ggplot2

Devan Morehouse

2023-09-17

Establishing the requisite datasets and configuring the essential libraries for the task at hand.

```
library("datasets")
library(tidyverse)## ggplot included
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ 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
```

```
# Cars dataset
## Change cyl, am & wt to factors
cars <- mtcars
cars$am <- factor(cars$am, labels = c("Auto","Manual"))
cars$cyl <- factor(cars$cyl, levels=c("4", "6", "8"),
                   labels = c("Four","Six","Eight"), ordered=TRUE)
brks <- c(0., 2.5, 3.5, 6.)
labs <- c("Light", "Medium", "Heavy")
cars$carwt <- cut(cars$wt, breaks = brks, labels = labs,
                  ordered_result=TRUE)

#Convert carwt to a factor for forward compatibility reasons
cars$carwt <- factor(cars$carwt, ordered = TRUE)


#Titanic dataset
dt <- read_csv("C:\\Users\\dev46\\OneDrive\\Desktop\\School Documents\\Summer 2023\\DAT 500\\Dat
a\\titanic.csv")
```

```
## Rows: 887 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): Name, Sex
## dbl (6): Survived, Pclass, Age, Siblings/Spouses Aboard, Parents/Children Ab...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dt1 <- dt
##  Factor variables
dt1$Pclass <- factor(dt1$Pclass, levels=c("1", "2", "3"),
                     labels=c("1st", "2nd", "3rd"), ordered = TRUE)
dt1$Sex <- factor(dt1$Sex)
dt1$Survived <- factor(dt1$Survived, levels=c("1", "0"),
  labels = c("Survived", "Died"))
#Violin plot Data
dt1a <- dt1 |> filter(Fare < 300)
#Tile plot data
dt4 <- dt1 |> group_by(Survived, Pclass) |> summarize(Number = n())
```

```
## `summarise()` has grouped output by 'Survived'. You can override using the
## `.groups` argument.
```

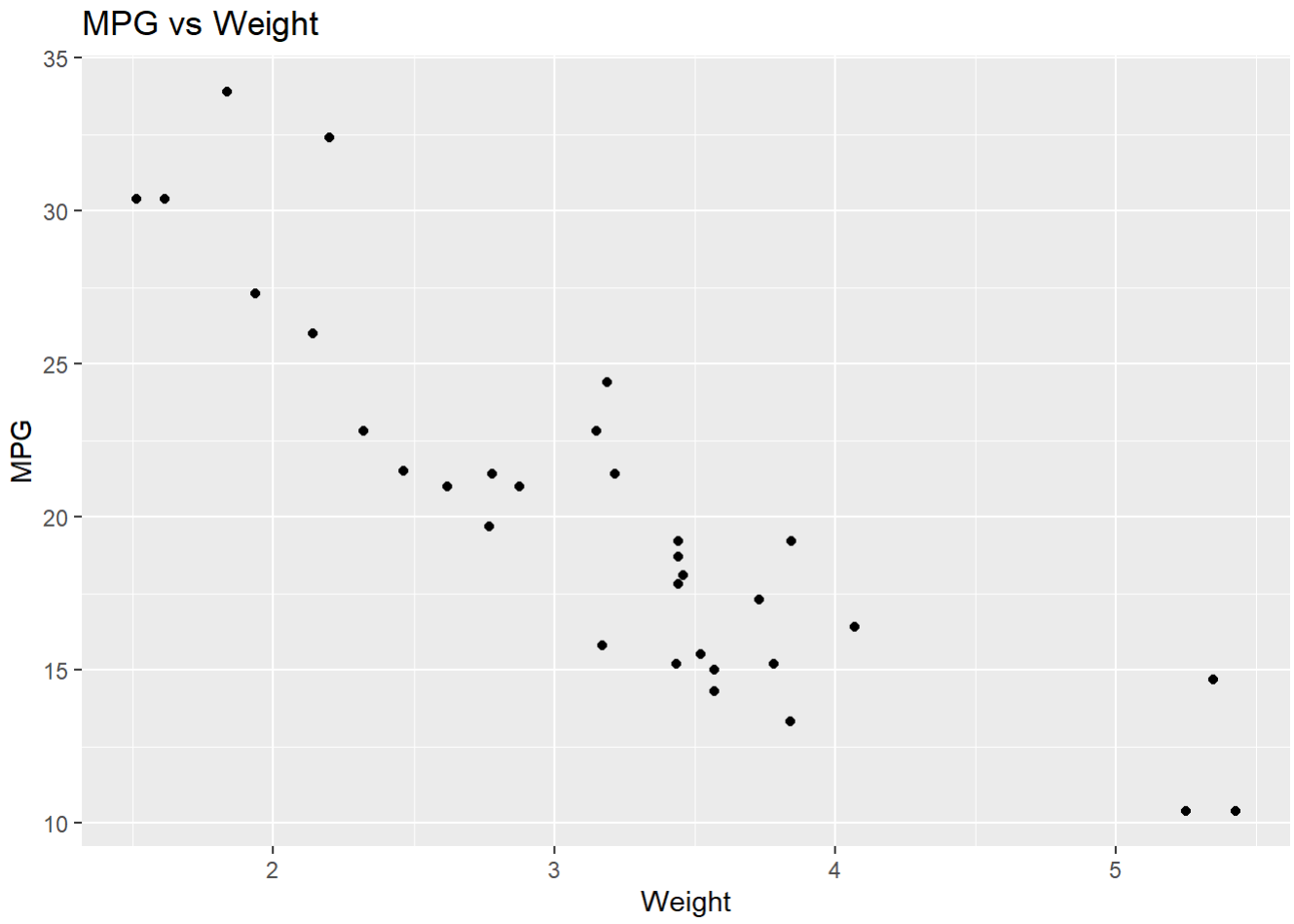
```
#Mosaic plot data
dt5 <- with(dt1, table(Survived, Pclass))
dt51 <- with(dt1, table(Pclass, Survived))
dt52 <- with(dt1, table(Pclass, Survived, Sex))

#Pie chart Data
dt2 <- dt1 |>
  #Retain only rows we care about
  select(Survived, Pclass, Sex) |>
  #We want to make a pie chart of classes
  group_by(Pclass) |>
  summarize(Number = n()) |>
  #Put in proportions
  mutate(Prop = Number / sum(Number))

# Rose plot
dt3 <- dt1 |> select(Survived, Pclass, Sex)



#Degrees and Race Dataset
degrees_by_race <- read_csv("C:\\Users\\dev46\\OneDrive\\Desktop\\School Documents\\Summer 2023
\\DAT 500\\Data\\masters_degrees_race.csv", show_col_types = FALSE,
                            na = c("NA", ""))

degrees_by_race$Year_fct <- factor(degrees_by_race$Year, levels = min(degrees_by_race$Year):max
(degrees_by_race$Year), ordered = TRUE)

dbr_longer <- degrees_by_race |> pivot_longer(White:Non_res, names_to = "Race", values_to = "Deg
rees")
```

Scatterplots

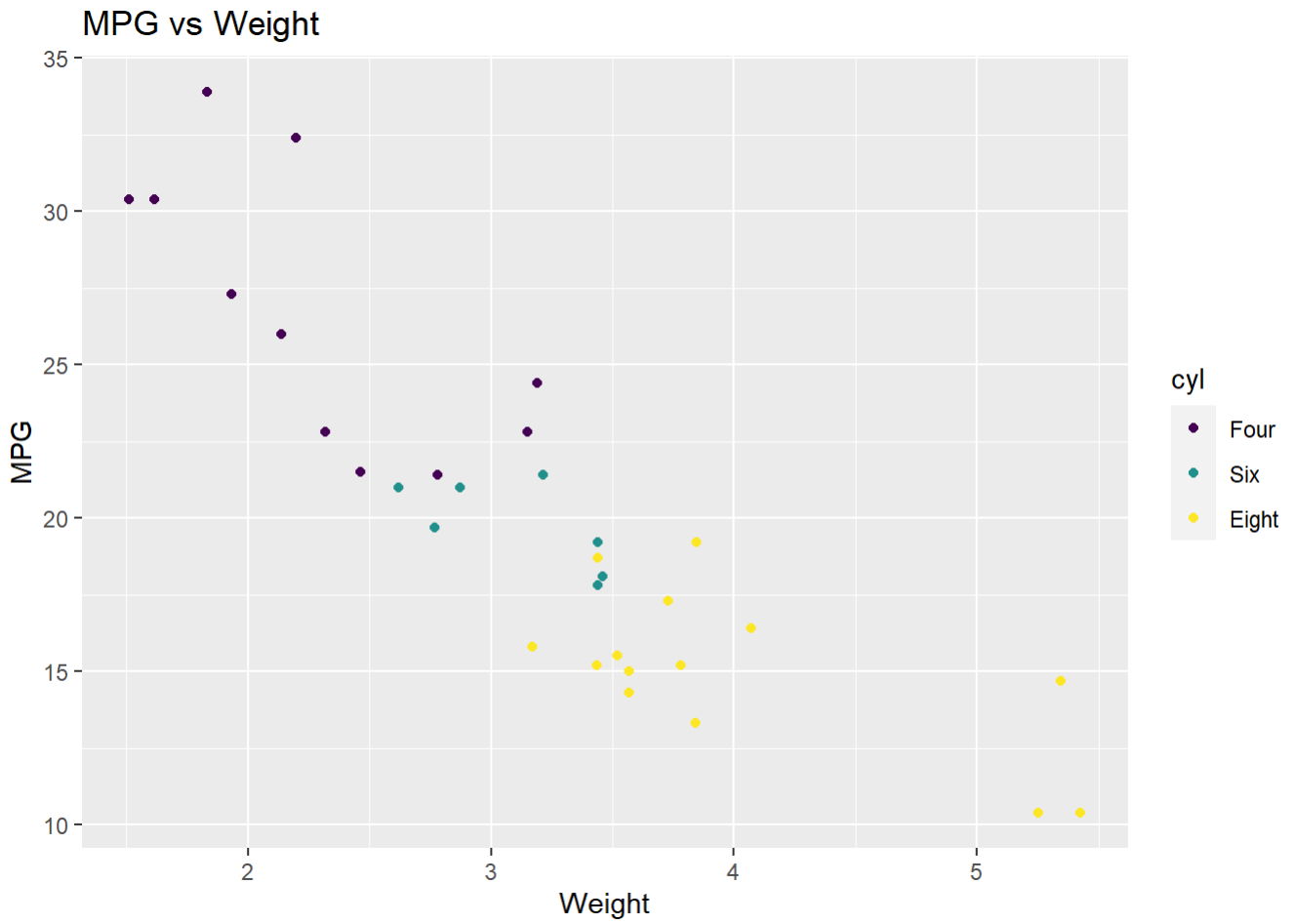
Scatterplots are a valuable tool in data analysis, particularly when you need to visualize the relationship between two continuous variables. They are particularly useful when exploring correlations, patterns, or trends in your data. In R, you typically use numeric data types for scatterplots, as they allow you to represent the range and distribution of values effectively. By plotting one numeric variable on the x-axis and another on the y-axis, you can gain insights into how these variables interact, whether they exhibit a positive or negative relationship, or if there are any clustering or outliers present. Scatterplots are especially handy when assessing variables for potential regression analysis, identifying data points that deviate from the norm, or simply exploring the underlying structure of your data.

Examples:

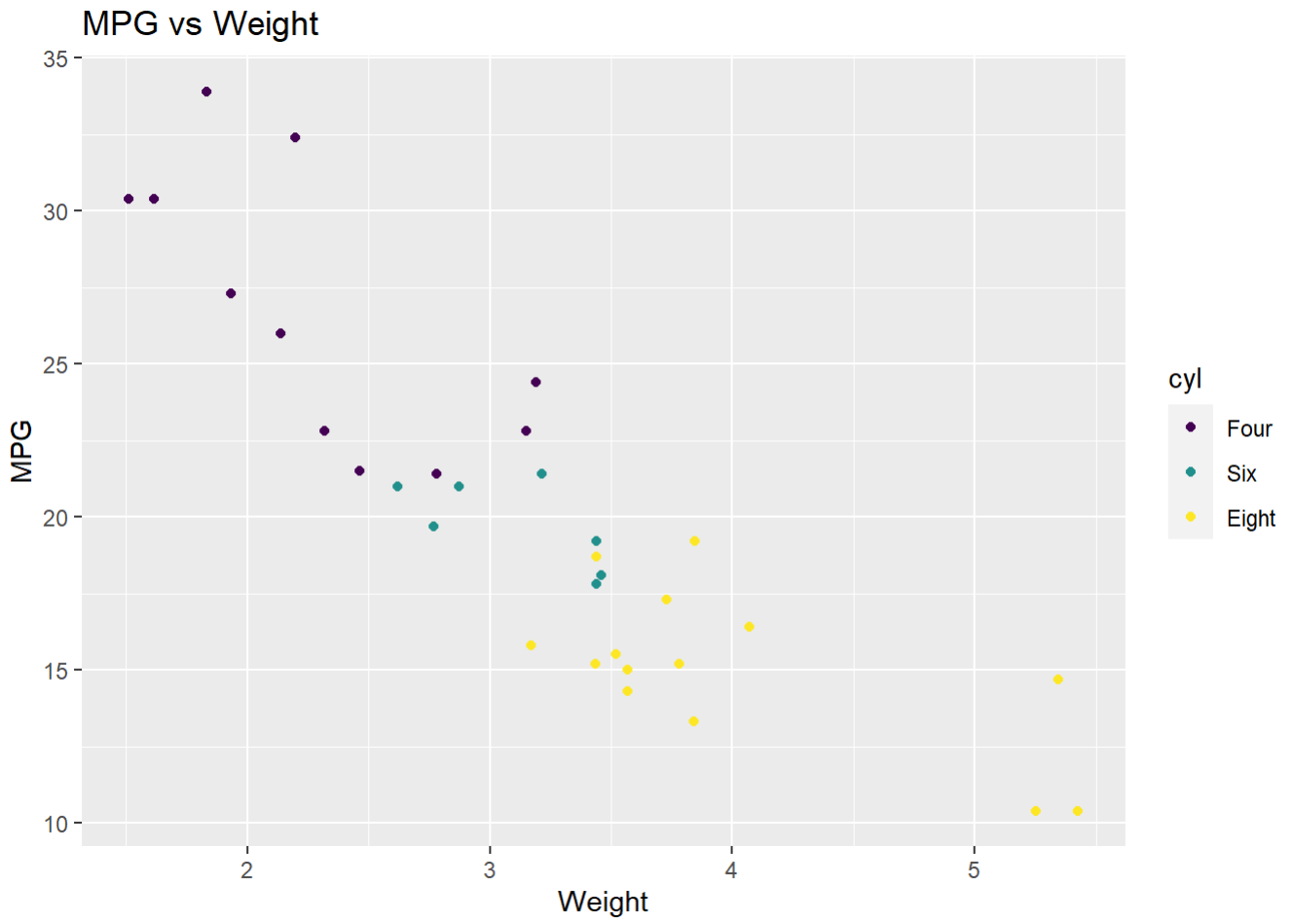
```
#Basic scatter plot
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```



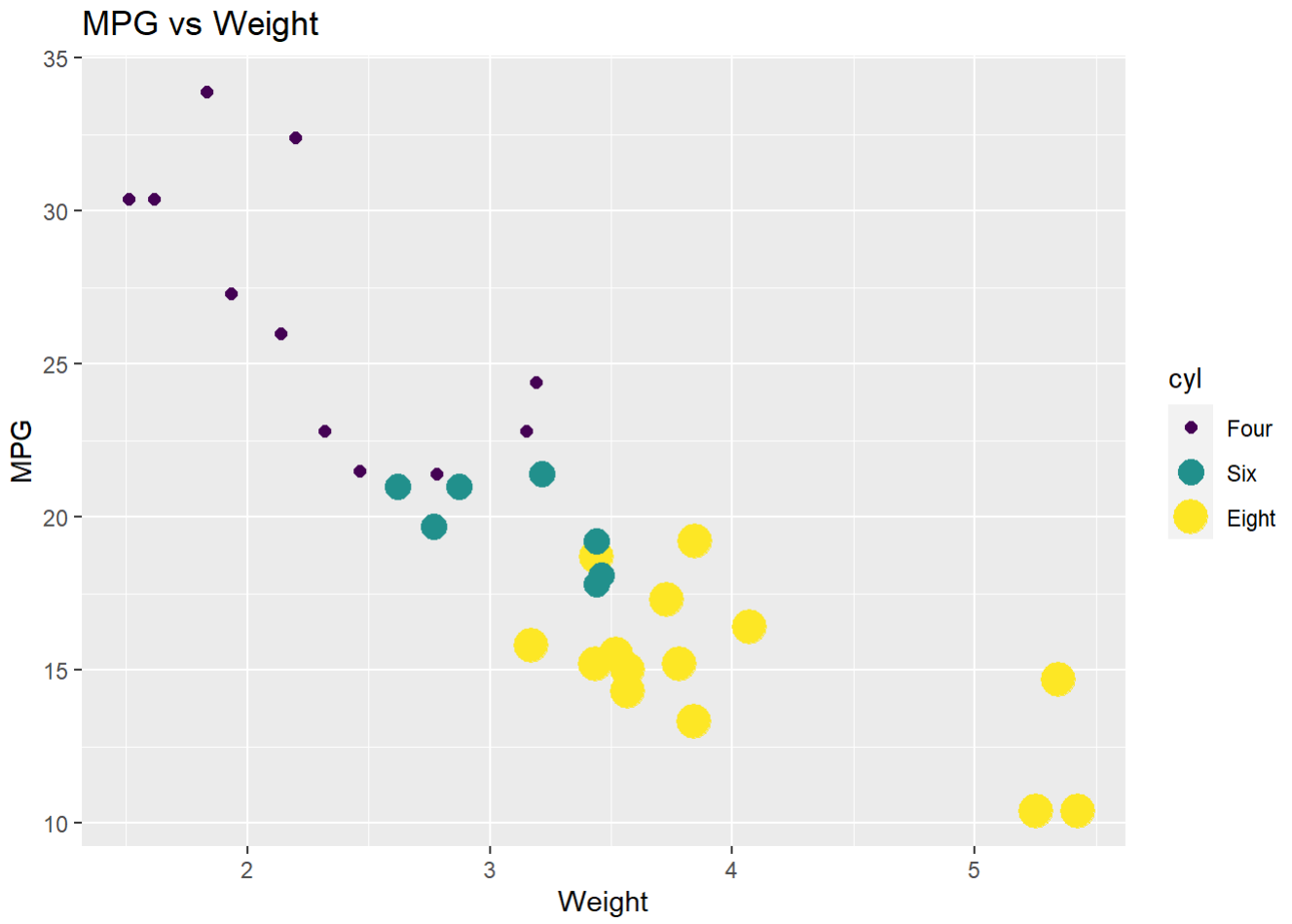

```
# Separating each cylinder type by color
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point(aes(color=cyl)) +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```




```
# or (Both yeild the same result)
ggplot(cars, aes(x=wt, y=mpg, color=cyl)) +
  geom_point() +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```

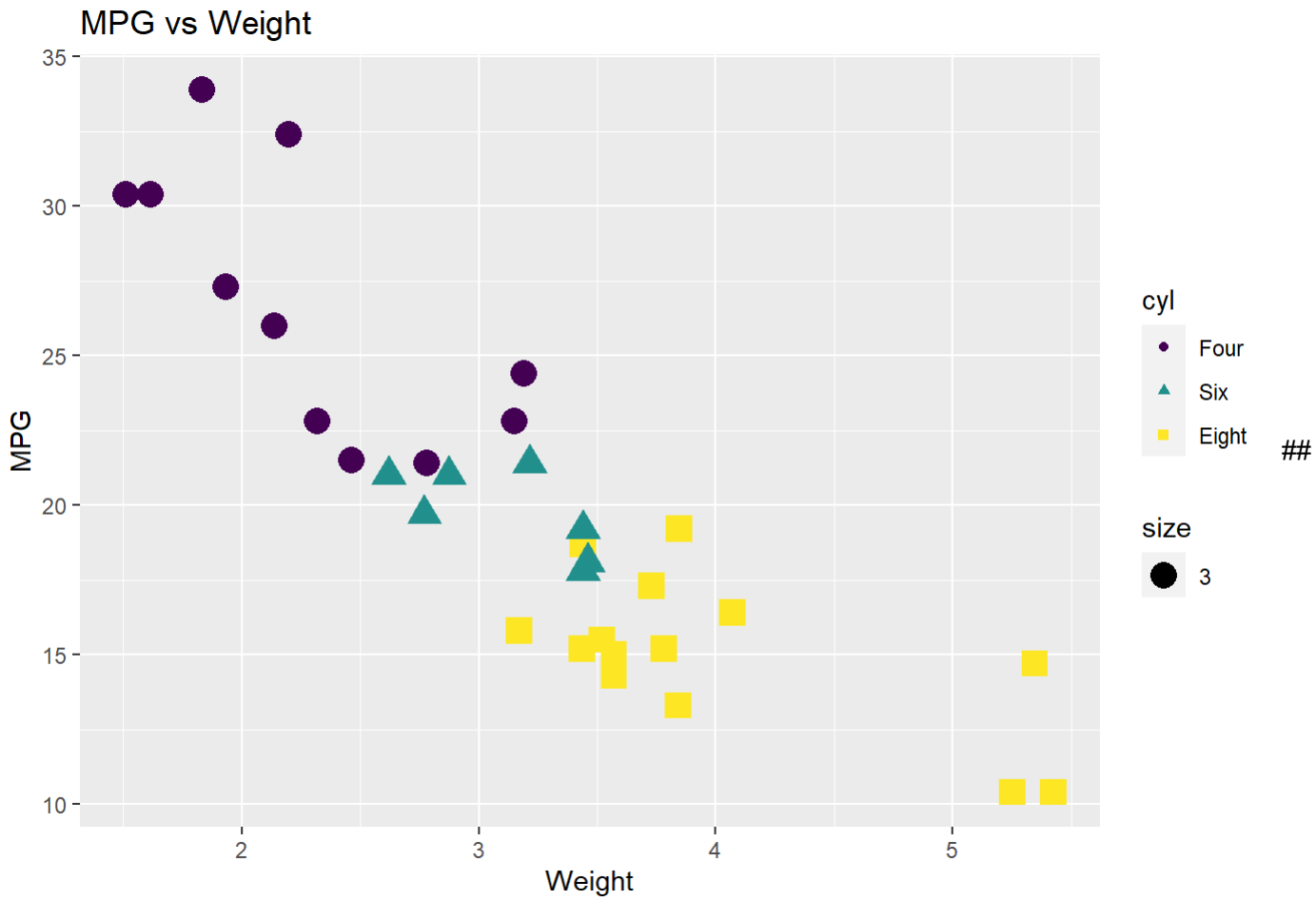



```
# Changing the size of the dots
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point(aes(color=cyl, size=cyl)) +
    labs(title="MPG vs Weight", x="Weight", y="MPG")
```




```
# Changing the shapes of the dots
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point(aes(color=cyl, shape=cyl, size =3)) +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```

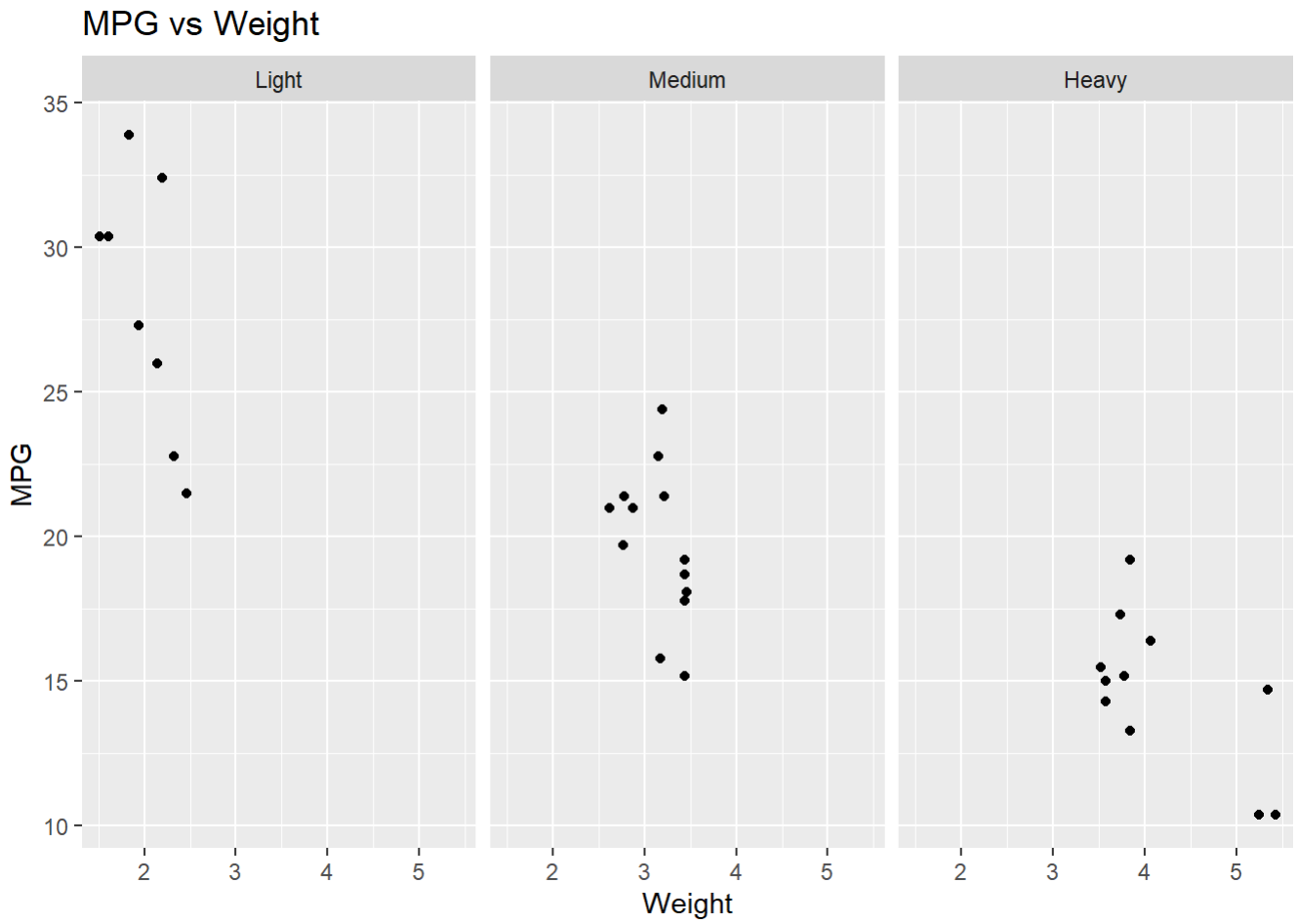
```
## Warning: Using shapes for an ordinal variable is not advised
```



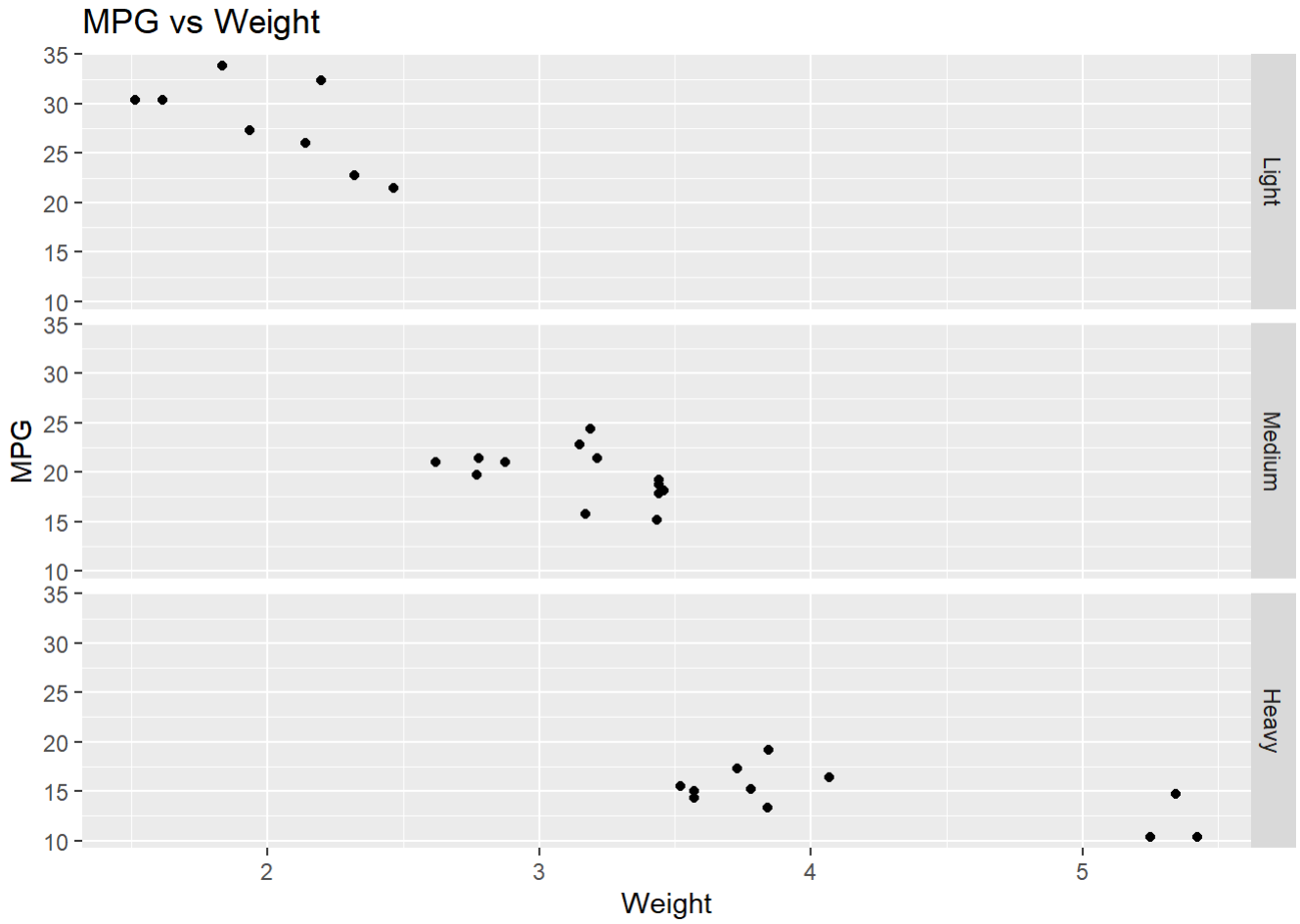
Facets

Facets, are incredibly useful when you want to examine the relationships within your data across different categories or subgroups. They are particularly valuable when dealing with categorical or discrete data types, such as factors or character variables. Faceting allows you to split your data into subsets based on one or more categorical variables and create a series of smaller, related plots, each corresponding to a unique category or combination of categories. This approach is ideal for exploring variations or patterns within your data that might be obscured when examining it as a whole. Facets can provide a comprehensive view of your data and help you make more informed decisions, especially in scenarios like comparing performance across different product categories, periods, or geographical regions.

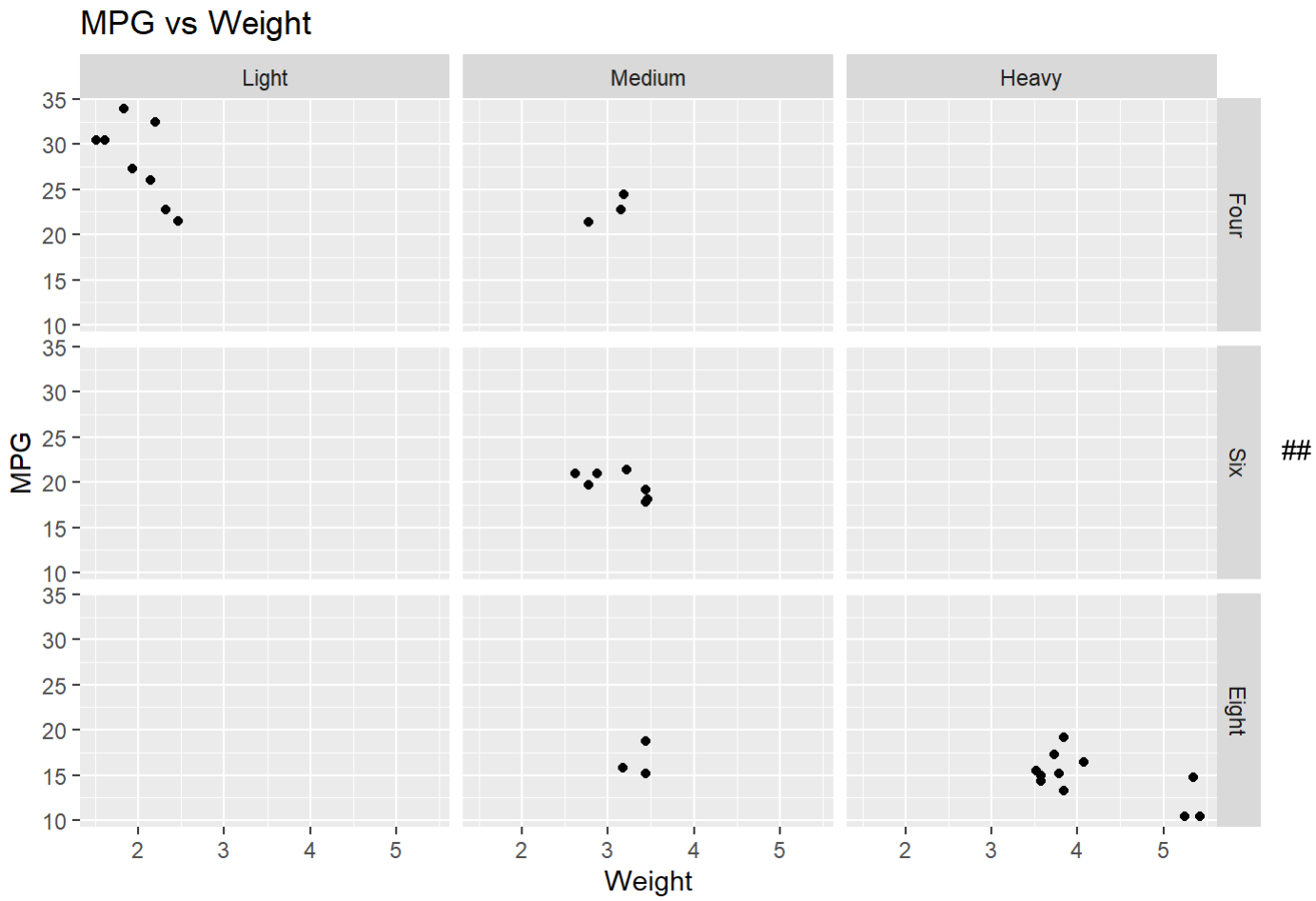
```
# Basic vertical facet
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  facet_grid(. ~ carwt) +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```




```
# Basic horizontal facet
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  facet_grid(carwt ~ .) +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```




```
#Grid shape facet
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  facet_grid(cyl ~ carwt) +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```



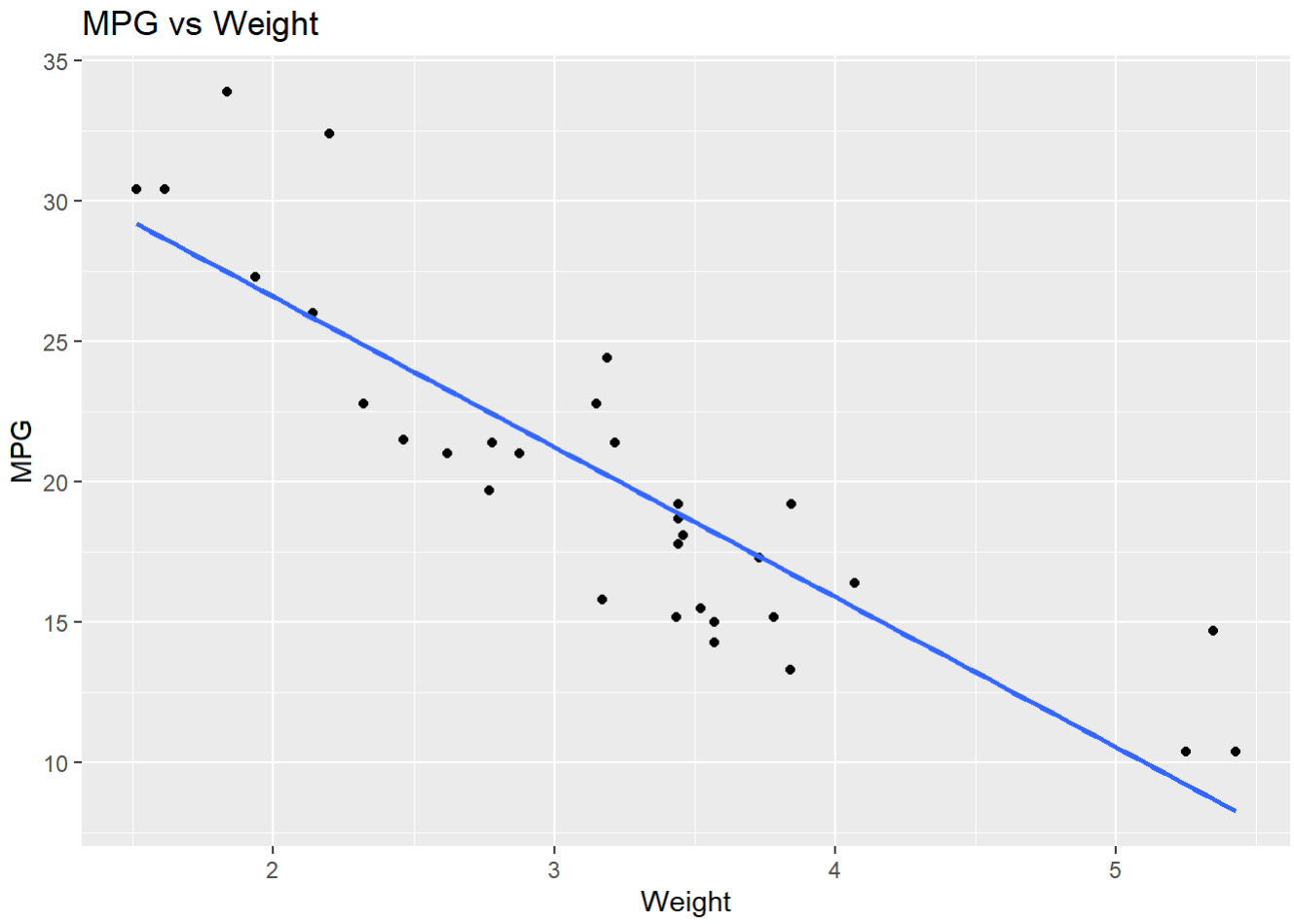
##

Smoothing

Smoothing is a valuable technique in data analysis, often employed when you want to identify underlying trends or patterns in noisy data. It's particularly useful when dealing with continuous or time-series data in R. This is especially beneficial when working with data that exhibits fluctuations or seasonality, allowing you to extract meaningful insights and make predictions. Smoothing helps simplify complex data and provides a clearer picture of the underlying patterns, making it a powerful tool for tasks like forecasting stock prices, analyzing temperature trends, or studying economic indicators.

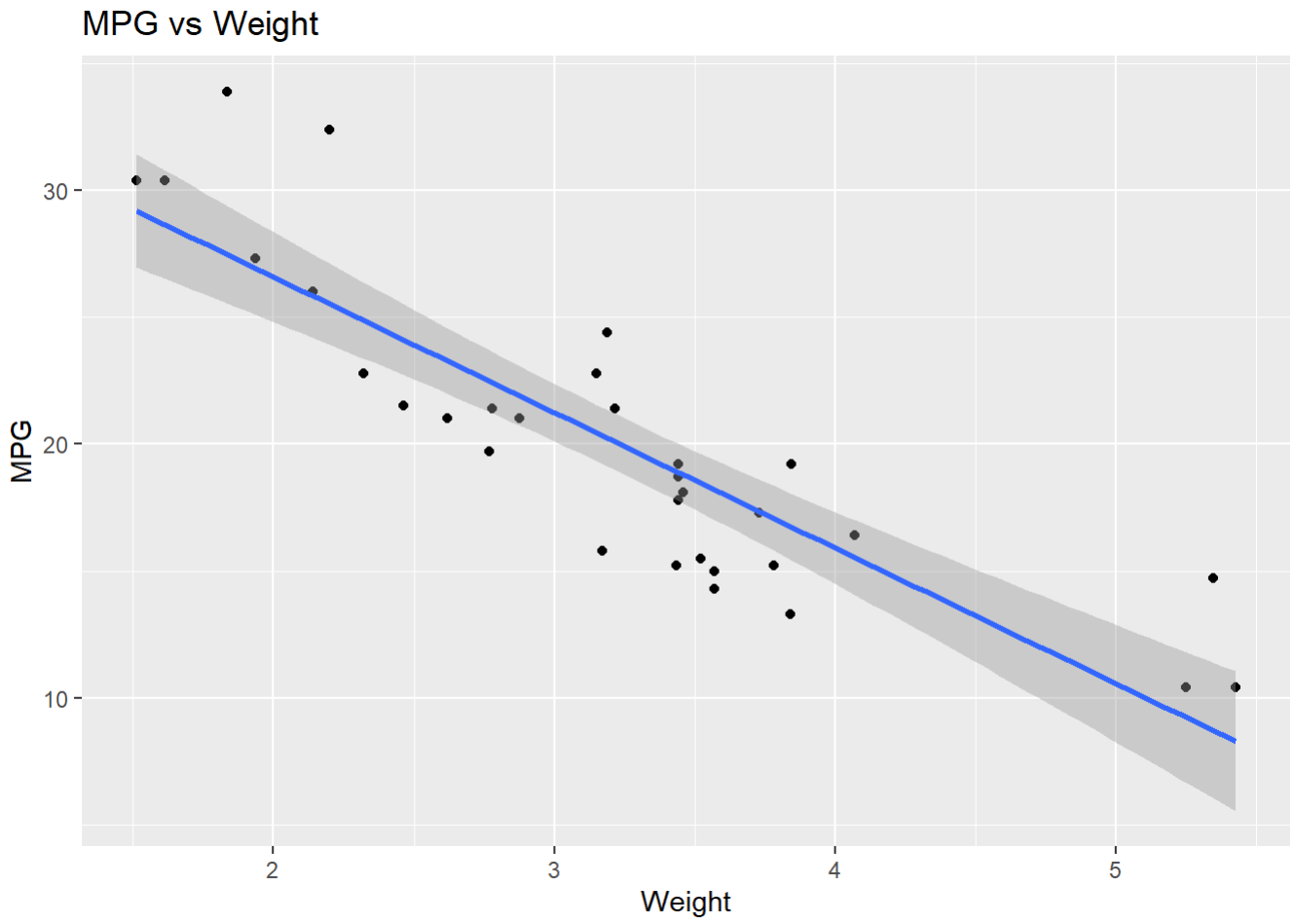
```
# Basic smoothing (line of best fit)
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se=FALSE) +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



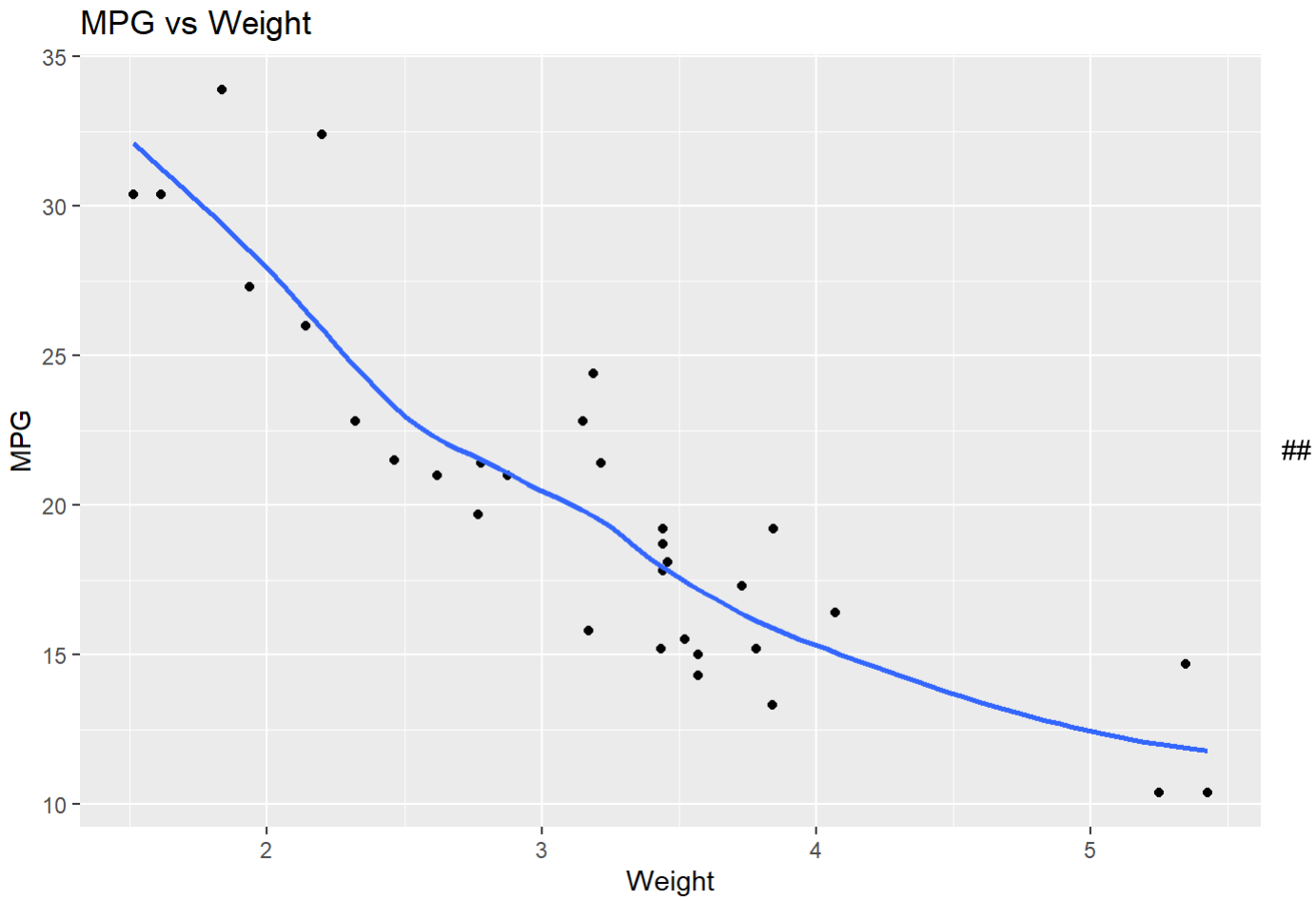

```
# With std. errors visible
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title="MPG vs Weight", x="Weight", y="MPG")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Smoothing average
ggplot(cars, aes(x=wt, y=mpg)) +
  geom_point() +
  geom_smooth(method = "loess", se= FALSE) +
  labs(title= "MPG vs Weight", x="Weight", y="MPG")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

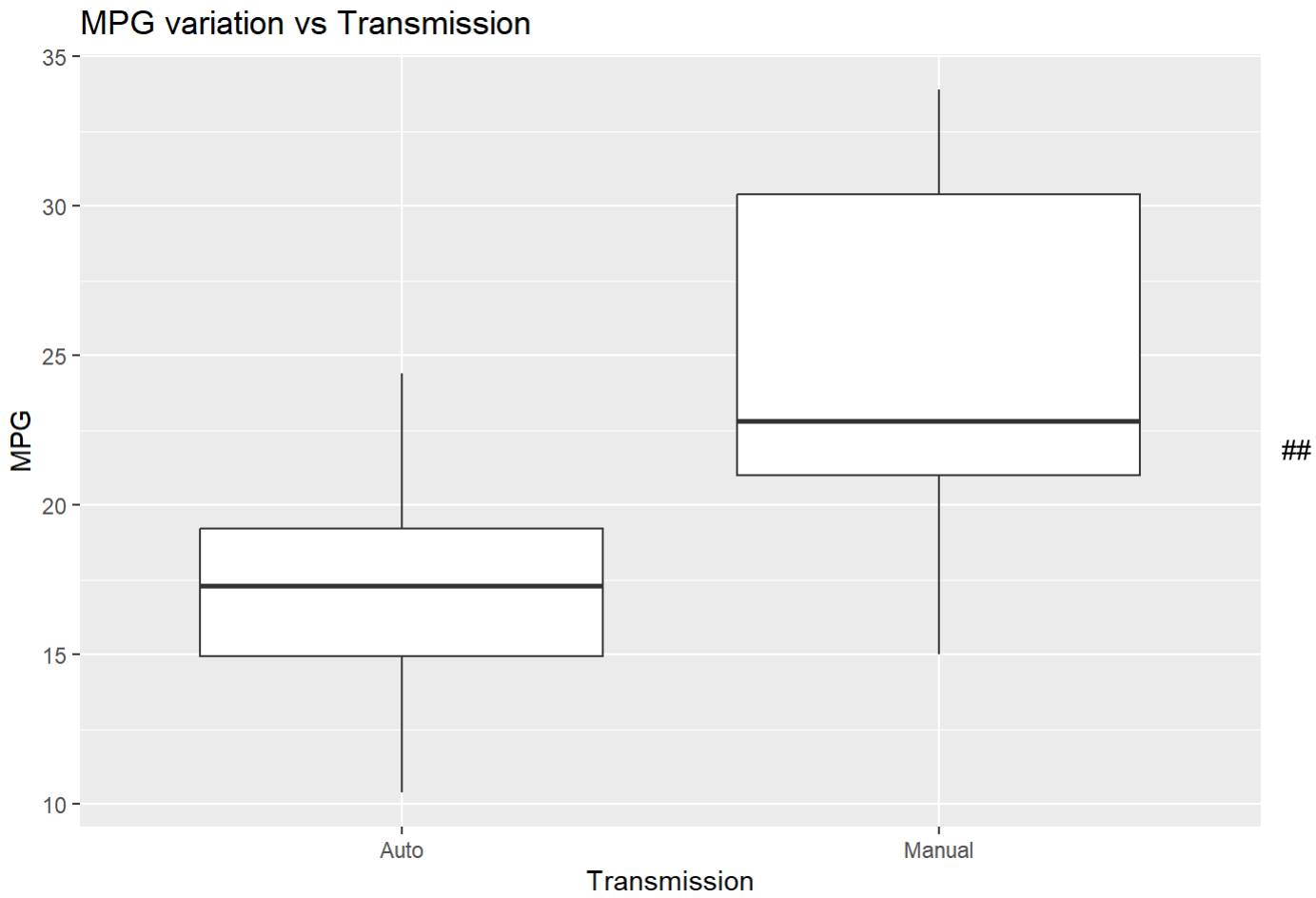


##

Boxplot

Box plots are an effective choice when you need to visualize the distribution of numerical data and detect potential outliers. They are particularly valuable in R for summarizing and comparing the spread and central tendency of data across different categories or groups. Box plots work well with continuous or numeric data types, helping you understand the variability within each category or group, as well as identifying any skewness or extreme values. They are especially handy in scenarios like comparing the distribution of salaries across various job roles, assessing the performance of different products in a sales dataset, or examining the spread of test scores among different student cohorts. Box plots provide a clear and concise way to visualize key statistics such as medians, quartiles, and potential outliers, making them an essential tool in data exploration and analysis.

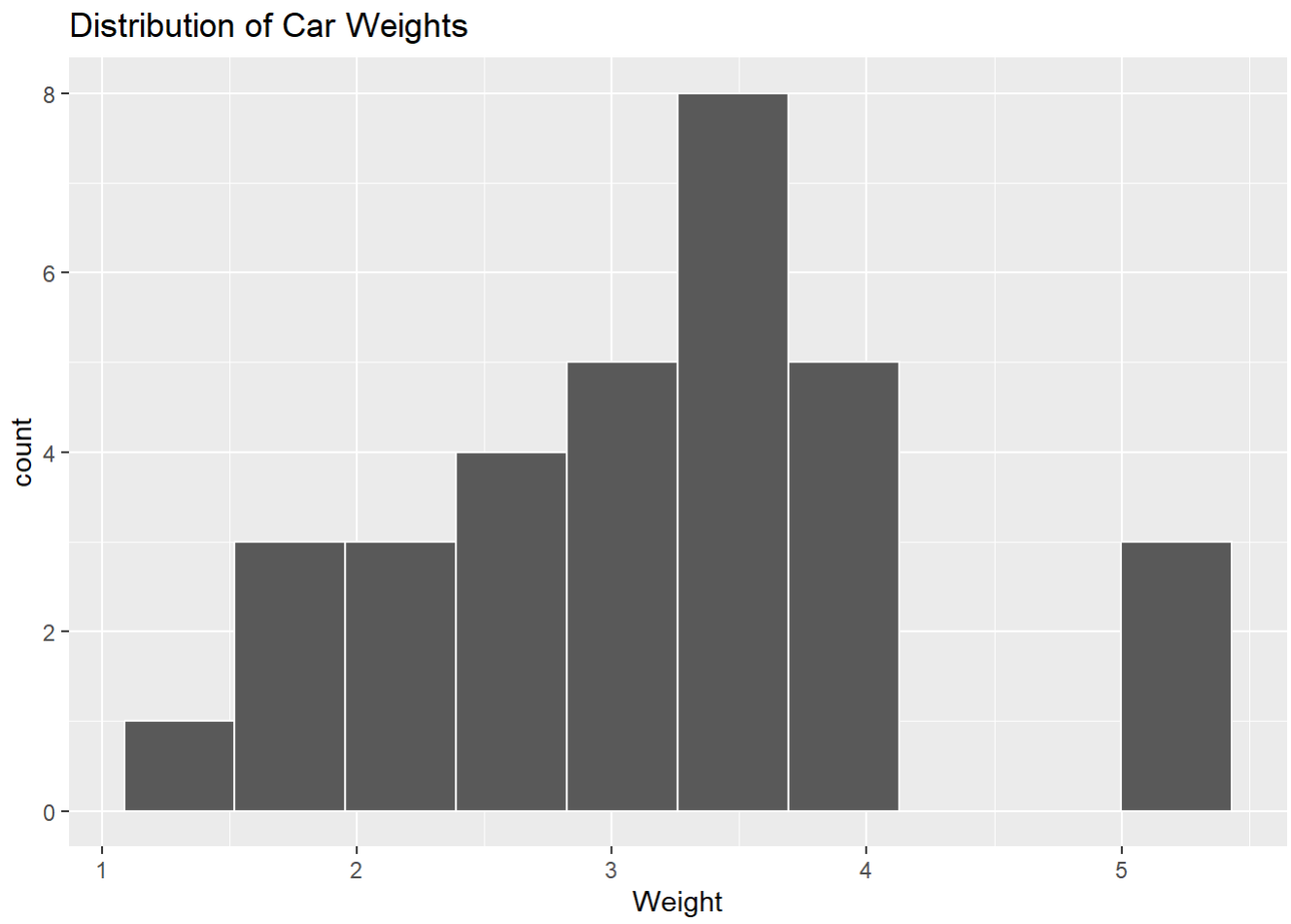
```
# Basic Boxplot
ggplot(cars, aes(x=am, y=mpg)) +
  geom_boxplot() +
  labs(title="MPG variation vs Transmission", x="Transmission", y="MPG")
```



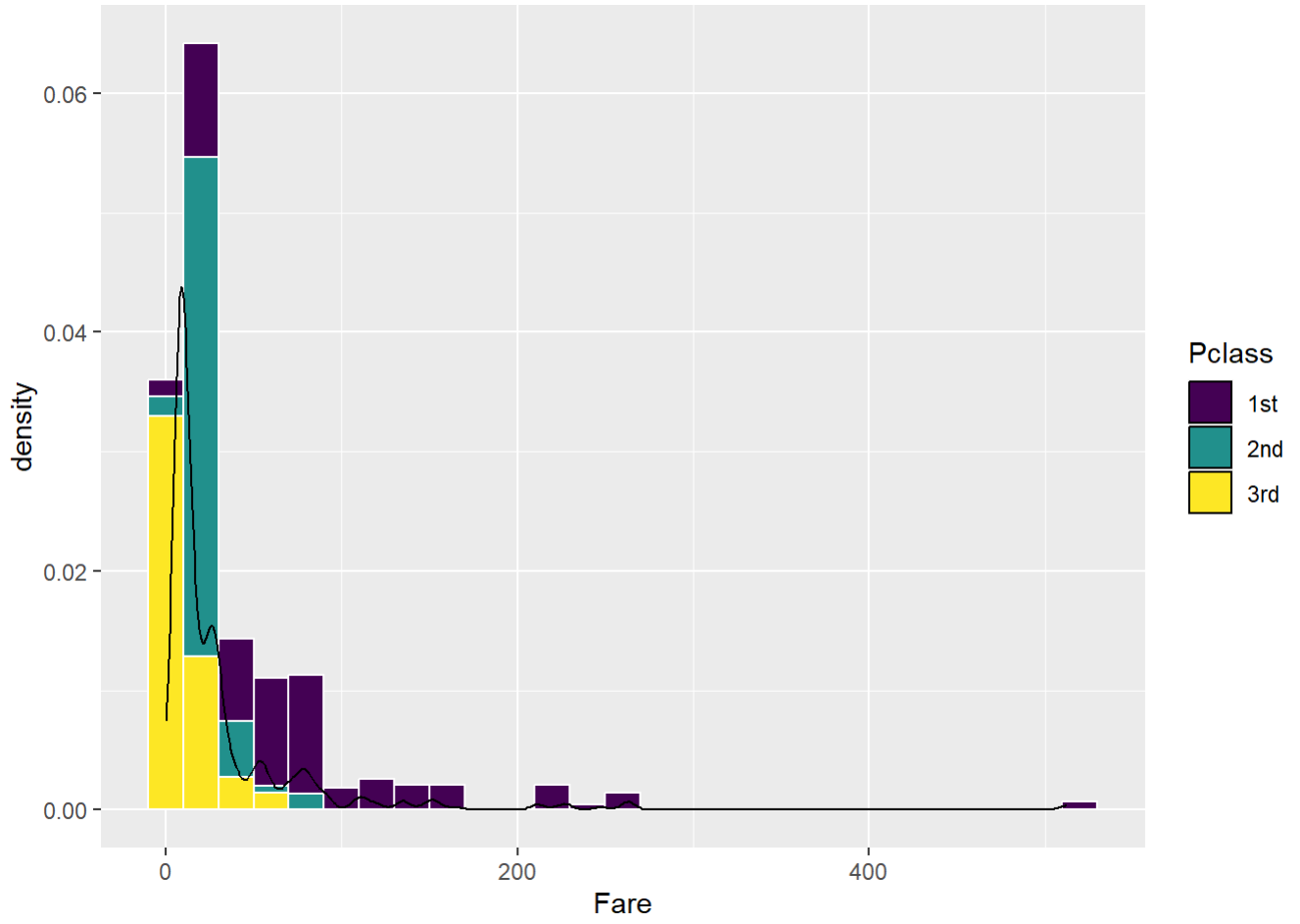
Histograms

Histograms are a fundamental tool in data analysis, especially when you want to understand the distribution of a single variable. In R, you typically use numeric data types for creating histograms. They are particularly valuable when dealing with continuous data, such as age, income, or test scores. Histograms help you visualize the frequency and spread of values within a dataset, making it easy to spot patterns, central tendencies (like mean or median), and the presence of outliers. Whether you're exploring the distribution of customer ages in a marketing campaign or analyzing the income distribution of a population, histograms provide a quick and insightful summary of your data's structure, aiding in decision-making and hypothesis testing.

```
# Basic Histogram
ggplot(cars, aes(x=wt)) +
  geom_histogram(bins=10, color = "white") +
  labs(title="Distribution of Car Weights", x="Weight")
```



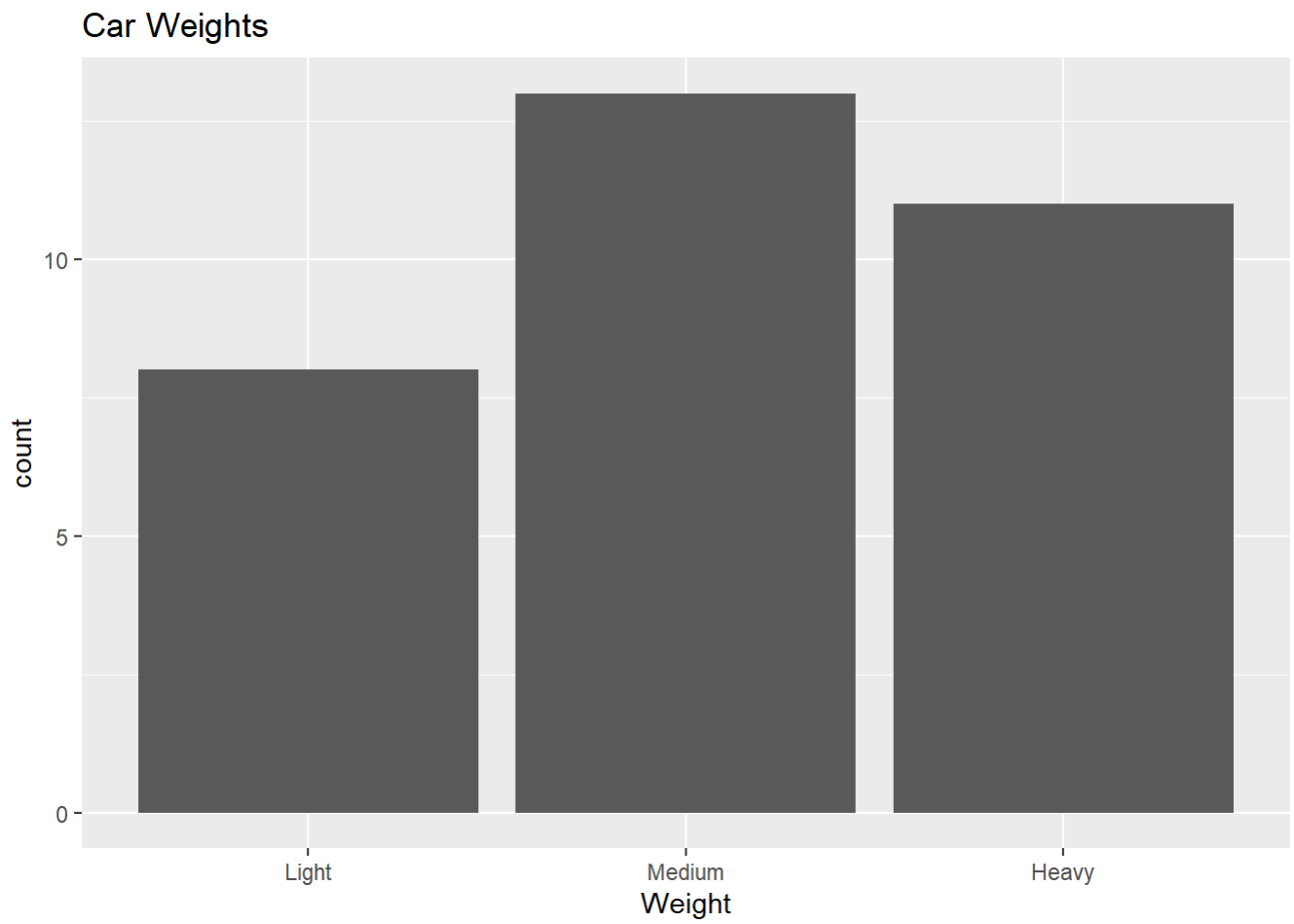

```
# Histogram by density to get them on comparable scales
ggplot(dt1, aes(x = Fare)) +
  geom_histogram(color = "white", binwidth = 20, aes(y = after_stat(density), fill = Pclass)) +
  geom_density()
```



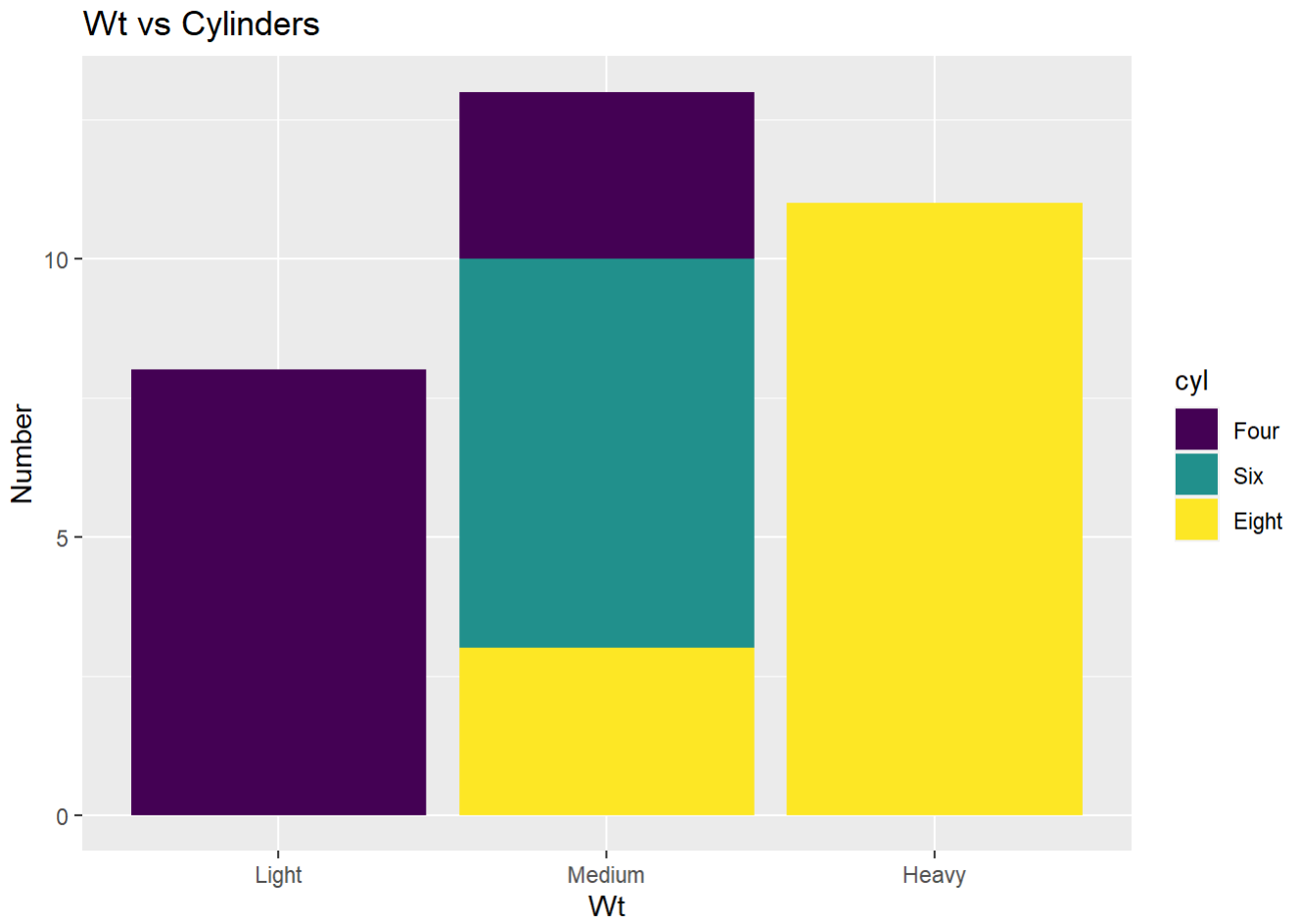
Bar plots

Bar plots are a useful choice when you want to display and compare discrete categories or groups within your data. In R, bar plots are typically used with categorical data types, such as factors or character variables, to visualize the distribution or frequency of each category. They are especially effective for presenting data like survey responses, product sales by category, or demographic information. Bar plots provide a clear and intuitive way to compare the sizes or counts of different categories, making them a valuable tool for making data-driven decisions and identifying trends or disparities among various groups.

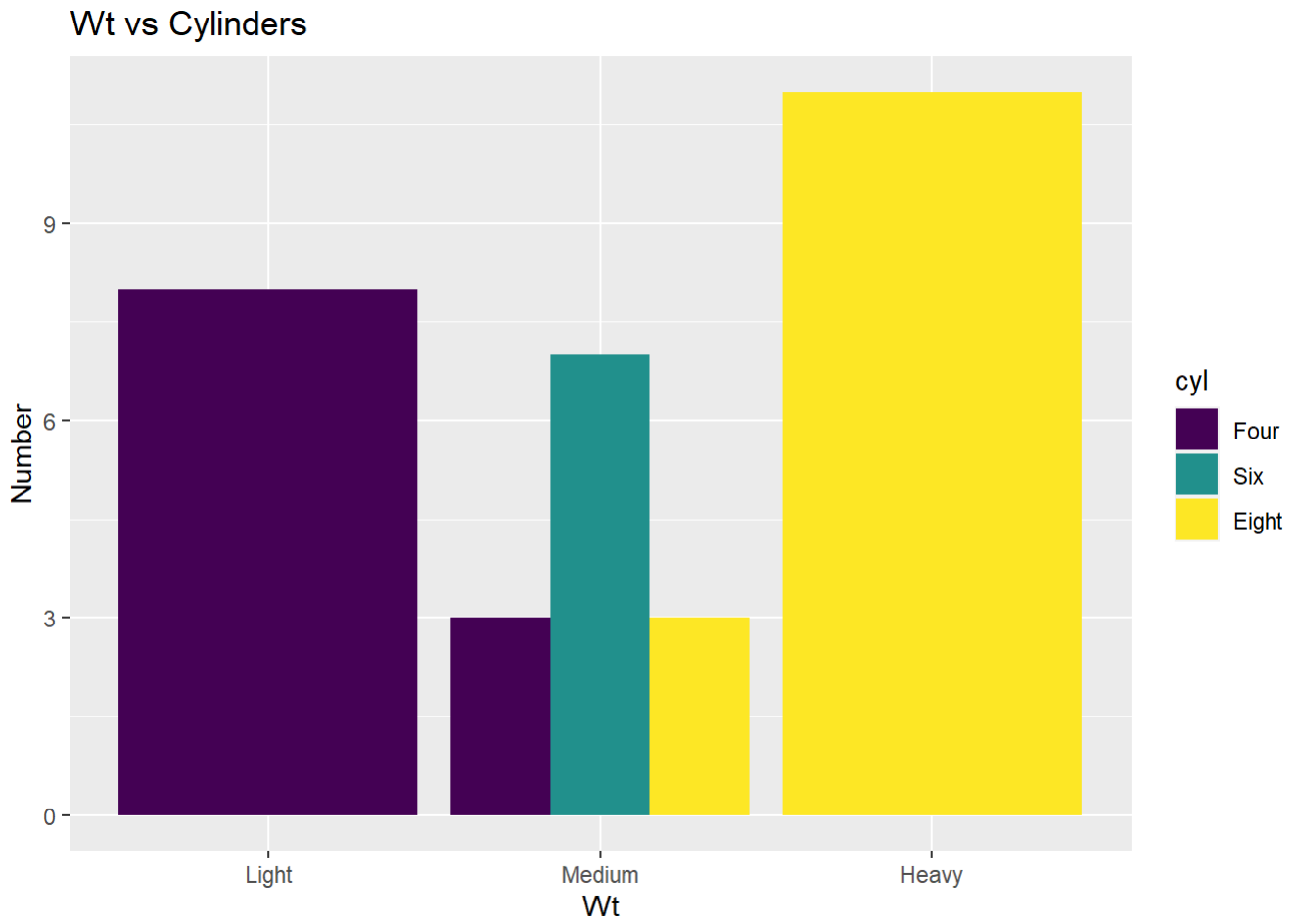
```
# Basic bar plot
ggplot(cars, aes(x=carwt)) +
  geom_bar() +
  labs(title="Car Weights", x="Weight")
```



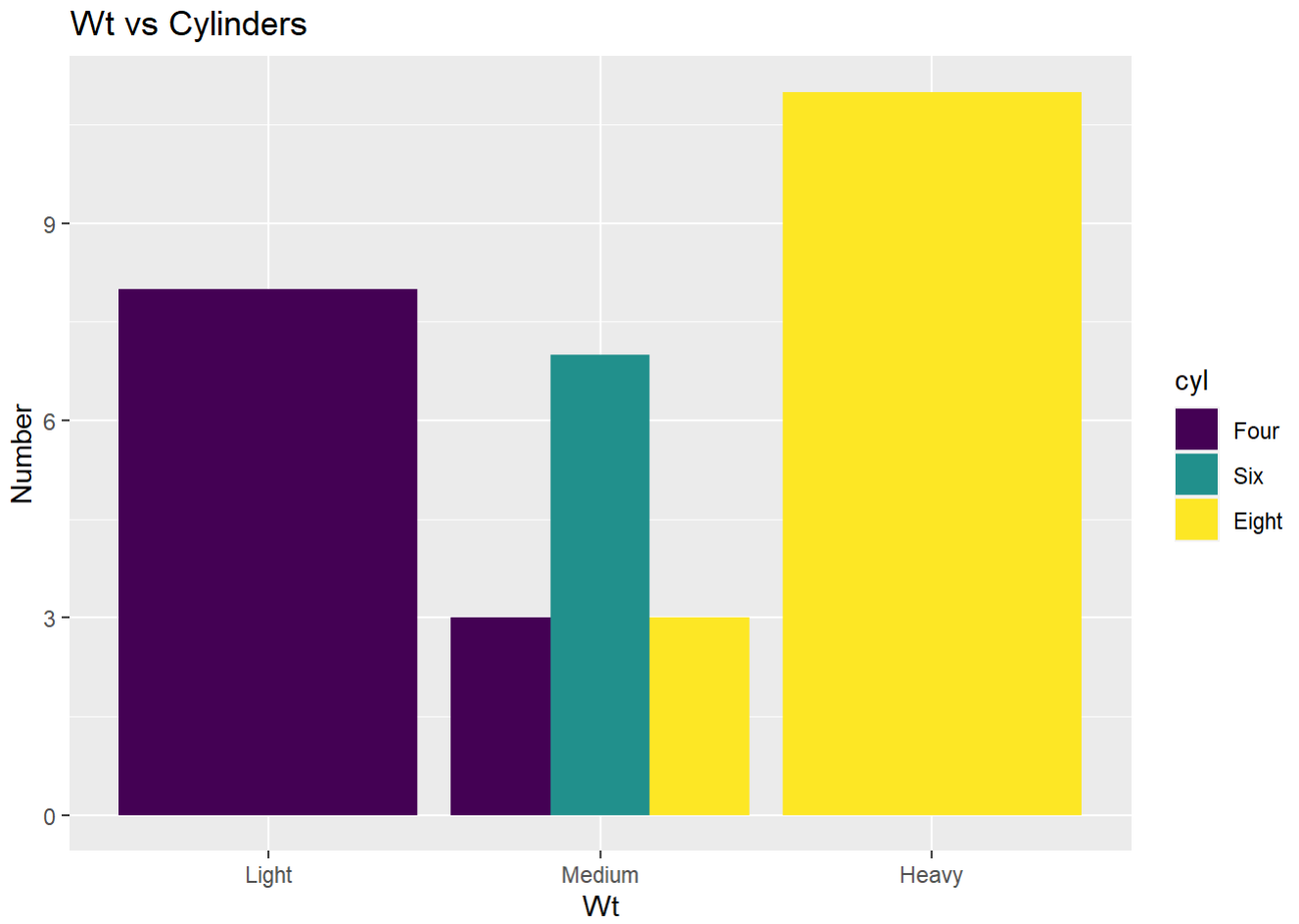

```
# Stacked bar plot
ggplot(cars, aes(x=carwt, fill=cyl)) +
  geom_bar(position="stack") +
  labs(title="Wt vs Cylinders", x="Wt", y ="Number")
```



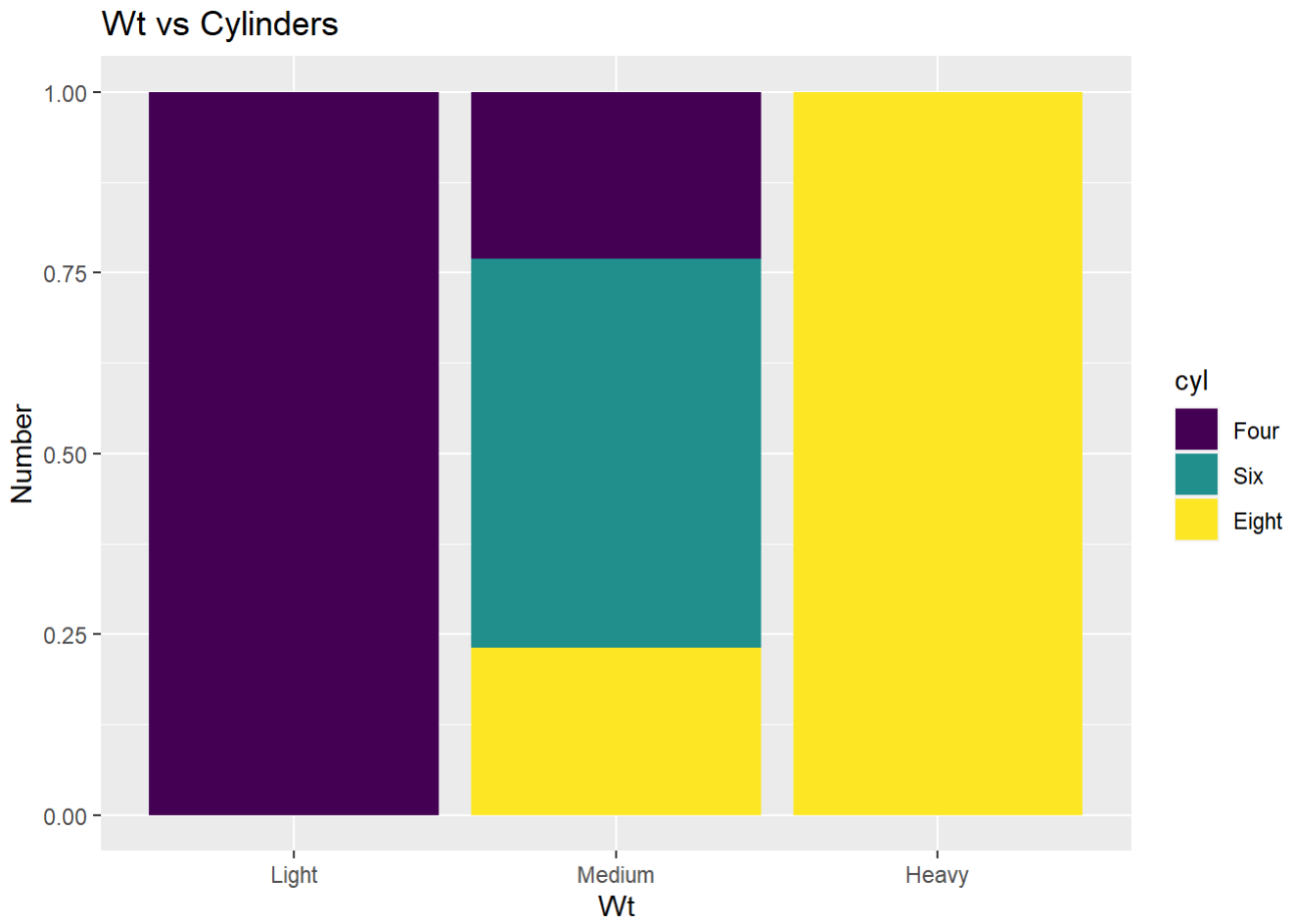

```
# Dodge bar plot
ggplot(cars, aes(x=carwt, fill=cyl)) +
  geom_bar(position="dodge") +
  labs(title="Wt vs Cylinders", x="Wt", y ="Number")
```



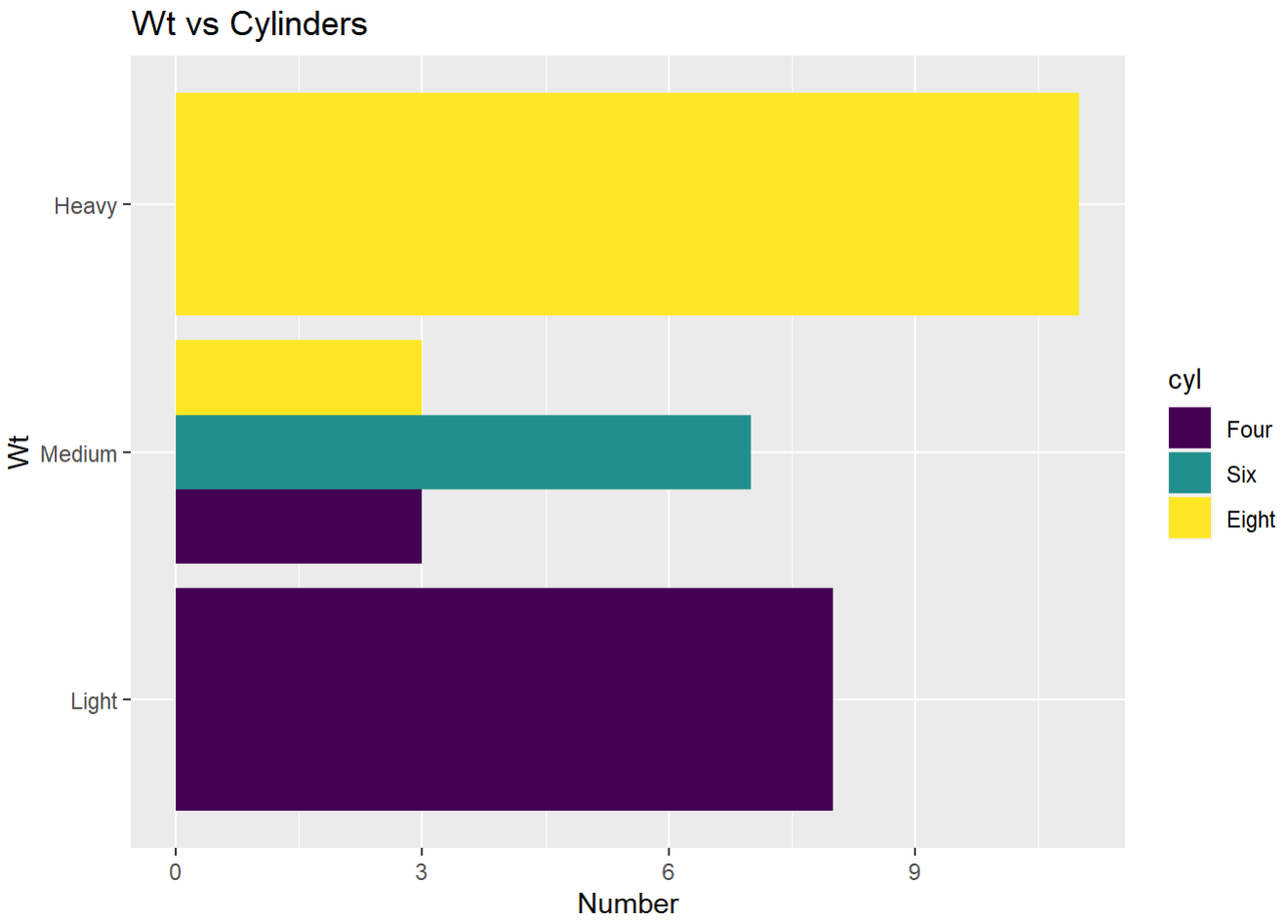

```
#or
ggplot(cars, aes(x=carwt)) +              ##  Aesthetics work both places
  geom_bar(aes(fill=cyl), position="dodge") +
  labs(title="Wt vs Cylinders", x="Wt", y ="Number")
```




```
# Percentage bar plot
ggplot(cars, aes(x=carwt, fill=cyl)) +
  geom_bar(position="fill") +
  labs(title="Wt vs Cylinders", x="Wt", y ="Number")
```

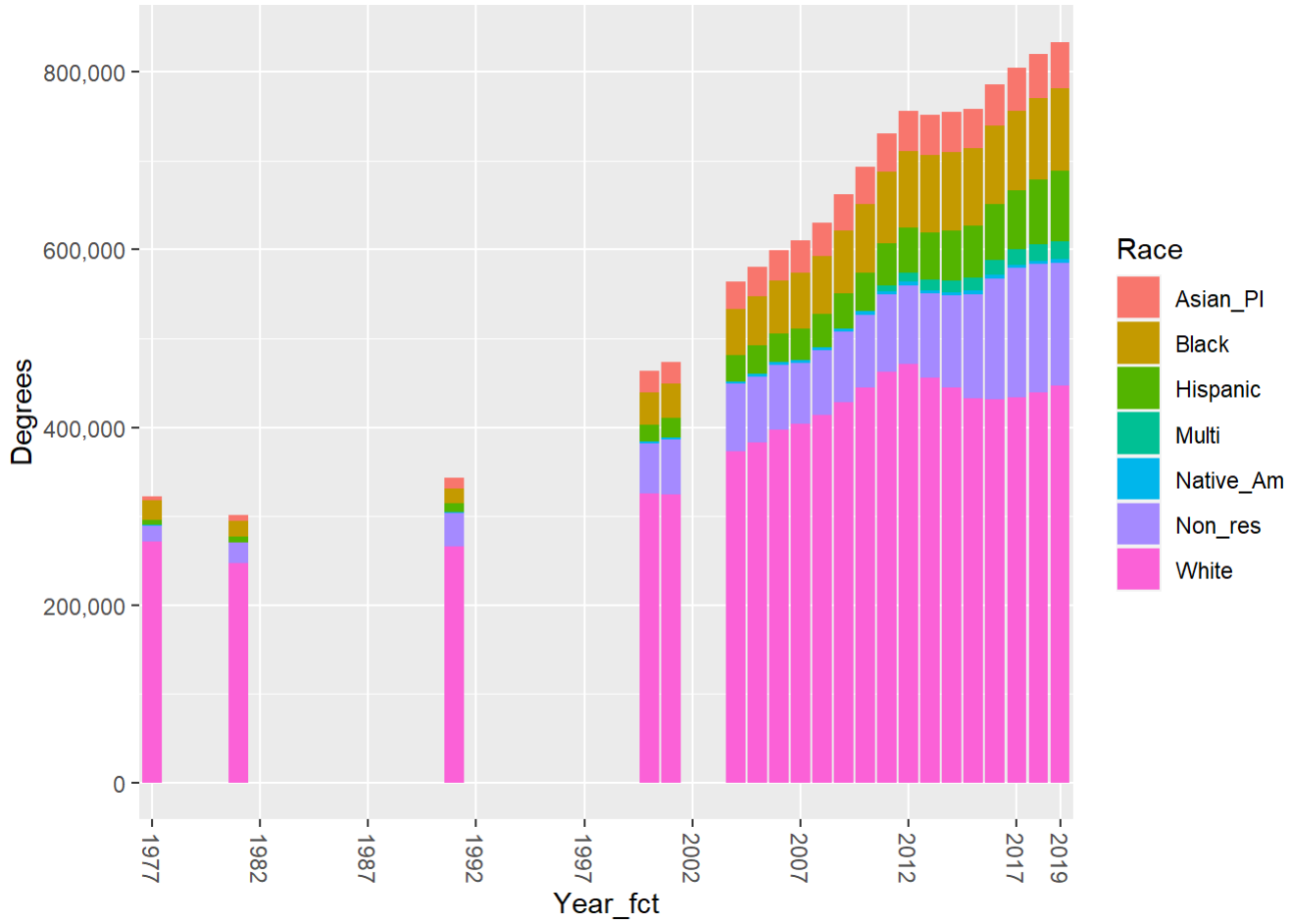



```
# Horizontal bar plot
ggplot(cars, aes(x=carwt)) +
  geom_bar(aes(fill=cyl), position="dodge") +
  coord_flip() +
  labs(title="Wt vs Cylinders", x="Wt", y ="Number")
```




```
# Bar plot adding years that don't have data and cleaning x-axis
min_year <- min(dbr_longer$Year)
max_year <- max(dbr_longer$Year)
ggplot(dbr_longer, aes(x = Year_fct, y = Degrees)) +
  geom_col(aes(fill = Race)) +
  scale_y_continuous(labels = scales::comma) +
  scale_x_discrete(drop = FALSE, breaks = c(seq(min_year, max_year-1, 5), max_year)) +
  theme(axis.text.x = element_text(angle = 270, vjust = 0.5, hjust=1))
```

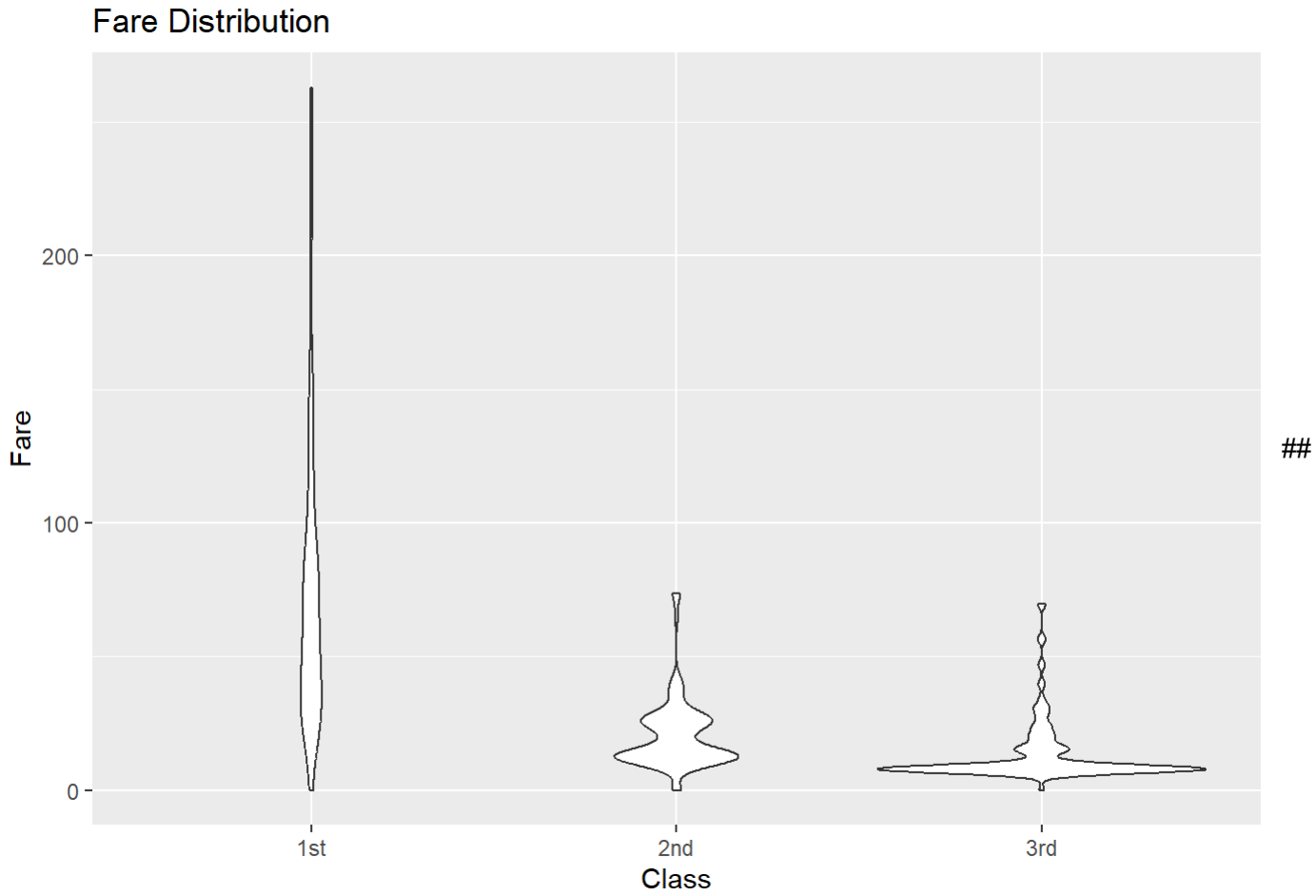
```
## Warning: Removed 12 rows containing missing values (`position_stack()`).
```



Violin plot

Violin plots are an excellent choice when you want to visualize the distribution of a continuous variable or compare distributions across different categories or groups in your data. They are particularly effective with numeric data types, such as integers or floating-point values. Violin plots combine elements of box plots and kernel density plots, allowing you to see not only the summary statistics like medians and quartiles but also the full distribution shape. This makes them valuable for tasks like comparing the distribution of income across different occupations or understanding the spread of test scores among various school districts. Violin plots provide a richer representation of data compared to traditional box plots and are especially useful when dealing with complex datasets where multiple factors need to be considered simultaneously.

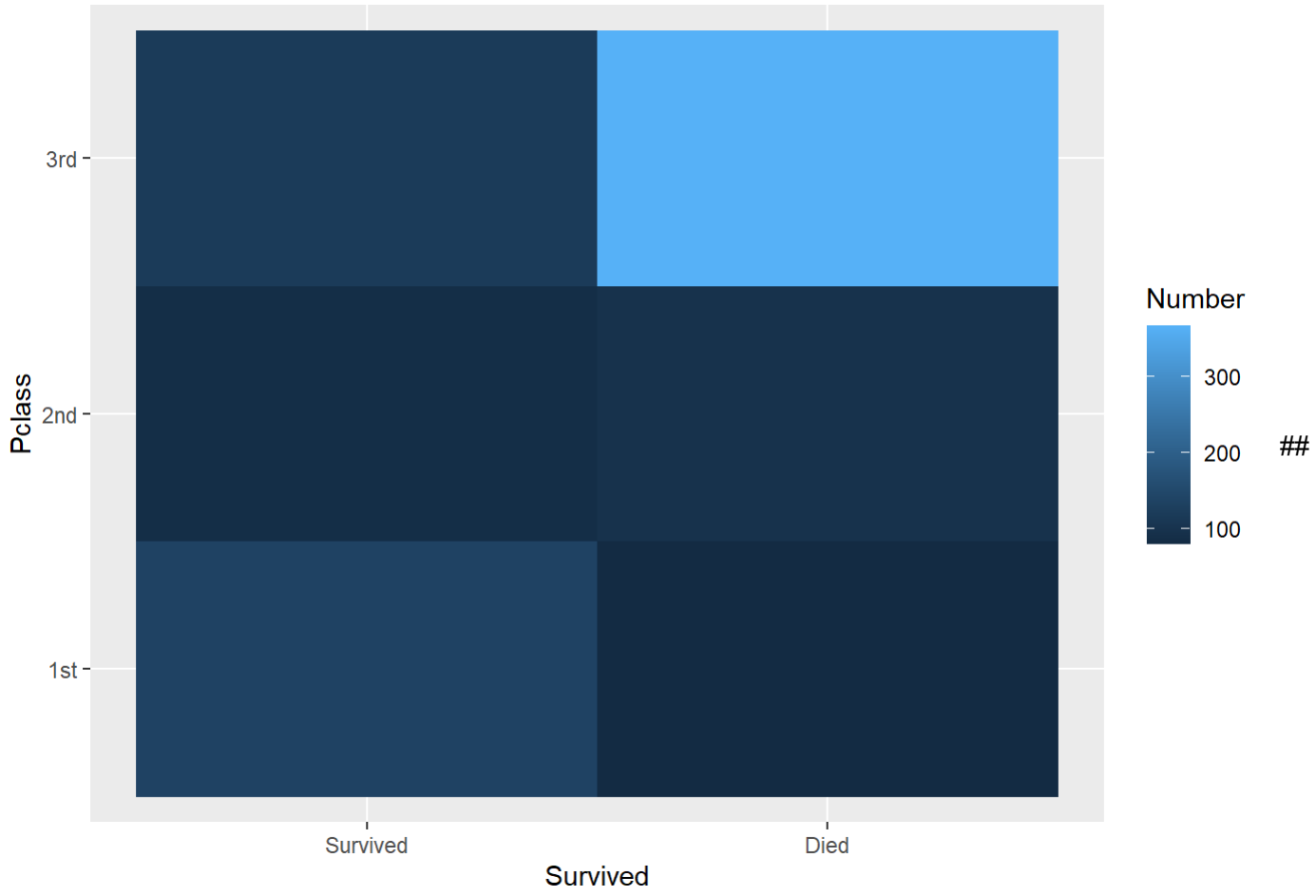
```
# Basic violin plot
ggplot(dt1a, aes(x = Pclass, y = Fare)) +
  geom_violin() +
  labs(title = "Fare Distribution", x = "Class", y = "Fare")
```



Tile plot

Tile plots are an excellent choice when you need to visualize the distribution of data across two categorical variables. These plots are particularly effective when working with discrete data types, such as factors or character variables. By representing data as tiles or rectangles, you can quickly discern patterns, associations, or variations in your data, especially when you have a large number of categories to compare. Tile plots are valuable in scenarios like visualizing the frequency of product sales across different regions and time periods or assessing user engagement on a website by categorizing it into user types and actions. They provide a comprehensive and intuitive way to explore relationships within categorical data, making it easier to derive actionable insights.

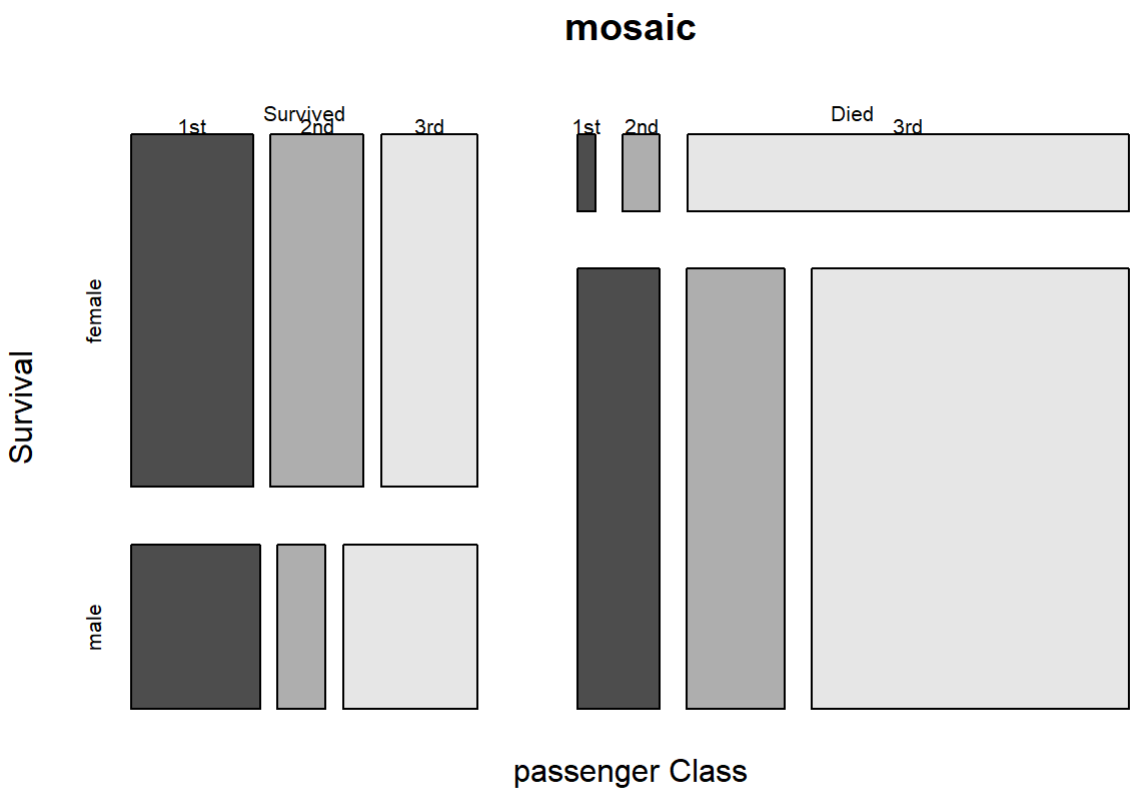
```
#Basic tile plot
ggplot(dt4, mapping = aes(x = Survived, y = Pclass)) +
  geom_tile(mapping = aes(fill = Number))
```



Mosaic plot

Mosaic plots are an excellent choice when you need to visualize and understand the relationship between two categorical variables in your data, making them a valuable tool in exploratory data analysis in R. They are particularly suited for nominal or ordinal data types, where you want to assess how the proportions of one category change within the levels of another. By dividing the plot into rectangles whose sizes are proportional to the joint frequencies of the categories, mosaic plots allow you to spot associations, dependencies, or deviations from independence between the two variables. These plots are especially useful for examining data in fields like market research (comparing customer preferences across demographics) or social sciences (studying the relationship between education and income levels) where categorical data plays a crucial role in analysis and decision-making.

```
# Mosaic plot
mosaicplot(data=dt52, ~ Survived + Sex + Pclass, color=TRUE, main="mosaic",
    xlab="passenger Class", ylab="Survival")
```

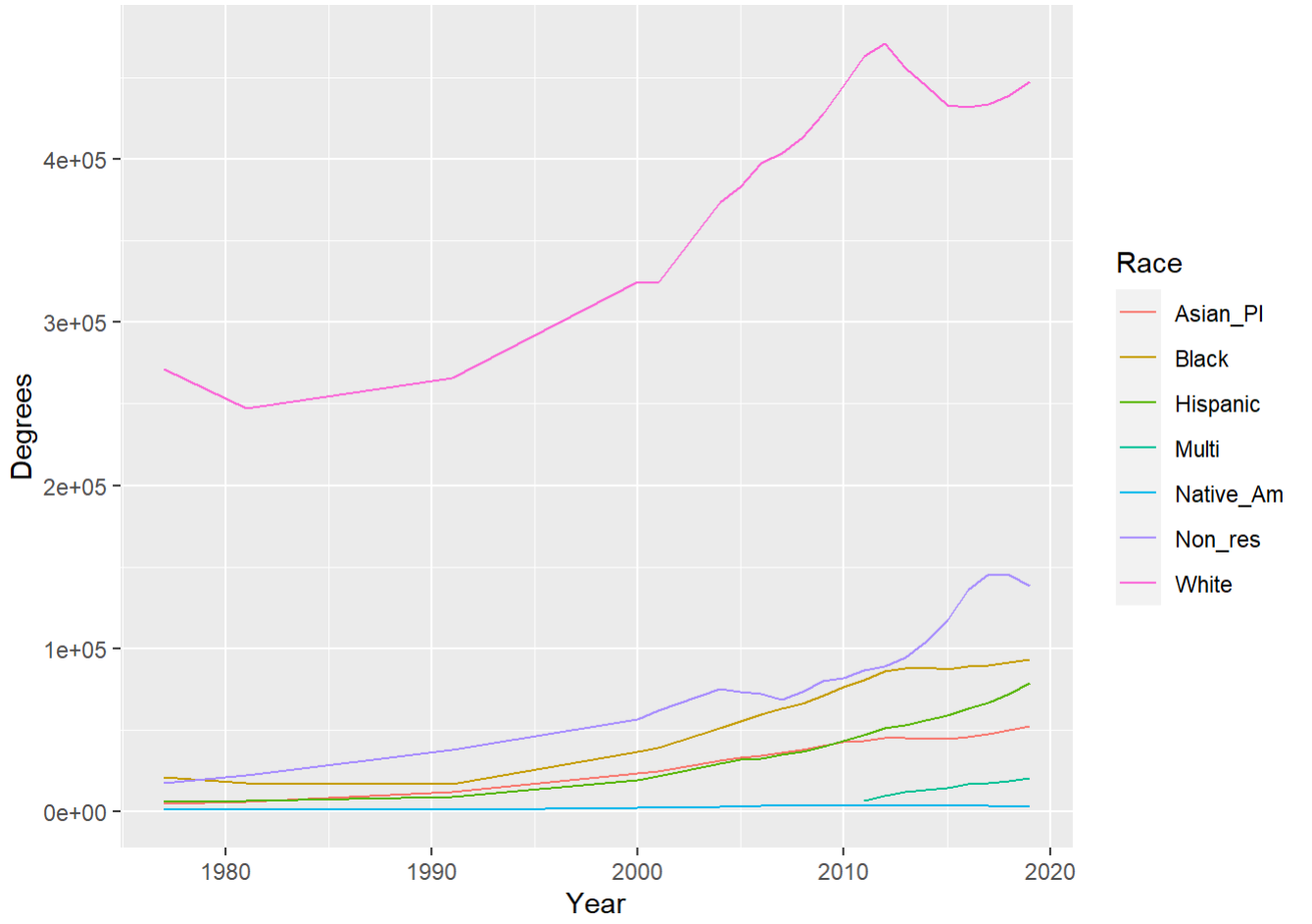


Line plot

Line plots, also known as line charts or time series plots, are valuable when you need to visualize and understand trends, patterns, or changes in data over time. They are typically used with time-series data, where the x-axis represents time and the y-axis represents a numeric variable. Line plots are excellent for illustrating trends in continuous data types, such as numerical values, percentages, or counts, across different time points. They are particularly useful for monitoring stock prices, tracking sales performance over months, or examining temperature variations throughout the year. Line plots in R are an essential tool for time-series analysis, enabling you to make informed decisions and predictions based on historical data trends.

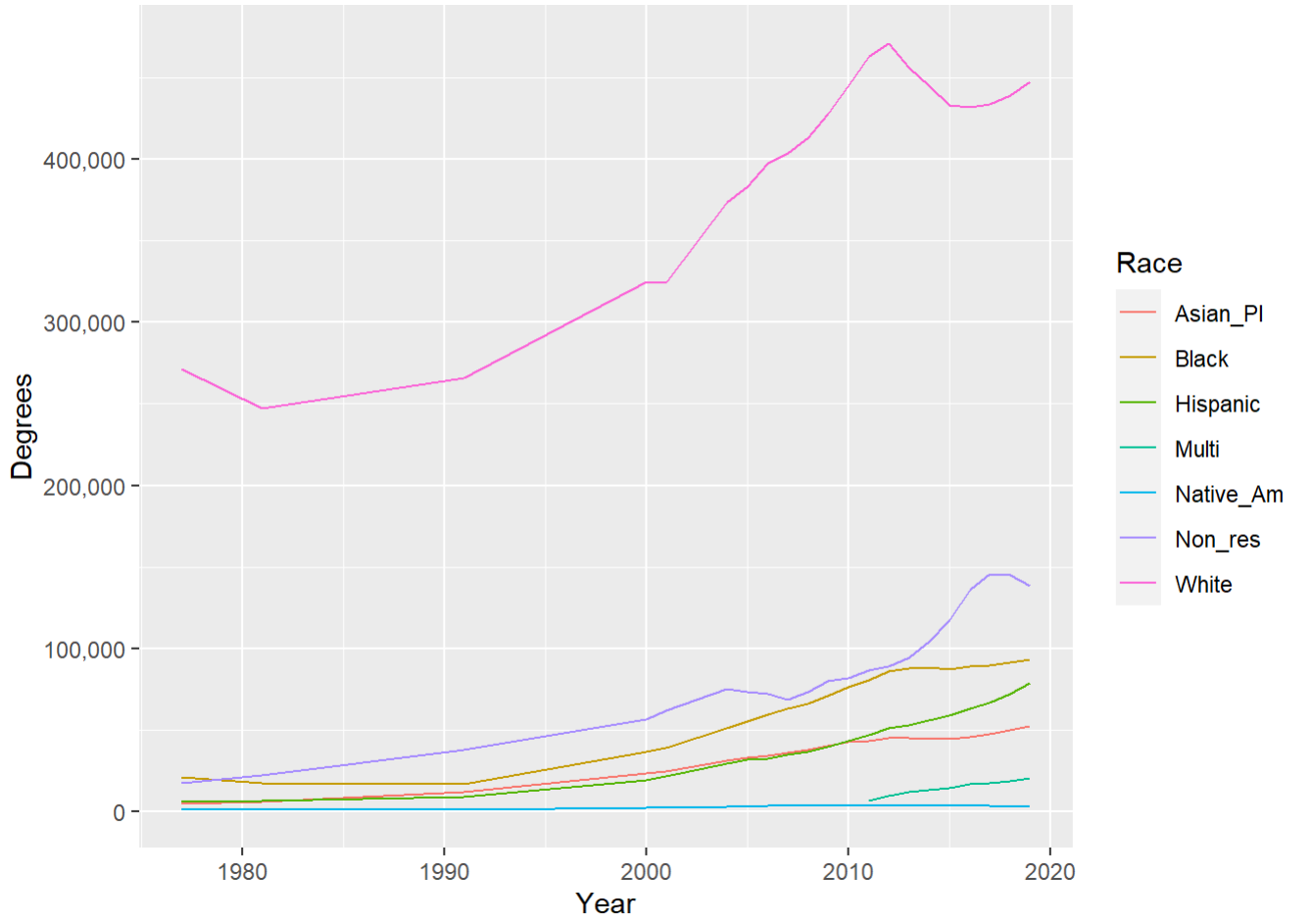
```
# Basic line plot
ggplot(dbr_longer, aes(x = Year, y = Degrees)) +
  geom_line(aes(col = Race))
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```



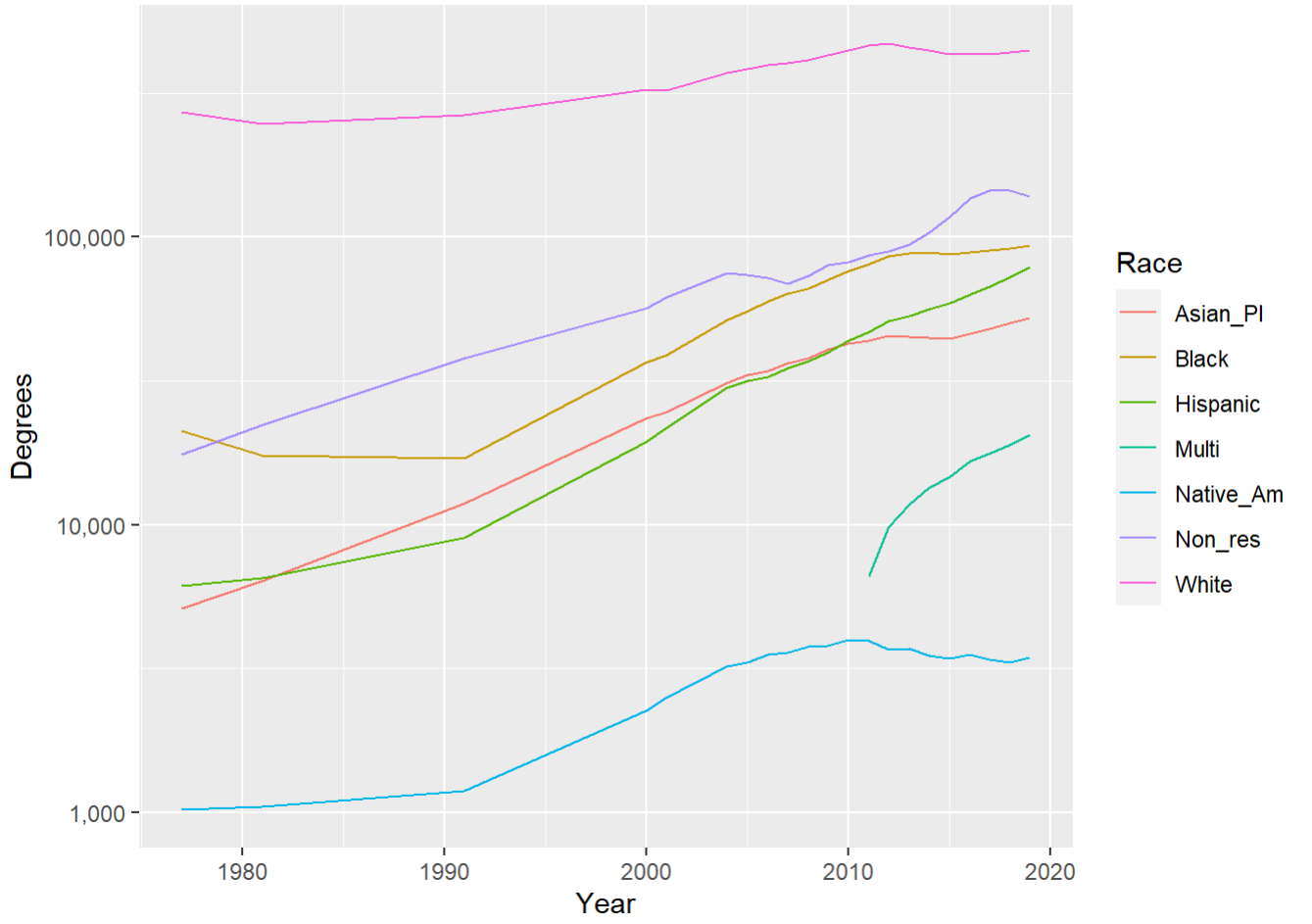

```
# Removing scientific notation
ggplot(dbr_longer, aes(x = Year, y = Degrees)) +
  geom_line(aes(col = Race)) +
  #This specifies the format of the labels on the y-axis
  scale_y_continuous(labels = scales::comma)
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```



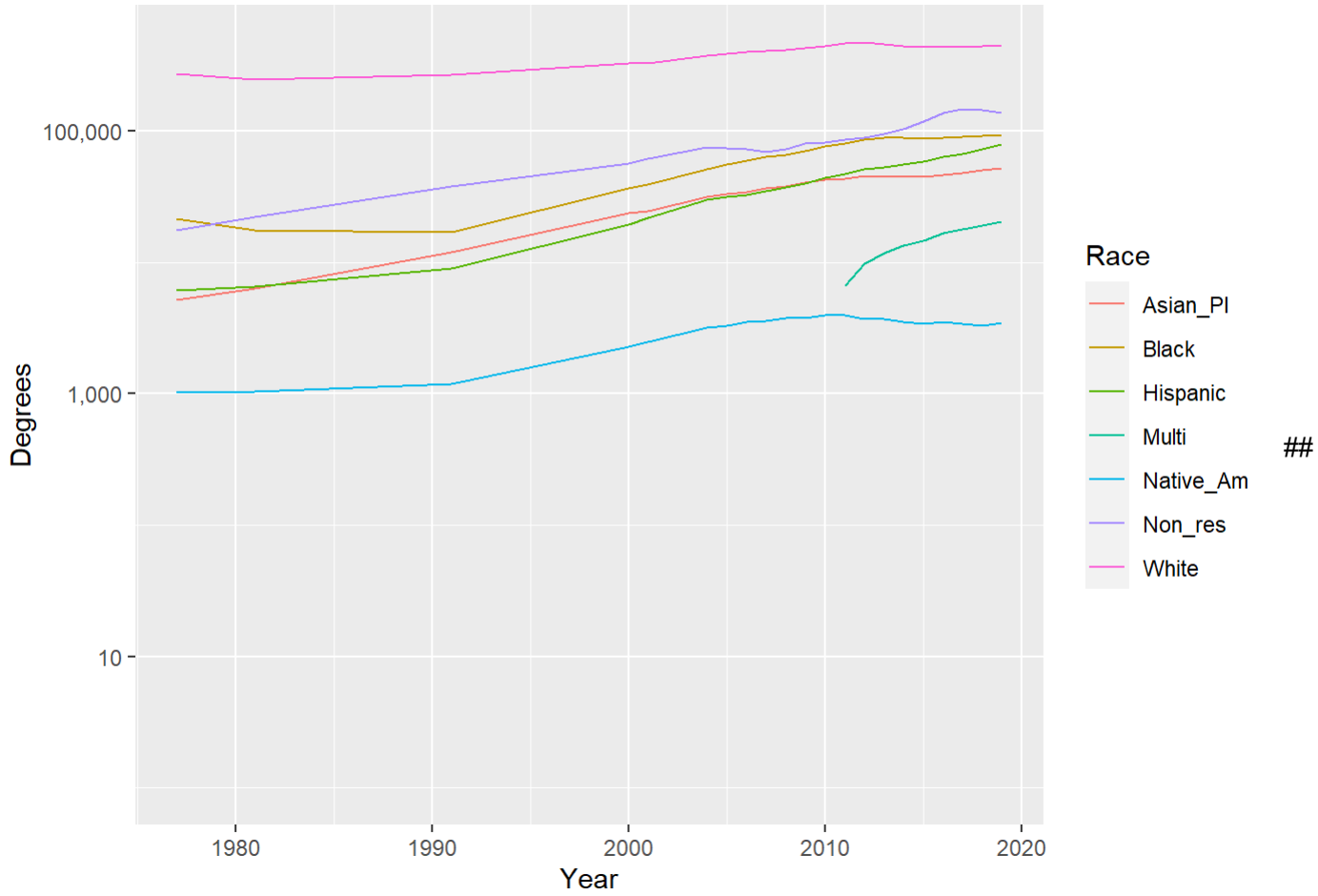

```
# Log scale
ggplot(dbr_longer, aes(x = Year, y = Degrees)) +
  geom_line(aes(col = Race)) +
  scale_y_log10(labels = scales::comma)
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```




```
# Setting the range of the graph
ggplot(dbr_longer, aes(x = Year, y = Degrees)) +
  geom_line(aes(col = Race)) +
  #Can't use 0 as a limit for a log scale
  #So, let's start at 1
  scale_y_log10(labels = scales::comma, limits = c(1, NA))
```

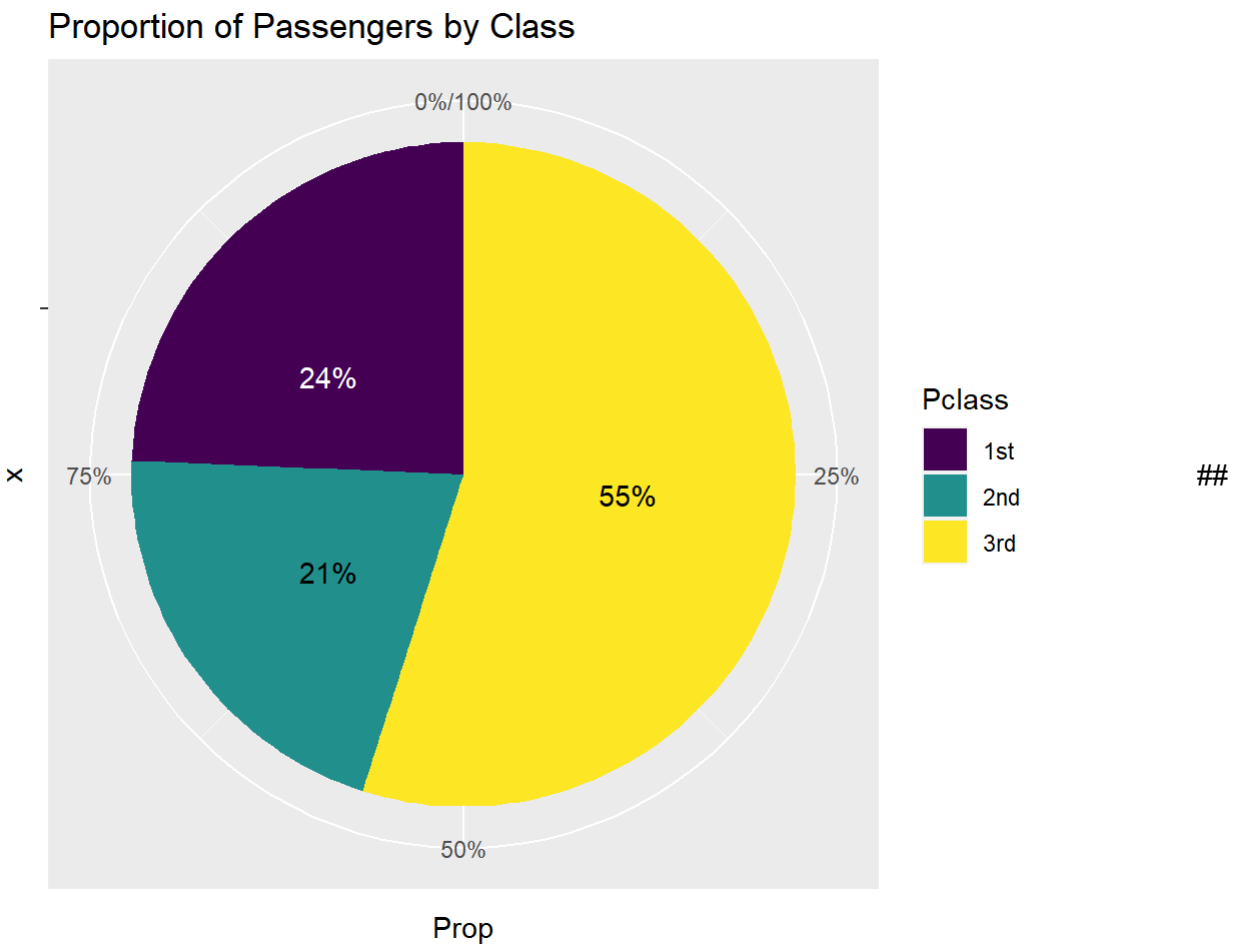
```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```



Pie chart

Pie charts are a suitable choice when you want to represent the composition or distribution of categorical data in a visually intuitive way. In R, you typically use factors or character vectors to create pie charts. This type of chart is most effective when you have a small number of distinct categories and want to show the proportions or percentages of each category relative to the whole. For example, you can use a pie chart to display the market share of different product categories in a company's portfolio or the distribution of survey responses by category. However, it's important to be cautious when using pie charts with a large number of categories, as they can become cluttered and less informative. In such cases, other visualization techniques like bar charts or treemaps might be more suitable.

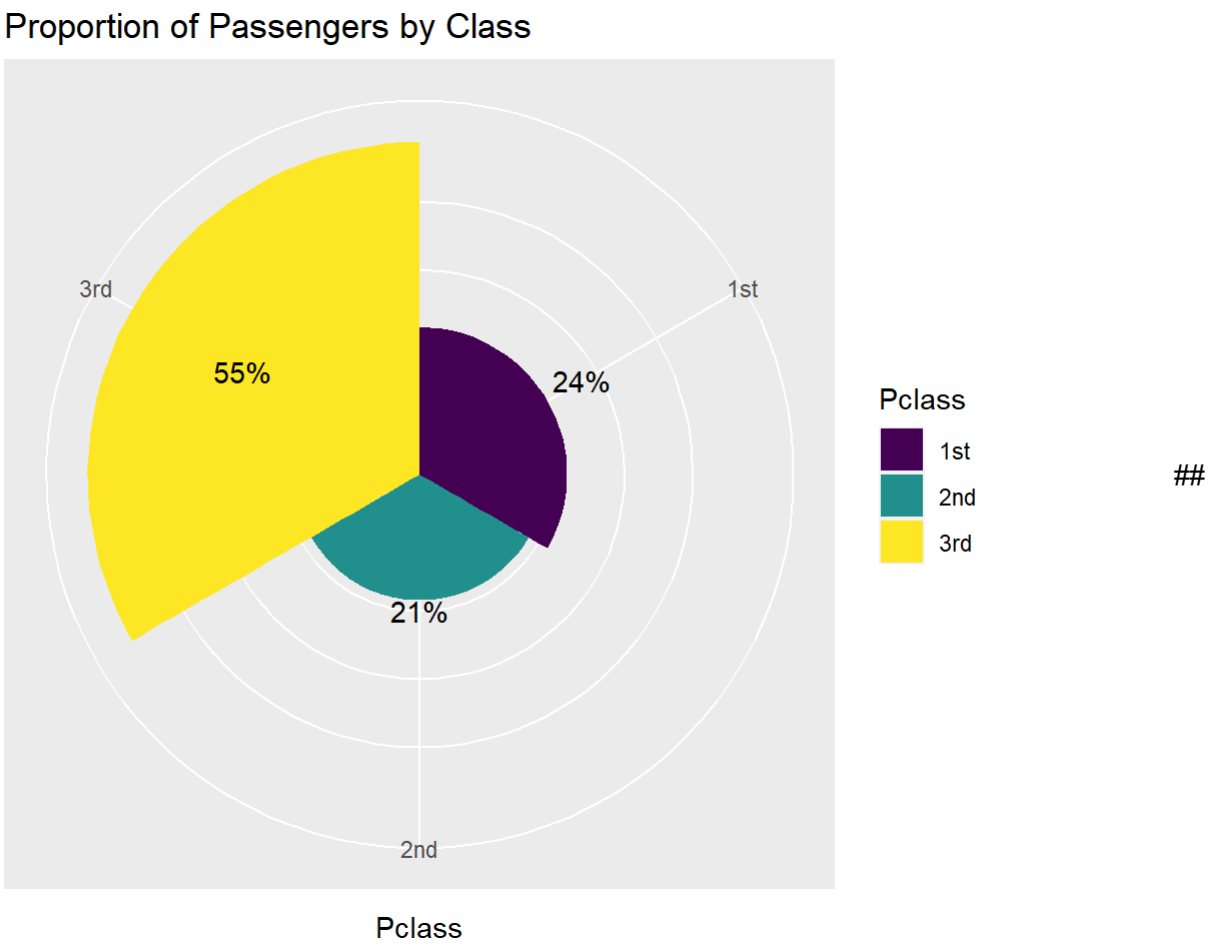
```
# Basic pie chart
ggplot(dt2, aes(x = "", y = Prop, fill = Pclass)) +
  geom_col(width = 1) +
  annotate("text", x = 1, y = c(.27, .65, .85), label = c("55%", "21%", "24%"),
           color = c("black", "black", "white")) +
  coord_polar("y") +
  labs(title = "Proportion of Passengers by Class") +
  scale_y_continuous(labels = scales::percent)
```



Rose plot

Rose plots, also known as polar area diagrams or wind rose diagrams, are valuable for visualizing directional data, making them particularly useful in fields like meteorology, environmental science, and geology. In R, you typically use rose plots when you want to understand the distribution and frequency of data points across different directions or angles. These plots are ideal for visualizing data with circular or directional patterns, such as wind speed and direction, ocean currents, or animal migration routes. To create a rose plot in R, you often work with circular data types, representing angles or compass directions, and the frequency or magnitude associated with each direction. Rose plots effectively convey the predominant directions and strengths of patterns within your dataset, making them a powerful tool for spatial and directional analysis.

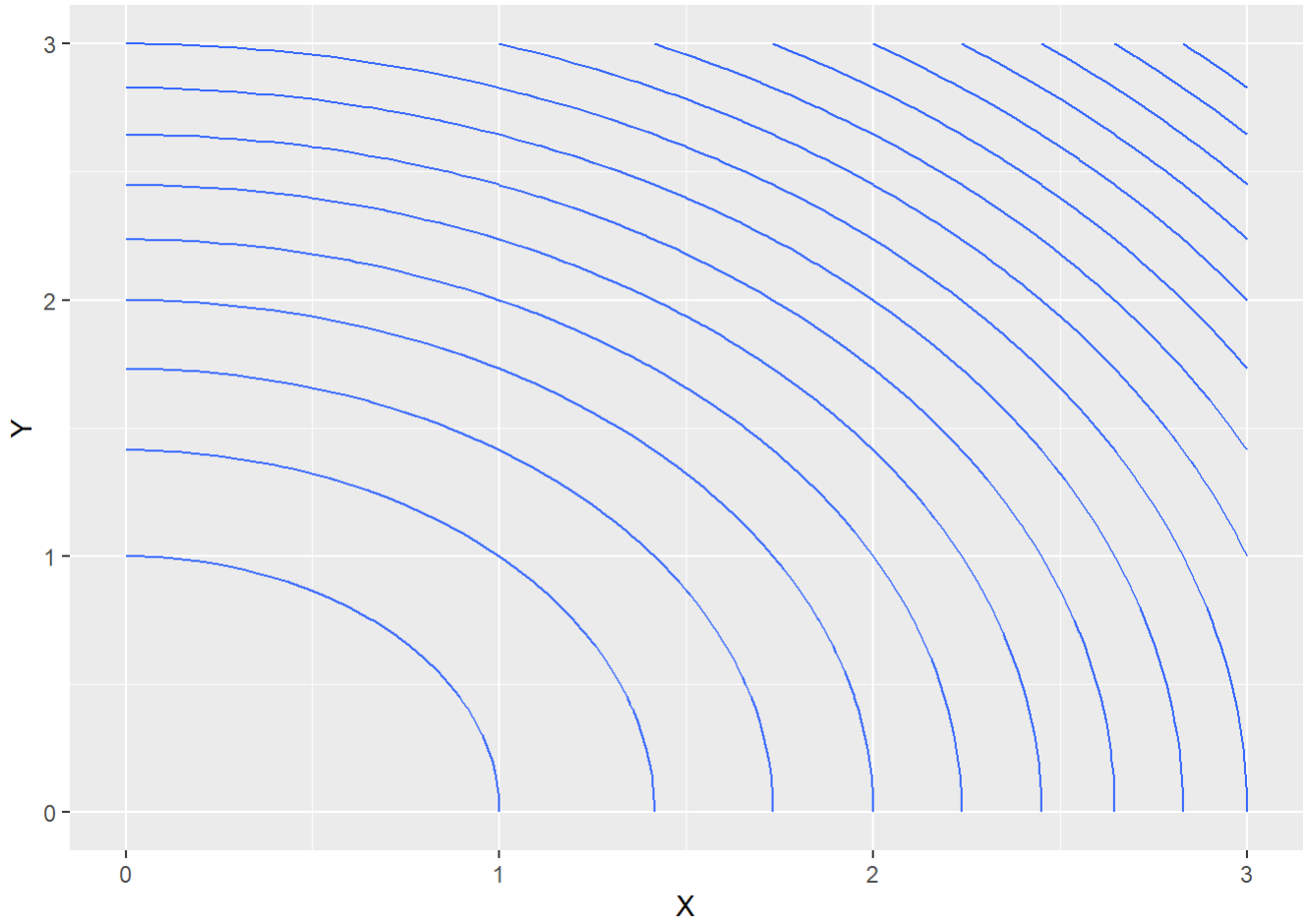
```
# Rose plot
ggplot(dt3, aes(x = Pclass)) +
  geom_bar(aes(fill=Pclass), width = 1) +
  annotate("text", x = c(1,2,3), y=c(275, 200, 300), label = c("24%", "21%", "55%")) +
  coord_polar() +
  labs(title = "Proportion of Passengers by Class") +
  theme(axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        axis.title.y = element_blank())
```



Density plot

Density plots are a valuable choice when you want to visualize the distribution of a continuous variable in R. These plots are especially useful for understanding the shape, spread, and modes within your data. Typically, density plots are employed when dealing with numeric data types, such as integers or real numbers. They help you identify patterns like bimodality, skewness, or multimodality in your data distribution. Density plots provide a smooth representation of the data's probability density function, making it easier to spot trends and variations compared to traditional histograms. This technique is handy for tasks like examining income distributions, analyzing exam scores, or assessing the age distribution in a population, allowing you to draw insights from data with continuous, numeric values.

```
# 2D Density plot
x2 <- seq(0, 3, .05)
y2 <- seq(0, 3, .05)
df3 <- expand.grid(x2, y2) |>
  rename(X = Var1, Y = Var2) |>
  mutate(Z = X^2 + Y^2)
ggplot(df3, aes(x = X, y = Y, z = Z)) +
  geom_contour(binwidth = 1)
```


3
2
1
0
Y
0
1
2
3
X