~~~
---
title: "Data Visualization with ggplot2 (Part 3)"
subtitle: "Statistical Plots (Chapter 1)"
author: "Seun Odeyemi"
date: "`r Sys.Date()`"
output:
  pdf_document:
    df_print: kable
    toc: yes
    toc_depth: 4
  # prettydoc::html_pretty:
  #   theme: cayman
  #   highlight: github
  #   toc: yes
  #   toc_depth: 4
  # bibliography: dataviz.bib
---


```{r setup, include=FALSE, cache=FALSE}
knitr::opts_chunk$set(error = TRUE, collapse = TRUE, comment = "#>")
```

## Load Libraries

```{r load libraries, include=TRUE, message=FALSE}
library(readr)
library(dplyr)
library(ggplot2)
# library(ggplot2movies)
library(tidyr)
library(skimr)
library(knitr)
library(kableExtra)
library(RColorBrewer)
library(grid)
library(ggthemes)
library(forcats)
library(GGally)
library(here)
```



```{r refresher_1, fig.height=4, fig.width=4, fig.align='center'}
# Using the sample function to do a random sampling of a dataset
# library(ggplot2movies)
# set.seed(123)
# movies_small <- movies[sample(nrow(movies), 1000), ]
# movies_small$rating <- factor(round(movies_small$rating))

(movies_small <- readRDS(here("datasets/ch1_movies_small.RDS"))) %>% as_tibble()

# take a look at the variables in movies_small
~~~

~~~
glimpse(movies_small)

# Build a scatter plot with mean and 95% CI
ggplot(movies_small, aes(x = rating, y = votes)) +
  geom_point() +
  stat_summary(fun.data = "mean_cl_normal",
               geom = "crossbar",
               width = 0.2,
               col = "red") +
  scale_y_log10()
```

```{r refresher_2, fig.height=4, fig.width=4, fig.align='center'}
# Reproduce the plot
ggplot(diamonds, aes(x = carat, y = price, col = color)) +
  geom_point(alpha = 0.5, size = 0.5, shape = 16) +
  scale_x_log10(expression(log[10](Carat)), limits = c(0.1,10)) +
  scale_y_log10(expression(log[10](Price)), limits = c(100,100000)) +
  scale_color_brewer(palette = "YlOrRd") +
  coord_equal() +
  theme_classic()
```

Good job! This is a nice way of transforming data and then plotting it in one command.

## Refresher (3)

```{r refresher_3, fig.height=4, fig.width=4, fig.align='center'}
# Add smooth layer and facet the plot
ggplot(diamonds, aes(x = carat, y = price, col = color)) +
  stat_smooth(method = "lm") +
  # geom_point(alpha = 0.5, size = 0.5, shape = 16) +
  scale_x_log10(expression(log[10](Carat)), limits = c(0.1,10)) +
  scale_y_log10(expression(log[10](Price)), limits = c(100,100000)) +
  scale_color_brewer(palette = "YlOrRd") +
  coord_equal() +
  theme_classic()
```

```{r transformations, fig.height=4, fig.width=4, fig.align='center'}
# movies_small is available

# Add a boxplot geom
d <- ggplot(movies_small, aes(x = rating, y = votes)) +
  geom_point() +
  geom_boxplot() +
  stat_summary(fun.data = "mean_cl_normal",
               geom = "crossbar",
               width = 0.2,
               col = "red")

# Untransformed plot
d
~~~

~~~
# Transform the scale
d + scale_y_log10() # the transformation happens before calculating the statistics

# Transform the coordinates
d + coord_trans(y = "log10") # the transformation happens after calculating the statistics
```

```{r cut_it_up, fig.height=4, fig.width=4, fig.align='center'}
# Plot object p
p <- ggplot(diamonds, aes(x = carat, y = price))

# Use cut_interval
p + geom_boxplot(aes(group = cut_interval(carat, n = 10)))

# Use cut_number
p + geom_boxplot(aes(group = cut_number(carat, n = 10)))

# Use cut_width
p + geom_boxplot(aes(group = cut_width(carat, width = 0.25)))
```

### Understanding quartiles


```{r plot_quart_function}
plot_quart <- function(n) {
  set.seed(123)
  playData <- data.frame(raw.values = rnorm(n, 1, 6))

  quan.summary <- data.frame(t(sapply(1:9, function(x) quantile(playData$raw.values, type = x))))
  names(quan.summary) <- c("Min", "Q1", "Median", "Q3", "Max")
  quan.summary$Type <- as.factor(1:9)

  library(reshape2)
  quan.summary <- melt(quan.summary, id = "Type")
  quan.summary <- list(quartiles = quan.summary, values = playData)

  ggplot(quan.summary$quartiles, aes(x = Type, y = value, col = variable)) +
    geom_point() +
    geom_rug(data = quan.summary$values, aes(y = raw.values), sides = "l", inherit.aes = F)
}
```

```{r use_plot_quart, fig.height=4, fig.width=4, fig.align='center'}
plot_quart(4)

plot_quart(10)

plot_quart(50)

plot_quart(100)
```
~~~

~~~
```{r geom_density, fig.height=6, fig.width=6, fig.align='center'}
load("datasets/test_datasets.RData")

test_data <- ch1_test_data

# Calculating density: d
d <- density(test_data$norm)

# Use which.max() to calculate mode
mode <- d$x[which.max(d$y)]

# Finish the ggplot call
ggplot(test_data, aes(x = norm)) +
  geom_rug() +
  geom_density() +
  geom_vline(xintercept = mode, col = "red")
```
~~~

### Combine Density Plots and Histogram

Sometimes it is useful to compare a histogram with a density plot. However, the histogram's y-scale must first be converted to frequency instead of absolute count. After doing so, you can add an empirical PDF using `geom_density()` or a theoretical PDF using `stat_function()`.

Can you finish the plot below by following the steps?

~~~
```{r density_plus_hist, fig.height=6, fig.width=6, fig.align='center'}
# Arguments you'll need later on
fun_args <- list(mean = mean(test_data$norm), sd = sd(test_data$norm))

# Finish the ggplot
ggplot(test_data, aes(x = norm)) + # set the y aesthetic to the internal ..density.. variable, overriding the default
..count...
  geom_histogram(aes(y = ..density..)) +
  geom_density(col = "red") +
  stat_function(fun = dnorm, args = fun_args, col = "blue")
```
~~~

~~~
```{r adjusting_density_plots, fig.height=6, fig.width=6, fig.align='center'}
small_data <- data.frame(x = c(-3.5, 0.0, 0.5, 6.0))

# Get the bandwith
get_bw <- density(small_data$x)$bw

# Basic plotting object
p <- ggplot(small_data, aes(x = x)) +
  geom_rug() +
  coord_cartesian(ylim = c(0,0.5))

# Create three plots
~~~

p + geom_density()
p + geom_density(adjust = 0.25)
p + geom_density(bw = 0.25 * get_bw)

# Create two plots
p + geom_density(kernel = "r")
p + geom_density(kernel = "e")

## Multiple Groups / Variables

By **Groups** we mean levels within a factor variable. In this case it is the eating habits of different mammals. This distribution we are interested in is the amount of total sleep time experienced by each mammal. Up until this point we would have used a `geom_point()` with `position_jitter(0.2)`, but we've seen that we can also use *_boxplots_*. (I should point out that although we could use boxplots in this case it is not really reasonable since the insectivore group only has five observations. The problem with boxplots is that they don't show information about the number of observations. We can remedy this situation by setting the width of each box relative to the _n_ value of each group.).

```{r load_mammals}
(mammals <- read_rds(here("datasets/mammals.RDS")))

```

```{r boxplot_mammals, fig.height=6, fig.width=6, fig.align='center'}
# Not recommended for this dataset
ggplot(mammals, aes(x = vore, y = sleep_total)) +
  geom_boxplot(varwidth = TRUE)
```

```{r density_plot_mammals, fig.height=6, fig.width=6, fig.align='center'}
ggplot(mammals, aes(x = sleep_total, fill = vore)) +
  geom_density(col = NA, alpha = 0.35)

# Add weights
mammals <- mammals %>%
  group_by(vore) %>%
  mutate(n = n()/nrow(mammals))

ggplot(mammals, aes(x = sleep_total, fill = vore)) +
  geom_density(aes(weight = n), col = NA, alpha = 0.35)
```

```{r violin_plots, fig.height=6, fig.width=6, fig.align='center'}
ggplot(mammals, aes(x = vore, y = sleep_total)) +
  geom_violin()

ggplot(mammals, aes(x = vore,
                    y = sleep_total,
                    fill = vore)) +
  geom_violin(aes(weight = n), col = NA)
```

### Comparing separate variables

~~~
```{r old_faithful, fig.height=6, fig.width=6, fig.align='center'}
data("faithful")

dim(faithful)

head(faithful)

# Scatter Plot
ggplot(faithful, aes(x = waiting, y = eruptions)) +
  geom_point()

# 2-D Density Plot
ggplot(faithful, aes(x = waiting, y = eruptions)) +
  geom_density_2d()

# 2-D Density Plot w/ Monochromatic Color Scales
ggplot(faithful, aes(x = waiting, y = eruptions)) +
  stat_density_2d(geom = "tile",
                  aes(fill = ..density..),
                  contour = FALSE)

library(viridis)
ggplot(faithful, aes(x = waiting, y = eruptions)) +
  stat_density_2d(geom = "tile", aes(fill = ..density..),
                  contour = FALSE) +
  scale_fill_viridis()

ggplot(faithful, aes(x = waiting, y = eruptions)) +
  stat_density_2d(geom = "point",
                  aes(size = ..density..),
                  n = 20, contour = FALSE) +
  scale_size(range = c(0, 9))
```
~~~