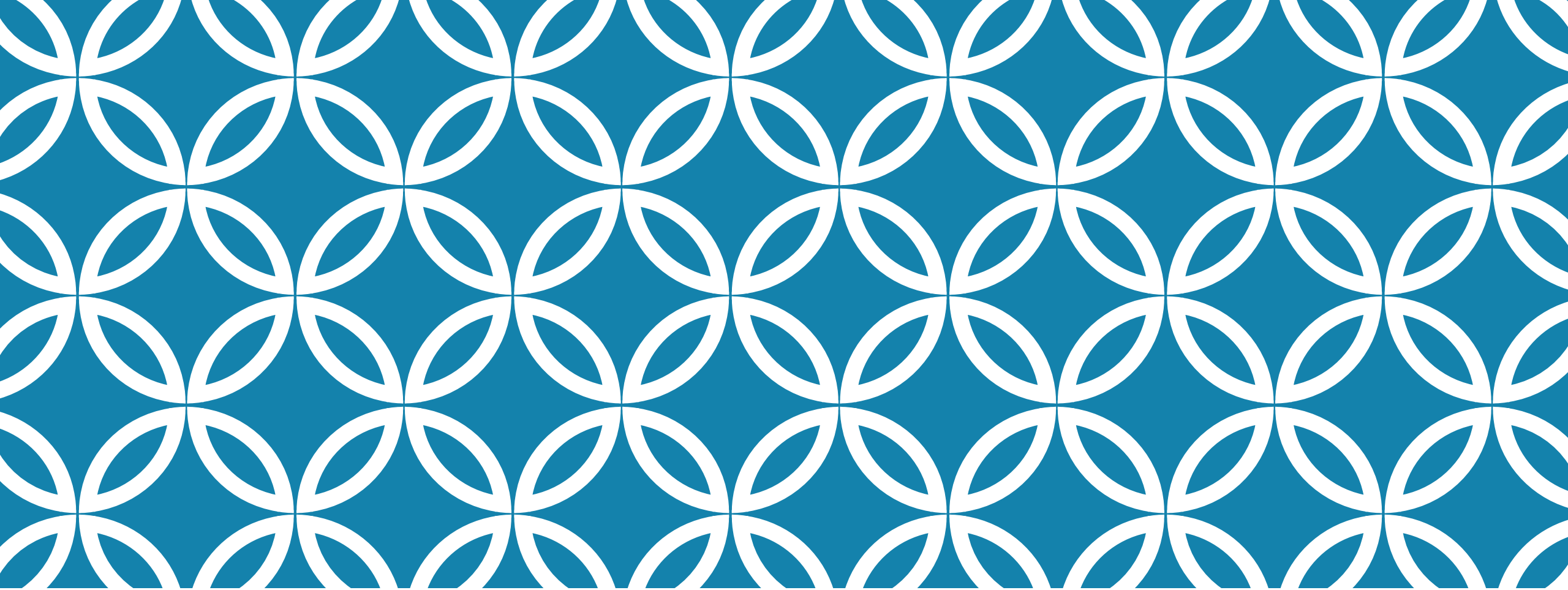


Data Visualization

Session 3

Stephan Kadauke

Session	Instructor
Instructor Introductions, Introduction to technology	Amrom Obstfeld
Introduction to R and RStudio	Joe Rudolf
Reproducible Reporting	Joe Rudolf
Data Visualization	Stephan Kadauke
Data Transformation	Amrom Obstfeld
Statistical Analysis	Dan Herman
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

covid_testing

covid_testing

Filter

	mrn	first_name	last_name	gender	pan_day	test_id	clinic_name	result
1	5001412	jhezane	westerling	female	4	covid	inpatient ward a	negative
2	5000533	penny	targaryen	female	7	covid	clinical lab	negative
3	5009134	grunt	rivers	male	7	covid	clinical lab	negative
4	5008518	melisandre	swyft	female	8	covid	clinical lab	negative
5	5008967	rolley	karstark	male	8	covid	emergency dept	negative
6	5011048	megga	karstark	female	8	covid	oncology day hosp	negative
7	5000663	ithoke	targaryen	male	9	covid	clinical lab	negative
8	5002158	ravella	frey	female	9	covid	emergency dept	negative
9	5003794	styr	tyrell	male	9	covid	clinical lab	negative
10	5004706	wynafryd	seaworth	male	9	covid	clinical lab	negative
11	5008115	patrek	frey	male	9	covid	clinical lab	negative
12	5009309	maege	sand	female	9	covid	medical center	negative
13	5008943	myria	rivers	female	9	covid	picu	negative

Showing 1 to 14 of 15,524 entries, 17 total columns

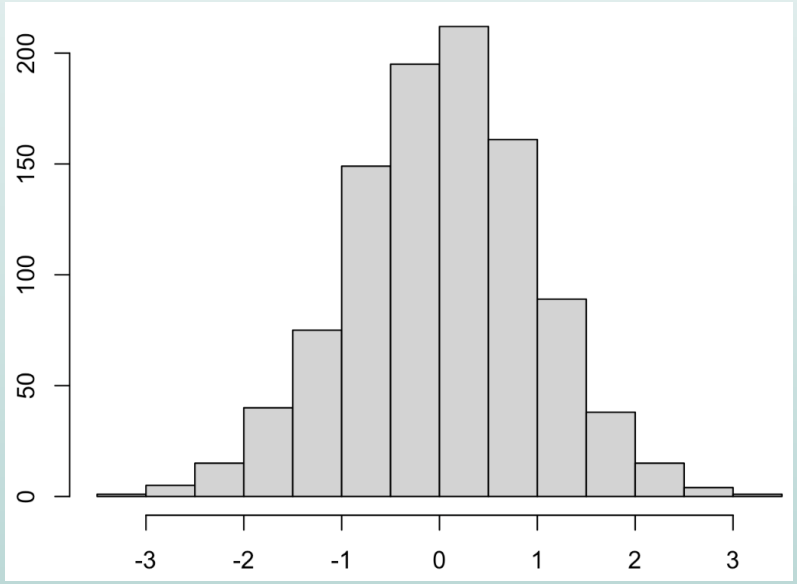
Your Turn 1

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!

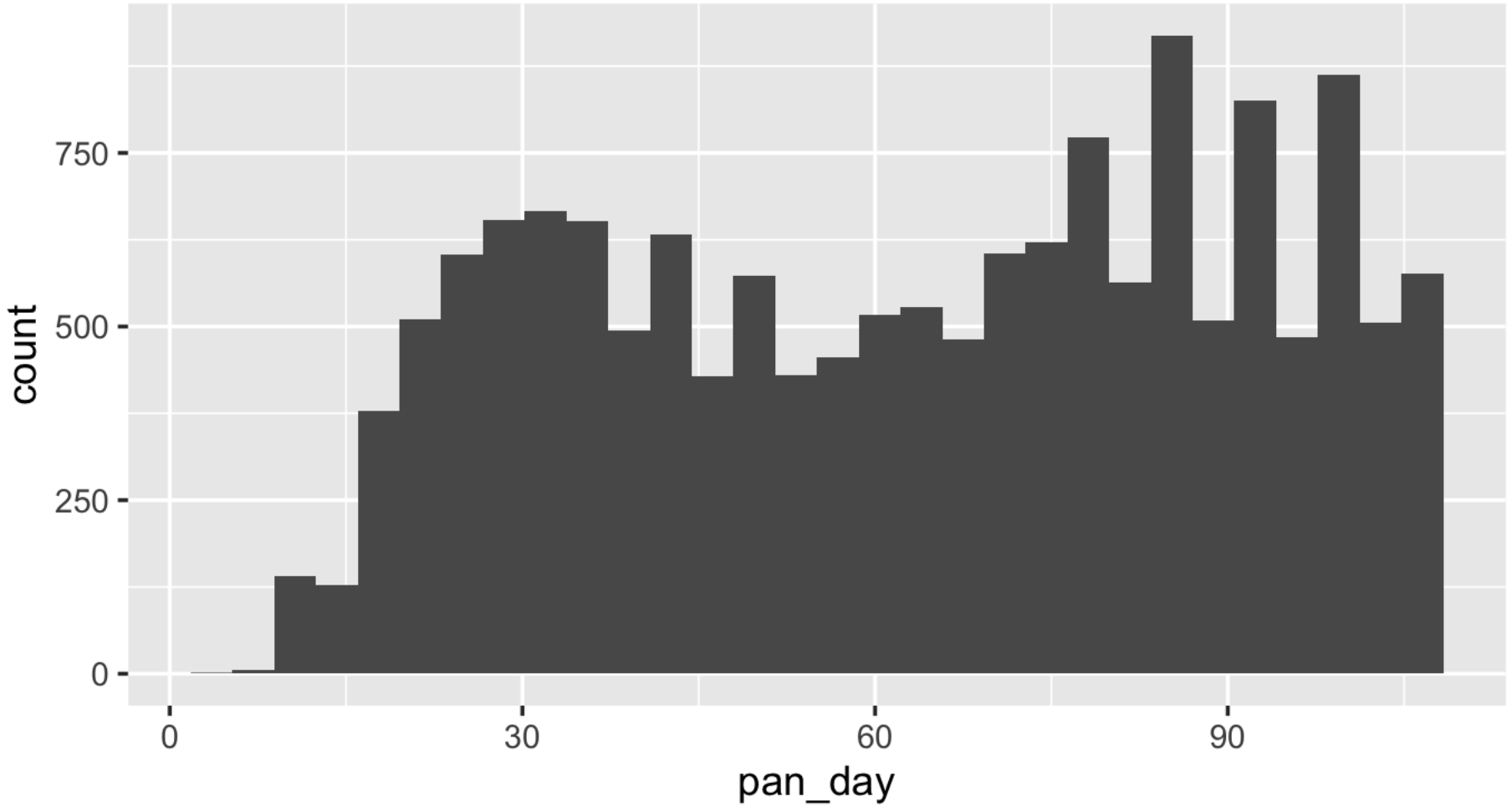
```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day))
```

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

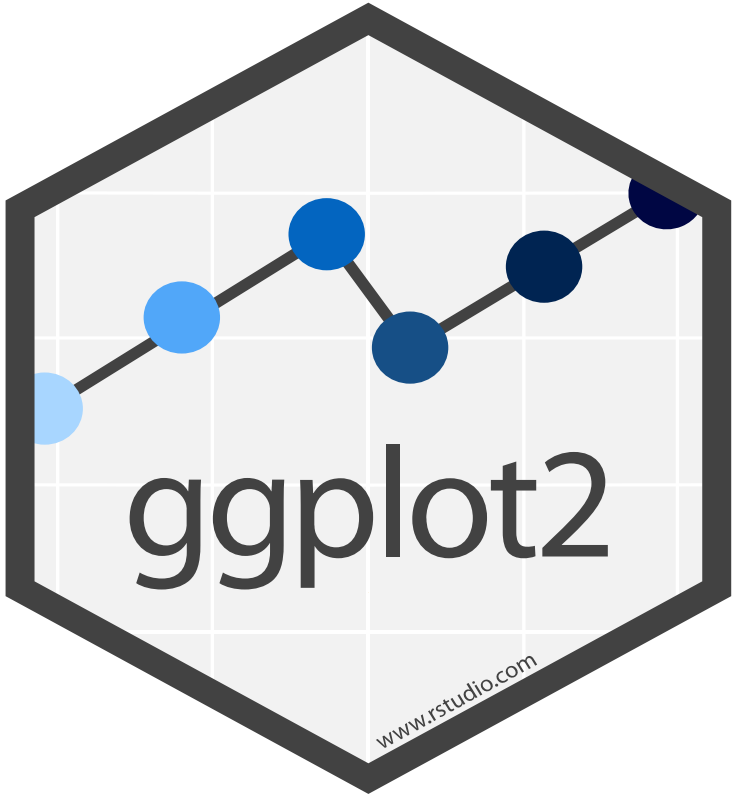
Often times, things that look like an error in R are actually just a message.

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.

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




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


ggplot2
www.rstudio.com

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. Pick a "tidy" **data** frame

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

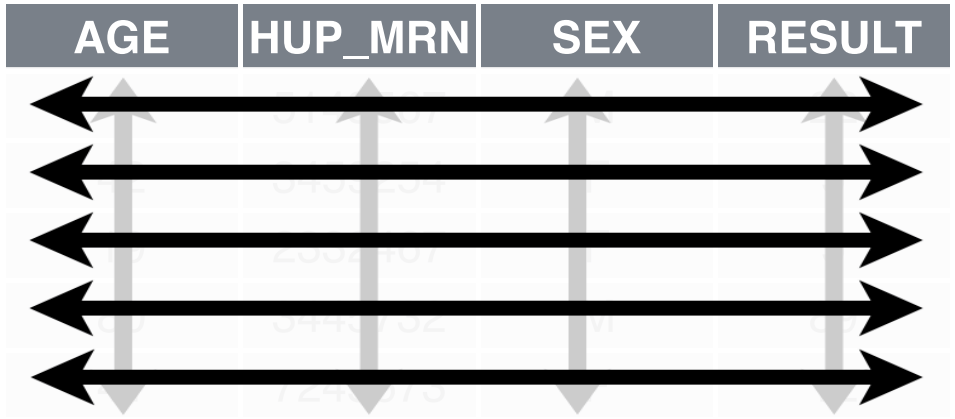
2. Pick a "**geom**" function

3. Write aesthetic **mappings**

1. Pick a “tidy” **data** frame

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

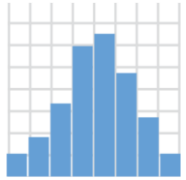
Wickham H. **Tidy Data.** *J Stat Soft* 2014.

1. Pick a “tidy” **data** frame

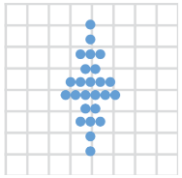
```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings))
```

2. Pick a “**geom**” function

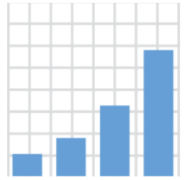
2. Pick a “Geom” Function



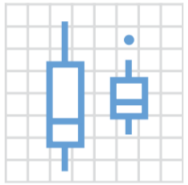
```
geom_histogram()
```



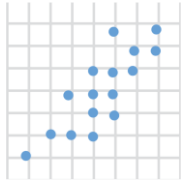
```
geom_dotplot()
```



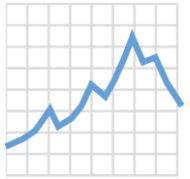
```
geom_bar()
```



```
geom_boxplot()
```



```
geom_point()
```



```
geom_line()
```

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

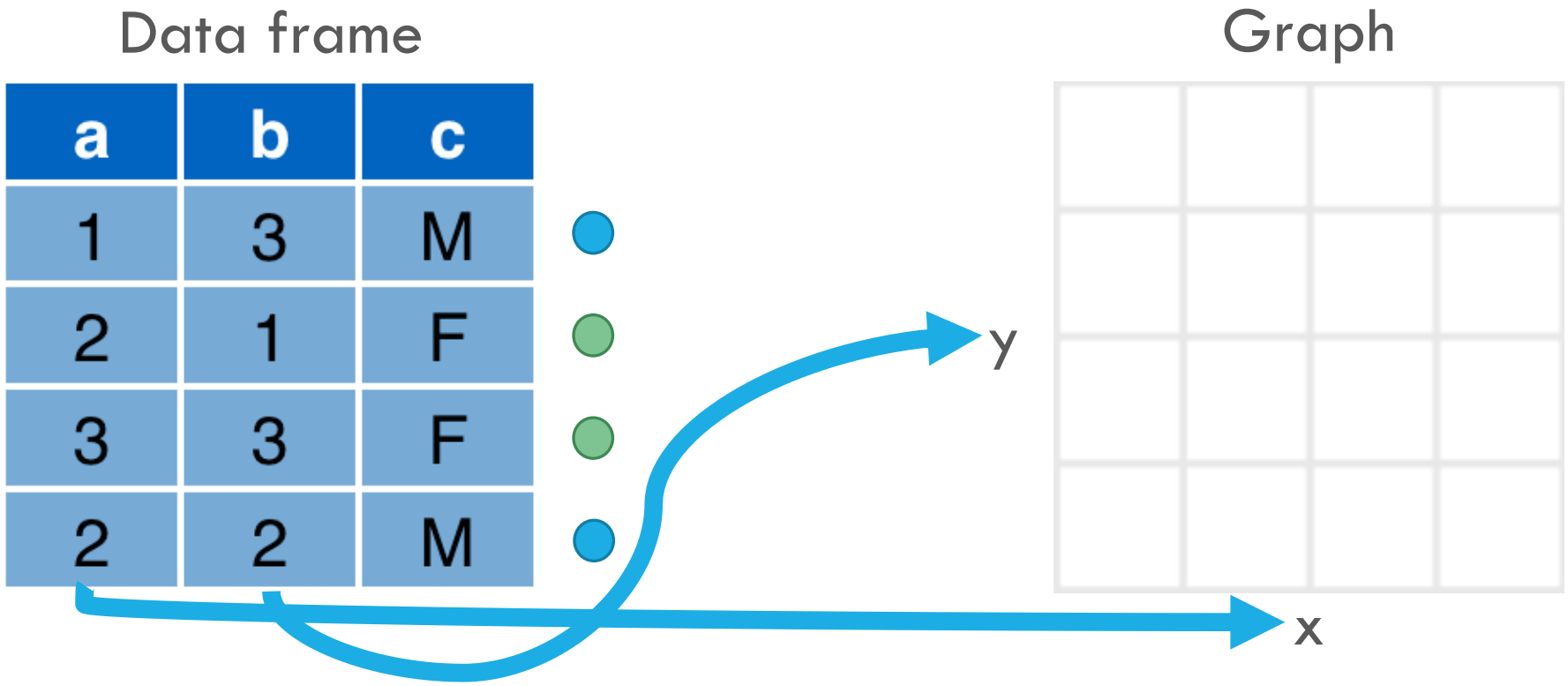
```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings))
```

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

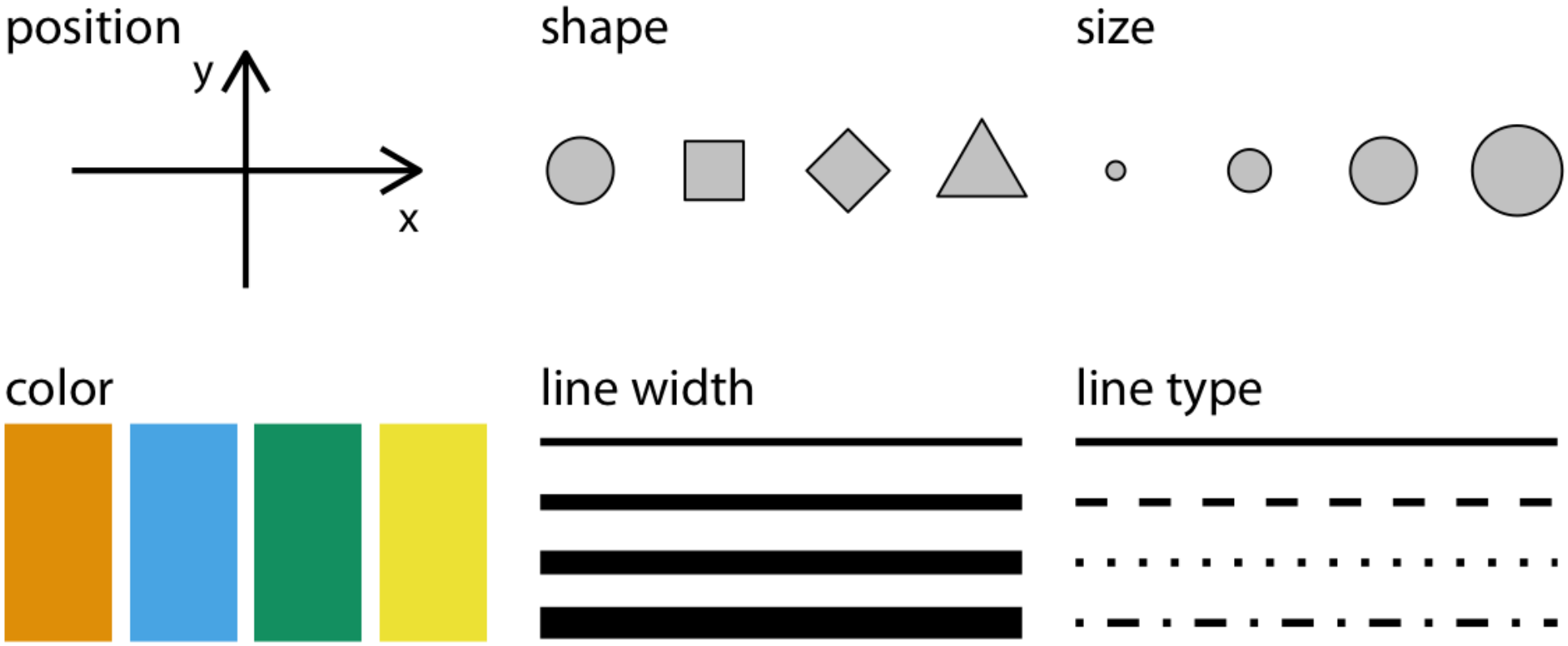
3. Write aesthetic **mappings**

3. Write Aesthetic Mappings

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



Aesthetics



From Fundamentals of Data Visualization, by Claus Wilke, licensed under CC-BY-NC-ND

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 4

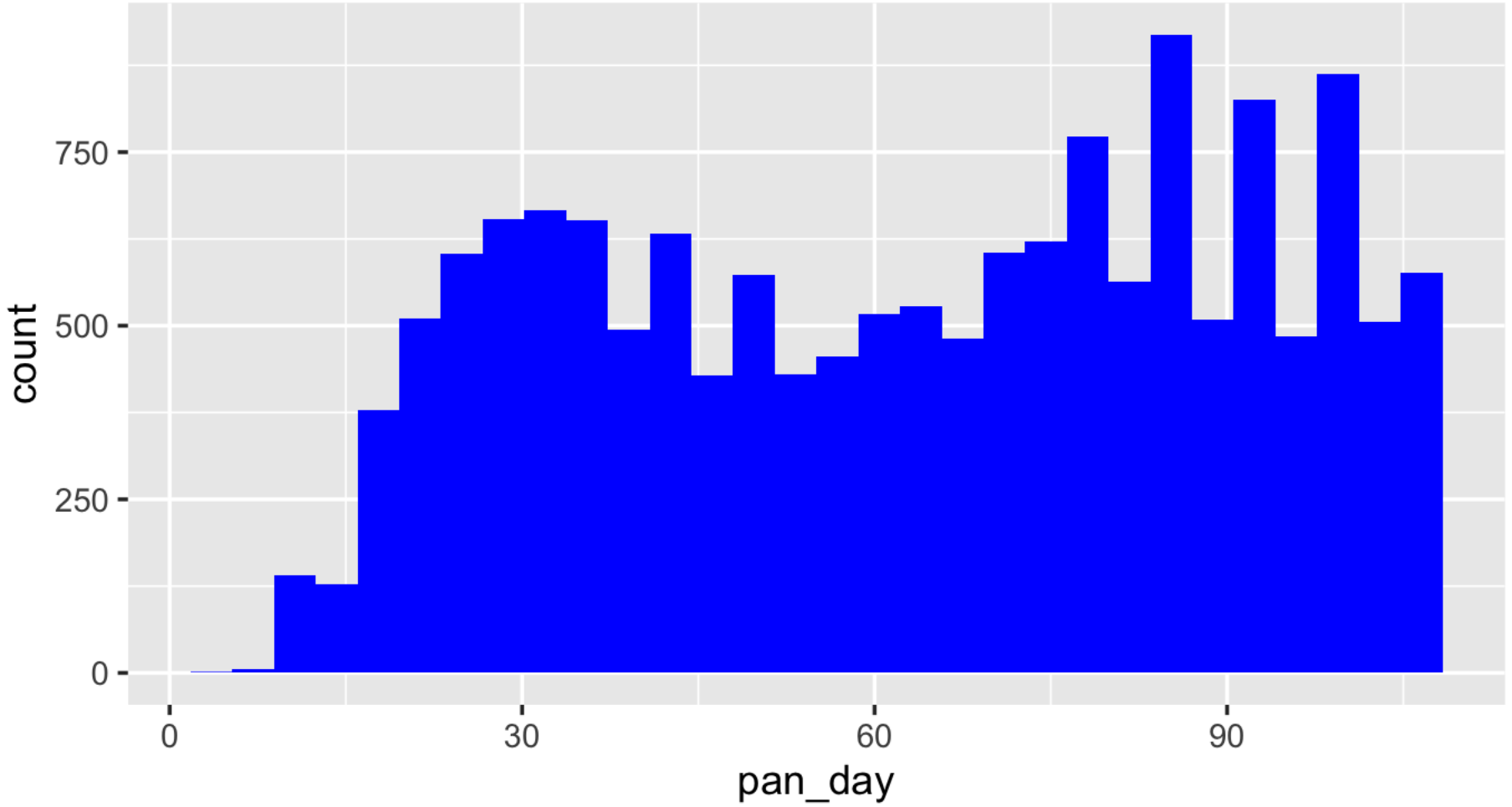
Open `03 - Visualize.Rmd`. Work through the exercises of the section titled "Your Turn 4".

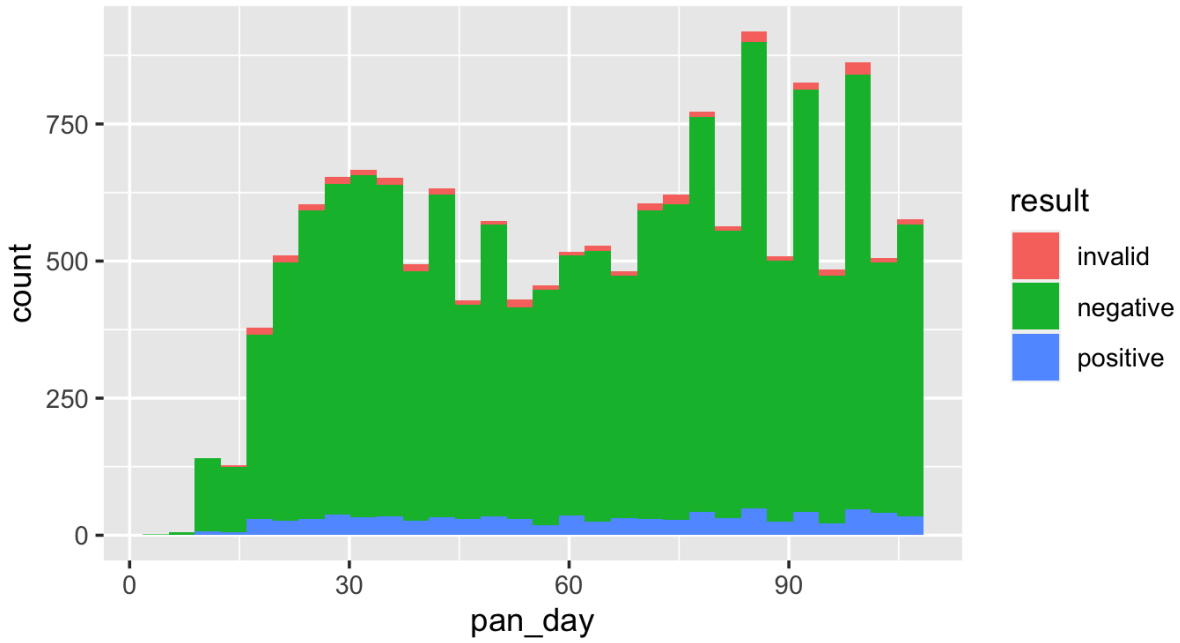
Stop when it says "Stop Here".

Click "yes" when you're done!

Setting vs Mapping Aesthetics

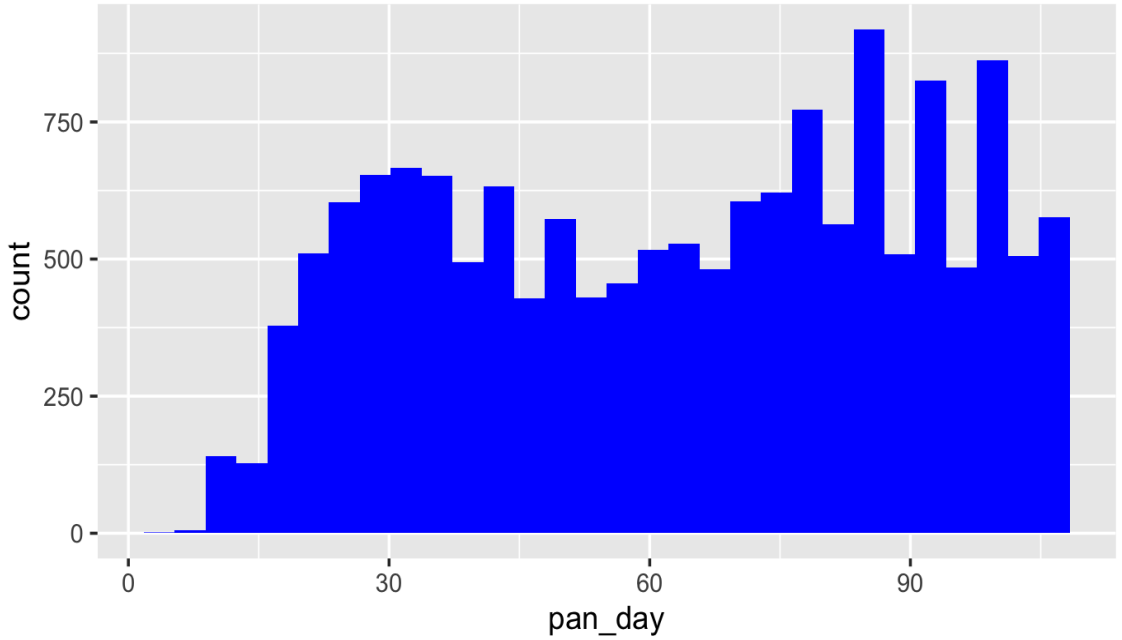
How would you make this plot?





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

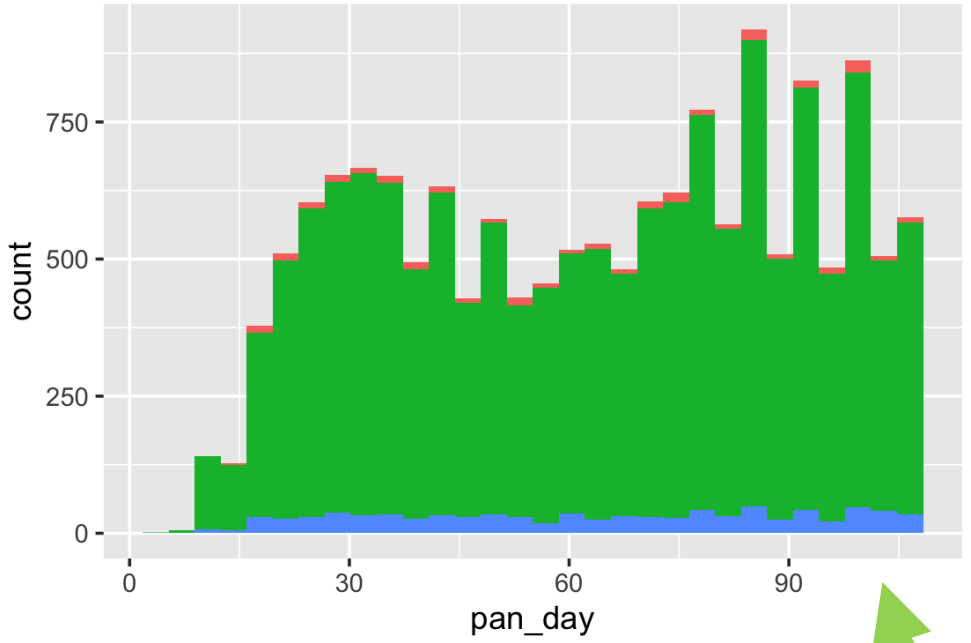
```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day, fill = result))
```

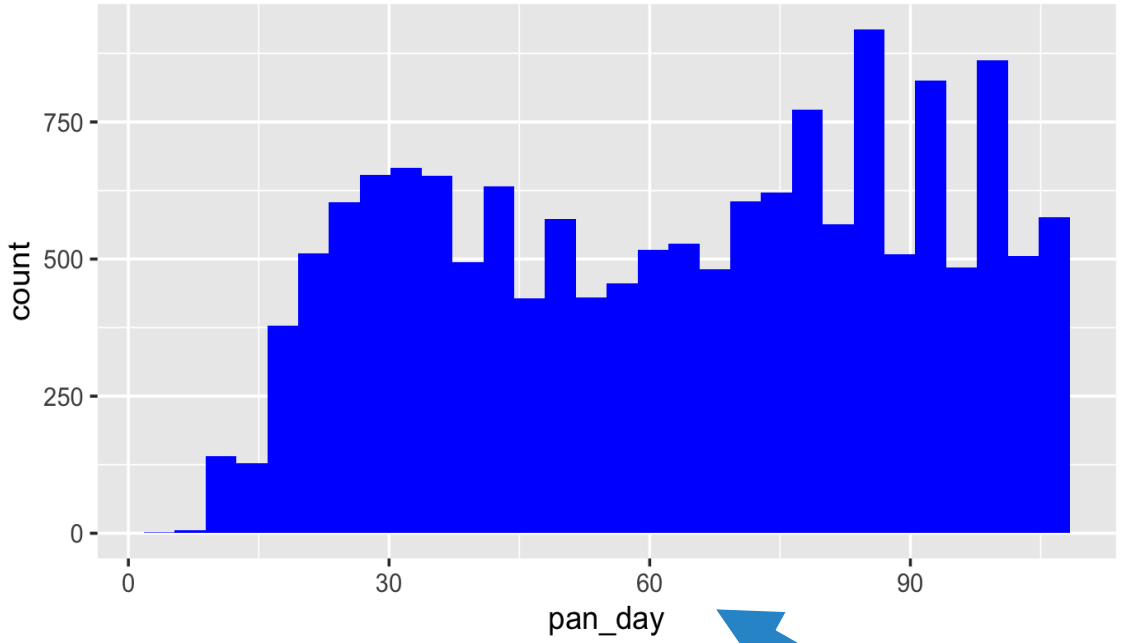


Outside of `aes()`:
set an aesthetic to a **value**

color name in "quotes"

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



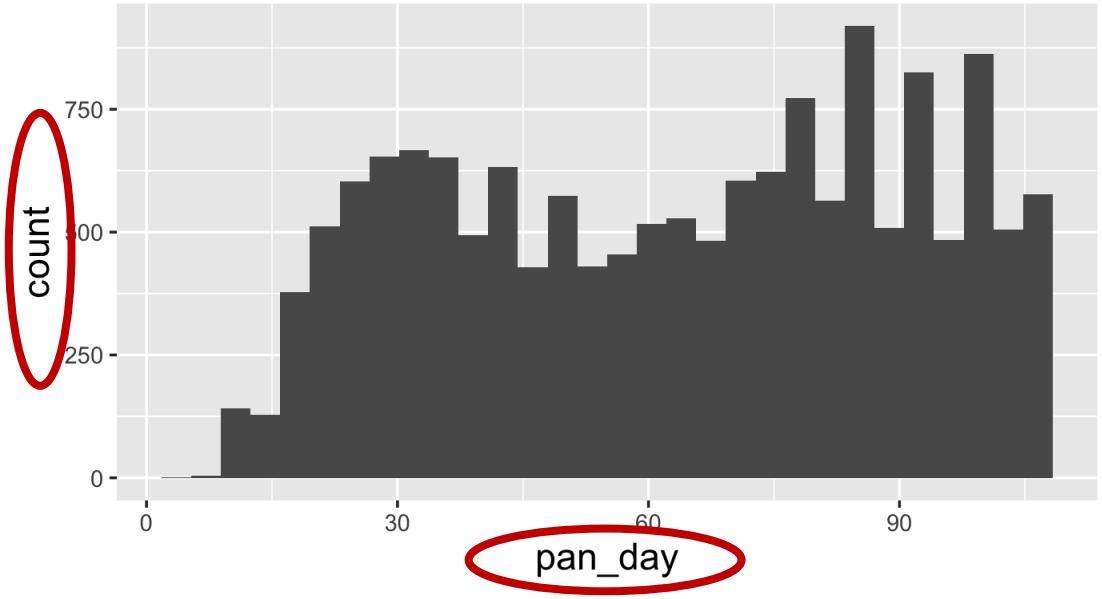


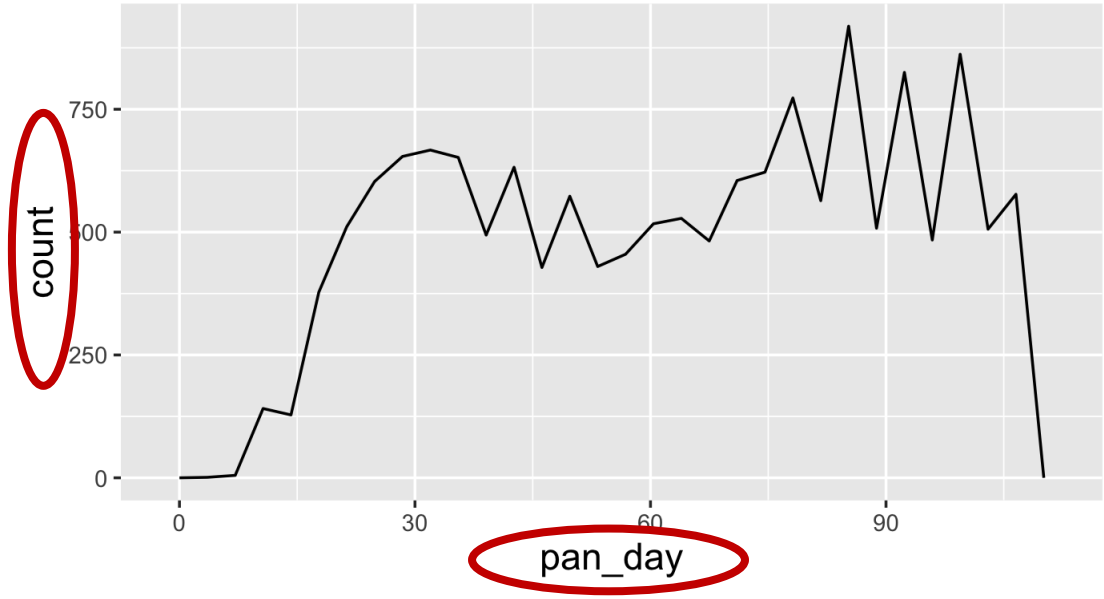

```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day, fill = result))
```

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

Geom Functions

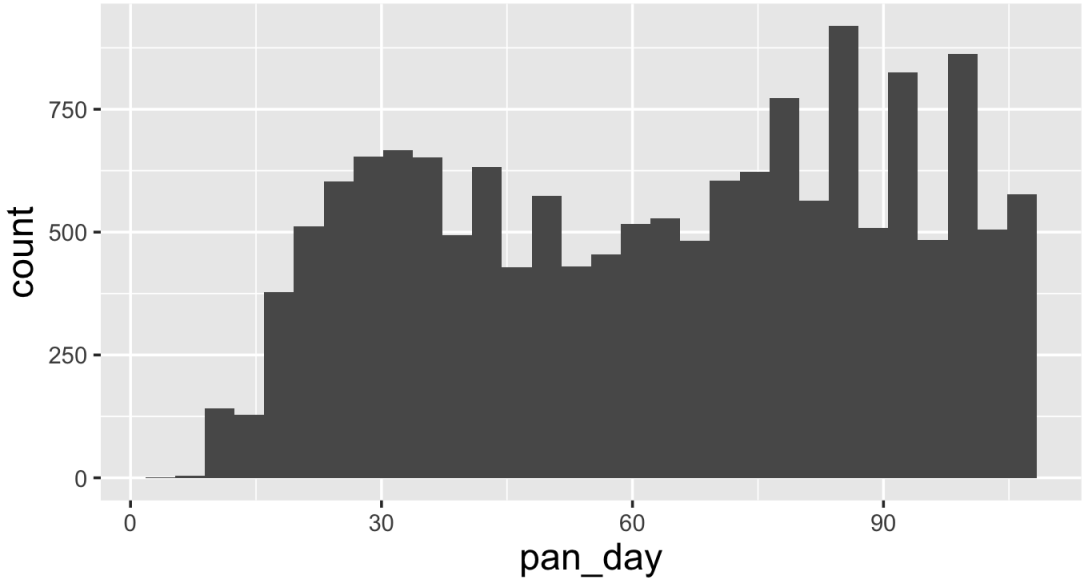
How are these plots similar?

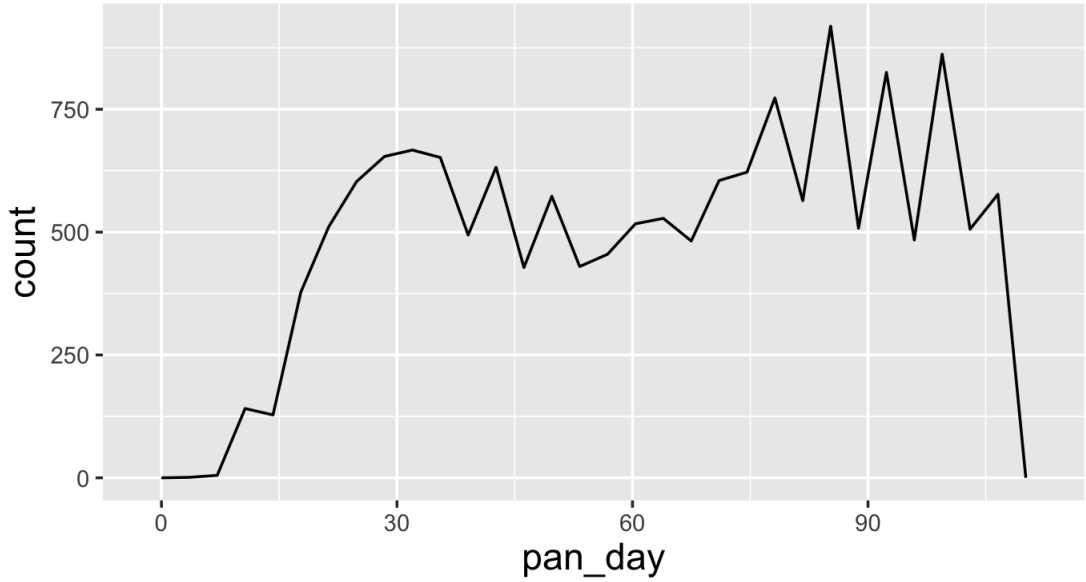




Same: x axis, y axis, data

How are these plots different?





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

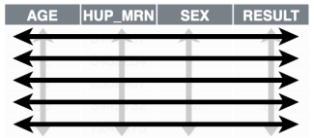
Your Turn 5

Return to `03-Visualize.Rmd`. Work through the exercises of the section titled "Your Turn 5."

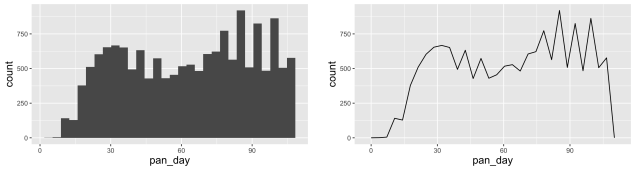
Click "yes" when you're done!

Recap

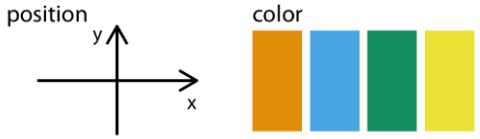
ggplot2 is a package that provides a **grammar of graphics.** You can create **any type of plot** using a simple template to which you provide:



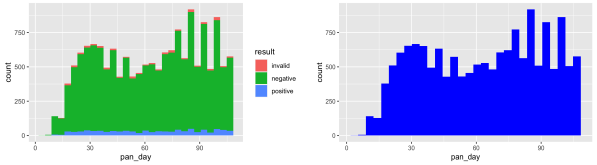
1. A **tidy data frame,** in which each **variable** is in its own **column**, each **observation** is in its own **row**, each **value** is in its own **cell**;



2. A **geom function,** which tells R what kind of plot to make; and



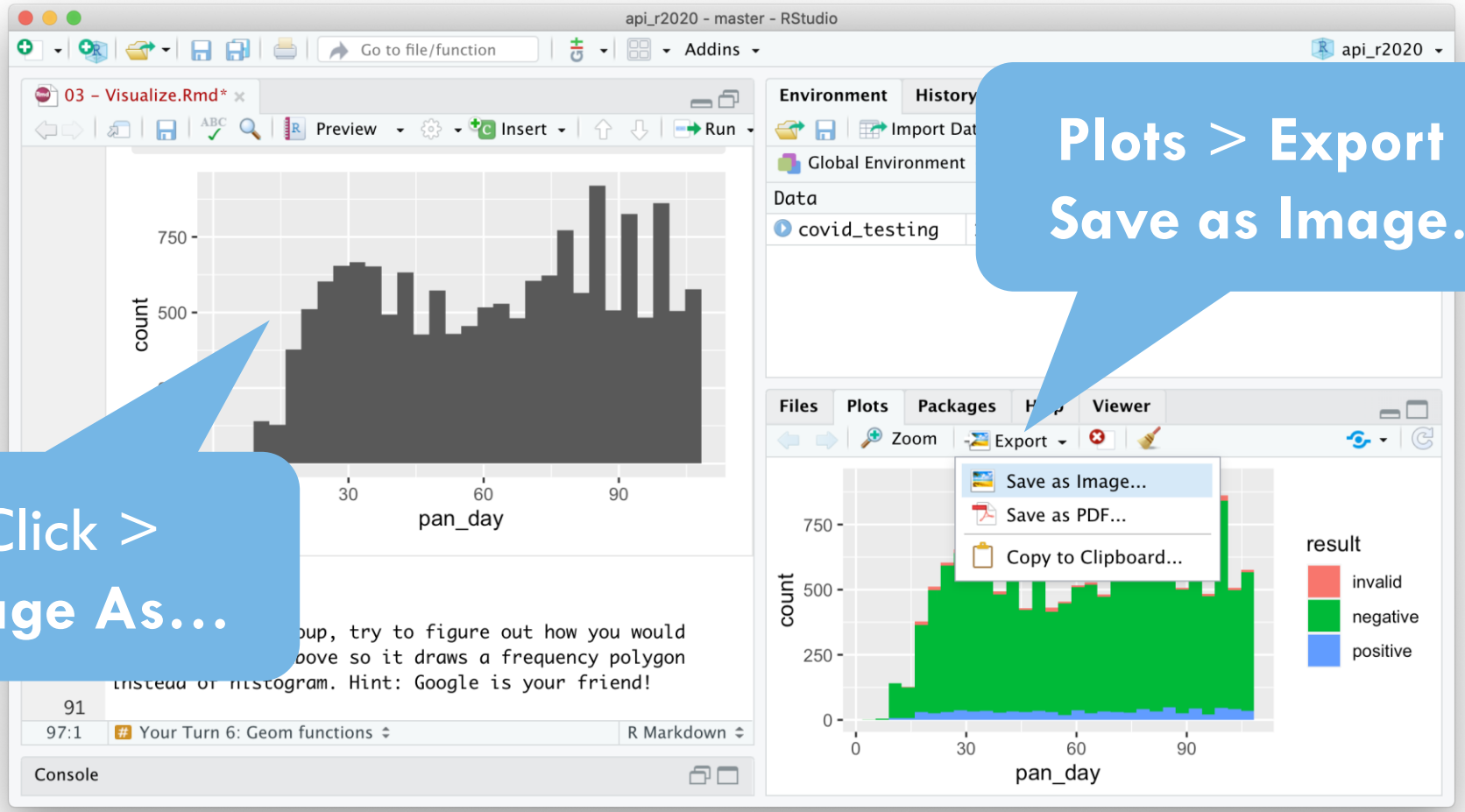
3. **Aesthetic mappings,** which tell R how to represent data as graphical markings on the plot.



Aesthetics can be **mapped** to a variable or **set** to a constant value.

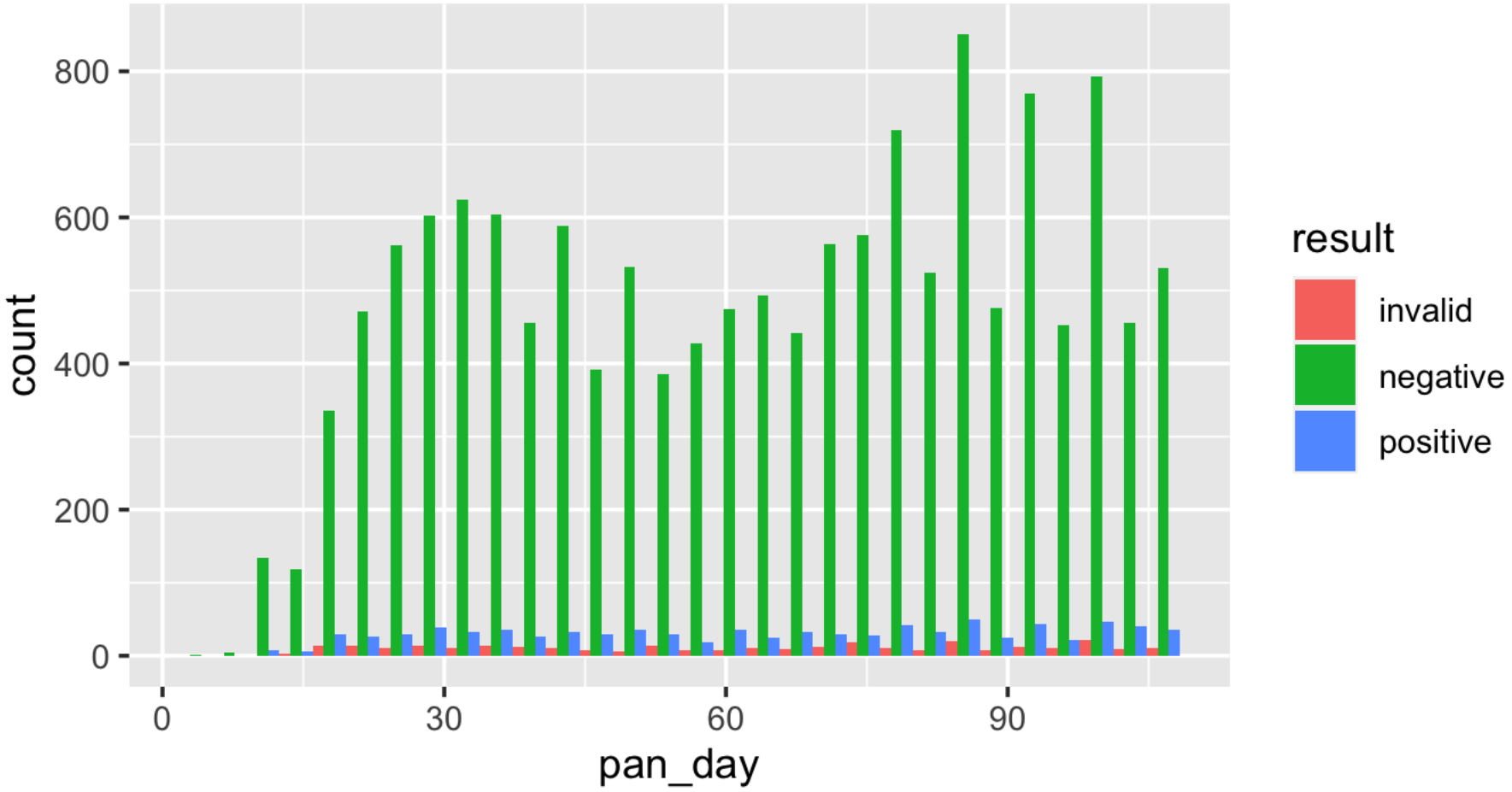
What Else?

Manually saving plots



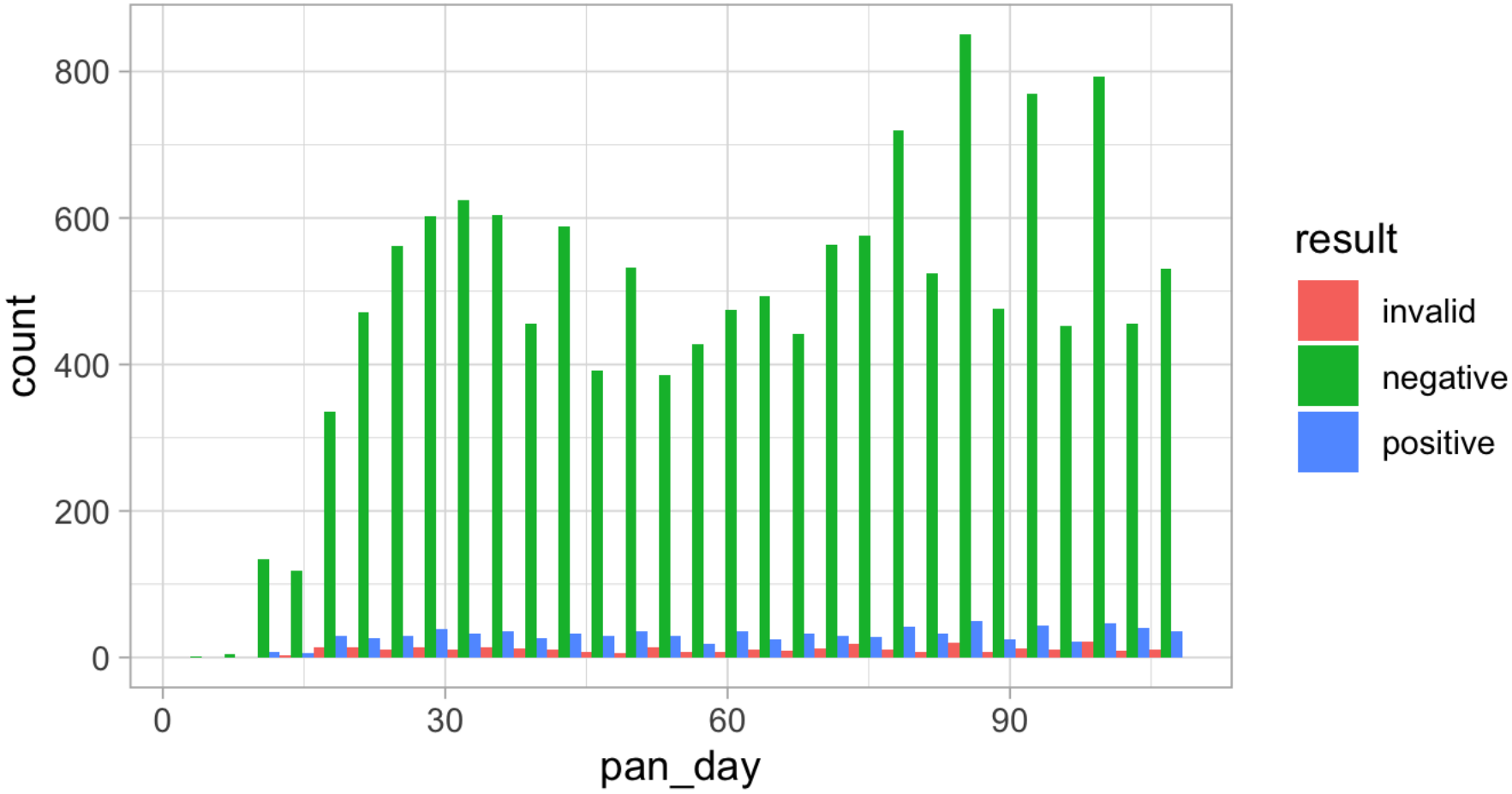
Position adjustments

How overlapping objects are arranged



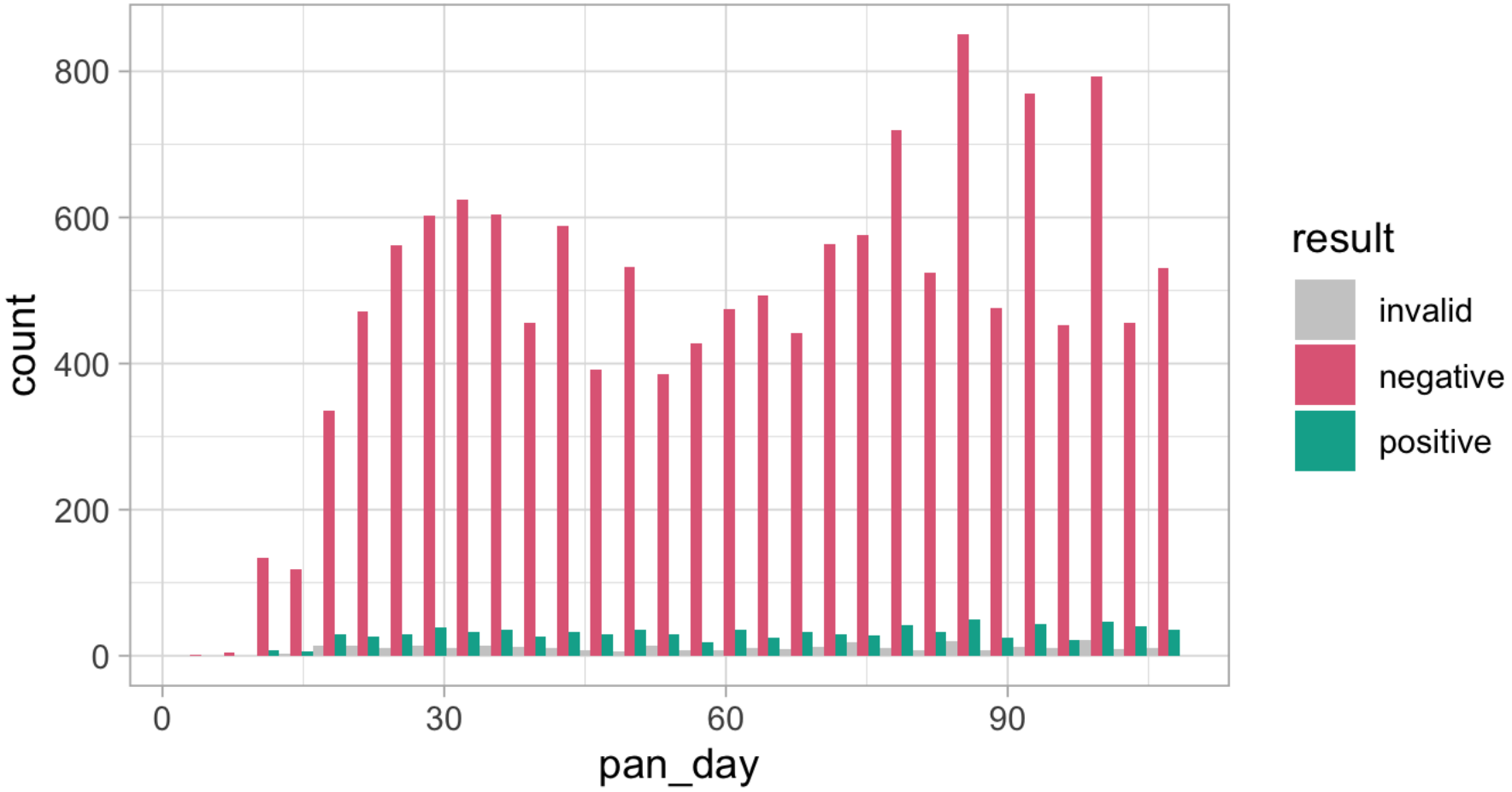
Themes

Visual appearance of non-data elements



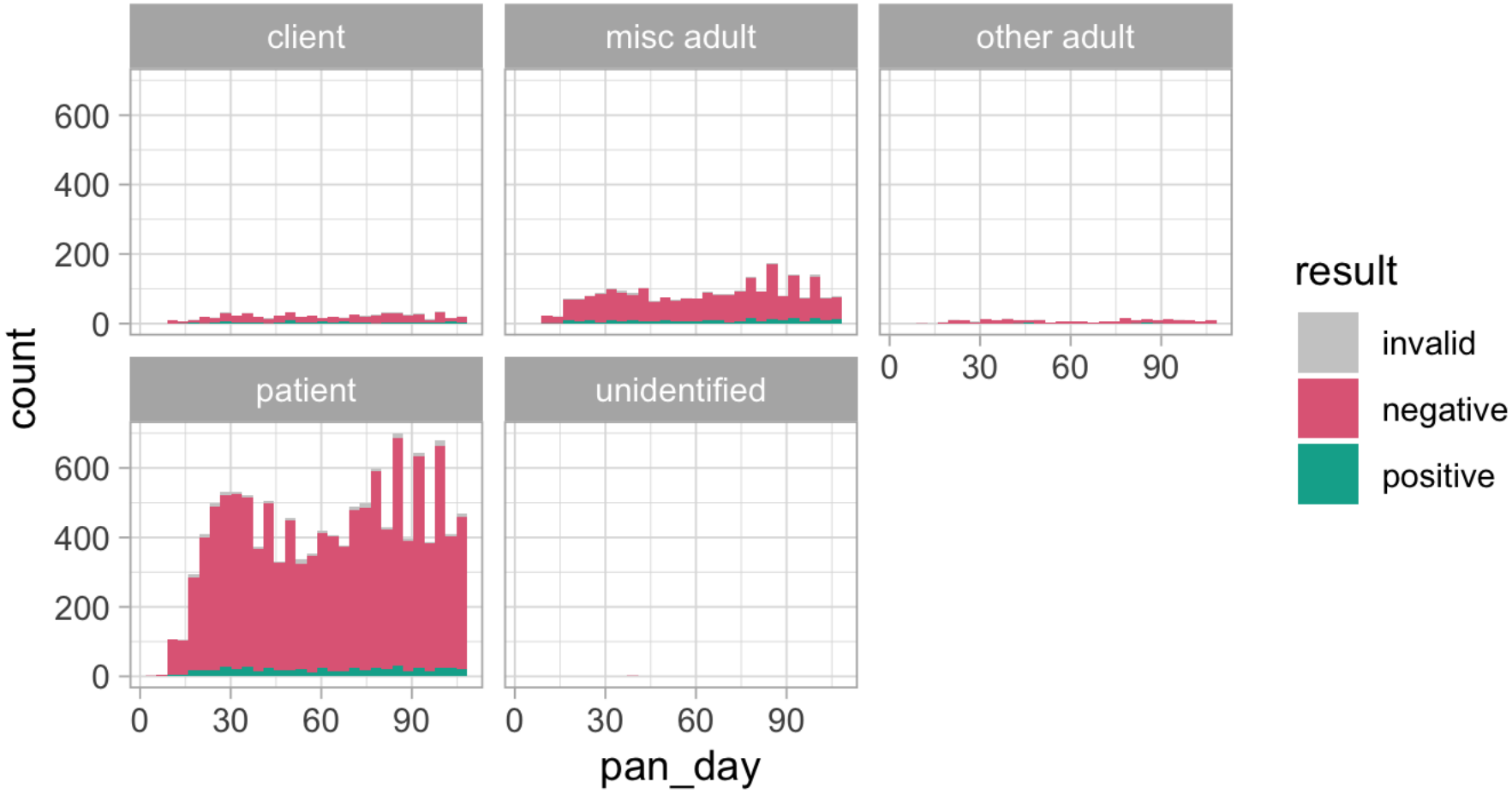
Scales

Customize color scales and other mappings

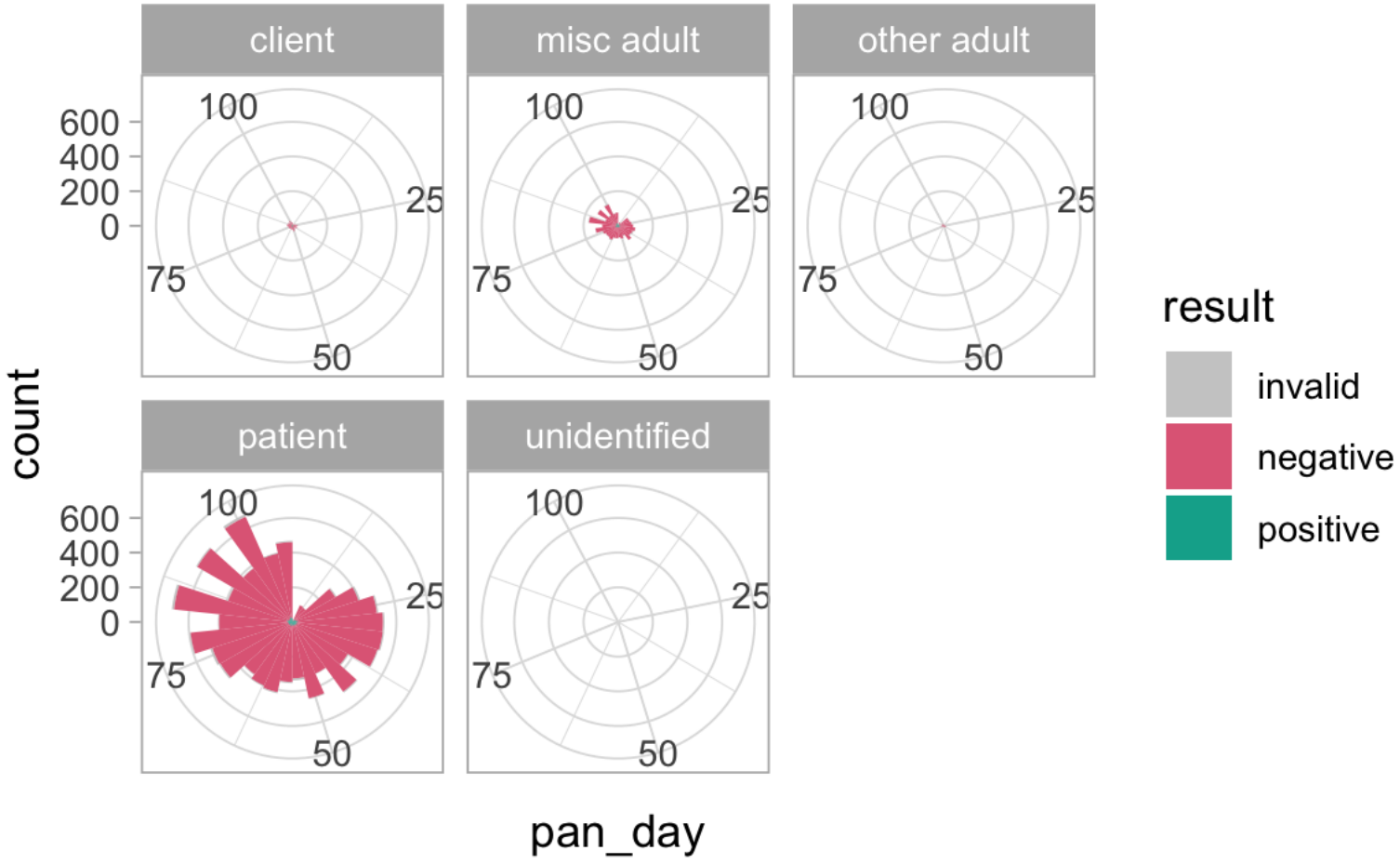


Facets

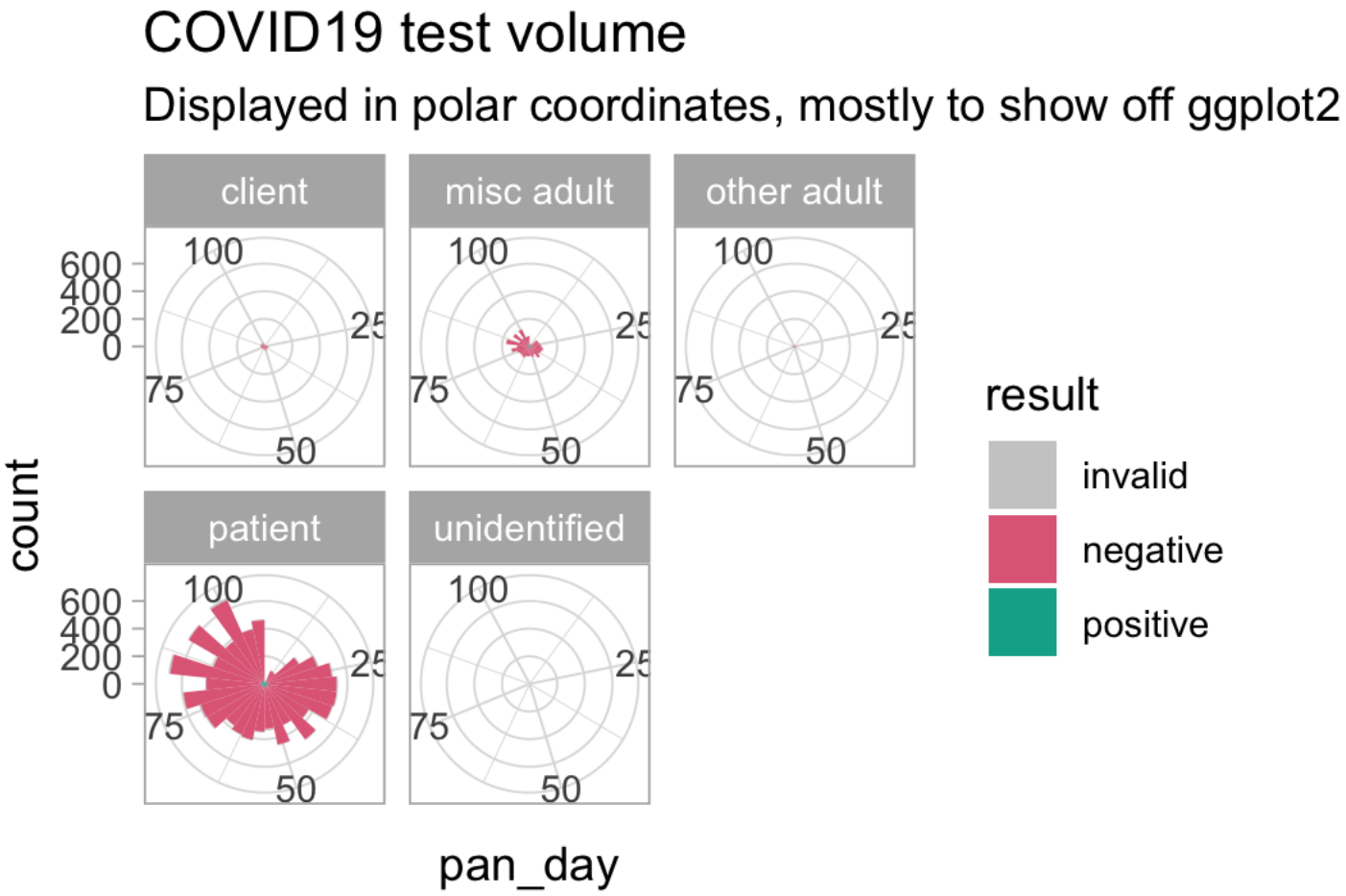
Subplots that display subsets of the data



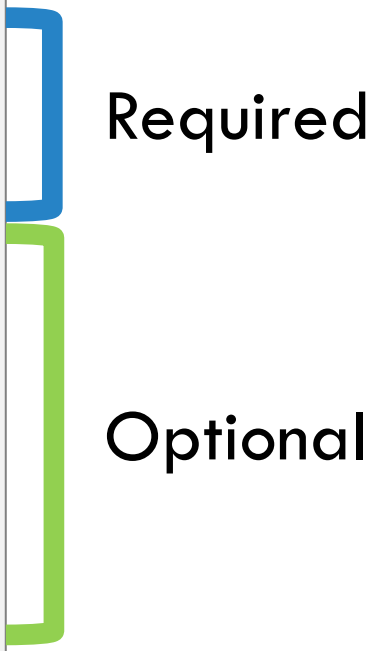
Coordinate systems



Titles and captions



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



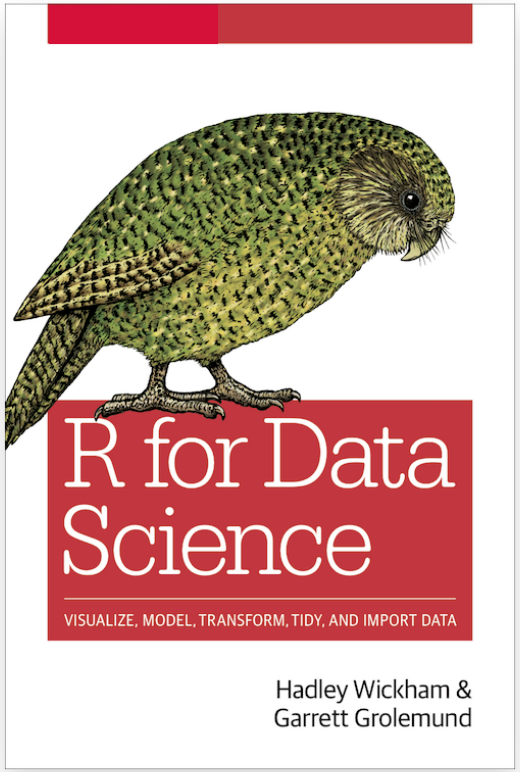
https://r4ds.had.co.nz/

R for Data Science

Search

Table of contents

Welcome



This is the website for **"R for Data Science"**. This book will teach you how to do data science with R: You'll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you'll learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You'll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You'll also learn how to manage cognitive resources to facilitate discoveries when wrangling, visualising, and exploring data.

https://www.rstudio.com/resources/cheatsheets/

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

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=1**)** - 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

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,

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

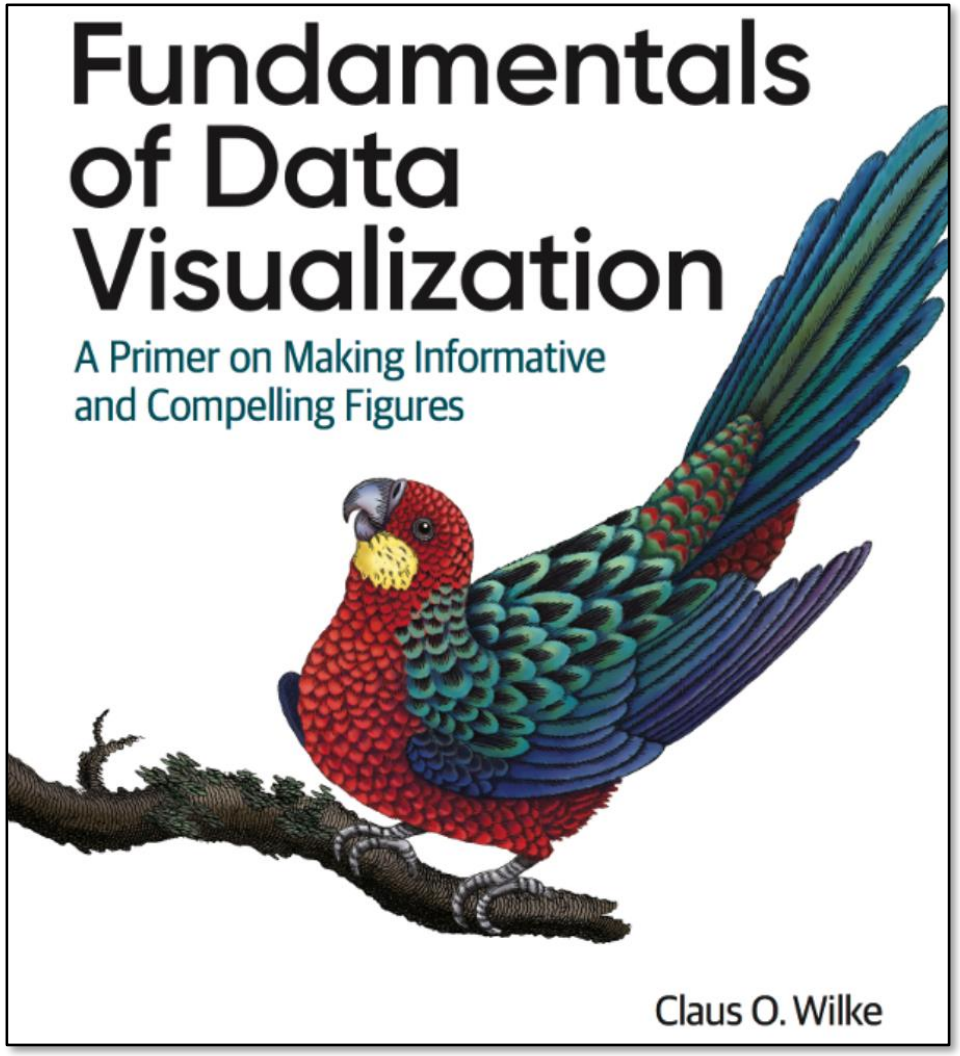
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

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,



https://clauswilke.com/dataviz/

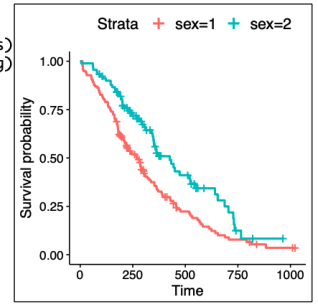
Creating Survival Plots

Informative and Elegant with *survminer*

Survival Curves

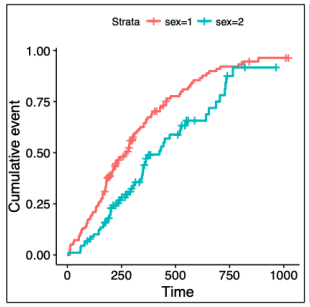
The **ggsurvplot()** function creates **ggplot2** plots from **survfit** objects.

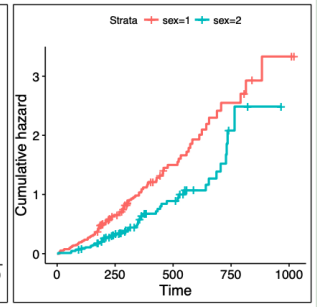
```
library("survival")
fit <- survfit(Surv(time,status)
               ~ sex, data = lung)

class(fit)
## [1] "survfit"

library("survminer")
ggsurvplot(fit, data = lung)
```



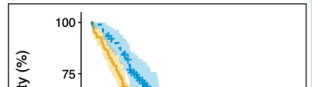
Use the **fun** argument to set the transformation of the survival curve. E.g. **"event"** for cumulative events, **"cumhaz"** for the cumulative hazard function or **"pct"** for survival probability in percentage.

```
ggsurvplot(fit, data = lung, fun = "event")
ggsurvplot(fit, data = lung, fun = "cumhaz")
```





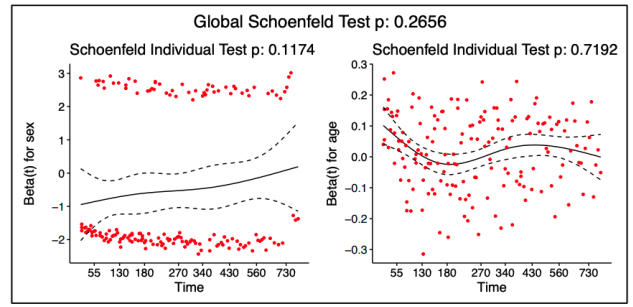
With lots of graphical parameters you have full control over look and feel of the survival



Diagnostics of Cox Model

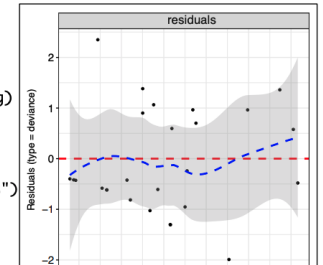
The function **cox.zph()** from **survival** package may be used to test the proportional hazards assumption for a Cox regression model fit. The graphical verification of this assumption may be performed with the function **ggcoxzph()** from the **survminer** package. For each covariate it produces plots with scaled Schoenfeld residuals against the time.

```
library("survival")
fit <- coxph(Surv(time, status) ~ sex + age, data = lung)
ftest <- cox.zph(fit)
ftest
##            rho chisq     p
## sex     0.1236 2.452 0.117
## age    -0.0275 0.129 0.719
## GLOBAL      NA 2.651 0.266
library("survminer")
ggcoxzph(ftest)
```



The function **ggcoxdiagnostics()** plots different types of residuals as a function of time, linear predictor or observation id. The type of residual is selected with **type** argument. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta"', "dfbetas", and "scaledsch".
The **ox.scale** argument defines what shall be plotted on the OX axis. Possible values are "linear.predictions", "observation.id", "time".
Logical arguments **hline** and **sline** may be used to add horizontal line or smooth line to the plot.

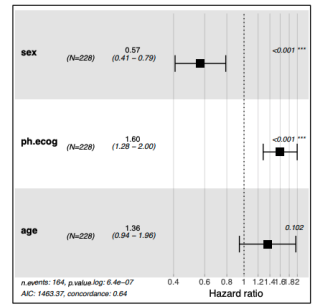
```
library("survival")
library("survminer")
fit <- coxph(Surv(time, status) ~
             sex + age, data = lung)


ggcoxdiagnostics(fit,
     type = "deviance",
     ox.scale = "linear.predictions")


ggcoxdiagnostics(fit,
     type = "schoenfeld",
```



Summary of Cox Model

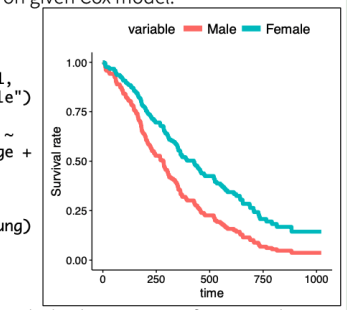
The function **ggforest()** from the **survminer** package creates a forest plot for a Cox regression model fit. Hazard ratio estimates along with confidence intervals and p-values are plotter for each variable.

```
library("survival")
library("survminer")
lung$age <- ifelse(lung$age > 70, ">70","<= 70")
fit <- coxph( Surv(time, status) ~ sex + ph.ecog + age, data = lung)
fit

## Call:
## coxph(formula = Surv(time, status) ~ sex+ph.ecog+age, data=lung)
##
##            coef exp(coef) se(coef)     z       p
## sex      -0.567     0.567    0.168 -3.37 0.00075
## ph.ecog   0.470     1.600    0.113  4.16 3.1e-05
## age>70    0.307     1.359    0.187  1.64 0.10175
##
## Likelihood ratio test=31.6  on
## n= 227, number of events= 164

ggforest(fit)
```

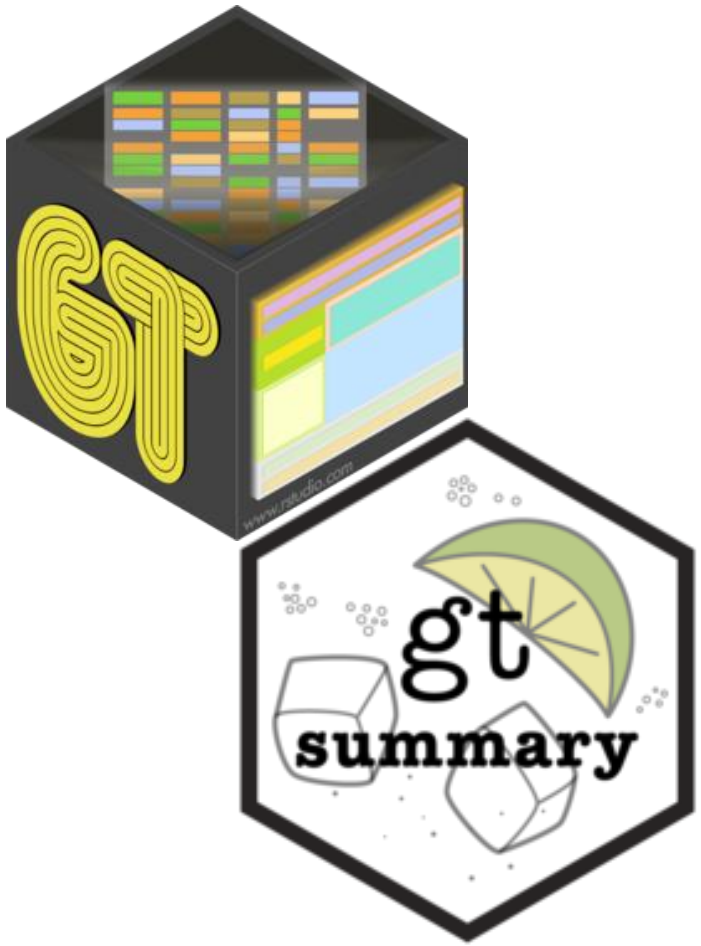


The function **ggadjustedcurves()** from the **survminer** package plots Adjusted Survival Curves for Cox Proportional Hazards Model. Adjusted Survival Curves show how a selected factor influences survival estimated from a Cox model.
Note that these curves differ from Kaplan Meier estimates since they present expected ssurvival based on given Cox model.

```
library("survival")
library("survminer")

lung$sex <- ifelse(lung$sex == 1,
                   "Male", "Female")

fit <- coxph(Surv(time, status) ~
             sex + ph.ecog + age +
             strata(sex),
             data = lung)

ggcoxadjustedcurves(fit, data=lung)
```



A grammar for **tables**



Variable	**N**	**Drug A**, N = 98[1]	**Drug B**, N = 102[1]	**p-value**[2]
Age	189	46 (37, 59)	48 (39, 56)	0.7
Grade	200			0.9
I		35 (36%)	33 (32%)	
II		32 (33%)	36 (35%)	
III		31 (32%)	33 (32%)	
Tumor Response	193	28 (29%)	33 (34%)	0.6

[1] Statistics presented: median (IQR); n (%)

[2] Statistical tests performed: Wilcoxon rank-sum test; chi-square test of independence

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