

Designing Charts in R

Reproducible Graphic Design with {ggplot2}

Cédric Scherer // Data Visualization Society // March 9, 2023



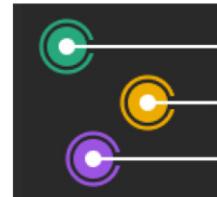
Welcome!



HELLO
my name is

Cédric





CÉDRIC SCHERER

Data Visualization & Information Design



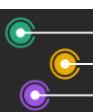
Consulting

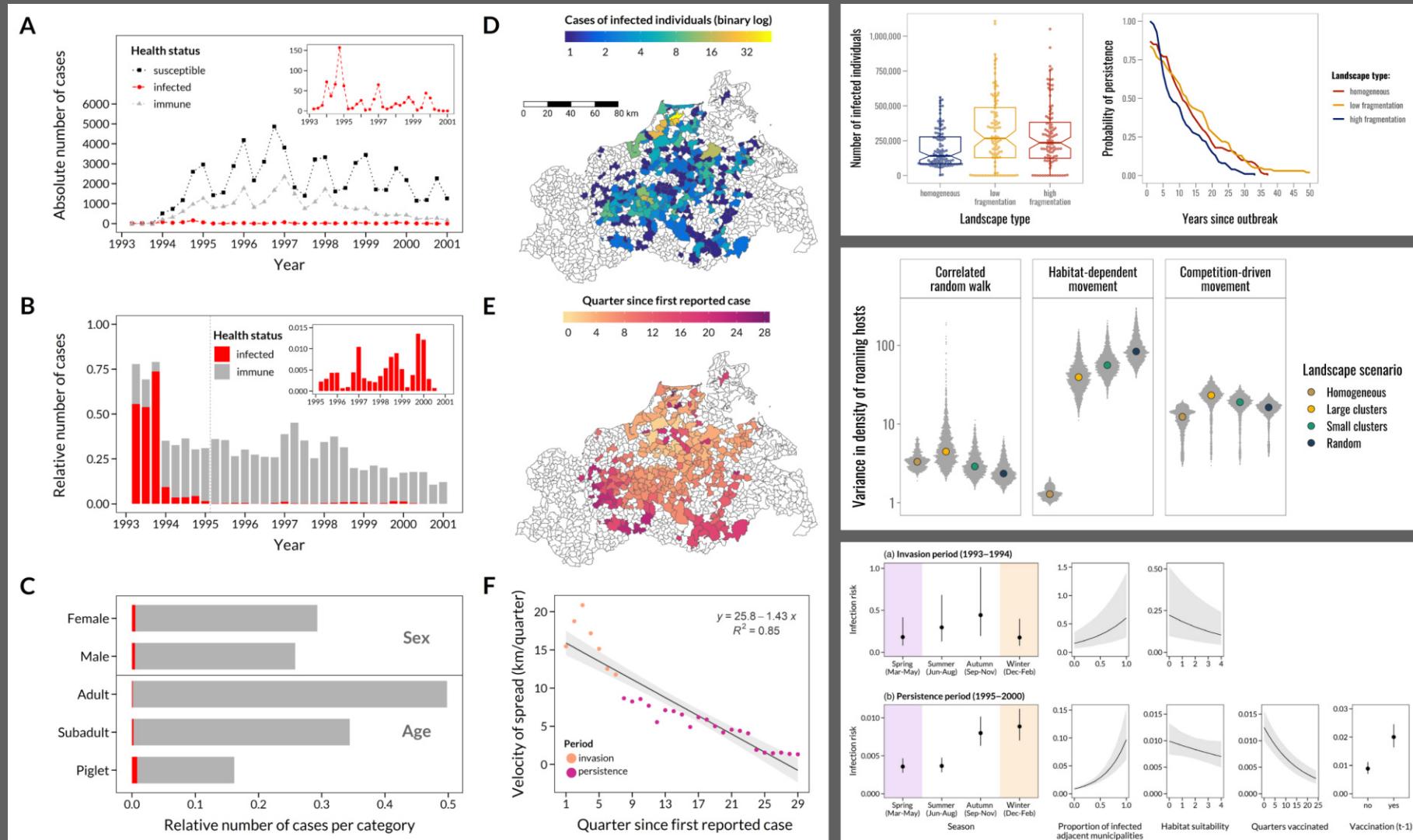


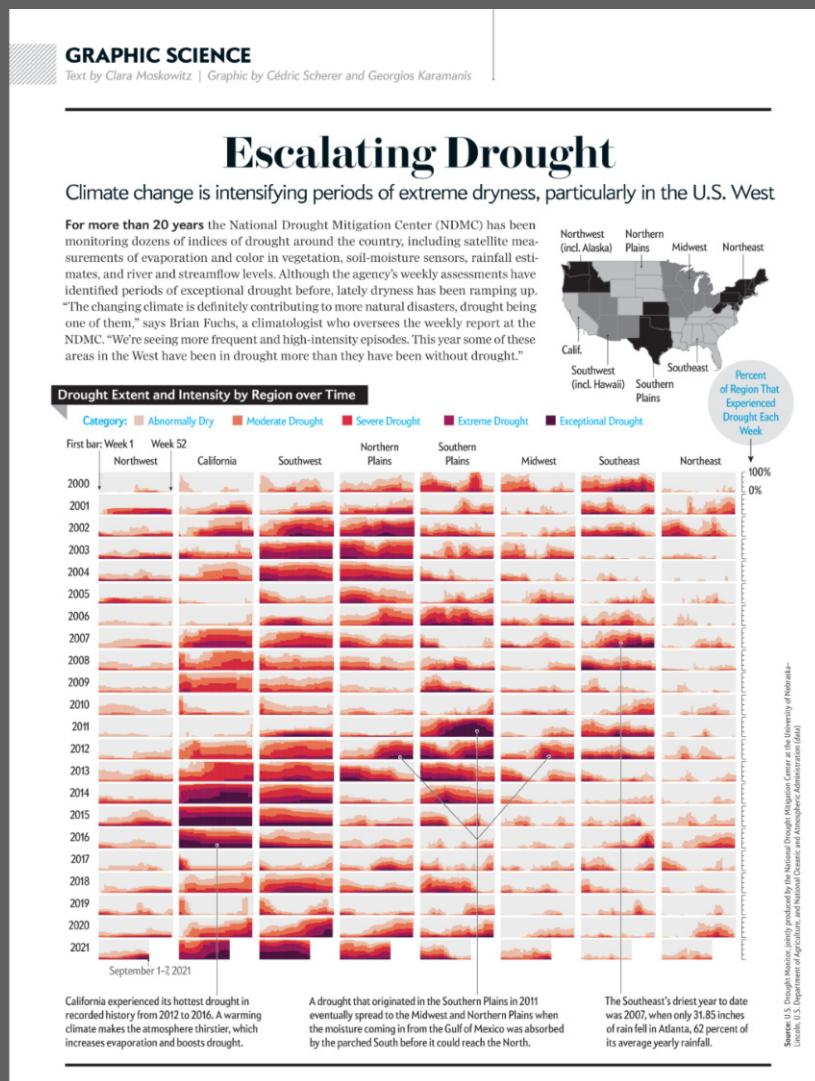
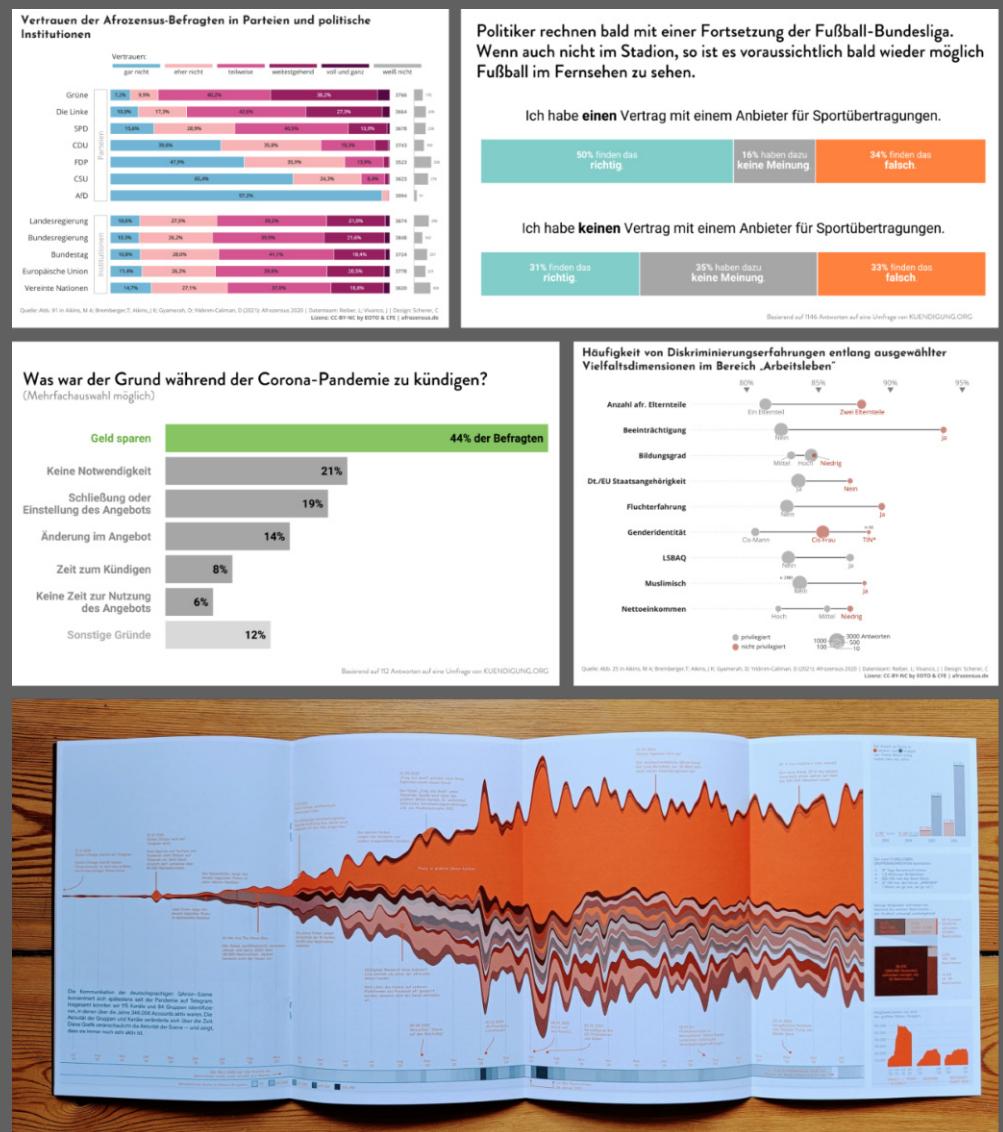
Coaching

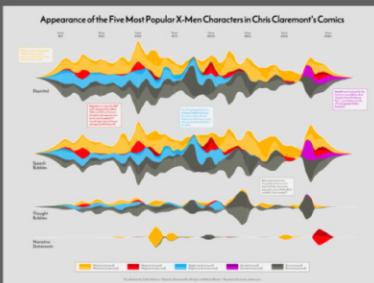
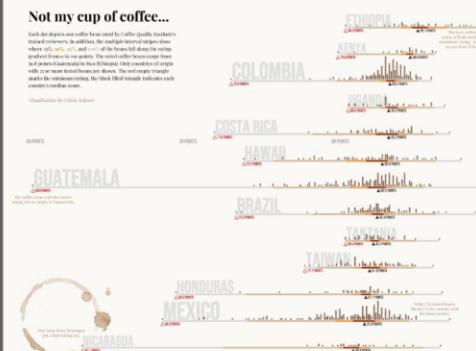
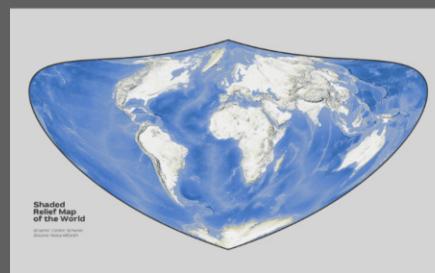
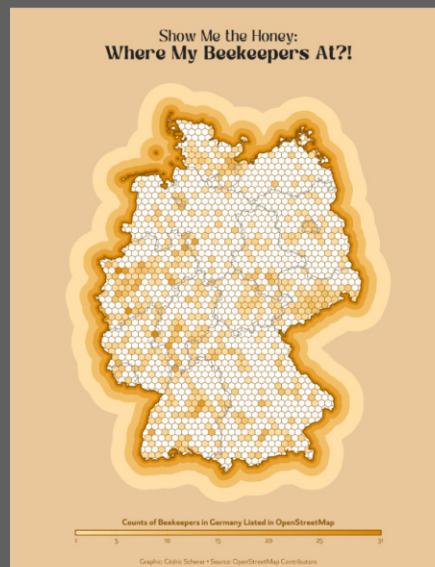
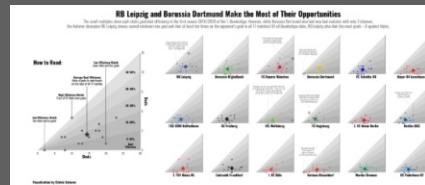


Coding



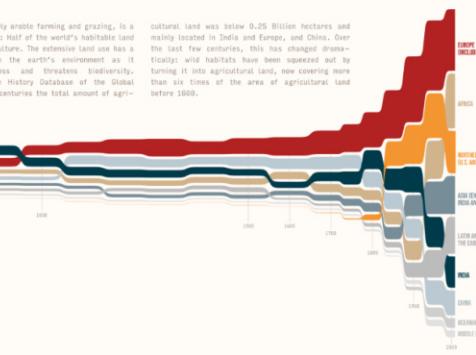
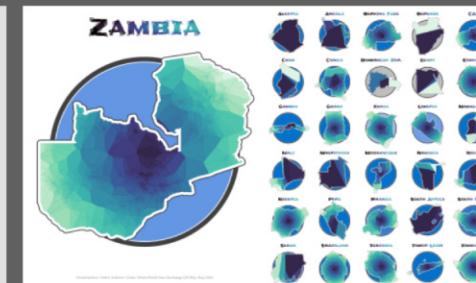
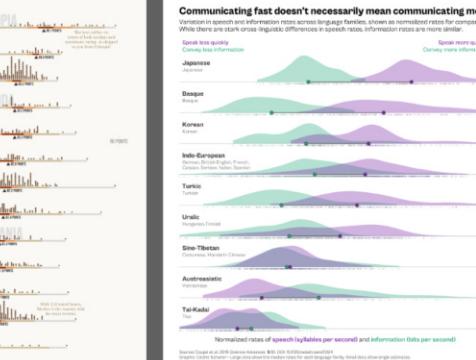






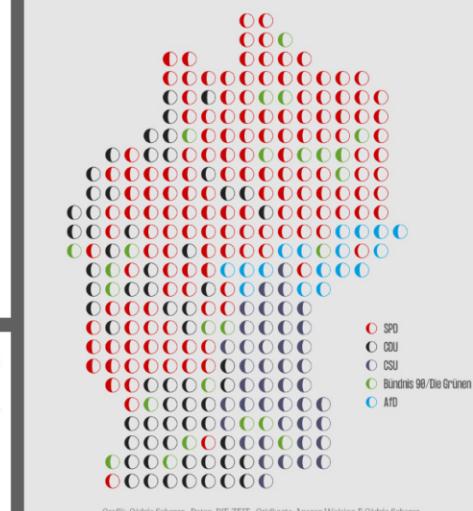
IN 2016, THE GLOBAL AREAL LAND USE FOR AGRICULTURE COVERED 1.6 BILLION HECTARES – AN AREA ALMOST SIX TIMES AS LARGE AS IN 1600 AND TWO TIMES THE AREA OF 1900.

Visualization: Cédric Scherer
Data: History Database of the Global Environment (HDE)

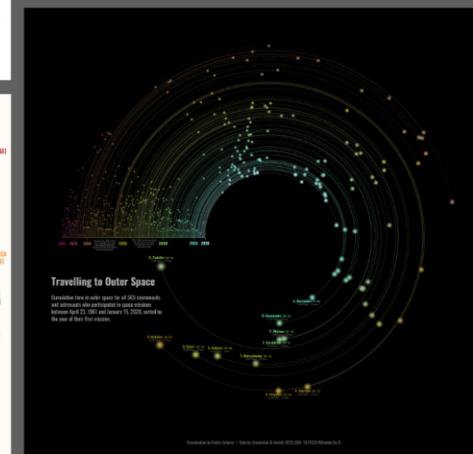


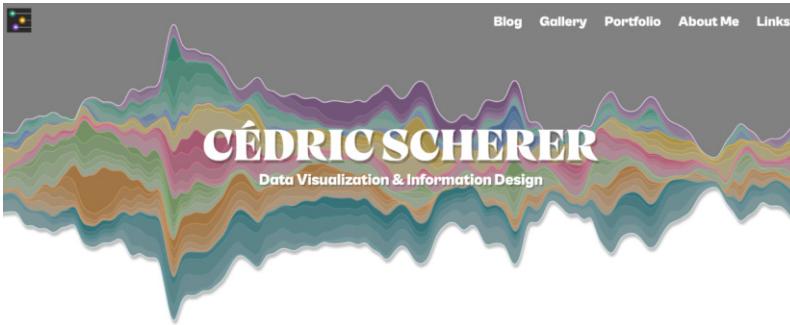
Ergebnisse der Bundestagswahl 2021

Die stärksten Parteien nach Prozent der Zweitstimmen



Grafik: Cédric Scherer · Daten: DIE ZEIT · Gridkarte: Ansgar Wolsing & Cédric Scherer





Hi, I am Cédric

Data Visualization Designer, Consultant and Instructor
for Engaging and Effective Graphical Storytelling.

[↳ Gallery](#) [↳ Portfolio](#) [↳ About Me](#)



Always coding. Passionate about
design. Worried about nature and
inequality. Proud dad.



[Schedule a discovery call](#)

Quick Links

[ggplot2 Tutorial](#)

[Evolution of a ggplot](#)

Featured Tags



Friends

DataVisSociety From Data to Viz
CorreVal Marco Sosial
Georgios Karanikas Julie Brunet
Matthias Stoffl Inbal Reif Will Chase
Anne-Marie Dufour Heureka Labs

2-Day Workshop on "Graphic Design with ggplot2" at rstudio::conf 2022

End of July, I had the honor to teach an in-person ggplot2 workshop at the rstudio::conf in Washington DC. All course resources are available on the course webpage featuring slides,

hands-on R codes, recap notes, and exercises including prepared scripts and step-by-step

solutions.

Tuesday - August 9, 2022

International Women's Day 2022: The Pay Gap in Europe

Sadly, inequality between women and men is still an issue. In 2020, women's gross hourly earnings were on average 13 percentage points below those of male paid employees in the European Union. Only five countries in the EU report a gap smaller than 5 percentage points with Luxembourg being closest to zero.

Tuesday - March 8, 2022

The World's Countries Colored by Their First Letter

While preparing the mapping section for a Pearson-O'Reilly training, I got the idea to visualize the first letter of each country. And got especially curious about how much landmass each letter covers. Turns out: A, C and R are covering the largest areas!

Friday - August 27, 2021

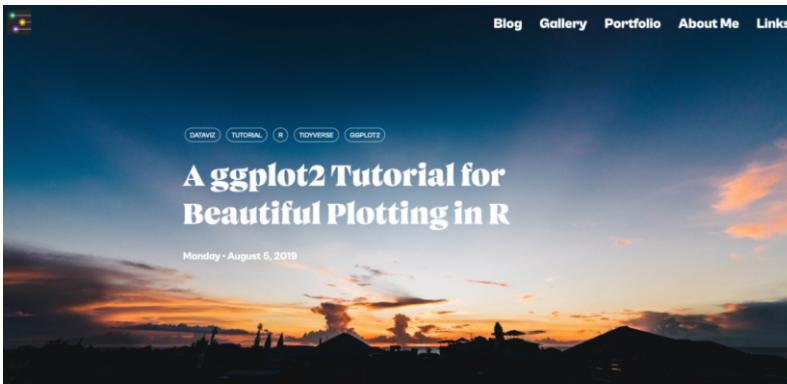
A Quick How-to on Labelling Bar Graphs in ggplot2

Bar charts are likely the most common chart type out there and come in several varieties. Most notably, direct labels can increase accessibility of a bar graph. I got a request how one can add percentage labels inside the bars and how to highlight specific bars with ggplot2. This short tutorial shows you multiple ways how to do so.

Monday - July 5, 2021

Cédric Scherer // Data Visualization Society // March 9, 2023



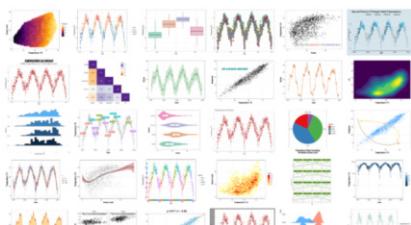


Introductory Words

I don't care, just show me the content!

Back in 2016, I had to prepare my PhD introductory talk and I started using [ggplot2](#) to visualize my data. I never liked the syntax and style of base plots in R, so I was quickly in love with ggplot. Especially useful was its faceting utility. But because I was short on time, I plotted these figures by trial and error and with the help of lots of googling. The resource I come always back to was a blog entry called [Beautiful plotting in R: A ggplot2 cheatsheet](#) by Zev Ross, updated last in January 2016. After giving the talk which contained some decent plots thanks to the blog post, I decided to go through this tutorial step-by-step. I learned so much from it and directly started modifying the codes and over the time I added additional code snippets, chart types and resources.

Since the blog entry by Zev Ross was not updated for some years and step by step this became a unique version of a tutorial, I decided to host the updated version on my GitHub. Now it finds its proper place on this homepage! (Plus I added a ton of other updates-just to name a few: The fantastic [patchwork](#), [eggtext](#) and [eggforce](#) packages. How to deal with custom fonts and colors. A collection of R packages tailored to create interactive charts. And several other chart types including pie charts because everyone looooves pie charts!)





Blog Gallery Portfolio About Me Links

2-Day Workshop on "Graphic Design with ggplot2" at rstudio::conf 2022

Tuesday • August 9, 2022

End of July, I had the honor to teach a 2-day, in-person workshop on "Graphic Design with ggplot2" at the [rstudio::conf\(2022\)](#) in Washington DC. Invited by RStudio (now named [Posit](#)), I developed a new course that covers the most important steps and helpful tips to create visually appealing, engaging and complex graphics with ggplot2. The course focused on the main concepts of the grammar of graphics and used hands-on examples to explore ggplot2's utility to create multi-layered, more complex graphs.



All course resources are available as open-source material on the [course page](#).

The course webpage as well as the slide decks and the exercises and solutions were developed with the new open-source scientific and technical publishing system [Quarto](#). The new workshop development was a perfect opportunity to give it a try and the experience was overall wonderful—the [reveal.js](#) integration for the slides works perfect and allows for a lot of customization. Thanks to Marco Scaini for helping me setting up the course webpage which was, after learning about a few quirks, a smooth experience as well.

Graphic Design with ggplot2

How to Create Engaging and Complex Visualizations in R

An [rstudio::conf\(2022\) Workshop](#) by Cédric Scherer

July 25 – July 26, 2022
9h00 – 17h00
Berlin, Berlin, Germany
Online

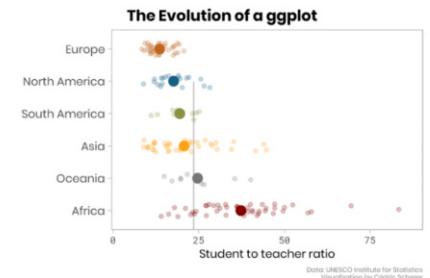
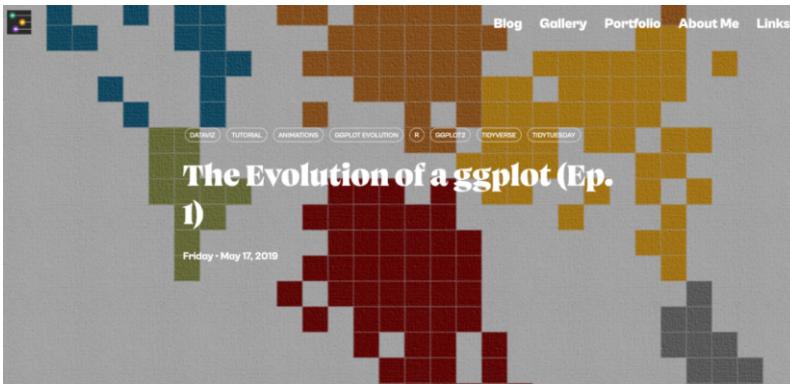
Course Overview

Creating engaging and accessible data visualizations of high quality is an efficient and preferable way to communicate data and findings. After attending this workshop, participants will have a solid understanding of data visualization principles and the functionality of the ggplot2 graphics library and helpful extension libraries to create highly customized graphics without the need for extensive coding.

We will discuss the main concepts of the grammar of graphics and use hands-on examples to explore ggplot2's utility to create multi-layered, more complex graphs. The workshop covers a short overview of the history of ggplot2 and its evolution, then we'll cover the most important concepts and build it from scratch.

The [course webpage](#) which gives an overview of the course objectives and links to all session and the [GitHub repository](#).

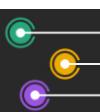




- [Aim of this Tutorial](#)
- [Data Preparation](#)
- [The Default Boxplot](#)
- [Sort Your Data!](#)
- [Let Your Plot Shine—Get Rid of the Default Settings](#)
- [The Choice of the Chart Type](#)
- [More Geoms, More Fun, More Info!](#)
- [Add Text Boxes to Let The Plot Speak for Itself](#)
- [Bonus: Add a Tile Map as Legend](#)
- [The Final Evolved Visualization](#)
- [Complete Code for Final Plot](#)
- [Post Scriptum: Mean versus Median](#)

■■ Aim of this Tutorial

In this series of blog posts, I aim to show you how to turn a default ggplot into a plot that visualizes information in an appealing and easily understandable way. The goal of each blog post is to provide a step-by-step tutorial explaining how my visualization have evolved from a typical basic ggplot. All plots are going to be created with 100% `ggplot2` and 0% Inkscape.



Cédric Scherer
z3tt

Data Visualization Specialist • Computational Ecologist

Edit profile

4.1k followers • 119 following

Self-Employed • IZW Berlin
Berlin
cedricscherer.com
@CedScherer

Achievements

Organizations

Pinned

My GitHub Activity
aka "How it looks to be a dataviz coder"
A comedic chart showing my GitHub activity. Made up data! I actually code a lot—and fast.

What I Actually Do
aka "Do you need a dataviz coder?"
I design effective graphics for custom needs
I help you to find a suitable chart and design to your needs effectively!
... using an efficient code-first approach
I help you to create scripts based workflows to automate reporting and easily generate graphics!

Commits trying to make fancy interfaces work
Commits that actually contain code changes
Commits fixing typos, preprocessor errors, or preexisting errors

Buy me a coffee

TidyTuesday Public My contributions to the #TidyTuesday challenge, a weekly data visualization challenge. All plots are created in R with ggplot2.
R ⭐ 850 ⚡ 132

OutlierConf2021 Public Slides and hands-on codes for my talk "ggplot Wizardry: My Favorite Tricks and Secrets for Beautiful Plots in R" at the 1st OutlierConf, February 4-7 2021.
HTML ⭐ 404 ⚡ 72

rstudio-conf-2022/ggplot2-graphic-design Public "Graphic Design with ggplot2: How to Create Engaging and Complex Visualizations in R", an rstudio::conf(2022) Workshop by Cédric Scherer
R ⭐ 122 ⚡ 56

beyond-bar-and-box-plots Public Slides, recording and hands-on tutorial for the USGS Data Science webinar
HTML ⭐ 219 ⚡ 37

30DayMapChallenge2019 Public My contributions to the #30DayMapChallenge 2019, a daily challenge focusing on spatial visualizations happening throughout November.
HTML ⭐ 172 ⚡ 36

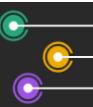
thebioengineer/camcoder Public Record plots generated during an R session and replay as a gif!
R ⭐ 154 ⚡ 4

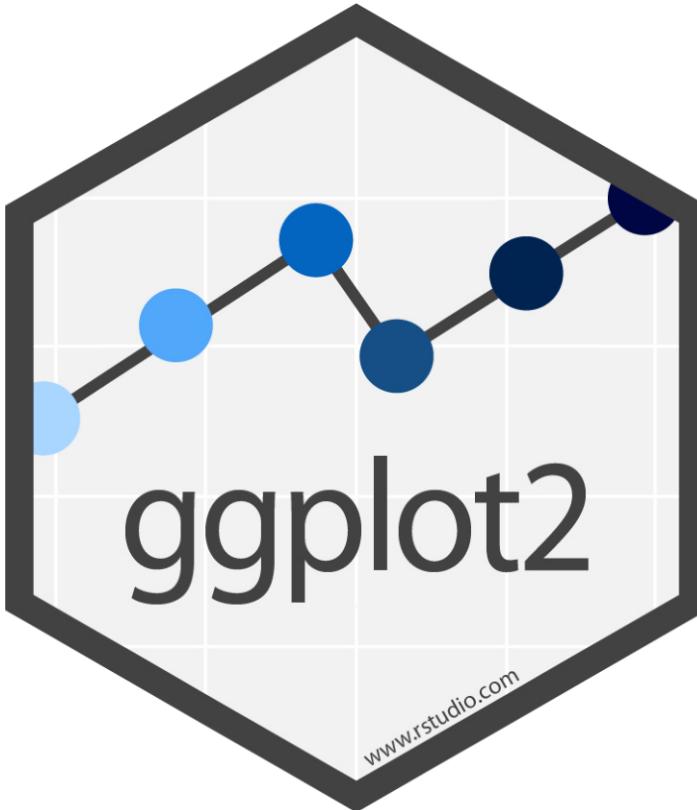
822 contributions in the last year Contribution settings 2023

Cédric Scherer // Data Visualization Society // March 9, 2023



The ggplot2 Package





{ggplot2} is a system for declaratively creating graphics,
based on “The Grammar of Graphics” (Wilkinson, 2005).

You provide the data, tell **{ggplot2}** how to map variables to aesthetics,
what graphical primitives to use, and it takes care of the details.



Advantages of {ggplot2}

- code-first approach → reproducible and transparent workflow
- consistent underlying “grammar of graphics”
- very flexible, layered plot specification
- theme system for polishing plot appearance
- lots of additional functionality thanks to extensions
- active and helpful community

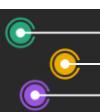


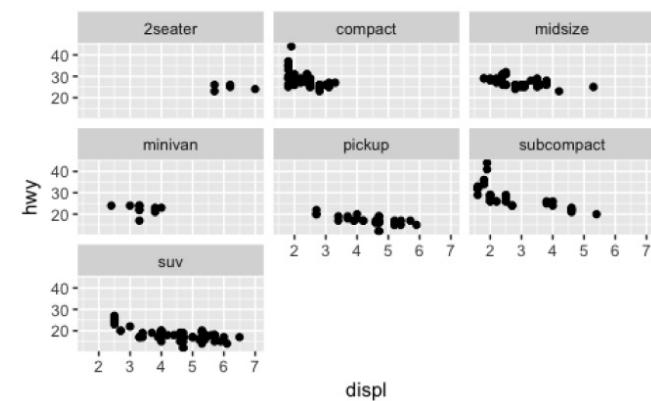
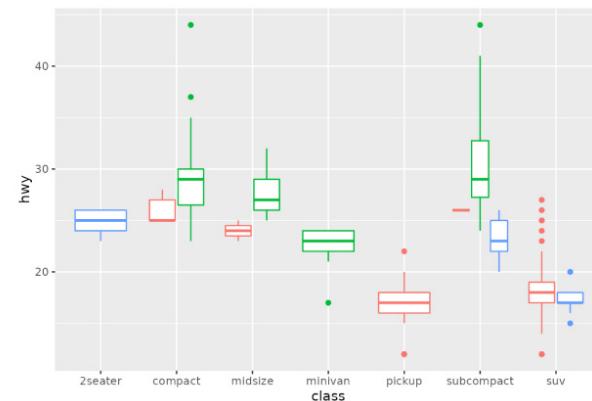
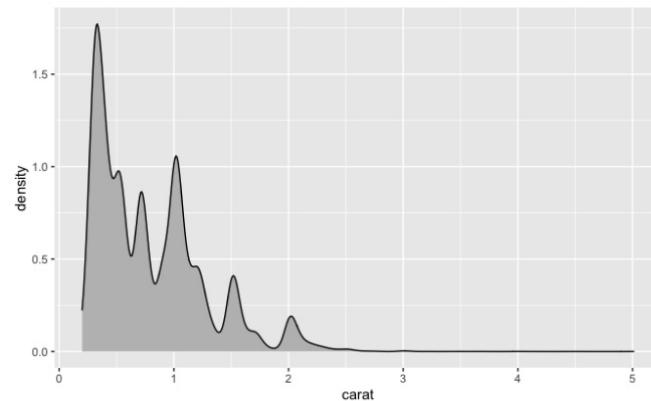
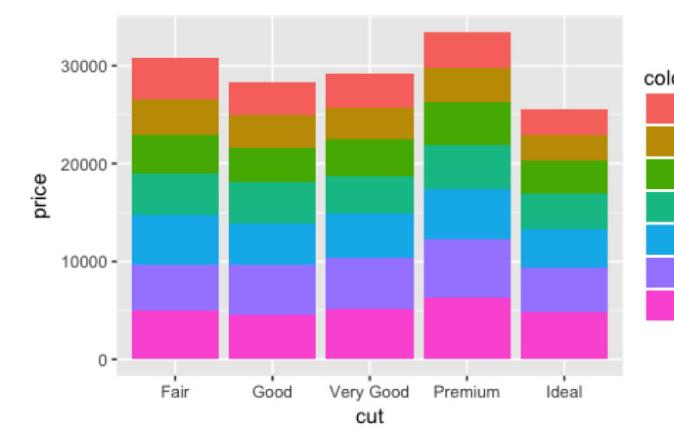
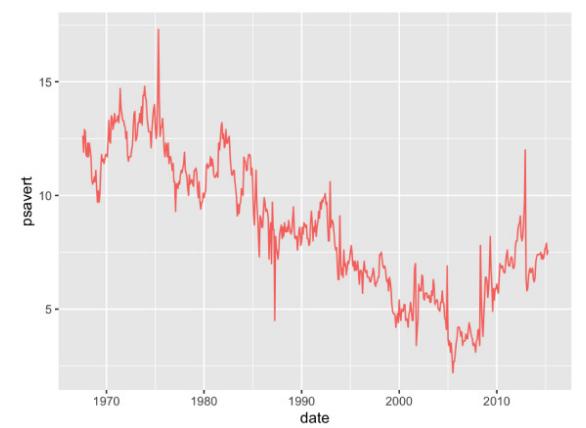
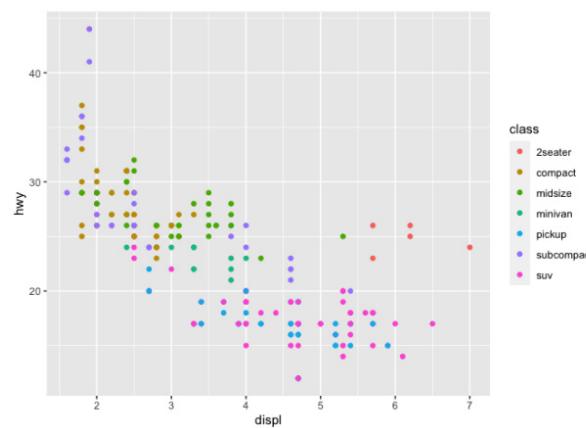
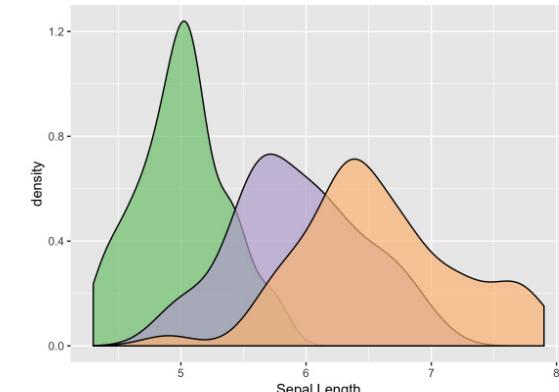
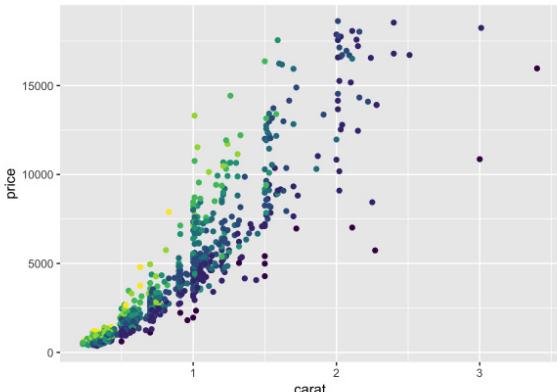
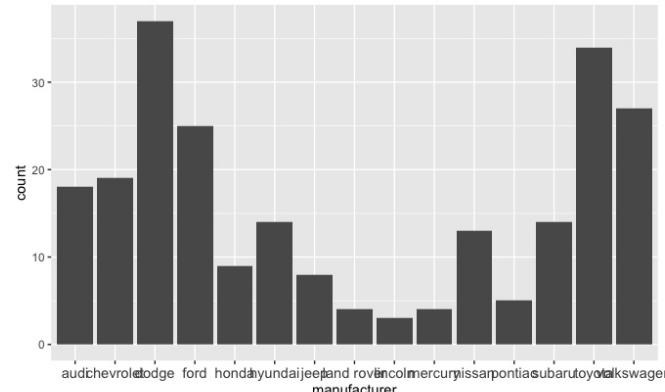
ggplot2: VISUAL DATA EXPLORATION



Illustration by [Allison Horst](#)

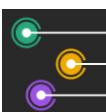
Cédric Scherer // Data Visualization Society // March 9, 2023





ggplot2 Examples featured on ggplot2.tidyverse.org

Cédric Scherer // Data Visualization Society // March 9, 2023



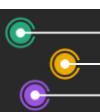
ggplot2:

Build a data
MASTERPIECE

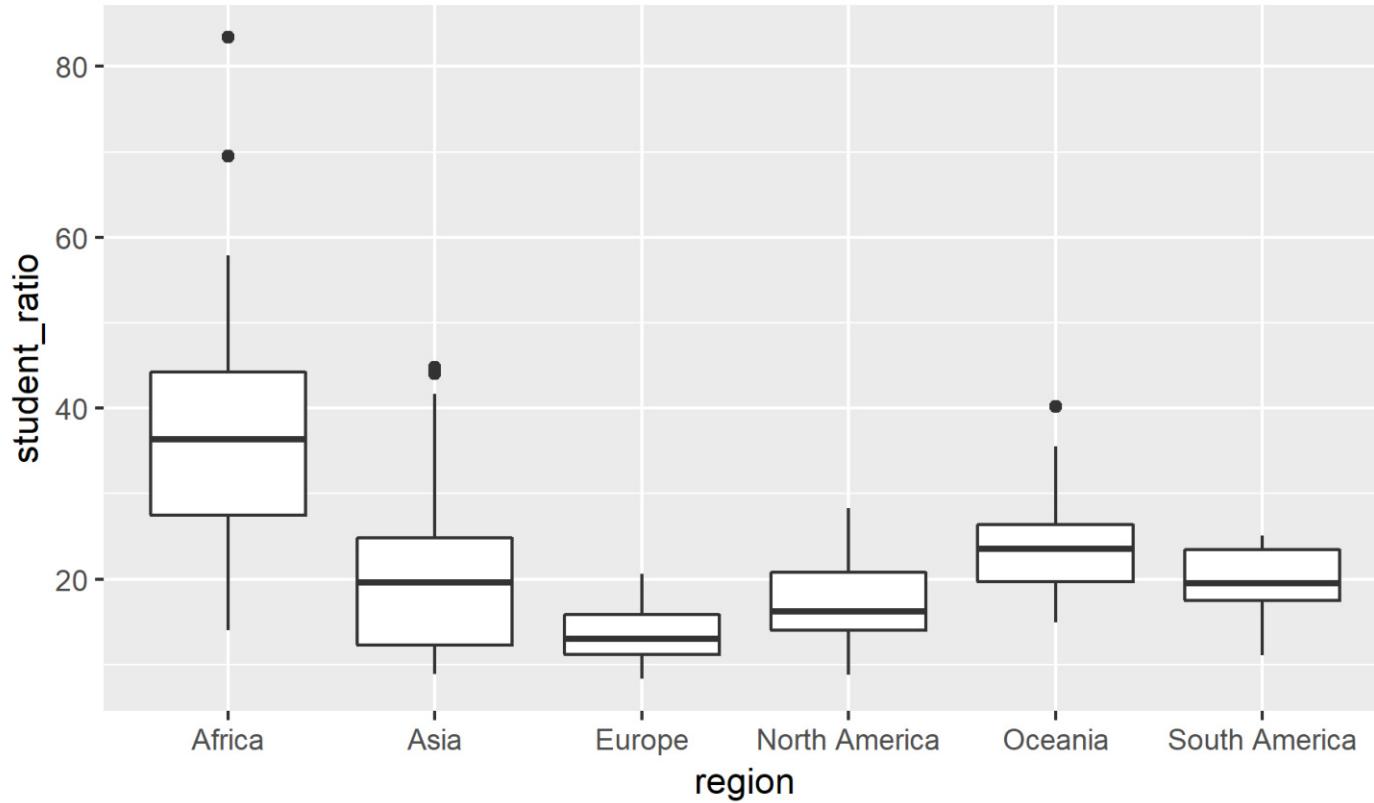


Illustration by [Allison Horst](#)

Cédric Scherer // Data Visualization Society // March 9, 2023



The Evolution of a ggplot



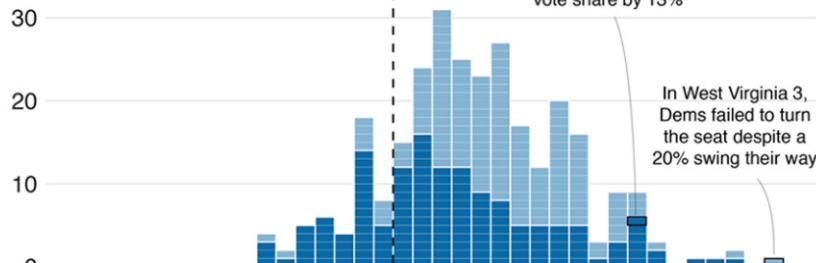
Data: UNESCO Institute for Statistics
Visualization by Cédric Scherer



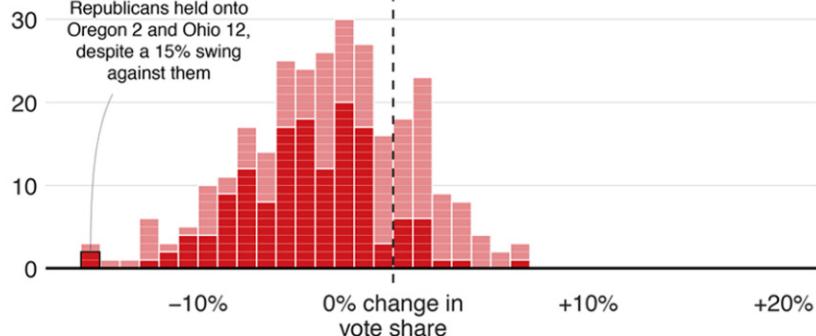
Blue wave

■ Won seat ■ Didn't win

Democrat candidates



Republican candidates

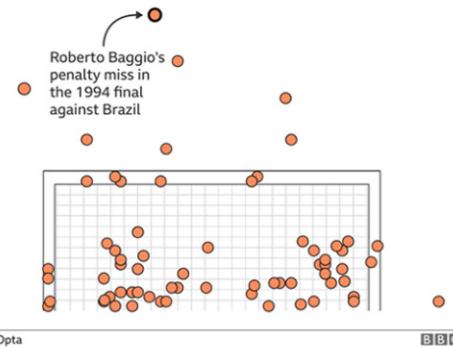


Source: AP, 19:01 ET

BBC

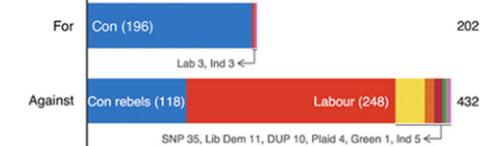
Where penalties are saved

World Cup shootout misses and saves, 1982-2014



Source: Opta

MPs rejected Theresa May's deal by 230 votes

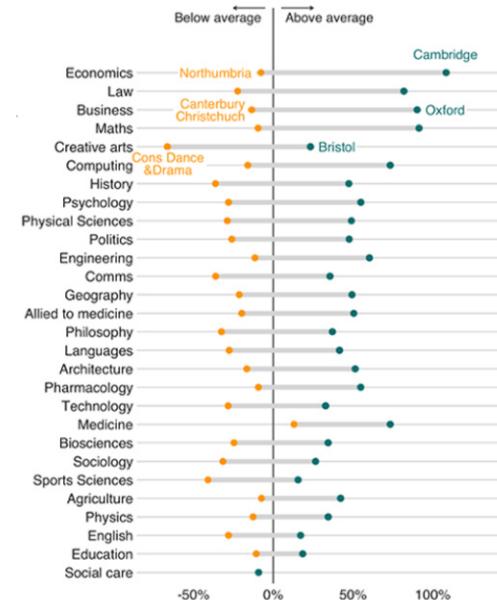


Source: Commons Votes Services. Excludes 'tellers', the Speaker and deputies

BBC

Earnings vary across unis even within subjects

Impact on men's earnings relative to the average degree



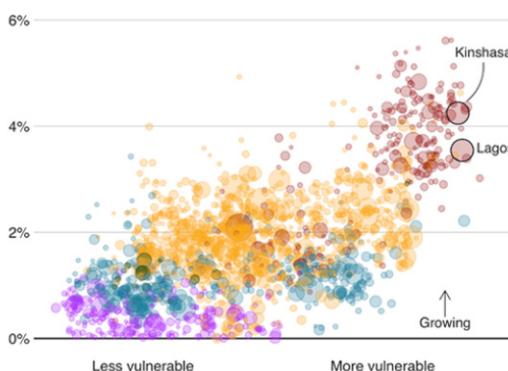
Source: Institute for Fiscal Studies

BBC

Fast-growing cities face worse climate risks

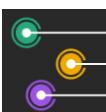
Population growth 2018-2035 over climate change vulnerability

■ Africa ■ Asia ■ Americas ■ Europe ■ Oceania



Source: Verisk Maplecroft. Circle size represents current population.

BBC



Blue wave

■ Won seat ■ Didn't win

Democrat candidates

Sharice Davids, the joint-first Native American woman in congress, increased Dem vote share by 13%

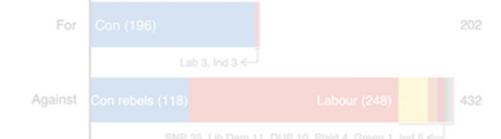


Where penalties are saved

World Cup shootout misses and saves, 1982-2014



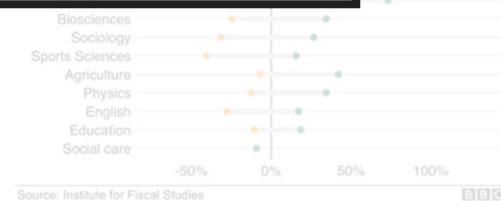
MPs rejected Theresa May's deal by 230 votes



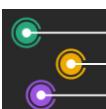
“At the **BBC data team**, we have developed an R package and an R package to make the **process of creating publication-ready graphics** in our in-house style using R’s ggplot2 library **a more reproducible process.**”

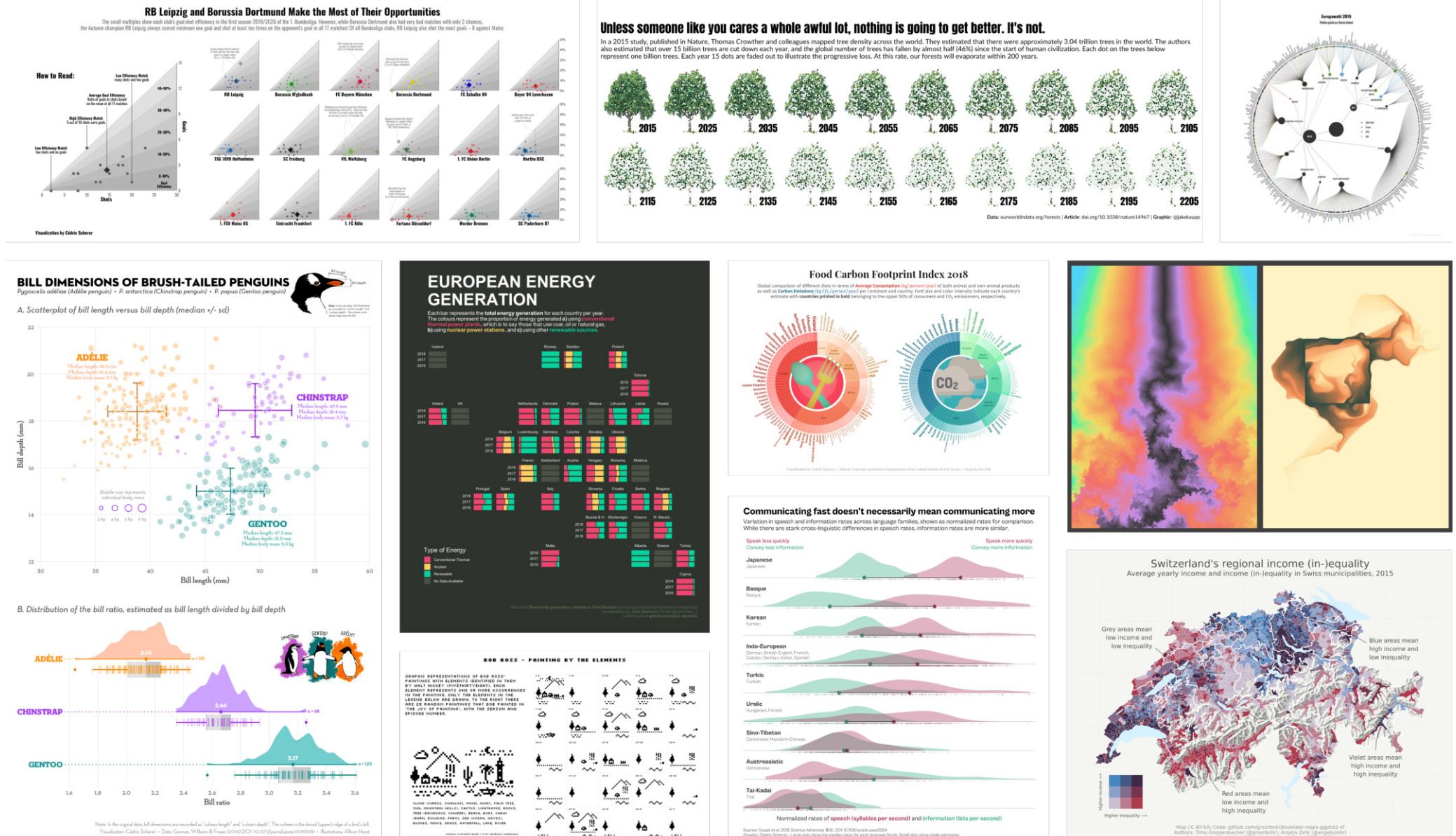


Source: Verisk Maplecroft. Circle size represents current population.

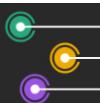


Source: Institute for Fiscal Studies



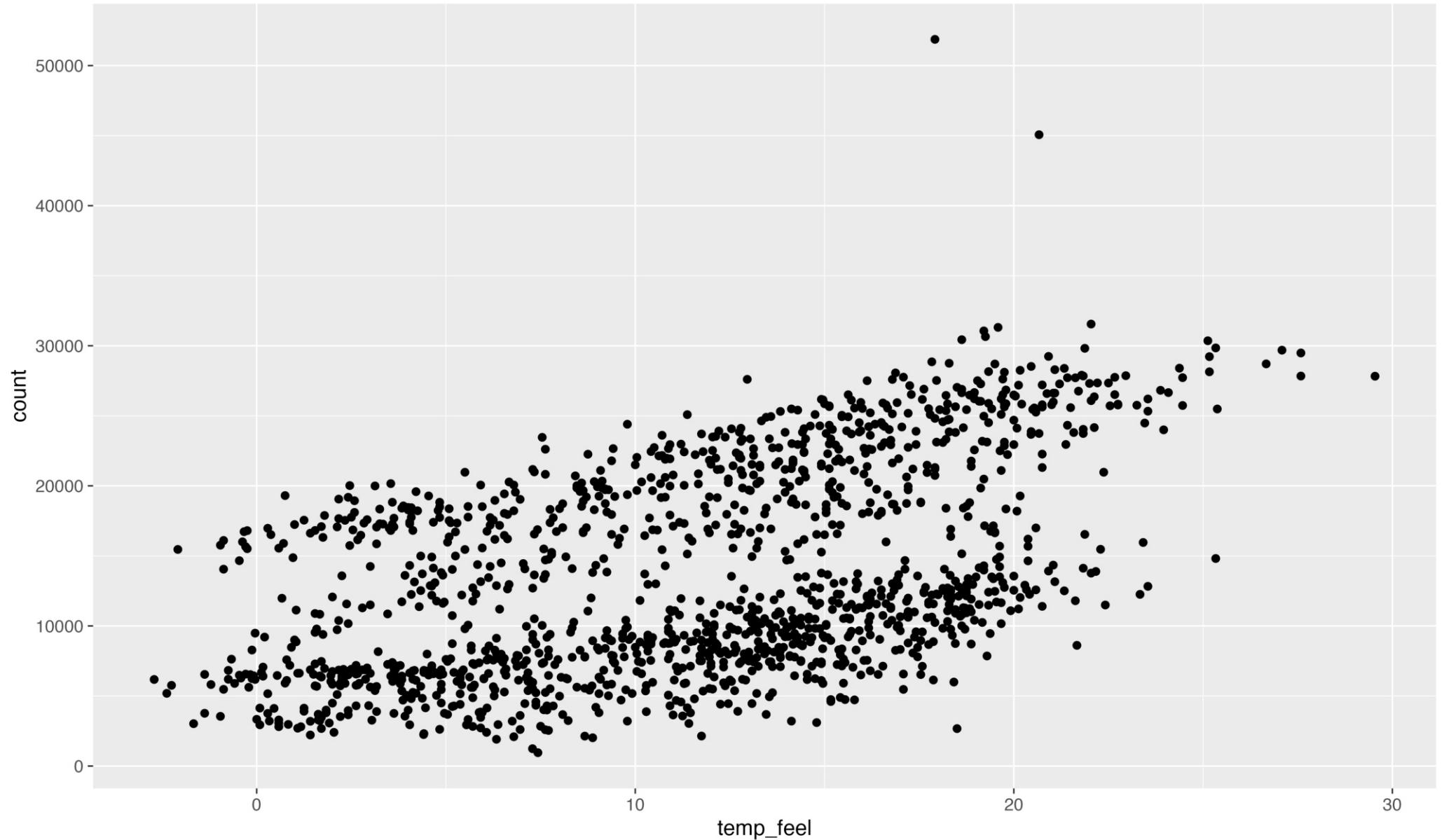


Selection of visualizations created 100% with ggplot2 by Thomas Linn Pedersen, Georgios Karamanis, Timo Gossenbacher, Torsten Sprengler, Jake Kaupp, Jack Davison, and myself.

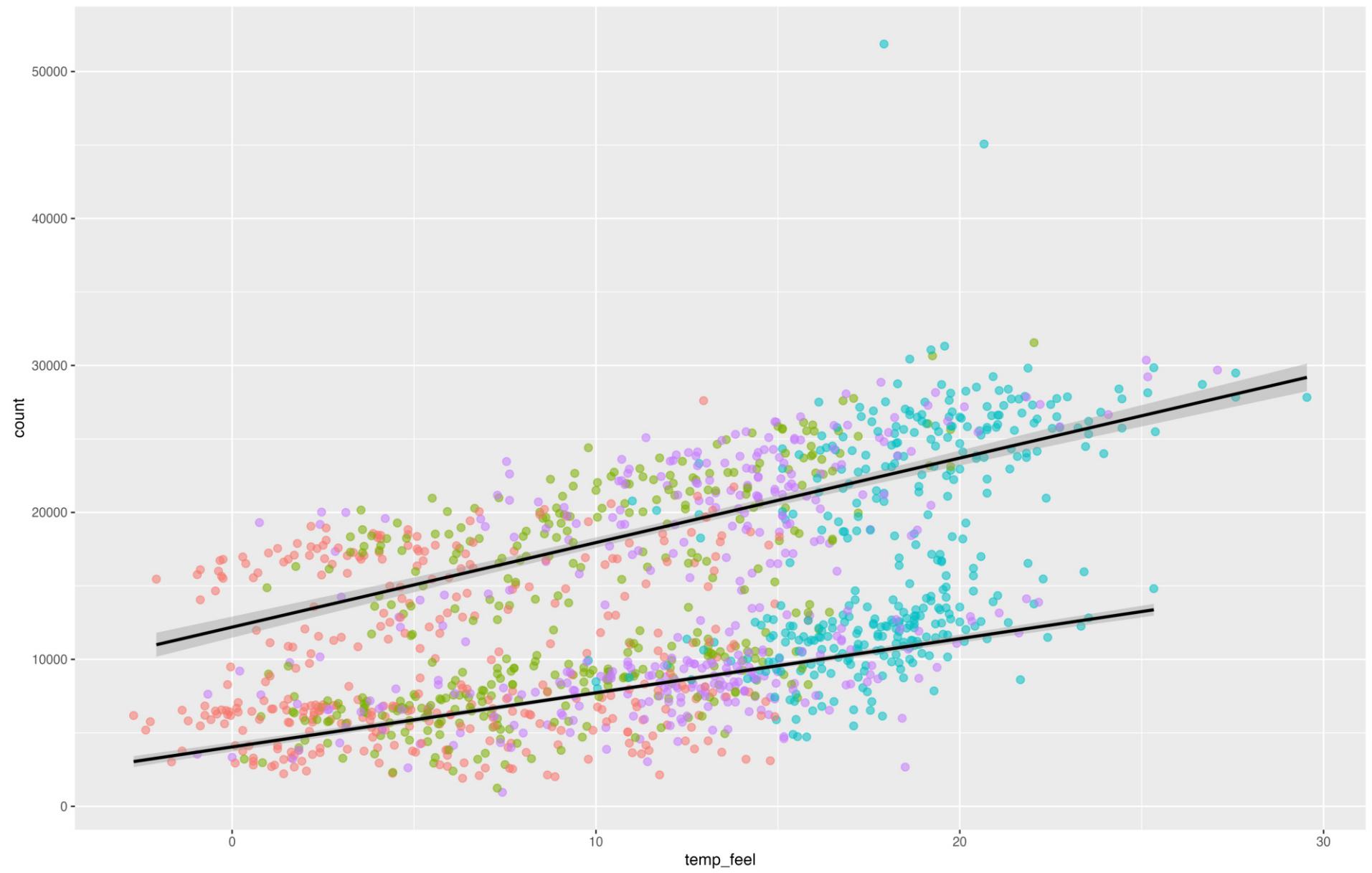


A Motivational Example

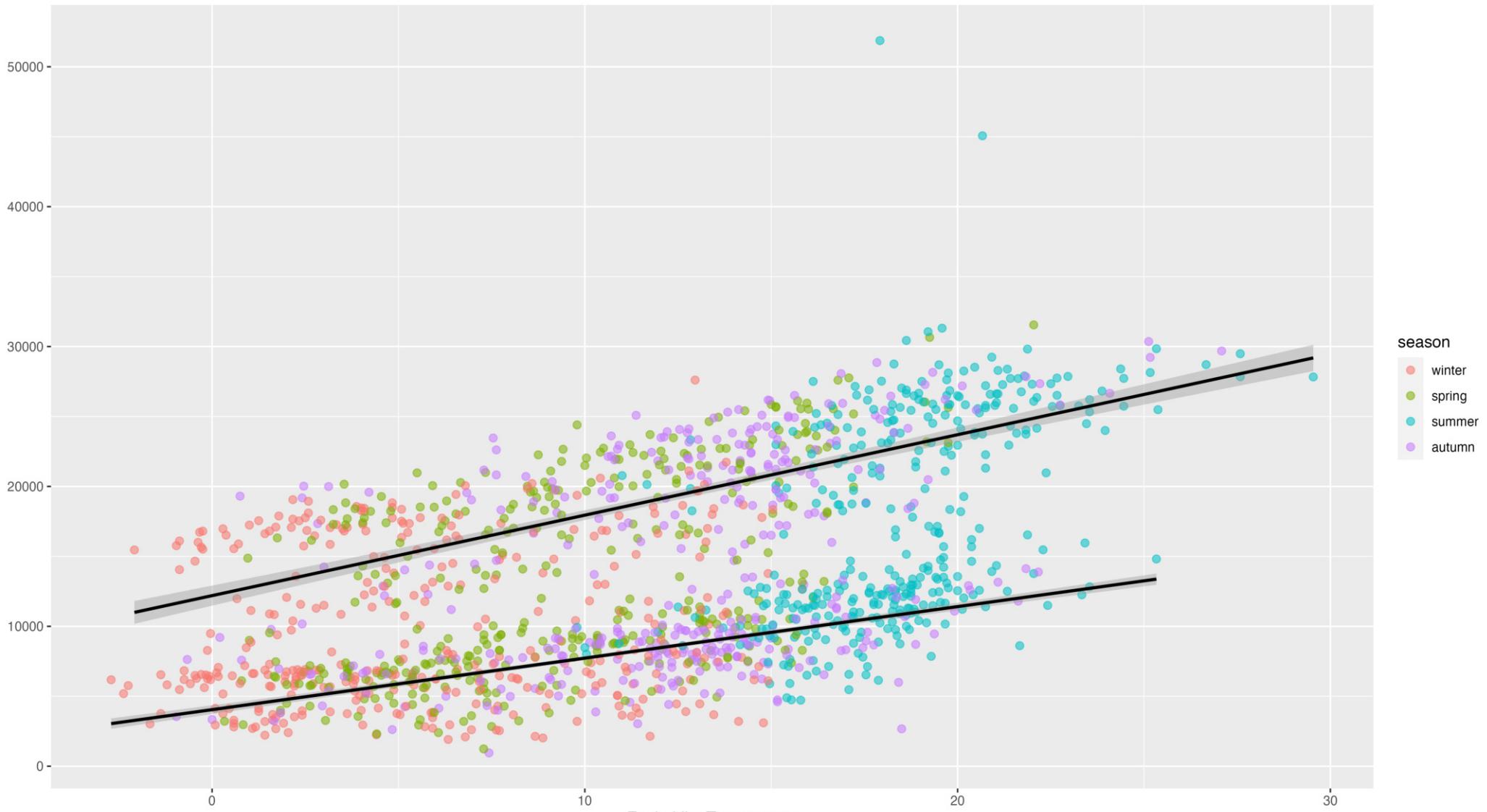




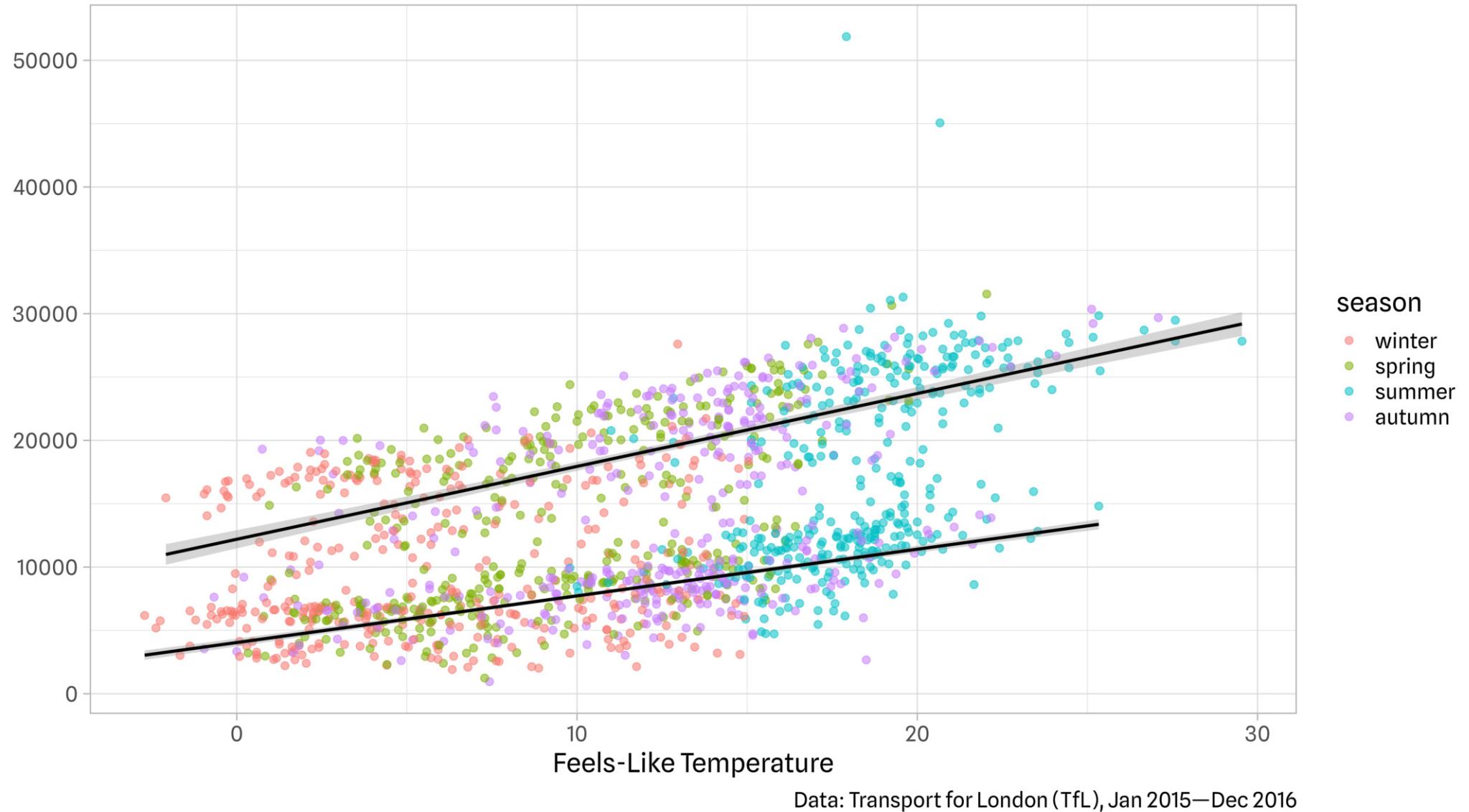




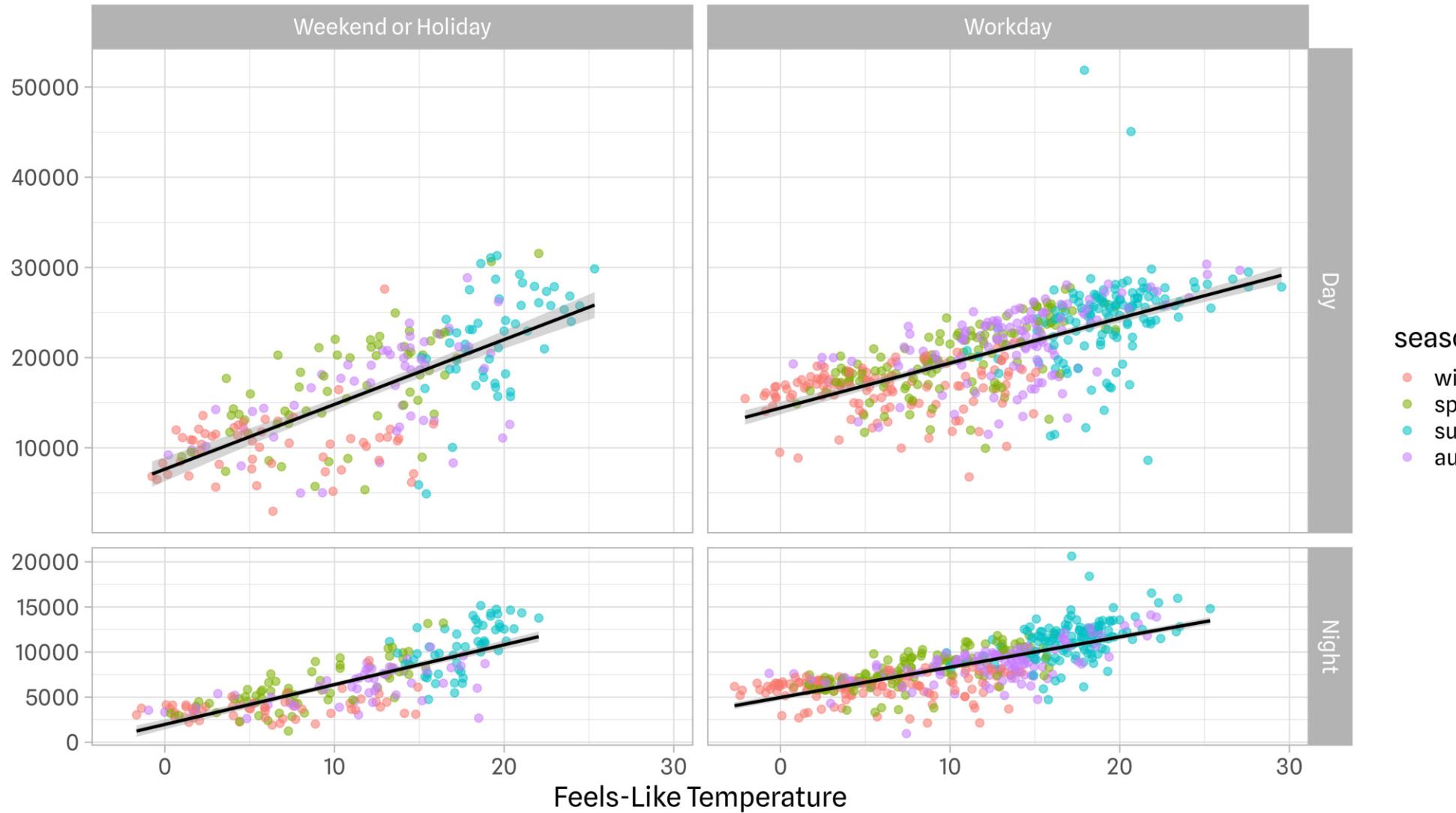
Reported TfL bike rents versus feels-like temperature in London, 2015–2016



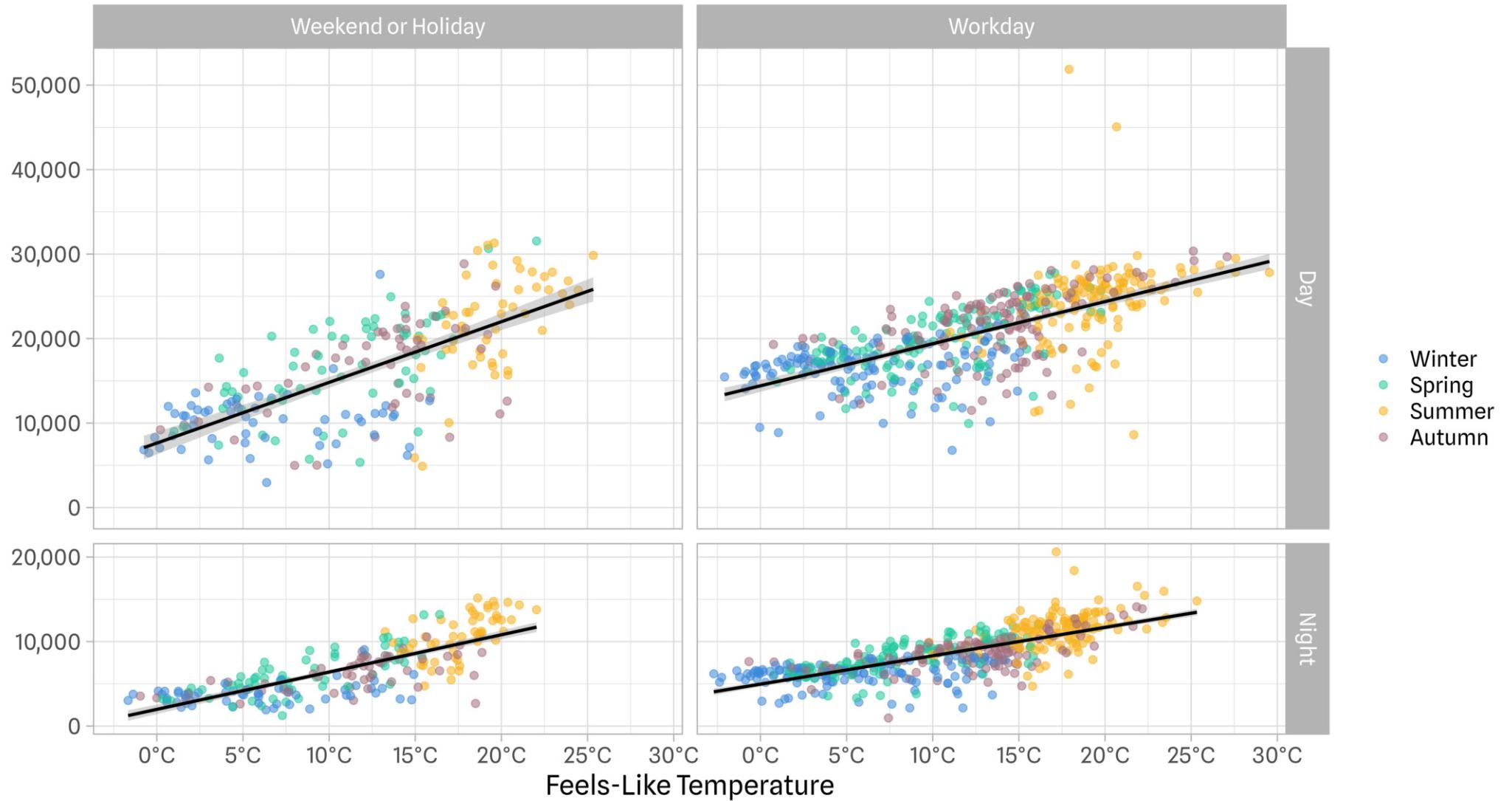
Reported TfL bike rents versus feels-like temperature in London, 2015–2016



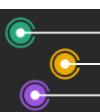
Reported TfL bike rents versus feels-like temperature in London, 2015–2016



Reported TfL bike rents versus feels-like temperature in London, 2015–2016

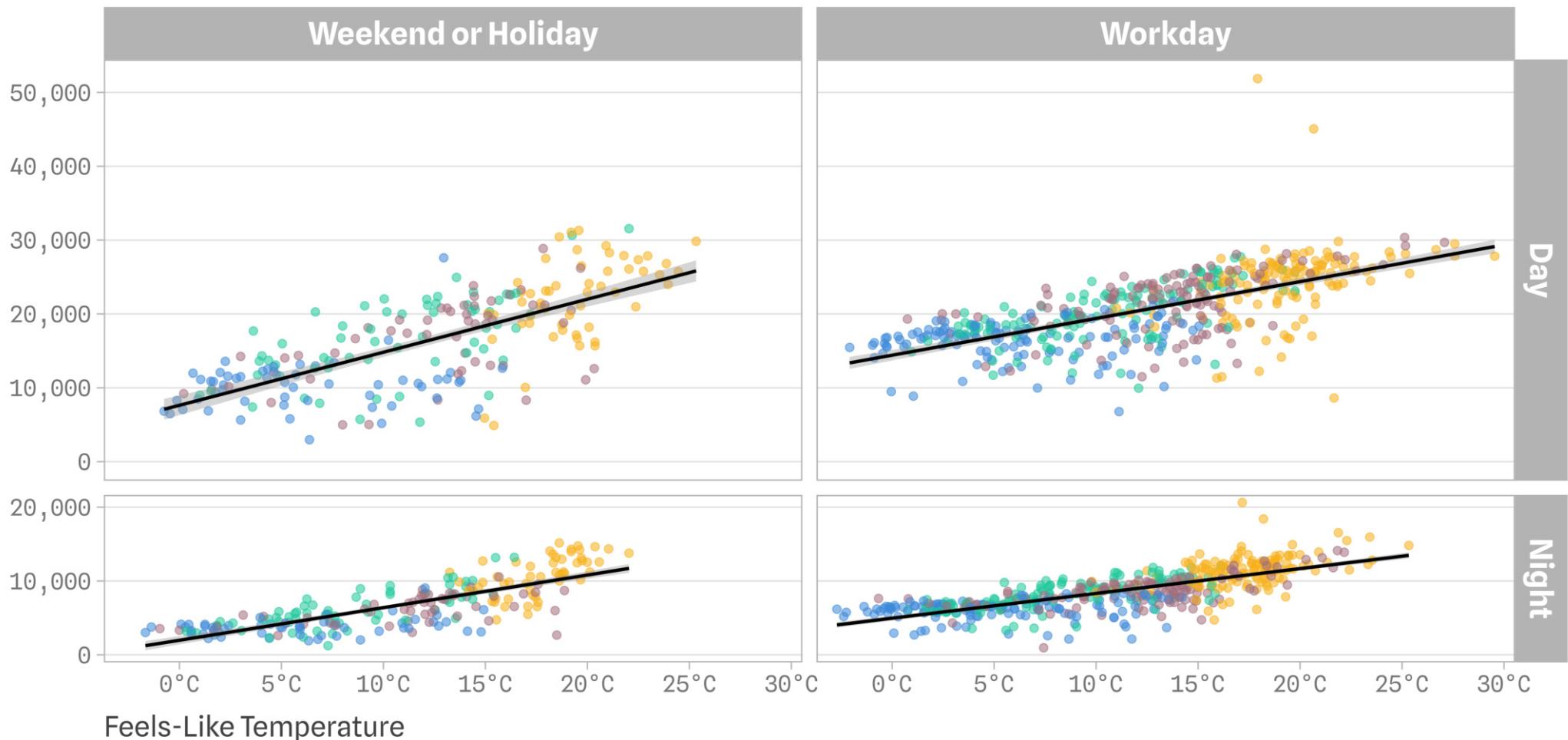


Data: Transport for London (TfL), Jan 2015—Dec 2016



Reported TfL bike rents versus feels-like temperature in London, 2015–2016

• Winter • Spring • Summer • Autumn



Data: Transport for London (TfL), Jan 2015—Dec 2016



Setup



The ggplot2 Package

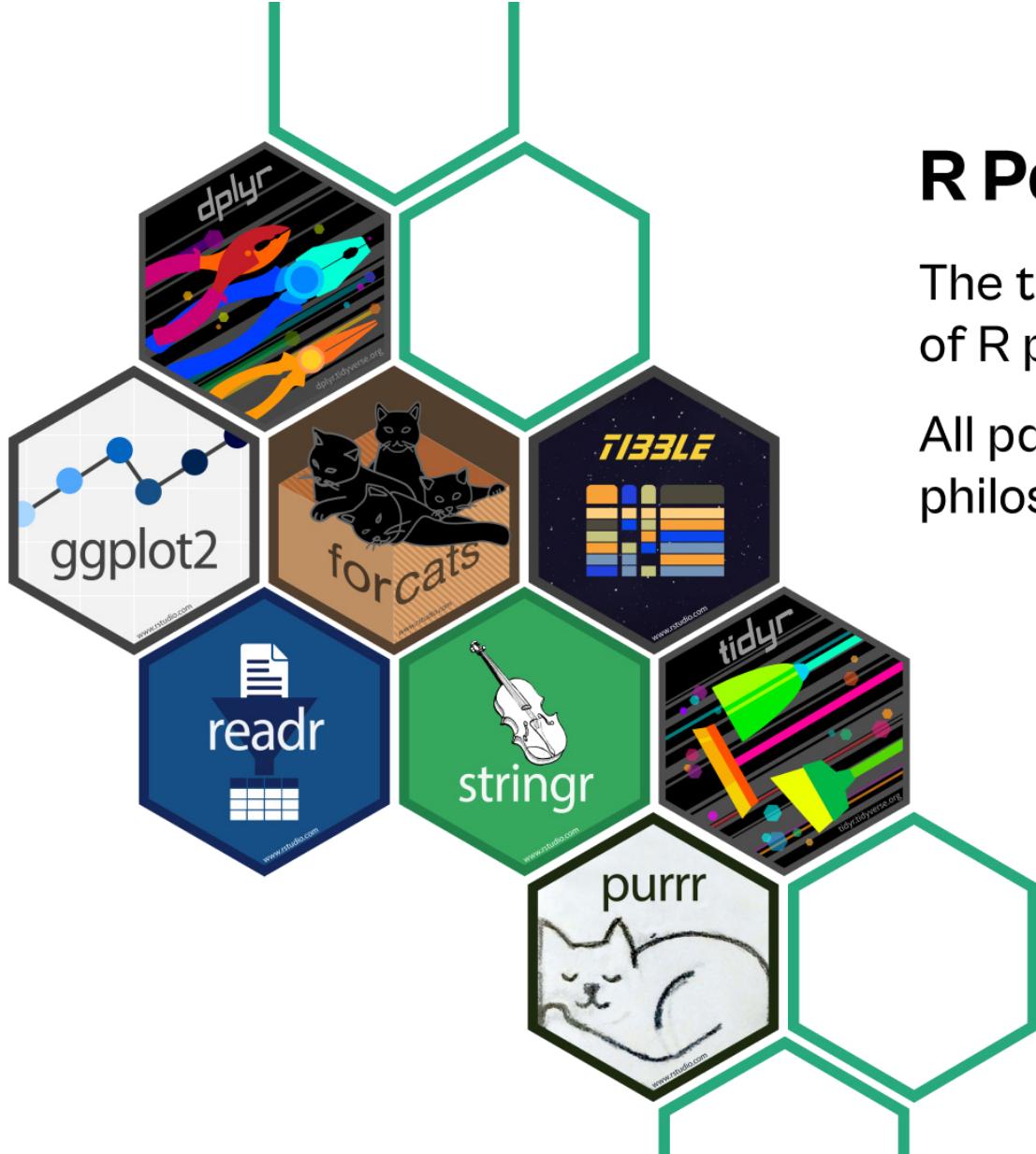
... is an **R package to visualize data** created by Hadley Wickham in 2005

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

... is part of the **{tidyverse}**

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



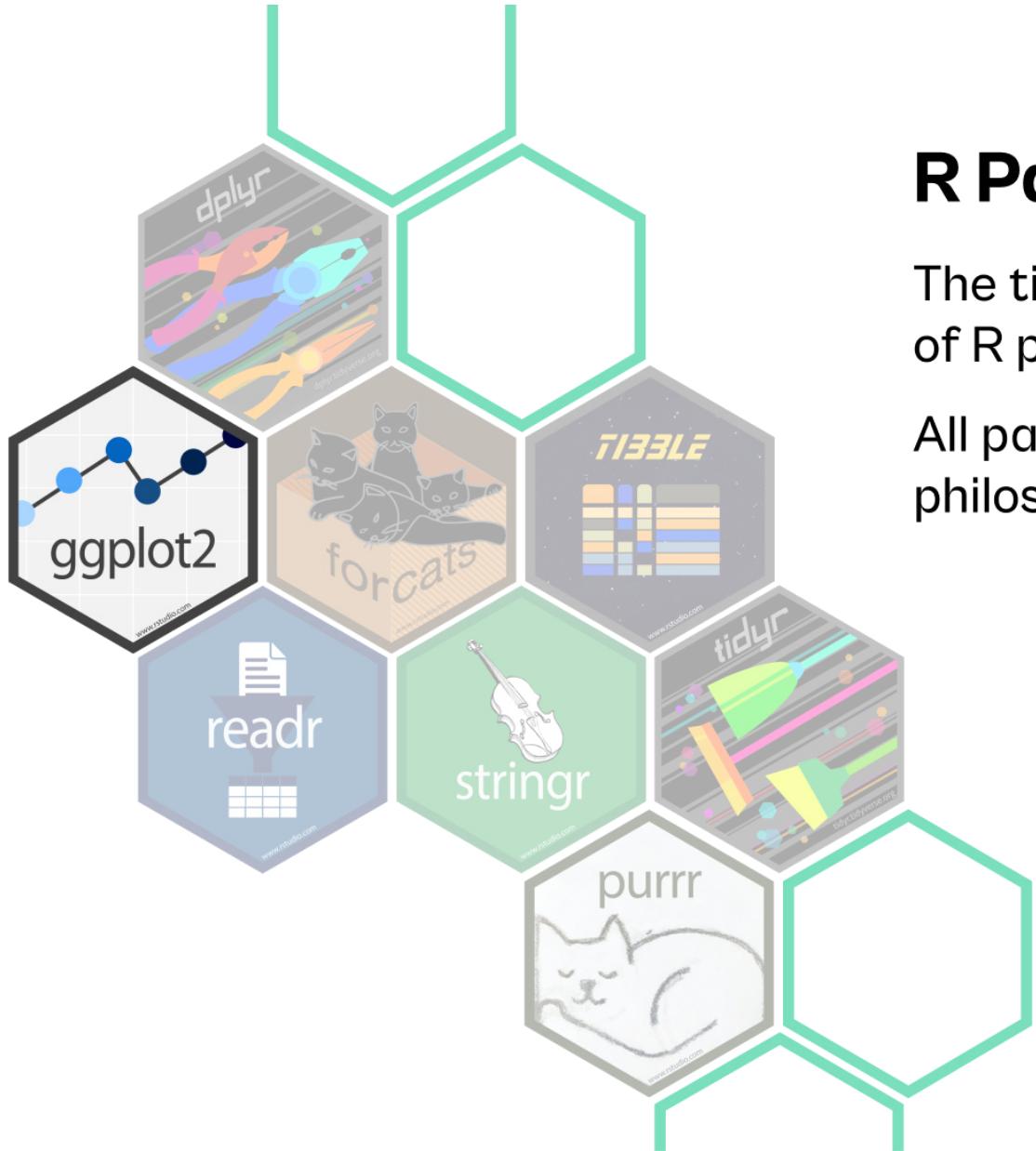


R Packages for Data Science

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

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





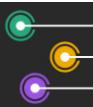
R Packages for Data Science

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

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



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>	<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>
Statistics	<code>stat_*</code> ()	<i>The statistical transformations applied to the data.</i>
Scales	<code>scale_*</code> ()	<i>Maps between the data and the aesthetic dimensions.</i>
Facets	<code>facet_*</code> ()	<i>The arrangement of the data into a grid of plots.</i>
Coordinate System	<code>coord_*</code> ()	<i>Maps data into the plane of the data rectangle.</i>
Visual Themes	<code>theme()</code> and <code>theme_*</code> ()	<i>The overall visual defaults of a plot.</i>



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

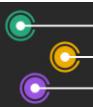
```
1 bikes <- readr::read_csv(  
2   "./data/london-bikes-custom.csv",  
3   ## or: "https://cedricscherer.com/data/london-bikes-custom.csv"  
4   col_types = "Dcffffillllddddc"  
5 )  
6  
7 bikes$season <- forcats::fct_inorder(bikes$season)
```



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



Fundamentals of *{ggplot2}*



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

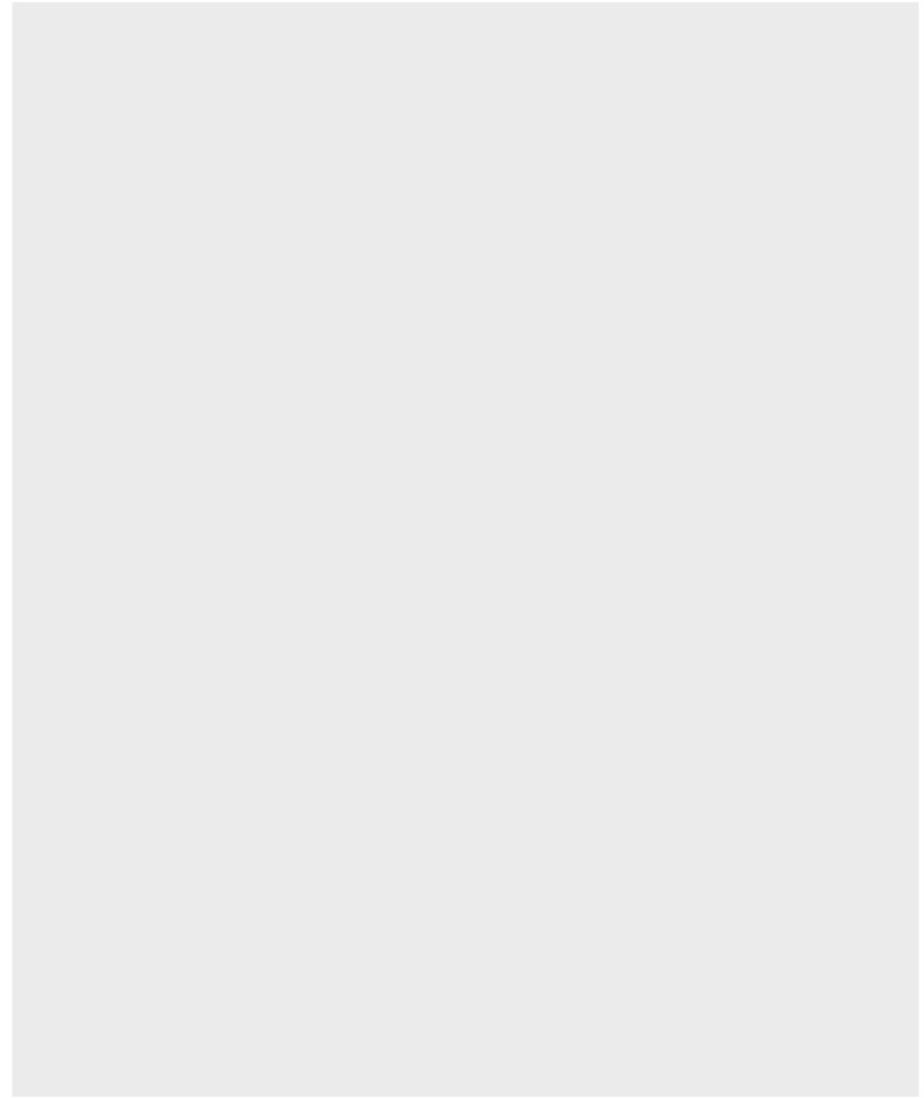
`ggplot()` is used to construct the initial plot object, and is almost always followed by `+` to add component to the plot. There are three common ways to invoke `ggplot()`:

- `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

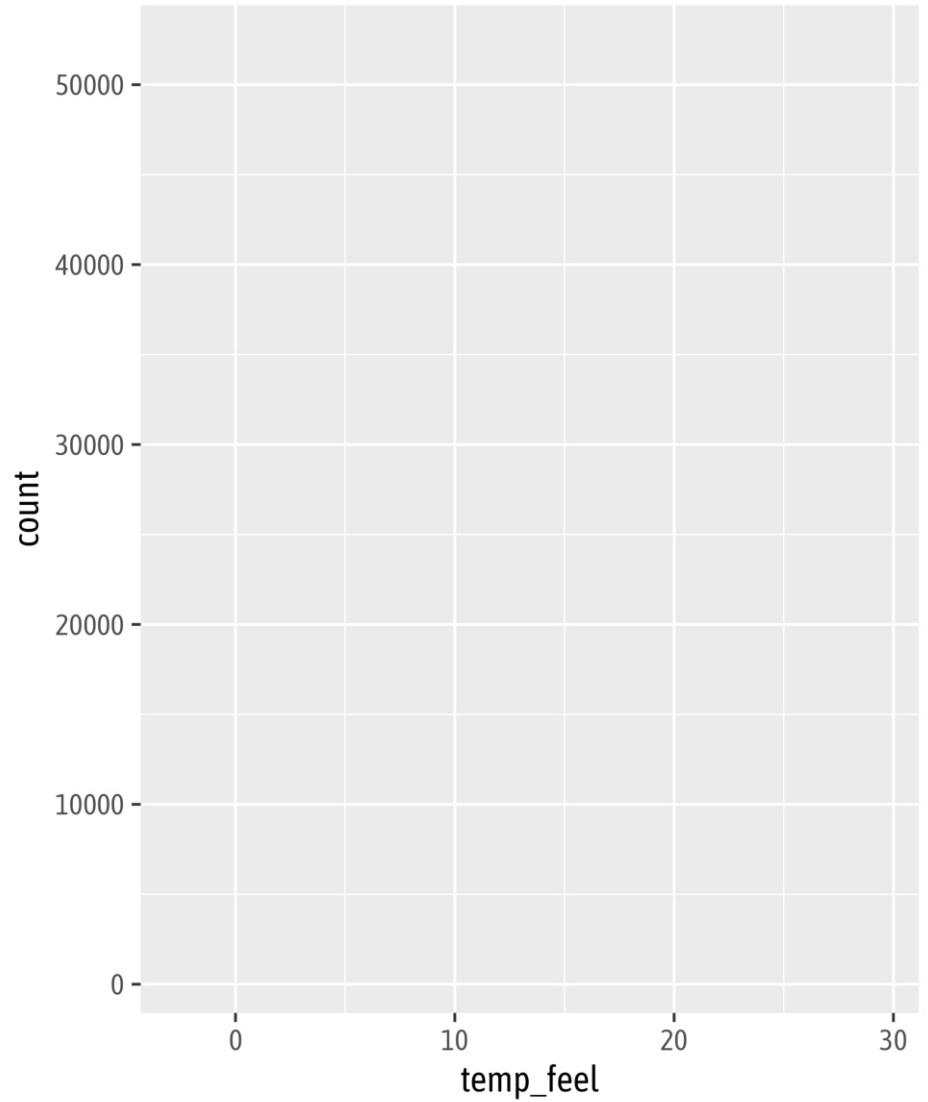
= link variables to graphical properties

- feels-like temperature \rightleftharpoons x
- reported bike shares \rightleftharpoons y
- season \rightleftharpoons color
- year \rightleftharpoons shape
- ...



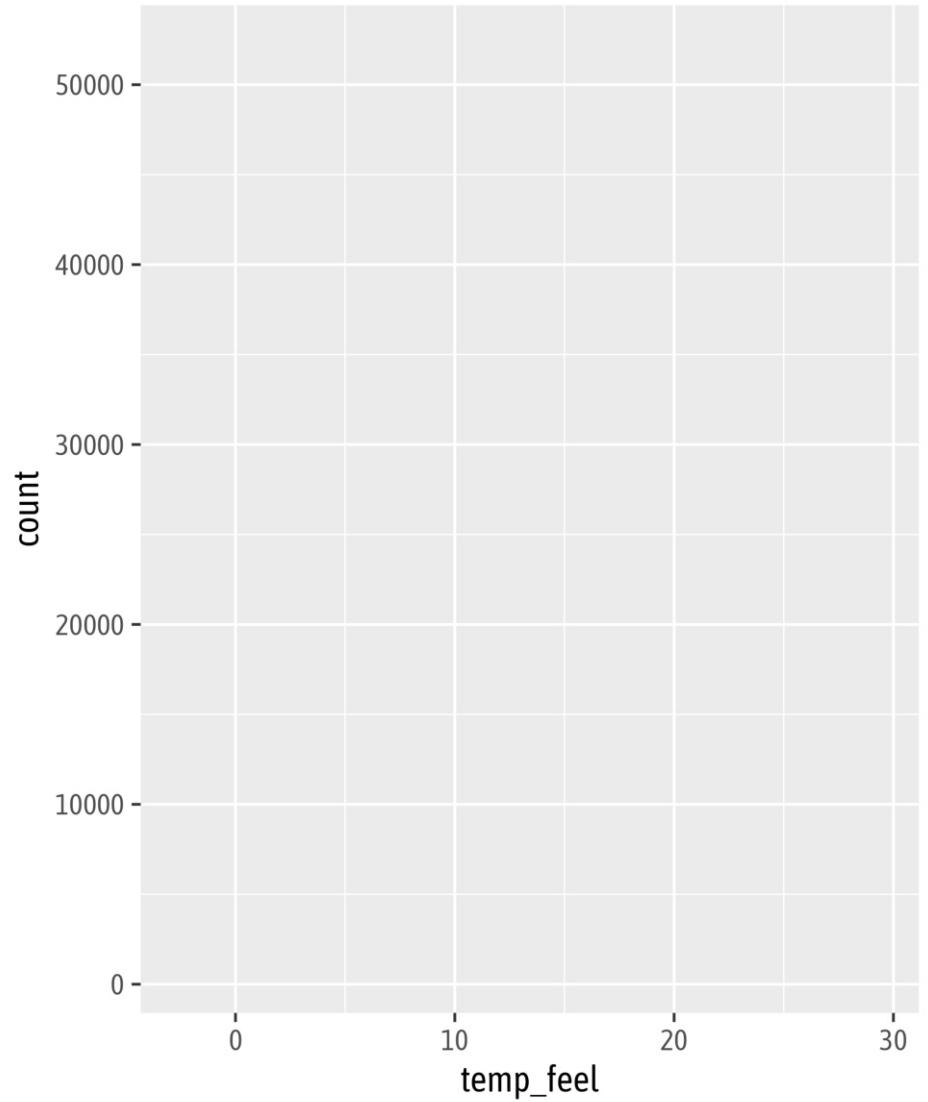
Aesthetic Mapping

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



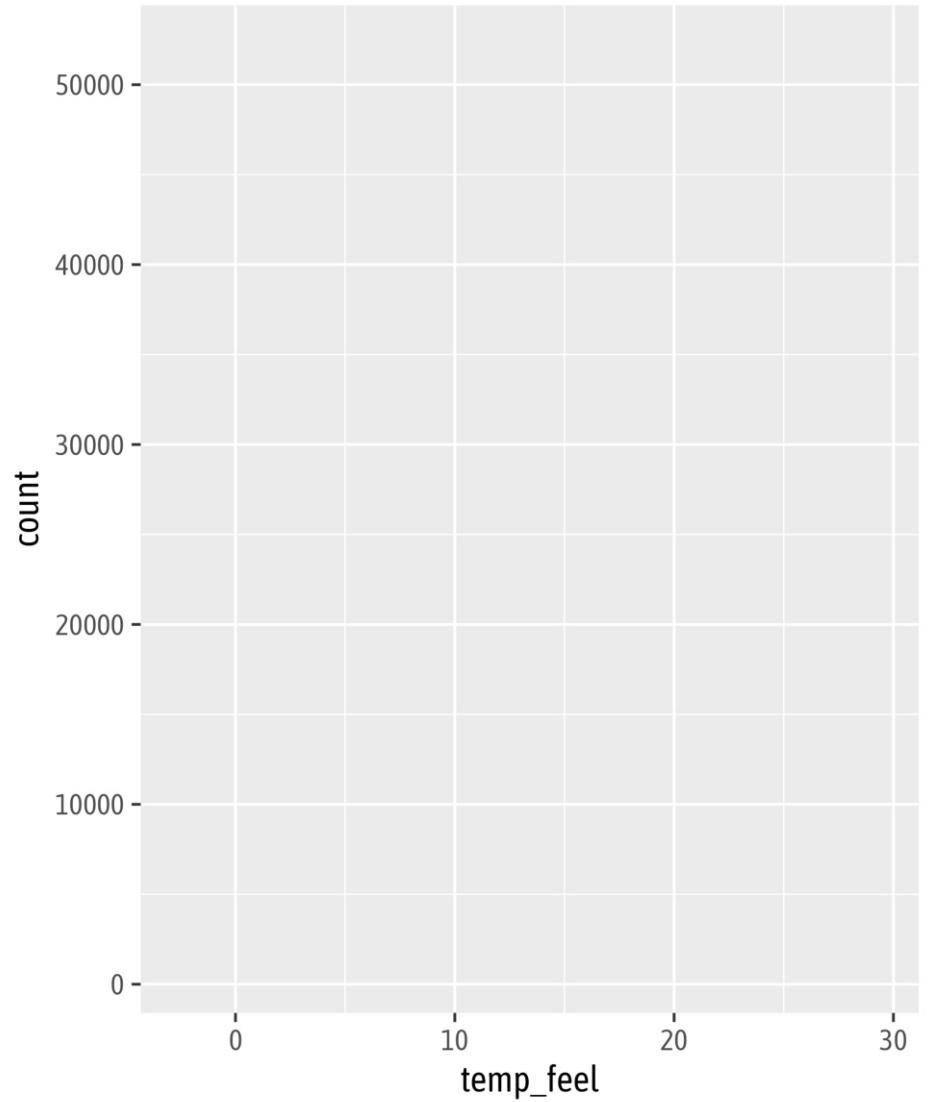
Aesthetic Mapping

```
1 ggplot(  
2   data = bikes,  
3   mapping = aes(x = temp_feel, y = count))  
4 )
```

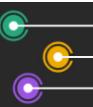


Aesthetic Mapping

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



Geometrical Layers



Geometries

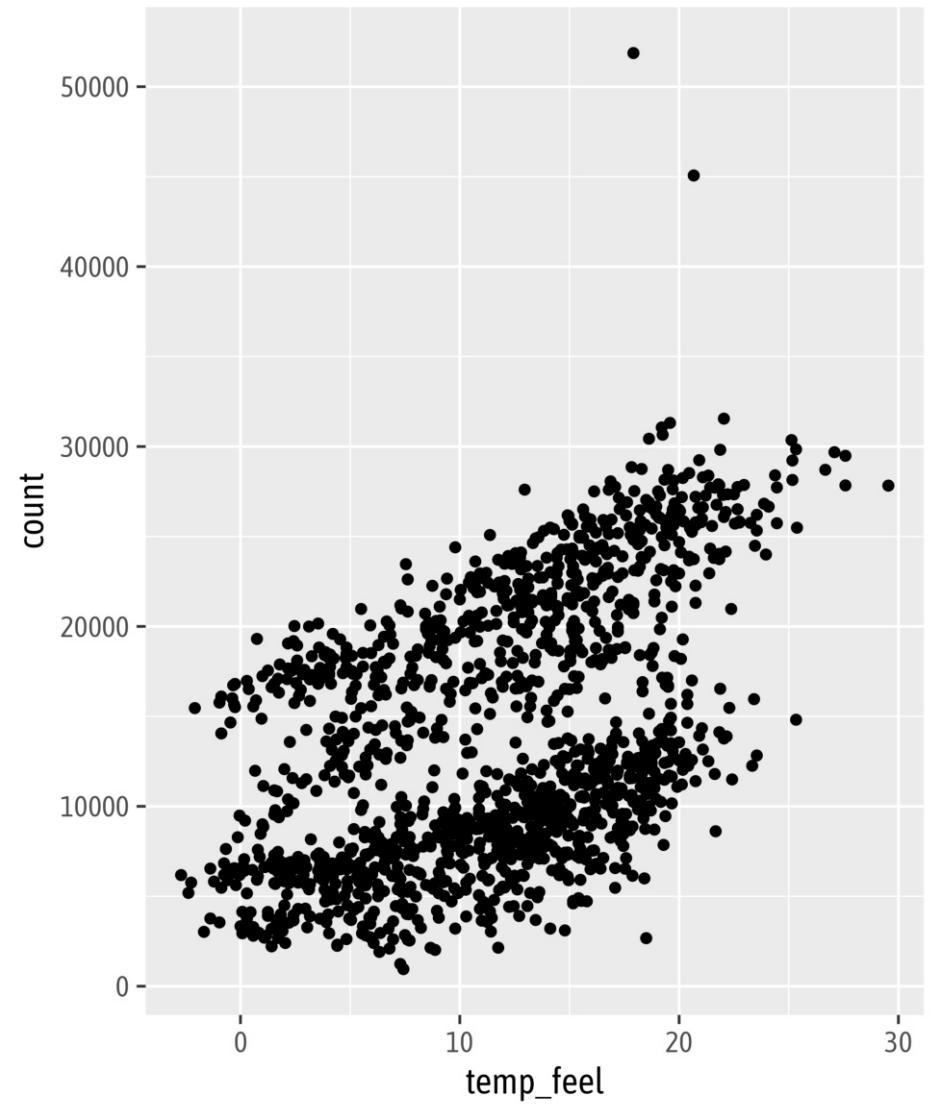
= interpret aesthetics as graphical representations

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



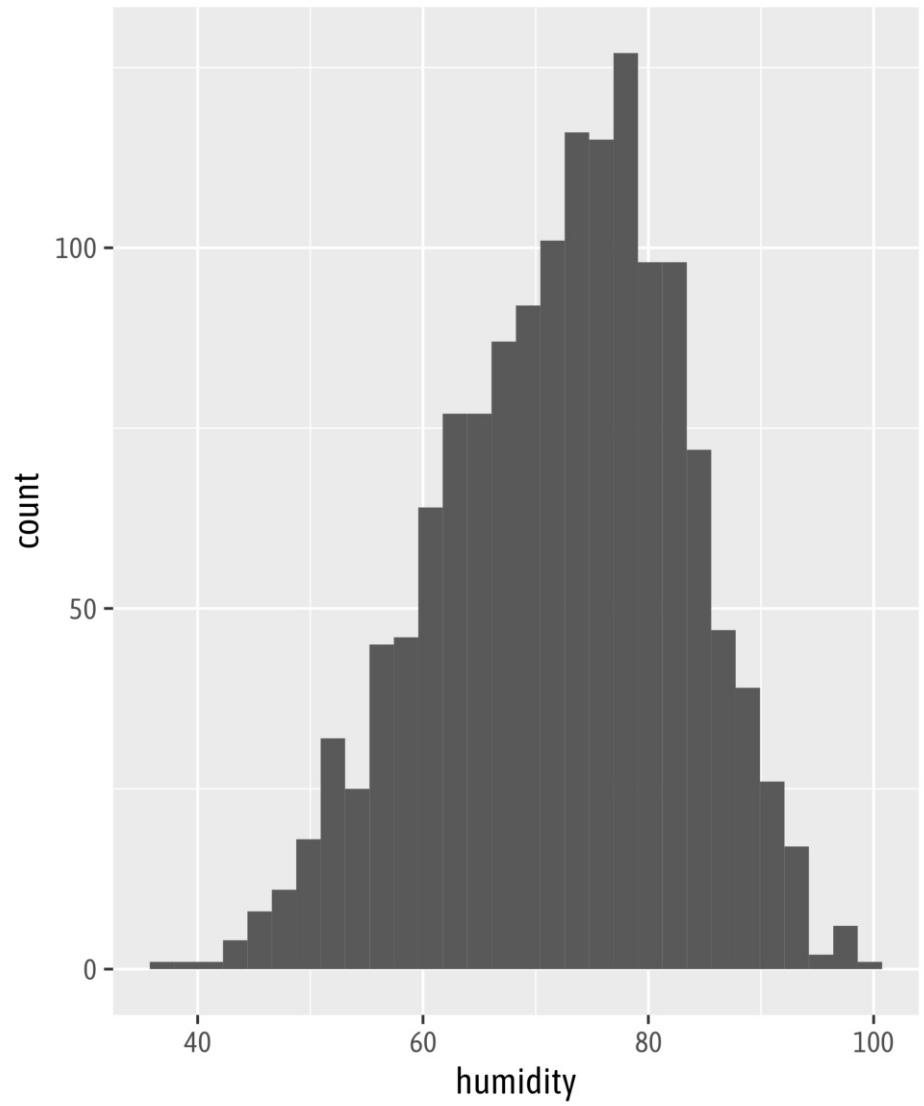
Geometries

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



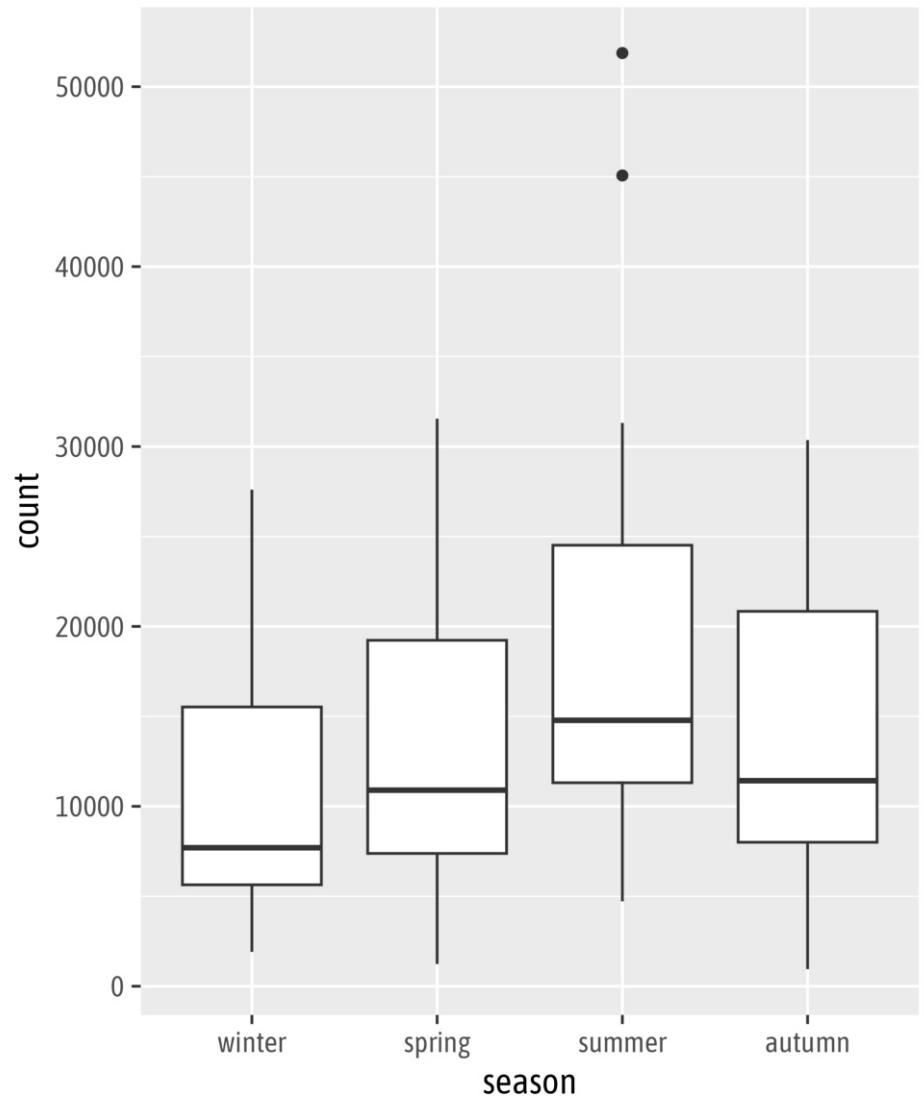
Geometries

```
1 ggplot(  
2   bikes,  
3   aes(x = humidity)  
4 ) +  
5 geom_histogram()
```



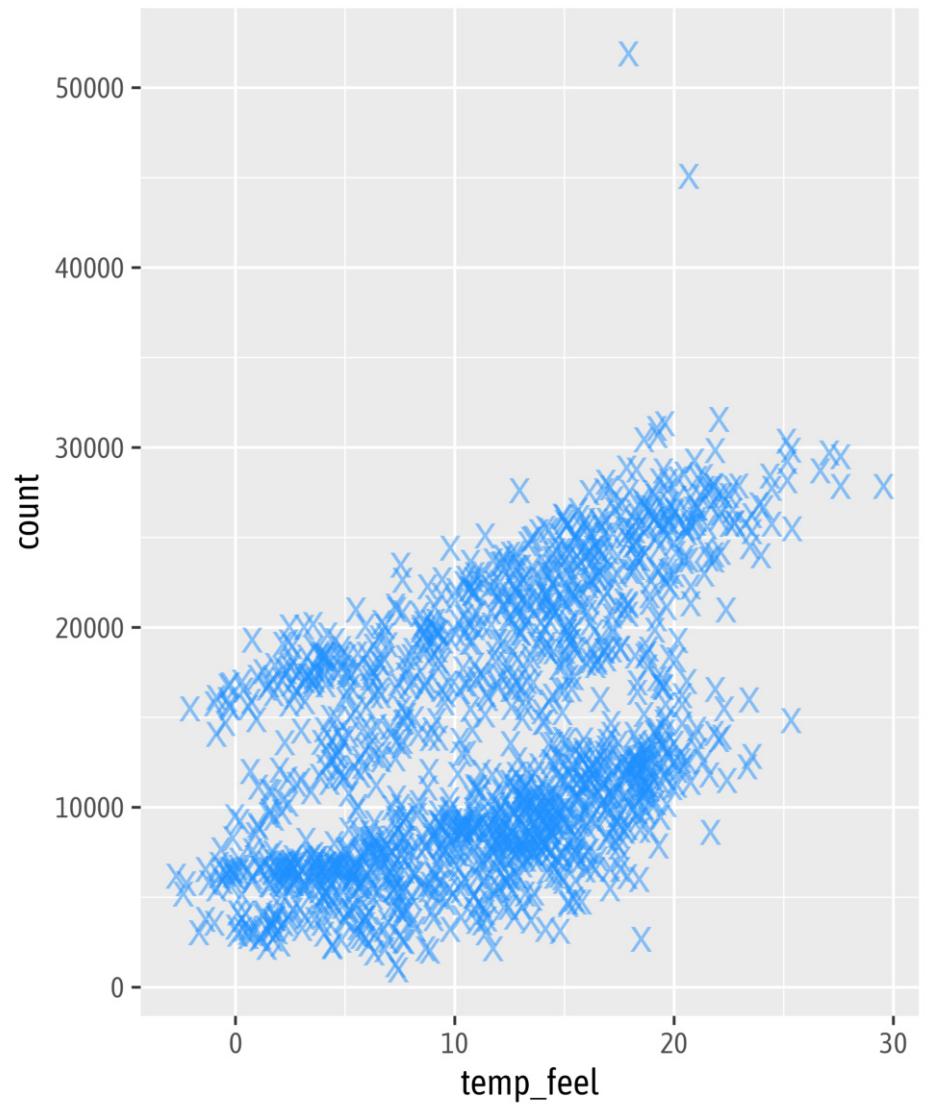
Geometries

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



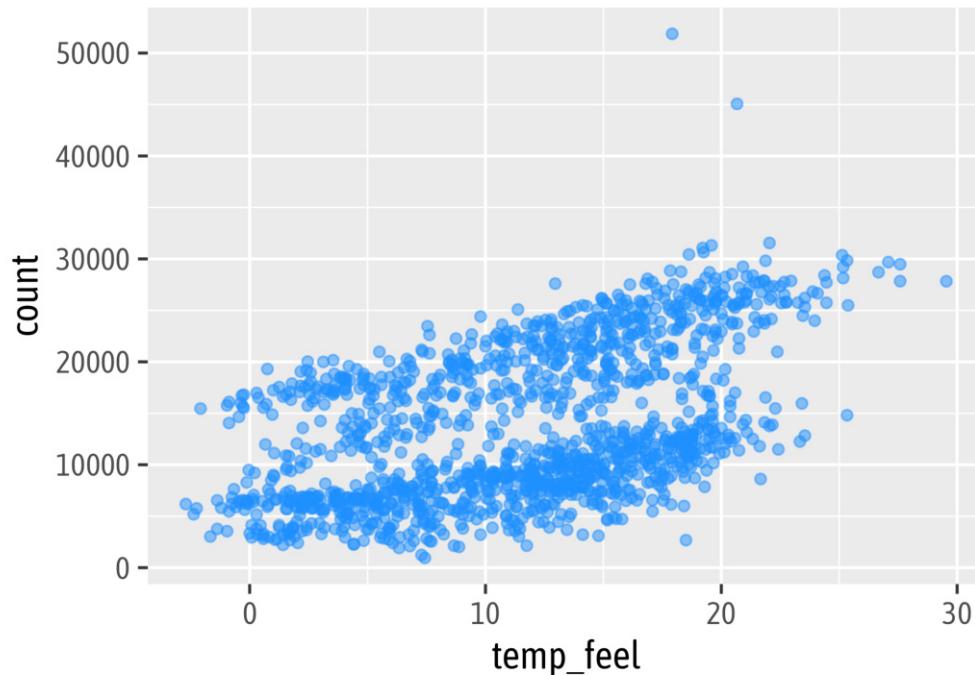
Visual Properties of Layers

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

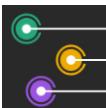
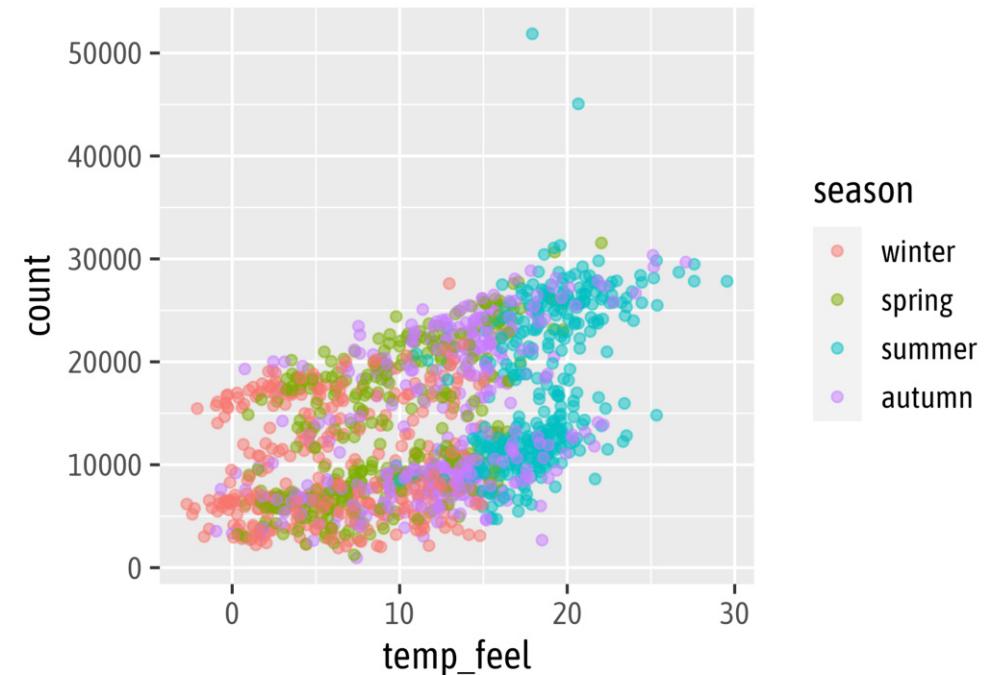


Setting vs Mapping of Visual Properties

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

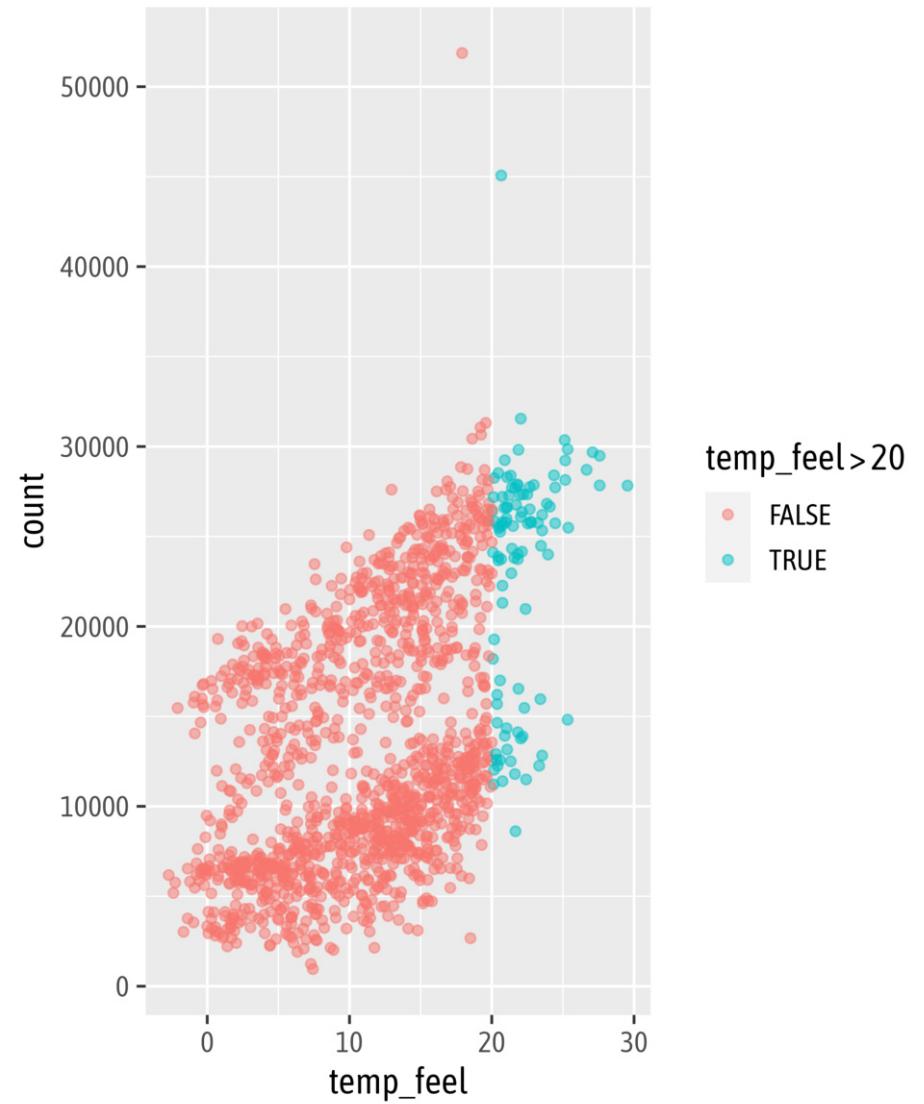


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5   geom_point(  
6   aes(color = season),  
7   alpha = .5  
8 )
```



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 )
```



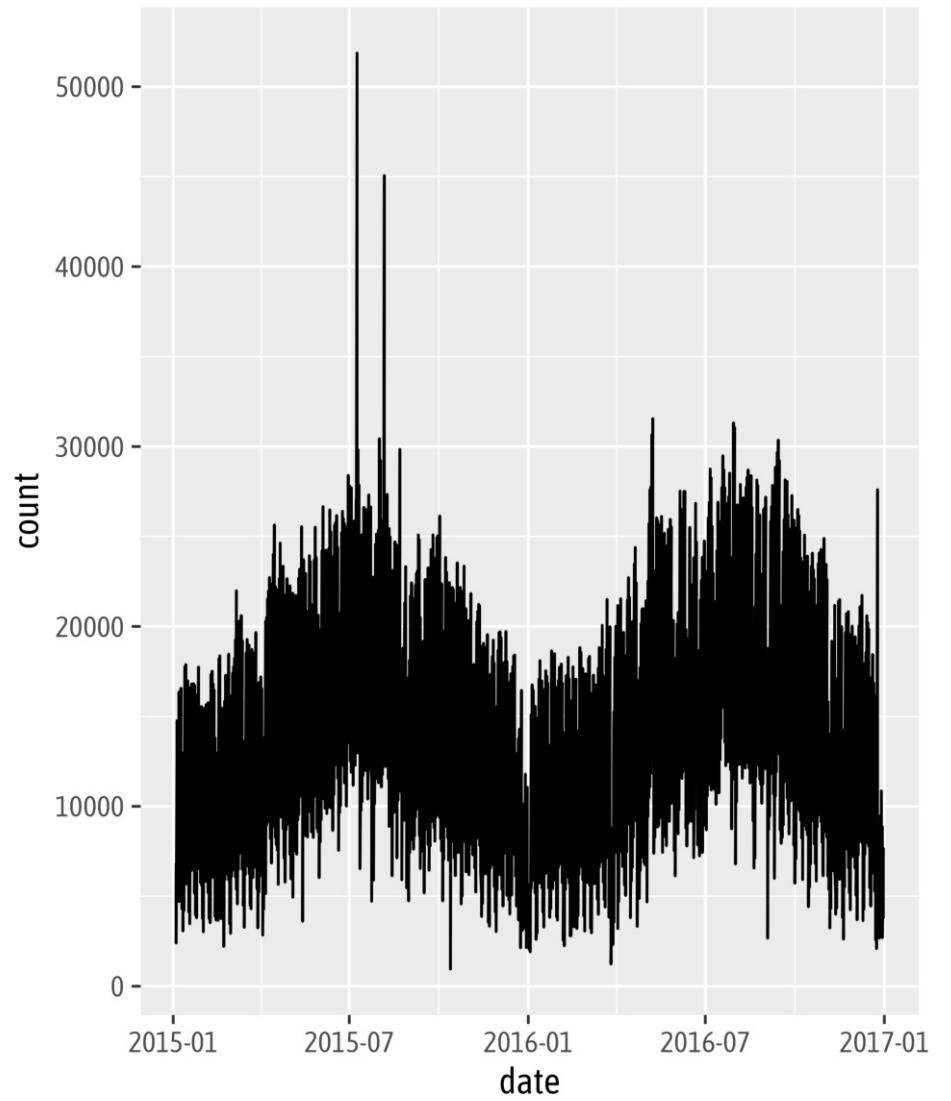
Your Turn

- Create a chart showing a time series of reported bike shares
 - What is the difference between `geom_line()` and `geom_path()`?
 - Map the variable `day_night` to the color of the lines.
 - Add points for each observation, encoded by the same variable.
 - Turn the points into diamonds.



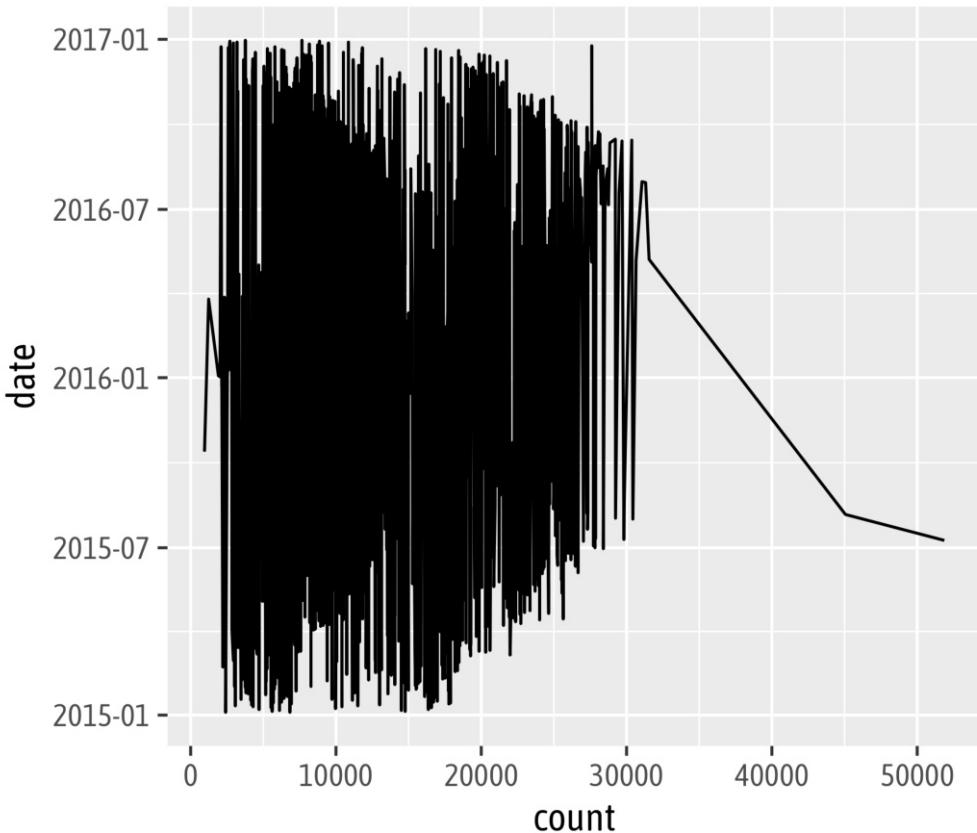
Solution Exercise

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

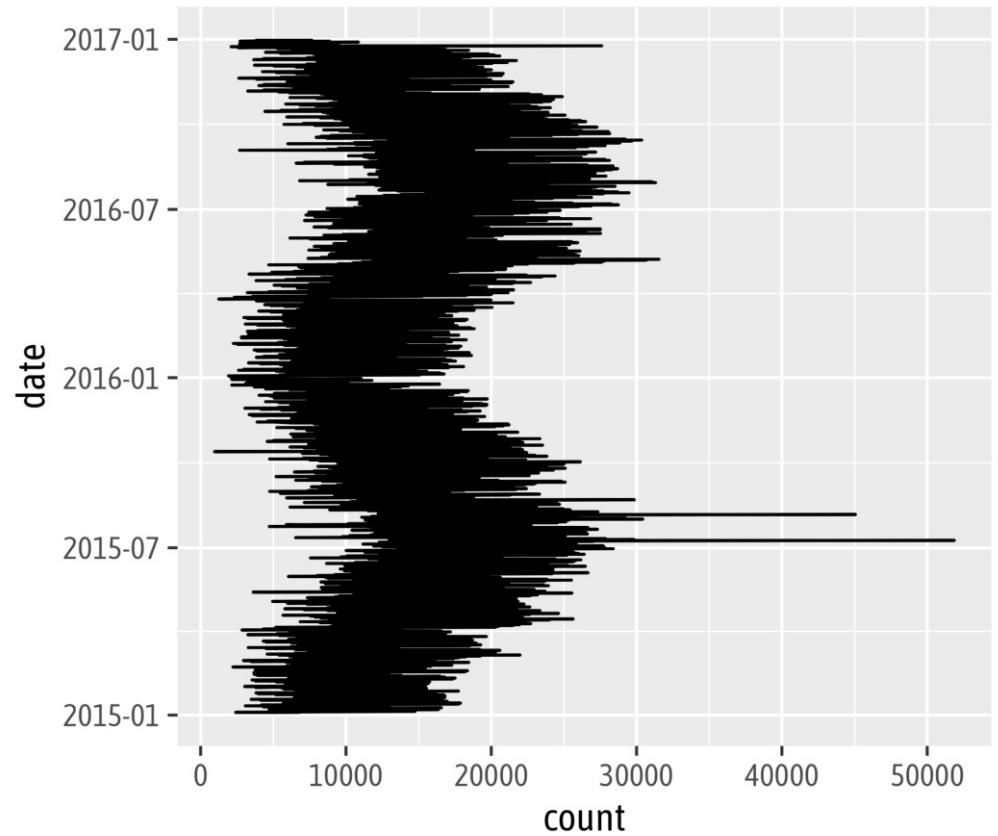


Solution Exercise

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

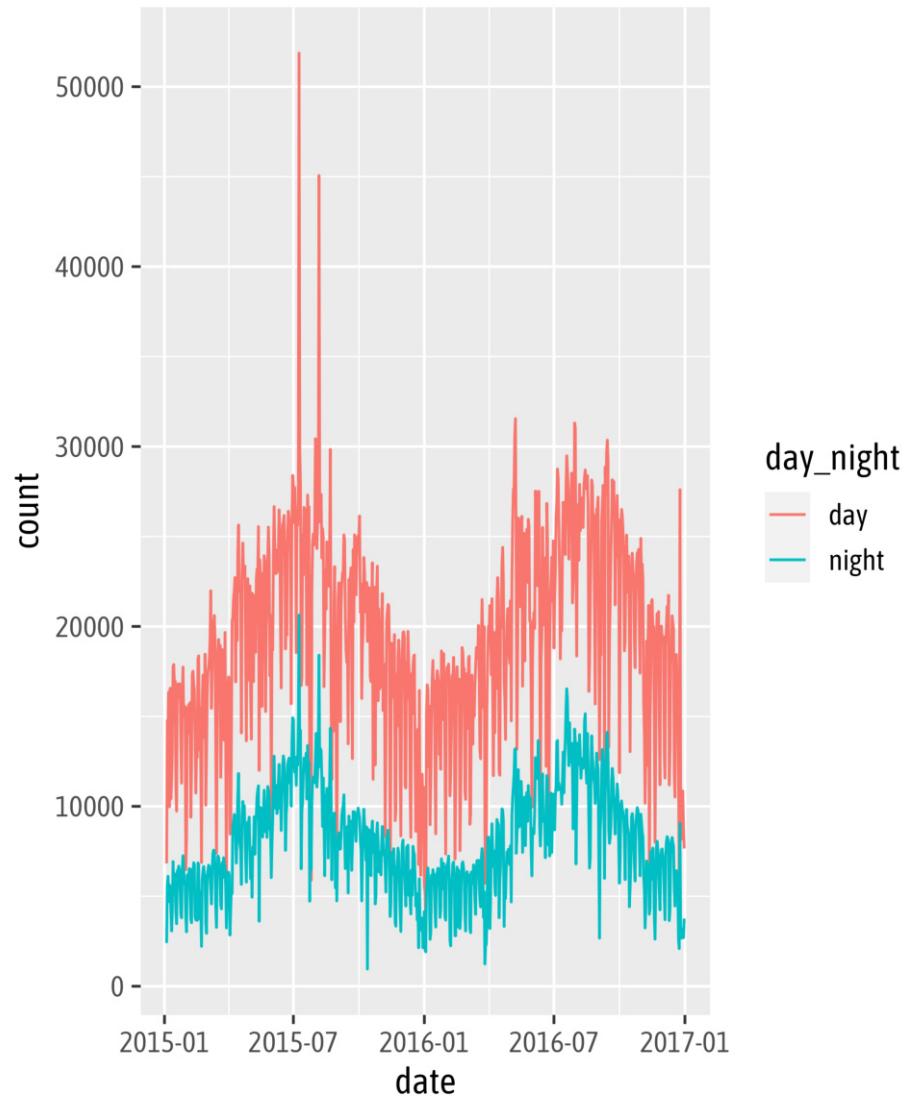


```
1 ggplot(  
2   bikes,  
3   aes(x = count, y = date)  
4 ) +  
5 geom_path()
```



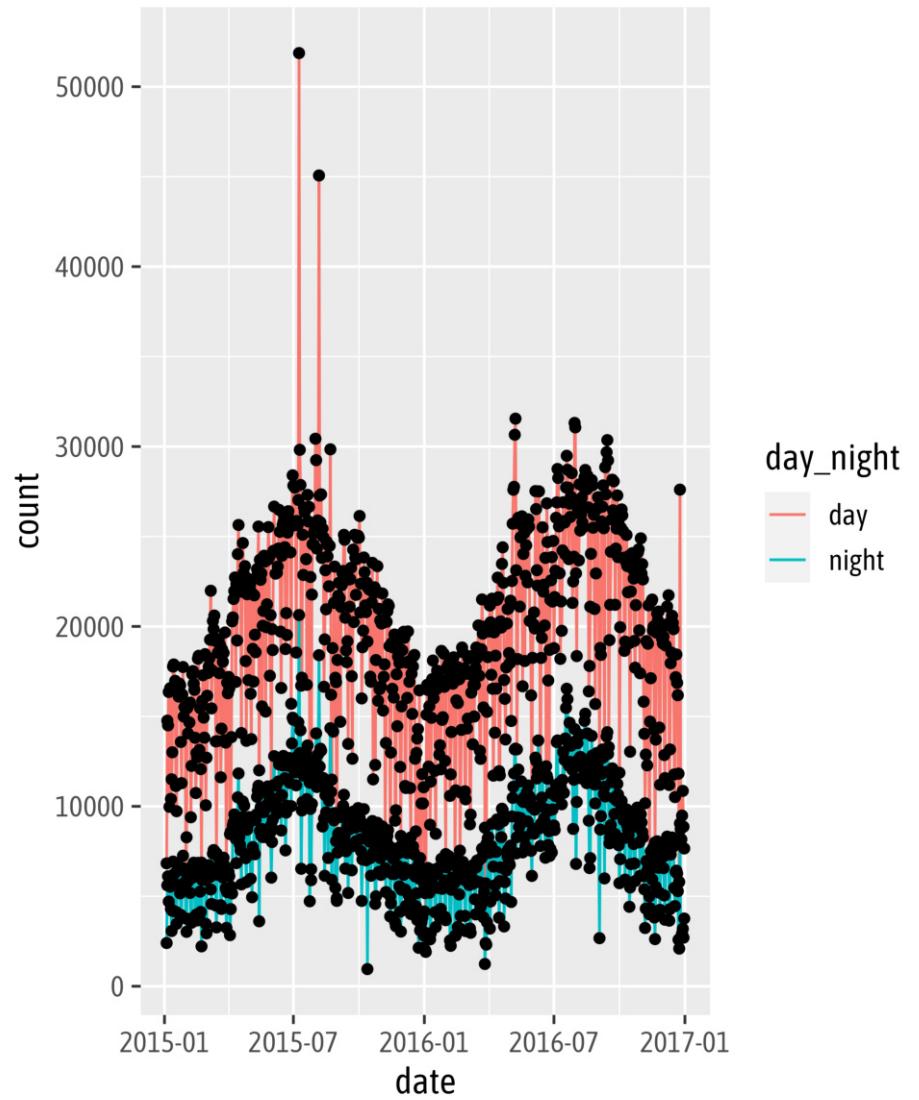
Solution Exercise

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count)  
4 ) +  
5 geom_line(  
6   aes(color = day_night)  
7 )
```



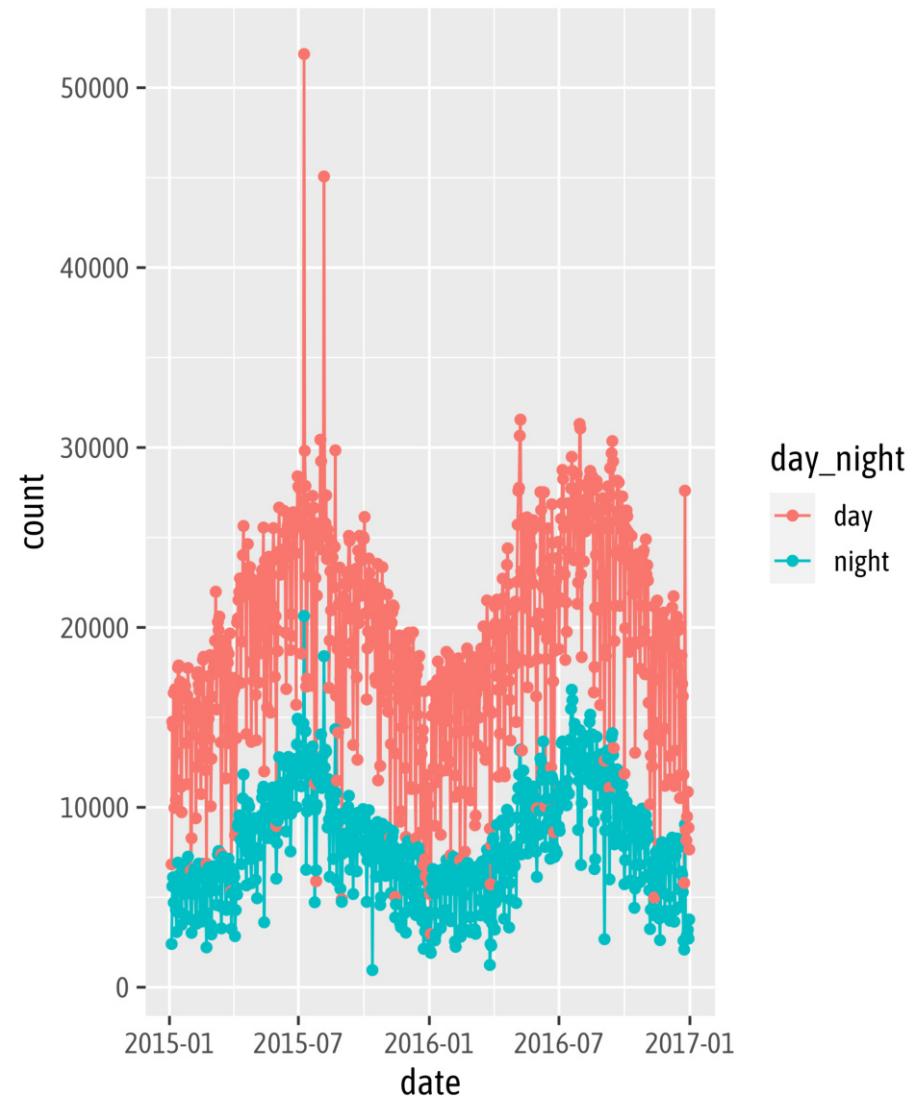
Solution Exercise

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count)  
4 ) +  
5 geom_line(  
6   aes(color = day_night)  
7 ) +  
8 geom_point()
```



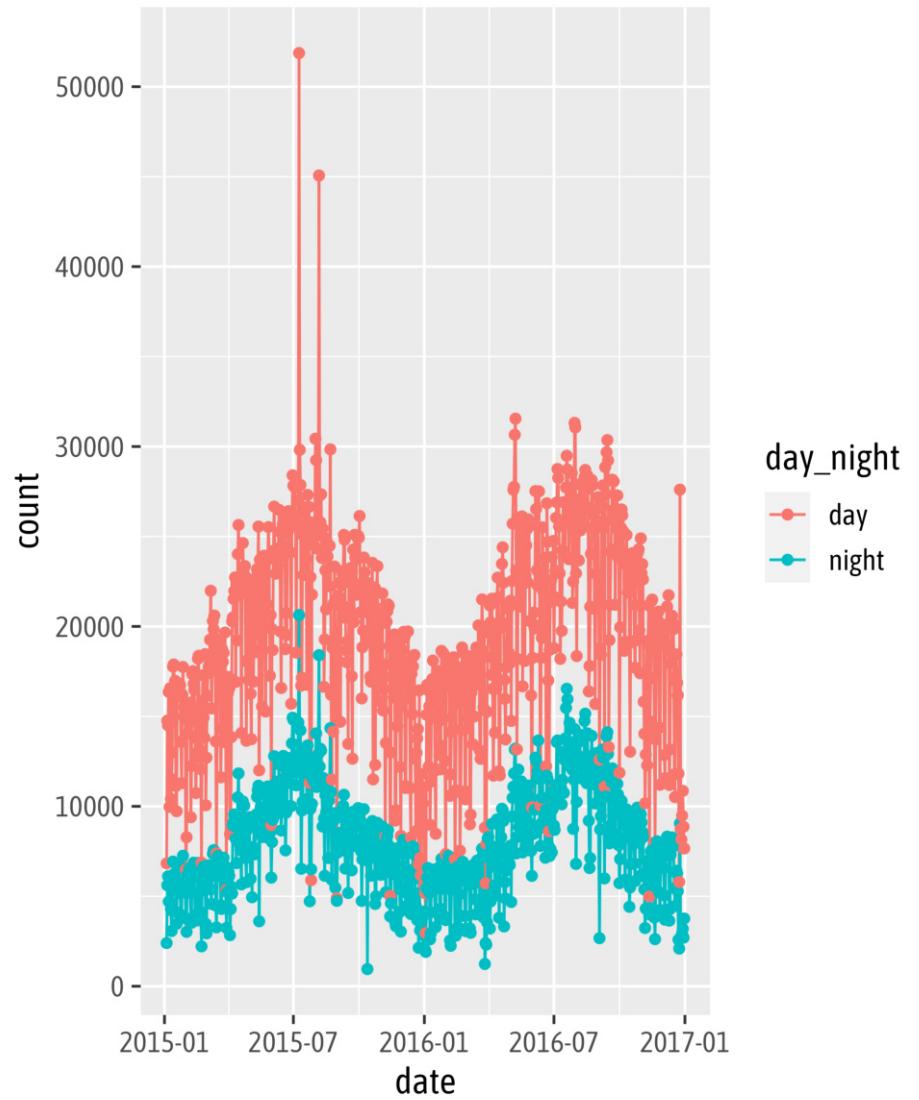
Solution Exercise

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count)  
4 ) +  
5 geom_line(  
6   aes(color = day_night)  
7 ) +  
8 geom_point(  
9   aes(color = day_night)  
10 )
```



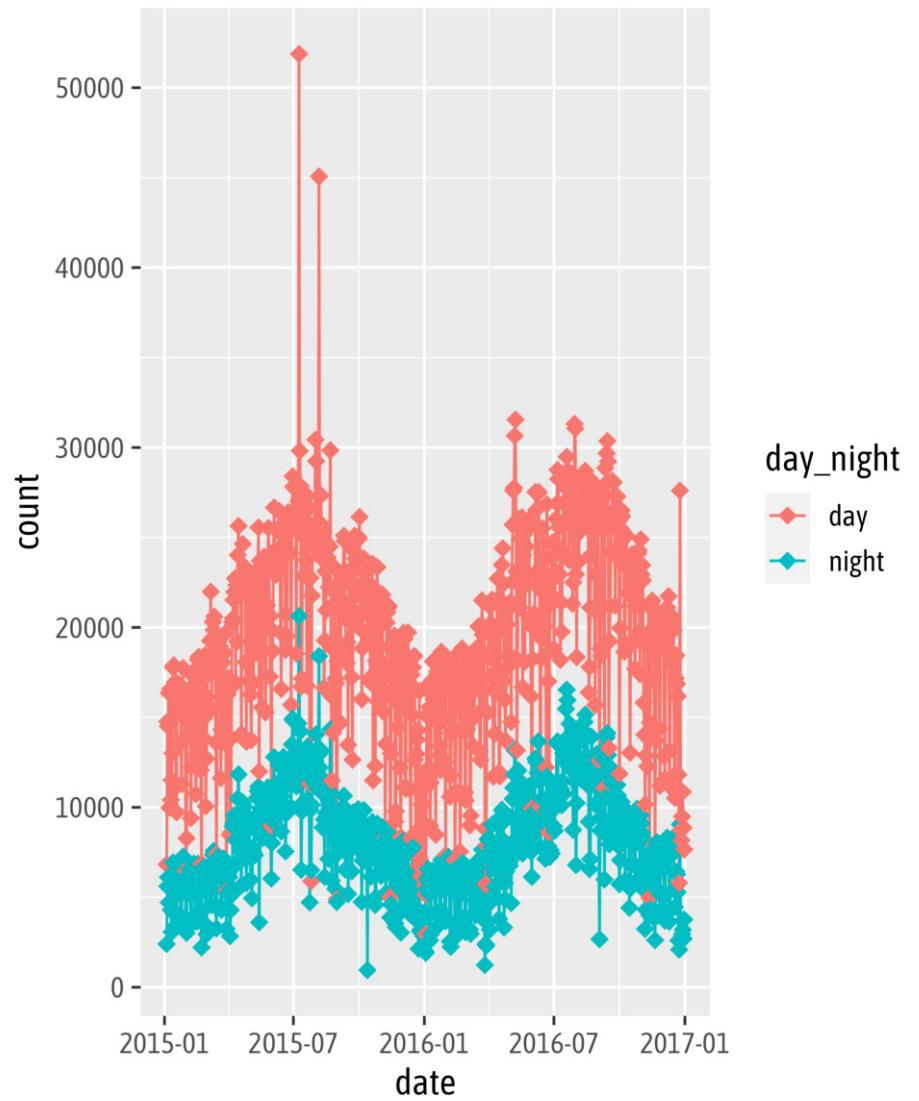
Solution Exercise

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = day_night)  
5 ) +  
6 geom_line() +  
7 geom_point()
```



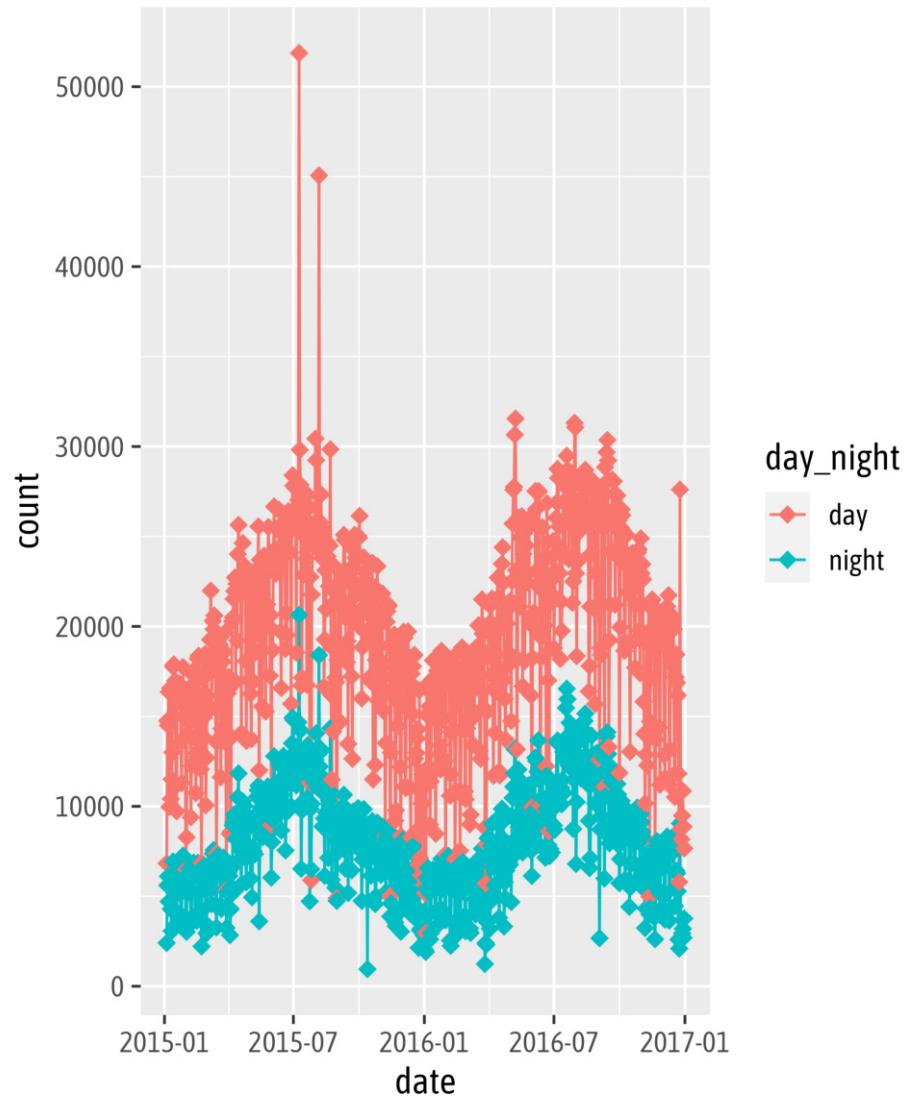
Solution Exercise

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = day_night)  
5 ) +  
6 geom_line() +  
7 geom_point(  
8   shape = "diamond",  
9   size = 3  
10 )
```



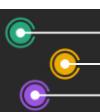
Solution Exercise

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = day_night)  
5 ) +  
6 geom_line() +  
7 geom_point(  
8   shape = 18,  
9   size = 3  
10 )
```



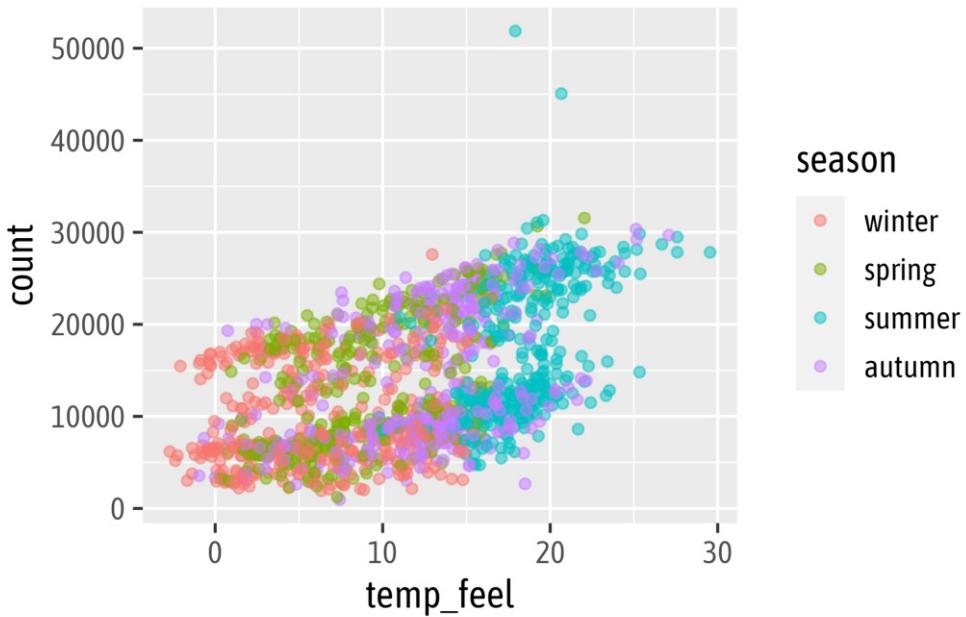
Circles	1 	10 	13 	16 	19 	20 	21
Triangles	2 	6 	17 	24 	25 		
Diamonds	5 	9 	18 	23 			
Squares	0 	7 	12 	14 	15 	22 	
Other	3 	4 	8 	11 			

Source: Albert's Blog

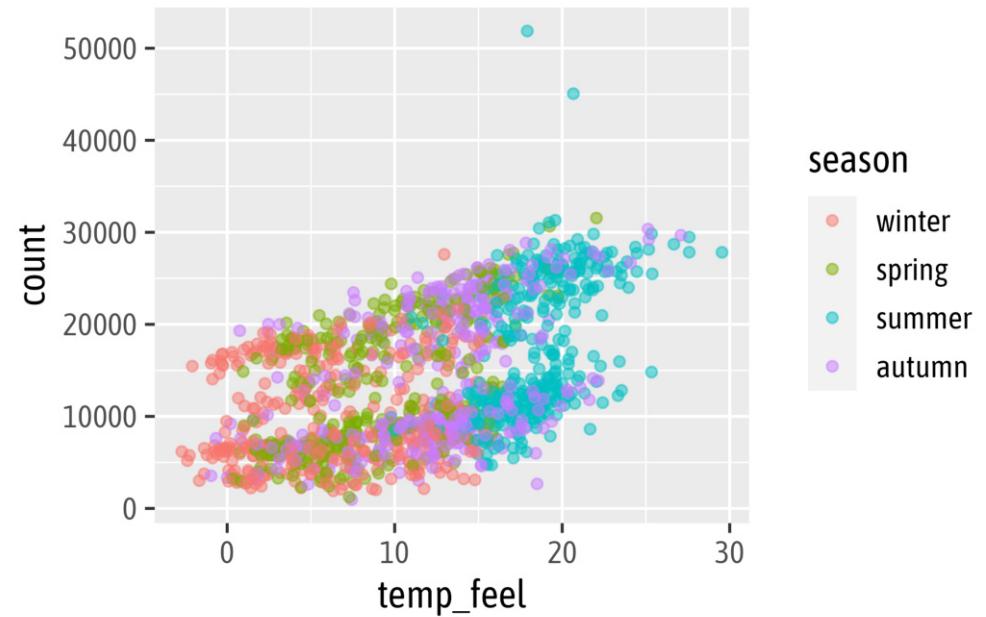


Local vs. Global Encoding

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

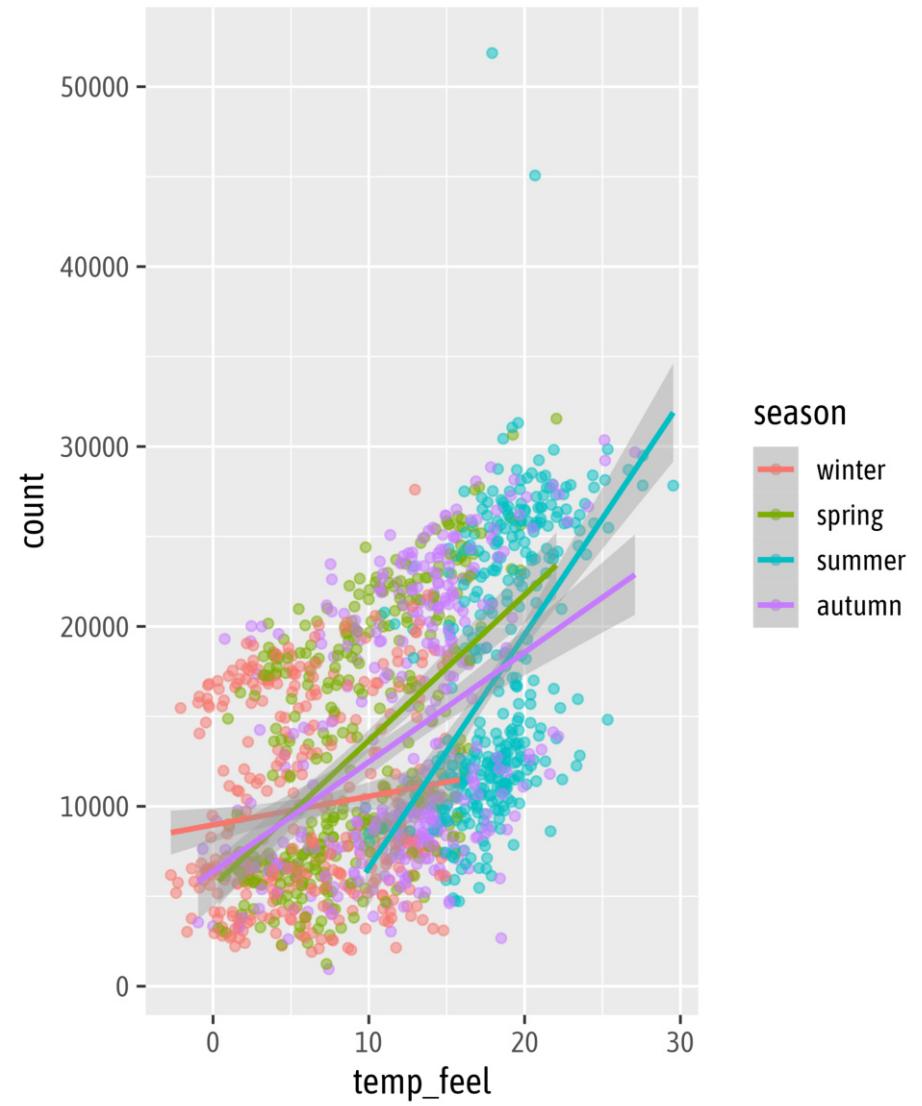


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4         color = season)  
5 ) +  
6   geom_point(  
7   alpha = .5  
8 )
```



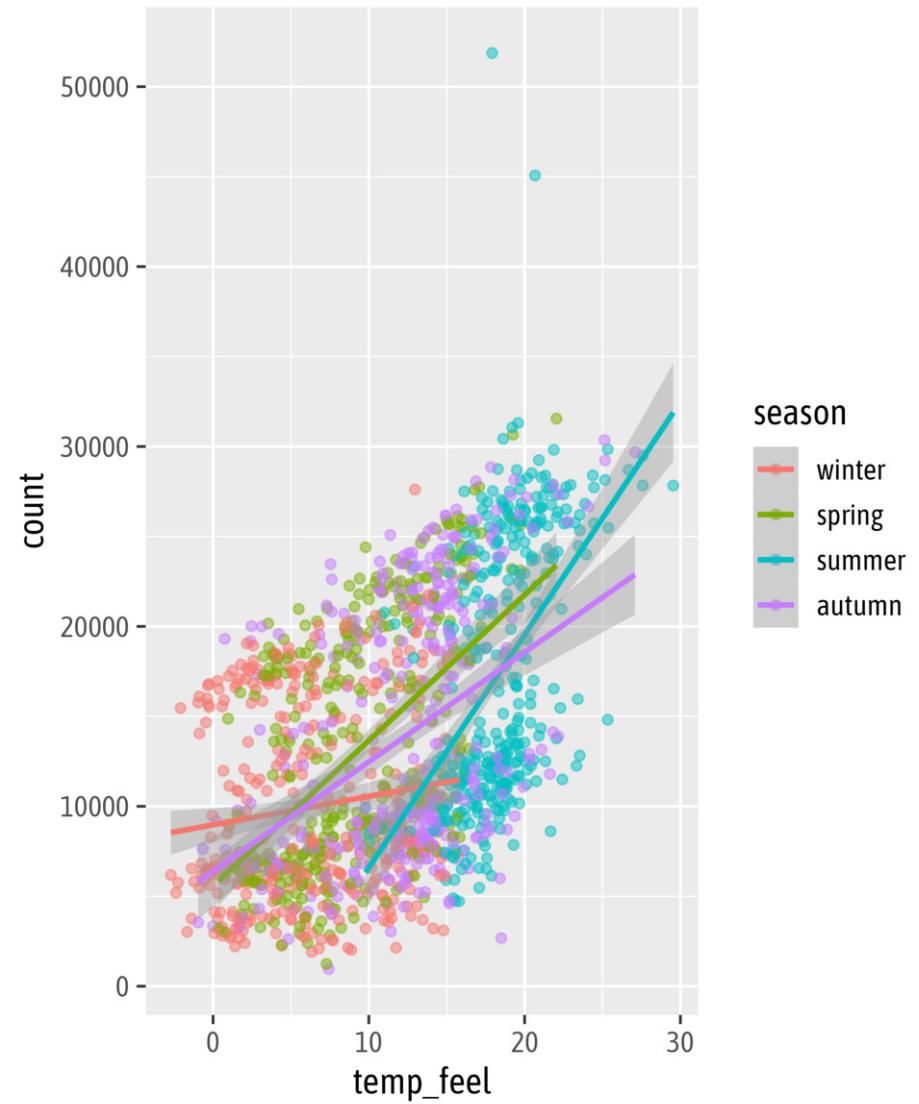
Adding More Layers

```
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 )
```



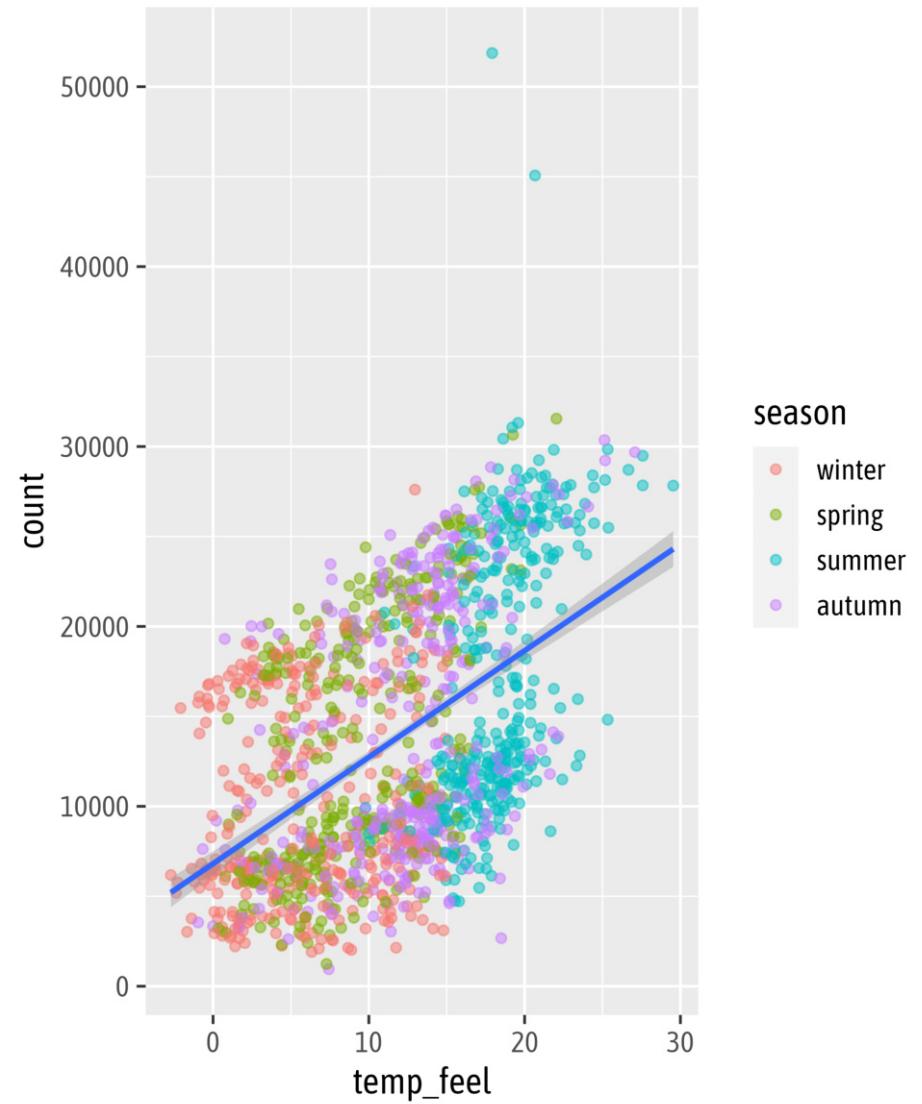
Global Color Encoding

```
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 )
```



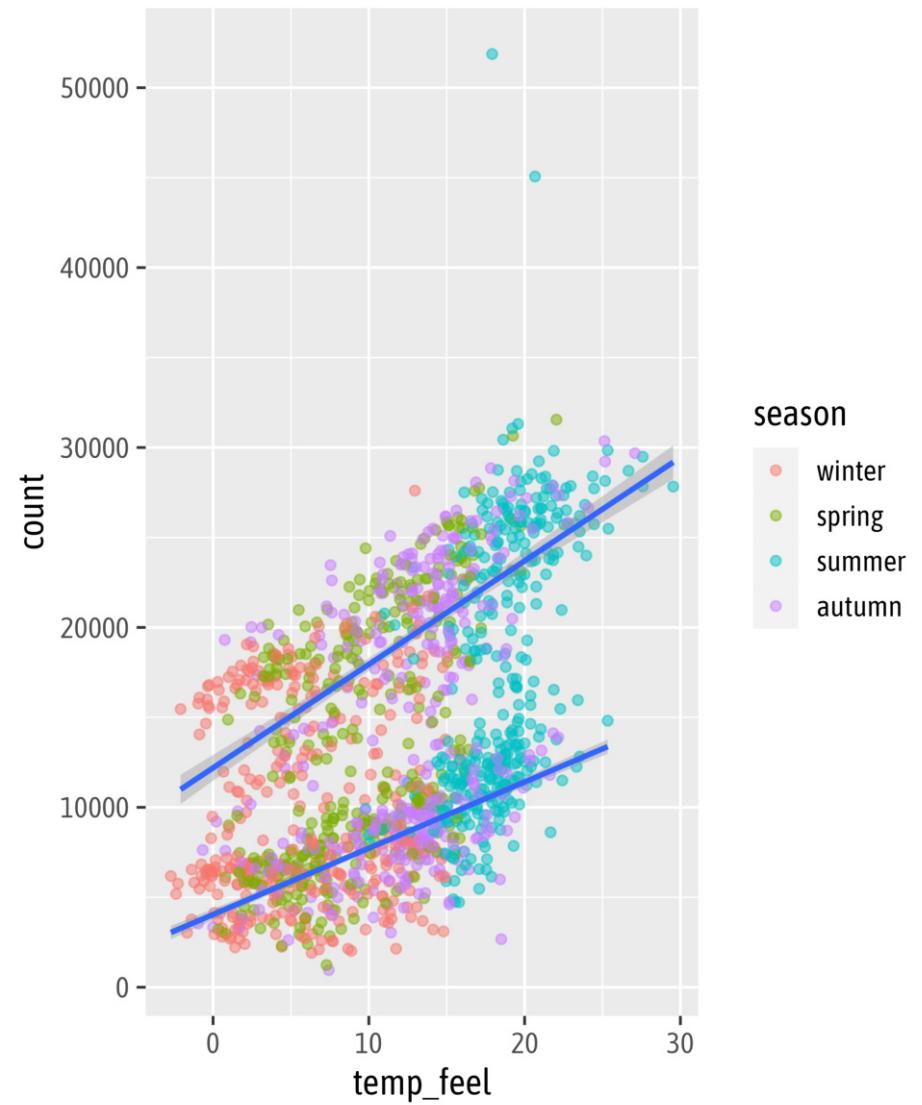
Local Color Encoding

```
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 )
```



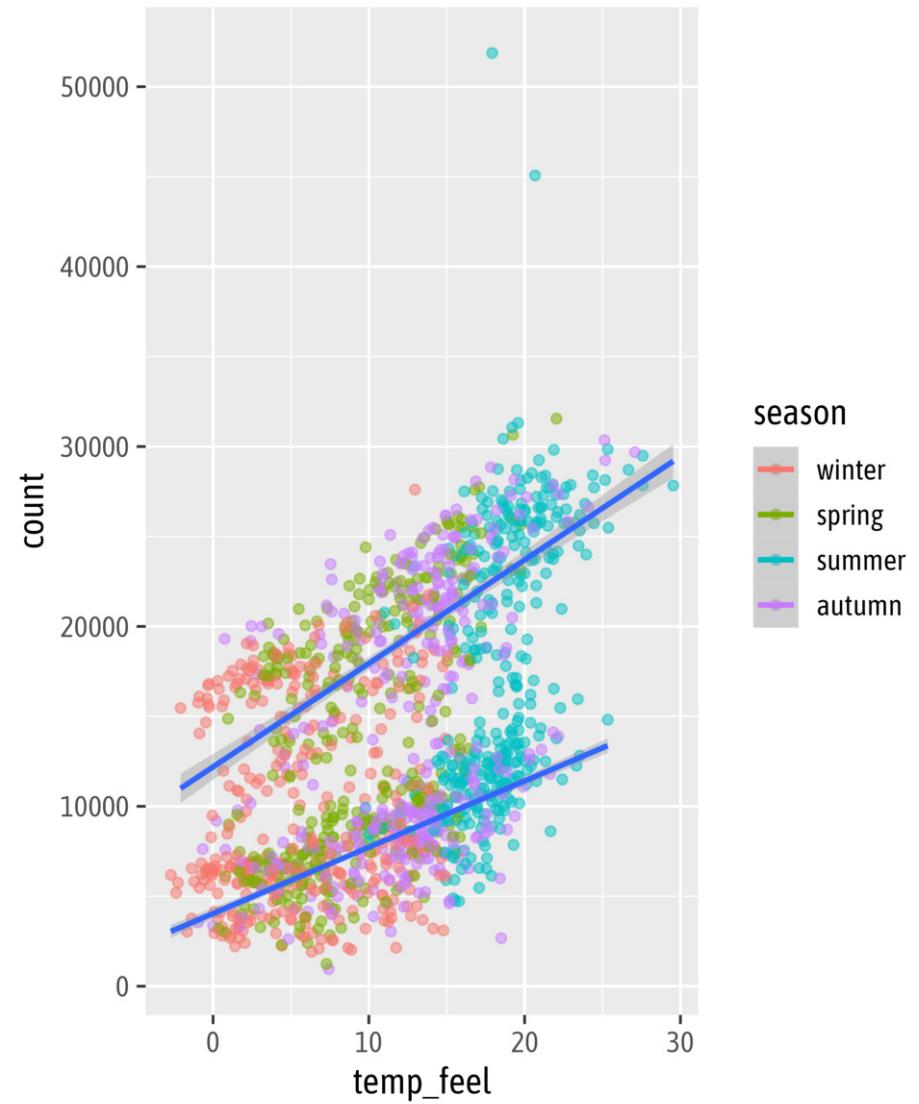
The `group` Aesthetic

```
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 )
```



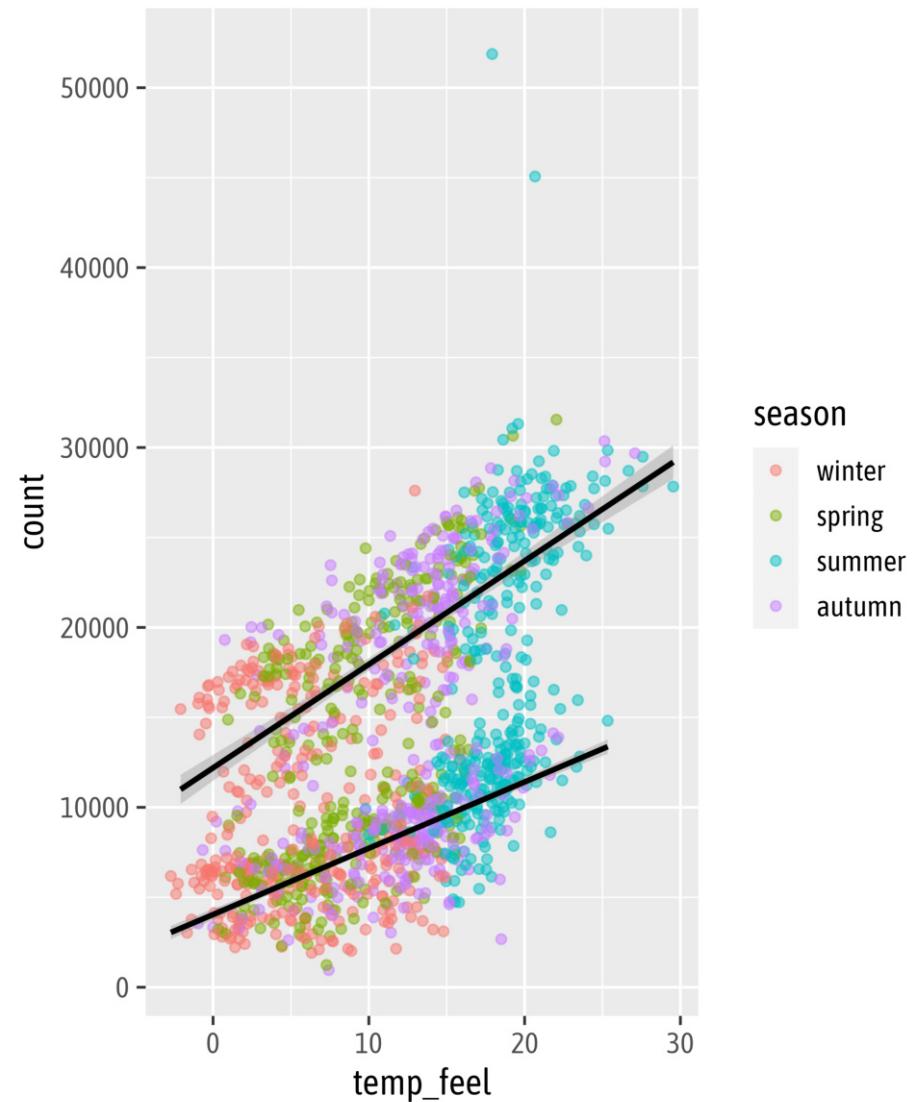
Set Both as Global Aesthetics

```
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 )
```



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  )
```

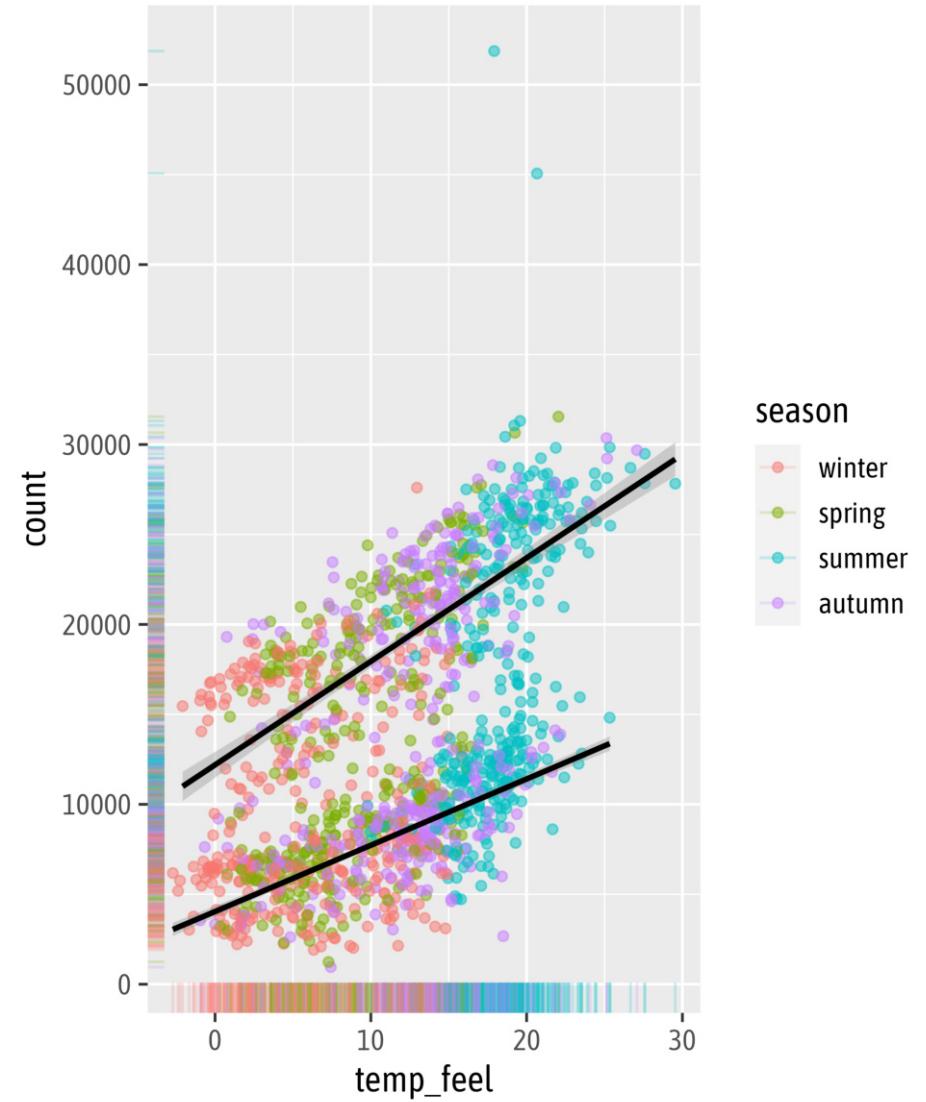
```
1 class(g)
```

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



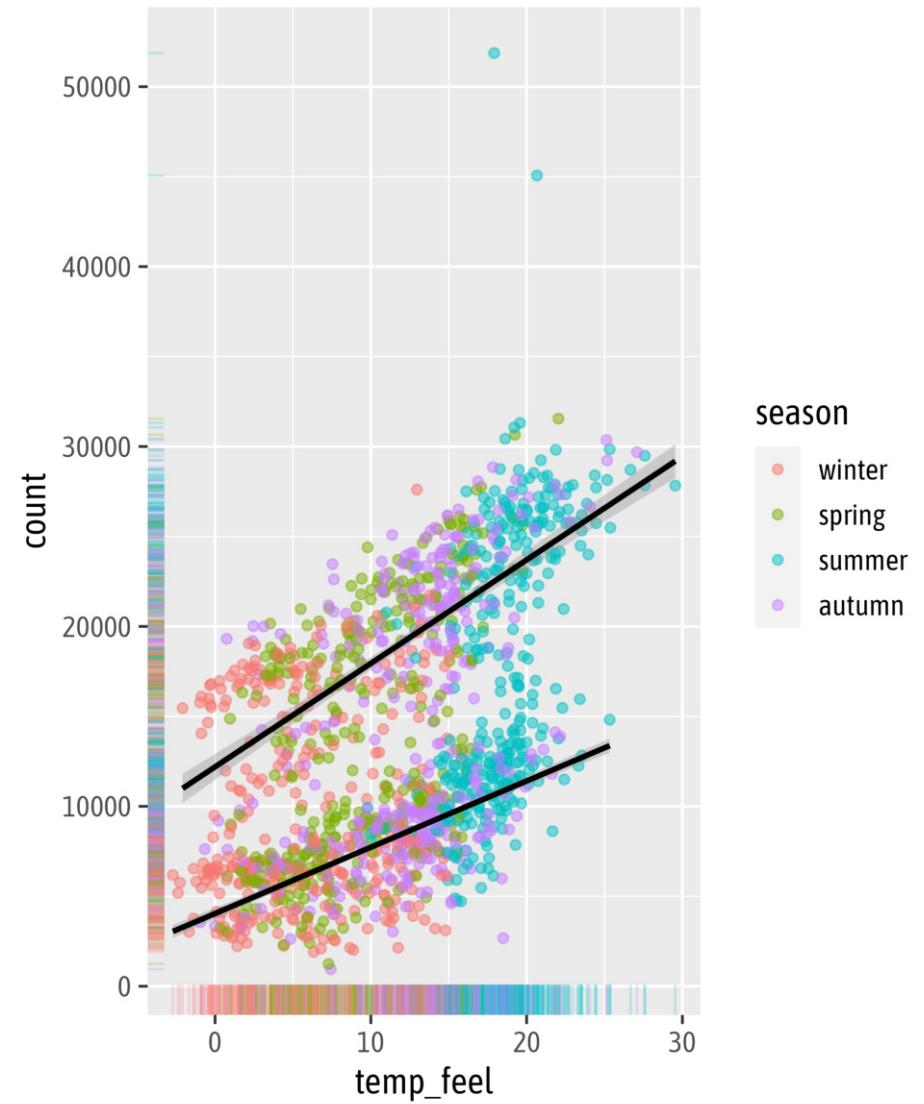
Add More Layers

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



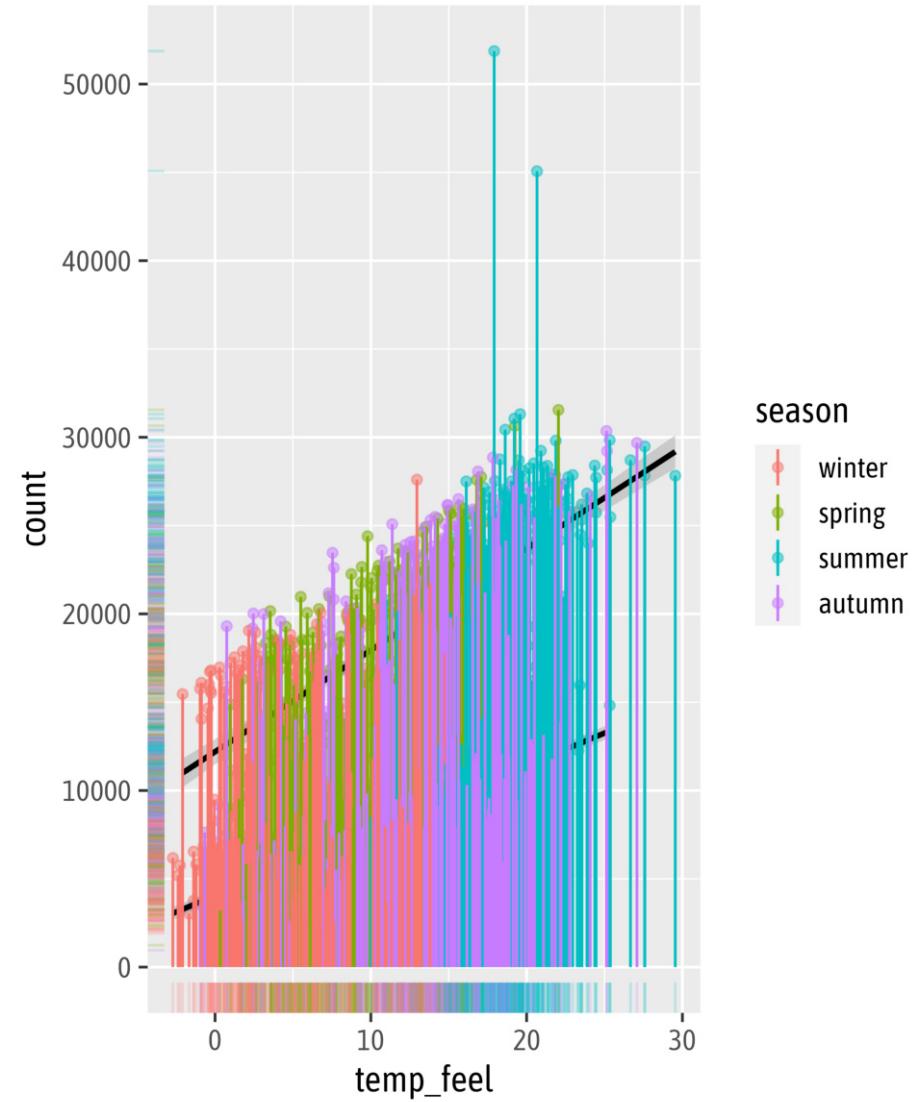
Remove a Layer from the Legend

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

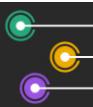


Add More Layers

```
1 g +  
2   geom_rug(  
3     alpha = .2,  
4     show.legend = FALSE  
5   ) +  
6   geom_linerange(  
7     aes(ymin = 0, ymax = count))  
8 )
```

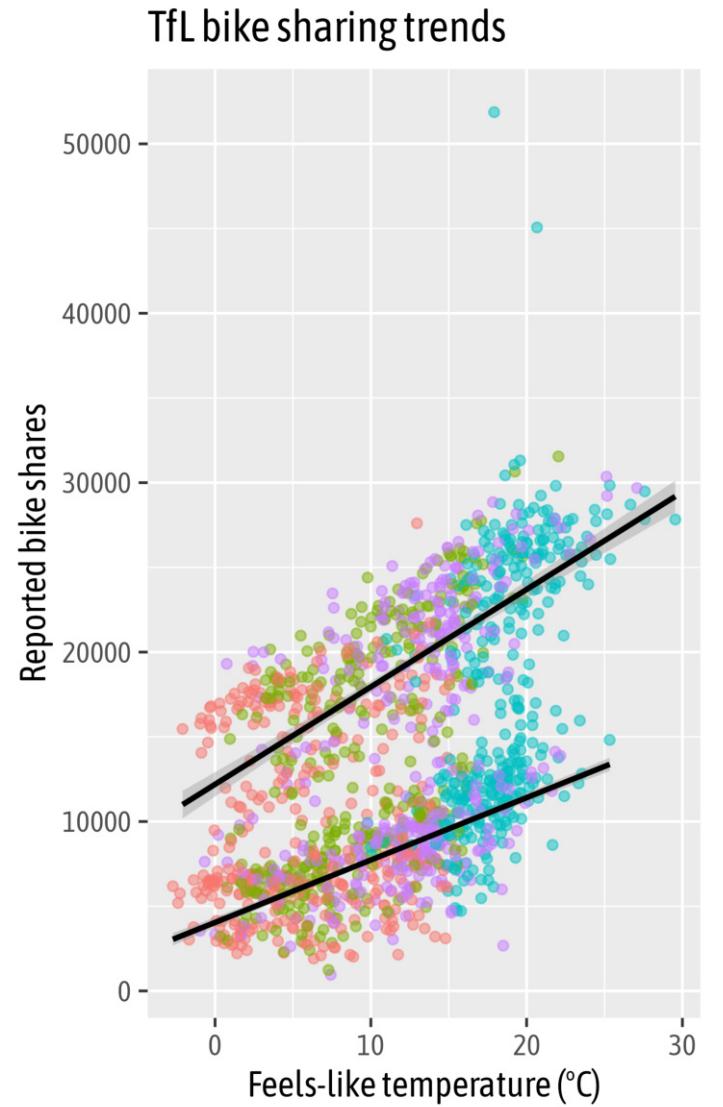


A Polished *ggplot* Example



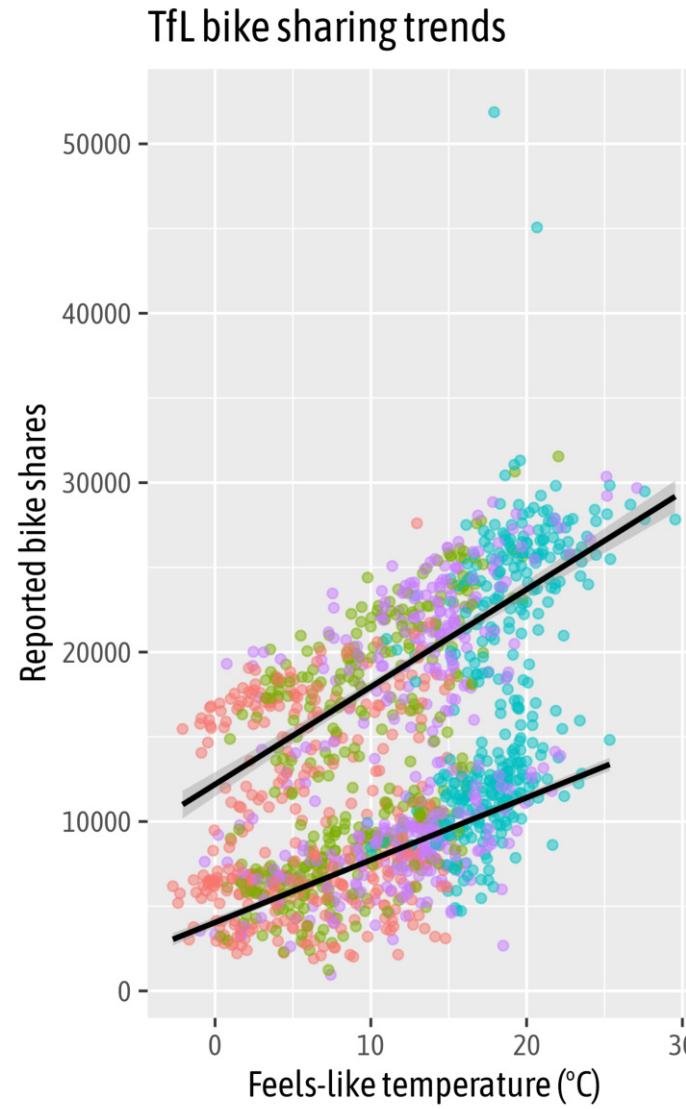
Add Labels

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



Add Labels

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



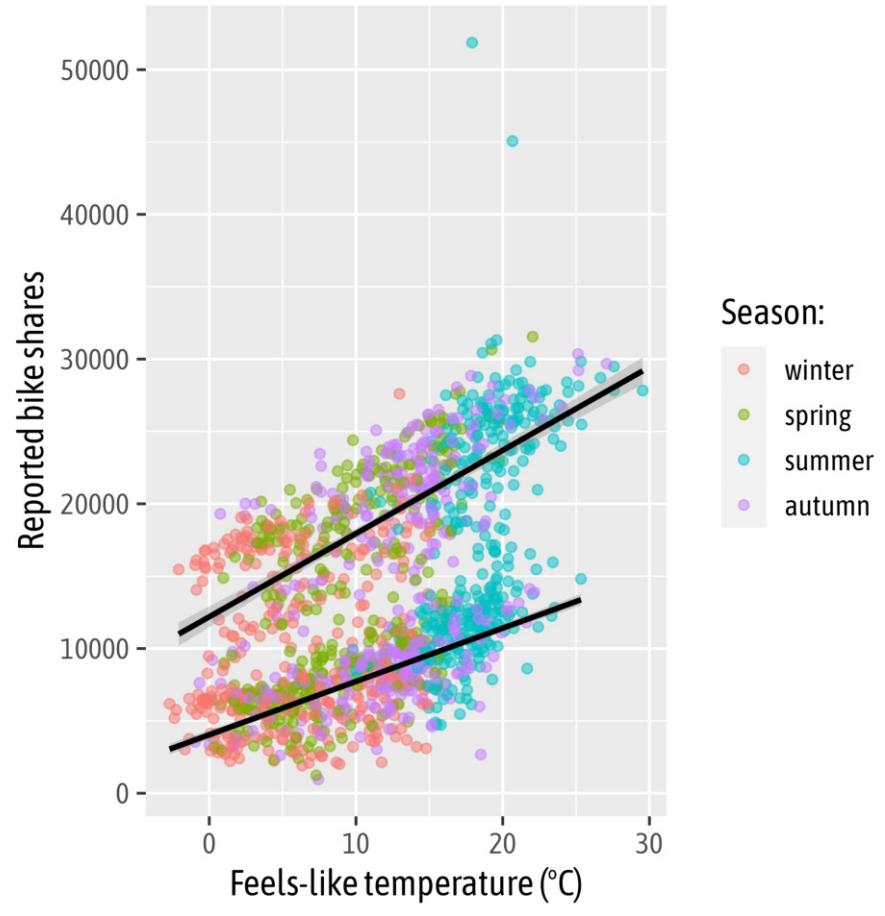
Add Labels

```
1 g +  
2   labs(  
3     subtitle = "Reported bike rents versus fe  
4     caption = "Data: TfL",  
5     tag = "A)",  
6     color = "Season:"  
7   )
```

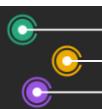
A)

TfL bike sharing trends

Reported bike rents versus feels-like temperature in London

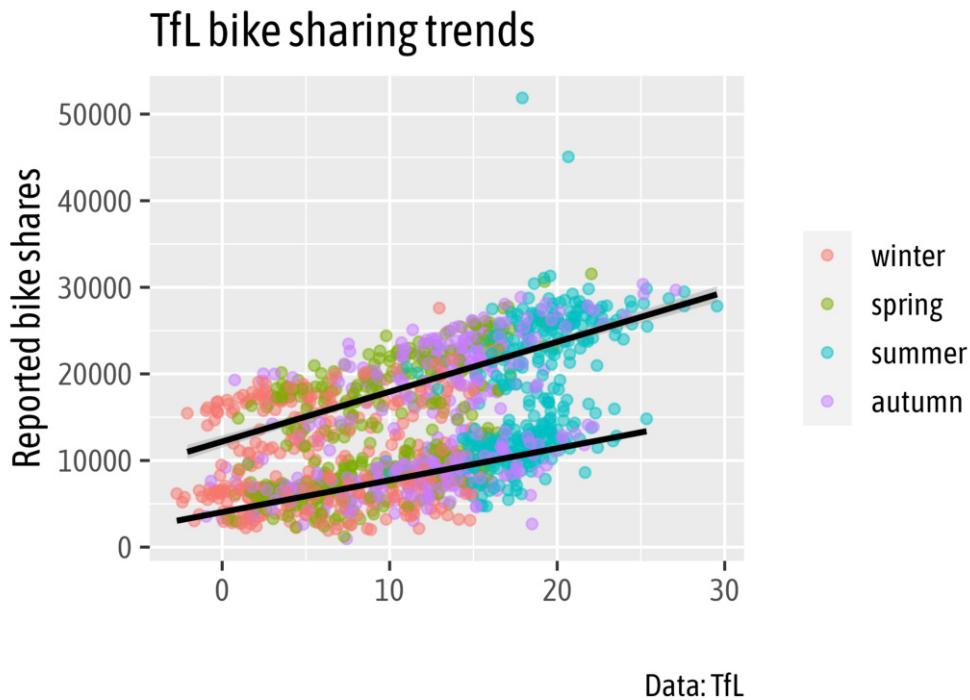


Data: TfL

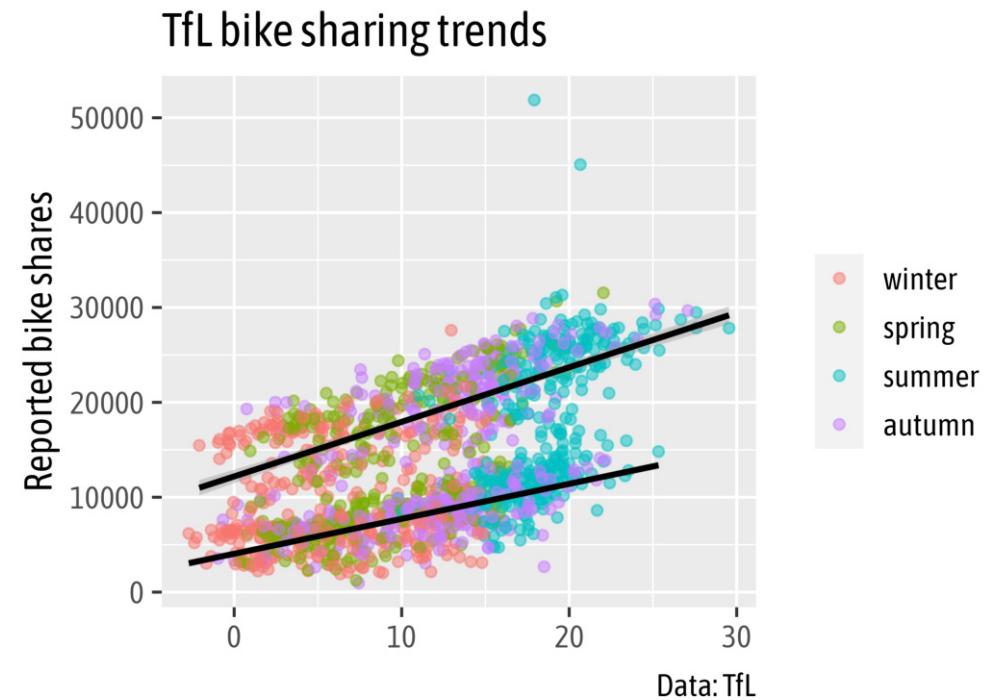


Add Labels

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

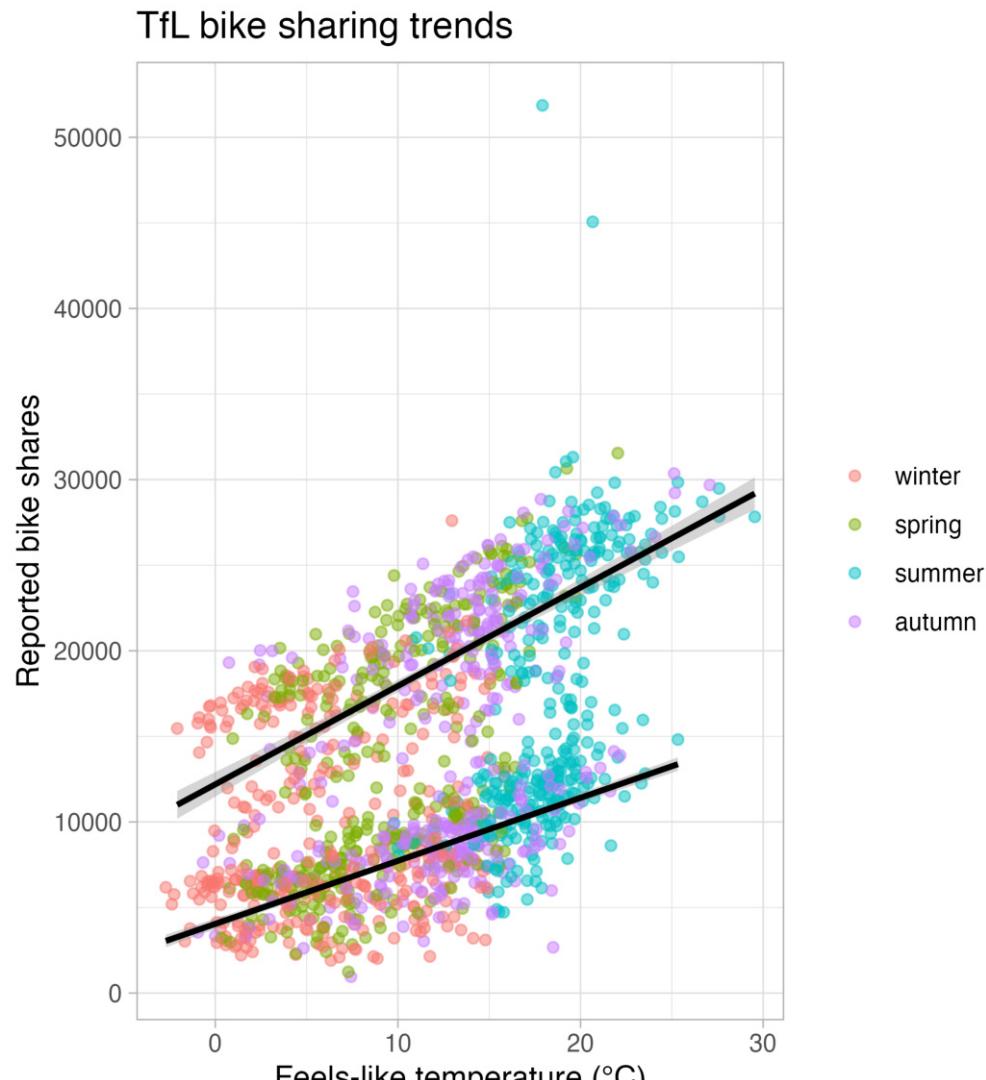


```
1 g +  
2   labs(  
3     x = NULL,  
4     caption = "Data: TfL"  
5   )
```

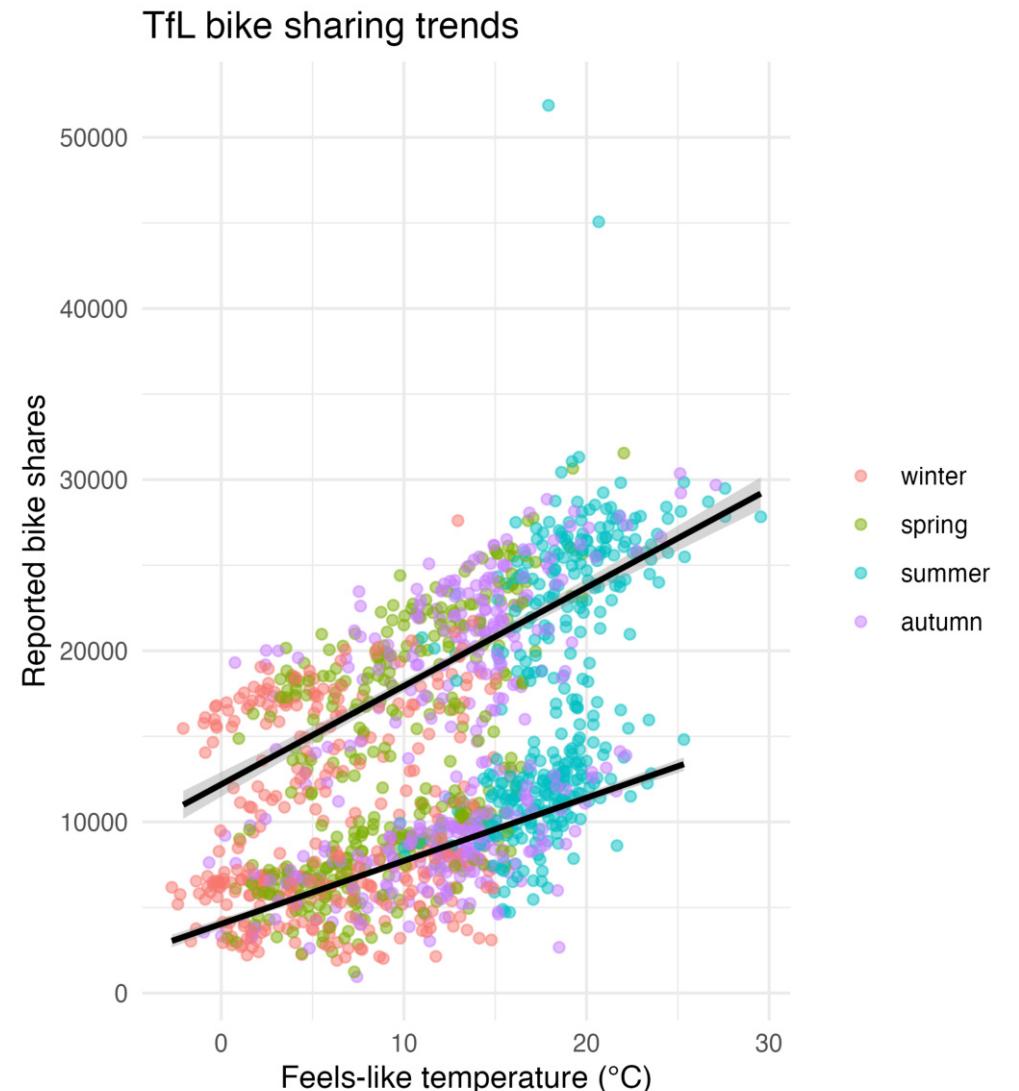


Themes

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



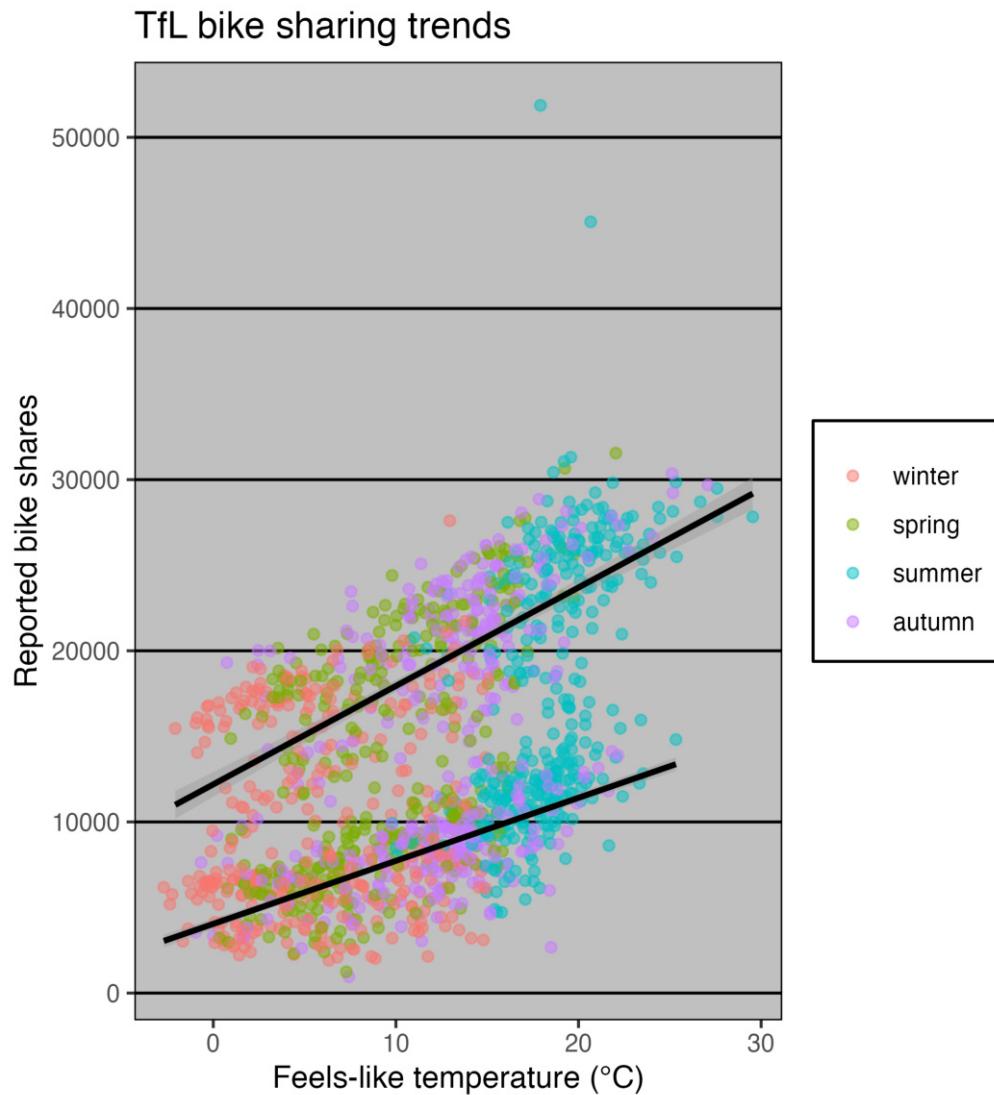
```
1 g + theme_minimal()
```



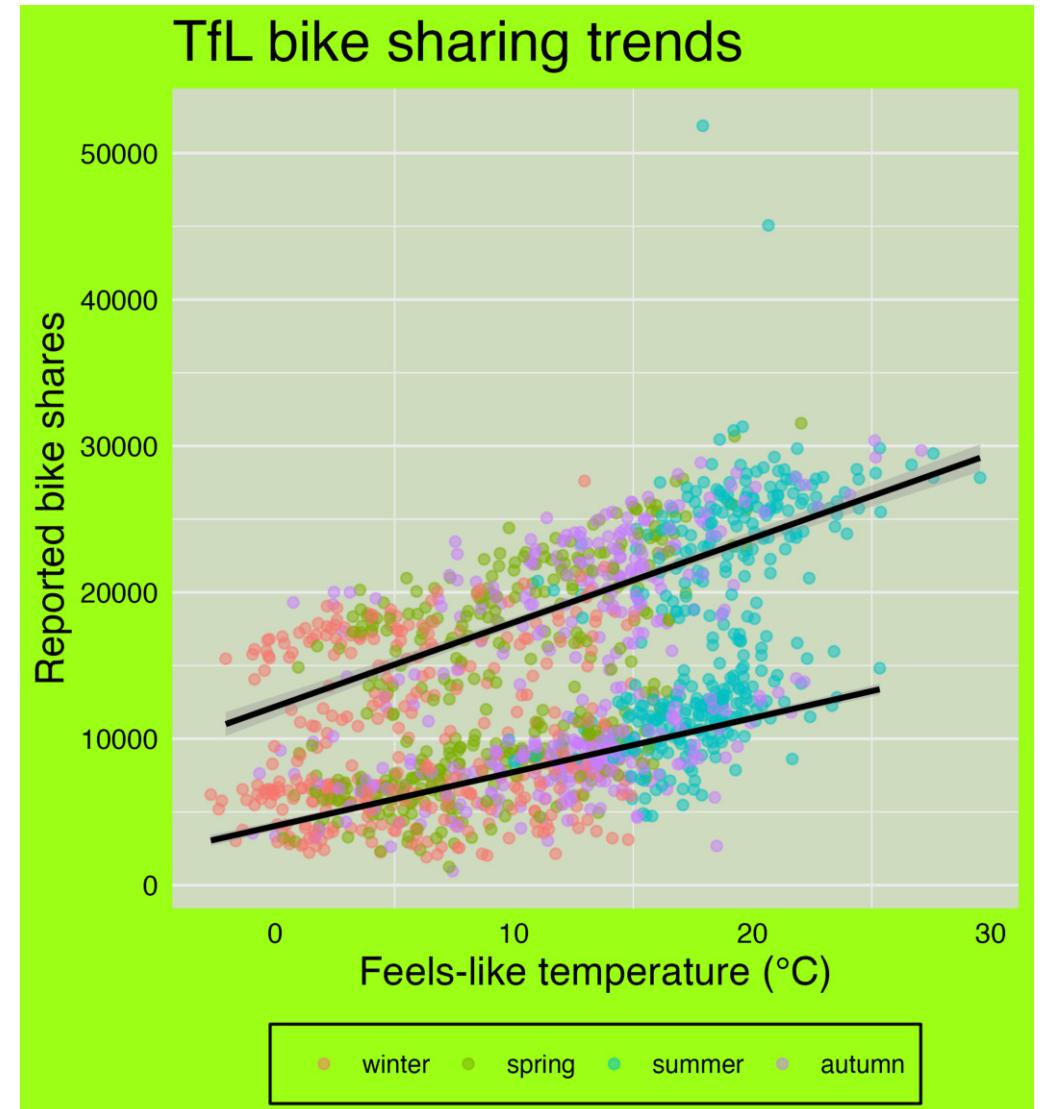


Themes

```
1 g + ggthemes::theme_excel()
```



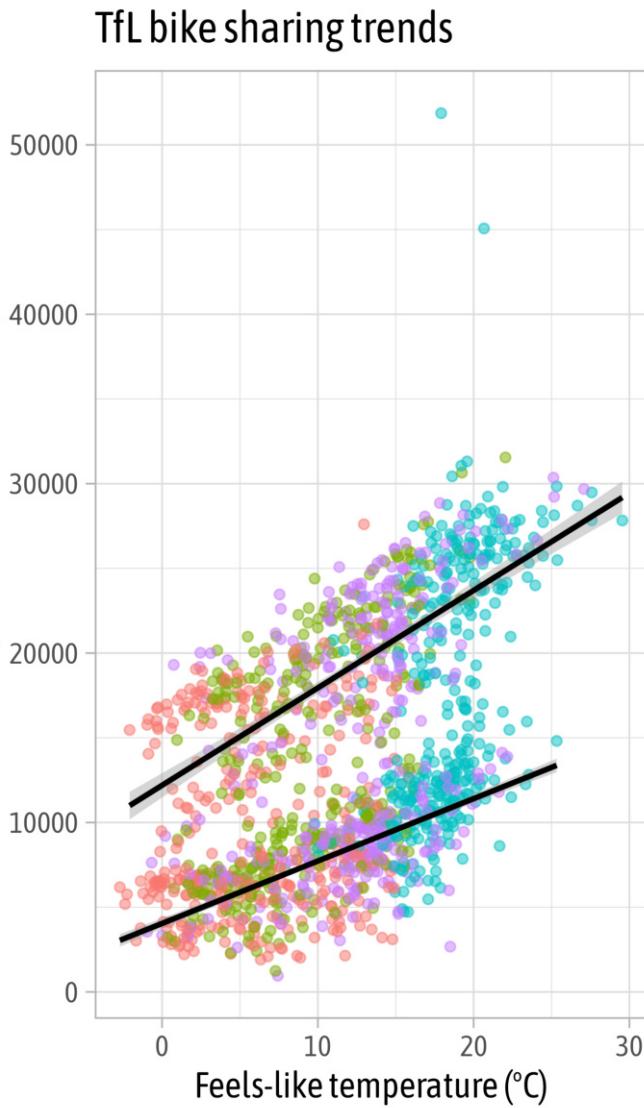
```
1 g + tvthemes::theme_rickAndMorty()
```





Change the Theme Base Settings

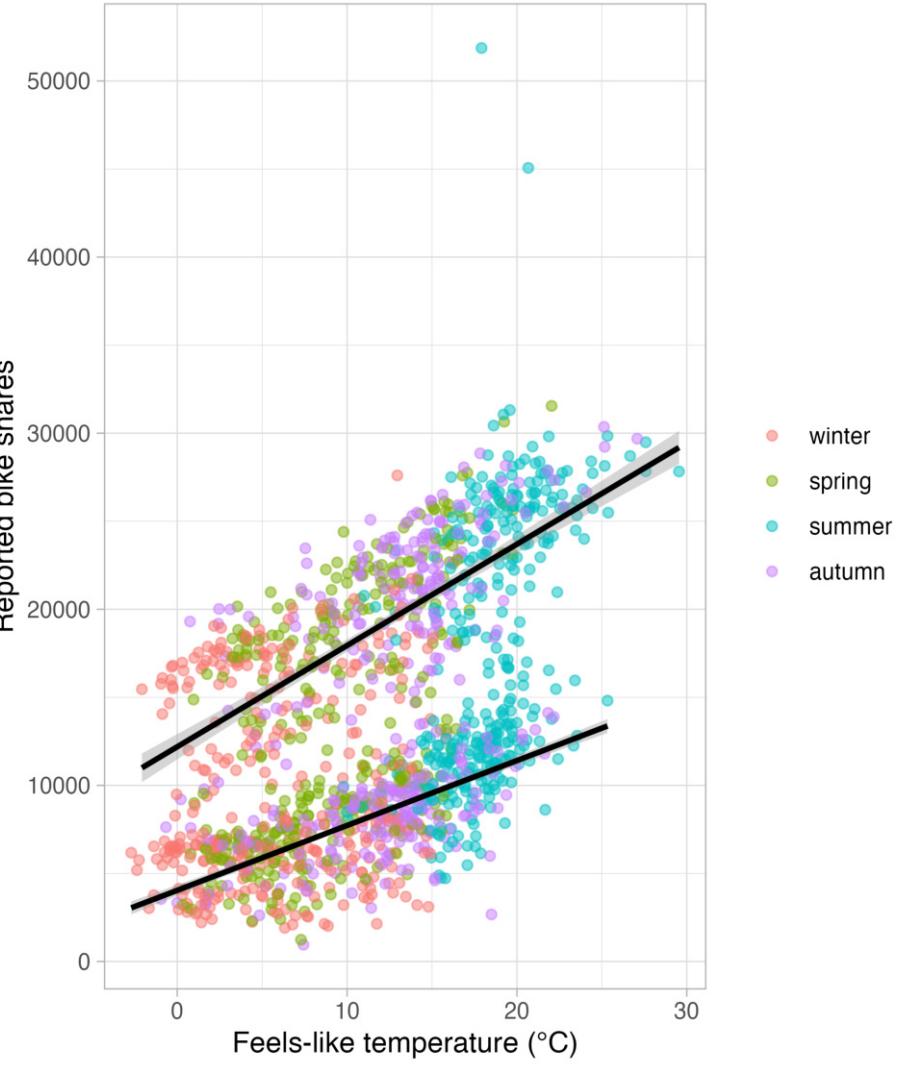
```
1 g + theme_light(  
2   base_size = 14,  
3   base_family = "Asap Condensed"  
4 )
```



Set a Theme Globally

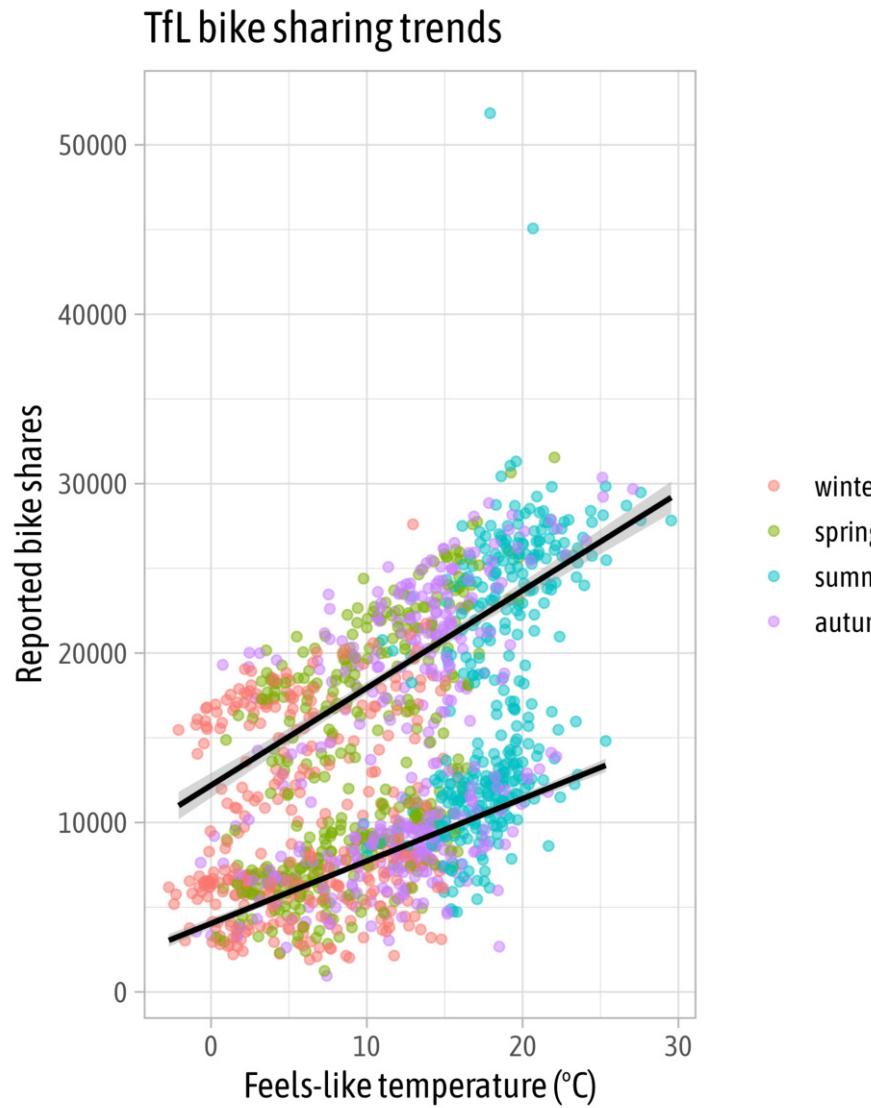
```
1 theme_set(theme_light())  
2  
3 g
```

TfL bike sharing trends



Change the Theme Base Settings

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



{systemfonts}

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

[1] "Asap" "Asap Condensed" "Asap Expanded" "Asap SemiCondensed" "Asap SemiExpanded"



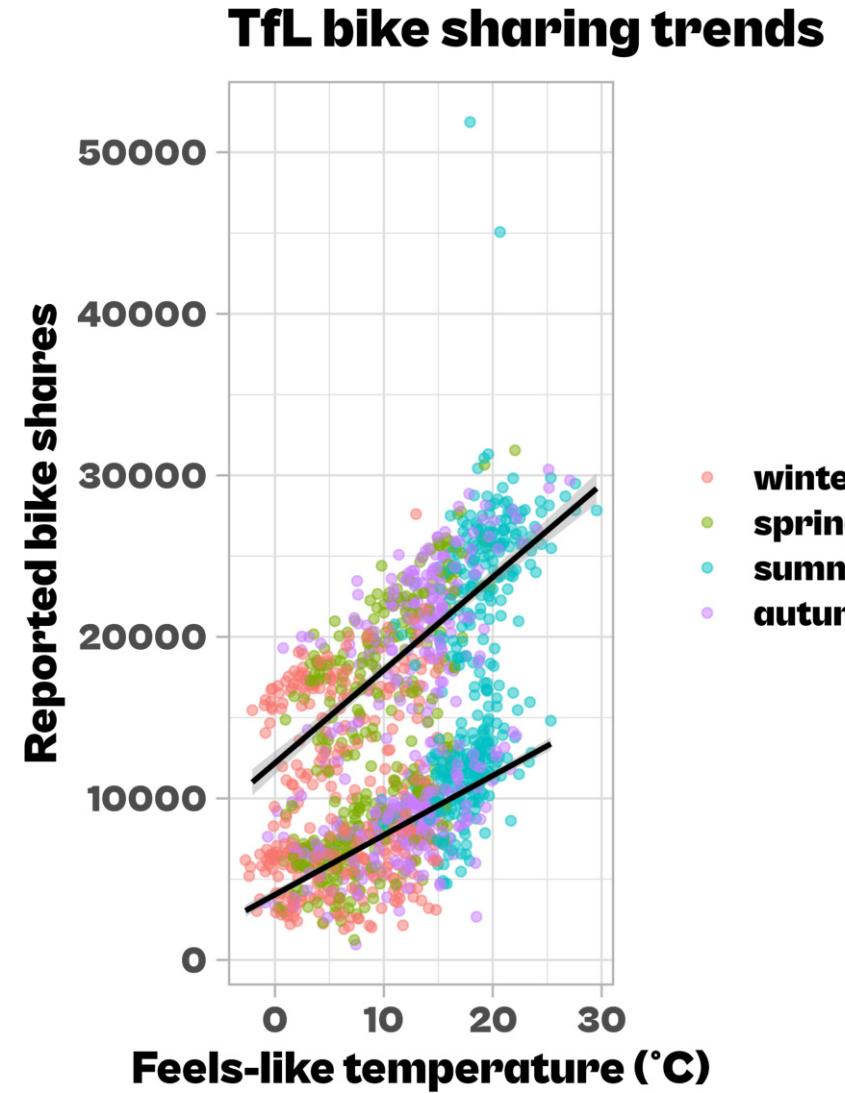
{systemfonts}

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



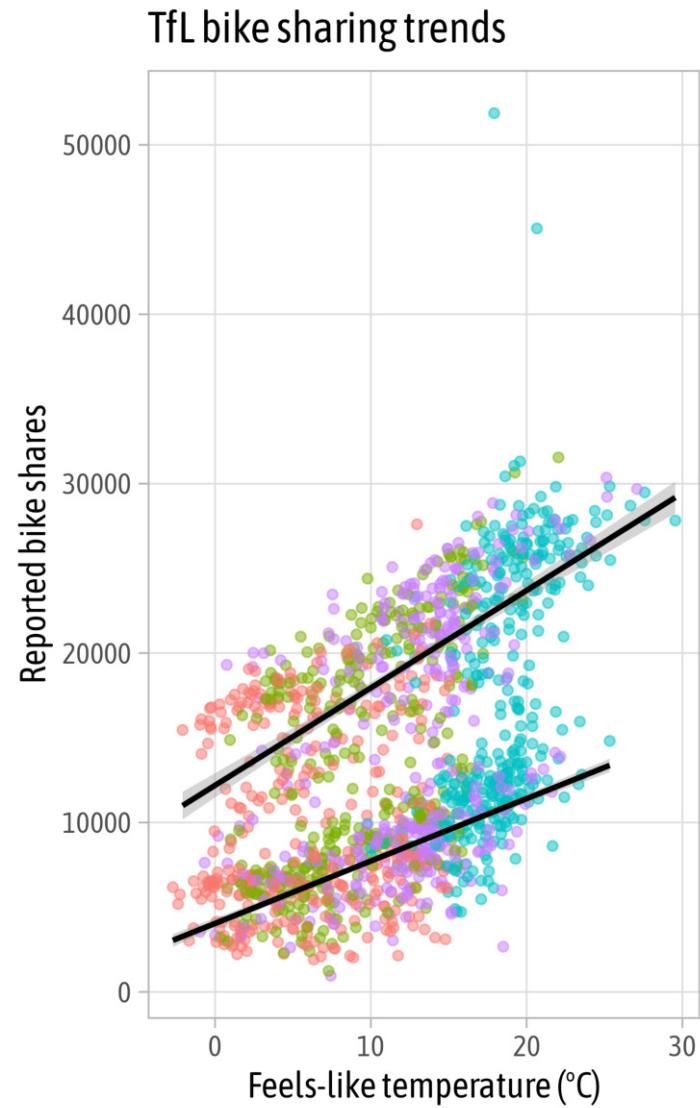
{systemfonts} + {ggplot2}

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



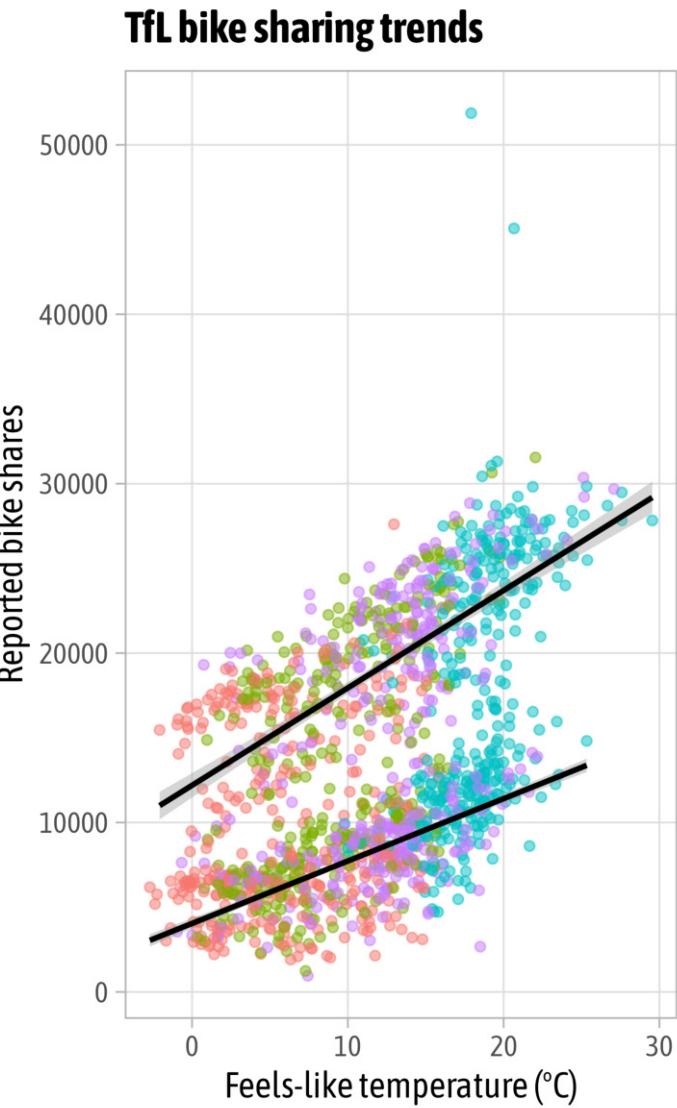
Overwrite Specific Theme Settings

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



Overwrite Specific Theme Settings

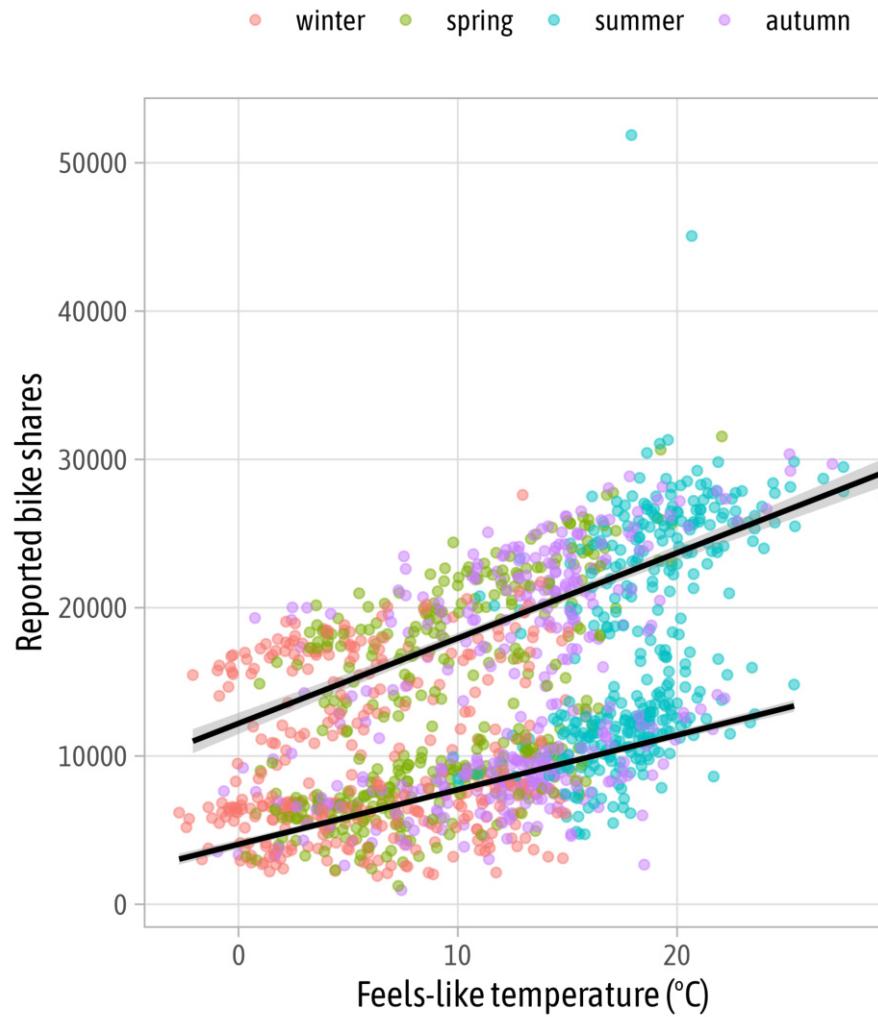
```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold")  
5   )
```



Overwrite Specific Theme Settings

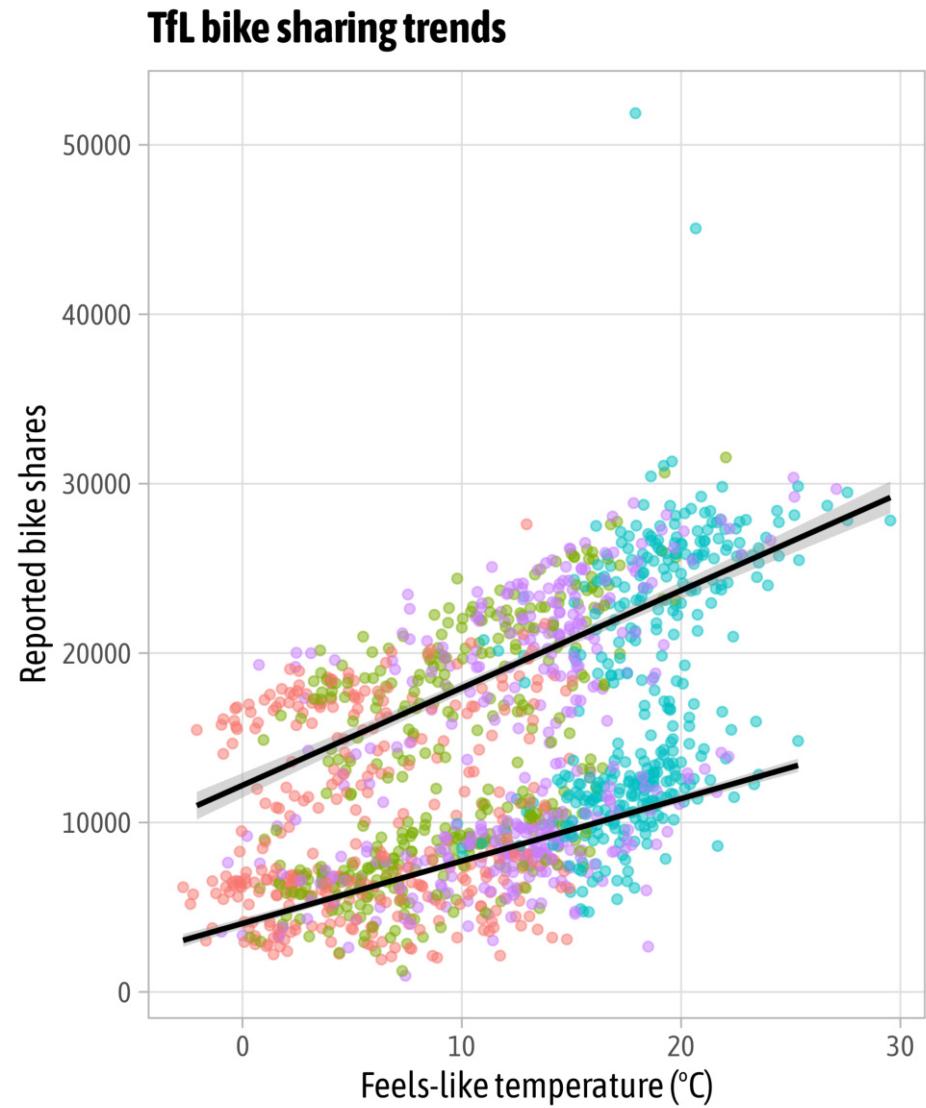
```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold"),  
5     legend.position = "top"  
6   )
```

TfL bike sharing trends



Overwrite Specific Theme Settings

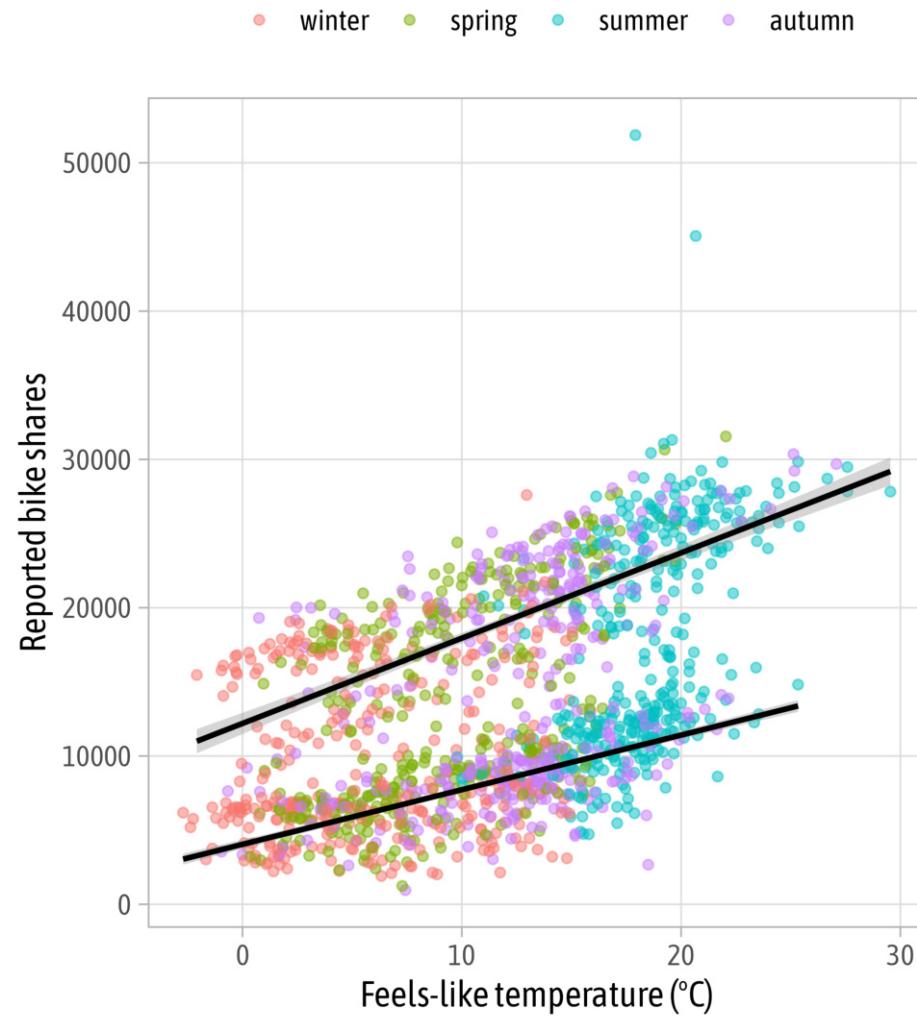
```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold"),  
5     legend.position = "none"  
6   )
```



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   )
```

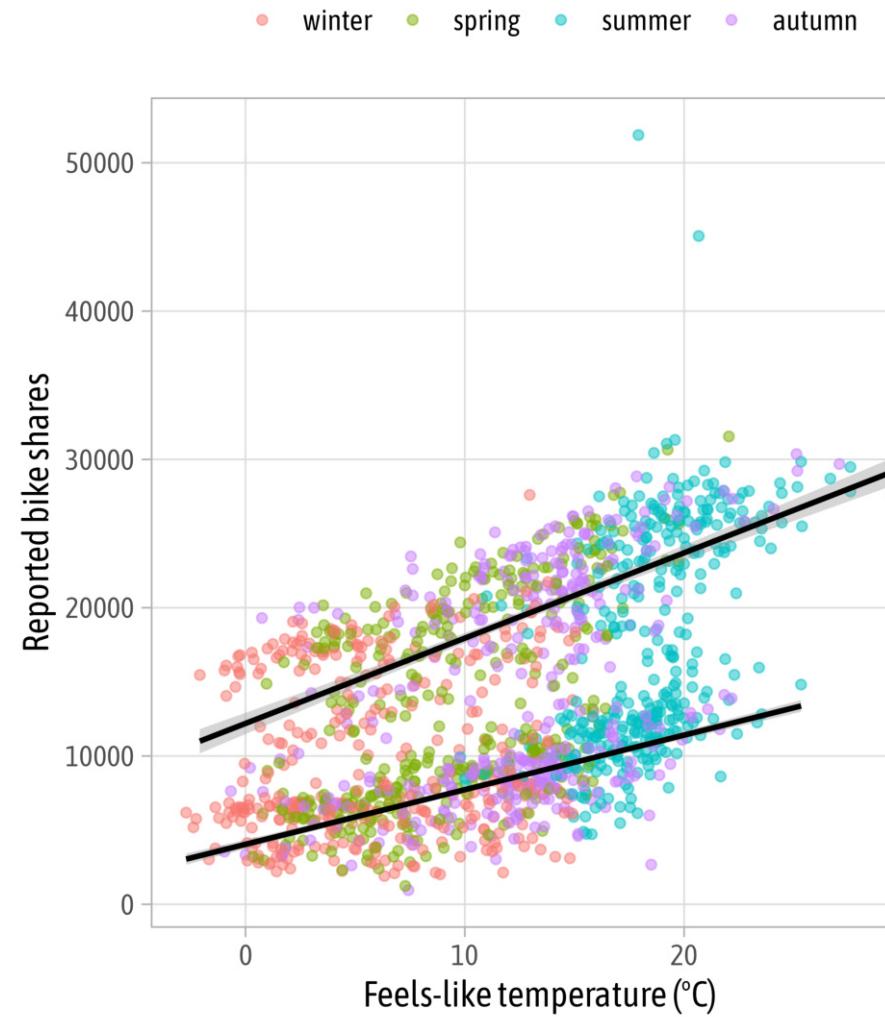
TfL bike sharing trends



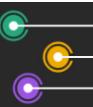
Overwrite Theme Settings Globally

```
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



Export Your Graphic



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)
```





Vector graphic

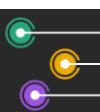
e.g. svg, pdf, eps



Raster graphic

e.g. png, jpeg, bmp, tiff

Modified from canva.com



Scales



Scales

= translate between variable and property ranges

- feels-like temperature \rightleftharpoons x
- reported bike shares \rightleftharpoons y
- season \rightleftharpoons color
- year \rightleftharpoons shape
- ...



Scales

The `scale_*`() components control the properties of all the
aesthetic dimensions mapped to the data.

Consequently, there are `scale_*`() functions for all aesthetics such as:

- **positions** via `scale_x_*`() and `scale_y_*`()
- **colors** via `scale_color_*`() and `scale_fill_*`()
- **sizes** via `scale_size_*`() and `scale_radius_*`()
- **shapes** via `scale_shape_*`() and `scale_linetype_*`()
- **transparency** via `scale_alpha_*`()

... with tons of options for `*` such as `continuous`, `discrete`, `manual`, `log10`, `gradient`, and many more



Scales

The `scale_*`() components control the properties of all the **aesthetic dimensions mapped to the data.**

Use scales to:

- set breaks (tick marks)
- overwrite axis / legend labels
- adjust axis / legend title
- modify colors and palettes
- define shapes and line types
- scale sizes (e.g. bubbles)



CONTINUOUS

measured data, can have ∞ values within possible range.



I AM 3.1" TALL

I WEIGH 34.16 grams

DISCRETE

OBSERVATIONS CAN ONLY EXIST AT LIMITED VALUES, OFTEN COUNTS.

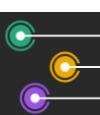


I HAVE 8 LEGS
and
4 SPOTS!

@allison_horst

Illustration by Allison Horst

Cédric Scherer // Data Visualization Society // March 9, 2023



Continuous vs. Discrete in {ggplot2}

Continuous:

quantitative or numerical data

- height
- weight
- age
- counts

Discrete:

qualitative or categorical data

- species
- sex
- study sites
- age group



Continuous vs. Discrete in {ggplot2}

Continuous:
quantitative or numerical data

- height (continuous)
- weight (continuous)
- age (continuous or discrete)
- counts (discrete)

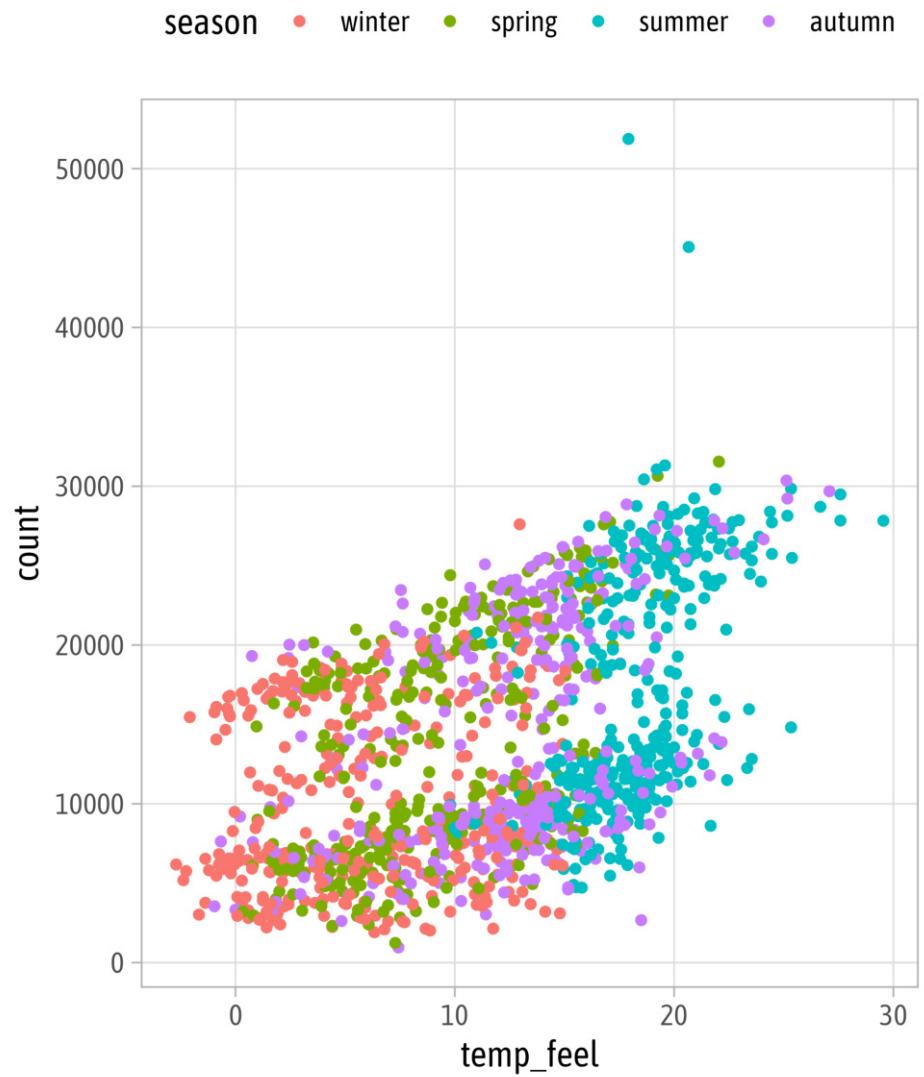
Discrete:
qualitative or categorical data

- species (nominal)
- sex (nominal)
- study site (nominal or ordinal)
- age group (ordinal)



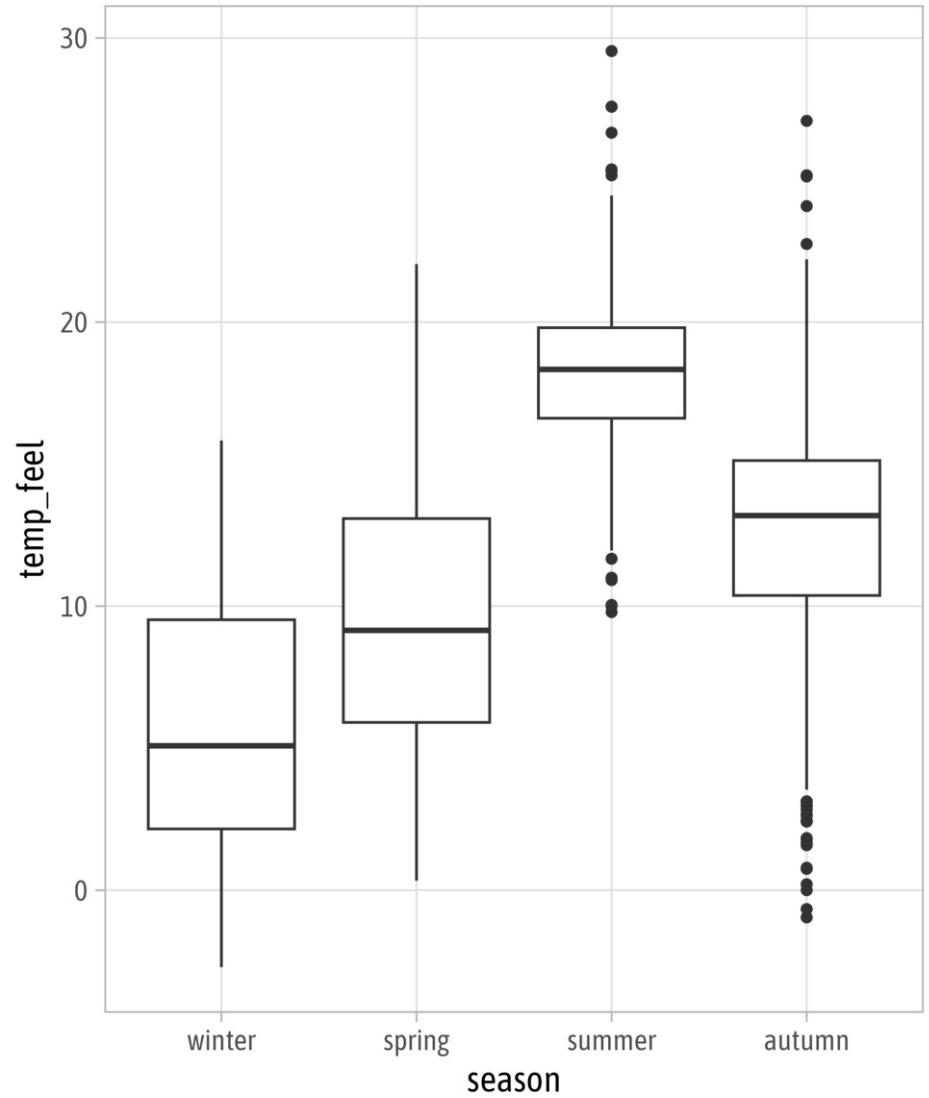
Aesthetics + Scales

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



Aesthetics + Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = temp_feel)  
4 ) +  
5 geom_boxplot() +  
6 scale_x_discrete() +  
7 scale_y_continuous()
```

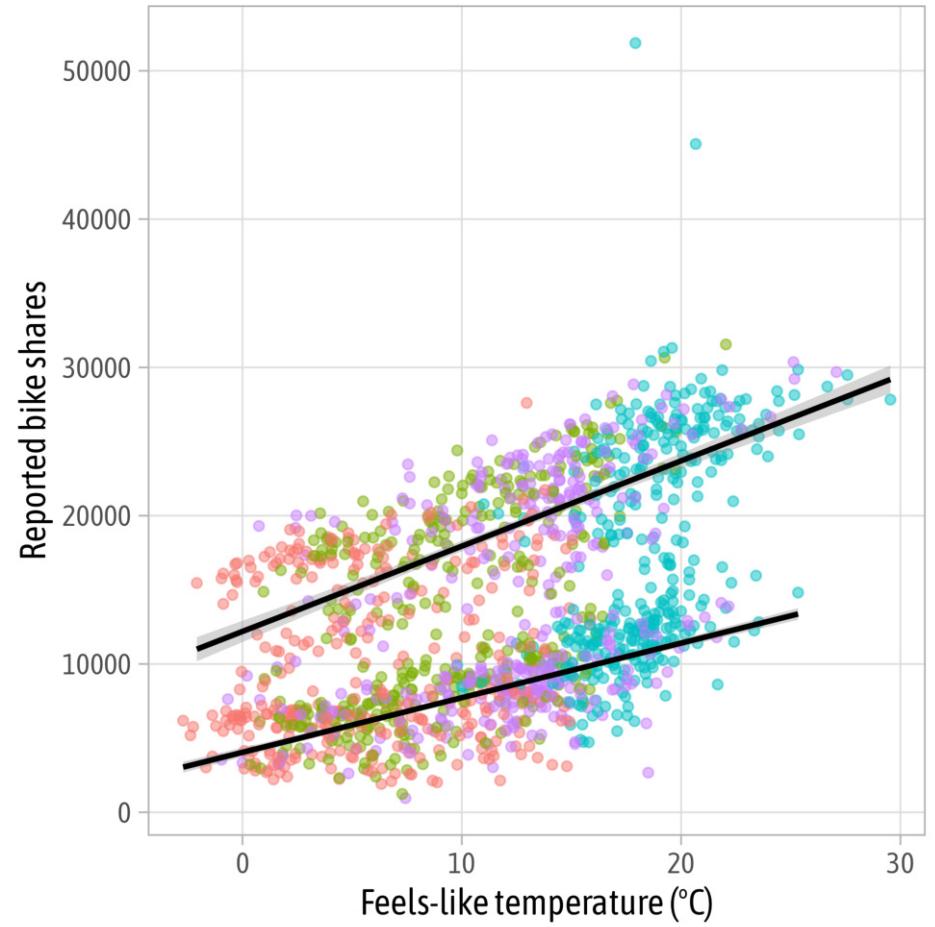


Aesthetics + Scales

```
1 g +  
2   scale_x_continuous() +  
3   scale_y_continuous() +  
4   scale_color_discrete()
```

TfL bike sharing trends

● winter ● spring ● summer ● autumn

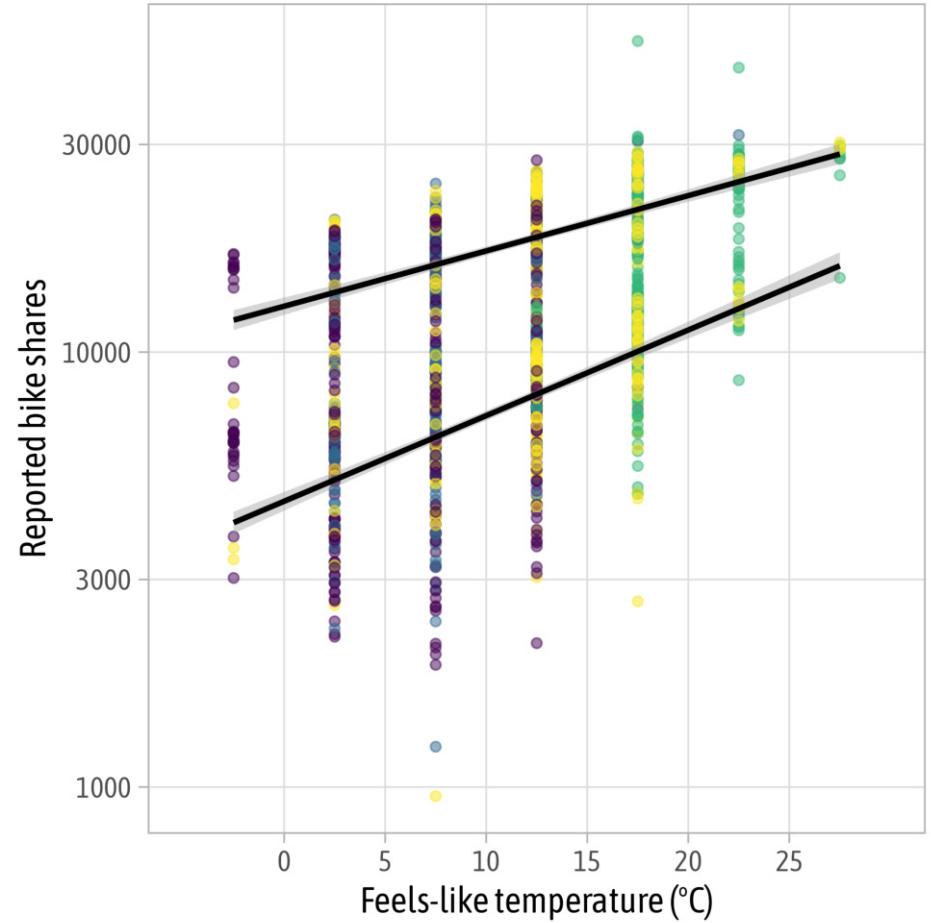


Overwrite Scales

```
1 g +  
2   scale_x_binned() +  
3   scale_y_log10() +  
4   scale_color_viridis_d()
```

TfL bike sharing trends

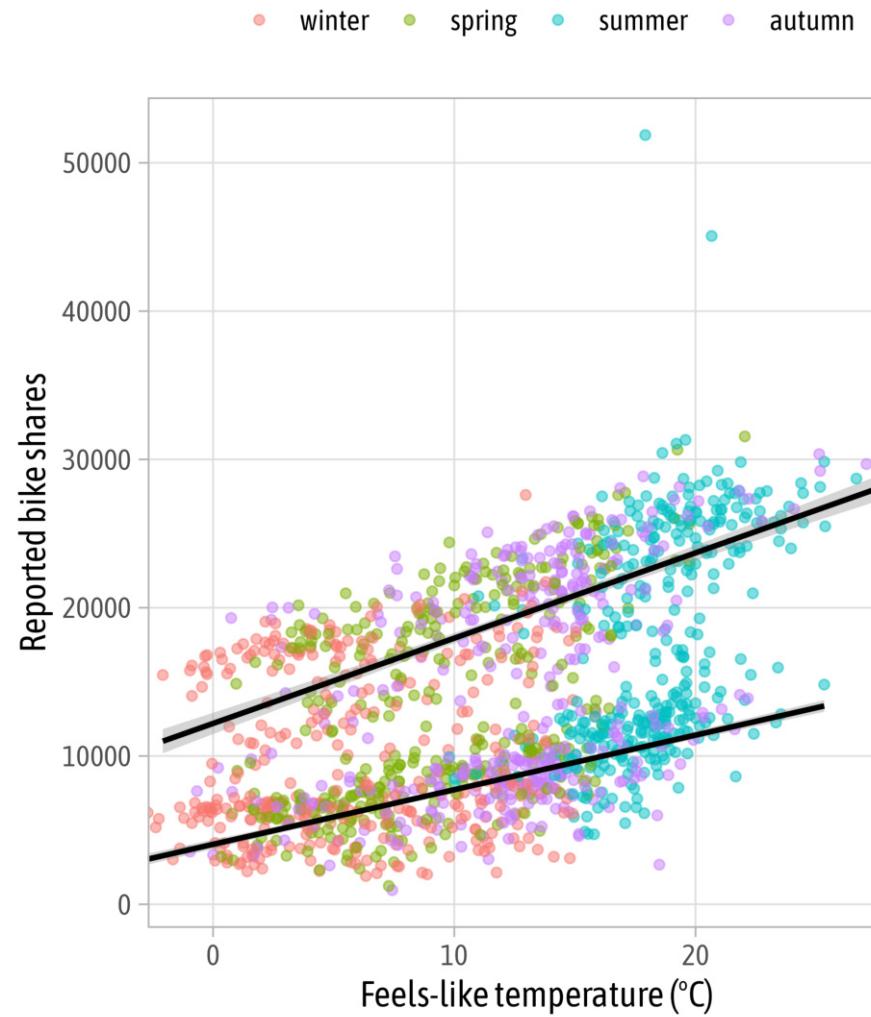
● winter ● spring ● summer ● autumn



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0)
4   ) +
5   scale_y_continuous() +
6   scale_color_discrete()
```

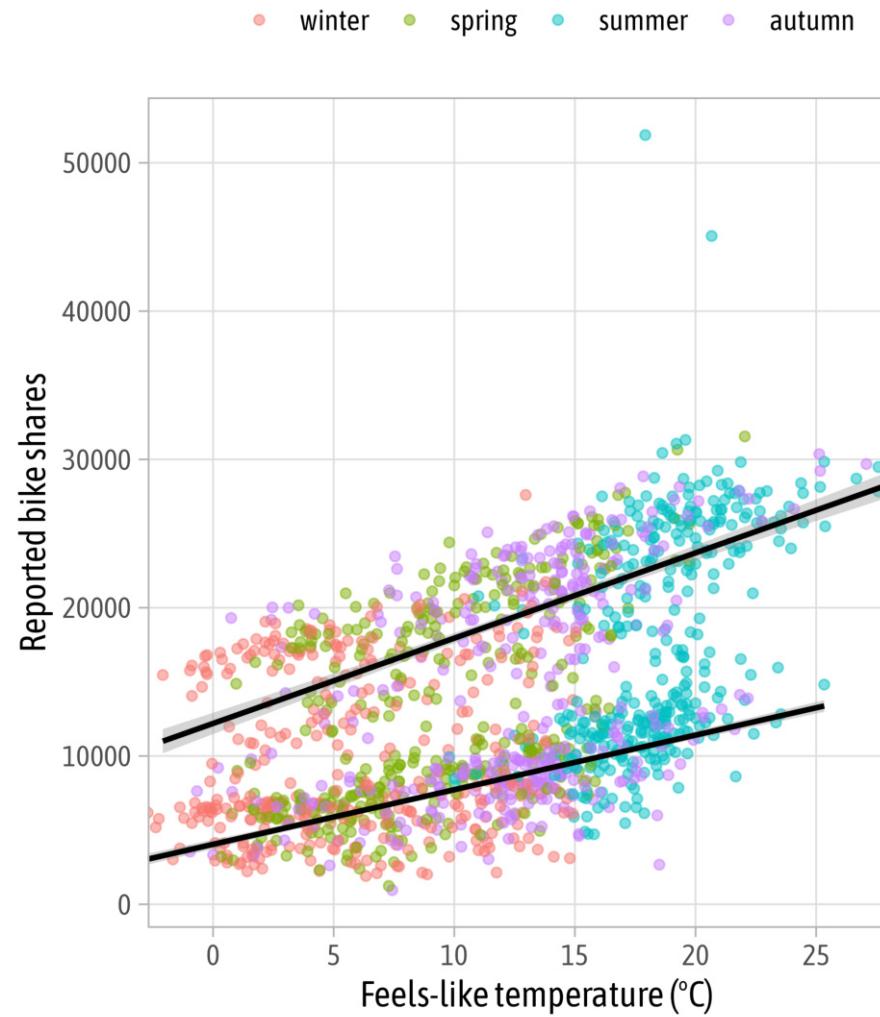
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5)
5   ) +
6   scale_y_continuous() +
7   scale_color_discrete()
```

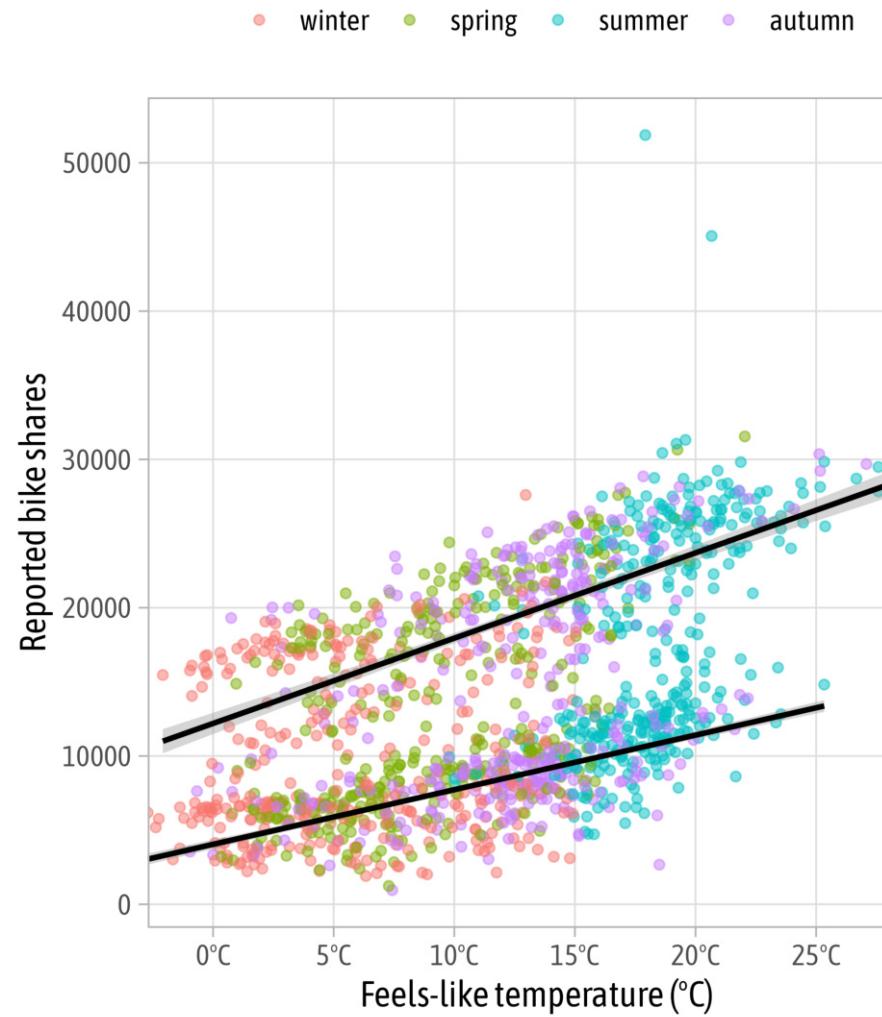
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"))
6 ) +
7   scale_y_continuous() +
8   scale_color_discrete()
```

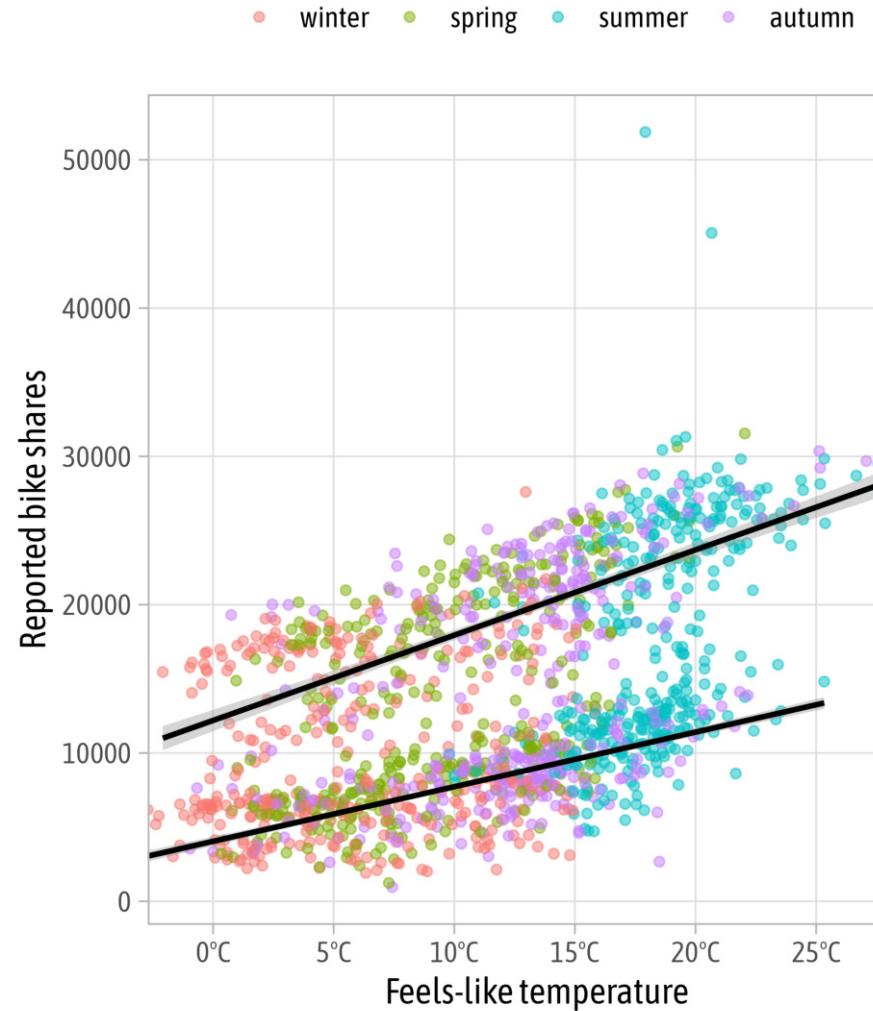
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous() +
9   scale_color_discrete()
```

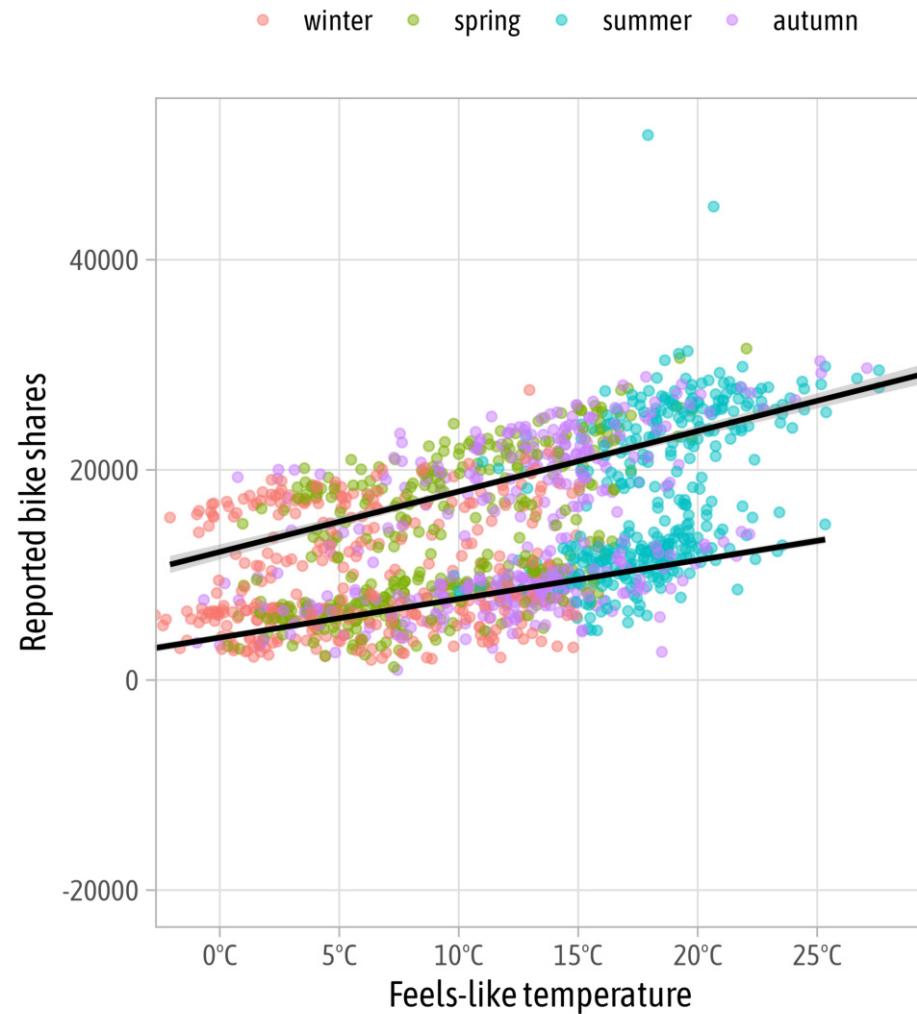
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     limits = c(-20000, NA)
10  ) +
11  scale_color_discrete()
```

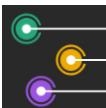
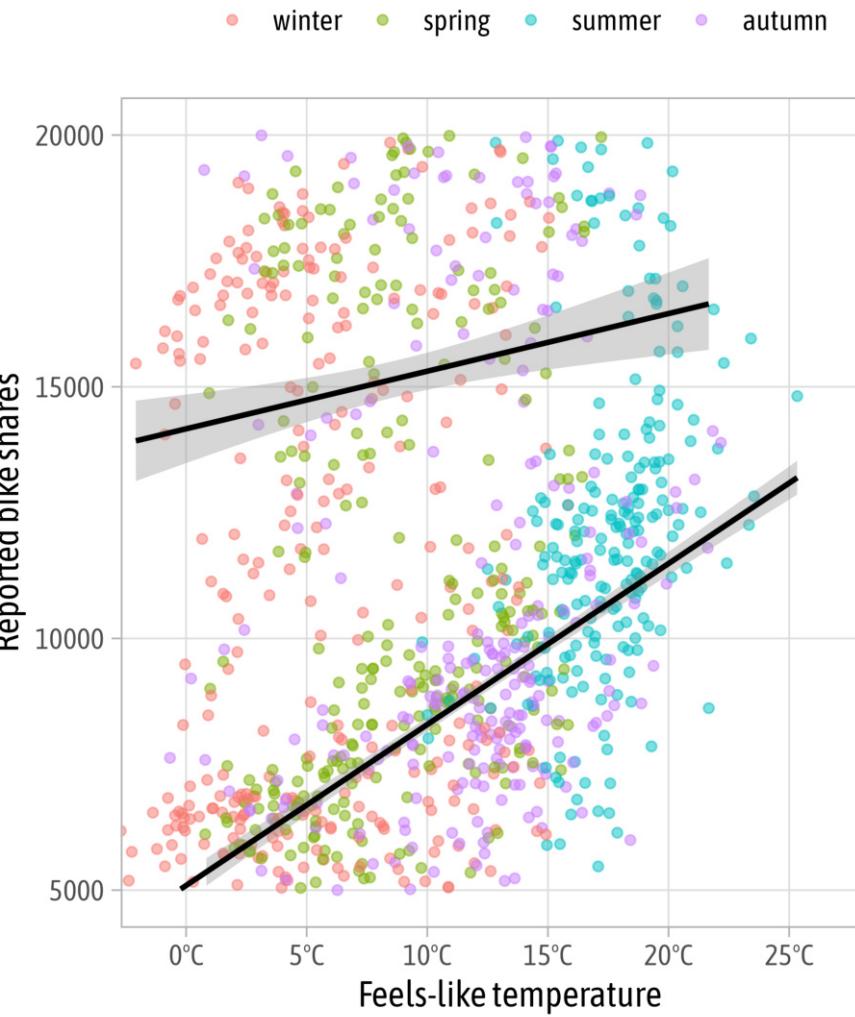
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     limits = c(5000, 20000)
10  ) +
11  scale_color_discrete()
```

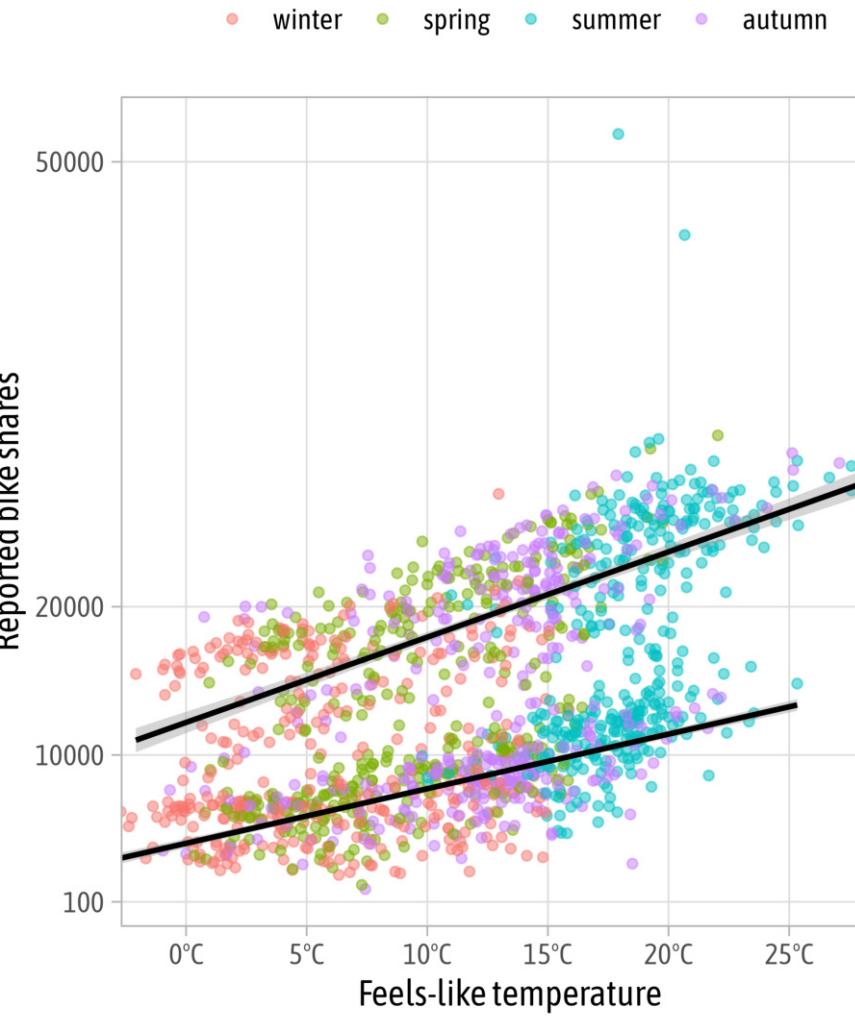
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = c(100, 10000, 20000, 50000)
10  ) +
11  scale_color_discrete()
```

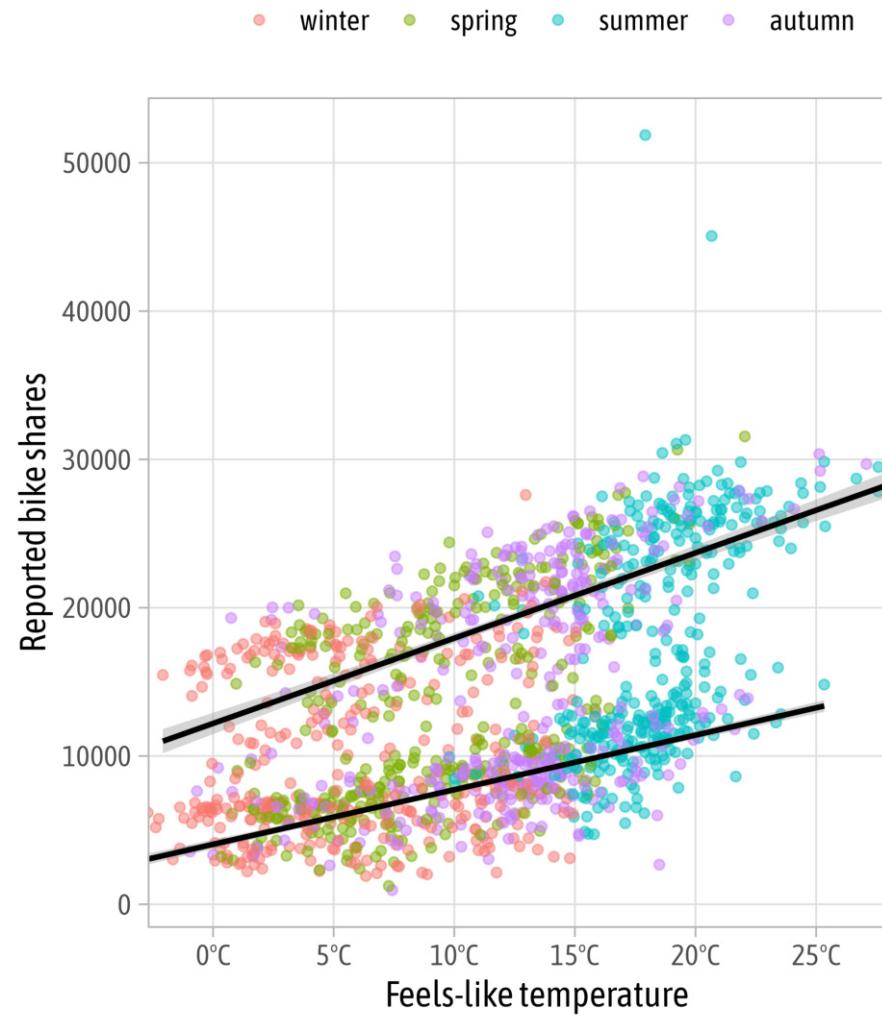
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000
10  ) +
11  scale_color_discrete()
```

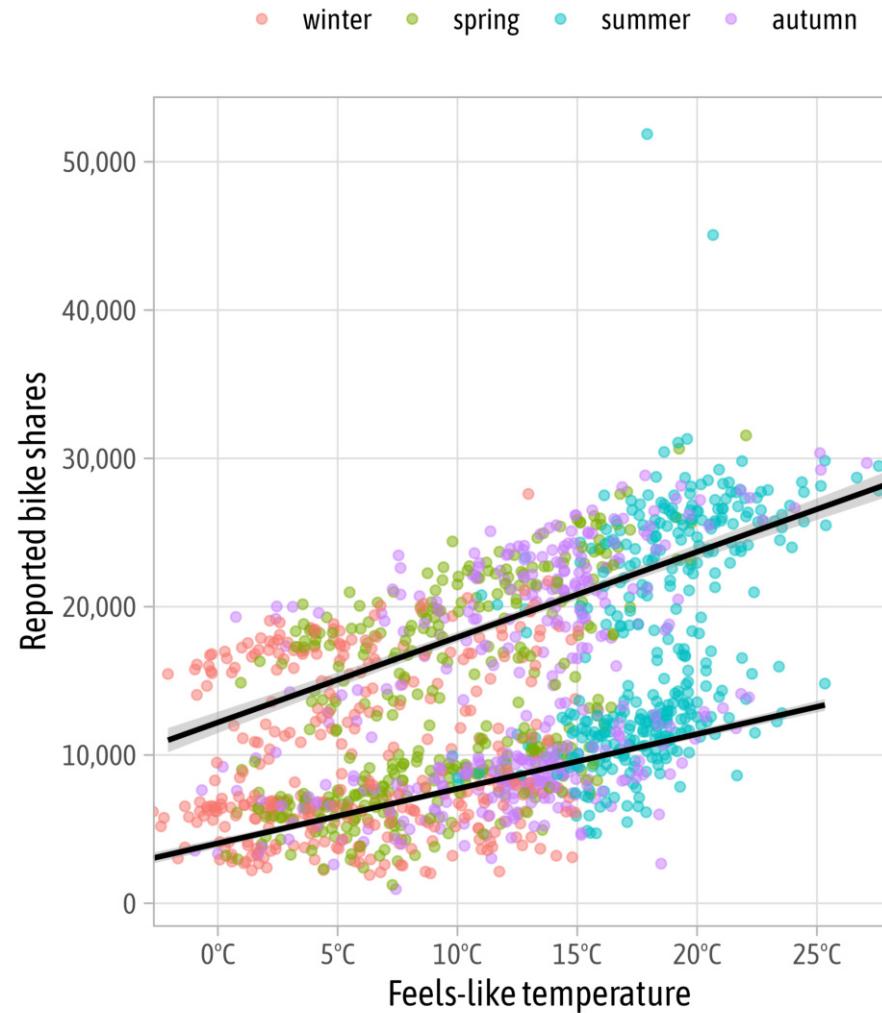
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000,
10    labels = scales::label_comma()
11  ) +
12  scale_color_discrete()
```

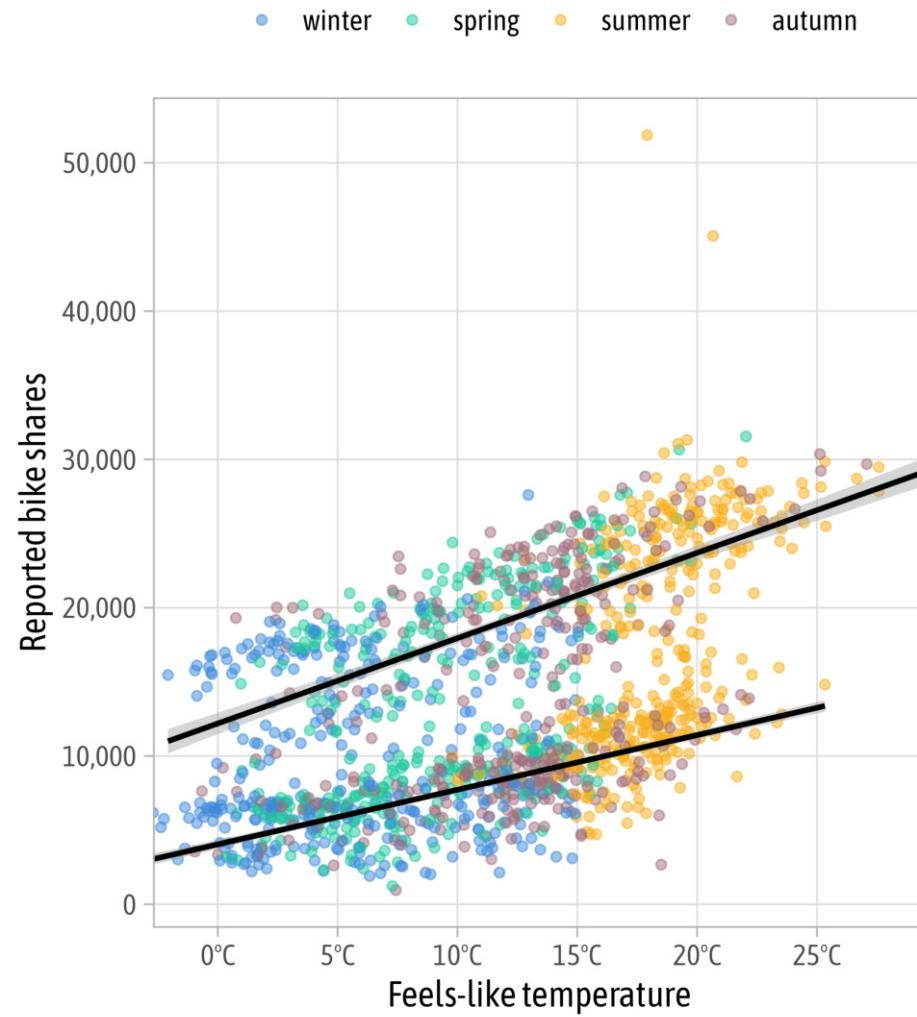
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000,
10    labels = scales::label_comma()
11  ) +
12  scale_color_discrete(
13    type = c("#3c89d9", "#1ec99b", "#f7b01b",
14  )
```

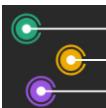
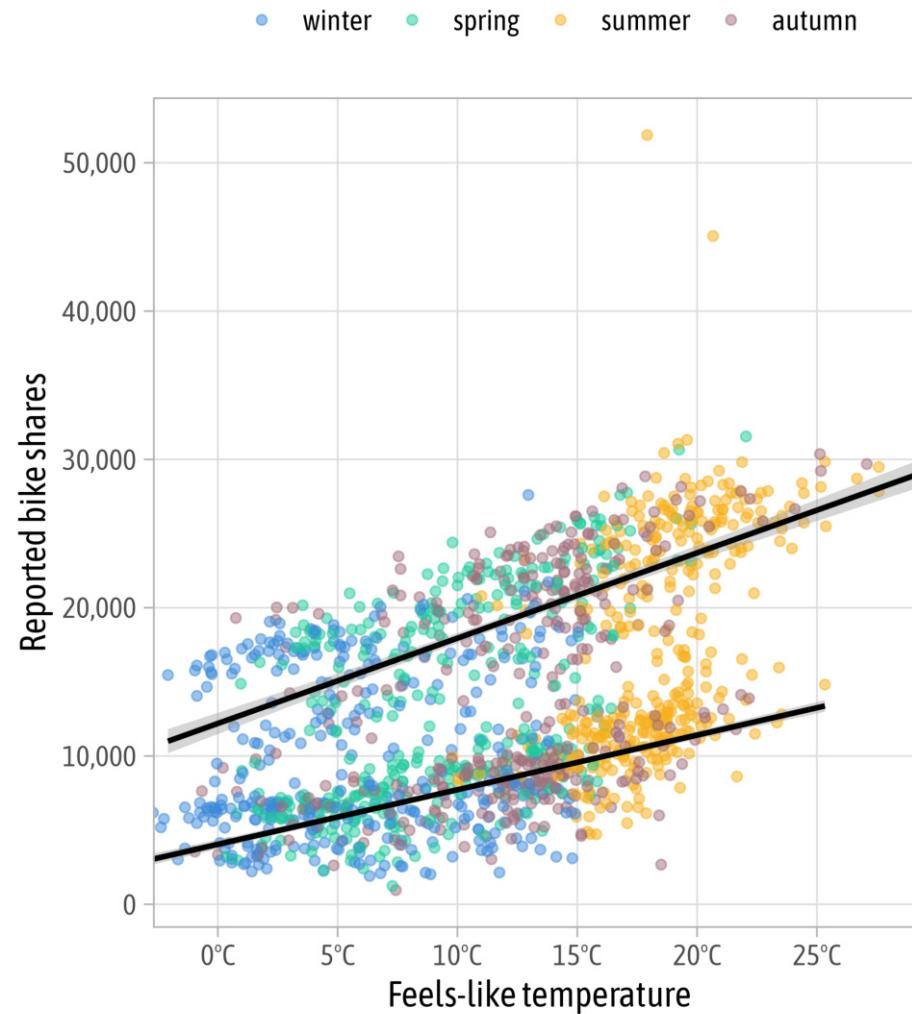
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000,
10    labels = scales::label_comma()
11  ) +
12  scale_color_manual(
13    values = c("#3c89d9", "#1ec99b", "#f7b01b")
14 )
```

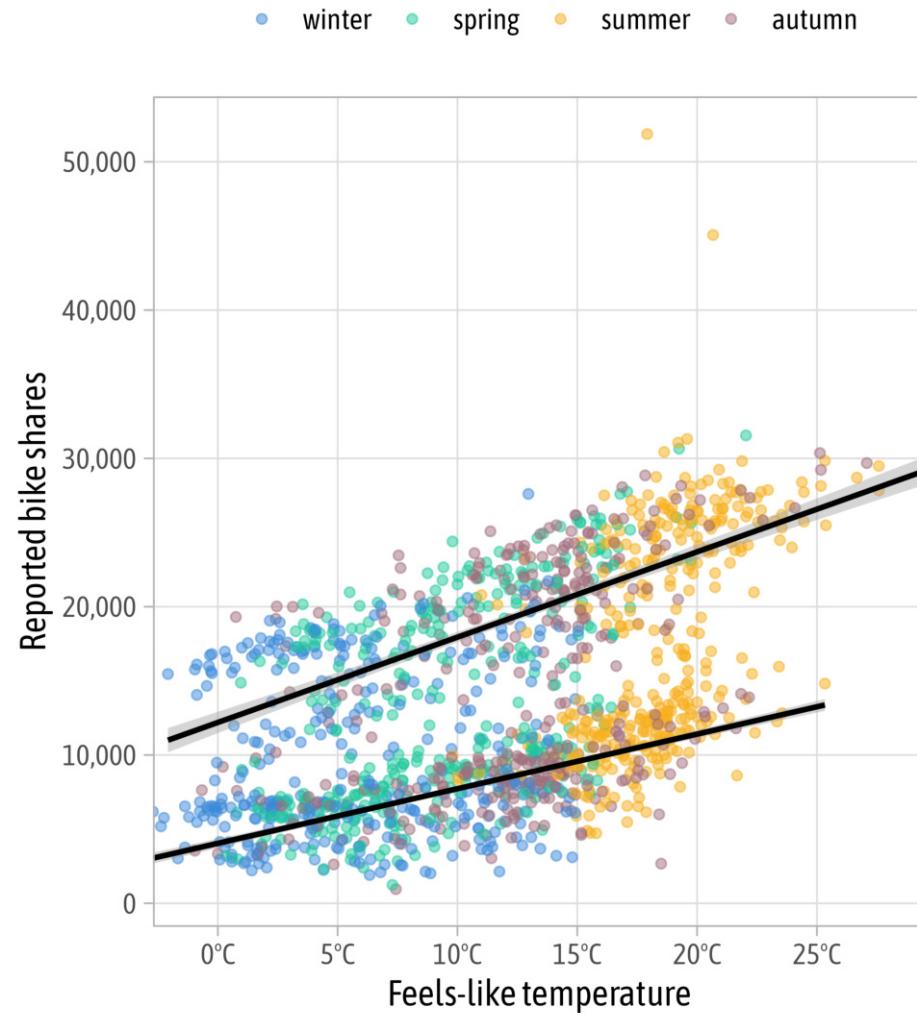
TfL bike sharing trends



Modify Scales

```
1 colors_sorted <- c(
2   autumn = "#a26e7c",
3   spring = "#1ec99b",
4   summer = "#f7b01b",
5   winter = "#3c89d9"
6 )
7
8 g +
9   scale_x_continuous(
10   expand = c(mult = 0, add = 0),
11   breaks = seq(0, 30, by = 5),
12   labels = function(x) paste0(x, "°C"),
13   name = "Feels-like temperature"
14 ) +
15   scale_y_continuous(
16   breaks = 0:5*10000,
17   labels = scales::label_comma()
18 ) +
19   scale_color_manual(
20   values = colors_sorted
21 )
```

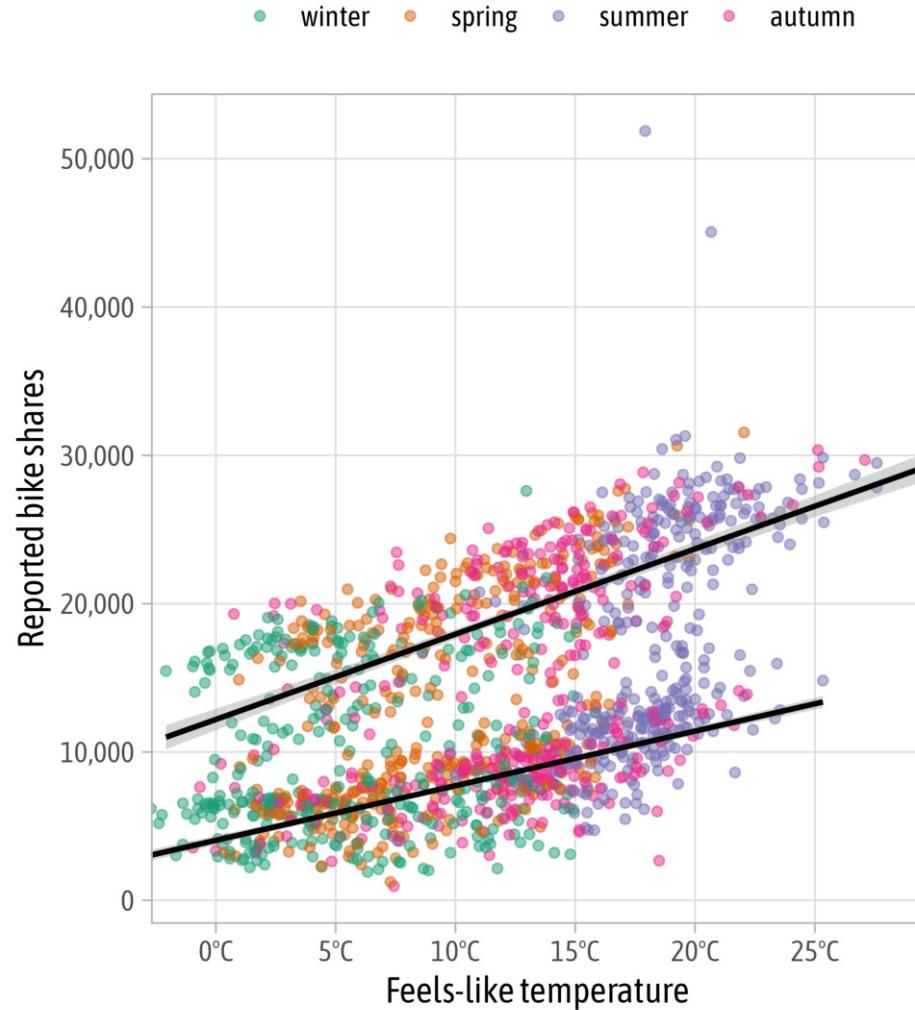
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000,
10    labels = scales::label_comma()
11  ) +
12  scale_color_brewer(
13    palette = "Dark2"
14  )
```

TfL bike sharing trends



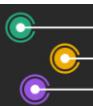
{RColorBrewer}

```
1 RColorBrewer::display.brewer.all()
```



{RColorBrewer}

```
1 RColorBrewer::display.brewer.all(colorblindFriendly = TRUE)
```



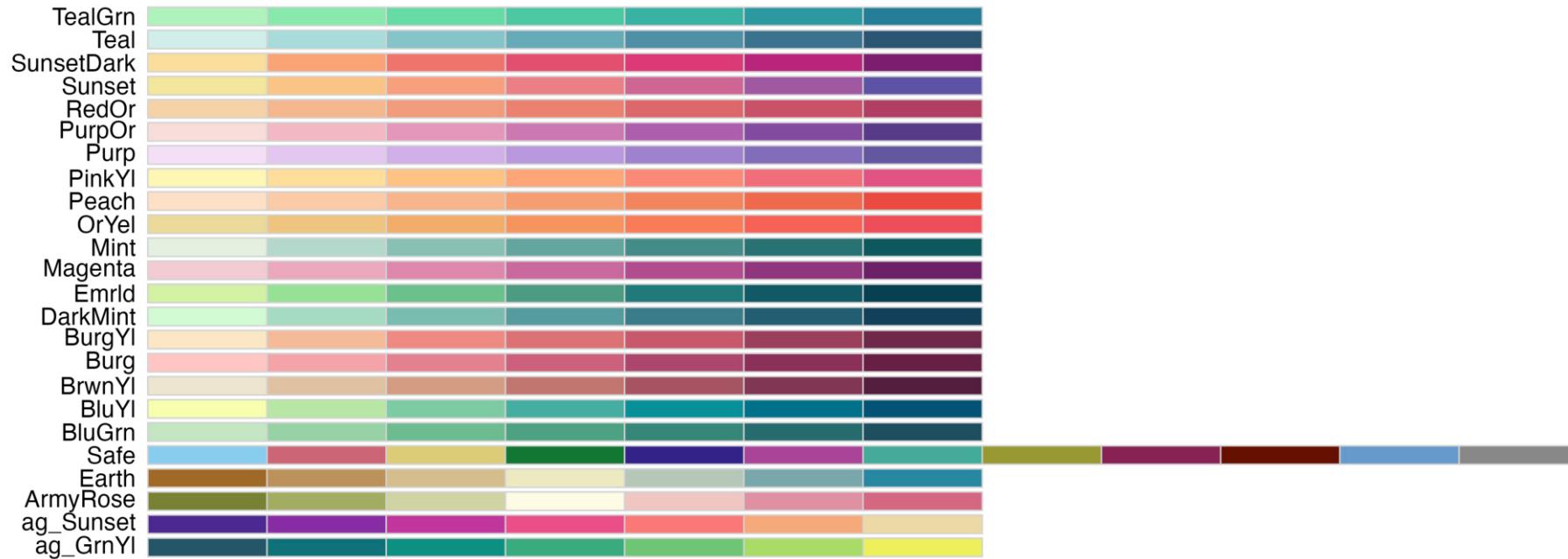
{rcartocolor}

```
1 # install.packages("rcartocolor")
2 rcartocolor::display_carto_all()
```



{rcartocolor}

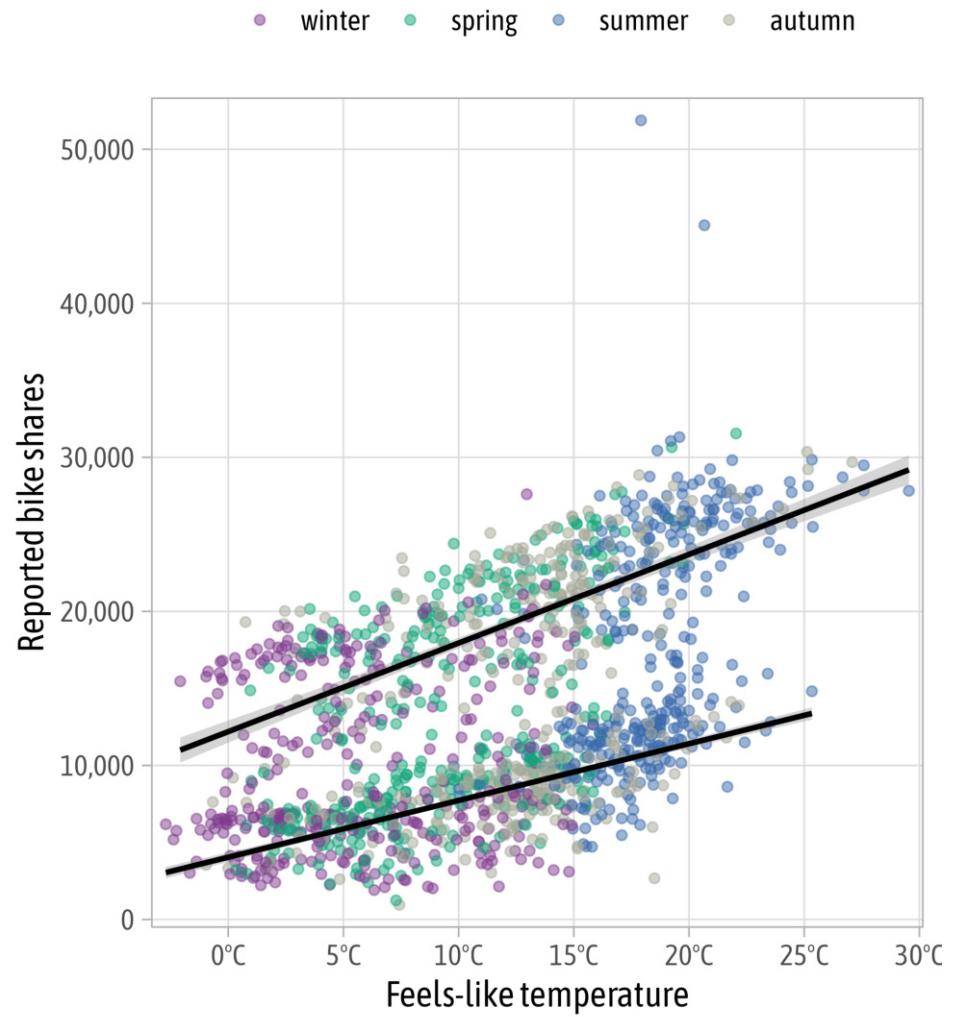
```
1 # install.packages("rcartocolor")
2 rcartocolor::display_carto_all(colorblind_friendly = TRUE)
```



{rcartocolor}

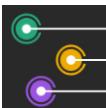
```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0.02, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     expand = c(mult = 0, add = 1500),
10    breaks = 0:5*10000,
11    labels = scales::label_comma()
12  ) +
13  rcartocolor::scale_color_carto_d(
14    palette = "Bold"
15  )
```

TfL bike sharing trends



{scico}

```
1 # install.packages("scico")
2 scico::scico_palette_show()
```



{scico}

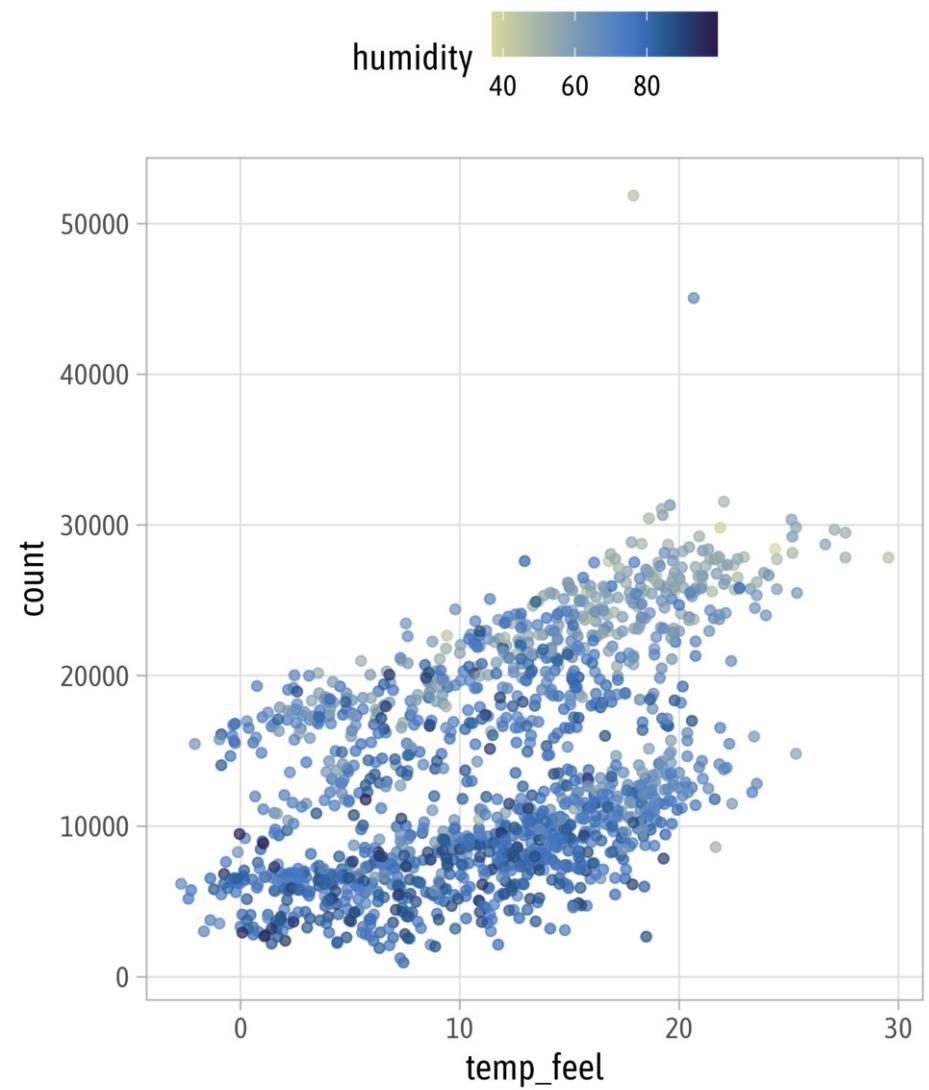
```
1 g +
  scale_x_continuous(
    expand = c(mult = 0.02, add = 0),
    breaks = seq(0, 30, by = 5),
    labels = function(x) paste0(x, "°C"),
    name = "Feels-like temperature"
  ) +
  scale_y_continuous(
    expand = c(mult = 0, add = 1500),
    breaks = 0:5*10000,
    labels = scales::label_comma()
  ) +
  scico::scale_color_scico_d(
    palette = "hawaii"
  )
```

TfL bike sharing trends



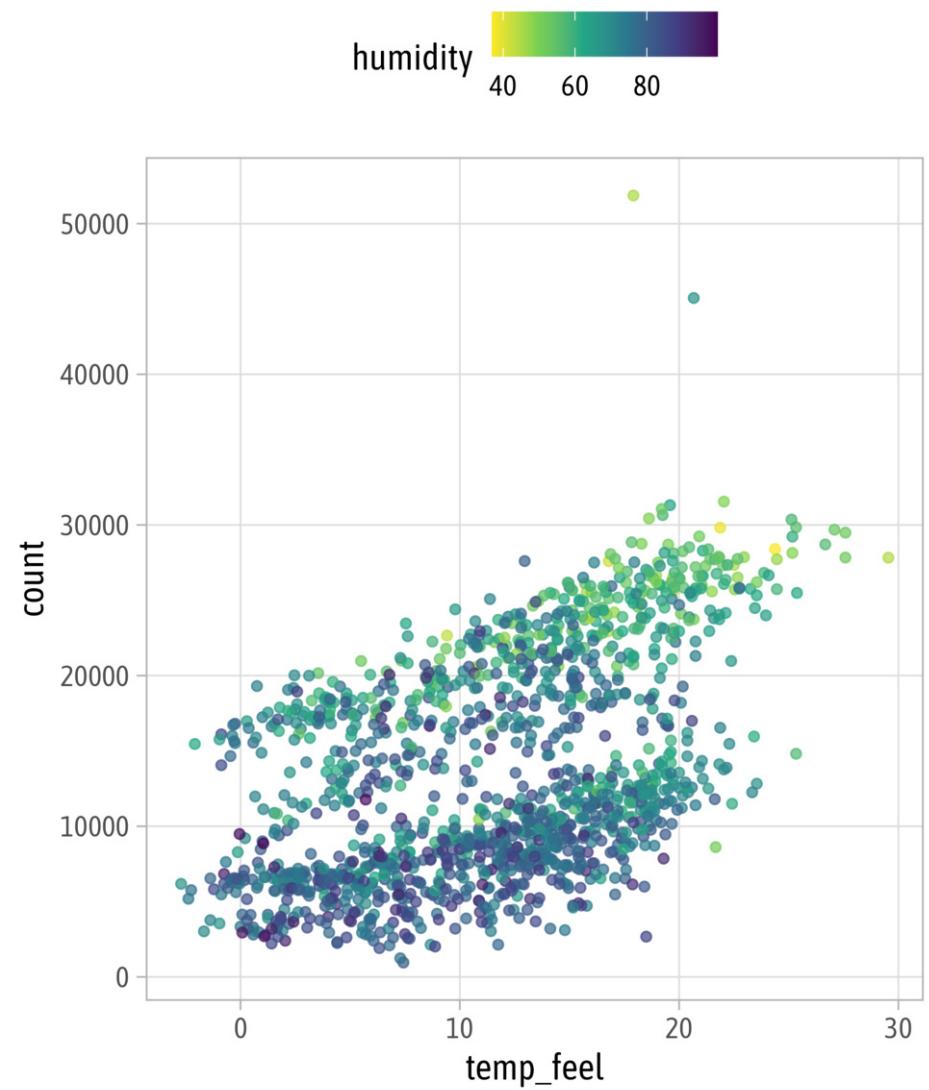
{scico}

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = humidity)  
5 ) +  
6 geom_point(alpha = .7) +  
7 scico::scale_color_scico(  
8   palette = "davos",  
9   direction = -1,  
10  end = .8  
11 )
```



{scico}

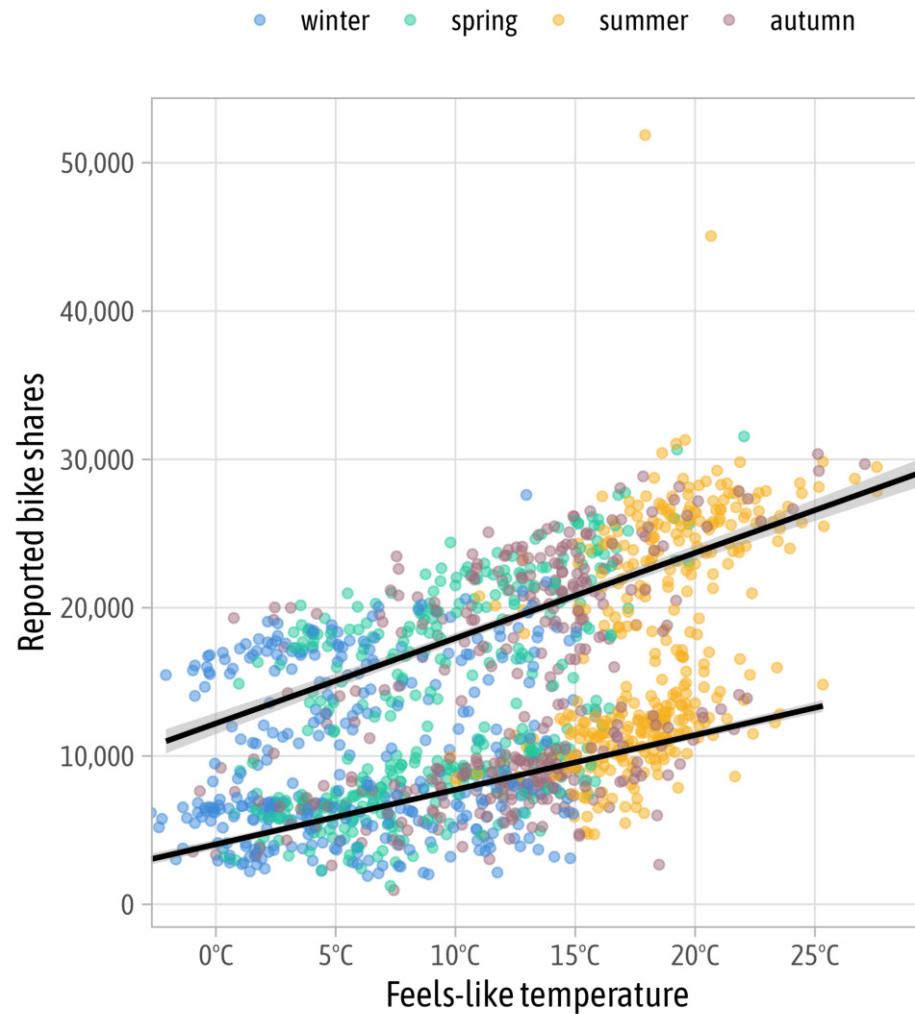
```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = humidity)  
5 ) +  
6 geom_point(alpha = .7) +  
7 scale_color_viridis_c(  
8   direction = -1  
9 )
```



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000,
10    labels = scales::label_comma()
11  ) +
12  scale_color_manual(
13    values = colors_sorted,
14    name = NULL
15  )
```

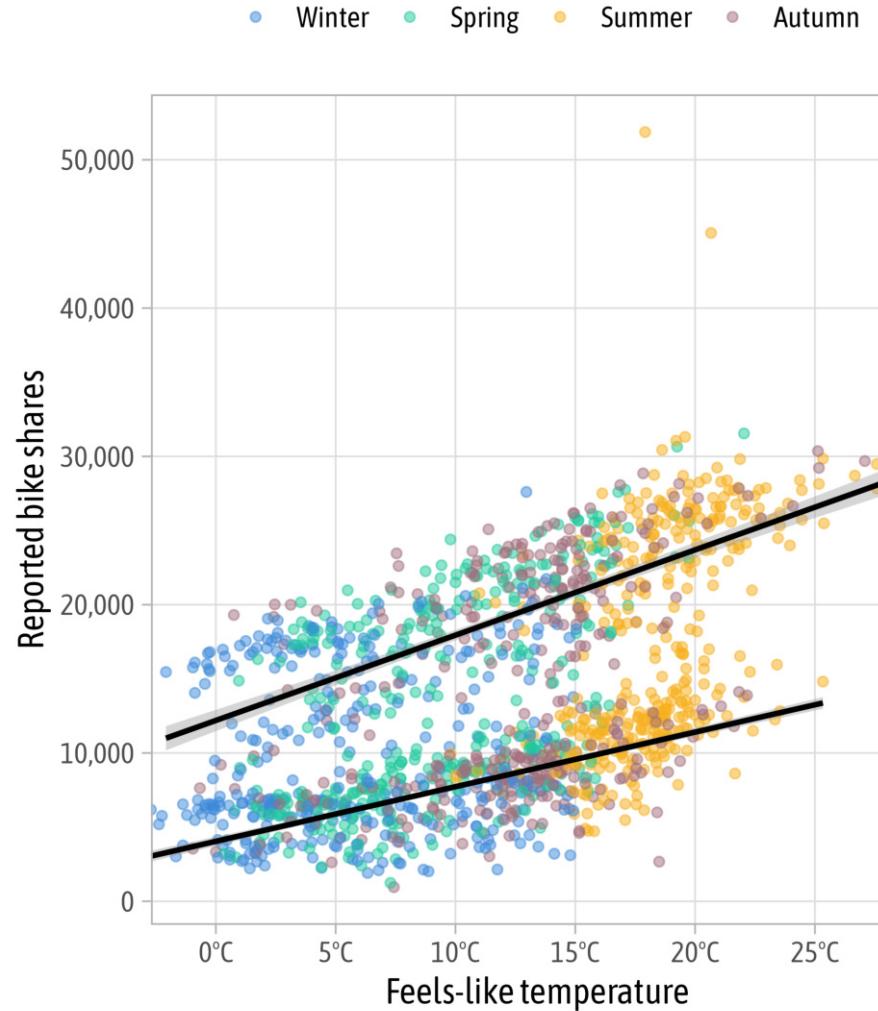
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     breaks = 0:5*10000,
10    labels = scales::label_comma()
11  ) +
12  scale_color_manual(
13    values = colors_sorted,
14    name = NULL,
15    labels = stringr::str_to_title
16  )
```

TfL bike sharing trends



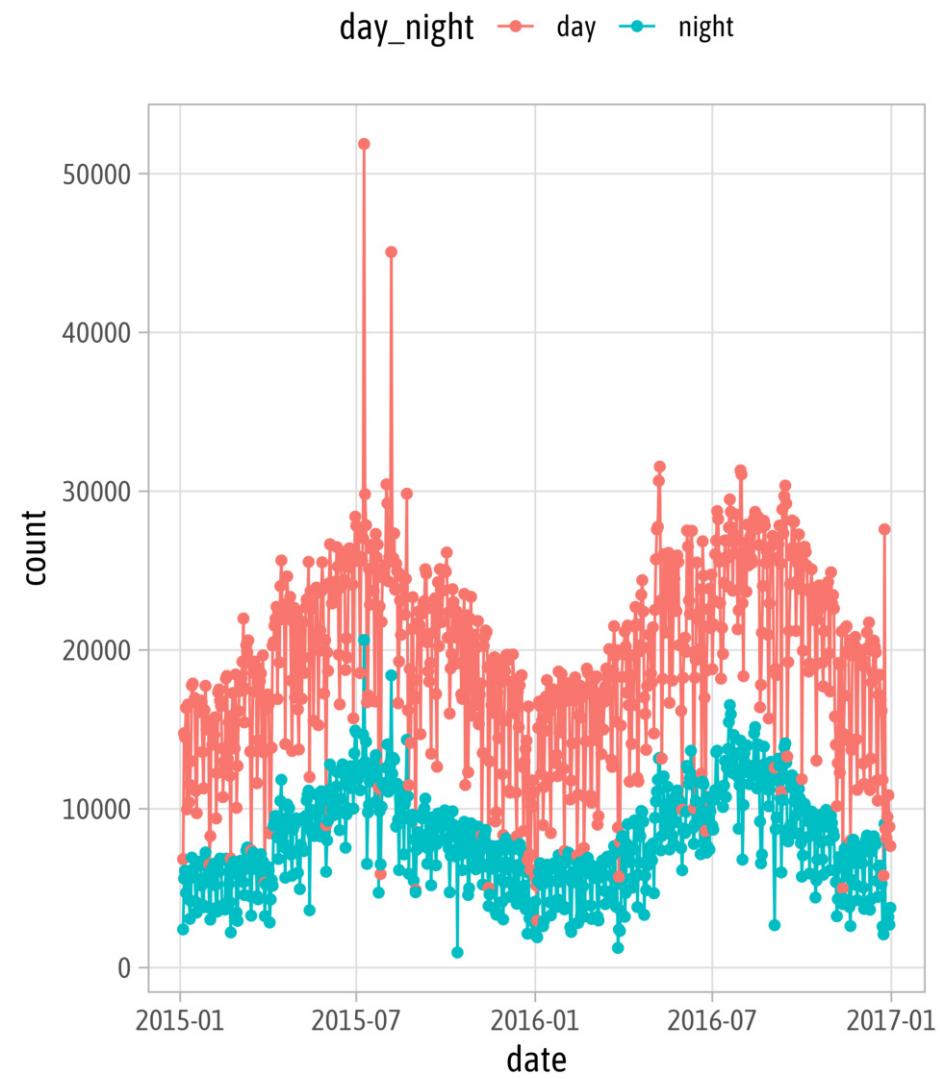
Your Turn

- **Style the time series of reported bike shares.**
 - Add a plot title and meaningful axis and legend titles.
 - Use a custom set of colors for day and night.
 - Explore complete themes and pick your favorite.
 - **Bonus:** Modify the x axis to show every four months along with the year.



Solution Exercise

```
1 p <-
2   ggplot(
3     bikes,
4     aes(x = date, y = count,
5           color = day_night)
6   ) +
7   geom_line() +
8   geom_point()
9 )
```

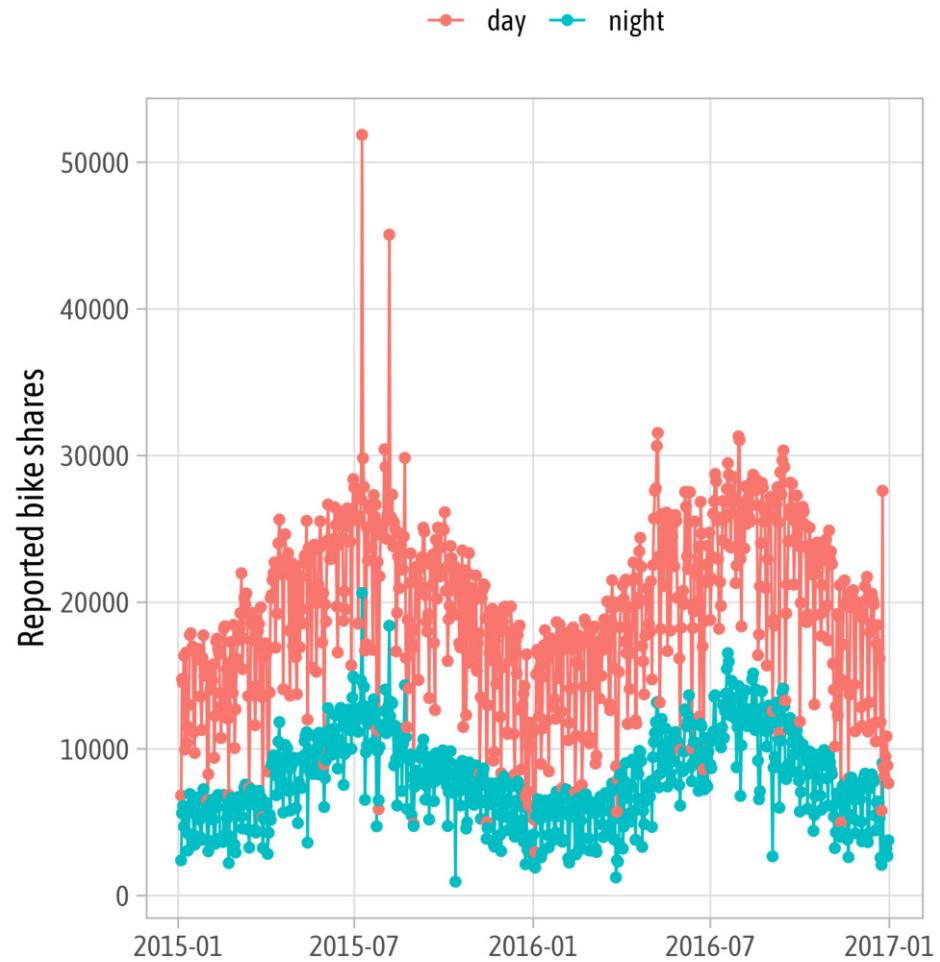


Solution Exercise

```
1 p +
2   labs(
3     title = "Most bikes are rented during sum-
4     subtitle = "Reported rents of TfL bikes f-
5     x = NULL,
6     y = "Reported bike shares",
7     color = NULL
8   )
```

Most bikes are rented during summer days

Reported rents of TfL bikes for 2015 and 2016.



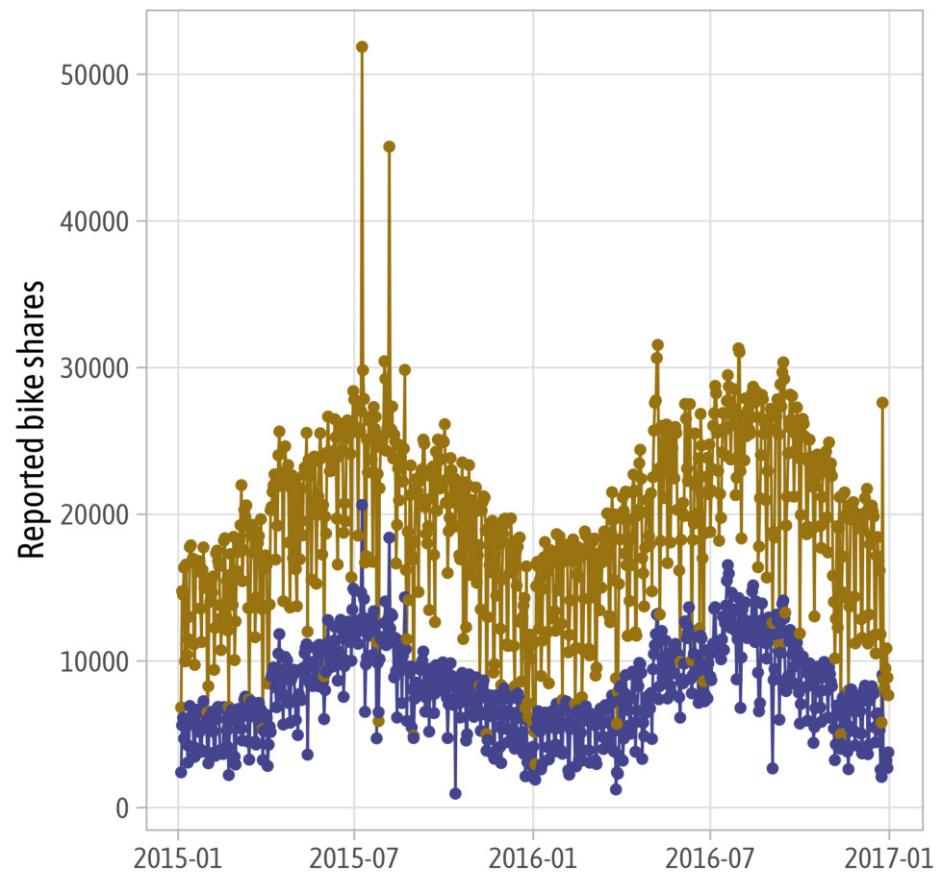
Solution Exercise

```
1 p +
2   scale_color_manual(
3     values = c("#98730F", "#44458e"),
4     labels = c("Day (6am-6pm)", "Night (6pm-6"),
5   ) +
6     labs(title = "Most bikes are rented during",
7       subtitle = "Reported rents of TfL bike",
8       x = NULL, y = "Reported bike shares",
```

Most bikes are rented during summer days

Reported rents of TfL bikes for 2015 and 2016.

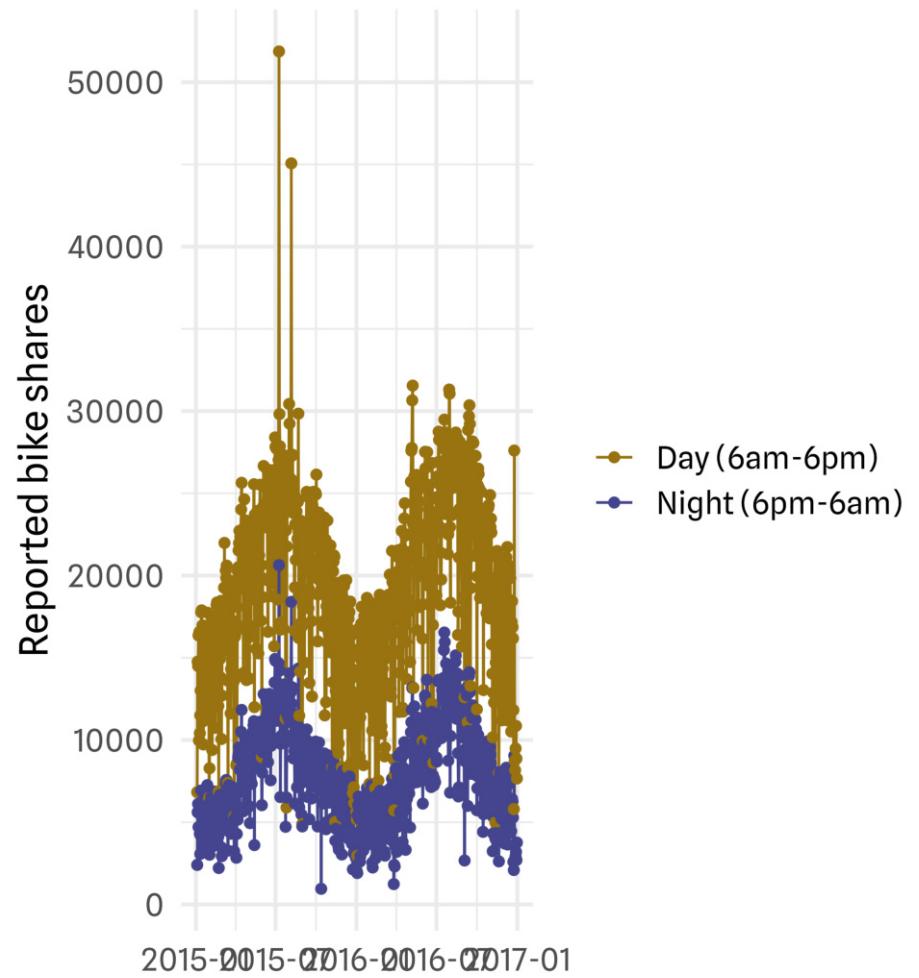
—●— Day (6am-6pm) —●— Night (6pm-6am)



Solution Exercise

```
1 p +
2   scale_color_manual(
3     values = c("#98730F", "#44458e"),
4     labels = c("Day (6am-6pm)", "Night (6pm-6am)"),
5   ) +
6   theme_minimal(base_family = "Spline Sans",
7   labs(title = "Most bikes are rented during summer",
8       subtitle = "Reported rents of TfL bike shares",
9       x = NULL, y = "Reported bike shares",
```

Most bikes are rented during summer
Reported rents of TfL bikes for 2015 and 2016

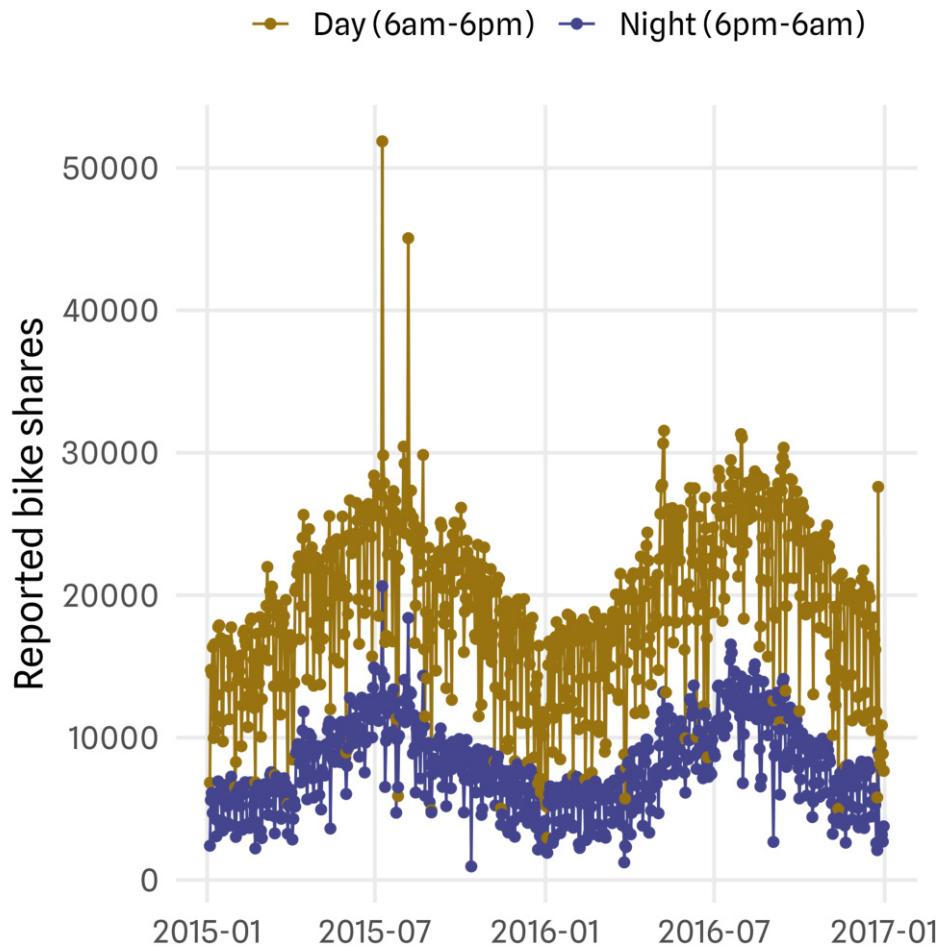


Solution Exercise

```
1 p +
2   scale_color_manual(
3     values = c("#98730F", "#44458e"),
4     labels = c("Day (6am-6pm)", "Night (6pm-6am)"),
5   ) +
6   theme_minimal(base_family = "Spline Sans",
7   theme(
8     panel.grid.minor = element_blank(),
9     plot.title = element_text(face = "bold"),
10    plot.title.position = "plot",
11    legend.position = "top"
12  ) +
13  labs(title = "Most bikes are rented during summer days",
14    subtitle = "Reported rents of TfL bike",
15    x = NULL, y = "Reported bike shares",
```

Most bikes are rented during summer days

Reported rents of TfL bikes for 2015 and 2016.

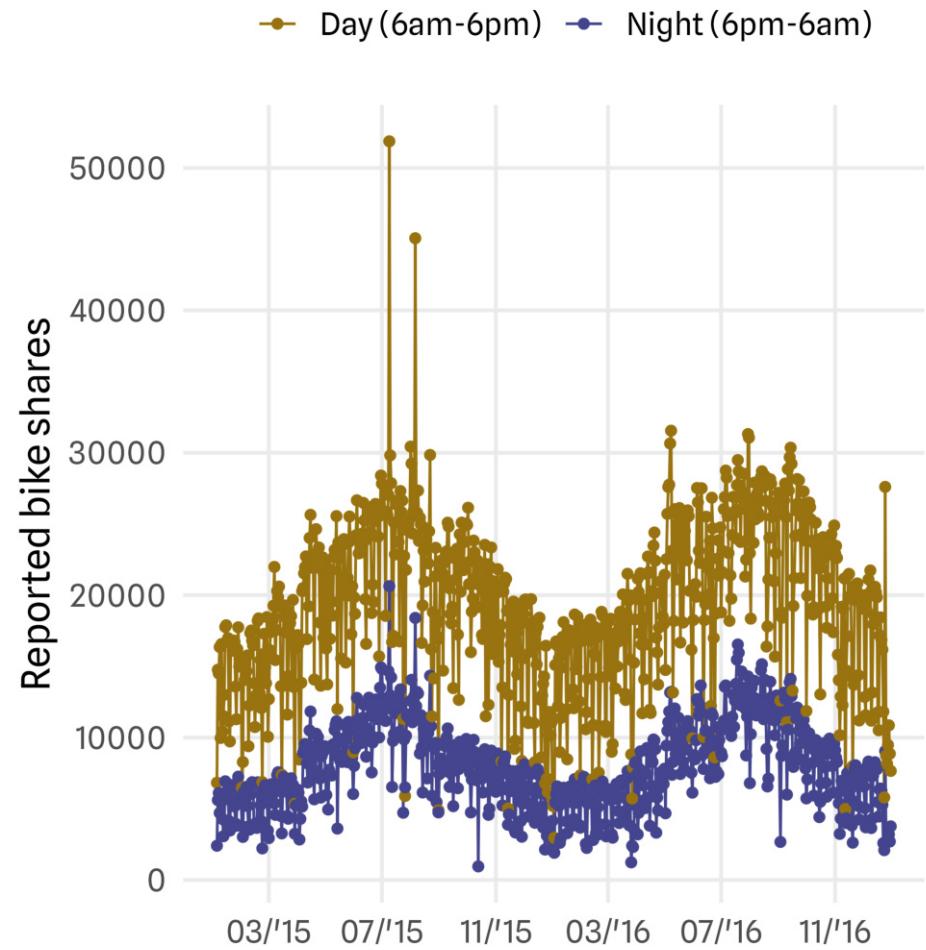


Solution Exercise

```
1 p +
2   scale_x_date(
3     date_breaks = "4 months",
4     date_labels = "%m/%y"
5   ) +
6   scale_color_manual(
7     values = c("#98730F", "#44458e"),
8     labels = c("Day (6am-6pm)", "Night (6pm-6am")
9   ) +
10  theme_minimal(base_family = "Spline Sans",
11    theme(
12      panel.grid.minor = element_blank(),
13      plot.title = element_text(face = "bold"),
14      plot.title.position = "plot",
15      legend.position = "top"
16    ) +
17    labs(title = "Most bikes are rented during summer days",
18         subtitle = "Reported rents of TfL bike",
19         x = NULL, y = "Reported bike shares",
```

Most bikes are rented during summer days

Reported rents of TfL bikes for 2015 and 2016.



Facets



Facets

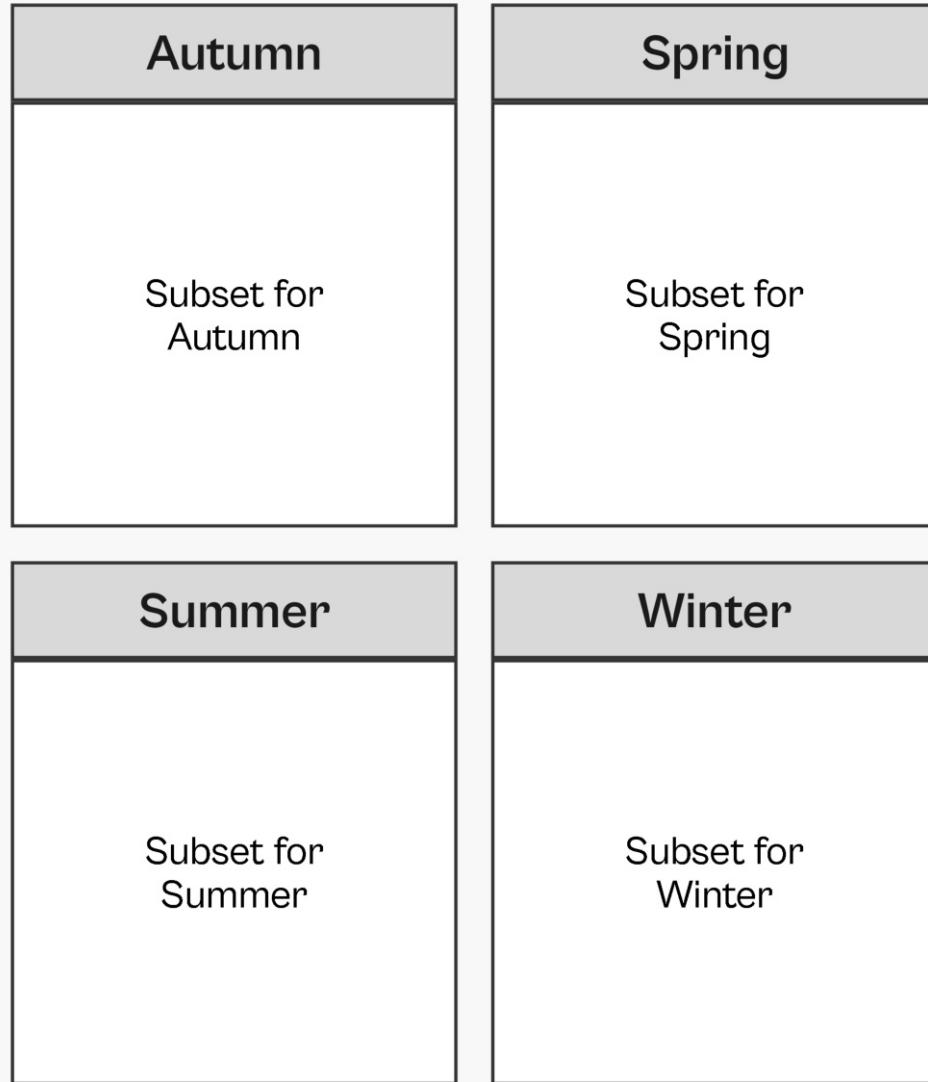
= split variables to multiple panels

Facets are also known as:

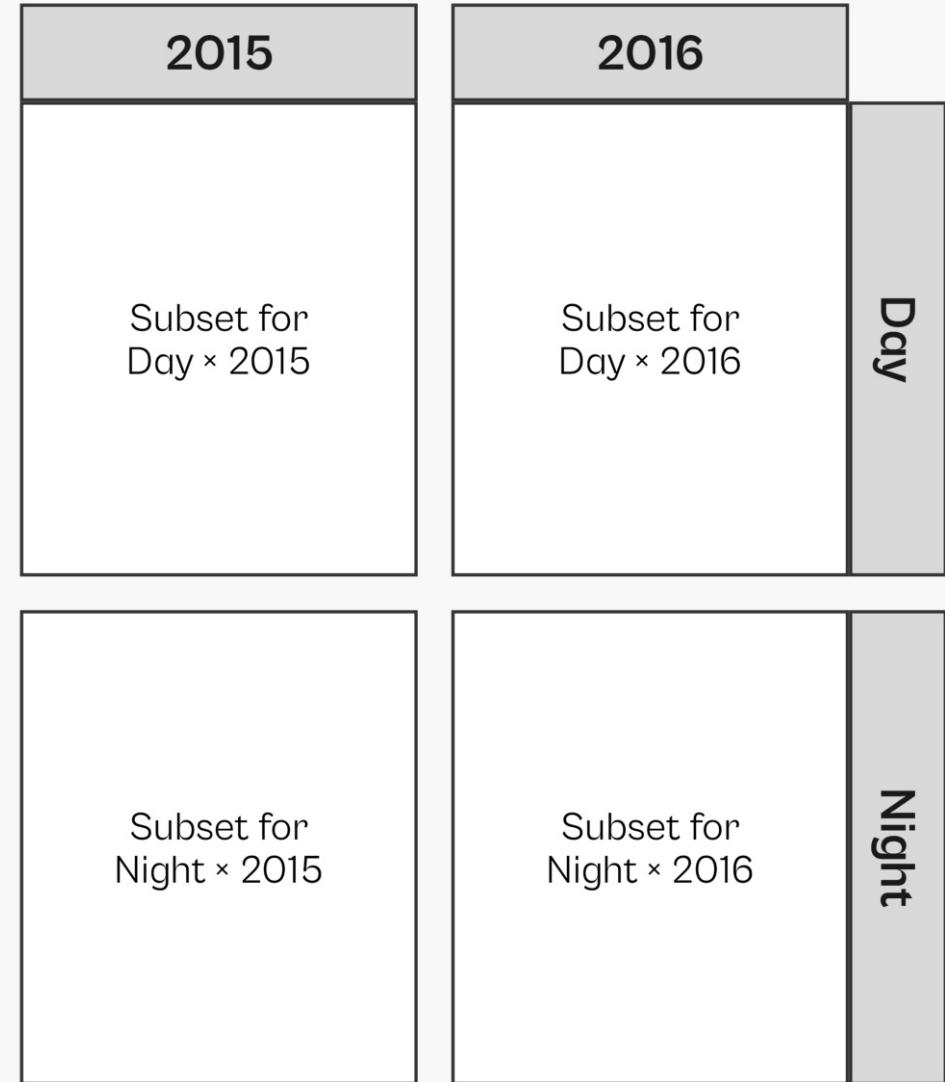
- small multiples
- trellis graphs
- lattice plots
- conditioning



`facet_wrap()`



`facet_grid()`

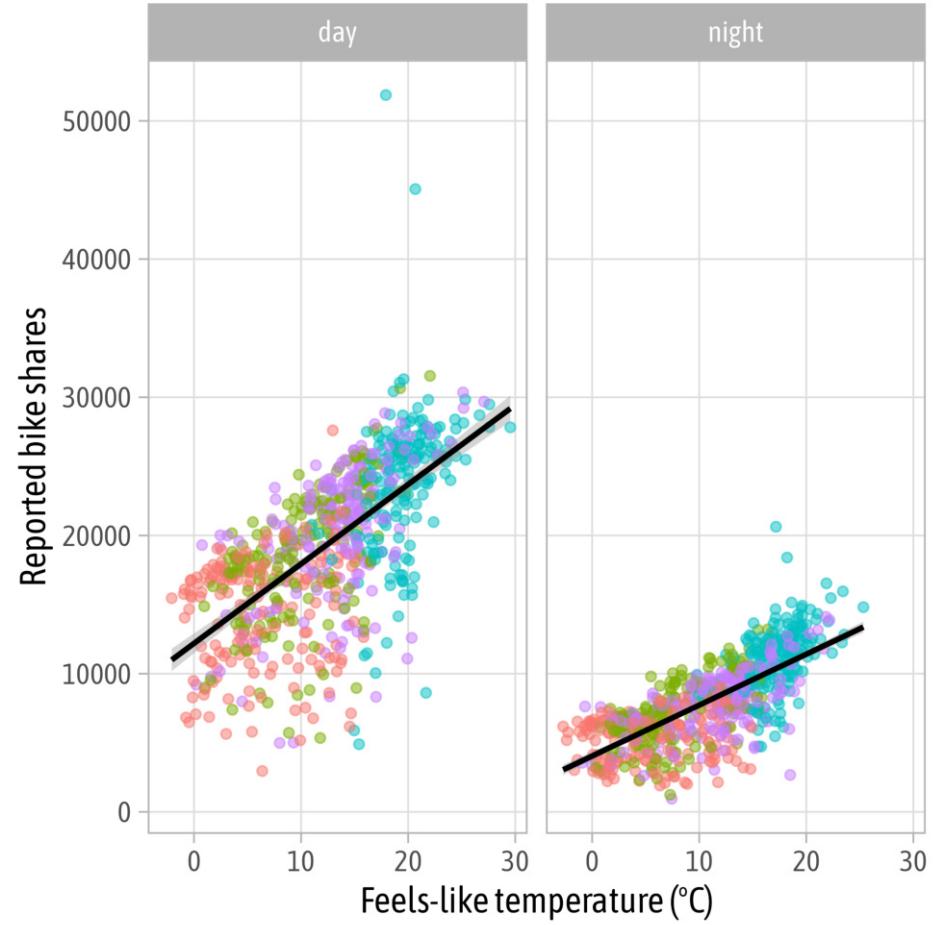


Wrapped Facets

```
1 g +  
2 facet_wrap(  
3 vars(day_night)  
4 )
```

TfL bike sharing trends

● winter ● spring ● summer ● autumn

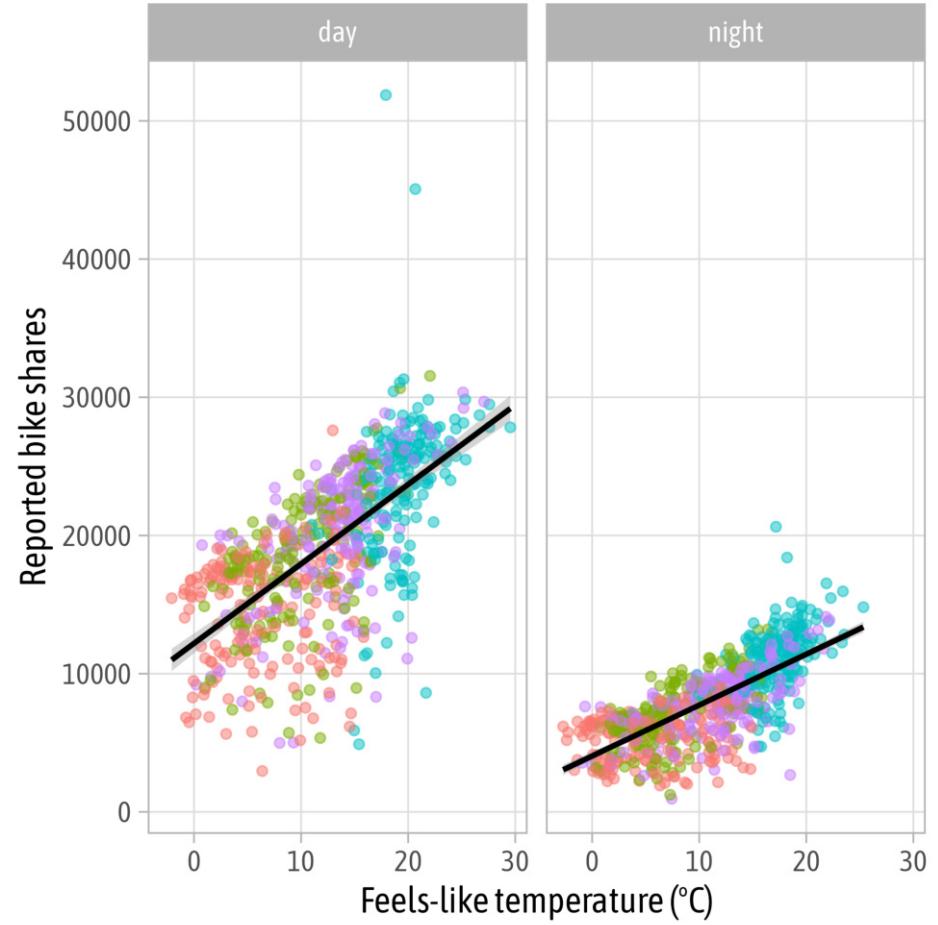


Wrapped Facets

```
1 g +  
2   facet_wrap(  
3     ~ day_night  
4   )
```

TfL bike sharing trends

● winter ● spring ● summer ● autumn

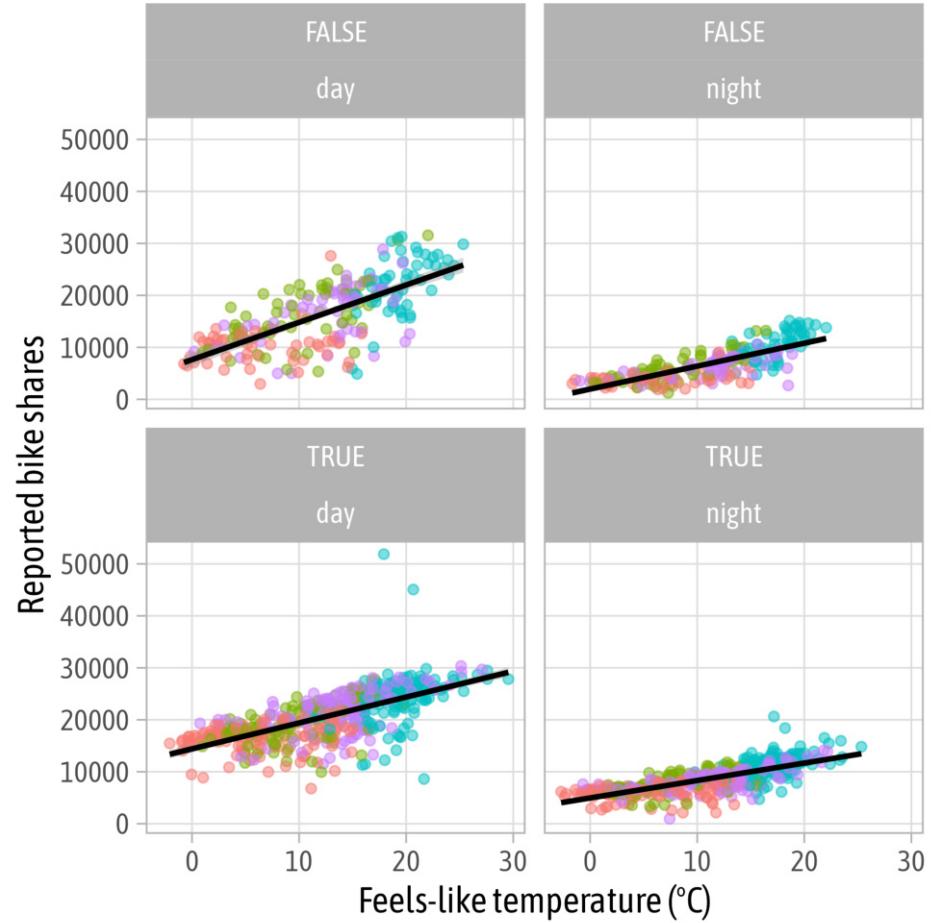


Facet Multiple Variables

```
1 g +  
2 facet_wrap(  
3   ~ is_workday + day_night  
4 )
```

TfL bike sharing trends

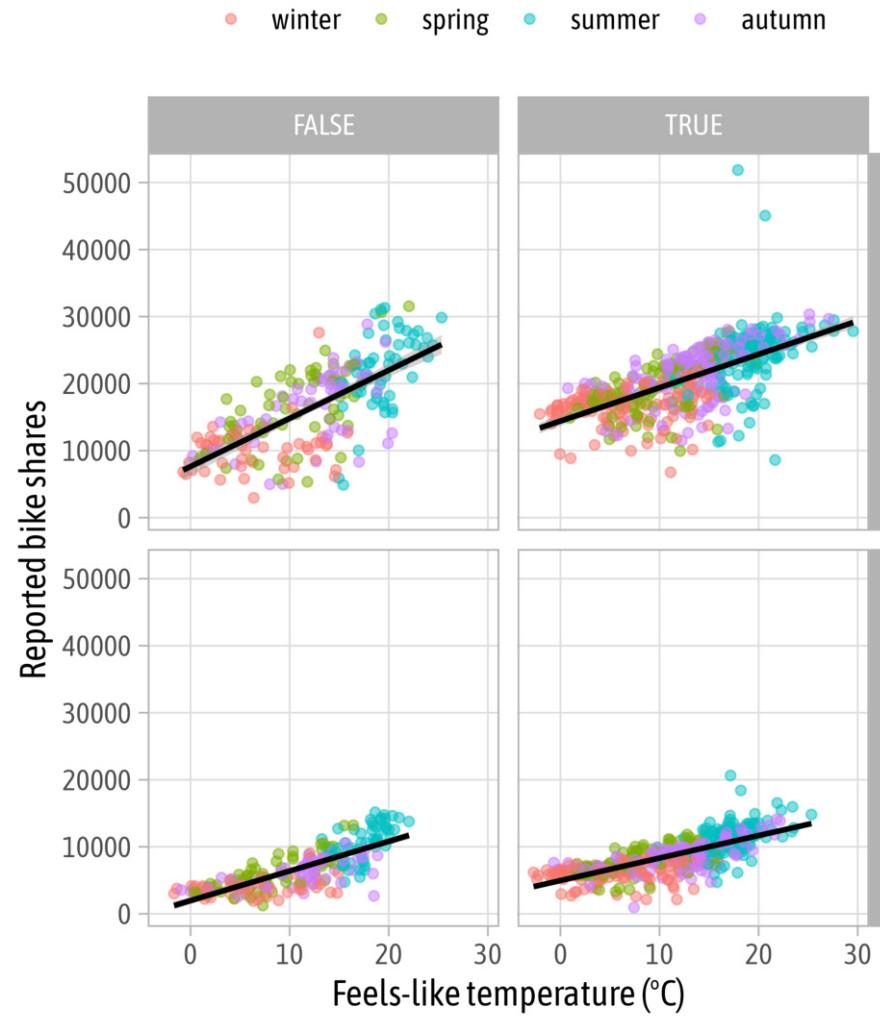
● winter ● spring ● summer ● autumn



Gridded Facets

```
1 g +  
2   facet_grid(  
3     rows = vars(day_night),  
4     cols = vars(is_workday)  
5   )
```

TfL bike sharing trends

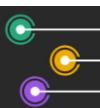
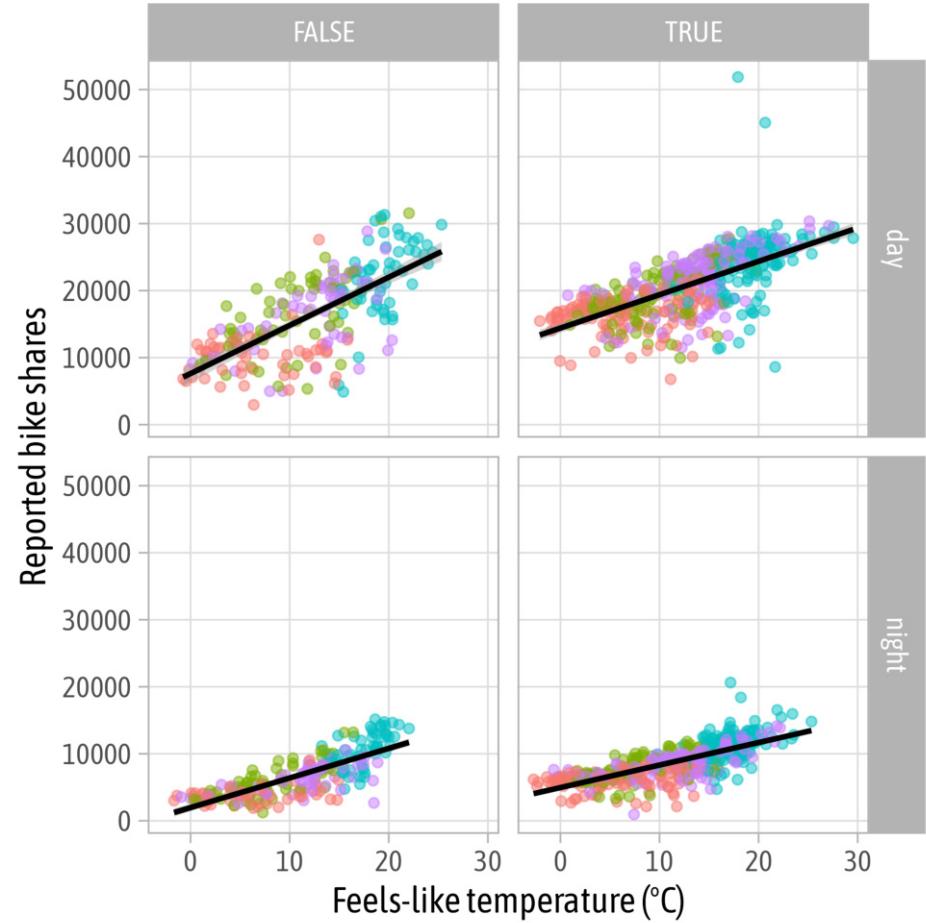


Gridded Facets

```
1 g +  
2 facet_grid(  
3   day_night ~ is_workday  
4 )
```

TfL bike sharing trends

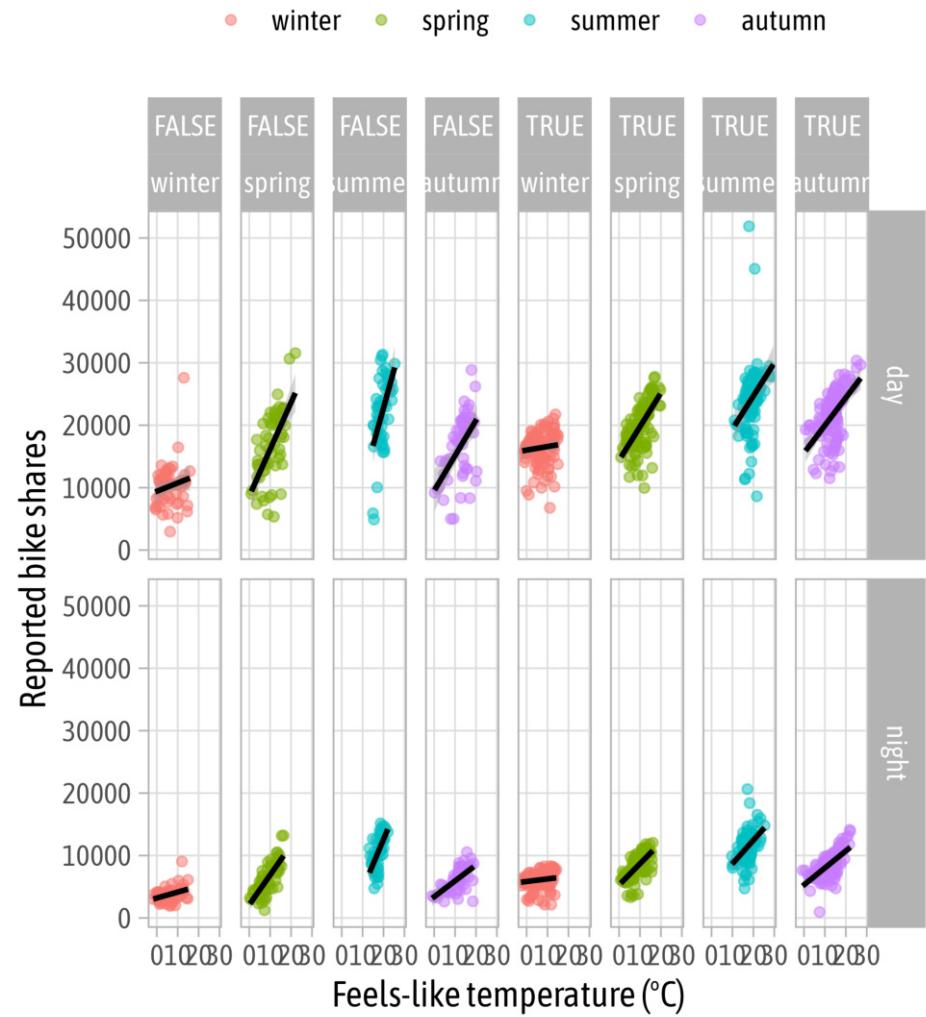
● winter ● spring ● summer ● autumn



Facet Multiple Variables

```
1 g +  
2 facet_grid(  
3   day_night ~ is_workday + season  
4 )
```

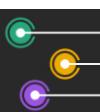
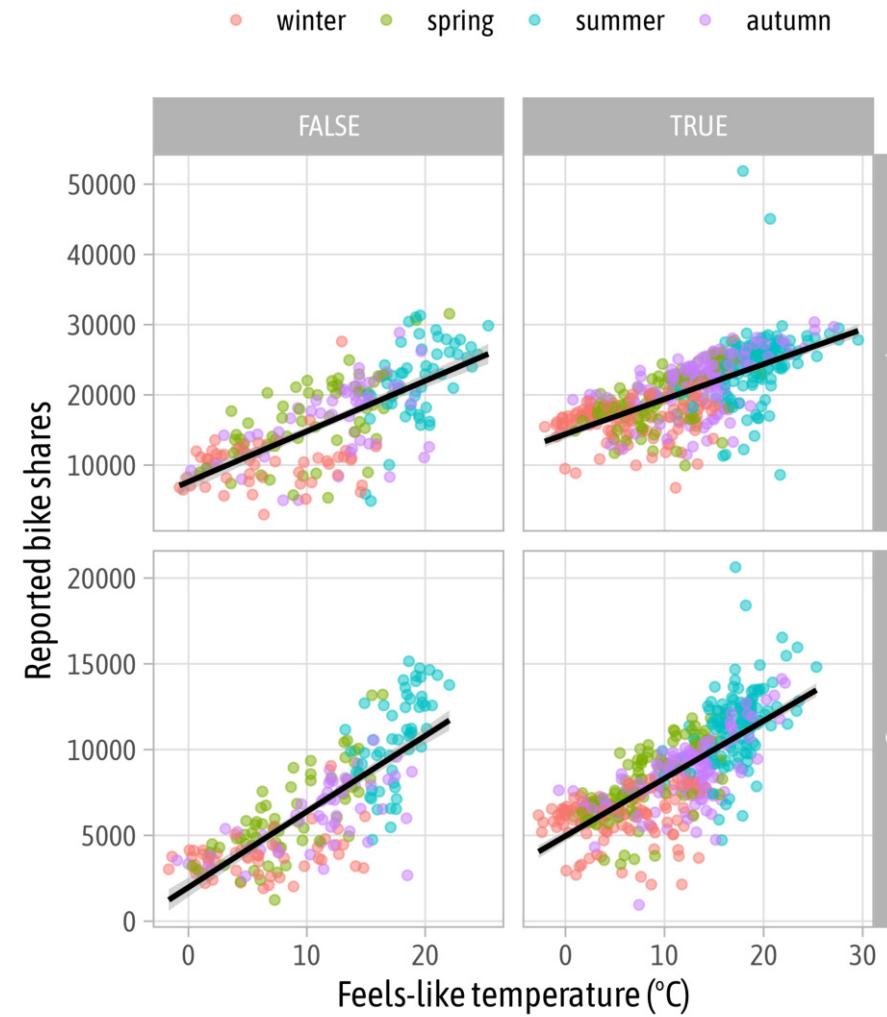
TfL bike sharing trends



Facet Options: Free Scaling

```
1 g +  
2   facet_grid(  
3     day_night ~ is_workday,  
4     scales = "free"  
5   )
```

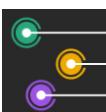
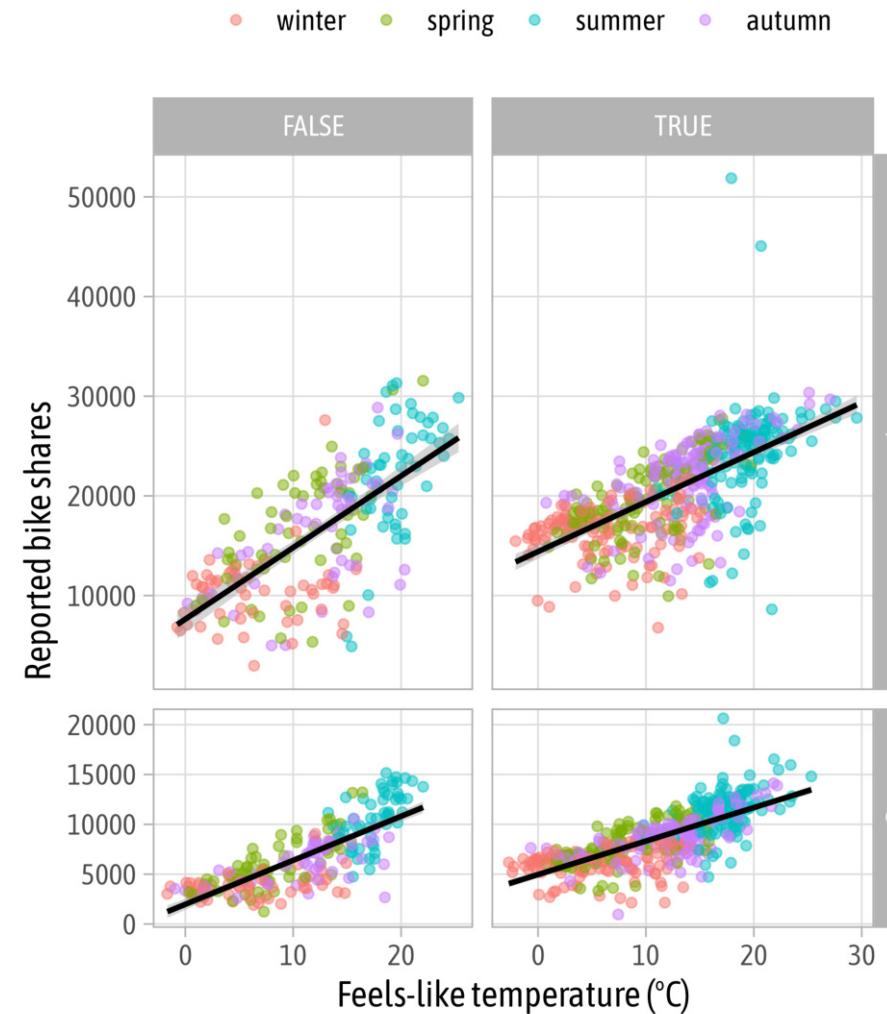
TfL bike sharing trends



Facet Options: Proportional Spacing

```
1 g +  
2   facet_grid(  
3     day_night ~ is_workday,  
4     scales = "free",  
5     space = "free"  
6   )
```

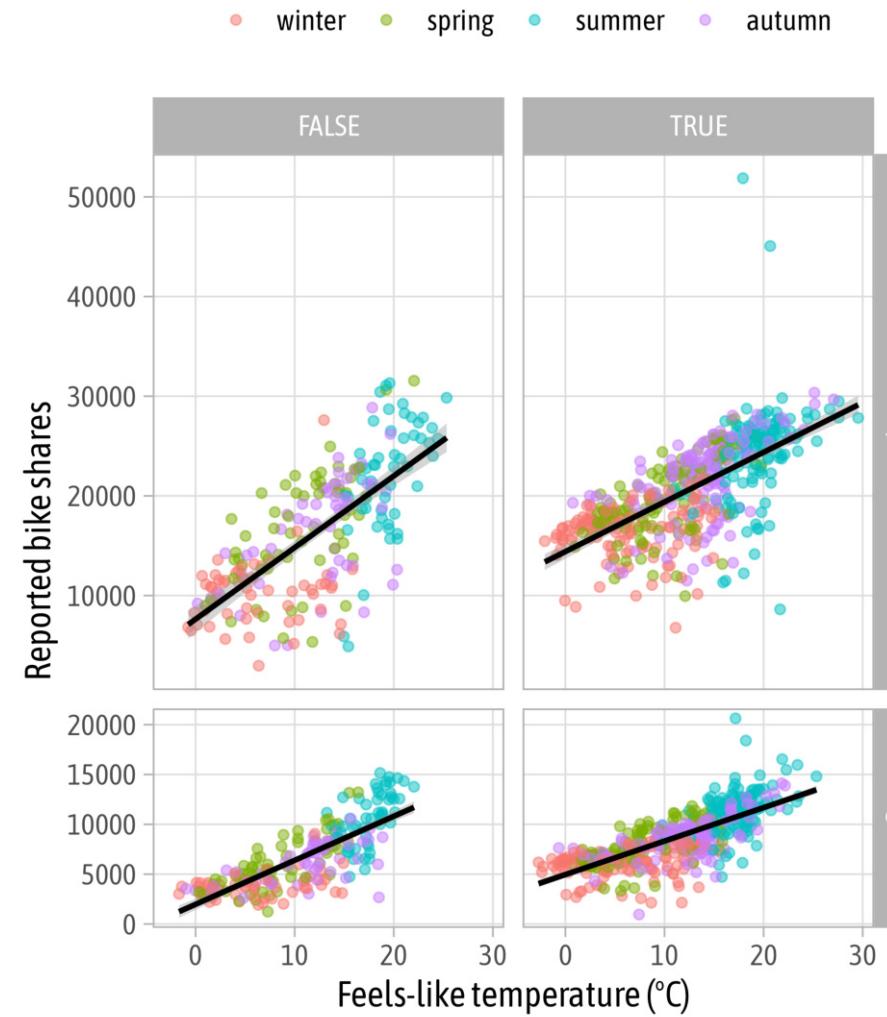
TfL bike sharing trends



Facet Options: Proportional Spacing

```
1 g +  
2   facet_grid(  
3     day_night ~ is_workday,  
4     scales = "free_y",  
5     space = "free_y"  
6   )
```

TfL bike sharing trends

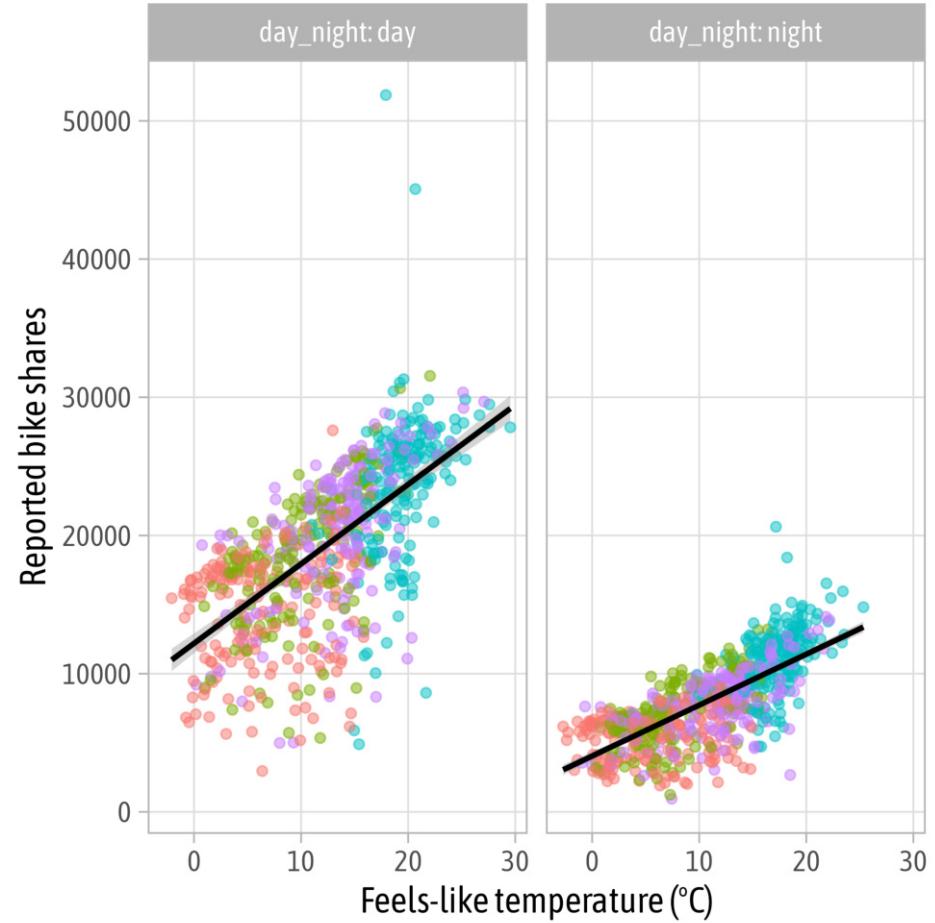


Facet Labellers

```
1 g +  
2   facet_wrap(  
3     ~ day_night,  
4     labeller = label_both  
5   )
```

TfL bike sharing trends

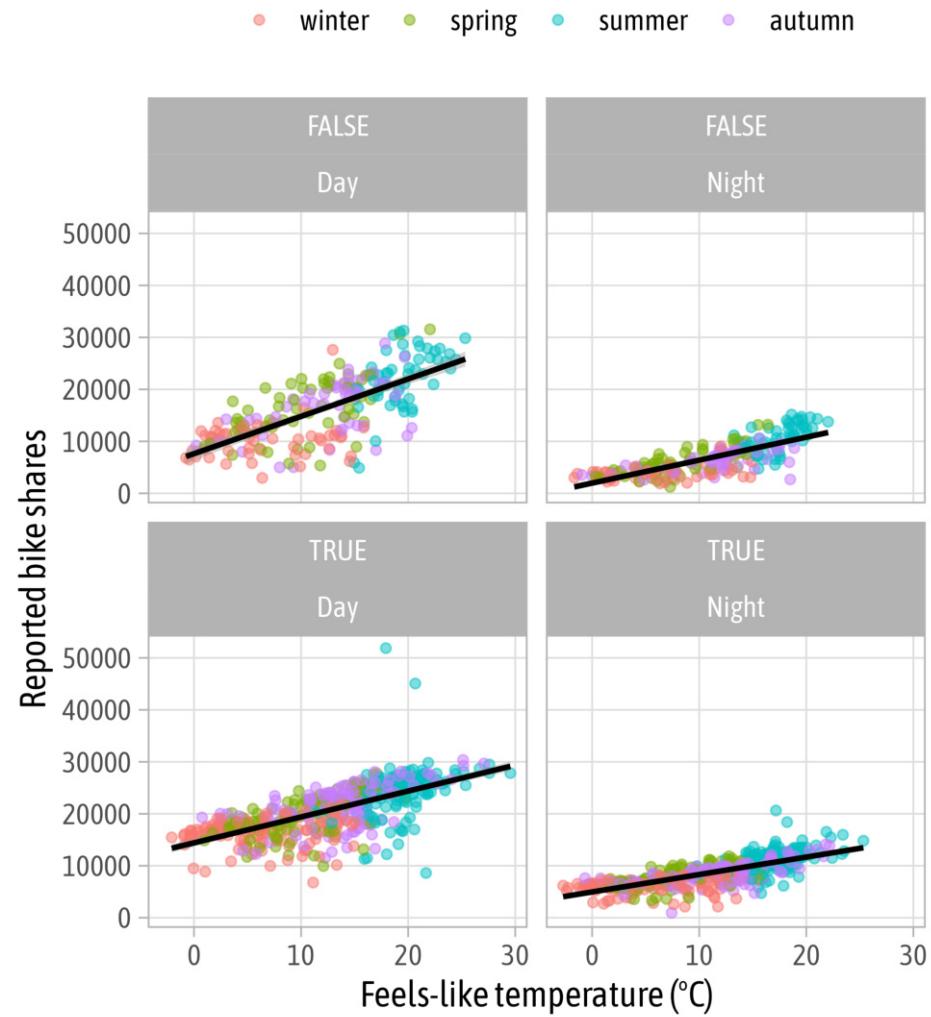
● winter ● spring ● summer ● autumn



Facet Labellers

```
1 g +
2   facet_wrap(
3     ~ is_workday + day_night,
4     labeller = labeller(
5       day_night = stringr::str_to_title
6     )
7   )
```

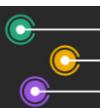
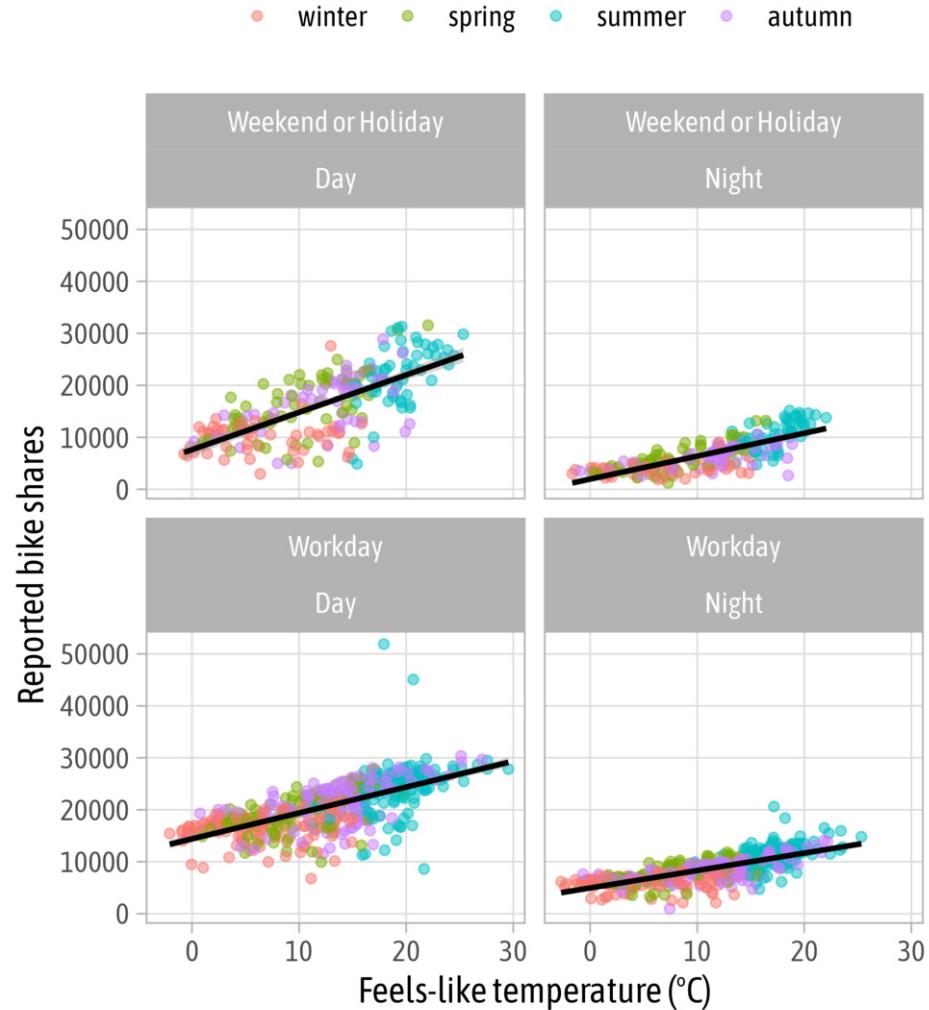
TfL bike sharing trends



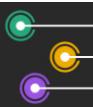
Facet Labellers

```
1 codes <- c(  
2   `TRUE` = "Workday",  
3   `FALSE` = "Weekend or Holiday"  
4 )  
5  
6 g +  
7 facet_wrap(  
8   ~ is_workday + day_night,  
9   labeller = labeller(  
10    day_night = stringr::str_to_title,  
11    is_workday = codes  
12  ))  
13 )
```

TfL bike sharing trends



Coordinate Systems



Coordinate Systems

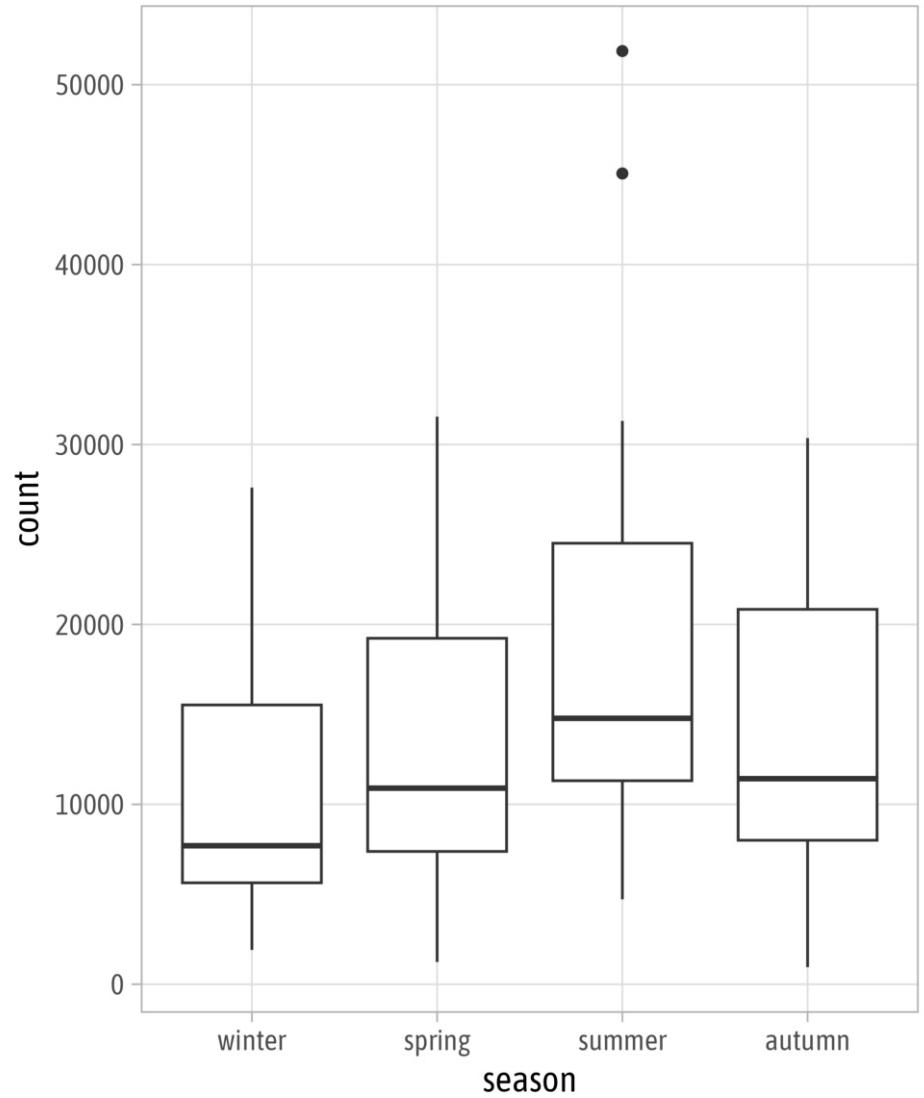
= interpret the position aesthetics

- **linear coordinate systems:** preserve the geometrical shapes
 - `coord_cartesian()`
 - `coord_fixed()`
 - `coord_flip()`
- **non-linear coordinate systems:** likely change the geometrical shapes
 - `coord_polar()`
 - `coord_map()` and `coord_sf()`
 - `coord_trans()`



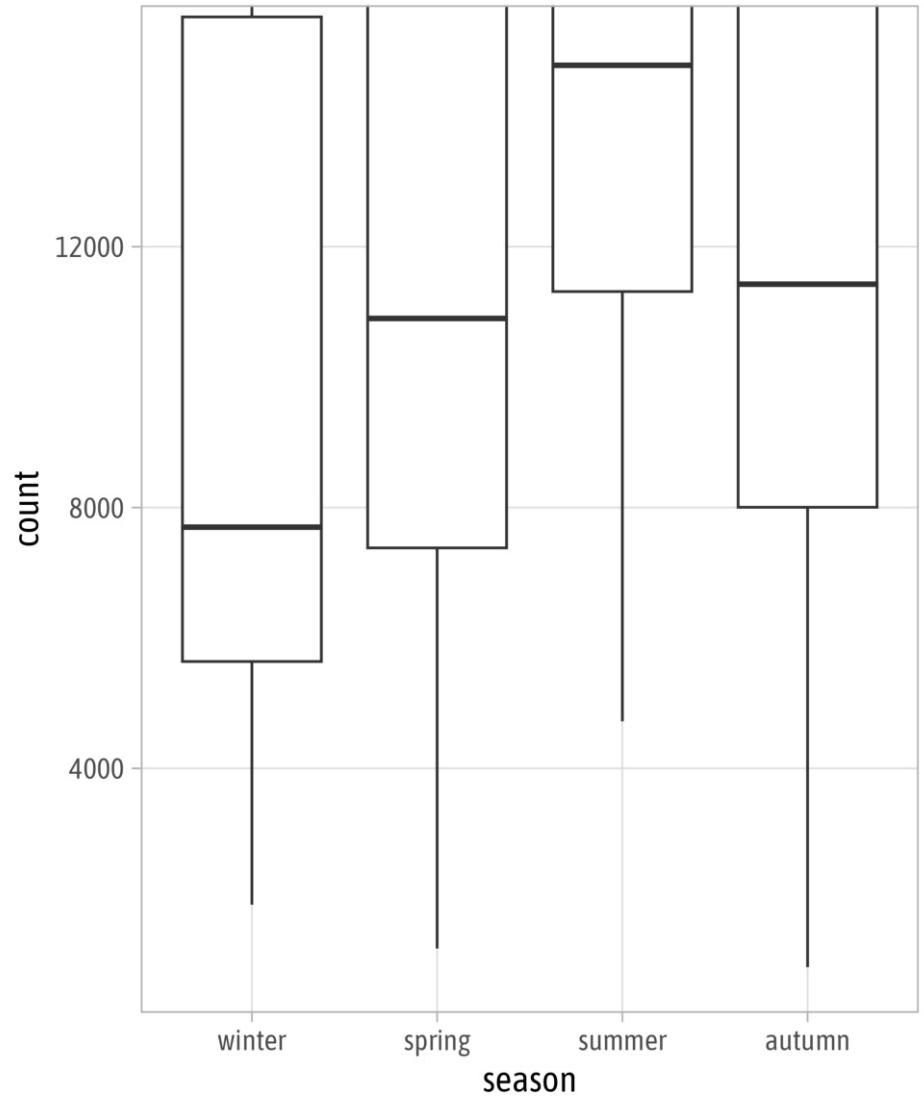
Cartesian Coordinate System

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 coord_cartesian()
```



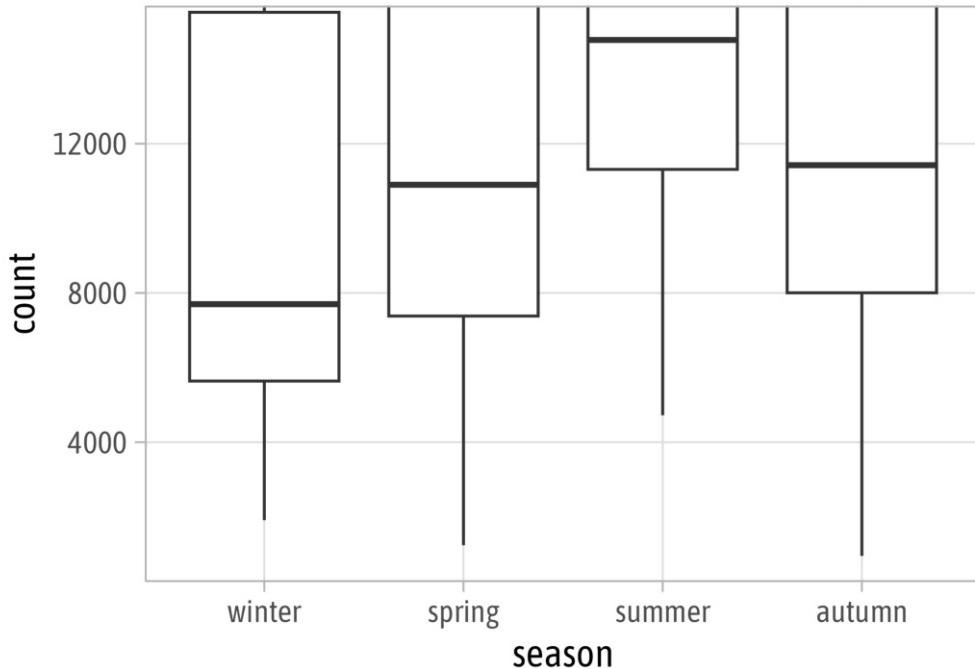
Cartesian Coordinate System

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 coord_cartesian(  
7   ylim = c(NA, 15000)  
8 )
```

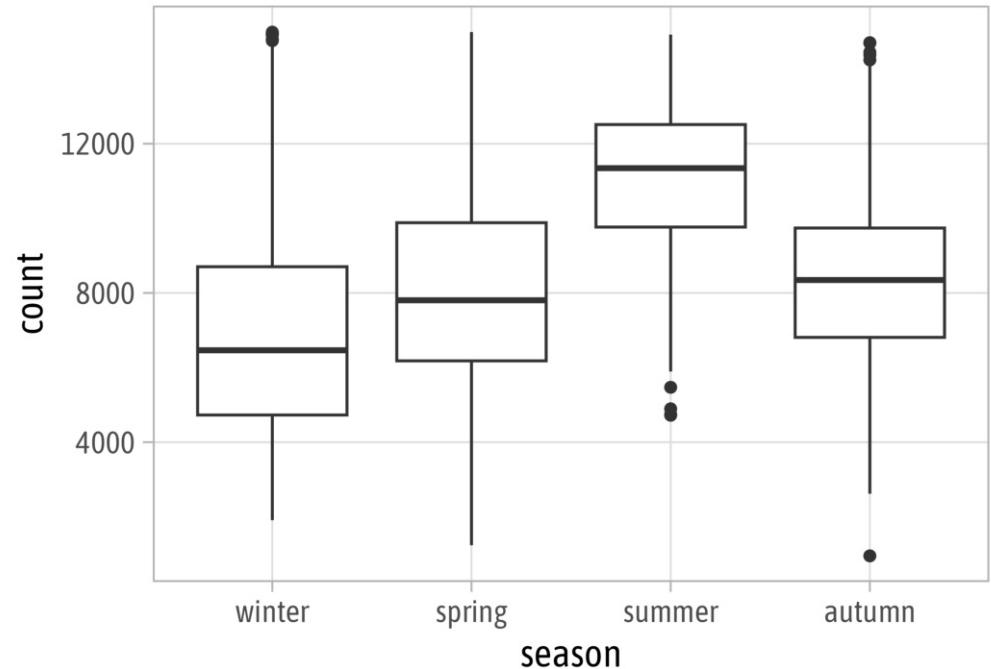


Changing Limits

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 coord_cartesian(  
7   ylim = c(NA, 15000)  
8 )
```

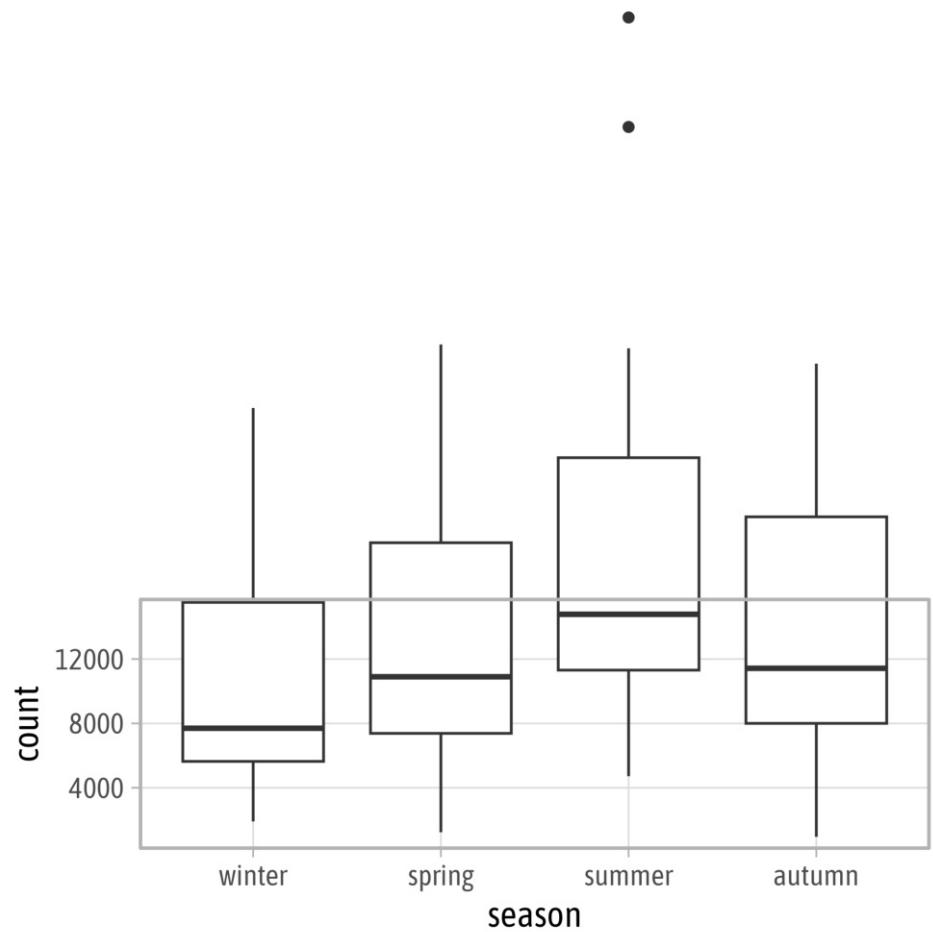


```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 scale_y_continuous(  
7   limits = c(NA, 15000)  
8 )
```



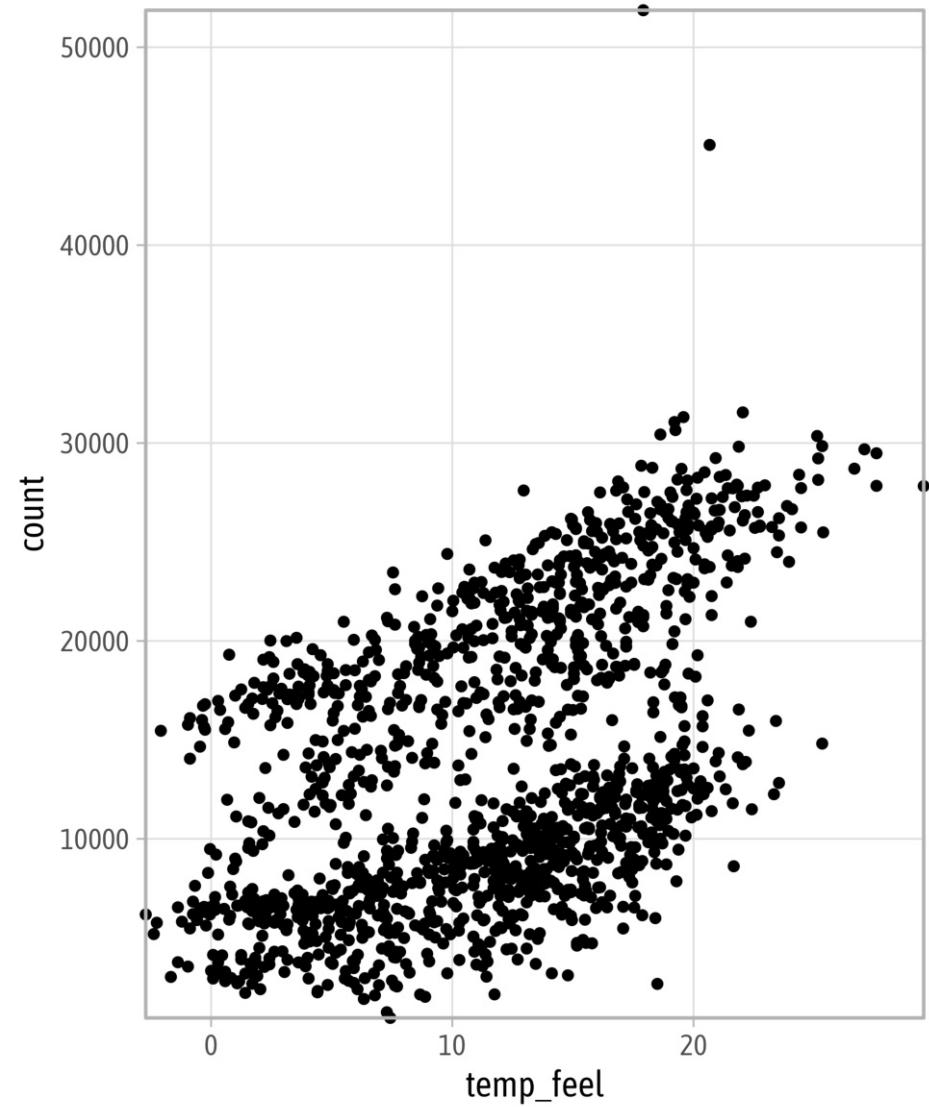
Clipping

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 coord_cartesian(  
7   ylim = c(NA, 15000),  
8   clip = "off"  
9 ) +  
10 theme(plot.margin = margin(300, 5, 5, 5))
```



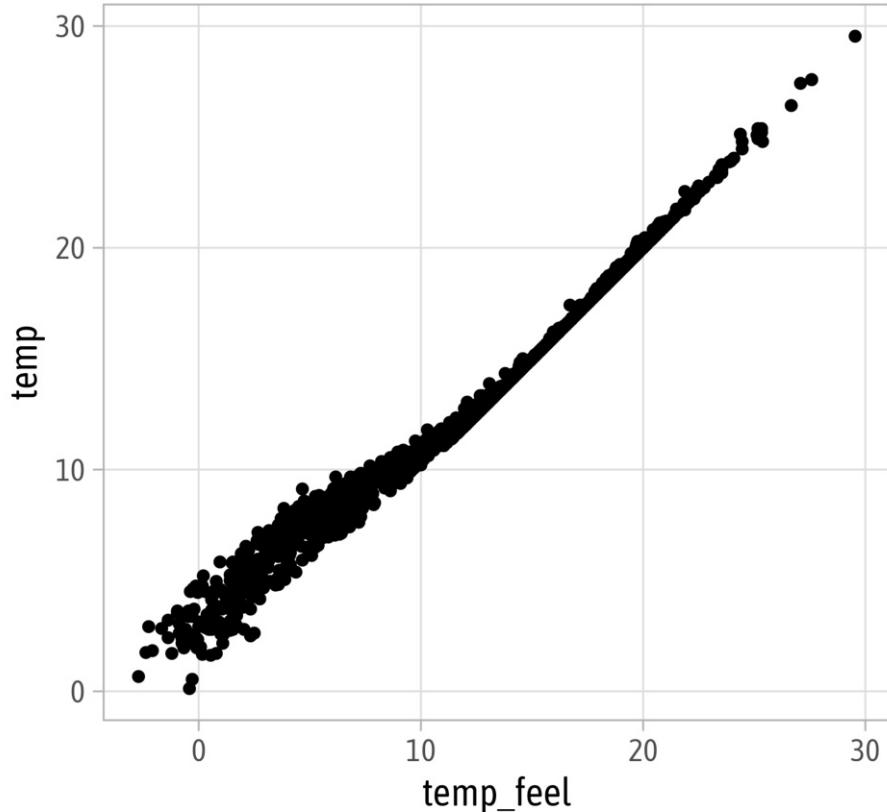
Remove All Padding

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point() +  
6 coord_cartesian(  
7   expand = FALSE,  
8   clip = "off"  
9 )
```

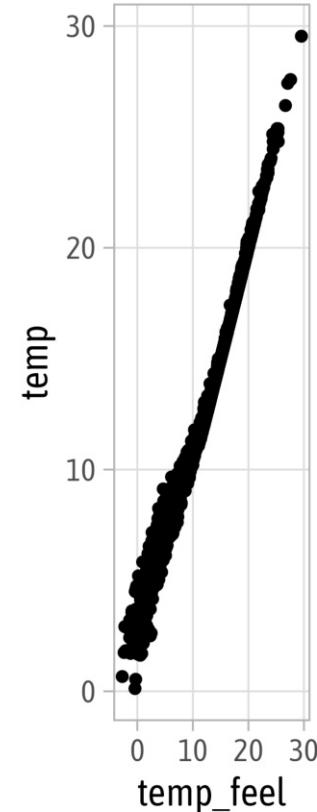


Fixed Coordinate System

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

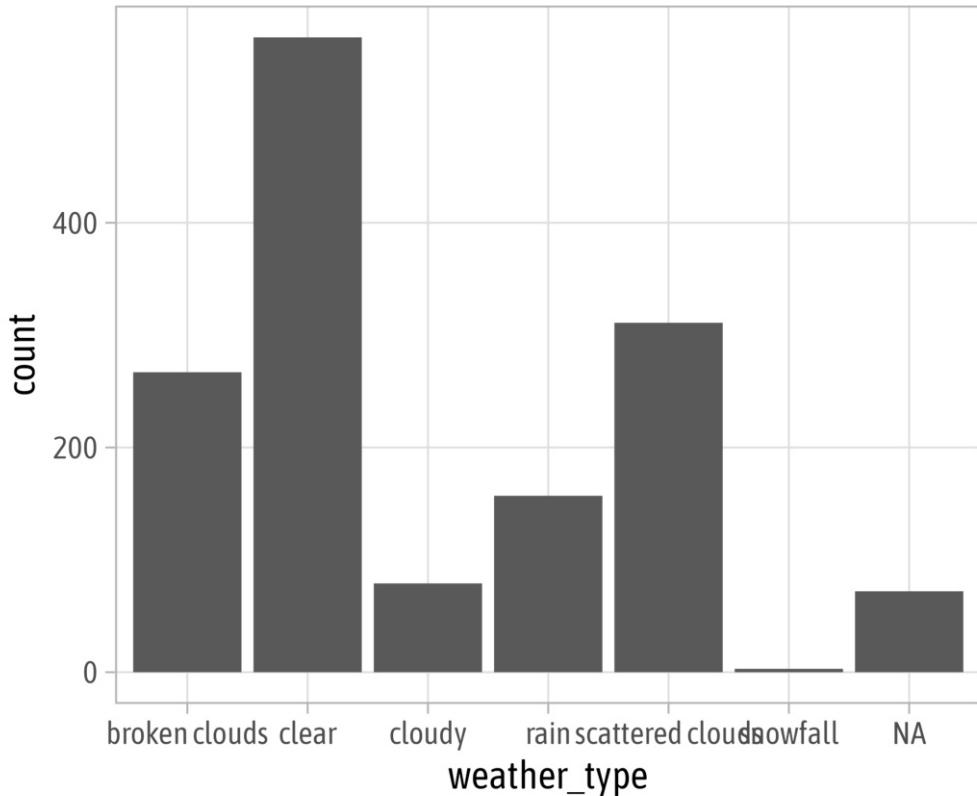


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = temp)  
4 ) +  
5   geom_point() +  
6   coord_fixed(ratio = 4)
```

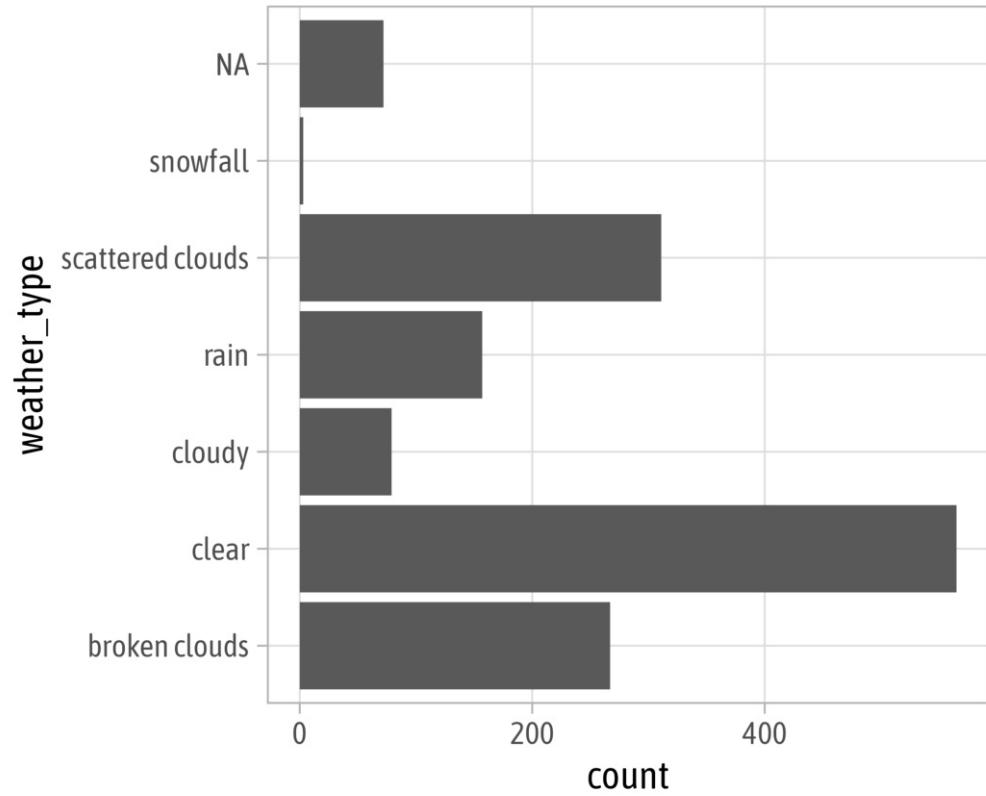


Flipped Coordinate System

```
1 ggplot(  
2   bikes,  
3   aes(x = weather_type)  
4 ) +  
5 geom_bar() +  
6 coord_cartesian()
```

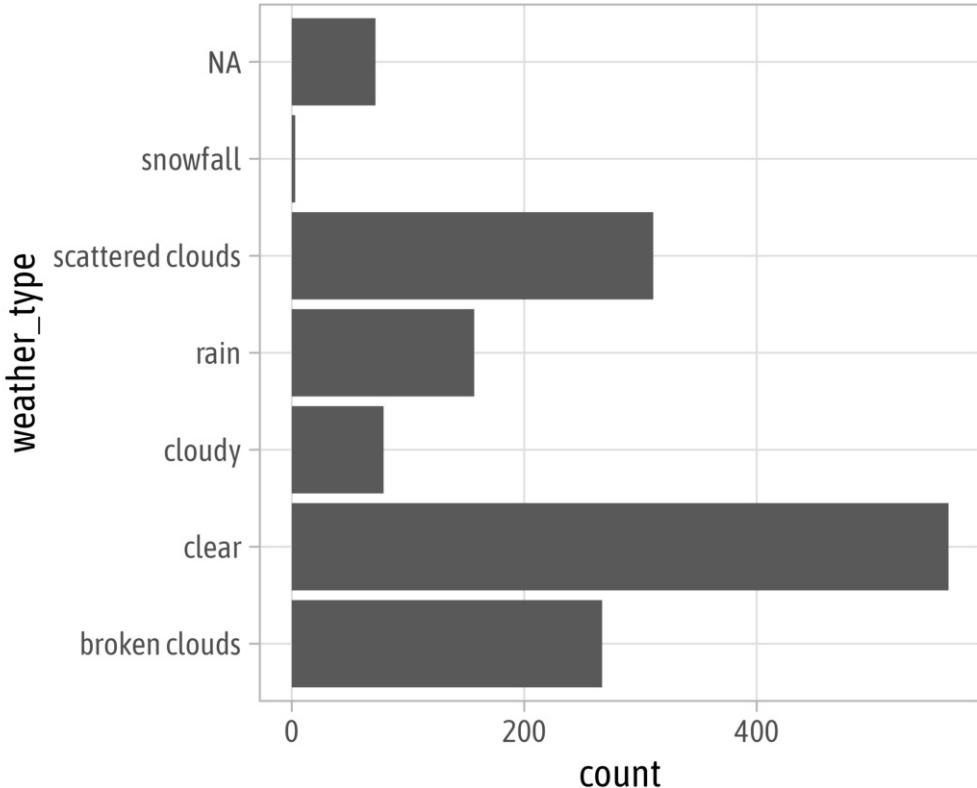


```
1 ggplot(  
2   bikes,  
3   aes(x = weather_type)  
4 ) +  
5 geom_bar() +  
6 coord_flip()
```

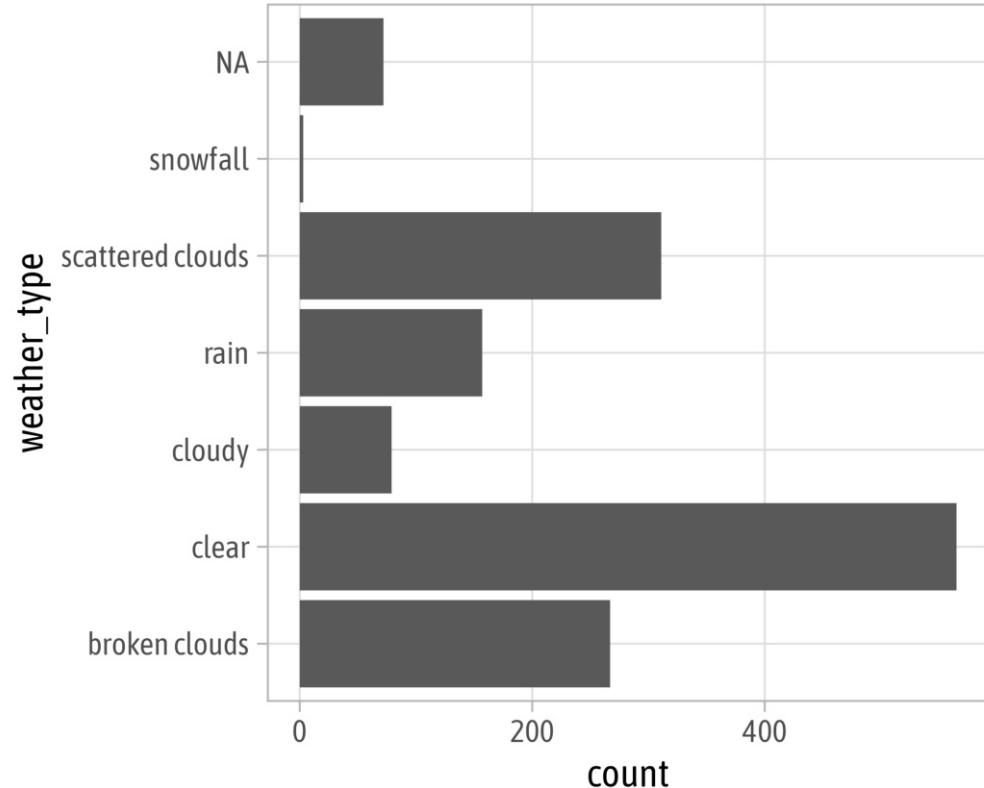


Flipped Coordinate System

```
1 ggplot(  
2   bikes,  
3   aes(y = weather_type)  
4 ) +  
5 geom_bar() +  
6 coord_cartesian()
```

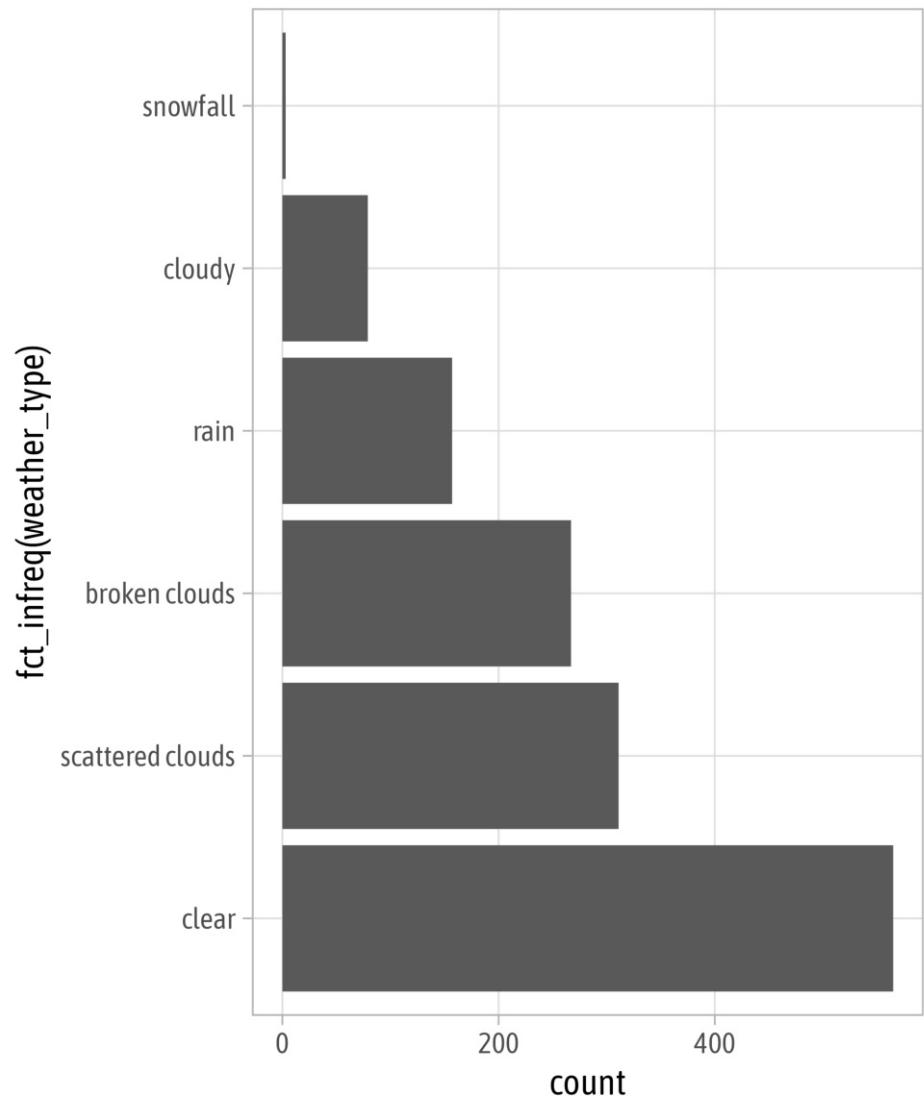


```
1 ggplot(  
2   bikes,  
3   aes(x = weather_type)  
4 ) +  
5 geom_bar() +  
6 coord_flip()
```



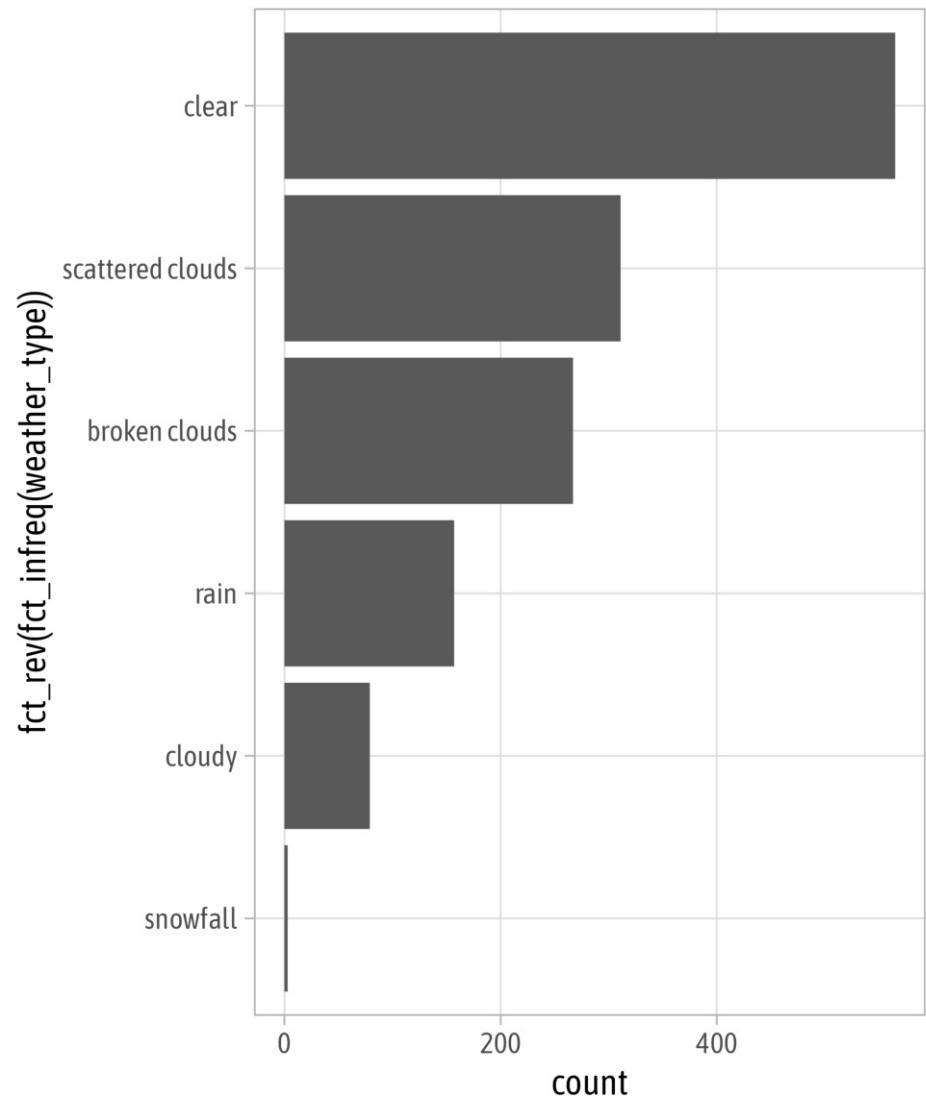
Reminder: Sort Your Bars!

```
1 library(forcats)
2
3 ggplot(
4   filter(bikes, !is.na(weather_type)),
5   aes(y = fct_infreq(weather_type)))
6 ) +
7 geom_bar()
```



Reminder: Sort Your Bars!

```
1 library(forcats)
2
3 ggplot(
4   filter(bikes, !is.na(weather_type)),
5   aes(y = fct_rev(
6     fct_infreq(weather_type)
7   )))
8 ) +
9 geom_bar()
```



Wrap-up



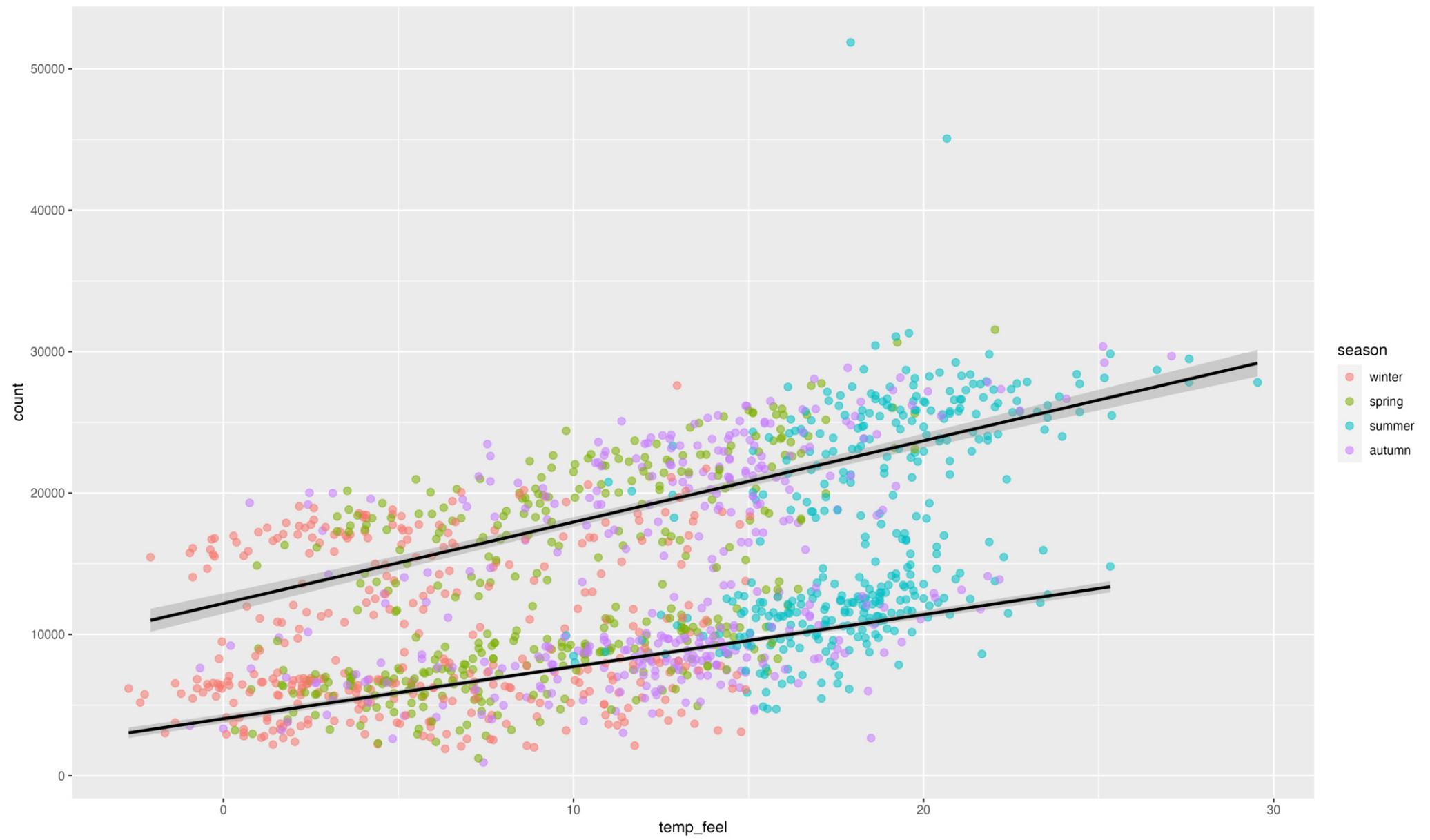
```
1 g1 <-
2   ## create scatter plot, encoded based on season
3   ggplot(bikes, aes(temp_feel, count)) +
4     geom_point(
5       aes(color = season),
6       size = 2.2, alpha = .55
7     )
8
9 g1
```





```
1 g2 <- g1 +
2   ## add a linear fitting for each time of the day
3   geom_smooth(
4     aes(group = day_night),
5     method = "lm", color = "black"
6   )
7
8 g2
```

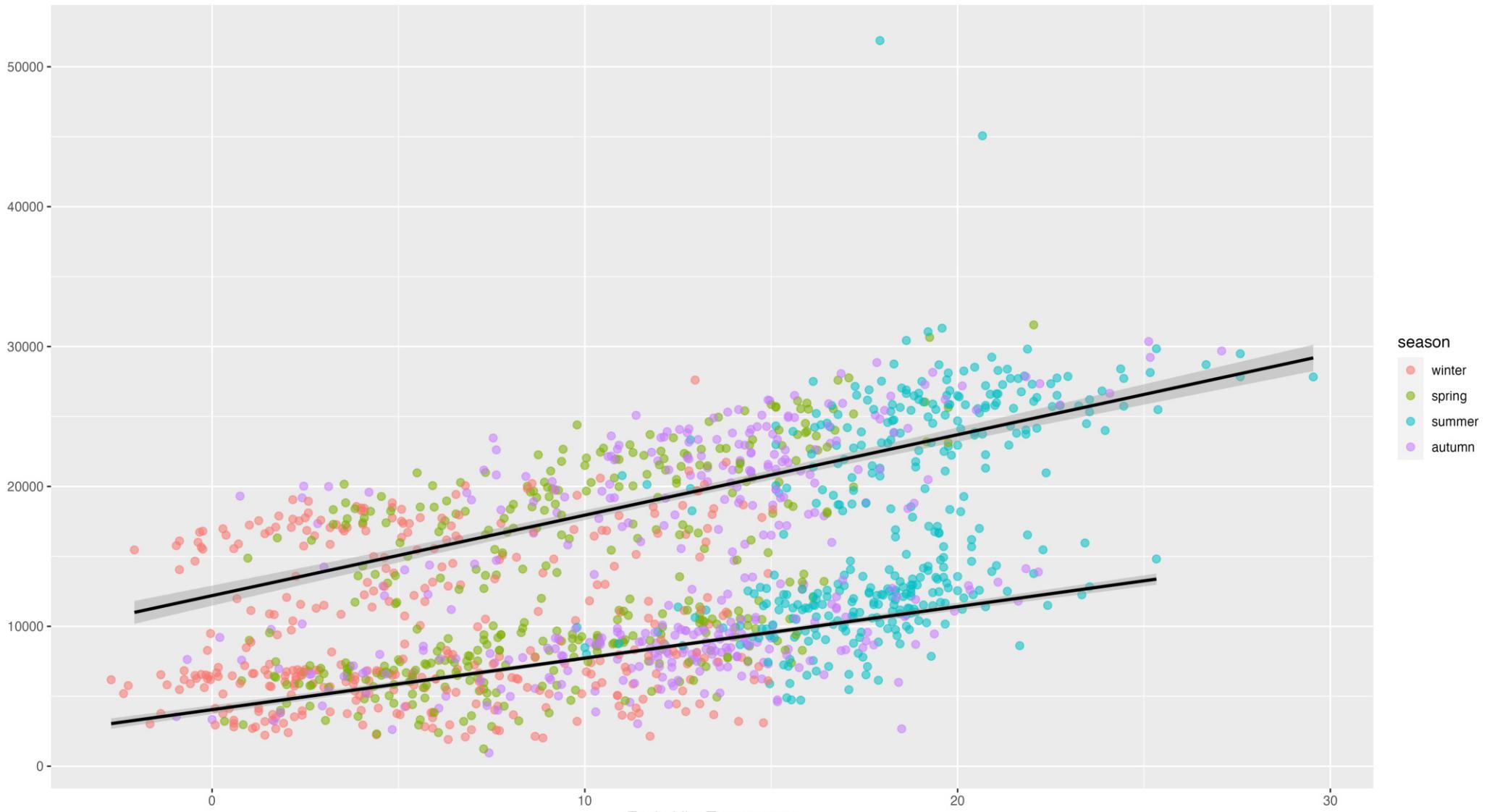




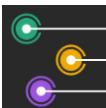
```
1 g3 <- g2 +
2 ## add titles + labels
3 labs(
4   x = "Feels-Like Temperature", y = NULL,
5   caption = "Data: Transport for London (TfL), Jan 2015–Dec 2016",
6   title = "Reported TfL bike rents versus feels-like temperature in London, 2015–2016"
7 )
8
9 g3
```



Reported TfL bike rents versus feels-like temperature in London, 2015–2016



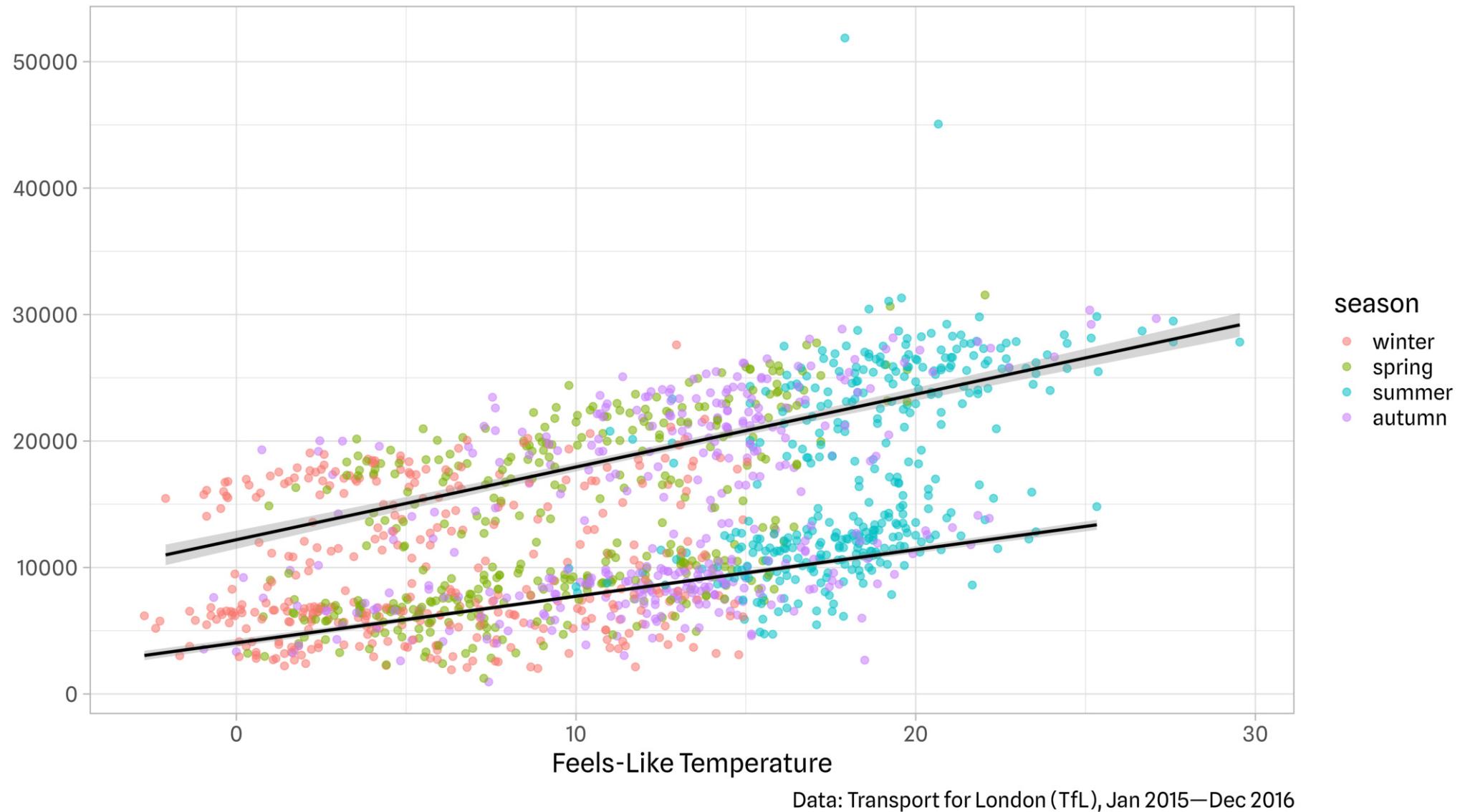
Data: Transport for London (TfL), Jan 2015–Dec 2016



```
1 g4 <- g3 +
2   theme_light(base_size = 18, base_family = "Spline Sans")
3
4 g4
```



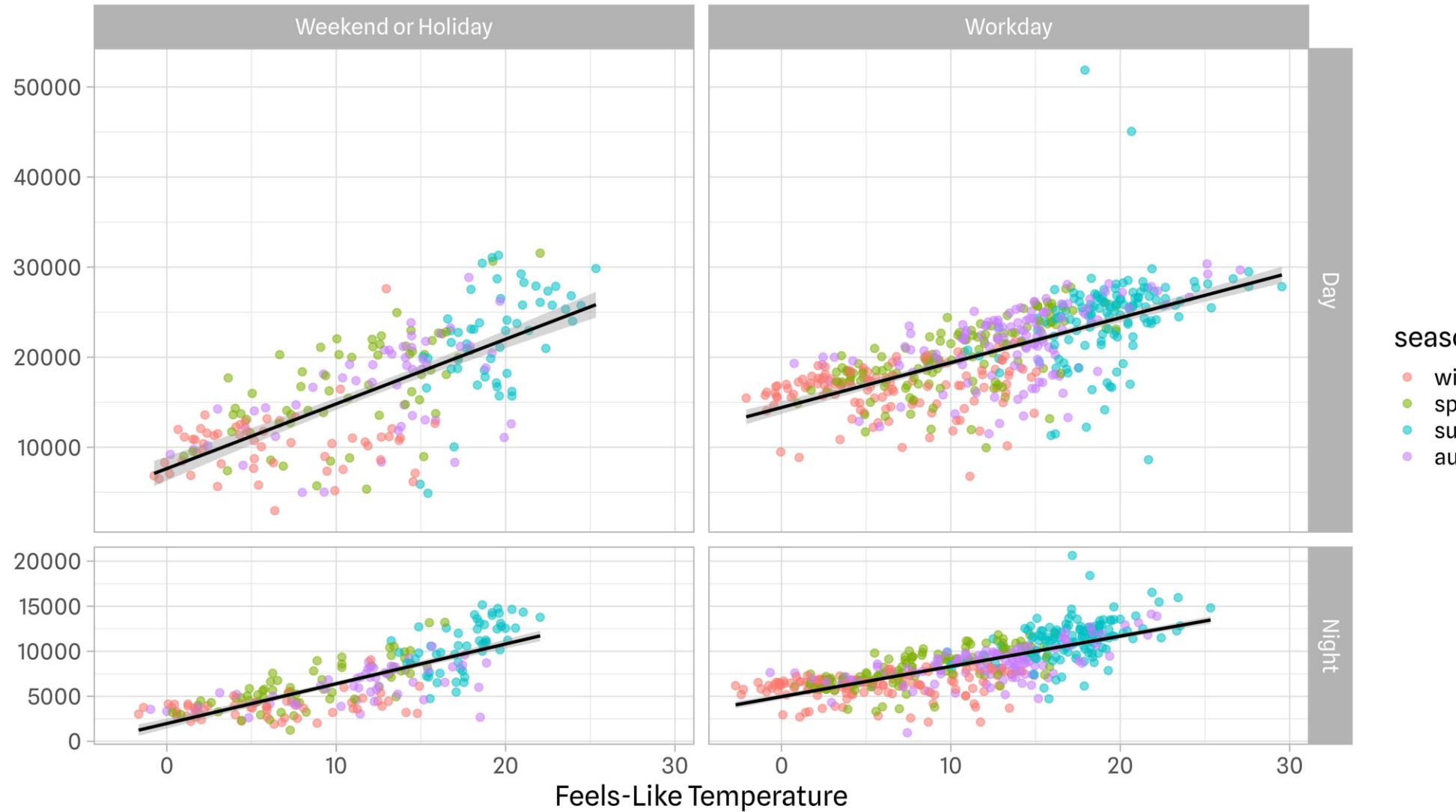
Reported TfL bike rents versus feels-like temperature in London, 2015–2016



```
1 codes <- c(
2   `TRUE` = "Workday",
3   `FALSE` = "Weekend or Holiday"
4 )
5
6 g5 <- g4 +
7   facet_grid(
8     day_night ~ is_workday,
9     scales = "free_y", space = "free_y",
10    labeller = labeller(
11      day_night = stringr::str_to_title,
12      is_workday = codes
13    )
14  )
15
16 g5
```



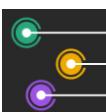
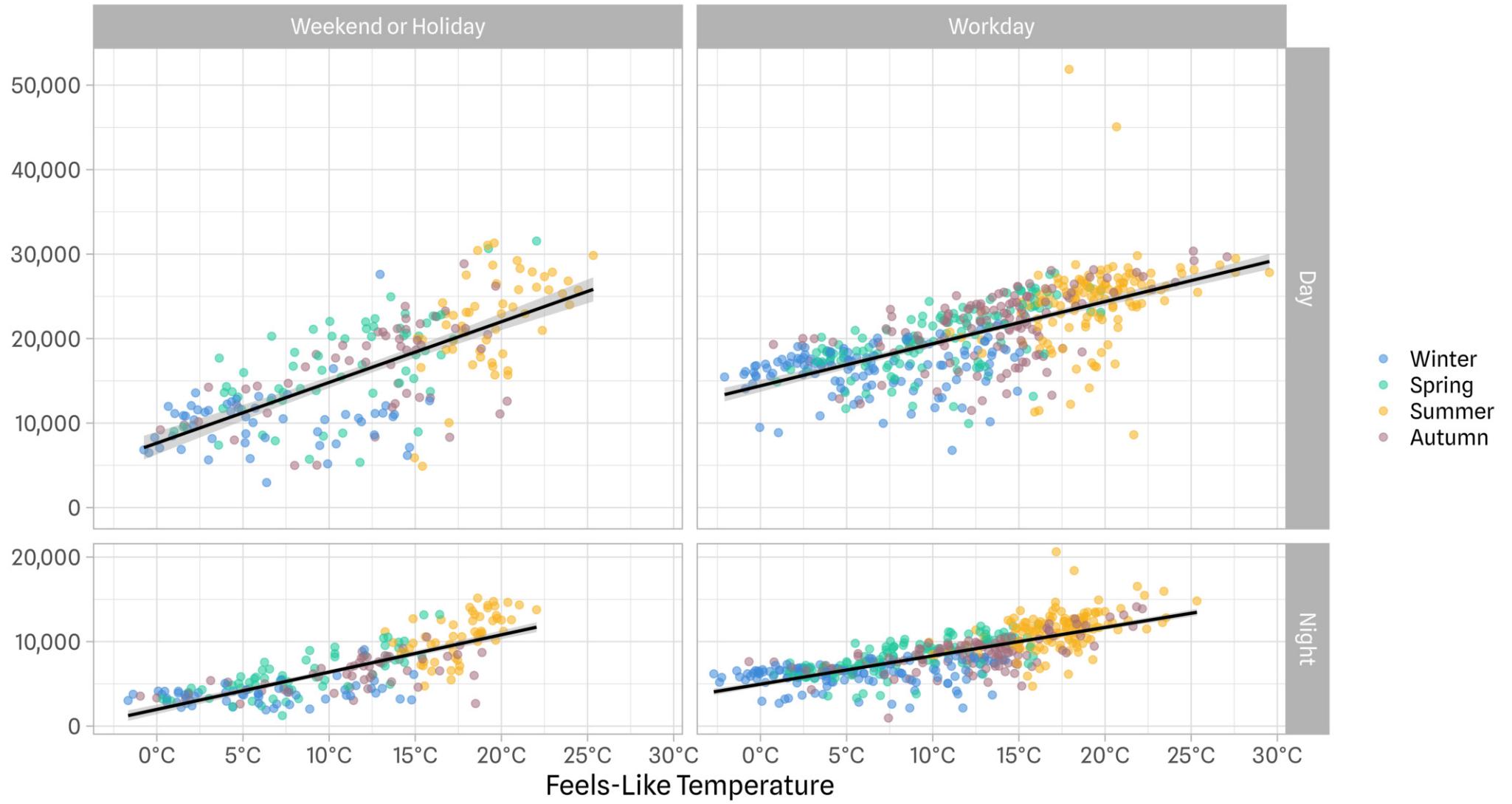
Reported TfL bike rents versus feels-like temperature in London, 2015–2016



```
1 g6 <- g5 +
2   ## adjust labels x-axis
3   scale_x_continuous(
4     expand = c(mult = 0, add = 1),
5     breaks = 0:6*5, labels = function(x) paste0(x, "°C"))
6   ) +
7   ## adjust labels y-axis
8   scale_y_continuous(
9     expand = c(mult = .05, add = 0), limits = c(0, NA),
10    breaks = 0:5*10000, labels = scales::label_comma()
11  ) +
12  ## modify colors + legend
13  scale_color_manual(
14    values = c("#3c89d9", "#1ec99b", "#F7B01B", "#a26e7c"),
15    name = NULL, labels = stringr::str_to_title
16  )
17
18 g6
```



Reported TfL bike rents versus feels-like temperature in London, 2015–2016

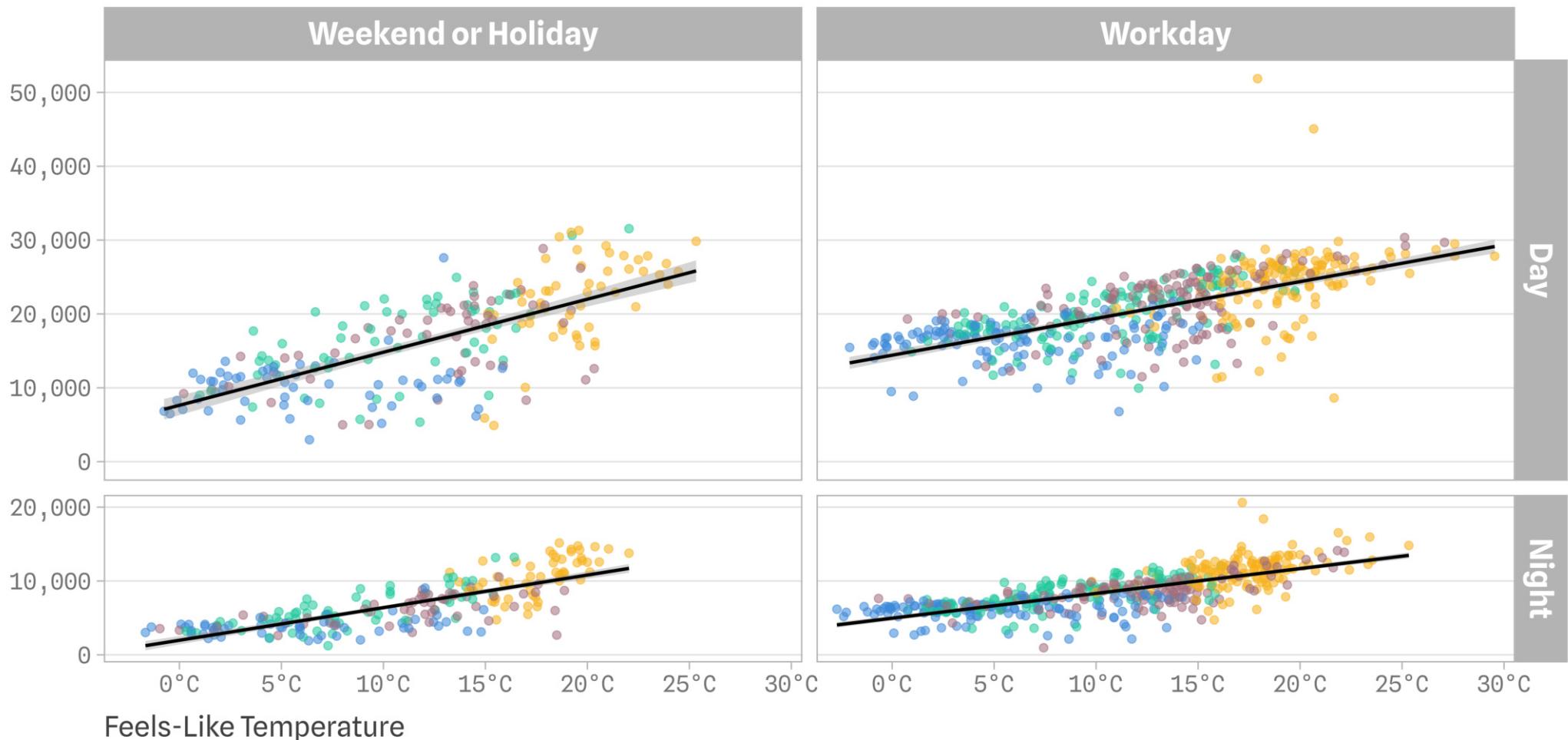


```
1 g6 +
2 ## theme modifications
3 theme(
4   plot.title.position = "plot",
5   plot.caption.position = "plot",
6   plot.title = element_text(face = "bold", size = rel(1.5)),
7   axis.text = element_text(family = "Spline Sans Mono", color = "grey45"),
8   axis.title.x = element_text(hjust = 0, margin = margin(t = 12), color = "grey25"),
9   strip.text = element_text(face = "bold", size = rel(1.1)),
10  panel.grid.major.x = element_blank(),
11  panel.grid.minor = element_blank(),
12  legend.text = element_text(size = rel(.9)),
13  legend.position = "top",
14  legend.justification = "left"
15 )
```



Reported TfL bike rents versus feels-like temperature in London, 2015–2016

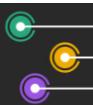
• Winter • Spring • Summer • Autumn



Data: Transport for London (TfL), Jan 2015—Dec 2016



***That's it Folks...
— Thank you! —***



Appendix

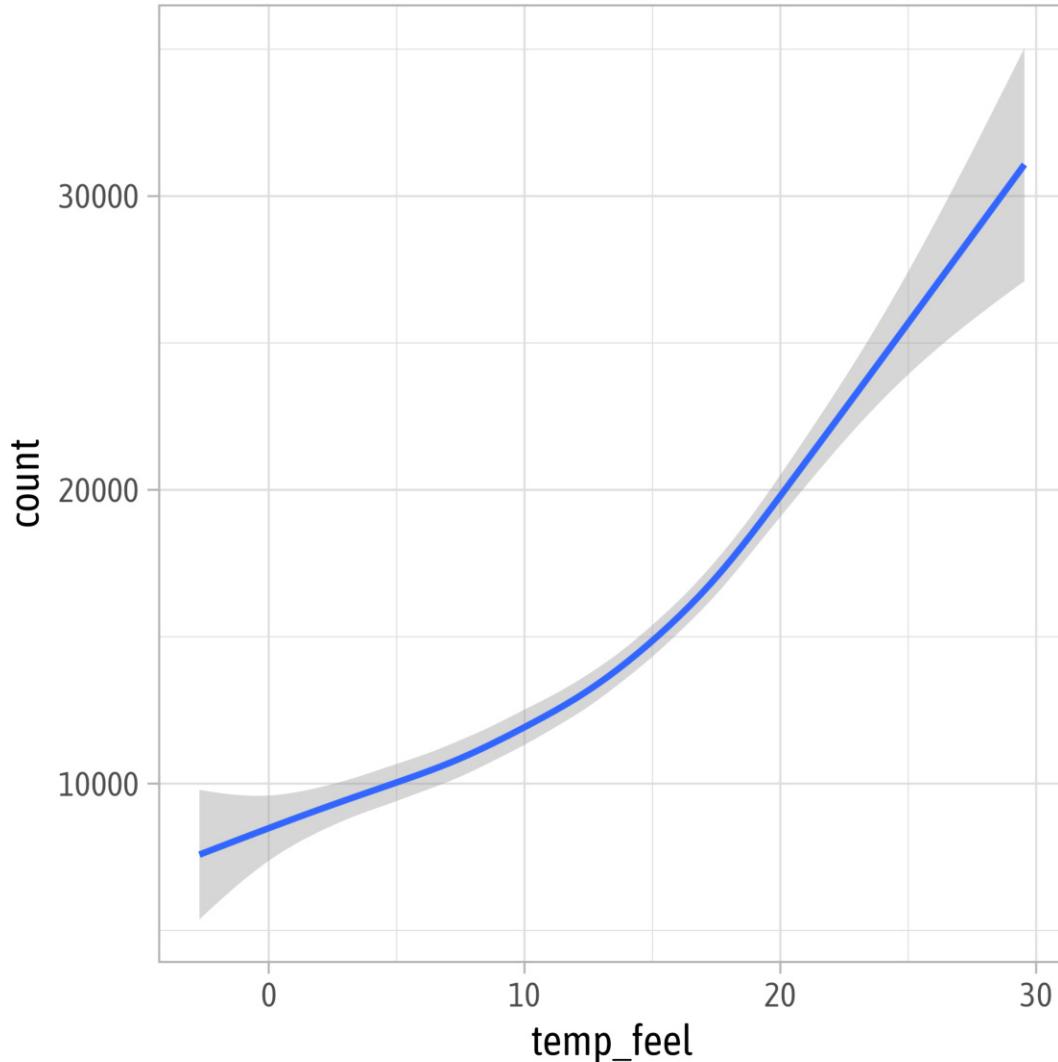


Statistical Layers

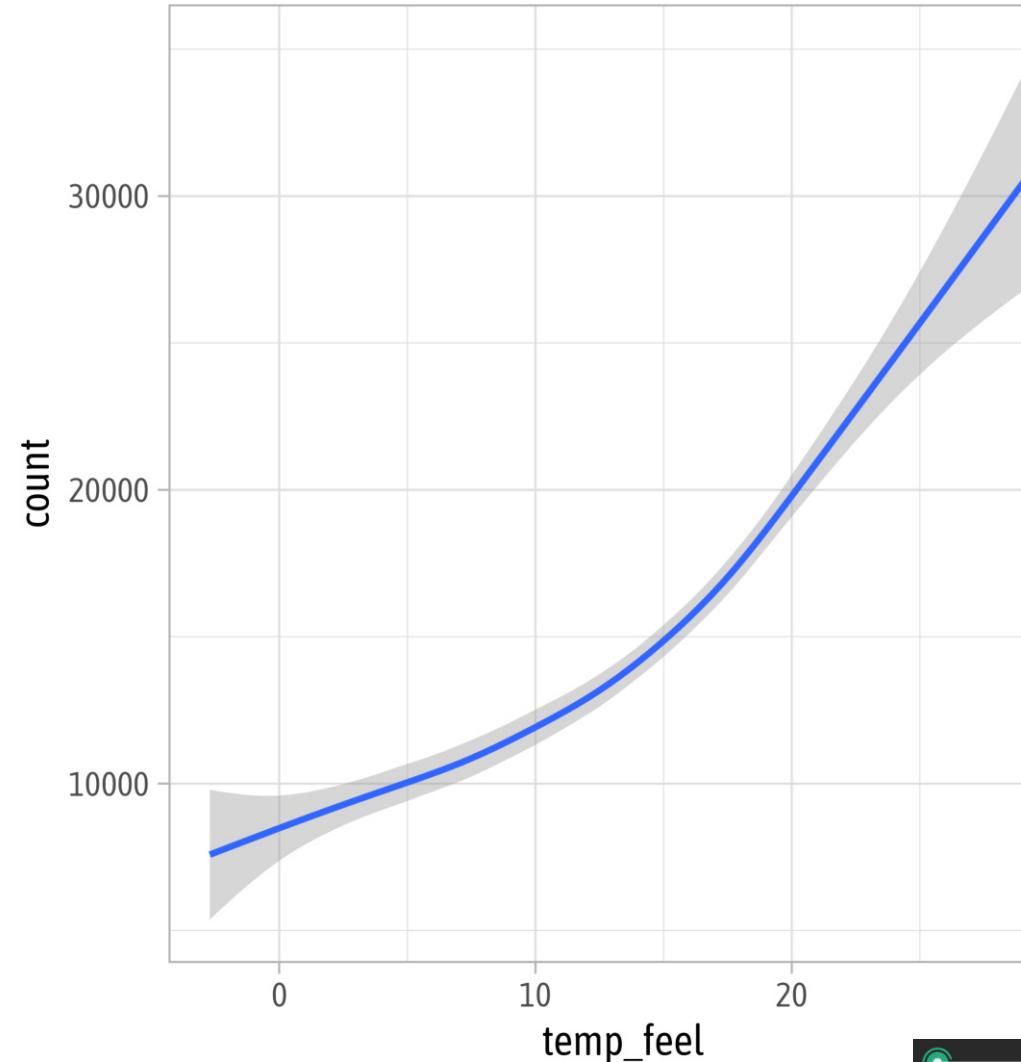


geom_*() and stat_*()

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

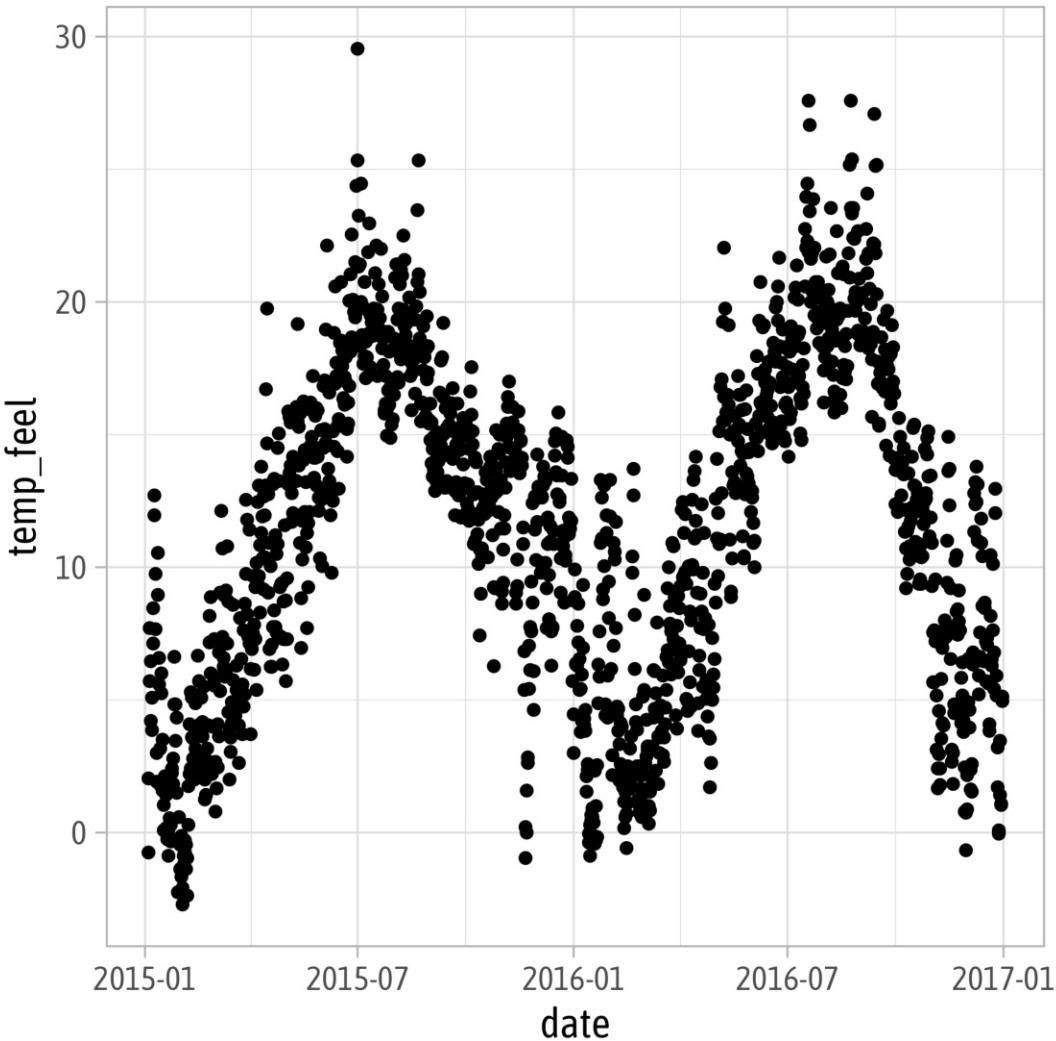


```
1 ggplot(bikes, aes(x = temp_feel, y = count)) +  
2   stat_smooth(geom = "smooth")
```

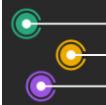
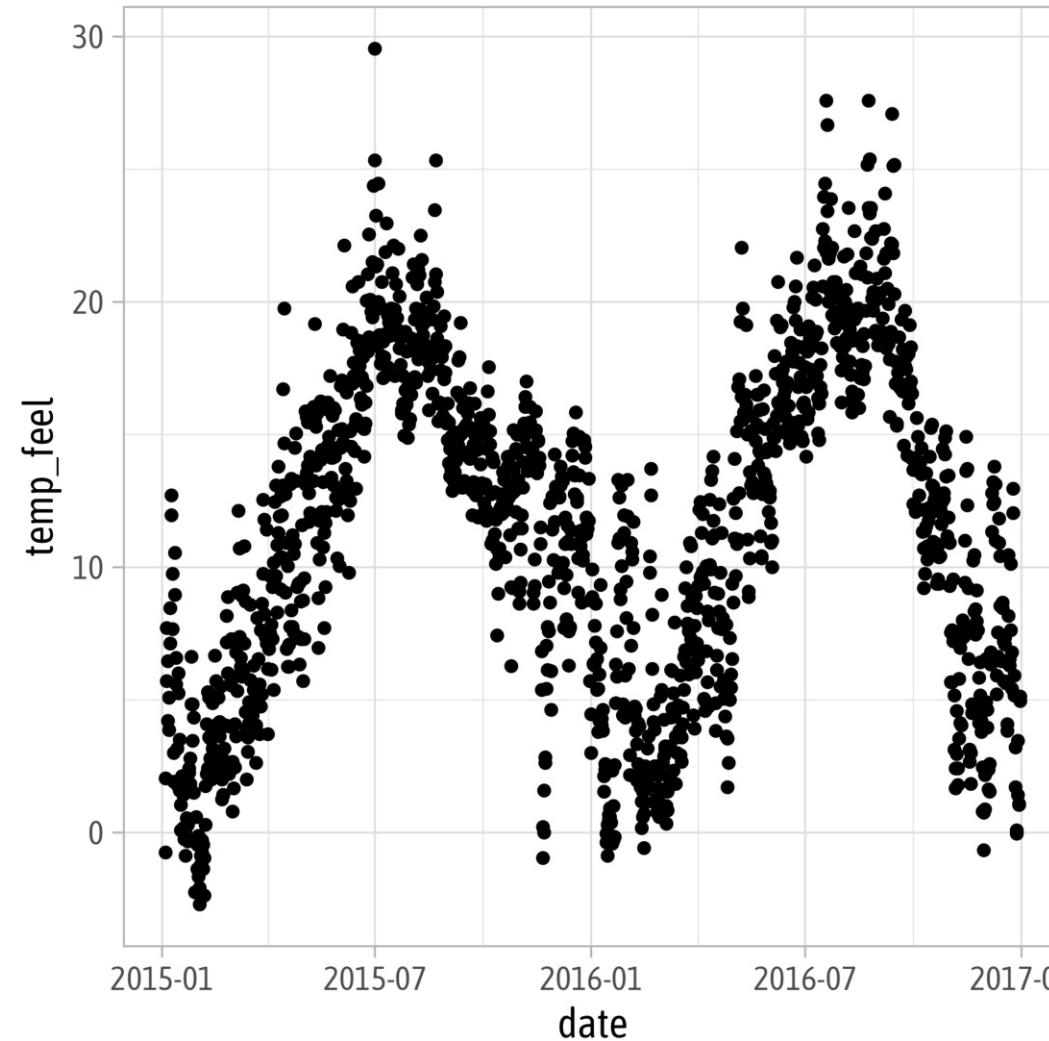


geom_*() and stat_*()

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

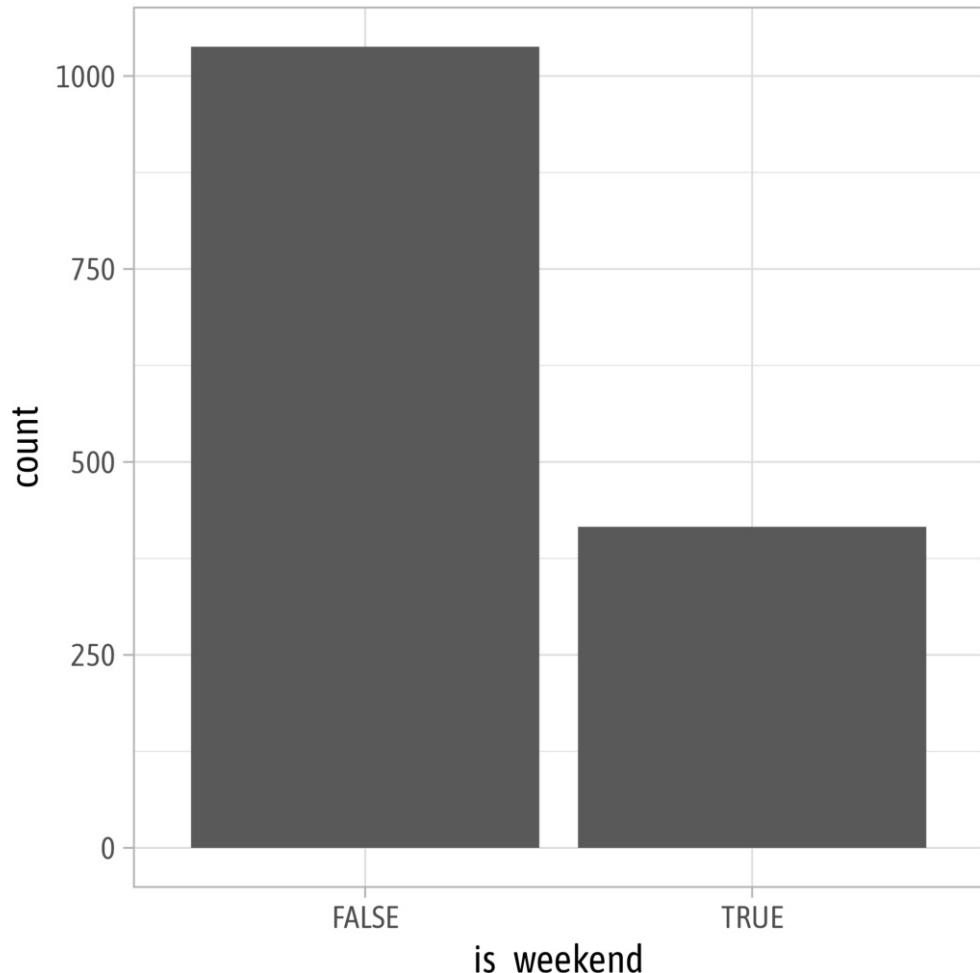


```
1 ggplot(bikes, aes(x = date, y = temp_feel)) +  
2   stat_identity(geom = "point")
```

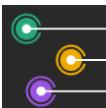
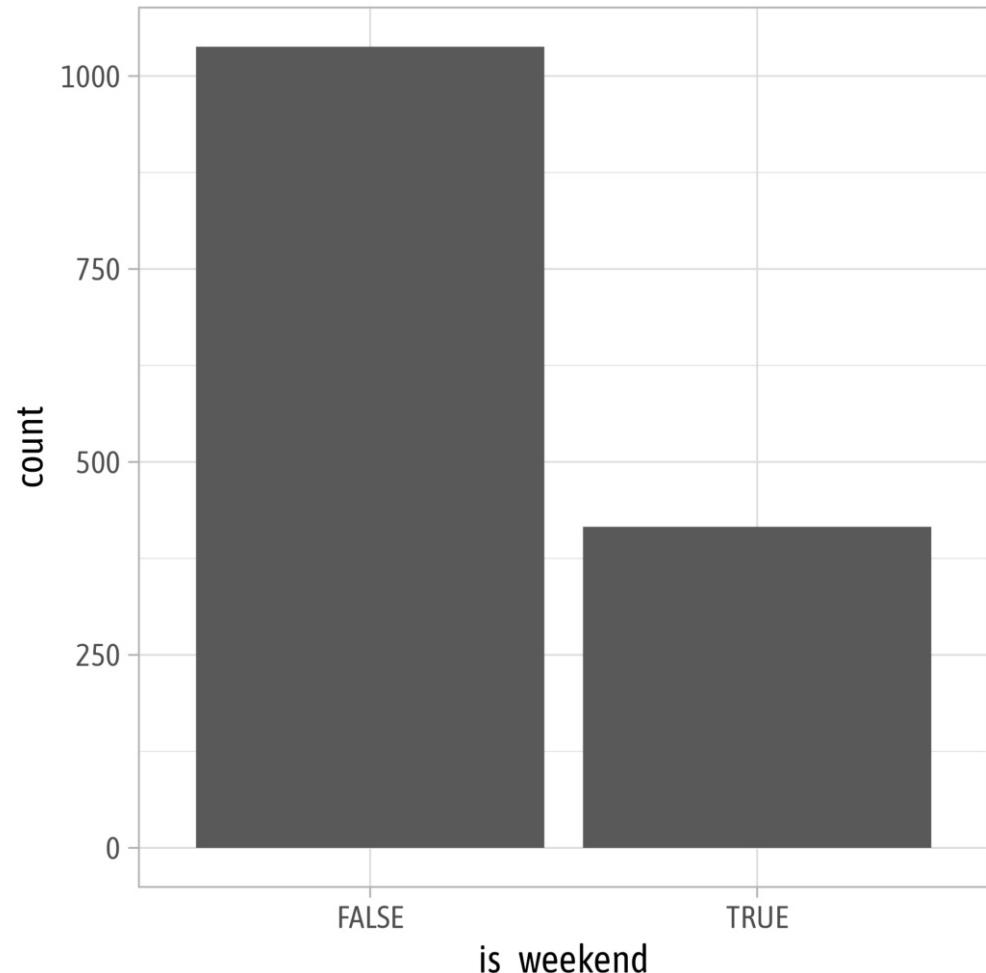


geom_*() and stat_*()

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

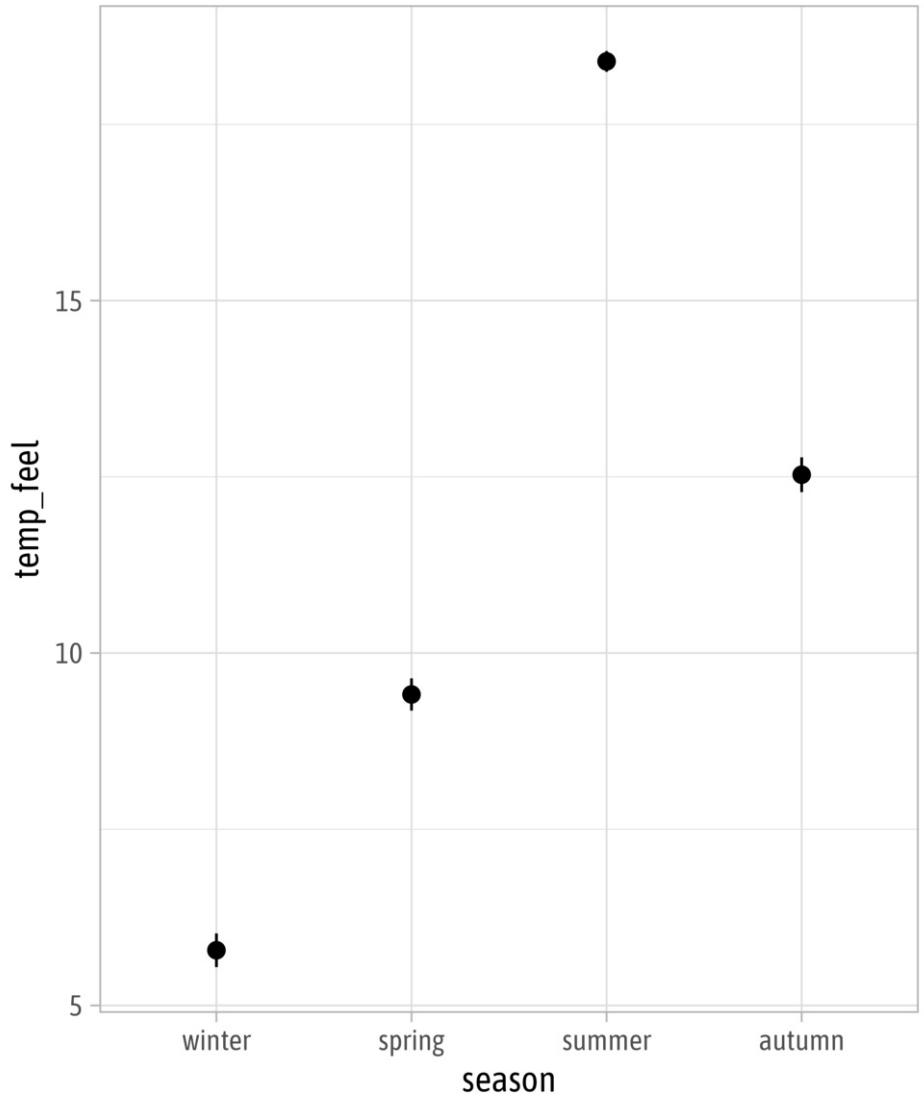


```
1 ggplot(bikes, aes(x = is_weekend)) +  
2   stat_count(geom = "bar")
```



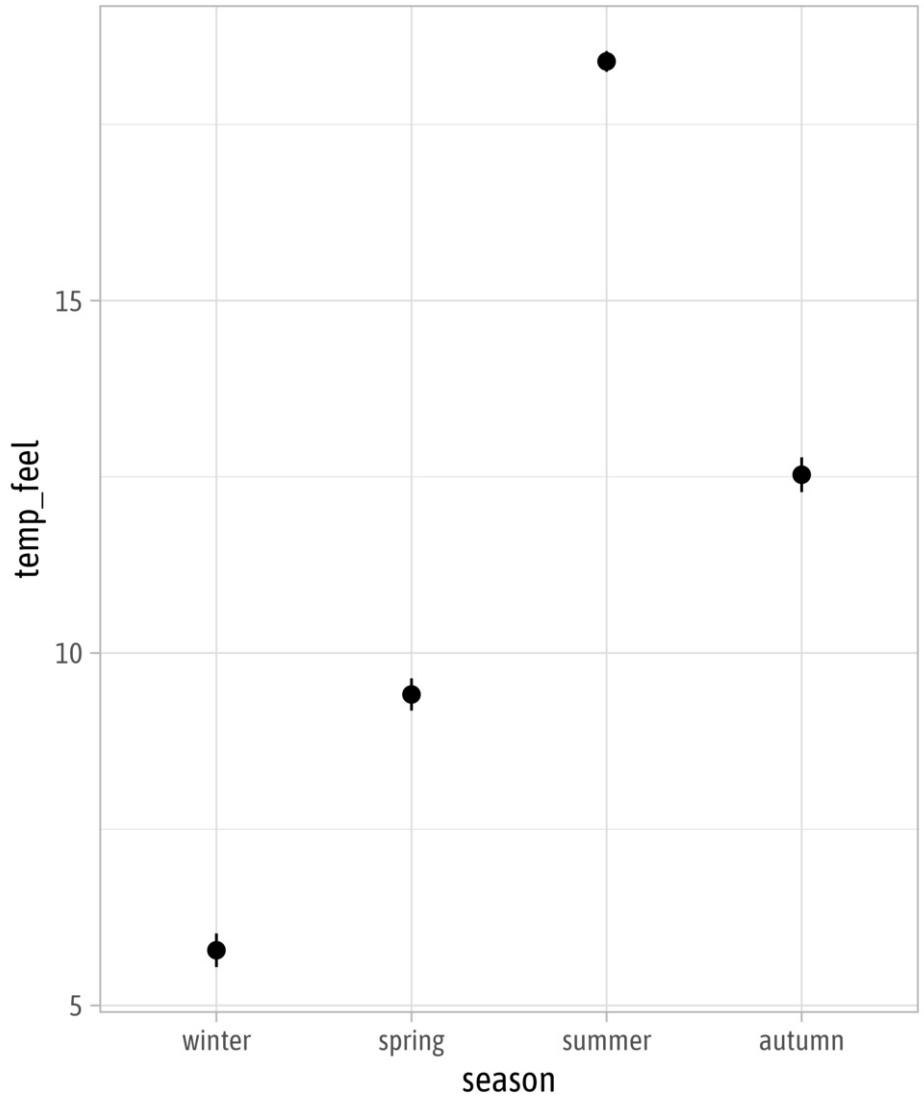
Statistical Summaries

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



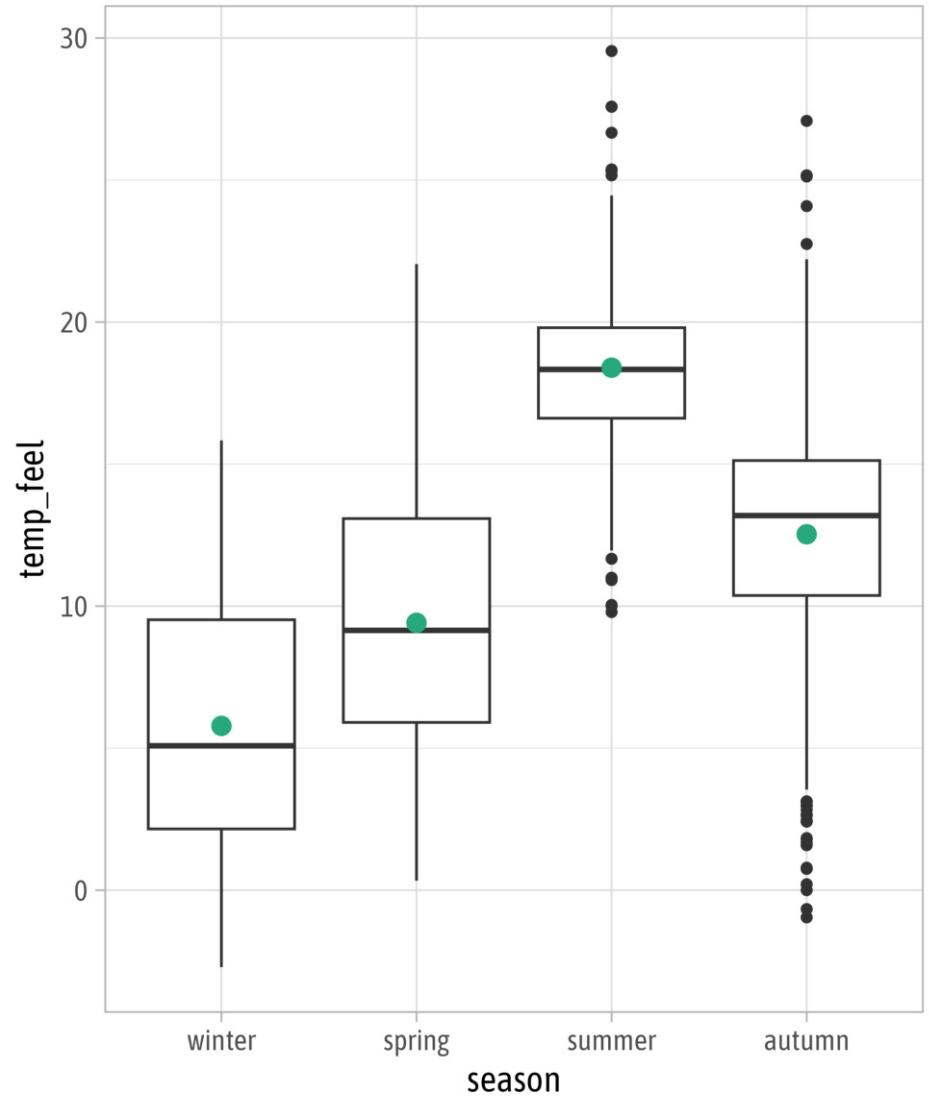
Statistical Summaries

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



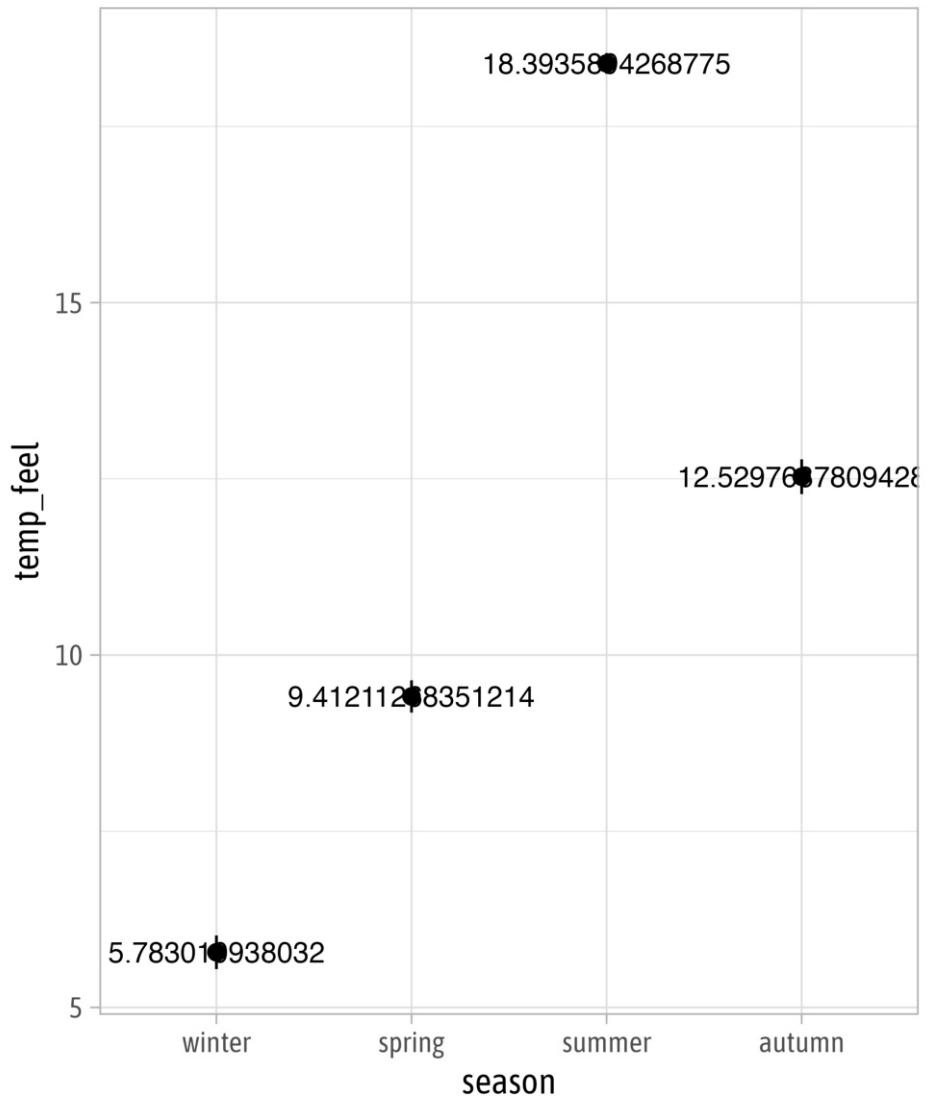
Statistical Summaries

```
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 )
```



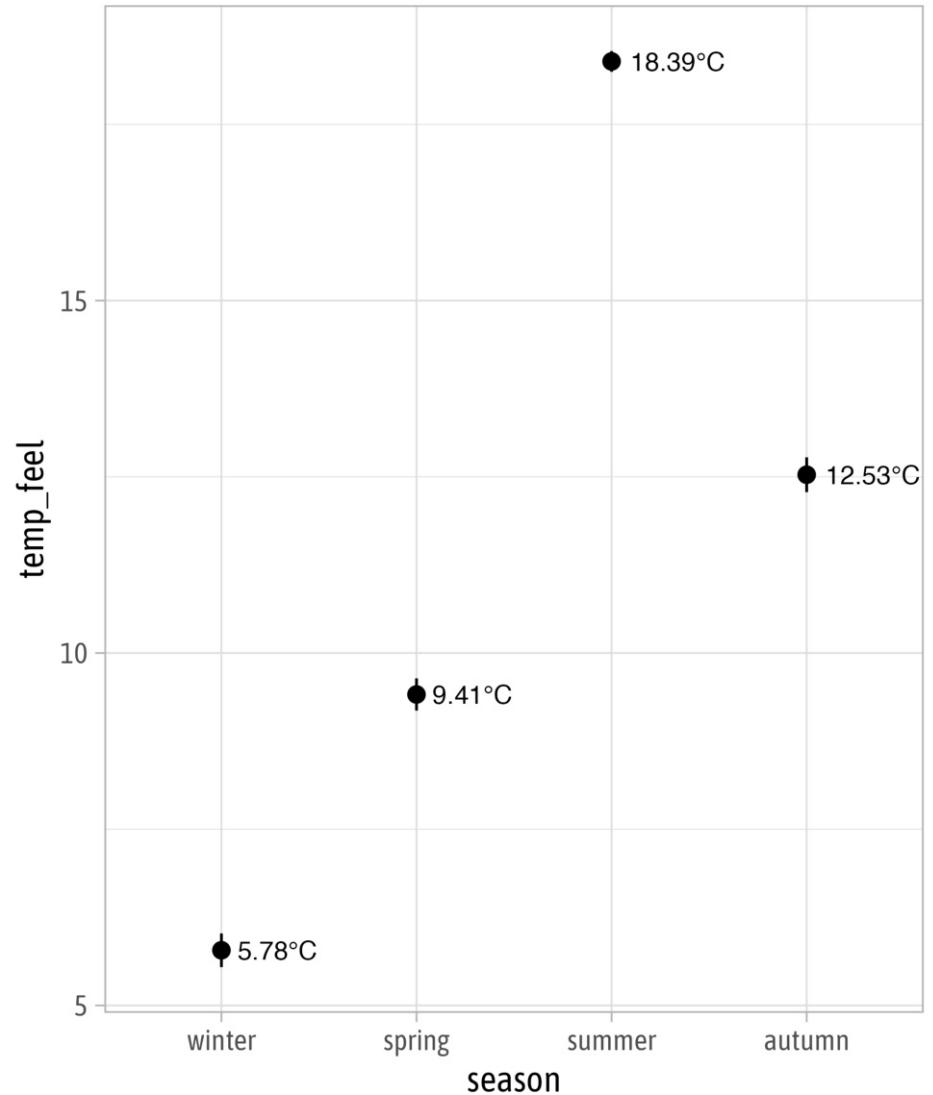
Statistical Summaries

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = temp_feel)  
4 ) +  
5 stat_summary() +  
6 stat_summary(  
7   fun = mean,  
8   geom = "text",  
9   aes(label = after_stat(y))  
10 )
```

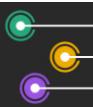


Statistical Summaries

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = temp_feel)  
4 ) +  
5 stat_summary() +  
6 stat_summary(  
7   fun = mean,  
8   geom = "text",  
9   aes(label = after_stat(  
10    paste0(round(y, 2), "°C"))  
11 ),  
12 hjust = -.2,  
13 size = 3.5  
14 )
```



Aspect Ratios



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}

