Data Visualization Analysis

Sean

2024-12-20

Table of contents

1 Introduction 2

2 Preparation 2

3 Datasets 2

4 Bar chart 3

5 Bar chart with color 4

6 Line chart 7

7 Histogram 8

8 Correlation chart 9

9 Correlation chart: Color by group 10

10 Multigroup histogram 11

11 Density chart 13

12 Histogram and Density chart 14

13 Box plot 15

14 Adding Title, Captions and Axis Labels 16

15 Theme 17

16 Adding and Removing Legend 19

1 Introduction

This tutorial is designed to help you learn data visualization analysis by providing simple and useful information in a way that is easy to follow and understand.

2 Preparation

In order to draw a chart, we need to include the required packages for visualization and dataset. For example, `ggplot2` package is for drawing charts and `gcookbook` is for using `pg_mean` dataset.

```
library(ggplot2)
library(gcookbook)
```

3 Datasets

Before we start with this tutorial, We will be discussing all the datasets we are going to use 1. `pg_mean` dataset. The dataset has two columns: `group`, `weight`.

```
pg_mean
```

This dataset compares the weight across three groups:

- `ctrl`: Control group (baseline, weight = 5.032).
- `trt1`: Treatment 1 group (weight = 4.661).
- `trt2`: Treatment 2 group (weight = 5.526).

2. `BOD`, The dataset has two columns: `Time` and `demmand`

```
BOD
```

This dataset compares Oxygen demand over time - `Time`: Time needed - `demmand`: Biochemical Oxygen demanded

3. `iris`, The dataset has 5 columns:

```
iris
```

- `Sepal.Length`: Length of the sepal (in cm).
- `Sepal.Width`: Width of the sepal (in cm).

- `Petal.Length`: Length of the petal (in cm).
- `Petal.Width`: Width of the petal (in cm).
- `Species`: The species of the flower (`setosa`, `versicolor`, or `virginica`)

4. `birthwt`, The dataset has 10 columns:

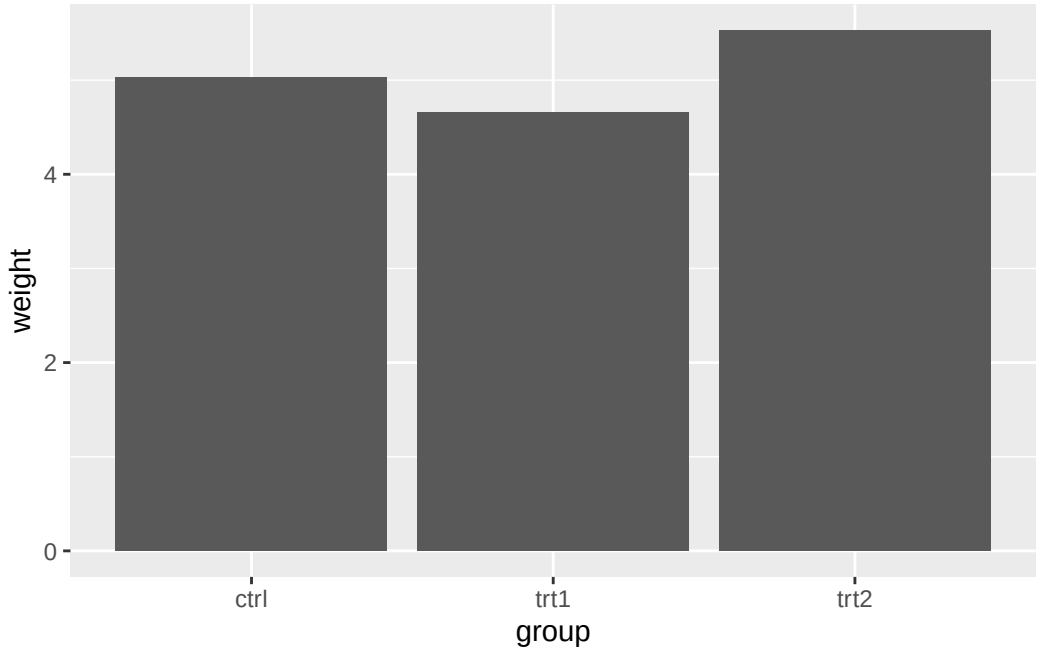
```
birthwt
```

- `low`: Indicator of birth weight less than 2.5 kg (0 = `no`, 1 = `yes`).
- `age`: Mother's age (in years).
- `lwt`: Mother's weight at last menstrual period (in pounds).
- `race`: Mother's race (1 = `White`, 2 = `Black`, 3 = `Other`).
- `smoke`: Smoking status during pregnancy (0 = `no`, 1 = `yes`).
- `ptl`: Number of previous premature labors.
- `ht`: History of hypertension (0 = `no`, 1 = `yes`).
- `ui`: Presence of uterine irritability (0 = `no`, 1 = `yes`).
- `ftv`: Number of physician visits during the first trimester (0, 1, 2, or more).
- `bwt`: Birth weight of the baby (in grams).

4 Bar chart

In this section, we will draw a bar chart using `pg_mean` dataset.

```
ggplot(pg_mean, aes(x = group, y = weight)) +
  geom_col()
```



It initializes a ggplot with the dataset `pg_mean`.

`aes(x = group, y = weight)` specifies the aesthetics:

- `x = group`: Assign the `group` variable to the x-axis (categorical data, such as `ctrl, trt1, trt2`).
- `y = weight`: Assign the `weight` variable to the y-axis (numerical data).

`geom_col()`:

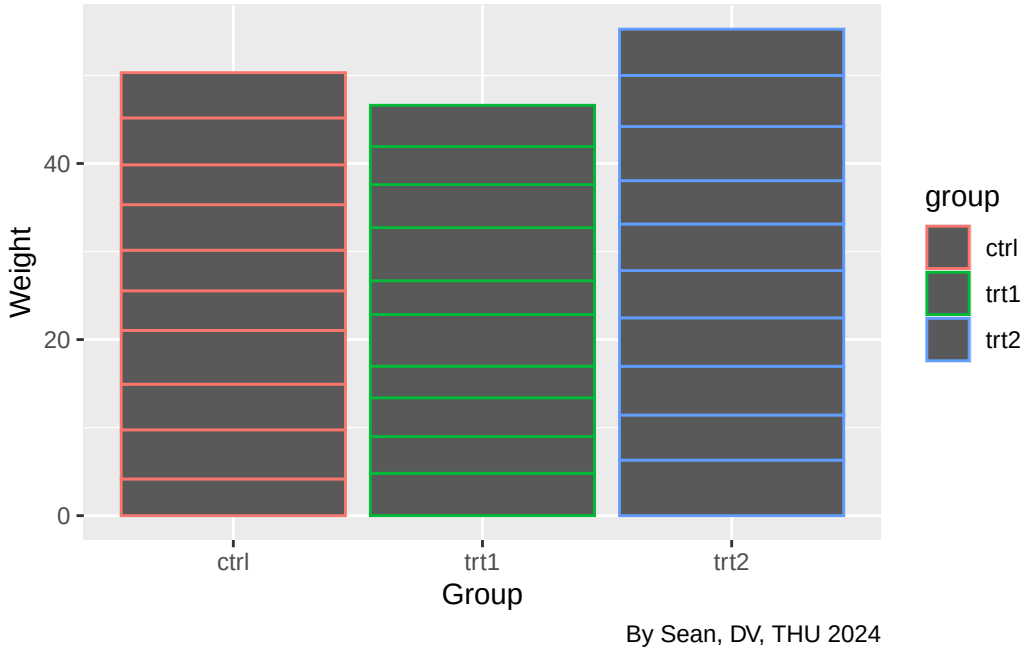
- Adds a column geometry to the plot.
- `geom_col()` creates bars where the height of each bar corresponds to the value of weight for each group.

5 Bar chart with color

In order to give better visualization, we can use `colors` in order to create a more attractive appearance of your data visualization. However, it is very important that `color` and `fill` are different functions

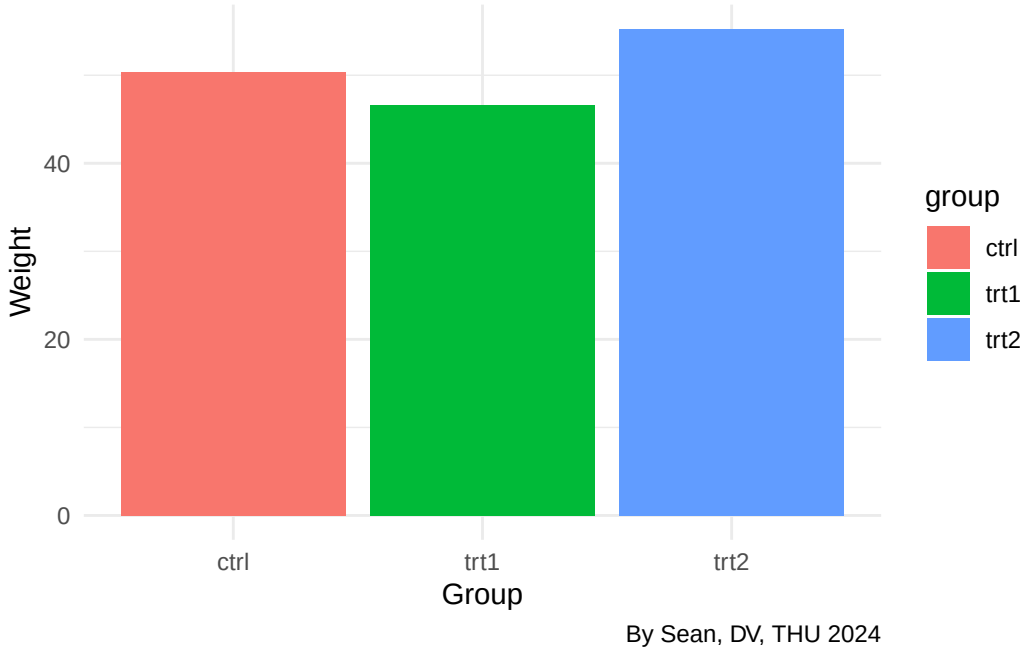
We use `color` to add colors to the outline of the chart according to the `group`

```
ggplot(PlantGrowth, aes(x = group, y = weight, color = group)) +
  geom_col()+
  labs(x= 'Group',
      y= 'Weight',
      captions= 'By Sean, DV, THU 2024')
```



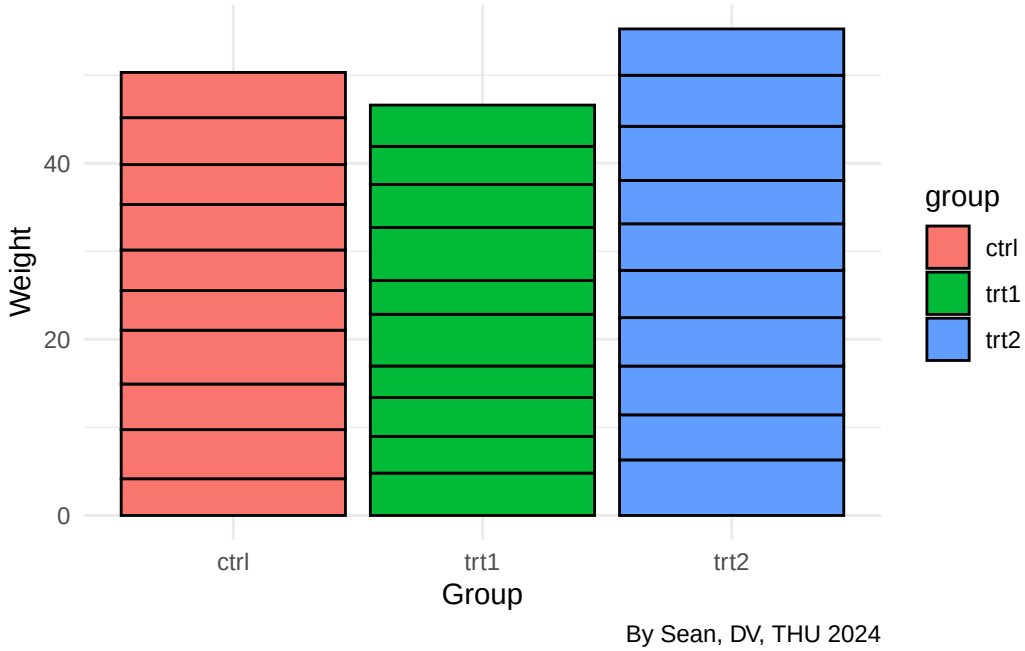
We use `fill` to add colors to the bars according to the `group`

```
ggplot(PlantGrowth, aes(x = group, y = weight, fill = group)) +
  geom_col()+
  theme_minimal()+
  labs(x= 'Group',
      y= 'Weight',
      captions= 'By Sean, DV, THU 2024')
```



When we combine the 2, it creates a better visualization

```
ggplot(PlantGrowth, aes(x = group, y = weight, fill = group)) +
  geom_col(color= 'black')+
  theme_minimal()+
  labs(x= 'Group',
      y= 'Weight',
      captions= 'By Sean, DV, THU 2024')
```

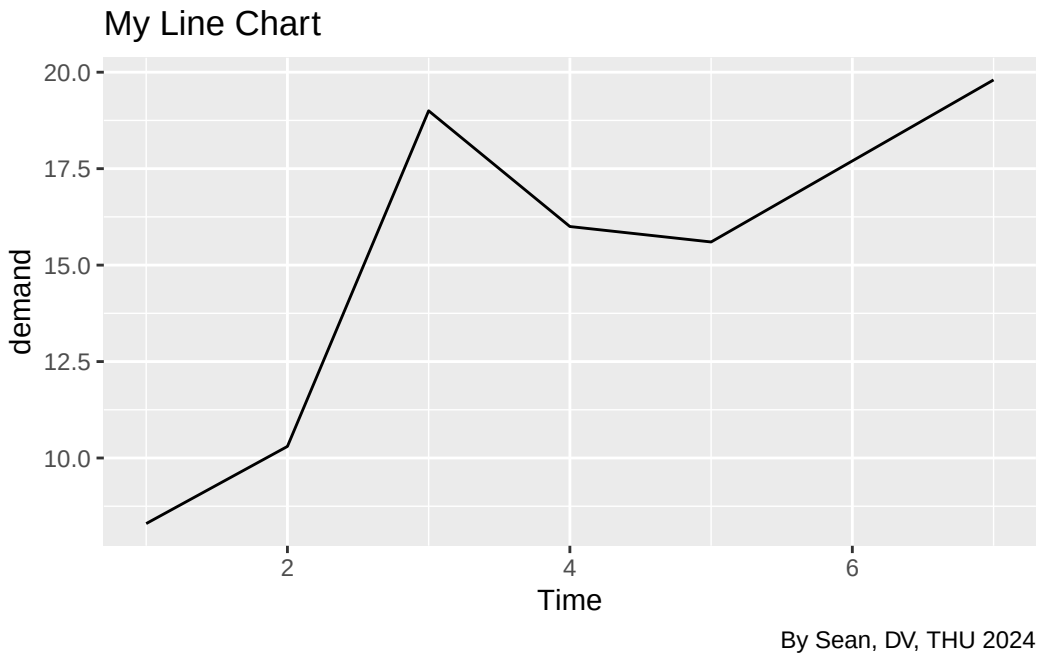


6 Line chart

In this part of the tutorial, we will also teach you how to make line chart using `geom_line()`. This is one of the many geometries or chart that are provided by the `ggplot2` package.

For this demonstration we are going to use the `BOD` dataset provided by the `ggplot2` package

```
ggplot(BOD, aes (x = Time, y = demand)) +
  geom_line() +
  labs(title= "My Line Chart",
       caption = 'By Sean, DV, THU 2024')
```



`aes()`: Defines the following aesthetic:

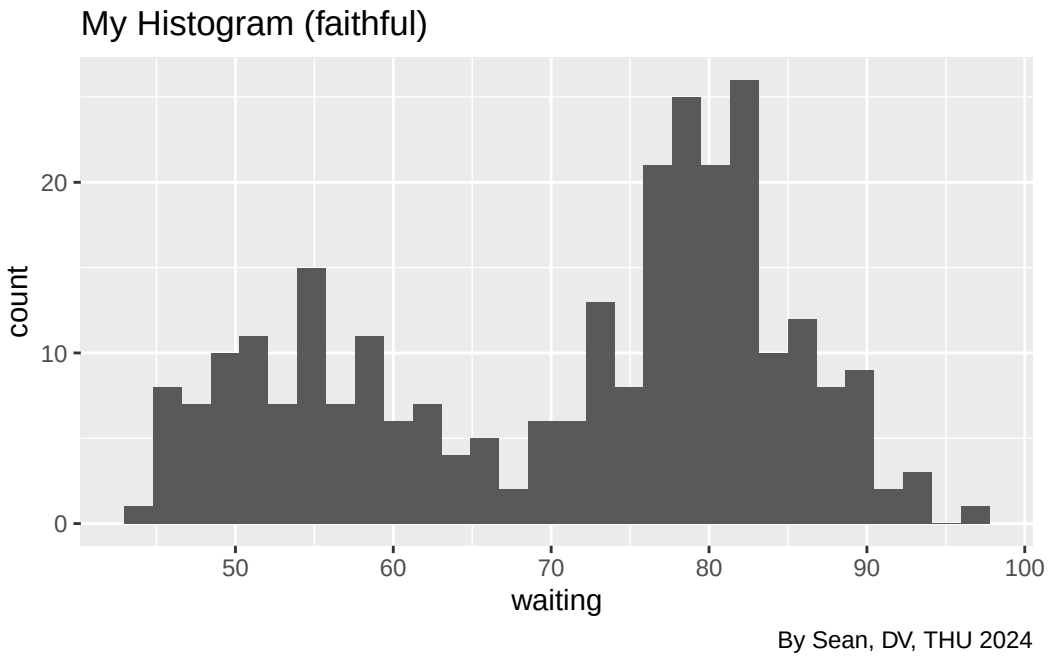
- `x = Time`: Plot the `Time` variable to the x-axis
- `y = demand`: Plot the `demand` variable to the y-axis
- `geom_line()`: Is used to draw a line chart

7 Histogram

To draw histogram we can use the `geom_histogram()` function provided by the `ggplot2` package. In this demonstration, we are going to use the `faithful` dataset provided by the `ggplot2`

```
ggplot(faithful, aes (x=waiting))+
  geom_histogram()+
  labs(title = "My Histogram (faithful)",
       caption= "By Sean, DV, THU 2024")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

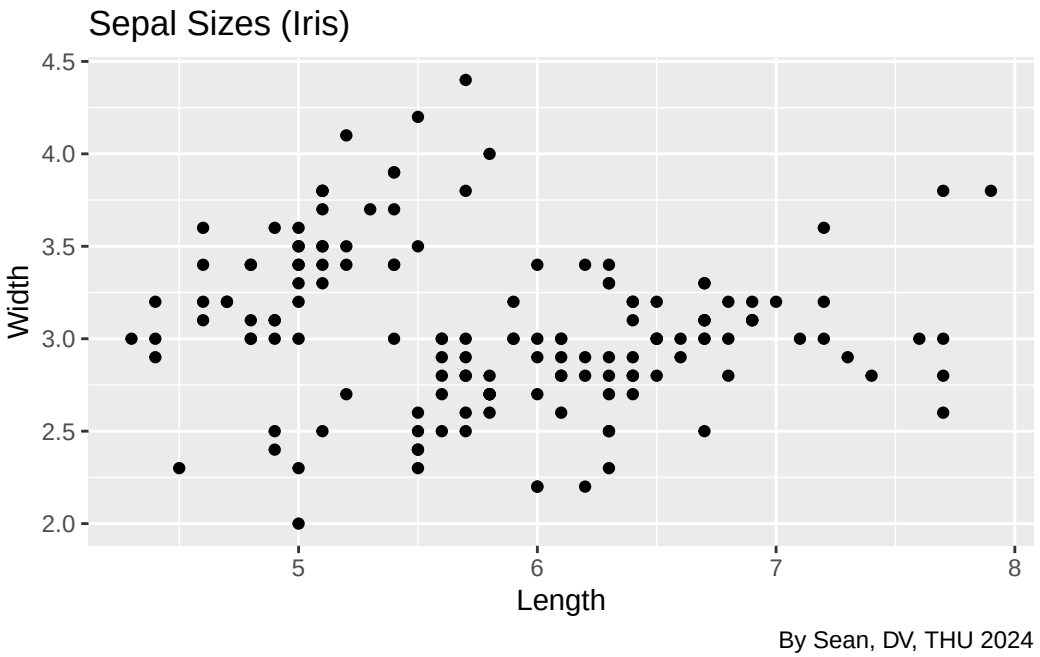


Unlike previous chart, the `geom_histogram` does not require the y-axis variables, this is because the histogram is a chart to show distribution, so y-axis variable is not required.

8 Correlation chart

Correlation chart or in other words scatter plot is also part of the geometrics provided by the `ggplot2` package. In order to plot a scatter plot, we will be using `geom_point`. In this demonstration, we will be using `iris` dataset provided by `ggplot2` package

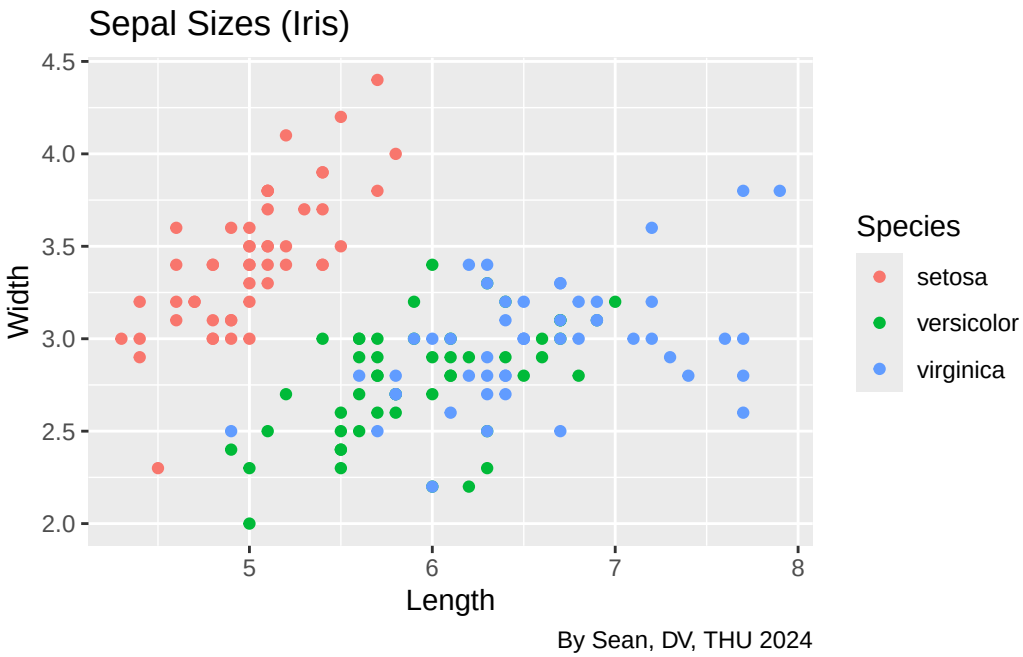
```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width )) +
  geom_point()+
  labs(title="Sepal Sizes (Iris)",
       x= 'Length',
       y= 'Width',
       caption = 'By Sean, DV, THU 2024')
```



9 Correlation chart: Color by group

When adding `colors` by group, similarly to the Bar chart, we can use `color=species`. This means that we color the Scatter plot and categorizing them by the `species` variable in the `iris` dataset

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color= Species )) +
  geom_point()+
  labs(title="Sepal Sizes (Iris)",
       x= 'Length',
       y= 'Width',
       caption = 'By Sean, DV, THU 2024')
```



10 Multigroup histogram

A multigroup histogram is multiple histogram overlayed with each other. In this demonstration we will be using the `birthwt` dataset provided by the `MASS`. As this demonstration needed extra function and `dataset`, we will be loading extra packages `dplyr` and `MASS`.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following object is masked from 'package:MASS':

    select

The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(MASS)
```

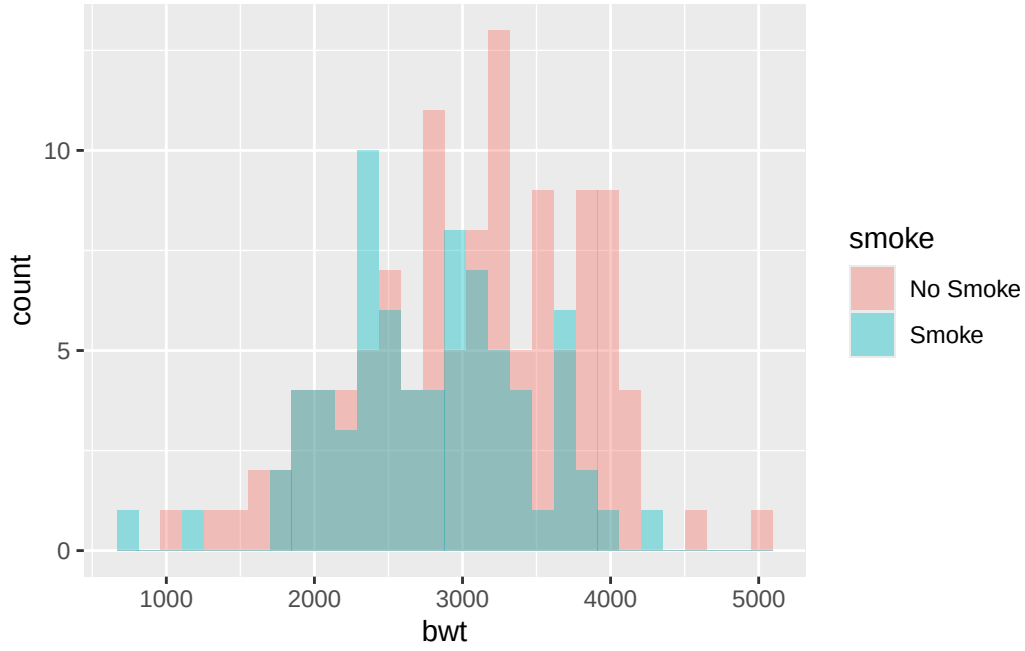
We first need to load `dplyr` package in order to use the `recode_factor` function and `MASS` in order to use the `birthwt` dataset

```
birthwt_mod <- birthwt
birthwt_mod$smoke <- recode_factor(birthwt_mod$smoke, '0' = 'No Smoke', '1' = 'Smoke')
```

Next, we will make a temporary dataset so that the original dataset will not be altered by `birthwt_mod <- birthwt`. The `recode_factor` function is used to change binary data into variables, in this case `1` and `0` is changed into `Smoke` and `No Smoke`. In order to let R know where the data is, we will need to write `birthwt_mod$smoke` to indicate which dataset and column to recode

```
ggplot(birthwt_mod, aes(x = bwt, fill = smoke)) +
  geom_histogram(position = "identity", alpha = 0.4)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



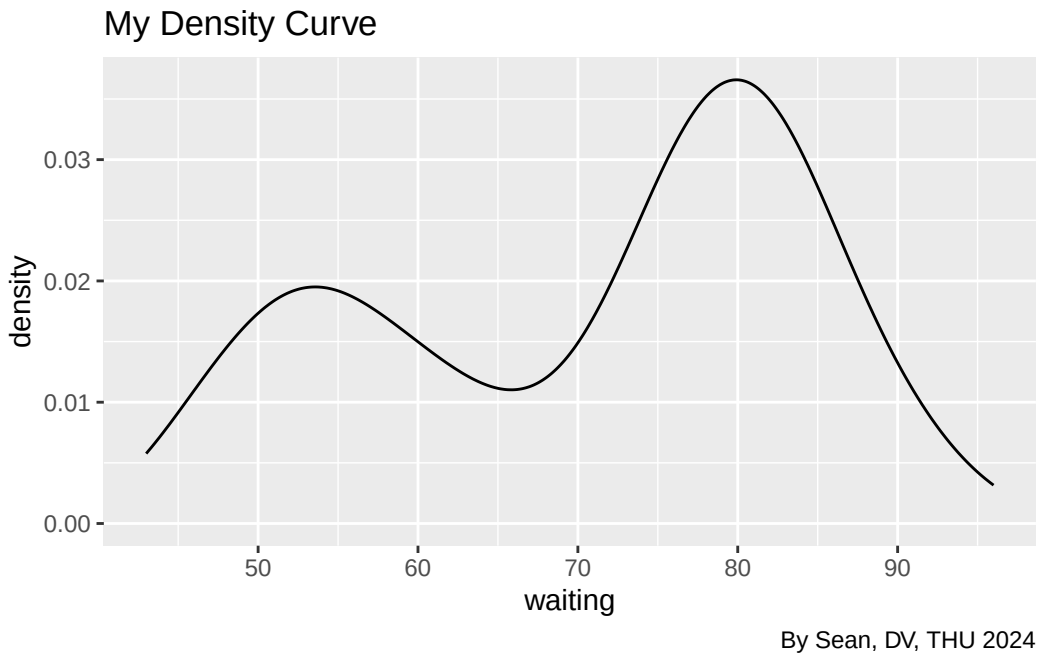
Once the preperation is done, we can use `geom_histogram` to create the histogram. However, additional function is needed to create the multigroup histogram, which is `position = identity` which means that we are able to make the histogram overlay with each other, one being `No smoke` and the other representing `smoke`

`alpha` is used to change the transparency of the chart and `fill = smoke` is the grouped variables

11 Density chart

Density chart is geometric provided by `ggplot2` package that displays the distribution in a smooth line. In this demonstration, we will be using the `faithful` dataset provided by `ggplot2` package

```
ggplot(faithful, aes(x=waiting)) +
   geom_density() +
  labs(title="My Density Curve",
       caption = 'By Sean, DV, THU 2024')
```



To create the Density Curve, we will be using the `geom_density()` function provided by `ggplot2` package.

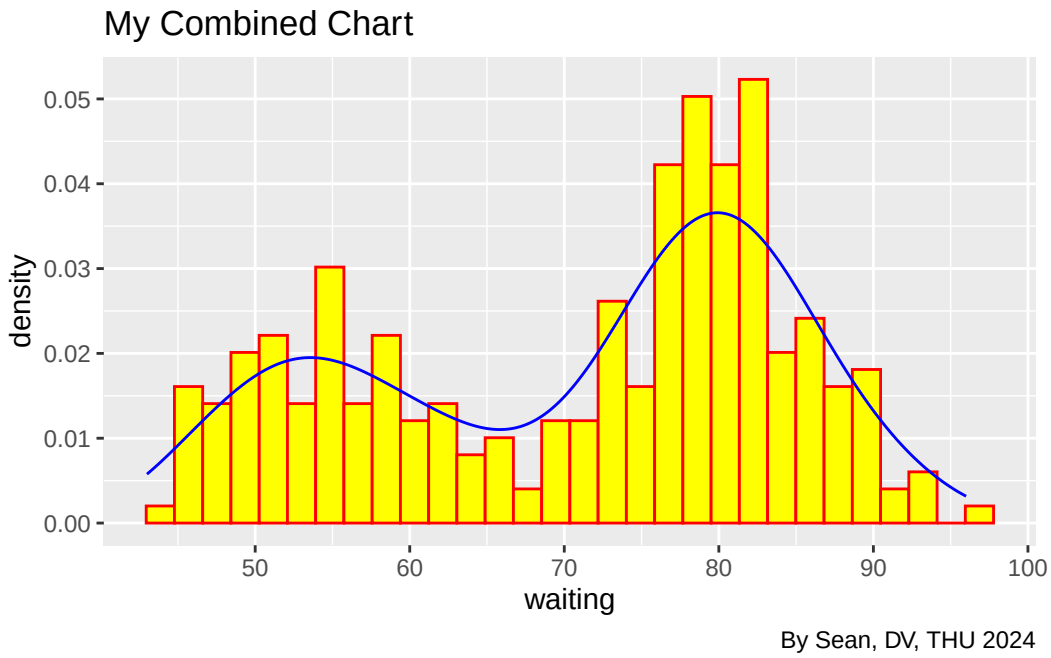
12 Histogram and Density chart

After learning the Histogram and Density Chart using the `geom_histogram` and `geom_density`. We can combine the 2 for a better visualization. The histogram will serve as a raw visualization of the data and the Density chart will visualize the smooth approximations

```
ggplot(faithful, aes(x=waiting, y=..density..)) +
  geom_histogram(color='red', fill='yellow') +
  geom_density(color='blue') +
  labs(title="My Combined Chart",
       caption = 'By Sean, DV, THU 2024')
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



In order to create the combined chart, we must first place the appropriate aesthetics (`aes()`). Althought the, `geom_density` does not require a y-axis `variable`, the `geom_histogram` needs the y-axis `variable`. So in this demonstration we added `y=..density..` so it ensures that it will represent the density data points

For better visualization, `color` and `fill` have been added to the appropriate geometrics

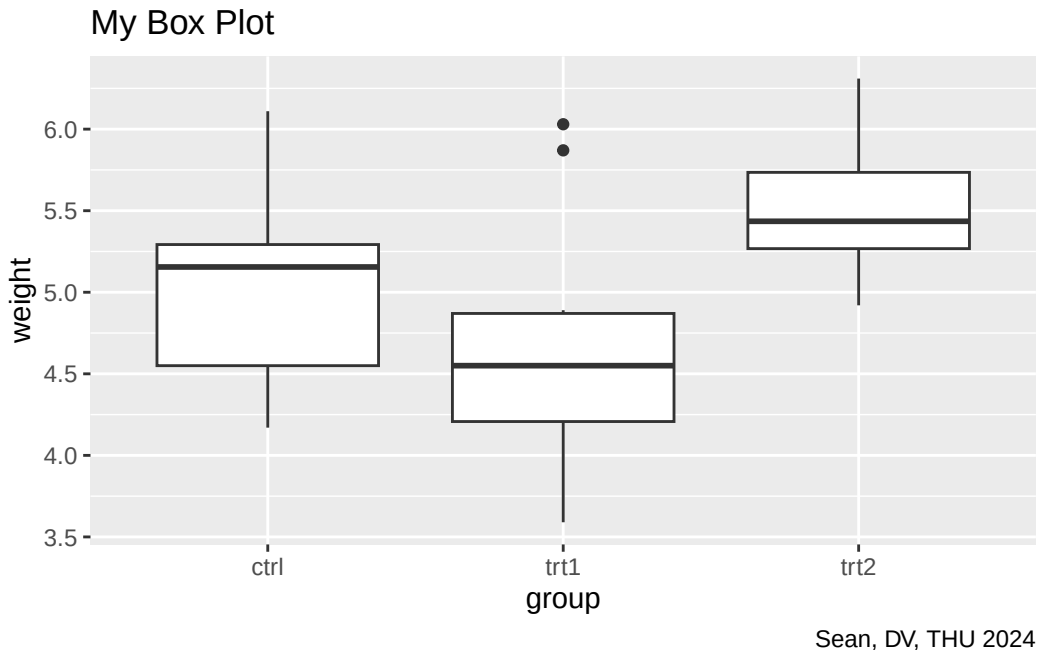
It's also very important to where to place the `geom_histogram`, `geom_density` and other geometrics when you want to display multiple chart. If you place the `geom_histogram` first and then `geom_density`, this means that the density curves will overlay the histogram.

13 Box plot

The last geometrics that will be discussed in this tutorial is Box Plot. Box plot is really useful to display `mean`, `median`, `error bars` and etc. In this demonstration, we will plot a simple box plot.

We will be using the `PlantGrowth` dataset

```
ggplot(PlantGrowth, aes(x = group, y = weight)) +
  geom_boxplot() +
  labs(title = 'My Box Plot',
       caption = 'Sean, DV, THU 2024')
```



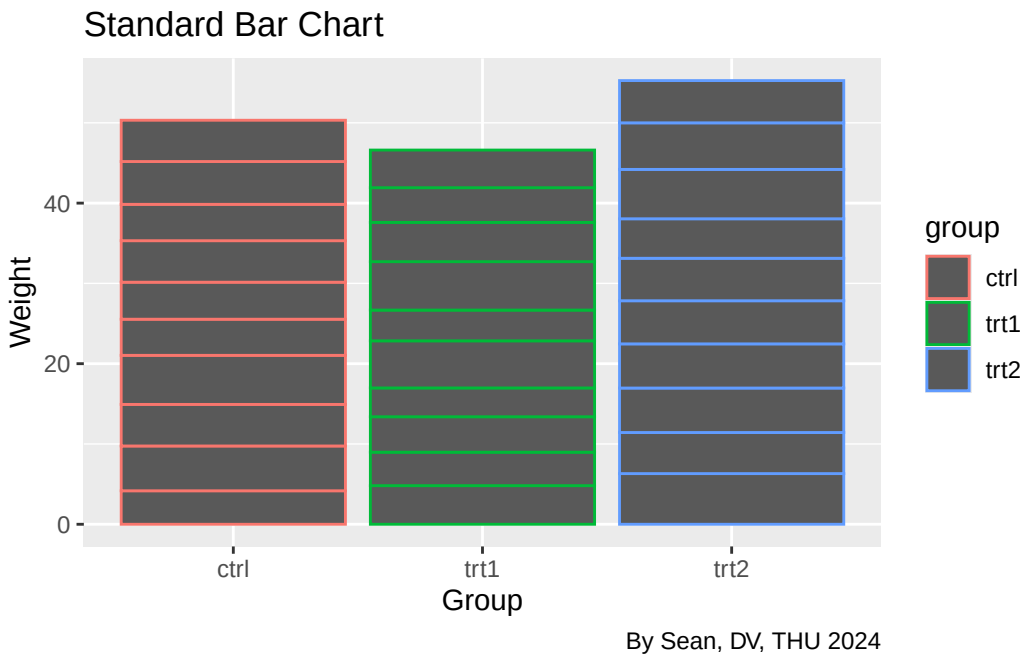
To make the box plot, we use `geom_boxplot()`. The following explains the plot's aesthetic

- `x = group`: Plot the `experimental treatment`.
- `y = weight`: Plot the `weight` of the plant

14 Adding Title, Captions and Axis Labels

Another important thing to do when creating our chart or visualization, is to add labels. Labels serve as a function to further enhance visualization by labeling plots whcih gives the reader a better understanding and direction. Moreover, it's also used to claim credits of your work.

```
ggplot(PlantGrowth, aes(x = group, y = weight, color = group)) +
  geom_col()+
  labs(title = 'Standard Bar Chart',
      x= 'Group',
      y= 'Weight',
      captions= 'By Sean, DV, THU 2024')
```

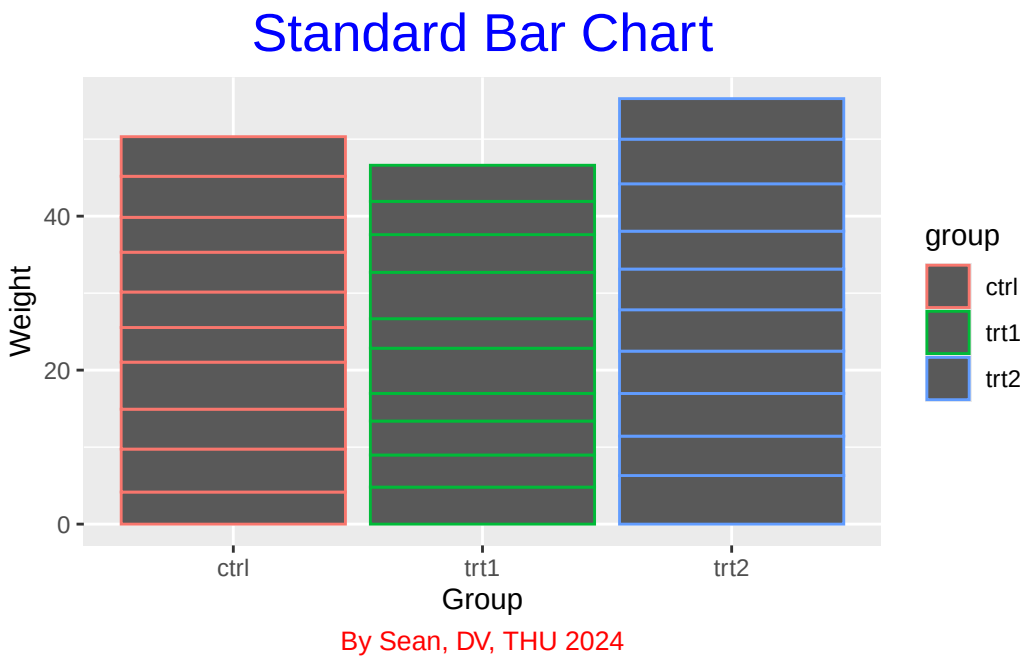


We can use `labs()` to change add title, axis labels and captions:

- `labs(title=` : Is a function to add `'text'` to the title of the chart
- `labs(x=`: Is a function to change the x-axis label
- `labs(y-`: Is a function to change the y-axis label
- `labs(captions=`: Is a function to add `'text'` on the bottom right of the chart

It is to note that whenever you are adding `'text'` in R, we have to use `' '` between the text, otherwise it will be recognize as a function, instead of a `'text'`

```r
ggplot(PlantGrowth, aes(x = group, y = weight, color = group)) +
  geom_col()+
  labs(title = 'Standard Bar Chart',
      x= 'Group',
      y= 'Weight',
      captions= 'By Sean, DV, THU 2024') +
  theme(plot.title = (element_text(size=20, hjust= 0.5, color = 'blue')),
        plot.caption = (element_text(size=10, hjust= 0.5, color = 'red')))
```

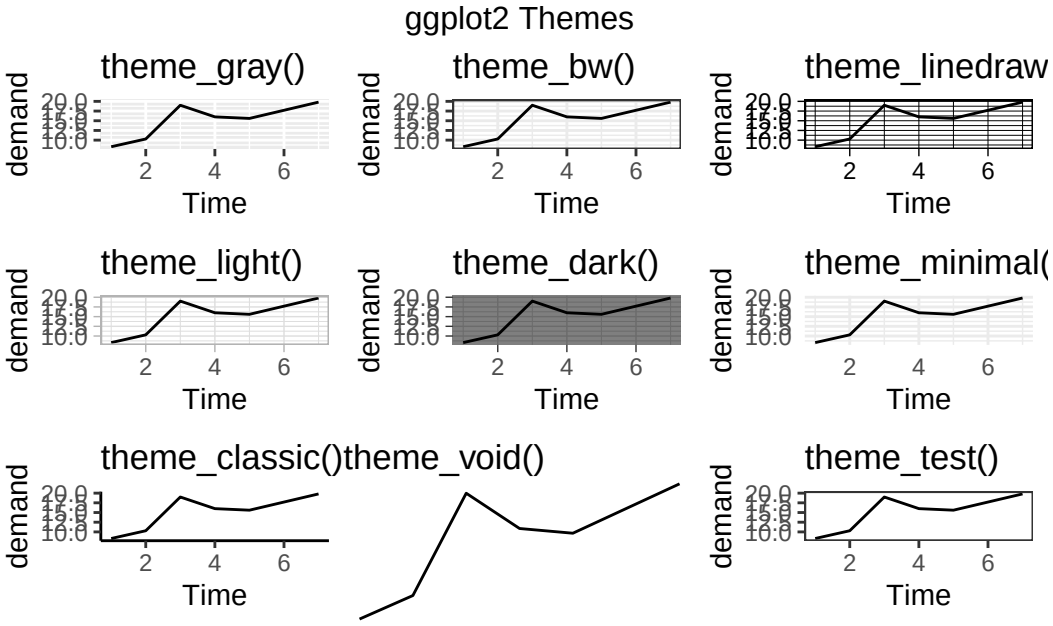


Furthermore, we can also change the appearance of the `'text'` in the `theme()` which we will discuss in more detail in the next section

- `theme(plot.title`: A function to let the R know to change the `title`'s element
- `element_text`: Which element to change the appearance
- `size`: change the font size
- `hjust`: changes the position of the text
- `theme(plot.caption`: A function to let R know to change the `caption`'s element

15 Theme

For the last part of this tutorial, we can enhance our visualization further using `theme()`, it is a feature that is provided by the `ggplot2` packages



- `theme_gray()` (default):
  - The signature `ggplot2` theme with a grey background and white gridlines, designed to put the data forward yet make comparisons easy.
- `theme_bw()` (black and white):
  - The classic dark-on-light `ggplot2` theme. May work better for presentations displayed with a projector.
- `theme_linedraw()`:
  - A theme with only black lines of various widths on white backgrounds, reminiscent of a line drawing. Serves a purpose similar to `theme_bw()`. Note that this theme has some very thin lines (`<< 1 pt`) which some journals may refuse.
- `theme_light()`:
  - A theme similar to `theme_linedraw()` but with light grey lines and axes, to direct more attention towards the data.
- `theme_dark()`:
  - The dark cousin of `theme_light()`, with similar line sizes but a dark background. Useful to make thin coloured lines pop out.
- `theme_minimal()`:
  - A minimalistic theme with no background annotations.

- `theme_classic()`:
  - A classic-looking theme, with x and y axis lines and no gridlines.
- `theme_void()`:
  - A completely empty theme.
- `theme_test()`:
  - A theme for visual unit tests. It should ideally never change except for new features.

16 Adding and Removing Legend

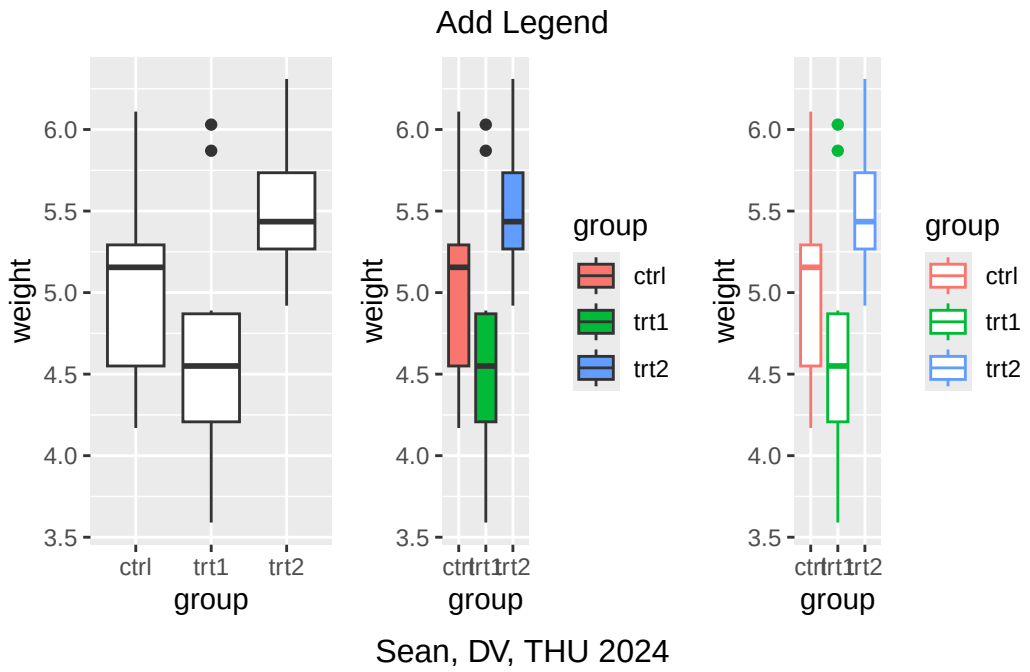
To add Legend:

By adding `fill` or `color` options to `aes()`, a legend is created automatically on the right side of the plot.

- `library(gridExtra)`: This loads the `gridExtra` library to use `grid.arrange`
- `fill`: Fill in the `colors` inside the chart
- `color`: Change the `color` of the outlines of the chart
- `grid.arrange`: Arrange the order of the chart
- `top`: Add `text` on the top of the chart
- `bottom`: Add `text` on the bottom of the chart

```
library(gridExtra)

p1 <- ggplot(PlantGrowth, aes(x = group, y = weight)) +
  geom_boxplot()
p2 <- ggplot(PlantGrowth, aes(x = group, y = weight, fill = group)) +
  geom_boxplot()
p3 <- ggplot(PlantGrowth, aes(x = group, y = weight, color = group)) +
  geom_boxplot()
grid.arrange(p1,p2,p3,ncol=3,top='Add Legend', bottom = 'Sean, DV, THU 2024')
```

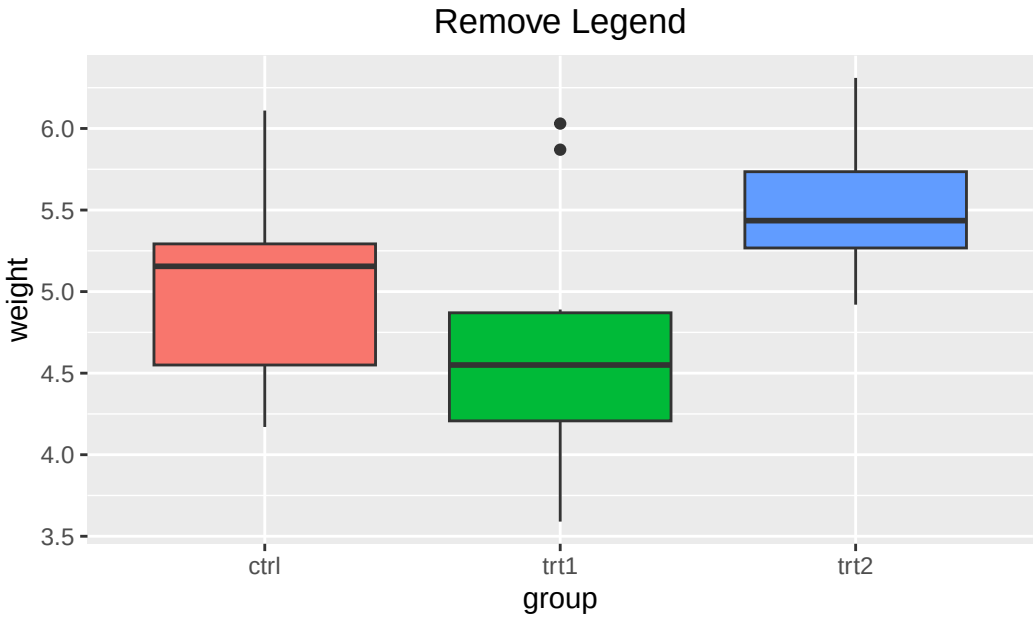


To Remove Legend

The legend is removed by `guides(fill = FALSE)`

```
ggplot(PlantGrowth, aes(x = group, y = weight, fill = group)) +
  geom_boxplot() +
  guides(fill = FALSE) +
  ggtitle('Remove Legend') +
  labs(caption = 'Sean, DV, THU 2024') +
  theme(plot.title = element_text(hjust=0.5))
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
of ggplot2 3.3.4.
```



Sean, DV, THU 2024