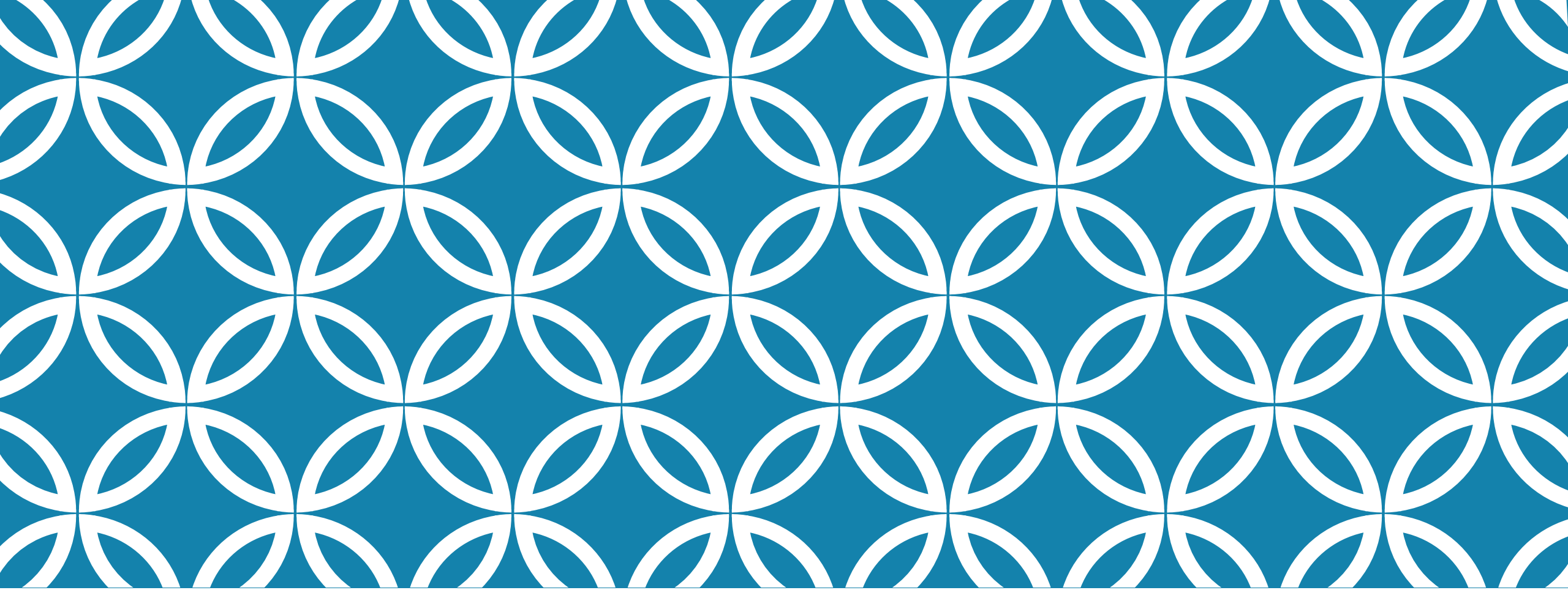


Introduction to R Workshop

Session 4

Visualize Data

May 6, 2019

7:00 am - 8:00 am	BREAKFAST - BALLROOM LOBBY - 2ND FLOOR
8:00 am - 8:10 am	Instructor Instructions
8:10 am - 9:50 am	Introduction to R and RStudio for Reproducible Reporting
9:50 am - 10:10 am	REFRESHMENT BREAK - BALLROOM LOBBY - 2ND FLOOR
10:10 am - 11:50 am	Data Wrangling
12:00 pm - 1:00 pm	LUNCH - BALLROOM LOBBY - 2ND FLOOR
1:00 pm - 2:50 pm	Data Understanding
2:50 pm - 3:10 pm	REFRESHMENT BREAK - BALLROOM LOBBY - 2ND FLOOR
3:10 pm - 5:00 pm	Exploratory Data Analysis

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?

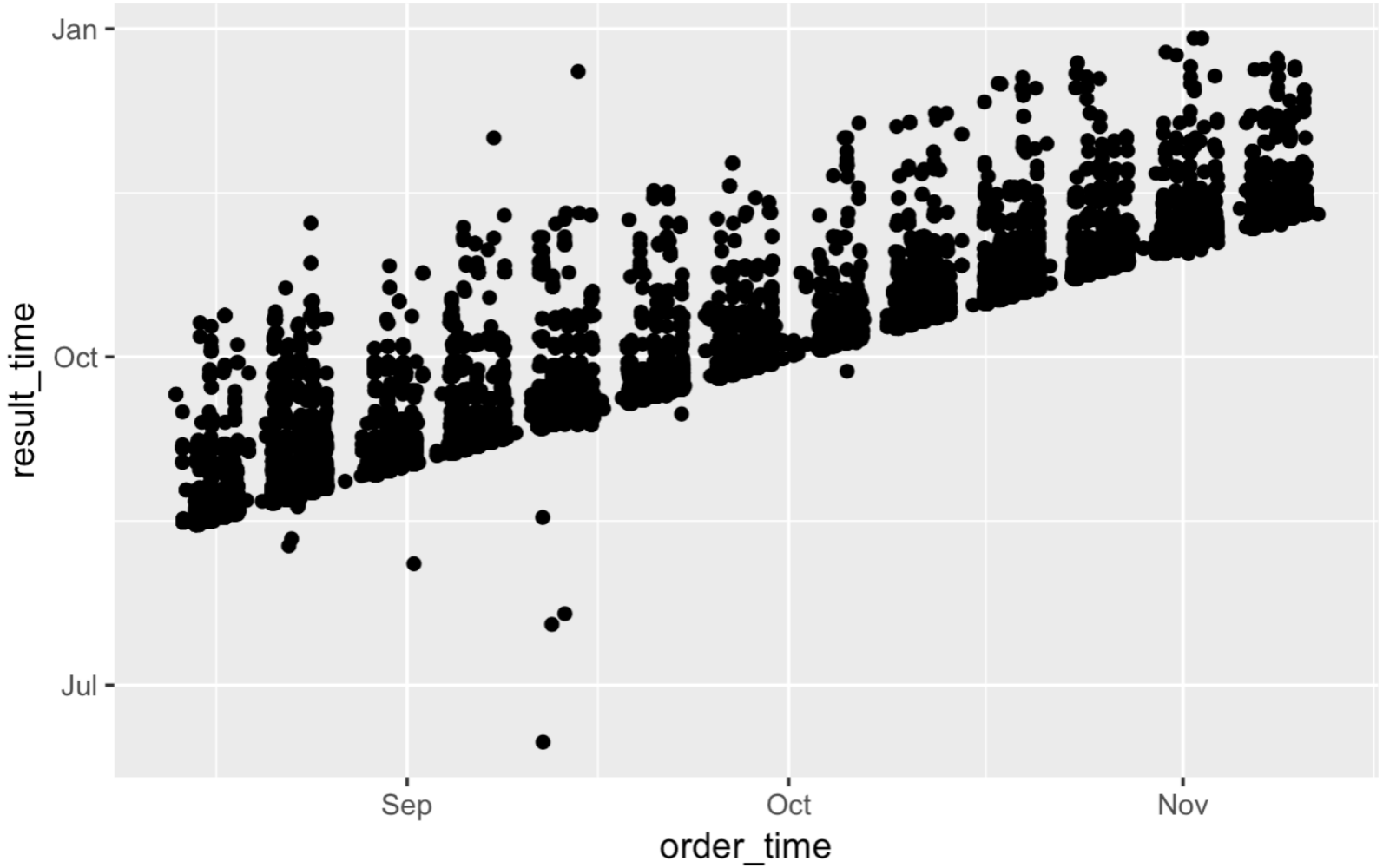
Pair up and discuss.

Your Turn 2

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

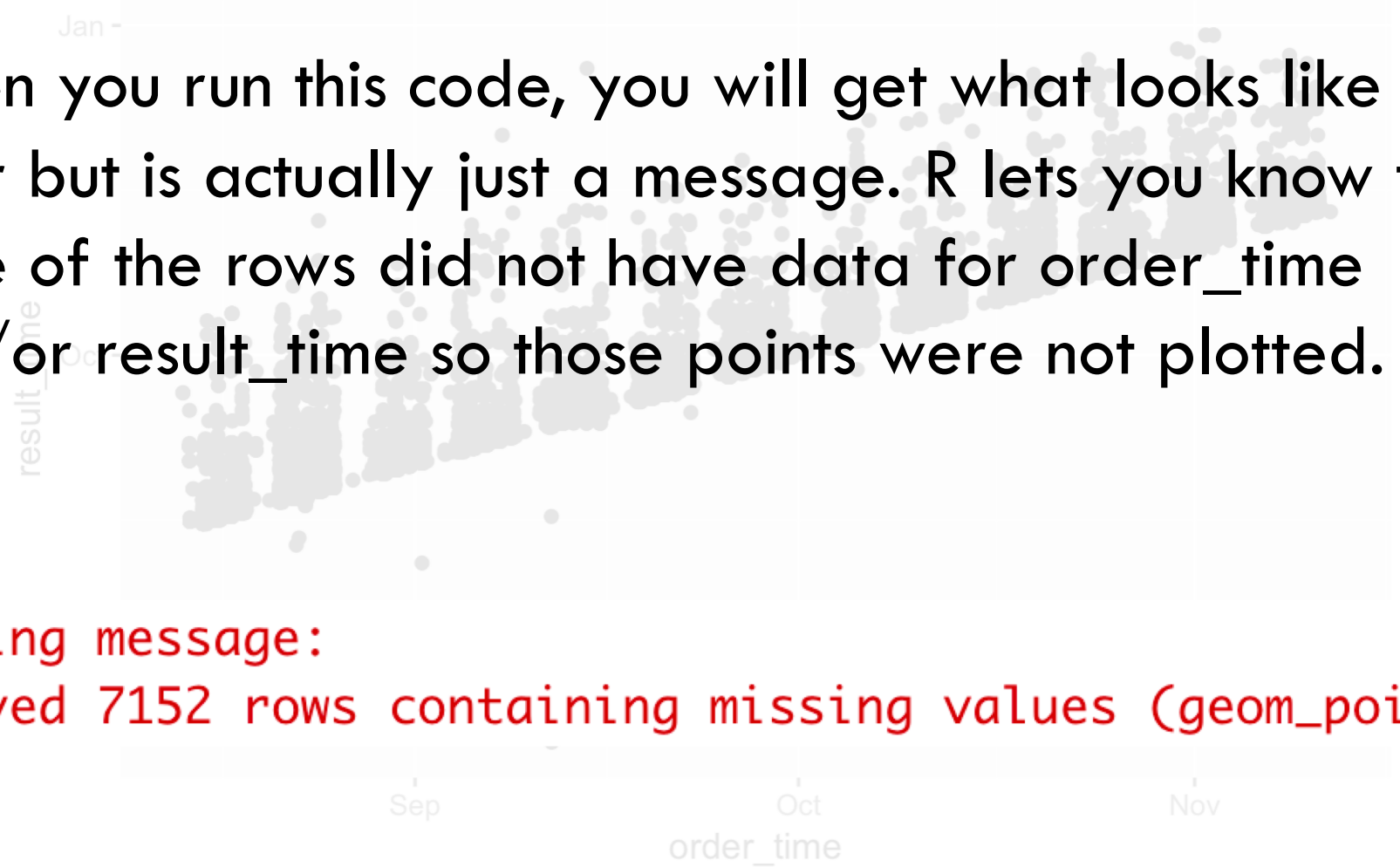
Pay attention to the spelling, capitalization, and parentheses!

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



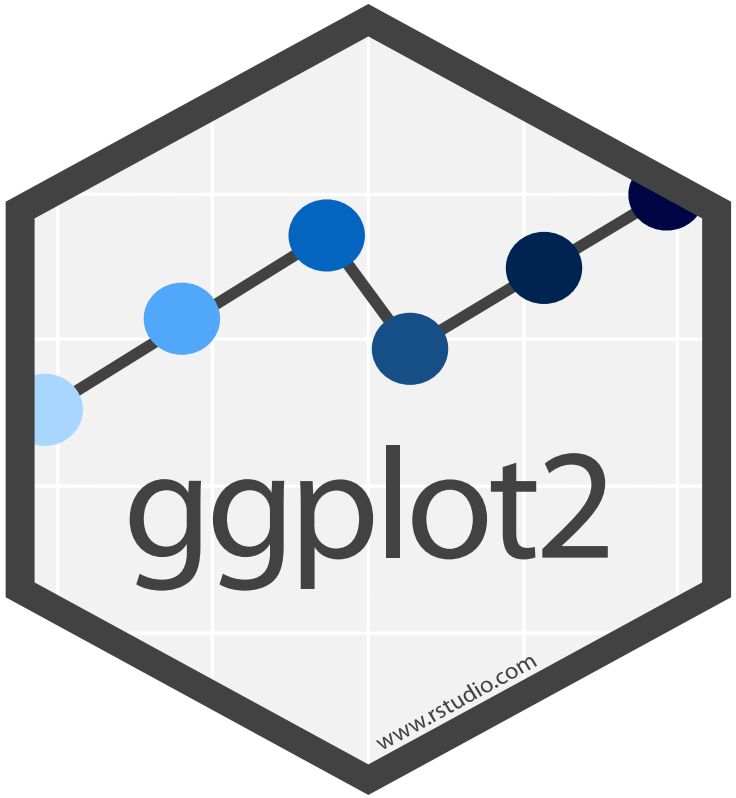

```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time))
```

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))
```


ggplot2
www.rstudio.com

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:

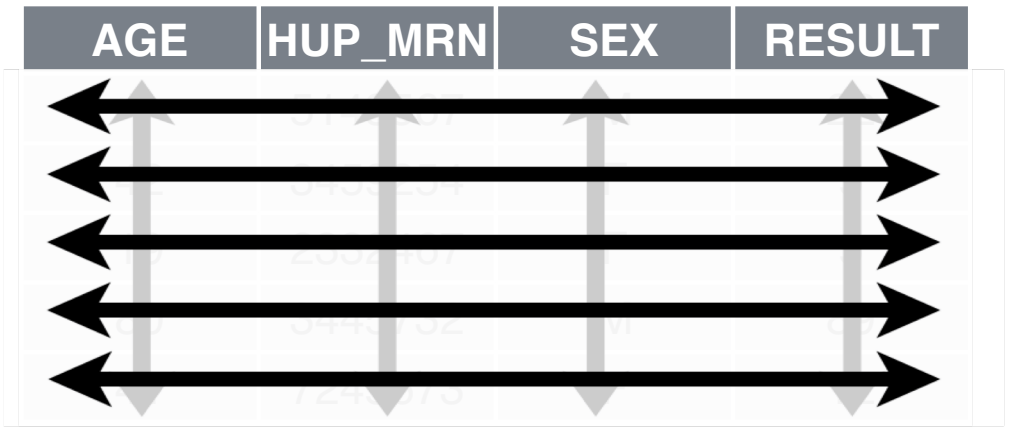
1. Choose a "tidy" **data** frame

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

2. Pick a "**geom**" function

3. Write aesthetic **mappings**

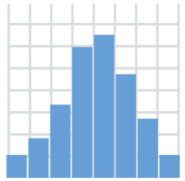
1. Pick a "Tidy" Data Frame



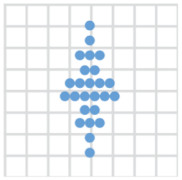
A data set is **tidy** if:

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

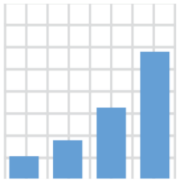
2. Choose a "Geom" Function



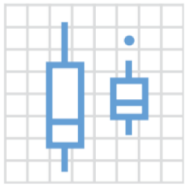
`geom_histogram()`



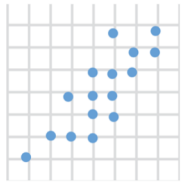
`geom_dotplot()`



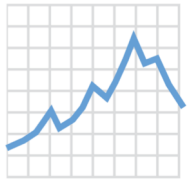
`geom_bar()`



`geom_boxplot()`



`geom_point()`

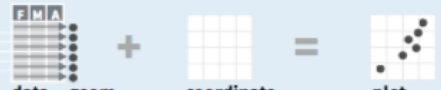


`geom_line()`

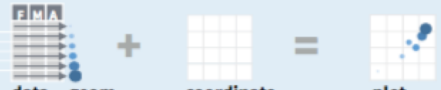
Data Visualization with ggplot2 : : CHEAT SHEET

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>) +                                  required
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
 stat = <STAT> , position = <POSITION> ) +                Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

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, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=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(ymin=unemploy - 900, ymax=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(yintercept = lat))
b + geom_vline(aes(xintercept = 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,

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, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

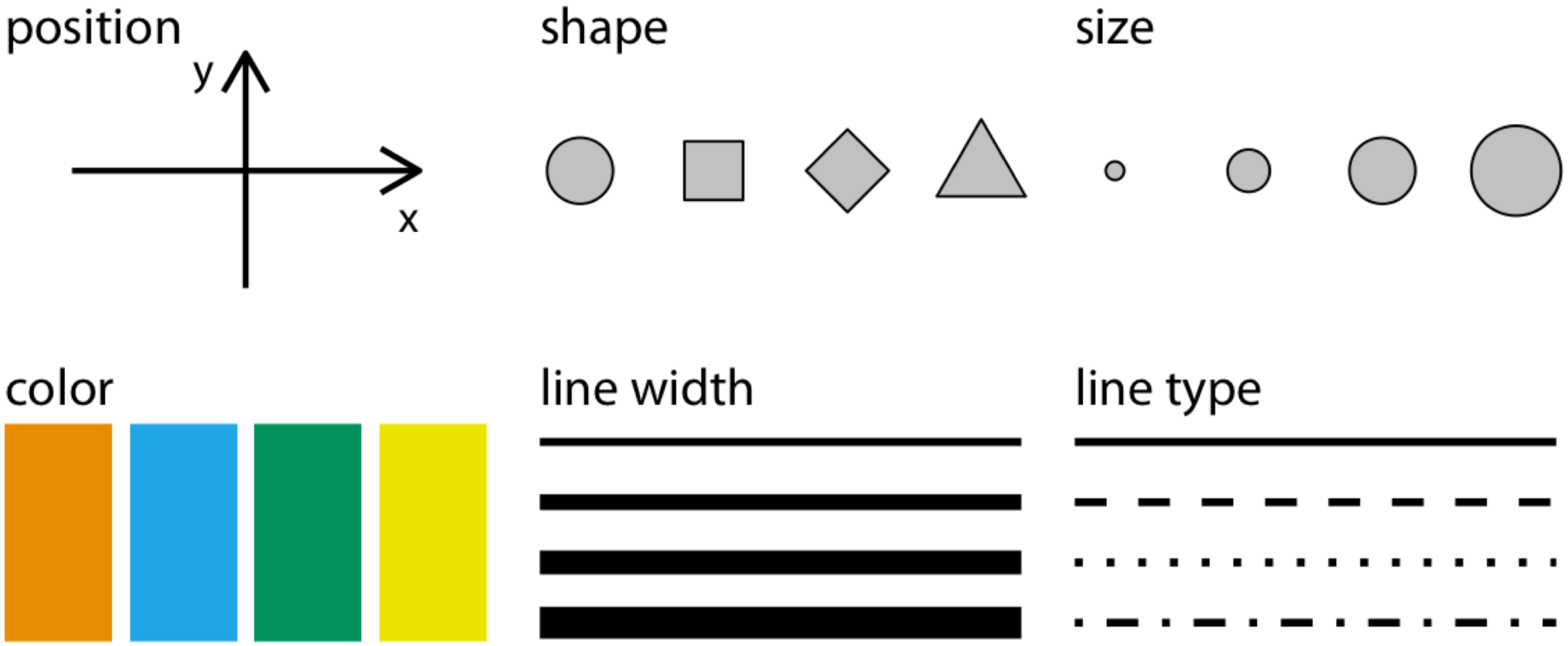
j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

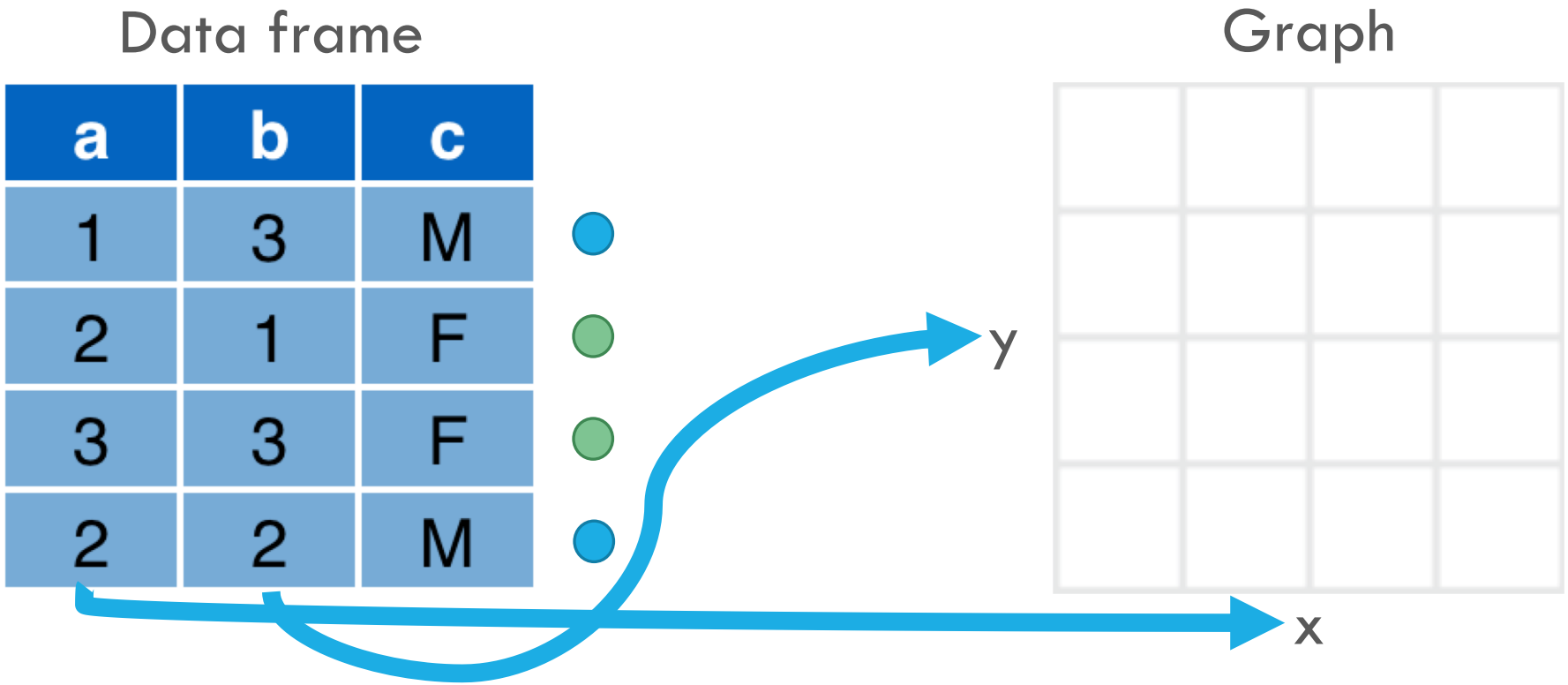
```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
```

Aesthetics



3. Write Aesthetic Mappings

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



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?

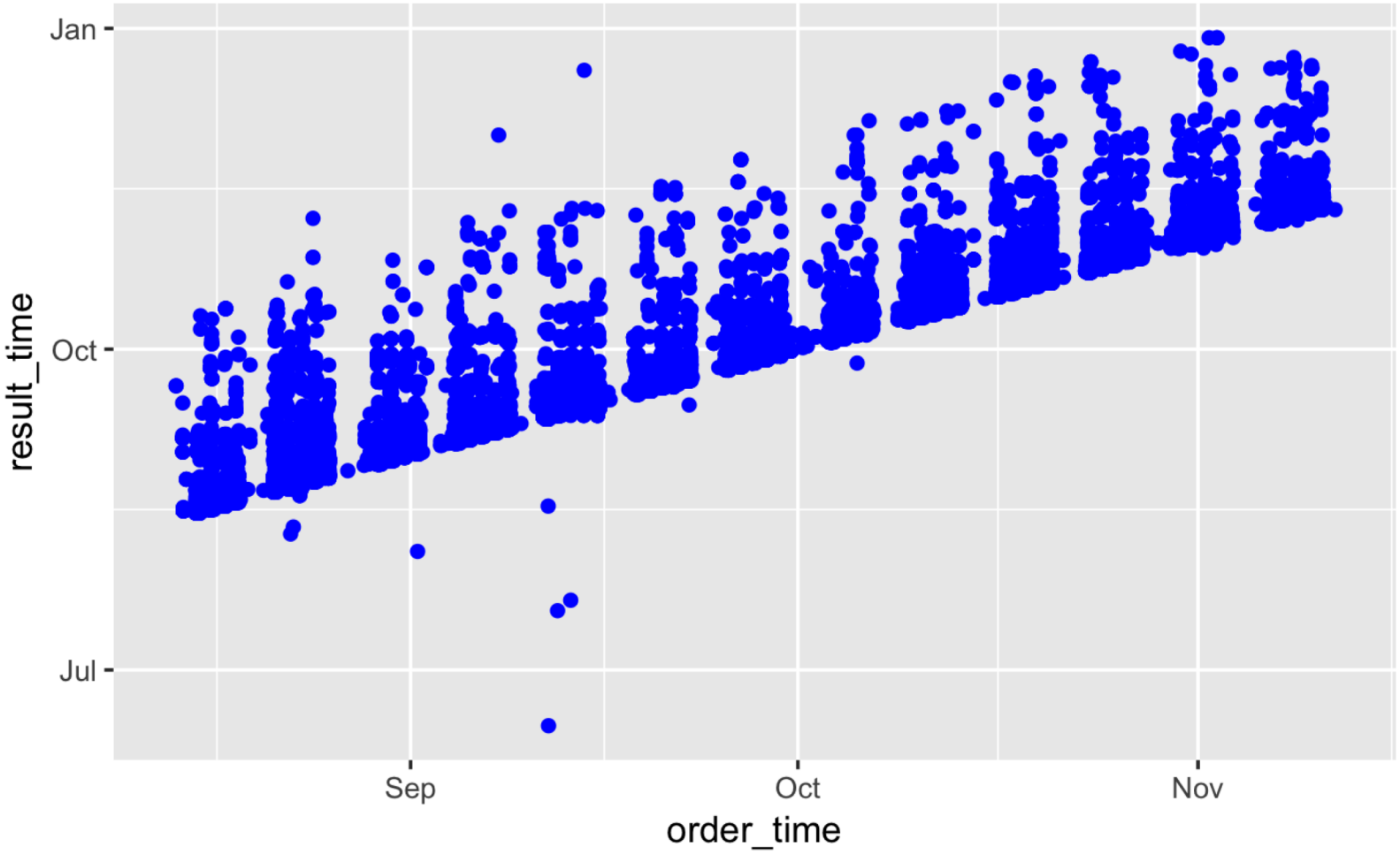
A. Color
B. Line type
C. Shape
D. B + C
E. A + B + C

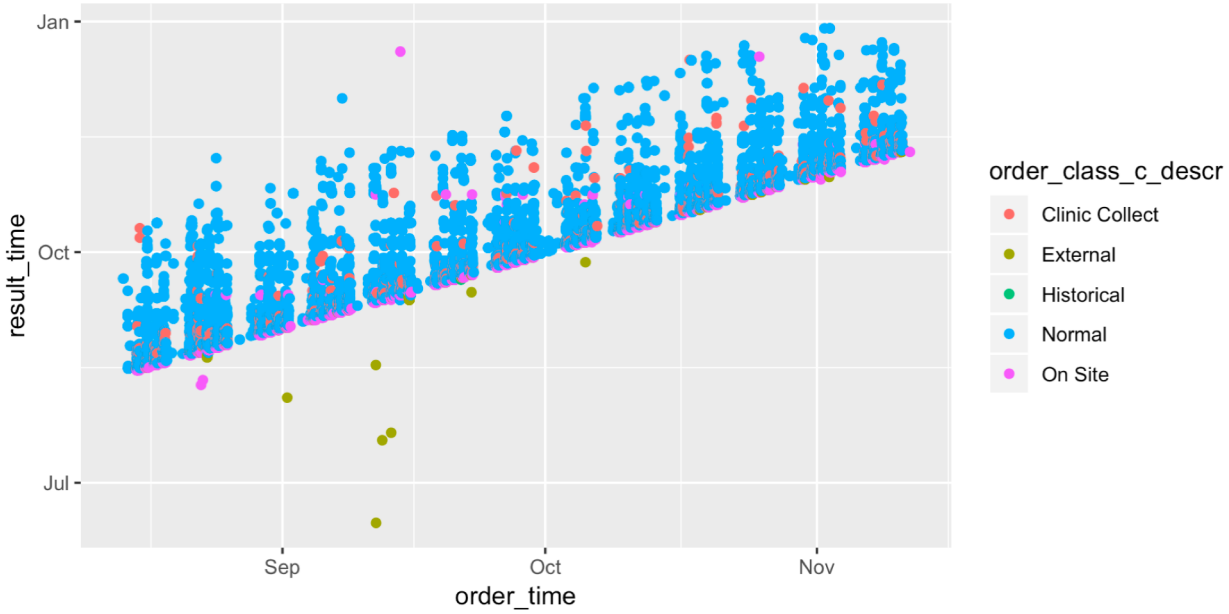
Your Turn 4

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

Setting vs Mapping Aesthetics

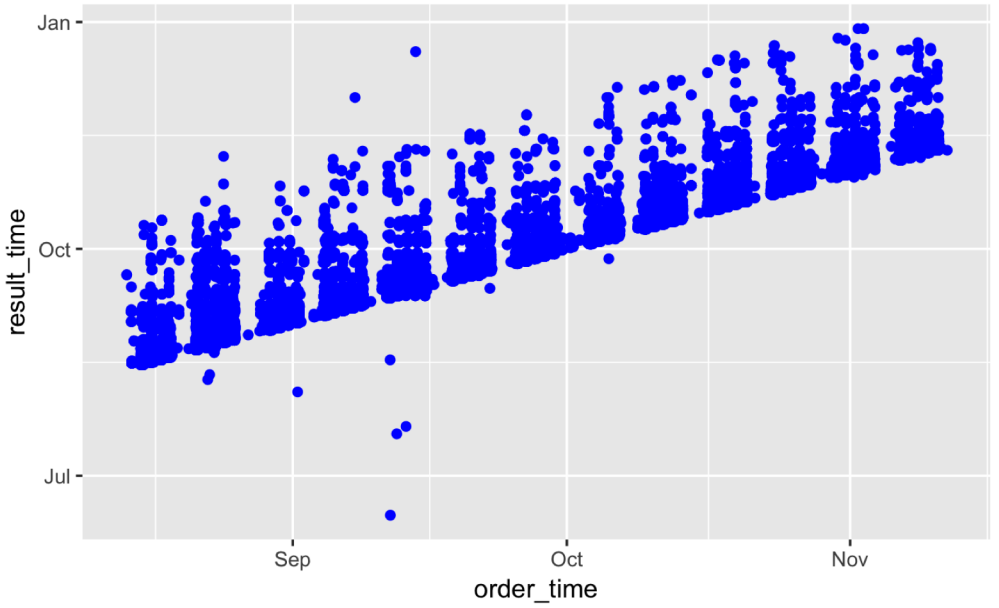
How would you make this plot?





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))
```

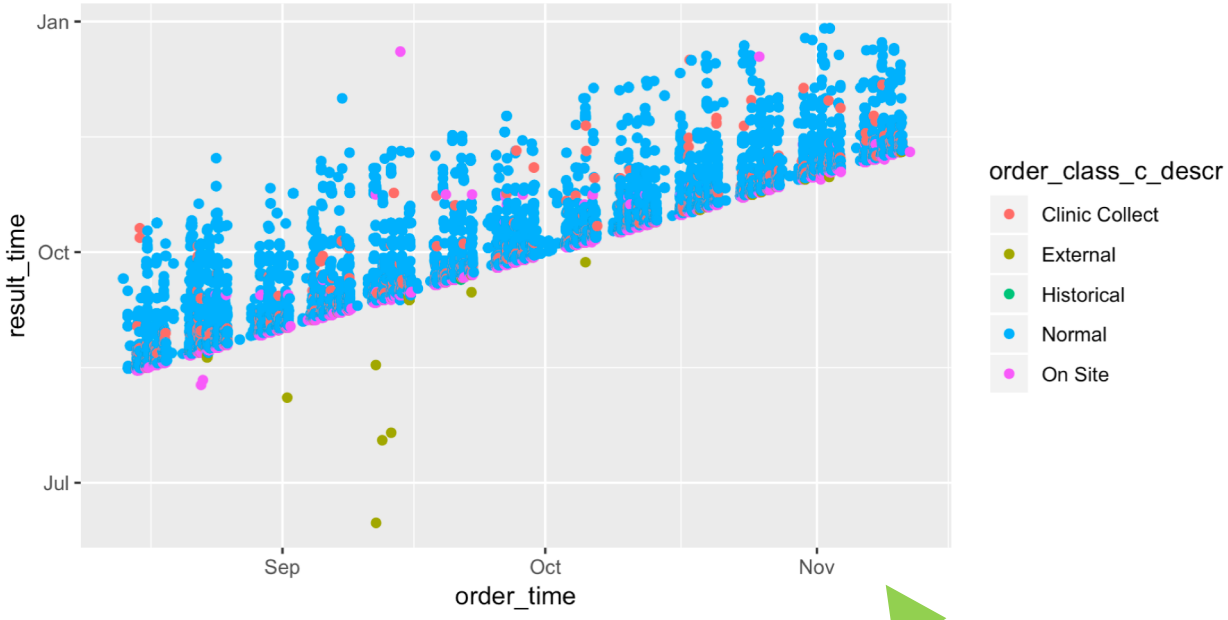


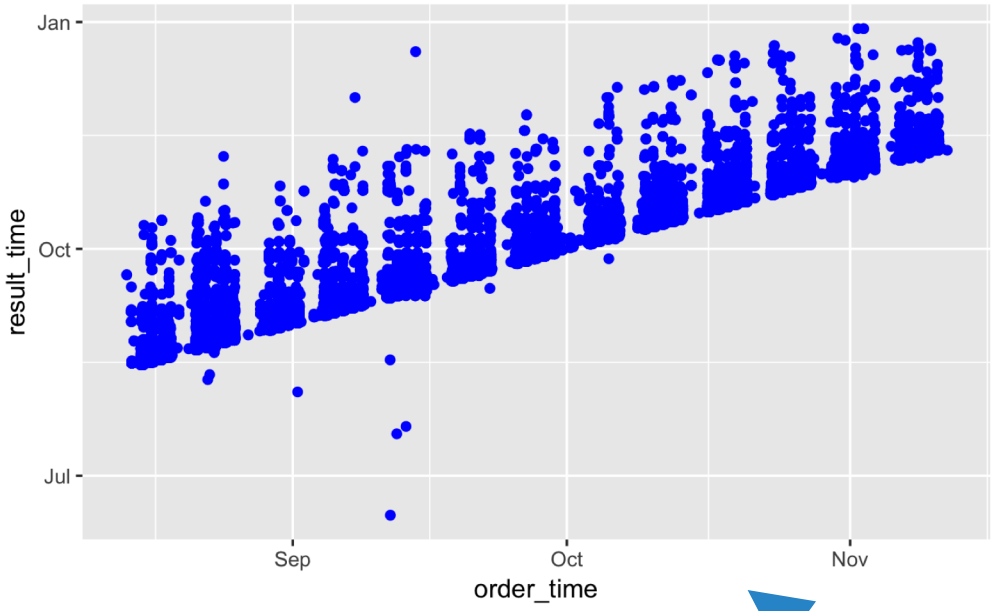
Outside of **aes()**:

set an aesthetic to

a **value**

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



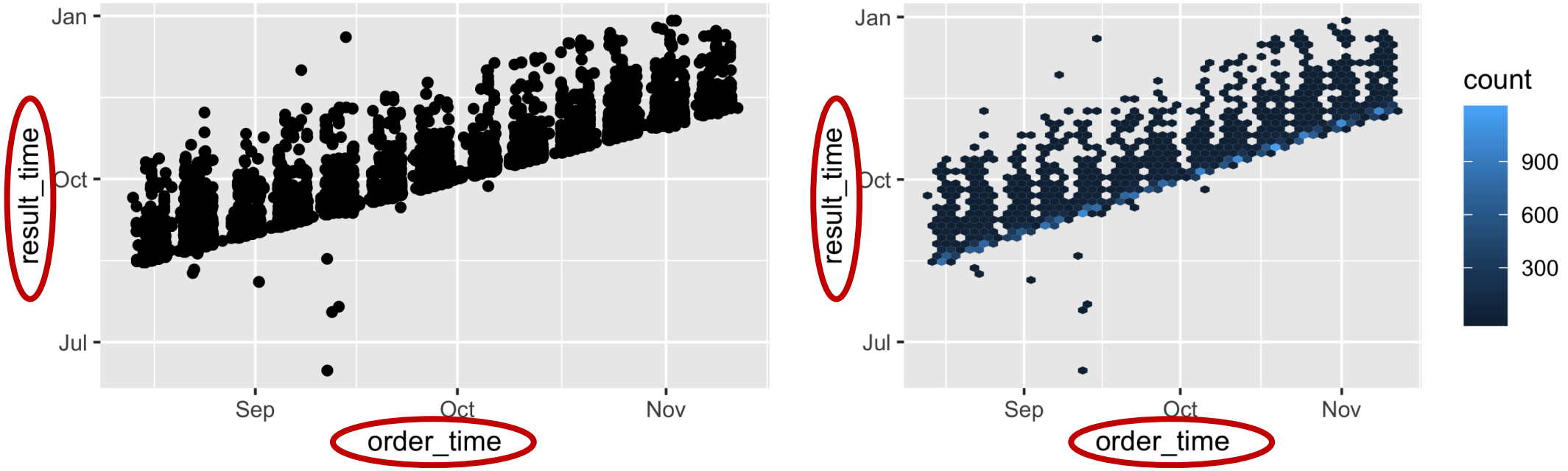



```
ggplot(data = orders) +
  geom_point(mapping = aes(x = order_time, y = result_time, color = order_class_c_descr))
```

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

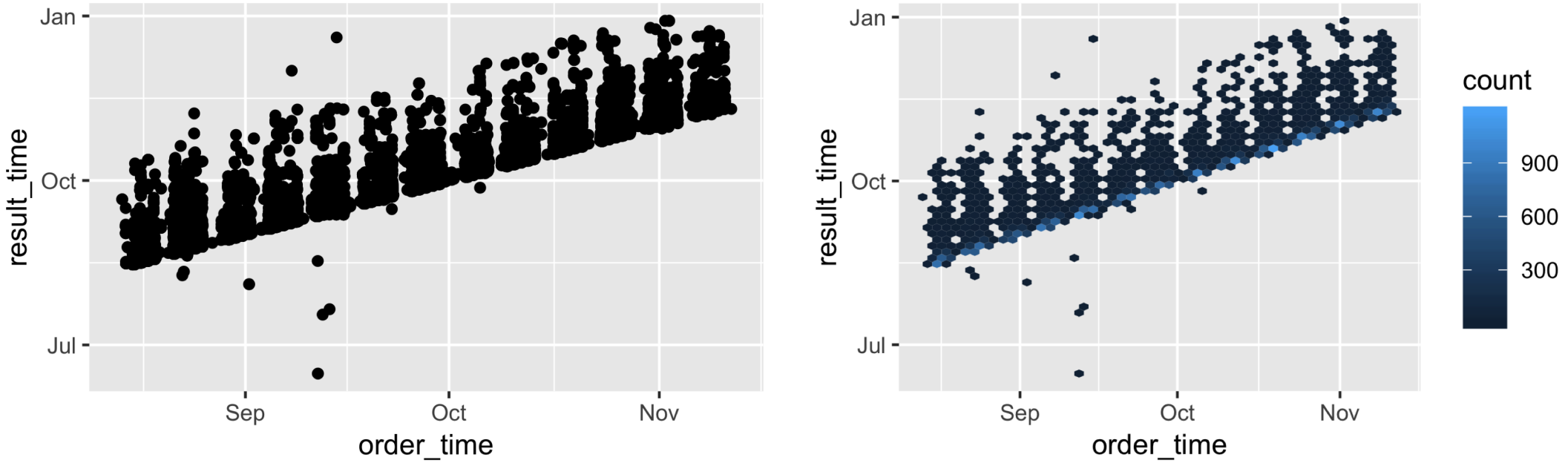
Geom Functions

How are these plots similar?



Same: x axis, y axis, data

How are these plots different?



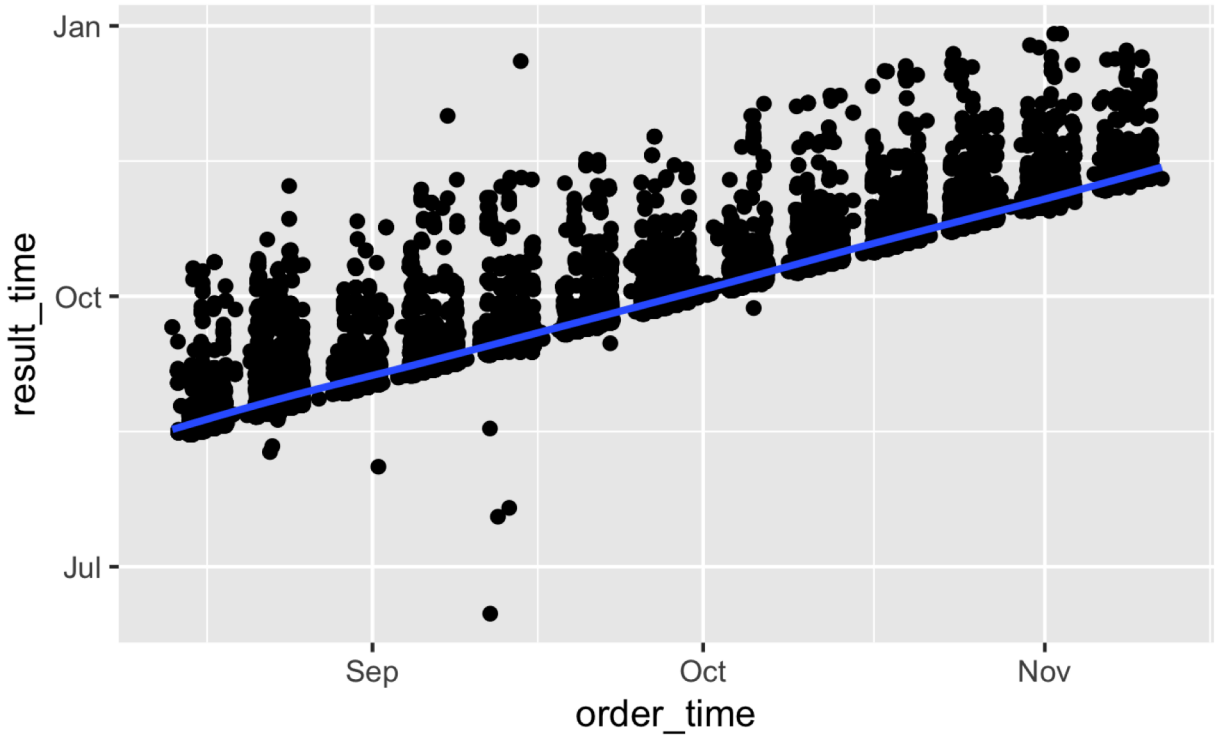
Different **geometric object** ("geom") used to represent the data

```
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."

Global vs Local Settings



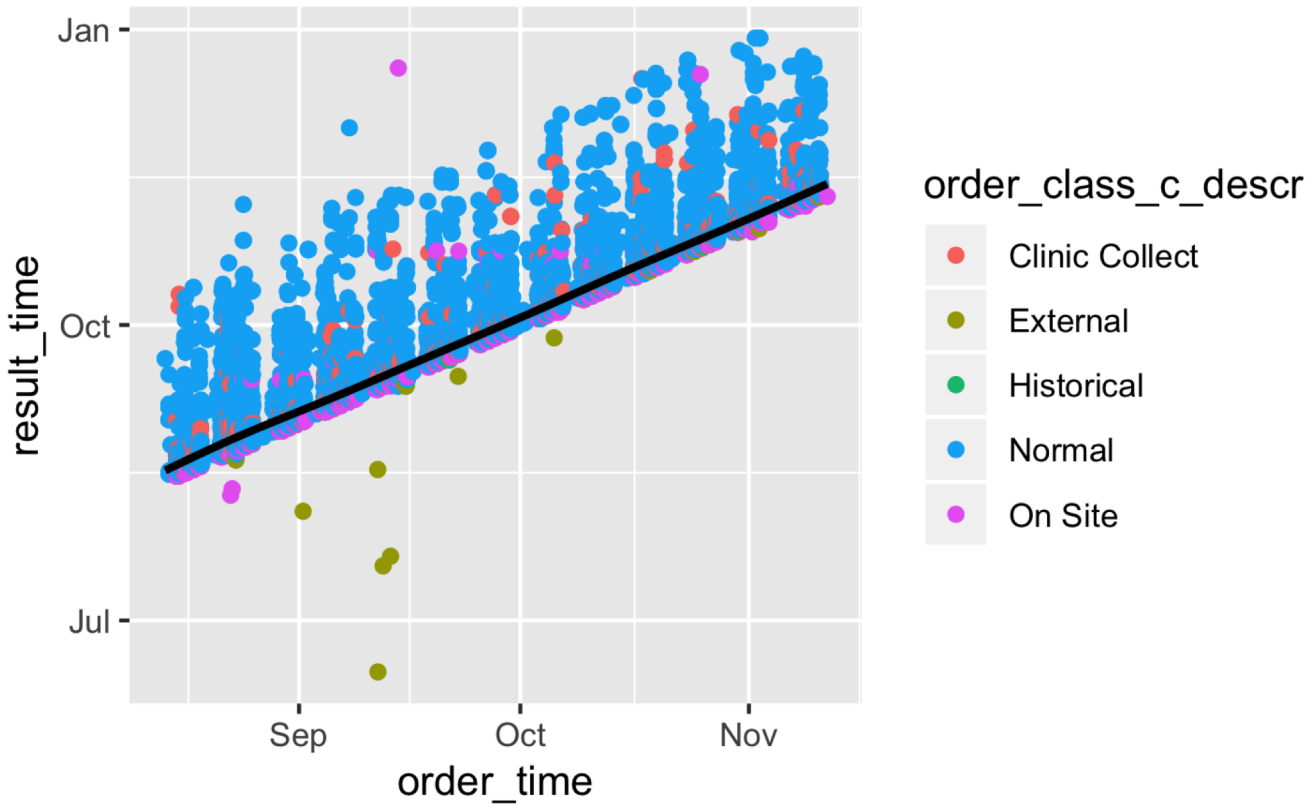
Inside of geom_ function: apply **locally** to only the **current layer**

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?

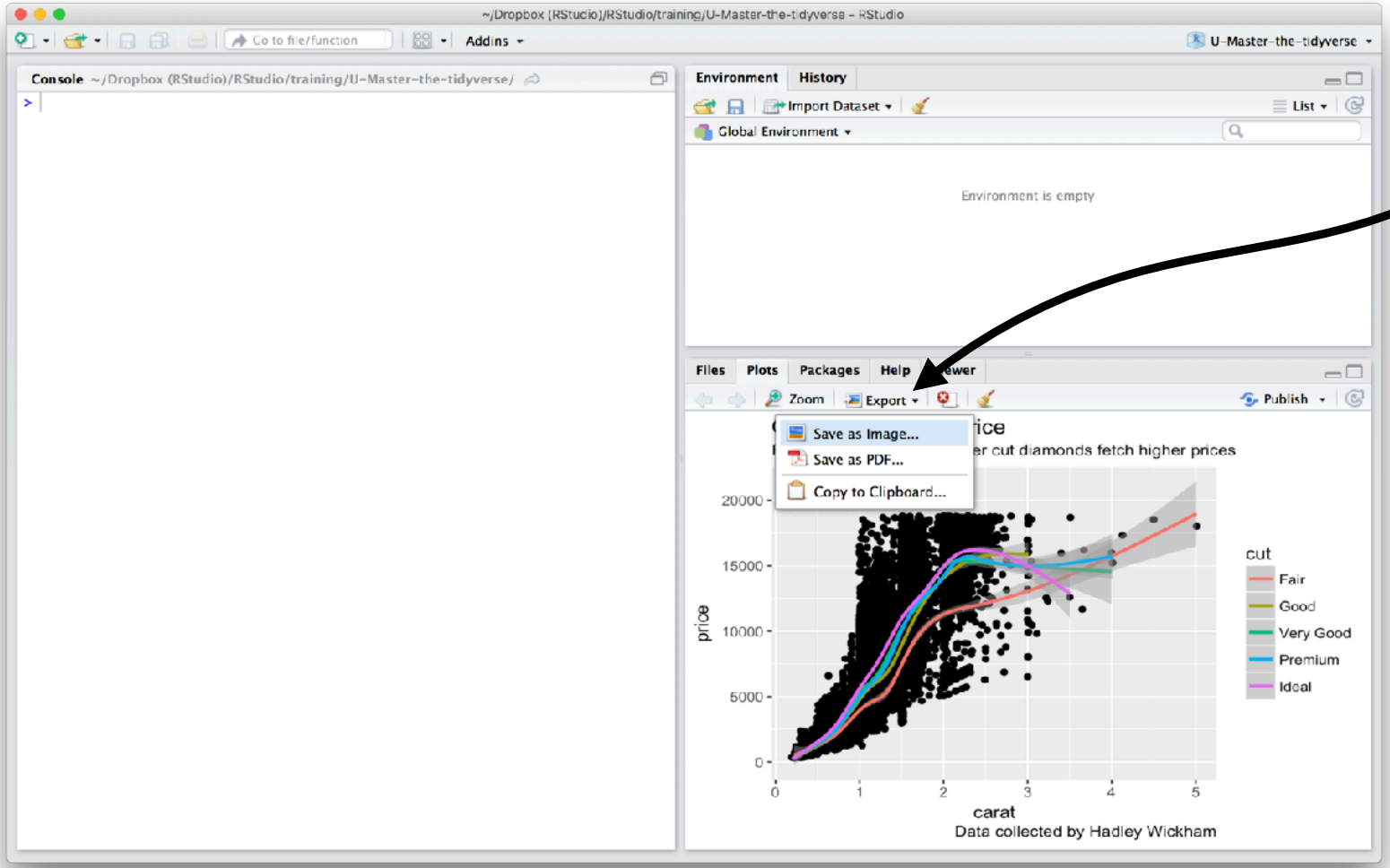



```
ggplot(data = orders, mapping = aes(x = order_time, y = result_time)) +
  geom_point(mapping = aes(color = order_class_c_descr)) +
  geom_smooth(color = "black")
```

What Else?

Manually saving plots

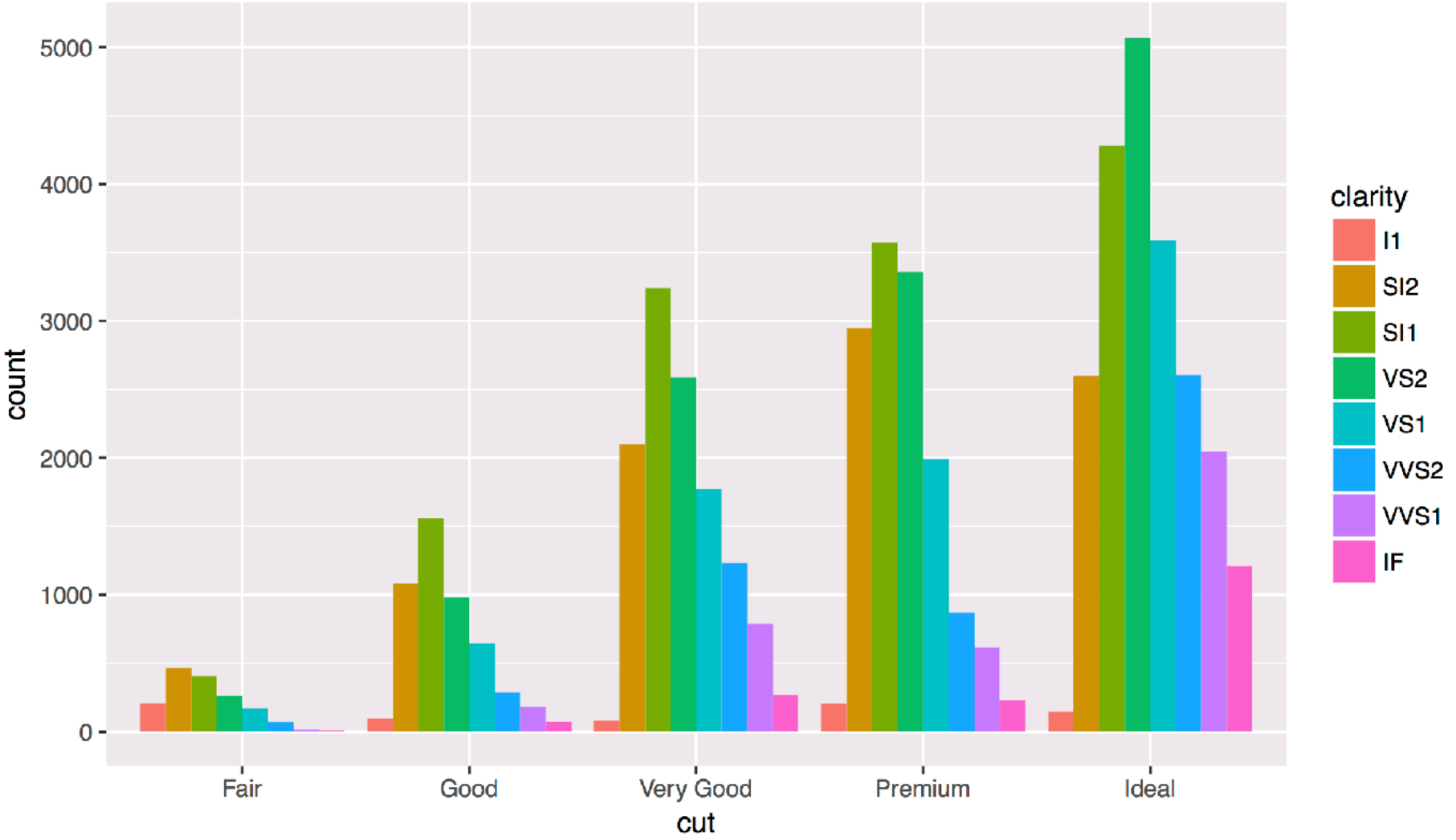
Save plots manually with the export menu



ggplot2

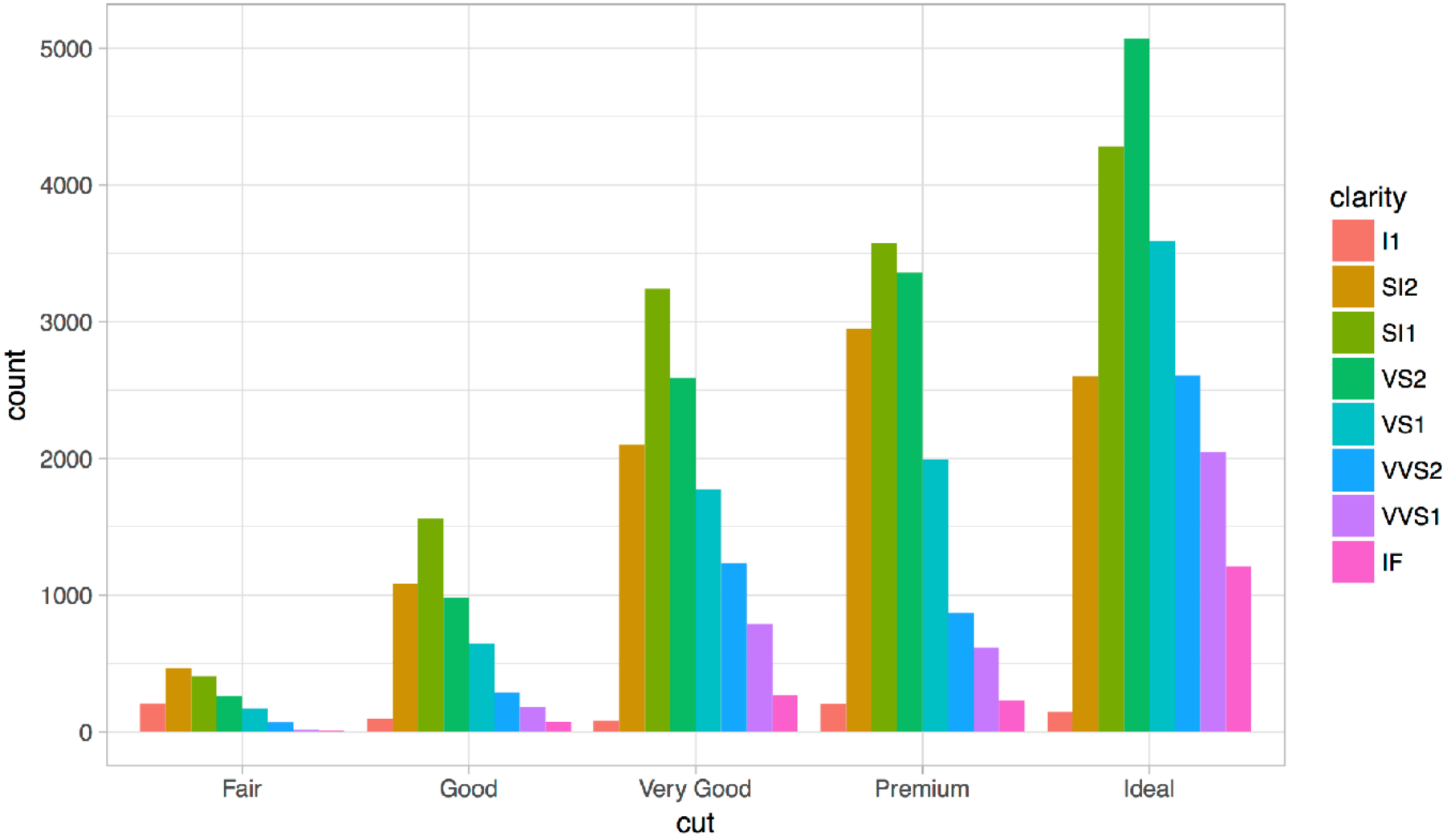
Position Adjustments

How overlapping objects are arranged



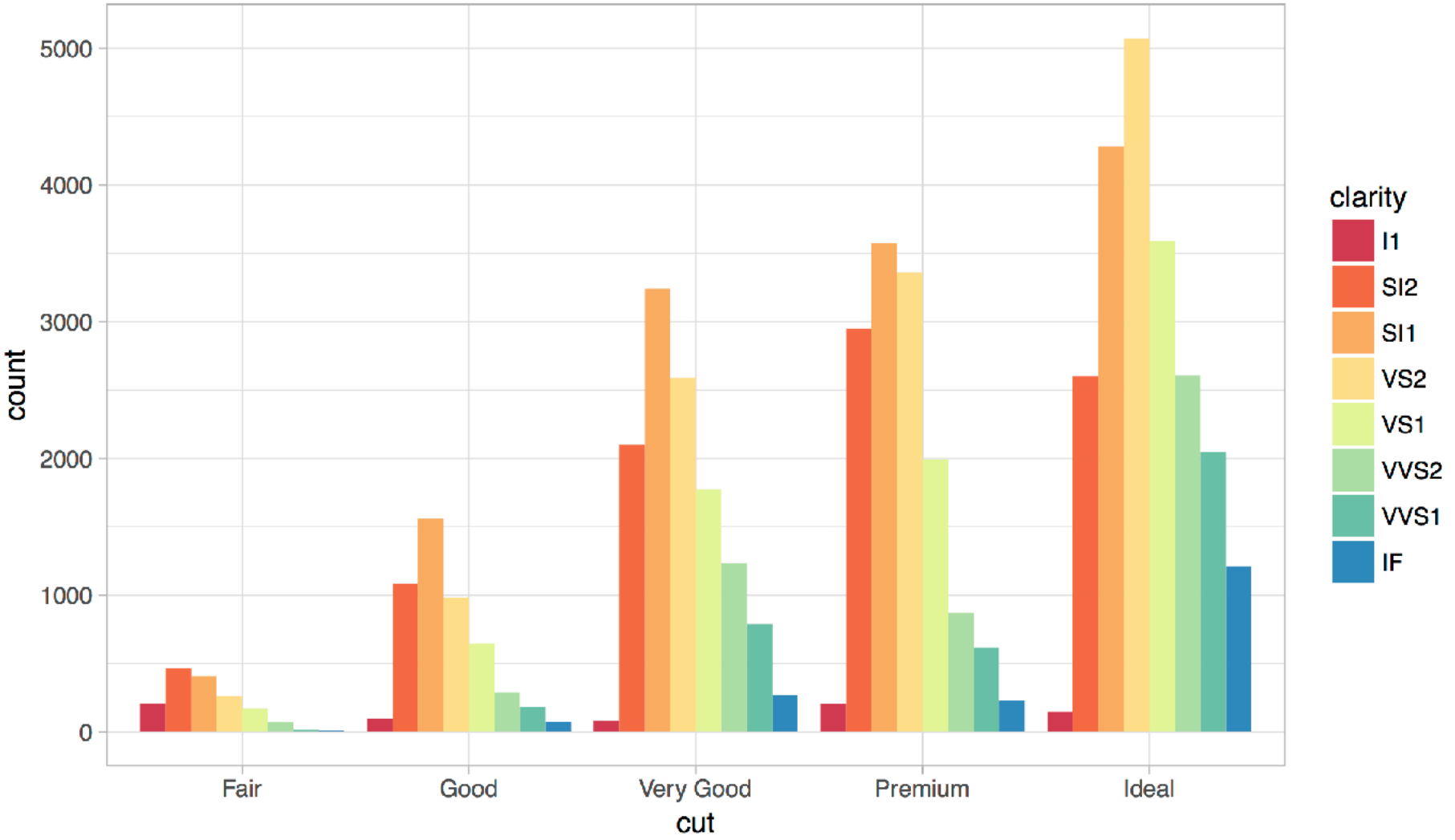
Themes

Visual appearance of non-data elements



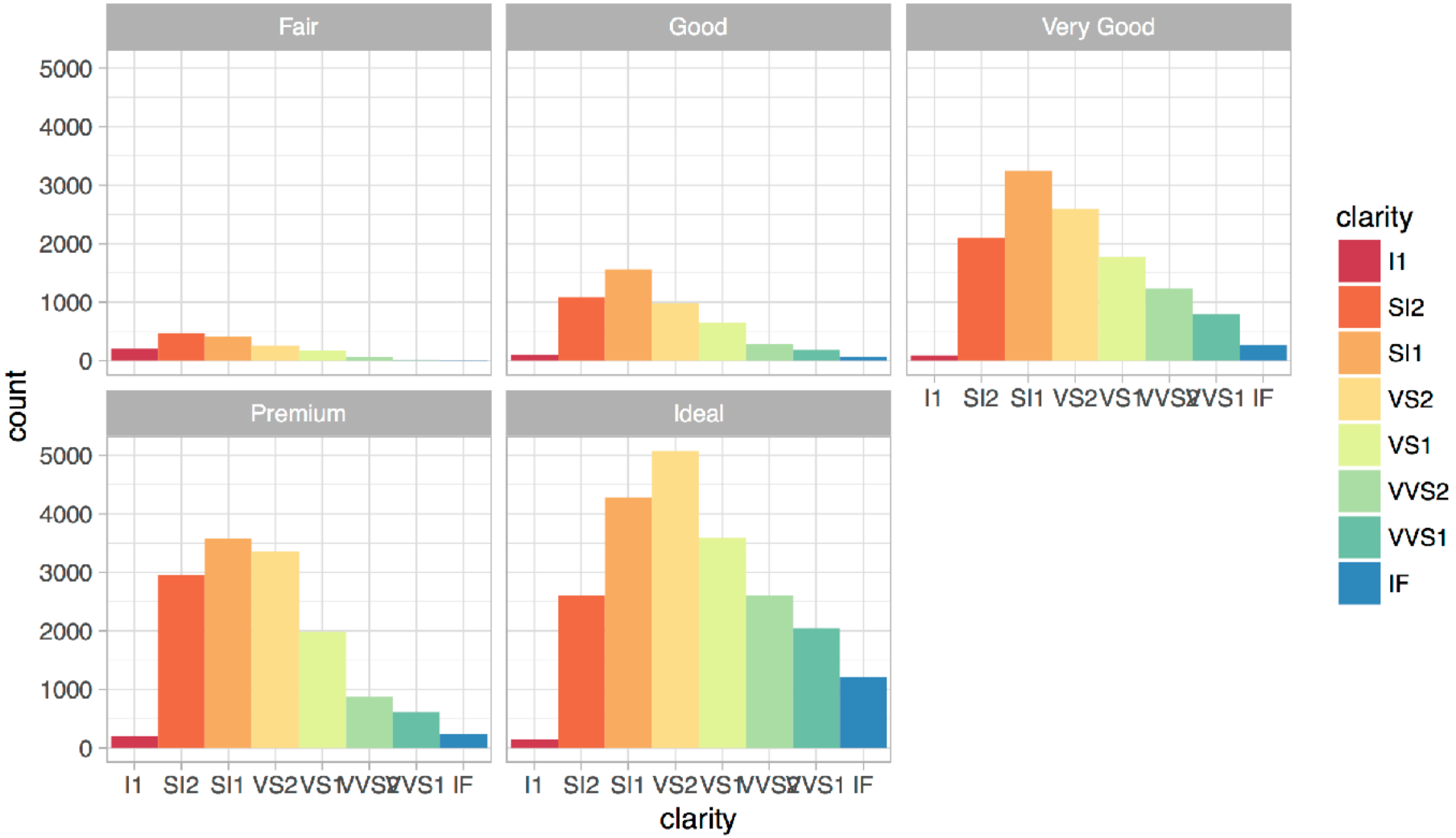
Scales

Customize color scales, other mappings

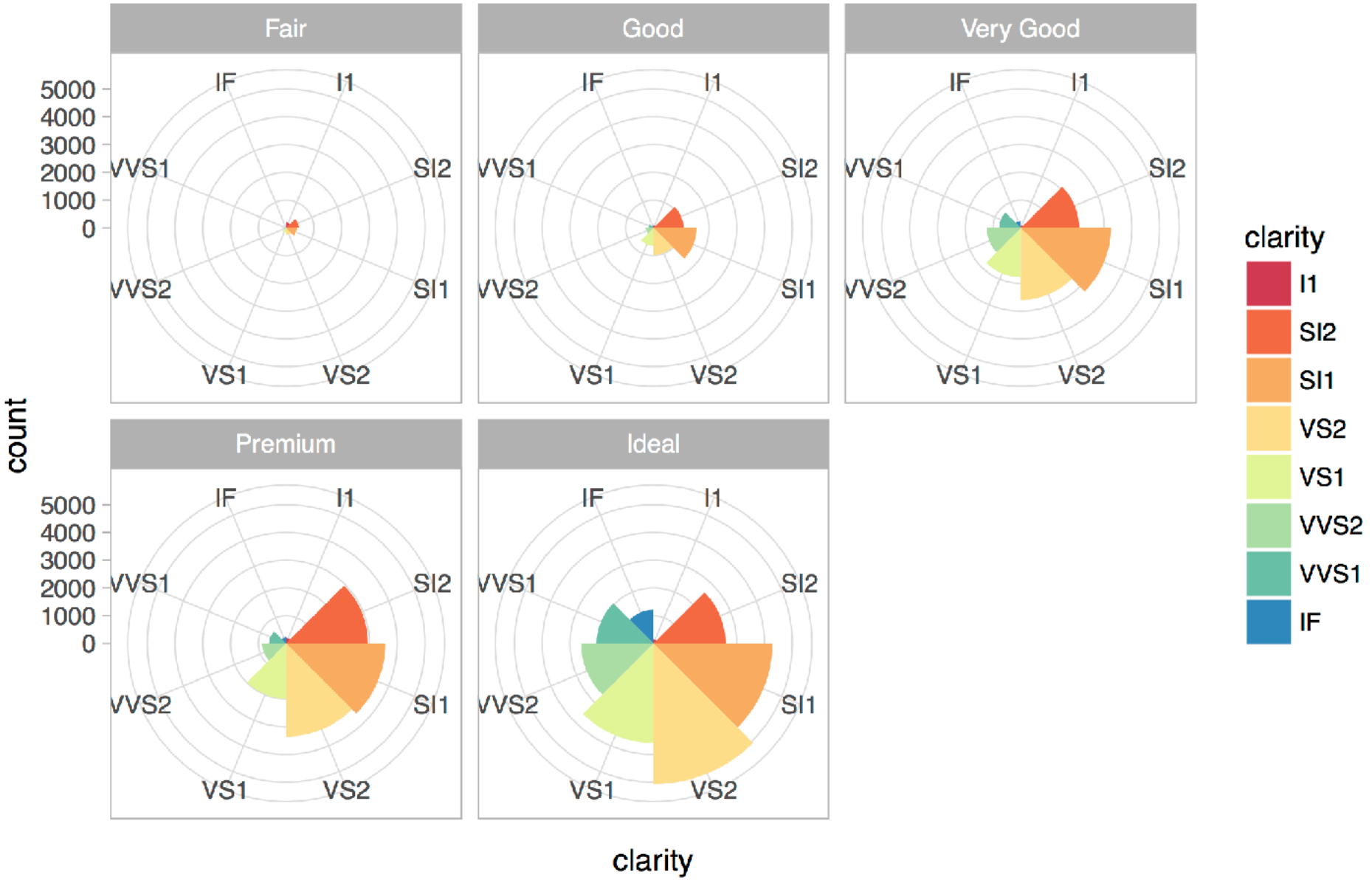


Facets

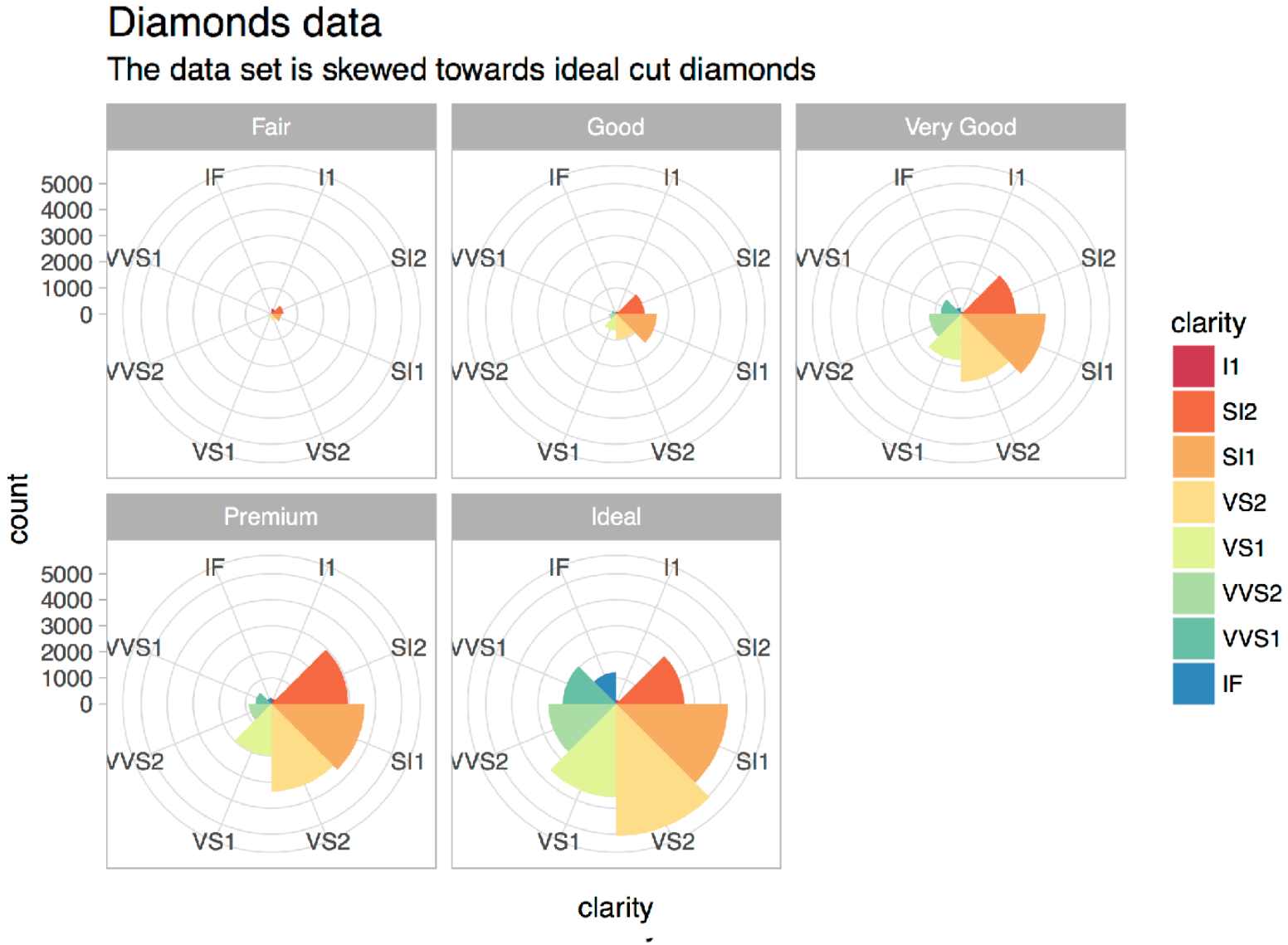
Subplots that display subsets of the data.



Coordinate systems



Titles and captions



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

Required

Optional