Week 9: Factors & Pull Requests

Miscellany

Joe Nese

University of Oregon

Fall 2021

Factors & Pull Requests

Week 9

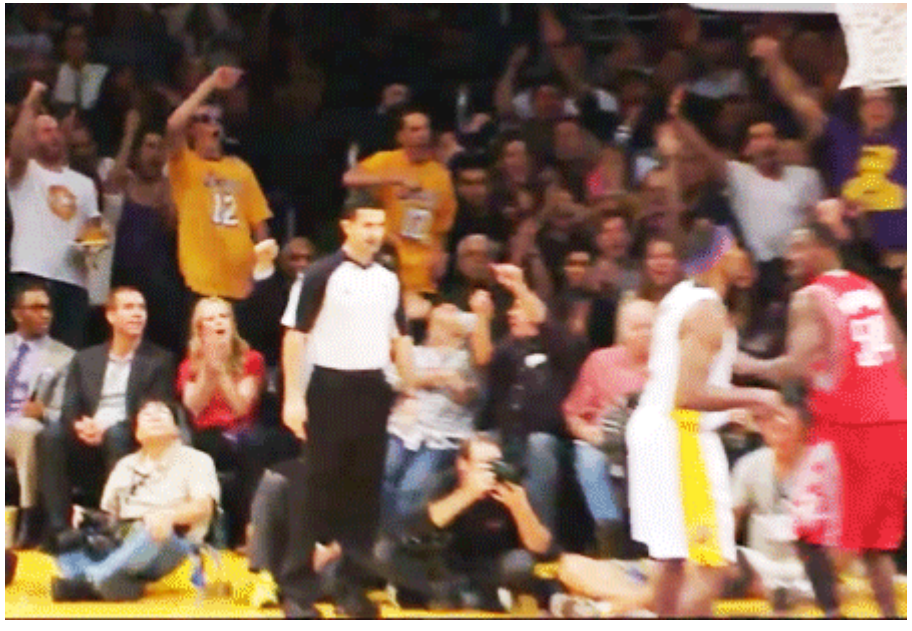
Agenda

- Shares!
- Citation additions
- Final Project Review
- Discuss factors and factor re-leveling

Overall Purpose

- Understand factors and how to manipulate them
- Understand how to complete a *pull request (PR)*

Lab 8



Shares!

Amy, Errol, Havi

Citation Styles

(cheat sheet)

Citation Styles

Citation Style (using the tag)	Output
@Briggs11	Briggs and Weeks (2011)
[see @Baldwin2014; @Caruso2000]	(see Baldwin et al. 2014; Caruso 2000)
[@Linn02, p. 9]	(Linn and Haug 2002, p. 9)
[-@Goldhaber08]	(2008)

I forgot to share this slide earlier!

Also, cite R!

I also forgot to share this this with you! (*be kind in your teacher evals*)

```
citation()
```

```
##
## To cite R in publications use:
##
##   R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statist
##   Austria. URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2021},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it when using it for data analysis
```

Final Project

Final Project - Data Prep Script

- Expected to be a work in progress
- Provided to your peers so they can learn from you as much as you can learn from their feedback

Peer Review

- Understand the purpose of the exercise
- Conducted as a professional product
- Should be **very** encouraging
- Zero tolerance policy for inappropriate comments

Final Project – Presentation

Groups are expected to present for about **18-20 minutes** (split evenly among members). Group order randomly assigned.

Email me your presentation by midnight 11/30 so I can share through my machine.

Final Project – Presentation

Presentation cover the following:

- Share your journey (everyone, at least for a minute or two)
- Discuss challenges you had along the way
- Celebrate your successes
- Discuss challenges you are still facing
- Discuss substantive findings
- Show off your cool figures!
- Discuss next R hurdle you want to address

Final Project – Paper

- R Markdown document
  - Abstract, Intro, Methods, Results, Discussion, References
  - Should be brief: 3,500 words max
- No code displayed - should look similar to a manuscript being submitted for publication
- Include at least 1 table
- Include at least 2 plots
- Should be fully open, reproducible, and housed on GitHub
  - I should be able to clone your repository, open the R Studio Project, and reproduce the full manuscript (by knitting the R Markdown doc)

Final Project

The following functions:

- `pivot_longer()`
- `mutate()`
- `select()`
- `filter()`
- `pivot_wider()`
- `group_by()`
- `summarize()`

Scoring Rubric

Check the syllabus for Presentation and Final Paper scoring rubrics

Revisiting git

Before we jump in...

...let's revisit git

Talk with neighbor. What do these terms mean? Talk about them in the order you would encounter them in your workflow

- *clone*
- *pull*
- *stage*
- *commit*
- *push*
- *repo*
- *remote*

03:00

Factors

just the basics

When do we really want factors?

Generally two reasons to declare a factor

1. Only finite number of categories
    - treatment/control
    - income categories
    - performance levels
    - etc.
2. Use in modeling

Creating factos

Imagine you have a vector of months

```
months_4 <- c("Dec", "Apr", "Jan", "Mar")
```

We could store this as a string, but there are issues with this:

- There are only 12 possible months
  - factors will help us weed out values that don't conform to our predefined levels, which helps safeguard against typos, etc.
- You can't sort this vector in a meaningful way
  - default is alphabetic sorting

```
sort(months_4)
```

```
## [1] "Apr" "Dec" "Jan" "Mar"
```

Define it as a factor

```
months_4 <- factor(months_4, levels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep
months_4
```

```
## [1] Dec Apr Jan Mar
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

Now we can sort

```
sort(months_4)
```

```
## [1] Jan Mar Apr Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

Accessing and modifying levels

Uset the `levels()` function

```
levels(months_4)
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

Provides an error check of sorts

```r
months_4[5] <- "Jam"
```

```
## Warning in `[<-.factor`(`*tmp*`, 5, value = "Jam"): invalid factor level, NA generated
```

```r
months_4
```

```
## [1] Dec  Apr  Jan  Mar  <NA>
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

What if we don't specify levels?

If you define a factor without specifying the levels, it will assign them alphabetically

```
mnths <- factor(c("Dec", "Apr", "Jan", "Mar"))
```

```
mnths
```

```
## [1] Dec Apr Jan Mar
## Levels: Apr Dec Jan Mar
```

{forcats}

- When working with factors, we can use the `{forcats}` package
  - `for cat`egorical variables
  - anagram of factors
- Part of the `{tidyverse}` so should be good to go
- All functions start with `fct_`
  - use the autofill in RStudio

Change level order – `fct_inorder()`

In order they are entered

```
(mnths <- factor(c("Dec", "Apr", "Jan", "Mar")))
```

```
## [1] Dec Apr Jan Mar
## Levels: Apr Dec Jan Mar
```

```
mnths %>%
  factor(., levels = c("Jan", "Mar", "Apr", "Dec")) %>%
  sort(.)
```

```
## [1] Jan Mar Apr Dec
## Levels: Jan Mar Apr Dec
```

Change level order – `fct_inorder()`

In order they are entered

```
(mnths <- factor(c("Dec", "Apr", "Jan", "Mar")))
```

```
## [1] Dec Apr Jan Mar
## Levels: Apr Dec Jan Mar
```

```
mnths %>%
  factor(., levels = c("Jan", "Mar", "Apr", "Dec")) %>%
  fct_inorder() %>%
  sort(.)
```

```
## [1] Dec Apr Jan Mar
## Levels: Dec Apr Jan Mar
```

Change level order – `fct_infreq()`

In order of frequency

```
c("b", "b", "c", "a", "a", "a") %>%
    fct_infreq()
```

```
## [1] b b c a a a
## Levels: a b c
```

This can be **especially** useful for plotting

```
ggplot(aes(x, fct_infreq(y))
```

Investigate factors

- `{tidyverse}` gives you convenient way to evaluate factors
  - `count()`
  - `geom_bar()` or `geom_col()`) with `{ggplot2}`
- But don't forget about the base function `unique()`
  - e.g., `unique(df$factor_variable)`

General Social Survey (GSS)

```
forcats::gss_cat
```

```
## # A tibble: 21,483 x 9
##     year marital         age race  rincome        partyid            relig              denom
##    <int> <fct>         <int> <fct> <fct>          <fct>              <fct>              <fct>
##  1  2000 Never married    26 White $8000 to 9999  Ind,near rep       Protestant         Southern baptis
##  2  2000 Divorced         48 White $8000 to 9999  Not str republican Protestant         Baptist-dk whic
##  3  2000 Widowed          67 White Not applicable Independent        Protestant         No denomination
##  4  2000 Never married    39 White Not applicable Ind,near rep       Orthodox-christian Not applicable
##  5  2000 Divorced         25 White Not applicable Not str democrat   None               Not applicable
##  6  2000 Married          25 White $20000 - 24999 Strong democrat    Protestant         Southern baptis
##  7  2000 Never married    36 White $25000 or more Not str republican Christian          Not applicable
##  8  2000 Divorced         44 White $7000 to 7999  Ind,near dem       Protestant         Lutheran-mo syn
##  9  2000 Married          44 White $25000 or more Not str democrat   Protestant         Other
## 10  2000 Married          47 White $25000 or more Strong republican  Protestant         Southern baptis
## # ... with 21,473 more rows
```

```r
gss_cat %>%
    count(partyid)
```

```
## # A tibble: 10 x 2
##    partyid               n
##    <fct>             <int>
##  1 No answer           154
##  2 Don't know            1
##  3 Other party         393
##  4 Strong republican  2314
##  5 Not str republican 3032
##  6 Ind,near rep       1791
##  7 Independent        4119
##  8 Ind,near dem       2499
##  9 Not str democrat   3690
## 10 Strong democrat    3490
```

```r
levels(gss_cat$partyid)
```

```
##  [1] "No answer"          "Don't know"         "Other party"        "Strong republican"  "Not str repub
##  [7] "Independent"        "Ind,near dem"       "Not str democrat"   "Strong democrat"
```

```r
unique(gss_cat$partyid)
```

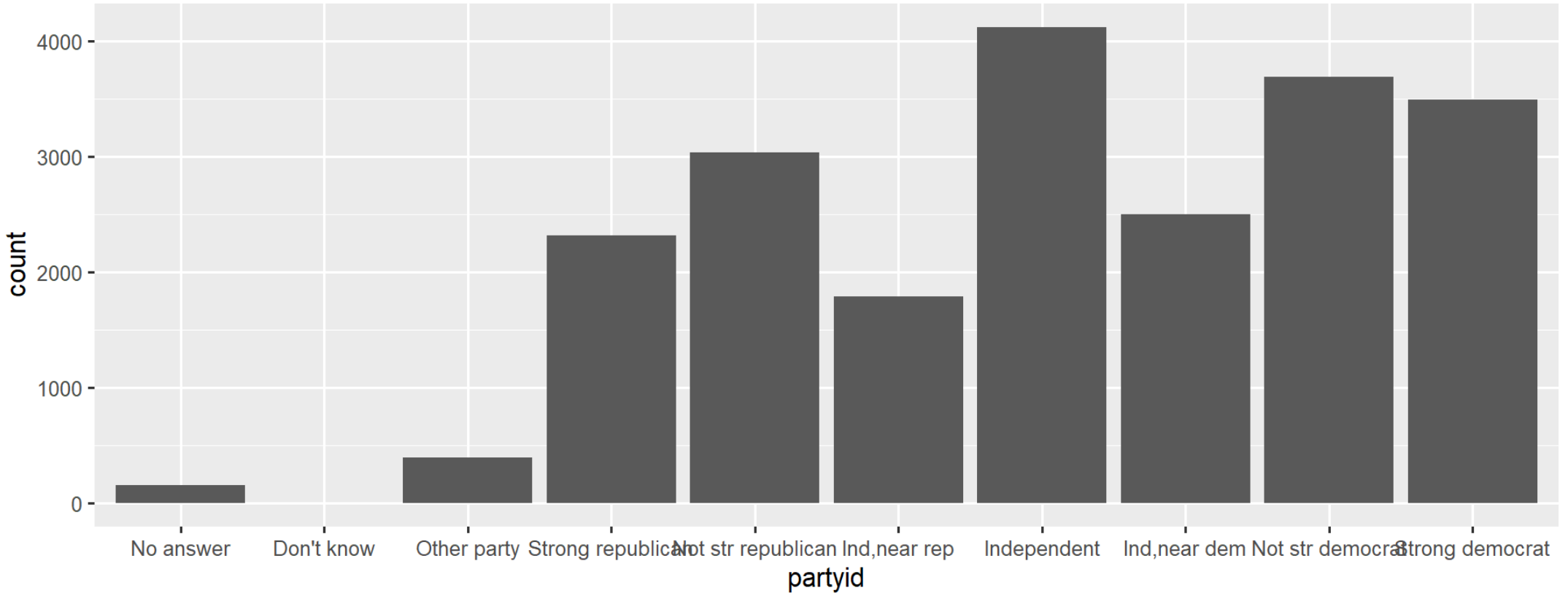
```
##  [1] Ind,near rep       Not str republican Independent        Not str democrat   Strong democrat    Ind
##  [8] Other party        No answer          Don't know
## 10 Levels: No answer Don't know Other party Strong republican Not str republican Ind,near rep Independe
```

How many `unique` categories are there (if you have a lot)?

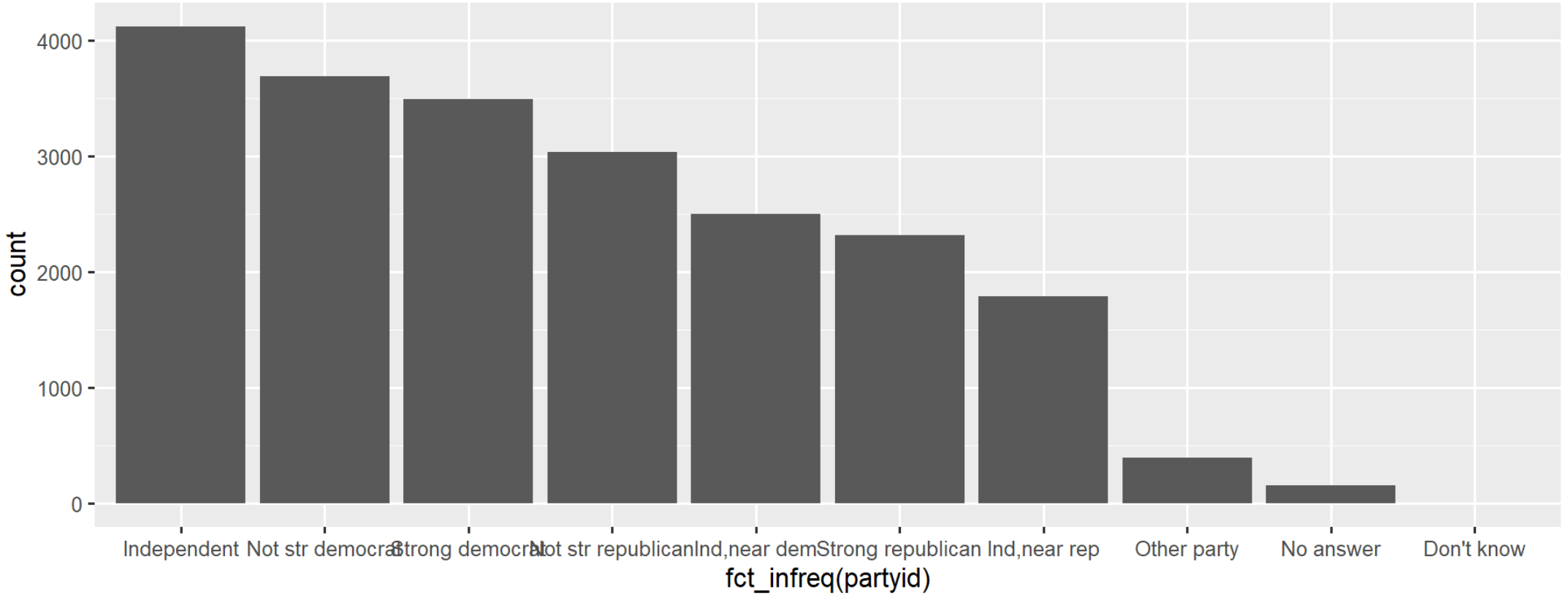
```r
length(unique(gss_cat$partyid))
```

```
## [1] 10
```

```
ggplot(gss_cat, aes(partyid)) +
    geom_bar()
```



```
ggplot(gss_cat, aes(fct_infreq(partyid))) +
    geom_bar()
```



Change level order – `fct_infreq()`

Change level order by hand

- *probably one I use most*

```
fct_relevel(variable_name,
            "first_level",
            "second_level",
            "third_level",
            ...)
```

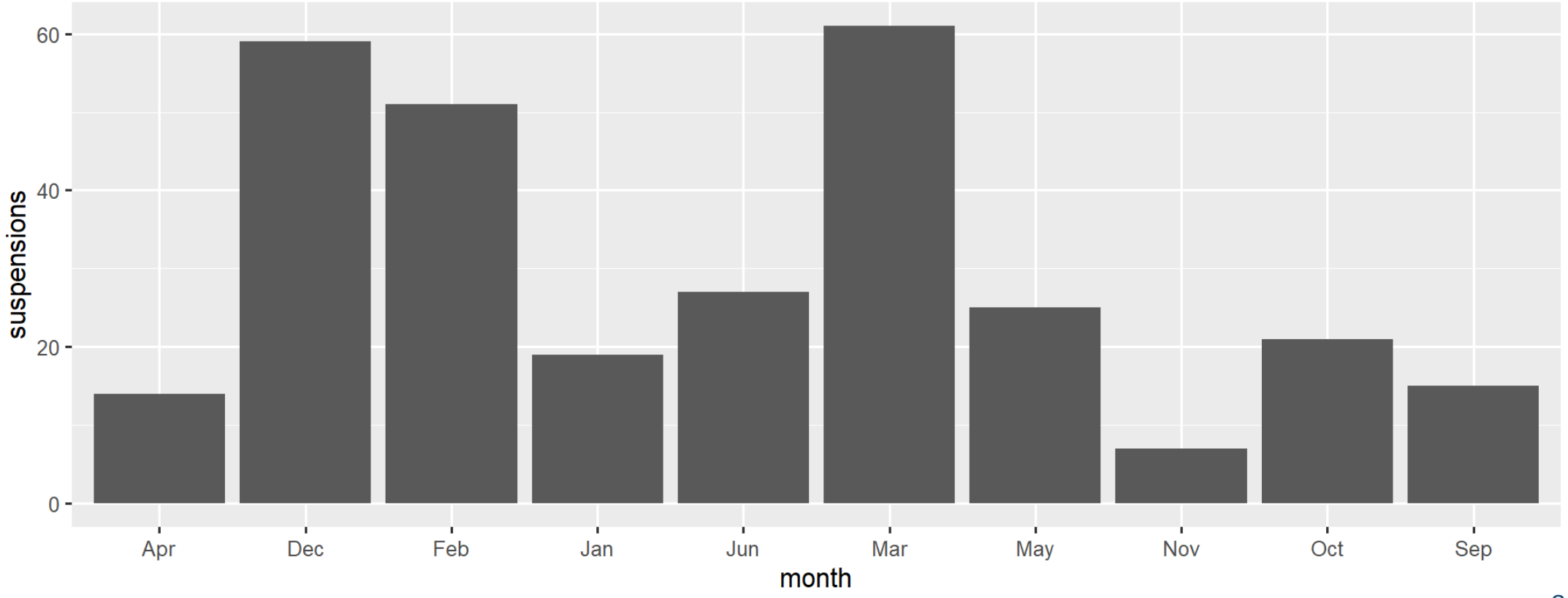
```r
set.seed(3000)
tibble(
 month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Sep", "Oct", "Nov", "Dec"),
 suspensions = sample(c(5:75), size = 10)
)
```

```
## # A tibble: 10 x 2
##    month suspensions
##    <chr>       <int>
##  1 Jan            19
##  2 Feb            51
##  3 Mar            61
##  4 Apr            14
##  5 May            25
##  6 Jun            27
##  7 Sep            15
##  8 Oct            21
##  9 Nov             7
## 10 Dec            59
```

```r
set.seed(3000)
tibble(
 month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Sep", "Oct", "Nov", "Dec"),
suspensions = sample(c(5:75), size = 10)
)
```

```
## # A tibble: 10 x 2
##    month suspensions
##    <chr>       <int>
##  1 Jan            19
##  2 Feb            51
##  3 Mar            61
##  4 Apr            14
##  5 May            25
##  6 Jun            27
##  7 Sep            15
##  8 Oct            21
##  9 Nov             7
## 10 Dec            59
```

```r
set.seed(3000)
tibble(
 month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Sep", "Oct", "Nov", "Dec"),
 suspensions = sample(c(5:75), size = 10)
) %>%
  ggplot(aes(month, suspensions)) +
  geom_col()
```



```r
set.seed(3000)
tibble(
 month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Sep", "Oct", "Nov", "Dec"),
 suspensions = sample(c(5:75), size = 10)
) %>%
  mutate(month = fct_relevel(month,
                             "Sep", "Oct", "Nov", "Dec", "Jan", "Feb", "Mar", "Apr", "May"))
```

```r
set.seed(3000)
tibble(
 month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Sep", "Oct", "Nov", "Dec"),
 suspensions = sample(c(5:75), size = 10)
) %>%
  mutate(month = fct_relevel(month,
                             "Sep", "Oct", "Nov", "Dec", "Jan", "Feb", "Mar", "Apr", "May")) %>%
  ggplot(aes(month, suspensions)) +
  geom_col()
```

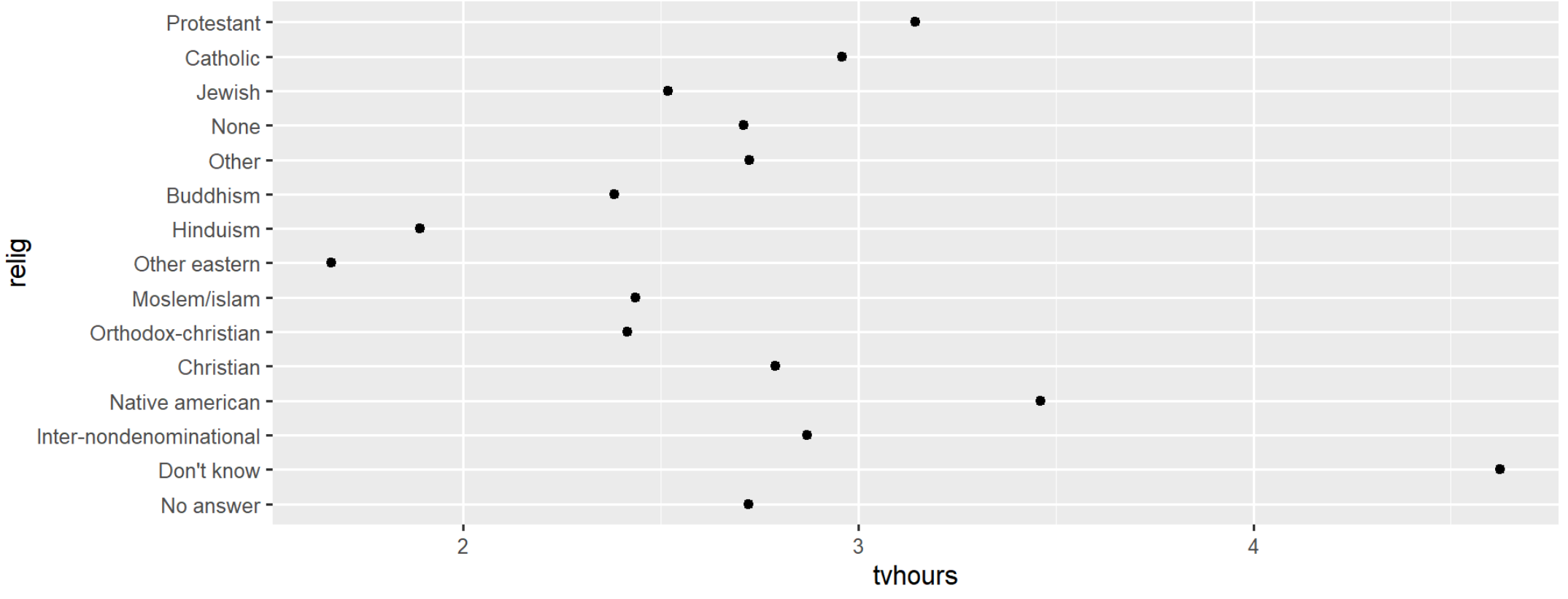
Change level order – `fct_reorder()`

Reorder according to another variable

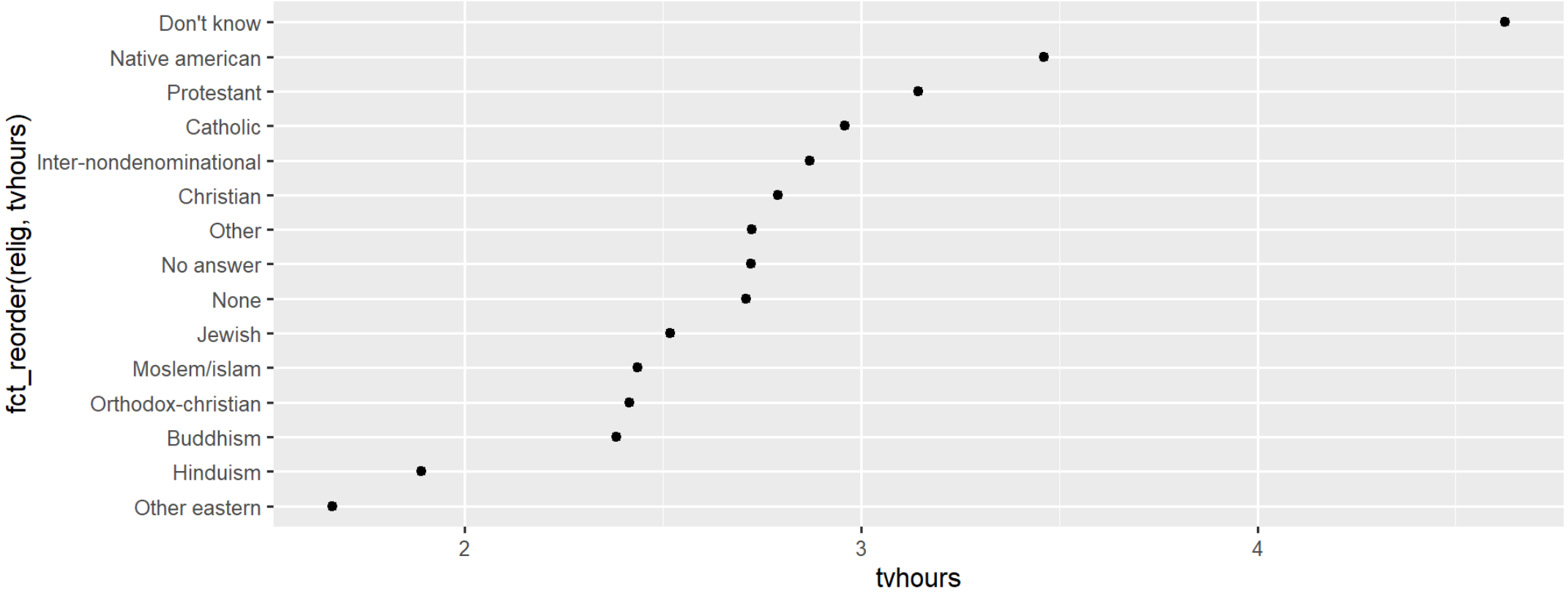
```
(relig_summary <- gss_cat %>%
  group_by(relig) %>%
  summarise(tvhours = mean(tvhours, na.rm = TRUE),
            n = n()))
```

```
## # A tibble: 15 x 3
##    relig                   tvhours     n
##    <fct>                     <dbl> <int>
##  1 No answer                  2.72    93
##  2 Don't know                 4.62    15
##  3 Inter-nondenominational    2.87   109
##  4 Native american            3.46    23
##  5 Christian                  2.79   689
##  6 Orthodox-christian         2.42    95
##  7 Moslem/islam               2.44   104
##  8 Other eastern              1.67    32
##  9 Hinduism                   1.89    71
## 10 Buddhism                   2.38   147
## 11 Other                      2.73   224
```

```
ggplot(relig_summary, aes(tvhours, relig)) +
  geom_point()
```



```
ggplot(relig_summary, aes(tvhours, fct_reorder(relig, tvhours))) +
    geom_point()
```



Or `mutate()` the factor reorder

```
relig_summary %>%
  mutate(relig = fct_reorder(relig, tvhours)) %>%
  ggplot(aes(tvhours, relig)) +
  geom_point()
```

Quick aside for error bars

```
(relig_summary_eb <- gss_cat %>%
  group_by(relig) %>%
  summarise(tvhours_mean = mean(tvhours, na.rm = TRUE),
            tvhours_se   = sqrt(var(tvhours, na.rm = TRUE) /
                                  length(na.omit(tvhours))),
            n = n()))
```

```
## # A tibble: 15 x 4
##    relig                   tvhours_mean tvhours_se     n
##    <fct>                          <dbl>      <dbl> <int>
##  1 No answer                       2.72     0.326     93
##  2 Don't know                      4.62     3.01      15
##  3 Inter-nondenominational         2.87     0.363    109
##  4 Native american                 3.46     1.13      23
##  5 Christian                       2.79     0.126    689
##  6 Orthodox-christian              2.42     0.355     95
##  7 Moslem/islam                    2.44     0.269    104
##  8 Other eastern                   1.67     0.449     32
##  9 Hinduism                        1.89     0.197     71
## 10 Buddhism                        2.38     0.235    147
## 11 Other                           2.73     0.203    224
## 12 None                            2.71     0.0590  3523
```

Quick aside for error bars

```
(relig_summary_eb <- gss_cat %>%
  group_by(relig) %>%
  summarise(tvhours_mean = mean(tvhours, na.rm = TRUE),
            tvhours_se   = sqrt(var(tvhours, na.rm = TRUE) /
                                  length(na.omit(tvhours))),
            n = n()))
```

```
## # A tibble: 15 x 4
##    relig                   tvhours_mean tvhours_se     n
##    <fct>                          <dbl>      <dbl> <int>
##  1 No answer                       2.72     0.326     93
##  2 Don't know                      4.62     3.01      15
##  3 Inter-nondenominational         2.87     0.363    109
##  4 Native american                 3.46     1.13      23
##  5 Christian                       2.79     0.126    689
##  6 Orthodox-christian              2.42     0.355     95
##  7 Moslem/islam                    2.44     0.269    104
##  8 Other eastern                   1.67     0.449     32
##  9 Hinduism                        1.89     0.197     71
## 10 Buddhism                        2.38     0.235    147
## 11 Other                           2.73     0.203    224
## 12 None                            2.71     0.0590  3523
```

Quick aside for error bars

```
(relig_summary_eb <- gss_cat %>%
  group_by(relig) %>%
  summarise(tvhours_mean = mean(tvhours, na.rm = TRUE),
            tvhours_se   = sqrt(var(tvhours, na.rm = TRUE) /
                                  length(na.omit(tvhours))),
            n = n()))
```

```
## # A tibble: 15 x 4
##    relig                   tvhours_mean tvhours_se     n
##    <fct>                          <dbl>      <dbl> <int>
##  1 No answer                       2.72     0.326     93
##  2 Don't know                      4.62     3.01      15
##  3 Inter-nondenominational         2.87     0.363    109
##  4 Native american                 3.46     1.13      23
##  5 Christian                       2.79     0.126    689
##  6 Orthodox-christian              2.42     0.355     95
##  7 Moslem/islam                    2.44     0.269    104
##  8 Other eastern                   1.67     0.449     32
##  9 Hinduism                        1.89     0.197     71
## 10 Buddhism                        2.38     0.235    147
## 11 Other                           2.73     0.203    224
## 12 None                            2.71     0.0590  3523
```

Quick aside for error bars

```
(relig_summary_eb <- gss_cat %>%
  group_by(relig) %>%
  summarise(tvhours_mean = mean(tvhours, na.rm = TRUE),
            tvhours_se   = sqrt(var(tvhours, na.rm = TRUE) /
                                  length(na.omit(tvhours))),
            n = n()))
```

```
## # A tibble: 15 x 4
##    relig                   tvhours_mean tvhours_se     n
##    <fct>                          <dbl>      <dbl> <int>
##  1 No answer                       2.72     0.326     93
##  2 Don't know                      4.62     3.01      15
##  3 Inter-nondenominational         2.87     0.363    109
##  4 Native american                 3.46     1.13      23
##  5 Christian                       2.79     0.126    689
##  6 Orthodox-christian              2.42     0.355     95
##  7 Moslem/islam                    2.44     0.269    104
##  8 Other eastern                   1.67     0.449     32
##  9 Hinduism                        1.89     0.197     71
## 10 Buddhism                        2.38     0.235    147
## 11 Other                           2.73     0.203    224
## 12 None                            2.71     0.0590  3523
```

Quick aside for error bars

```
ggplot(relig_summary_eb,
       aes(tvhours_mean, fct_reorder(relig, tvhours_mean))) +
  geom_errorbarh(aes(xmin = tvhours_mean - 1.96 * tvhours_se,
                     xmax = tvhours_mean + 1.96 * tvhours_se),
                 color = "cornflowerblue") +
  geom_point()
```

Quick aside for error bars

```
ggplot(relig_summary_eb,
       aes(tvhours_mean, fct_reorder(relig, tvhours_mean))) +
  geom_errorbarh(aes(xmin = tvhours_mean - 1.96 * tvhours_se,
                     xmax = tvhours_mean + 1.96 * tvhours_se),
                 color = "cornflowerblue") +
  geom_point()
```

Quick aside for error bars

```
ggplot(relig_summary_eb,
       aes(tvhours_mean, fct_reorder(relig, tvhours_mean))) +
  geom_errorbarh(aes(xmin = tvhours_mean - 1.96 * tvhours_se,
                     xmax = tvhours_mean + 1.96 * tvhours_se),
                 color = "cornflowerblue") +
  geom_point()
```

Quick aside for error bars

```
ggplot(relig_summary_eb,
       aes(tvhours_mean, fct_reorder(relig, tvhours_mean))) +
  geom_errorbarh(aes(xmin = tvhours_mean - 1.96 * tvhours_se,
                     xmax = tvhours_mean + 1.96 * tvhours_se),
                 color = "cornflowerblue") +
  geom_point()
```

Modifying factor levels – `fct_recode()`

Make modifying factors more explicit

`fct_recode(var_name, "new level" = "old level"...`

```
gss_cat %>%
  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")) %>%
  count(partyid)
```

```r
gss_cat %>%
  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")) %>%
  count(partyid)
```

```
## # A tibble: 10 x 2
##    partyid                   n
##    <fct>                 <int>
##  1 No answer               154
##  2 Don't know                1
##  3 Other party             393
##  4 Republican, strong     2314
##  5 Republican, weak       3032
##  6 Independent, near rep  1791
##  7 Independent            4119
##  8 Independent, near dem  2499
##  9 Democrat, weak         3690
## 10 Democrat, strong       3490
```

Collapsing levels – `fct_recode()`

`fct_recode()` can also be used to collapse levels easily

```
gss_cat %>%
  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",
    "Other"                 = "No answer",
    "Other"                 = "Don't know",
    "Other"                 = "Other party")) %>%
  count(partyid)
```

```r
gss_cat %>%
  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",
    "Other"                 = "No answer",
    "Other"                 = "Don't know",
    "Other"                 = "Other party")) %>%
  count(partyid)
```

```
## # A tibble: 8 x 2
##   partyid                   n
##   <fct>                 <int>
## 1 Other                   548
## 2 Republican, strong     2314
## 3 Republican, weak       3032
## 4 Independent, near rep  1791
## 5 Independent            4119
## 6 Independent, near dem  2499
## 7 Democrat, weak         3690
## 8 Democrat, strong       3490
```

Collapsing levels – `fct_collapse()`

`fct_collapse()` is one of the more useful functions in `{forcats}`

- Collapse all categories into Republican, Democrat, Independent, or Other

```
gss_cat %>%
  mutate(partyid = fct_collapse(partyid,
         Other = c("No answer", "Don't know", "Other party"),
         Rep = c("Strong republican", "Not str republican"),
         Ind = c("Ind,near rep", "Independent", "Ind,near dem"),
         Dem = c("Not str democrat", "Strong democrat")
  )) %>%
  count(partyid)
```

```
## # A tibble: 4 x 2
##   partyid     n
##   <fct>   <int>
## 1 Other     548
## 2 Rep      5346
## 3 Ind      8409
## 4 Dem      7180
```

Collapsing levels – `fct_lump_?()`

`fct_lump_?()` – **"lump" a bunch of categories together**

- `fct_lump_n(factor_variable, n)`: lumps all levels except for the `n` most frequent (or least frequent if `n` < 0) into "Other" level
- `fct_lump_min(factor_variable, min)`: lumps levels that appear fewer than `min` times
- `fct_lump_prop(factor_variable, prop)`: lumps levels that appear in fewer `prop` × `n` times

Collapsing levels – `fct_lump_n()`

Collapse to `n` = **9** religious groups: top 8 groups plus "Other"

```
gss_cat %>%
  mutate(rel = fct_lump_n(relig, 9)) %>%
  count(rel)
```

```
## # A tibble: 9 x 2
##   rel                        n
##   <fct>                  <int>
## 1 Inter-nondenominational   109
## 2 Christian                 689
## 3 Moslem/islam              104
## 4 Buddhism                  147
## 5 None                     3523
## 6 Jewish                    388
## 7 Catholic                 5124
## 8 Protestant              10846
## 9 Other                     553
```

Collapsing levels – `fct_lump_min()`

Collapse to all religious groups that appear less than `min` = 200 into "Other"

```
gss_cat %>%
  mutate(rel = fct_lump_min(relig, min = 200)) %>%
  count(rel)
```

```
## # A tibble: 6 x 2
##   rel            n
##   <fct>      <int>
## 1 Christian    689
## 2 None        3523
## 3 Jewish       388
## 4 Catholic    5124
## 5 Protestant 10846
## 6 Other        913
```

Collapsing levels – `fct_lump_prop()`

Collapse to all religious groups that appear less than `prop` = 10% into "Other"

```
gss_cat %>%
  mutate(rel = fct_lump_prop(relig, prop = .10)) %>%
  count(rel)
```

```
## # A tibble: 4 x 2
##   rel            n
##   <fct>      <int>
## 1 None        3523
## 2 Catholic    5124
## 3 Protestant 10846
## 4 Other       1990
```

Missing levels

```
levels(gss_cat$race)
```

```
## [1] "Other"          "Black"          "White"          "Not applicable"
```

```
gss_cat %>%
  count(race)
```

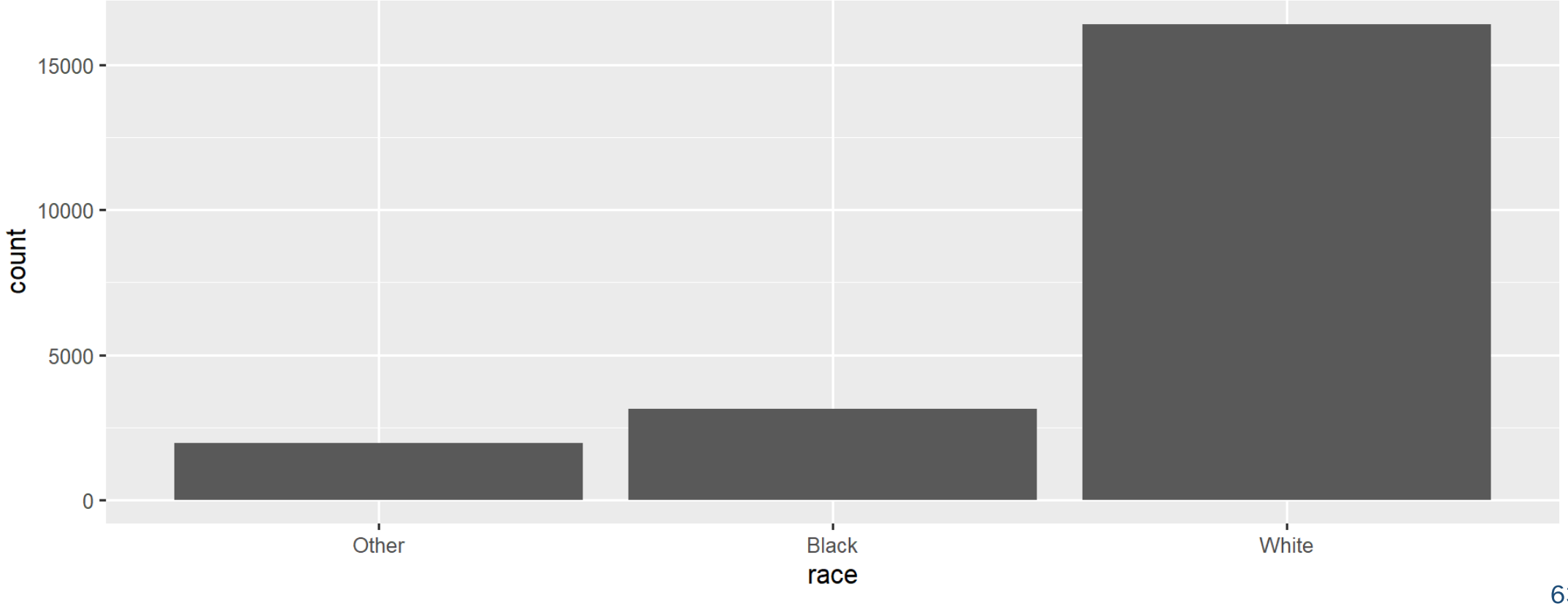
```
## # A tibble: 3 x 2
##   race      n
##   <fct> <int>
## 1 Other  1959
## 2 Black  3129
## 3 White 16395
```

```
table(gss_cat$race)
```

```
##
##          Other          Black          White Not applicable
##           1959           3129          16395              0
```

Missing levels

```
ggplot(gss_cat, aes(race)) +
    geom_bar()
```



Missing levels

```
ggplot(gss_cat, aes(race)) +
    geom_bar() +
    scale_x_discrete(drop = FALSE)
```

Review

`fct_inorder()`: Levels ordered as entered

`fct_infreq()`: Change level order in order of frequency (largest first)

`fct_relevel()`: Change level order by hand

`fct_reorder()`: Change level order according to another variable

`fct_recode()`: Recode (collapse) levels into new named levels

`fct_collapse()`: Recode many levels into fewer levels

`fct_lump_?()`: Recode all levels into "Other":

- except for the `n` most frequent - `fct_lump_n()`
- that appear fewer than `min` times - `fct_lump_min()`
- that appear less than `prop`% - `fct_lump_prop()`

Try it

- Go back to your Homework 9 assignment
- Reproduce the plot using `fct_relevel()` and `fct_recode()`

Pull Requests

Peer Review of Data Prep Script

Expectations

Feedback:

1. Note at least three areas of strength
2. Note at least one thing you learned from reviewing their script
3. Note at least one and no more than three areas for improvement

Making your code publicly available can feel daunting

- The purpose of this portion of the final project is to help us all learn from each other
- We are all learning
  - Be constructive in your feedback
  - Be kind
- Under no circumstances will negative comments be tolerated
  - Any comments that could be perceived as negative, and outside the scope of the code, will result in an immediate score of zero

Peer Review GitHub Process

1. Locate GitHub repo of assigned peer to review

2. Fork the repo

3. Clone the repo

4. Provide script feedback

    - edit the .Rmd file directly
    - edit code
    - provide comments in code and/or text (`Ctrl/Command + Shift + C`)
    - *commit* & *push*

5. Create Pull Request (PR)

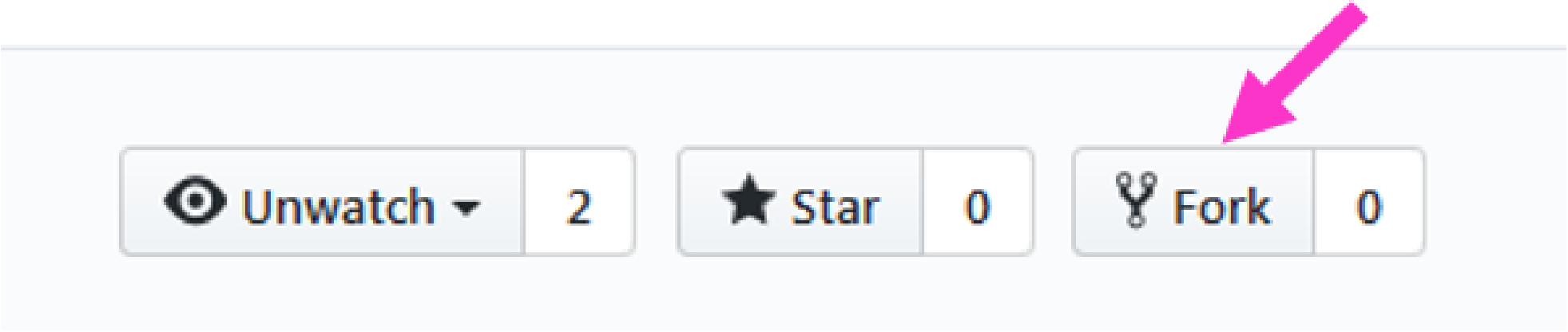
- Write brief summary of the PR that includes
    - >= 3 strengths
    - >= 1 thing you learned
    - 1 to 3 three areas of improvement

1. Locate GitHub repo of assigned peer to review

Student	Repo to Review	File to Review
Errol Kaylor	https://github.com/laurenberny/final_git	markdown.Rmd
Havisha Khurana	https://github.com/laurenberny/final_git	markdown.Rmd
Merly Klaas	https://github.com/ekgekd/EDLD651_Final.git	EDLD651_finalproj.Rmd
Cassie Malcom	https://github.com/ekgekd/EDLD651_Final.git	EDLD651_finalproj.Rmd
Manuel Vazquez Cano	https://github.com/laurenberny/final_git	markdown.Rmd
Yijun Cheng	https://github.com/ZF-MPH/final_project.git	final_project.Rmd
Claire Guidinger	https://github.com/ZF-MPH/final_project.git	final_project.Rmd
Marielena McWhirter	https://github.com/heatherleonard/EDLD651Final_Project.git	DraftData.Rmd
Mandi Ward	https://github.com/cguidin4/final_project.git	final_project.Rmd
Amy Warnock	https://github.com/ZF-MPH/final_project.git	final_project.Rmd
Lauren Berny	https://github.com/cguidin4/final_project.git	final_project.Rmd
Adriana Conn	https://github.com/ekgekd/EDLD651_Final.git	EDLD651_finalproj.Rmd
Mavis Gallo	https://github.com/mandiward17/Draft_data_set.git	draft_data_script.Rmd
Aubrey Green	https://github.com/merlyklaas/final_project.git	final_project_script.Rmd
Shawn McWeeny	https://github.com/mandiward17/Draft_data_set.git	draft_data_script.Rmd
Rebecca Gordon	https://github.com/mandiward17/Draft_data_set.git	draft_data_script.Rmd
Heather Leonard	https://github.com/merlyklaas/final_project.git	final_project_script.Rmd
Abbie Sanders	https://github.com/laurenberny/final_git	markdown.Rmd
Elizabeth Bates	https://github.com/merlyklaas/final_project.git	final_project_script.Rmd
Esmeralda Castro	https://github.com/laurenberny/final_git	markdown.Rmd
Zach Farley	https://github.com/heatherleonard/EDLD651Final_Project.git	DraftData.Rmd

2. Fork the repo

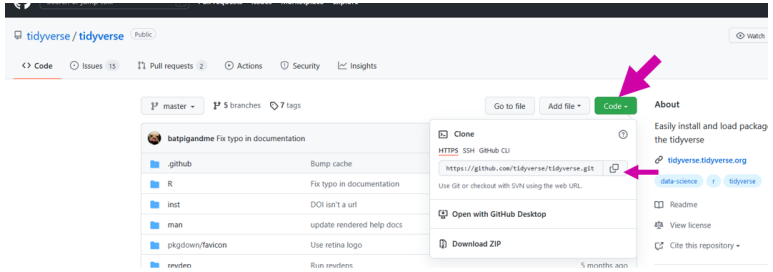
1. Navigate to the (host) GitHub repo
2. Click Fork in the upper right corner
3. Where to fork? – **your GitHub account**



3. Clone the repo

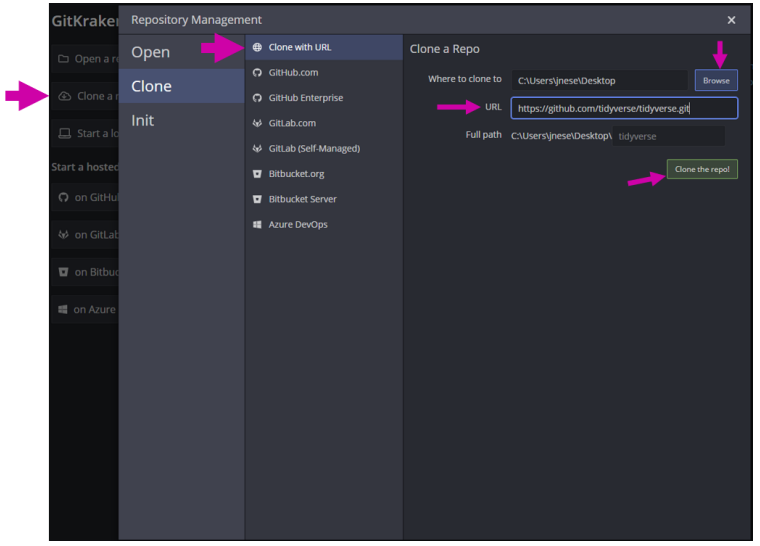
(1) Clone the repo

- copy the URL



(2) Open GitKraken

- *Clone with URL*
- Where will it live on your local machine?
  - it's own folder, with no other RProjects



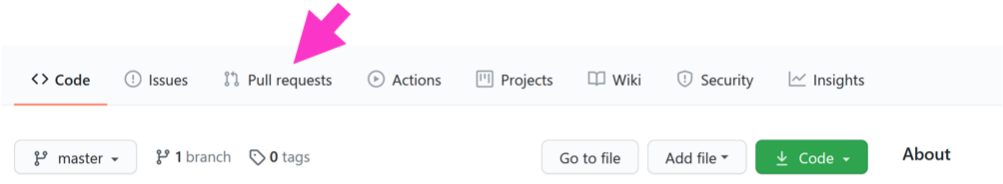
4. Provide script feedback

- Open R Project in your *local*
- Find the .Rmd document you will be reviewing
  - it should be an Rmd document
- Make your edits/comments
  - edit code as you like
  - include a comment for each edit!
  - Provide comments in code **and/or** text (`Ctrl/Command + Shift + C`)
- Commit as you go (if you are working on this across sessions/days)
- **Push only when you are finished**

5. Create Pull Request (PR)

(1) Navigate back to the (host) GitHub repo

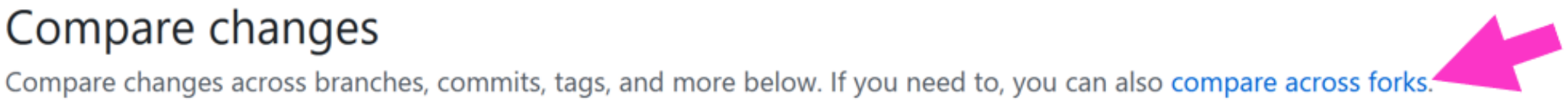
(2) Click “Pull requests”



(3) Click “New pull request”

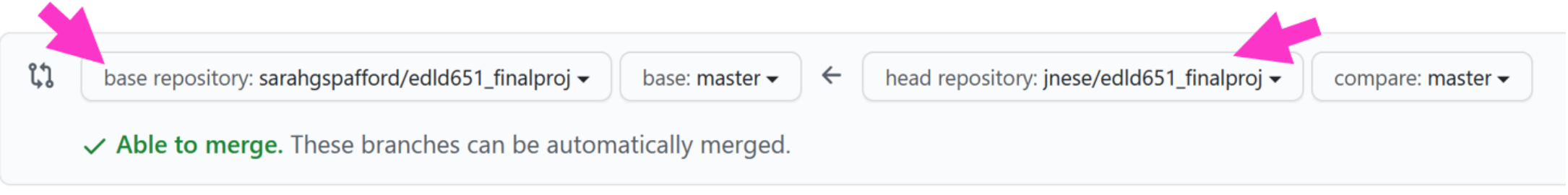
5. Create Pull Request (PR)

(4) Click "*Compare across forks*"



Use drop-downs so that:

- *host* repo is on **left**
- *your* repo is on the **right**

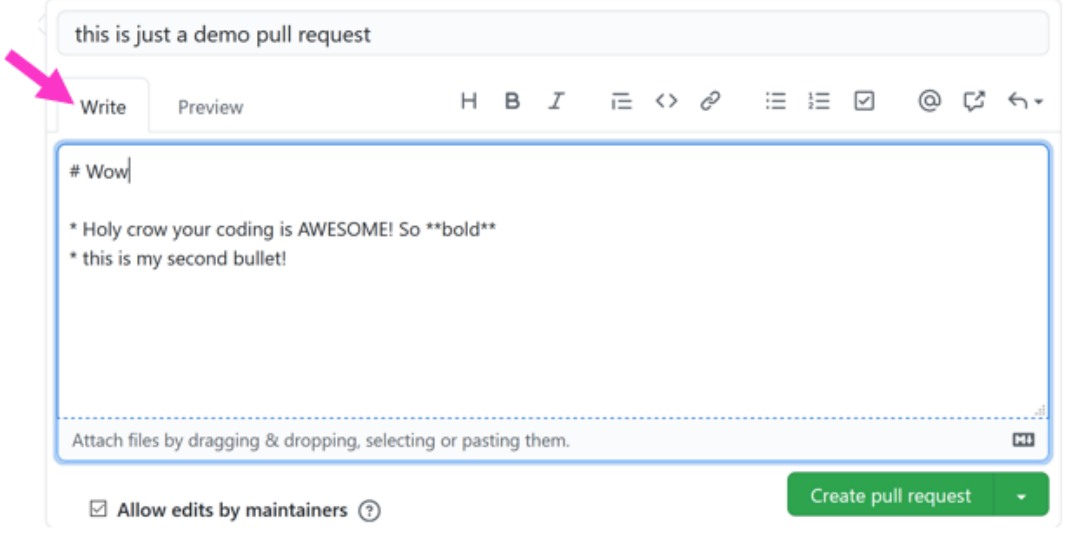


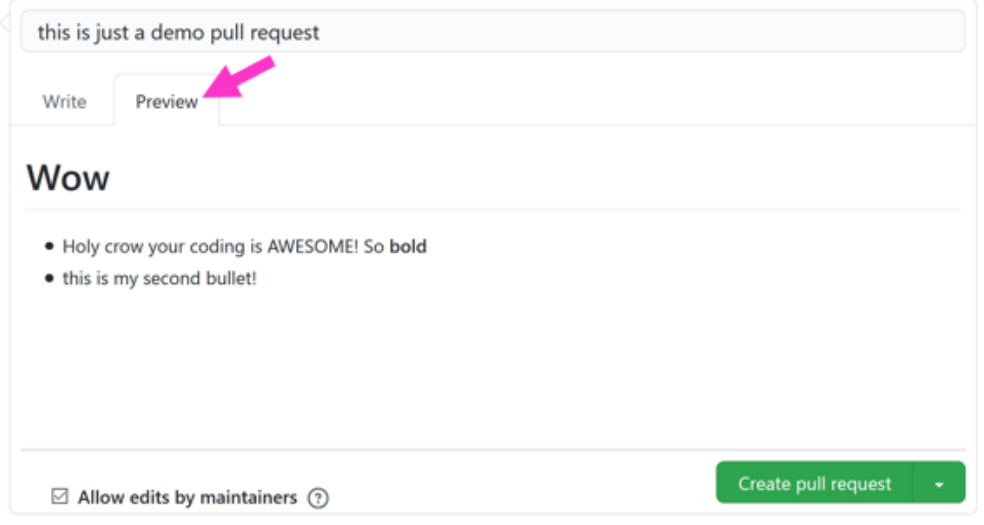
You will be able view the changes you made to the .Rmd document

5. Create Pull Request (PR)

Write a brief summary list of the PR that includes

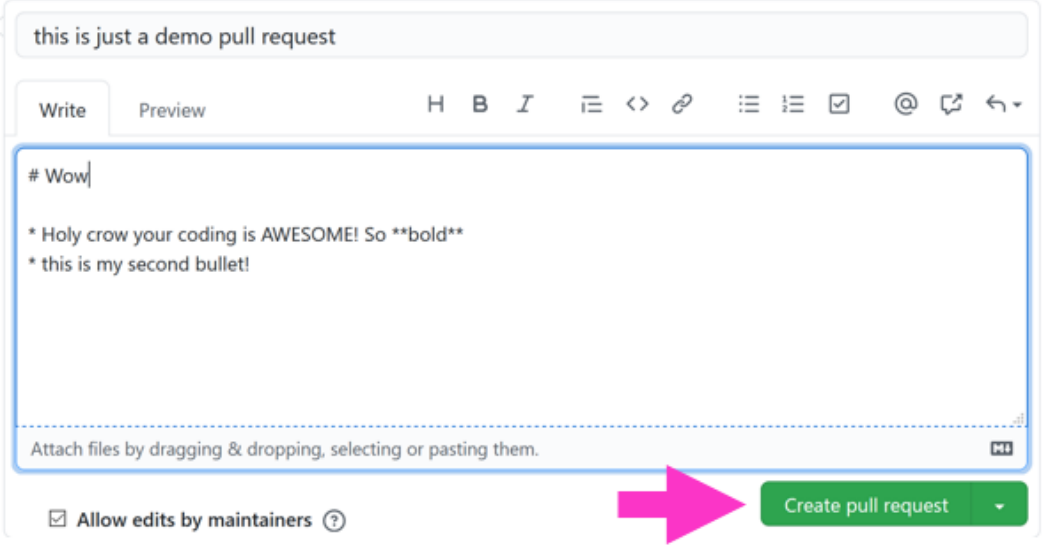
- >= 3 strengths
- >= 1 thing you learned
- 1 to 3 three areas of improvement
- **Use markdown formatting, headers or list!**

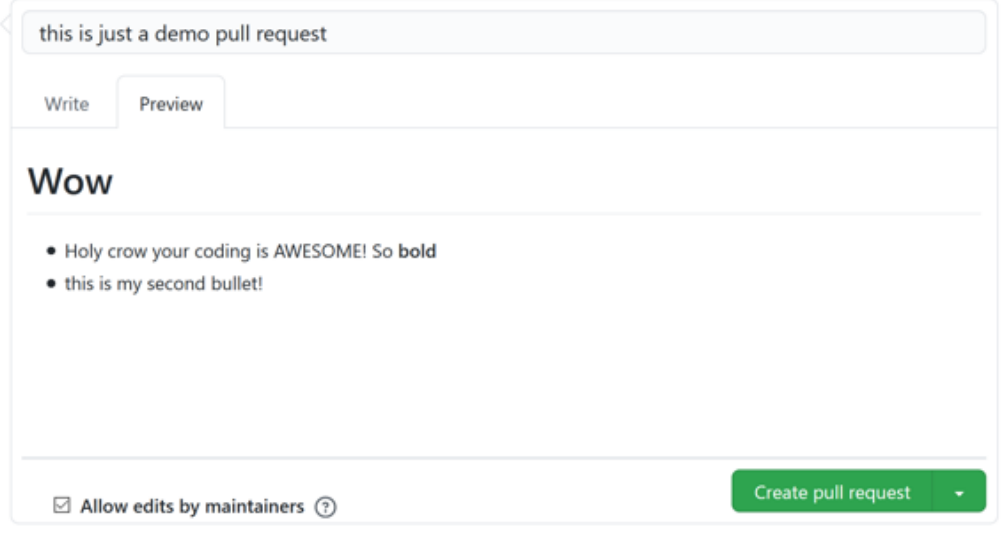




5. Create Pull Request (PR)

(6) Click "*Create pull request*" when you're done





5. Create Pull Request (PR)

Recap

(1) Navigate back to the (host) GitHub repo

(2) Click "*Pull requests*"

(3) Click "*New pull request*"

(4) Click "*Compare across forks*"

- Use drop-downs so that:
  - Host repo is on the left, your repo is on the right
- View changes (5) Click "*Create pull request*"
- Write brief summary list of the PR that includes
  - >= 3 strengths
  - >= 1 thing you learned
  - 1 to 3 three areas of improvement
  - **Use markdown formatting, headers, or list!** (6) Click "*Create pull request*"

Reviewing your PRs

You will get an email from GitHub

1. Click on first link, for PR
2. Click "*Commits*" tab
3. Click on "*File changes*" to see changes
4. Copy/paste all desired changes
5. Don't close "*Close PR*" just yet; I want to review

Next time

Before next class

- Final Project
  - Final Project: Peer Review of Script
  - Final Project: Presentations - email me your content before class
- Homework
  - **Homework 10**