

Data Visualization

Session 3
Stephan Kadauke
July 16, 2020

July 16 2020	Session	Instructor
1:00 pm - 1:30 pm	Instructor Introductions, Introduction to technology	Amrom Obstfeld
1:30 pm - 2:15 pm	Introduction to R and RStudio	Joe Rudolf
2:30 pm - 3:15 pm	Reproducible Reporting	Patrick Mathias
3:30 pm - 5:00 pm	Data Visualization	Stephan Kadauke
July 17 2020		
1:00 pm - 2:30 pm	Data Transformation	Amrom Obstfeld
2:45 pm - 4:15 pm	Statistical Analysis	Dan Herman
4:30 pm - 5:00 pm	Advanced Reporting	Patrick Mathias

Goals

1. Appreciate the importance of visualization for understanding data
2. Learn how to use ggplot2 to visualize data

Objectives

1. Create a basic visualization using a simple template
2. Define “aesthetic mapping” and explain how aesthetic mappings relate variables of a data frame to features of graphic markings on a plot
3. Write the code to specify a type of plot and fine tune its appearance using “geom” functions
4. Explain how to add layers to a ggplot object to create complex and highly customized visualizations

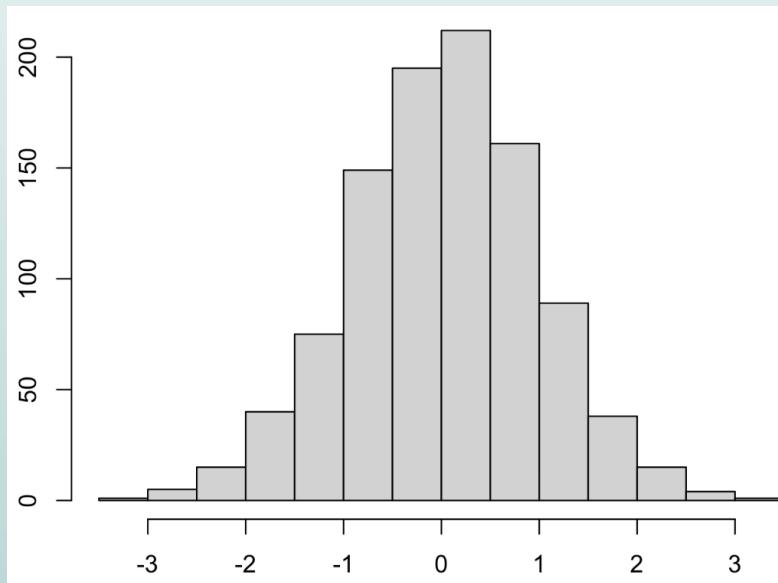
Your Turn 1

Consider the `covid_testing` data frame.

What do you think plot would look like in which:

- the x-axis represents `pan_day` (day of the pandemic), and
- the y-axis represents the number of tests that were performed on that day?

Your Turn 2



What is the name of this kind of plot?
Type the answer into the chat!

Your Turn 3

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

Pay attention to the spelling, capitalization, and parentheses!

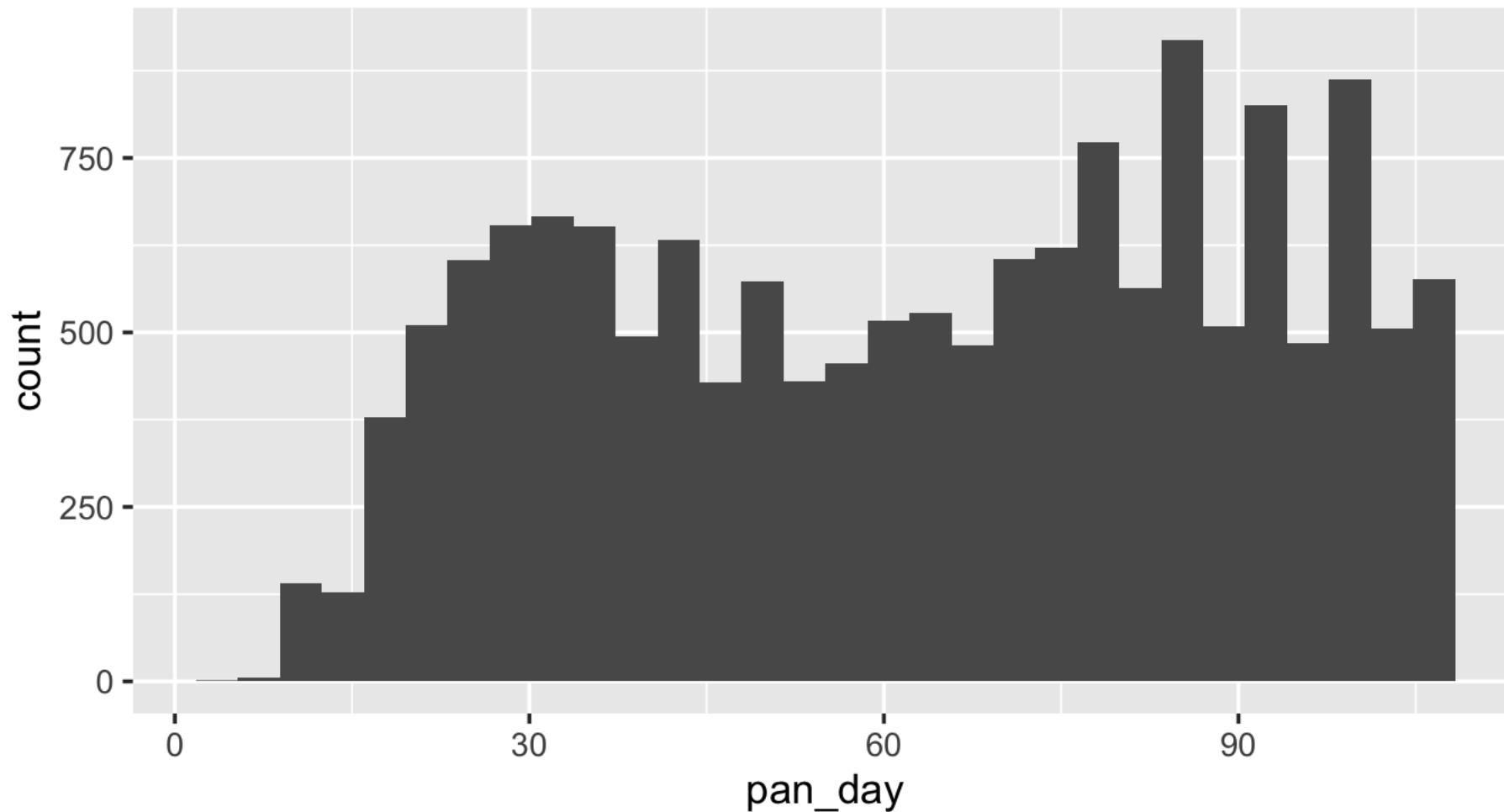
```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day))
```

When you run this code, you will get what looks like an error but is actually just a message.

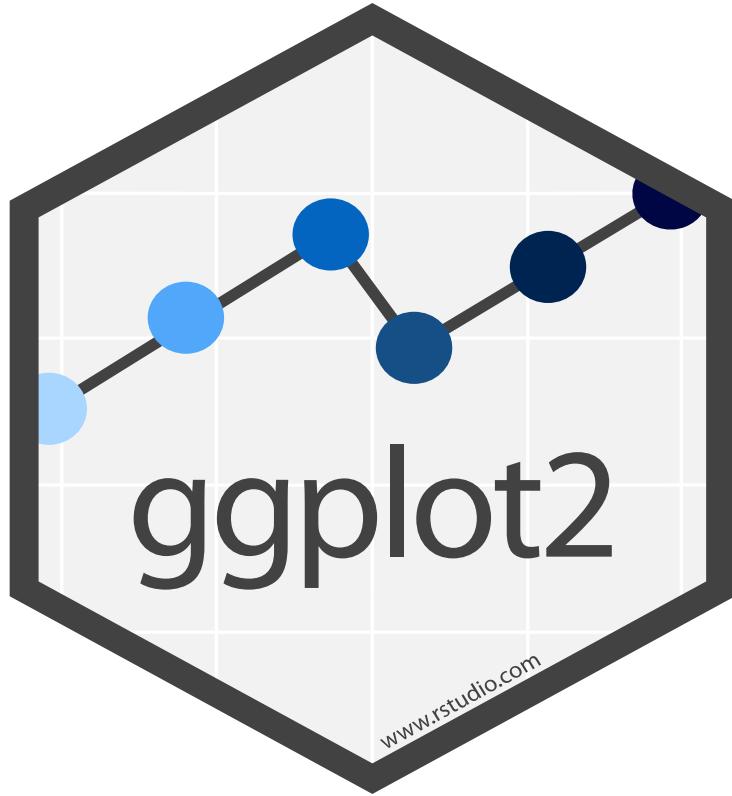
R lets you know that when you ask it to draw a histogram you should tell it how wide each bin should be, because this affects the granularity of the data displayed.

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day))
```



```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day))
```



ggplot()

Always start
with ggplot()

data frame

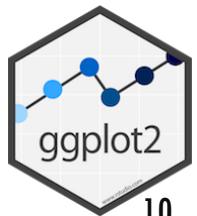
+ sign
before new line

```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day))
```

type of plot

mappings inside
aes() function

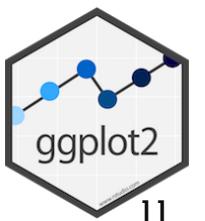
x axis
mapping



To make **any** kind of graph:

1. Choose a “tidy”
data frame

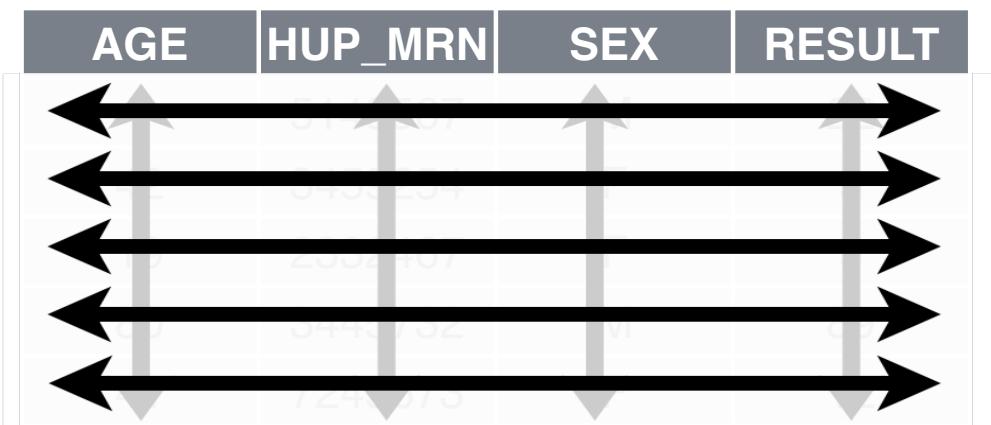
```
ggplot(data = data_frame) +  
  geom_function(mapping = aes(mappings))
```



1. Pick a “Tidy” Data Frame

A data set is **tidy** if:

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



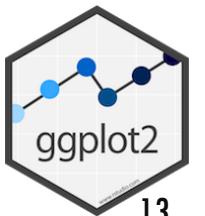
1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

To make **any** kind of graph:

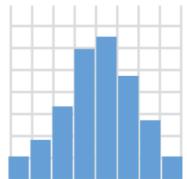
1. Choose a “tidy”
data frame

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

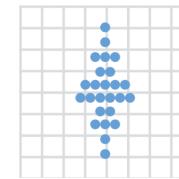
2. Pick a “**geom**”
function



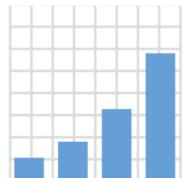
2. Choose a “Geom” Function



`geom_histogram()`



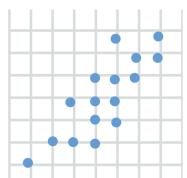
`geom_dotplot()`



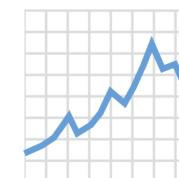
`geom_bar()`



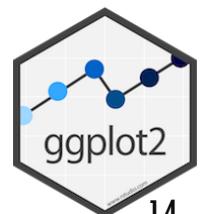
`geom_boxplot()`



`geom_point()`



`geom_line()`



Data Visualization with ggplot2 :: CHEAT SHEET

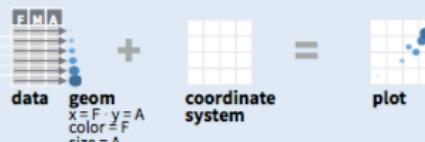


Basics

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>
```

Annotations for the template:

- required**: **GEOM_FUNCTION**
- 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  
ggplot(x = cty, y = hwy, data = mpg, geom = "point")
```

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",
linemetre = 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**

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**

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
```

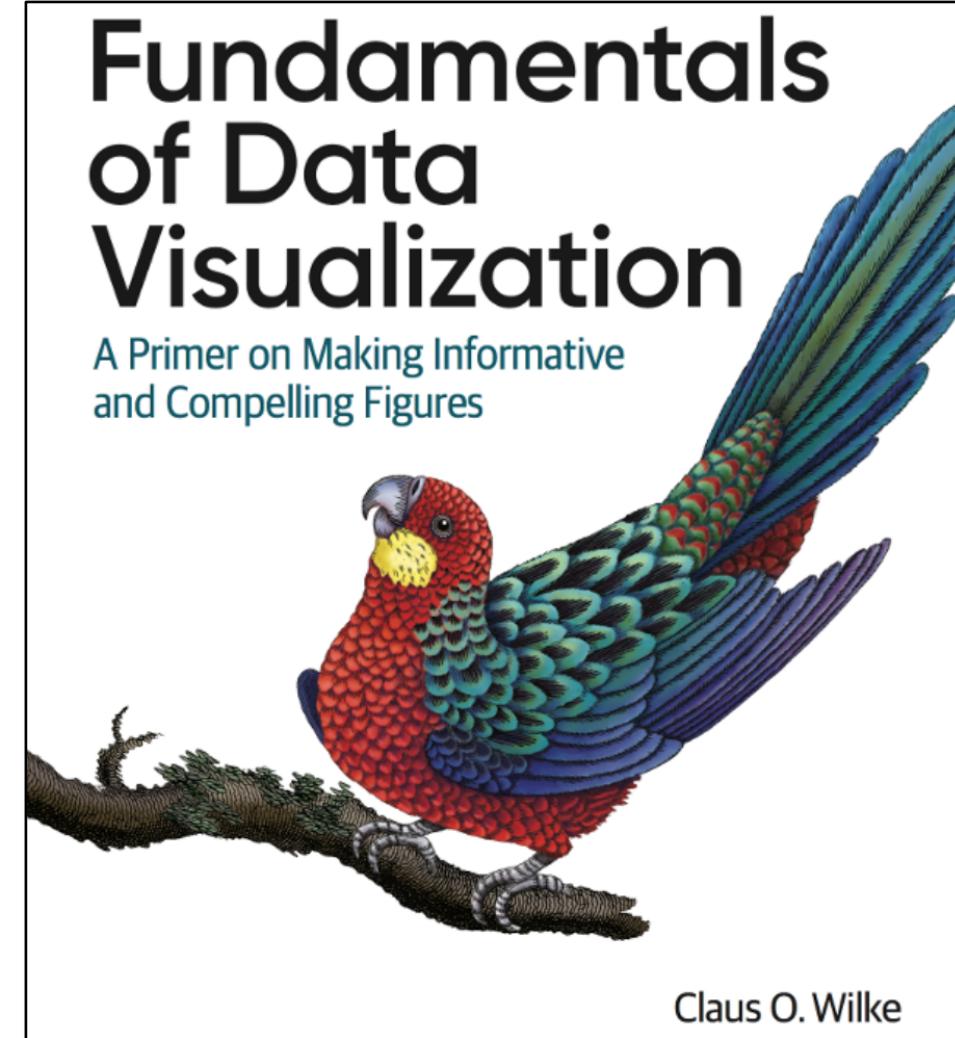
**j + geom_errorbar(), x, y, max, min, alpha, color,
group, linetype, size, width (also
geom_errorbarh())**

**j + geom_linerange()
x, y, min, max, alpha, color, group, linetype, size**

**j + geom_pointrange()
x, y, min, max, alpha, color, fill, group, linetype,
shape, size**

maps

```
data <- data.frame(murder = USArrests$Murder)
```



<https://serialmentor.com/dataviz>

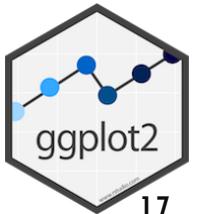
To make **any** kind of graph:

1. Choose a “tidy”
data frame

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

2. Pick a “**geom**”
function

3. Write aesthetic
mappings



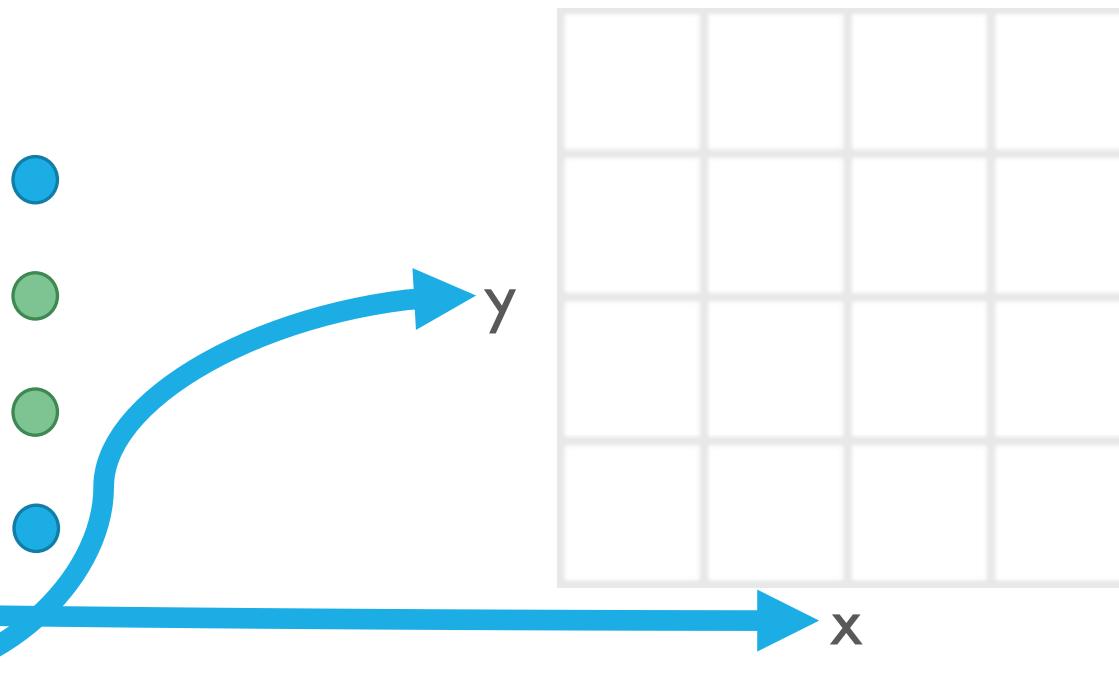
3. Write Aesthetic Mappings

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

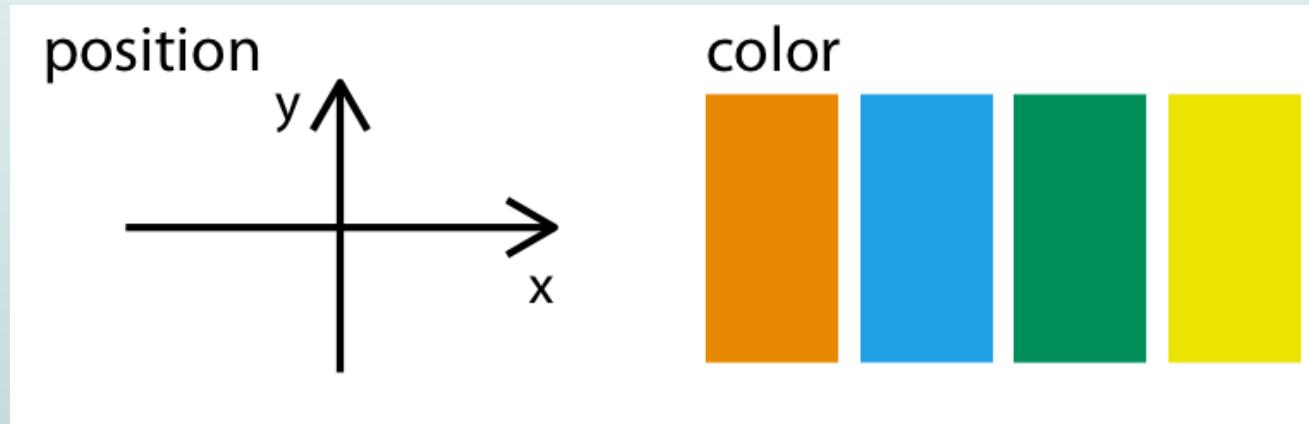
Data frame

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

Graph



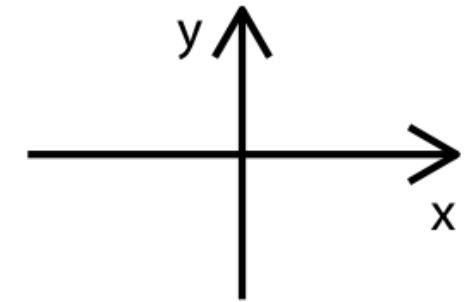
Your Turn 4



In addition to x/y position and color, what other aesthetics can you think of?
Type your answers in the chat!

Aesthetics

position



shape



size



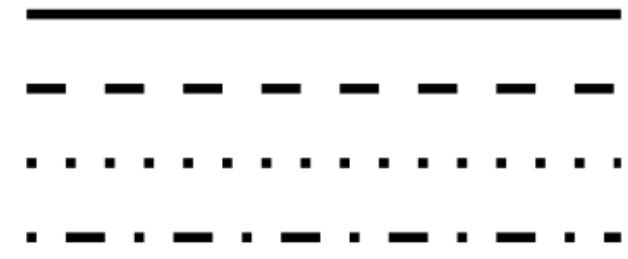
color



line width



line type



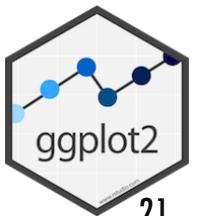
To make **any** kind of graph:

1. Choose a “tidy”
data frame

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

2. Pick a “**geom**”
function

3. Write aesthetic
mappings



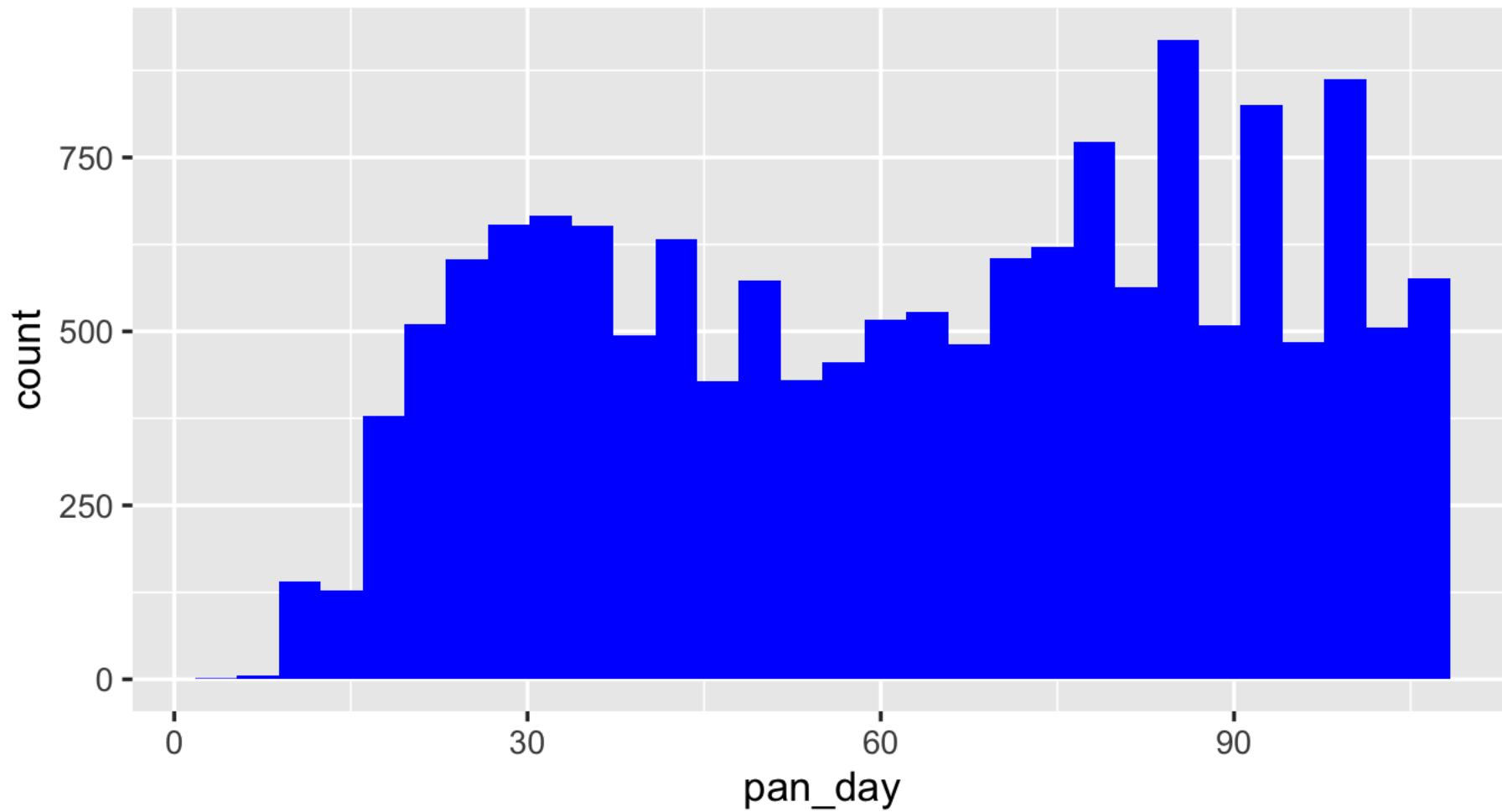
Your Turn 5

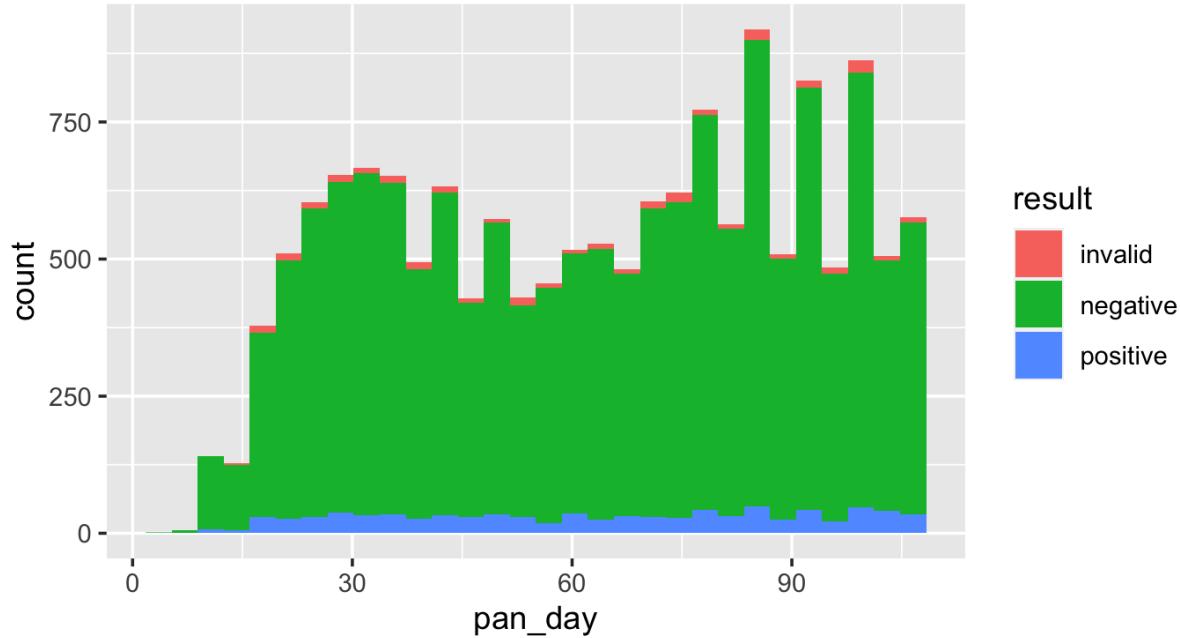
Open 03-Visualize.Rmd. Work through the exercises of the section titled “Your Turn 5.”



Setting vs **Mapping** Aesthetics

How would you make this plot?

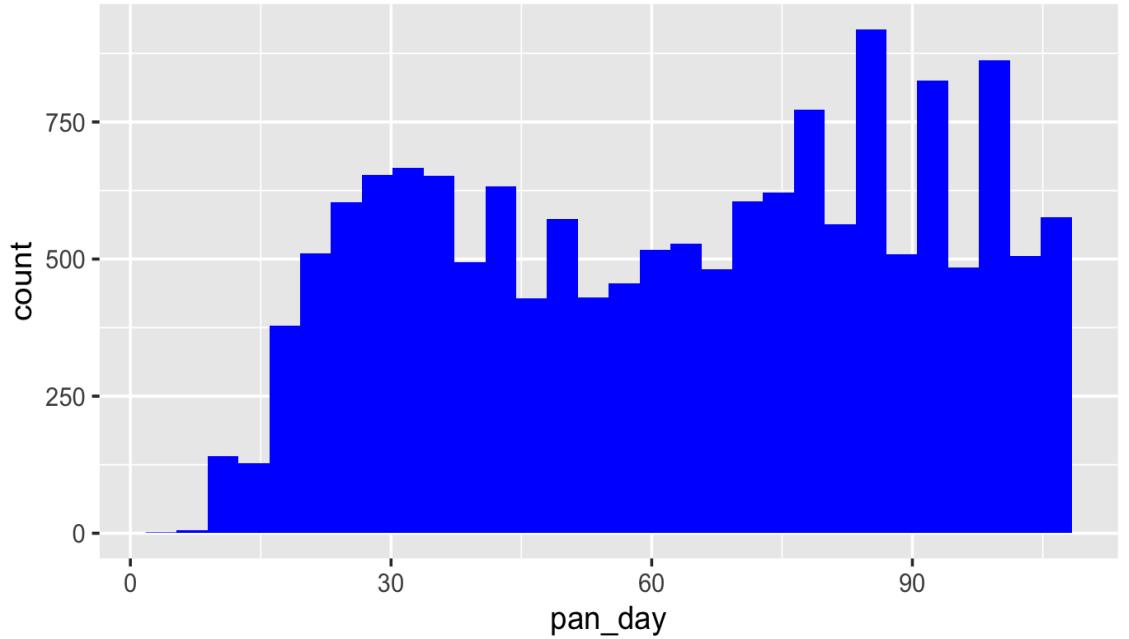




Inside of `aes()`:
map an aesthetic
to a variable

```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day, fill = result))
```

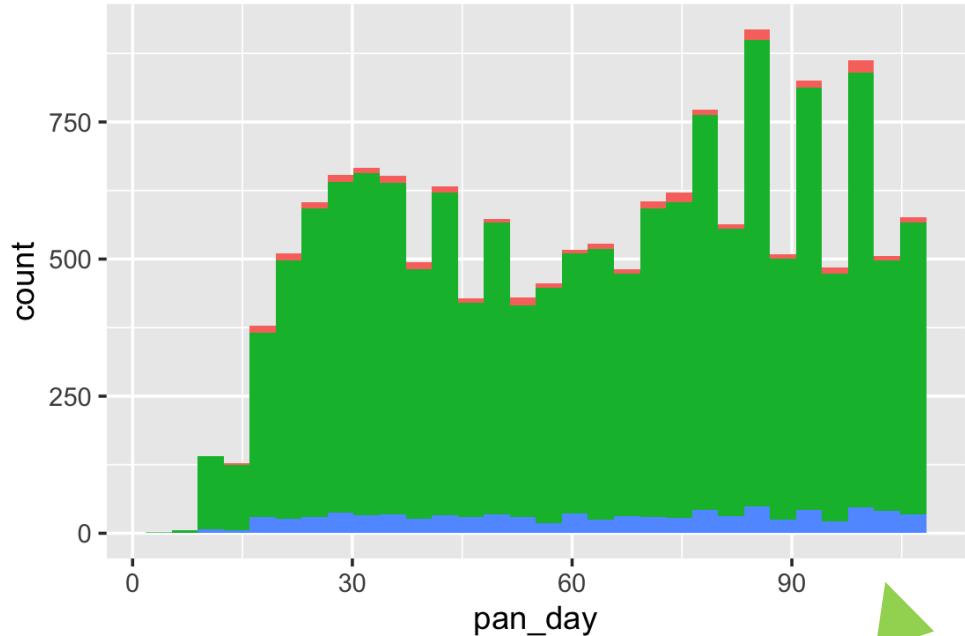




Outside of aes():
set an aesthetic to
a **value**

color name in
“quotes”

```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day), fill = "blue")
```

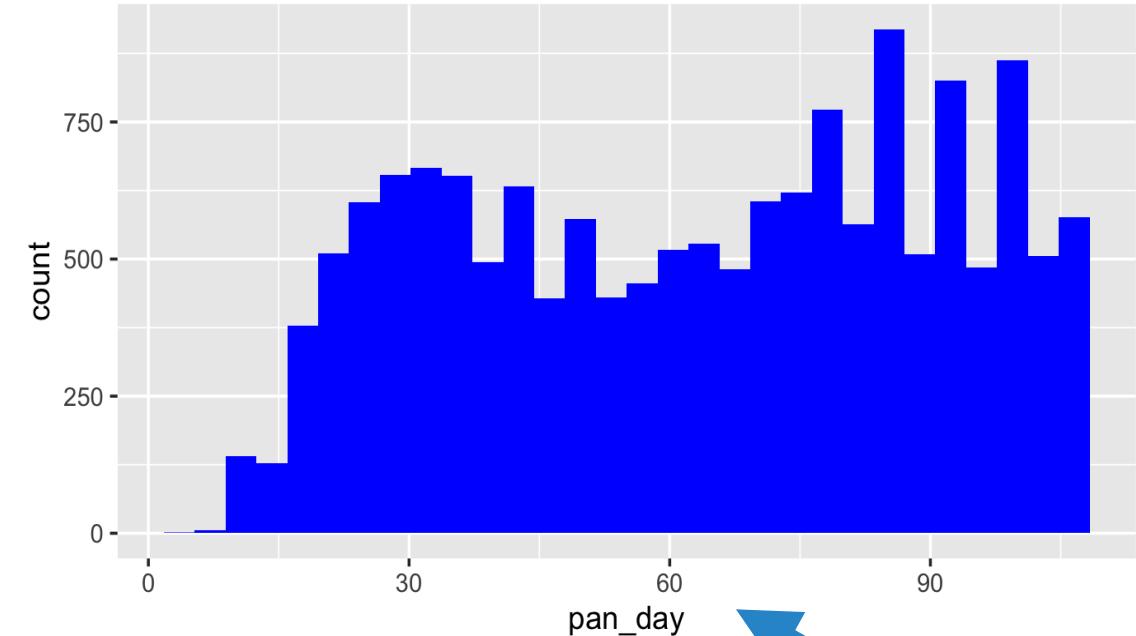


result

invalid

negative

positive

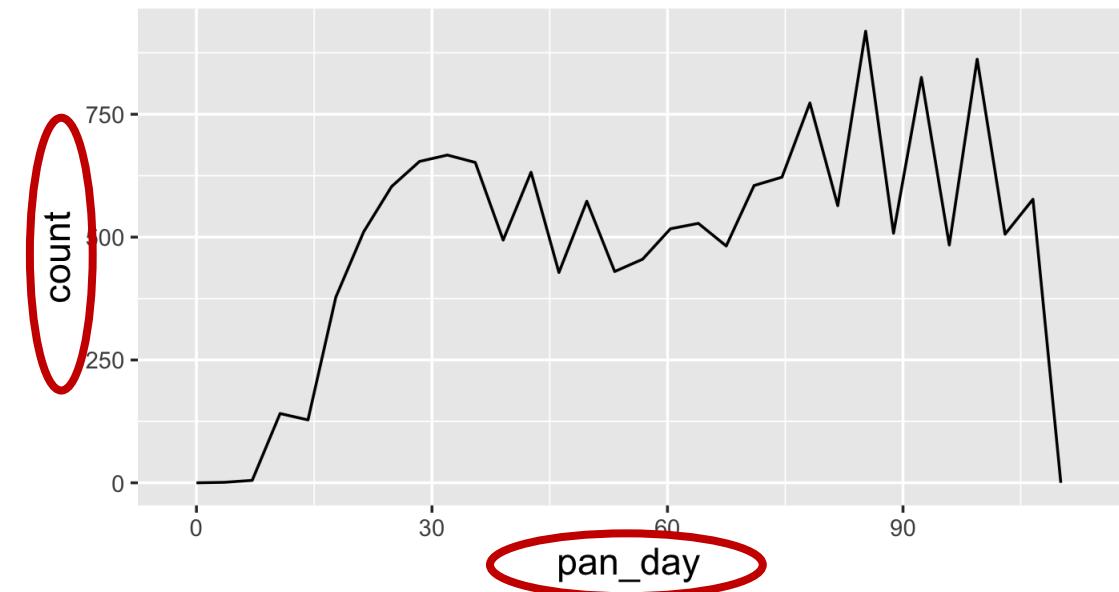
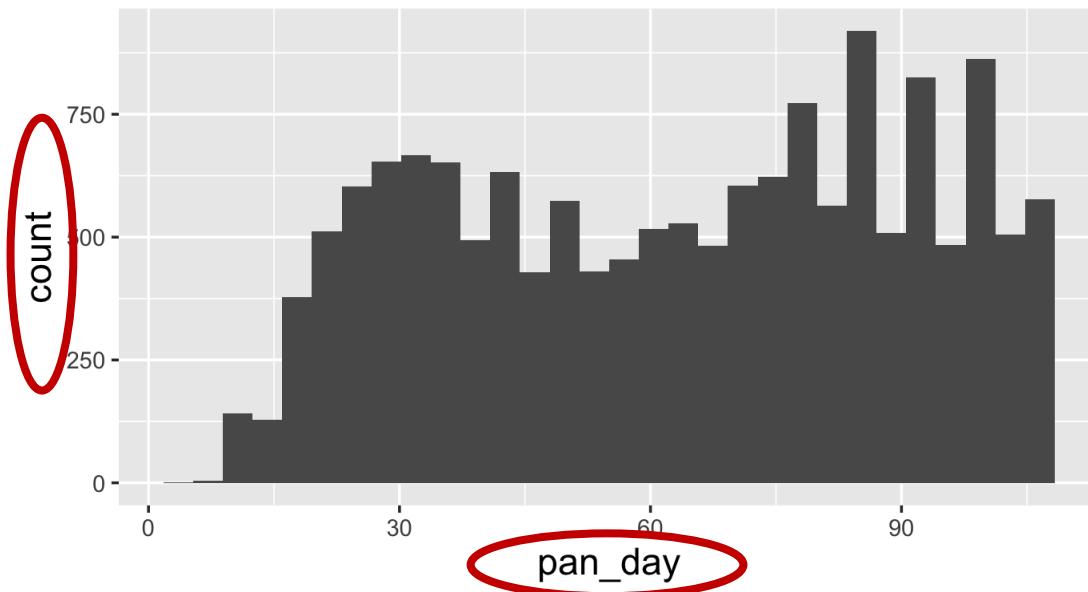


```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day, fill = result))
```

```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day), fill = "blue")
```

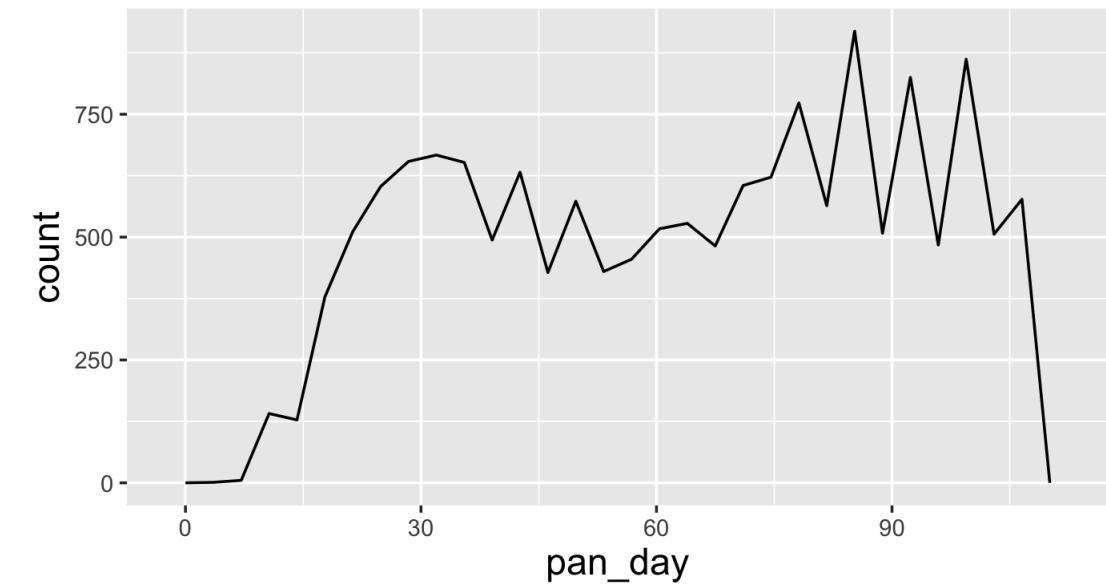
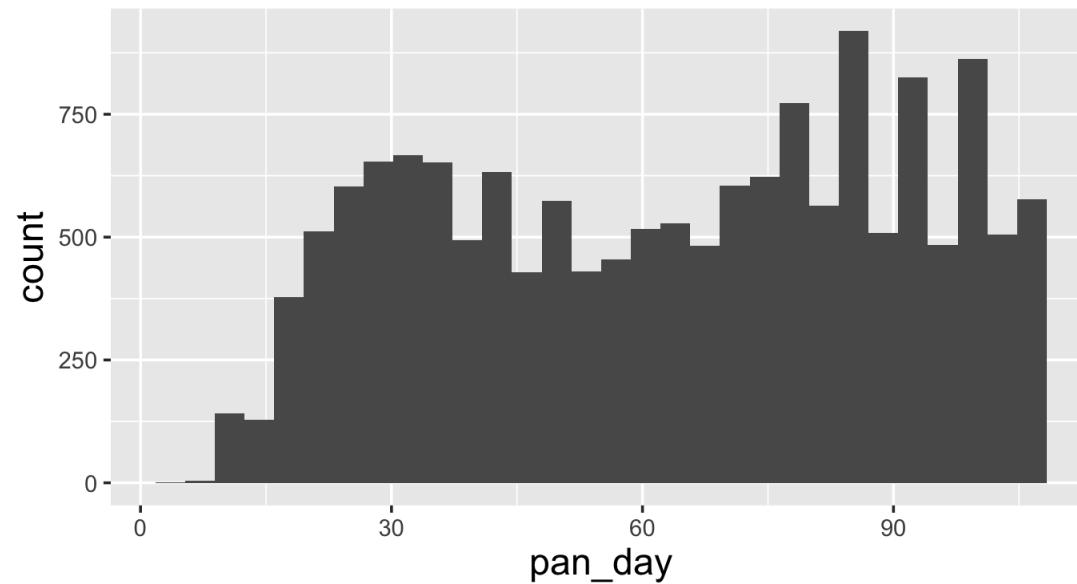
Geom Functions

How are these plots similar?



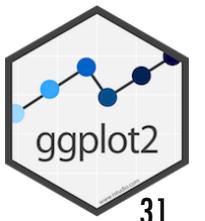
Same: x axis, y axis, data

How are these plots different?



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

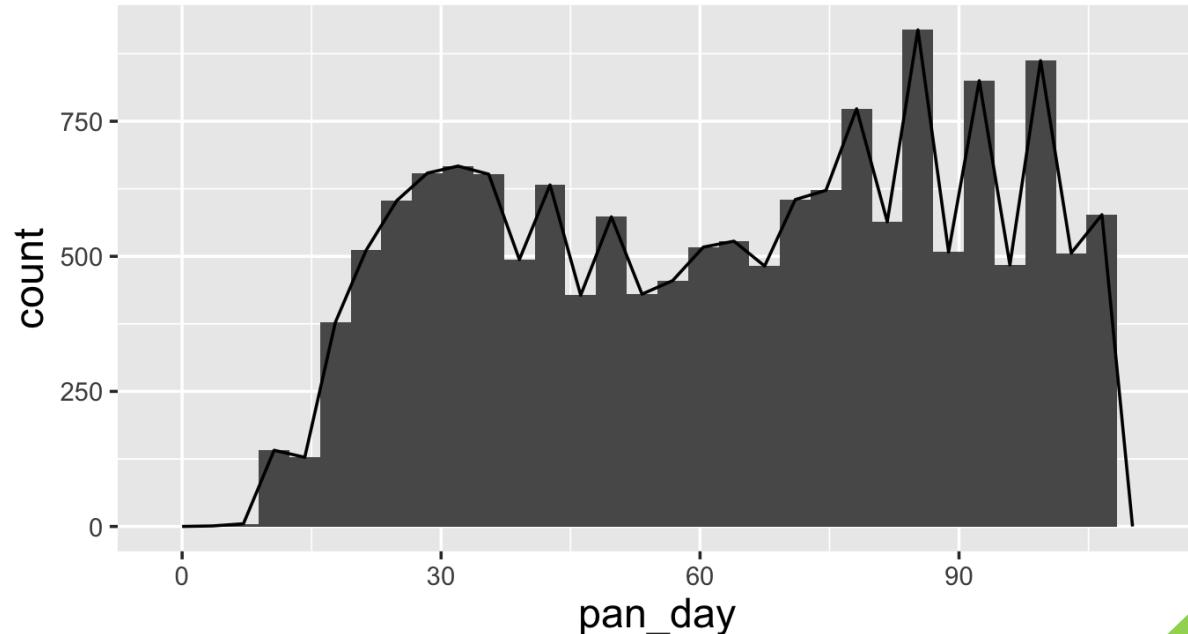
```
ggplot(data = data_frame) +  
  geom_function(mapping = aes(mappings))
```



Your Turn 6

Open 03-Visualize.Rmd. Work through the exercises of the section titled “Your Turn 6.”

Global vs Local Settings



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

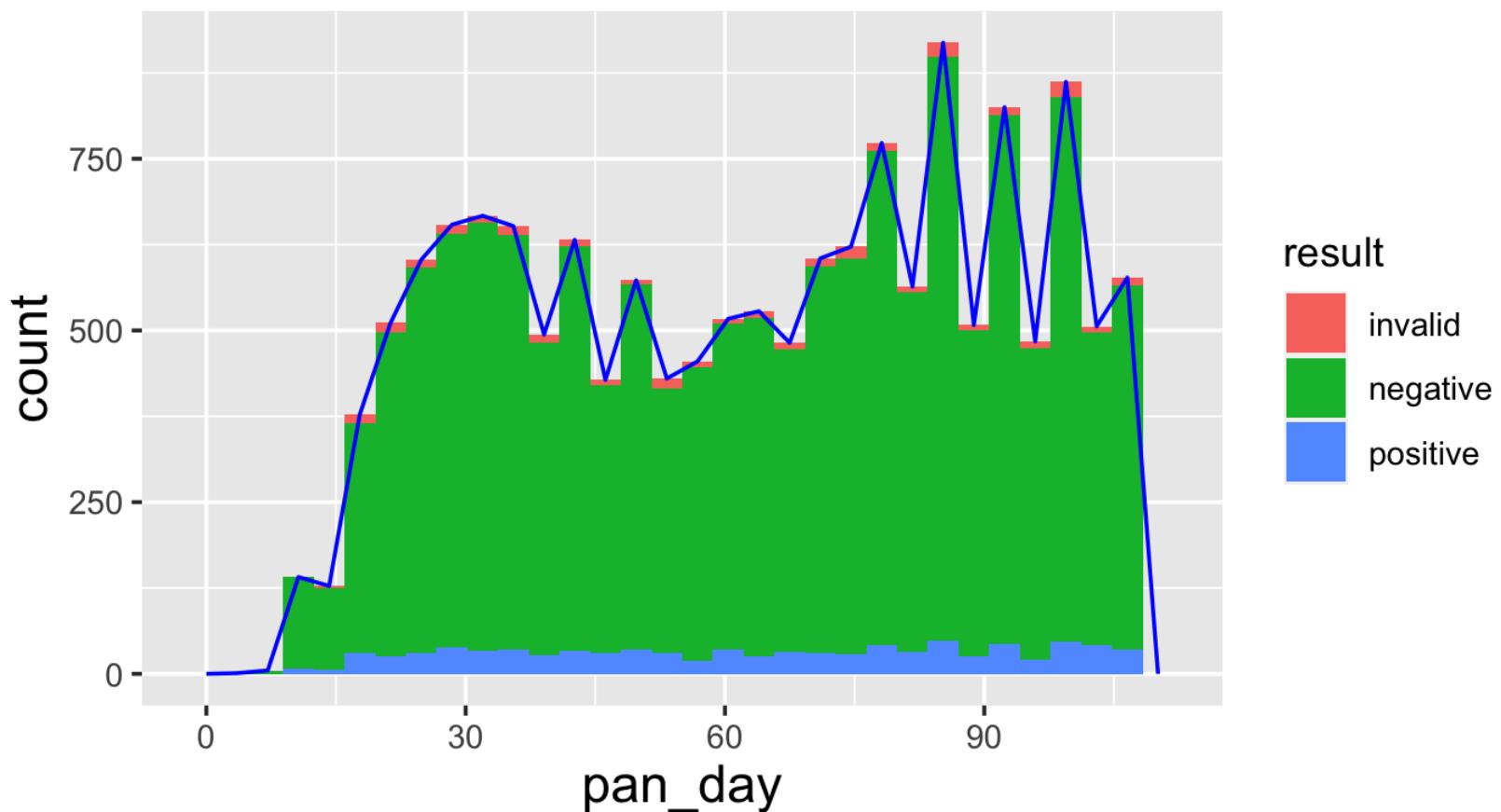
Inside of `ggplot()`:
apply **globally** to
every layer

```
ggplot(data = covid_testing) +  
  geom_histogram(mapping = aes(x = pan_day)) +  
  geom_freqpoly(mapping = aes(x = pan_day))
```

```
ggplot(data = covid_testing, mapping = aes(x = pan_day)) +  
  geom_histogram() +  
  geom_freqpoly()
```

empty ()

How would you make this plot?

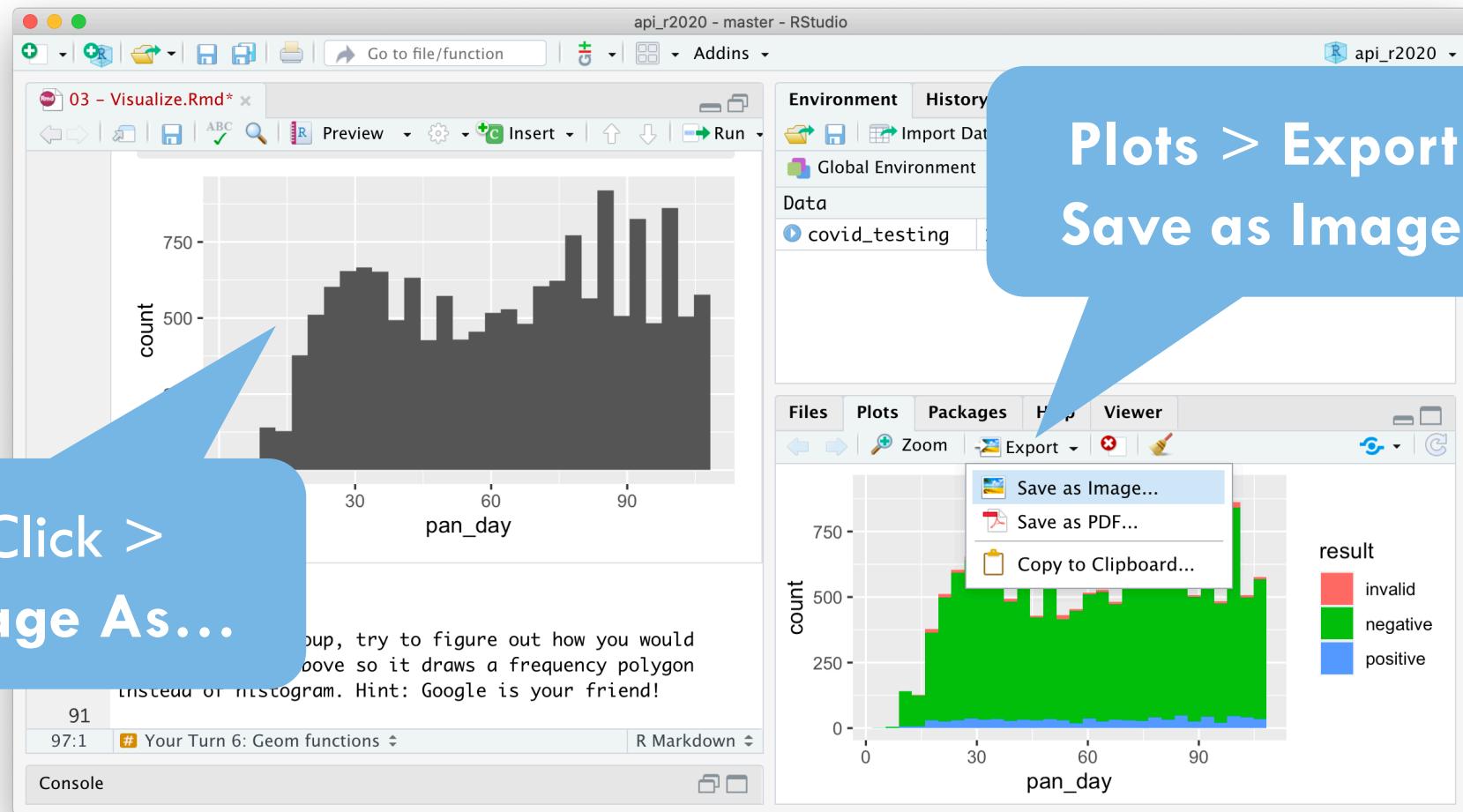


```
ggplot(data = covid_testing, mapping = aes(x = pan_day)) +  
  geom_histogram(mapping = aes(fill = result)) +  
  geom_freqpoly(color = "blue")
```



What Else?

Manually saving plots

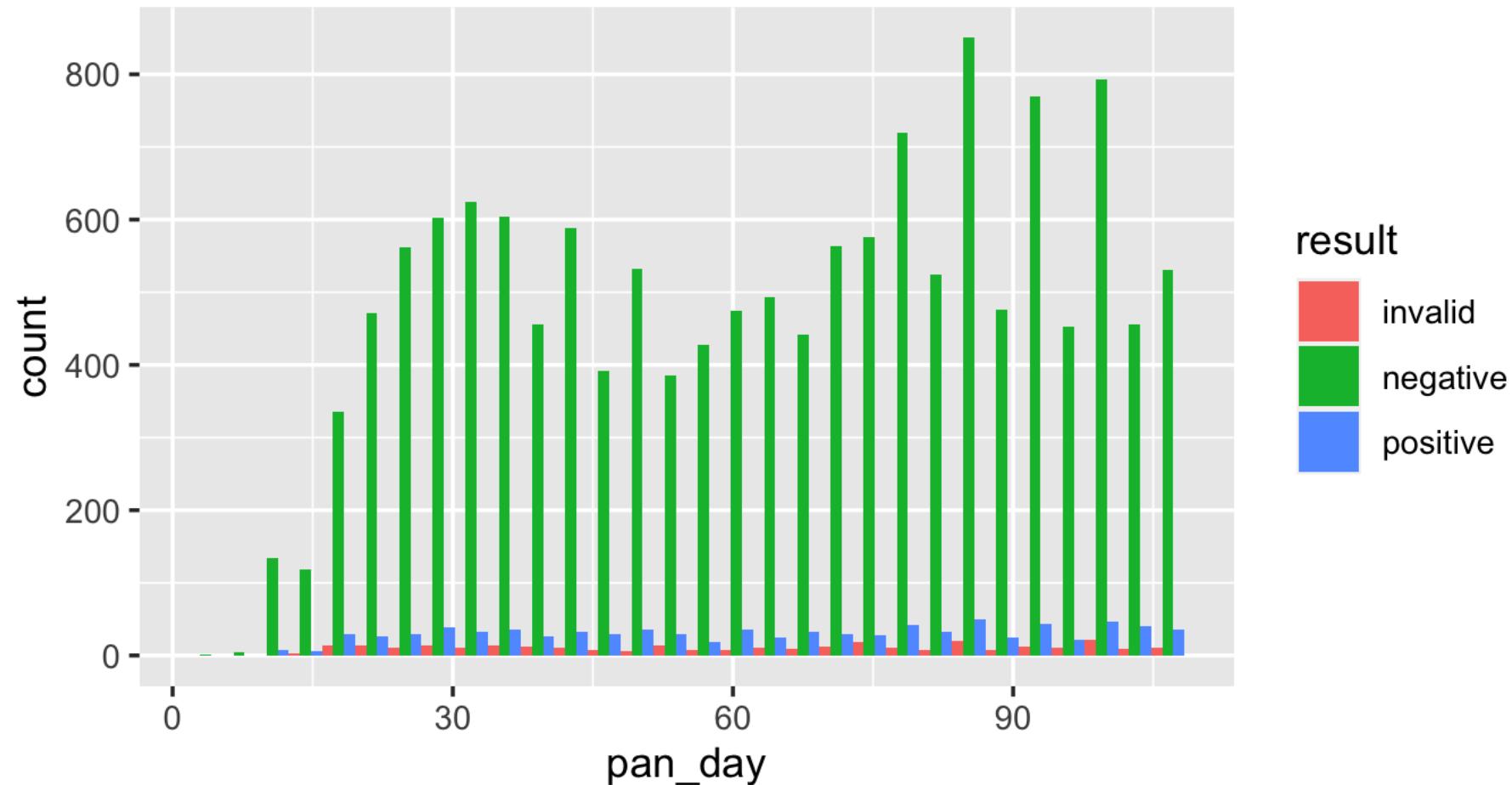


Plots > Export >
Save as Image...

Right Click >
Save Image As...

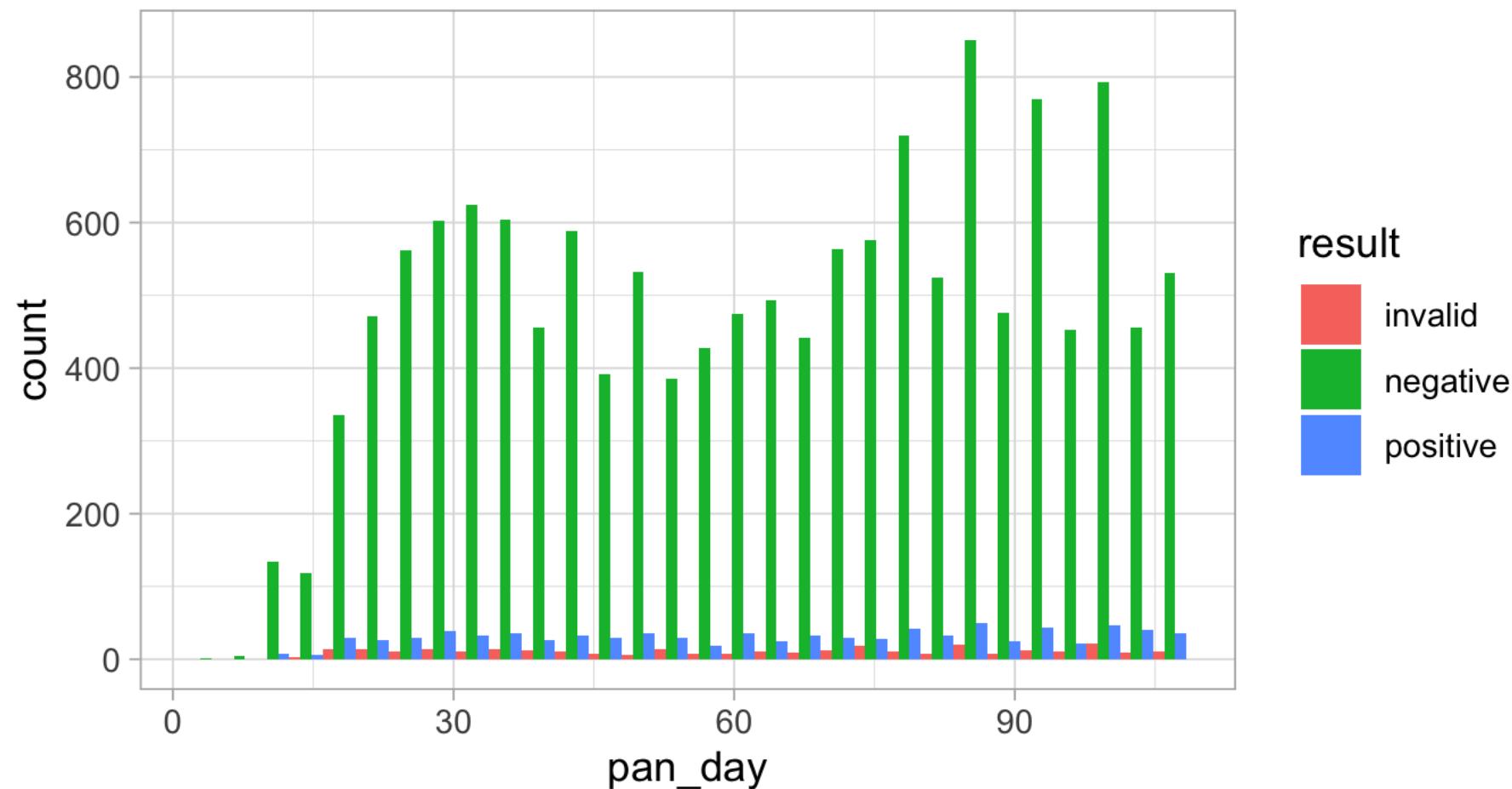
Position adjustments

How overlapping objects are arranged



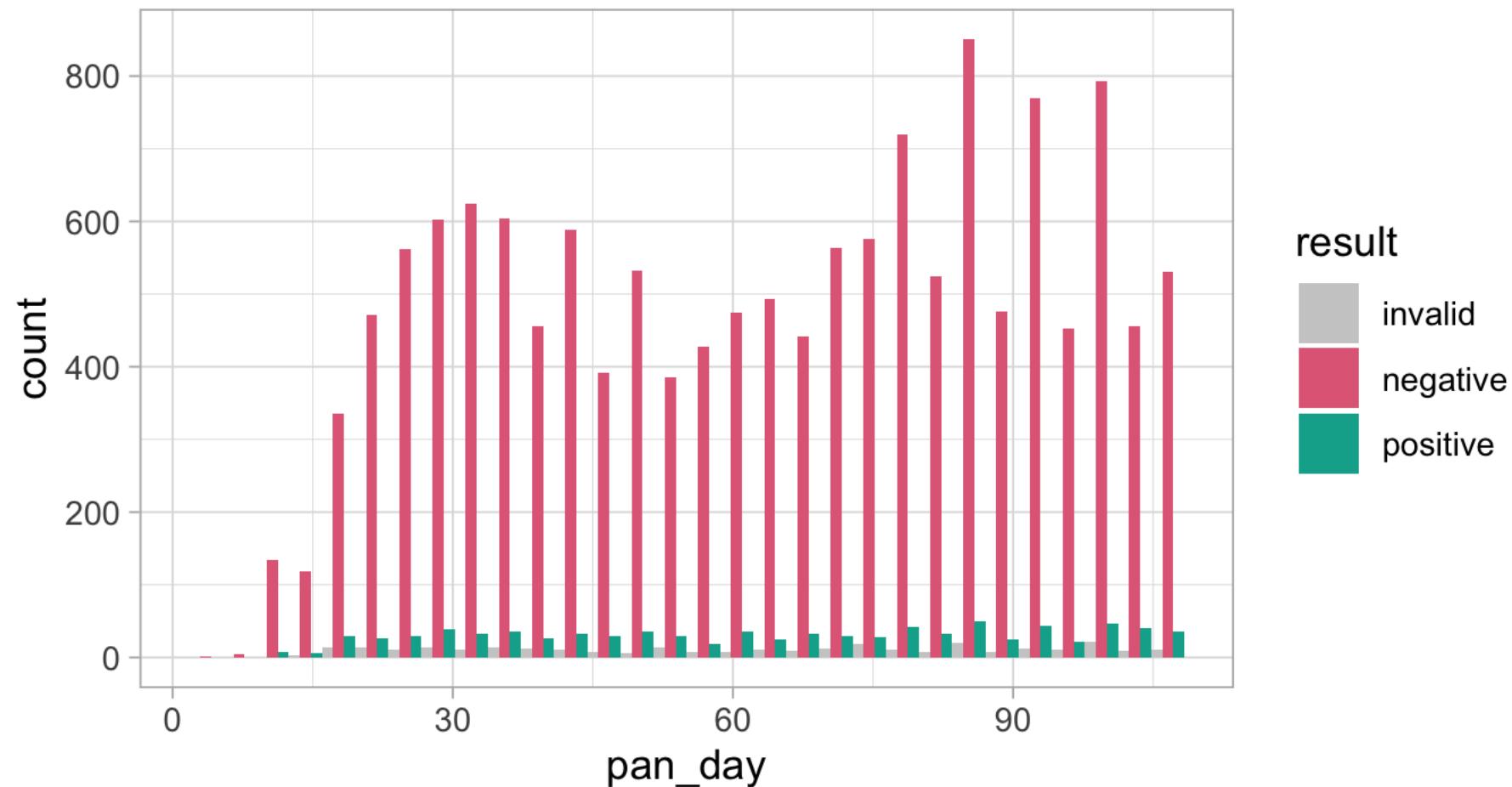
Themes

Visual appearance of non-data elements



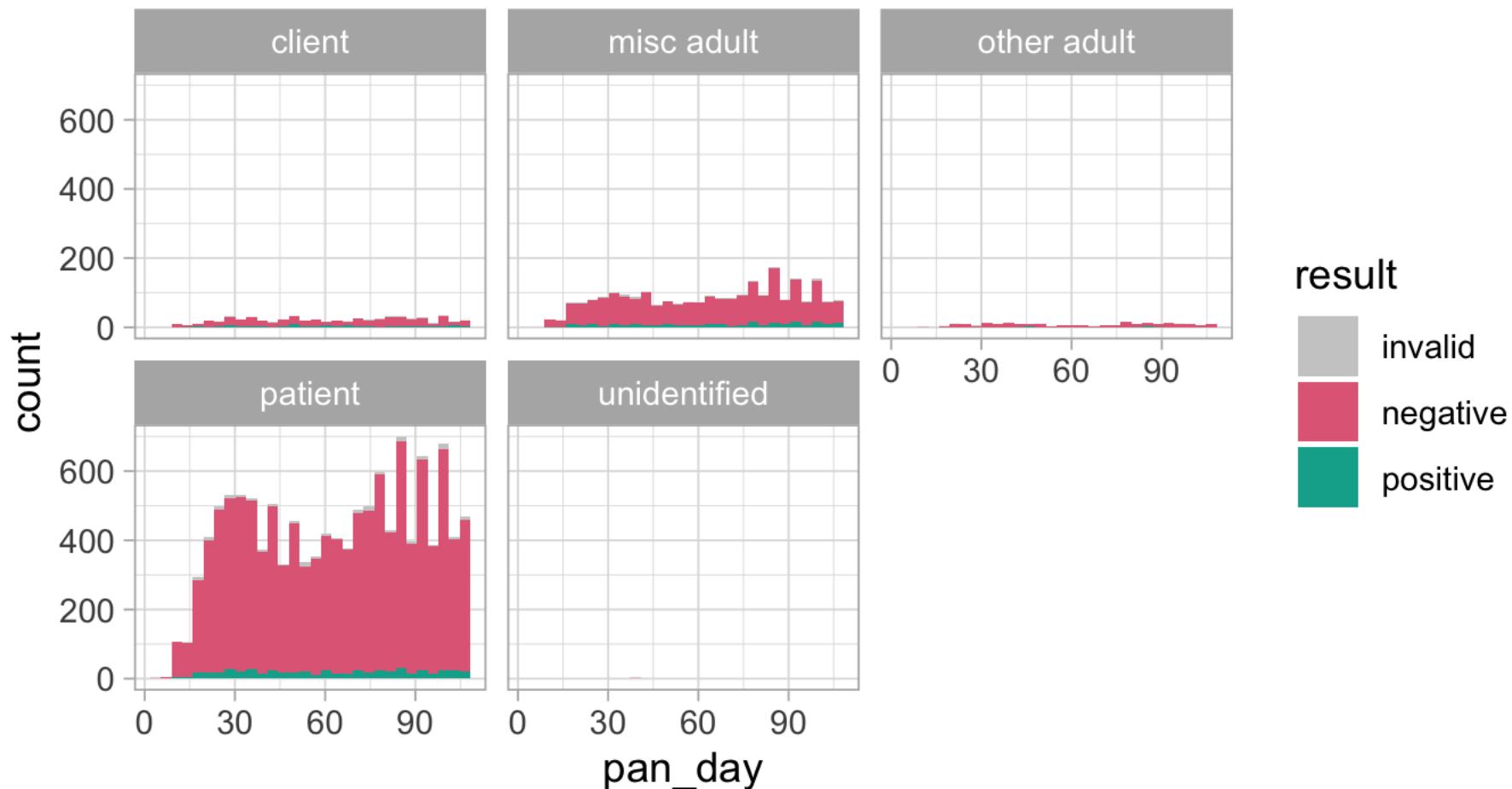
Scales

Customize color scales and other mappings

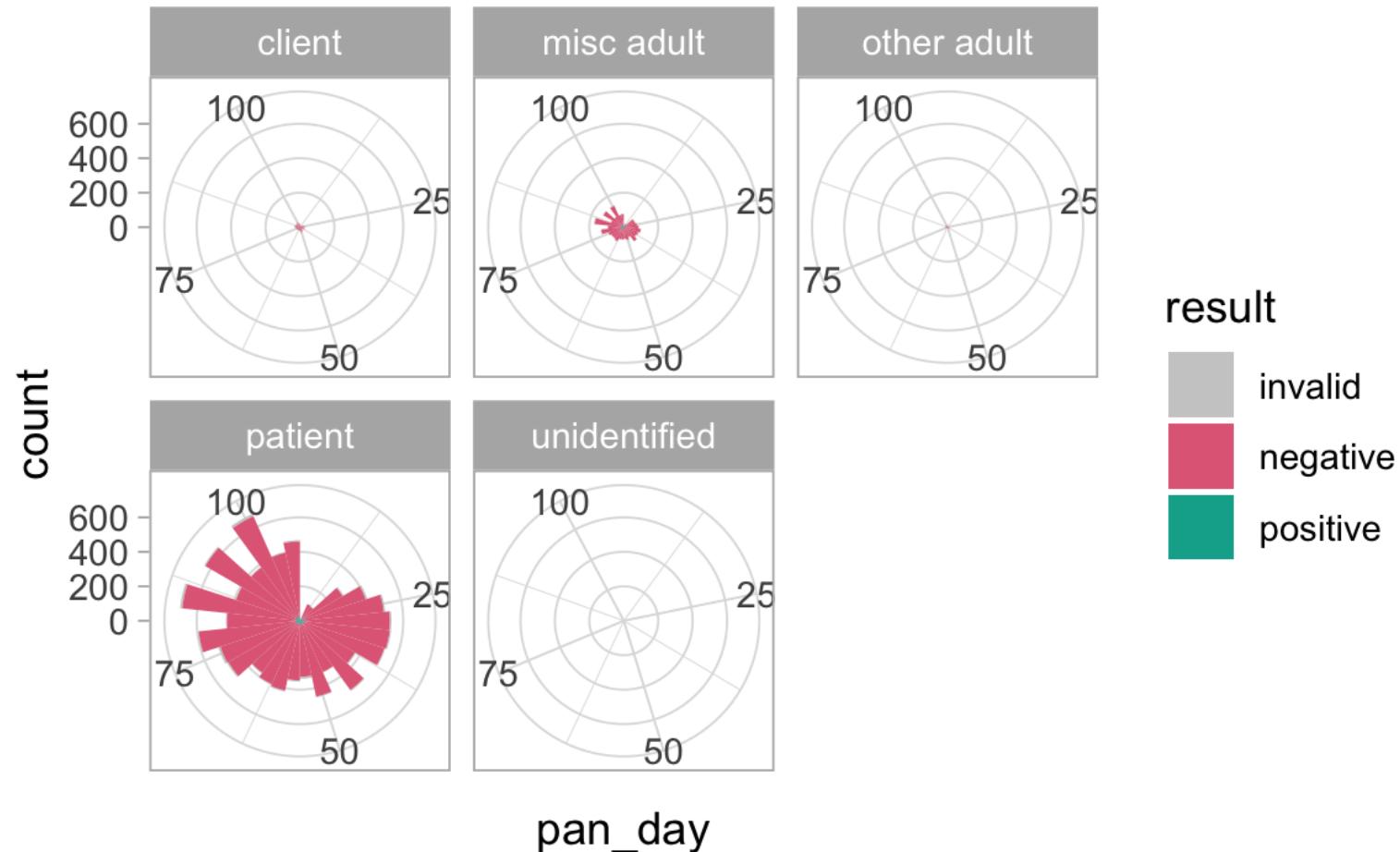


Facets

Subplots that display subsets of the data



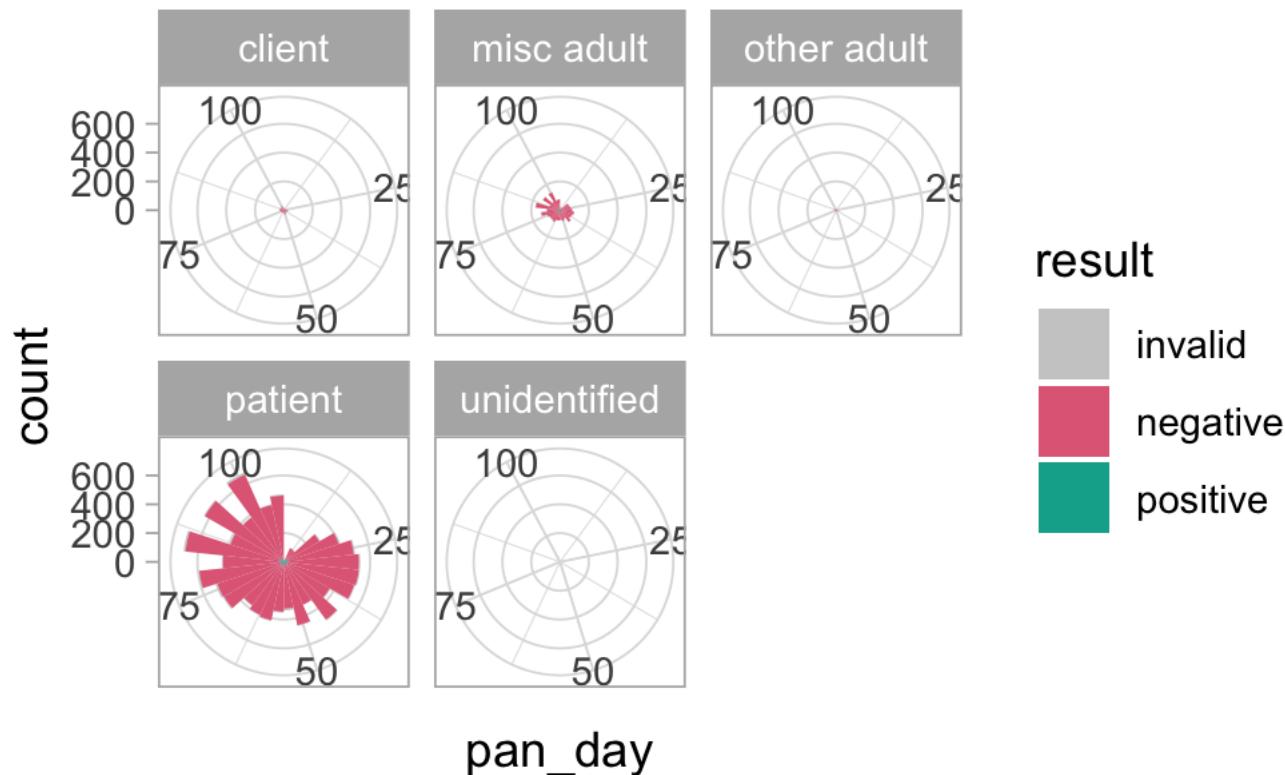
Coordinate systems



Titles and captions

COVID19 test volume

Displayed in polar coordinates, mostly to show off ggplot2



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



Required

Optional

Goals

1. Appreciate the importance of visualization for understanding data
2. Learn how to use ggplot2 to visualize data

Objectives

1. Create a basic visualization using a simple template
2. Define “aesthetic mapping” and explain how aesthetic mappings relate variables of a data frame to features of graphic markings on a plot
3. Write the code to specify a type of plot and fine tune its appearance using “geom” functions
4. Explain how to add layers to a ggplot object to create complex and highly customized visualizations