

Designing data visualizations in R

making clear figures that communicate

2021-11-14

"We need to do everything we can to help our readers understand the meaning of our visualizations and see the same patterns in the data that we see. This usually means less is more. Simplify your figures as much as possible. Remove all features that are tangential to your story"

— Claus O. Wilke

act one: focus and declutter

act one: focus and declutter

act two: narrate and put in context

The cast of characters

Packages

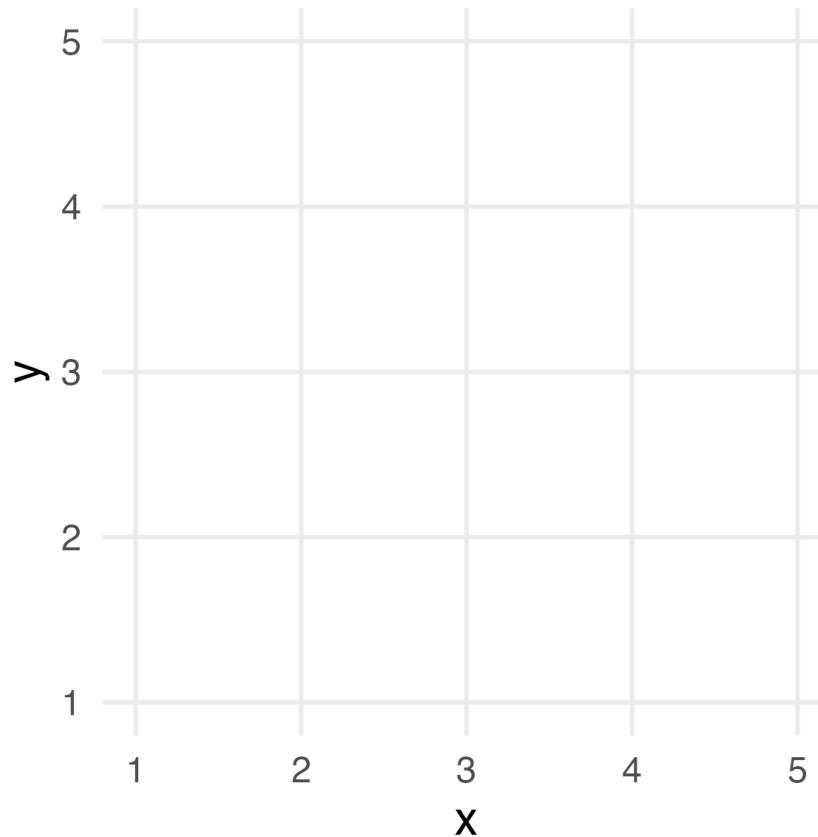
- 1 ggplot2
- 2 gghighlight
- 3 cowplot
- 4 ggttext
- 5 prismatic
- 6 patchwork

data

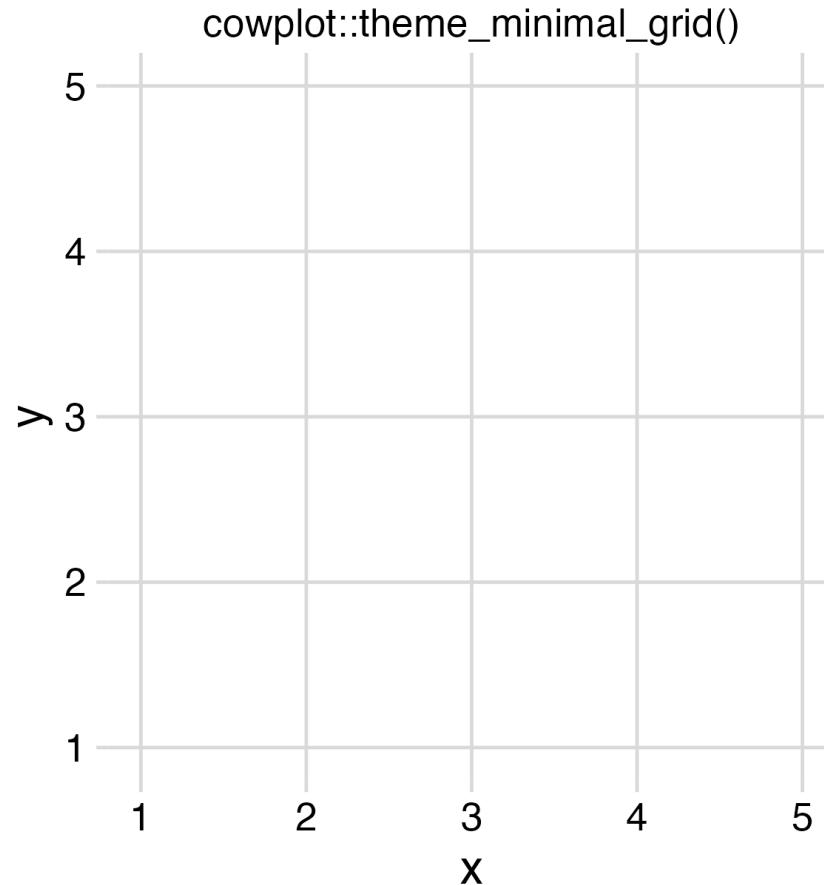
- 1 emperors (data/emperors.csv)
- 2 gapminder (library(gapminder))
- 3 nyc_squirrels (data/nyc_squirrels.csv + data/central_park/)
- 4 diabetes (data/diabetes.csv)
- 5 la_heat_income (data/los-angeles.geojson)

Themes

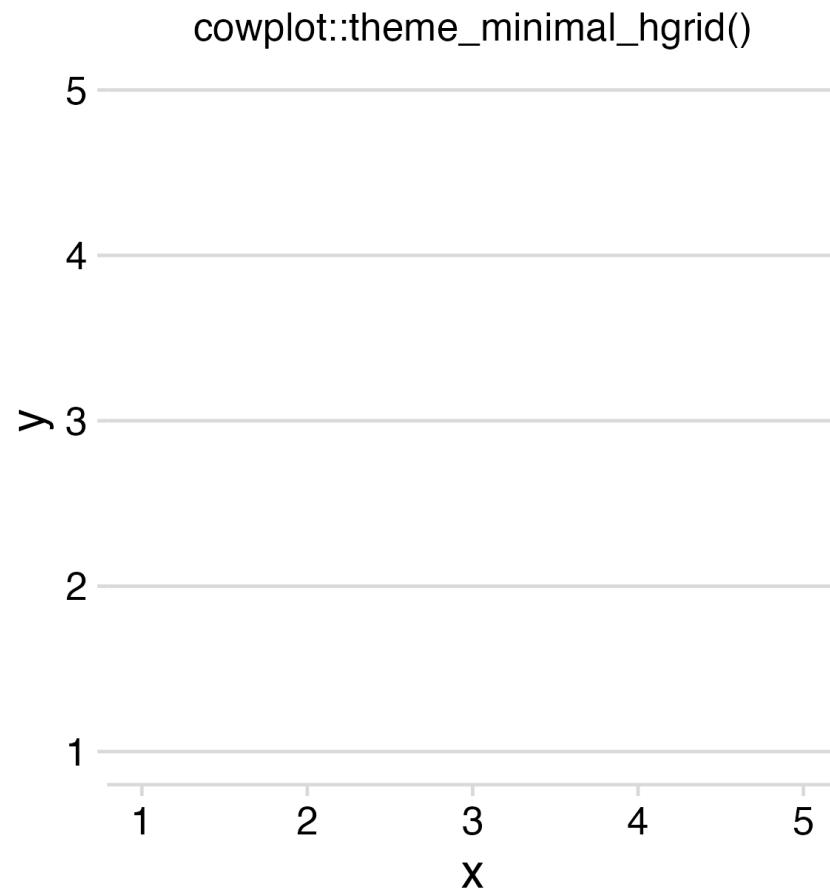
```
theme_minimal(14) +  
theme(panel.grid.minor = element_blank())
```



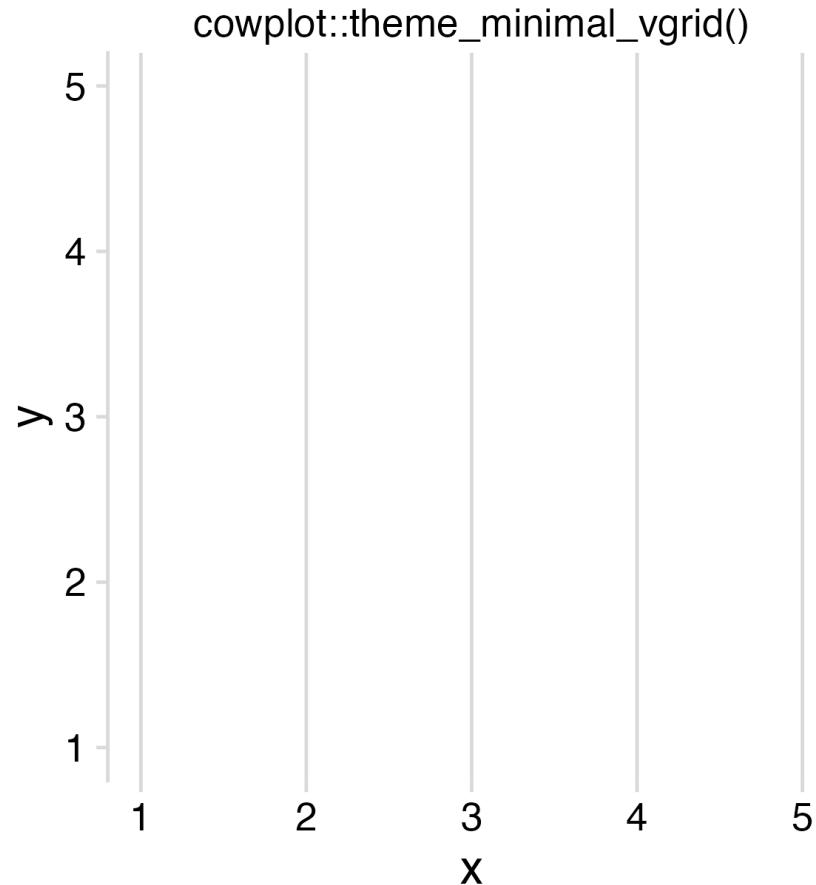
Themes: cowplot



Themes: cowplot



Themes: cowplot

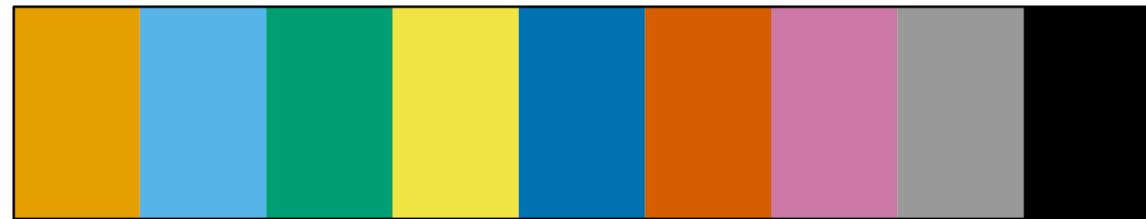


Themes: cowplot

`cowplot::theme_map()`



Palettes



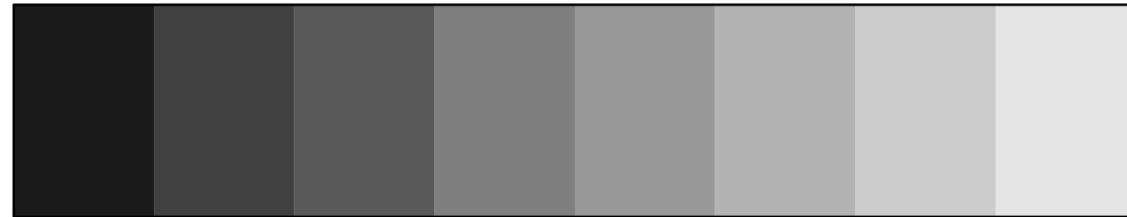
Okabe-Ito
(`ggokabeito::palette_okabe_ito()`)

Palettes



viridis inferno (viridis::inferno())

Palettes



greys ("grey**", grey(.**))

**act one: focus and declutter
or: reducing mental burden in figures**

How do we reduce mental burden in our plots?

How do we reduce mental burden in our plots?

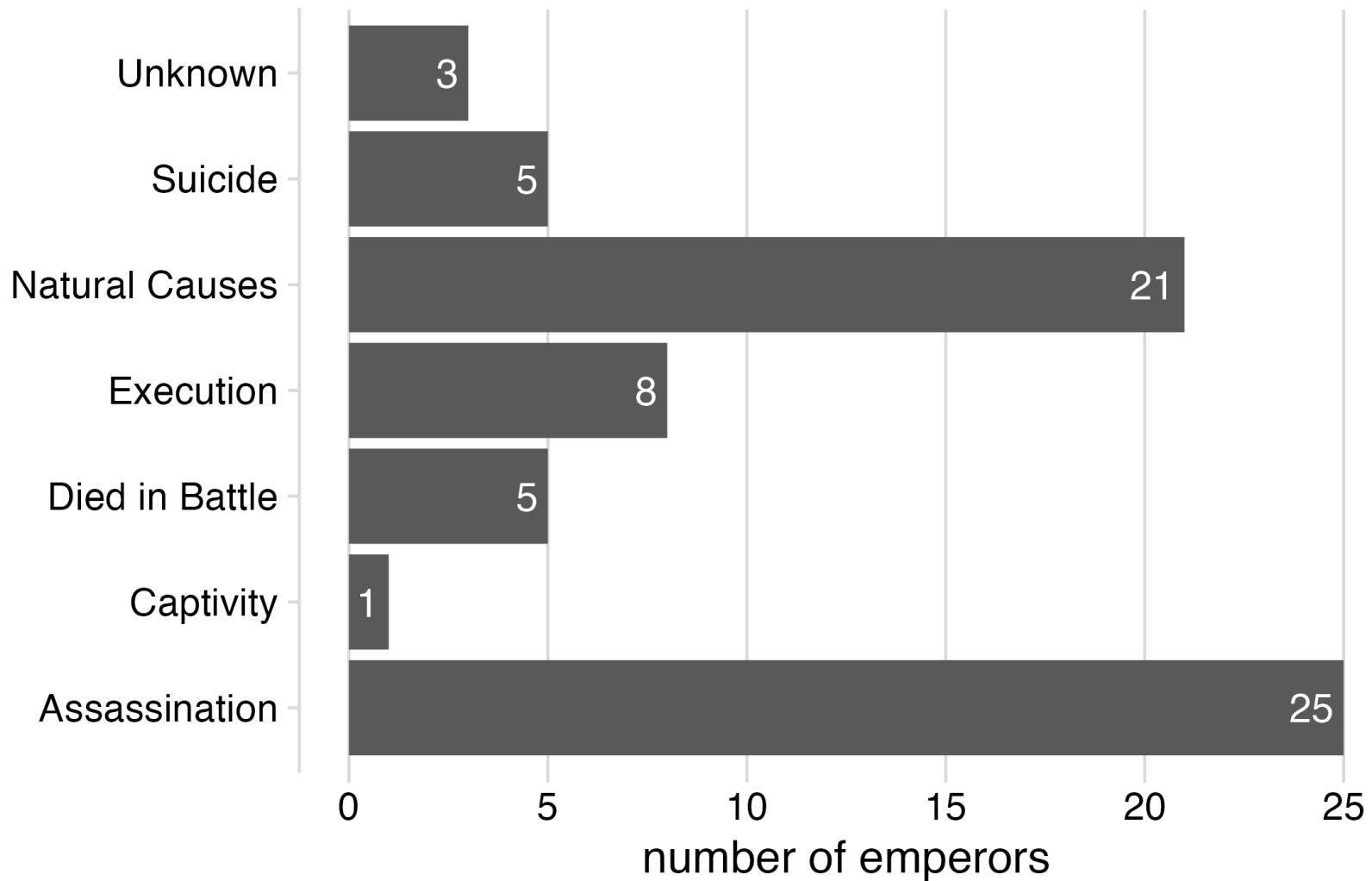
simplify aesthetics and highlight

```
emperors <- read_csv(file.path("data", "emperors.csv"))
```

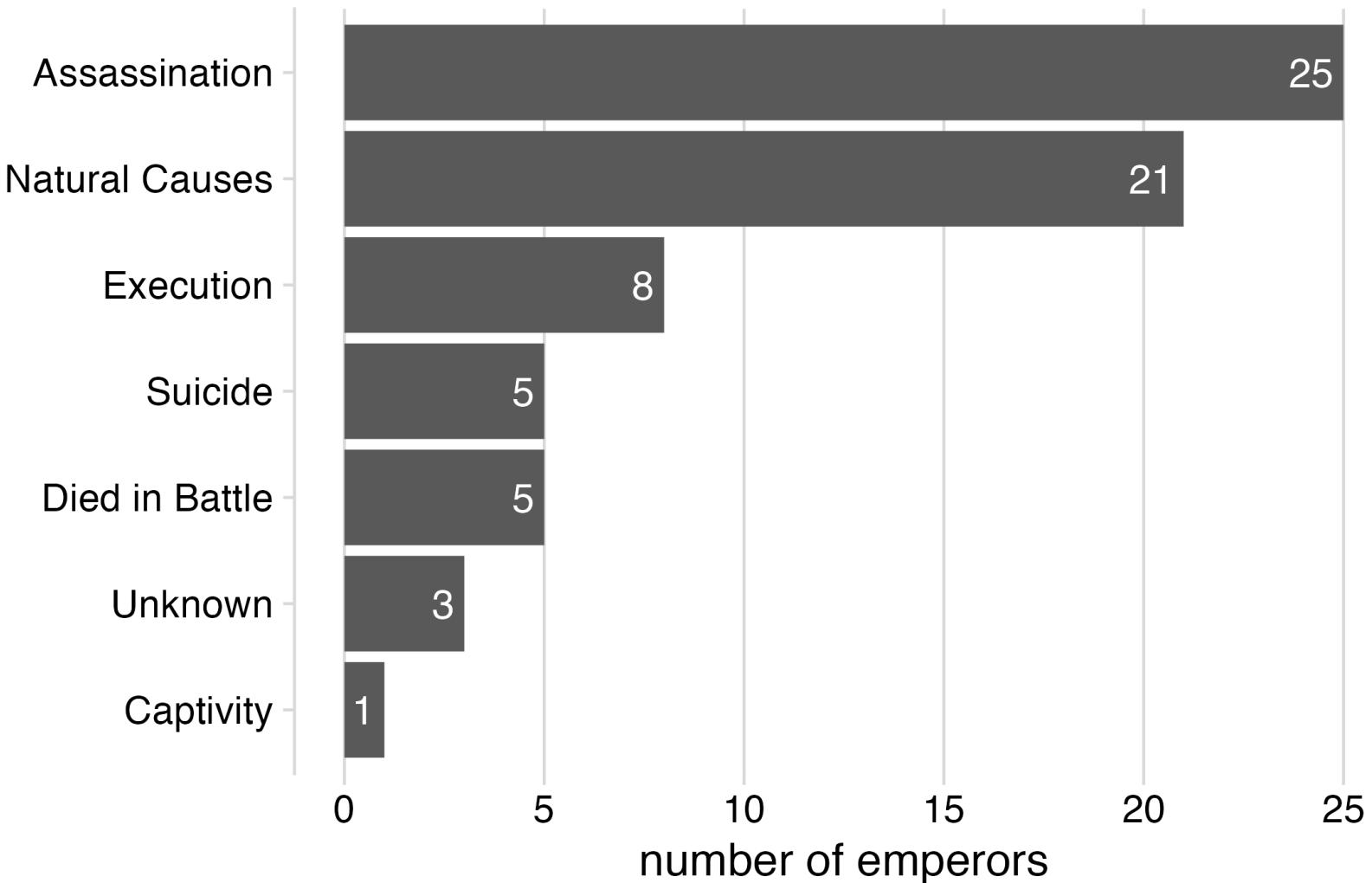
```
emperors
```

```
## # A tibble: 68 × 16
##   index name      name_full birth       death      birth_cty
##   <dbl> <chr>    <chr>     <date>     <date>    <chr>
## 1     1 Augustus IMPERATO... 0062-09-23 0014-08-19 Rome
## 2     2 Tiberius TIBERIVS... 0041-11-16 0037-03-16 Rome
## 3     3 Caligula GAIVS IV... 0012-08-31 0041-01-24 Antitum
## 4     4 Claudius TIBERIVS... 0009-08-01 0054-10-13 Lugdunum
## 5     5 Nero      NERO CLA... 0037-12-15 0068-06-09 Antitum
## 6     6 Galba     SERVIVS ... 0002-12-24 0069-01-15 Terracina
## 7     7 Otho      MARCVS S... 0032-04-28 0069-04-16 Terentin...
## 8     8 Vitellius AVLVS VI... 0015-09-24 0069-12-20 Rome
## 9     9 Vespasian TITVS FL... 0009-11-17 0079-06-24 Falacrine
## 10    10 Titus     TITVS FL... 0039-12-30 0081-09-13 Rome
## # ... with 58 more rows, and 10 more variables:
## #   birth_prv <chr>, rise <chr>, reign_start <date>,
## #   reign_end <date>, cause <chr>, killer <chr>, ...
```

```
emperors %>%
  count(cause) %>%
  ggplot(aes(x = n, y = cause)) +
  geom_col() +
  geom_text(
    aes(label = n, x = n - .25),
    color = "white",
    size = 5,
    hjust = 1
  ) +
  cowplot::theme_minimal_vgrid(16) +
  theme(
    axis.title.y = element_blank(),
    legend.position = "none"
  ) +
  xlab("number of emperors")
```

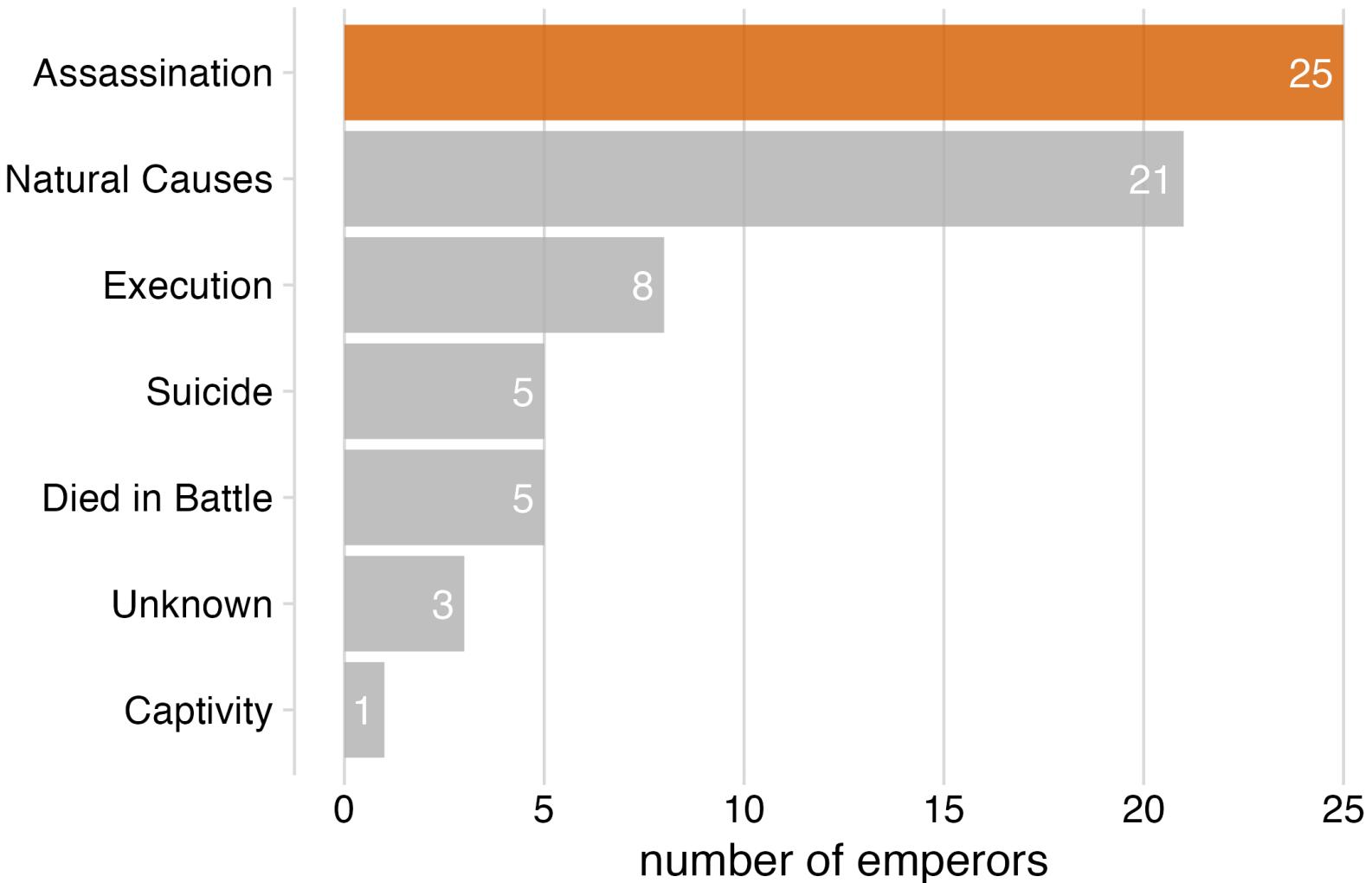


```
emperors %>%
  count(cause) %>%
  arrange(n) %>%
  mutate(cause = fct_inorder(cause)) %>%
  ggplot(aes(x = n, y = cause)) +
  geom_col() +
  geom_text(
    aes(label = n, x = n - .25),
    color = "white",
    size = 5,
    hjust = 1
  ) +
  cowplot::theme_minimal_vgrid(16) +
  theme(
    axis.title.y = element_blank(),
    legend.position = "none"
  ) +
  xlab("number of emperors")
```

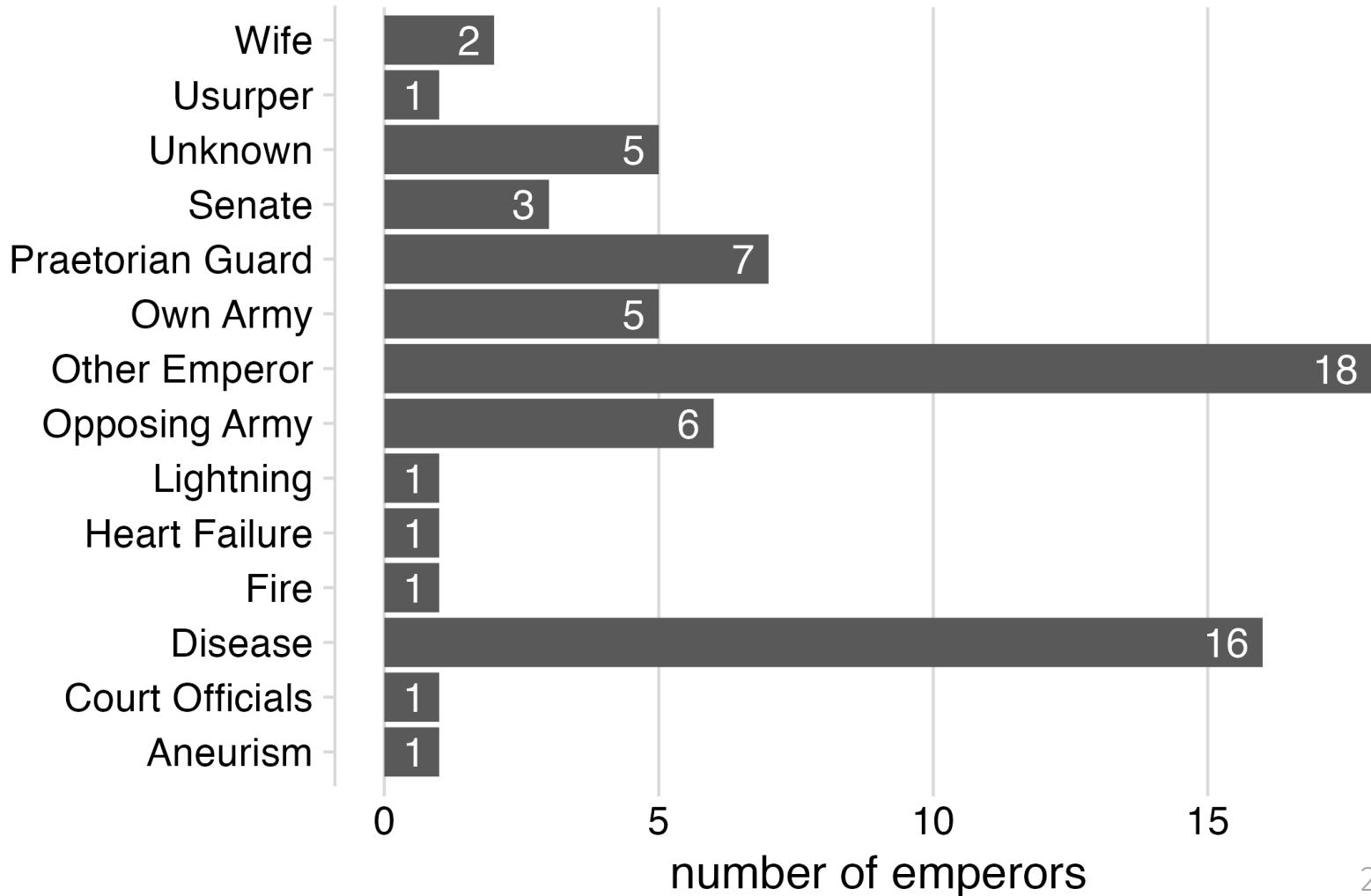


```
emperors_assassinated <- emperors %>%
  count(cause) %>%
  arrange(n) %>%
  mutate(
    assassinated = ifelse(cause == "Assassination", TRUE, FALSE),
    cause = fct_inorder(cause)
  )
```

```
emperors_assassinated %>%
  ggplot(aes(x = n, y = cause, fill = assassinated)) +
  geom_col() +
  geom_text(
    aes(label = n, x = n - .25),
    color = "white",
    size = 5,
    hjust = 1
  ) +
  cowplot::theme_minimal_vgrid(16) +
  theme(
    axis.title.y = element_blank(),
    legend.position = "none"
  ) +
  scale_fill_manual(
    name = NULL,
    values = c("#B0B0B0D0", "#D55E00D0")
  ) +
  xlab("number of emperors")
```



Your Turn 1



Your Turn 1

Read in the emperors data (no need to change this part of the code)

Sort the data using arrange() by the number of each type of killer

Take a look at the data up until this point. Pick something you find interesting that you want to highlight. Then, in mutate(), create a new variable that is TRUE if killer matches the category you want to highlight and FALSE otherwise

Use the variable you just created in the fill aesthetic of the ggplot call

Finally, use scale_fill_manual() to add the fill colors. Set values to c("#B0B0B0D0", "#D55E00D0").

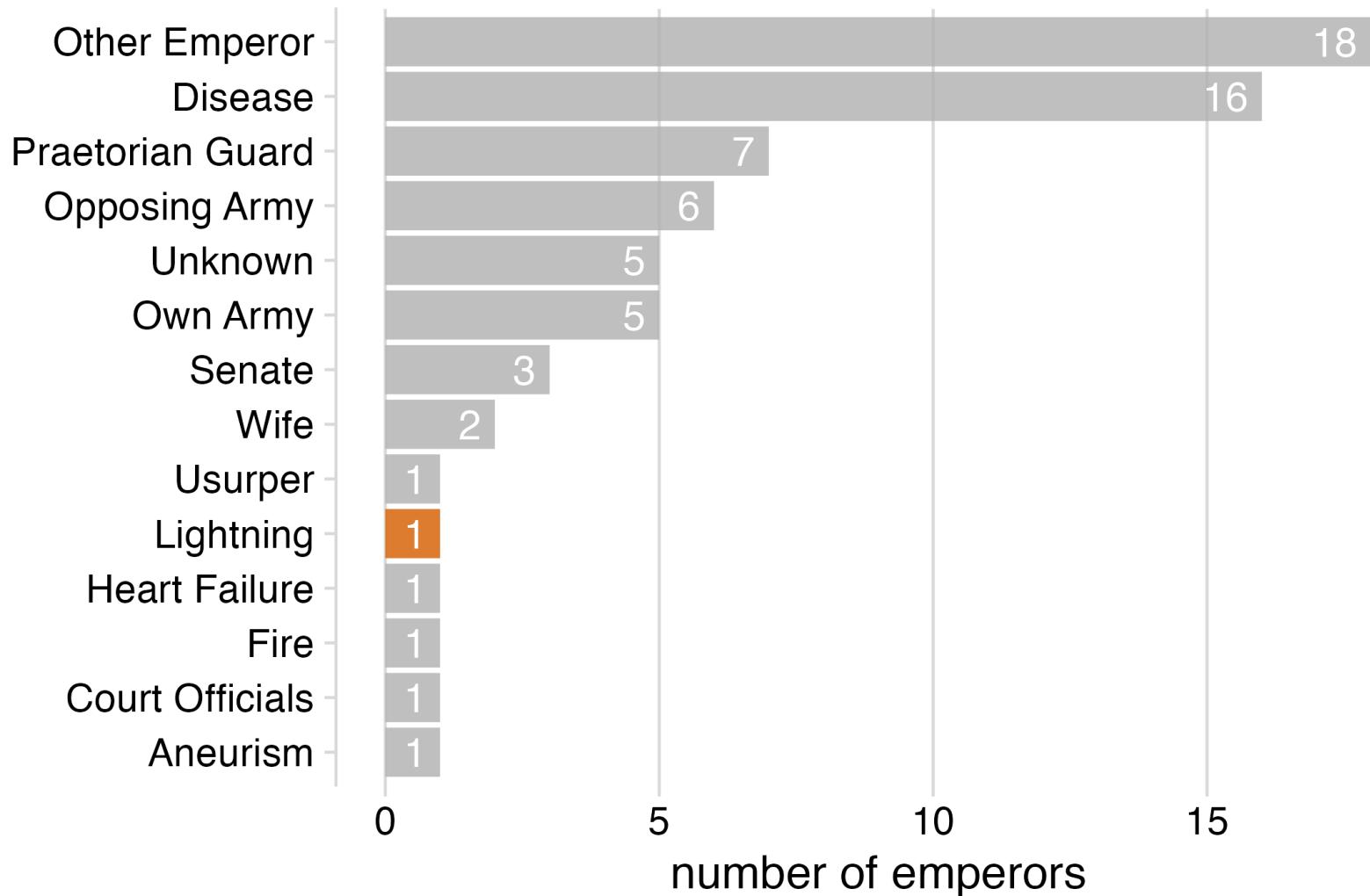
```
emperor_killers <- emperors %>%
  # group the least common killers to "other"
  mutate(killer = fct_lump(killer, 10)) %>%
  count(killer) %>%
  arrange(n)
```

emperor_killers

```
## # A tibble: 14 × 2
##   killer              n
##   <fct>            <int>
## 1 Aneurism          1
## 2 Court Officials  1
## 3 Fire              1
## 4 Heart Failure    1
## 5 Lightning         1
## 6 Usurper           1
## 7 Wife              2
## 8 Senate            3
## 9 Own Army          5
## 10 Unknown          5
## 11 Opposing Army    6
## 12 Praetorian Guard 7
## 13 Disease          16
## 14 Other Emperor    18
```

```
lightning_plot <- emperor_killers %>%
  mutate(
    lightning = ifelse(killer == "Lightning", TRUE, FALSE),
    # use `fct_inorder()` to maintain the way we sorted the data
    killer = fct_inorder(killer)
  ) %>%
  ggplot(aes(x = n, y = killer, fill = lightning)) +
  geom_col() +
  geom_text(
    aes(label = n, x = n - .25),
    color = "white",
    size = 5,
    hjust = 1
  ) +
  cowplot::theme_minimal_vgrid(16) +
  theme(
    axis.title.y = element_blank(),
    legend.position = "none"
  ) +
  scale_fill_manual(values = c("#B0B0B0D0", "#D55E00D0")) +
  xlab("number of emperors")

lightning_plot
```



Use color to focus attention

1 2 3 4 5 6 7 8 9

Use color to focus attention

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

ggtext: Improved text rendering in ggplot2

```
library(ggtext)
```

Use markdown in ggplot2 with
element_markdown()

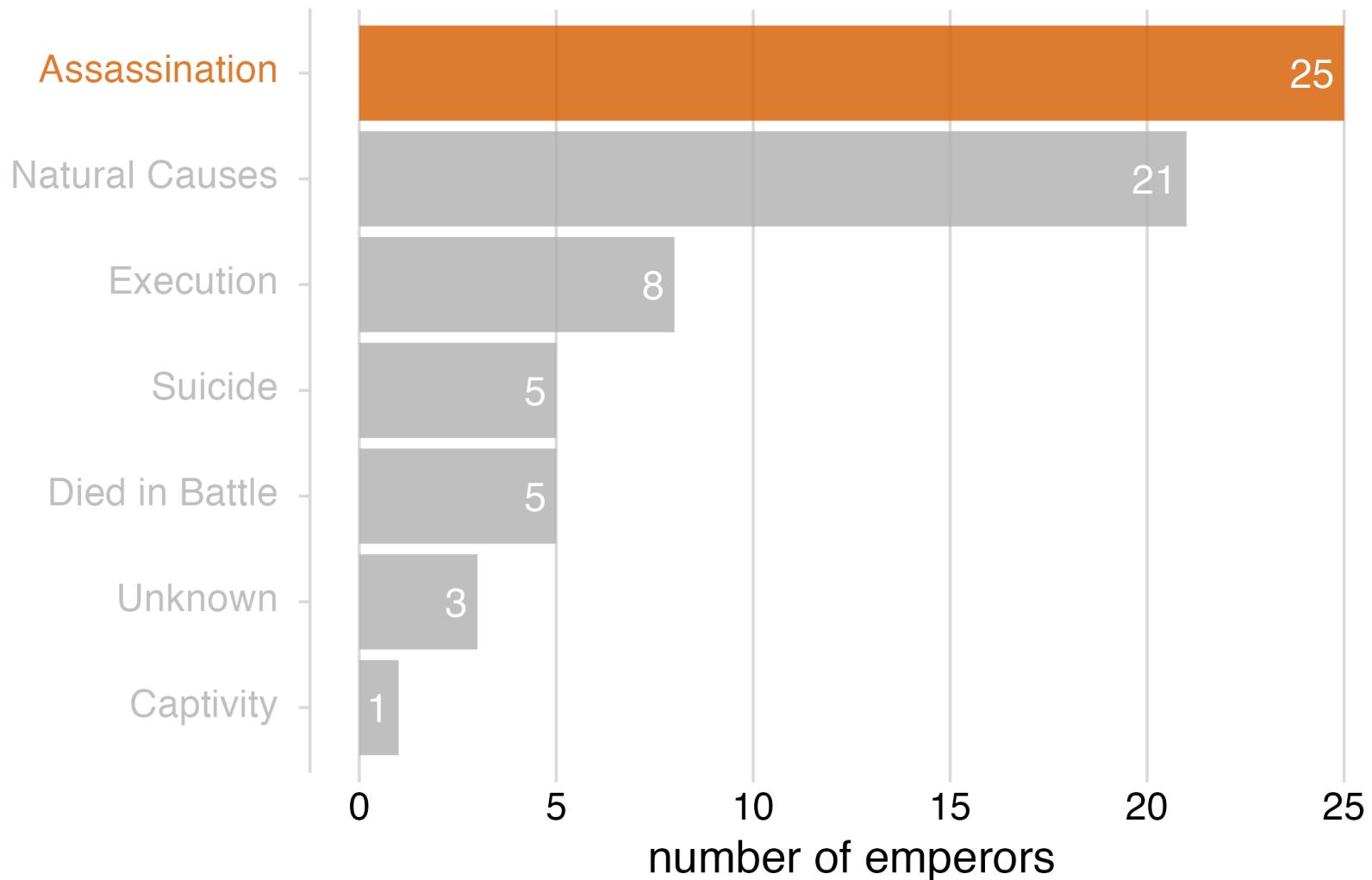
Allows styled text anywhere!

Can also insert images

```
library(glue)
library(ggtext)

emperors_assassinated <- emperors_assassinated %>%
  mutate(
    cause = glue(
      "<span style='color:{ifelse( \\"cause == 'Assassination', \\"#D5E0DD\", \\"#B0B0B0\")}'> \\"{cause}\\"</span>"
    ),
    cause = fct_inorder(cause)
  )
```

```
emperors_assassinated %>%
  ggplot(aes(x = n, y = cause, fill = assassinated)) +
  geom_col() +
  geom_text(
    aes(label = n, x = n - .25),
    color = "white",
    size = 5,
    hjust = 1
  ) +
  cowplot::theme_minimal_vgrid(16) +
  theme(
    axis.text.y = element_markdown(),
    axis.title.y = element_blank(),
    legend.position = "none",
  ) +
  scale_fill_manual(
    name = NULL,
    values = c("#B0B0B0D0", "#D55E00D0")
  ) +
  xlab("number of emperors")
```



How do we reduce mental burden in our plots?

simplify aesthetics and highlight
design figures without legends

```
library(gapminder)  
gapminder
```

```
## # A tibble: 1,704 × 6  
##   country    continent  year lifeExp      pop gdpPercap  
##   <fct>      <fct>     <int>   <dbl>    <int>     <dbl>  
## 1 Afghanistan Asia      1952     28.8  8425333    779.  
## 2 Afghanistan Asia      1957     30.3  9240934    821.  
## 3 Afghanistan Asia      1962     32.0  10267083   853.  
## 4 Afghanistan Asia      1967     34.0  11537966   836.  
## 5 Afghanistan Asia      1972     36.1  13079460   740.  
## 6 Afghanistan Asia      1977     38.4  14880372   786.  
## 7 Afghanistan Asia      1982     39.9  12881816   978.  
## 8 Afghanistan Asia      1987     40.8  13867957   852.  
## 9 Afghanistan Asia      1992     41.7  16317921   649.  
## 10 Afghanistan Asia     1997     41.8  22227415   635.  
## # ... with 1,694 more rows
```

```
gapminder %>%
  filter(year == 2007) %>%
  ggplot(aes(log(gdpPercap), lifeExp)) +
  geom_point(
    aes(color = country),
    size = 3.5,
    alpha = .9
  ) +
  theme_minimal(14) +
  theme(panel.grid.minor = element_blank()) +
  labs(
    x = "log(GDP per capita)",
    y = "life expectancy"
  )
```

- | | | | | |
|---------------|--------------------|------------------|------------|-------------|
| ican Republic | Dominican Republic | Honduras | Lebanon | Netherlands |
| . | Ecuador | Hong Kong, China | Lesotho | New Zealand |
| . | Egypt | Hungary | Liberia | Nicaragua |
| 1. Rep. | El Salvador | Iceland | Libya | Niger |
| . | Equatorial Guinea | India | Madagascar | Nigeria |
| . | Eritrea | Indonesia | Malawi | Norway |
| . | Ethiopia | Iran | Malaysia | Oman |
| . | Finland | Iraq | Mali | Pakistan |
| . | France | Ireland | Mauritania | Panama |
| . | Gabon | Israel | Mauritius | Paraguay |
| . | Gambia | Italy | Mexico | Peru |
| . | Germany | Jamaica | Mongolia | Philippines |
| . | Ghana | Japan | Montenegro | Poland |
| . | Greece | Jordan | Morocco | Portugal |
| iblic | Guatemala | Kenya | Mozambique | Puerto Rico |
| . | Guinea | Korea, Dem. Rep. | Myanmar | Reunion |
| . | Guinea-Bissau | Korea, Rep. | Namibia | Romania |
| . | Haiti | Kuwait | Nepal | Rwanda |

Direct labeling

- 1 Label data directly (maybe a subset)
- 2 Remove the legend
- 3 Use proximity and similarity (e.g. same color)

```
ggplot(<data>, <mappings>) +  
  <other geoms> +  
  geom_text(  
    data = <function or data>,  
    aes(label = <label>)  
  ) +  
  theme(legend.position = "none")
```

```
ggplot(<data>, <mappings>) +  
  <other geoms> + label with a text  
geom_text(← geom or ggrepel  
  data = <function or data>,  
  aes(label = <label>)  
  ) +  
  theme(legend.position = "none")
```

remove the legend ↗

option 1:
pre-process data

```
ggplot(<data>, <mappings>) +  
<other geoms> +  
geom_text(  
  data = <function or data>,  
  aes(label = <label>)  
) +  
theme(legend.position = "none")
```

option 2:
subset data

ggrepel: Repel overlapping text

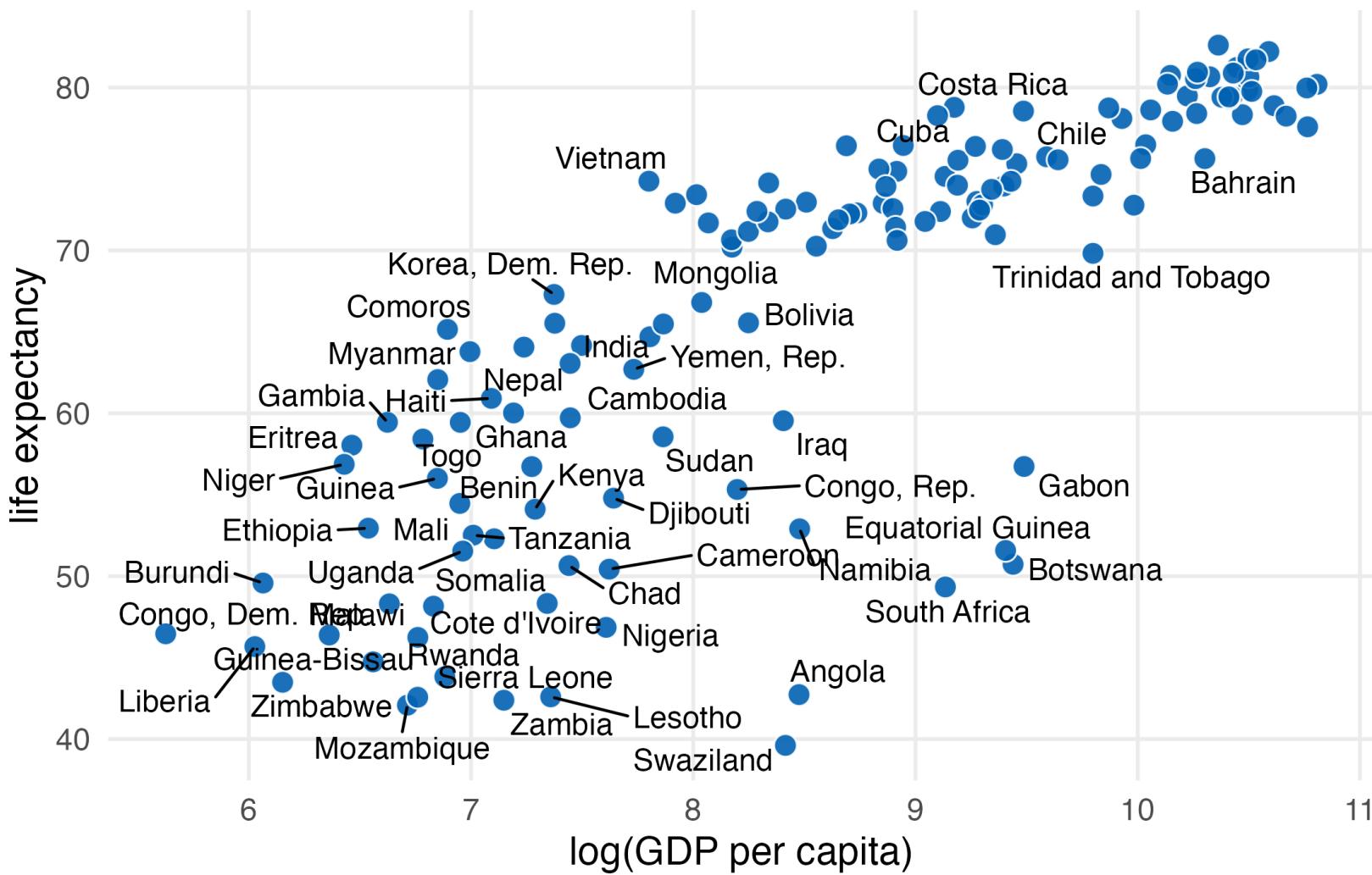
```
library(ggrepel)
```

geom_text_repel()

geom_label_repel()



```
gapminder %>%
  filter(year == 2007) %>%
  ggplot(aes(log(gdpPercap), lifeExp)) +
  geom_point(
    size = 3.5,
    alpha = .9,
    shape = 21,
    col = "white",
    fill = "#0162B2"
  ) +
  geom_text_repel(aes(label = country)) +
  theme_minimal(14) +
  theme(panel.grid.minor = element_blank()) +
  labs(
    x = "log(GDP per capita)",
    y = "life expectancy"
  )
```



Your Turn 2

**Use sample() to select 10 random countries to plot
(run the set.seed() line first if you want the same
results)**

**In the mutate() call, check if country is one of the
countries in ten_countries. If it's not, make the label
an empty string (""),**

**Add the text repel geom from ggrepel. Set the label
aesthetic using the variable just created in mutate()**

```
library(gapminder)
library(ggrepel)

set.seed(42)

ten_countries <- gapminder$country %>%
  levels() %>%
  sample(10)

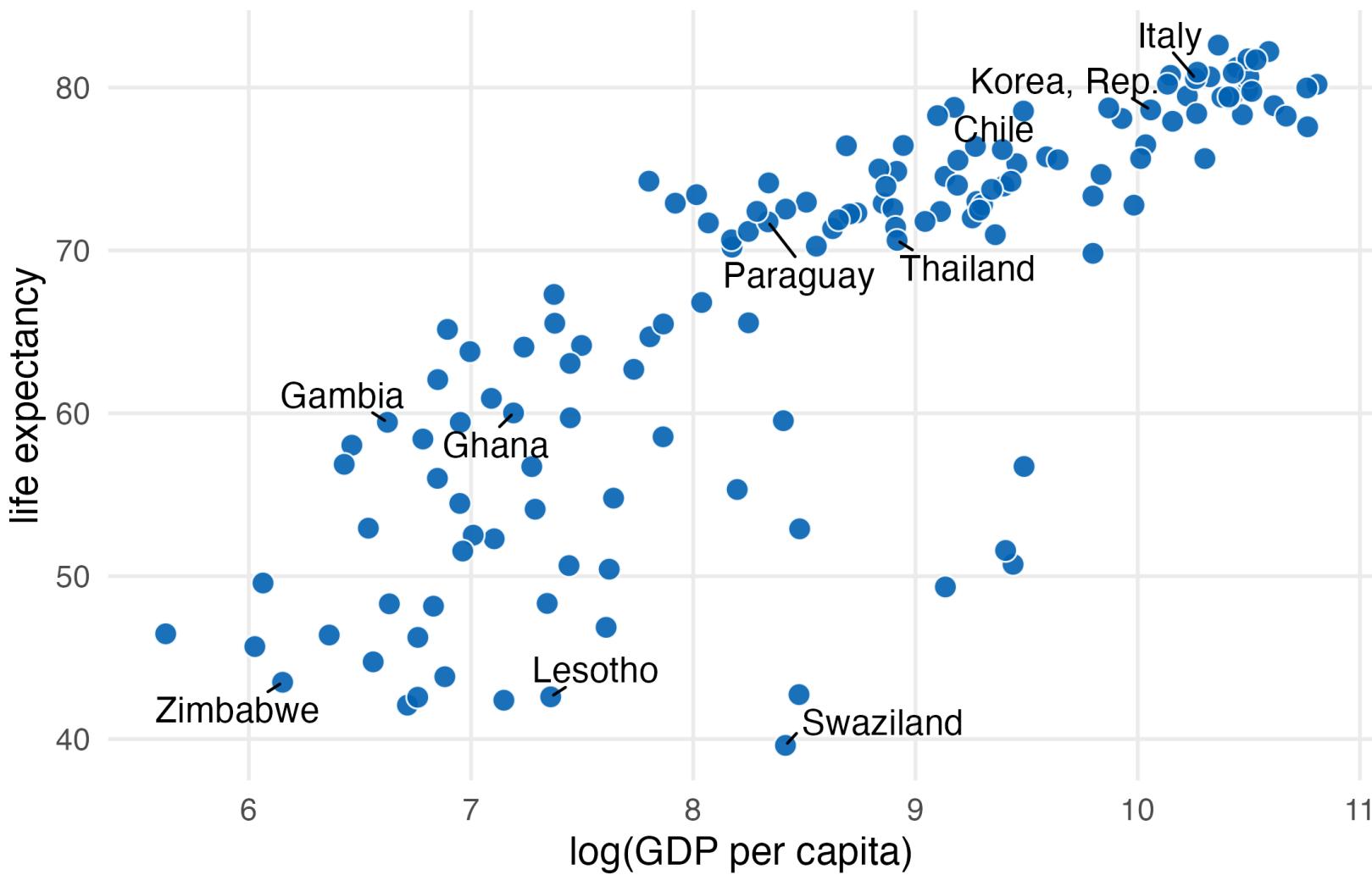
ten_countries

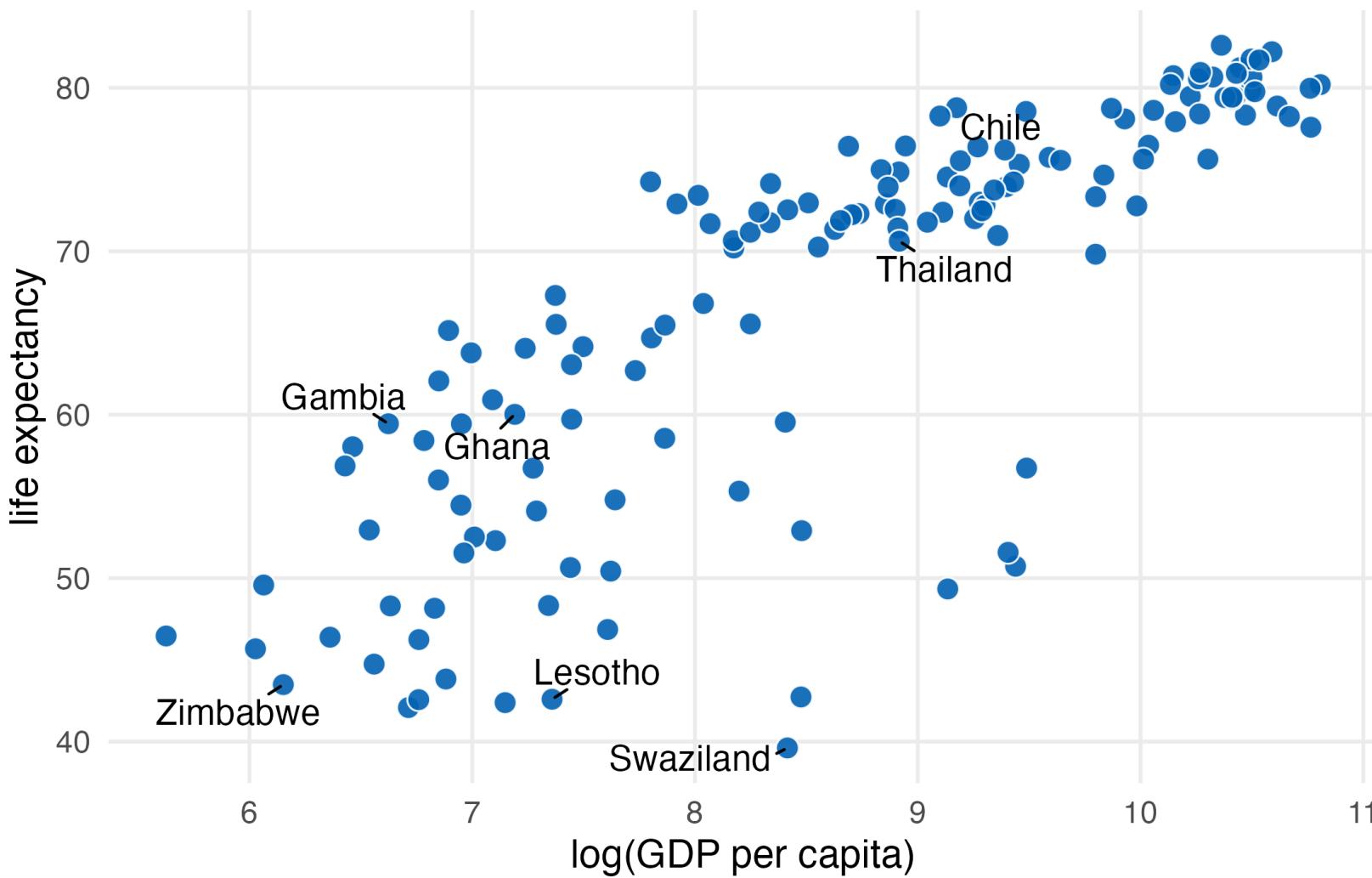
## [1] "Ghana"        "Italy"         "Lesotho"       "Swaziland"     "Zimbabwe"
## [9] "Korea, Rep."  "Paraguay"
```

```
p1 <- gapminder %>%
  filter(year == 2007) %>%
  mutate(
    label = ifelse(
      country %in% ten_countries,
      as.character(country),
      ""
    )
  ) %>%
  ggplot(aes(log(gdpPercap), lifeExp)) +
  geom_point(
    size = 3.5,
    alpha = .9,
    shape = 21,
    col = "white",
    fill = "#0162B2"
  )
```

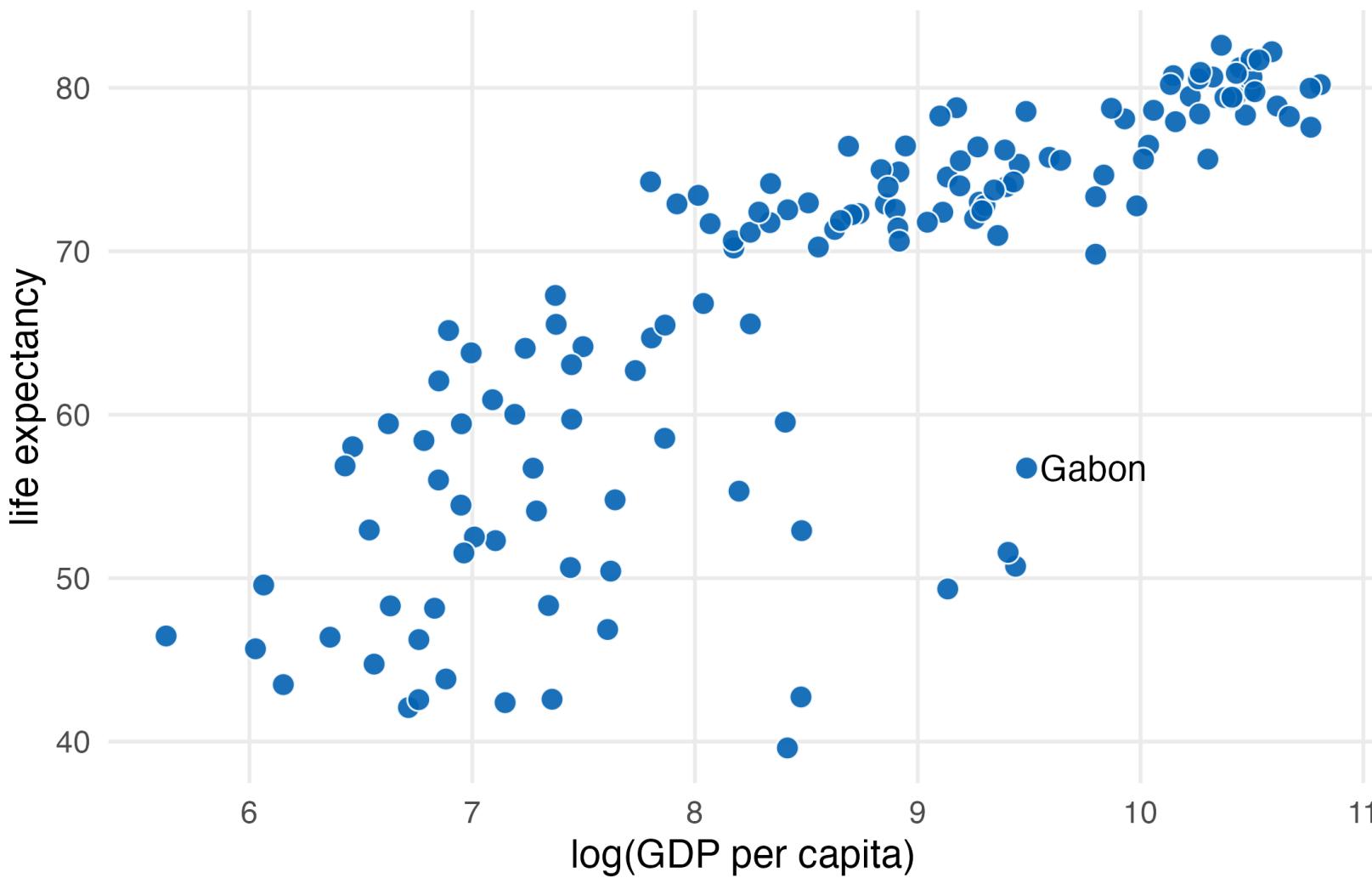
```
scatter_plot <- p1 +
  geom_text_repel(
    aes(label = label),
    size = 4.5,
    max.overlaps = Inf,
    box.padding = .3,
    force = 1,
    min.segment.length = 0
  ) +
  theme_minimal(14) +
  theme(
    legend.position = "none",
    panel.grid.minor = element_blank()
  ) +
  labs(
    x = "log(GDP per capita)",
    y = "life expectancy"
  )

scatter_plot
```





```
p1 +
  geom_text(
    data = function(x) filter(x, country == "Gabon"),
    aes(label = country),
    size = 4.5,
    hjust = 0,
    nudge_x = .06
  ) +
  theme_minimal(14) +
  theme(
    legend.position = "none",
    panel.grid.minor = element_blank()
  ) +
  labs(
    x = "log(GDP per capita)",
    y = "life expectancy"
  )
```



How do we reduce mental burden in our plots?

simplify aesthetics and highlight

design figures without legends

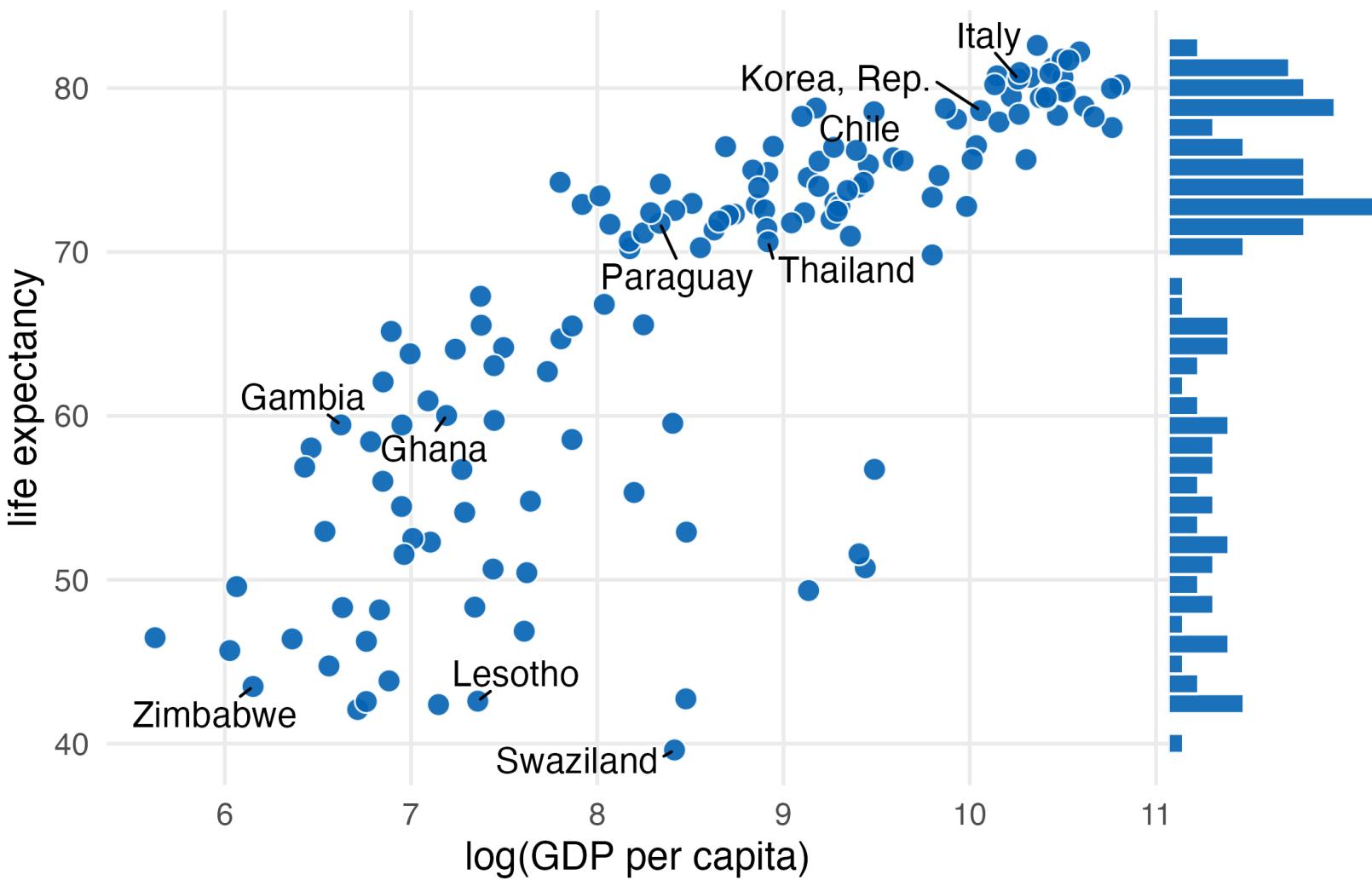
hack geoms and legends and even the
plot itself

Inserting plot objects into the axis

```
library(cowplot)
```

Inserting plot objects into the axis

```
library(cowplot)
marginal_histogram <- axis_canvas(scatter_plot, "y") +
  geom_histogram(
    data = gapminder %>% filter(year == 2007),
    bins = 40,
    aes(y = lifeExp),
    fill = "#0162B2E6",
    color = "white"
  )
scatter_plot %>%
  insert_yaxis_grob(marginal_histogram) %>%
  ggdraw()
```



```
ag <- axis_canvas(p, "y") +
<ggplot code>

p %>%
insert_yaxis_grob(ag) %>%
ggdraw()
```

get the axis of a plot

```
ag <- axis_canvas(p, "y") +
```

<ggplot code>

```
p %>%
```

```
  insert_yaxis_grob(ag) %>%
```

```
  ggdraw()
```

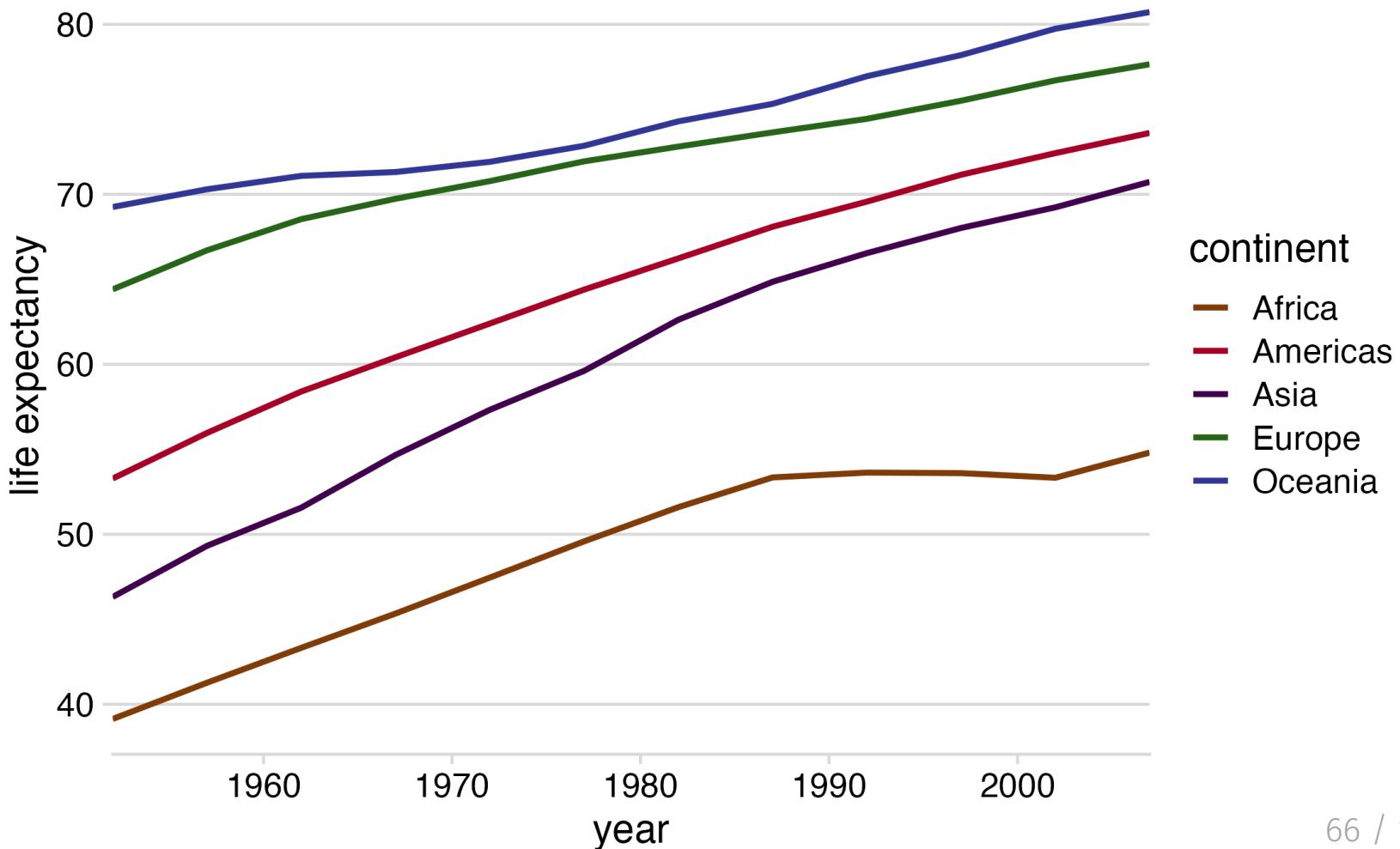
draw on it with
ggplot code

```
ag <- axis_canvas(p, "y") +  
<ggplot code>  
  put the axis drawing  
  into the axis of the  
p %>%      ↘ original plot  
insert_yaxis_grob(ag) %>%  
ggdraw()
```

```
ag <- axis_canvas(p, "y") +  
<ggplot code>  
  
p %>%  
  insert_yaxis_grob(ag) %>%  
  ggdraw()
```

draw the
custom plot

Your Turn 3



Your Turn 3

Calculate the placement of the labels: in the summarize() call, create a variable called y that is the maximum lifeExp value for every continent For the labels, we'll use the continent names, which will be retained automatically.

Remove the legend from the line plot. There are several ways to do so in ggplot2. I like setting legend.position = "none" in theme().

axis_canvas(line_plot, axis = "y") creates a new ggplot2 canvas based on the y axis from line_plot. Add a text geom (using + as you normally would). In the text geom: set data to direct_labels; in aes(), set y = y, label = continent; Outside of aes() set x to 0.05 (to add a little buffer); Make the size of the text 4.5; Set the horizontal justification to 0

Use insert_yaxis_grob() to take lineplot and insert direct_labels_axis.

Draw the new plot with ggdraw()

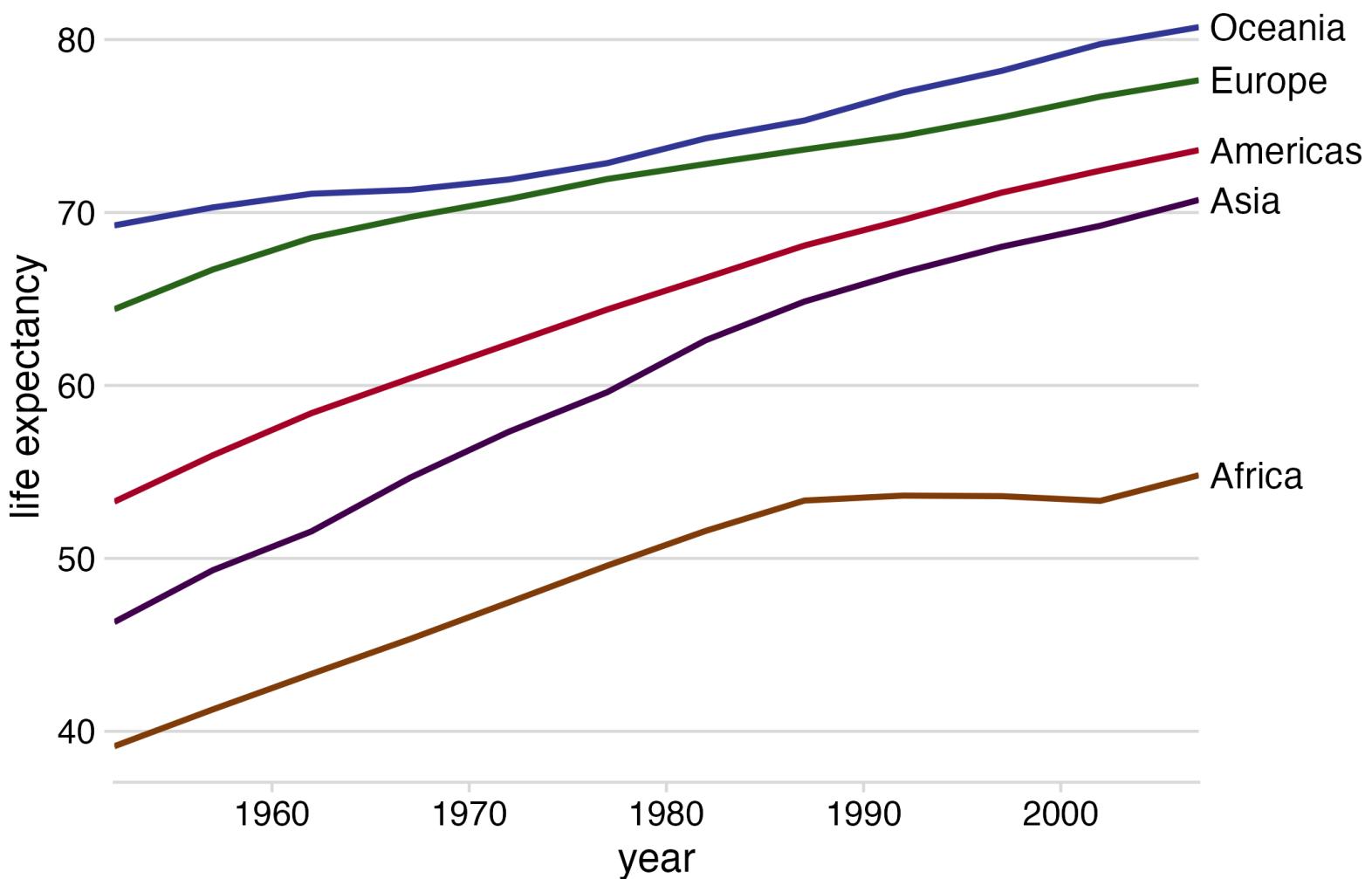
```
library(cowplot)

# get the mean life expectancy by continent and year
continent_data <- gapminder %>%
  group_by(continent, year) %>%
  summarise(lifeExp = mean(lifeExp))

direct_labels <- continent_data %>%
  group_by(continent) %>%
  summarize(y = max(lifeExp))
```

```
line_plot <- continent_data %>%
  ggplot(aes(year, lifeExp, col = continent)) +
  geom_line(size = 1) +
  theme_minimal_hgrid() +
  theme(legend.position = "none") +
  scale_color_manual(values = continent_colors) +
  scale_x_continuous(expand = expansion()) +
  labs(y = "life expectancy")
```

```
direct_labels_axis <- axis_canvas(line_plot, axis = "y") +  
  geom_text(  
    data = direct_labels,  
    aes(y = y, label = continent),  
    x = .05,  
    size = 4.5,  
    hjust = 0  
)  
  
p_direct_labels <- insert_yaxis_grob(line_plot, direct_labels_axis)  
  
ggdraw(p_direct_labels)
```



How do we reduce mental burden in our plots?

simplify aesthetics and highlight

design figures without legends

hack geoms and legends and even the plot itself

use facets and data shadows to plot overlapping data

Using facets to declutter data

- 1 Facets (or small multiples) are direct labeling for subsets
- 2 Put them into context with data shadows

```
nyc_squirrels <- read_csv(file.path("data", "nyc_squirrels.csv"))
central_park <- sf:::read_sf(file.path("data", "central_park"))
```

```
nyc_squirrels %>%
  drop_na(primary_fur_color) %>%
  ggplot() +
  geom_sf(data = central_park, color = "grey85") +
  geom_point(
    aes(x = long, y = lat, color = primary_fur_color),
    size = .8
  ) +
  cowplot::theme_map(16) +
  ggokabeito::scale_color_okabe_ito(name = "primary fur color")
```



primary fur color

- Black
- Cinnamon
- Gray

```
nyc_squirrels %>%
  drop_na(primary_fur_color) %>%
  ggplot() +
  geom_sf(data = central_park, color = "grey85") +
  geom_point(
    aes(x = long, y = lat, color = primary_fur_color),
    size = .8
  ) +
  facet_wrap(vars(primary_fur_color)) +
  cowplot::theme_map(16) +
  theme(legend.position = "none") +
  ggokabeito::scale_color_okabe_ito()
```

Black



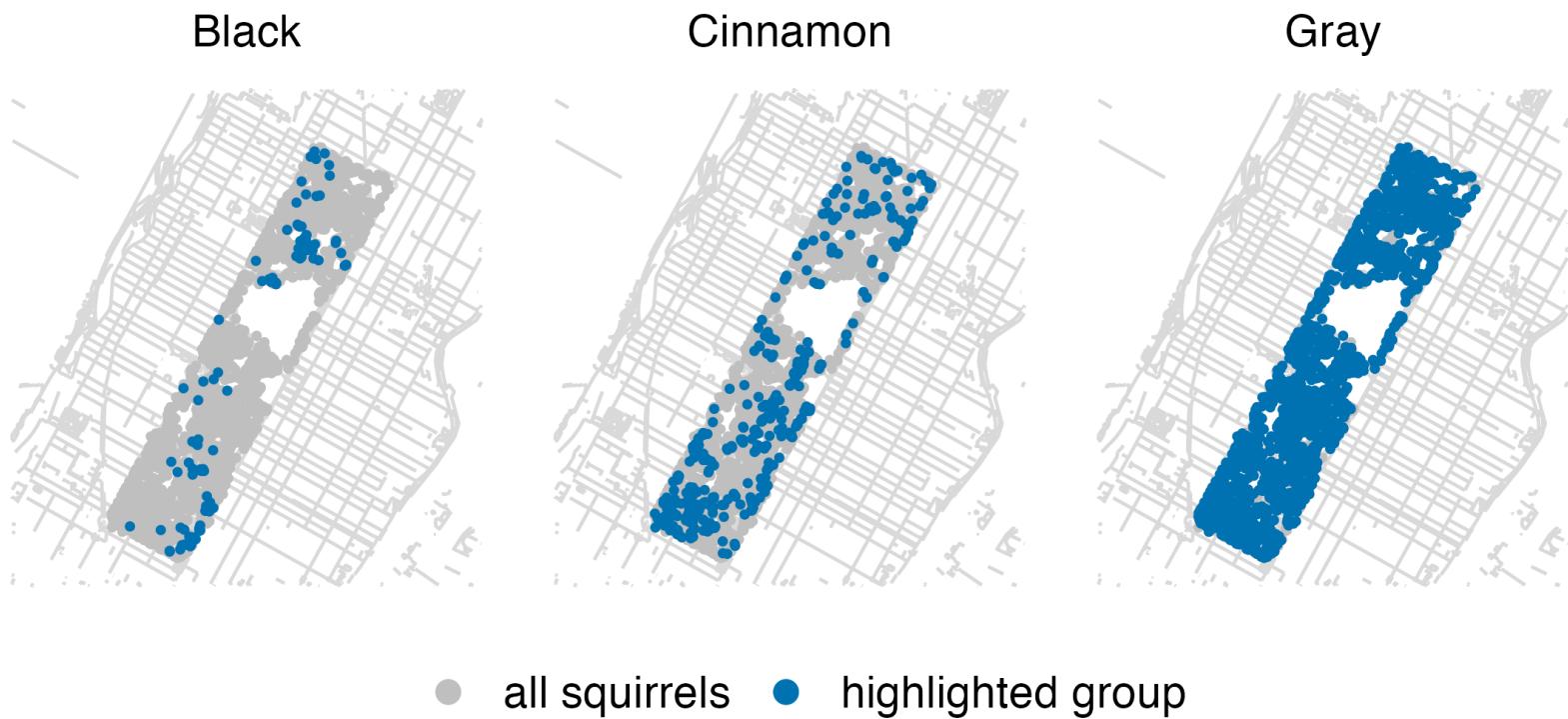
Cinnamon



Gray



```
label_colors <-  
  c("all squirrels" = "grey75", "highlighted group" = "#0072B2")  
  
nyc_squirrels %>%  
  drop_na(primary_fur_color) %>%  
  ggplot() +  
  geom_sf(data = central_park, color = "grey85") +  
  geom_point(  
    data = function(x) select(x, -primary_fur_color),  
    aes(x = long, y = lat, color = "all squirrels"),  
    size = .8  
  ) +  
  geom_point(  
    aes(x = long, y = lat, color = "highlighted group"),  
    size = .8  
  ) +  
  cowplot::theme_map(16) +  
  theme(  
    legend.position = "bottom",  
    legend.justification = "center"  
  ) +  
  facet_wrap(vars(primary_fur_color)) +  
  scale_color_manual(name = NULL, values = label_colors) +  
  guides(color = guide_legend(override.aes = list(size = 3)))
```



```
drop_facet <-
  function(x) select(x, -facet_var)

ggplot(data) +
  geom_point(
    data = drop_facet,
    aes(color = "all data")
  ) +
  geom_point(aes(color = facet_var)) +
  facet_wrap(vars(facet_var))
```

```
drop_facet <-
  function(x) select(x, -facet_var)

ggplot(data) +
  geom_point(
    data = drop_facet,
    aes(color = "all data")
  ) +
  geom_point(aes(color = facet_var)) +
  facet_wrap(vars(facet_var))


```

The diagram illustrates the structure of the R code. A grey bracket on the left groups the first two lines of code. An orange bracket on the right, labeled 'data shadow' in orange, groups the 'geom_point' call with its 'data' argument. A blue bracket on the right, labeled 'highlighted geom' in blue, groups the 'geom_point' call with its 'aes' argument.

```
drop_facet <-
  function(x) select(x, -facet_var)
ggplot(data) +
  geom_point(
    data = drop_facet,
    aes(color = "all data")
  ) +
  geom_point(aes(color = facet_var)) +
  facet_wrap(vars(facet_var))
```

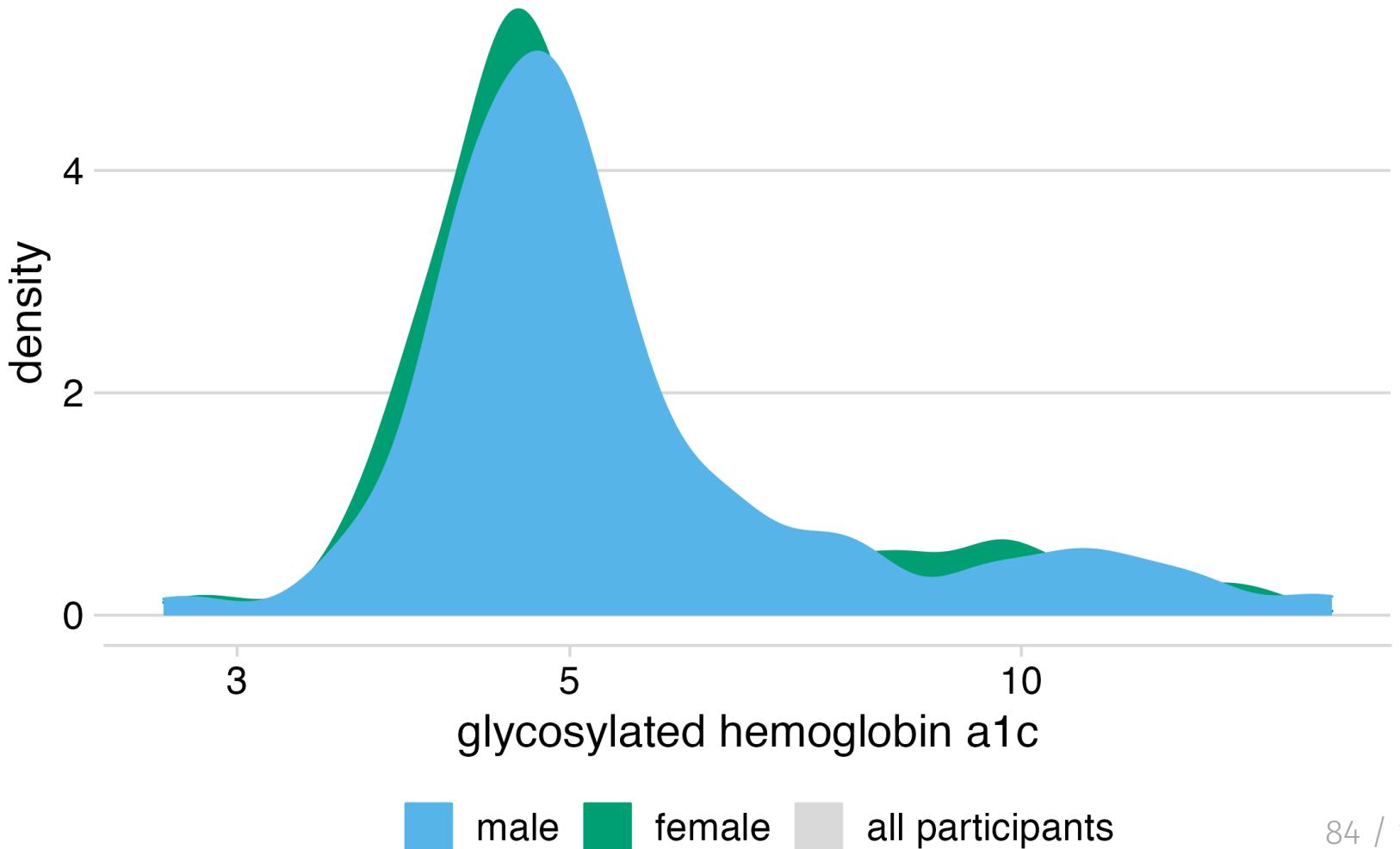
remove the facet variable from the data shadow layer

```
drop_facet <-
  function(x) select(x, -facet_var)

ggplot(data) +
  geom_point(
    data = drop_facet,
    aes(color = "all data"))
  +
  geom_point(aes(color = facet_var)) +
  facet_wrap(vars(facet_var))
```

create a legend
for the data
shadow

Your Turn 4



Your Turn 4

Run the code below and take a look at the resulting plot.

In the `ggplot()` function, add `y = after_stat(count)` to `aes()`

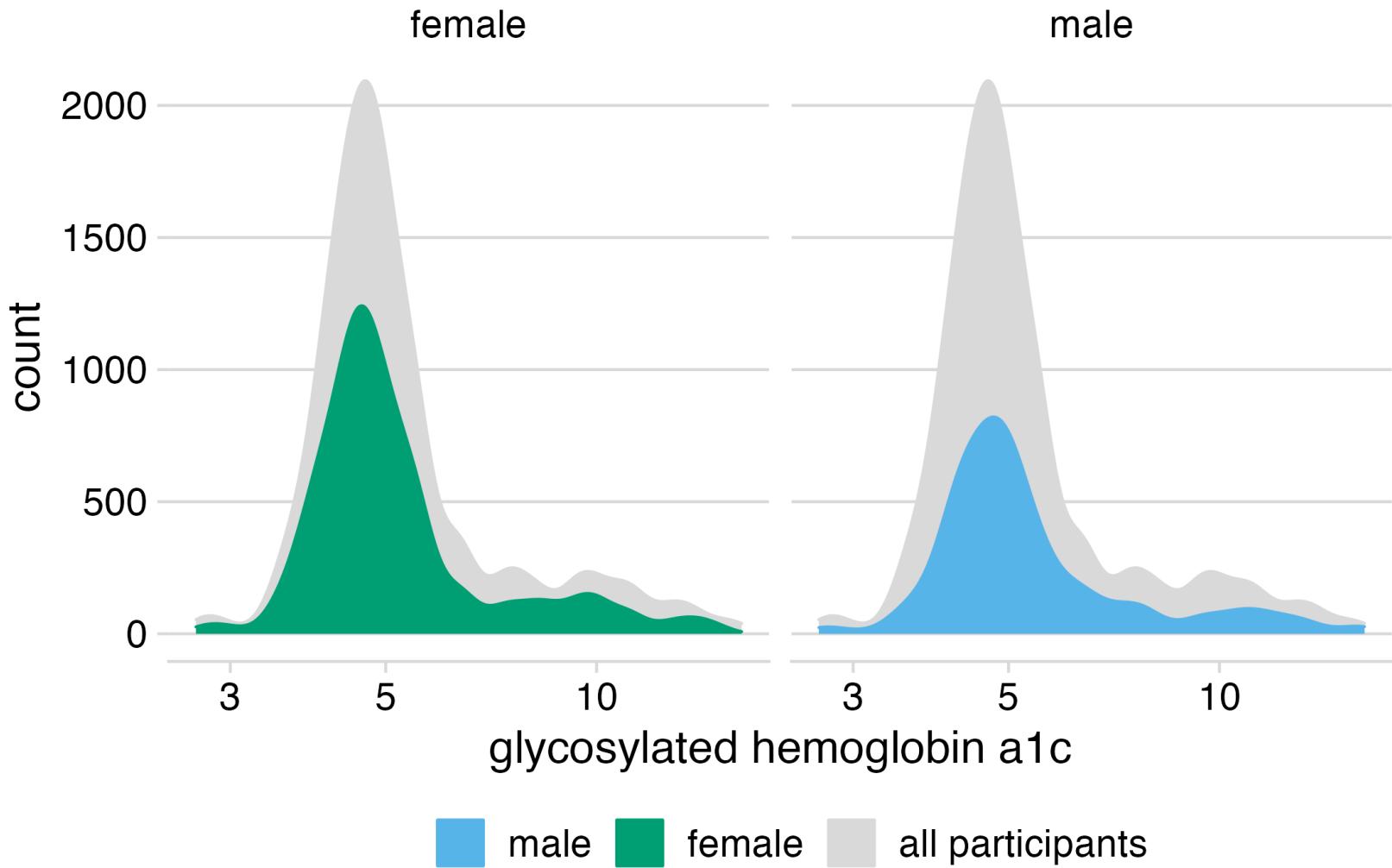
Add an additional `geom_density()` to the plot. This should go before the existing `geom_density()` so that it shows up in the background.

In the new `geom_density()`, set the data argument to be a function. This function should take a data frame and remove gender (which we're about to facet on).

Use `aes()` to set color and fill. Both should equal "all participants", not gender.

Use `facet_wrap()` to facet the plot by gender.

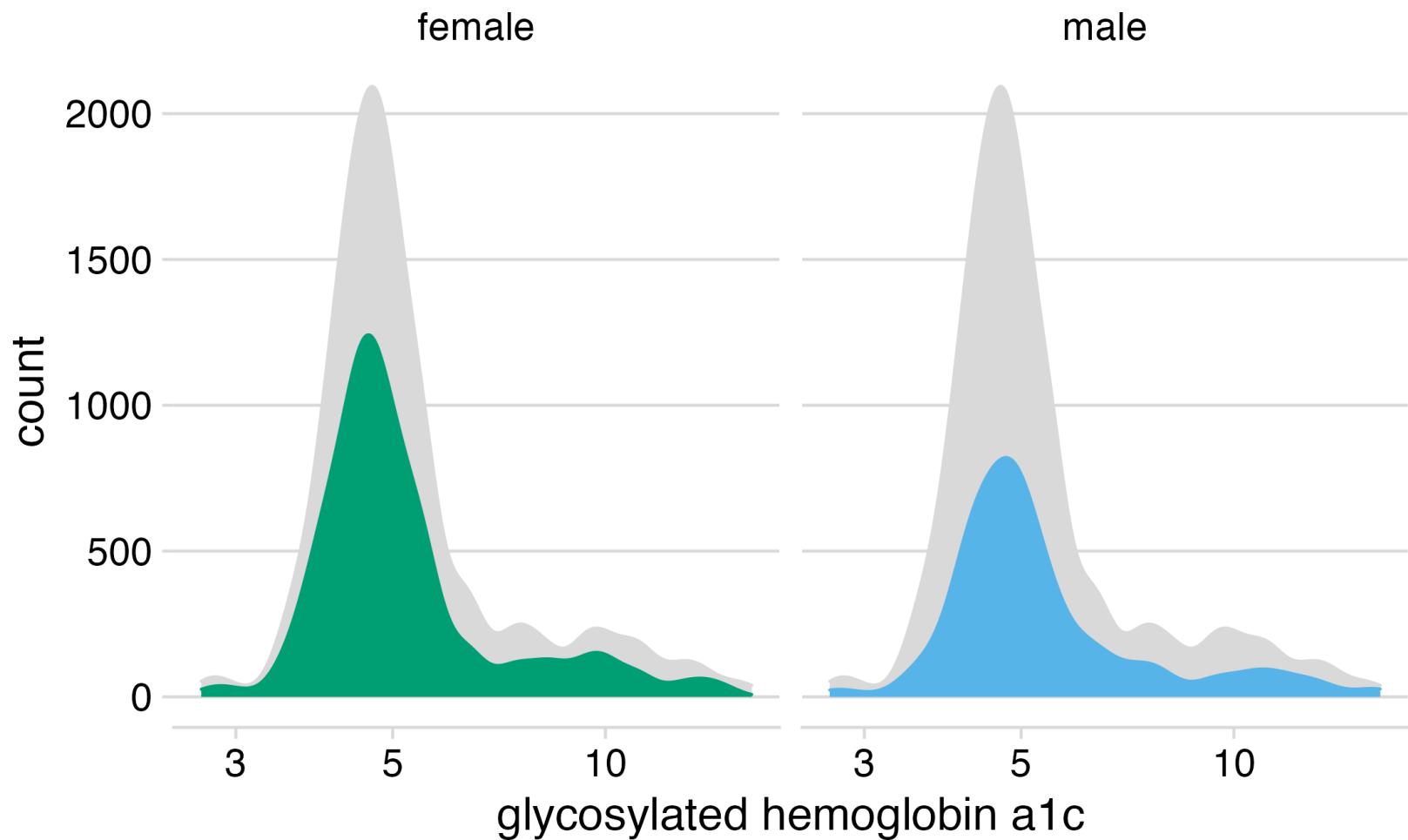
```
diabetes %>%
  drop_na(glyhb, gender) %>%
  ggplot(aes(glyhb, y = after_stat(count))) +
  geom_density(
    data = function(x) select(x, -gender),
    aes(fill = "all participants", color = "all participants")
  ) +
  geom_density(aes(fill = gender, color = gender)) +
  facet_wrap(vars(gender)) +
  scale_x_log10(name = "glycosylated hemoglobin a1c") +
  scale_color_manual(name = NULL, values = density_colors) +
  scale_fill_manual(name = NULL, values = density_colors) +
  theme_minimal_hgrid(16) +
  theme(legend.position = "bottom", legend.justification = "center")
```



```
title <- glue(
  "Among <b style='color:{density_colors[3]}>\\\
all participants</b>, there were more \\
<b style='color:{density_colors[2]}>females</b>\\\
than <b style='color:{density_colors[1]}>\\\
males</b>"
)

previous_plot +
  theme(
    legend.position = "none",
    plot.title.position = "plot",
    plot.title = element_textbox_simple(size = 14)
  ) +
  ggtitle(title)
```

Among all participants, there were more females than males



gghighlight: Highlight geoms

background-image:

url(<https://raw.githubusercontent.com/EmilHvitfeldt/prismatic/master/man/figures/logo.png>)

background-position: 90% 55%

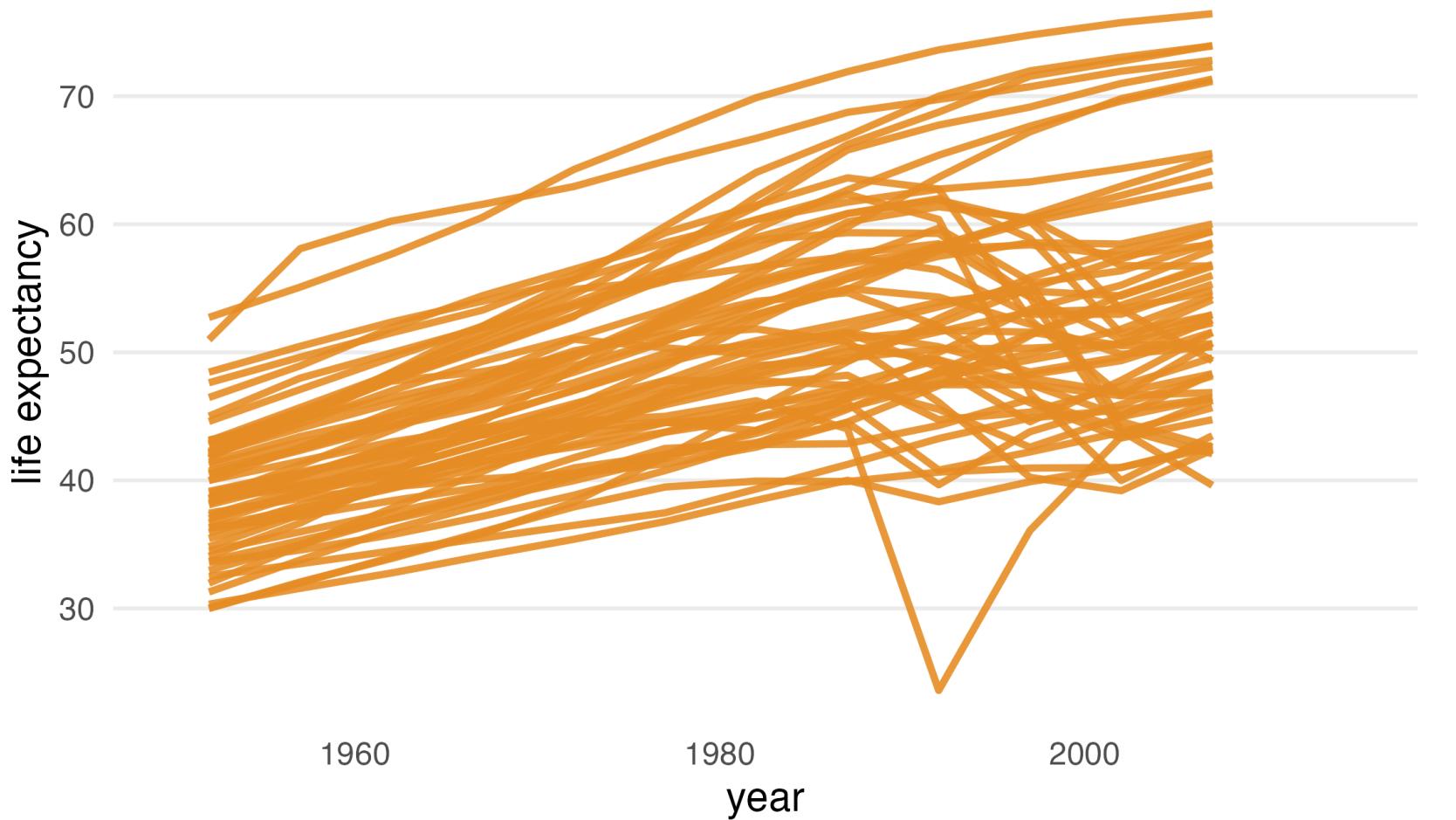
```
library(gghighlight)
```

gghighlight(predicate)

Works with points, lines, and histograms

Facets well

Your Turn 5



Your Turn 5

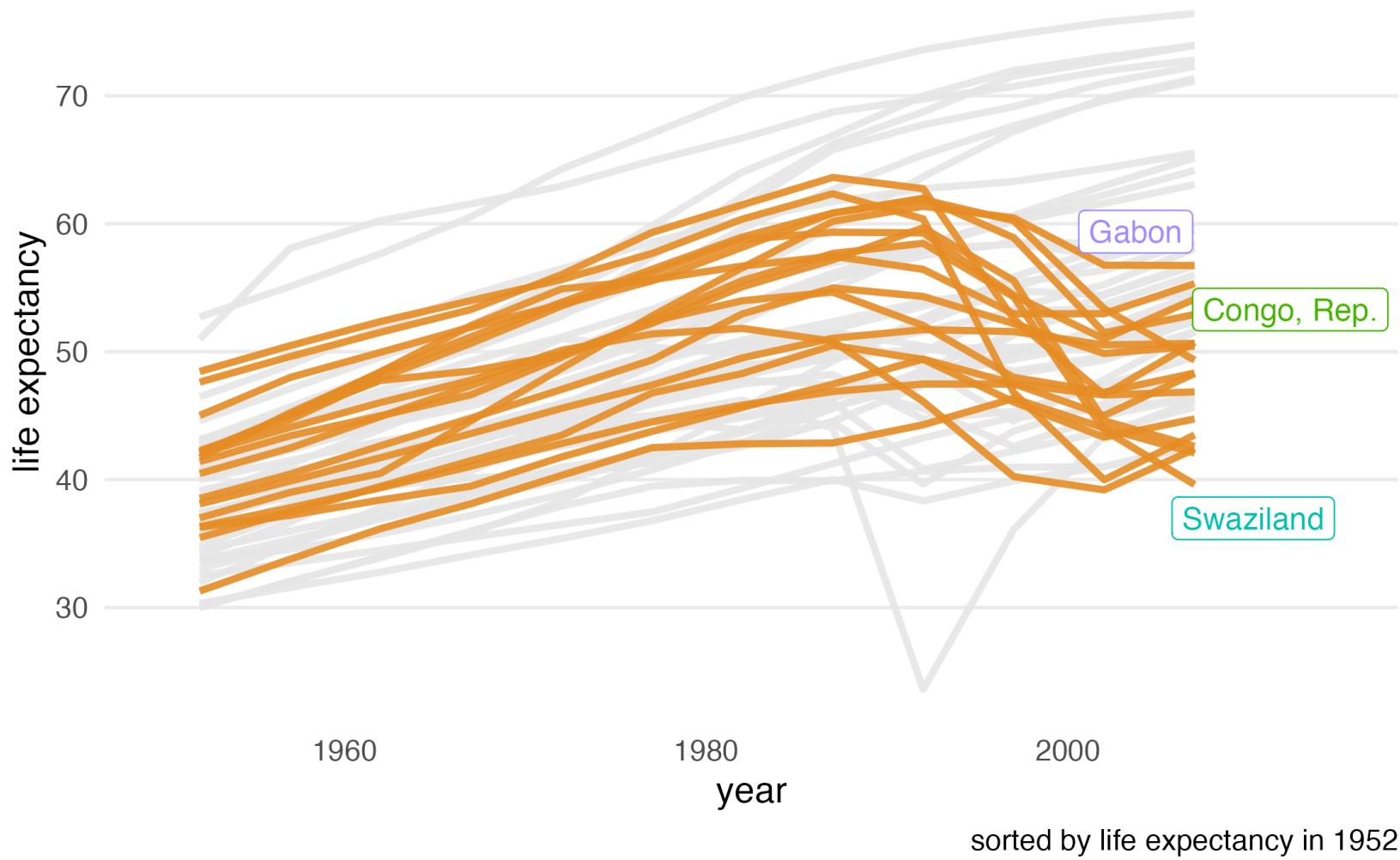
Take a look at the first few paragraphs of code. First, we're subsetting only African countries and sorting them by their life expectancy in 1952. Then, we're pivoting the data to be able to compare life expectancy in 1992 to 2007, creating a new variable, le_dropped, that is TRUE if life expectancy was higher in 1992. Then, we join le_dropped back to the data so we can use it in gghighlight(). Run the code at each step.

Remove the legend from the plot using the legend.position argument in theme(). Take a look at the base plot.

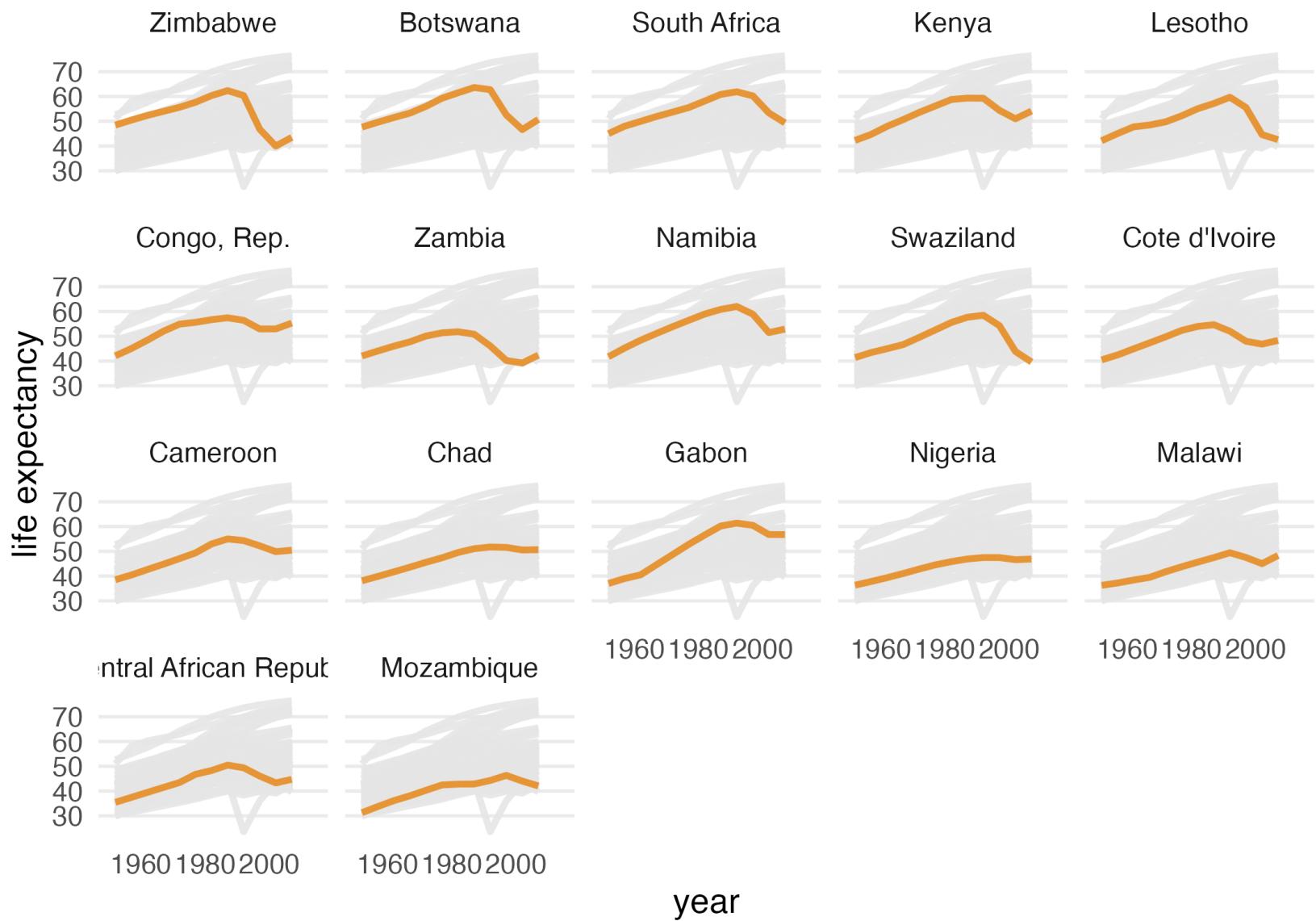
Use gghighlight() to add direct labels to the plot. For the first argument, tell it which lines to highlight using le_dropped. Also add the arguments use_group_by = FALSE and unhighlighted_params = list(color = "grey90").

Add use_direct_label = FALSE to gghighlight() and then facet the plot (using facet_wrap()) by country

```
le_line_plot +
  gghighlight(
    le_dropped,
    use_group_by = FALSE,
    unhighlighted_params = list(color = "grey90")
)
```



```
le_line_plot +
  gghighlight(
    le_dropped,
    use_group_by = FALSE,
    use_direct_label = FALSE,
    unhighlighted_params = list(color = "grey90")
  ) +
  facet_wrap(vars(country))
```



sorted by life expectancy in 1952

intermission: working with colors

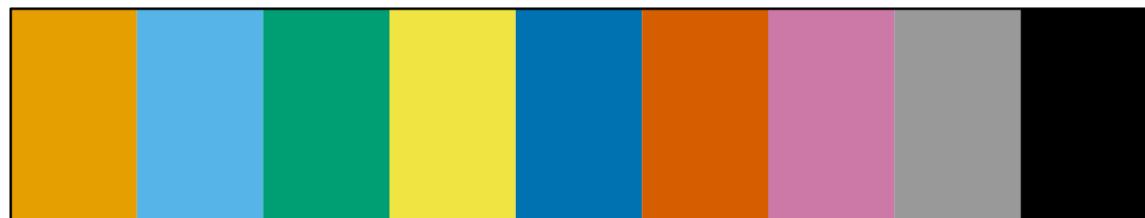
prismatic: manipulate colors in R

```
library(prismatic)
```



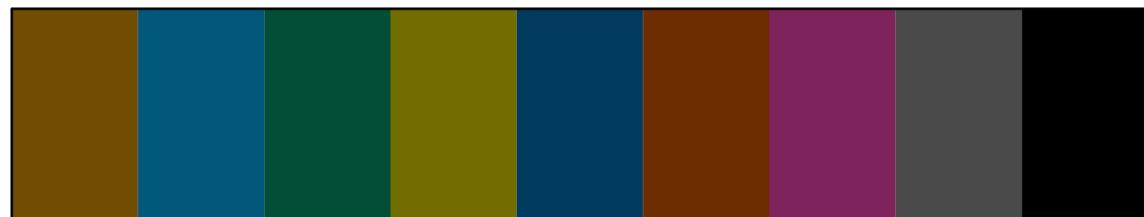
Visualize palettes

```
library(ggokabeito)
library(prismatic)
palette_okabe_ito() %>%
  color() %>%
  plot()
```



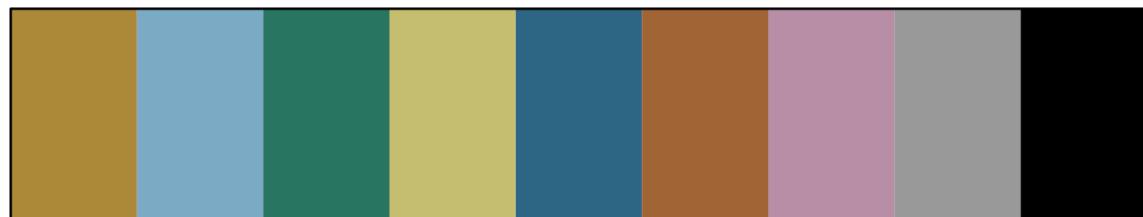
Darken and lighten colors

```
palette_okabe_ito() %>%  
  clr_darken() %>%  
  plot()
```



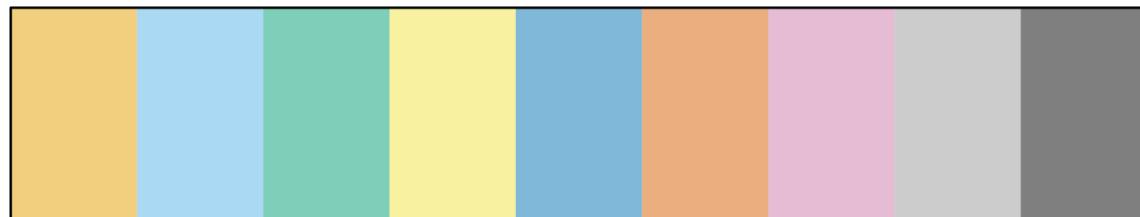
Saturate and desaturate colors

```
palette_okabe_ito() %>%  
  clr_desaturate() %>%  
  plot()
```



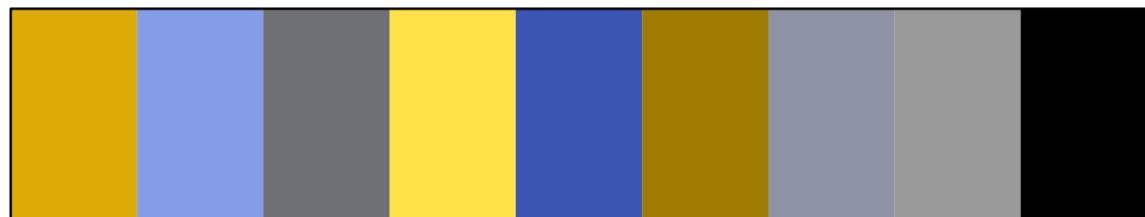
Set transparency

```
palette_okabe_ito() %>%  
  clr_alpha(.5) %>%  
  plot()
```



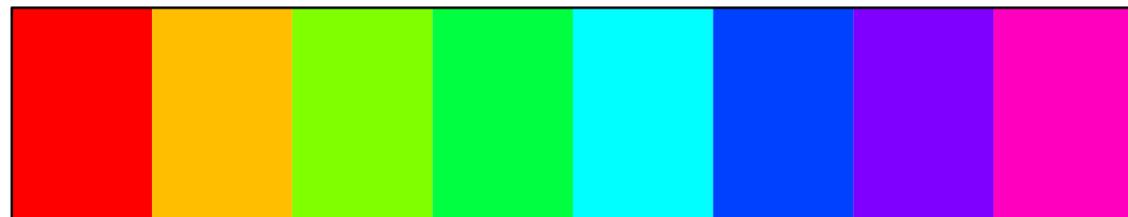
Simulate colorblindness

```
palette_okabe_ito() %>%  
  clr_deutan() %>%  
  plot()
```



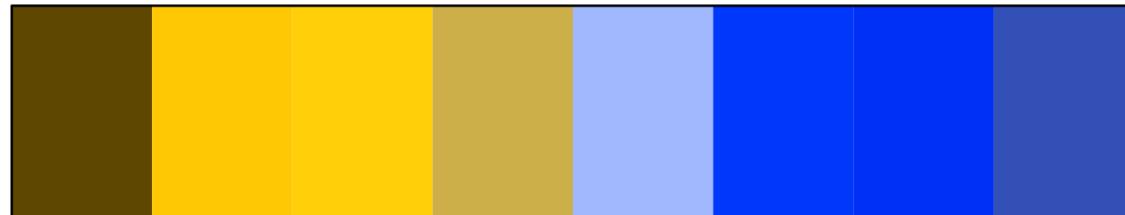
Simulate colorblindness

```
rainbow(8) %>%  
  color() %>%  
  plot()
```



Simulate colorblindness

```
rainbow(8) %>%  
  clr_deutan() %>%  
  plot()
```



act two: narrate and put in context
or: storytelling with data visualization

How do we augment plots to explain?

How do we augment plots to explain?

Annotate plots using text geoms and arrows

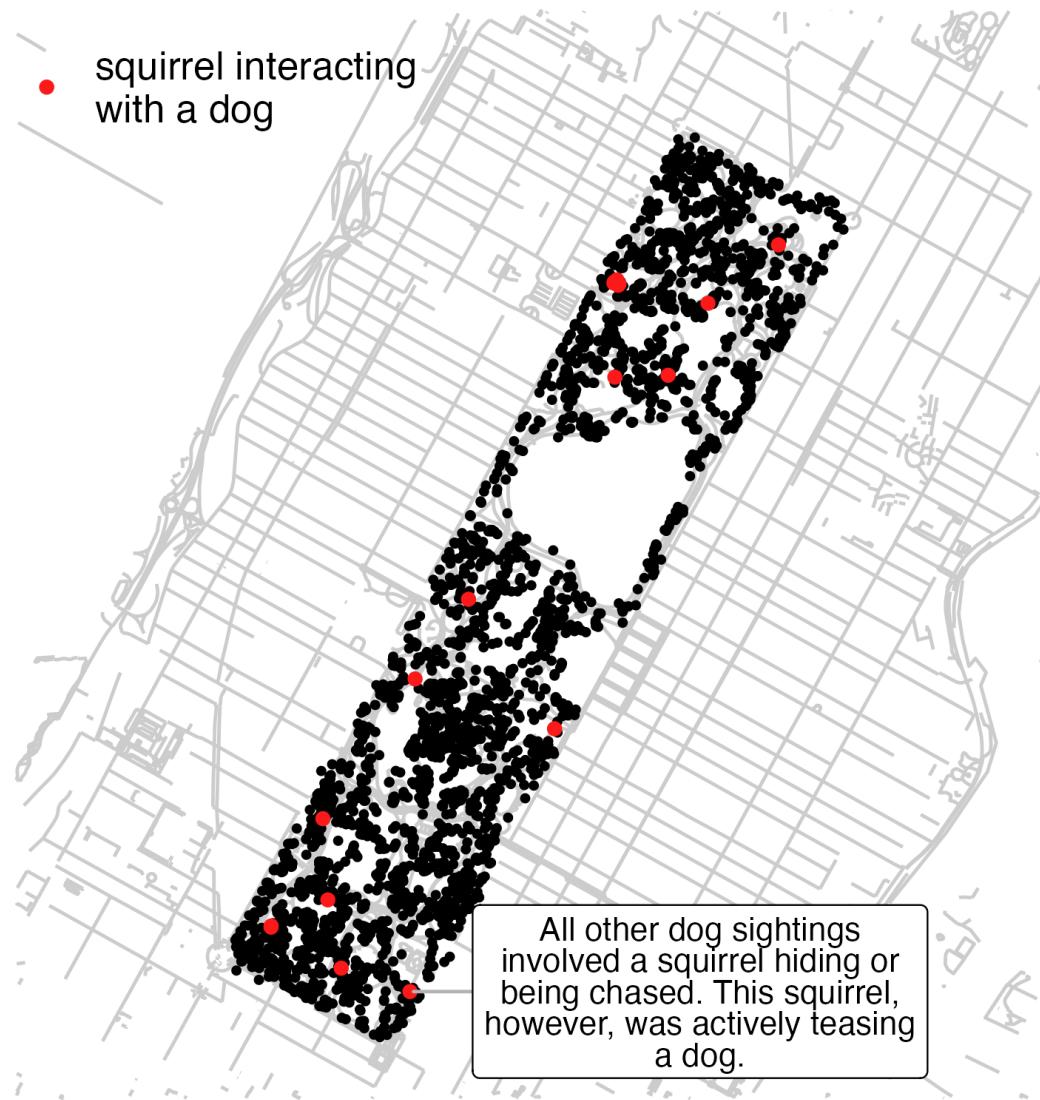
Squirrels and dogs

```
dog_sighting <- nyc_squirrels %>%
  mutate(dog = str_detect(other_activities, "dog")) %>%
  filter(dog)
```

```
lbl <- "All other dog sightings  
involved a squirrel hiding or  
being chased. This squirrel,  
however, was actively teasing  
a dog."  
  
dog_plot <- nyc_squirrels %>%  
  ggplot() +  
  geom_sf(data = central_park, color = "grey80") +  
  geom_point(aes(x = long, y = lat), size = .8) +  
  geom_point(  
    data = dog_sighting,  
    aes(  
      x = long,  
      y = lat,  
      color = "squirrel interacting\nwith a dog"  
    ),  
    size = 1.5  
)
```

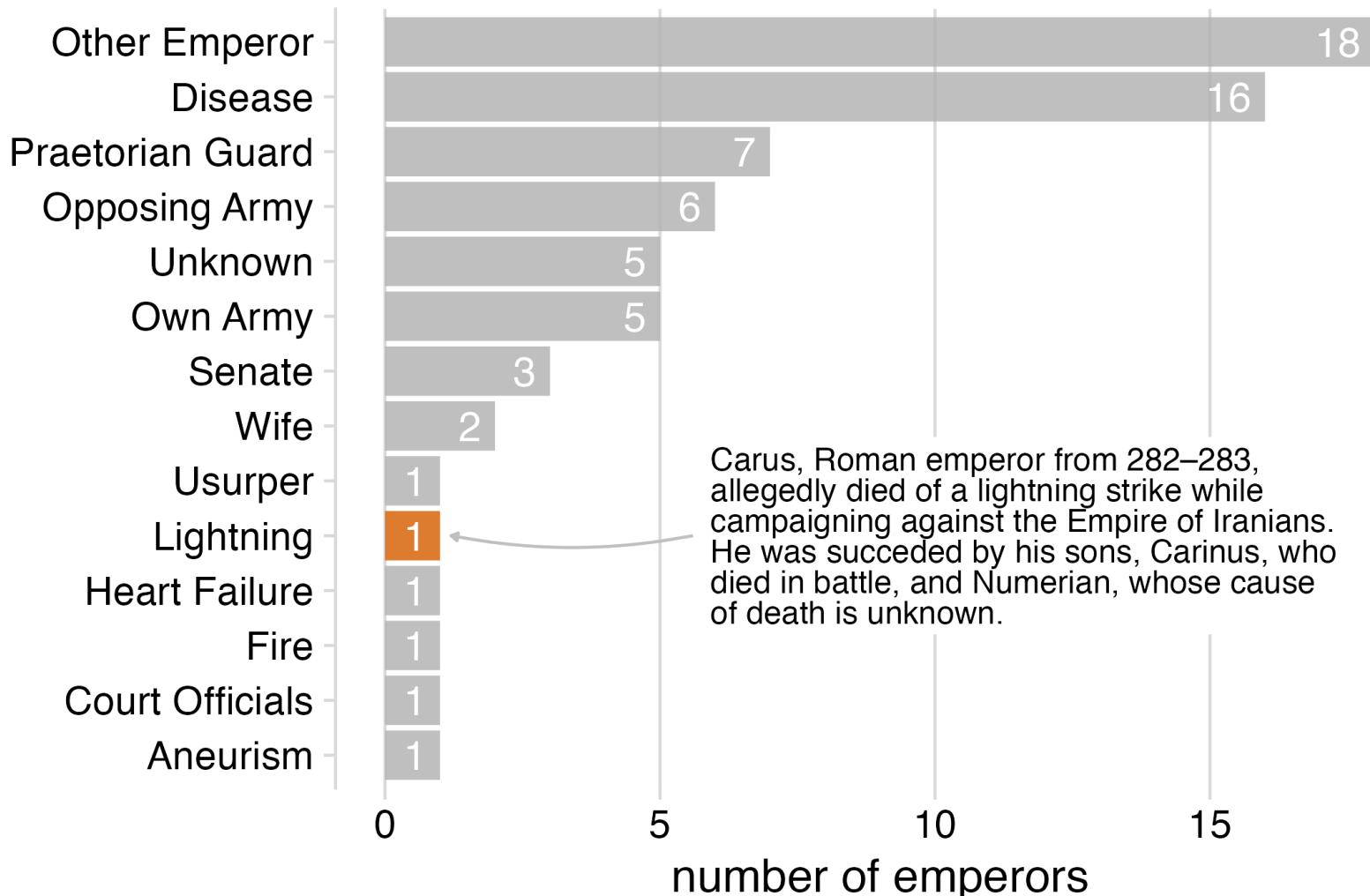
```
dog_plot +
  ggrepel::geom_label_repel(
    data = filter(
      dog_sighting,
      str_detect(other_activities, "teasing"))
  ),
  aes(x = long, y = lat, label = lbl),
  nudge_x = .015,
  size = 3.5,
  lineheight = .8,
  segment.color = "grey70"
) +
cowplot::theme_map() +
theme(legend.position = c(.05, .9)) +
scale_color_manual(name = NULL, values = "#FB1919")
```

- squirrel interacting with a dog



```
label <- "Carus, Roman emperor from 282–283,  
allegedly died of a lightning strike while  
campaigning against the Empire of Iranians.  
He was succeeded by his sons, Carinus, who  
died in battle, and Numerian, whose cause  
of death is unknown."
```

```
lightning_plot +  
  geom_label(  
    data = data.frame(x = 5.8, y = 5, label = label),  
    aes(x = x, y = y, label = label),  
    hjust = 0,  
    lineheight = .8,  
    inherit.aes = FALSE,  
    label.size = NA  
  ) +  
  geom_curve(  
    data = data.frame(x = 5.6, y = 5, xend = 1.2, yend = 5),  
    mapping = aes(x = x, y = y, xend = xend, yend = yend),  
    colour = "grey75",  
    size = 0.5,  
    curvature = -0.1,  
    arrow = arrow(length = unit(0.01, "npc"), type = "closed"),  
    inherit.aes = FALSE  
)
```



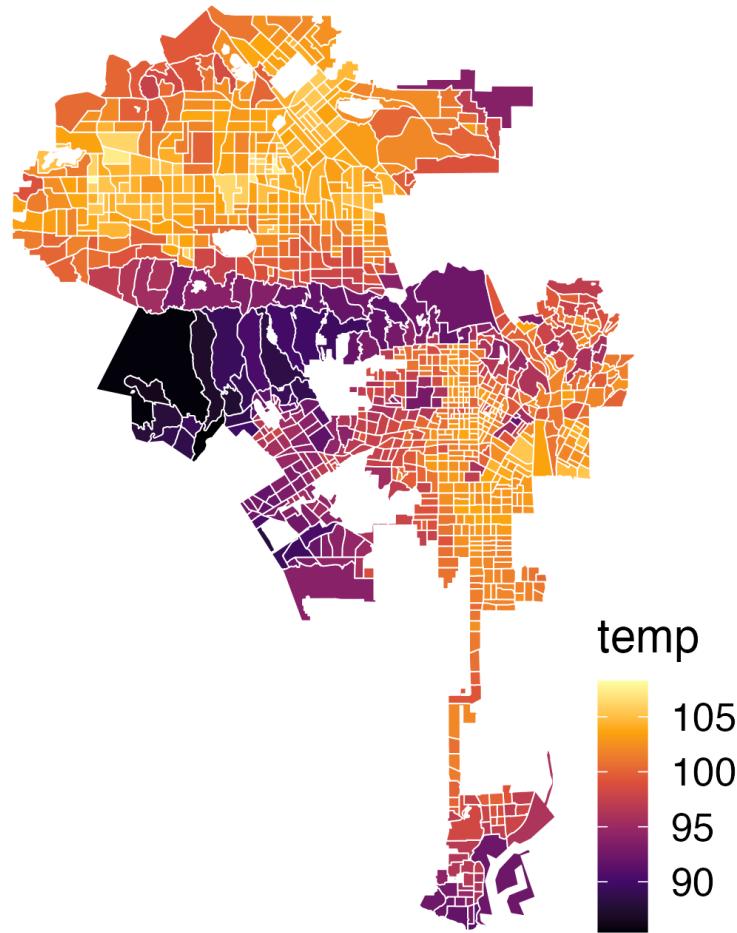
```
plot +
  geom_text(
    data = text_df,
    mapping = aes(<mappings>),
    hjust = 0
  ) +
  geom_curve(
    data = arrow_df,
    mapping = aes(<mappings>),
    curvature = -0.1,
    arrow = arrow()
  )
```

```
plot +
  geom_text(
    data = text_df,
    mapping = aes(<mappings>),
    hjust = 0
  ) +
  geom_curve(
    data = arrow_df,
    mapping = aes(<mappings>),
    curvature = -0.1,
    arrow = arrow()
  )

```

The diagram illustrates the mapping of data frames to ggplot2 geoms. It shows two separate code blocks. The first block uses `geom_text` and the second uses `geom_curve`. Each `data` argument is connected by an orange arrow to its corresponding data frame definition on the right.

Your Turn 6



Your Turn 6

Run the first code chunk and take a look at the map of surface heat in Los Angeles.

In the second code chunk, we need to create two new data frames to draw the annotations and arrows. Pick a name for each.

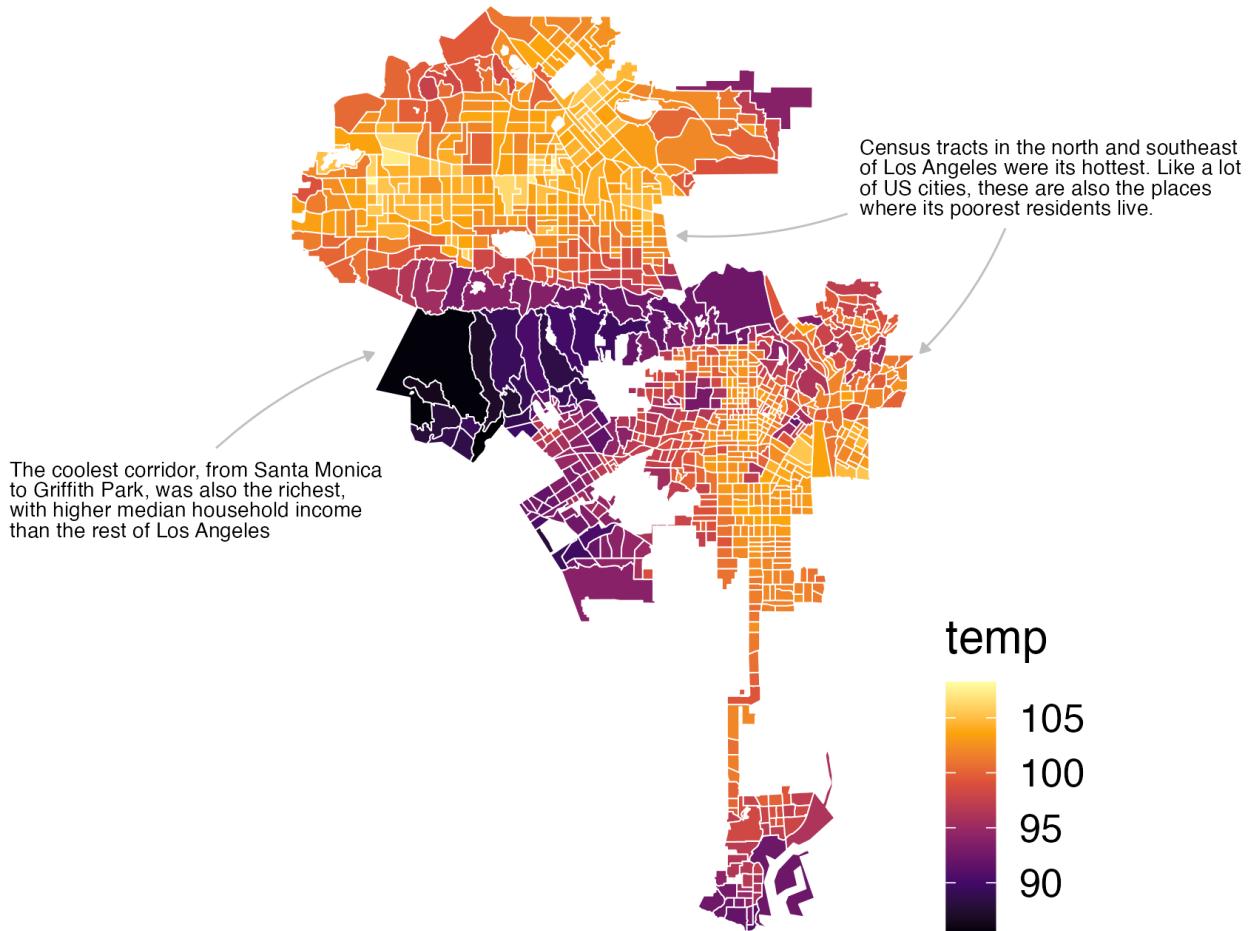
Add `geom_text()` and `geom_curve()` to `base_map`. Give each geom the relevant data that you just named in (2).

Let's clean up the geoms a bit. Reduce the lineheight of the text geom to 0.8. Then, add arrows to the curve geom using the `arrow()` function. Give it two arguments: `length = unit(0.01, "npc")` and `type = "closed"`. Run the plot.

One of the labels is being clipped because it runs off the main plotting panel. Add `coord_sf(clip = "off")` to prevent clipping the text.

```
text_labels <- tibble::tribble(  
  ~x,      ~y,      ~label,  
  -118.90, 34.00,  west_label,  
  -118.20, 34.22,  east_label  
)  
  
arrows <- tibble::tribble(  
  ~x, ~y, ~xend, ~yend,  
  -118.73, 34.035, -118.60, 34.10,  
  -118.21, 34.195, -118.35, 34.18,  
  -118.08, 34.185, -118.15, 34.10  
)
```

```
base_map +
  geom_text(
    data = text_labels,
    aes(x, y, label = label),
    hjust = 0,
    vjust = 0.5,
    lineheight = .8
  ) +
  geom_curve(
    data = arrows,
    aes(x = x, y = y, xend = xend, yend = yend),
    colour = "grey75",
    size = 0.3,
    curvature = -0.1,
    arrow = arrow(length = unit(0.01, "npc"), type = "closed")
  ) +
  coord_sf(clip = "off")
```



How do we augment plots to explain?

Annotate plots using text geoms and arrows

Combine plots to build a cohesive narrative

Combine plots to tell a story

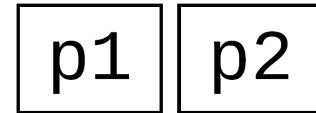
- 1 Build plots up from simpler to more complex
- 2 Don't use the same type of plot in each panel
- 3 Use consistent color

patchwork: Compose ggplots

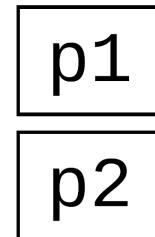
```
library(patchwork)
```



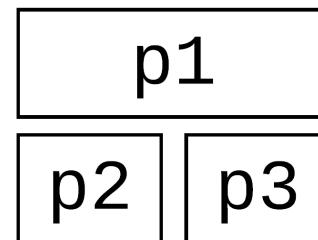
$p_1 + p_2$



p_1 / p_2



$p_1 /$
 $(p_2 + p_3)$



p1 + p2

combine
with +



p1 / p2

stack
with /



p1 /
(p2 + p3)

group
with ()



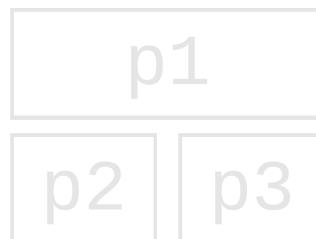
```
p1 +  
  plot_spacer() +  
p2
```



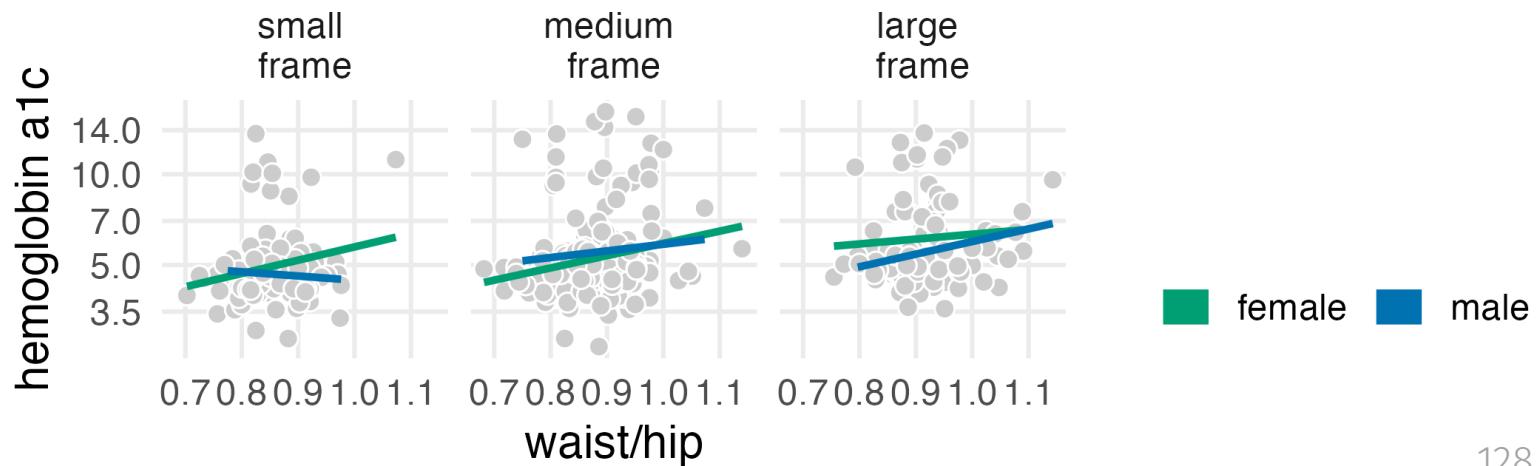
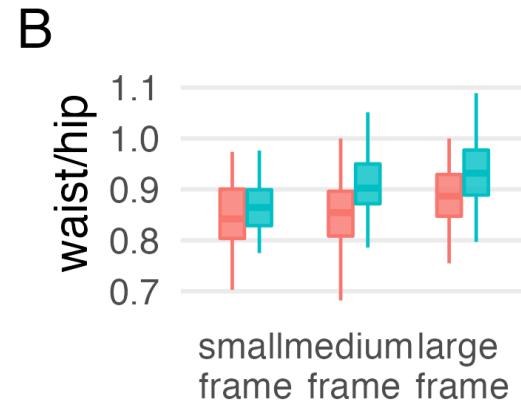
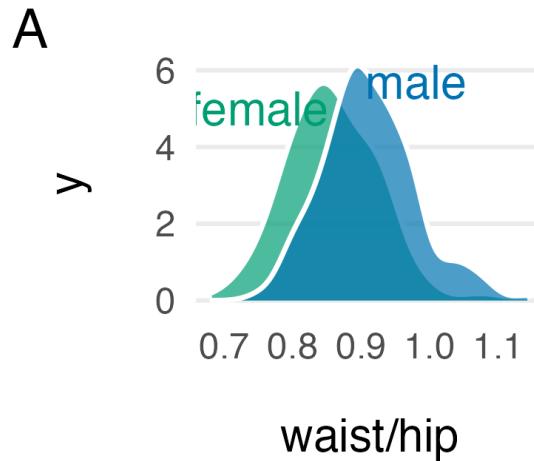
```
p1 / p2 +  
  plot_layout(  
    heights = c(3, 1)  
)
```



```
p1 /  
(p2 + p3)
```



Your Turn 7



Run the first code chunk. `label_frames()` will help us label the frame variable better. `theme_multiplot()` is the theme we'll add to each plot. We'll use `diabetes_complete` for the plots (removing the missing values of the variables we're plotting produce the same plots as `diabetes` would, but it prevents `ggplot2` from warning us that it's dropping the data internally). Nothing to change here!

Run the code for `plot_a` and take a look. Nothing to change here, either!

The colors in `plot_b` don't match `plot_a`. Add `scale_color_manual()` to make the colors consistent.

Also add `scale_fill_manual()`. For the fill colors, we'll add a bit of transparency. Paste "B3" to the end of the colors in `plot_colors`. "B3" is equivalent to 70% transparency (or `alpha = .7`) in hex code (see [this GitHub page with translations from percent to hex](#), but note that in R you need to put the transparency at the end of the six character hex code).

This plot doesn't have a tag label like the other two plots. Add one to the `labs()` call.

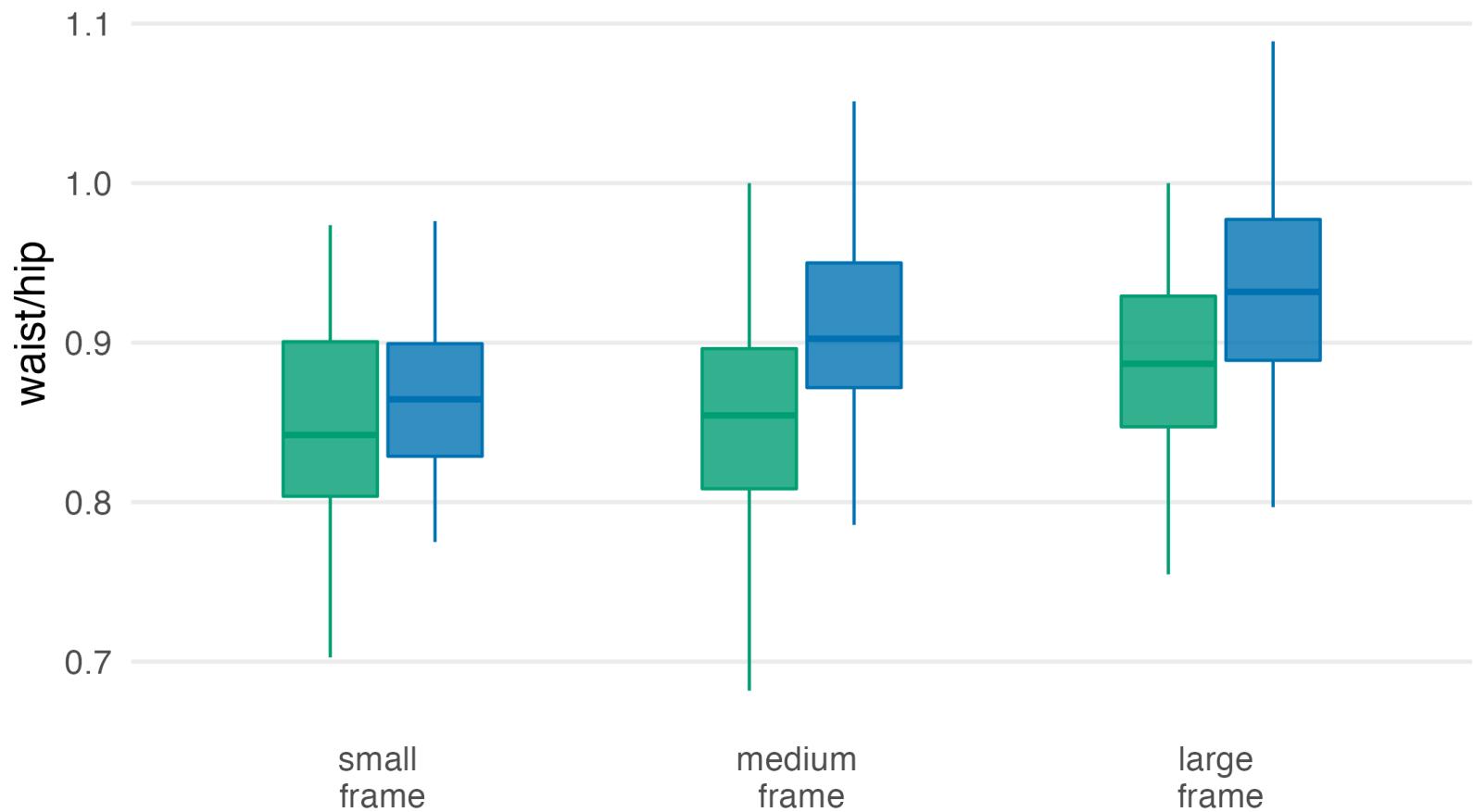
The legend isn't working well, but let's take advantage of it. We'll move the legend to [above](#) the plot by setting `legend.position` to `c(1, 1.25)` in `theme()`. We won't be able to see it in `plot_c`, but it will show up in the combined plot!

Finally, combine the 3 plots using patchwork. Have `plot_a` and `plot_b` on top and `plot_c` on the bottom.

```
plot_b <- diabetes_complete %>%
  ggplot(
    aes(fct_rev(frame),
        waist / hip,
        fill = gender,
        col = gender
    )
  ) +
  geom_boxplot(
    outlier.color = NA,
    alpha = .8,
    width = .5
  ) +
  theme_multiplot() +
  theme(axis.title.x = element_blank()) +
  scale_color_manual(values = plot_colors) +
  scale_fill_manual(values = paste0(plot_colors, "B0")) +
  scale_x_discrete(labels = label_frames) +
  labs(tag = "B")
```

```
plot_b
```

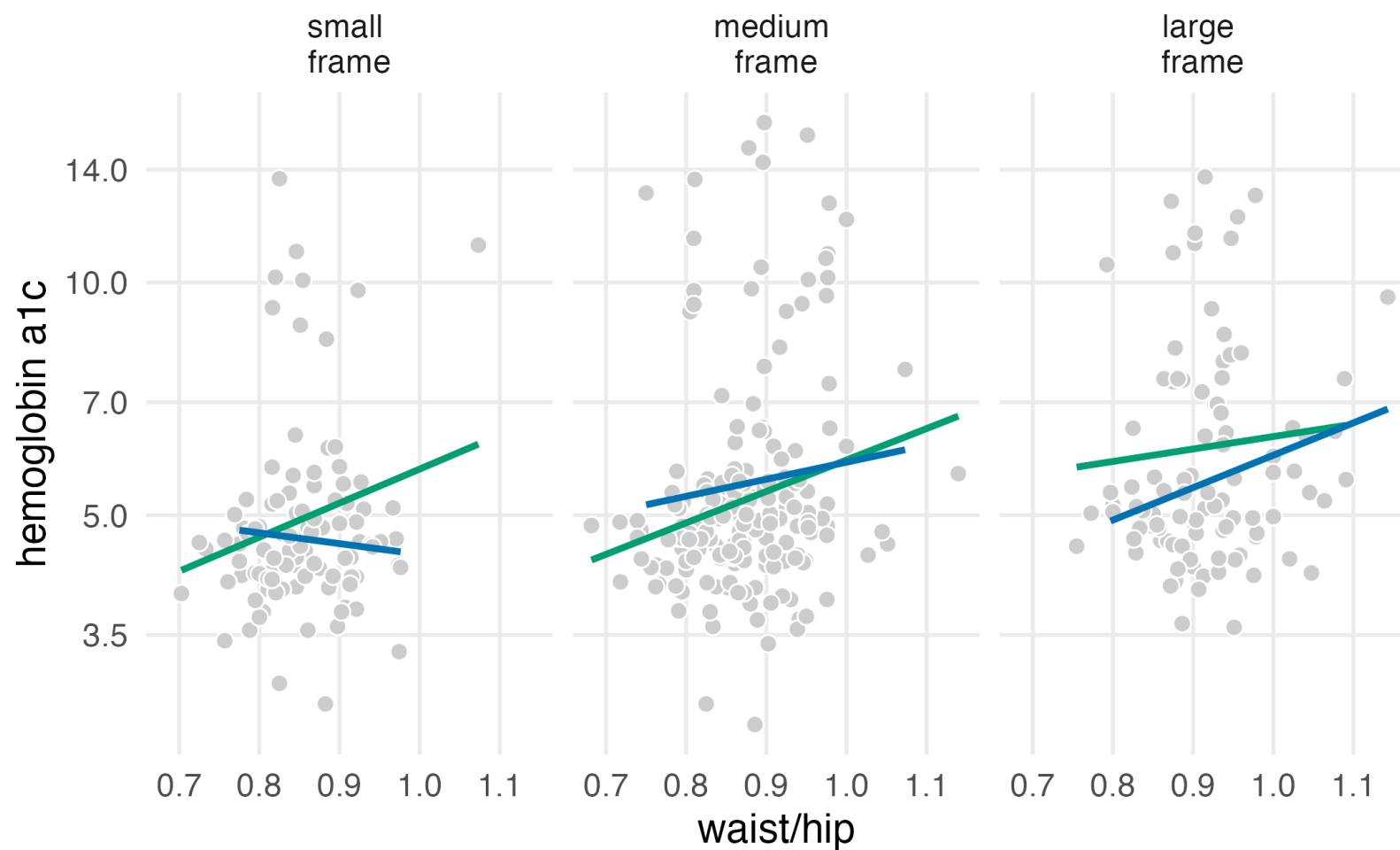
B



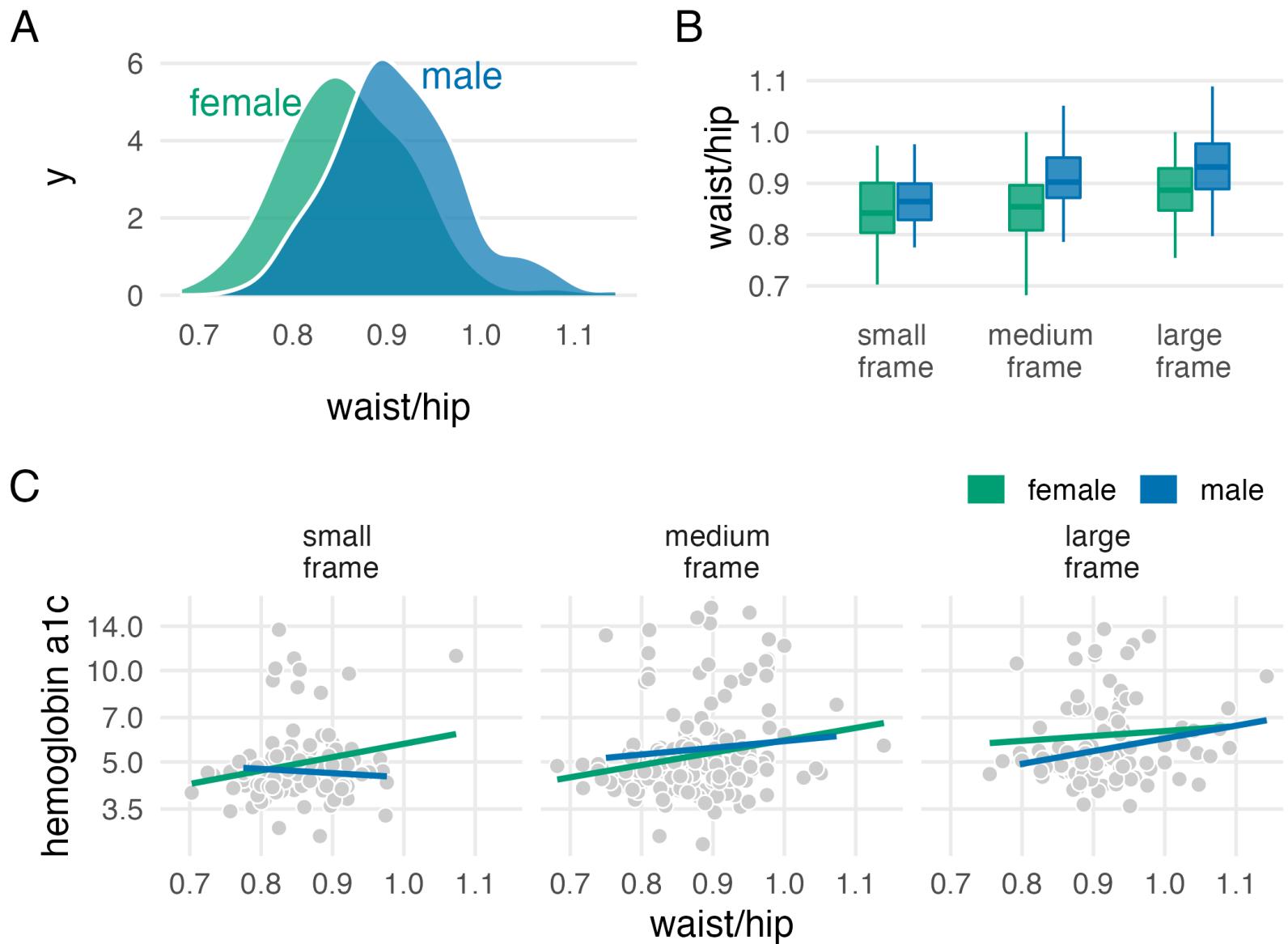
```
plot_c <- diabetes_complete %>%
  ggplot(aes(waist / hip, glyhb, col = gender)) +
  geom_point(
    shape = 21,
    col = "white",
    fill = "grey80",
    size = 2.5
  ) +
  geom_smooth(
    method = "lm",
    formula = y ~ x,
    se = FALSE,
    size = 1.1
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = c(1, 1.25),
    legend.justification = c(1, 0),
    legend.direction = "horizontal",
    panel.grid.minor = element_blank()
  ) +
  facet_wrap(~ fct_rev(frame), labeller = as_labeller(label_frames)) +
  scale_y_log10(breaks = c(3.5, 5.0, 7.0, 10.0, 14.0)) +
  scale_color_manual(name = "", values = plot_colors) +
  guides(color = guide_legend(override.aes = list(size = 5))) +
  labs(y = "hemoglobin a1c", tag = "C")
```

```
plot_c
```

C



```
library(patchwork)  
(plot_a + plot_b) / plot_c
```



Resources

Fundamentals of Data Visualization by
Claus O. Wilke

Storytelling with Data by Cole
Nussbaumer Knaflic

Better Presentations by Jonathan
Schwabish