Predict Taxi Fares in Miami with Random Forests

Avery Holloman

2024-08-06

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(readxl)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(lubridate)
library(tree)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Read in the taxi data from your Excel file
taxi <- read_excel("C:/Users/jacob/Downloads/florida_taxi_data.xlsx")

# Define the columns you want to keep
columns <- c('medallion', 'pickup_datetime', 'pickup_longitude', 'pickup_latitude',
             'trip_time_in_secs', 'fare_amount', 'tip_amount')

# Filter the dataset based on existing columns and apply transformations
taxi <- taxi %>%
    select(all_of(columns)) %>%  # Select only the relevant columns
    rename(lat = pickup_latitude, long = pickup_longitude) %>%  # Rename columns
    filter(fare_amount > 0)  # Filter out rows where the fare is zero or negative

# View the first few rows of the cleaned dataset
head(taxi)
```

```
## # A tibble: 6 × 7
##   medallion        pickup_datetime      long   lat trip_time_in_secs fare_amount
##   <chr>            <dttm>              <dbl> <dbl>             <dbl>       <dbl>
## 1 184F153AAB28A66… 2013-11-30 20:49:41 -81.4  29.2              3465        58.7
## 2 3CB9B937EBC9CE2… 2013-02-04 02:10:02 -81.5  25.1               925        82.9
## 3 2580D929DC6DAC4… 2013-12-26 09:49:06 -81.4  29.1               217        41.2
## 4 FBEC42464E15317… 2013-08-30 09:35:26 -81.5  26.5               206        89.7
## 5 F3A77E1608334F4… 2013-08-20 18:59:16 -81.1  26.3              2311        80.2
## 6 C92053C1314A4E8… 2013-01-15 10:48:34 -81.4  28.1              1505        54.7
## # ℹ 1 more variable: tip_amount <dbl>
```

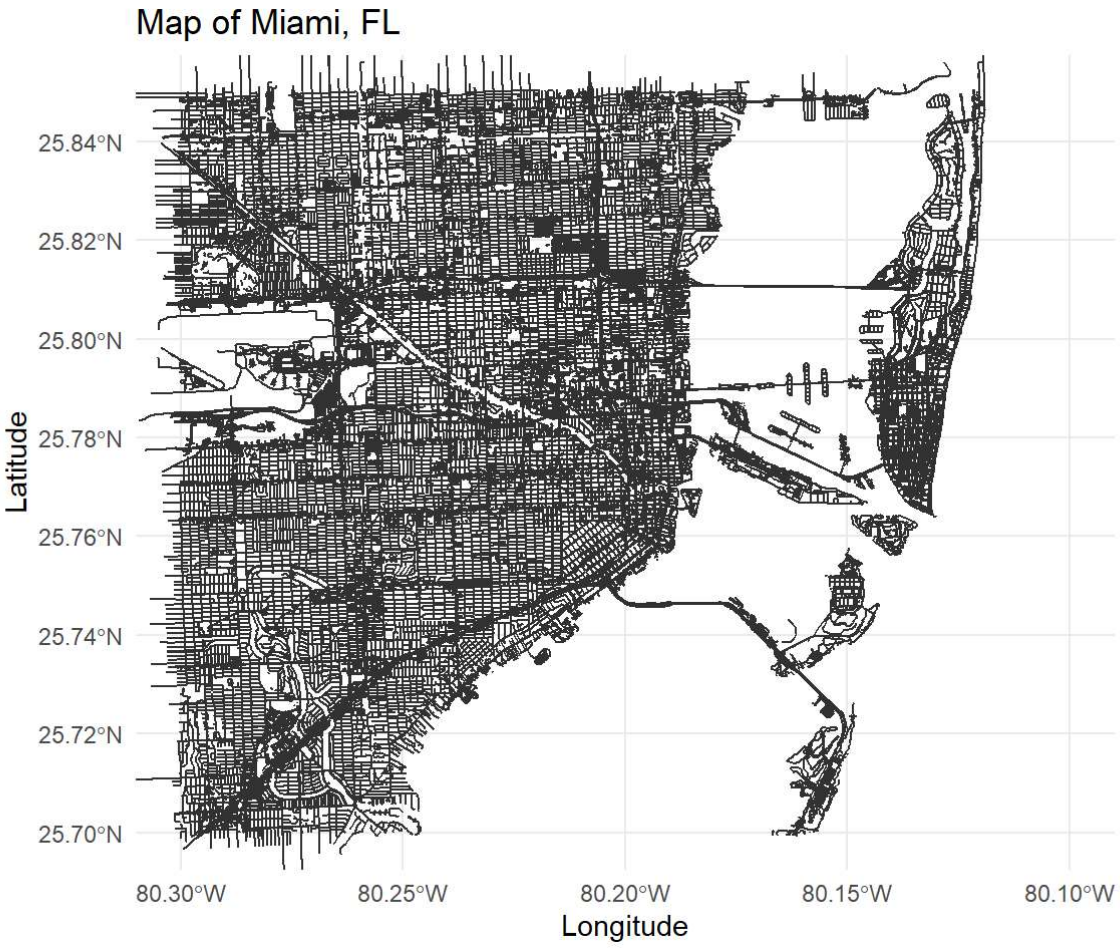
```
library(osmdata)
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. https://www.openstreetmap.org/copyright
```

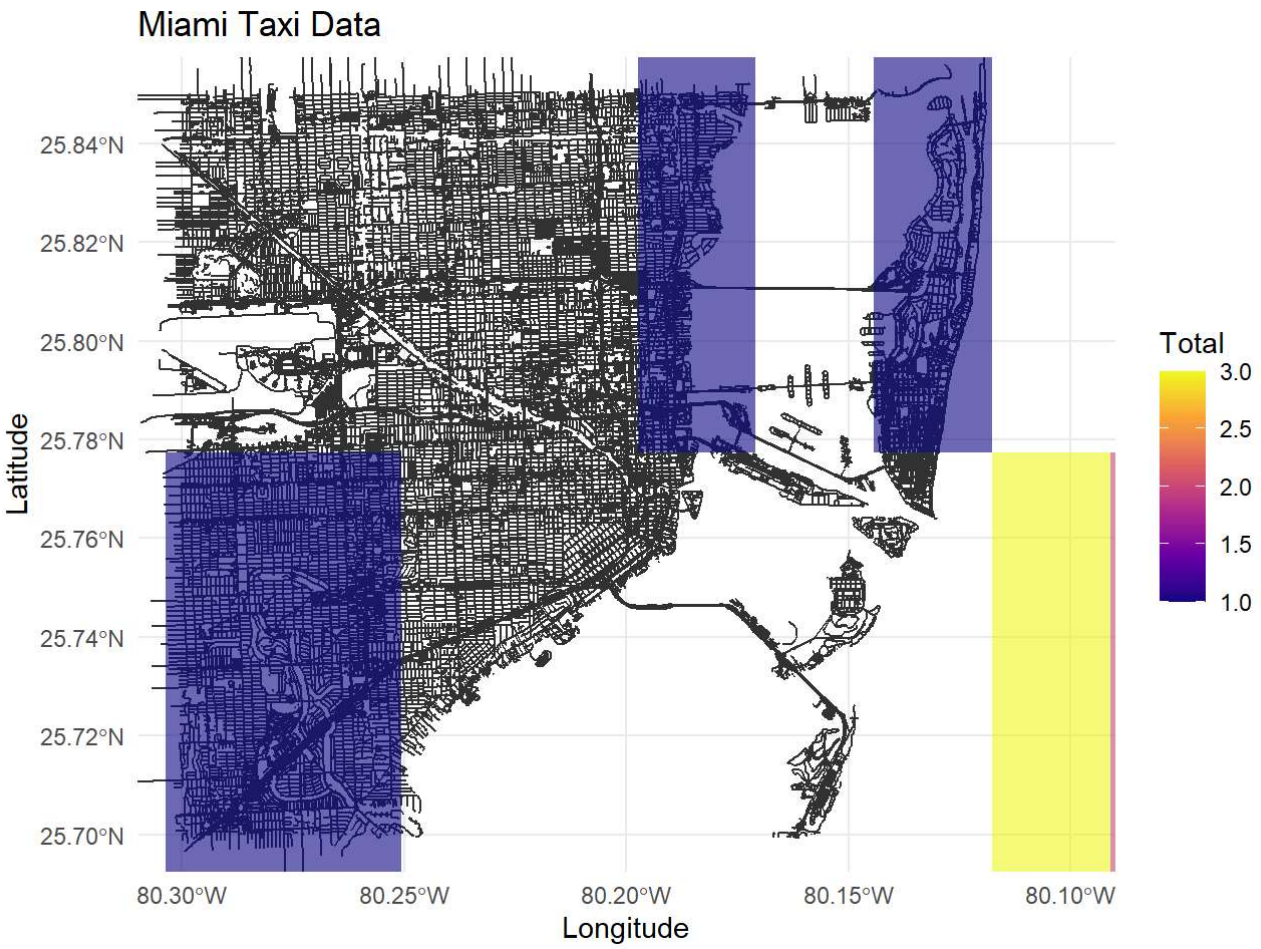
```
library(sf)
```

```
## Linking to GEOS 3.12.1, GDAL 3.8.4, PROJ 9.3.1; sf_use_s2() is TRUE
```

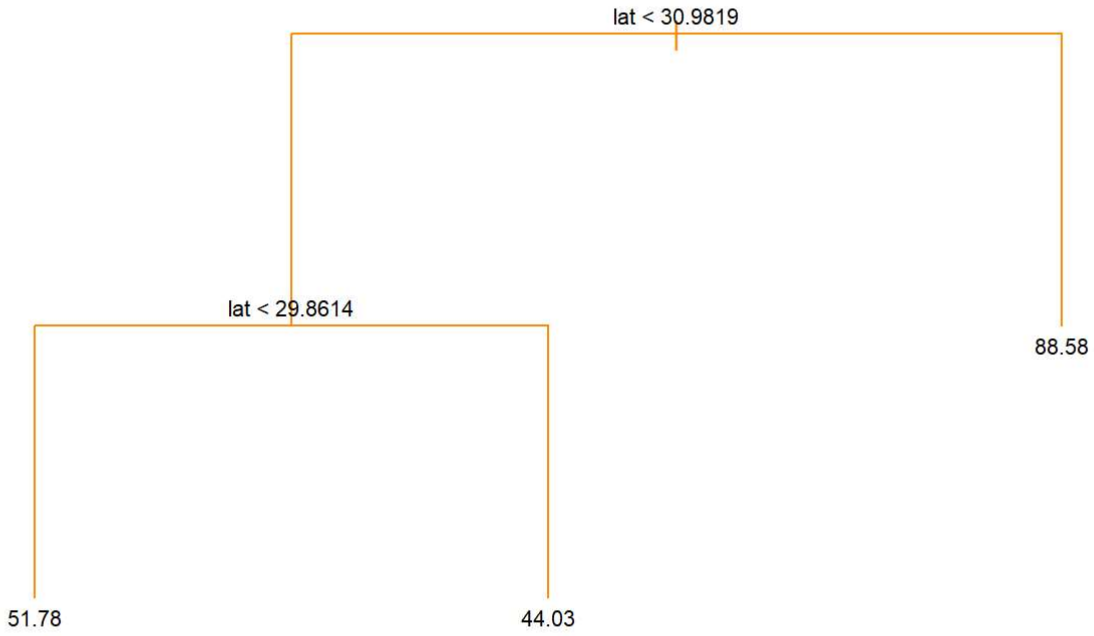
```
library(ggplot2)

# Define the bounding box for Miami
bbox <- c(left = -80.30, bottom = 25.70, right = -80.10, top = 25.85)

# Query OSM data for streets and other features
miami_osm <- opq(bbox = bbox) %>%
  add_osm_feature(key = 'highway') %>%
  osmdata_sf()

# Plot the Miami map using ggplot2
ggplot() +
  geom_sf(data = miami_osm$osm_lines, color = "grey20", size = 0.5) +
  coord_sf(xlim = c(-80.30, -80.10), ylim = c(25.70, 25.85)) +
  theme_minimal() +
  labs(title = "Map of Miami, FL", x = "Longitude", y = "Latitude")
```



```
# Plot the Miami map with taxi data
ggplot() +
  geom_sf(data = miami_osm$osm_lines, color = "grey20", size = 0.5) +
  geom_bin2d(data = taxi, aes(x = long, y = lat), bins = 60, alpha = 0.6) +
  scale_fill_viridis_c(option = 'plasma', name = "Total") +
  coord_sf(xlim = c(-80.30, -80.10), ylim = c(25.70, 25.85)) +
  labs(title = "Miami Taxi Data", x = "Longitude", y = "Latitude") +
  theme_minimal()
```




```
# Prepare the data with additional features
taxi <- taxi %>%
    mutate(hour = hour(pickup_datetime),
           wday = wday(pickup_datetime, label = TRUE),
           month = month(pickup_datetime, label = TRUE))

# Fit a regression tree model
fitted_tree <- tree(fare_amount ~ lat + long + hour + wday + month, data = taxi)

# Plot the regression tree
plot(fitted_tree, col = "darkorange")
text(fitted_tree, pretty = 0, cex = 0.7)
```




```
# Summary of the tree model
summary(fitted_tree)
```

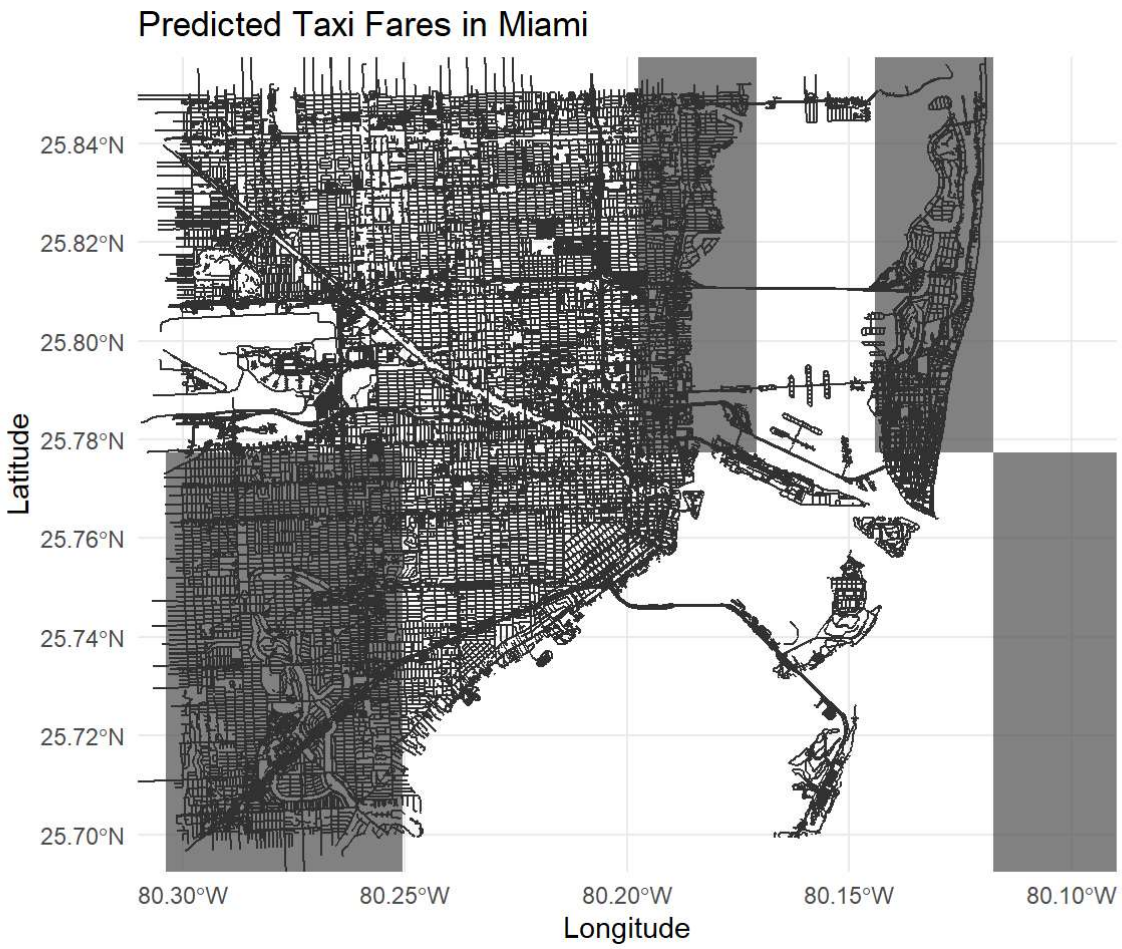
```
##
## Regression tree:
## tree(formula = fare_amount ~ lat + long + hour + wday + month,
##     data = taxi)
## Variables actually used in tree construction:
## [1] "lat"
## Number of terminal nodes:  3
## Residual mean deviance:  784.3 = 781900 / 997
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -48.500 -24.560  -1.023   0.000  23.660  54.320
```

```
# Check the number of rows in your dataset
num_rows <- nrow(taxi)

# Fit a random forest model with an appropriate sampsize
fitted_forest <- randomForest(fare_amount ~ lat + long + hour + wday + month,
                              data = taxi, ntree = 80, sampsize = min(10000, num_rows))

# Print the random forest model
print(fitted_forest)
```

```
##
## Call:
##  randomForest(formula = fare_amount ~ lat + long + hour + wday +      month, data = taxi, ntr
ee = 80, sampsize = min(10000, num_rows))
##                Type of random forest: regression
##                      Number of trees: 80
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 845.1973
##                     % Var explained: -5.84
```

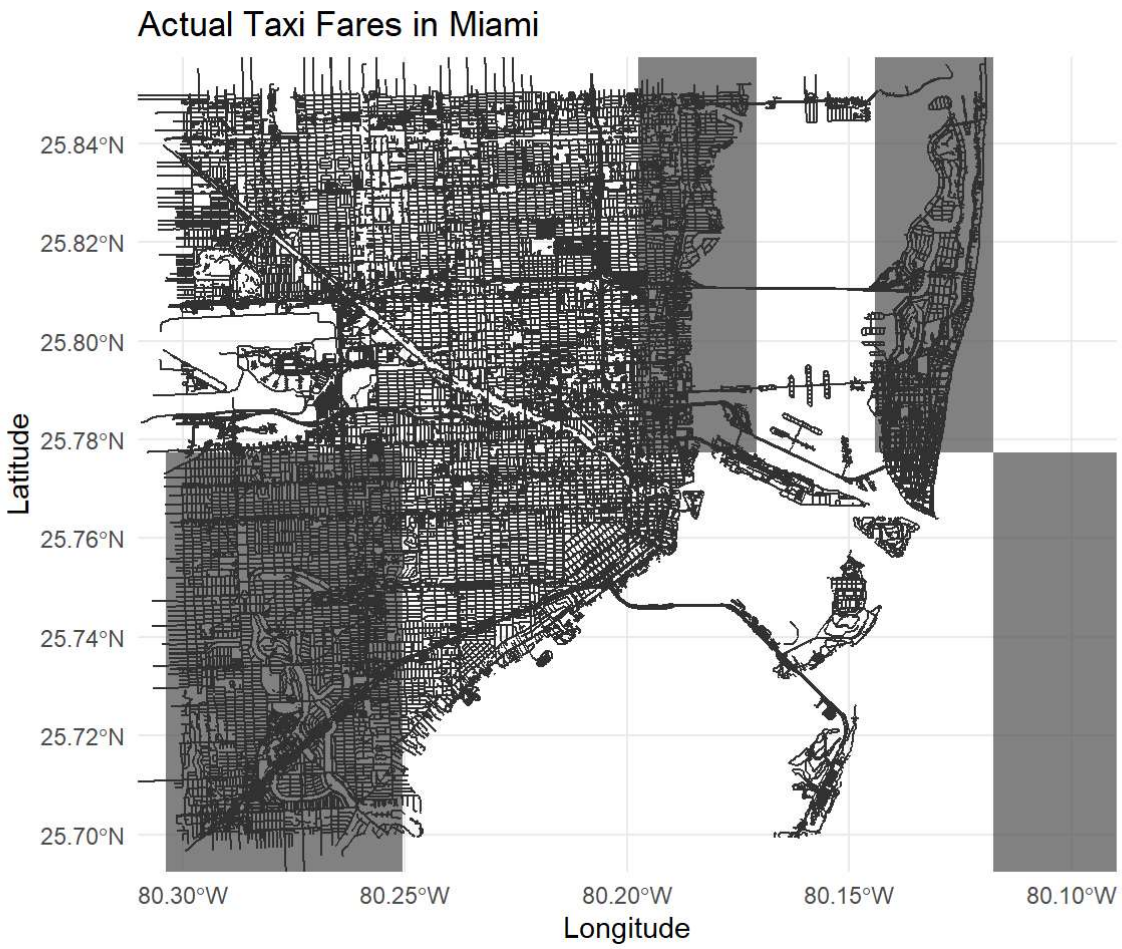
```
# Add predictions to the taxi dataset
taxi$pred_total <- fitted_forest$predicted

# Predicted fare map
ggplot() +
  geom_sf(data = miami_osm$osm_lines, color = "grey20", size = 0.5) +
  geom_bin2d(data = taxi, aes(x = long, y = lat, fill = pred_total), bins = 60, alpha = 0.6) +
  scale_fill_viridis_c(option = 'plasma', name = "Predicted Fare") +
  coord_sf(xlim = c(-80.30, -80.10), ylim = c(25.70, 25.85)) +
  labs(title = "Predicted Taxi Fares in Miami", x = "Longitude", y = "Latitude") +
  theme_minimal()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill.
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```




```
# Actual fare map
ggplot() +
  geom_sf(data = miami_osm$osm_lines, color = "grey20", size = 0.5) +
  geom_bin2d(data = taxi, aes(x = long, y = lat, fill = fare_amount), bins = 60, alpha = 0.6) +
  scale_fill_viridis_c(option = 'plasma', name = "Total Fare") +
  coord_sf(xlim = c(-80.30, -80.10), ylim = c(25.70, 25.85)) +
  labs(title = "Actual Taxi Fares in Miami", x = "Longitude", y = "Latitude") +
  theme_minimal()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill.
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```




```
# Calculate average predicted and actual fares by hour
taxi_hourly <- taxi %>%
    group_by(hour) %>%
    summarize(pred_per_hour = mean(pred_total, na.rm = TRUE),
              per_hour = mean(fare_amount, na.rm = TRUE))

# Plotting predicted vs actual fares by hour
colors <- c("darkorange", "darkblue")

ggplot(taxi_hourly, aes(x = hour)) +
    geom_line(aes(y = pred_per_hour, color = colors[1]), size = 1.2) +
    geom_line(aes(y = per_hour, color = colors[2]), size = 1.2) +
    scale_color_manual(name = '', values = colors, labels = c("Predicted", "Actual")) +
    labs(x = "Hour of Day", y = "Average Fare") +
    theme_minimal() +
    theme(legend.position = c(0.8, 0.8), legend.title = element_blank())
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## ℹ Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```