Data Visualization with R - Part 1

`plot()` Function

Load Library

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Read E-Commerce Data

```
raw_data <- read_csv(file = "data/ecommerce_data.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   order_date = col_date(format = ""),
##   ship_date = col_date(format = ""),
##   aging = col_double(),
##   sales = col_double(),
##   quantity = col_double(),
##   discount = col_double(),
##   profit = col_double(),
##   shipping_cost = col_double()
## )
## i Use `spec()` for the full column specifications.
```

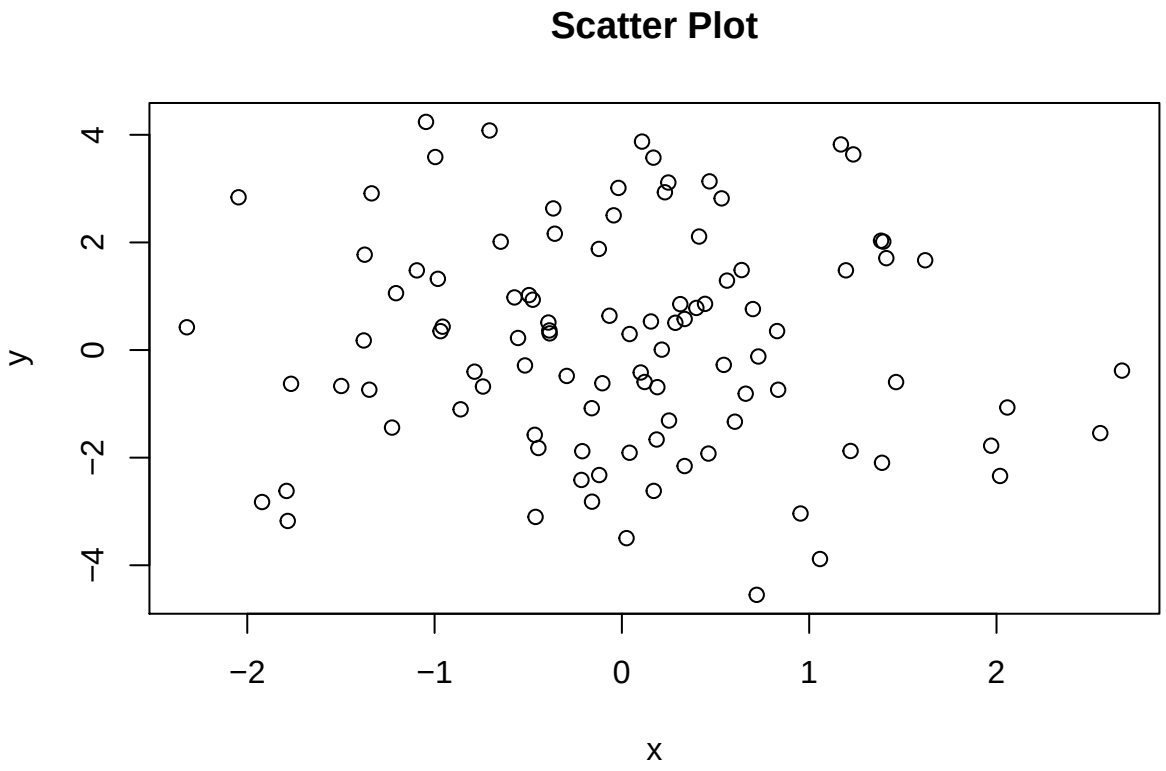
Data Inspection

```
raw_data %>%
  glimpse()
```

```
## Rows: 51,279
## Columns: 21
## $ order_id         <chr> "AU-2015-1", "AU-2015-2", "AU-2015-3", "AU-2015-4", "~
## $ order_date       <date> 2015-11-09, 2015-06-30, 2015-12-05, 2015-05-09, 2015~
## $ ship_date        <date> 2015-11-17, 2015-07-02, 2015-12-13, 2015-05-16, 2015~
## $ aging            <dbl> 8, 2, 8, 7, 9, 8, 1, 7, 7, 10, 10, 1, 7, 5, 9, 10, 4,~
## $ ship_mode        <chr> "First Class", "First Class", "First Class", "First C~
## $ product_category <chr> "Auto & Accessories", "Auto & Accessories", "Auto & A~
## $ product          <chr> "Car Media Players", "Car Speakers", "Car Body Covers~
## $ sales            <dbl> 140, 211, 117, 118, 250, 72, 54, 114, 231, 140, 211, ~
## $ quantity         <dbl> 2, 3, 5, 2, 1, 3, 1, 5, 5, 2, 4, 4, 1, 3, 4, 2, 2, 5,~
## $ discount         <dbl> 0.05, 0.03, 0.01, 0.05, 0.04, 0.04, 0.05, 0.02, 0.03,~
## $ profit           <dbl> 46.0, 112.0, 31.2, 26.2, 160.0, 24.0, 54.0, 22.6, 116~
## $ shipping_cost    <dbl> 4.6, 11.2, 3.1, 2.6, 16.0, 2.4, 5.4, 2.3, 11.6, 5.4, ~
## $ order_priority   <chr> "Medium", "Medium", "Critical", "High", "Critical", "~
## $ customer_id      <chr> "LS-001", "IZ-002", "EN-003", "AN-004", "ON-005", "TO~
## $ customer_name    <chr> "Lane Daniels", "Alvarado Kriz", "Moon Weien", "Sanch~
## $ segment          <chr> "Consumer", "Home Office", "Consumer", "Corporate", "~
## $ city             <chr> "Brisbane", "Berlin", "Porirua", "Kabul", "Townsville~
## $ state            <chr> "Queensland", "Berlin", "Wellington", "Kabul", "Queen~
## $ country          <chr> "Australia", "Germany", "New Zealand", "Afghanistan",~
## $ region           <chr> "Oceania", "Central", "Oceania", "Central Asia", "Oce~
## $ months           <chr> "Nov", "Jun", "Dec", "May", "Jul", "Feb", "Apr", "Mar~
```

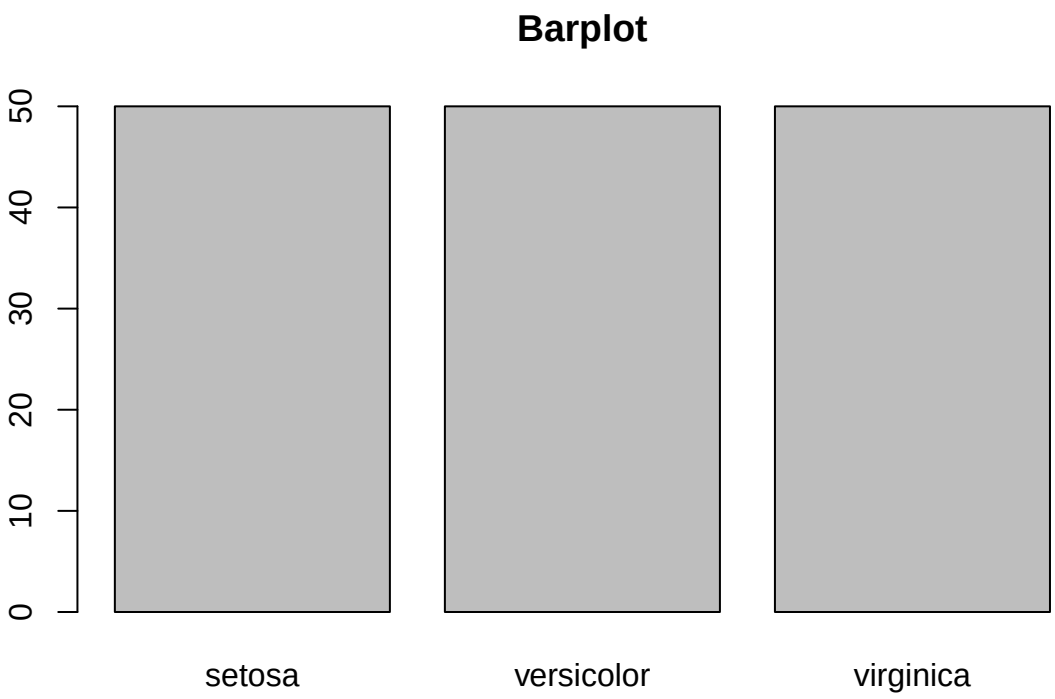
Scatter Plot

```
set.seed(1000)
x <- rnorm(100)
y <- 2*rnorm(100) + rnorm(100)
plot(x,y, main = "Scatter Plot")
```



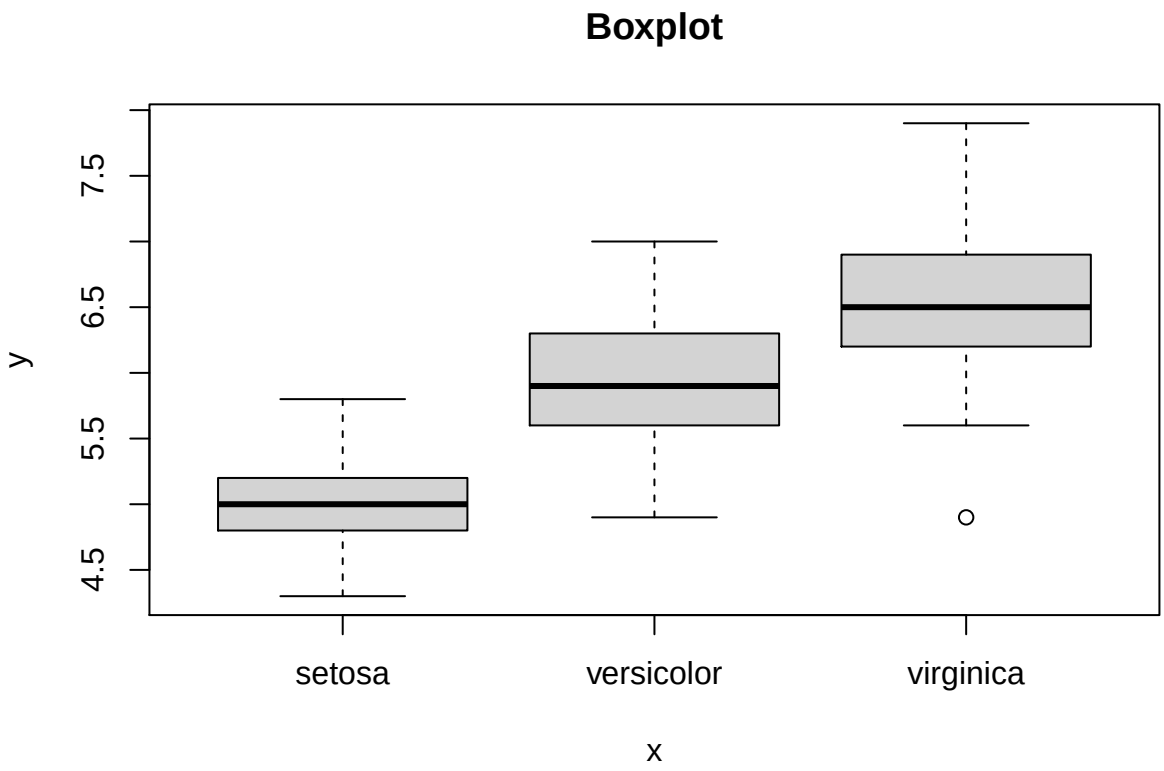
Barplot

```
plot(iris$Species, main = "Barplot")
```



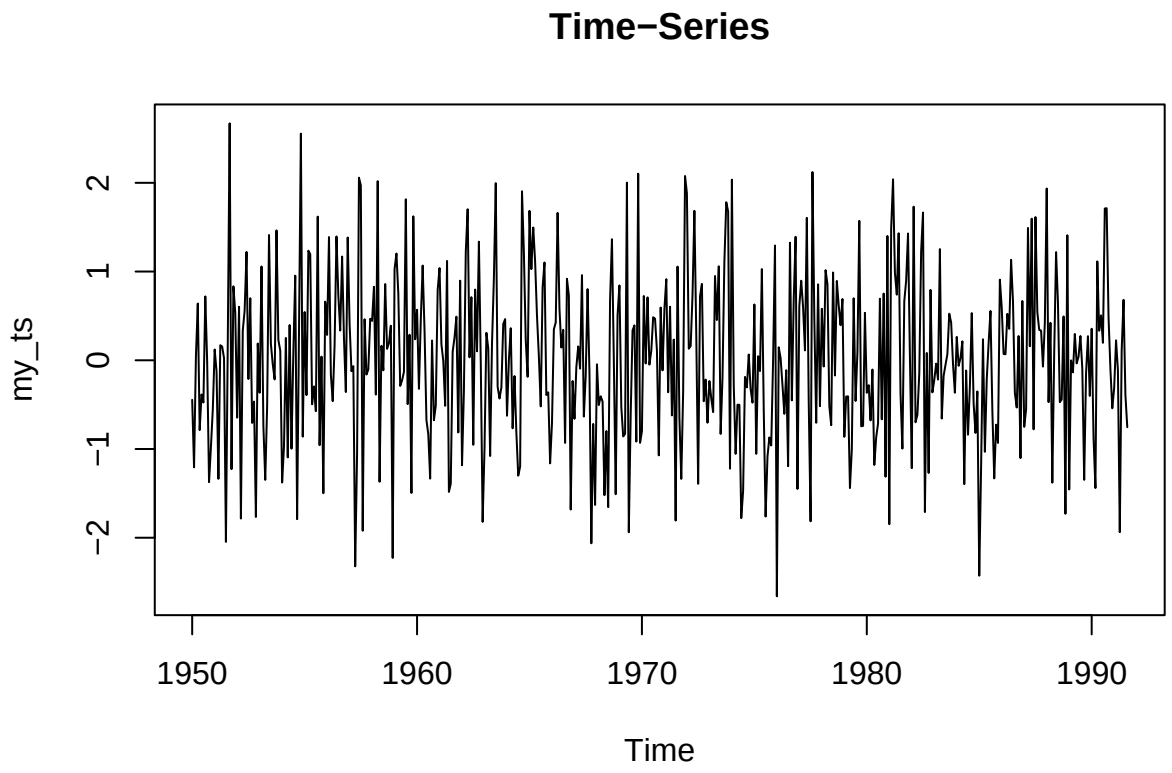
Boxplot

```
plot(x = iris$Species, y = iris$Sepal.Length, main = "Boxplot")
```



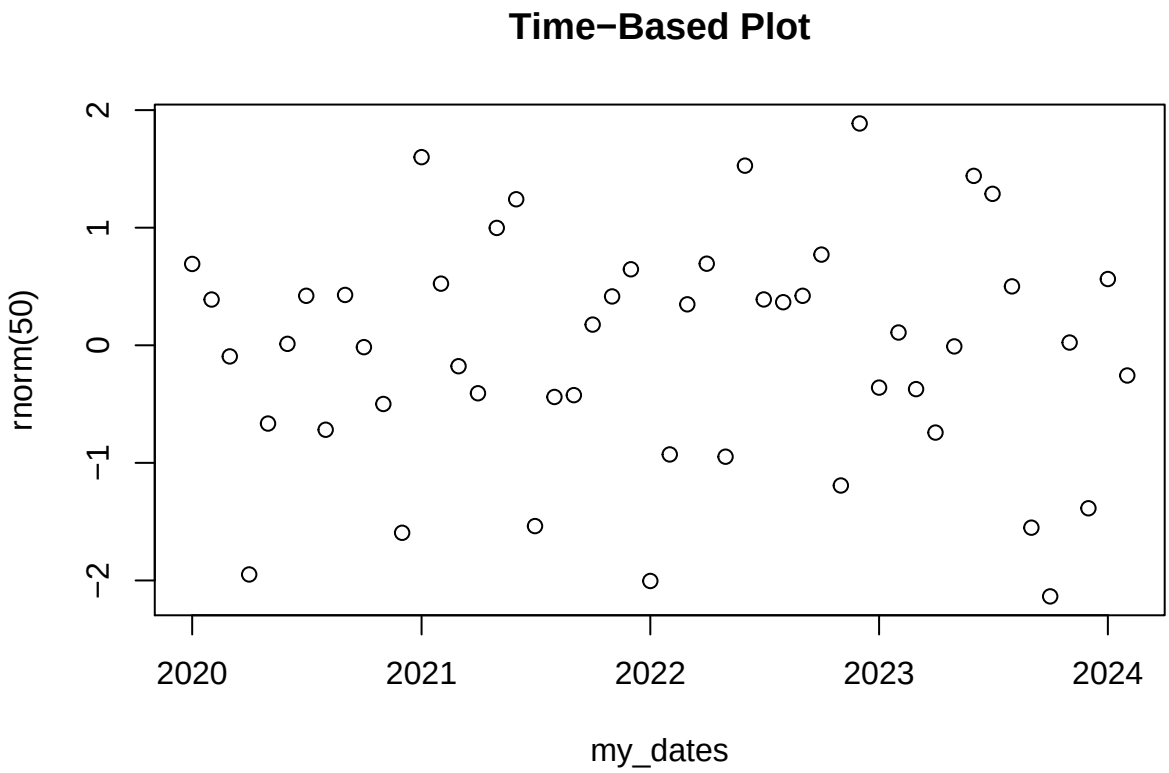
Time Series Plot

```
# time series object
set.seed(1000)
my_ts <- ts(data = rnorm(500), start = c(1950, 1), frequency = 12)
plot(my_ts, main = "Time-Series")
```



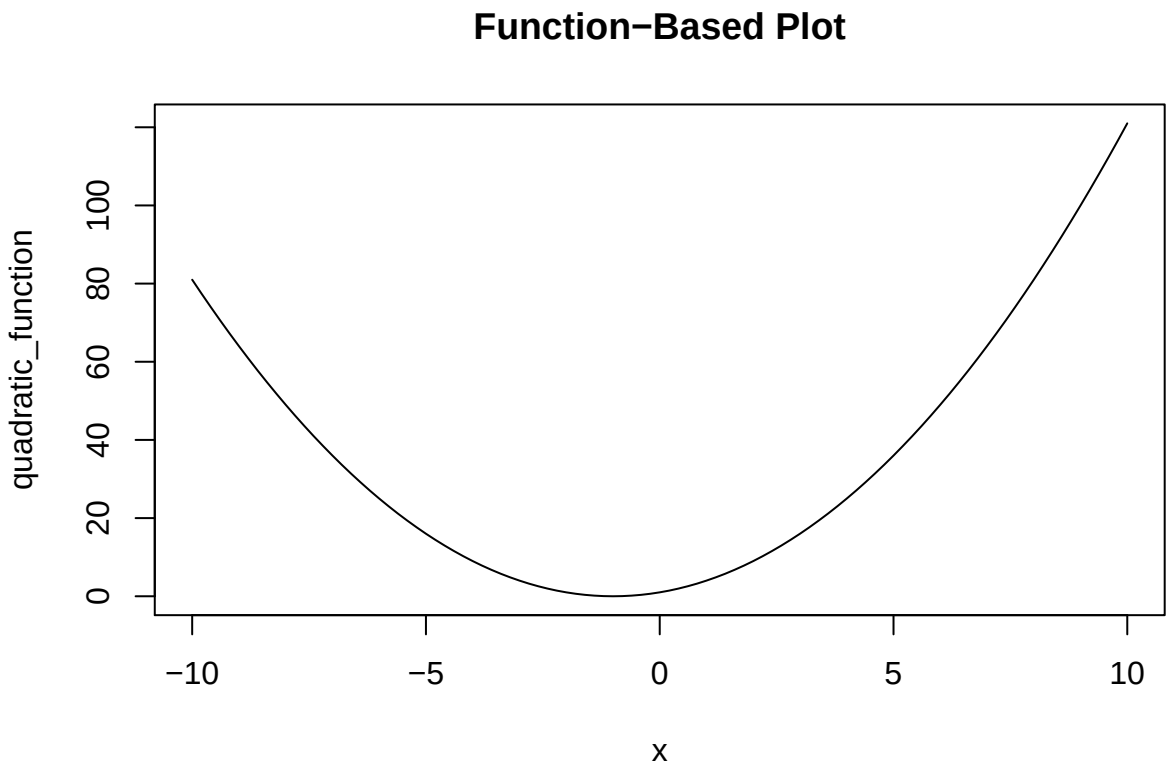
Time-Based Plot

```
# date object
my_dates <- seq(as.Date("2020-01-01"), by = "month", length = 50)
plot(my_dates, rnorm(50), main = "Time-Based Plot")
```



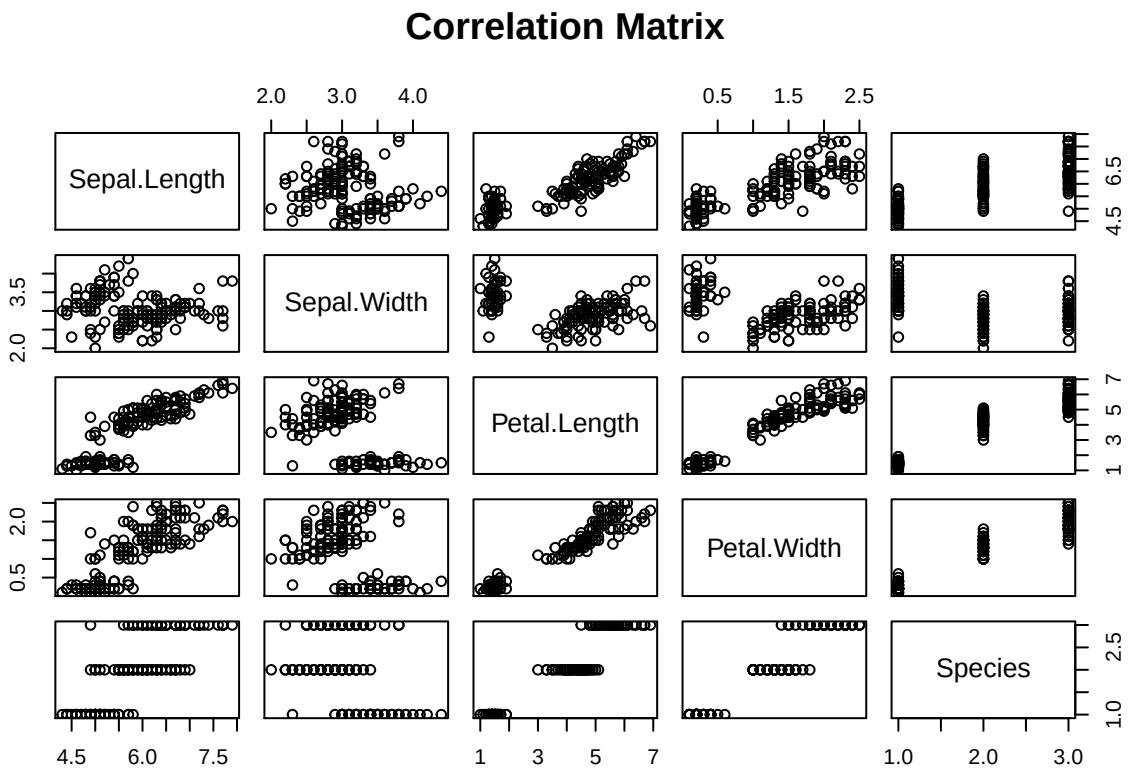
Function-Based Plot

```
# function object
quadratic_function <- function(x) x^2+2*x+1

plot(quadratic_function, -10, 10, main = "Function-Based Plot")
```



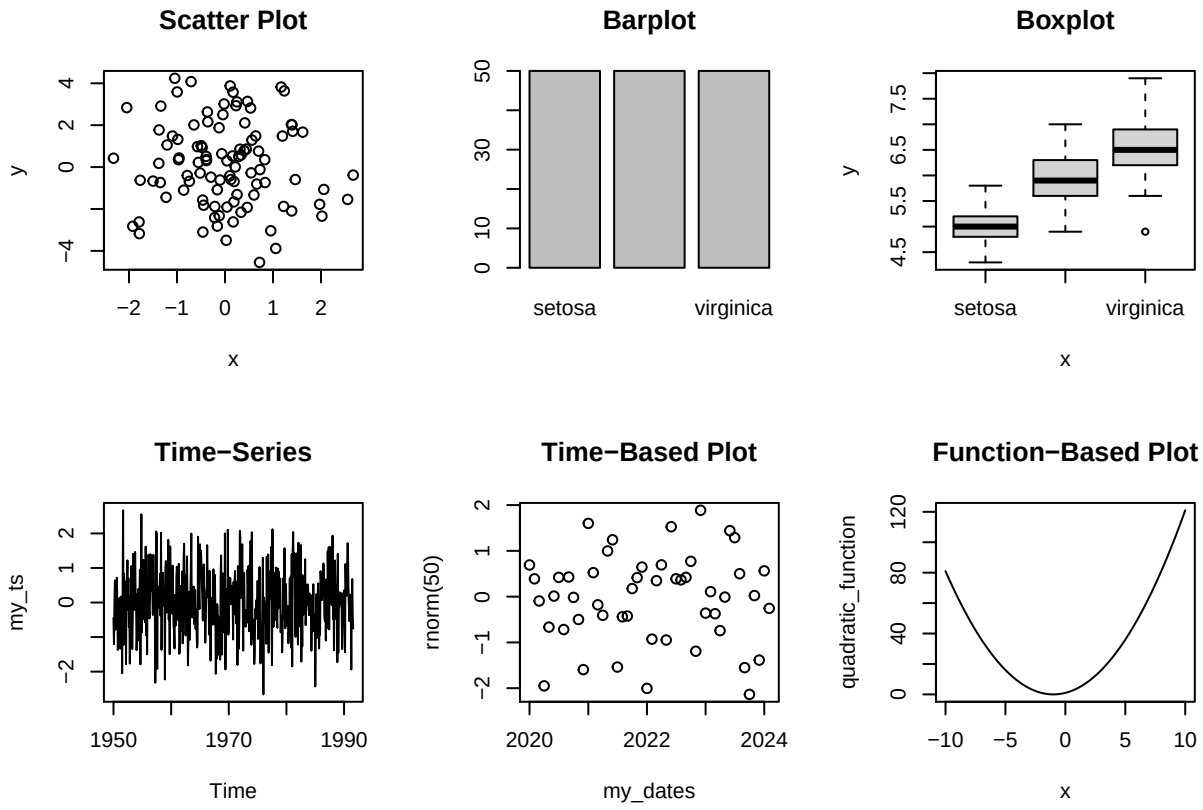
Correlation Matrix

```
plot(iris, main = "Correlation Matrix")
```



Bind Visualization to One 'Canvas'

```r
# build plot canvas structure 2 by 3
par(mfrow = c(2,3))

# scatter plot
set.seed(1000)
x <- rnorm(100)
y <- 2*rnorm(100) + rnorm(100)
plot(x,y, main = "Scatter Plot")

# barplot
plot(iris$Species, main = "Barplot")

# boxplot
plot(x = iris$Species, y = iris$Sepal.Length, main = "Boxplot")

# time-series plot
set.seed(1000)
my_ts <- ts(data = rnorm(500), start = c(1950, 1), frequency = 12)
plot(my_ts, main = "Time-Series")

# time-based plot
my_dates <- seq(as.Date("2020-01-01"), by = "month", length = 50)
plot(my_dates, rnorm(50), main = "Time-Based Plot")

# function-based plot
quadratic_function <- function(x) x^2+2*x+1
plot(quadratic_function, -10, 10, main = "Function-Based Plot")
```


Scatter Plot
y
x
Barplot
setosa
virginica
Boxplot
y
setosa
virginica
x
Time−Series
my_ts
Time
Time−Based Plot
rnorm(50)
my_dates
Function−Based Plot
quadratic_function
x

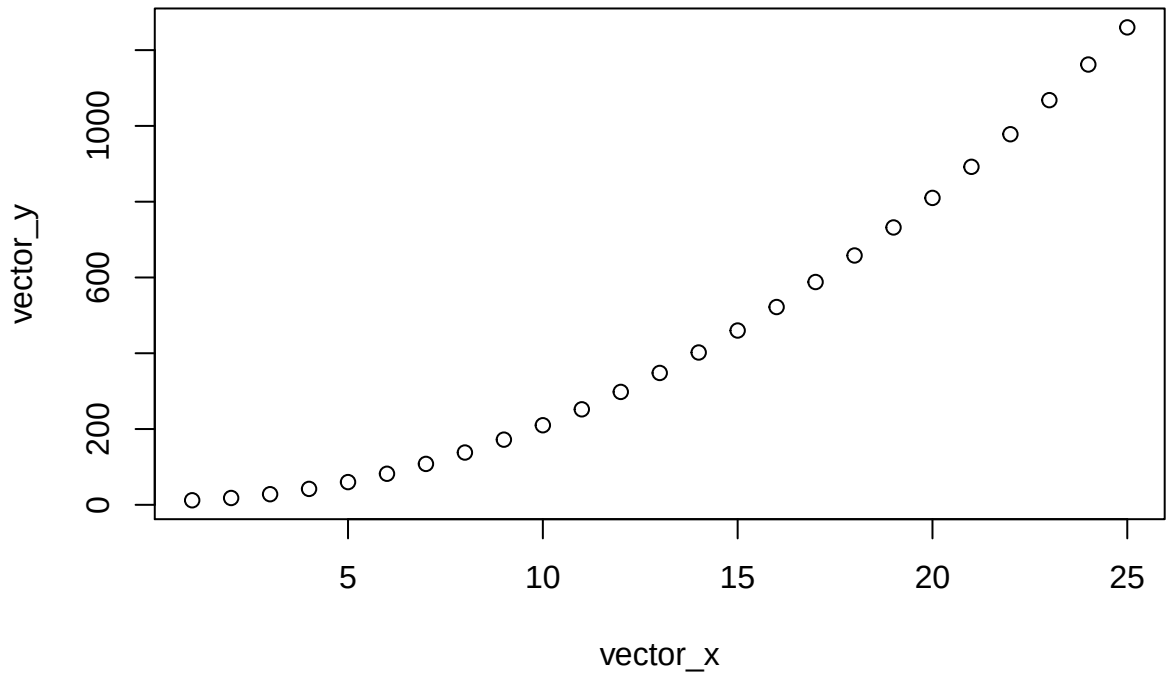
Scatter Plot and Line Plot

Init Values

```
# assign x with vector valued from 1 to 25
vector_x <- 1:25
# assign y with vector valued by 2 * x^2 + 10
vector_y <- 2*vector_x^2 + 10
```

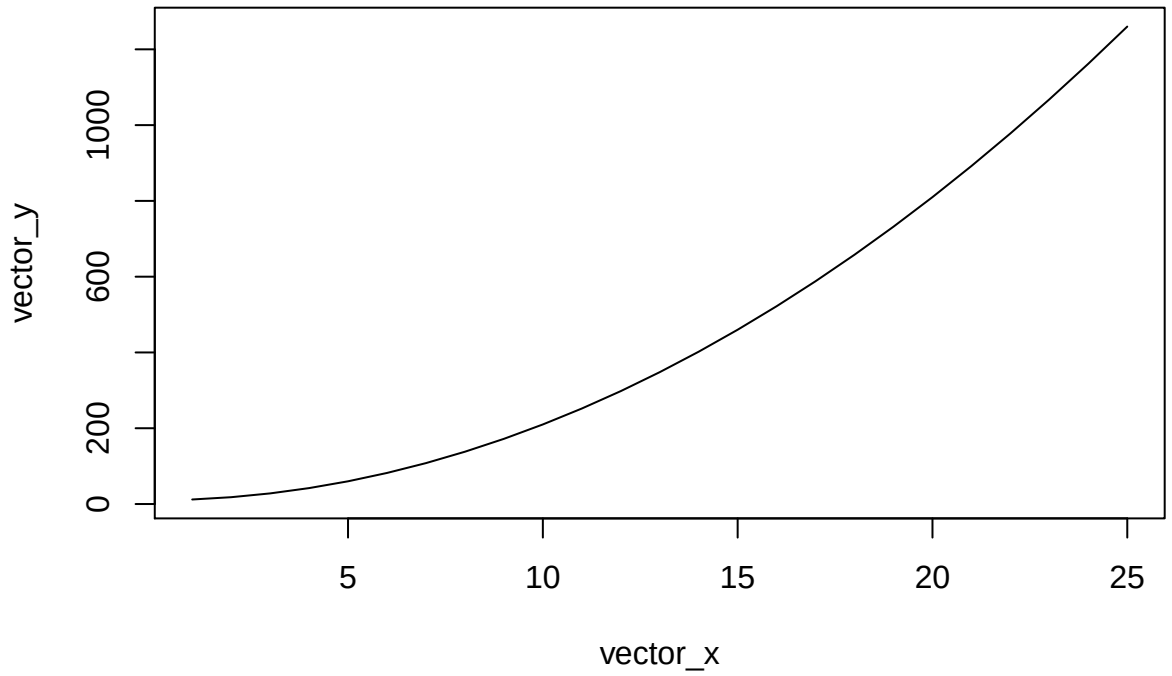
Scatter Plot (Default)

```
plot(x = vector_x, y = vector_y)
```



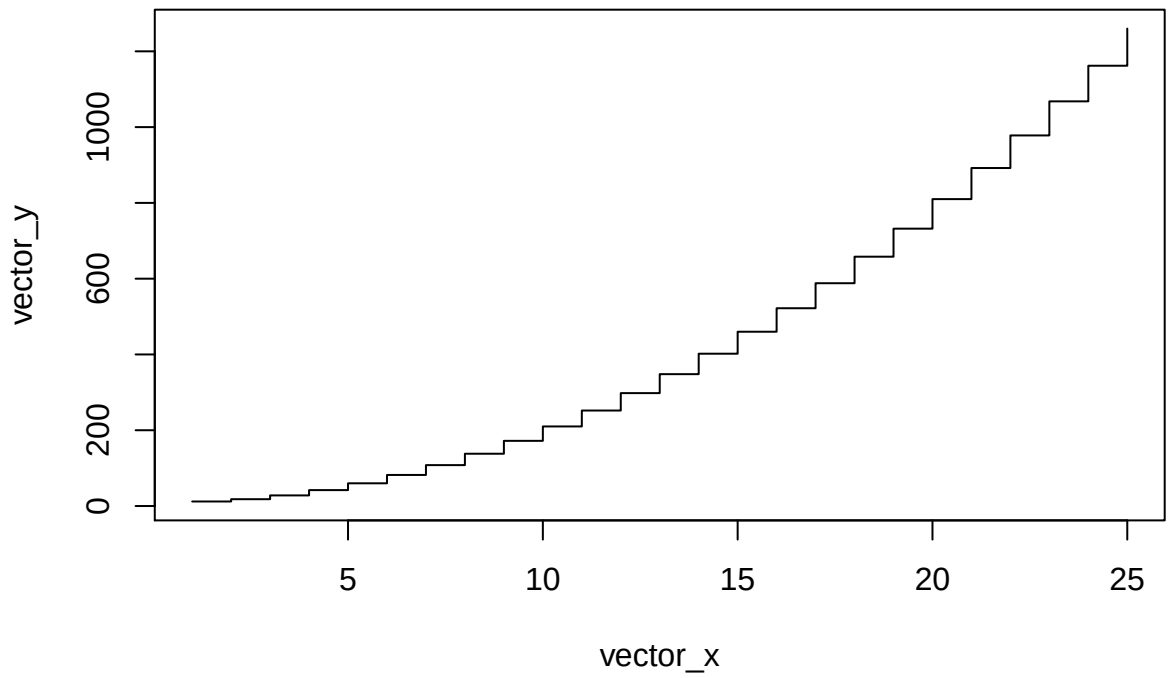
Line Plot

```
plot(x = vector_x, y = vector_y, type = "l")
```



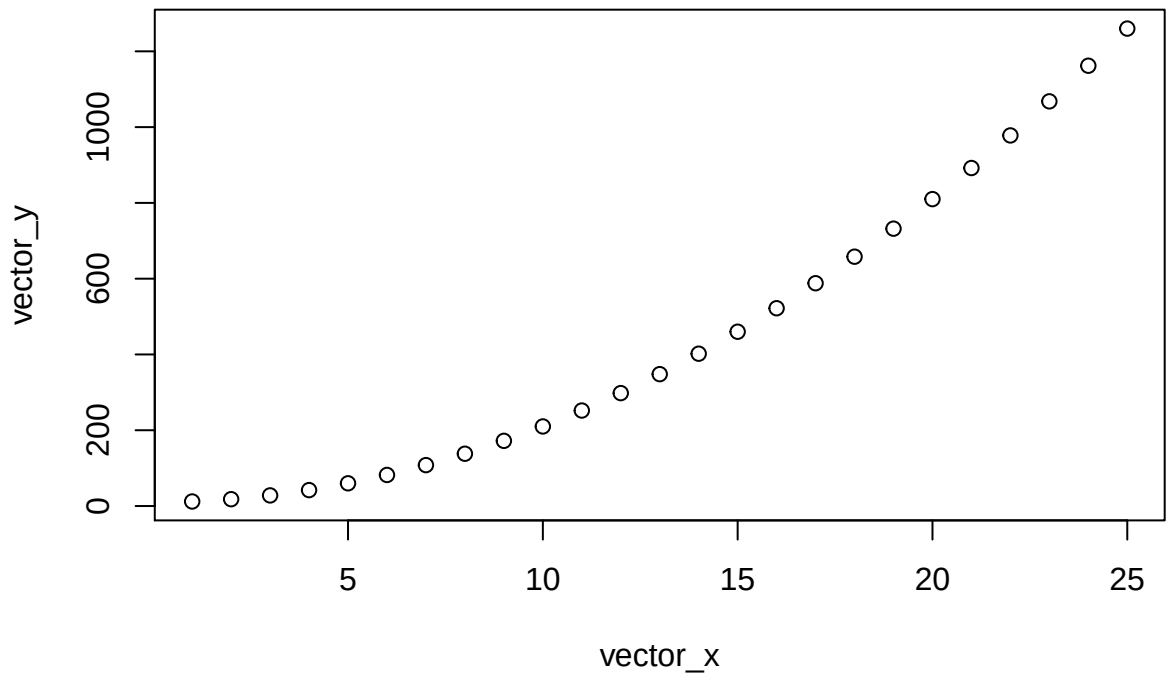
Stairs Plot

```
plot(x = vector_x, y = vector_y, type = "s")
```



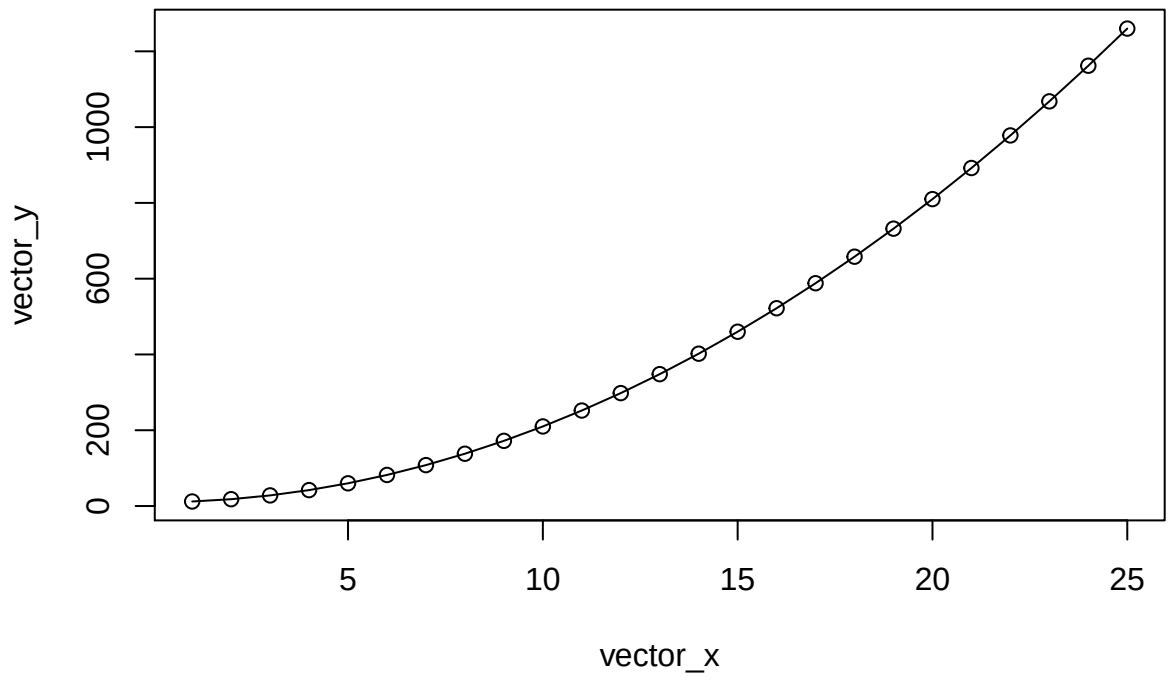
Point Plot

```
plot(x = vector_x, y = vector_y, type = "p")
```



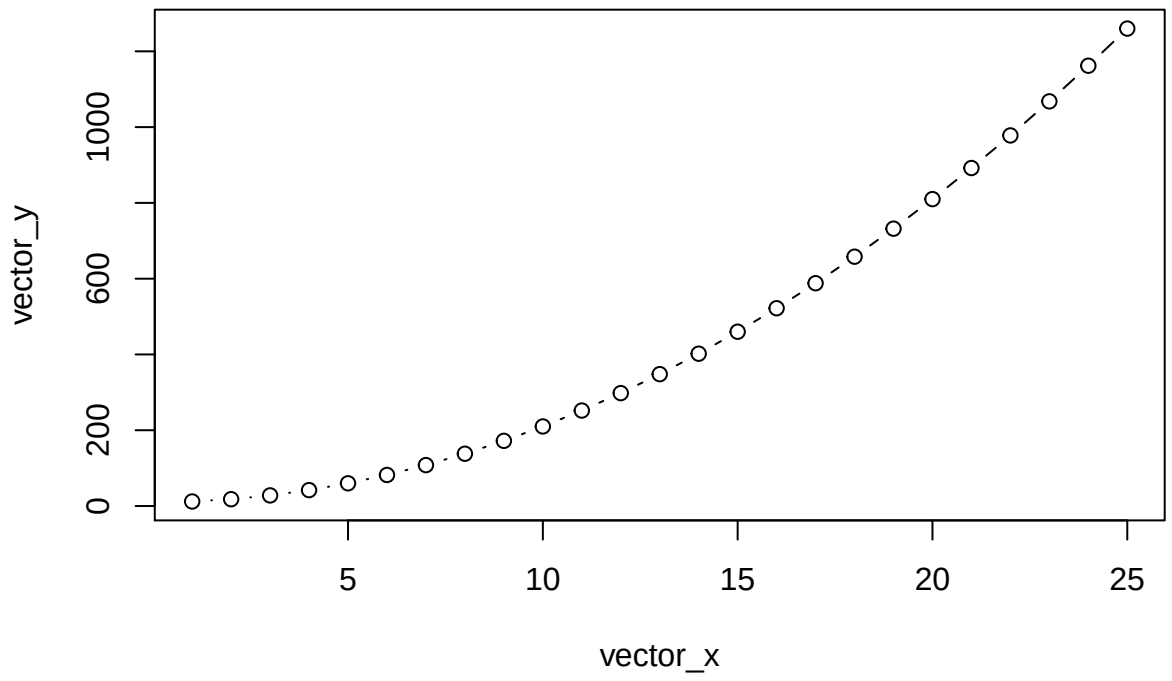
Point and Line 1

```
plot(x = vector_x, y = vector_y, type = "o")
```



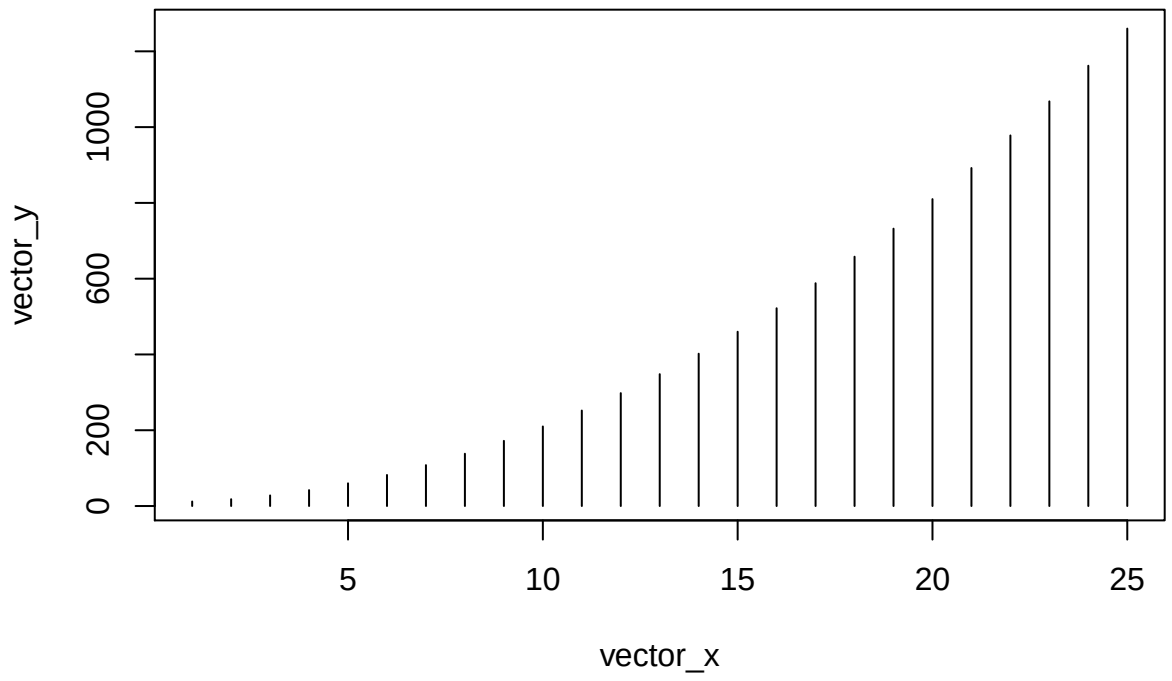
Point and Line 2

```
plot(x = vector_x, y = vector_y, type = "b")
```



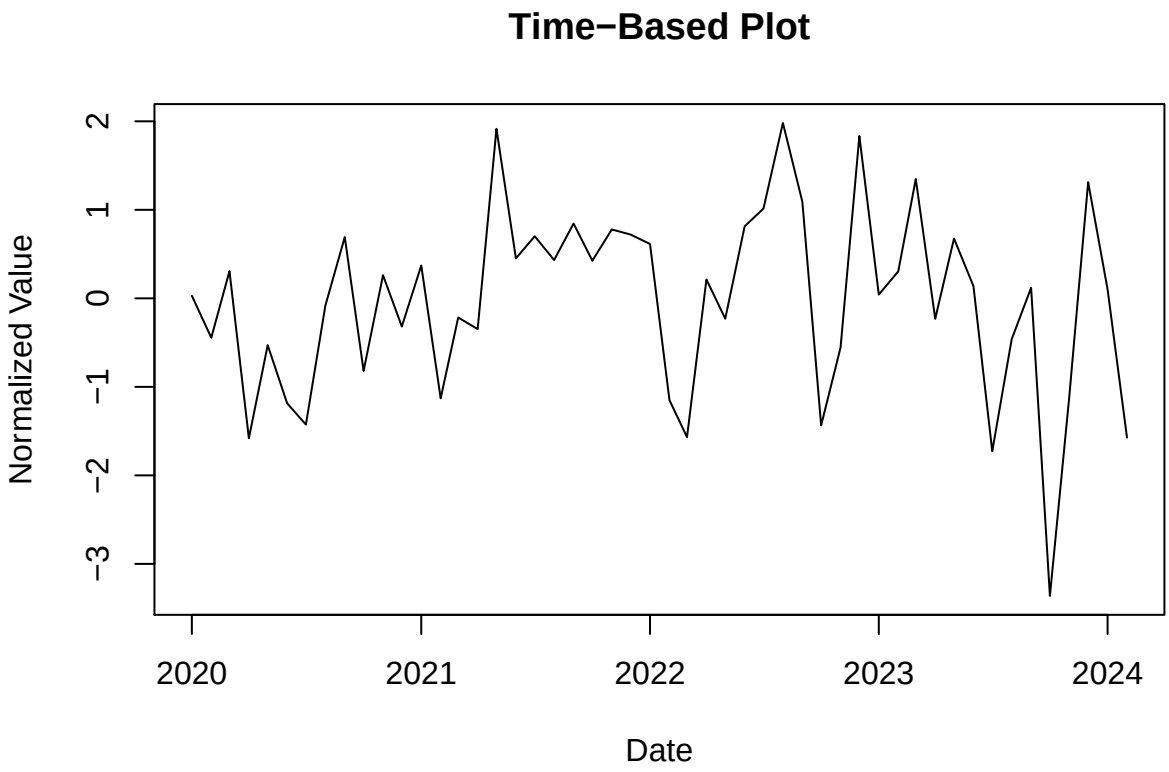
Line Histogram

```
plot(x = vector_x, y = vector_y, type = "h")
```



Time-Based Plot

```r
# assign my_dates with date from 2020-01-01 to next 50 month
my_dates <- seq(as.Date("2020-01-01"), by = "month", length = 50)

# plot
plot(
  # add x values
  my_dates,
  # add y values
  rnorm(50),
  # add plot title
  main = "Time-Based Plot",
  # plot type
  type = "l",
  # change y label
  ylab = "Normalized Value",
  # change x label
  xlab = "Date"
)
```



E-commerce Data

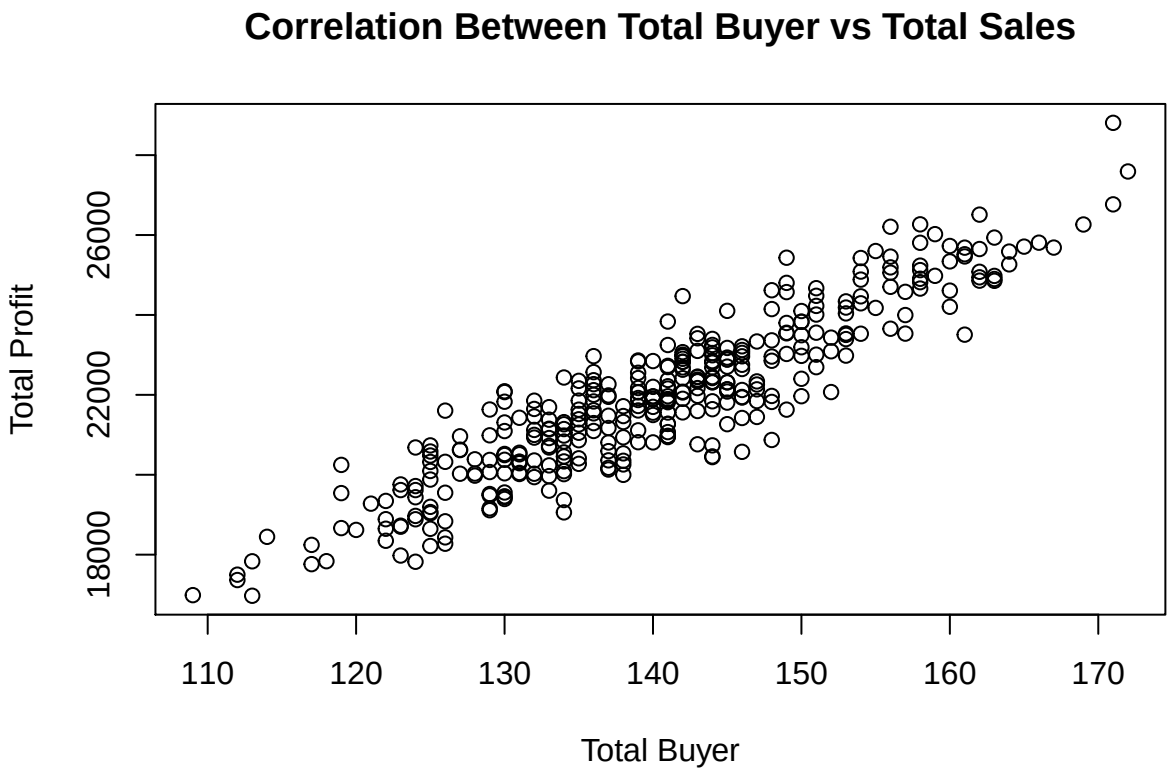
Correlation Between Sales and Profit

Initialize Data

```
# from raw_data
raw_data %>%
  # group data by order date
  group_by(order_date) %>%
  # summarise data
  summarise(
    # find total buyer
    total_buyer = n_distinct(customer_id),
    # find total sales
    total_sales = sum(sales),
    # find total profit
    total_profit = sum(profit),
    # find total cost
    total_cost = sum(shipping_cost)
  ) -> agg_data
```

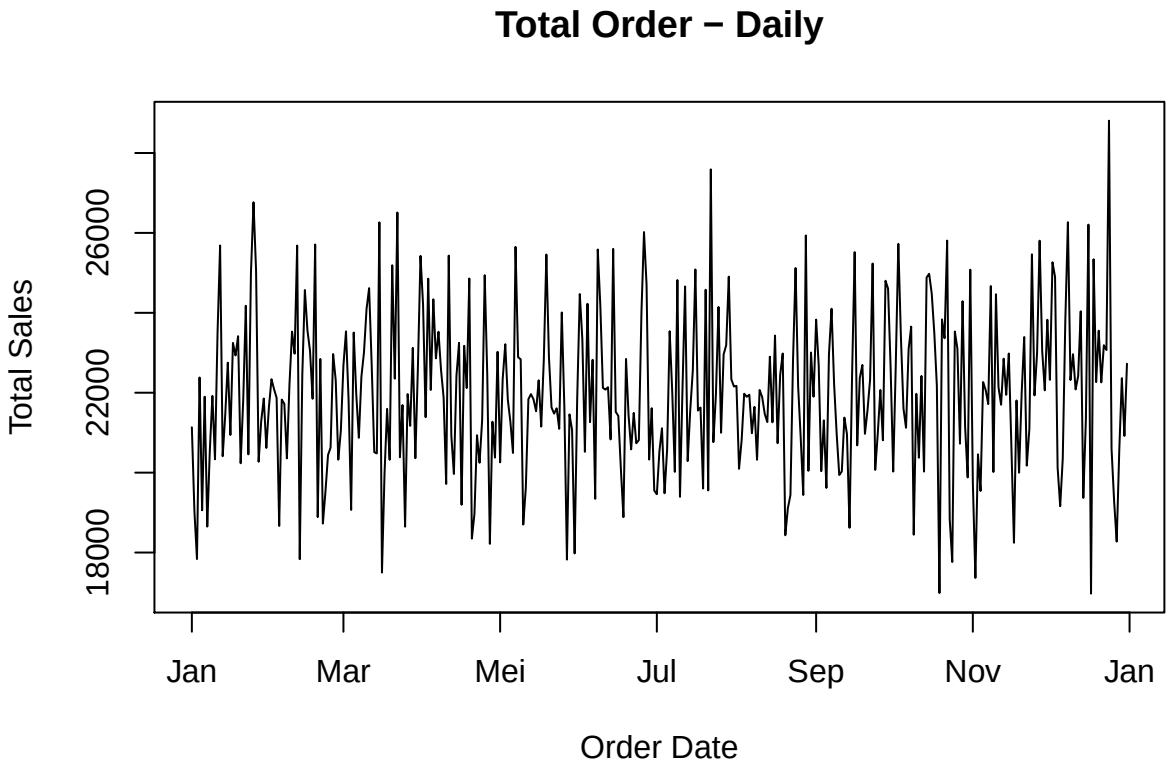
Build Scatter Plot

```
plot(
  x = agg_data$total_buyer,
  y = agg_data$total_sales,
  xlab = "Total Buyer",
  ylab = "Total Profit",
  main = "Correlation Between Total Buyer vs Total Sales"
)
```



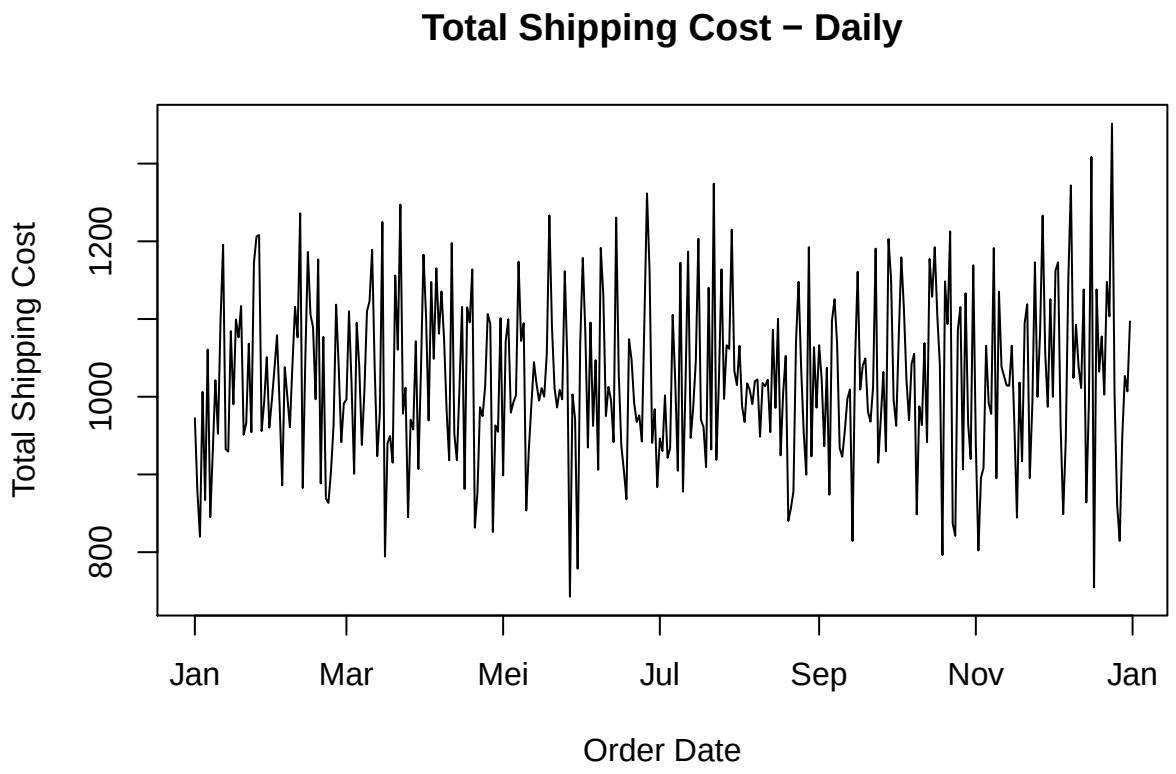
Find Total Sales on Daily Basis

```
plot(
  x = agg_data$order_date,
  y = agg_data$total_sales,
  type = "l",
  main = "Total Order - Daily",
  xlab = "Order Date",
  ylab = "Total Sales"
)
```



Find Total Shipping Cost on Daily Basis

```
plot(
  x = agg_data$order_date,
  y = agg_data$total_cost,
  type = "l",
  main = "Total Shipping Cost - Daily",
  xlab = "Order Date",
  ylab = "Total Shipping Cost"
)
```



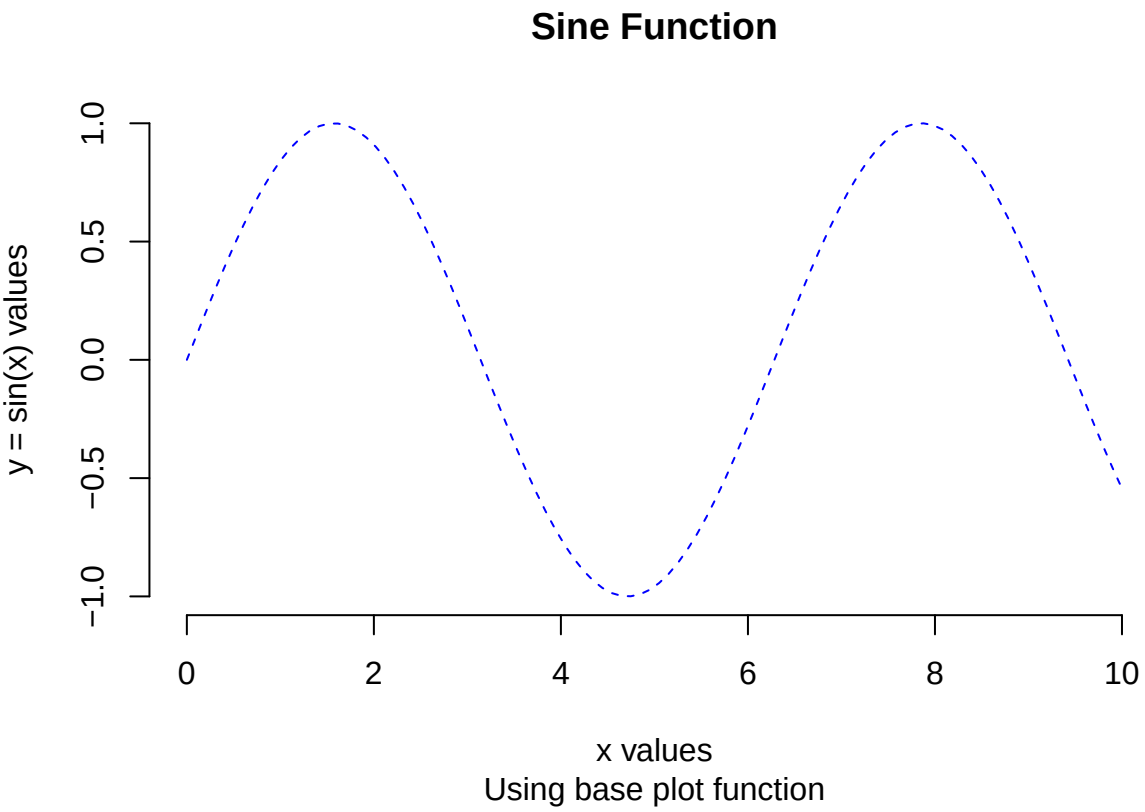
Plot Config

Initialize Value

```r
# assign x_vec with 100 values between 0 to 10
x <- seq(from = 0, to = 10, length.out = 100)
# assign y_sin_vec with sin(x_vec)
y <- sin(x)
```

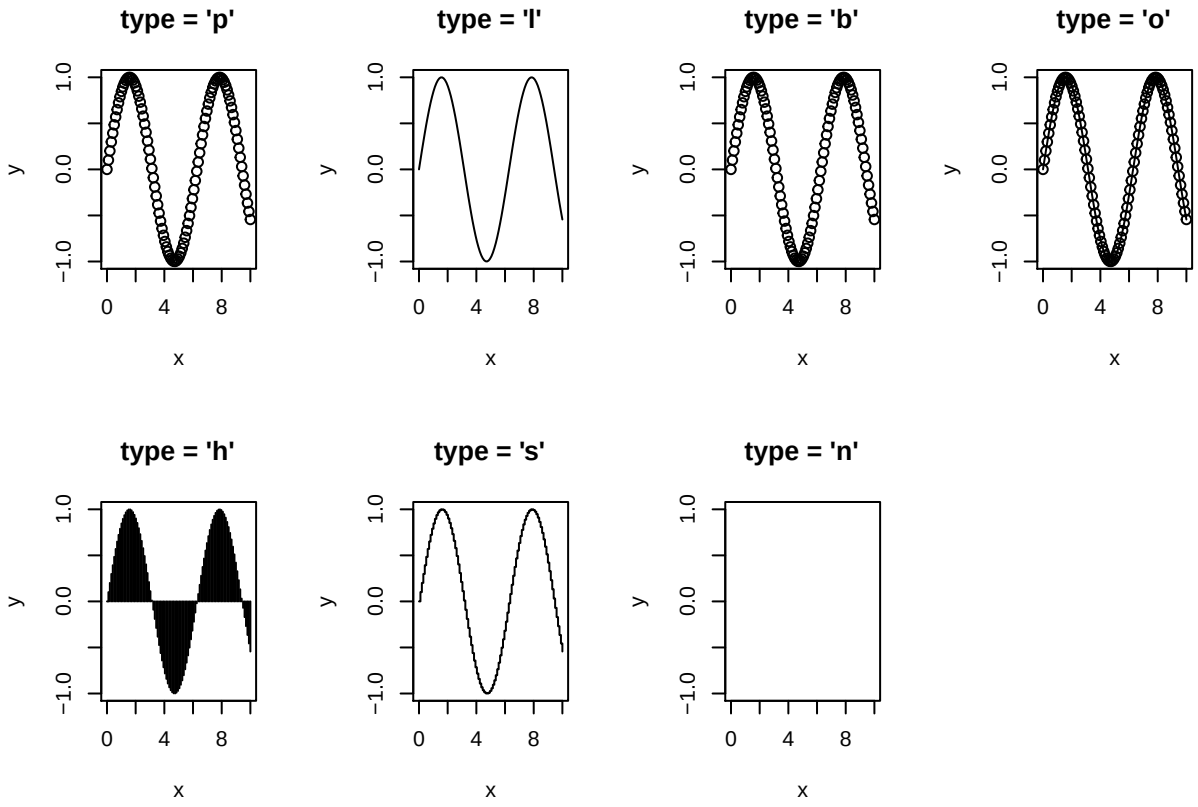
Plot with all basic config

```r
plot(
  x = x,
  y = y,
  # use 'line' type
  type = "l",
  # change x label
  xlab = "x values",
  # change y label
  ylab = "y = sin(x) values",
  # add title
  main = "Sine Function",
  # add subtitle
  sub = "Using base plot function",
  # add line color
  col = "blue",
  bty = "n",
  lty = 2
)
```


Sine Function
y = sin(x) values
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
x values
Using base plot function

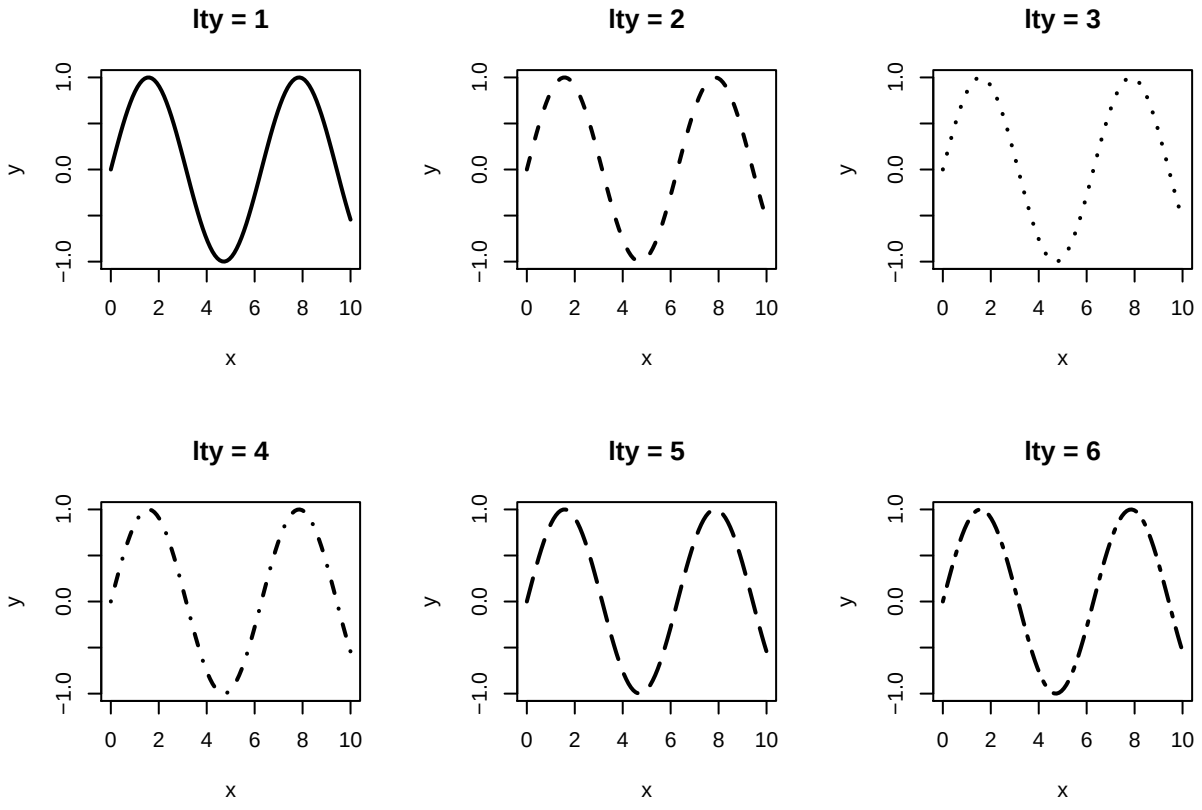
type Parameter

```
par(mfrow = c(2, 4))
plot(x, y, main = "type = 'p'")
plot(x, y, type = "l", main = "type = 'l'")
plot(x, y, type = "b", main = "type = 'b'")
plot(x, y, type = "o", main = "type = 'o'")
plot(x, y, type = "h", main = "type = 'h'")
plot(x, y, type = "s", main = "type = 's'")
plot(x, y, type = "n", main = "type = 'n'")
```



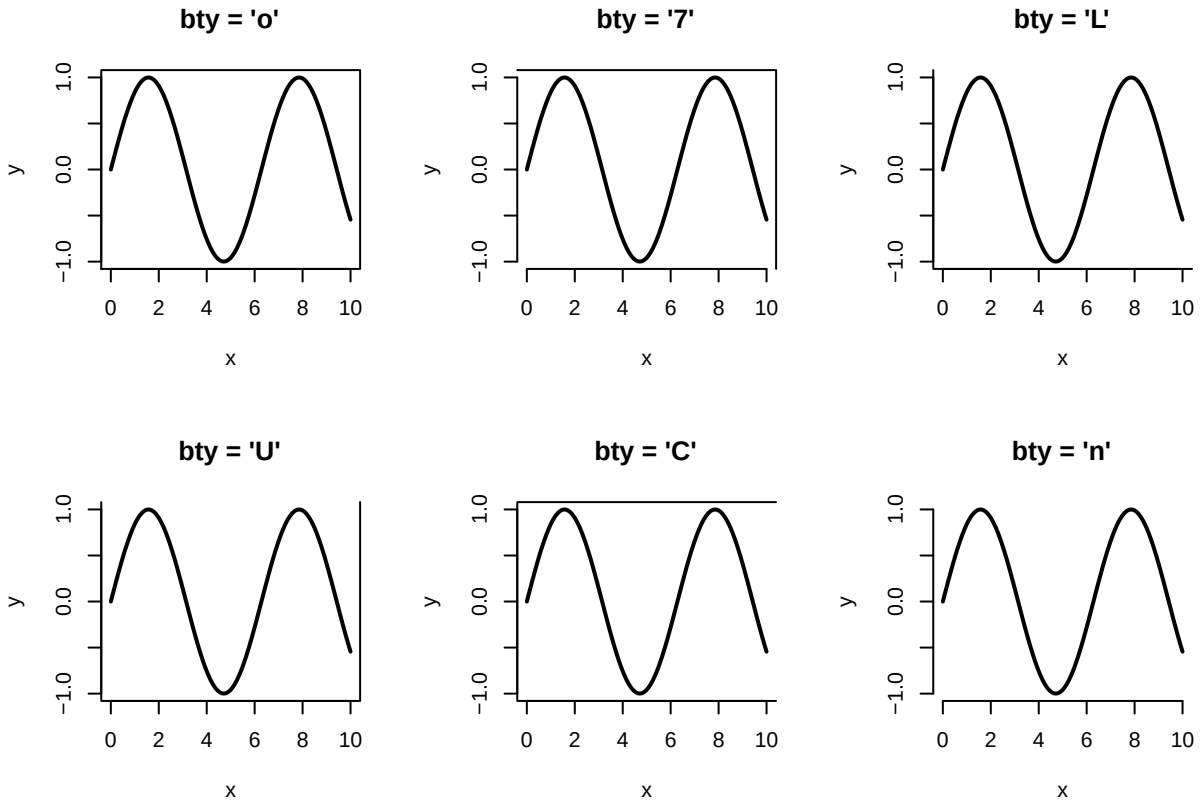
lty Parameter

```
par(mfrow=c(2,3))
plot(x, y, type = "l", lwd = 2, lty = 1, main = "lty = 1")
plot(x, y, type = "l", lwd = 2, lty = 2, main = "lty = 2")
plot(x, y, type = "l", lwd = 2, lty = 3, main = "lty = 3")
plot(x, y, type = "l", lwd = 2, lty = 4, main = "lty = 4")
plot(x, y, type = "l", lwd = 2, lty = 5, main = "lty = 5")
plot(x, y, type = "l", lwd = 2, lty = 6, main = "lty = 6")
```



bty Parameter

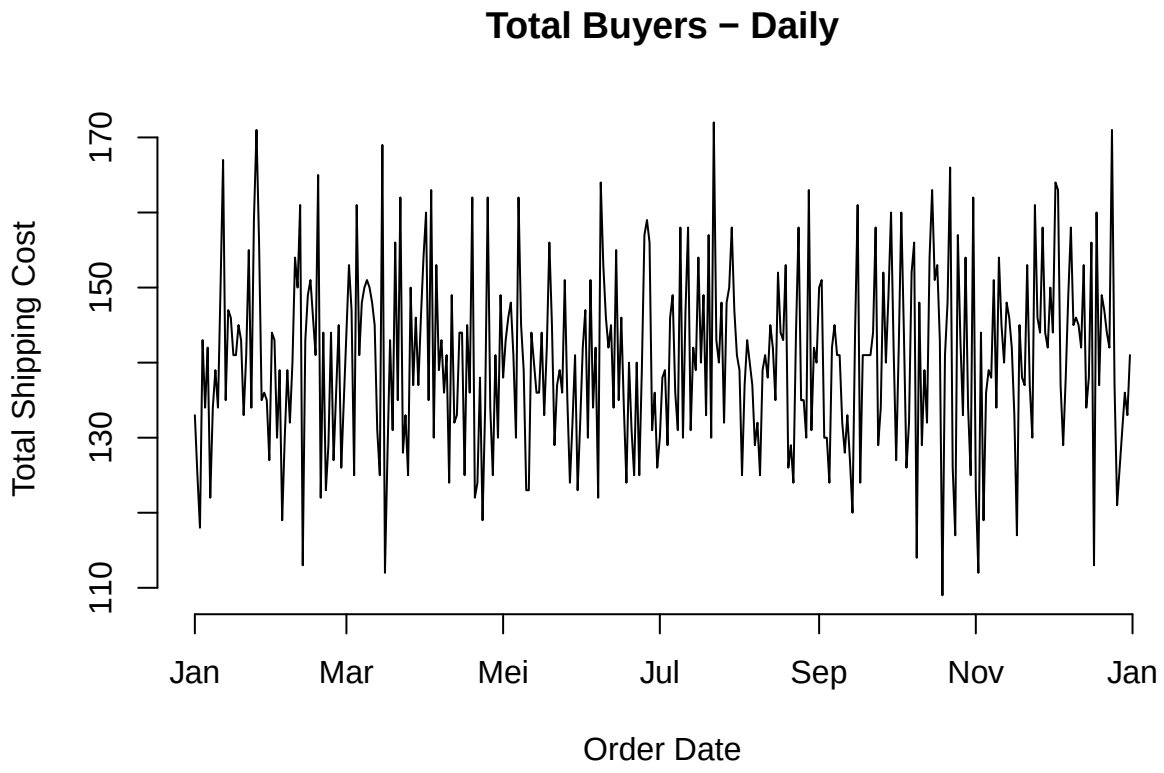
```
par(mfrow = c(2, 3))
plot(x, y, type = "l", lwd = 2, main = "bty = 'o'")
plot(x, y, type = "l", lwd = 2, bty = "7", main = "bty = '7'")
plot(x, y, type = "l", lwd = 2, bty = "L", main = "bty = 'L'")
plot(x, y, type = "l", lwd = 2, bty = "U", main = "bty = 'U'")
plot(x, y, type = "l", lwd = 2, bty = "C", main = "bty = 'C'")
plot(x, y, type = "l", lwd = 2, bty = "n", main = "bty = 'n'")
```



E-Commerce Data

Total Buyers - Daily

```
plot(
  x = agg_data$order_date,
  y = agg_data$total_buyer,
  type = "l",
  main = "Total Buyers - Daily",
  xlab = "Order Date",
  ylab = "Total Shipping Cost",
  bty = "n"
)
```



Other Base Plot Function

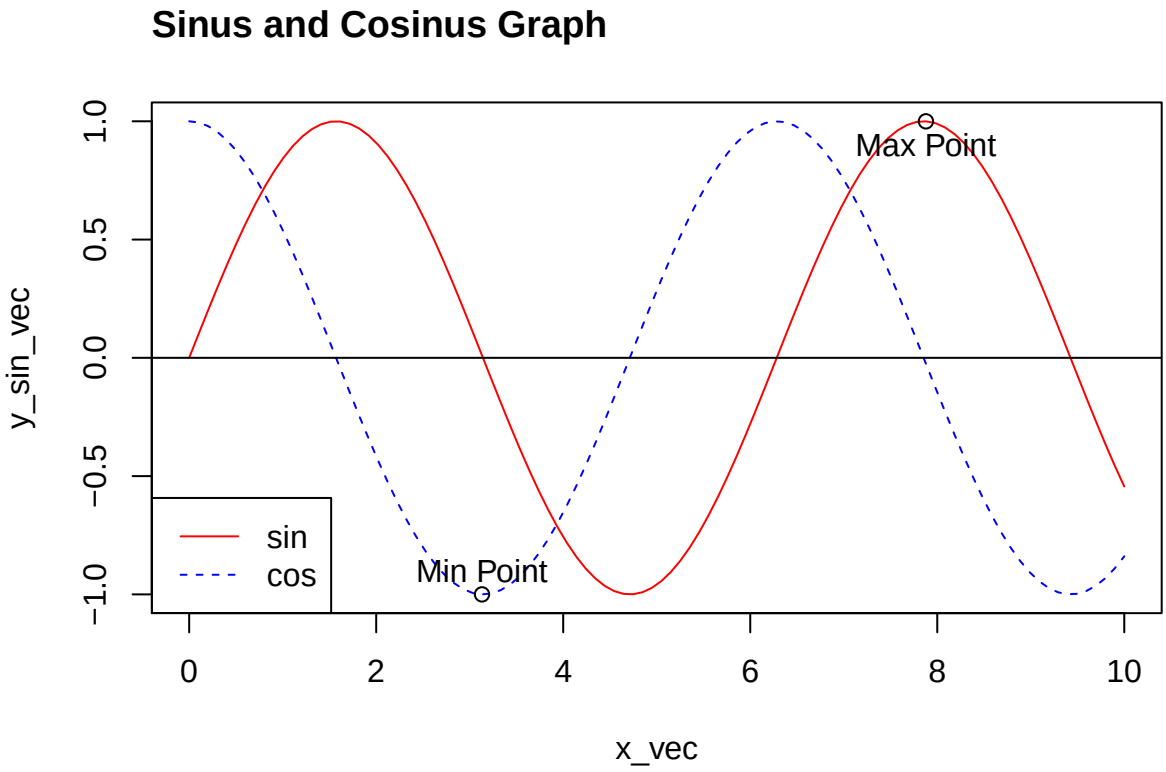
Sine and Cosine Function Graph

Initialize Value

```
x_vec <- seq(from = 0, to = 10, length.out = 100)
y_sin_vec <- sin(x_vec)
y_cos_vec <- cos(x_vec)
```

Create Plot

```
# add line plot of sin(x)
plot(x = x_vec, y = y_sin_vec, type = "l", col = "red", lty = 1)
# add new lines of cos(x)
lines(x = x_vec, y_cos_vec, col = "blue", lty = 2)
# add horizontal line
abline(h = 0)
# add legend
legend("bottomleft", legend = c("sin", "cos"), col = c("red", "blue"), lty = c(1,2))
# add title
title(main = "Sinus and Cosinus Graph", adj = 0)
# add point
points(
  x = x_vec[which.max(y_sin_vec)],
  y = y_sin_vec[which.max(y_sin_vec)]
)
points(
  x = x_vec[which.min(y_cos_vec)],
  y = y_cos_vec[which.min(y_cos_vec)]
)
# add text
text(
  x = x_vec[which.max(y_sin_vec)],
  y = y_sin_vec[which.max(y_sin_vec)] - 0.1,
  labels = "Max Point"
)
text(
  x = x_vec[which.min(y_cos_vec)],
  y = y_cos_vec[which.min(y_cos_vec)] + 0.1,
  labels = "Min Point"
)
```


Sinus and Cosinus Graph
y_sin_vec
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
x_vec
sin
cos
Max Point
Min Point

Regression with Trendline

Initialize Value

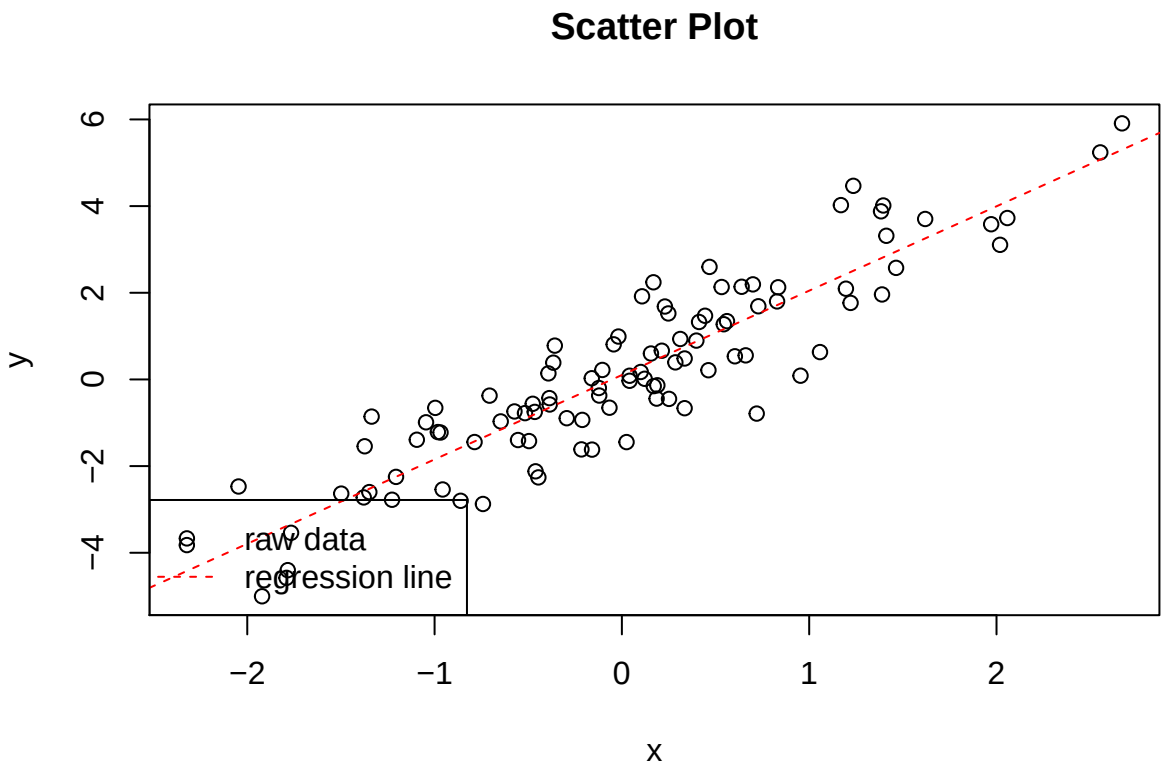
```
set.seed(1000)
x <- rnorm(100)
y <- 2*x + rnorm(100)
```

Create Regression Model

```
linear_regresion_model <- lm(y ~ x)
```

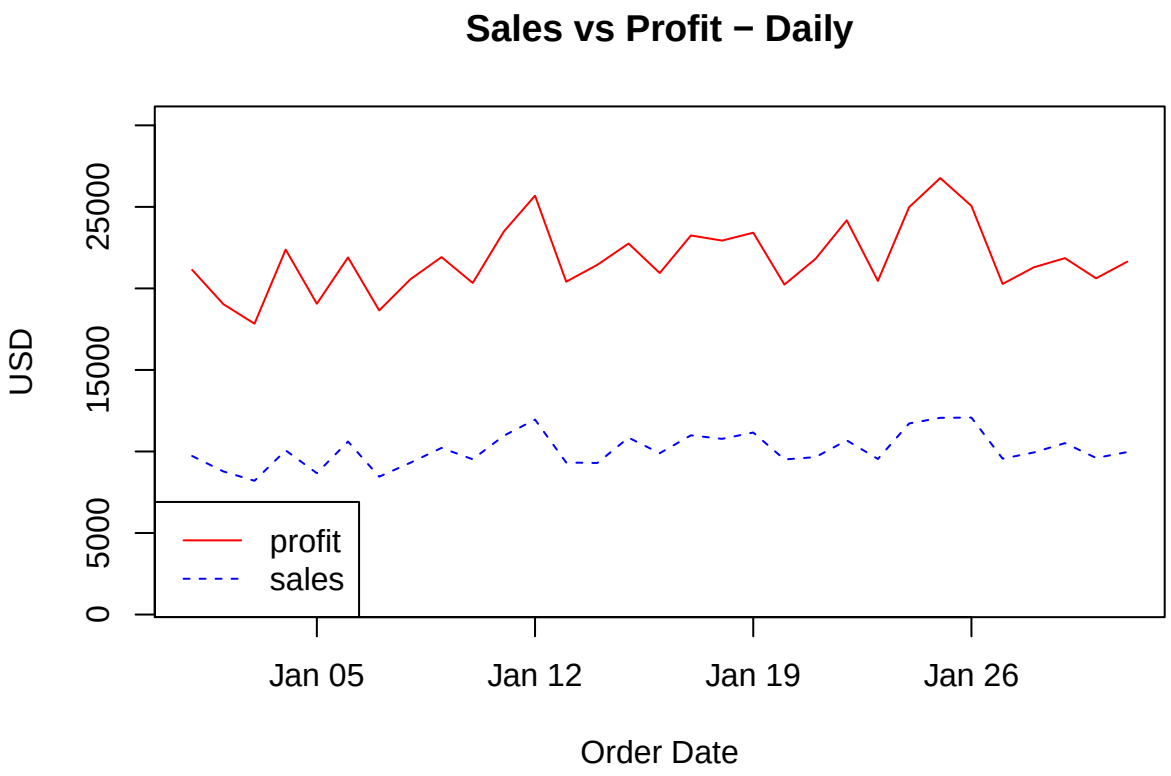
Create Plot

```
plot(x,y, main = "Scatter Plot")
abline(linear_regresion_model, col = "red", lty = 2)
# add legend
legend(
  "bottomleft",
  legend = c("raw data", "regression line"),
  col = c("black", "red"),
  pch = c(1, NA),
  lty = c(NA, 2)
)
```



Sales vs Profit

```
raw_data %>%
  filter(order_date >= "2015-01-01", order_date <= "2015-01-31") %>%
  group_by(order_date) %>%
  summarise(
    total_profit = sum(profit),
    total_sales = sum(sales)
  ) -> agg_jan_data

plot(
  x = agg_jan_data$order_date,
  y = agg_jan_data$total_sales,
  type = "l",
  col = "red",
  lty = 1,
  # add ylimit
  ylim = c(1000,30000),
  xlab = "Order Date",
  ylab = "USD"
)
# add new lines of cos(x)
lines(
  x = agg_jan_data$order_date,
  y = agg_jan_data$total_profit,
  col = "blue",
  type = "l",
  lty = 2
)

# add legend
legend(
  "bottomleft",
  legend = c("profit", "sales"),
  col = c("red", "blue"),
  lty = c(1,2)
)

# add title
title(main = "Sales vs Profit - Daily")
```


Sales vs Profit − Daily
USD
0
5000
15000
25000
profit
sales
Jan 05
Jan 12
Jan 19
Jan 26
Order Date

Histogram and Density Plot

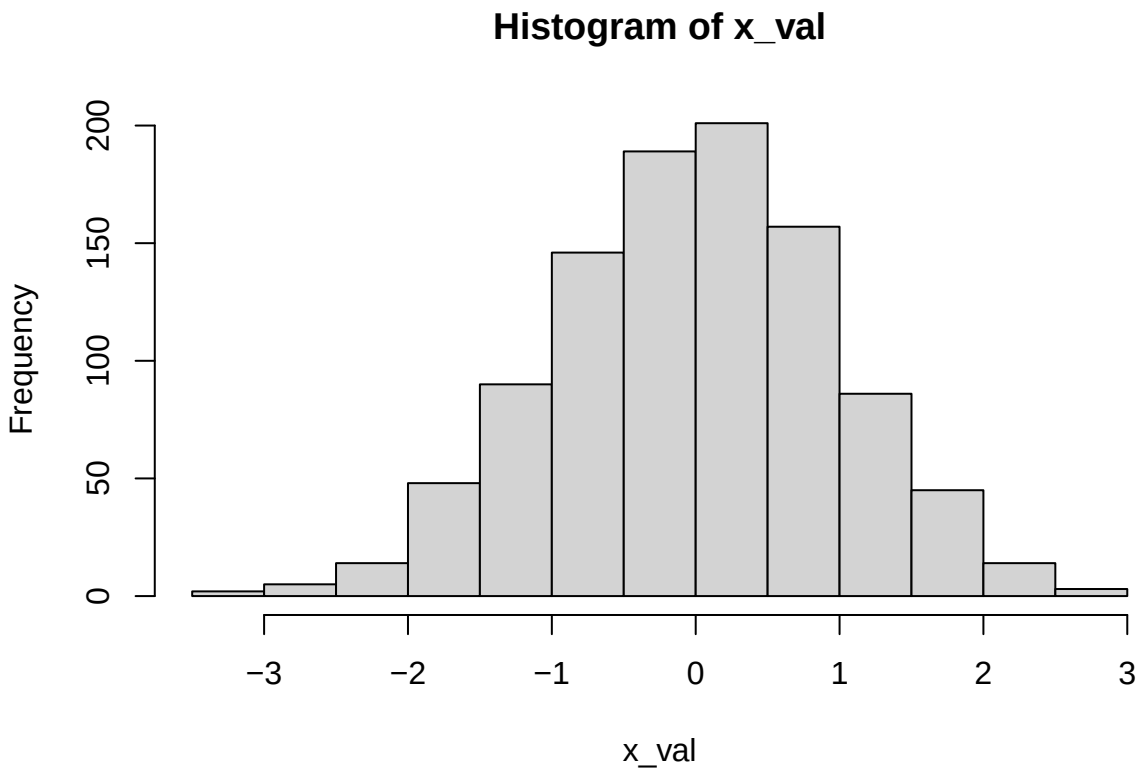
Initialize Value

```
# set seed number
set.seed(1000)
# assign x_val with random value from normal distribution
x_val <- rnorm(1000)
```

Histogram

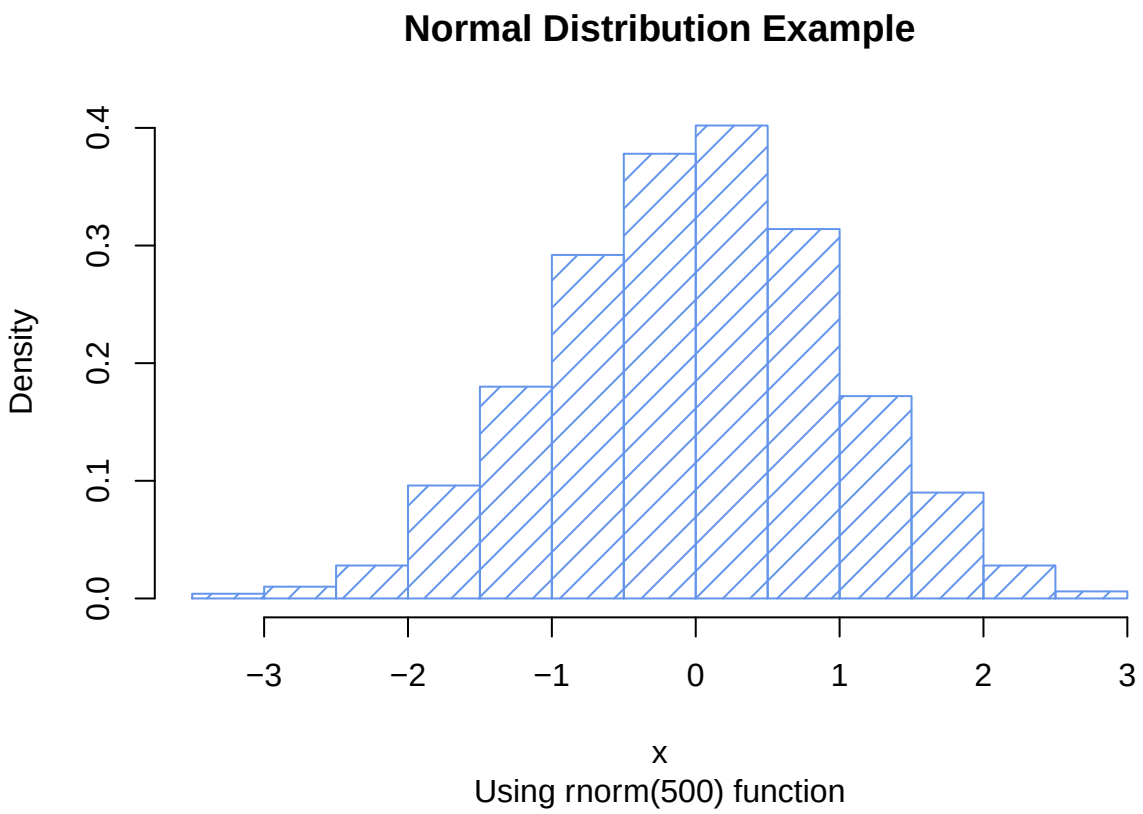
Basic Histogram

```
hist(x_val)
```



Histogram with Density and Shading

```
hist(
  # add values
  x_val,
  # add breaks
  breaks = 10,
  # use frequency?
  freq = FALSE,
  # add shading
  density = 10,
  # add title
  main = "Normal Distribution Example",
  # add subtitle
  sub = "Using rnorm(500) function",
  # change x label
  xlab = "x",
  # add color
  col = "cornflowerblue"
)
```

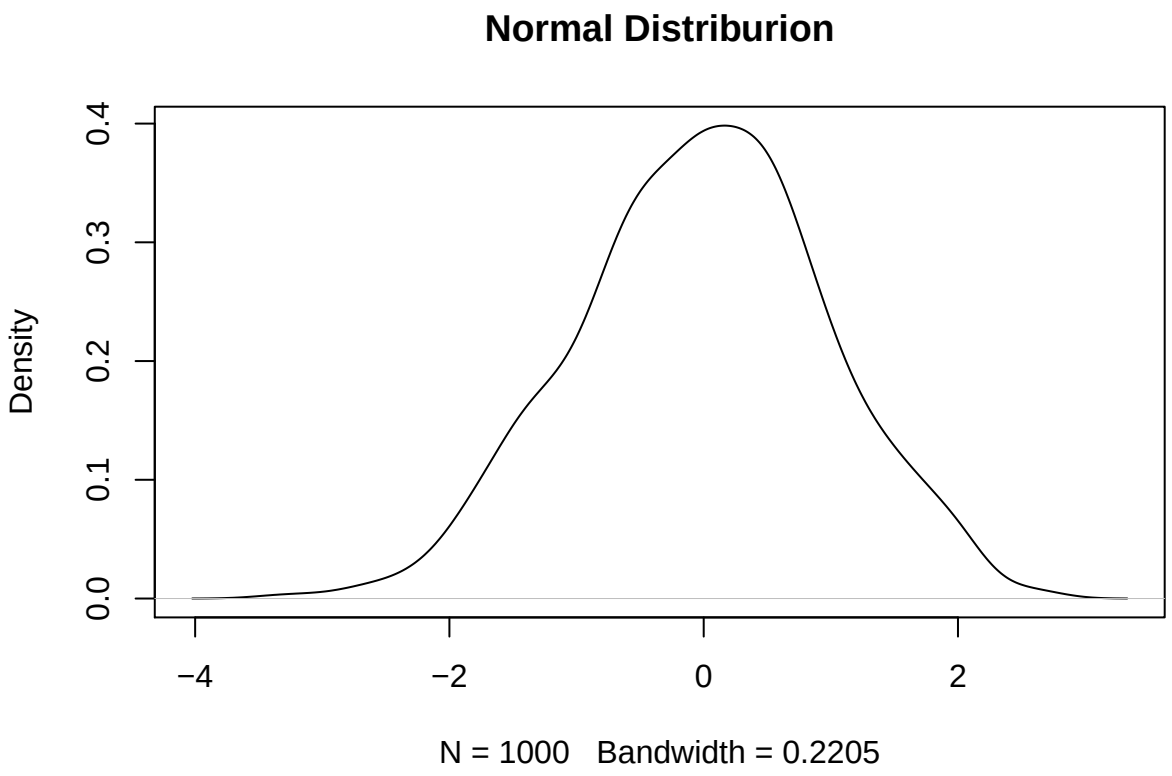


Notes

breaks parameter use Sturges' formula to create number of histogram bins then prettify the plot on background. That's why the number of breaks is not always equal with what plot show to us.

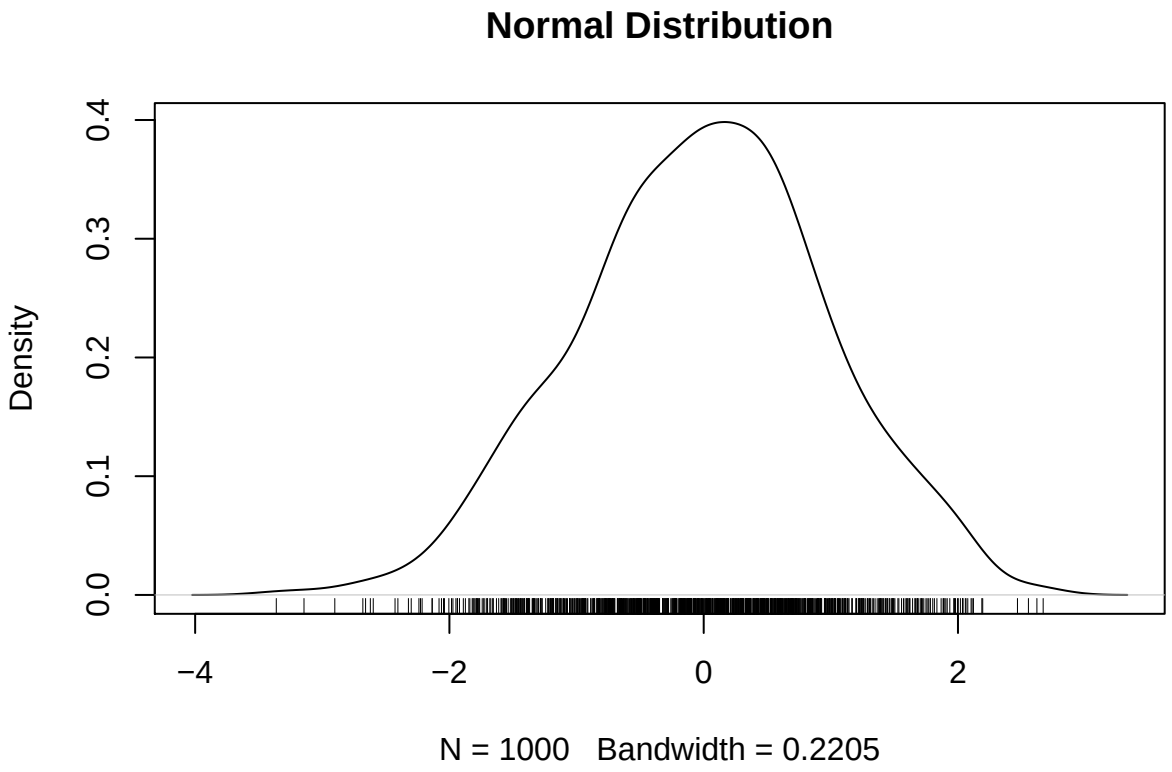
Density Plot

```
# create density plot
plot(density(x = x_val), main = "Normal Distriburion")
```



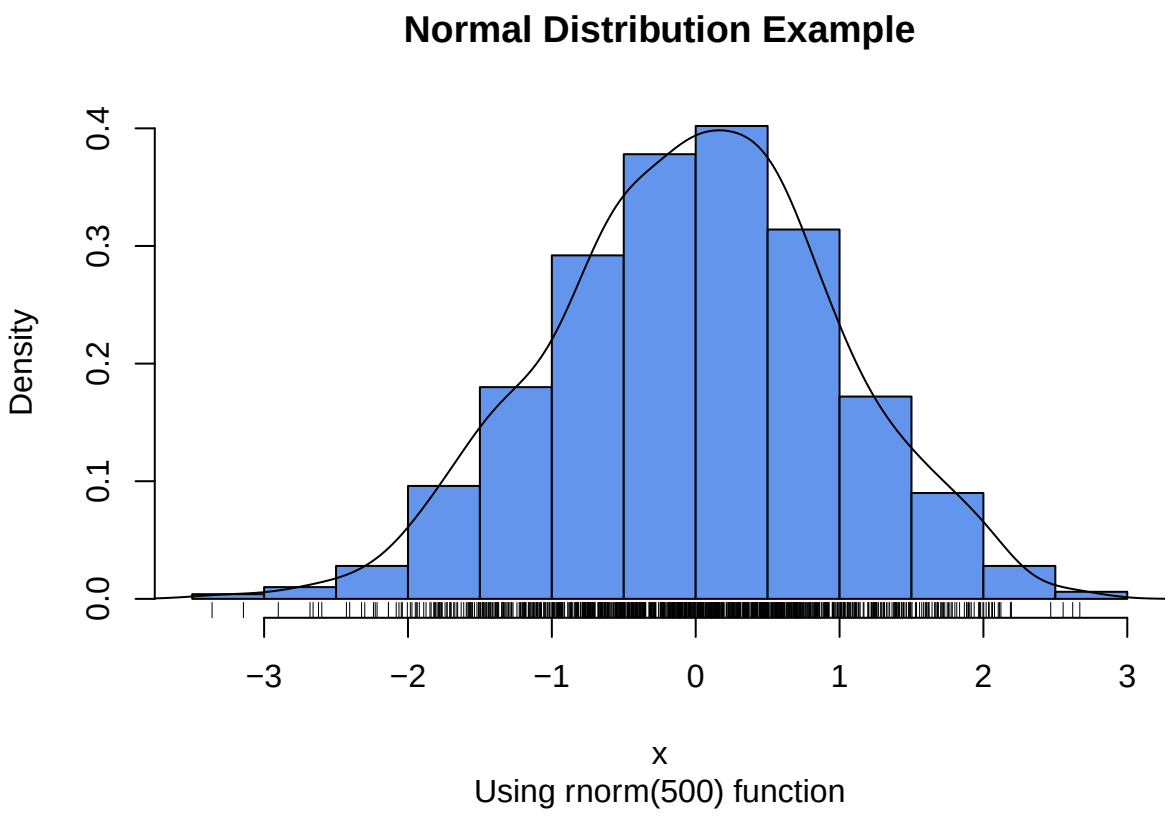
Density Plot With Point Density

```
plot(density(x_val), main = "Normal Distribution")
# add 1-d point representation with rug() function
rug(x_val)
```



Histogram with Density Lines and Point Density

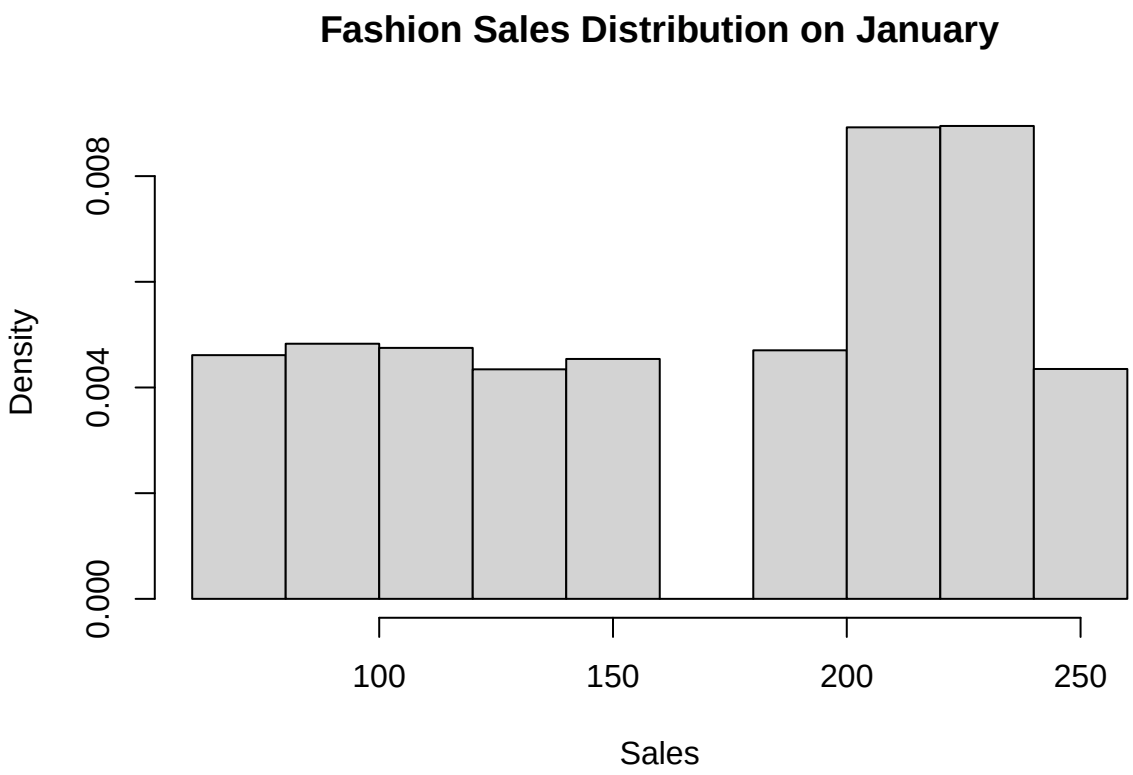
```r
# histogram
hist(
  x_val,
  # add breaks
  breaks = 10,
  # use probability
  freq = FALSE,
  # add title
  main = "Normal Distribution Example",
  # add subtitle
  sub = "Using rnorm(500) function",
  # change x label
  xlab = "x",
  # add color
  col = "cornflowerblue"
)

# add density line
lines(
  x = density(x_val)
)

# add point density
rug(x = x_val)
```



E-commerce Data

Q1 Fashion Sales Distribution

```
# from raw_data
raw_data %>%
  # filter Q1 data and only Fashion product category
  filter(
    order_date >= "2015-01-01", order_date <= "2015-03-31",
    product_category == "Fashion"
  ) %>%
  # pull sales data
  pull(sales) -> fashion_q1_sales

hist(
  x = fashion_q1_sales,
  main = "Fashion Sales Distribution on January",
  breaks = 10,
  freq = F,
  xlab = "Sales"
)
```

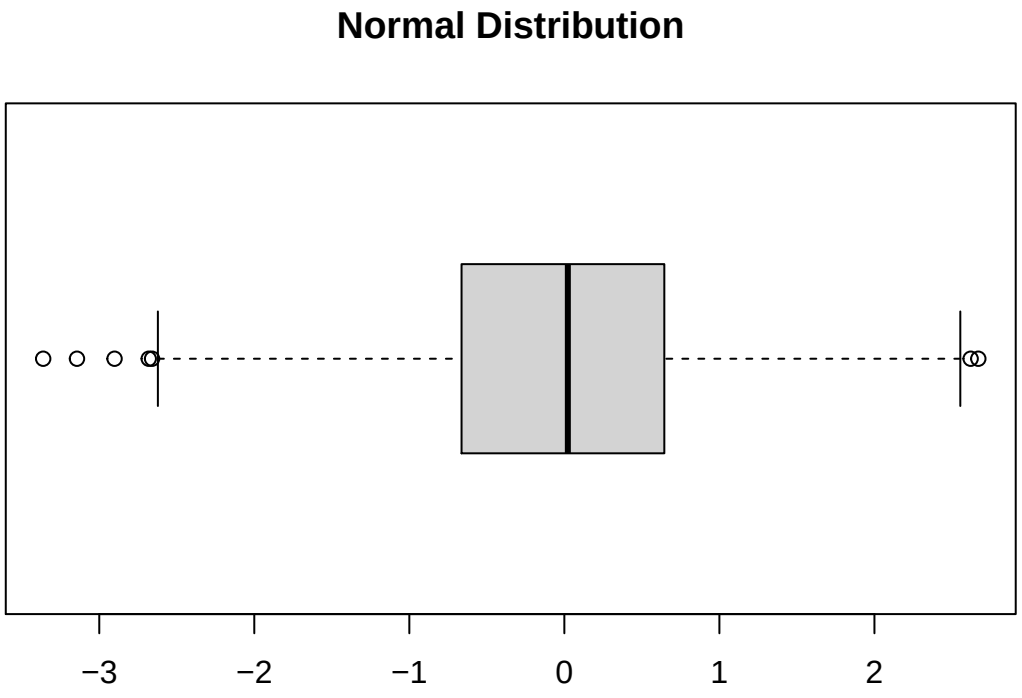


Boxplot and Violin Plot

Boxplot

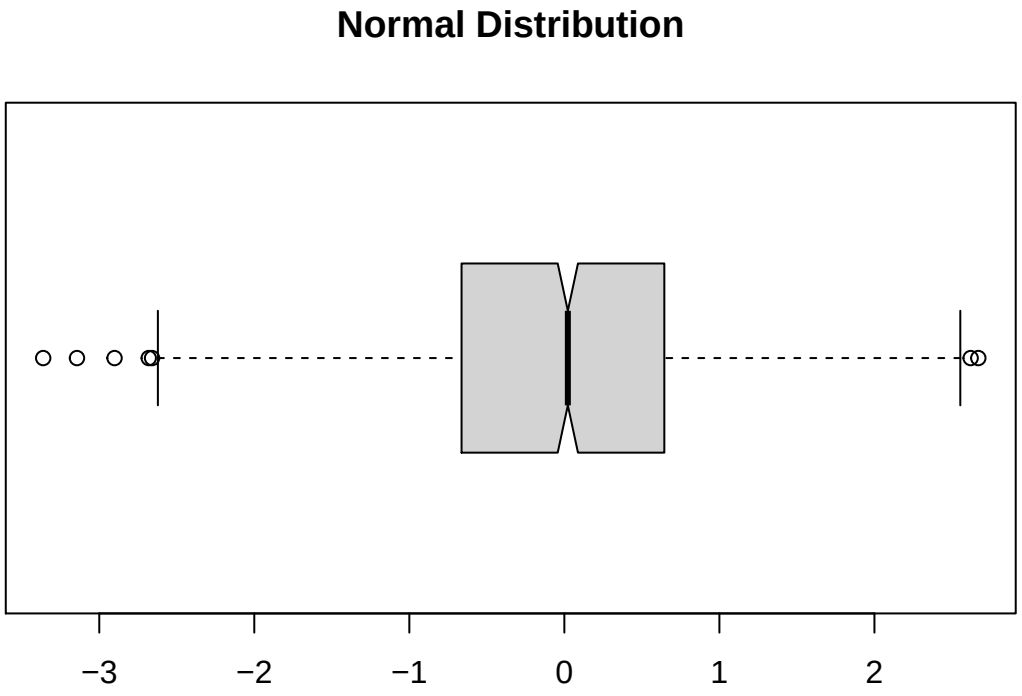
Basic Boxplot

```
boxplot(
  x_val,
  # plot horizontally?
  horizontal = TRUE,
  # add title
  main = "Normal Distribution"
)
```



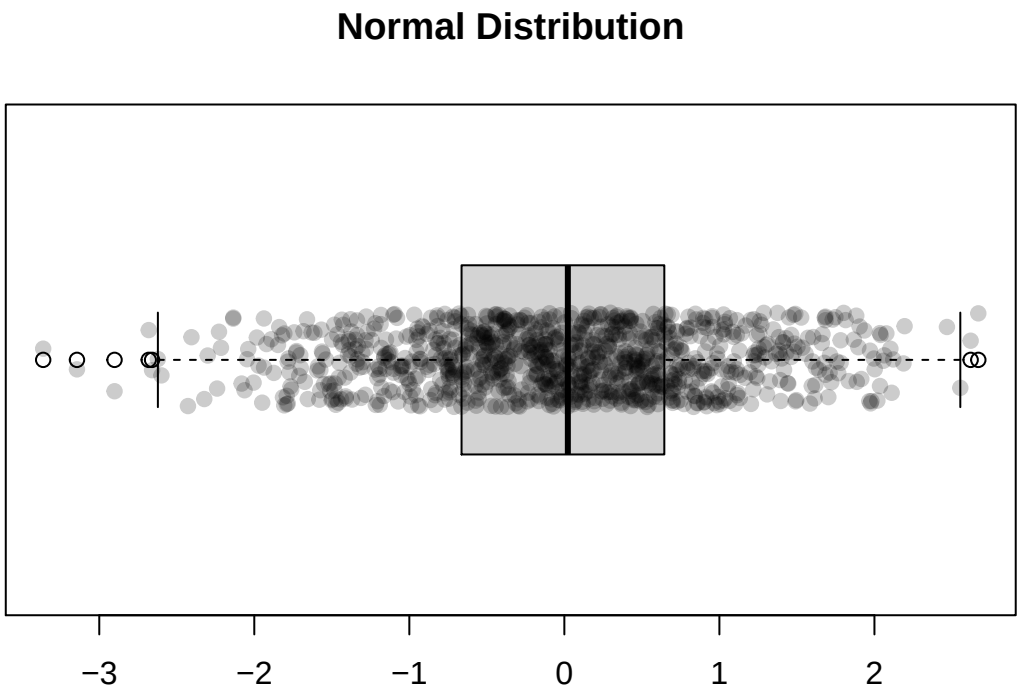
Boxplot with Notch

```
boxplot(
  x_val,
  # plot horizontally?
  horizontal = TRUE,
  # add title
  main = "Normal Distribution",
  # add notch?
  notch = T
)
```



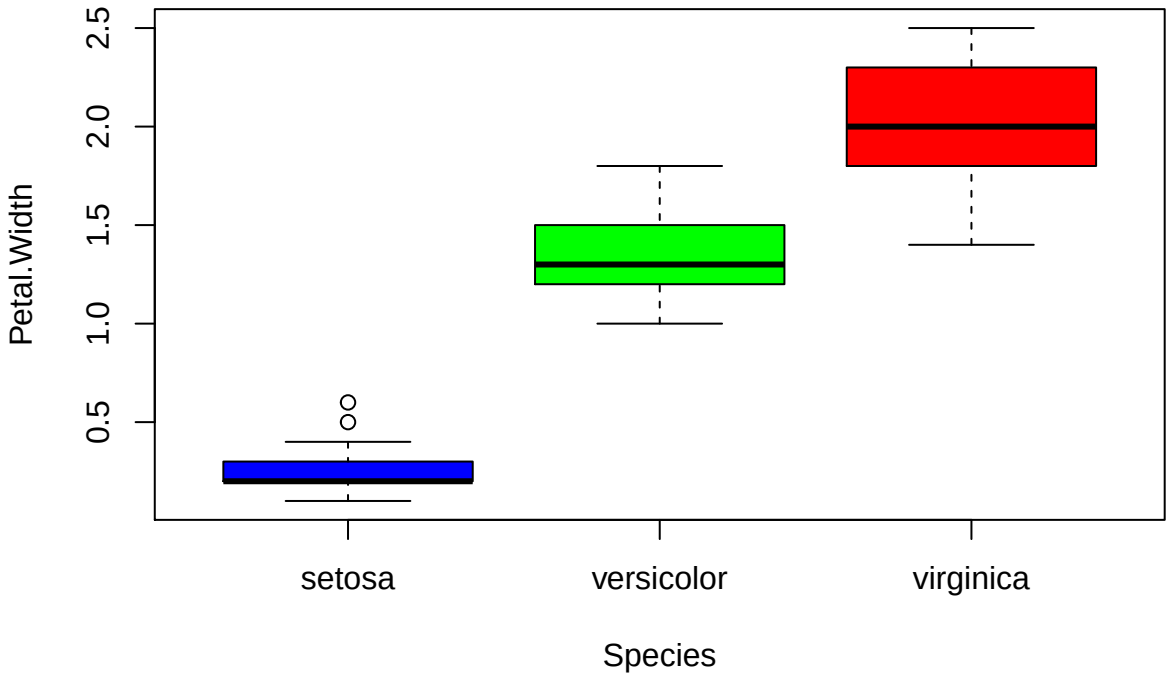
Boxplot with Jitter Point

```
boxplot(
  x_val,
  # plot horizontally?
  horizontal = TRUE,
  # add title
  main = "Normal Distribution"
)

stripchart(
  x_val,
  # use jitter method
  method = "jitter",
  # point type
  pch = 19,
  # add jitter to base plot?
  add = TRUE,
  # add transparent color to plot
  col = rgb(red = 0, green = 0, blue = 0, alpha = 0.2)
)
```



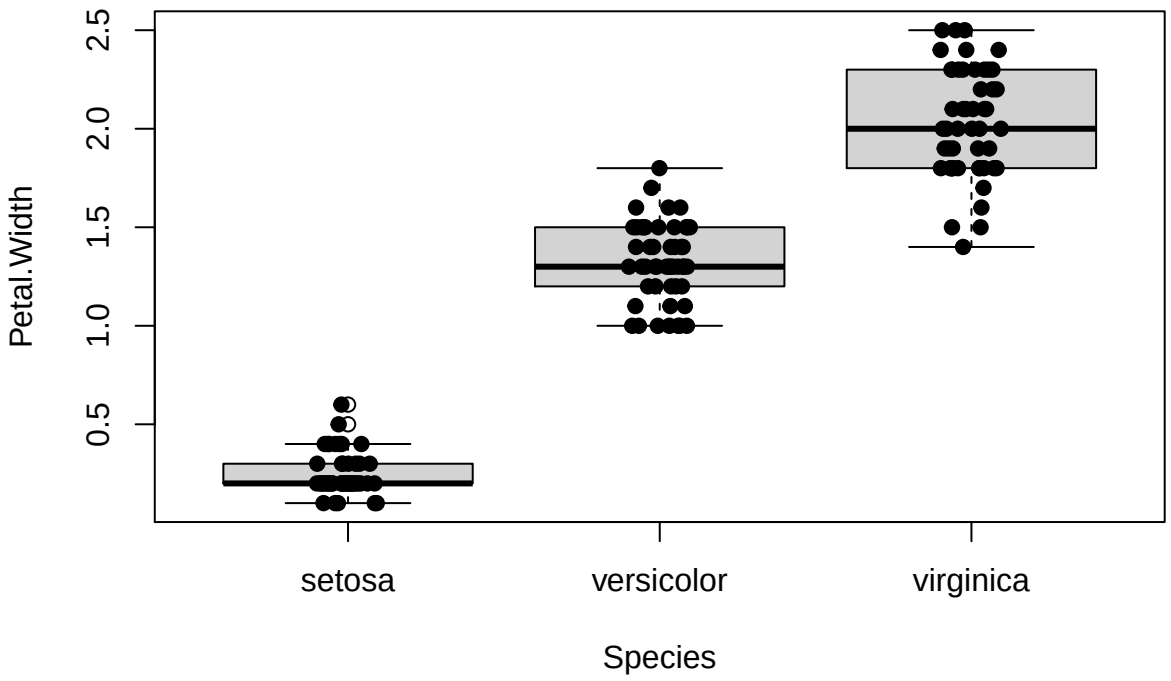
Basic Grouped Boxplot

```
boxplot(
  #formula: numeric ~ category
  formula = Petal.Width ~ Species,
  # add data
  data = iris,
  # add color
  col = c("blue", "green", "red")
)
```



Basic Grouped Boxplot with Stripchart

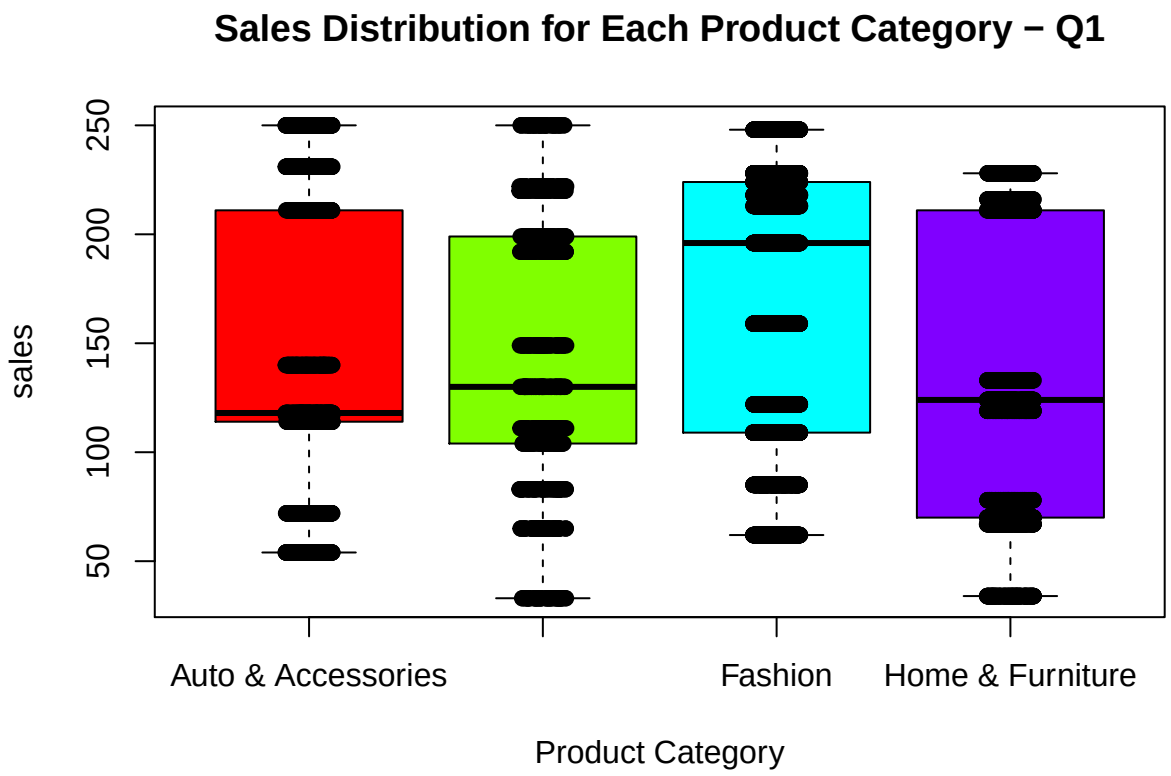
```
boxplot(
  #formula: numeric ~ category
  formula = Petal.Width ~ Species,
  # add data
  data = iris
)

stripchart(
  Petal.Width ~ Species,
  data = iris,
  method = "jitter",
  add = TRUE,
  vertical = TRUE,
  pch = 19
)
```



E-commerce Data

Q1 Each Product Category Sales Distribution

```
# from raw_data
raw_data %>%
  # filter order date to Q1 data
  filter(order_date >= "2015-01-01", order_date <= "2015-03-30") %>%
  # change product_category data type to factor
  mutate(product_category = as.factor(product_category)) -> q1_data

# create boxplot
boxplot(
  # set formula
  sales ~ product_category,
  # add data
  data = q1_data,
  # add color
  col = rainbow(4),
  # change x label
  xlab = "Product Category",
  # add title
  main = "Sales Distribution for Each Product Category - Q1"
)

stripchart(
  sales ~ product_category,
  data = q1_data,
  method = "jitter",
  add = TRUE,
  vertical = TRUE,
  pch = 19
)
```


Sales Distribution for Each Product Category − Q1
sales
50
100
150
200
250
Auto & Accessories
Fashion
Home & Furniture
Product Category

Violin Plot

call library

```
library(vioplot)
```

```
## Loading required package: sm
```

```
## Package 'sm', version 2.2-5.6: type help(sm) for summary information
```

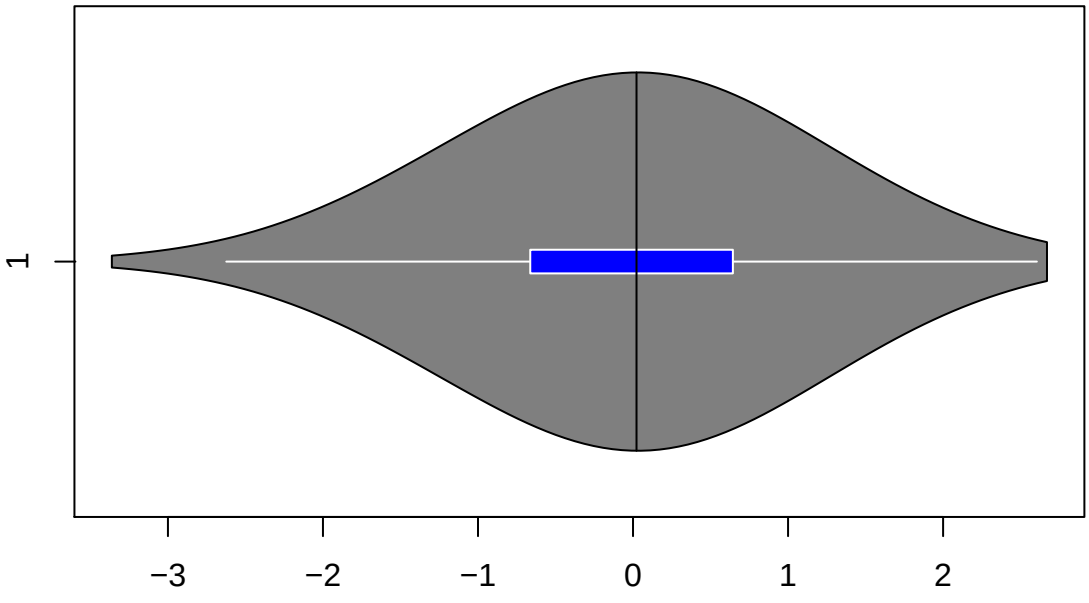
```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

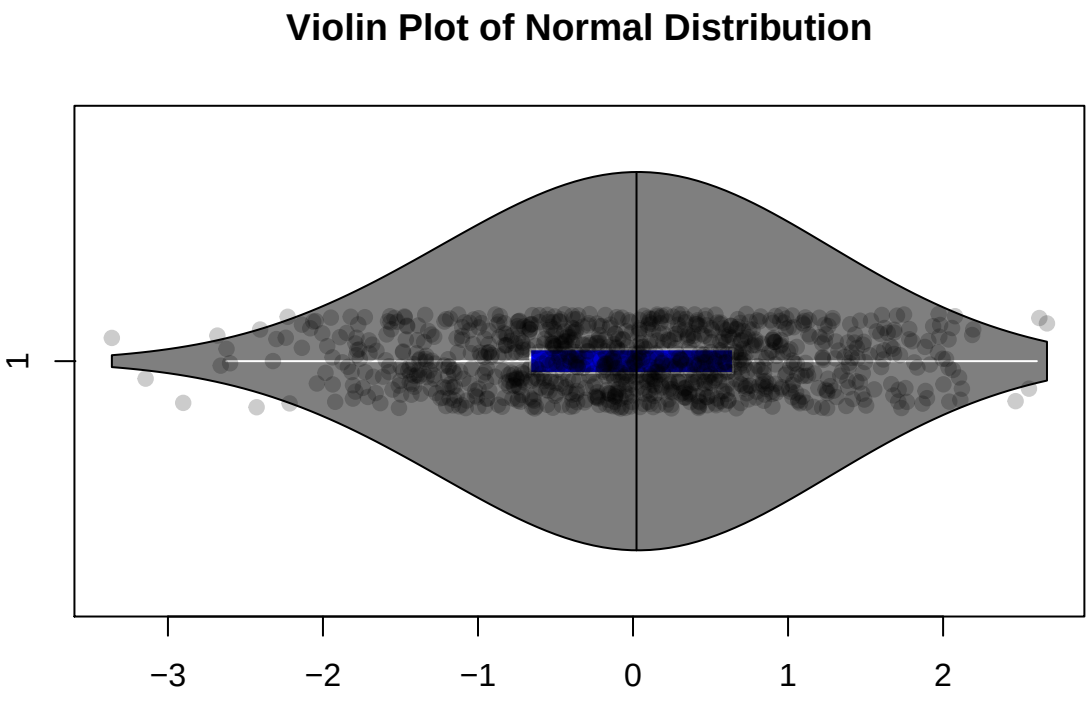
Create Violin Plot

```
vioplot(
  x_val,
  # plot horizontally?
  horizontal = T,
  # center plot type
  plotCentre = "line",
  # rectangle color
  rectCol = "blue",
  # line/whisker color
  lineCol = "white"
)
```



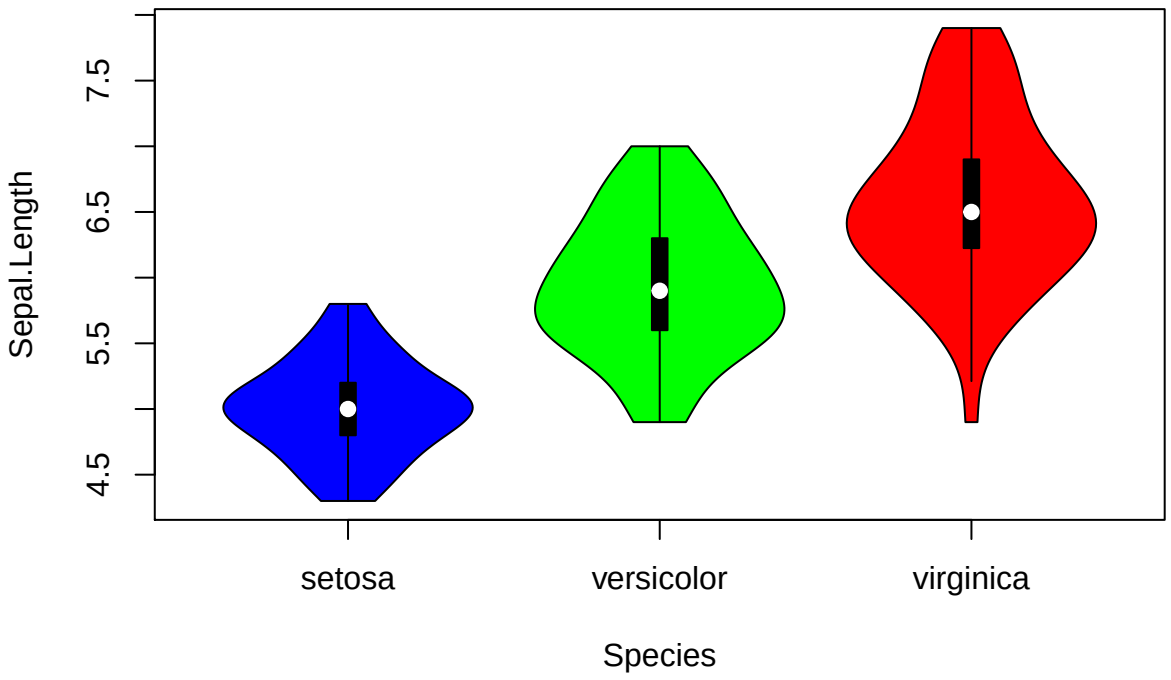
Violin Plot with Jitter

```
vioplot(
  x_val,
  # plot horizontally?
  horizontal = T,
  # center plot type
  plotCentre = "line",
  # rectangle color
  rectCol = "blue",
  # line/whisker color
  lineCol = "white",
  # Add Title
  main = "Violin Plot of Normal Distribution"
)

stripchart(
  x_val,
  # use jitter method
  method = "jitter",
  # point type
  pch = 19,
  # add jitter to base plot?
  add = TRUE,
  # add transparent color to plot
  col = rgb(red = 0, green = 0, blue = 0, alpha = 0.2)
)
```



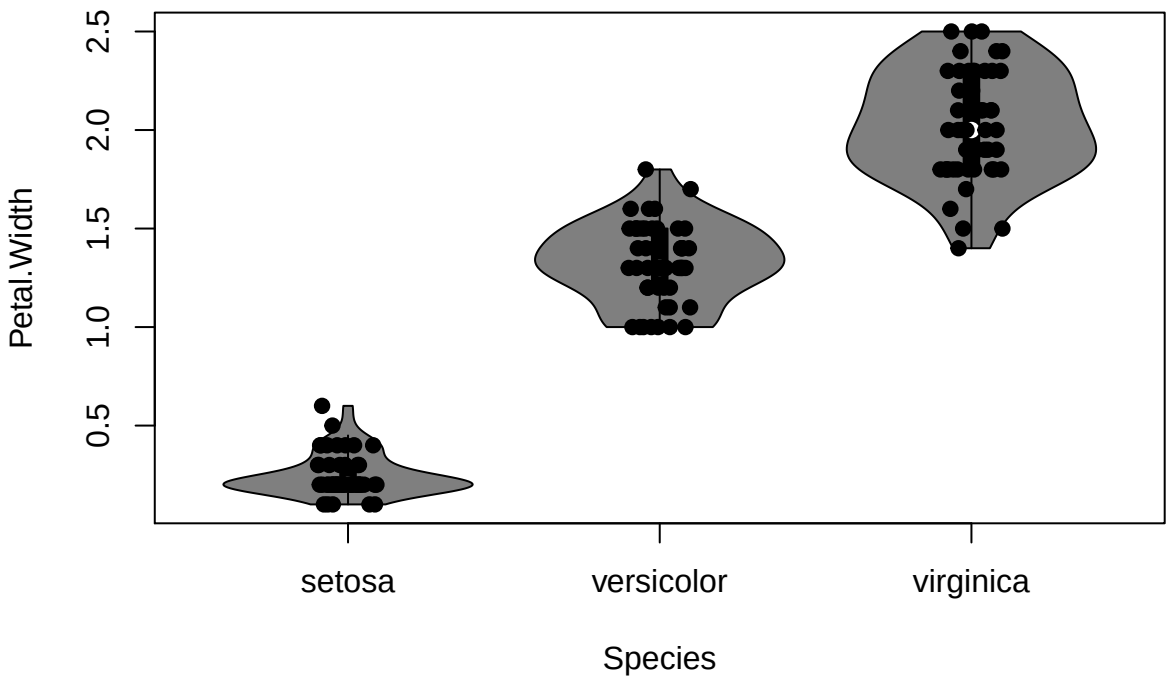
Basic Grouped Violin Plot

```
vioplot(
  # add formula: numeric ~ category
  formula = Sepal.Length ~ Species,
  # add data/variable
  data = iris,
  # add color to violin
  col = c("blue", "green", "red")
)
```



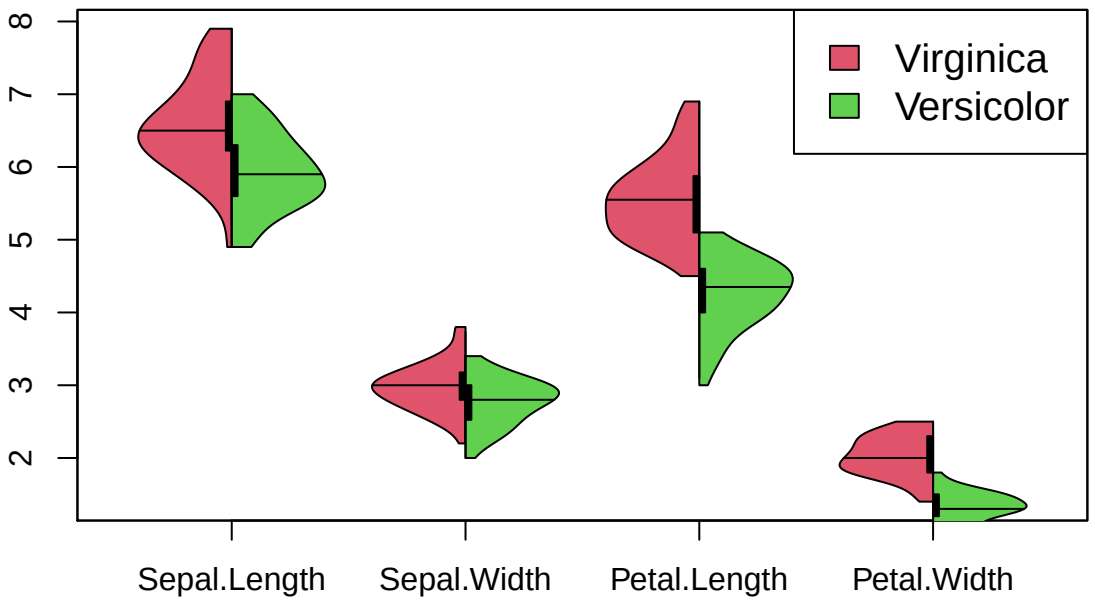
Grouped Boxplot with Jitter

```
vioplot(
  #formula: numeric ~ category
  formula = Petal.Width ~ Species,
  # add data
  data = iris
)

stripchart(
  # define formula
  Petal.Width ~ Species,
  # add data
  data = iris,
  # add method
  method = "jitter",
  # add as additional plot?
  add = TRUE,
  # plot vertically?
  vertical = TRUE,
  # point type
  pch = 19
)
```



Grouped-Split Violin Plot

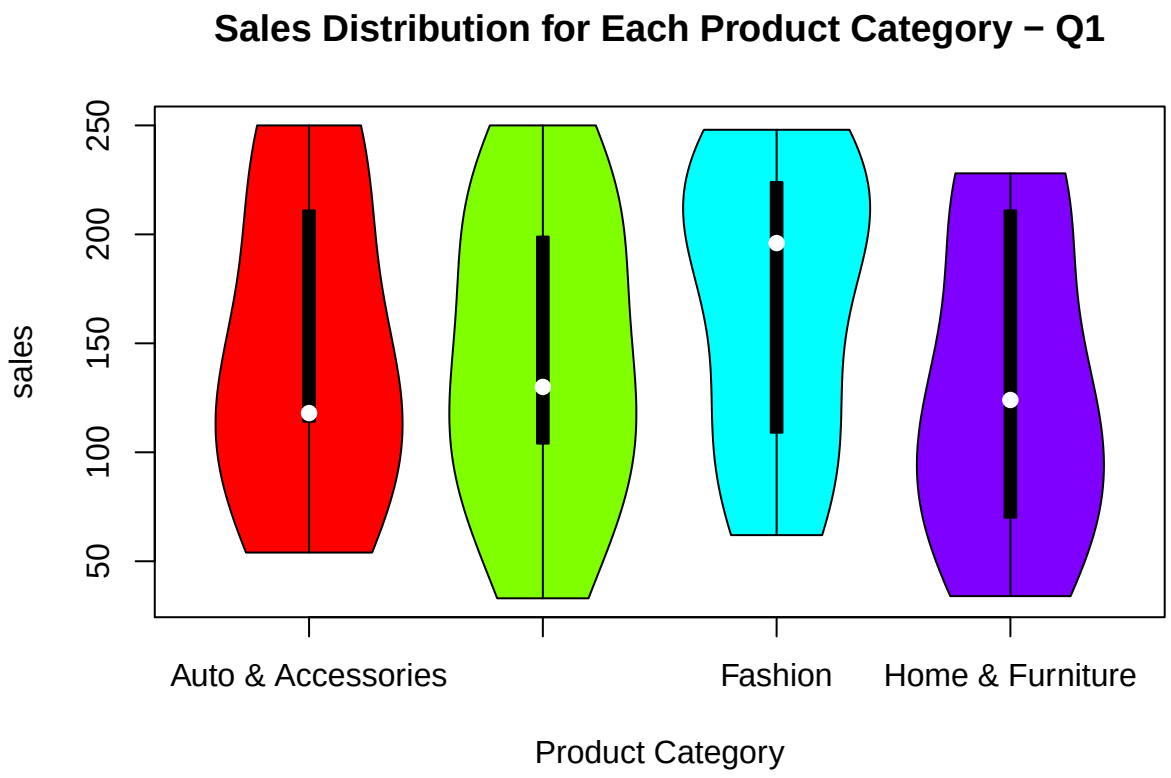
```
# add virginica species data
virginica <- iris[iris$Species == 'virginica', 1:4]
# add versicolor species data
versicolor <- iris[iris$Species == 'versicolor', 1:4]

# create violinplot for virginica as left side of violin plot
vioplot(virginica, side = "left", plotCentre = "line", col = 2)
# create violinplot for versicolor as right side of violin plot
vioplot(versicolor, side = "right", plotCentre = "line", col = 3, add = TRUE)

# add legend
legend("topright", legend = c("Virginica", "Versicolor"), fill = c(2, 3), cex = 1.25)
```



E-commerce Data

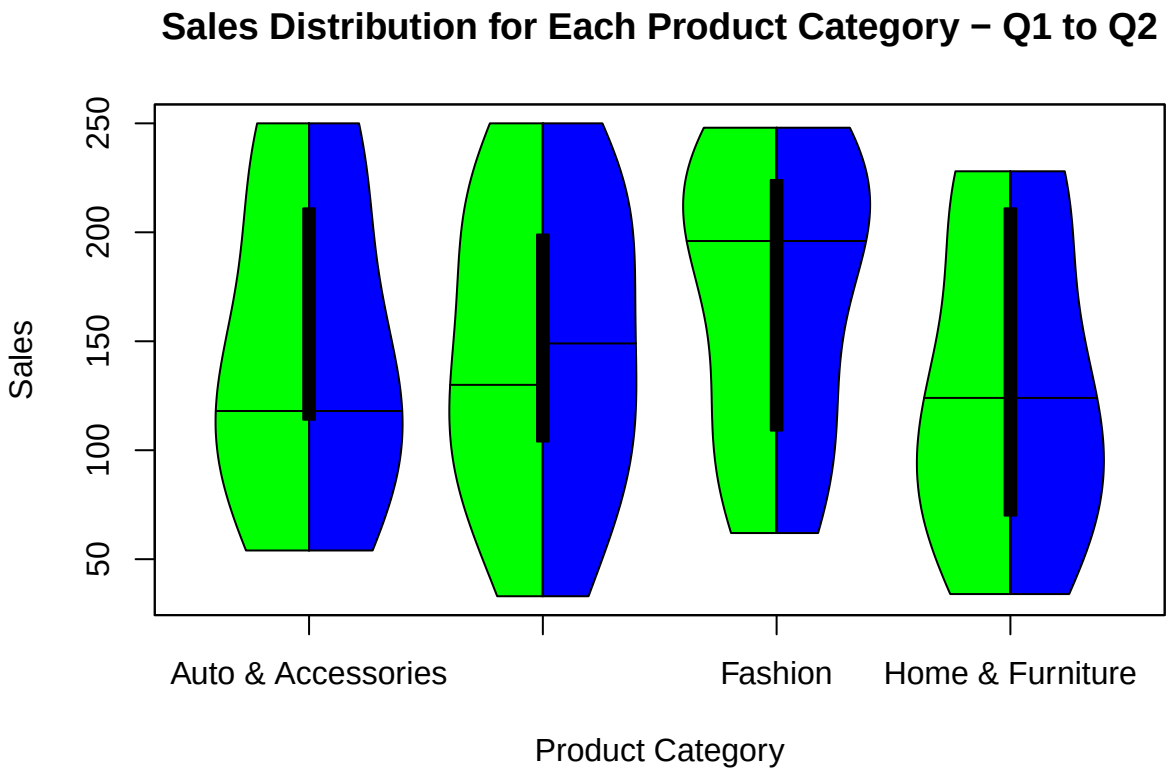
Re-plot Sales Distribution for Each Category on Q1

```
# create violinplot
vioplot(
  # set formula
  sales ~ product_category,
  # add data
  data = q1_data,
  # add color
  col = rainbow(4),
  # change x label
  xlab = "Product Category",
  # add title
  main = "Sales Distribution for Each Product Category - Q1"
)
```



Compare Sales Distribution on Q1 and Q2 for Each Category

```
# from raw_data
raw_data %>%
  # filter order date to Q1 data
  filter(order_date >= "2015-04-01", order_date <= "2015-06-30") %>%
  # change product_category data type to factor
  mutate(product_category = as.factor(product_category)) -> q2_data


vioplot(
  # set formula
  sales ~ product_category,
  # add data
  data = q1_data,
  # add color
  col = "green",
  # change x label
  xlab = "Product Category",
  # change y label
  ylab = "Sales",
  # add title
  main = "Sales Distribution for Each Product Category - Q1 to Q2",
  # which side?
  side = "left",
  # line as plot centre
  plotCentre = "line"
)

vioplot(
  # set formula
  sales ~ product_category,
  # add data
  data = q2_data,
  # add color
  col = "blue",
  # which side?
  side = "right",
  # line as plot centre
  plotCentre = "line",
  # add as additional plot?
  add = TRUE
)
```


Sales Distribution for Each Product Category − Q1 to Q2
Sales
50
100
150
200
250
Auto & Accessories
Fashion
Home & Furniture
Product Category

Barplot and Dot Plot

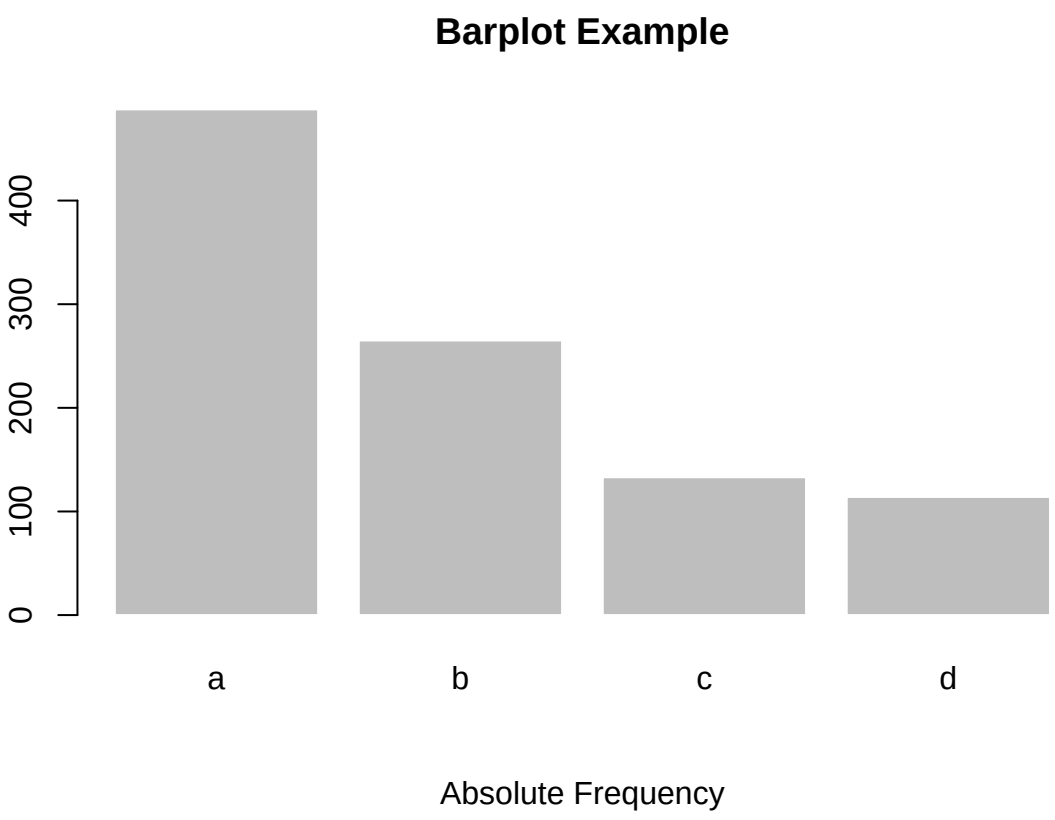
Basic Barplot (using barplot function)

Initialie Data

```
# create random categorical data
set.seed(1000)
categorical_data <- sample(c("a", "b", "c", "d"), size = 1000, replace = T, prob = c(0.4, 0.2, 0.1, 0.1

# turn to factor data type
categorical_data <- factor(categorical_data)

# create named vector using table function
named_vector <- table(categorical_data)
```

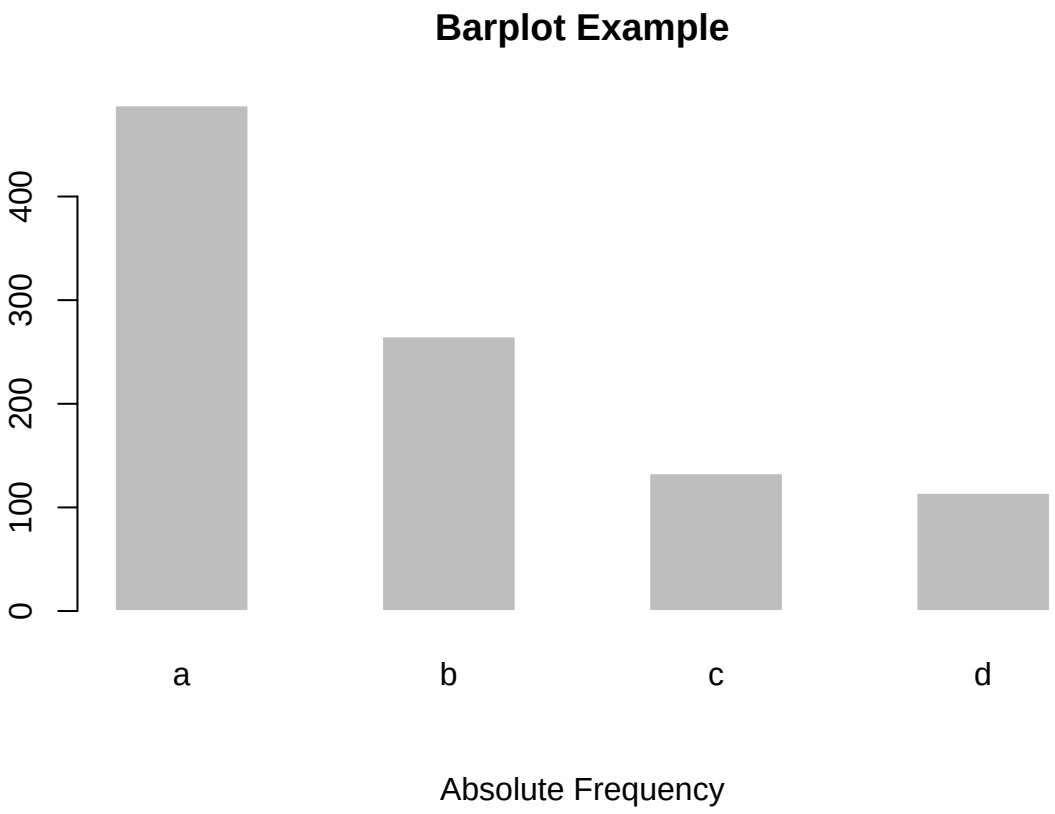
Basic Barplot

```
barplot(
  # add named vector
  height = named_vector,
  # add border color
  border = "white",
  # add title
  main = "Barplot Example",
  ## add subtitle
  sub = "Absolute Frequency"
)
```



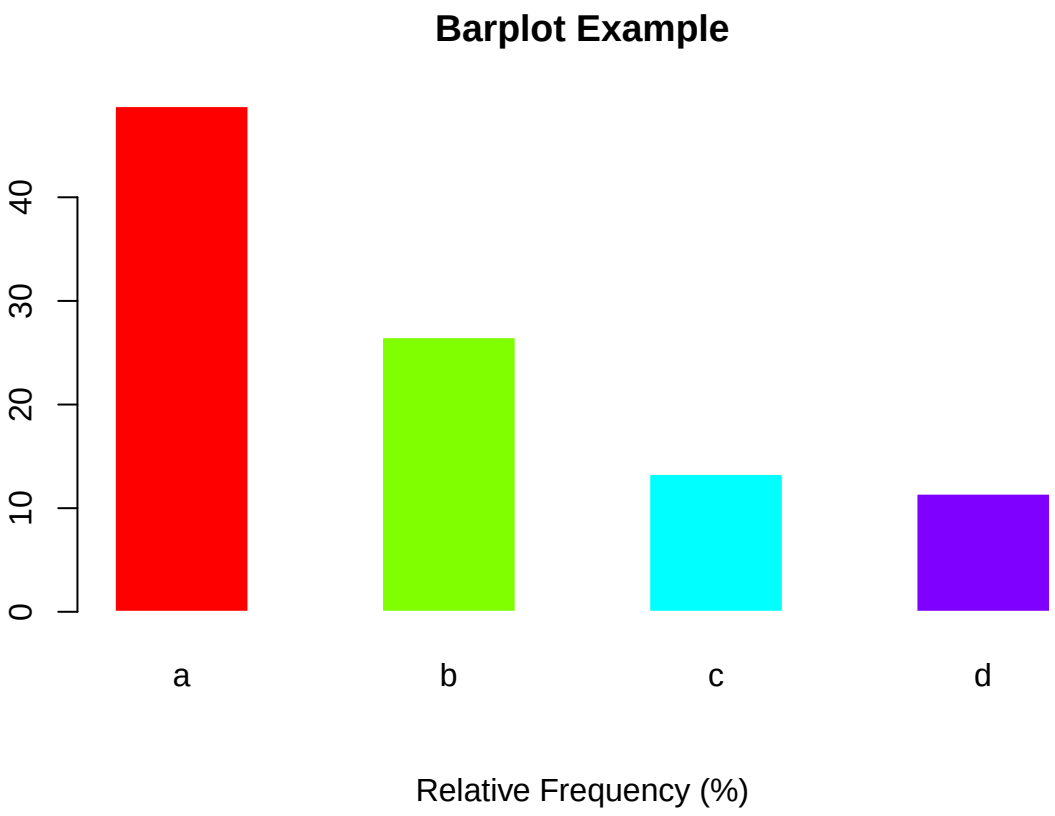
Basic Barplot with Space

```
barplot(
  # add named vector
  height = named_vector,
  # add border color
  border = "white",
  # add title
  main = "Barplot Example",
  ## add subtitle
  sub = "Absolute Frequency",
  # add space between bar
  space = 1
)
```



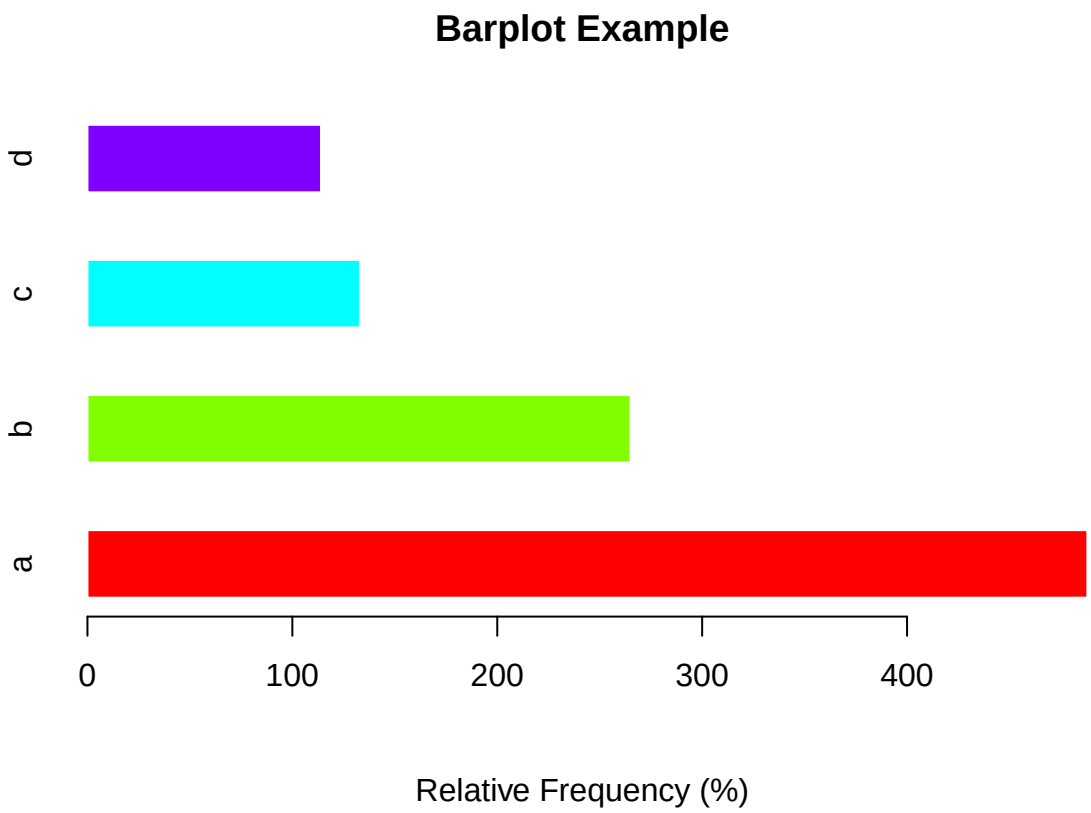
Relative Frequency

```
barplot(
  # turn named vector to proportion table
  height = prop.table(named_vector) * 100,
  # add border color
  border = "white",
  # add title
  main = "Barplot Example",
  ## add subtitle
  sub = "Relative Frequency (%)",
  # add space between bar
  space = 1,
  # add color automatically
  col = rainbow(4)
)
```



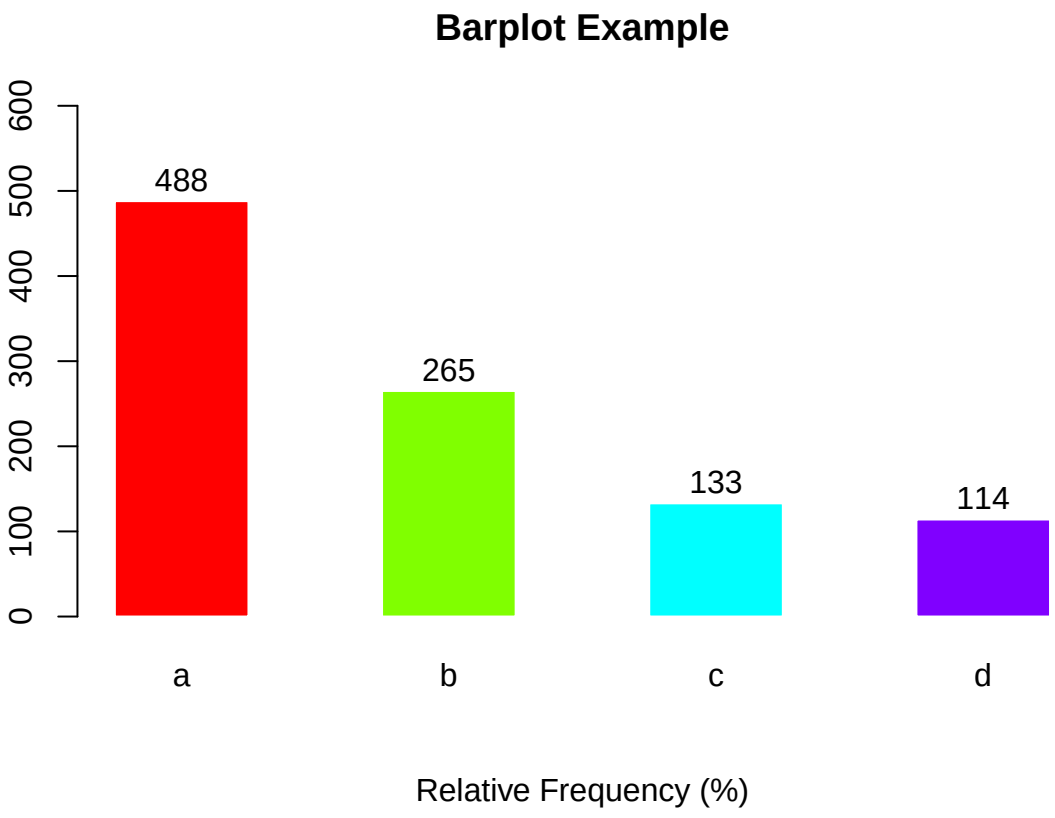
Barplot Using Plot Function

```
plot(
  # add factorized vector
  categorical_data,
  # add border color
  border = "white",
  # add title
  main = "Barplot Example",
  ## add subtitle
  sub = "Relative Frequency (%)",
  # add space between bar
  space = 1,
  # add color automatically
  col = rainbow(4),
  # draw plot horizontally
  horiz = TRUE
)
```



Adding Text to Barplot

```
bar_plot <- barplot(
  # add factorized vector
  named_vector,
  # add border color
  border = "white",
  # add title
  main = "Barplot Example",
  ## add subtitle
  sub = "Relative Frequency (%)",
  # add space between bar
  space = 1,
  # add color automatically
  col = rainbow(4),
  # change y limit
  ylim = c(0,600),
)

## add text label to barplot
text(
  bar_plot,
  # adjust position
  named_vector + 25,
  # add labels
  labels = named_vector
)
```



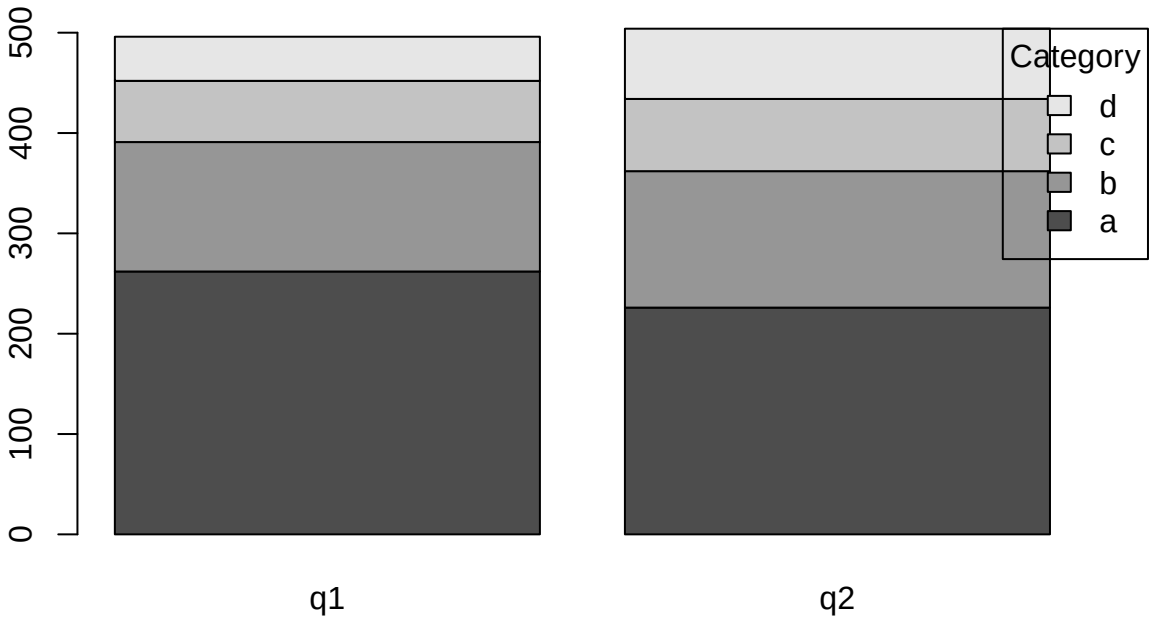
Grouped Barplot

Initialize Value

```
set.seed(1000)
categorical_data_1 <- sample(c("a", "b", "c", "d"), size = 1000, replace = T, prob = c(0.4, 0.2, 0.1, 0
categorical_data_2 <- sample(c("q1", "q2"), size = 1000, replace = T)

# turn to factor data type
categorical_data_1 <- factor(categorical_data_1)
categorical_data_2 <- factor(categorical_data_2)

# create named vector using table function
named_vector_1 <- table(categorical_data_1, categorical_data_2)
named_vector_2 <- table(categorical_data_2, categorical_data_1)
```

Basic Stacked Barplot with Legend

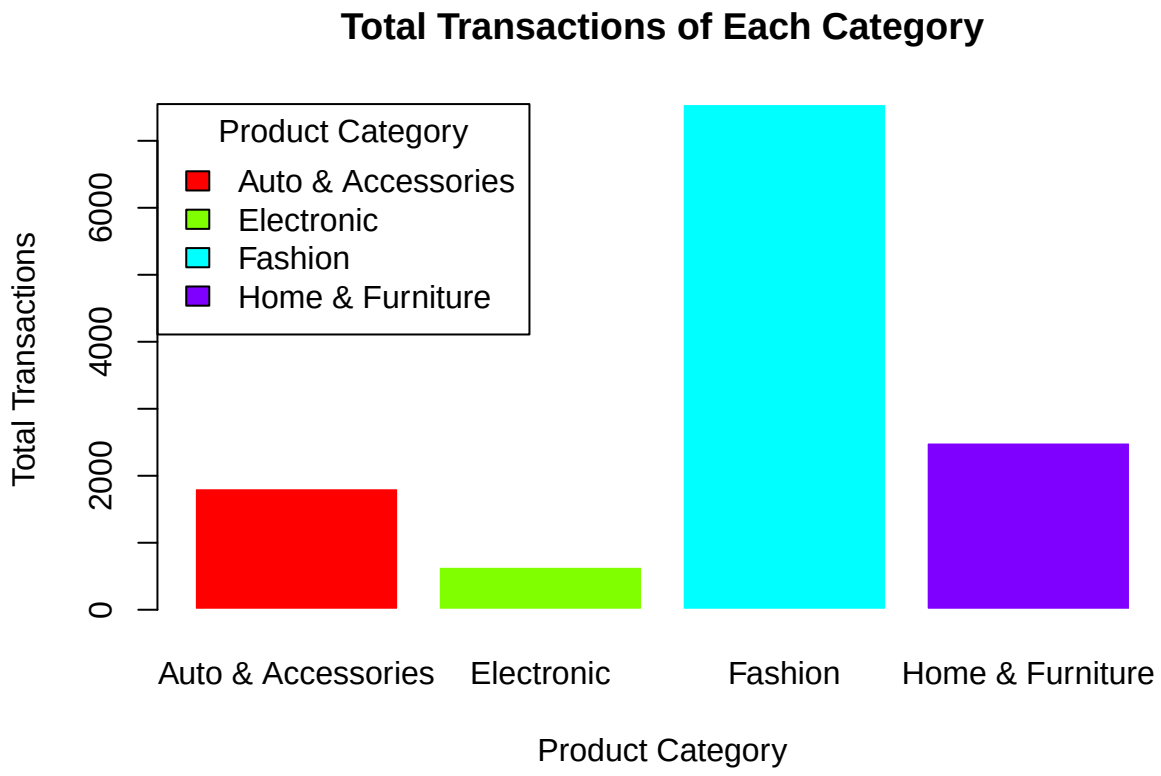
```
barplot(
  named_vector_1,
  legend.text = rownames(named_vector_1),
  args.legend = list(
    title = "Category",
    x = "topright",
    inset = c(-0.06, 0)
  )
)
```



E-commerce Data

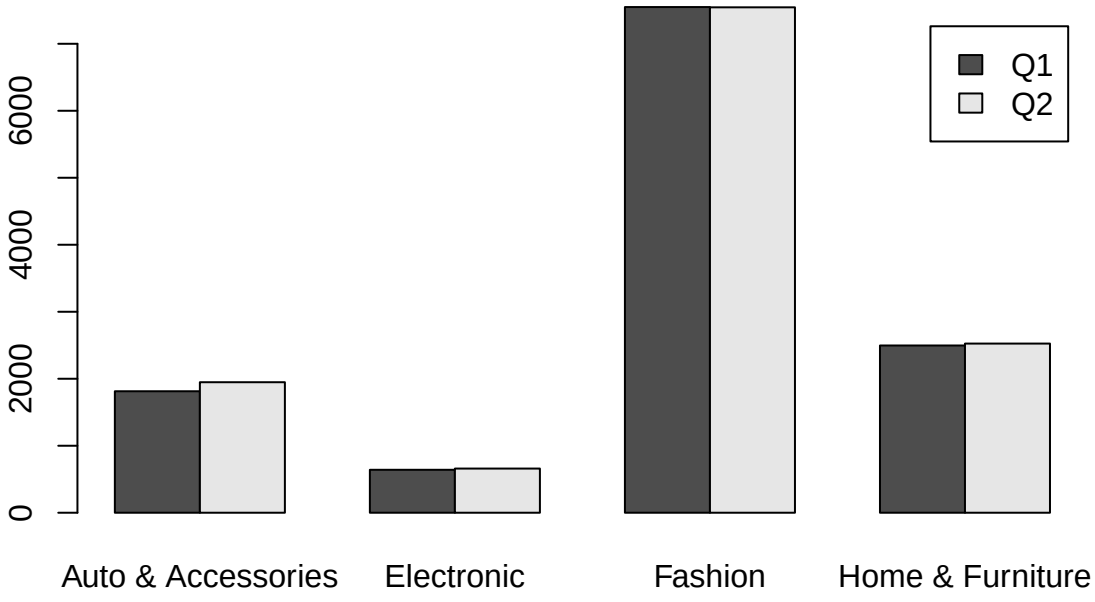
Barplot of Total Transaction of Each Product Category on Q1

```
barplot(
  # add named vector
  table(q1_data$product_category),
  # add title
  main = "Total Transactions of Each Category",
  # add color
  col = rainbow(4),
  # change border color
  border = "white",
  # add x label
  xlab = "Product Category",
  # add y label
  ylab = "Total Transactions",
  # add legend
  legend.text = rownames(table(q1_data$product_category)),
  # config legend position
  args.legend = list(
    title = "Product Category",
    x = "topleft"
  )
)
```



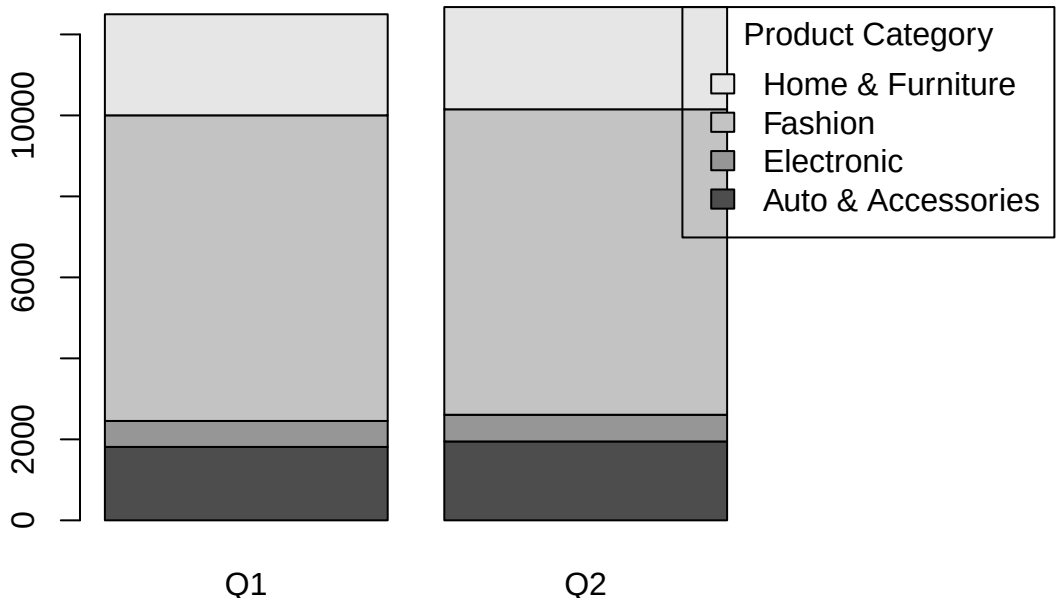
Grouped Barplot of Total Transactions of Fashion Product Category - Q1 vs Q2

```
# from q1_data
q1_data %>%
  # add q2_data by rows
  bind_rows(q2_data) %>%
  # add quarter column
  mutate(
    quarter = ifelse(order_date <= "2015-03-30", "Q1", "Q2")
  ) %>%
  # select only quarter and product category column
  select(quarter, product_category) %>%
  # create named_vector by cross tabulation
  table(.) -> named_vector

# create barplot
barplot(
  named_vector,
  # group by side?
  beside = TRUE,
  # add legend
  legend.text = rownames(named_vector)
)
```



Stacked Barplot of Total Transactions of Fashion Product Category - Q1 vs Q2

```
q1_data %>%
  bind_rows(q2_data) %>%
  mutate(
    quarter = ifelse(order_date <= "2015-03-30", "Q1", "Q2")
  ) %>%
  select(product_category, quarter) %>%
  table(.) -> named_vector

# set margin
par(mar = c(5,5,4,10))

barplot(
  named_vector,
  # beside?
  beside = FALSE,
  # add legend
  legend.text = rownames(named_vector),
  # config legend position
  args.legend = list(
    title = "Product Category",
    x = "topright",
    inset = c(-0.45, 0)
  )
)
```



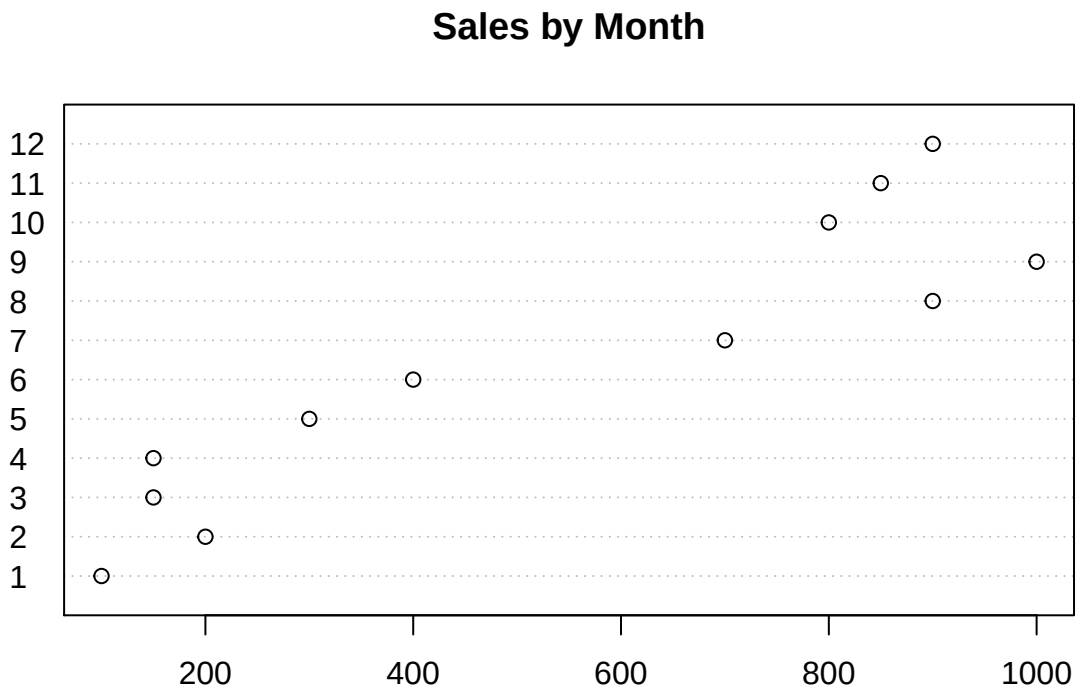
Dot Plot

Initialize Data

```
month <- 1:12
quarter <- c(1,1,1,2,2,2,3,3,3,4,4,4)
frequency <- c(100, 200, 150, 150, 300, 400, 700, 900, 1000, 800, 850, 900)
```

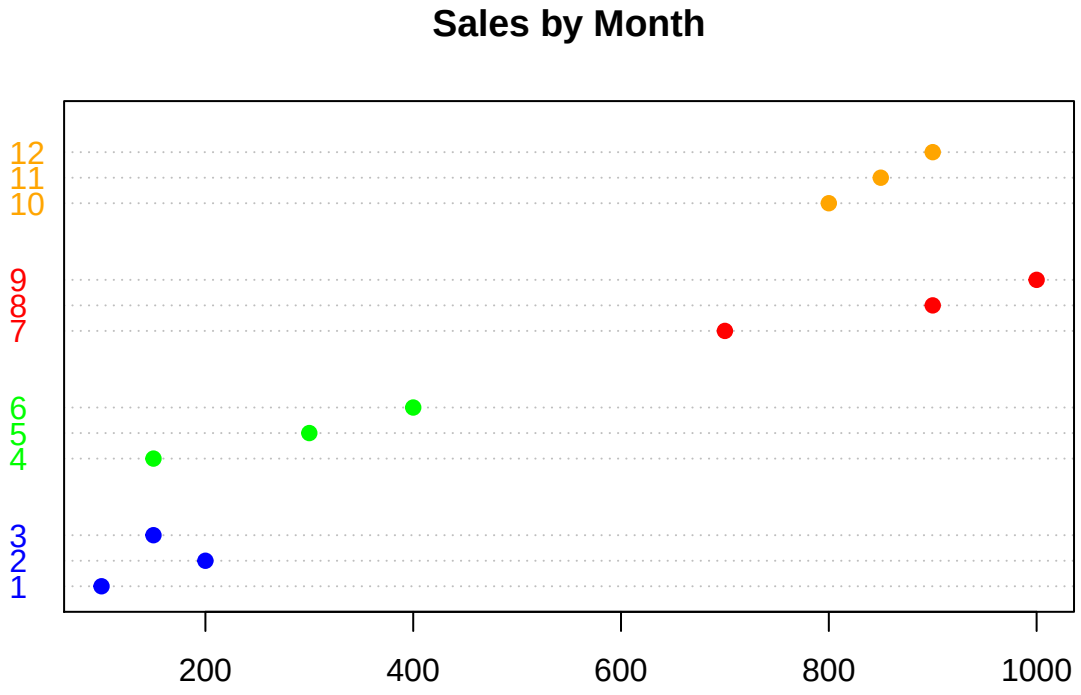
Basic Dot Plot

```
dotchart(
  # add x
  x = frequency,
  # add label
  labels = month,
  # add title
  main = "Sales by Month"
)
```



Grouped Dotplot

```
dotchart(
  # add x
  x = frequency,
  # add label
  labels = factor(month),
  # group data
  groups = rev(quarter),
  # add title
  main = "Sales by Month",
  # add color
  color = rep(c("blue", "green", "red", "orange"), c(3,3,3,3)),
  # change point type
  pch = 19
)
```



Dumbbell Plot

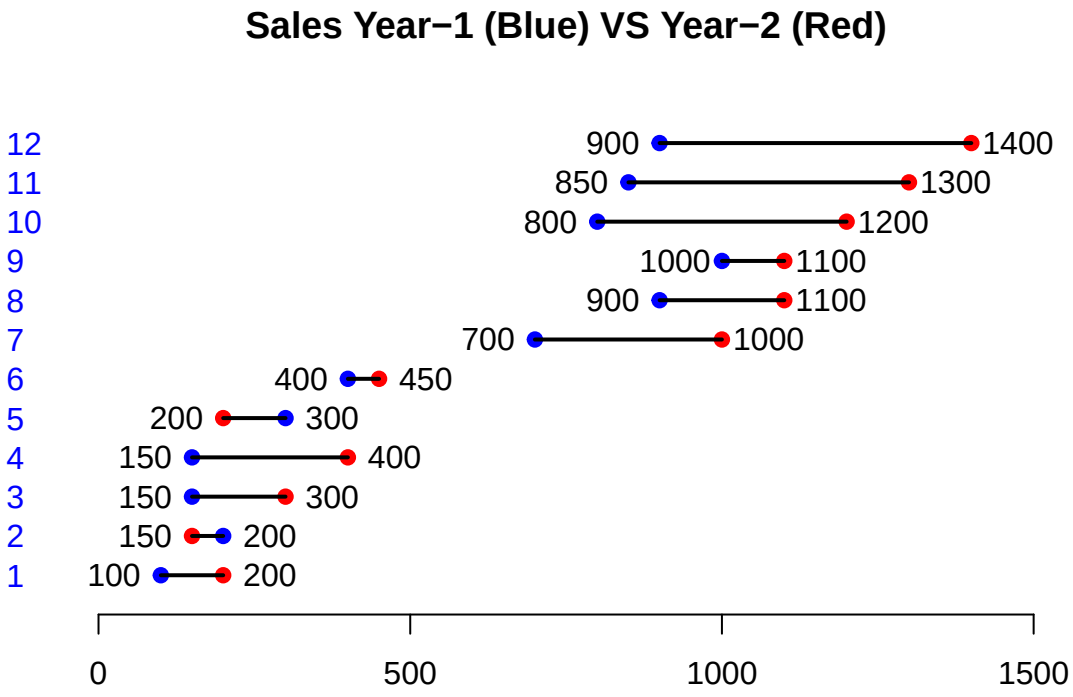
Initialize Data

```
month <- 1:12
frequency_y1 <- c(100, 200, 150, 150, 300, 400, 700, 900, 1000, 800, 850, 900)
frequency_y2 <- c(200, 150, 300, 400, 200, 450, 1000, 1100, 1100, 1200, 1300, 1400)
```

Create Dumbbell Plot

```
dotchart(
  x = frequency_y1,
  labels = month,
  # change point type
  pch = 19,
  # add color
  color = "blue",
  # add x limit
  xlim = c(0, 1500),
  # add line color
  lcolor = "white",
  # add title
  main = "Sales Year-1 (Blue) VS Year-2 (Red)",
  # add frame?
  frame.plot = F
)

# add new points
points(
  frequency_y2,
  month,
  # change point type
  pch = 19,
  # add color
  col = "red"
)

# add line segment and text for each point every month
invisible(
  sapply(1:length(month), function(i) {

  # add segment
  segments(
    # add min value
    x0 = min(frequency_y1[i], frequency_y2[i]), y0 = i,
    # add max value
    x1 = max(frequency_y1[i], frequency_y2[i]), y1 = i,
    # config line width
    lwd = 2
  )

  # add text for min values
  text(
    min(frequency_y1[i], frequency_y2[i]) - 75, i,
```

```
    # add label
    labels = min(frequency_y1[i], frequency_y2[i])
  )
  # add text for max values
  text(
    max(frequency_y1[i], frequency_y2[i]) + 75, i,
    # add label
    labels = max(frequency_y1[i], frequency_y2[i])
  )
  })
)
```



E-Commerce Data

Total Sales Q1 vs Q2

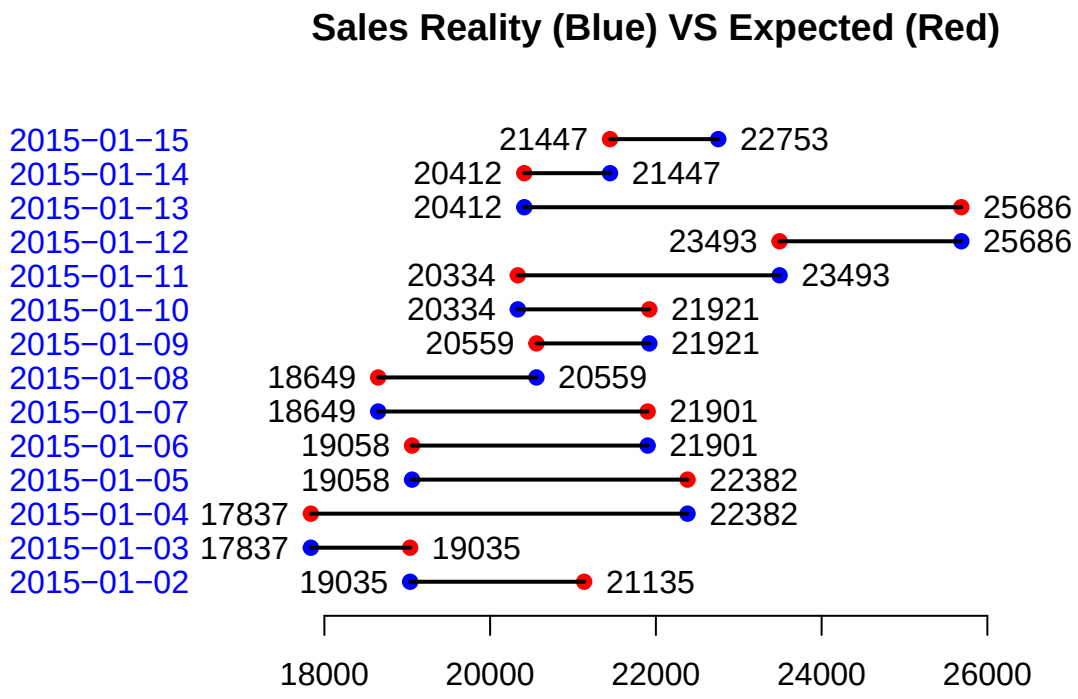
Initialize Data

```
raw_data %>%
  filter(order_date >= "2015-01-01", order_date <= "2015-01-15") %>%
  group_by(order_date) %>%
  summarise(total_sales = sum(sales)) %>%
  ungroup() %>%
  mutate(expected_sales = lag(total_sales)) %>%
  arrange(order_date) %>%
  na.omit() -> sales_data
```

Build Dumbell Plot

```
dotchart(
  x = sales_data$total_sales,
  labels = sales_data$order_date,
  # change point type
  pch = 19,
  # add color
  color = "blue",
  # add line color
  lcolor = "white",
  # add title
  main = "Sales Reality (Blue) VS Expected (Red)",
  # add frame?
  frame.plot = F,
  # add x limit
  xlim = c(17000, 27000)
)

# add new points
points(
  sales_data$expected_sales,
  1:nrow(sales_data),
  # change point type
  pch = 19,
  # add color
  col = "red"
)

# add line segment and text for each point every month
invisible(
  sapply(1:nrow(sales_data), function(i) {

  # add segment
  segments(
    # add min value
    x0 = min(sales_data$total_sales[i], sales_data$expected_sales[i]), y0 = i,
    # add max value
    x1 = max(sales_data$total_sales[i], sales_data$expected_sales[i]), y1 = i,
    # config line width
```

```
    lwd = 2
  )

  # add text for min values
  text(
    min(sales_data$total_sales[i], sales_data$expected_sales[i]) - 800, i,
    # add label
    labels = min(sales_data$total_sales[i], sales_data$expected_sales[i])
  )
  # add text for max values
  text(
    max(sales_data$total_sales[i], sales_data$expected_sales[i]) + 800, i,
    # add label
    labels = max(sales_data$total_sales[i], sales_data$expected_sales[i])
  )
  })
)
```



Reference

- R Graphics Cookbook, 2nd edition by Winston Chang
- Data Visualization - A practical introduction by Kieran Healy
- R Graph Gallery
- R for Data Science
- From Data to Viz
- Exploratory Data Analysis with R by Roger D. Peng
- Intro to R by Alex Douglas, Deon Roos, Francesca Mancini, Ana Couto & David Lusseau