

Graphic Design with ggplot2

Concepts of the {ggplot2} Package Pt. 1:

Data, Aesthetics, and Layers + **Misc Stuff**

Modified from Cedric Scherer RStudio::conf July 2022 workshop

The Grammar of `{ggplot2}`

The Grammar of {ggplot2}

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

The Grammar of {ggplot2}

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

```
1 bikes <- readr::read_csv(  
2   here::here("data", "london-bikes-custom.csv"),  
3   ## or: "https://raw.githubusercontent.com/z3tt/graphic-design-ggplot2/main/data/london-bikes-custo  
4   col_types = "Dcffffillllddddc"  
5 )  
6  
7 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.

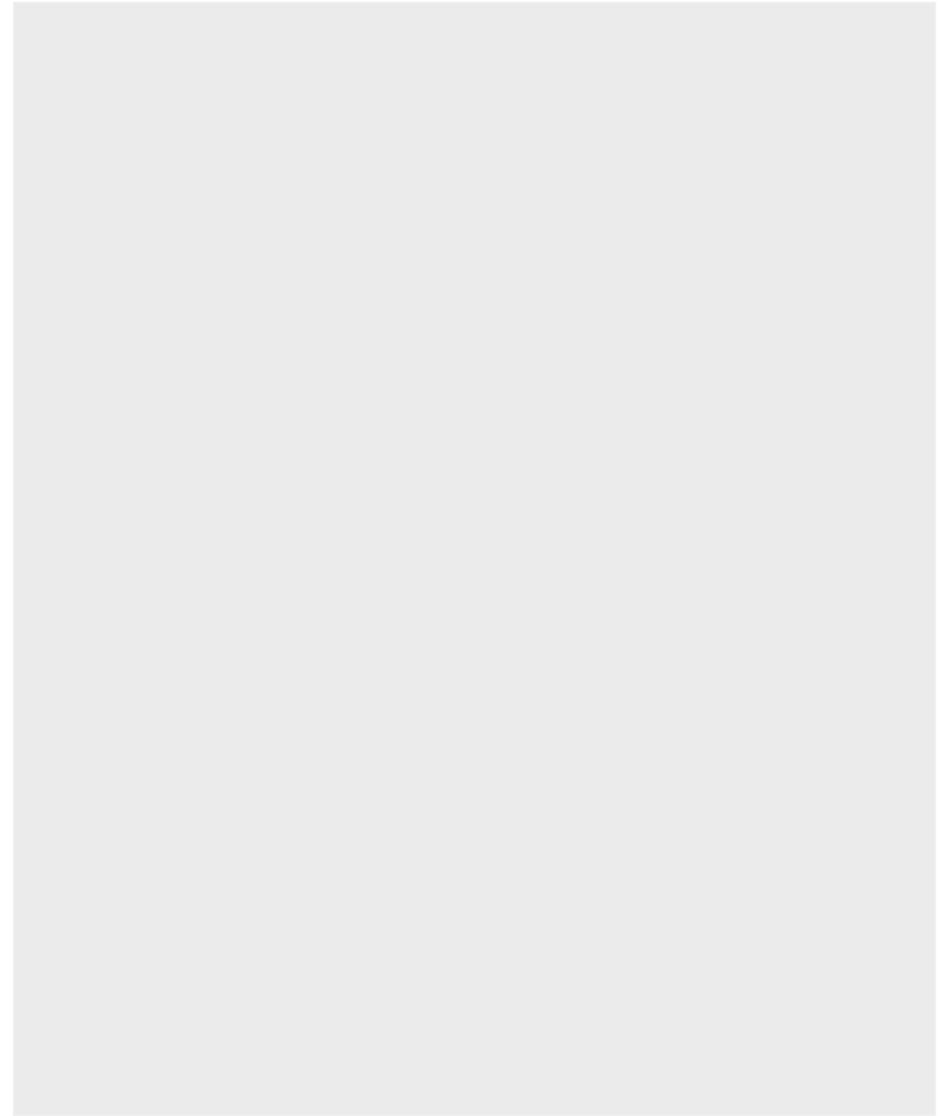
Details

`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)
```



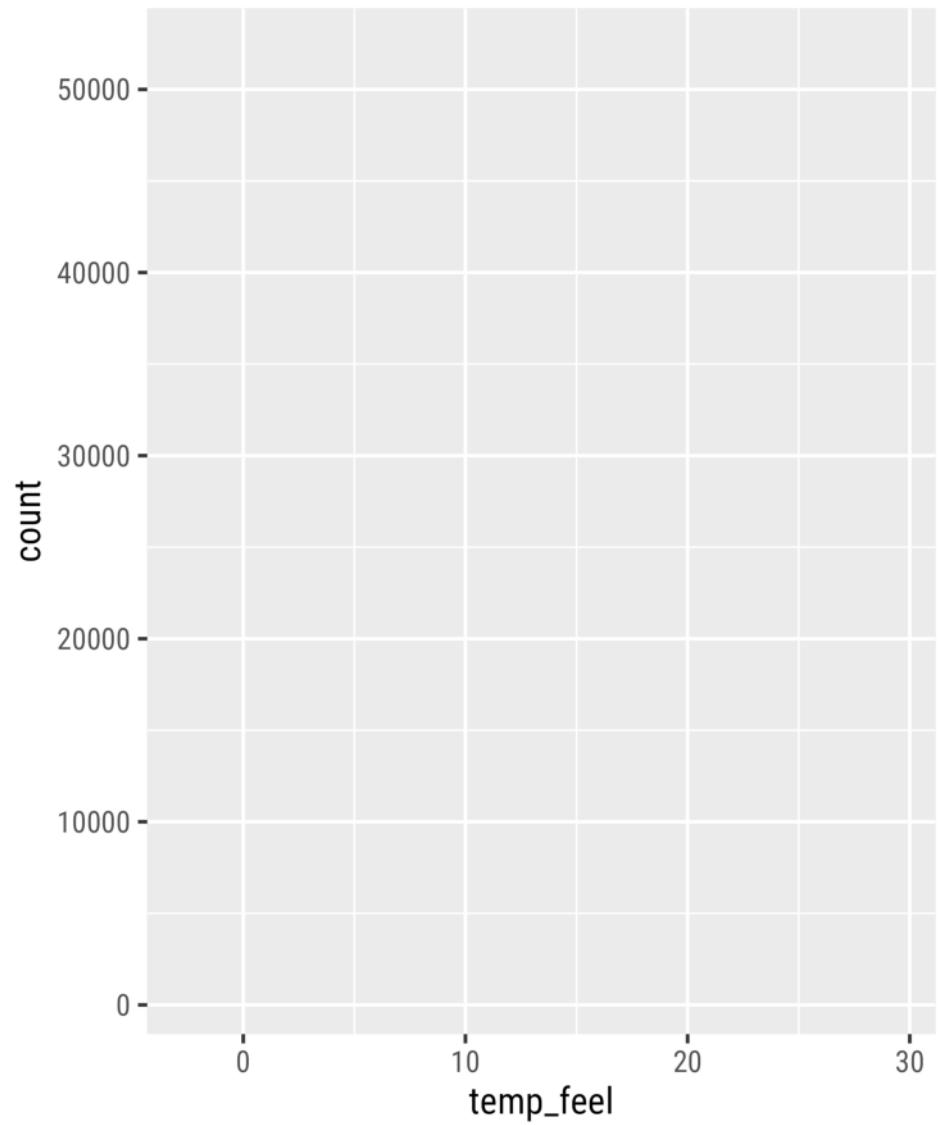
Aesthetic Mapping

= 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

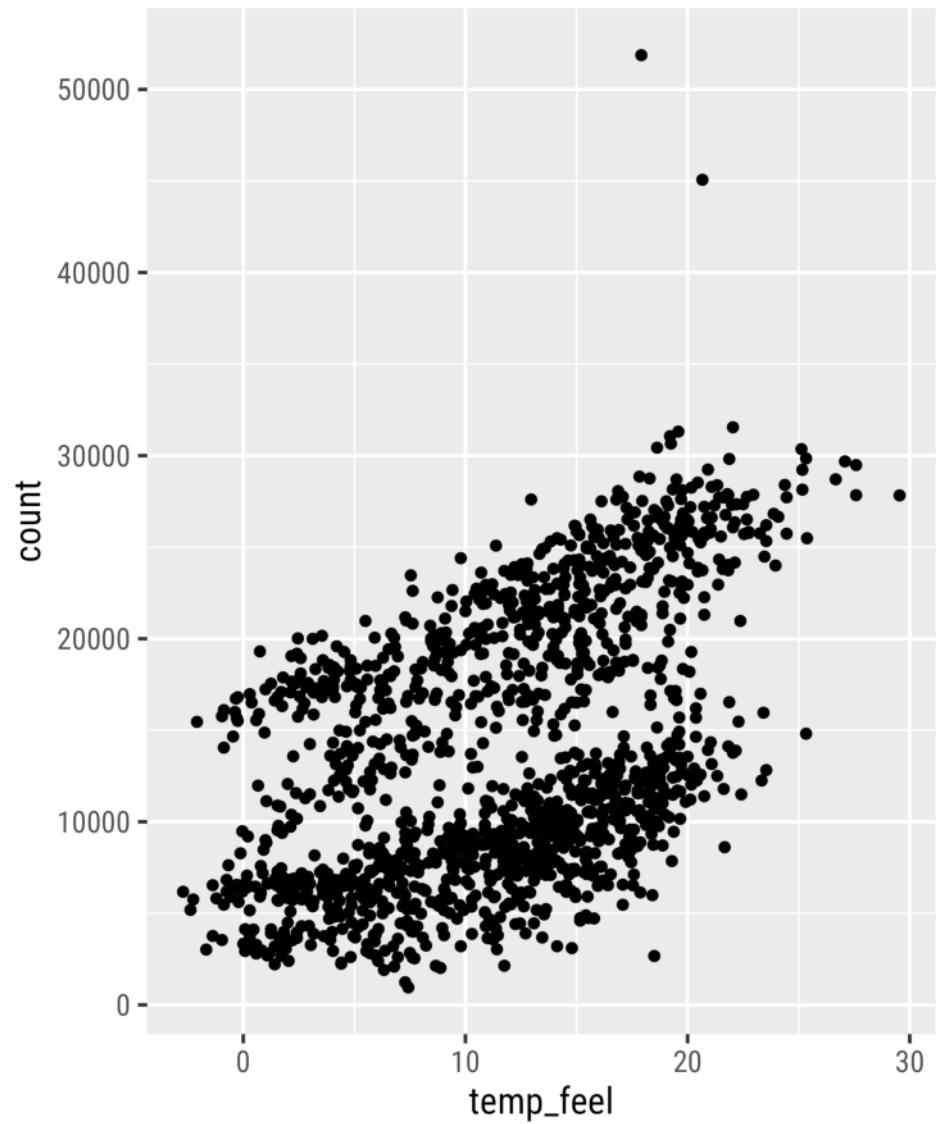
Geometries

= interpret aesthetics as graphical representations

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

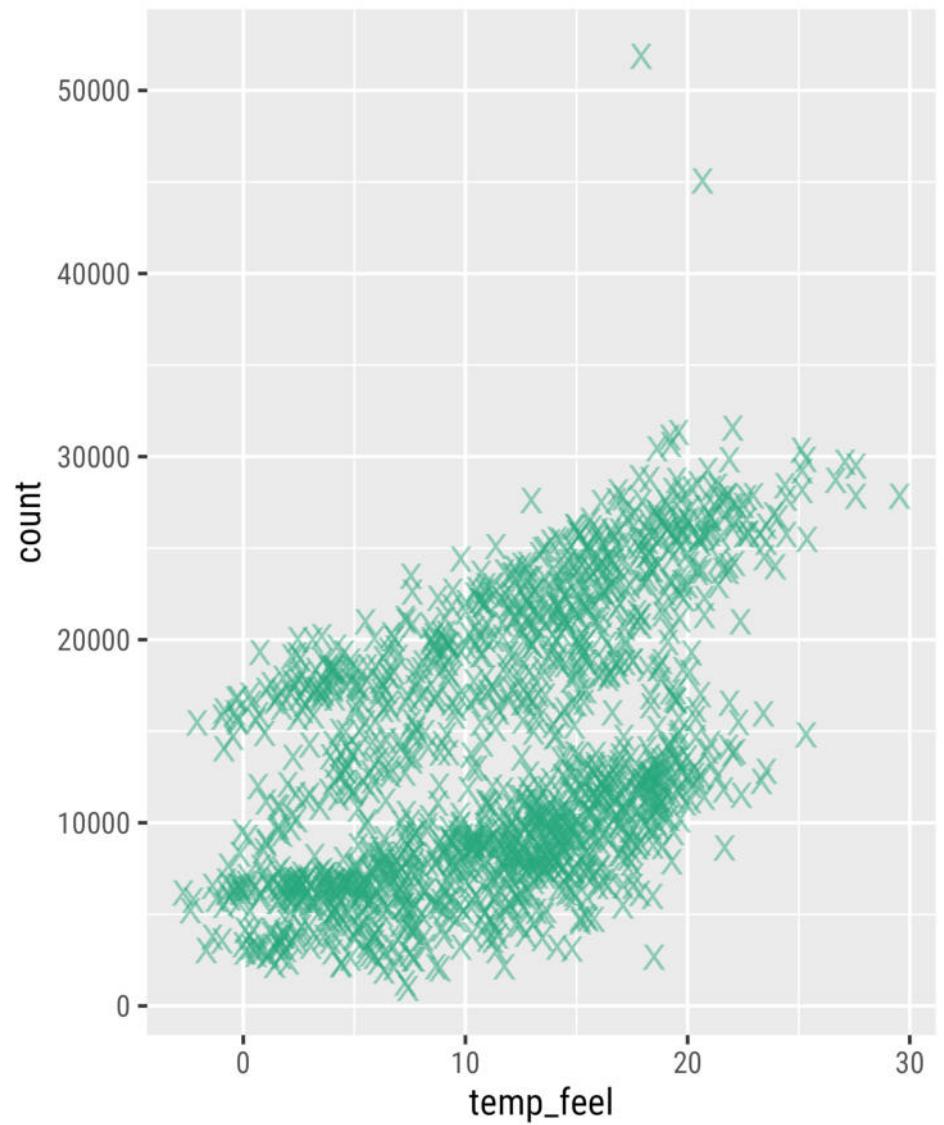
Geometries

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



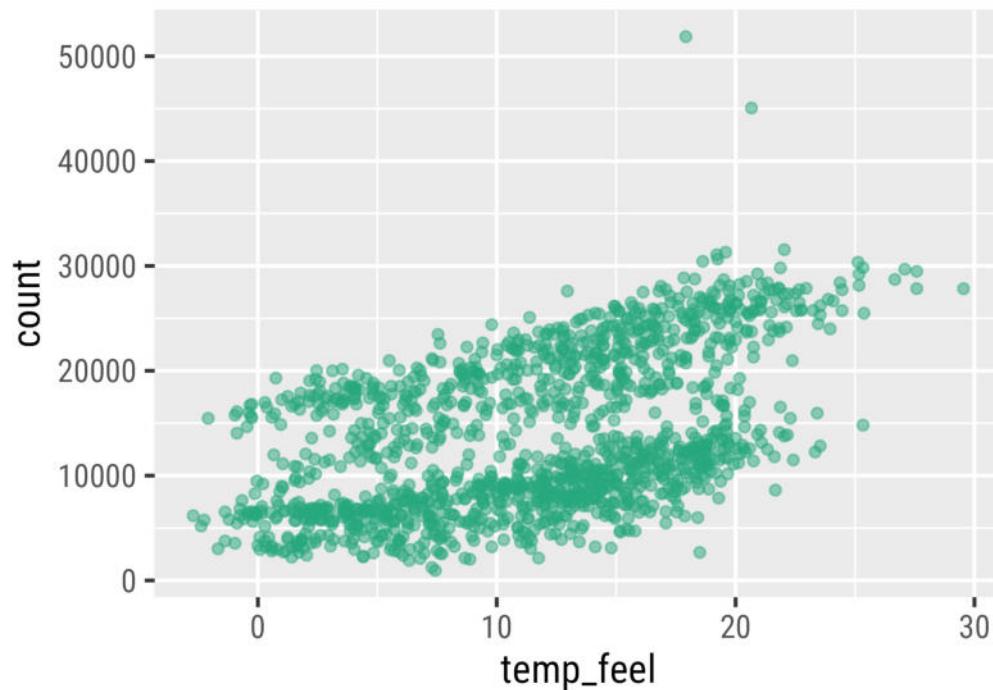
Visual Properties of Layers

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count))  
4 ) +  
5   geom_point(  
6     color = "#28a87d",  
7     alpha = .5,  
8     shape = "X",  
9     stroke = 1,  
10    size = 4  
11 )
```

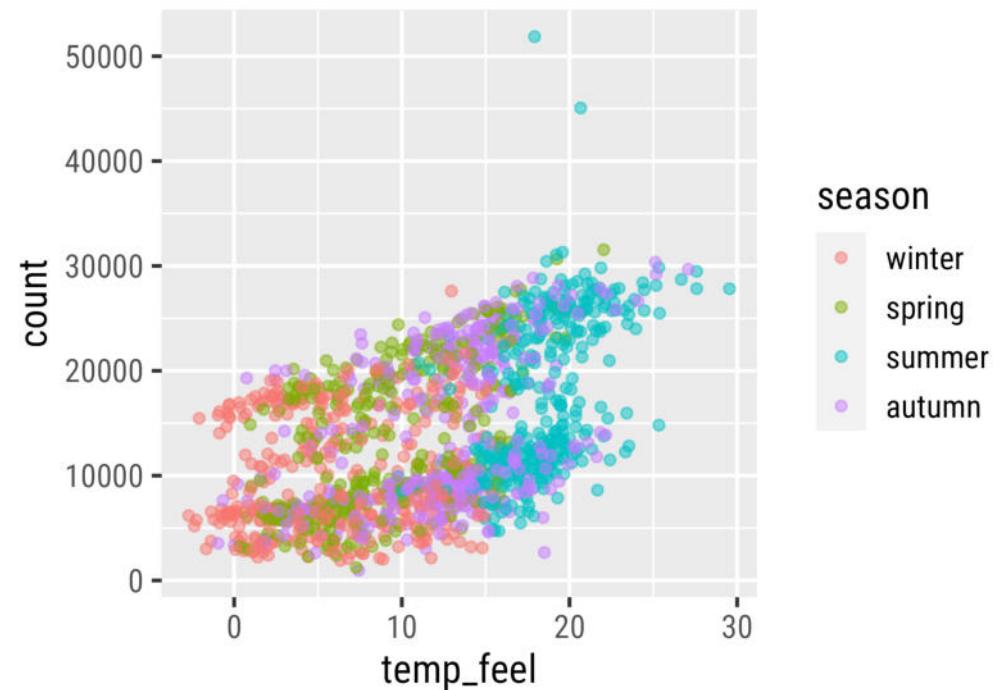


Setting vs Mapping of Visual Properties

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

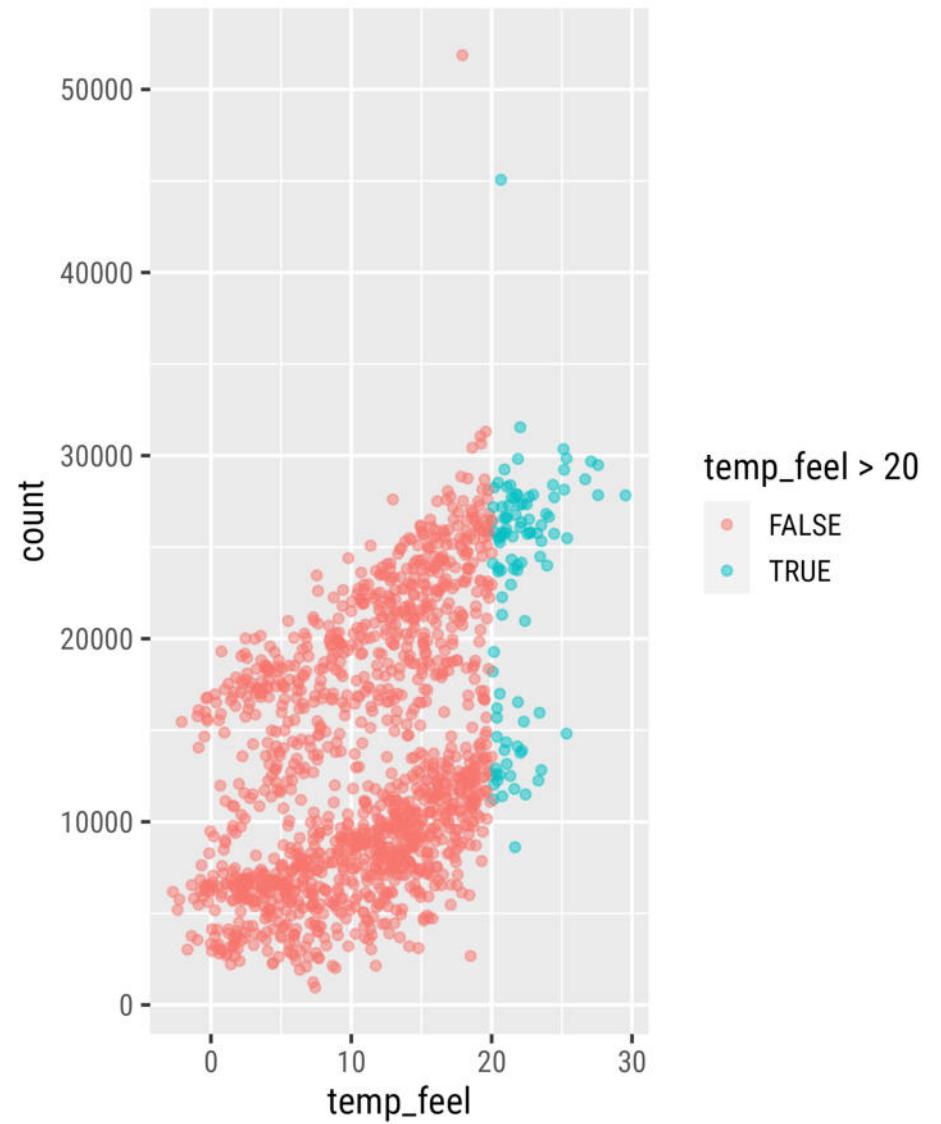


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5   geom_point(  
6     aes(color = season),  
7     alpha = .5  
8 )
```



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 )
```

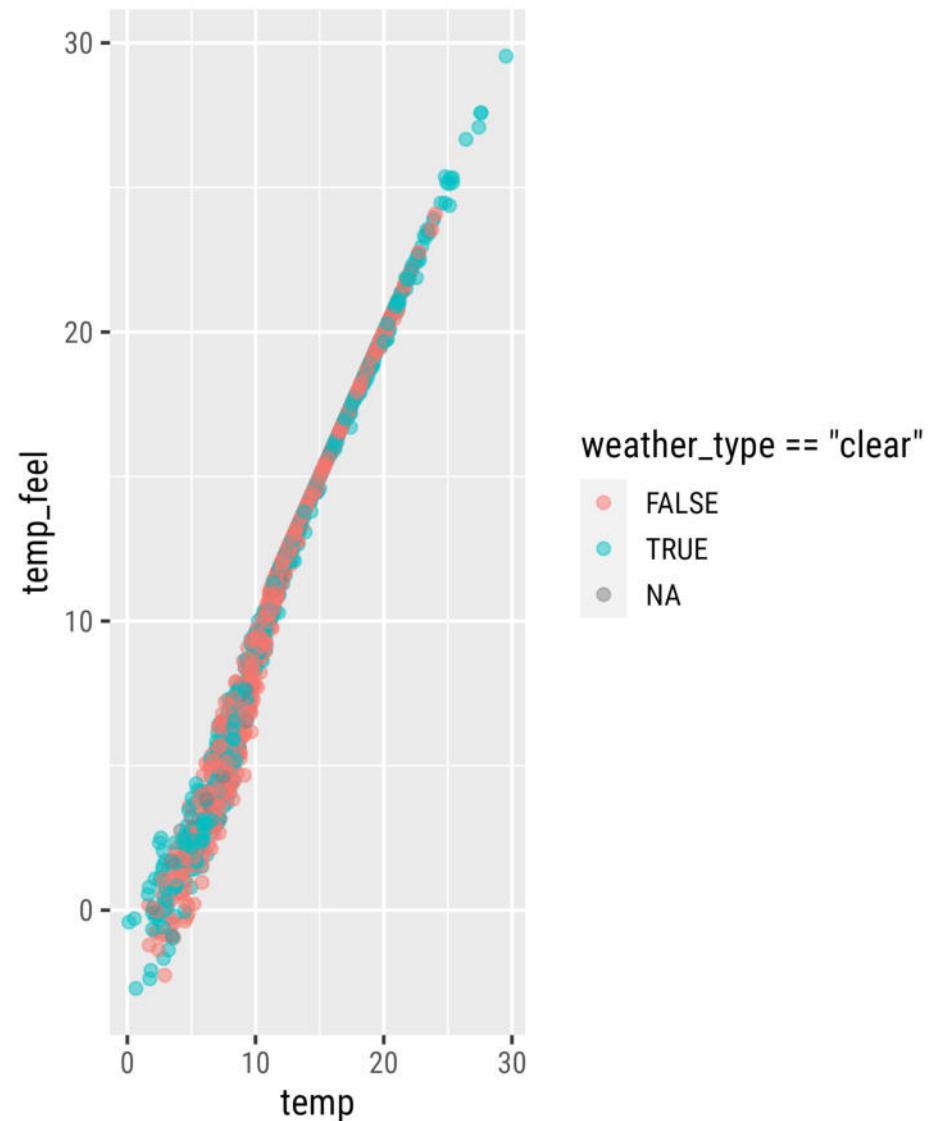


Your Turn!

- 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?

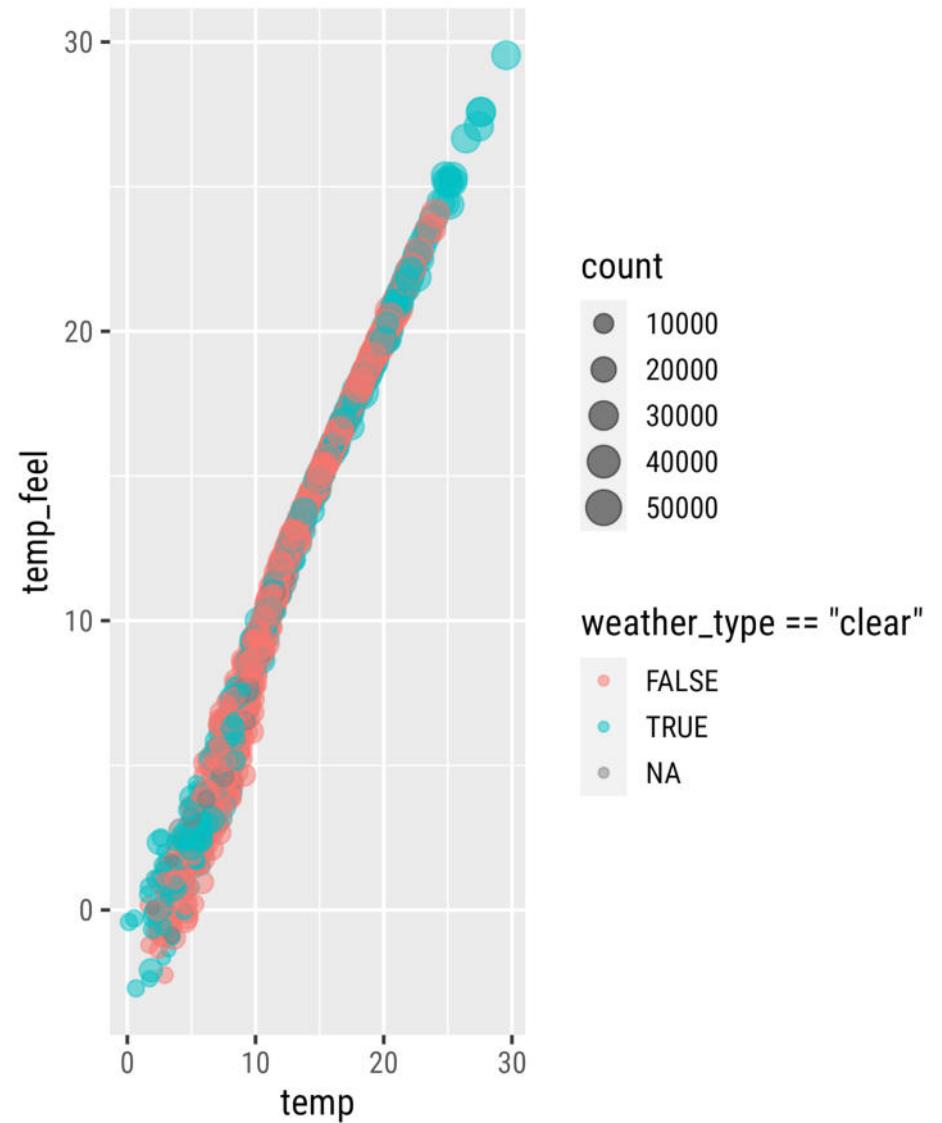
Mapping Expressions

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6   aes(color = weather_type == "clear"),  
7   alpha = .5,  
8   size = 2  
9 )
```



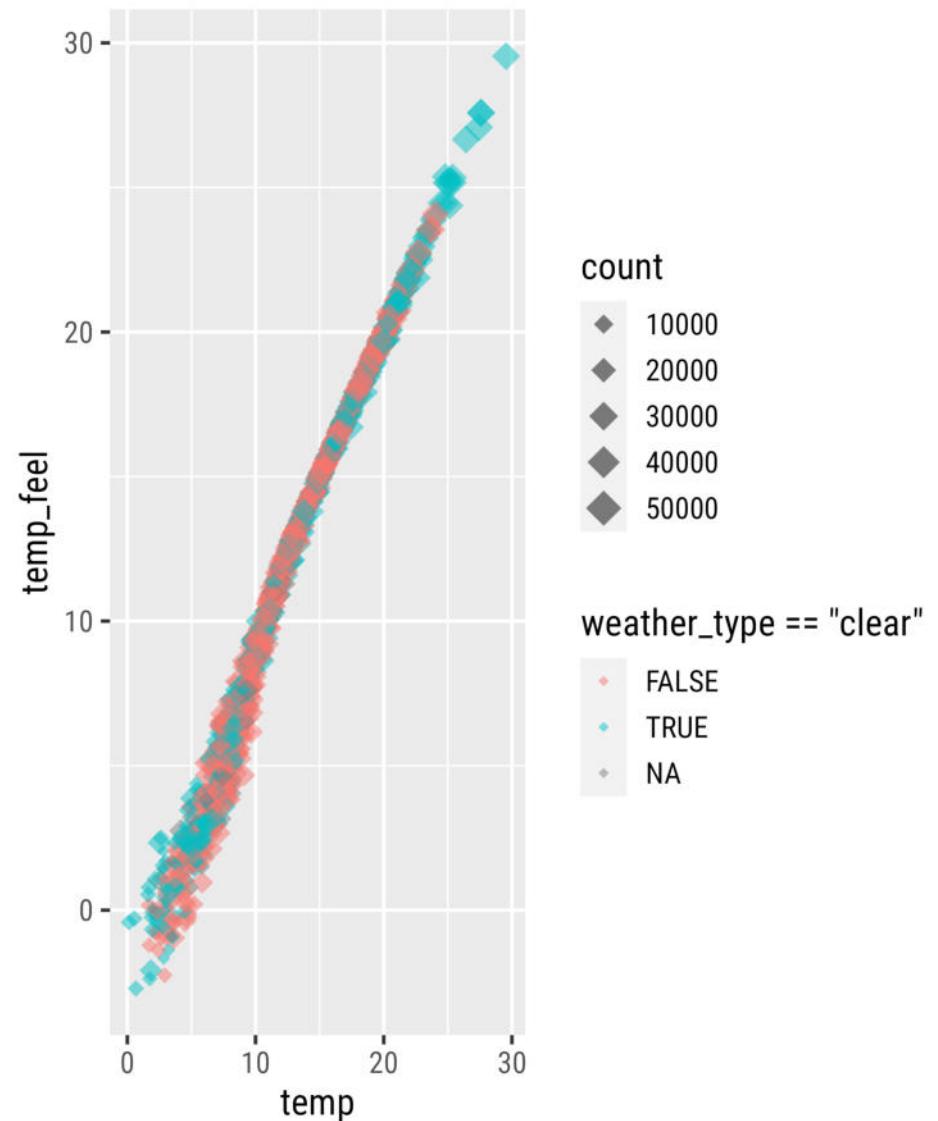
Mapping to Size

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(color = weather_type == "clear",  
7           size = count),  
8     alpha = .5  
9   )
```



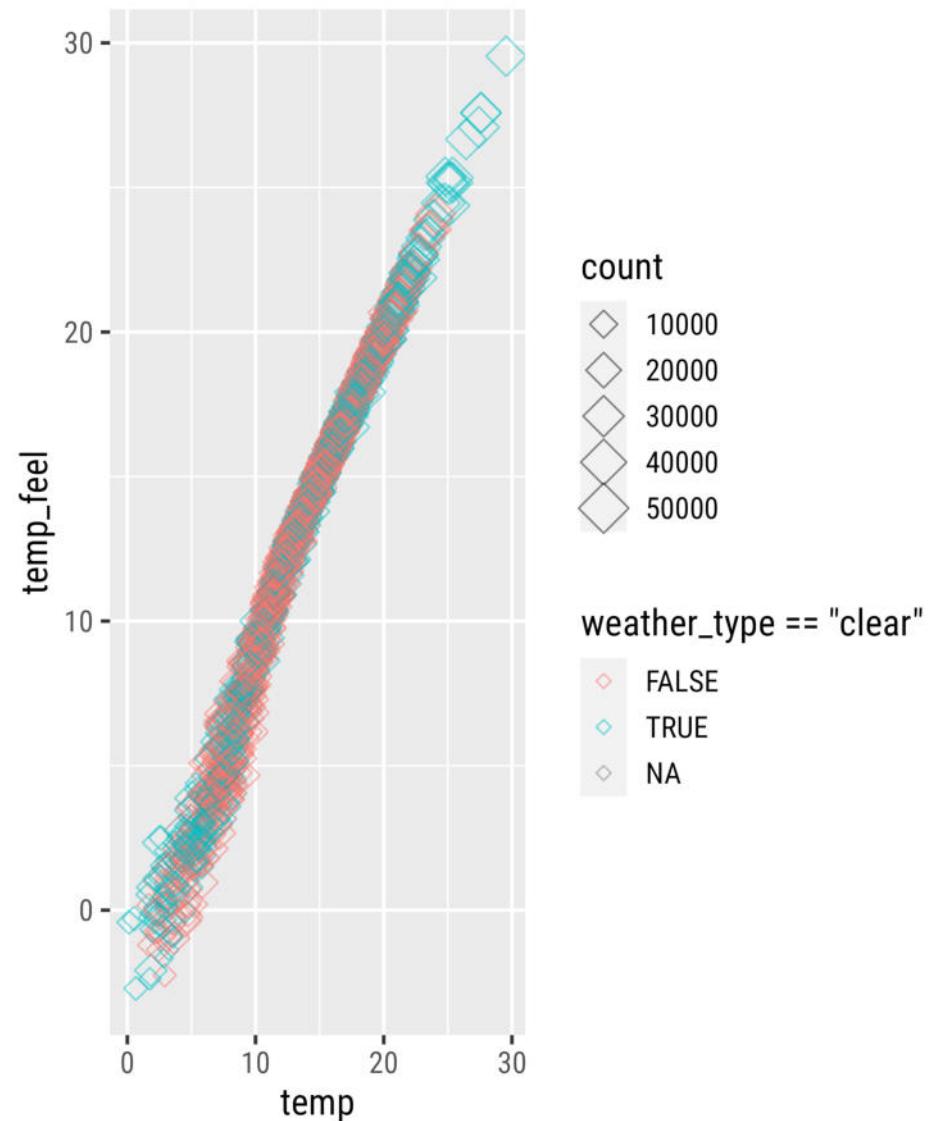
Setting a Constant Property

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(color = weather_type == "clear",  
7           size = count),  
8     shape = 18,  
9     alpha = .5  
10 )
```



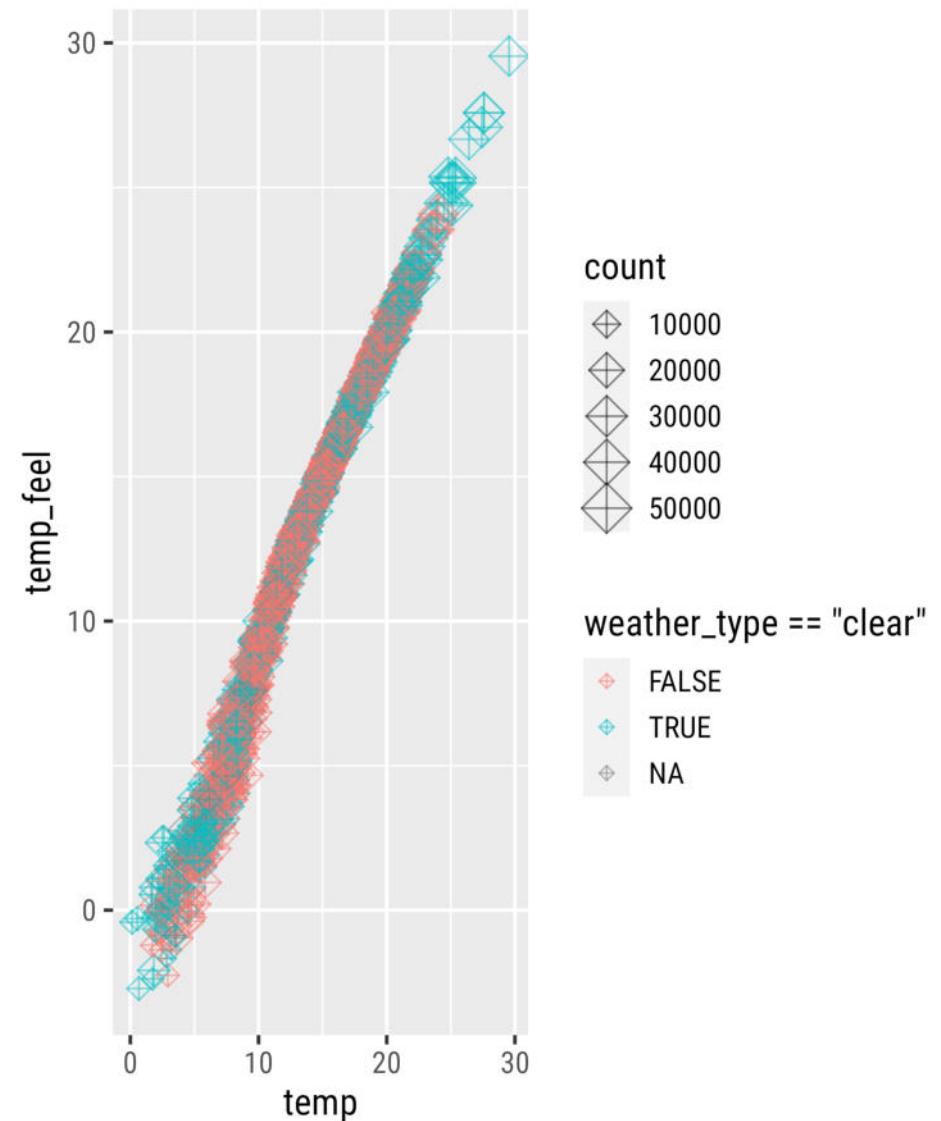
Setting a Constant Property

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(color = weather_type == "clear",  
7           size = count),  
8     shape = 5,  
9     alpha = .5  
10 )
```



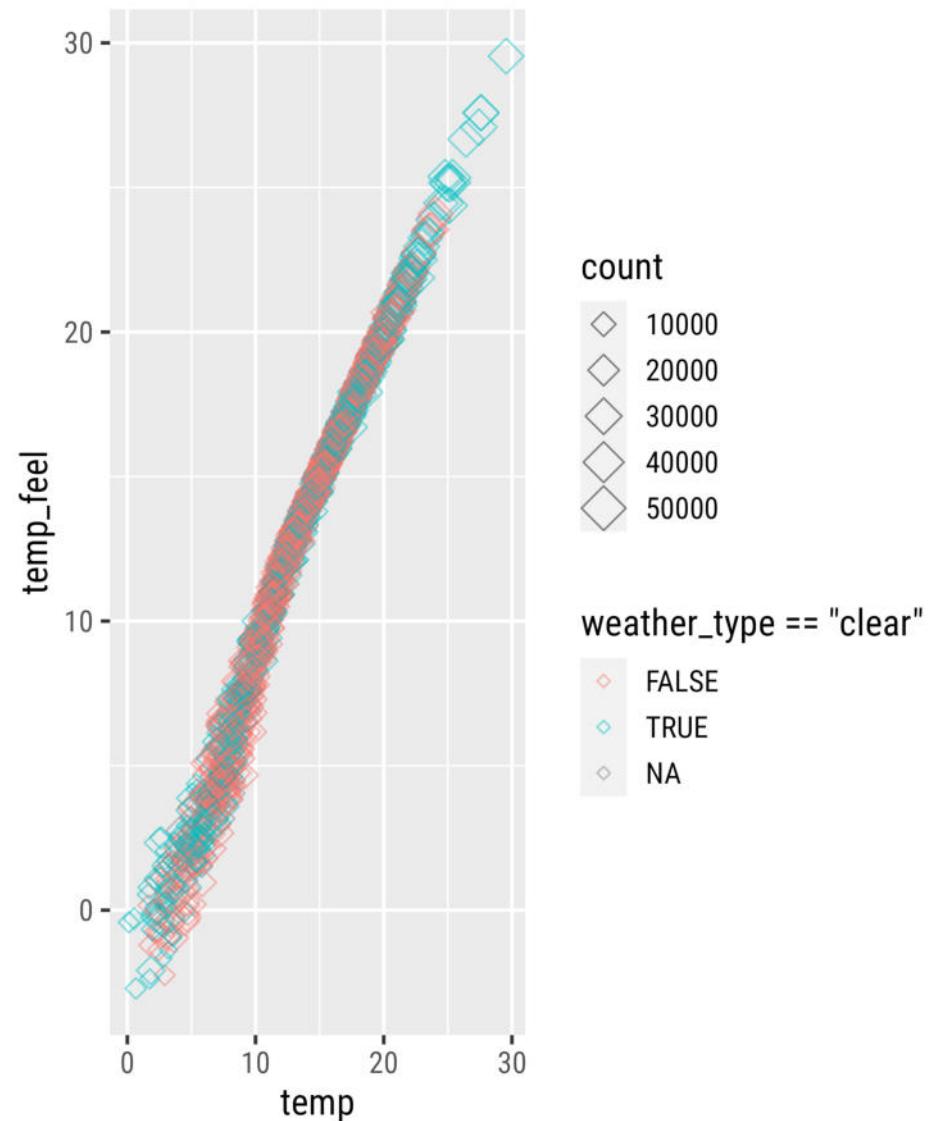
Setting a Constant Property

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6   aes(color = weather_type == "clear",  
7       size = count),  
8   shape = 9,  
9   alpha = .5  
10 )
```



Setting a Constant Property

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6   aes(color = weather_type == "clear",  
7       size = count),  
8   shape = 23,  
9   alpha = .5  
10 )
```

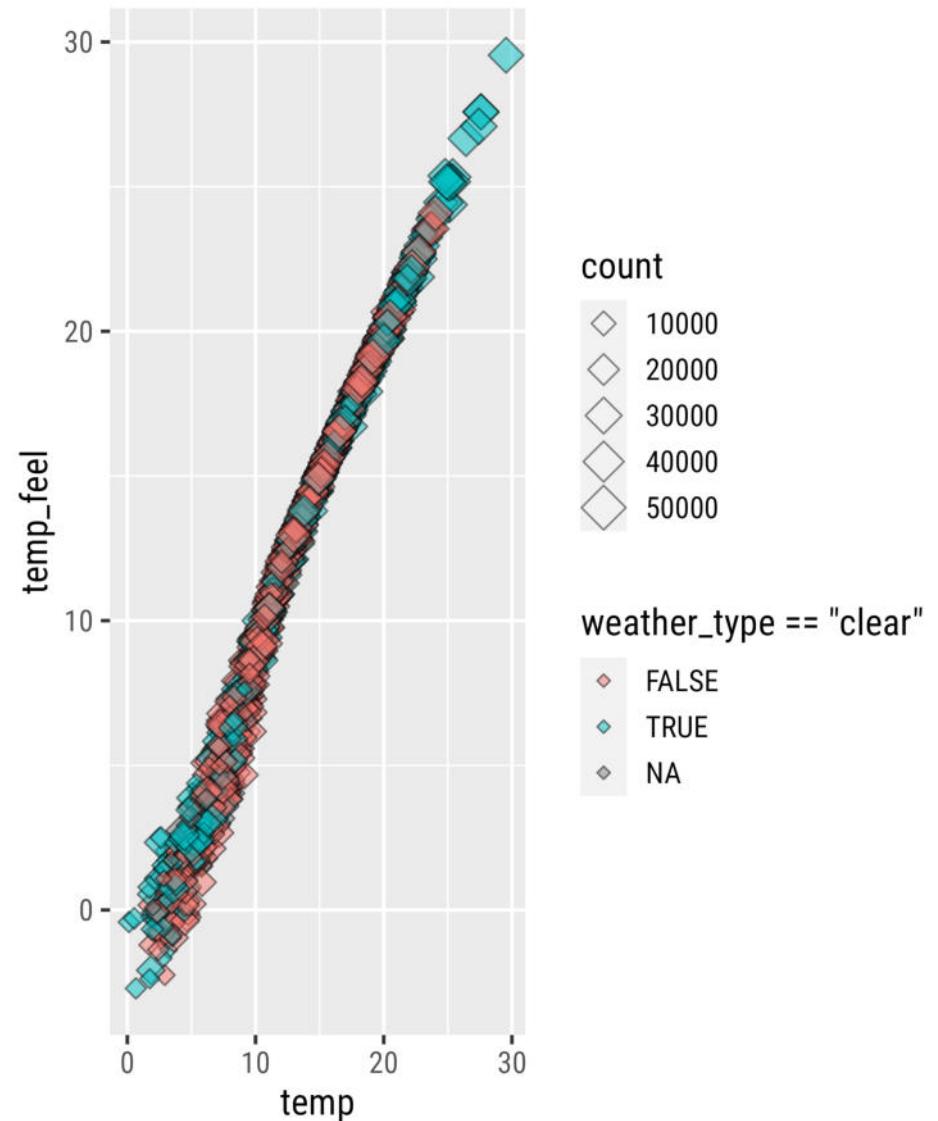


Circles	1 	10 	13 	16 	19 	20 	21
Triangles	2 	6 	17 	24 	25 		
Diamonds	5 	9 	18 	23 			
Squares	0 	7 	12 	14 	15 	22 	
Other	3 	4 	8 	11 			

Source: Albert's Blog

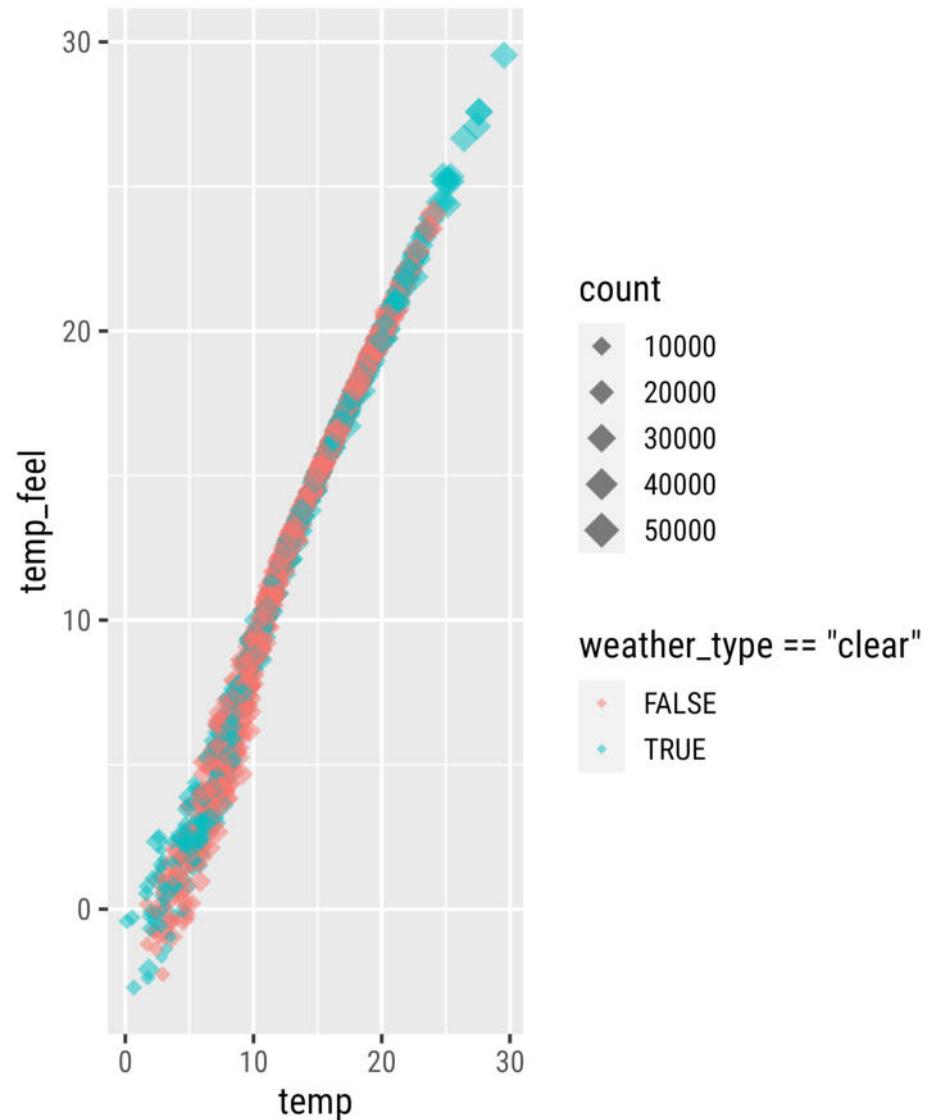
Setting a Constant Property

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(fill = weather_type == "clear",  
7           size = count),  
8     shape = 23,  
9     color = "black",  
10    alpha = .5  
11 )
```



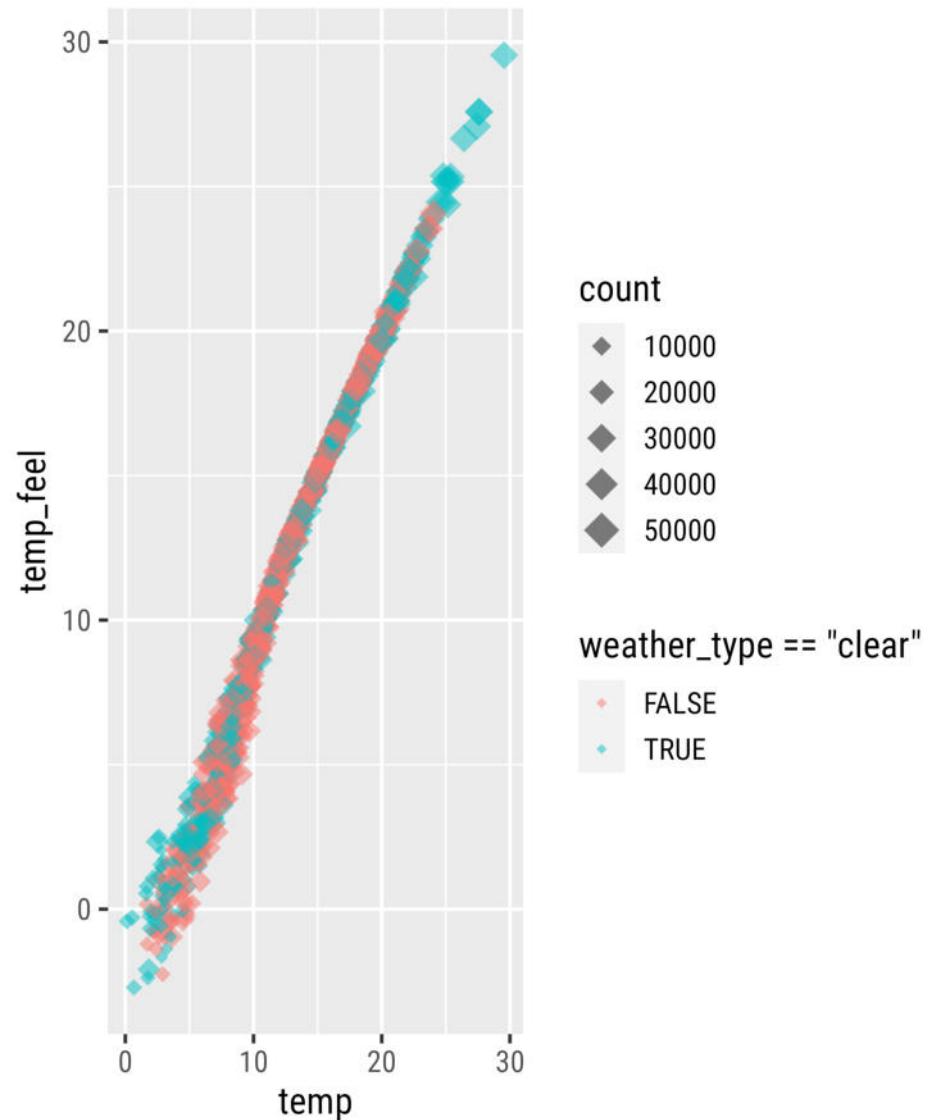
Filter Data

```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(color = weather_type == "clear",  
7           size = count),  
8     shape = 18,  
9     alpha = .5  
10 )
```



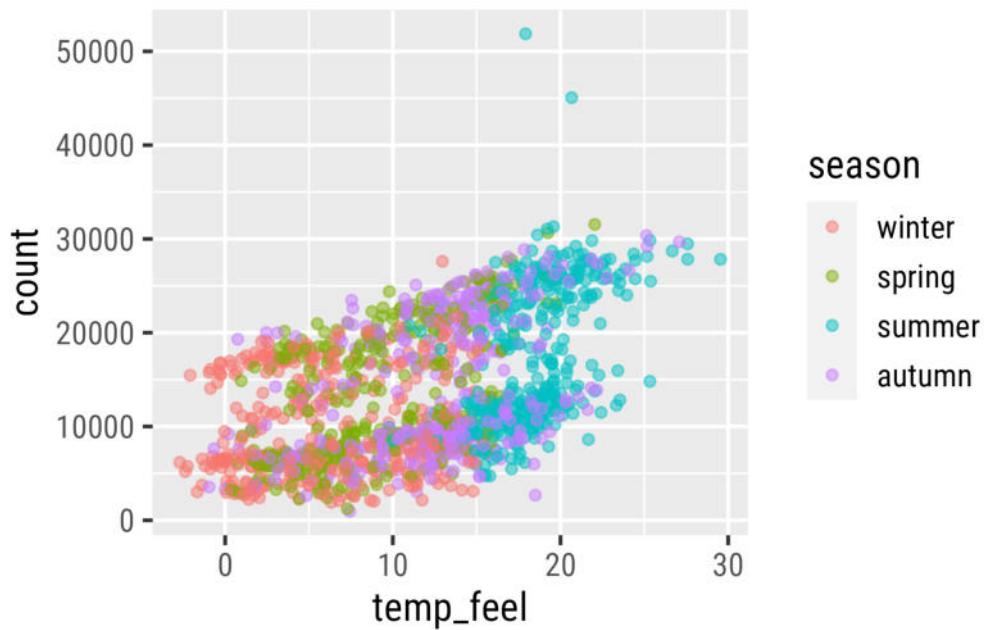
Filter Data

```
1 ggplot(  
2   bikes %>% filter(!is.na(weather_type)),  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(color = weather_type == "clear",  
7           size = count),  
8     shape = 18,  
9     alpha = .5  
10 )
```

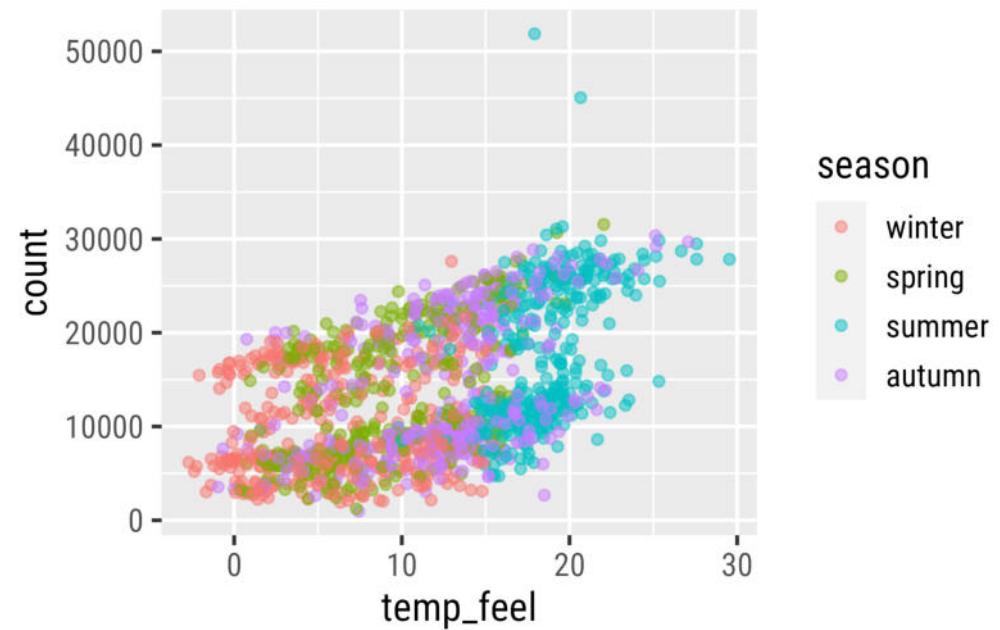


Local vs. Global Encoding

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

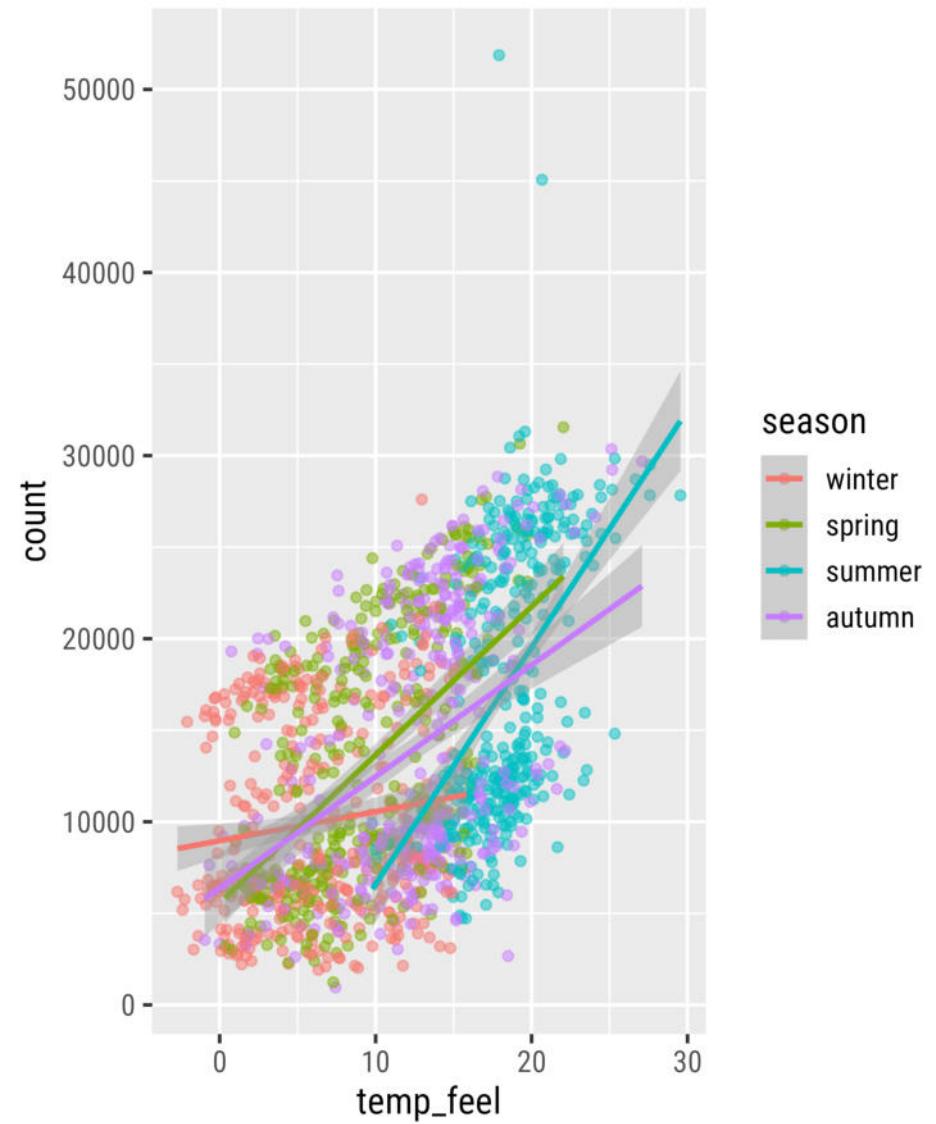


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4         color = season)  
5 ) +  
6   geom_point(  
7   alpha = .5  
8 )
```



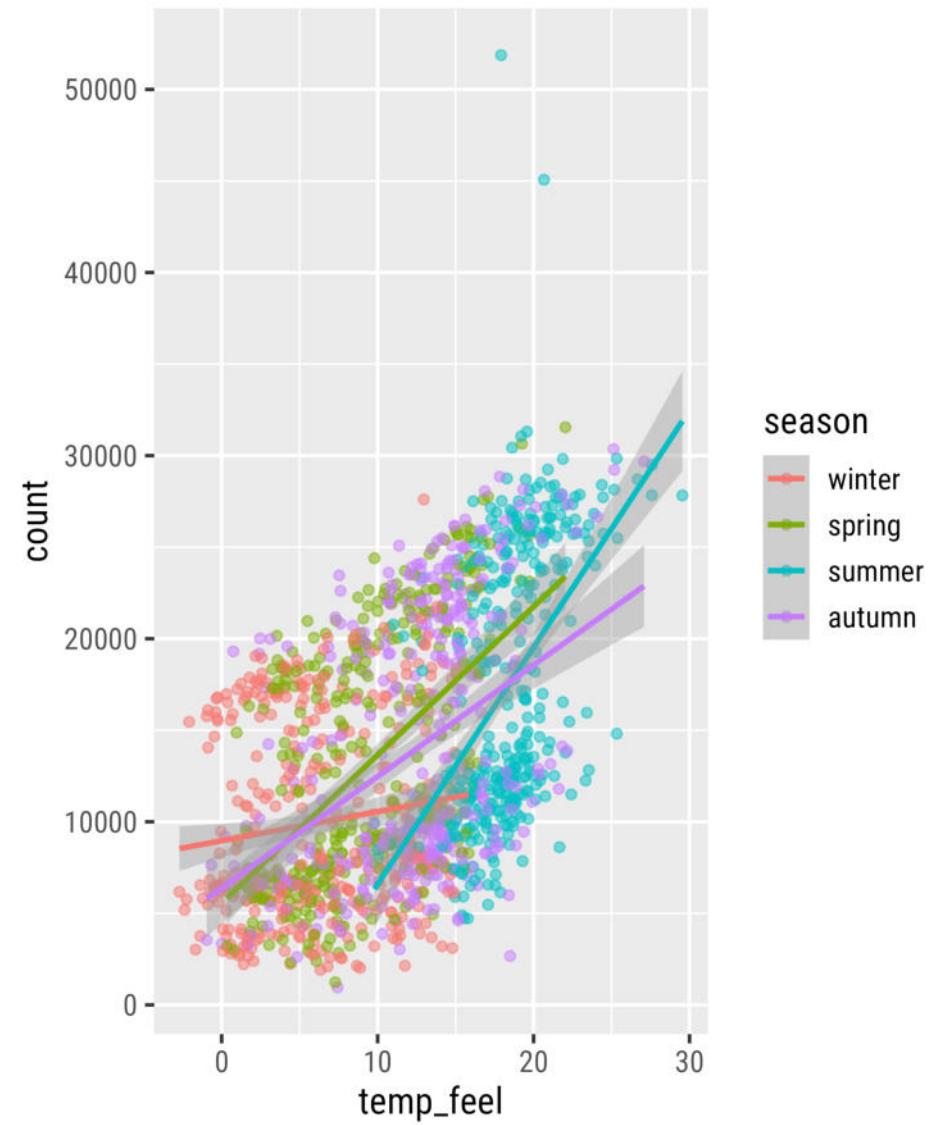
Adding More Layers

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = season)  
5 ) +  
6   geom_point(  
7     alpha = .5  
8 ) +  
9   geom_smooth(  
10    method = "lm"  
11 )
```



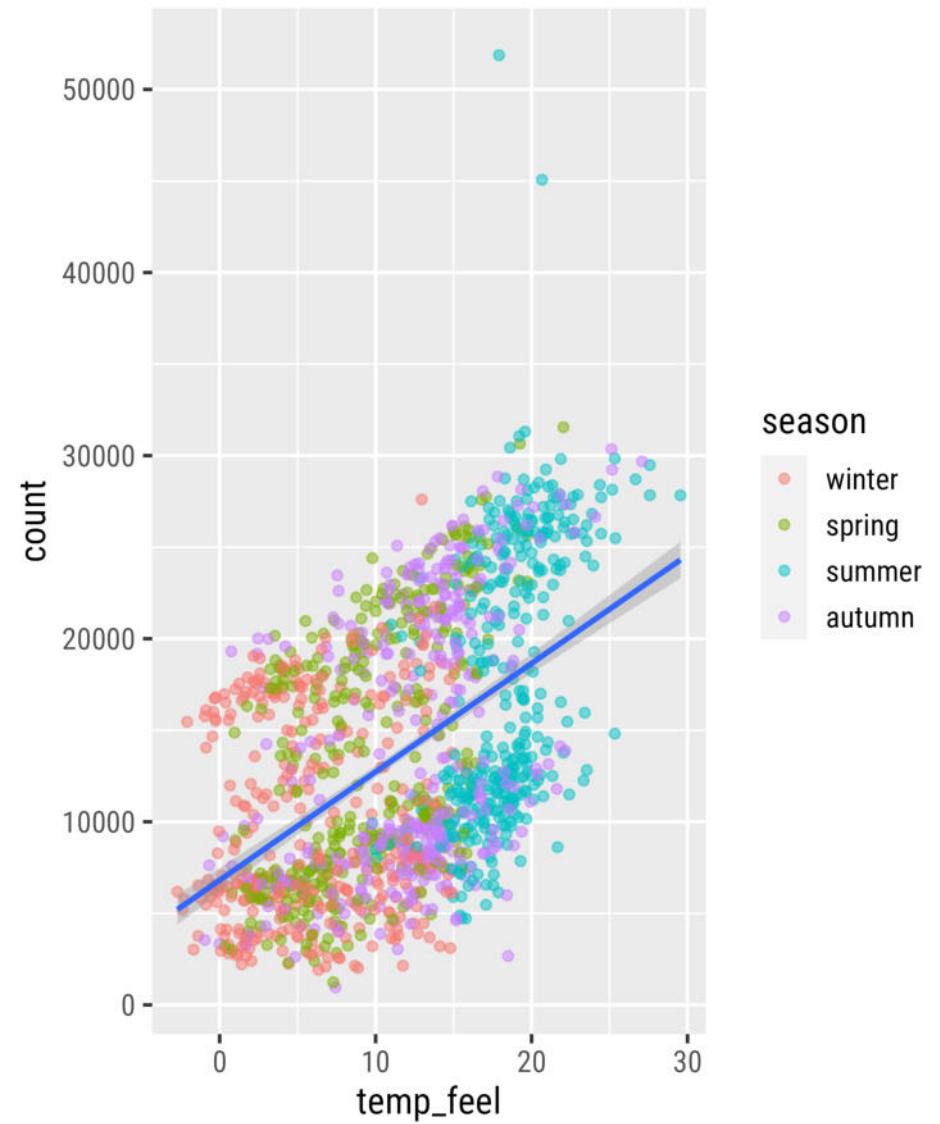
Global Color Encoding

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = season)  
5 ) +  
6   geom_point(  
7     alpha = .5  
8 ) +  
9   geom_smooth(  
10    method = "lm"  
11 )
```



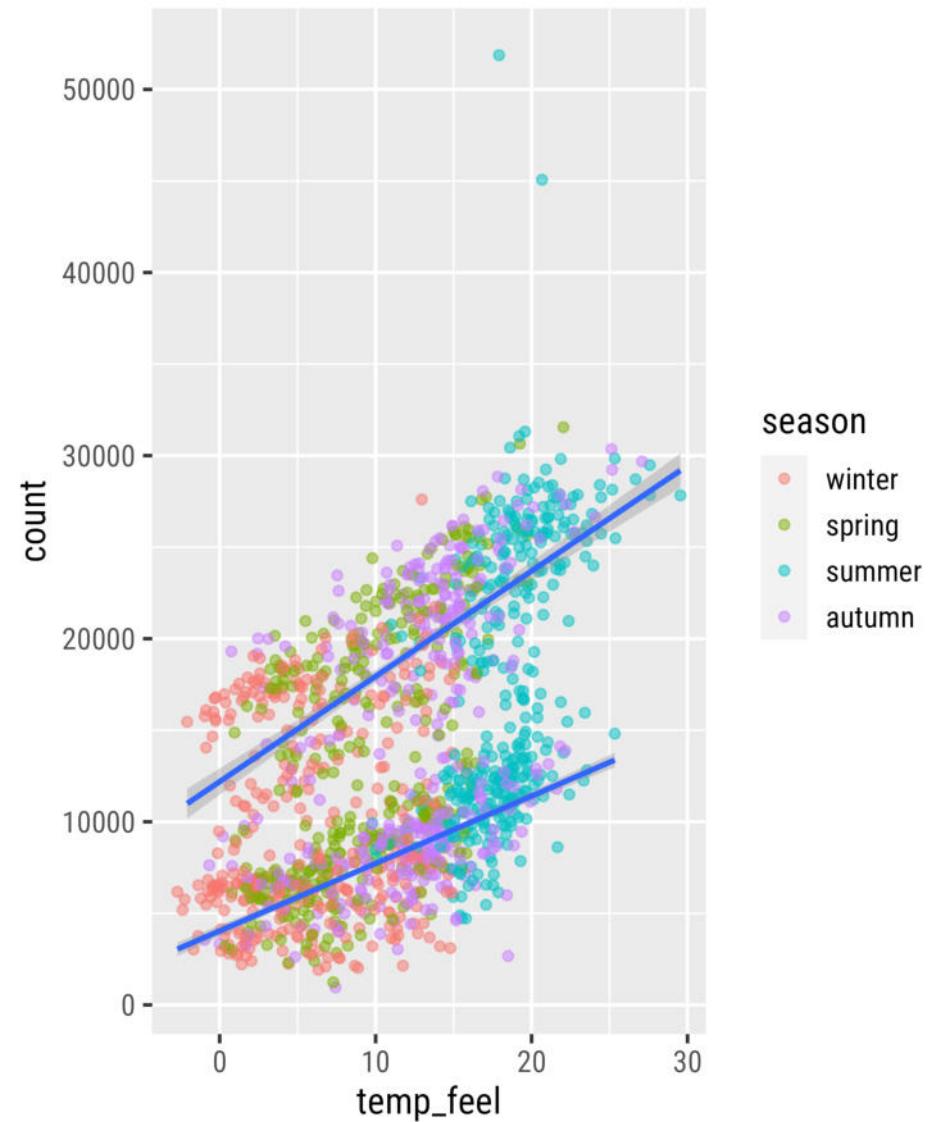
Local Color Encoding

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count))  
4 ) +  
5   geom_point(  
6     aes(color = season),  
7     alpha = .5  
8 ) +  
9   geom_smooth(  
10    method = "lm"  
11 )
```



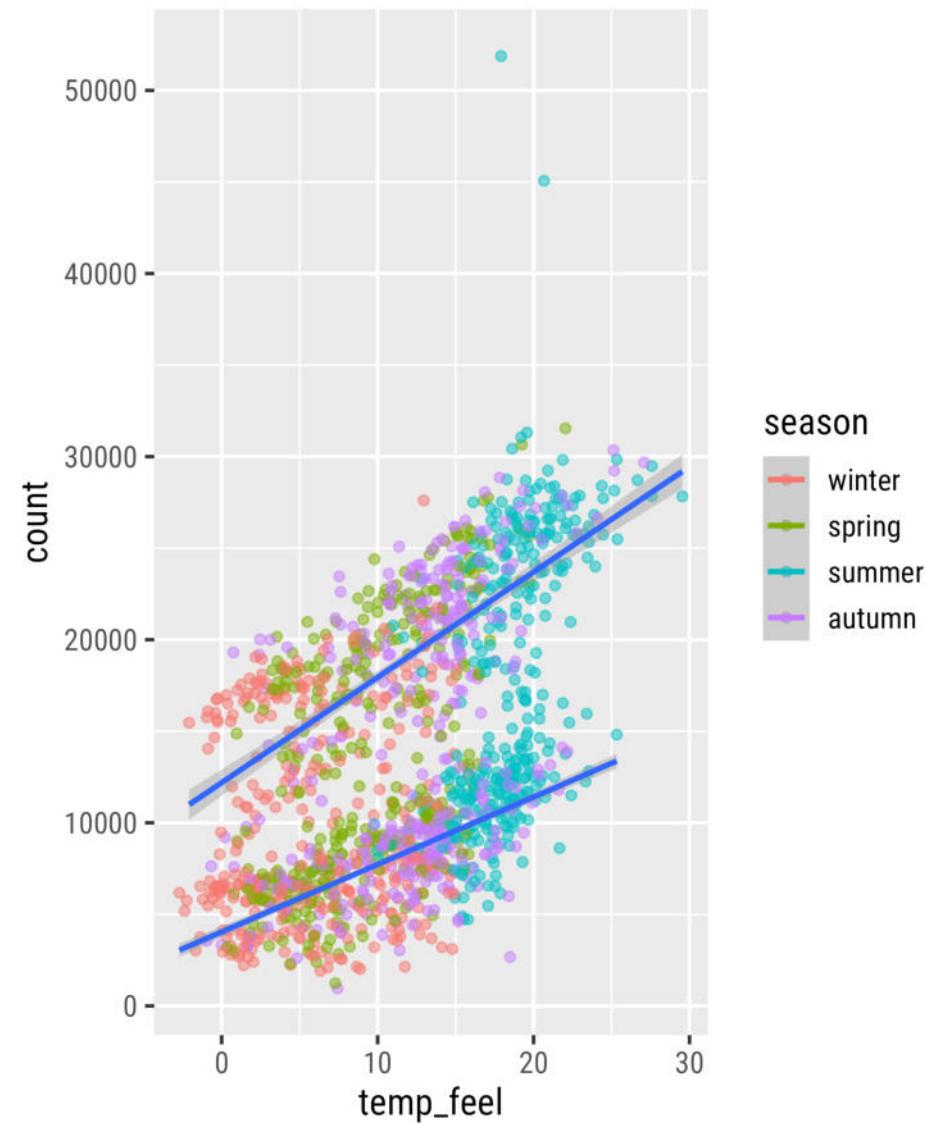
The `group` Aesthetic

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count))  
4 ) +  
5   geom_point(  
6     aes(color = season),  
7     alpha = .5  
8 ) +  
9   geom_smooth(  
10    aes(group = day_night),  
11    method = "lm"  
12 )
```



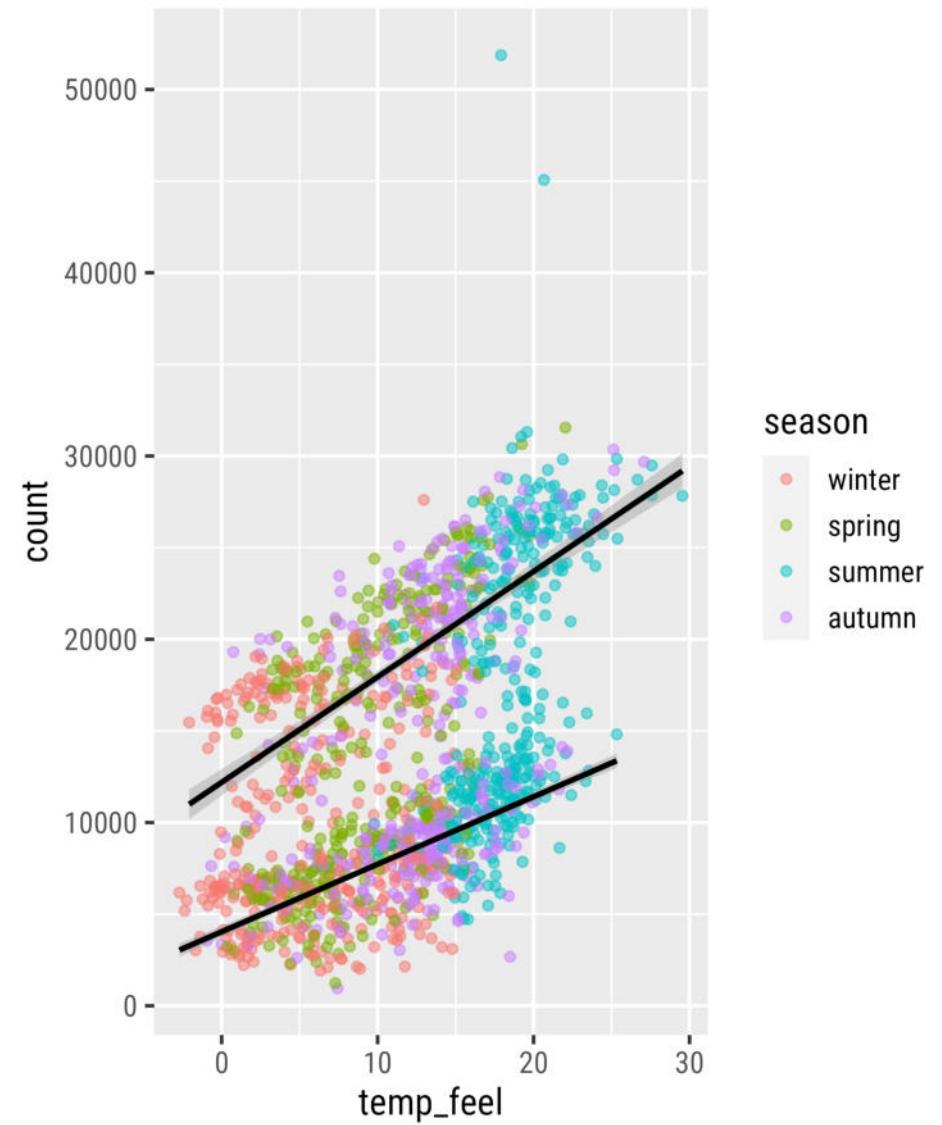
Set Both as Global Aesthetics

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = season,  
5       group = day_night)  
6 ) +  
7   geom_point(  
8     alpha = .5  
9   ) +  
10  geom_smooth(  
11    method = "lm"  
12 )
```



Overwrite Global Aesthetics

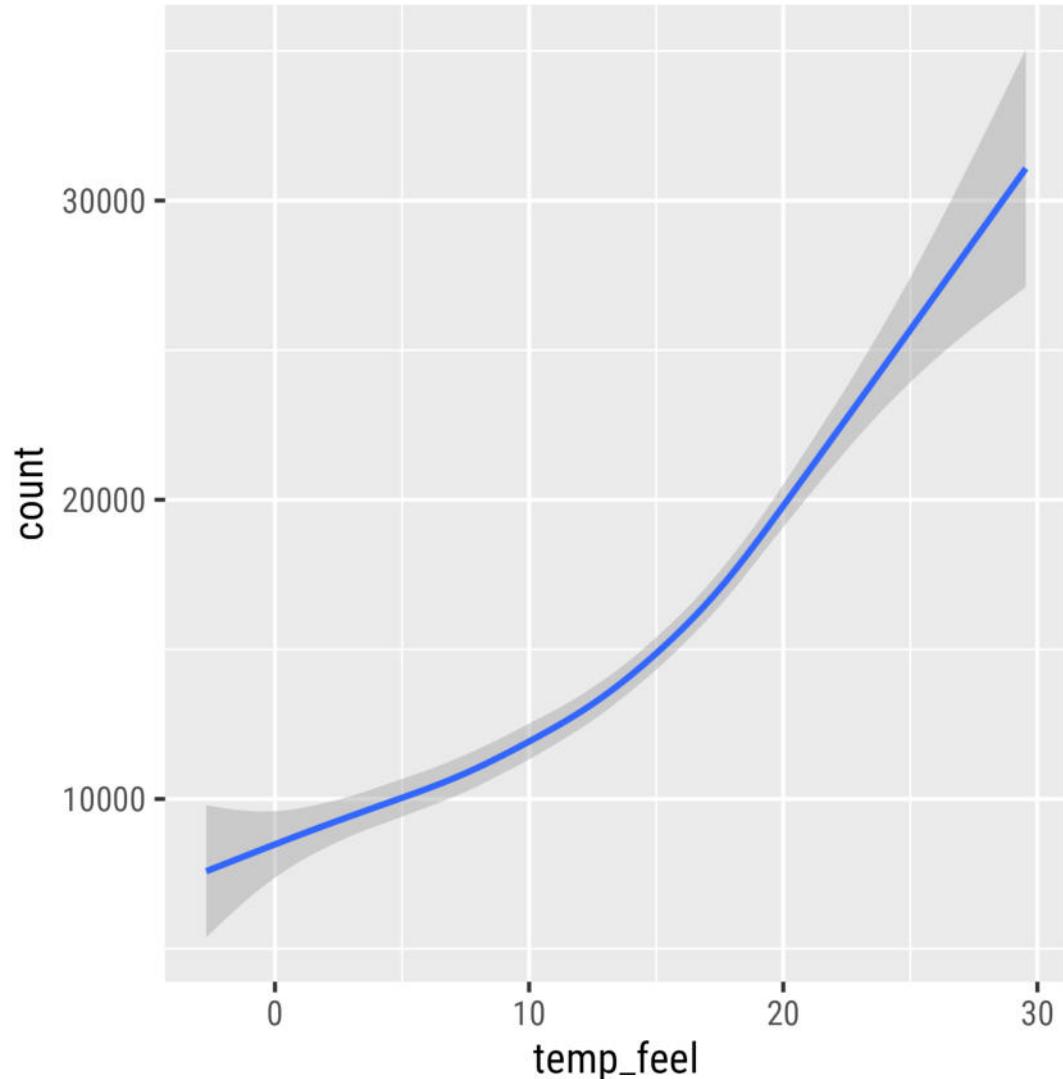
```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = season,  
5       group = day_night)  
6 ) +  
7   geom_point(  
8     alpha = .5  
9   ) +  
10  geom_smooth(  
11    method = "lm",  
12    color = "black"  
13 )
```



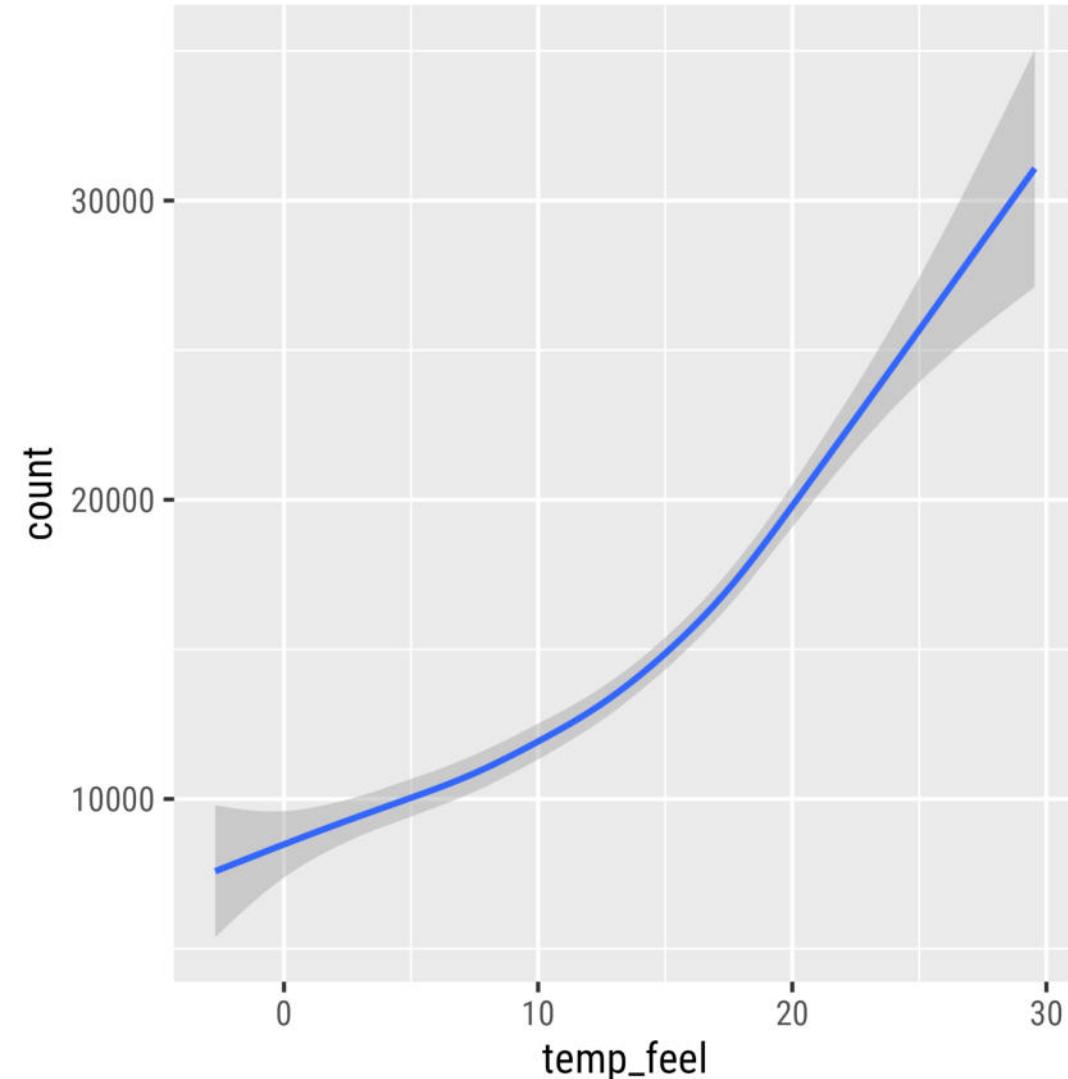
Statistical Layers

``stat_*`() and ``geom_*`()

```
1 ggplot(bikes, aes(x = temp_feel, y = count)) +  
2   stat_smooth(geom = "smooth")
```

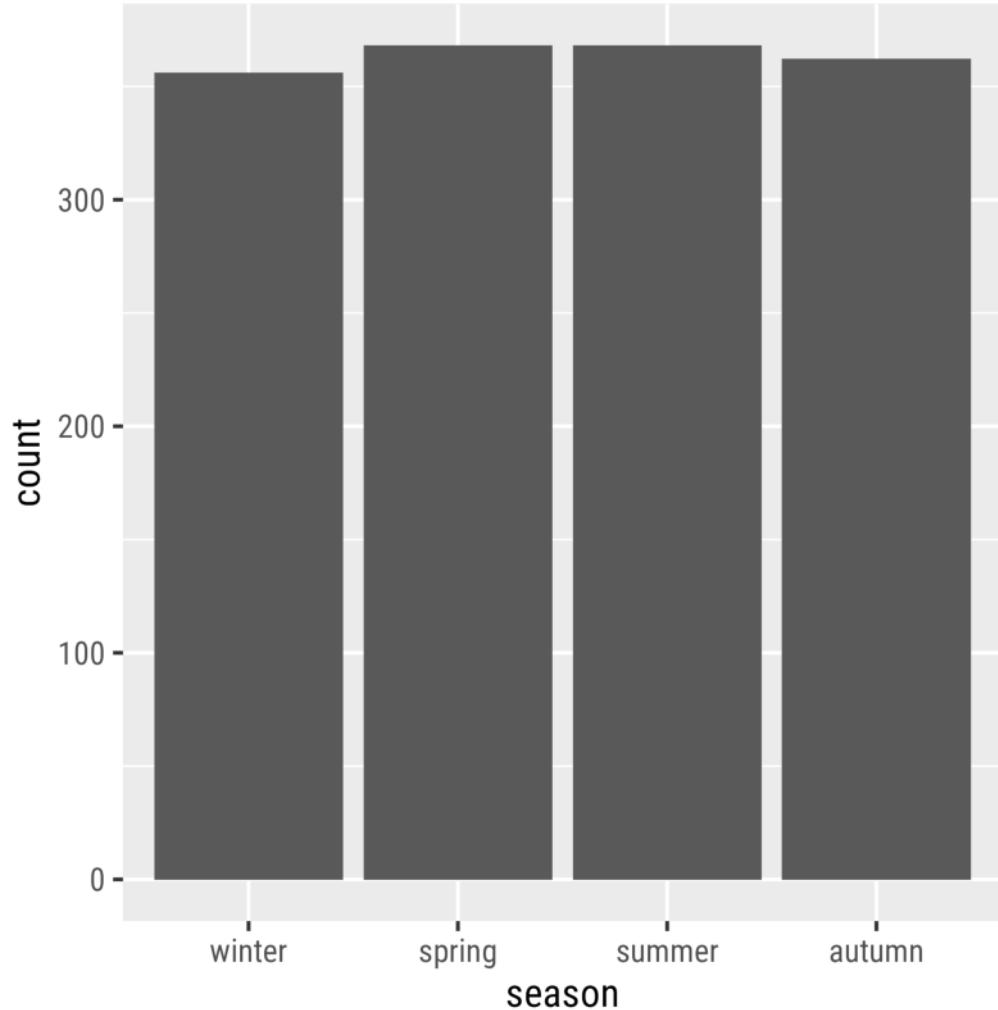


```
1 ggplot(bikes, aes(x = temp_feel, y = count)) +  
2   geom_smooth(stat = "smooth")
```

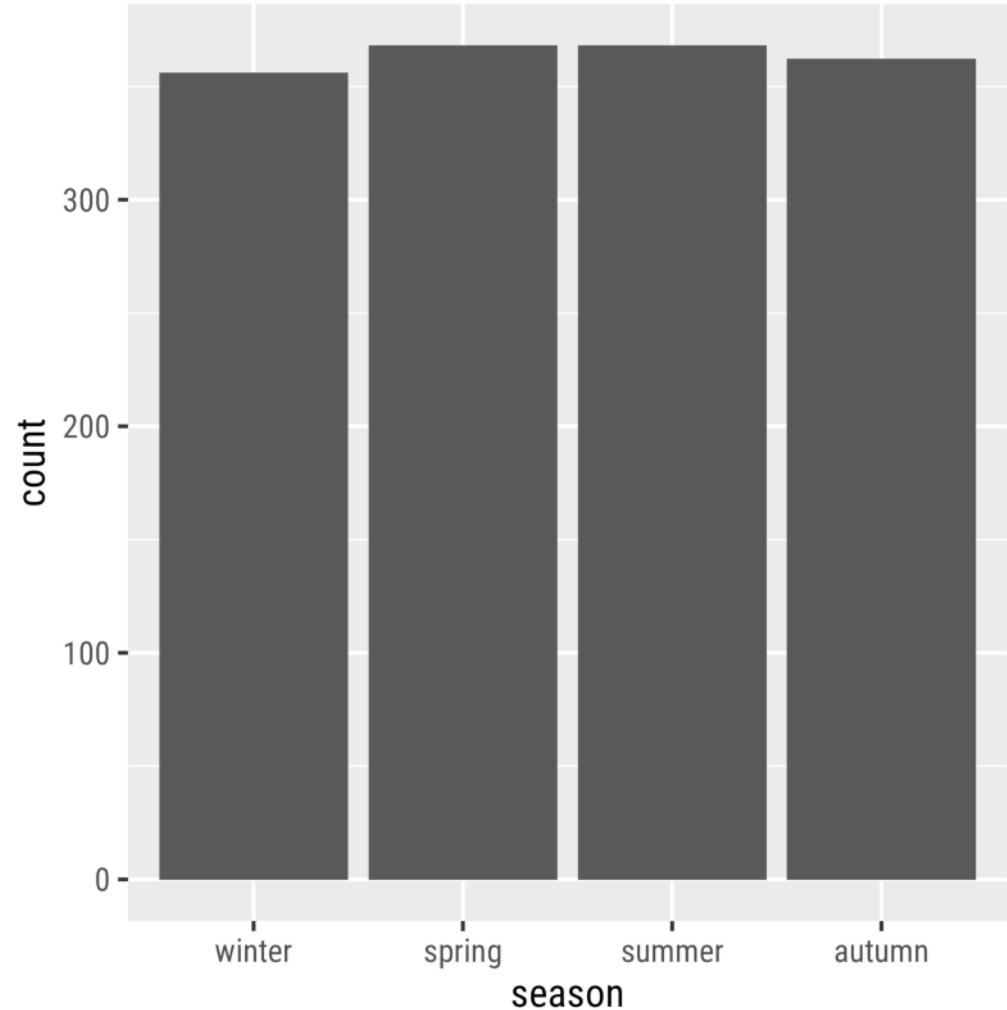


``stat_*`() and ``geom_*`()

```
1 ggplot(bikes, aes(x = season)) +  
2   stat_count(geom = "bar")
```

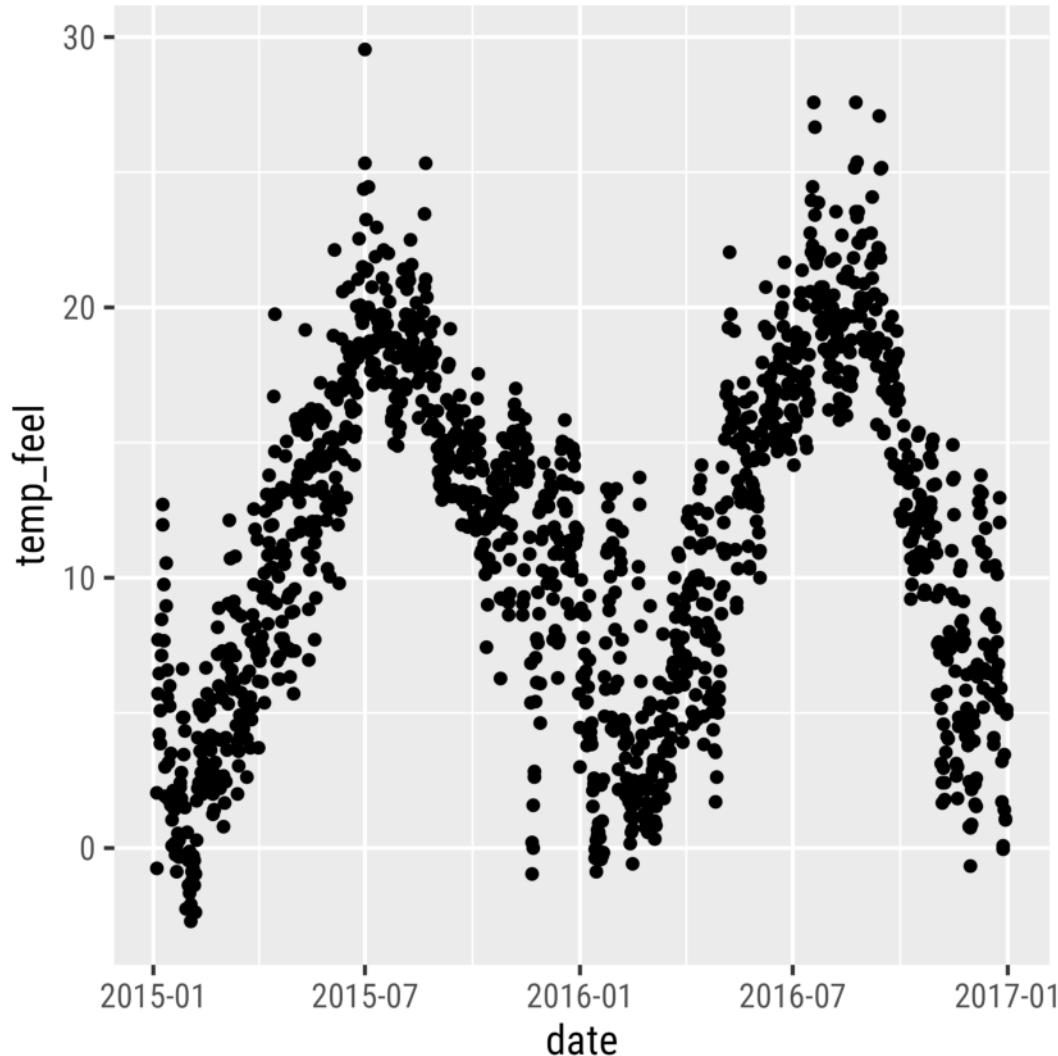


```
1 ggplot(bikes, aes(x = season)) +  
2   geom_bar(stat = "count")
```

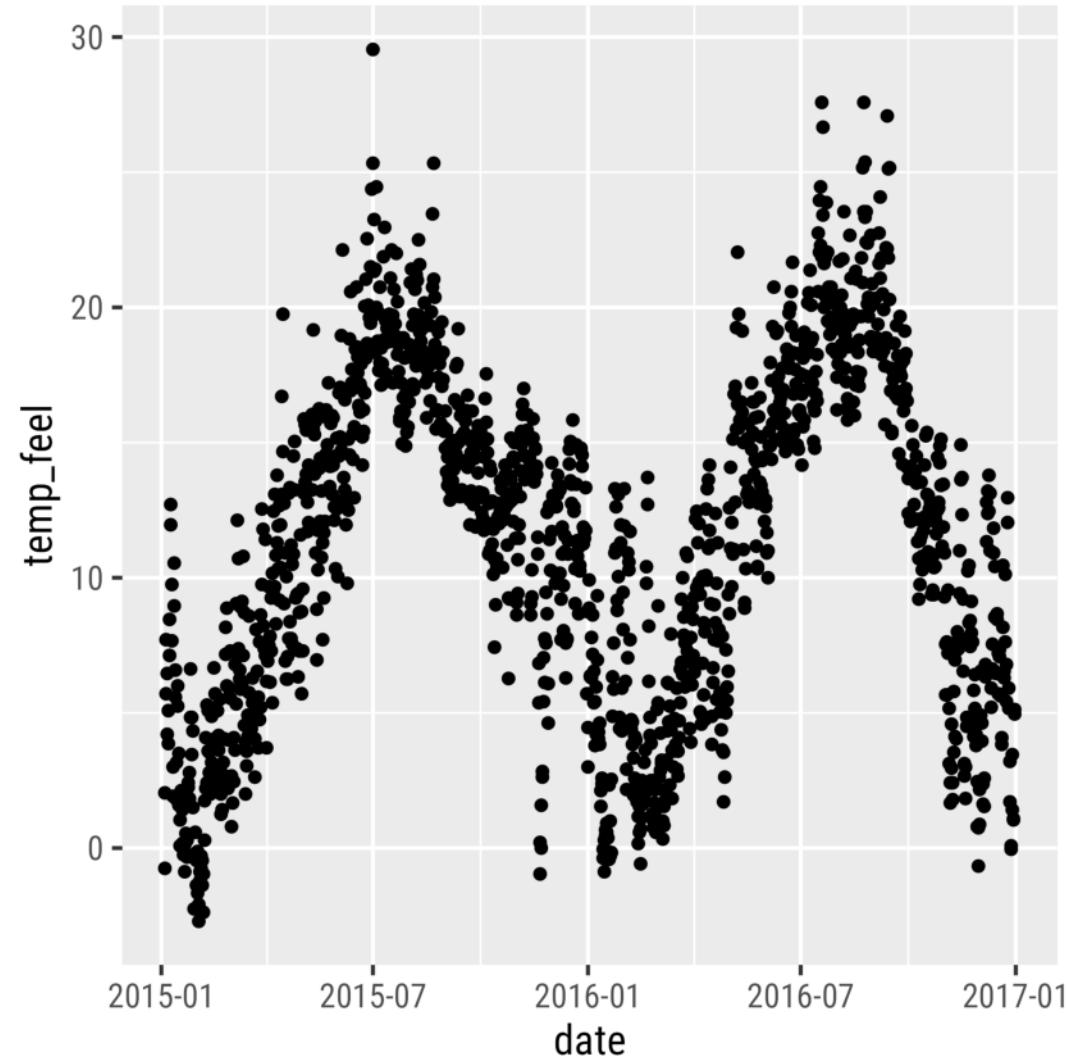


``stat_`()` and ``geom_`()`

```
1 ggplot(bikes, aes(x = date, y = temp_feel)) +  
2   stat_identity(geom = "point")
```

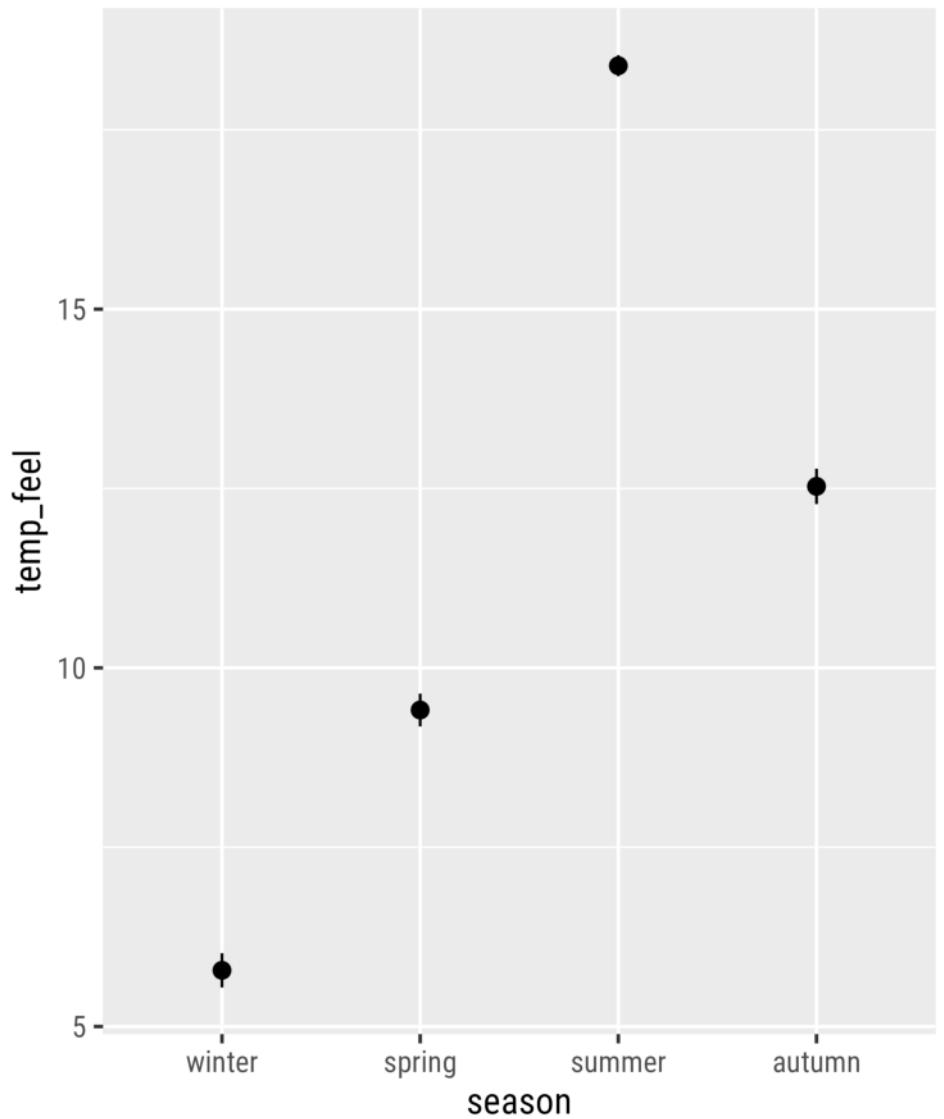


```
1 ggplot(bikes, aes(x = date, y = temp_feel)) +  
2   geom_point(stat = "identity")
```



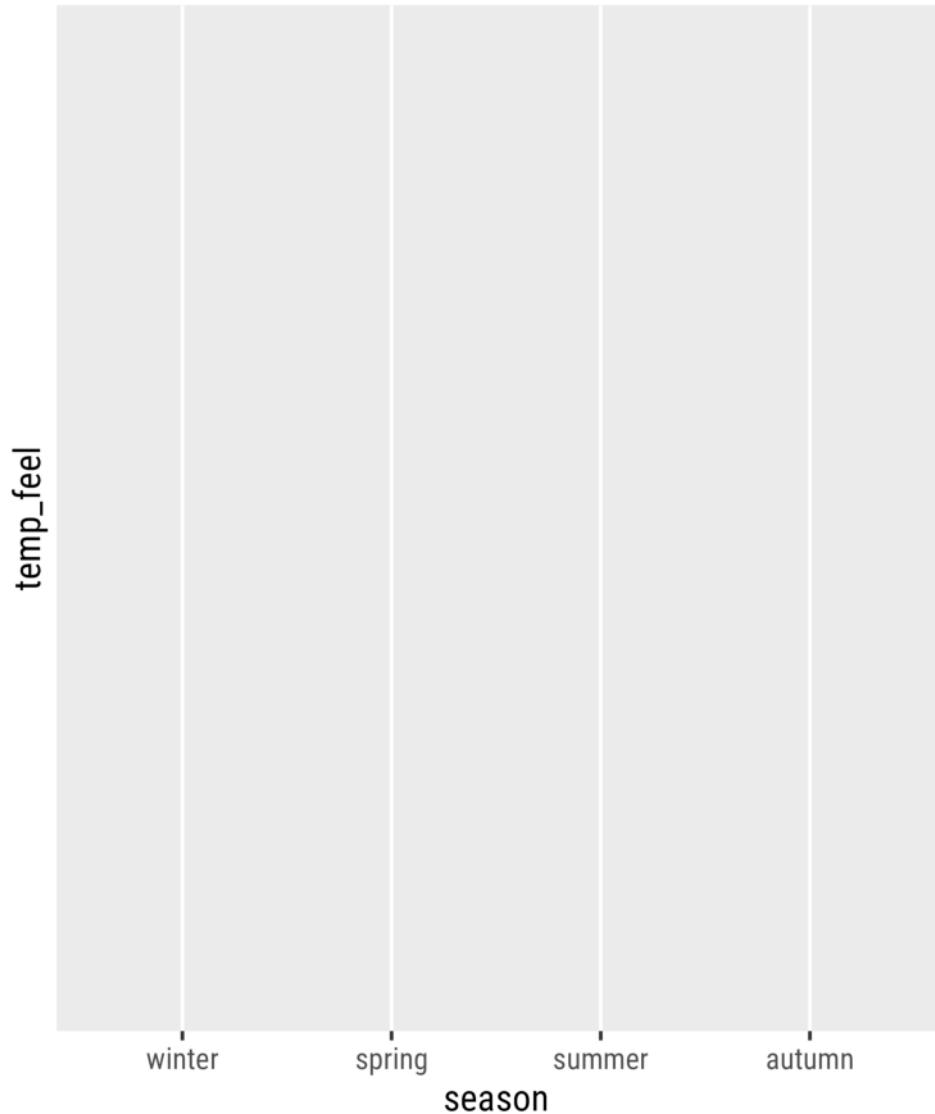
Statistical Summaries

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



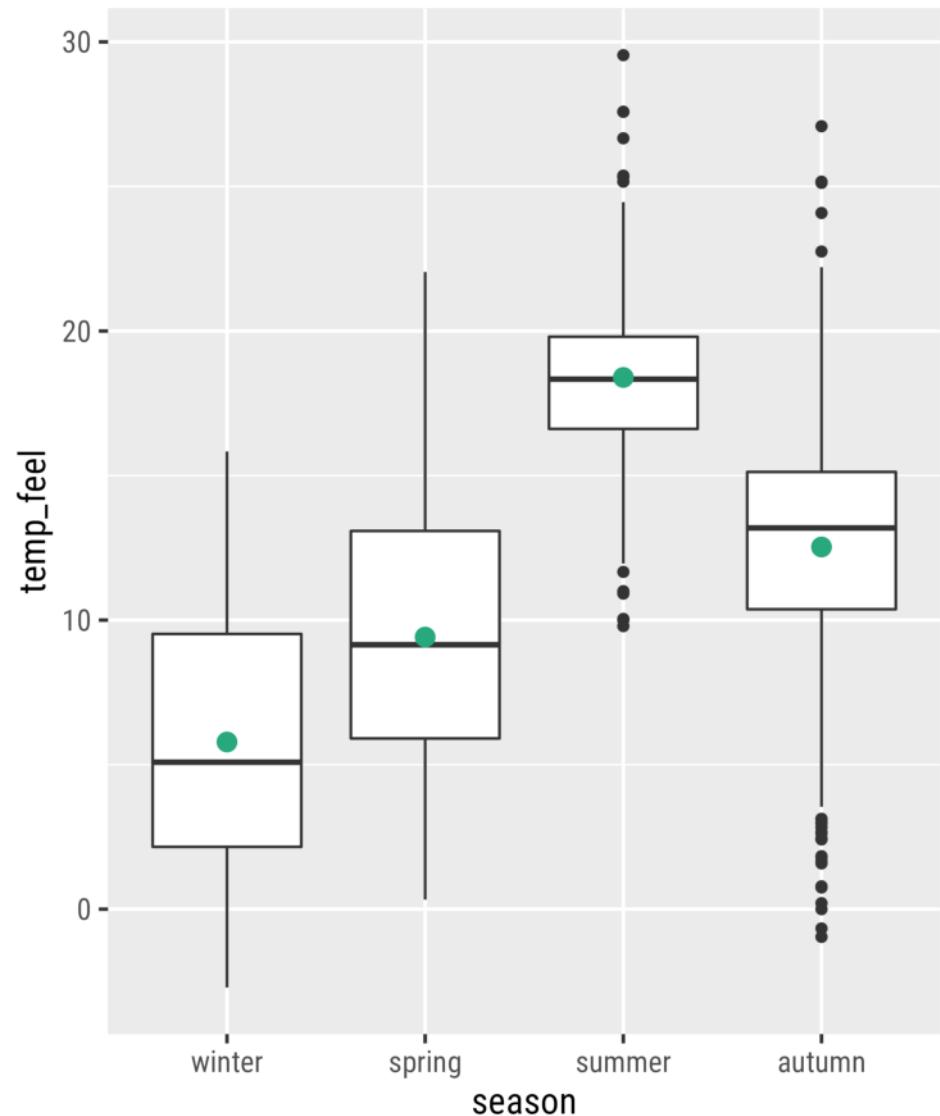
Statistical Summaries

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = temp_feel)  
4 ) +  
5 stat_summary(  
6   fun = mean_se, ## the default  
7   geom = "pointrange" ## the default  
8 )
```



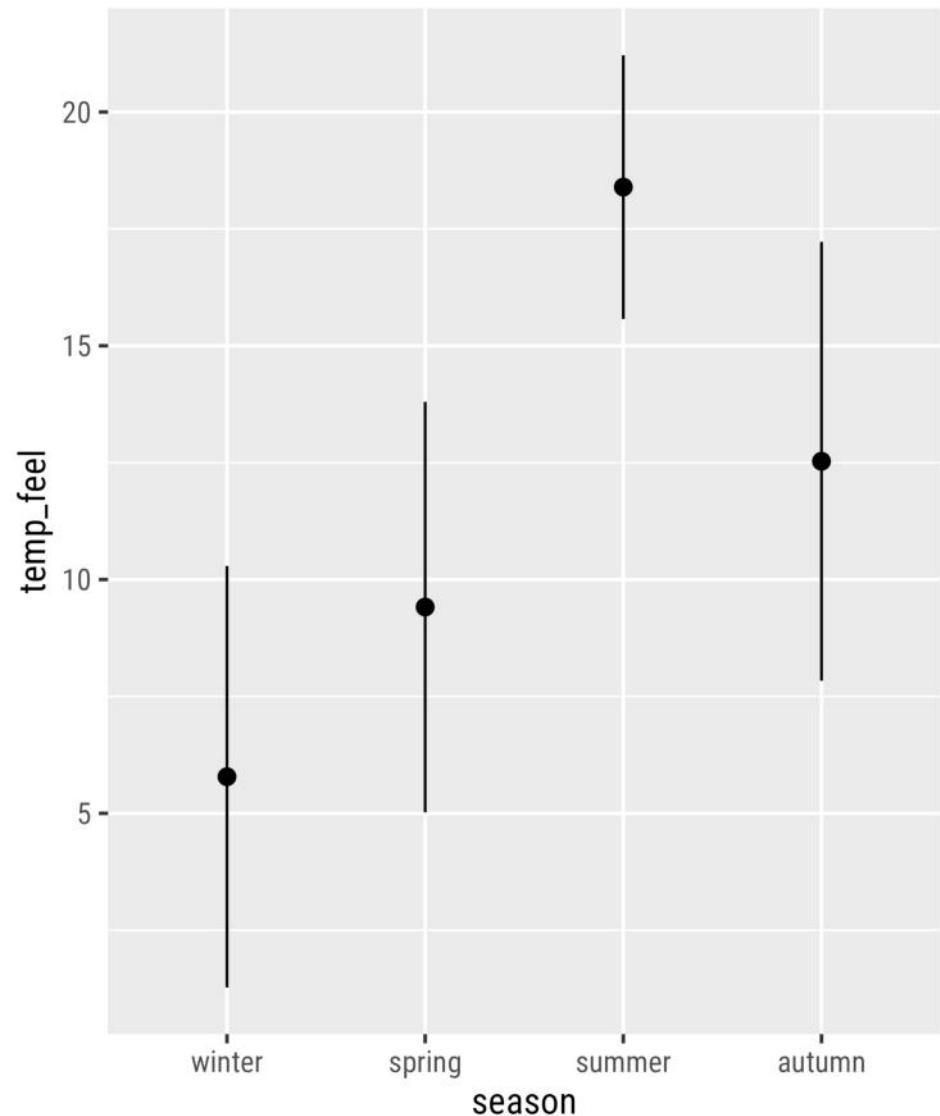
Statistical Summaries

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = temp_feel)  
4 ) +  
5 geom_boxplot() +  
6 stat_summary(  
7   fun = mean,  
8   geom = "point",  
9   color = "#28a87d",  
10  size = 3  
11 )
```



Statistical Summaries

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = temp_feel)  
4 ) +  
5   stat_summary(  
6     fun = mean,  
7     fun.max = function(y) mean(y) + sd(y),  
8     fun.min = function(y) mean(y) - sd(y)  
9 )
```



Extending a ggplot

Store a ggplot as Object

```
1 g <-  
2   ggplot(  
3     bikes,  
4     aes(x = temp_feel, y = count,  
5           color = season,  
6           group = day_night)  
7   ) +  
8   geom_point(  
9     alpha = .5  
10  ) +  
11  geom_smooth(  
12    method = "lm",  
13    color = "black"  
14  )  
15  
16 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

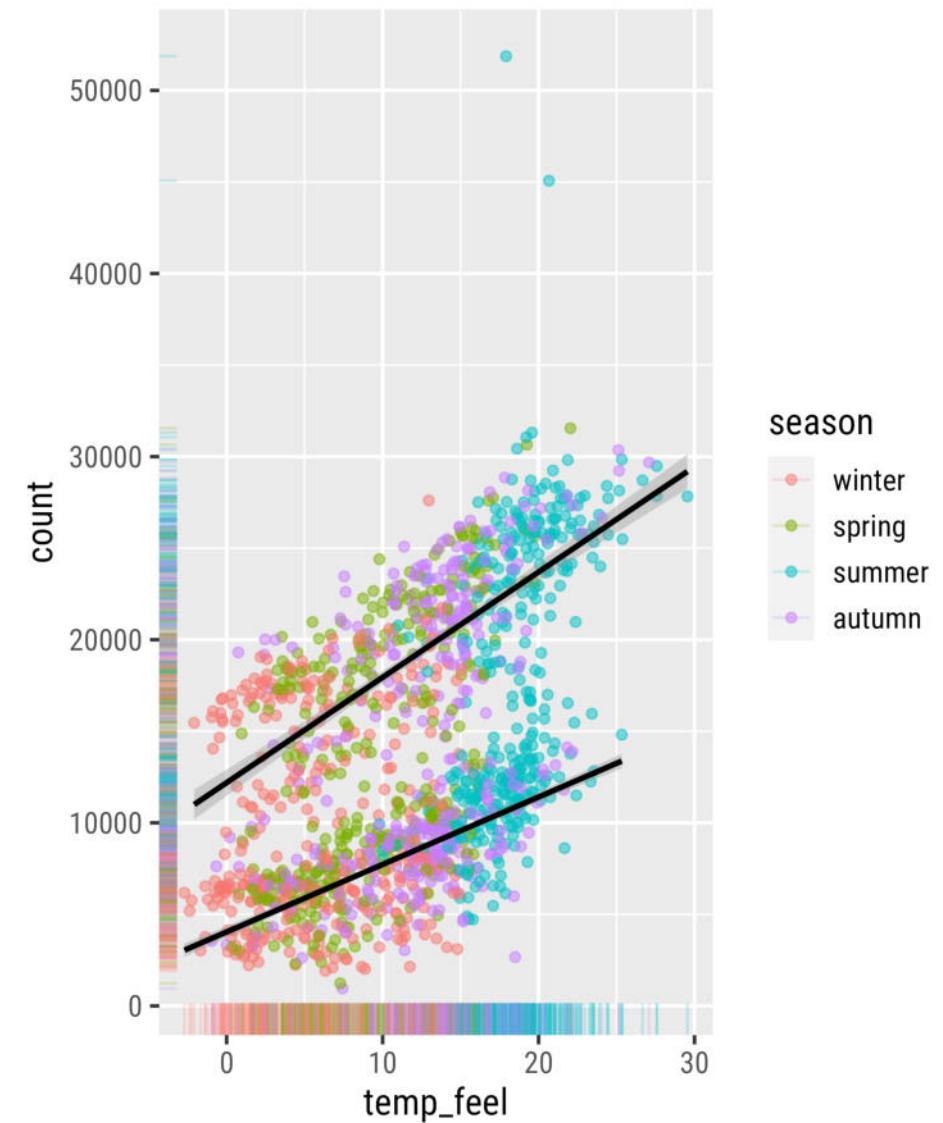
```
1 g$mapping
```

Aesthetic mapping:

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

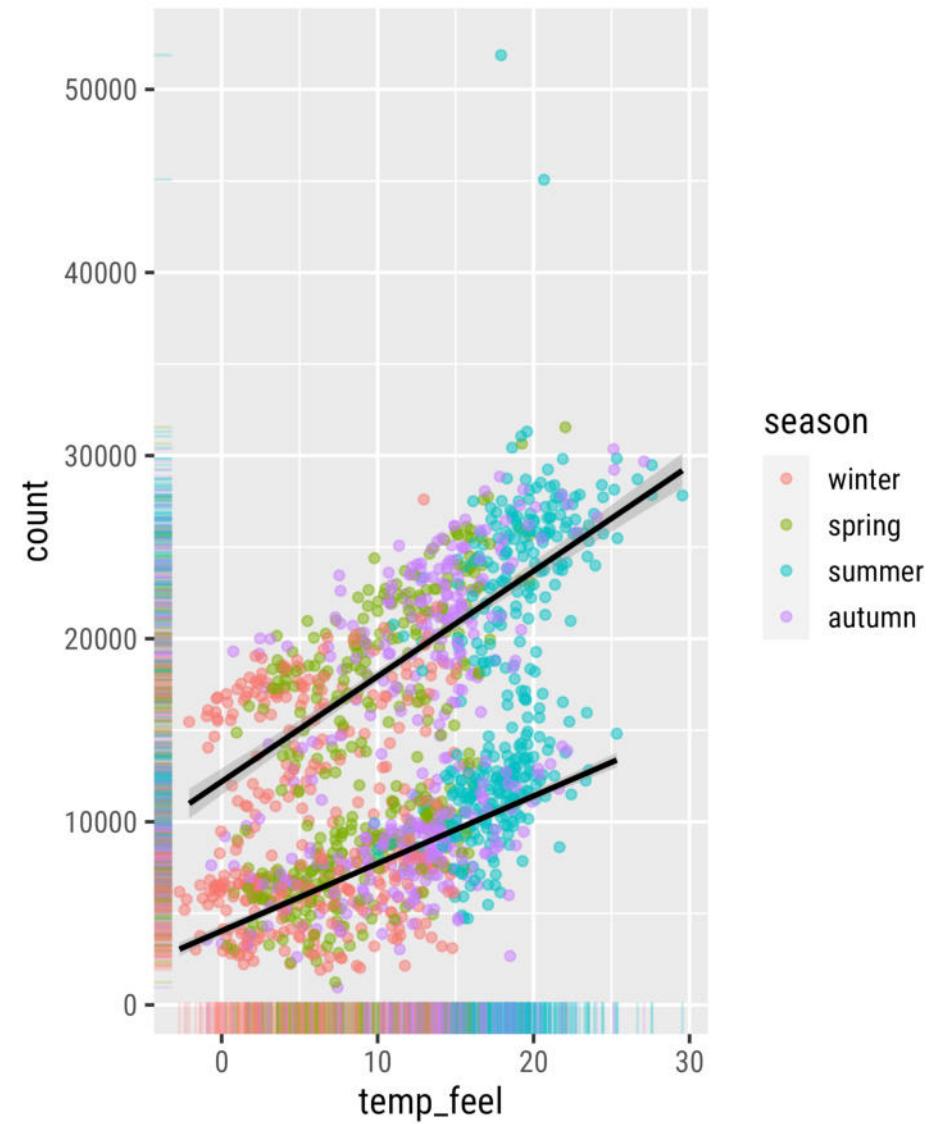
Extend a ggplot Object: Add Layers

```
1 g +  
2   geom_rug(  
3     alpha = .2  
4   )
```



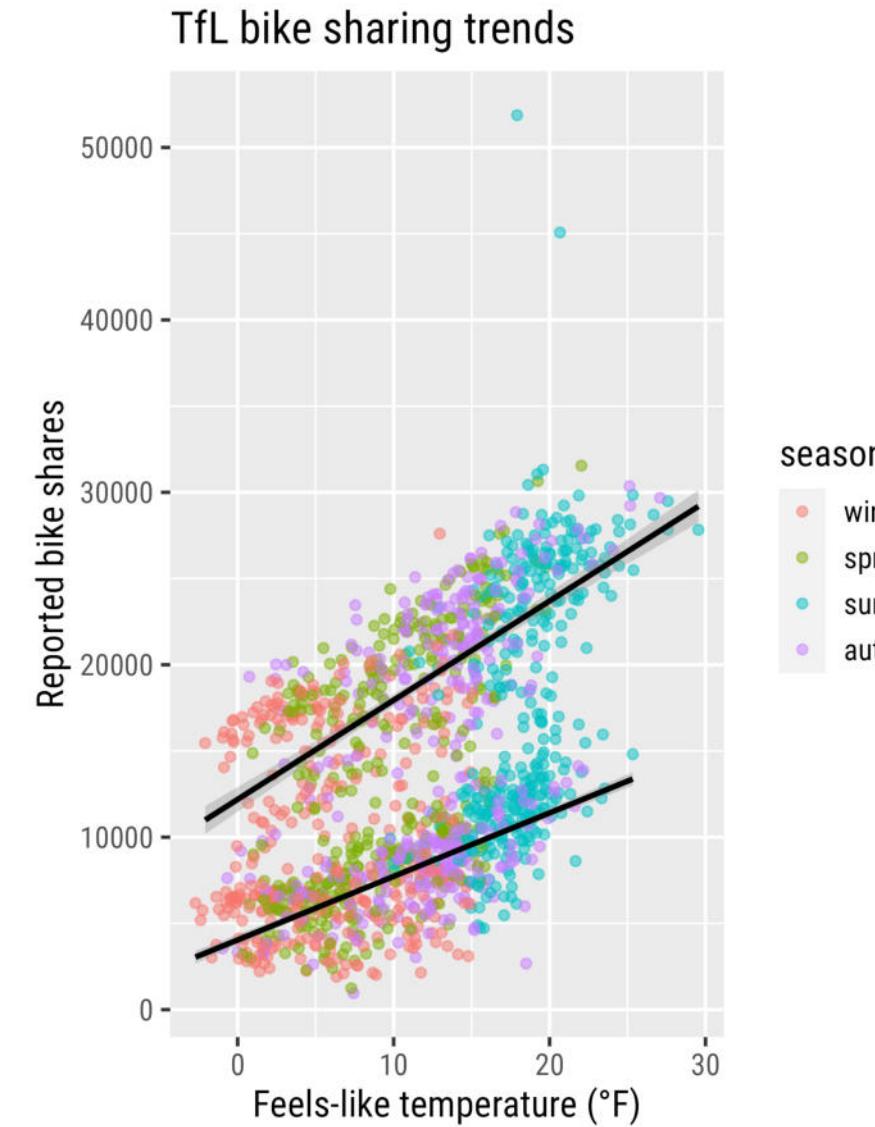
Remove a Layer from the Legend

```
1 g +
2   geom_rug(
3     alpha = .2,
4     show.legend = FALSE
5   )
```



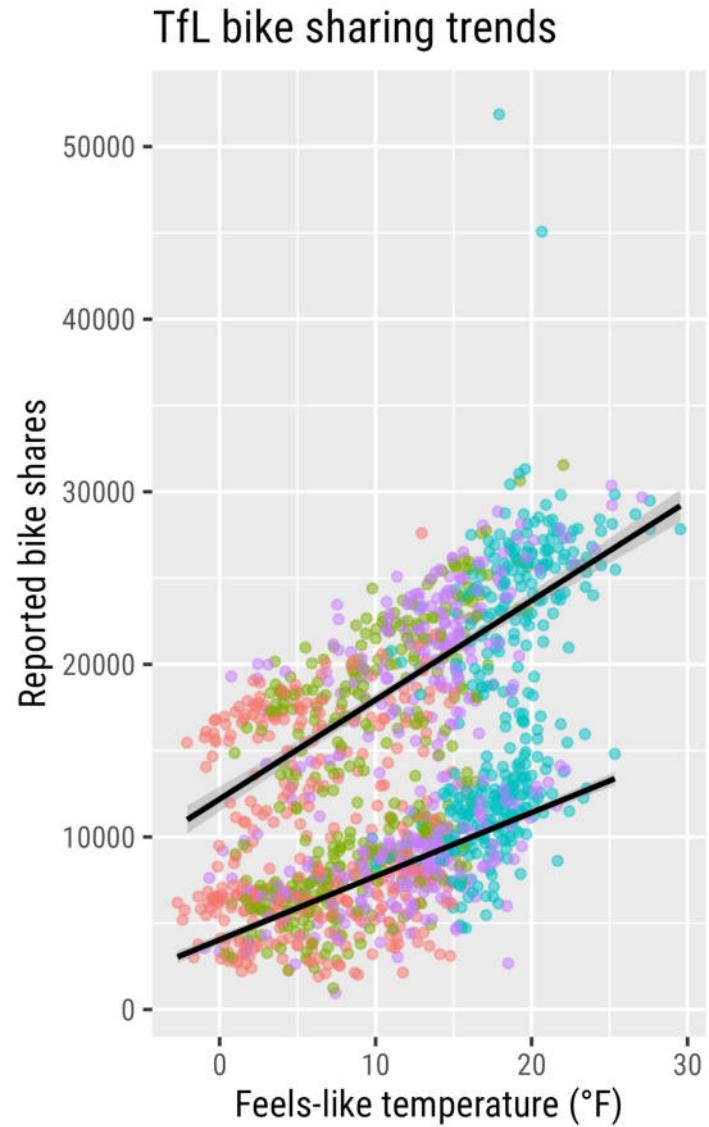
Extend a ggplot Object: Add Labels

```
1 g +  
2   xlab("Feels-like temperature (°F)") +  
3   ylab("Reported bike shares") +  
4   ggtitle("TfL bike sharing trends")
```



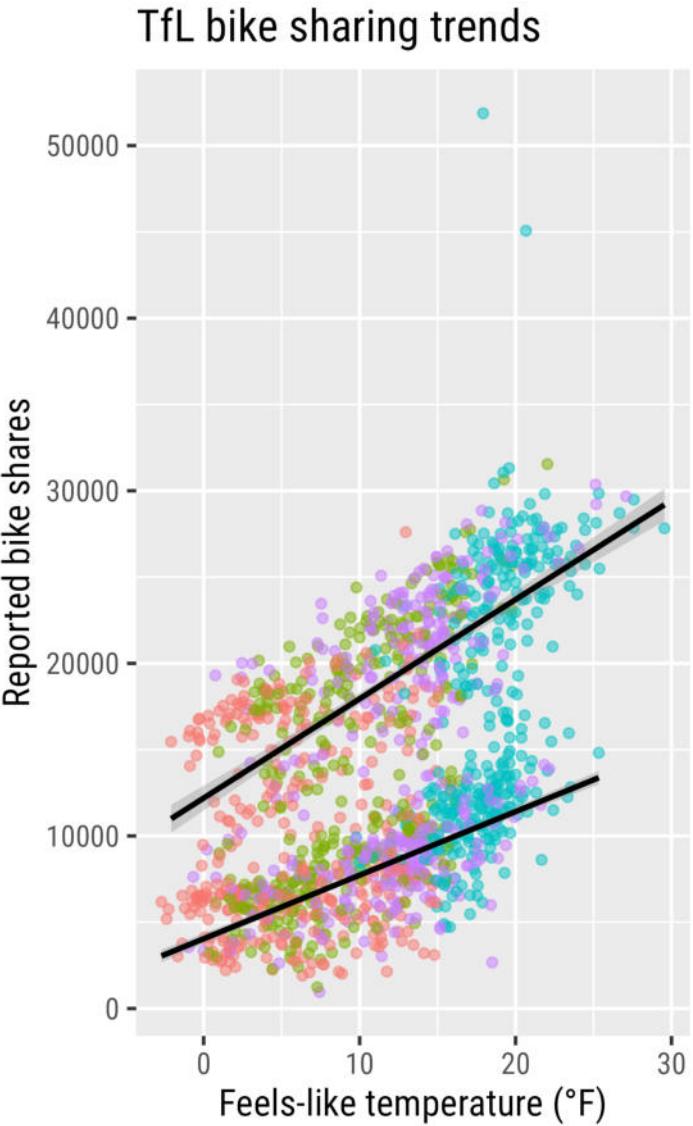
Extend a ggplot Object: Add Labels

```
1 g +  
2   labs(  
3     x = "Feels-like temperature (°F)",  
4     y = "Reported bike shares",  
5     title = "TfL bike sharing trends"  
6   )
```



Extend a ggplot Object: Add Labels

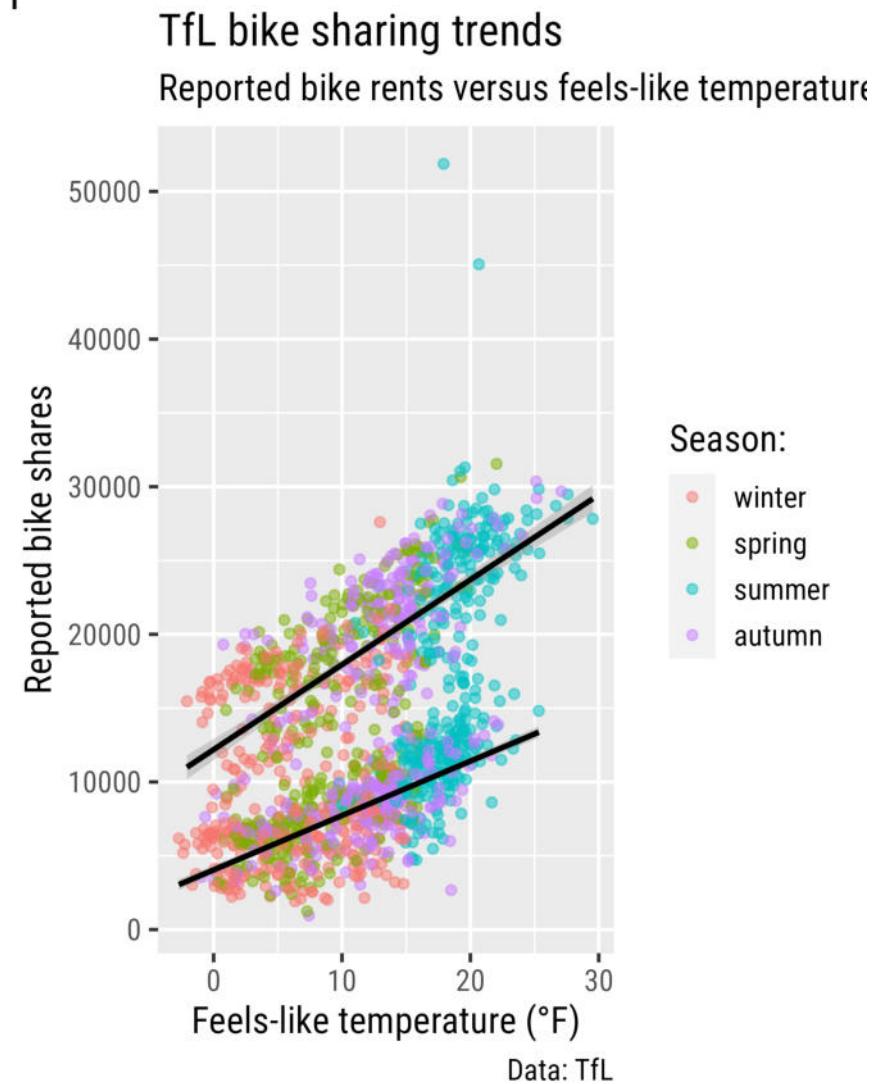
```
1 g <- g +
2   labs(
3     x = "Feels-like temperature (°F)",
4     y = "Reported bike shares",
5     title = "TfL bike sharing trends",
6     color = "Season:"
7   )
8
9 gg
```



Extend a ggplot Object: Add Labels

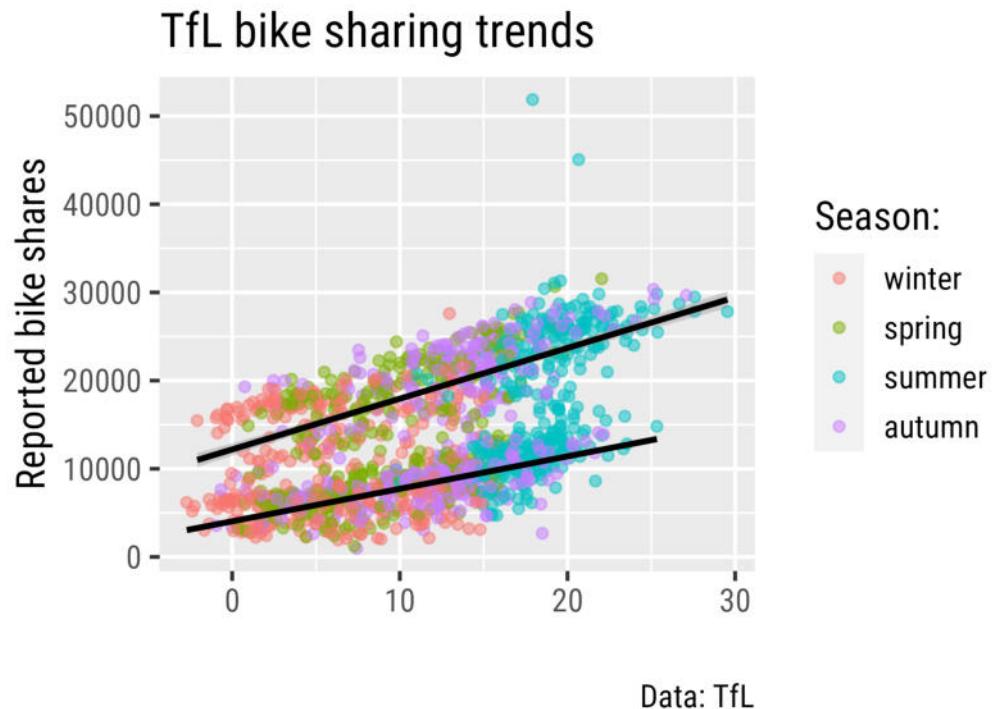
```
1 g +  
2   labs(  
3     x = "Feels-like temperature (°F)",  
4     y = "Reported bike shares",  
5     title = "TfL bike sharing trends",  
6     subtitle = "Reported bike rents versus feels-like temperature",  
7     caption = "Data: TfL",  
8     color = "Season:",  
9     tag = "Fig. 1"  
10 )
```

Fig. 1

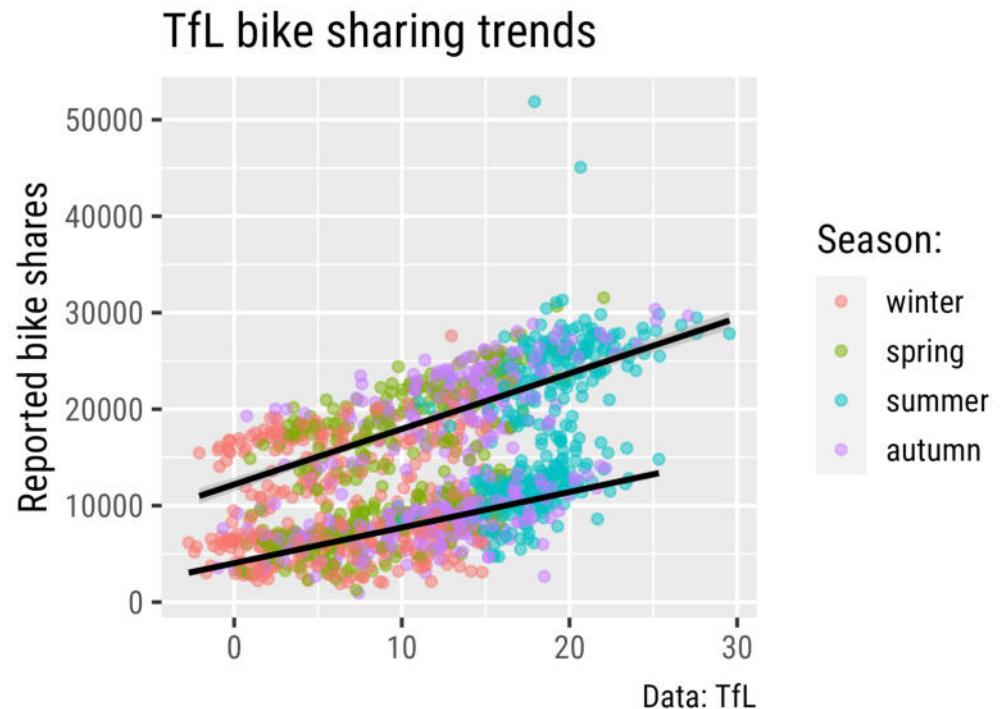


Extend a ggplot Object: Add Labels

```
1 g +  
2   labs(  
3     x = "",  
4     caption = "Data: TfL"  
5   )
```



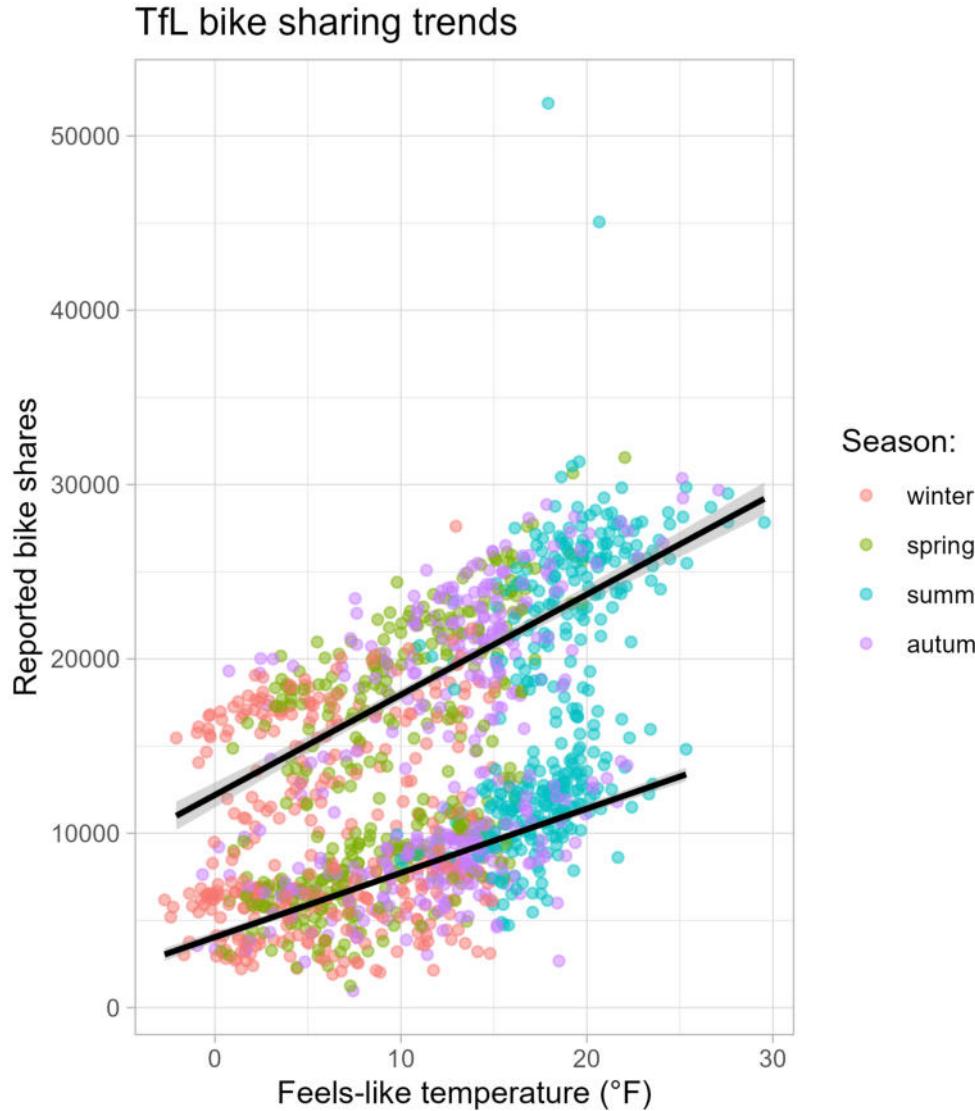
```
1 g +  
2   labs(  
3     x = NULL,  
4     caption = "Data: TfL"  
5   )
```



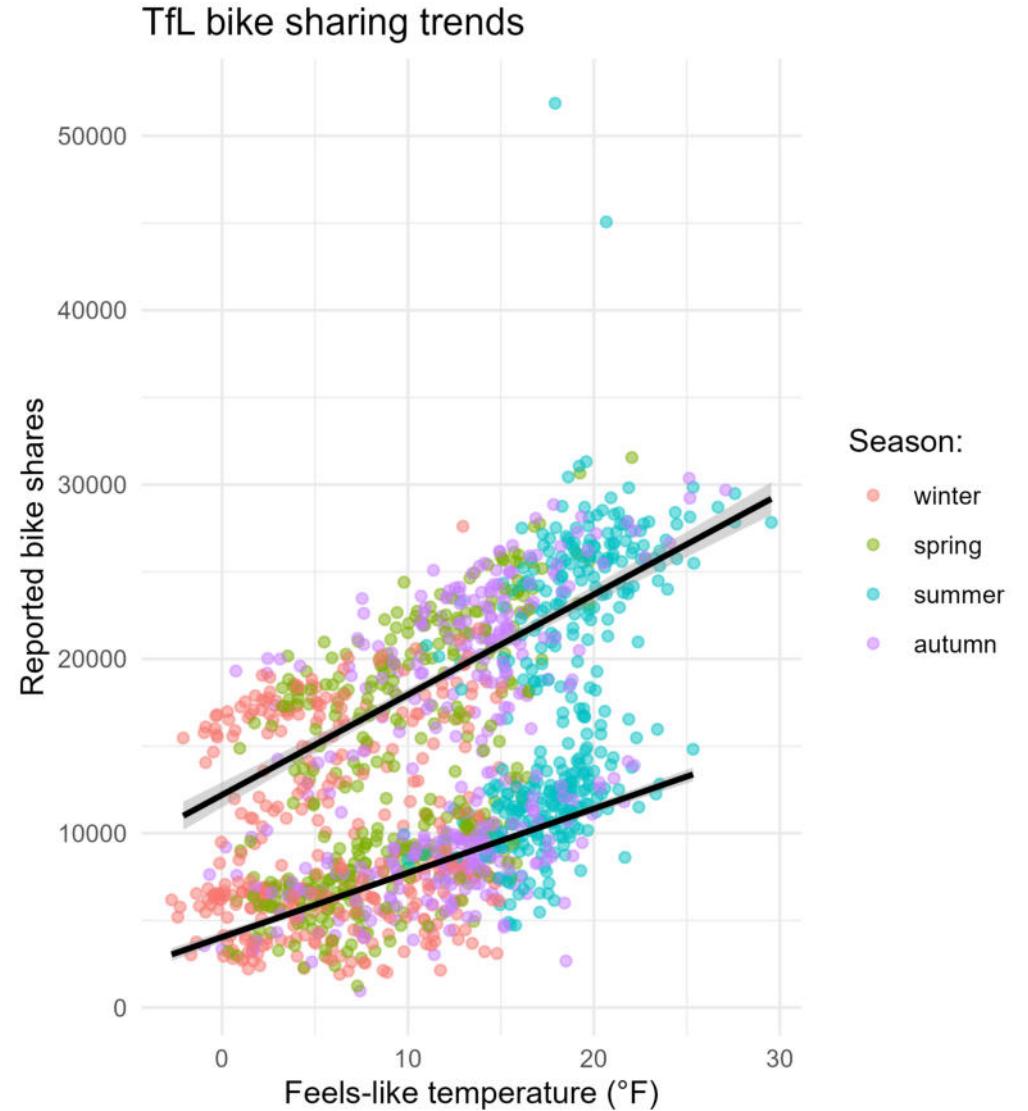
A Polished ggplot Example

Extend a ggplot Object: Themes

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

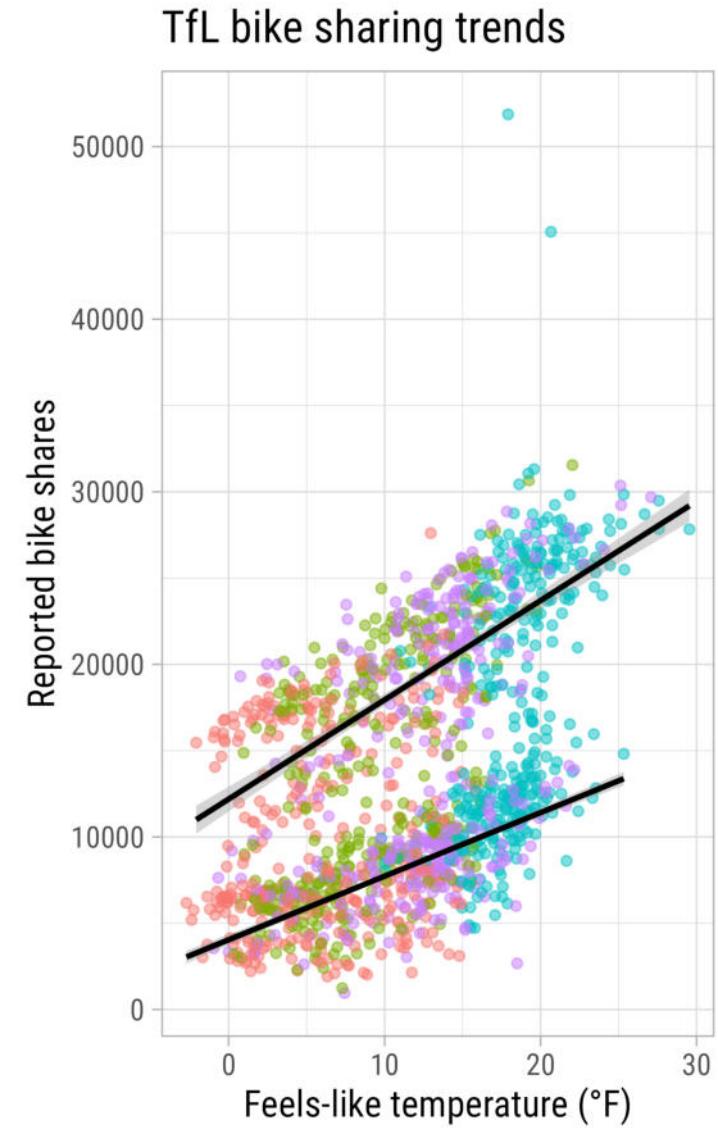


```
1 g + theme_minimal()
```



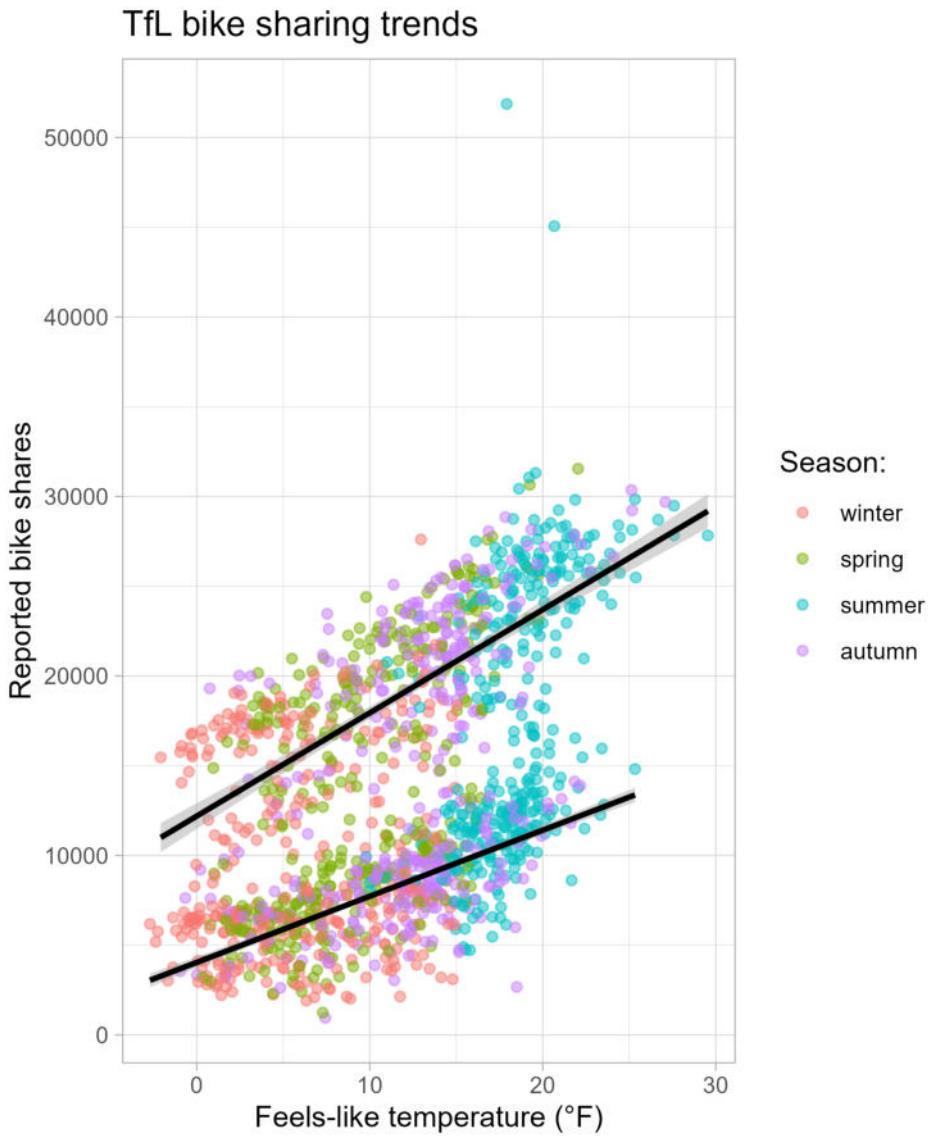
Change the Theme Base Settings

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



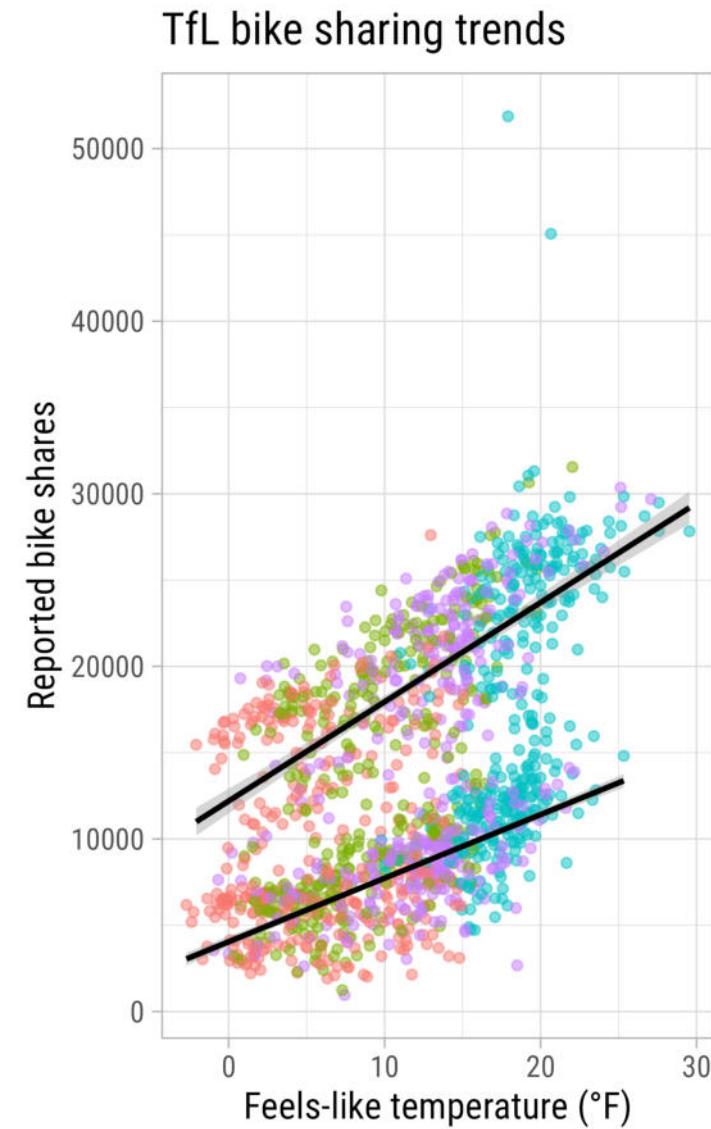
Set a Theme Globally

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



Change the Theme Base Settings

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



{systemfonts}

```
1 # install.packages("systemfonts")
2
3 library(systemfonts)
4
5 system_fonts() %>%
6   filter(str_detect(family, "Cabinet")) %>%
7   pull(name) %>%
8   sort()

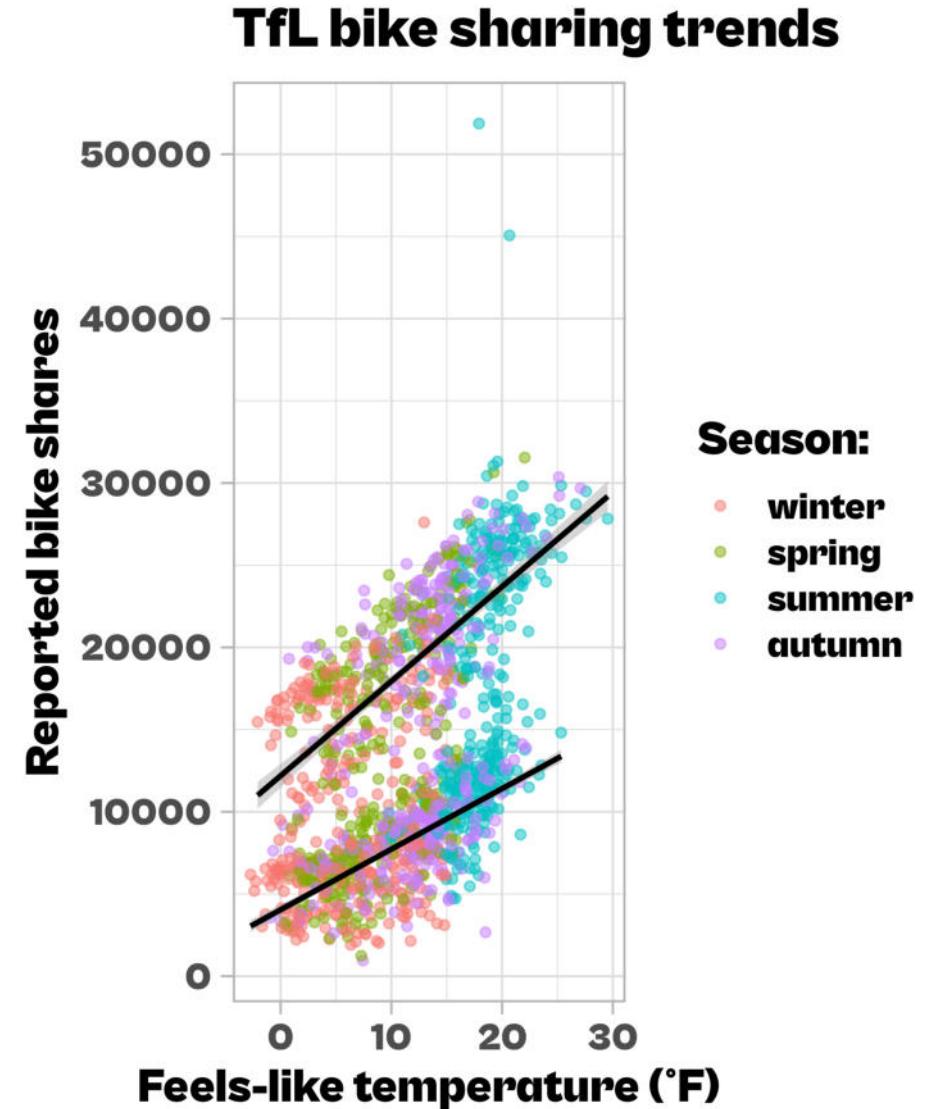
[1] "CabinetGrotesk-Black"      "CabinetGrotesk-Black"
[3] "CabinetGrotesk-Bold"       "CabinetGrotesk-Bold"
[5] "CabinetGrotesk-Extrabold"  "CabinetGrotesk-Extrabold"
[7] "CabinetGrotesk-Extralight" "CabinetGrotesk-Extralight"
[9] "CabinetGrotesk-Light"      "CabinetGrotesk-Light"
[11] "CabinetGrotesk-Medium"     "CabinetGrotesk-Medium"
[13] "CabinetGrotesk-Regular"    "CabinetGrotesk-Regular"
[15] "CabinetGrotesk-Thin"       "CabinetGrotesk-Thin"
```

{systemfonts}

```
1 register_variant(  
2     name = "Cabinet Grotesk Black",  
3     family = "Cabinet Grotesk",  
4     weight = "heavy",  
5     features = font_feature(letters = "stylistic")  
6 )
```

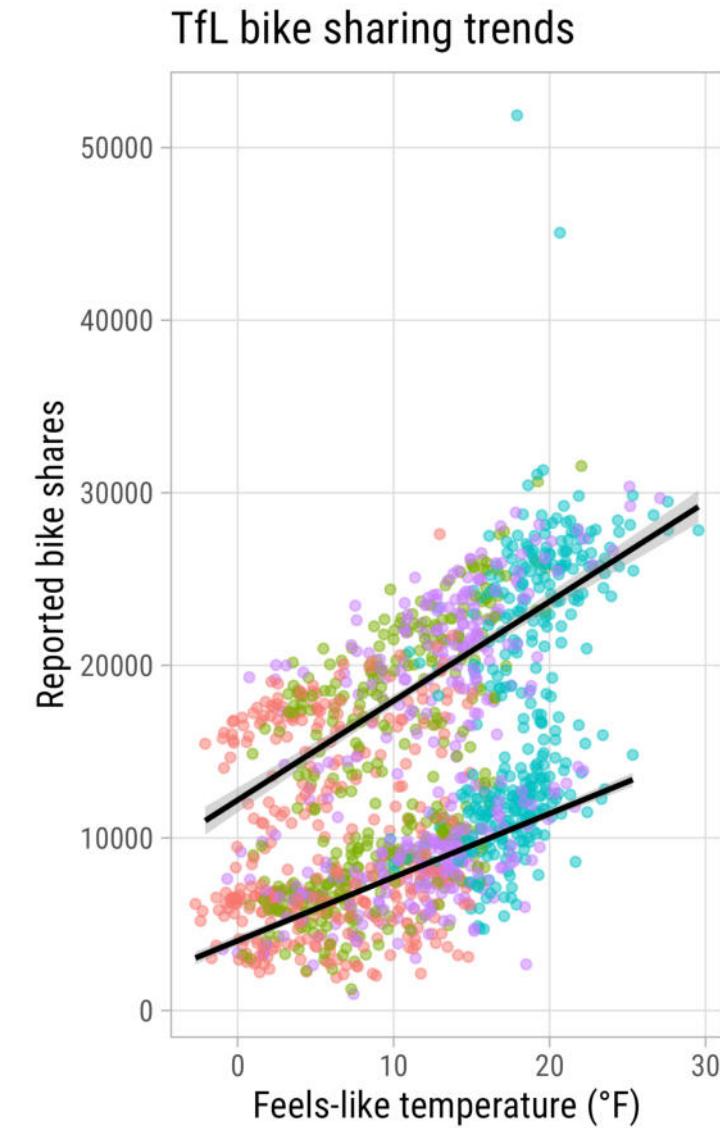
{systemfonts} + {ggplot2}

```
1 g +  
2   theme_light(  
3     base_size = 18,  
4     base_family = "Cabinet Grotesk Black"  
5   )
```



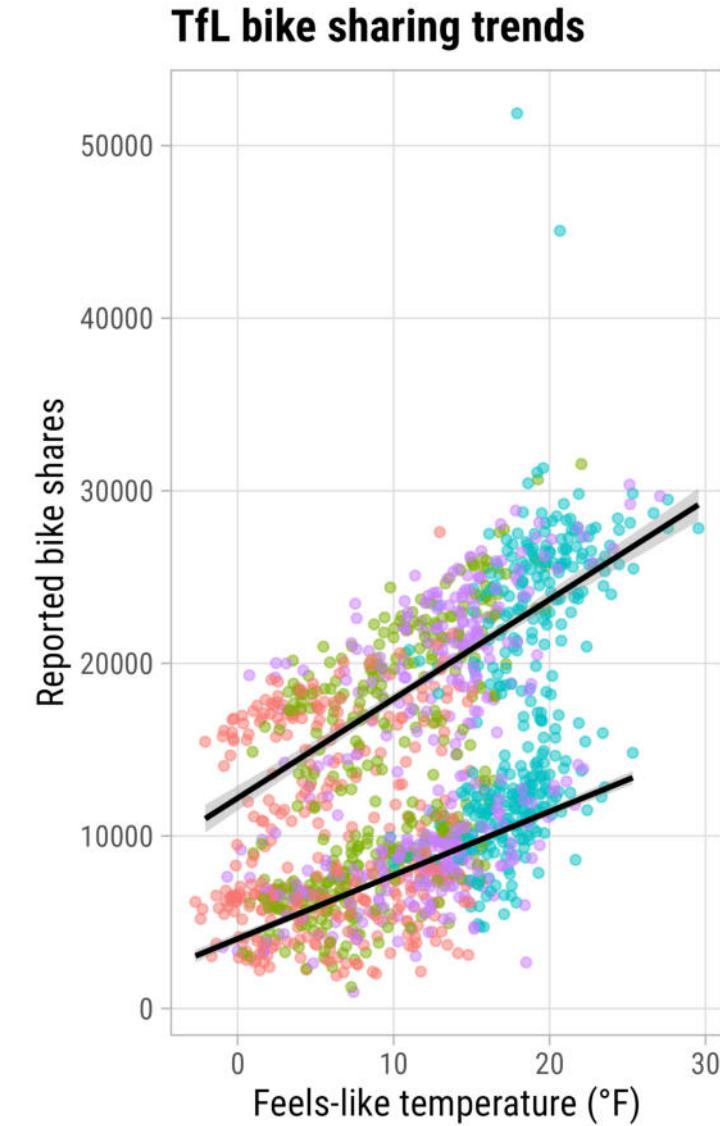
Overwrite Specific Theme Settings

```
1 g +  
2 theme(  
3   panel.grid.minor = element_blank()  
4 )
```



Overwrite Specific Theme Settings

```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold")  
5   )
```

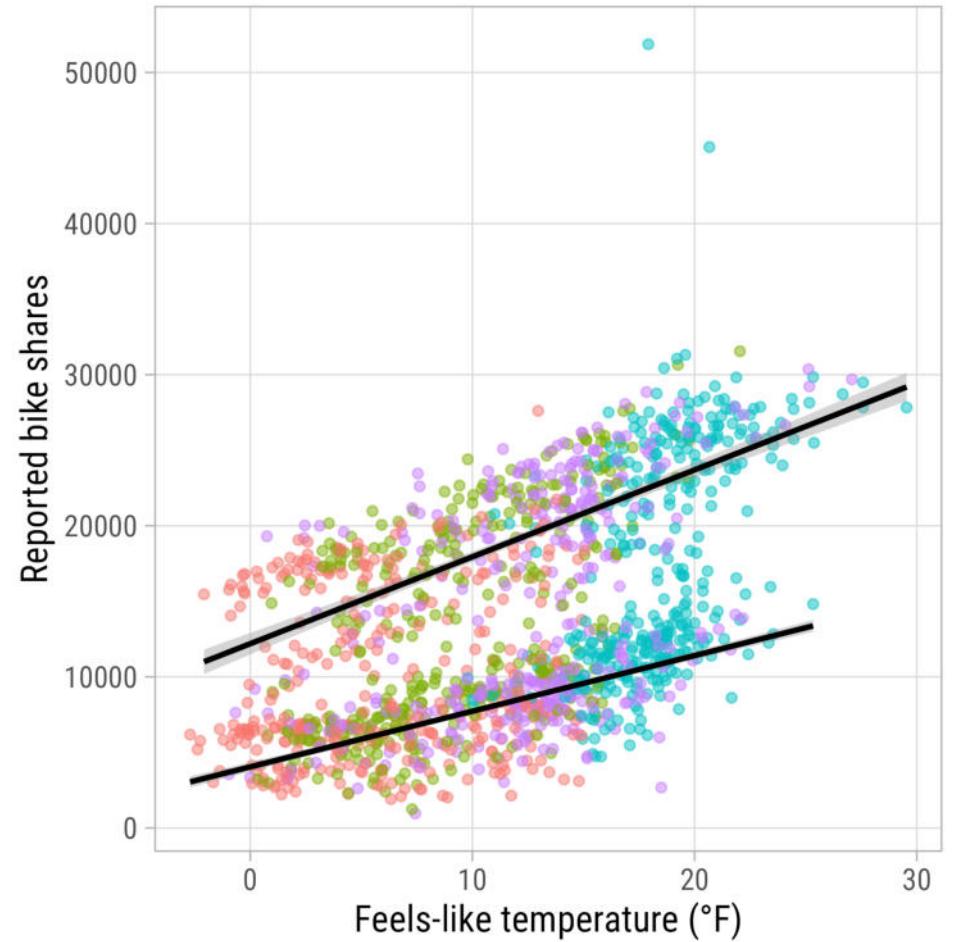


Overwrite Specific Theme Settings

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

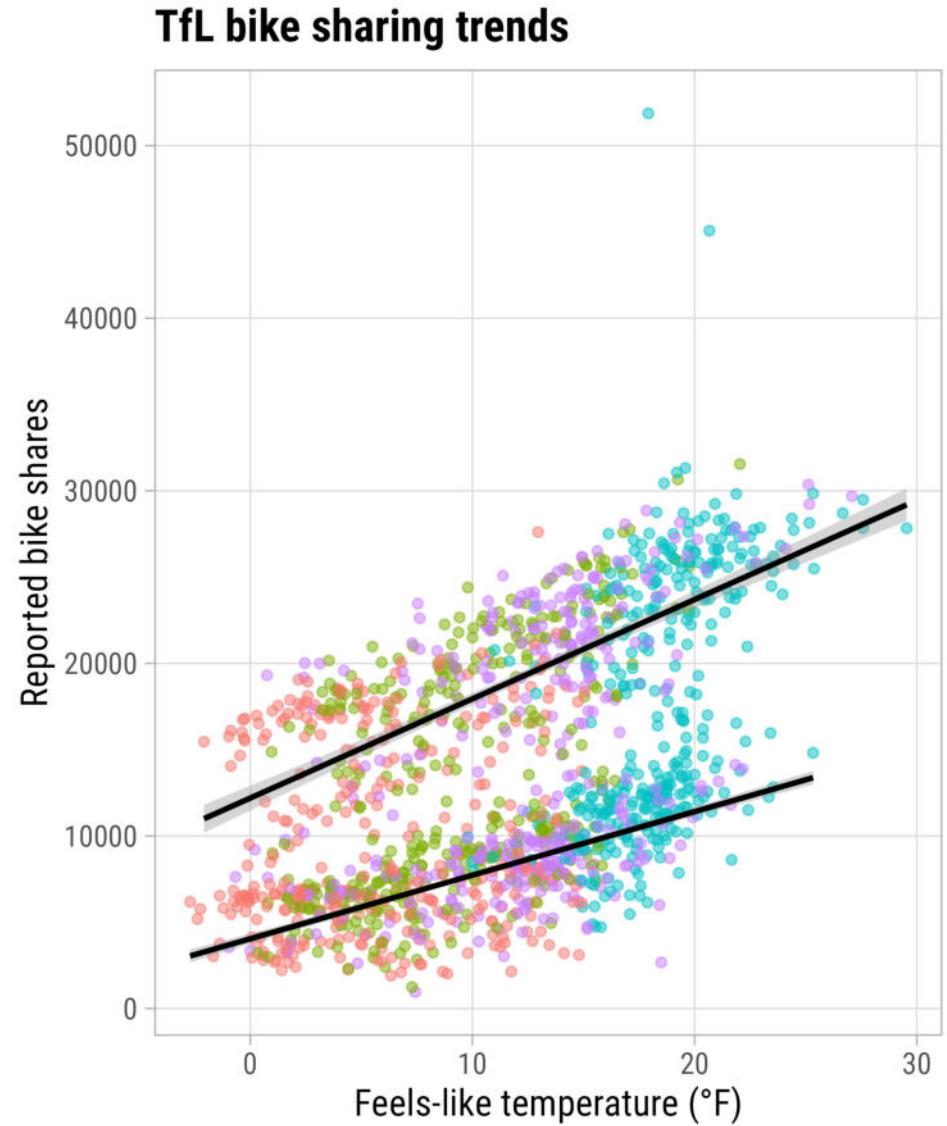
TfL bike sharing trends

Season: ● winter ● spring ● summer ● autumn



Overwrite Specific Theme Settings

```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold"),  
5     legend.position = "none"  
6   )
```

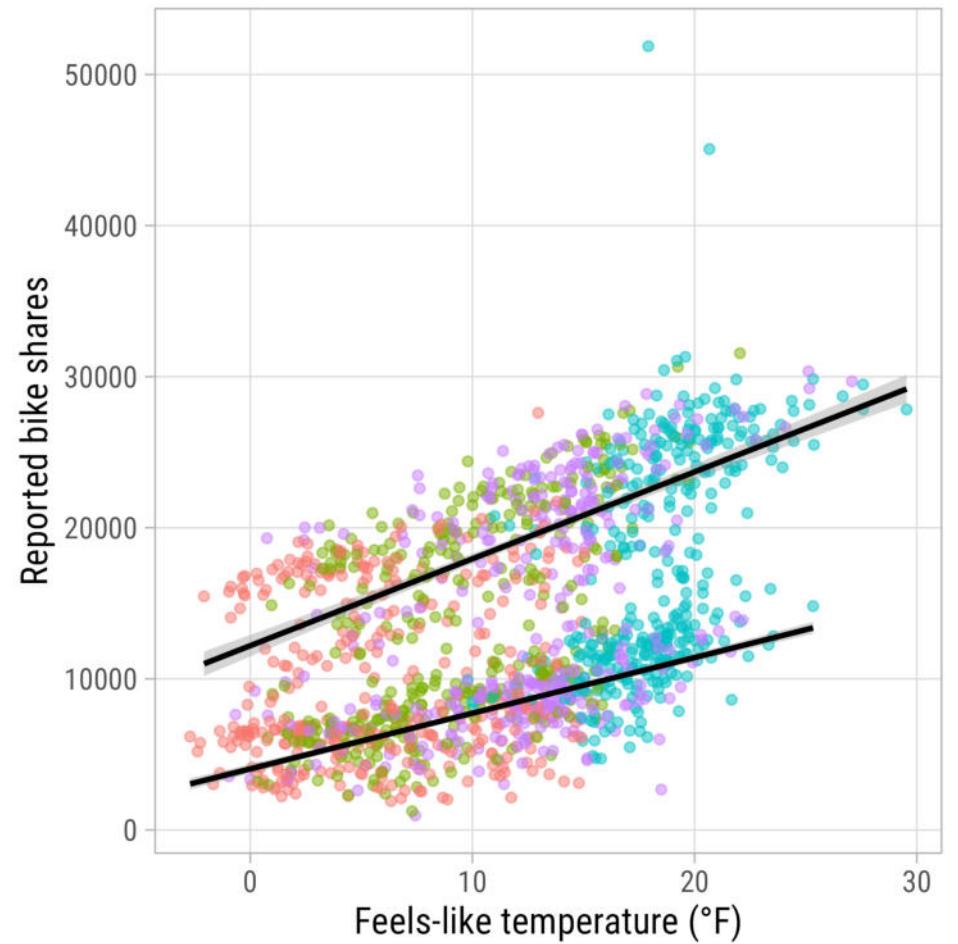


Overwrite Specific Theme Settings

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

TfL bike sharing trends

Season: ● winter ● spring ● summer ● autumn

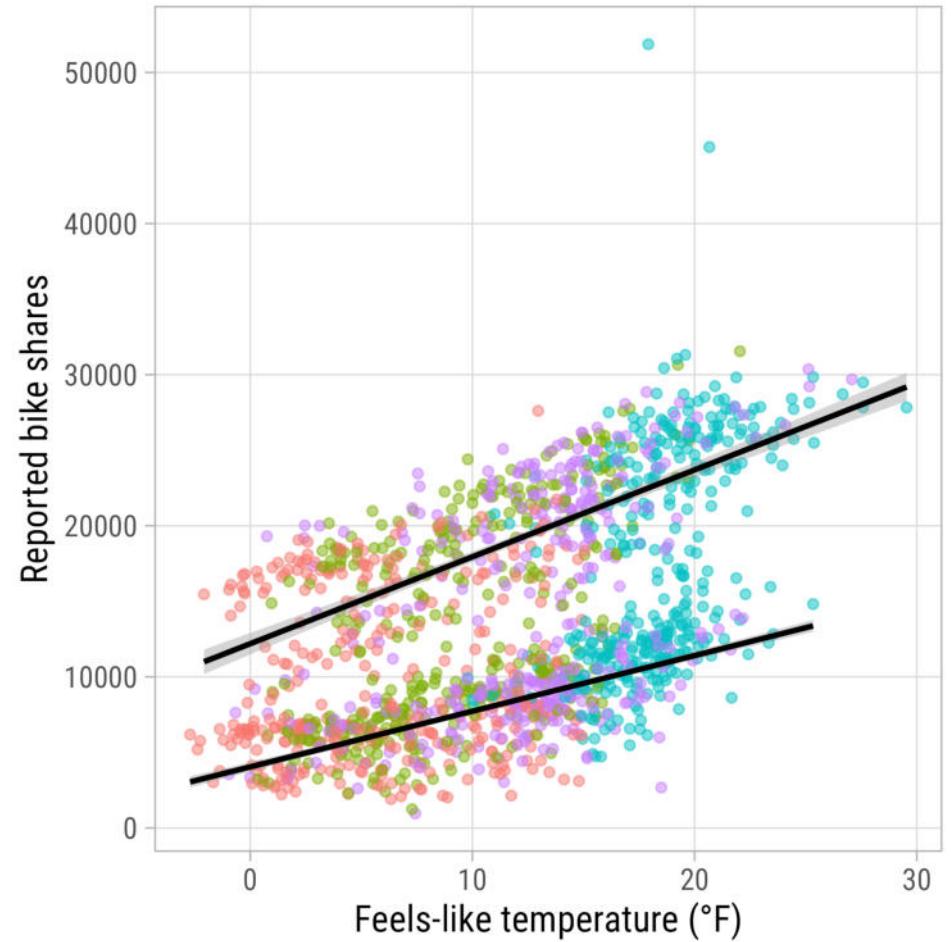


Overwrite Theme Settings Globally

```
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
```

TfL bike sharing trends

Season: ● winter ● spring ● summer ● autumn



Save the Graphic

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

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

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

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

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



Vector graphic

e.g. svg, pdf, eps



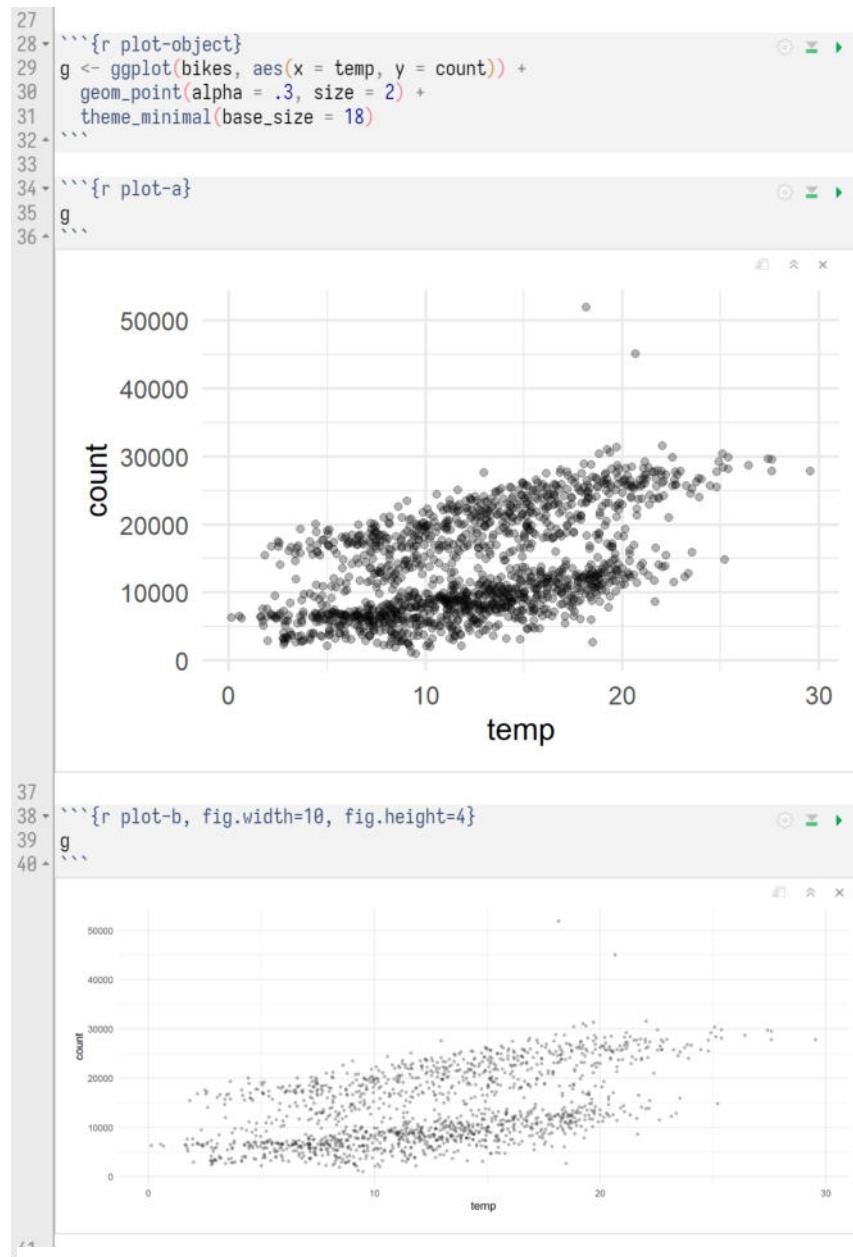
Raster graphic

e.g. png, jpeg, bmp, tiff

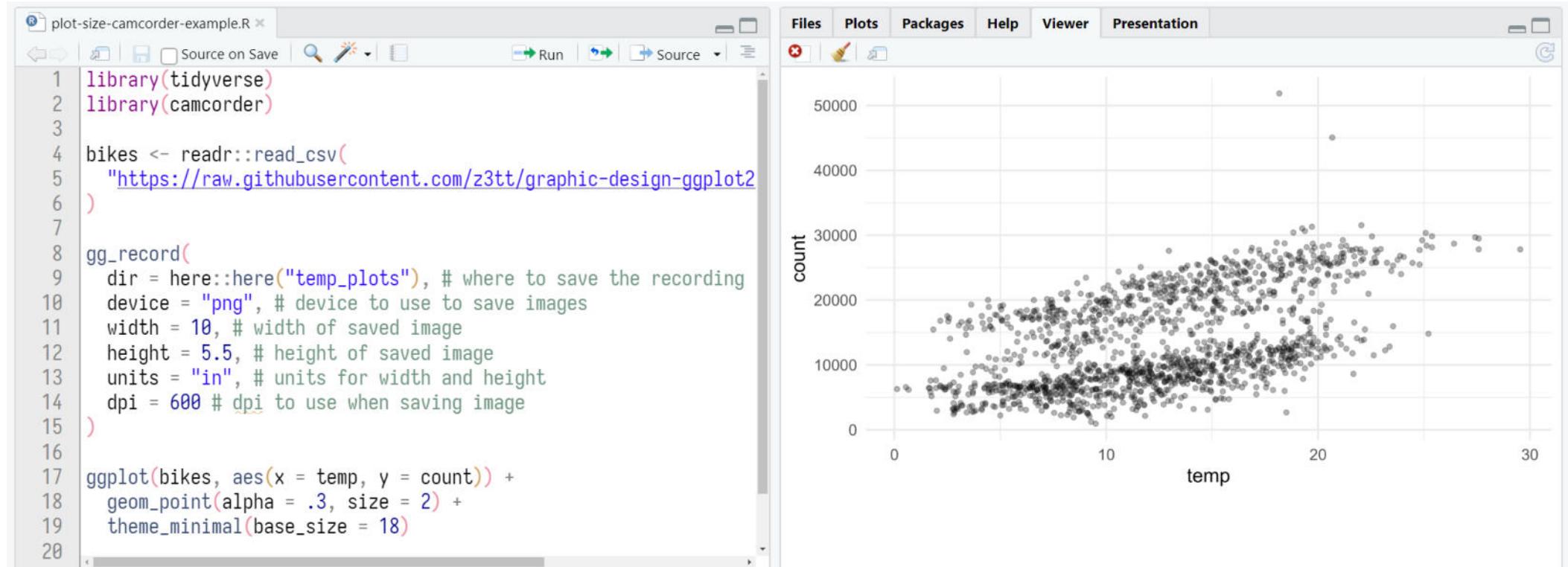
How to Work with Aspect Ratios

- 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



Setting Plot Sizes via {camcorder}



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 properties 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.

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.