Week-7: Visualizing data using `ggplot`

NM2207: Computational Media Literacy

Narayani Vedam, Ph.D.

Department of Communications and New Media

NUS National University of Singapore | Faculty of Arts & Social Sciences

This week

Table of contents

I. All about `ggplot2` package

ggplot2

- It is the `tidyverse` library's visualization package
- `gg` in `ggplot` stands for **G**rammar of **G**raphics
- Enables us to describe different components of a graphic
- Structure of the code for plots can be summarized as

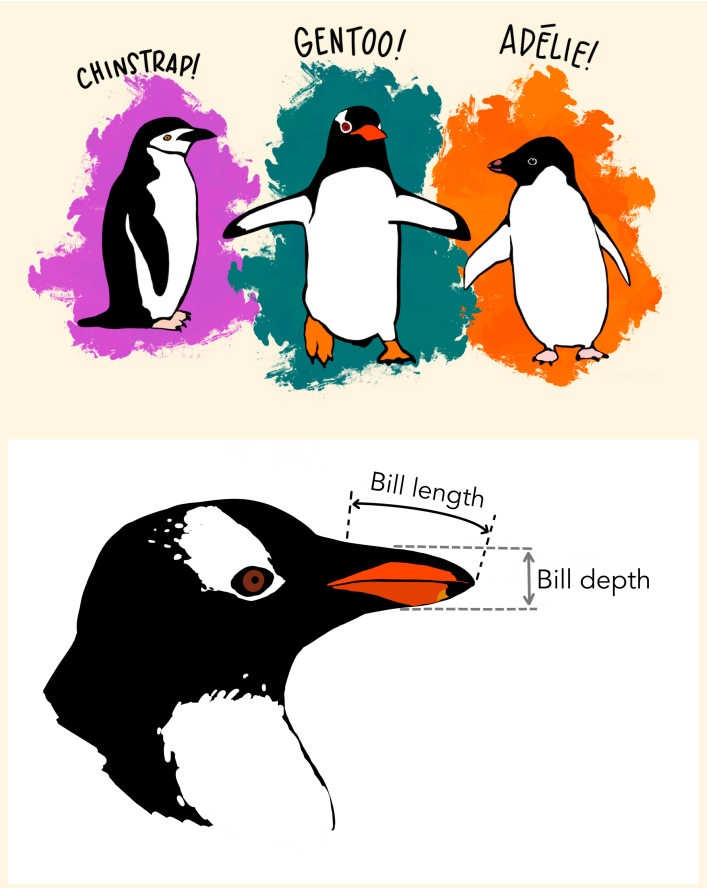
```
ggplot(data = [dataset],
       mapping = aes(x = [x-variable],
                     y = [y-variable])) +
   geom_xxx() +
   other options
```

Aesthetics specified in ggplot() are inherited by subsequent layers. Aesthetic specified in particular layers are specific only to that layer

Figure: Different layers of ggplot

Data: Palmer Penguins

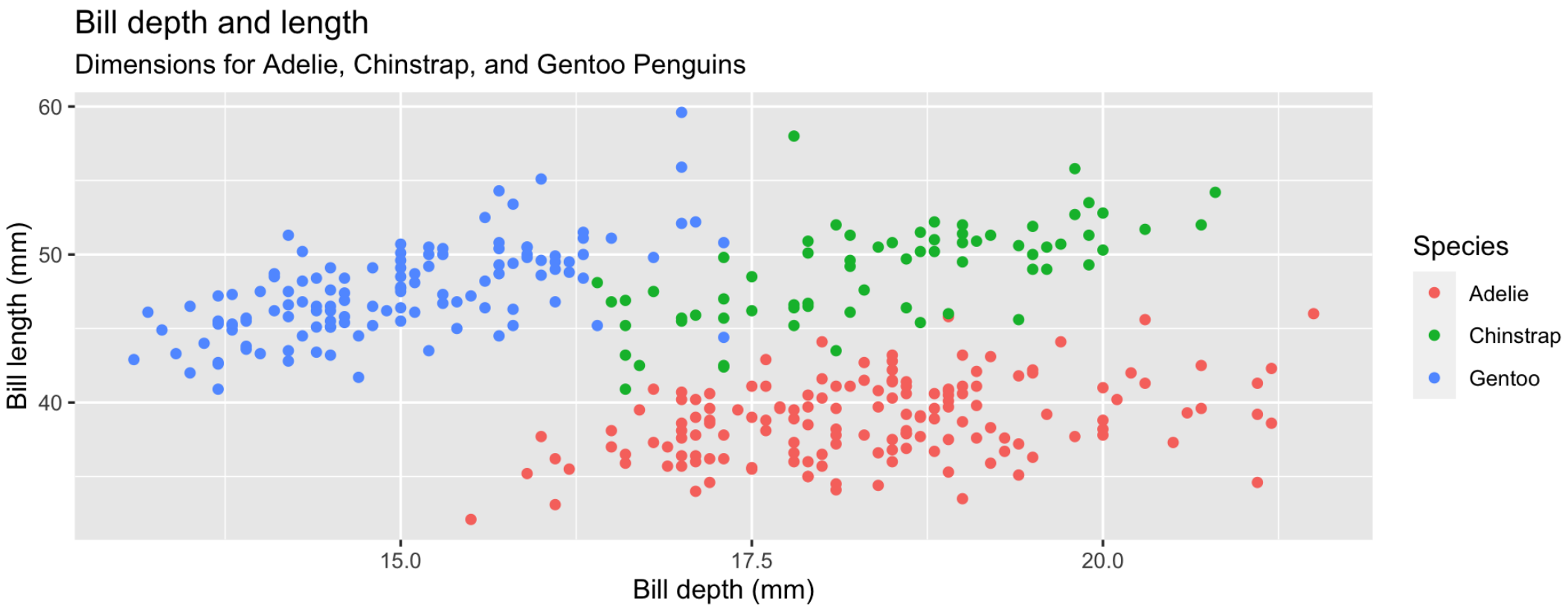
Measurements for penguin species, island in Palmer Archipelago, size (flipper length, body mass, bill dimensions), and gender.




```
library(tidyverse)
library(palmerpenguins)
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Ad
## $ island            <fct> Torgersen, Torgers
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3,
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0,
## $ flipper_length_mm <int> 181, 186, 195, NA
## $ body_mass_g       <int> 3750, 3800, 3250,
## $ sex               <fct> male, female, fema
## $ year              <int> 2007, 2007, 2007,
```

Plots: Palmer Penguins



Palmer Penguins: Plot recreation

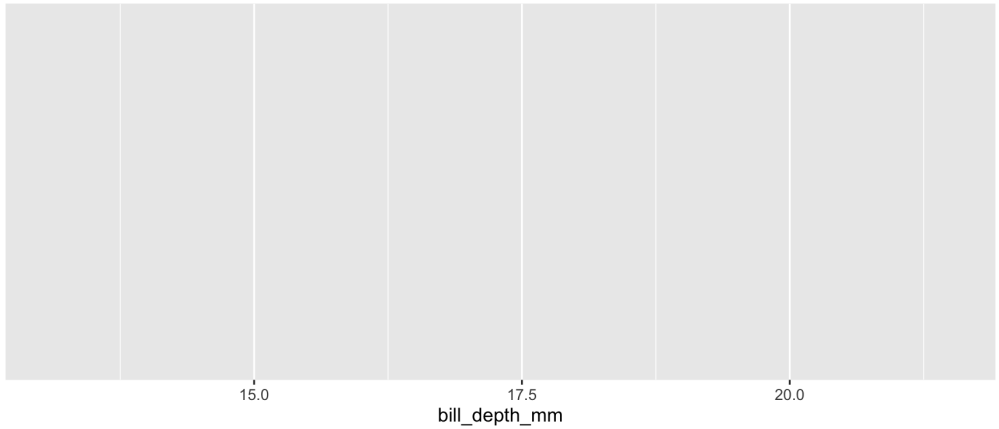
a. Start with the `penguins` data frame

```
ggplot(data = penguins)
```

Palmer Penguins: Plot recreation

b. Map **bill depth** to the x-axis

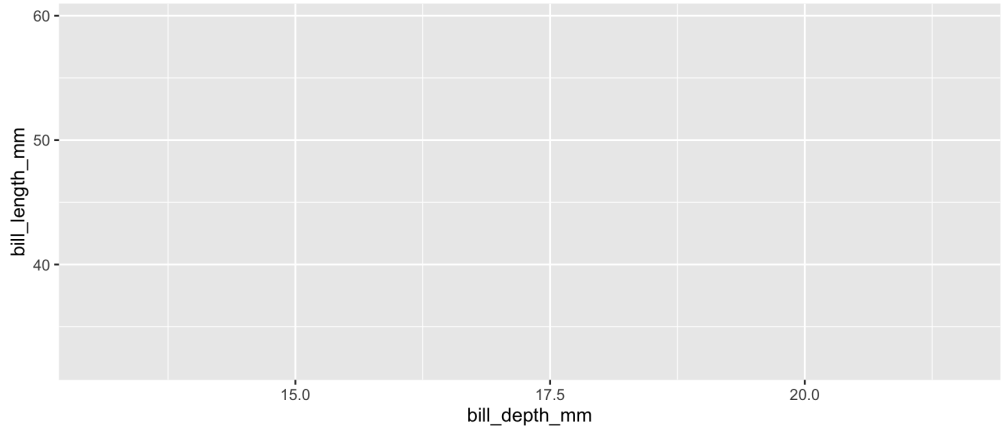
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm))
```



Palmer Penguins: Plot recreation

c. Map **bill length** to the y-axis

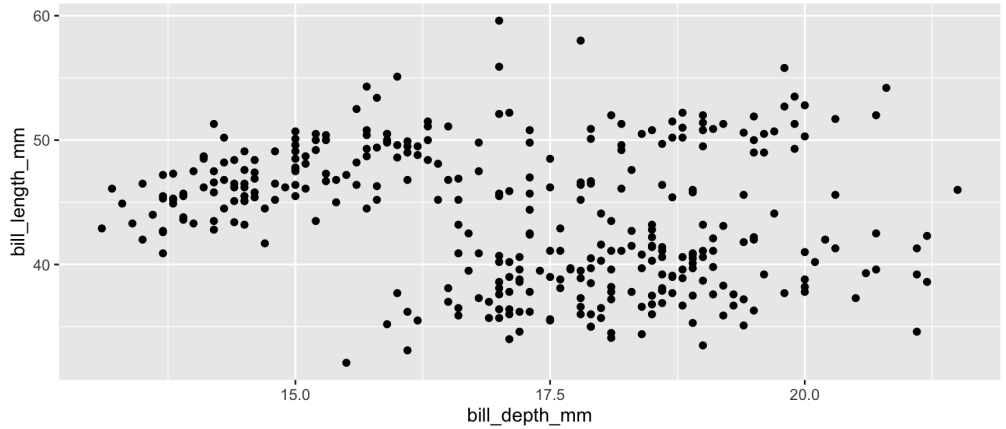
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm))
```



Palmer Penguins: Plot recreation

d. Represent each observation with a point

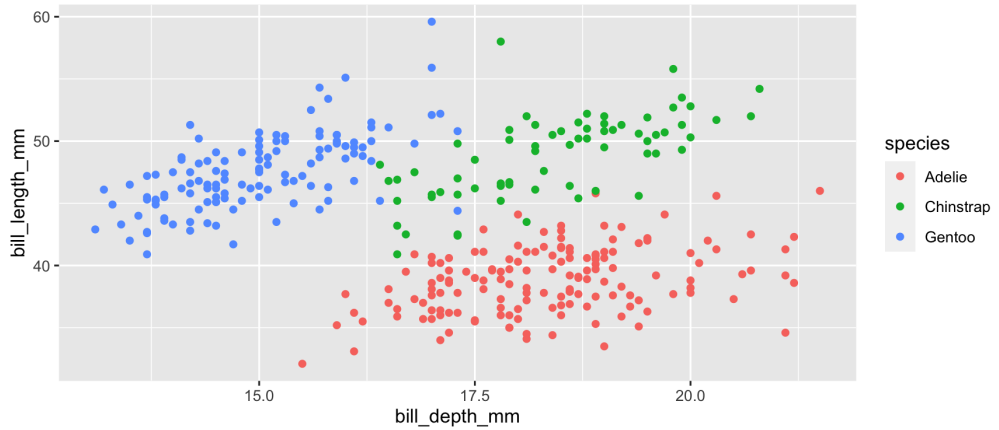
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm)) +
  geom_point()
```



Palmer Penguins: Plot recreation

e. Map species to the colour of each point

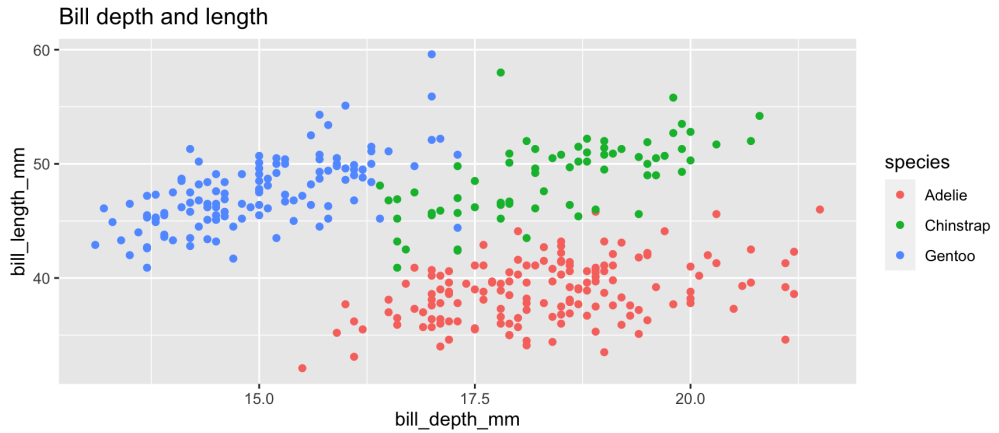
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point()
```



Palmer Penguins: Plot recreation

f. Title the plot "Bill depth and length"

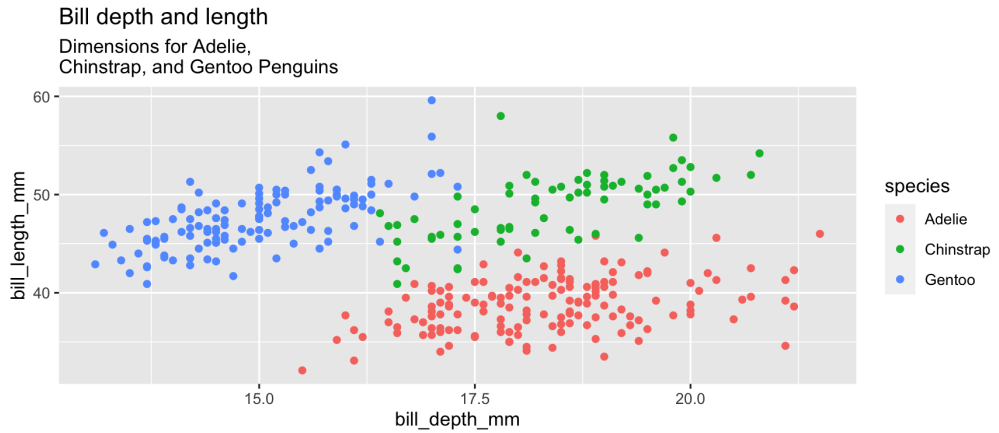
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length")
```



Palmer Penguins: Plot recreation

g. Add the subtitle "Dimensions for Adelie, Chinstrap, and Gentoo Penguins"

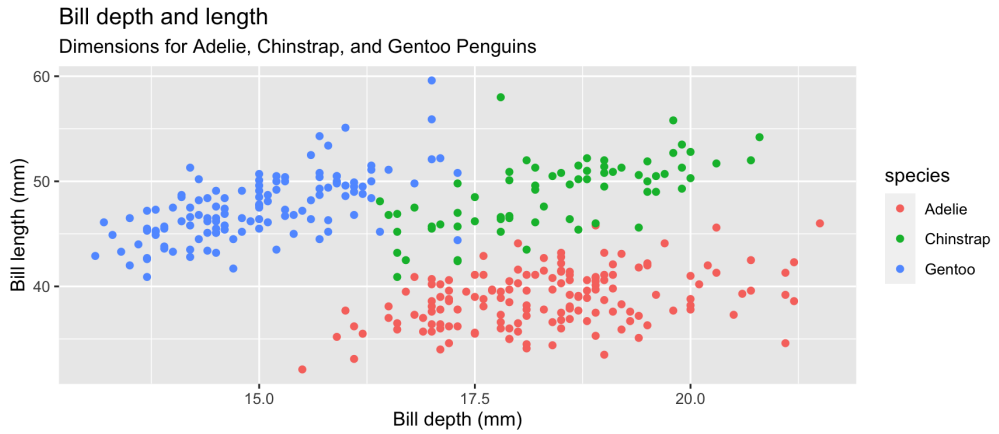
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length",
subtitle = "Dimensions for Adelie,
Chinstrap, and Gentoo Penguins")
```



Palmer Penguins: Plot recreation

h. Label the x and y axes as "Bill depth (mm)" and "Bill length (mm)", respectively

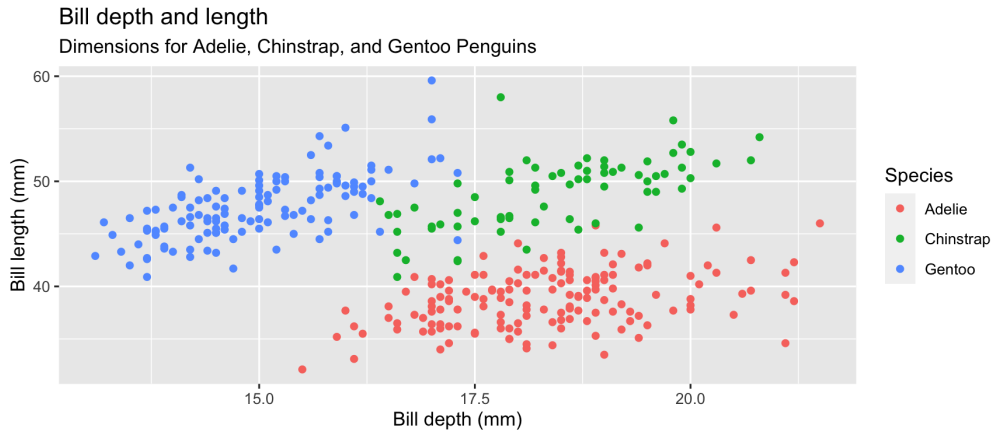
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length",
subtitle = "Dimensions for Adelie, Chinstrap,
       x = "Bill depth (mm)",
       y = "Bill length (mm)")
```



Palmer Penguins: Plot recreation

i. Label the legend "Species"

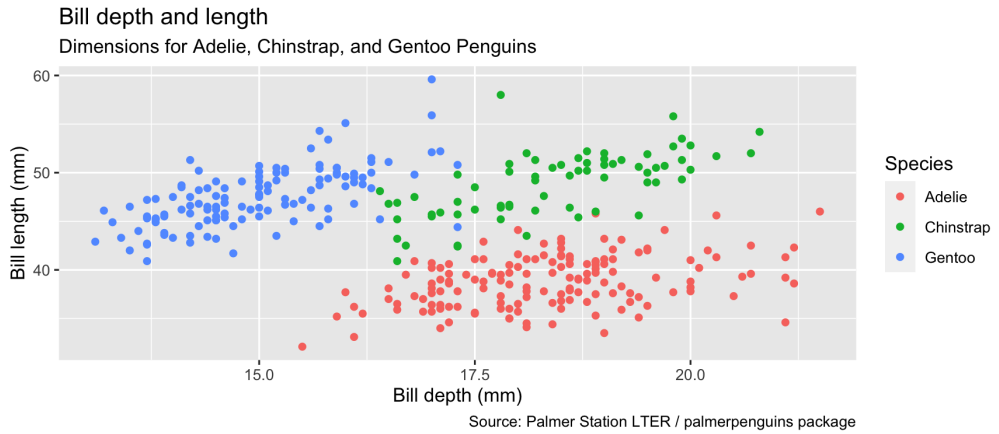
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length",
       subtitle = "Dimensions for Adelie, Chi
       x = "Bill depth (mm)", y = "Bill lengt
       colour = "Species")
```



Palmer Penguins: Plot recreation

j. Add a caption for the data source

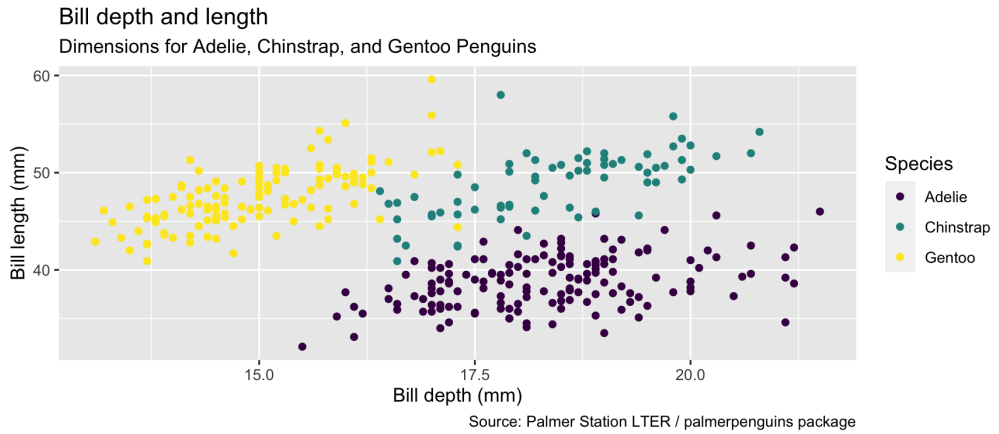
```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length",
       subtitle = "Dimensions for Adelie, Chi
       x = "Bill depth (mm)", y = "Bill lengt
       colour = "Species",
       caption = "Source: Palmer Station LTER
```



Palmer Penguins: Plot recreation

k. Finally, use a discrete colour scale that is designed to be perceived by viewers with common forms of colour blindness.

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  labs(title = "Bill depth and length",
       subtitle = "Dimensions for Adelie, Chi
       x = "Bill depth (mm)", y = "Bill lengt
       colour = "Species",
       caption = "Source: Palmer Station LTER
  scale_colour_viridis_d()
```



Palmer Penguins: Recreated plot

a. Start with the `penguins` data frame,

b. Map bill depth to the x-axis

c. Map bill length to the y-axis

d. Represent each observation with a point

e. Map species to the colour of each point

f. Title the plot "Bill depth and length"

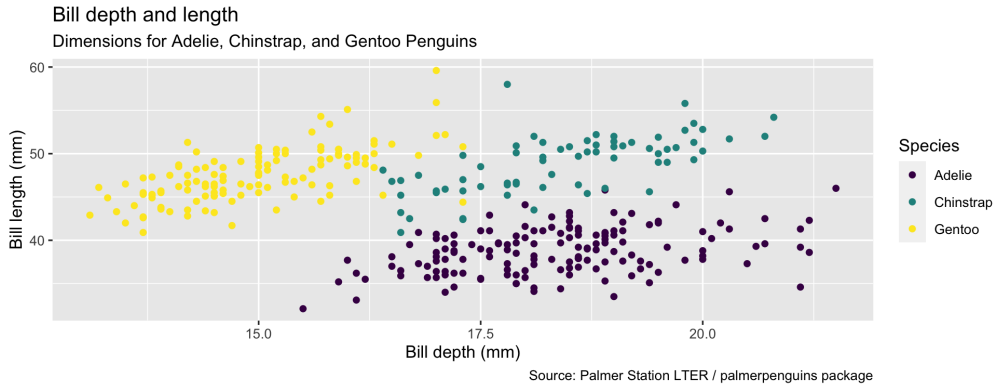
g. Add the subtitle "Dimensions for Adelie, Chinstrap, and Gentoo Penguins"

h. Label the x and y axes as "Bill depth (mm)" and "Bill length (mm)", respectively

i. Label the legend "Species"

j. Add a caption for the data source

k. Finally, use a discrete colour scale that is designed to be



Palmer Penguins: Argument names

You can omit the names of first two arguments when building plots with `ggplot()`

```
ggplot(data = penguins,
       mapping = aes(x = bill_depth_mm,
                     y = bill_length_mm,
                     colour = species)) +
  geom_point() +
  scale_colour_viridis_d()
```

```
ggplot(penguins,
       aes(x = bill_depth_mm,
           y = bill_length_mm,
           colour = species)) +
  geom_point() +
  scale_colour_viridis_d()
```

```
ggplot(penguins) + # Data layer
       aes(x = bill_depth_mm,
           y = bill_length_mm,
           colour = species) + # Aesthetics layer
              geom_point() + # Geometric layer
                scale_colour_viridis_d()
```

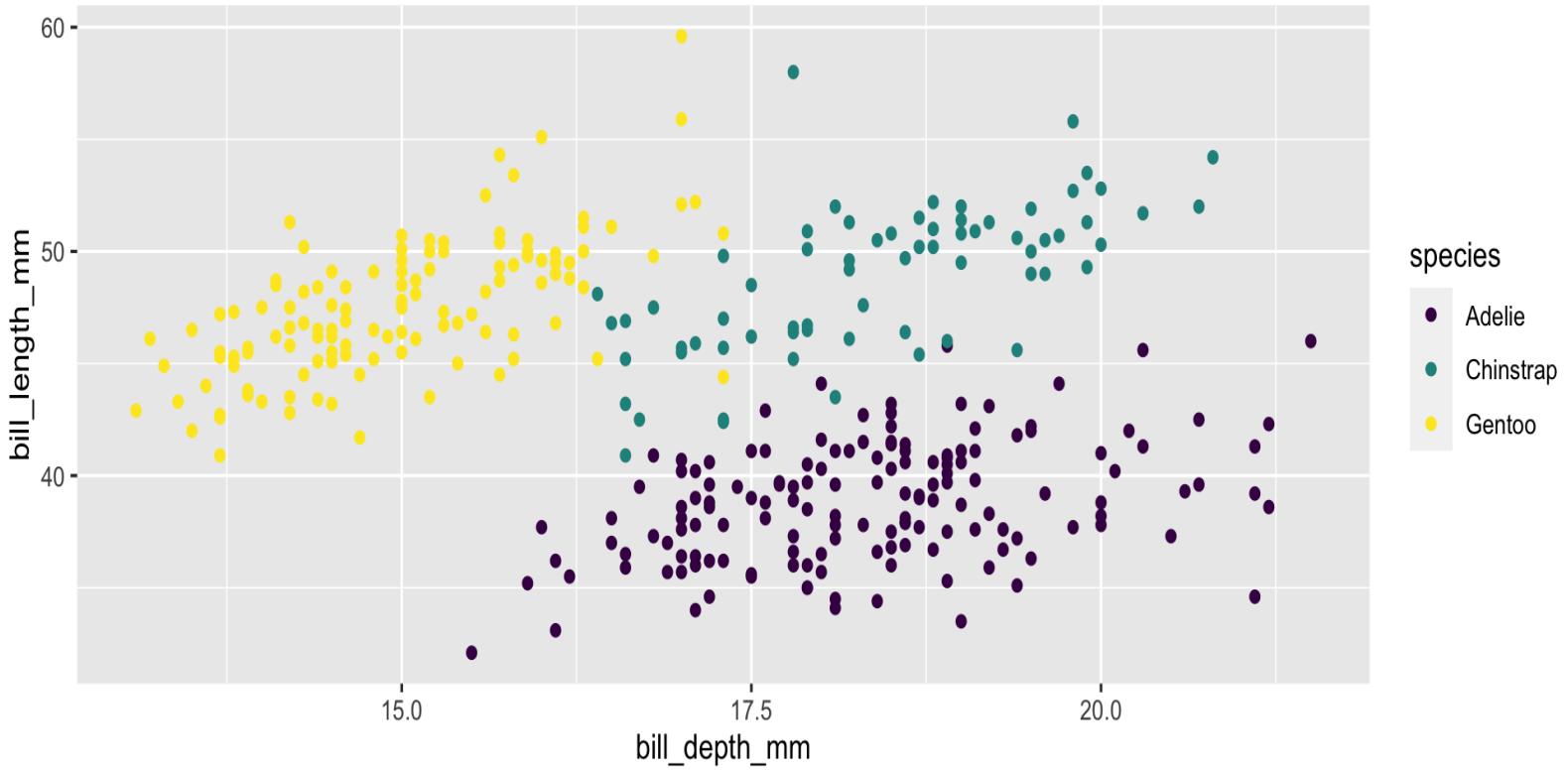
Aesthetics options

Commonly used features of `ggplot` that can be **mapped to a specific variable** in the data are,

- `colour`
- `shape`
- `size`
- `alpha` (transparency)

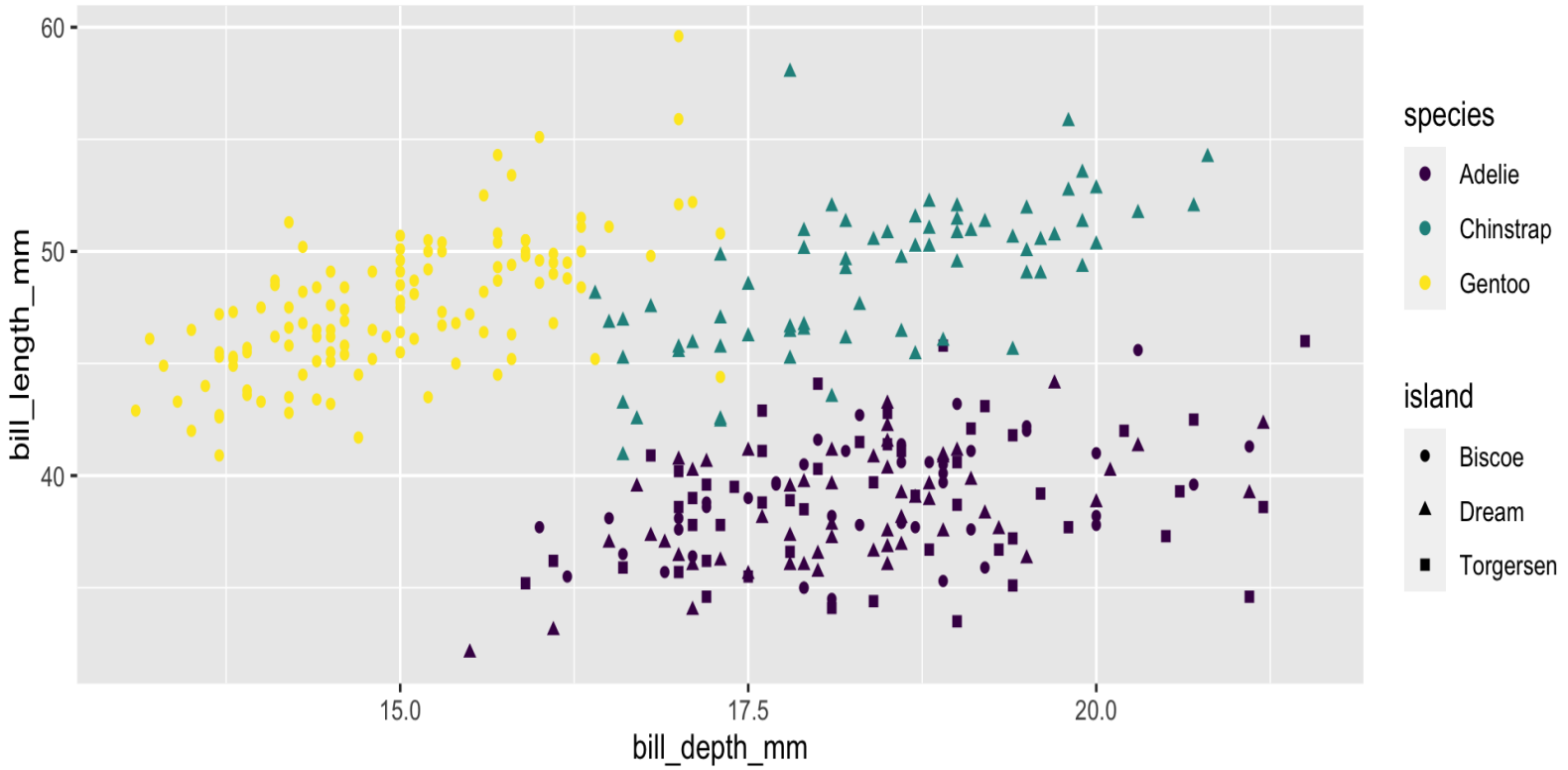
Palmer Penguins: Colour

```
ggplot(penguins) + aes(x = bill_depth_mm, y = bill_length_mm,
            colour = species) +
             geom_point() + scale_colour_viridis_d()
```



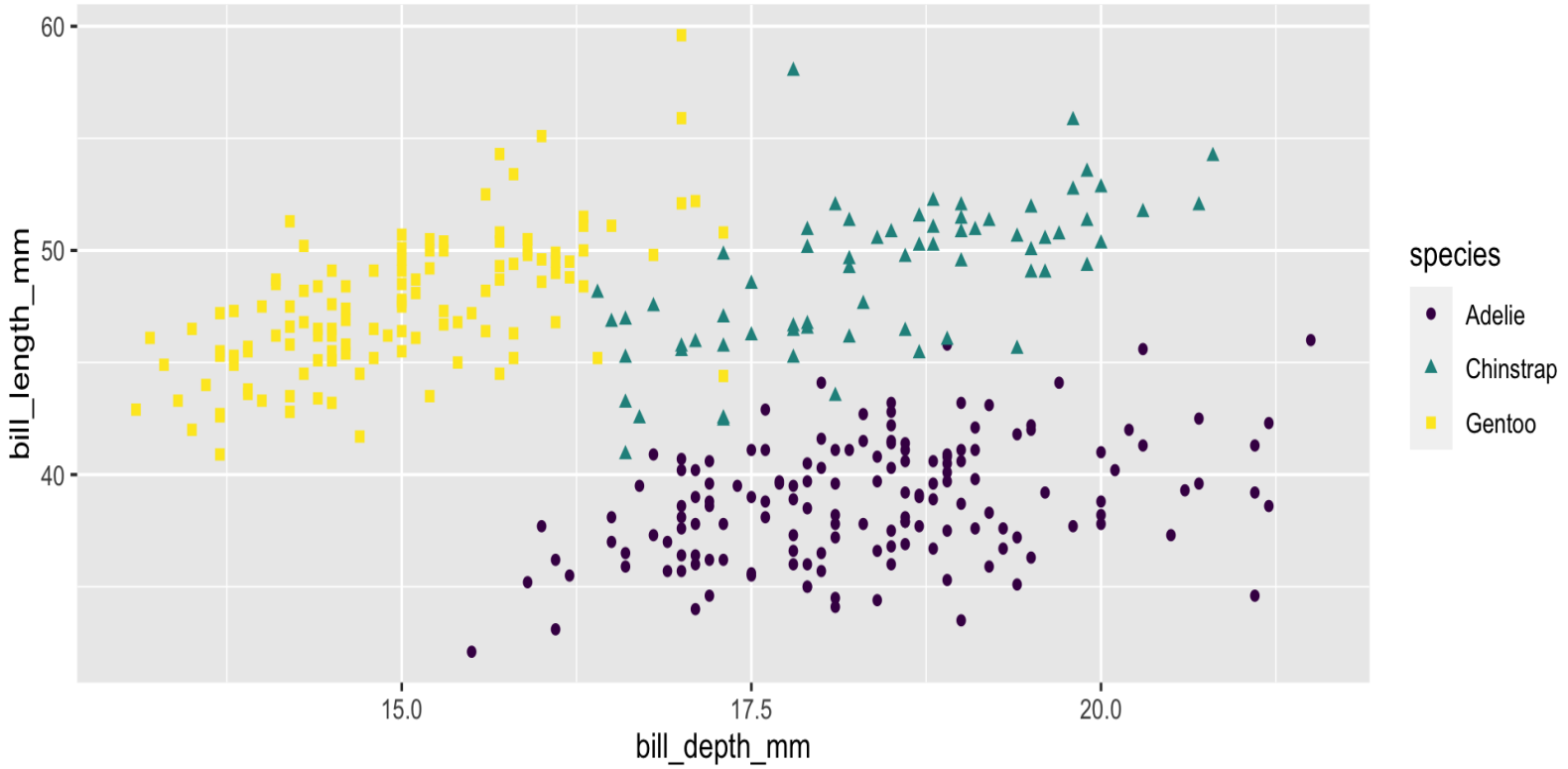
Palmer Penguins: Shape

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm, colour = species,
           shape = island)) +
  geom_point() + scale_colour_viridis_d()
```



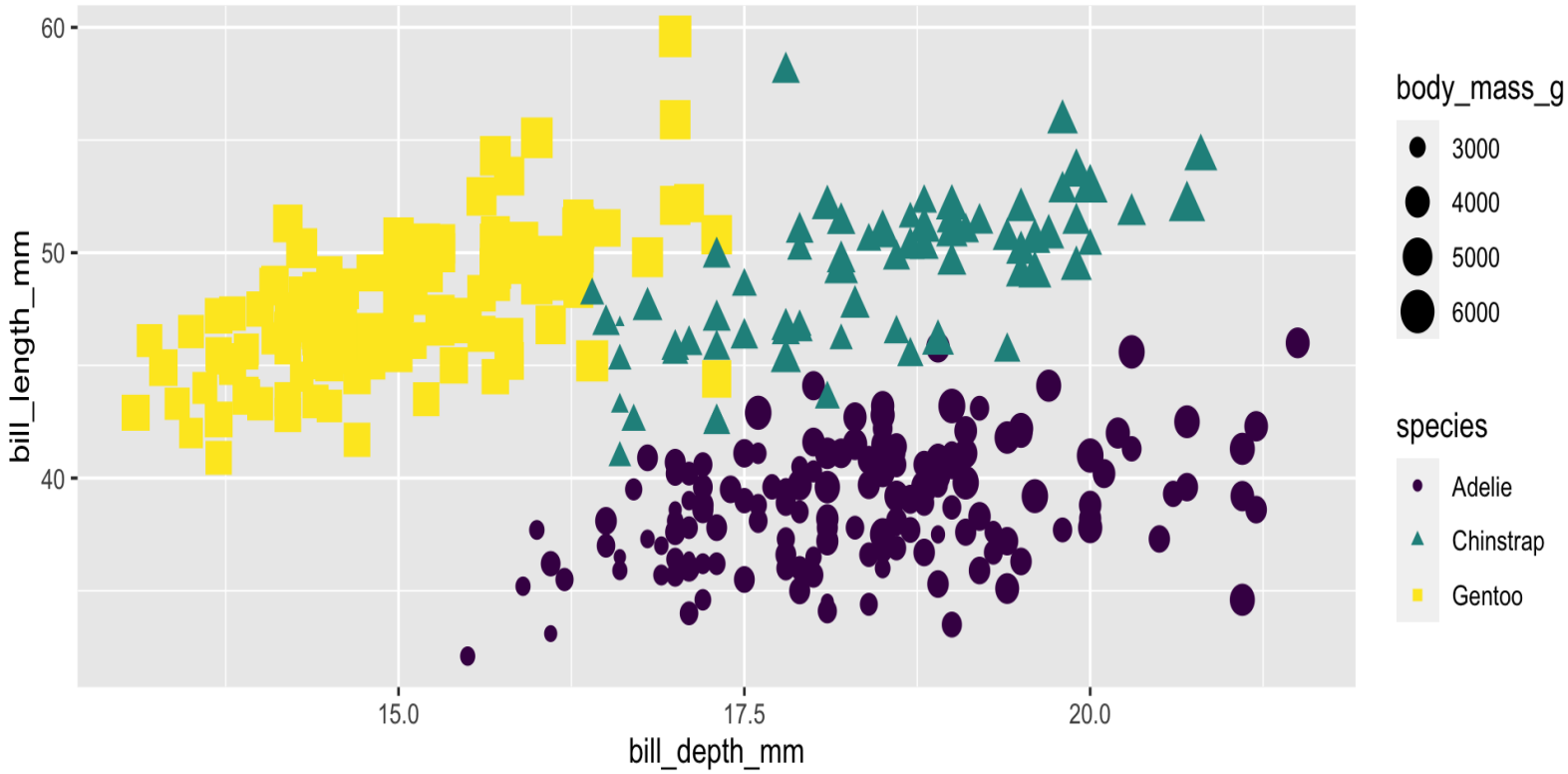
Palmer Penguins: Shape

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm, colour = species,
           shape = species)) +
  geom_point() + scale_colour_viridis_d()
```



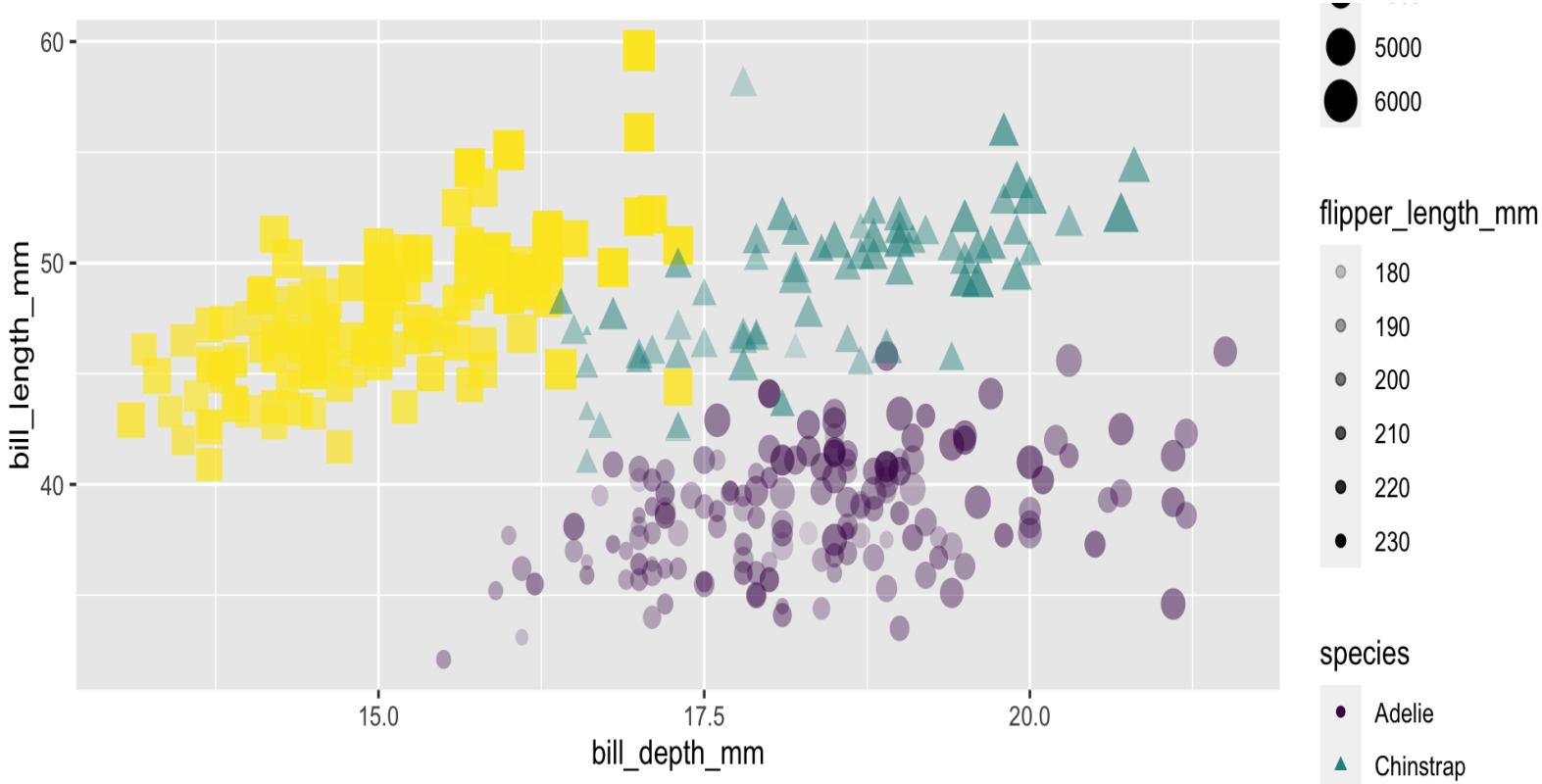
Palmer Penguins: Size

```
ggplot(penguins, aes(x = bill_depth_mm,  y = bill_length_mm, colour = species, shape = species,
       size = body_mass_g)) +
  geom_point() + scale_colour_viridis_d()
```



Alpha

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm, colour = species,
       shape = species, size = body_mass_g, alpha = flipper_length_mm)) +
  geom_point() + scale_colour_viridis_d()
```



Mapping vs. Setting

- **Mapping:** Determine the size, alpha, etc. of points based on the values of a variable in the data
  - goes into `aes()`
- **Setting:** Determine the size, alpha, etc. of points **not** based on the values of a variable in the data
  - goes into `geom_*()` (this was `geom_point()` in the previous example, but we'll learn about other geoms soon!)

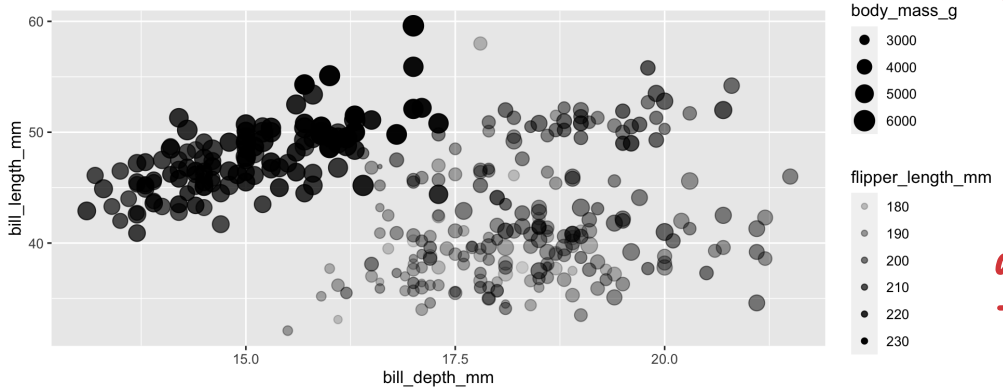
Mapping vs. Setting

aesthetic mapping

geom_'anything'

Mapping

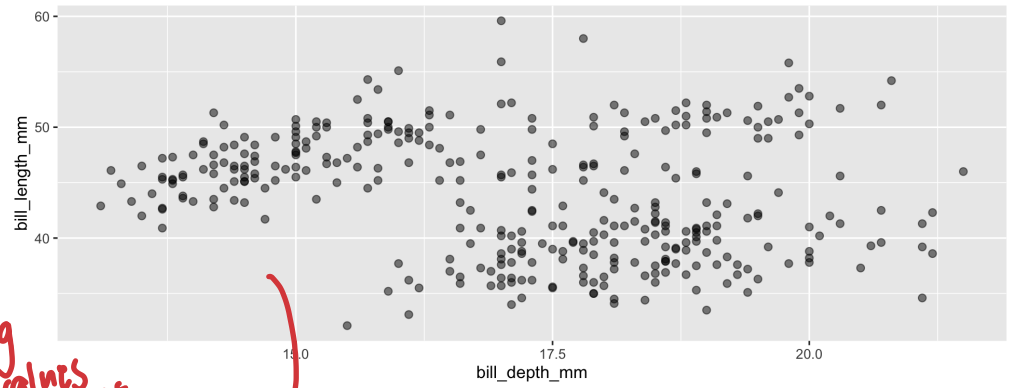
```
ggplot(penguins) +
  aes(x = bill_depth_mm,
      y = bill_length_mm,
      size = body_mass_g,
      alpha = flipper_length_mm) +
  geom_point()
```



according to the values in the dataset, different sizes

Setting

```
ggplot(penguins) +
  aes(x = bill_depth_mm,
      y = bill_length_mm) +
  geom_point(size = 2, alpha = 0.5)
```



uniform, declared sizes by you!

Faceting

- Smaller plots that display different subsets of the data
- Useful for exploring conditional relationships and large data

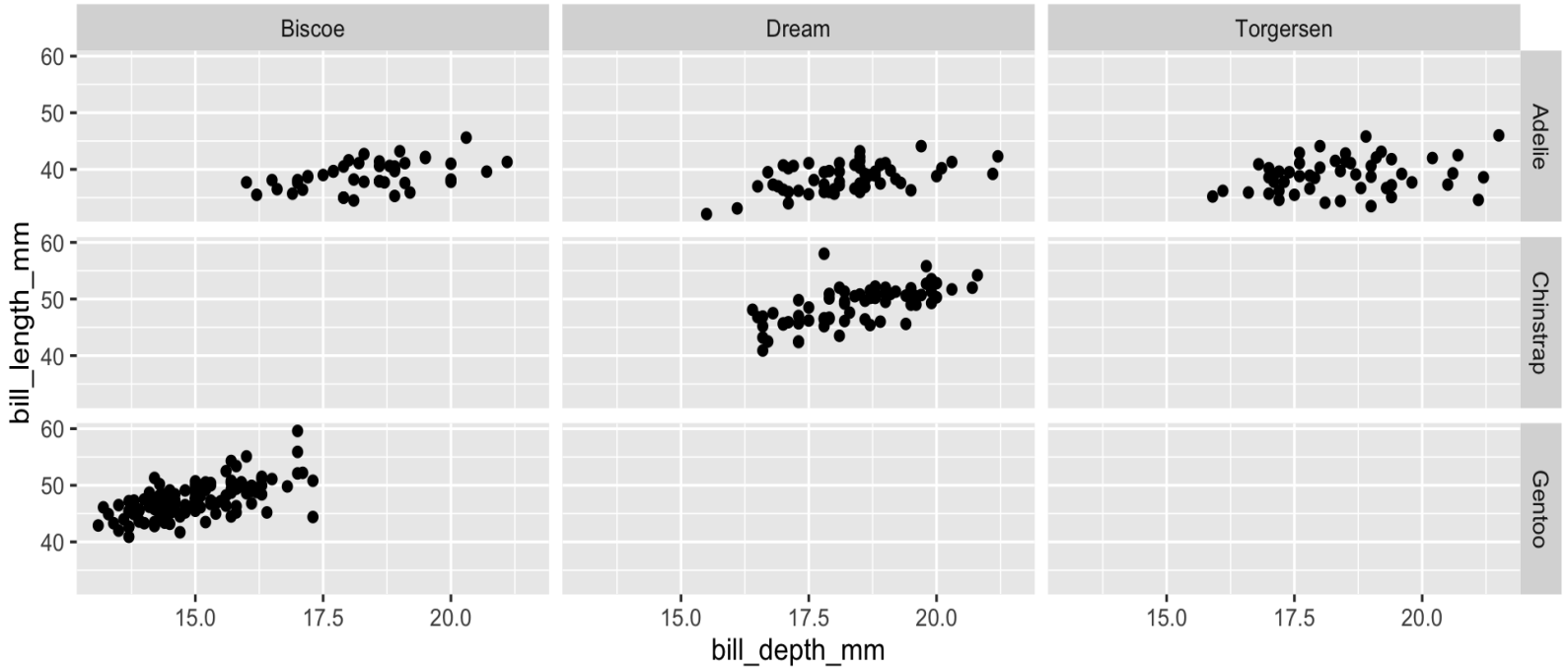
```
ggplot(penguins) +
  aes(x = bill_depth_mm,
      y = bill_length_mm) +
  geom_point() +
  facet_grid(species ~ island)
```

row columns

↳ species as rows, islands as columns

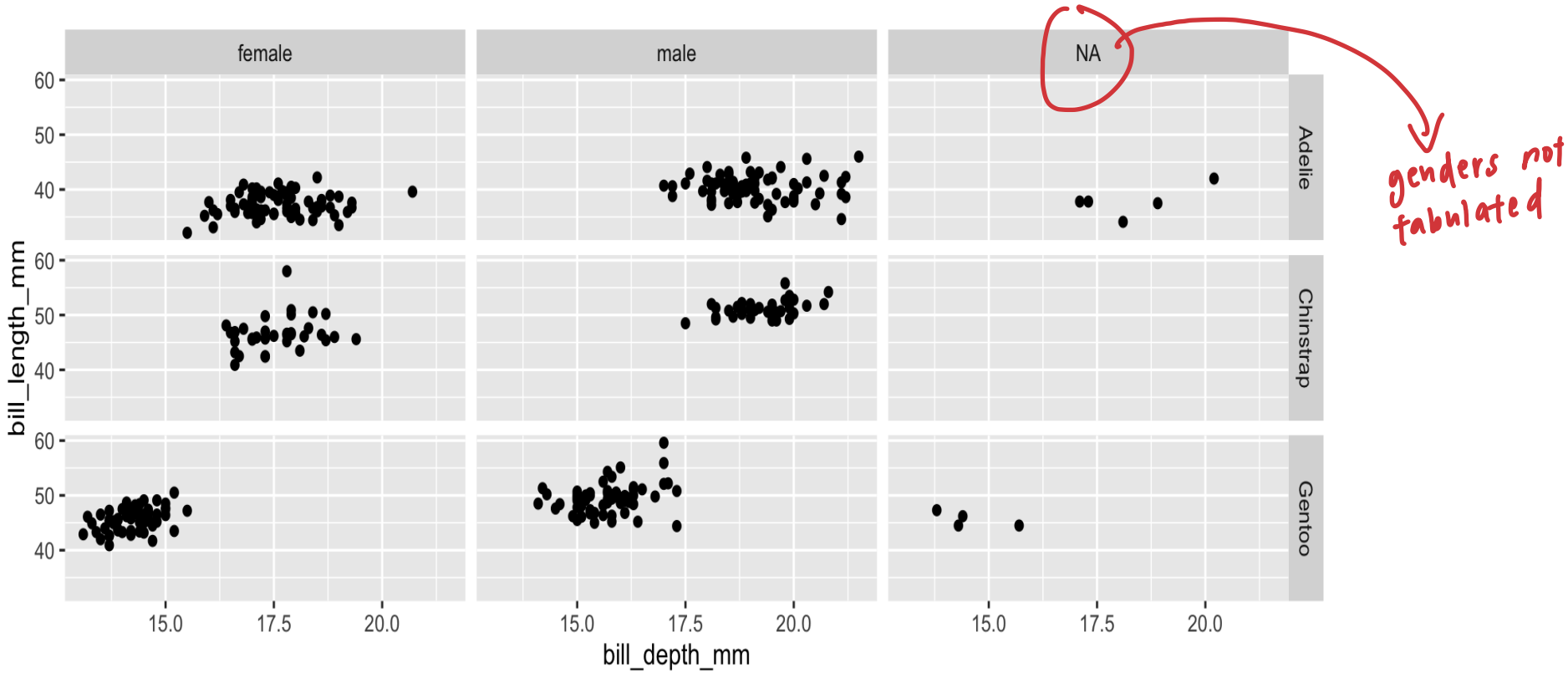
islands

species



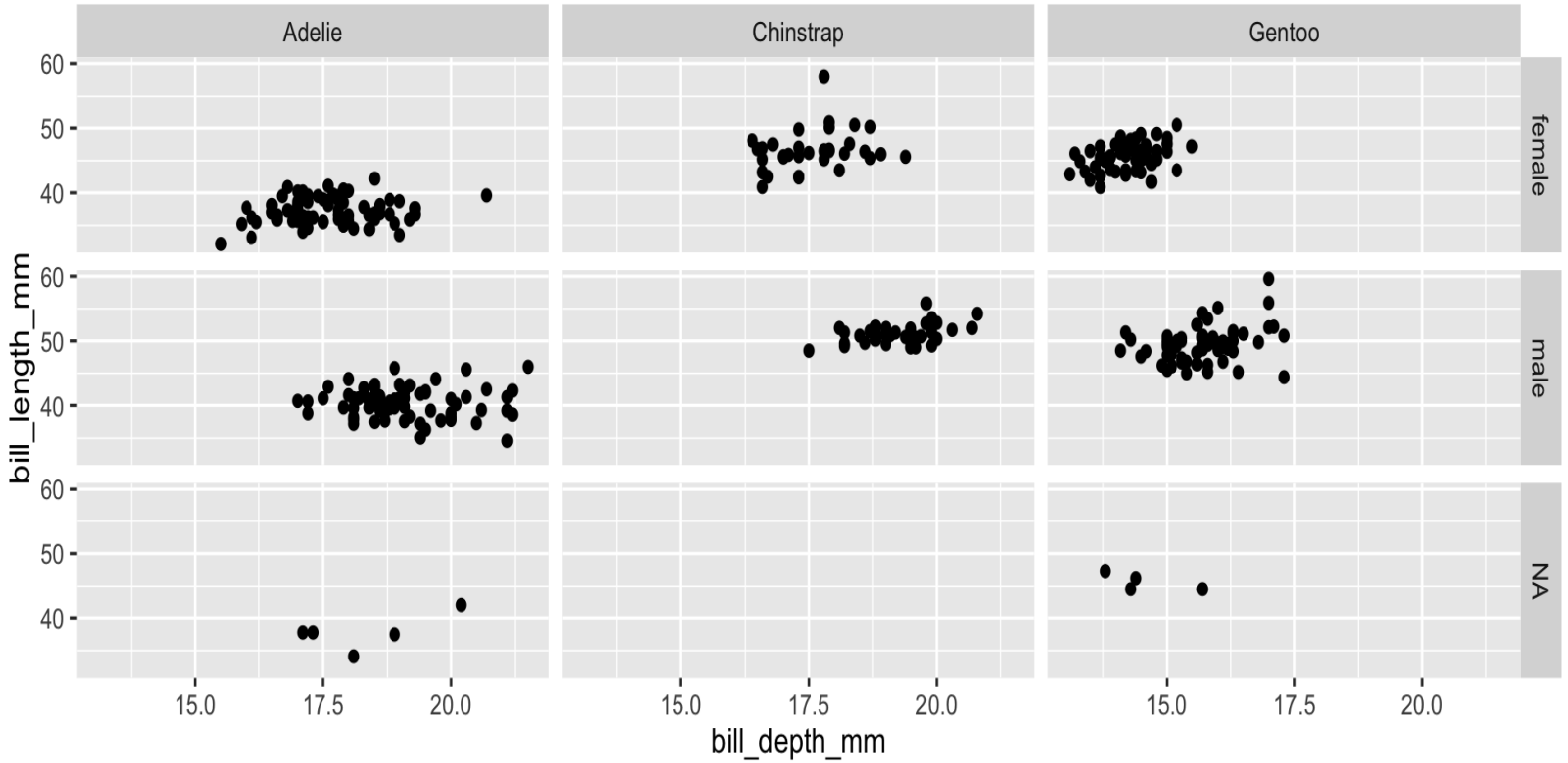
Facet 2

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm)) + geom_point() +
  facet_grid(species ~ sex)
```



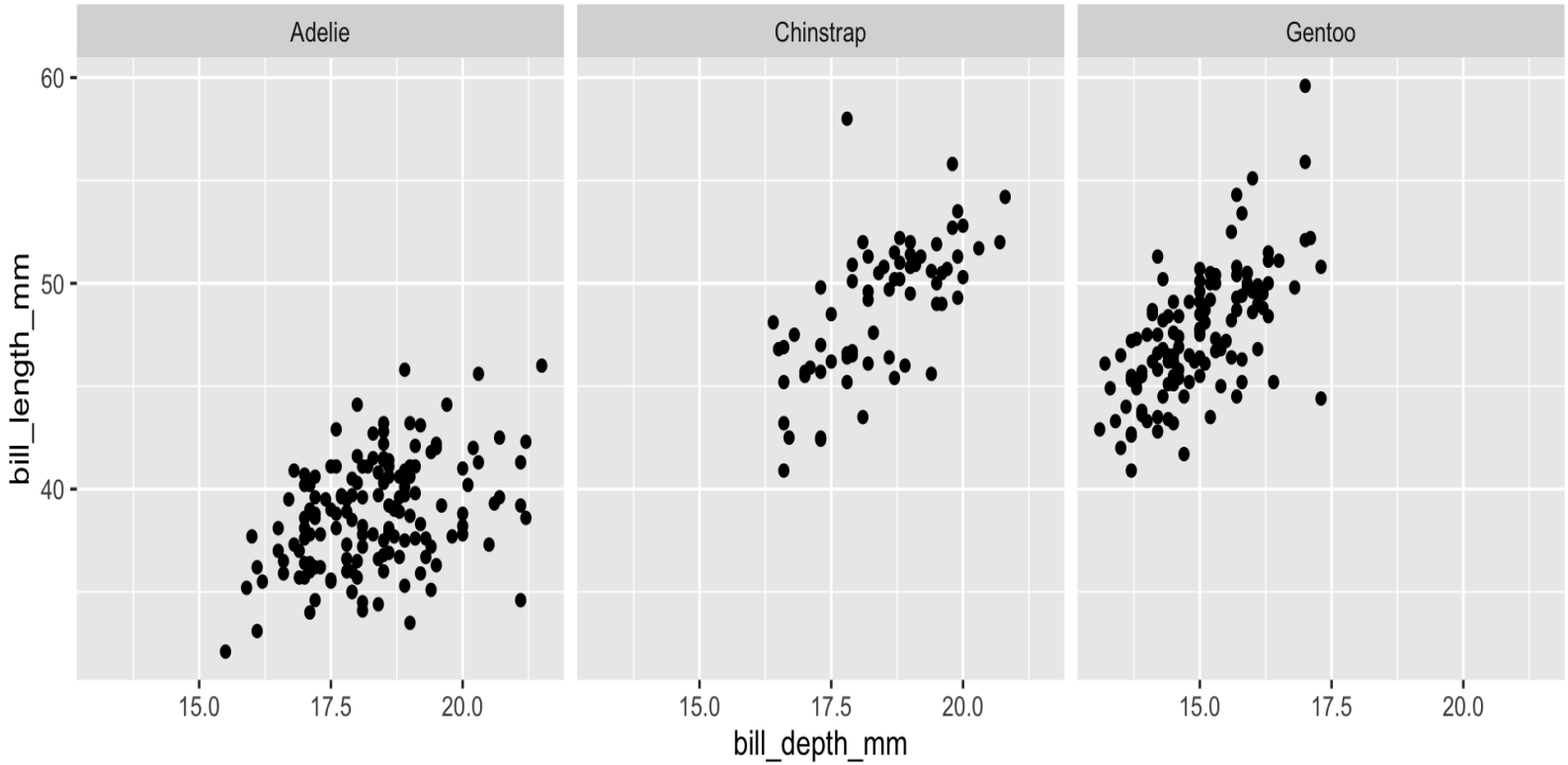
Facet 3

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm)) + geom_point() +
  facet_grid(sex ~ species)
```



Facet 4

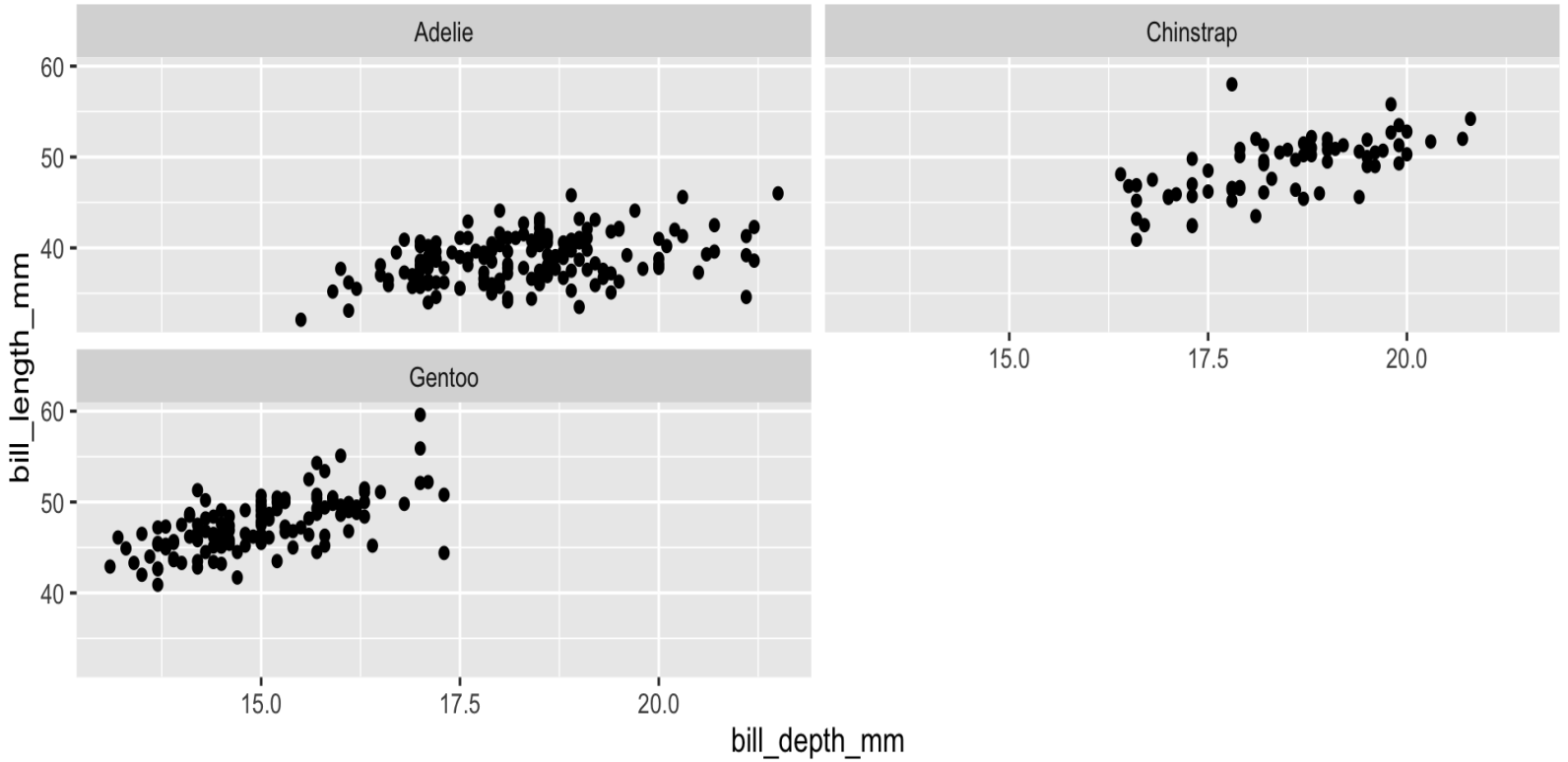
```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm)) + geom_point() +
  facet_wrap(~ species)
```



Facet 5

2 columns

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm)) + geom_point() +
  facet_wrap(~ species, ncol = 2)
```

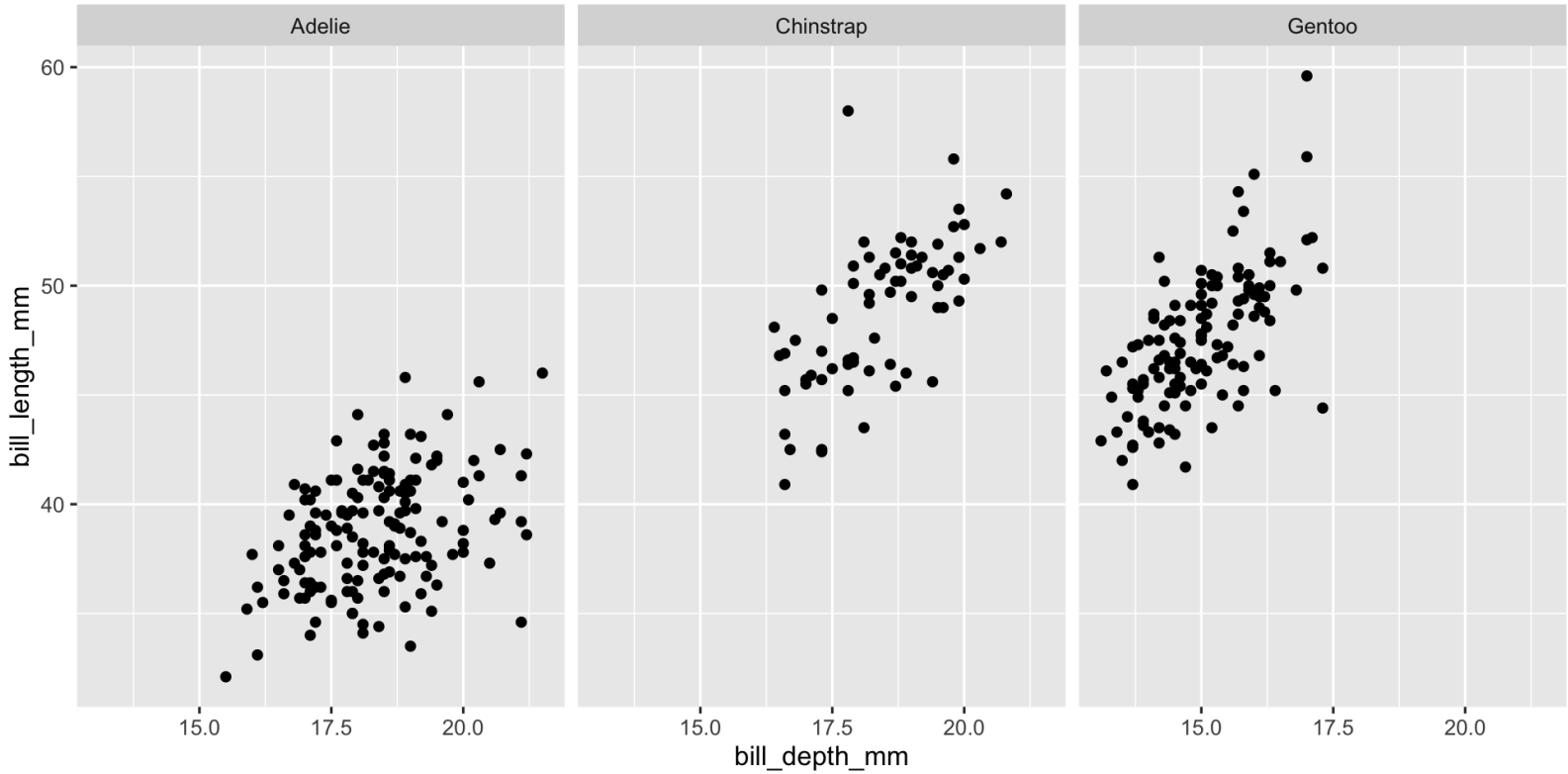


Facet 6

facet_grid = will produce a grid of plots for each combination of variables that you specify, even if some plots are empty.

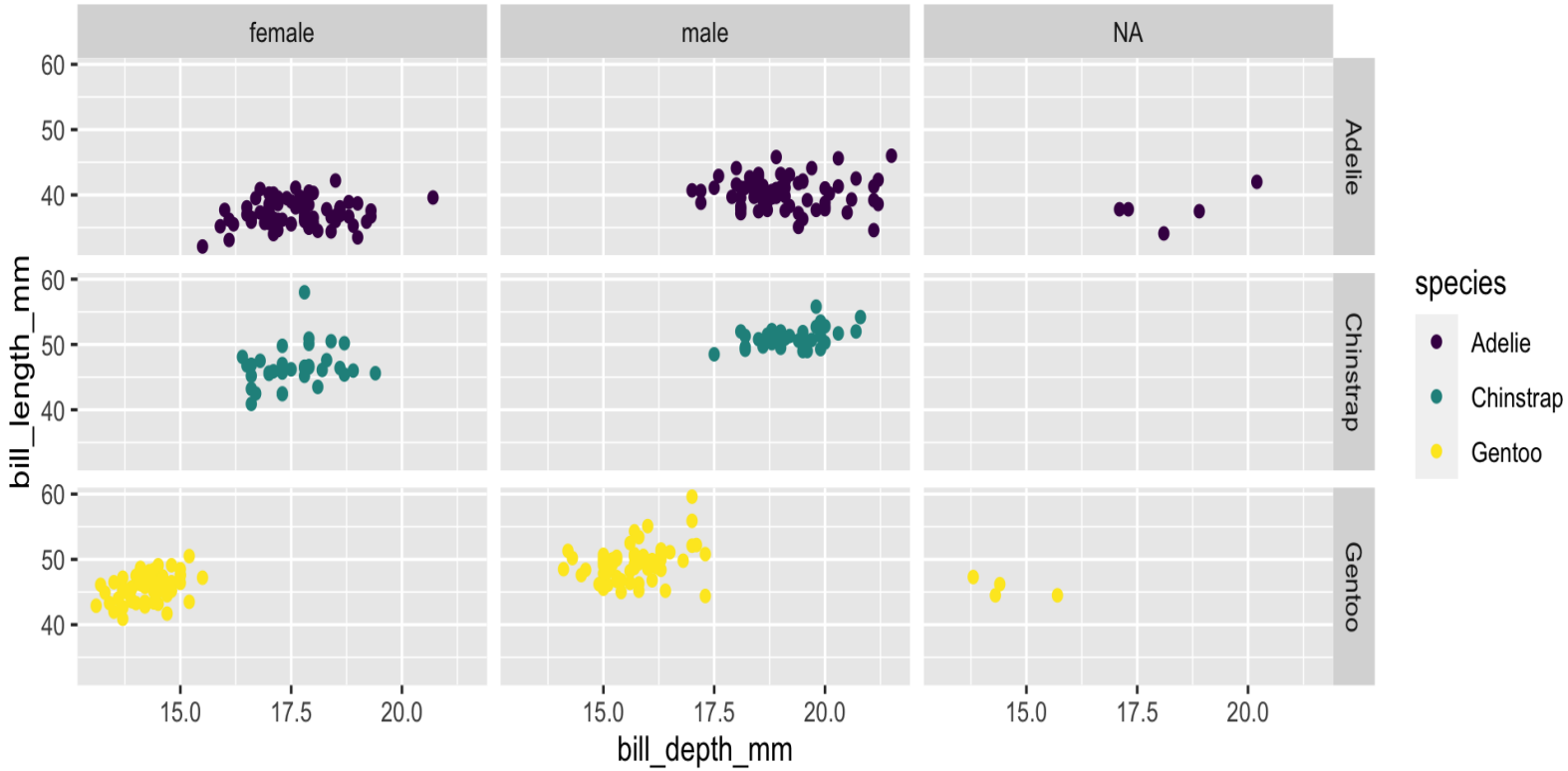
facet_wrap = will only produce plots for the combination of variables that have values, which means it will not produce any empty plots

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm)) + geom_point() +
  facet_grid(. ~ species)
```



Facet and color

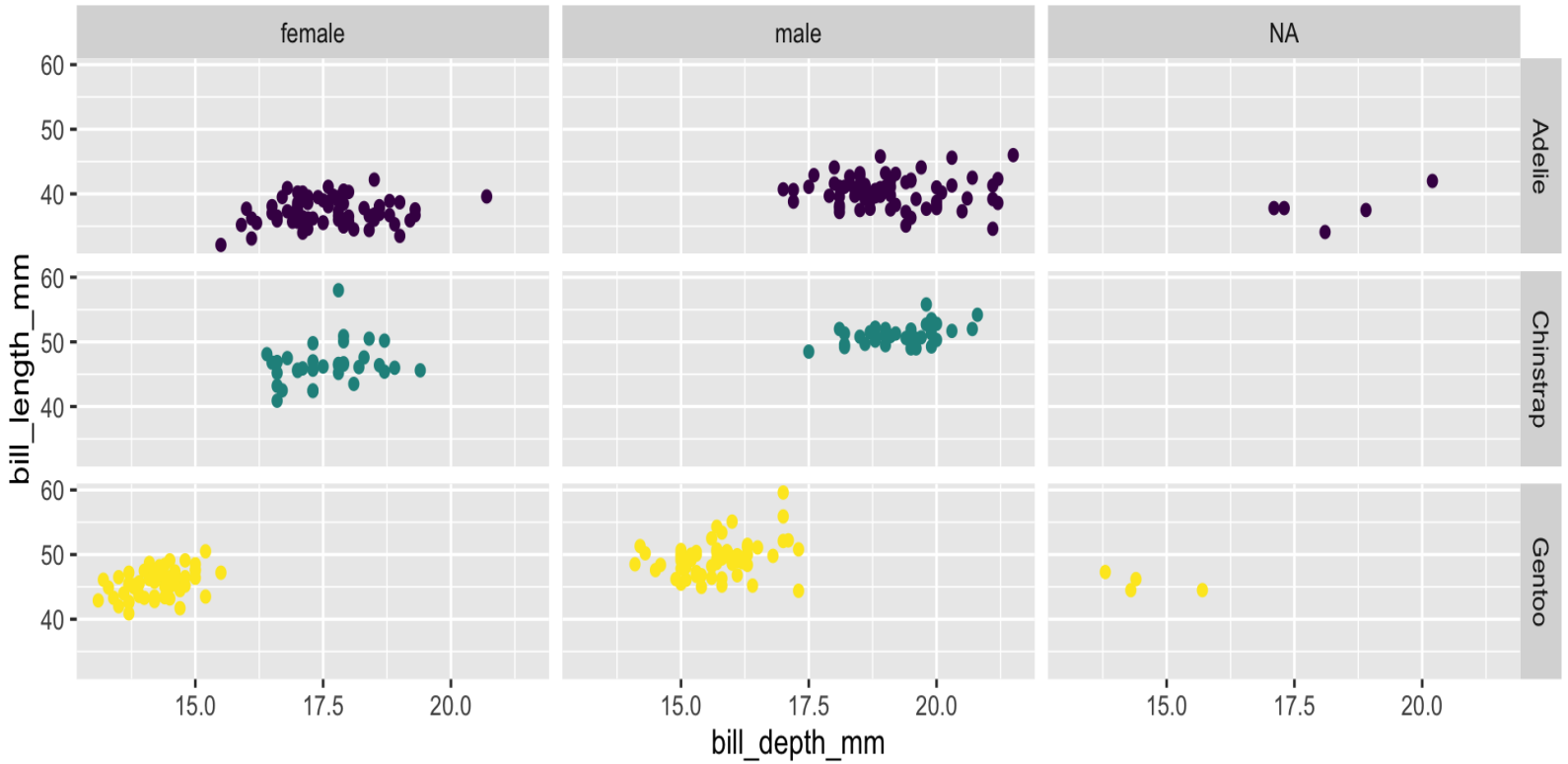
```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm, color = species)) +
  geom_point() + facet_grid(species ~ sex) + scale_color_viridis_d()
```



Face and color, no legend

```
ggplot(penguins, aes(x = bill_depth_mm, y = bill_length_mm, color = species)) +
  geom_point() + facet_grid(species ~ sex) + scale_color_viridis_d() +
  guides(color = "none")
```

no legend



II. Visualizing numeric variables

Dataset 2: Lending Club

- Thousands of loans made through the Lending Club, which is a platform that allows individuals to lend to other individuals
- Not all loans are created equal -- ease of getting a loan depends on (apparent) ability to pay back the loan
- Data includes loans *made*, these are not loan applications

Take a peek at data

```
library(openintro)
glimpse(loans_full_schema)
```

```
## Rows: 10,000
## Columns: 55
## $ emp_title                    <chr> "global config engineer ", "warehouse…
## $ emp_length                   <dbl> 3, 10, 3, 1, 10, NA, 10, 10, 10, 3, 1…
## $ state                        <fct> NJ, HI, WI, PA, CA, KY, MI, AZ, NV, I…
## $ homeownership                <fct> MORTGAGE, RENT, RENT, RENT, RENT, OWN…
## $ annual_income                <dbl> 90000, 40000, 40000, 30000, 35000, 34…
## $ verified_income              <fct> Verified, Not Verified, Source Verifi…
## $ debt_to_income               <dbl> 18.01, 5.04, 21.15, 10.16, 57.96, 6.4…
## $ annual_income_joint          <dbl> NA, NA, NA, NA, 57000, NA, 155000, NA…
## $ verification_income_joint    <fct> , , , , Verified, , Not Verified, , ,…
## $ debt_to_income_joint         <dbl> NA, NA, NA, NA, 37.66, NA, 13.12, NA,…
## $ delinq_2y                    <int> 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0…
## $ months_since_last_delinq     <int> 38, NA, 28, NA, NA, 3, NA, 19, 18, NA…
## $ earliest_credit_line         <dbl> 2001, 1996, 2006, 2007, 2008, 1990, 2…
## $ inquiries_last_12m           <int> 6, 1, 4, 0, 7, 6, 1, 1, 3, 0, 4, 4, 8…
## $ total_credit_lines           <int> 28, 30, 31, 4, 22, 32, 12, 30, 35, 9,…
## $ open_credit_lines            <int> 10, 14, 10, 4, 16, 12, 10, 15, 21, 6,…
## $ total_credit_limit           <int> 70795, 28800, 24193, 25400, 69839, 42…
## $ total_credit_utilized        <int> 38767, 4321, 16000, 4997, 52722, 3898…
```

Selected variables

```
loans <- loans_full_schema %>%
  select(loan_amount, interest_rate, term, grade,
         state, annual_income, homeownership, debt_to_income)
glimpse(loans)
```

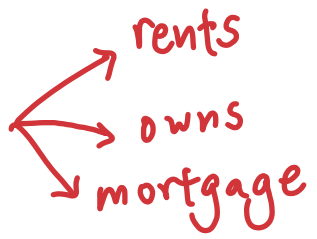
```
## Rows: 10,000
## Columns: 8
## $ loan_amount    <int> 28000, 5000, 2000, 21600, 23000, 5000, 24000, 20000, 20…
## $ interest_rate  <dbl> 14.07, 12.61, 17.09, 6.72, 14.07, 6.72, 13.59, 11.99, 1…
## $ term           <dbl> 60, 36, 36, 36, 36, 36, 60, 60, 36, 36, 60, 60, 36, 60,…
## $ grade          <fct> C, C, D, A, C, A, C, B, C, A, C, B, C, B, D, D, D, F, E…
## $ state          <fct> NJ, HI, WI, PA, CA, KY, MI, AZ, NV, IL, IL, FL, SC, CO,…
## $ annual_income  <dbl> 90000, 40000, 40000, 30000, 35000, 34000, 35000, 110000…
## $ homeownership  <fct> MORTGAGE, RENT, RENT, RENT, RENT, OWN, MORTGAGE, MORTGA…
## $ debt_to_income <dbl> 18.01, 5.04, 21.15, 10.16, 57.96, 6.46, 23.66, 16.19, 3…
```

Selected variables

variable	description
`loan_amount`	Amount of the loan received, in US dollars
`interest_rate`	Interest rate on the loan, in an annual percentage
`term`	The length of the loan, which is always set as a whole number of months
`grade`	Loan grade, which takes a values A through G and represents the quality of the loan and its likelihood of being repaid
`state`	US state where the borrower resides
`annual_income`	Borrower's annual income, including any second income, in US dollars
`homeownership`	Indicates whether the person owns, owns but has a mortgage, or rents
`debt_to_income`	Debt-to-income ratio

Variable types

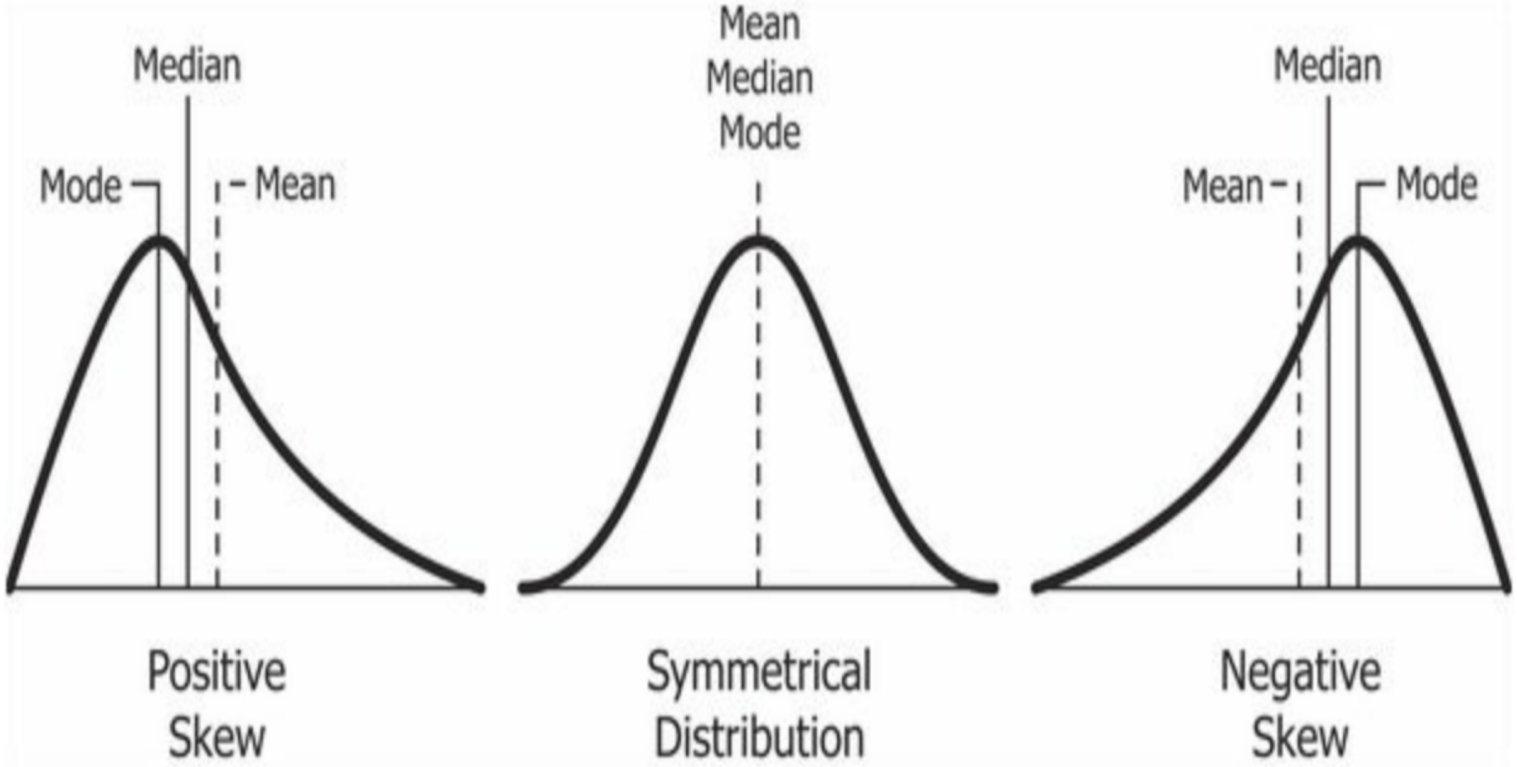
variable	type
`loan_amount`	numerical, continuous
`interest_rate`	numerical, continuous
`term`	numerical, discrete
`grade`	categorical, ordinal
`state`	categorical, not ordinal
`annual_income`	numerical, continuous
`homeownership`	categorical, not ordinal
`debt_to_income`	numerical, continuous



Numerical distributions

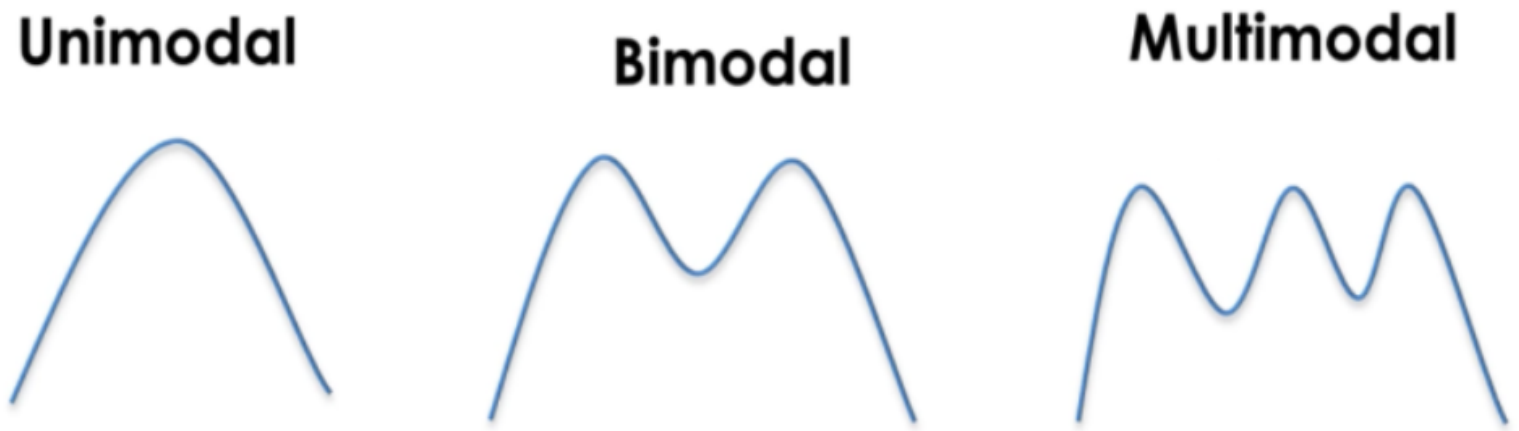
- **Shape**
  - Skewness: right-skewed, left-skewed, symmetric
  - Modality: unimodal, bimodal, multimodal, uniform
- **Center** mean (`mean`), median (`median`), mode (not always useful)
- **Spread** range (`range`), standard deviation (`sd`), inter-quartile range (`IQR`)
- unusual observations

Numerical distributions: Skewness



Source: Wikipedia

Numerical distributions: Modality

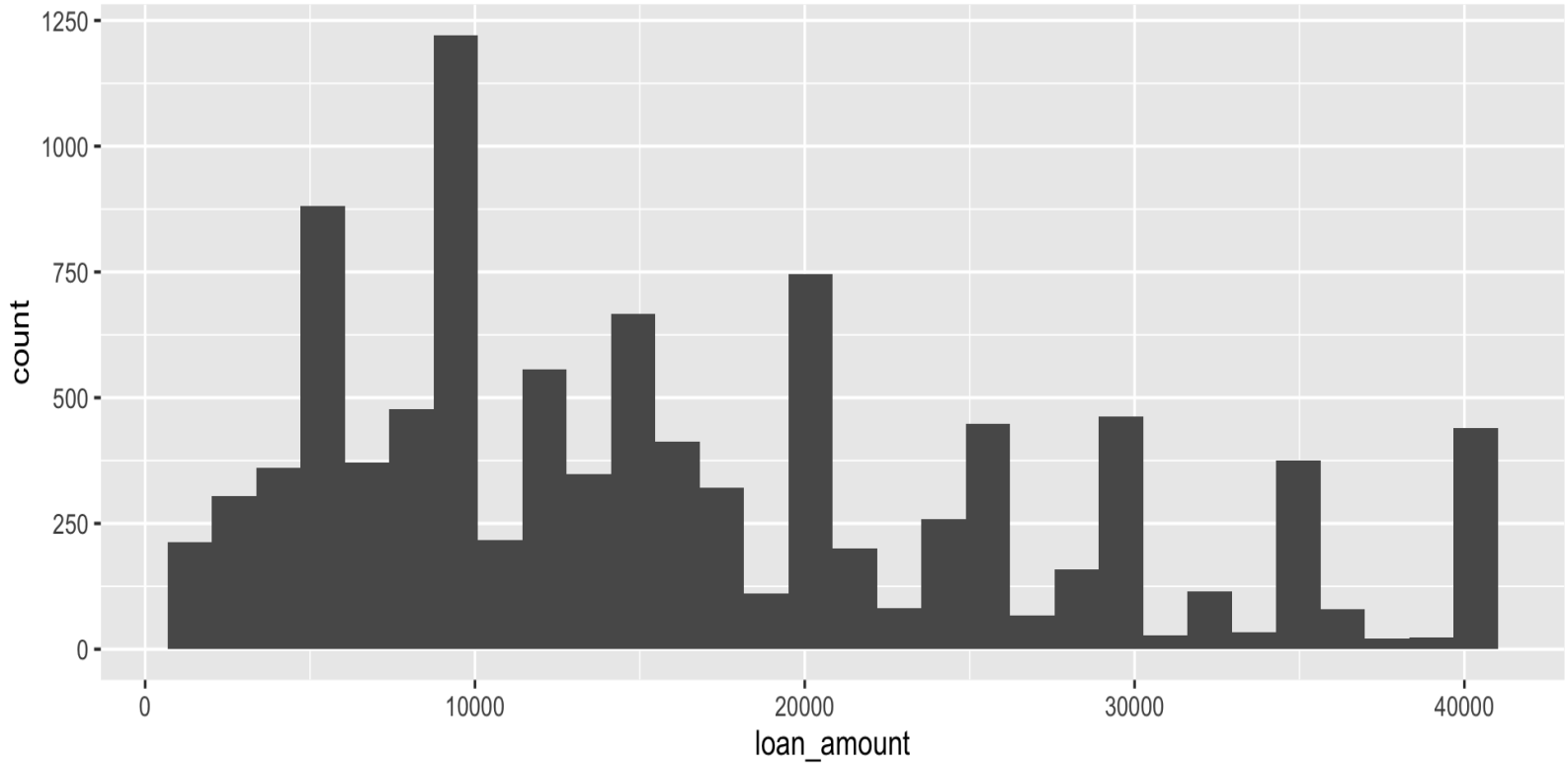


Source: https://builtin.com/data-science/intro-descriptive-statistics

Histogram

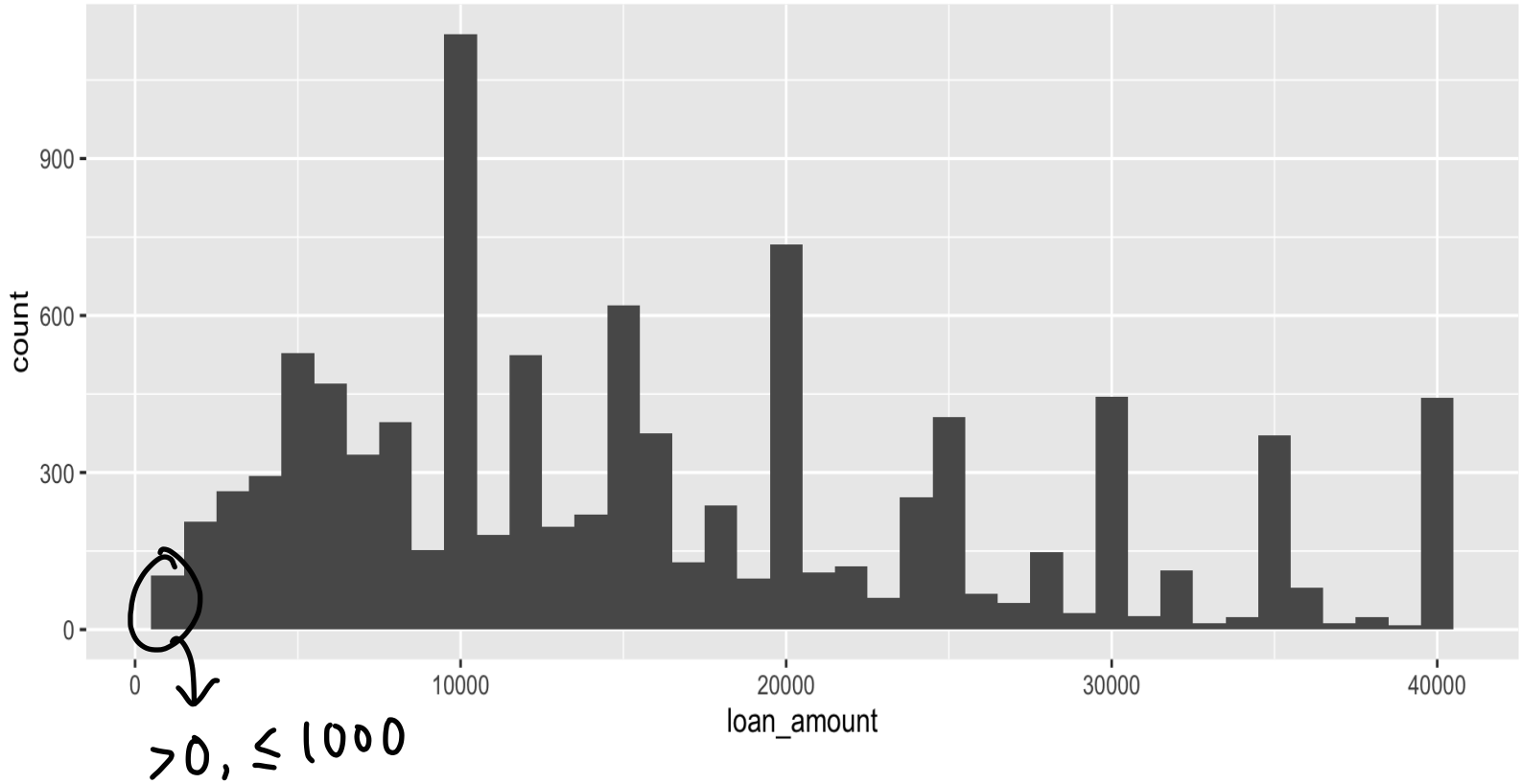
name of x-axis & y-axis

```
ggplot(loans) + aes(x = loan_amount) +
  geom_histogram()
```



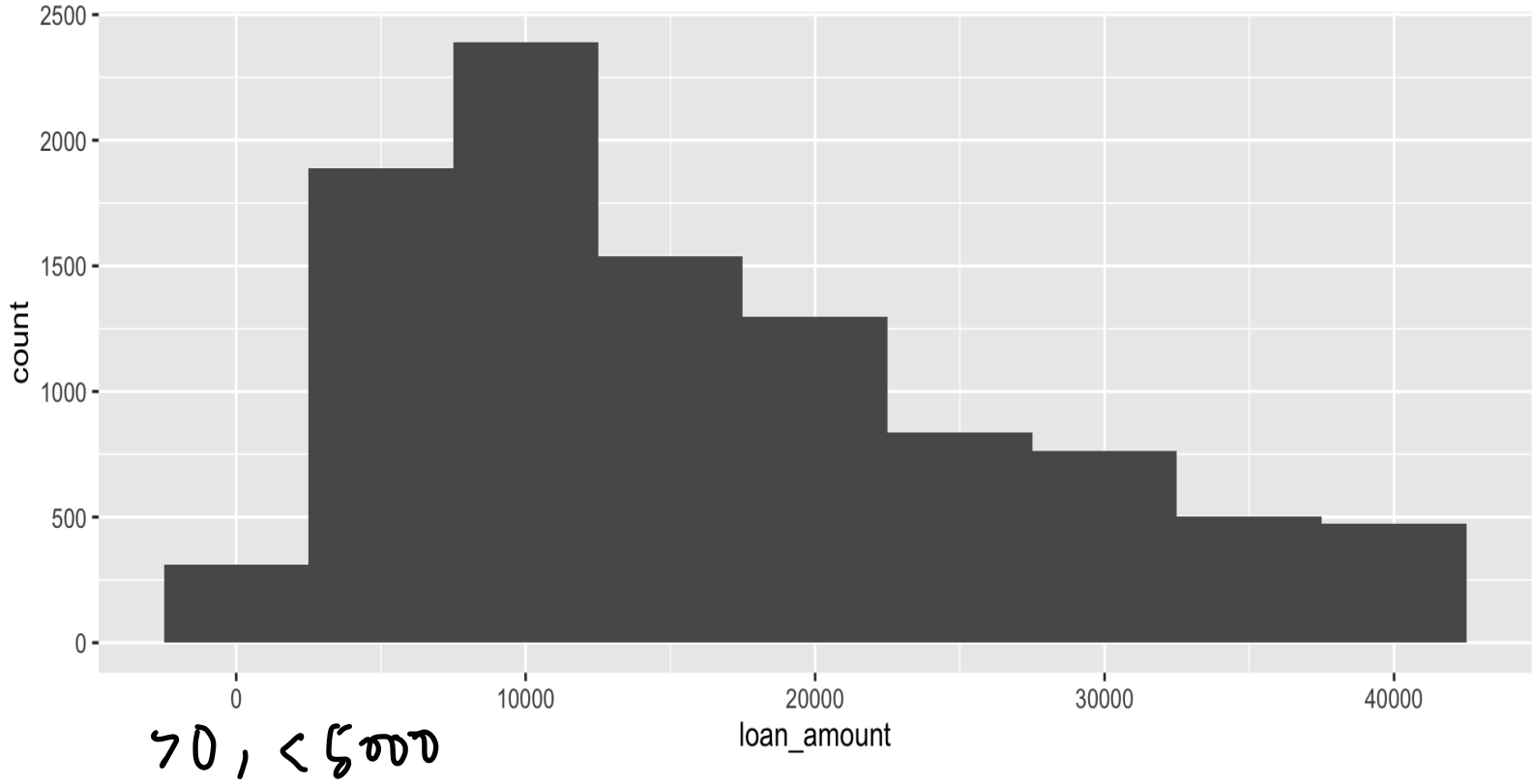
Histograms and `binwidth=1000`

```
# binwidth = 1000
ggplot(loans, aes(x = loan_amount)) +
  geom_histogram(binwidth = 1000)
```



Histograms and `binwidth = 5000`

```
# binwidth = 5000
ggplot(loans, aes(x = loan_amount)) +
  geom_histogram(binwidth = 5000)
```

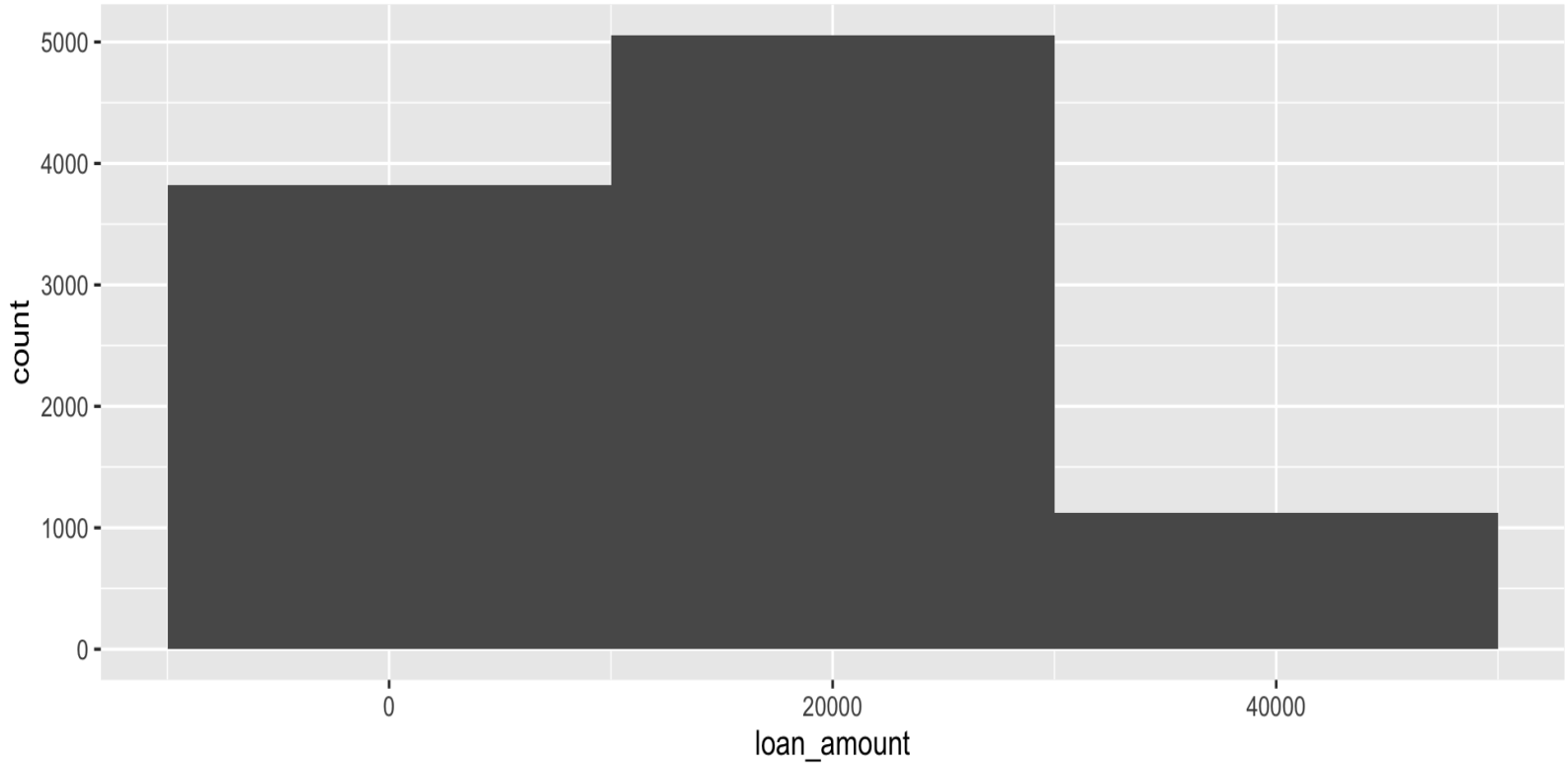


Histograms and `binwidth=20000`

subset

```
# binwidth = 20000
ggplot(loans, aes(x = loan_amount)) +
  geom_histogram(binwidth = 20000)
```

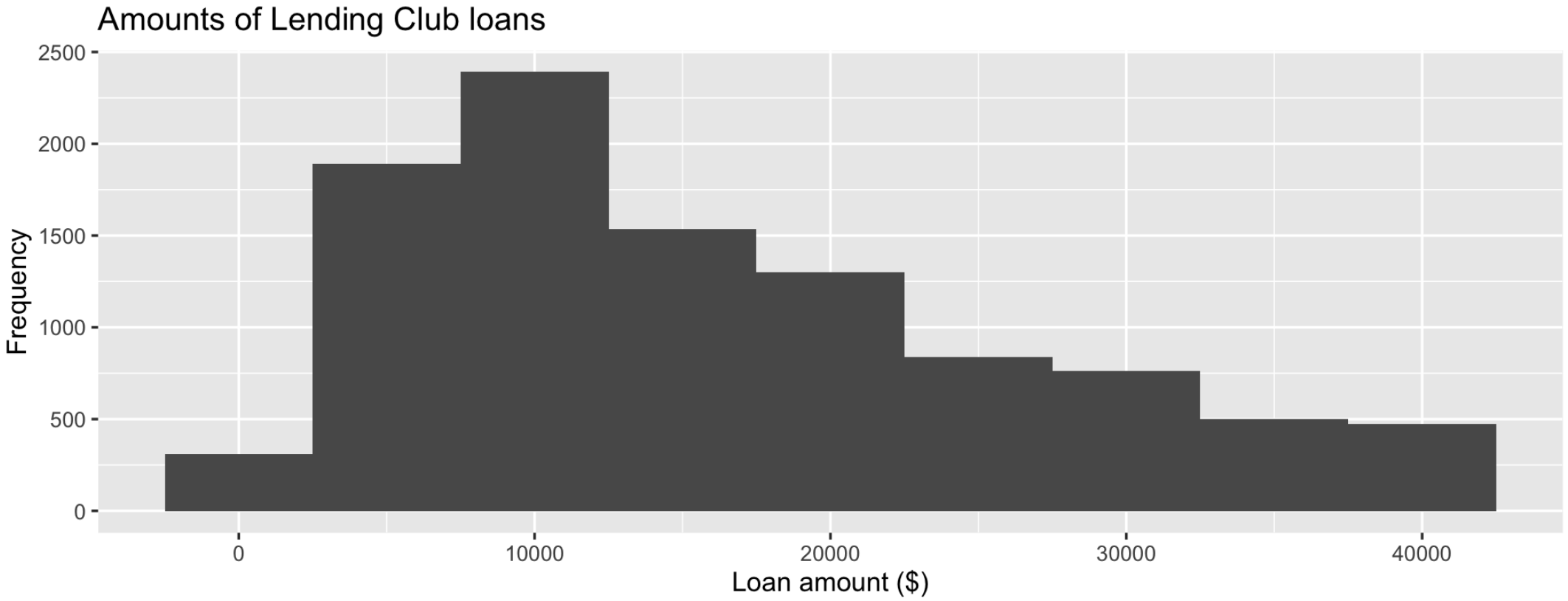
(applying a layer)



Customizing histograms

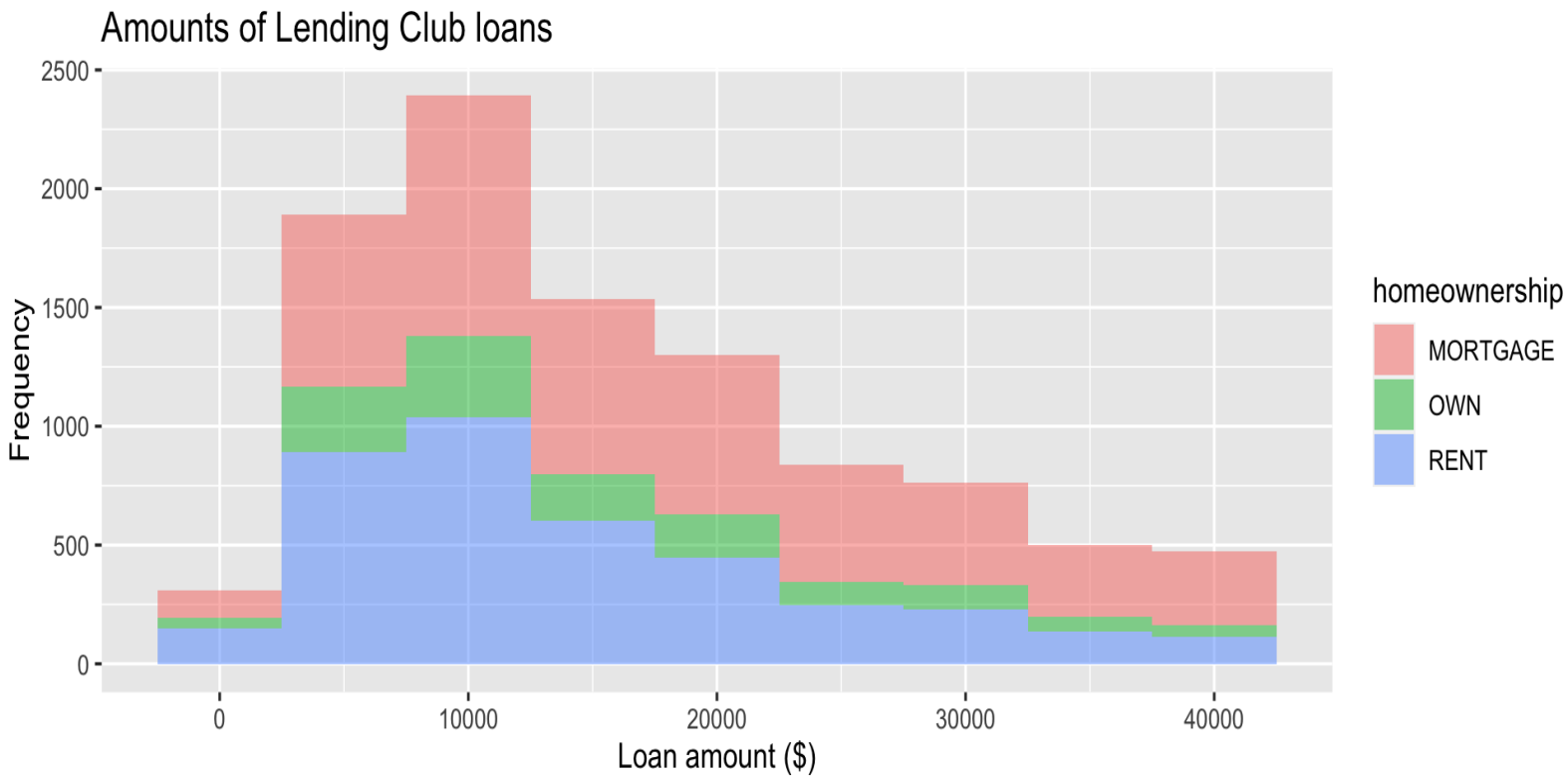
labels

```
ggplot(loans, aes(x = loan_amount)) + geom_histogram(binwidth = 5000) +
  labs(x = "Loan amount ($)", y = "Frequency", title = "Amounts of Lending Club loans" )
```



Fill with a categorical variable

```
ggplot(loans, aes(x = loan_amount, fill = homeownership)) +
  geom_histogram(binwidth = 5000, alpha = 0.5) +
  labs(x = "Loan amount ($)",y = "Frequency",title = "Amounts of Lending Club loans")
```

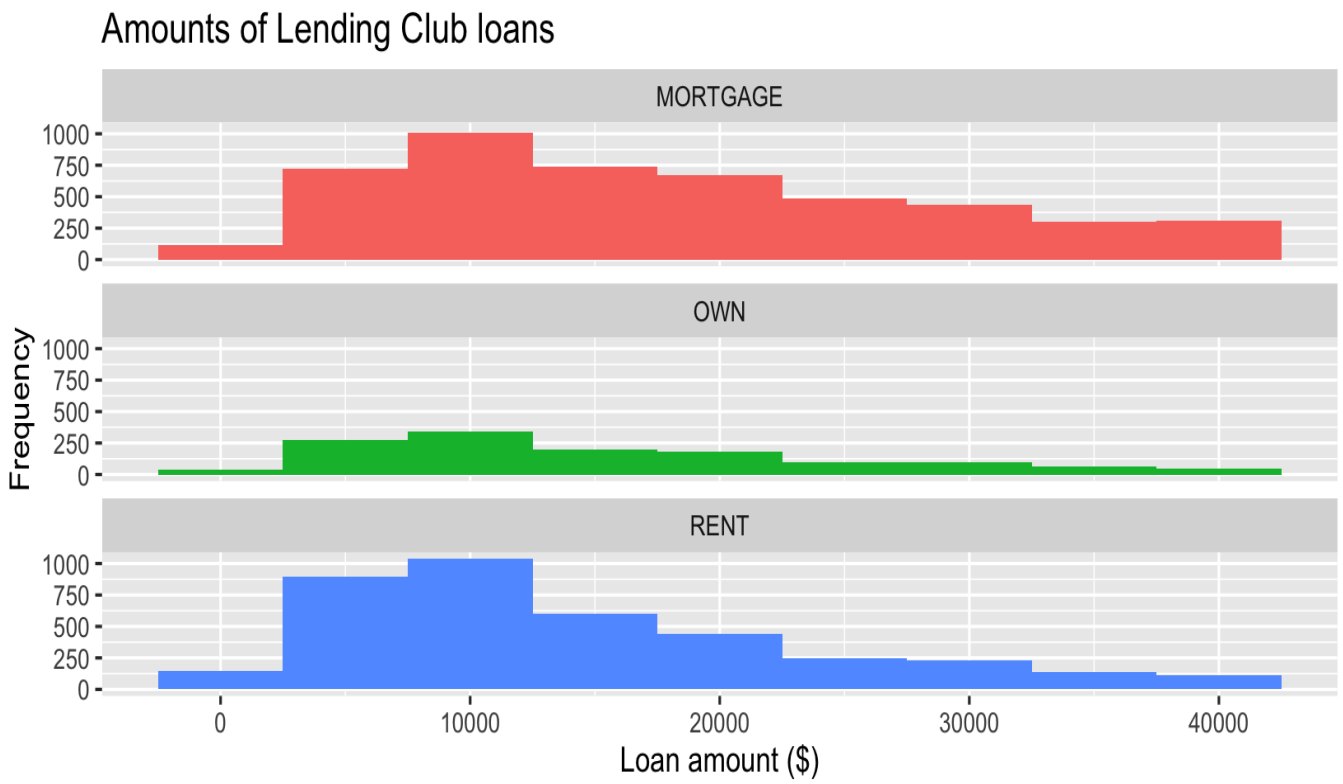


Facet with a categorical variable

define what type of graph you want

```
ggplot(loans, aes(x = loan_amount, fill = homeownership)) + geom_histogram(binwidth = 5000) +
  labs(x = "Loan amount ($)",y = "Frequency",title = "Amounts of Lending Club loans") +
  facet_wrap(~ homeownership, nrow = 3)
```

row



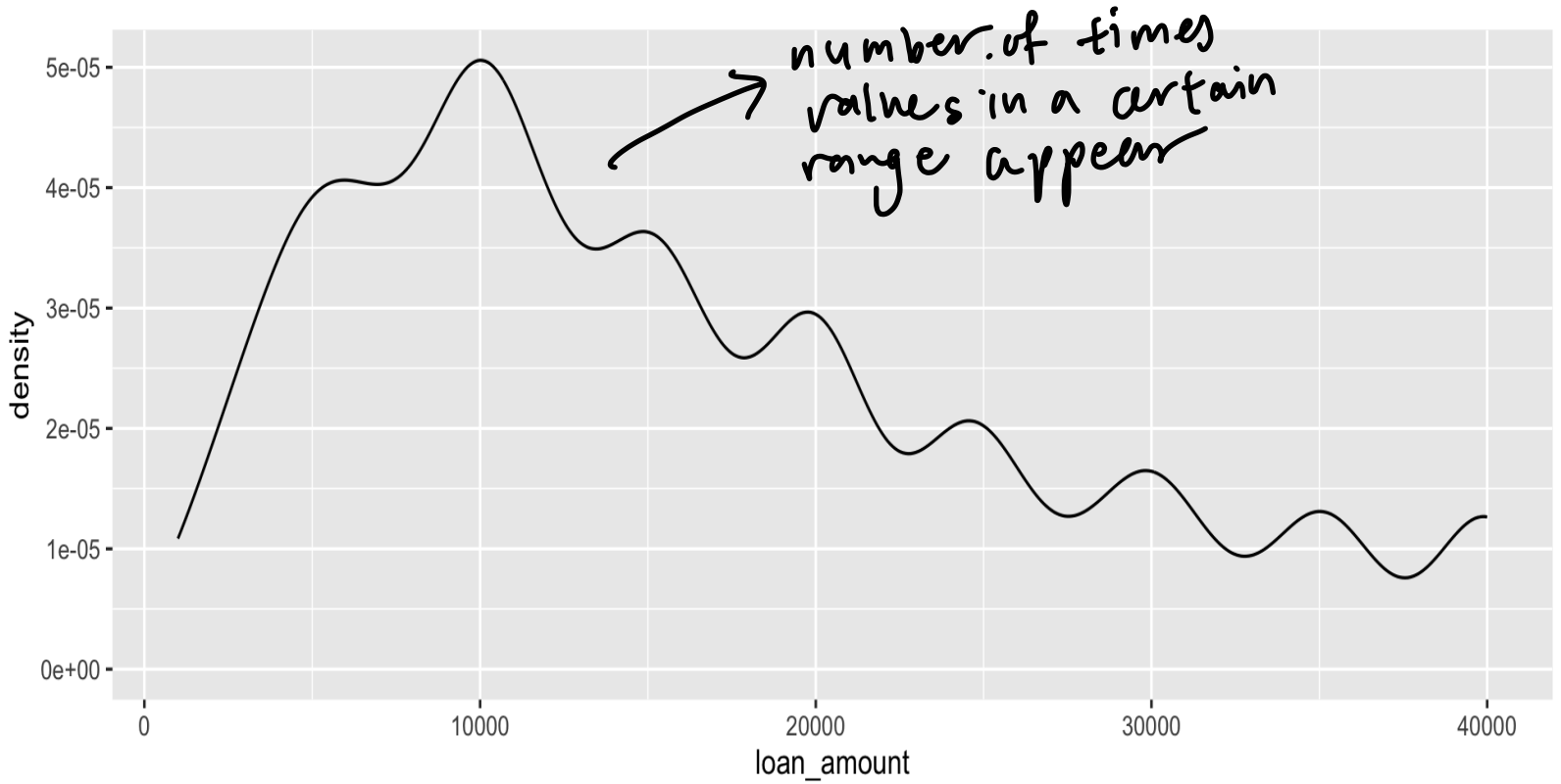
facet_wrap

↳ makes a long ribbon of panels

this is 1 variable, but with different "levels" ≈ categories since this is categorical

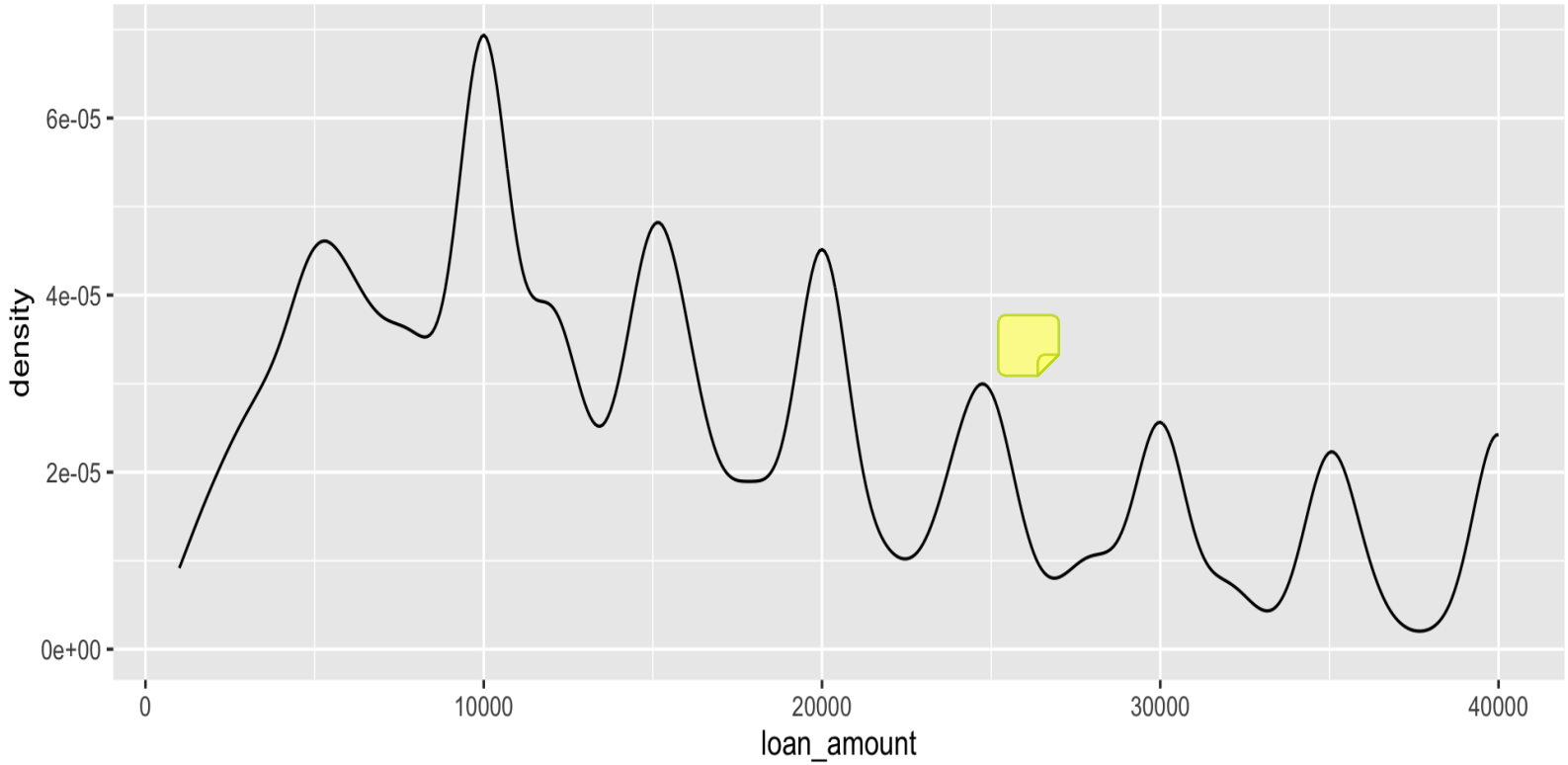
Density plot

```
ggplot(loans, aes(x = loan_amount)) +
  geom_density()
```



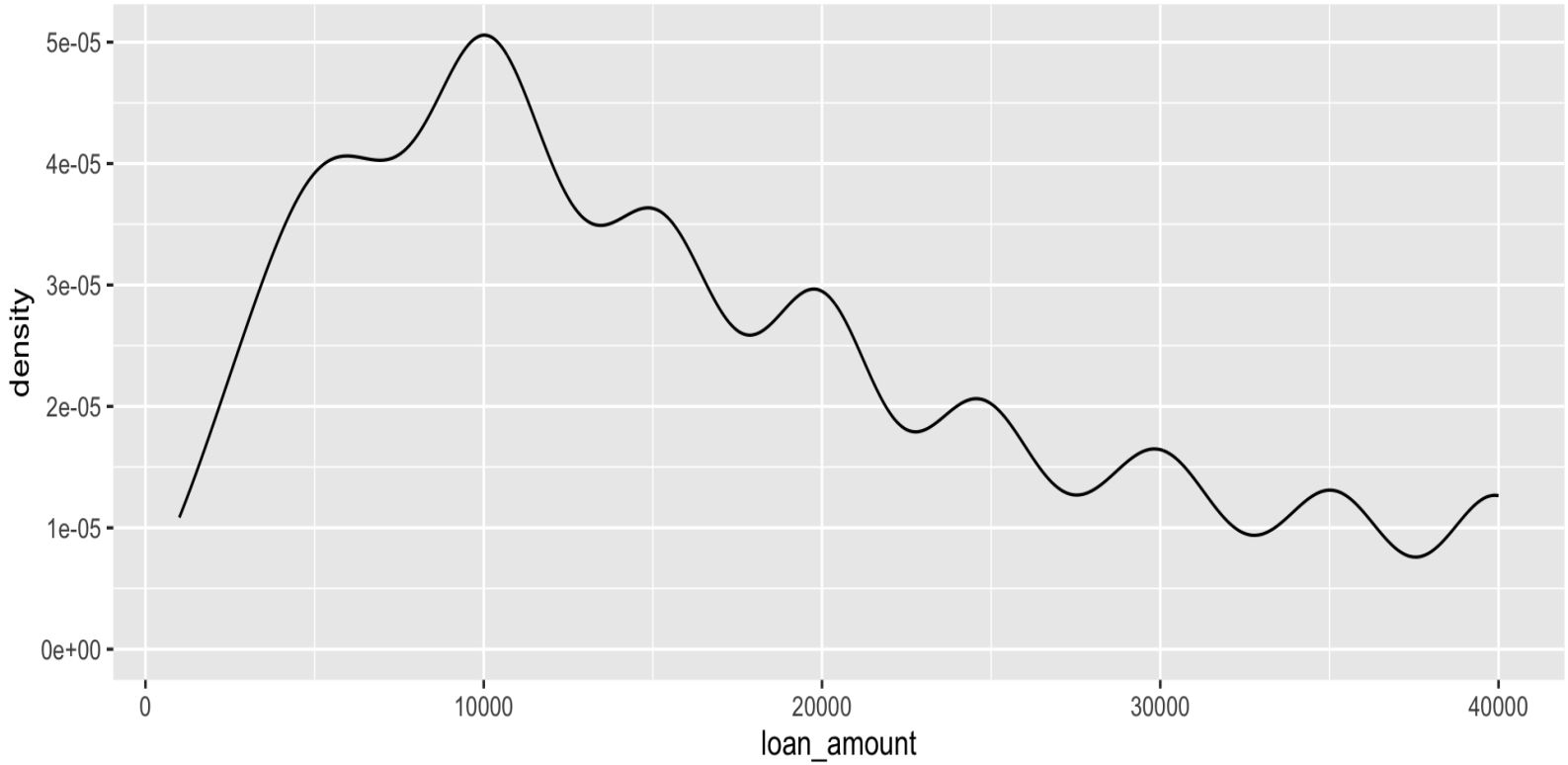
Density plots and adjusting bandwidth

```
ggplot(loans, aes(x = loan_amount)) +
  geom_density(adjust = 0.5)
```



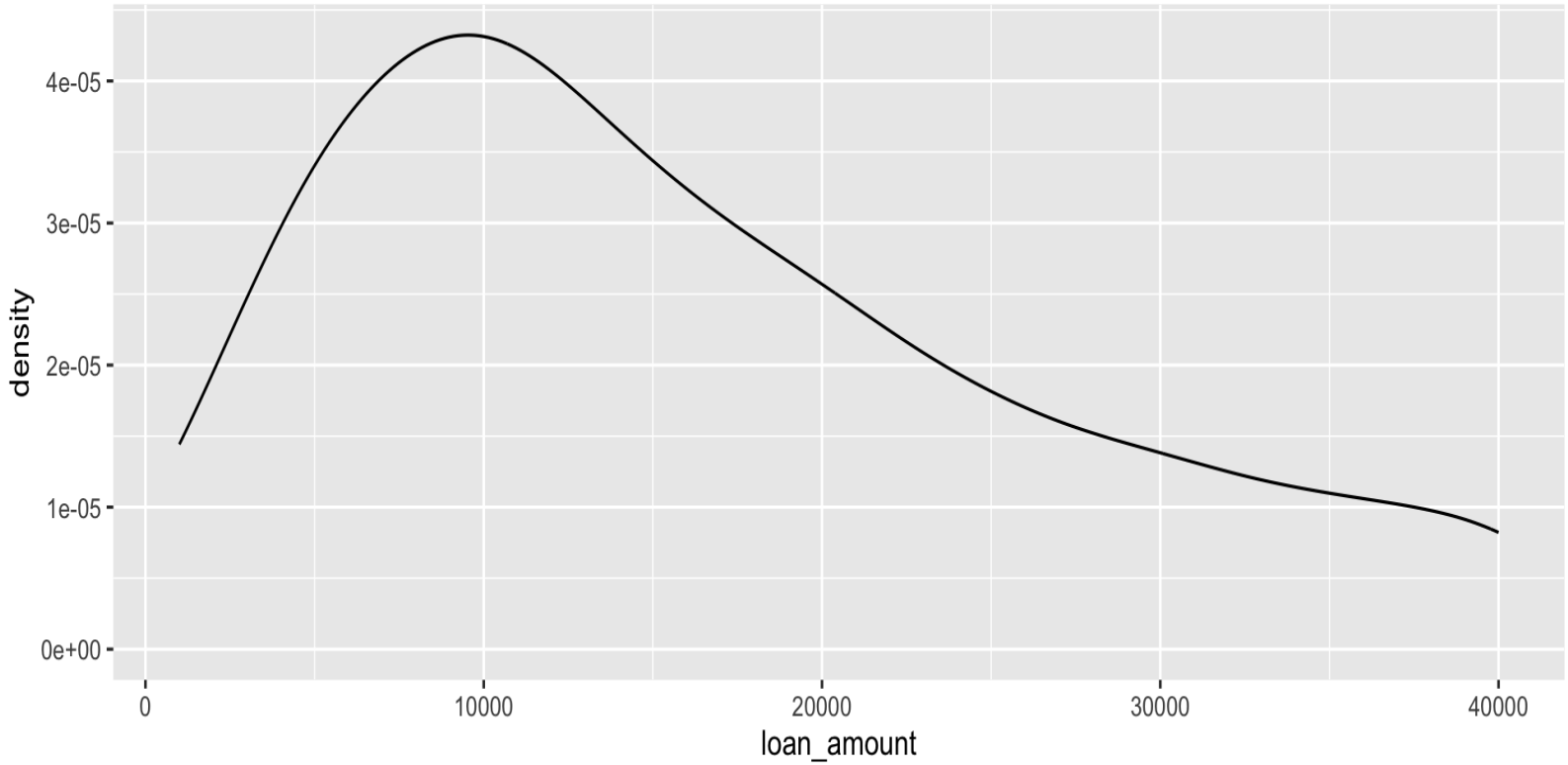
Density plots and adjusting bandwidth

```
ggplot(loans, aes(x = loan_amount)) +
  geom_density(adjust = 1) # default bandwidth
```



Density plots and adjusting bandwidth

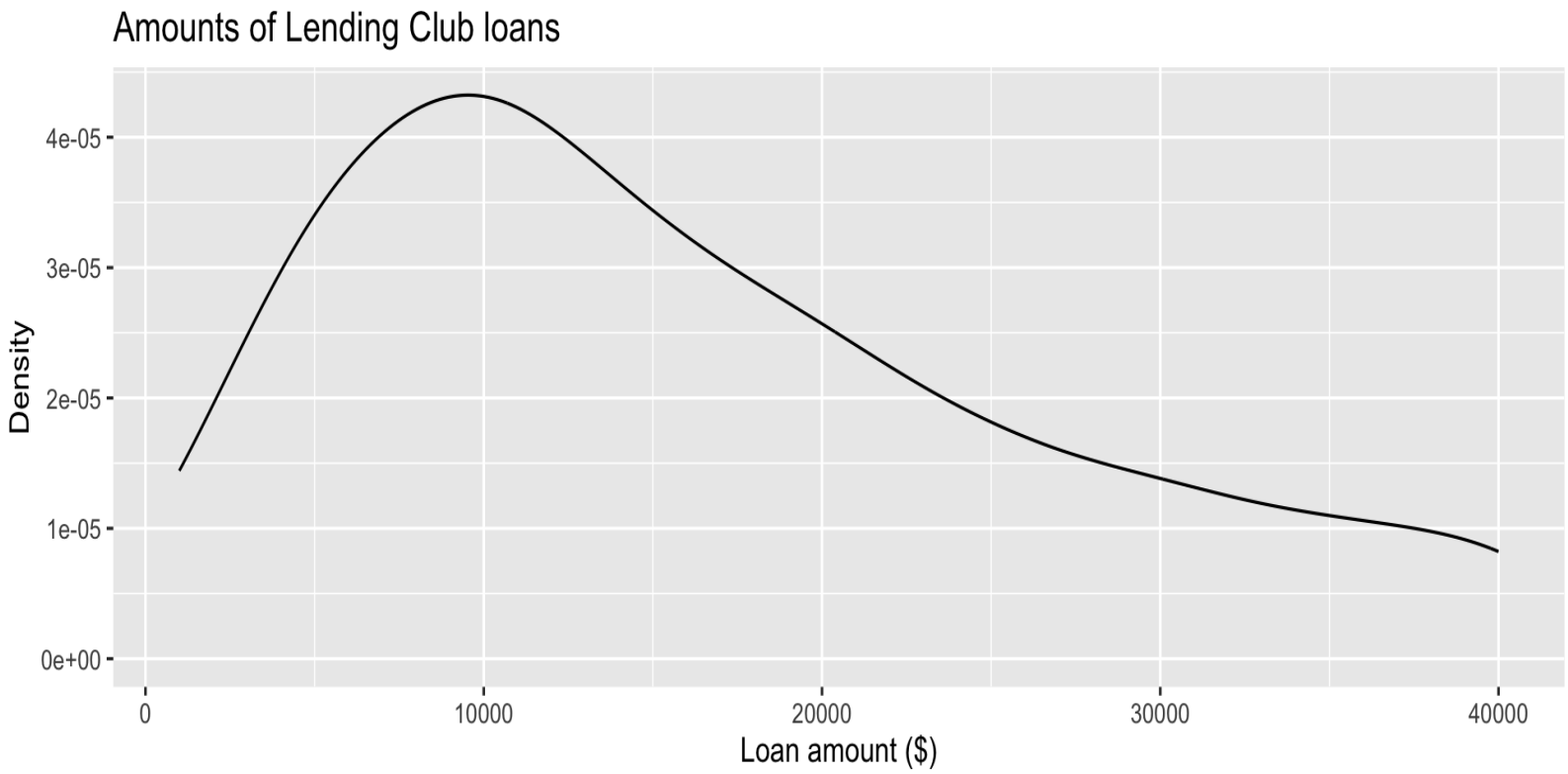
```
ggplot(loans, aes(x = loan_amount)) +
  geom_density(adjust = 2)
```



Customizing density plots

The higher the adjust, the smoother the curve but possibly not capture fluctuations

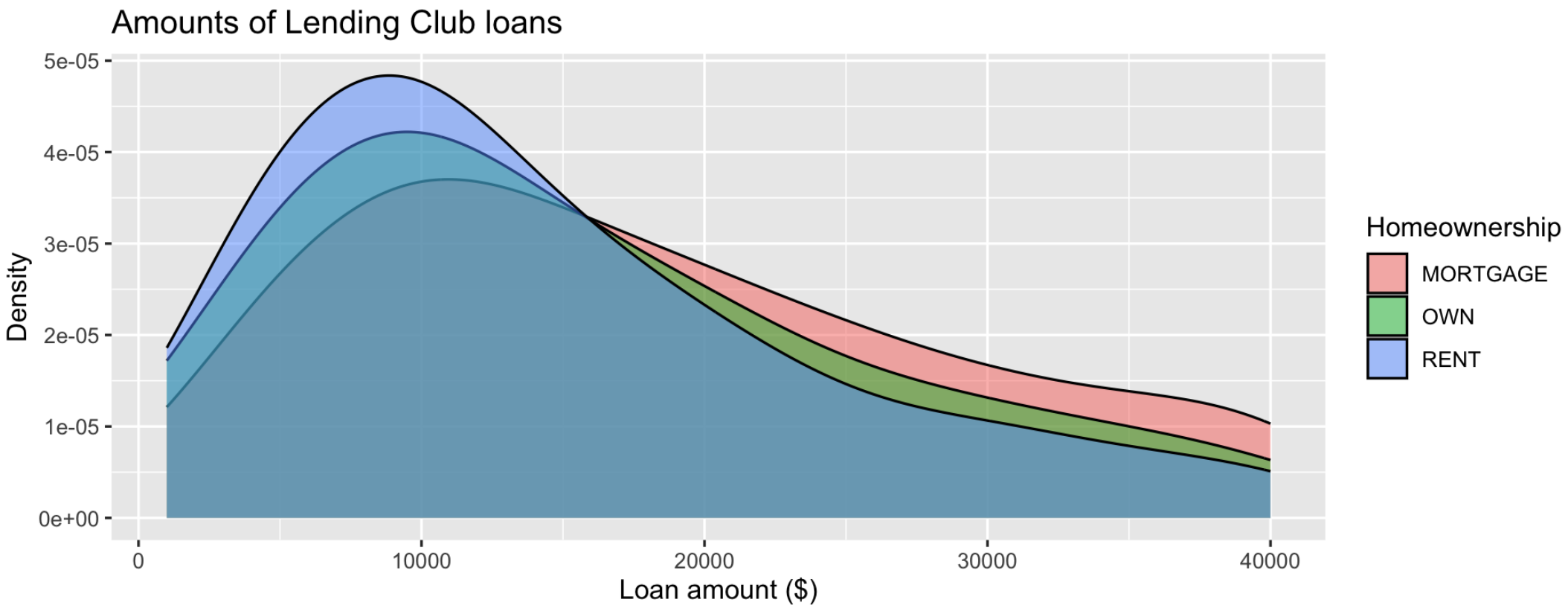
```
ggplot(loans, aes(x = loan_amount)) +
  geom_density(adjust = 2) +
  labs( x = "Loan amount ($)", y = "Density", title = "Amounts of Lending Club loans" )
```



Adding a categorical variable

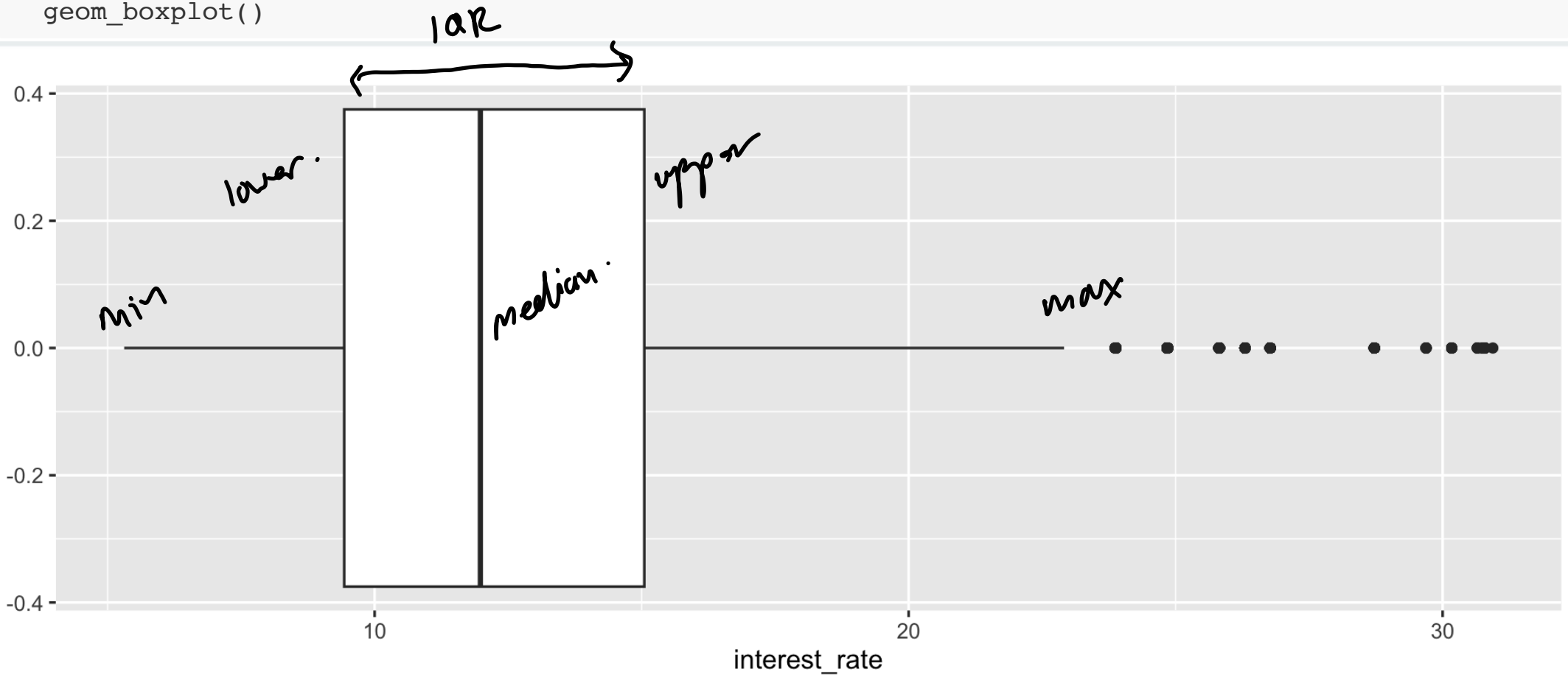
opacity of the fill (variable)

```
ggplot(loans, aes(x = loan_amount, fill = homeownership)) +
  geom_density(adjust = 2, alpha = 0.5) +
  labs(x = "Loan amount ($)",y = "Density",title = "Amounts of Lending Club loans", fill = "Homeow
```



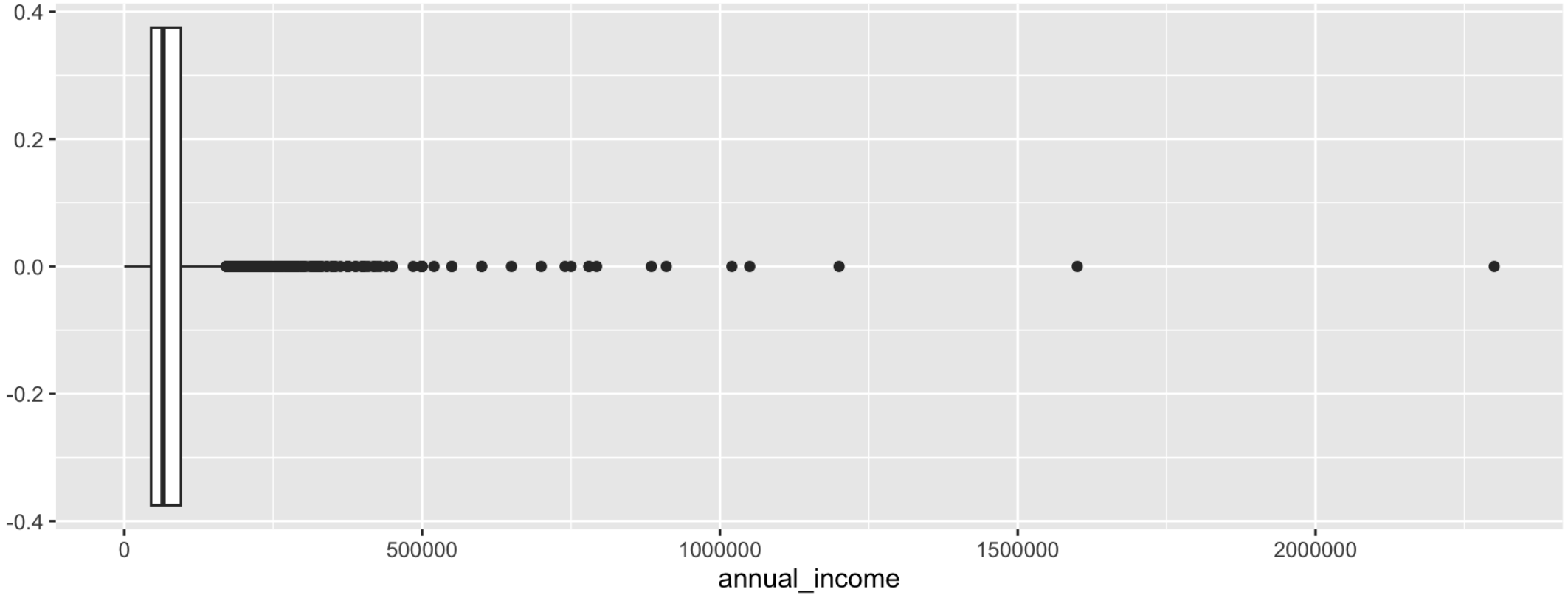
Box plot

```
ggplot(loans, aes(x = interest_rate)) +
  geom_boxplot()
```



Box plot and outliers

```
ggplot(loans, aes(x = annual_income)) +
  geom_boxplot()
```

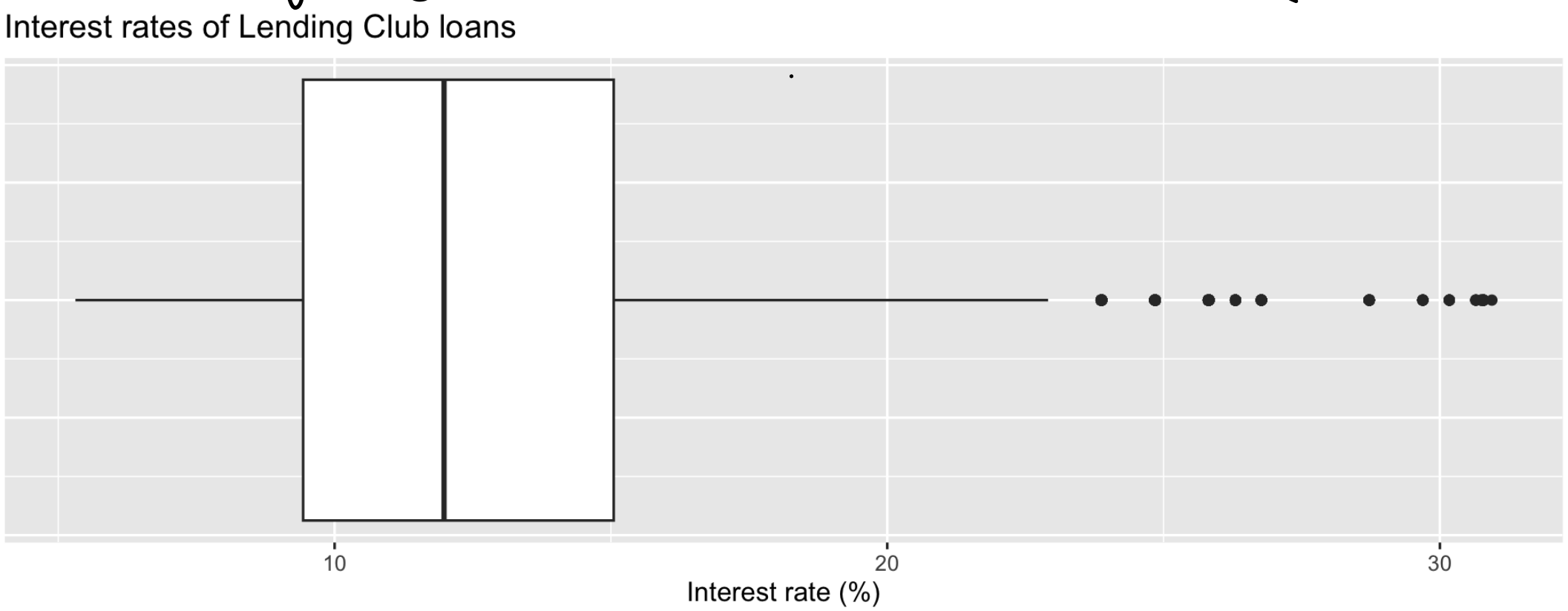


Customizing box plots

layer 1 layer 2

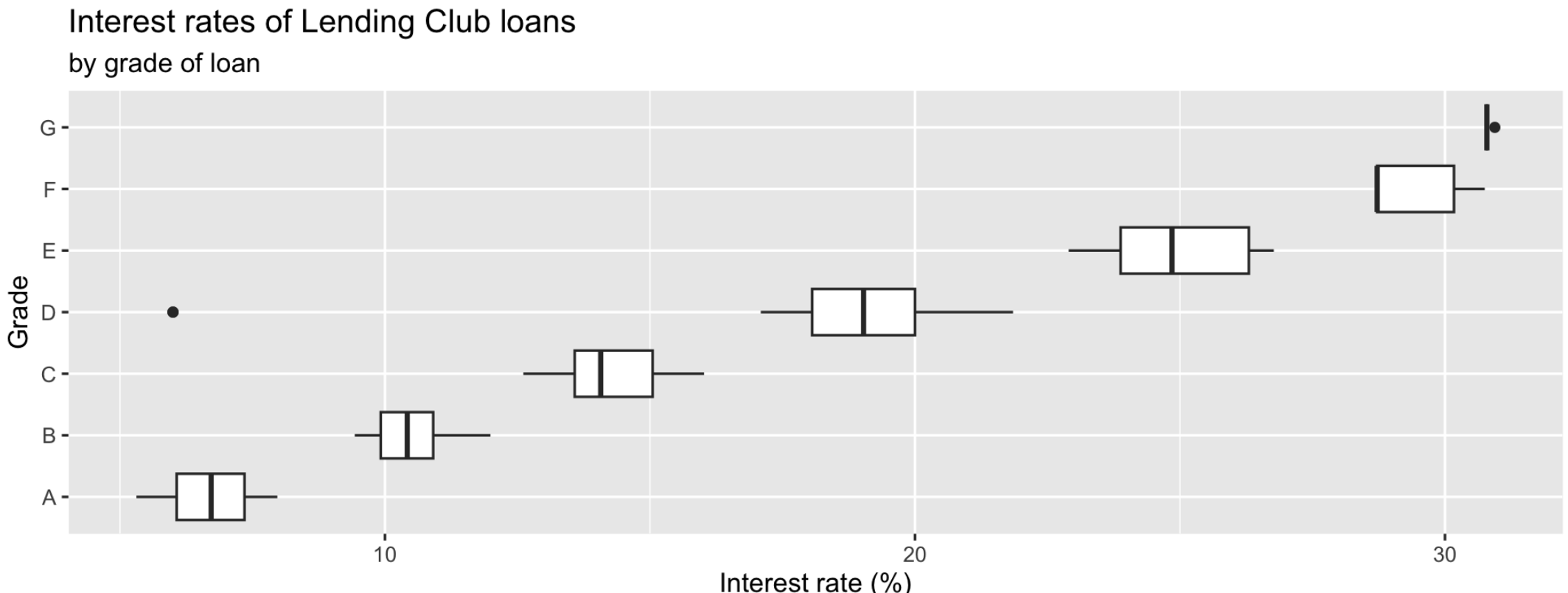
```
ggplot(loans, aes(x = interest_rate)) +geom_boxplot() +labs(x = "Interest rate (%)",y = NULL,
  title = "Interest rates of Lending Club loans") +
 theme( axis.ticks.y = element_blank(), axis.text.y = element_blank() )
```

layer 3



Adding a categoric variable

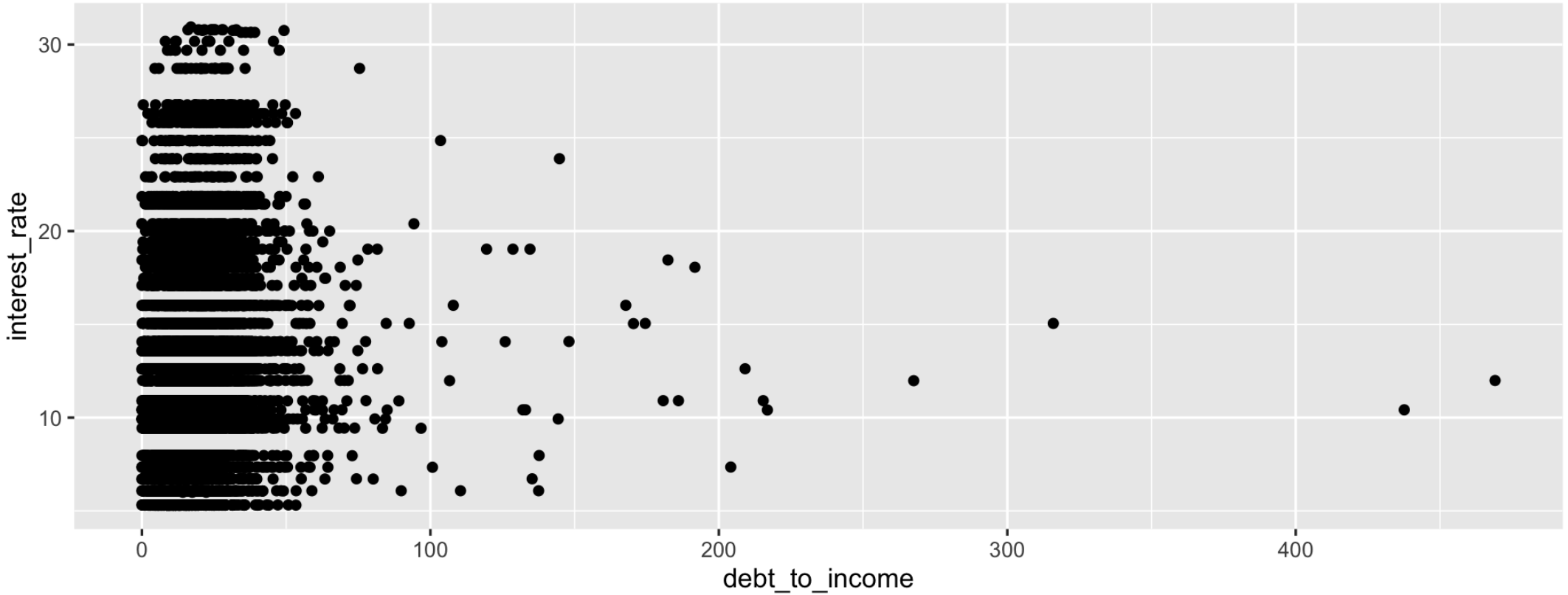
```
ggplot(loans, aes(x = interest_rate,
                  y = grade)) +
  geom_boxplot() +
  labs(x = "Interest rate (%)",y = "Grade",title = "Interest rates of Lending Club loans",subtitle
```



Scatterplot

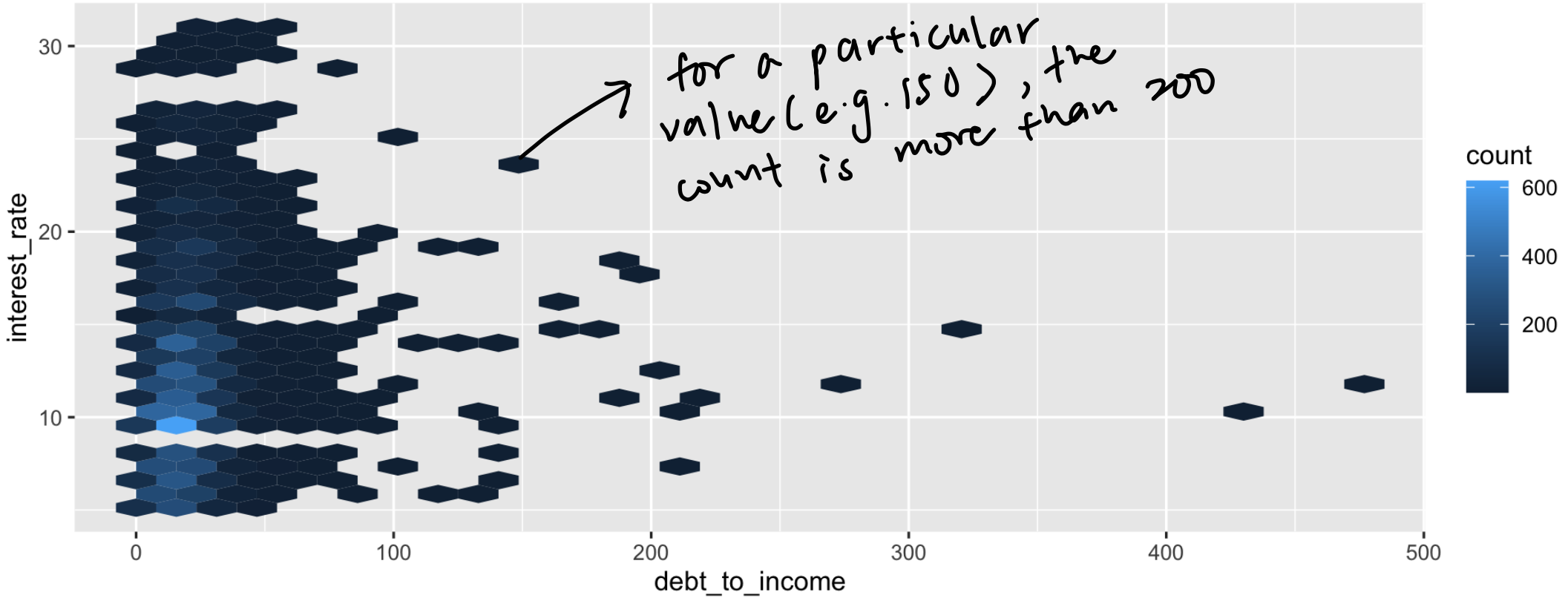
```
ggplot(loans, aes(x = debt_to_income, y = interest_rate)) +
  geom_point()
```

scatter plot.



Hex plot

```
ggplot(loans, aes(x = debt_to_income, y = interest_rate)) +
  geom_hex()
```



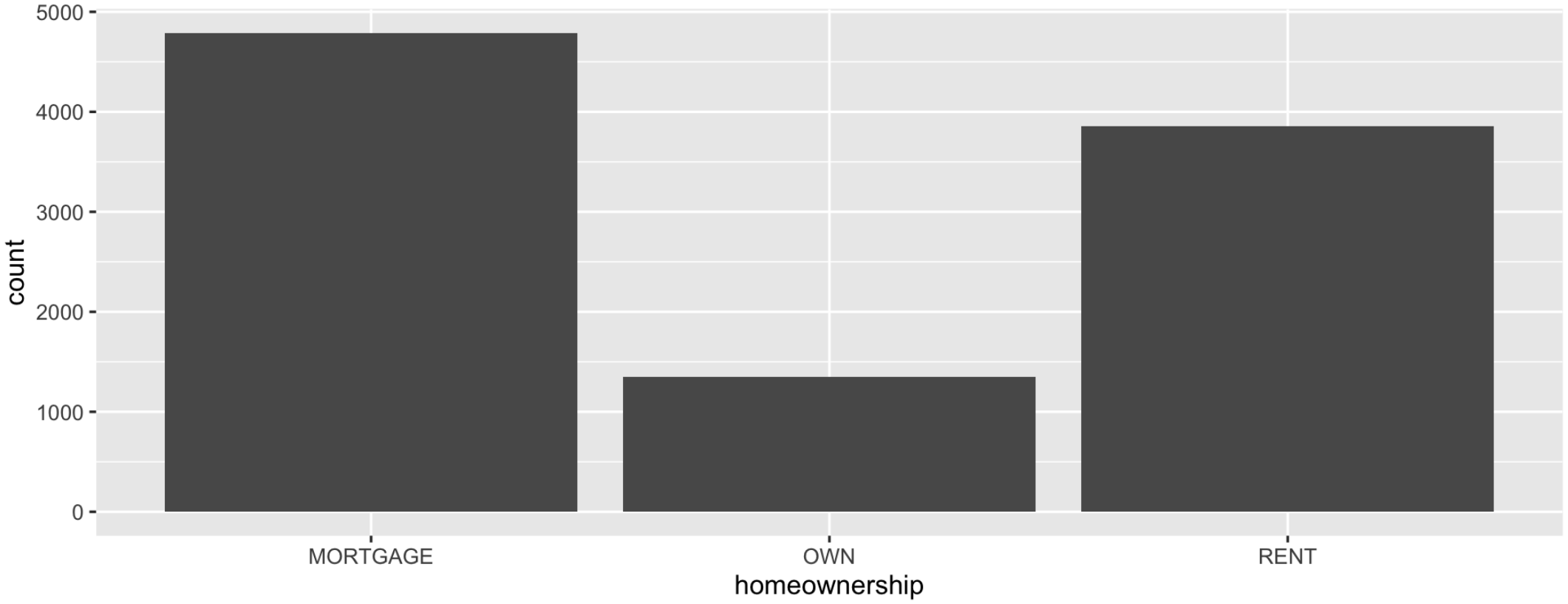
Hex plot

```
ggplot(loans %>% filter(debt_to_income < 100),
       aes(x = debt_to_income, y = interest_rate)) +
  geom_hex()
```

III. Visualizing categoric variables

Bar plot

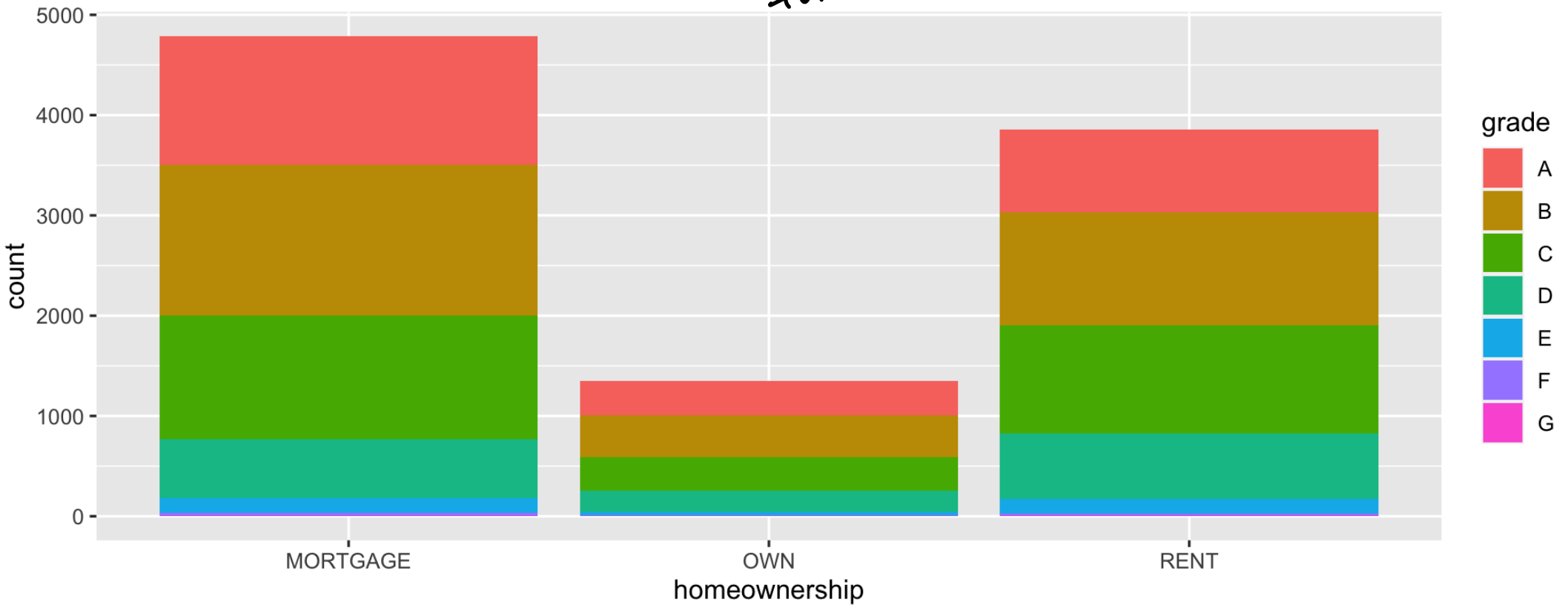
```
ggplot(loans, aes(x = homeownership)) +
  geom_bar()
```



Segmented bar plot

```
ggplot(loans, aes(x = homeownership,
                  fill = grade)) +
  geom_bar()
```

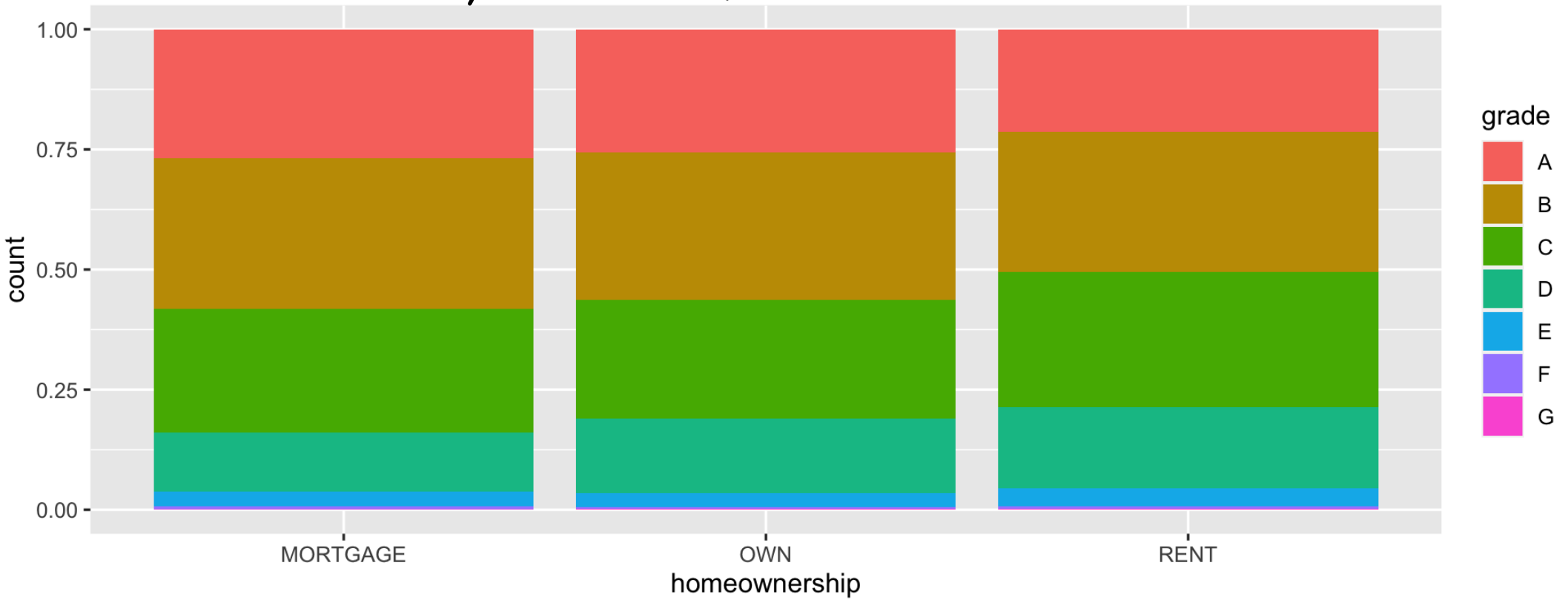
fill is used to distinguish datapoints in a plot by a categorical variable (grade) in this case.



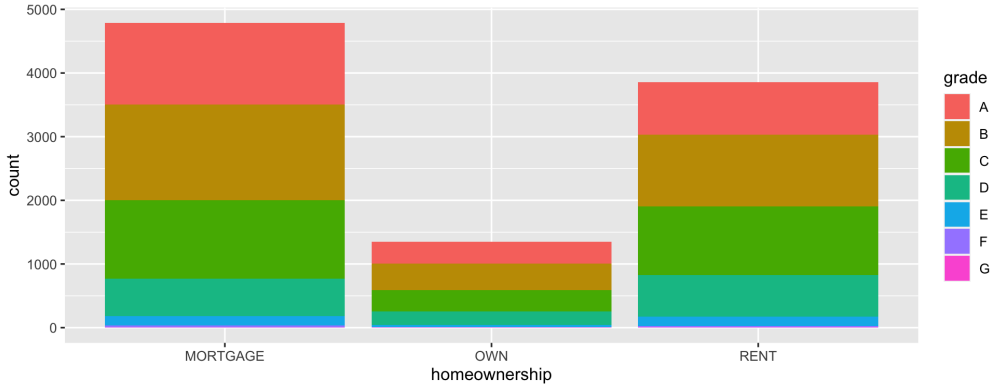
Segmented bar plot

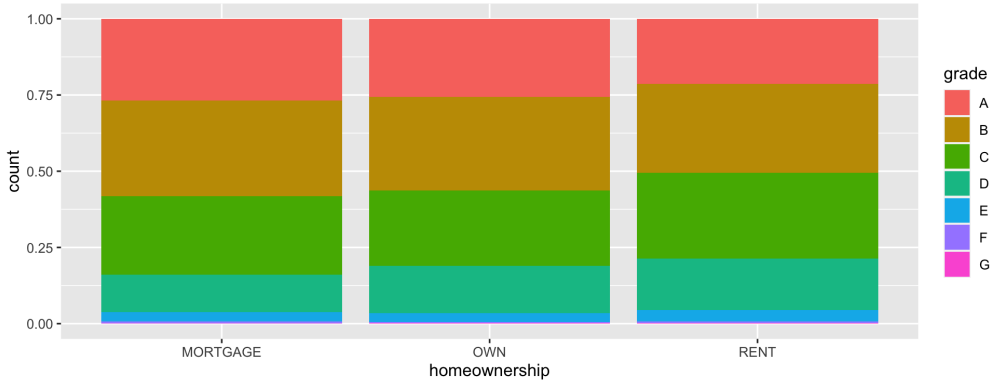
```
ggplot(loans, aes(x = homeownership, fill = grade)) +
  geom_bar(position = "fill")
```

↳ creates a stacked bar



Comparison


5000
4000
3000
2000
1000
0
count
MORTGAGE
OWN
RENT
homeownership
grade
A
B
C
D
E
F
G


1.00
0.75
0.50
0.25
0.00
count
MORTGAGE
OWN
RENT
homeownership
grade
A
B
C
D
E
F
G

Customizing bar plots

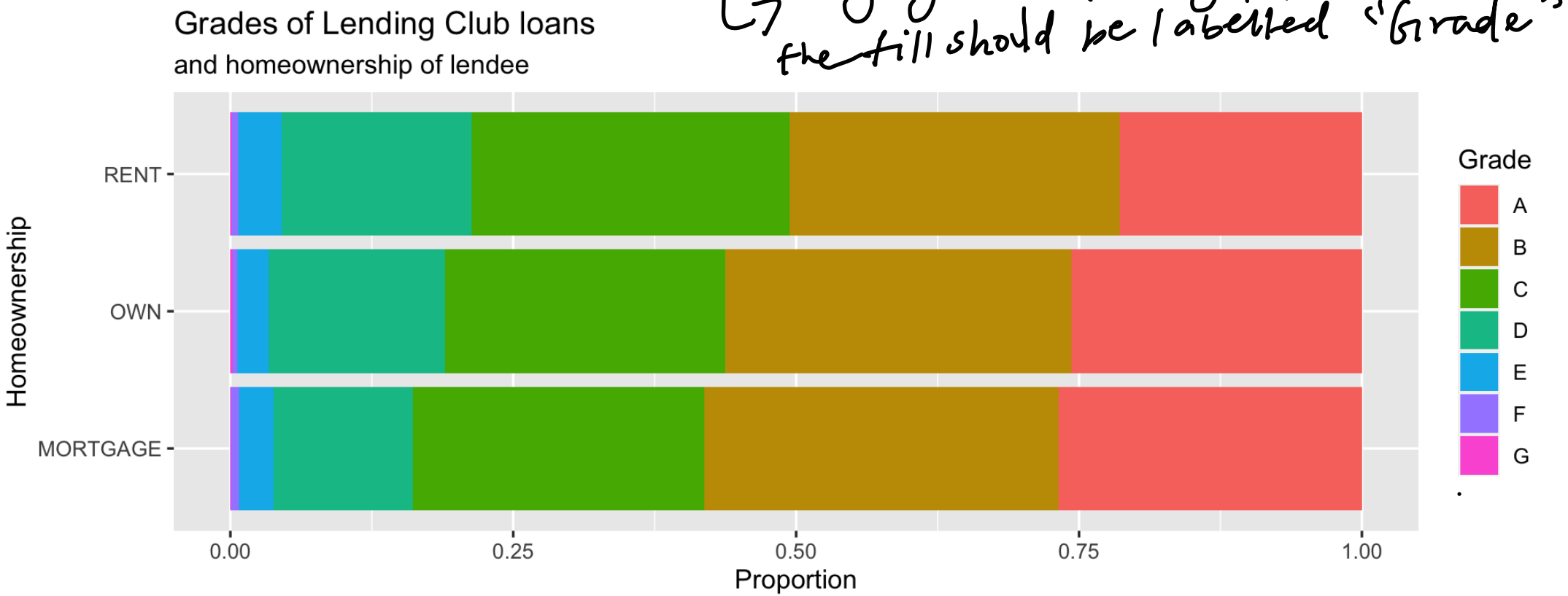
not defining x-axis; to see proportion

this is saying you want the categorical variable to be 'grade'

but this is asking to fill in the position of the bars

```
ggplot(loans, aes(y = homeownership, fill = grade)) + geom_bar(position = "fill") +
labs( x = "Proportion", y = "Homeownership", fill = "Grade", title = "Grades of Lending Club loans
```

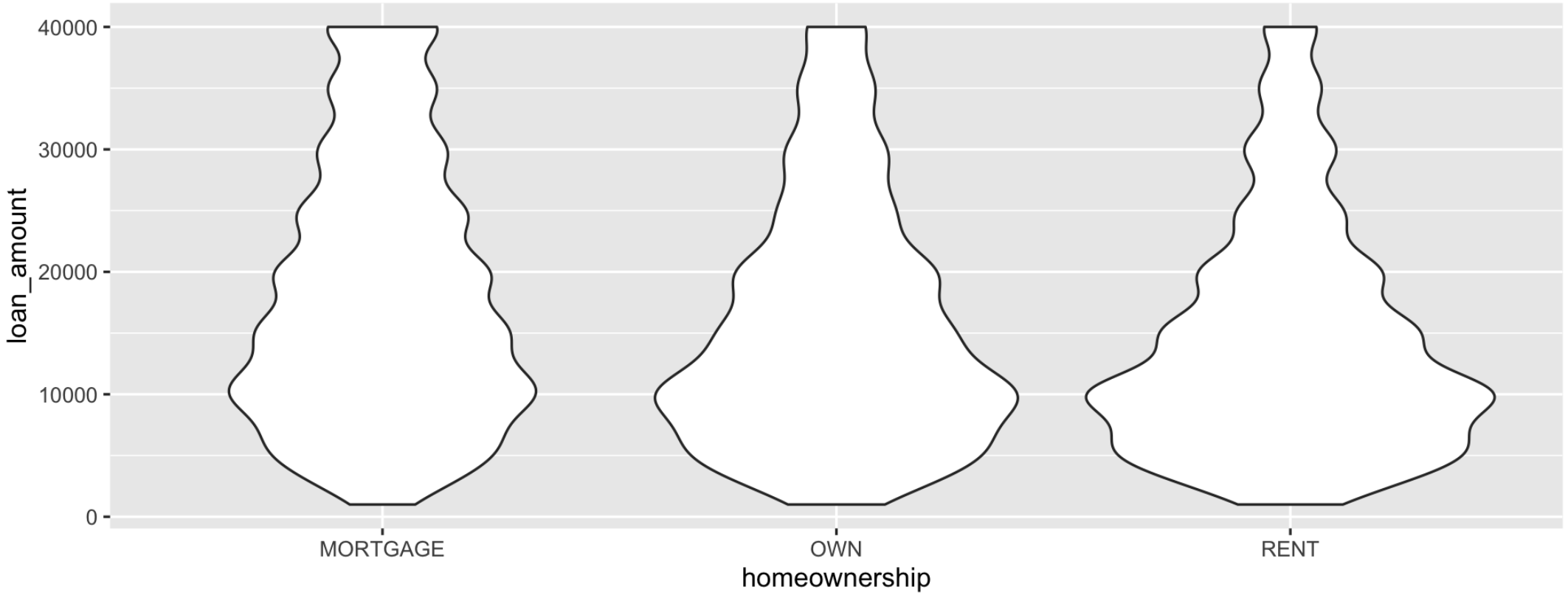
↳ saying that the legend for the fill should be labelled "Grade"



IV. Visualizing variables of varied types

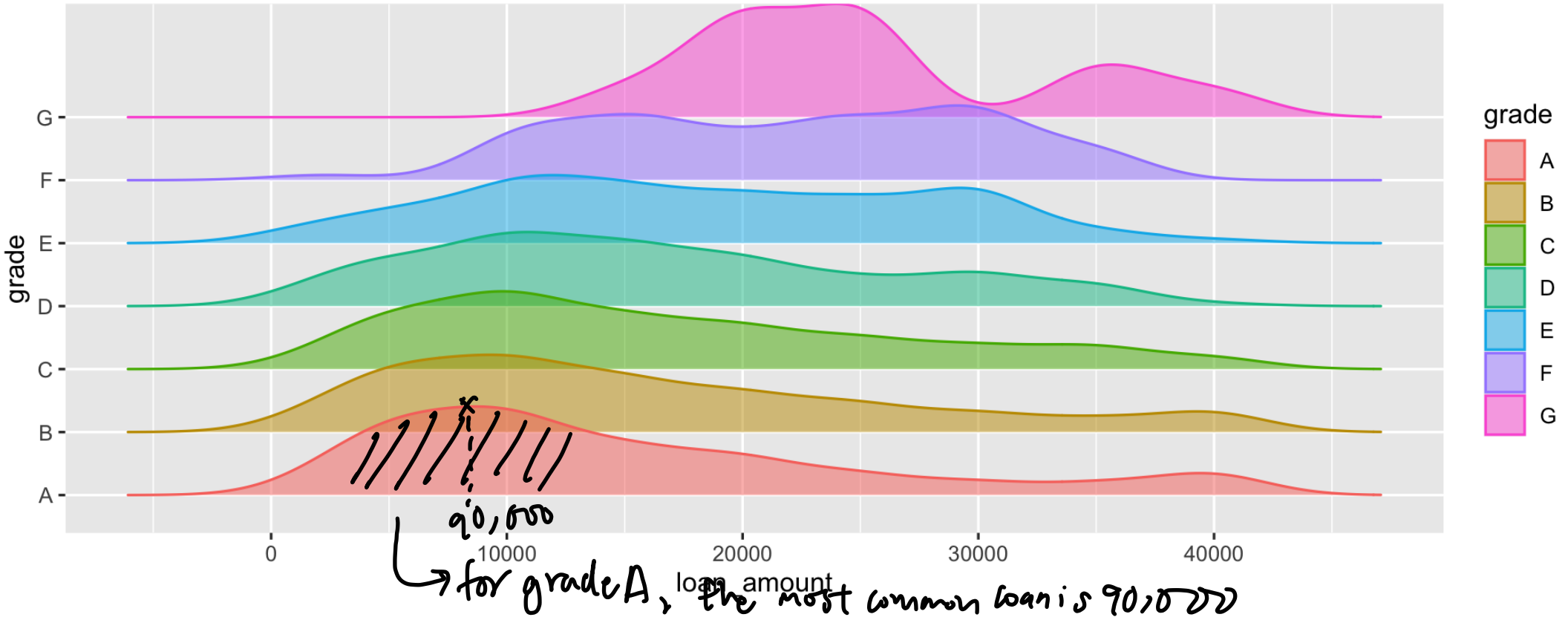
Violin plots

```
ggplot(loans, aes(x = homeownership, y = loan_amount)) +
  geom_violin()
```



Ridge plots

```
library(ggridges)
ggplot(loans, aes(x = loan_amount, y = grade, fill = grade, color = grade)) +
  geom_density_ridges(alpha = 0.5)
```



Thanks!

Slides created via the R packages:

xaringan
gadenbuie/xaringanthemer.