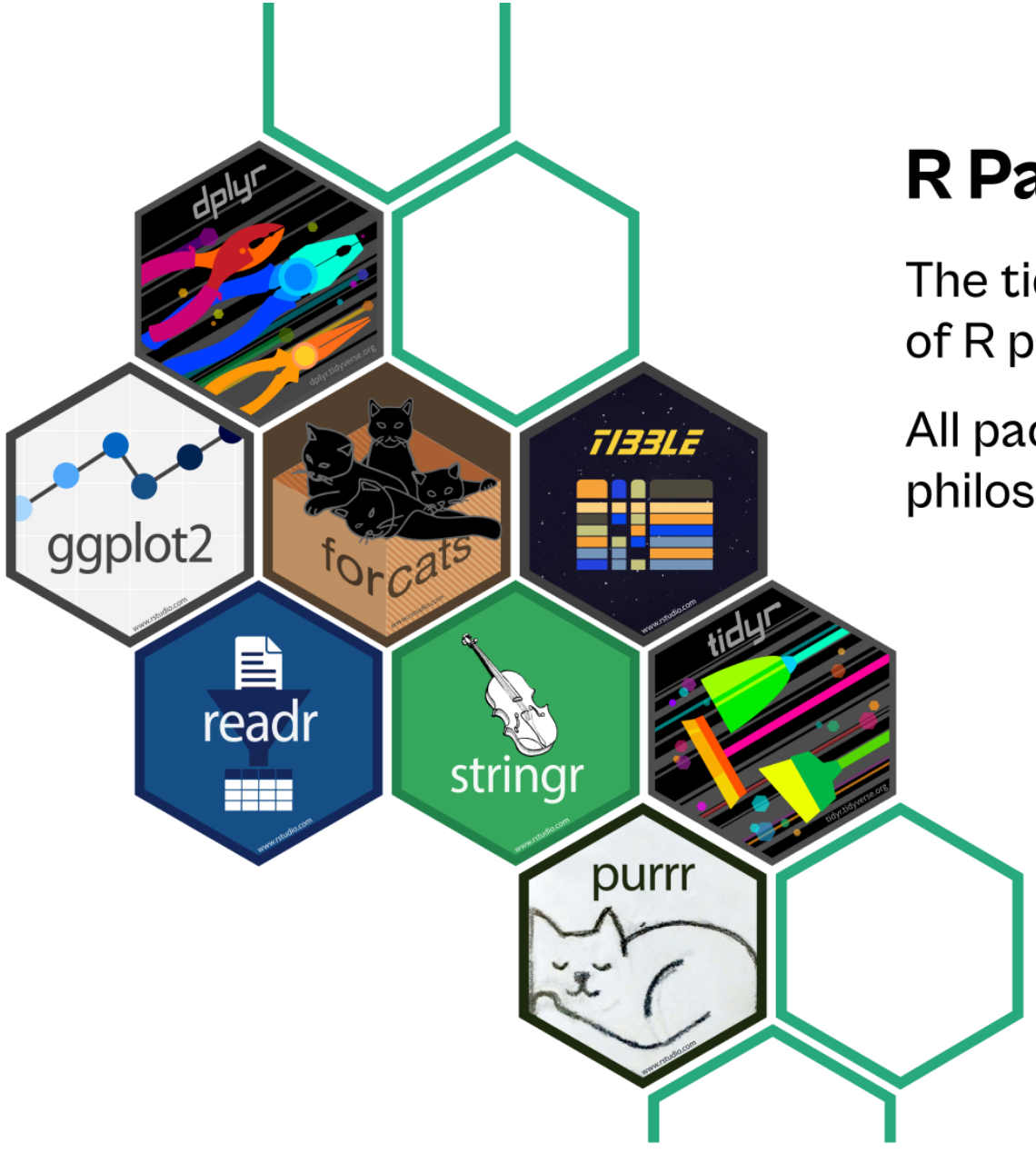
Graphic Design with ggplot2

Concepts of the {ggplot2} Package Pt. 1:
Data, Aesthetics, and Geometries + Misc Stuff

Cédric Scherer // rstudio::conf // July 2022

Setup



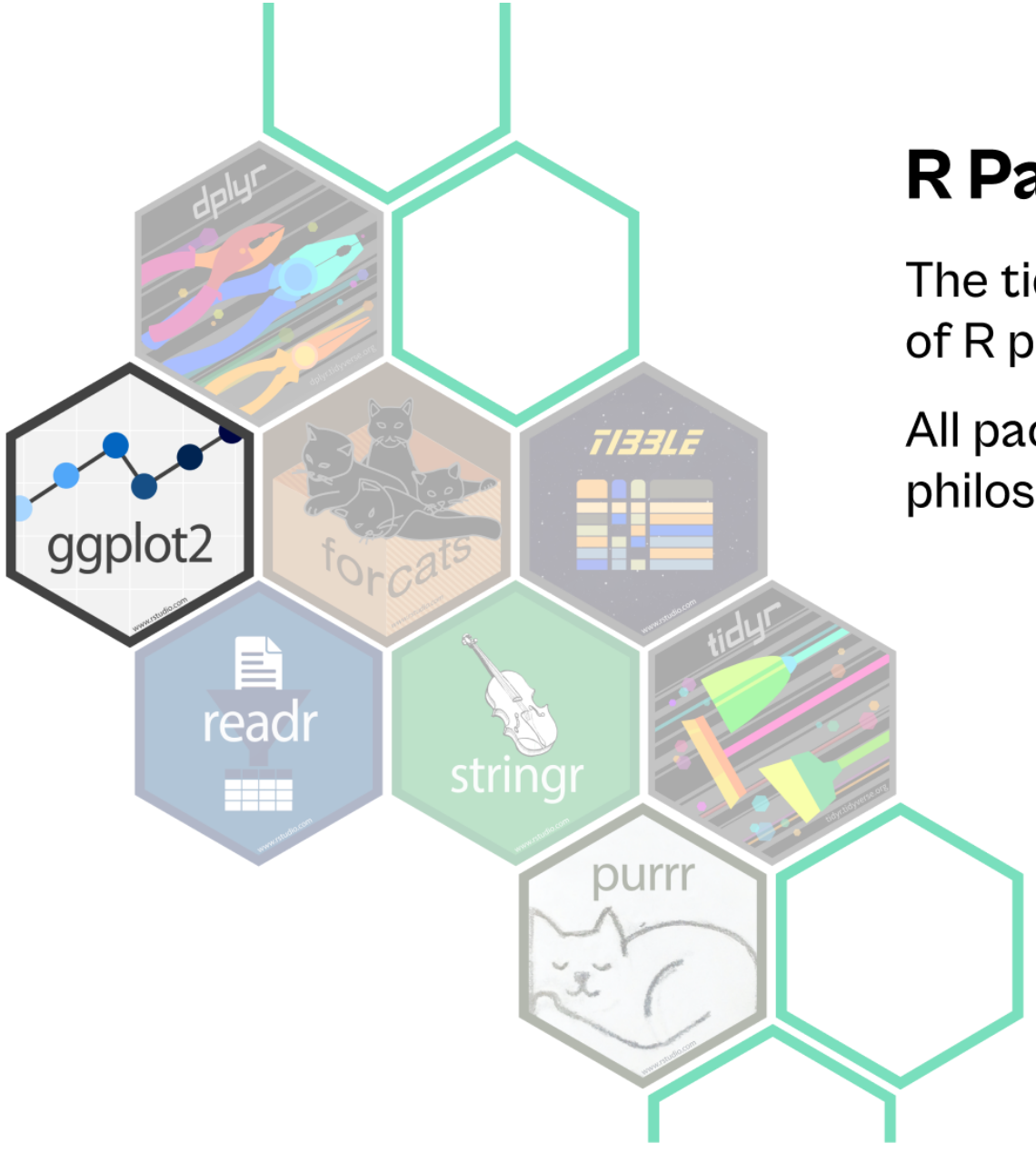
R Packages for Data Science

The tidyverse is an opinionated collection of R packages designed for data science.

All packages share an underlying design philosophy, grammar, and data structures.

Developed by RStudio®

Cédric Scherer // rstudio::conf // July 2022



R Packages for Data Science

The tidyverse is an opinionated collection of R packages designed for data science.

All packages share an underlying design philosophy, grammar, and data structures.

Developed by RStudio®

Cédric Scherer // rstudio::conf // July 2022

The ggplot2 Package

... is an **R package to visualize data** created by Hadley Wickham in 2005

```
# install.packages("ggplot2")
library(ggplot2)
```

... is part of the {tidyverse}

```
# install.packages("tidyverse")
library(tidyverse)
```

The Grammar of {ggplot2}

Cédric Scherer // rstudio::conf // July 2022

The Grammar of {ggplot2}

Component	Function	Explanation
Data	`ggplot(data)`	*The raw data that you want to visualise.*
Aesthetics	`aes()`	*Aesthetic mappings between variables and visual properties..*
Geometries	`geom_*()`	*The geometric shapes representing the data.*

The Grammar of {ggplot2}

Component	Function	Explanation
Data	`ggplot(data)`	*The raw data that you want to visualise.*
Aesthetics	`aes()`	*Aesthetic mappings between variables and visual properties..*
Geometries	`geom_*()`	*The geometric shapes representing the data.*
Statistics	`stat_*()`	*The statistical transformations applied to the data.*
Scales	`scale_*()`	*Maps between the data and the aesthetic dimensions.*
Coordinate System	`coord_*()`	*Maps data into the plane of the data rectangle.*
Facets	`facet_*()`	*The arrangement of the data into a grid of plots.*
Visual Themes	`theme()` and `theme_*()`	*The overall visual defaults of a plot.*

A Basic ggplot Example

The Data

Bike sharing counts in London, UK, powered by TfL Open Data

- covers the years 2015 and 2016
- incl. weather data acquired from freemeteo.com
- prepared by Hristo Mavrodiev for Kaggle
- further modification by myself

```
bikes <- readr::read_csv(
  "https://raw.githubusercontent.com/z3tt/graphic-design-ggplot2/main/data/london-bikes-custom.csv",
  col_types = "Dcfffillddddc"
)

bikes$season <- forcats::fct_inorder(bikes$season)
```

Variable	Description	Class
date	Date encoded as `YYYY-MM-DD`	date
day_night	`day` (6:00am–5:59pm) or `night` (6:00pm–5:59am)	character
year	`2015` or `2016`	factor
month	`1` (January) to `12` (December)	factor
season	`winter`, `spring`, `summer`, or `autumn`	factor
count	Sum of reported bikes rented	integer
is_workday	`TRUE` being Monday to Friday and no bank holiday	logical
is_weekend	`TRUE` being Saturday or Sunday	logical
is_holiday	`TRUE` being a bank holiday in the UK	logical
temp	Average air temperature (˚C)	double
temp_feel	Average feels like temperature (˚C)	double
humidity	Average air humidity (%)	double
wind_speed	Average wind speed (km/h)	double
weather_type	Most common weather type	character

ggplot2::ggplot()

ggplot: Create a new ggplot

Description

`ggplot()` initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

Arguments

`data`	Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
`mapping`	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
`...`	Other arguments passed on to methods. Not currently used.
`environment`	DEPRECATED. Used prior to tidy evaluation.

Details

`ggplot()` is used to construct the initial plot object, and is almost always followed by `+` to add component to the plot. There are three common ways to invoke `ggplot()`:

- `ggplot(df, aes(x, y, other aesthetics))`
- `ggplot(df)`
- `ggplot()`

Data

```
1  ggplot(data = bikes)
```

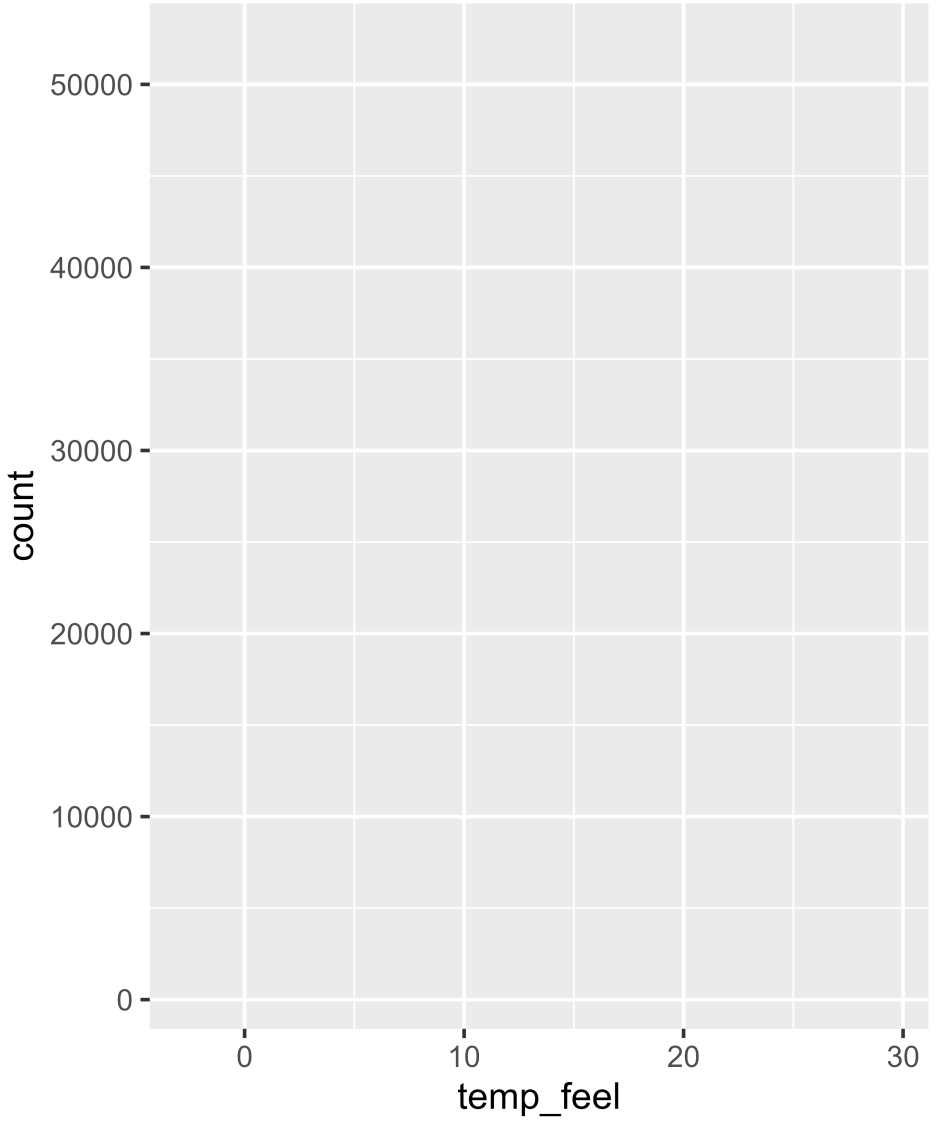
Aesthetic Mapping

= link variables to graphical properties

- positions (`x`, `y`)
- colors (`color`, `fill`)
- shapes (`shape`, `linetype`)
- size (`size`)
- transparency (`alpha`)
- groupings (`group`)

Aesthetic Mapping

```
1  ggplot(data = bikes) +
2    aes(x = temp_feel, y = count)
```



aesthetics

`aes()` outside as component

```
1  ggplot(data = bikes) +
2    aes(x = temp_feel, y = count)
```

`aes()` inside, explicit matching

```
1  ggplot(data = bikes, mapping = aes(x = temp_feel, y = count))
```

`aes()` inside, implicit matching

```
1  ggplot(bikes, aes(temp_feel, count))
```

`aes()` inside, mixed matching

```
1  ggplot(bikes, aes(x = temp_feel, y = count))
```

Geometrical Layers

Cédric Scherer // rstudio::conf // July 2022

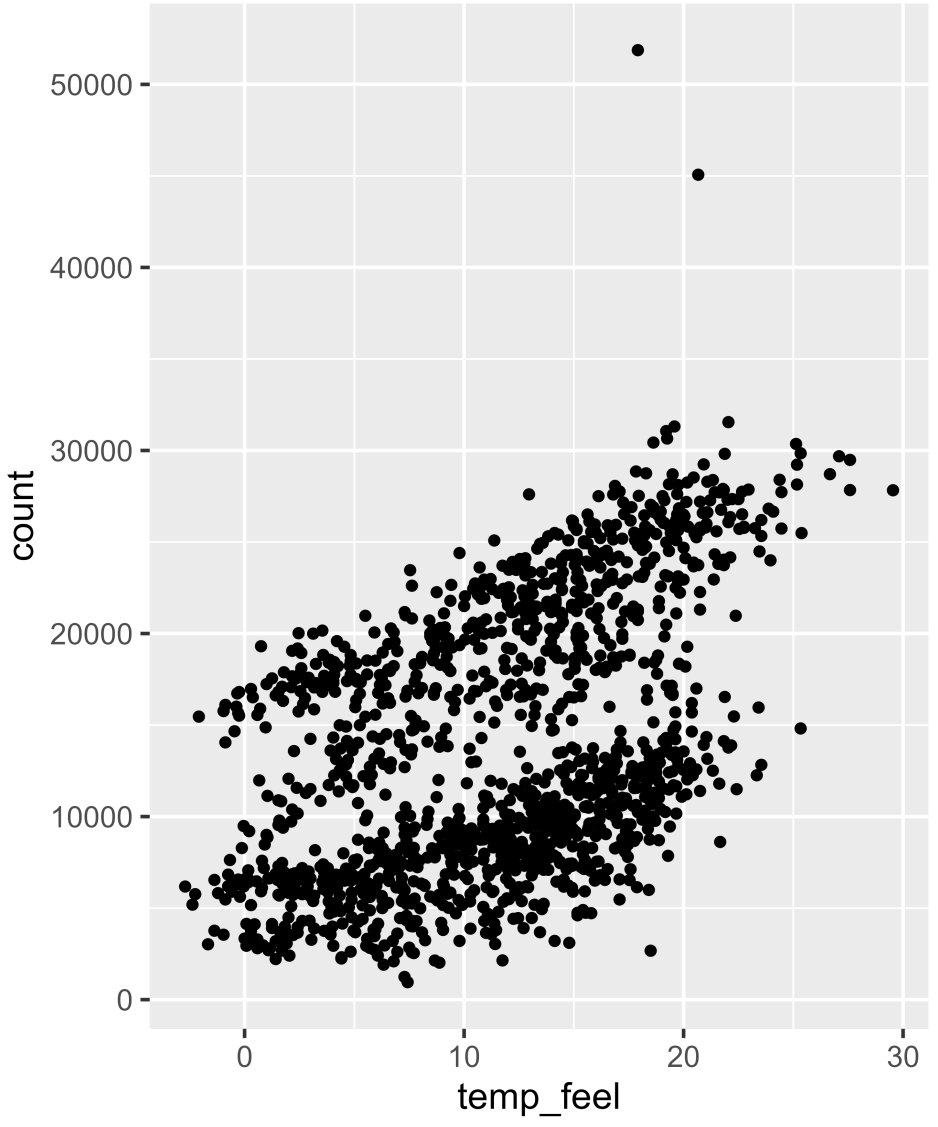
Geometries

= interpret aesthetics as graphical representations

- points
- lines
- polygons
- text labels
- ...

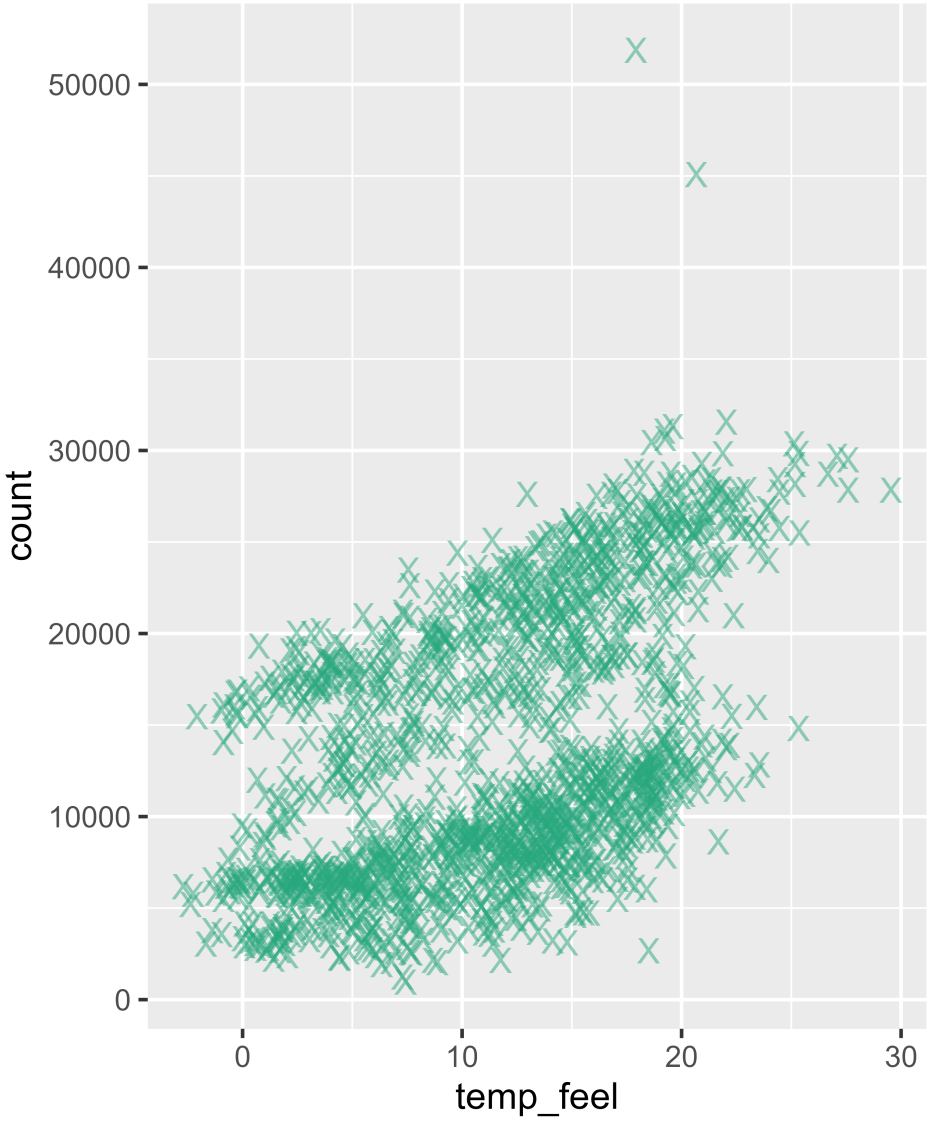
Geometries

```
1 ggplot(
2     bikes,
3     aes(x = temp_feel, y = count)
4   ) +
5   geom_point()
```



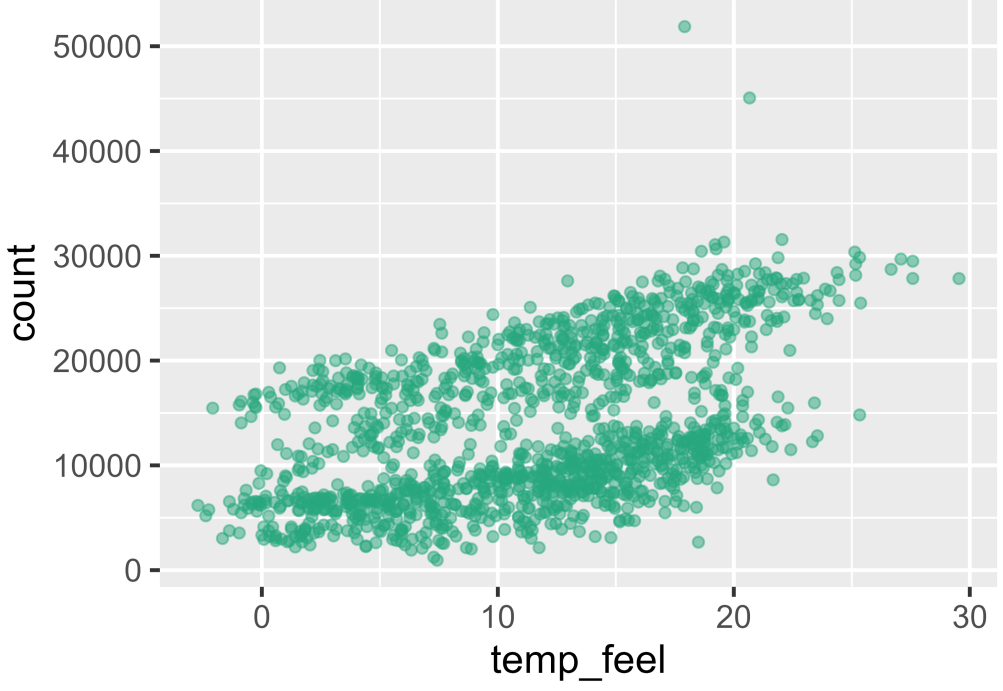
Visual Properties of Layers

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    color = "#28a87d",
    alpha = .5,
    shape = "X",
    stroke = 1,
    size = 4
  )
```

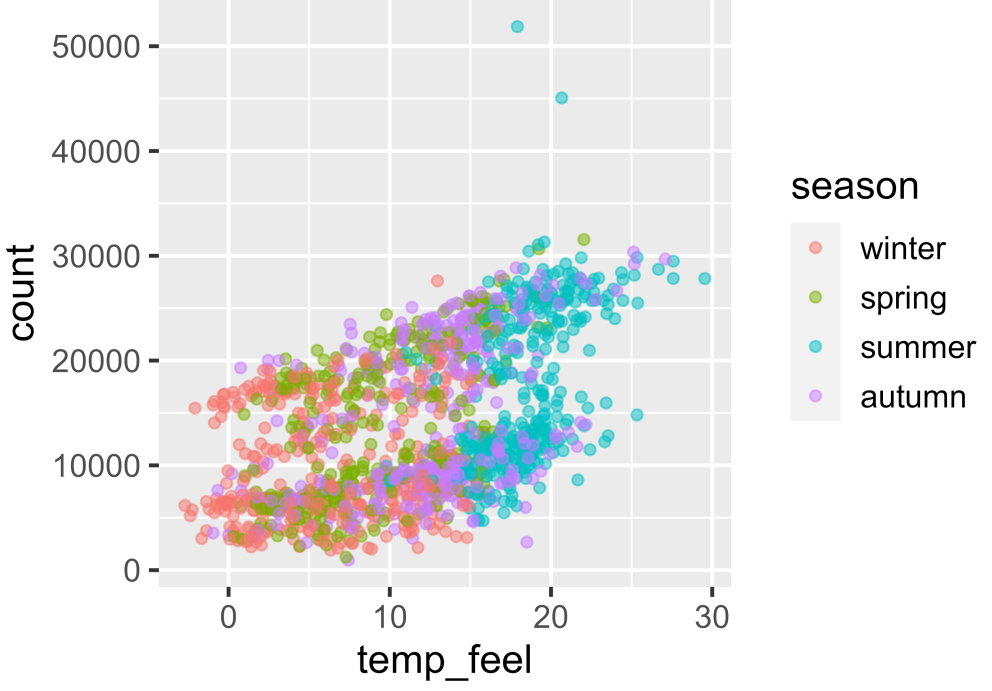


Setting vs Mapping of Visual Properties

```
1  ggplot(
2      bikes,
3      aes(x = temp_feel, y = count)
4    ) +
5    geom_point(
6      color = "#28a87d",
7      alpha = .5
8    )
```

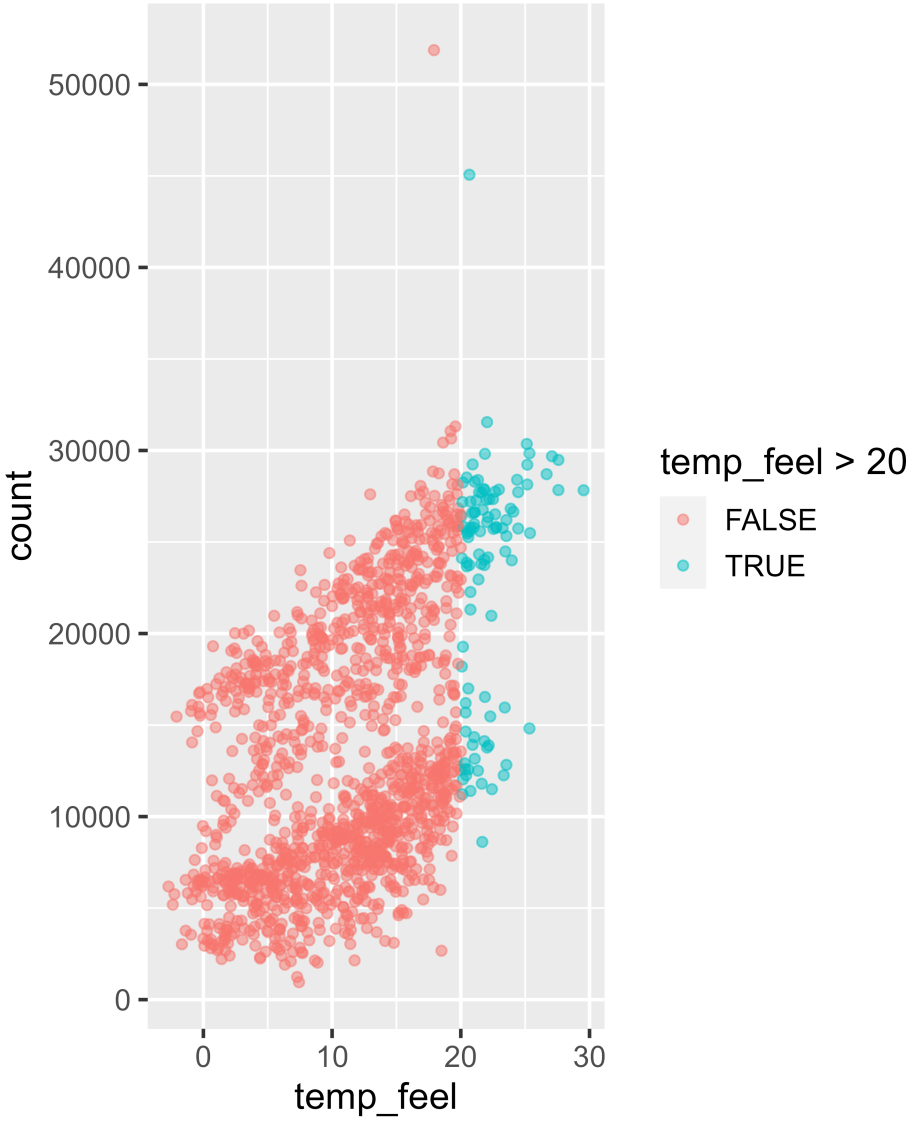



```
1  ggplot(
2      bikes,
3      aes(x = temp_feel, y = count)
4    ) +
5    geom_point(
6      aes(color = season),
7      alpha = .5
8    )
```



Mapping Expressions

```
1 ggplot(
2     bikes,
3     aes(x = temp_feel, y = count)
4   ) +
5   geom_point(
6     aes(color = temp_feel > 20),
7     alpha = .5
8   )
```



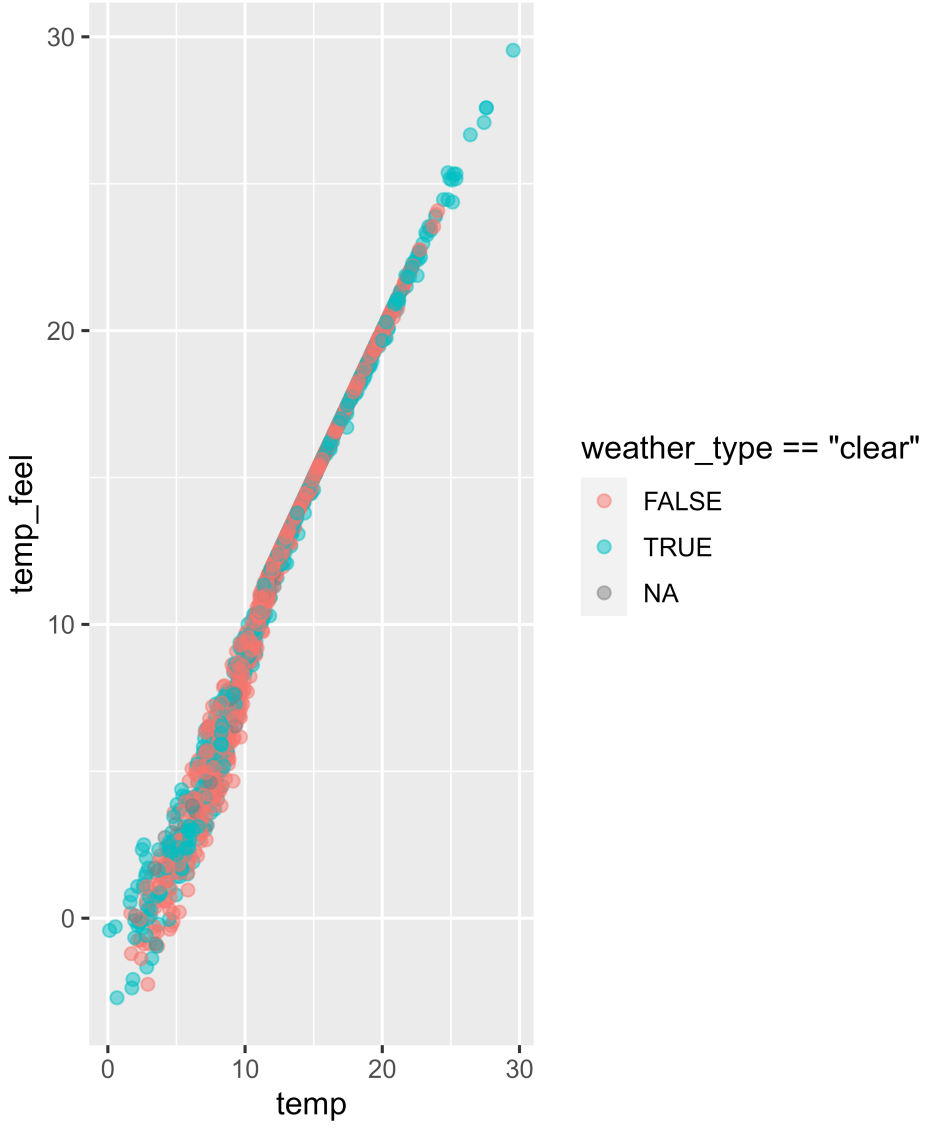
Your Turn!

- **Create a scatter plot of `temp_feel` vs `temp`.**
  - Map the color of the points to clear weather.
  - Map the size of the points to count.
  - Turn the points into diamonds.
  - Bonus: What do you notice in the legend? How could you fix it?

05:00

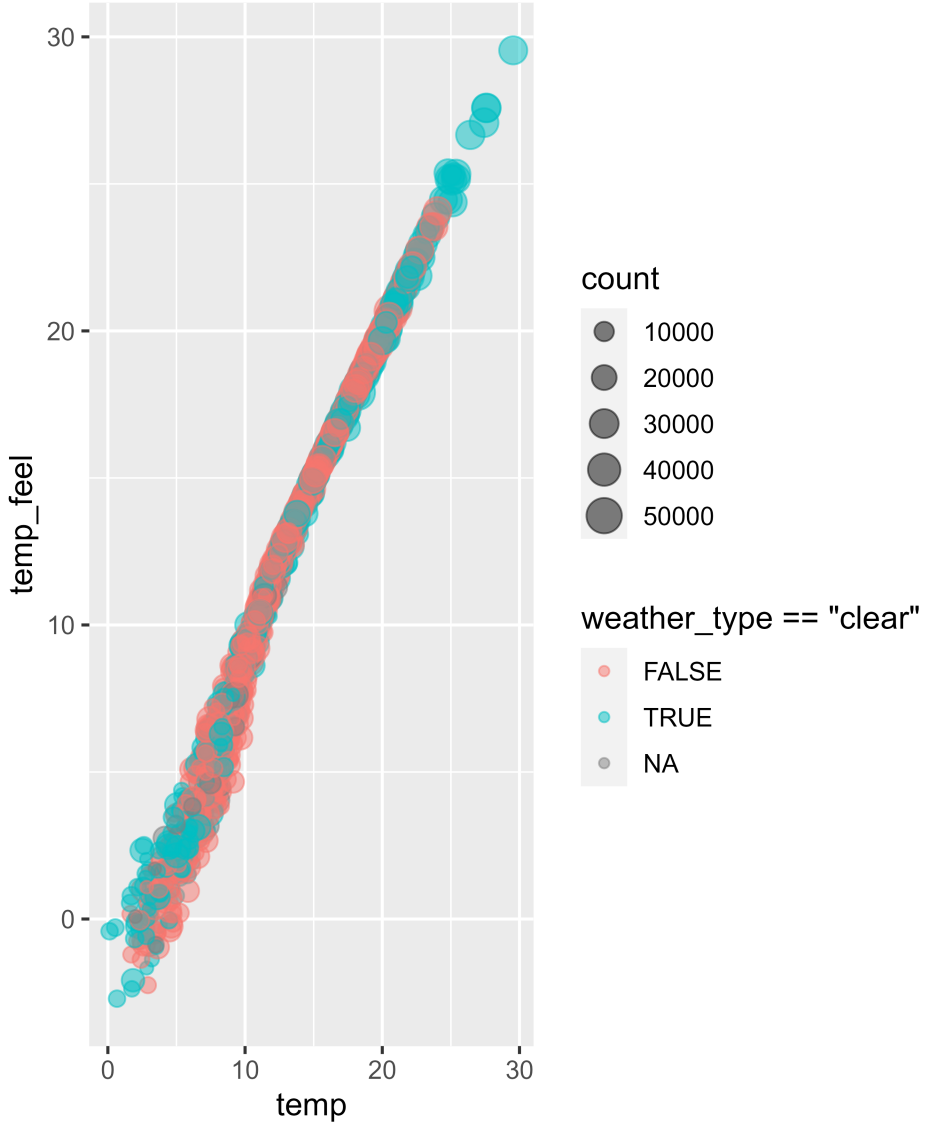
Mapping Expressions

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear"),
    alpha = .5,
    size = 2
  )
```



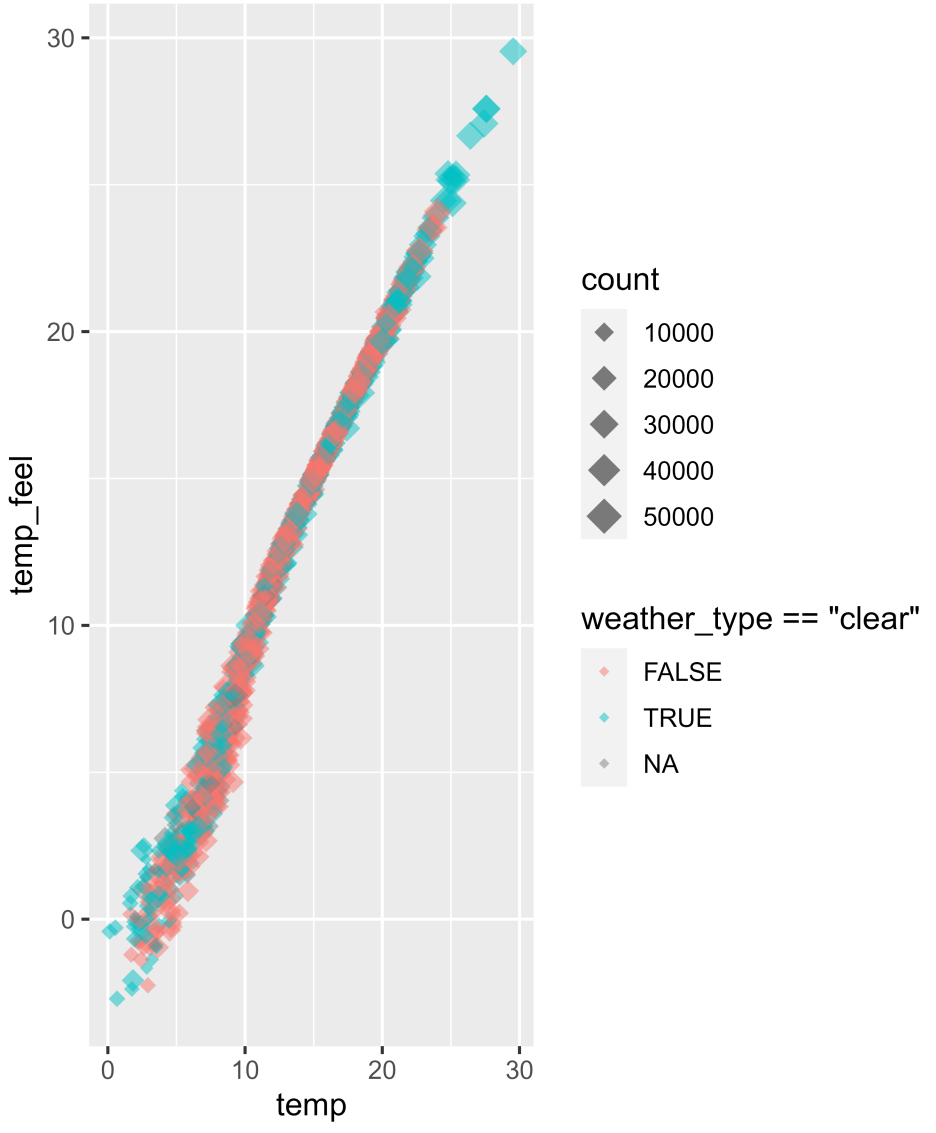
Mapping to Size

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    alpha = .5
  )
```



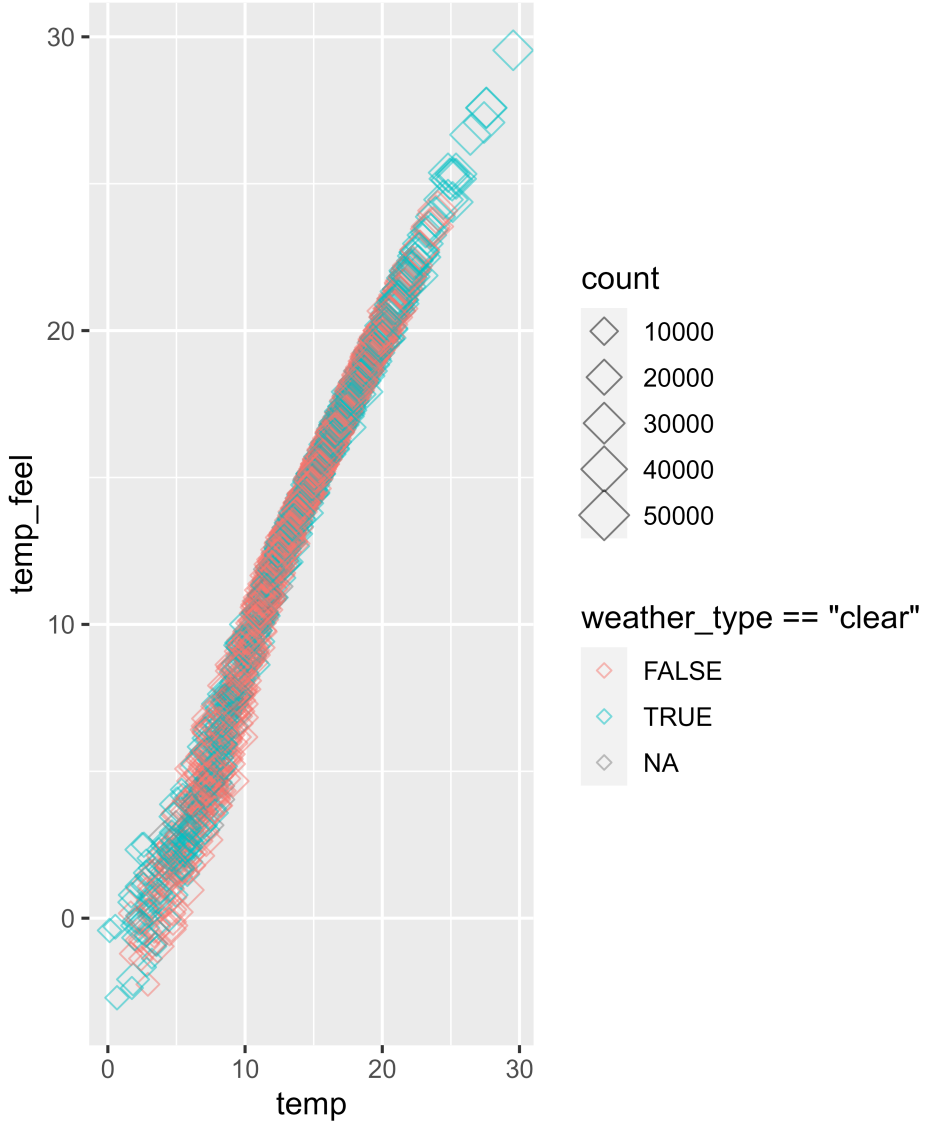
Setting a Constant Property

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    shape = 18,
    alpha = .5
  )
```



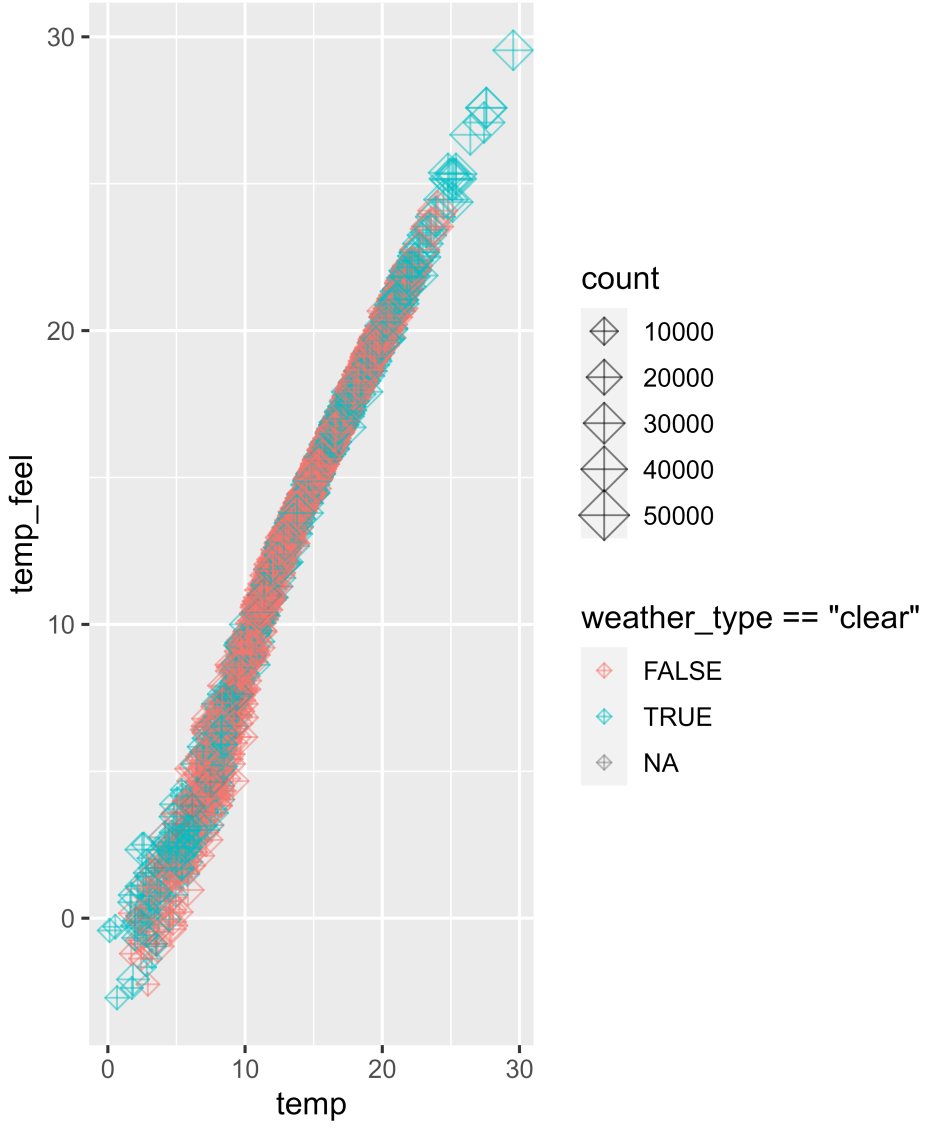
Setting a Constant Property

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    shape = 5,
    alpha = .5
  )
```



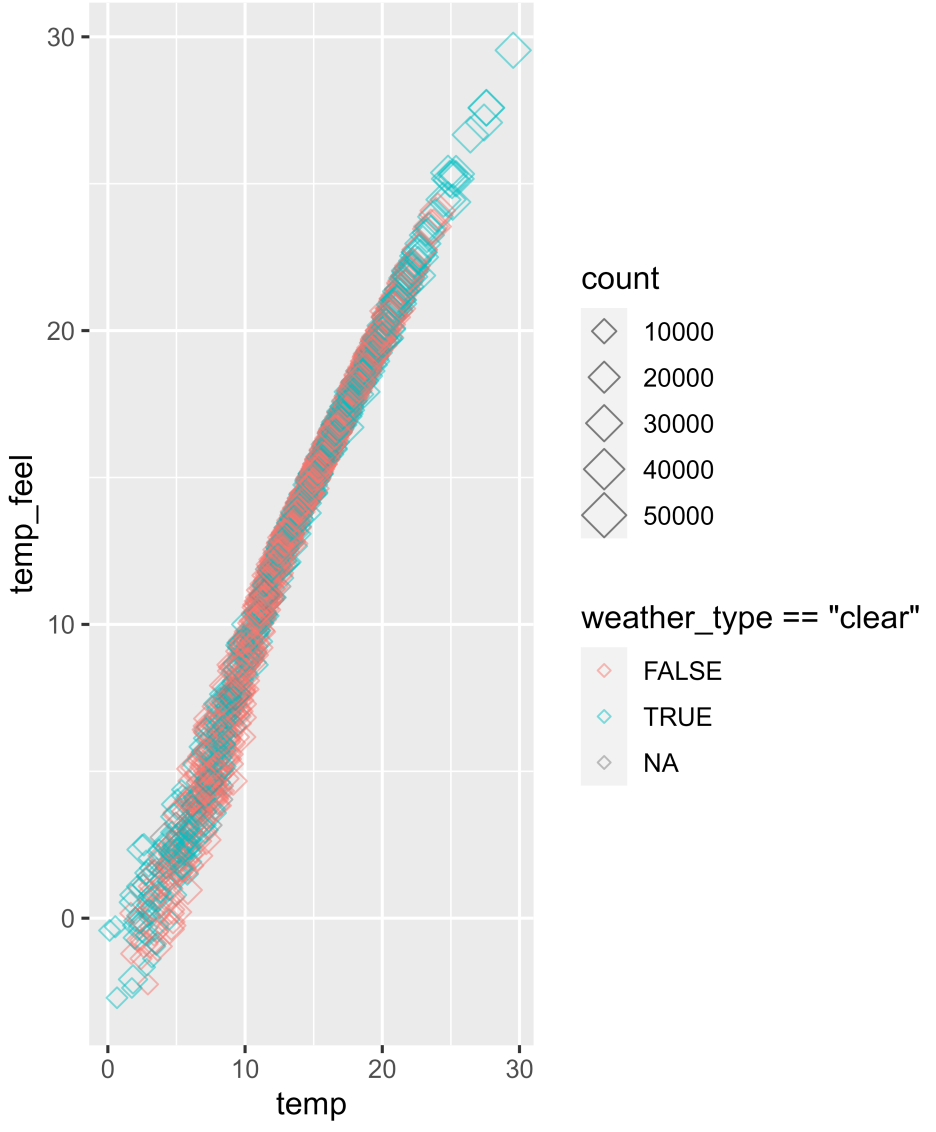
Setting a Constant Property

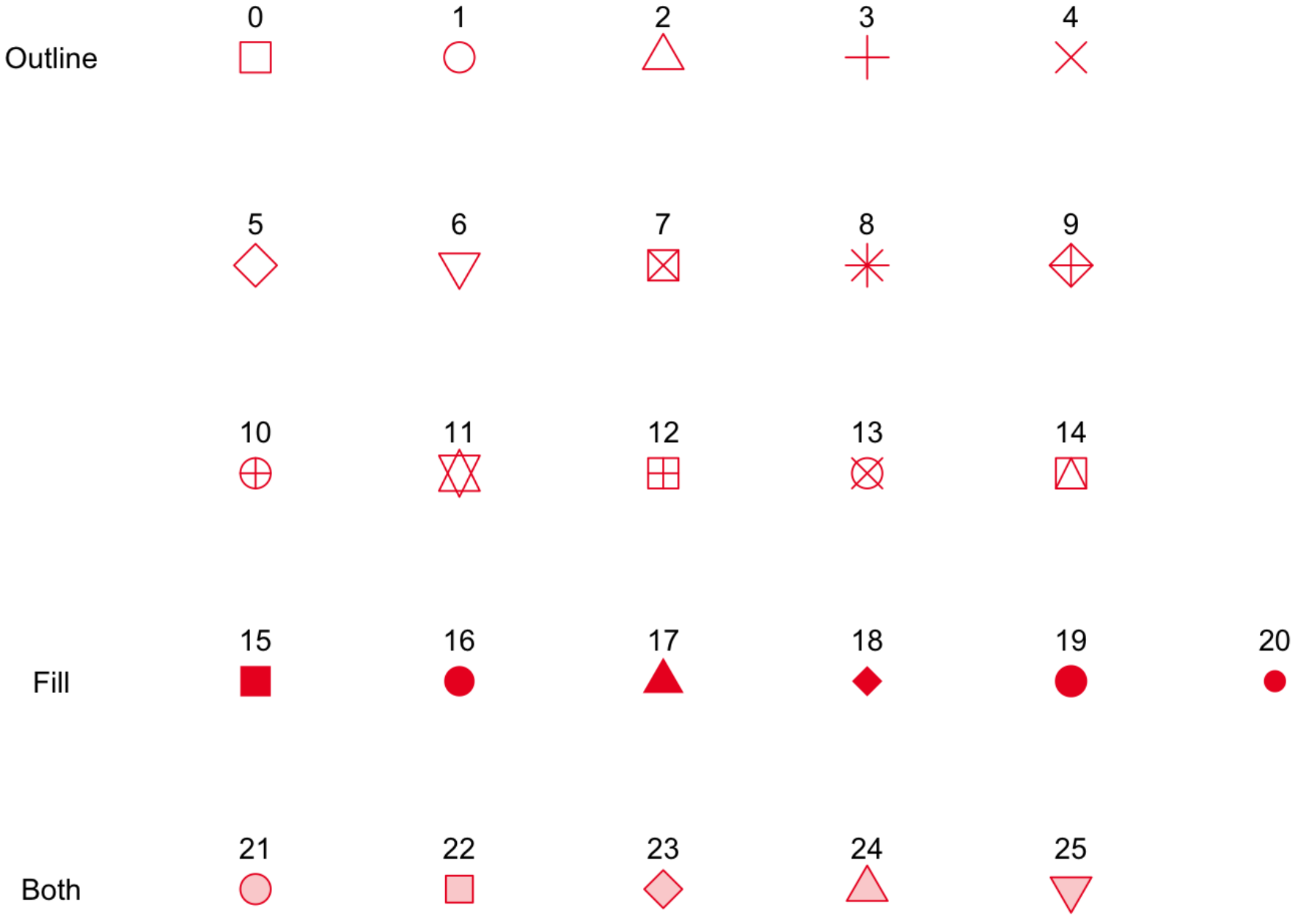
```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    shape = 9,
    alpha = .5
  )
```



Setting a Constant Property

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    shape = 23,
    alpha = .5
  )
```



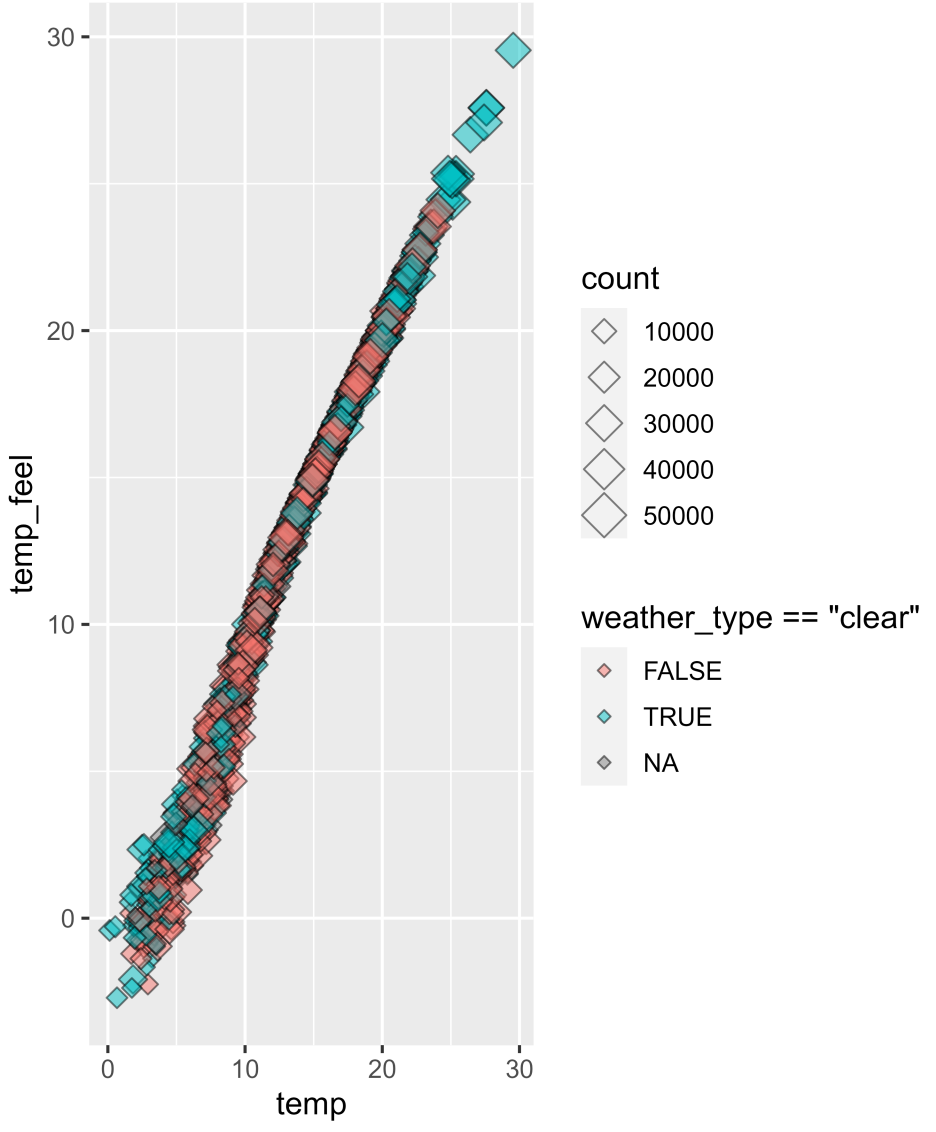


Source: Albert's Blog

Source: Albert's Blog

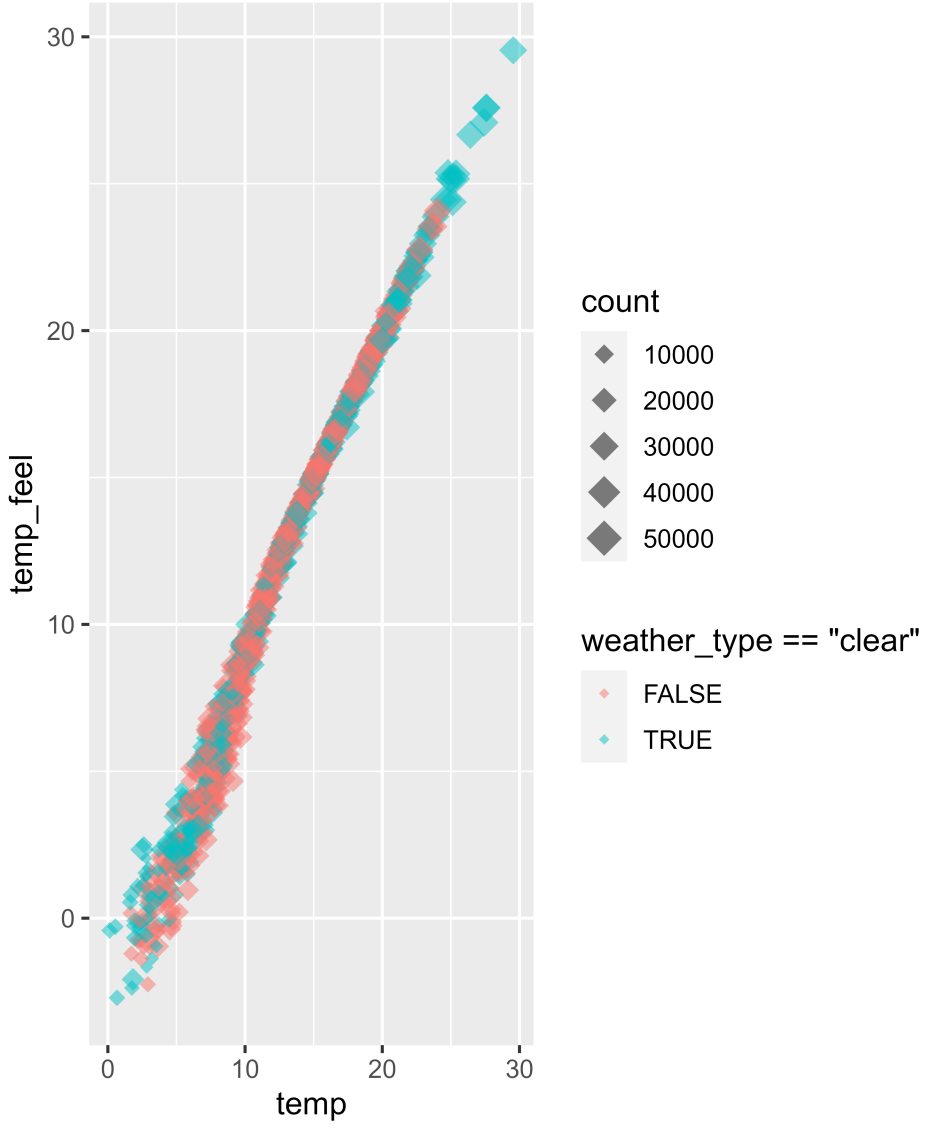
Setting a Constant Property

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(fill = weather_type == "clear",
        size = count),
    shape = 23,
    color = "black",
    alpha = .5
  )
```



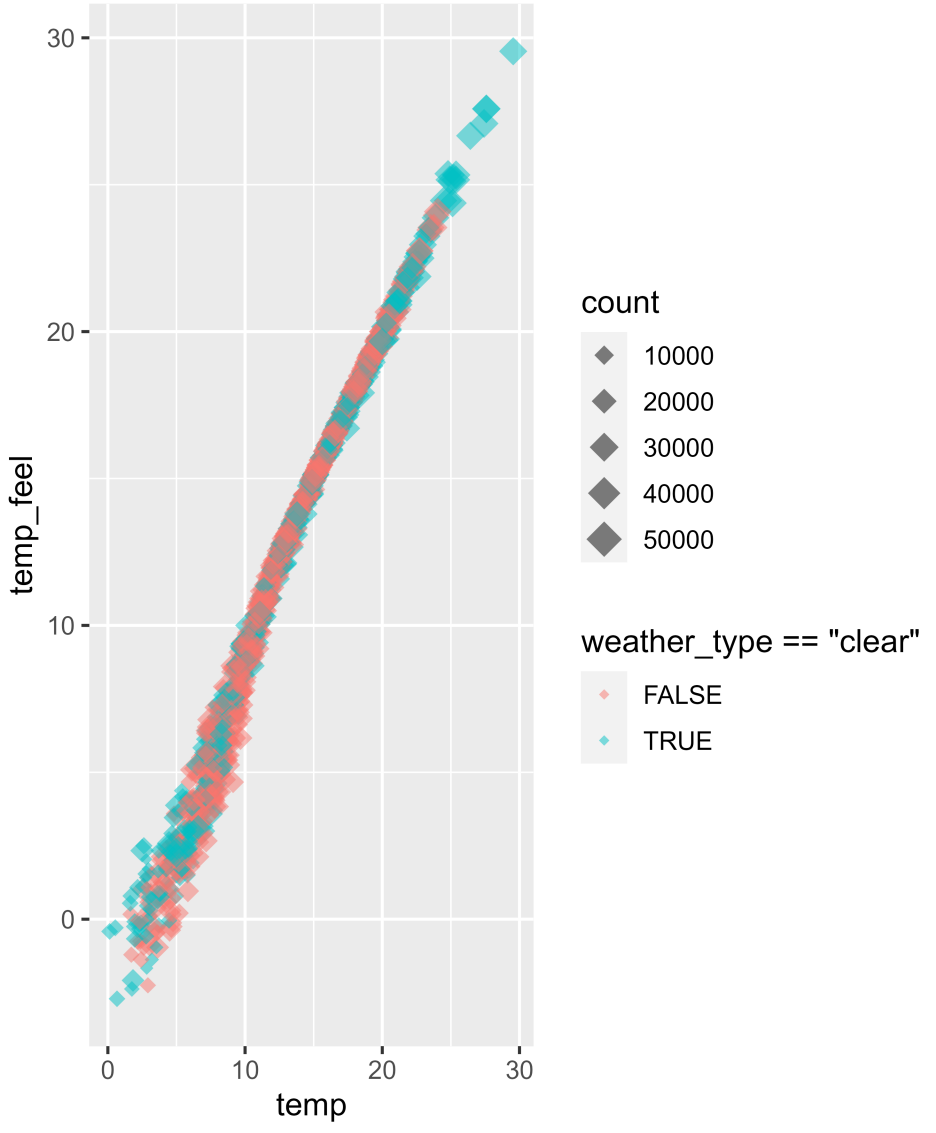
Filter Data

```
ggplot(
    filter(bikes, !is.na(weather_type)),
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    shape = 18,
    alpha = .5
  )
```



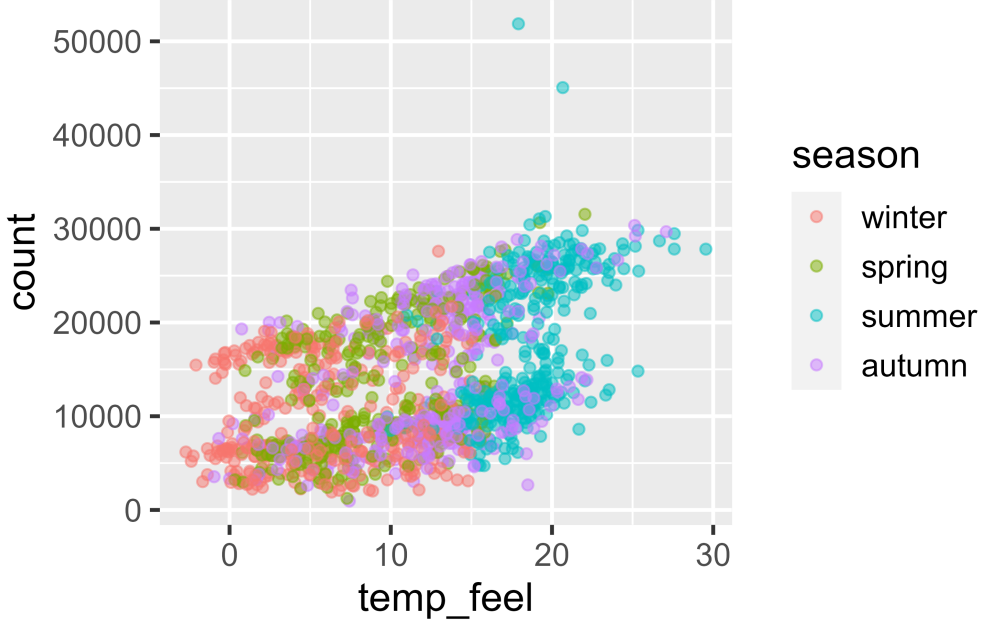
Filter Data

```
ggplot(
    bikes %>% filter(!is.na(weather_type)),
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "clear",
        size = count),
    shape = 18,
    alpha = .5
  )
```

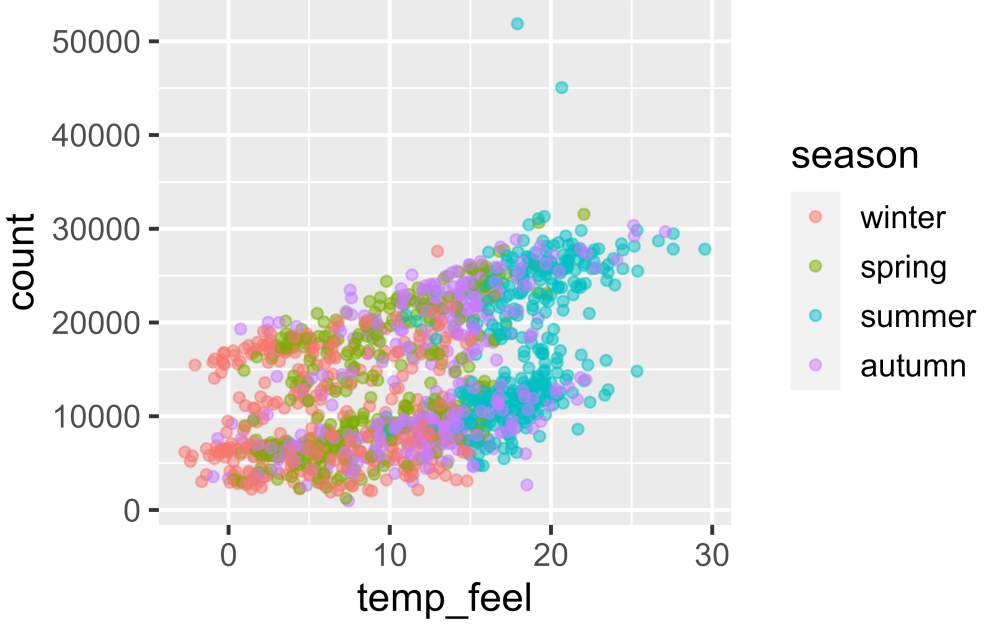


Local vs. Global Encoding

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    aes(color = season),
    alpha = .5
  )
```

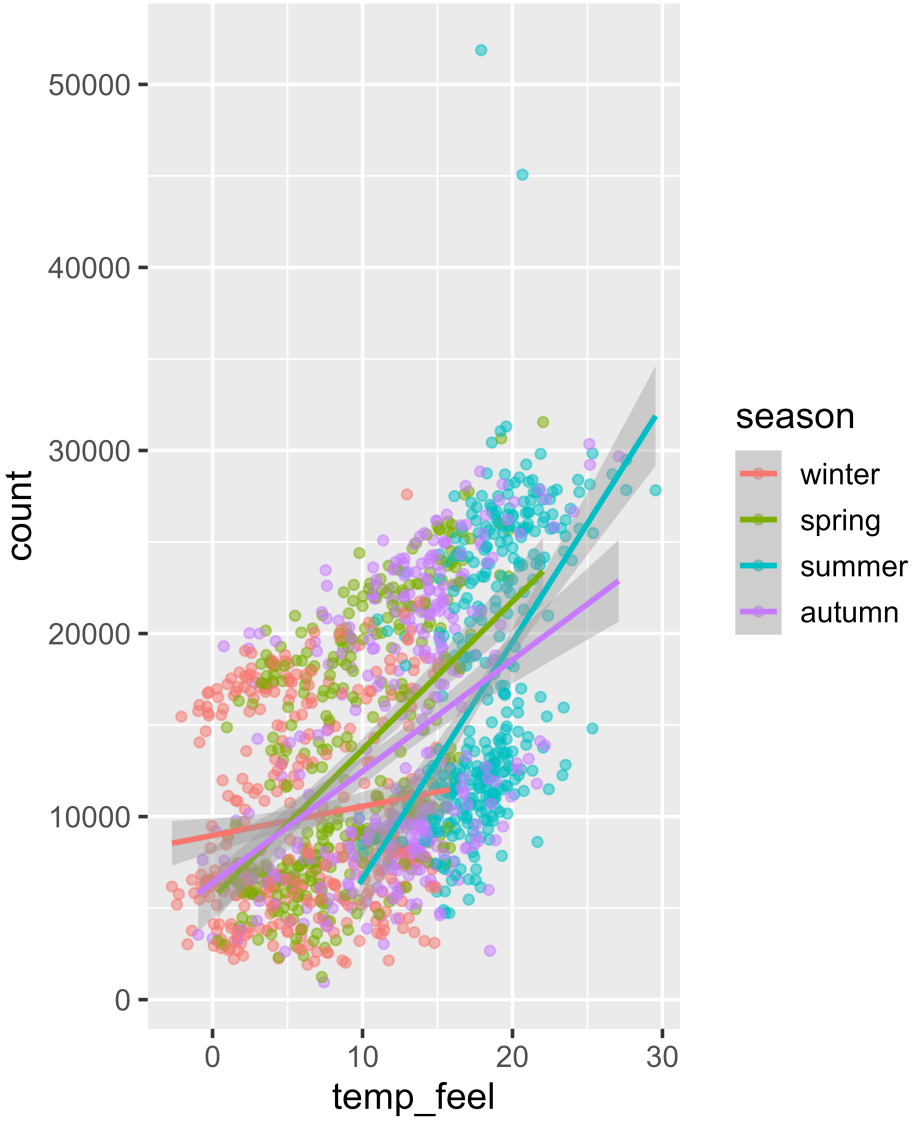



```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season)
  ) +
  geom_point(
    alpha = .5
  )
```



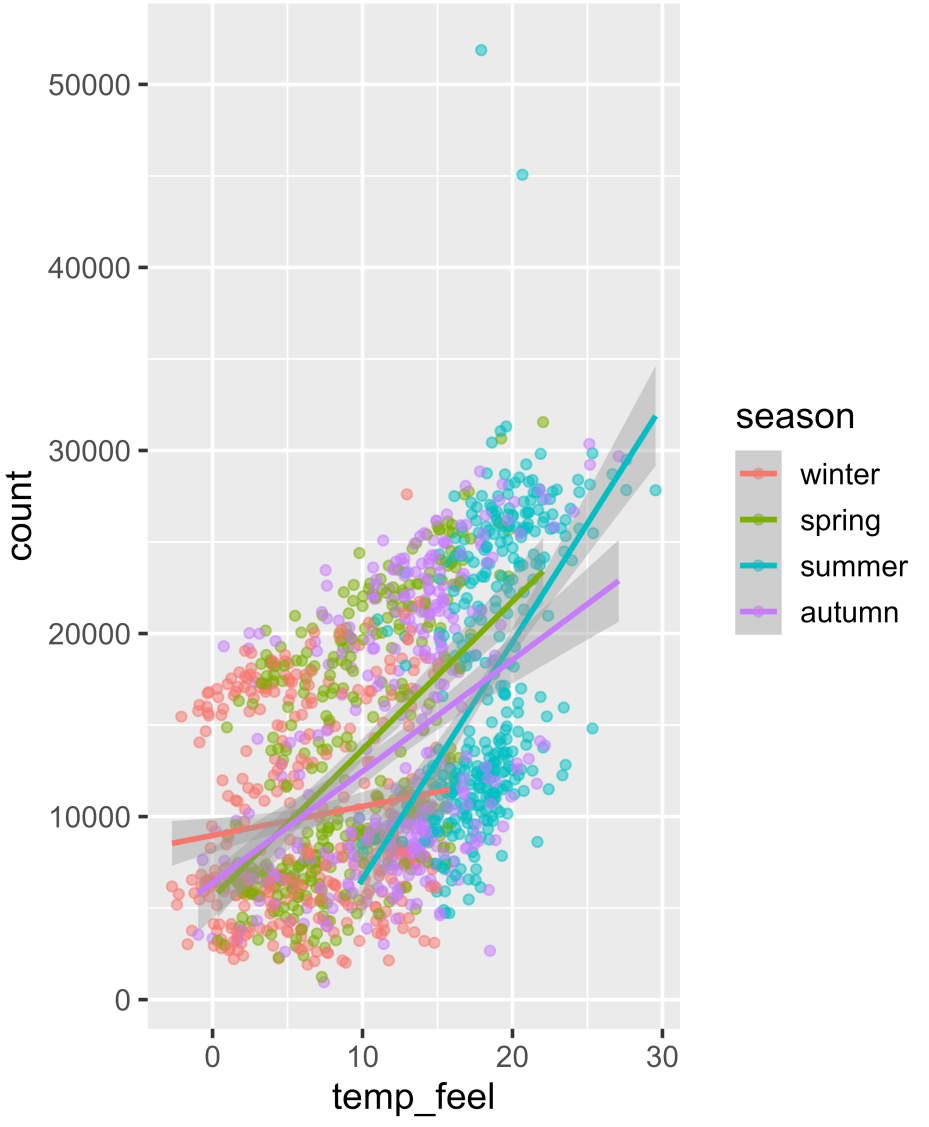
Adding More Layers

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm"
  )
```



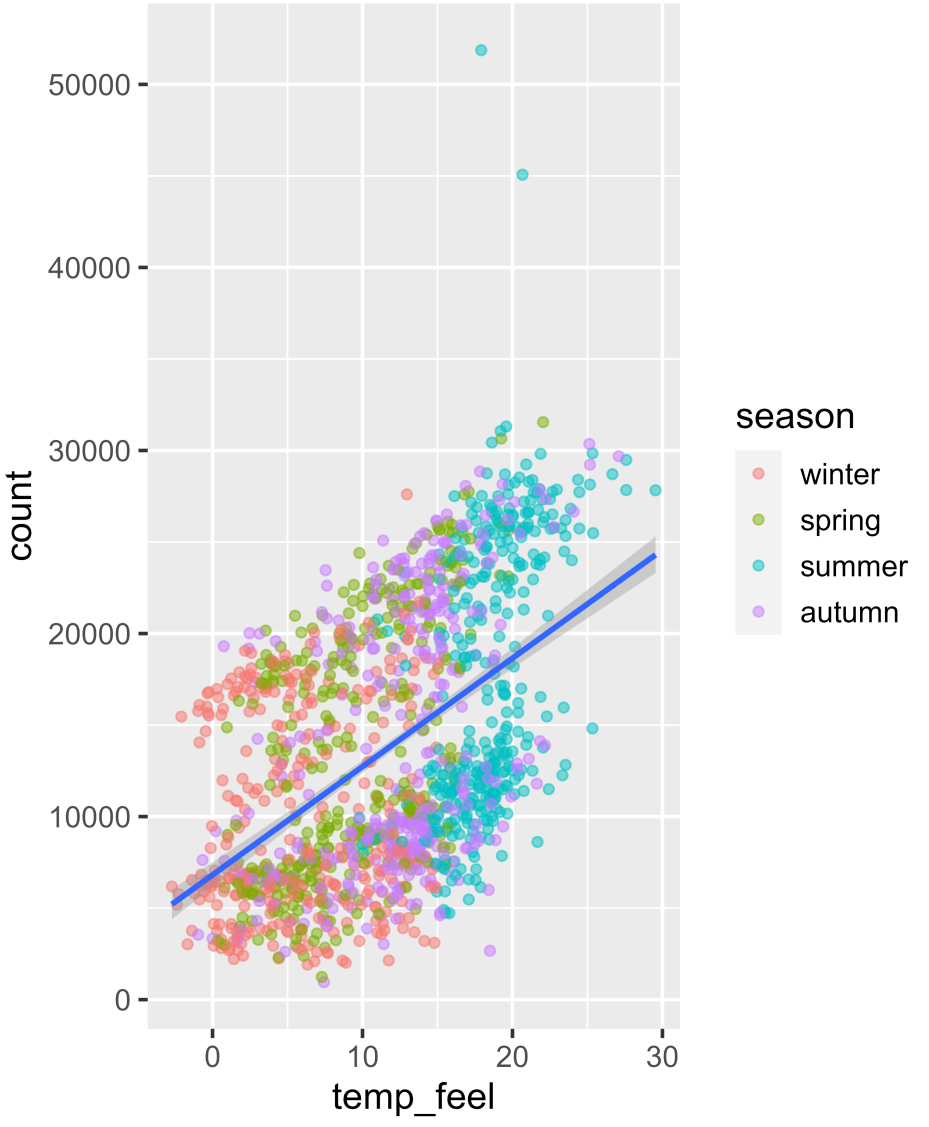
Global Color Encoding

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm"
  )
```



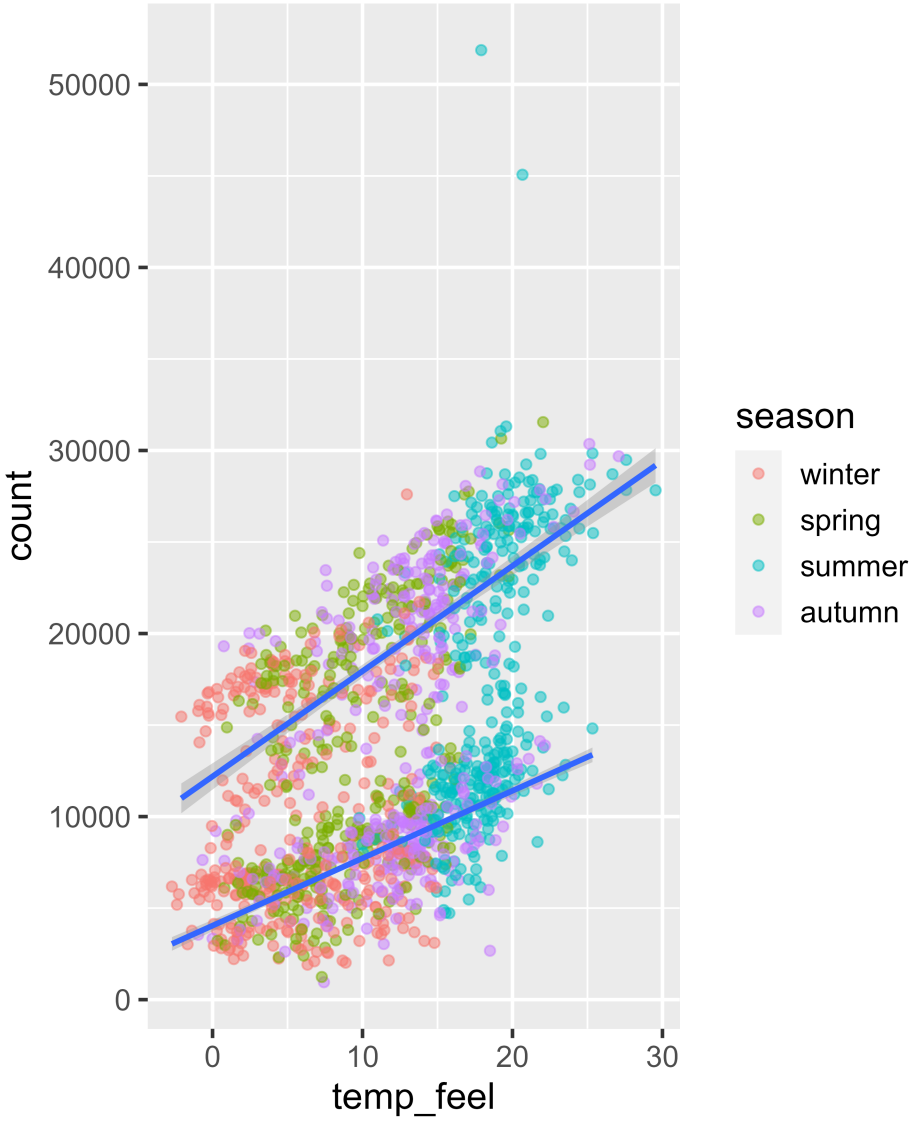
Local Color Encoding

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    aes(color = season),
    alpha = .5
  ) +
  geom_smooth(
    method = "lm"
  )
```



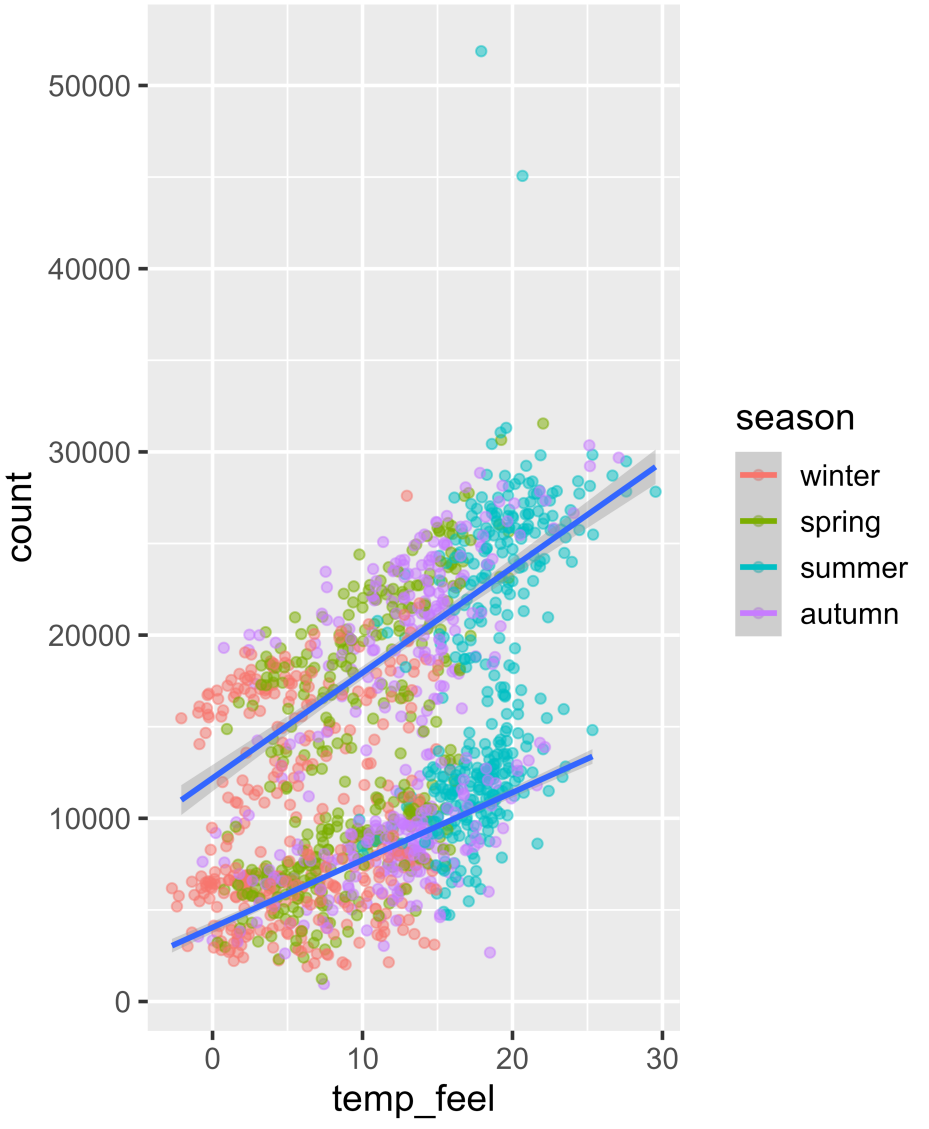
The `group` Aesthetic

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    aes(color = season),
    alpha = .5
  ) +
  geom_smooth(
    aes(group = day_night),
    method = "lm"
  )
```



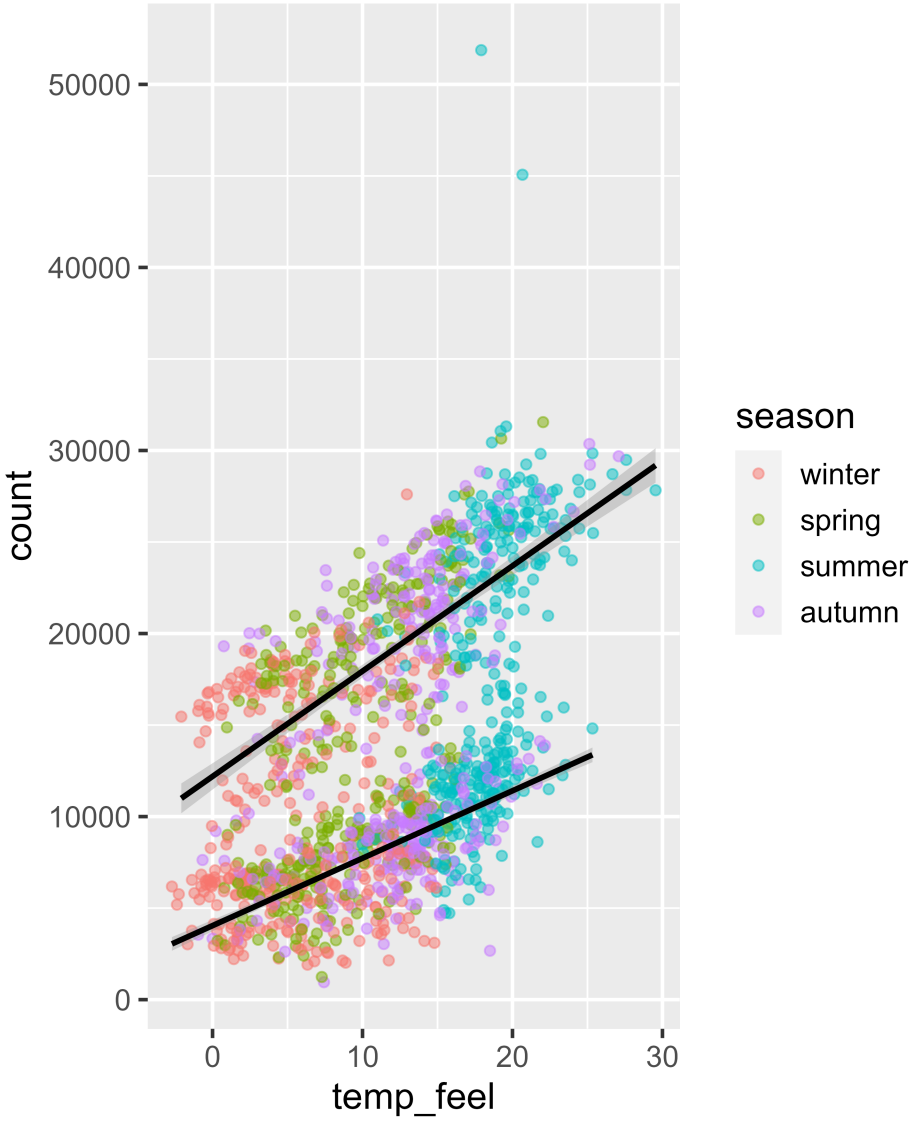
Set Both as Global Aesthetics

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season,
        group = day_night)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm"
  )
```



Overwrite Global Aesthetics

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season,
        group = day_night)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm",
    color = "black"
  )
```



Extending a ggplot

Cédric Scherer // rstudio::conf // July 2022

Store a ggplot as Object

```
g <-
  ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season,
        group = day_night)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm",
    color = "black"
  )

class(g)
```

```
[1] "gg"     "ggplot"
```

Inspect a ggplot Object

```
1 g$data
```

```
# A tibble: 1,454 x 14
   date       day_night year  month season count is_workday is_weekend
   <date>     <chr>     <fct> <fct> <fct>  <int> <lgl>      <lgl>
 1 2015-01-04 day       2015  1     winter  6830 FALSE      TRUE
 2 2015-01-04 night     2015  1     winter  2404 FALSE      TRUE
 3 2015-01-05 day       2015  1     winter 14763 TRUE       FALSE
 4 2015-01-05 night     2015  1     winter  5609 TRUE       FALSE
 5 2015-01-06 day       2015  1     winter 14501 TRUE       FALSE
 6 2015-01-06 night     2015  1     winter  6112 TRUE       FALSE
 7 2015-01-07 day       2015  1     winter 16358 TRUE       FALSE
 8 2015-01-07 night     2015  1     winter  4706 TRUE       FALSE
 9 2015-01-08 day       2015  1     winter  9971 TRUE       FALSE
10 2015-01-08 night     2015  1     winter  5630 TRUE       FALSE
# ... with 1,444 more rows, and 6 more variables: is_holiday <lgl>, temp
<dbl>,
```

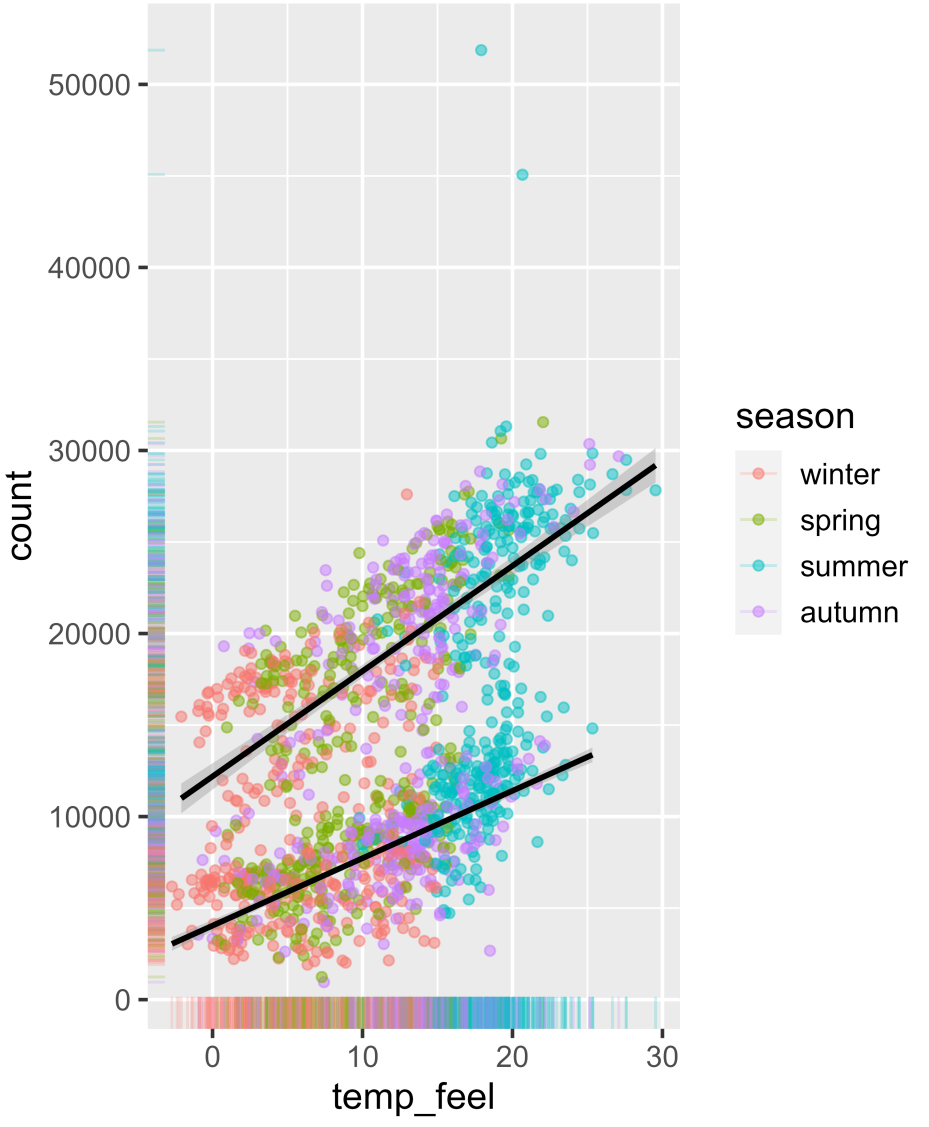
Inspect a ggplot Object

```
1 g$mapping
```

```
Aesthetic mapping:
* `x`      -> `temp_feel`
* `y`      -> `count`
* `colour` -> `season`
* `group`  -> `day_night`
```

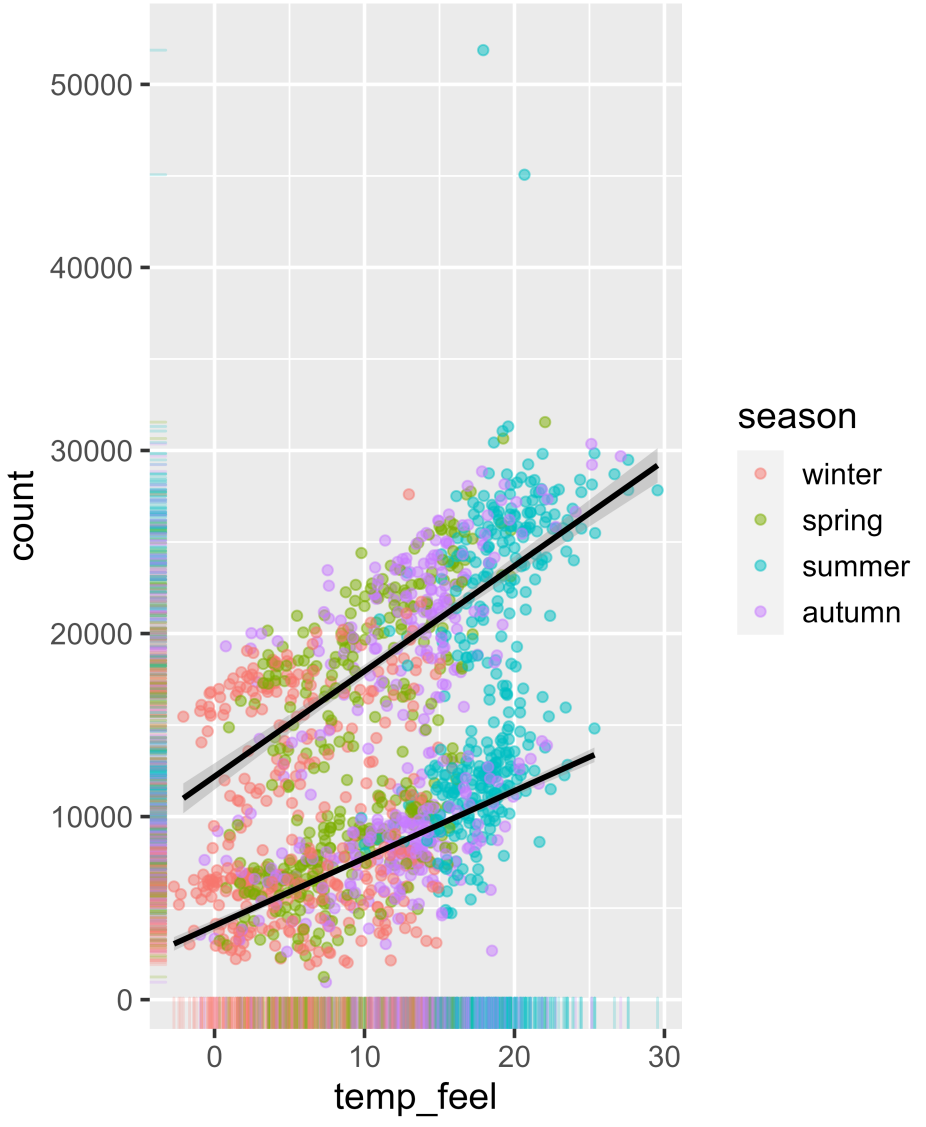
Extend a ggplot Object: Add Layers

```
g +
  geom_rug(
    alpha = .2
  )
```



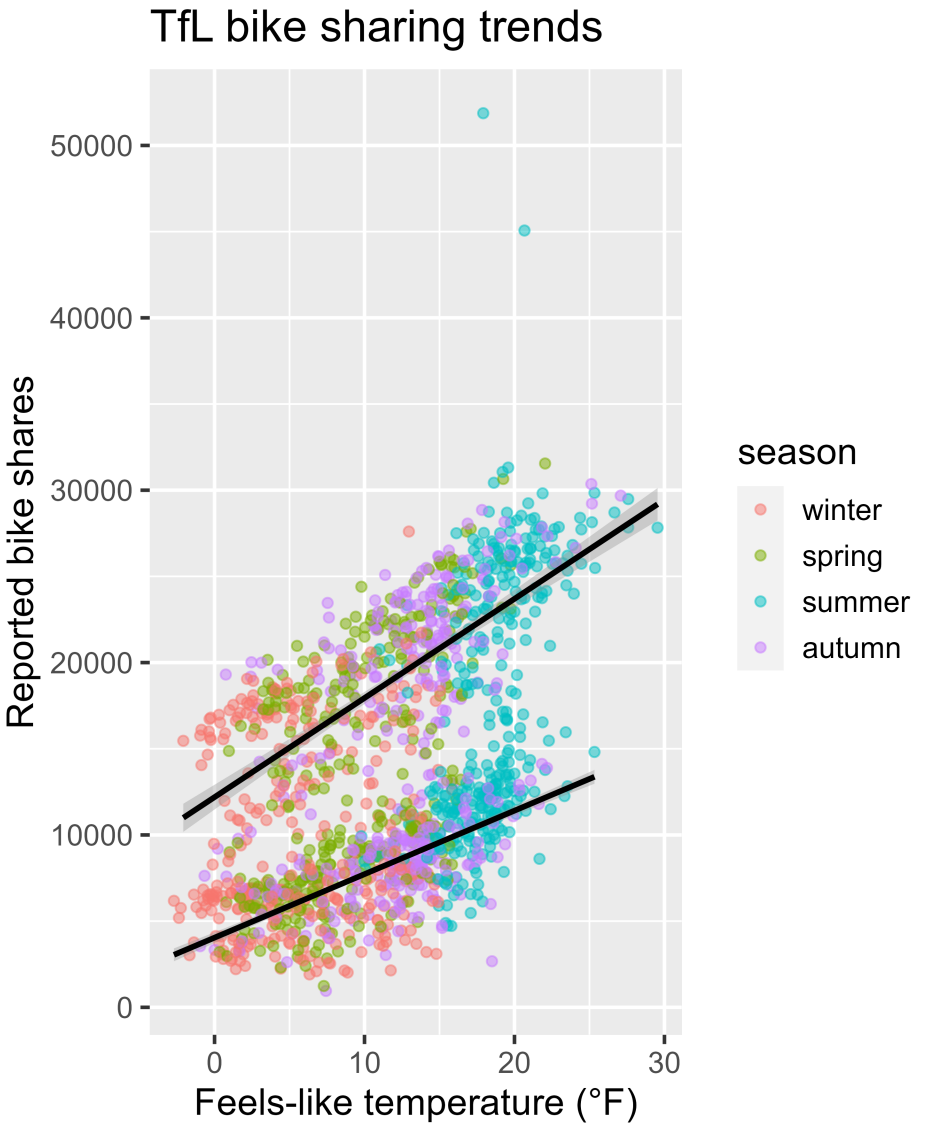
Remove a Layer from the Legend

```
g +
  geom_rug(
    alpha = .2,
    show.legend = FALSE
  )
```



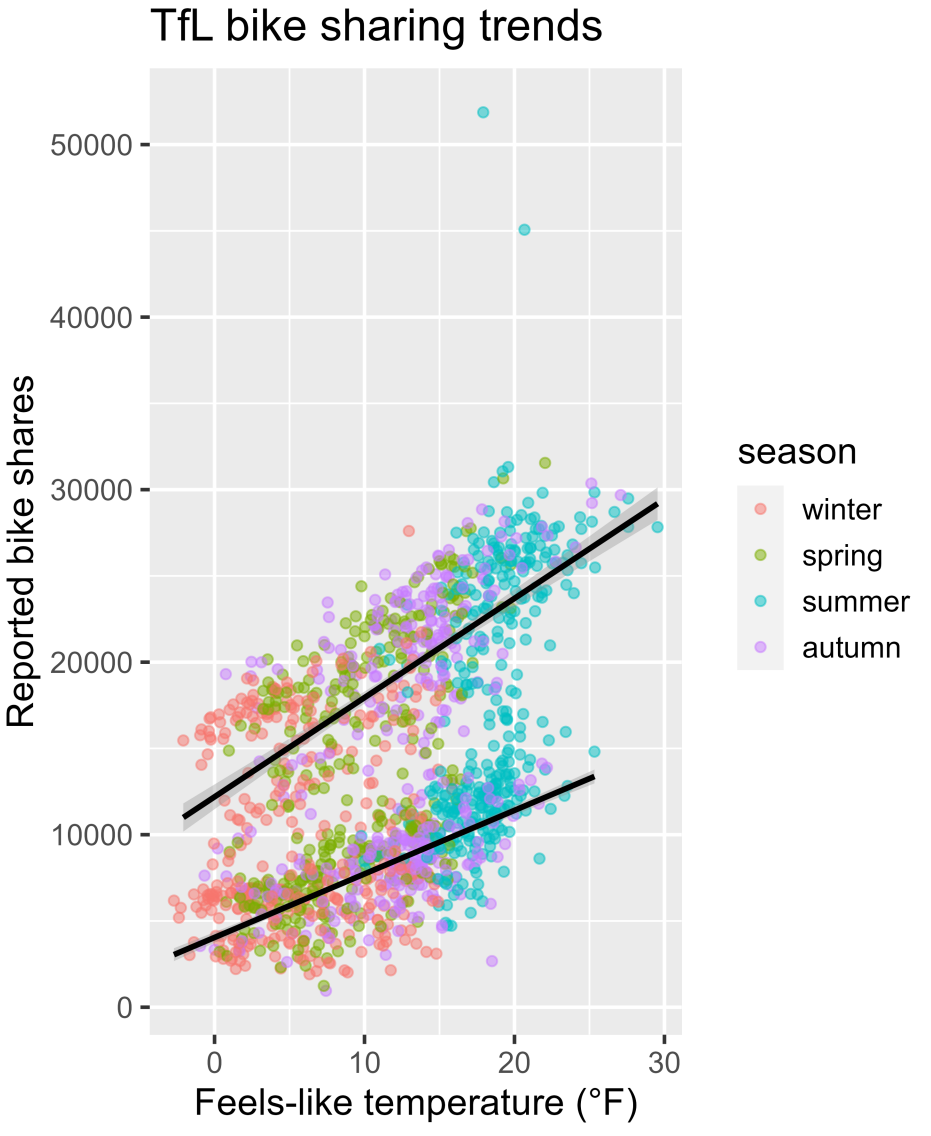
Extend a ggplot Object: Add Labels

```
g +
  xlab("Feels-like temperature (ºF)") +
  ylab("Reported bike shares") +
  ggtitle("TfL bike sharing trends")
```



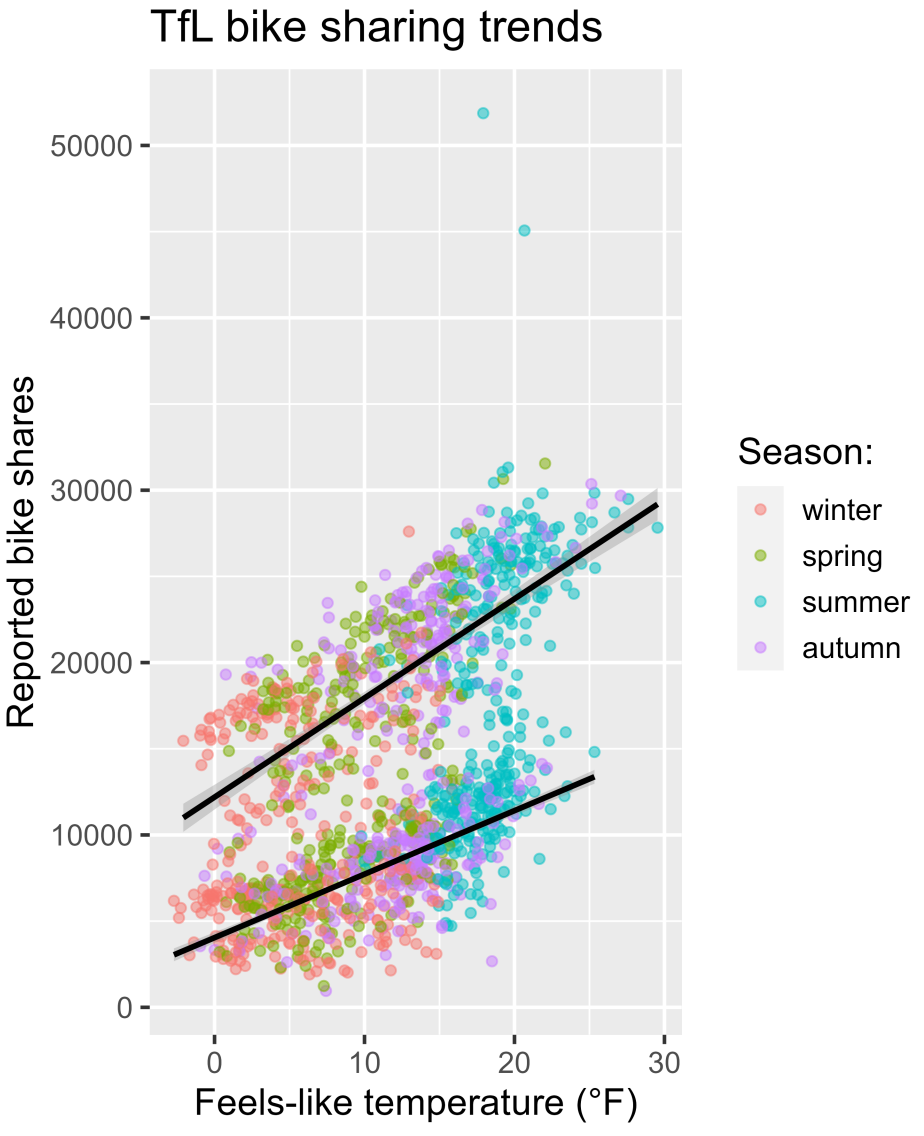
Extend a ggplot Object: Add Labels

```
g +
  labs(
    x = "Feels-like temperature (°F)",
    y = "Reported bike shares",
    title = "TfL bike sharing trends"
  )
```



Extend a ggplot Object: Add Labels

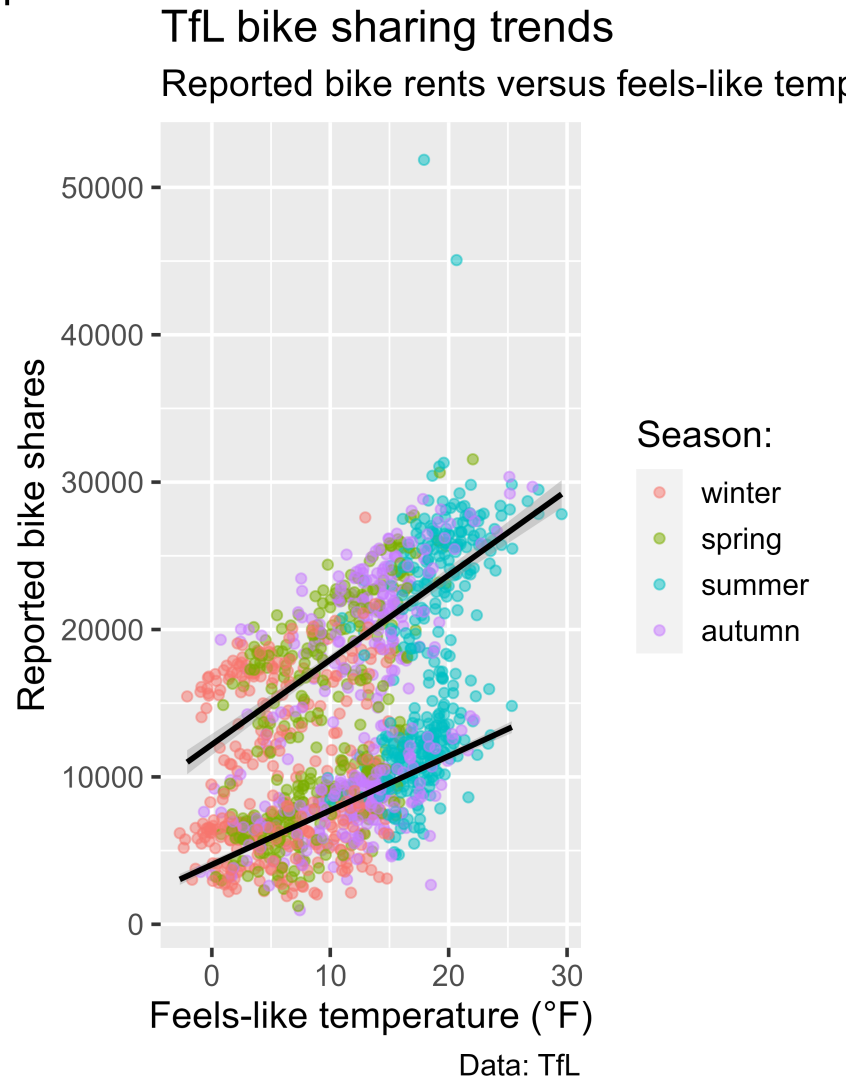
```
g <- g +
  labs(
    x = "Feels-like temperature (°F)",
    y = "Reported bike shares",
    title = "TfL bike sharing trends",
    color = "Season:"
  )

g
```



Extend a ggplot Object: Add Labels

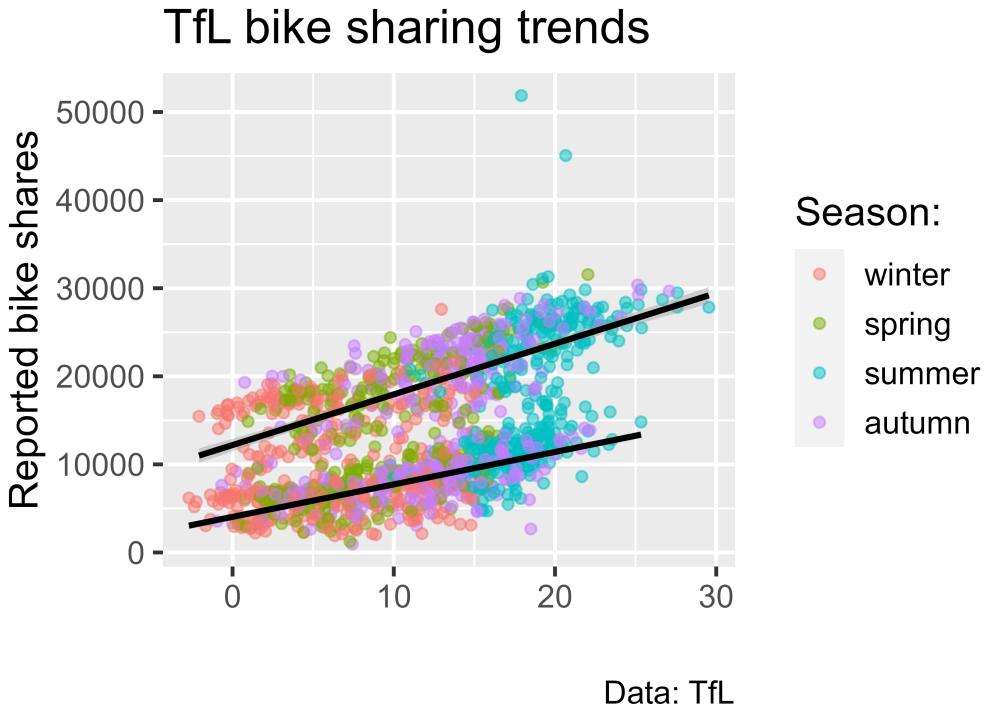
```
g +
  labs(
    x = "Feels-like temperature (°F)",
    y = "Reported bike shares",
    title = "TfL bike sharing trends",
    subtitle = "Reported bike rents versus fee
    caption = "Data: TfL",
    color = "Season:",
    tag = "Fig. 1"
  )
```

Fig. 1

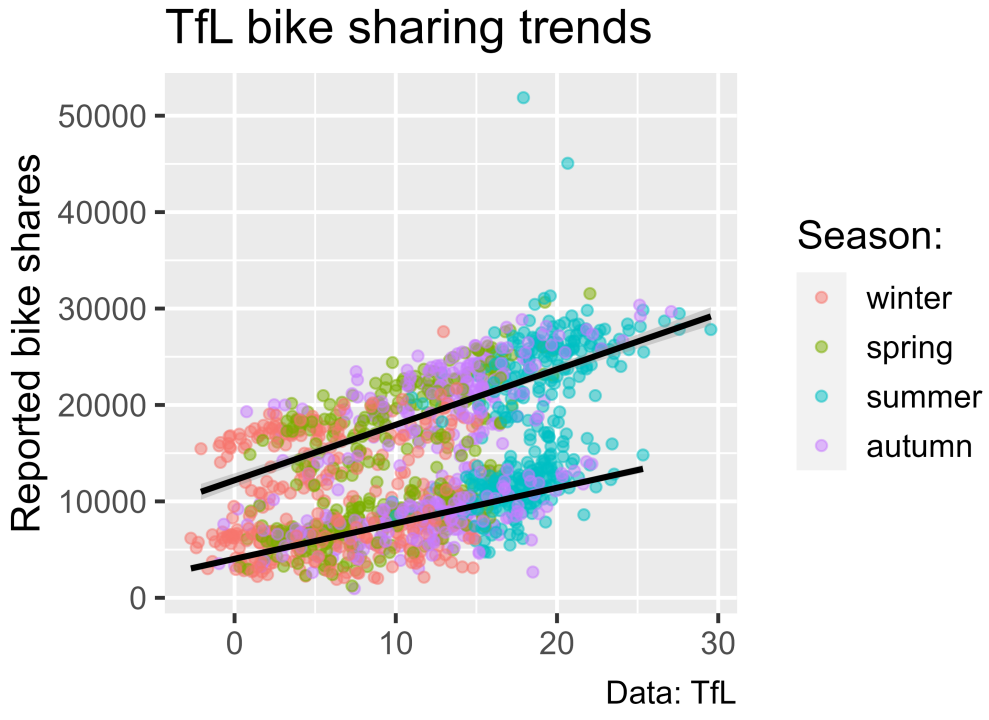


Extend a ggplot Object: Add Labels

```
g +
  labs(
    x = "",
    caption = "Data: TfL"
  )
```




```
g +
  labs(
    x = NULL,
    caption = "Data: TfL"
  )
```

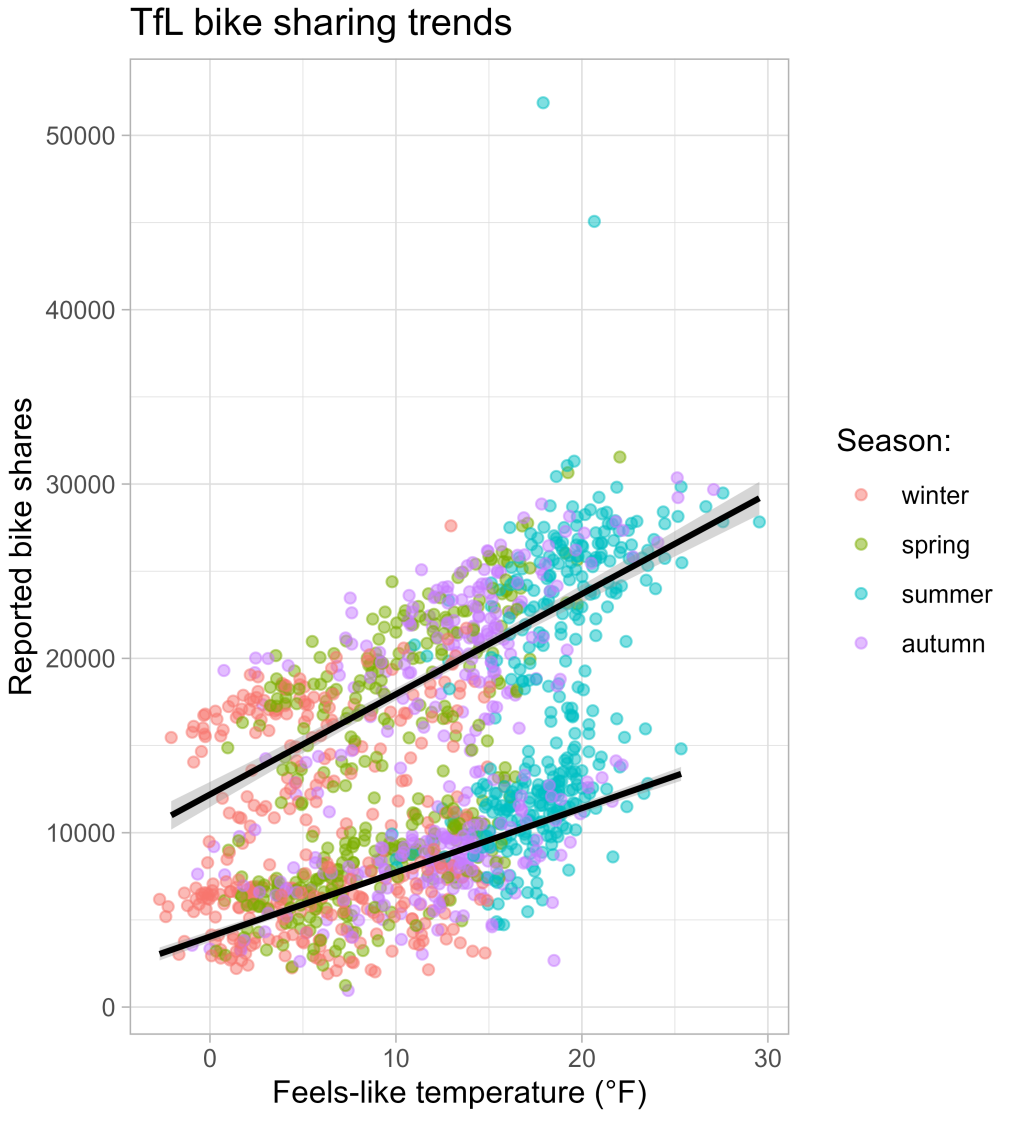


A Polished ggplot Example

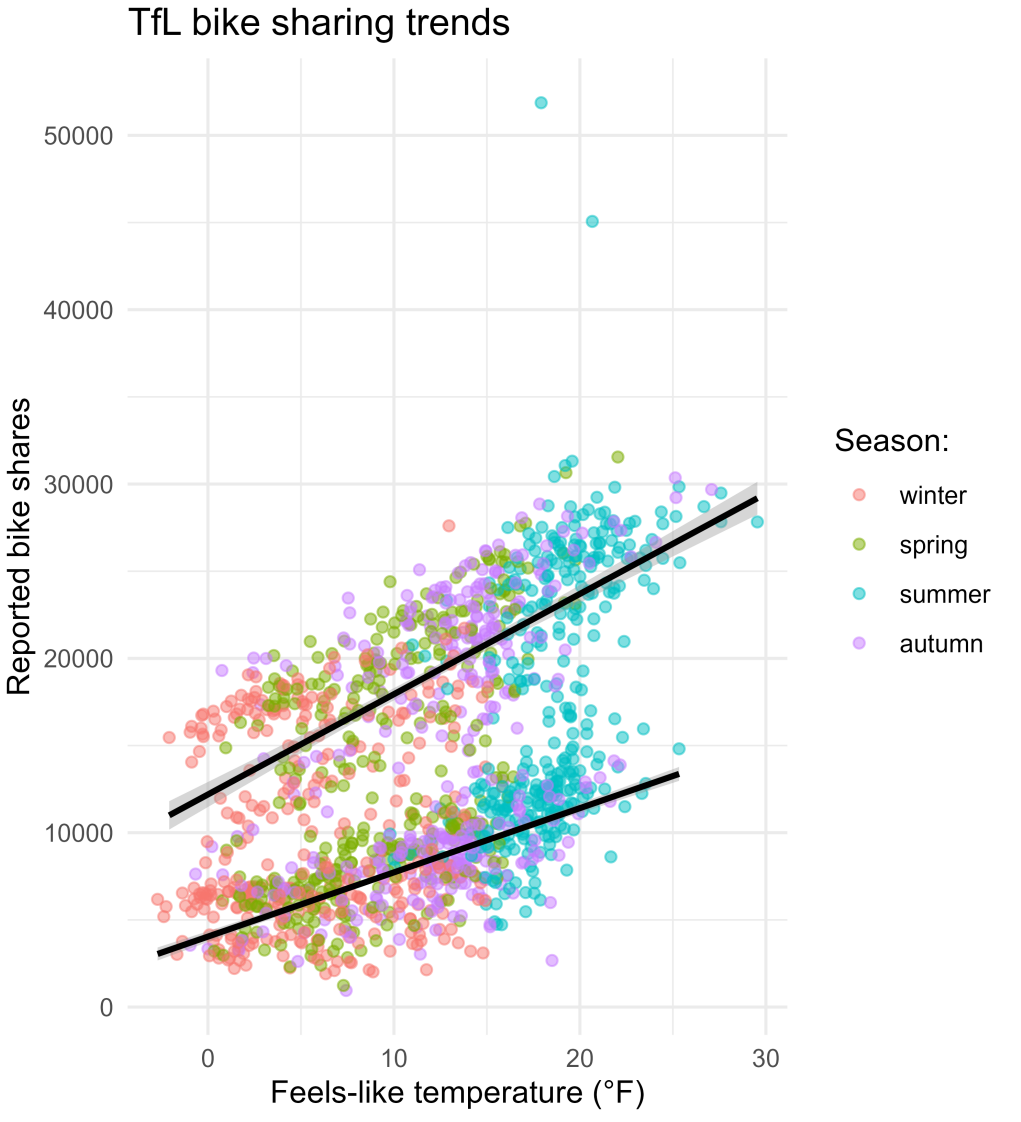
Cédric Scherer // rstudio::conf // July 2022

Extend a ggplot Object: Themes

```
1  g + theme_light()
```

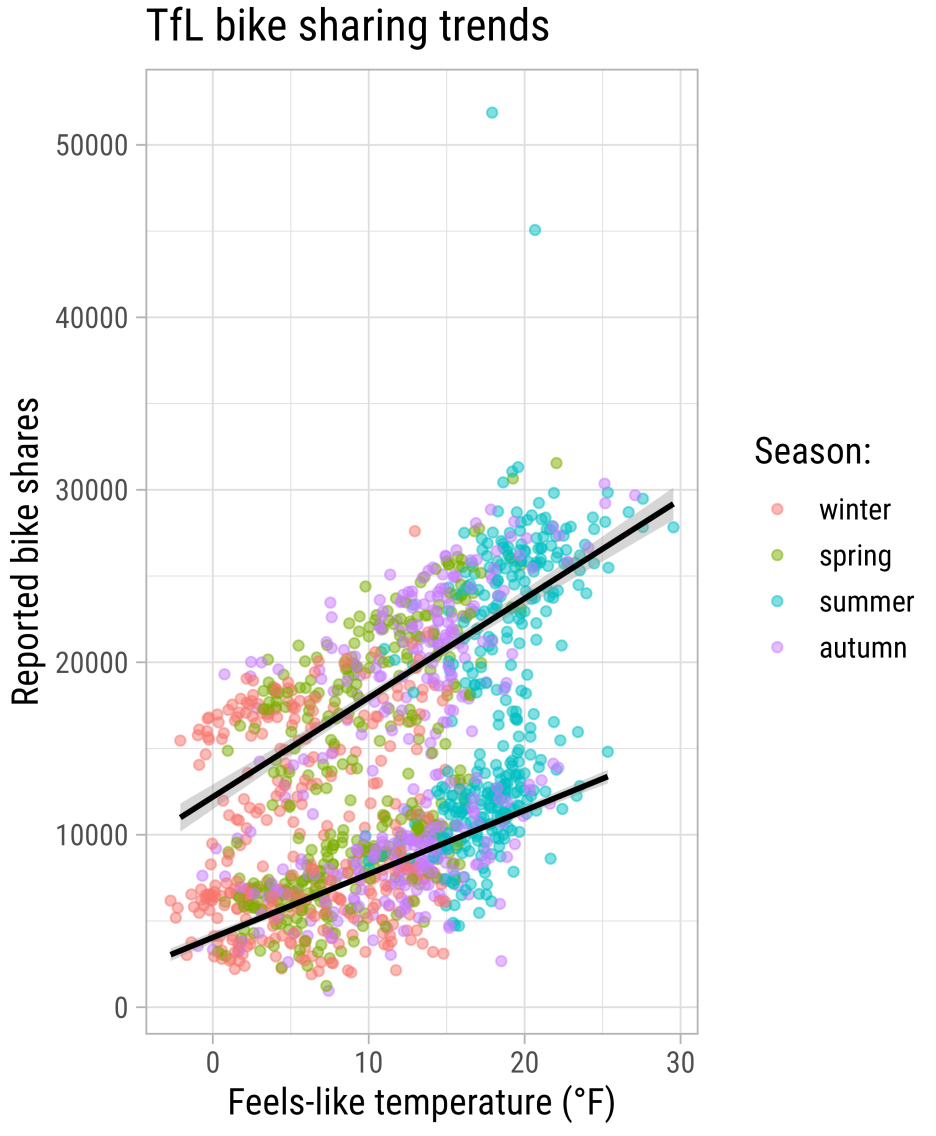



```
1  g + theme_minimal()
```



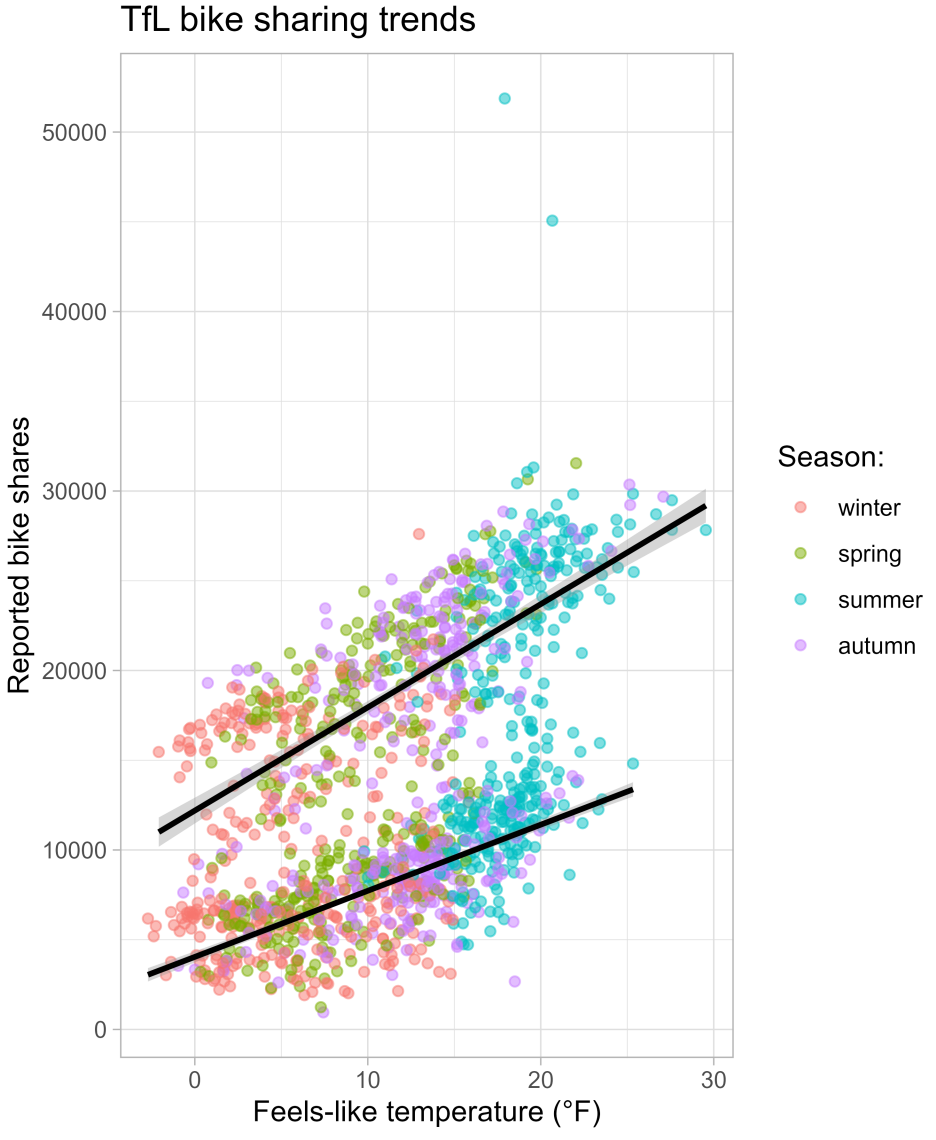
Change the Theme Base Settings

```
1 g + theme_light(
2   base_size = 14,
3   base_family = "Roboto Condensed"
4 )
```



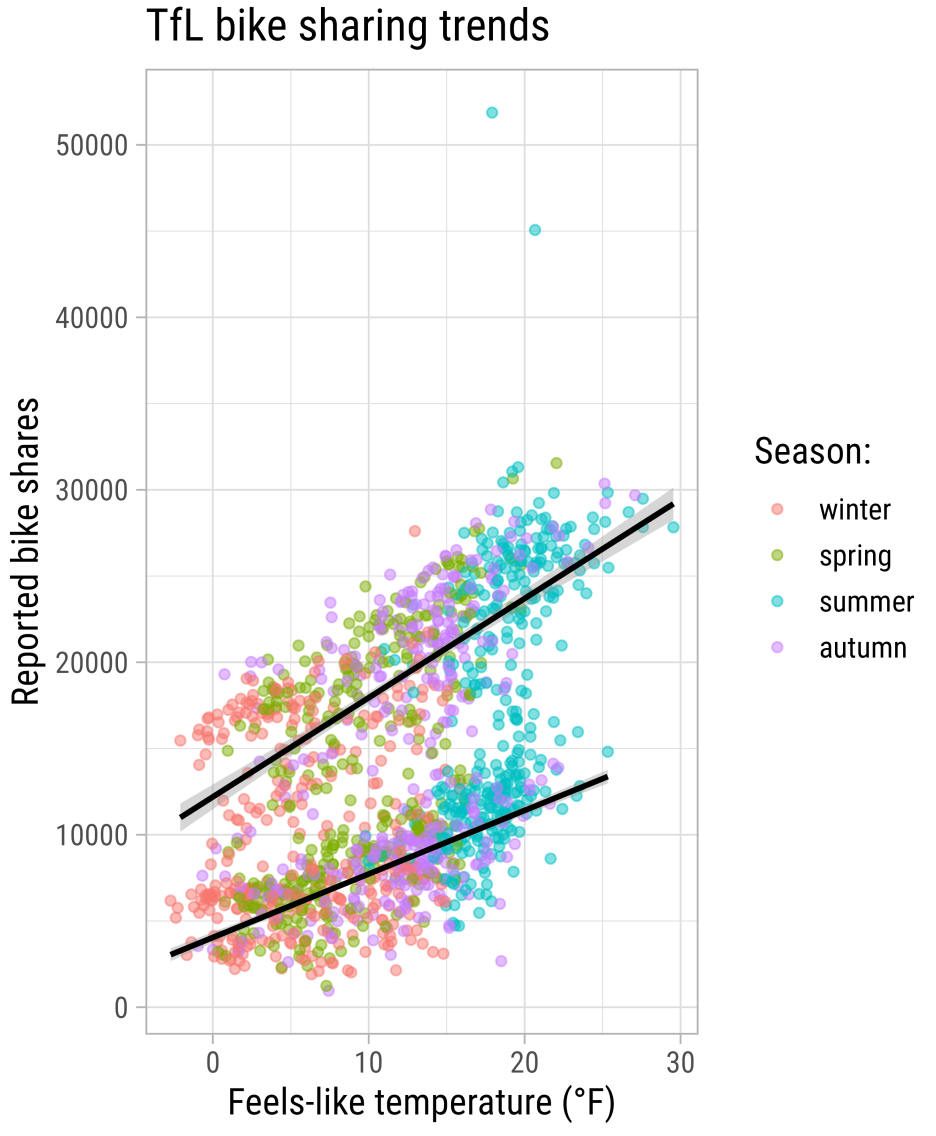
Set a Theme Globally

```
1  theme_set(theme_light())
2
3  g
```



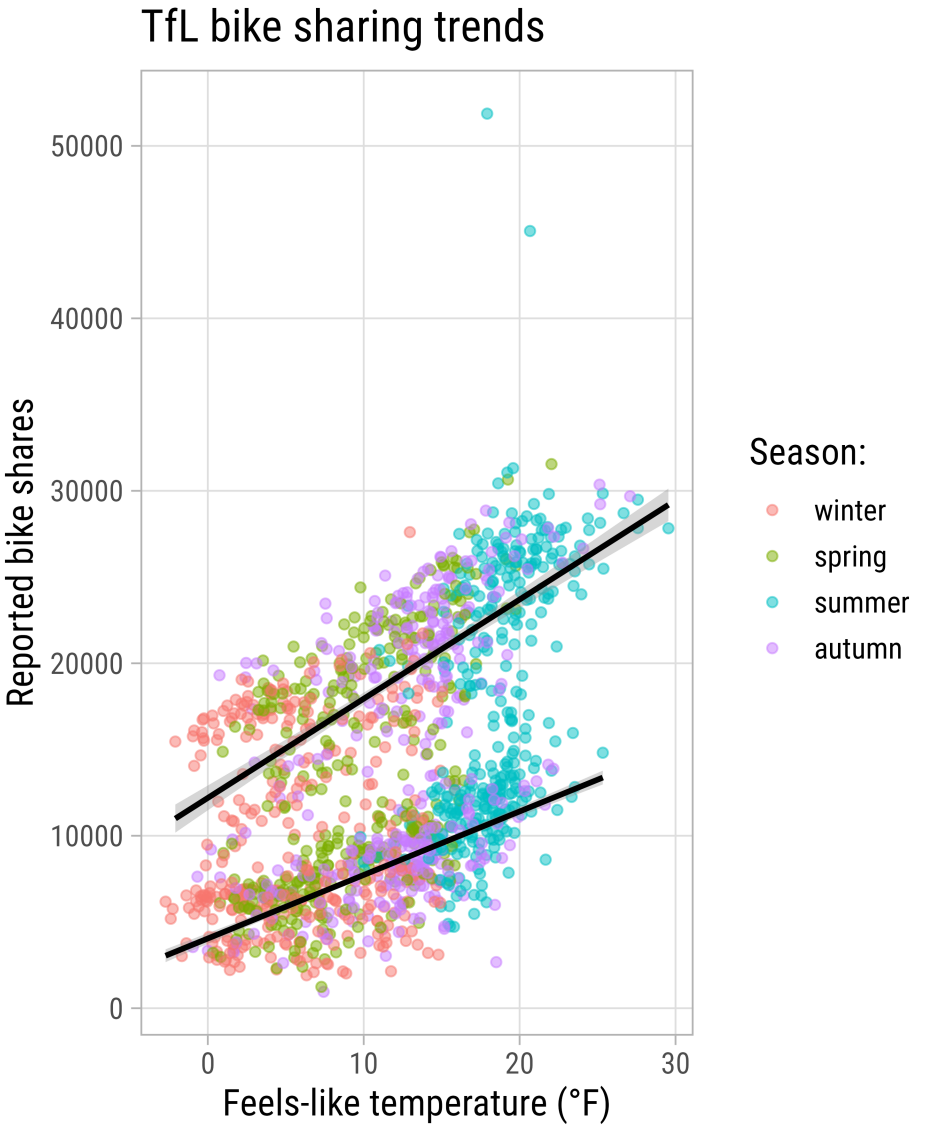
Change the Theme Base Settings

```
1 theme_set(theme_light(
2   base_size = 14,
3   base_family = "Roboto Condensed"
4 ))
5
6 g
```



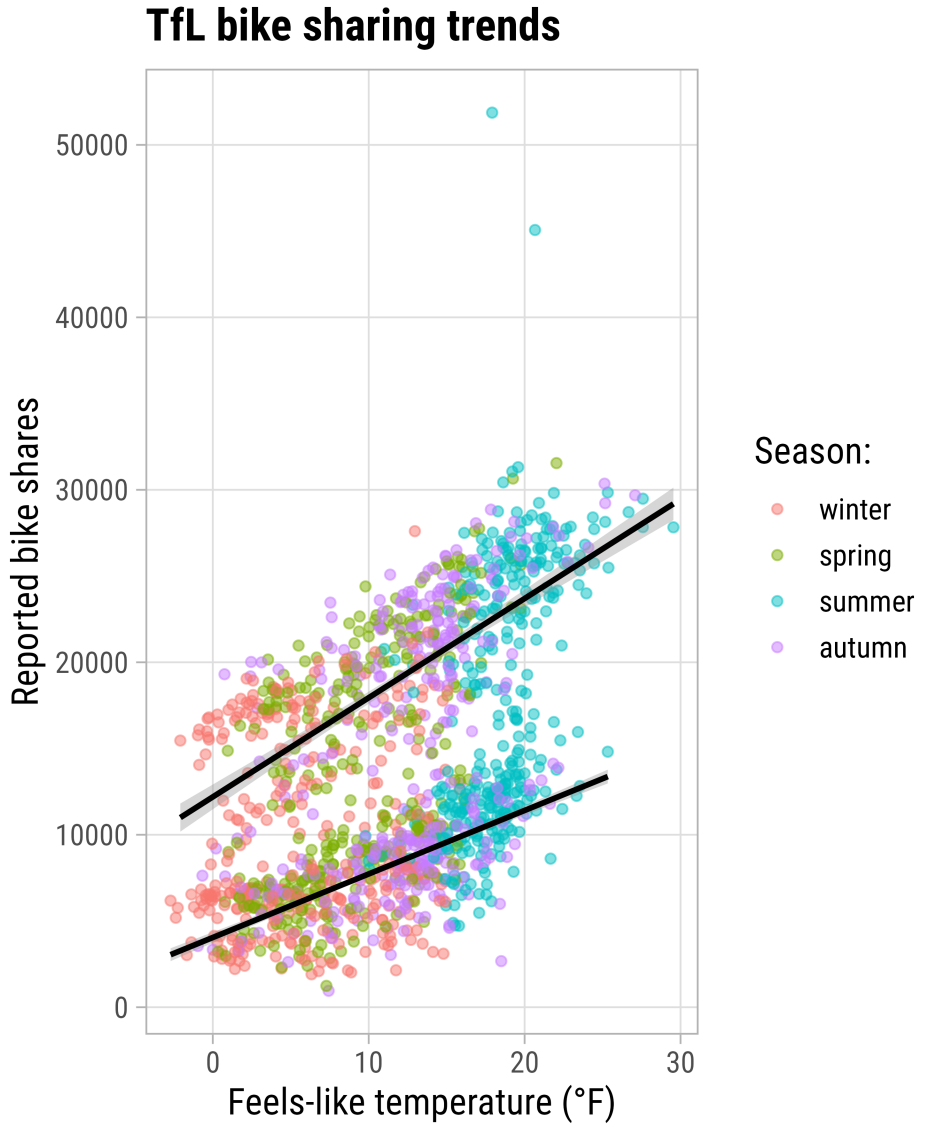
Overwrite Specific Theme Settings

```
g +
  theme(
    panel.grid.minor = element_blank()
  )
```



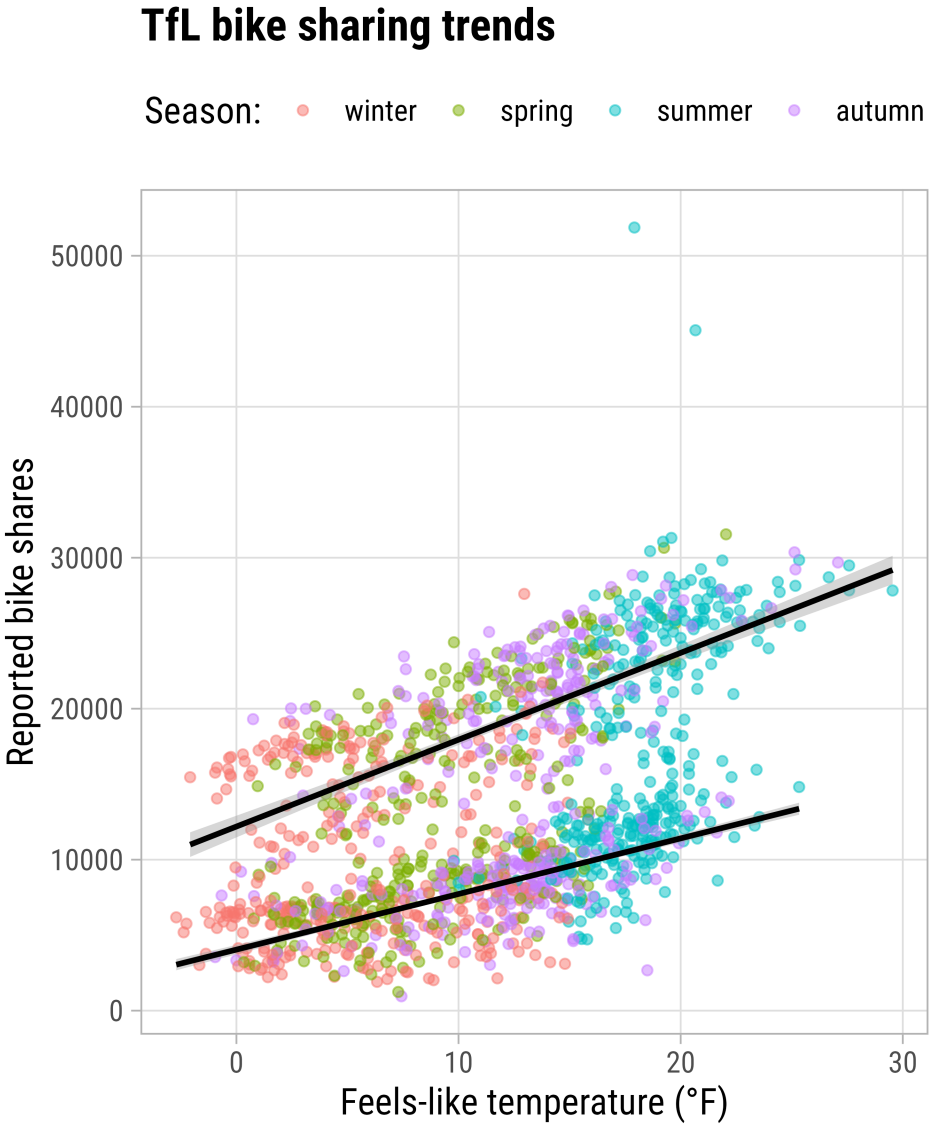
Overwrite Specific Theme Settings

```
g +
  theme(
    panel.grid.minor = element_blank(),
    plot.title = element_text(face = "bold")
  )
```



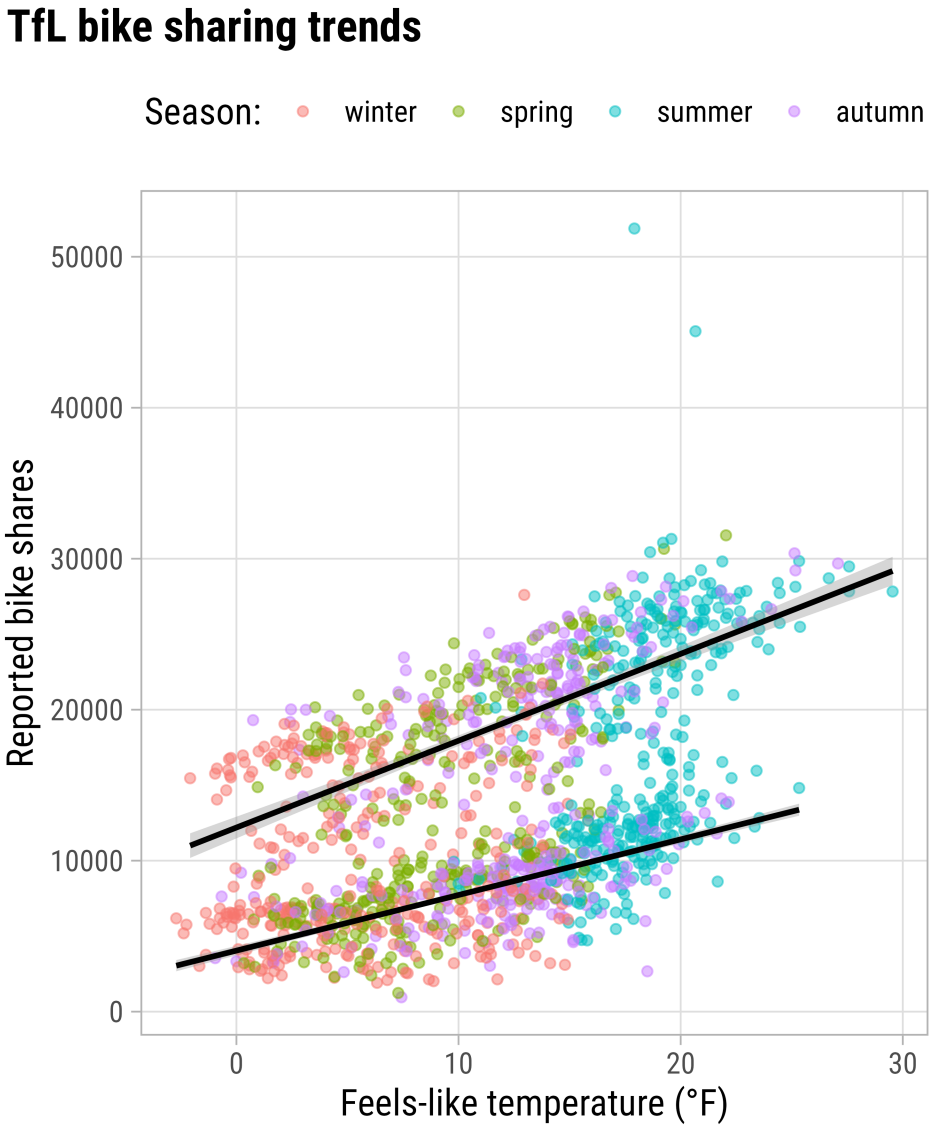
Overwrite Specific Theme Settings

```
g +
  theme(
    panel.grid.minor = element_blank(),
    plot.title = element_text(face = "bold"),
    legend.position = "top"
  )
```



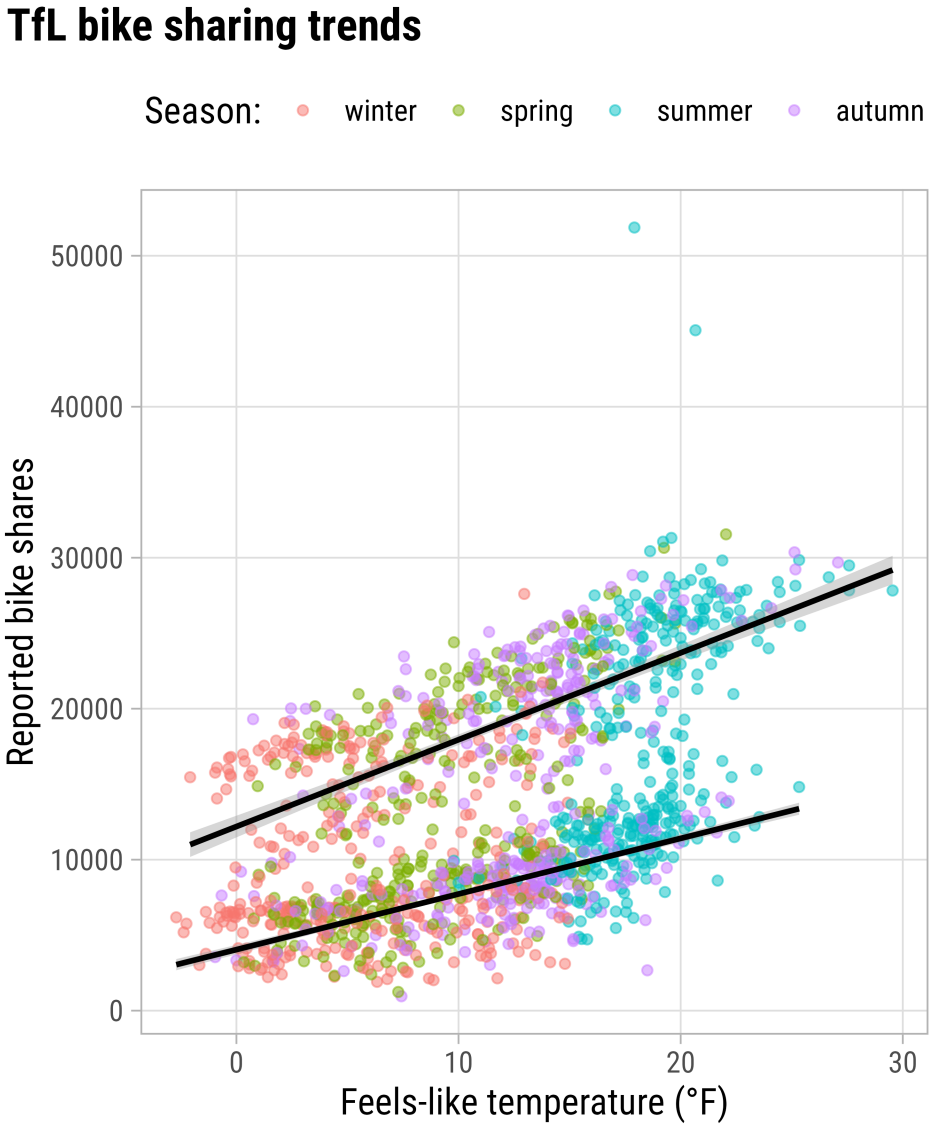
Overwrite Specific Theme Settings

```
g +
  theme(
    panel.grid.minor = element_blank(),
    plot.title = element_text(face = "bold"),
    legend.position = "top",
    plot.title.position = "plot"
  )
```



Overwrite Theme Settings Globally

```
1  theme_update(
2    panel.grid.minor = element_blank(),
3    plot.title = element_text(face = "bold"),
4    legend.position = "top",
5    plot.title.position = "plot"
6  )
7
8  g
```



Save the Graphic

```
ggsave(g, filename = "my_plot.png")
```

```
ggsave("my_plot.png")
```

```
ggsave("my_plot.png", width = 8, height = 5, dpi = 600)
```

```
ggsave("my_plot.pdf", width = 20, height = 12, unit = "cm", device = cairo_pdf)
```

```
grDevices::cairo_pdf("my_plot.pdf", width = 10, height = 7)
g
dev.off()
```

Modified from canva.com

How to Work with Aspect Ratios

- don't rely on the Rstudio viewer pane!
- once you have a "it's getting close" prototype, settle on a plot size

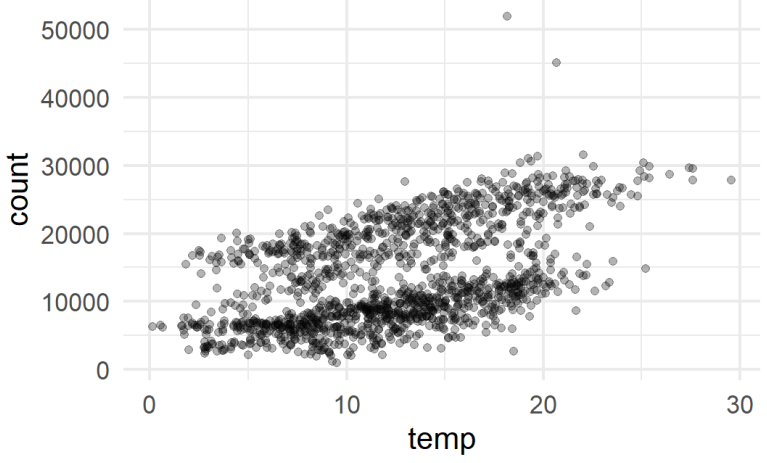
- **Approach 1:** save the file to disk and inspect it; go back to your IDE
  - tedious and time-consuming...

- **Approach 2:** use a qmd or rmd with inline output and chunk settings
  - set `fig.width` and `fig.height` per chunk or globally

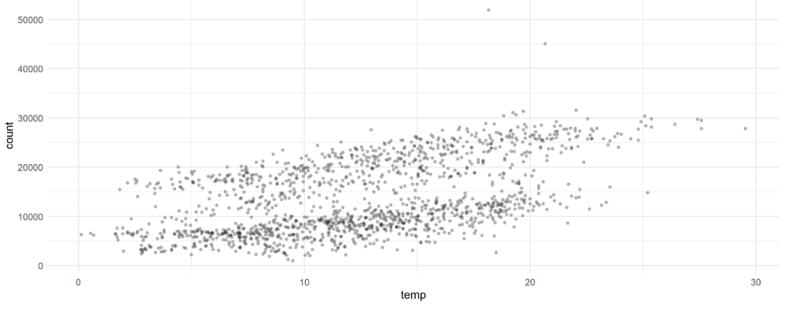
- **Approach 3:** use our {camcorder} package
  - saves output from all `ggplot()` calls and displays it in the viewer pane

Setting Plot Sizes in Rmd's

````
```{r plot-object}
g <- ggplot(bikes, aes(x = temp, y = count)) +
  geom_point(alpha = .3, size = 2) +
  theme_minimal(base_size = 18)
```

```{r plot-a}
g
```
````




````
```{r plot-b, fig.width=10, fig.height=4}
g
```
````



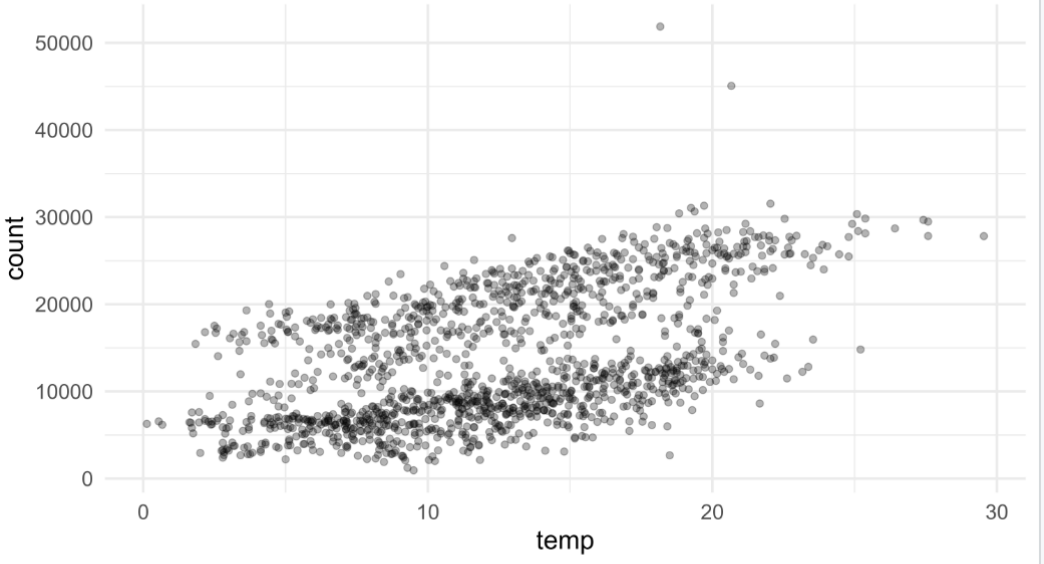
Setting Plot Sizes via {camcorder}

plot-size-camcorder-example.R

Source on Save | Run | Source

```r
library(tidyverse)
library(camcorder)

bikes <- readr::read_csv(
  "https://raw.githubusercontent.com/z3tt/graphic-design-ggplot2
)

gg_record(
  dir = here::here("temp_plots"), # where to save the recording
  device = "png", # device to use to save images
  width = 10, # width of saved image
  height = 5.5, # height of saved image
  units = "in", # units for width and height
  dpi = 600 # dpi to use when saving image
)

ggplot(bikes, aes(x = temp, y = count)) +
  geom_point(alpha = .3, size = 2) +
  theme_minimal(base_size = 18)
```

Files | Plots | Packages | Help | Viewer | Presentation



Recap

- `{ggplot2}` is a powerful library for reproducible graphic design
- the components follow a consistent syntax
- each ggplot needs at least **data**, some **aesthetics**, and a **layer**
- we **set** constant propeties outside `aes()`
- ... and **map** data-related properties inside `aes()`
- local settings and mappings override global properties
- grouping allows applying layers for subsets
- we can store a ggplot object and extend it afterwards
- we can change the appearance for all plots with `theme_set()` and `theme_update()`

Exercises

Exercise 1

- Open the script `exercises/02_concepts_pt1_ex1.qmd`.
- Explore the TfL bike share data visually:
  create a timeseries of reported bike shares on weekend days
  - Highlight day and night encoded by colors and shapes.
  - Connect the points of each period with lines.
    - What is the difference between `geom_line()` and `geom_path()`?
  - Apply your favorite theme to the plot.
  - Add meaningful labels.
- Save the plot as a vector graphic with a decent plot size.

I am done with the exercise!
What now?

↓

Put on a **green** sticky note.

↓

Is my neighbor done
with the exercise as well?

Yes | **No**

Compare and discuss your solutions.

Ask if you can help out.

... or take a short break.

Exercise 2

- Open the script `exercises/02_concepts_pt1_ex2.qmd`.
- Explore the TfL bike sharing data visually:
  create a boxplot of counts per weather type
  - Turn the plot into a jitter strips plot (random noise across the x axis)
  - Combine both chart types (jittered points on top of the boxplots)
  - Bonus: Sort the boxplot-jitter hybrid by median counts
  - Apply your favorite theme to the plot.
  - Add meaningful labels.
  - Bonus: Explore other chart types to visualize the distributions.
- Save the plot as a vector graphic with a decent plot size.

I am done with the exercise!
What now?

↓

Put on a **green** sticky note.

↓

Is my neighbor done
with the exercise as well?

Yes | **No**

Compare and discuss your solutions.

Ask if you can help out.

... or take a short break.