Colors/Themes + Inclusive Design Principles

Maithreyi Gopalan

Week 4

Agenda

- Learn about C.R.A.P
- Color basics
  - 3 basic ways color is used
- Color blindness
- Some common problems with color use
- Quick discussion of palettes
- Quick discussion of themes
- Lab 4
- Q & A with Dr. Daniel Anderson, who pioneered this EDS specialization and class!

Learning Objectives

- Learn a bit more about good/inclusive design principles
- Understand different types of color palettes
  - ...and when you should use one versus another
- Understand and be able to effectively evaluate concerns related to color blindness
- Be able to fluently change colors/fills within ggplot

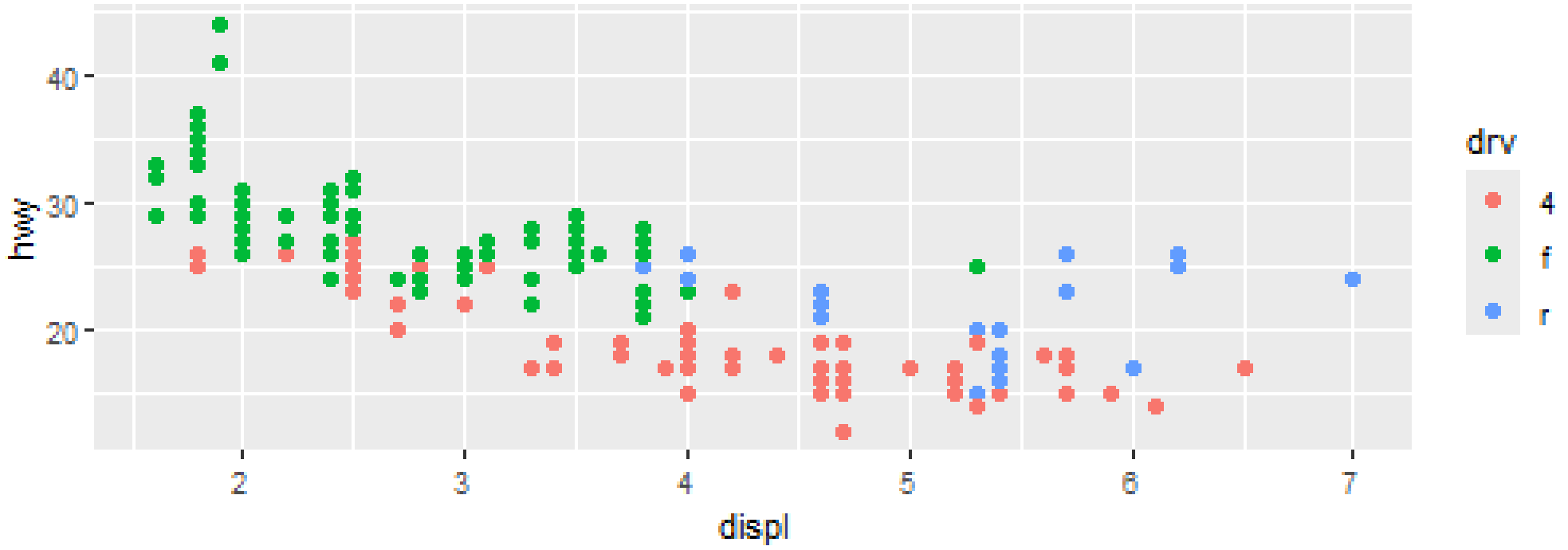
Universal principles

- Contrast
- Repetition
- Alignment
- Proximity
  - Pioneered by Robin Patricia Williams

- These design principles apply everywhere!
- Graphic design, art, music, architecture… and graphs!

Is that gray background okay?

```
theme_set(theme_gray())
ggplot(mpg, aes(x = displ, y = hwy, color = drv)) +
  geom_point(size = 2)
```



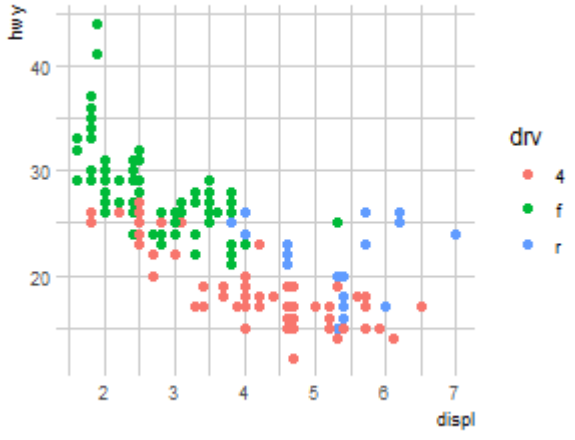

```
  theme(gray)
```

Applying CRAP to ggplot

- We can follow CRAP principles to make big improvements to our plots
- Claus Wilke's chapter covers lots of these graph-specific principles
- We can apply these principles to ggplot plots

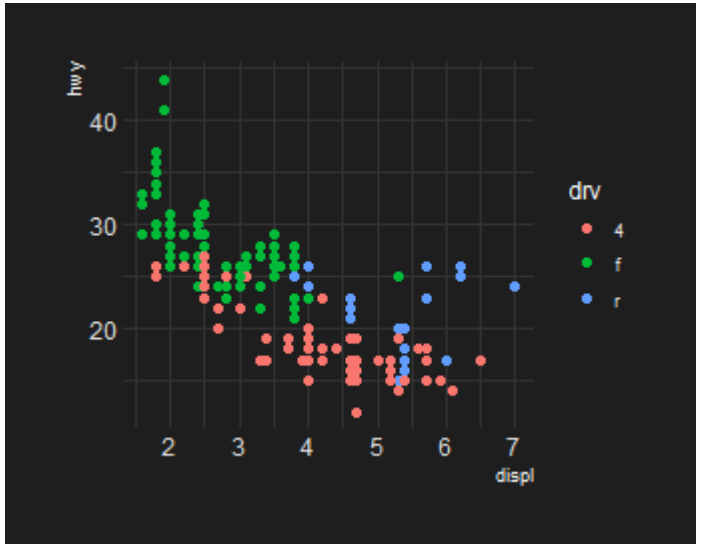
Like this!

```
library(hrbrthemes)

ggplot(mpg, aes(x = displ, y = hwy,
                color = drv)) +
  geom_point(size = 2) +
  theme_ipsum_ps()
```



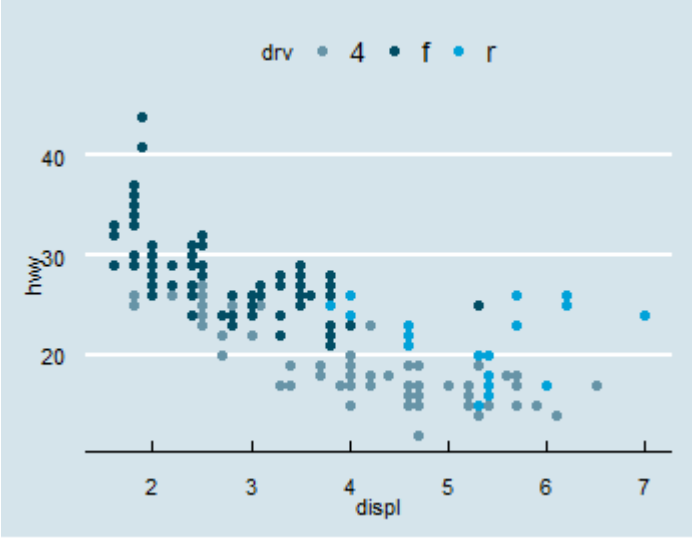
And this!

```
library(hrbrthemes)

ggplot(mpg, aes(x = displ, y = hwy,
                color = drv)) +
  geom_point(size = 2) +
  theme_modern_rc()
```



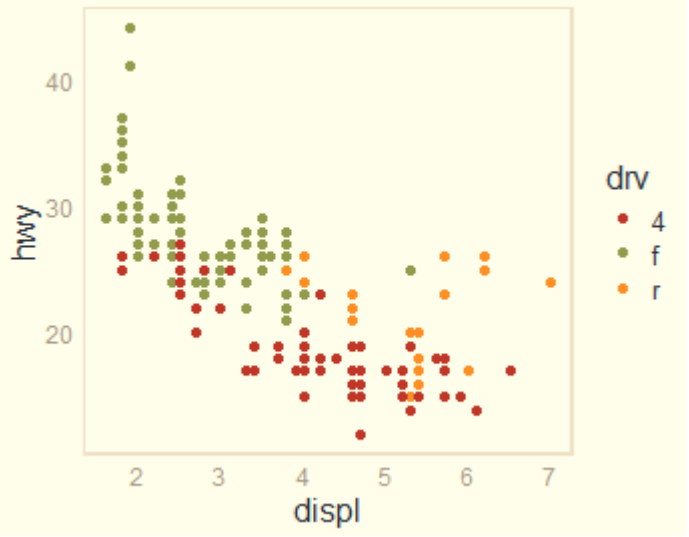
Or this!

```
library(ggthemes)

ggplot(mpg, aes(x = displ, y = hwy,
                color = drv)) +
  geom_point(size = 2) +
  scale_color_economist() +
  theme_economist()
```



And even this!

```
#remotes::install_github("gadenbuie/ggpomologic
library(ggpomological)

ggplot(mpg, aes(x = displ, y = hwy,
                color = drv)) +
  geom_point(size = 2) +
  scale_color_pomological() +
  theme_pomological_fancy()
```



One magic, powerful function

`theme()`

- Depending on time we come back to an anatomy of `theme` the end today or next week!

Before we get too deep into themes of which colors is an essential element...

Some very practical advice about colors

- Keep straight when color is mapped to a variable through `aes` and when it's modifying an element overall
    - Former requires `scale_color_*` or `scale_fill_*` while the latter does not
- Keep straight colors and fills (see former bullet)
- Use advice of others to your advantage (e.g., http://colorbrewer2.org/)

Why color choice matters

Why color choice matters

Another quick example

{rayshader}

3 fundamental uses of color

1. Distinguish groups from each other
2. Represent data values
3. Highlight

Color as a tool to distinguish

Discrete items

- Often no intrinsic order

Qualitative color scale

- Finite number of colors
  - Chosen to maximize distinctness, while also be equivalent
  - Equivalent
    - No color should stand out
    - No impression of order

Some Examples

Okabe Ito

ColorBrewer Dark2

ggplot2 hue

See more about the Okabe Ito palette origins here

How do we use them?

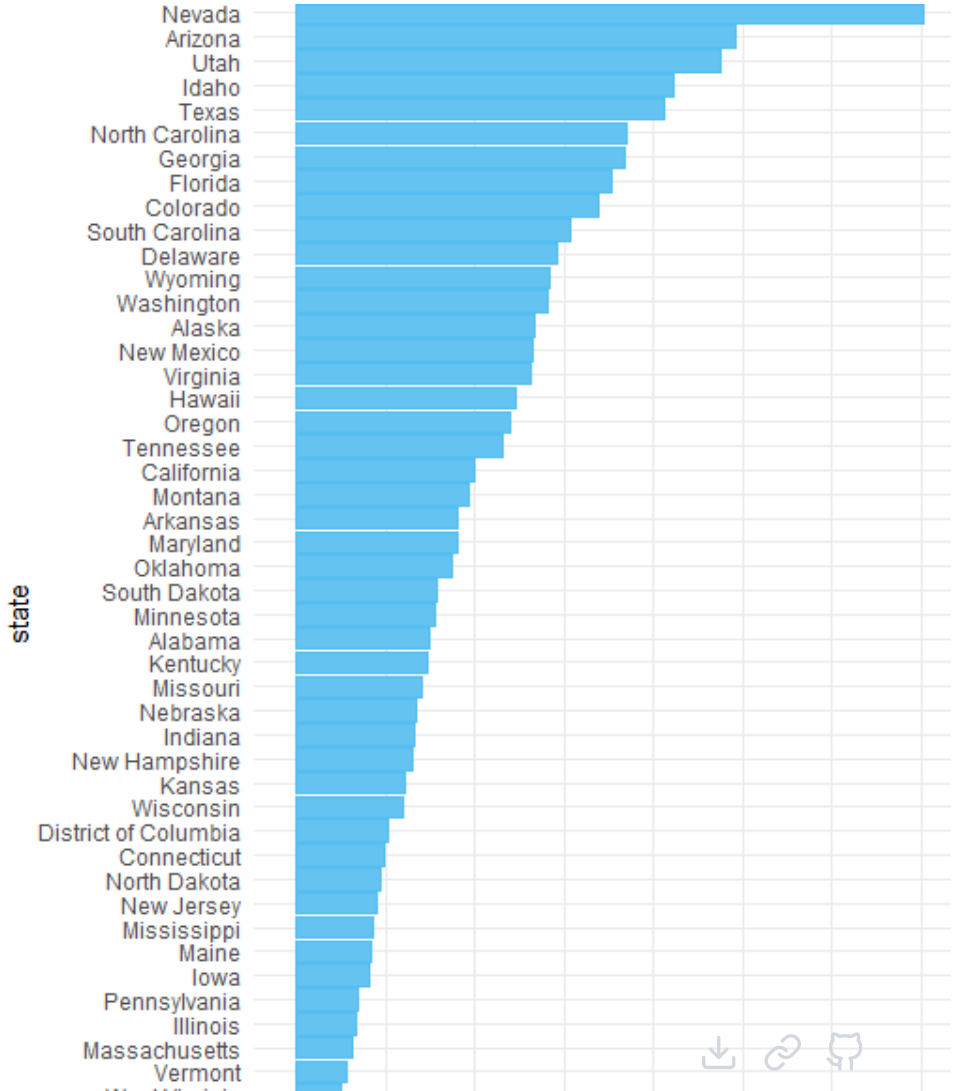
Imagine we have data like this

```
popgrowth_df
```

```
## # A tibble: 51 × 8
##   region division           state          pop2000  pop2010 popgrowth    area fill_color
##   <fct>  <chr>              <fct>            <dbl>    <dbl>     <dbl>   <dbl> <chr>
## 1 West   Mountain           Nevada         1998257  2700551     0.351 109781. West
## 2 West   Mountain           Arizona        5130632  6392017     0.246 113594. West
## 3 West   Mountain           Utah           2233169  2763885     0.238  82170. West
## 4 West   Mountain           Idaho          1293953  1567582     0.211  82643. West
## 5 South  West South Central Texas         20851820 25145561     0.206 261232. South
## 6 South  South Atlantic     North Carolina  8049313  9535483     0.185  48618. South
## # ℹ 45 more rows
```

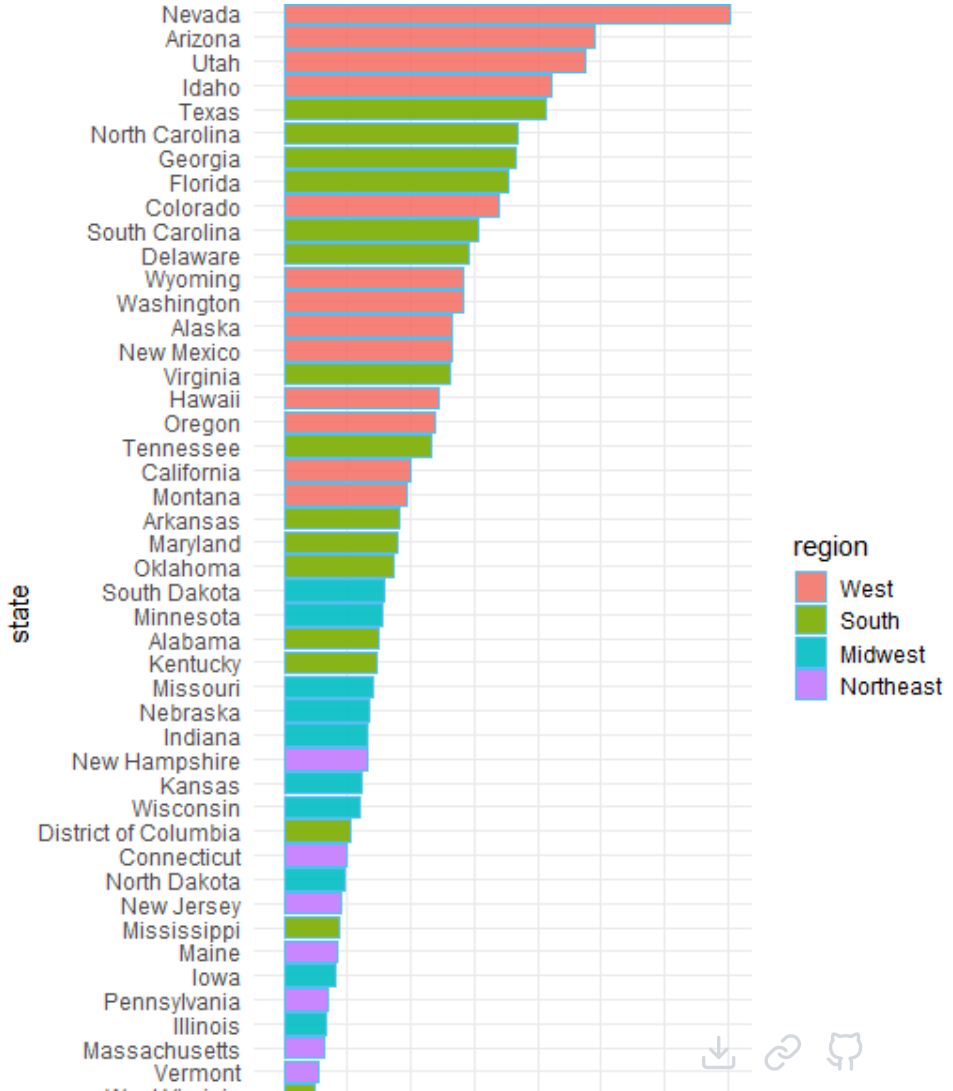
Maybe a plot like this

```
ggplot(popgrowth_df,
       aes(x = popgrowth,
           y = state)) +
  geom_col(alpha = 0.9)
```

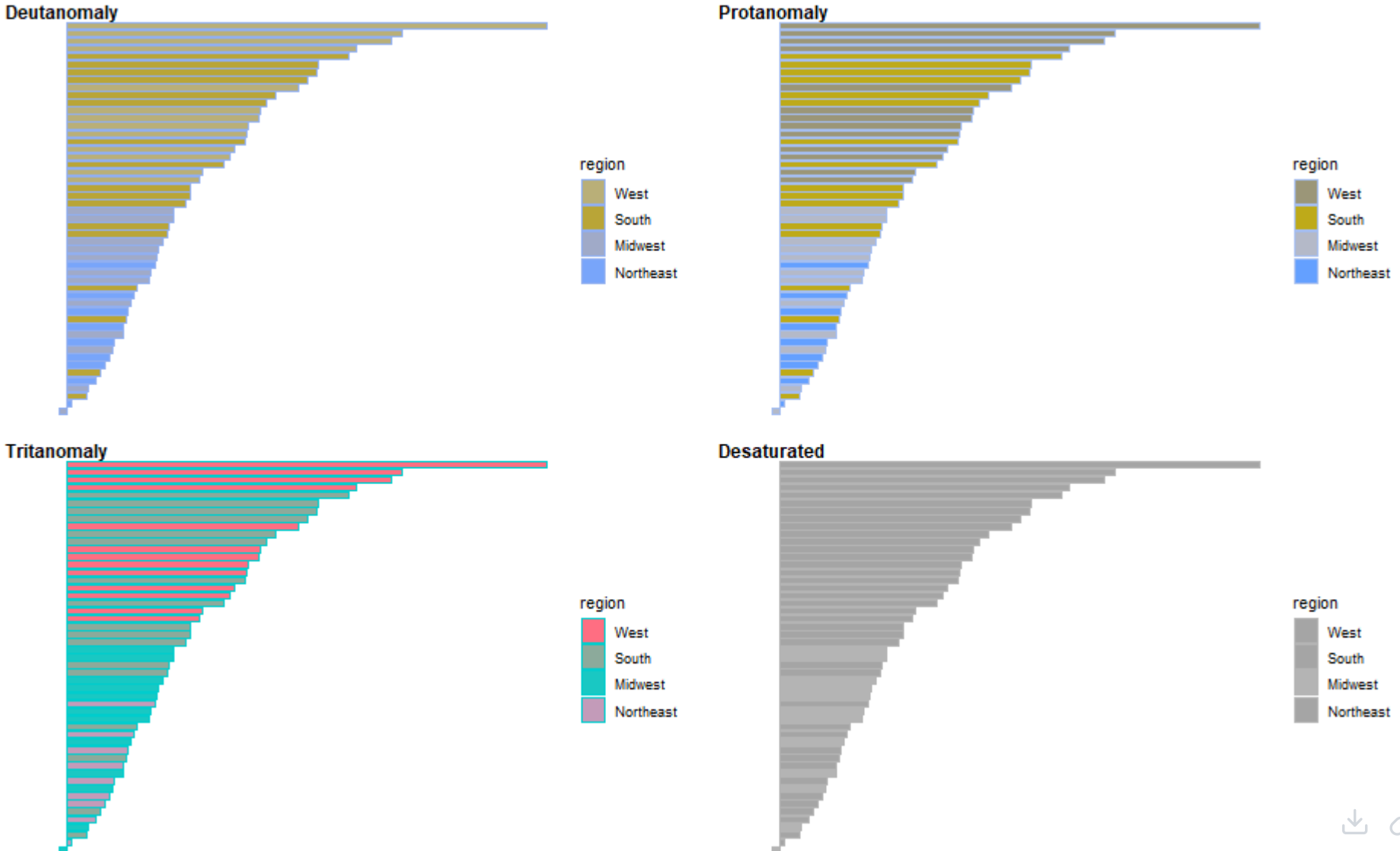


Alternatively, fill by region

```
ggplot(popgrowth_df,
       aes(x = popgrowth,
           y = state)) +
  geom_col(aes(fill = region),
           alpha = 0.9)
```

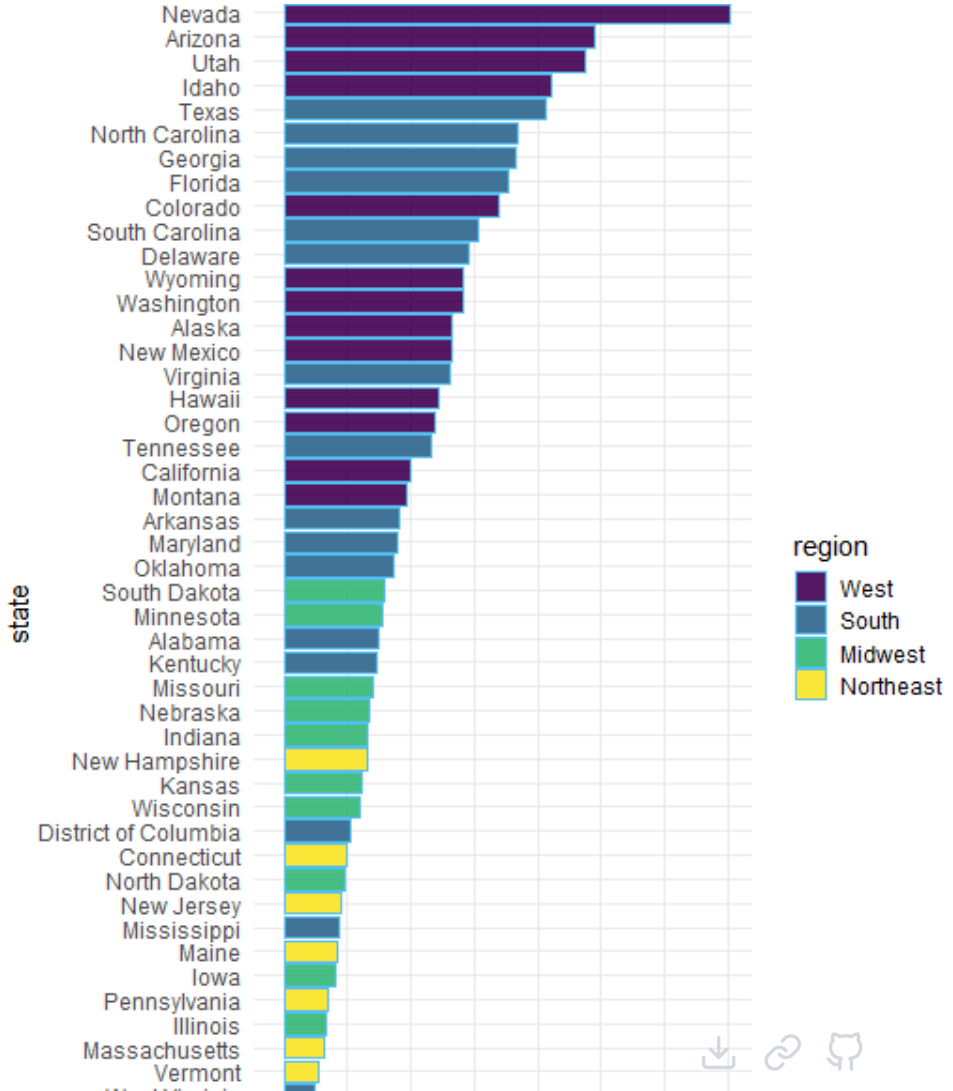


Problem with default palette

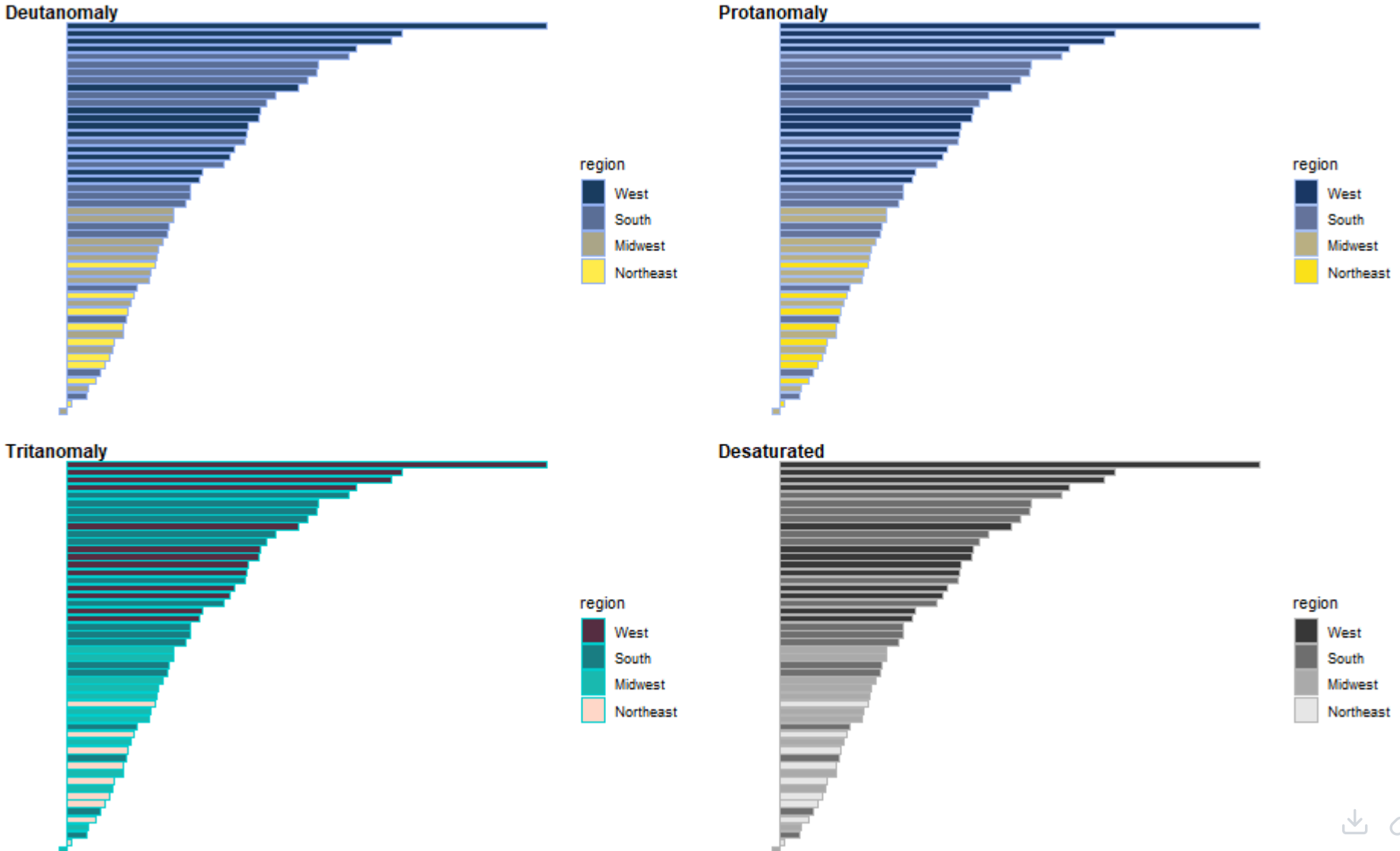


Alternative: viridis

```
ggplot(popgrowth_df,
       aes(x = popgrowth,
           y = state)) +
  geom_col(aes(fill = region),
           alpha = 0.9) +
  scale_fill_viridis_d()
```



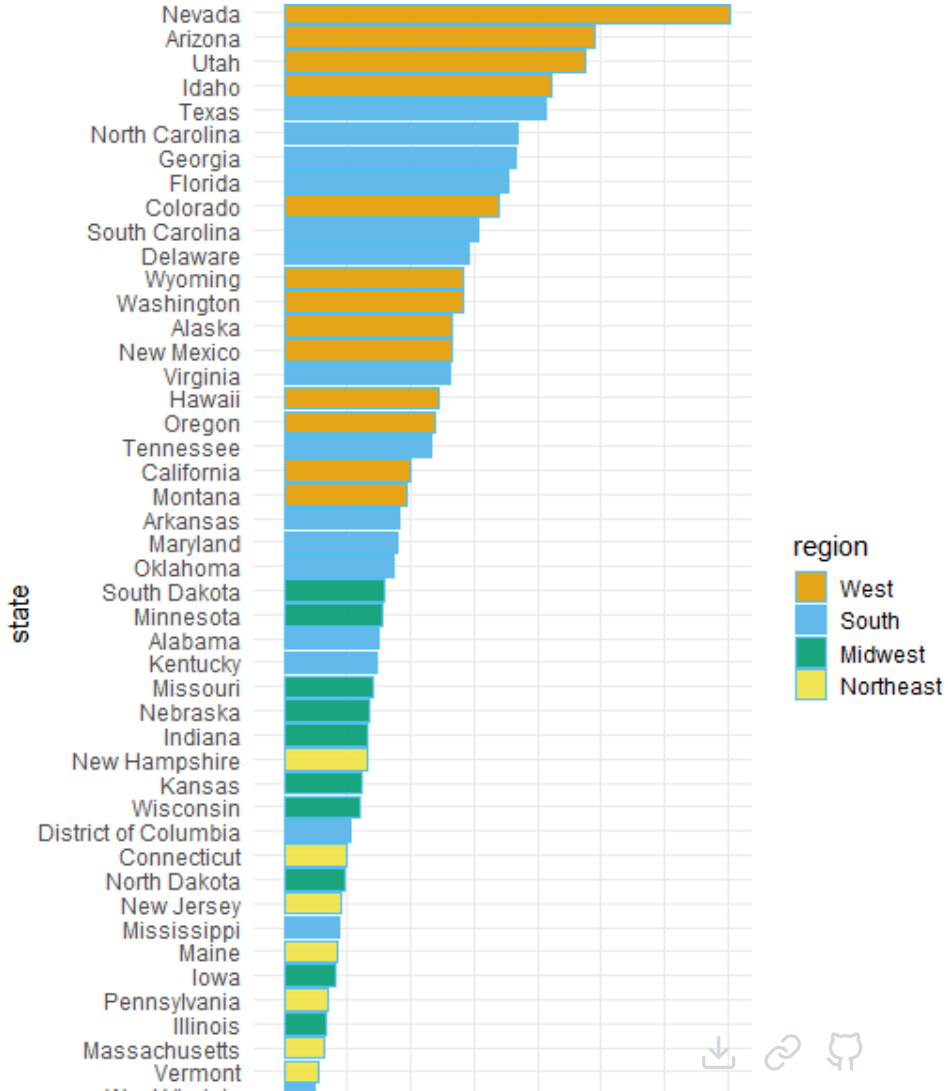
Revised version



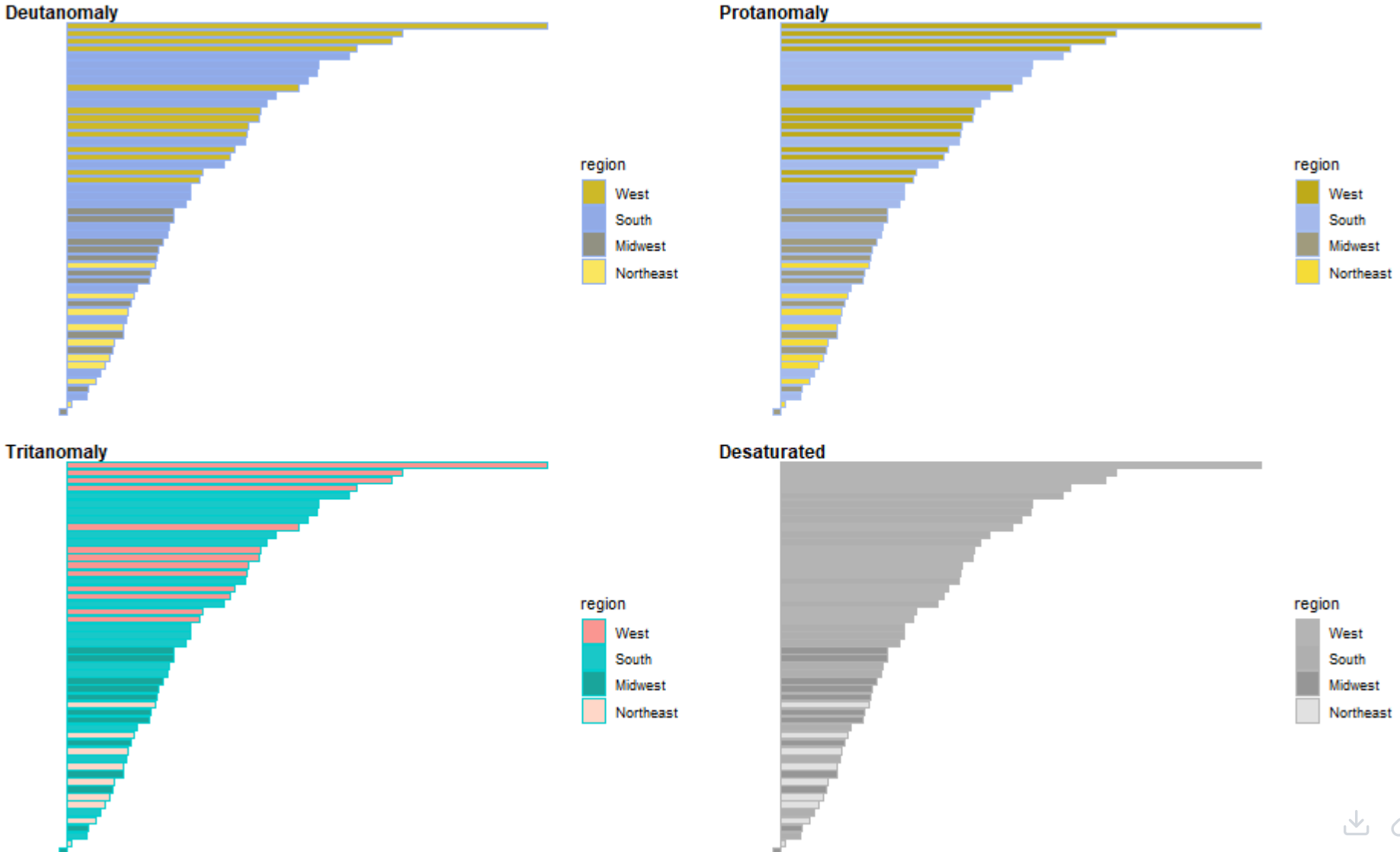
The Okabe Ito palette

- From Color Universal Design

```
library(colorblindr)

ggplot(popgrowth_df,
       aes(x = popgrowth,
           y = state)) +
  geom_col(aes(fill = region),
           alpha = 0.9) +
  scale_fill_OkabeIto()
```



Okabe Ito for colorblindness



How am I checking for colorblindness?

- Also part of the **{colorblindr}** package (here)
  - depends on the dev versions of **{colorspace}** and **{cowplot}**, which are useful packages in their own right

```
#devtools::install_github("wilkelab/cowplot")
#install.packages("colorspace", repos = "http://R-Forge.R-project.org")

#devtools::install_github("clauswilke/colorblindr")
```

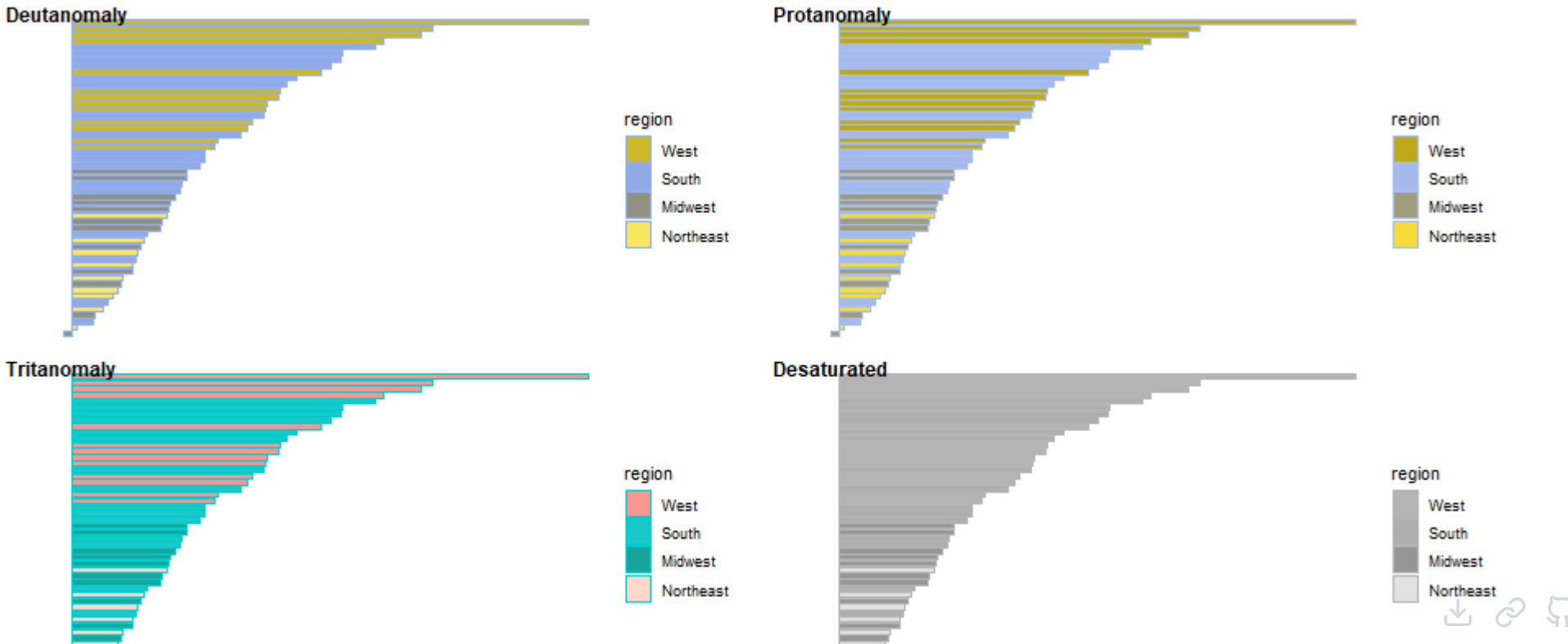
A note on installation

Occasionally people have run into issues with the install on the previous slide.

If that happens, try `install.packages("colorBlindness")` instead, and use the `cvdPlot` function.

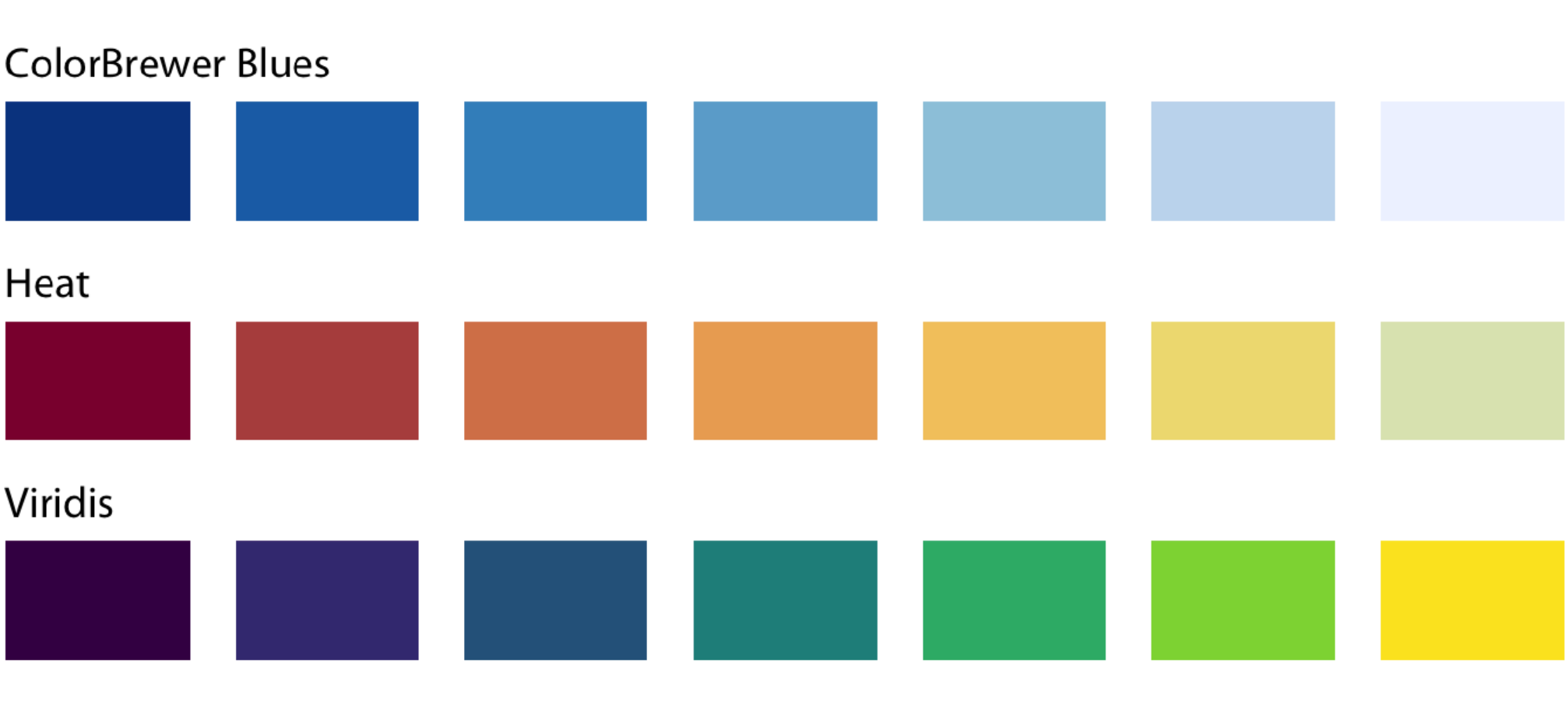
(It's just a modified version of `cvd_grid` from *colorblindr*)

```
p <- ggplot(popgrowth_df,
       aes(x = popgrowth,
           y = state)) +
  geom_col(aes(fill = region),
           alpha = 0.9) +
  scale_fill_OkabeIto() +
  theme_void() # not necessary but I like it

colorblindr::cvd_grid(p)
```



Colors for continuous values

Sequential scale examples



Sequential scales

- Which values are larger/smaller
- How distant two values are from each other
  - Scale must be perceptually uniform across its entire range
  - Similar to an interval scale, but for color
- Often based on a single hue
- Multi-hue sequential scales tend to follow gradients in the natural world

Common uses of sequential palettes

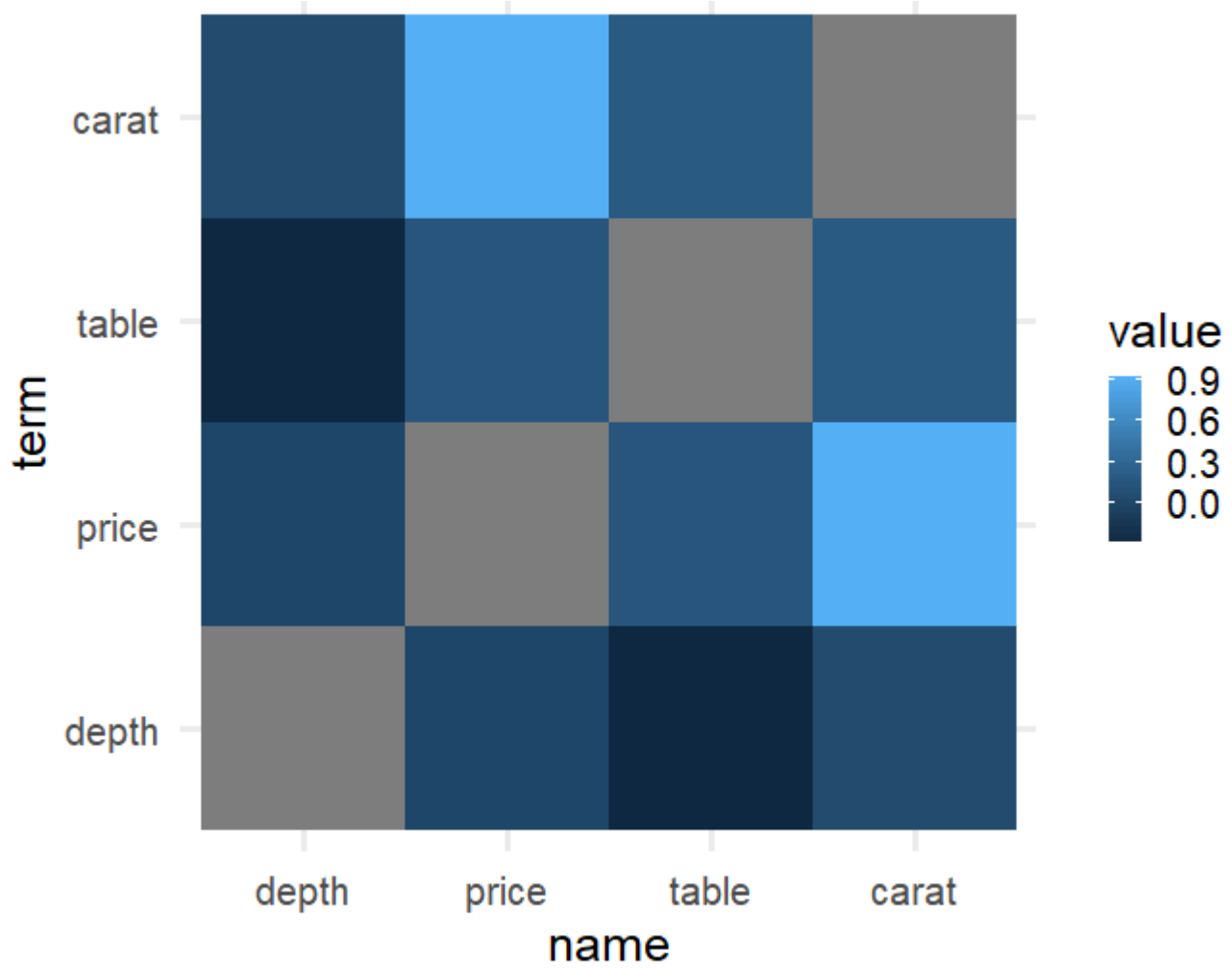
Heatmaps

First the data:

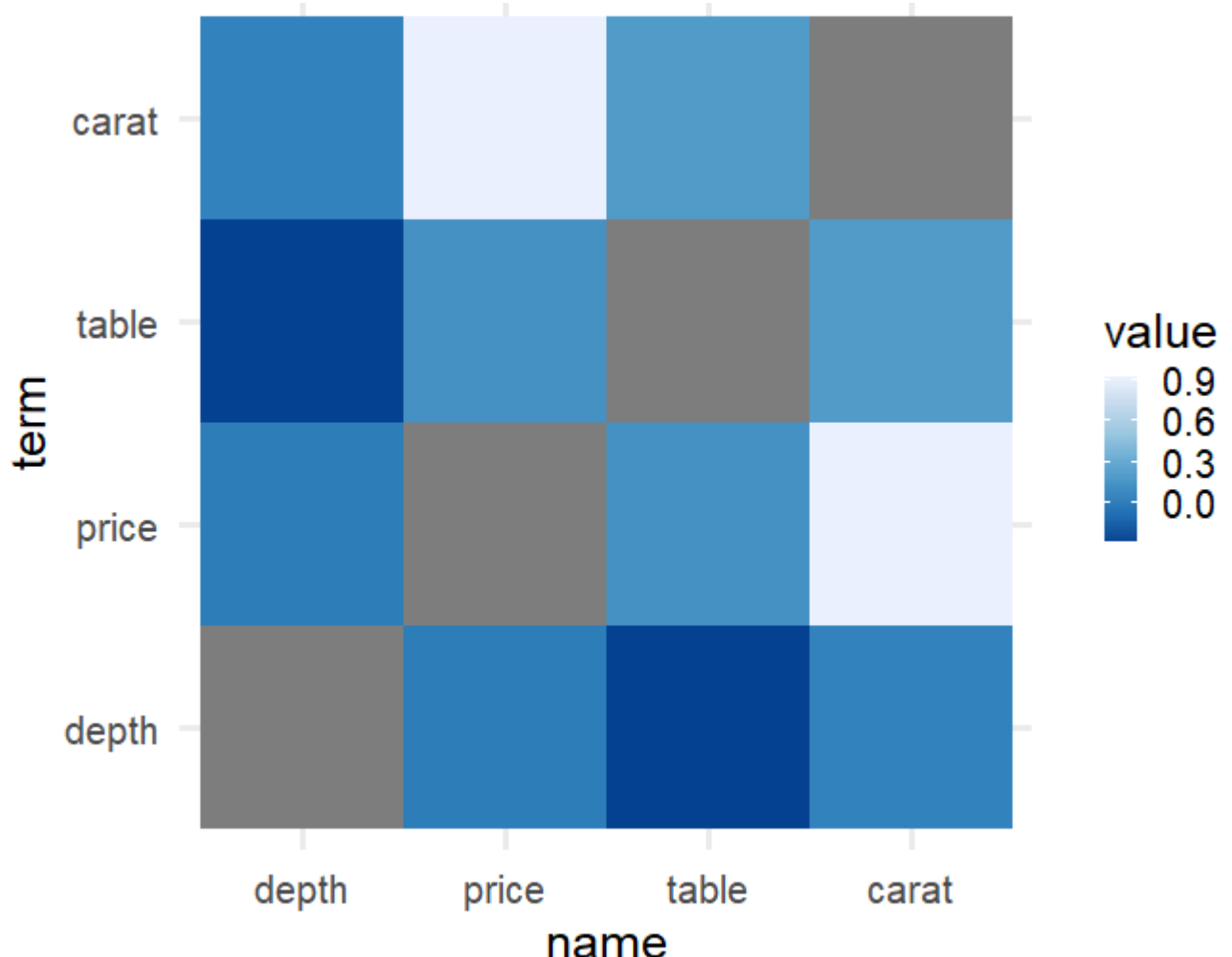
```
hm <- diamonds %>%
  select(table, price, depth, carat) %>%
  corrr::correlate() %>%
  pivot_longer(-term) %>%
  mutate(name = fct_reorder(name, value),
         term = fct_reorder(term, value))
hm
```

```
## # A tibble: 16 × 3
##   term  name   value
##   <fct> <fct>  <dbl>
## 1 table table NA
## 2 table price  0.127
## 3 table depth -0.296
## 4 table carat  0.182
## 5 price table  0.127
## 6 price price NA
## # ℹ 10 more rows
```

```
ggplot(hm, aes(name, term)) +
  geom_tile(aes(fill = value)) +
  coord_fixed()
```



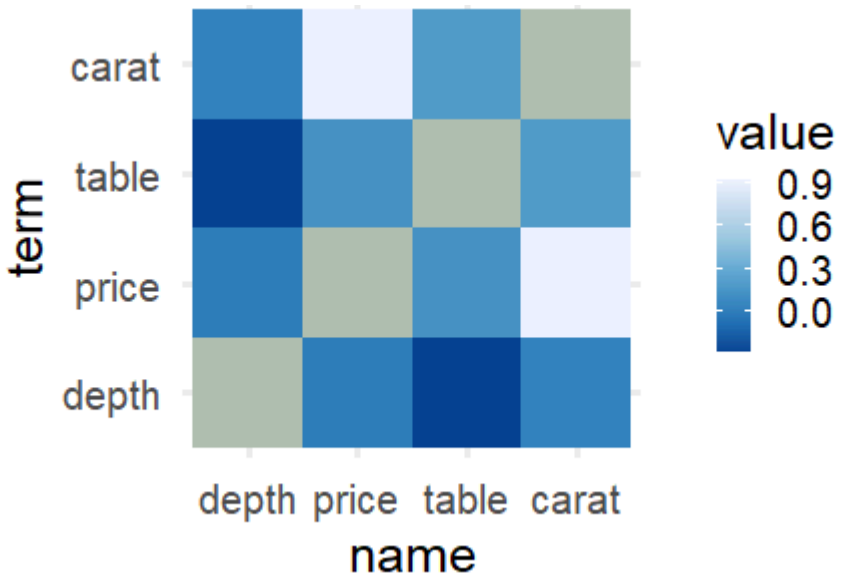
```
ggplot(hm, aes(name, term)) +
  geom_tile(aes(fill = value)) +
  coord_fixed() +
scale_fill_distiller(palette = "Blues")
```



Change the NA value

In any `scale_*` you can change the `NA` value, including to `"transparent"`.

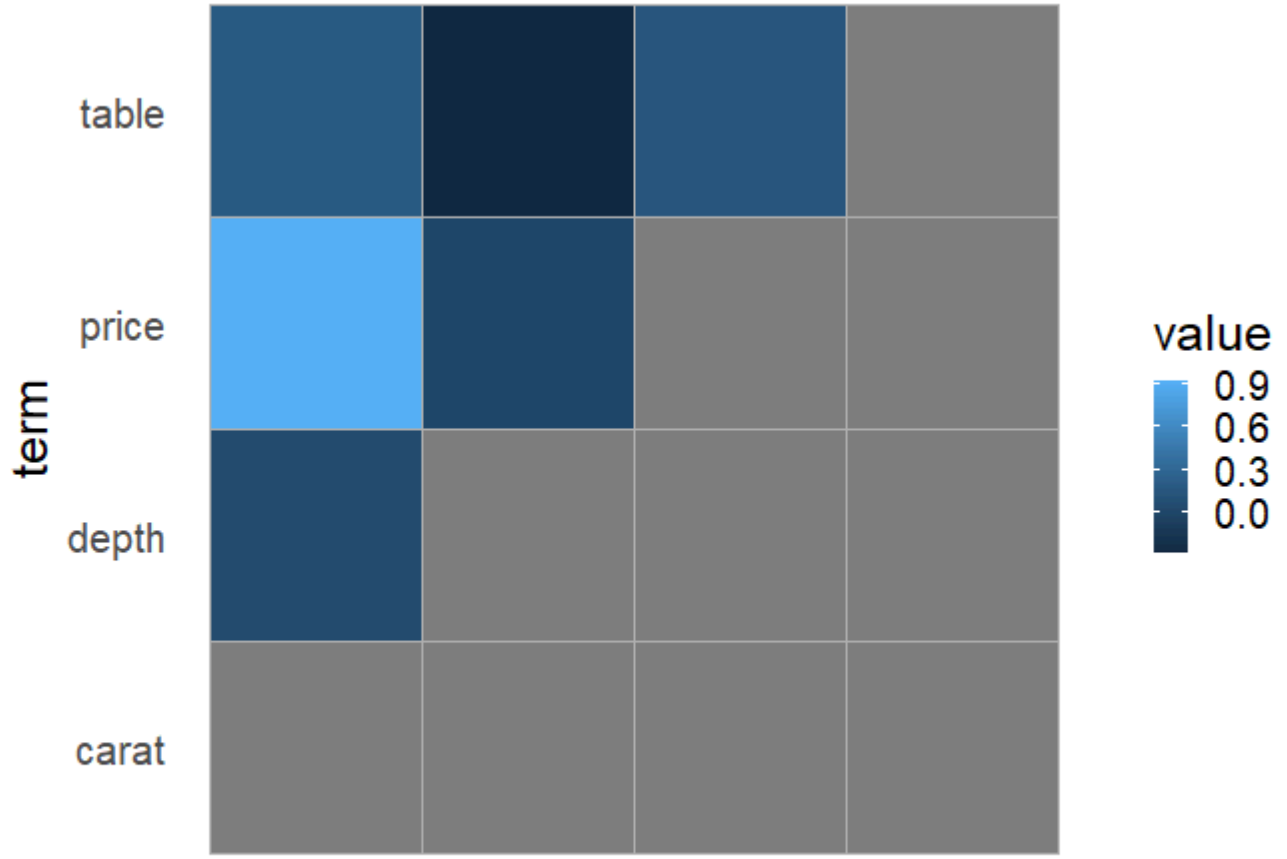
```
ggplot(hm, aes(name, term)) +
  geom_tile(aes(fill = value)) +
  coord_fixed() +
    scale_fill_distiller(palette = "Blues",
                         na.value = "#b0bfb0")
```



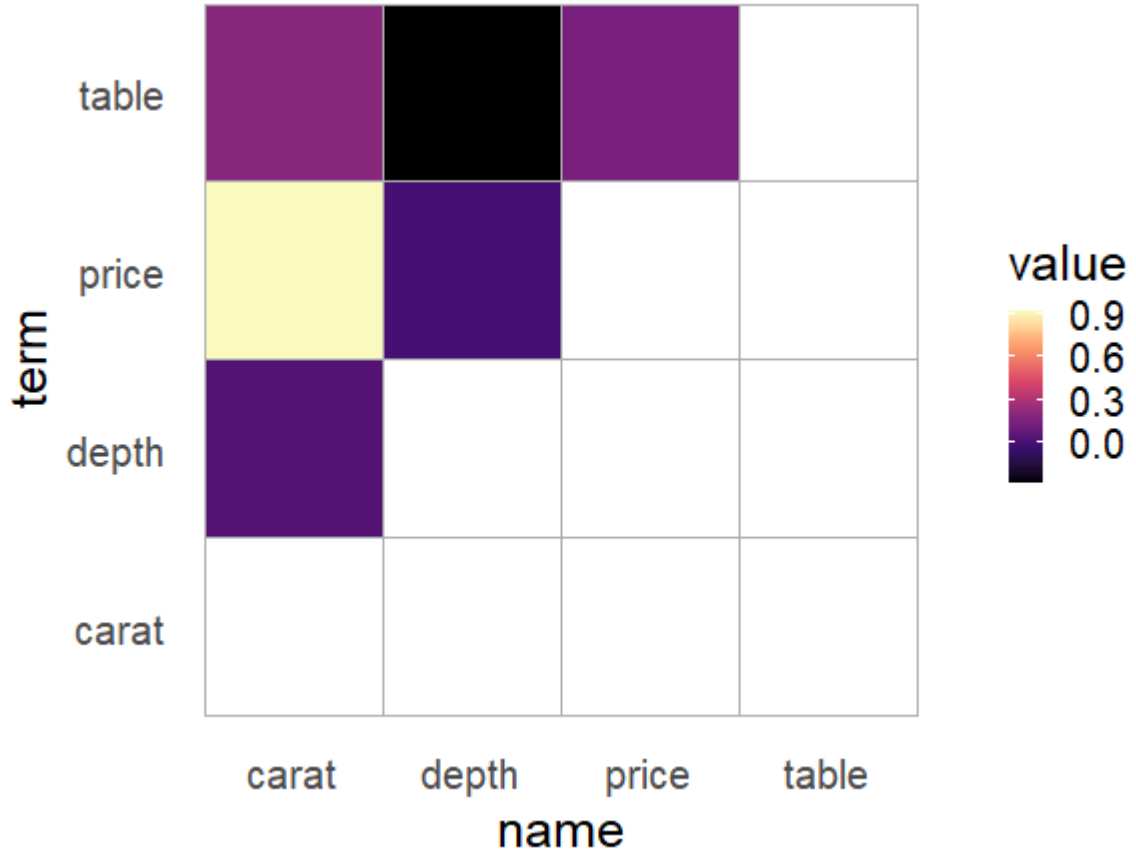
Use this to our advantage

```
hm2 <- diamonds %>%
  select(table, price, depth, carat) %>%
  corrr::correlate() %>%
  corrr::shave(upper = FALSE) %>%
  pivot_longer(-term)
```

```
hm2_default <- ggplot(hm2, aes(name, term)) +
  geom_tile(aes(fill = value), color = "gray70") +
  coord_fixed() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
hm2_default
```



```
hm2_default +
    scale_fill_viridis_c(
   option = "magma",
   na.value = "transparent"
  )
```



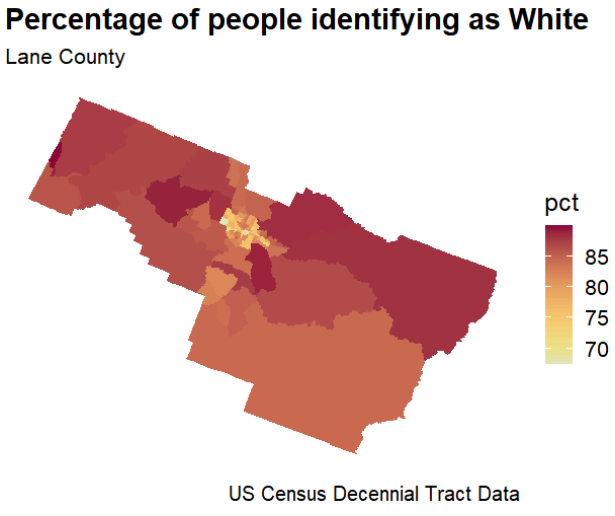
Choropleths

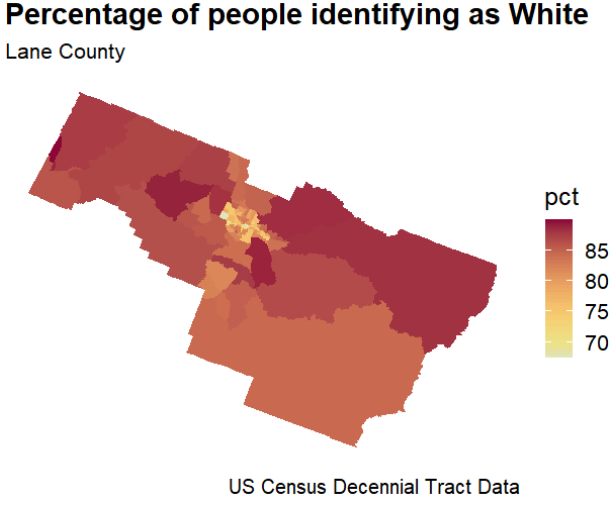
```
library(tidycensus)
library(tigris)
options(tigris_use_cache = TRUE)
options(tigris_class="sf")

#census_api_key("57878ac2a009c05cc3f232f348b88d1434839be2", install=TRUE)

vars_pl_2020 <-  load_variables(2020, "pl")
#View(vars)
```

```
lane <- get_decennial(geography = "tract", variables = "P1_003N",
                  state = "OR", county = "Lane", geometry = TRUE,
                  summary_var = "P1_001N")

or <- get_decennial(geography = "county", variables = "P1_003N",
                  state = "OR", geometry = TRUE,
                  summary_var = "P1_001N")

ca <- get_decennial(geography = "county", variables = "P1_003N",
                  state = "CA", geometry = TRUE,
                  summary_var = "P1_001N")

lane %>%
  mutate(pct = 100 * (value / summary_value)) %>%
  ggplot(aes(fill = pct, color = pct)) +
```

Heat palette

```
lane %>%
  mutate(pct = 100 * (value / summary_value)) %
  ggplot(aes(fill = pct, color = pct)) +
  geom_sf() +
  coord_sf(crs = 26915) +
  theme_dviz_map(font_size = 25, font_family =
  scale_fill_continuous_sequential("Heat") +
  scale_color_continuous_sequential("Heat") +
  labs(title = "Percentage of people identifyin
       subtitle = "Lane County",
       caption = "US Census Decennial Tract Dat
```

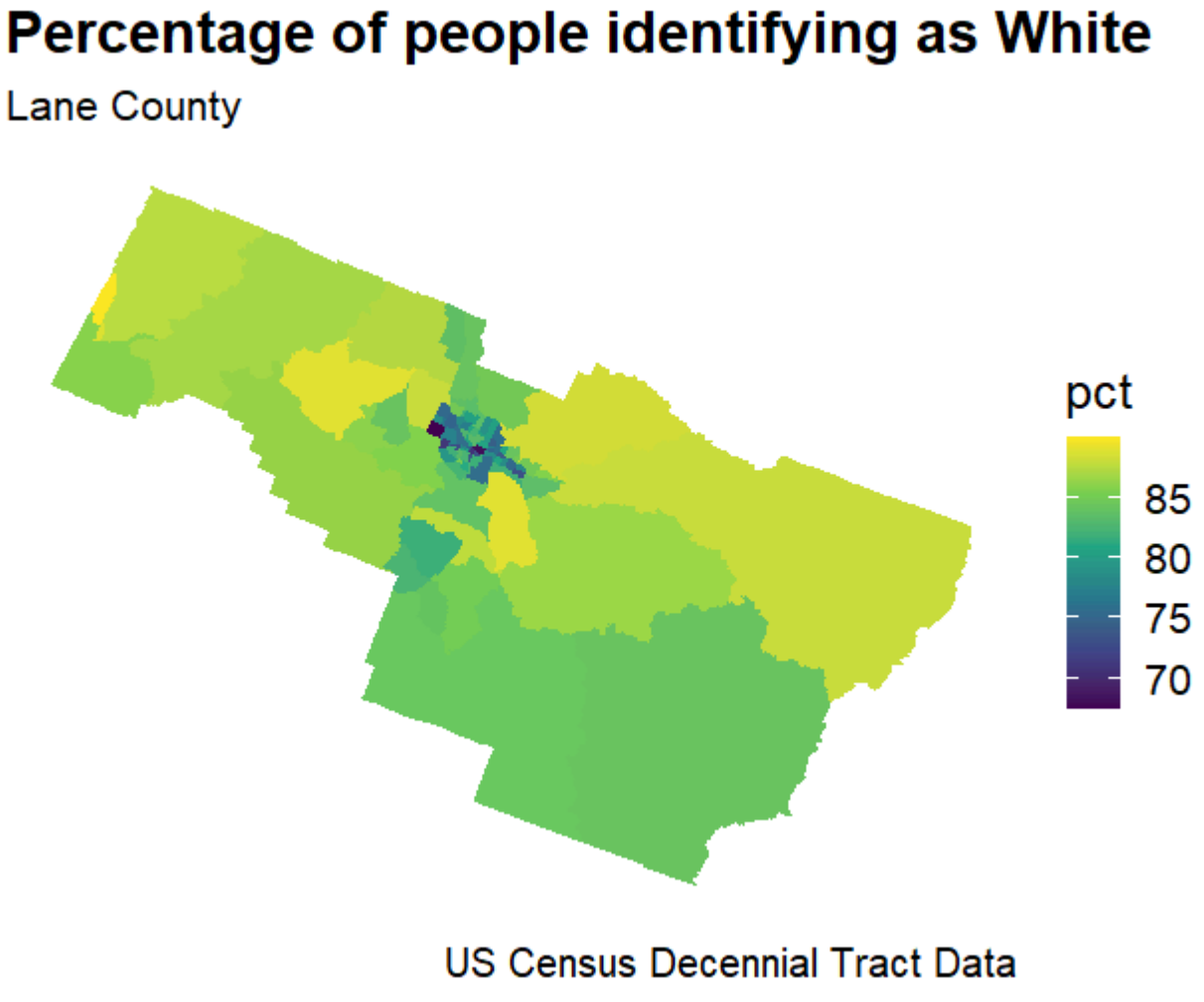




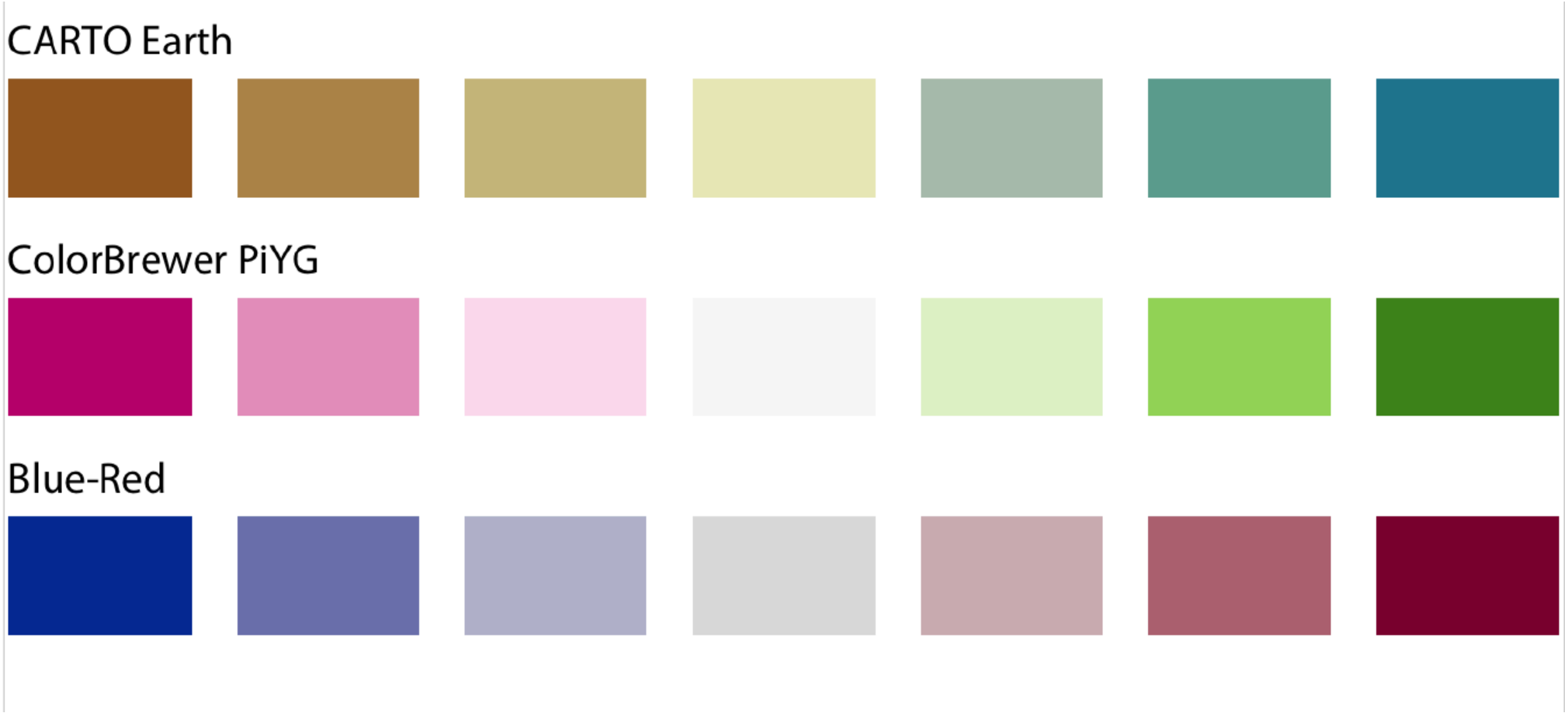
Options

- `scale_fill_continuous_sequential("Heat")`
- `scale_color_continuous_sequential("Heat")`
- `scale_fill_discrete_sequential("Heat")`
- `scale_color_discrete_sequential("Heat")`

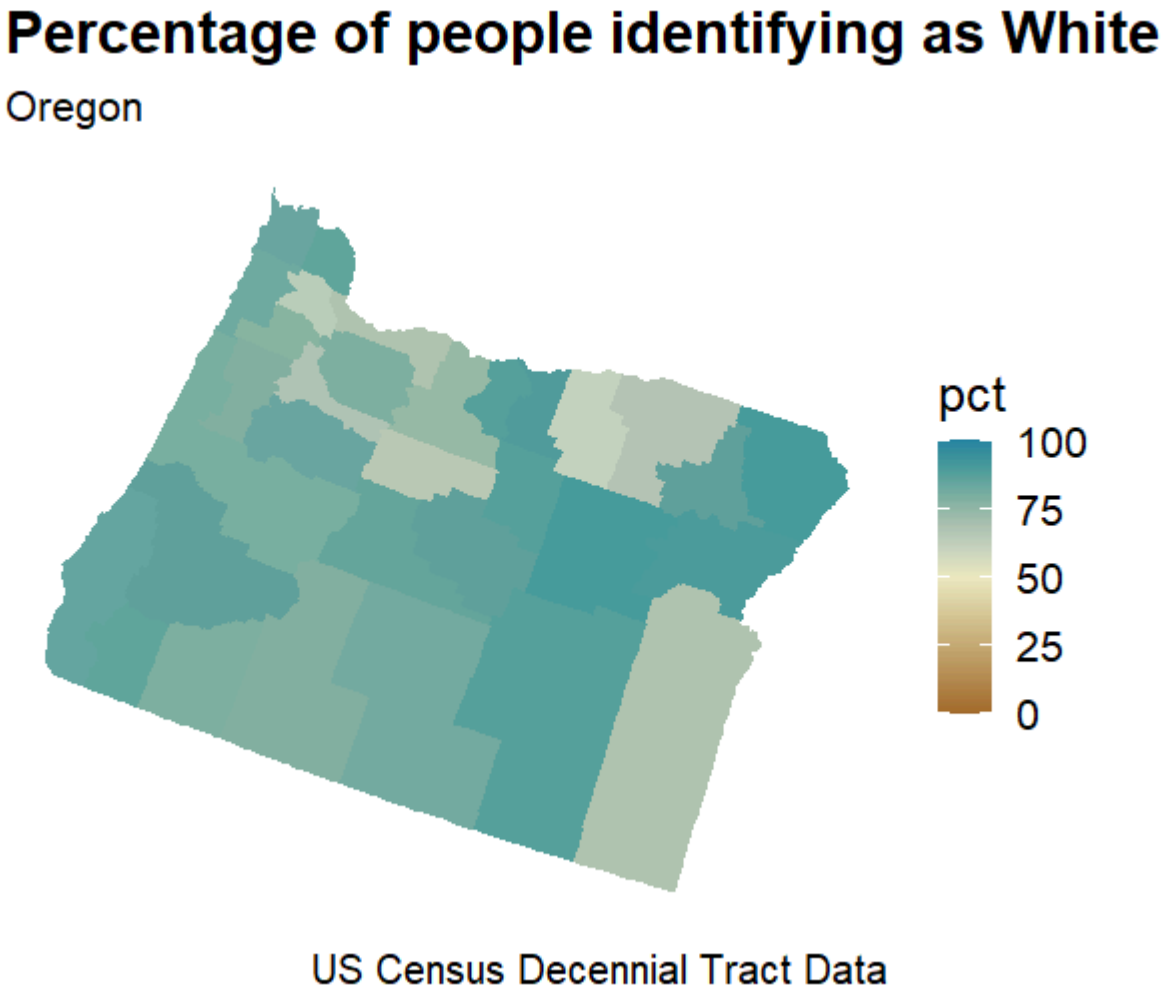
viridis palette

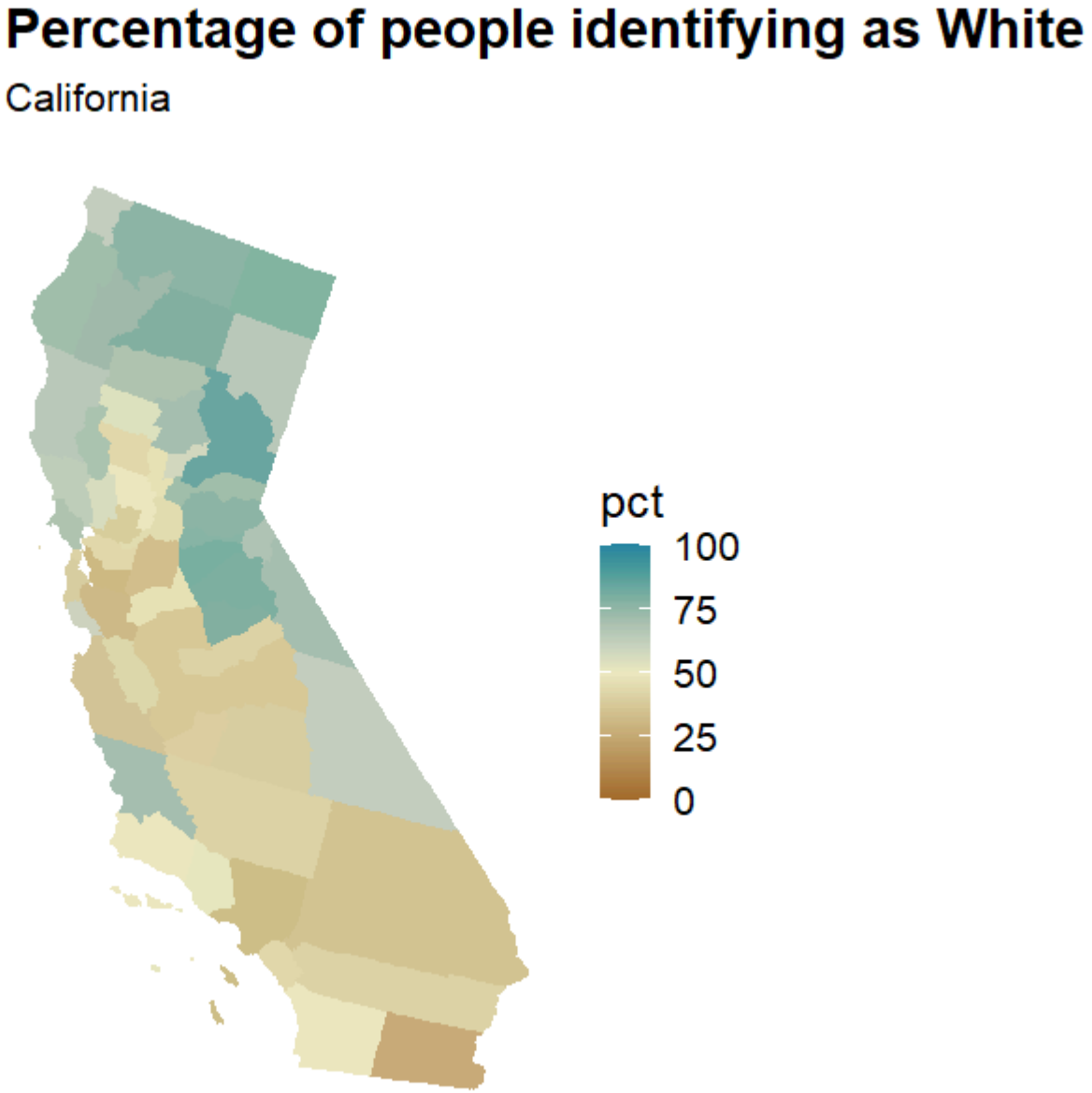


Diverging palettes



Earth palette




Percentage of people identifying as White
California
pct
100
75
50
25
0
US Census Decennial County Data

Back to our heatmap

Use a diverging palette that balances at zero.

Notice the transparency is now a bit problematic...

```
library(colorspace)
hm2_default +
  scale_fill_continuous_divergingx(
    palette = "Tropic",
    mid = 0,
    limits = c(-1, 1),
    rev = TRUE,
    na.value = "transparent"
  )
```

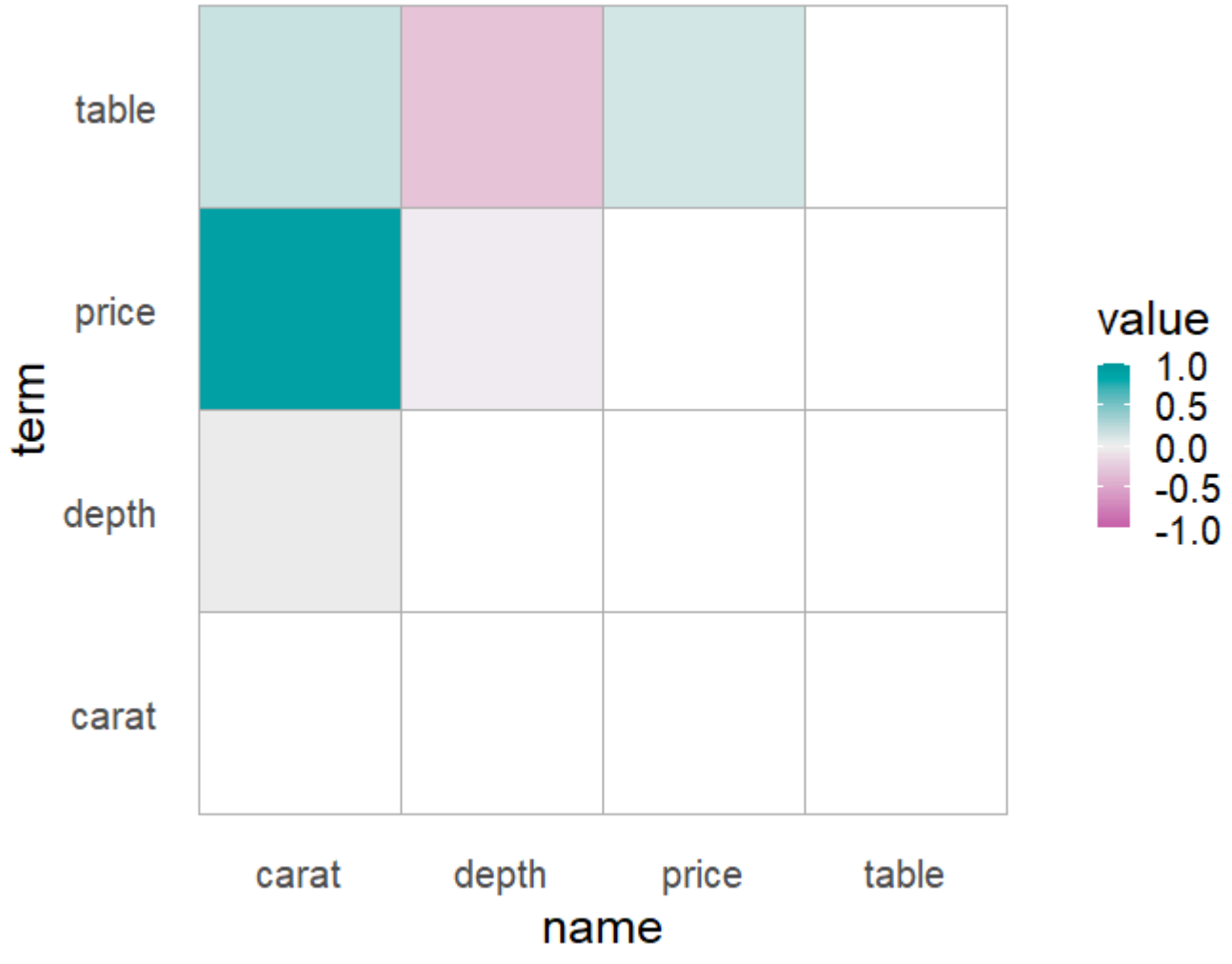


table
price
depth
carat
term
carat
depth
price
table
name
value
1.0
0.5
0.0
-0.5
-1.0

Try a different palette

Most diverging palettes have a light gray or white center point. Some have black. Let's try one of those.

```
hm2_default +
  scale_fill_continuous_diverging(
    palette = "Tofino",
    mid = 0,
    limits = c(-1, 1),
    rev = TRUE,
    na.value = "transparent"
  )
```

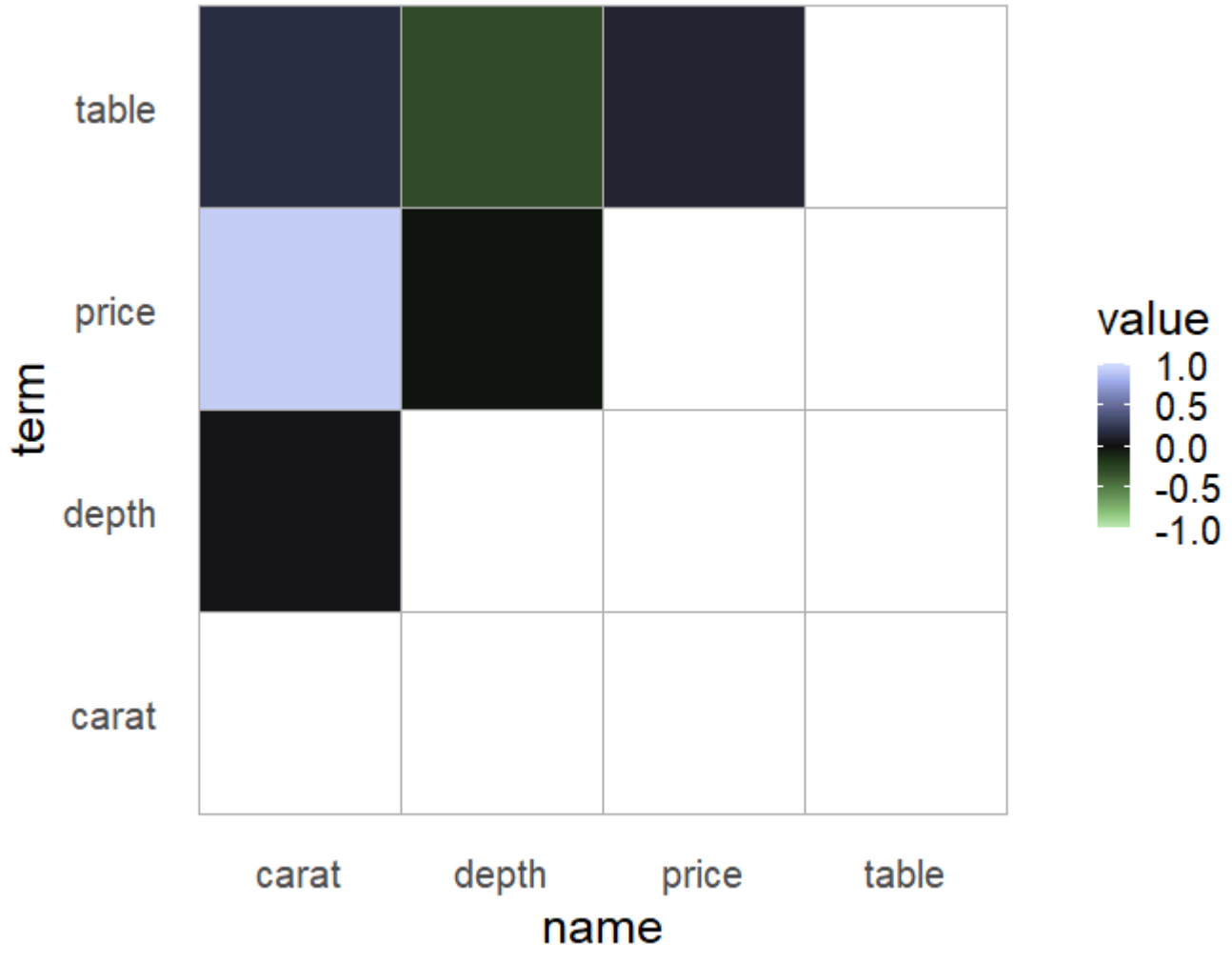


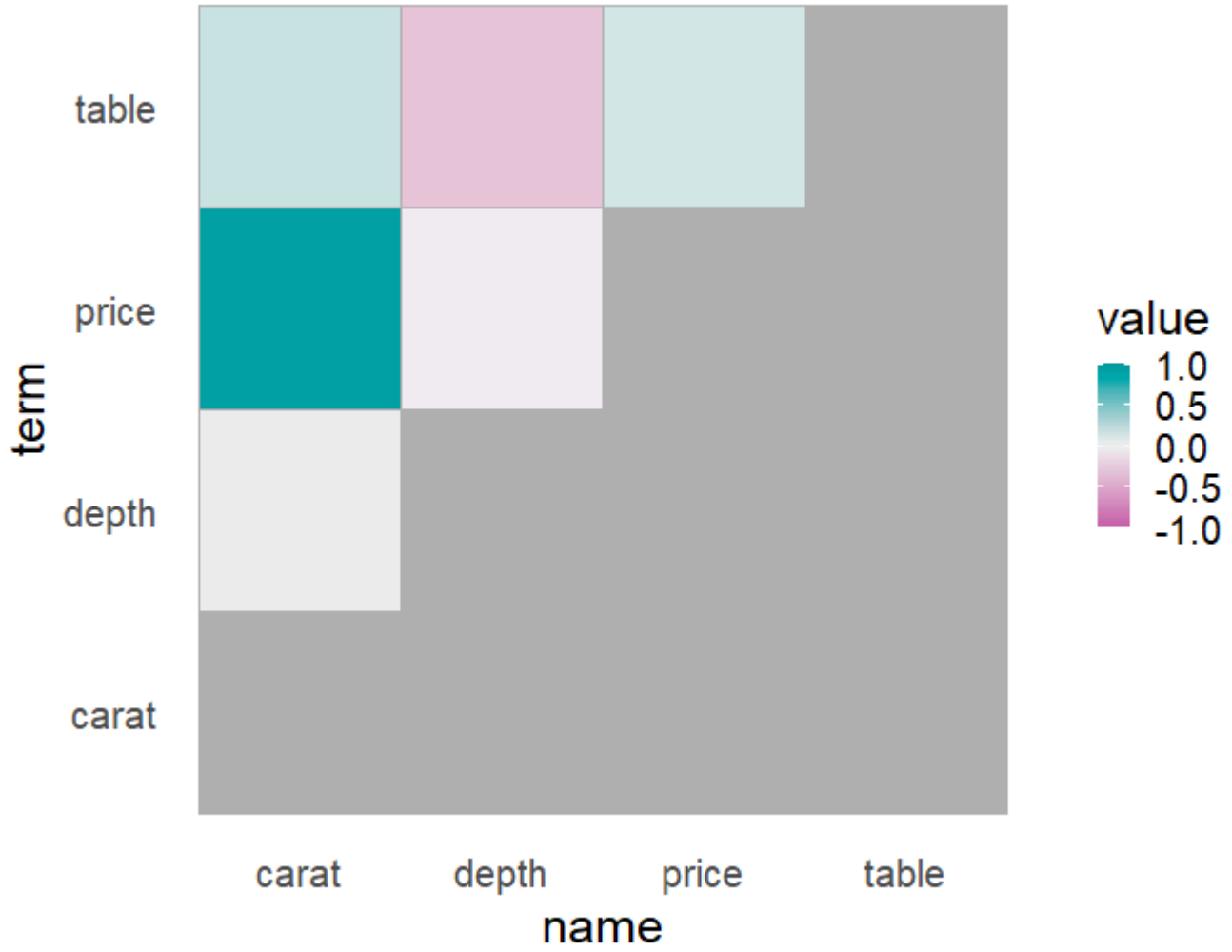
table
price
depth
carat
term
carat
depth
price
table
name
value
1.0
0.5
0.0
-0.5
-1.0

don't love it

One more try!

Back to our last palette, but change the NA value to a light gray

```
hm2_default +
  scale_fill_continuous_divergingx(
    palette = "Tropic",
    mid = 0,
    limits = c(-1, 1),
    rev = TRUE,
    na.value = "gray70"
  )
```



Maybe better? I give up for now...

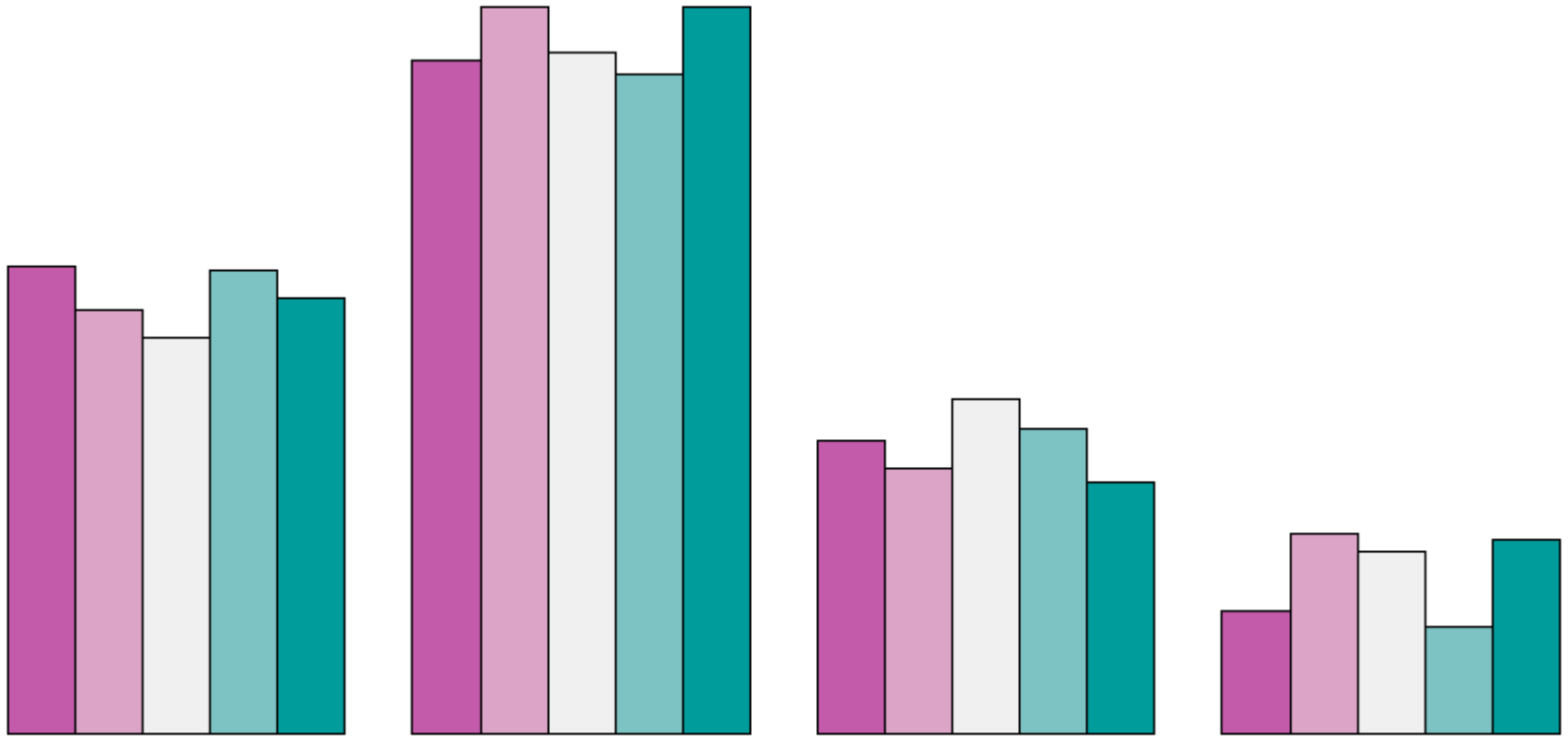
More on colorspace

Great package with great documentation. Let's go look!

colorspace

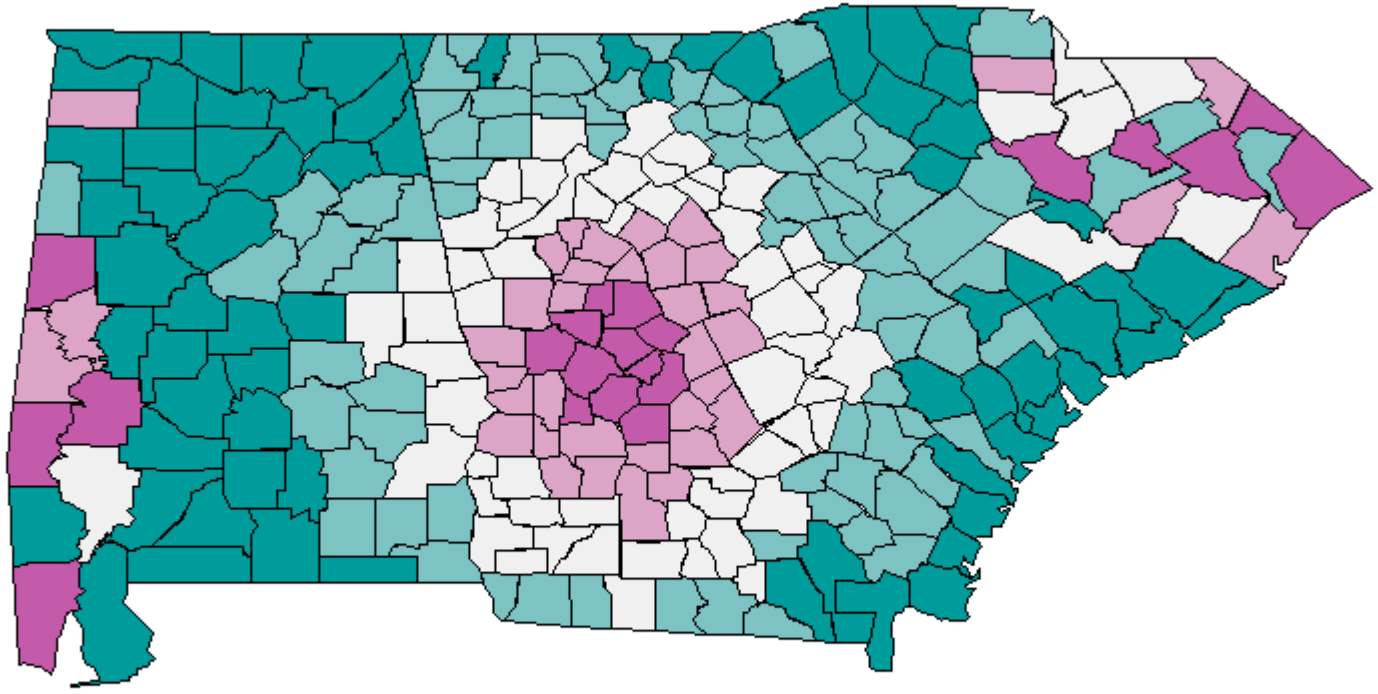
More diagnostics

```
our_scale <- diverge_hcl(5, palette = "Tropic", rev = TRUE)
demoplot(our_scale, "bar")
```

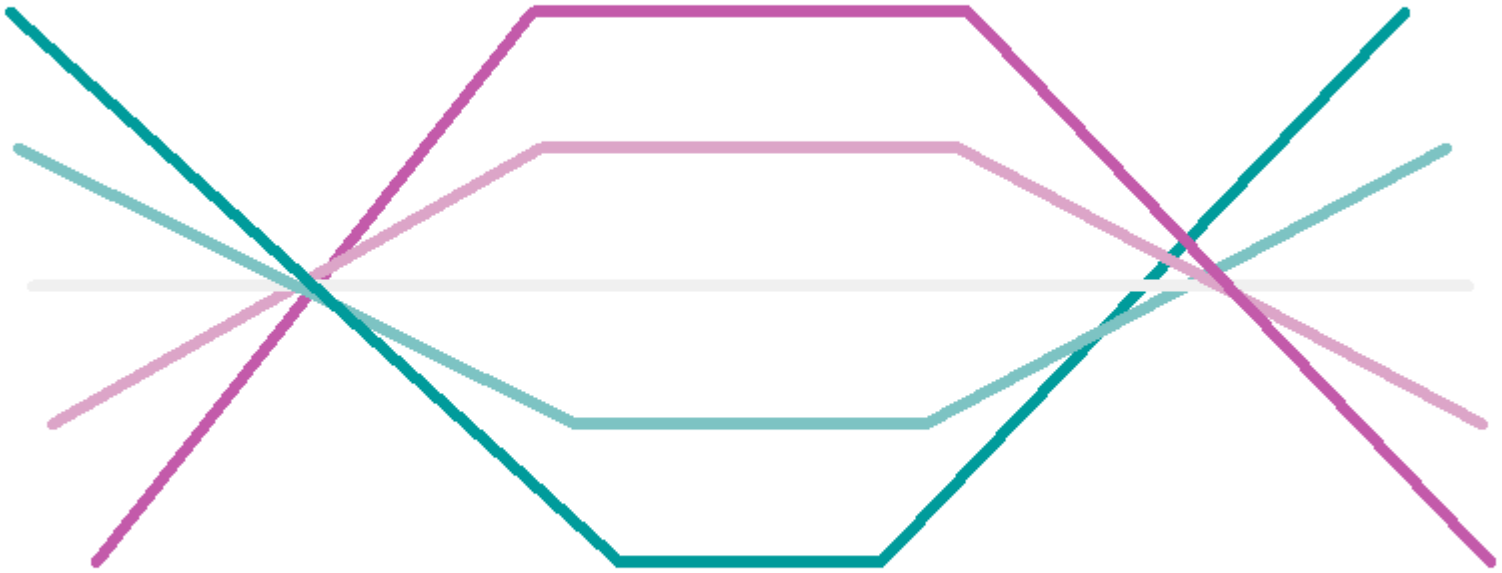


```
demoplot(our_scale, "heatmap")
```

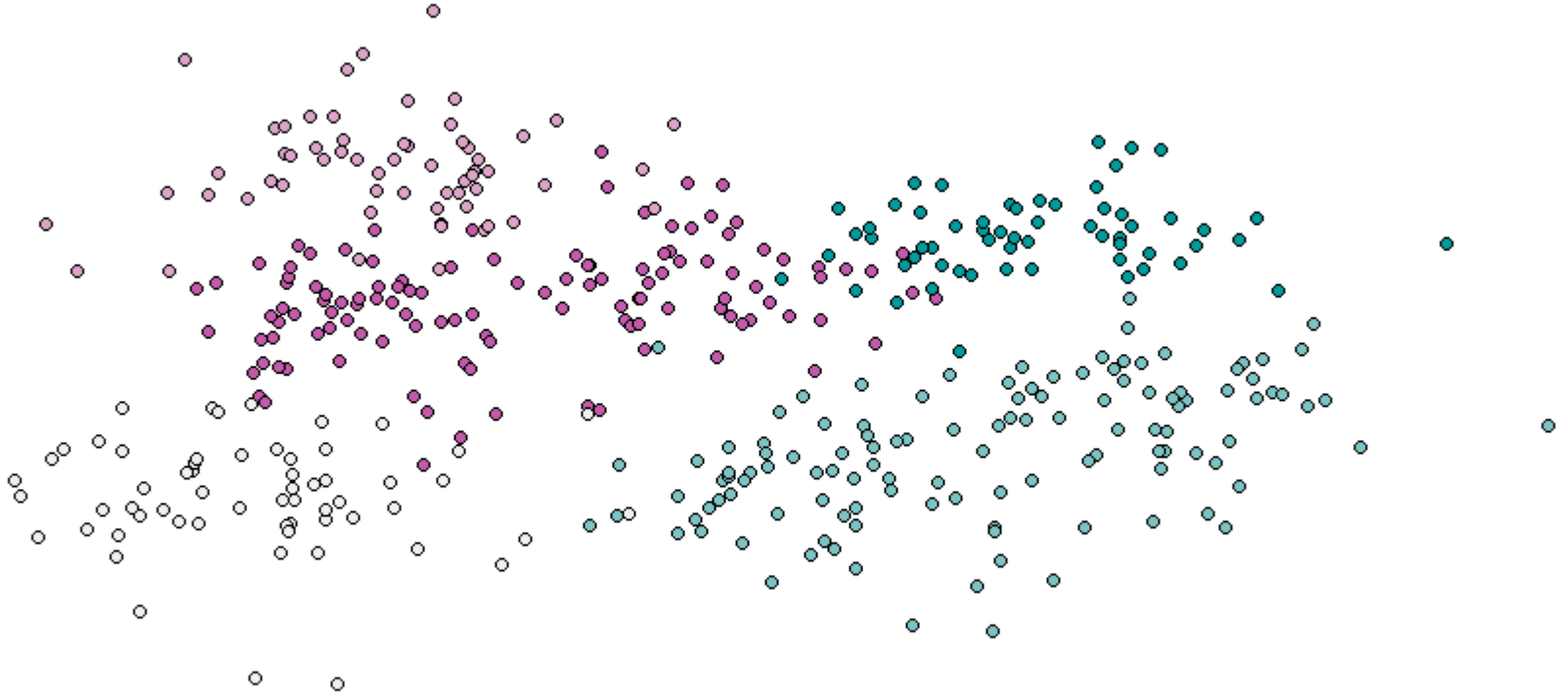
```
demoplot(our_scale, "map")
```



```
demoplot(our_scale, "lines")
```



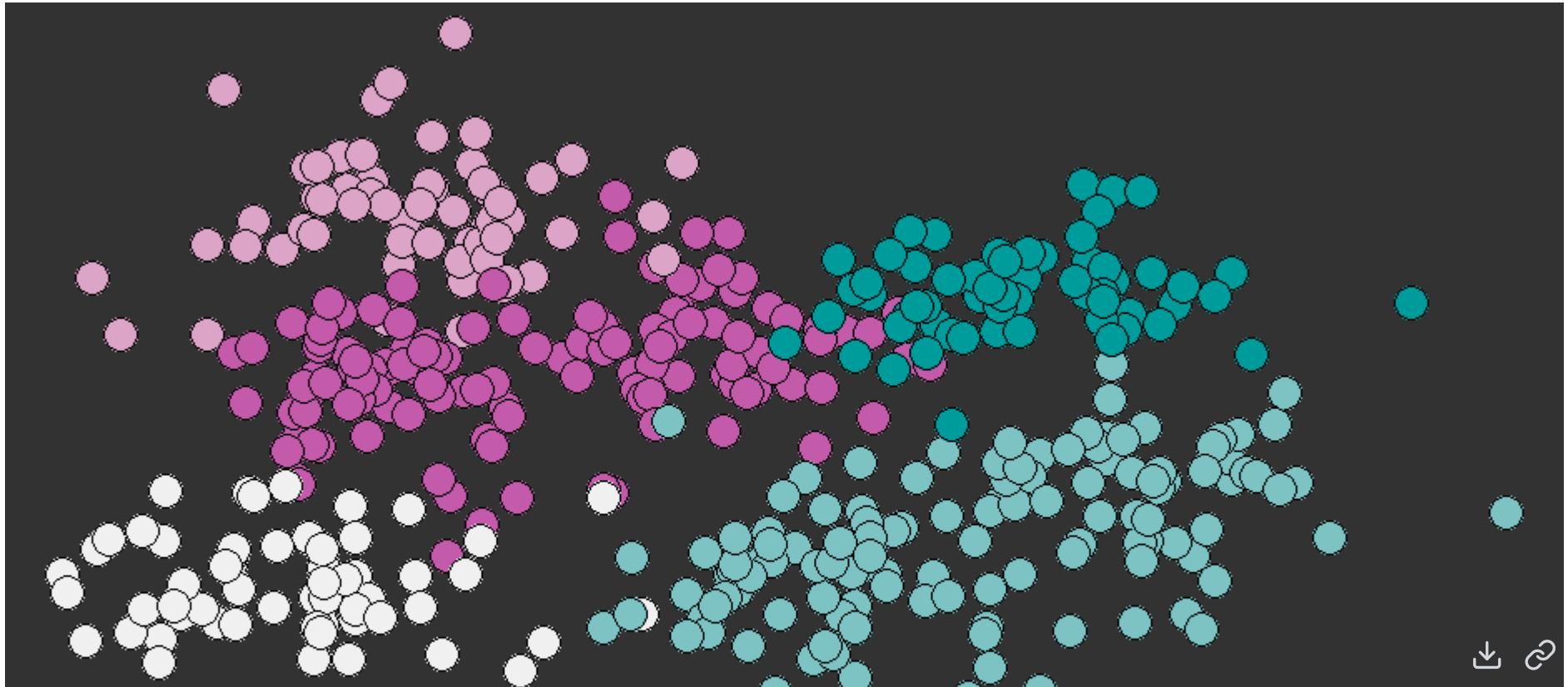
```
demoplot(our_scale, "scatter")
```



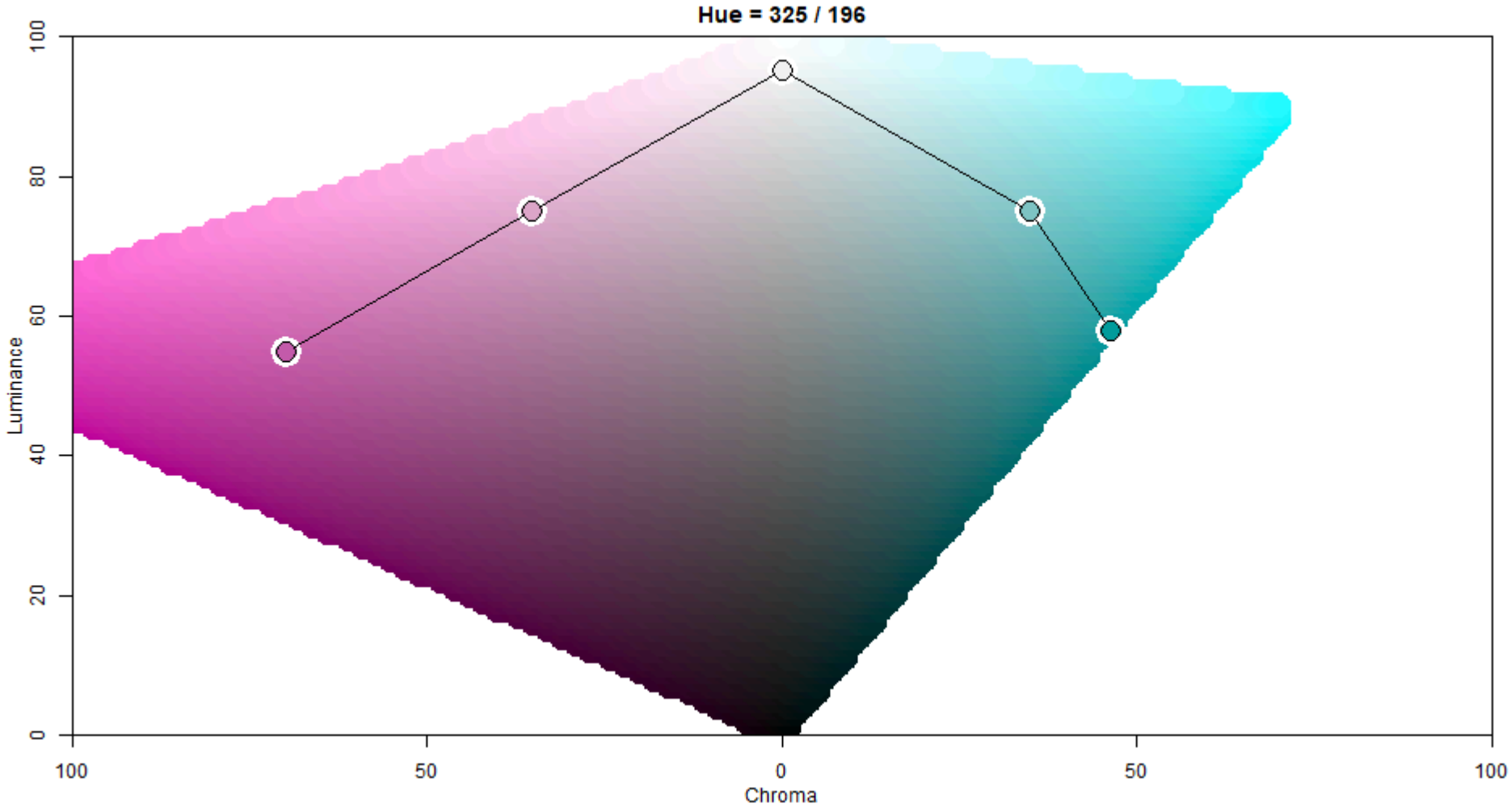
Diferent background

Note - this is base graphics

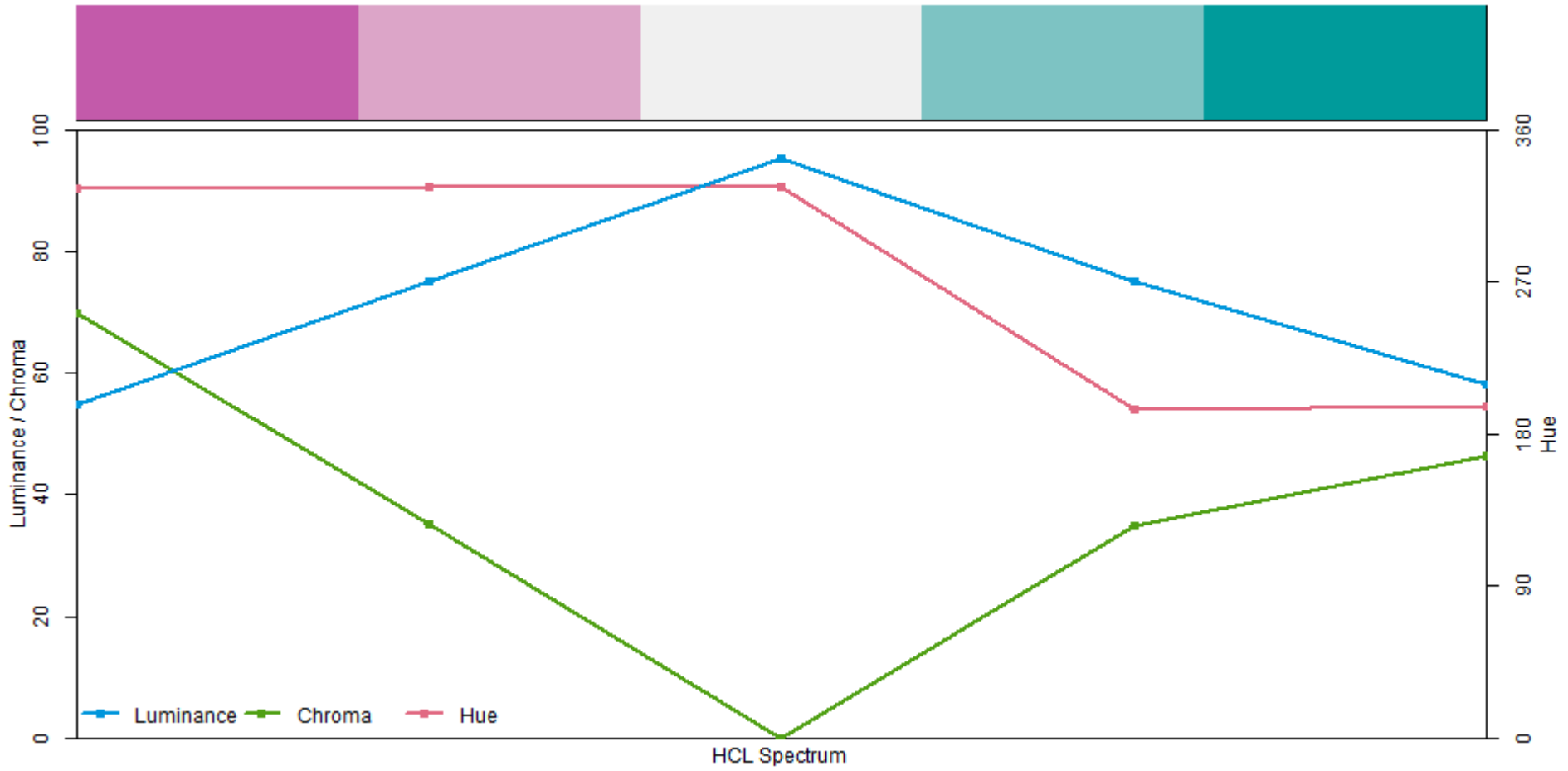
```
par(bg = "gray20", cex = 3, mar = rep(0, 4))
demoplot(our_scale, "scatter")
```



```
hclplot(our_scale)
```



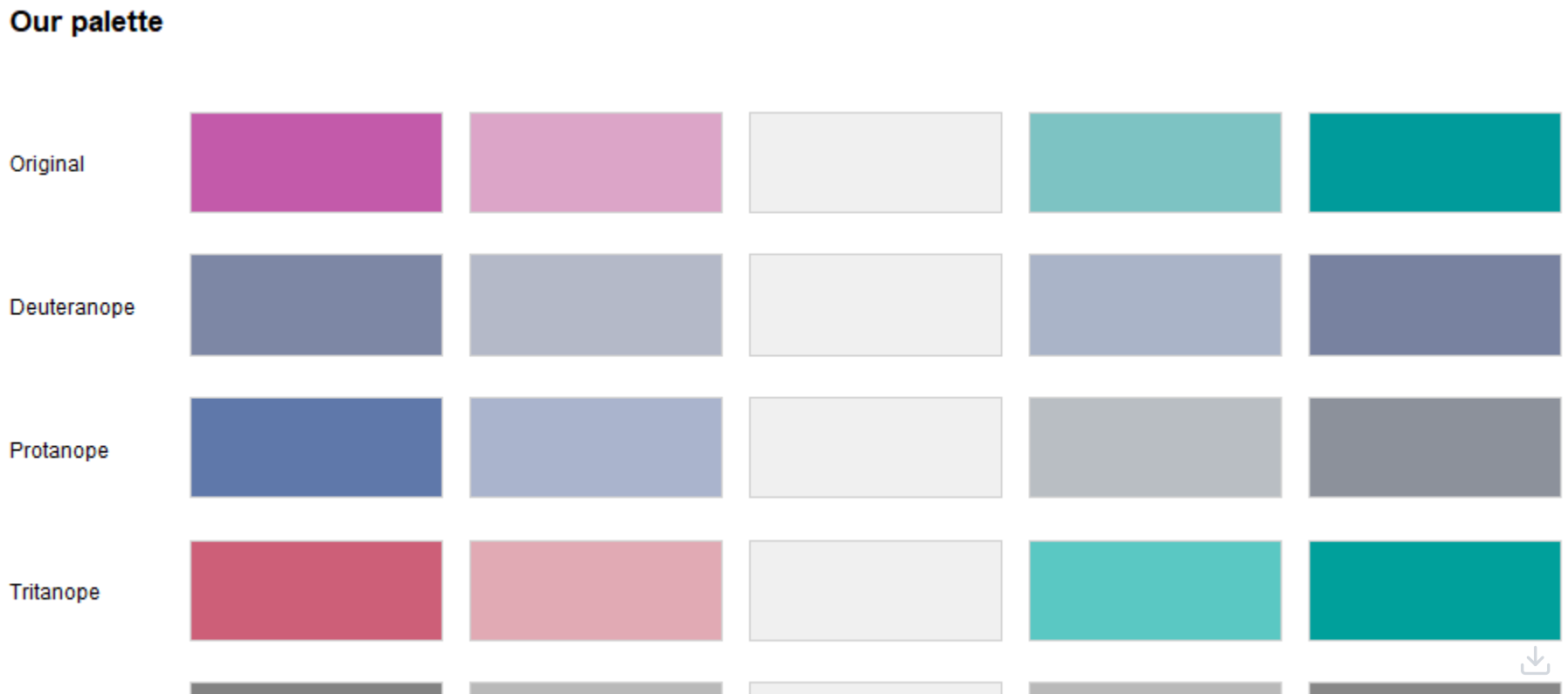
```
specplot(our_scale, type = "o")
```



Check colorblindness

We can check for our palette, in addition to our figure overall

```
swatchplot("Our palette" = our_scale, cvd = TRUE)
```



Interactivity

You can do this all online interactively with the demo plots

http://hclwizard.org:3000/hclwizard/

CVD Emulator online

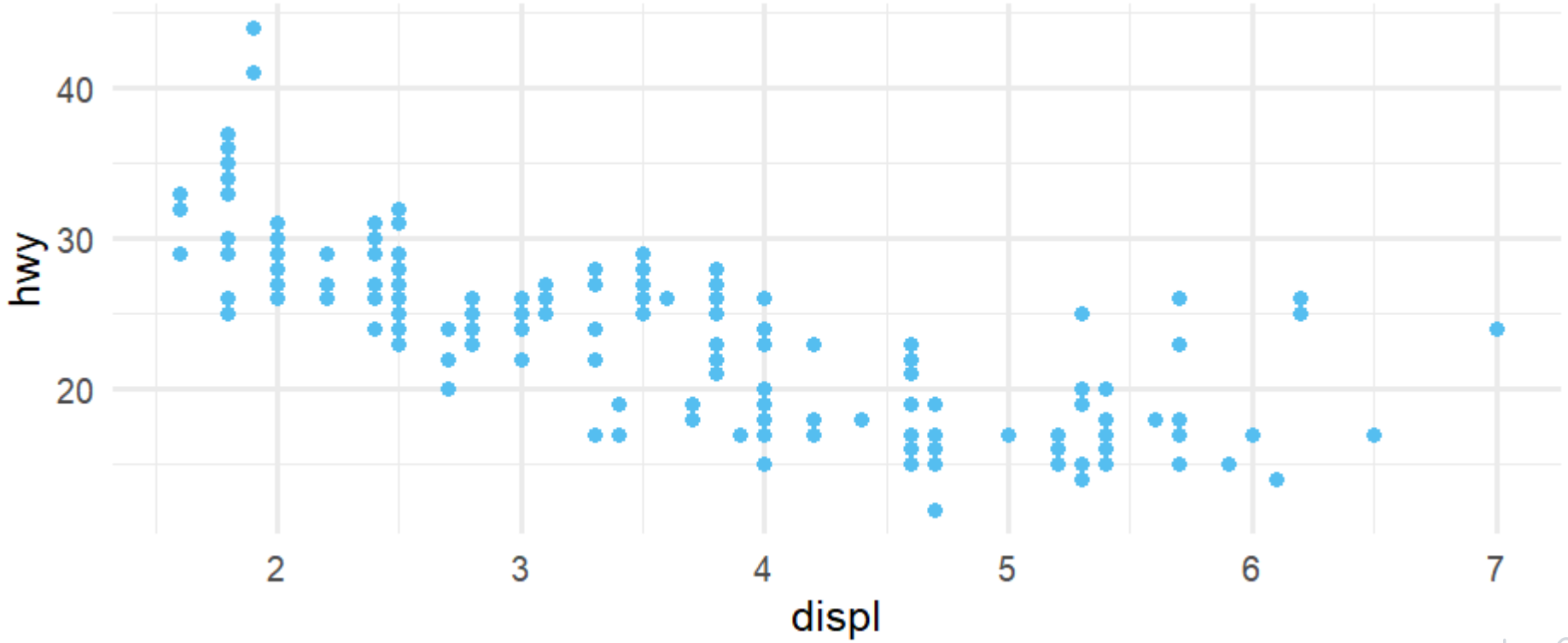
http://hclwizard.org:3000/cvdemulator/

Color as a tool to highlight

MPG data

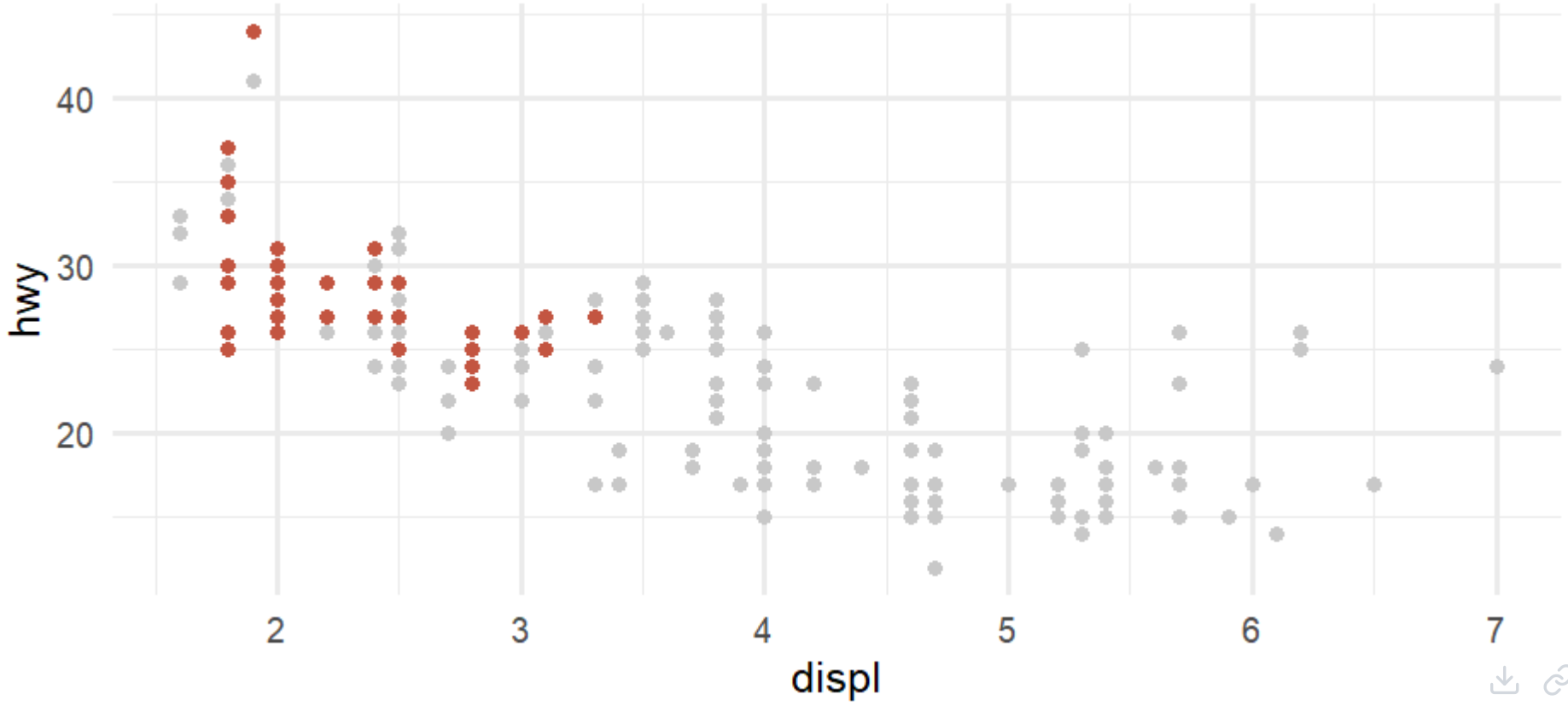
Basic scatterplot of weight to highway mpg

```
ggplot(mpg, aes(displ, hwy)) +
    geom_point()
```



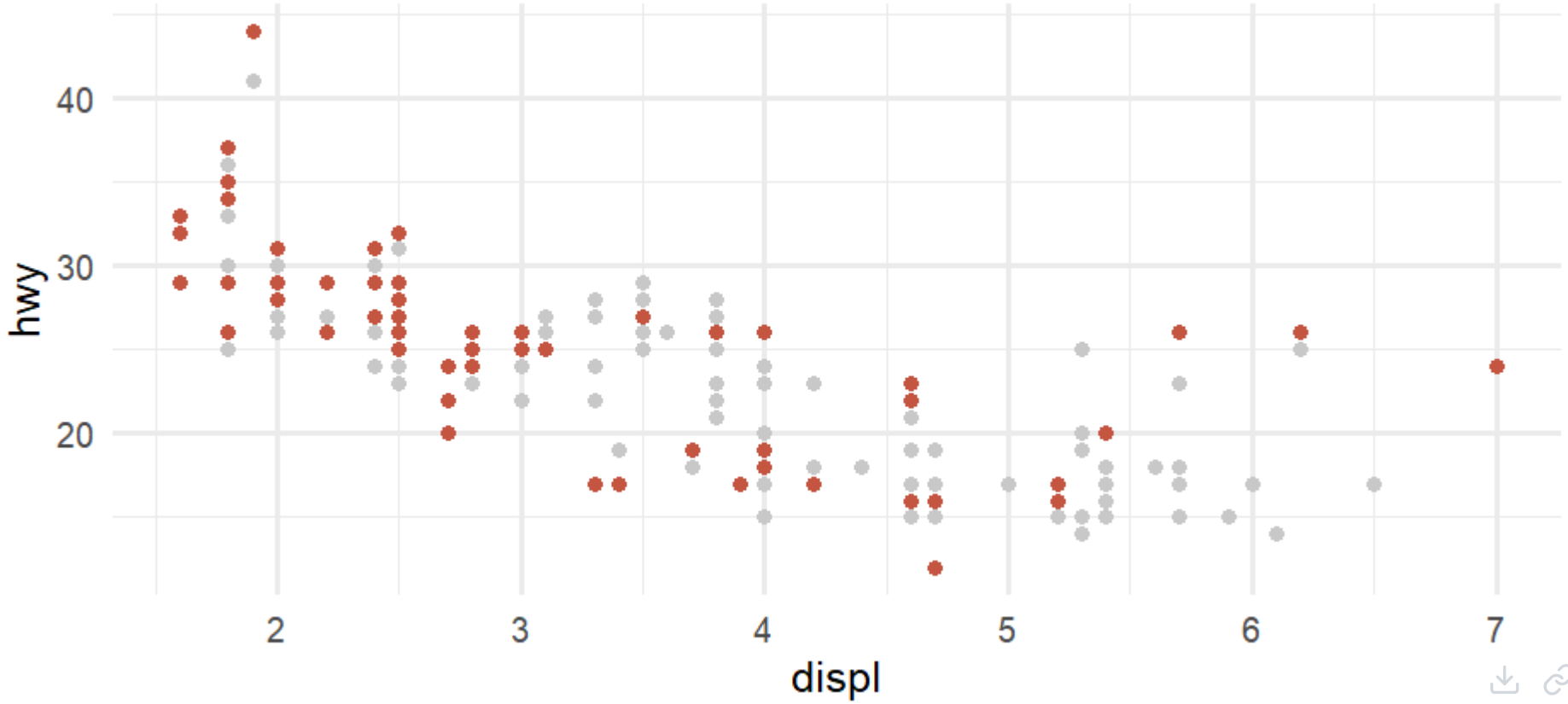
Highlight compact cars

```
ggplot(mpg, aes(displ, hwy)) +
    geom_point(color = "gray80") +
    geom_point(data = filter(mpg, class == "compact"),
               color = "#C55644")
```



Highlight manual cars

```
ggplot(mpg, aes(displ, hwy)) +
    geom_point(color = "gray80") +
    geom_point(data = filter(mpg, str_detect(trans, "manual")),
               color = "#C55644")
```

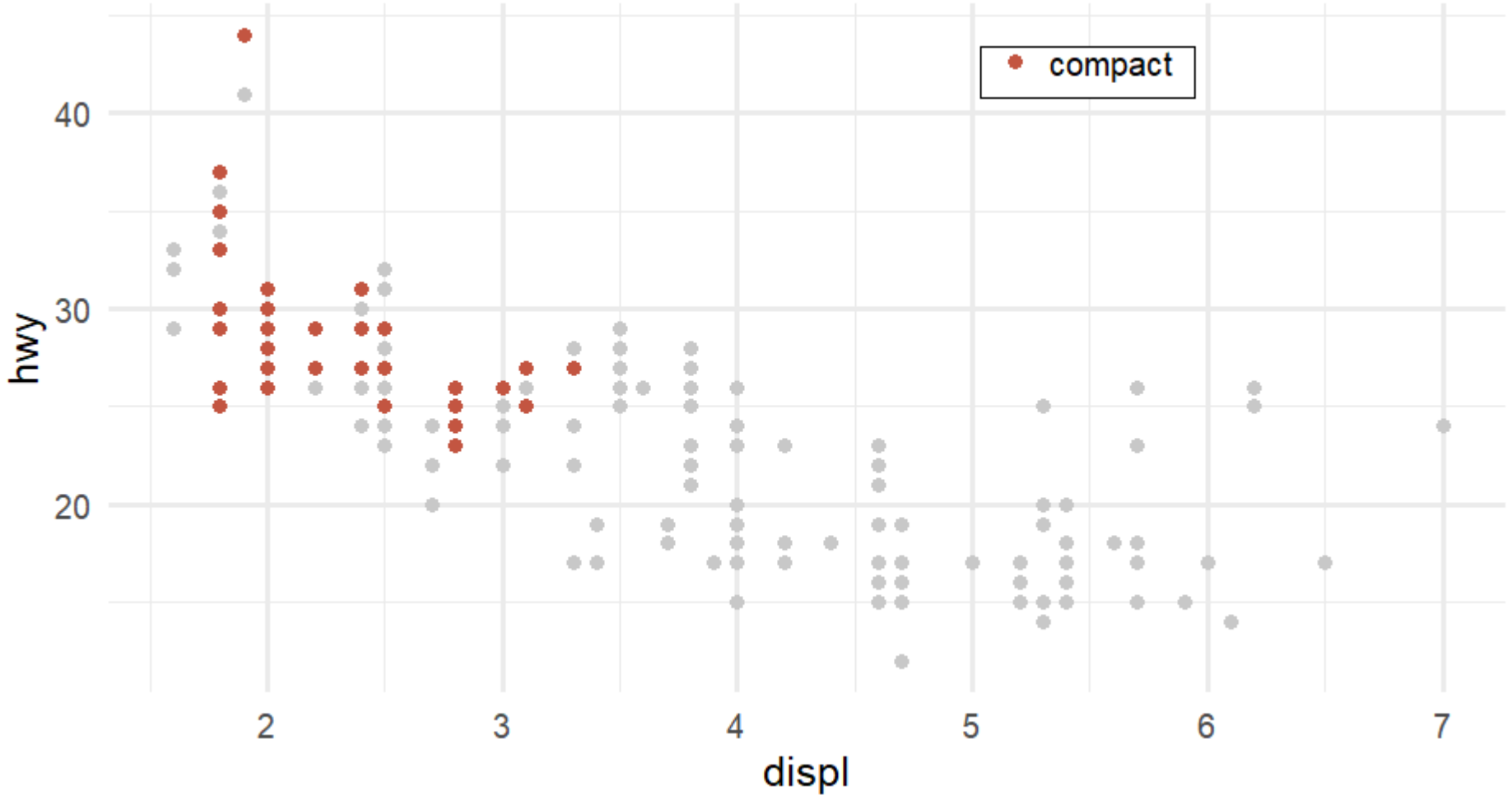


Add a legend

Couple of different ways - mostly hacky

- Pretend you have a column called "compact"

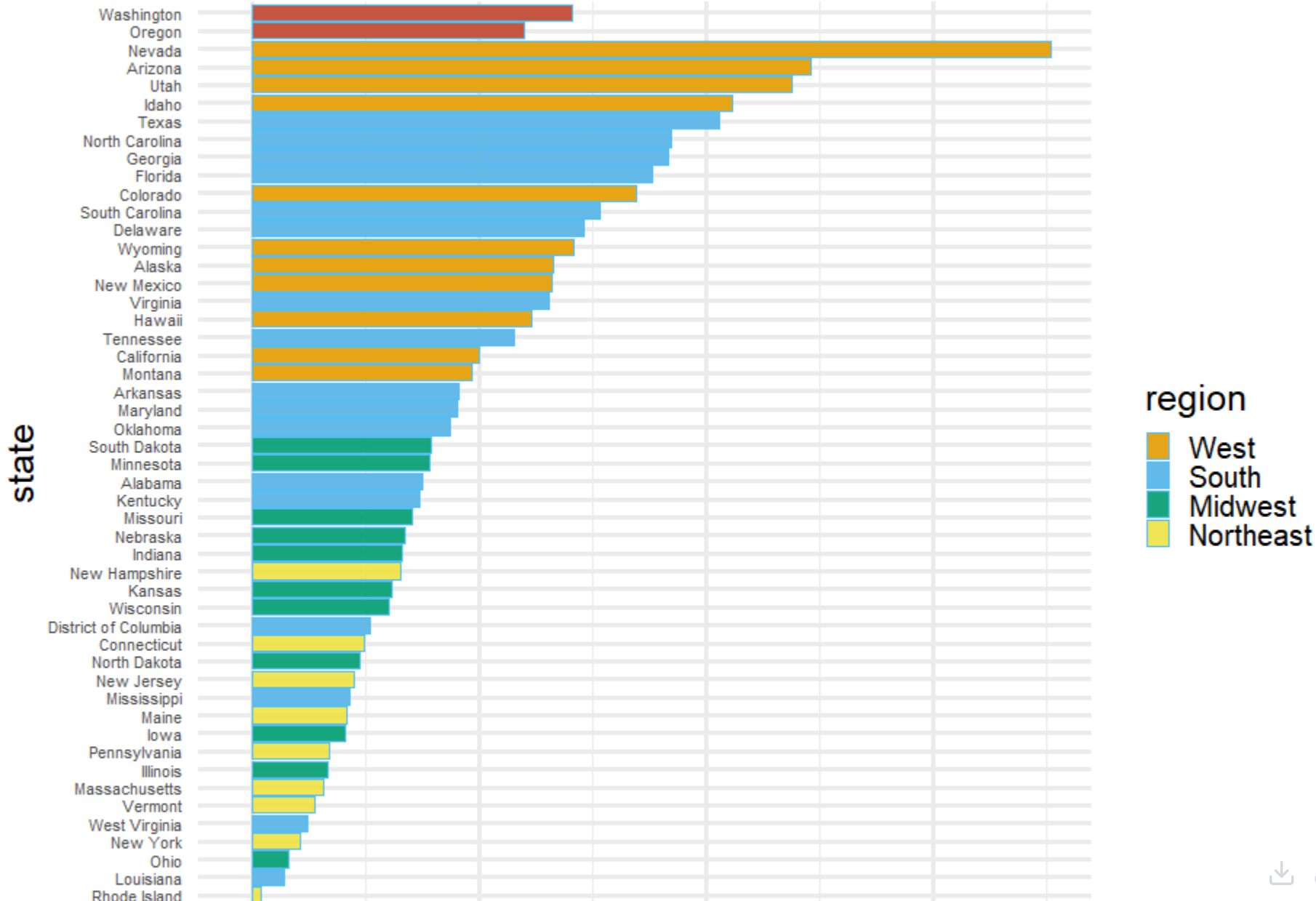
```
ggplot(mpg, aes(displ, hwy)) +
    geom_point(color = "gray80") +
    geom_point(data = filter(mpg, class == "compact"),
               aes(color = "compact")) +
  scale_color_manual(values = "#C55644") +
  theme(
    legend.title = element_blank(),
    legend.position = c(0.7, 0.9),
    legend.box.background = element_rect(colour = "black"),
    legend.box.margin = margin(t = -0.5, unit = "cm")
  )
```


compact
40
30
20
hwy
2
3
4
5
6
7
displ

Back to our states plot

Highlight Oregon and Washington

```
popgrowth_df <- popgrowth_df %>%
  arrange(desc(popgrowth)) %>%
  mutate(state = factor(state, levels = state))

ggplot(popgrowth_df, aes(x = popgrowth, y = state)) +
  geom_col(data = filter(popgrowth_df,
                         !(state %in% c("Oregon", "Washington"))),
           aes(fill = region),
           alpha = 0.9) +
    geom_col(data = filter(popgrowth_df,
                           state == "Oregon" |
                           state == "Washington"),
           fill = "#C55644") +
  scale_fill_OkabeIto()
```


state
Washington
Oregon
Nevada
Arizona
Utah
Idaho
Texas
North Carolina
Georgia
Florida
Colorado
South Carolina
Delaware
Wyoming
Alaska
New Mexico
Virginia
Hawaii
Tennessee
California
Montana
Arkansas
Maryland
Oklahoma
South Dakota
Minnesota
Alabama
Kentucky
Missouri
Nebraska
Indiana
New Hampshire
Kansas
Wisconsin
District of Columbia
Connecticut
North Dakota
New Jersey
Mississippi
Maine
Iowa
Pennsylvania
Illinois
Massachusetts
Vermont
West Virginia
New York
Ohio
Louisiana
Rhode Island
region
West
South
Midwest
Northeast

Color labels

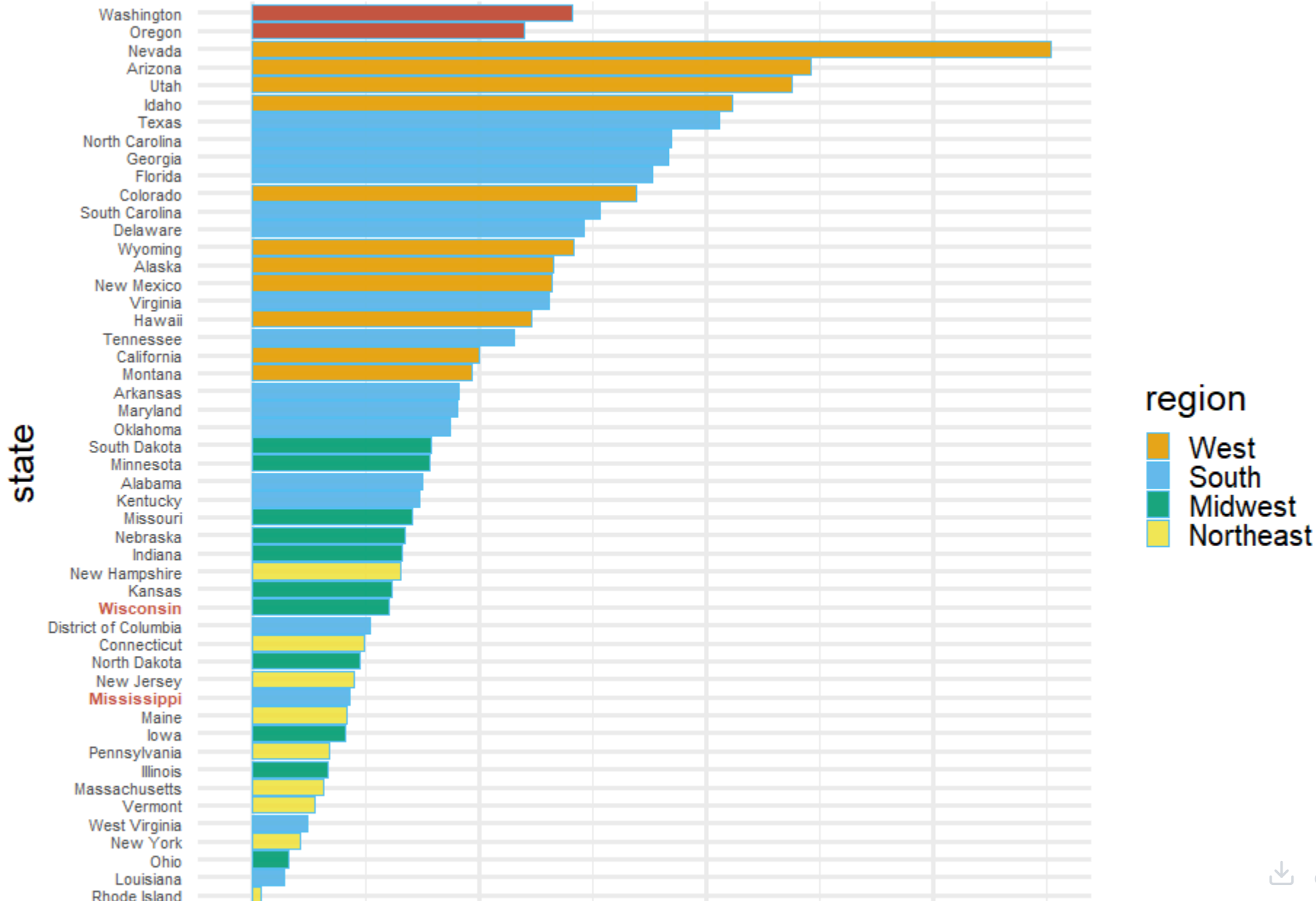
```
states <- unique(popgrowth_df$state)

label_color <- ifelse(states == "Oregon" | states == "Washington",
                "#C55644",
                "gray30")
label_color
```

```
##  [1] "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"
##  [9] "gray30"  "gray30"  "gray30"  "gray30"  "#C55644" "gray30"  "gray30"  "gray30"
## [17] "gray30"  "#C55644" "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"
## [25] "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"
## [33] "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"
## [41] "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"  "gray30"
## [49] "gray30"  "gray30"  "gray30"
```

```
label_face <- ifelse(states == "Oregon" | states == "Washington",
                "bold",
                "plain")

label_face
```

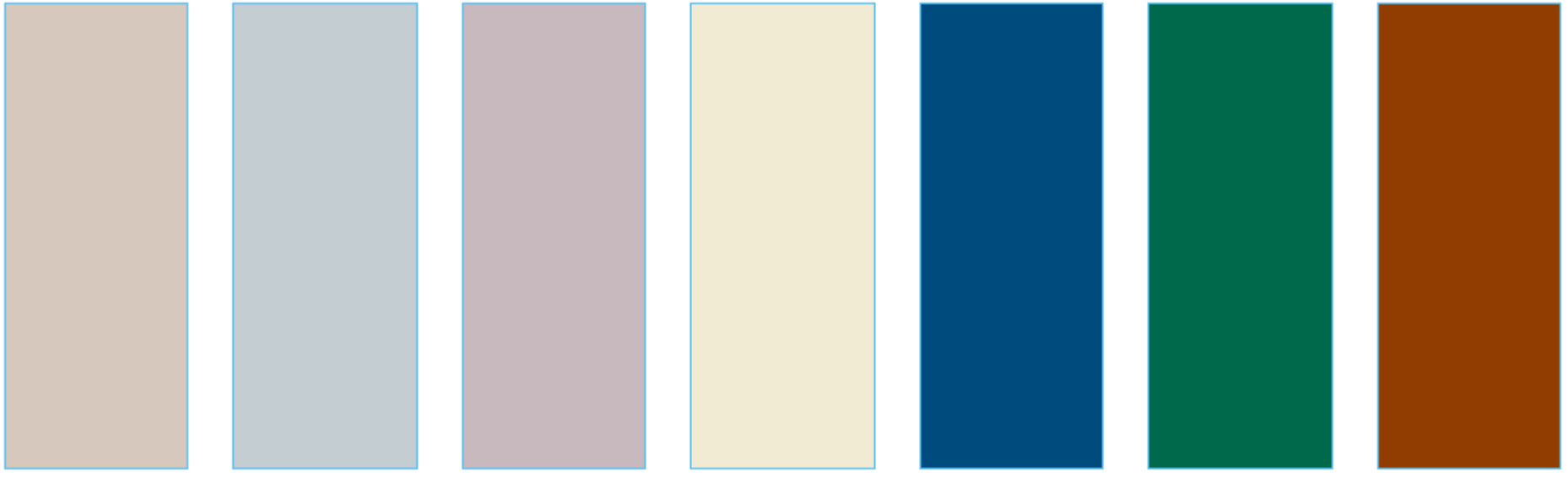
```
##  [1] "plain" "plain" "plain" "plain" "plain" "plain" "plain" "plain" "plain" "plain"
## [11] "plain" "plain" "bold"  "plain" "plain" "plain" "plain" "bold"  "plain" "plain"
## [21] "plain" "plain" "plain" "plain" "plain" "plain" "plain" "plain" "plain" "plain"
```

```
ggplot(popgrowth_df, aes(x = popgrowth, y = state)) +
  geom_col(data = filter(popgrowth_df,
                         !(state %in% c("Oregon", "Washington"))),
           aes(fill = region),
           alpha = 0.9) +
    geom_col(data = filter(popgrowth_df,
                           state == "Oregon" |
                           state == "Washington"),
           fill = "#C55644") +
  scale_fill_OkabeIto() +
  theme(
    axis.text.y = element_text(
      color = label_color,
      face = label_face
    )
  )
```

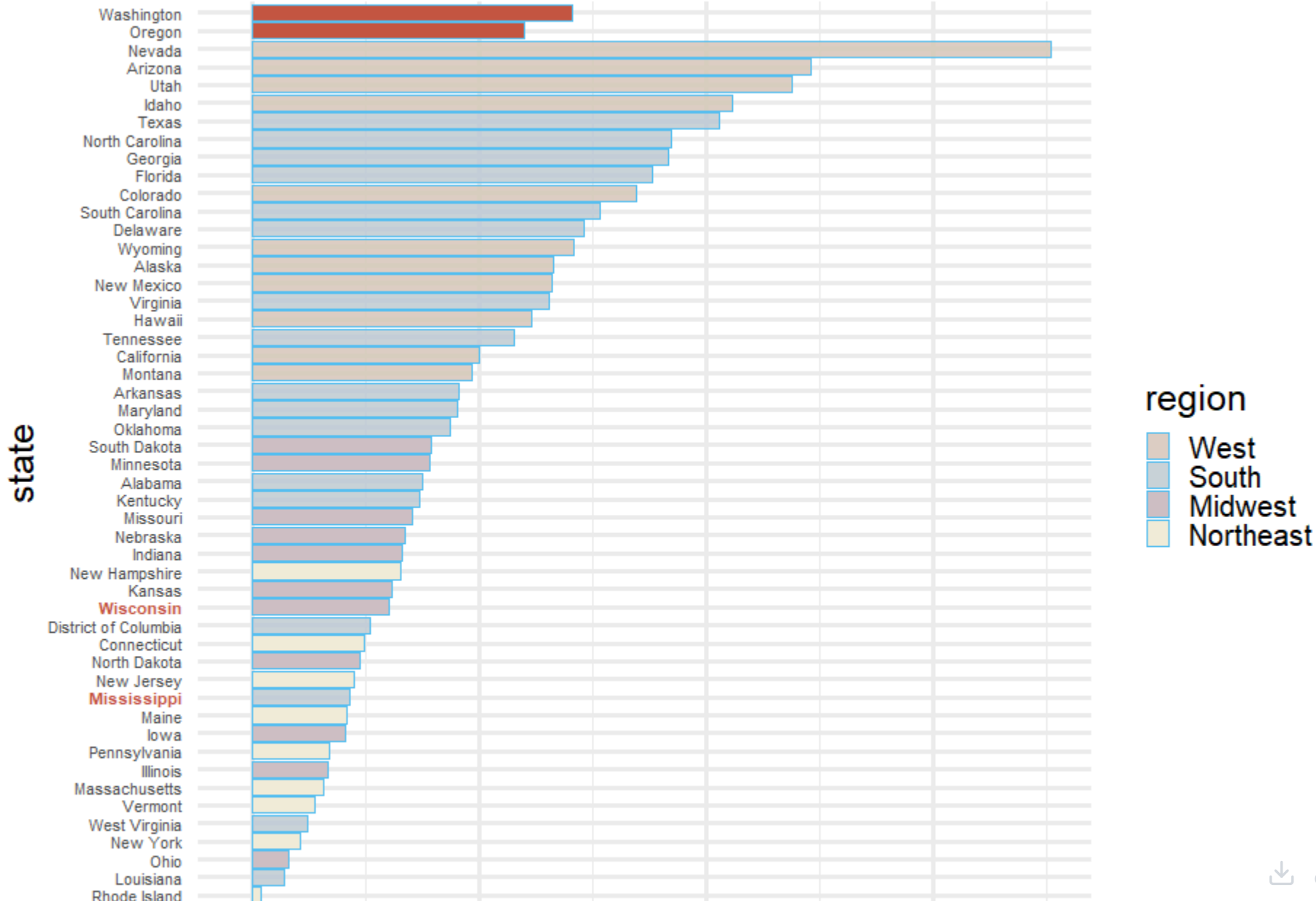

state
Washington
Oregon
Nevada
Arizona
Utah
Idaho
Texas
North Carolina
Georgia
Florida
Colorado
South Carolina
Delaware
Wyoming
Alaska
New Mexico
Virginia
Hawaii
Tennessee
California
Montana
Arkansas
Maryland
Oklahoma
South Dakota
Minnesota
Alabama
Kentucky
Missouri
Nebraska
Indiana
New Hampshire
Kansas
Wisconsin
District of Columbia
Connecticut
North Dakota
New Jersey
Mississippi
Maine
Iowa
Pennsylvania
Illinois
Massachusetts
Vermont
West Virginia
New York
Ohio
Louisiana
Rhode Island
region
West
South
Midwest
Northeast

Even better

```
accent_OkabeIto <- palette_OkabeIto[c(1, 2, 7, 4, 5, 3, 6)]
accent_OkabeIto[1:4] <- desaturate(lighten(accent_OkabeIto[1:4], .4), .8)
accent_OkabeIto[5:7] <- darken(accent_OkabeIto[5:7], .3)
gg_color_swatches(7) +
    scale_fill_manual(values = accent_OkabeIto)
```



```
ggplot(popgrowth_df, aes(x = popgrowth, y = state)) +
  geom_col(data = filter(popgrowth_df,
                         !(state %in% c("Oregon", "Washington"))),
           aes(fill = region),
           alpha = 0.9) +
    geom_col(data = filter(popgrowth_df,
                           state == "Oregon" |
                           state == "Washington"),
           fill = "#C55644") +
  scale_fill_manual(values = accent_OkabeIto) +
  theme(axis.text.y = element_text(color = label_color,
                                   face = label_face))
```

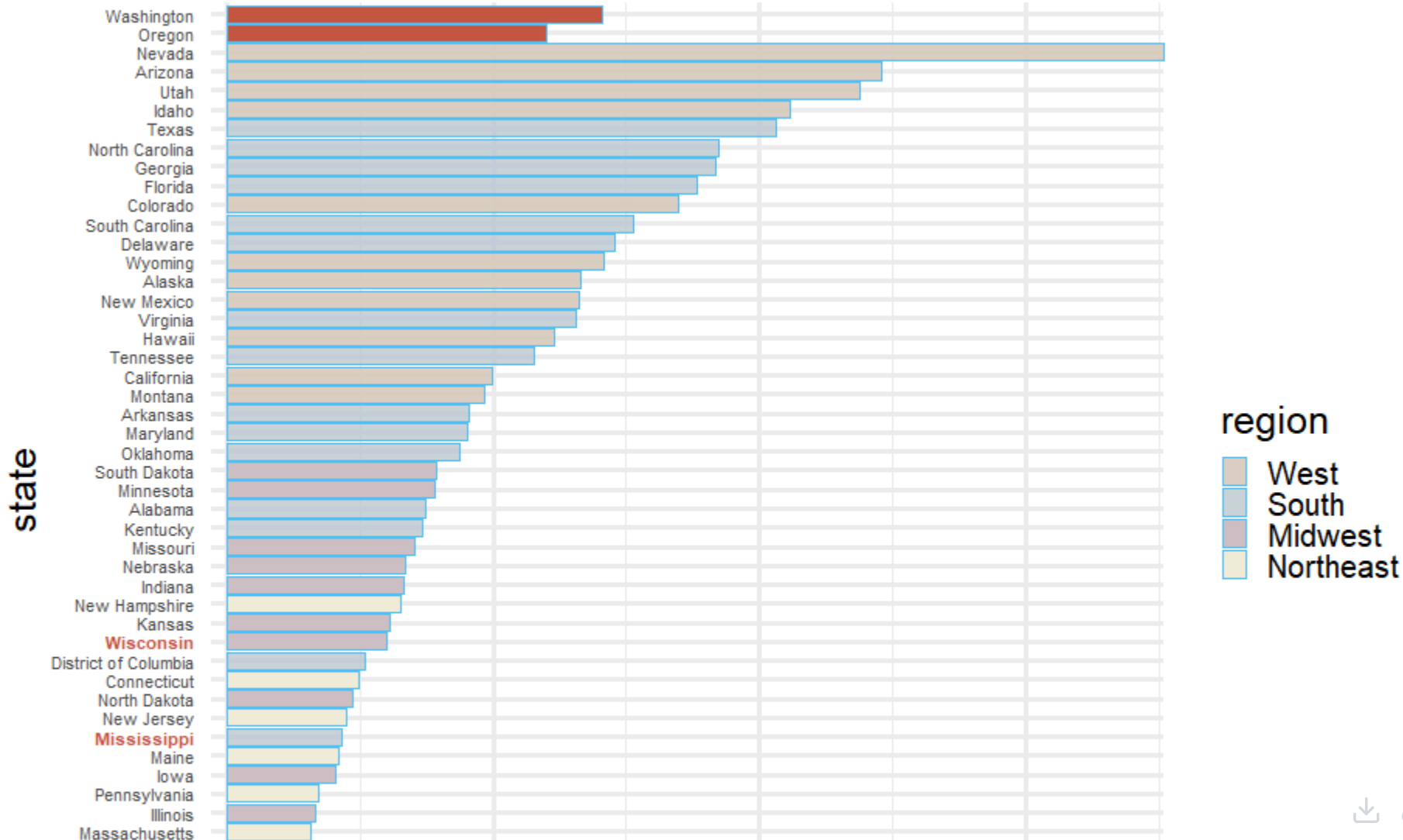

state
Washington
Oregon
Nevada
Arizona
Utah
Idaho
Texas
North Carolina
Georgia
Florida
Colorado
South Carolina
Delaware
Wyoming
Alaska
New Mexico
Virginia
Hawaii
Tennessee
California
Montana
Arkansas
Maryland
Oklahoma
South Dakota
Minnesota
Alabama
Kentucky
Missouri
Nebraska
Indiana
New Hampshire
Kansas
Wisconsin
District of Columbia
Connecticut
North Dakota
New Jersey
Mississippi
Maine
Iowa
Pennsylvania
Illinois
Massachusetts
Vermont
West Virginia
New York
Ohio
Louisiana
Rhode Island
region
West
South
Midwest
Northeast

Or even better (imo)

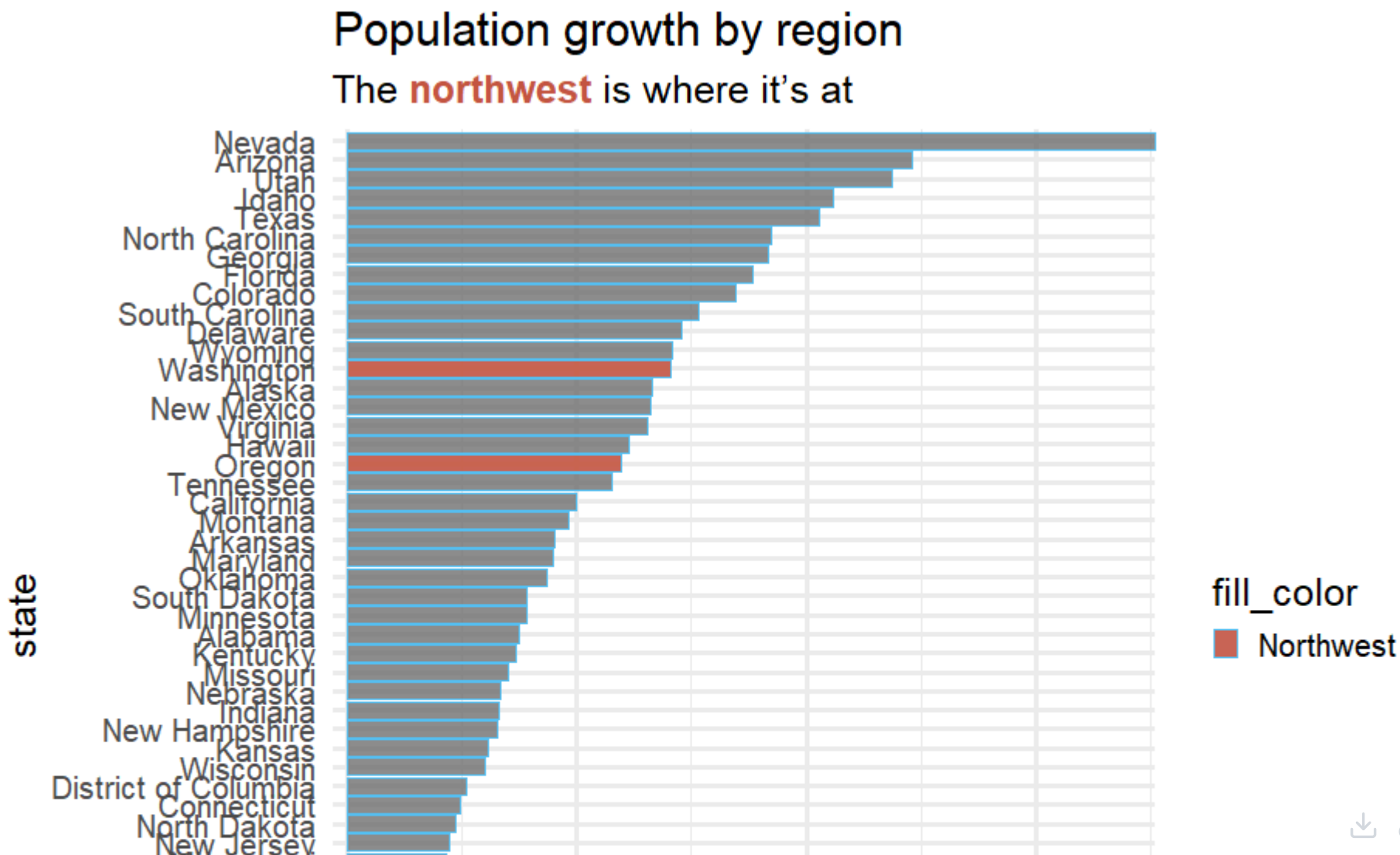
```
library(ggtext)
ggplot(popgrowth_df, aes(x = popgrowth, y = state)) +
  geom_col(data = filter(popgrowth_df,
                         !(state %in% c("Oregon", "Washington"))),
           aes(fill = region),
           alpha = 0.9) +
    geom_col(data = filter(popgrowth_df,
                           state == "Oregon" |
                           state == "Washington"),
           fill = "#C55644") +
  scale_fill_manual(values = accent_OkabeIto) +
  scale_x_continuous(expand = c(0, 0)) +
  labs(title = "Population growth by region",
       subtitle = "The <span style = 'color: #C55644'>**northwest**</span> is where it's at") +
  theme(axis.text.y = element_text(color = label_color,
                                   face = label_face),
        plot.subtitle = element_markdown())
```

Population growth by region

The **northwest** is where it's at



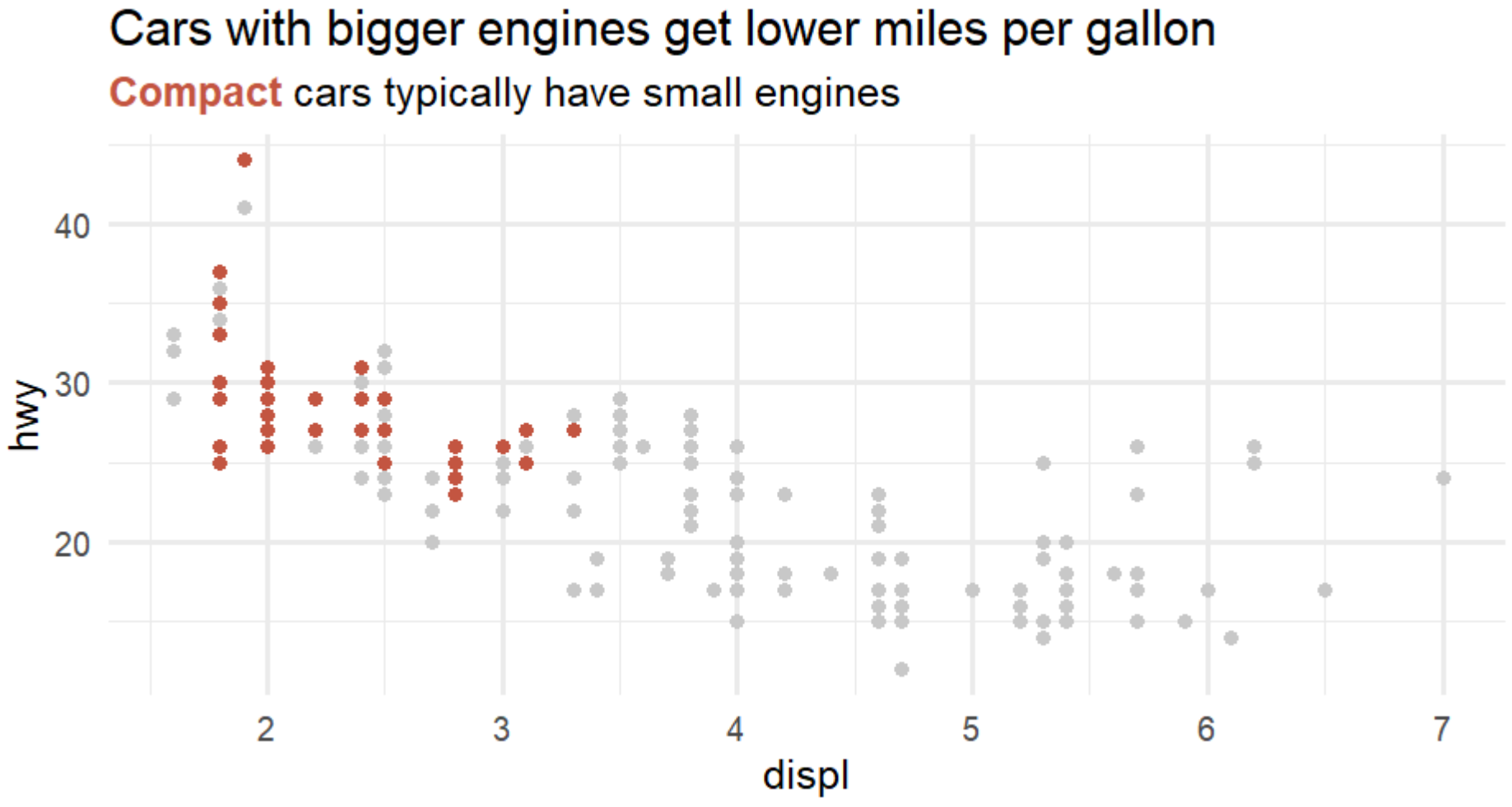
One last attempt



Back to cars plot

We can do this same thing and avoid the legend altogether.

```
ggplot(mpg, aes(displ, hwy)) +
    geom_point(color = "gray80") +
    geom_point(data = filter(mpg, class == "compact"),
               color = "#C55644") +
  labs(
    title = "Cars with bigger engines get lower miles per gallon",
    subtitle = "<span style = 'color: #C55644'>**Compact**</span> cars typically have small engines"
  ) +
  theme(plot.subtitle = element_markdown())
```


Cars with bigger engines get lower miles per gallon
Compact cars typically have small engines
hwy
40
30
20
2
3
4
5
6
7
displ

Data viz in Wild

Erick

Emily

Saratessa and Sophia on deck

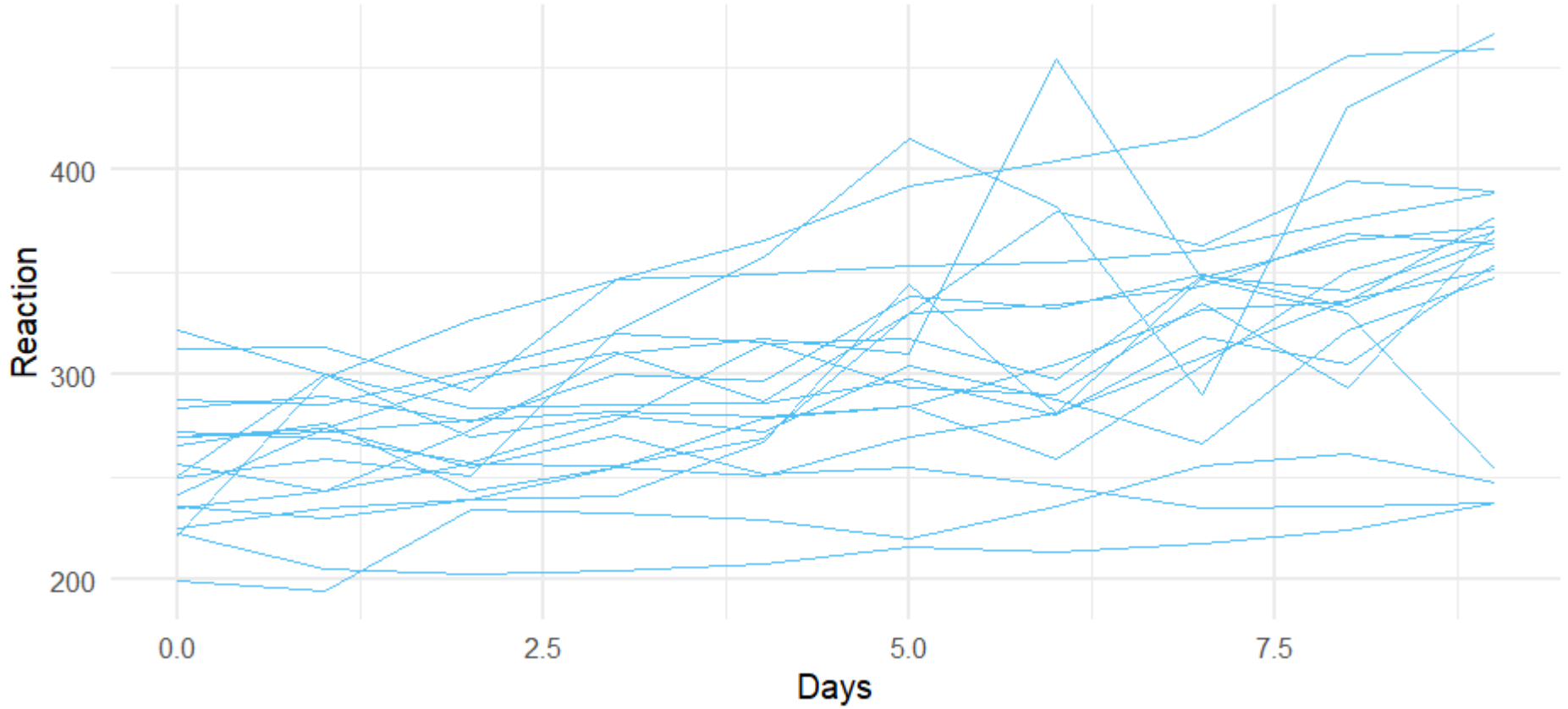
Last example

```
data(sleepstudy, package = "lme4")
head(sleepstudy)
```

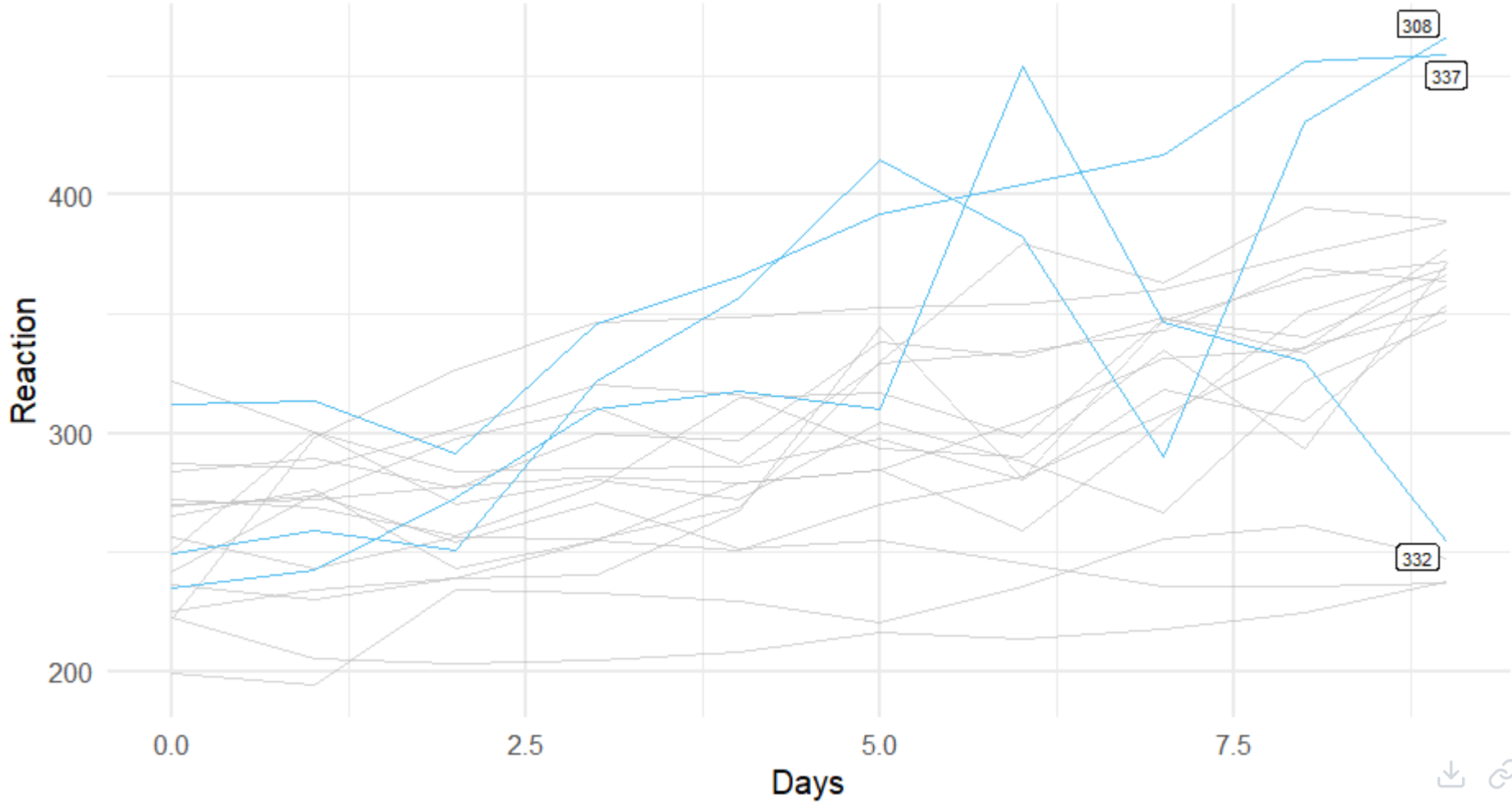
```
##   Reaction Days Subject
## 1      250    0     308
## 2      259    1     308
## 3      251    2     308
## 4      321    3     308
## 5      357    4     308
## 6      415    5     308
```

Plot by subject

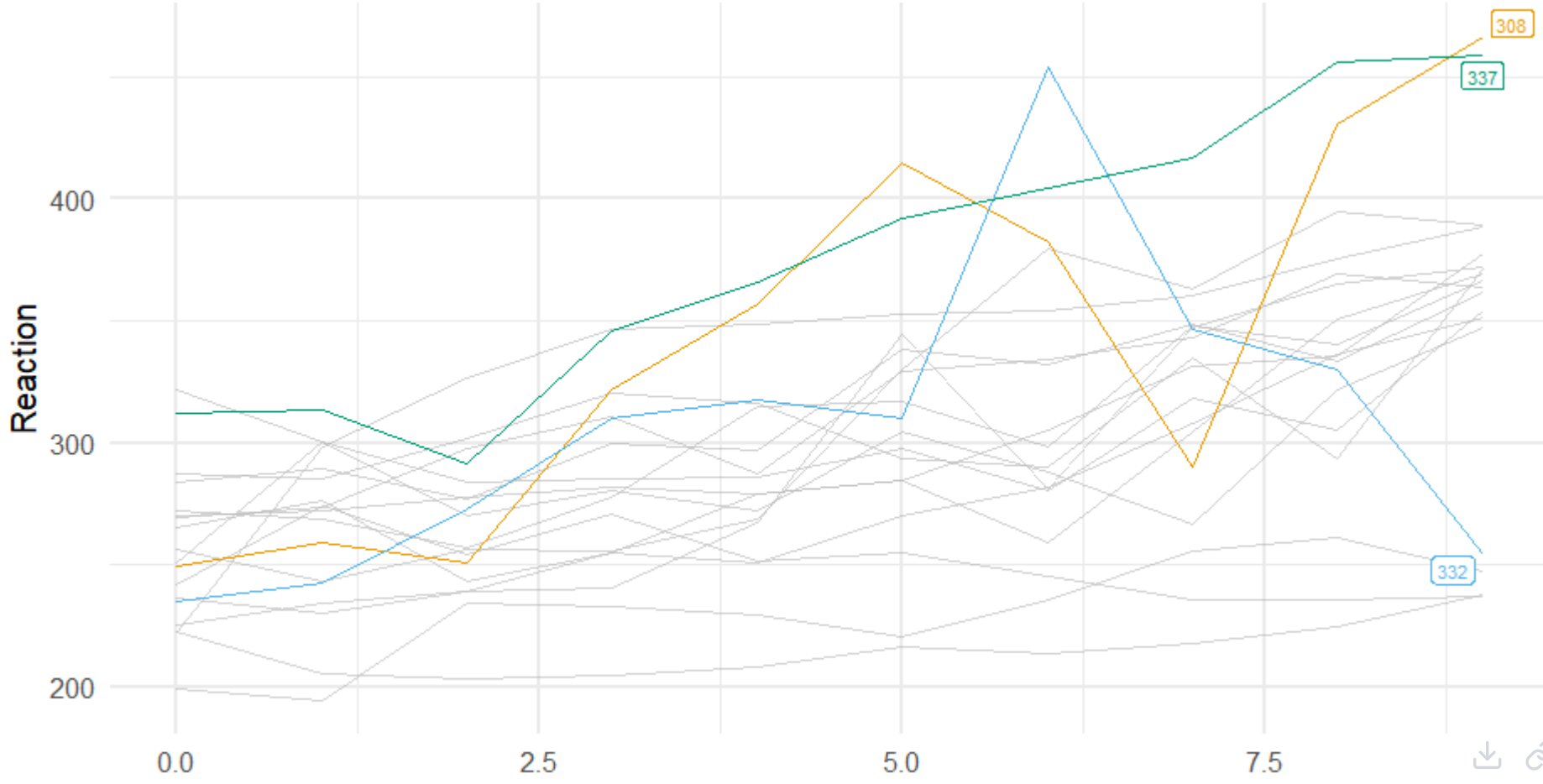
```
ggplot(sleepstudy, aes(Days, Reaction, group = Subject)) +
    geom_line()
```



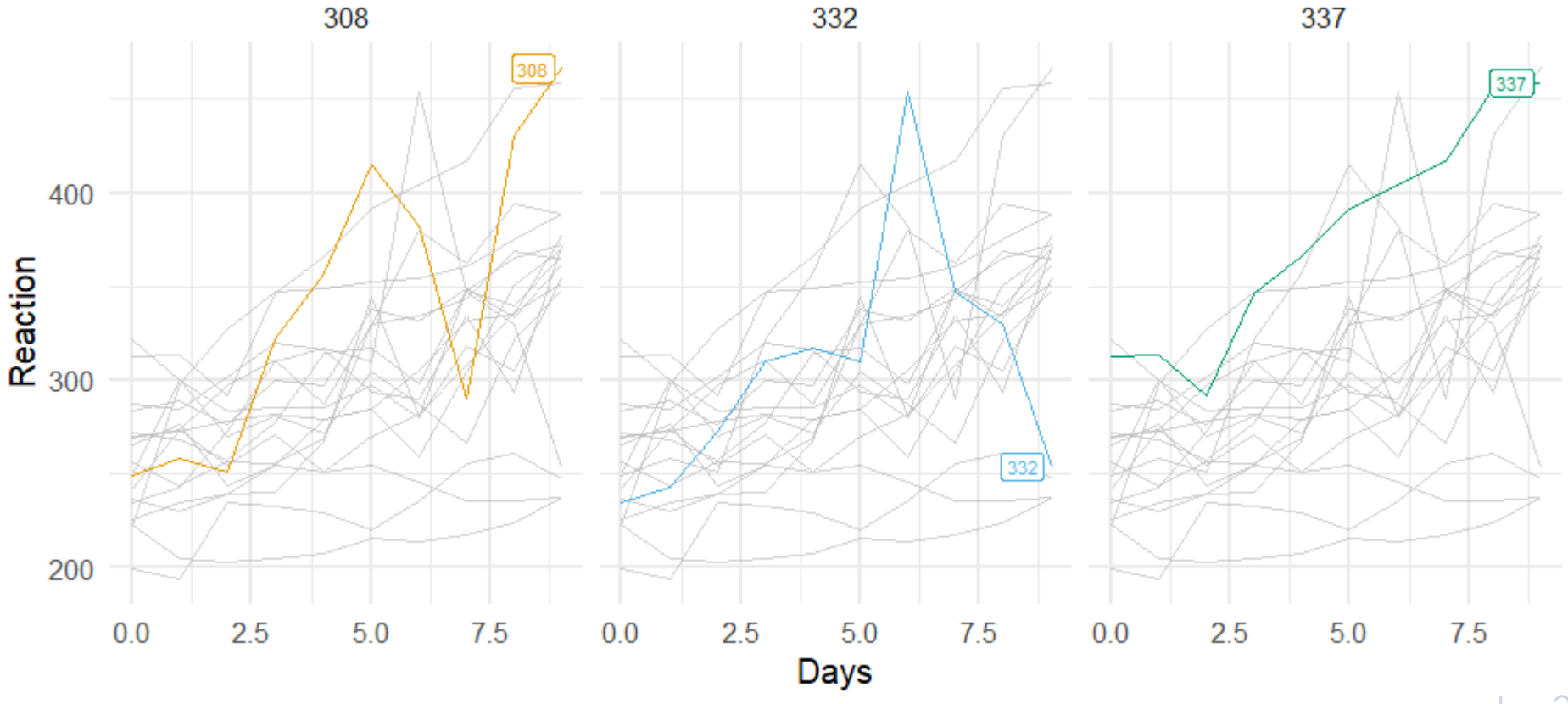
```
library(gghighlight)
ggplot(sleepstudy, aes(Days, Reaction, group = Subject)) +
    geom_line() +
gghighlight(max(Reaction) > 400)
```



```
library(gghighlight)
ggplot(sleepstudy, aes(Days, Reaction, color = Subject)) +
    geom_line() +
    gghighlight(max(Reaction) > 400) +
scale_color_OkabeIto()
```



```r
library(gghighlight)
ggplot(sleepstudy, aes(Days, Reaction, color = Subject)) +
    geom_line() +
facet_wrap(~Subject) +
    gghighlight(max(Reaction) > 400) +
    scale_color_OkabeIto()
```



Highlighting

The gghighlight package is really neat, but I have had issues with it in the past.

If you have troubles, you can always create them more manually has we did previously

Example

This is a little tricky. First, create a subset of the data that only has the cases you want to highlight

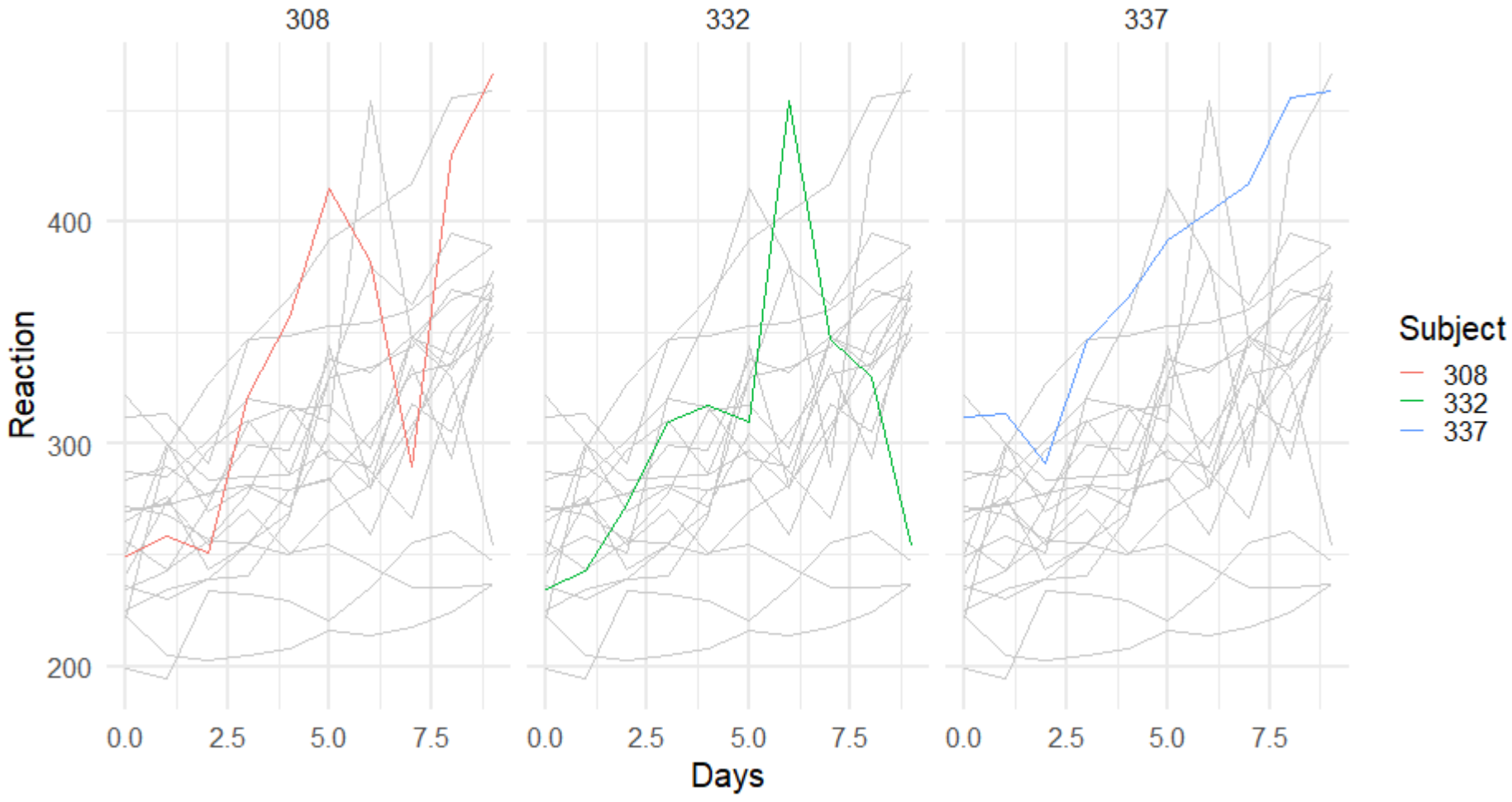
```
# filter
high <- sleepstudy %>%
  filter(Reaction > 400) %>%
  select(Subject)

# highlight dataframe
hl <- semi_join(sleepstudy, high)
```

Next, create the background data. We usually just *drop* the faceting column, but in this case we still need the id, so we'll rename it.

```
# rename facet column; use for grouping
bg_data <- sleepstudy %>%
  rename(id = Subject)
```

Finally, plot the highlight data, adding a layer with the background data

```
ggplot(hl, aes(Days, Reaction)) +
  geom_line(
    aes(group = id),
    color = "gray80",
    data = bg_data
  ) +
    geom_line(
      aes(color = Subject)
  ) +
    facet_wrap(~Subject)
```

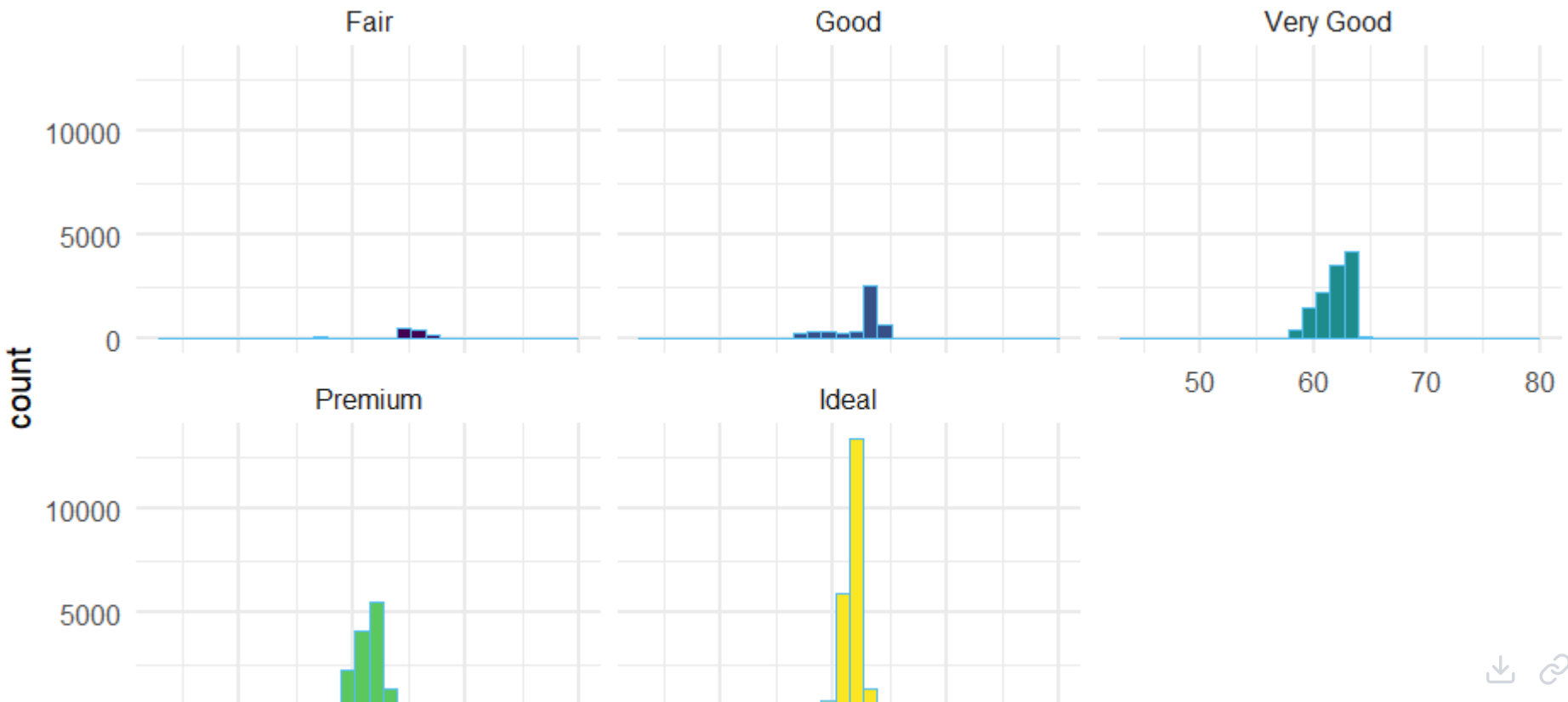

308
332
337
400
300
200
Reaction
0.0
2.5
5.0
7.5
Days
Subject
308
332
337

Very quick aside

You can do this with any data - a great alternative to stacked histograms.

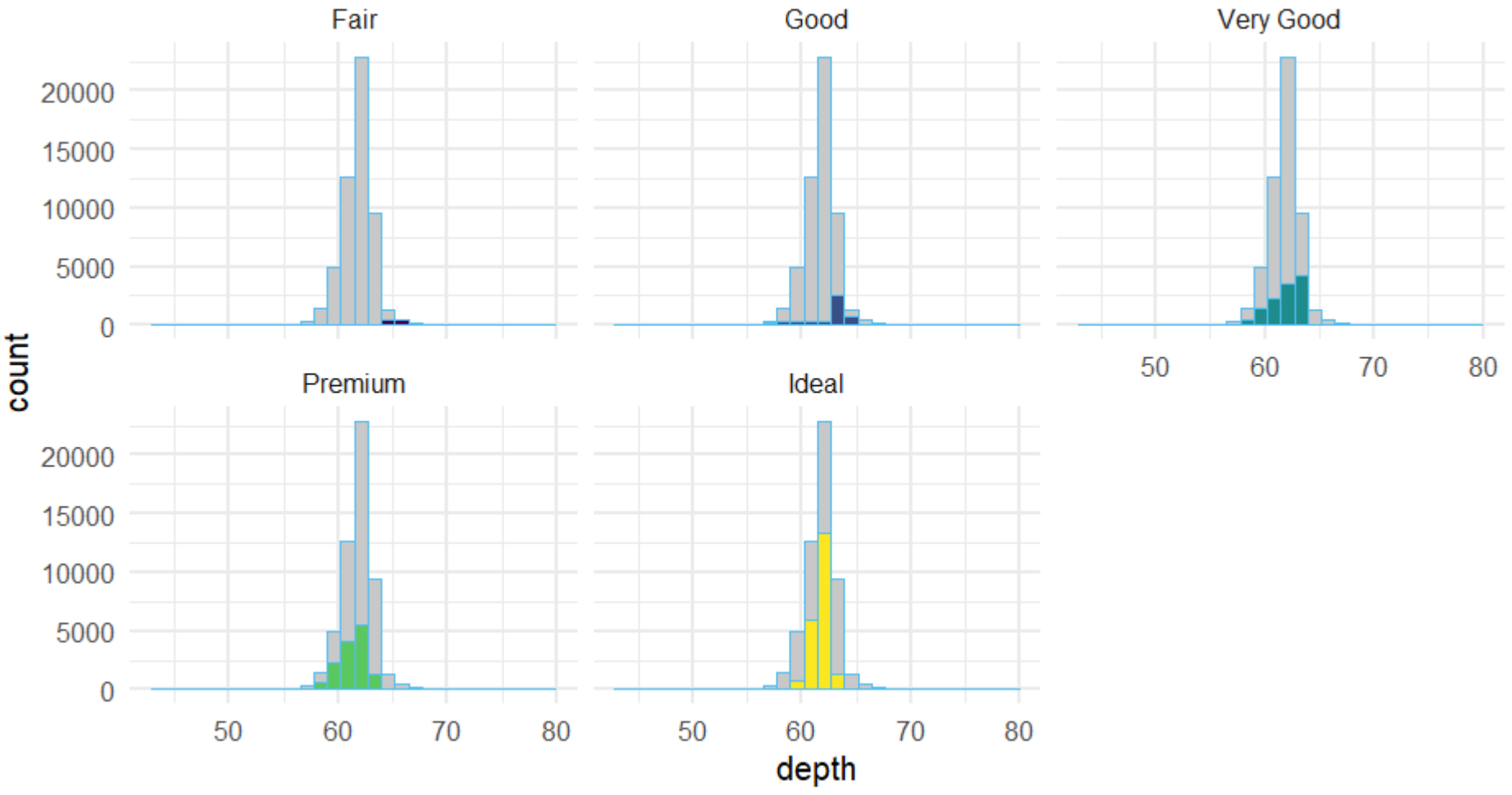
Standard faceted histogram

```
ggplot(diamonds, aes(depth)) +
  geom_histogram(aes(fill = cut)) +
  facet_wrap(~cut) +
  guides(fill = "none") # drop legend
```



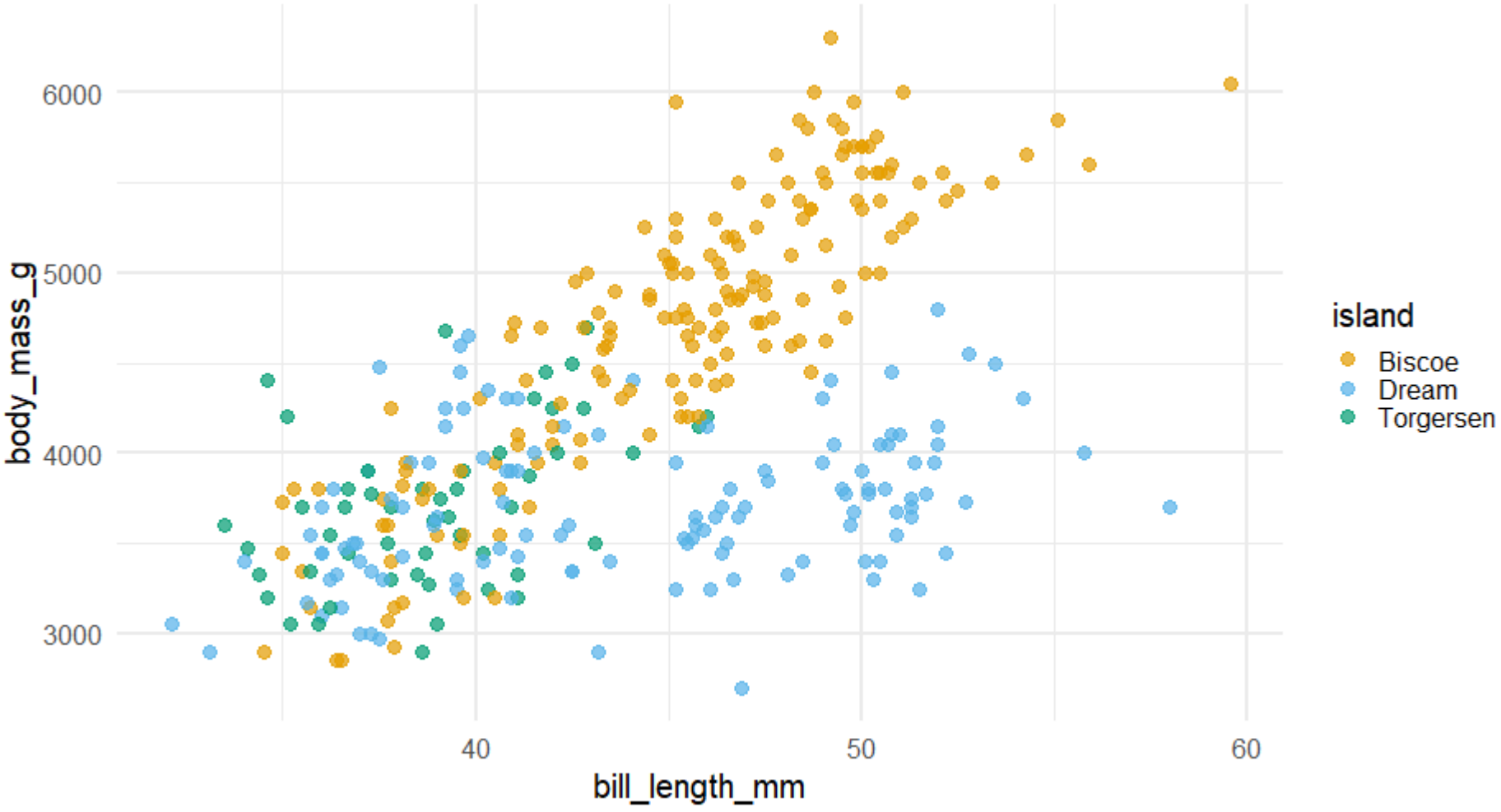
Add a background layer

```
ggplot(diamonds, aes(depth)) +
  geom_histogram(
    data = select(diamonds, -cut),
    fill = "gray80"
  ) +
  geom_histogram(aes(fill = cut)) +
  facet_wrap(~cut) +
  guides(fill = "none") # drop legend
```


Fair
Good
Very Good
Premium
Ideal
count
depth
0
5000
10000
15000
20000
50
60
70
80

A few other things to consider

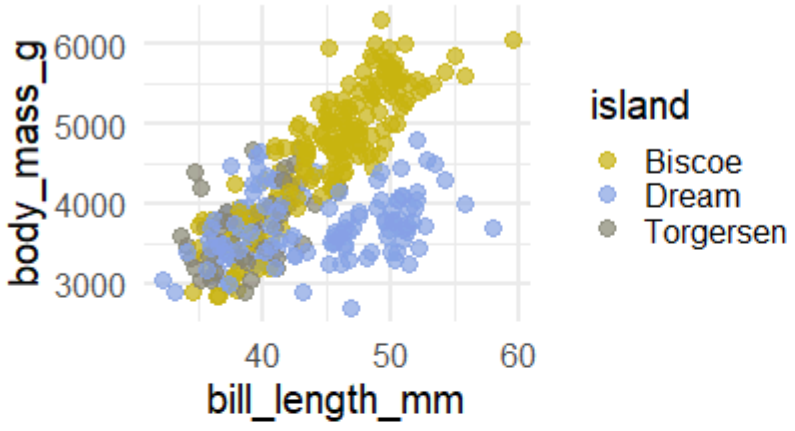
Double encodings



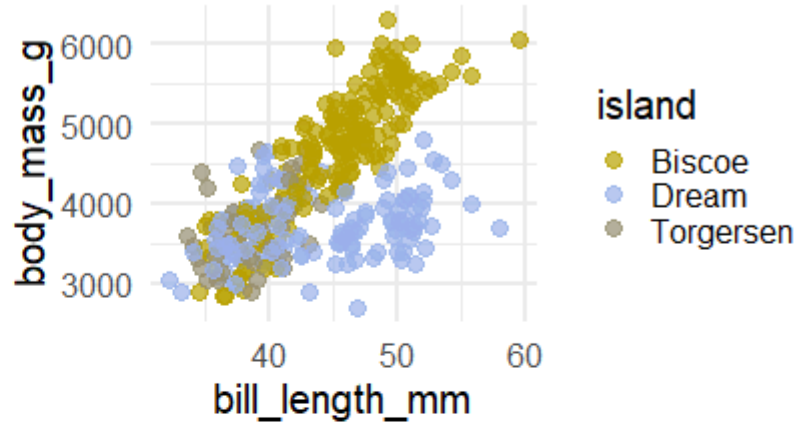
This plot is less than ideal. Why?

Color blindness

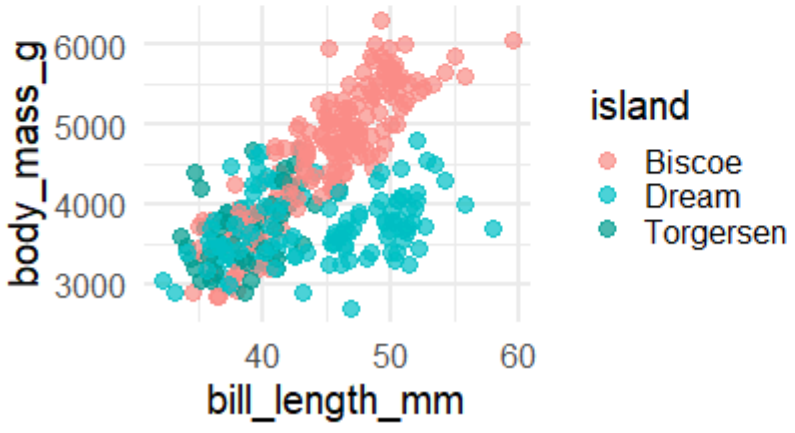
Deutanomaly



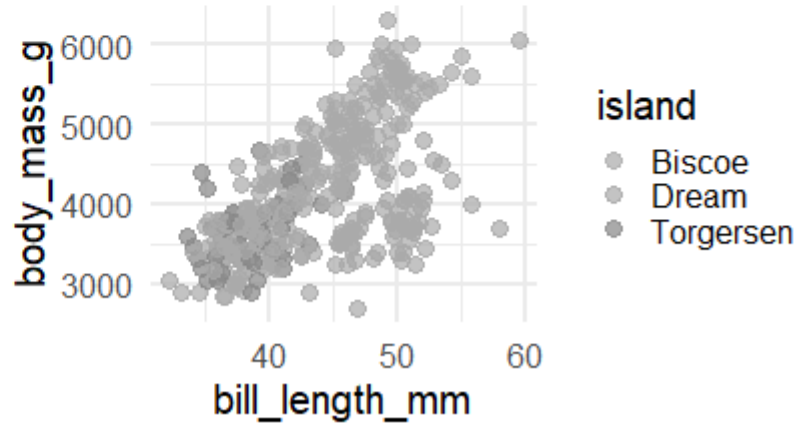
Protanomaly



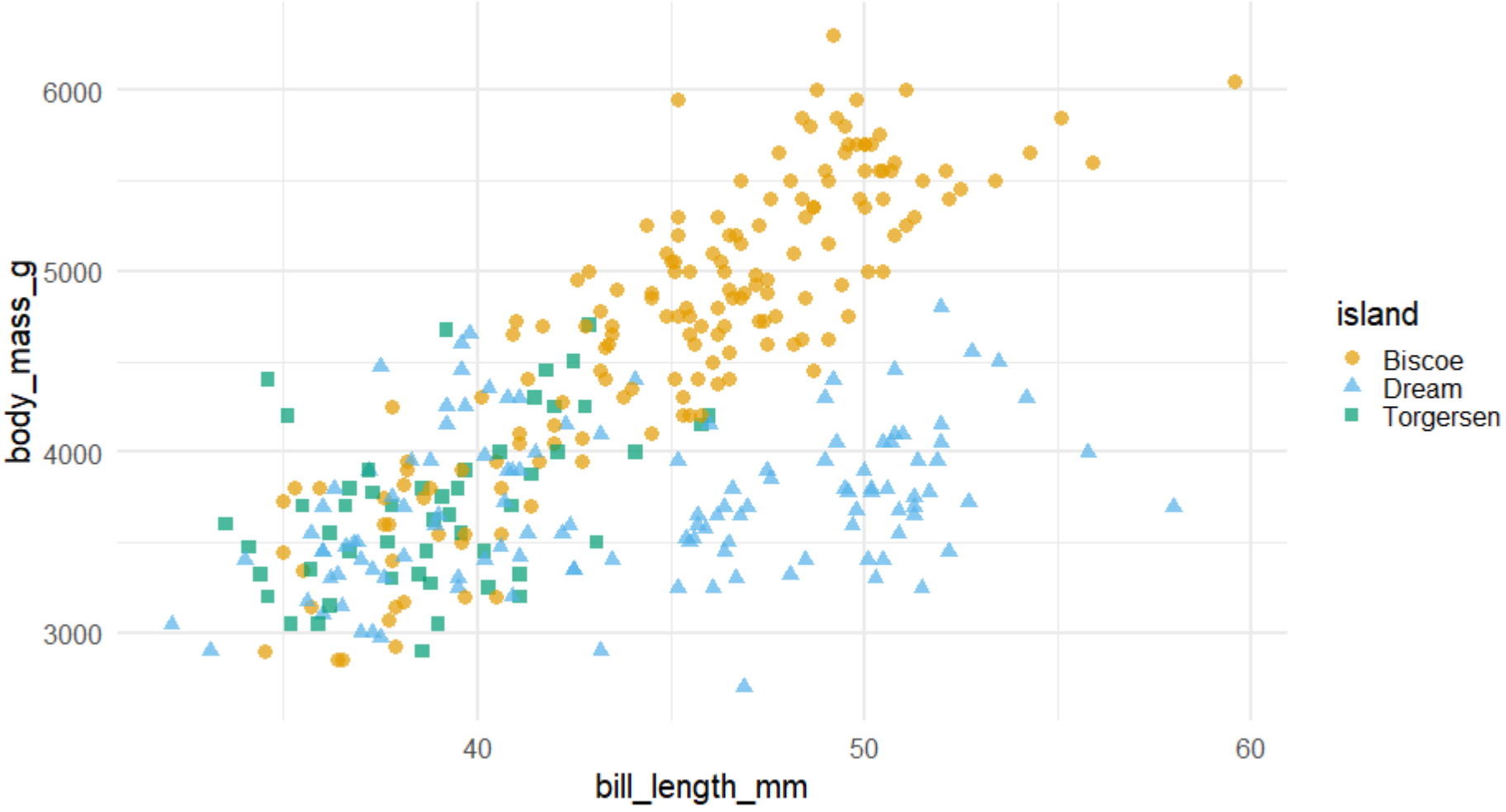
Tritanomaly



Desaturated

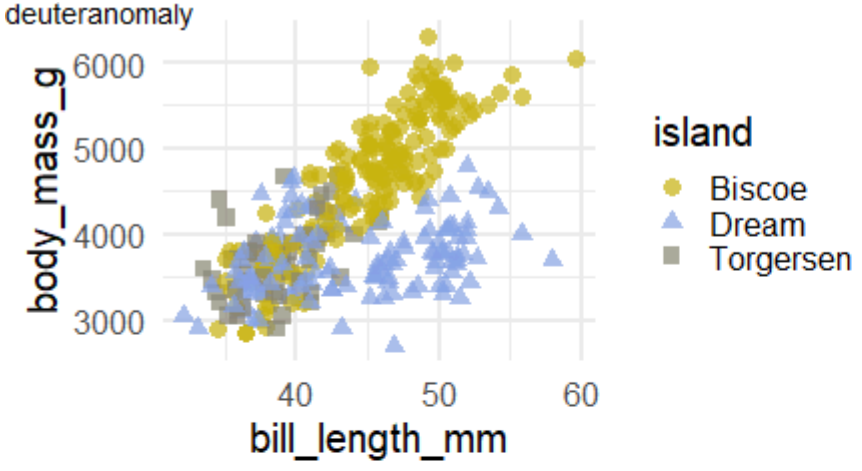


Better version

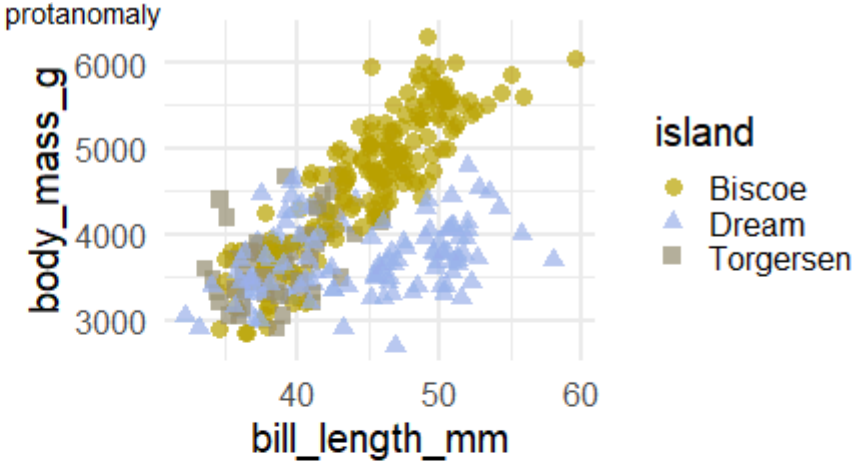


Color blindness check

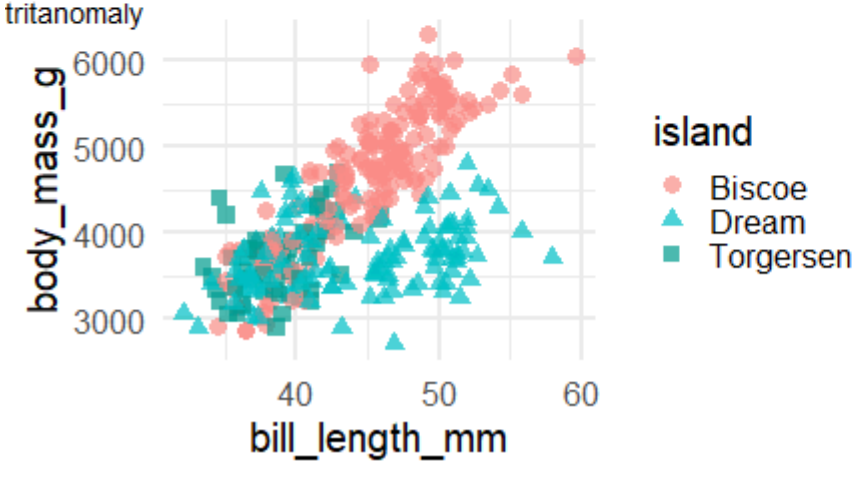
deuteranomaly



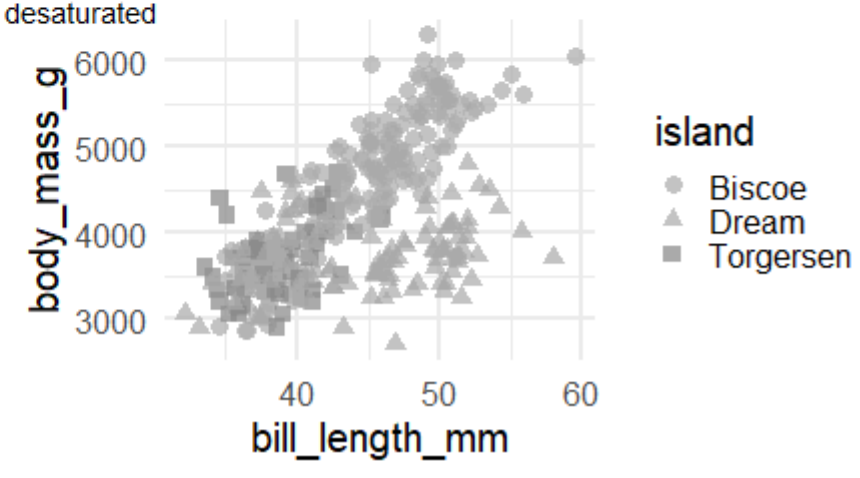
protanomaly



tritanomaly



desaturated

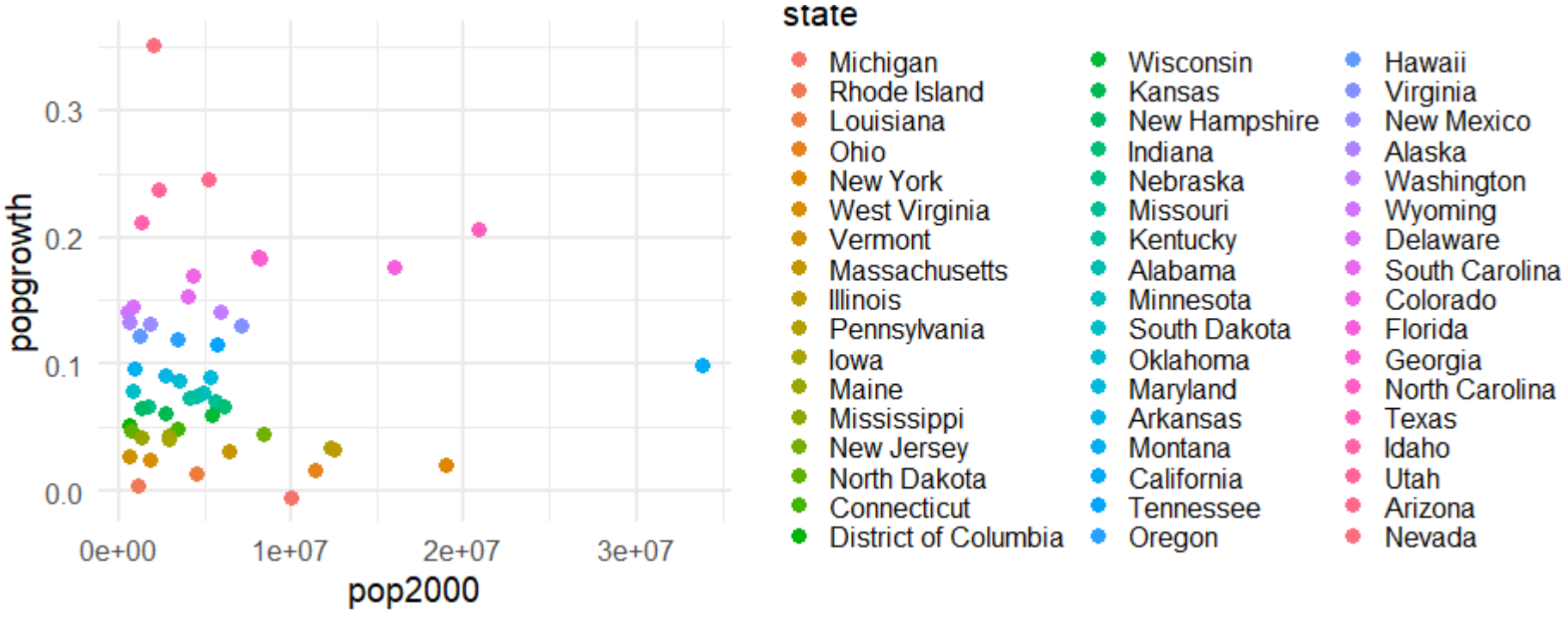


Common problems with color

Too many colors

More than 5-ish categories generally becomes too difficult to track

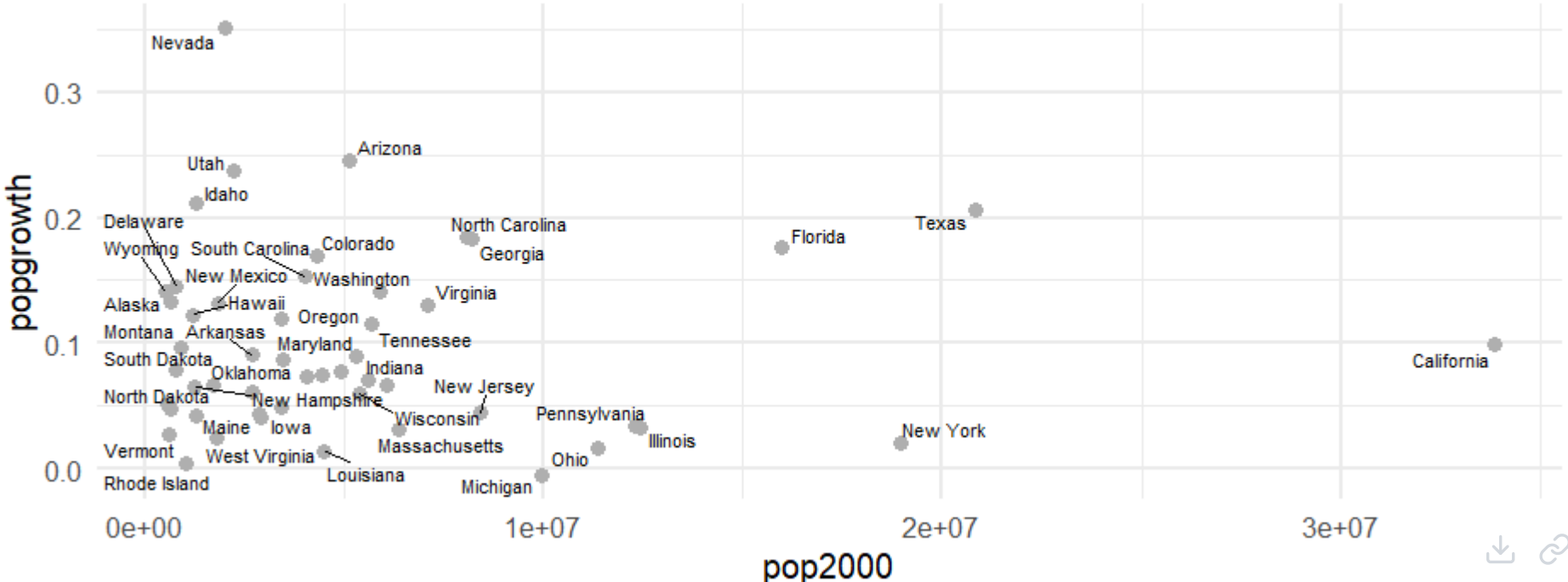
```
ggplot(popgrowth_df, aes(pop2000, popgrowth, color = state)) +
    geom_point()
```



Use labels

More than 5-ish categories generally becomes too difficult to track

```
library(ggrepel)

ggplot(popgrowth_df, aes(pop2000, popgrowth)) +
    geom_point(color = "gray70") +
geom_text_repel(aes(label = state))
```

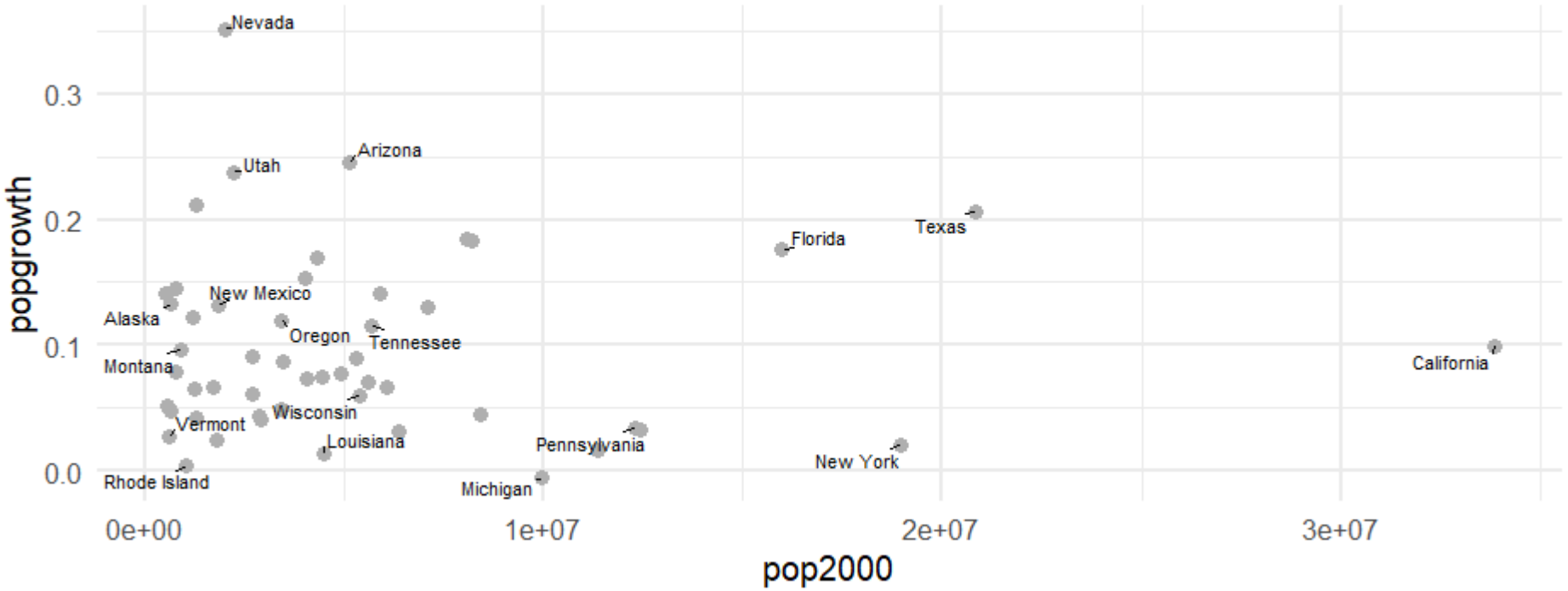


Better

Get a subset

```
to_label <- c("Alaska", "Arizona", "California", "Florida", "Wisconsin",
              "Louisiana", "Nevada", "Michigan", "Montana", "New Mexico",
              "Pennsylvania", "New York", "Oregon", "Rhode Island",
              "Tennessee", "Texas", "Utah", "Vermont")
subset_states <- popgrowth_df %>%
    filter(state %in% to_label)
```

```
library(ggrepel)

ggplot(popgrowth_df, aes(pop2000, popgrowth)) +
    geom_point(color = "gray70") +
geom_text_repel(aes(label = state),
                  data = subset_states,
                  min.segment.length = 0)
```



(still lots more cleaning up we could do here...)

Rainbow palette

```
rainbow(3)
```

```
## [1] "#FF0000" "#00FF00" "#0000FF"
```

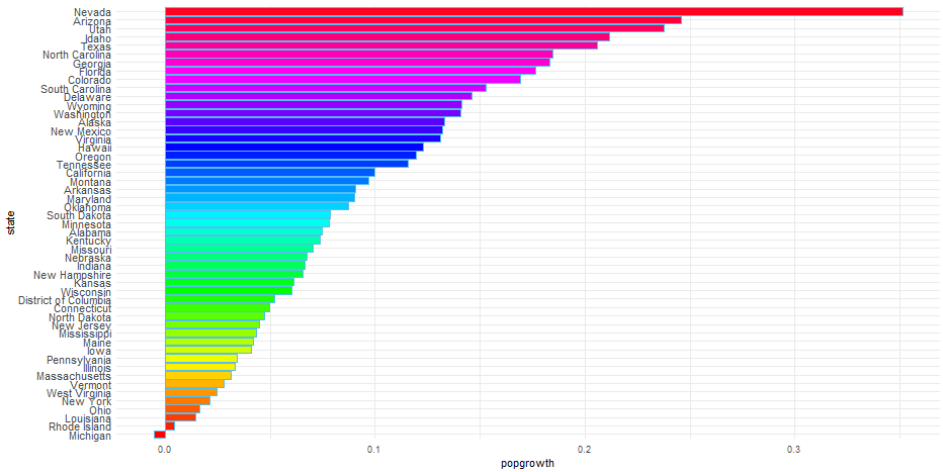
```
rainbow(7)
```

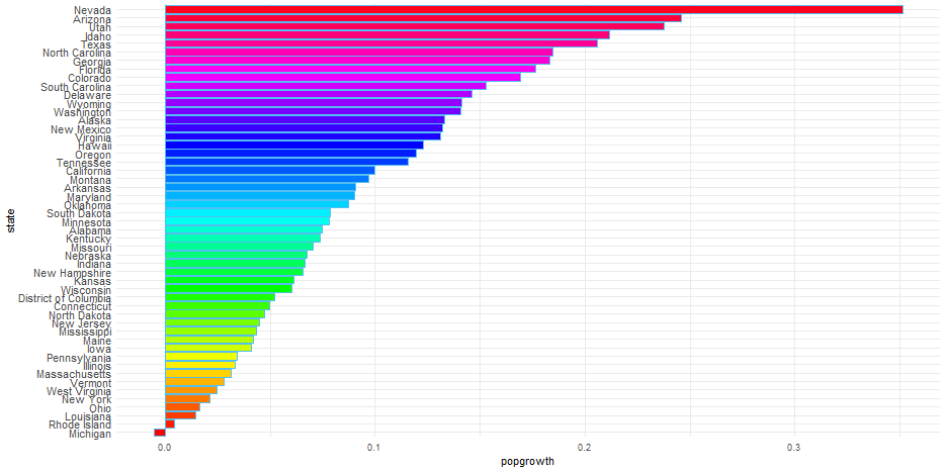
```
## [1] "#FF0000" "#FFDB00" "#49FF00" "#00FF92" "#0092FF" "#4900FF" "#FF00DB"
```

Pretty! Doesn't work well

See here for one (of many) articles on why this is the case

```
ggplot(popgrowth_df, aes(x = popgrowth, y = sta
  geom_col(aes(fill = state)) +
  scale_fill_manual(values = rainbow(51)) +
  guides(fill = "none")
```



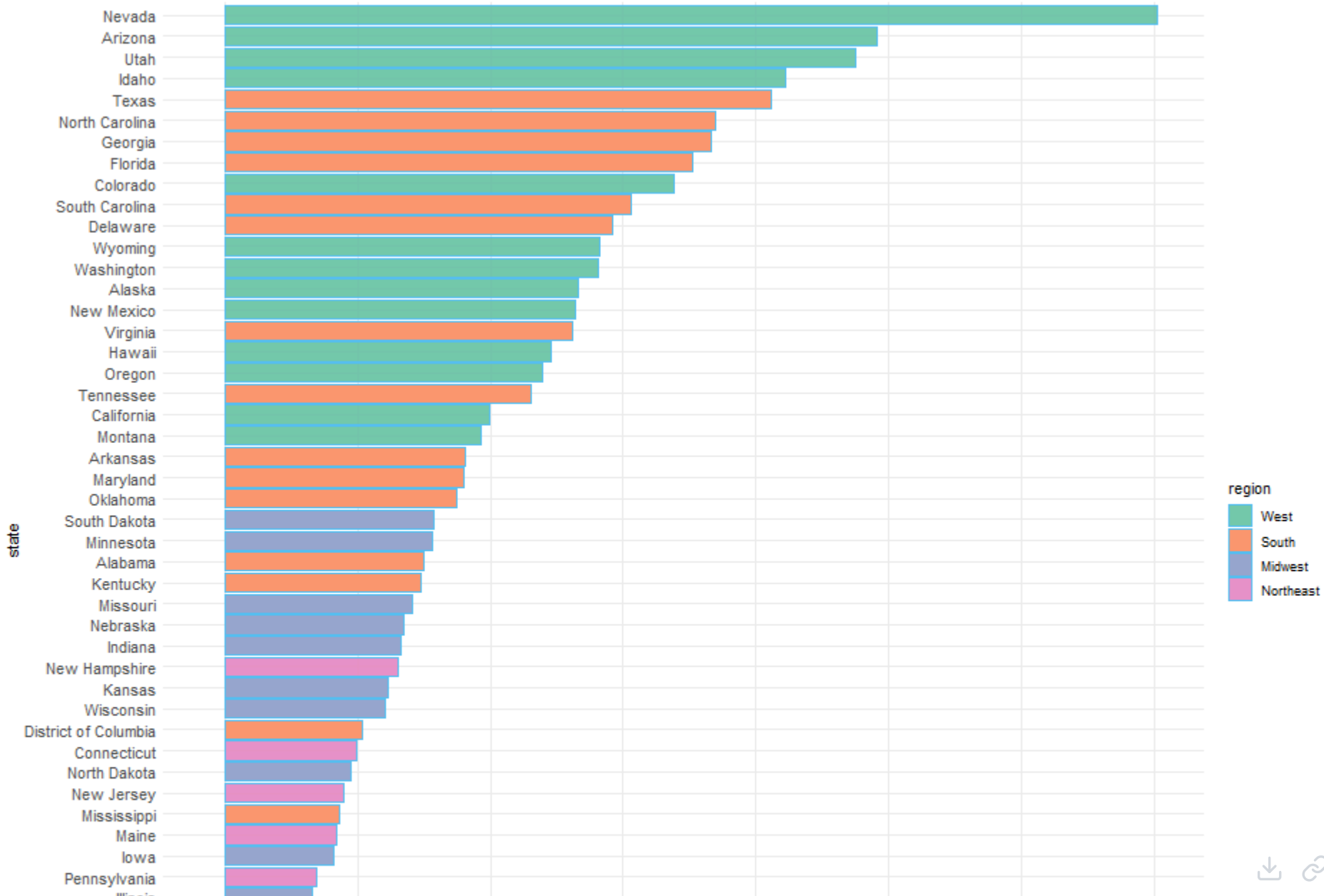


Last few notes on palettes

- Do some research, find what you like **and** what tends to work well
- Check for colorblindness
- Look into http://colorbrewer2.org/
    - `scale_color_brewer()` and `scale_fill_brewer()` ship with ggplot2

For example

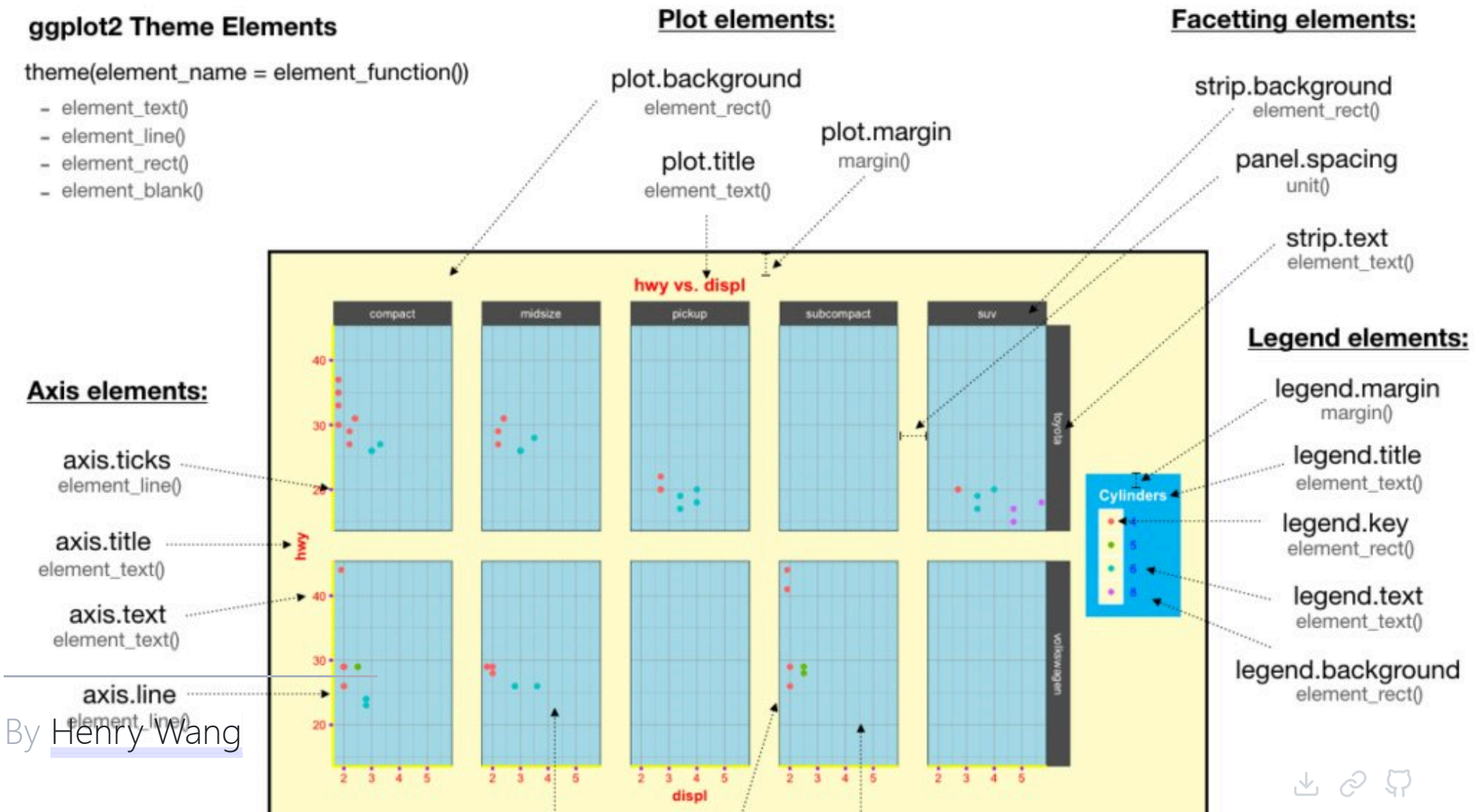
```
ggplot(popgrowth_df, aes(x = popgrowth, y = state)) +
  geom_col(aes(fill = region),
           alpha = 0.9) +
  scale_fill_brewer(palette = "Set2")
```


Nevada
Arizona
Utah
Idaho
Texas
North Carolina
Georgia
Florida
Colorado
South Carolina
Delaware
Wyoming
Washington
Alaska
New Mexico
Virginia
Hawaii
Oregon
Tennessee
California
Montana
Arkansas
Maryland
Oklahoma
South Dakota
Minnesota
Alabama
Kentucky
Missouri
Nebraska
Indiana
New Hampshire
Kansas
Wisconsin
District of Columbia
Connecticut
North Dakota
New Jersey
Mississippi
Maine
Iowa
Pennsylvania
state
region
West
South
Midwest
Northeast

Paleteer package

The anatomy of a `ggplot()` theme

Theme system



By Henry Wang

Theme elements

- Each element in the plot can be targeted
- Plot title = `plot.title`
- Grid lines = `panel.grid`
- Legend background = `legend.background`]

Theme functions

- Use special functions to manipulate specific elements
- Text-based things = `element_text()`
- Rectangular things (backgrounds) = `element_rect()`
- Line-based things (axis lines, grid lines) = `element_line()`
- Disable element completely = `element_blank()`

How to learn `theme()`

- The `theme()` function has **94** possible arguments(!!!)
- You can get hyper-specific with things like `axis.ticks.length.x.bottom`
- The only way to learn how to use `theme()`
  is to use it and tinker with it

Review Lab 3

Lab 4

Q & A with Dr. Daniel Anderson

Next time

More colors, themes, refinement of plots