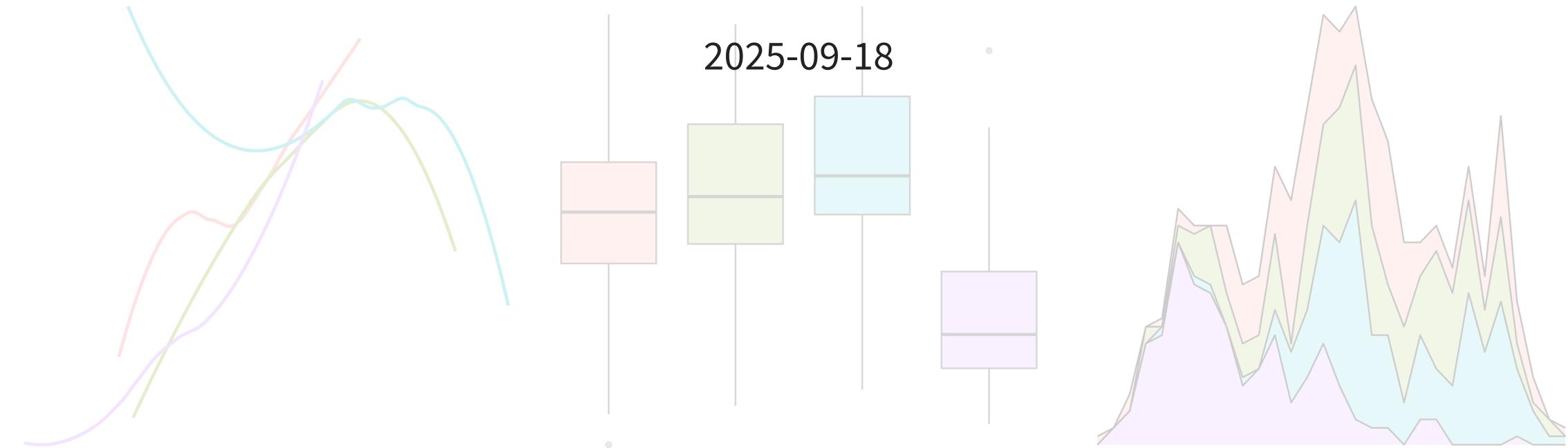


# Visualizations with ggplot



# Exploratory data analysis

- **Exploratory data analysis (EDA)** is an approach to analyzing data sets to summarize the main characteristics.
  - Often visual through plots
- Because of its name “exploratory”, we typically perform EDA at the beginning of a project
- Can also calculate summary statistics and perform data wrangling/manipulation/transformation at (or before) this stage of the analysis

# Tidy data

- When working with data in `R`, always look at the data to ensure it is in **tidy** format:
  - Each row represents an observation, each column represents a variable describing the observations
- `anscombe` data frame: four datasets each with 11 observations each and the same two variables x and y

# Data visualization

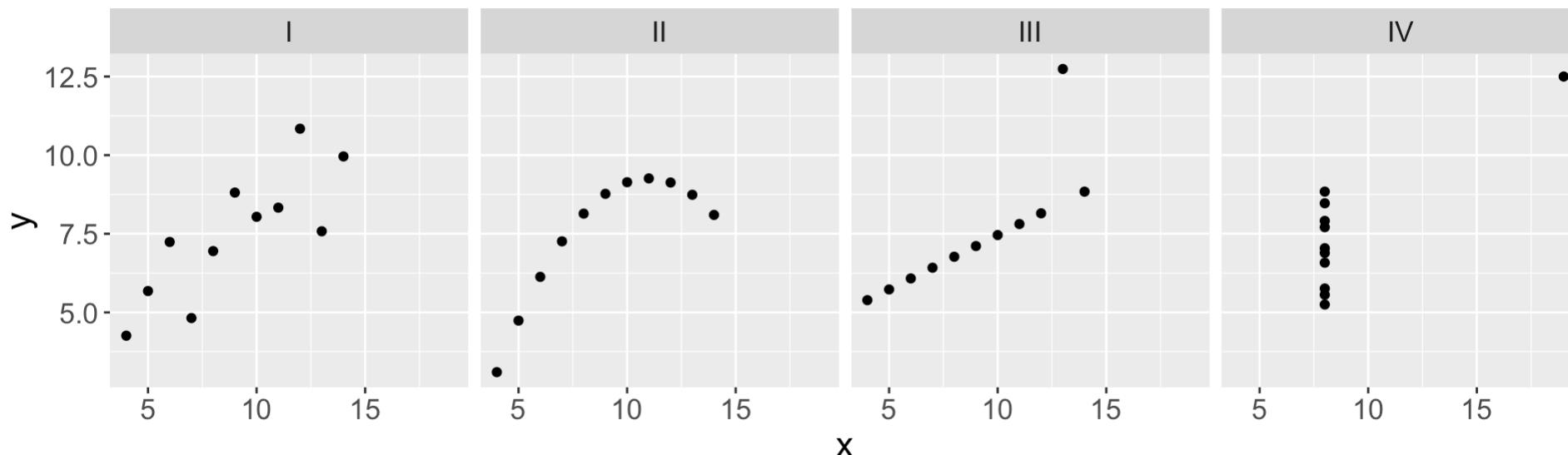
- **Data visualization** is the creation and study of the visual representation of data
- We have learned how to use base `R` code to make simple histograms. We will now use `ggplot2` which makes customization easier

# Why do we visualize?

- Summary statistics from each of the four datasets in `anscombe`:

```
# A tibble: 4 × 5
  set    mean_x  mean_y  sd_x  sd_y
  <fct>  <dbl>   <dbl>  <dbl> <dbl>
1 I      9.00    7.50   3.32  2.03
2 II     9.00    7.50   3.32  2.03
3 III    9.00    7.50   3.32  2.03
4 IV     9.00    7.50   3.32  2.03
```

- Let's visualize the four data sets. What would be an appropriate type of plot to examine the relationship between the two quantitative variables `x` and `y`?



# Introduction to ggplot

- We will learn how to create histograms, box plots, and scatterplots using the `ggplot()` function from the `ggplot2` library (included in `tidyverse`)
  - Plots are constructed in layers
- At a *minimum*, we need to specify 1) the dataset, 2) variable(s) from the dataset we'd like to plot, and 3) the type of plot
  - How does this differ from the previous coding practice?
- This is what the code will generally look like. Values in `<>` and `xxx` denote what you as the coder need to specify.

```
1 ggplot(data = <dataset>, # specify data frame
2         mapping = aes(x = <x-var>)) + # specify variables to be used in plot
3         geom_xxx() + # specify plot type
4         <other options>
```

# Inheriting arguments

- Many functions related to plotting in ggplot take the form `geom_xxx()`
- The Help file for these functions show that the first two arguments are `mapping` and `data`. These are *automatically inherited* from the `mapping` and `data` arguments in the first layer `ggplot()` function
  - i.e. you don't need to re-specify them, unless you are trying to add a new data frame's data to your visualization

## Description

These geoms add reference lines (sometimes called rules) to a plot, either horizontal, vertical, or diagonal (specified by slope and intercept). These are useful for annotating plots.

## Usage

```
geom_abline(  
  mapping = NULL,  
  data = NULL,  
  ...,  
  slope,  
  intercept,  
  na.rm = FALSE,  
  show.legend = NA  
)
```

Inherited (the = \_\_\_\_\_ means that the function has some pre-defined, default arguments that allow function to run without further specification)

Since slope and intercept are not pre-specified, you need to provide values for these arguments

# Additional variables + modifications

- We emphasize making informative and useful visualizations.
  - Informative titles and labels
  - Plot should tell a meaningful story
- Depending on the plot and data, we can map additional variables by:
  1. Specifying visual cues via **aesthetics**: color, size, shape, alpha (transparency)
  2. *Faceting* (will see this next week)

# Live code

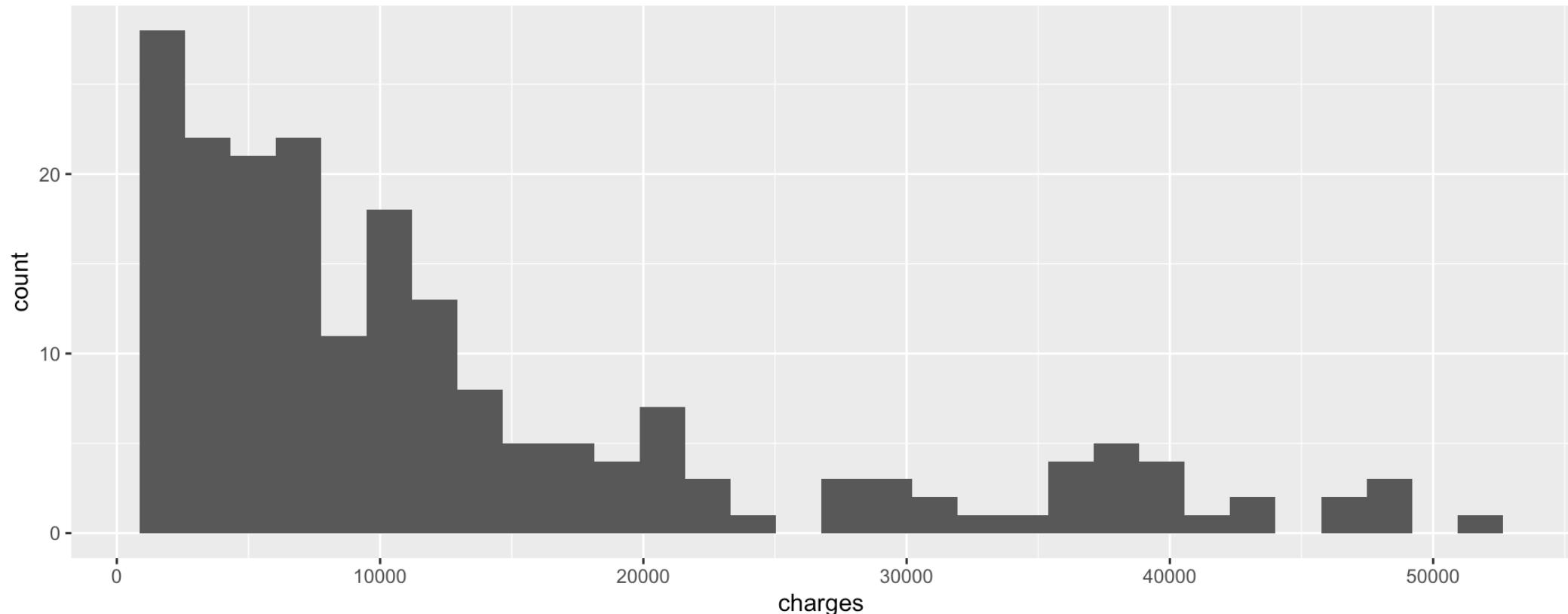
Note: most of the code I will show is included in the remaining slides. However, we will most likely go off-script based on questions from the class!

# Slides on ggplot()

# geom\_histogram()

```
1 ggplot(data = insurance, mapping = aes(x = charges)) +  
2   geom_histogram()
```

`stat\_bin()` using `bins = 30` . Pick better value with `binwidth` .

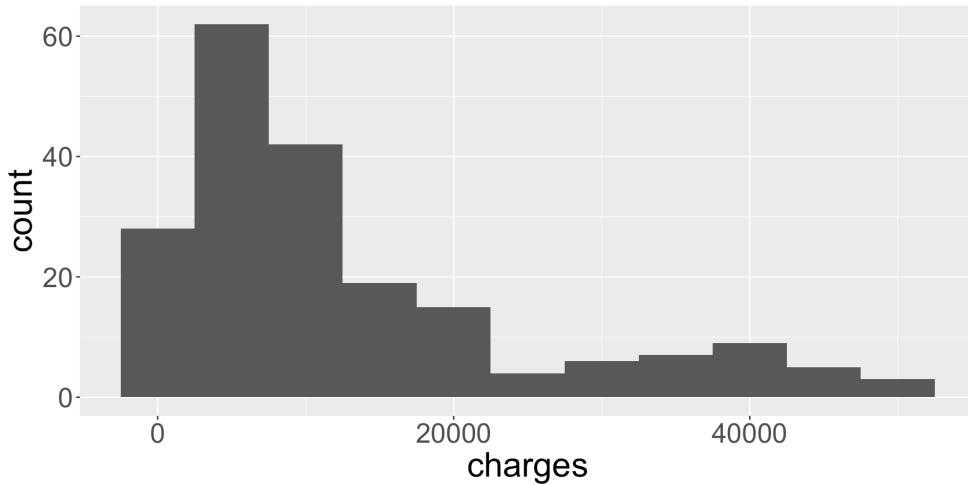


Note the message provided when you execute this code!

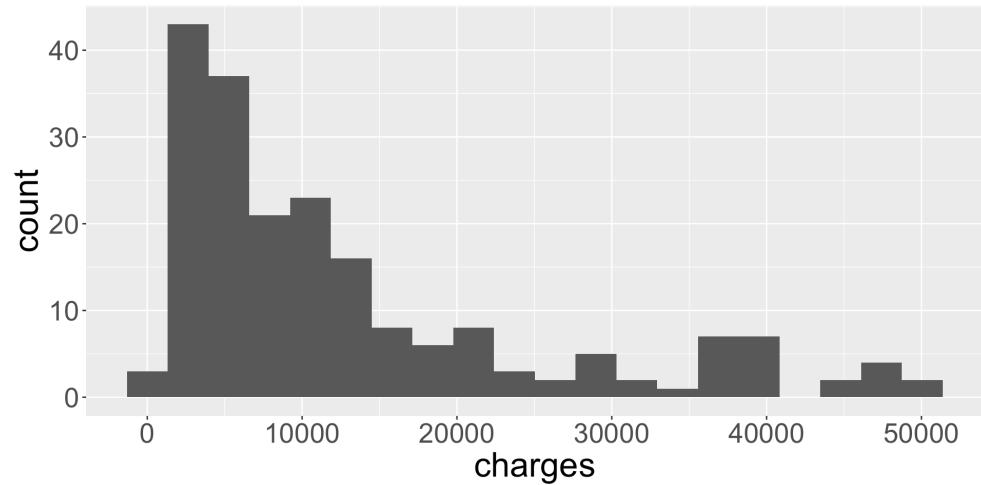
# geom\_histogram() cont.

To improve on histogram we change the bin width.

```
1 ggplot(data = insurance,  
2         mapping = aes(x = charges)) +  
3         geom_histogram(binwidth = 5000)
```

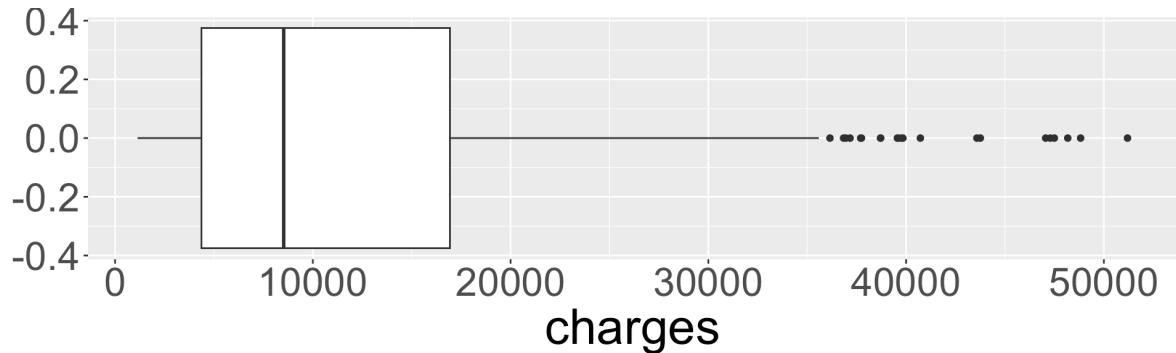


```
1 ggplot(data = insurance,  
2         mapping = aes(x = charges)) +  
3         geom_histogram(bins = 20)
```

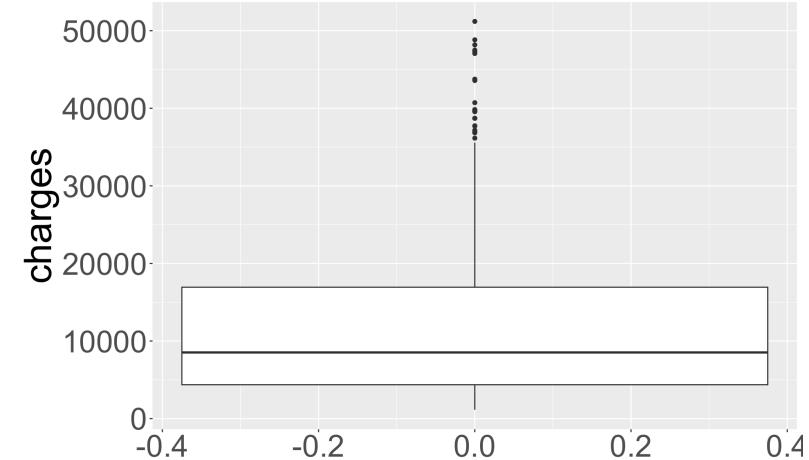


# geom\_boxplot()

```
1 ggplot(data = insurance, mapping = aes(x = charges)) +  
2   geom_boxplot()
```

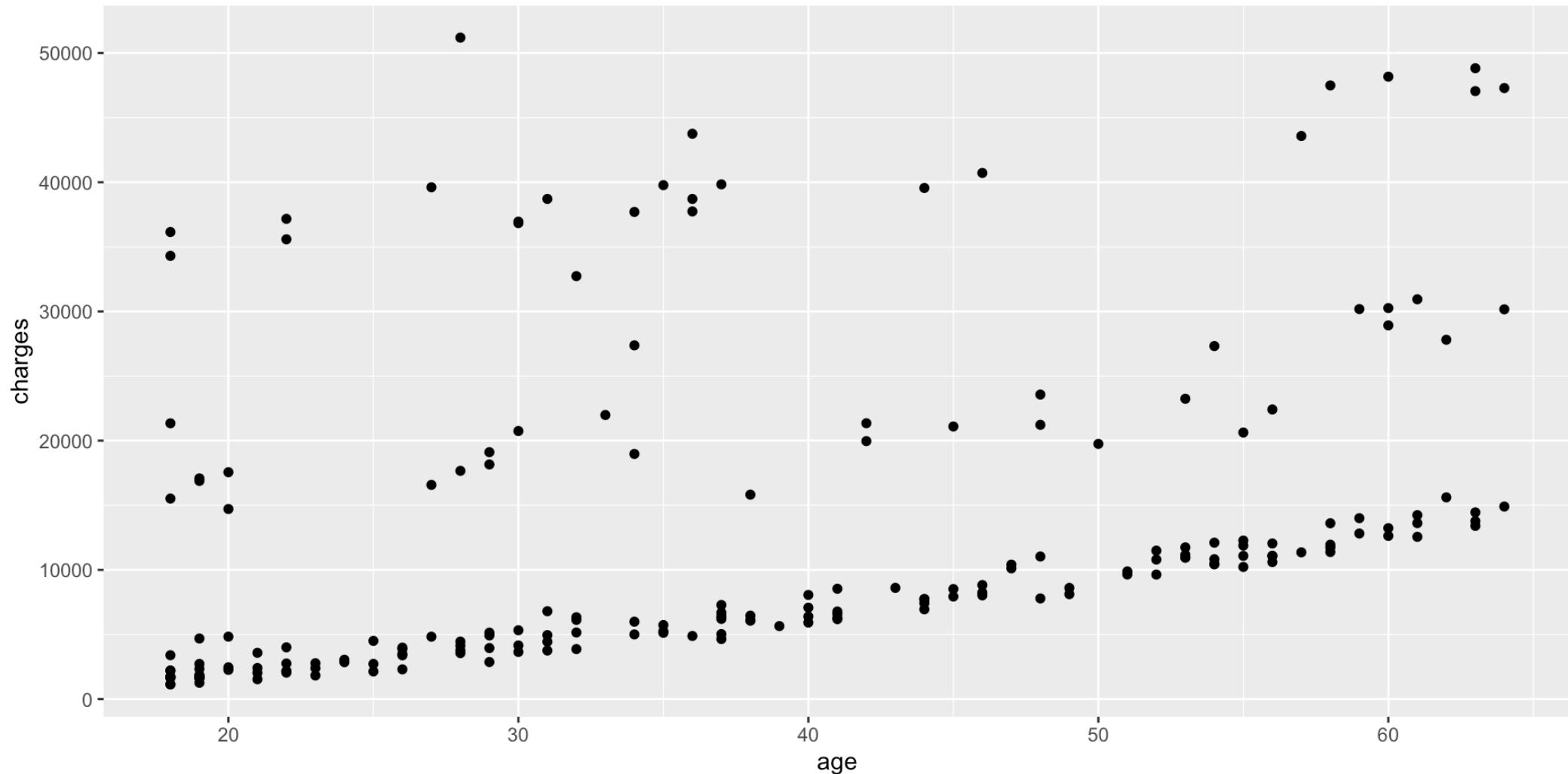


```
1 ggplot(data = insurance,  
2         mapping = aes(y = charges)) +  
3   geom_boxplot()
```



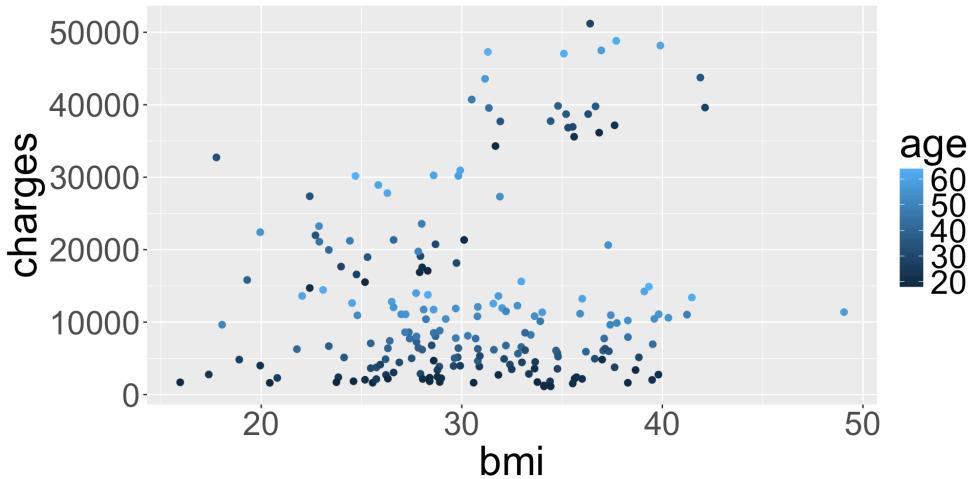
# geom\_point()

```
1 ggplot(data = insurance, mapping = aes(x = age, y = charges)) +  
2   geom_point()
```

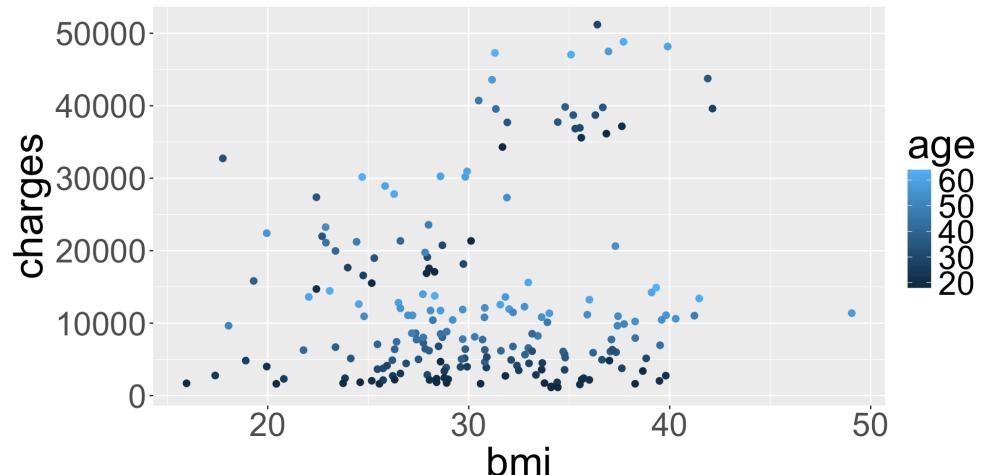


# Aesthetics: color

```
1 ggplot(data = insurance,  
2         mapping = aes(x = bmi, y = charges,  
3                             col = age)) +  
4     geom_point()
```

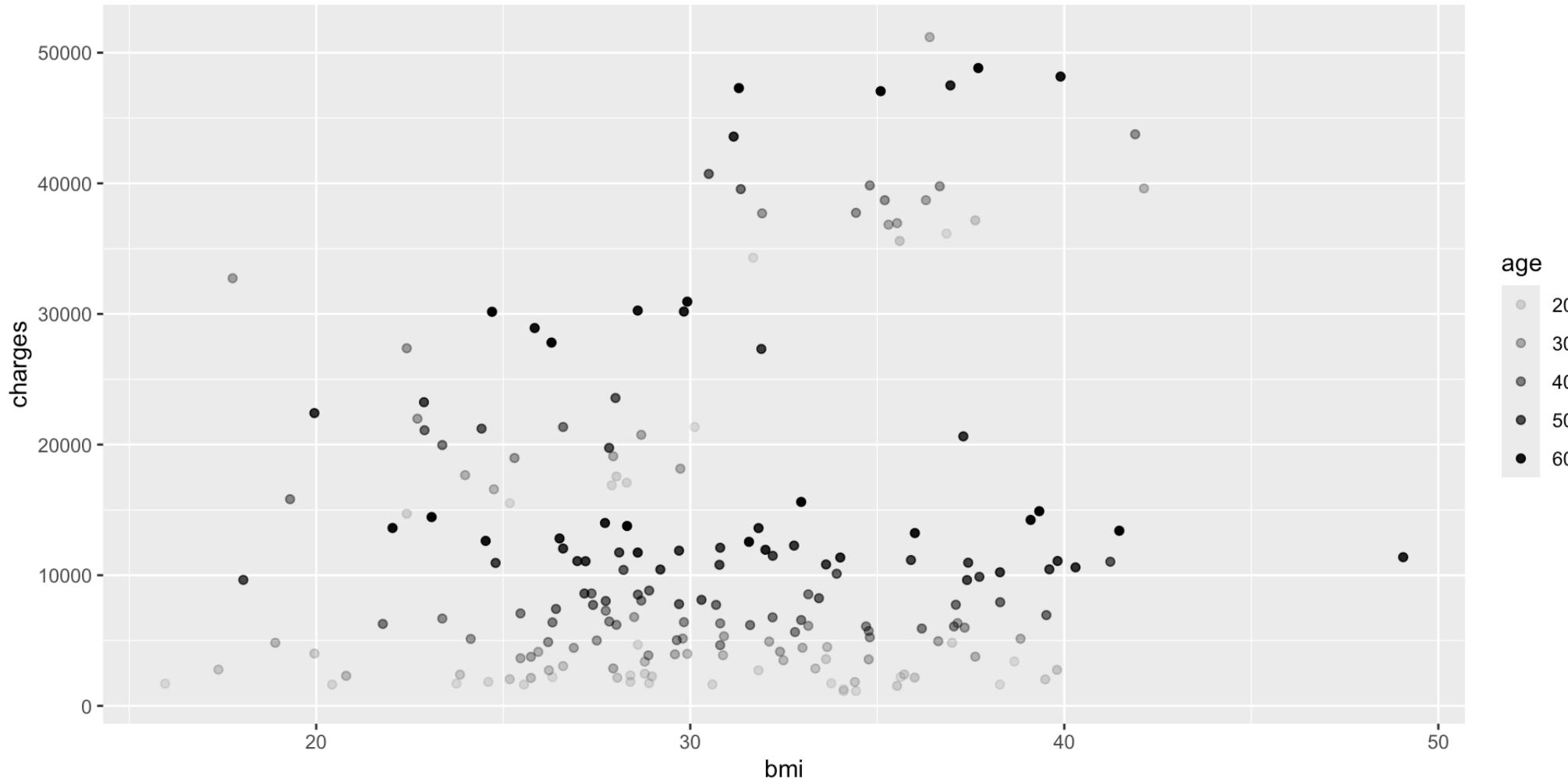


```
1 ggplot(data = insurance,  
2         mapping = aes(x = bmi, y = charges)) +  
3     geom_point(aes(col = age))
```



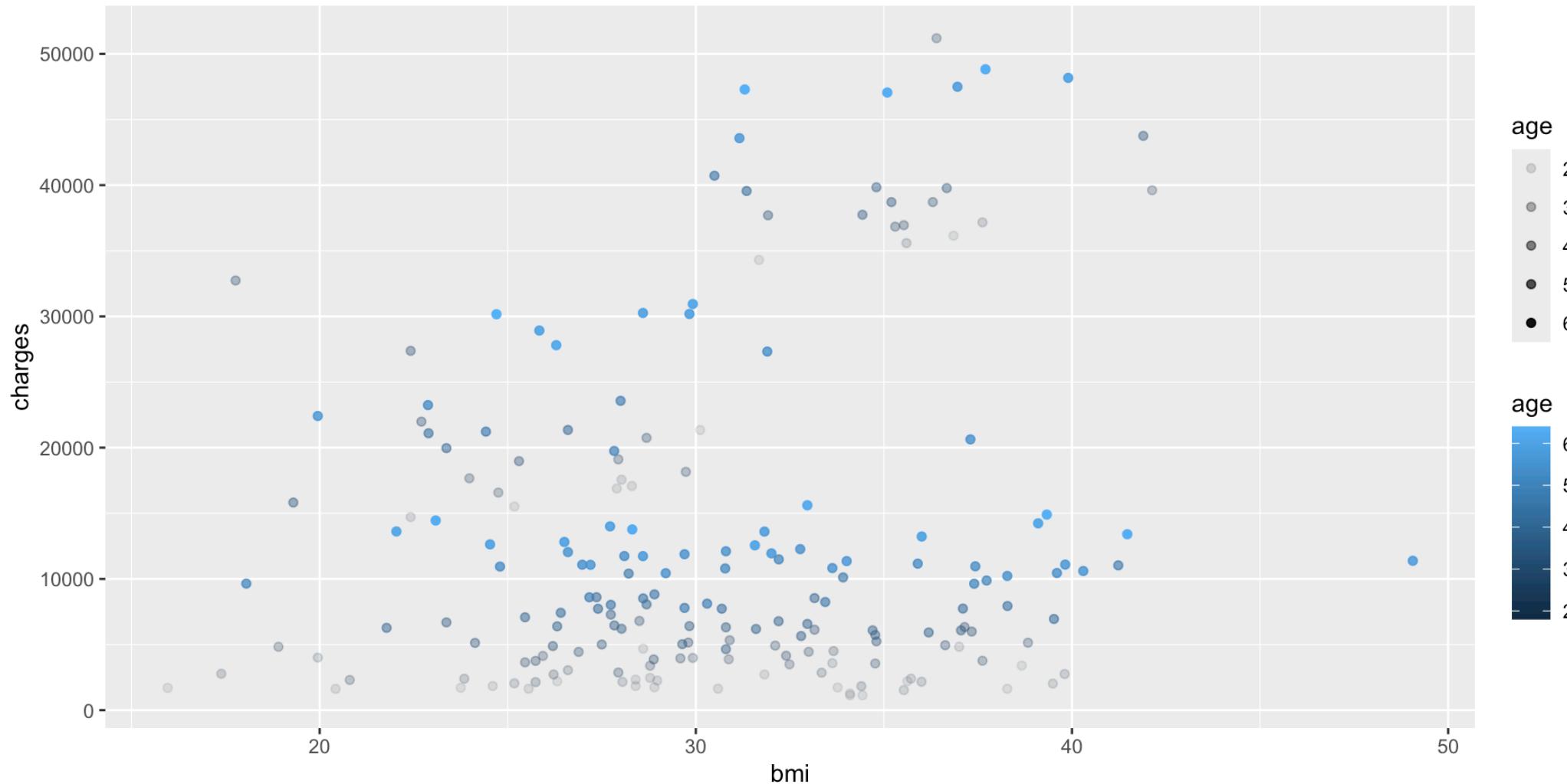
# Aesthetics: transparency

```
1 ggplot(data = insurance, mapping = aes(x = bmi, y = charges,
2                                         alpha = age)) +
3   geom_point()
```



# Specifying multiple aesthetics

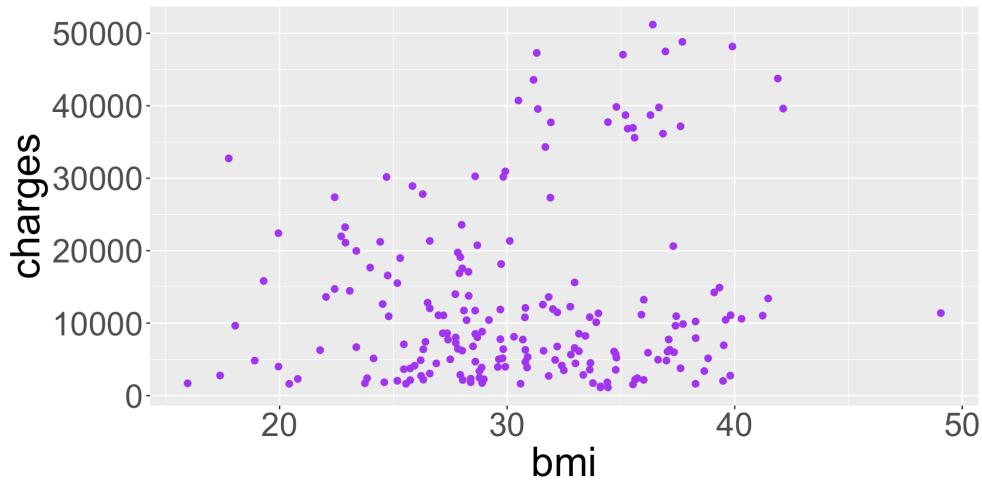
```
1 ggplot(data = insurance, mapping = aes(x = bmi, y = charges, col = age, alpha = age)) +  
2 geom_point()
```



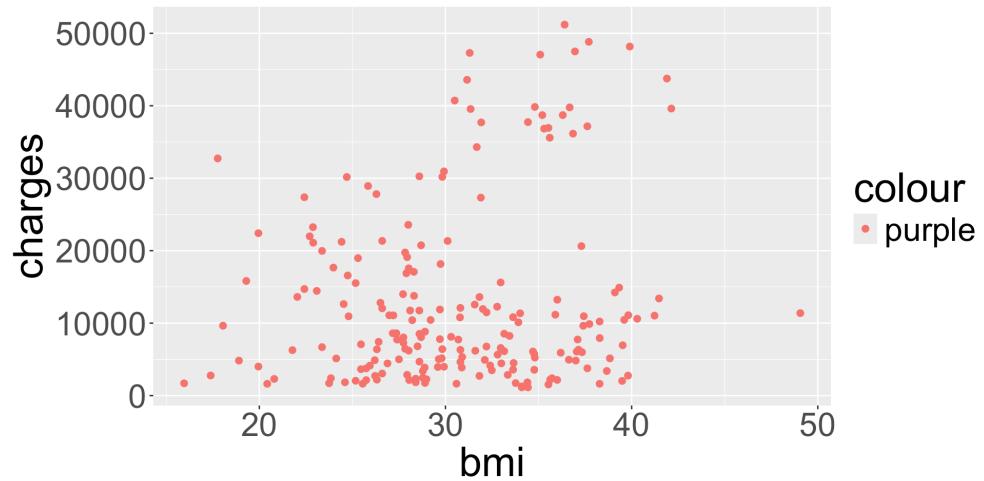
# When to map to variable

What's going on here?

```
1 ggplot(data = insurance,  
2         mapping = aes(x = bmi, y = charges)) +  
3         geom_point(col = "purple")
```



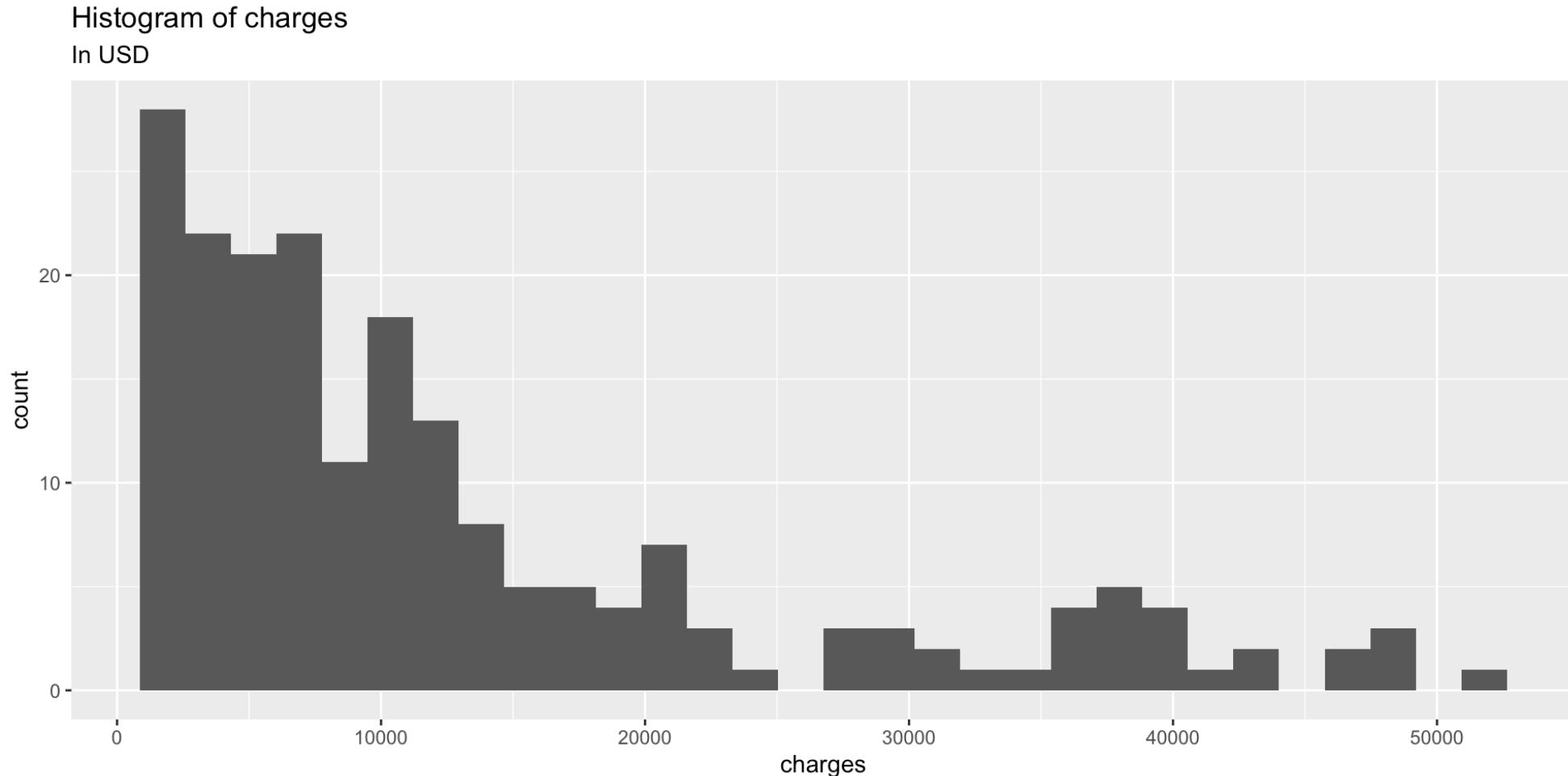
```
1 ggplot(data = insurance,  
2         mapping = aes(x = bmi, y = charges)) +  
3         geom_point(aes(col = "purple"))
```



- Key takeaway: aesthetics should correspond/map to a variable in the data frame
  - “Fixed” visual cues are set outside of `aes()`

# Adding a title

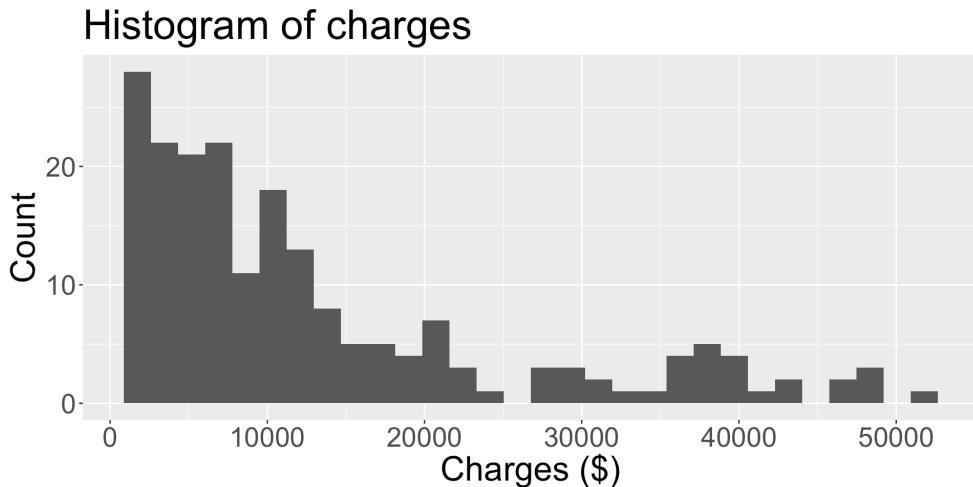
```
1 ggplot(data = insurance, mapping = aes(x = charges)) +  
2   geom_histogram() +  
3   ggtitle("Histogram of charges",  
4           subtitle = "In USD")
```



# Changing axis labels

By default, axis titles are taken from variable name specified in `aes()`. To change:

```
1 ggplot(data = insurance,  
2         mapping = aes(x = charges)) +  
3         geom_histogram() +  
4         ggtitle("Histogram of charges") +  
5         xlab("Charges ($)")
```



```
1 ggplot(data = insurance,  
2         mapping = aes(x = charges)) +  
3         geom_histogram() +  
4         labs(title = "Histogram of charges",  
5               x = "Charges ($)", y = "Count")
```

