Week 8: Polishing your charts

EMSE 4197 | John Paul Helveston | March 04, 2020

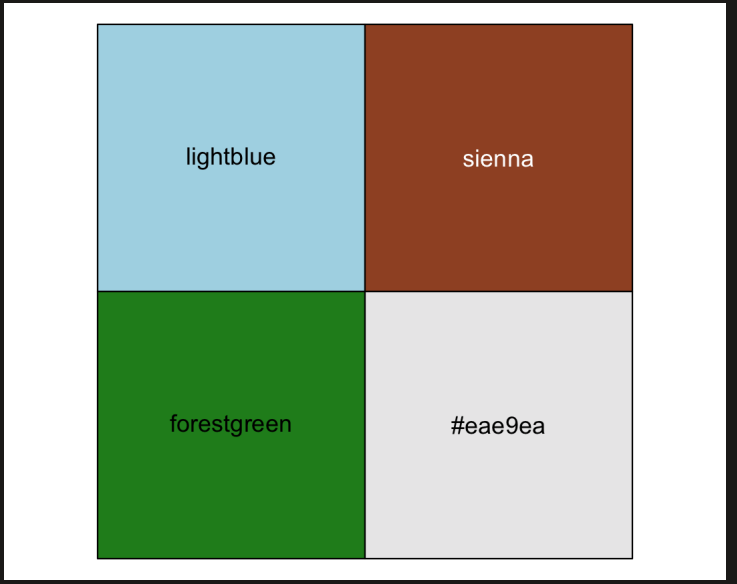
R tip of the week

Ever wondered what colors look like *before* you plot them?

Use `scales::show_col()` to preview them

Example:

```
colors <- c('lightblue', 'sienna', 'forestgreen', '#eae9ea')
scales::show_col(colors)
```



Today's data

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

New package:

```
install.packages('hrbrthemes')
```

Polishing your charts

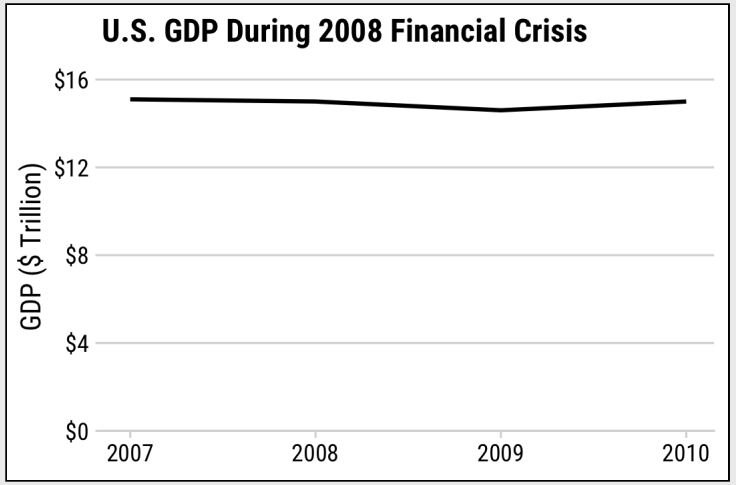
1. Scales
2. Annotations
3. Colors
4. Fonts
5. Rmd tricks

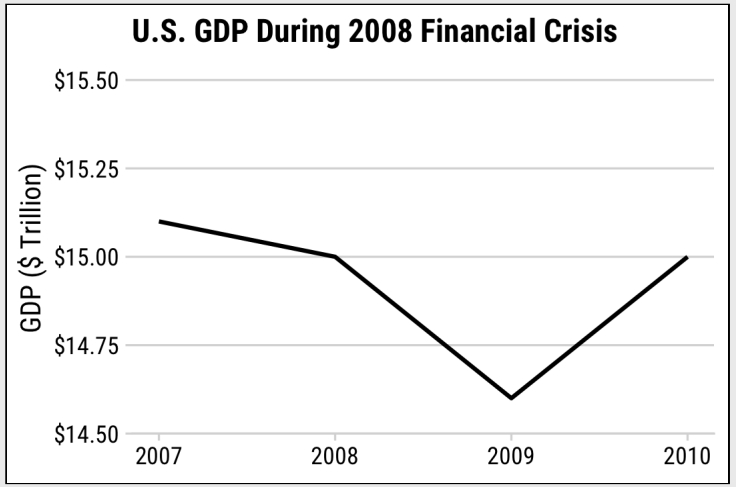
Polishing your charts

1. Scales
2. Annotations
3. Colors
4. Fonts
5. Rmd tricks

When is it okay to to truncate an axis?

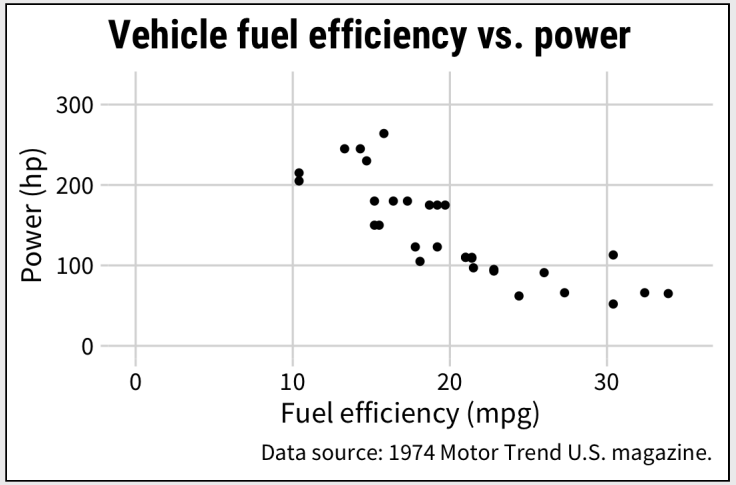
- When small movements matter

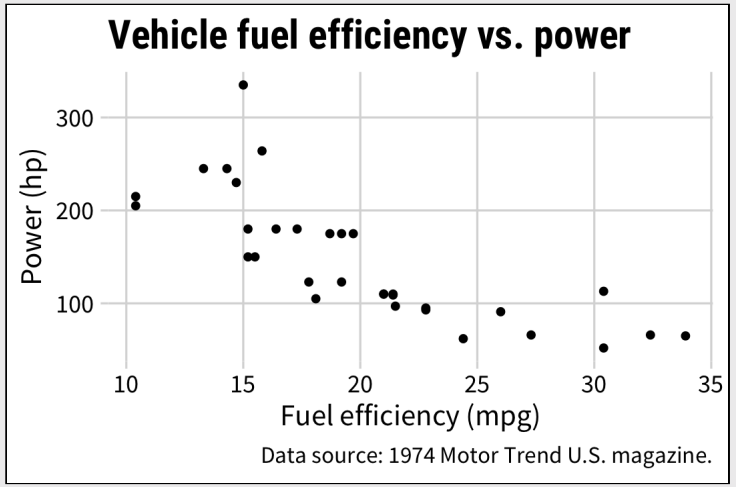




When is it okay to to truncate an axis?

- When small movements matter
- When zero values are impossible



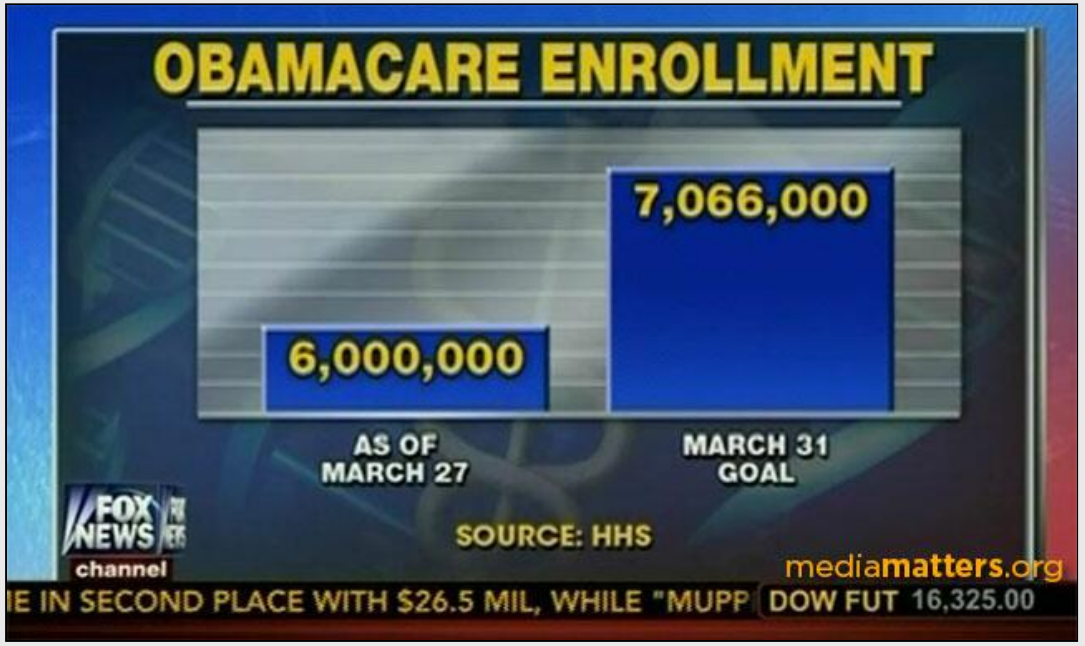


When is it okay to to truncate an axis?

- When small movements matter
- When zero values are impossible
- When it's normal / a convention

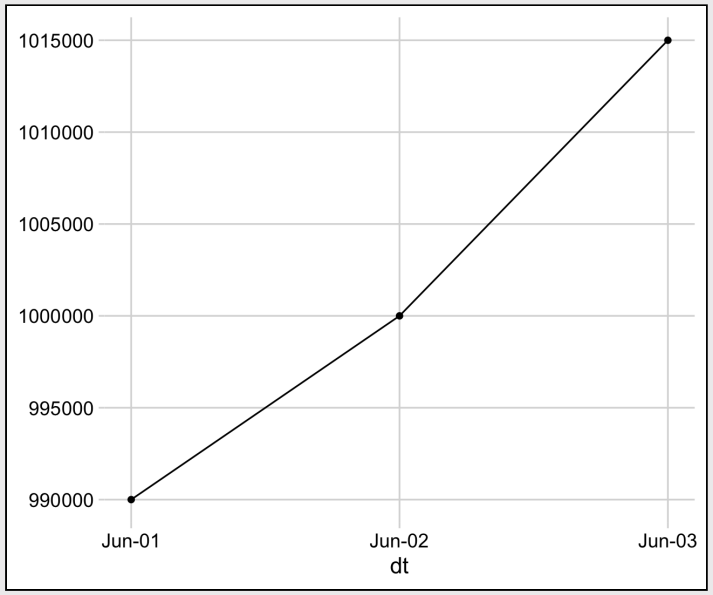
When is it okay to to truncate an axis?

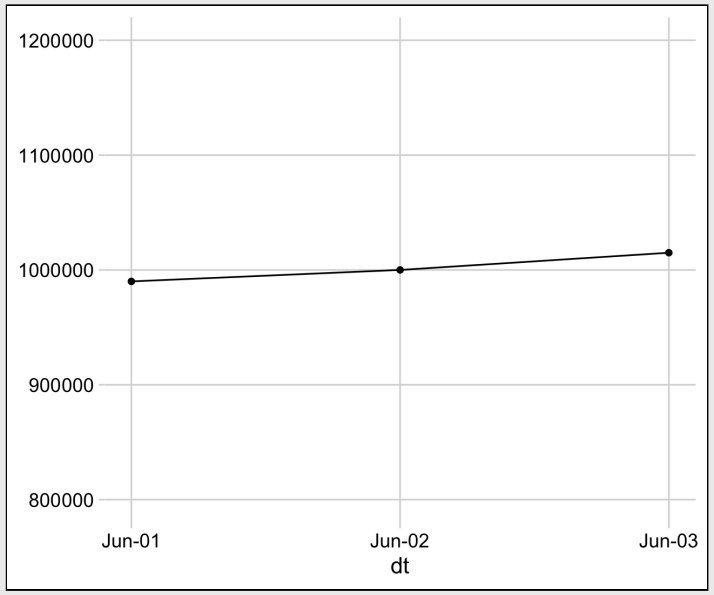
- When small movements matter
- When zero values are impossible
- When it's normal / a convention
- Never on a bar chart

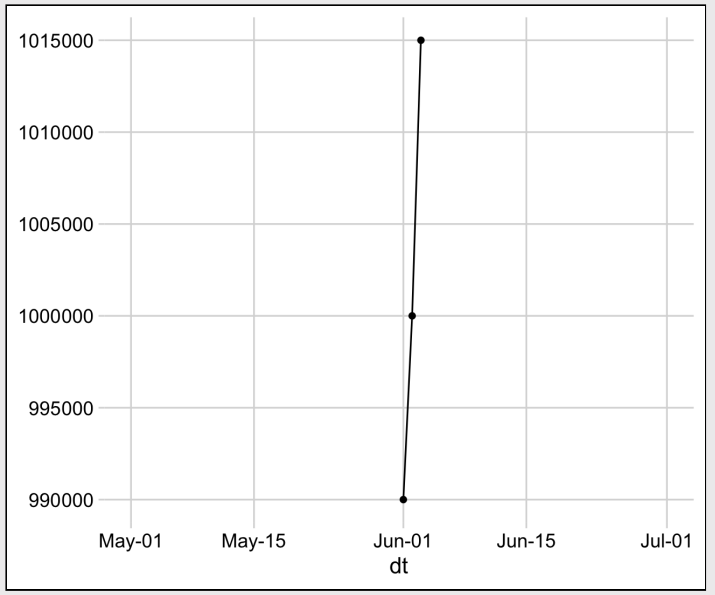


You are most sensitive to changes in angle close to 45 degrees

You are most sensitive to changes in angle close to 45 degrees

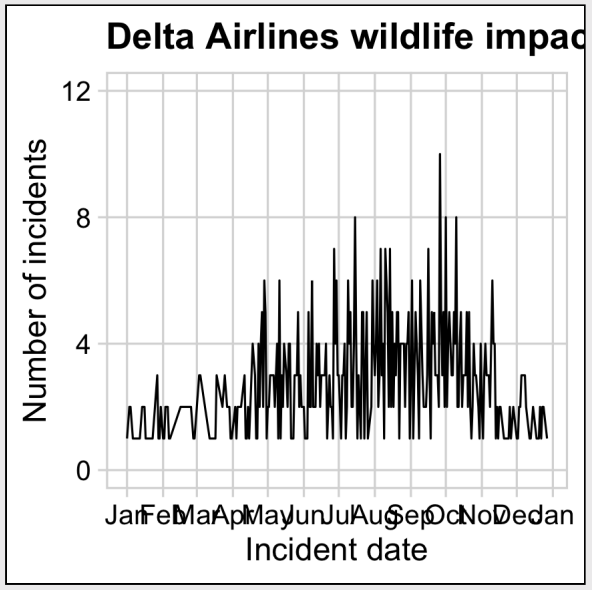




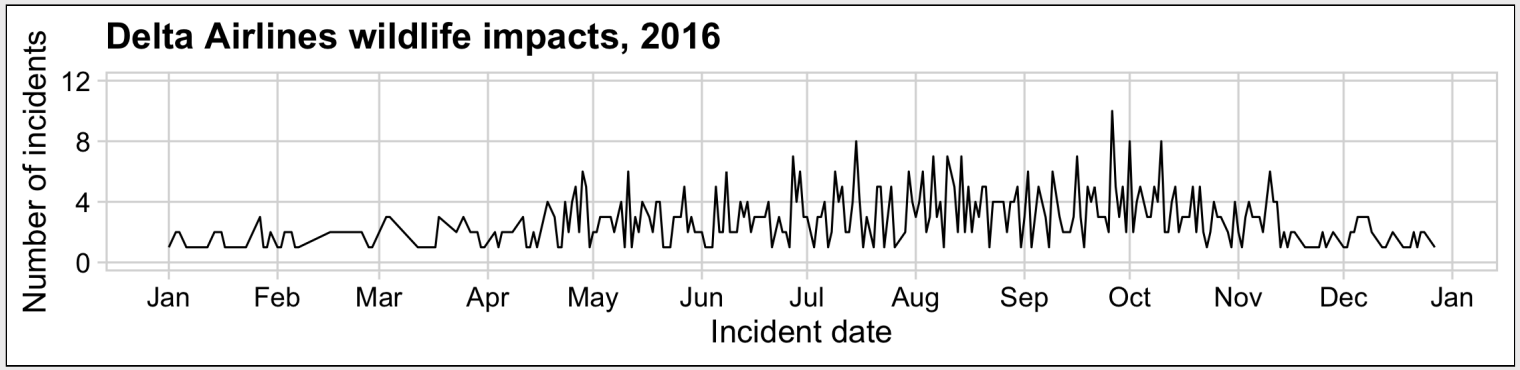


You are most sensitive to changes in angle close to 45 degrees

Bad



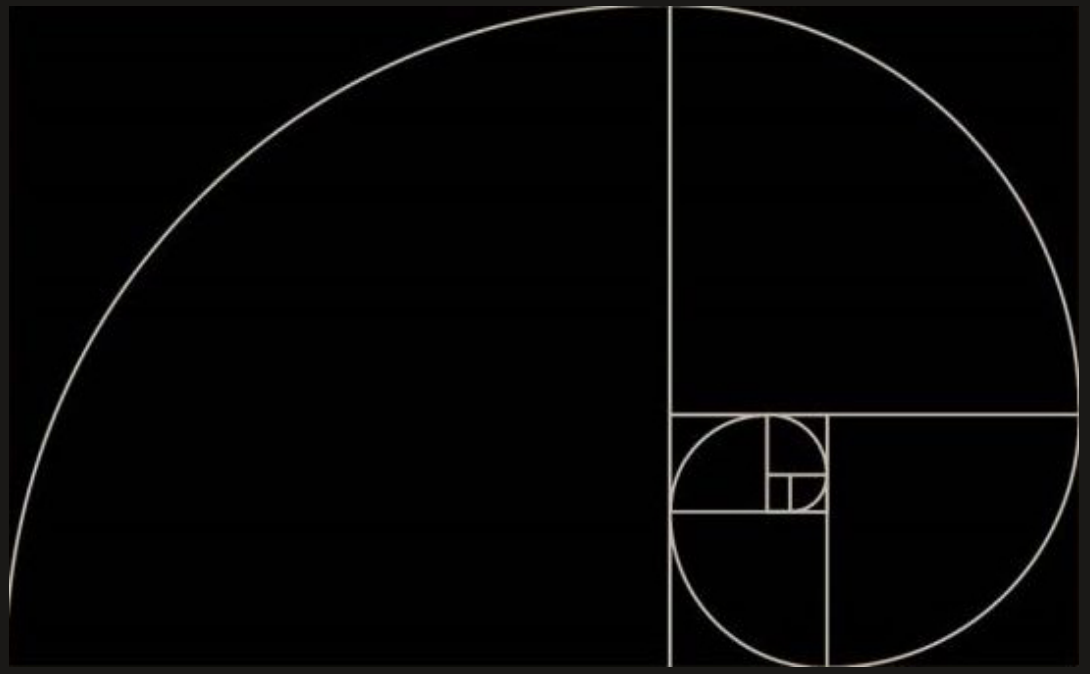
Better



Set image dimensions in R chunk header:

```
```{r, fig.width = 5, fig.height = 3.1}
```
```

Consider setting dimensions to "Golden Ratio" (1 : 1.618)



Approx. to golden ratio:

width	height
5	3.1
6	3.7
7	4.3

Also check out Donald Duck in Mathemagic Land

Adjust axes with `scale_` functions

Continous variables

```
scale_x_continuous()
scale_y_continuous()
```

Discrete variables

```
scale_x_discrete()
scale_y_discrete()
```

Others

```
scale_x_log10()
scale_y_log10()
scale_x_date()
```

Common arguments for continuous variables

```
scale_y_continuous(
  # Set the lower & upper boundaries
  limits = c(lower, upper),

  # Explicitly set the break points
  breaks = c(break1, break2, etc.)

  # Adjust the axis so bars start at 0
  expand = expand_scale(mult = c(0, 0.05))
)
```

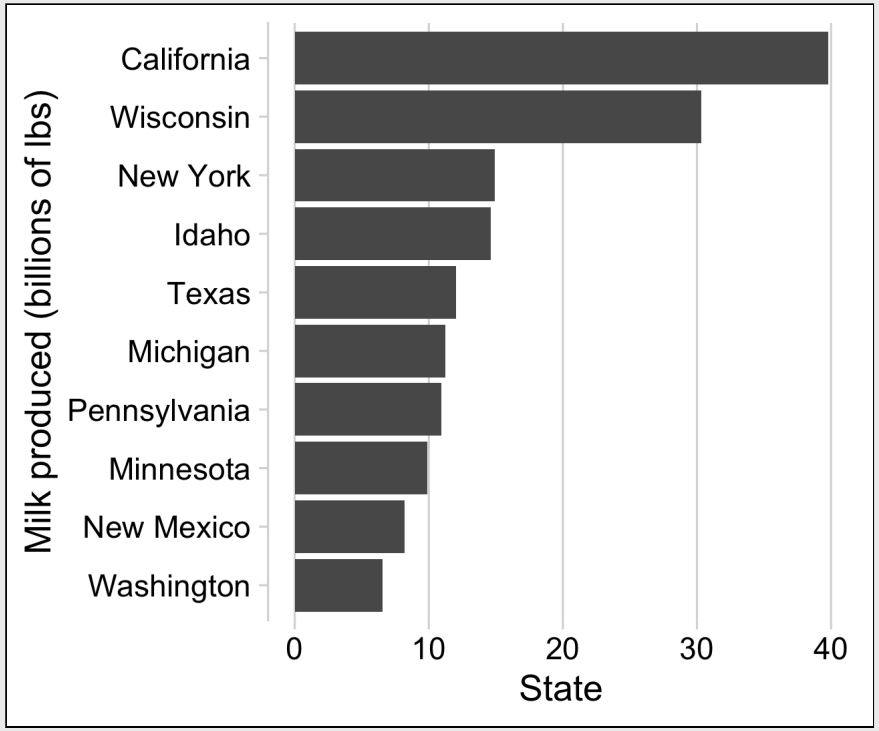
Adjusting **continuous** scales

Summarise the data

```
milk_top10states <- milk_production %>%
    filter(year == 2017) %>%
    arrange(desc(milk_produced)) %>%
    slice(1:10) %>%
    mutate(
        milk_produced = milk_produced / 10^9,
        state = fct_reorder(state, milk_produced))
```

Make the plot

```
ggplot(milk_top10states) +
  geom_col(aes(x = state, y = milk_produced)) +
  coord_flip() +
  theme_minimal_vgrid(font_size = 18) +
  labs(x = 'Milk produced (billions of lbs)',
       y = 'State')
```



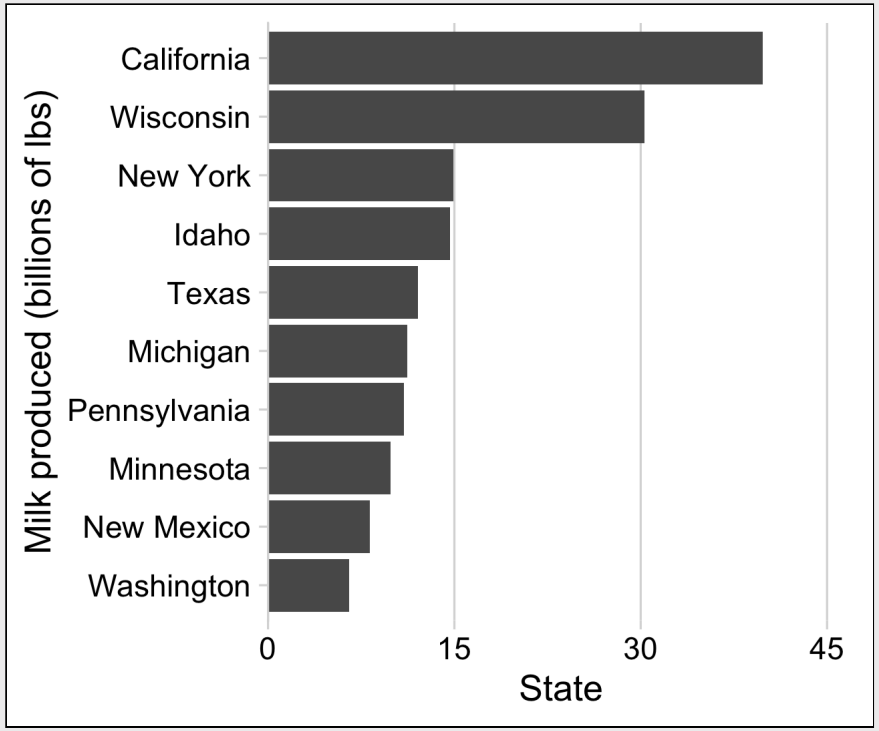
Adjusting **continuous** scales

Summarise the data

```
milk_top10states <- milk_production %>%
    filter(year == 2017) %>%
    arrange(desc(milk_produced)) %>%
    slice(1:10) %>%
    mutate(
        milk_produced = milk_produced / 10^9,
        state = fct_reorder(state, milk_produced))
```

Make the plot

```
ggplot(milk_top10states) +
  geom_col(aes(x = state, y = milk_produced)) +
  coord_flip() +
  scale_y_continuous(
    breaks = c(0, 15, 30, 45),
    limits = c(0 , 45),
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid(font_size = 18) +
  labs(x = 'Milk produced (billions of lbs)',
       y = 'State')
```



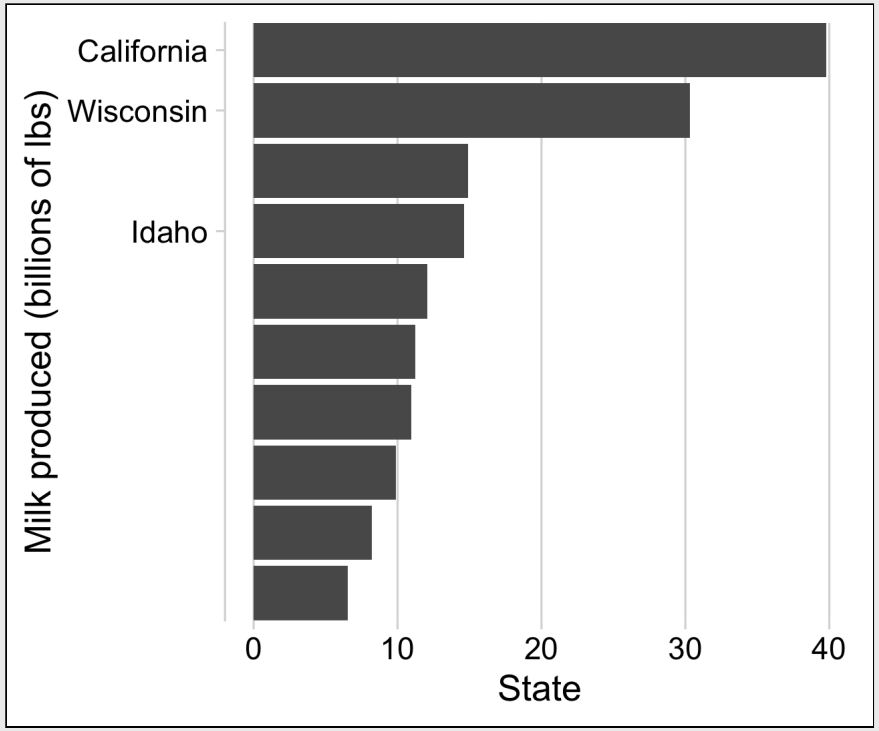
Adjusting **discrete** scales

Summarise the data

```
milk_top10states <- milk_production %>%
    filter(year == 2017) %>%
    arrange(desc(milk_produced)) %>%
    slice(1:10) %>%
    mutate(
        milk_produced = milk_produced / 10^9,
        state = fct_reorder(state, milk_produced))
```

Make the plot

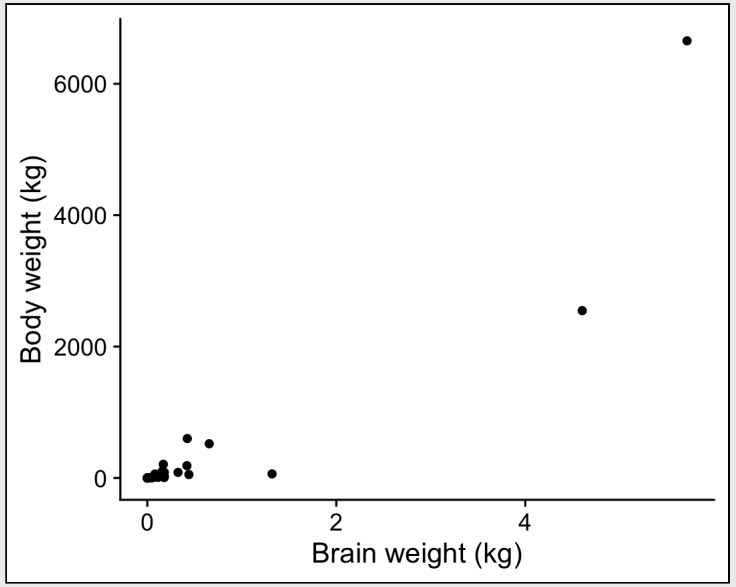
```
ggplot(milk_top10states) +
  geom_col(aes(x = state, y = milk_produced)) +
  coord_flip() +
  scale_x_discrete(
    breaks = c('California', 'Wisconsin', 'Idaho'),
    expand = expand_scale(mult = c(0, 0.05))) +
  theme_minimal_vgrid(font_size = 18) +
  labs(x = 'Milk produced (billions of lbs)',
       y = 'State')
```



Adjusting **log** scales

Regular scaling

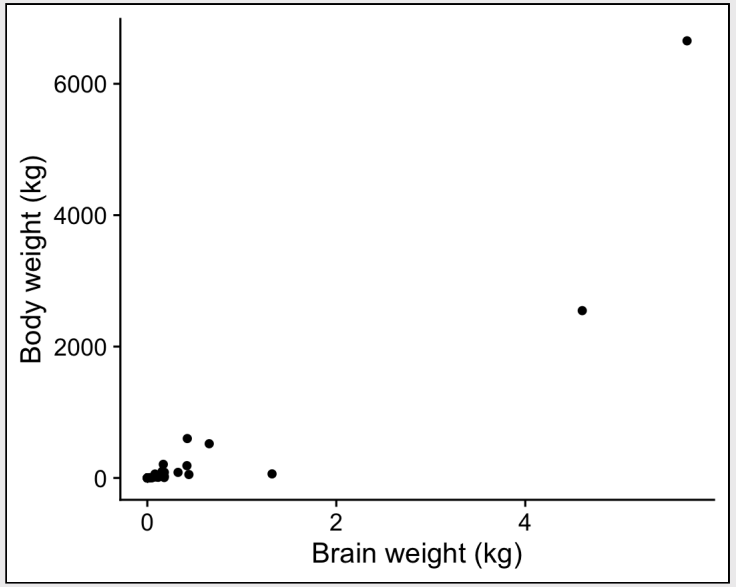
```
ggplot(msleep) +
    geom_point(aes(x = brainwt, y = bodywt)) +
    theme_half_open() +
    labs(x = 'Brain weight (kg)',
         y = 'Body weight (kg)')
```



Adjusting **log** scales

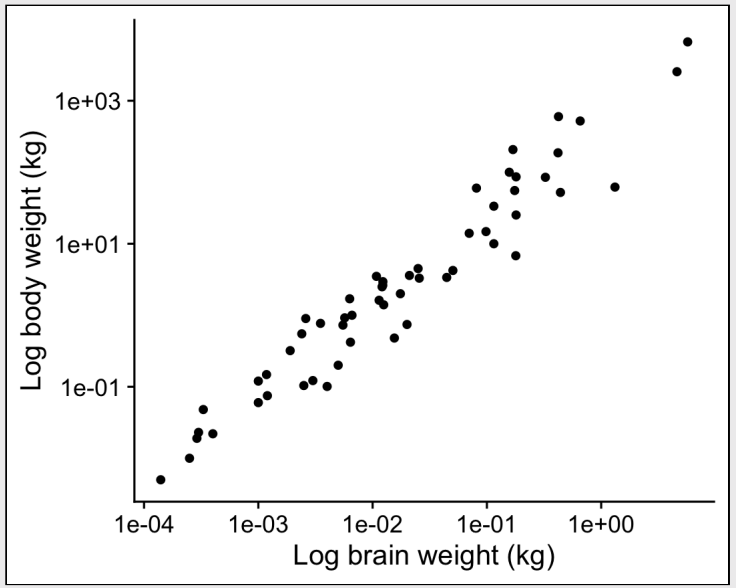
Regular scaling

```
ggplot(msleep) +
    geom_point(aes(x = brainwt, y = bodywt)) +
    theme_half_open() +
    labs(x = 'Brain weight (kg)',
         y = 'Body weight (kg)')
```



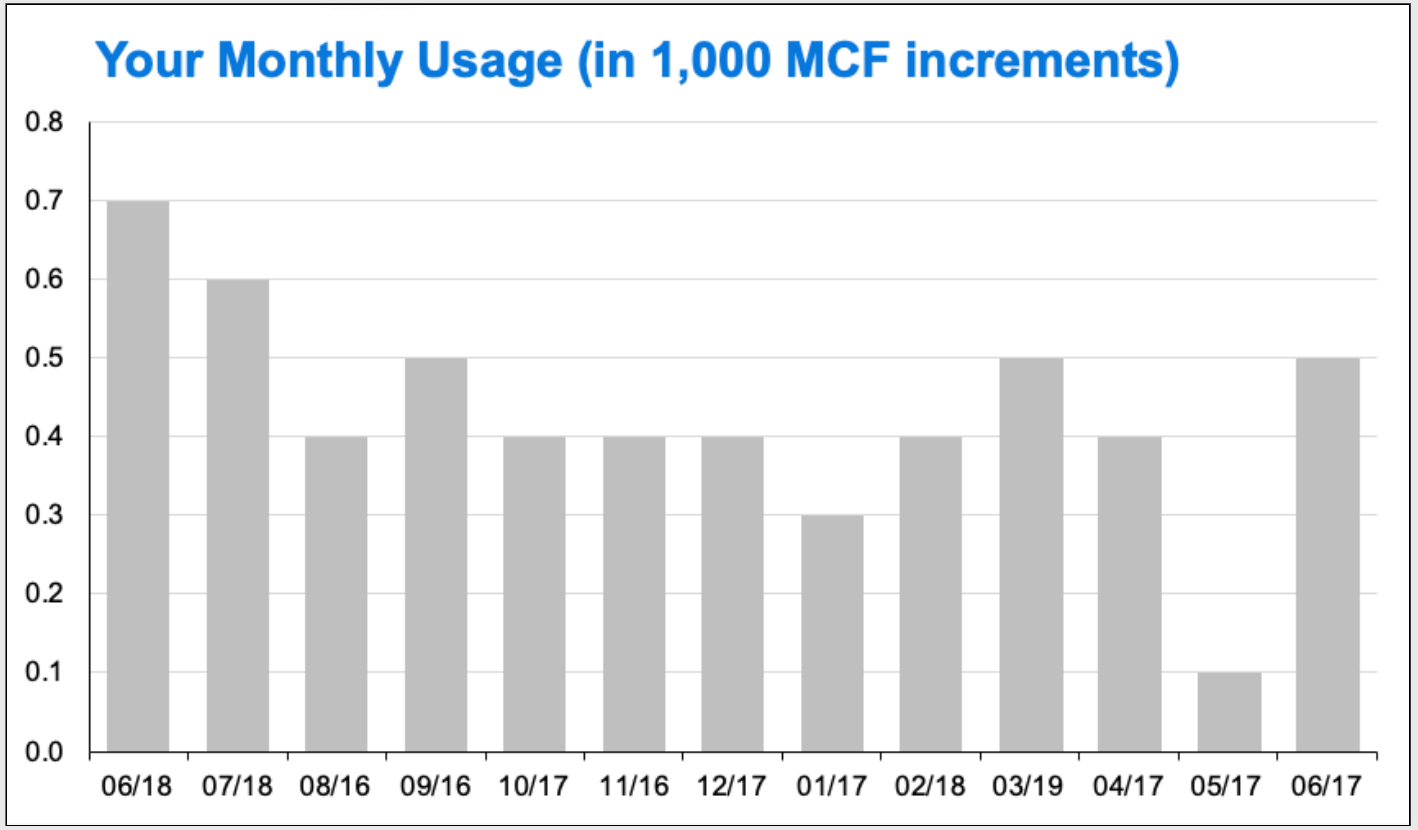
Log scaling

```
ggplot(msleep) +
    geom_point(aes(x = brainwt, y = bodywt)) +
    scale_x_log10() +
    scale_y_log10() +
    theme_half_open() +
    labs(x = 'Log brain weight (kg)',
         y = 'Log body weight (kg)')
```

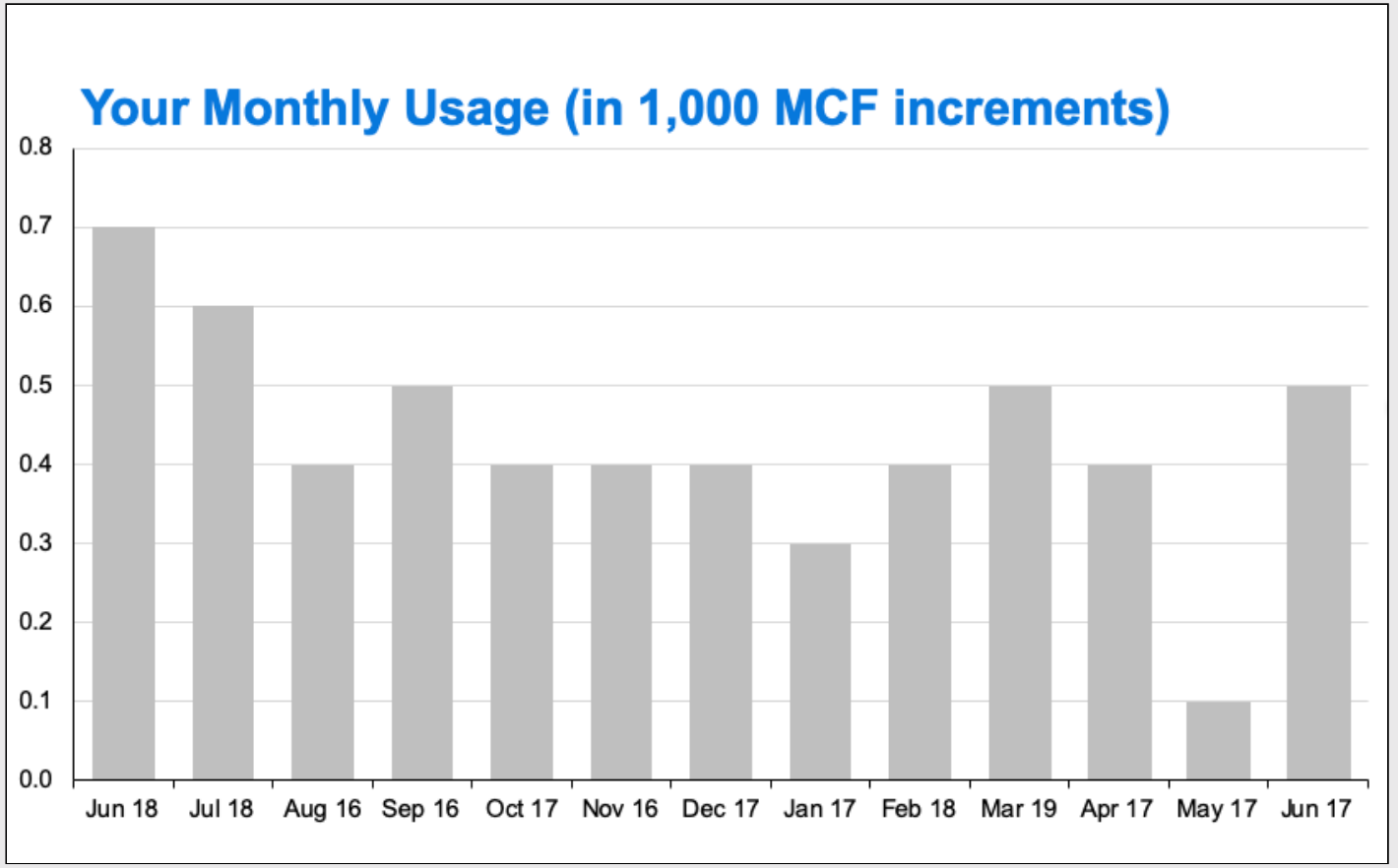


Date scales can be confusing

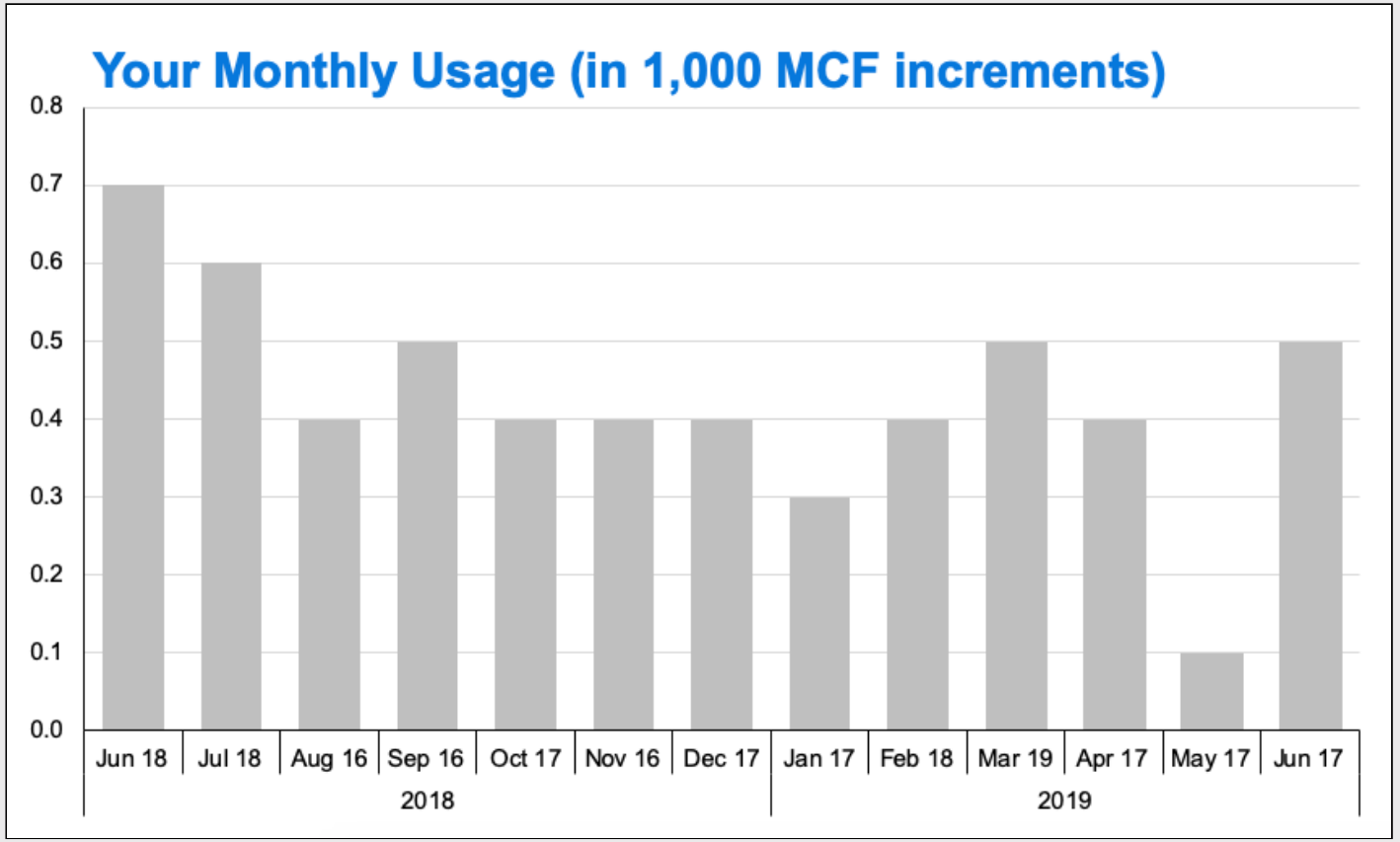
What's wrong with this chart?



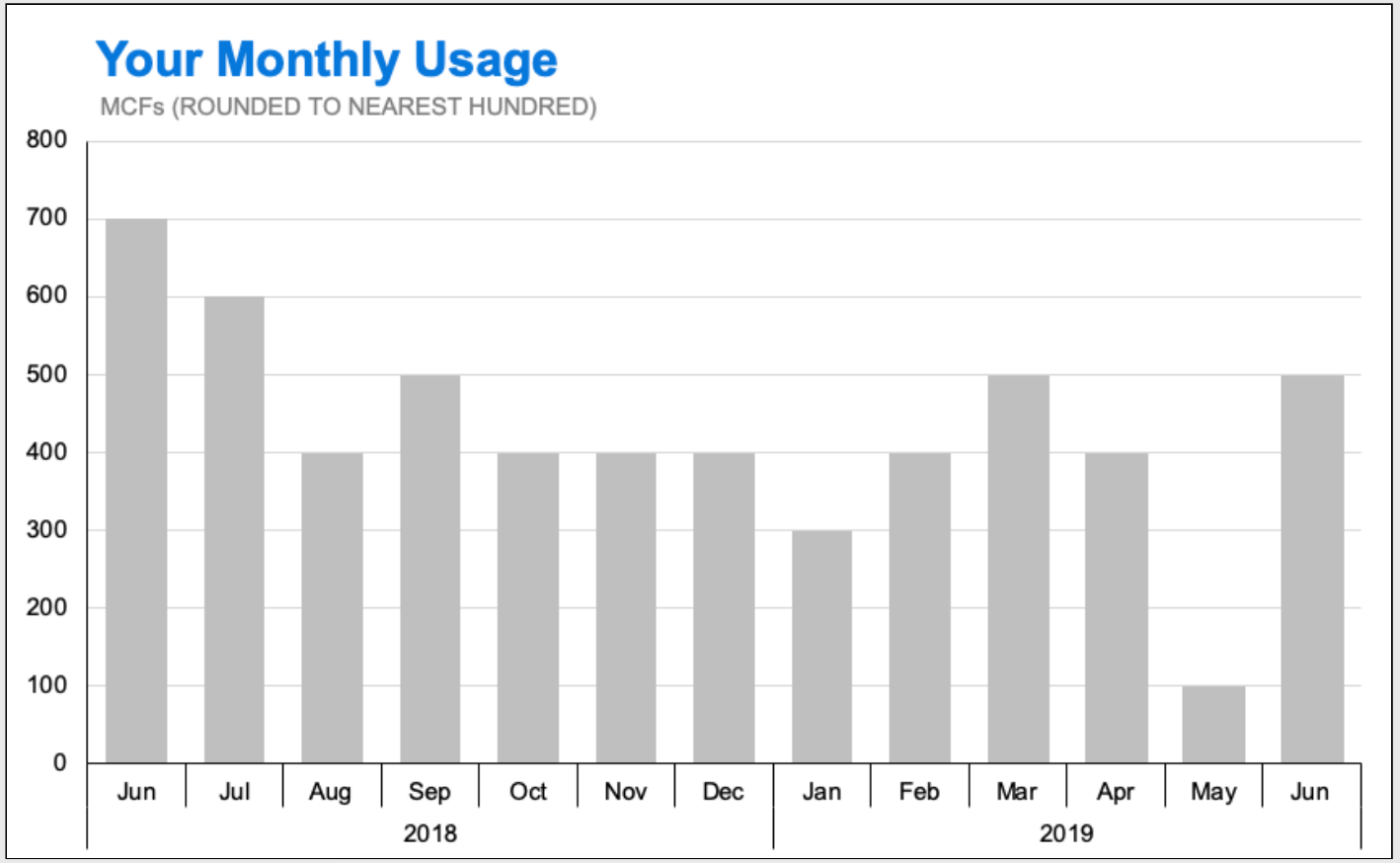
What's wrong with this chart?



What's wrong with this chart?



What's wrong with this chart?



Adjusting **date** scales

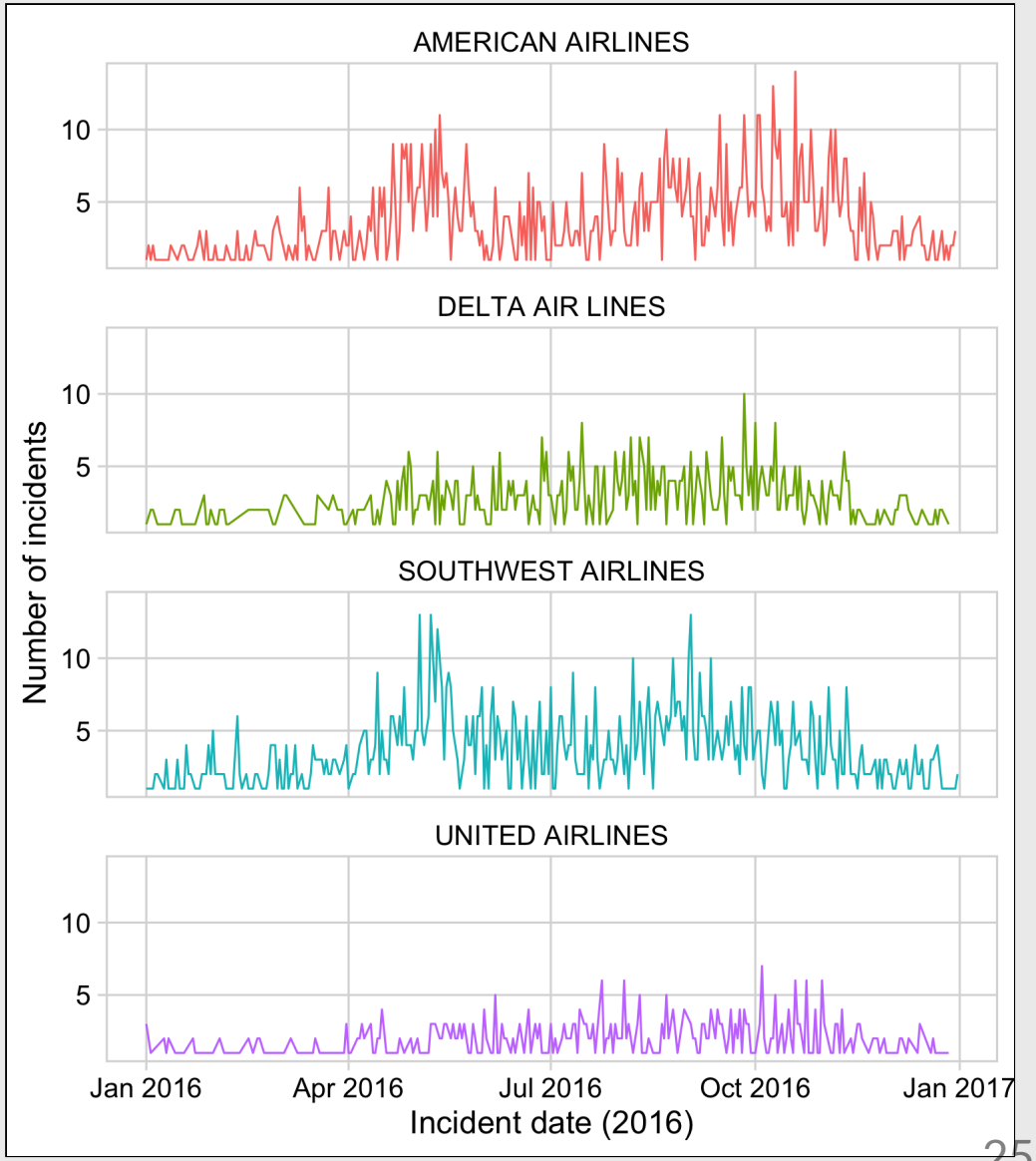
Summarise the data

```
library(lubridate)

wildlife_summary <- wildlife_impacts %>%
    filter(incident_year == 2016) %>%
    count(operator, incident_date) %>%
    mutate(incident_date = ymd(incident_date))
```

Make the plot

```
ggplot(wildlife_summary) +
    geom_line(aes(x = incident_date, y = n,
                  color = operator)) +
    facet_wrap(~operator, ncol = 1) +
    theme_minimal_grid(font_size = 16) +
    panel_border() +
    theme(legend.position = 'none') +
    labs(x = 'Incident date (2016)',
         y = 'Number of incidents')
```



Adjusting **date** scales

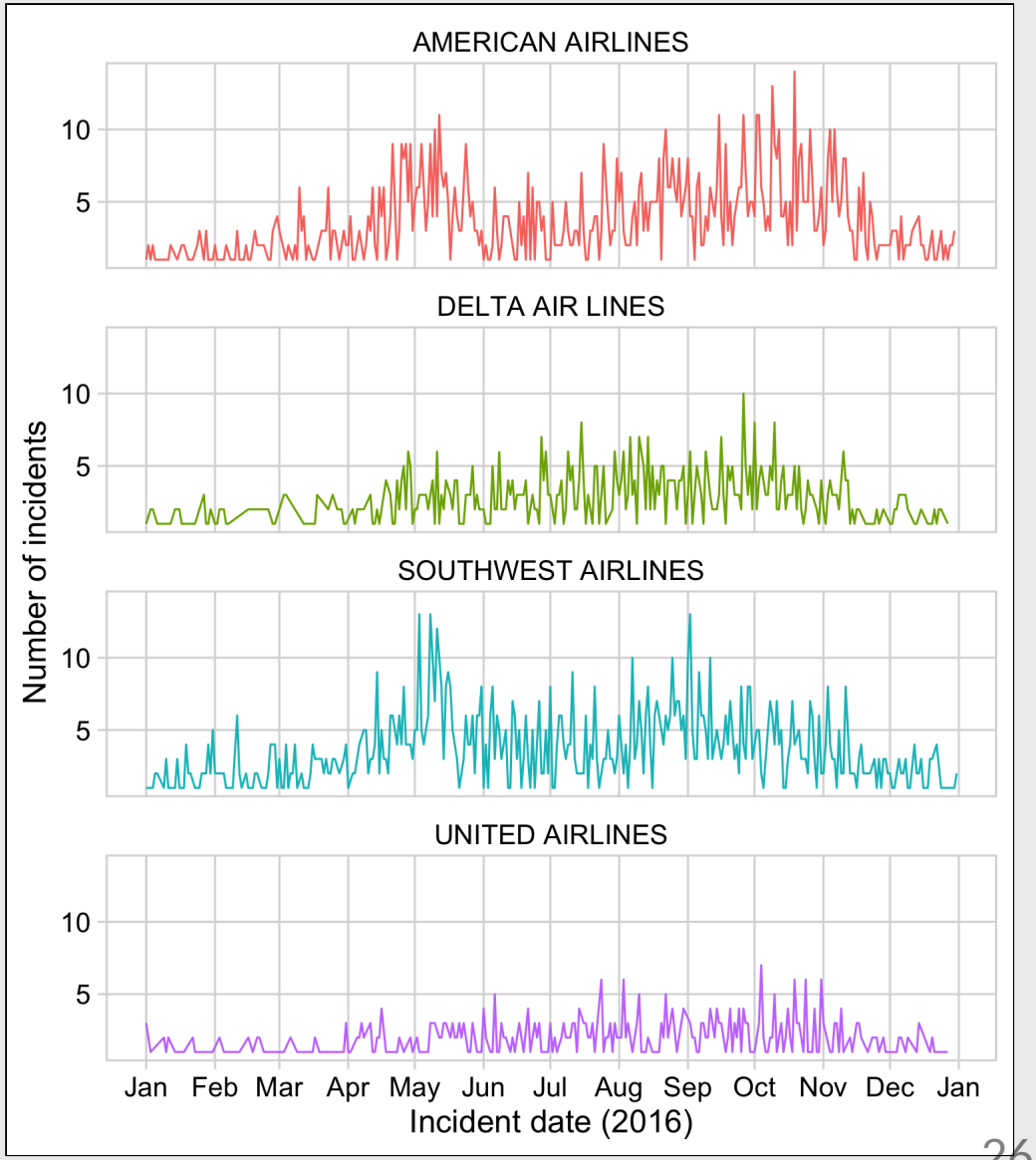
Summarise the data

```
library(lubridate)

wildlife_summary <- wildlife_impacts %>%
    filter(incident_year == 2016) %>%
    count(operator, incident_date) %>%
    mutate(incident_date = ymd(incident_date))
```

Make the plot

```
ggplot(wildlife_summary) +
  geom_line(aes(x = incident_date, y = n,
                color = operator, group = operator)) +
  facet_wrap(~operator, ncol = 1) +
  scale_x_date(
      date_breaks = '1 month',
      date_labels = '%b') +
  theme_minimal_grid(font_size = 16) +
  panel_border() +
  theme(legend.position = 'none') +
  labs(x = 'Incident date (2016)',
       y = 'Number of incidents')
```



Adjusting **date** scales

```
scale_x_date(
  date_breaks = '1 month',
  date_labels = '%b')
```

```
date_breaks = '1 month'
```

- '1 day'
- '10 days'
- '1 month'
- '3 months'
- '1 year'
- '3 years'

```
date_labels = '%b'
```

Example date: March 04, 2020

- `%Y` = 2020
- `%y` = 20
- `%B` = March
- `%b` = Mar
- `%D` = 03/04/2020
- `%d` = 03

Use **scales** library to modify scale **text**

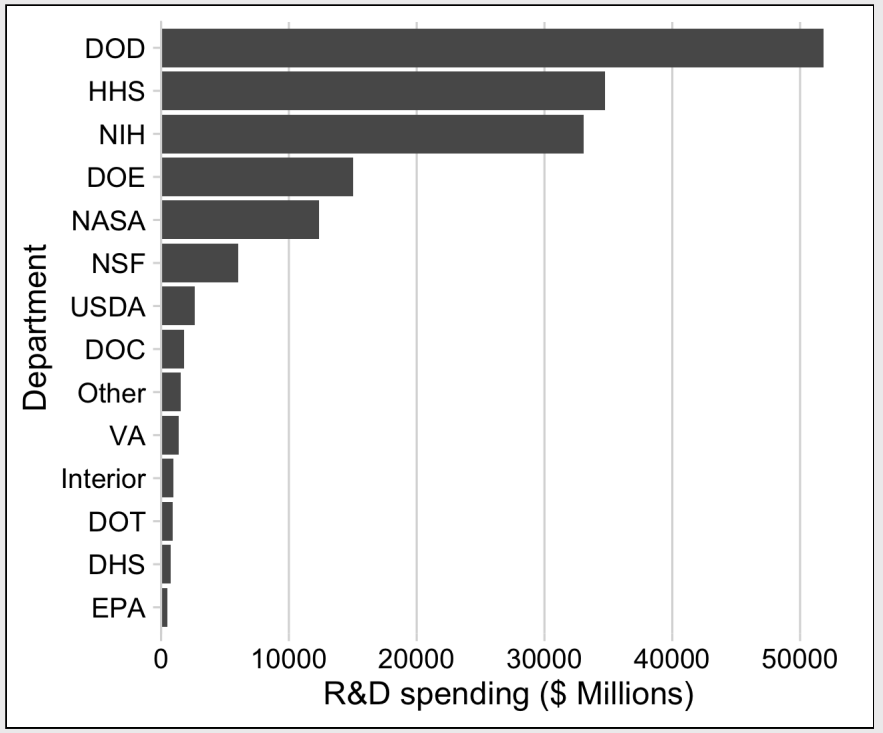
Use **scales** library to modify scale text

Summarise the data

```
federal_spending_2017 <- federal_spending %>%
    filter(year == 2017) %>%
    mutate(department = fct_reorder(
        department, rd_budget))
```

Make the plot

```
ggplot(federal_spending_2017) +
    geom_col(aes(x = department, y = rd_budget)) +
    scale_y_continuous(
      expand = expand_scale(mult = c(0, 0.05))) +
    coord_flip() +
    theme_minimal_vgrid(font_size = 16) +
    labs(x = 'Department',
         y = 'R&D spending ($ Millions)')
```



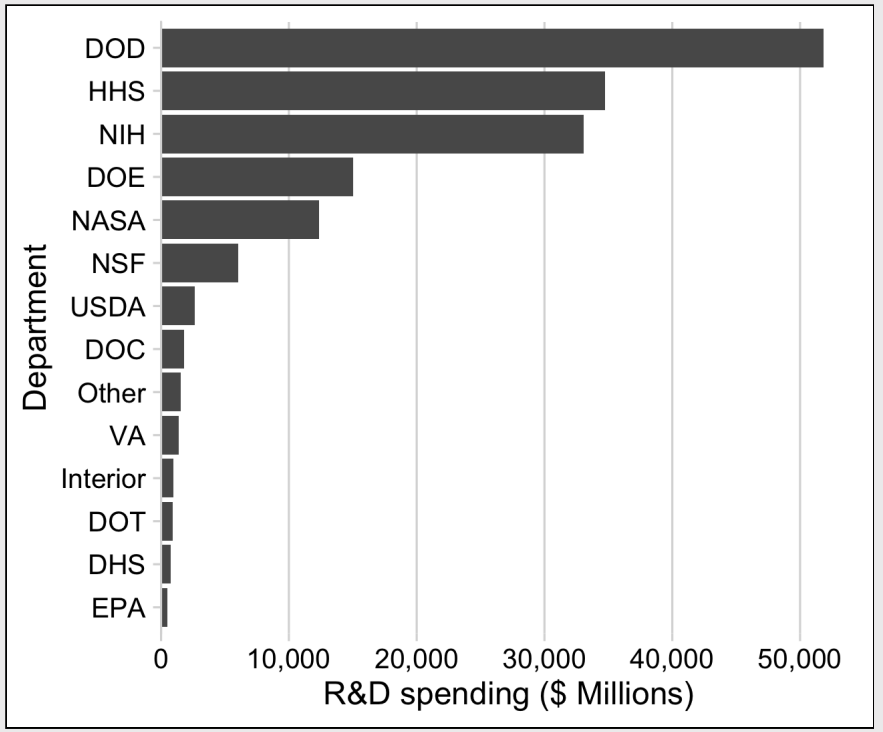
Use **scales** library to modify scale text

Summarise the data

```
federal_spending_2017 <- federal_spending %>%
    filter(year == 2017) %>%
    mutate(department = fct_reorder(
        department, rd_budget))
```

Add comma separation in x-axis labels

```
ggplot(federal_spending_2017) +
    geom_col(aes(x = department, y = rd_budget)) +
    scale_y_continuous(
      labels = scales::comma,
      expand = expand_scale(mult = c(0, 0.05))) +
    coord_flip() +
    theme_minimal_vgrid(font_size = 16) +
    labs(x = 'Department',
         y = 'R&D spending ($ Millions)')
```



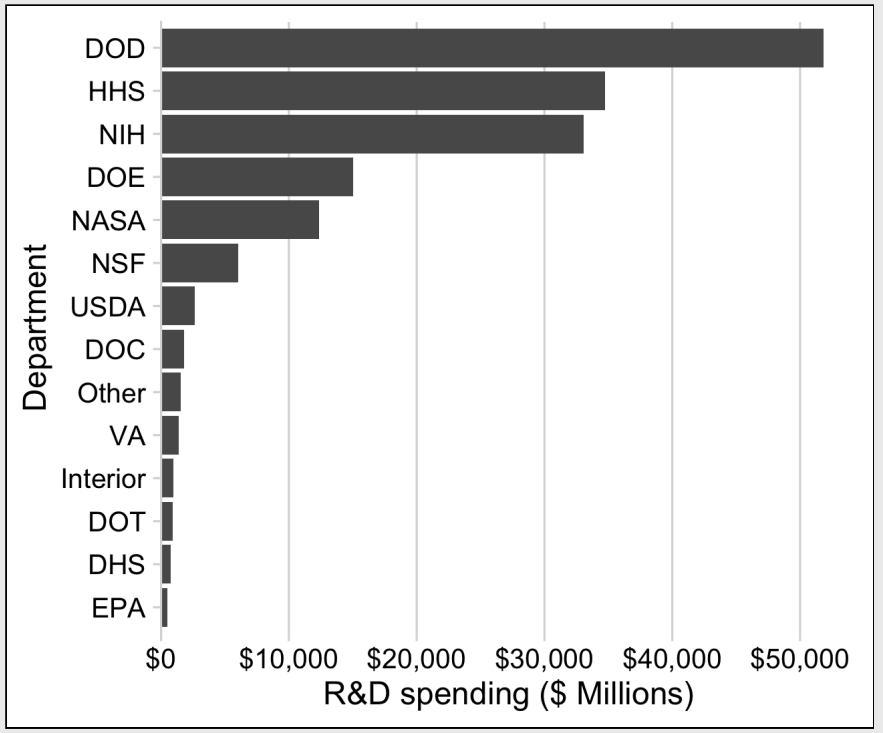
Use **scales** library to modify scale text

Summarise the data

```
federal_spending_2017 <- federal_spending %>%
    filter(year == 2017) %>%
    mutate(department = fct_reorder(
        department, rd_budget))
```

Add $ sign in x-axis labels

```
ggplot(federal_spending_2017) +
    geom_col(aes(x = department, y = rd_budget)) +
    scale_y_continuous(
      labels = scales::dollar,
      expand = expand_scale(mult = c(0, 0.05))) +
    coord_flip() +
    theme_minimal_vgrid(font_size = 16) +
    labs(x = 'Department',
         y = 'R&D spending ($ Millions)')
```



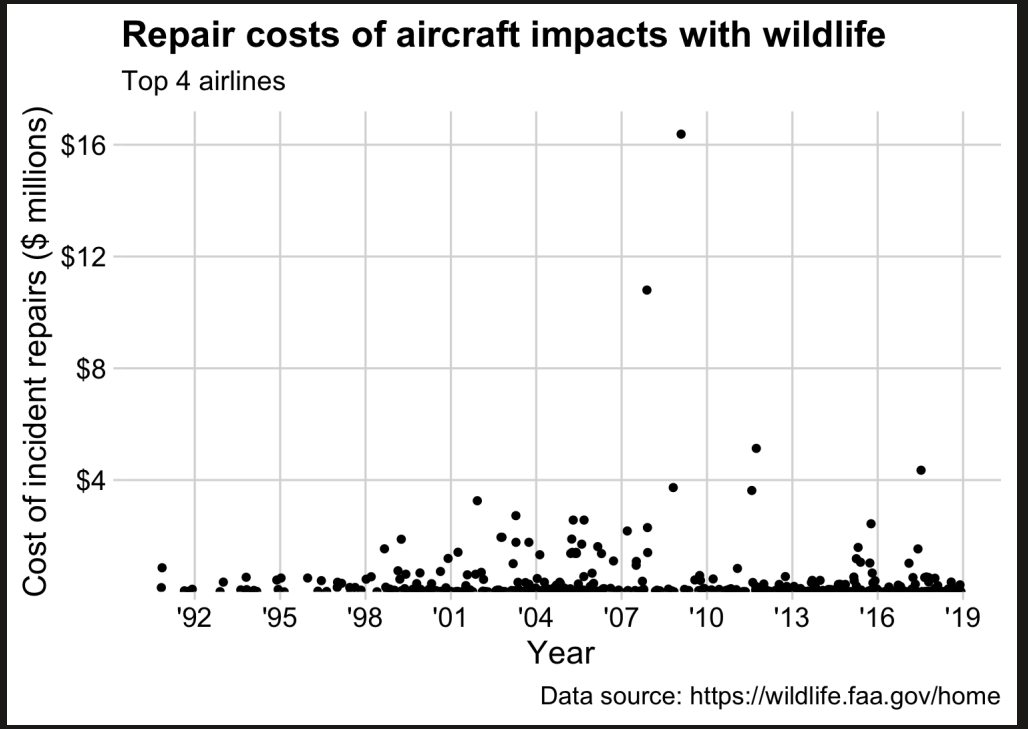
Your turn

Use the `wildlife_costs` data frame to create the following chart.

Hints:

- Note the x and y axis labels!

15:00

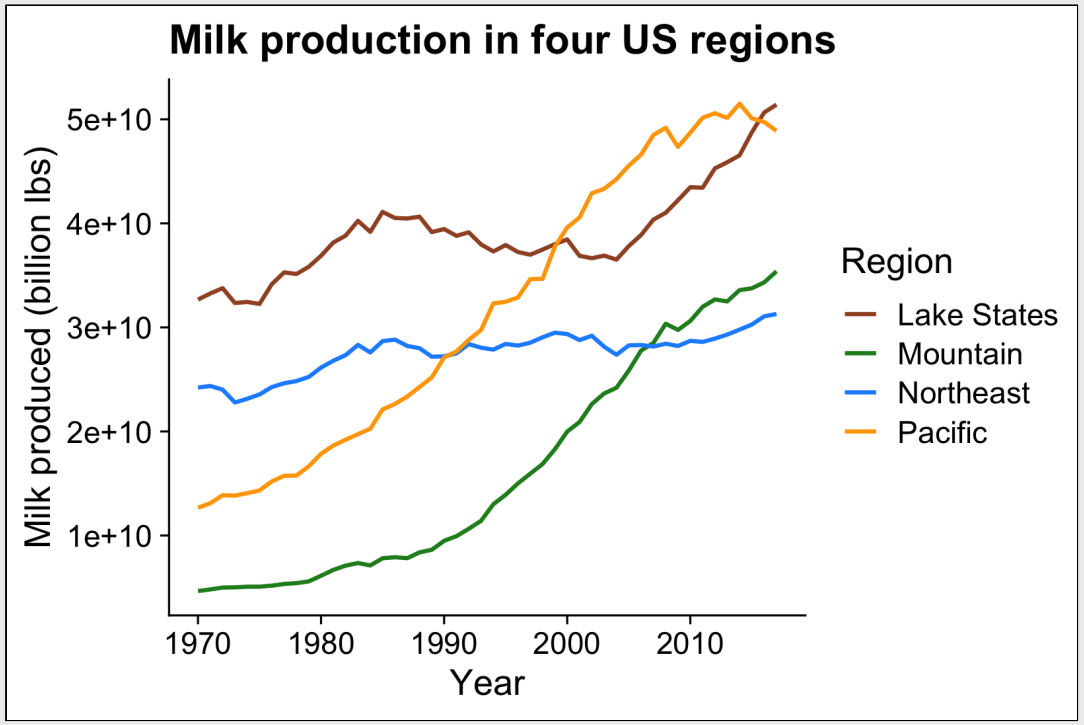


Polishing your charts

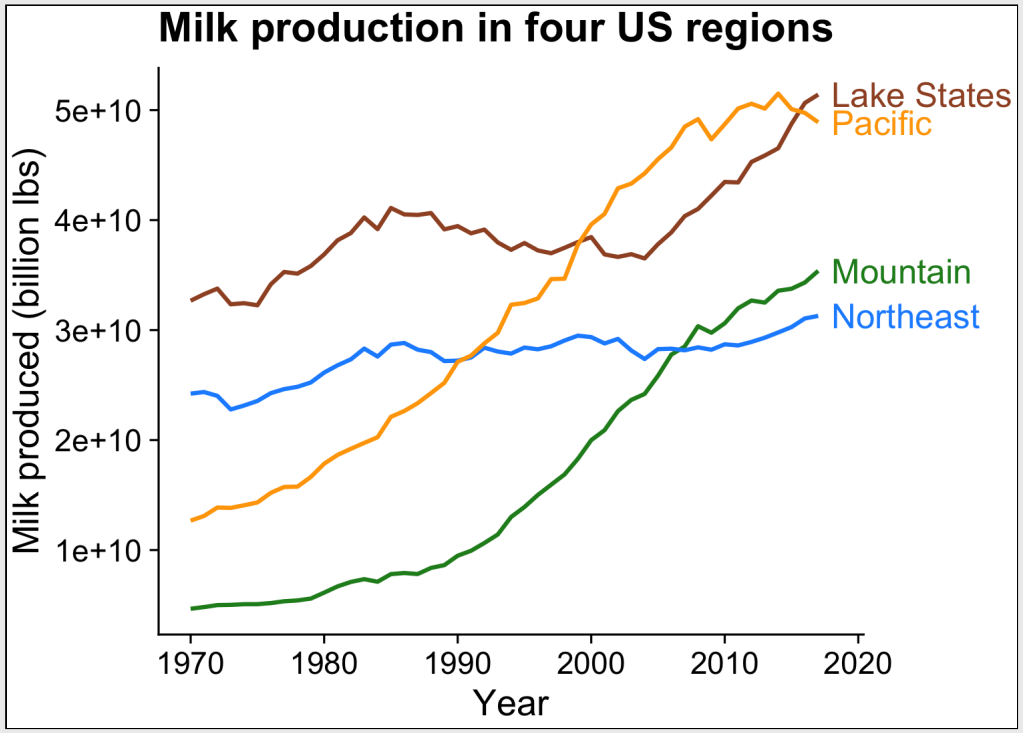
1. Scales
2. Annotations
3. Colors
4. Fonts
5. Rmd tricks

Legends suck

Legends require look-up task

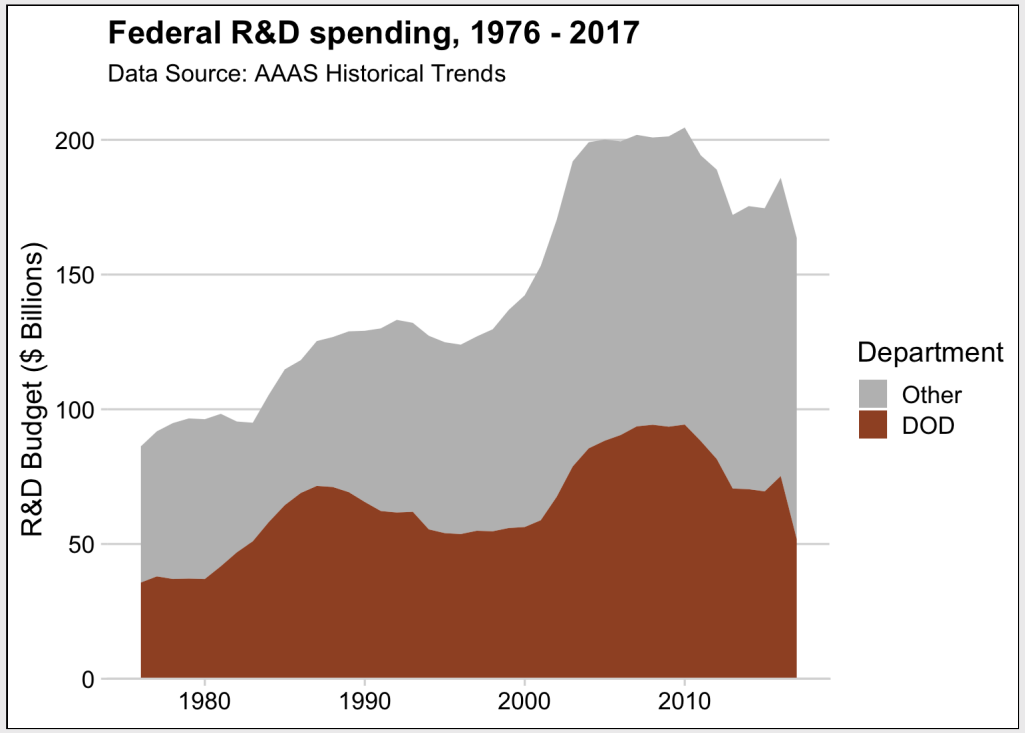


Direct labeling is much better

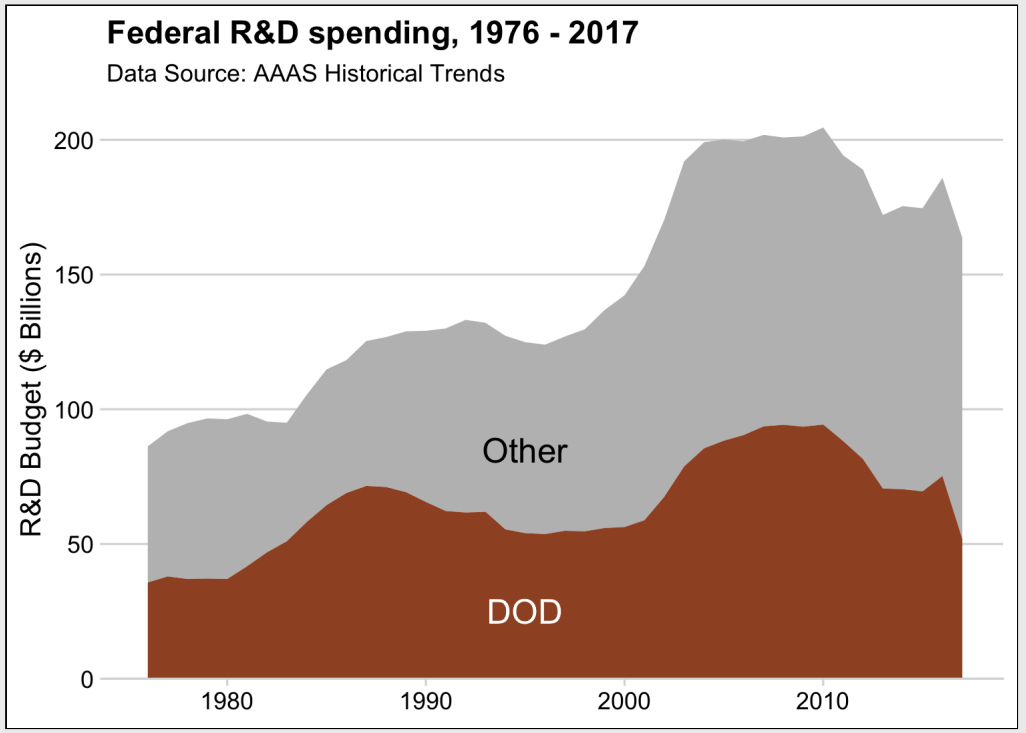


Legends suck

Legends require look-up task

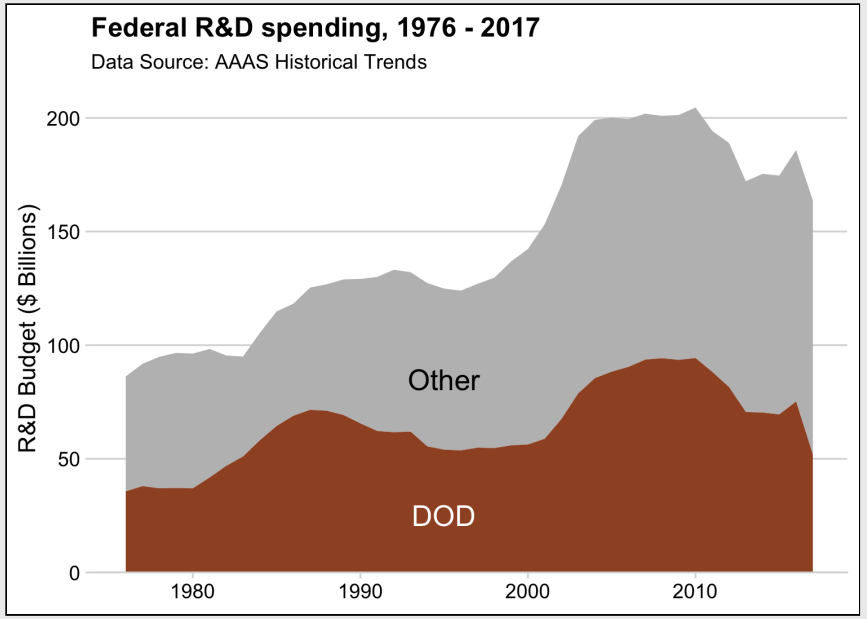


Direct labeling is much better



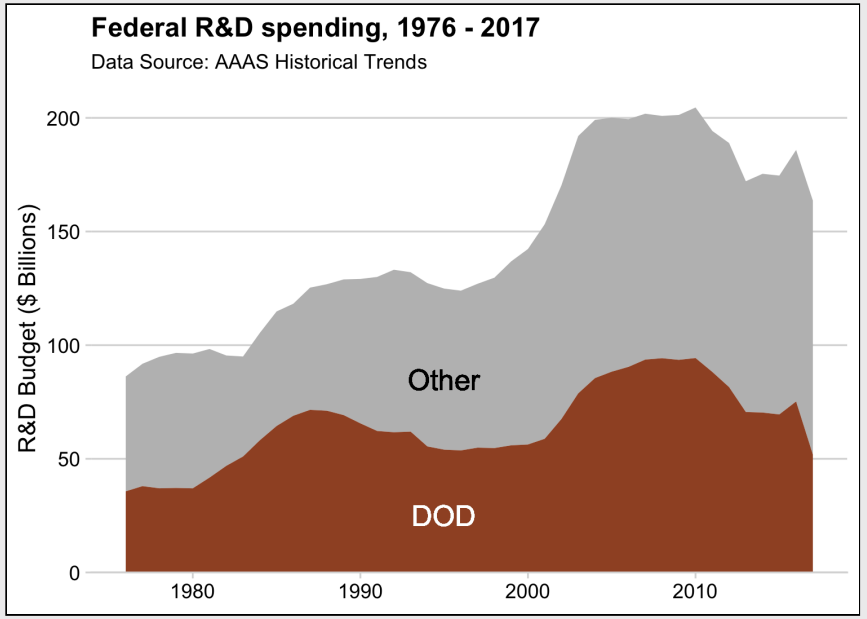
Use `annotate()` to add text to chart

```
dod_spending <- ggplot(federal_spending_summary) +
    geom_area(aes(x = year, y = rd_budget,
                  fill = department)) +
    annotate(geom = 'text', x = 1995, y = 85,
             label = 'Other', size = 6, color = 'black') +
    annotate(geom = 'text', x = 1995, y = 25,
             label = 'DOD', size = 6, color = 'white') +
    scale_y_continuous(
        expand = expand_scale(mult = c(0, 0.05))) +
    scale_fill_manual(values = c('grey', 'sienna')) +
    theme_minimal_hgrid() +
    theme(legend.position = 'none') +
    labs(x = NULL,
         y = 'R&D Budget ($ Billions)',
         title = 'Federal R&D spending, 1976 - 2017',
         subtitle = 'Data Source: AAAS Historical Trends')
```



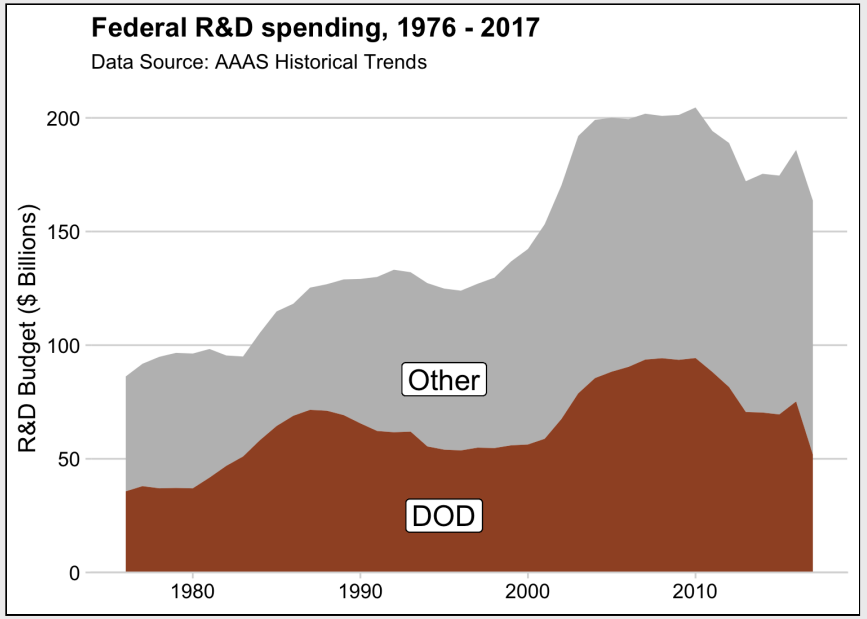
Use `geom_text()` to add text to chart

```
dod_spending <- ggplot(federal_spending_summary) +
    geom_area(aes(x = year, y = rd_budget,
                  fill = department)) +
    geom_text(aes(x = 1995, y = 85, label = 'Other'),
              size = 6, color = 'black') +
    geom_text(aes(x = 1995, y = 25, label = 'DOD'),
              size = 6, color = 'white') +
    scale_y_continuous(
        expand = expand_scale(mult = c(0, 0.05))) +
    scale_fill_manual(values = c('grey', 'sienna')) +
    theme_minimal_hgrid() +
    theme(legend.position = 'none') +
    labs(x = NULL,
         y = 'R&D Budget ($ Billions)',
         title = 'Federal R&D spending, 1976 - 2017',
         subtitle = 'Data Source: AAAS Historical Trends')
```



Use `geom_label()` to add text to chart **with a background**

```
dod_spending <- ggplot(federal_spending_summary) +
    geom_area(aes(x = year, y = rd_budget,
                  fill = department)) +
    geom_label(aes(x = 1995, y = 85, label = 'Other'),
               size = 6) +
    geom_label(aes(x = 1995, y = 25, label = 'DOD'),
               size = 6) +
    scale_y_continuous(
        expand = expand_scale(mult = c(0, 0.05))) +
    scale_fill_manual(values = c('grey', 'sienna')) +
    theme_minimal_hgrid() +
    theme(legend.position = 'none') +
    labs(x = NULL,
         y = 'R&D Budget ($ Billions)',
         title = 'Federal R&D spending, 1976 - 2017',
         subtitle = 'Data Source: AAAS Historical Trends')
```



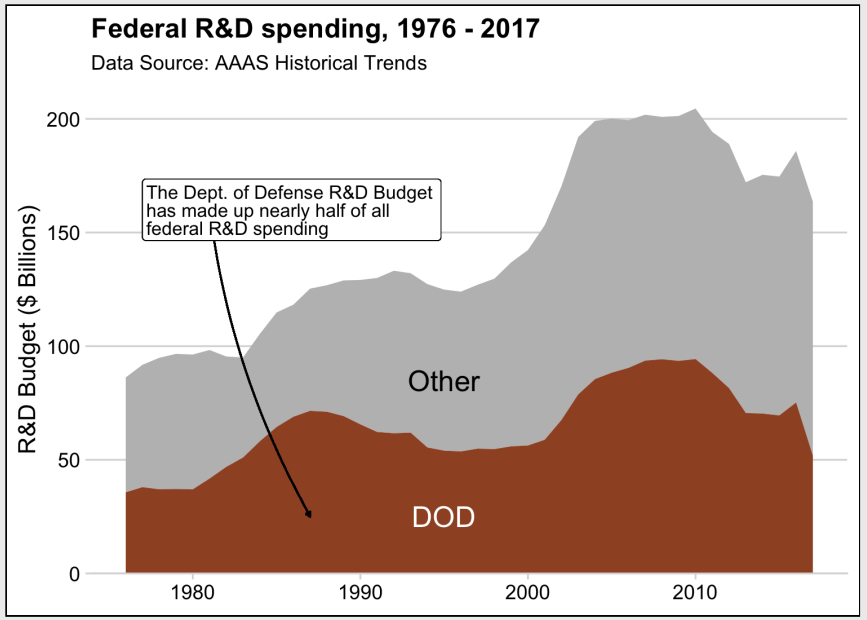
Use `geom_curve()` + `geom_label()` to direct attention

Make the label

```
label <- "The Dept. of Defense R&D Budget
has made up nearly half of all
federal R&D spending"
```

Add label with curved arrow

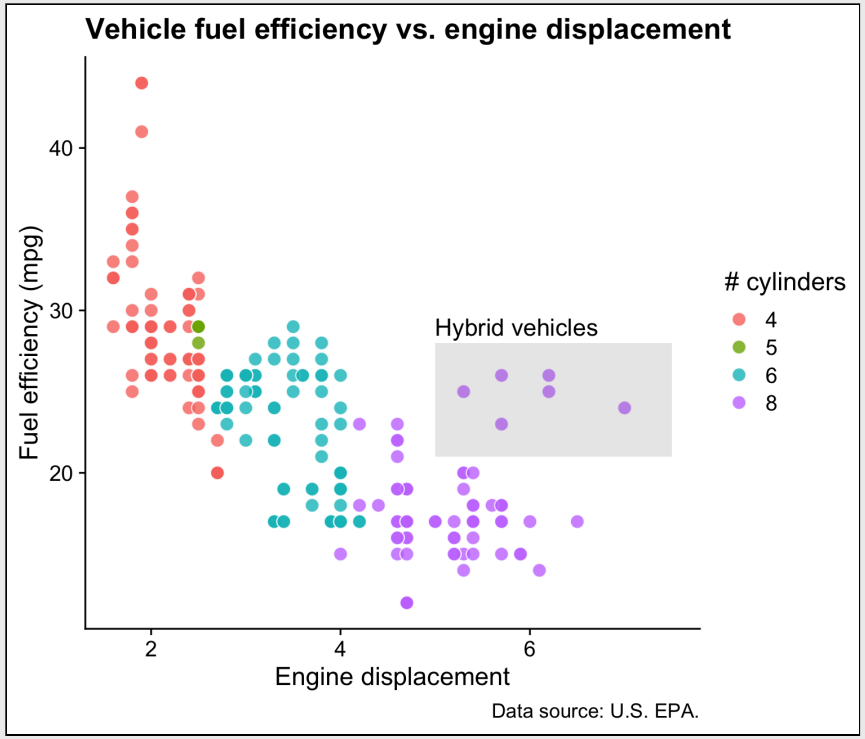
```
dod_spending +
    geom_curve(
        aes(x = 1981, xend = 1987,
            y = 160, yend = 25),
        size = 0.5,
        curvature = 0.1,
        arrow = arrow(length = unit(0.01, "npc"),
                      type = "closed")
    ) +
    geom_label(
        aes(x = 1977, y = 160, label = label),
        hjust = 0, lineheight = 0.8
    )
```



Use `annotate()` to direct attention

Use `geom = "rect"` for box, `geom = "text"` for label

```
ggplot(mpg, aes(x = displ, y = hwy)) +
    geom_point(aes(fill = as.factor(cyl)),
               color = 'white', alpha = 0.8,
               size = 3.5, shape = 21) +
    annotate(geom = "rect",
             xmin = 5, xmax = 7.5,
             ymin = 21, ymax = 28,
             fill = "grey55", alpha = 0.2) +
    annotate(geom = "text",
             x = 5, y = 29,
             label = "Hybrid vehicles",
             hjust = 0, size = 5) +
    theme_half_open(font_size = 15) +
    labs(x = "Engine displacement",
         y = "Fuel efficiency (mpg)",
         fill = '# cylinders',
         title = "Vehicle fuel efficiency vs. engine displacement",
         caption = "Data source: U.S. EPA.")
```

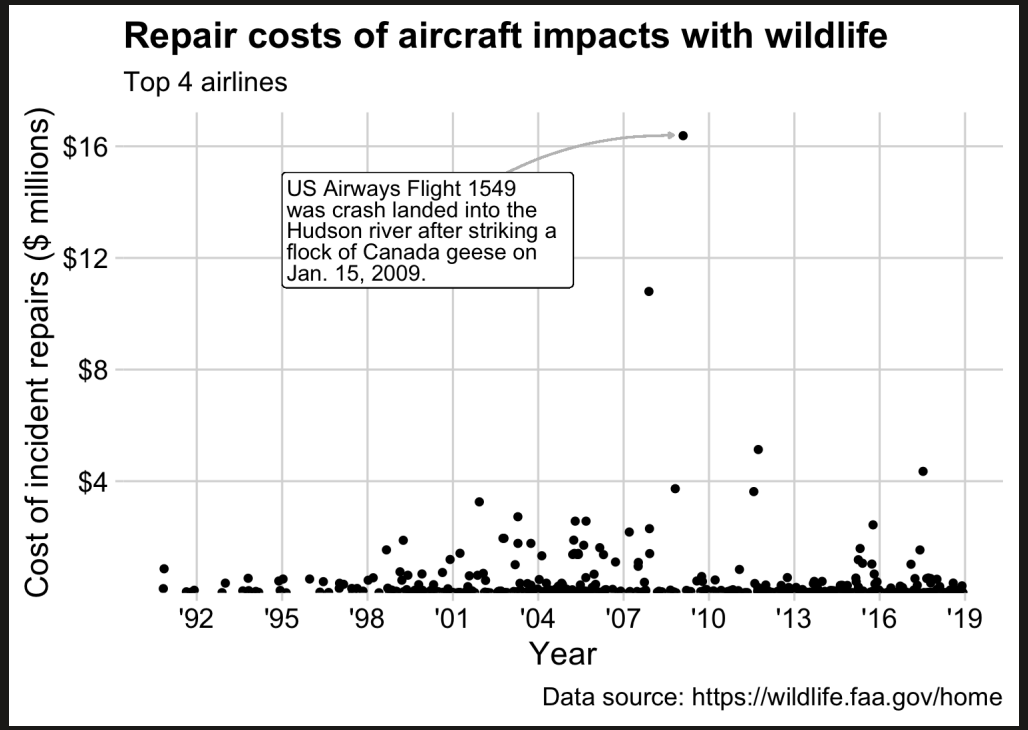


Your turn

Use the `wildlife_costs` data frame to create the following chart.

Hints:

- Use the **lubridate** package to create dates.
- For the `geom_label()`, use these points:
  - `x = ymd('1995-01-01')`
  - `y = 13`
- For the `geom_curve()`, use these points:
  - `x = ymd('2000-01-01')`
  - `xend = ymd('2008-10-01')`
  - `y = 13`
  - `yend = 16.4`



5 minute break!

Stand up

Move around

Stretch!

05:00

Polishing your charts

1. Scales
2. Annotations
3. Colors
4. Fonts
5. Rmd tricks

Color is hard

How do I know what colors look good together?

Use the color wheel

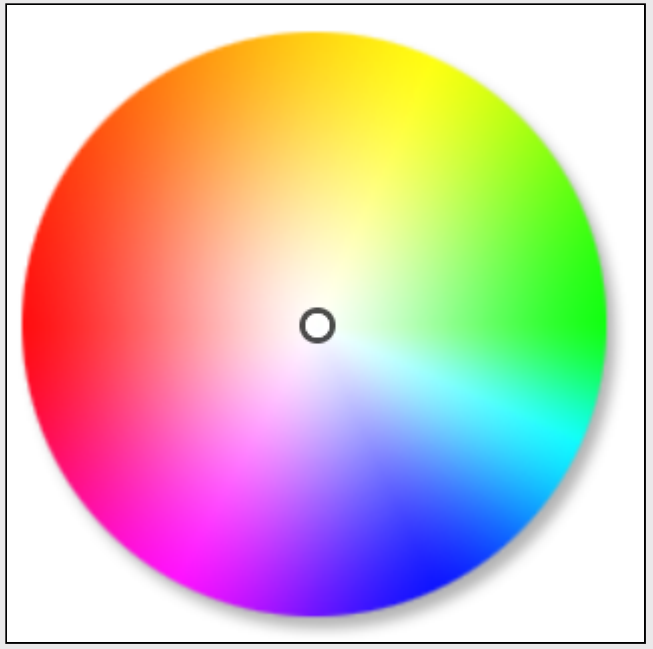


Image from this color wheel tool

How do I know what colors look good together?

Use the color wheel

1. **Complementary**: High contrast

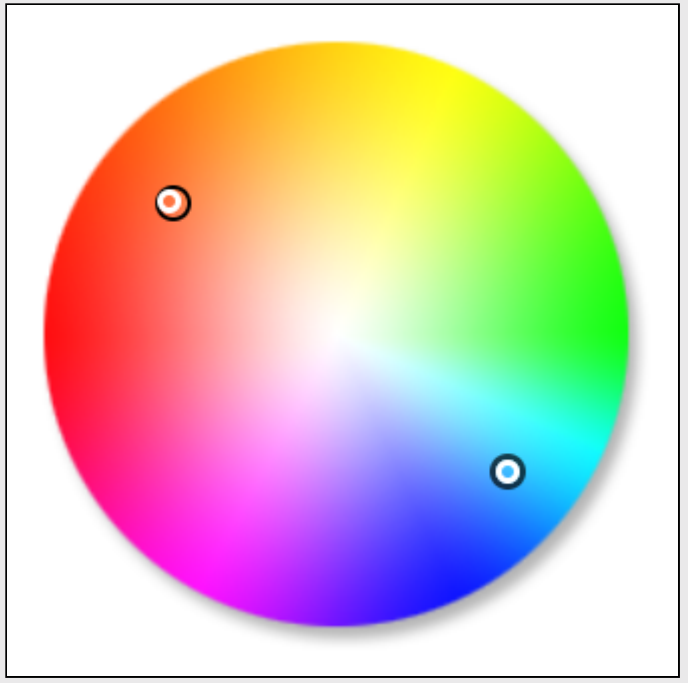


Image from this color wheel tool

How do I know what colors look good together?

Use the color wheel

1. Complementary: High contrast
2. **Analogous**: Calm, harmonious

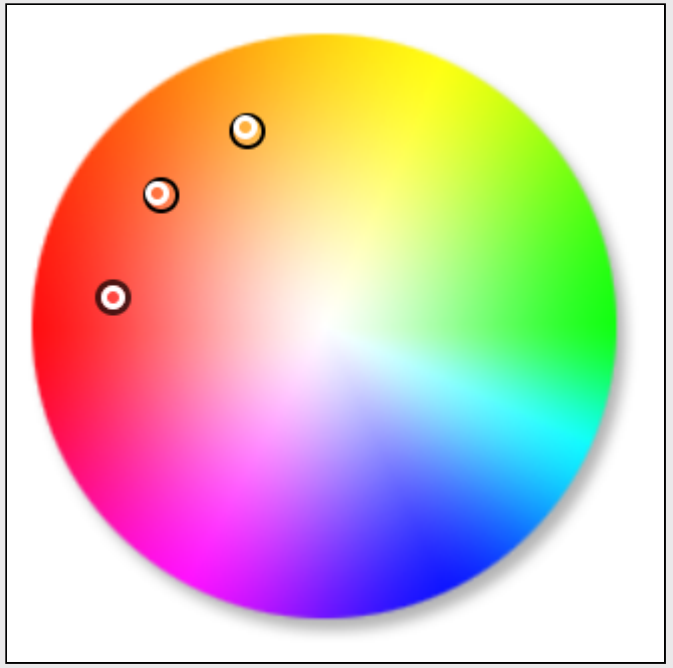


Image from this color wheel tool

How do I know what colors look good together?

Use the color wheel

1. Complementary: High contrast
2. Analogous: Calm, harmonious
3. **Triadic**: Vibrant, contrast

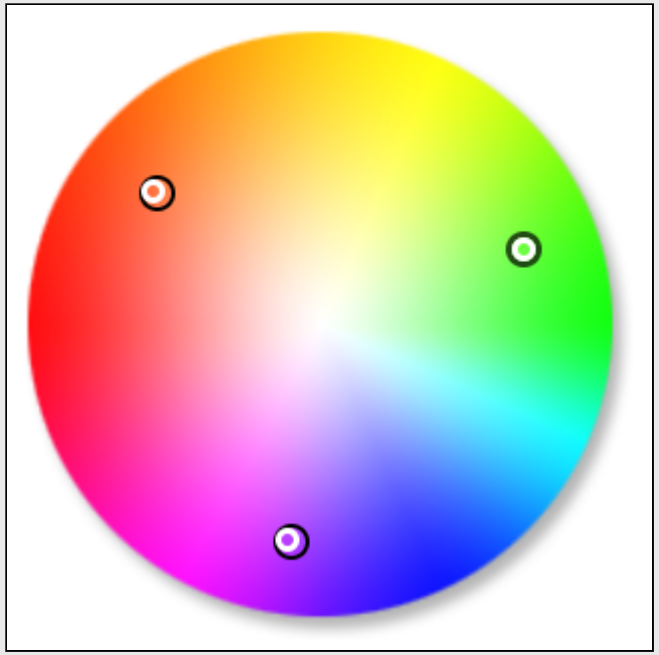


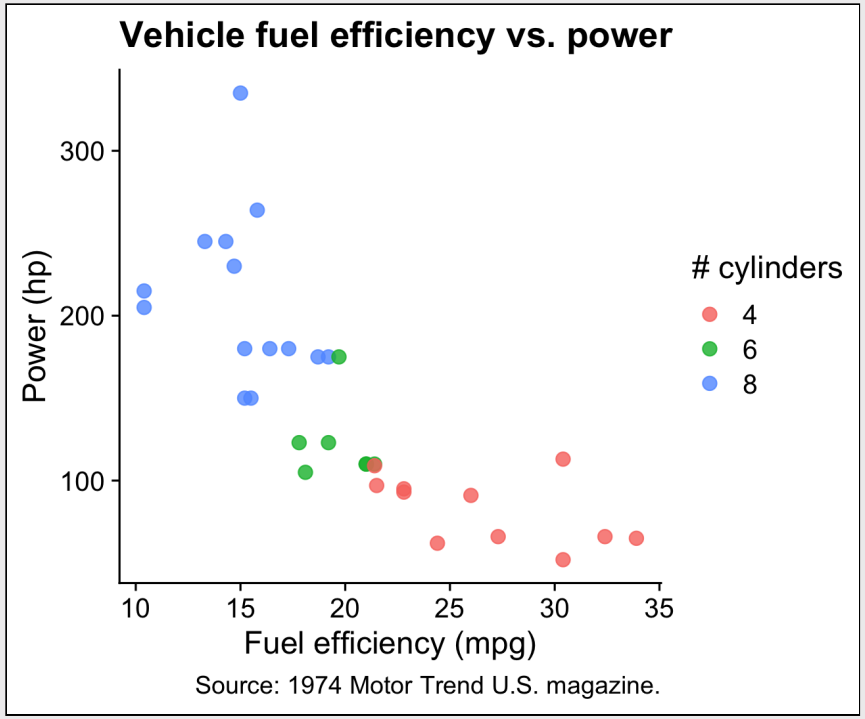
Image from this color wheel tool

Steal colors with the eye dropper tool

Using your own colors

Map color to variable

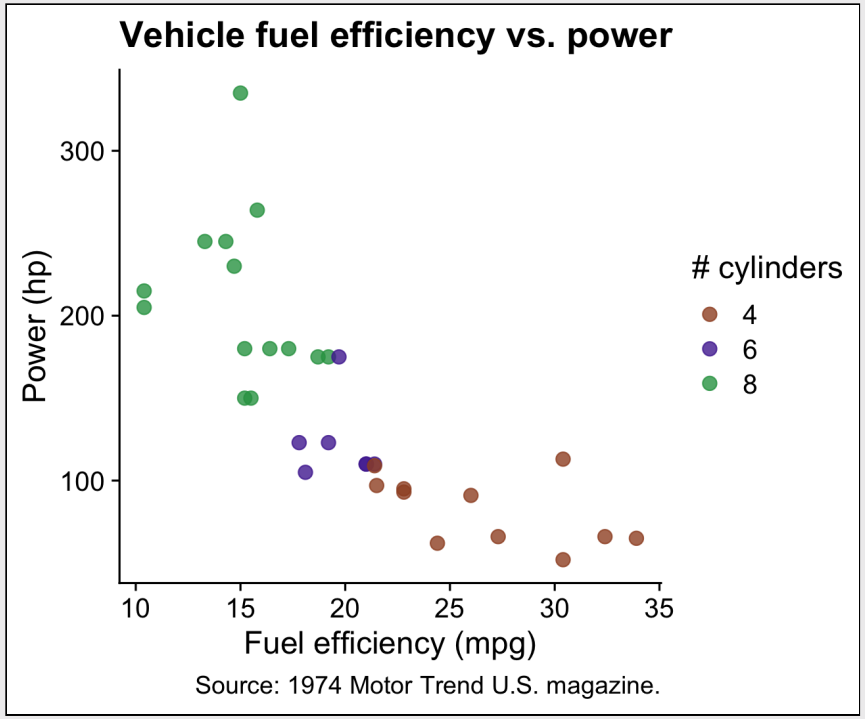
```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
    geom_point(aes(color = as.factor(cyl)),
               alpha = 0.8, size = 3) +
    theme_half_open(font_size = 16) +
    labs(x = "Fuel efficiency (mpg)",
         y = "Power (hp)",
         color = '# cylinders',
         title = "Vehicle fuel efficiency vs. power",
         caption = "Source: 1974 Motor Trend U.S. magazine.")
```



Using your own colors

Map color to variable

```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
    geom_point(aes(color = as.factor(cyl)),
               alpha = 0.8, size = 3) +
    theme_half_open(font_size = 16) +
    labs(x = "Fuel efficiency (mpg)",
         y = "Power (hp)",
         color = '# cylinders',
         title = "Vehicle fuel efficiency vs. power",
         caption = "Source: 1974 Motor Trend U.S. magazine.")
```



Manually change colors

```
mpg_plot +
    scale_color_manual(values = c(
        '#a0522d', '#522da0', '#2da052'))
```

Choose colors carefully

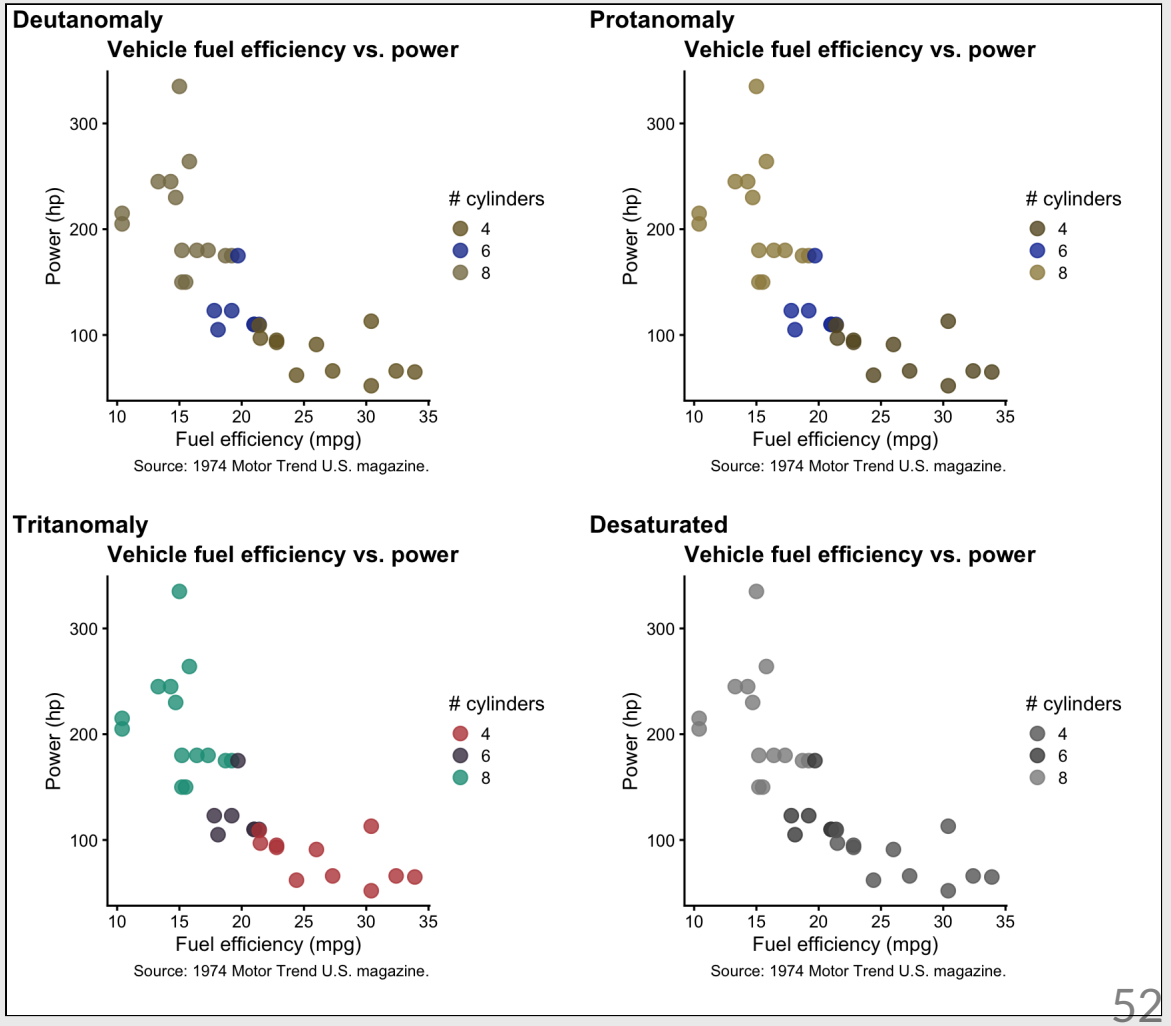
Manually change colors

```
mpg_plot_mycolors <- mpg_plot +
    theme_half_open(font_size = 10) +
    scale_color_manual(values = c(
        '#a0522d', '#522da0', '#2da052'))
```

Simulate color blindness with colorblindr

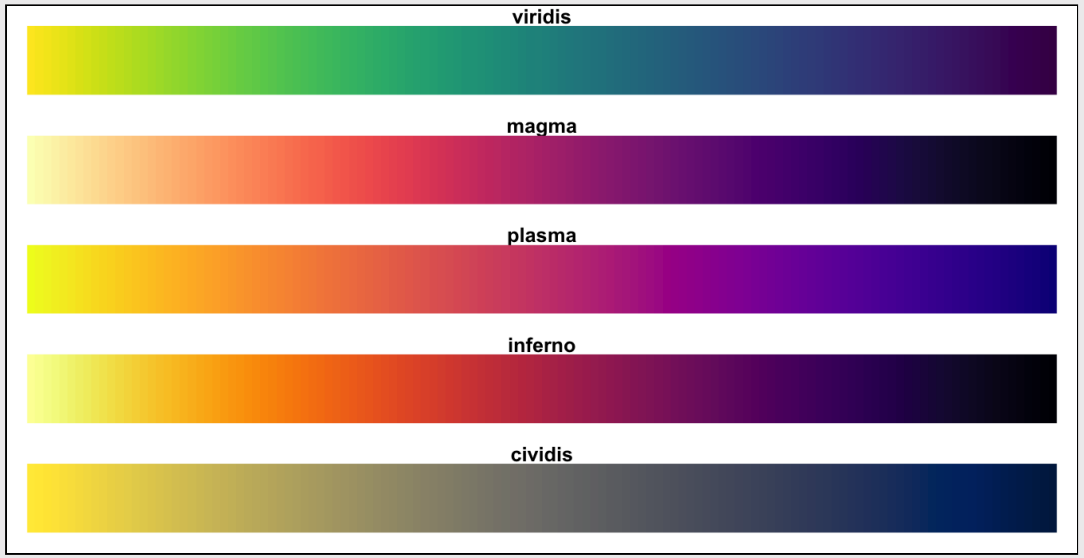
```
library(colorblindr)

cvd_grid(mpg_plot_mycolors)
```

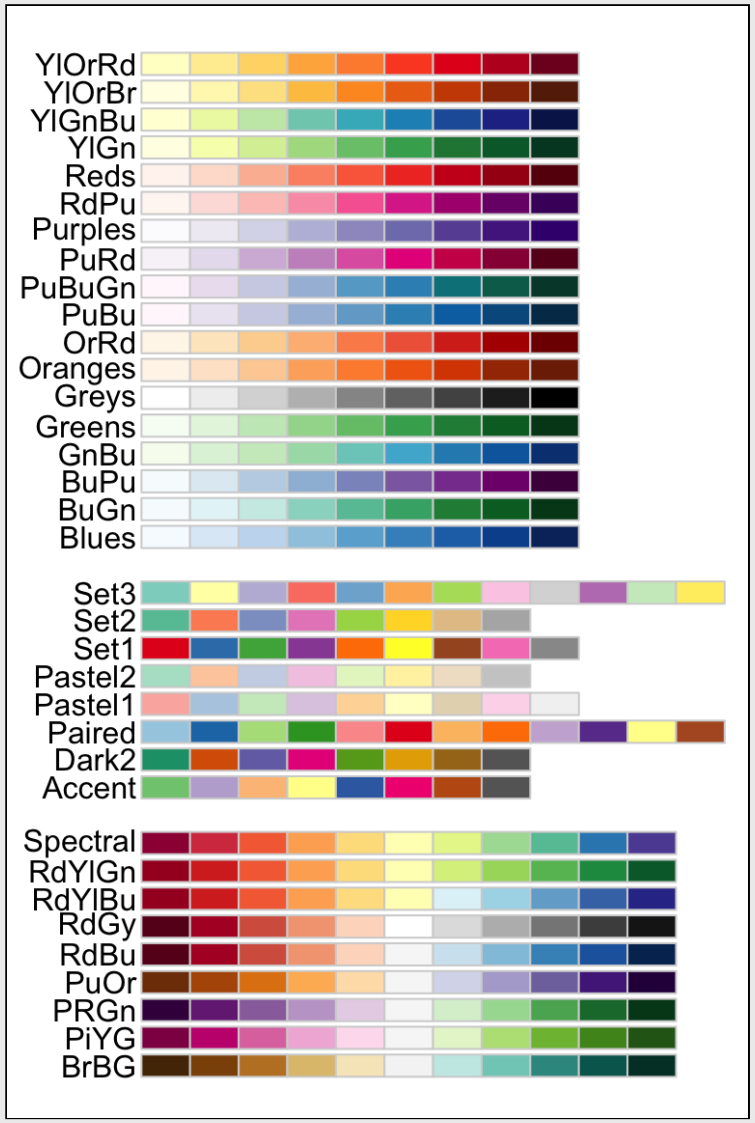


Use palettes

Viridis



ColorBrewer



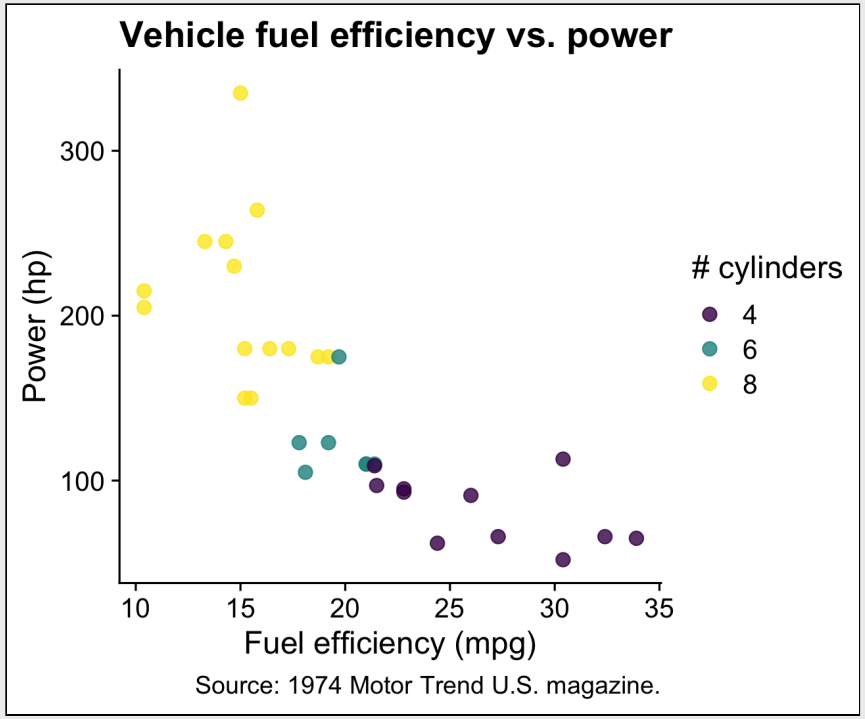
viridis palettes

Map color to variable

```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
    geom_point(aes(color = as.factor(cyl)),
               alpha = 0.8, size = 3) +
    theme_half_open(font_size = 16) +
    labs(x = "Fuel efficiency (mpg)",
         y = "Power (hp)",
         color = '# cylinders',
         title = "Vehicle fuel efficiency vs. power",
         caption = "Source: 1974 Motor Trend U.S. magazine.")
```

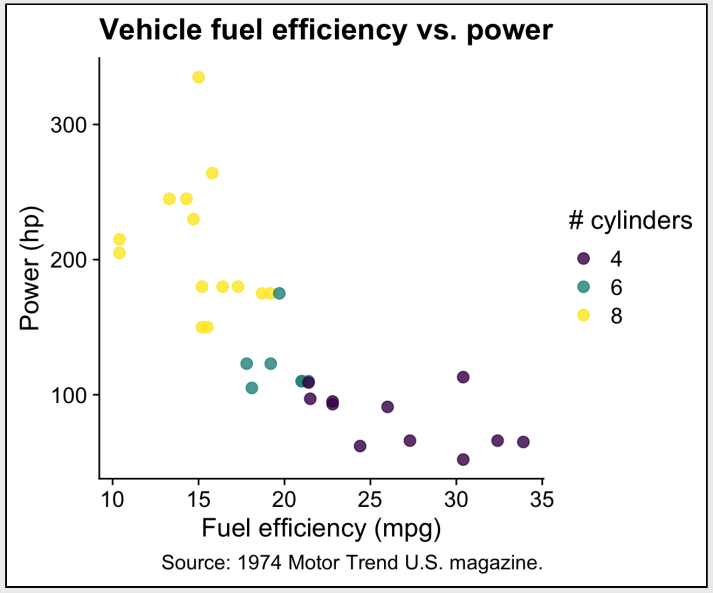
Use viridis colors

```
mpg_plot +
    scale_color_viridis(discrete = TRUE)
```

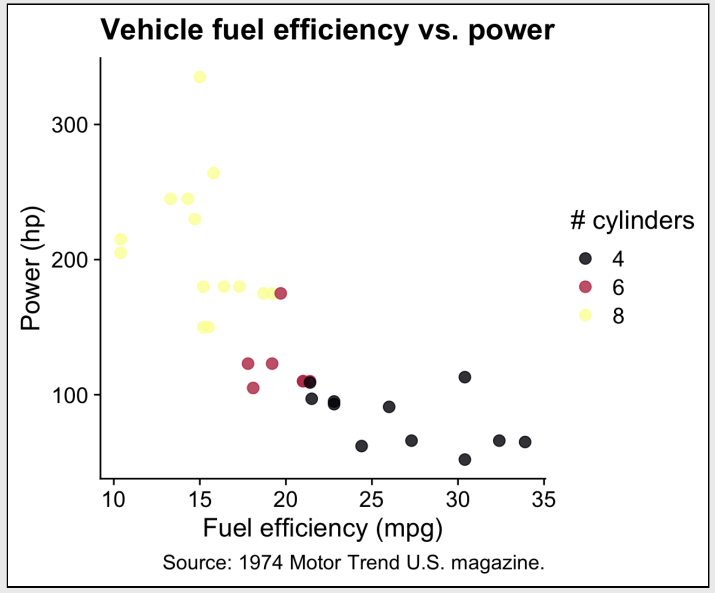


viridis palettes

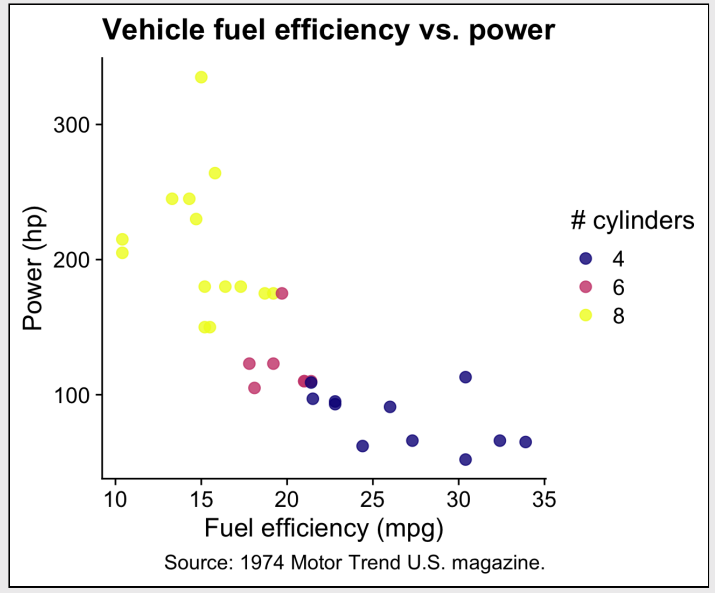
```
mpg_plot +
    scale_color_viridis(
        discrete = TRUE,
        option = 'viridis')
```




```
mpg_plot +
    scale_color_viridis(
        discrete = TRUE,
        option = 'inferno')
```



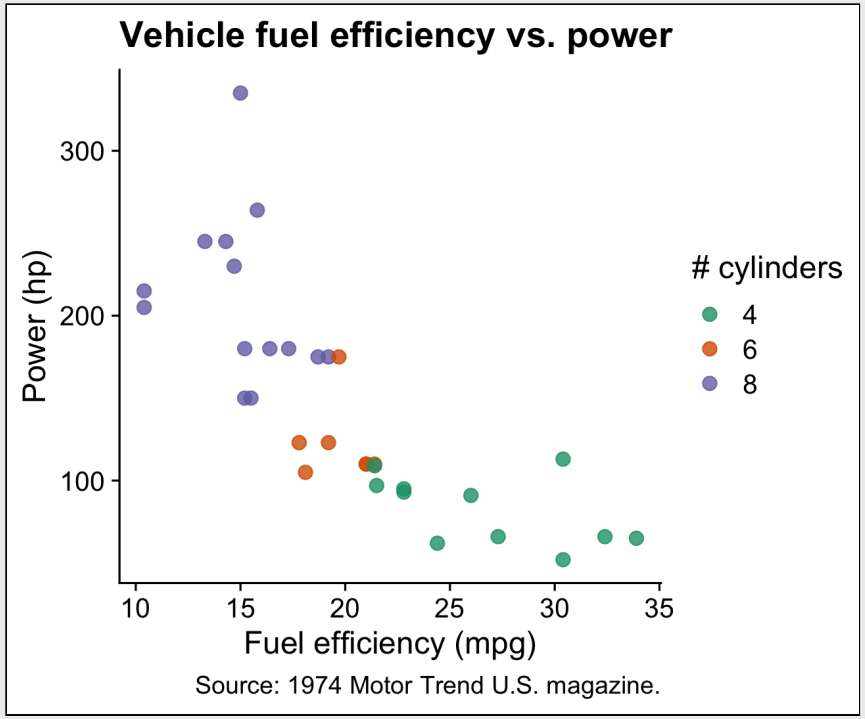

```
mpg_plot +
    scale_color_viridis(
        discrete = TRUE,
        option = 'plasma')
```



ColorBrewer palettes

Map color to variable

```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
    geom_point(aes(color = as.factor(cyl)),
               alpha = 0.8, size = 3) +
    theme_half_open(font_size = 16) +
    labs(x = "Fuel efficiency (mpg)",
         y = "Power (hp)",
         color = '# cylinders',
         title = "Vehicle fuel efficiency vs. power",
         caption = "Source: 1974 Motor Trend U.S. magazine.")
```

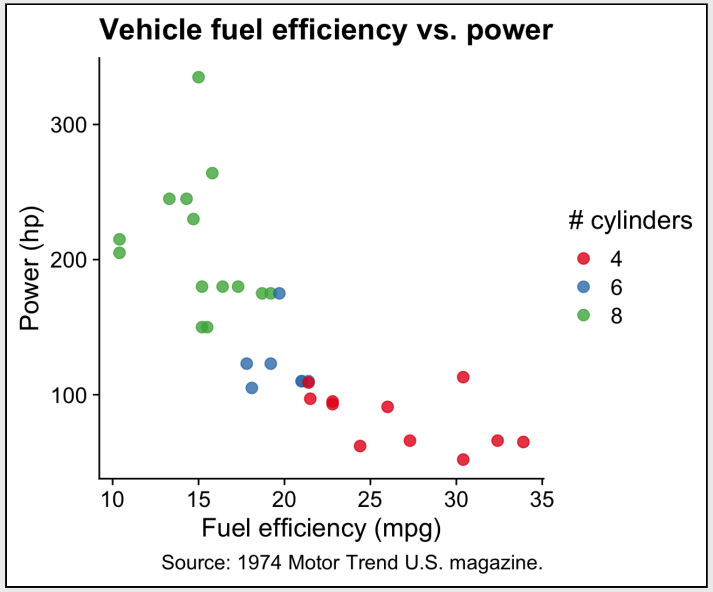


Use "Dark2" palette

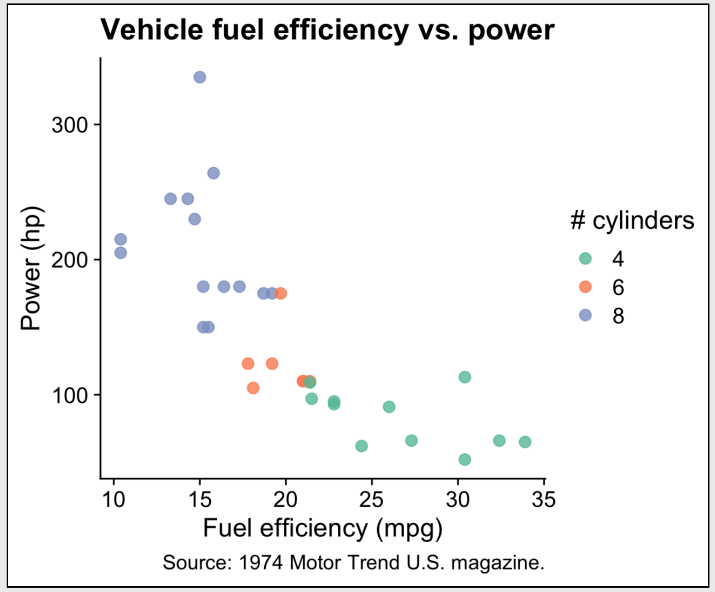
```
mpg_plot +
    scale_color_brewer(palette = 'Dark2')
```

ColorBrewer palettes

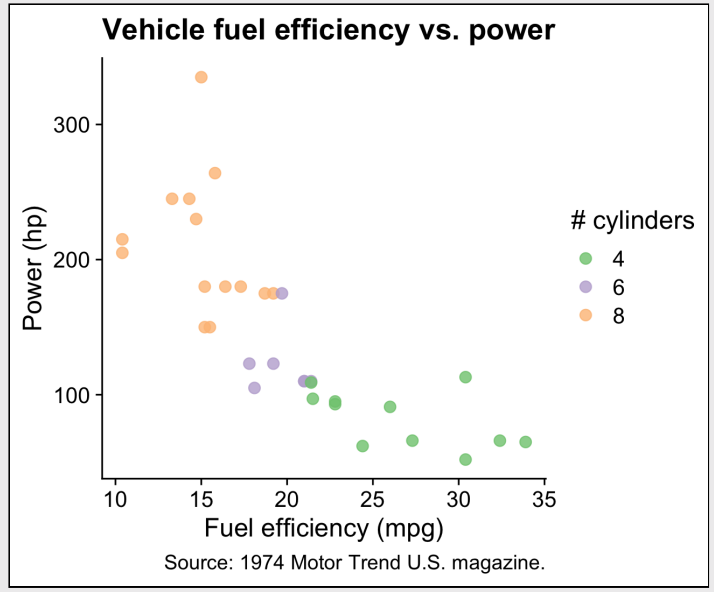
```
mpg_plot +
    scale_color_brewer(
        palette = 'Set1')
```




```
mpg_plot +
    scale_color_brewer(
        palette = 'Set2')
```



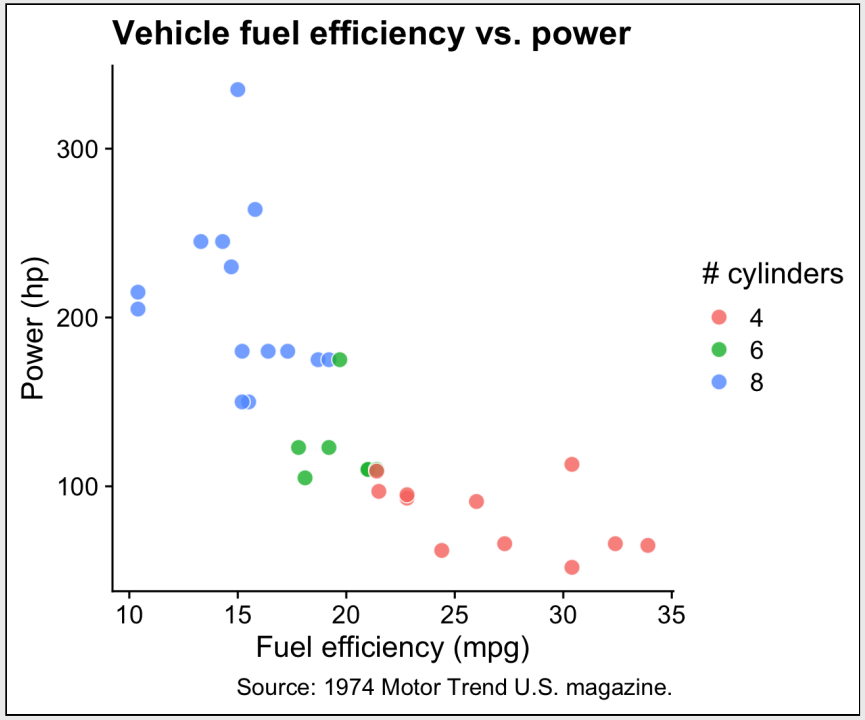

```
mpg_plot +
    scale_color_brewer(
        palette = 'Accent')
```



Consider using both `color` and `fill`

- Map `fill` to variable
- Add "white" line around points with `color`
- Use `shape = 21` for filled points

```
ggplot(mtcars, aes(x = mpg, y = hp)) +
    geom_point(aes(fill = as.factor(cyl)),
               color = 'white', shape = 21,
               size = 3.5, alpha = 0.8) +
    scale_color_brewer(palette = 'Dark2') +
    theme_half_open(font_size = 15) +
    labs(x = "Fuel efficiency (mpg)",
         y = "Power (hp)",
         fill = '# cylinders',
         title = "Vehicle fuel efficiency vs. power",
         caption = "Source: 1974 Motor Trend U.S. magazine.")
```



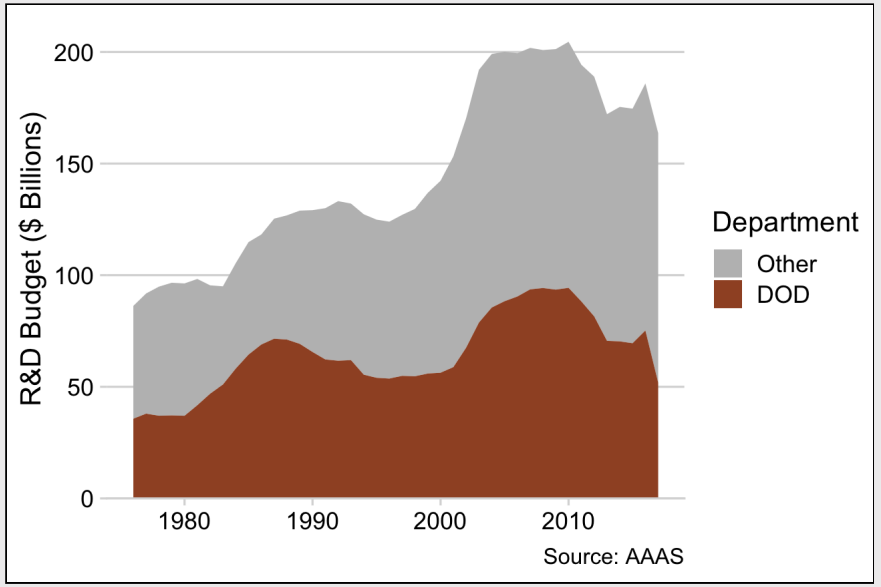
Summarise data

```
federal_spending_summary <- federal_spending %>%
    mutate(department = fct_other(
        department, keep = 'DOD')) %>%
    group_by(department, year) %>%
    summarise(rd_budget = sum(rd_budget) / 10^3) %>%
    ungroup() %>%
    mutate(department = fct_relevel(
        department, c('Other', 'DOD')))
```

Make plot

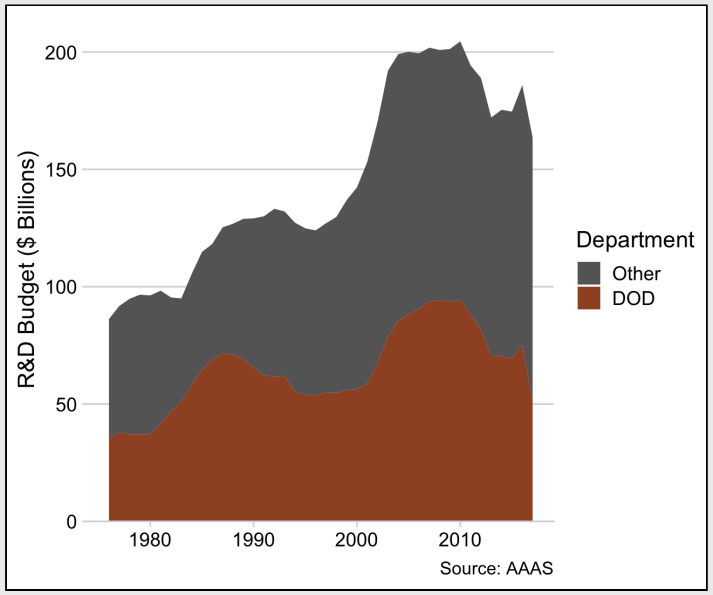
```
dod_spending_plot <- ggplot(federal_spending_summary) +
    geom_area(aes(x = year, y = rd_budget,
                  fill = department)) +
    scale_y_continuous(
        expand = expand_scale(mult = c(0, 0.05))) +
    scale_fill_manual(values = c('grey', 'sienna')) +
    theme_minimal_hgrid() +
    labs(x = NULL,
         y = 'R&D Budget ($ Billions)',
         fill = 'Department',
         caption = 'Source: AAAS')
```

`grey` = "Other"

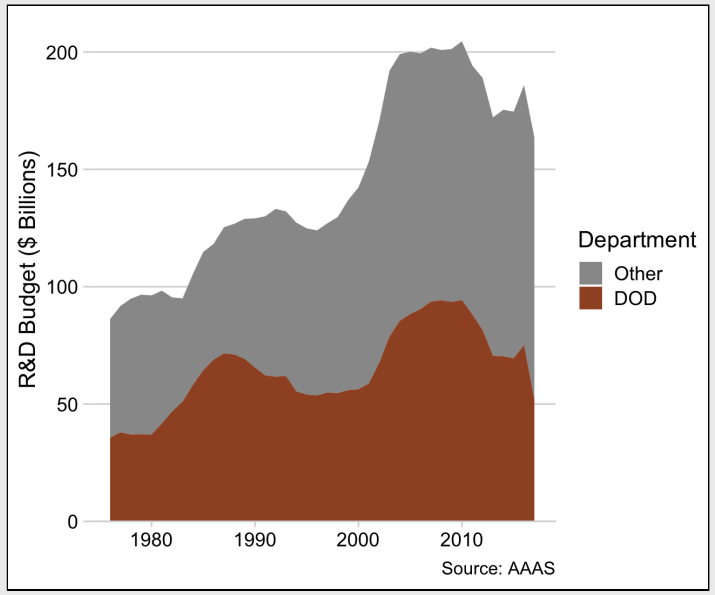


grey = "Other"

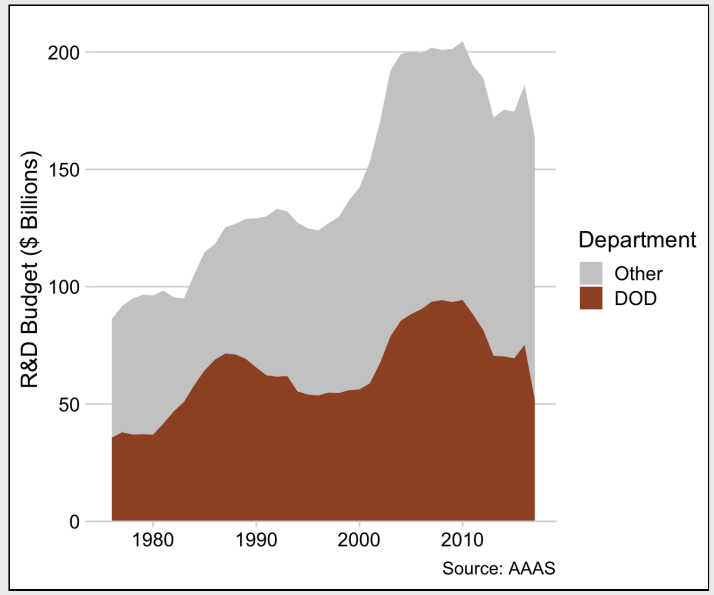
```
dod_spending_plot +
    scale_fill_manual(
        values = c('grey40', 'sienna'))
```




```
dod_spending_plot +
    scale_fill_manual(
        values = c('grey60', 'sienna'))
```




```
dod_spending_plot +
    scale_fill_manual(
        values = c('grey80', 'sienna'))
```

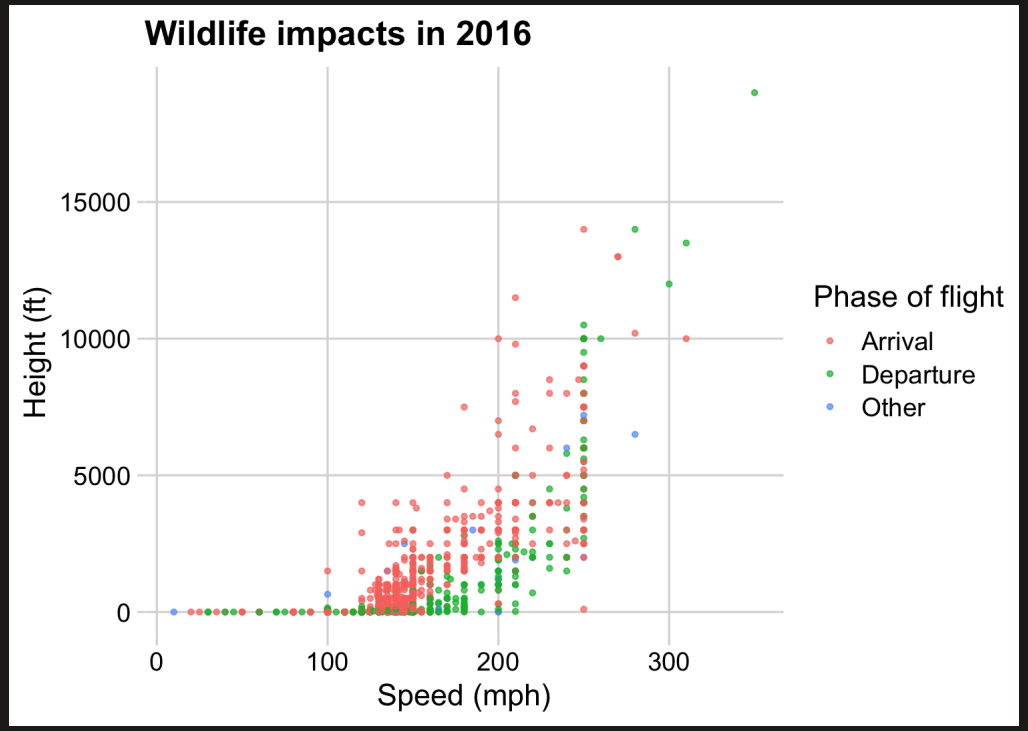


Your turn

15:00

Follow these steps to make variations of this plot with different colors

1. Change the colors to the `"RdYlBu"` ColorBrewer palette.
2. Change the colors to the `"inferno"` palette from the **viridis** library.
3. Use the "eye dropper" tool in Google Chrome to select a color from a website
4. Use your chosen color and this color wheel tool to find a triadic color palette.
5. Change the colors to your custom triadic palette.

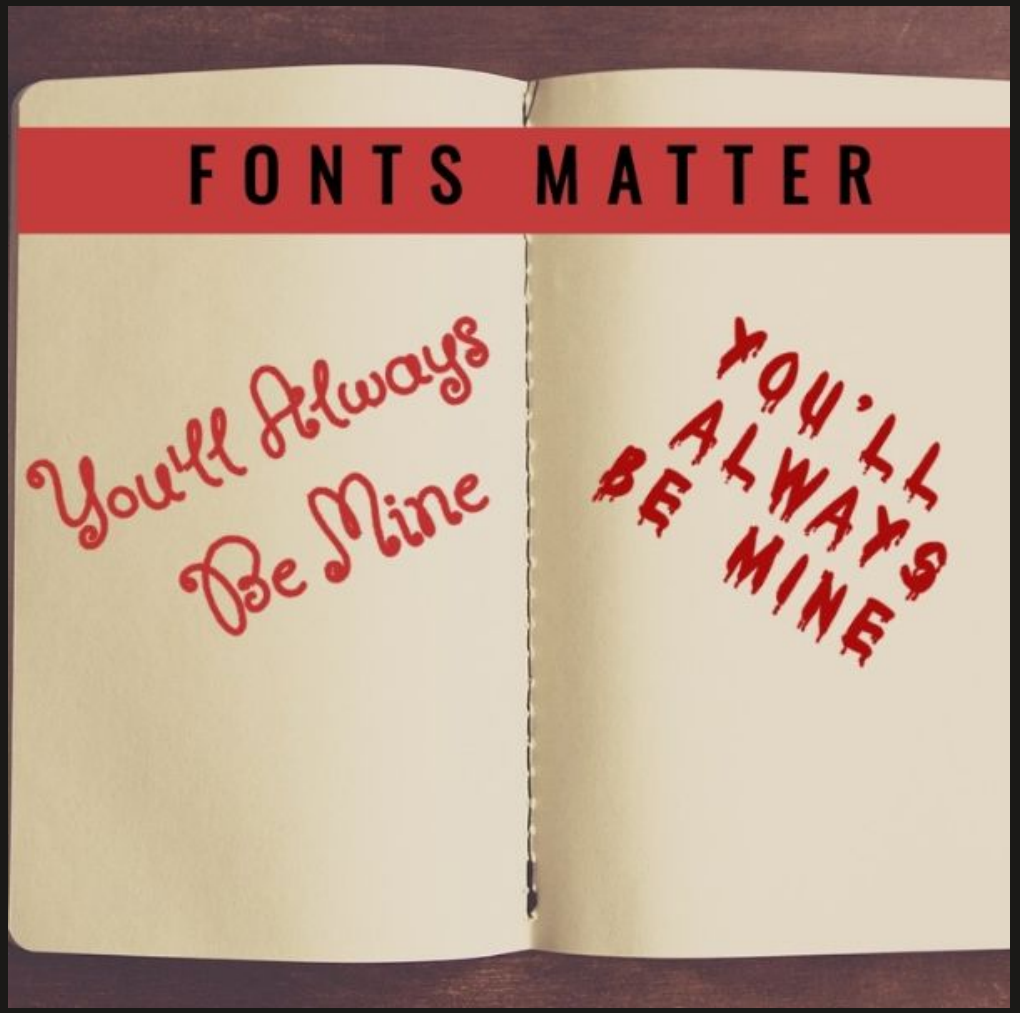


Polishing your charts

1. Scales
2. Annotations
3. Colors
4. Fonts
5. Rmd tricks

"Fast Taco"

"Mega Flicks"


FONTS MATTER
You'll Always Be Mine
YOU'LL ALWAYS BE MINE

Best resource on fonts:

practicaltypography.com

Font families you should consider using

Roboto

Source

Fira

Alegreya

Lato

Exo

Download:

- Individually from https://fonts.google.com/
- All of these with this zip file

Use fonts to create **hierarchy**

```
# Hierarchy

## Hierarchy

### Hierarchy

#### Hierarchy
```

Hierarchy

Hierarchy

Hierarchy

Hierarchy

Title
This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Size

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Weight

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Color

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Spacing

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Typeface

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Title

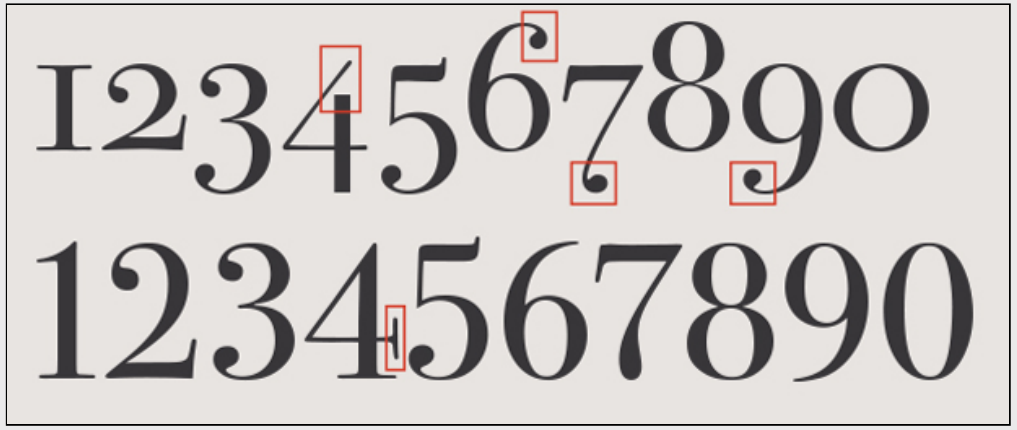
Subtitle

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Use fonts with **same-height** numbers

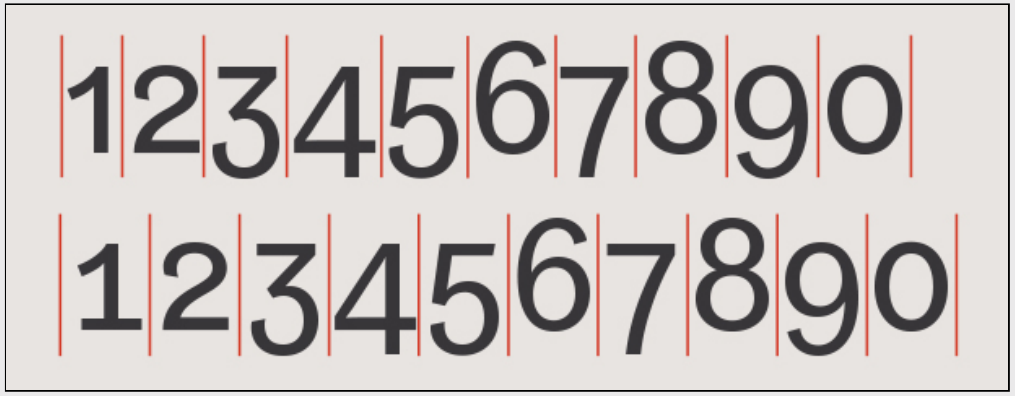
"Oldstyle" (bad)

Lining (good)



Use fonts with **same-width** numbers

"Proportional" (bad)



"Tabular" (good)

How to customize fonts in ggplot

1. Change the whole theme

For "Base R" themes, use `base_family`:

```
theme_minimal(base_family = "Roboto Condensed")
```

```
theme_bw(base_family = "Roboto Condensed")
```

For "cowplot" themes, use `font_family`:

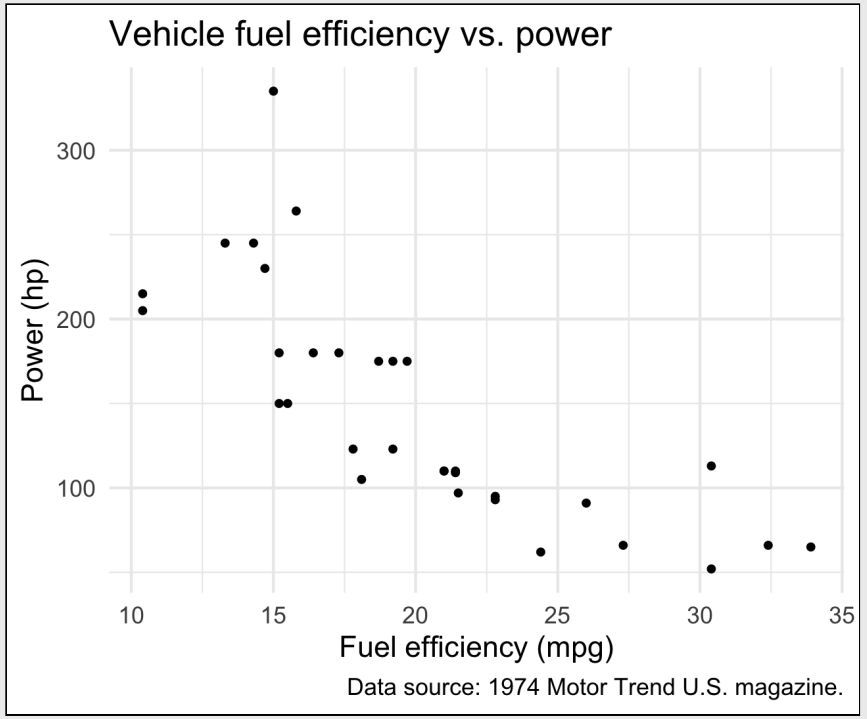
```
theme_half_open(font_family = "Roboto Condensed")
```

```
theme_minimal_grid(font_family = "Roboto Condensed")
```

1. Change the whole theme font

Make the base plot

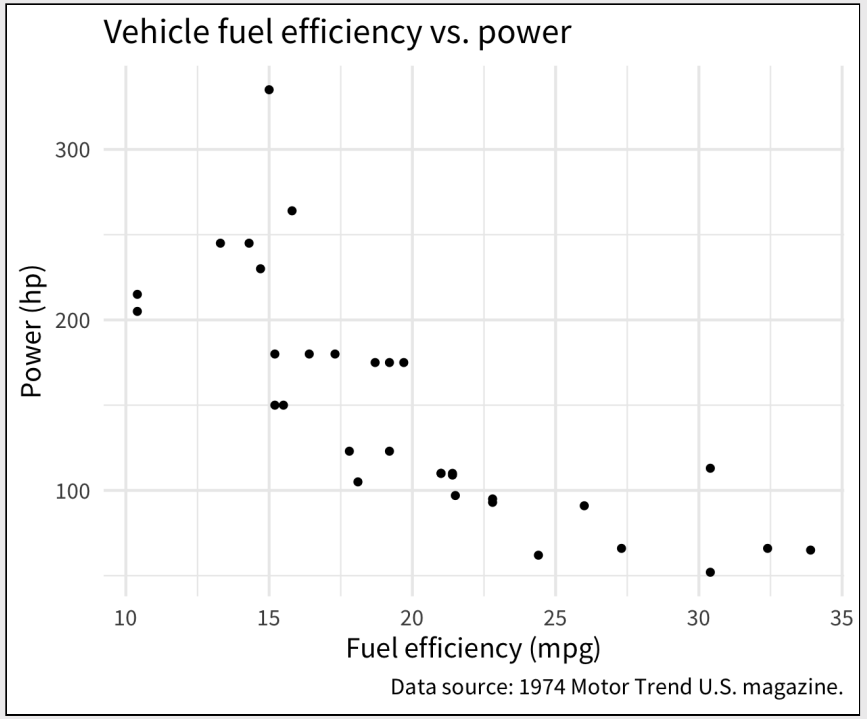
```
mpg_plot <- ggplot(mtcars) +
  geom_point(aes(x = mpg, y = hp)) +
  theme_minimal(base_size = 15) +
  labs(x = "Fuel efficiency (mpg)",
       y = "Power (hp)",
       title = "Vehicle fuel efficiency vs. power",
       caption = "Data source: 1974 Motor Trend U.S. magazine.")
```



1. Change the whole theme font

Use `base_family` with base themes

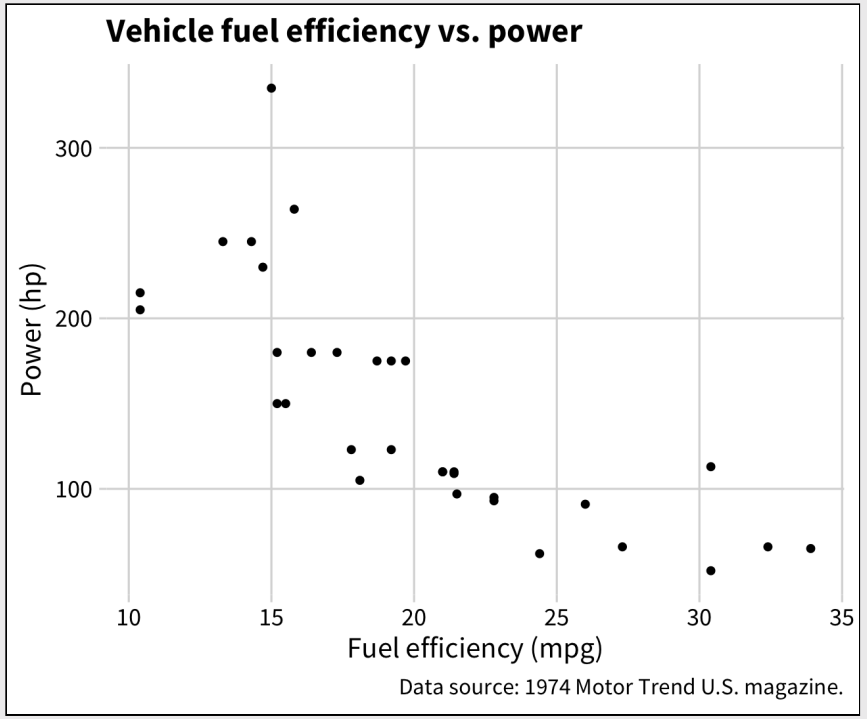
```
mpg_plot <- ggplot(mtcars) +
  geom_point(aes(x = mpg, y = hp)) +
  theme_minimal(base_family = 'Source Sans Pro',
                base_size = 15) +
  labs(x = "Fuel efficiency (mpg)",
       y = "Power (hp)",
       title = "Vehicle fuel efficiency vs. power",
       caption = "Data source: 1974 Motor Trend U.S. magazine.")
```



1. Change the whole theme font

Use `font_family` with cowplot themes

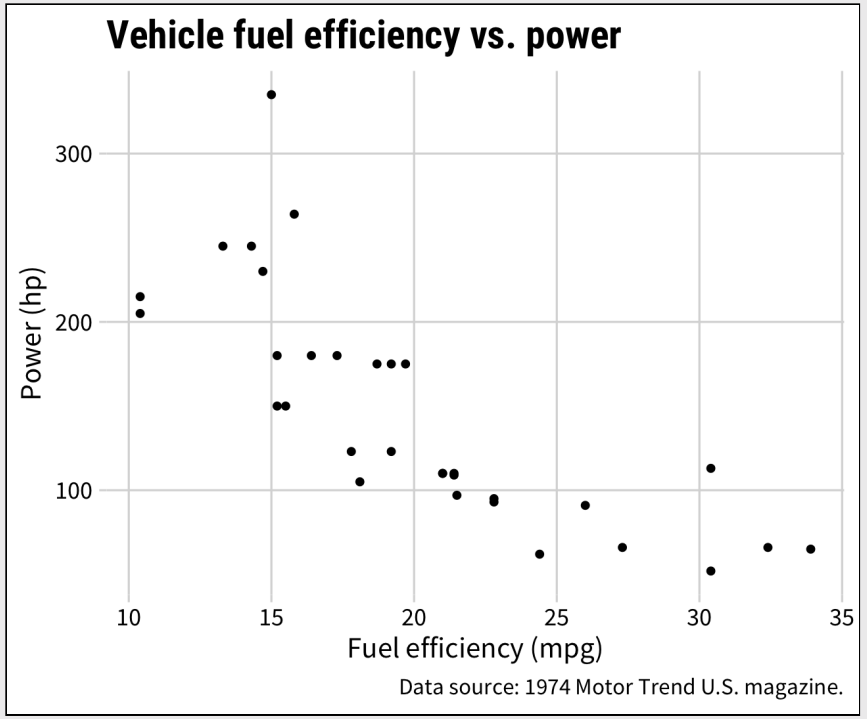
```
mpg_plot <- ggplot(mtcars) +
  geom_point(aes(x = mpg, y = hp)) +
  theme_minimal_grid(font_family = 'Source Sans Pro',
                     font_size = 15) +
  labs(x = "Fuel efficiency (mpg)",
       y = "Power (hp)",
       title = "Vehicle fuel efficiency vs. power",
       caption = "Data source: 1974 Motor Trend U.S. magazine.")
```



2. Adjust theme elements with `element_text()`

Use `font_family` with cowplot themes

```
mpg_plot <- ggplot(mtcars) +
  geom_point(aes(x = mpg, y = hp)) +
  theme_minimal_grid(font_family = 'Source Sans Pro',
                     font_size = 15) +
  theme(plot.title = element_text(family = "Roboto Condensed",
                                  size = 20)) +
  labs(x = "Fuel efficiency (mpg)",
       y = "Power (hp)",
       title = "Vehicle fuel efficiency vs. power",
       caption = "Data source: 1974 Motor Trend U.S. magazine.")
```

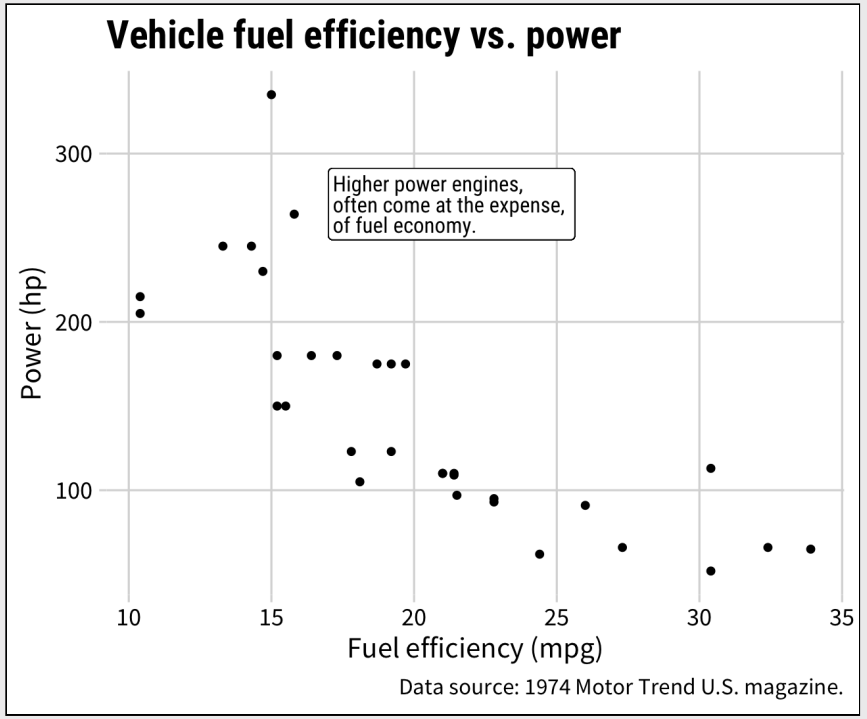


3. Adjust annotations:

`geom_text()`, `geom_label()`, and `annotate()`

```
label <- "Higher power engines,
often come at the expense,
of fuel economy."

mpg_plot +
    geom_label(aes(x = 17, y = 270, label = label),
               lineheight = .8, hjust = 0,
               family = 'Roboto Condensed')
```



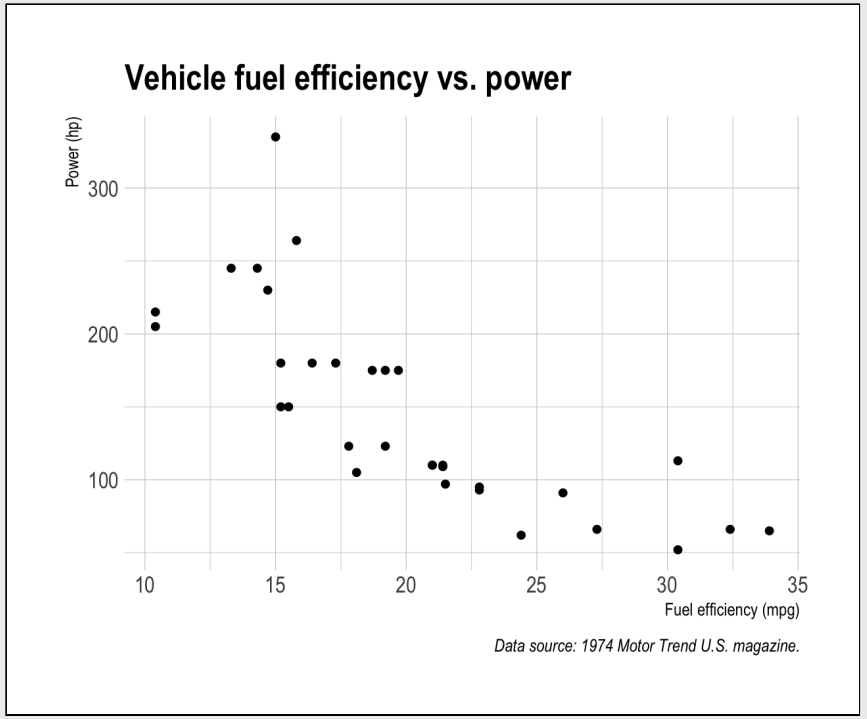
The hrbrthemes package:

Great themes + great fonts

```
library(hrbrthemes)

mpg_plot <- ggplot(mtcars) +
  geom_point(aes(x = mpg, y = hp)) +
  labs(x = "Fuel efficiency (mpg)",
       y = "Power (hp)",
       title = "Vehicle fuel efficiency vs. power",
       caption = "Data source: 1974 Motor Trend U.S. magazi

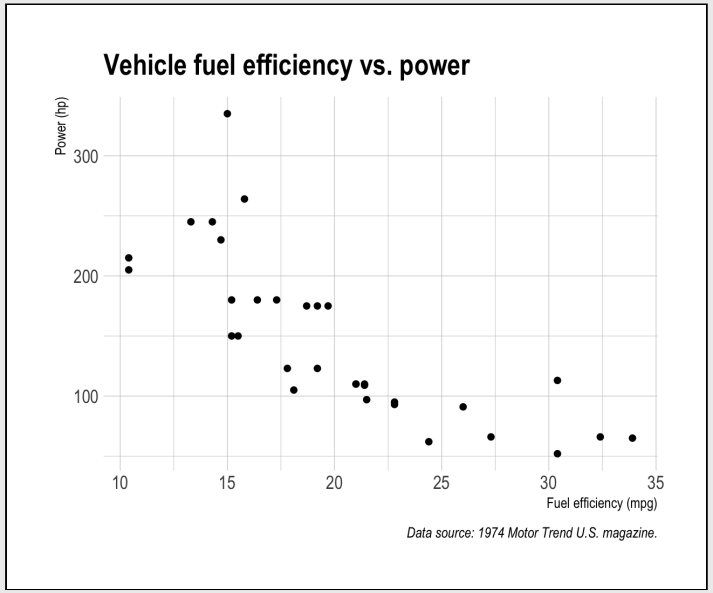
mpg_plot +
  theme_ipsum()
```



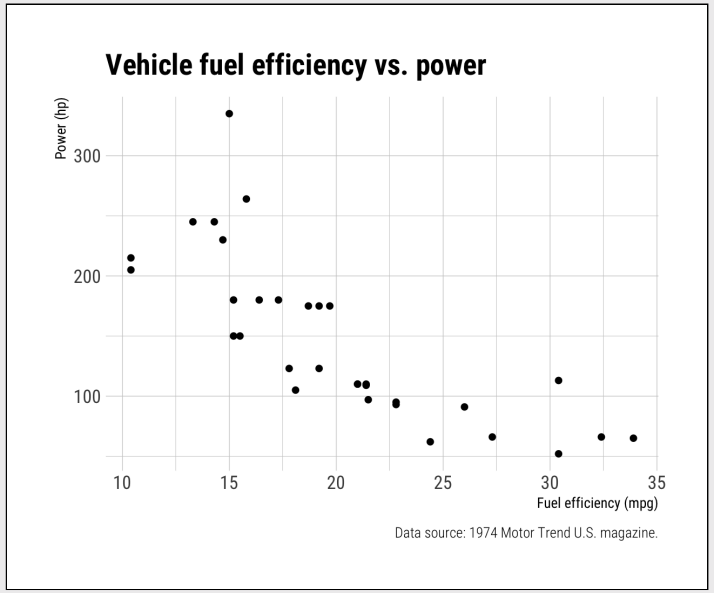
The hrbrthemes package:

Great themes + great fonts

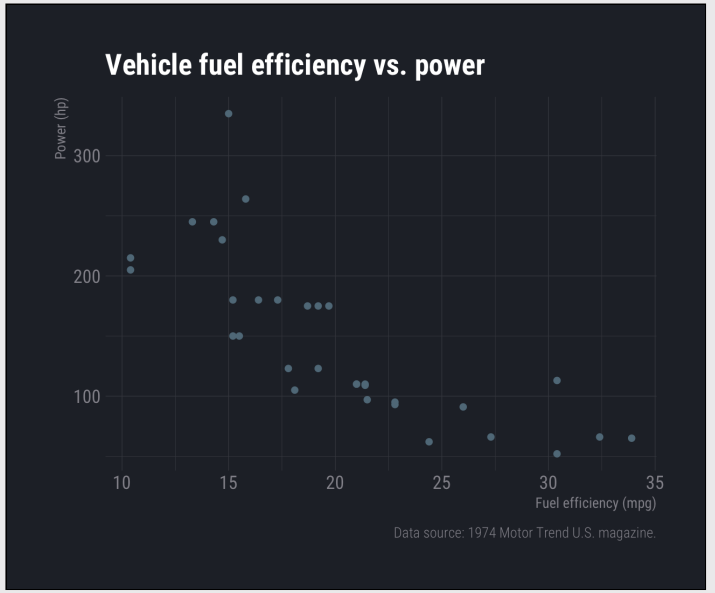
```
mpg_plot +
  theme_ipsum()
```




```
mpg_plot +
  theme_ipsum_rc()
```




```
mpg_plot +
  theme_ft_rc()
```



Your turn

15:00

Use the `milk_summary_dumbbell` data frame to create the following chart.

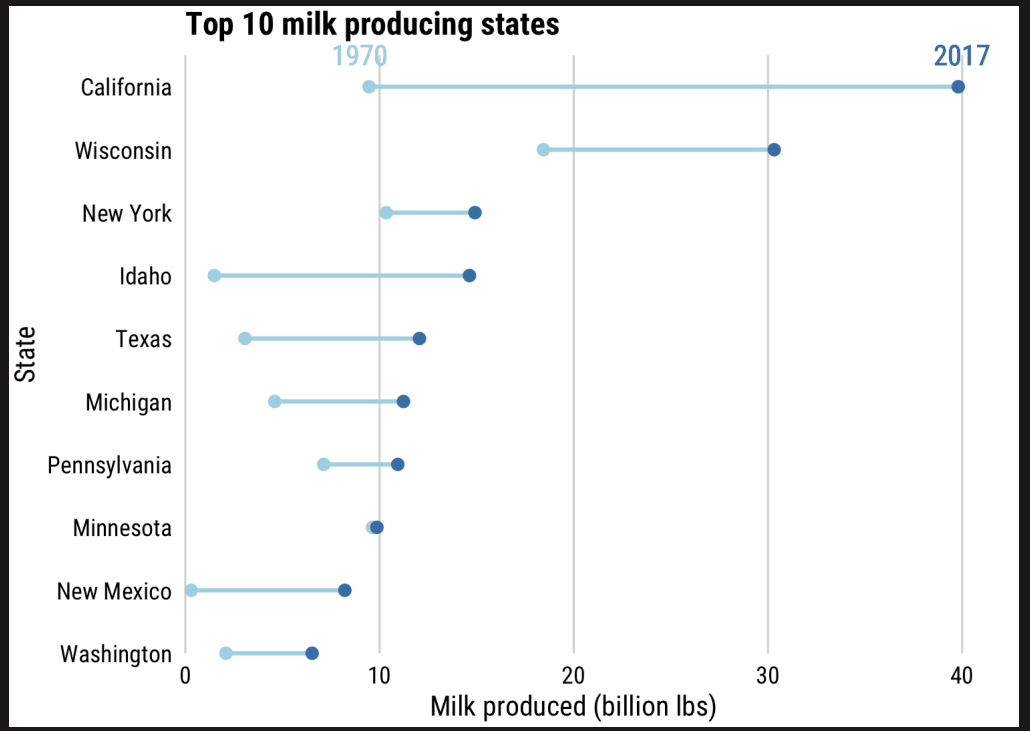
The main font is `'Roboto Condensed'`.

Once you've recreated the plot, try other fonts and themes, such as:

- The `'Source Sans Pro'` font.
- The `'Lato'` font.
- The `theme_ipsum()` theme from the `hrbrthemes` library.

Hints:

- Use `geom_text()` or `annotate()` to insert the year labels:
  - 1970: `x = 9, y = 10.5`
  - 2017: `x = 40, y = 10.5`



Polishing your charts

1. Scales
2. Annotations
3. Colors
4. Fonts
5. Rmd tricks

Use themes to change global "look"

Check out Pimp my RMD, by Yan Holtz

Change theme in YAML header:

```
title: "your title"
output:
    html_document:
        theme: united
        highlight: tango
```

Preview themes here

Quick practice

Try changing the theme of today's notes Rmd file:

Themes: -

Highlighting styles: -

Extra practice

Your turn

Use the `lotr_summary` data frame to create the following chart.

Hints:

- The main font is `"Fira Sans Condensed"`
- For the `geom_label()`, use these points:
  - `x = 0.6`
  - `y = 2100`
- For the `geom_curve()`, use these points:
  - `x = 1.2`
  - `xend = 1`
  - `y = 1300`
  - `yend = 200`