Week 4: Graphing Amounts & Proportions

EMSE 4197 | John Paul Helveston | February 05, 2020

Before we start

1. Course site updates
2. R tip of the week
3. Refresher: Tidy data & the `gather()` function

Check out the `fcuk` package

Install:

```
install.packages("fcuk")
```

Error message without the `fcuk` package:

```
maen(c(1, 2, 3, 4, 5))
```

```
Error in maen(c(1, 2, 3, 4, 5)) : could not find function "maen"
```

Check out the `fcuk` package

Install:

```
install.packages("fcuk")
```

Error message **with** the `fcuk` package:

```
library(fcuk)

maen(c(1, 2, 3, 4, 5))
```

```
Error in maen(c(1, 2, 3, 4, 5)) : could not find function "maen"

Did you mean : mean or rename ?
```

Automatically load:

```
fcuk::add_fcuk_to_rprofile()
```

Bonus tip:

Check out this site on how to make the 50 most common plots

Tidy-ing your data with `gather()`

```
dataPath <- here('data', 'federal_spending_wide.csv')
spending_wide <- read_csv(dataPath)

head(spending_wide)
```

```
## # A tibble: 6 x 15
##    year   DHS   DOC   DOD   DOE   DOT   EPA   HHS Interior  NASA   NIH   N
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <db
## 1  1976     0   819 35696 10882  1142   968  9226     1152 12513  8025  23
## 2  1977     0   837 37967 13741  1095   966  9507     1082 12553  8214  23
## 3  1978     0   871 37022 15663  1156  1175 10533     1125 12516  8802  24
## 4  1979     0   952 37174 15612  1004  1102 10127     1176 13079  9243  24
## 5  1980     0   945 37005 15226  1048   903 10045     1082 13837  9093  24
## 6  1981     0   829 41737 14798   978   901  9644      990 13276  8580  23
##   Other  USDA    VA
##   <dbl> <dbl> <dbl>
## 1  1191  1837   404
## 2  1280  1796   374
## 3  1237  1962   356
## 4  2321  2054   353
## 5  2468  1887   359
## 6  1925  1964   382
```

```
spending_long <- spending_wide %>%
    gather(key = "department",
           value = "rd_budget",
           DHS:VA)

head(spending_long)
```

```
## # A tibble: 6 x 3
##    year department rd_budget
##   <dbl> <chr>           <dbl>
## 1  1976 DHS                 0
## 2  1977 DHS                 0
## 3  1978 DHS                 0
## 4  1979 DHS                 0
## 5  1980 DHS                 0
## 6  1981 DHS                 0
```

Tidy tricks

`key` = "name of header", `value` = "name of cells"

```
dataPath <- here('data', 'federal_spending_wide.csv')
spending_wide <- read_csv(dataPath)

head(spending_wide)
```

```
## # A tibble: 6 x 15
##    year   DHS   DOC   DOD   DOE   DOT   EPA   HHS Interior  NASA   NIH   N
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <db
## 1  1976     0   819 35696 10882  1142   968  9226     1152 12513  8025  23
## 2  1977     0   837 37967 13741  1095   966  9507     1082 12553  8214  23
## 3  1978     0   871 37022 15663  1156  1175 10533     1125 12516  8802  24
## 4  1979     0   952 37174 15612  1004  1102 10127     1176 13079  9243  24
## 5  1980     0   945 37005 15226  1048   903 10045     1082 13837  9093  24
## 6  1981     0   829 41737 14798   978   901  9644      990 13276  8580  23
##   Other  USDA    VA
##   <dbl> <dbl> <dbl>
## 1  1191  1837   404
## 2  1280  1796   374
## 3  1237  1962   356
## 4  2321  2054   353
## 5  2468  1887   359
## 6  1925  1964   382
```

```
spending_long <- spending_wide %>%
    gather(key = "department",
           value = "rd_budget",
           DHS:VA)

head(spending_long)
```

```
## # A tibble: 6 x 3
##    year department rd_budget
##   <dbl> <chr>           <dbl>
## 1  1976 DHS                 0
## 2  1977 DHS                 0
## 3  1978 DHS                 0
## 4  1979 DHS                 0
## 5  1980 DHS                 0
## 6  1981 DHS                 0
```

Tidy tricks

`key` = "name of header", `value` = "name of cells"

```
head(spending_long)
```

```
## # A tibble: 6 x 3
##    year department rd_budget
##   <dbl> <chr>           <dbl>
## 1  1976 DHS                 0
## 2  1977 DHS                 0
## 3  1978 DHS                 0
## 4  1979 DHS                 0
## 5  1980 DHS                 0
## 6  1981 DHS                 0
```

```
spending_wide <- spending_long %>%
    spread(key = "department",
           value = "rd_budget")

head(spending_wide)
```

```
## # A tibble: 6 x 15
##    year   DHS   DOC   DOD   DOE   DOT   EPA   HHS Interior  NASA   NIH   N
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <db
## 1  1976     0   819 35696 10882  1142   968  9226     1152 12513  8025  23
## 2  1977     0   837 37967 13741  1095   966  9507     1082 12553  8214  23
## 3  1978     0   871 37022 15663  1156  1175 10533     1125 12516  8802  24
## 4  1979     0   952 37174 15612  1004  1102 10127     1176 13079  9243  24
## 5  1980     0   945 37005 15226  1048   903 10045     1082 13837  9093  24
## 6  1981     0   829 41737 14798   978   901  9644      990 13276  8580  23
##   Other  USDA    VA
##   <dbl> <dbl> <dbl>
## 1  1191  1837   404
## 2  1280  1796   374
## 3  1237  1962   356
## 4  2321  2054   353
## 5  2468  1887   359
## 6  1925  1964   382
```

Are your data tidy?

"is `cell value` a `column name`?"

```
head(spending_wide)
```

```
## # A tibble: 6 x 15
##    year   DHS   DOC   DOD   DOE   DOT   EPA   HHS Interior  NASA   NIH   N
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <db
## 1  1976     0   819 35696 10882  1142   968  9226     1152 12513  8025  23
## 2  1977     0   837 37967 13741  1095   966  9507     1082 12553  8214  23
## 3  1978     0   871 37022 15663  1156  1175 10533     1125 12516  8802  24
## 4  1979     0   952 37174 15612  1004  1102 10127     1176 13079  9243  24
## 5  1980     0   945 37005 15226  1048   903 10045     1082 13837  9093  24
## 6  1981     0   829 41737 14798   978   901  9644      990 13276  8580  23
##   Other  USDA    VA
##   <dbl> <dbl> <dbl>
## 1  1191  1837   404
## 2  1280  1796   374
## 3  1237  1962   356
## 4  2321  2054   353
## 5  2468  1887   359
## 6  1925  1964   382
```

```
spending_long <- spending_wide %>%
    gather(key = "department",
           value = "rd_budget",
           DHS:VA)

head(spending_long)
```

```
## # A tibble: 6 x 3
##    year department rd_budget
##   <dbl> <chr>           <dbl>
## 1  1976 DHS                 0
## 2  1977 DHS                 0
## 3  1978 DHS                 0
## 4  1979 DHS                 0
## 5  1980 DHS                 0
## 6  1981 DHS                 0
```

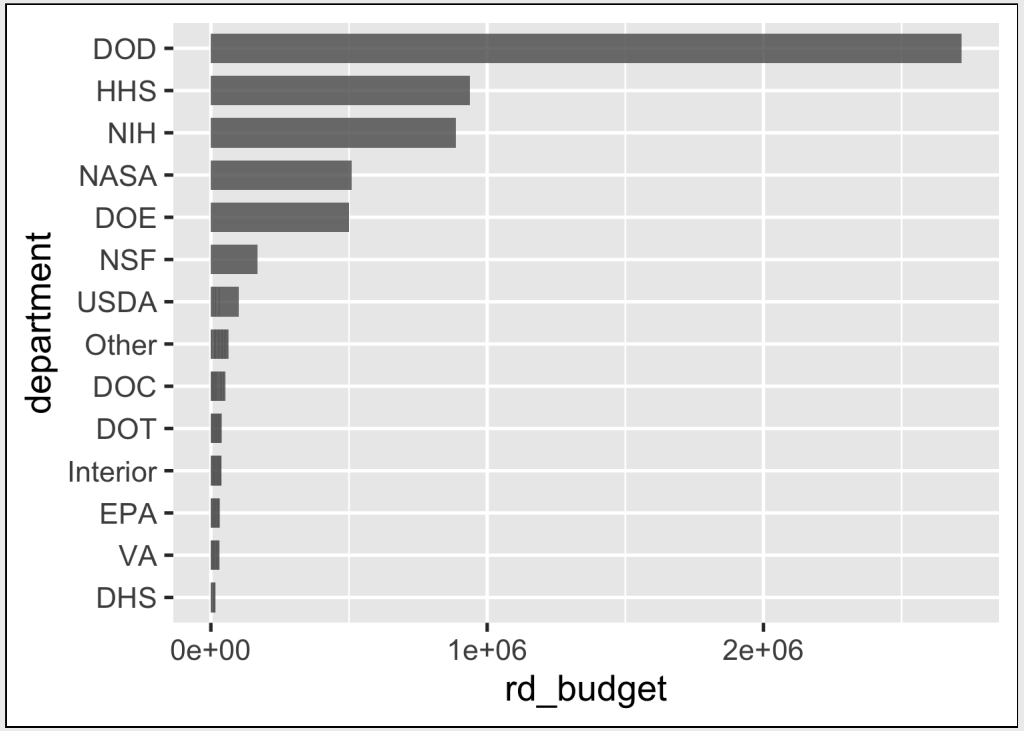
Why do we need tidy data?

Tidyverse was built for tidy data

```
head(spending_long)
```

```
## # A tibble: 6 x 3
##    year department rd_budget
##   <dbl> <fct>           <dbl>
## 1  1976 DHS                 0
## 2  1977 DHS                 0
## 3  1978 DHS                 0
## 4  1979 DHS                 0
## 5  1980 DHS                 0
## 6  1981 DHS                 0
```

```
ggplot(spending_long) +
  geom_col(aes(x = department, y = rd_budget),
           width = 0.7, alpha = 0.8) +
  coord_flip()
```



Topics

1. Making a (good) ggplot
2. Using facets
3. Manipulating factors
4. Graphing proportions

Read in the data

```
wildlife_impacts <- read_csv(here('data', 'wildlife_impacts.csv'))
college_all_ages <- read_csv(here('data', 'college_all_ages.csv'))
lotr_words       <- read_csv(here('data', 'lotr_words.csv'))
federal_spending <- read_csv(here('data', 'federal_spending_long.csv'))
avengers         <- read_csv(here('data', 'avengers.csv'))
milk_production  <- read_csv(here('data', 'milk_production.csv'))
```

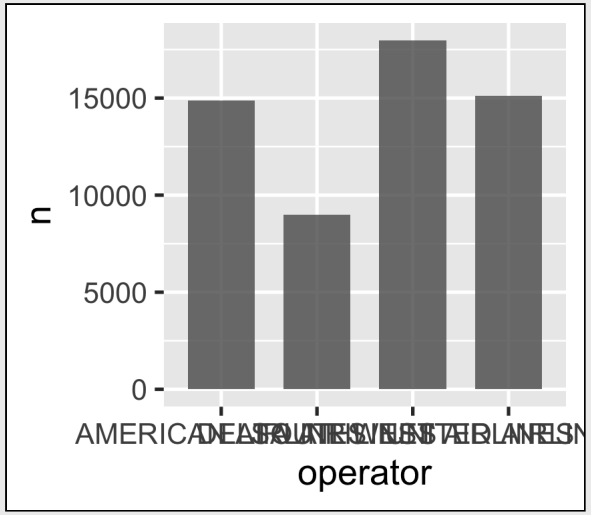
Topics

1. Making a (good) ggplot
2. Using facets
3. Manipulating factors
4. Graphing proportions

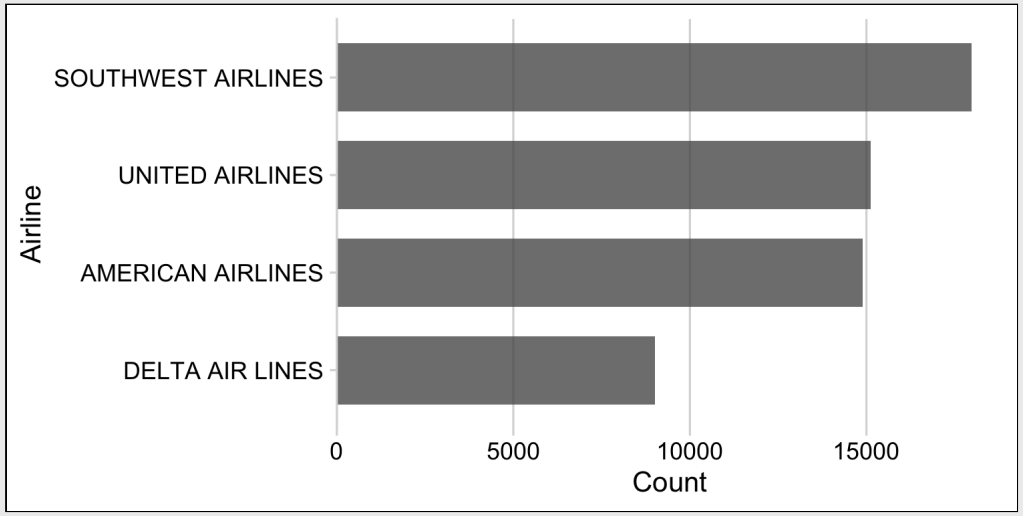
Making a (good) ggplot

1. Format data frame
2. Add main geoms
3. (Flip coordinate)
4. (Reorder factors)
5. Adjust scales
6. Adjust theme
7. Annotate

Before:



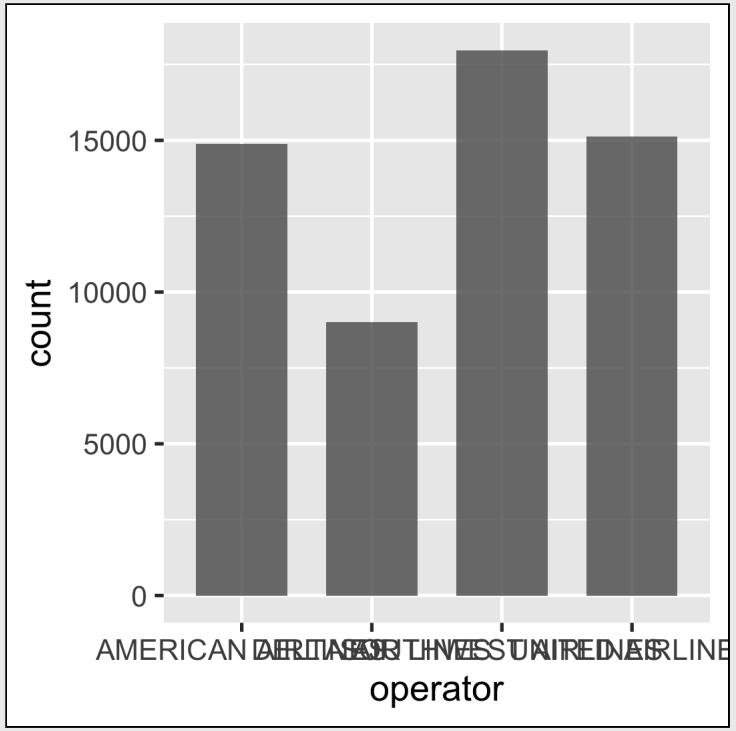
After:



1) Format data frame + 2) add main geoms

Method 1: Use `geom_bar()`

```
ggplot(wildlife_impacts) +
  geom_bar(aes(x = operator),
           width = 0.7, alpha = 0.8)
```



1) Format data frame + 2) add main geoms

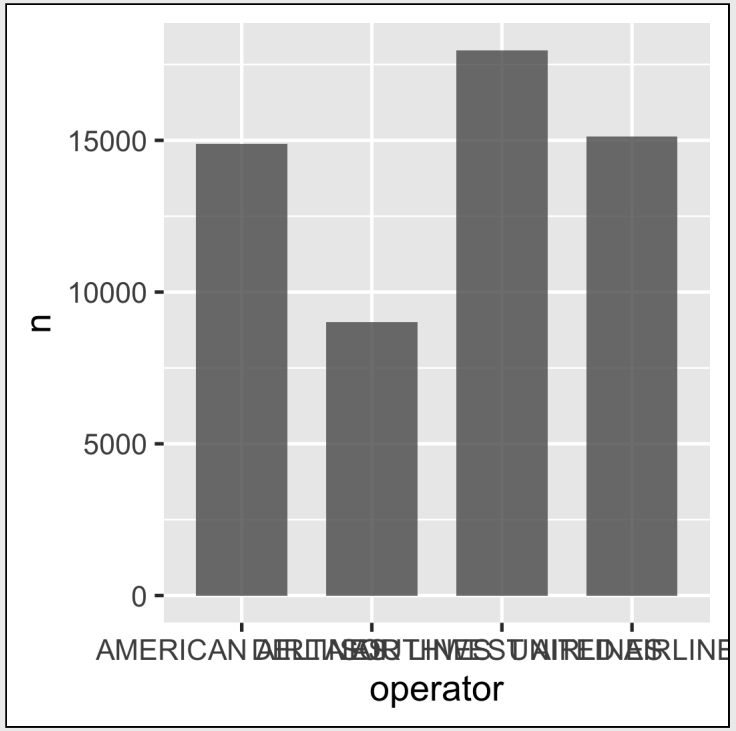
Method 2: Summarize the data first

```
wildlife_summary <- wildlife_impacts %>%
  count(operator)

wildlife_summary
```

```
## # A tibble: 4 x 2
##   operator              n
##   <chr>             <int>
## 1 AMERICAN AIRLINES  14887
## 2 DELTA AIR LINES     9005
## 3 SOUTHWEST AIRLINES 17970
## 4 UNITED AIRLINES    15116
```

...then use `geom_col()`

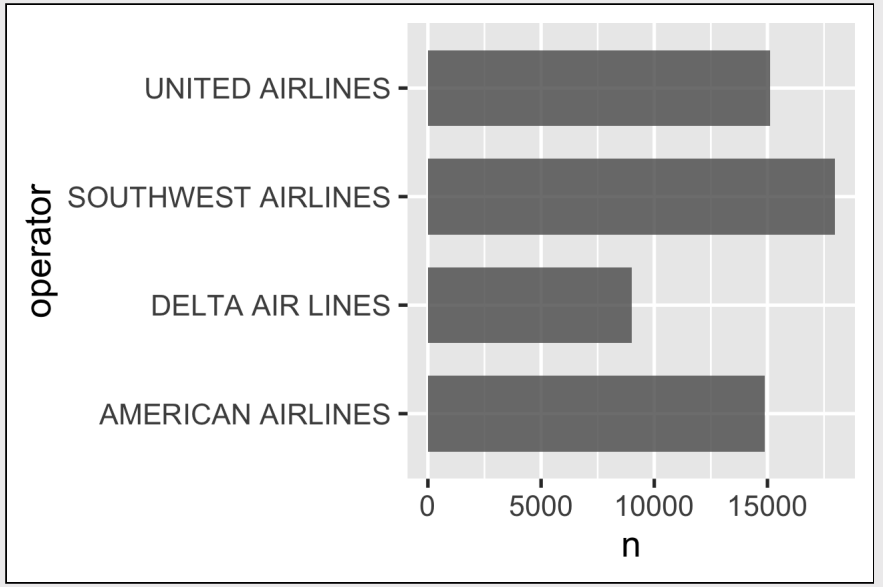
```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8)
```



3) Flip coordinates

```
wildlife_summary <- wildlife_impacts %>%
  count(operator)
```

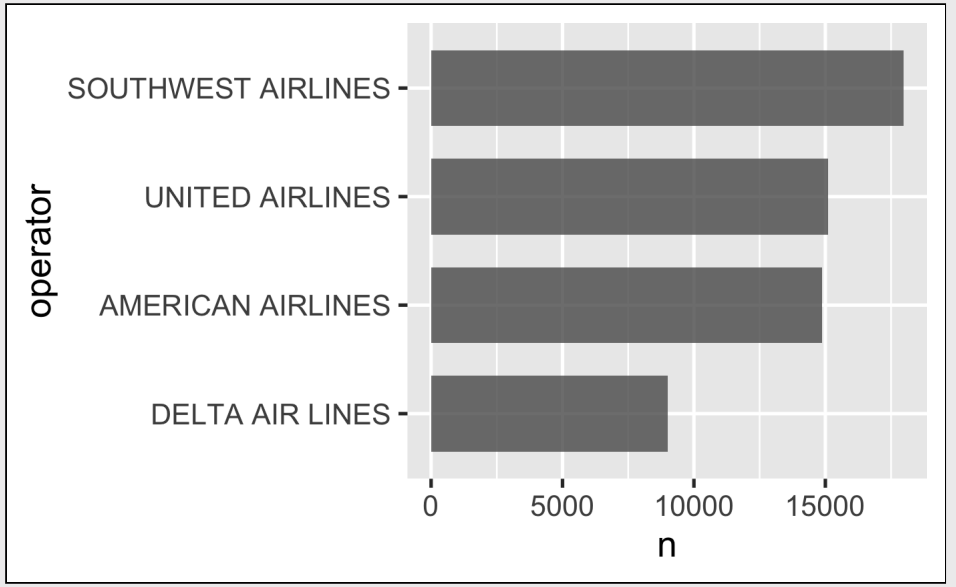
```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  coord_flip()
```



4) Reorder factors with `fct_reorder()`

```
wildlife_summary <- wildlife_impacts %>%
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))
```

```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  coord_flip()
```

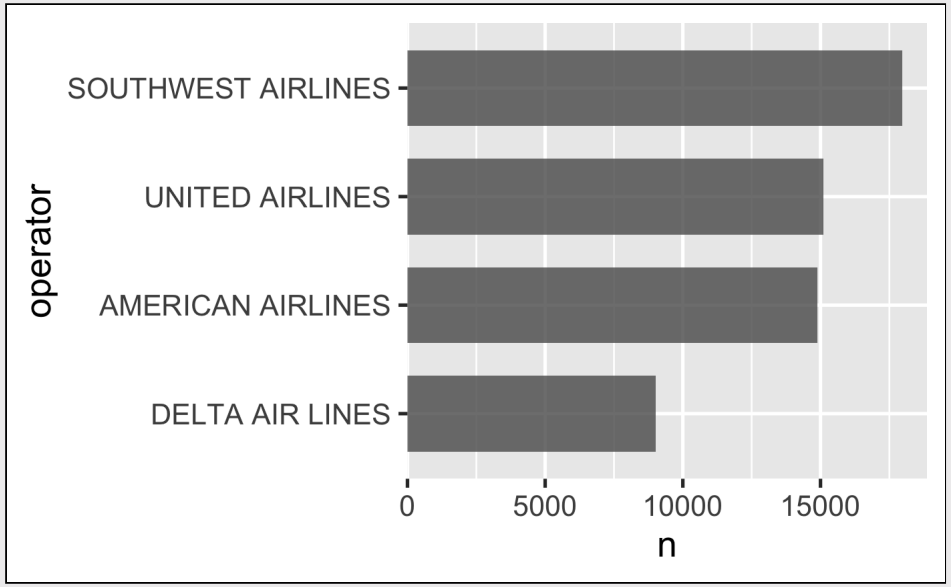


5) Adjust scales

```
wildlife_summary <- wildlife_impacts %>%
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))
```

Set y axis limit to 0

```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05)))
```

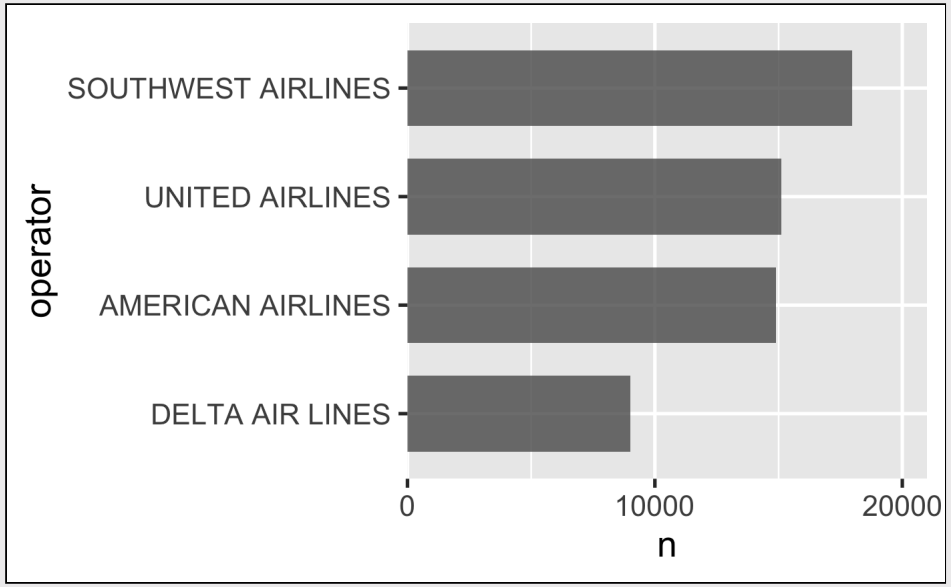


5) Adjust scales

```
wildlife_summary <- wildlife_impacts %>%
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))
```

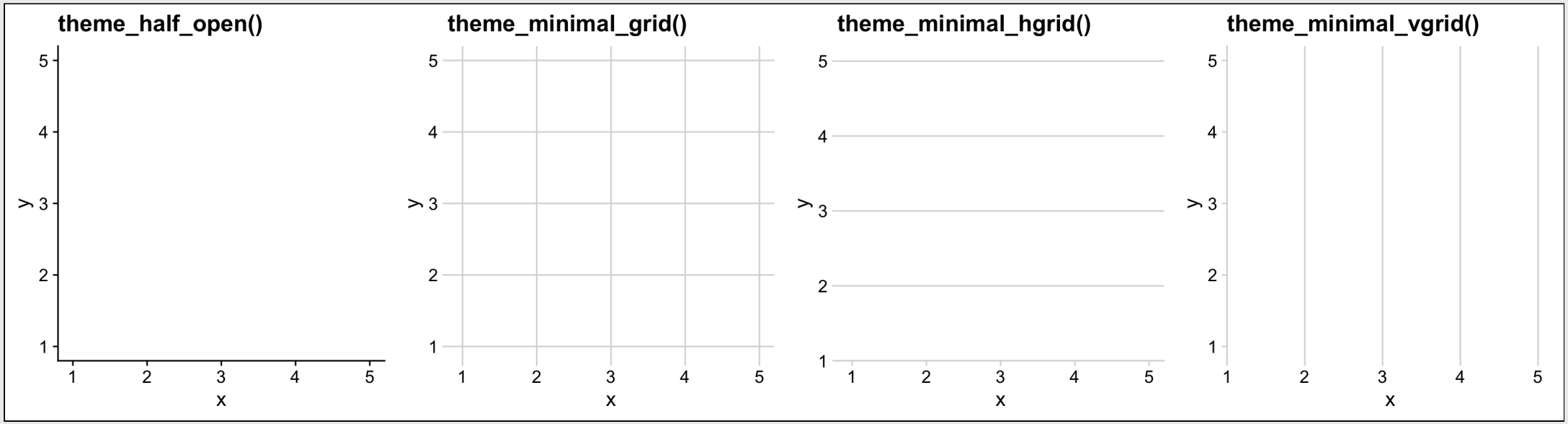
Customize y-axis break points

```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05)),
    breaks = c(0, 10000, 20000),
    limits = c(0, 20000))
```



6) Adjust theme

4 `cowplot` themes you should know



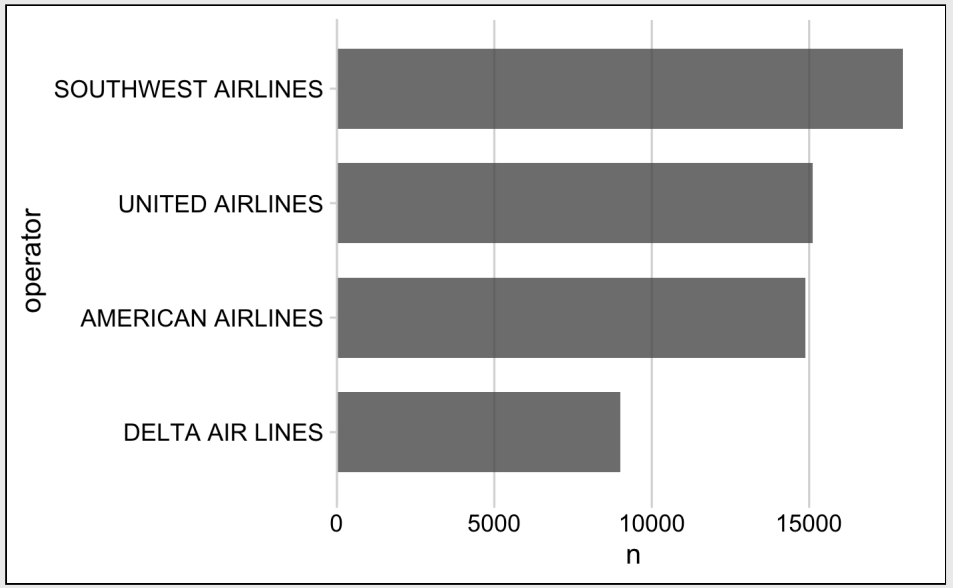
6) Adjust theme

For horizontal bars, add only vertical grid

`theme_minimal_vgrid()`

```
wildlife_summary <- wildlife_impacts %>%
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))
```

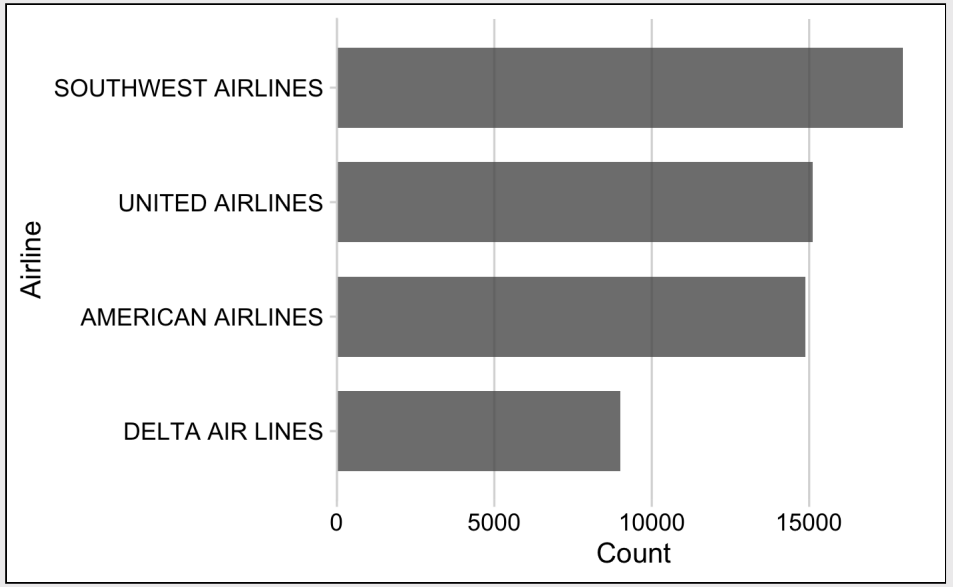
```
library(cowplot)

ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid()
```



7) Annotate

```
wildlife_summary <- wildlife_impacts %>%
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))
```

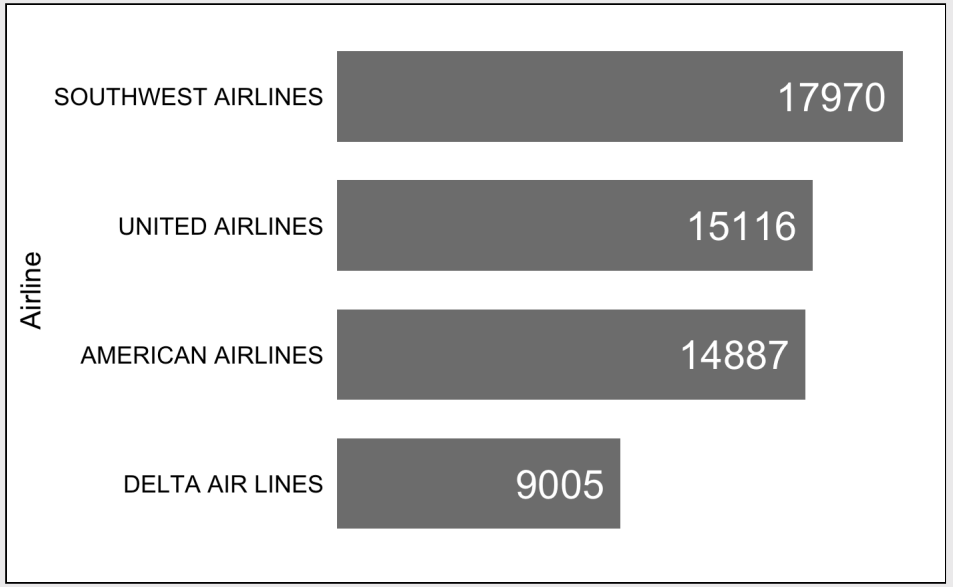
```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid() +
  labs(
    x = 'Airline',
    y = 'Count' )
```



7) Annotate - Can also just remove y axis

```
wildlife_summary <- wildlife_impacts %>%
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))
```

```
ggplot(wildlife_summary) +
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
  geom_text(aes(x = operator, y = n, label = n),
            hjust = 1, color = 'white',
            nudge_y = -500, size = 7) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid() +
  theme(
    line = element_blank(),
    axis.title.x = element_blank(),
    axis.text.x = element_blank()) +
  labs(
    x = 'Airline',
    y = 'Count' )
```



Making a (good) ggplot

Data frame:

```
wildlife_summary <- wildlife_impacts %>%          # 1. Format data frame
  count(operator) %>%
  mutate(
    operator = fct_reorder(operator, n))          # 4. (Reorder factors)
```

Plot:

```
ggplot(wildlife_summary) +                        # 2. Add main geoms
  geom_col(aes(x = operator, y = n),
           width = 0.7, alpha = 0.8) +
```

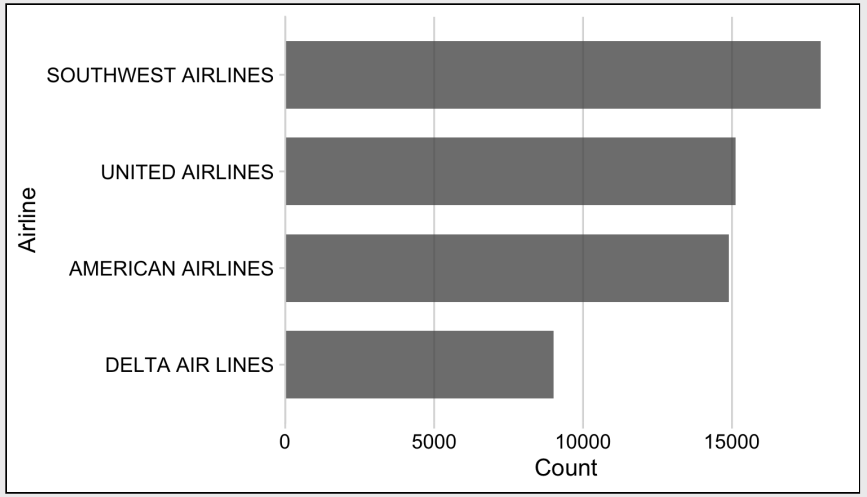
```
  coord_flip() +                                  # 3. (Flip coordinates)
```

```
  scale_y_continuous(                             # 5. Adjust scales
    expand = expand_scale(mult = c(0, 0.05))) +
```

```
  theme_minimal_vgrid() +                         # 6. Adjust theme
```

```
  labs(                                           # 7. Annotate
    x = 'Airline',
    y = 'Count' )
```

Final product



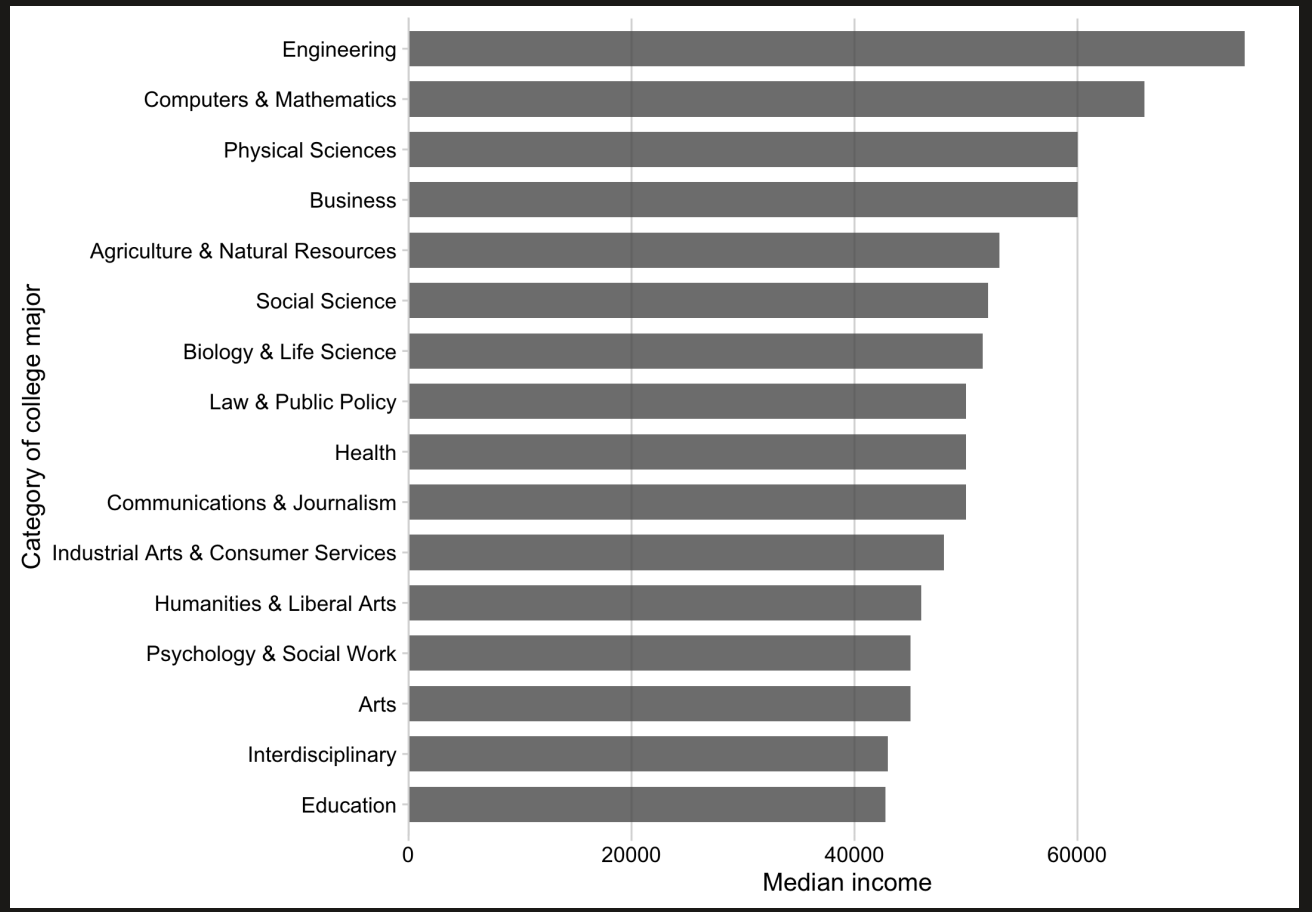
Your turn

Use the `college_all_ages.csv` data to create the following plot

Hint: You'll need to compute the median income for each category of major

Making a (good) ggplot:

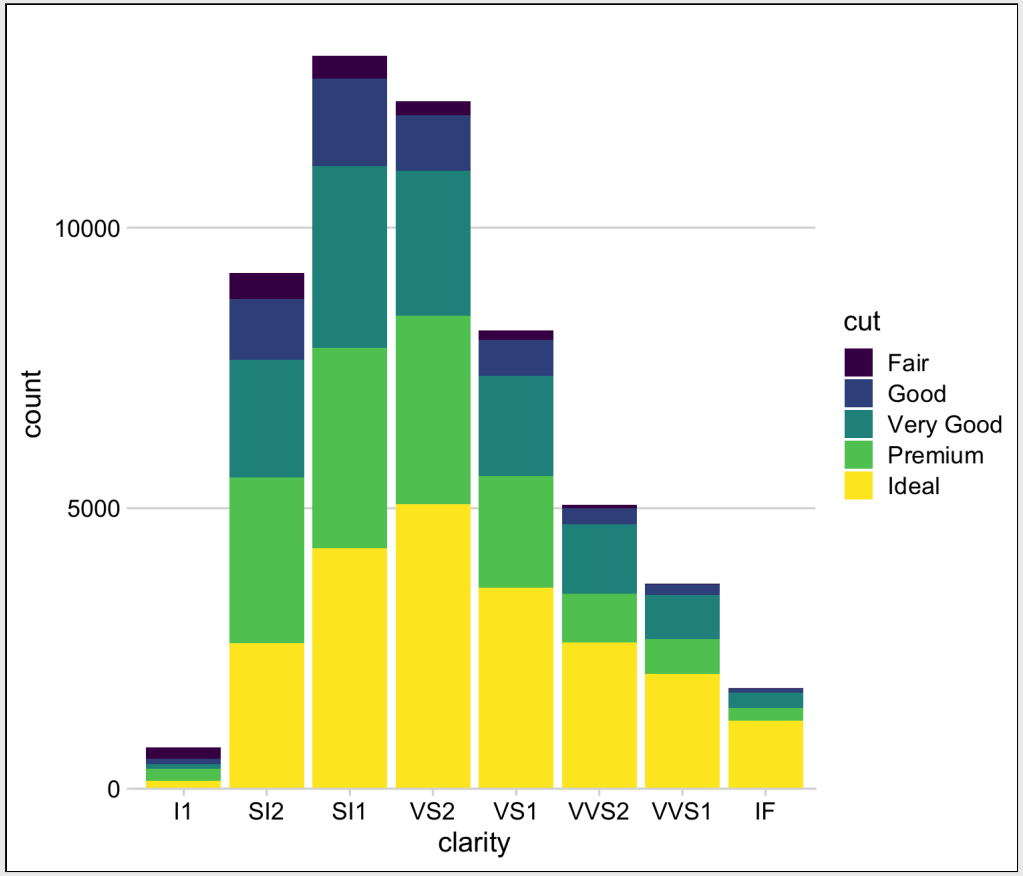
1. Format data frame
2. Add main geoms
3. (Flip coordinate)
4. (Reorder factors)
5. Adjust scales
6. Adjust theme
7. Annotate

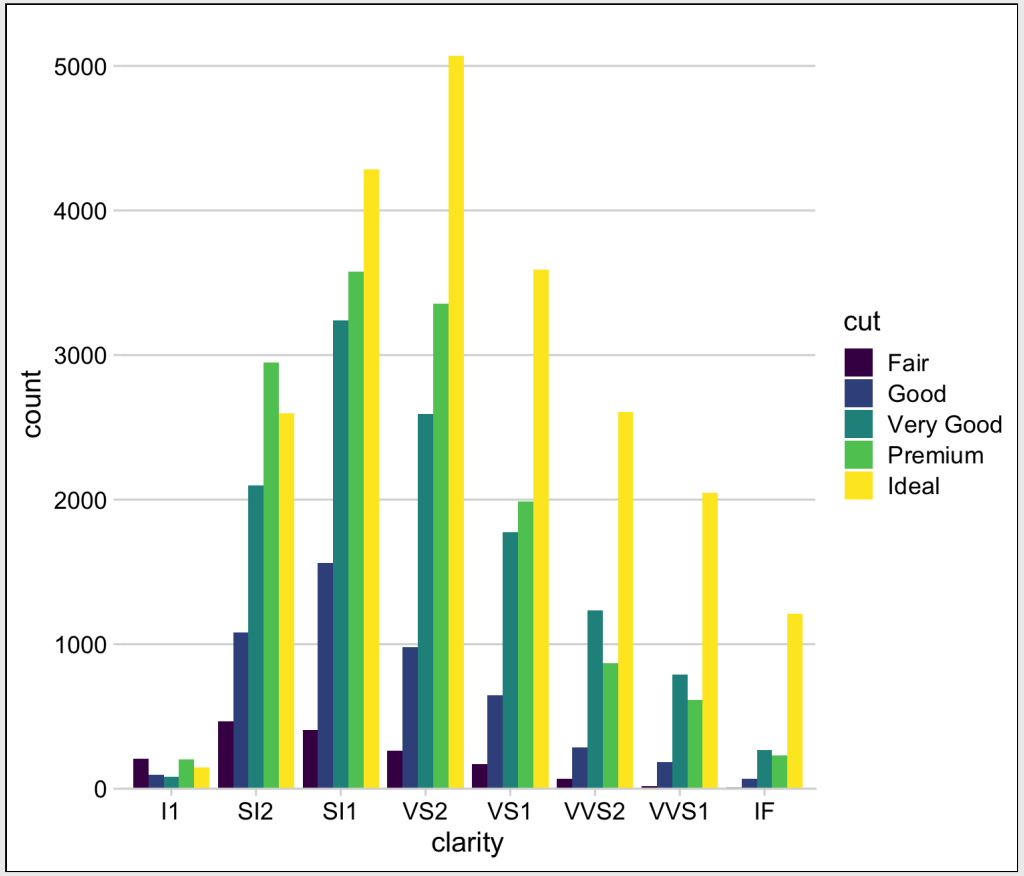


Topics

1. Making a (good) ggplot
2. Using facets
3. Manipulating factors
4. Graphing proportions

When # of categories is large, consider facets

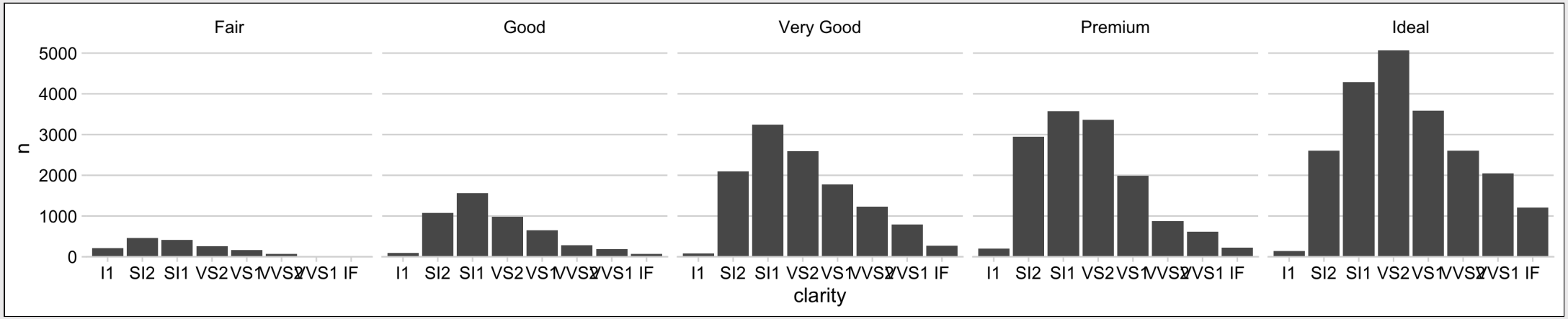




When # of categories is large, consider facets

```
diamonds_summary <- diamonds %>%
  count(clarity, cut)
```

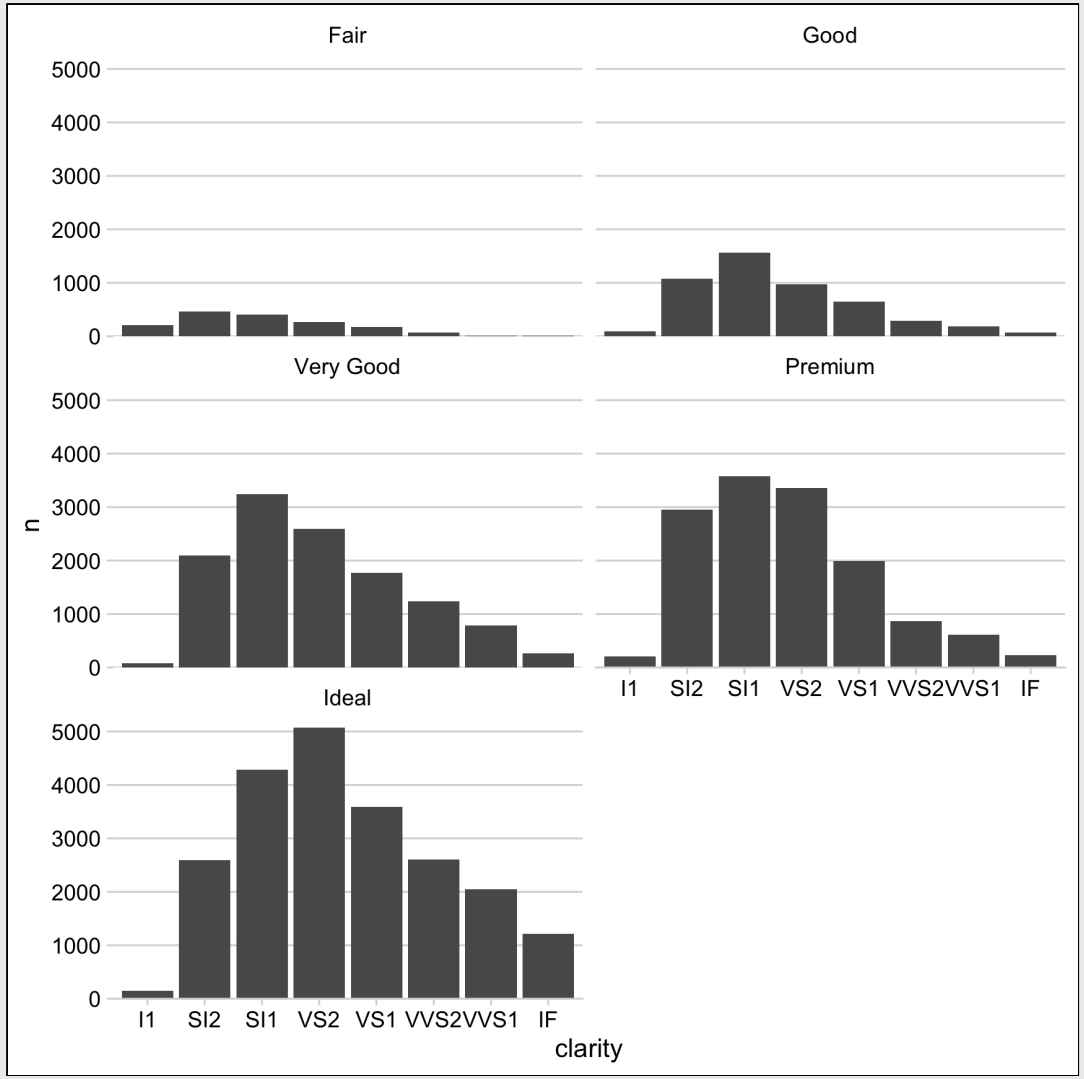
```
ggplot(diamonds_summary) +
  geom_col(aes(x = clarity, y = n)) +
  facet_wrap(~cut, nrow = 1) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_hgrid()
```



When # of categories is large, consider facets

```
diamonds_summary <- diamonds %>%
  count(clarity, cut)
```

```
ggplot(diamonds_summary) +
  geom_col(aes(x = clarity, y = n)) +
  facet_wrap(~cut, ncol = 2) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_hgrid()
```



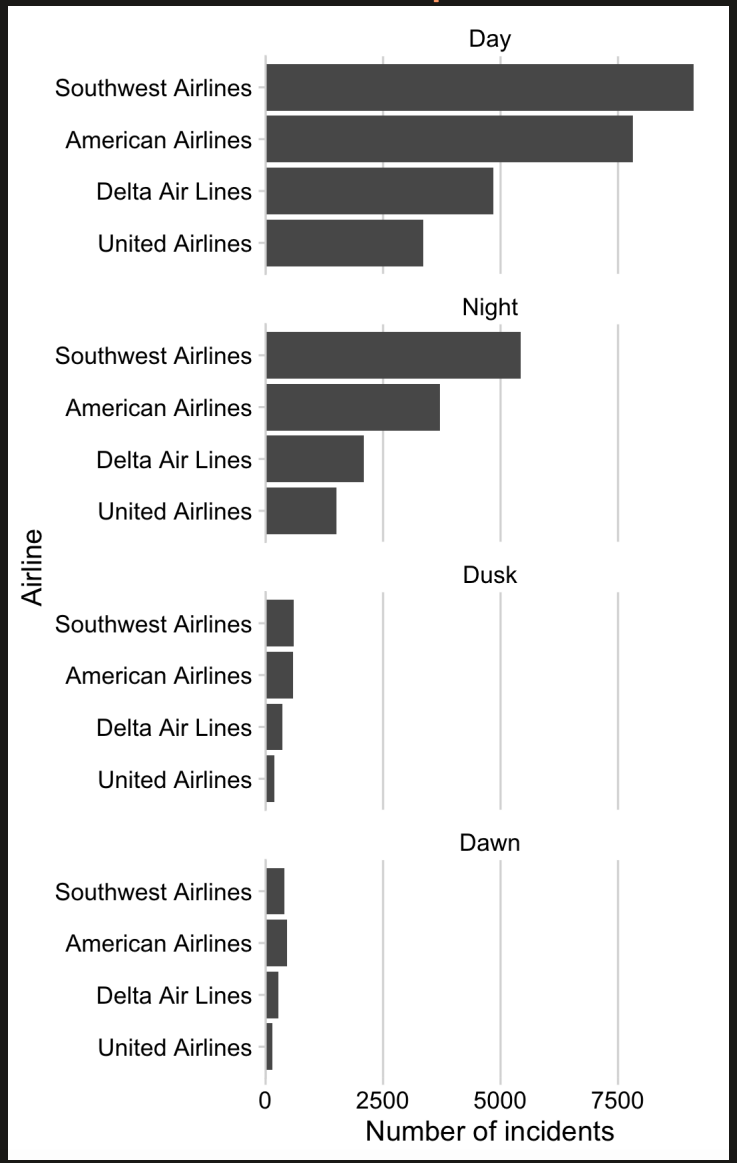
Your turn

Create the following charts. For this exercise, you can remove `NA` values.

Hint: You may need to reshape and / or summarize the data before plotting!

Data: `lotr_words.csv`

Data: `wildlife_impacts.csv`



5 minute break!

Stand up

Move around

Stretch!

Topics

1. Making a (good) ggplot
2. Using facets
3. Manipulating factors
4. Graphing proportions

Common situations:

1. What if the factor order doesn't follow another variable?
2. What if the factors are too long?
3. What if there are too many levels?

1. What if the factor order doesn't follow another variable?

```
lotr_long <- lotr_words %>%
  gather(key = 'gender', value = 'wordCount',
         Female:Male)
```

```
ggplot(lotr_long) +
  geom_col(aes(x = Film, y = wordCount),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```

1. What if the factor order doesn't follow another variable?

Use `fct_relevel()` to manually change the order

```
lotr_long <- lotr_words %>%
  gather(key = 'gender', value = 'wordCount',
         Female:Male) %>%
  mutate(
    Film = fct_relevel(Film, levels = c(
      'The Fellowship Of The Ring',
      'The Two Towers',
      'The Return Of The King')))
```

```
ggplot(lotr_long) +
  geom_col(aes(x = Film, y = wordCount),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```

2. What if the factors are too long?

```
lotr_long <- lotr_words %>%
  gather(key = 'gender', value = 'wordCount',
         Female:Male)
```

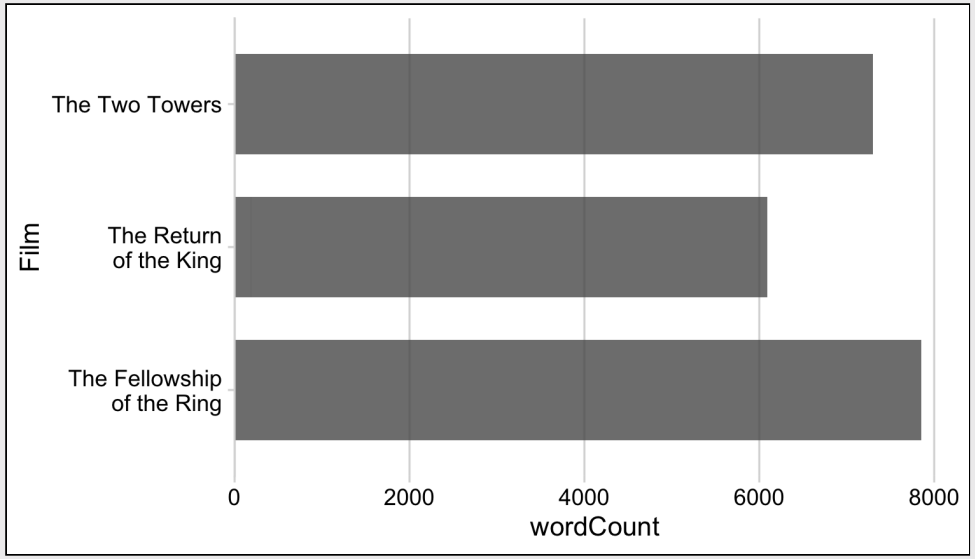
```
ggplot(lotr_long) +
  geom_col(aes(x = Film, y = wordCount),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```

2. What if the factors are too long?

Use `fct_recode()` to manually change the levels

```
lotr_long <- lotr_words %>%
  gather(key = 'gender', value = 'wordCount',
         Female:Male) %>%
  mutate(
    Film = fct_recode(Film,
      'The Fellowship\nof the Ring' = 'The Fellowship Of The Ring',
      'The Return\nof the King' = 'The Return Of The King'))
```

```
ggplot(lotr_long) +
  geom_col(aes(x = Film, y = wordCount),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```



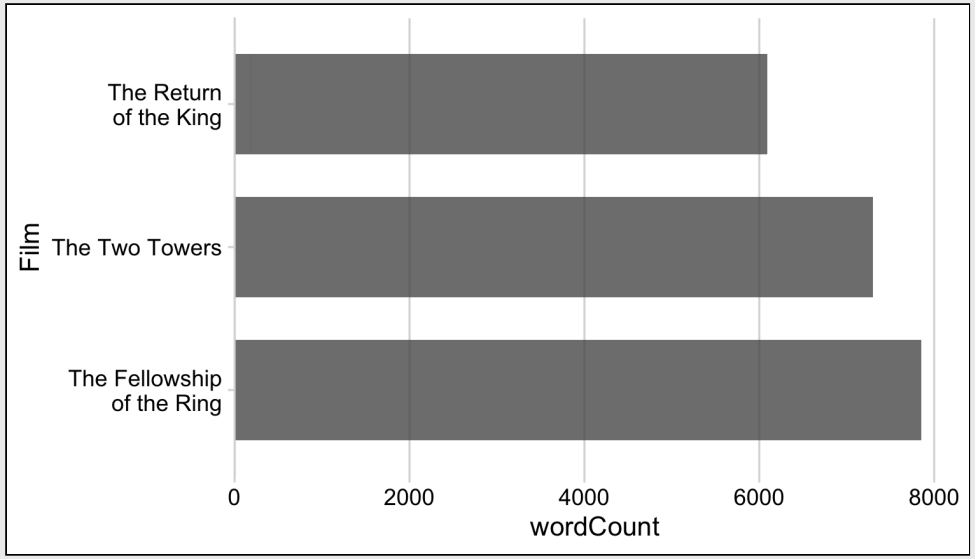
2. What if the factors are too long?

Use `fct_recode()` to manually change the levels

...and don't forget to also update the order

```
lotr_long <- lotr_words %>%
  gather(key = 'gender', value = 'wordCount',
         Female:Male) %>%
  mutate(
    Film = fct_recode(Film,
      'The Fellowship\nof the Ring' = 'The Fellowship Of The Ring',
      'The Return\nof the King' = 'The Return Of The King'),
    Film = fct_relevel(Film,  c(
      'The Fellowship\nof the Ring', 'The Two Towers',
      'The Return\nof the King')))
```

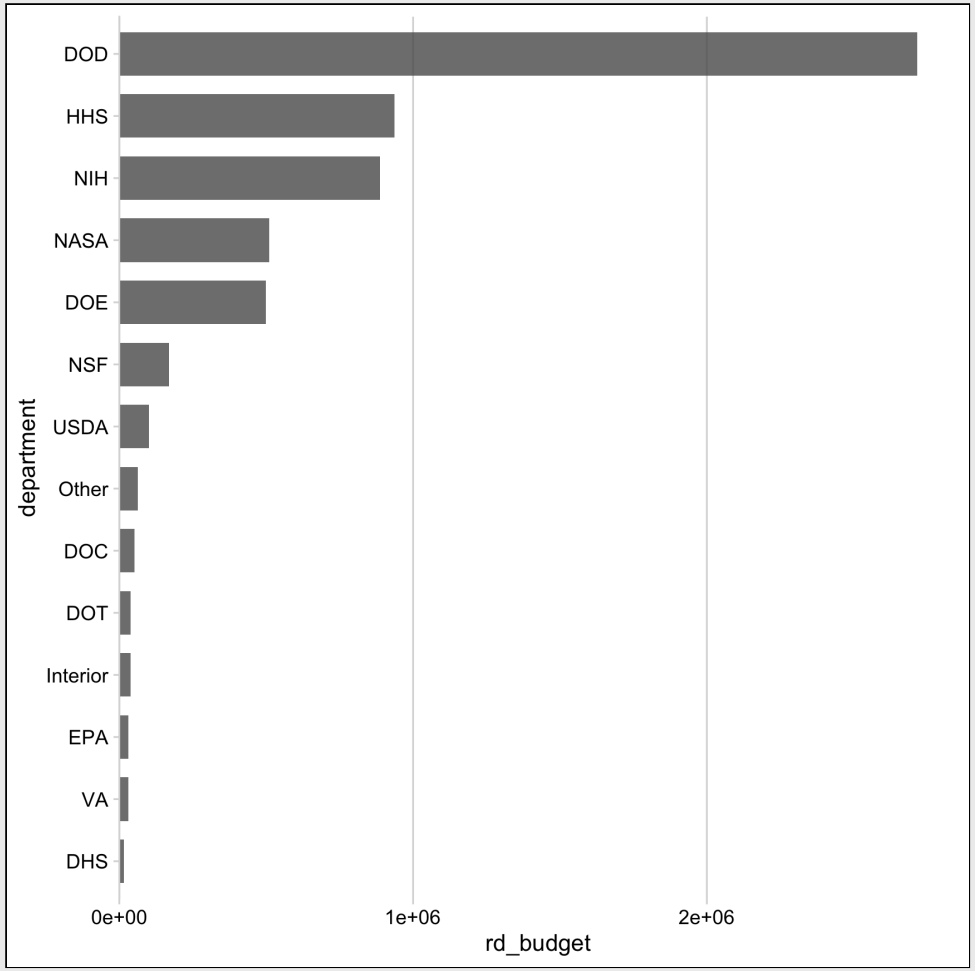
```
ggplot(lotr_long) +
  geom_col(aes(x = Film, y = wordCount),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```



3. What if there are too many levels?

```
federal_spending_summary <- federal_spending %>%
  group_by(department) %>%
  summarise(rd_budget = sum(rd_budget)) %>%
  mutate(
    department = fct_reorder(department, rd_budget))
```

```
ggplot(federal_spending_summary) +
  geom_col(aes(x = department, y = rd_budget),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```

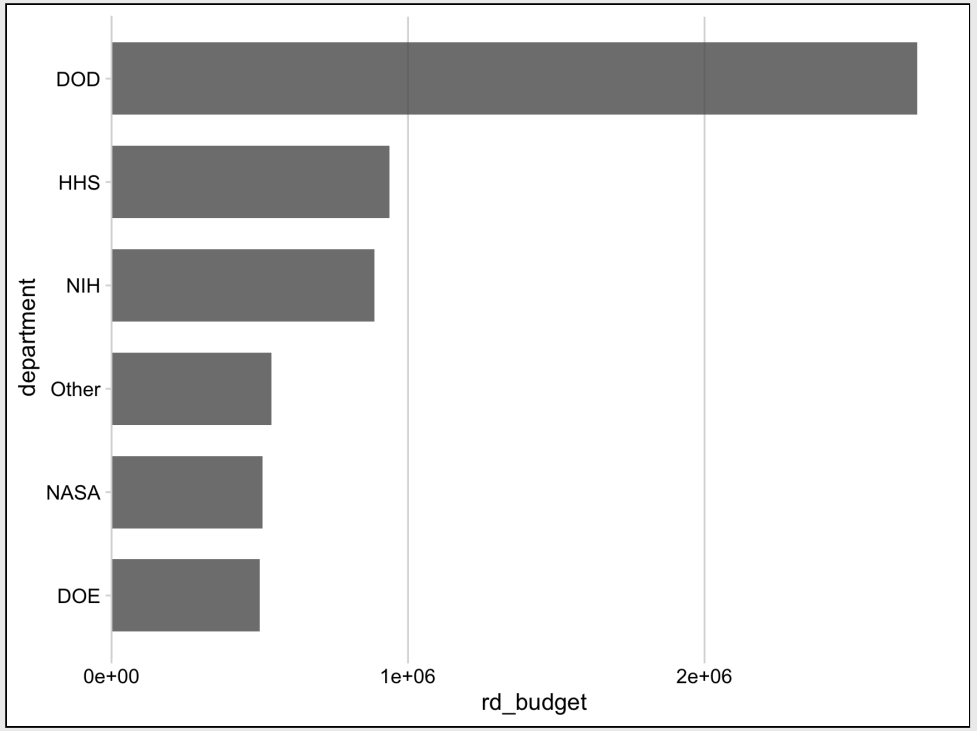


3. What if there are too many levels?

Keep top N, use `fct_other()` to merge rest into "Other"

```
federal_spending_summary <- federal_spending %>%
  mutate(
    department = fct_other(department,
      keep = c('DOD', 'HHS', 'NIH', 'NASA', 'DOE'))) %>%
  group_by(department) %>%
  summarise(rd_budget = sum(rd_budget)) %>%
  mutate(
    department = fct_reorder(department, rd_budget))
```

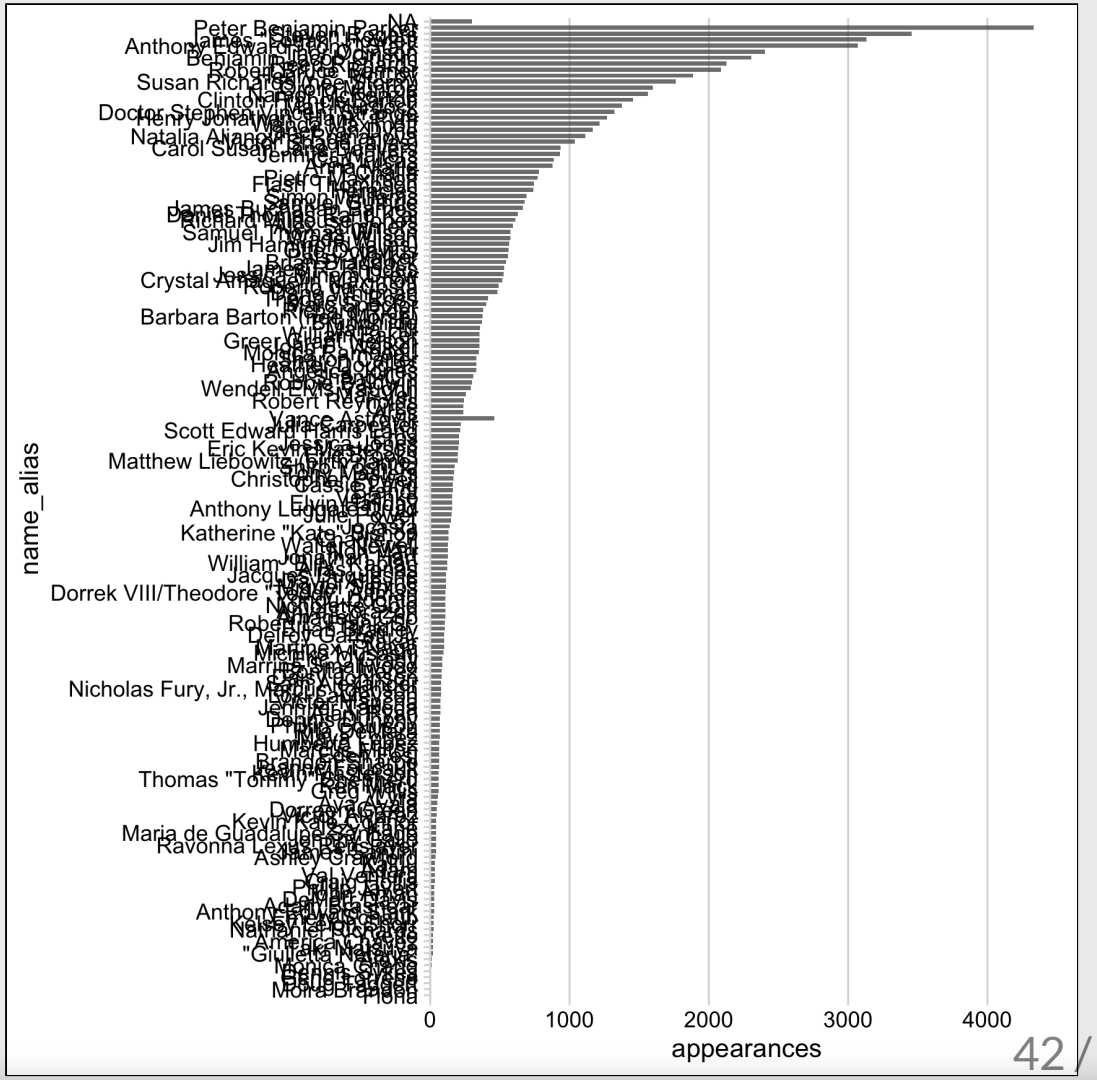
```
ggplot(federal_spending_summary) +
  geom_col(aes(x = department, y = rd_budget),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```



3. What if there are *really* too many levels?

```
avengers <- avengers %>%
  mutate(name_alias =
         fct_reorder(name_alias, appearances))
```

```
ggplot(avengers) +
  geom_col(aes(x = name_alias, y = appearances),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```

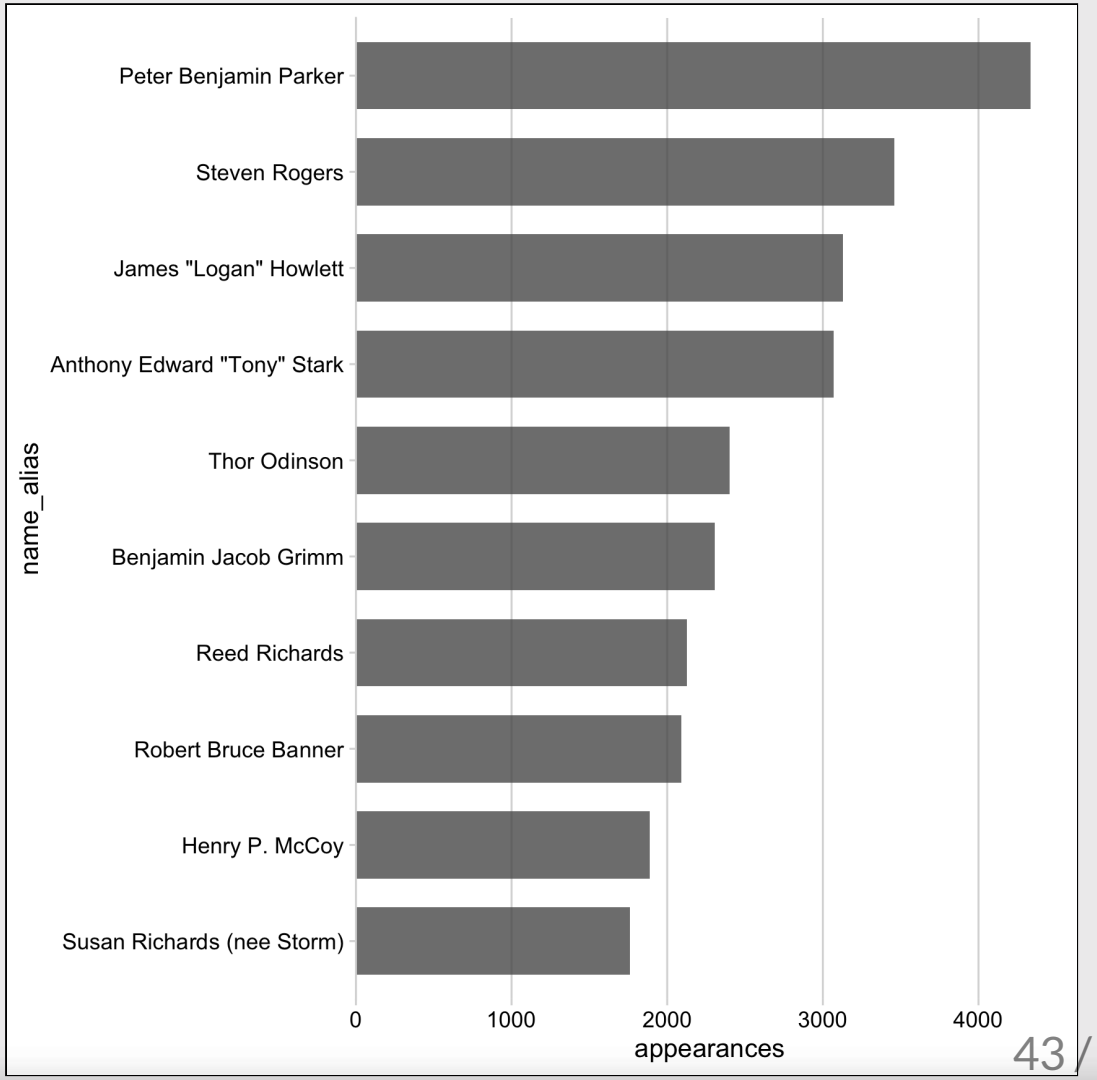


3. What if there are *really* too many levels?

Keep top N, drop the rest

```
avengers_top10 <- avengers %>%
  mutate(name_alias =
           fct_reorder(name_alias, appearances)) %>%
  arrange(desc(appearances)) %>%
  slice(1:10)
```

```
ggplot(avengers_top10) +
  geom_col(aes(x = name_alias, y = appearances),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid()
```

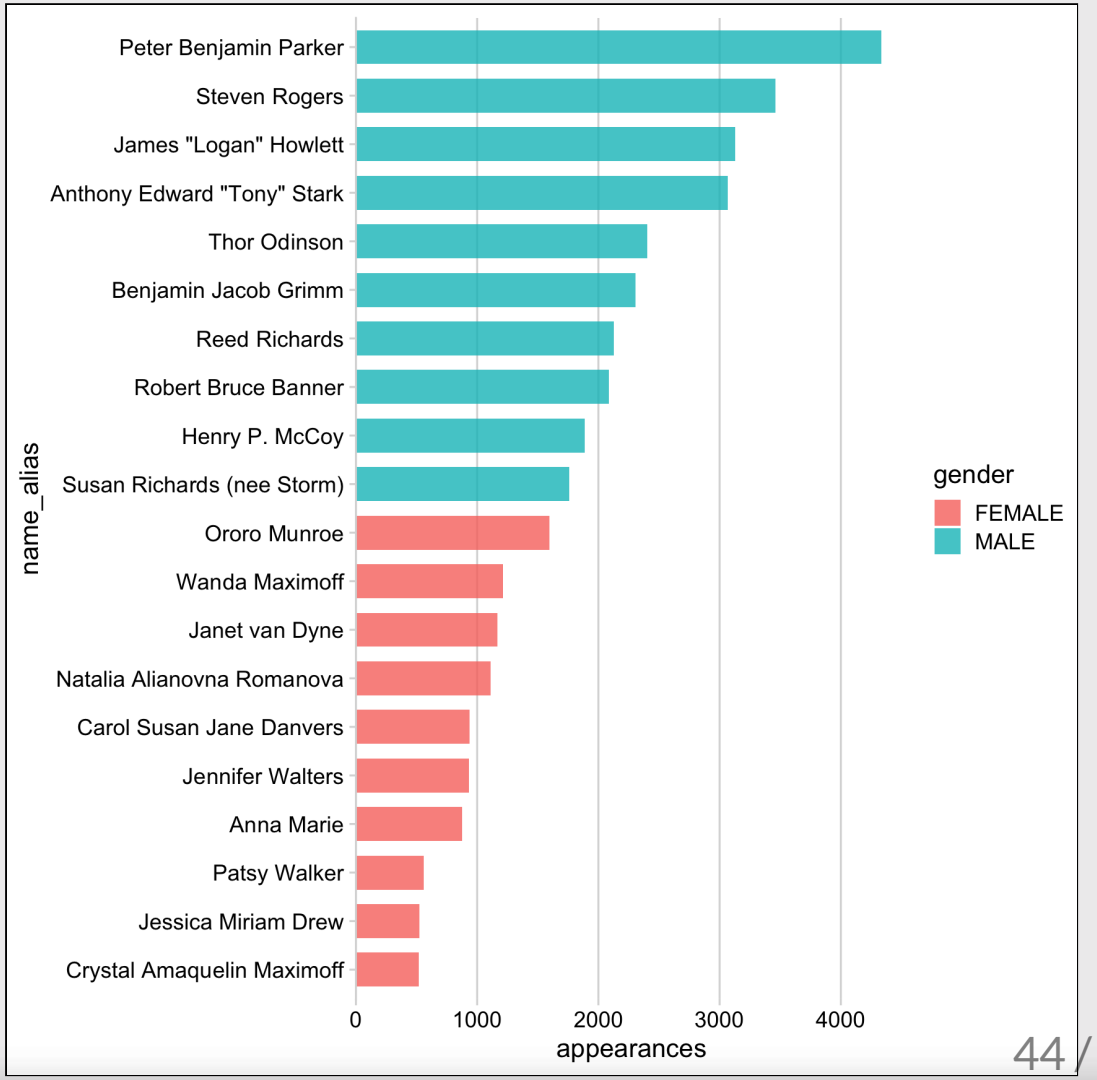


3. What if there are *really* too many levels?

Keep top N, drop the rest

```
avengers_top10 <- avengers %>%
  mutate(name_alias =
           fct_reorder(name_alias, appearances)) %>%
  arrange(desc(appearances)) %>%
  group_by(gender) %>%
  slice(1:10)
```

```
ggplot(avengers_top10) +
  geom_col(aes(x = name_alias, y = appearances,
               fill = gender),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  coord_flip() +
  theme_minimal_vgrid() +
  labs('Top 10 male and female avengers\nin order of a
```

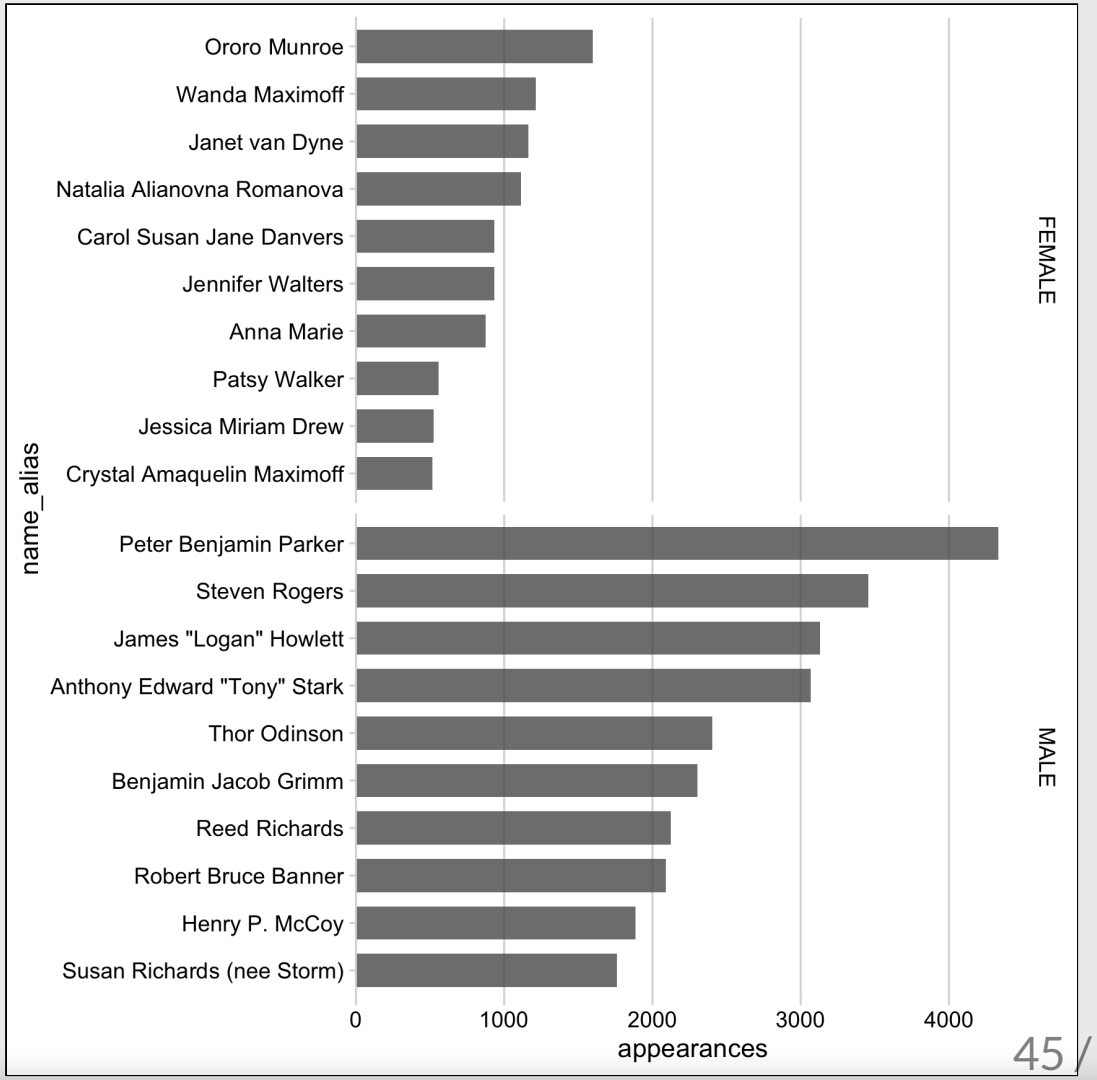


3. What if there are *really* too many levels?

Keep top N, drop the rest

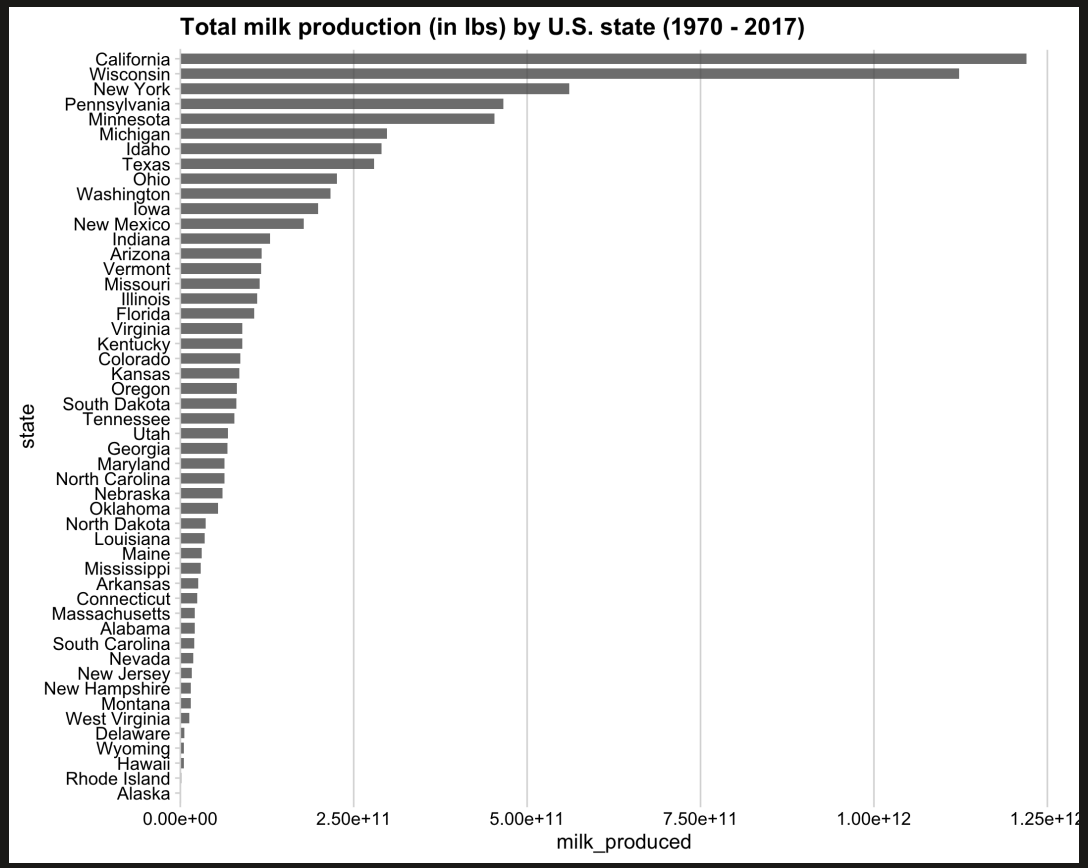
```
avengers_top10 <- avengers %>%
  mutate(name_alias =
         fct_reorder(name_alias, appearances)) %>%
  arrange(desc(appearances)) %>%
  group_by(gender) %>%
  slice(1:10)
```

```
ggplot(avengers_top10) +
  geom_col(aes(x = name_alias, y = appearances),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  facet_wrap(~gender, ncol = 1,
             scales = 'free_y',
             strip.position = 'right') +
  coord_flip() +
  theme_minimal_vgrid()
```

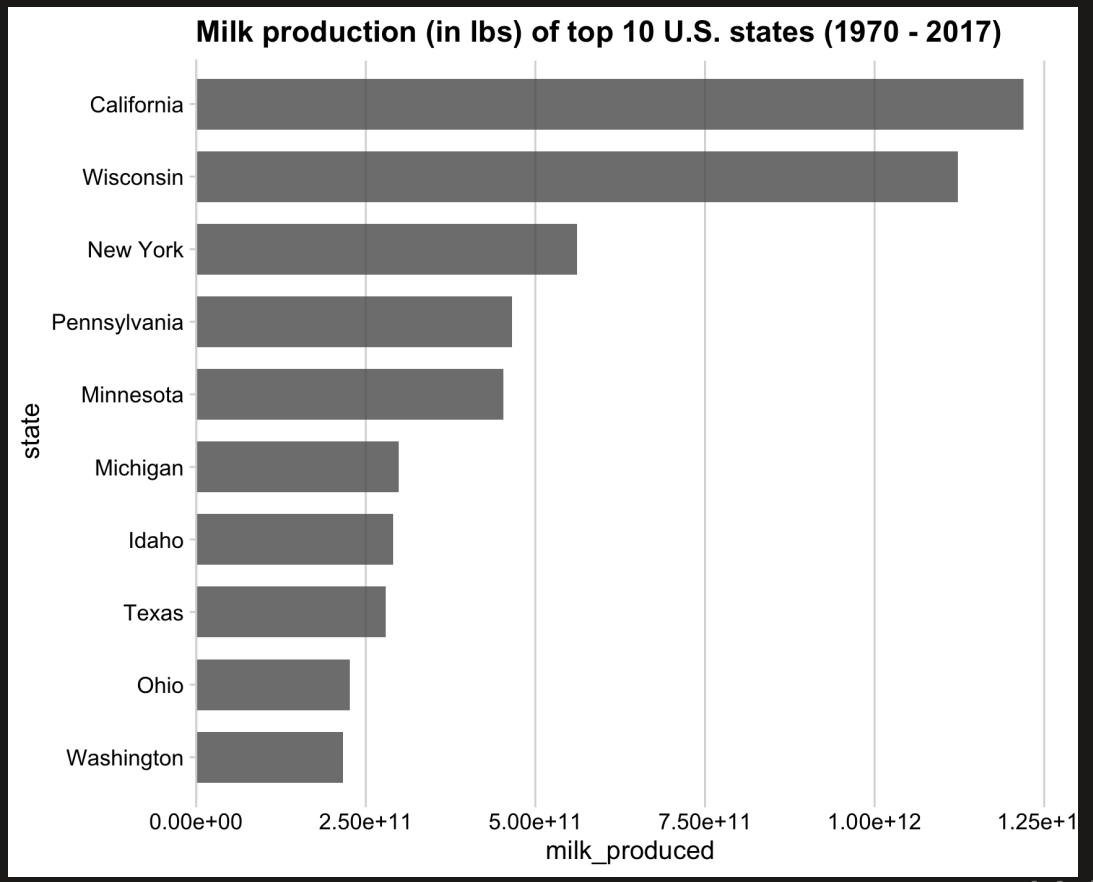


Your turn

Use the `milk_production.csv` data to create the following plots

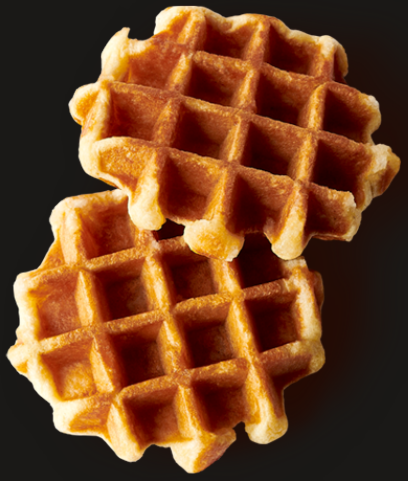


Hint: You'll need to use `summarise()` to get the total milk production for each state between 1970 - 2017



Topics

1. Making a (good) ggplot
2. Using facets
3. Manipulating factors
4. Graphing proportions



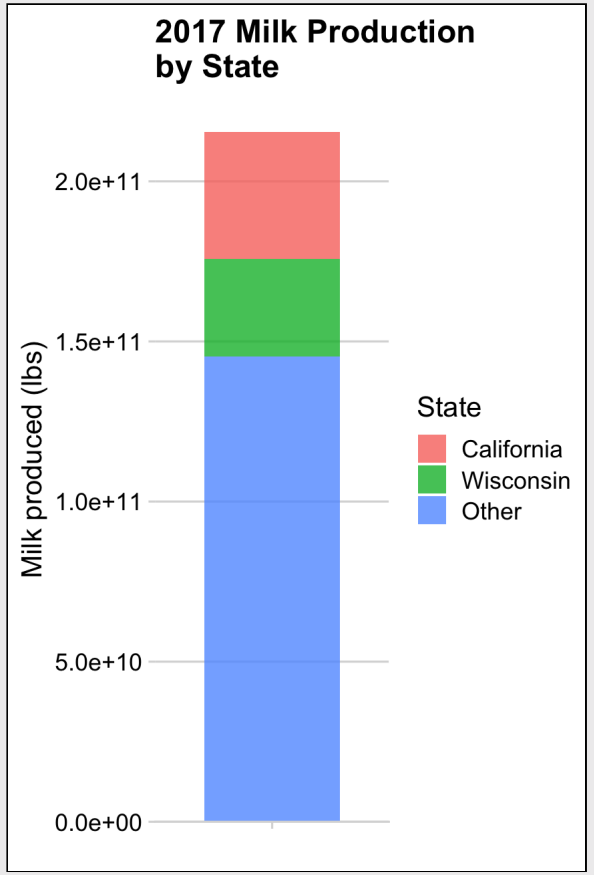
Stacked bars

Summarize data:

```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced))
```

Make plot:

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = "", y = milk_produced, fill = state),
           width = 0.7, alpha = 0.8) +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_hgrid() +
  labs(x = NULL,
       y = 'Milk produced (lbs)',
       fill = 'State',
       title = '2017 Milk Production\nby State')
```

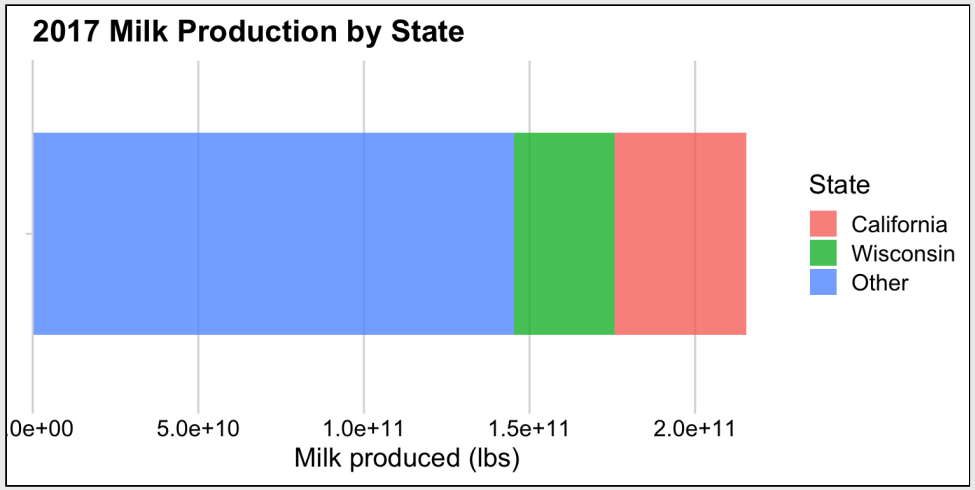


Stacked bars

Rotated also looks good

```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced))
```

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = "", y = milk_produced, fill = state),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid() +
  labs(x = NULL,
       y = 'Milk produced (lbs)',
       fill = 'State',
       title = '2017 Milk Production by State')
```

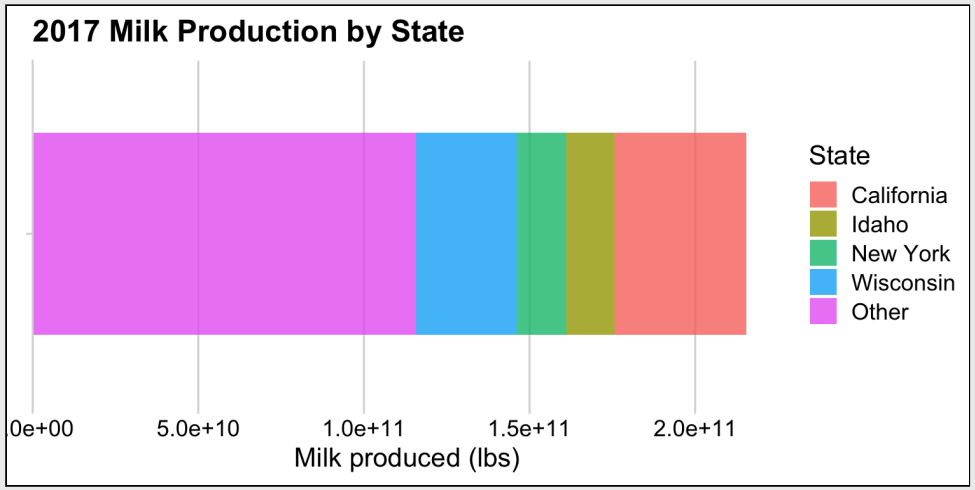


Stacked bars

Not great for more than a few categories

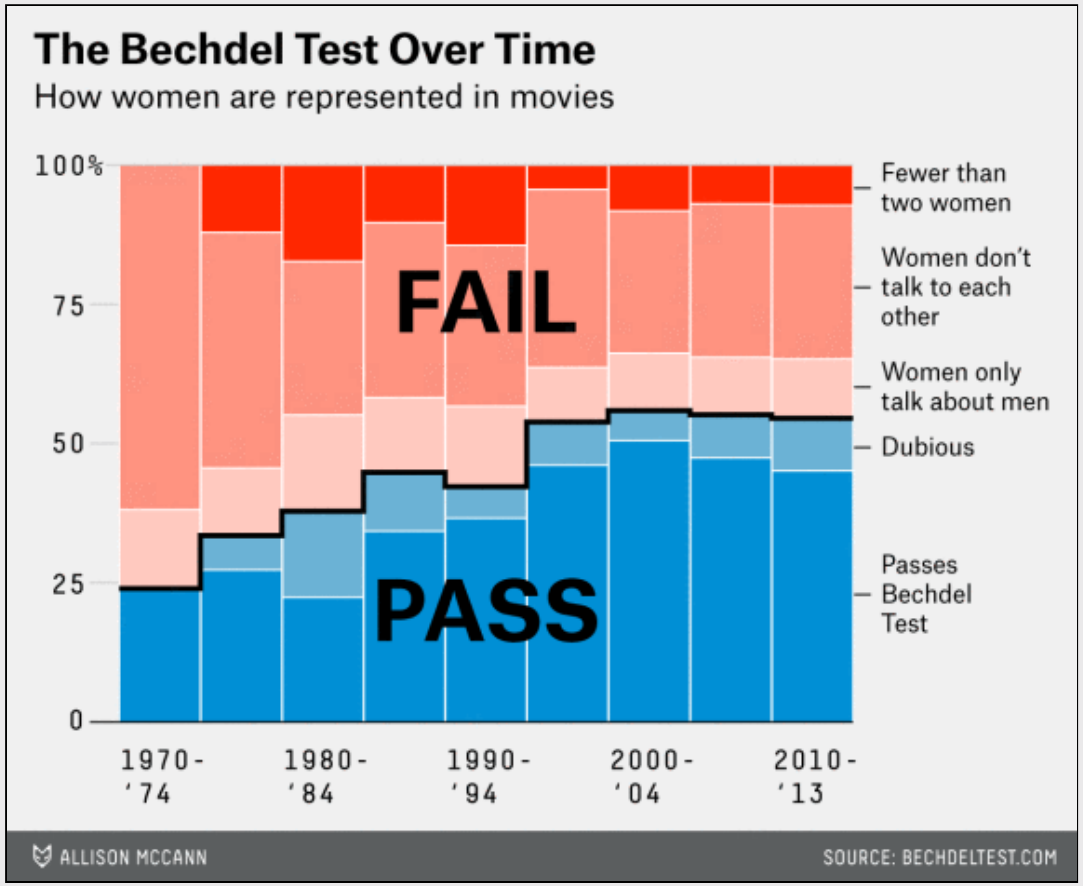
```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin',
             'New York', 'Idaho'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced))
```

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = "", y = milk_produced, fill = state),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid() +
  labs(x = NULL,
       y = 'Milk produced (lbs)',
       fill = 'State',
       title = '2017 Milk Production by State')
```



Where stacking is useful

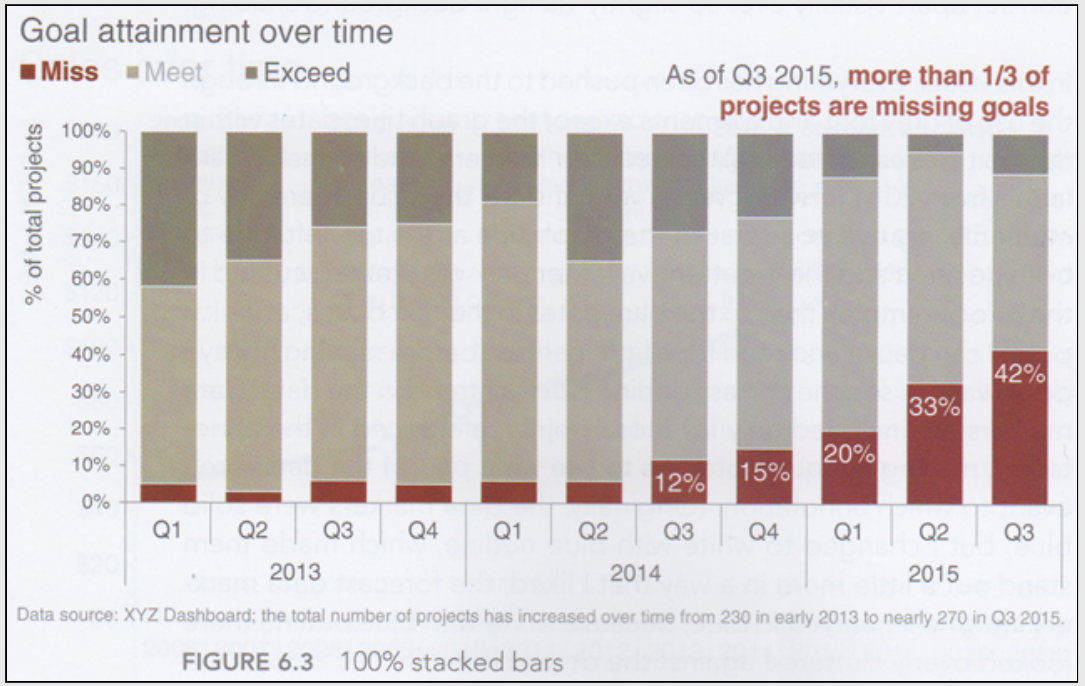
- **2 - 3 groups**
- Proportions over time



https://fivethirtyeight.com/features/the-dollar-and-cents-case-against-hollywoods-exclusion-of-women/

Where stacking is useful

- 2 - 3 groups
- **Proportions over time**



https://www.perceptualedge.com/blog/?p=2239

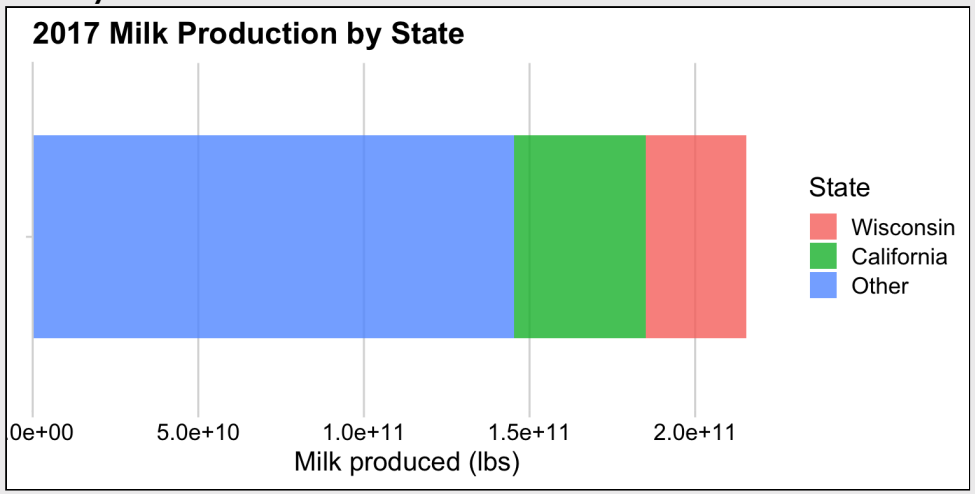
Dodged bars

Better for showing **single part-to-whole comparison**

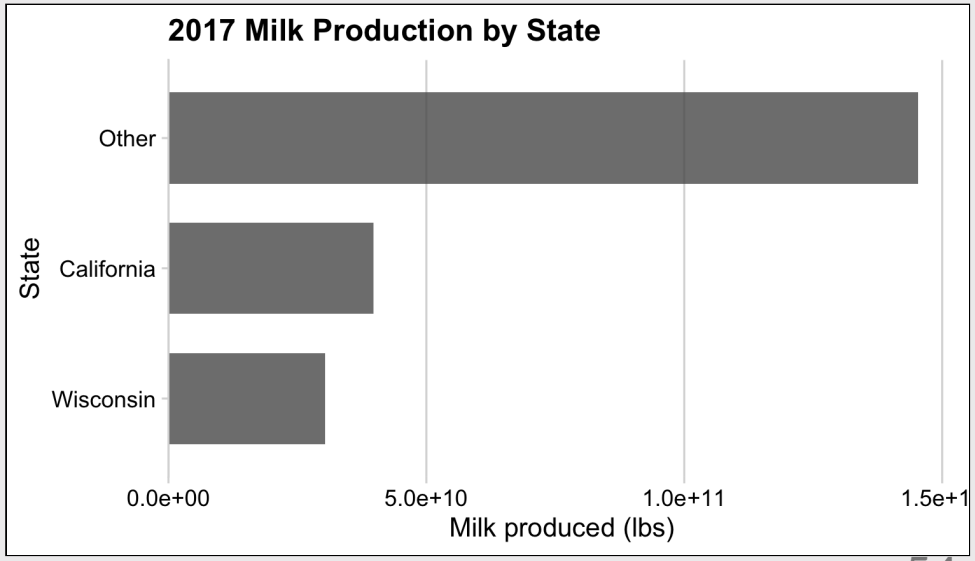
```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced)) %>%
  mutate(state = fct_reorder(state, milk_produced))
```

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = state, y = milk_produced),
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid() +
  labs(x = 'State',
       y = 'Milk produced (lbs)',
       title = '2017 Milk Production by State')
```

Okay:



Better:



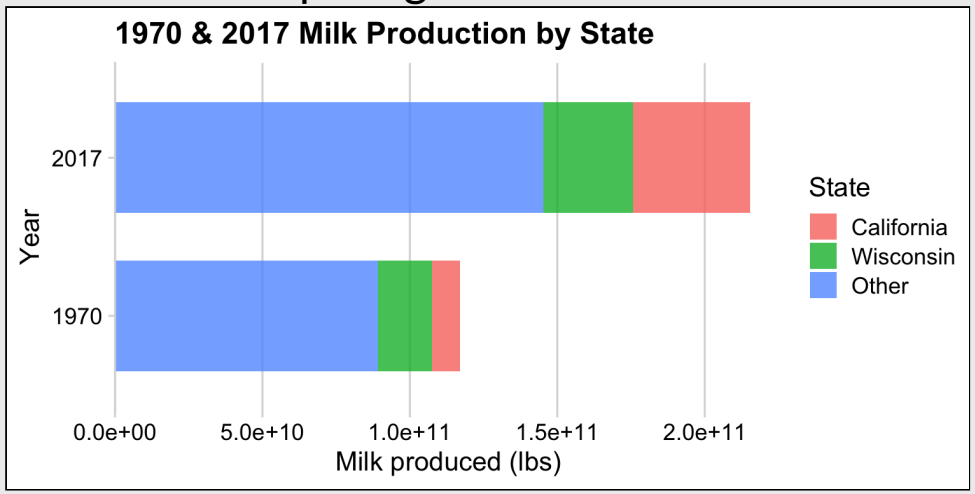
Dodged bars

Better for **comparing individual components**

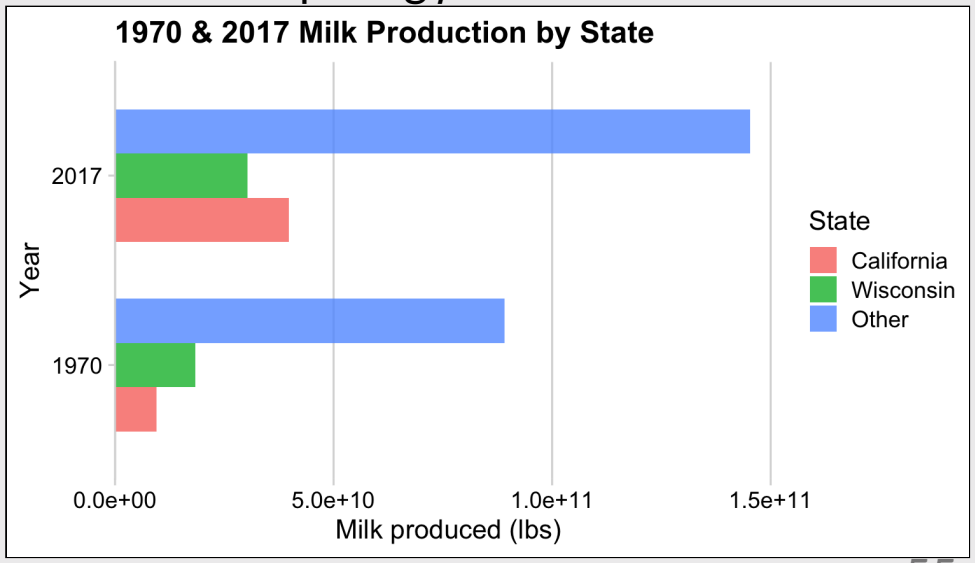
```
milk_summary_2017 <- milk_production %>%
  filter(year %in% c(1970, 2017)) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(year, state) %>%
  summarise(milk_produced = sum(milk_produced))
```

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = as.factor(year),
               y = milk_produced, fill = state),
           position = 'dodge',
           width = 0.7, alpha = 0.8) +
  coord_flip() +
  scale_y_continuous(
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid() +
  labs(x = 'Year',
       y = 'Milk produced (lbs)',
       fill = 'State',
       title = '1970 & 2017 Milk Production by State')
```

Better for comparing *total*:



Better for comparing *parts*:

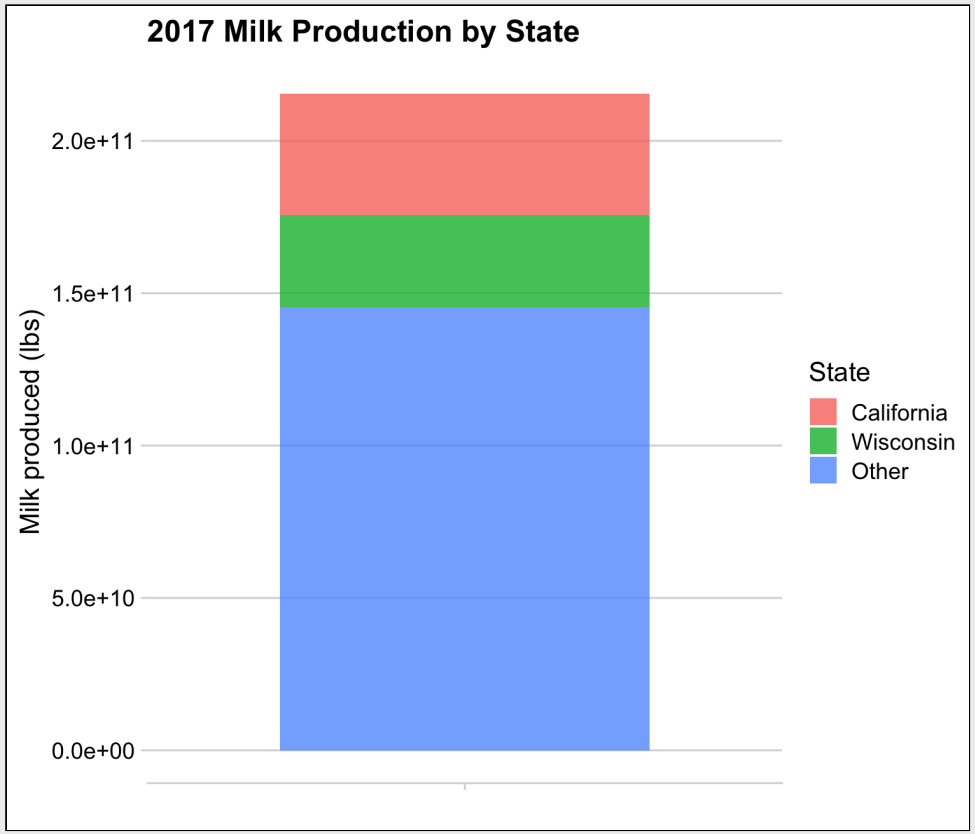


The Notorious P-I-E

Start with a bar chart

```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced))
```

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = "", y = milk_produced, fill = state),
           width = 0.7, alpha = 0.8) +
  theme_minimal_hgrid() +
  labs(x = NULL,
       y = 'Milk produced (lbs)',
       fill = 'State',
       title = '2017 Milk Production by State')
```

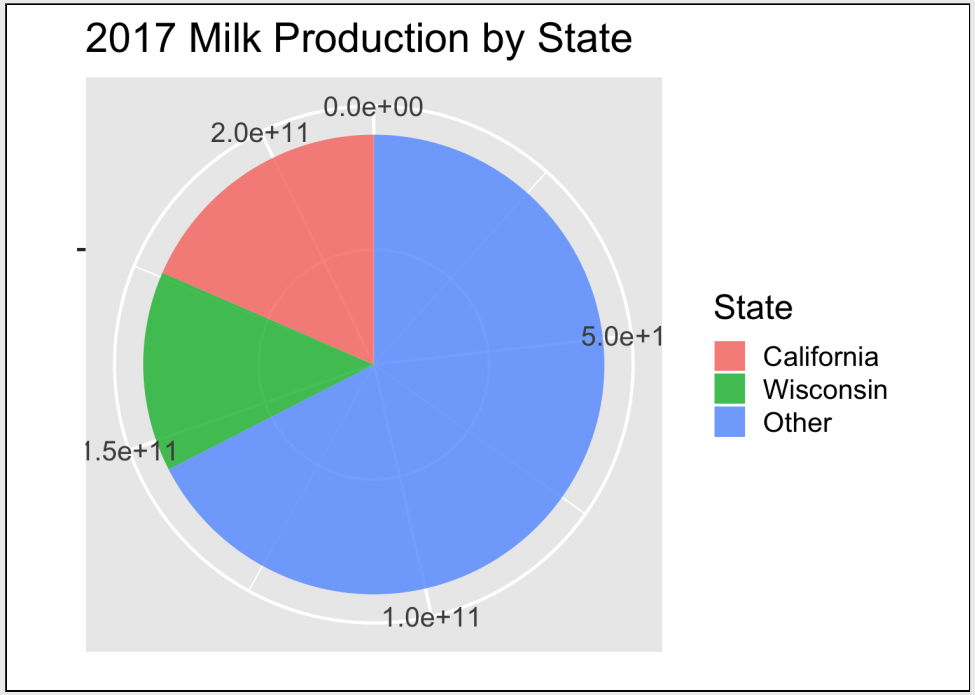


The Notorious P-I-E

Convert bar to pie

```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced))
```

```
ggplot(milk_summary_2017) +
  geom_col(aes(x = "", y = milk_produced, fill = state),
           width = 0.7, alpha = 0.8) +
  coord_polar(theta = "y") +
  labs(x = NULL,
       y = NULL,
       fill = 'State',
       title = '2017 Milk Production by State')
```

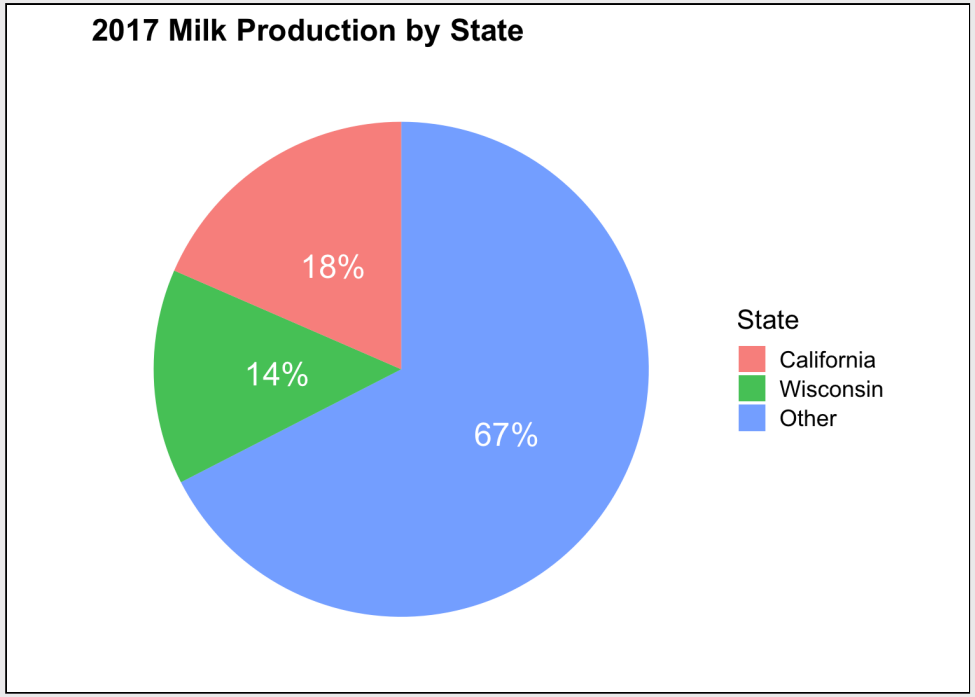


The Notorious P-I-E

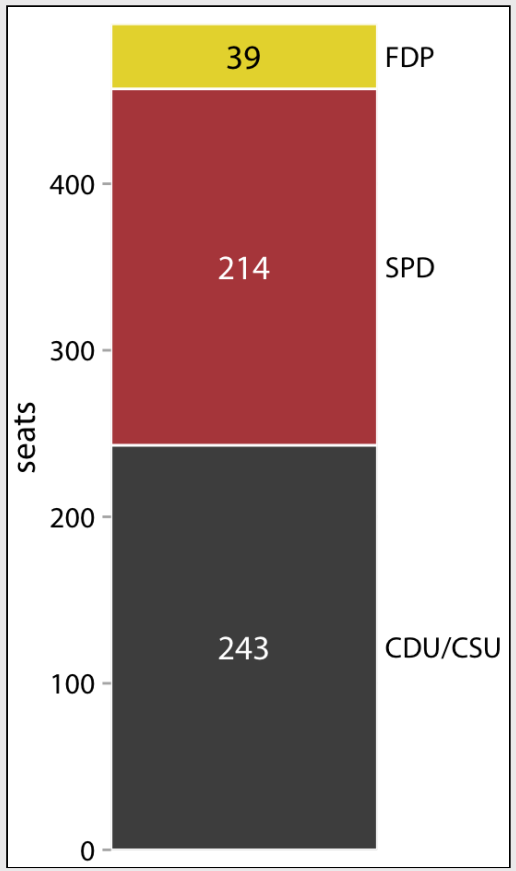
Add labels & apply `theme_map()`

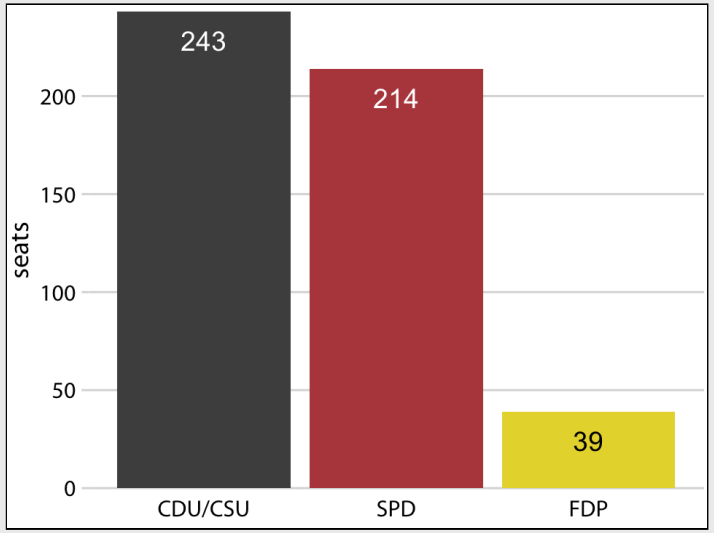
```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced)) %>%
  arrange(desc(state)) %>%
  mutate(p = 100*(milk_produced / sum(milk_produced)),
         label = str_c(round(p), '%'))
```

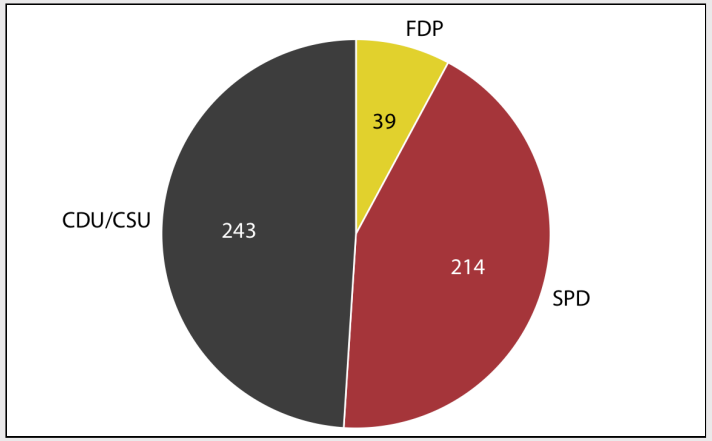
```
ggplot(milk_summary_2017) +
  geom_col(aes(x = "", y = milk_produced, fill = state),
           width = 0.7, alpha = 0.8) +
  coord_polar(theta = "y") +
  geom_text(aes(x = "", y = milk_produced, label = label),
            color = "white", size = 6,
            position = position_stack(vjust = 0.5)) +
  theme_map() +
  labs(x = NULL,
       y = NULL,
       fill = 'State',
       title = '2017 Milk Production by State')
```



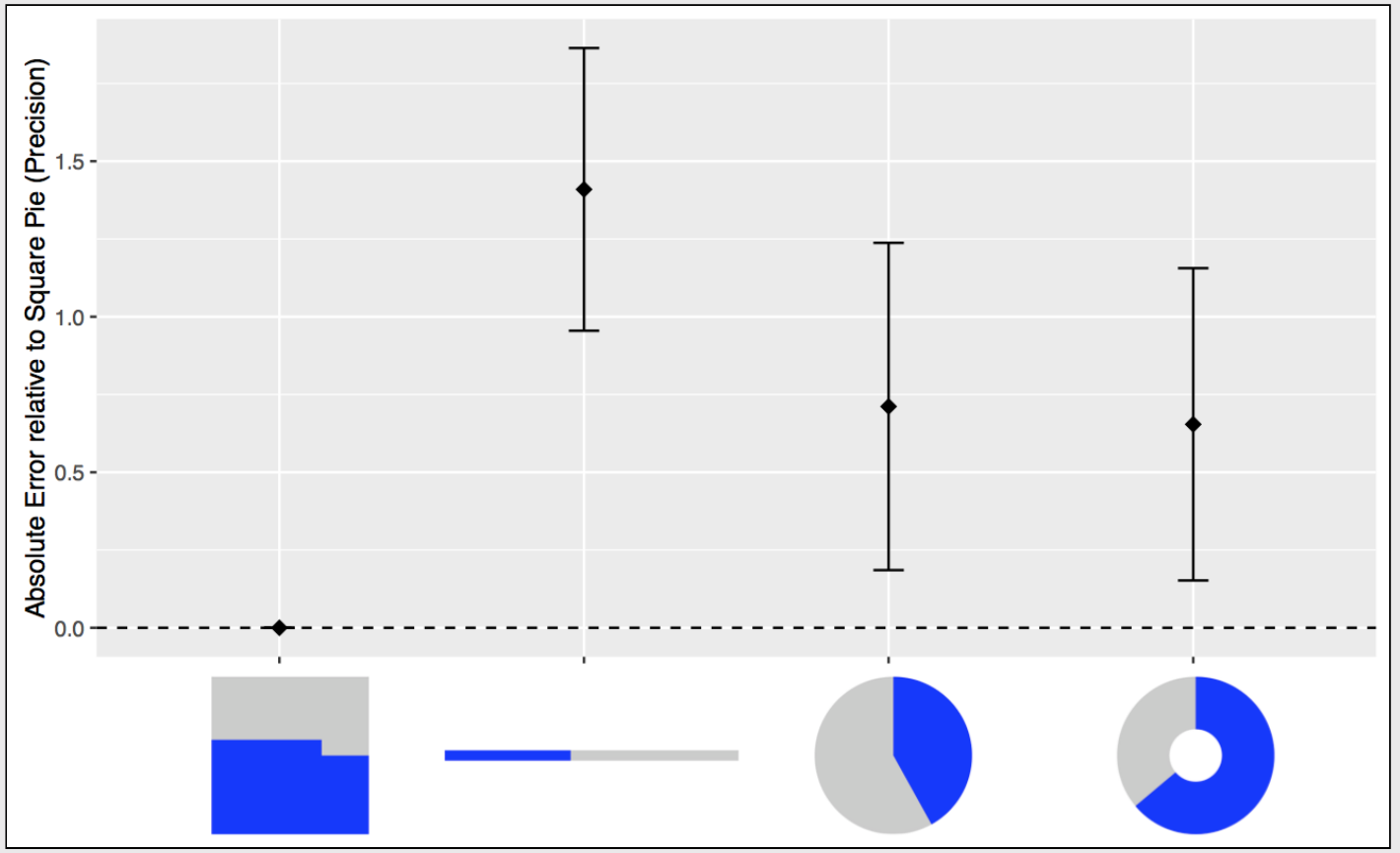
Where pies are useful







The best pies are **square pies**



https://eagereyes.org/blog/2016/a-reanalysis-of-a-study-about-square-pie-charts-from-2009]

Waffle plots

Summarize data, then plot:

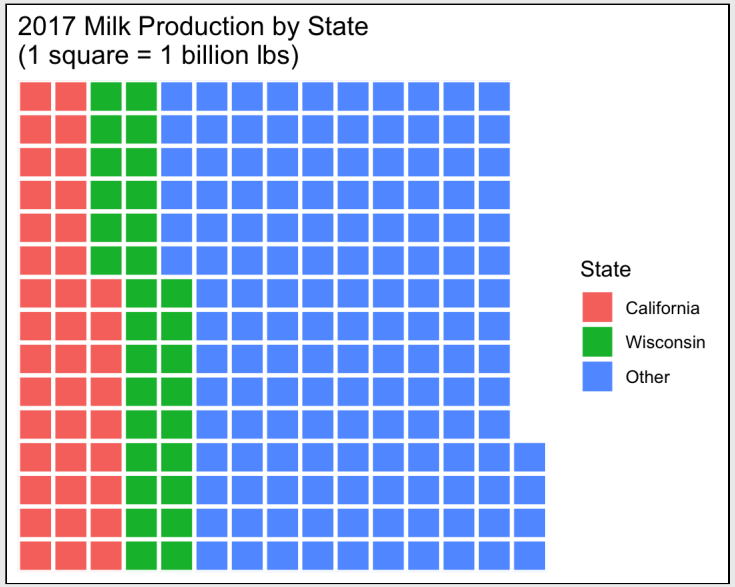
```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced)) %>%
  mutate(milk_produced = milk_produced / 10^9)
```

```
library(waffle)

ggplot(milk_summary_2017) +
  geom_waffle(aes(fill = state, values = milk_produced),
              color = "white", size = 1, n_rows = 15) +
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0)) +
  theme_minimal() +
  labs(fill = 'State',
       x = NULL,
       y = NULL,
       title = str_c('2017 Milk Production by State\n',
                     '(1 square = 1 billion lbs)'))
```

Use values between 100 - 1,000

(You don't want 1,000,000,000 boxes!)

```
## # A tibble: 3 x 2
##   state      milk_produced
##   <fct>              <dbl>
## 1 California          39.8
## 2 Wisconsin           30.3
## 3 Other              145.
```



Waffle plots

Summarize data, then plot:

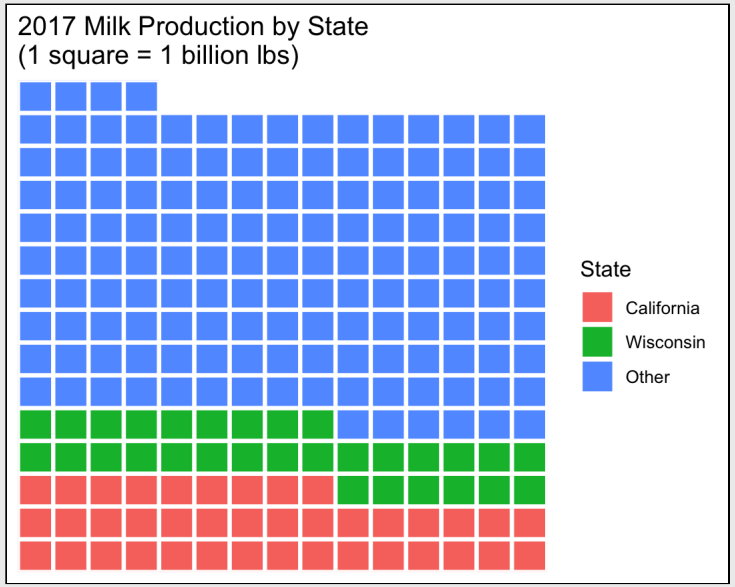
```
milk_summary_2017 <- milk_production %>%
  filter(year == 2017) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(state) %>%
  summarise(milk_produced = sum(milk_produced)) %>%
  mutate(milk_produced = milk_produced / 10^9)
```

```
library(waffle)

ggplot(milk_summary_2017) +
  geom_waffle(aes(fill = state, values = milk_produced),
              color = "white", size = 1, n_rows = 15,
              flip = TRUE) +
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0)) +
  theme_minimal() +
  labs(fill = 'State',
       x = NULL,
       y = NULL,
       title = str_c('2017 Milk Production by State\n',
                     '(1 square = 1 billion lbs)'))
```

Use values between 100 - 1,000

(You don't want 1,000,000,000 boxes!)

```
## # A tibble: 3 x 2
##   state      milk_produced
##   <fct>              <dbl>
## 1 California          39.8
## 2 Wisconsin           30.3
## 3 Other              145.
```



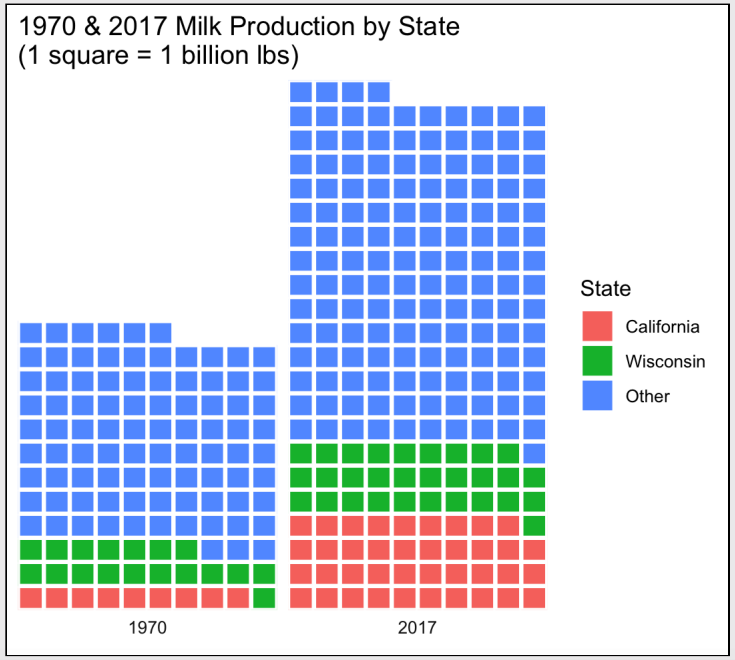
Waffle comparison

Summarize data, then plot:

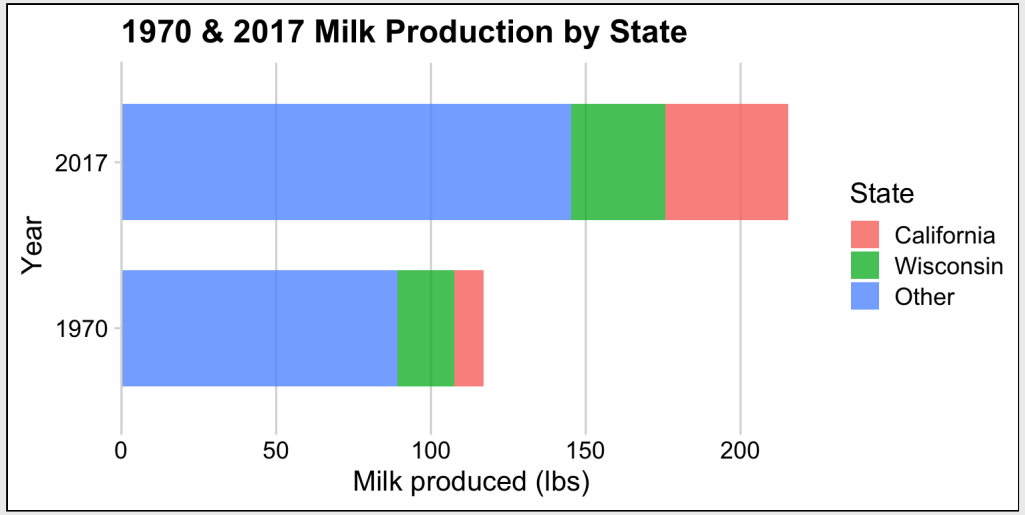
```
milk_summary_2017 <- milk_production %>%
  filter(year %in% c(1970, 2017)) %>%
  mutate(state = fct_other(state,
    keep = c('California', 'Wisconsin'))) %>%
  group_by(year, state) %>%
  summarise(milk_produced = sum(milk_produced)) %>%
  mutate(milk_produced = milk_produced / 10^9)
```

```
library(waffle)

ggplot(milk_summary_2017) +
  geom_waffle(aes(fill = state, values = milk_produced),
              color = "white", size = 1, n_rows = 10,
              flip = TRUE) +
  facet_wrap(~year, strip.position = 'bottom') +
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0)) +
  theme_minimal() +
  labs(fill = 'State',
       x = NULL,
       y = NULL,
       title = str_c('1970 & 2017 Milk Production by State\n
                     '(1 square = 1 billion lbs)'))
```

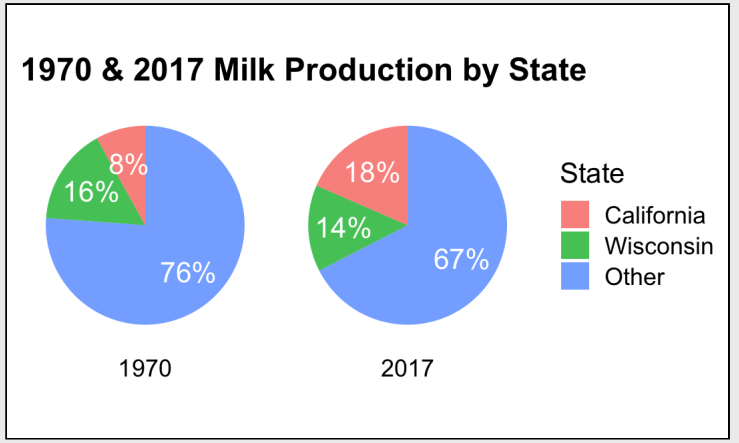
```
## # A tibble: 6 x 3
## # Groups:   year [2]
##    year state      milk_produced
##   <dbl> <fct>              <dbl>
## 1  1970 California          9.46
## 2  1970 Wisconsin          18.4
## 3  1970 Other              89.1
## 4  2017 California         39.8
## 5  2017 Wisconsin          30.3
## 6  2017 Other             145.
```



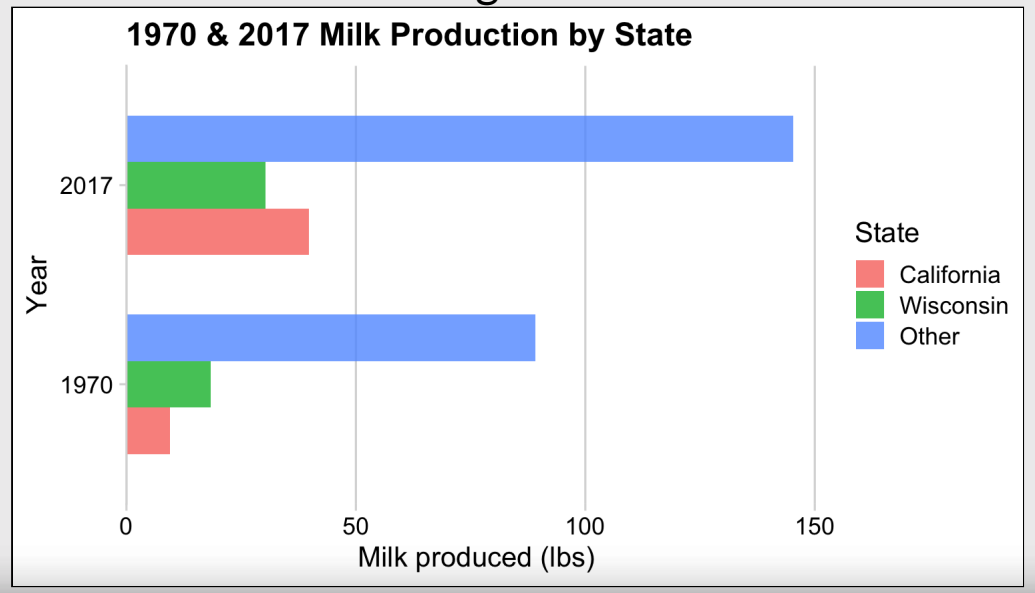
Stacked bars



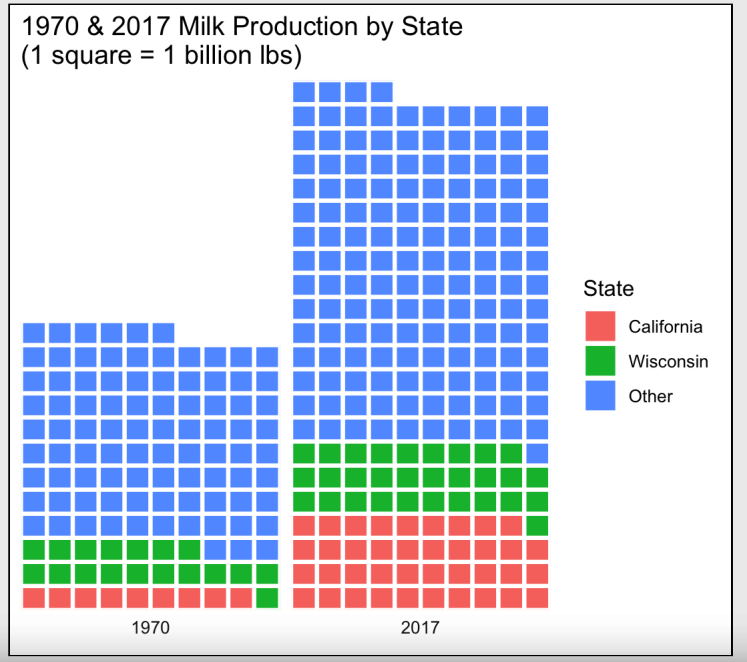
Pie chart



Dodged bars



Waffle chart



Your turn

Using the `wildlife_impacts` data, create plots that shows the proportion of incidents that occur at each different time of day.

For this exercise, you can remove `NA` values.

Try to create the following plots:

- Stacked bars
- Dodged bars
- Pie chart
- Waffle chart

To get started, you'll need to first summarize the data:

```
wildlife_summary <- wildlife_impacts %>%
  filter(!is.na(time_of_day)) %>%
  count(time_of_day)

wildlife_summary
```

```
## # A tibble: 4 x 2
##   time_of_day     n
##   <chr>       <int>
## 1 Dawn         1270
## 2 Day         25123
## 3 Dusk         1717
## 4 Night       12735
```