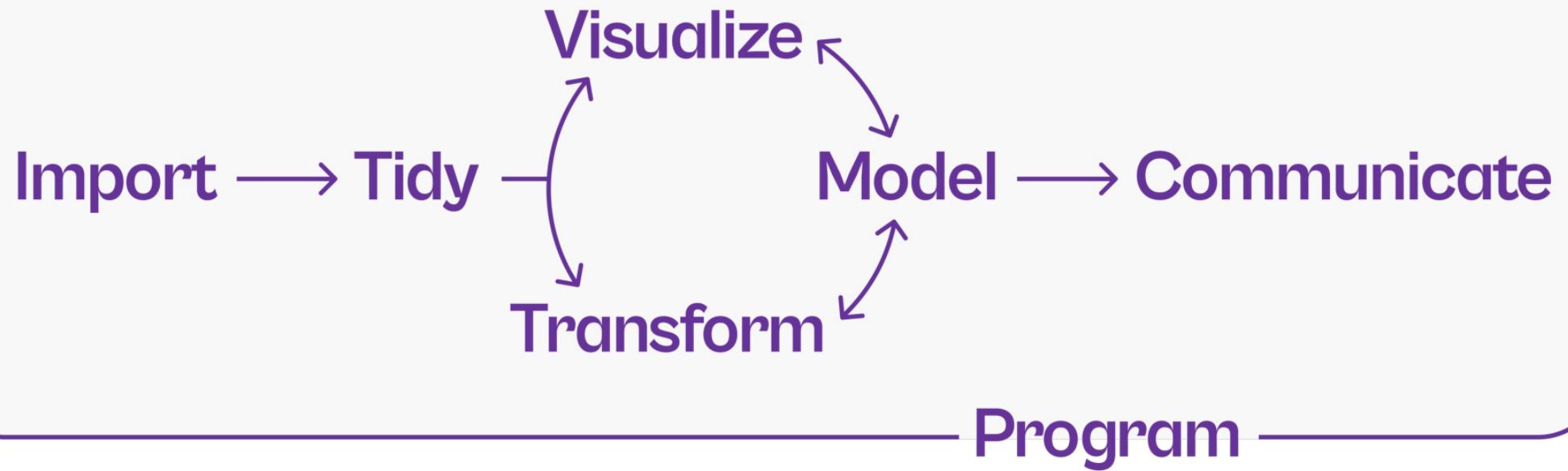


Reproducible Data Analysis with R

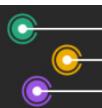
— Data Visualization with *{ggplot2}* —

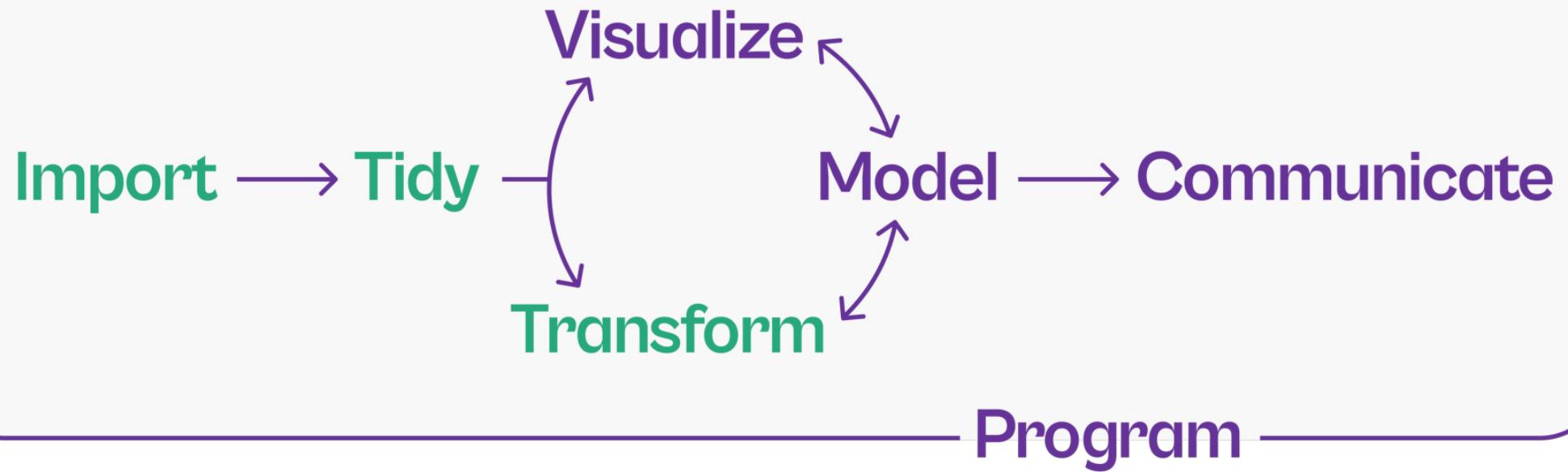
Cédric Scherer // R Course TU Dresden // Feb 27-Mar 3, 2023





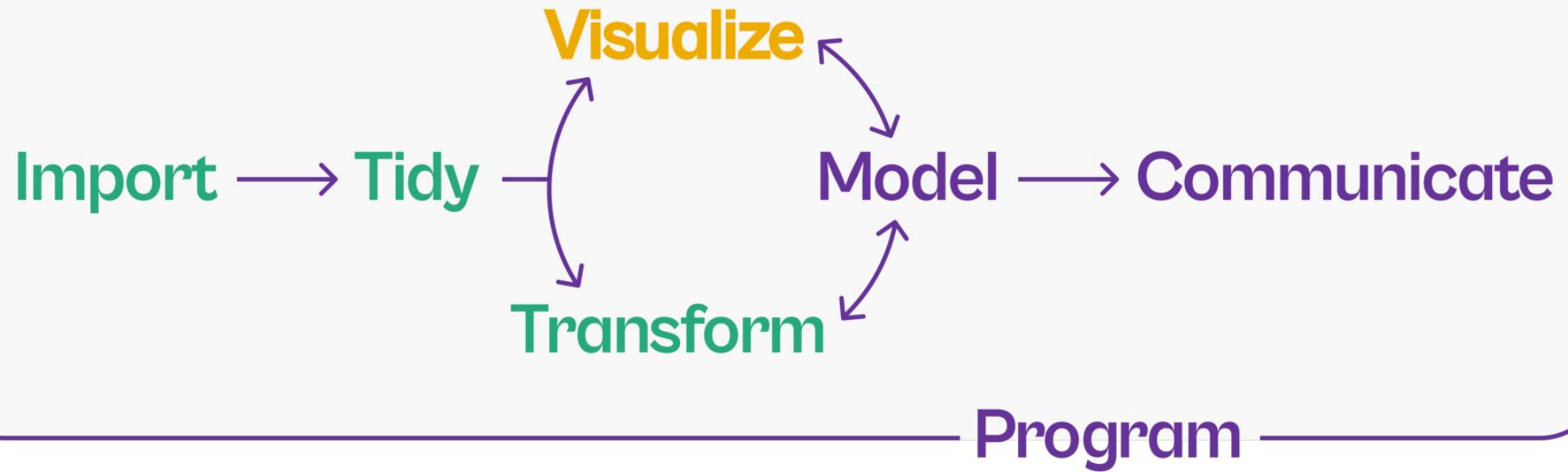
The data science workflow, modified from "R for Data Science"





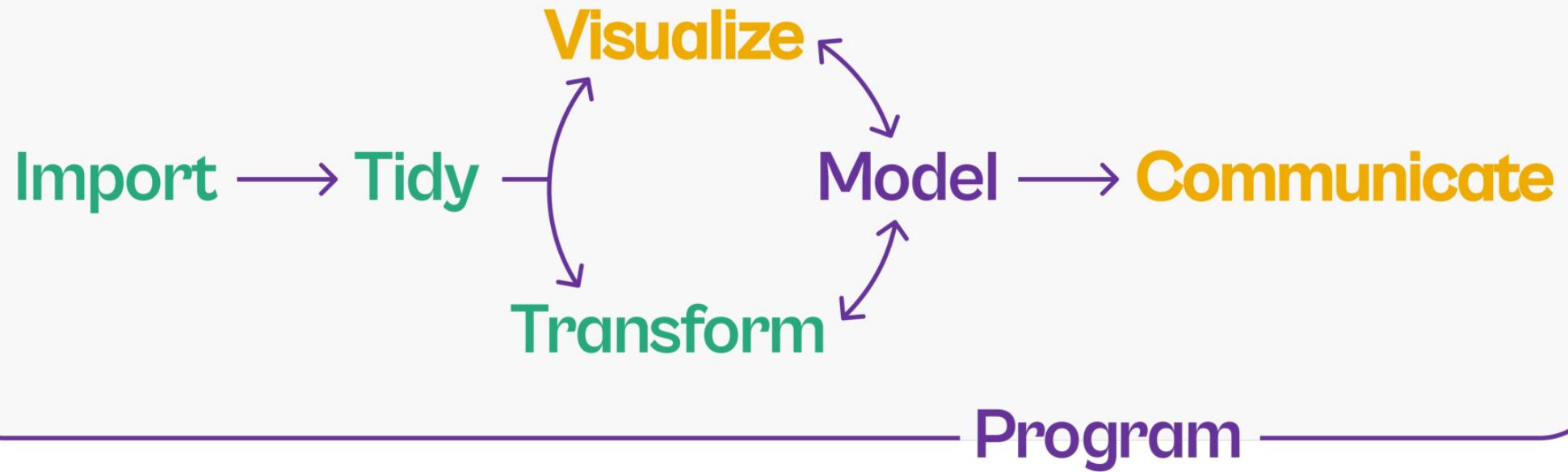
The data science workflow, modified from "R for Data Science"





The data science workflow, modified from "R for Data Science"

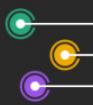


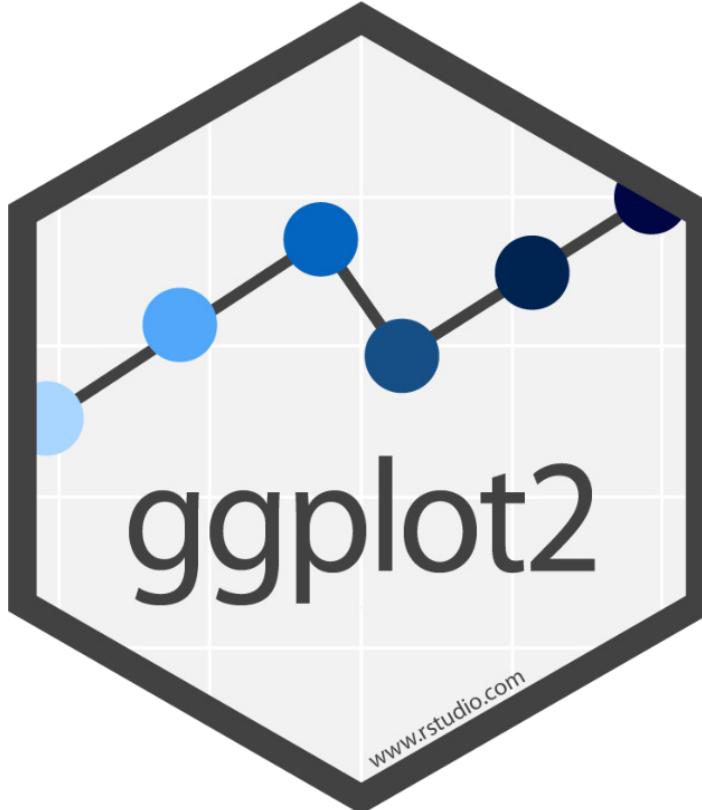


The data science workflow, modified from "R for Data Science"



Data Visualization with {ggplot2}





{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



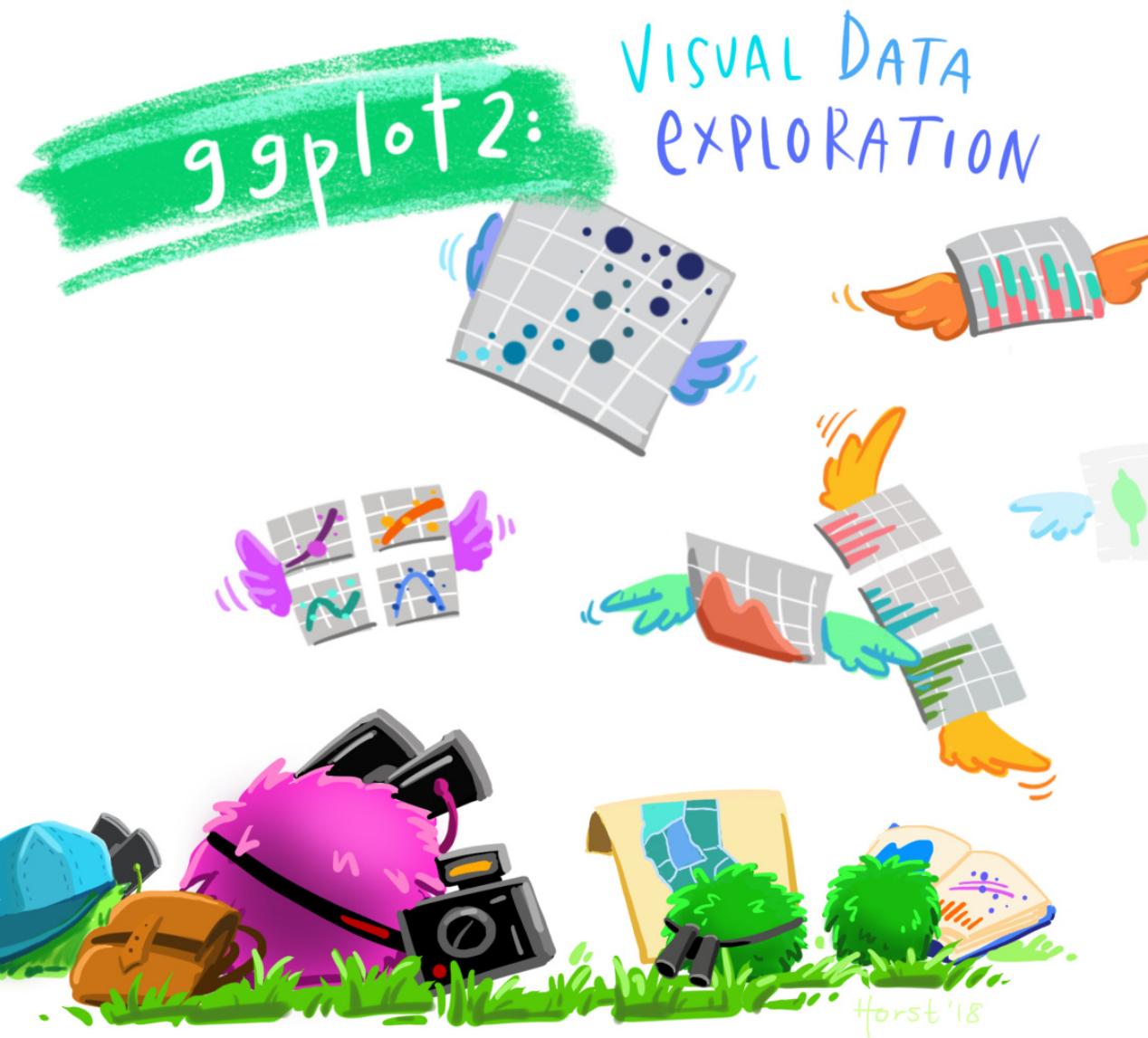
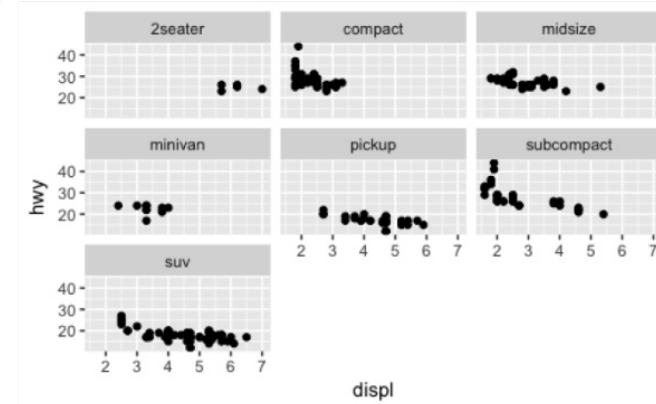
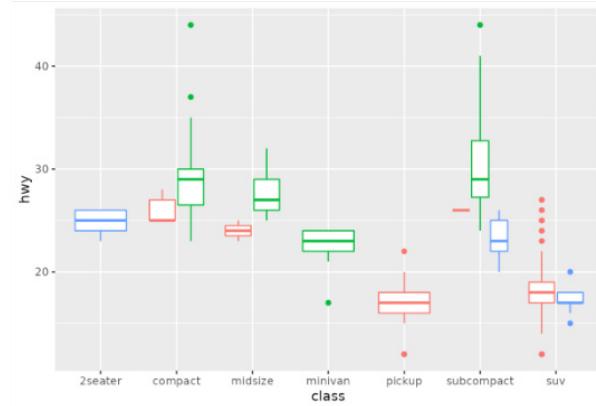
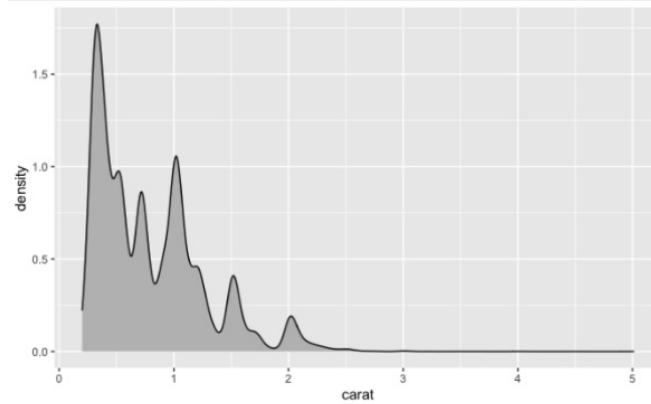
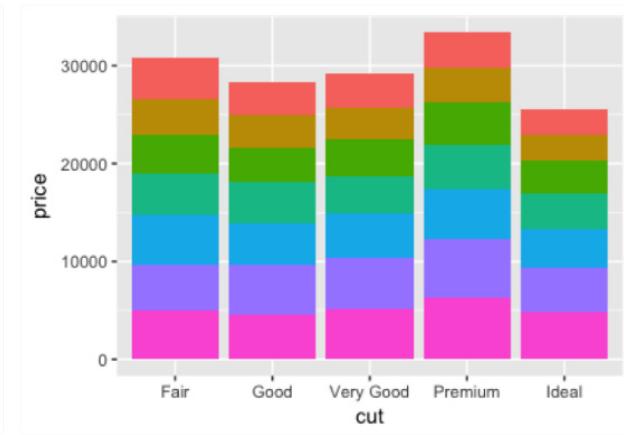
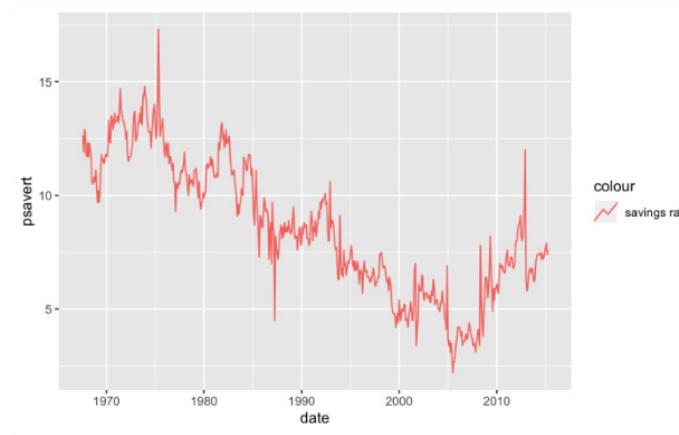
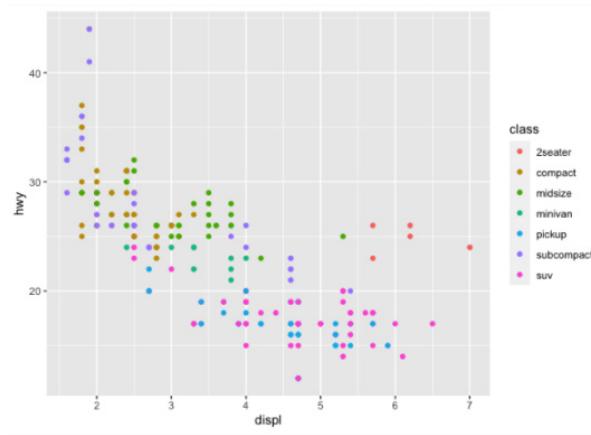
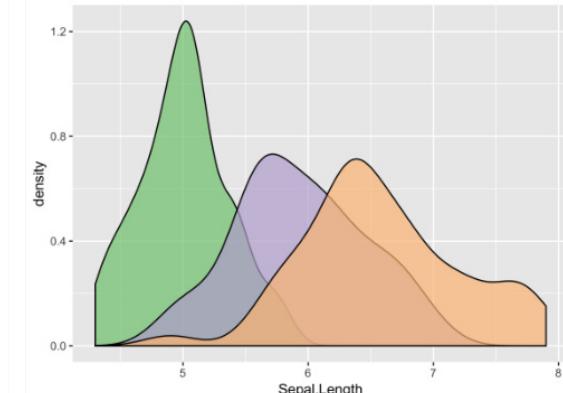
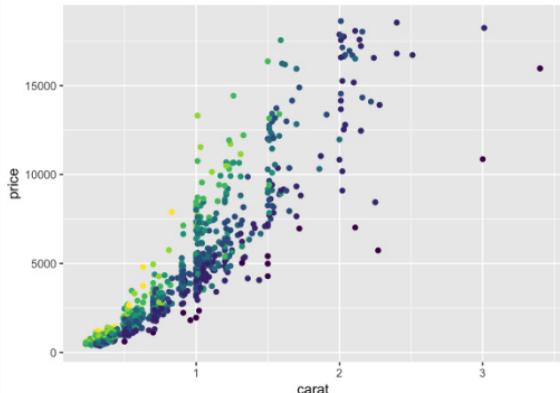
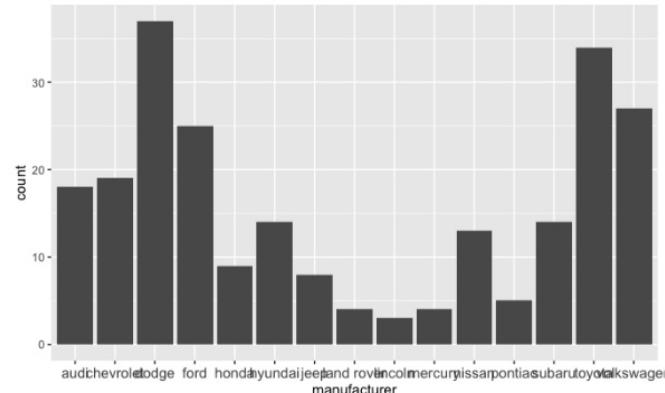


Illustration by [Allison Horst](#)

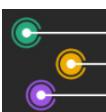
Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}





ggplot2 Examples featured on ggplot2.tidyverse.org

Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}



ggplot2:

Build a data
MASTERpiece

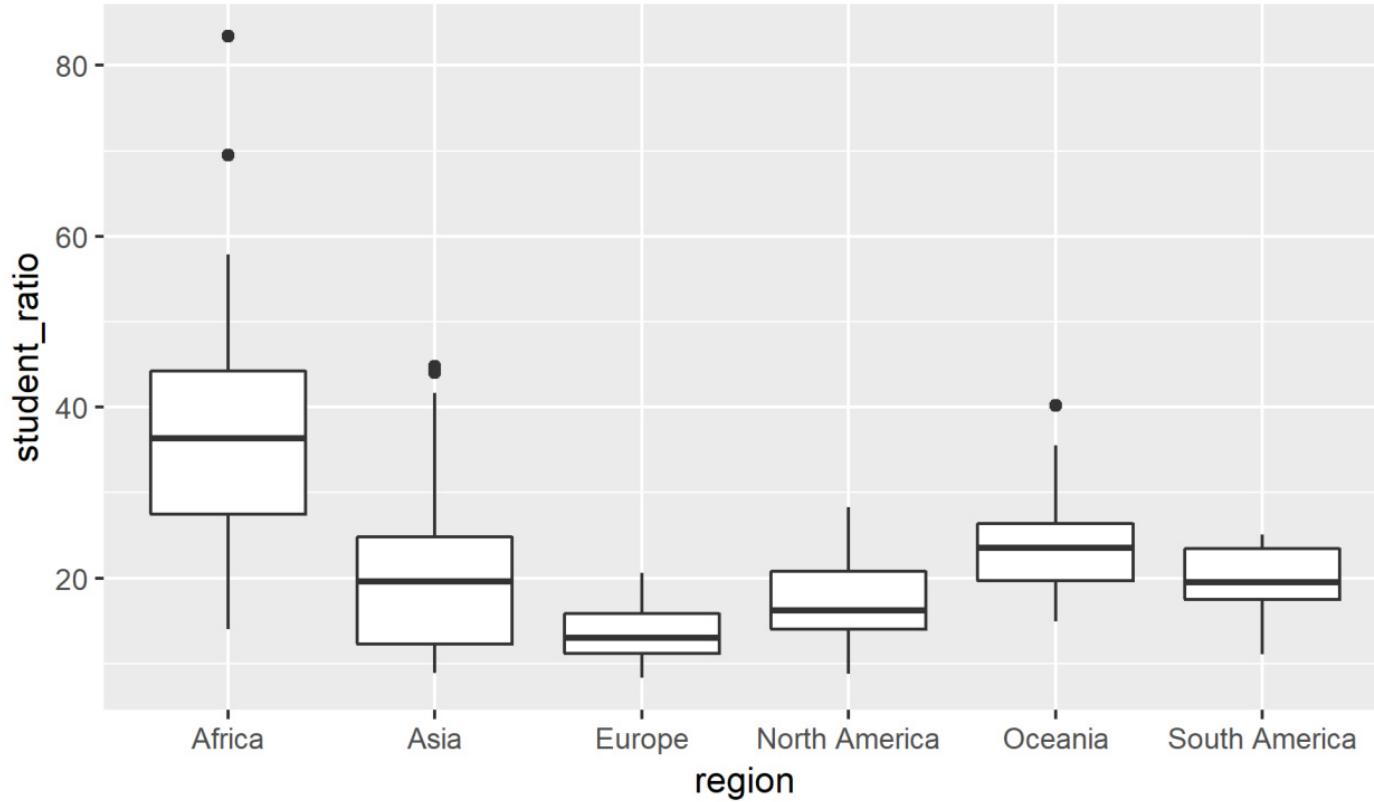


Illustration by [Allison Horst](#)

Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}

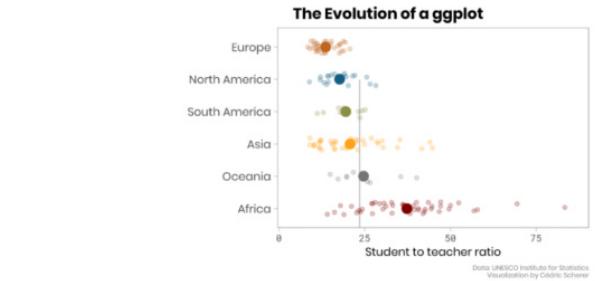
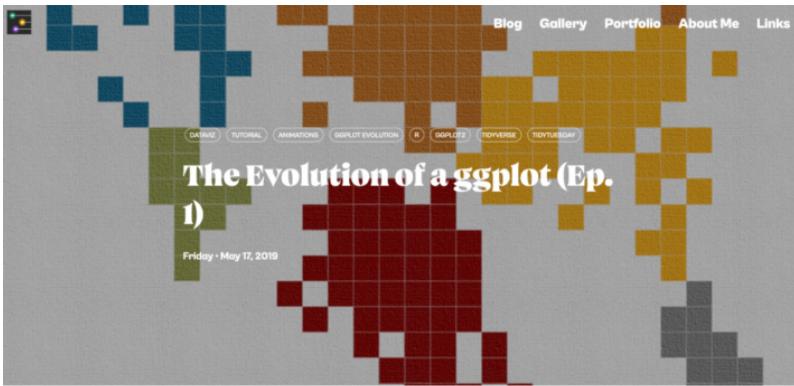


The Evolution of a ggplot



Data: UNESCO Institute for Statistics
Visualization by Cédric Scherer



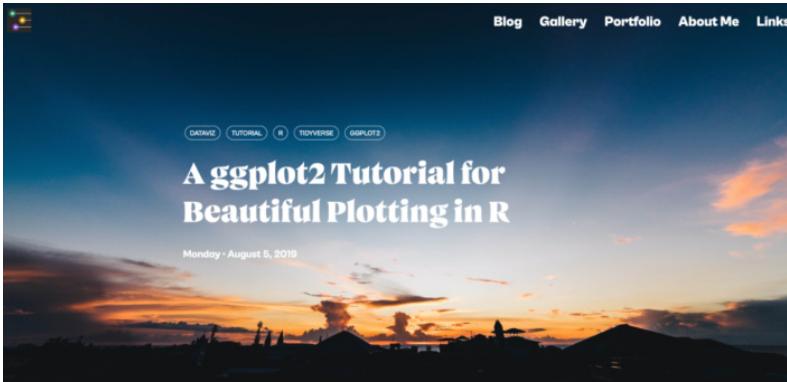


- [Aim of this Tutorial](#)
- [Data Preparation](#)
- [The Default Boxplot](#)
- [Sort Your Data!](#)
- [Let Your Plot Shine—Get Rid of the Default Settings](#)
- [The Choice of the Chart Type](#)
- [More Geoms, More Fun, More Info!](#)
- [Add Text Boxes to Let The Plot Speak for Itself](#)
- [Bonus: 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.





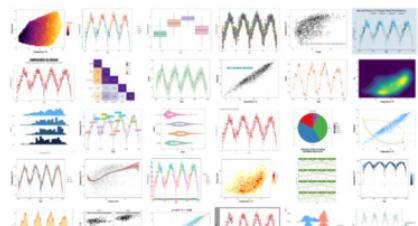
Last update: 2022-03-24

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 loves 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 Scialini 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

9h45 – 17h00

Location: Harbor 4/5

Collaboration

Course Overview

The course webpage which gives an overview of the course objectives and links to all session and the GitHub repository.

Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}



The screenshot shows Cédric Scherer's GitHub profile page. At the top, there is a navigation bar with links for Search or jump to..., Pull requests, Issues, Codepaces, Marketplace, and Explore. Below the navigation bar, there is a header with a profile picture of Cédric, his name, and a bio: "Data Visualization Designer, Consultant and Instructor for Engaging and Effective Graphical Storytelling". The bio also includes a link to his LinkedIn profile. A pie chart titled "My GitHub Activity" is displayed, showing the distribution of commits across different types. The chart has two segments: one yellow segment labeled "Commits fixing typos/rebased across types" and one small white segment labeled "Commits that actually contain code changes". Below the chart, there is a section titled "My GitHub Activity" with several small thumbnail images of data visualizations. At the bottom of the main content area, there is a yellow button with the text "Buy me a coffee".

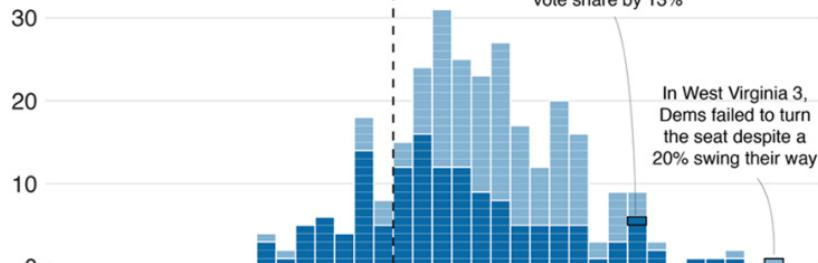
Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}



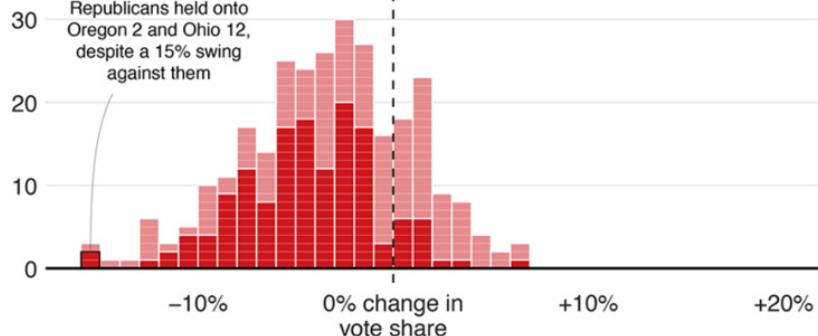
Blue wave

■ Won seat ■ Didn't win

Democrat candidates



Republican candidates

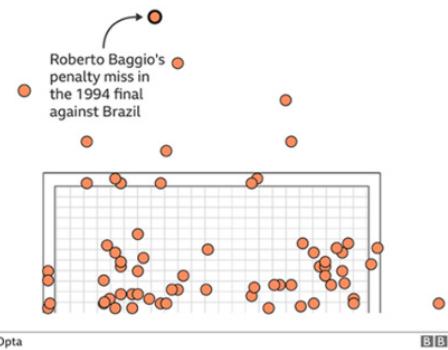


Source: AP, 19:01 ET

BBC

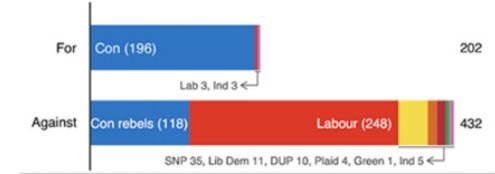
Where penalties are saved

World Cup shootout misses and saves, 1982-2014



Source: Opta

MPs rejected Theresa May's deal by 230 votes

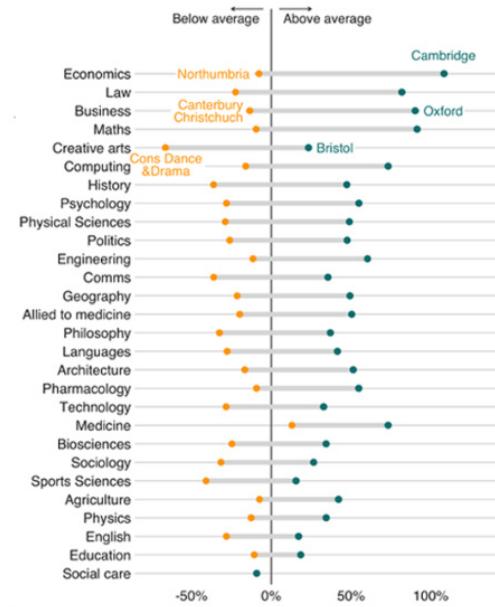


Source: Commons Votes Services. Excludes 'tellers', the Speaker and deputies

BBC

Earnings vary across unis even within subjects

Impact on men's earnings relative to the average degree



Source: Institute for Fiscal Studies

BBC



Blue wave

■ Won seat ■ Didn't win

Democrat candidates

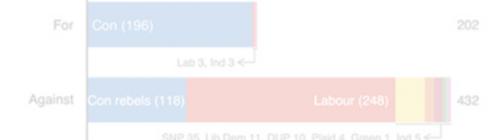


Where penalties are saved

World Cup shootout misses and saves, 1982-2014



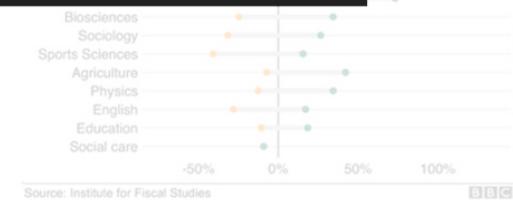
MPs rejected Theresa May's deal by 230 votes



“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.**”



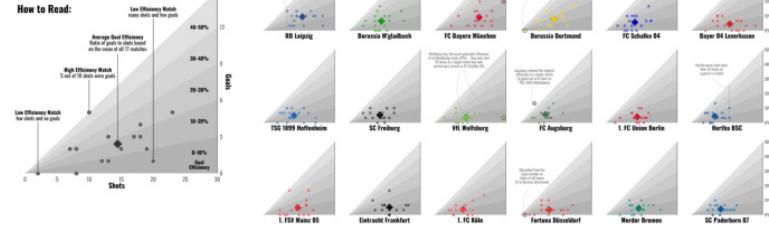
Source: Verisk Maplecroft. Circle size represents current population.



RB Leipzig and Borussia Dortmund Make the Most of Their Opportunities

The small multiple dot plots show the shooting efficiency of the two teams in the 2018/2019 season. The x-axis shows the number of shots taken per match with only two chances. The y-axis shows the percentage of goals scored.

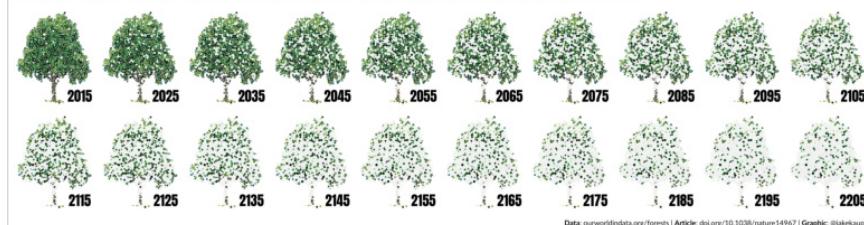
How to Read:



Visualization by Cédric Scherer

Unless someone like you cares a whole awful lot, nothing is going to get better. It's not.

In a 2015 study, published in Nature, Thomas Cropper and colleagues mapped tree density across the world. They estimated that there were approximately 3.04 trillion trees in the world. The authors also estimated that over 15 billion trees are cut down each year, and the global number of trees has fallen by almost half (46%) since the start of human civilization. Each dot on the trees below represent one billion trees. Each year 15 dots are faded out to illustrate the progressive loss. At this rate, our forests will evaporate within 200 years.



Europapark 2010

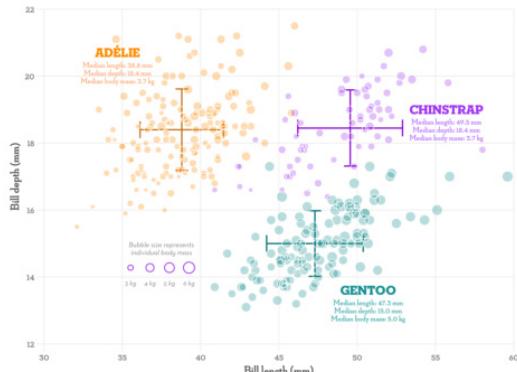
Waldgebiete Deutschlands

BILL DIMENSIONS OF BRUSH-TAILED PENGUINS

Pygoscelis eddeline (Adélie penguin) • *P. antarctica* (Chinstrap penguin) • *P. papua* (Gentoo penguin)



A. Scatterplot of bill length versus bill depth (median +/- sd)



B. Distribution of the bill ratio, estimated as bill length divided by bill depth



Note: In the original data, bill dimensions are recorded as "bill length" and "bill depth". The column is the dorsal (upper) edge of a bird's bill.
Visualization Cédric Scherer • Data: Gormez, Williams & Fraser (2014) DOI: 10.1371/journal.pone.0090087 • Illustration: Allison Horst

EUROPEAN ENERGY GENERATION

Each bar represents the total energy generation for each country per year.

The colours represent the proportion of energy generated a) using conventional thermal power stations, b) to say that use coal, or natural gas,

b) using nuclear power stations, and c) using other renewable resources

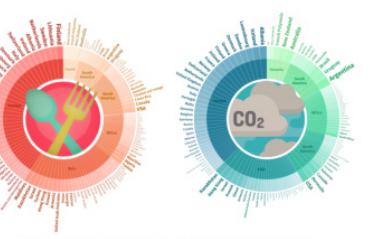
Data from: Electricity generation statistics - Eurostat (europa.eu/rapid/indicator_en)

Visualization by Cédric Scherer • Based on Jack Thorne's (datacamp) 'DataCamp' package

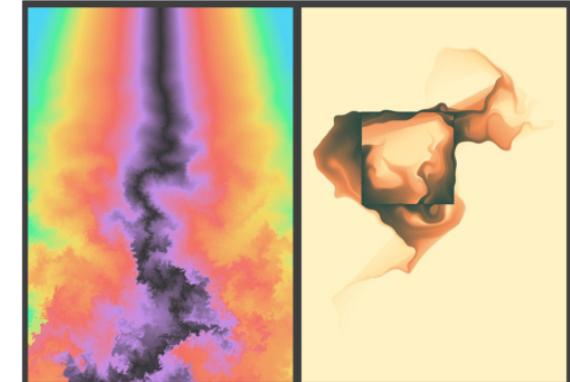
Code: Cédric Scherer

Food Carbon Footprint Index 2018

Global comparison of different data in terms of Average Consumption (kg) / person/year of both animal and non-animal products as well as Carbon Emissions (kg CO₂/person/year) per continent and country. Font size and color intensity indicate each country's estimate with countries printed in bold belonging to the upper 50% of consumers and CO₂ emissions, respectively.



Visualization by Cédric Scherer • Based on Jack Thorne's (datacamp) 'FoodCarbon' package



Communicating fast doesn't necessarily mean communicating more

More

Less

Normalized rates of speech (syllables per second) and information (bits per second)

Japanese

Basque

Korean

Indo-European

Turkic

Uralic

Sino-Tibetan

Austrasiatic

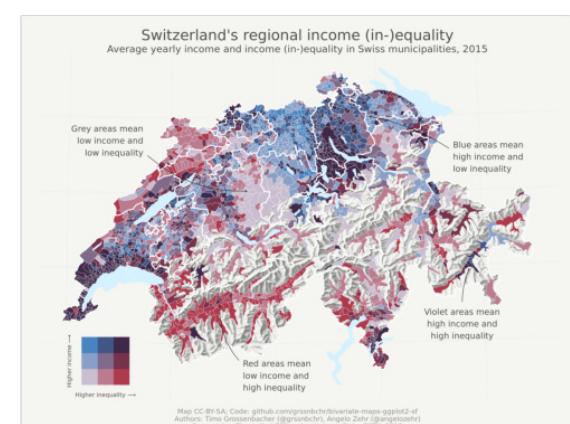
Tai-Kadai

Normalized rates of speech (syllables per second) and information (bits per second)

Source: Coupe et al. 2019 Science Advances 8(8): eaav2594

Graphic: Cédric Scherer

Large dots show the median rates for each language family. Small dots show single estimates.



Switzerland's regional income (in-)equality

Average yearly income and income (in-)equality in Swiss municipalities, 2015

Higher income

Higher inequality

Grey areas mean low income and low inequality

Blue areas mean high income and low inequality

Red areas mean low income and high inequality

Violet areas mean high income and high inequality

Map CC-BY-SA. Code: gitlab.com/cscherer/uni-variate-mean-ggplot2

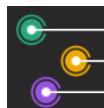
Authors: Timo Gossenbacher (@gossenbacher), Angelo Zehl (@angelozehl)

Geometries: Themakart BFS and swisstopo

Data: ESTV, 2015

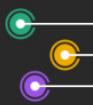
Selection of visualizations created 100% with ggplot2 by Thomas Linn Pedersen, Georgios Karamanis, Timo Gossenbacher, Torsten Sprengler, Jake Kaupp, Jack Davison, and myself.

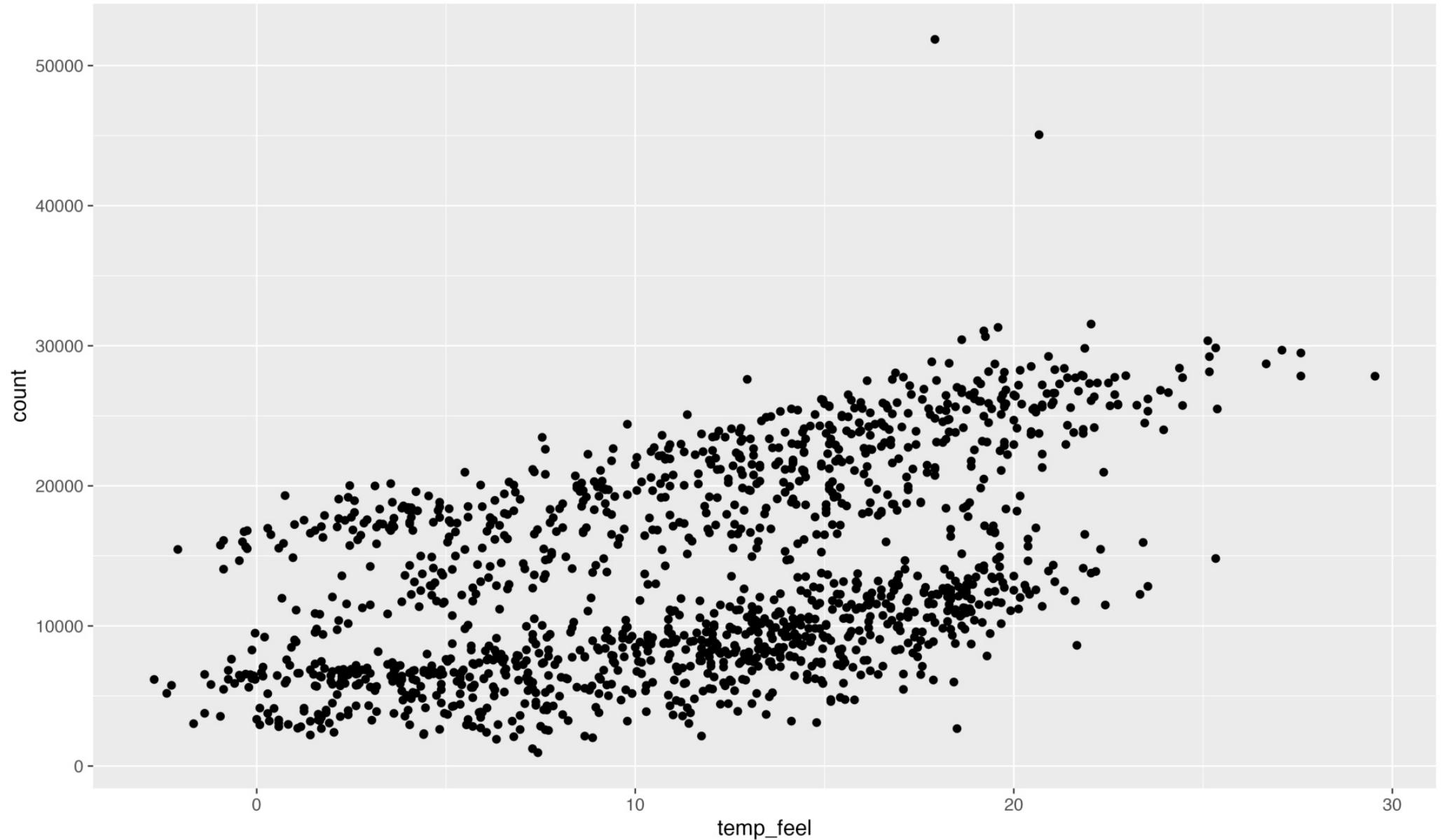
Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}

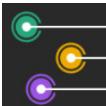
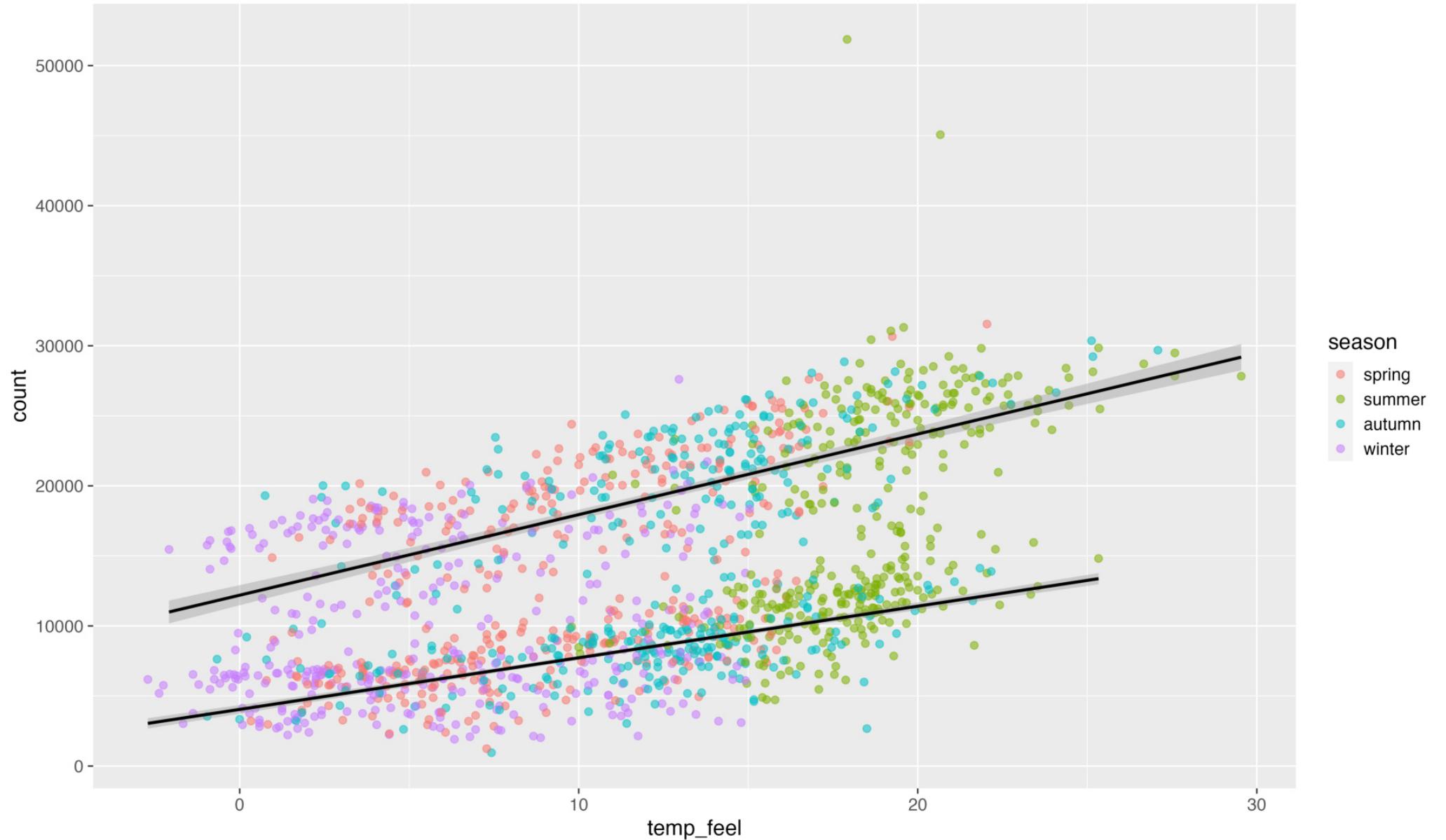


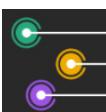
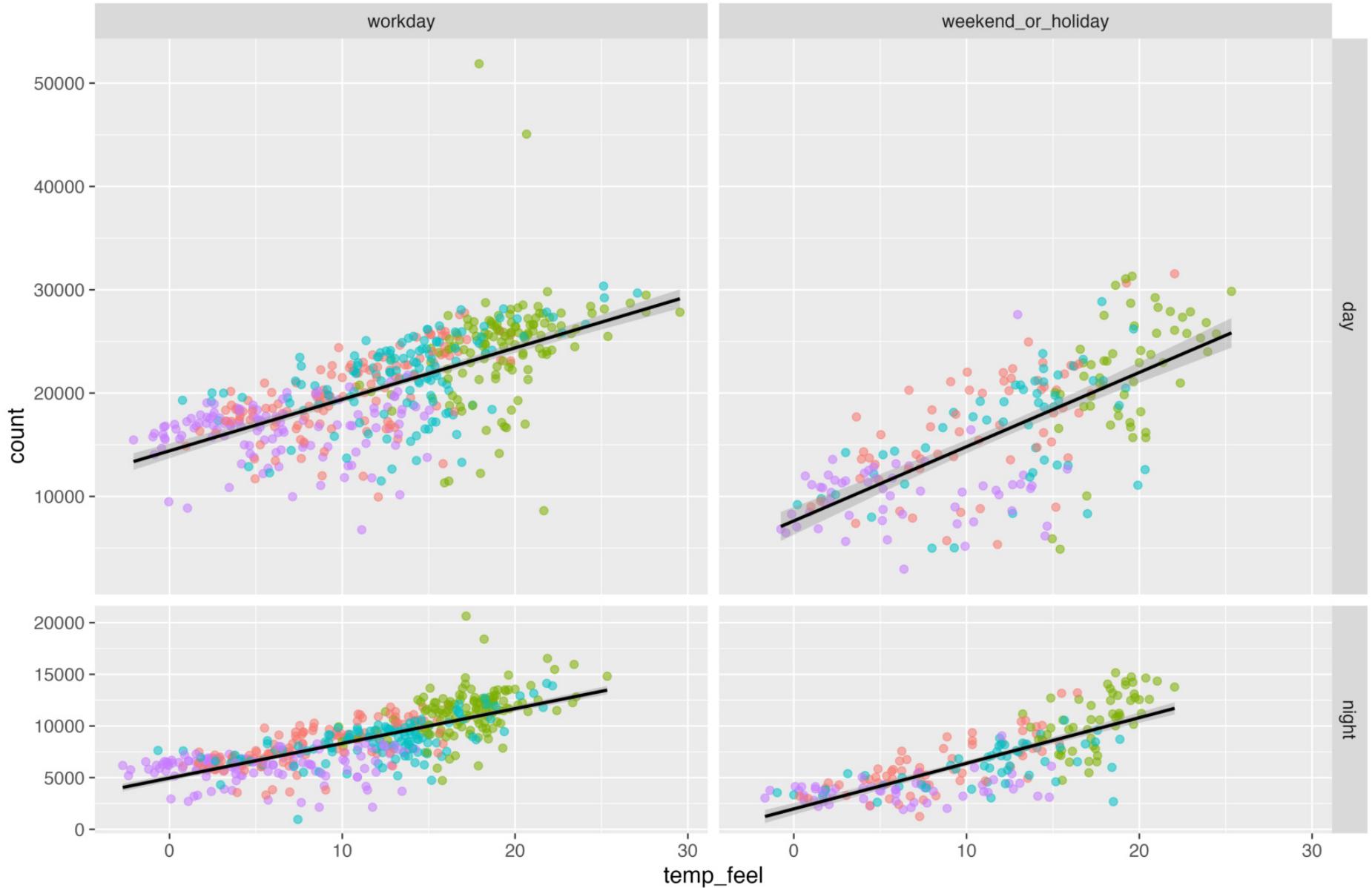
Data Visualization with {ggplot2}

— A Motivational Example —

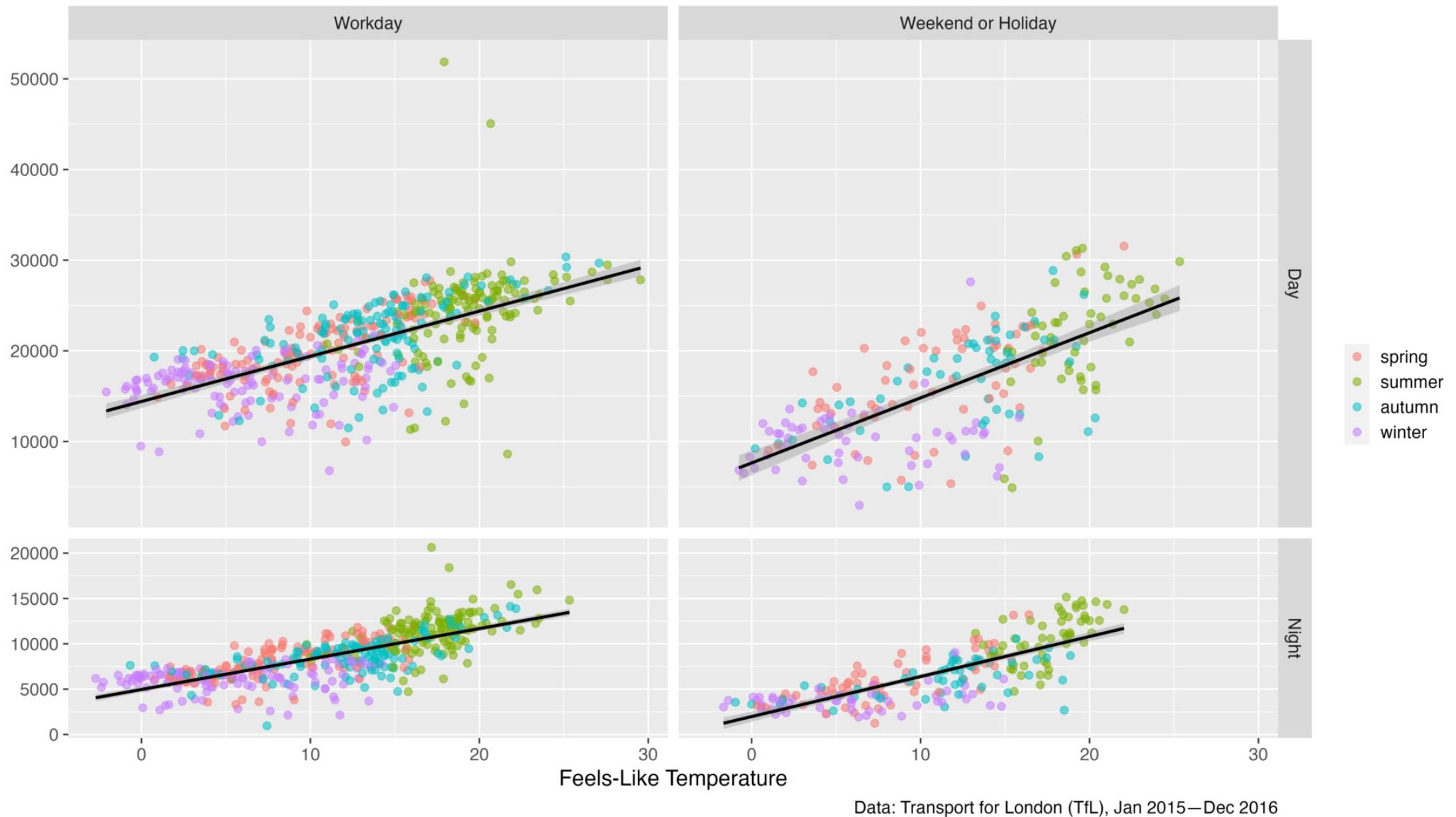




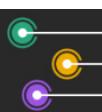
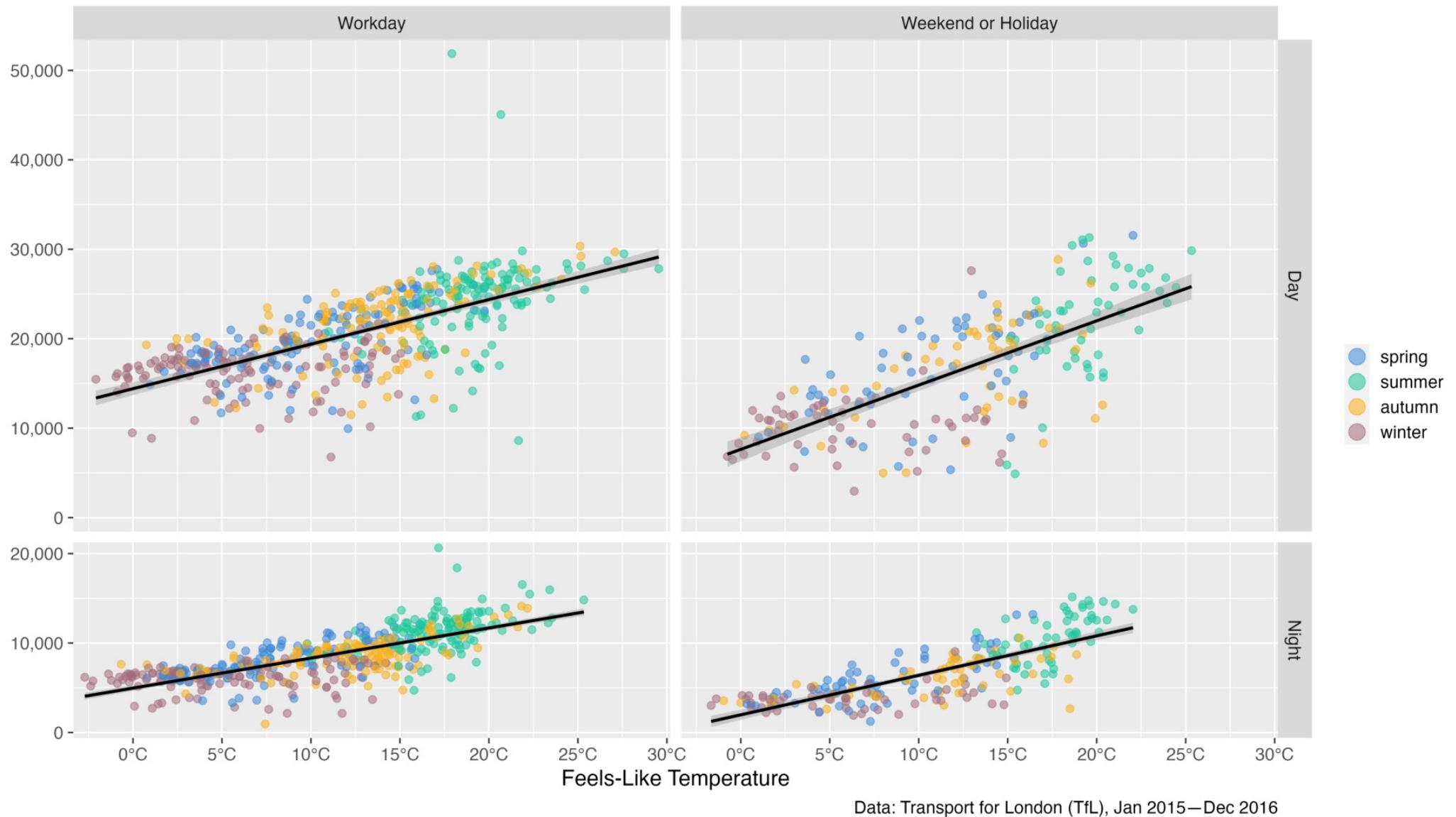




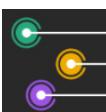
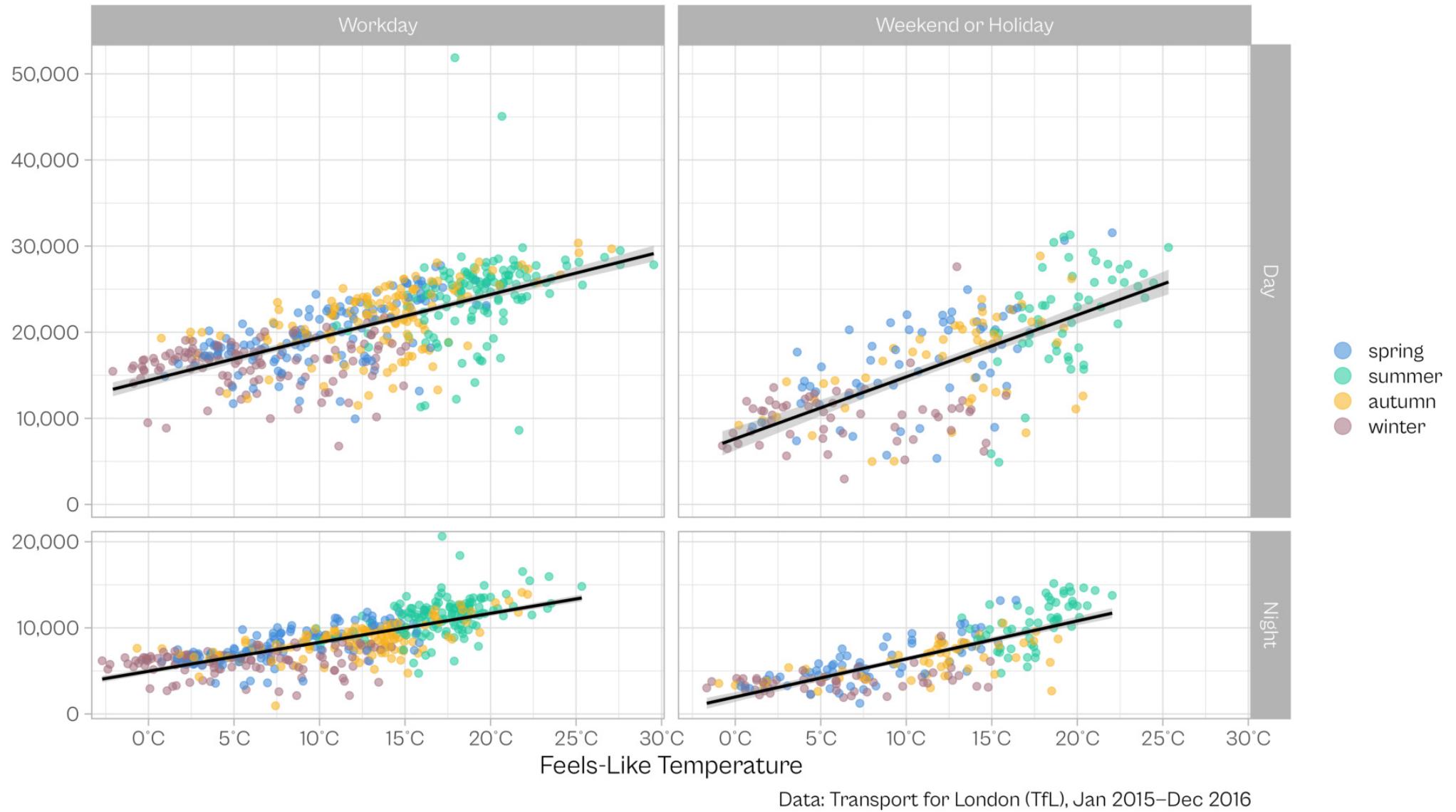
Reported TfL bike rents versus feels-like temperature in London, 2015–2016



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

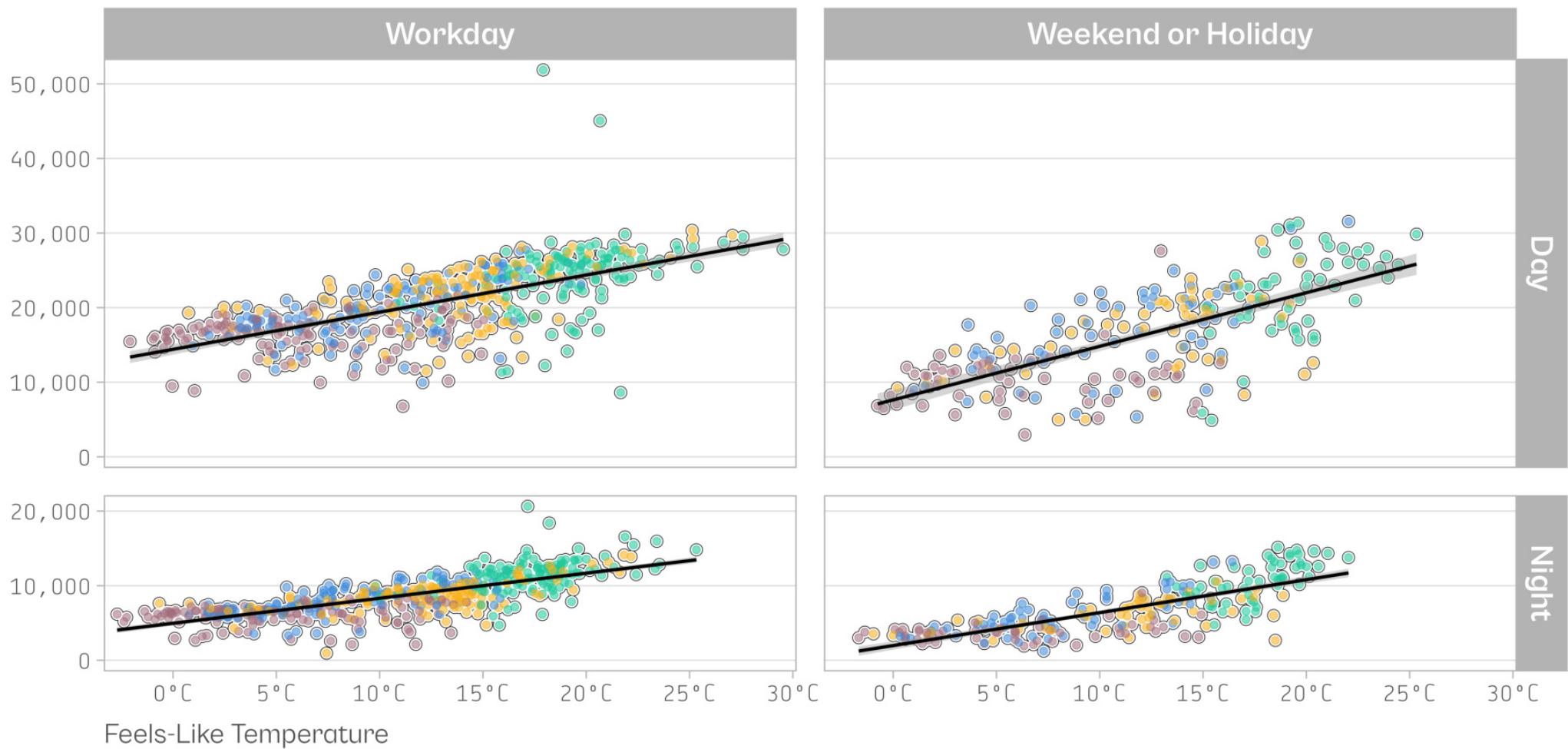


Reported TfL bike rents versus feels-like temperature in London, 2015–2016



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

● Spring ● Summer ● Autumn ● Winter

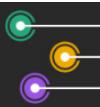


Data: Transport for London (TfL), Jan 2015–Dec 2016



Data Visualization with {ggplot2}

— The Setup —



The ggplot2 Package

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

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

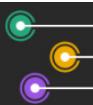
... is part of the **{tidyverse}**

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



Data Visualization with {ggplot2}

— The Grammar of Graphics —



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>
Coordinate System	<code>coord_*</code> ()	<i>Maps data into the plane of the data rectangle.</i>
Facets	<code>facet_*</code> ()	<i>The arrangement of the data into a grid of plots.</i>
Visual Themes	<code>theme()</code> and <code>theme_*</code> ()	<i>The overall visual defaults of a plot.</i>



The Data



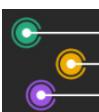
Bike sharing counts in London, UK, powered by TfL Open Data

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

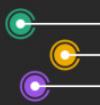


Variable	Description	Class
date	Date encoded as `YYYY-MM-DD`	date
day_night	`day` (6:00am–5:59pm) or `night` (6:00pm–5:59am)	character
year	`2015` or `2016`	factor
month	`1` (January) to `12` (December)	factor
season	`winter`, `spring`, `summer`, or `autumn`	factor
count	Sum of reported bikes rented	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



Data Visualization with {ggplot2}

— Fundamental Concepts —



ggplot2::ggplot()

1 ?ggplot

ggplot: Create a new ggplot

Description

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

Usage

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

Arguments

data Default dataset to use for plot. If not already a `data.frame`, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.

mapping Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.

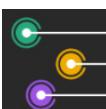
... Other arguments passed on to methods. Not currently used.

environment DEPRECATED. Used prior to tidy evaluation.

Details

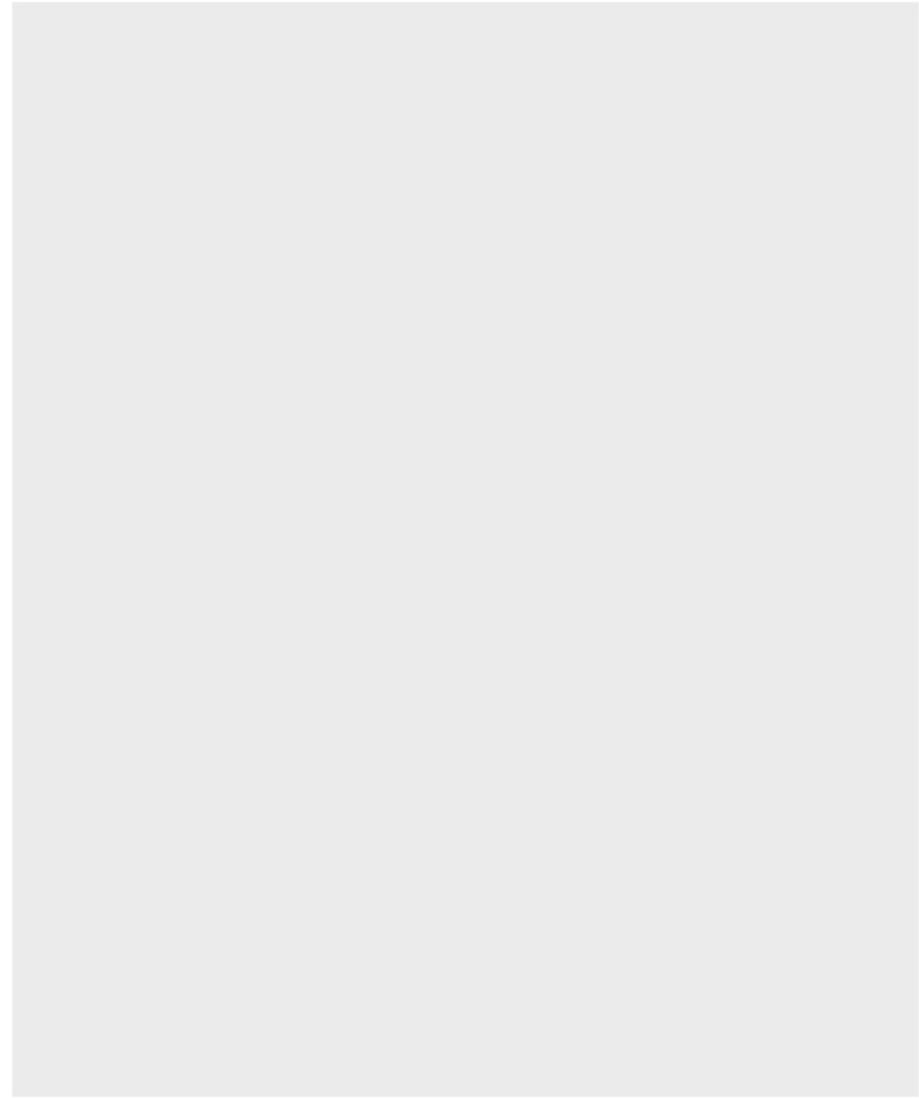
`ggplot()` is used to construct the initial plot object, and is almost always followed by `+` to add component to the plot. There are three common ways to invoke `ggplot()`:

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



Data

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



Aesthetic Mapping

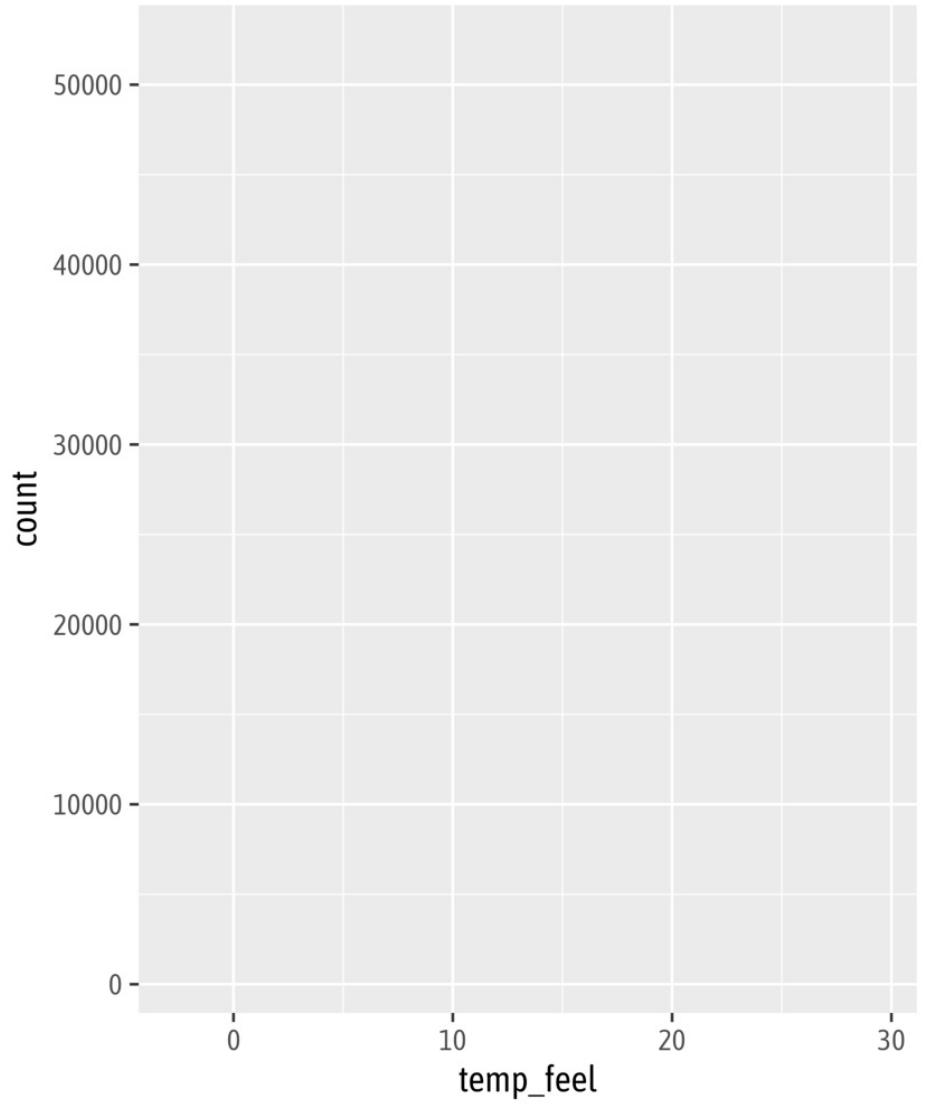
= link variables to graphical properties

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



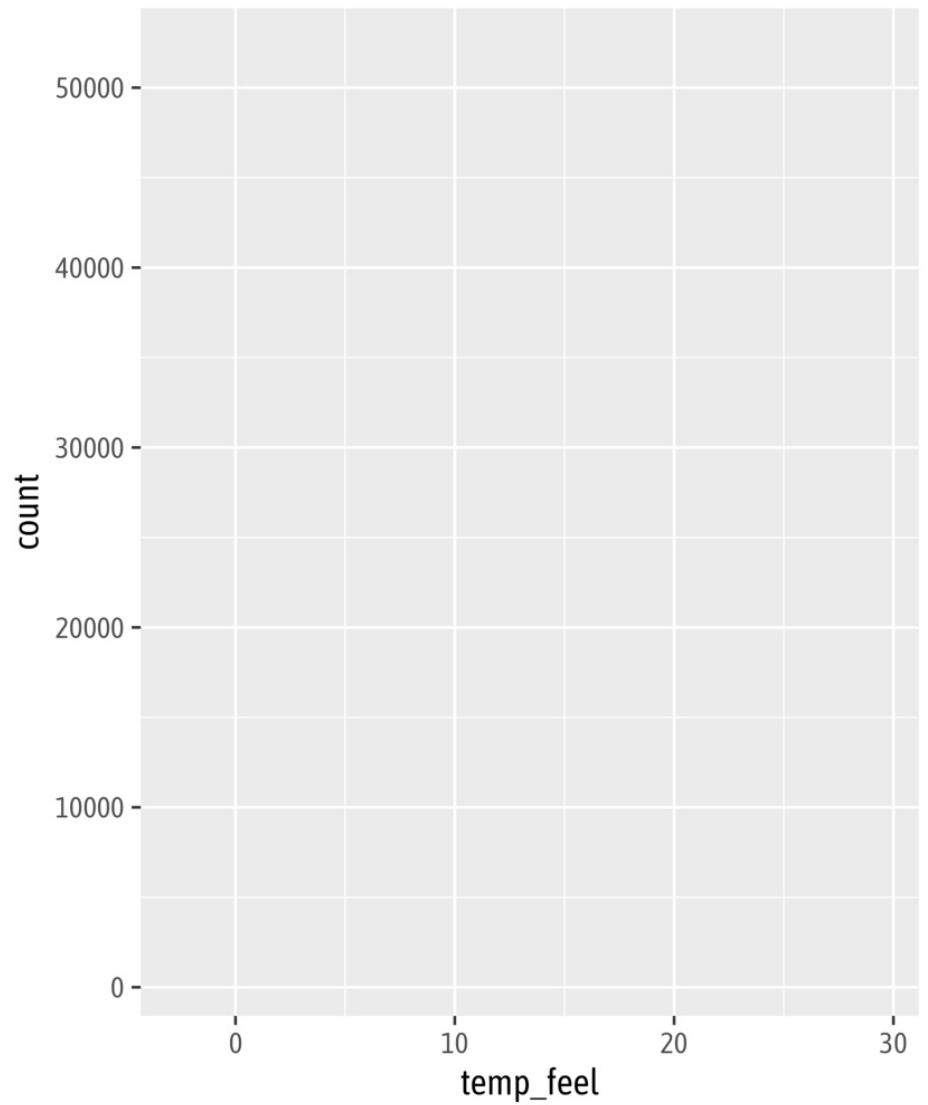
Aesthetic Mapping

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



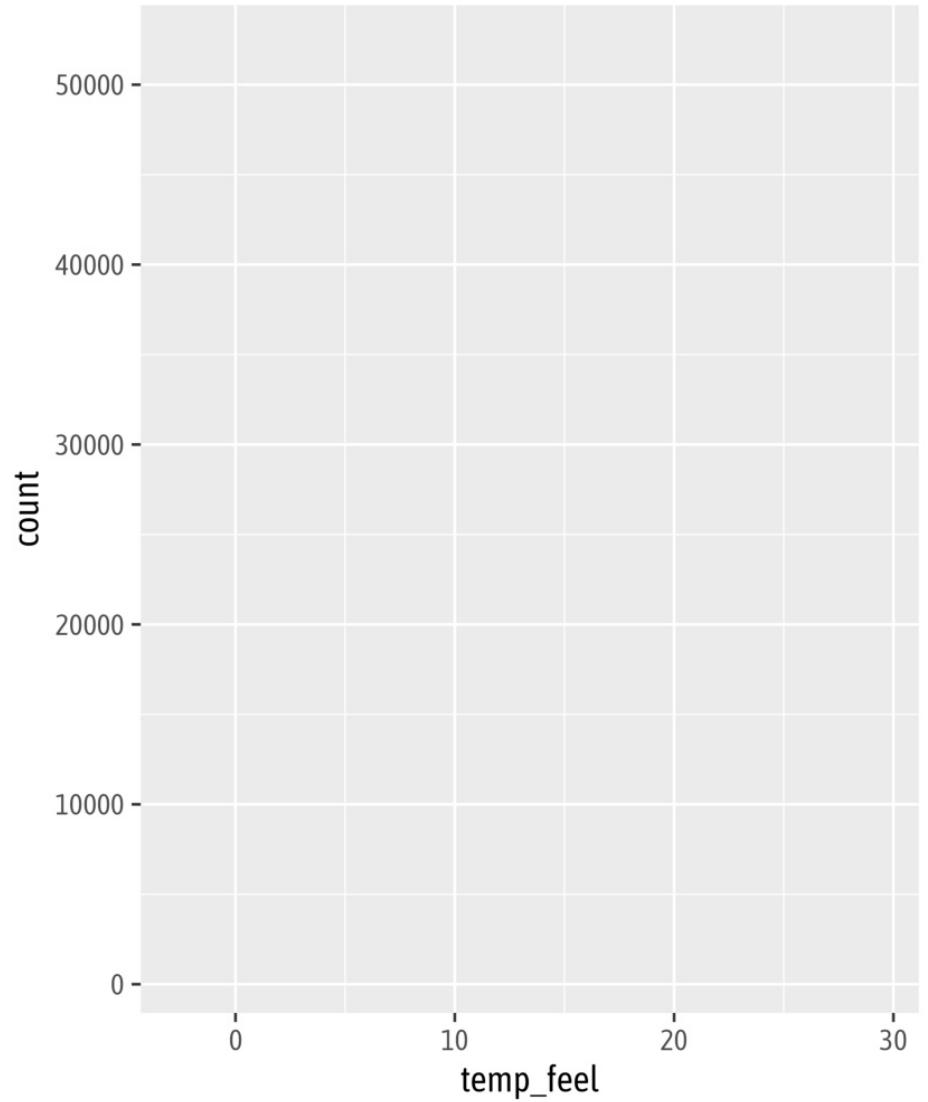
Aesthetic Mapping

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



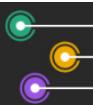
Aesthetic Mapping

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



Data Visualization with {ggplot2}

— Geometrical Layers —



Geometries

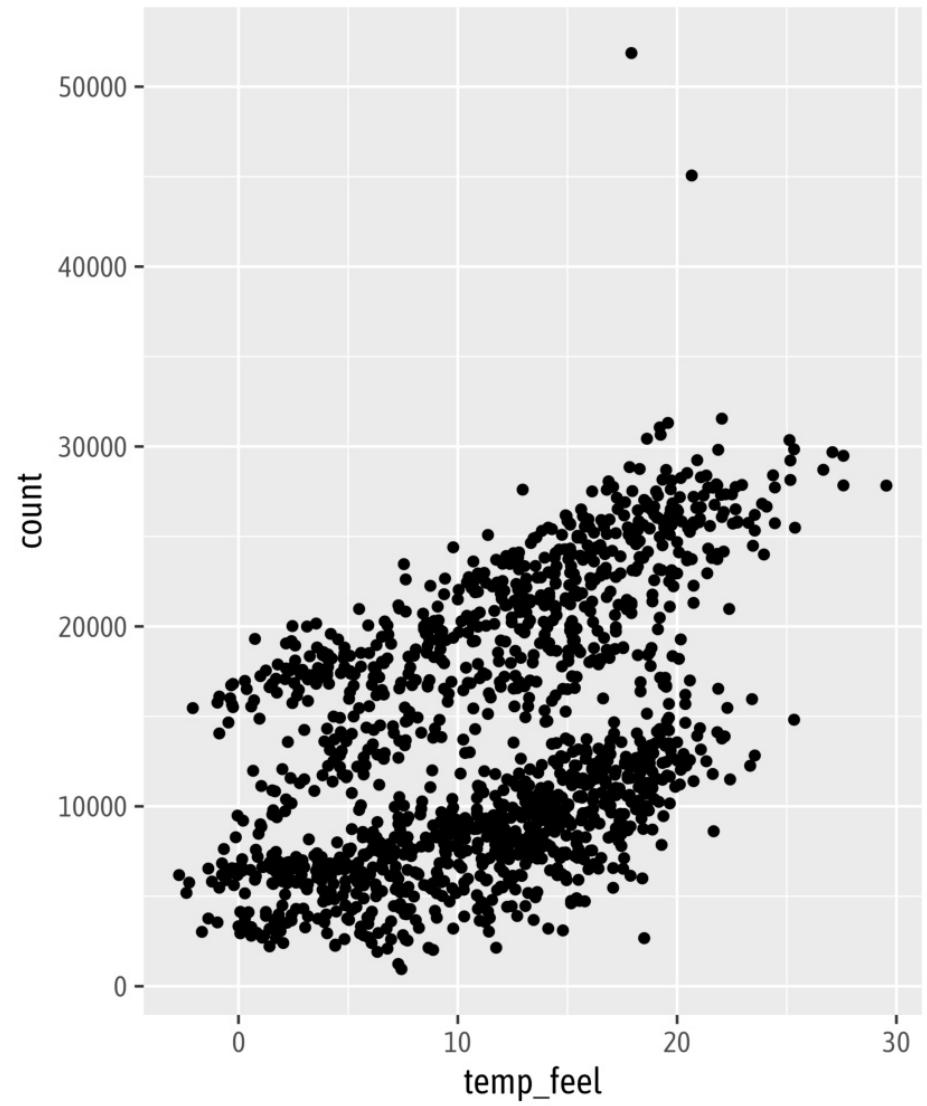
= interpret aesthetics as graphical representations

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



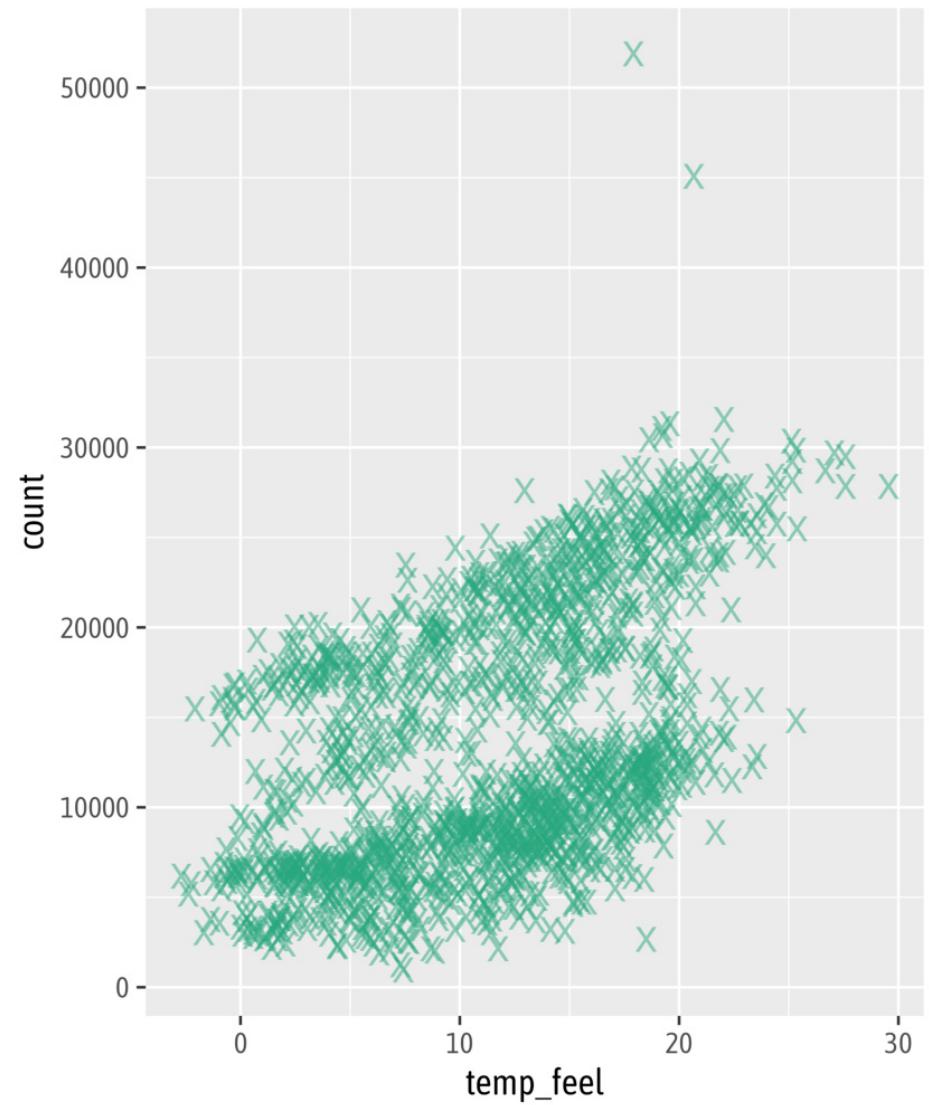
Geometries

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



Visual Properties of Layers

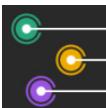
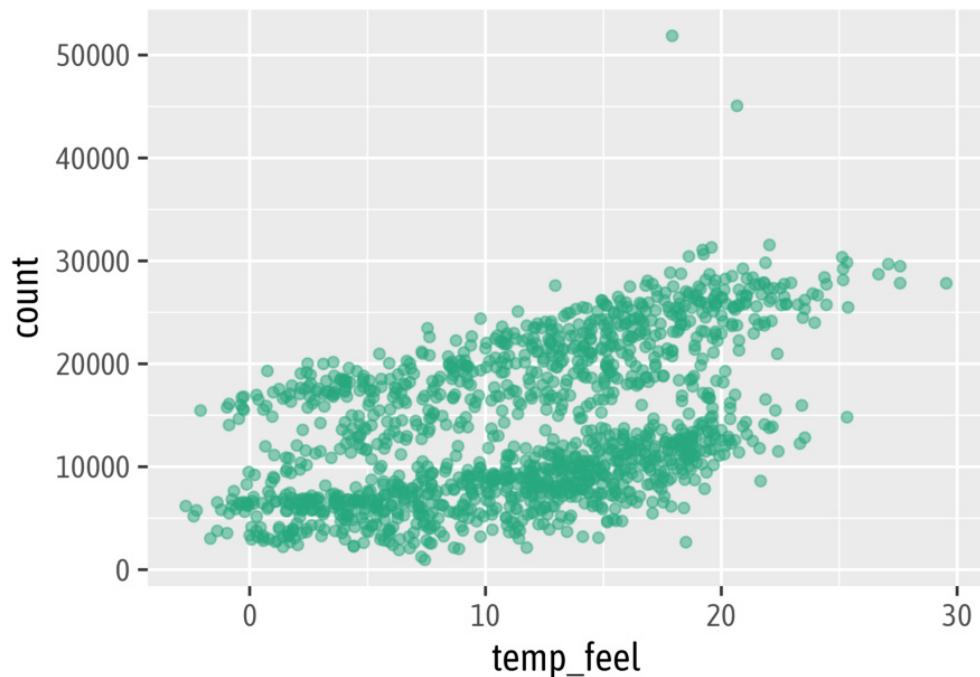
```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5   geom_point(  
6   color = "#28a87d",  
7   alpha = .5,  
8   shape = "X",  
9   stroke = 1,  
10  size = 4  
11 )
```



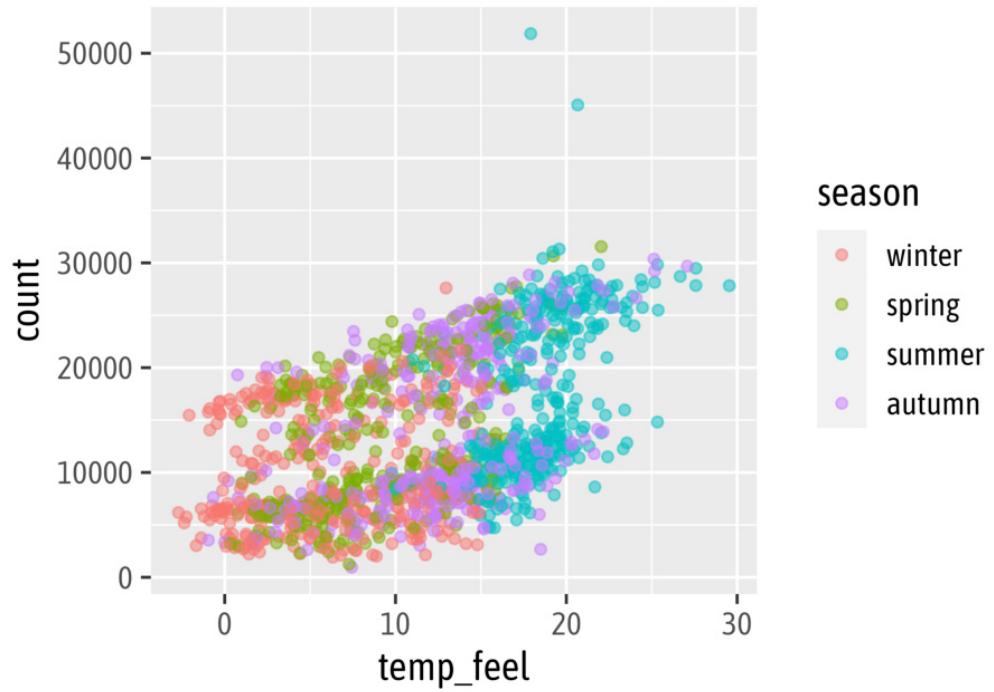
Setting vs Mapping of Visual Properties

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

...

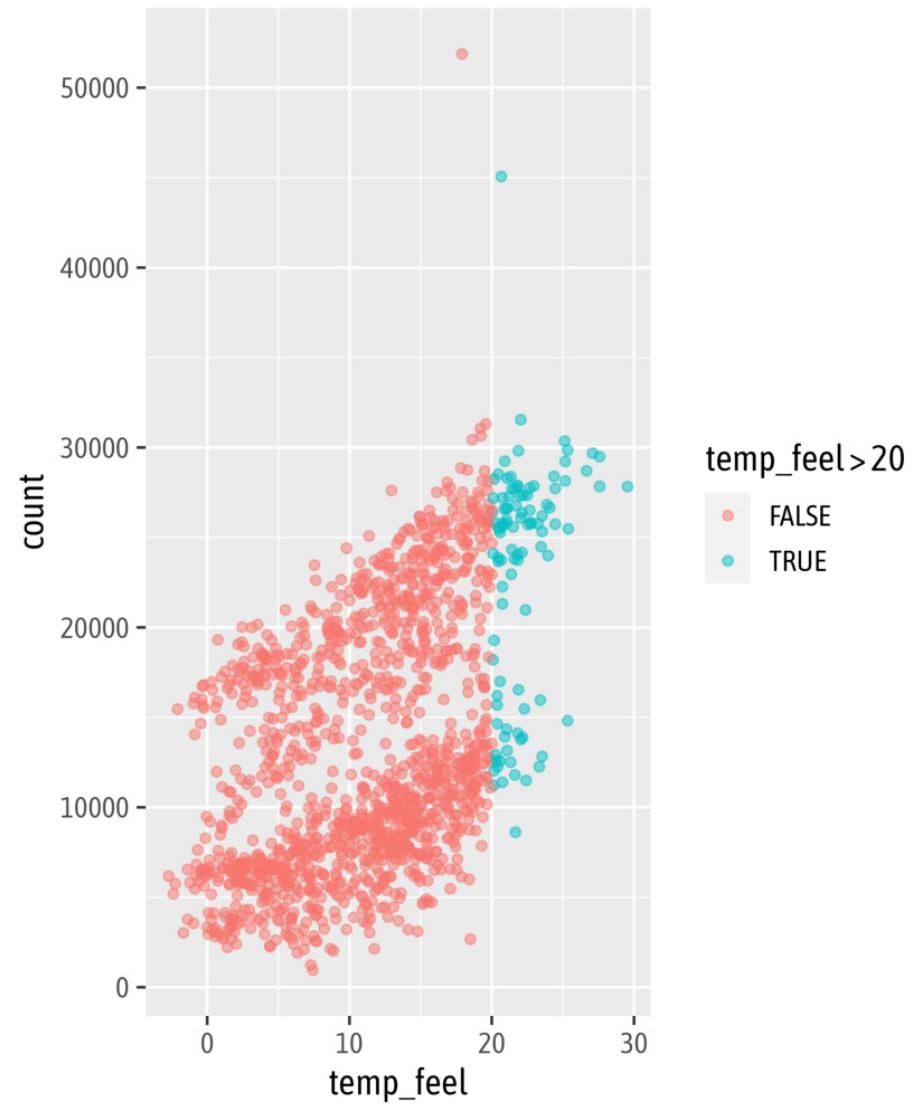


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point(  
6   aes(color = season),  
7   alpha = .5  
8 )
```



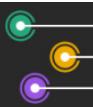
Mapping Expressions

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



Data Visualization with {ggplot2}

— Exercise —



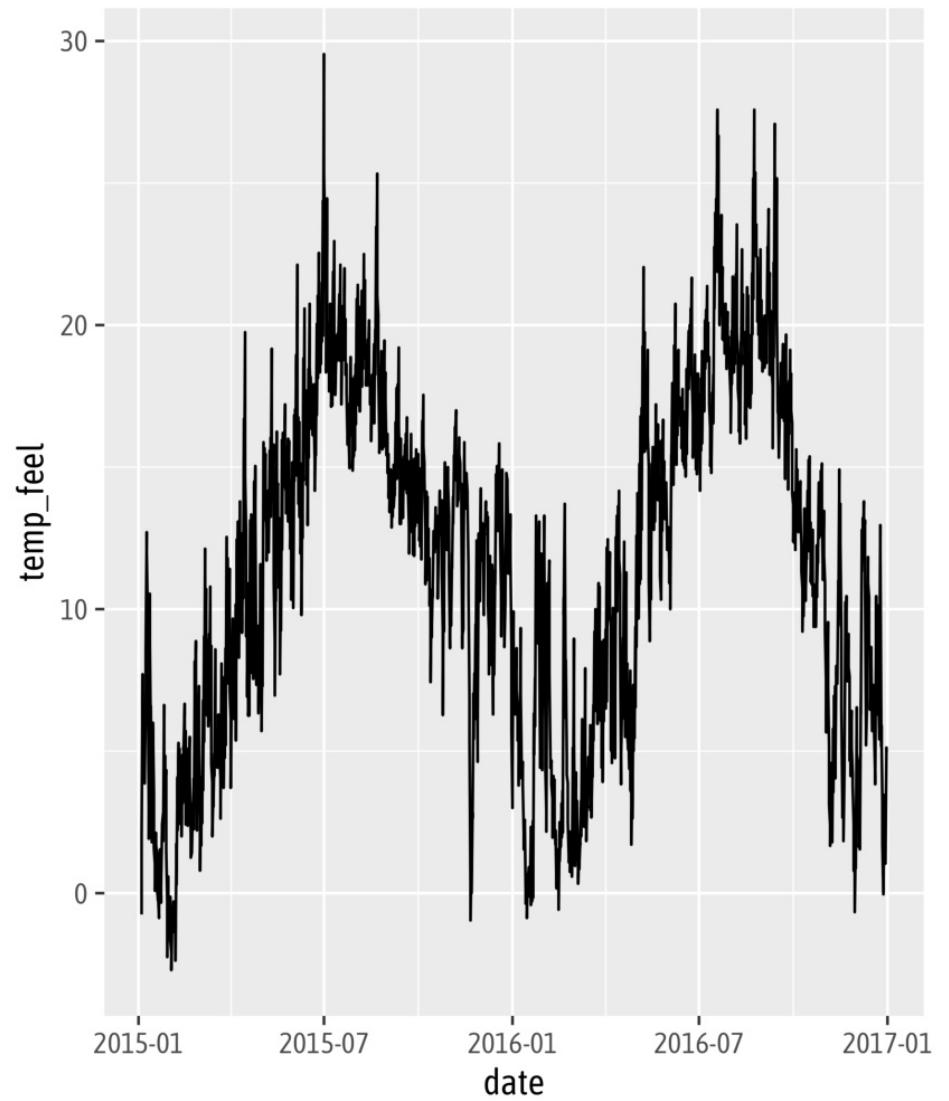
Your Turn: Geometries & Aesthetics

- Create a chart showing a time series of `temp_feel`.
 - What is the difference between `geom_line()` and `geom_path()`?
 - Map the color of the lines to `day_night`.
 - Add points for each observation.
 - Turn the points into diamonds.
- **Bonus:** Create a time series of bike counts in 2016 during nights only, with points colored by season.



Your Turn: Geometries & Aesthetics

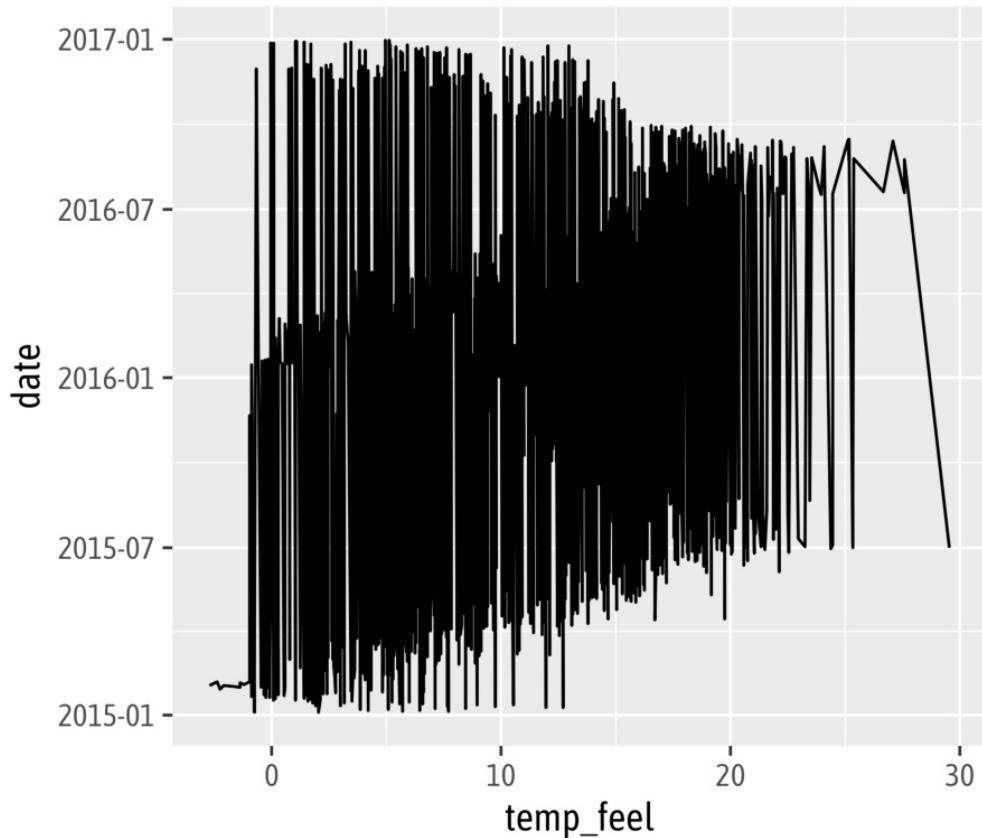
```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = temp_feel)  
4 ) +  
5 geom_line()
```



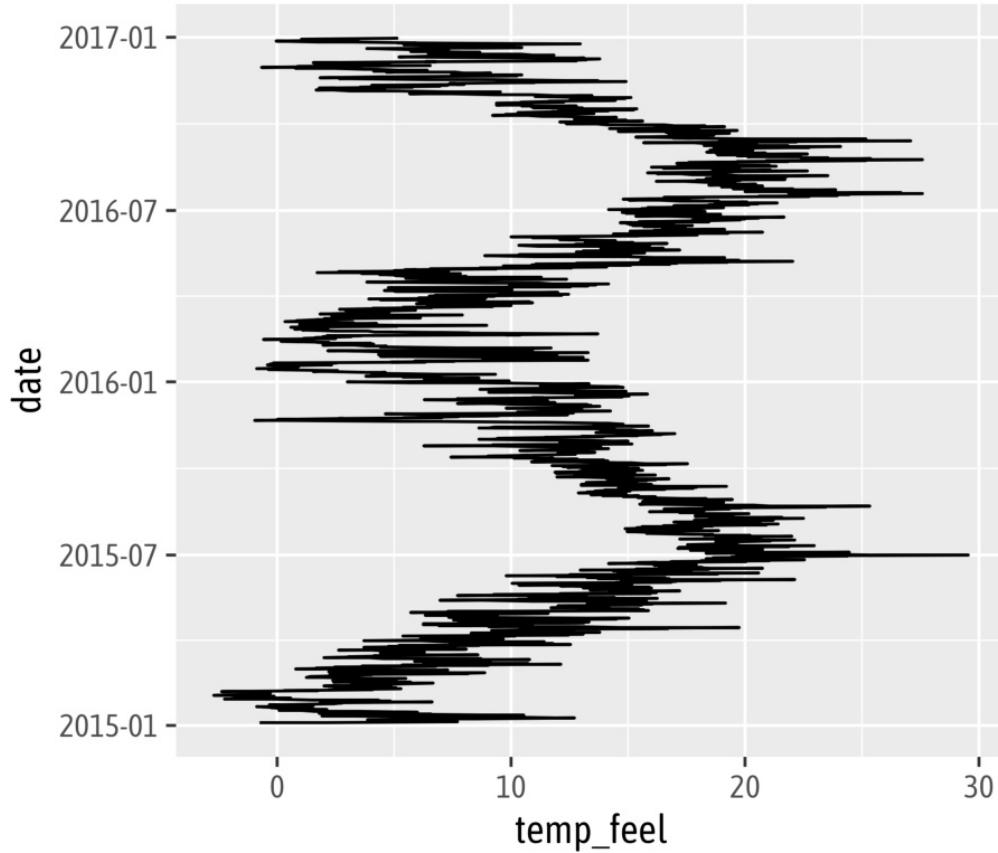
Your Turn: Geometries & Aesthetics

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

...

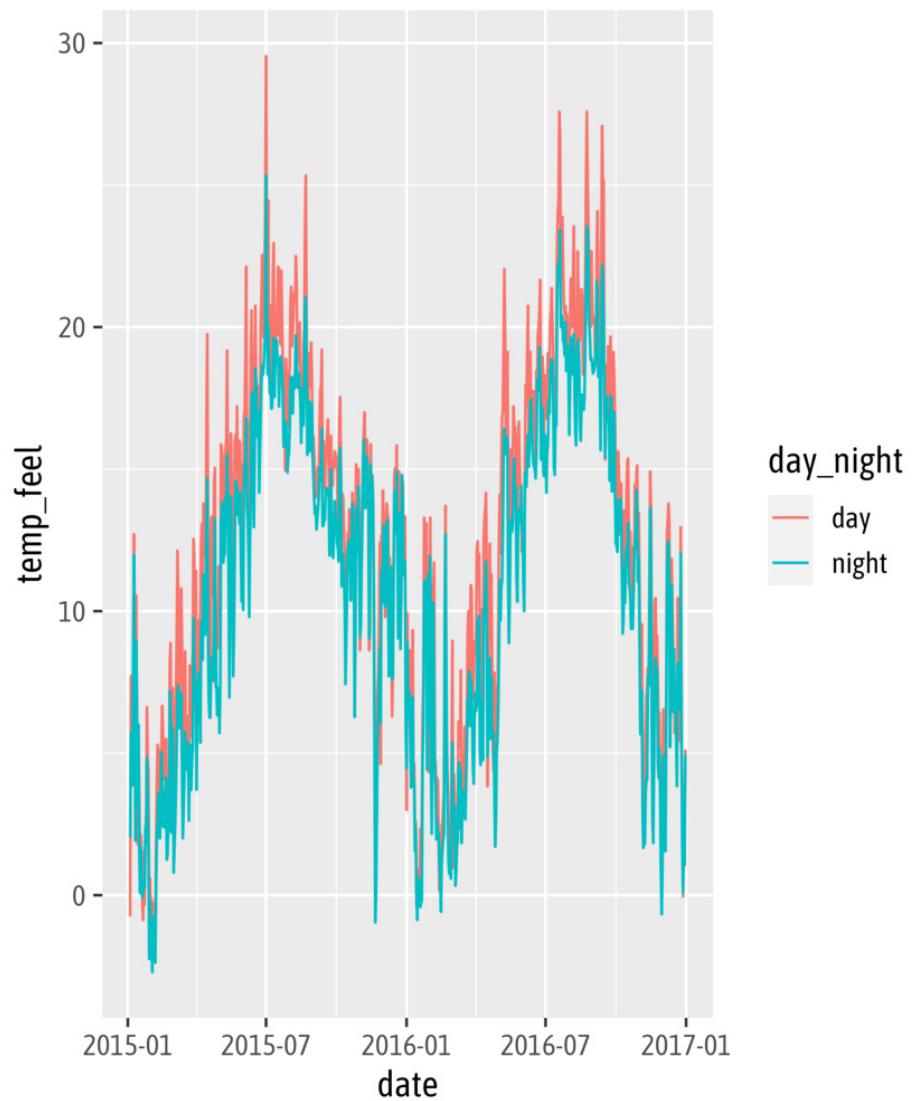


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = date)  
4 ) +  
5 geom_path()
```



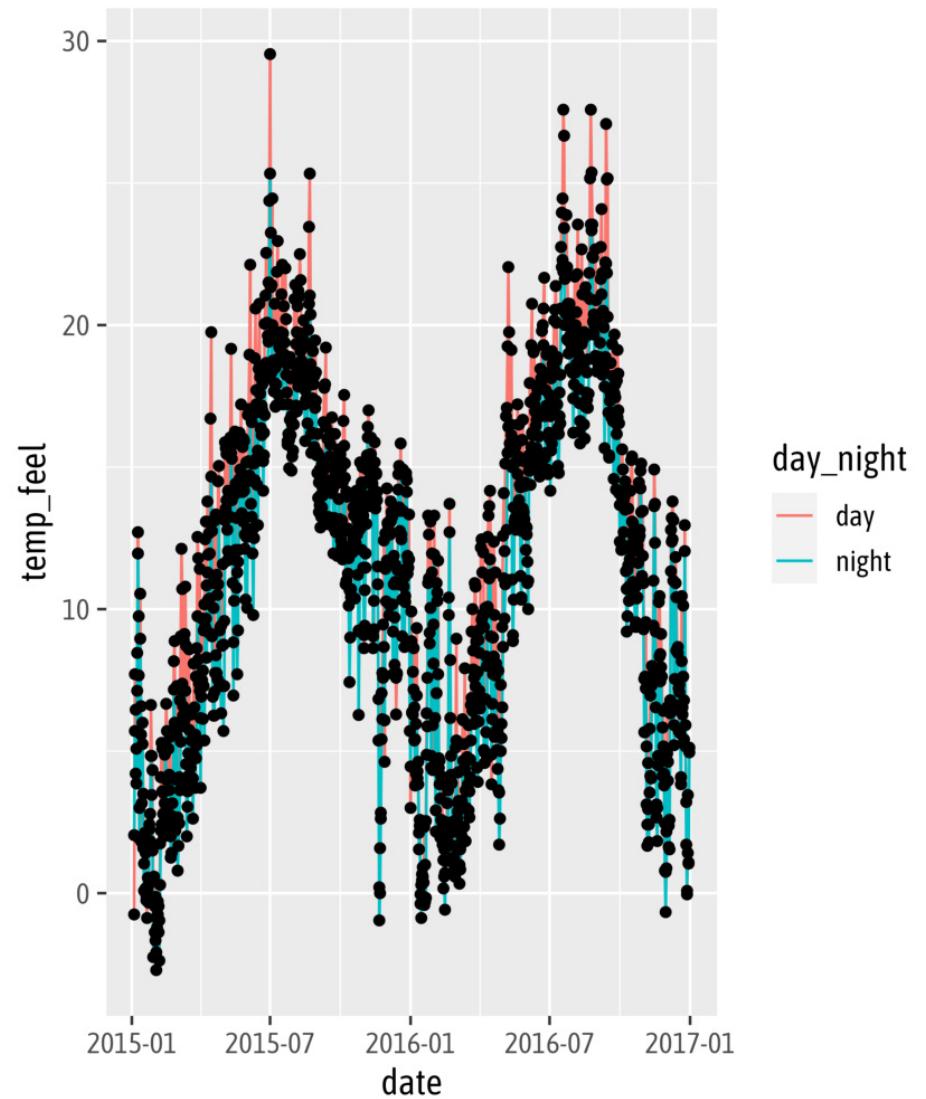
Your Turn: Geometries & Aesthetics

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



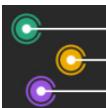
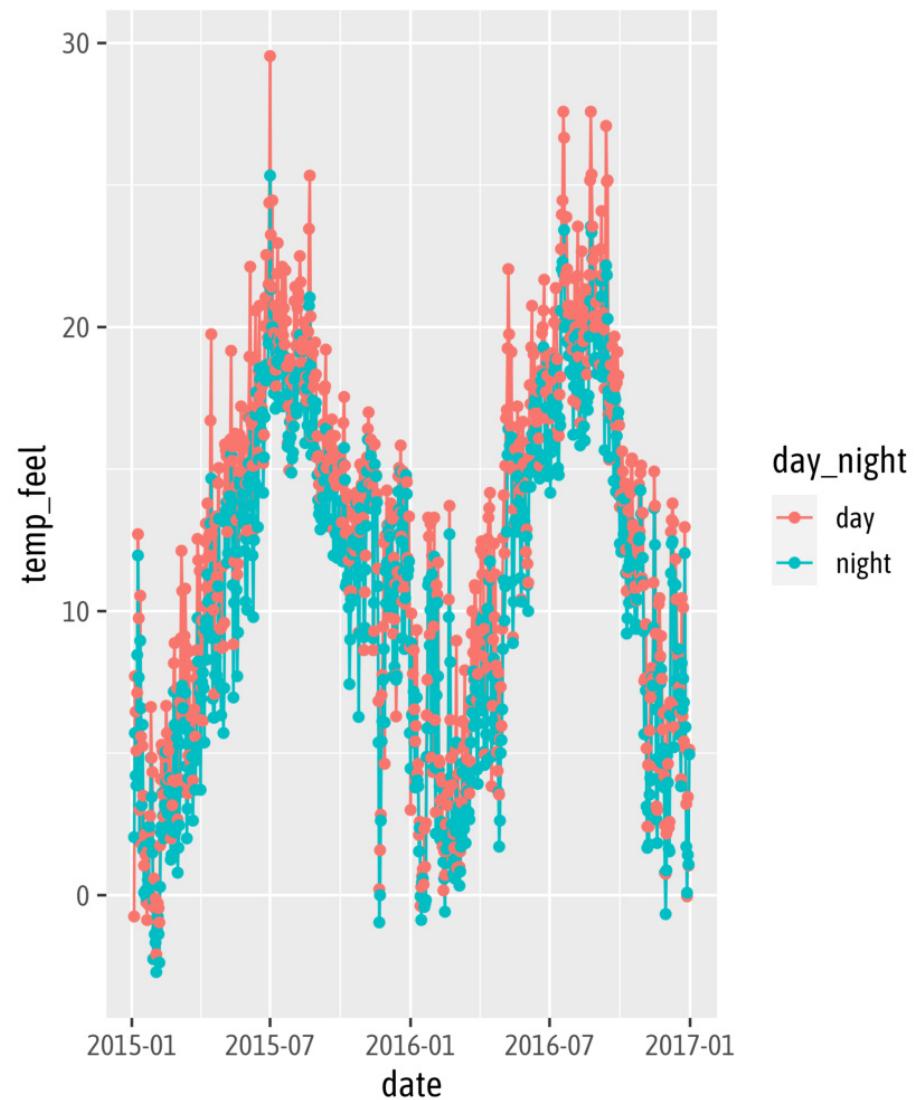
Your Turn: Geometries & Aesthetics

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



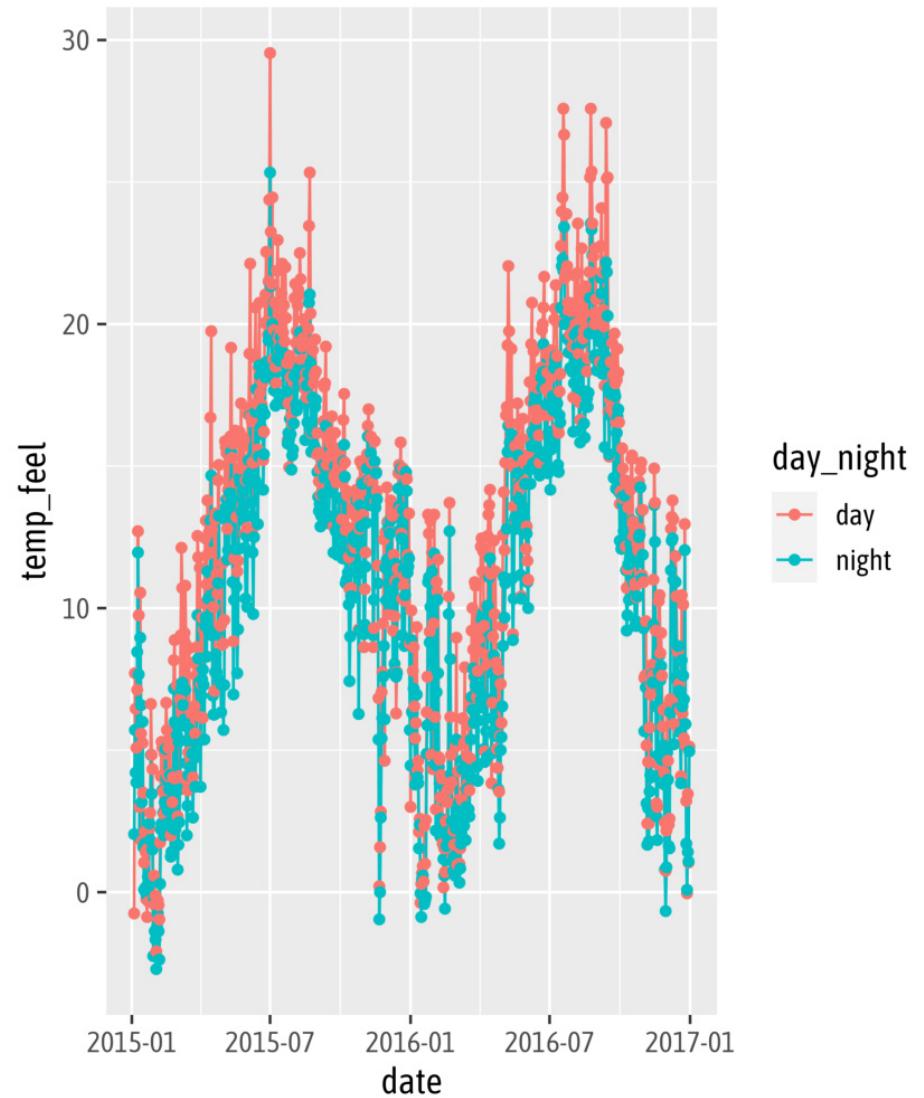
Your Turn: Geometries & Aesthetics

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



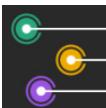
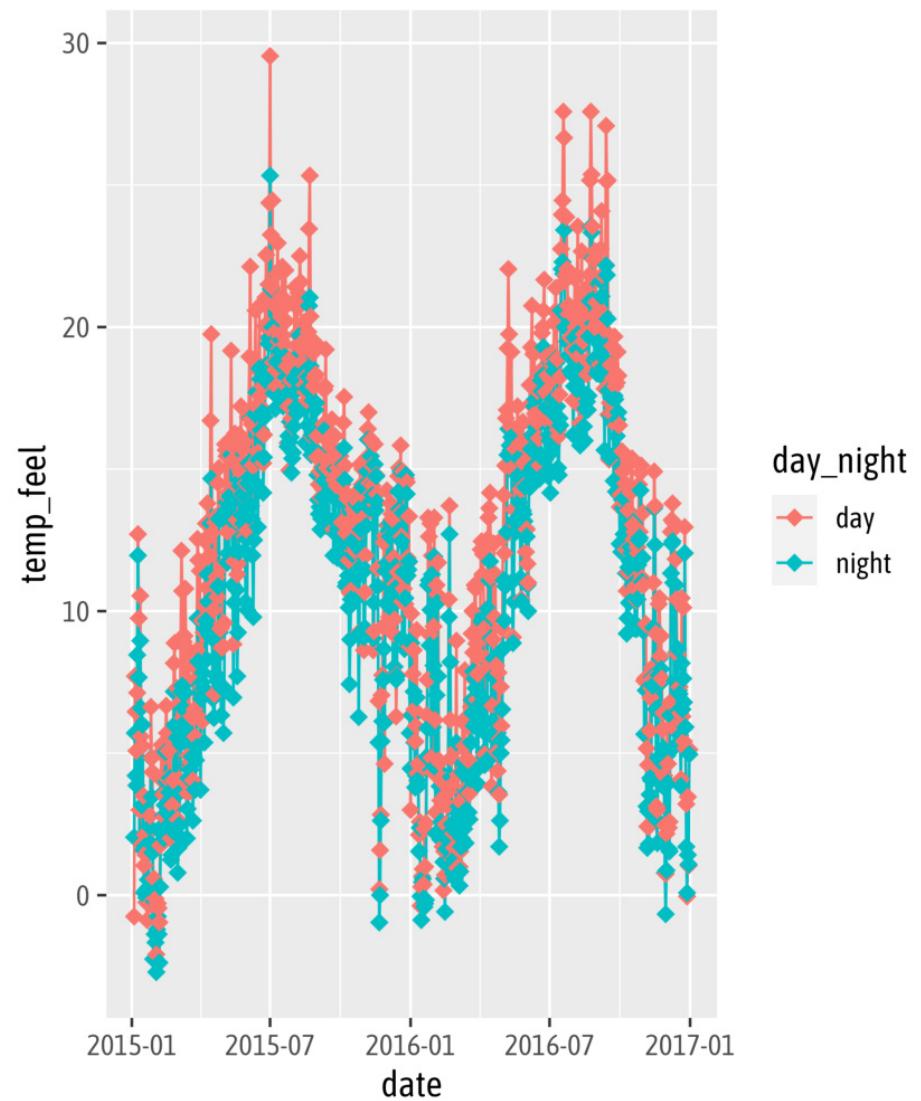
Your Turn: Geometries & Aesthetics

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



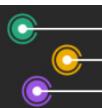
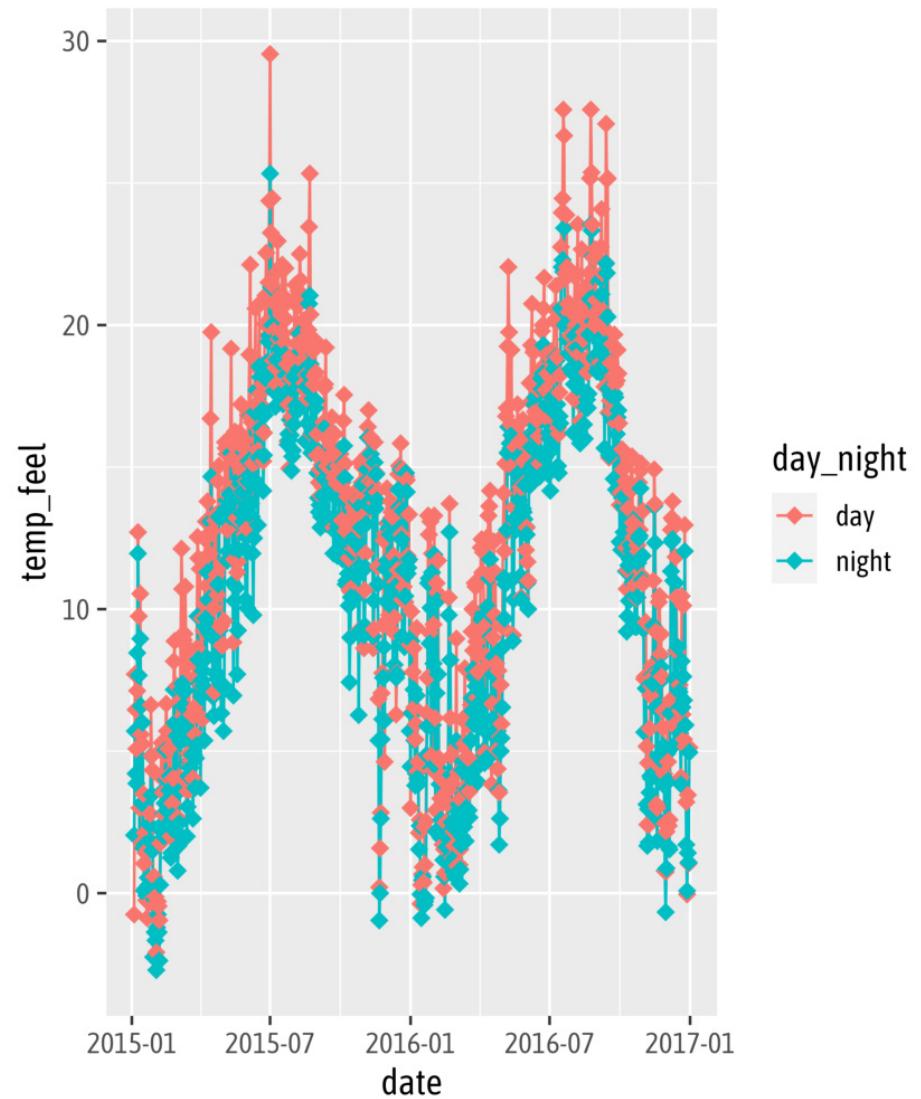
Your Turn: Geometries & Aesthetics

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



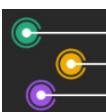
Your Turn: Geometries & Aesthetics

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



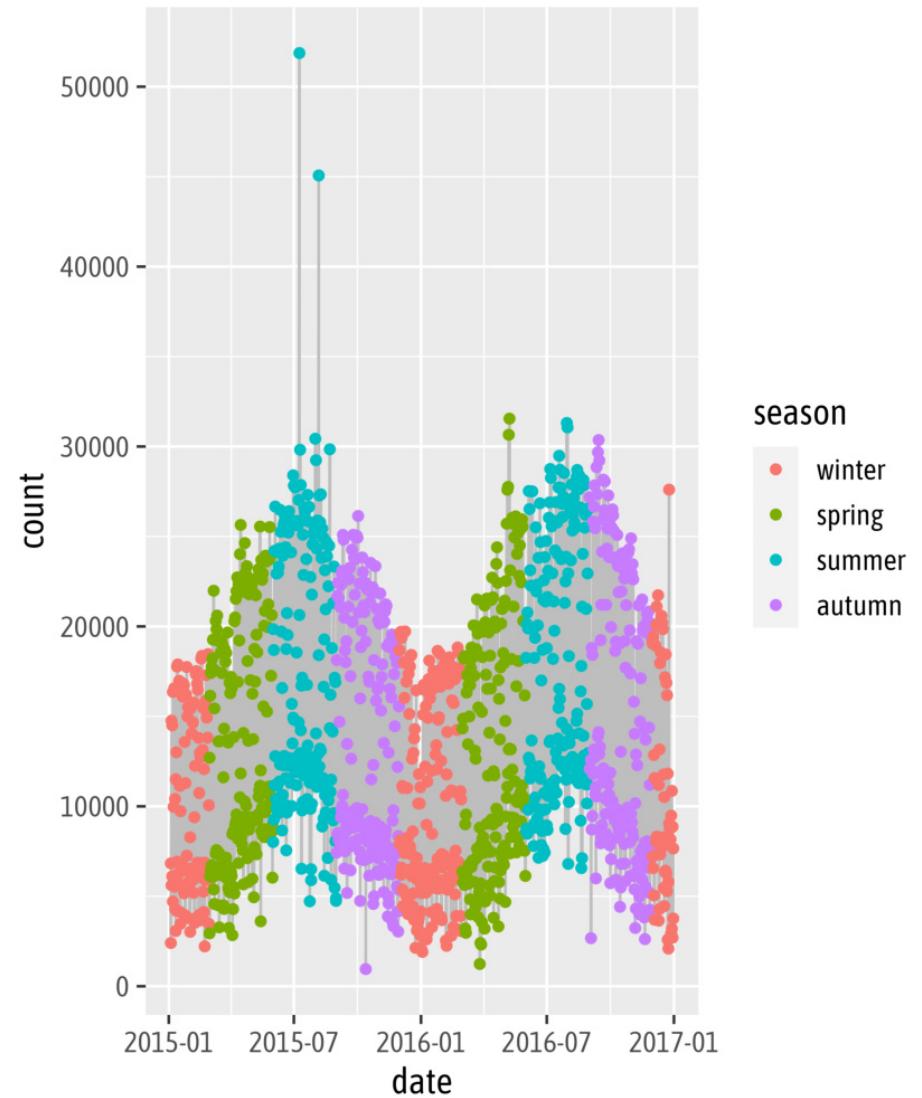


Source: Albert's Blog



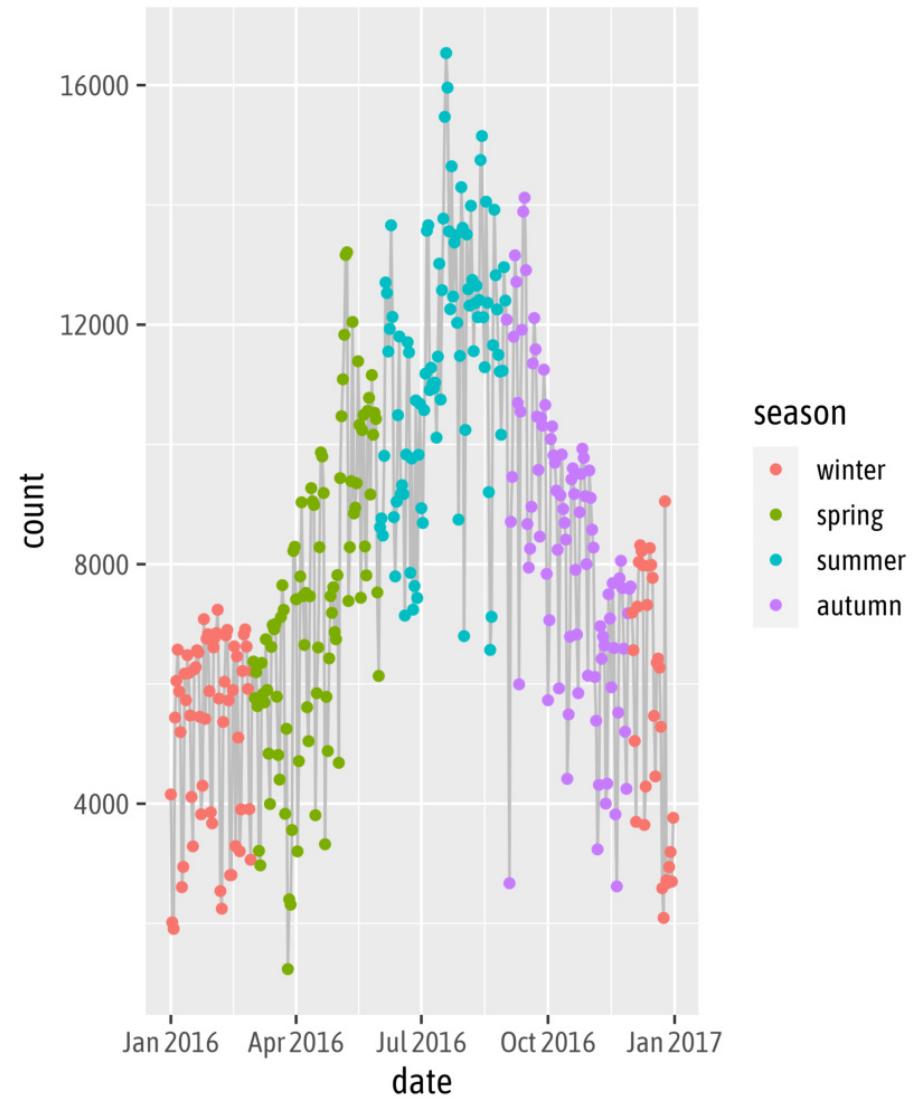
Your Turn: Bonus Exercise

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



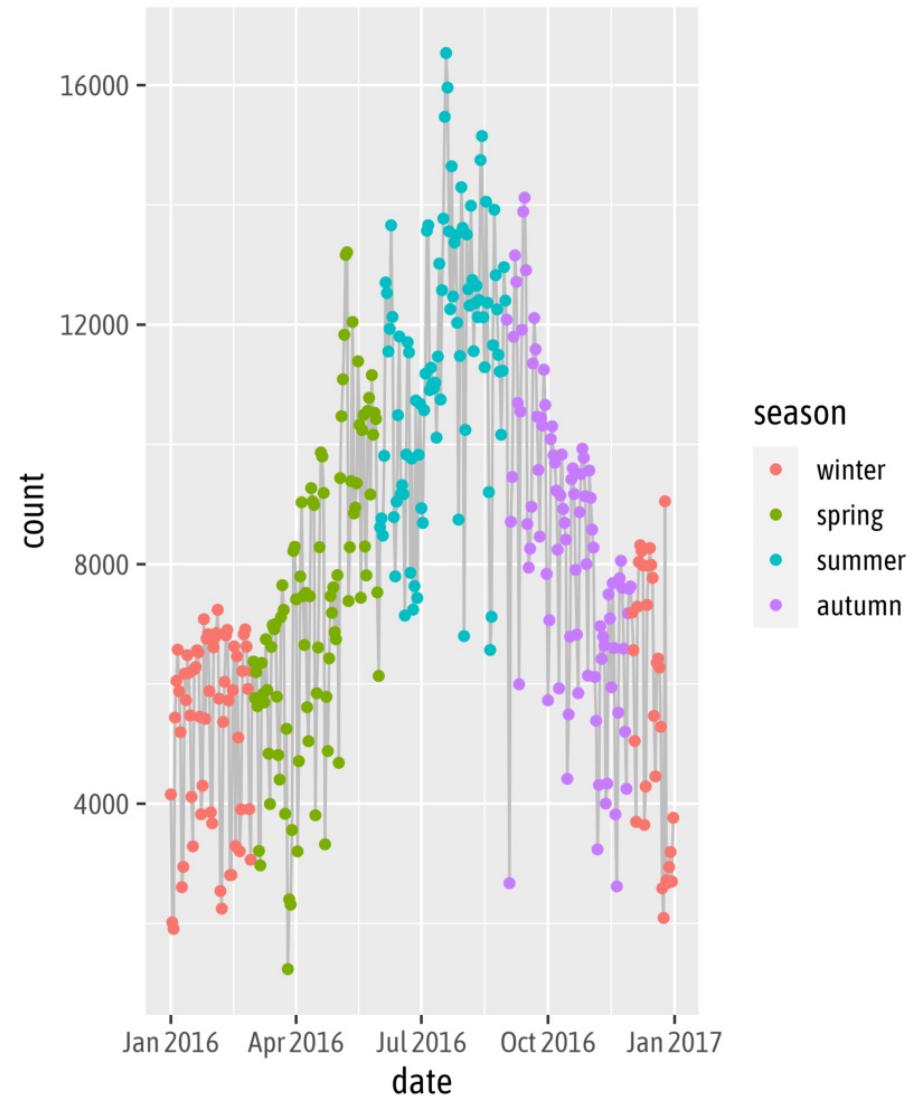
Your Turn: Bonus Exercise

```
1 ggplot(  
2   data = filter(  
3     bikes, year == 2016 &  
4       day_night == "night"  
5   ),  
6   aes(x = date, y = count))  
7 ) +  
8 geom_line(  
9   color = "grey")  
10 ) +  
11 geom_point(  
12   aes(color = season))  
13 )
```



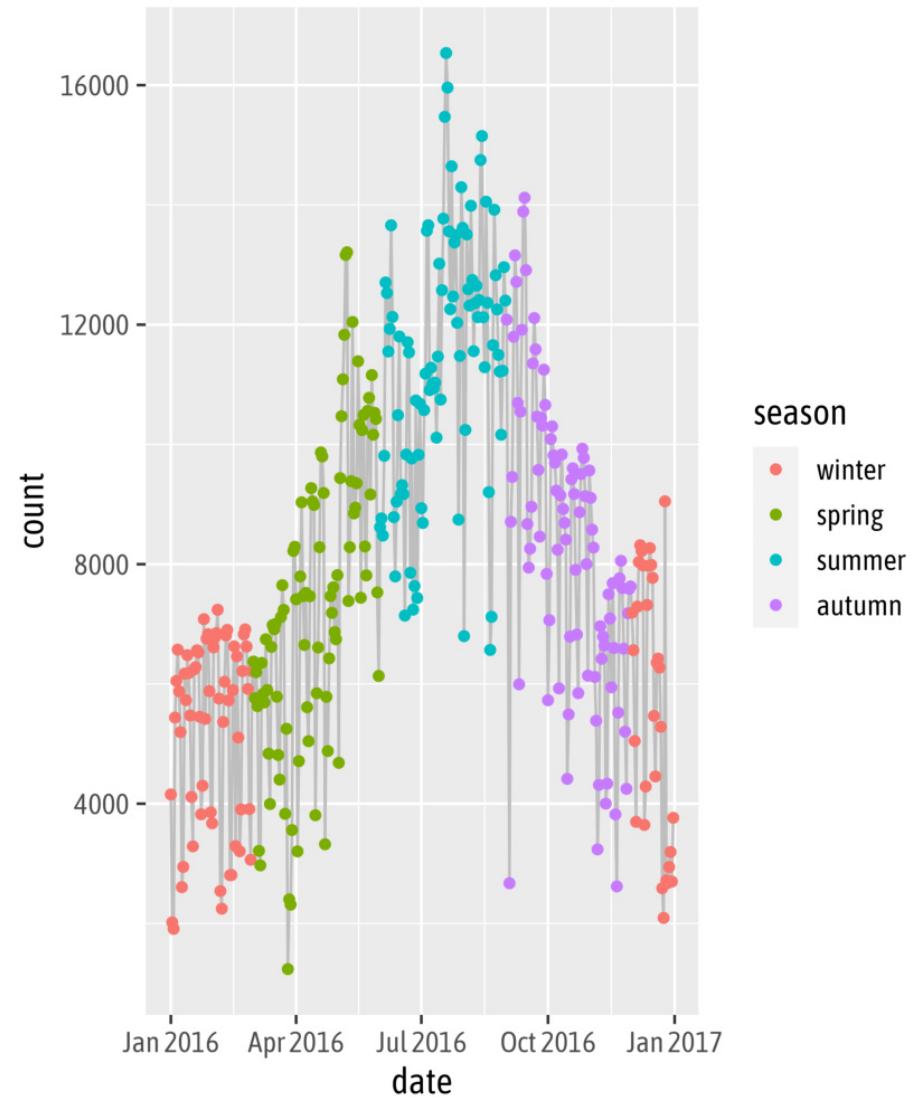
Your Turn: Bonus Exercise

```
1 bikes %>%
2   filter(
3     year == 2016 &
4     day_night == "night"
5   ) %>%
6   ggplot(
7     aes(x = date, y = count)
8   ) +
9   geom_line(
10    color = "grey"
11  ) +
12  geom_point(
13    aes(color = season)
14  )
```



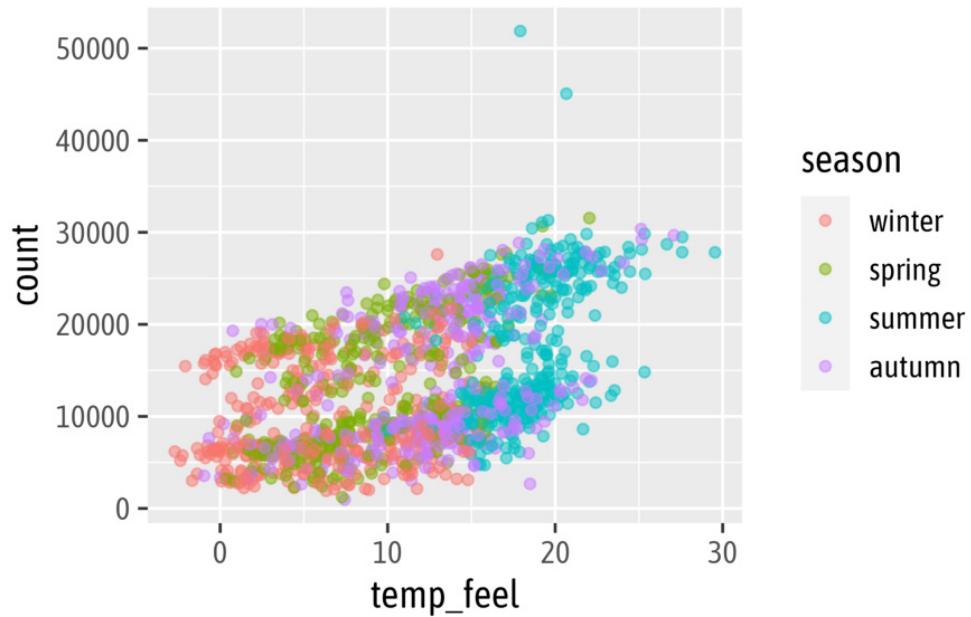
Your Turn: Bonus Exercise

```
1 bikes_sub <- filter(  
2   bikes,  
3   year == 2016 & day_night == "night"  
4 )  
5  
6 ggplot(  
7   data = bikes_sub,  
8   aes(x = date, y = count)  
9 ) +  
10 geom_line(  
11   color = "grey"  
12 ) +  
13 geom_point(  
14   aes(color = season)  
15 )
```

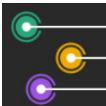
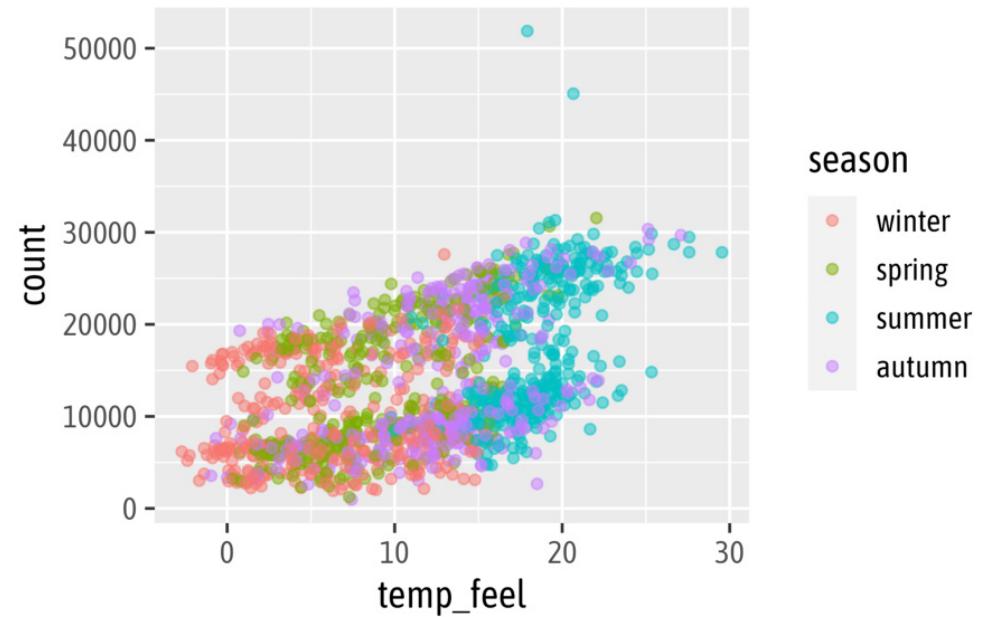


Local vs. Global Encoding

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

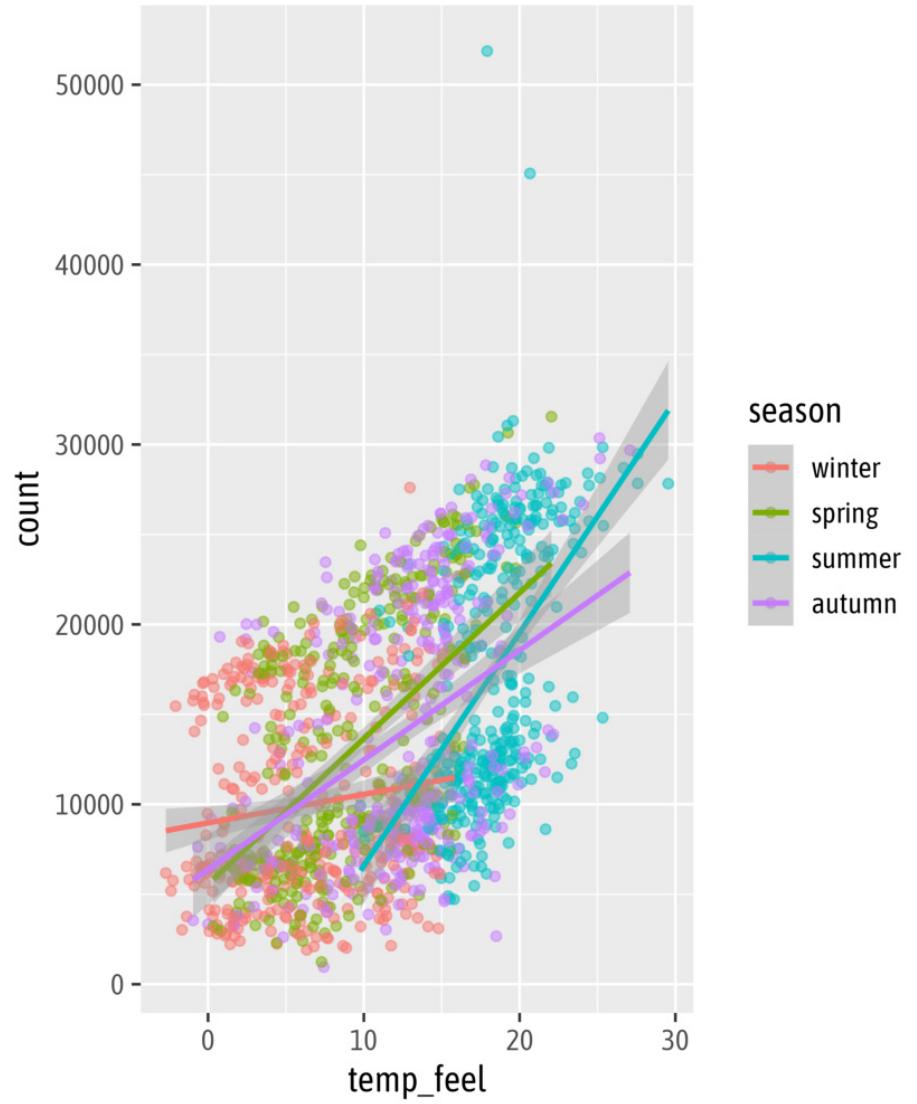


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4         color = season)  
5 ) +  
6   geom_point(  
7   alpha = .5  
8 )
```



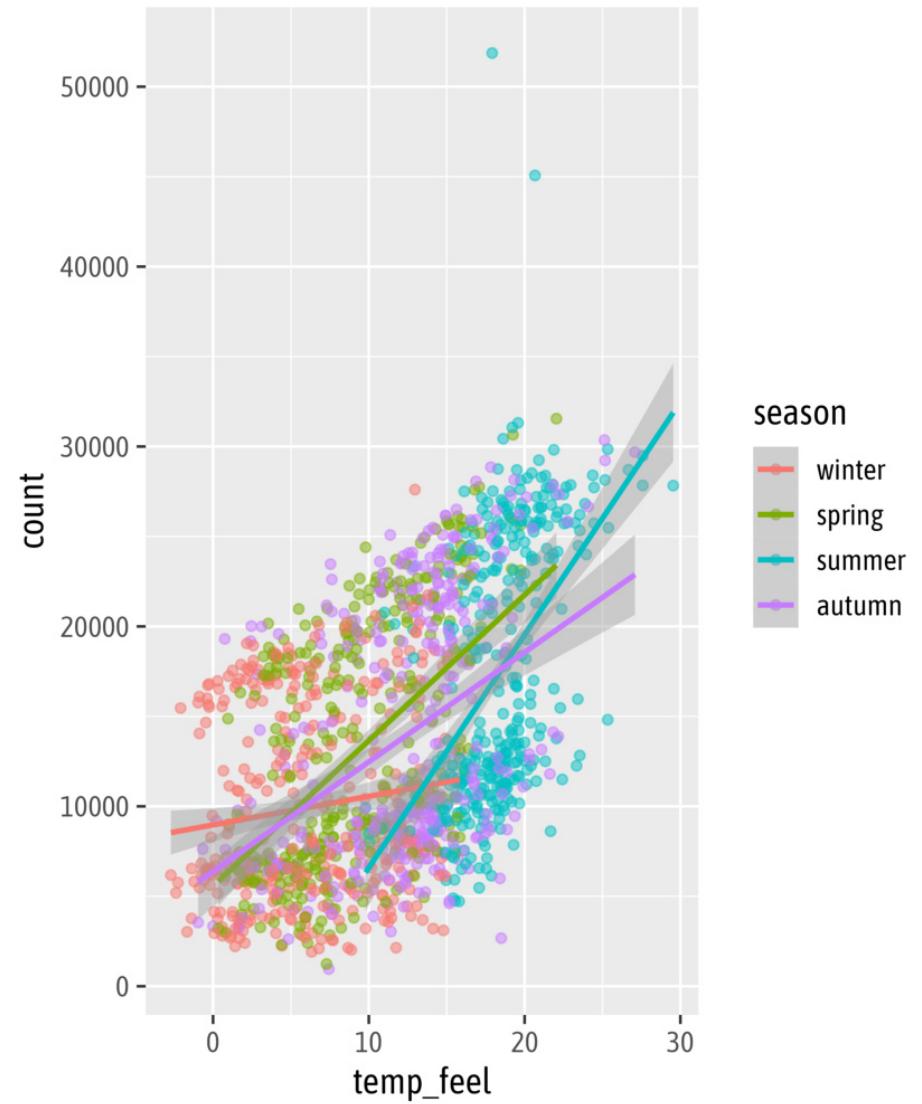
Adding More Layers

```
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 )
```



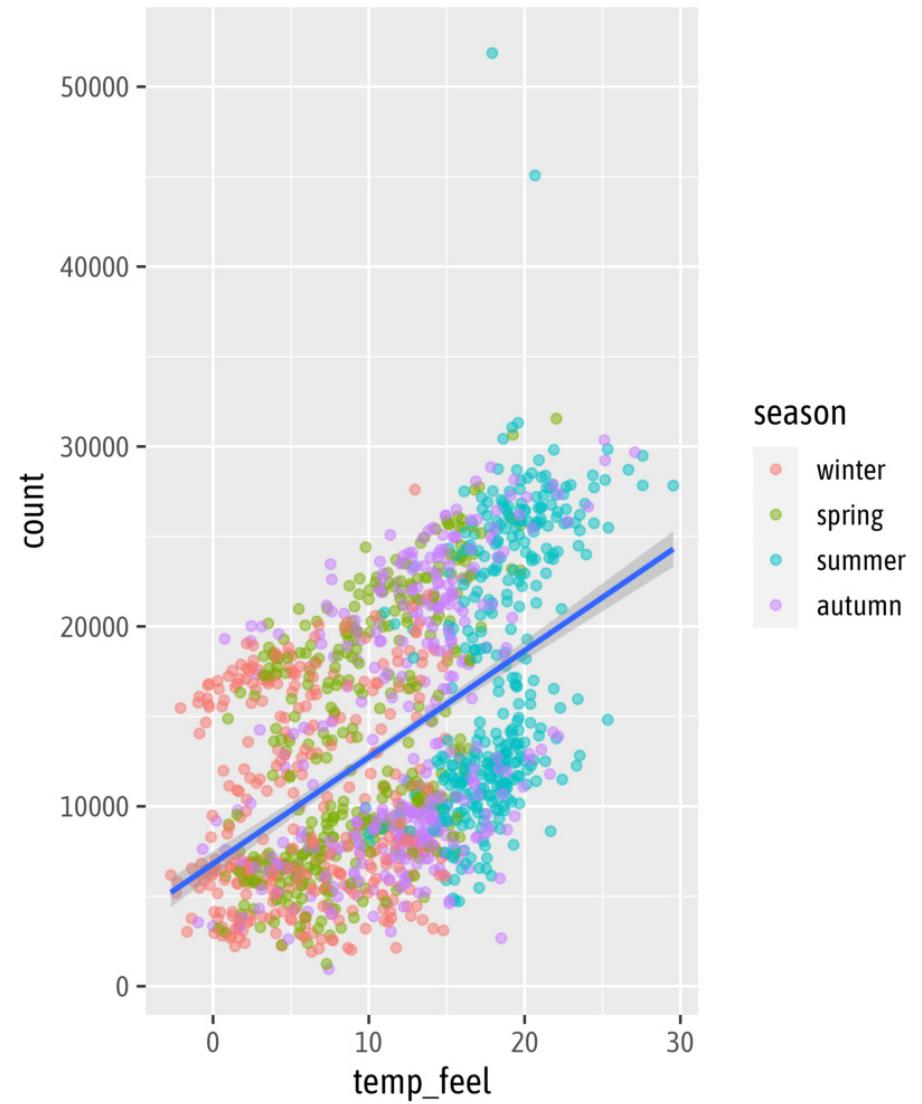
Global Color Encoding

```
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 )
```



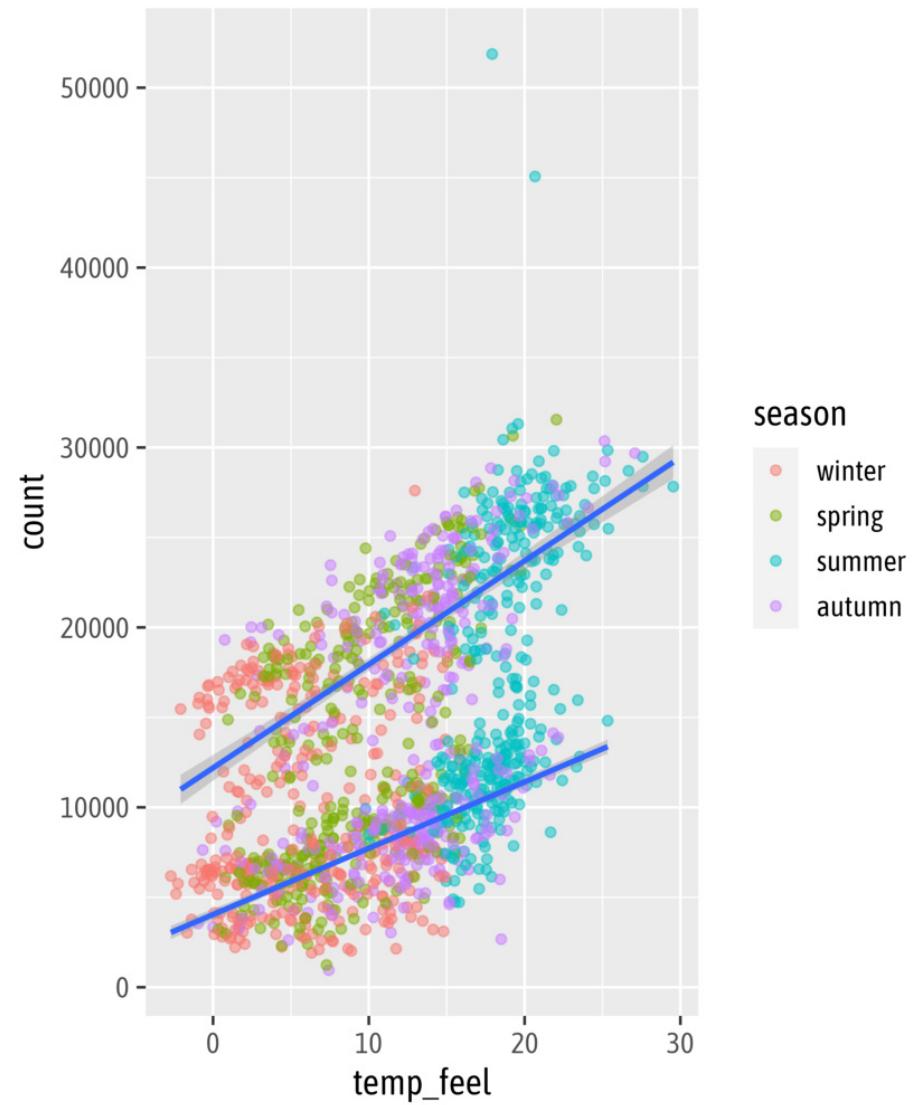
Local Color Encoding

```
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 )
```



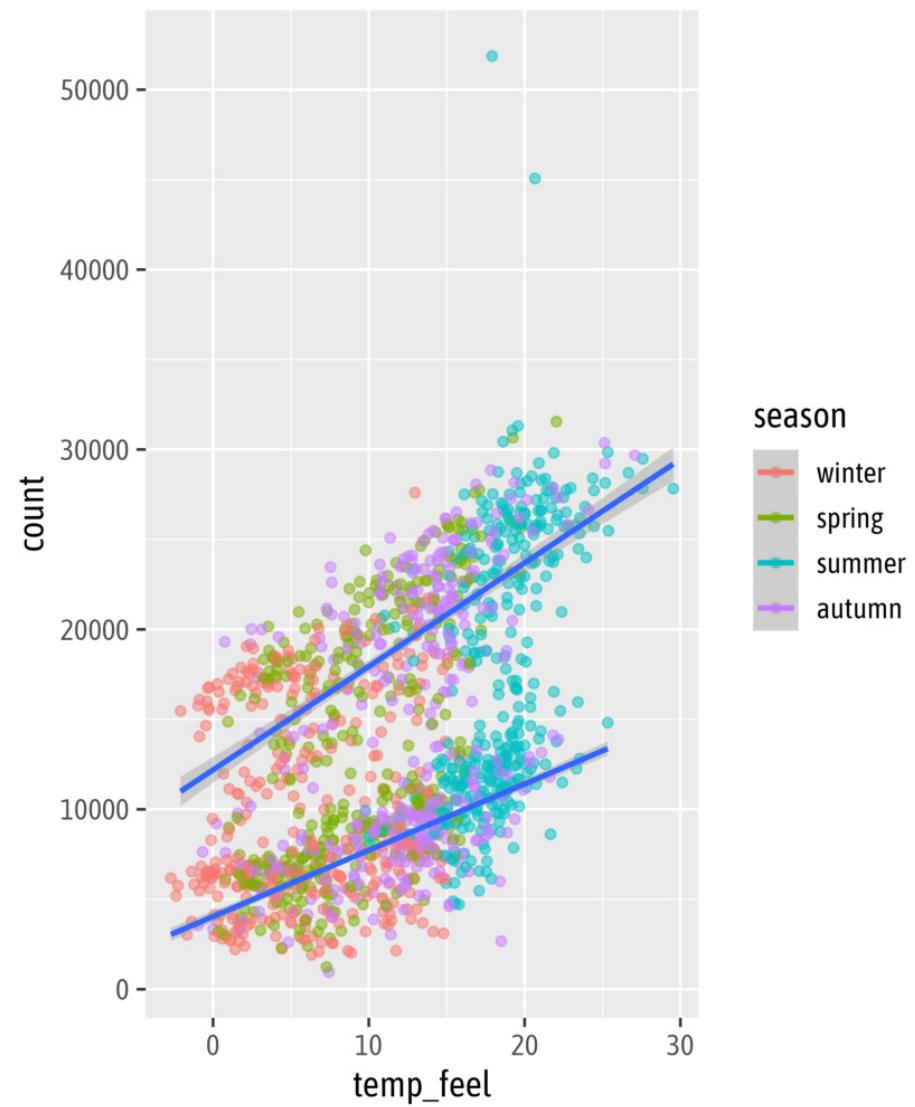
The `group` Aesthetic

```
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 )
```



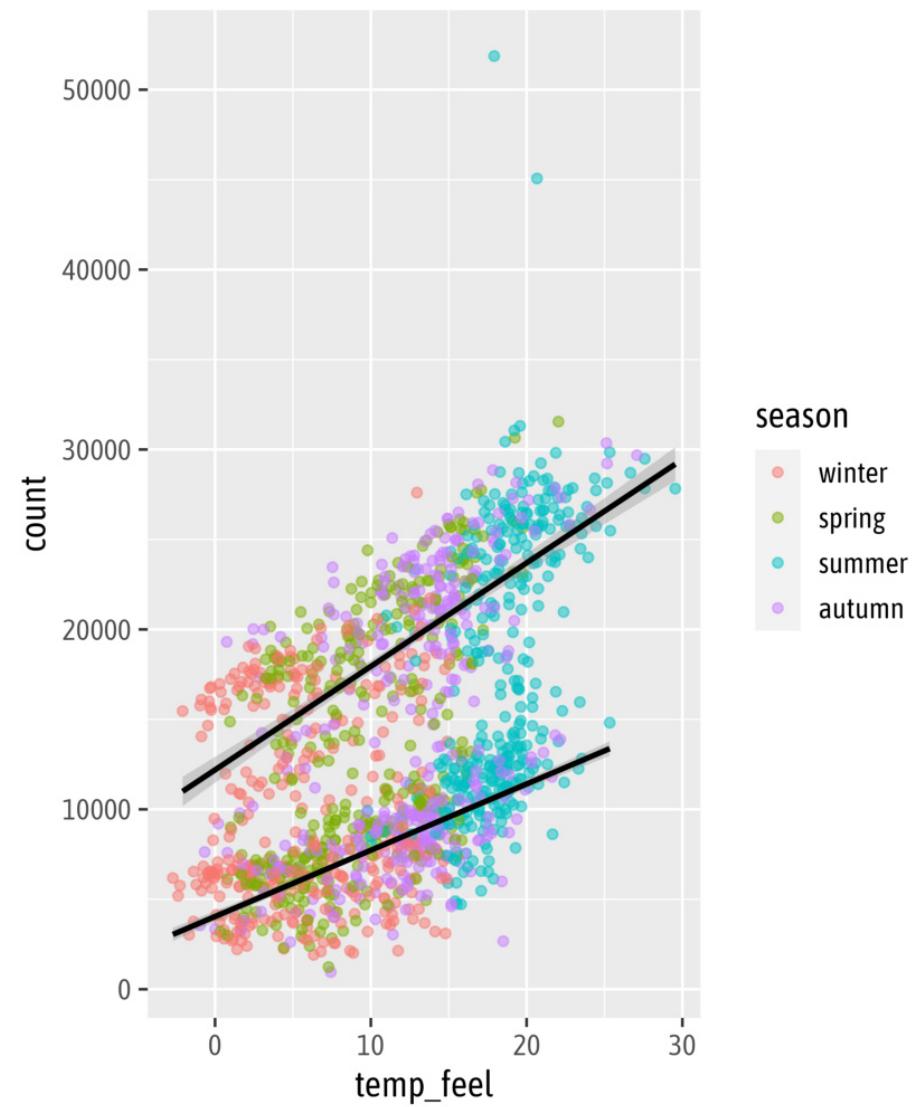
Set Both as Global Aesthetics

```
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 )
```



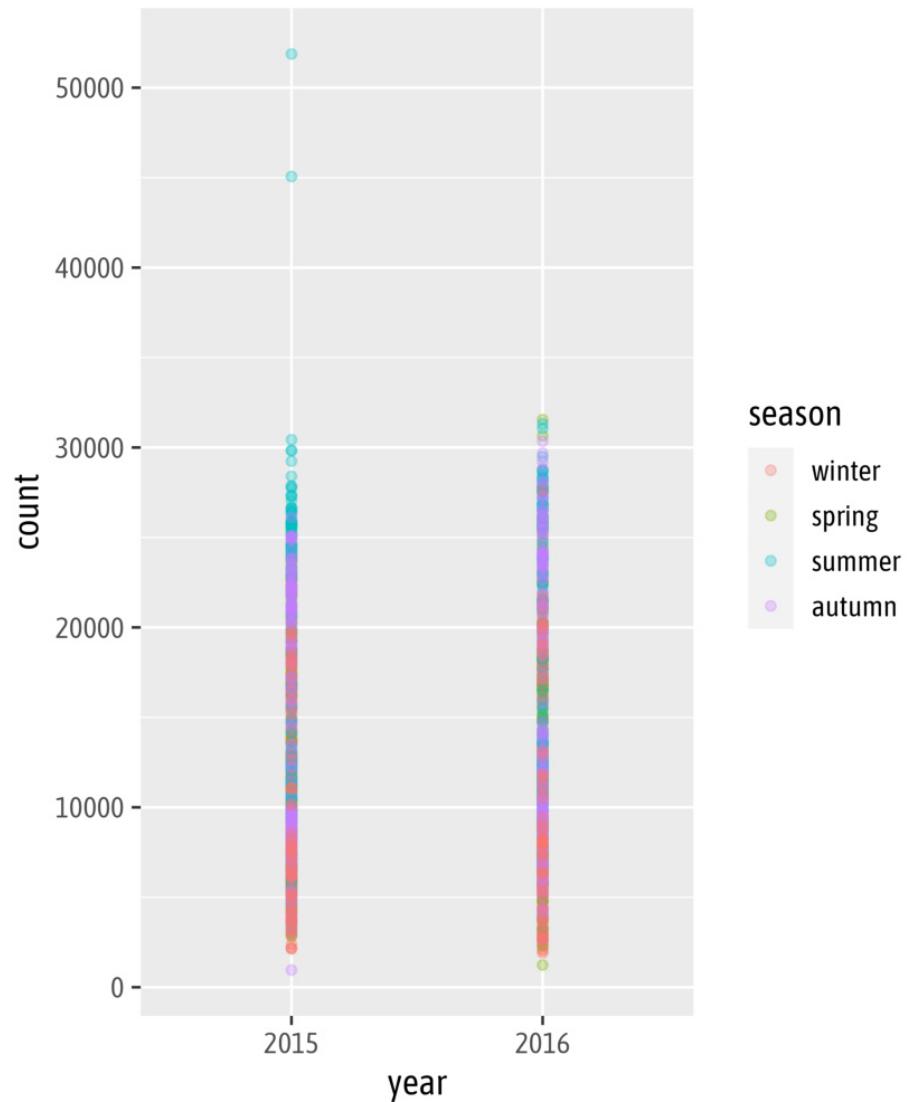
Overwrite Global Aesthetics

```
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 )
```



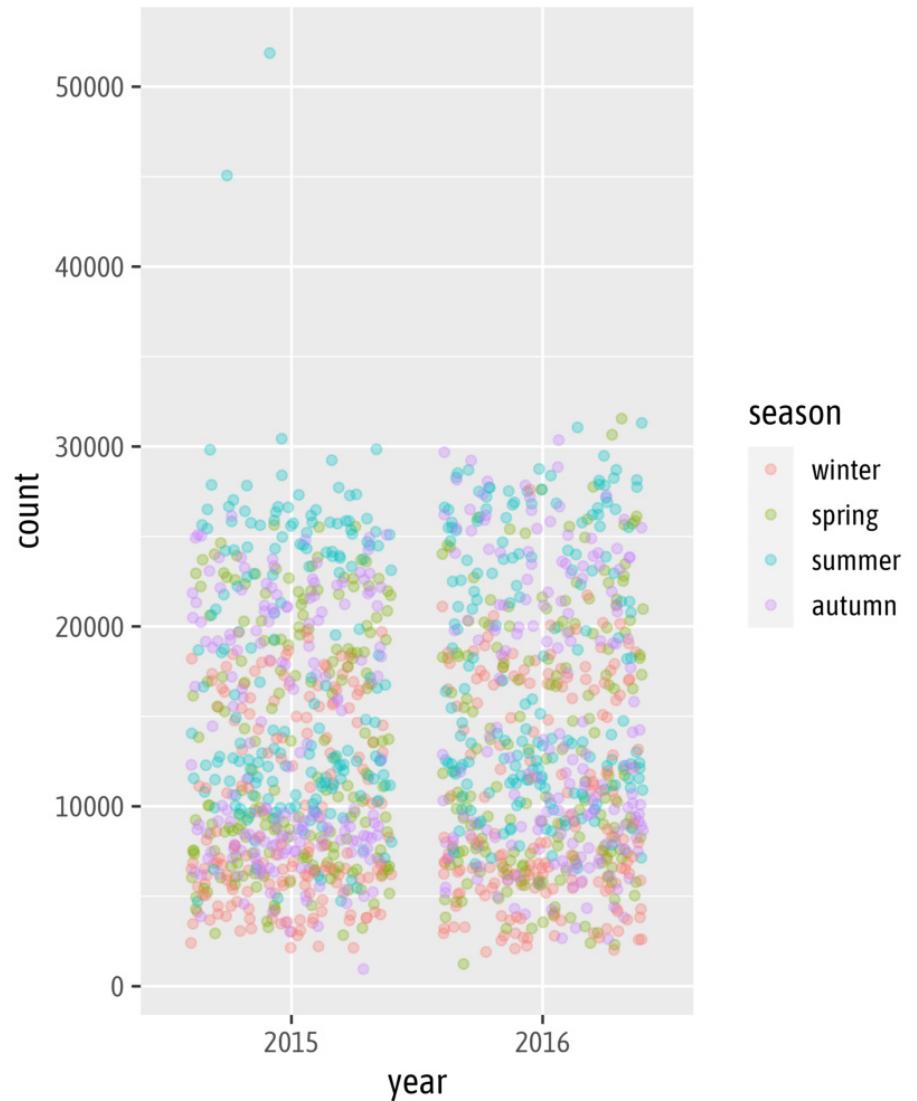
The position Argument

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



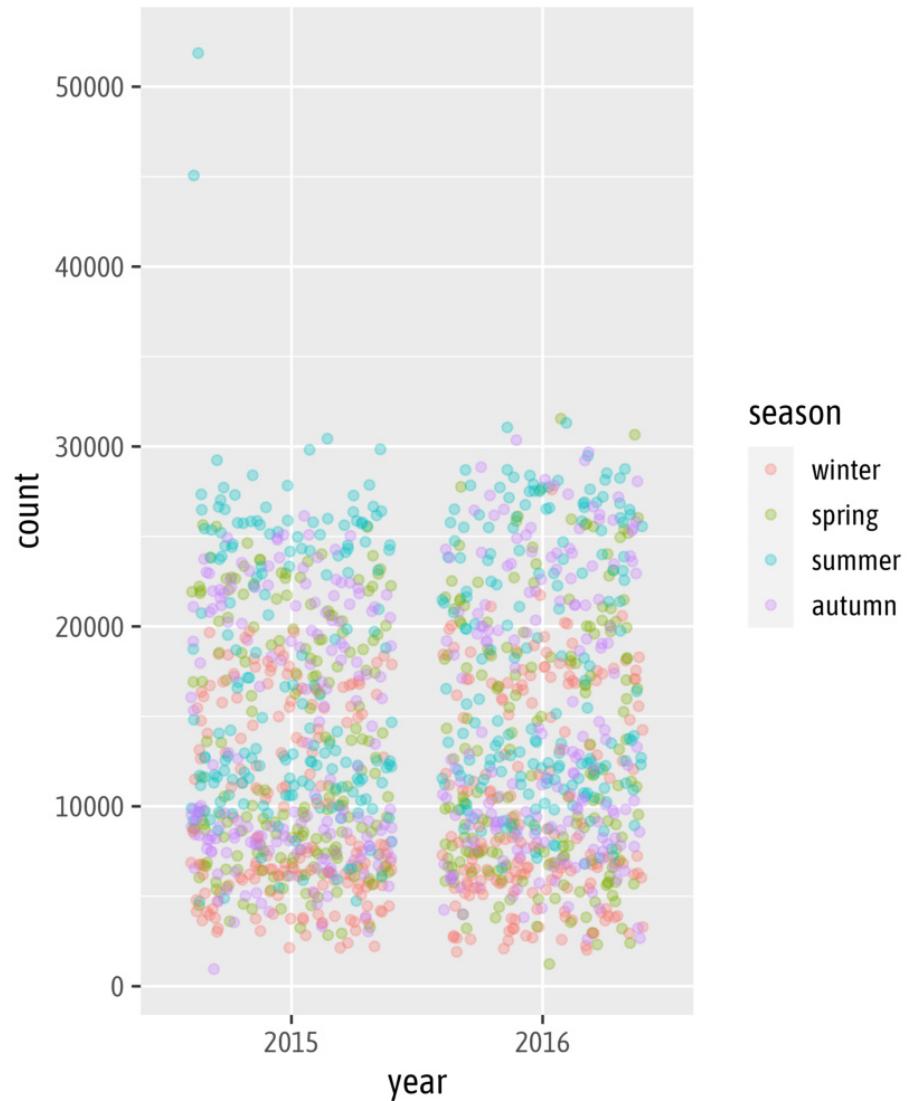
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_point(  
3     aes(color = season),  
4     alpha = .3,  
5     position = "jitter"  
6   )
```



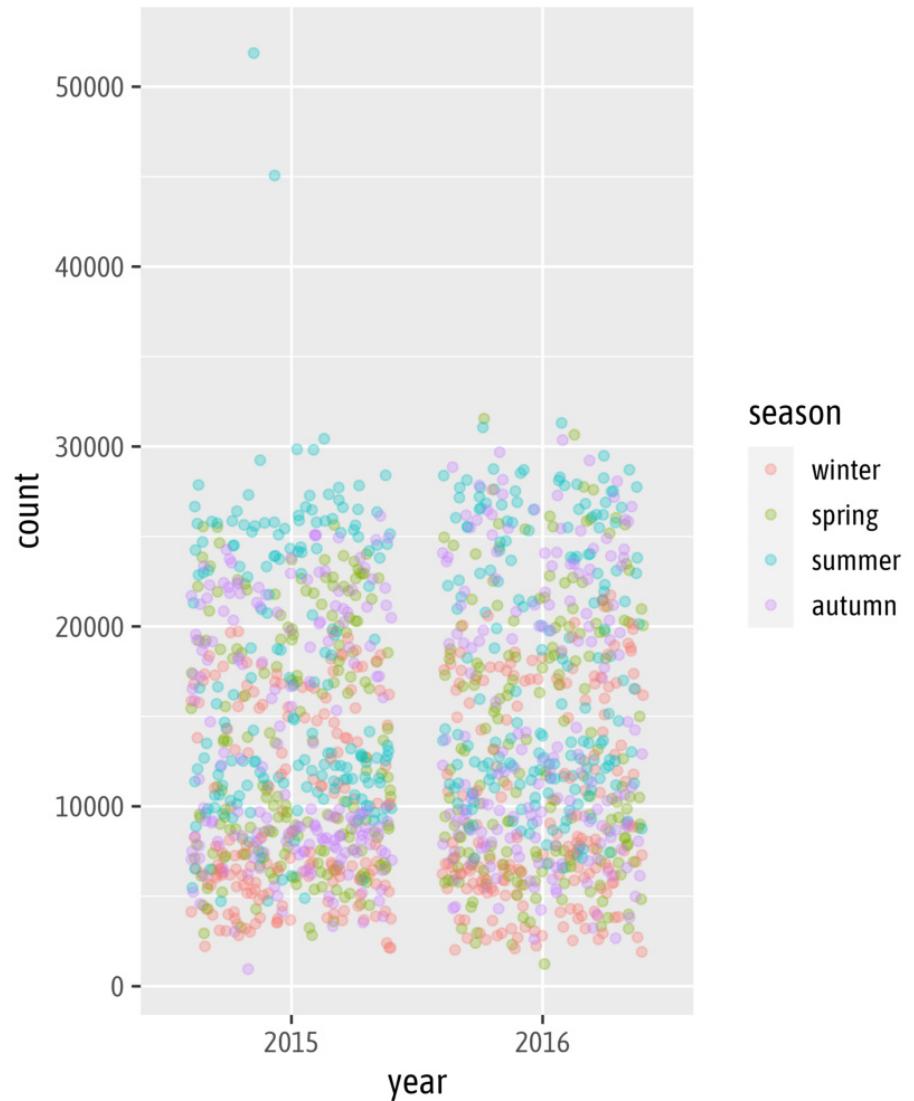
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_point(  
3     aes(color = season),  
4     alpha = .3,  
5     position = position_jitter()  
6   )
```



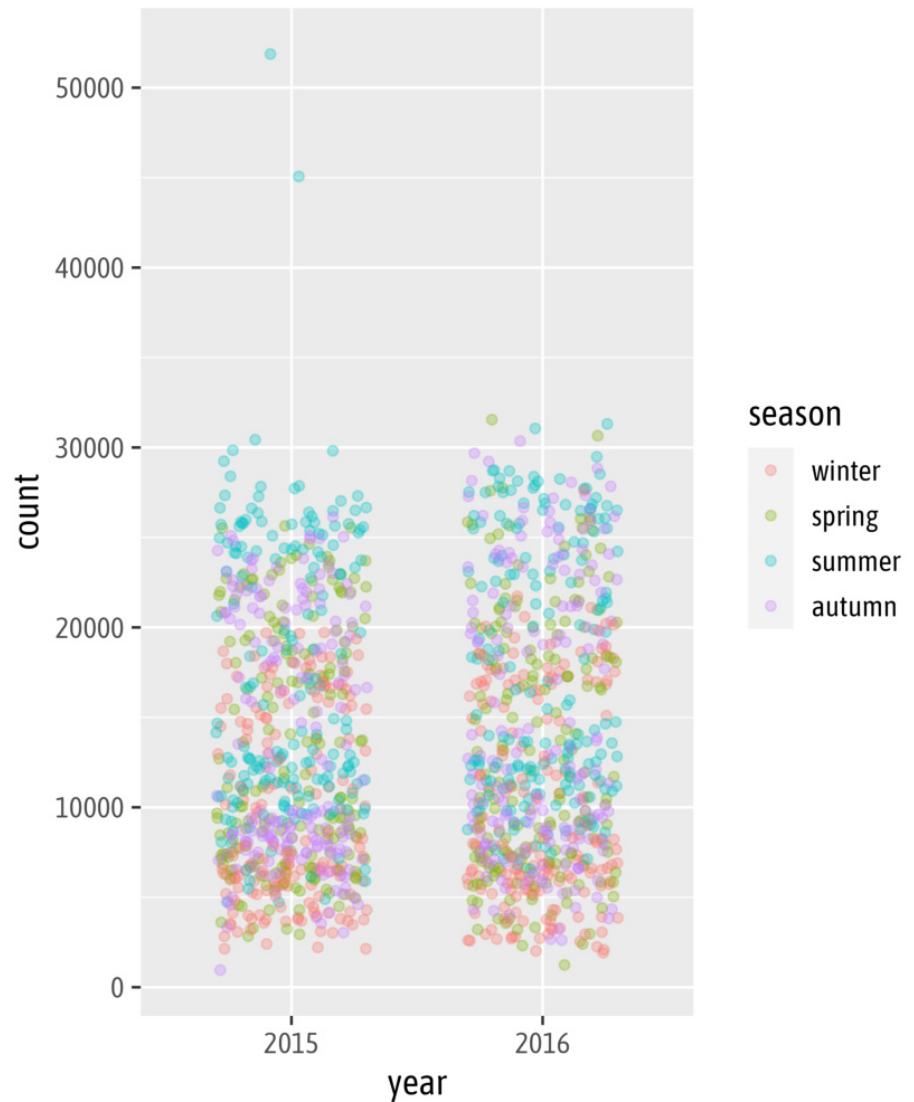
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_jitter(  
3     aes(color = season),  
4     alpha = .3  
5   )
```



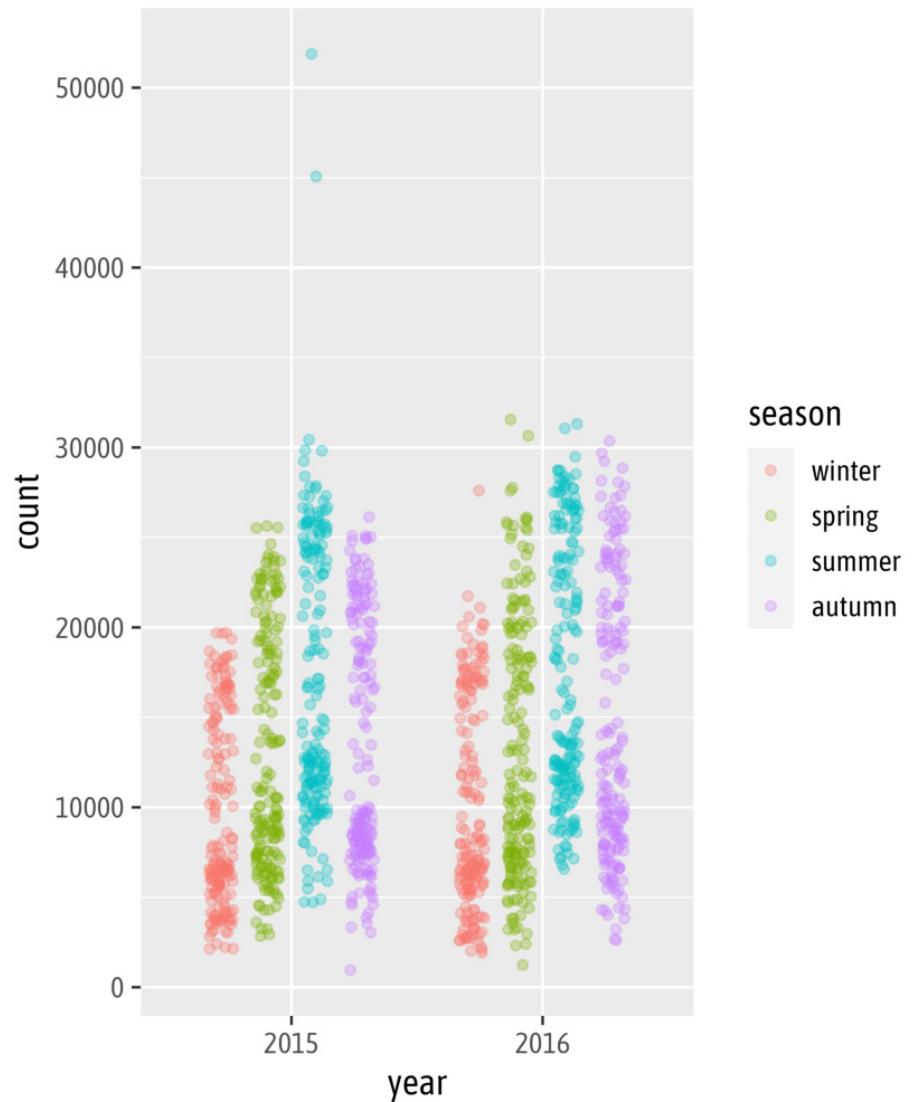
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_point(  
3     aes(color = season),  
4     alpha = .3,  
5     position = position_jitter(  
6       width = .3, seed = 2023  
7     )  
8   )
```



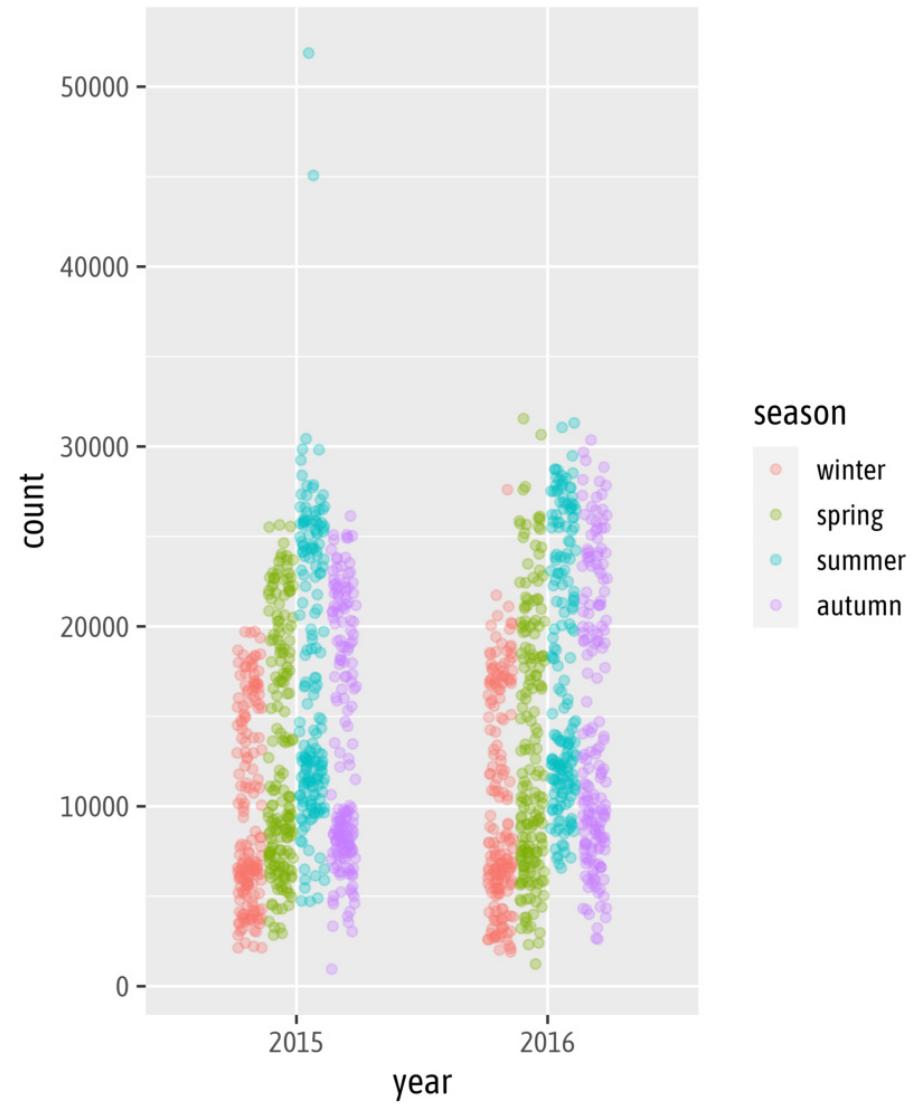
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_point(  
3     aes(color = season),  
4     alpha = .3,  
5     position = position_jitterdodge(  
6       jitter.width = .3, seed = 2023  
7     )  
8   )
```



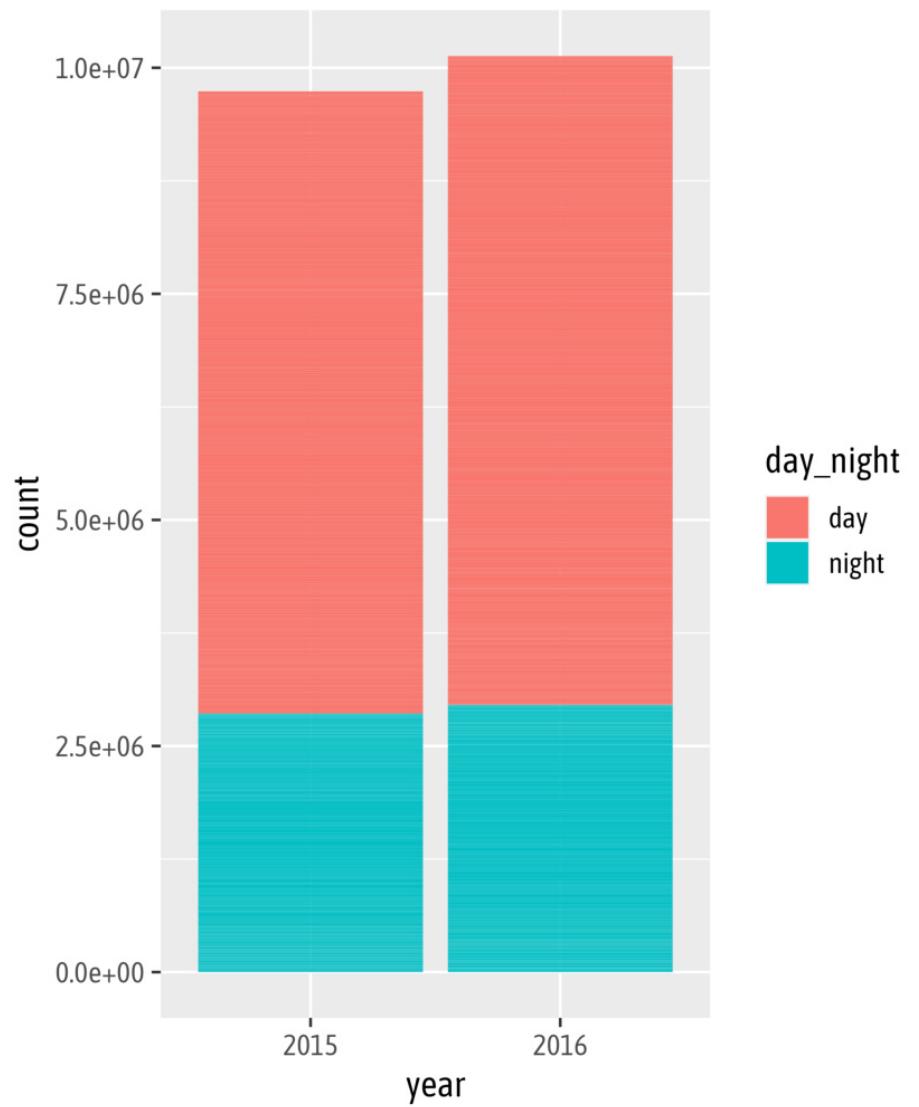
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_point(  
3     aes(color = season),  
4     alpha = .3,  
5     position = position_jitterdodge(  
6       jitter.width = .3, seed = 2023,  
7       dodge.width = .5  
8     )  
9   )
```



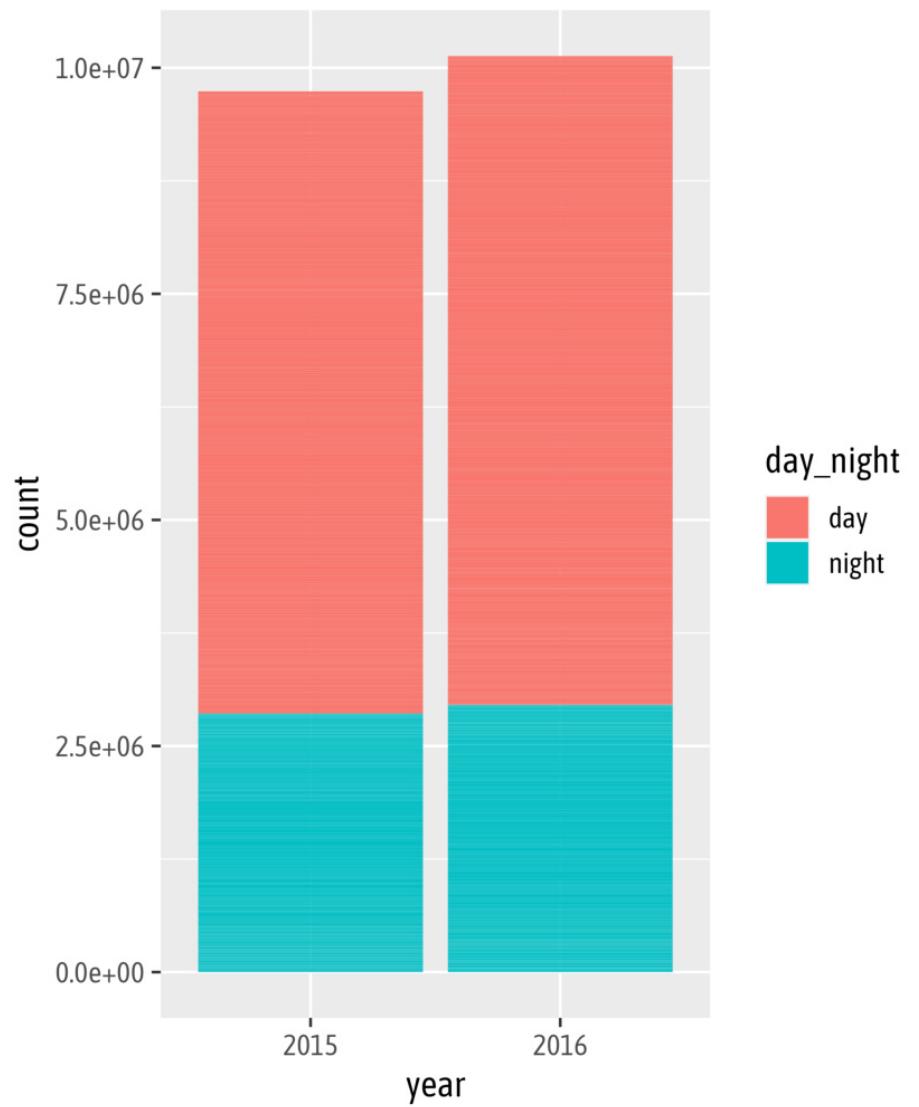
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_col(  
3     aes(fill = day_night)  
4   )
```



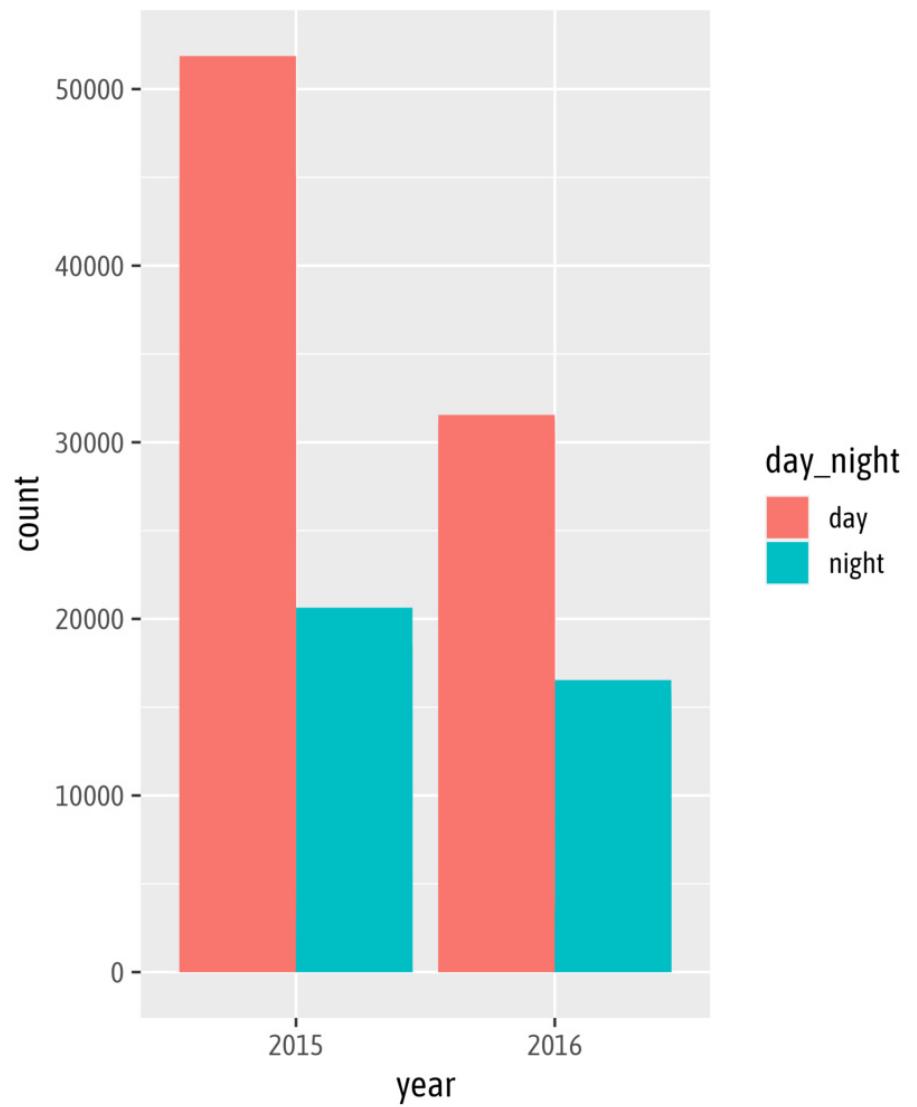
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_col(  
3     aes(fill = day_night),  
4     position = "stack"  
5   )
```



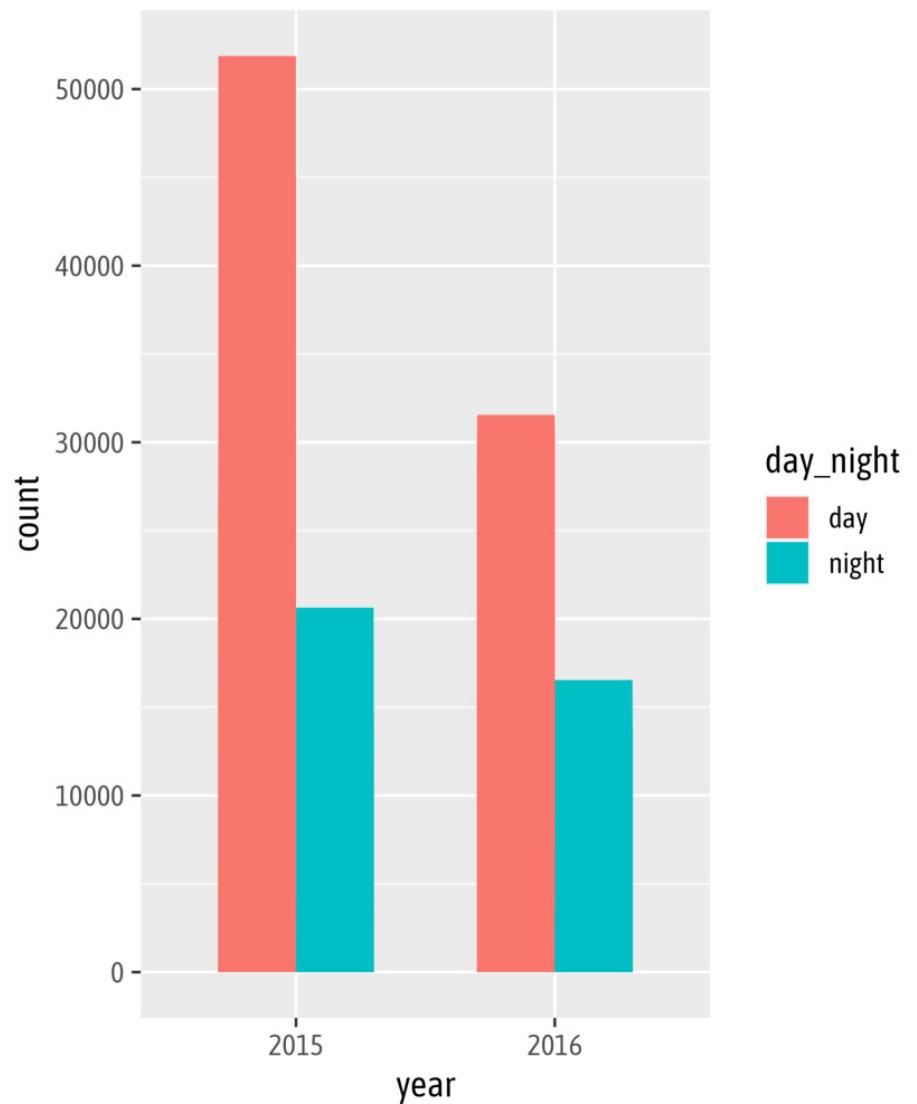
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_col(  
3     aes(fill = day_night),  
4     position = "dodge"  
5   )
```



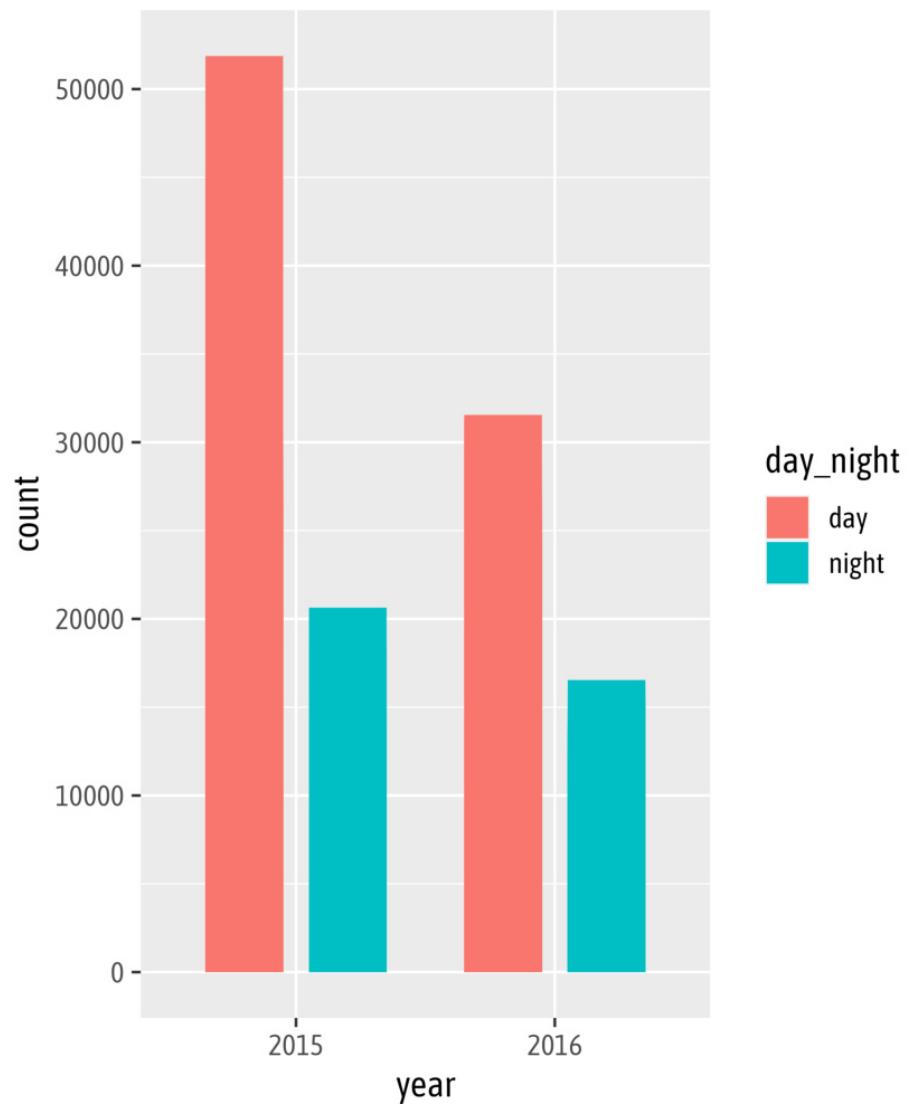
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_col(  
3     aes(fill = day_night),  
4     width = .6,  
5     position = "dodge"  
6   )
```



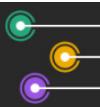
The position Argument

```
1 ggplot(bikes, aes(x = year, y = count)) +  
2   geom_col(  
3     aes(fill = day_night),  
4     width = .6,  
5     position = position_dodge(  
6       width = .8 ## by default same as col_w  
7     )  
8   )
```



Data Visualization with {ggplot2}

— Export Your Graphics —



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

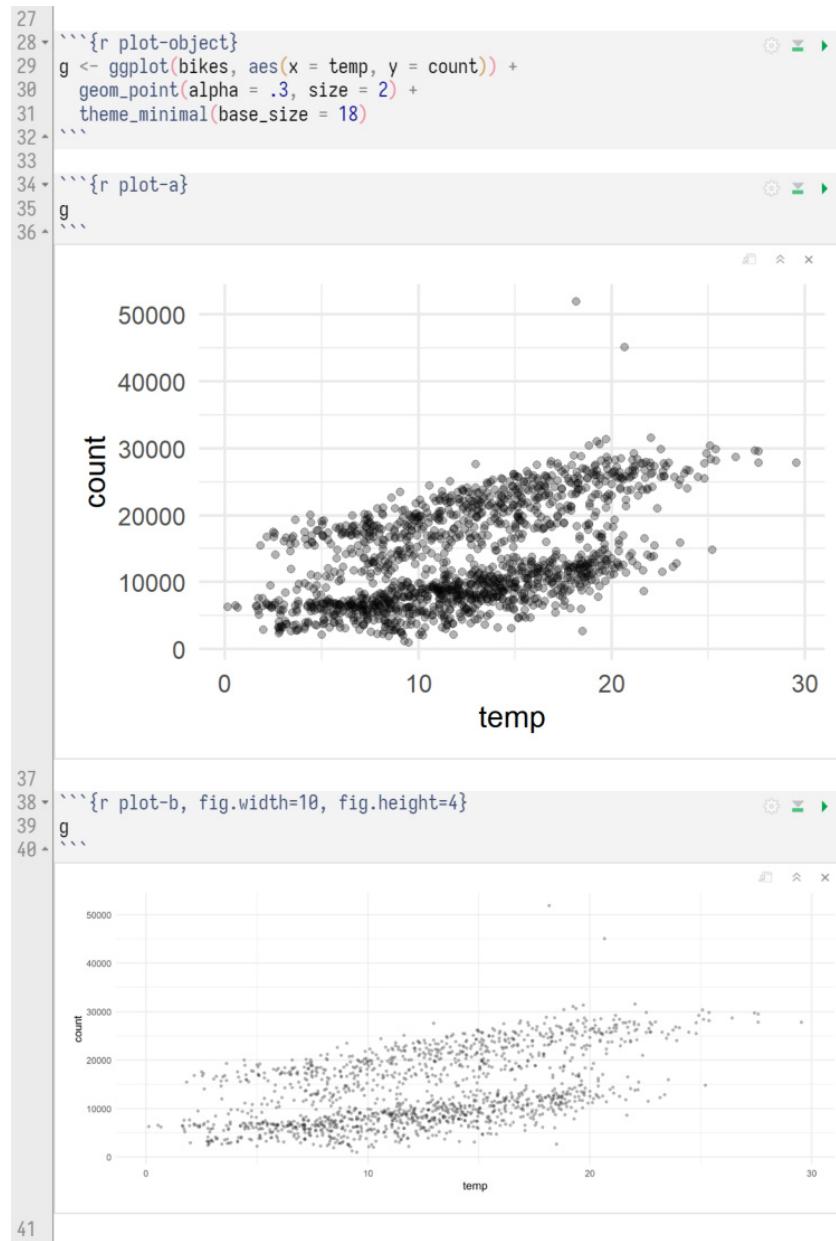


How to Work with Aspect Ratios

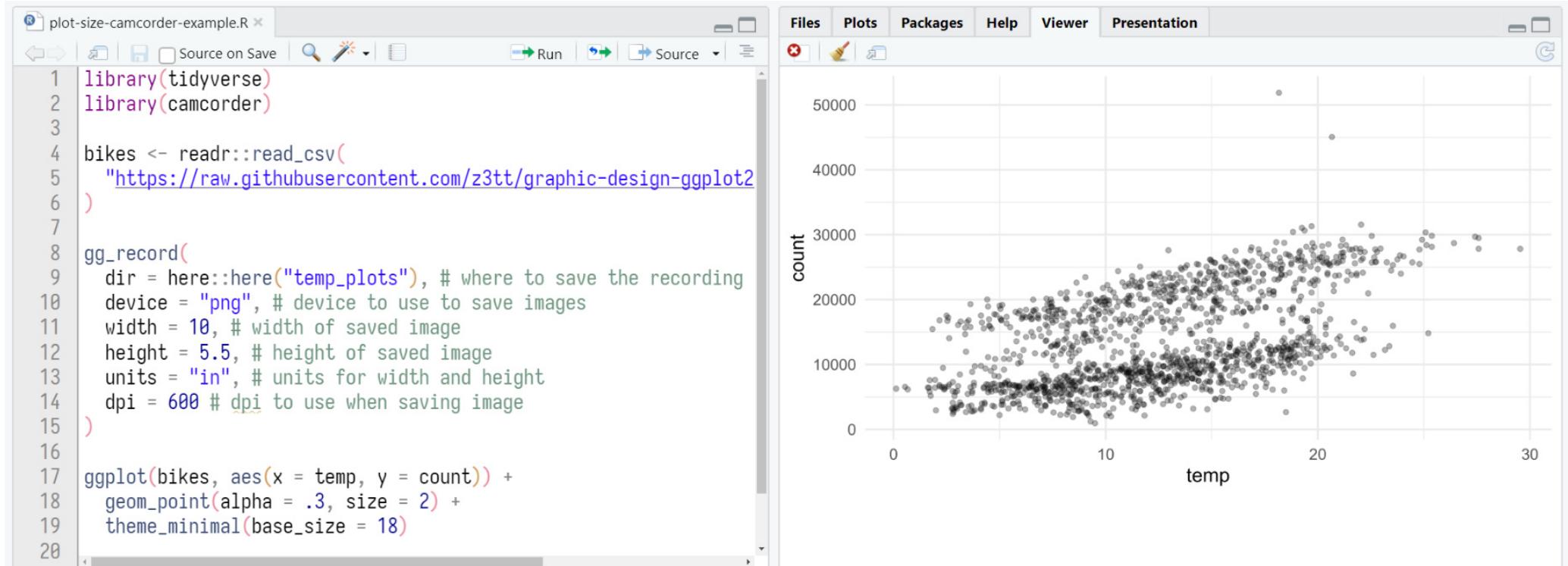
- don't rely on the Rstudio viewer pane!
- once you have a "it's getting close" prototype, settle on a plot size
- **Approach 1:** save the file to disk and inspect it; go back to your IDE
 - tedious and time-consuming...
- **Approach 2:** use a qmd or rmd with inline output and chunk settings
 - set `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

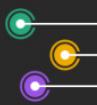


Setting Plot Sizes via {camcorder}



Data Visualization with {ggplot2}

— Extending a ggplot —



Store a ggplot as Object

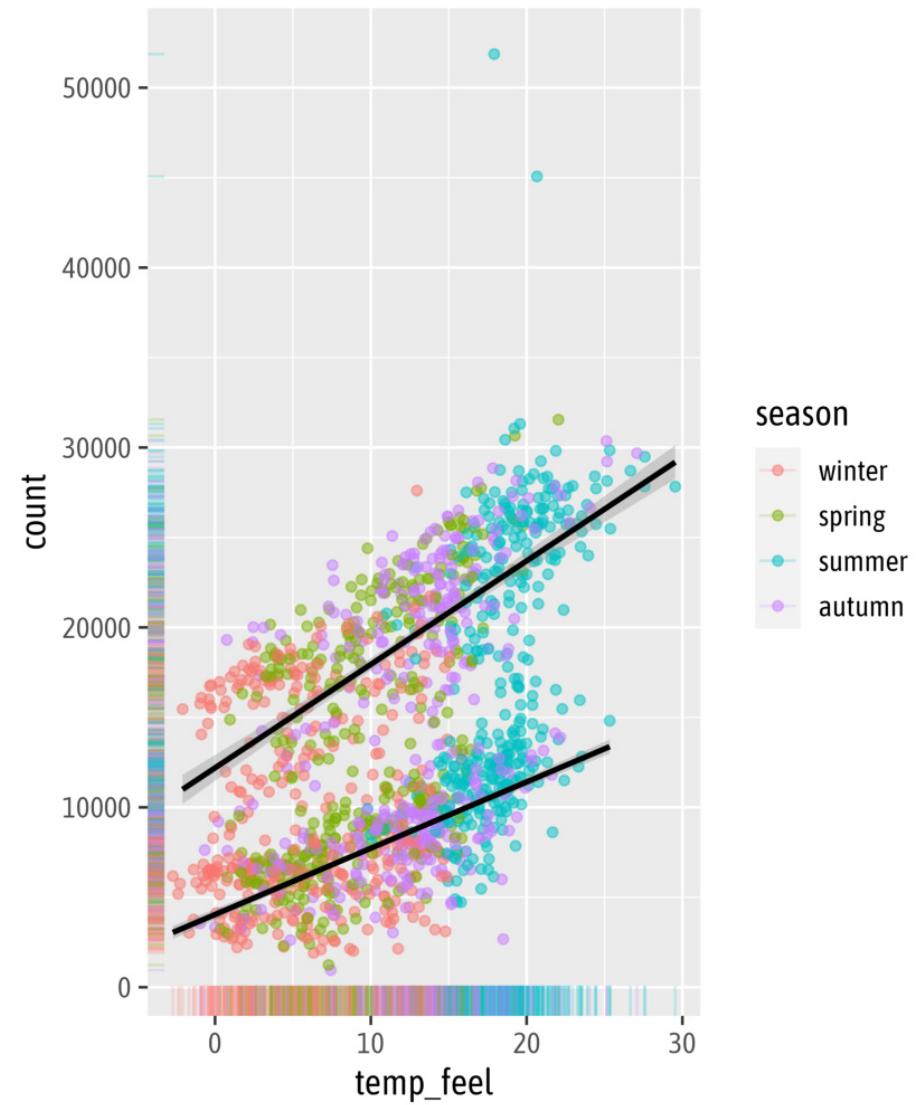
```
1 g <-  
2   ggplot(  
3     bikes,  
4     aes(x = temp_feel, y = count,  
5           color = season,  
6           group = day_night)  
7   ) +  
8   geom_point(  
9     alpha = .5  
10  ) +  
11  geom_smooth(  
12    method = "lm",  
13    color = "black"  
14  )  
15  
16 class(g)
```

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



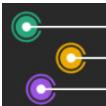
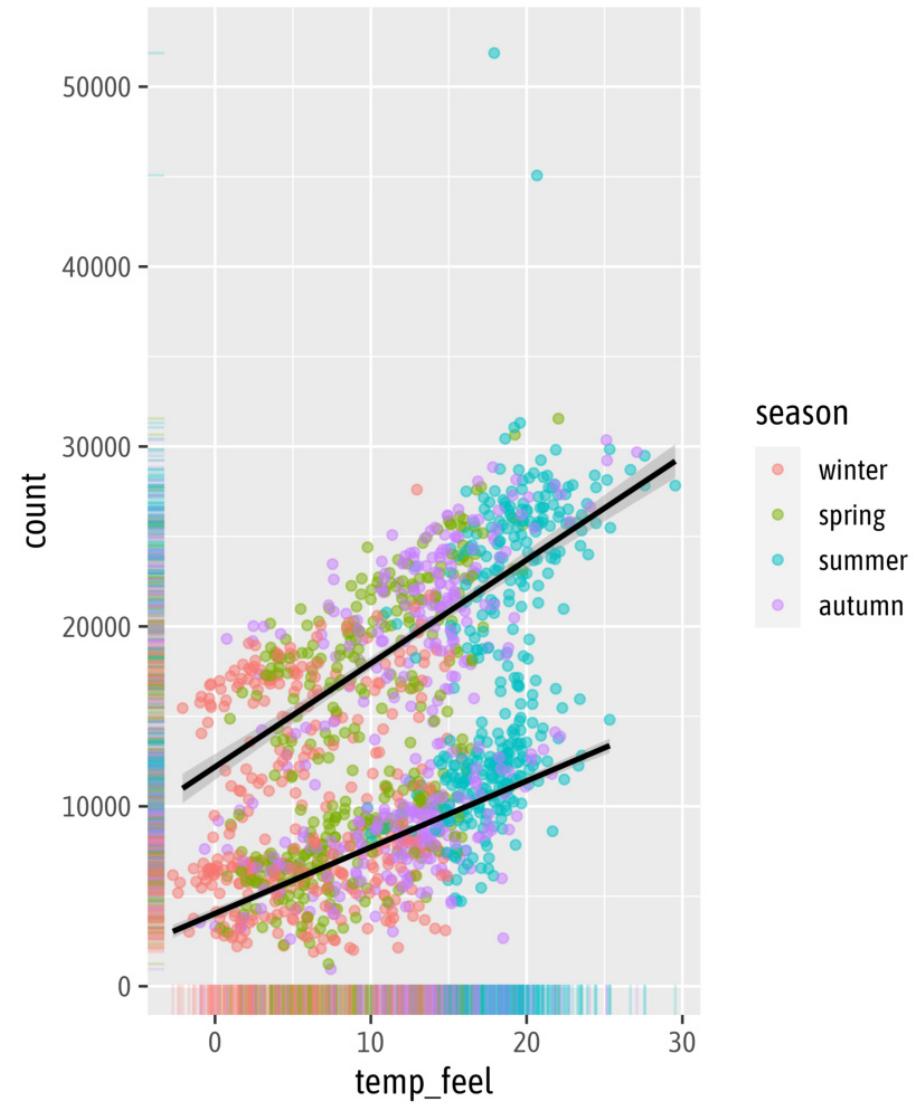
Extend a ggplot Object: Add Layers

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



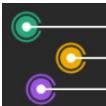
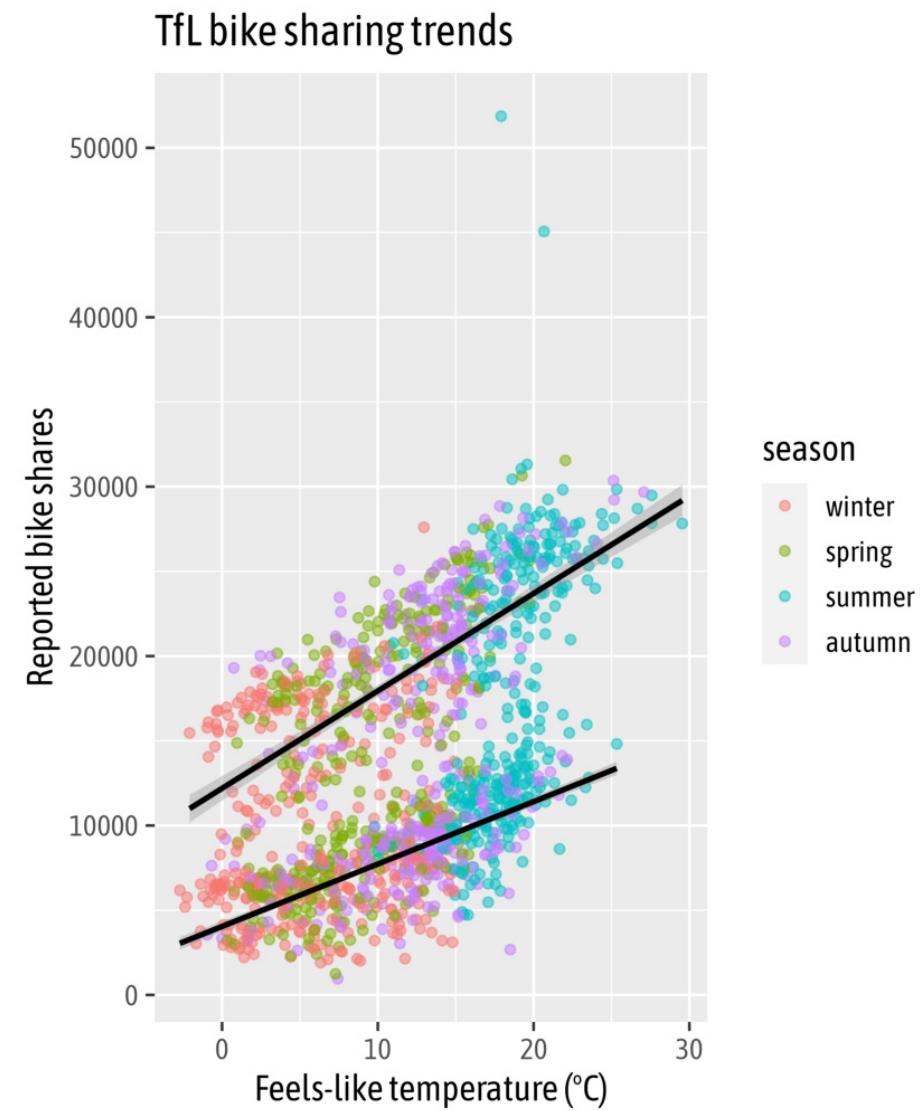
Remove a Layer from the Legend

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



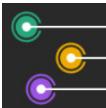
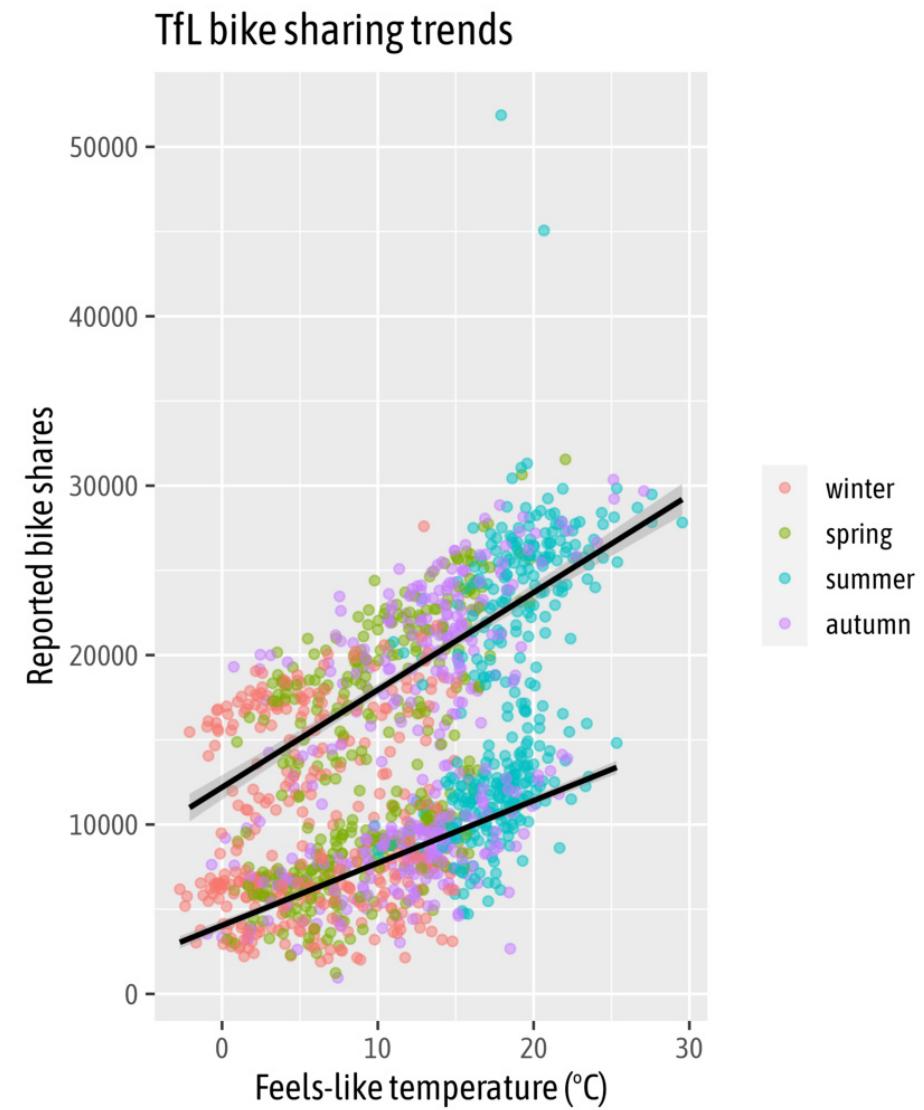
Extend a ggplot Object: Add Labels

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



Extend a ggplot Object: Add Labels

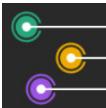
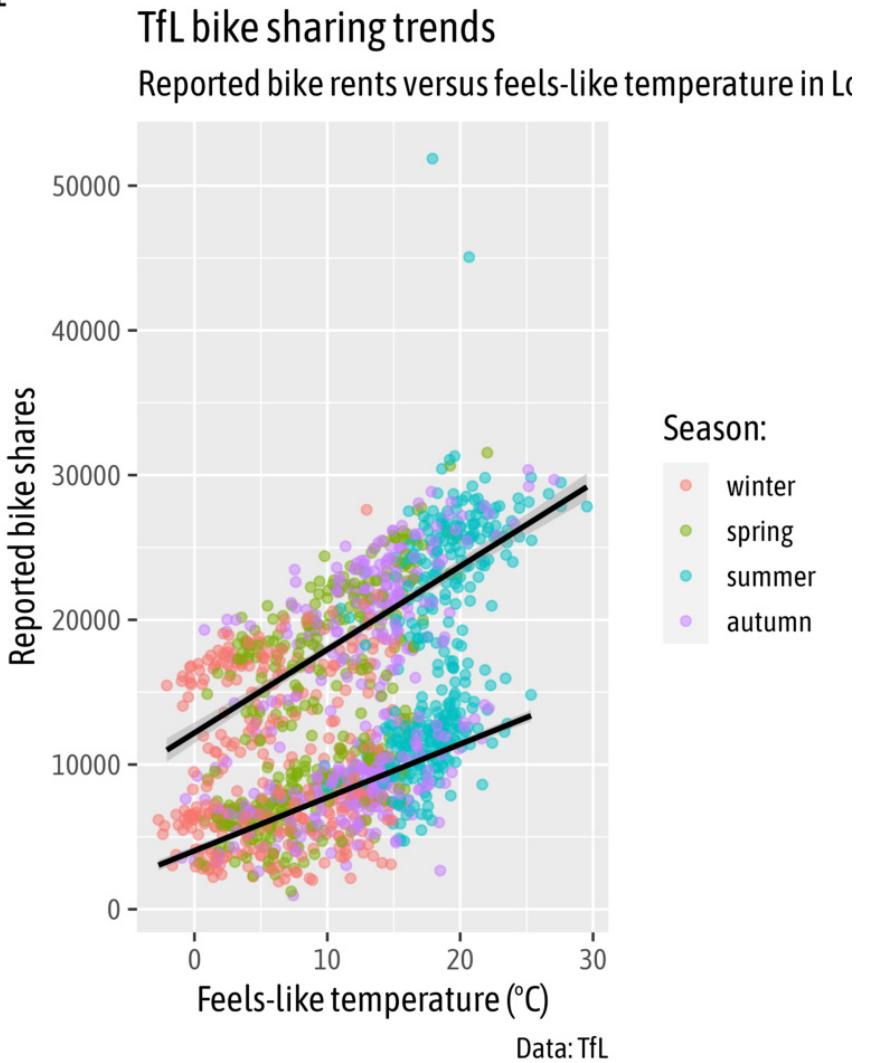
```
1 g <- g +  
2   labs(  
3     x = "Feels-like temperature (°C)",  
4     y = "Reported bike shares",  
5     title = "TfL bike sharing trends",  
6     color = NULL  
7   )  
8  
9 gg
```



Extend a ggplot Object: Add Labels

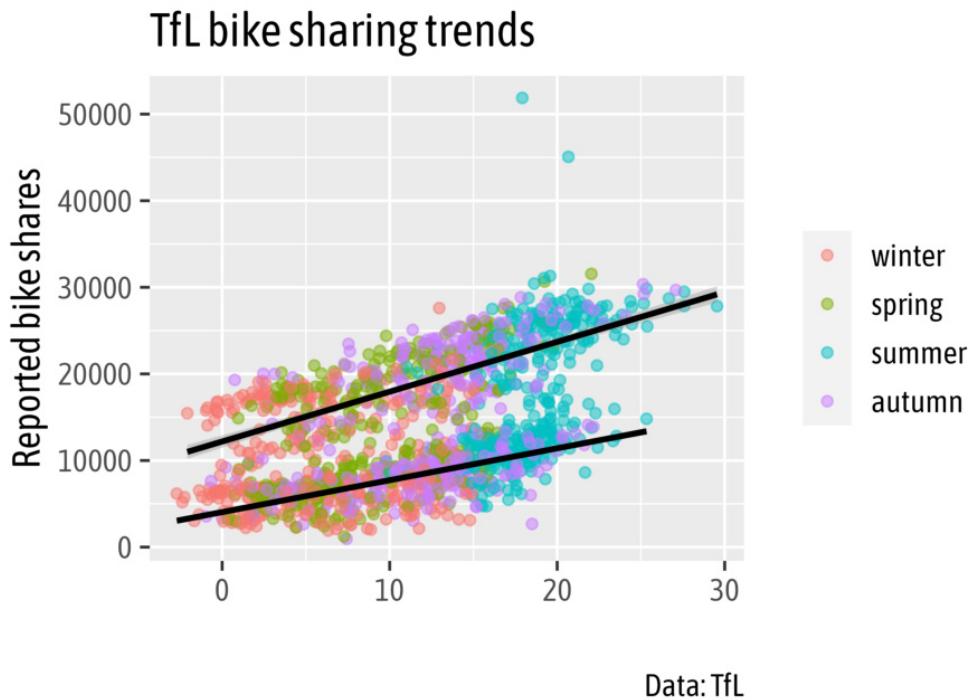
```
1 g +  
2   labs(  
3     subtitle = "Reported bike rents versus fe  
4     caption = "Data: TfL",  
5     tag = "Fig. 1",  
6     color = "Season:"  
7   )
```

Fig.1

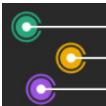
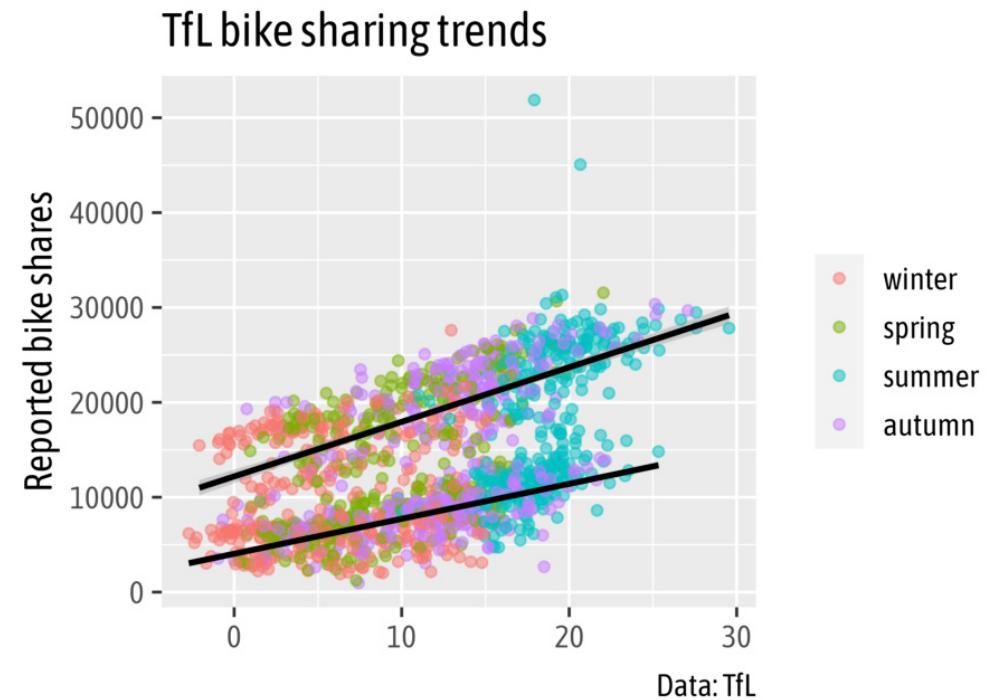


Extend a ggplot Object: Add Labels

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

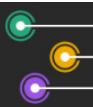


```
1 g +  
2   labs(  
3     x = NULL,  
4     caption = "Data: TfL"  
5   )
```



Data Visualization with {ggplot2}

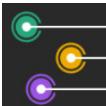
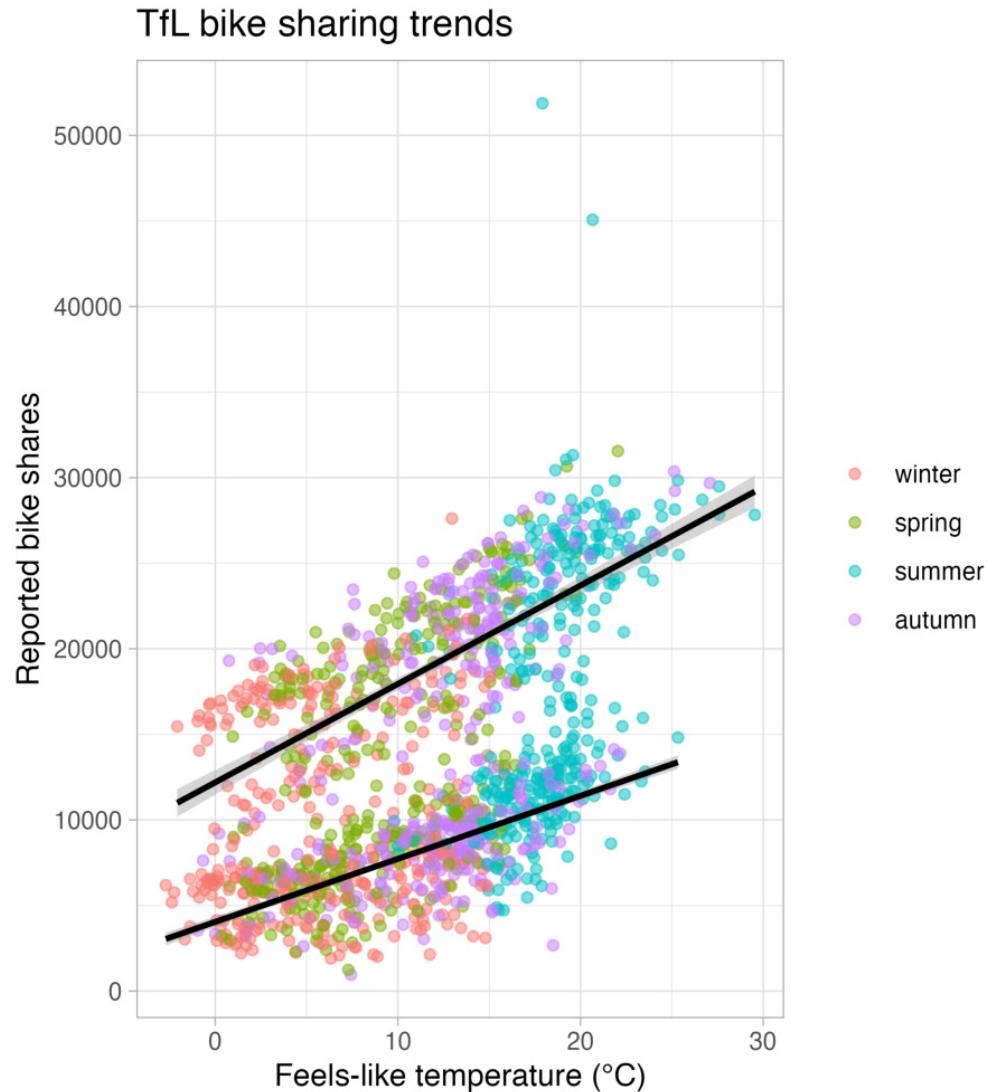
— Polishing Your Graphic —



Complete Themes

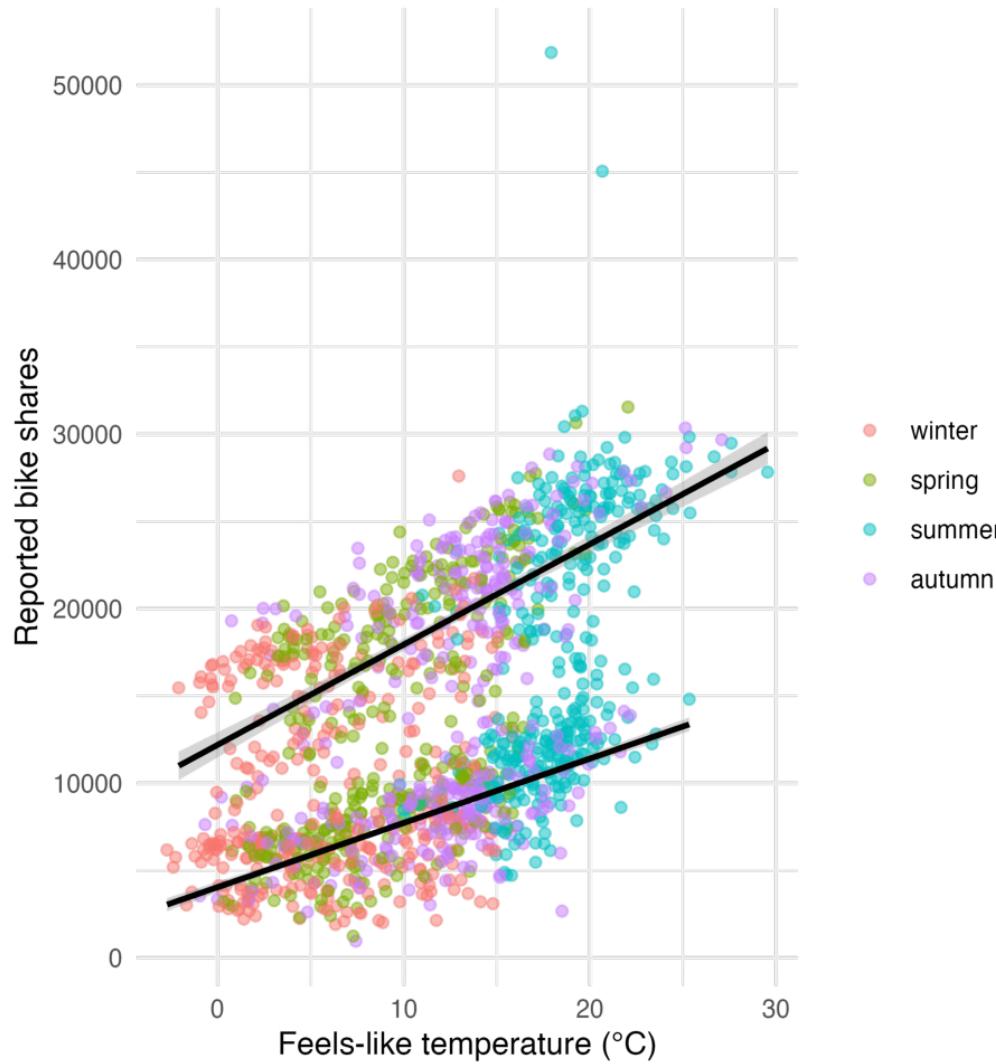
```
1 g + theme_light()
```

...



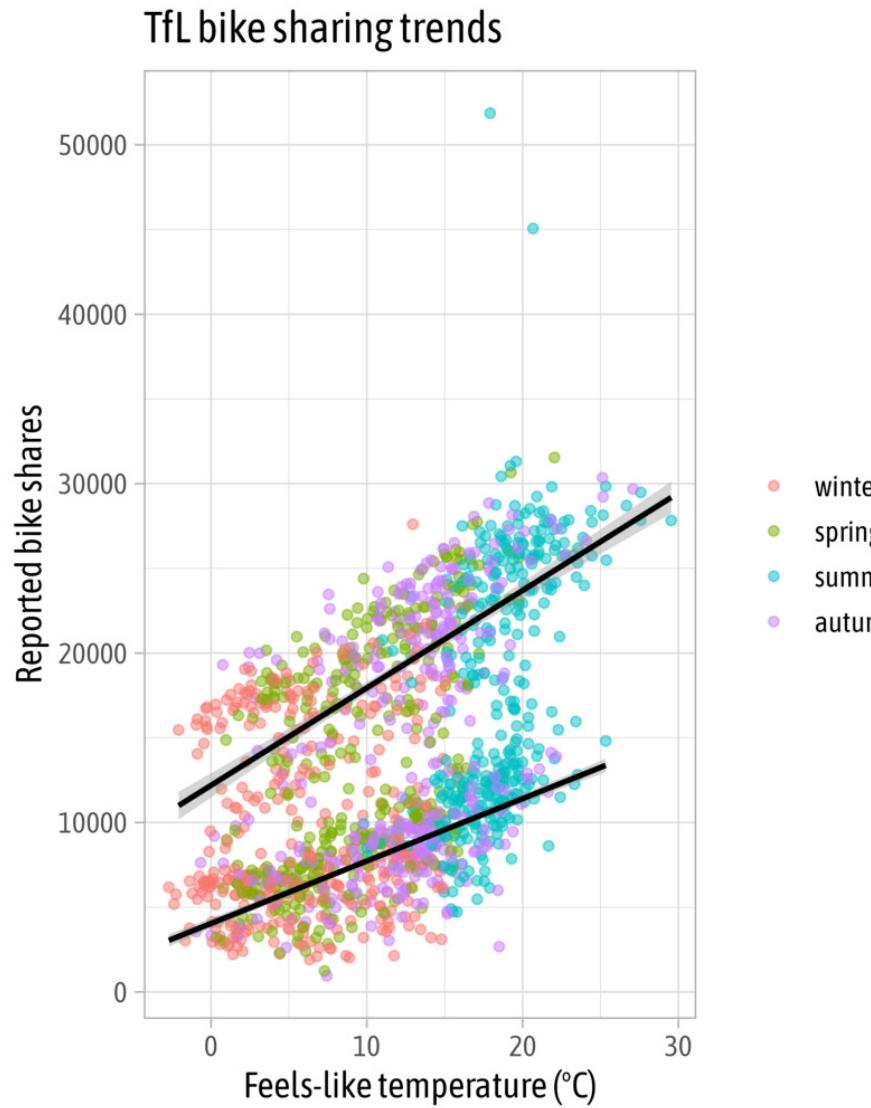
```
1 g + theme_minimal()
```

TfL bike sharing trends



Change the Theme Base Settings

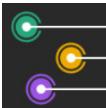
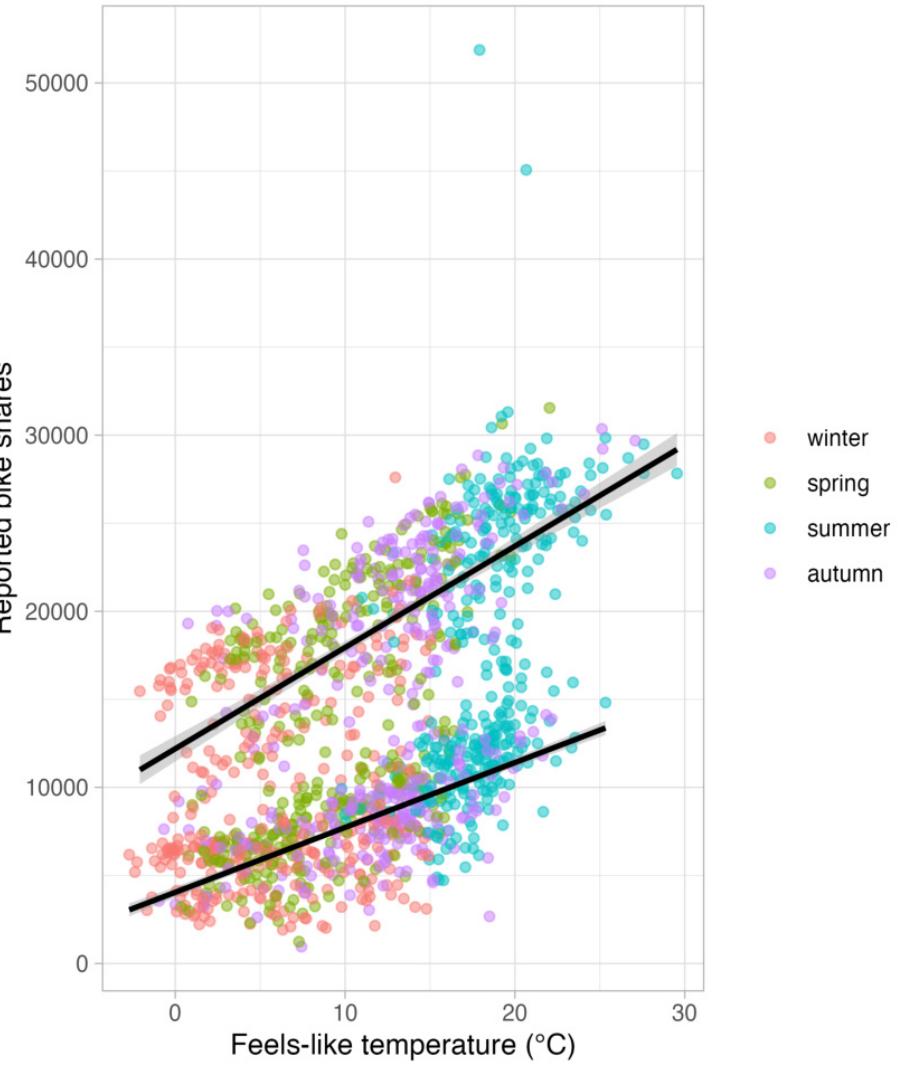
```
1 g + theme_light(  
2   base_size = 14,  
3   base_family = "Asap Condensed"  
4 )
```



Set a Theme Globally

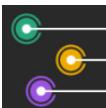
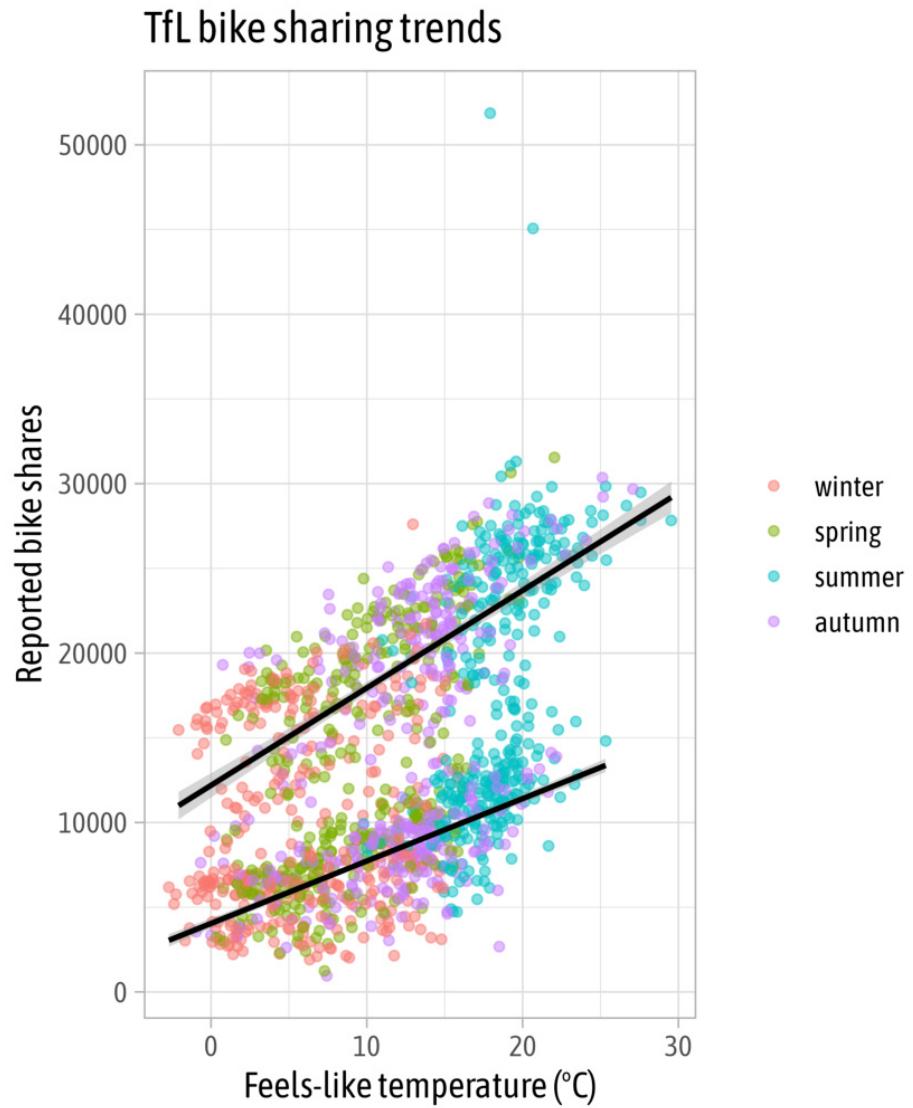
```
1 theme_set(theme_light())  
2  
3 g
```

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
3 library(systemfonts)
4
5 system_fonts() %>%
6   filter(stringr::str_detect(family, "Cabinet")) %>%
7   pull(name) %>%
8   sort()
```

```
[1] "CabinetGrotesk-Black"          "CabinetGrotesk-Bold"
[3] "CabinetGrotesk-Extrabold"       "CabinetGrotesk-Extralight"
[5] "CabinetGrotesk-Light"          "CabinetGrotesk-Medium"
[7] "CabinetGrotesk-Regular"         "CabinetGrotesk-Thin"
[9] "CabinetGroteskVariable-Bold"    "CabinetGroteskVariable-Bold_Bold"
[11] "CabinetGroteskVariable-Bold_Extra bold" "CabinetGroteskVariable-Bold_Extralight"
[13] "CabinetGroteskVariable-Bold_Light"  "CabinetGroteskVariable-Bold_Medium"
[15] "CabinetGroteskVariable-Bold_Regular" "CabinetGroteskVariable-Bold_Thin"
```



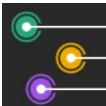
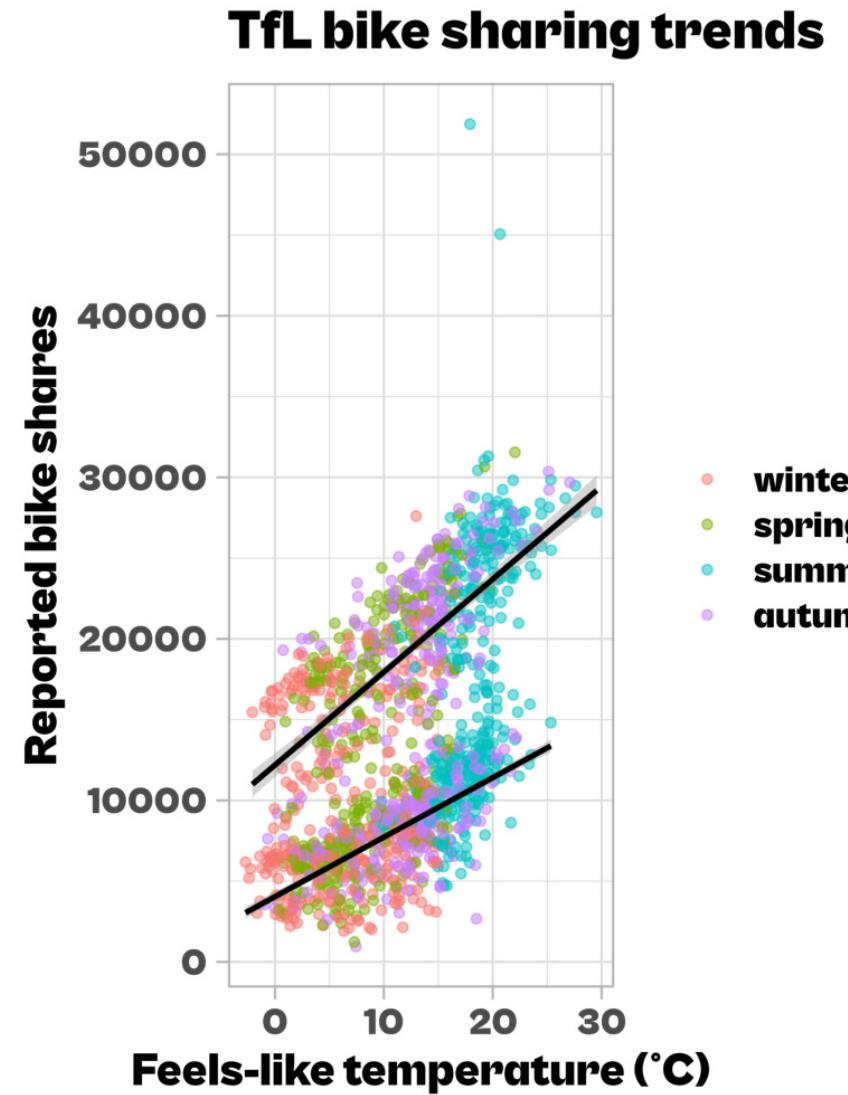
{systemfonts}

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



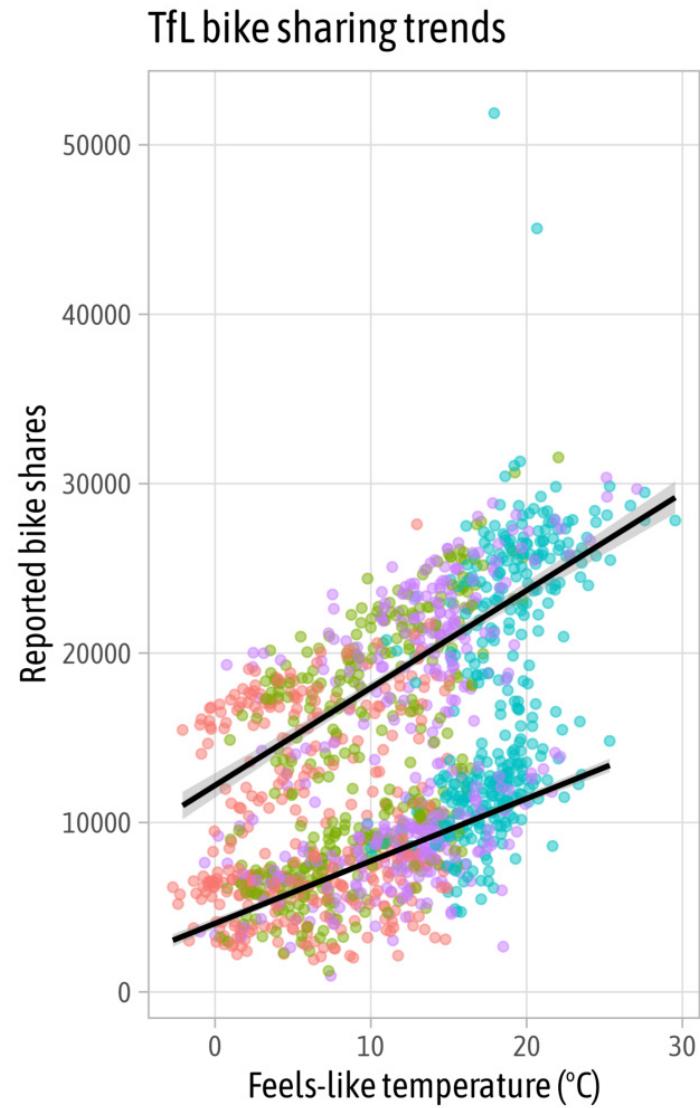
{systemfonts} + {ggplot2}

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



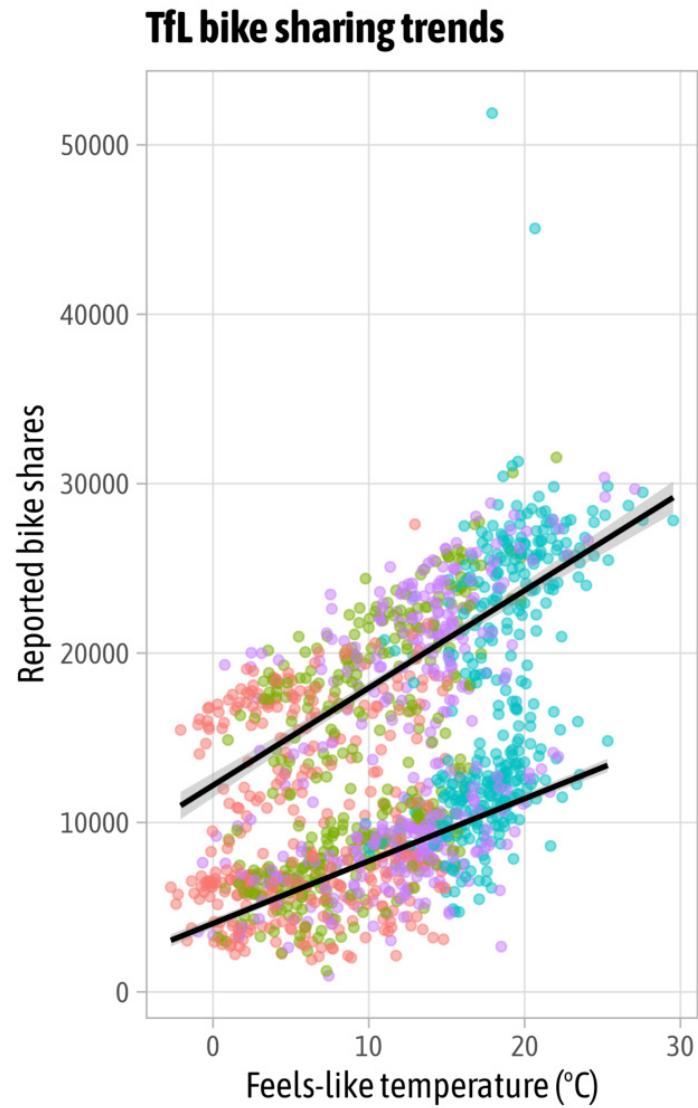
Overwrite Specific Theme Settings

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



Overwrite Specific Theme Settings

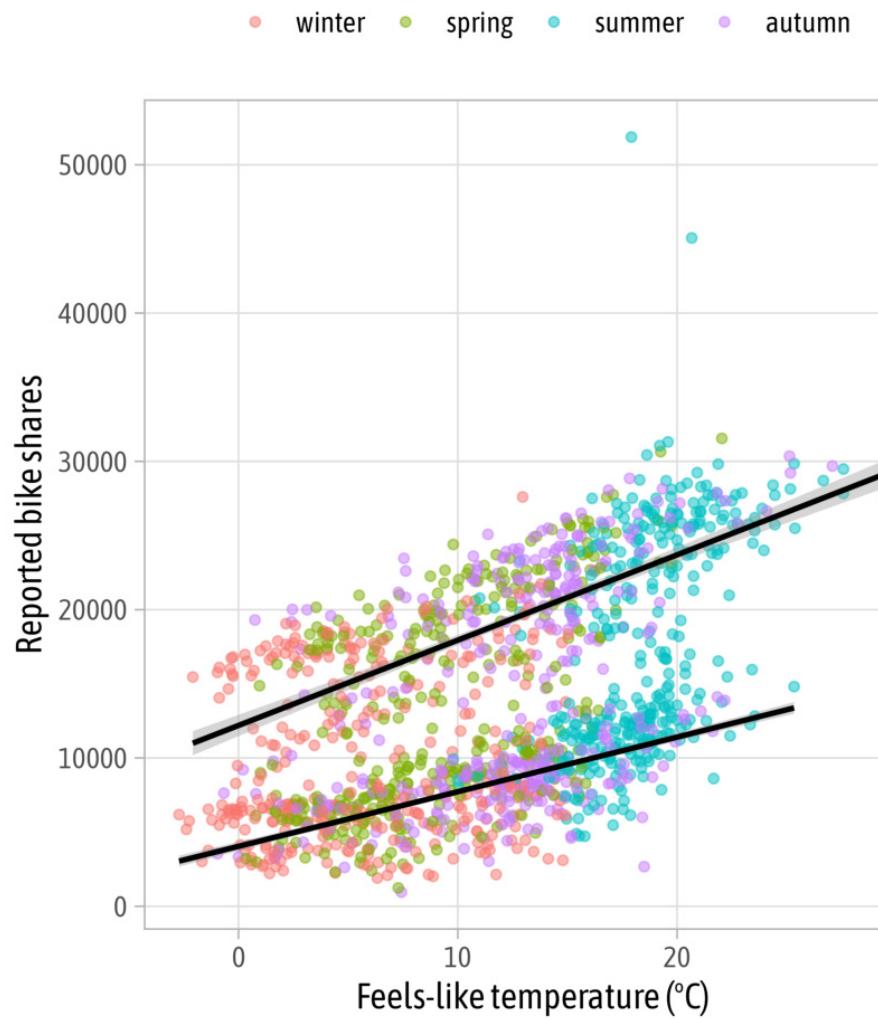
```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold")  
5   )
```



Overwrite Specific Theme Settings

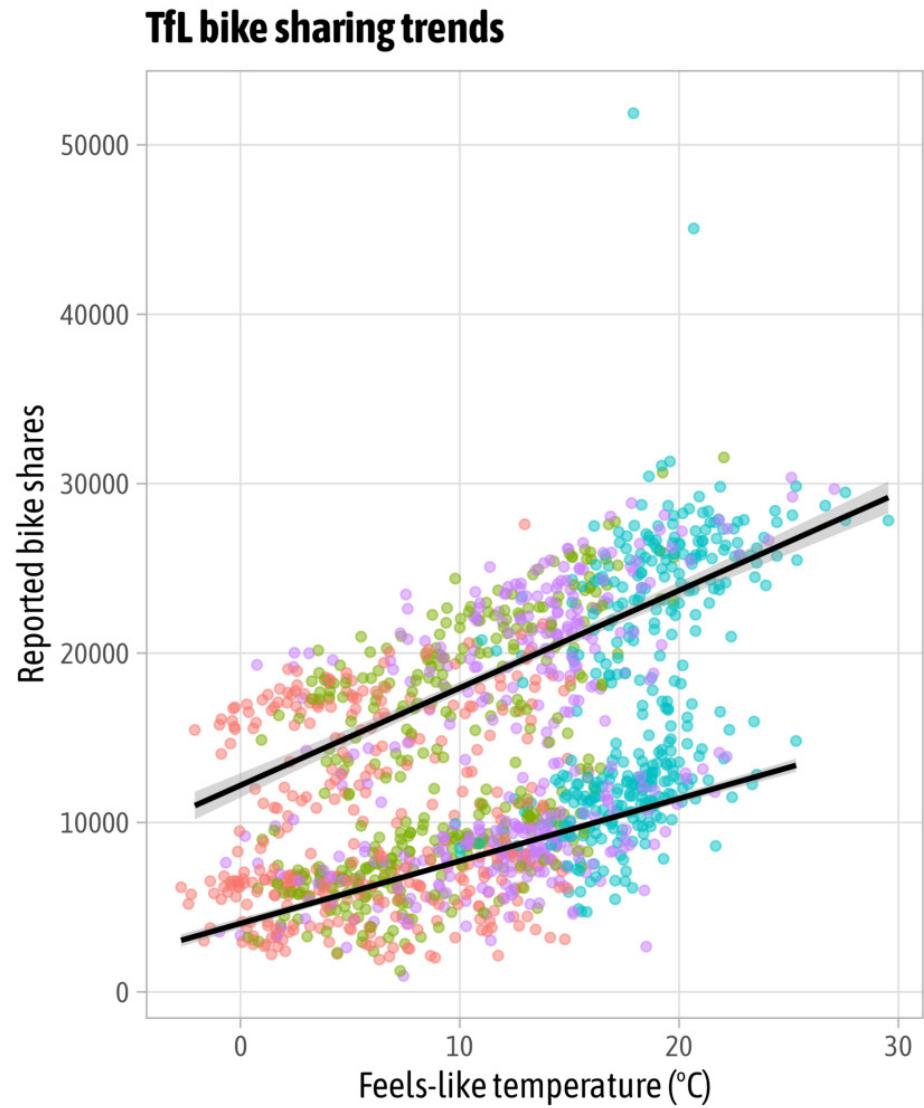
```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold"),  
5     legend.position = "top"  
6   )
```

TfL bike sharing trends



Overwrite Specific Theme Settings

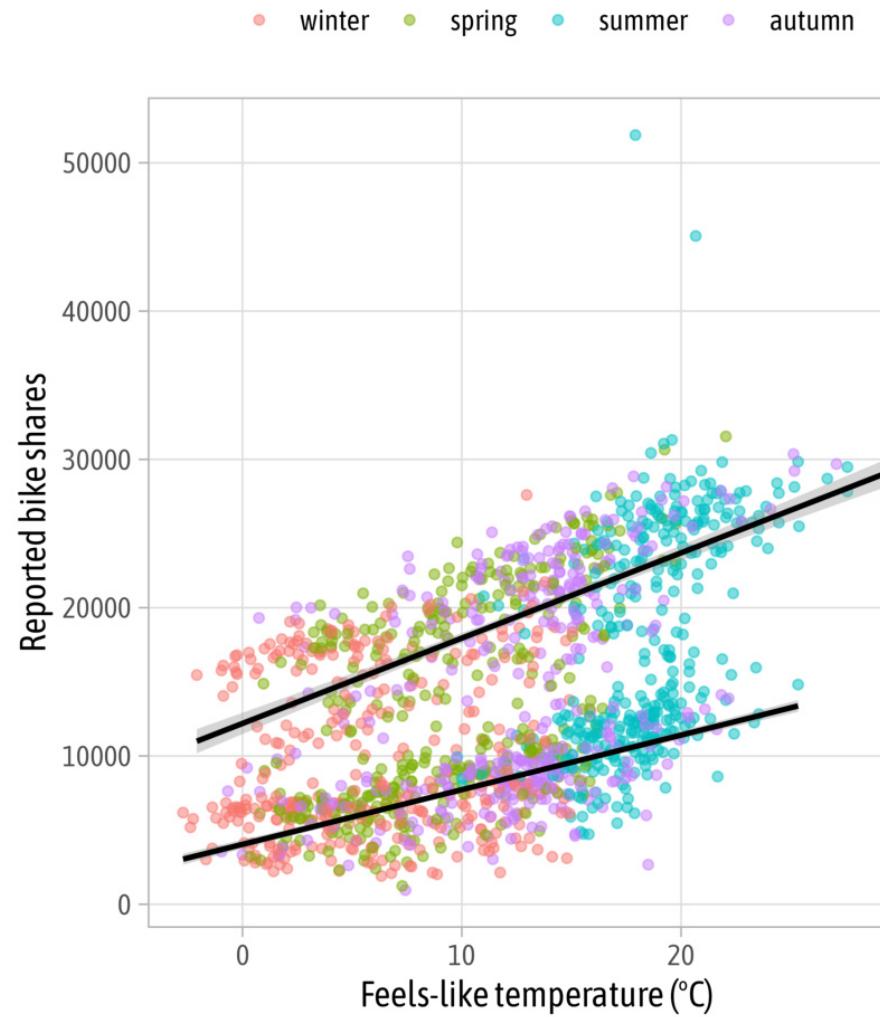
```
1 g +  
2   theme(  
3     panel.grid.minor = element_blank(),  
4     plot.title = element_text(face = "bold"),  
5     legend.position = "none"  
6   )
```



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   )
```

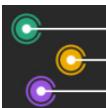
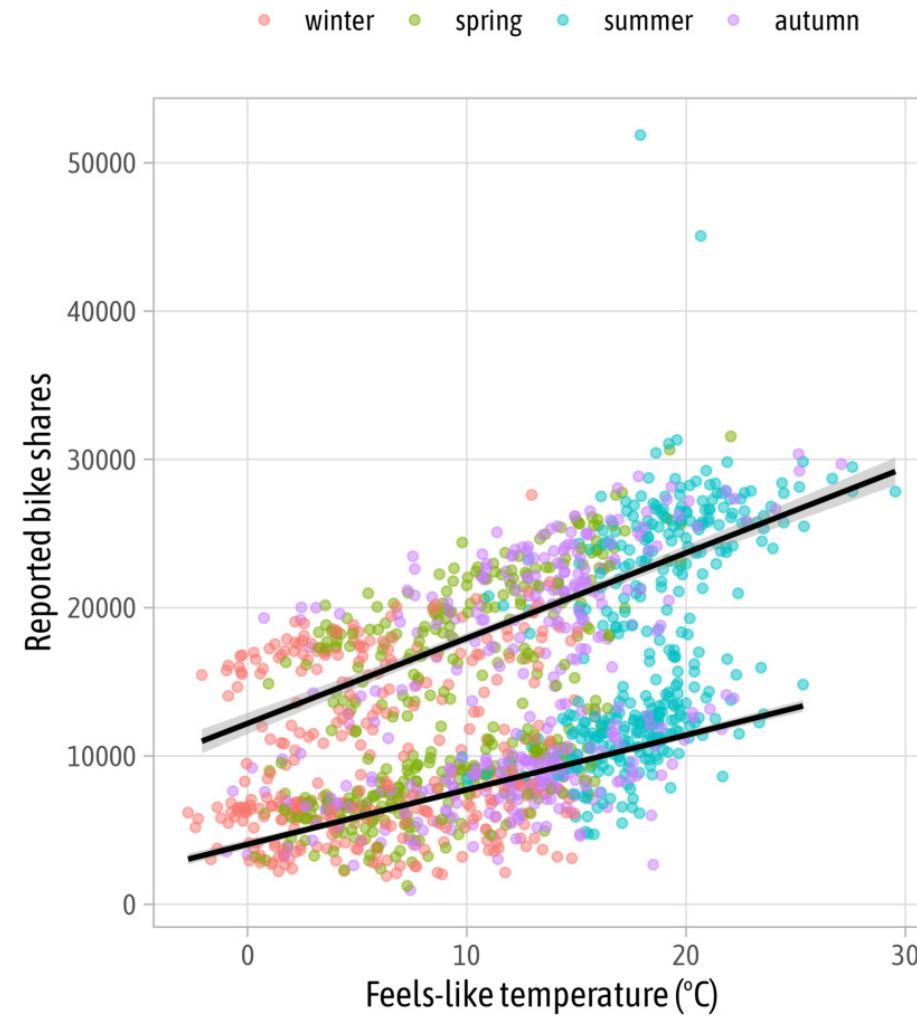
TfL bike sharing trends



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



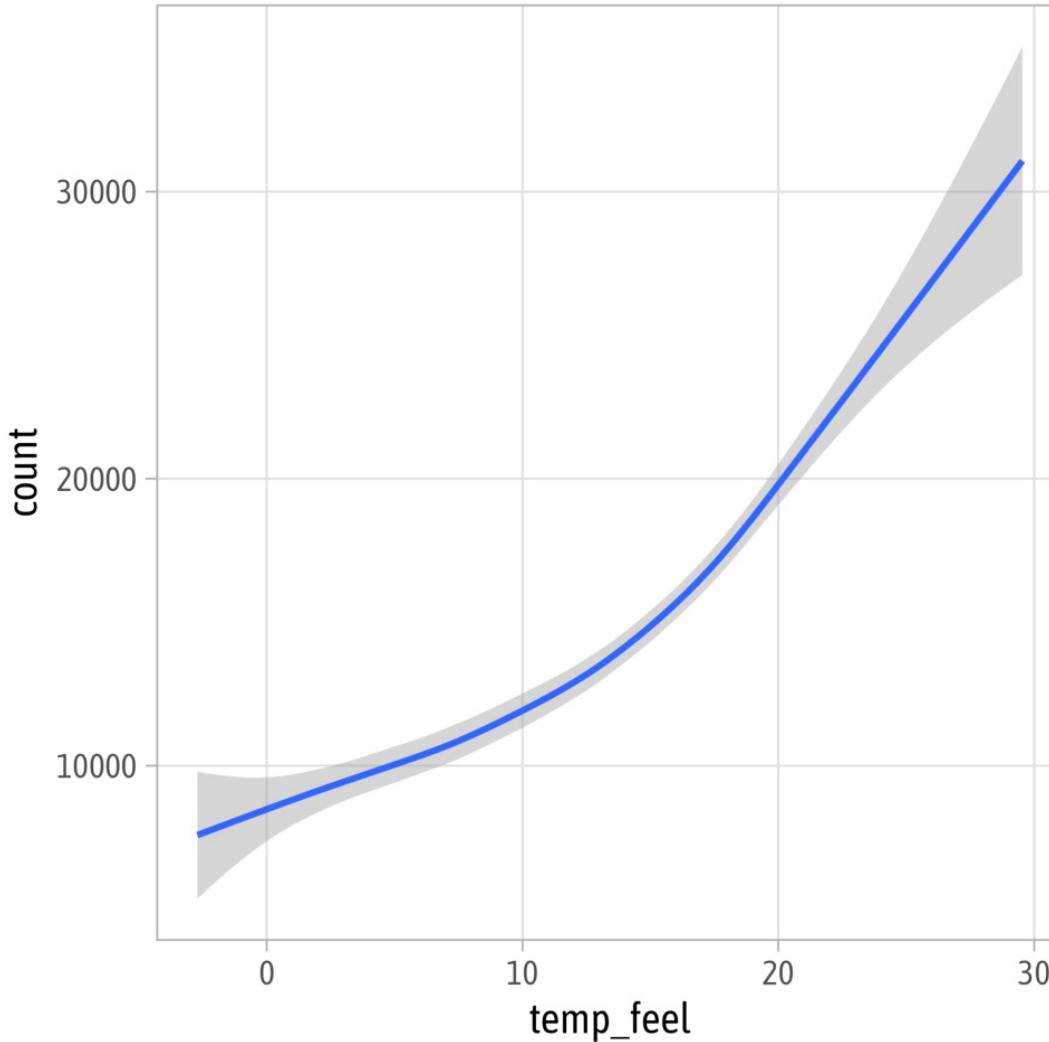
Data Visualization with {ggplot2}

— Statistical Layers —

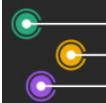
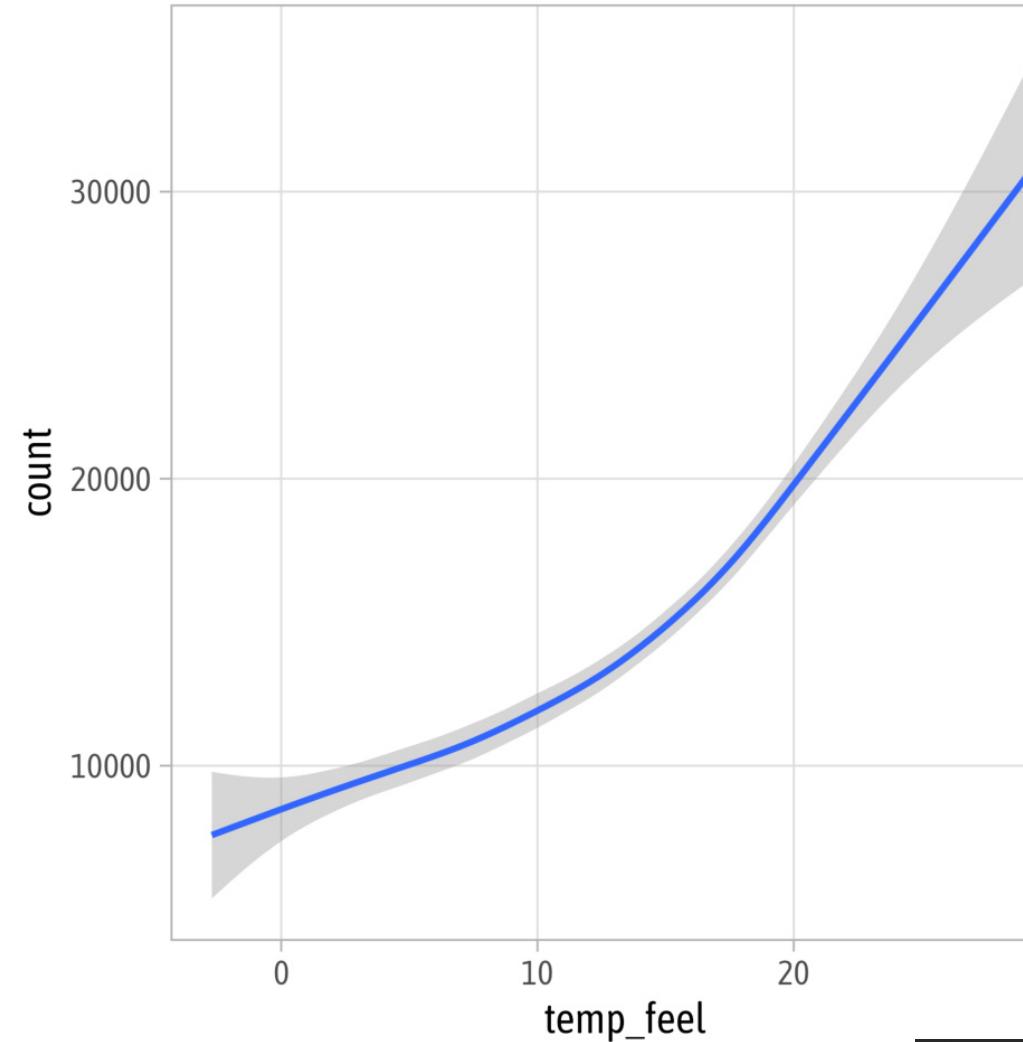


geom_*() vs stat_*()

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

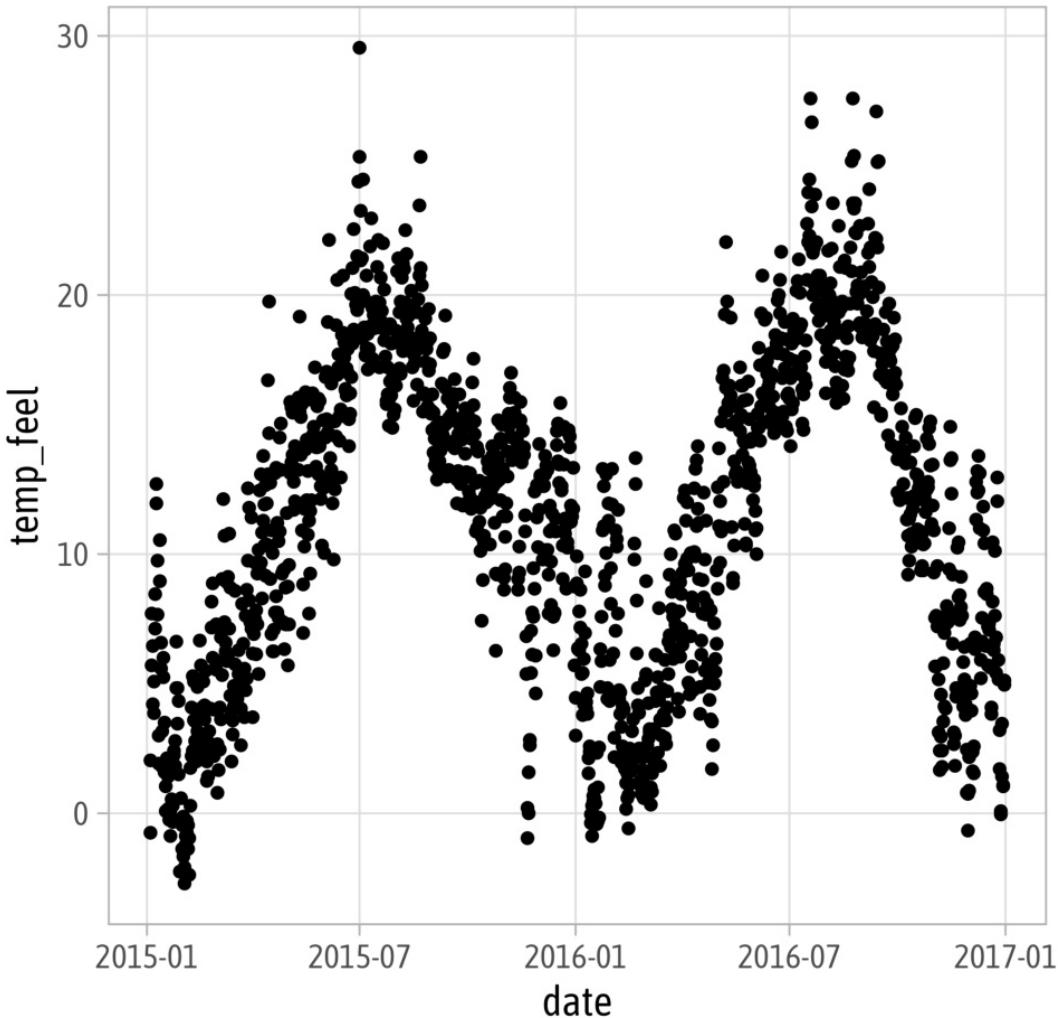


```
1 ggplot(bikes, aes(x = temp_feel, y = count)) +  
2   geom_smooth(stat = "smooth")
```

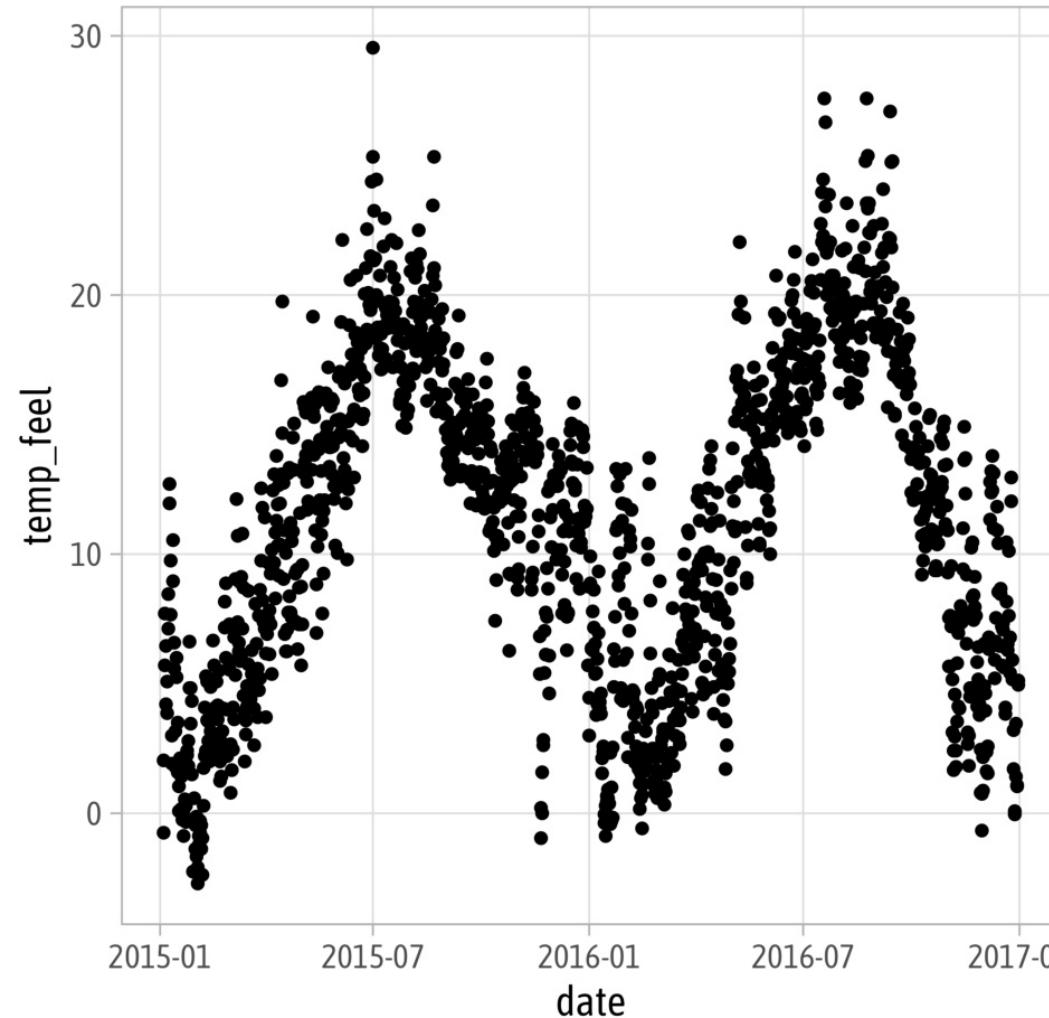


geom_*() vs stat_*()

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

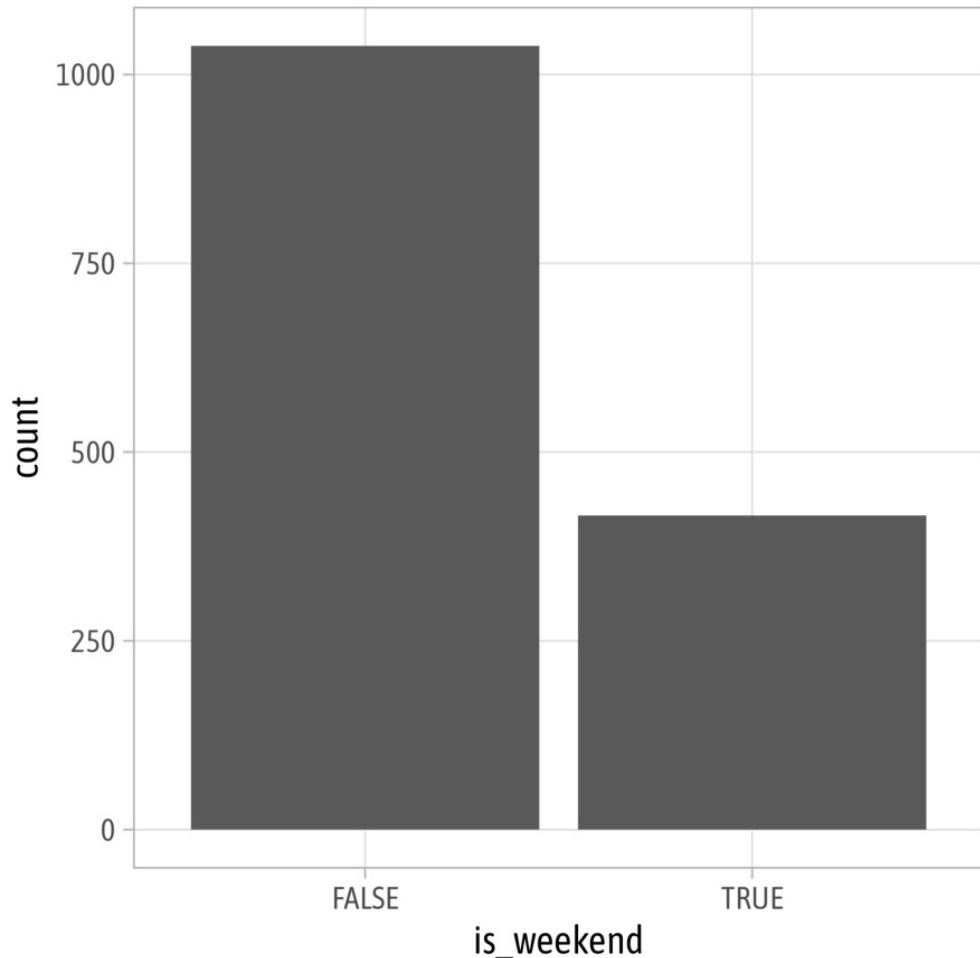


```
1 ggplot(bikes, aes(x = date, y = temp_feel)) +  
2   stat_identity(geom = "point")
```

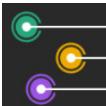
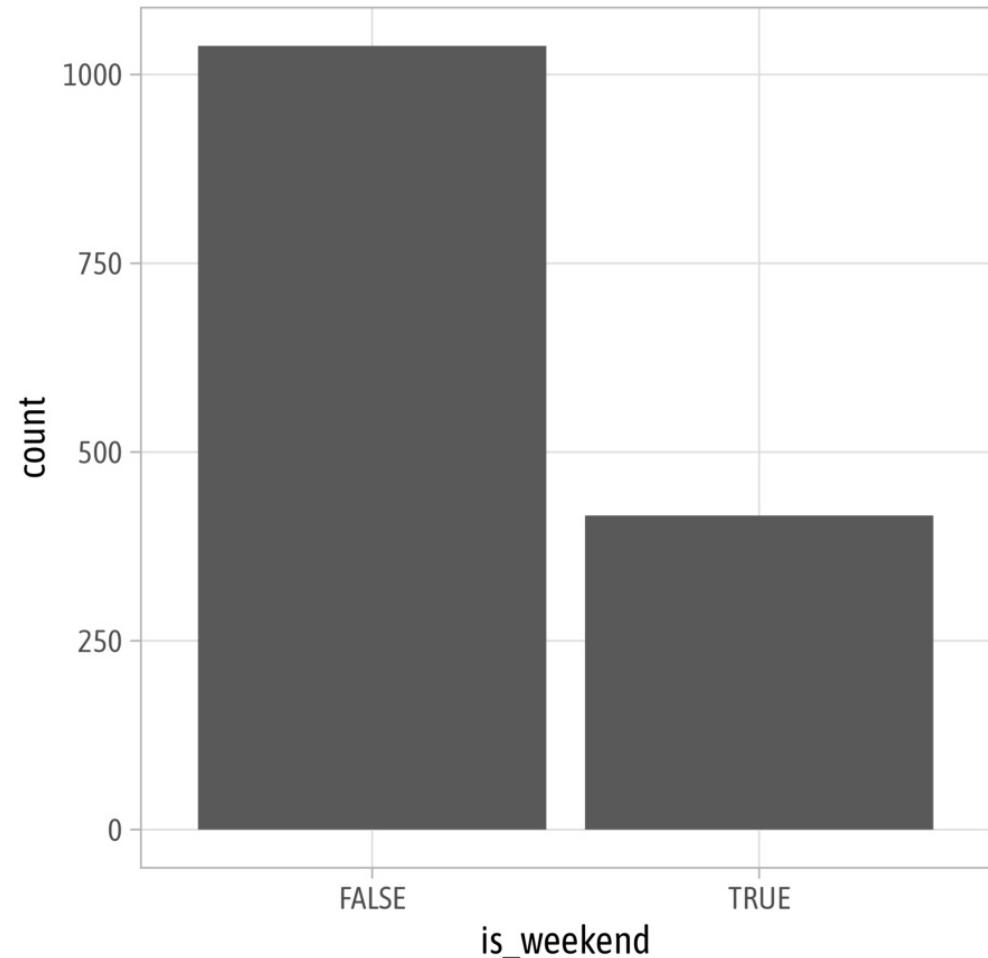


geom_*() vs stat_*()

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

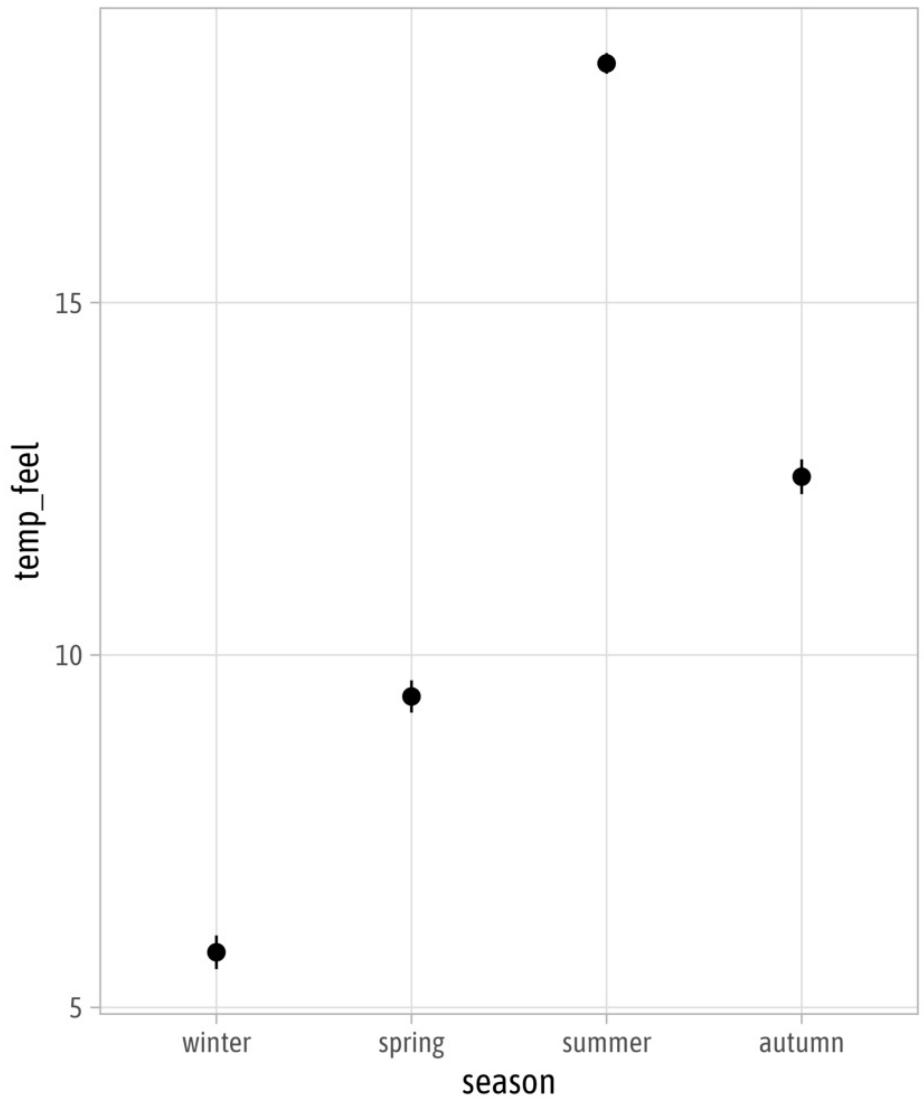


```
1 ggplot(bikes, aes(x = is_weekend)) +  
2   stat_count(geom = "bar")
```



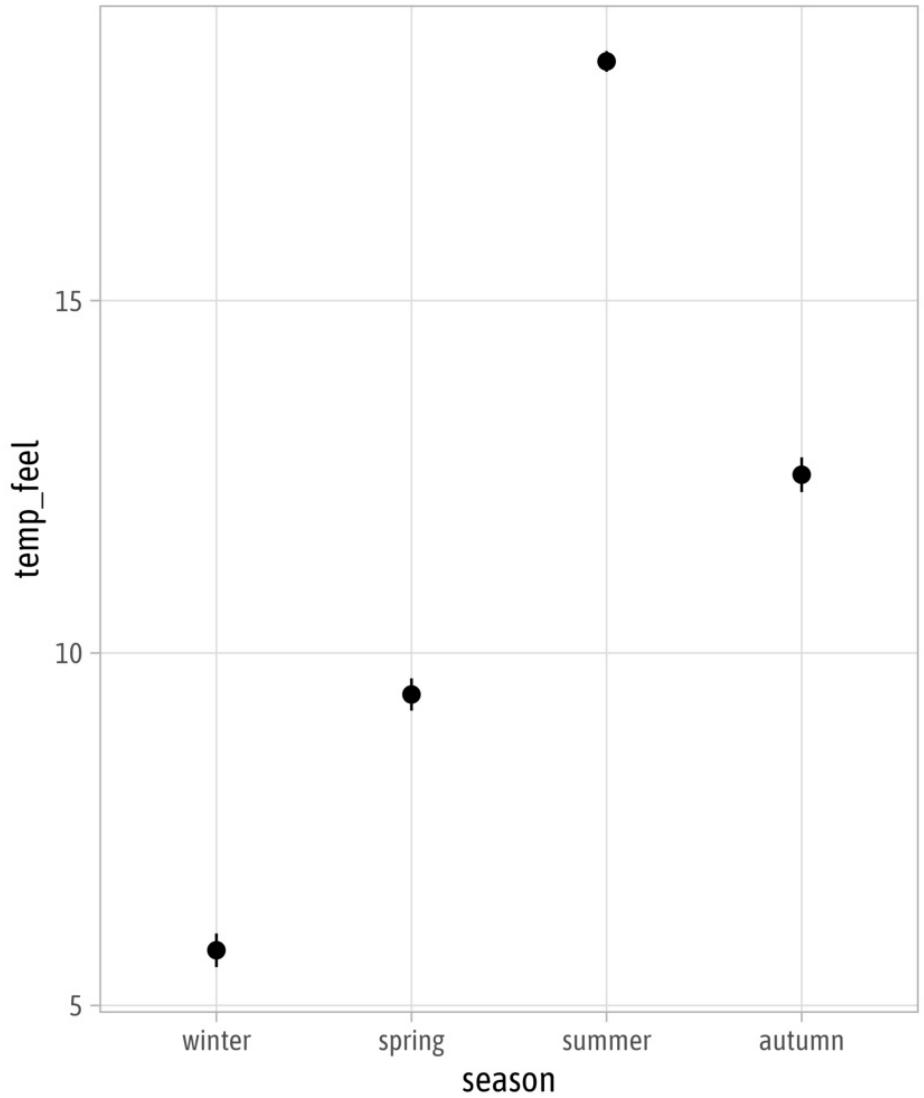
Statistical Summaries

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



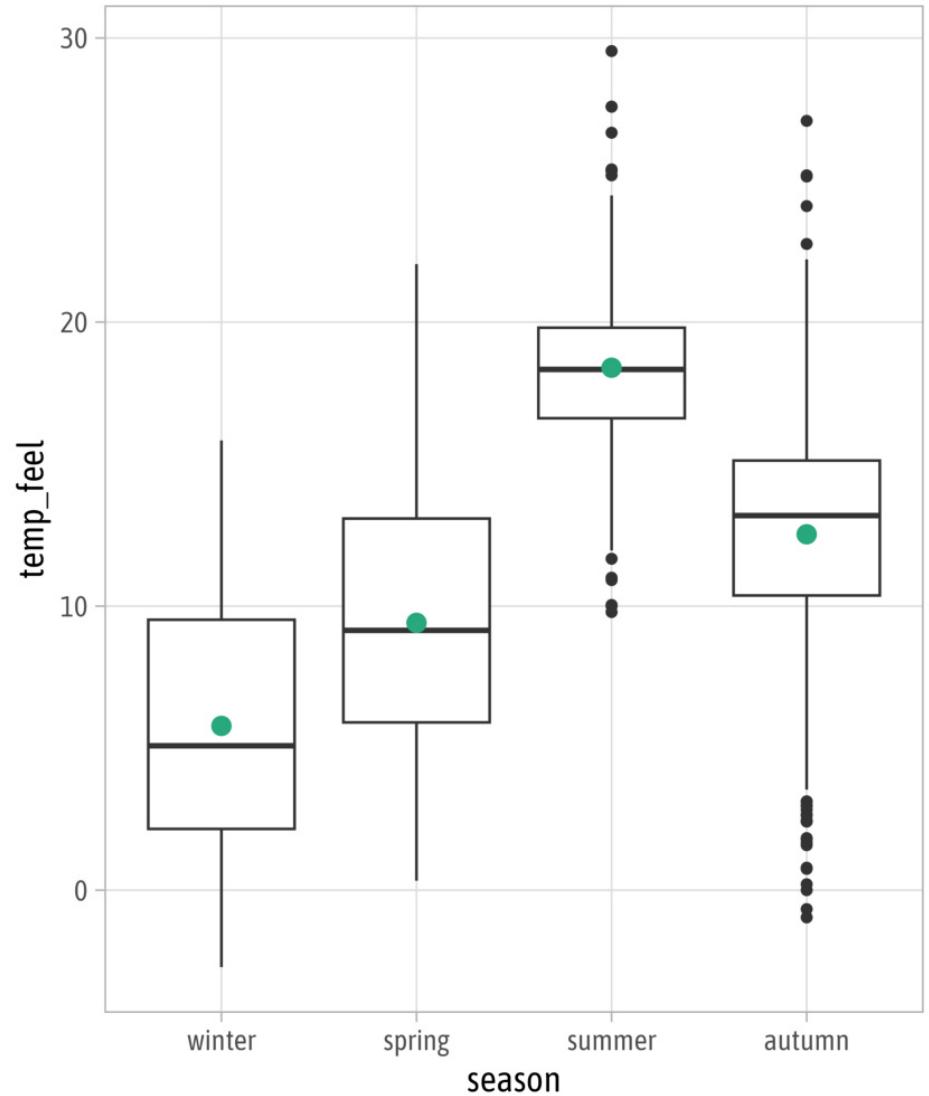
Statistical Summaries

```
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 )
```



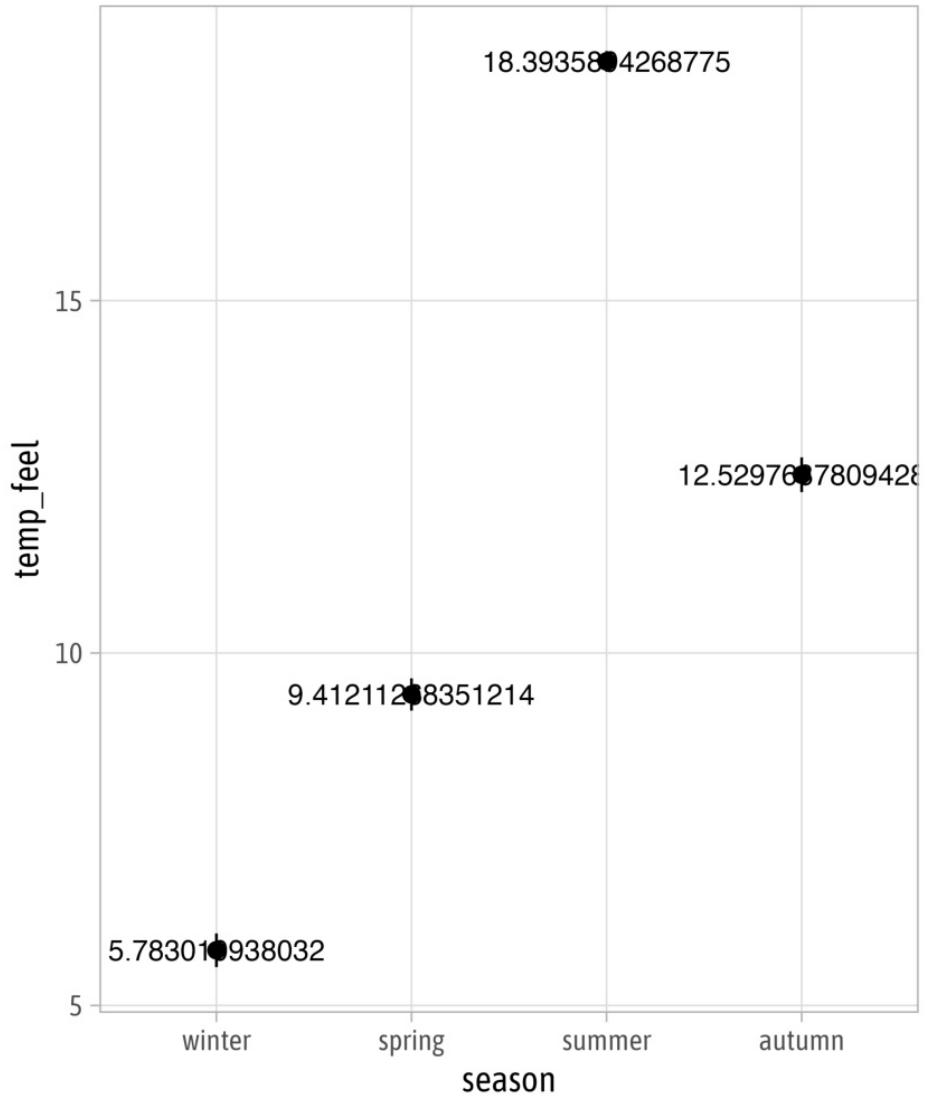
Statistical Summaries

```
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 )
```



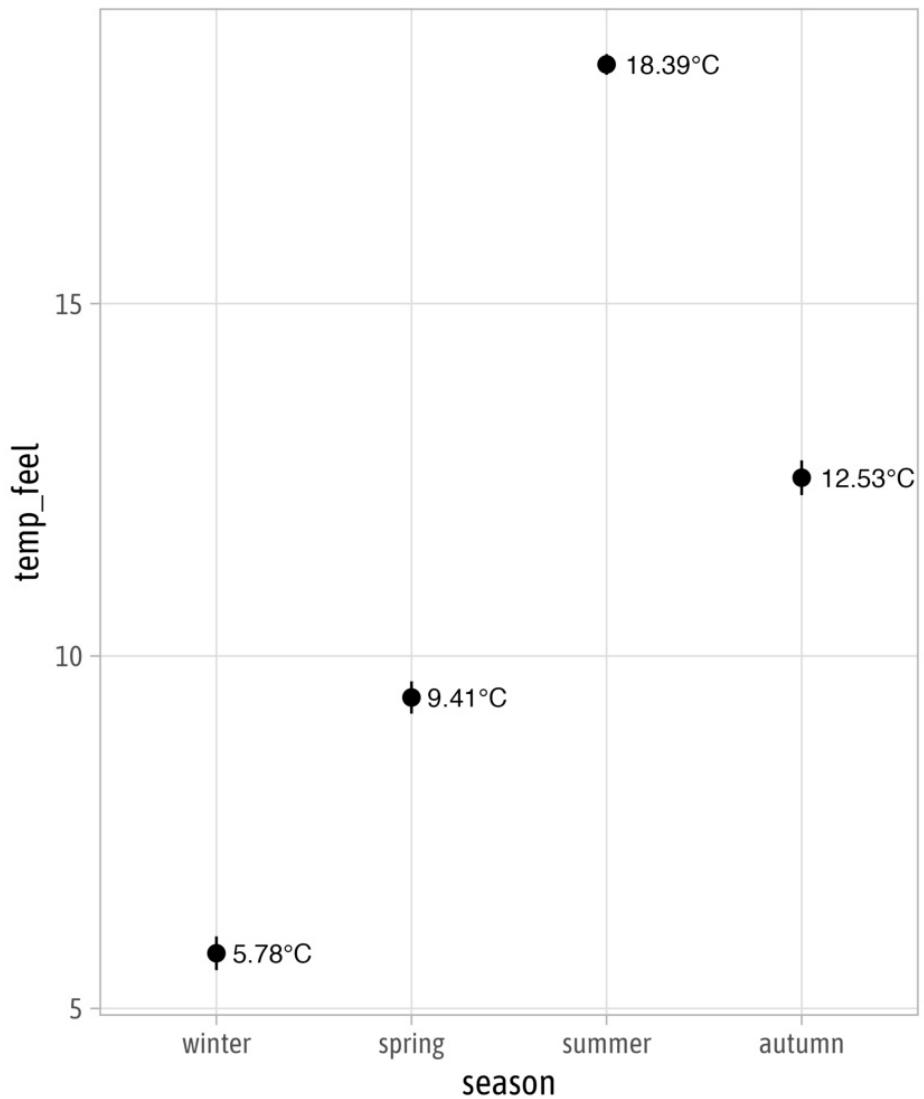
Statistical Summaries

```
1 ggplot(  
2   bikes,  
3   aes(x  
4     = season, y = temp_feel)  
5 ) +  
6 stat_summary() +  
7 stat_summary(  
8   fun = mean,  
9   geom = "text",  
10  aes(label = after_stat(y))  
11 )
```



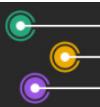
Statistical Summaries

```
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(  
10    paste0(round(y, 2), "°C"))  
11 ),  
12 hjust = -.2,  
13 size = 3.5  
14 )
```



Data Visualization with {ggplot2}

— Facets —



Facets

= split variables to multiple panels

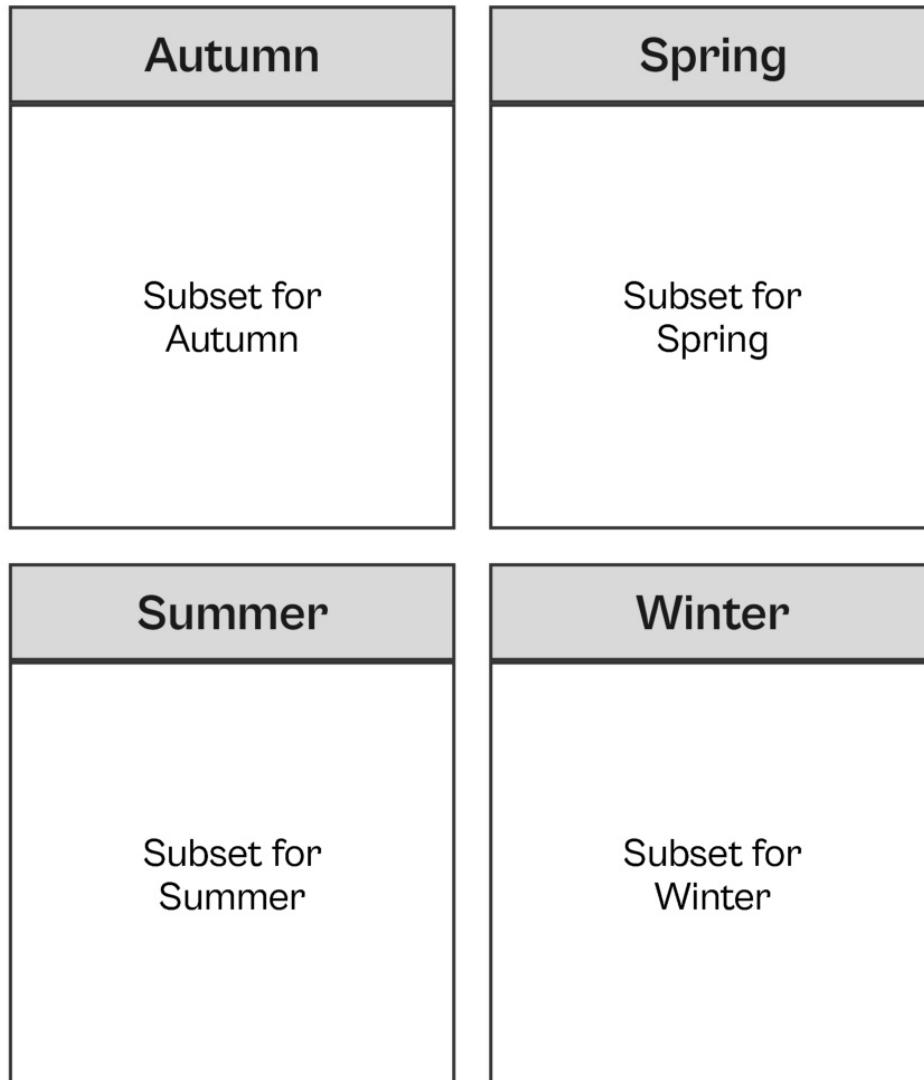
Facets are also known as:

- small multiples
- trellis graphs
- lattice plots
- conditioning

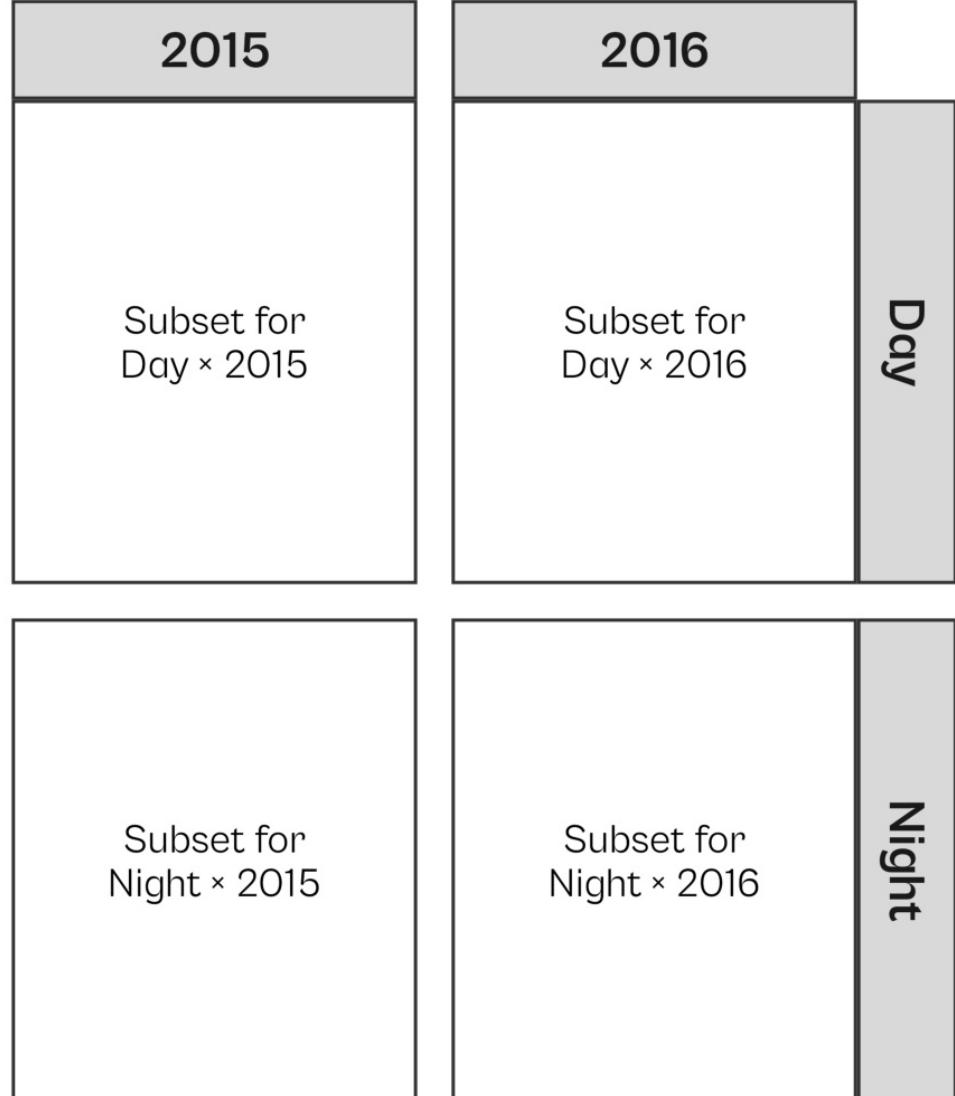


`facet_wrap()`

...



facet_grid()

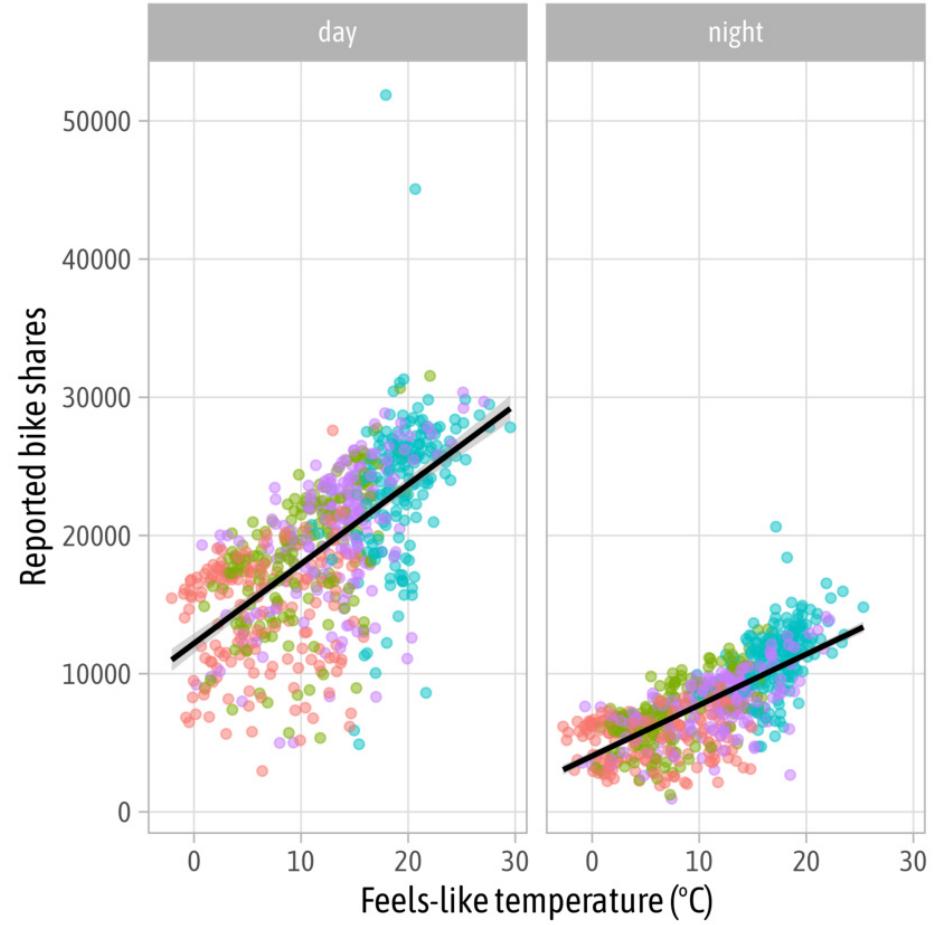


Wrapped Facets

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

TfL bike sharing trends

● winter ● spring ● summer ● autumn

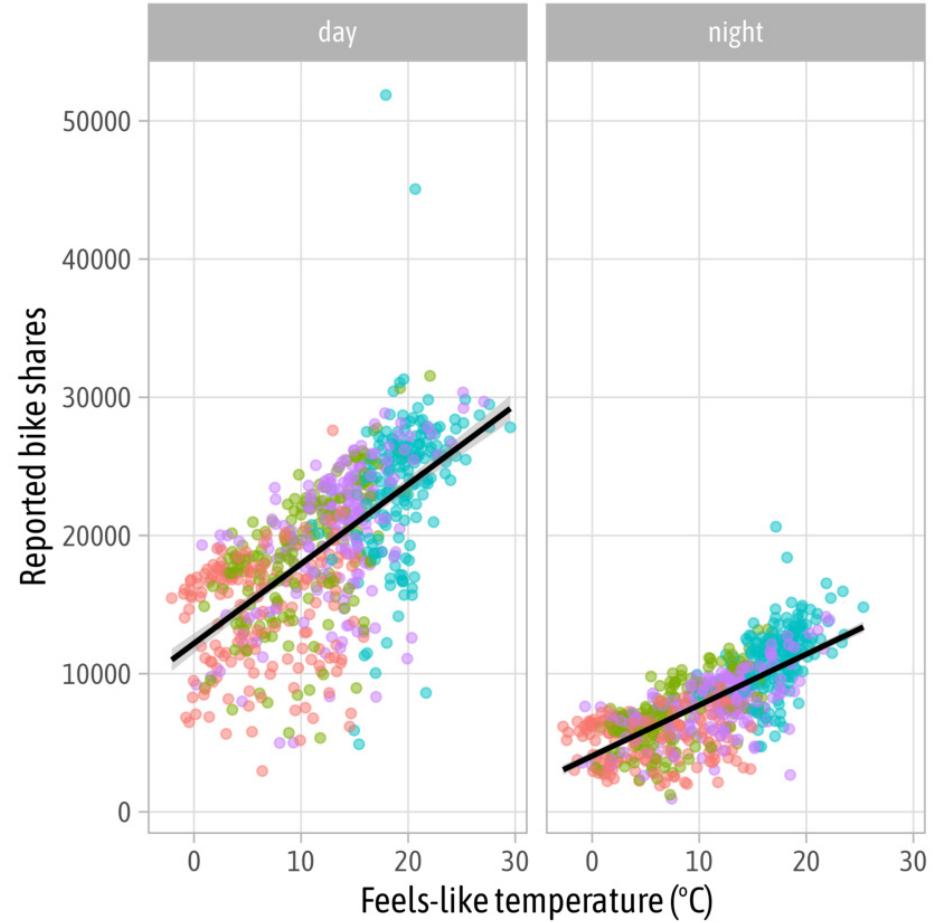


Wrapped Facets

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

TfL bike sharing trends

● winter ● spring ● summer ● autumn

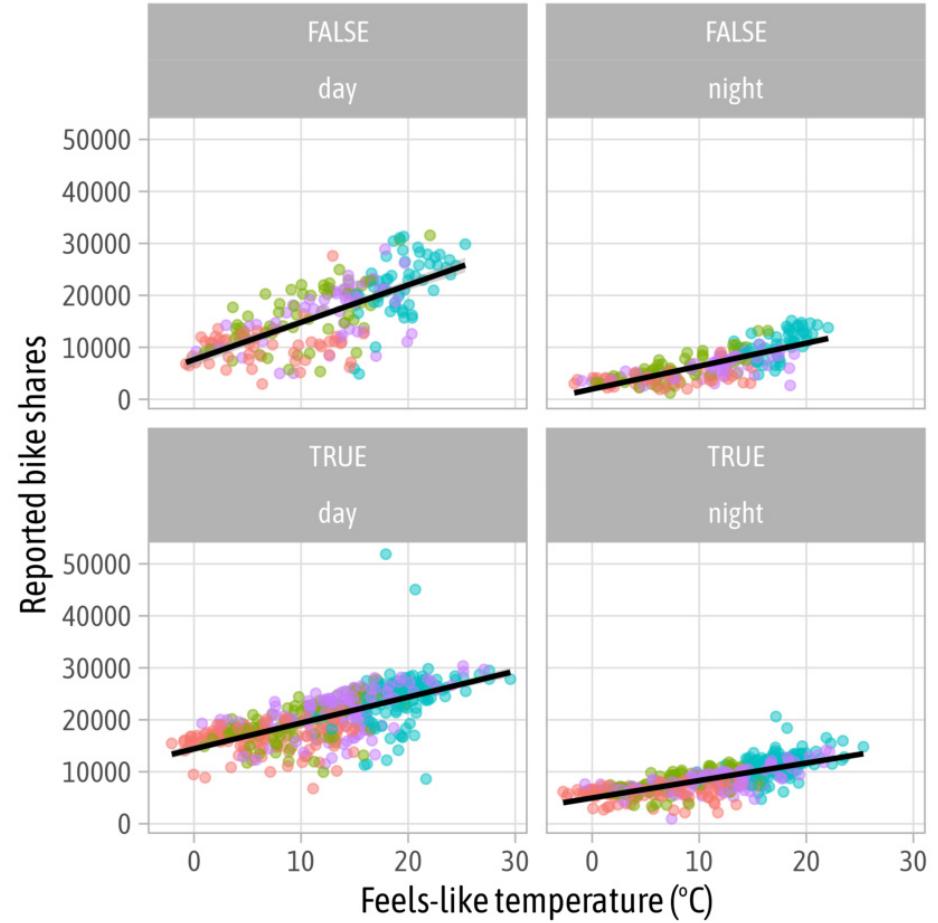


Facet Multiple Variables

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

TfL bike sharing trends

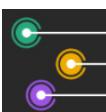
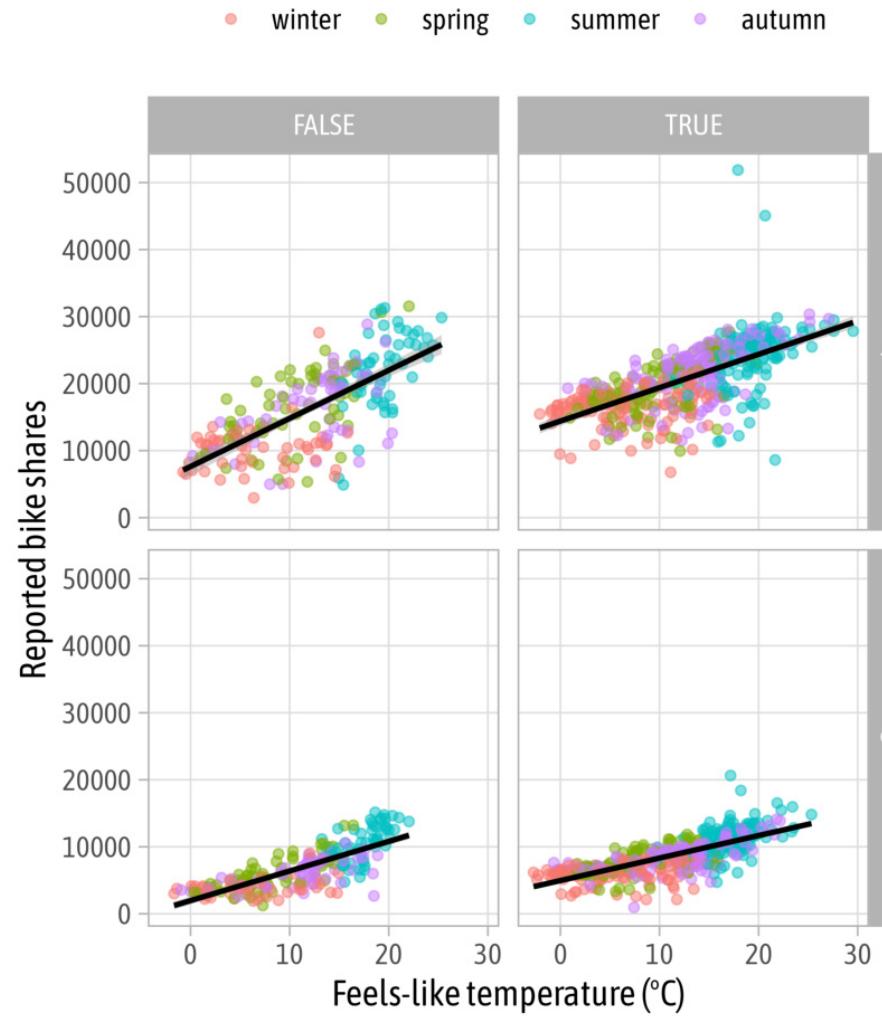
● winter ● spring ● summer ● autumn



Gridded Facets

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

TfL bike sharing trends

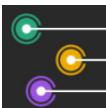
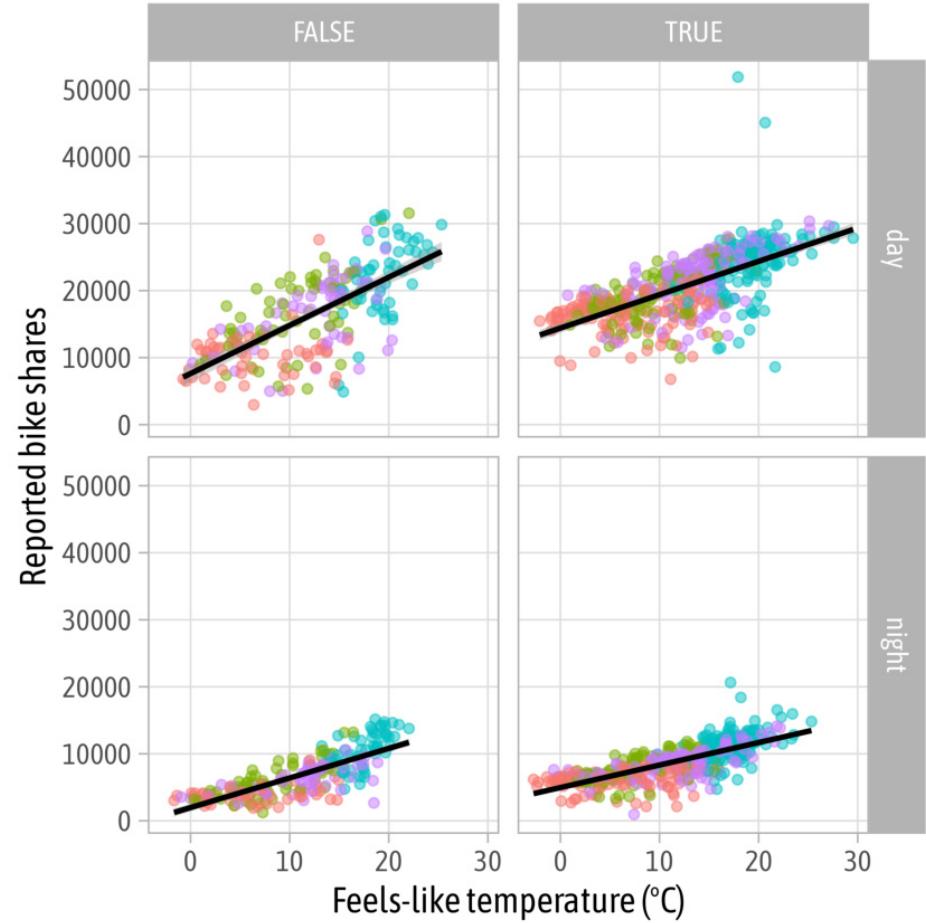


Gridded Facets

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

TfL bike sharing trends

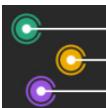
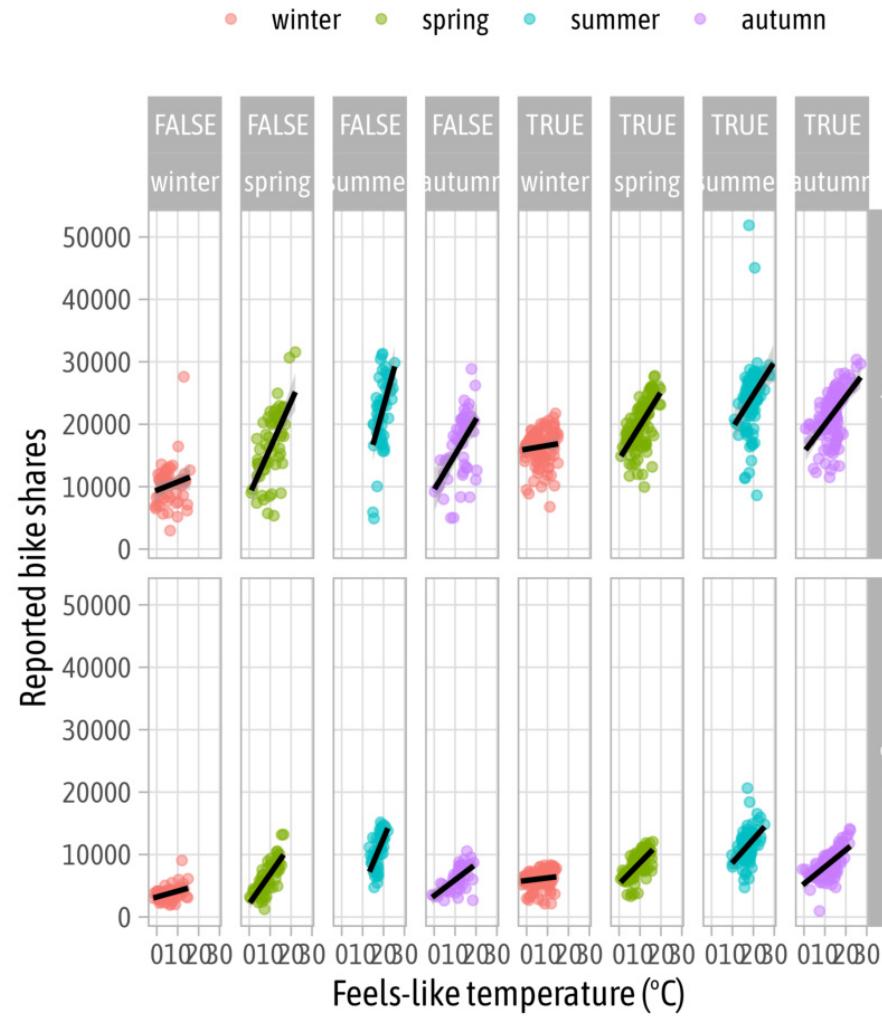
● winter ● spring ● summer ● autumn



Facet Multiple Variables

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

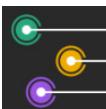
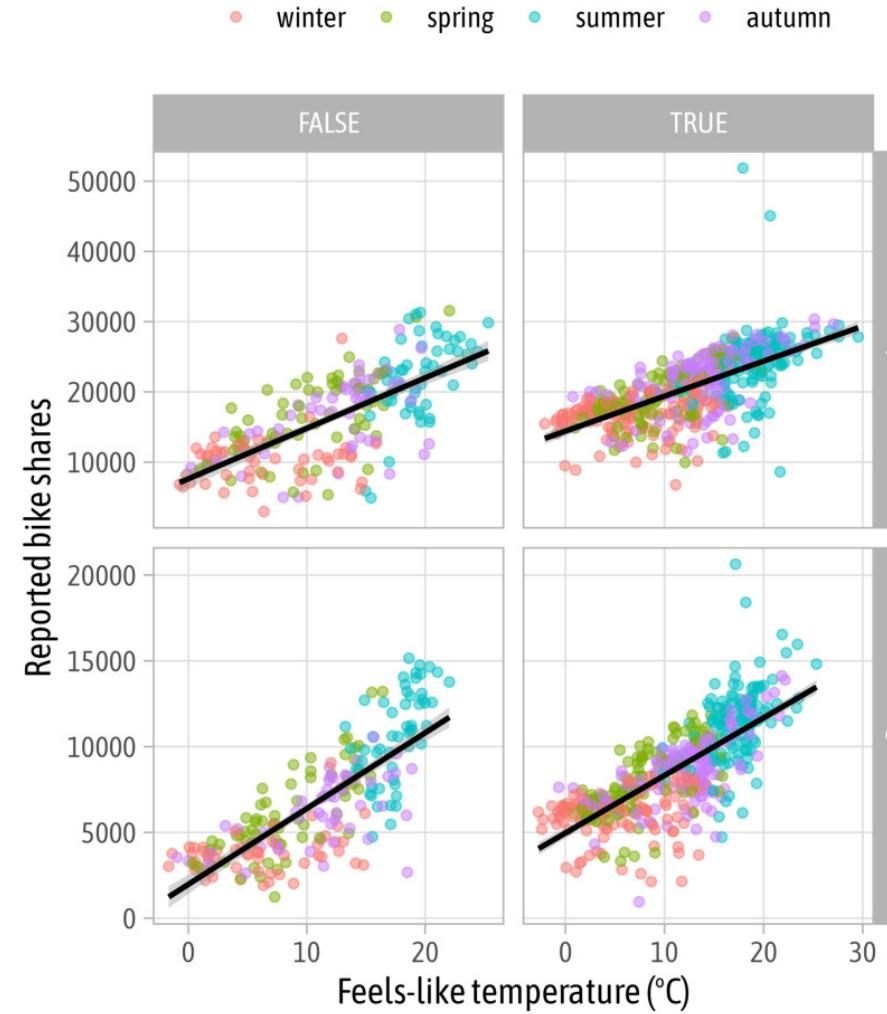
TfL bike sharing trends



Facet Options: Free Scaling

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

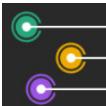
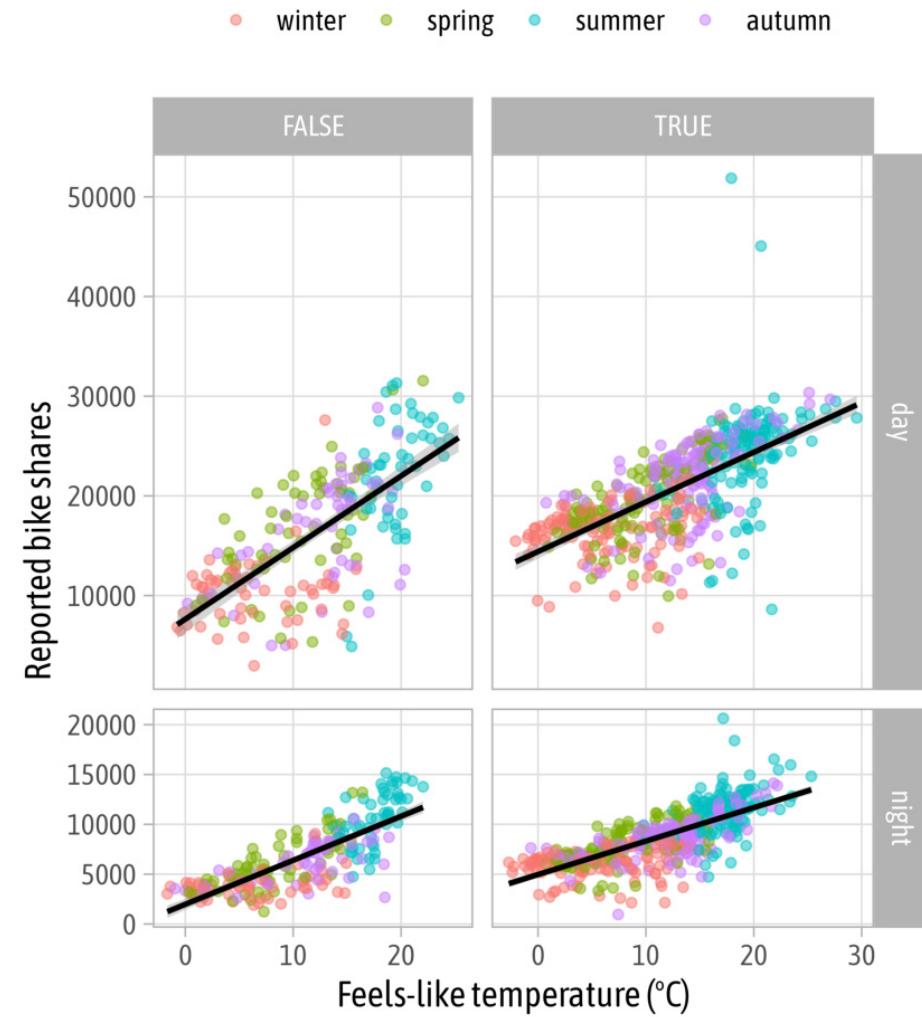
TfL bike sharing trends



Facet Options: Proportional Spacing

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

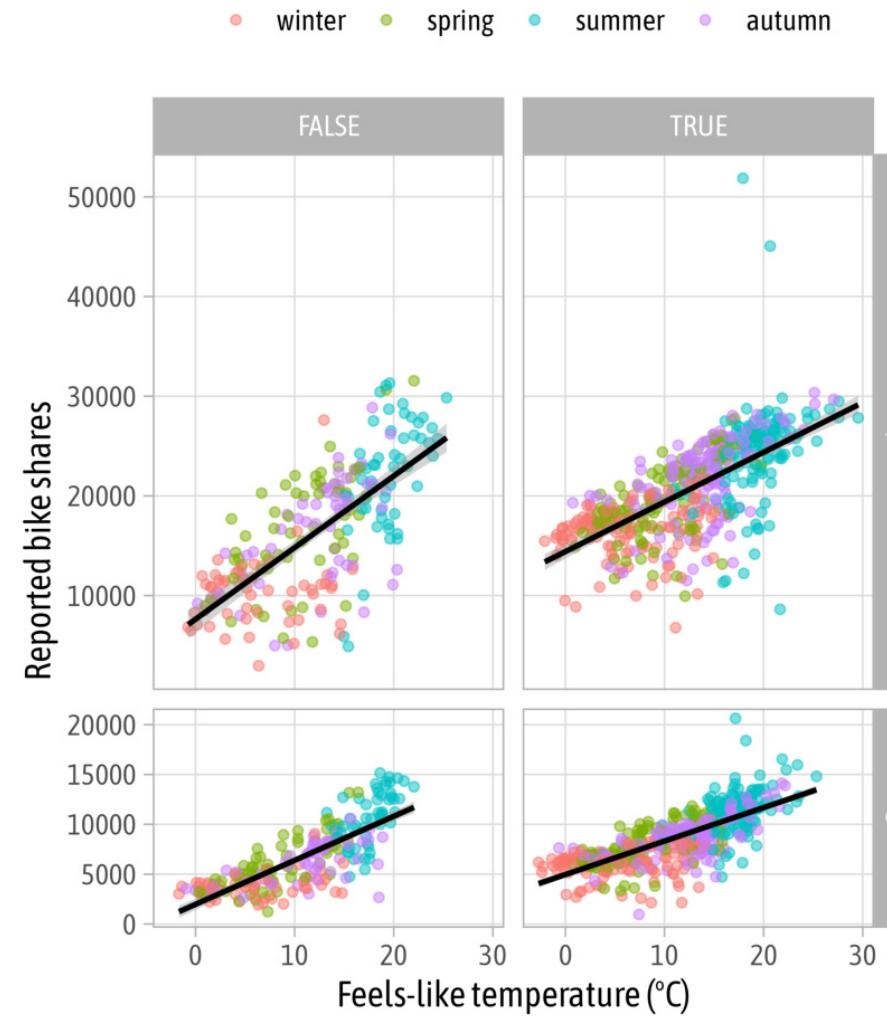
TfL bike sharing trends



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

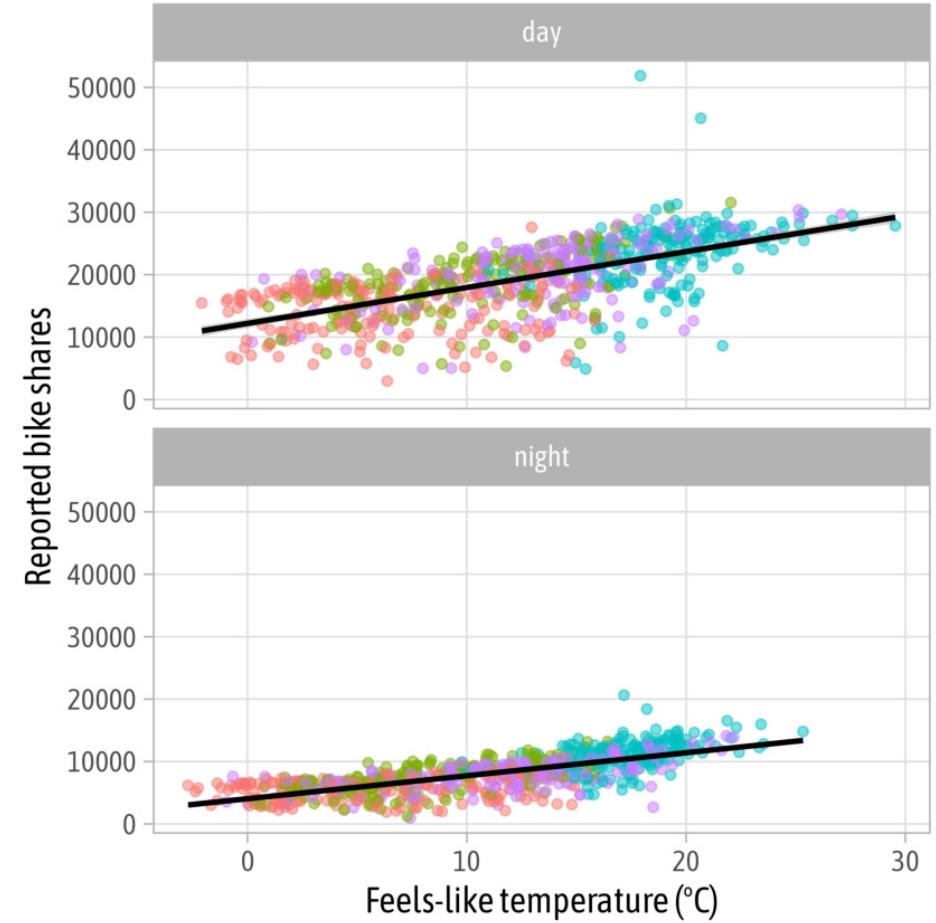


Facet Options: Cols + Rows

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

TfL bike sharing trends

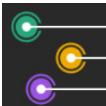
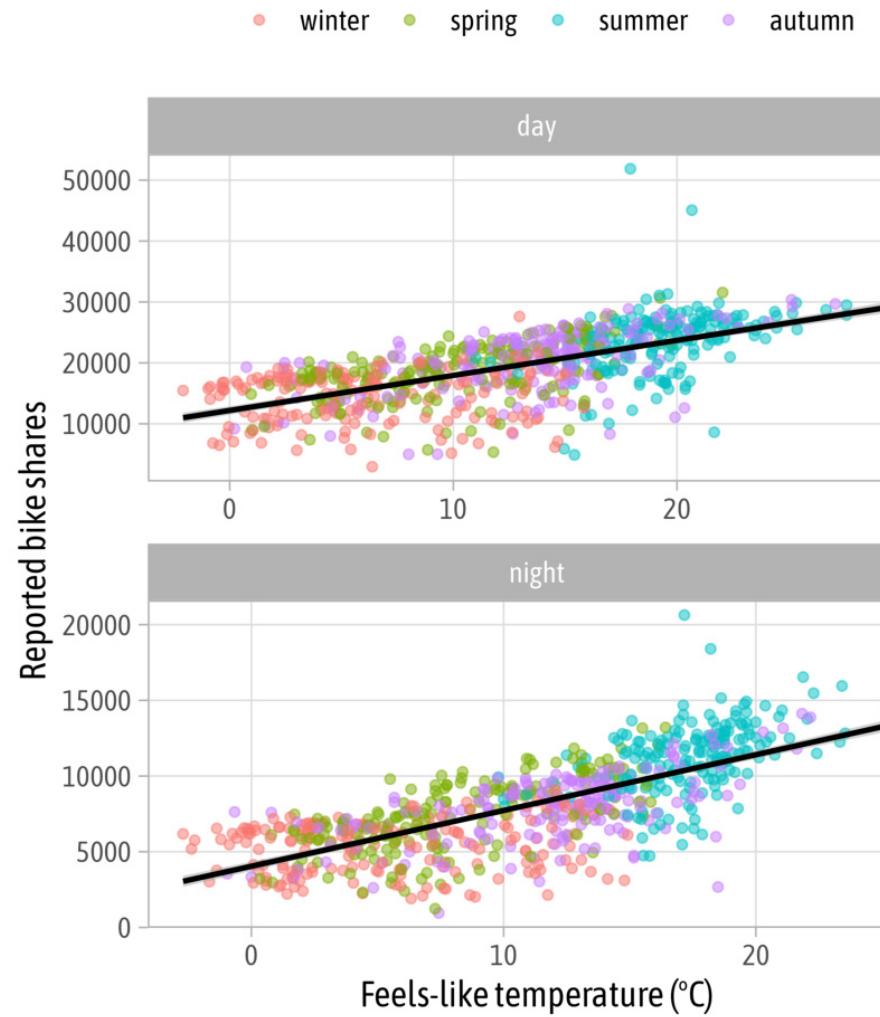
● winter ● spring ● summer ● autumn



Facet Options: Free Scaling

```
1 g +
2   facet_wrap(
3     ~ day_night,
4     ncol = 1,
5     scales = "free"
6   )
```

TfL bike sharing trends

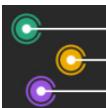
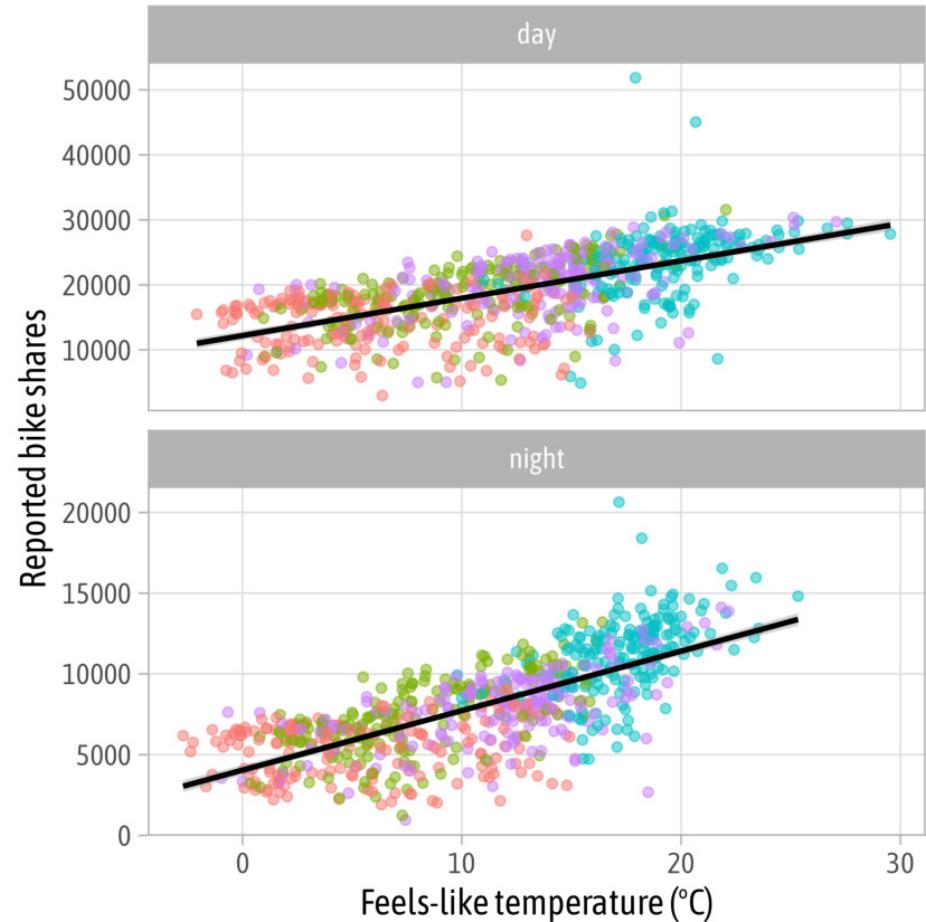


Facet Options: Free Scaling

```
1 g +  
2   facet_wrap(  
3     ~ day_night,  
4     ncol = 1,  
5     scales = "free_y"  
6   )
```

TfL bike sharing trends

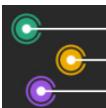
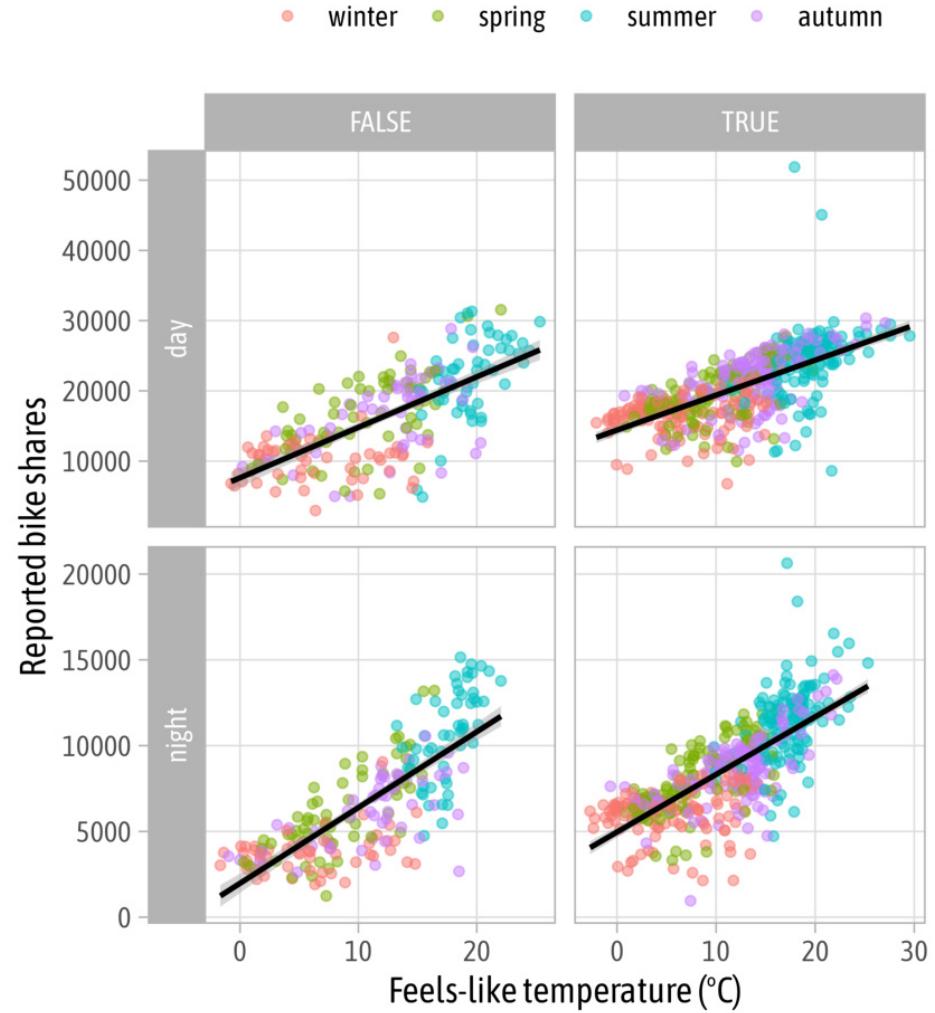
● winter ● spring ● summer ● autumn



Facet Options: Switch Labels

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

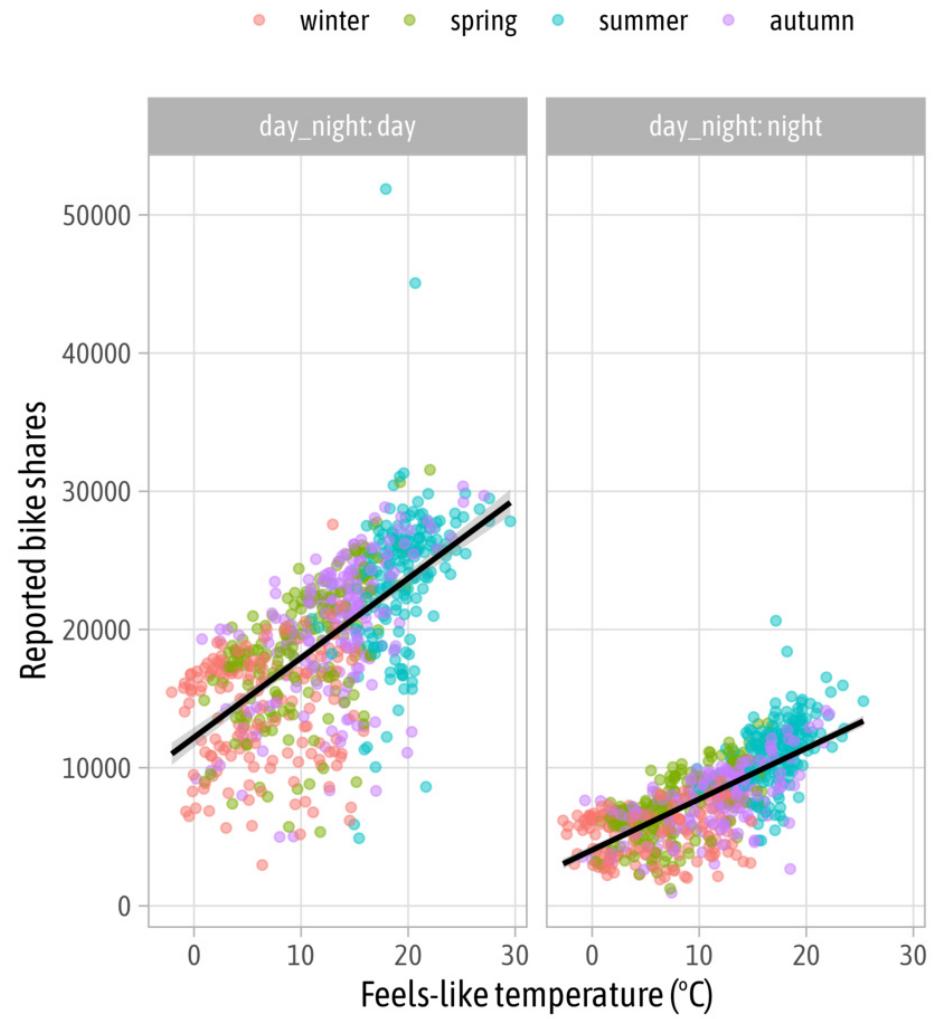
TfL bike sharing trends



Facet Labellers

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

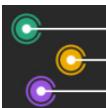
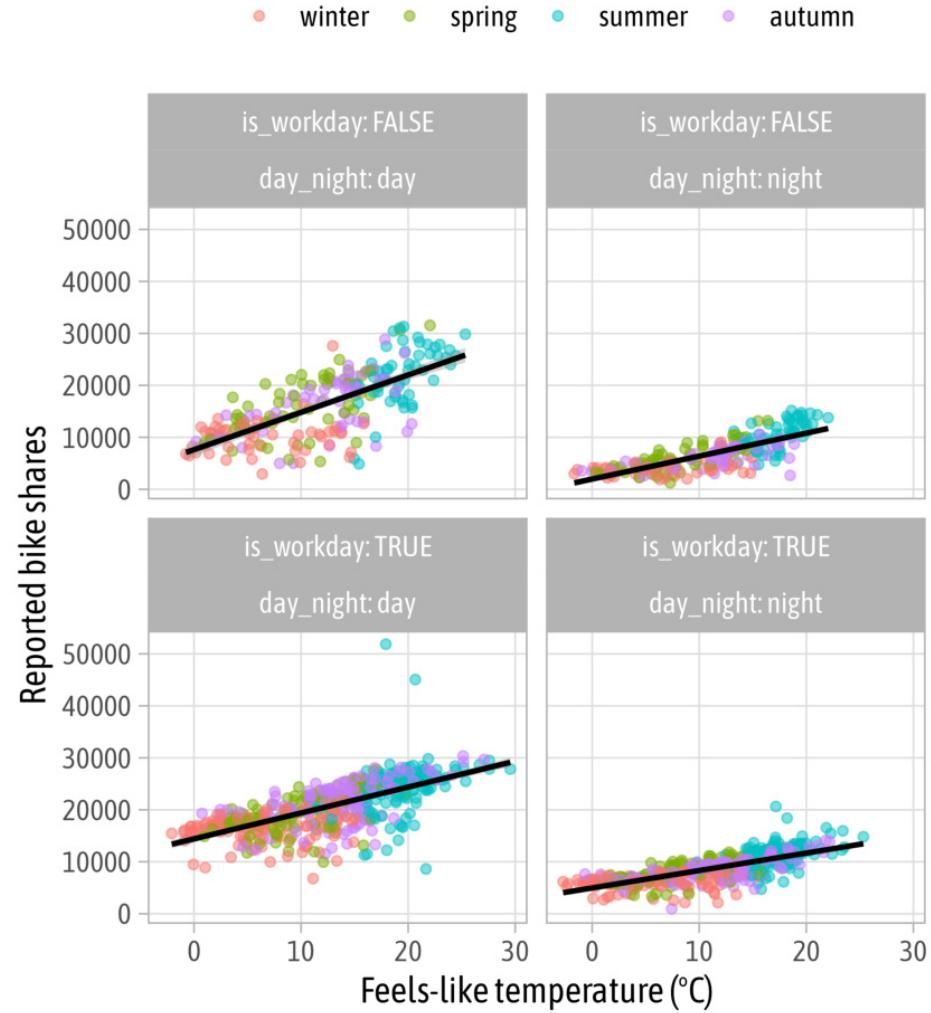
TfL bike sharing trends



Facet Labellers

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

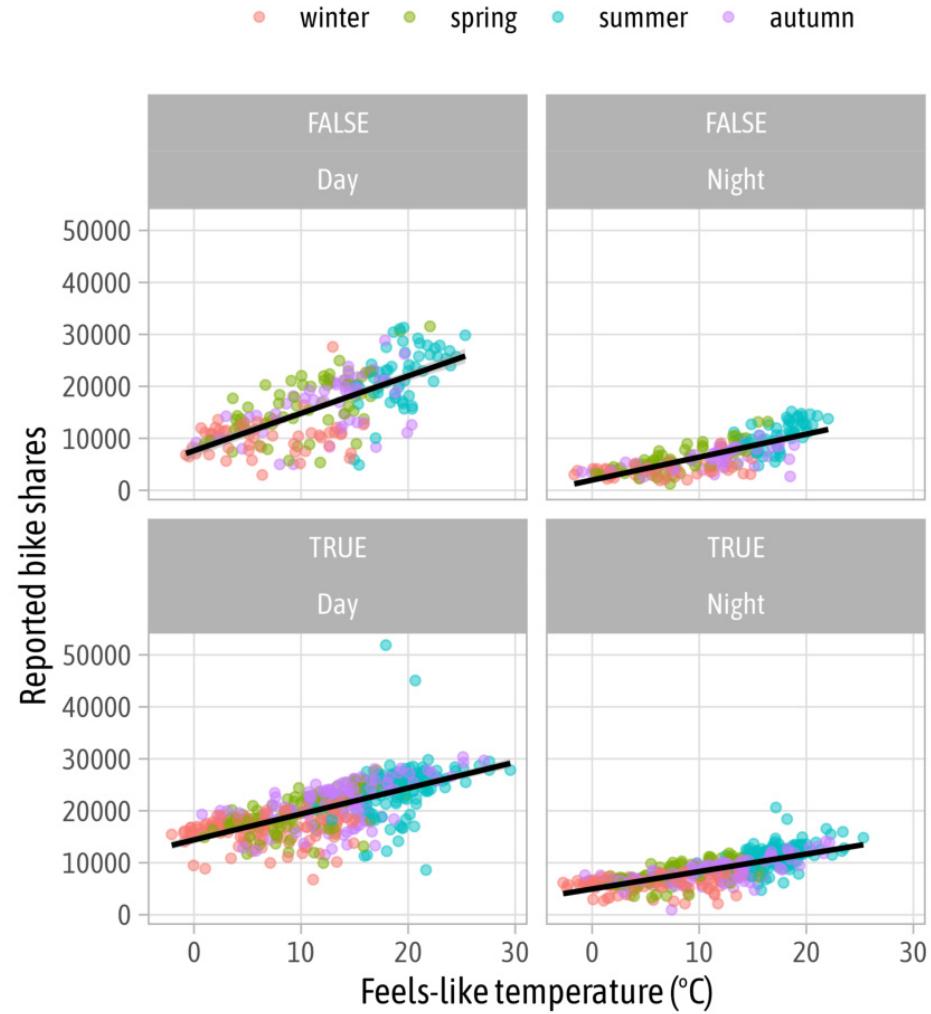
TfL bike sharing trends



Facet Labellers

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

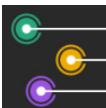
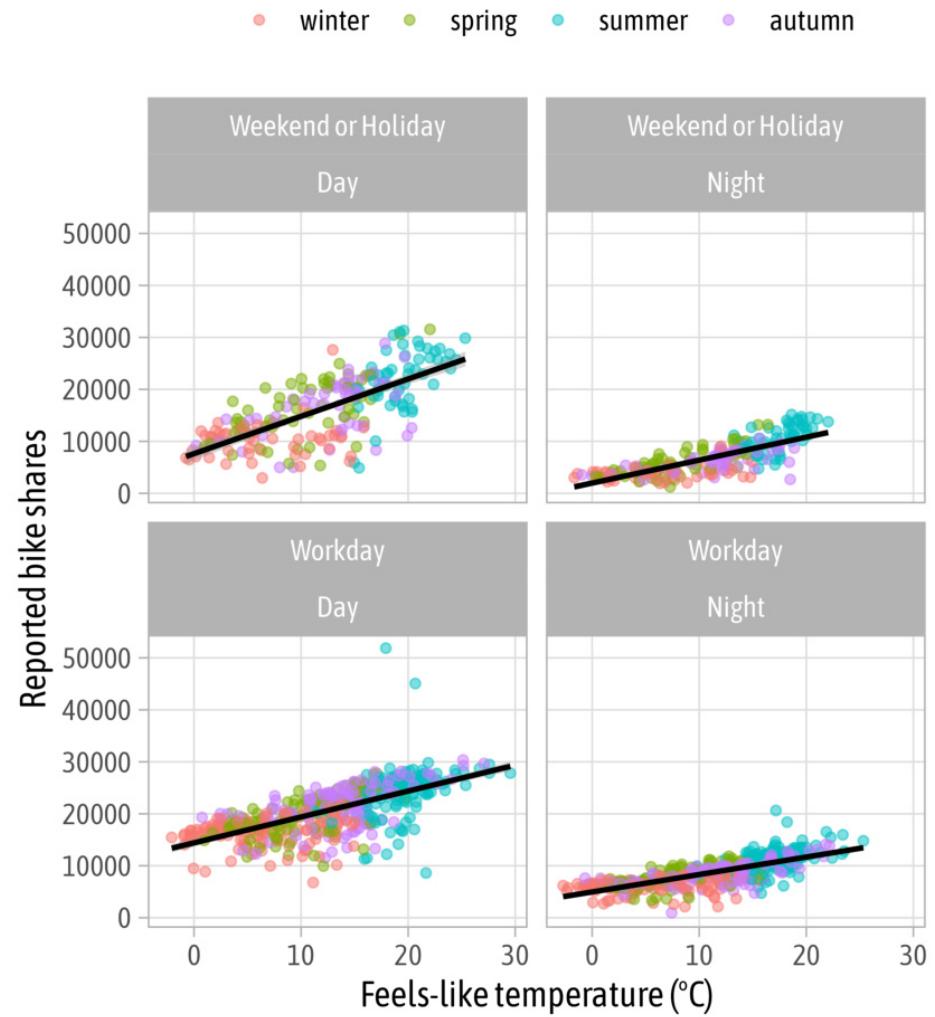
TfL bike sharing trends



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 )
```

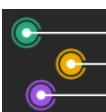
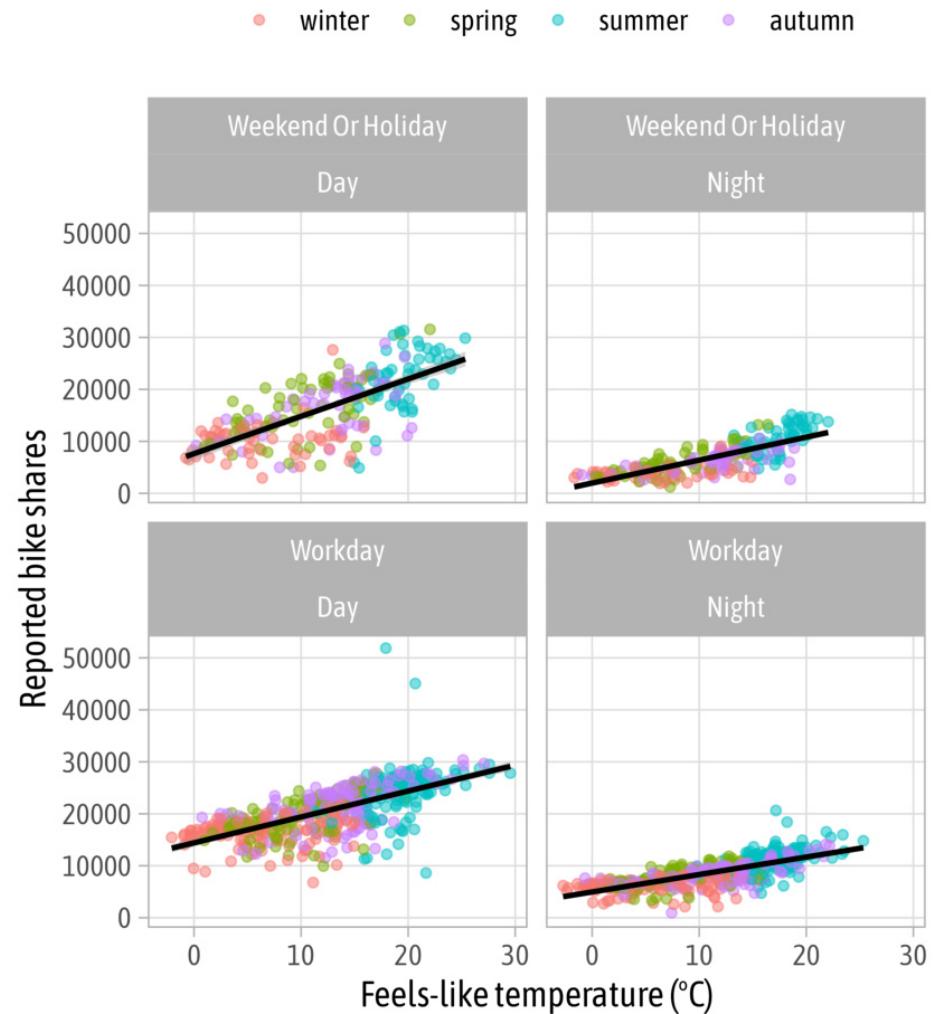
TfL bike sharing trends



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    .default = stringr::str_to_title,  
11    is_workday = codes  
12  ))  
13 )
```

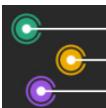
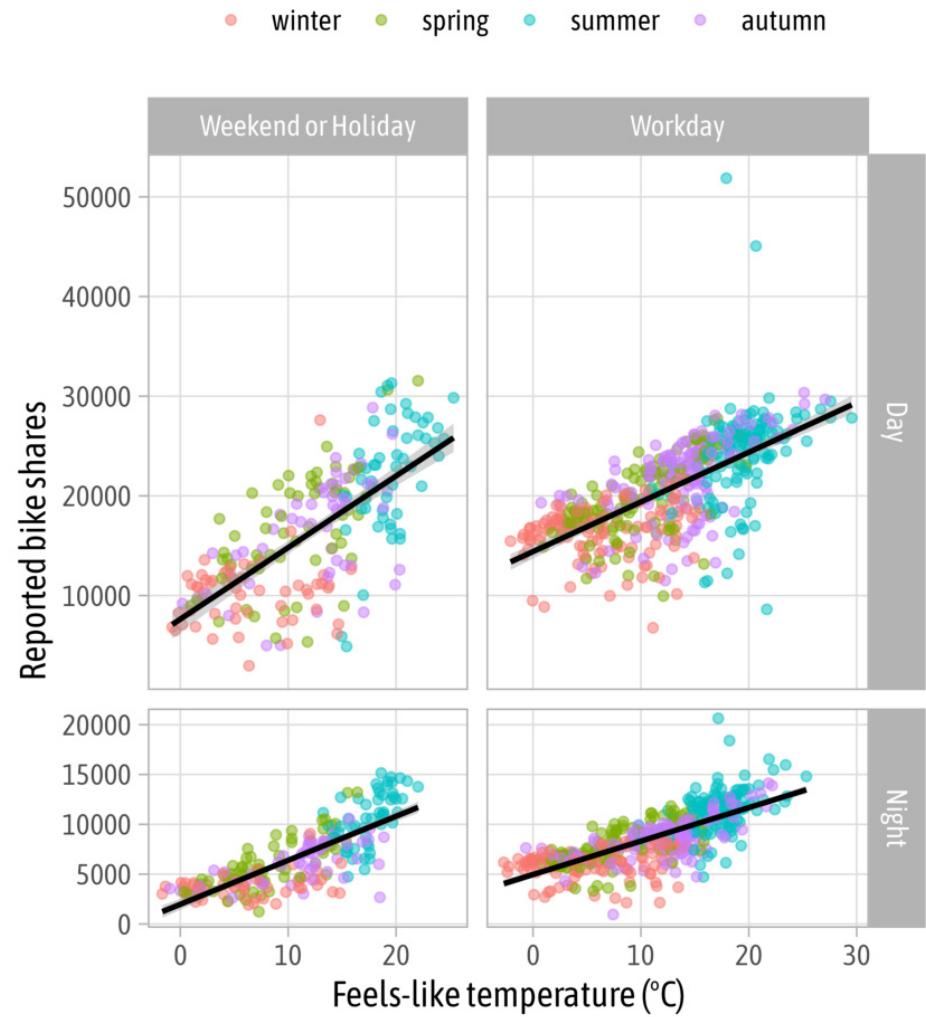
TfL bike sharing trends



Facet Labeller

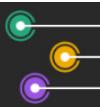
```
1 g +
2   facet_grid(
3     day_night ~ is_workday,
4     scales = "free",
5     space = "free",
6     labeller = labeller(
7       day_night = stringr::str_to_title,
8       is_workday = codes
9     )
10 )
```

TfL bike sharing trends



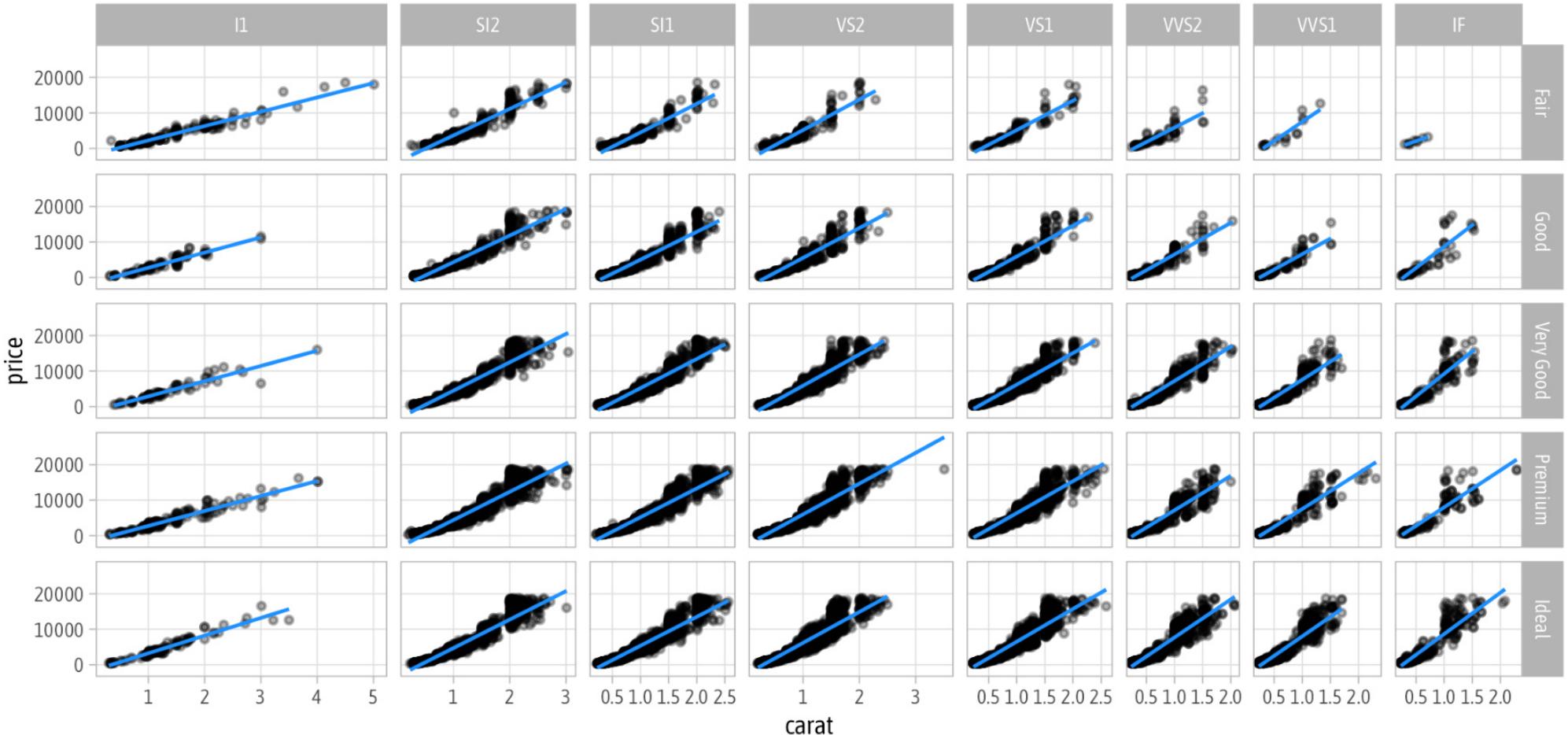
Data Visualization with {ggplot2}

— Exercise —



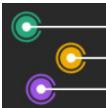
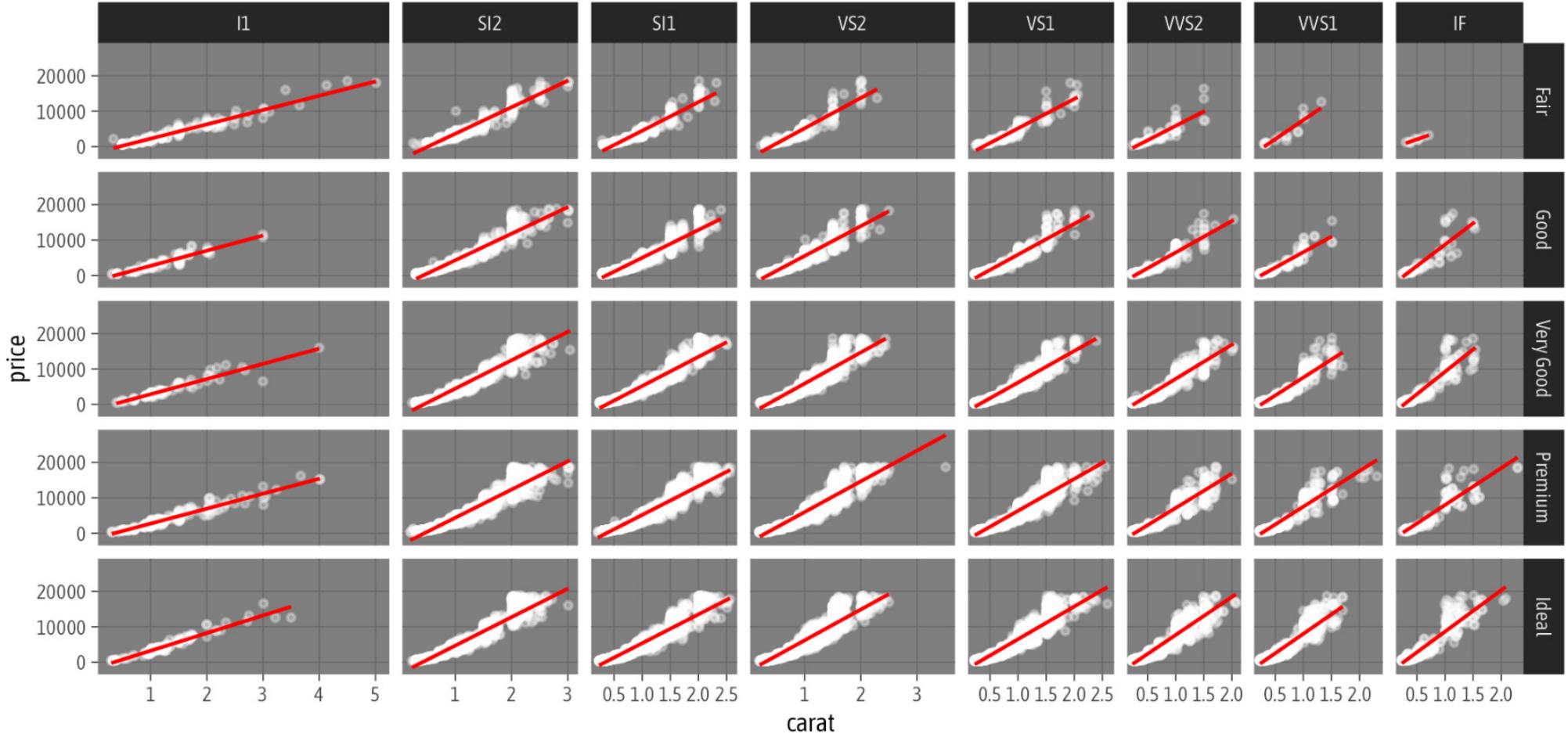
Your Turn: Diamonds Facet

Create the following facet from the `diamonds` data.



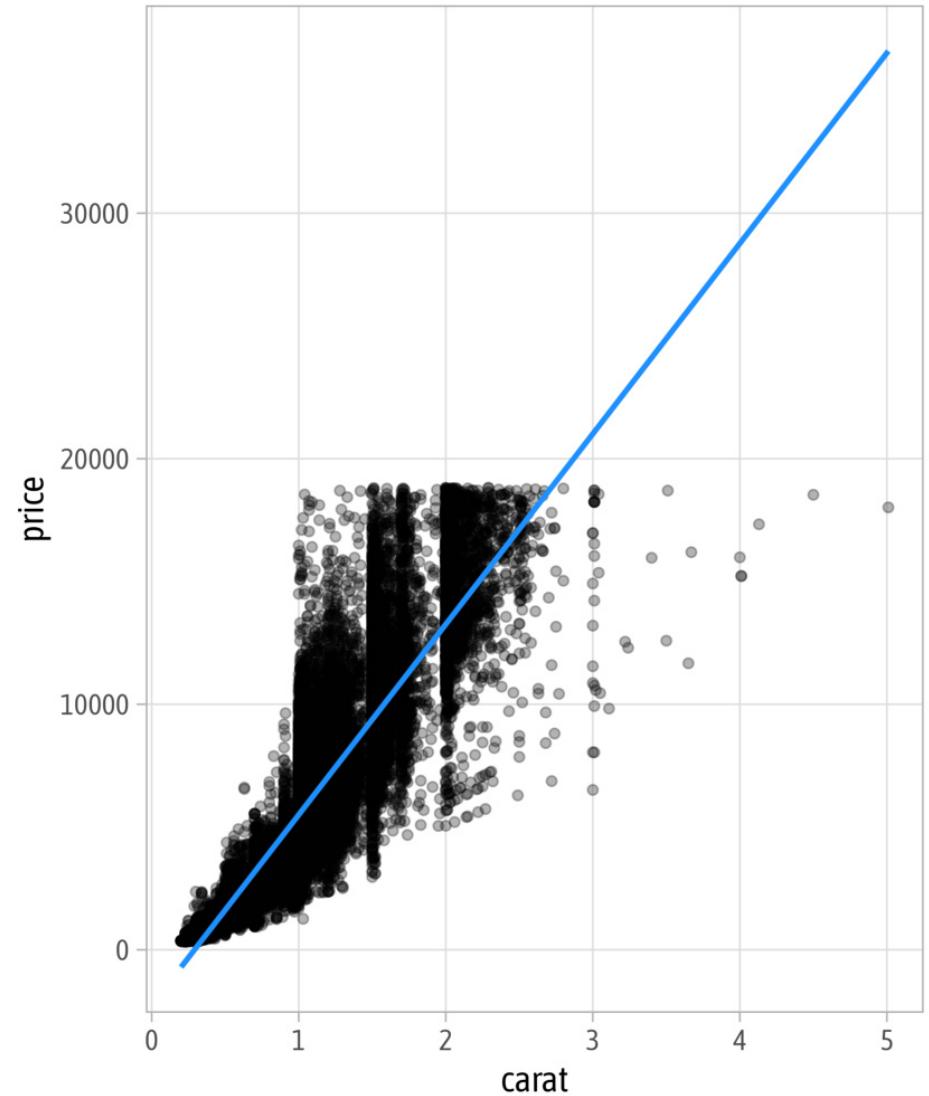
Your Turn: Diamonds Facet

Bonus: Create this bloody-dark version.



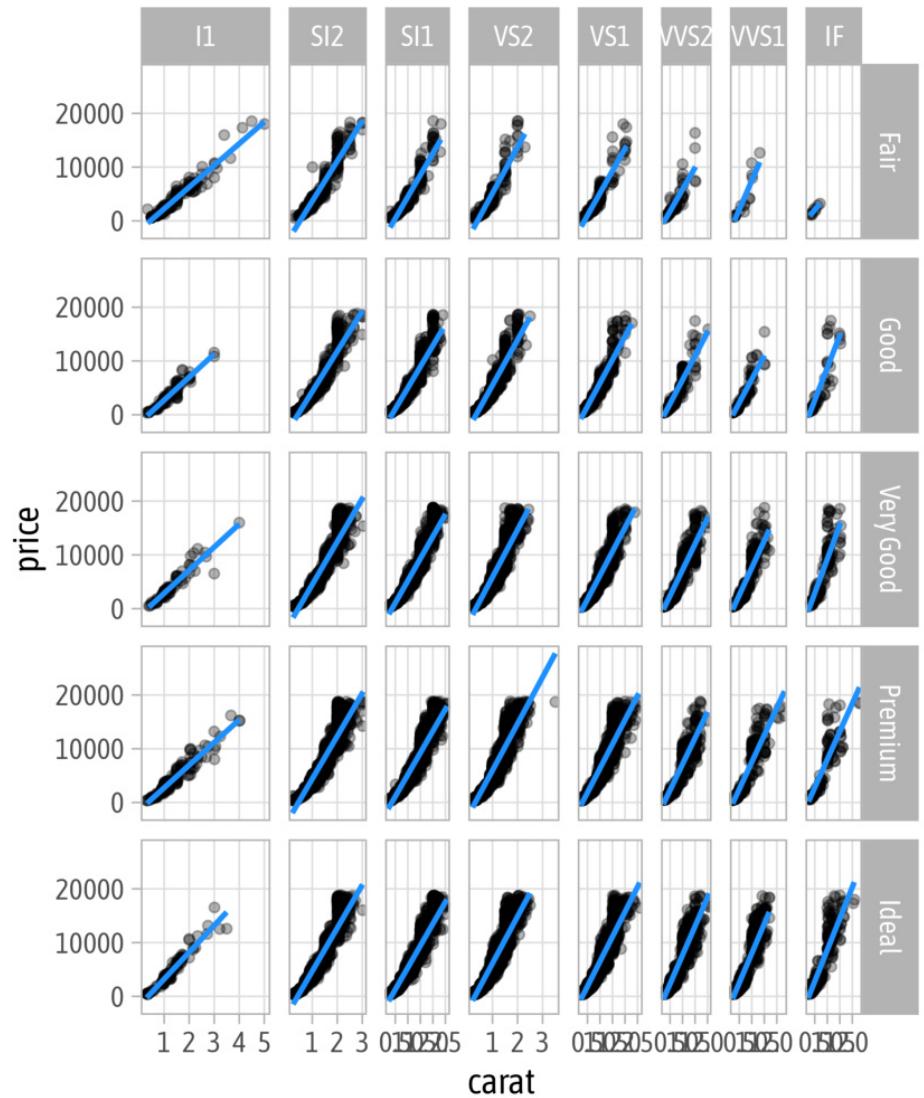
Your Turn: Diamonds Facet

```
1 ggplot(  
2   diamonds,  
3   aes(x = carat, y = price)  
4 ) +  
5 geom_point(  
6   alpha = .3  
7 ) +  
8 geom_smooth(  
9   method = "lm",  
10  se = FALSE,  
11  color = "dodgerblue"  
12 )
```



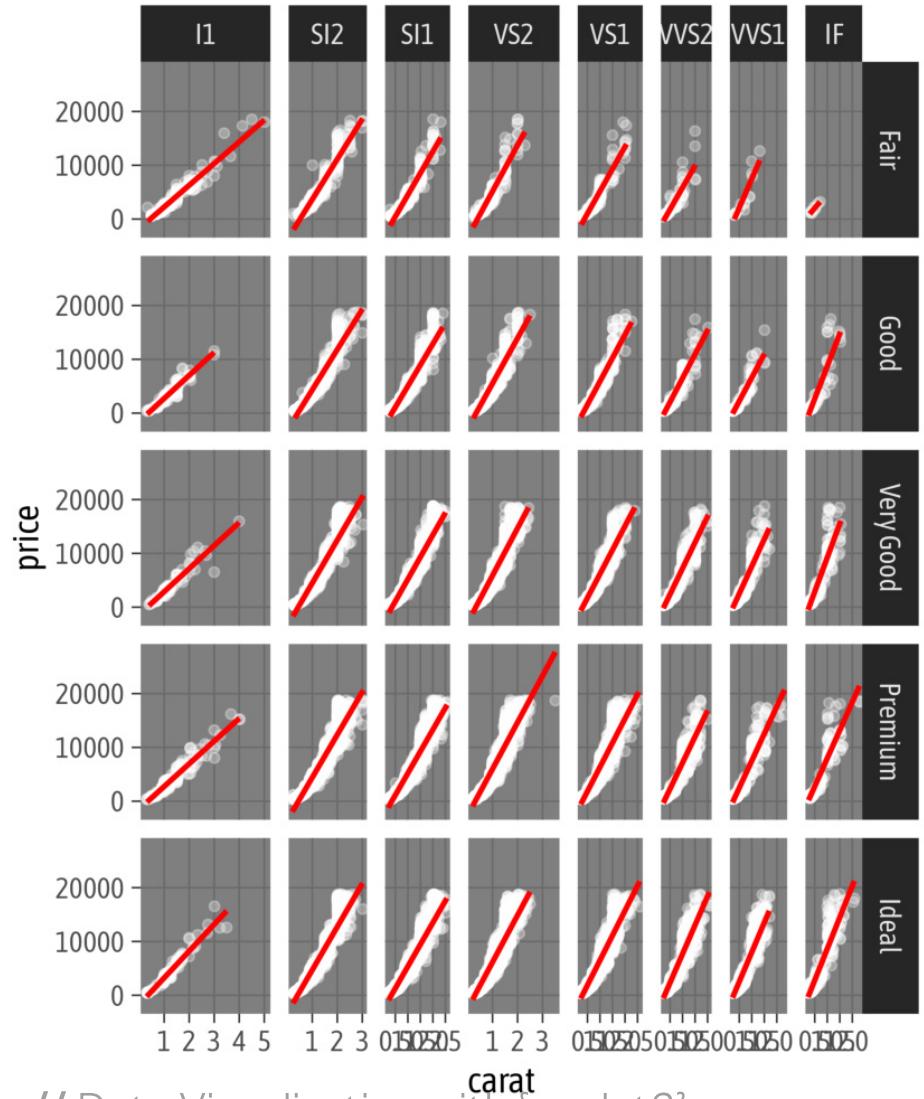
Your Turn: Diamonds Facet

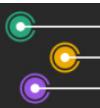
```
1 ggplot(  
2   diamonds,  
3   aes(x = carat, y = price)  
4 ) +  
5 geom_point(  
6   alpha = .3  
7 ) +  
8 geom_smooth(  
9   method = "lm",  
10  se = FALSE,  
11  color = "dodgerblue"  
12 ) +  
13 facet_grid(  
14   cut ~ clarity,  
15   space = "free_x",  
16   scales = "free_x"  
17 )
```



Your Turn: Diamonds Facet (Dark Theme Bonus)

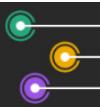
```
1 ggplot(  
2   diamonds,  
3   aes(x = carat, y = price)  
4 ) +  
5   geom_point(  
6     alpha = .3,  
7     color = "white"  
8 ) +  
9   geom_smooth(  
10    method = "lm",  
11    se = FALSE,  
12    color = "red"  
13 ) +  
14   facet_grid(  
15     cut ~ clarity,  
16     space = "free_x",  
17     scales = "free_x"  
18 ) +  
19   theme_dark(  
20     base_size = 14,  
21     base_family = "Asap Condensed"
```





Data Visualization with {ggplot2}

— Scales —



Scales

= translate between variable and property ranges

- feels-like temperature \Leftrightarrow x
- reported bike shares \Leftrightarrow y
- season \Leftrightarrow color
- year \Leftrightarrow 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_*`()



Scales

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

The extensions (*) can be filled by e.g.:

- `continuous()`, `discrete()`, `reverse()`, `log10()`, `sqrt()`, `date()` for positions
- `continuous()`, `discrete()`, `manual()`, `gradient()`, `gradient2()`, `brewer()` for colors
- `continuous()`, `discrete()`, `manual()`, `ordinal()`, `area()`, `date()` for sizes
- `continuous()`, `discrete()`, `manual()`, `ordinal()` for shapes
- `continuous()`, `discrete()`, `manual()`, `ordinal()`, `date()` for transparency



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.

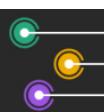


I HAVE 8 LEGS
and
4 SPOTS!

@allison_horst

Illustration by Allison Horst

Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}



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)

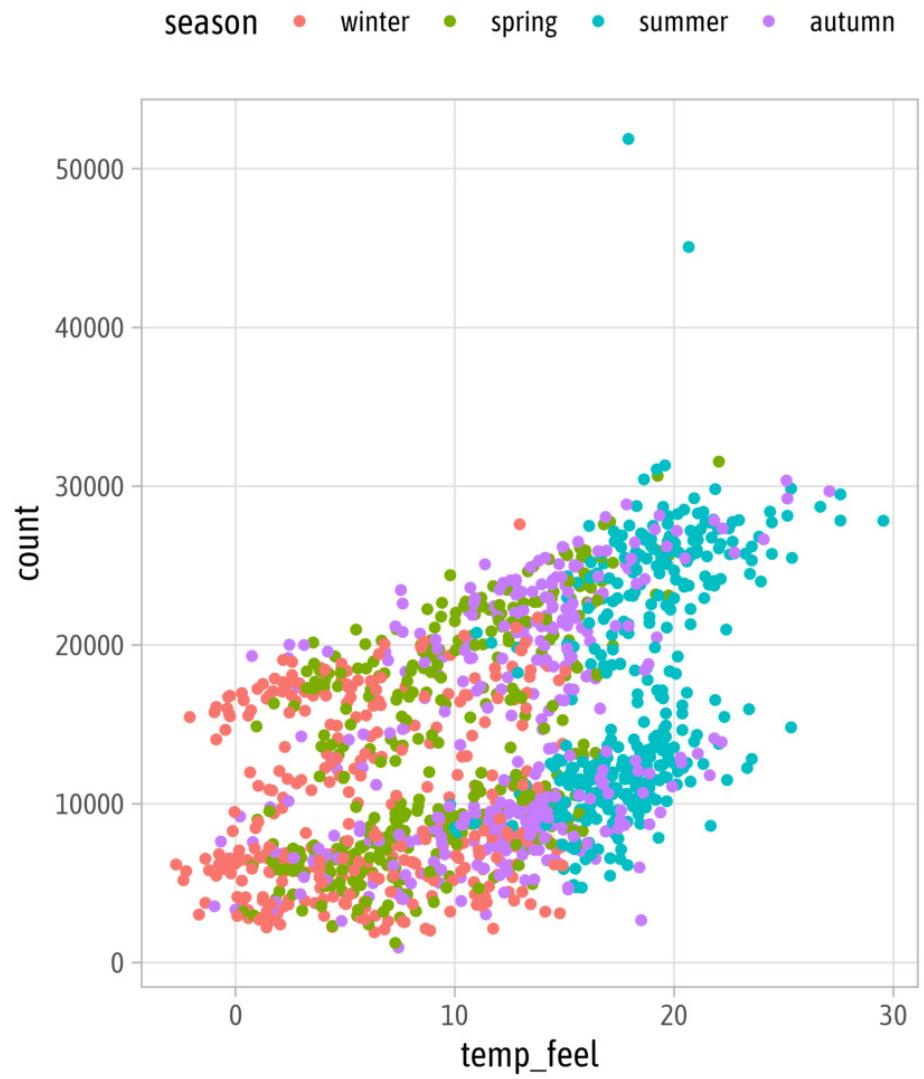
Discrete:
qualitative or categorical data

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



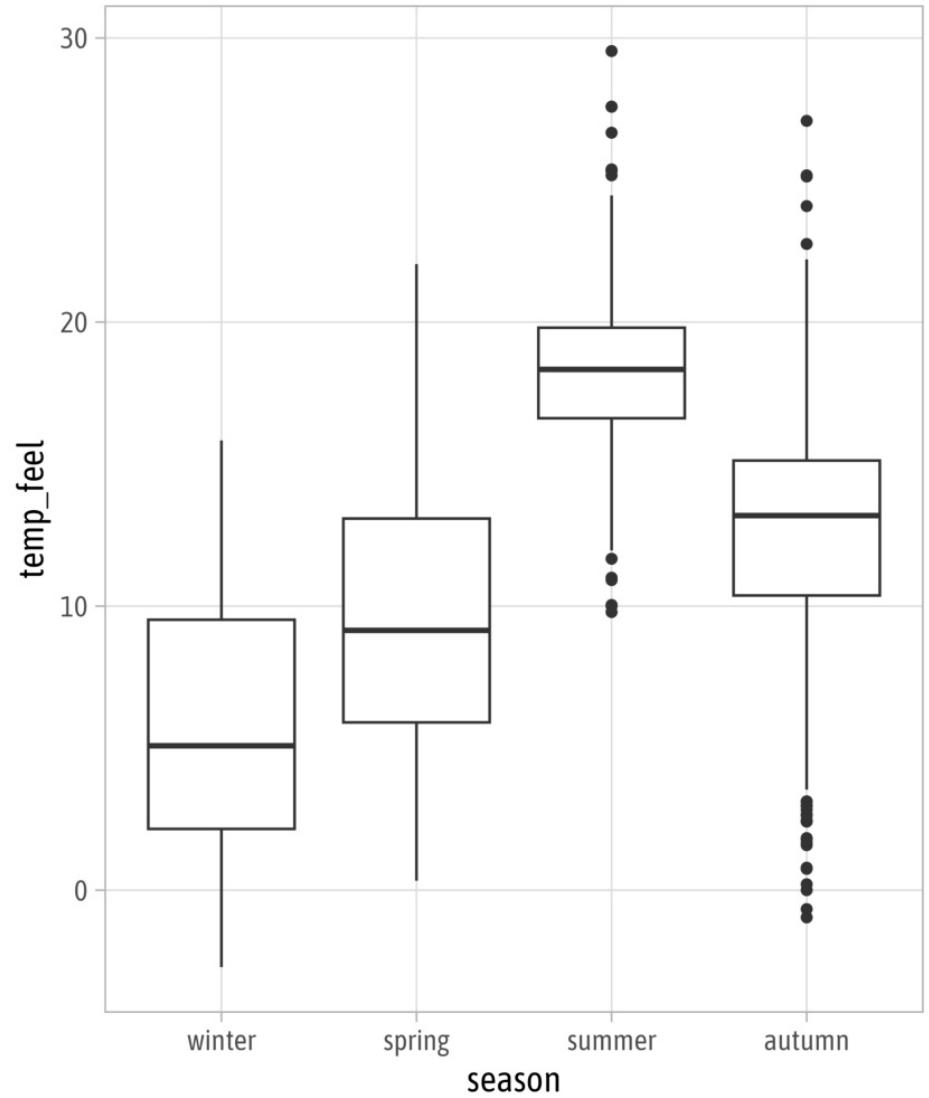
Aesthetics + Scales

```
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()
```

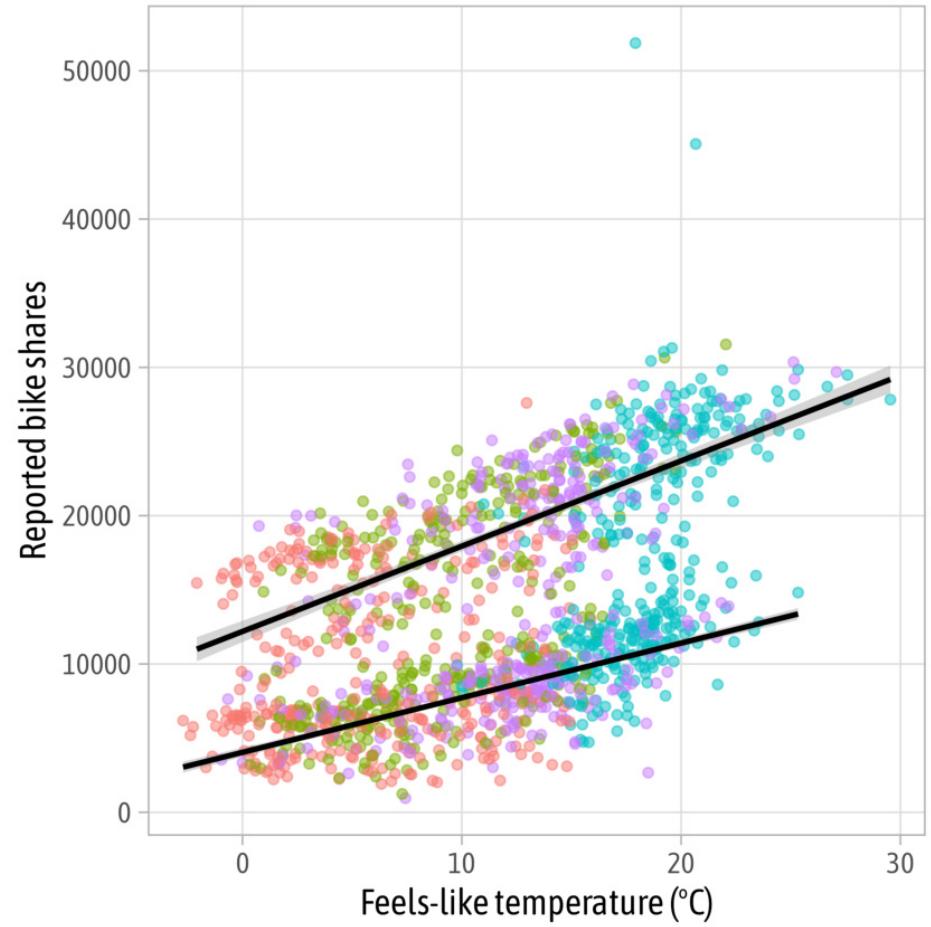


Aesthetics + Scales

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

TfL bike sharing trends

● winter ● spring ● summer ● autumn

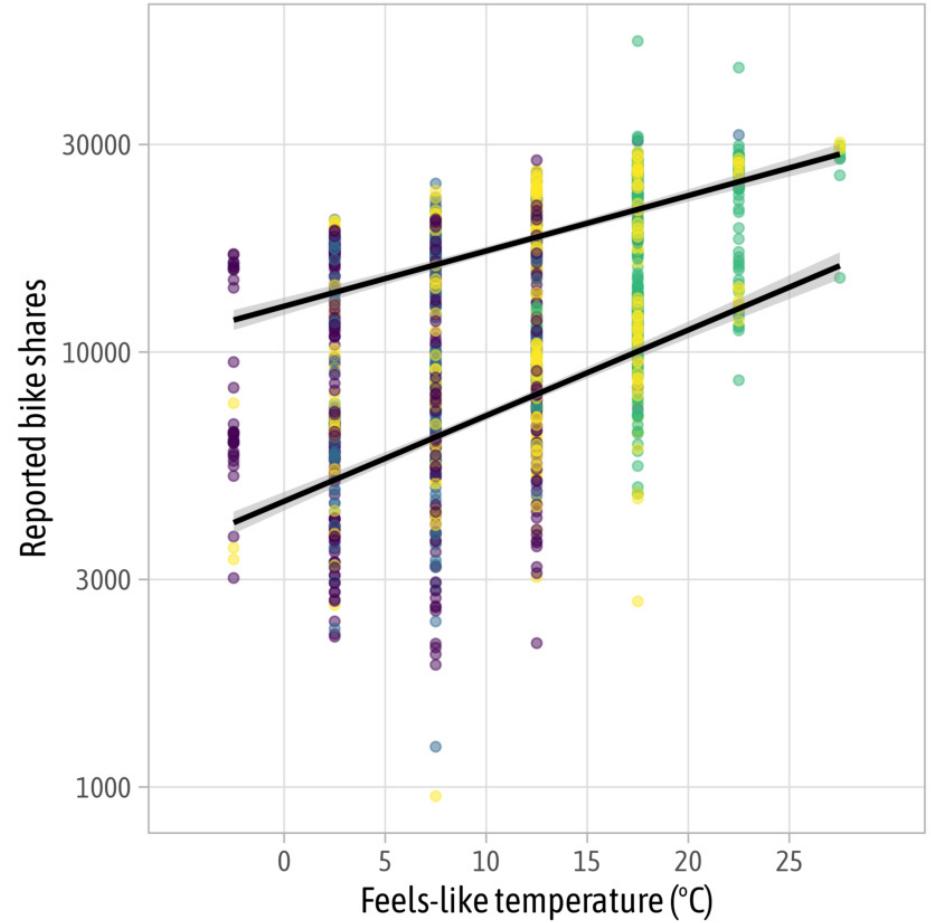


Overwrite Scales

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

TfL bike sharing trends

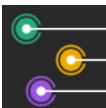
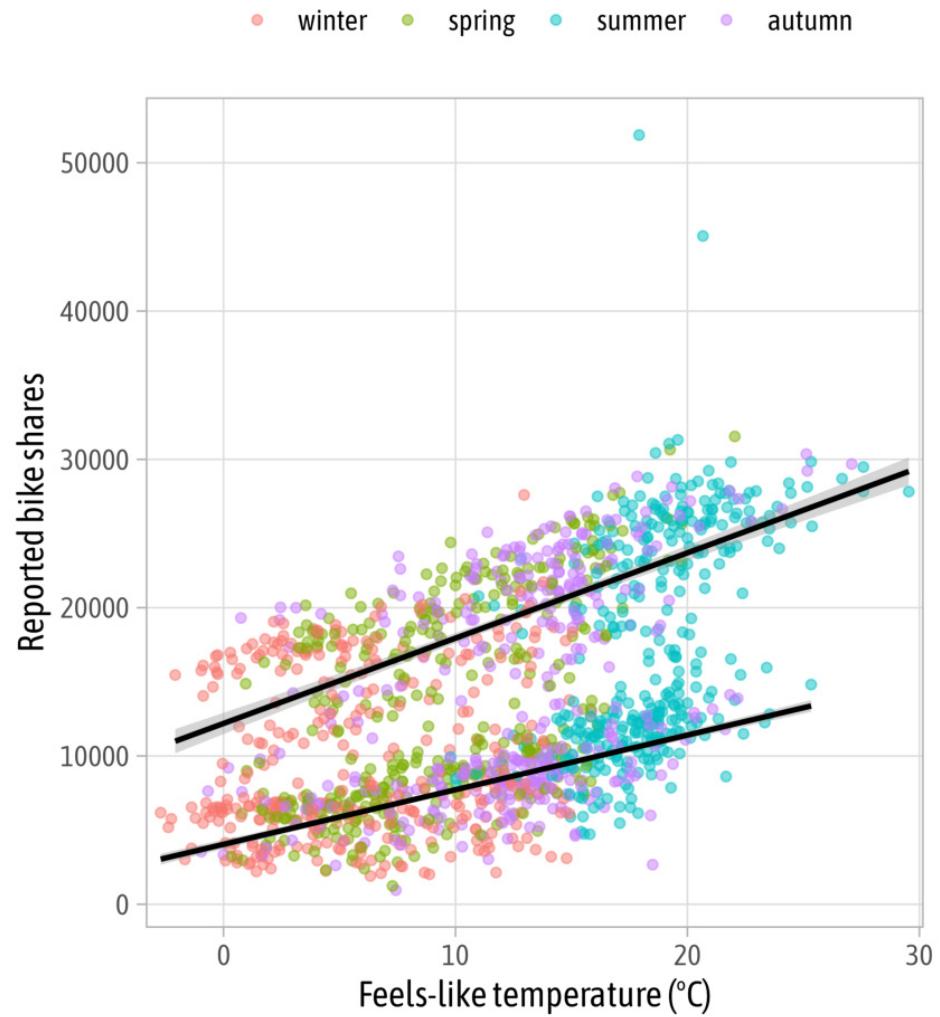
● winter ● spring ● summer ● autumn



Modify Scales

```
1 g +  
2   scale_x_continuous(  
3     expand = c(mult = 0.02, add = 0)  
4   ) +  
5   scale_y_continuous() +  
6   scale_color_discrete()
```

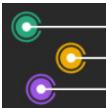
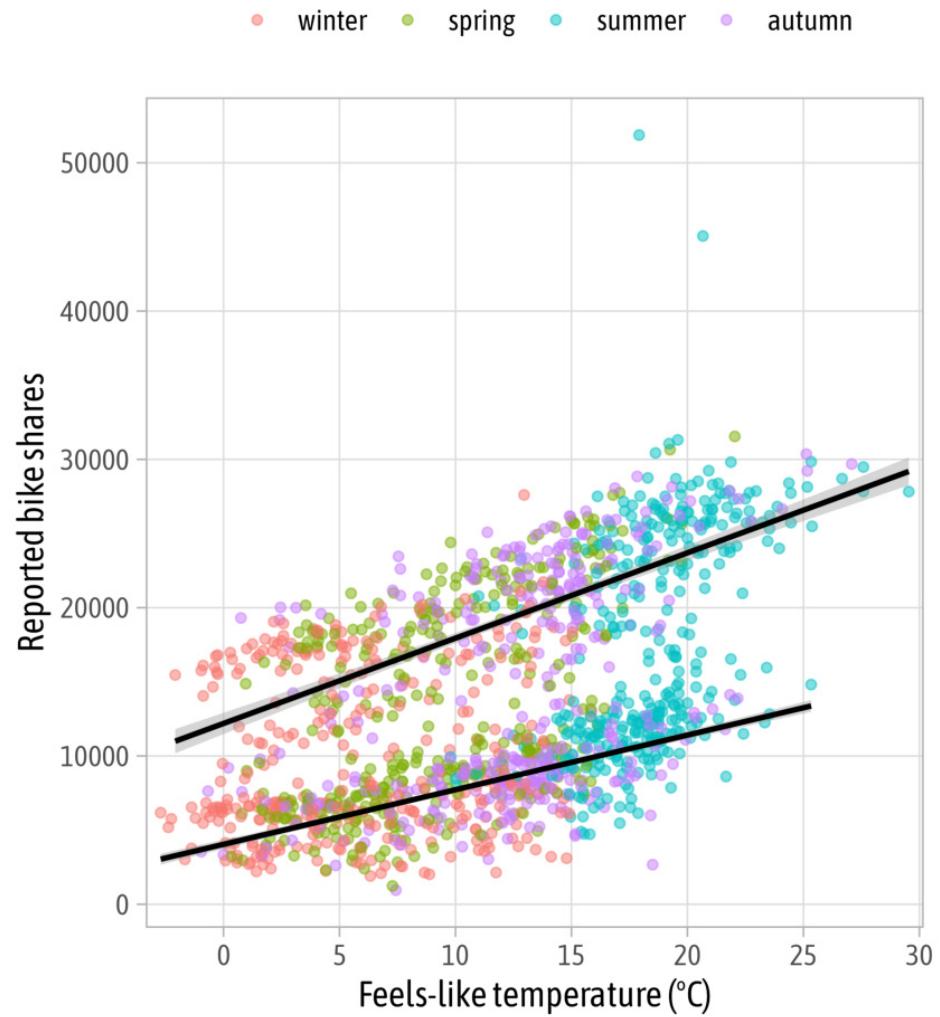
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0.02, add = 0),
4     breaks = seq(0, 30, by = 5)
5   ) +
6   scale_y_continuous() +
7   scale_color_discrete()
```

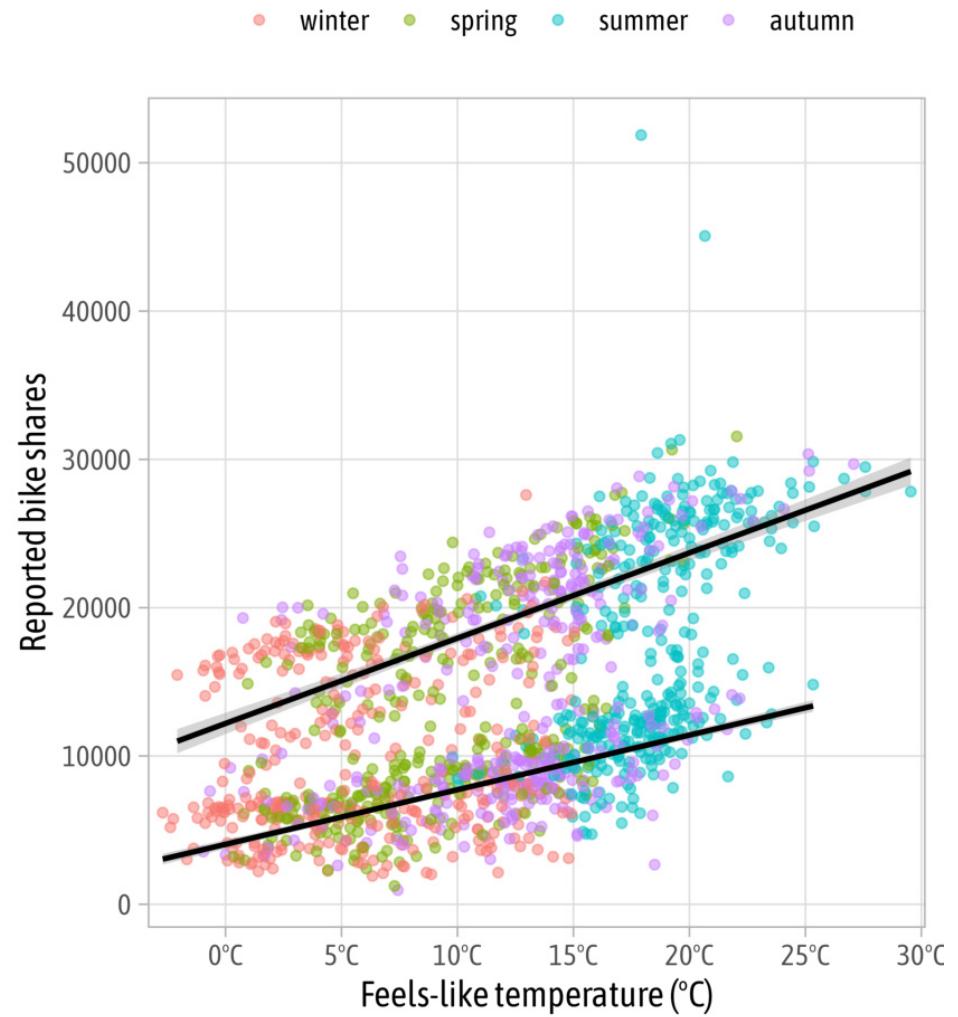
TfL bike sharing trends



Modify Scales

```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0.02, 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()
```

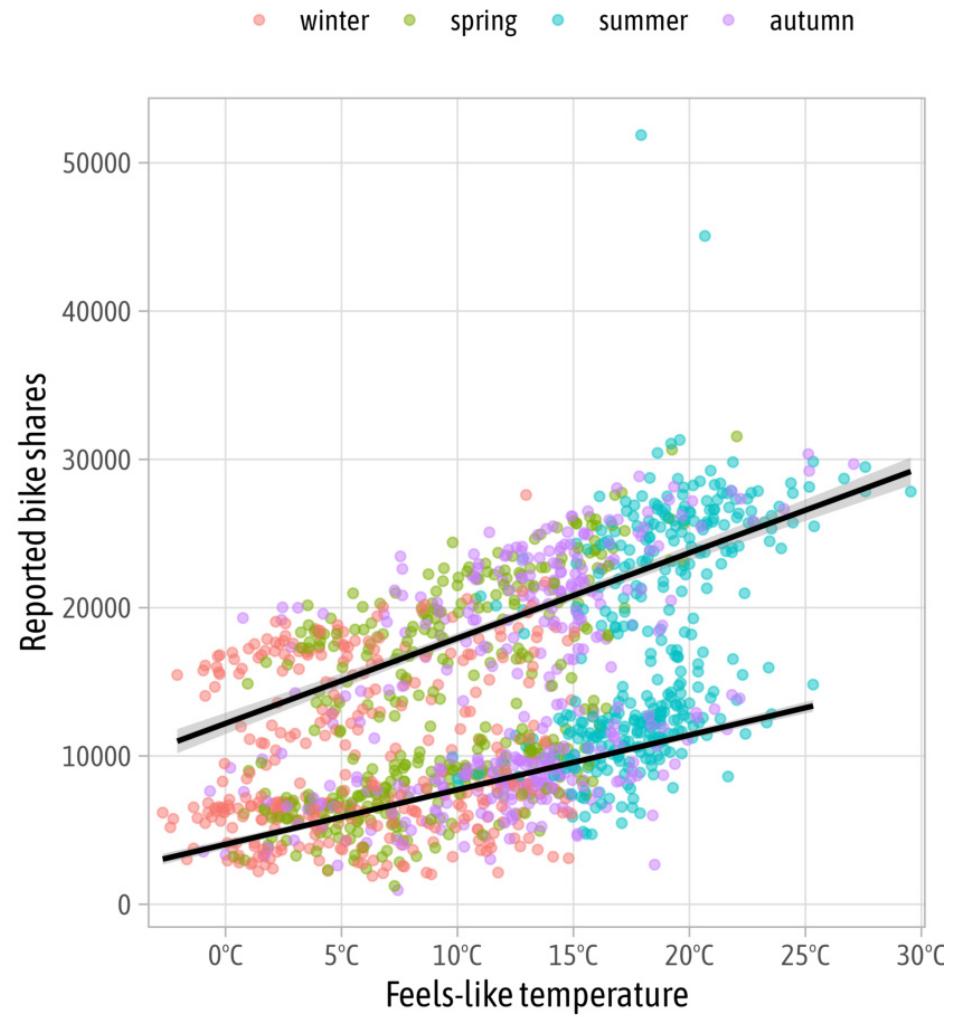
TfL bike sharing trends



Modify Scales

```
1 g +  
2   scale_x_continuous(  
3     expand = c(mult = 0.02, 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   scale_color_discrete()
```

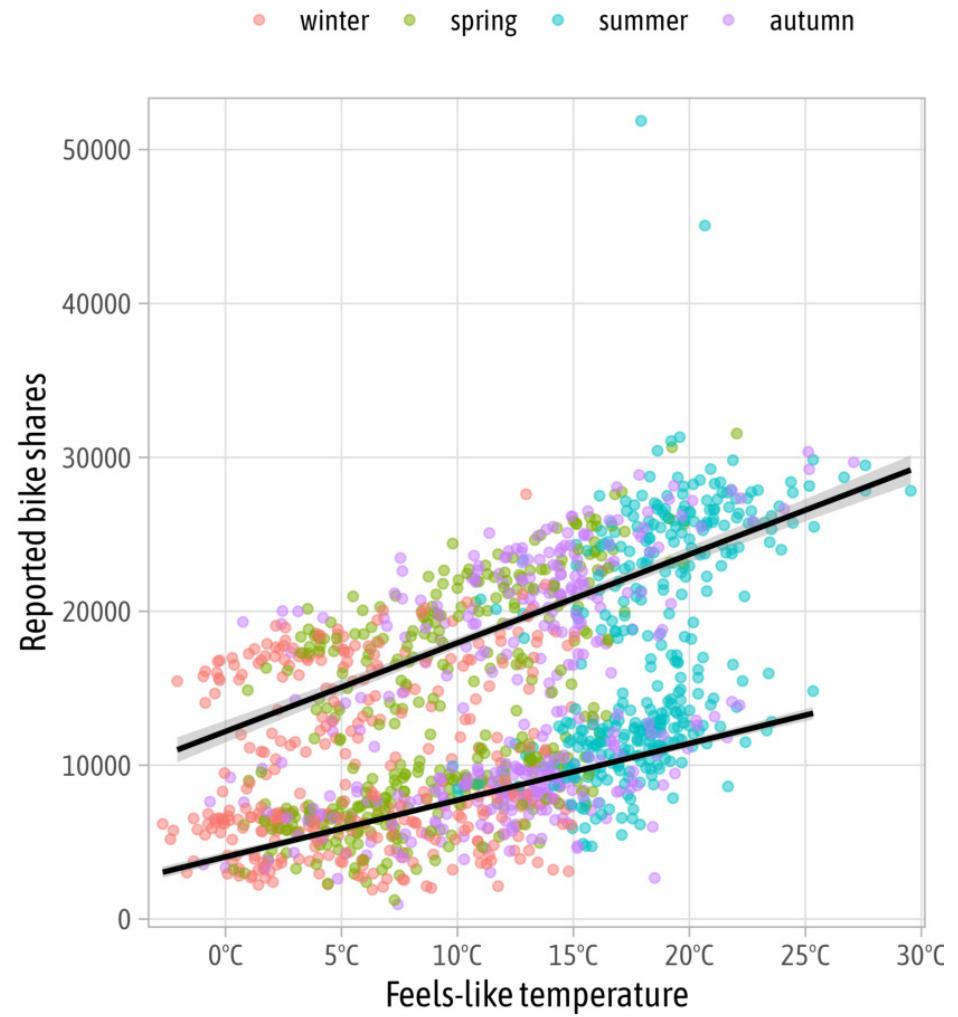
TfL bike sharing trends



Modify Scales

```
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 = .03, add = 0)
10  ) +
11  scale_color_discrete()
```

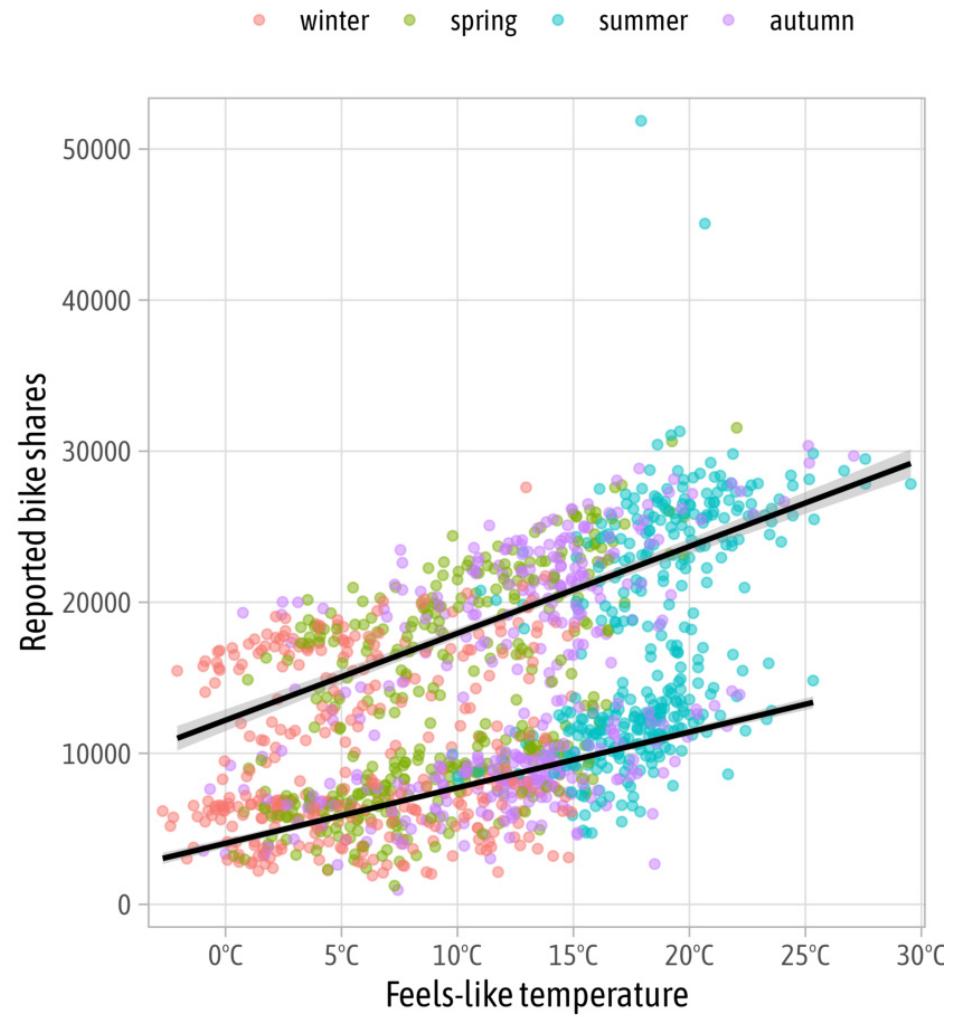
TfL bike sharing trends



Modify Scales

```
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 = .03, add = 0),
10    limits = c(0, NA)
11  ) +
12  scale_color_discrete()
```

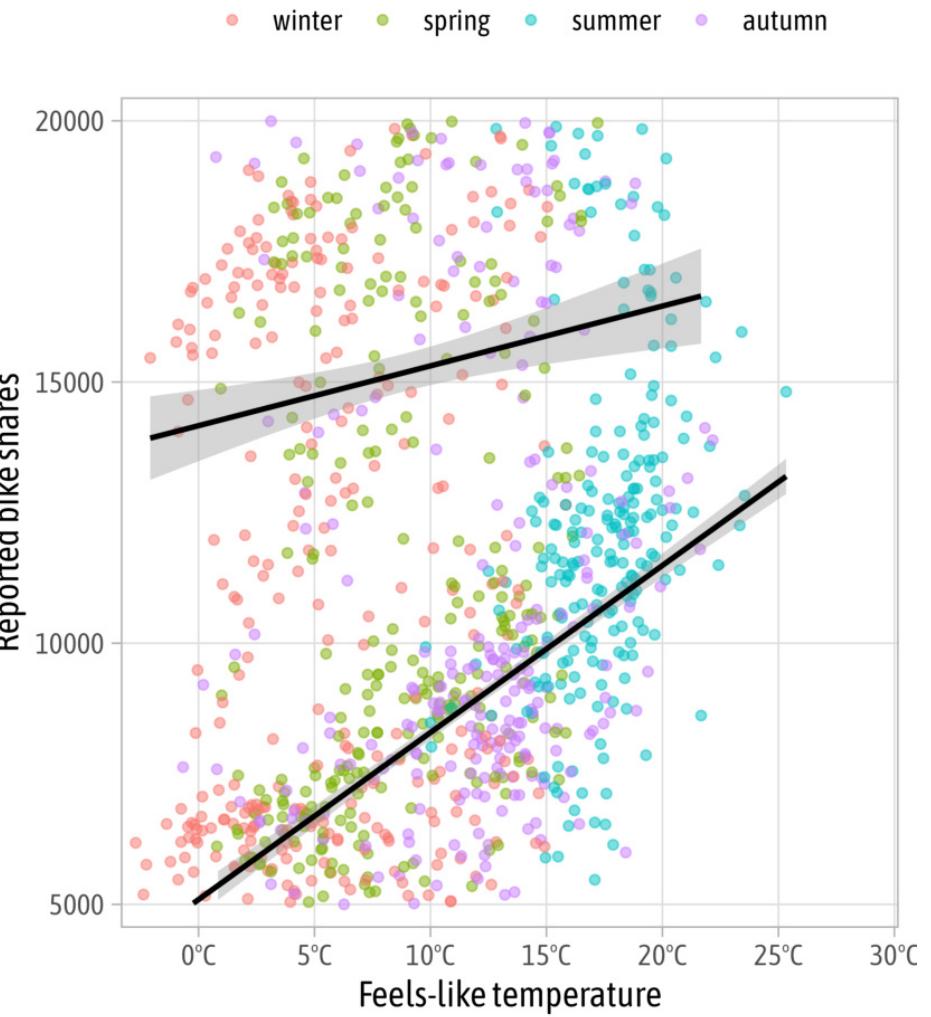
TfL bike sharing trends



Modify Scales

```
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 = .03, add = 0),
10    limits = c(5000, 20000)
11  ) +
12  scale_color_discrete()
```

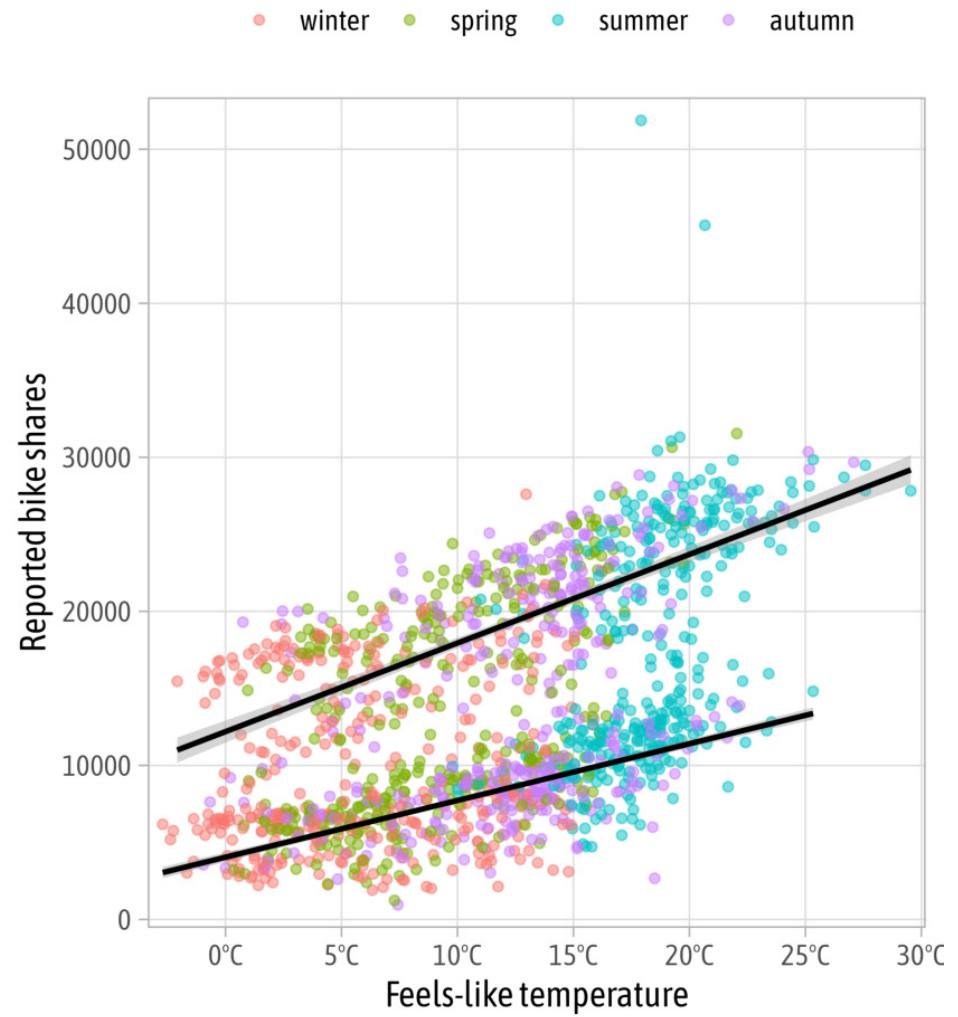
TfL bike sharing trends



Modify Scales

```
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  ) +
12  scale_color_discrete()
```

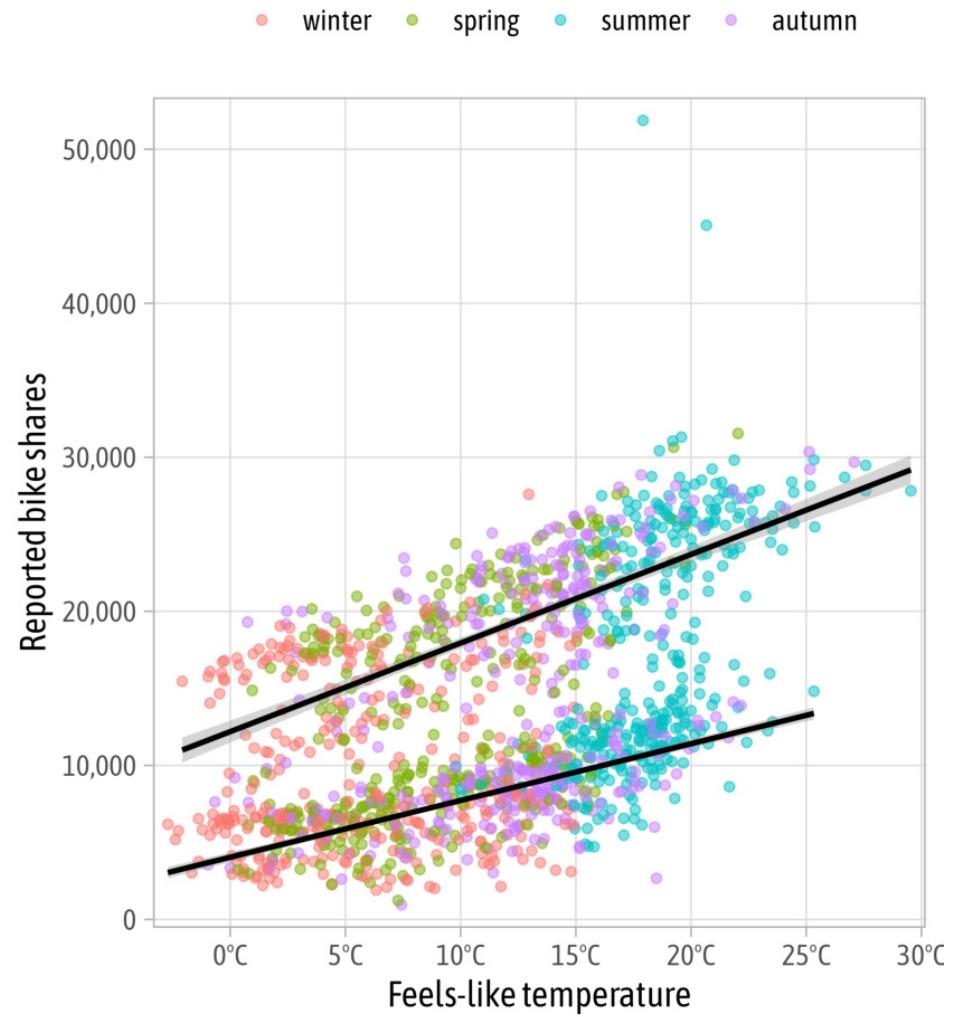
TfL bike sharing trends



Modify Scales

```
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  scale_color_discrete()
```

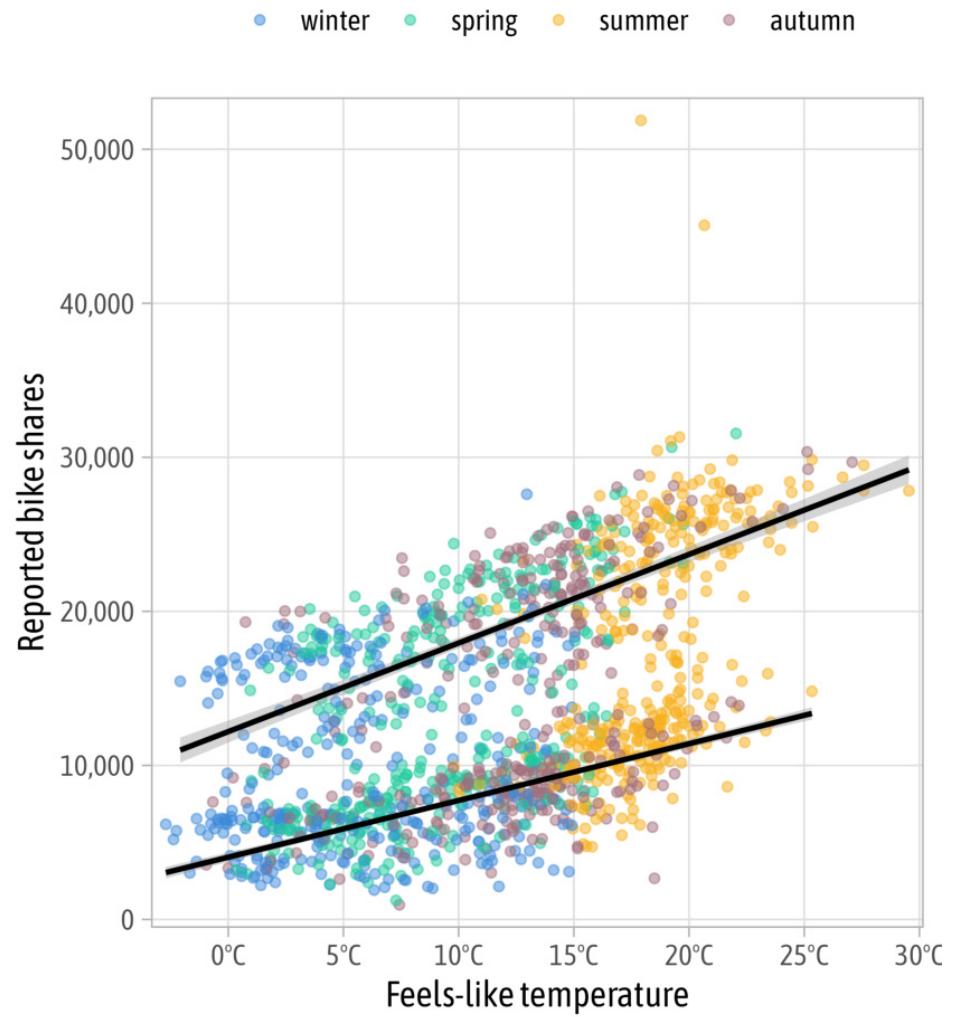
TfL bike sharing trends



Modify Scales

```
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  scale_color_discrete(
14    type = c("#3c89d9", "#1ec99b", "#f7b01b",
15  )
```

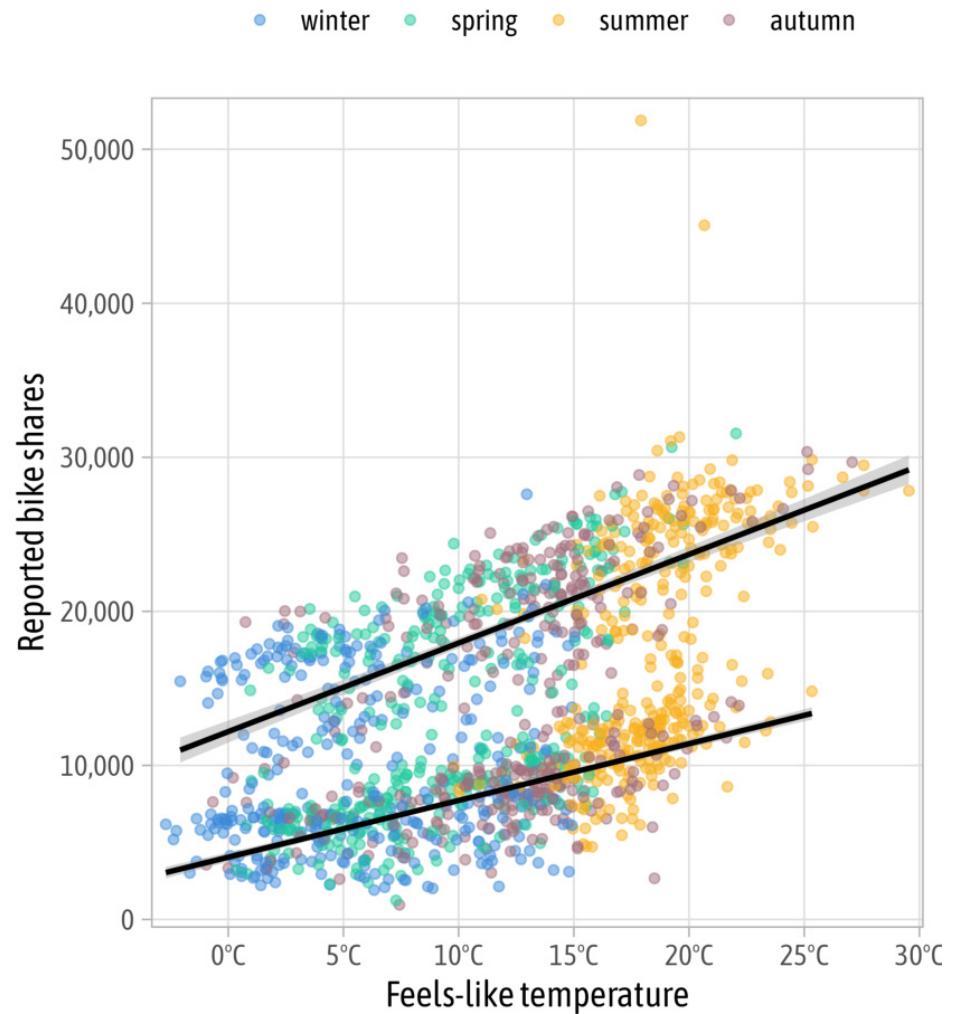
TfL bike sharing trends



Modify Scales

```
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  scale_color_manual(
14    values = c("#3c89d9", "#1ec99b", "#f7b01b")
15 )
```

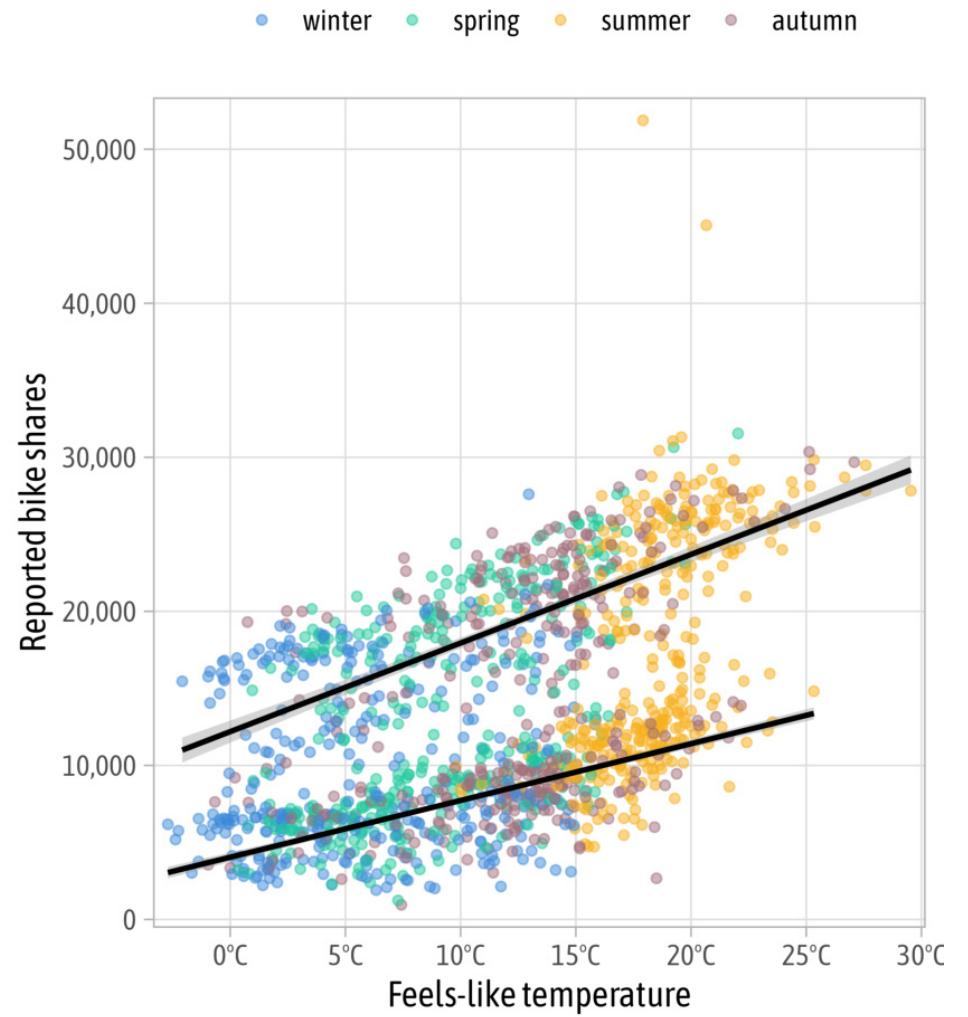
TfL bike sharing trends



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 = 0.02, 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     expand = c(mult = 0, add = 1500),
17     breaks = 0:5*10000,
18     labels = scales::label_comma()
19 ) +
20   scale_color_manual(
21     values = colors_sorted
22 )
```

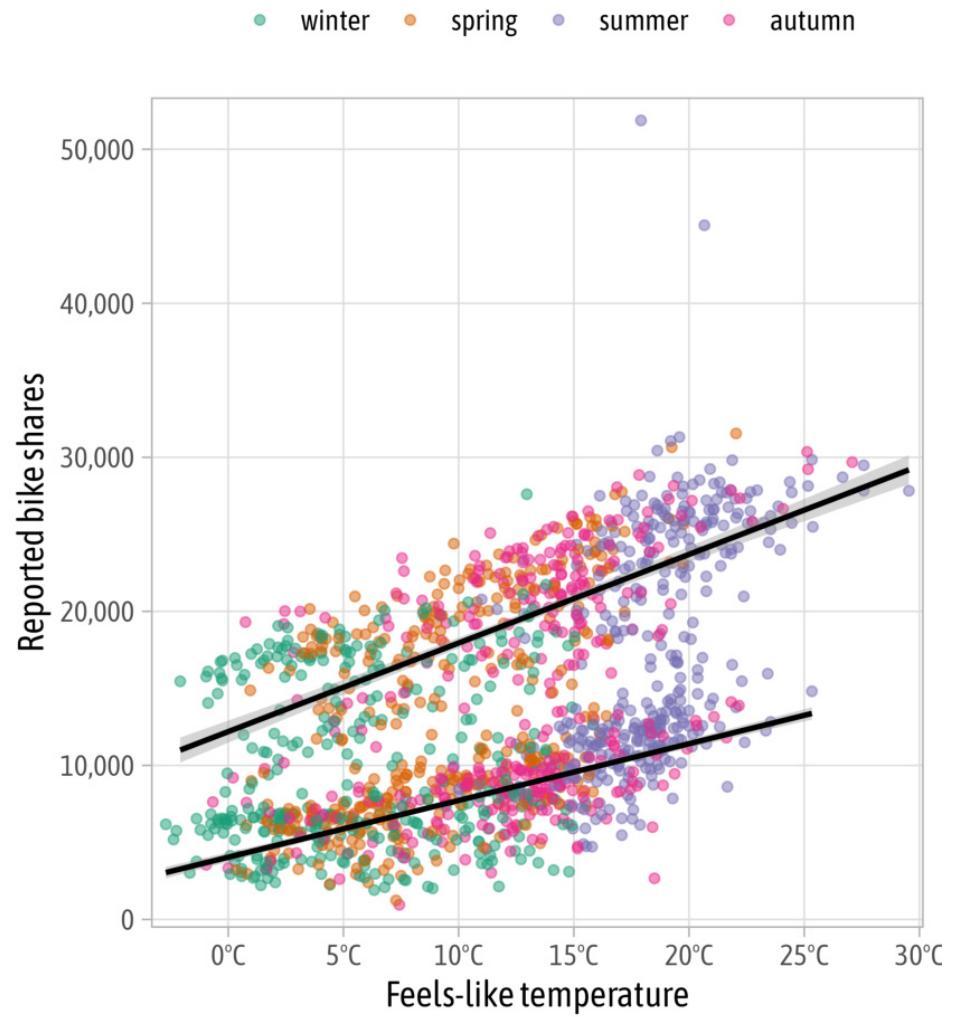
TfL bike sharing trends



Modify Scales

```
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  scale_color_brewer(
14    palette = "Dark2"
15  )
```

TfL bike sharing trends



{RColorBrewer}

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



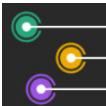
{RColorBrewer}

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



{scico}

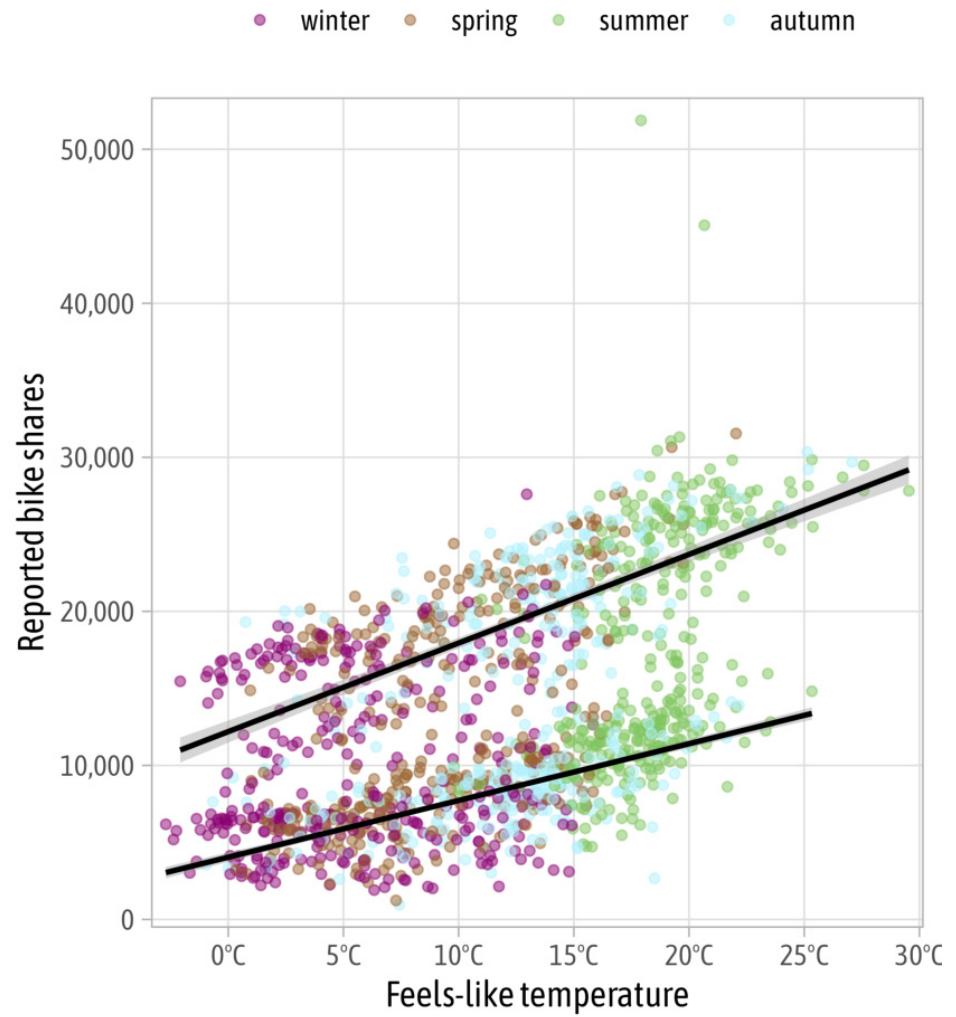
```
1 # install.packages("scico")
2 scico::scico_palette_show()
```



{scico}

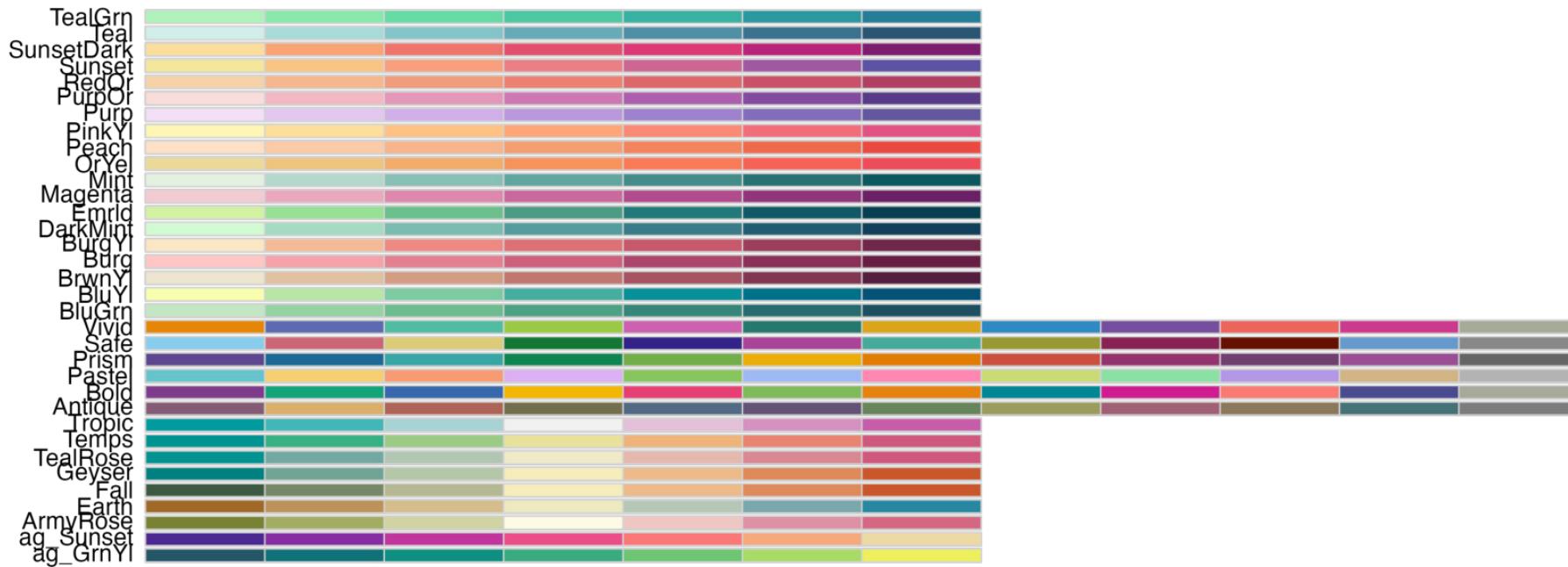
```
1 g +
  scale_x_continuous(
    expand = c(mult = 0.02, add = 0),
    breaks = seq(0, 30, by = 5),
    labels = function(x) paste0(x, "°C"),
    name = "Feels-like temperature"
  ) +
  scale_y_continuous(
    expand = c(mult = 0, add = 1500),
    breaks = 0:5*10000,
    labels = scales::label_comma()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

TfL bike sharing trends



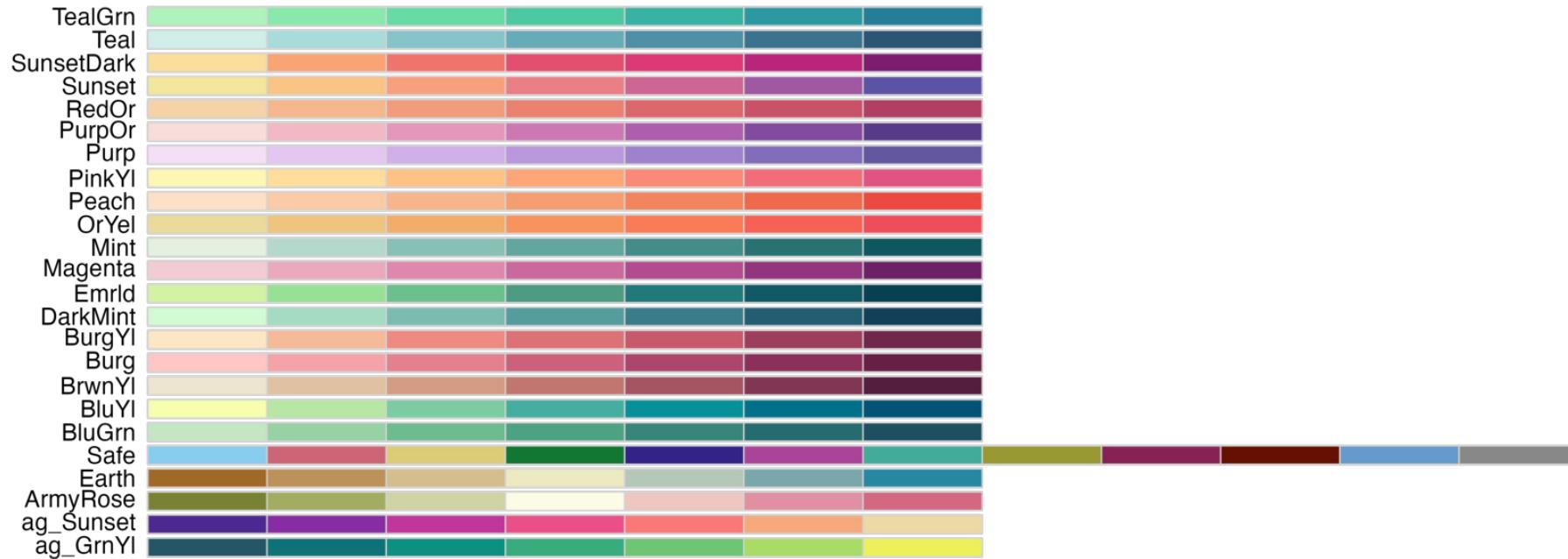
{rcartocolor}

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



{rcartocolor}

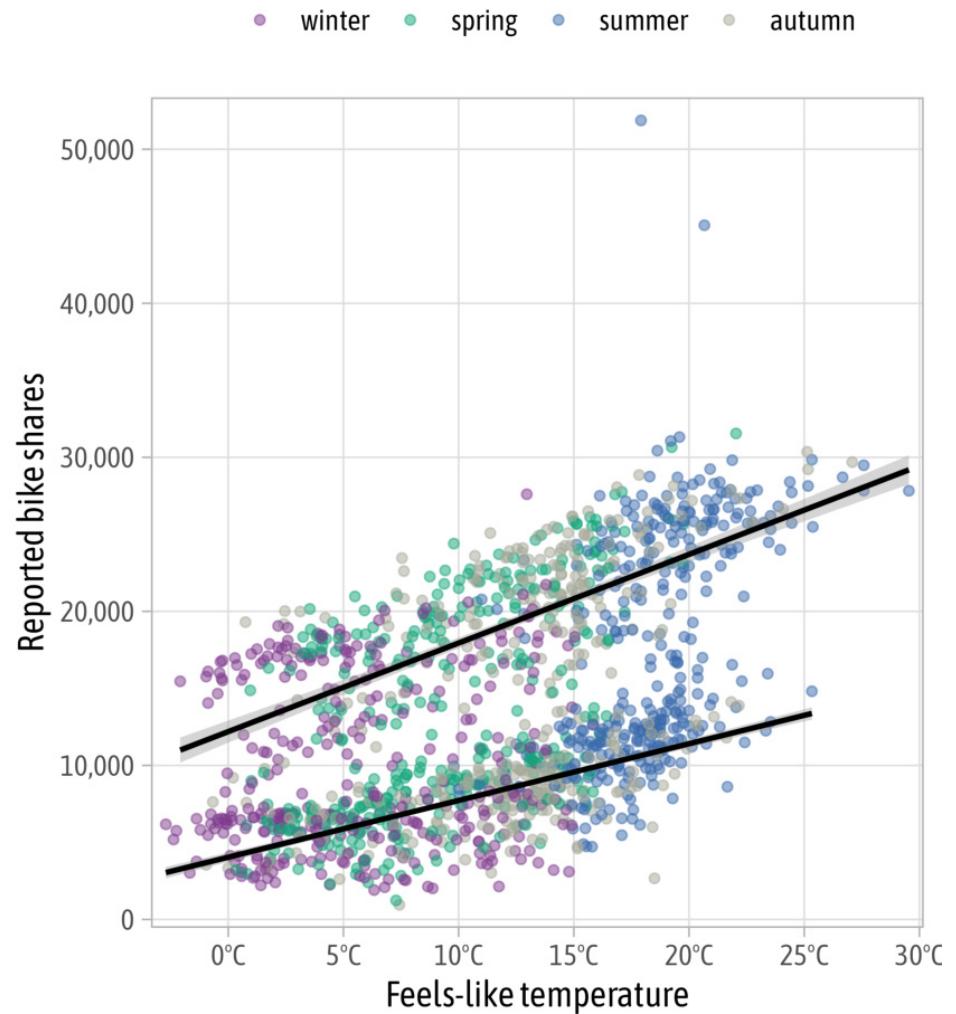
```
1 # install.packages("rcartocolor")
2 rcartocolor::display_carto_all(colorblind_friendly = TRUE)
```



{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  )
```

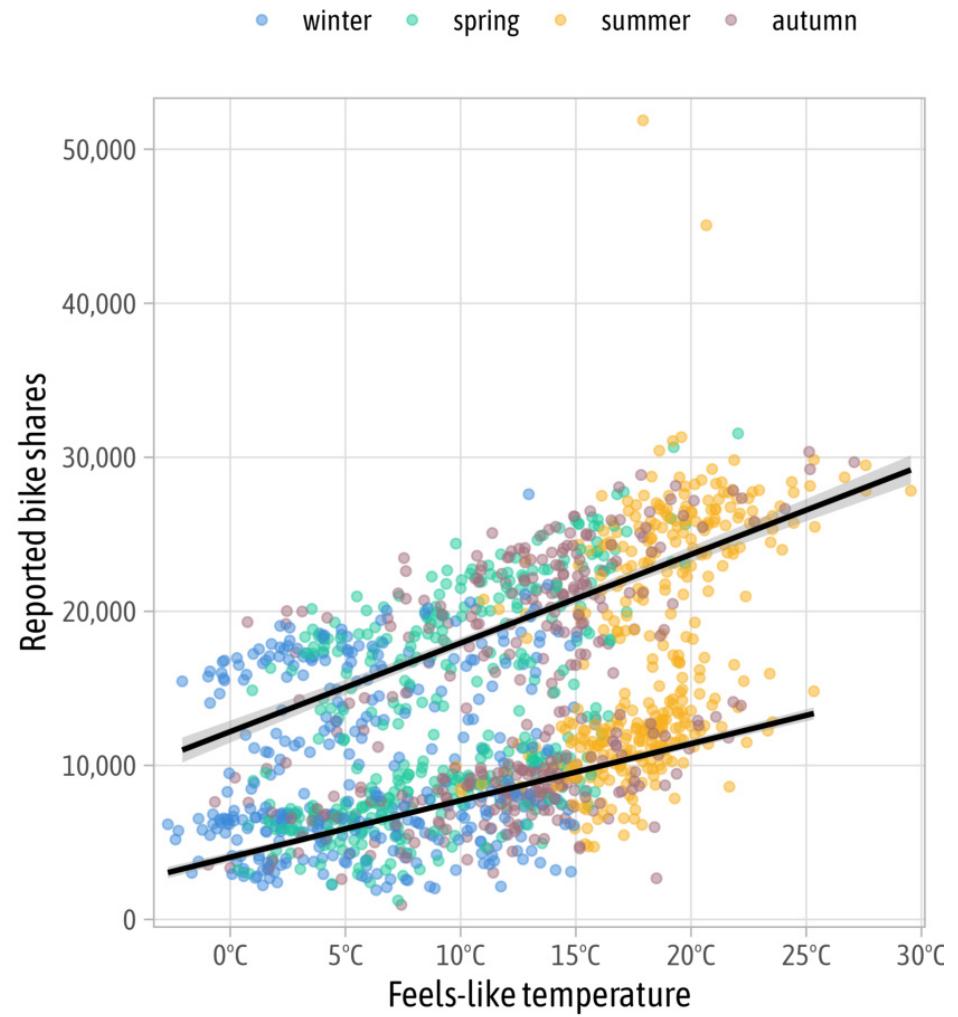
TfL bike sharing trends



Modify Scales

```
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  scale_color_manual(
14    values = colors_sorted,
15    name = NULL
16  )
```

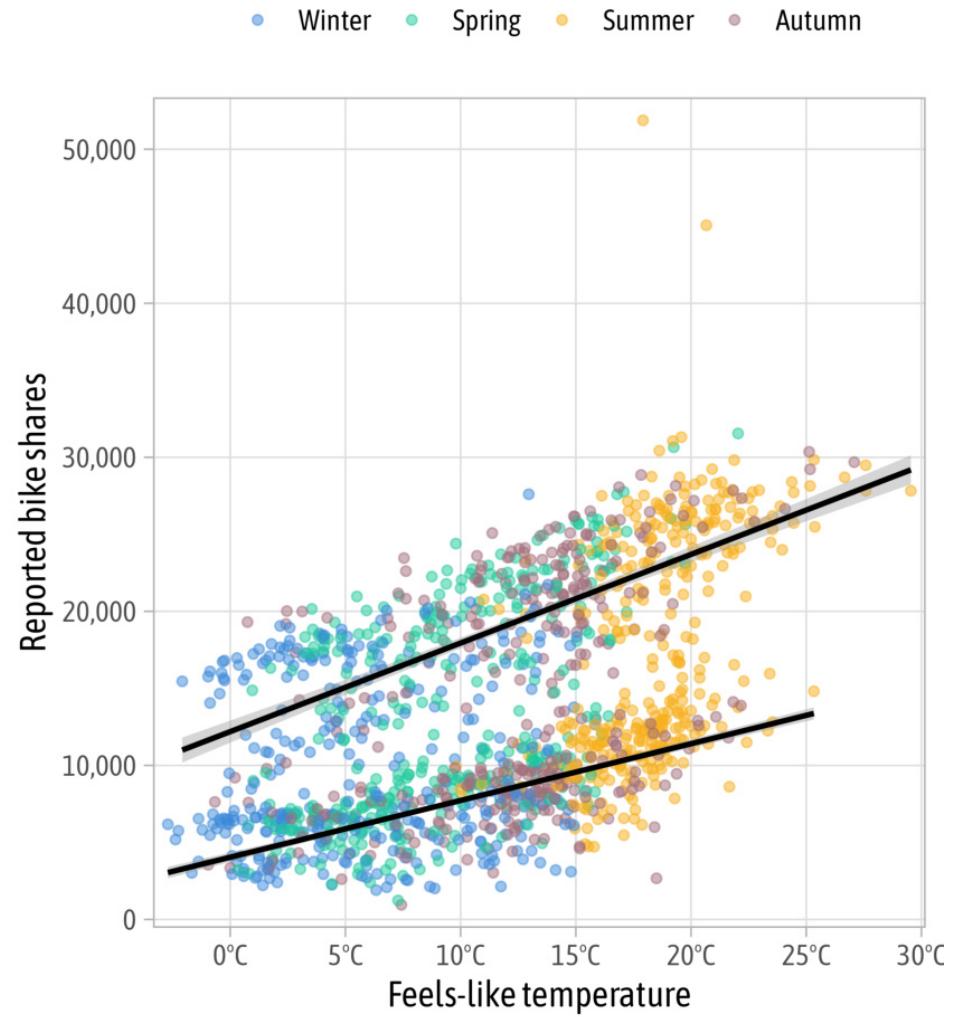
TfL bike sharing trends



Modify Scales

```
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  scale_color_manual(
14    values = colors_sorted,
15    name = NULL,
16    labels = stringr::str_to_title
17  )
```

TfL bike sharing trends



Modify Scales

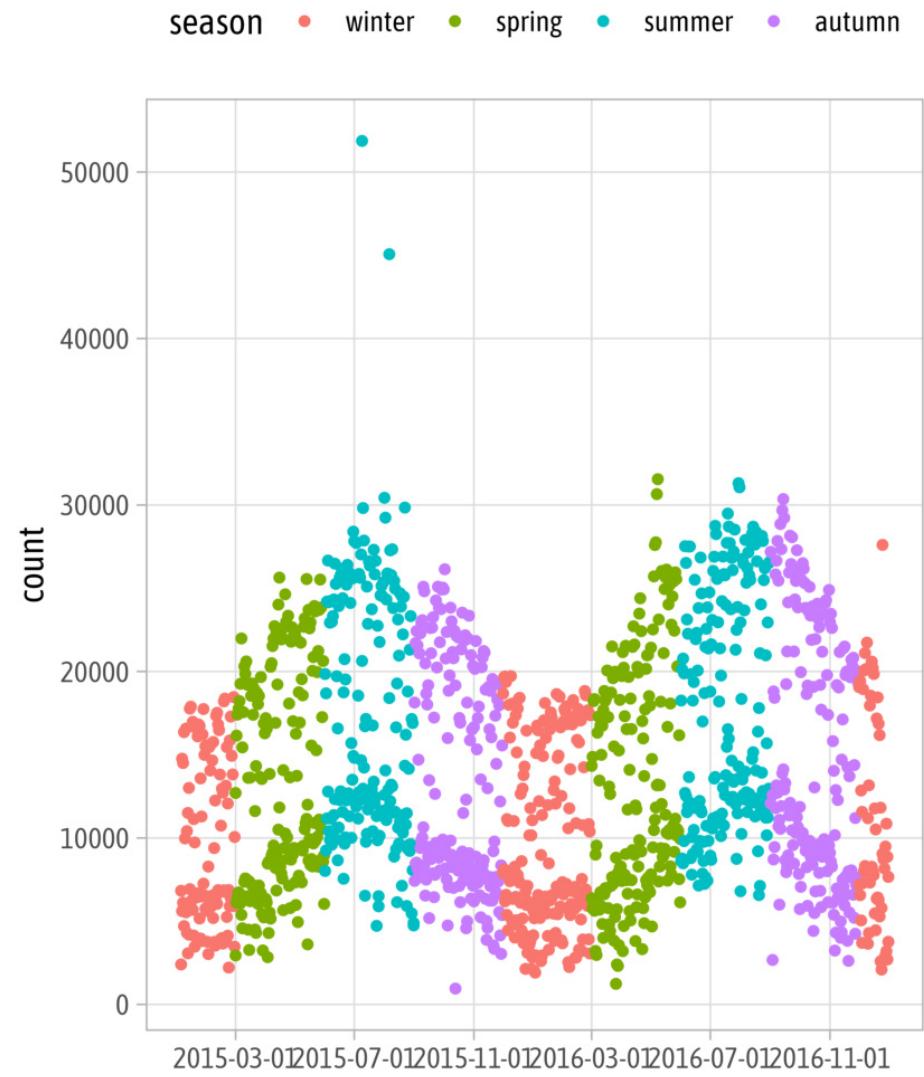
```
1 g +
2   scale_x_continuous(
3     expand = c(mult = 0.02, add = 0),
4     breaks = seq(0, 30, by = 5),
5     labels = function(x) paste0(x, "°C"),
6     name = "Feels-like temperature"
7   ) +
8   scale_y_continuous(
9     expand = c(mult = 0, add = 1500),
10    breaks = 0:5*10000,
11    labels = scales::label_comma()
12  ) +
13  scale_color_manual(
14    values = colors_sorted,
15    name = NULL,
16    labels = stringr::str_to_title,
17    guide = guide_legend(
18      override.aes = list(size = 5)
19    )
20  )
```

TfL bike sharing trends



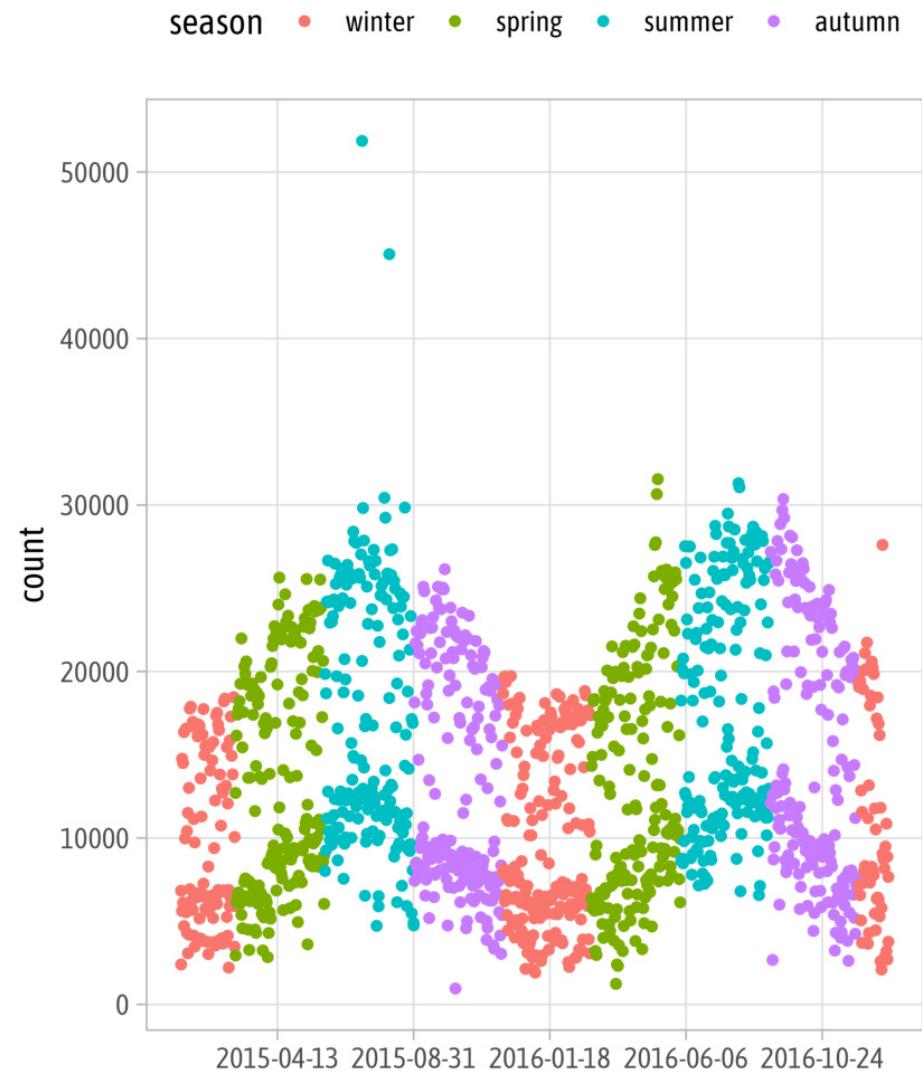
Date Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point() +  
7 scale_x_date(  
8   name = NULL,  
9   date_breaks = "4 months"  
10 )
```



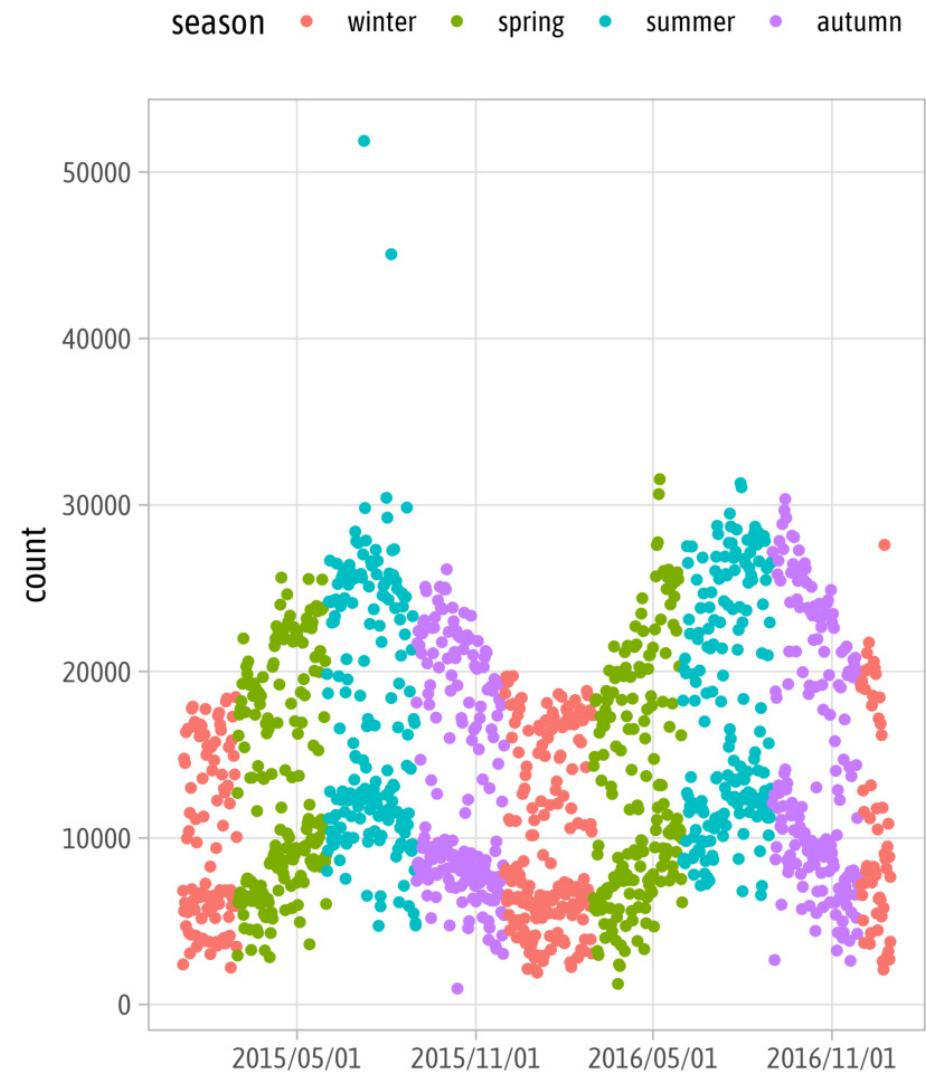
Date Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point() +  
7 scale_x_date(  
8   name = NULL,  
9   date_breaks = "20 weeks"  
10 )
```



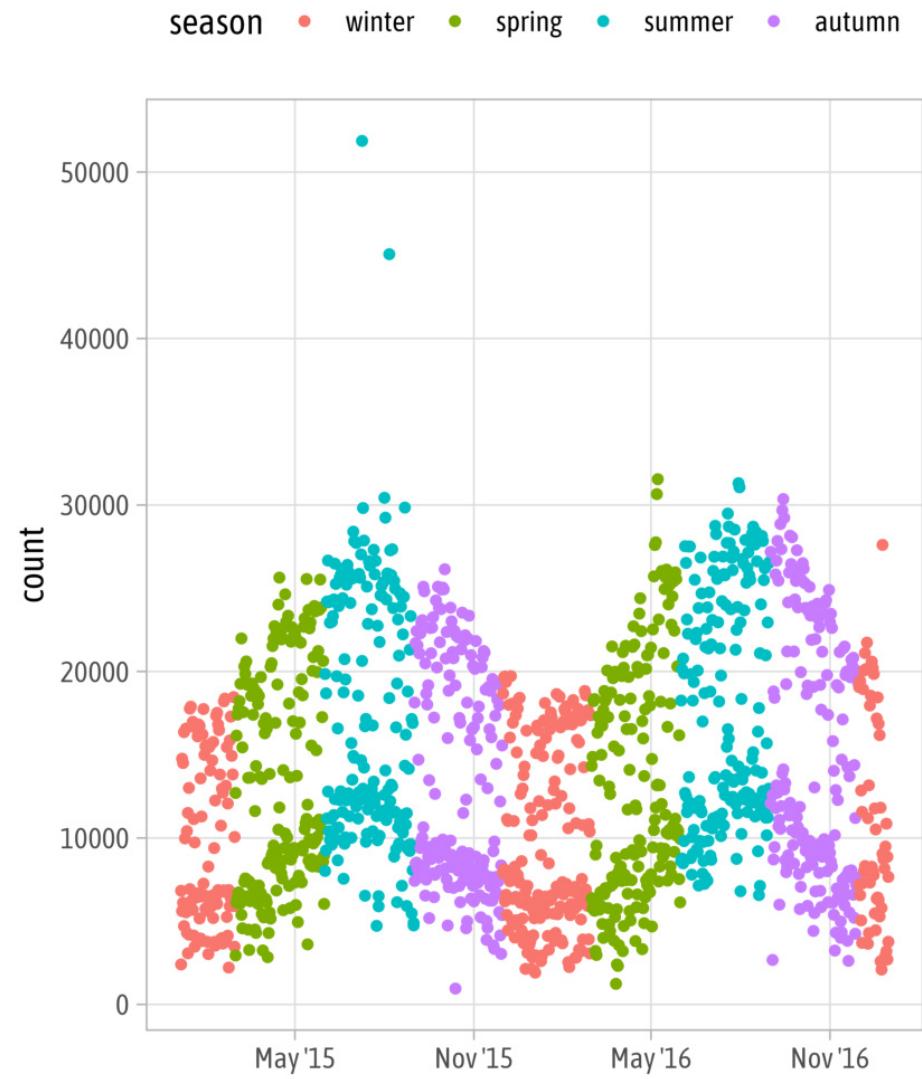
Date Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point() +  
7 scale_x_date(  
8   name = NULL,  
9   date_breaks = "6 months",  
10  date_labels = "%Y/%m/%d"  
11 )
```



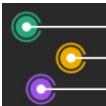
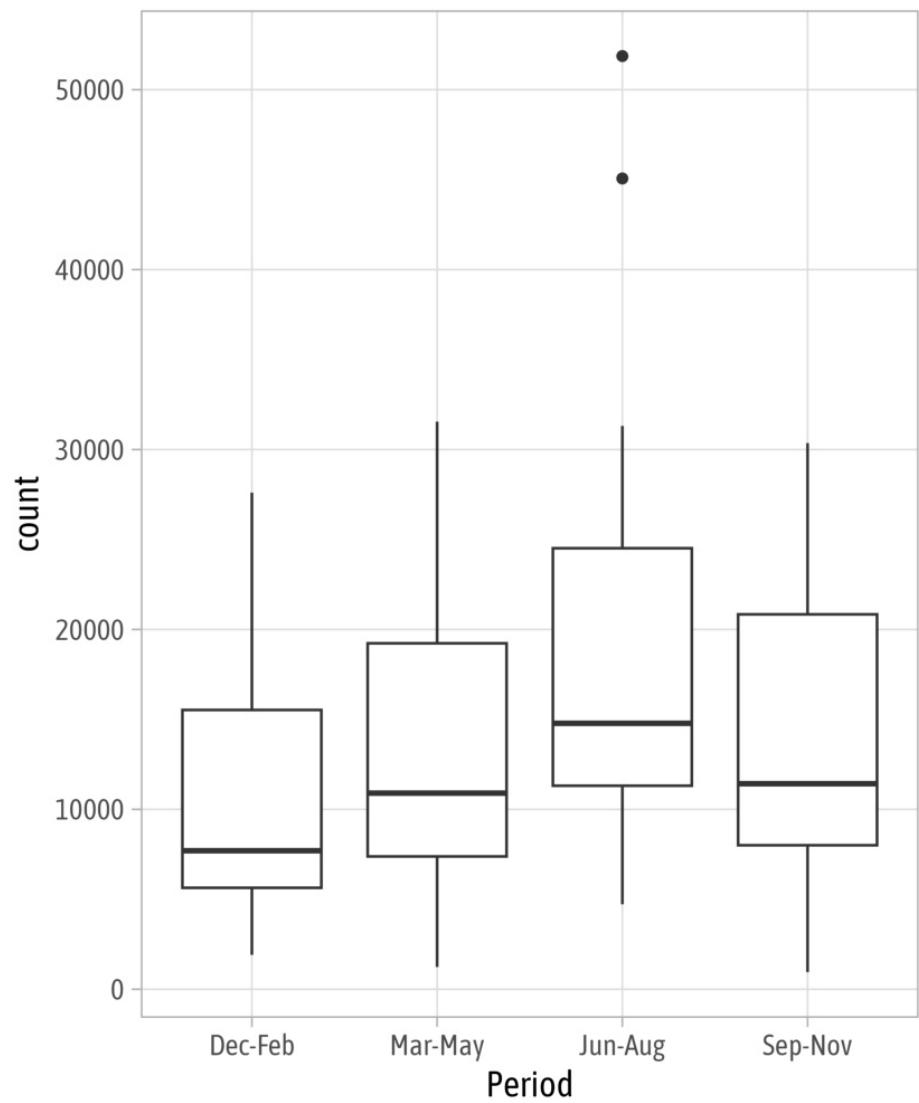
Date Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point() +  
7 scale_x_date(  
8   name = NULL,  
9   date_breaks = "6 months",  
10  date_labels = "%b '%y"  
11 )
```



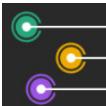
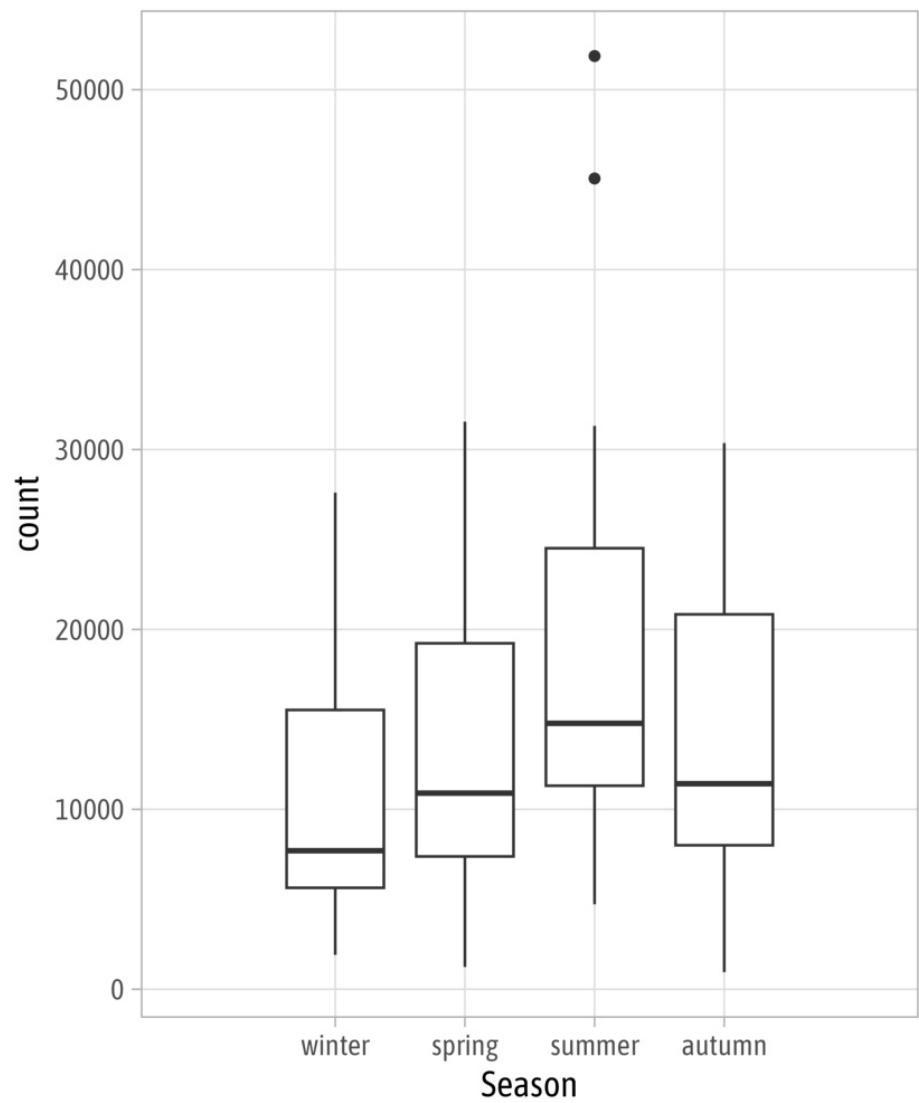
Discrete Positional Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 scale_x_discrete(  
7   name = "Period",  
8   labels = c("Dec-Feb", "Mar-May", "Jun-Aug"  
9 )
```



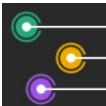
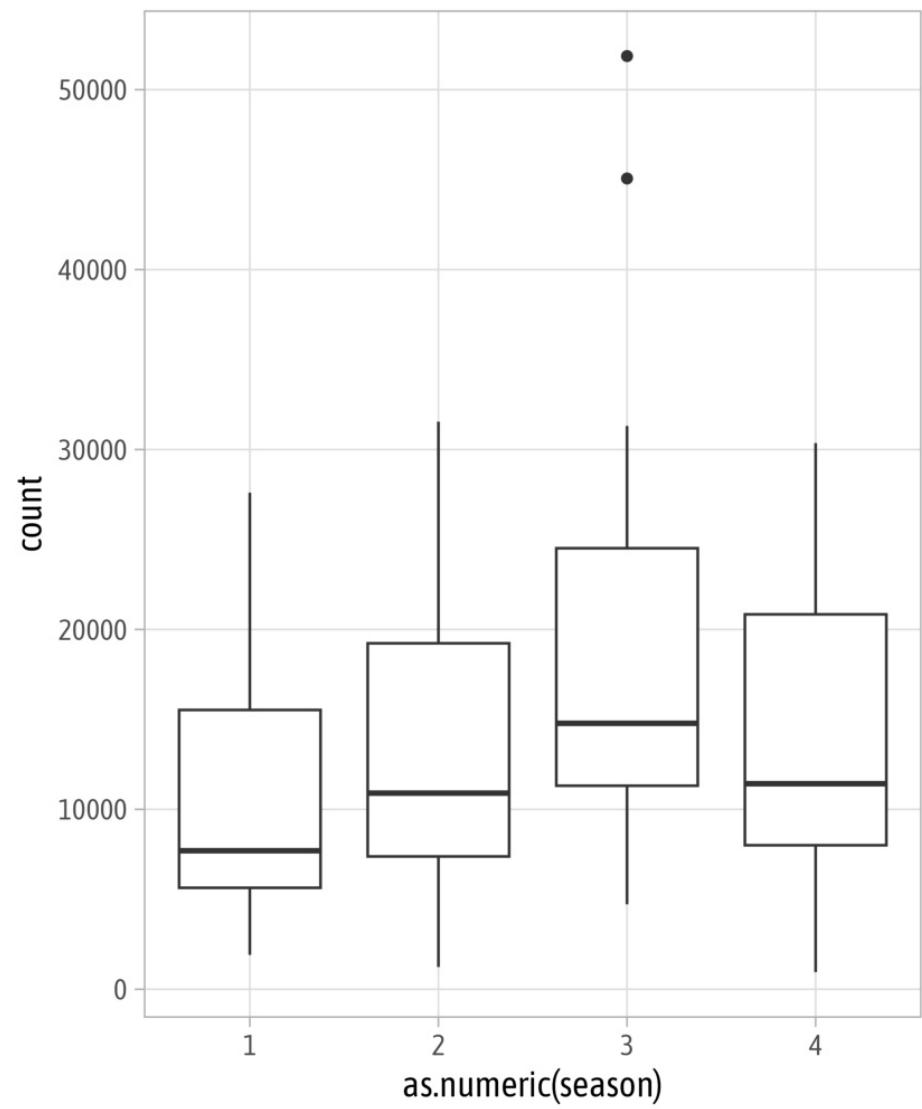
Discrete Positional Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 scale_x_discrete(  
7   name = "Season",  
8   expand = c(.5, 0) ## add, mult  
9 )
```



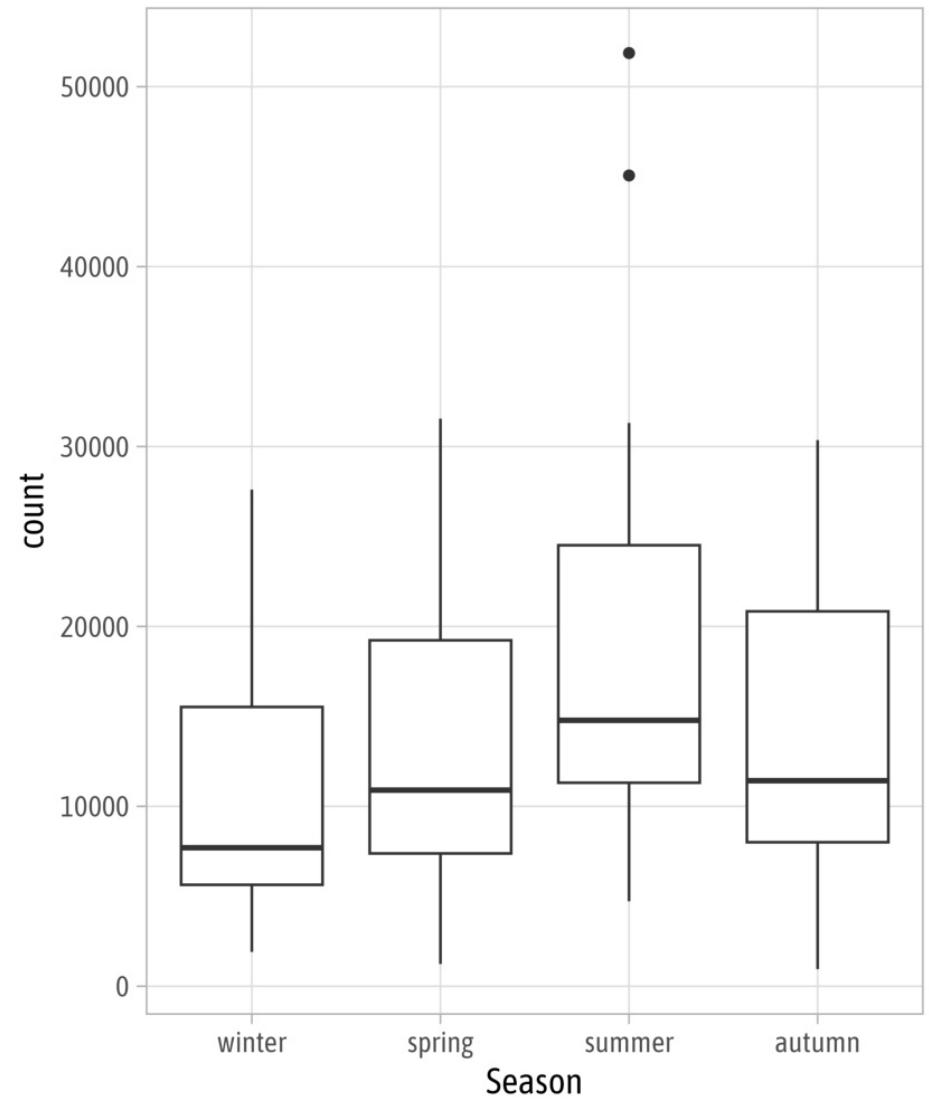
Discrete or Continuous Scales?

```
1 ggplot(  
2   bikes,  
3   aes(x = as.numeric(season), y = count)  
4 ) +  
5   geom_boxplot(  
6   aes(group = season)  
7 )
```



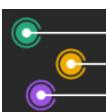
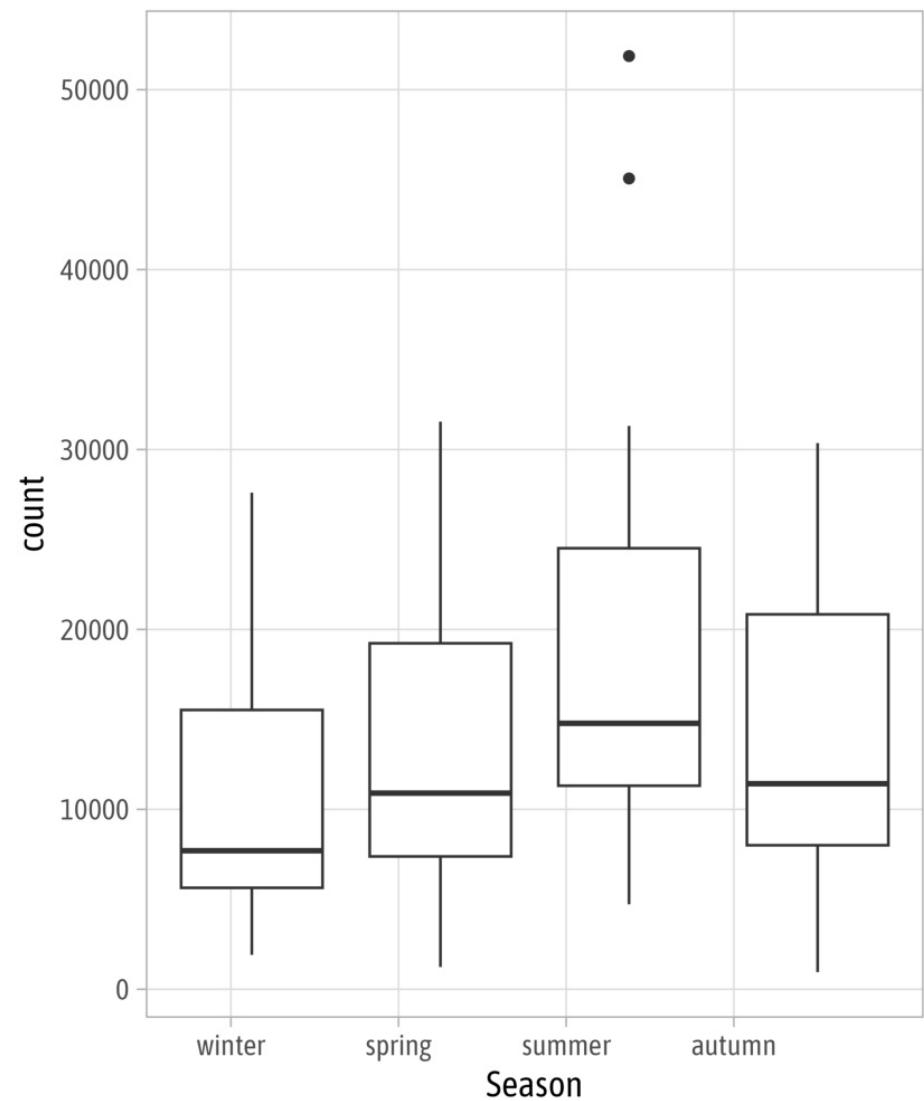
Discrete or Continuous Scales?

```
1 ggplot(  
2   bikes,  
3   aes(x = as.numeric(season),  
4       y = count)  
5 ) +  
6   geom_boxplot(  
7   aes(group = season)  
8 ) +  
9   scale_x_continuous(  
10  name = "Season",  
11  breaks = 1:4,  
12  labels = levels(bikes$season)  
13 )
```



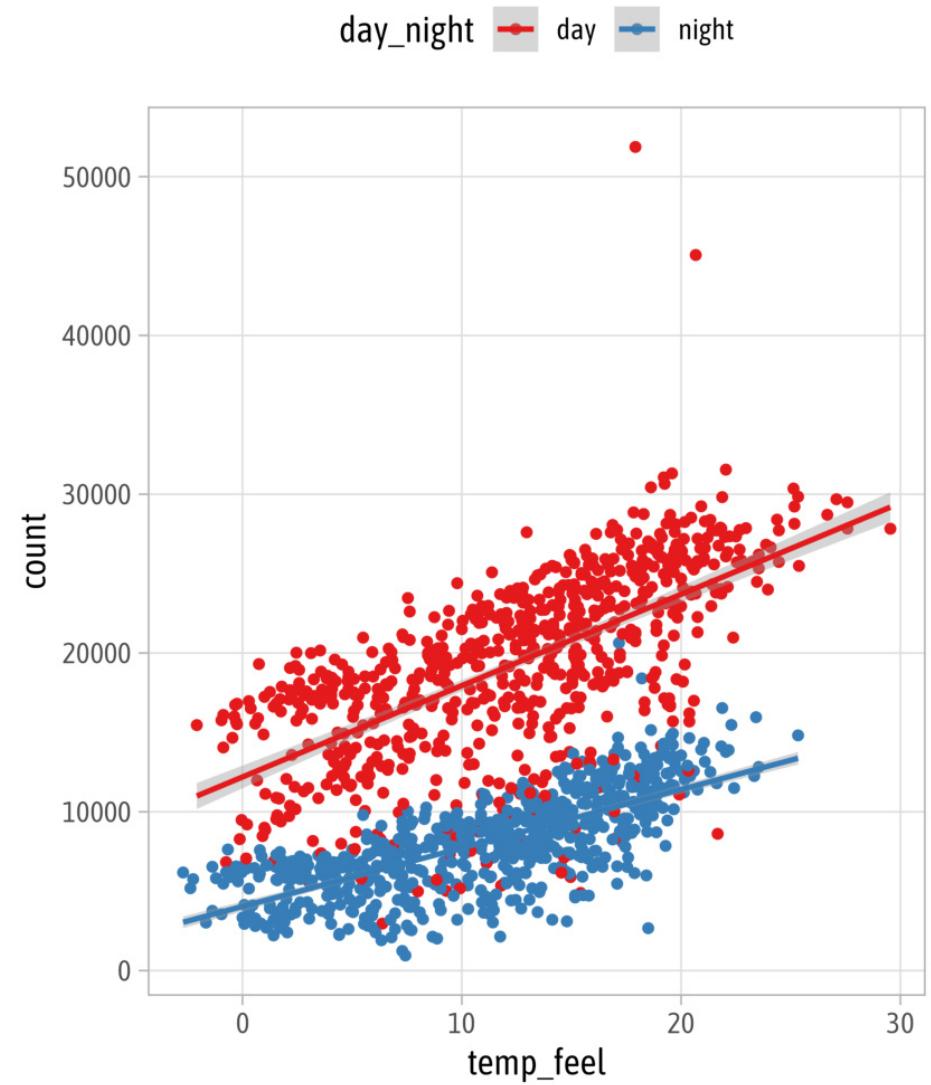
Discrete or Continuous Scales?

```
1 ggplot(  
2   bikes,  
3   aes(x = as.numeric(season) +  
4         as.numeric(season) / 8,  
5         y = count)  
6 ) +  
7 geom_boxplot(  
8   aes(group = season)  
9 ) +  
10 scale_x_continuous(  
11   name = "Season",  
12   breaks = 1:4,  
13   labels = levels(bikes$season)  
14 )
```



after_scale()

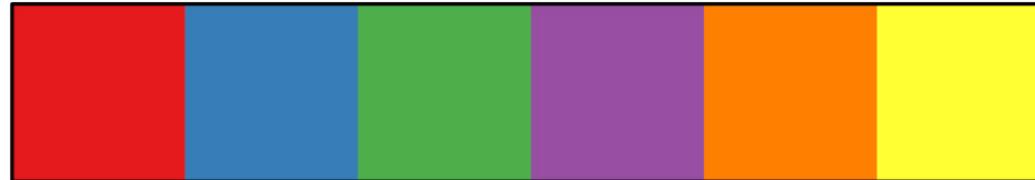
```
1 ggplot(bikes, aes(temp_feel, y = count)) +  
2   geom_point(  
3     aes(color = day_night)  
4   ) +  
5   stat_smooth(  
6     aes(color = day_night),  
7     method = "lm"  
8   ) +  
9   scale_color_brewer(palette = "Set1")
```



{prismatic}

```
1 # install.packages("prismatic")
2 colors <- RColorBrewer::brewer.pal(n = 6, name = "Set1")
3 colors
```

```
[1] "#E41A1C" "#377EB8" "#4DAF4A" "#984EA3" "#FF7F00" "#FFFF33"
```



{prismatic}

```
1 # install.packages("prismatic")
2 library(prismatic)
3 clr_lighten(colors, .5)
```



{prismatic}

```
1 # install.packages("prismatic")
2 library(prismatic)
3 clr_desaturate(colors, .3)
```



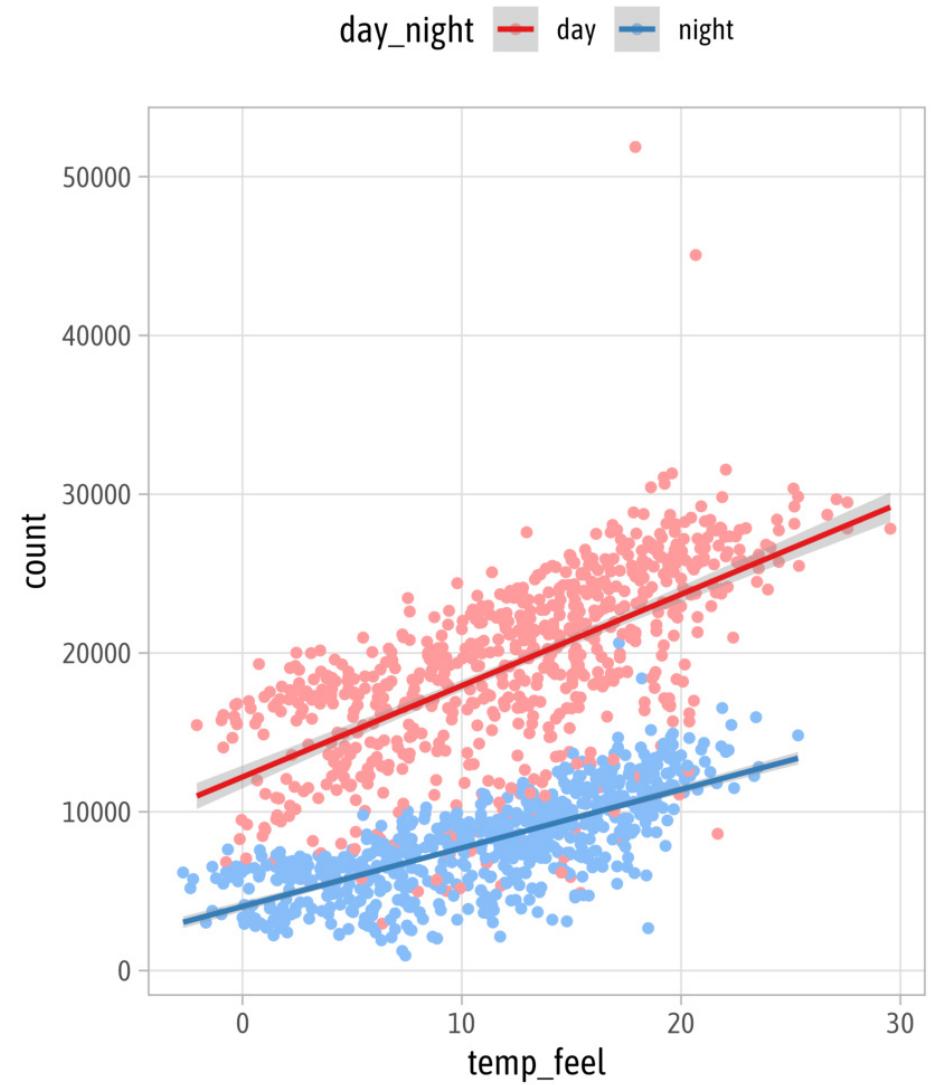
{prismatic}

```
1 # install.packages("prismatic")
2 library(prismatic)
3 clr_desaturate(clr_lighten(colors, .5), .3)
```



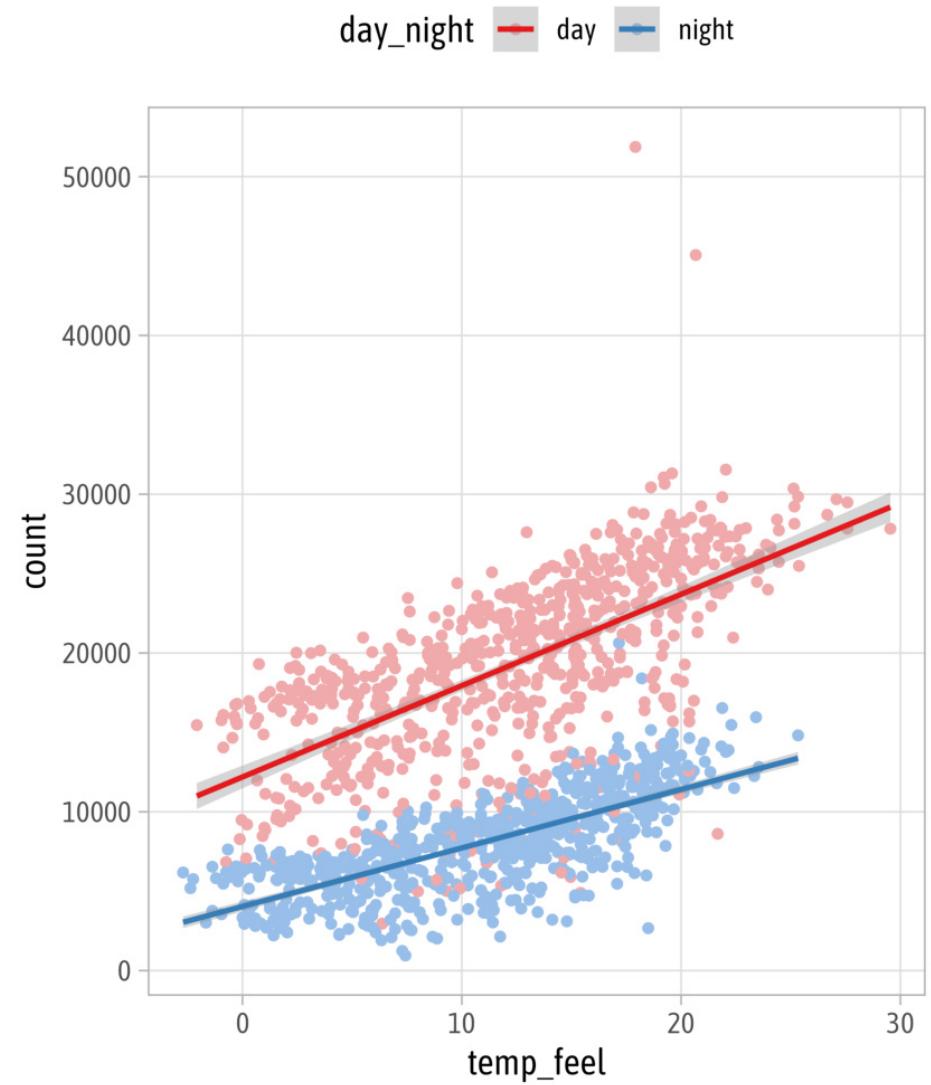
after_scale()

```
1 ggplot(bikes, aes(temp_feel, y = count)) +  
2   geom_point(  
3     aes(  
4       color = day_night,  
5       color = after_scale(  
6         clr_lighten(color, .5)  
7     )  
8   )  
9 ) +  
10 stat_smooth(  
11   aes(color = day_night),  
12   method = "lm"  
13 ) +  
14 scale_color_brewer(palette = "Set1")
```



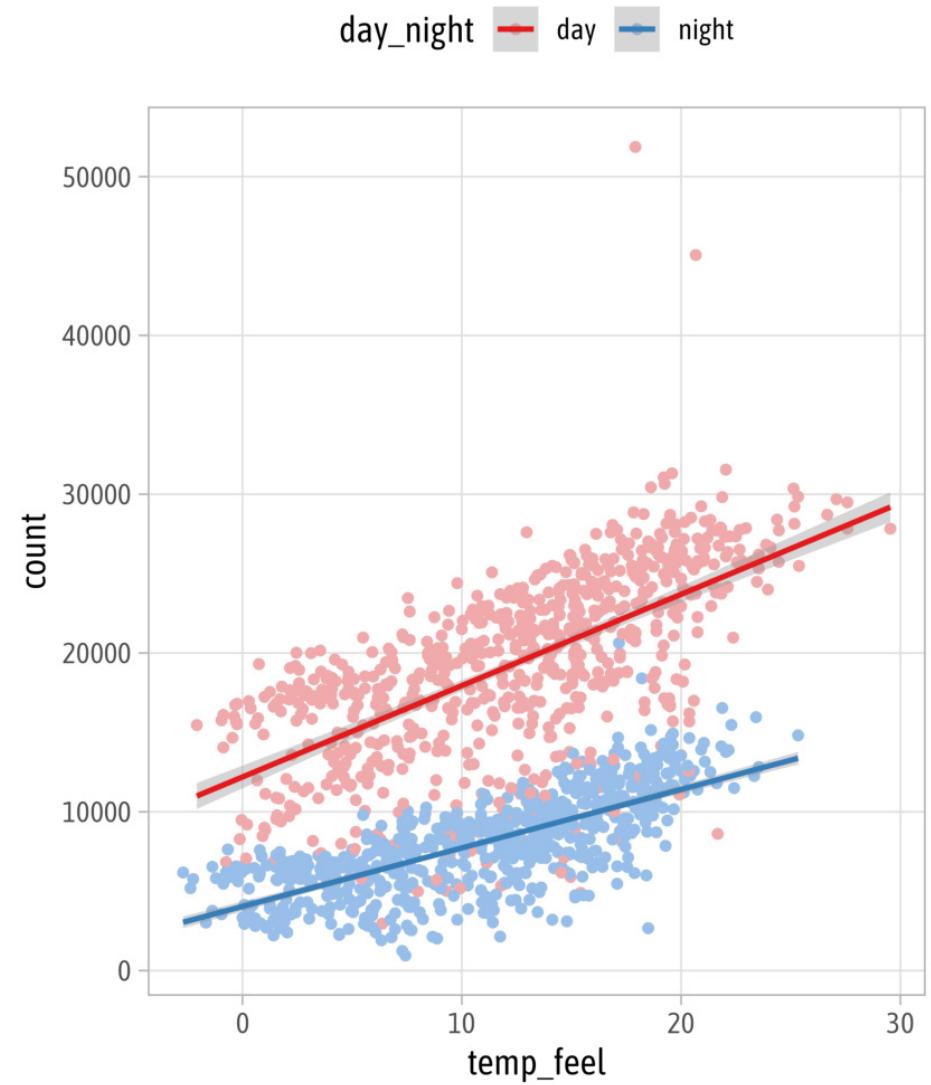
after_scale()

```
1 ggplot(bikes, aes(temp_feel, y = count)) +  
2   geom_point(  
3     aes(  
4       color = day_night,  
5       color = after_scale(  
6         clr_desaturate(  
7           clr_lighten(color, .5), .3  
8       )  
9     ))  
10    )  
11  ) +  
12  stat_smooth(  
13    aes(color = day_night),  
14    method = "lm"  
15  ) +  
16  scale_color_brewer(palette = "Set1")
```



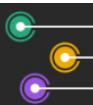
after_scale()

```
1 ggplot(bikes, aes(temp_feel, y = count)) +  
2   geom_point(  
3     aes(  
4       color = stage(  
5         day_night,  
6         after_scale = clr_desaturate(  
7           clr_lighten(color, .5), .3  
8       )  
9     ))  
10    )  
11  ) +  
12  stat_smooth(  
13    aes(color = day_night),  
14    method = "lm"  
15  ) +  
16  scale_color_brewer(palette = "Set1")
```



Data Visualization with {ggplot2}

— Coordinate Systems —



Coordinate Systems

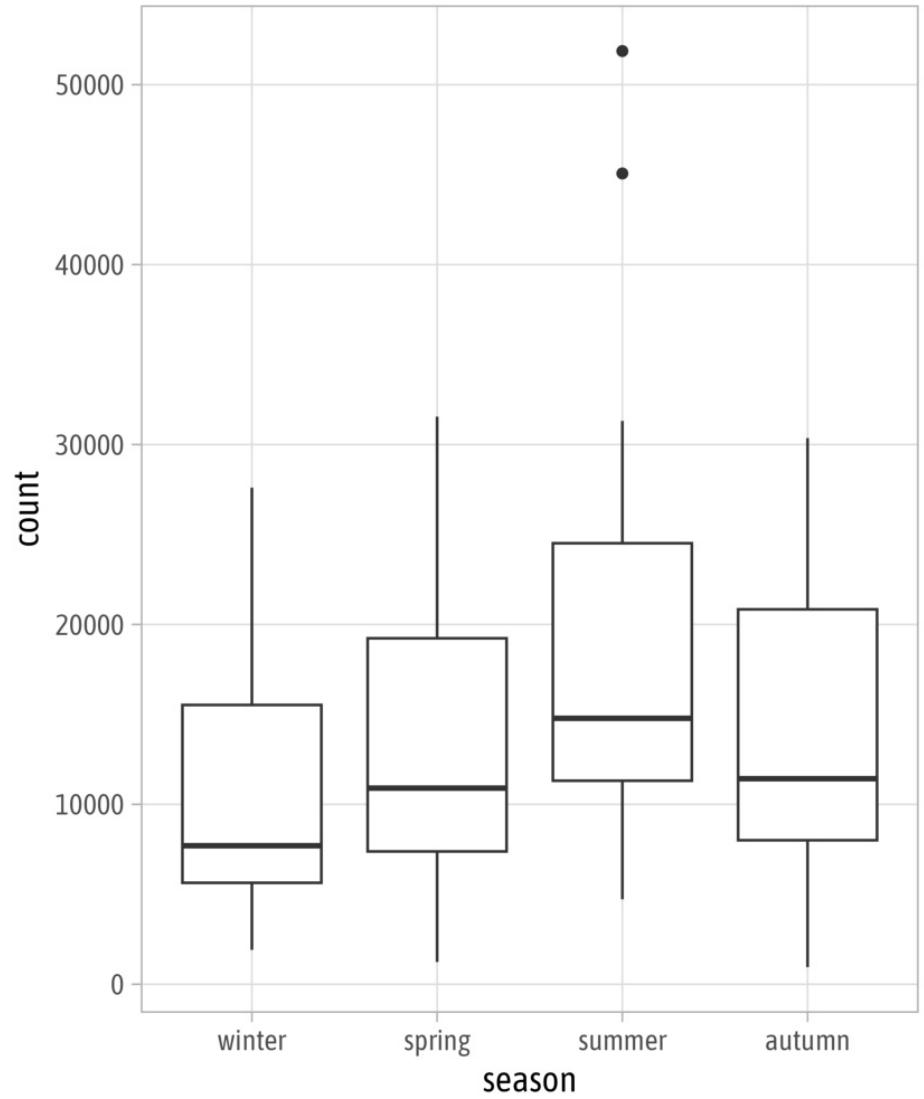
= interpret the position aesthetics

- **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_trans()`
 - `coord_map()` and `coord_sf()`



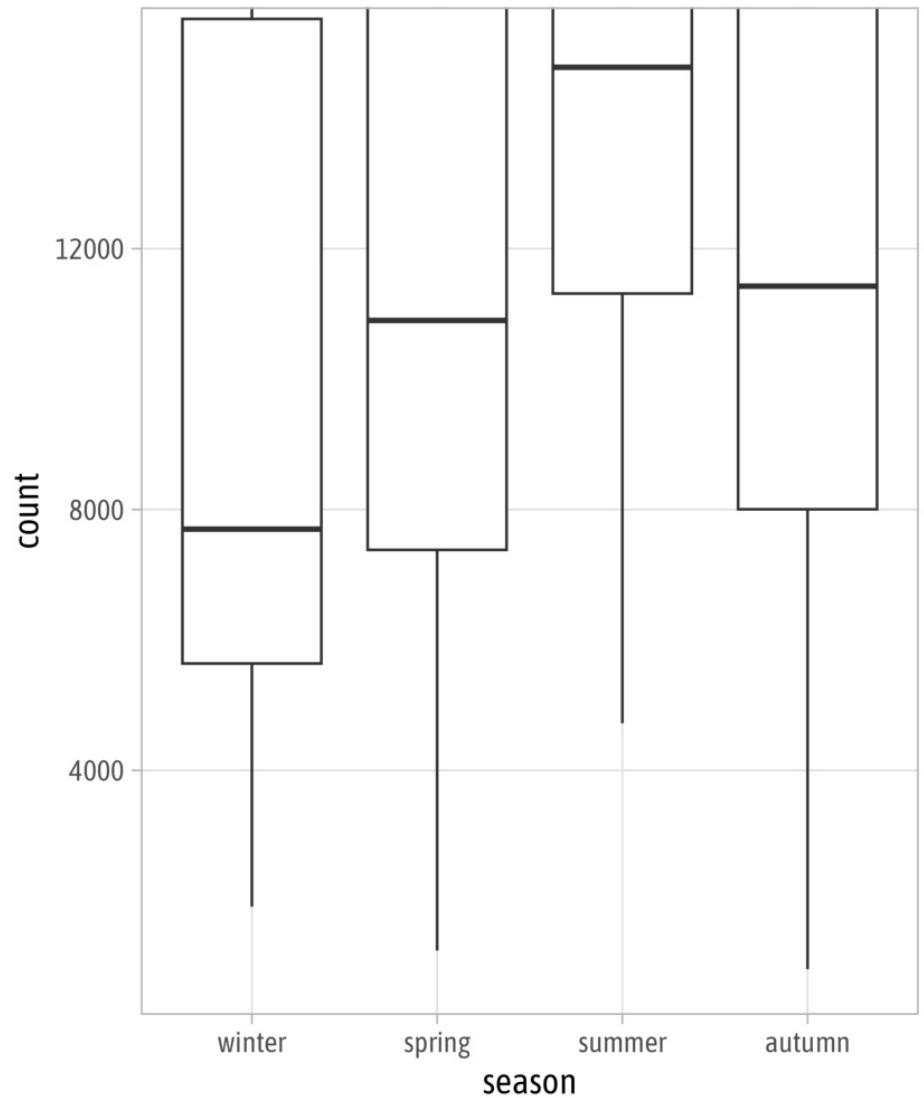
Cartesian Coordinate System

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



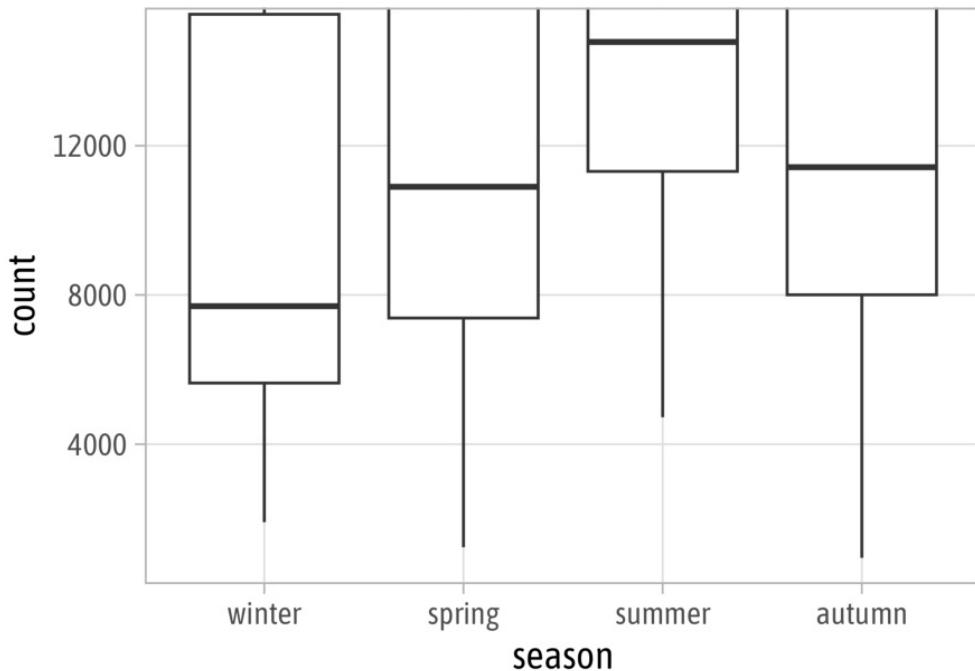
Cartesian Coordinate System

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

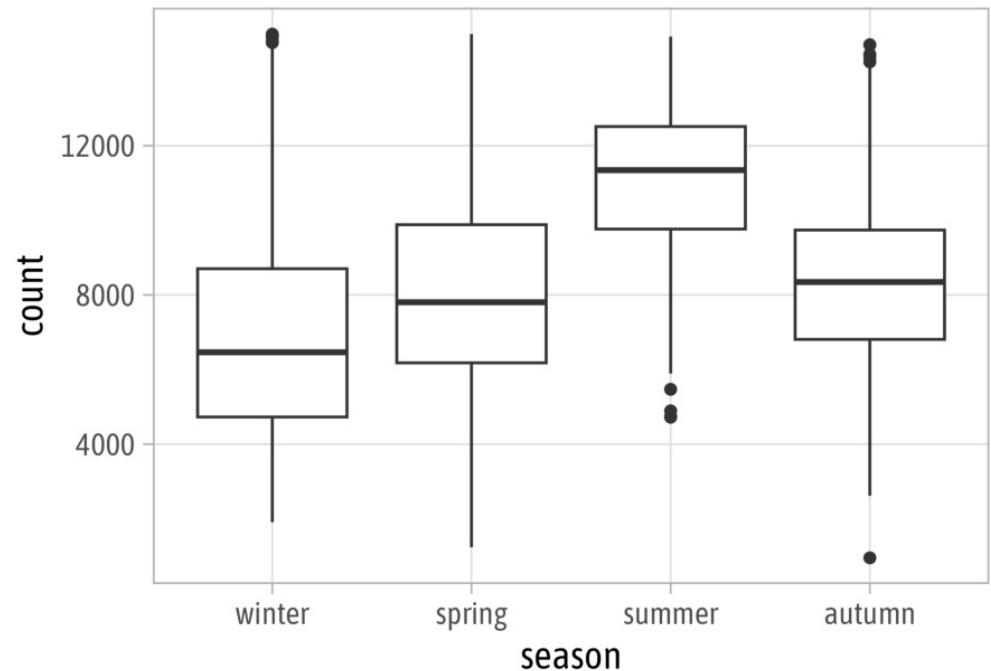


Changing Limits

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

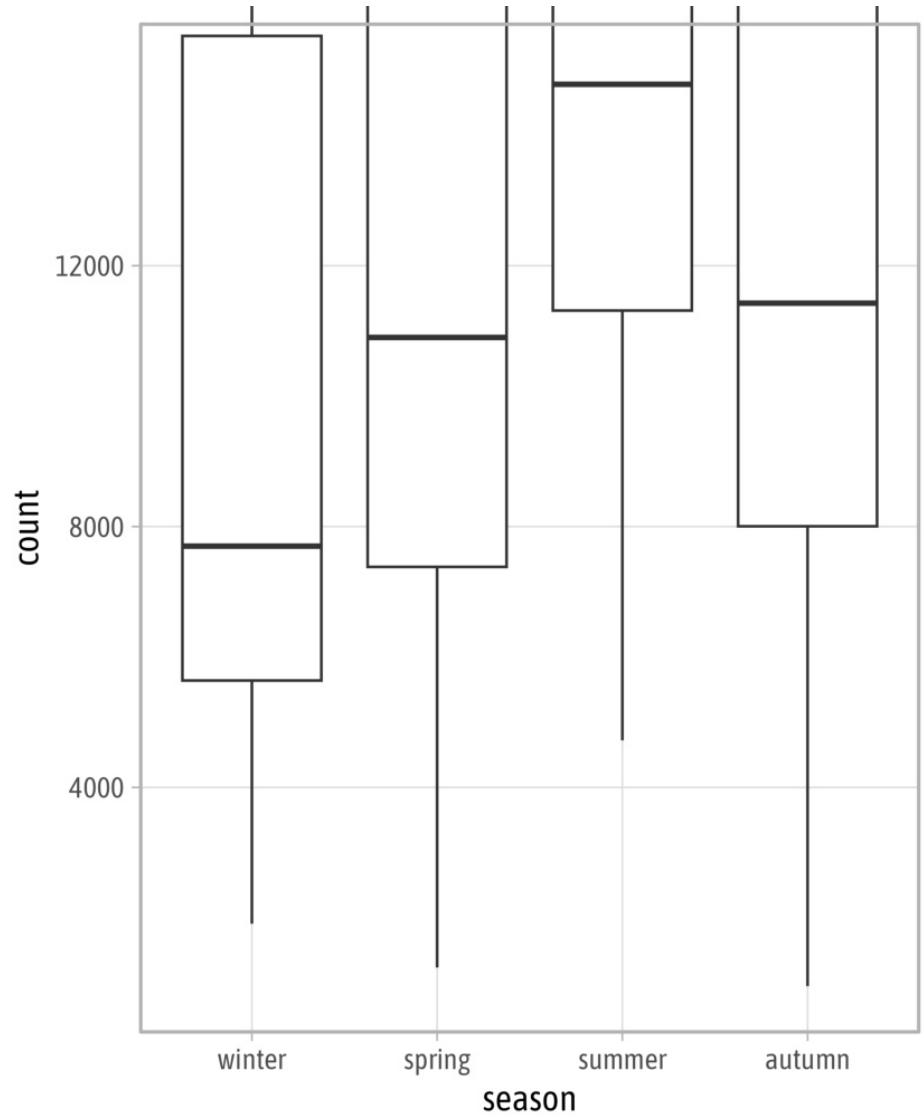


```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_boxplot() +  
6 scale_y_continuous(  
7   limits = c(NA, 15000)  
8 )
```



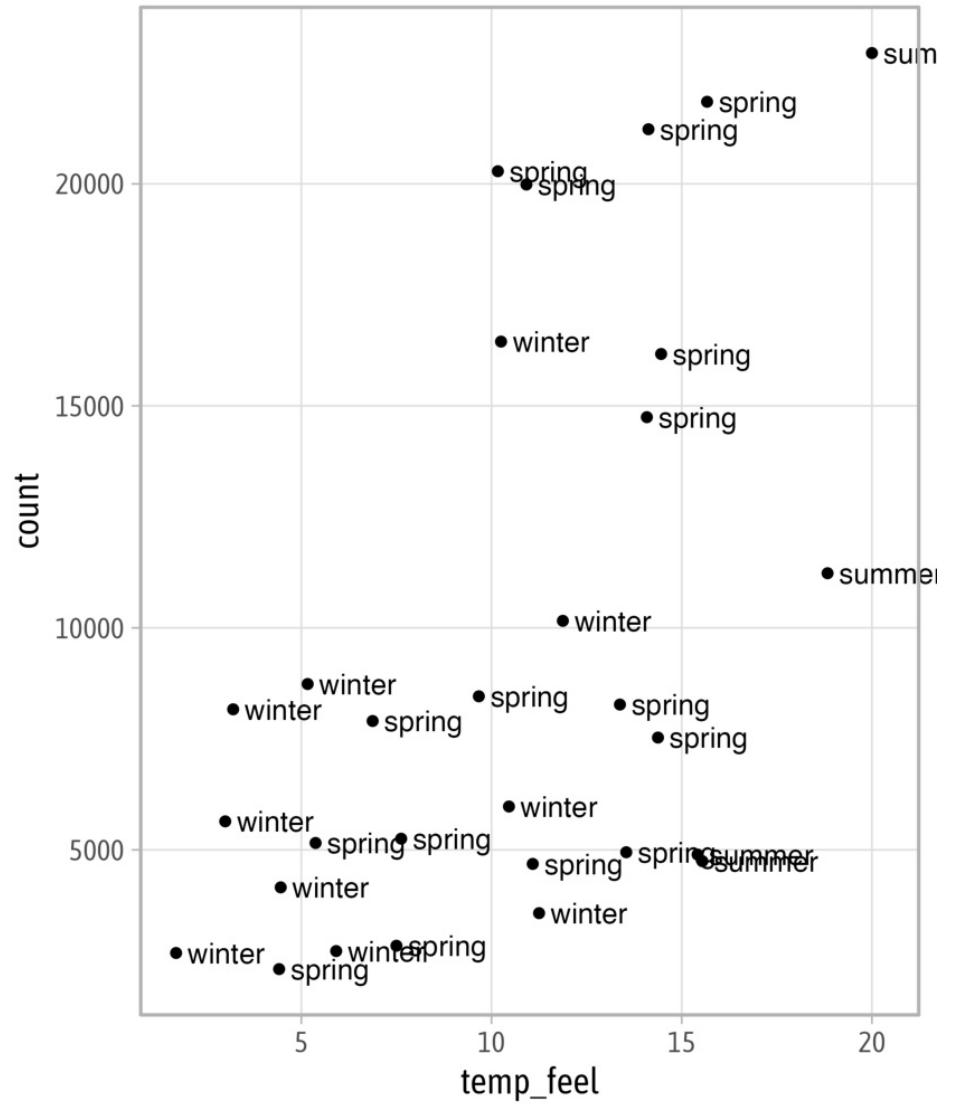
Clipping

```
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 )
```



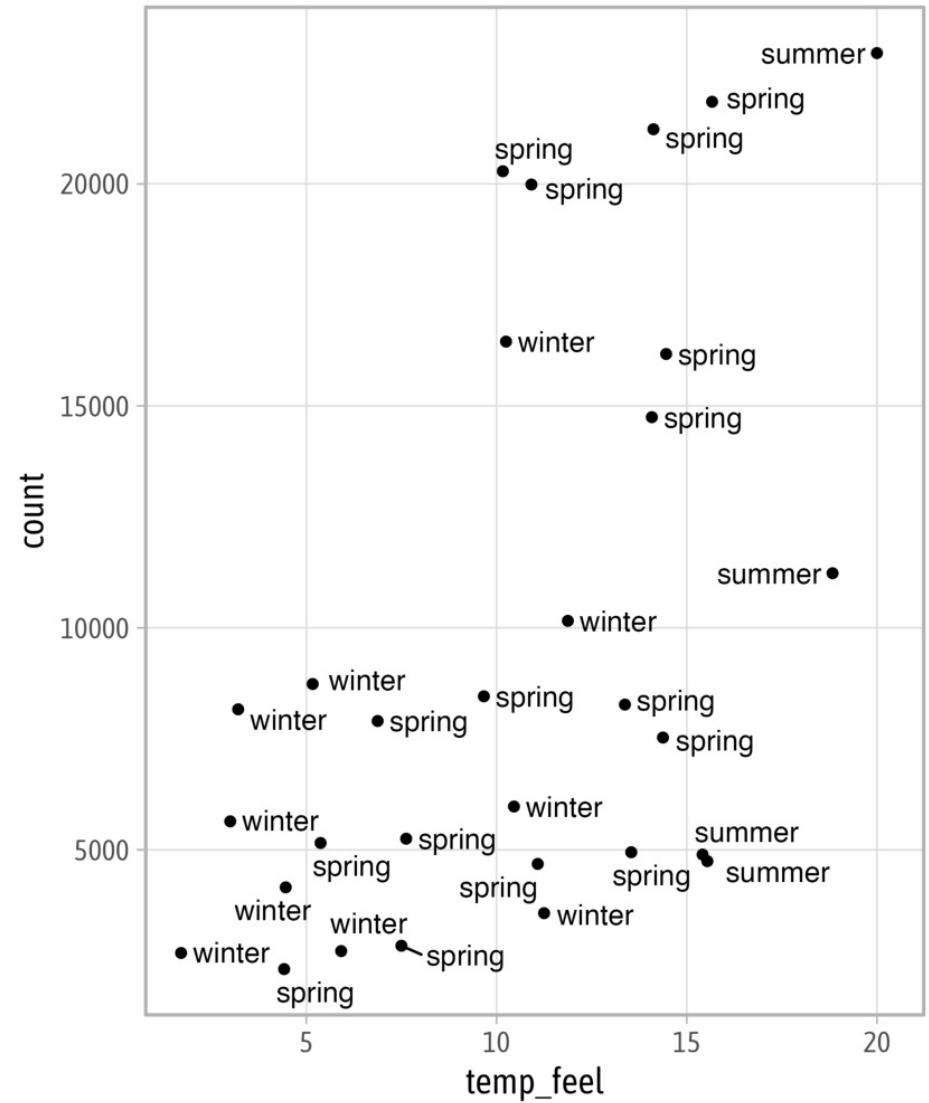
Clipping

```
1 ggplot(  
2   filter(bikes, is_holiday == TRUE),  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point() +  
6 geom_text(  
7   aes(label = season),  
8   nudge_x = .3,  
9   hjust = 0  
10) +  
11 coord_cartesian(  
12   clip = "off"  
13)
```



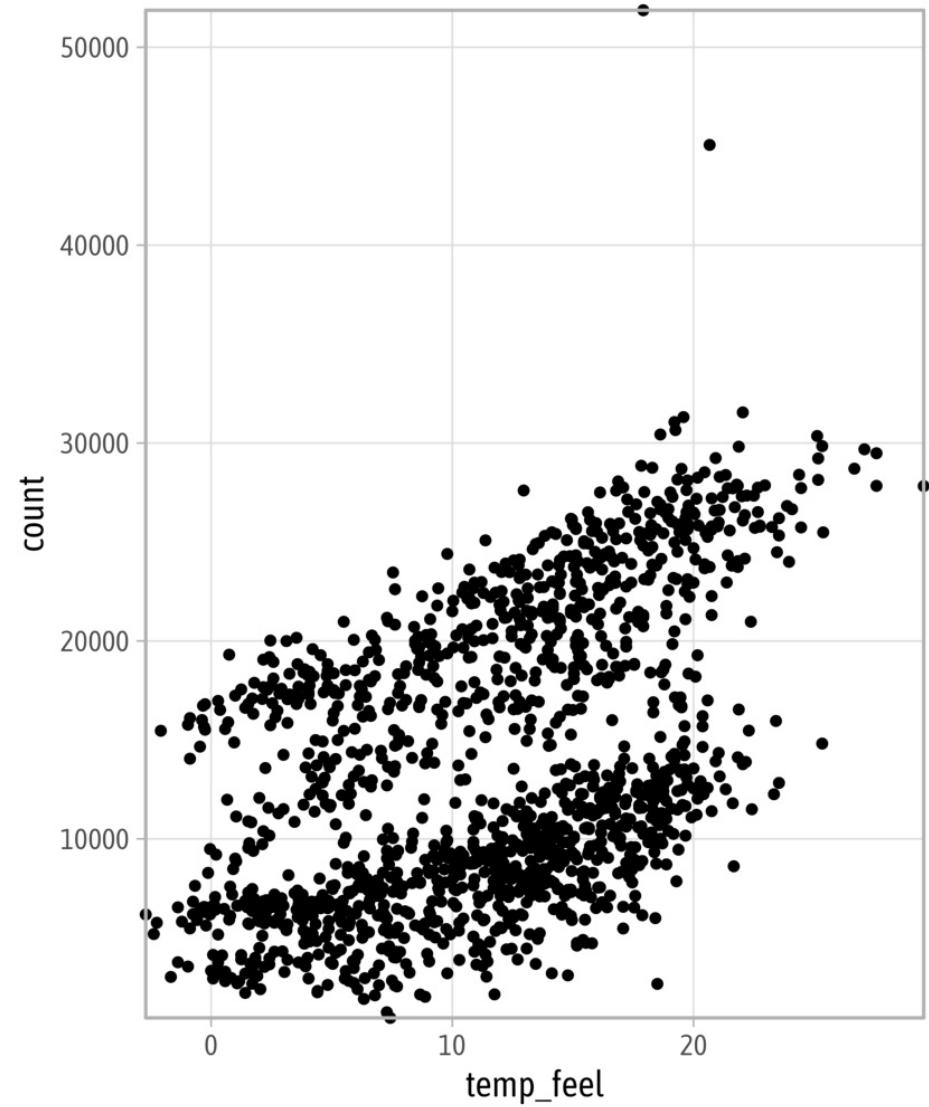
... or better use {ggrepel}

```
1 # install.packages("ggrepel")
2
3 ggplot(
4     filter(bikes, is_holiday == TRUE),
5     aes(x = temp_feel, y = count)
6 ) +
7 geom_point() +
8 ggrepel::geom_text_repel(
9     aes(label = season),
10    nudge_x = .3,
11    hjust = 0
12 ) +
13 coord_cartesian(
14     clip = "off"
15 )
```



Remove All Padding

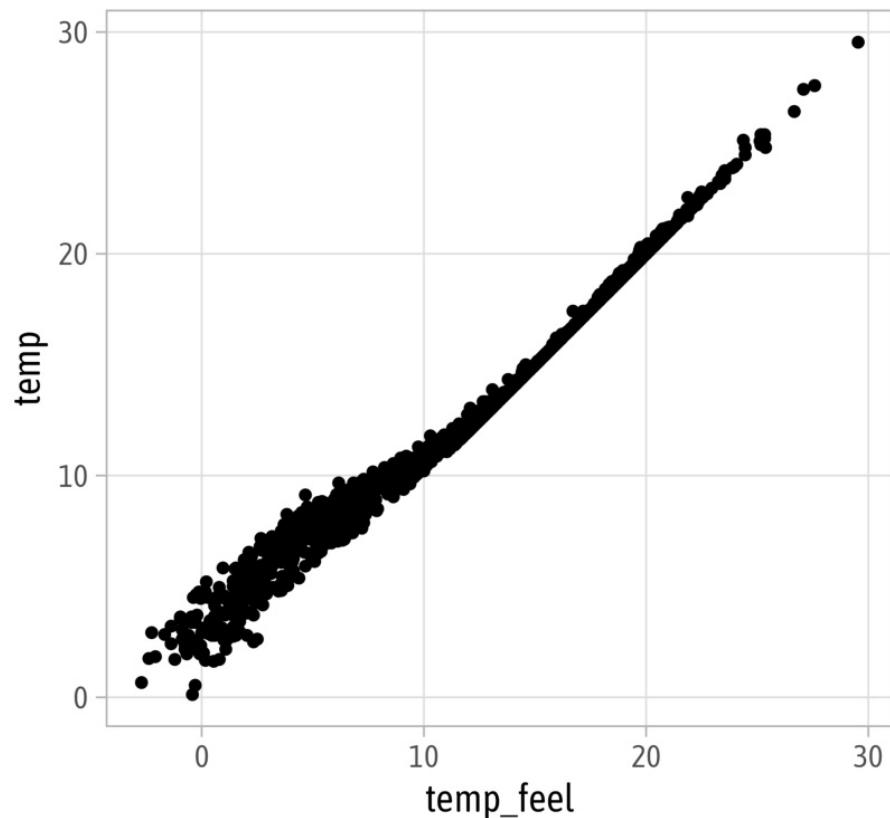
```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point() +  
6 coord_cartesian(  
7   expand = FALSE,  
8   clip = "off"  
9 )
```



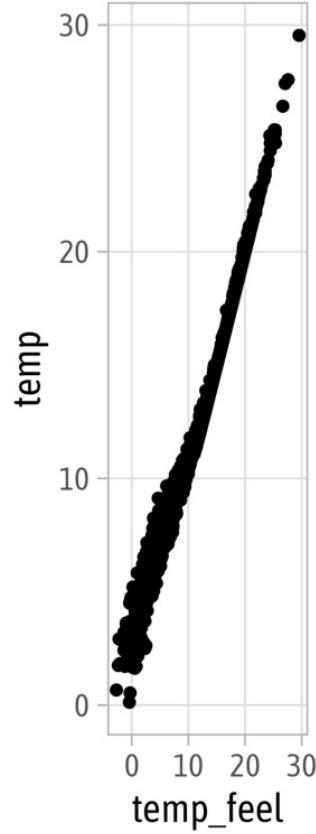
Fixed Coordinate System

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

...

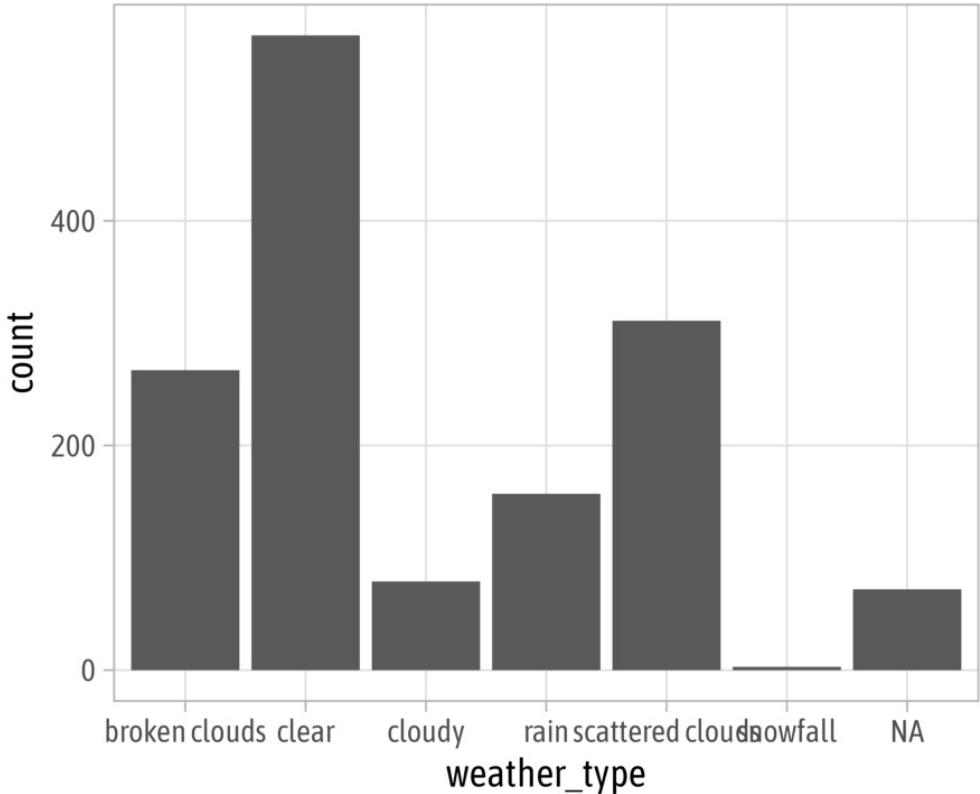


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = temp)  
4 ) +  
5 geom_point() +  
6 coord_fixed(ratio = 4)
```

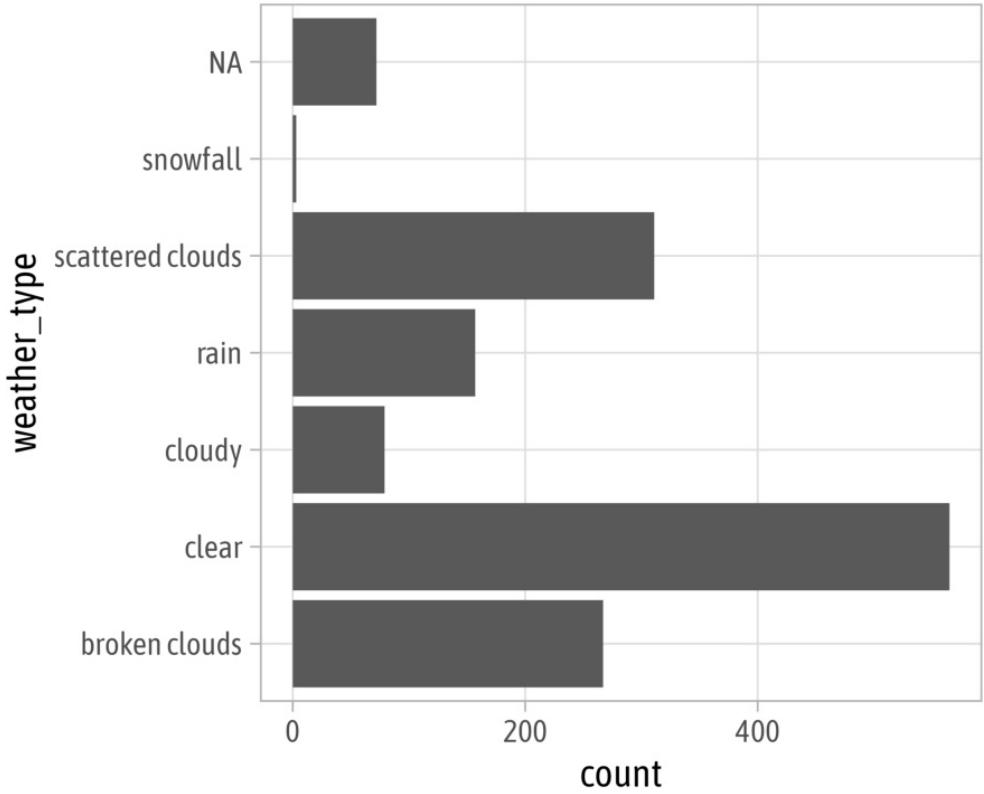


Flipped Coordinate System

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

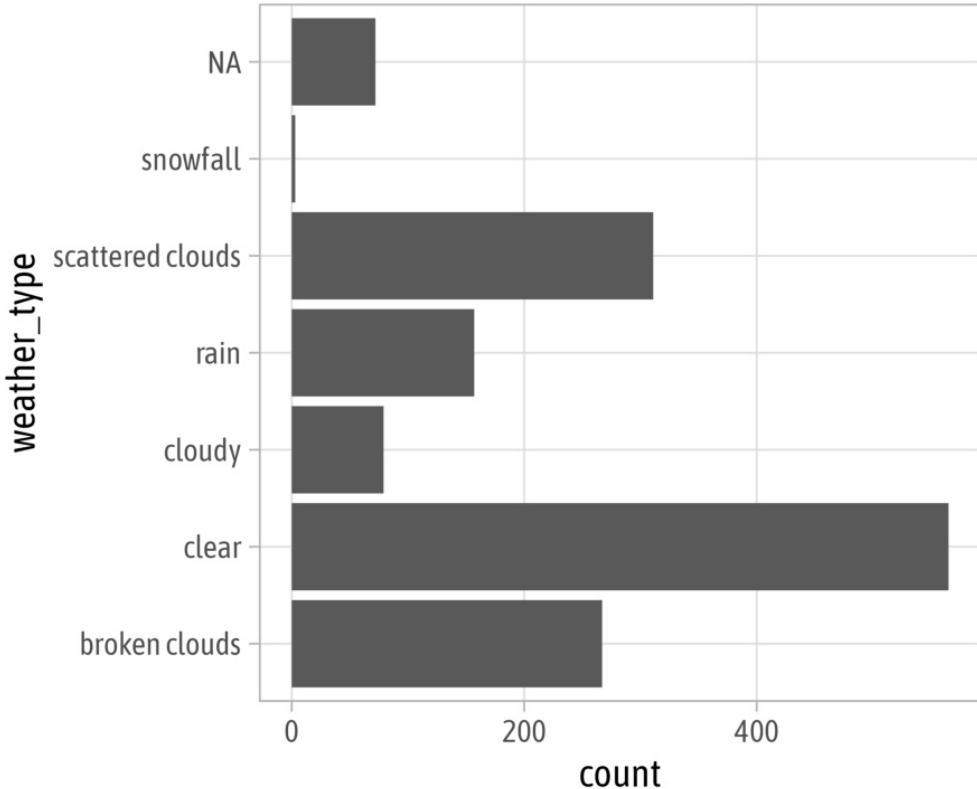


```
1 ggplot(  
2   bikes,  
3   aes(x = weather_type)  
4 ) +  
5 geom_bar() +  
6 coord_flip()
```

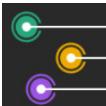
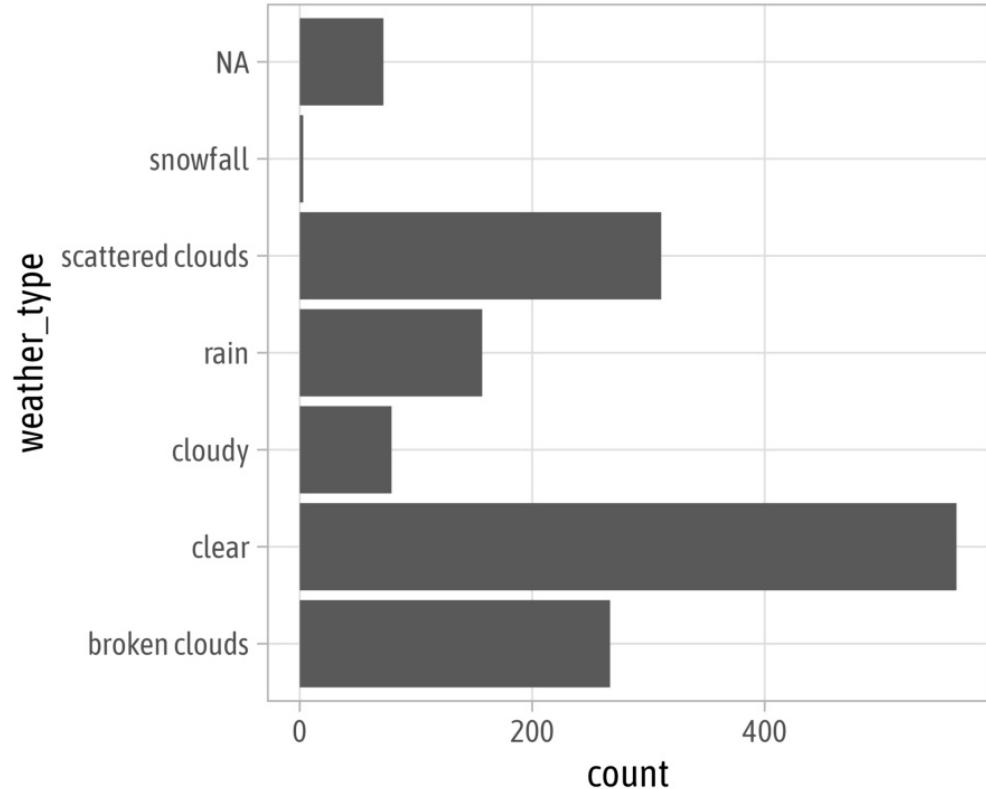


Flipped Coordinate System

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

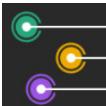
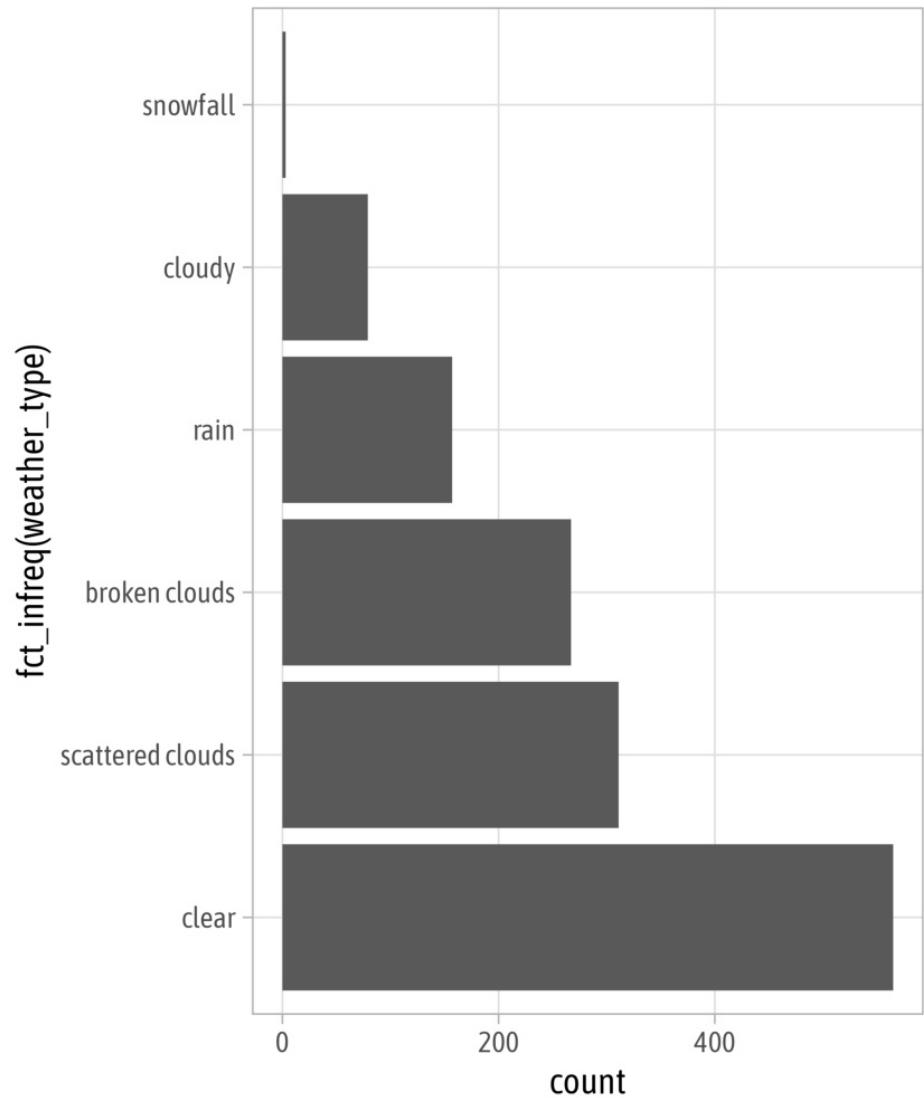


```
1 ggplot(  
2   bikes,  
3   aes(x = weather_type)  
4 ) +  
5 geom_bar() +  
6 coord_flip()
```



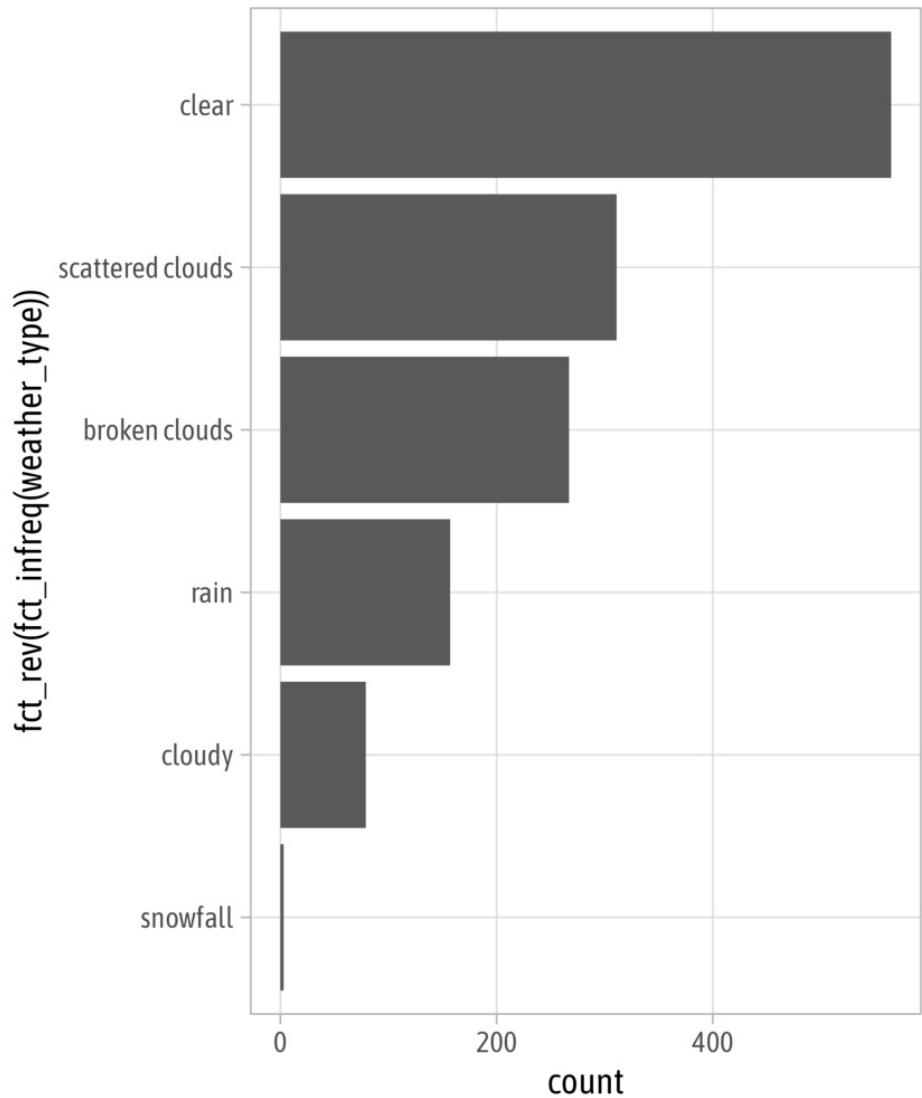
Reminder: Sort Your Bars!

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



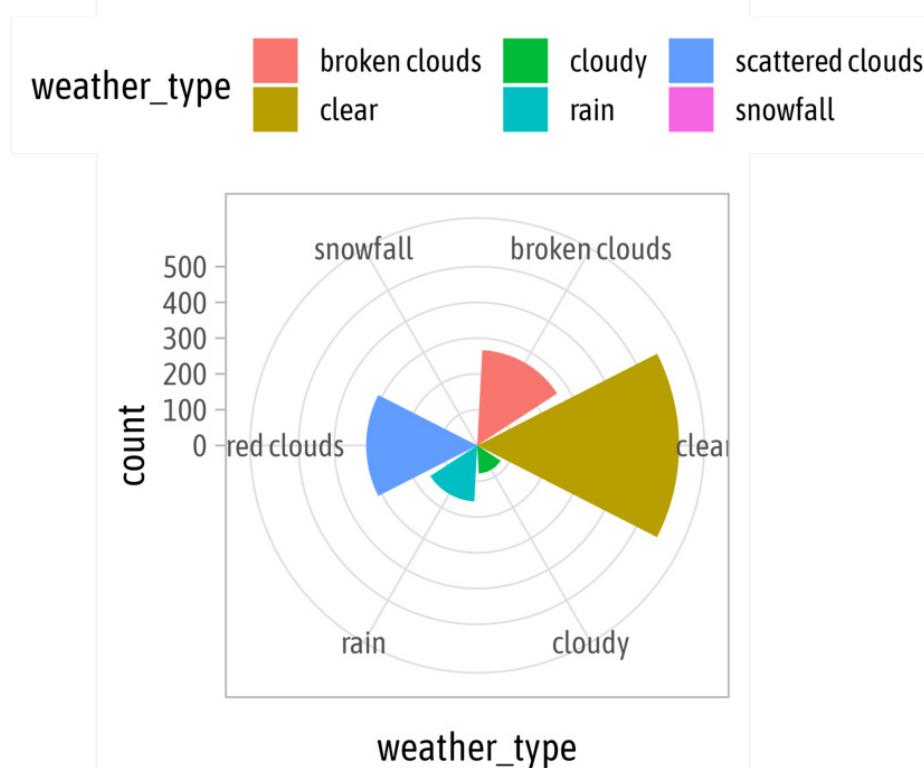
Reminder: Sort Your Bars!

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

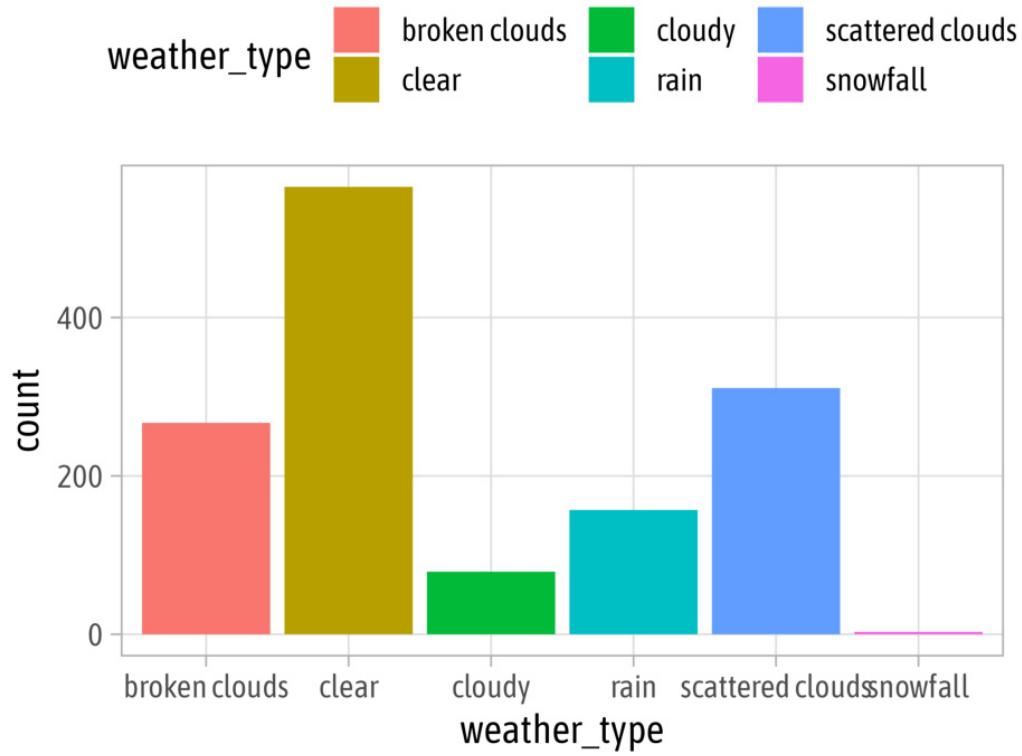


Circular Coordinate System

```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = weather_type,  
4        fill = weather_type)  
5 ) +  
6 geom_bar() +  
7 coord_polar()
```

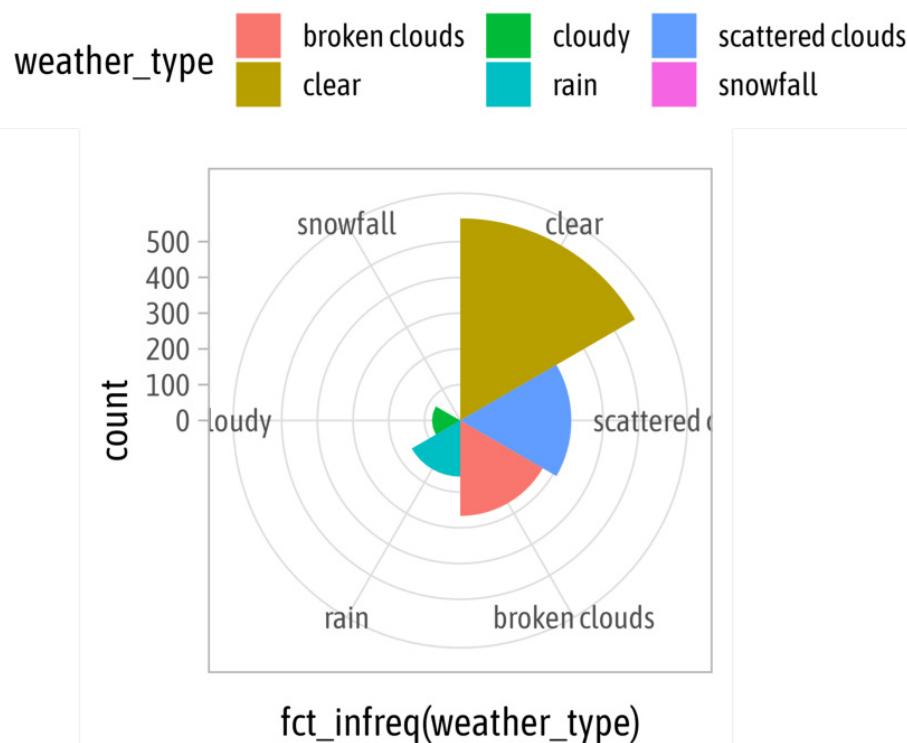


```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = weather_type,  
4        fill = weather_type)  
5 ) +  
6 geom_bar() +  
7 coord_cartesian()
```

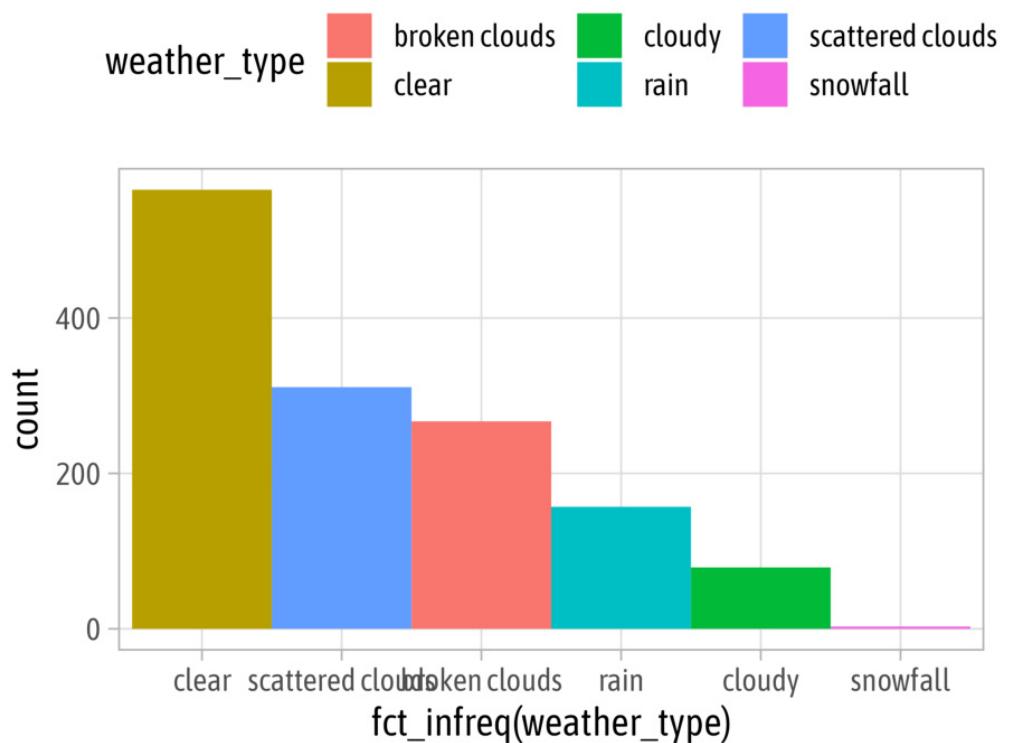


Circular Coordinate System

```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = fct_infreq(weather_type),  
4       fill = weather_type)  
5 ) +  
6 geom_bar(width = 1) +  
7 coord_polar()
```

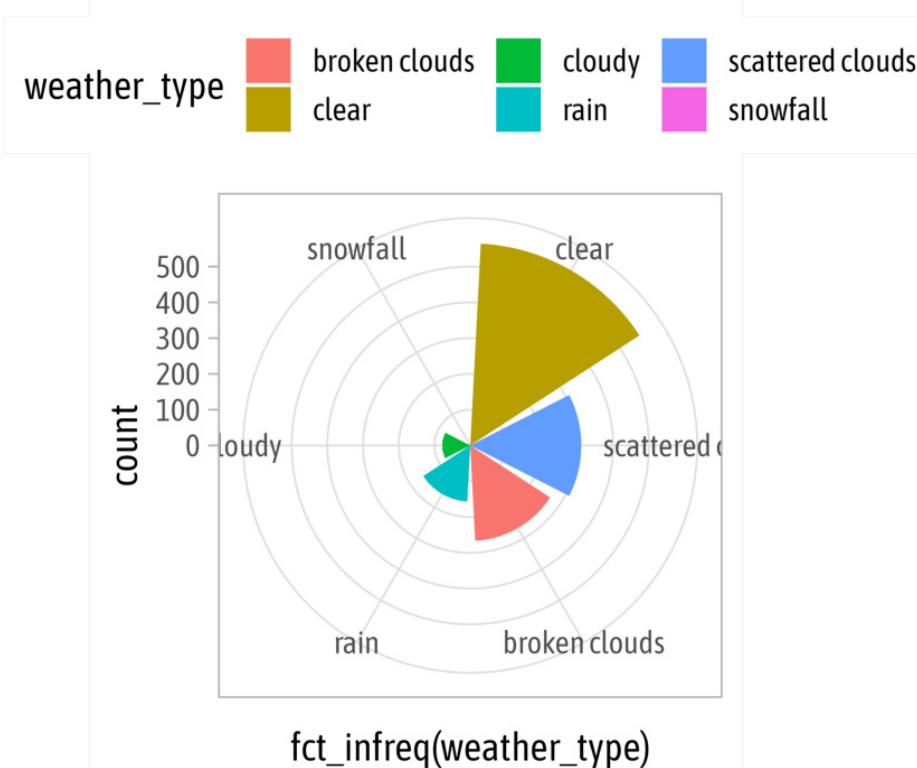


```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = fct_infreq(weather_type),  
4       fill = weather_type)  
5 ) +  
6 geom_bar(width = 1) +  
7 coord_cartesian()
```

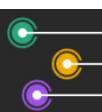
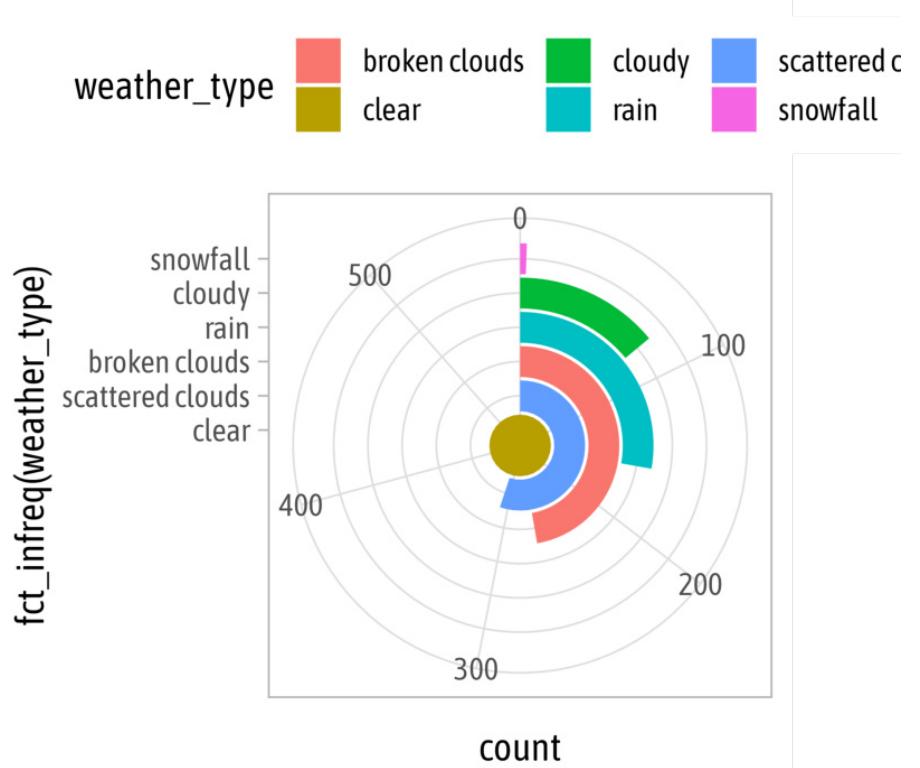


Circular Coordinate System

```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = fct_infreq(weather_type),  
4       fill = weather_type)  
5 ) +  
6 geom_bar() +  
7 coord_polar(theta = "x")
```

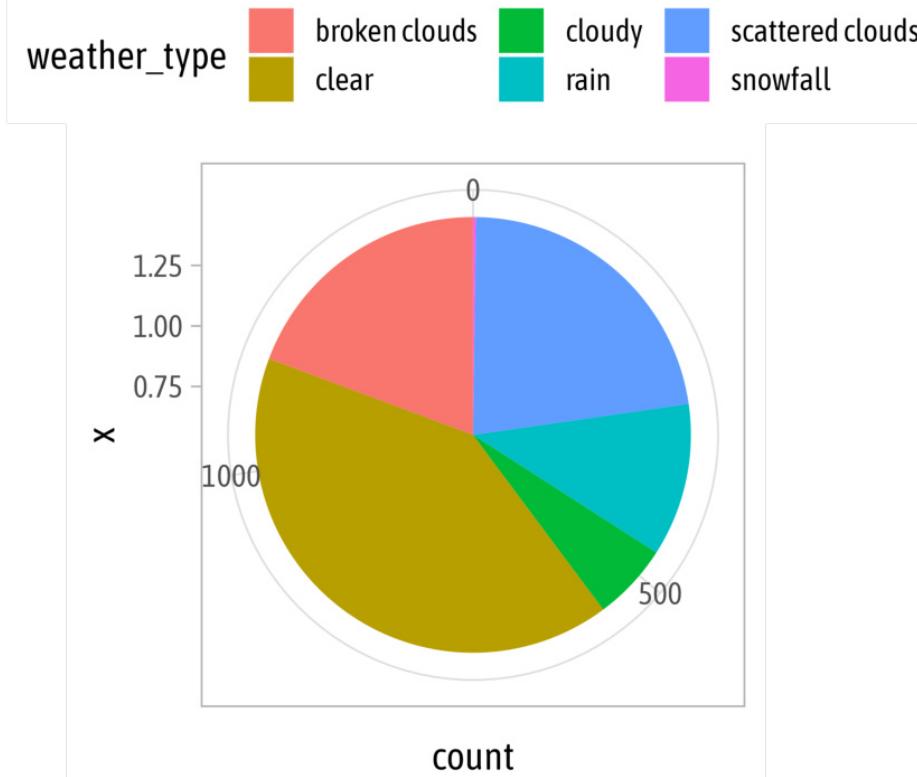


```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = fct_infreq(weather_type),  
4       fill = weather_type)  
5 ) +  
6 geom_bar() +  
7 coord_polar(theta = "y")
```

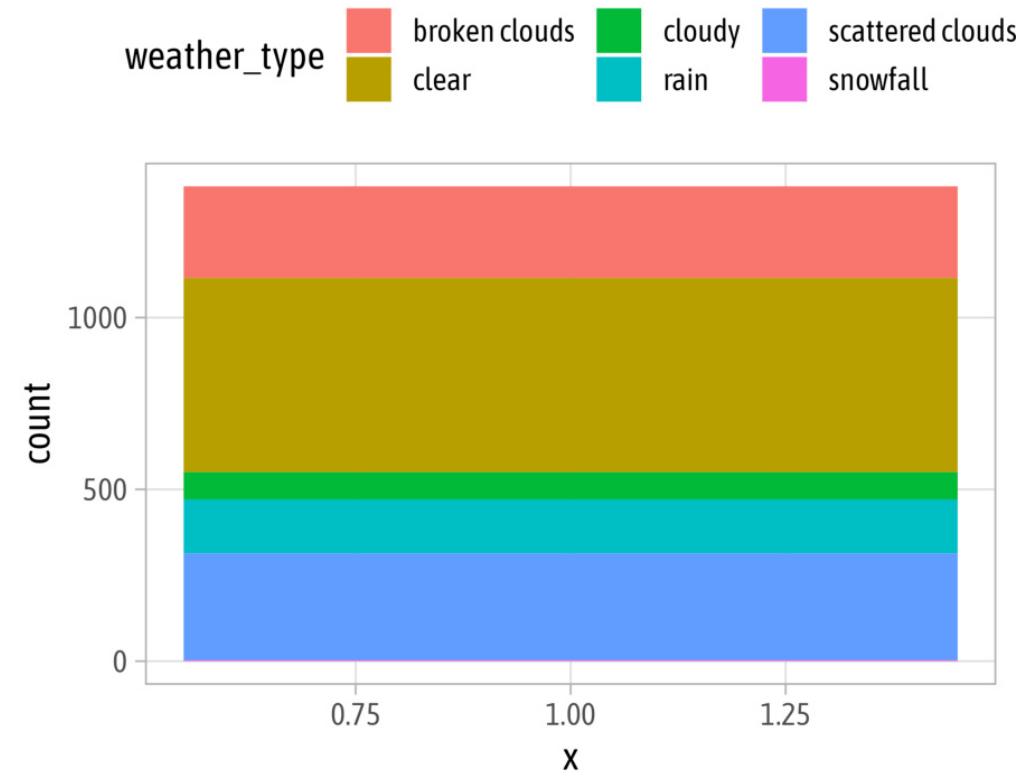


Circular Corrdinate System

```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = 1, fill = weather_type)  
4 ) +  
5 geom_bar(position = "stack") +  
6 coord_polar(theta = "y")
```

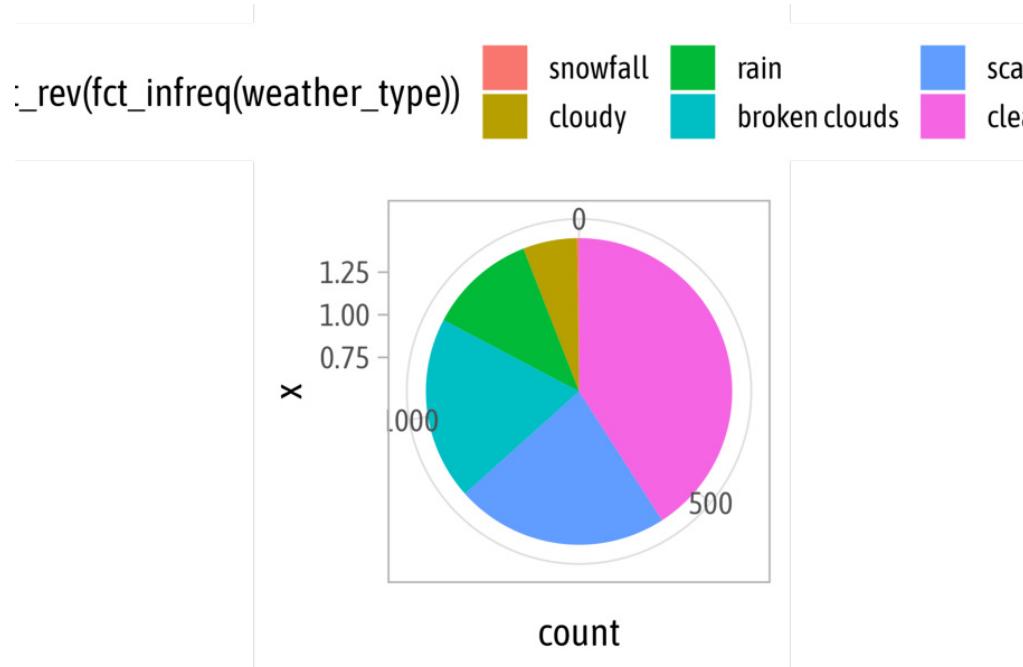


```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = 1, fill = weather_type)  
4 ) +  
5 geom_bar(position = "stack") +  
6 coord_cartesian()
```

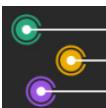
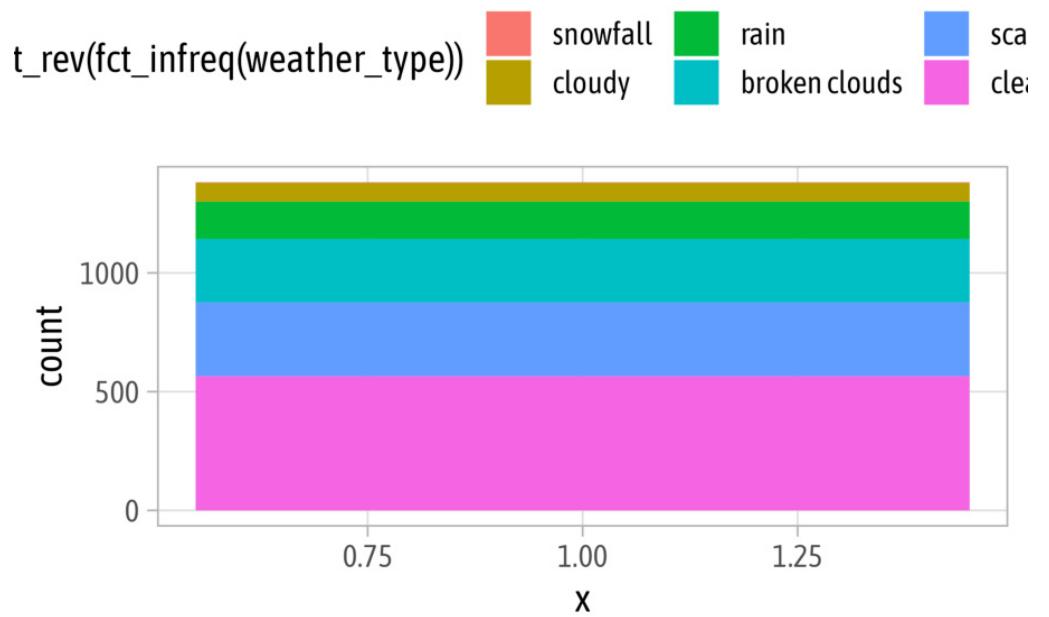


Circular Corrdinate System

```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = 1,  
4     fill = fct_rev(  
5       fct_infreq(weather_type))  
6     )  
7   ) +  
8   geom_bar(position = "stack") +  
9   coord_polar(theta = "y")
```

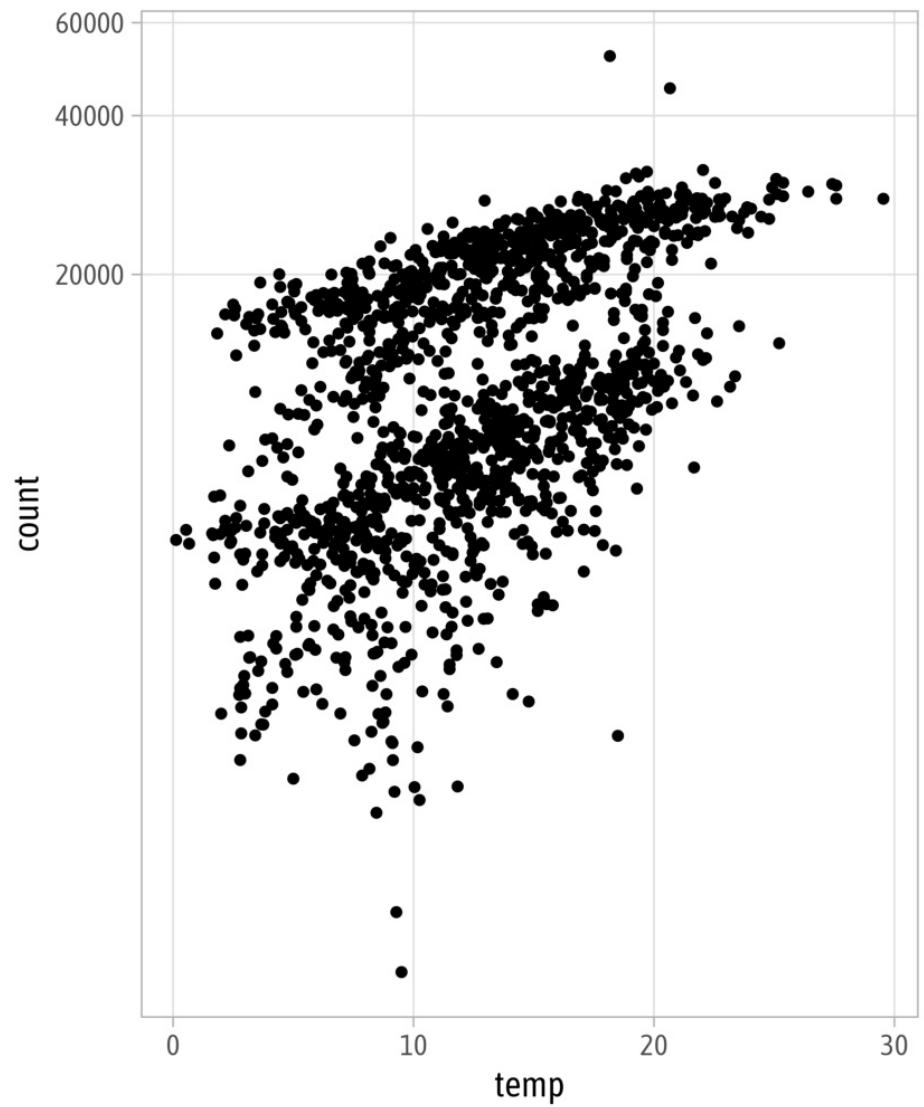


```
1 ggplot(  
2   filter(bikes, !is.na(weather_type)),  
3   aes(x = 1,  
4     fill = fct_rev(  
5       fct_infreq(weather_type))  
6     )  
7   ) +  
8   geom_bar(position = "stack") +  
9   coord_cartesian()
```



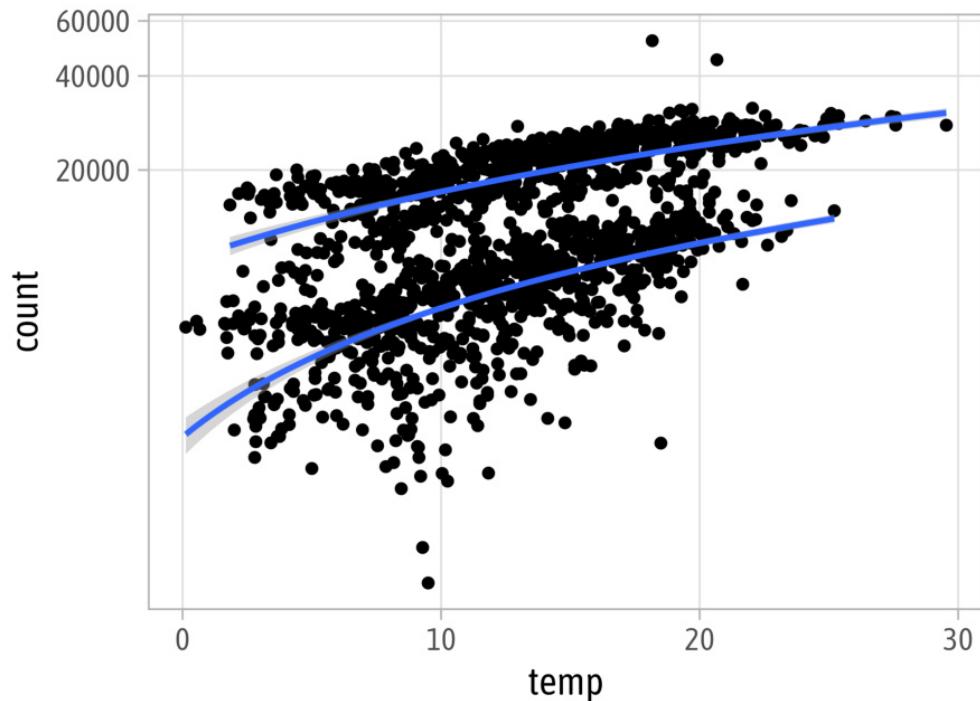
Transform a Coordinate System

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = count)  
4 ) +  
5 geom_point() +  
6 coord_trans(y = "log10")
```



Transform a Coordinate System

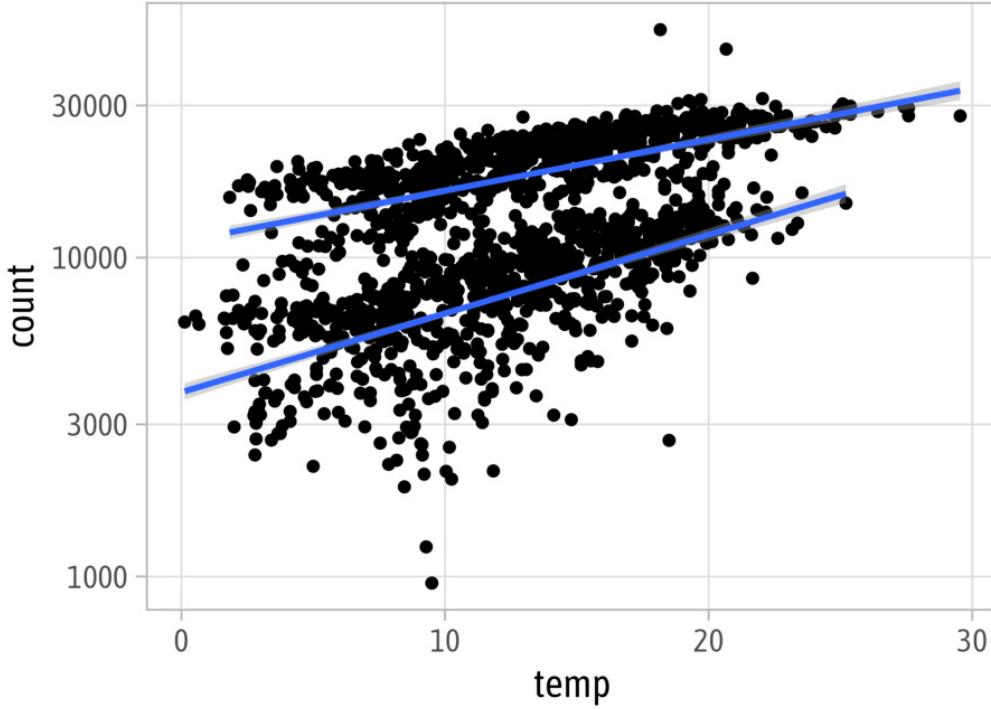
```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = count,  
4       group = day_night)  
5 ) +  
6 geom_point() +  
7 geom_smooth(method = "lm") +  
8 coord_trans(y = "log10")
```



...

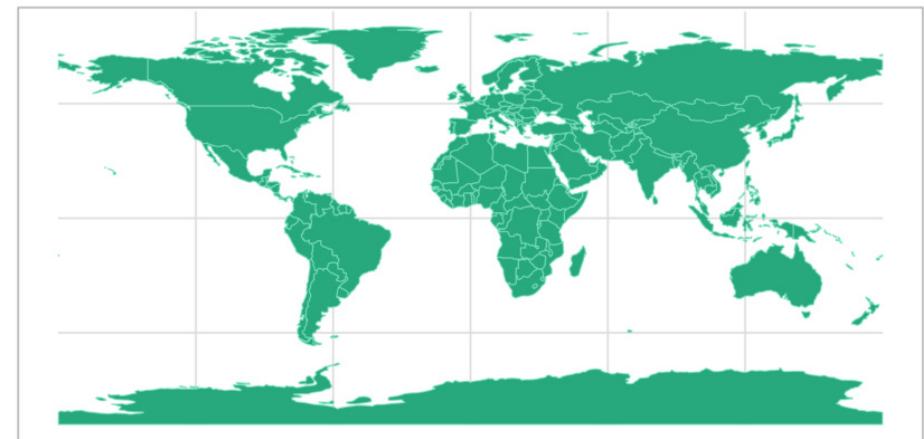


```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = count,  
4       group = day_night)  
5 ) +  
6 geom_point() +  
7 geom_smooth(method = "lm") +  
8 scale_y_log10()
```



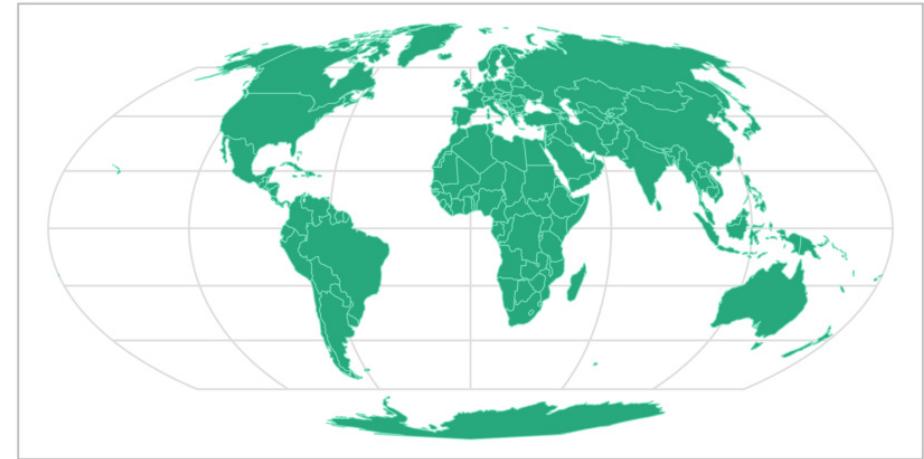
Spatial Coordinate (Reference) Systems

```
1 countries <- rnaturalearth::ne_countries(  
2   returnclass = "sf"  
3 )  
4  
5 ggplot() +  
6   geom_sf(  
7     data = countries,  
8     color = "#79dfbd",  
9     fill = "#28a87d",  
10    size = .3  
11  )
```



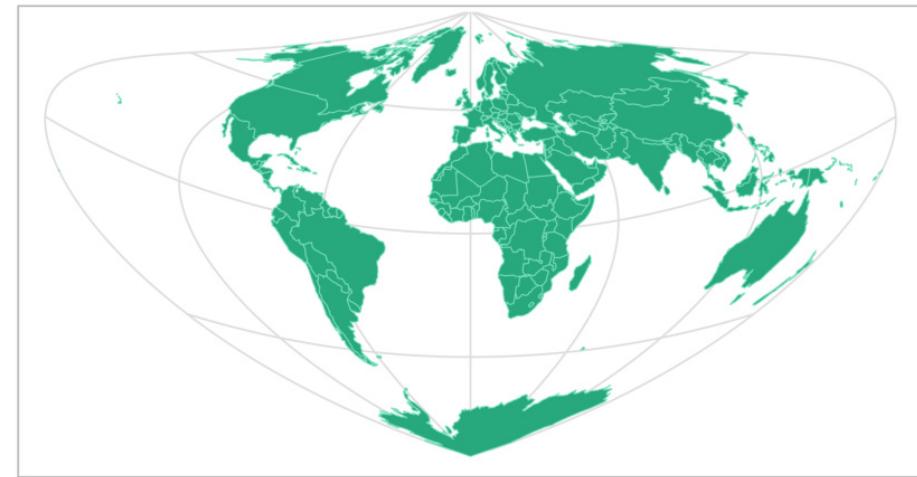
Spatial Coordinate (Reference) Systems

```
1 ggplot() +
2   geom_sf(
3     data = countries,
4     color = "#79dfbd",
5     fill = "#28a87d",
6     size = .3
7   ) +
8   coord_sf(crs = "+proj=moll")
```



Spatial Coordinate (Reference) Systems

```
1 ggplot() +
2   geom_sf(
3     data = countries,
4     color = "#79dfbd",
5     fill = "#28a87d",
6     size = .3
7   ) +
8   coord_sf(crs = "+proj=bonne +lat_1=10")
```



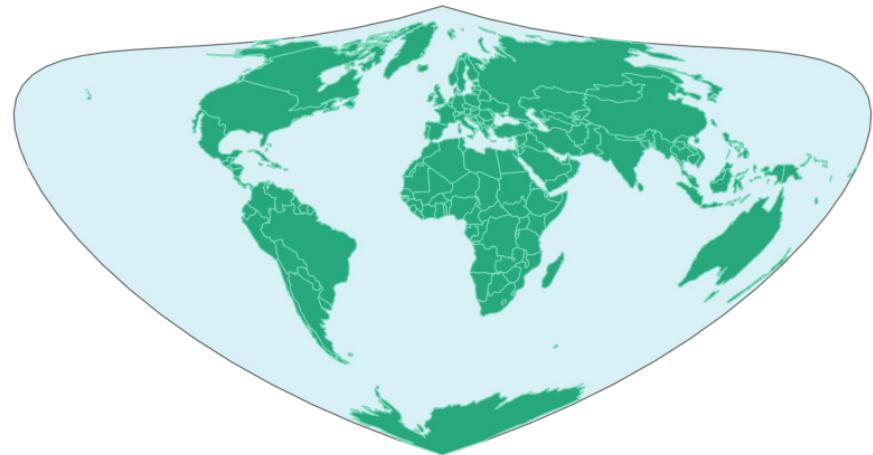
Spatial Coordinate (Reference) Systems

```
1 oceans <- rnaturalearth::ne_download(  
2   category = "physical", type = "ocean", returnclass = "sf"  
3 )
```



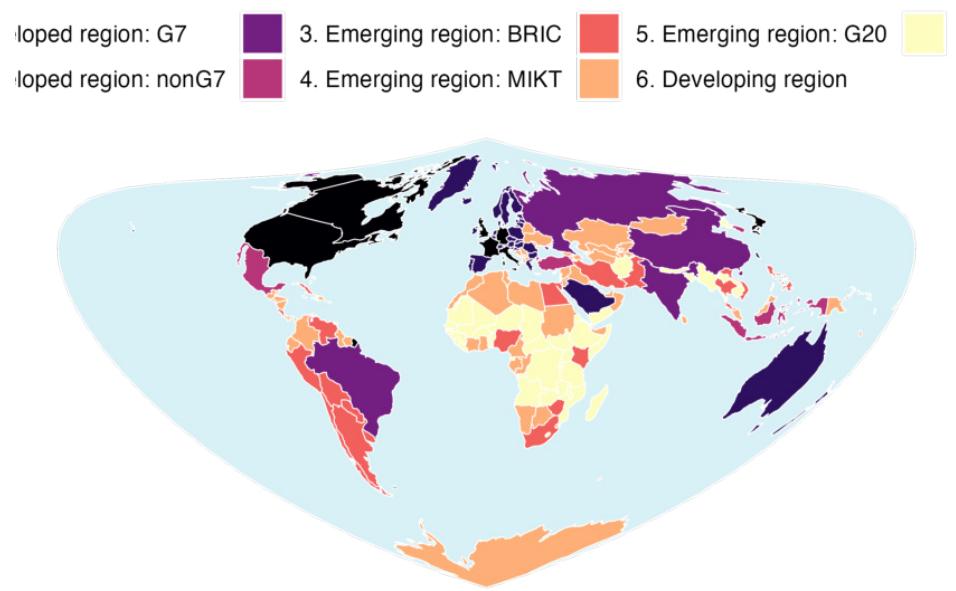
Spatial Coordinate (Reference) Systems

```
1 ggplot() +
2   geom_sf(
3     data = oceans,
4     fill = "#d8f1f6"
5   ) +
6   geom_sf(
7     data = countries,
8     color = "#79dfbd",
9     fill = "#28a87d",
10    size = .3
11  ) +
12  coord_sf(crs = "+proj=bonne +lat_1=10") +
13  theme_void()
```



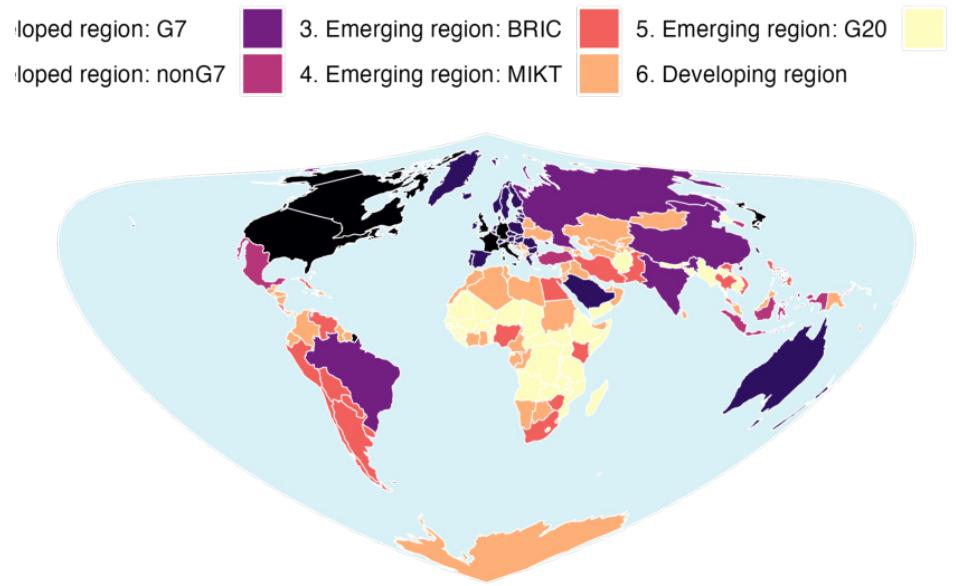
Mapping of Visual Properties

```
1 ggplot() +
2   geom_sf(
3     data = oceans,
4     fill = "#d8f1f6",
5     color = "white"
6   ) +
7   geom_sf(
8     data = countries,
9     aes(fill = economy),
10    color = "white",
11    size = .3
12  ) +
13  coord_sf(crs = "+proj=bonne +lat_1=10") +
14  scale_fill_viridis_d(option = "magma") +
15  theme_void() +
16  theme(legend.position = "top")
```



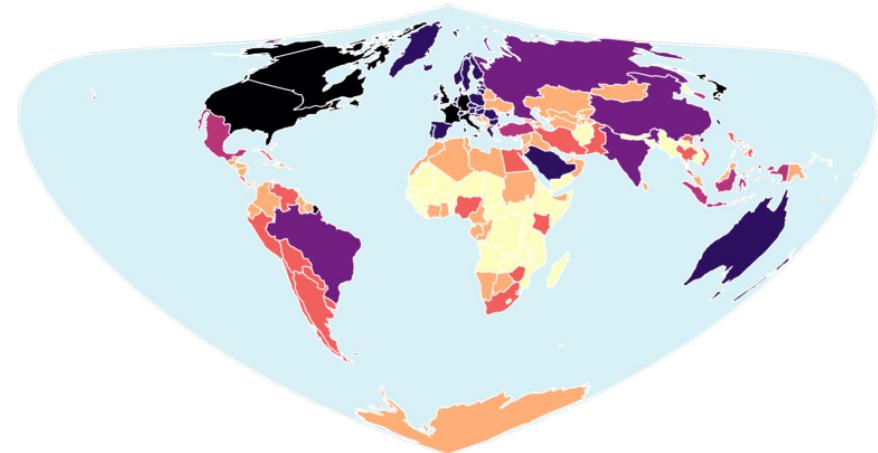
Mapping of Visual Properties

```
1 ggplot() +
2   geom_sf(
3     data = oceans,
4     fill = "#d8f1f6",
5     color = "white"
6   ) +
7   geom_sf(
8     data = countries,
9     aes(fill = economy),
10    color = "white",
11    size = .3
12  ) +
13  coord_sf(crs = "+proj=bonne +lat_1=10") +
14  scale_fill_viridis_d(option = "magma") +
15  theme_void() +
16  theme(legend.position = "top")
```



Mapping of Visual Properties

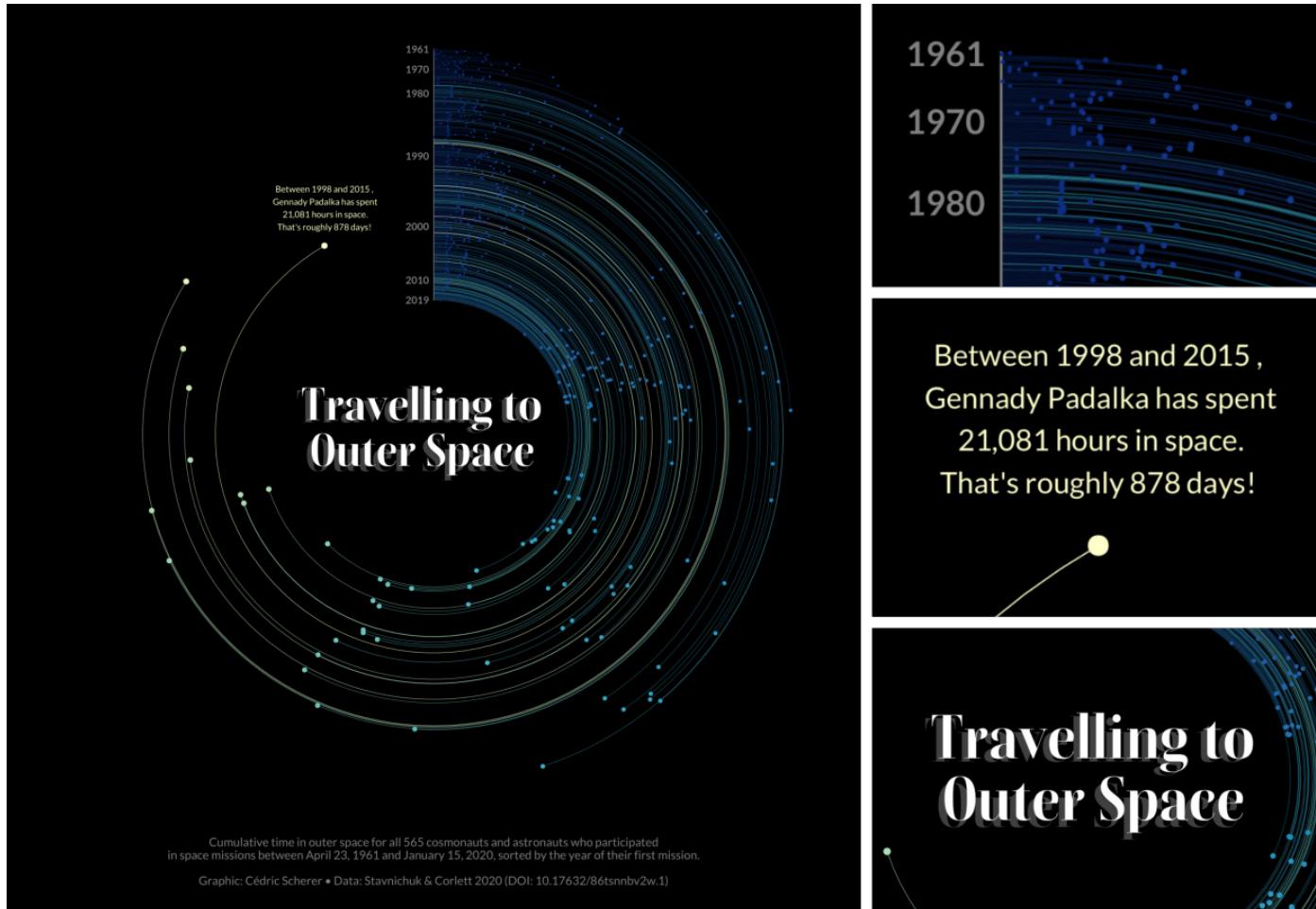
```
1 ggplot() +
2   geom_sf(
3     data = oceans,
4     fill = "#d8f1f6",
5     color = "white"
6   ) +
7   geom_sf(
8     data = countries,
9     aes(fill = economy),
10    color = "white",
11    size = .3
12  ) +
13  coord_sf(crs = "+proj=bonne +lat_1=10") +
14  scale_fill_viridis_d(
15    option = "magma",
16    guide = guide_legend(
17      ncol = 2, byrow = TRUE,
18      title.position = "top"
19    )
20  ) +
21  theme_void() +
22  theme(legend.position = "top")
```



Your Turn: Components

- Have a look at the following visualization of the cumulative time that cosmo- and astronauts have spent in outer space. The data also contains information on the year of their first and last travel, respectively.
- Together with your group, discuss which layers and modifications are needed to create such a chart with `{ggplot2}`.
- Note down the aesthetics, geometries, and scales used for each element of this graphic.
- What is the coordinate system? Have any adjustments been made?
- Which theme was used and how was it modified?





Layers

- **geom_point()**
 - `aes(x = id, y = hours, size = hours)`
- **geom_linerange()**
 - `aes(x = id, ymin = 0, ymax = hours, color = hours, alpha = hours)`
- **geom_point()**
 - `aes(x = id, y = 0), shape = 15, color = "#808080"`
- **geom_text()**
 - `aes(x = id, y = 0, label = year), size = 4.5, hjust = 1.2`
- **geom_text()**
 - `aes(x = id, y = hours, label = max), size = 3.9, vjust = -.35`



Scales

- **scale_x_continuous()**
 - limits = c(-300, NA), expand = c(0, 0)
- **scale_y_continuous()**
 - limits = c(0, 230000), expand = c(0, 0)
- **scale_color_distiller()**
 - palette = "YlGnBu", direction = -1
- **scale_size()**
 - range = c(.001, 3)
- **scale_alpha()**
 - range = c(.33, .95)



Coordinate System

- **coord_polar()**

- theta = "y"

Theme

- **theme_void()**

- legend.position = "none"
 - plot.background = element_rect(fill = "black")
 - plot.margin = margin(-70, -70, -70, -70)
 - plot.caption = element_text(hjust = .5, margin = margin(-100, 0, 100, 0), ...)

Title

- 2 x **annotate(geom = "text", x = -300, y = 0, ...)**



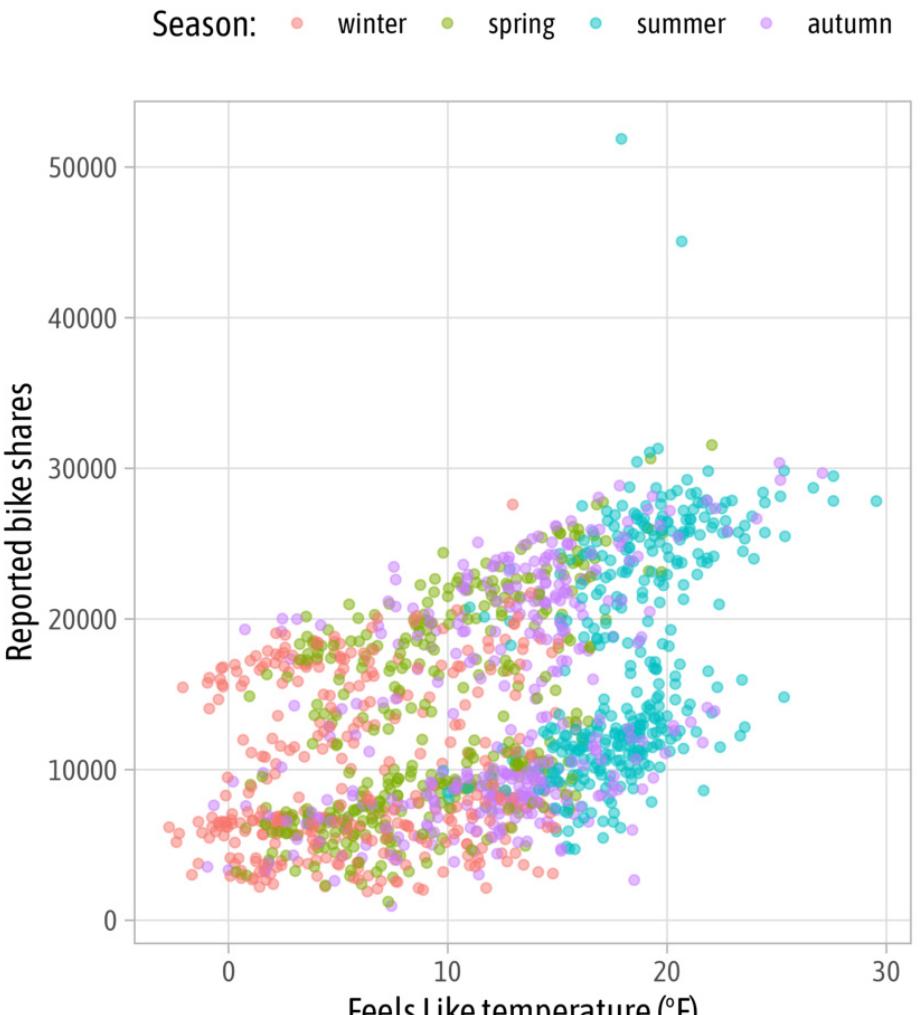
Data Visualization with {ggplot2}

— Styling Labels —



Styling Labels with {ggtext}

```
1 g <- ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count,  
4       color = season)  
5 ) +  
6 geom_point(  
7   alpha = .5  
8 ) +  
9 labs(  
10  x = "Feels Like temperature (°F)",  
11  y = "Reported bike shares",  
12  caption = "Data: TfL",  
13  color = "Season:"  
14 )  
15 g
```

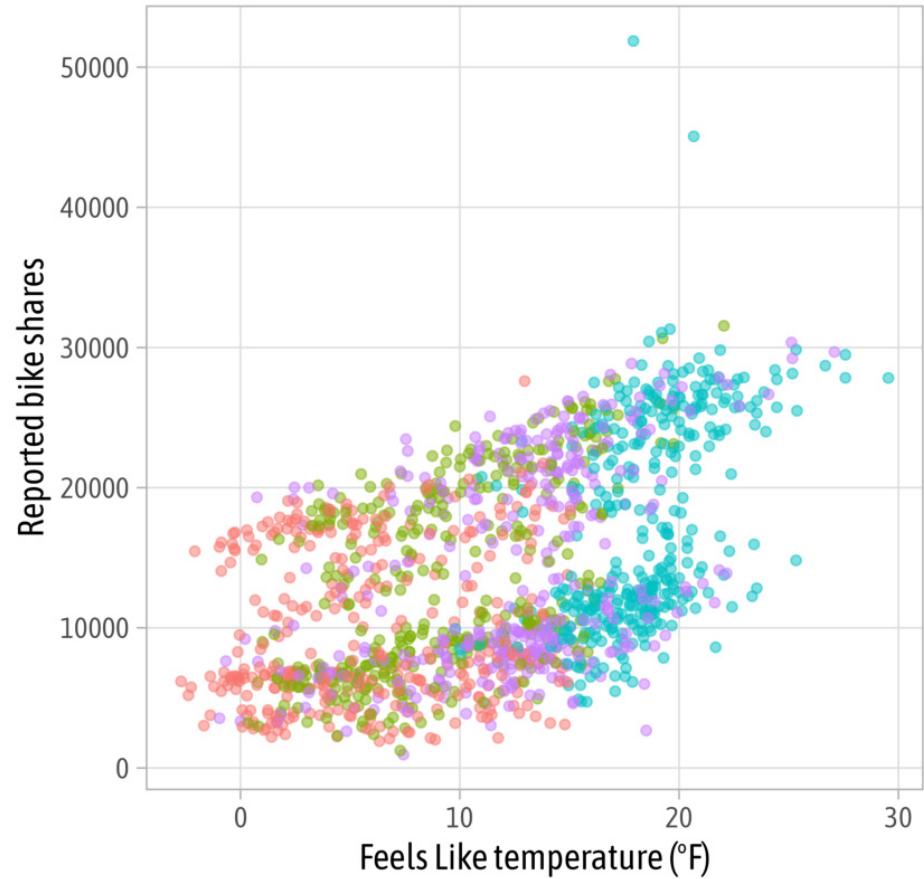


Styling Labels with {ggtext}

```
1 # install.packages("ggtext")
2 g +
3   ggttitle("**TfL bike sharing trends by _seas
```

TfL bike sharing trends by _season_

Season: ● winter ● spring ● summer ● autumn



Data: TfL

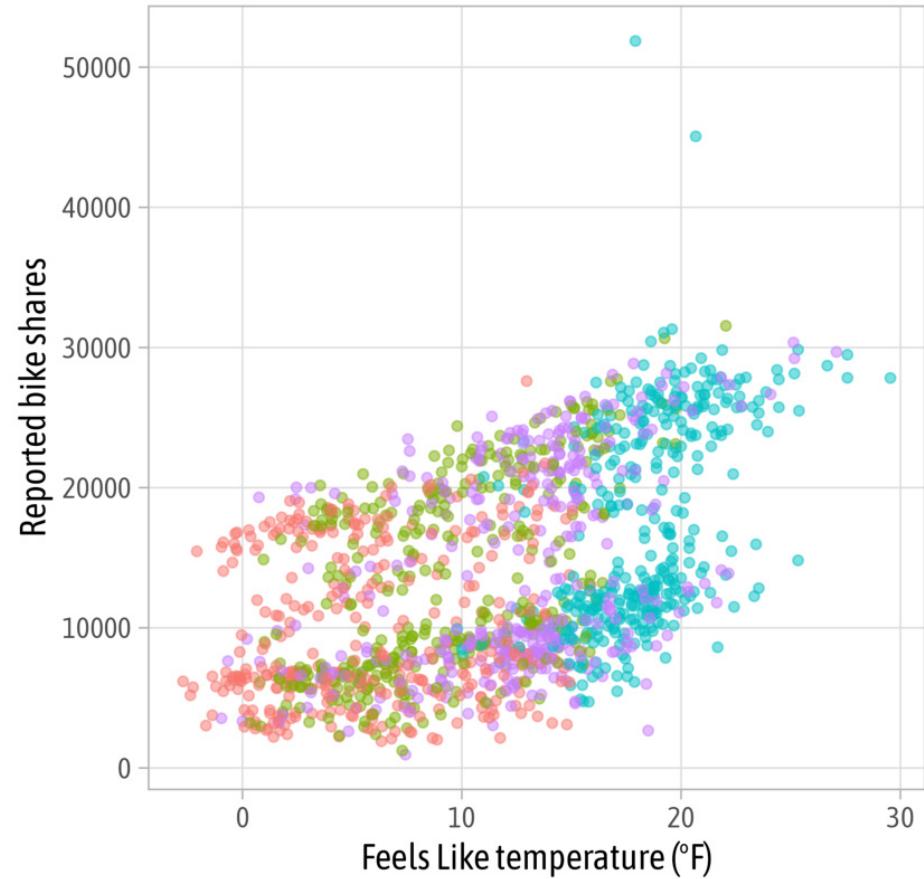


Styling Labels with {ggtext}

```
1 # install.packages("ggtext")
2 g +
3   ggtitle("**TfL bike sharing trends by _seas
4   theme(
5     plot.title = ggtext::element_markdown()
6   )
```

TfL bike sharing trends by season

Season: ● winter ● spring ● summer ● autumn



Data: TfL

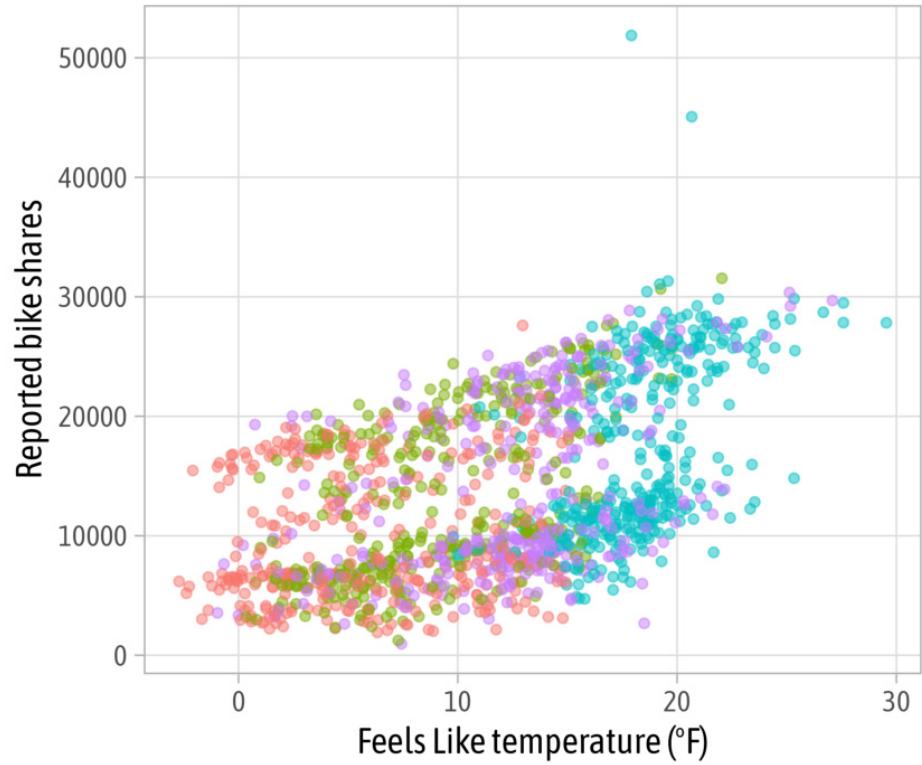


Styling Labels with {ggtext}

```
1 # install.packages("ggtext")
2 g +
3   ggtitle("<b style='font-family:tabular; font-size: 2em; font-weight: bold;'>TfL bike sharing trends by season</b>")
4   theme(
5     plot.title = ggtext::element_markdown()
6   )
```

TfL bike sharing trends by season

Season: ● winter ● spring ● summer ● autumn



Data: TfL



Data Visualization with {ggplot2}

— Multi-Panel Graphics —



patchwork

Combine + arrange
your ggplots!

PLAN:
 $(P1+P2)/P3$

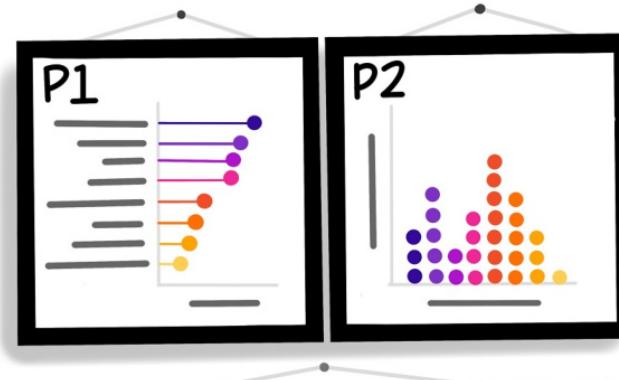
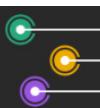
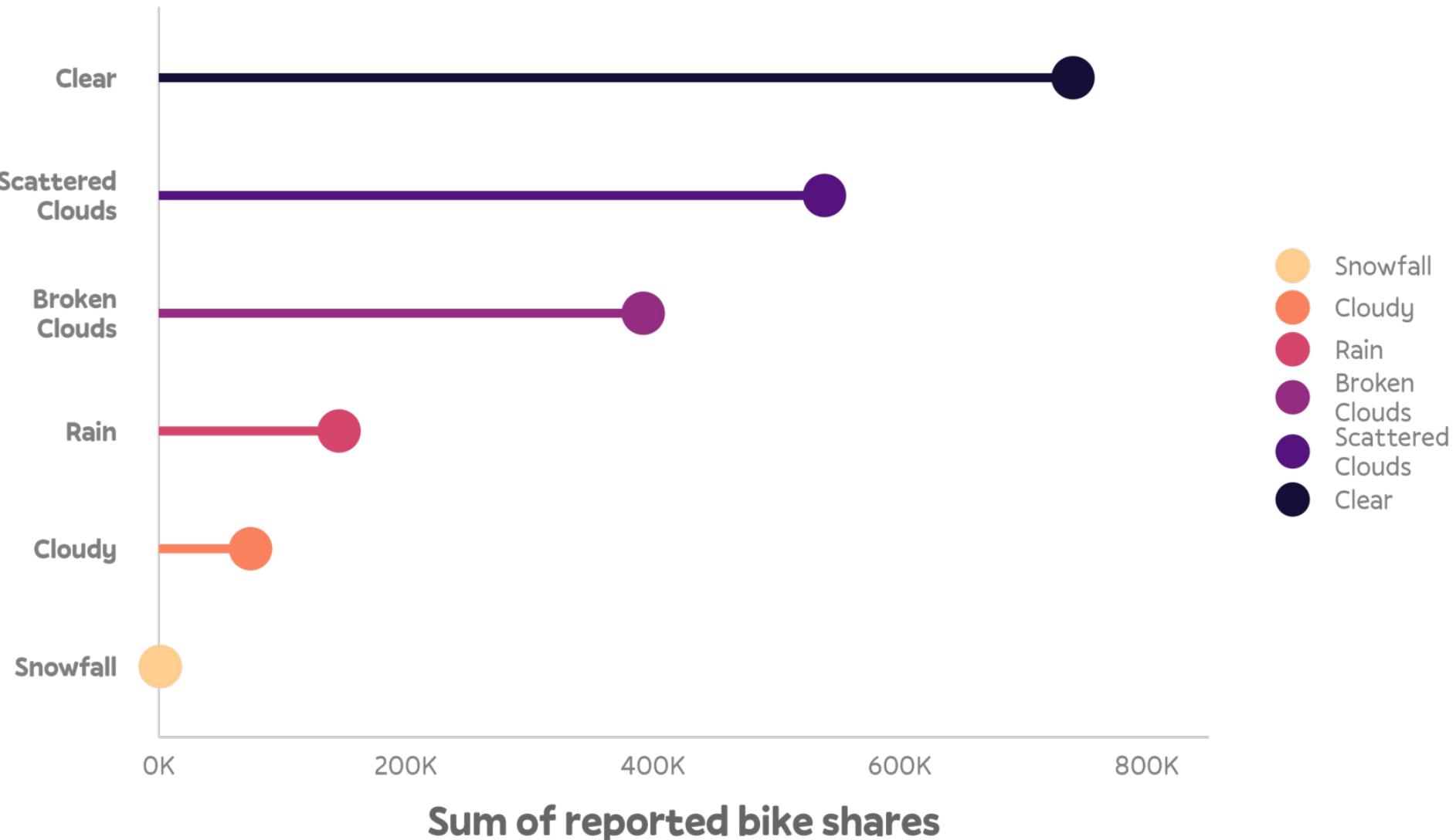
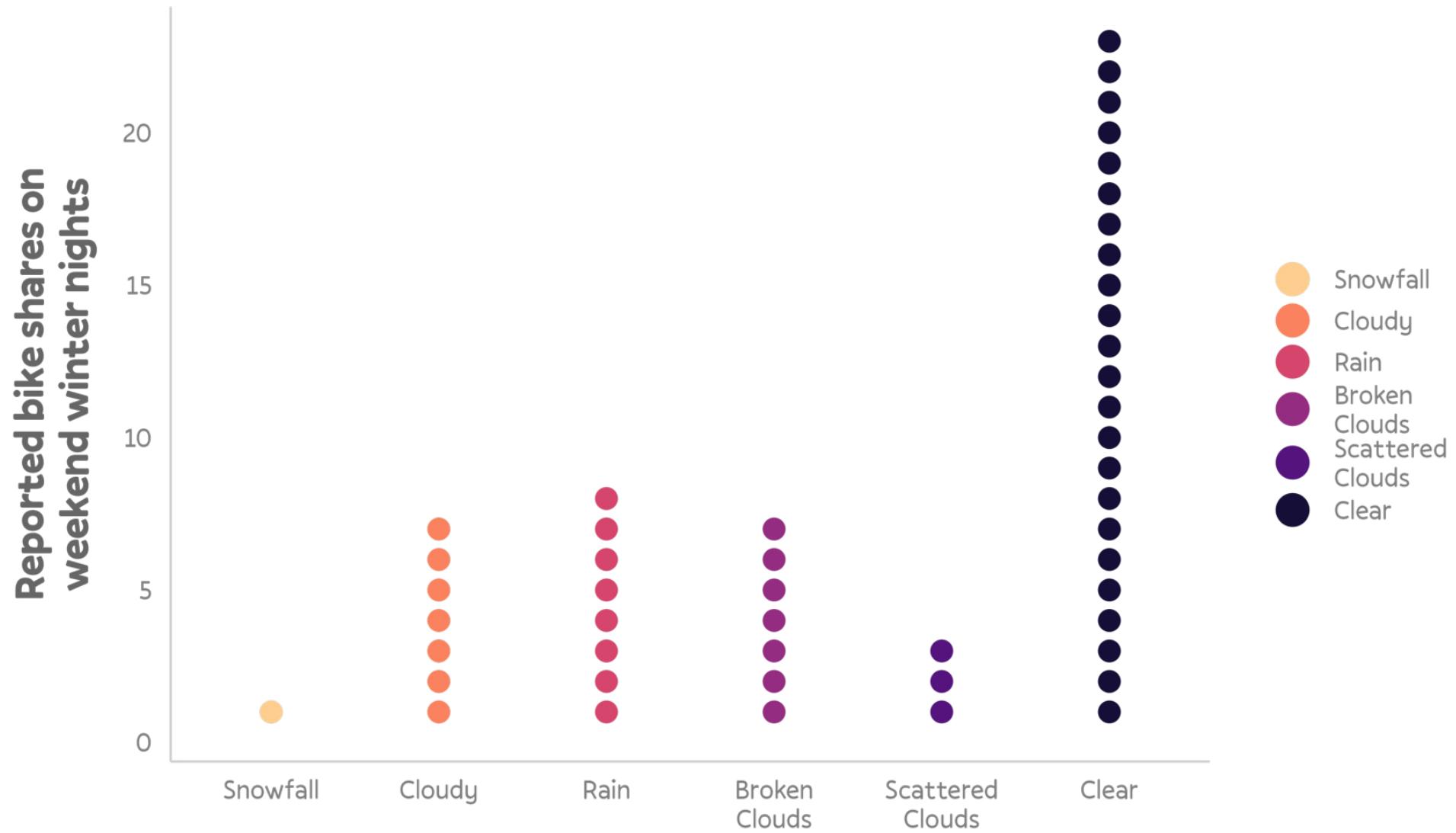


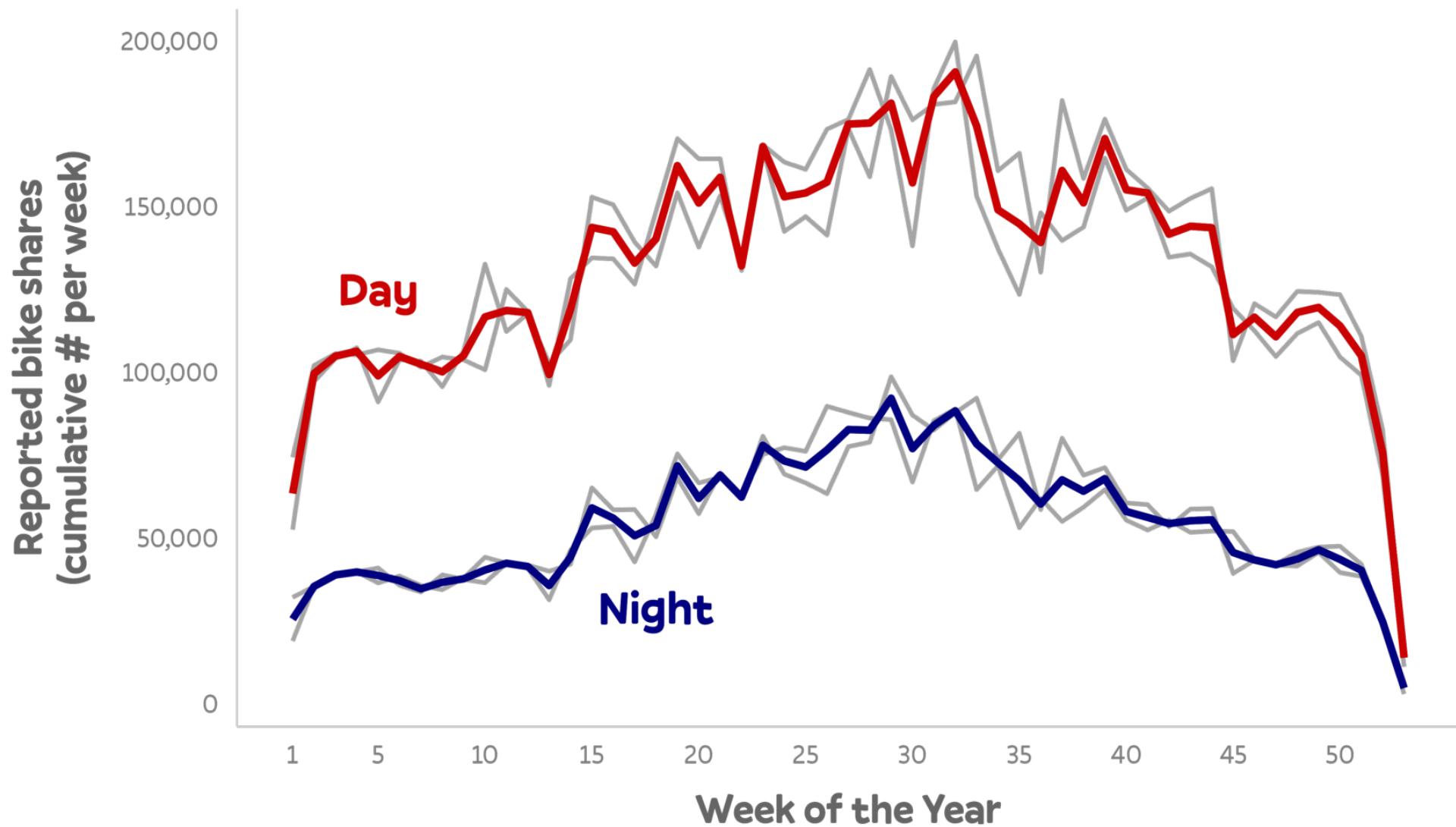
Illustration by Allison Horst

Cédric Scherer // R Course TU Dresden // Data Visualization with {ggplot2}



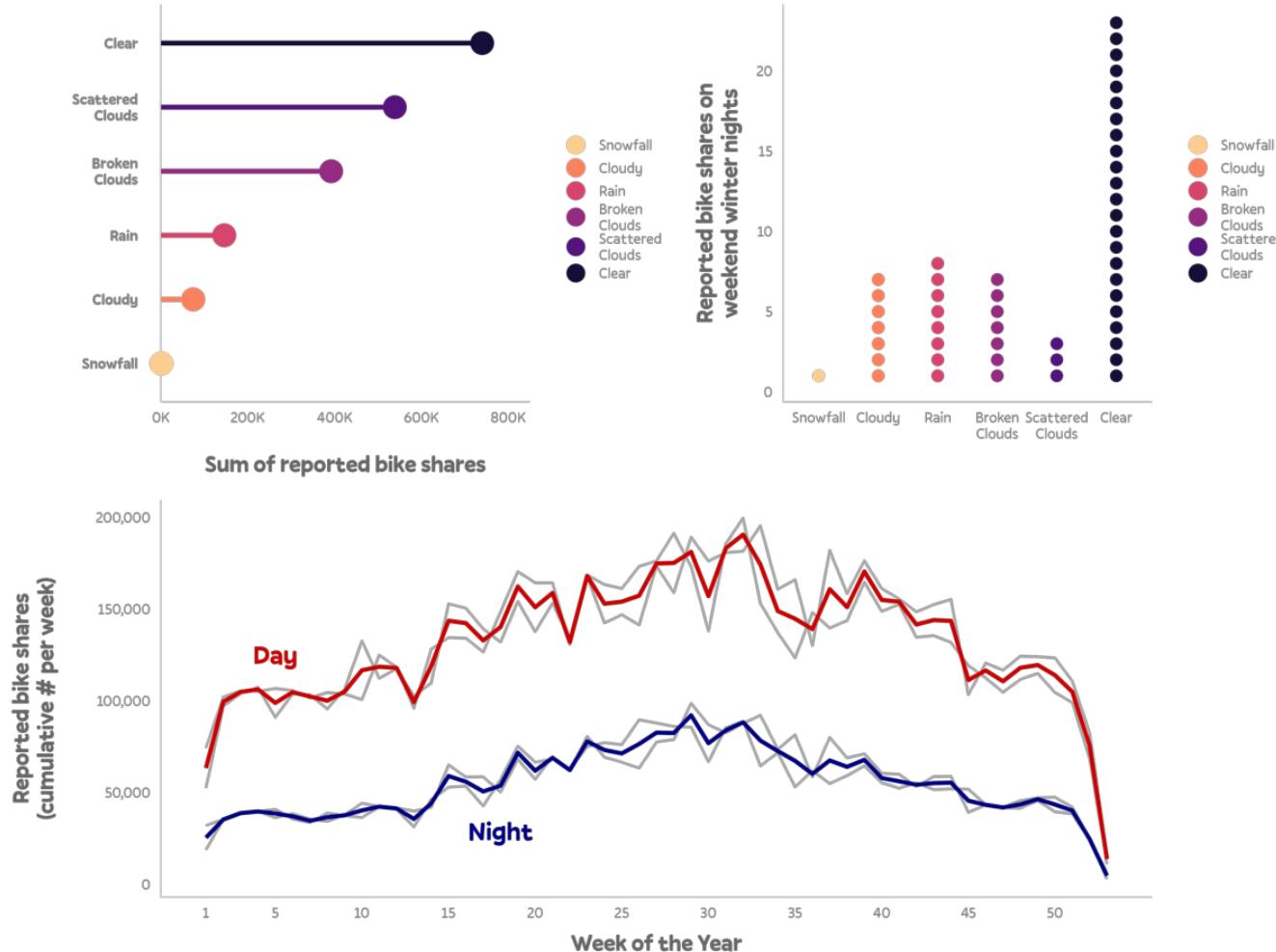






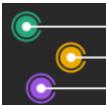
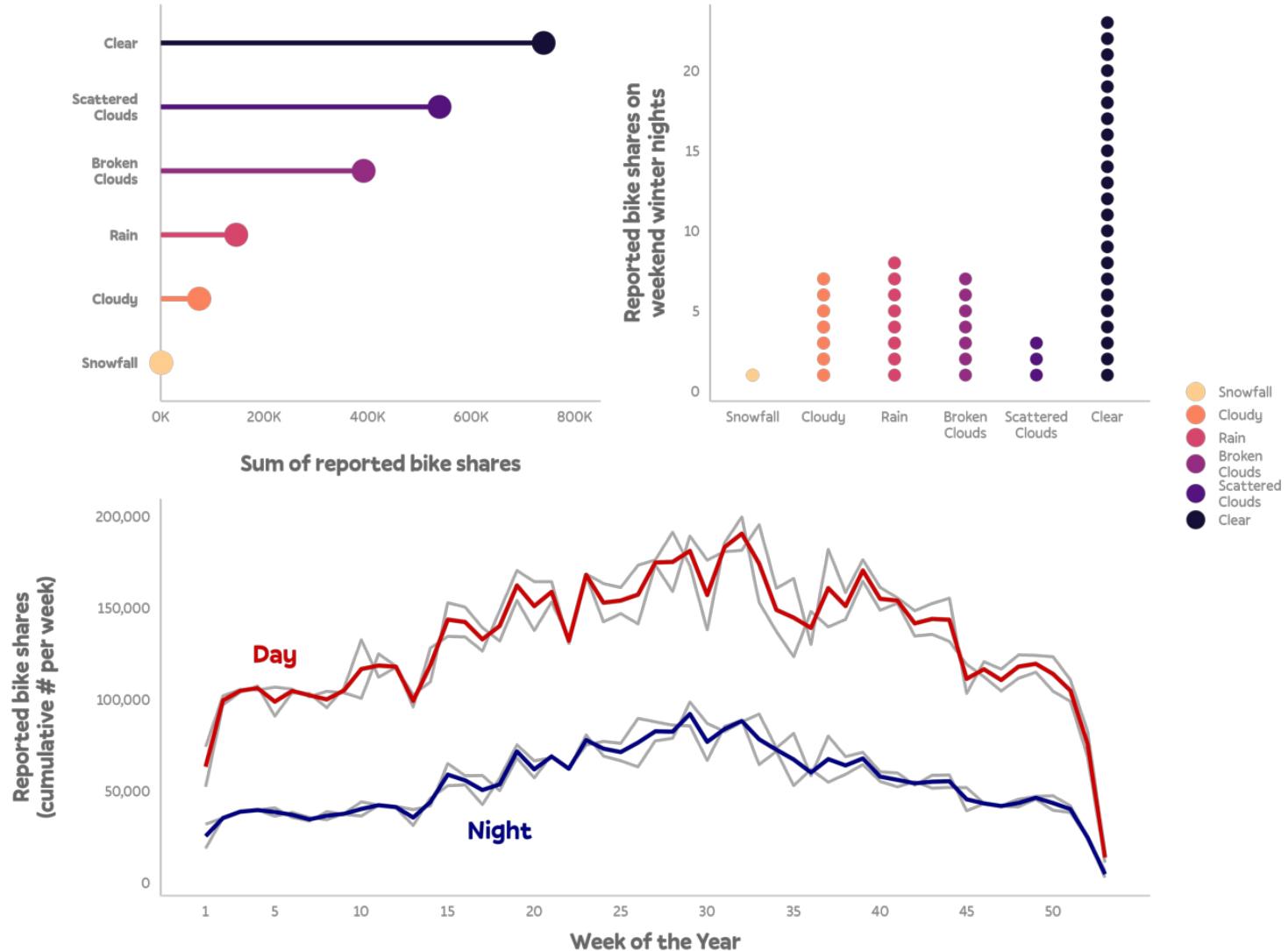
{patchwork}

```
1 # install.packages("patchwork")
2 library(patchwork)
3 (p1 + p2) / p3
```



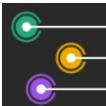
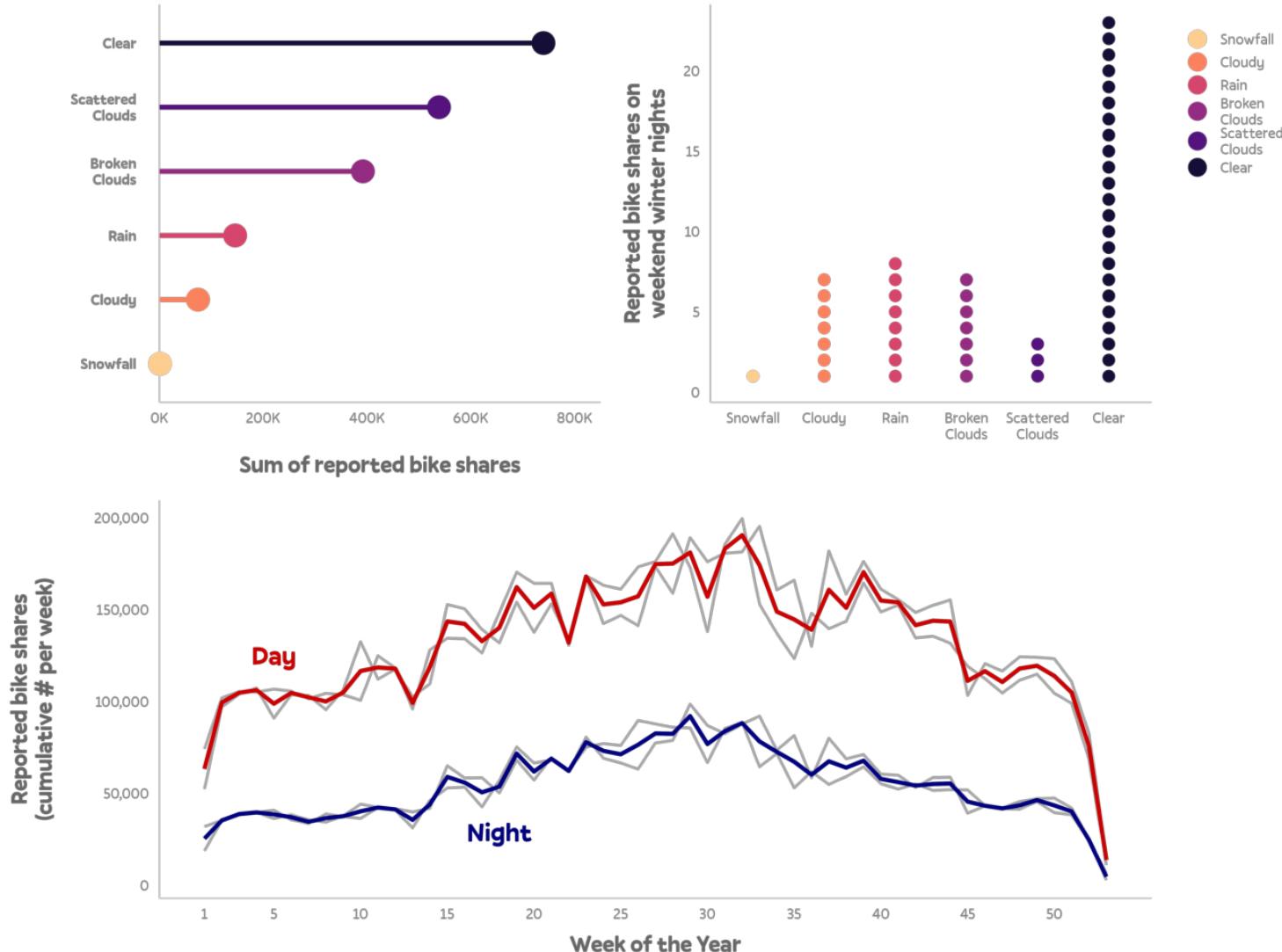
“Collect Guides”

```
1 (p1 + p2) / p3 + plot_layout(guides = "collect")
```



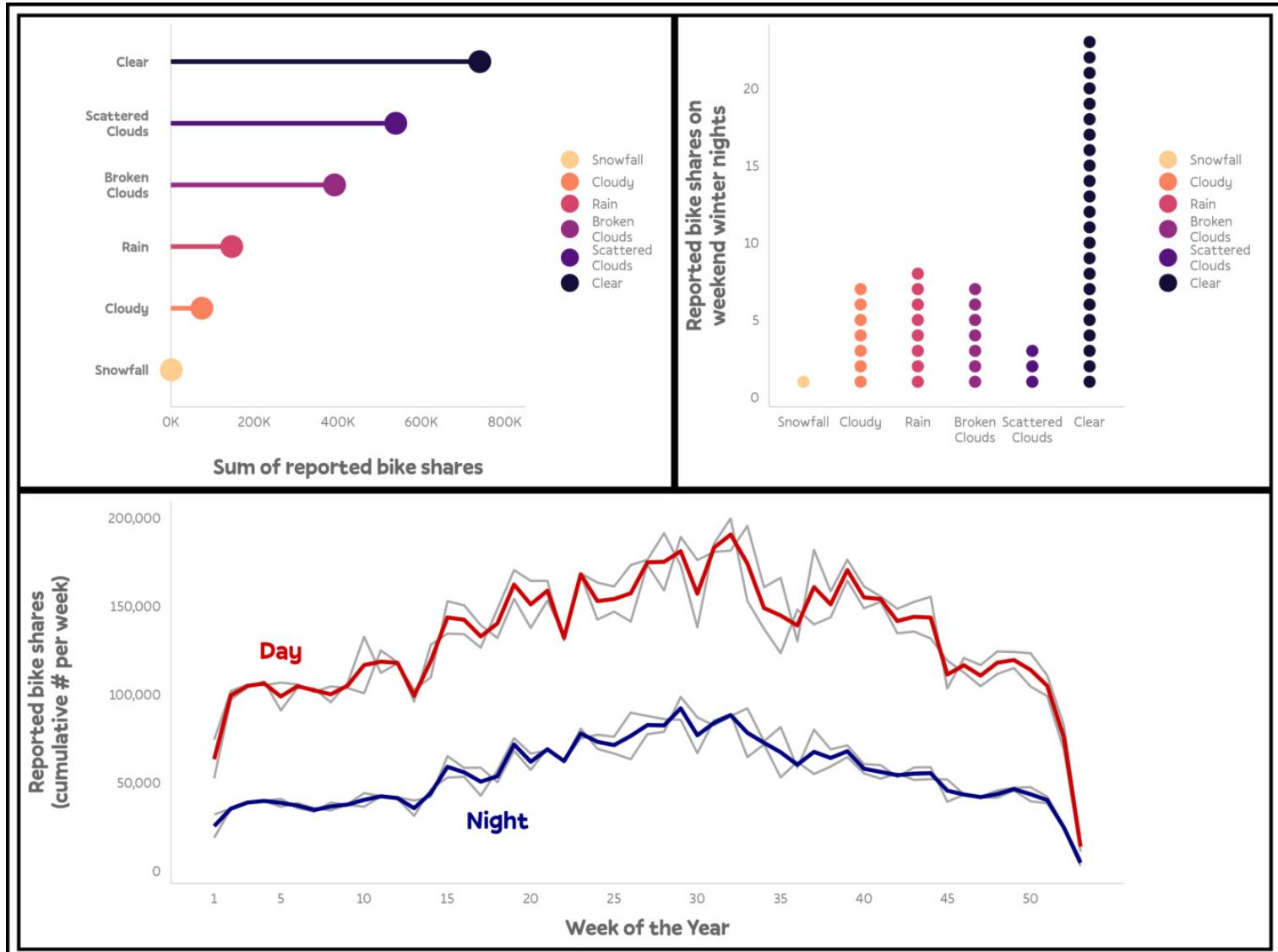
Apply Theming

```
1 ((p1 + p2) / p3 & theme(legend.justification = "top")) + plot_layout(guides = "collect")
```



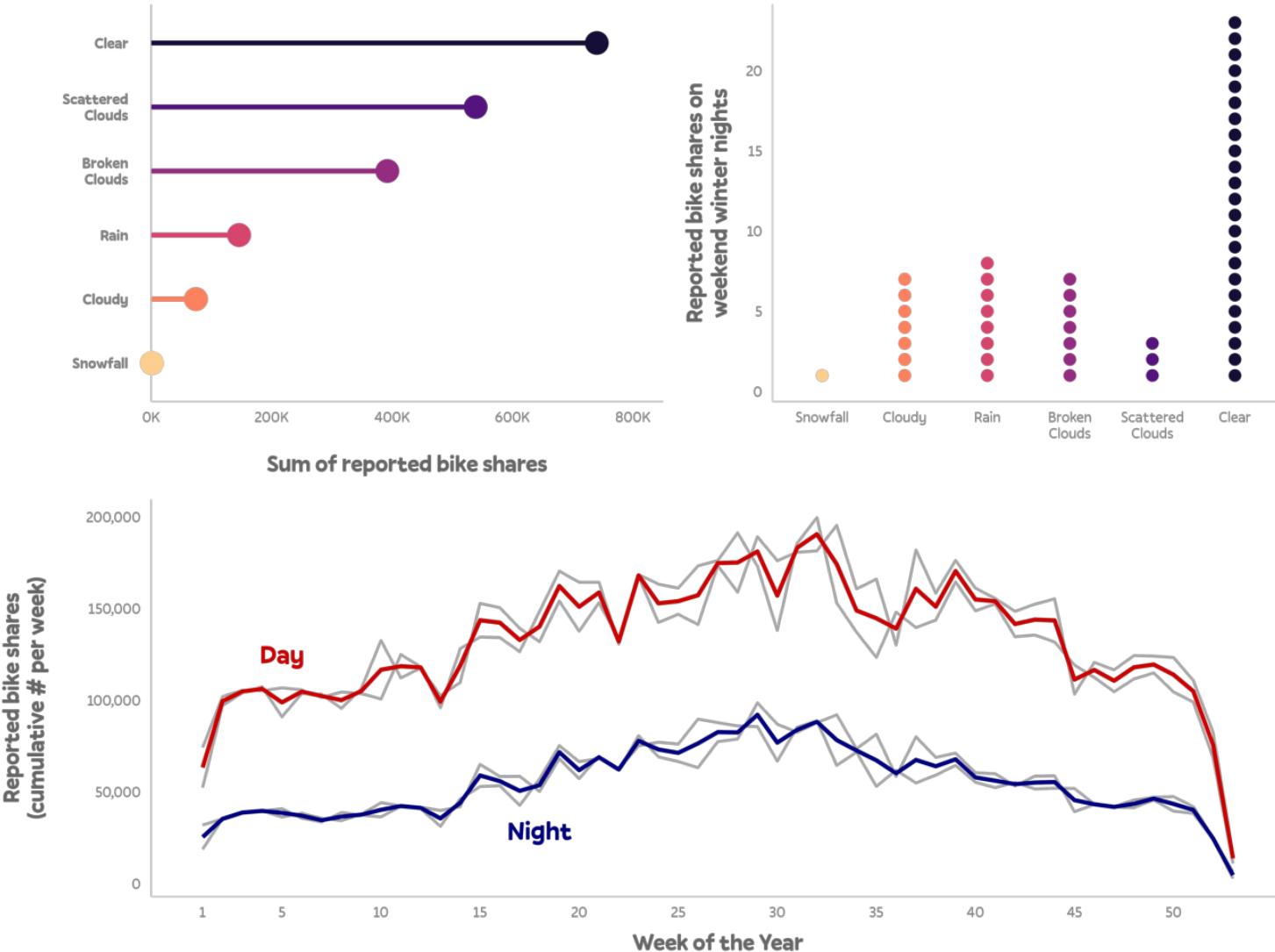
Apply Theming

```
1 (p1 + p2) / p3 & theme(plot.background = element_rect(color = "black", size = 3))
```



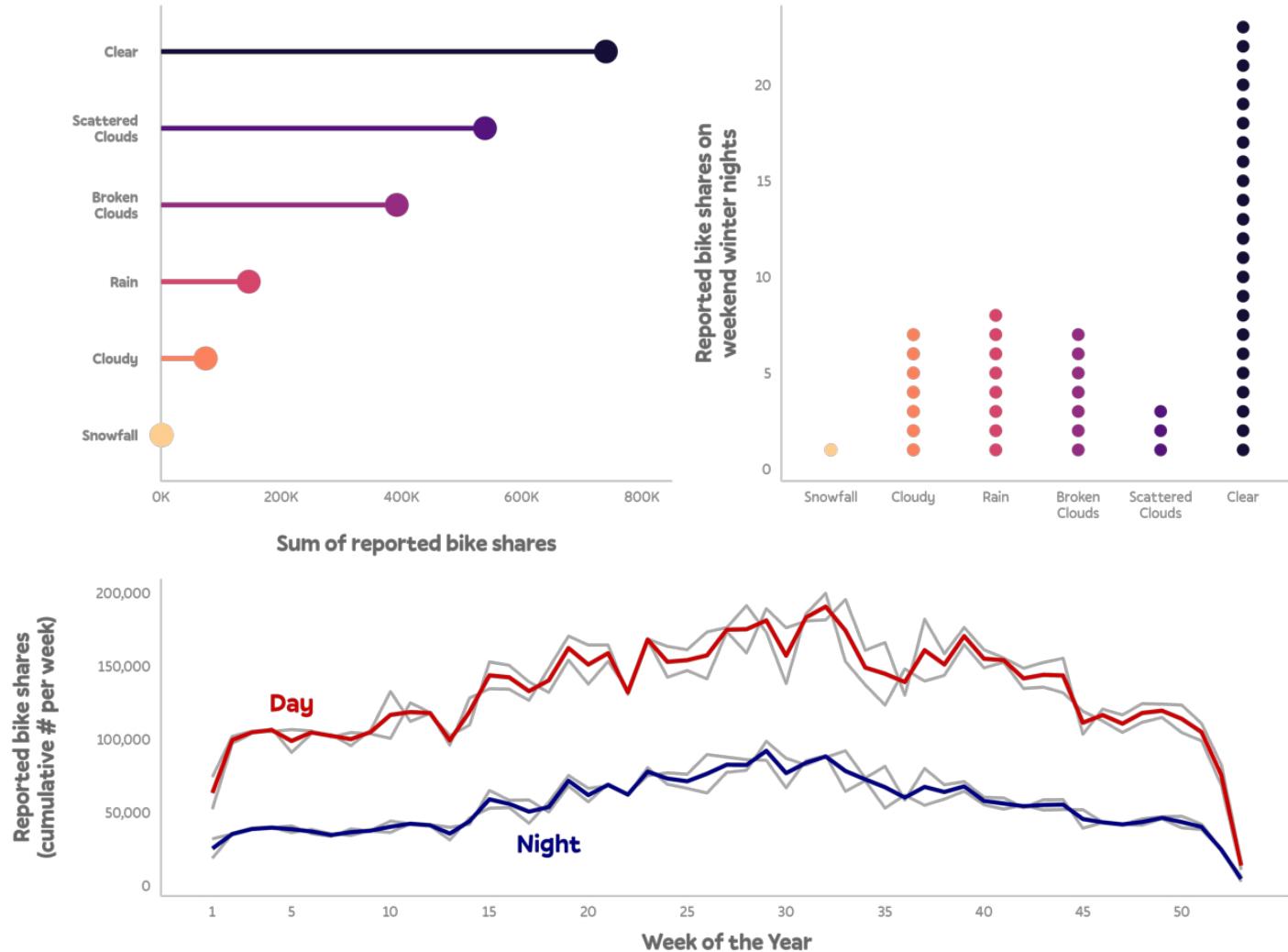
Apply Theming

```
1 (p1 + p2) / p3 & theme(legend.position = "none")
```



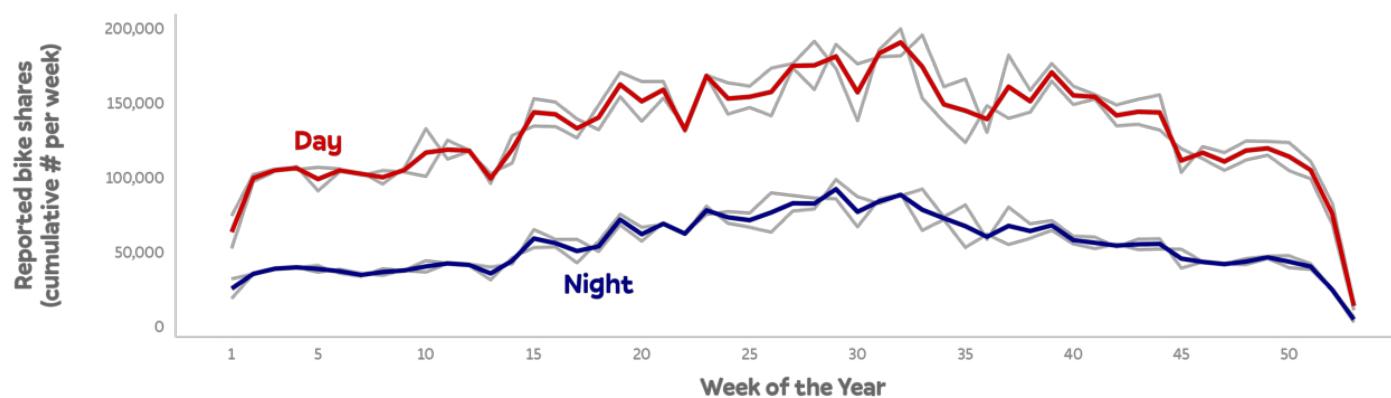
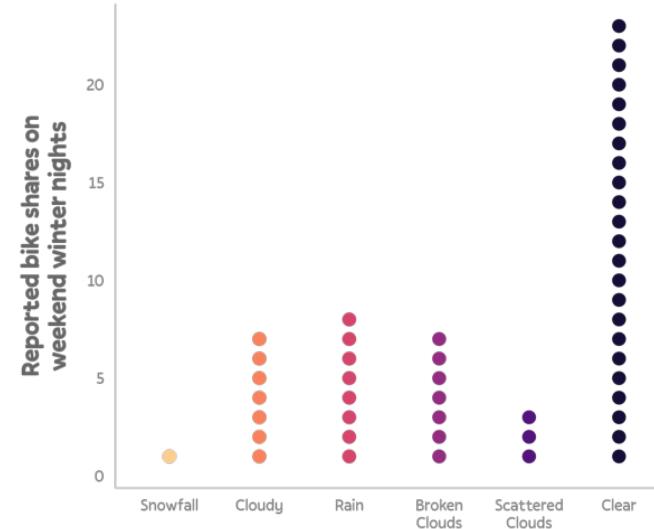
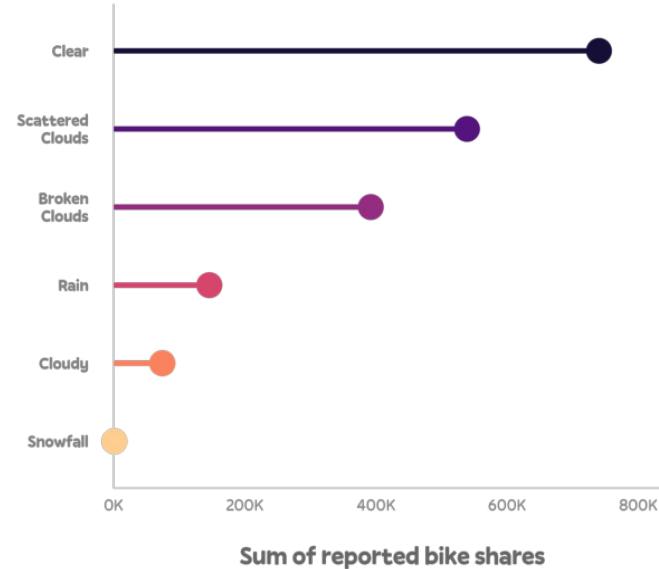
Adjust Widths and Heights

```
1 ((p1 + p2) / p3 & theme(legend.position = "none")) +  
2   plot_layout(heights = c(1.5, 1), widths = c(2, 1))
```



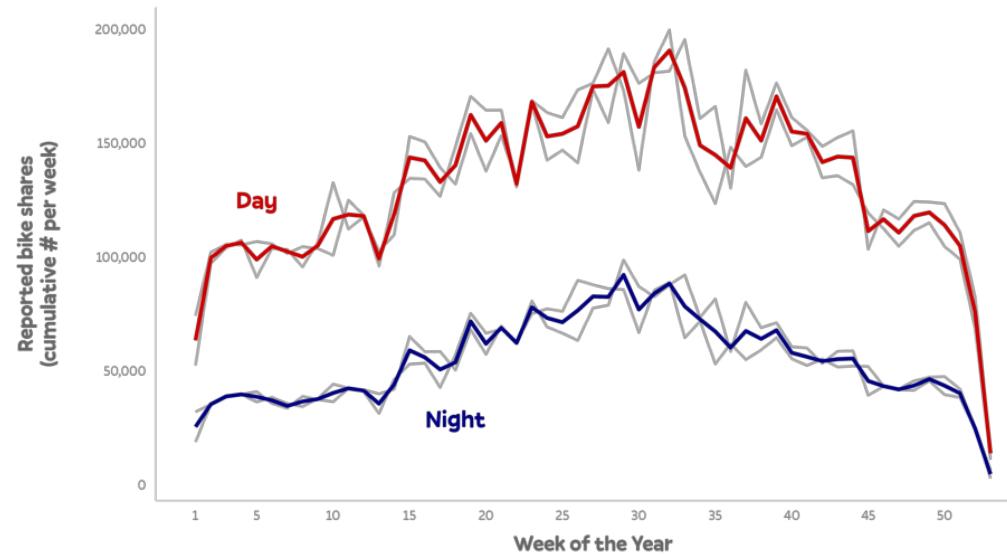
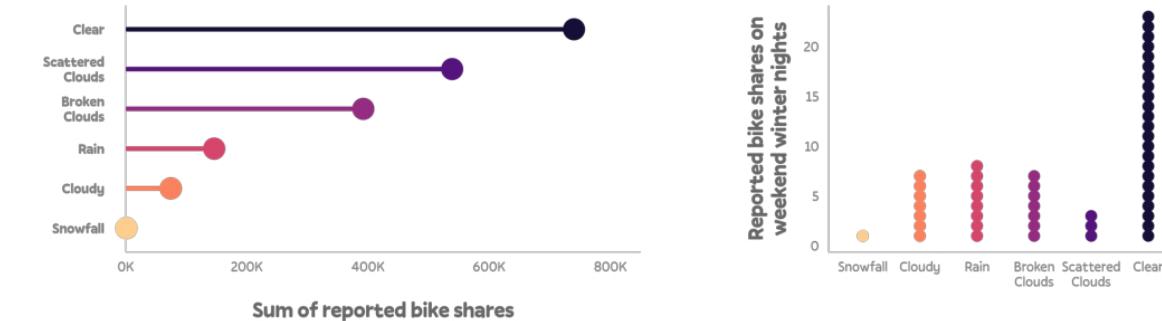
Add Some White Space

```
1 ((p1 + p2) / plot_spacer() / p3 & theme(legend.position = "none")) +  
2   plot_layout(heights = c(1.5, .1, 1), widths = c(2, 1))
```



Use A Custom Layout

```
1 picasso <- "  
2 #####BBBB  
3 #######  
4 #######"  
5 (p1 + p2 + p3 & theme(legend.position = "none")) + plot_layout(design = picasso)
```



Add Labels

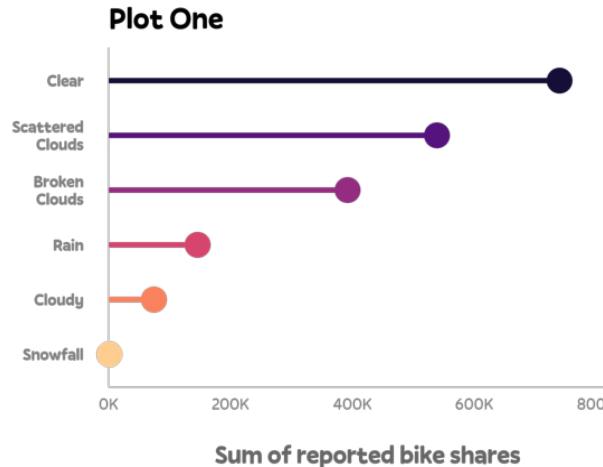
```
1 pt1 <- p1 + labs(title = "Plot One") + theme(legend.position = "none")
2 pt2 <- p2 + labs(title = "Plot Two") + theme(legend.position = "none")
3 pt3 <- p3 + labs(title = "Plot Three") + theme(legend.position = "none")
```



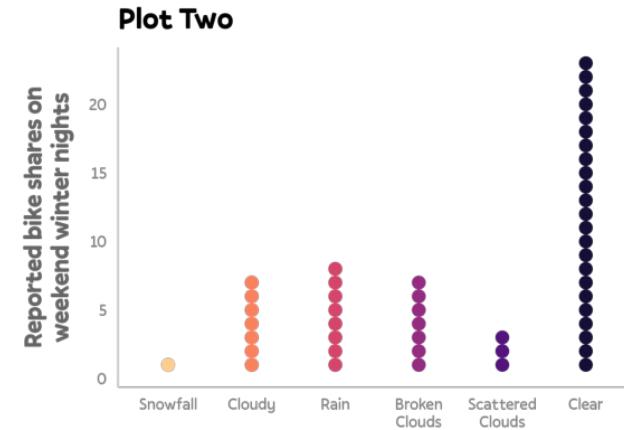
Add Labels

```
1 (pt1 + pt2) / pt3 +  
2 plot_annotation(tag_levels = "1", tag_prefix = "P")
```

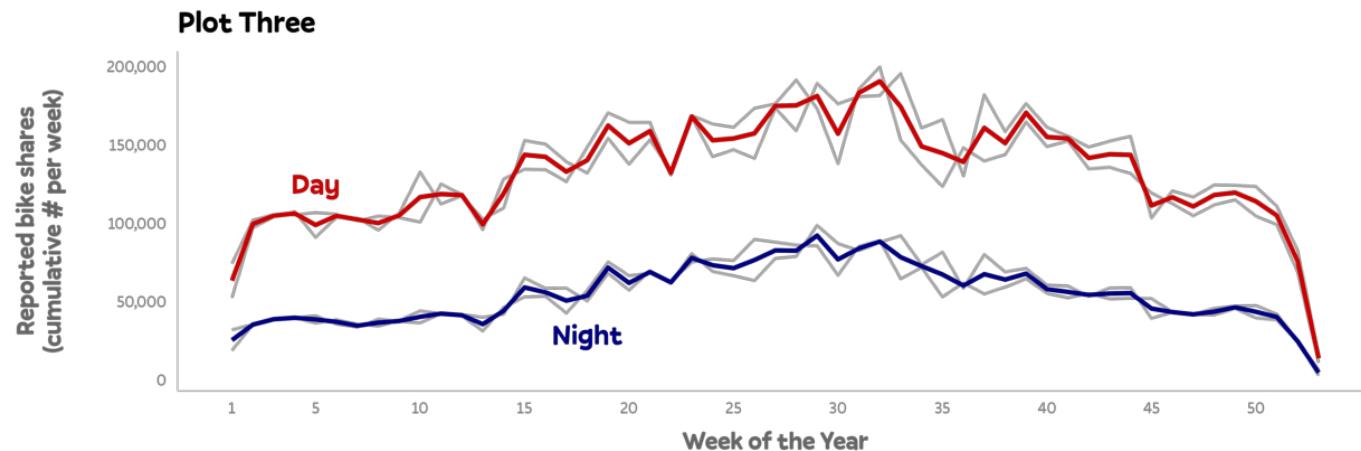
P1



P2



P3

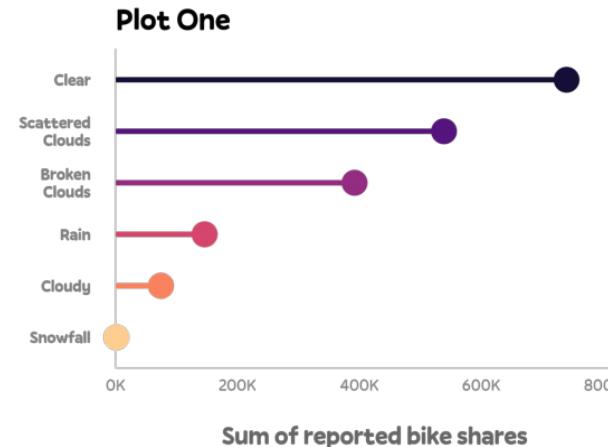


Add Labels

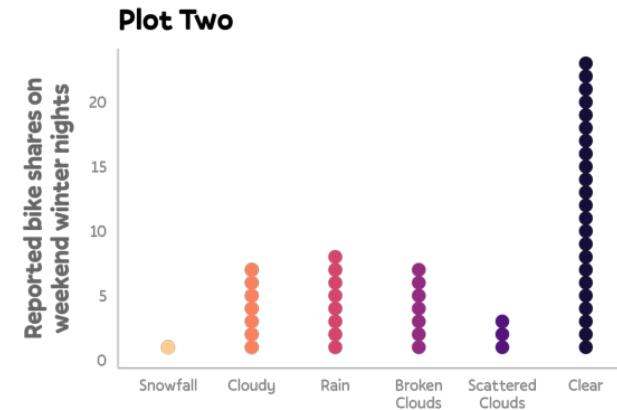
```
1 (pt1 + pt2) / pt3 +  
2 plot_annotation(tag_levels = "1", tag_prefix = "P", title = "An overarching title for all 3 plots")
```

An overarching title for all 3 plots, placed on the very top while all other titles are sitting below the tags.

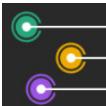
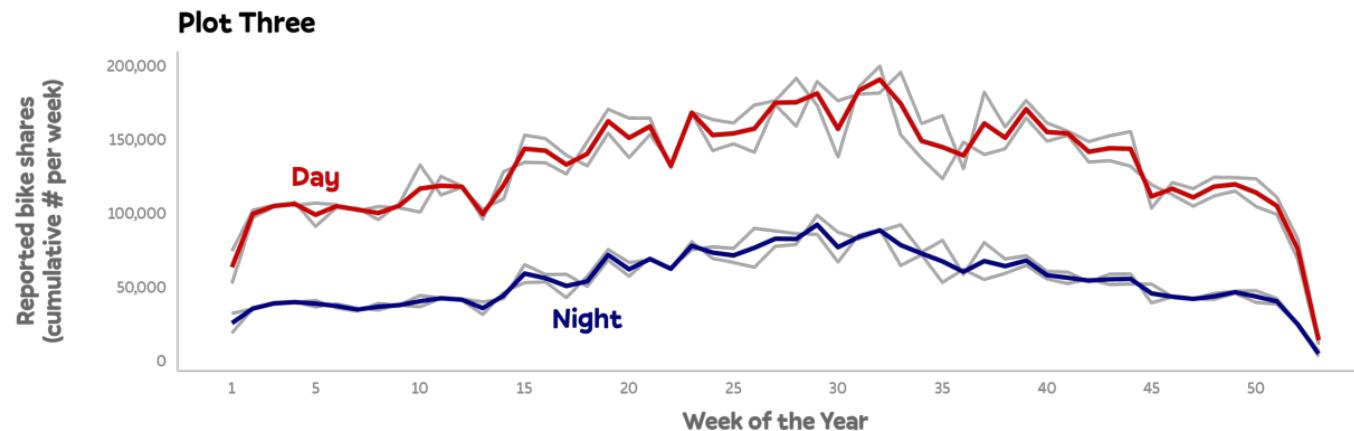
P1



P2

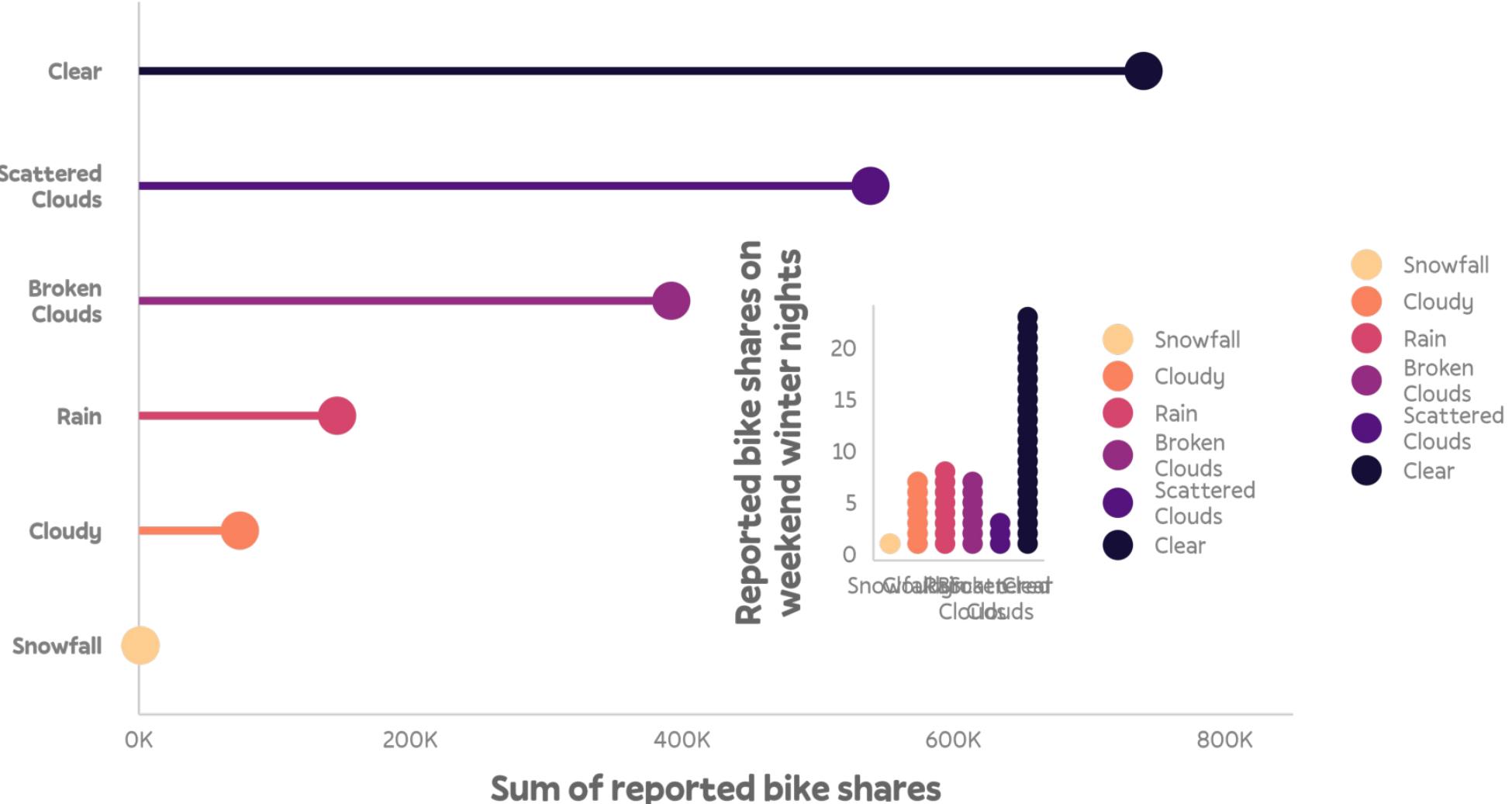


P3



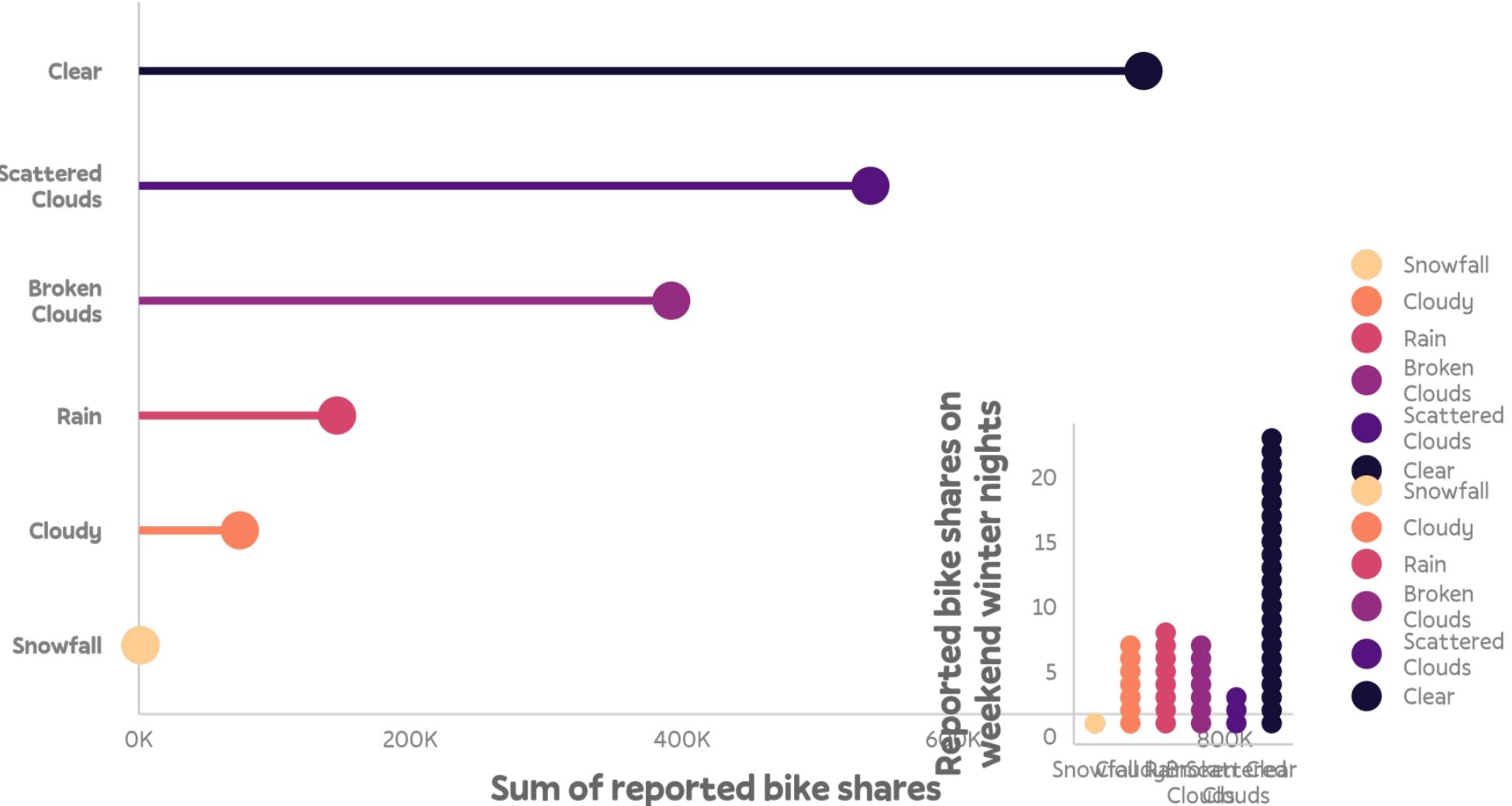
Add Inset Plots

```
1 p1 + inset_element(p2, l = .5, b = .1, r = 1, t = .6)
```



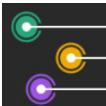
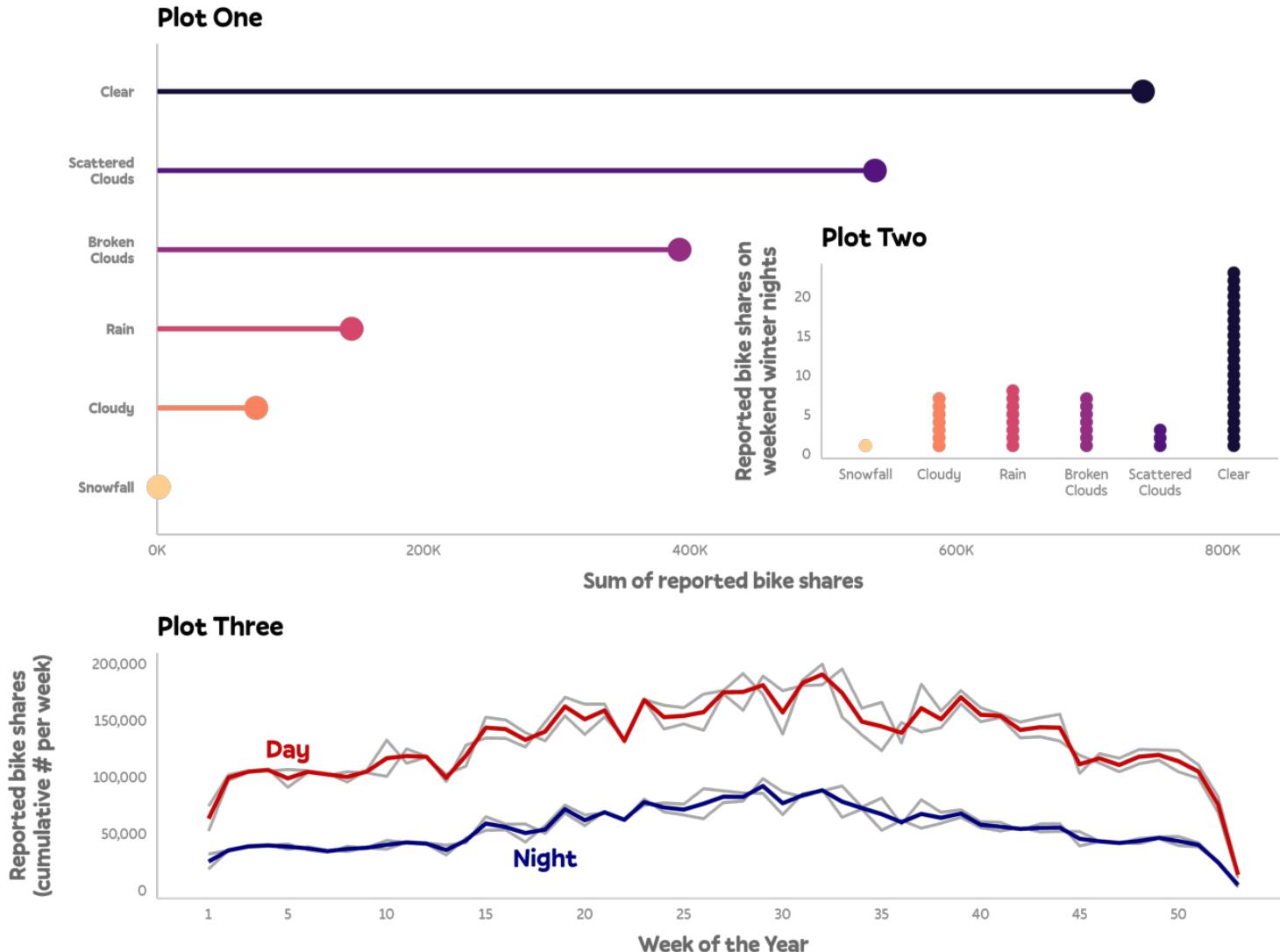
Add Inset Plots

```
1 p1 + inset_element(p2, l = .6, b = 0, r = 1, t = .5, align_to = 'full')
```



Add Inset Plots

```
1 (pt1 + inset_element(pt2, l = .5, b = .05, r = 1, t = .65)) / pt3 + plot_layout(heights = c(2, 1))
```



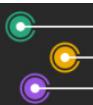
Exciting Extension Packages

- **{ggalt}** – alternative coords, geoms, stats & scales
- **{gganimate}** – create animations
- **{ggbump}** – geoms for bump charts
- **{ggforce}** – several interesting add-on features
- **{ggiraph}** – geoms for dynamic, interactive visualizations
- **{ggmaps}** – access to Google & Stamen maps
- **{ggplotly}** – create interactive plots
- **{ggraph}** – networks, graphs & trees
- **{ggrepel}** – prevent overlapping text labels
- **{ggridges}** – geoms for ridgeline plots
- **{ggsankey}** – geoms for sankey diagrams
- **{ggstream}** – geoms for stream graphs themes



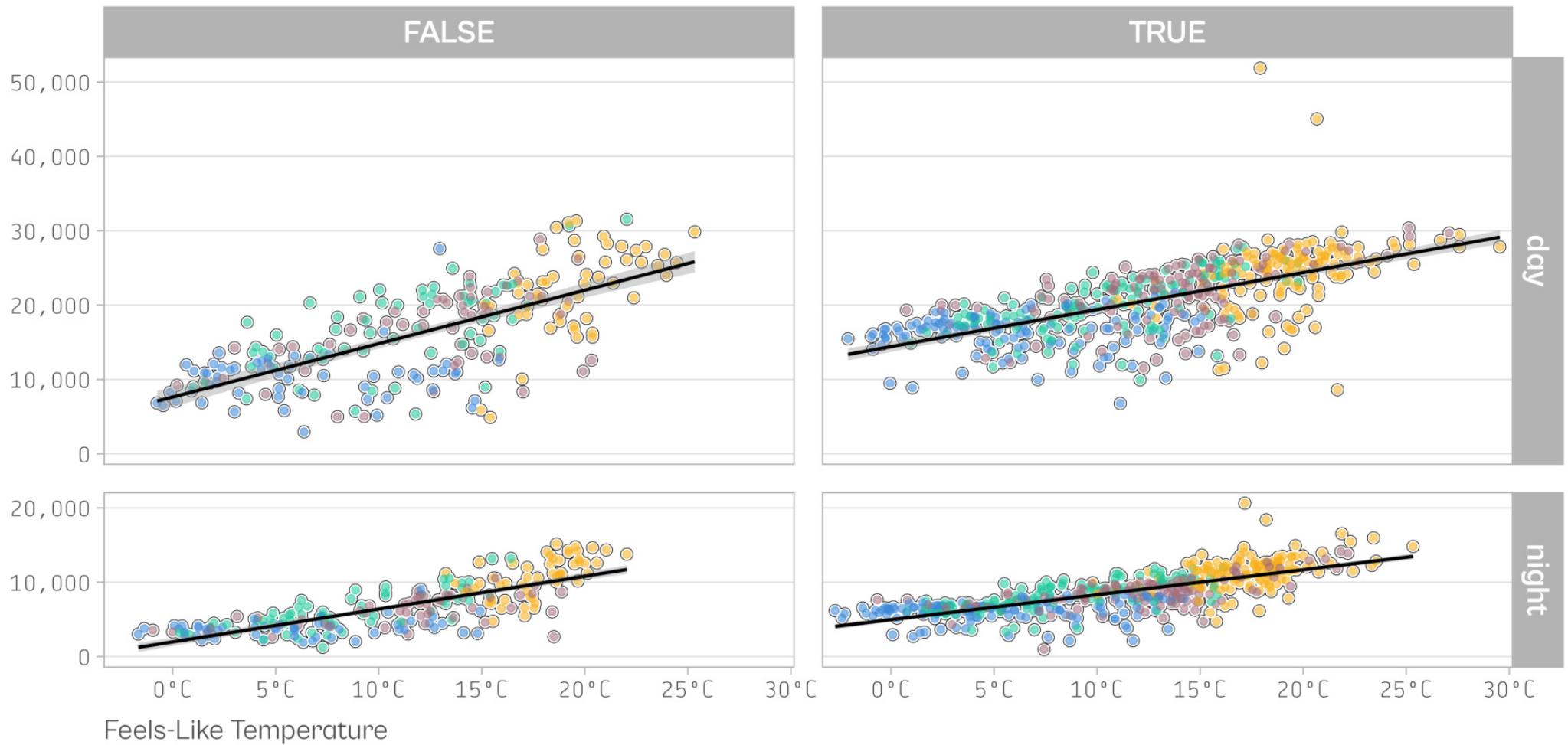
Data Visualization with {ggplot2}

— Wrap-Up —



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



```
1 library(dplyr)
2
3 bikes_adj <-
4   readr::read_csv(
5     "./data/london-bikes-custom.csv",
6     col_types = "Dcffffillllddddc"
7   ) %>%
8   mutate(
9     season = factor(season, levels = c("spring", "summer", "autumn", "winter")),
10    day_night = stringr::str_to_title(day_night),
11    is_workday = factor(is_workday, level = c(TRUE, FALSE),
12                          labels = c("Workday", "Weekend or Holiday"))
13  )
```



```
1 bikes_adj
```

```
# A tibble: 1,454 × 14
  date      day_night year month season count is_wo...¹ is_we...² is_ho...³ temp temp_...⁴ humid...⁵ wind_...⁶ weath...
  <date>    <chr>     <fct> <fct> <fct>  <int> <fct>   <lgl>  <lgl>   <dbl>  <dbl>   <dbl>  <dbl> <chr>
1 2015-01-04 Day       2015  1   winter  6830 Weeken... TRUE   FALSE  2.17  -0.75  95.2  10.4 broken...
2 2015-01-04 Night    2015  1   winter  2404 Weeken... TRUE   FALSE  2.79  2.04  93.4  4.58 clear
3 2015-01-05 Day       2015  1   winter 14763 Workday FALSE  FALSE  8.96  7.71  81.1  8.67 broken...
4 2015-01-05 Night    2015  1   winter  5609 Workday FALSE  FALSE  7.12  5.71  79.5  9.04 cloudy
5 2015-01-06 Day       2015  1   winter 14501 Workday FALSE  FALSE  9     6.46  80.2  19.2 broken...
6 2015-01-06 Night    2015  1   winter  6112 Workday FALSE  FALSE  6.71  4.21  77.6  12.8 clear
7 2015-01-07 Day       2015  1   winter 16358 Workday FALSE  FALSE  8.17  5.08  75.2  21.2 scatte...
8 2015-01-07 Night    2015  1   winter  4706 Workday FALSE  FALSE  6.68  3.86  81.3  18.1 clear
9 2015-01-08 Day       2015  1   winter  9971 Workday FALSE  FALSE  9.46  7.12  79.4  18.8 scatte...
10 2015-01-08 Night   2015  1   winter  5630 Workday FALSE  FALSE  10.0  8.46  79.2  22.2 clear
# ... with 1,444 more rows, and abbreviated variable names `¹is_workday, `²is_weekend, `³is_holiday, `⁴temp_feel,
#   `⁵humidity, `⁶wind_speed, `⁷weather_type
```



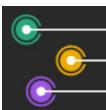
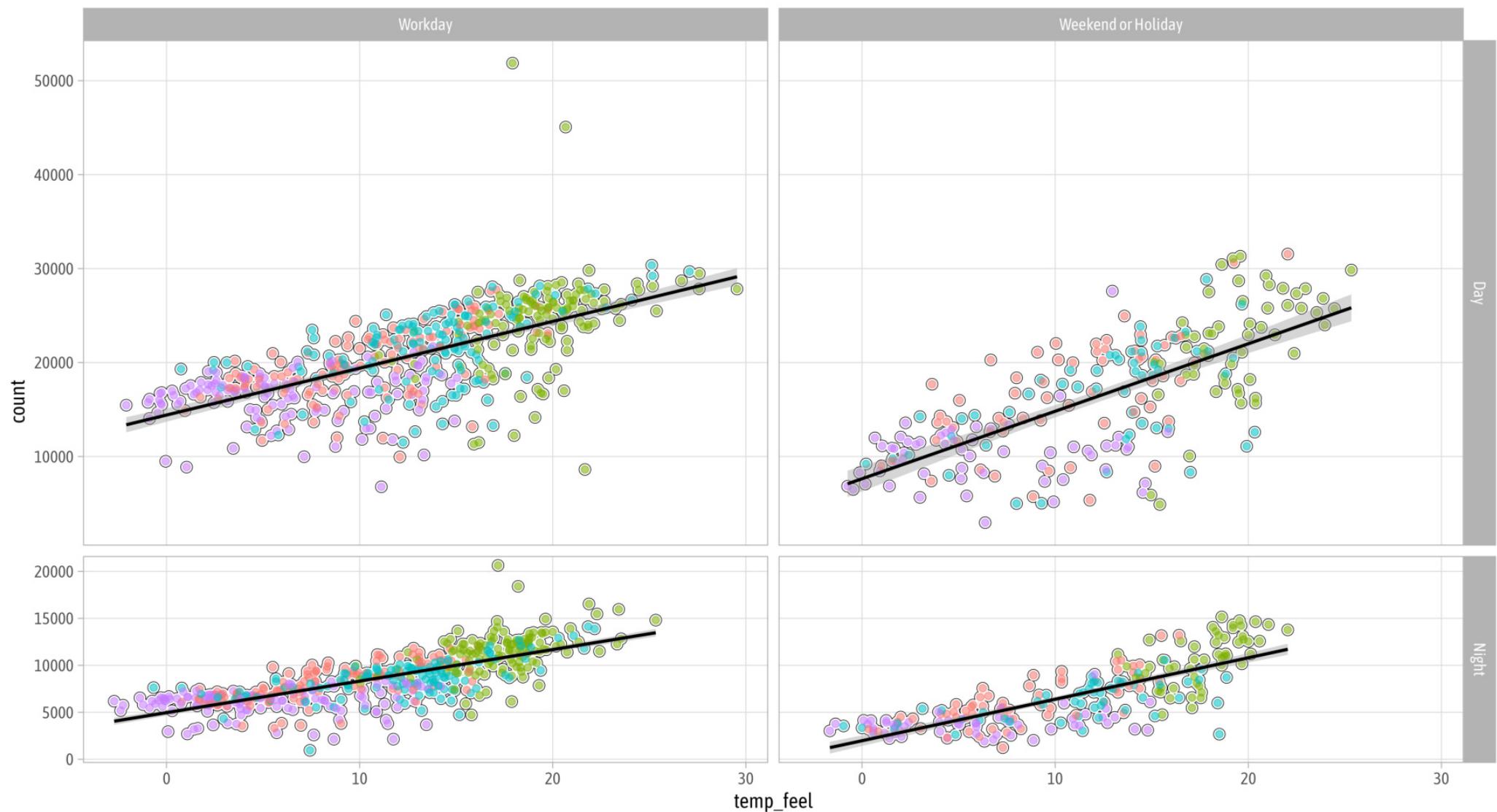
```
1 library(systemfonts)
2
3 register_variant(
4   name = "Cabinet Grotesk Regular S01",
5   family = "Cabinet Grotesk",
6   weight = "normal",
7   features = font_feature(letters = "stylistic")
8 )
9
10 register_variant(
11   name = "Cabinet Grotesk Bold S01",
12   family = "Cabinet Grotesk",
13   weight = "bold",
14   features = font_feature(letters = "stylistic")
15 )
16
17 register_variant(
18   name = "Cabinet Grotesk Black S01",
19   family = "Cabinet Grotesk",
```



```
1 g1 <-
2   ggplot(bikes_adj, aes(x = temp_feel, y = count)) +
3   ## point outline
4   geom_point(
5     color = "black", fill = "white",
6     shape = 21, stroke = .8, size = 2.8
7   ) +
8   ## opaque point background
9   geom_point(
10    color = "white", size = 2.8
11  ) +
12 ## colored, semi-transparent points
13 geom_point(
14   aes(color =forcats::fct_relabel(season, stringr::str_to_title)),
15   size = 2, alpha = .55
16 ) +
17 geom_smooth(
18   aes(group = day_night), method = "lm", color = "black"
19 ) +
```



forcats::fct_relabel(season, stringr::str_to_title) ● Spring ● Summer ● Autumn ● Winter



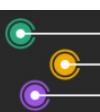
```
1 g2 <- g1 +
2   scale_x_continuous(
3     expand = c(mult = 0.02, add = 0),
4     breaks = 0:6*5,
5     labels = function(x) paste0(x, "°C")
6   ) +
7   scale_y_continuous(
8     expand = c(mult = 0, add = 1500),
9     limits = c(0, NA),
10    breaks = 0:5*10000,
11    labels = scales::label_comma()
12  ) +
13  scale_color_manual(
14    values = c("#3c89d9", "#1ec99b", "#F7B01B", "#a26e7c"),
15    guide = guide_legend(override.aes = list(size = 5))
16  ) +
17  labs(
18    x = "Feels-Like Temperature", y = NULL,
19    caption = "Data: Transport for London (TfL), Jan 2015–Dec 2016",
```



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



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

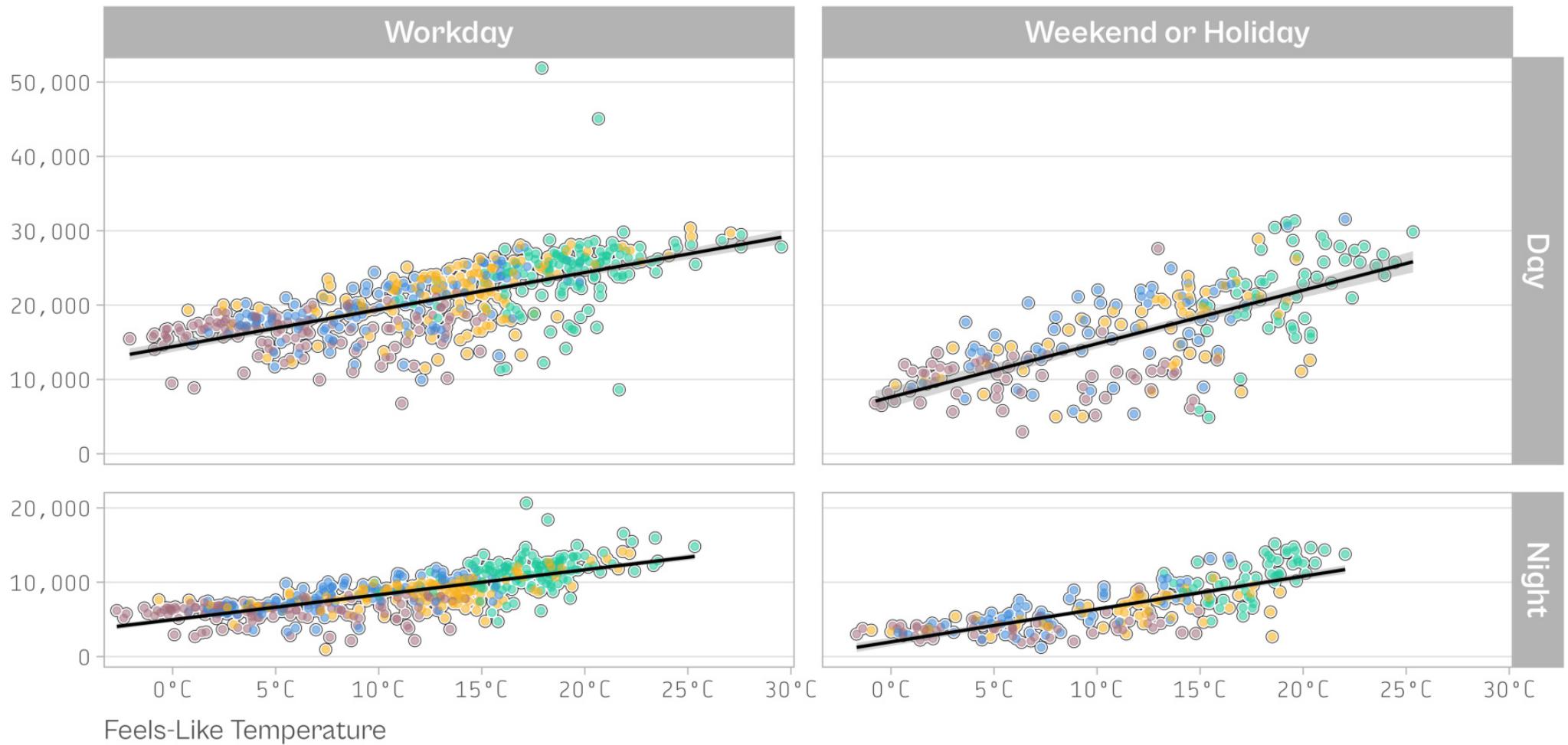


```
1 g3 <- g2 +
2   theme_light(
3     base_size = 18, base_family = "Cabinet Grotesk"
4   ) +
5   ## more theme adjustments
6   theme(
7     plot.title.position = "plot",
8     plot.caption.position = "plot",
9     plot.title = element_text(family = "Cabinet Grotesk Black S01", size = rel(1.6)),
10    axis.text = element_text(family = "Tabular"),
11    axis.title.x = element_text(hjust = 0, color = "grey30", margin = margin(t = 12)),
12    strip.text = element_text(family = "Cabinet Grotesk Bold S01", size = rel(1.15)),
13    panel.grid.major.x = element_blank(),
14    panel.grid.minor = element_blank(),
15    panel.spacing = unit(1.2, "lines"),
16    legend.position = "top",
17    legend.text = element_text(size = rel(1)),
18    ## for fitting my slide background
19    legend.key = element_rect(color = "transparent", fill = "transparent"),
```



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

● Spring ● Summer ● Autumn ● Winter



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

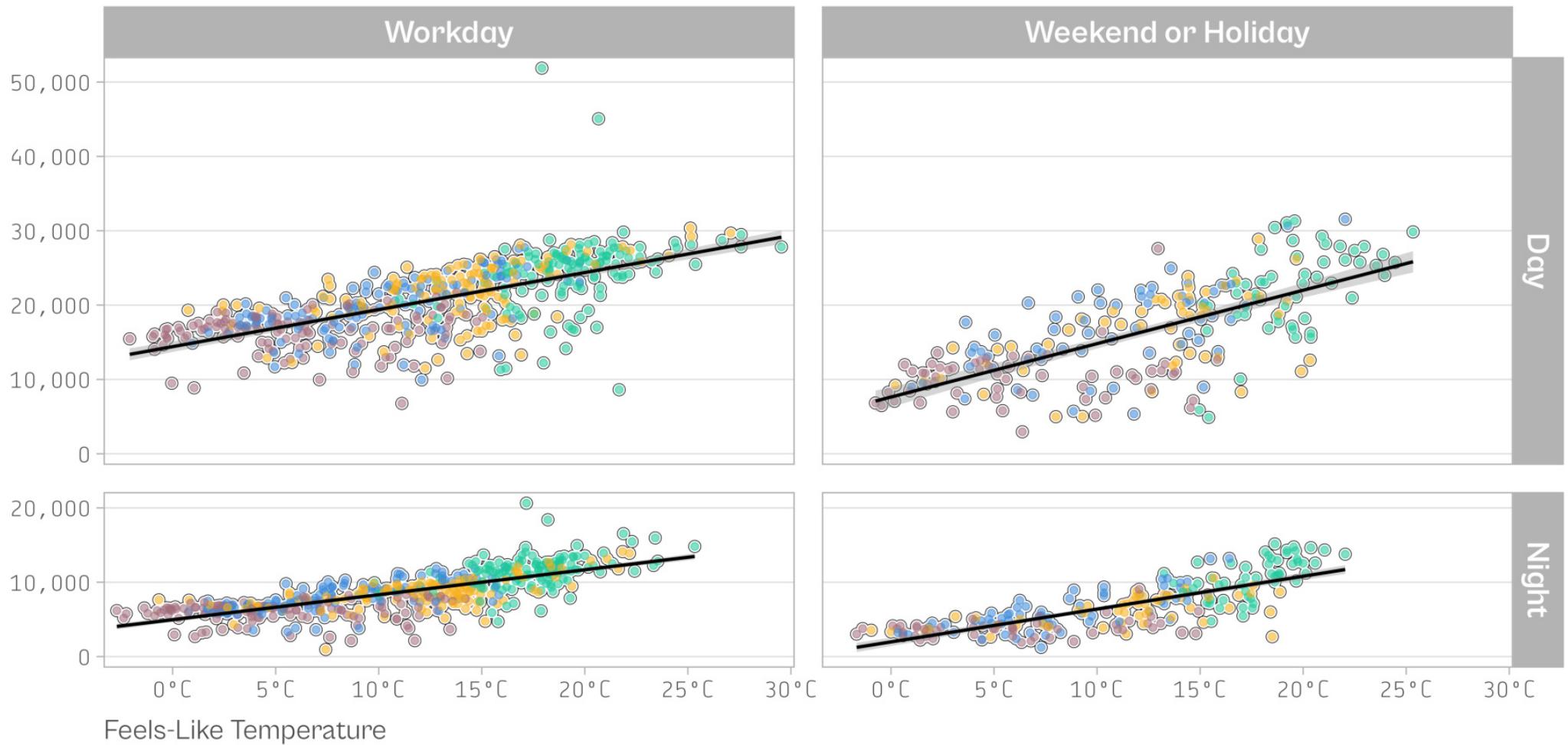


```
1 theme_set(  
2   theme_light(  
3     base_size = 18, base_family = "Cabinet Grotesk"  
4   )  
5 )  
6  
7 theme_update(  
8   plot.title.position = "plot",  
9   plot.caption.position = "plot",  
10  plot.title = element_text(family = "Cabinet Grotesk Black S01", size = rel(1.6)),  
11  axis.text = element_text(family = "Tabular"),  
12  axis.title.x = element_text(hjust = 0, color = "grey30", margin = margin(t = 12)),  
13  strip.text = element_text(family = "Cabinet Grotesk Bold S01", size = rel(1.15)),  
14  panel.grid.major.x = element_blank(),  
15  panel.grid.minor = element_blank(),  
16  panel.spacing = unit(1.2, "lines"),  
17  legend.position = "top",  
18  legend.text = element_text(size = rel(1)),  
19 ## for fitting my slide background
```



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

● Spring ● Summer ● Autumn ● Winter

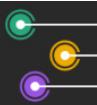


Data: Transport for London (TfL), Jan 2015–Dec 2016



Data Visualization with {ggplot2}

— Exercise —

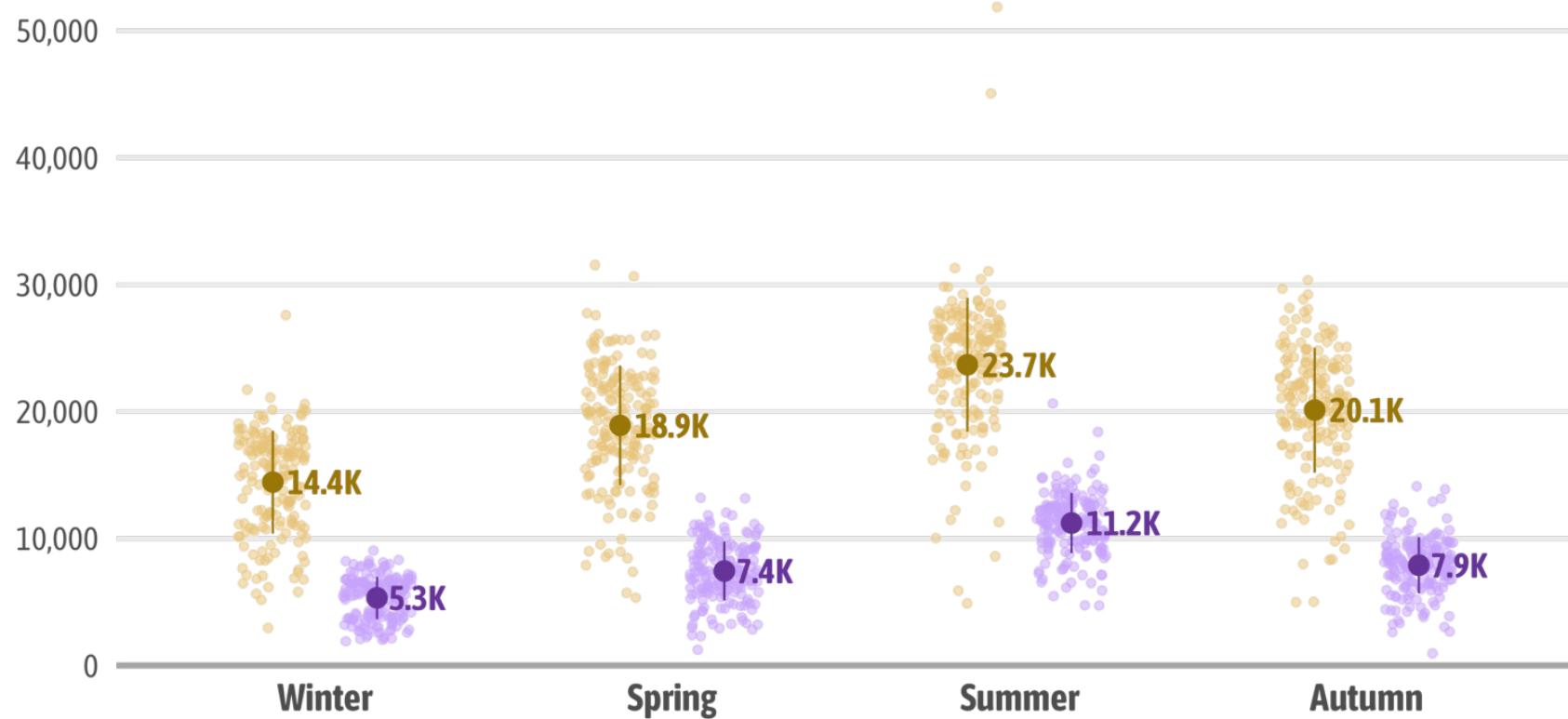


Your Turn: An Advanced ggplot

- Create a similar visualization as close as possible:

Reported bike shares in London during day and night times

TfL bike sharing data from 2015 to 2016 per season and time of day.
Errorbars show the mean \pm standard deviation.

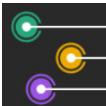
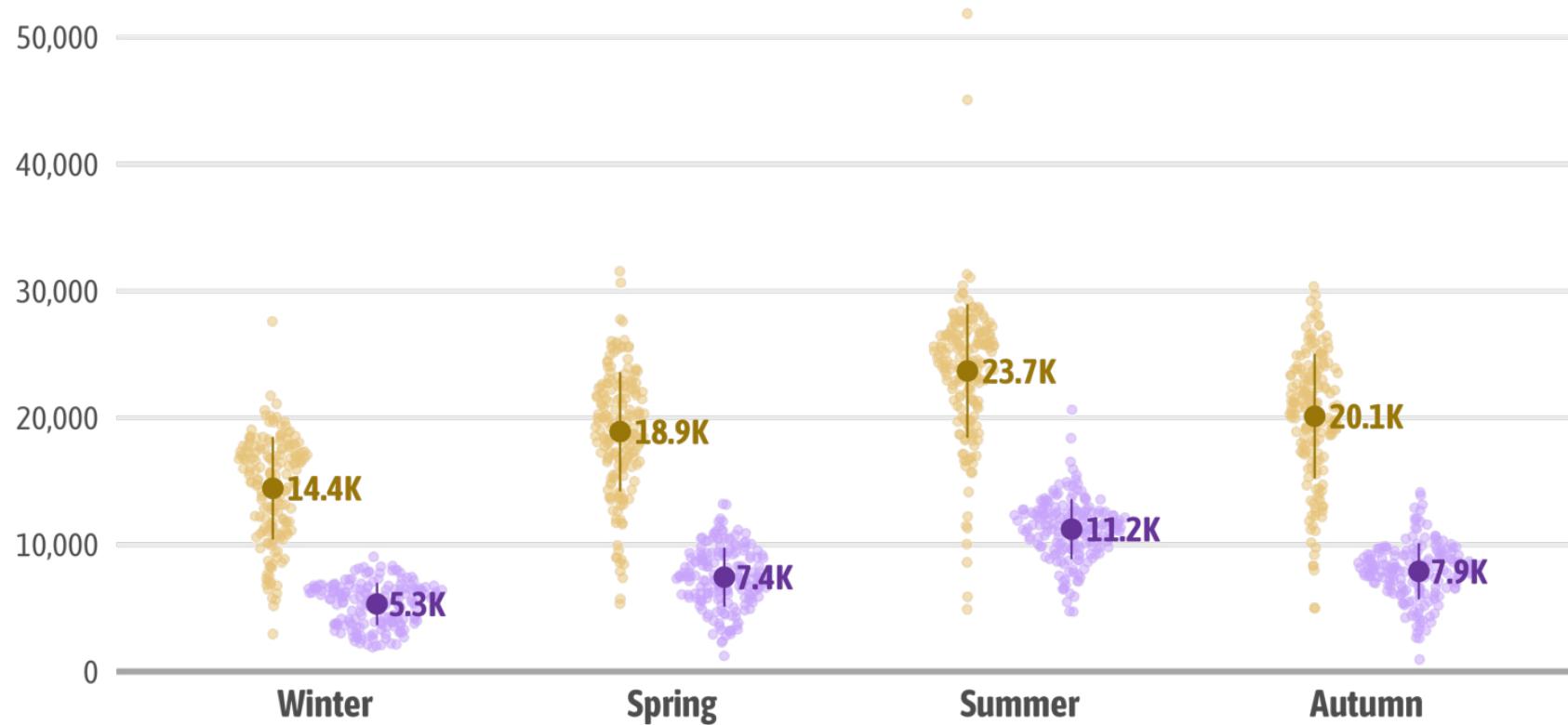


Your Turn: An Advanced ggplot

- Create a similar visualization as close as possible:

Reported bike shares in London during day and night times

TfL bike sharing data from 2015 to 2016 per season and time of day.
Errorbars show the mean \pm standard deviation.



Create Jitter Plot

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5 geom_point(  
6   aes(color = day_night),  
7   position = position_jitterdodge(  
8     jitter.width = .2, dodge.width = .6, seed = 1  
9   ),  
10  alpha = .5  
11 ) +  
12 theme_minimal(  
13   base_size = 18,  
14   base_family = "Asap Condensed"  
15 )
```



Create Sina Plot

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5   ggforce::geom_sina(  
6   aes(color = day_night),  
7   position = position_dodge(width = .6),  
8   alpha = .5  
9 ) +  
10 theme_minimal(  
11   base_size = 18,  
12   base_family = "Asap Condensed"  
13 )
```



Create Sina Plot

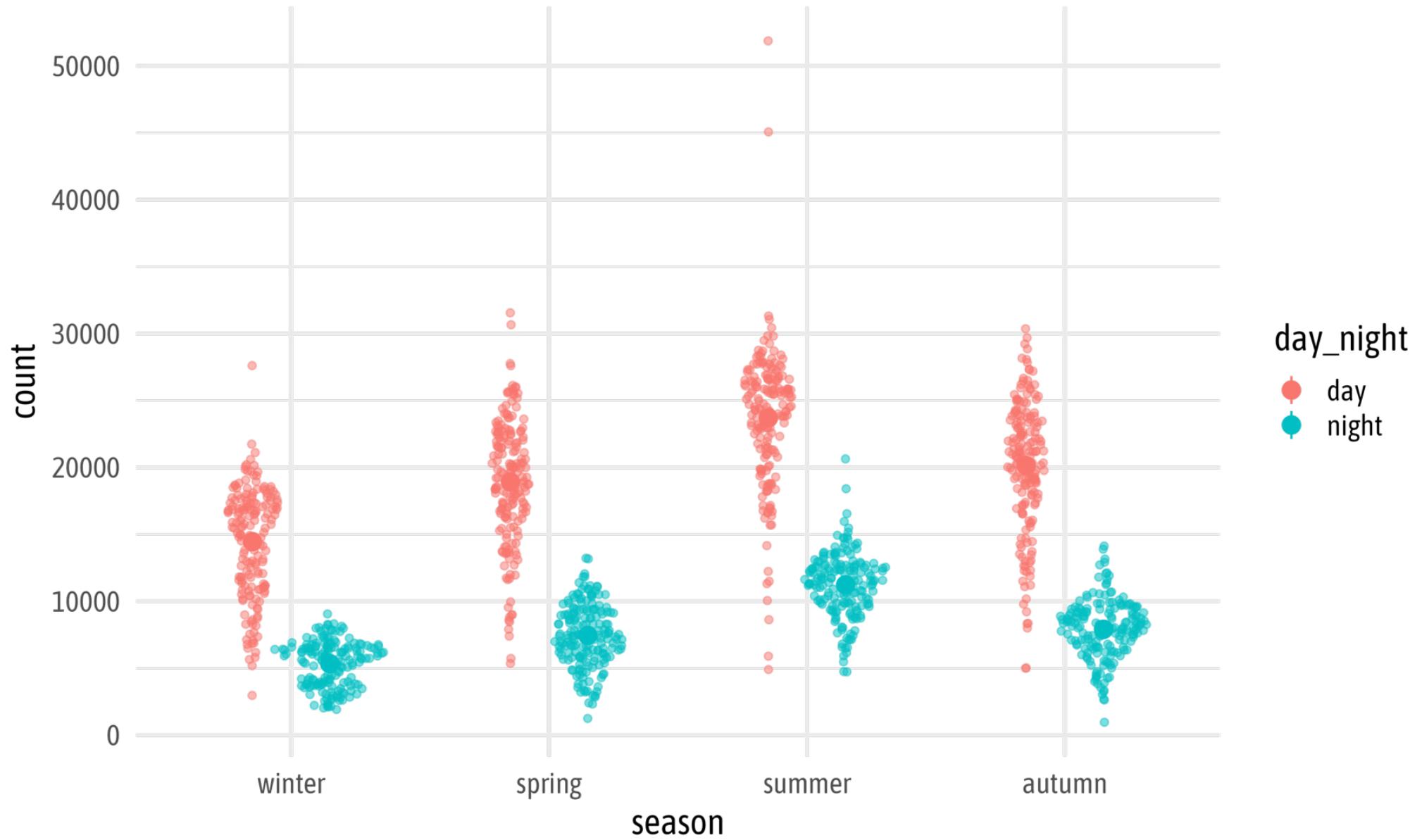


Add Errorbars

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5   ggforce::geom_sina(  
6   aes(color = day_night),  
7   position = position_dodge(width = .6),  
8   alpha = .5  
9 ) +  
10 stat_summary(  
11   aes(color = day_night),  
12   position = position_dodge(width = .6),  
13   size = .8  
14 ) +  
15 theme_minimal(  
16   base_size = 18,  
17   base_family = "Asap Condensed"  
18 )
```



Add Errorbars

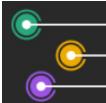
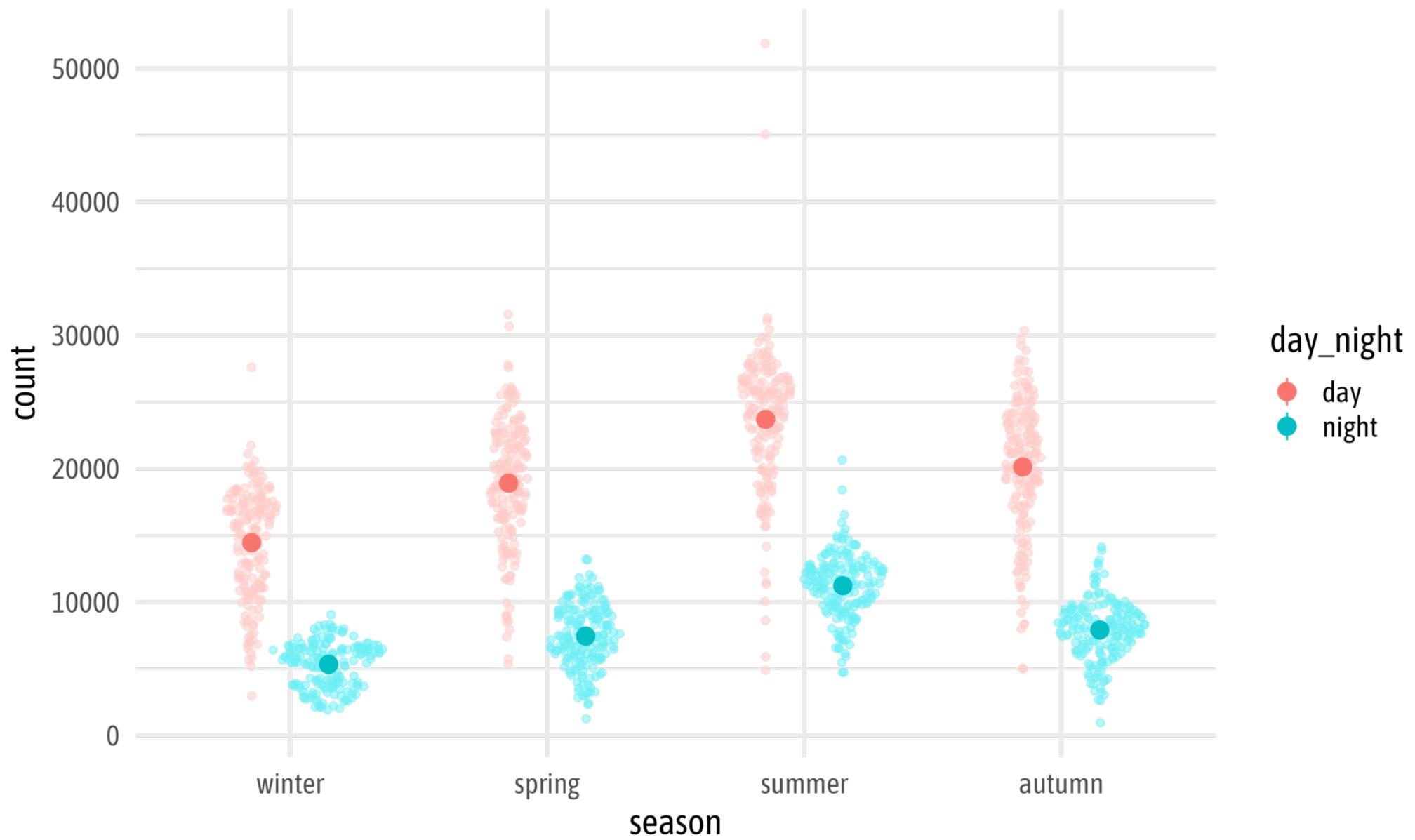


Use Lighter Point Colors

```
1 ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5   ggforce::geom_sina(  
6   aes(color = stage(  
7     day_night,  
8     after_scale = clr_lighten(color, .6)  
9   )),  
10  position = position_dodge(width = .6),  
11  alpha = .5  
12 ) +  
13 stat_summary(  
14   aes(color = day_night),  
15   position = position_dodge(width = .6),  
16   size = .8  
17 ) +  
18 theme_minimal(  
19   base_size = 18,
```



Use Lighter Point Colors

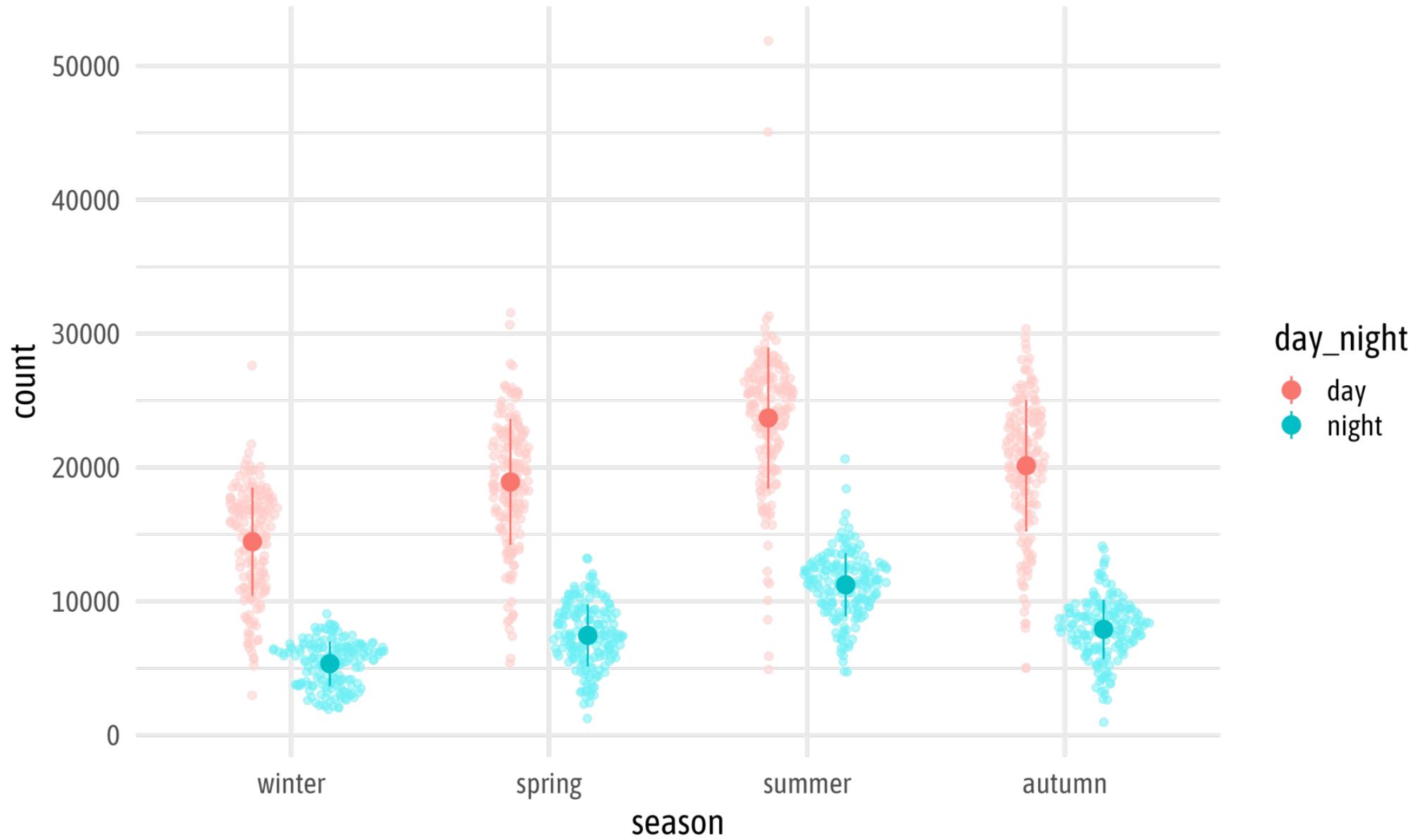


Use Standard Deviation

```
1 p1 <- ggplot(  
2   bikes,  
3   aes(x = season, y = count)  
4 ) +  
5   ggforce::geom_sina(  
6   aes(color = stage(  
7     day_night,  
8     after_scale = clr_lighten(color, .6)  
9   )),  
10  position = position_dodge(width = .6),  
11  alpha = .5  
12 ) +  
13 stat_summary(  
14   aes(color = day_night),  
15   fun = mean,  
16   fun.max = function(y) mean(y) + sd(y),  
17   fun.min = function(y) mean(y) - sd(y),  
18   position = position_dodge(width = .6),  
19   size = .8
```



Add Annotations

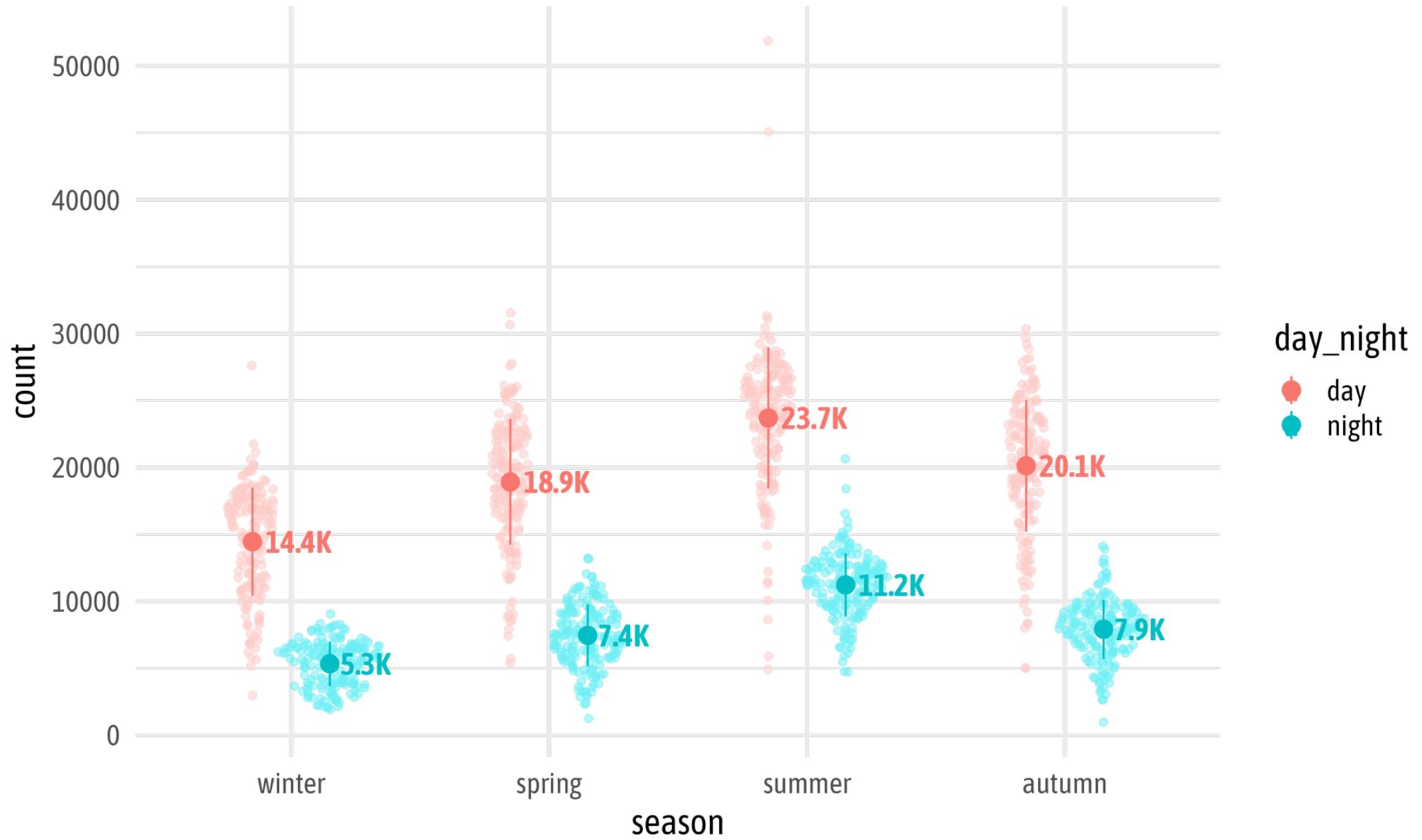


Add Annotations

```
1 p2 <- p1 +
2   stat_summary(
3     geom = "text",
4     aes(
5       color = day_night,
6       label = paste0(
7         sprintf("%2.1f", stat(y) / 1000), "K"
8       )
9     ),
10    position = position_dodge(width = .6),
11    hjust = -.2, family = "Asap Condensed",
12    size = 5.5, fontface = "bold"
13  )
14 p2
```



Add Annotations

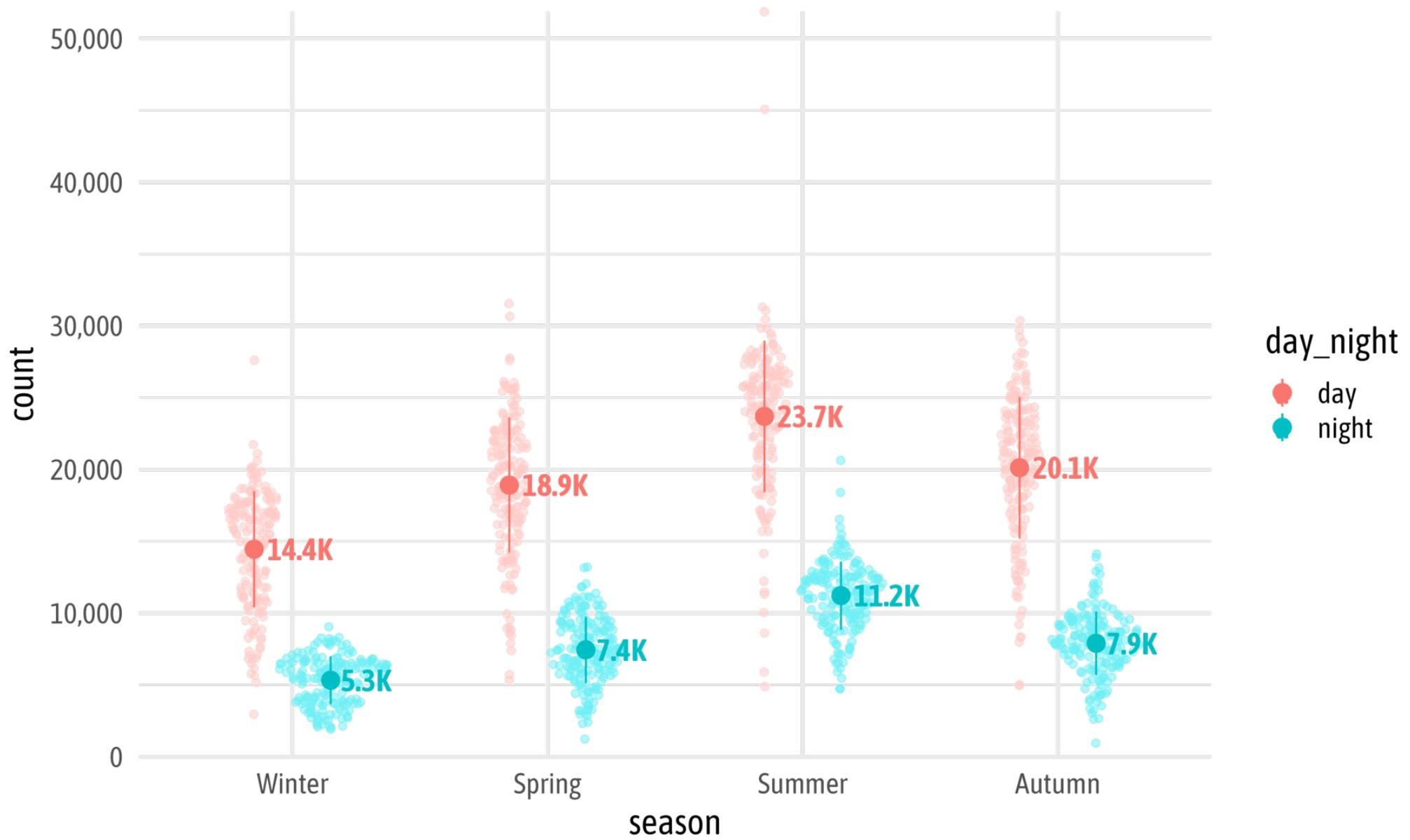


Adjust Axes + Clipping

```
1 p3 <- p2 +
2   coord_cartesian(clip = "off") +
3   scale_x_discrete(
4     labels = stringr::str_to_title
5   ) +
6   scale_y_continuous(
7     labels = scales::comma_format(),
8     expand = c(0, 0),
9     limits = c(0, NA)
10   )
11 p3
```



Adjust Axes + Clipping



Add Colors + Labels

```
1 colors <- c("#987708", "#663399")
2 p4 <- p3 +
3   scale_color_manual(
4     values = colors
5   ) +
6   labs(
7     x = NULL, y = NULL,
8     title = paste0("Reported bike shares in London during <span style='color:", colors[1], ";>day<
9     subtitle = "TfL bike sharing data from 2015 to 2016 per season and time of day.\nErrorbars show
10    )
11 p4
```



Add Colors + Labels

Reported bike shares in London during

Season	Time of Day	Mean (K)
Winter	day	14.4K
	night	5.3K
Spring	day	18.9K
	night	7.4K
Summer	day	23.7K
	night	11.2K
Autumn	day	20.1K
	night	7.9K



Theme Styling

```
1 p4 +
2   theme(
3     legend.position = "none",
4     panel.grid.major.x = element_blank(),
5     panel.grid.minor = element_blank(),
6     plot.title.position = "plot",
7     plot.title = ggtext::element_markdown(face = "bold", size = 26),
8     plot.subtitle = element_text(color = "grey30", margin = margin(t = 6, b = 12)),
9     axis.text.x = element_text(size = 17, face = "bold"),
10    axis.line.x = element_line(size = 1.2, color = "grey65"),
11    plot.margin = margin(rep(15, 4))
12  )
```

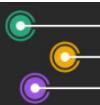
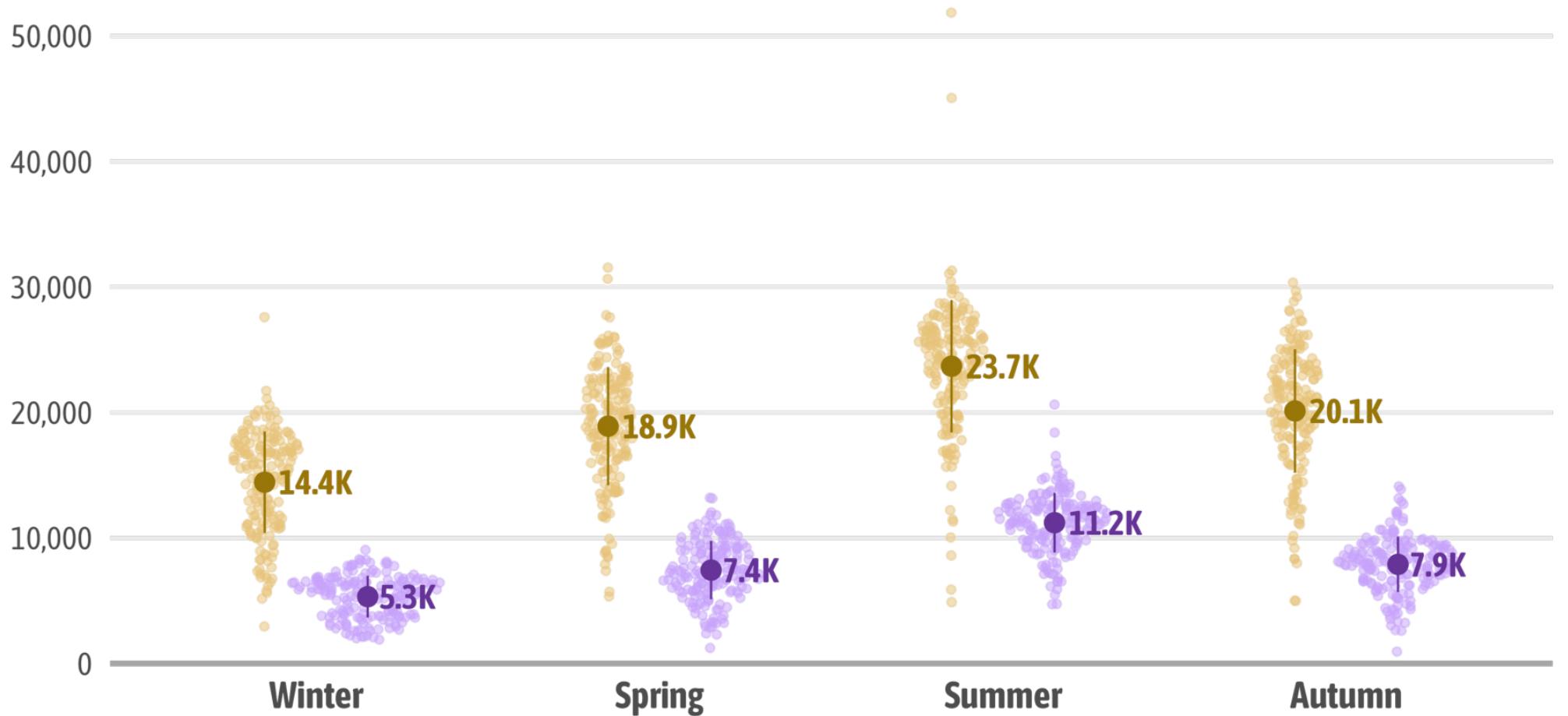


Theme Styling

Reported bike shares in London during day and night times

TfL bike sharing data from 2015 to 2016 per season and time of day.

Errorbars show the mean \pm standard deviation.



Full Code

```
1 library(prismatic)
2 library(ggtext)
3
4 bikes <- readr::read_csv(
5     "./data/london-bikes-custom.csv",
6     col_types = "Dcffffilllddddc"
7 )
8
9 bikes$season <- forcats::fct_inorder(bikes$season)
10 colors <- c("#987708", "#663399")
11
12 ggplot(bikes, aes(x = season, y = count)) +
13     ggforce::geom_sina(
14         aes(
15             color = stage(
16                 day_night, after_scale = clr_lighten(color, .6)
17             )),
18             position = position_dodge(width = .6),
19             alpha = .5
```

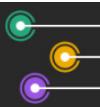


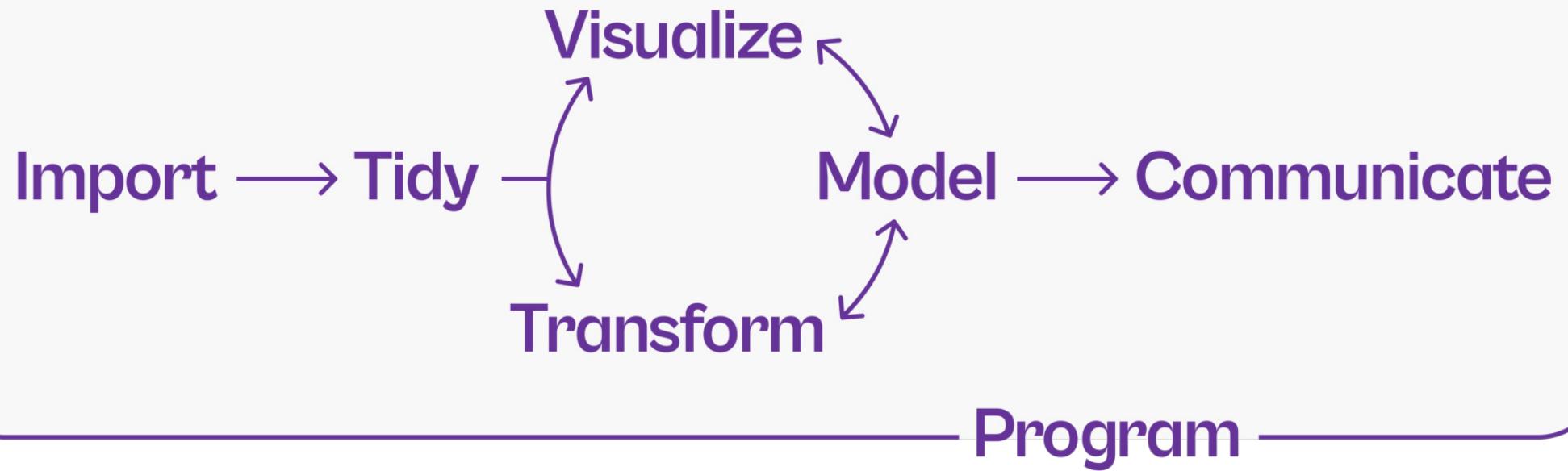
Resources

- “[ggplot2: Elegant Graphics for Data Analysis](#)” by Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen.
- “[Data Visualization: A Practical Introduction](#)” by Kieran Healy
- “[R Graphics Cookbook](#)” by Winston Chang
- [ggplot2 Extension Gallery](#)
- [R Graph Gallery](#)
- [Official ggplot2 reference](#)
- [Official ggplot2 cheatsheet](#)
- [aesthetics finder](#)
- My “[A {ggplot2} Tutorial for Beautiful Plotting in R](#)” tutorial



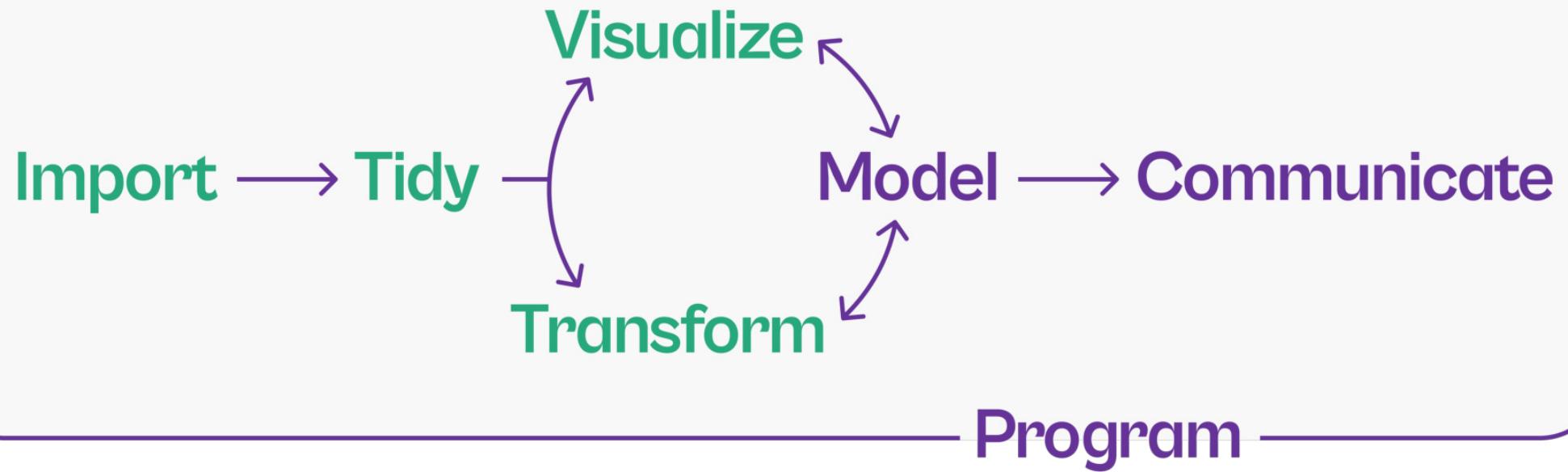
What's Next?





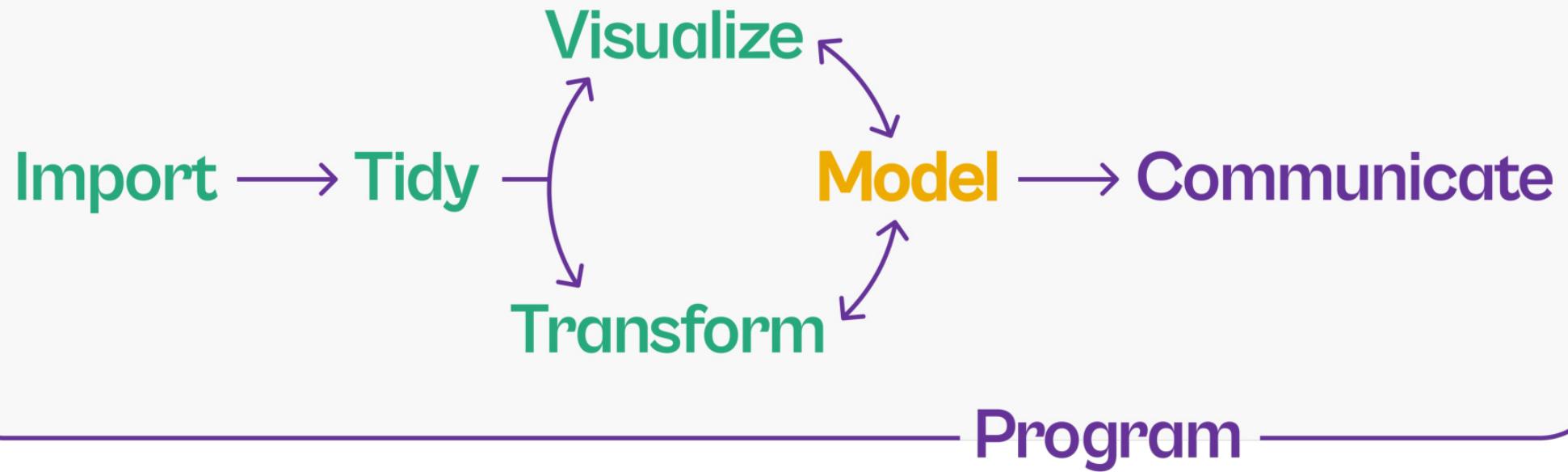
The data science workflow, modified from "R for Data Science"





The data science workflow, modified from "R for Data Science"





The data science workflow, modified from "R for Data Science"



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

