# Continuous Data (1 of 3)

*July 30, 2023. -=- This chapter is being heavily edited; It is very much Work in Progress*

## Univariate Continuous Data

1. Reading Data and Attaching Data to Memory

```
data(mtcars)
attach(mtcars)
```

## Measures of Central Tendency

2. In R, we can summarize continuous data using descriptive statistics such as measures of central tendency (mean, median, and mode).
3. Measure the mean and median of the `wt` of all the cars in the dataframe `mtcars`

```
# Mean of wt in the mtcars dataframe
mean(mtcars$wt)
```

```
[1] 3.21725
```

```
# Median of wt in the mtcars dataframe
median(mtcars$wt)
```

```
[1] 3.325
```

4. In the above code, we calculate the mean and median of the mpg column using the `mean()` and `median()` functions, respectively.
5. To calculate the mode of the mpg column, we first load the `modeest` package using the `library()` function, and then use the `mfv()` function to compute the mode.

```r
# Mode of wt in the mtcars dataframe
library(modeest)
mfv(mtcars$mpg) # Mode
```

```
[1] 10.4 15.2 19.2 21.0 21.4 22.8 30.4
```

6. Note that the mtcars dataset contains continuous data, and so it does not have a well-defined mode in the traditional sense. The `mfv()` function computes the mode using a kernel density estimator, which may not always correspond to a single value in the dataset.

## Measures of Variability

1. In R, we can calculate measures of variability (range, interquartile range, variance, and standard deviation).
2. To calculate these statistics, we can use built-in functions in R such as `range()`, `IQR()`, `var()`, and `sd()`.

```r
# Standard Deviation of wt in the mtcars dataframe
sd(mtcars$wt)
```

```
[1] 0.9784574
```

```r
# Variance of wt in the mtcars dataframe
var(mtcars$wt)
```

```
[1] 0.957379
```

```r
# Range of wt in the mtcars dataframe
range(mtcars$wt)
```

```
[1] 1.513 5.424
```

```r
# Inter-Quartile Range of wt in the mtcars dataframe
IQR(mtcars$wt)
```

```
[1] 1.02875
```

3. Note that the range() function returns the minimum and maximum values in the dataset, while the IQR() function returns the difference between the 75th and 25th percentiles.

## Other functions

```r
# Minimum wt in the mtcars dataframe
min(mtcars$mpg)
```

```
[1] 10.4
```

```r
# Maximum wt in the mtcars dataframe
max(mtcars$mpg)
```

```
[1] 33.9
```

## Summarizing a data column

**`summary()`**

1. Display a summary of `mpg` in the dataframe mtcars using `summary()`

```r
summary(mtcars$mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10.40   15.43   19.20   20.09   22.80   33.90 
```

**`describe()`**

2. Display a summary of the `mpg` in the dataframe mtcars using `describe()`

```r
library(psych)
```

```
Registered S3 method overwritten by 'psych':
  method         from
  plot.residuals rmutil
```

```r
describe(mtcars$mpg)
```

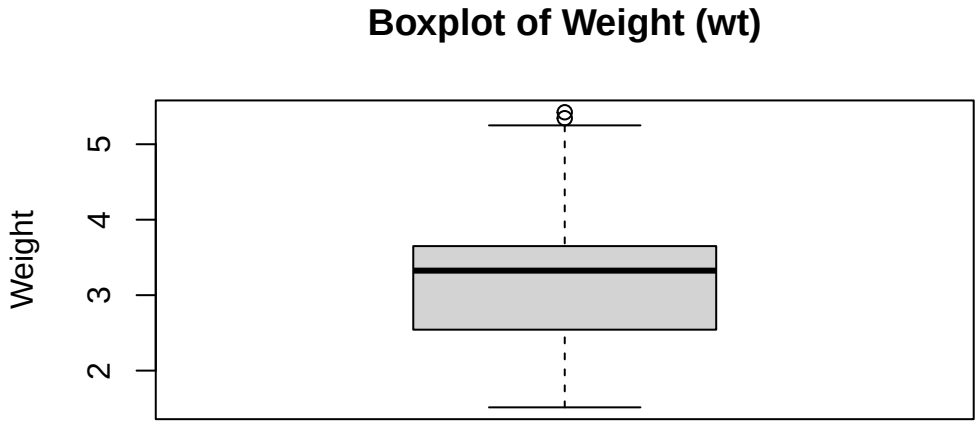
```
   vars  n  mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 32 20.09 6.03   19.2    19.7 5.41 10.4 33.9  23.5 0.61    -0.37 1.07
```

## Visualizing Univariate Continuous Data

## Boxplot

1. A boxplot is a graphical representation of the distribution of continuous data.
2. Display the Boxplot of the `wt` of the cars in the `mtcars` dataset

```r
boxplot(mtcars$wt,
        xlab = "Boxplot",
        ylab = "Weight",
        main = "Boxplot of Weight (wt)"
       )
```




Boxplot

3. The resulting boxplot will display the median, quartiles, and any outliers in the data.
4. The box represents the interquartile range, which contains the middle 50% of the data.
5. The whiskers extend to the minimum and maximum non-outlier values, or 1.5 times the interquartile range beyond the quartiles, whichever is shorter.
6. Any points outside of the whiskers are considered outliers and are plotted individually.

## Violin plot

1. A violin plot is similar to a boxplot, but instead of just showing the quartiles, it displays the full distribution of the data using a kernel density estimate.
2. We can create a violin plot in R using the violinplot() function from the vioplot package.

```
# Load the vioplot package
library(vioplot)
```

```
Loading required package: sm
```

```
Package 'sm', version 2.2-5.7: type help(sm) for summary information
```

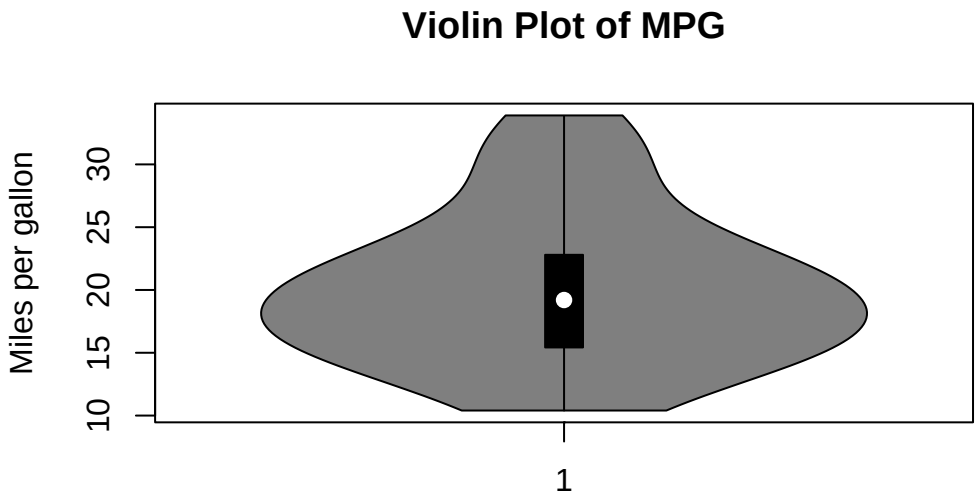
```
Loading required package: zoo
```

```
Attaching package: 'zoo'

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

    as.Date, as.Date.numeric
```

```r
# Create a violin plot of mpg column
vioplot(mtcars$mpg,
        main="Violin Plot of MPG",
        ylab="Miles per gallon")
```

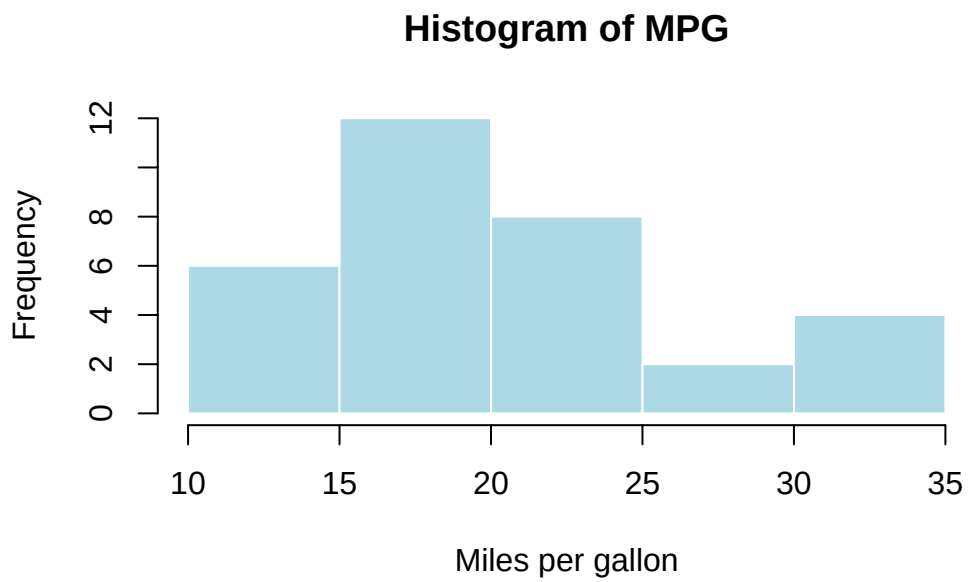



3. In the above code, we create a violin plot of the `mpg` column using the `vioplot()` function. The `main` argument is used to specify the title of the plot, and the `ylab` argument is used to specify the label for the y-axis.
4. The resulting plot will display the full distribution of the `mpg` data using a kernel density estimate, with thicker sections indicating a higher density of data points.
5. The plot also shows the median, quartiles, and any outliers in the data.

## Histogram

1. A histogram is a plot that shows the frequency of each value or range of values in a dataset.
2. It can be useful for showing the shape of the distribution of the data. We can create a histogram in R using the `hist()` function.

```r
# Create a histogram of mpg column
hist(mtcars$mpg,
     main="Histogram of MPG",
     xlab="Miles per gallon",
     col="lightblue",
     border="white")
```



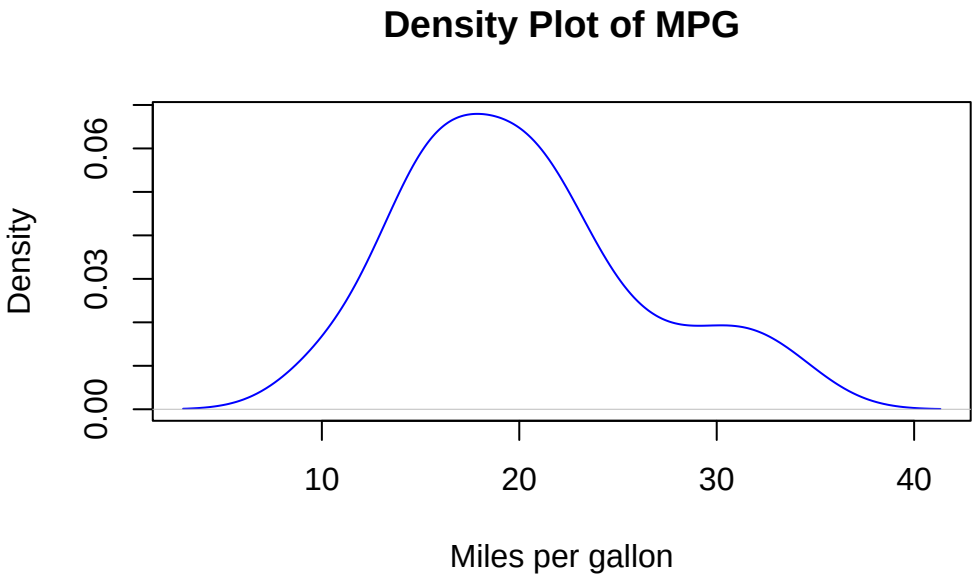

3. We create a histogram of the `mpg` column using the `hist()` function. The main argument is used to specify the title of the plot, and the xlab argument is used to specify the label for the x-axis.
4. The `col` argument is used to set the color of the bars in the histogram, and the border argument is used to set the color of the border around the bars.
5. The resulting histogram will display the frequency of `mpg` values in the dataset, with the bars representing the number of observations falling within a specific range of values.

## Density plot

1. A density plot is similar to a histogram, but instead of displaying the frequency of each value, it shows the probability density of the data.

```r
# Create a density plot of mpg column
plot(density(mtcars$mpg),
```

```
main="Density Plot of MPG",
xlab="Miles per gallon",
col="blue")
```



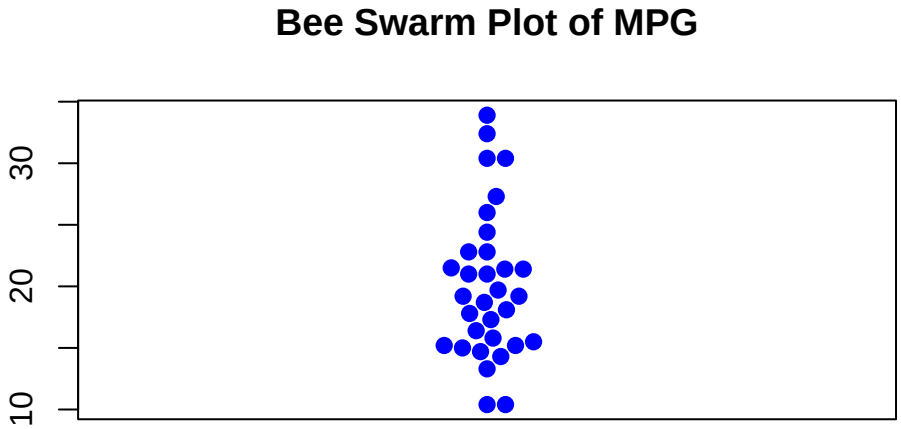

2. In the above code, we create a density plot of the mpg column using the `density()` function.
3. The `plot()` function is used to plot the resulting density object.
4. The `main` argument is used to specify the title of the plot, and the `xlab` argument is used to specify the label for the x-axis.
5. The `col` argument is used to set the color of the plot line.
6. The resulting plot will display the probability density of `mpg` values in the dataset, with the curve representing the distribution of the data.

## Bee Swarm plot

1. A bee swarm plot is a plot that displays all of the individual data points along with a visual representation of their distribution.
2. It can be useful for displaying the distribution of small datasets.

```r
# Load the beeswarm package
library(beeswarm)

# Create a bee swarm plot of mpg column
beeswarm(mtcars$mpg,
         main="Bee Swarm Plot of MPG",
         pch=16,
         cex=1.2,
         col="blue")
```




3. In the above code, we load the `beeswarm` package using the `library()` function.
4. We then create a bee swarm plot of the `mpg` column using the `beeswarm()` function.
5. The `main` argument is used to specify the title of the plot.
6. The `pch` argument is used to set the type of points to be plotted, and the `cex` argument is used to set the size of the points.
7. The `col` argument is used to set the color of the points.
8. The resulting plot will display the individual `mpg` values in the dataset as points on a horizontal axis, with no overlap between points. This provides a visual representation of the distribution of the data, as well as any outliers or gaps in the data.