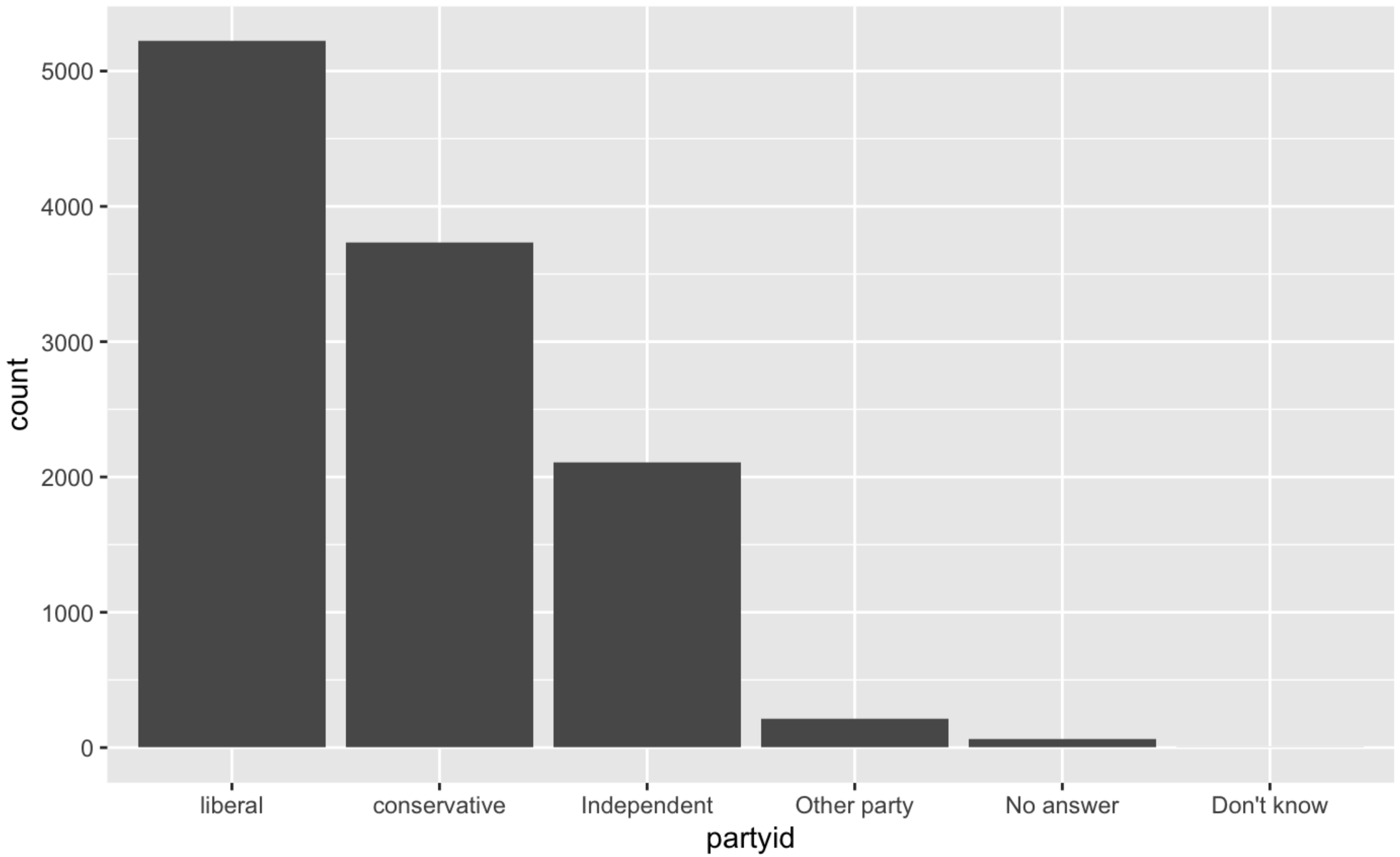
```
fct_collapse(f, ...)
```

factor with levels

named arguments set to a character vector (levels in the vector will be collapsed to the name of the argument)

```r
gss_cat %>%
  drop_na(tvhours) %>%
  mutate(partyid = fct_collapse(partyid,
    conservative = c("Strong republican",
                     "Not str republican",
                     "Ind,near rep"),
    liberal = c("Strong democrat",
                "Not str democrat",
                "Ind,near dem"))) %>%
  group_by(partyid) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(mapping = aes(x = tvhours,y = fct_reorder(partyid, tvhours, mean))) +
    geom_point() +
    labs(y = "partyid")
```


No answer
liberal
Independent
Other party
conservative
Don't know
partyid
2.00
2.25
2.50
2.75
3.00
3.25
tvhours
There are relatively few points in each of these groups


5000
4000
3000
2000
1000
0
count
liberal
conservative
Independent
Other party
No answer
Don't know
partyid

fct_lump()

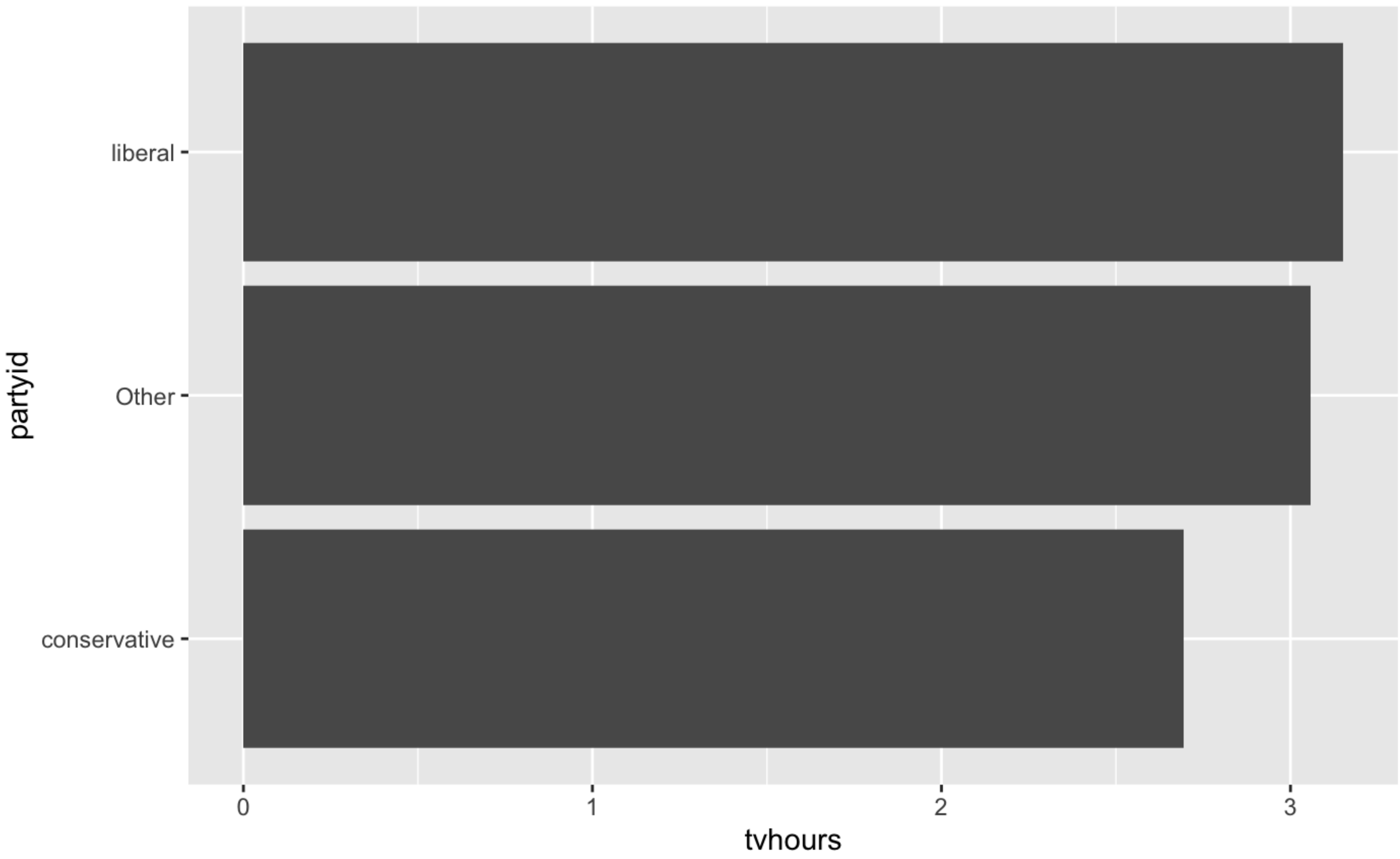
Collapses levels with fewest values into a single level. By default collapses as many levels as possible such that the new level is still the smallest.

```
fct_lump(f, other_level = "Other", ...)
```

factor with levels

name of new level

```r
gss_cat %>%
  drop_na(tvhours) %>%
  mutate(partyid = partyid %>%
    fct_collapse(
      conservative = c("Strong republican", "Not str republican",
                       "Ind,near rep"),
      liberal = c("Strong democrat", "Not str democrat", "Ind,near dem")) %>%
    fct_lump()
  ) %>%
  group_by(partyid) %>%
  summarize(tvhours = mean(tvhours)) %>%
  ggplot(aes(y = tvhours, x = fct_reorder(partyid, tvhours, mean))) +
   geom_col() + labs(y = "partyid") + coord_flip()
```



ATLAS

Date times

Quiz

Does every year have 365 days?

Does every day have 24 hours?

Does every minute have 60 seconds?

What does a month measure?

Most useful skills

1. Creating dates/times (i.e., *parsing*)
2. Access and change parts of a date
3. Deal with time zones
4. Do math with instants and time spans

ATLAS

Warm Up

Decide in your group:

- What is the best time of day to fly?
- What is the best day of the week to fly?

01:00

ATLAS

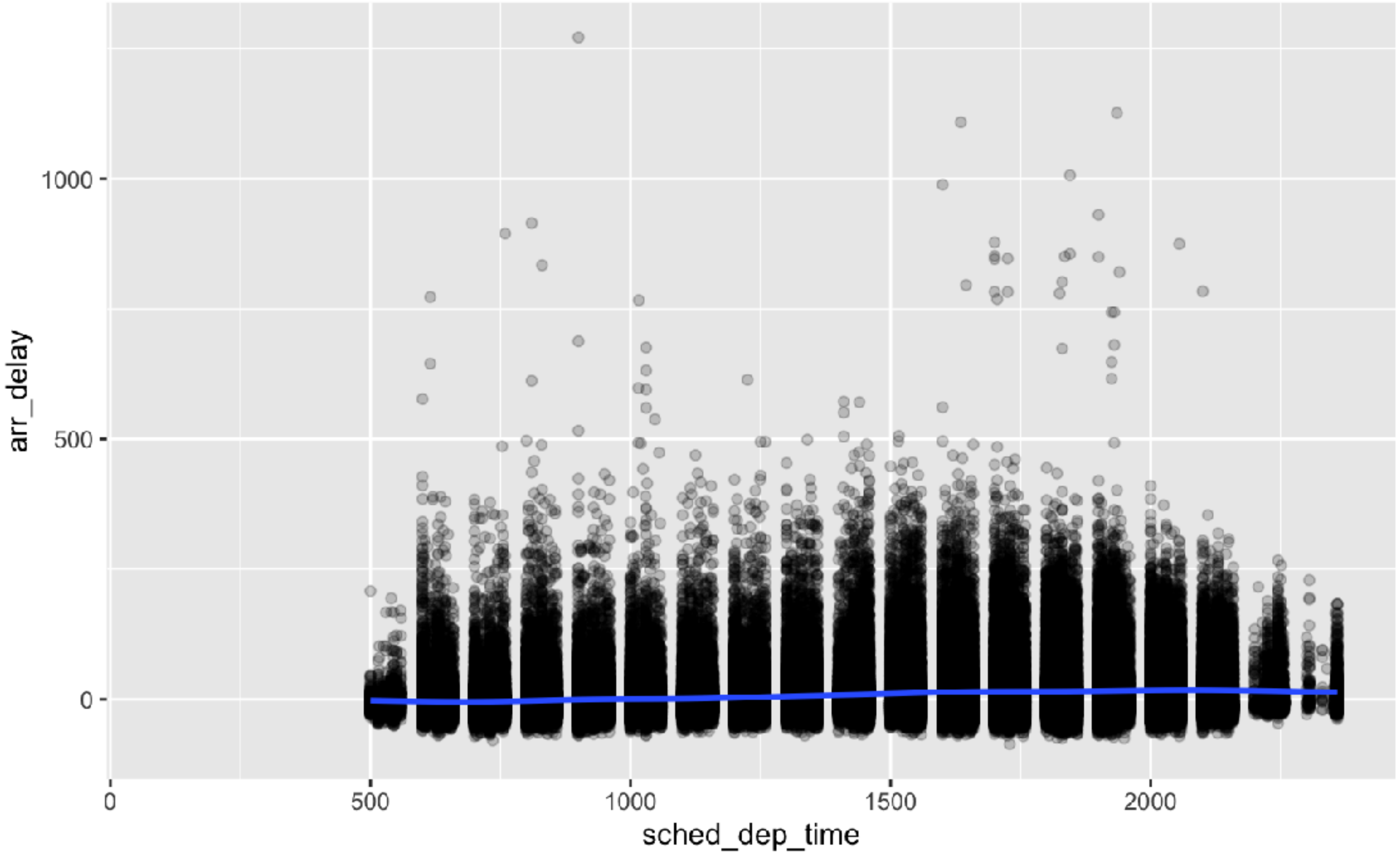
```r
flights %>% select(year, month, day, hour, minute, sched_dep_time, time_hour)
```

year	month	day	hour	minute	sched_dep_time	time_hour
<int>	<int>	<int>	<dbl>	<dbl>	<int>	<S3: POSIXct>
2013	1	1	5	15	515	2013-01-01 05:00:00
2013	1	1	5	29	529	2013-01-01 05:00:00
2013	1	1	5	40	540	2013-01-01 05:00:00
2013	1	1	5	45	545	2013-01-01 05:00:00
2013	1	1	6	0	600	2013-01-01 06:00:00
2013	1	1	5	58	558	2013-01-01 05:00:00
2013	1	1	6	0	600	2013-01-01 06:00:00
2013	1	1	6	0	600	2013-01-01 06:00:00
2013	1	1	6	0	600	2013-01-01 06:00:00
2013	1	1	6	0	600	2013-01-01 06:00:00

1–10 of 336,776 rows

Previous 1 2 3 4 5 6 ... 100 Next

```
flights %>%
  ggplot(mapping = aes(x = sched_dep_time, y = arr_delay)) +
    geom_point(alpha = 0.2) + geom_smooth()
```



Creating dates and times

In R4DS

Dates and times

hms

2019-01-01 12:34:56

Stored as the number of seconds since 00:00:00.*

```
library(hms)
hms(seconds = 56, min = 34, hour = 12)
# 12:34:56

unclass(hms(56, 34, 12))
# 45296
```

* on a typical day

hms()

2019-01-01 12:34:56

```
library(hms)
hms(seconds, minutes, hours, days)
```

Your Turn 5

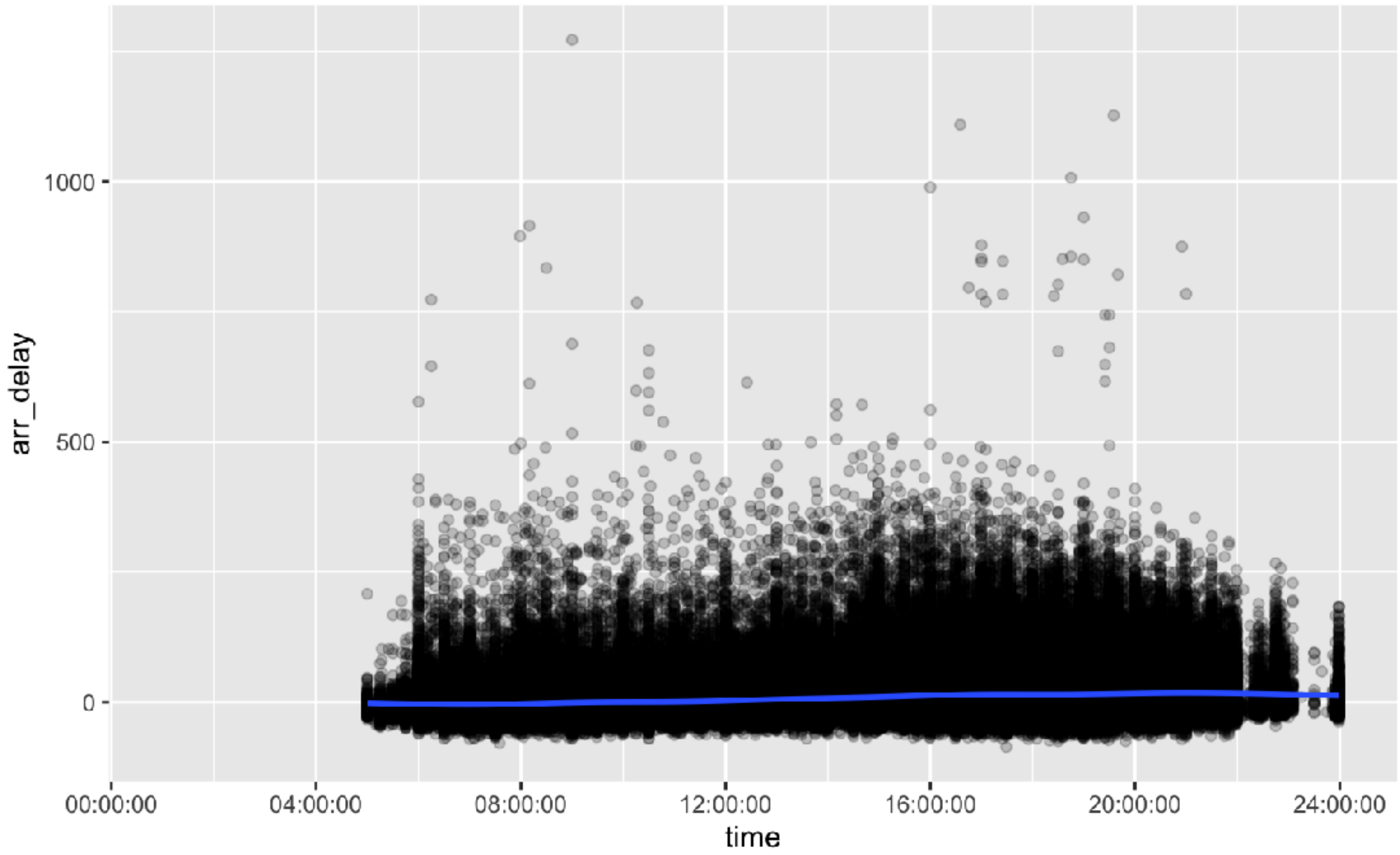
What is the best time of day to fly?

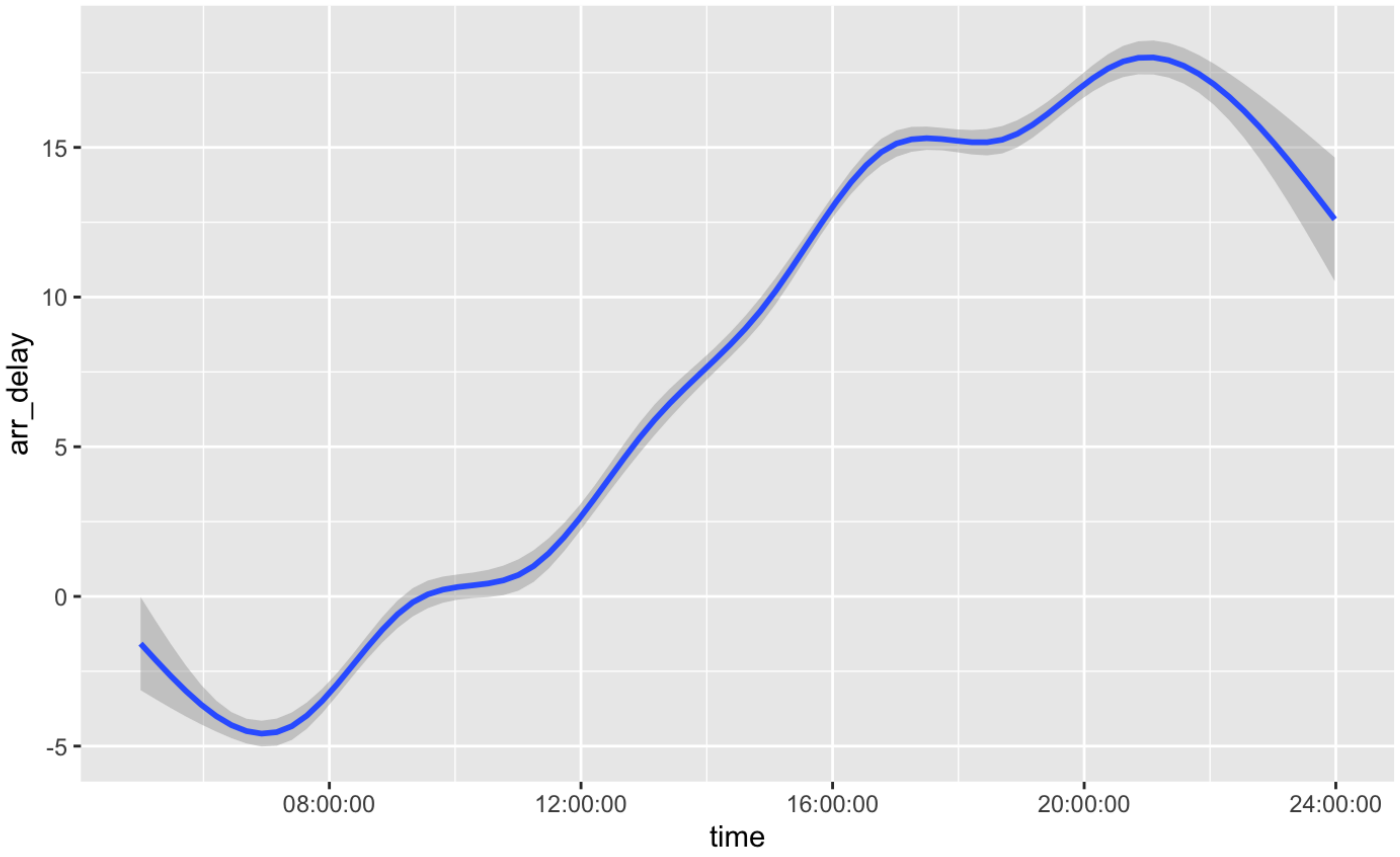
Use the **hour** and **minute** variables in flights to compute the time of day for each flight as an hms. Then use a smooth line to plot the relationship between time of day and **arr_delay**.

04 : 00

ATLAS

```
flights %>%
  mutate(time = hms(hour = hour, minute = minute)) %>%
  ggplot(mapping = aes(x = time, y = arr_delay)) +
    geom_point(alpha = 0.2) + geom_smooth()
```





ATLAS

hms

lubridate

In R4DS

Dates and times

ymd() family

To parse strings as dates, use a *y*, *m*, *d*, *h*, *m*, *s* combination

```
ymd("2019/01/09")
#  [1] "2019-01-09"

mdy("January 9, 2019")
#  [1] "2019-01-09"

ymd_hms("2019-01-09 13:42:32")
#  [1] "2019-01-09 13:42:32 UTC"
```

ATLAS

lubridate

Parsing functions

function	parses to

`ymd_hms()`, `ymd_hm()`, `ymd_h()`
`ydm_hms()`, `ydm_hm()`, `ydm_h()`
`dmy_hms()`, `dmy_hm()`, `dmy_h()`
`mdy_hms()`, `mdy_hm()`, `mdy_h()`
	POSIXct

`ymd()`, `ydm()`, `mdy()` `myd()`, `dmy()`, `dym()`, `yq()`	Date (POSIXct if `tz` specified)
`hms()`, `hm()`, `ms()`	Period

Accessing and changing components

Accessing components

Extract components by name with a **singular** name

```
date <- ymd("2019-01-09")
year(date)
# 2019
```

Setting components

Use the same function to set components

```
date
# "2019-01-09"

year(date) <- 1999
date
# "1999-01-09"
```

ATLAS

lubridate

Accessing date time components

function	extracts	extra arguments
`year()`	year	
`month()`	month	`label = FALSE, abbr = TRUE`
`week()`	week	
`day()`	day of month	
`wday()`	day week	`label = FALSE, abbr = TRUE`
`qday()`	day of quarter	
`yday()`	day of year	
`hour()`	hour	
`minute()`	minute	
`second()`	second	

Accessing components

```
wday(ymd("2019-01-09"))
# [1] 4

wday(ymd("2019-01-09"), label = TRUE)
# [1] Wed
# 7 Levels: Sun < Mon < Tues < Wed < Thurs < ... < Sat

wday(ymd("2019-01-09"), label = TRUE, abbr = FALSE)
# [1] Wednesday
# 7 Levels: Sunday < Monday < Tuesday < ... < Saturday
```

Your Turn 6

Fill in the blanks to:

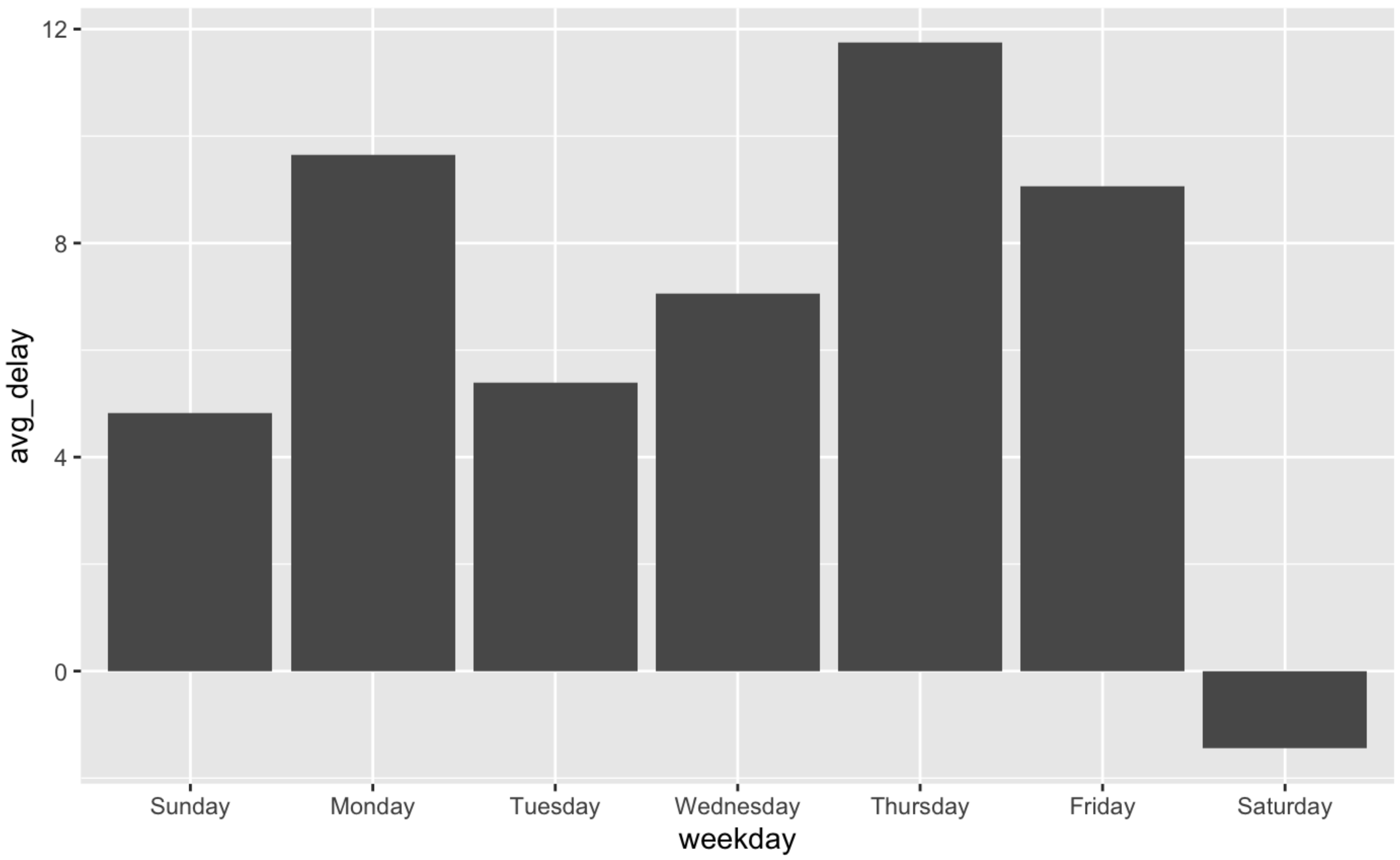
1. Extract the day of the week of each flight (as a full name) from **time_hour**.
2. Calculate the average **arr_delay** by day of the week.
3. Plot the results as a column chart (bar chart) with **geom_col()**.

05 : 00

ATLAS

lubridate

```
flights %>%
  mutate(weekday = wday(time_hour, label = TRUE, abbr = FALSE)) %>%
  group_by(weekday) %>%
  drop_na(arr_delay) %>%
  summarise(avg_delay = mean(arr_delay)) %>%
  ggplot() +
    geom_col(mapping = aes(x = weekday, y = avg_delay))
```


12
8
4
0
avg_delay
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
weekday

Data Types

stringr
forcats
lubridate
hms

ATLAS
Accessible Teaching, Learning, and Assessment Systems

wjakethompson.com wjakethompson@ku.edu @wjakethompson @wjakethompson