```r
library(ggplot2)
```

Import Data

```r
# Import the dataset
nflx_df <- read.csv("C:/Users/benog/OneDrive/Documents/Grad School/USD/ADS 503 Predictive Modeling/Datas

# Add headers to the dataframe
#names(glass_df) <- c("Id_number", "RI", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe", "Type_of_glass"

head(nflx_df)
```

```
##         Date   Open   High    Low  Close Adj.Close   Volume
## 1 2018-02-05 262.00 267.90 250.03 254.26    254.26 11896100
## 2 2018-02-06 247.70 266.70 245.00 265.72    265.72 12595800
## 3 2018-02-07 266.58 272.45 264.33 264.56    264.56  8981500
## 4 2018-02-08 267.08 267.62 250.00 250.10    250.10  9306700
## 5 2018-02-09 253.85 255.80 236.11 249.47    249.47 16906900
## 6 2018-02-12 252.14 259.15 249.00 257.95    257.95  8534900
```

Find Null Values

```r
# Find number of null values for each attribute
missing_data <- colSums(is.na(nflx_df))
missing_data
```

```
##      Date      Open      High       Low     Close Adj.Close    Volume
##         0         0         0         0         0         0         0
```

Find Outliers

```r
# Use Tukey's method to find outliers
find_outliers <- function(x) {
  q <- quantile(x, probs=c(0.25, 0.75), na.rm=TRUE)
  iqr <- IQR(x, na.rm=TRUE)
  lower_bound <- q[1] - 1.5 * iqr
  upper_bound <- q[2] + 1.5 * iqr
  outliers <- x[x < lower_bound | x > upper_bound]
  return(outliers)
}

# Exclude the Date column
nflx_df_no_date <- nflx_df[, !names(nflx_df) %in% "Date"]

# Find the outliers for each predictor variable
outliers_list <- lapply(nflx_df_no_date, find_outliers)

# Display the outliers
names(outliers_list) <- names(nflx_df_no_date)
outliers_list
```

```
## $Open
## numeric(0)
##
## $High
## numeric(0)
##
## $Low
## numeric(0)
##
## $Close
## numeric(0)
##
## $Adj.Close
## numeric(0)
##
## $Volume
##  [1] 18986100 18525800 20369200 18972900 19145500 20307900 33866500 18222800
##  [9] 18389900 22490900 22960000 58410400 21746300 18260700 17427300 17183100
## [17] 20156400 32610900 18461000 19039300 19616000 21698800 23685700 20360300
## [25] 21397600 19330100 18620100 19500400 21181200 18871200 26621000 17941400
## [33] 18740200 18054100 31287100 17718000 23832800 38258900 23429900 21730000
## [41] 18200300 17939700 23177600 21084800 21605600 18399000 24499000 24991400
## [49] 20373700 17405700 32637500 22897400 58904300 32346000 24324700 20047500
## [57] 22542300
```

```r
# Use Tukey's method to find outliers and output to a dataframe
get_outliers_df <- function(x, dates) {
  q <- quantile(x, probs=c(0.25, 0.75), na.rm=TRUE)
  iqr <- IQR(x, na.rm=TRUE)
  lower_bound <- q[1] - 1.5 * iqr
  upper_bound <- q[2] + 1.5 * iqr
  outliers <- x[x < lower_bound | x > upper_bound]
  outlier_dates <- dates[x < lower_bound | x > upper_bound]
  return(data.frame(Value = outliers, Date = outlier_dates))
}

# Add the corresponding date to the outlier's value
outliers_list <- lapply(nflx_df_no_date, get_outliers_df, dates = nflx_df$Date)

# Combine outlier values and dates into a single dataframe
outliers_df <- do.call(rbind, outliers_list)
outliers_df
```

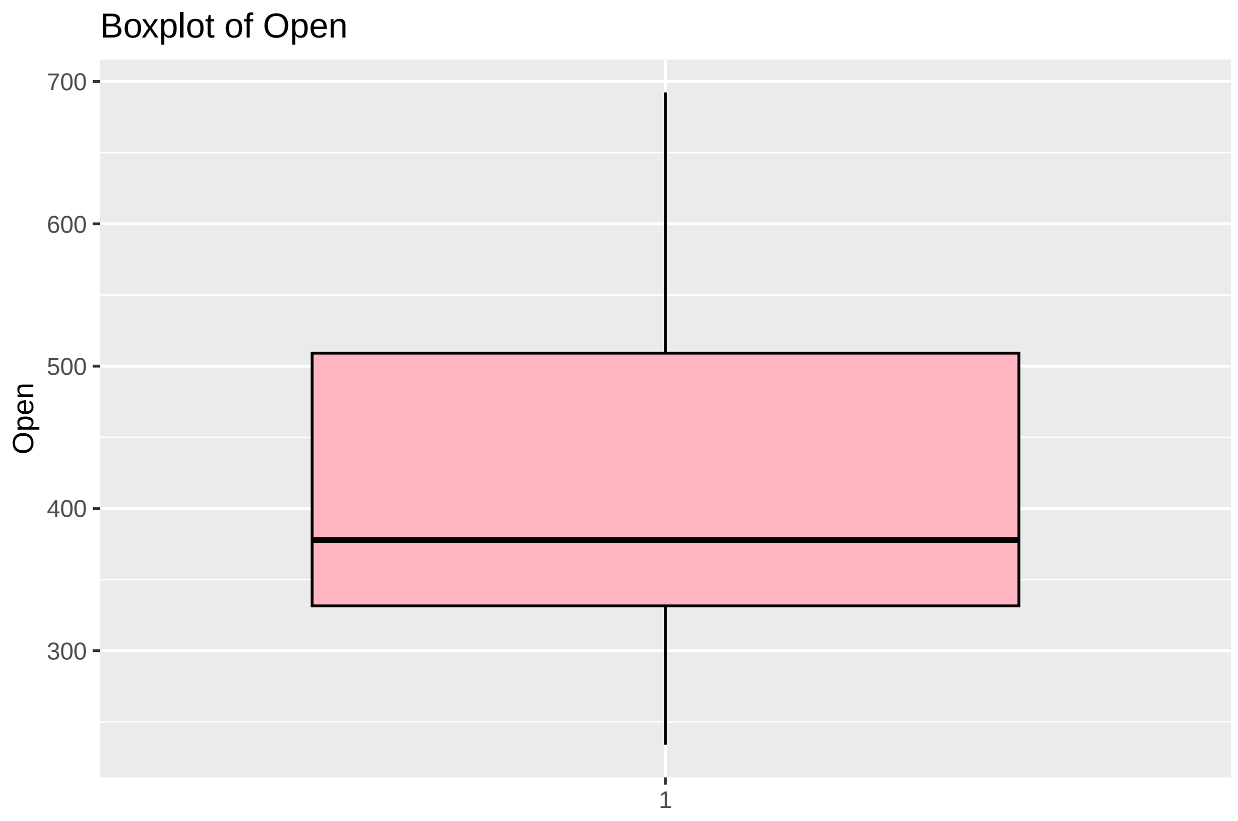
```
##                Value       Date
## Volume.1  18986100 2018-03-05
## Volume.2  18525800 2018-03-06
## Volume.3  20369200 2018-03-12
## Volume.4  18972900 2018-03-28
## Volume.5  19145500 2018-03-29
## Volume.6  20307900 2018-04-16
## Volume.7  33866500 2018-04-17
## Volume.8  18222800 2018-06-13
## Volume.9  18389900 2018-06-21
## Volume.10 22490900 2018-06-25
## Volume.11 22960000 2018-07-16
## Volume.12 58410400 2018-07-17
## Volume.13 21746300 2018-07-18
## Volume.14 18260700 2018-07-30
## Volume.15 17427300 2018-08-27
## Volume.16 17183100 2018-10-10
## Volume.17 20156400 2018-10-16
## Volume.18 32610900 2018-10-17
## Volume.19 18461000 2018-10-18
## Volume.20 19039300 2018-10-24
## Volume.21 19616000 2018-10-26
## Volume.22 21698800 2018-10-29
## Volume.23 23685700 2018-10-30
## Volume.24 20360300 2018-10-31
## Volume.25 21397600 2018-12-21
## Volume.26 19330100 2019-01-04
## Volume.27 18620100 2019-01-07
## Volume.28 19500400 2019-01-11
## Volume.29 21181200 2019-01-15
## Volume.30 18871200 2019-01-17
## Volume.31 26621000 2019-01-18
## Volume.32 17941400 2019-01-22
## Volume.33 18740200 2019-04-16
## Volume.34 18054100 2019-04-17
## Volume.35 31287100 2019-07-18
## Volume.36 17718000 2019-07-22
## Volume.37 23832800 2019-09-20
## Volume.38 38258900 2019-10-17
## Volume.39 23429900 2019-10-18
## Volume.40 21730000 2020-01-22
## Volume.41 18200300 2020-01-23
## Volume.42 17939700 2020-01-24
## Volume.43 23177600 2020-04-21
## Volume.44 21084800 2020-04-22
## Volume.45 21605600 2020-07-10
## Volume.46 18399000 2020-07-13
## Volume.47 24499000 2020-07-16
## Volume.48 24991400 2020-07-17
## Volume.49 20373700 2020-08-26
## Volume.50 17405700 2020-10-21
## Volume.51 32637500 2021-01-20
## Volume.52 22897400 2021-04-21
## Volume.53 58904300 2022-01-21
## Volume.54 32346000 2022-01-24
## Volume.55 24324700 2022-01-27
## Volume.56 20047500 2022-01-31
## Volume.57 22542300 2022-02-01
```

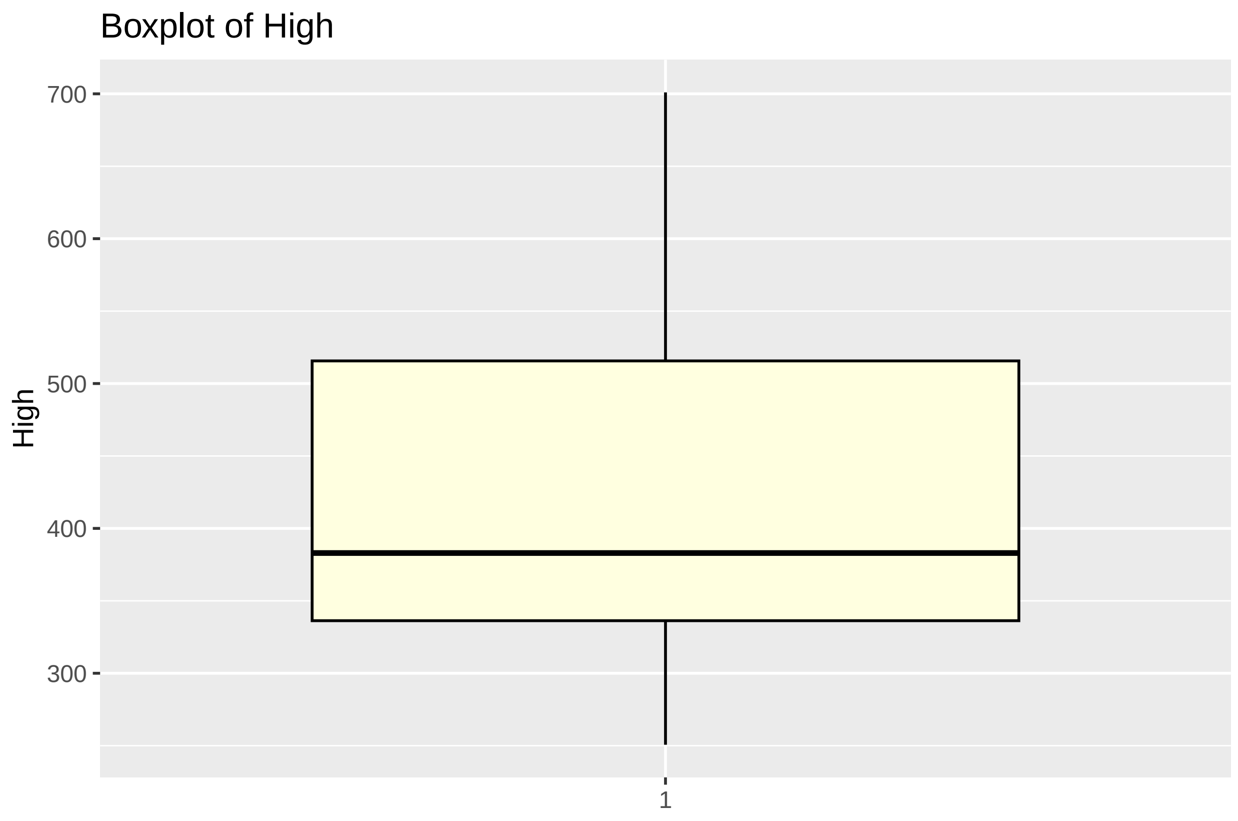
Create Box & Whisker Plots

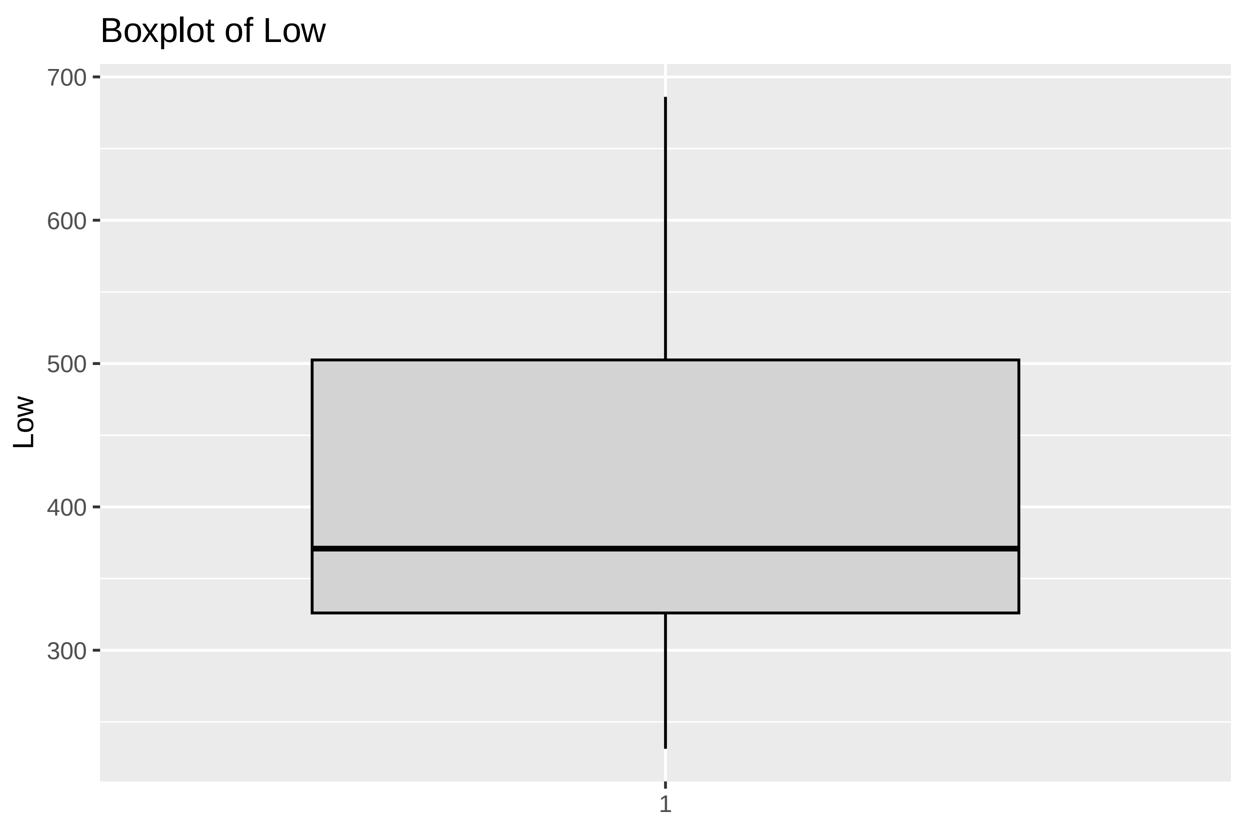
```r
# Generate a distinct color for each variable
set_colors <- function(index) {
  colors <- c("lightgreen", "lightcoral", "lightpink", "lightyellow", "lightgrey", "lightblue")
  colors[(index %% length(colors)) + 1]
}

# Create boxplots for each variable in nflx_df except Date
boxplots <- lapply(seq_along(names(nflx_df)), function(i) {
  var <- names(nflx_df)[i]
  if (var != "Date") {
    ggplot(nflx_df, aes_string(x = "factor(1)", y = var)) +
      geom_boxplot(fill = set_colors(i), color = "black") +
      labs(x = "", y = var) +
      ggtitle(paste("Boxplot of", var))
  }
})
```

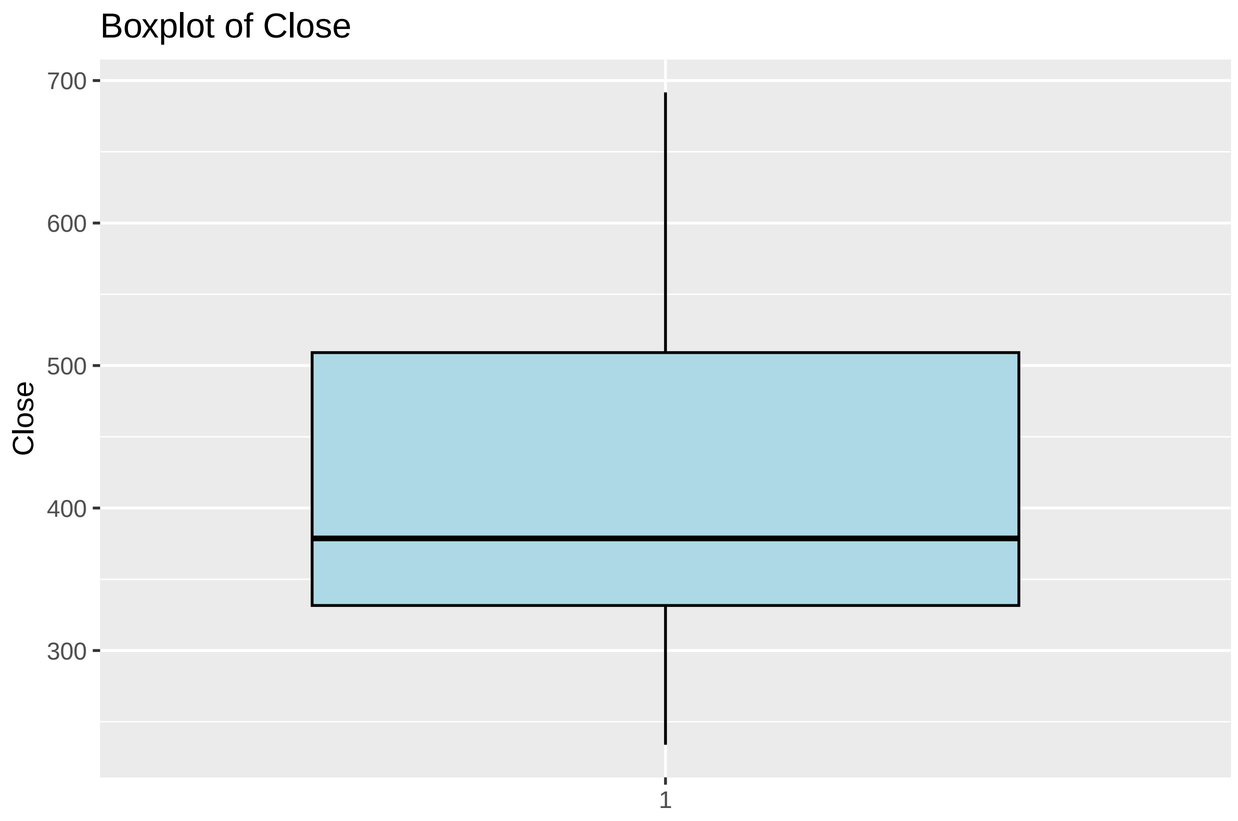
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

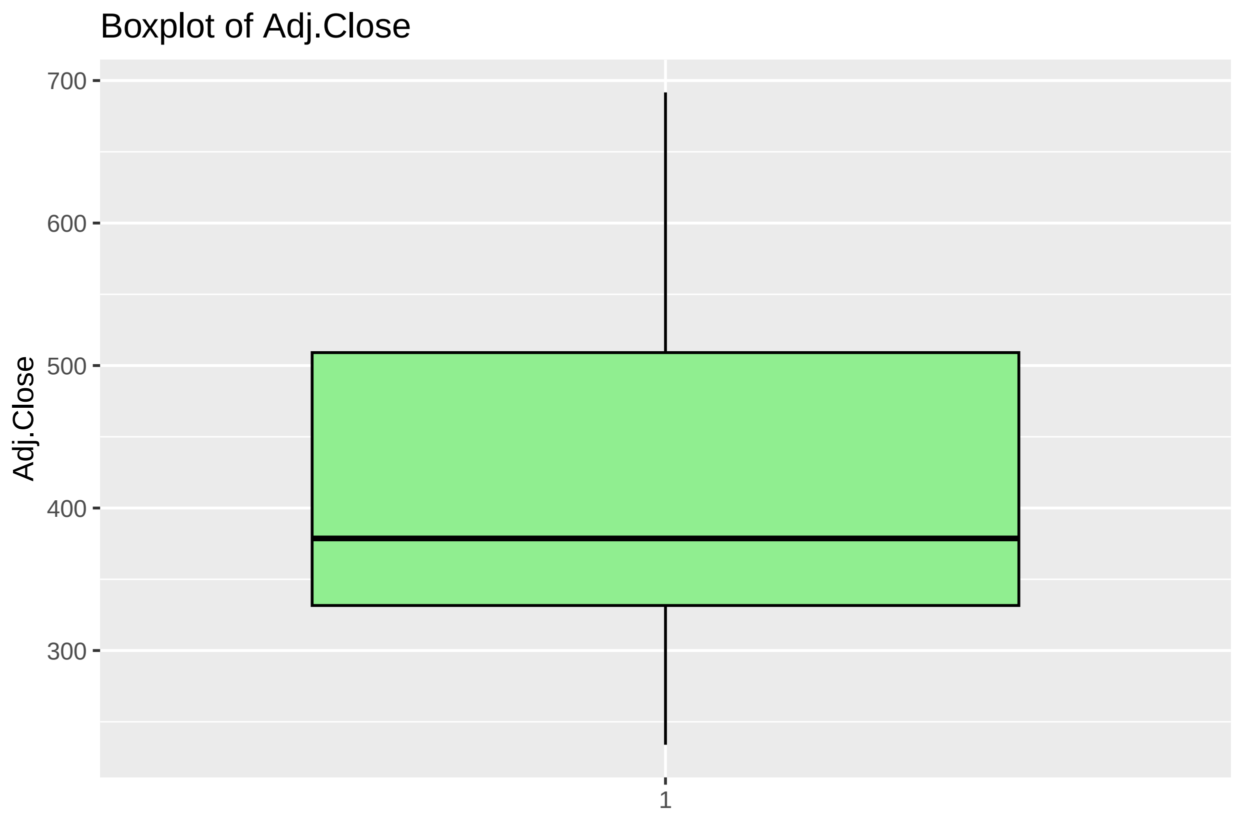
```r
# Remove NULL values from the list
boxplots <- Filter(Negate(is.null), boxplots)

# Print each boxplot
for (plot in boxplots) {
  print(plot)
}
```

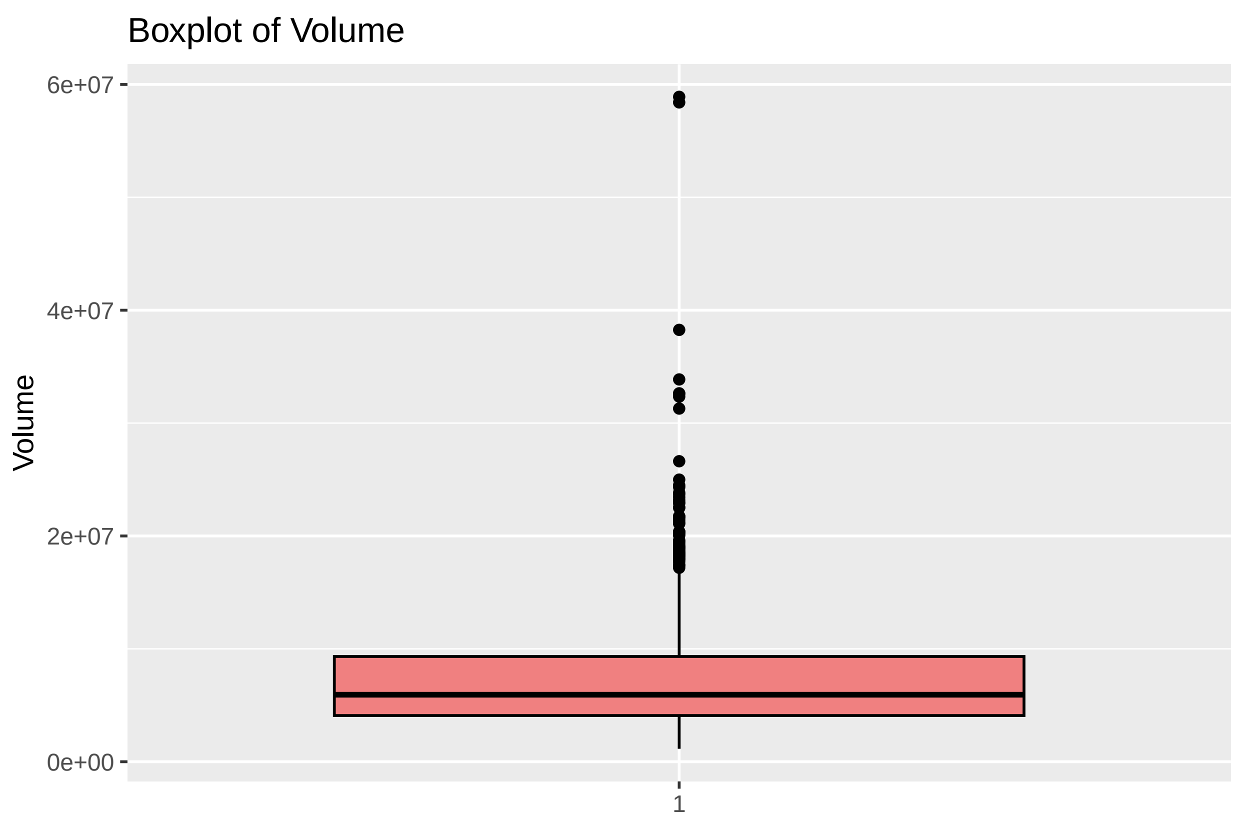

Boxplot of Open
700
600
500
400
300
Open
1


Boxplot of High
700
600
500
400
300
High
1


Boxplot of Low
Low
700
600
500
400
300
1

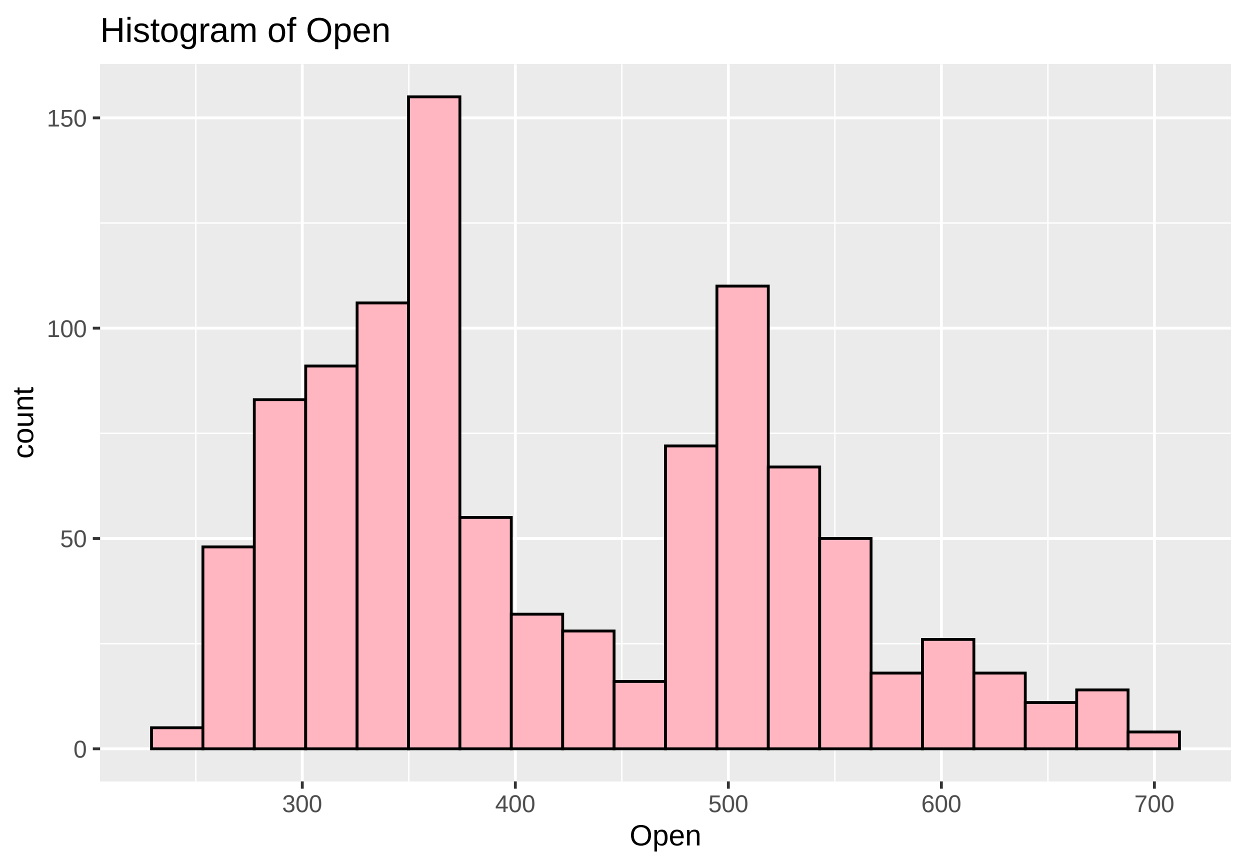

Boxplot of Close
Close
700
600
500
400
300
1

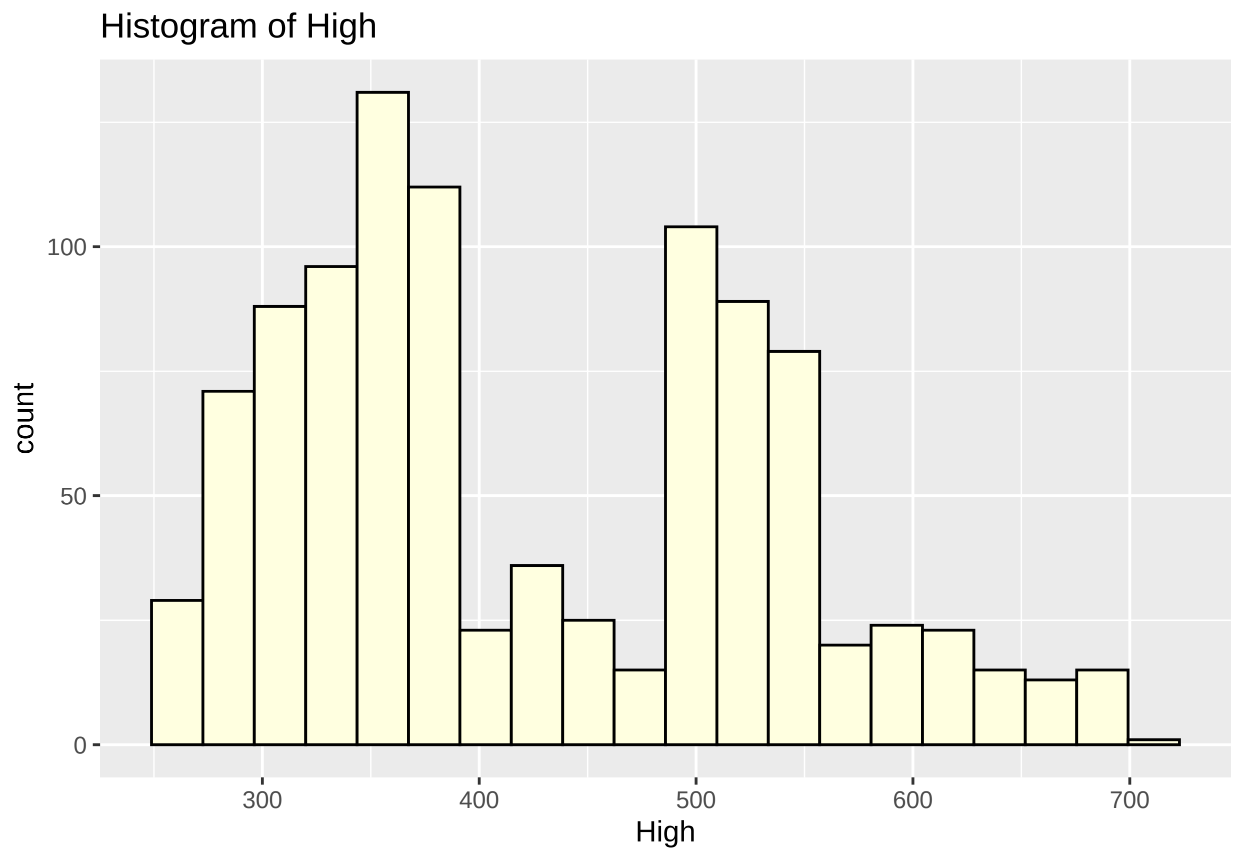

Boxplot of Adj.Close
Adj.Close
700
600
500
400
300
1

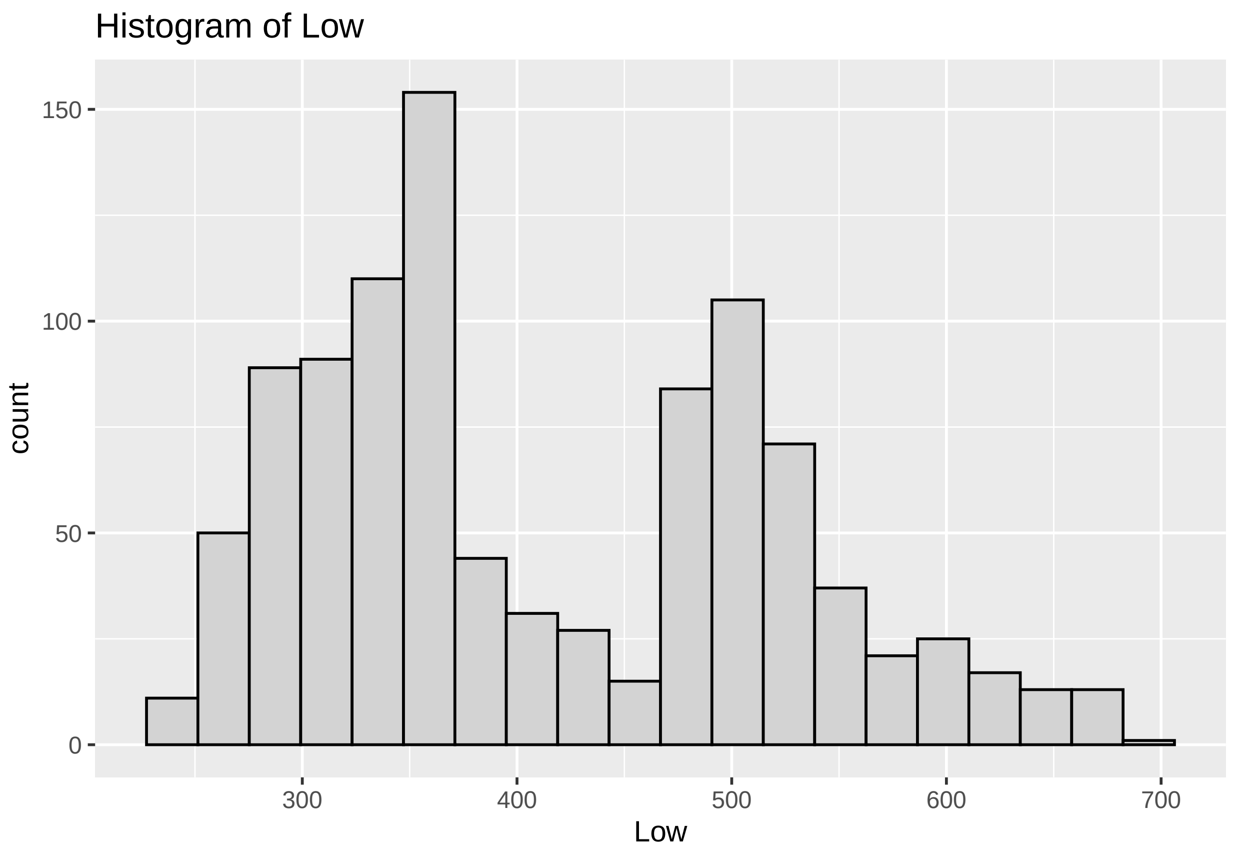


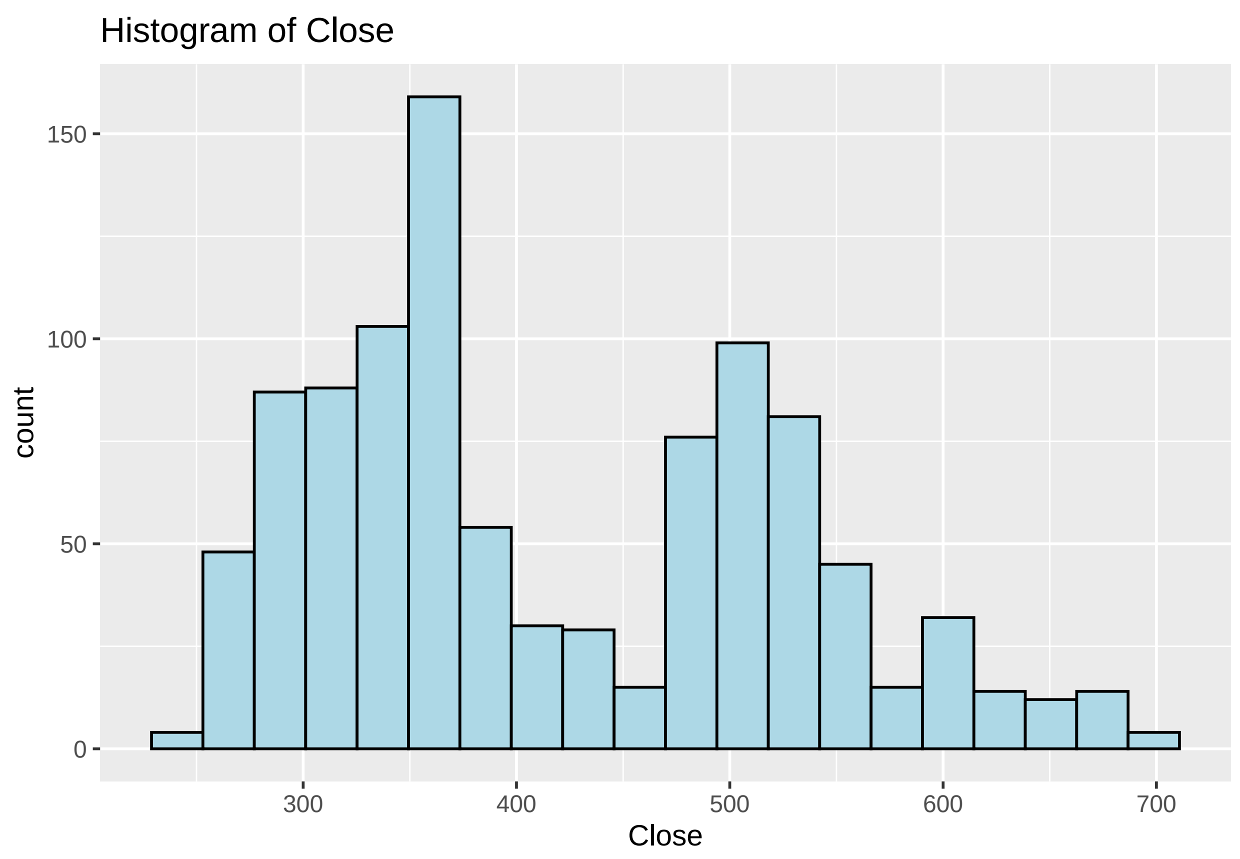
Create Histograms

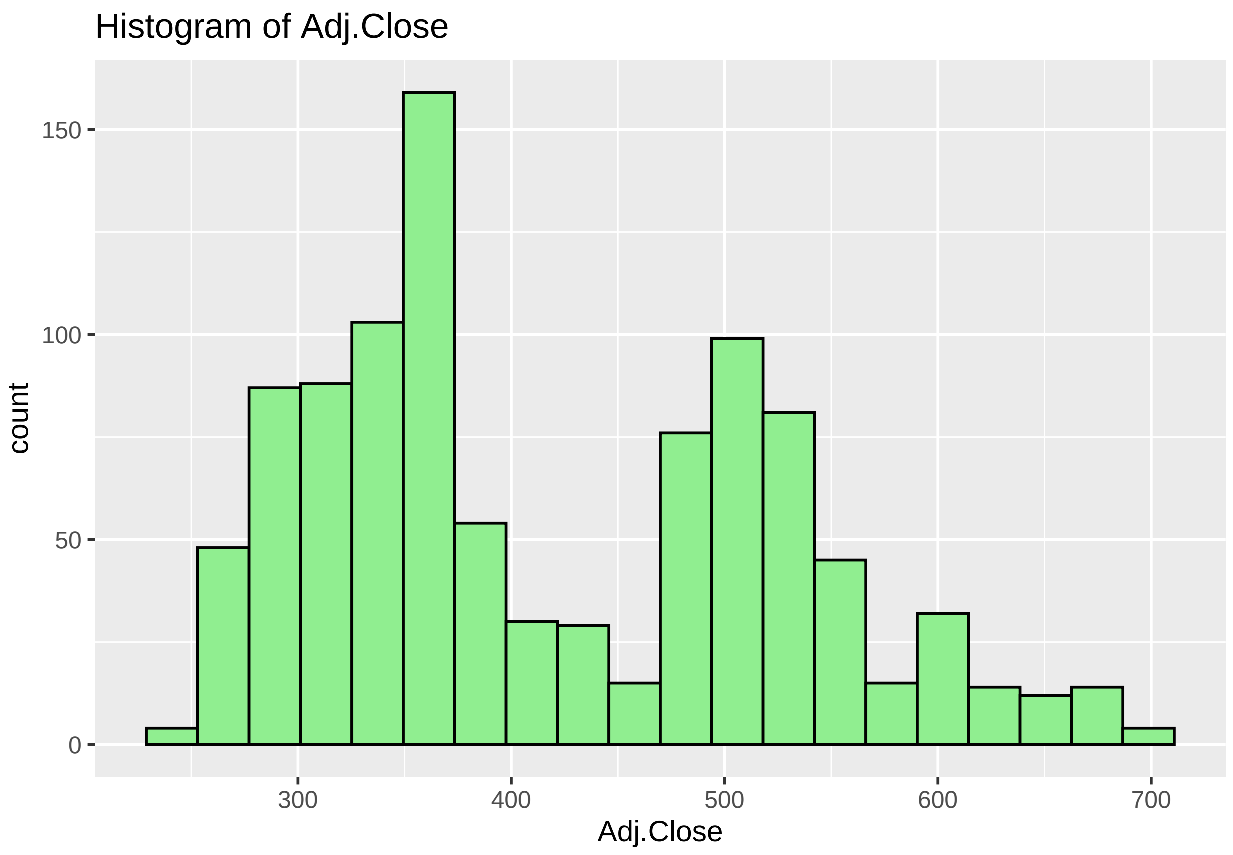
```r
# Create histograms for each variable in nflx_df except Date
histograms <- lapply(seq_along(names(nflx_df)), function(i) {
  var <- names(nflx_df)[i]
  if (var != "Date") {
    ggplot(nflx_df, aes_string(x = var)) +
      geom_histogram(bins = 20, fill = set_colors(i), color = "black") +
      labs(x = var) +
      ggtitle(paste("Histogram of", var))
  }
})

# Remove NULL values from the list
histograms <- Filter(Negate(is.null), histograms)

# Print each histogram
for (plot in histograms) {
  print(plot)
}
```

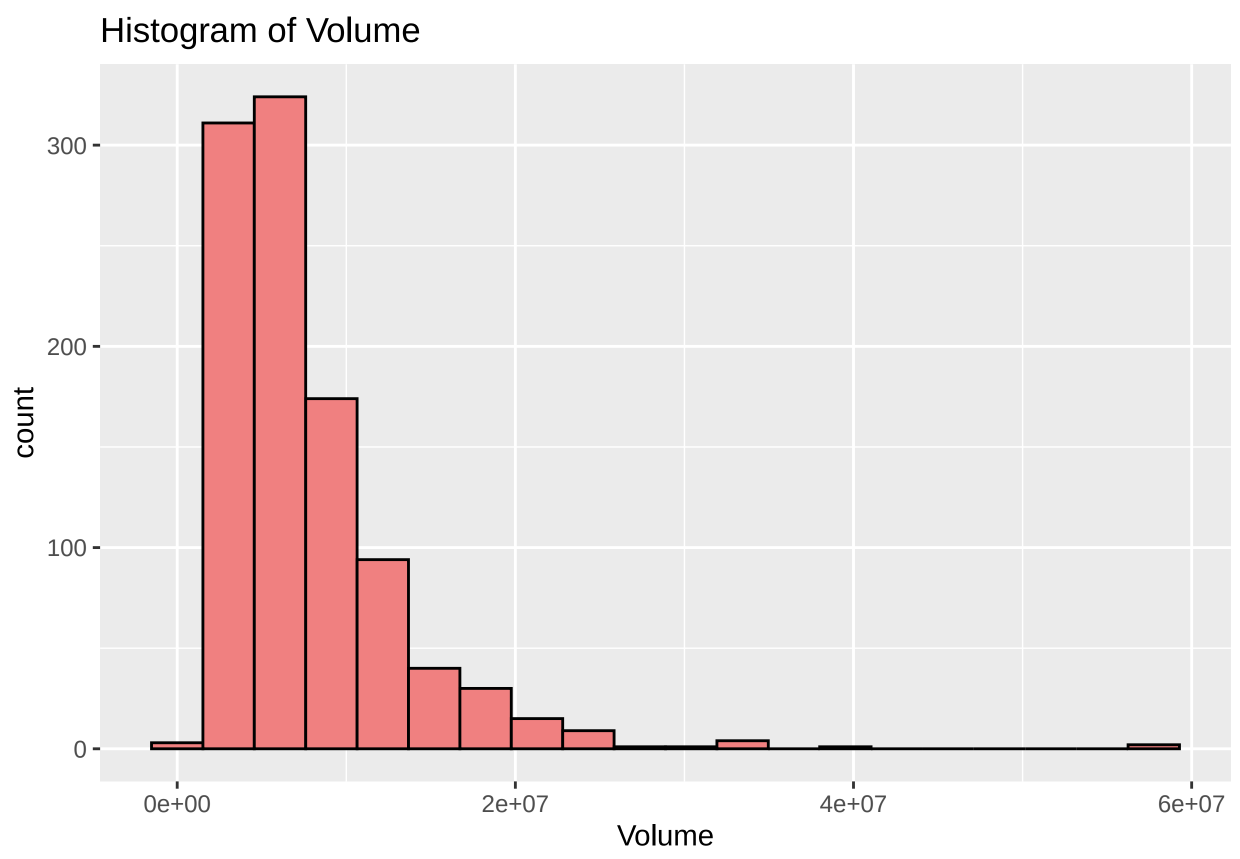

Histogram of Open
count
150
100
50
0
300
400
500
600
700
Open


Histogram of High
count
100
50
0
300
400
500
600
700
High


Histogram of Low
count
Low
0
50
100
150
300
400
500
600
700

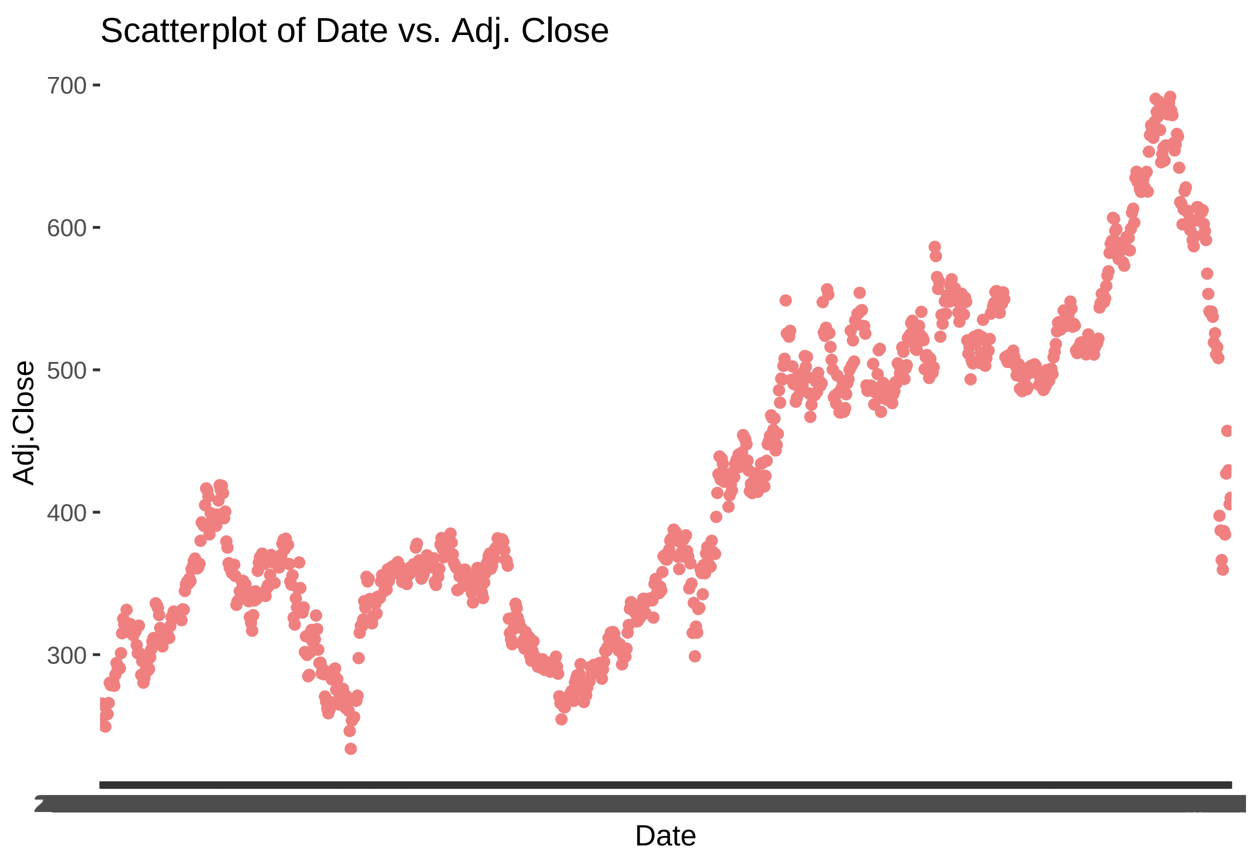

Histogram of Close
count
150
100
50
0
300
400
500
600
700
Close

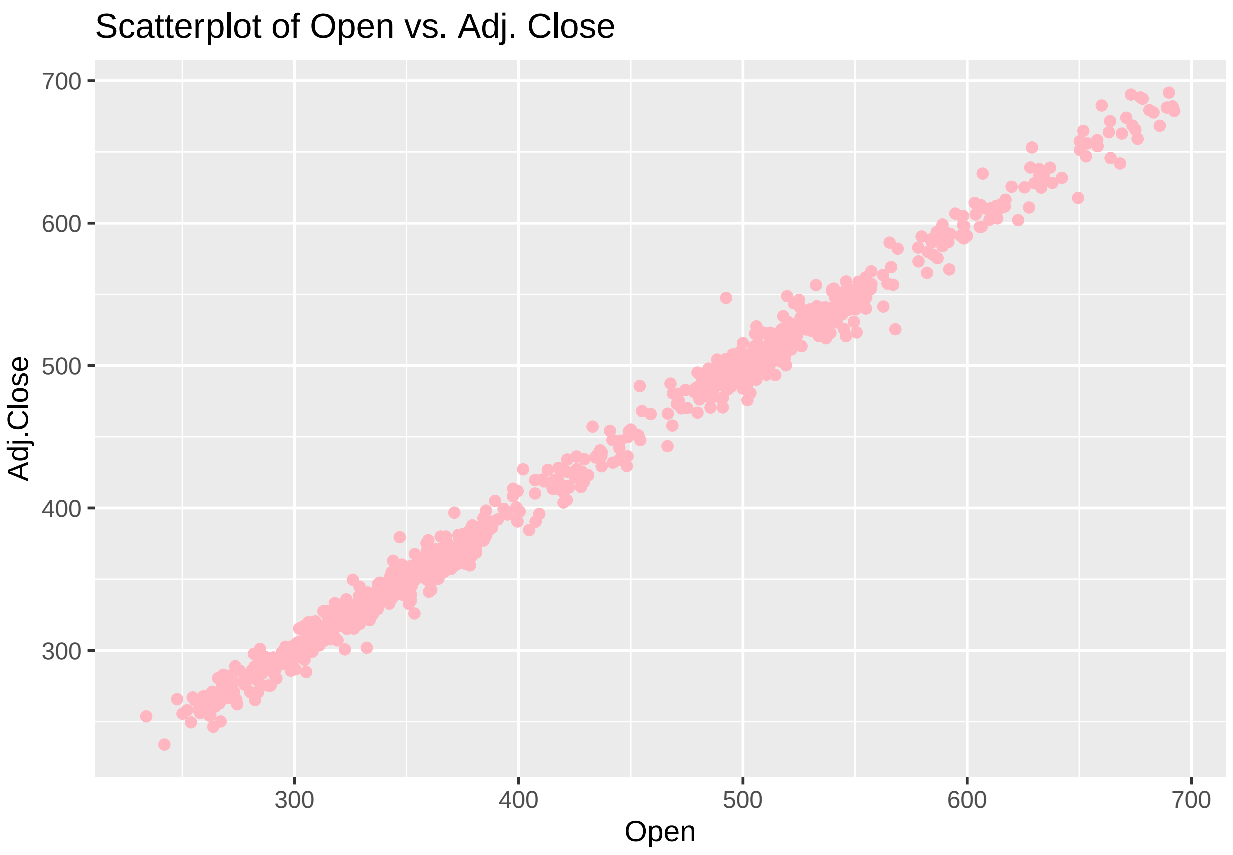

Histogram of Adj.Close
count
150
100
50
0
300
400
500
600
700
Adj.Close

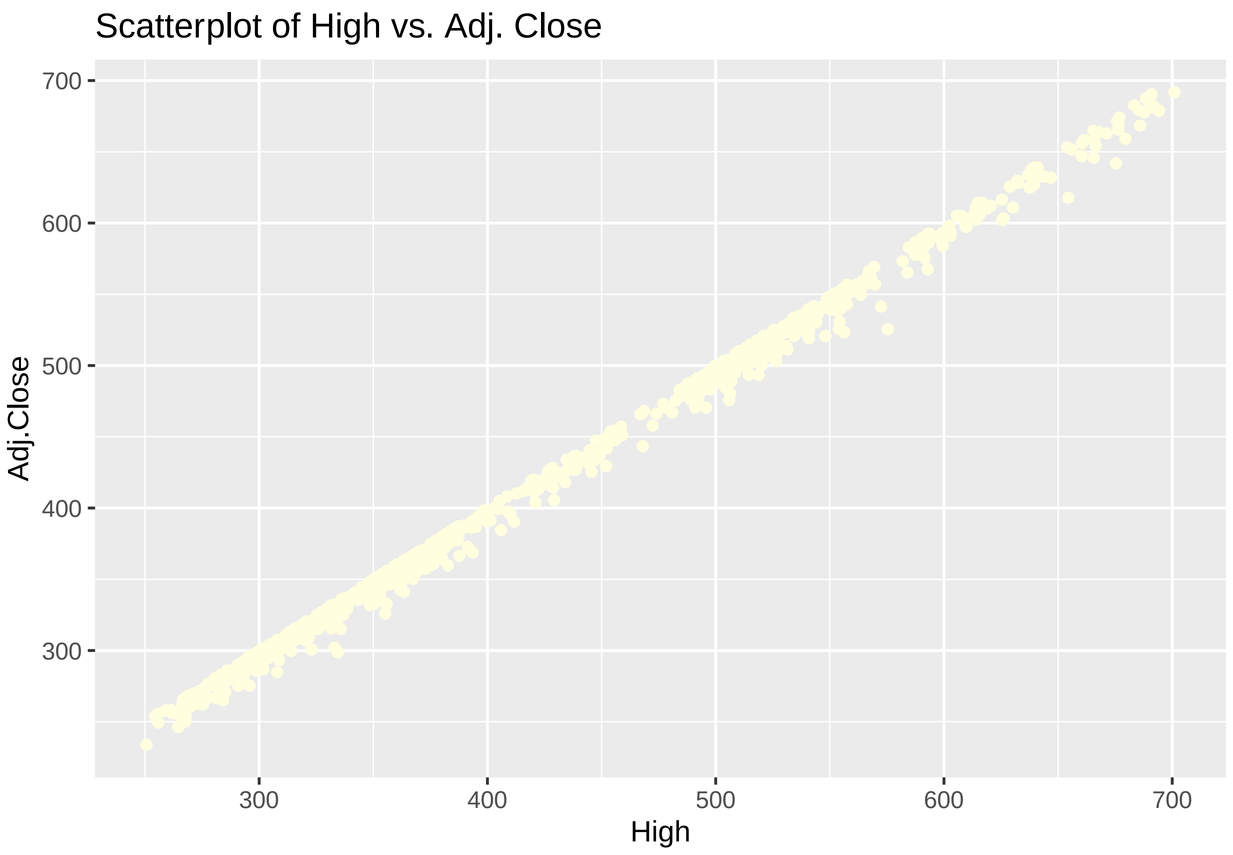


Create Scatterplots

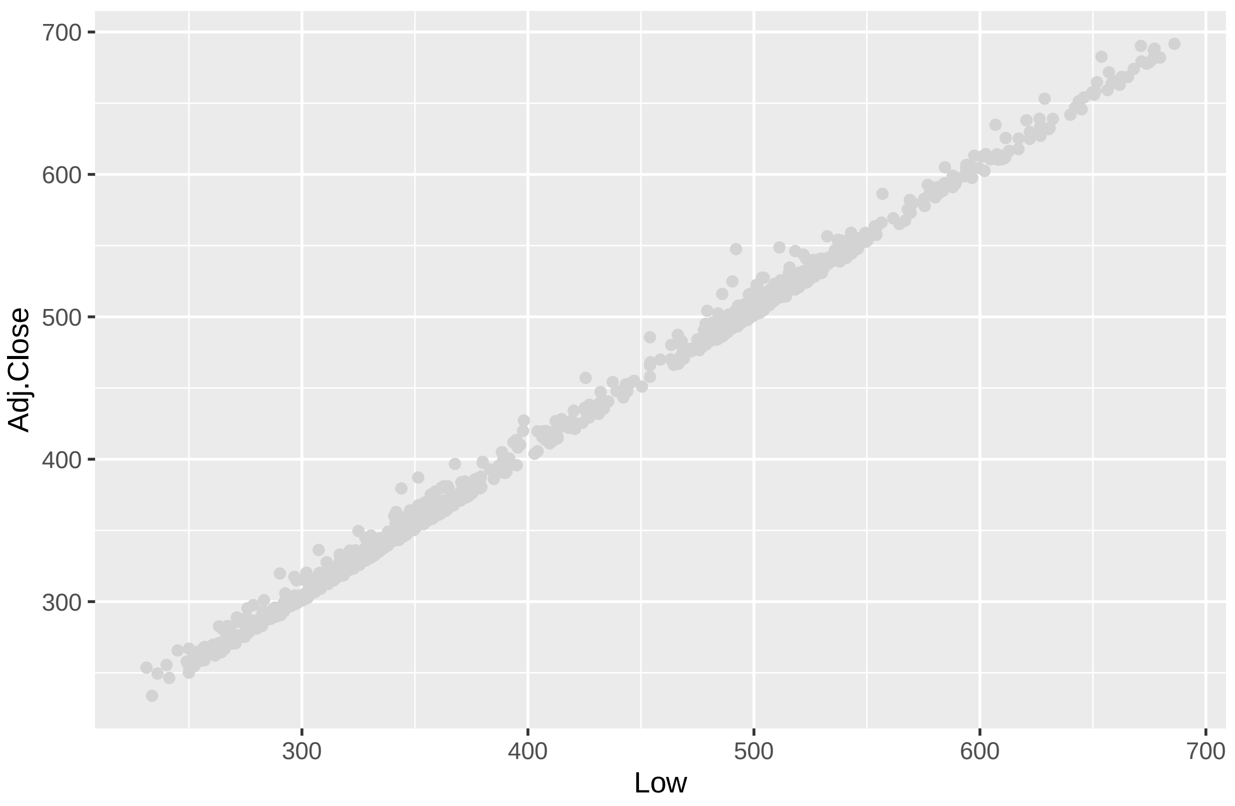
```
# Generate a distinct color for each variable
set_color_scatter <- function(index) {
  colors <- c("skyblue", "lightcoral", "lightpink", "lightyellow", "lightgrey", "lightblue")
  colors[(index %% length(colors)) + 1]
}

# Create scatterplots for each variable in nflx_df showing the relationship with Adj. Close
scatterplots <- lapply(seq_along(names(nflx_df)), function(i) {
  var <- names(nflx_df)[i]
  if (var != "Adj.Close") {
    ggplot(nflx_df, aes_string(x = var, y = "Adj.Close")) +
      geom_point(color = set_color_scatter(i)) +
      labs(x = var, y = "Adj.Close") +
      ggtitle(paste("Scatterplot of", var, "vs. Adj. Close"))
  }
})

# Remove NULL values from the list
scatterplots <- Filter(Negate(is.null), scatterplots)

# Print each scatterplot
for (plot in scatterplots) {
  print(plot)
}
```

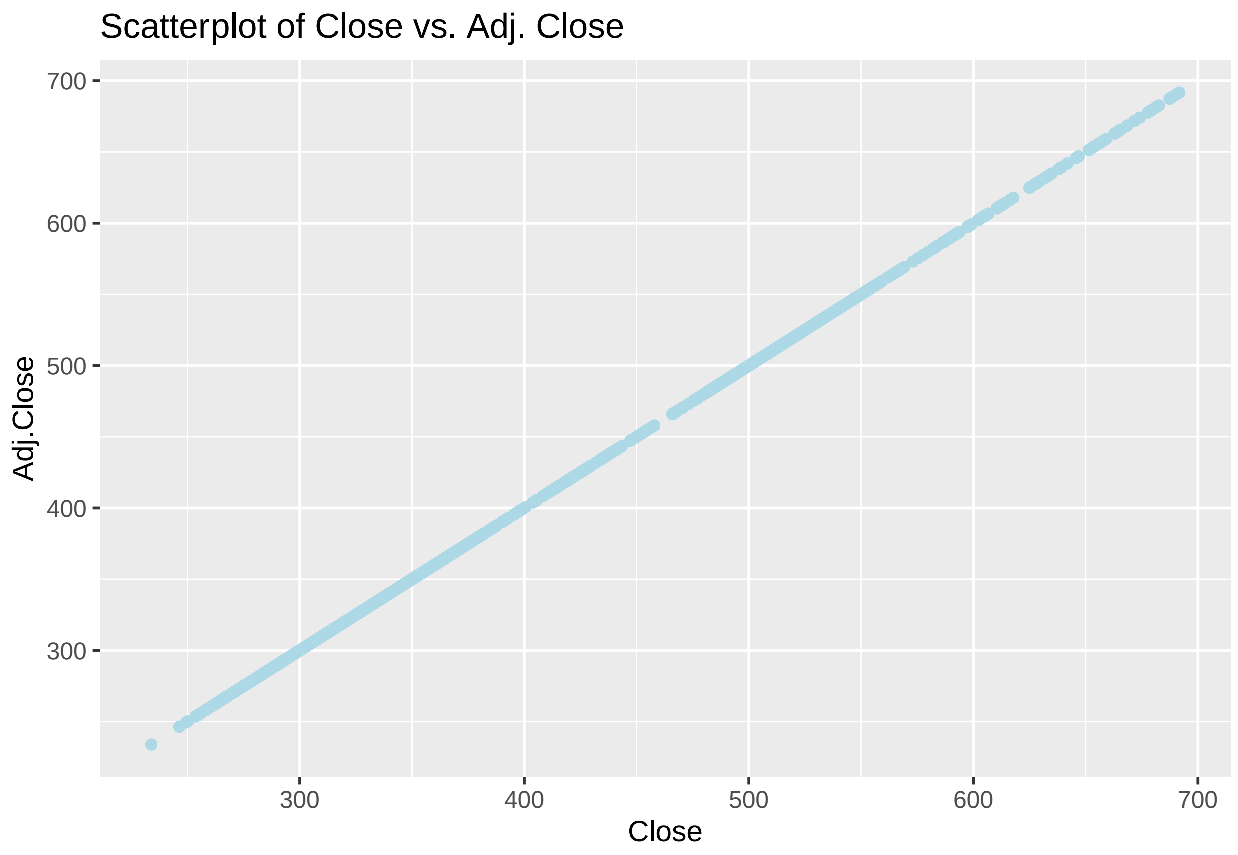

Scatterplot of Date vs. Adj. Close
700
600
500
400
300
Adj.Close
Date

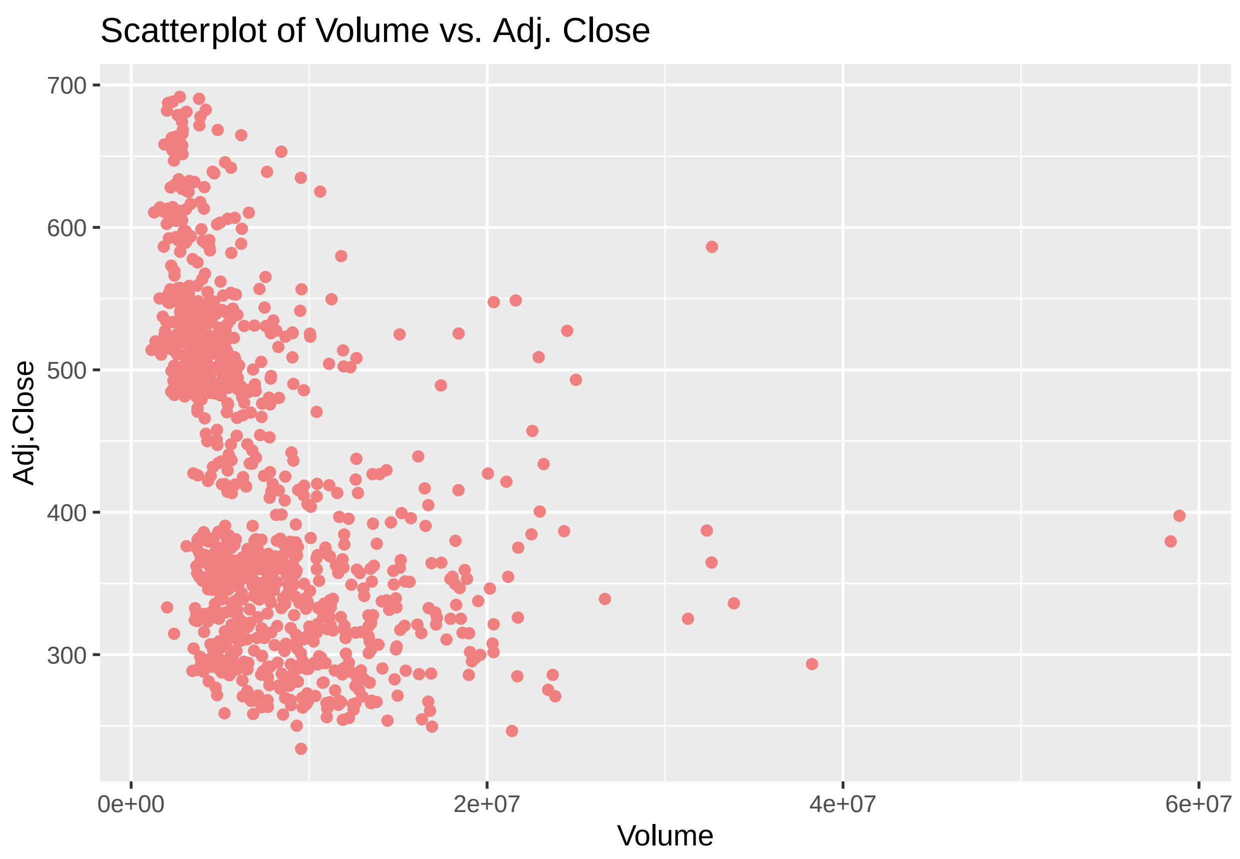

Scatterplot of Open vs. Adj. Close
Adj.Close
700
600
500
400
300
300
400
500
600
700
Open


Scatterplot of High vs. Adj. Close
Adj.Close
700
600
500
400
300
300
400
500
600
700
High

Scatterplot of Low vs. Adj. Close




Scatterplot of Close vs. Adj. Close
Adj.Close
700
600
500
400
300
300
400
500
600
700
Close



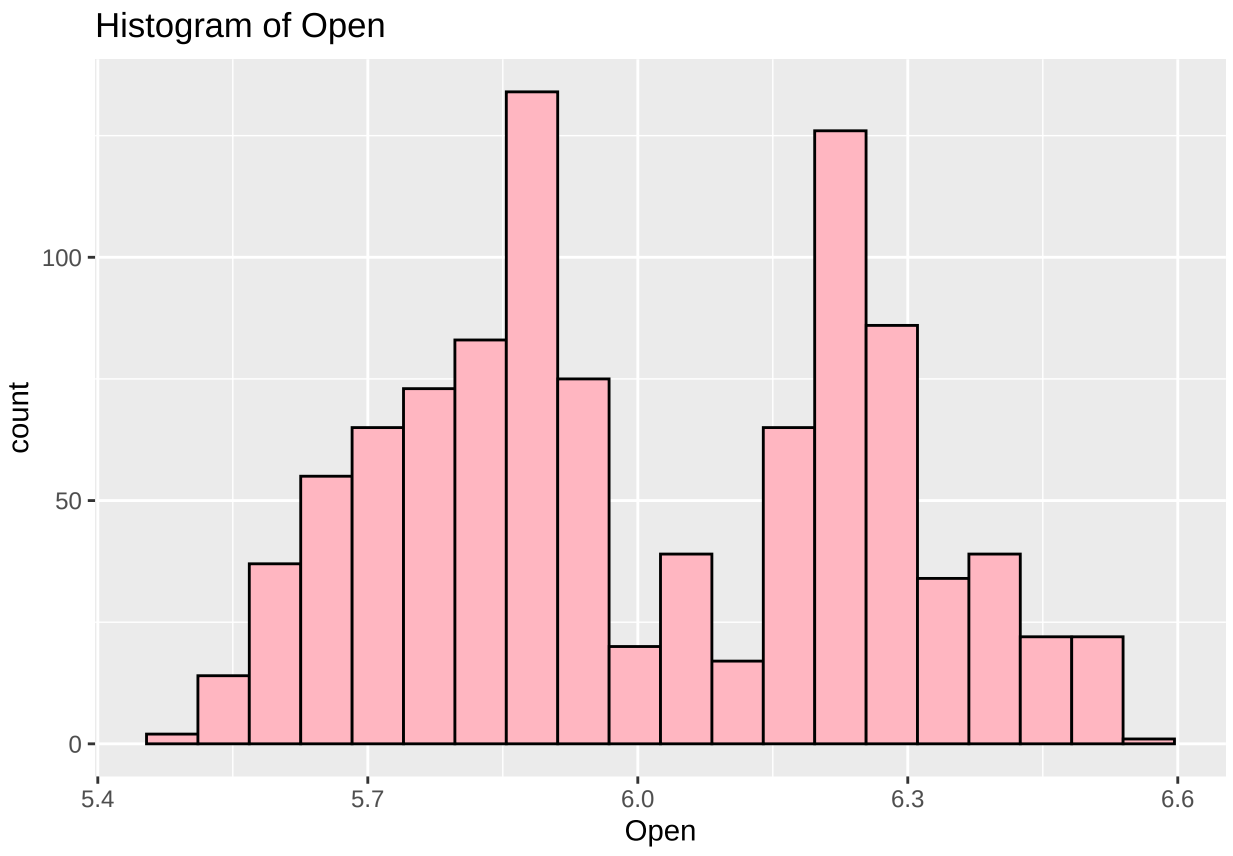
TODO: Not sure if using log transformations or any transformations is good for stock market data. This could lead to distorting true values and predictions since there are actual open, high, low, etc. numbers that represent an actual price or volume in time.

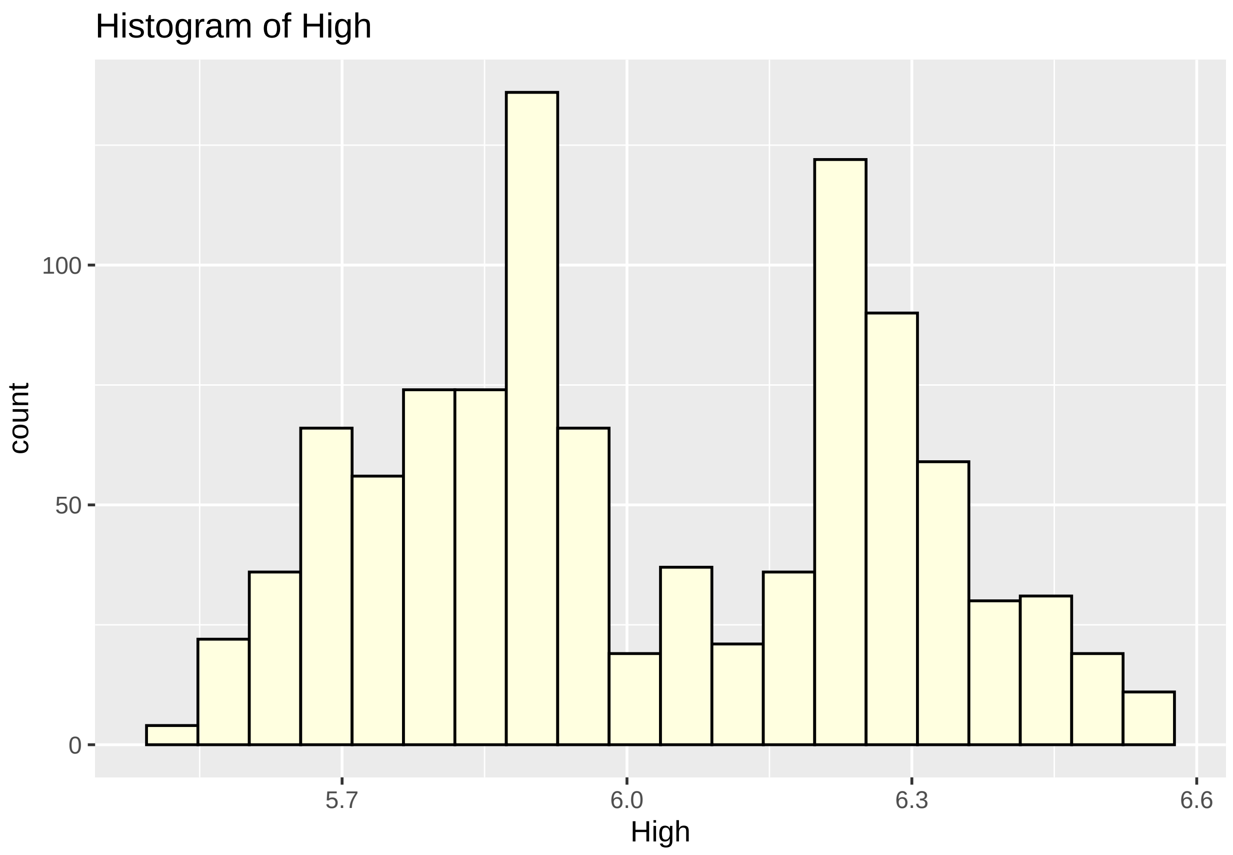
```r
# Create a copy of nflx_df
nflx_df_log <- nflx_df

# Exclude the "Date" and "Adj.Price" columns
cols_to_transform <- names(nflx_df_log)[!(names(nflx_df_log) %in% c("Date", "Adj.Price"))]

# Apply log transformation to numeric columns
for(col in cols_to_transform) {
  if(is.numeric(nflx_df_log[[col]])) {
    nflx_df_log[[col]] <- log(nflx_df_log[[col]])
  }
}

# Print the head of the transformed dataframe
print(head(nflx_df_log))
```

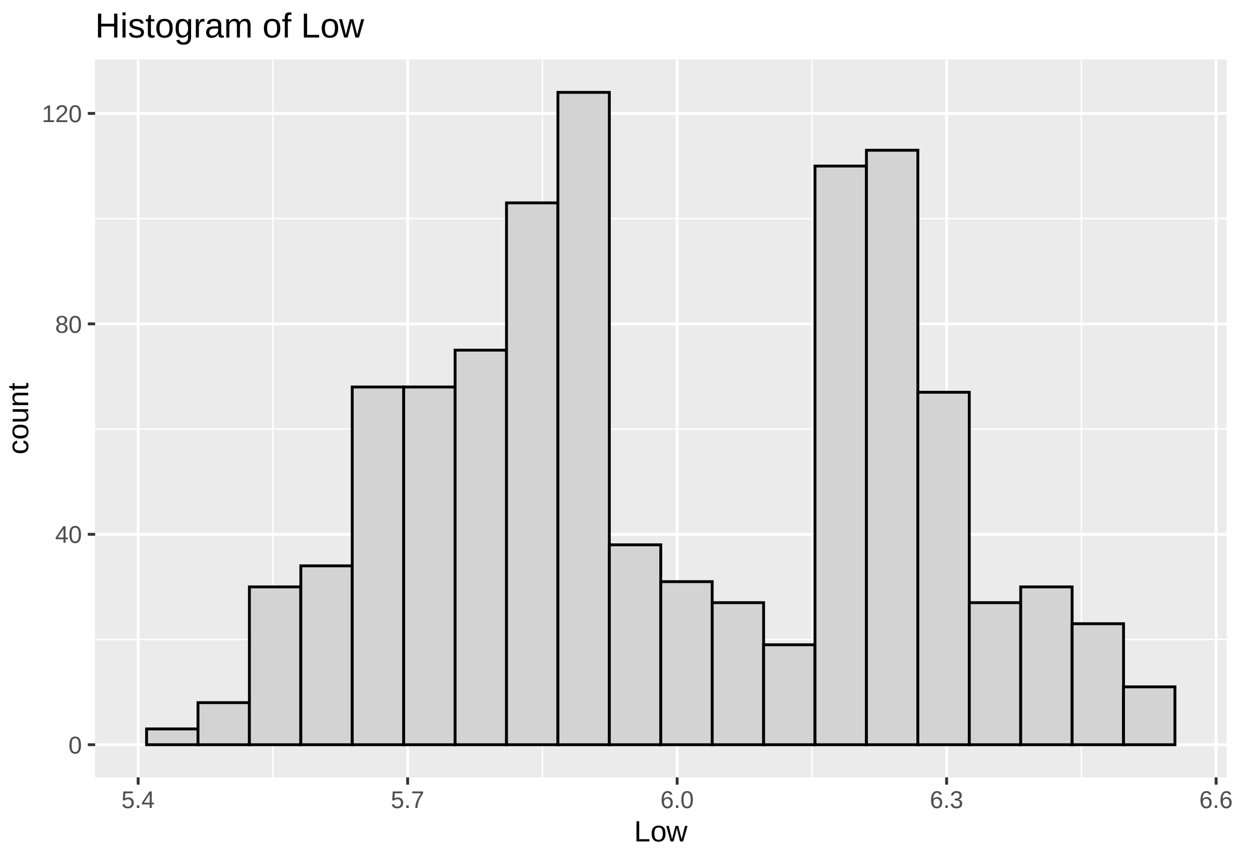
```
##         Date     Open     High      Low    Close Adj.Close   Volume
## 1 2018-02-05 5.568345 5.590614 5.521581 5.538357  5.538357 16.29172
## 2 2018-02-06 5.512218 5.586124 5.501258 5.582443  5.582443 16.34887
## 3 2018-02-07 5.585674 5.607455 5.577198 5.578068  5.578068 16.01068
## 4 2018-02-08 5.587548 5.589568 5.521461 5.521861  5.521861 16.04625
## 5 2018-02-09 5.536744 5.544396 5.464298 5.519339  5.519339 16.64323
## 6 2018-02-12 5.529984 5.557407 5.517453 5.552766  5.552766 15.95967
```

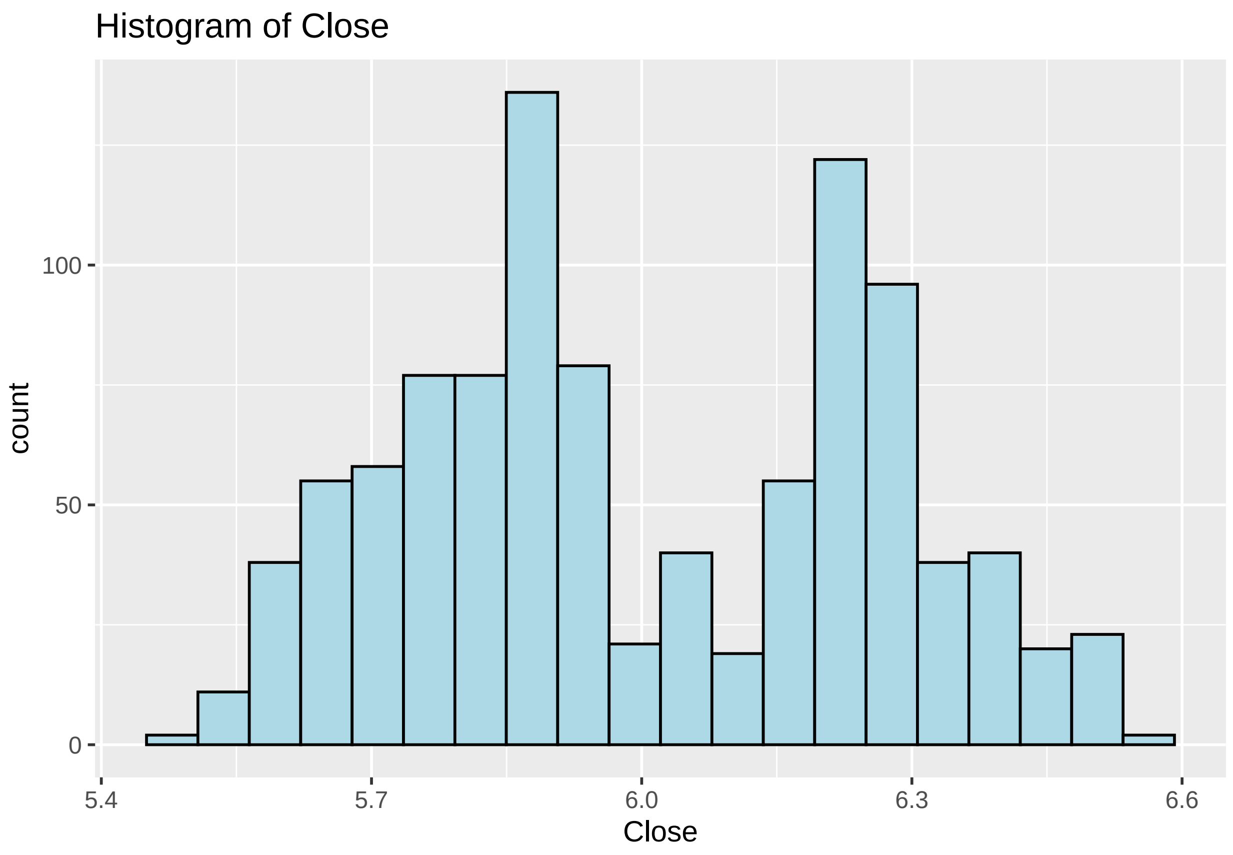
Create Histograms Using Normalized Data

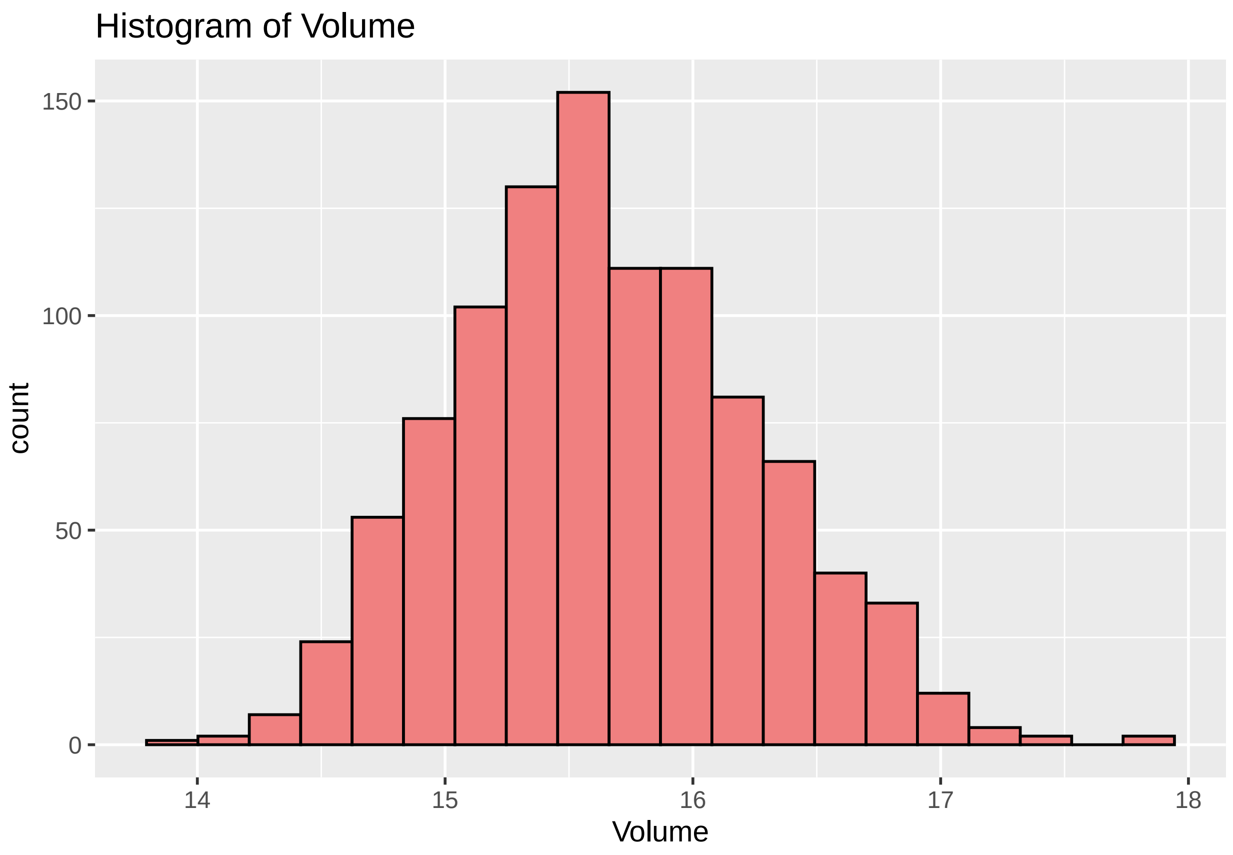
```r
# Create histograms for each variable in nflx_df except Date
histograms <- lapply(seq_along(names(nflx_df_log)), function(i) {
  var <- names(nflx_df_log)[i]
  if (var != "Date" && var != "Adj.Close") {
    ggplot(nflx_df_log, aes_string(x = var)) +
      geom_histogram(bins = 20, fill = set_colors(i), color = "black") +
      labs(x = var) +
      ggtitle(paste("Histogram of", var))
  }
})

# Remove NULL values from the list
histograms <- Filter(Negate(is.null), histograms)

# Print each histogram
for (plot in histograms) {
  print(plot)
}
```




Histogram of High
count
100
50
0
5.7
6.0
6.3
6.6
High


Histogram of Low
count
Low
0
40
80
120
5.4
5.7
6.0
6.3
6.6


Histogram of Close
count
100
50
0
5.4
5.7
6.0
6.3
6.6
Close


Histogram of Volume
count
150
100
50
0
14
15
16
17
18
Volume