

# **Introduction to R Workshop**

**Session 4**  
**Visualize Data**  
May 6, 2019

## Goals

1. Appreciate the importance of visualization as device to generate knowledge about data
2. Learn how to use ggplot2 to create visualizations from data frames

## Objectives

1. Explain the template to create any graph with ggplot2
2. Explain how aesthetic mappings relate variables to graphic markings on a plot
3. Distinguish between setting and mapping aesthetics
4. Explain how to use a geom function to add a layer to a graph
5. Distinguish between global and local ggplot2 settings

# Your Turn 1

Consider the `orders` data frame. How do you think `order_time` and `result_time` are related to each other?

Pair up and discuss.



# Your Turn 2

Type the following code in the console to make a graph.

Pay attention to the spelling, capitalization, and parentheses!

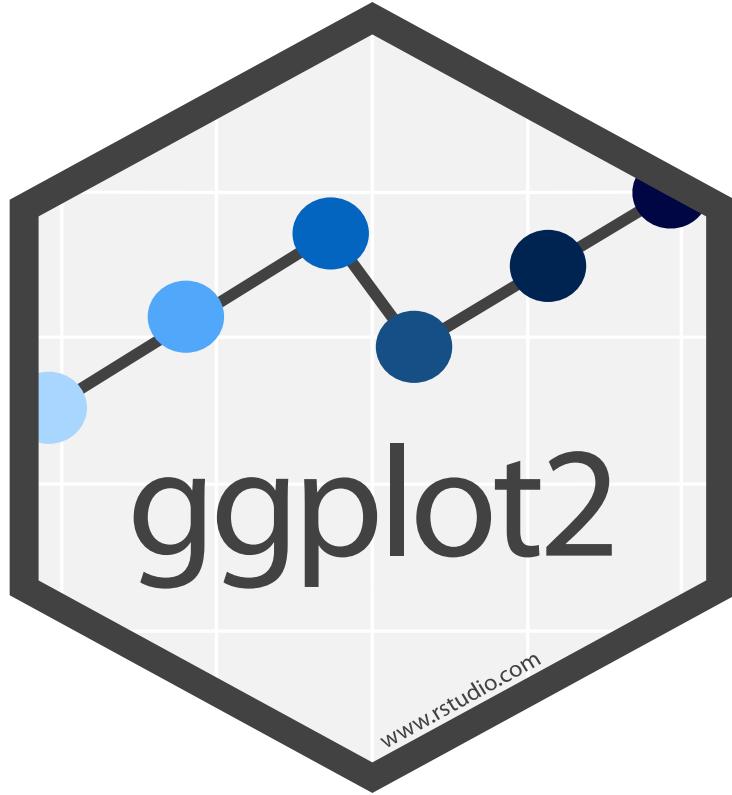
```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time))
```



When you run this code, you will get what looks like an error but is actually just a message. R lets you know that some of the rows did not have data for `order_time` and/or `result_time` so those points were not plotted.

Warning message:  
Removed 7152 rows containing missing values (geom\_point).

```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time))
```



# ggplot()

initialize a plot  
with ggplot()

data frame

+ sign  
before new line

```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time))
```

type of layer

mappings inside  
aes() function

x variable

y variable

# To make **any** kind of graph:

1. Choose a “tidy”  
**data frame**

```
ggplot(data = data_frame) +  
  geom_function(mapping = aes(mappings))
```

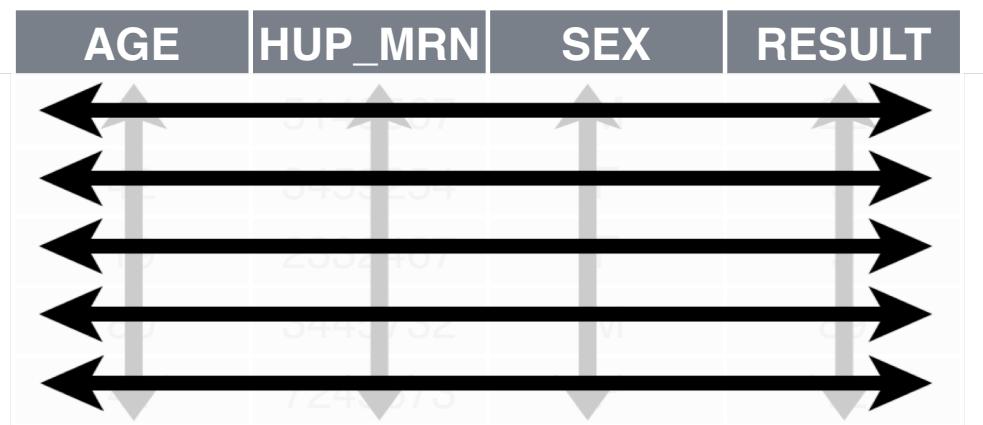
2. Pick a “**geom**”  
function

3. Write aesthetic  
**mappings**



# 1. Pick a “Tidy” Data Frame

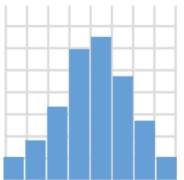
AGE	HUP_MRN	SEX	RESULT
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4



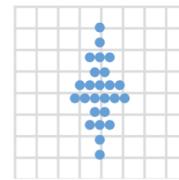
A data set is **tidy** if:

1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

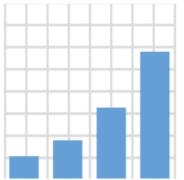
## 2. Choose a “Geom” Function



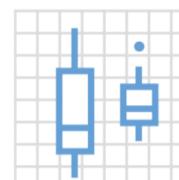
`geom_histogram()`



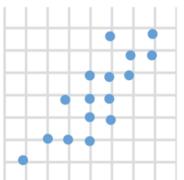
`geom_dotplot()`



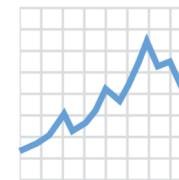
`geom_bar()`



`geom_boxplot()`



`geom_point()`



`geom_line()`



# Data Visualization with ggplot2 :: CHEAT SHEET

## Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

required  
Not required, sensible defaults supplied

`ggplot(data = mpg, aes(x = cty, y = hwy))` Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings    data    geom

`qplot(x = cty, y = hwy, data = mpg, geom = "point")` Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

## Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

### GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

- a + **geom\_blank()**  
(Useful for expanding limits)
- b + **geom\_curve**(aes(yend = lat + 1, xend = long + 1, curvature = z)) - x, yend, y, yend, alpha, angle, color, curvature, linetype, size
- a + **geom\_path**(lineend = "butt", linejoin = "round", linemiter = 1) - x, y, alpha, color, group, linetype, size
- a + **geom\_polygon**(aes(group = group)) - x, y, alpha, color, fill, group, linetype, size
- b + **geom\_rect**(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + **geom\_ribbon**(aes(ymax = unemploy - 900, ymin = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

### LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

- b + **geom\_abline**(aes(intercept = 0, slope = 1))
- b + **geom\_hline**(aes(intercept = lat))
- b + **geom\_vline**(aes(intercept = long))
- b + **geom\_segment**(aes(yend = lat + 1, xend = long + 1))
- b + **geom\_spoke**(aes(angle = 1:1155, radius = 1))

### ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

- c + **geom\_area**(stat = "bin") - x, y, alpha, color, fill, linetype, size
- c + **geom\_density**(kernel = "gaussian") - x, y, alpha, color, fill, group, linetype, size, weight
- c + **geom\_dotplot**() - x, y, alpha, color, fill
- c + **geom\_freqpoly**(x, y, alpha, color, group, map = map\_data("state"))

### TWO VARIABLES

#### continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

- e + **geom\_label**(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- e + **geom\_jitter**(height = 2, width = 2) - x, y, alpha, color, fill, shape, size
- e + **geom\_point**(x, y, alpha, color, fill, shape, size, stroke)
- e + **geom\_quantile**(x, y, alpha, color, group, linetype, size, weight)
- e + **geom\_rug**(sides = "bl") - x, y, alpha, color, linetype, size
- e + **geom\_smooth**(method = lm) - x, y, alpha, color, fill, group, linetype, size, weight
- e + **geom\_text**(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

#### discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

- f + **geom\_col**(x, y, alpha, color, fill, group, linetype, size)
- f + **geom\_boxplot**(x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight)
- f + **geom\_dotplot**(binaxis = "y", stackdir = "center") - x, y, alpha, color, fill, group
- f + **geom\_violin**(scale = "area") - x, y, alpha, color, fill, group, linetype, size, weight

#### discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```



### continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

- h + **geom\_bin2d**(binwidth = c(0.25, 500)) - x, y, alpha, color, fill, linetype, size, weight
- h + **geom\_density2d**() - x, y, alpha, colour, group, linetype, size
- h + **geom\_hex**() - x, y, alpha, colour, fill, size

### continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

- i + **geom\_area**() - x, y, alpha, color, fill, linetype, size
- i + **geom\_line**() - x, y, alpha, color, group, linetype, size
- i + **geom\_step**(direction = "hv") - x, y, alpha, color, group, linetype, size

### visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

- j + **geom\_crossbar**(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- j + **geom\_errorbar**(x, y, max, min, alpha, color, group, linetype, size, width (also **geom\_errorbarh**))
- j + **geom\_linerange**(x, ymin, ymax, alpha, color, group, linetype, size)
- j + **geom\_pointrange**(x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size)

### maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArests)))
map <- map_data("state")
```

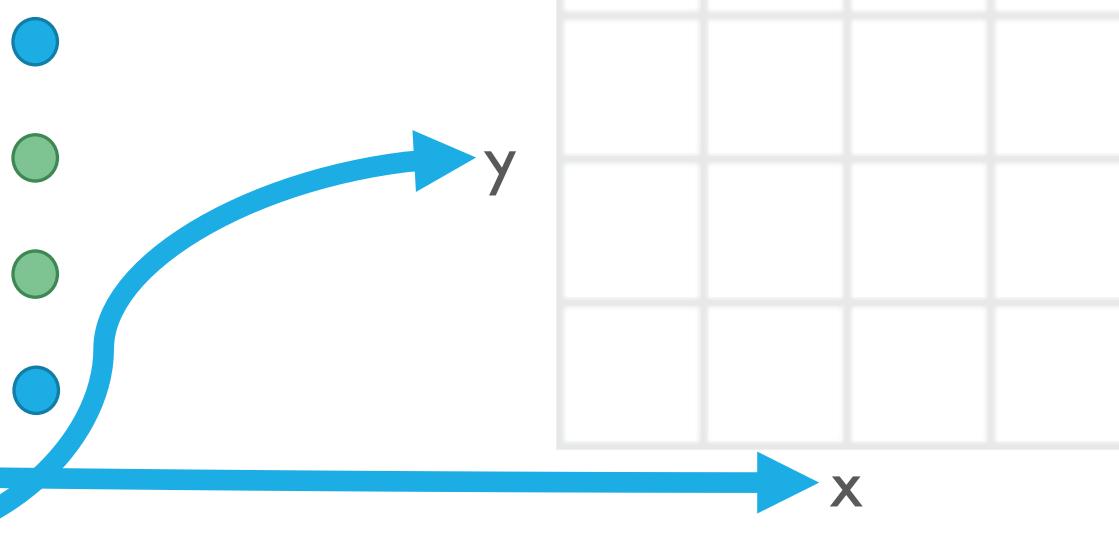
### 3. Write Aesthetic Mappings

```
aes(x = a, y = b, color = c)
```

Data frame

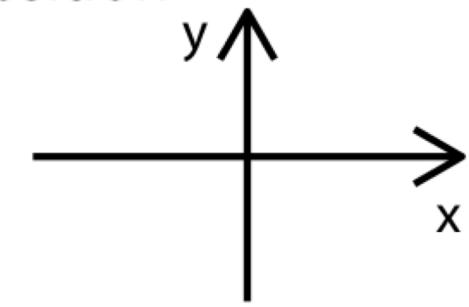
a	b	c
1	3	M
2	1	F
3	3	F
2	2	M

Graph

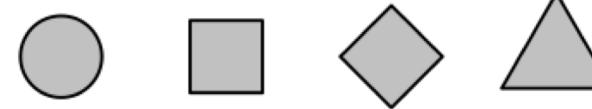


# Aesthetics

position



shape



size



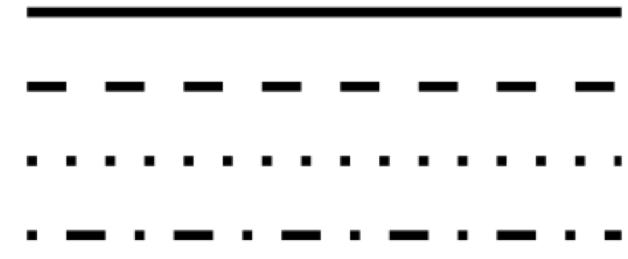
color



line width



line type



# Your Turn 3

Consider the 6 aesthetics shown just previously. All aesthetics can be used for discrete variables, but only some can be used for continuous variables.

Which aesthetics can be used only for discrete variables?

- A. Color
- B. Line type
- C. Shape
- D. B + C
- E. A + B + C



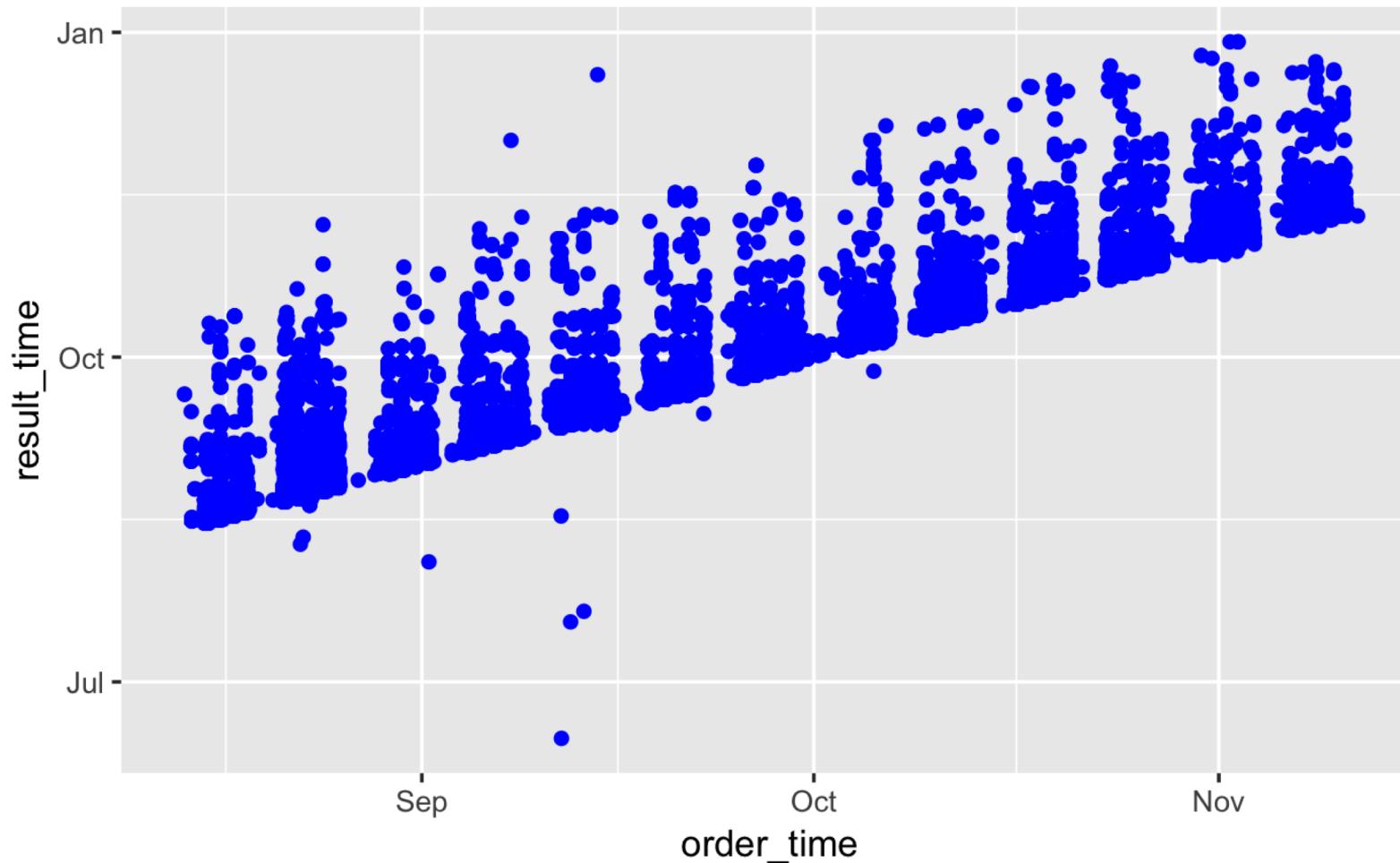
## Your Turn 4

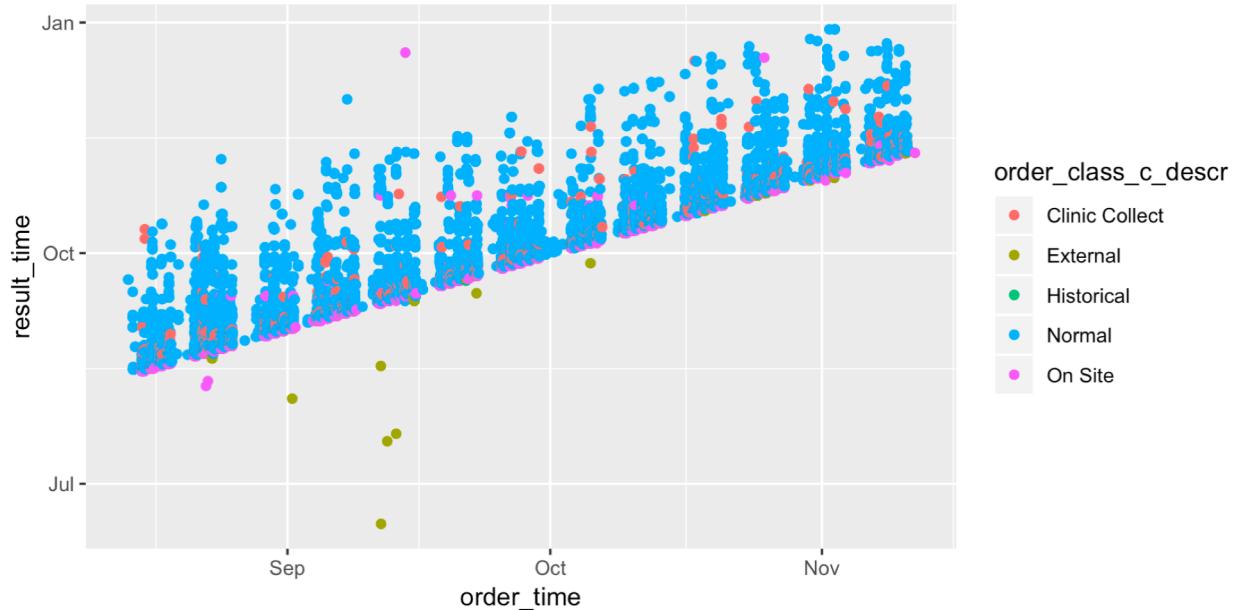
Open 04-visualize.Rmd. Work through the exercises of the section titled “Your Turn 4.”

05 : 00

# **Setting** vs **Mapping** Aesthetics

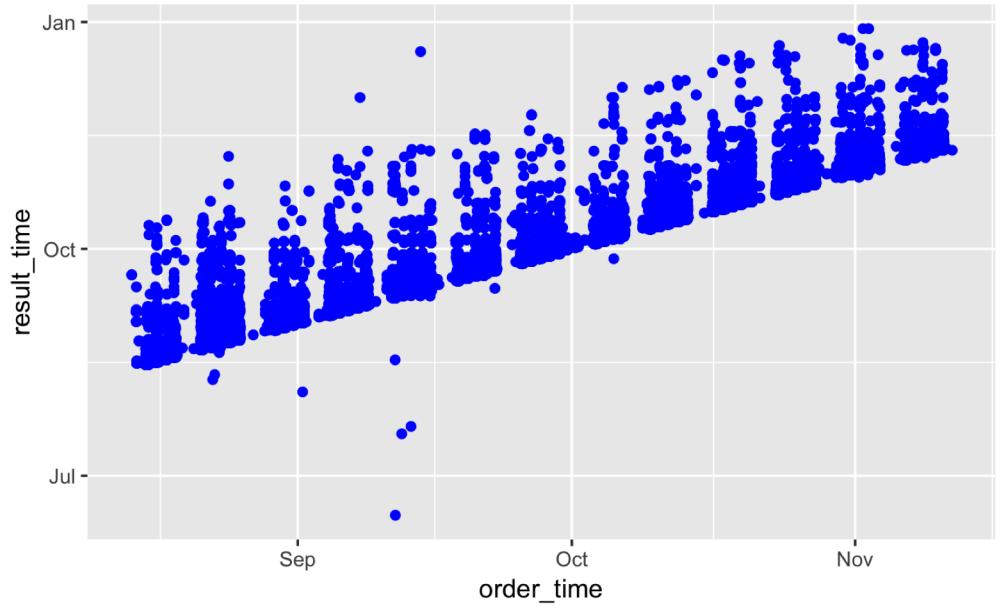
# How would you make this plot?





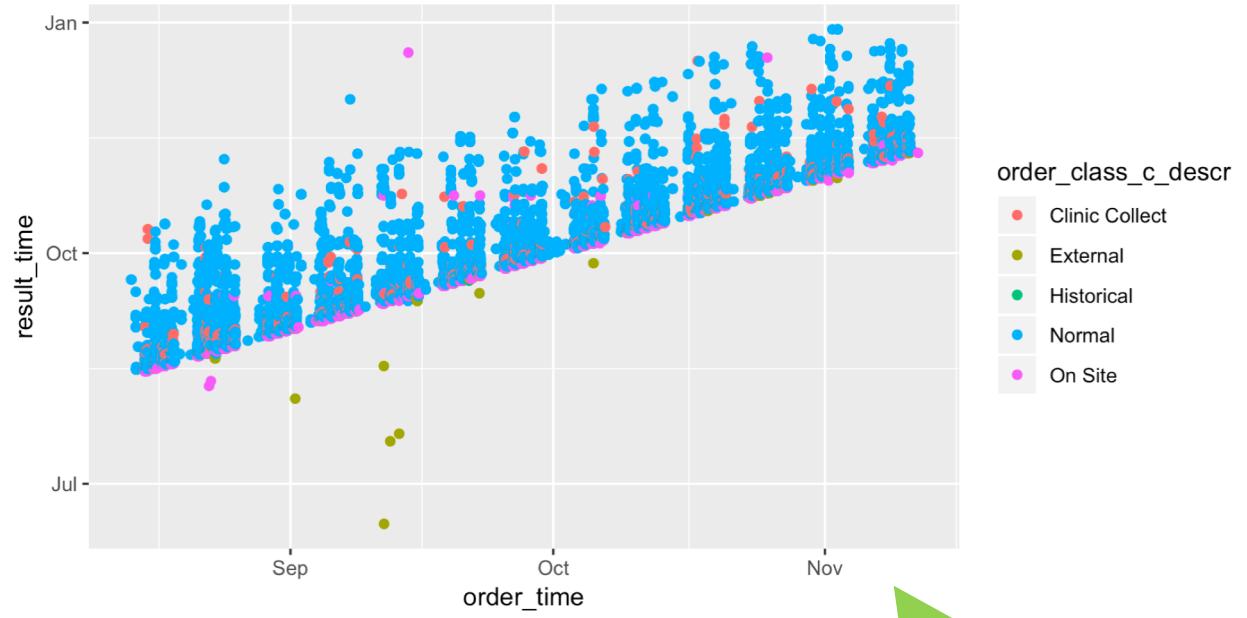
Inside of `aes()`:  
map an aesthetic  
to a variable

```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time, color = order_class_c_descr))
```

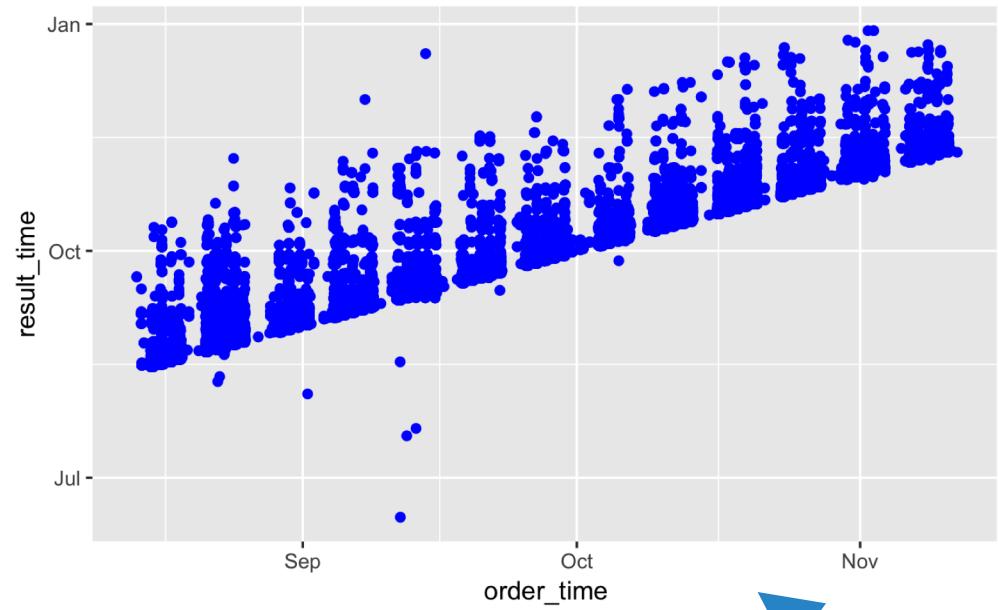


Outside of `aes()`:  
set an aesthetic to  
a **value**

```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time), color = "blue")
```



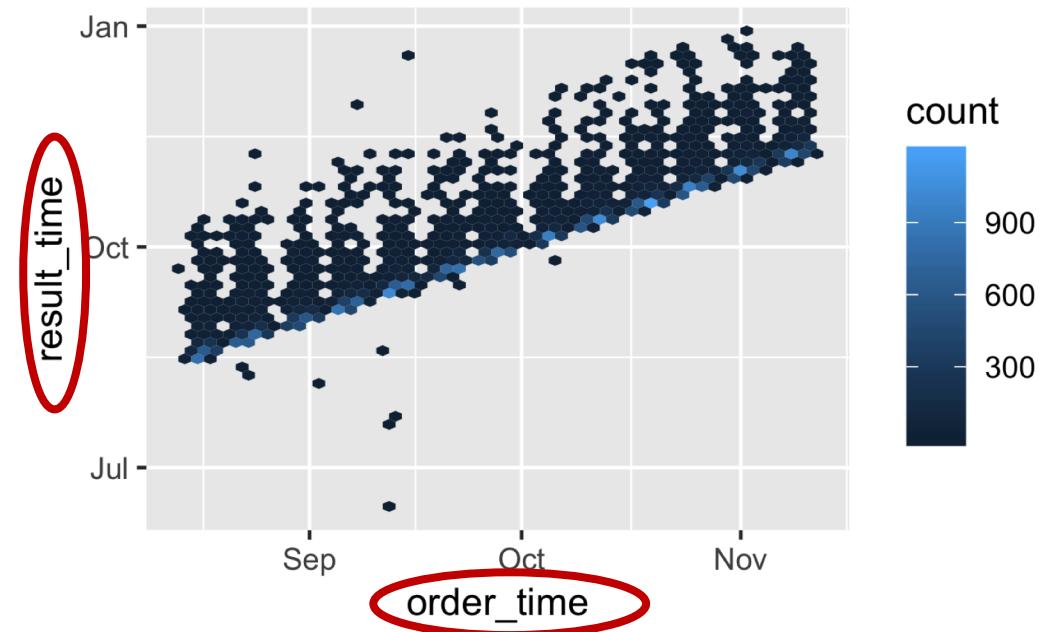
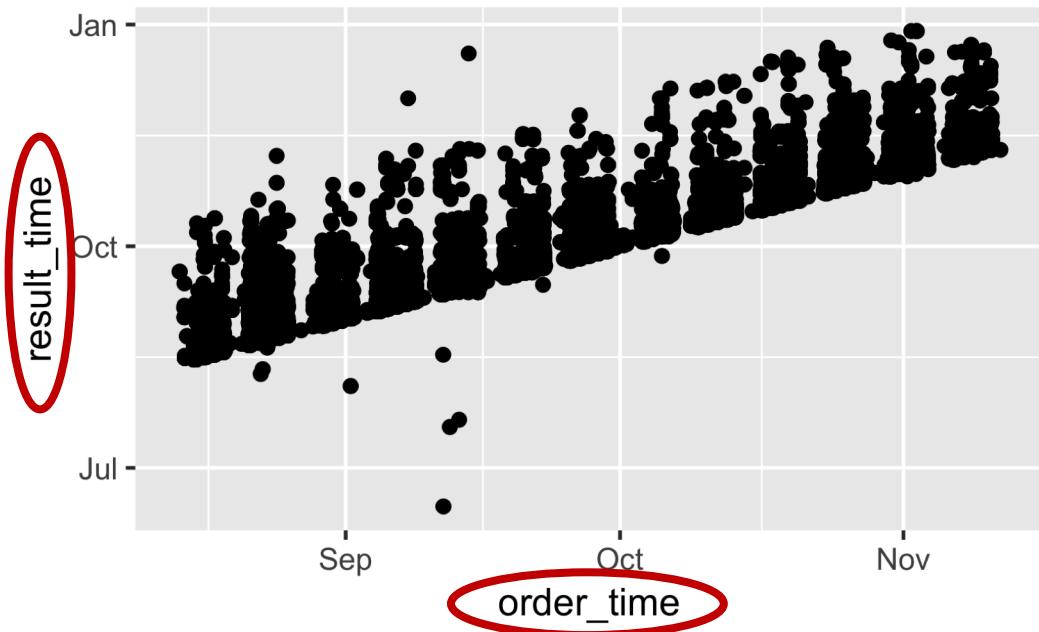
```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time, color = order_class_c_descr))
```



```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time), color = "blue")
```

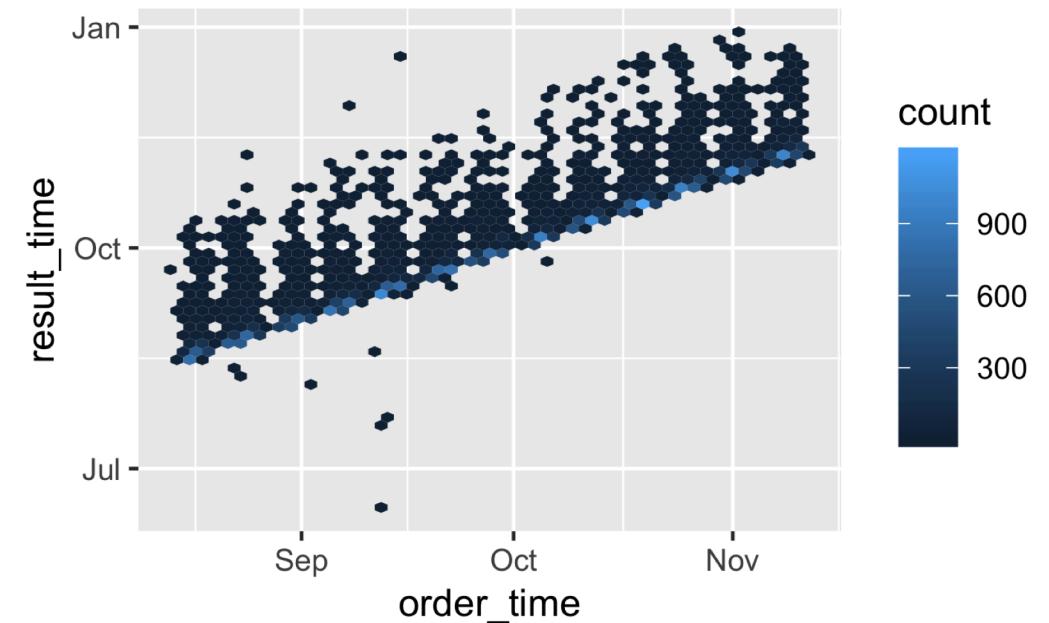
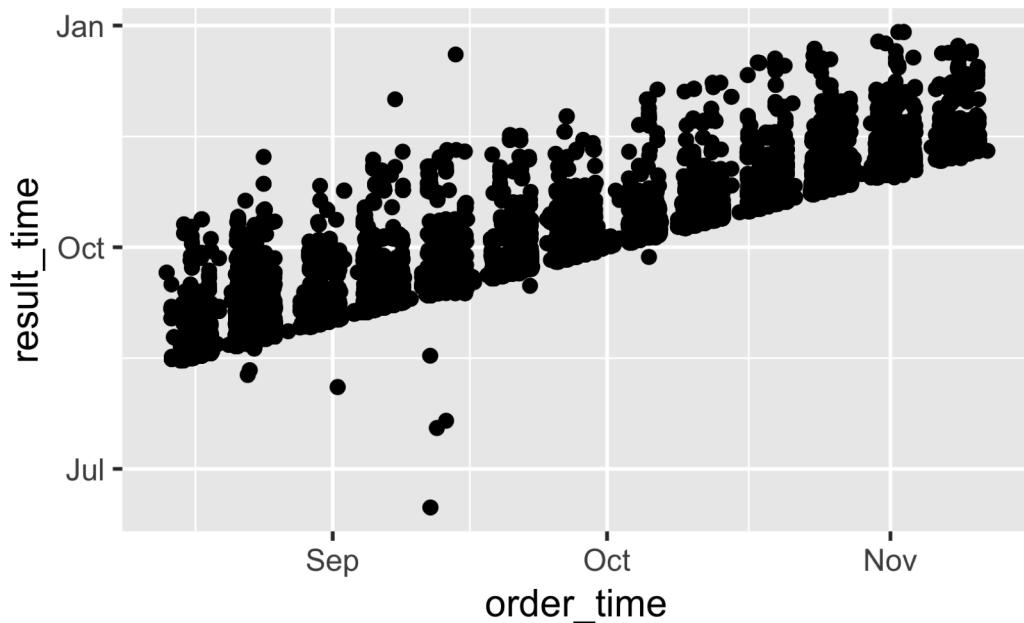
# Geom Functions

# How are these plots similar?



Same: x axis, y axis, data

# How are these plots different?



Different **geometric object** (“geom”) used to represent the data

```
ggplot(data = data_frame) +  
  geom_function(mapping = aes(mappings))
```

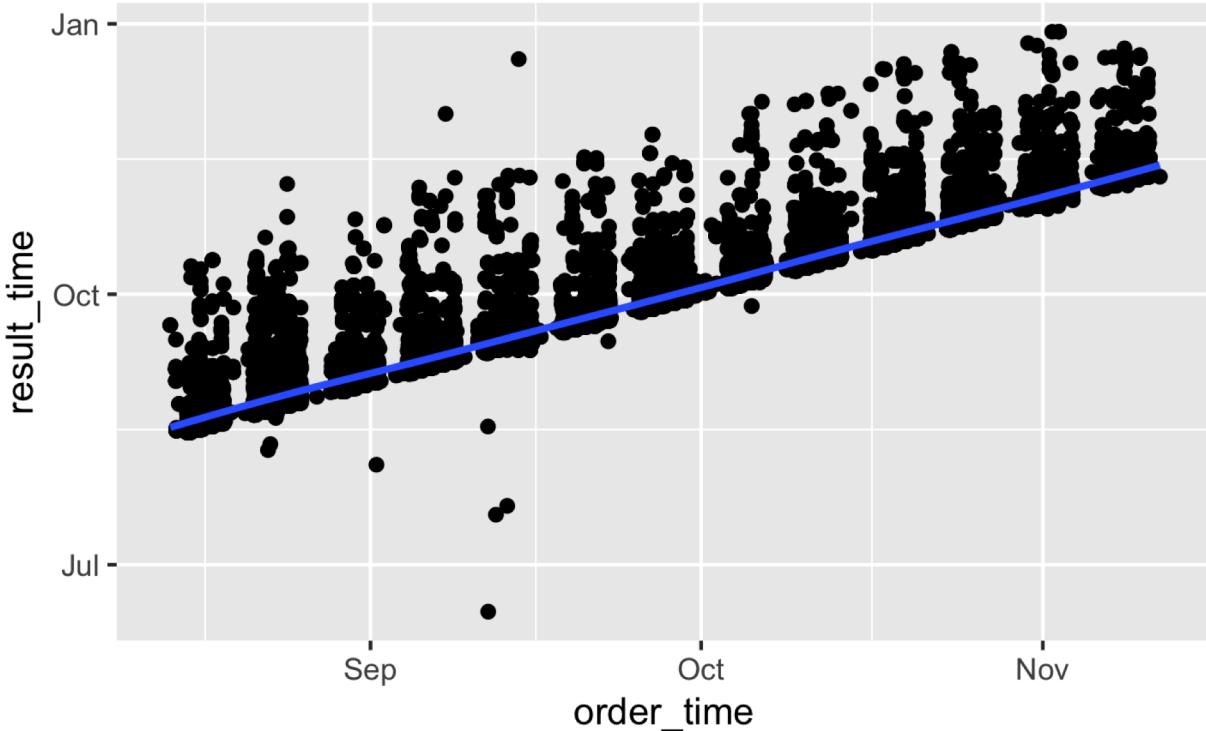


# Your Turn 5

Open 04-visualize.Rmd. Work through the exercises of the section titled “Your Turn 5.”

05 : 00

# Global vs Local Settings



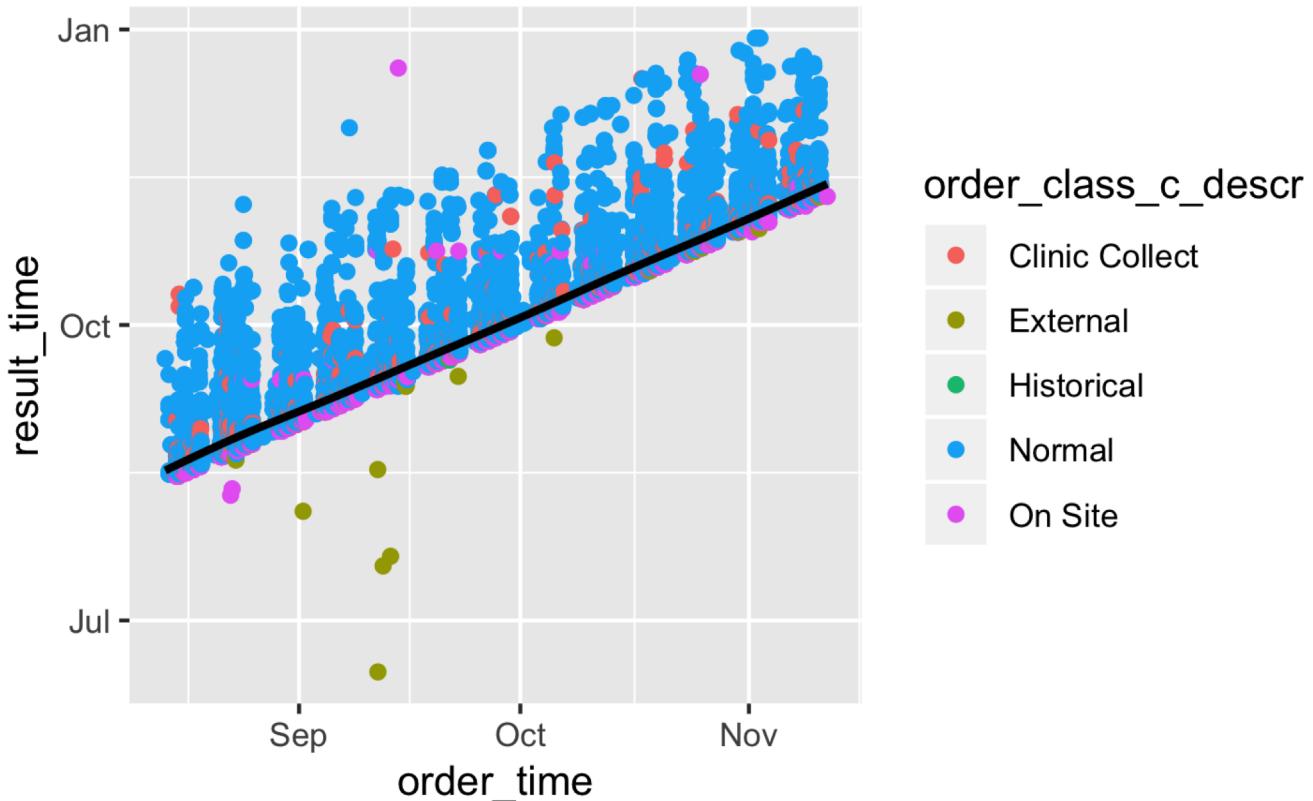
Inside of `geom_` function:  
apply **locally** to only the  
**current layer**

Inside of `ggplot()`:  
apply **globally** to  
**every layer**

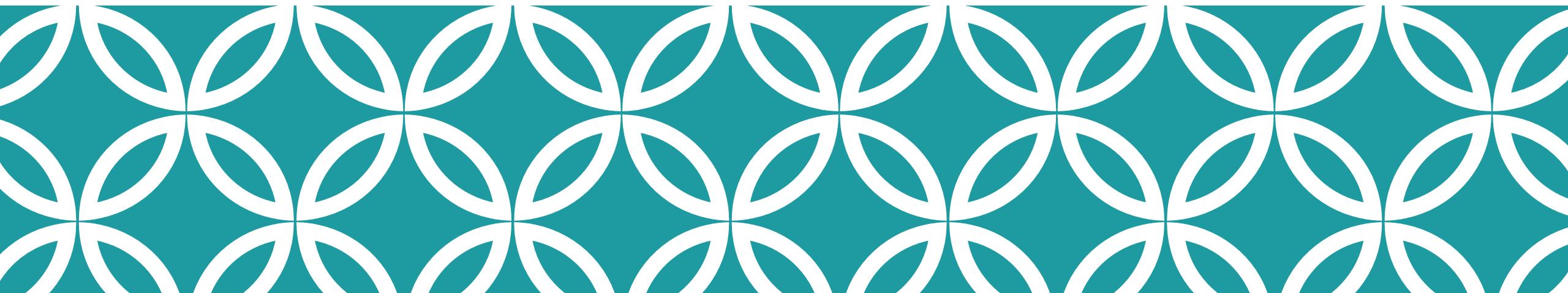
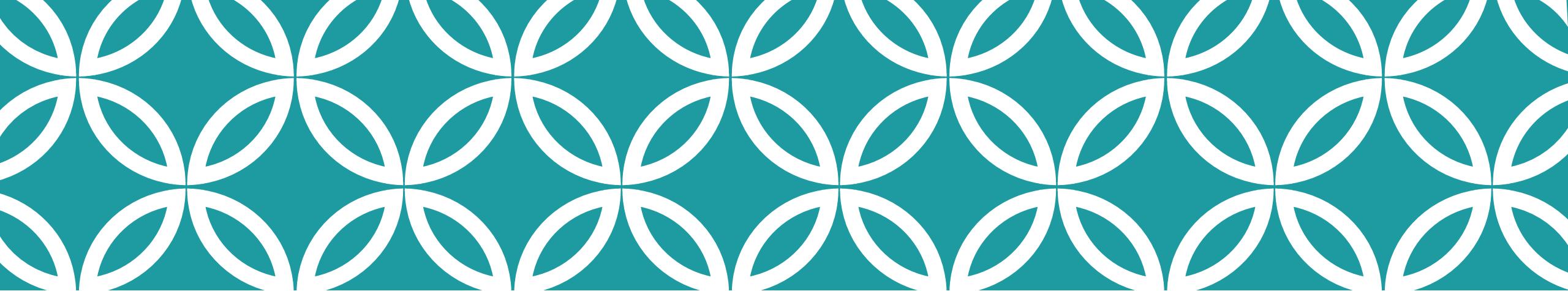
```
ggplot(data = orders) +  
  geom_point(mapping = aes(x = order_time, y = result_time)) +  
  geom_smooth(mapping = aes(x = order_time, y = result_time))
```

```
ggplot(data = orders, mapping = aes(x = order_time, y = result_time)) +  
  geom_point() +  
  geom_smooth()
```

# How would you make this plot?



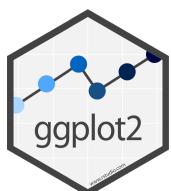
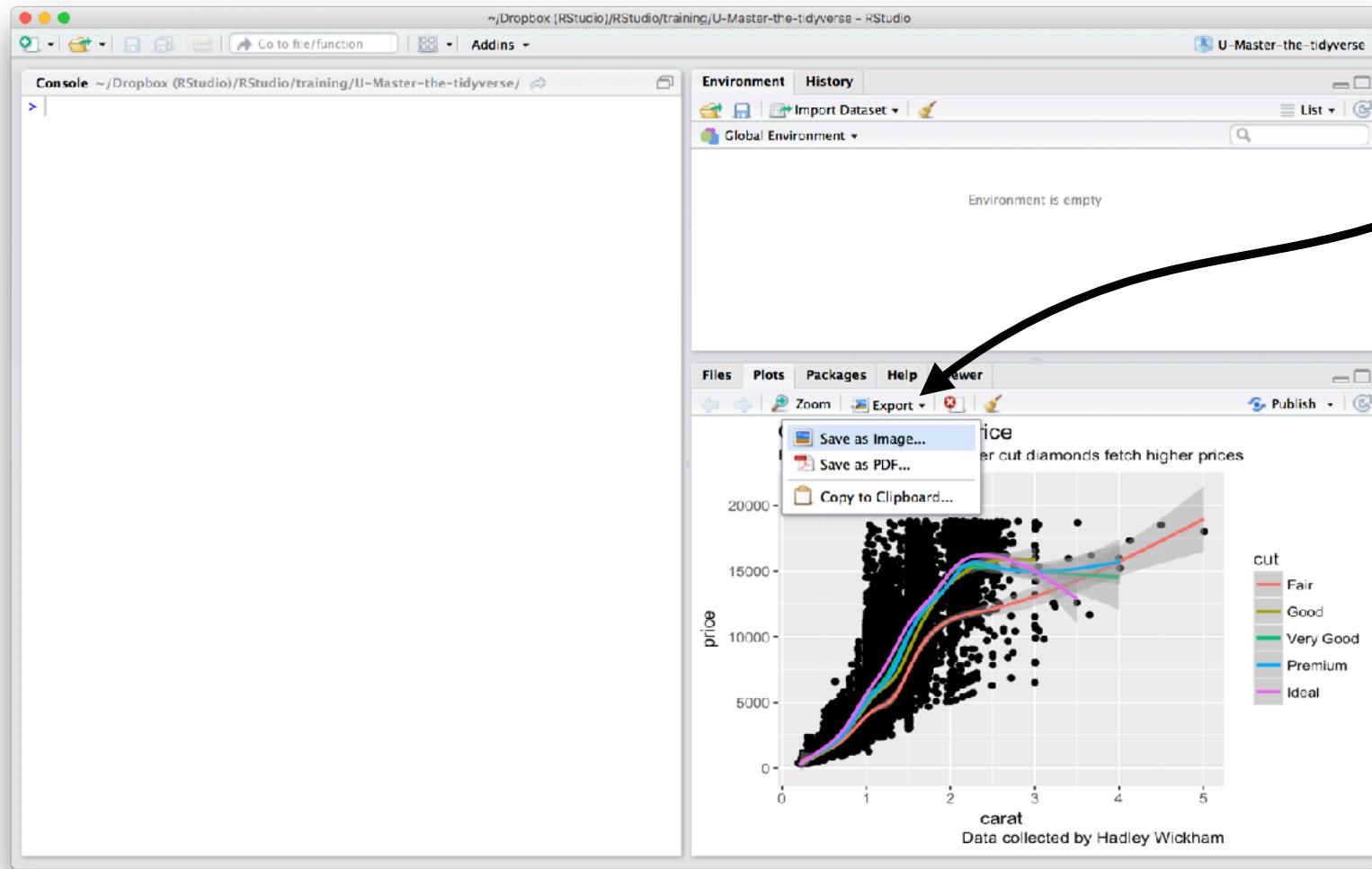
```
ggplot(data = orders, mapping = aes(x = order_time, y = result_time)) +  
  geom_point(mapping = aes(color = order_class_c_descr)) +  
  geom_smooth(color = "black")
```



**What Else?**

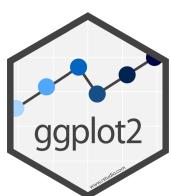
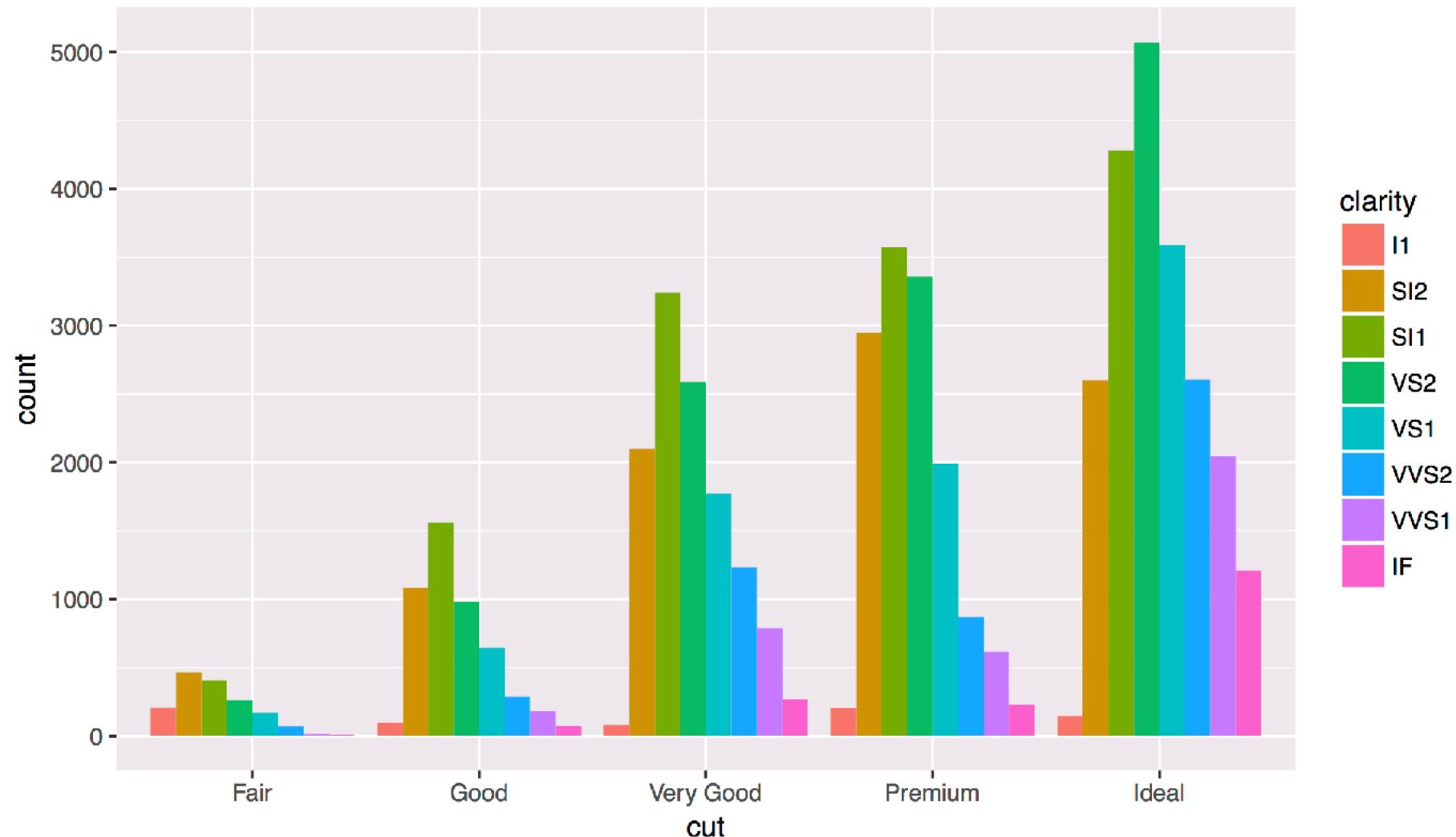
# Manually saving plots

Save plots manually with the export menu



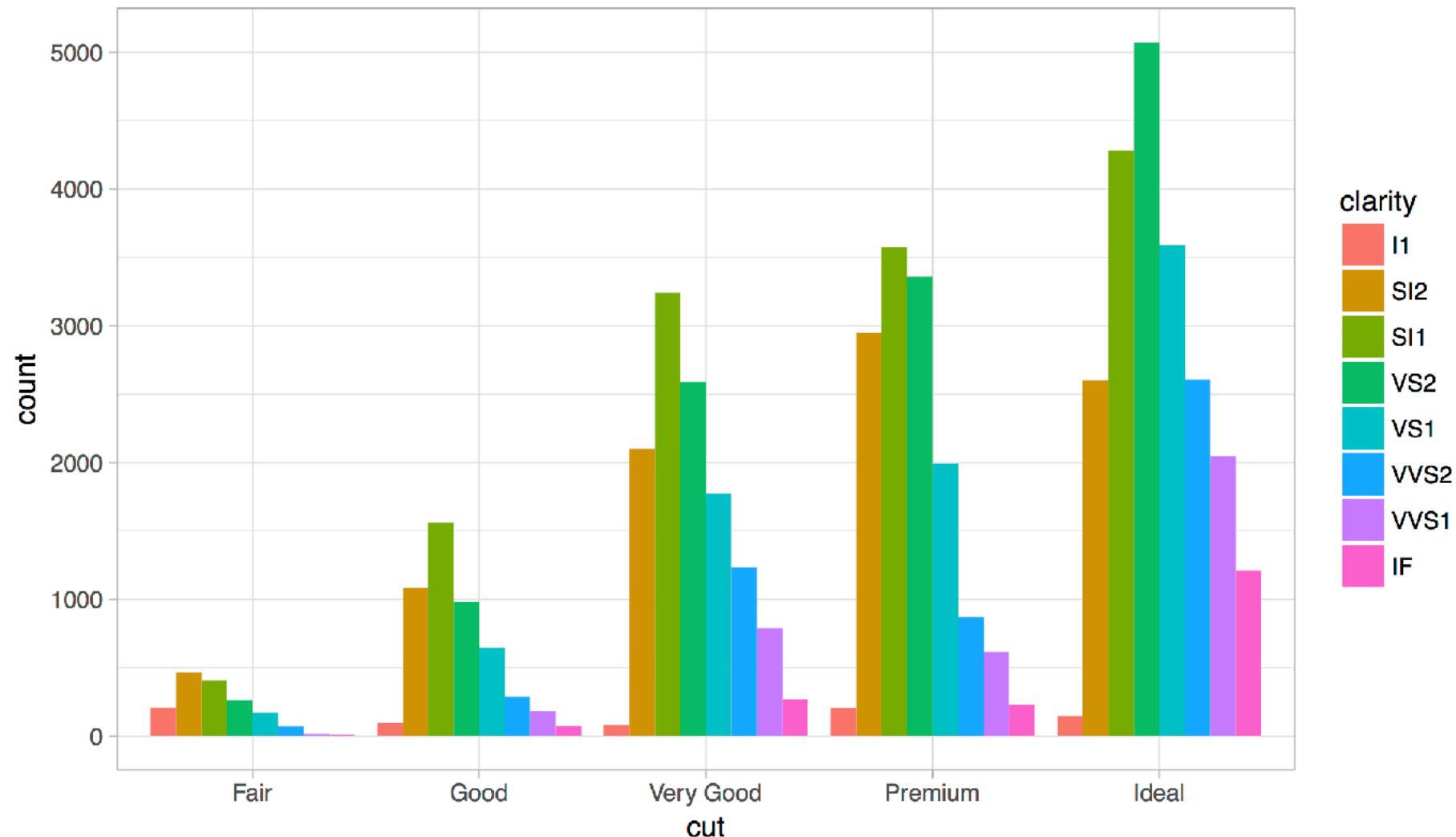
# Position Adjustments

How overlapping objects are arranged



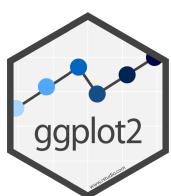
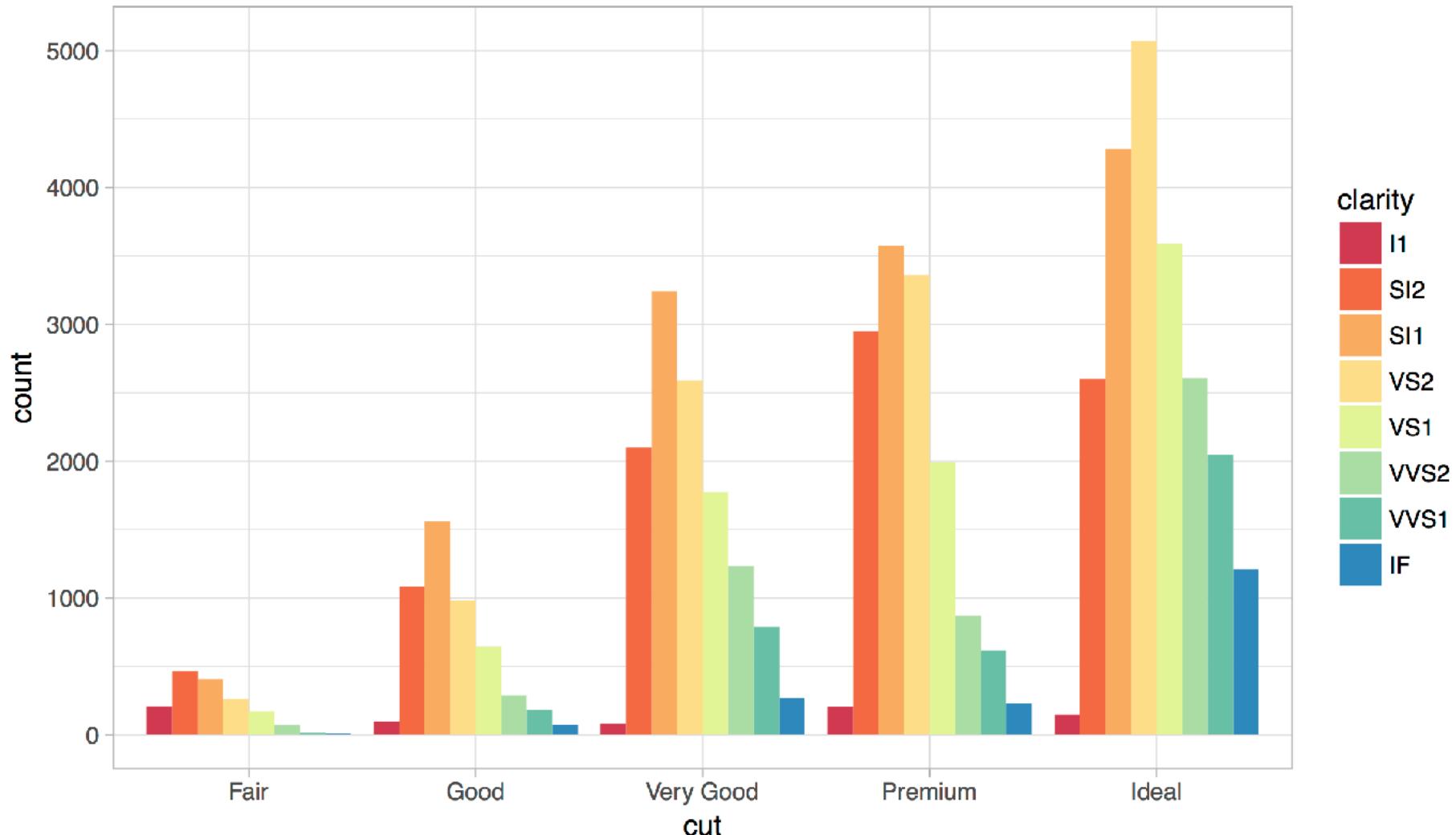
# Themes

## Visual appearance of non-data elements



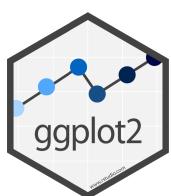
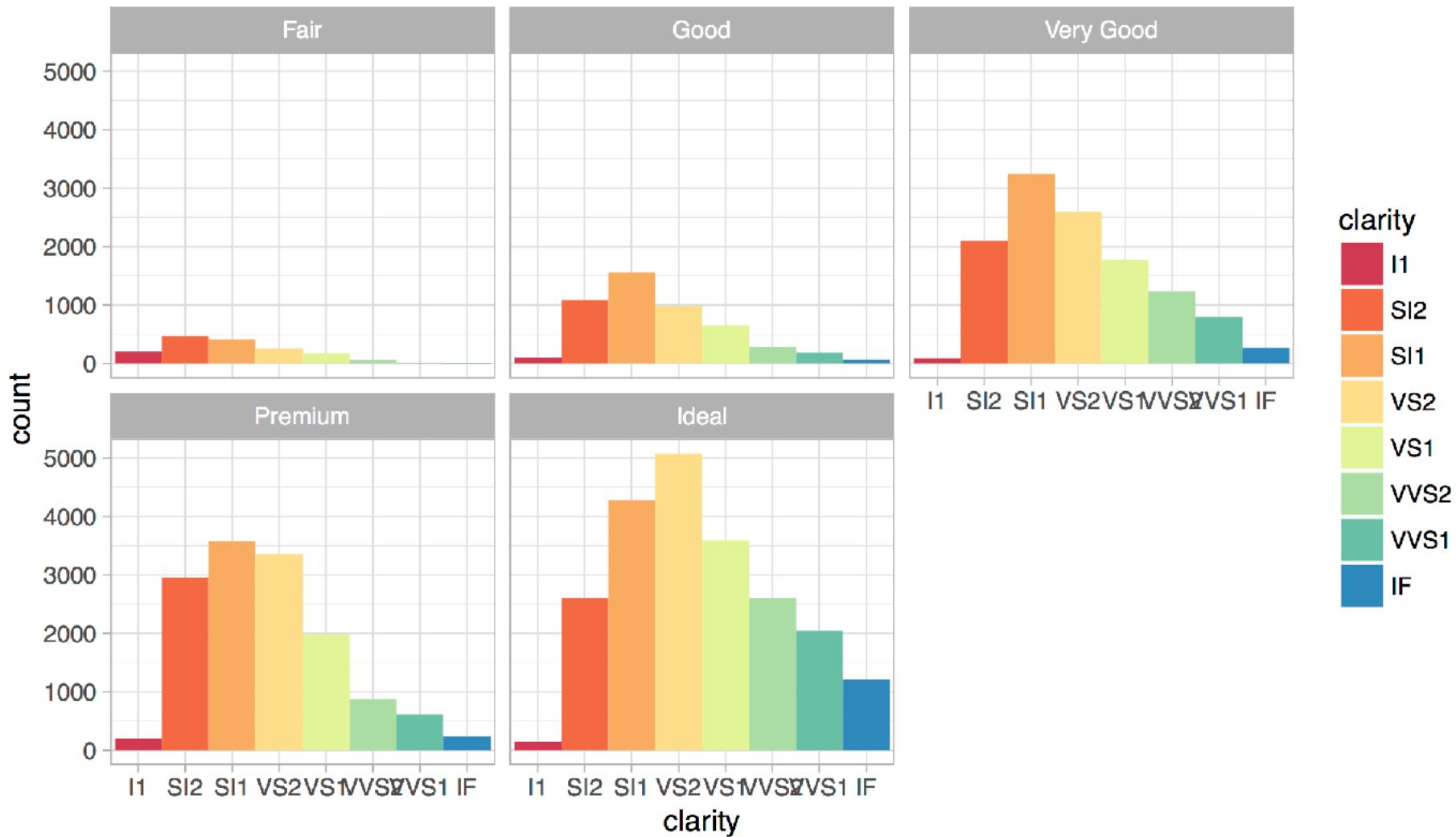
# Scales

Customize color scales, other mappings

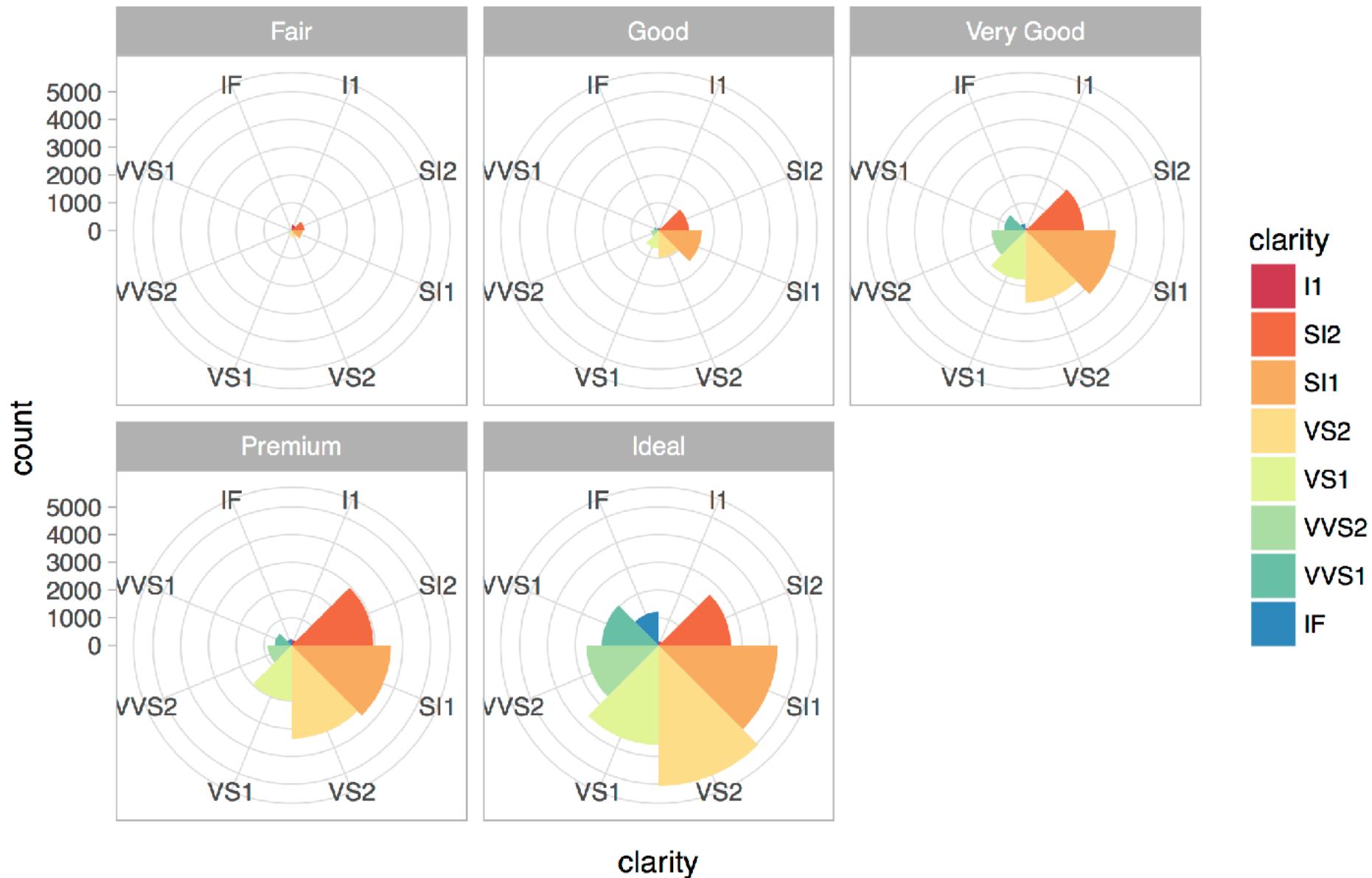


# Facets

Subplots that display subsets of the data.



# Coordinate systems



# Titles and captions

## Diamonds data

The data set is skewed towards ideal cut diamonds



```
ggplot(data = data_frame) +  
  geom_function(mapping = aes(mappings)) +  
  theme_function +  
  scale_function +  
  facet_function +  
  coordinate_function +  
  ...
```

Required

Optional