DS 116 Data Visualization

Single numeric variable

Habet Madoyan

American University of Armenia

Section 1

Histogram

Movies data set

```
movies <- read.csv('Data/movies.csv', stringsAsFactors = F)
str(movies)
## 'data.frame':    2911 obs. of  32 variables:
##  $ title             : chr  "Zoom" "Zoolander 2" "Zombieland" "Zodiac"
##  $ genre_first       : chr  "Action" "Comedy" "Adventure" "Crime" ...
##  $ year              : int  2006 2016 2009 2007 1998 2012 2005 2008 199
##  $ duration          : int  83 102 88 162 116 157 101 101 119 90 ...
##  $ gross_adjusted    : num  14142117 29451448 86365946 39077724 2978040
##  $ budget_adjusted   : int  42555556 51065177 26964263 76858659 7519018
##  $ gross             : int  11631245 28837115 75590286 33048353 1980338
##  $ budget            : int  35000000 50000000 23600000 65000000 5000000
##  $ cast_facebook_likes: int  5022 24107 28011 36928 1209 2759 32232 638
##  $ reviews           : int  176 376 998 966 232 1198 338 490 709 297 ..
##  $ index             : num  1.22 1.02 1.14 1.18 1.5 ...
##  $ Rated             : chr  "PG" "PG-13" "R" "R" ...
##  $ Genre             : chr  "Action, Adventure, Comedy" "Comedy" "Adven
##  $ Writer            : chr  "Adam Rifkin (screenplay), David Berenbaum
##  $ Actors            : chr  "Tim Allen, Courteney Cox, Chevy Chase, Spe
##  $ Plot              : chr  "Former superhero Jack is called back to wo
##  $ Language          : chr  "English" "English, Italian, Spanish" "Engl
```

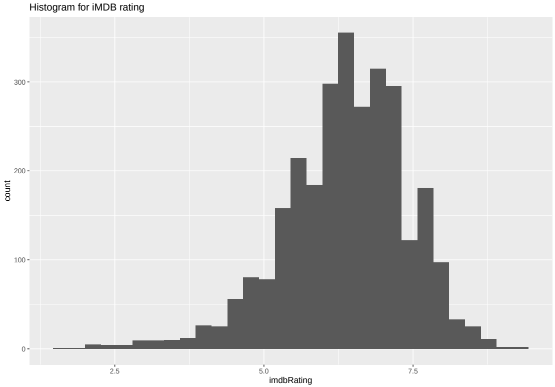
Histogram

- A histogram displays the frequency and distribution for a range of quantitative groups.
- Bar charts compare quantities for different categories, a histogram technically compares the number of observations across a range of value ‘bins’ using the size of lines/bars to represent the quantitative counts.
- Histogram allows to understand the shape of the distribution of the data

Doing with ggplot

- You need to specify only one aesthetics: x
- Use geom_histogram as a geometric object

```
ggplot(data = movies, aes(x = imdbRating)) +
  geom_histogram() + ggtitle('Histogram for iMDB rating')
```



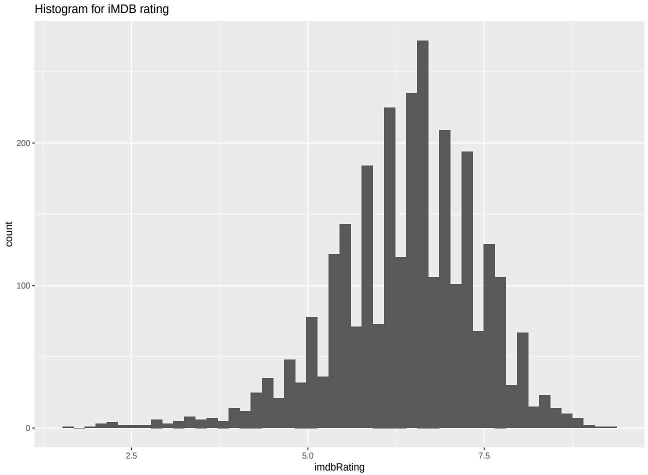
Doing with ggplot2

- in ggplot you can either specify the number of bins (bins) or bin width (binwidth). One can be derived from another.
- The default value for number of bins is 30.
- There is no one golden rule on choosing number of bins, however in general
    - More bins (smaller binwidth) will result in higher detalization
    - Less bins (larger binwidth) will result in lower detalization

Doing with ggplot

Number of bins = 50

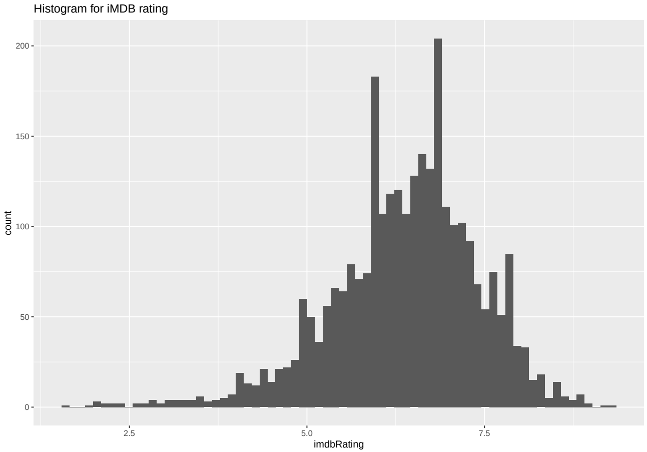
```
ggplot(data = movies, aes(x = imdbRating)) + geom_histogram(bins = 50) +
  ggtitle('Histogram for iMDB rating')
```



Doing with ggplot

Number of bins = 70

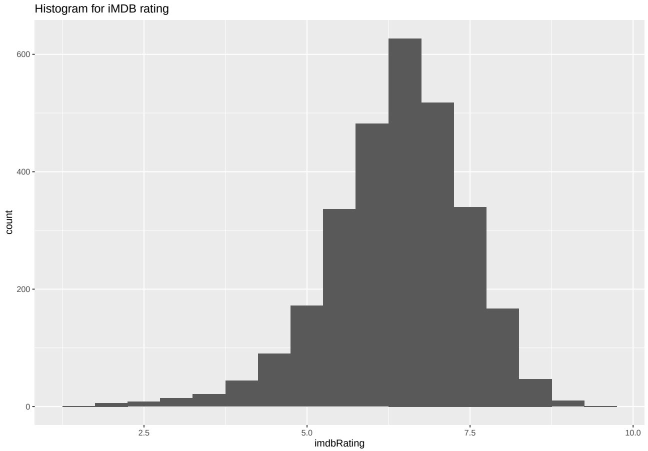
```
ggplot(data = movies, aes(x = imdbRating)) +
  geom_histogram(bins = 70) + ggtitle('Histogram for iMDB rating')
```



Doing with ggplot

Alternatively you can set up the binwidth: binwidth $= 0.5$

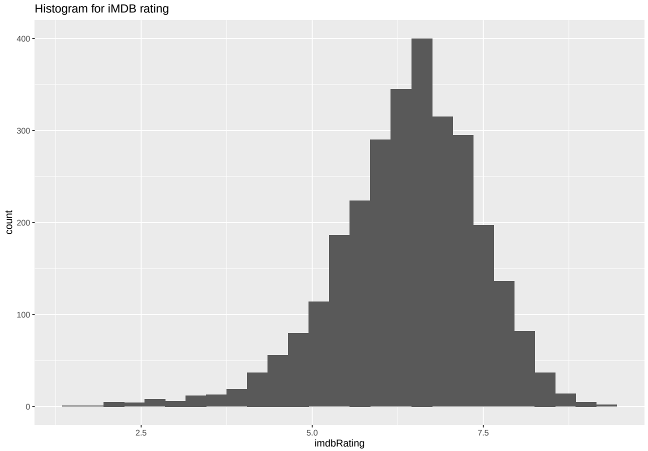
```
ggplot(data = movies, aes(x = imdbRating)) +
  geom_histogram(binwidth = 0.5) + ggtitle('Histogram for iMDB rating')
```



Doing with ggplot

Binwidth = 0.3

```r
ggplot(data = movies, aes(x = imdbRating)) + geom_histogram(binwidth = 0.3)
  ggtitle('Histogram for iMDB rating')
```



Choosing the number of bins

- There is no one golden rule on how many bins need to be there
- Do try and error until you get histogram that can be interpreted
- However, there are few approaches for the calculation of optimal number of bins

Methods for choosing the number of bins

1. Square root choice

$$k = \lceil \sqrt{n} \rceil$$

2. Rice rule

$$k = \lceil 2\sqrt[3]{n} \rceil$$

3. Sturges' formula

$$k = \lceil \log_2 n \rceil + 1$$

Methods for choosing the binwidth

1. Scott's normal reference rule (when data is approximately normal)

$$h = \frac{3.49\hat{\sigma}}{\sqrt[3]{n}}$$

2. Freedman–Diaconis' rule - a variation of Scott's rule but less sensitive to outliers

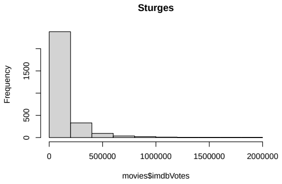
$$h = 2\frac{\mathrm{IQR}(x)}{\sqrt[3]{n}}$$

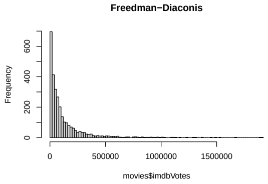
Doing in R

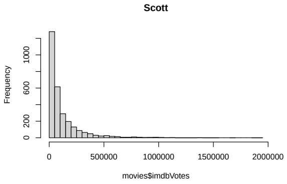
Basic R functionality for histogram allows to directly state the method for bin calculations

```
par(mfrow = c(2,2))
hist(movies$imdbVotes, breaks = 'sturges', main = 'Sturges', )
hist(movies$imdbVotes, breaks = 'fd', main = 'Freedman-Diaconis')
hist(movies$imdbVotes, breaks = 'scott', main = 'Scott')
```

Doing in R


Sturges
Frequency
0
500
1500
0
500000
1000000
1500000
2000000
movies$imdbVotes


Freedman−Diaconis
Frequency
0
200
400
600
0
500000
1000000
1500000
movies$imdbVotes

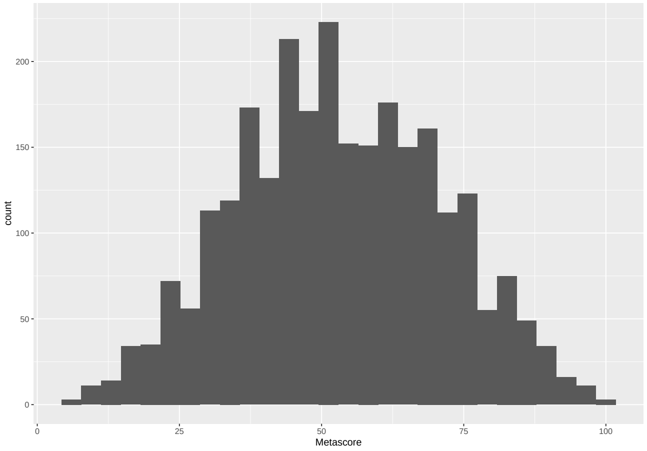

Scott
Frequency
0
200
600
1000
0
500000
1000000
1500000
2000000
movies$imdbVotes

Doing in ggplot2

In ggplot2 you need to calculate the number of the bins/binwidth then provide the result as an argument
Example: Rice rule

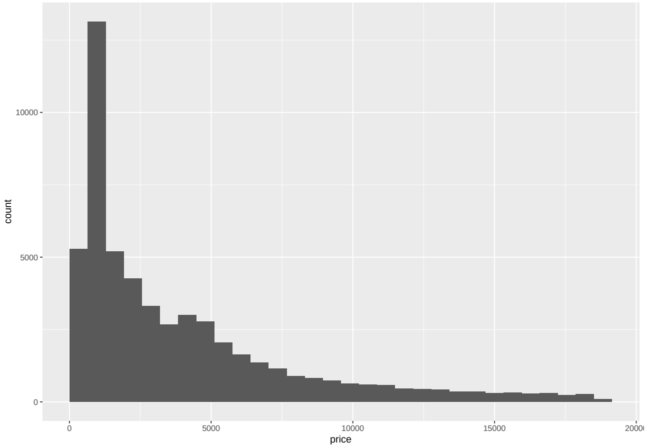
```r
n <- length(movies$Metascore[!is.na(movies$Metascore)])
k <- ceiling(2*(n^(1/3)))
ggplot(movies, aes(x = Metascore)) + geom_histogram(bins = k)
```

Doing in ggplot2



Bumpy distribution

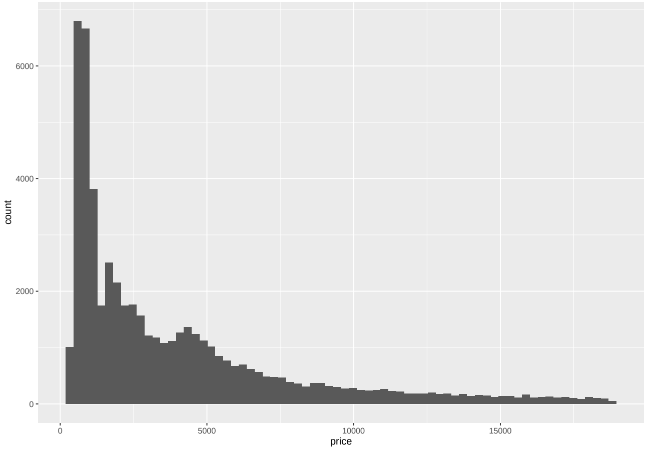
```
data("diamonds")
ggplot(diamonds, aes(price)) + geom_histogram()
```



Bumpy distribution

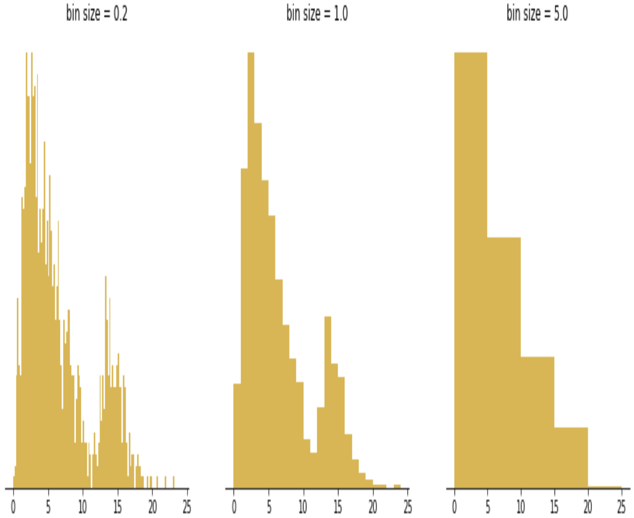
Increase the number of bins, the bump becomes more apparent

```
ggplot(diamonds, aes(price)) + geom_histogram(bins = 70)
```



Bumpy distribution

How can you detect it ?

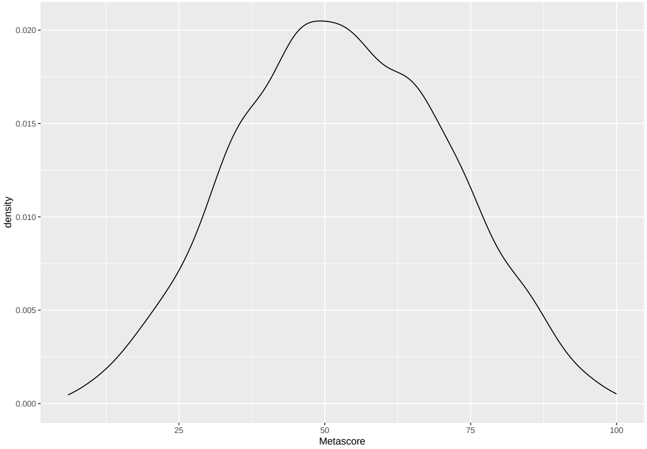


Density plots

- To visualize the distribution of the continuous variable, you can also use kernel density estimate - smoothed version of the histogram.
- In ggplot it is done with the geom_density()

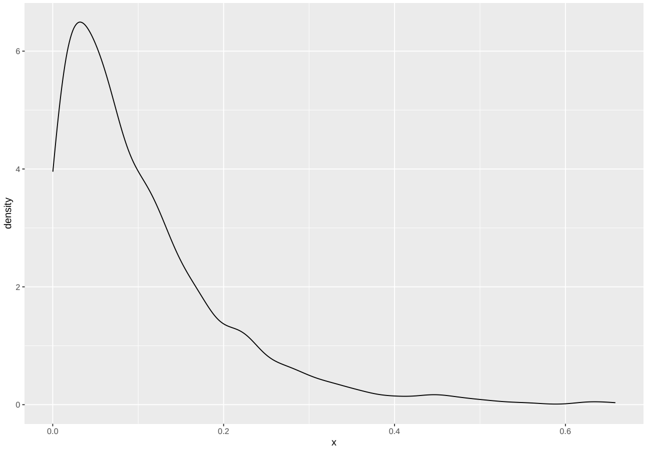
Density plots

```
ggplot(movies, aes(x = Metascore)) + geom_density()
```



Density plots: exponential distribution

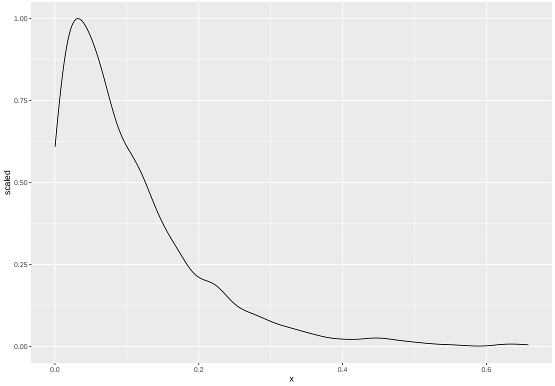
```r
set.seed(1)
x <- rexp(1000, rate = 10)
ggplot() + geom_density(aes(x))
```



Density plots

Scale the y axis

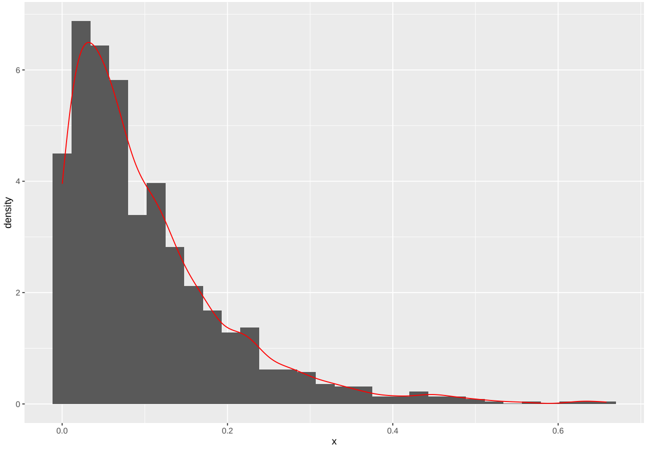
```
ggplot() + geom_density(aes(x, y = after_stat(scaled)))
```



Note: We use `after_stat(scaled)` instead of the deprecated `..scaled..` syntax (changed in ggplot2 3.4.0+).

Density plot over the histogram

```
ggplot(mapping = aes(x = x))  + geom_histogram(aes(y = after_stat(density))
  geom_density(color = 'red')
```



Note: `after_stat(density)` replaces the deprecated `..density..` notation. This makes it explicit that `density` is computed by the stat layer.

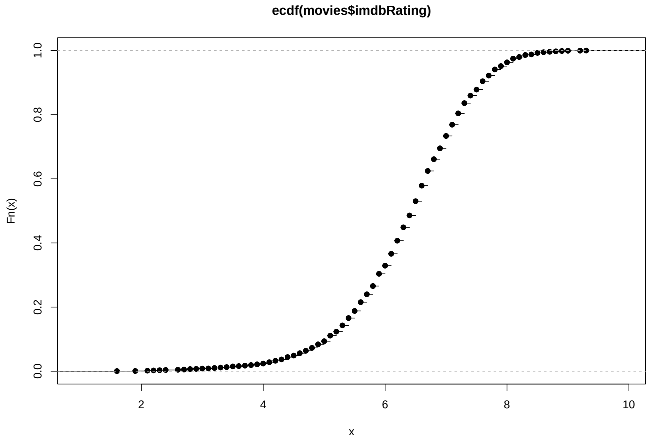
Cumulative distribution function

- The cumulative distribution function (CDF) of a random variable X, or just distribution function of X, evaluated at x is the $P(X < x)$.
- When we have the data, we have the empirical distribution, we can construct Empirical Cumulative Distribution Function:

$$\widehat{F}_n(x) = \frac{\text{number of elements in the sample} \leq x}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{I}(X_i \leq x)$$

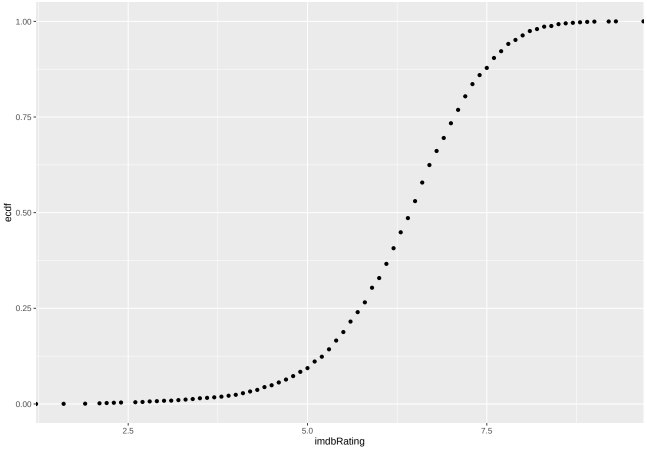
eCDF

```r
plot(ecdf(movies$imdbRating))
```



eCDF

```r
ggplot(movies, aes(x = imdbRating)) + stat_ecdf(geom = 'point')
```

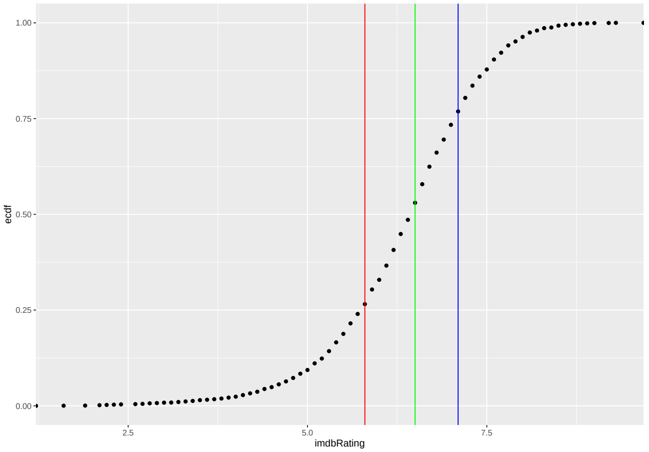


eCDF

To make eCDF more informative we can add quartiles to the plot

```
quant <- quantile(movies$imdbRating, probs = c(0.25,0.5,0.75), na.rm = T)
ggplot(movies, aes(x = imdbRating)) + stat_ecdf(geom = 'point') +
  geom_vline(xintercept = quant, color = c('red', 'green', 'blue'))
```

eCDF

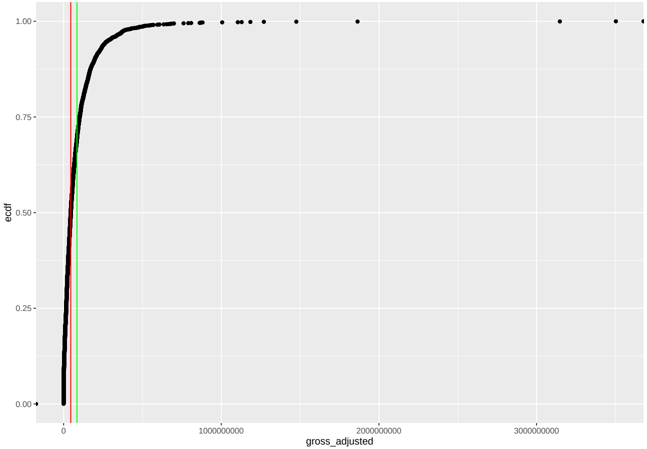


eCDF

Other distribution

```r
med <- median(movies$gross_adjusted)
avg <- mean(movies$gross_adjusted)
ggplot(movies, aes(x = gross_adjusted)) + stat_ecdf(geom = 'point') +
  geom_vline(xintercept = c(med, avg), color = c('red', 'green'))
```

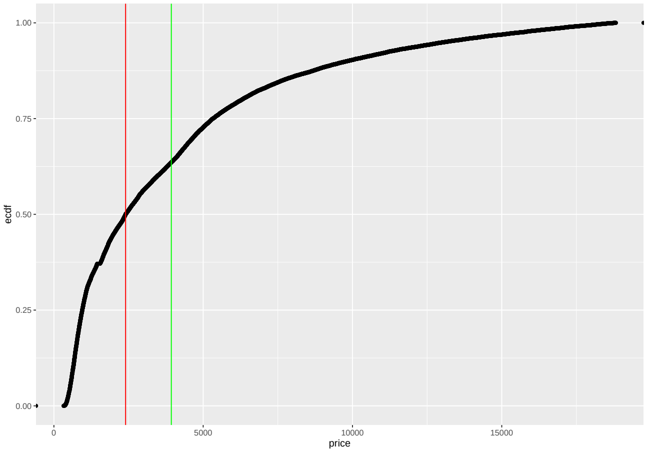
eCDF


1.00
0.75
0.50
0.25
0.00
ecdf
0
1000000000
2000000000
3000000000
gross_adjusted

Another skewed distribution

```r
med <- median(diamonds$price)
avg <- mean(diamonds$price)
ggplot(diamonds, aes(x = price)) + stat_ecdf(geom = 'point') +
  geom_vline(xintercept = c(med, avg), col = c("red", "green"))
```

Another skewed distribution



Section 2

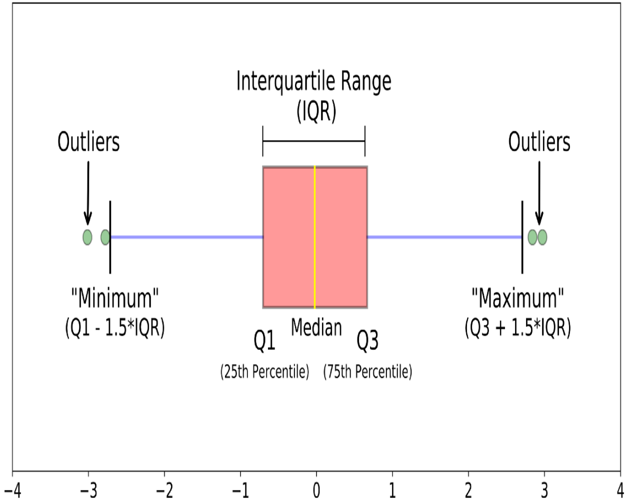
Boxplots

Boxplot

- Boxplot or box-whisker plot, is another way to display the distribution of the continuous variable
- Boxplots are usually used to visualize the distribution of some continuous variable by categories of a categorical variable
- They are also used to detect outliers (non-parametric way)

Boxplot

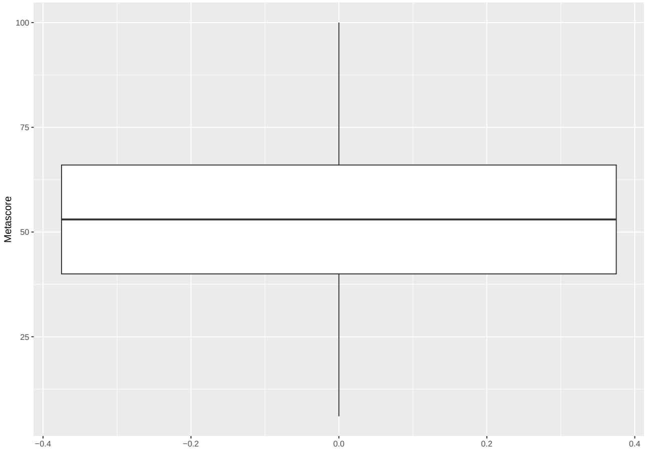
The structure of the boxplot



Boxplot

Vertical Boxplot

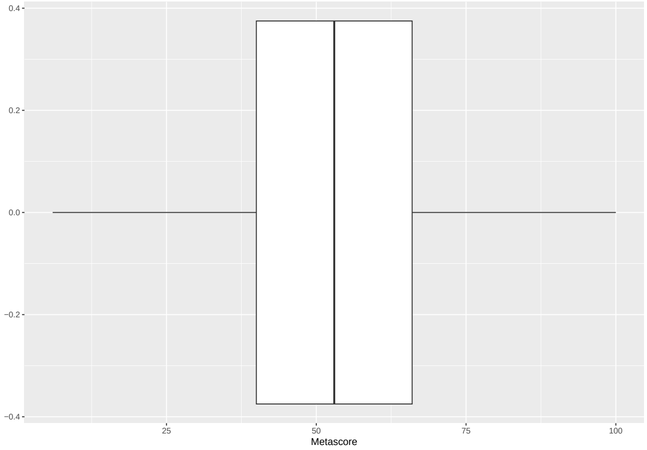
```
ggplot(data = movies, aes(y = Metascore)) + geom_boxplot()
```



Boxplot

Horizontal boxplot

```r
ggplot(data = movies, aes(x = Metascore)) + geom_boxplot()
```



Boxplot

Reading the boxplot

- The width of the box - IQR, is an indicator of the variance
- If the median is in the center and the whiskers have the same length with small to none outliers, then the variable has a bell shape

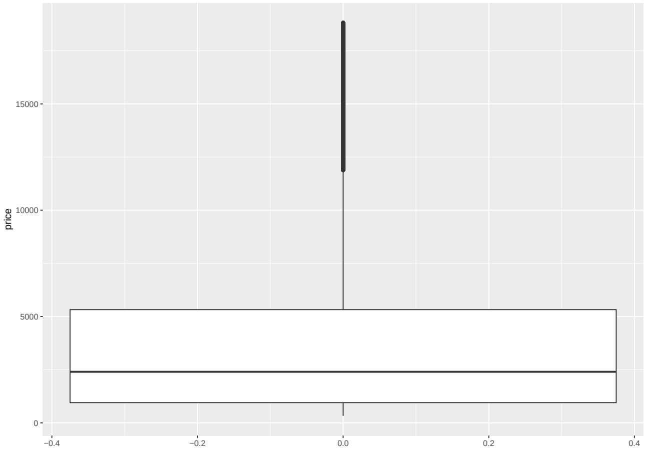
Boxplot

Boxplot of skewed distribution

- Top whisker is longer
- Outliers on the top
- Initial conclusion: We have right skewed distribution

Boxplot

```
ggplot(data = diamonds, aes(y = price)) + geom_boxplot()
```

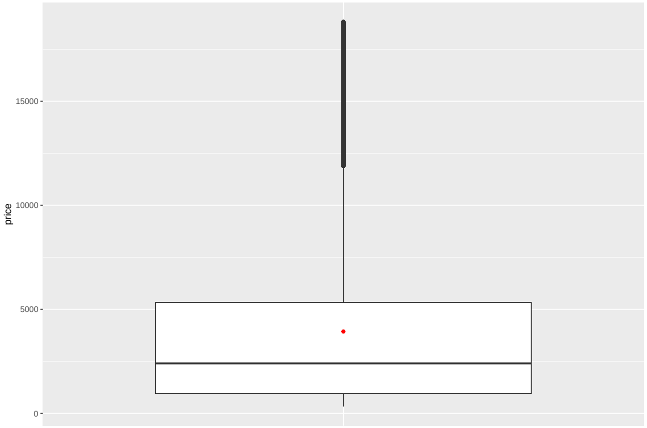


Boxplot

- You can also add mean to the graph (added empty x in aesthetics)
- Mean greater than median - right skewed distribution

Boxplot

```
ggplot(data = diamonds, aes(x = "", y = price)) + geom_boxplot() +
  stat_summary(fun = mean, geom = 'point', color = 'red') + xlab("") +
  theme(axis.ticks.x = element_blank())
```



Comparing distributions

Sometimes will need to compare the distribution of one continuous variable by different categories of a categorical variable

- The height of males and females
- Distribution of waiting time by weekdays
- Number of goals per position in football, etc

We can do this with histograms, boxplots and eCDF

Comparing distributions

- We will use Galton's hereditary data
- for the full analysis refer to Regression Towards Mediocrity in Hereditary Stature

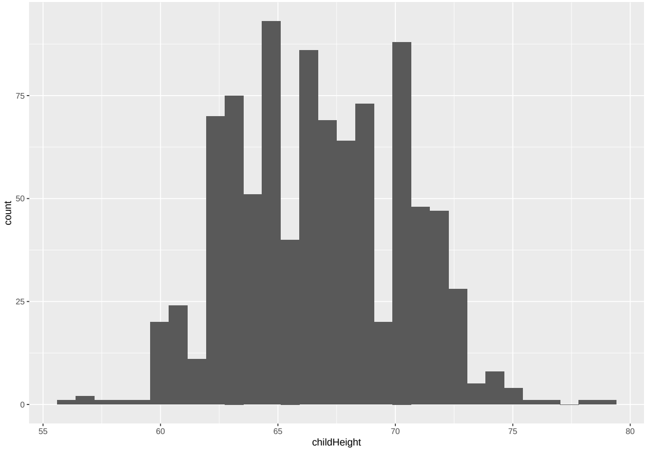
Comparing Distributions

```
data("GaltonFamilies")
str(GaltonFamilies)
## 'data.frame':    934 obs. of  8 variables:
##  $ family         : Factor w/ 205 levels "001","002","003",..: 1 1 1 1 2
##  $ father         : num  78.5 78.5 78.5 78.5 75.5 75.5 75.5 75.5 75 75 .
##  $ mother         : num  67 67 67 67 66.5 66.5 66.5 66.5 64 64 ...
##  $ midparentHeight: num  75.4 75.4 75.4 75.4 73.7 ...
##  $ children       : int  4 4 4 4 4 4 4 4 2 2 ...
##  $ childNum       : int  1 2 3 4 1 2 3 4 1 2 ...
##  $ gender         : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 1 1
##  $ childHeight    : num  73.2 69.2 69 69 73.5 72.5 65.5 65.5 71 68 ...
```

Comparing distributions

Histogram of height

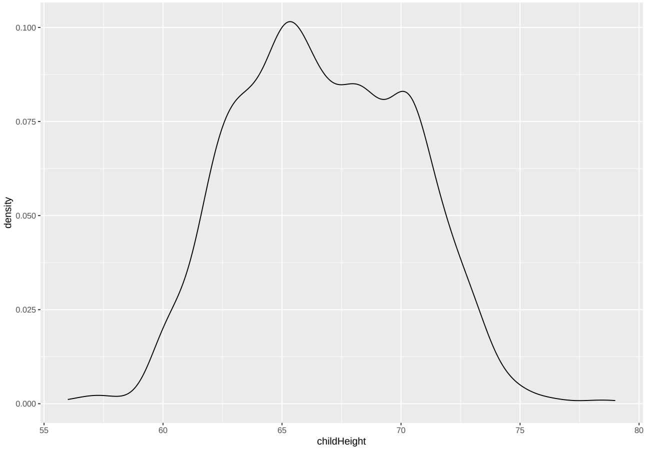
```
ggplot(data=GaltonFamilies, aes(x = childHeight)) + geom_histogram()
```



Comparing distributions

Density estimate

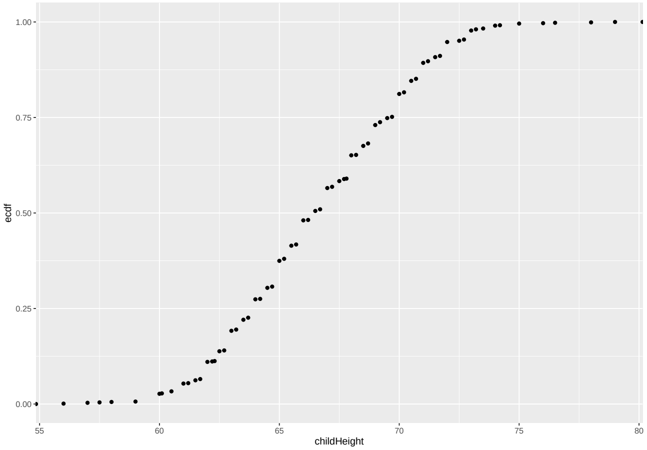
```
ggplot(data = GaltonFamilies, aes(x = childHeight)) + geom_density(bw = 0.7
```



Comparing distributions

eCDF

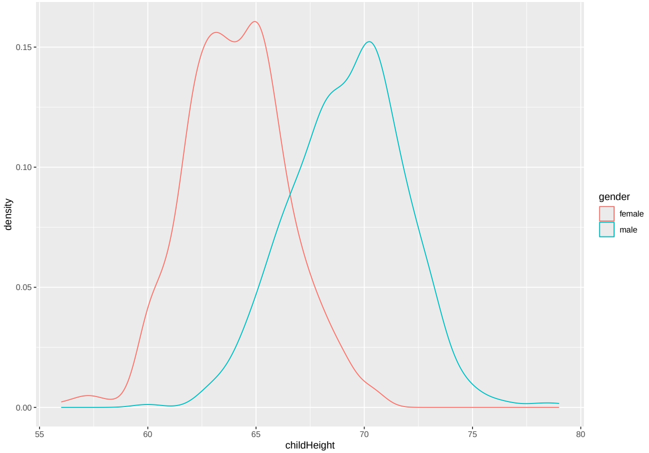
```
ggplot(data = GaltonFamilies, aes(childHeight)) + stat_ecdf(geom = 'point')
```



Comparing distributions

Compare height for male and female children There are two distinct distributions

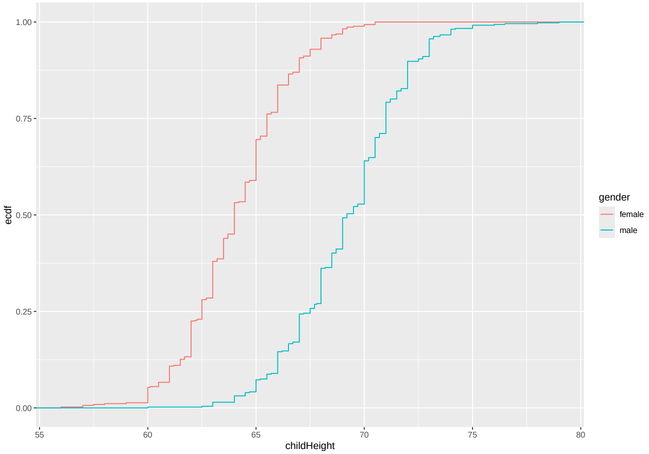
```
ggplot(data = GaltonFamilies, aes(x = childHeight, color = gender)) +
  geom_density()
```



Comparing distributions

Create the eCDF

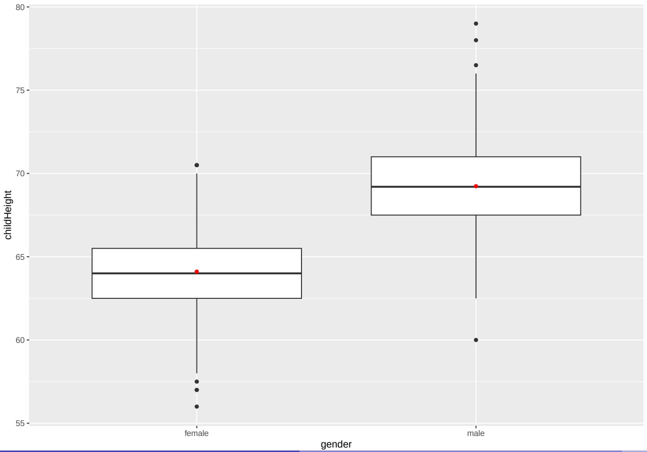
```
ggplot(data = GaltonFamilies, aes(x = childHeight, color = gender)) +
  stat_ecdf()
```



Comparing distributions

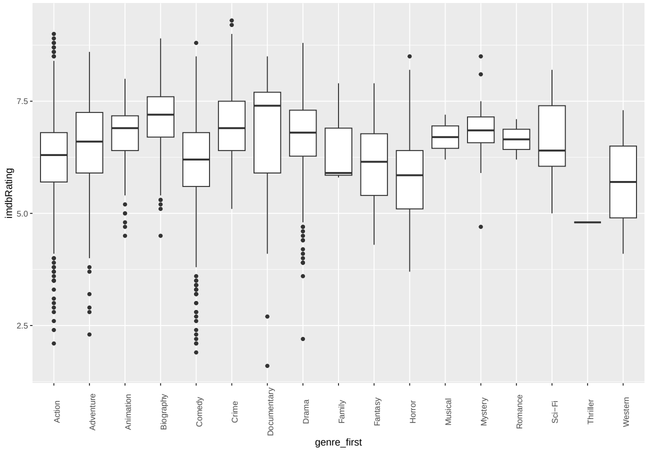
Boxplot

```
ggplot(data = GaltonFamilies, aes(x = gender, y = childHeight)) +
  geom_boxplot() +
  stat_summary(fun.y = mean, geom = 'point', color = 'red')
```



Comparing distributions

```
ggplot(movies, aes(x = genre_first, y = imdbRating)) + geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90))
```



Section 3

Testing for distribution

Probability plots

- The probability plot is a graphical technique for assessing whether or not a variable follows a given distribution.
- The data is plotted against a theoretical distribution in such a way that the points should form approximately a straight line.
- Departures from this straight line indicate departures from the specified distribution.

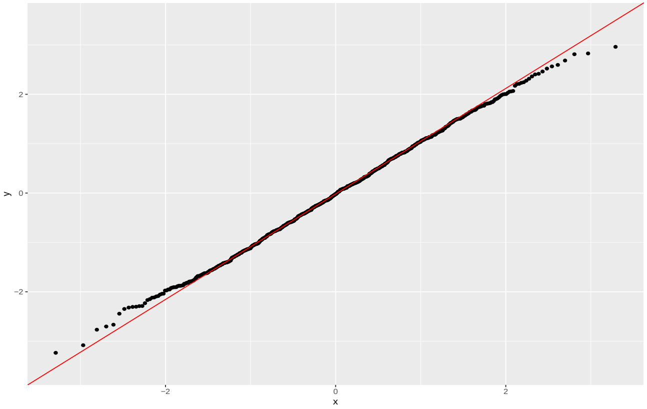
q-q plots

- Q-Q Plots (Quantile-Quantile plots) are plots of quantiles of two variables plotted against each other.
- A quantile is a fraction where certain values fall below that value.
- The purpose of q-q plots is to find out if two sets of data come from the same distribution.
- If we have standard normal distribution then 45° angle is plotted on the QQ plot; if the two data sets come from the same distribution, the points will fall on that reference line, thus quantiles of theoretical and sample distributions are the same.
- The greater the departure from the reference line, the greater the evidence for the conclusion that the two data sets have come from populations with different distributions.
- If the data does not follow standard normal distribution, then the reference line is formed with intercept = mean and slope = standard deviation

q-q plots

We have 45° line

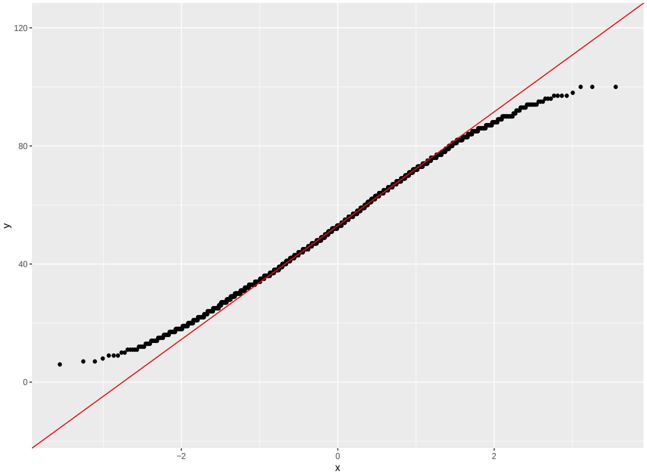
```
x <- rnorm(1000)
ggplot(mapping = aes(sample = x)) + geom_qq() +
  geom_qq_line(color = 'red')
```



q-q plots

q-q plot for Metascore

```
ggplot(movies, aes(sample = Metascore)) + geom_qq() +
  geom_qq_line(color = 'red')
```



q-q plots

Pay Attention

- We have corresponding Z scores for theoretical values on x axis
- The quantiles from the sample in the original scale
- Line is not 45° any more, but is rather estimated

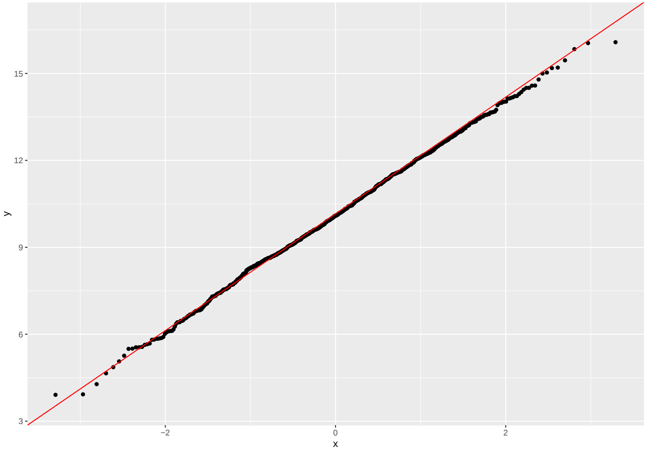
q-q plots

Lets see what happens when you define the distribution parameters

```
x <- rnorm(1000, mean=10, sd = 2)
p1 <- ggplot(mapping =  aes(sample = x)) + geom_qq() +
  geom_qq_line(color = 'red')
```

q-q plots

```
p1
```



q-q plots

Get the data used to draw the plot

```
df1 <- ggplot_build(p1)$data[[2]]
df1
##           x        y   slope intercept PANEL group colour linewidth line
## 1 -3.290527  3.52676 2.01425   10.1547     1    -1    red       0.5
## 2  3.290527 16.78265 2.01425   10.1547     1    -1    red       0.5
##   alpha
## 1    NA
## 2    NA
```

q-q plots

Calculate the slope

```
slope <- diff(df1$y)/diff(df1$x)
slope
## [1] 2.01425
```

Intercept

```
df1$y[1] - slope*df1$x[1]
## [1] 10.1547
```

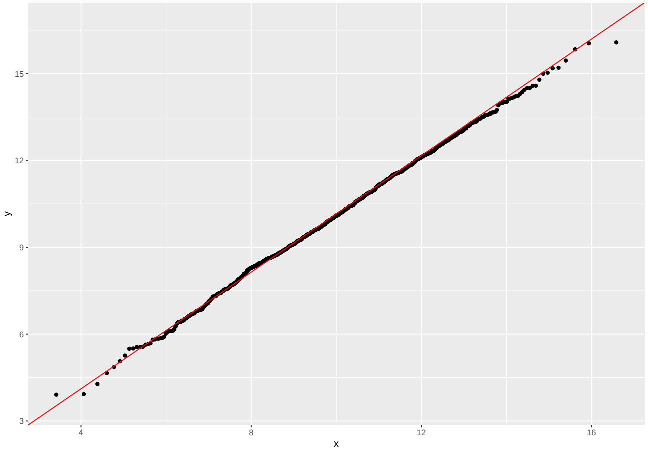
q-q plots

Define the right scale for theoretical distribution

```
p1 <- ggplot(mapping =  aes(sample = x)) +
  geom_qq(dparams = list(mean = 10, sd = 2)) +
  geom_qq_line(color = 'red', dparams = list(mean = 10, sd = 2))
```

q-q plots

```
p1
```



q-q plots

Calculate the slope

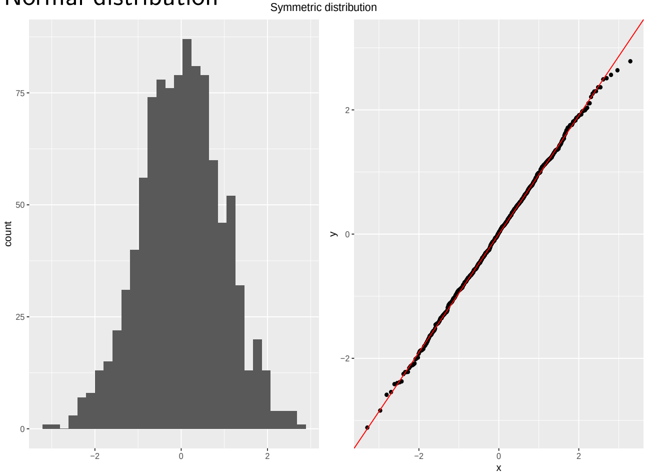
```
df1 <- ggplot_build(p1)$data[[2]]

slope <- diff(df1$y)/diff(df1$x)
slope
## [1] 1.007125
```

Intercept

```
df1$y[1] - slope*df1$x[1]
## [1] 0.08345323
```

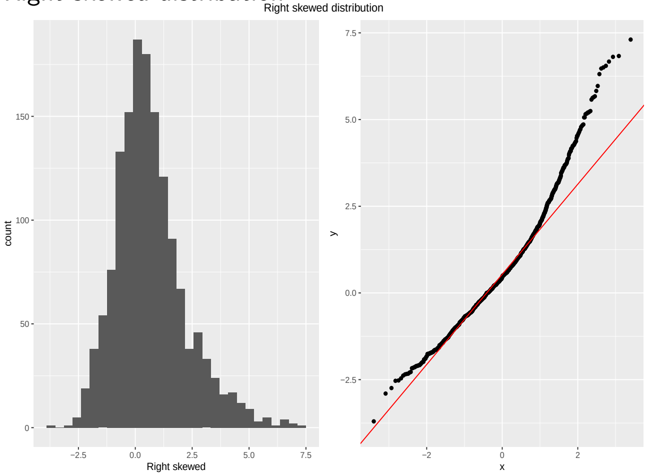
q-q plots

Normal distribution



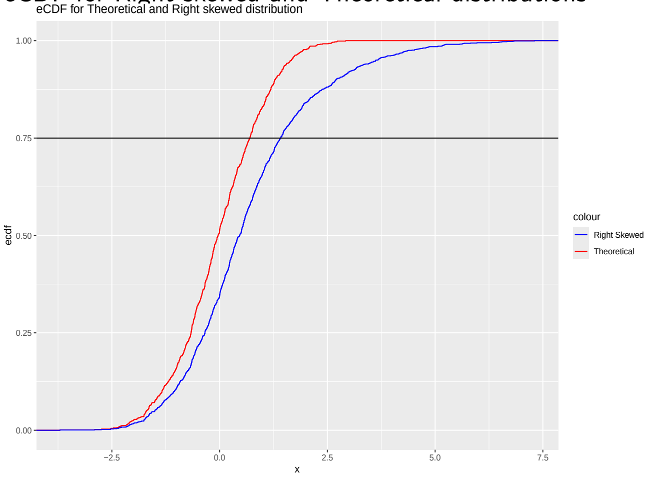
q-q plots

Right skewed distribution



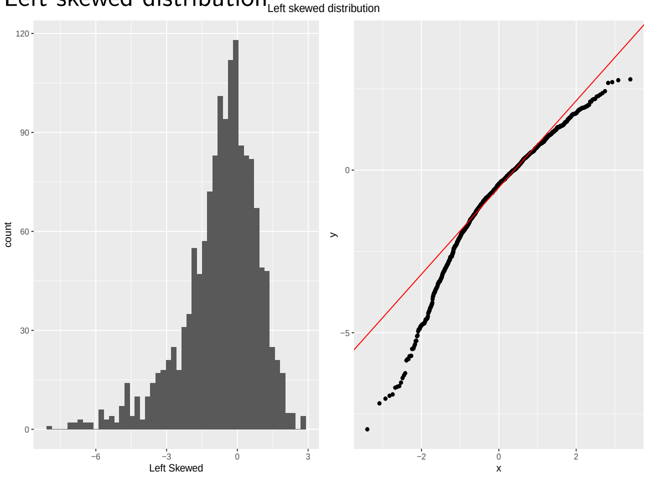
q-q plots

eCDF for Right skewed and Theoretical distributions



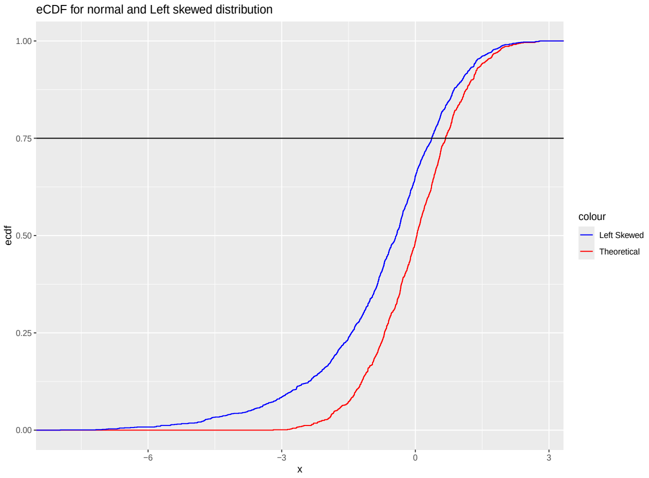
q-q plots

Left skewed distribution



q-q plots

eCDF for left skewed distribution



q-q plots

- Right skewed distribution: The points' upward trend shows that the sample quantiles are much greater than the theoretical quantiles.
- Left skewed distribution: The sample quantiles are going to be much lower than the theoretical quantiles.

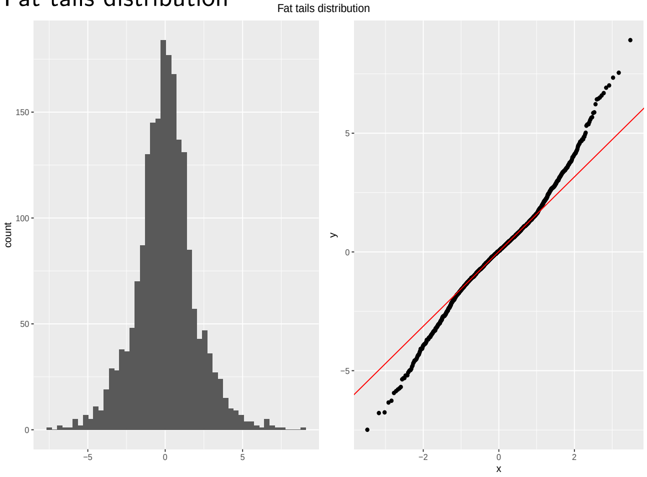
q-q plots

Heavy tail (Fat tail)

- Fat tail means that compared to the normal distribution there is more data located at the extremes of the distribution and less data in the center of the distribution.
- In terms of quantiles this means that the first quantile is much less than the first theoretical quantile and the last quantile is greater than the last theoretical quantile

q-q plots

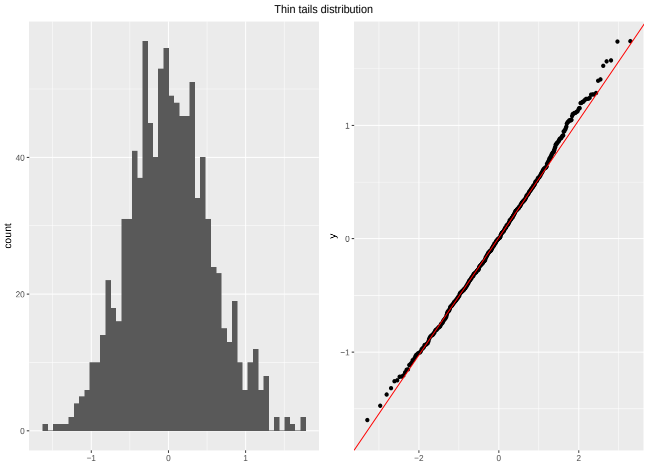
Fat tails distribution



q-q plots

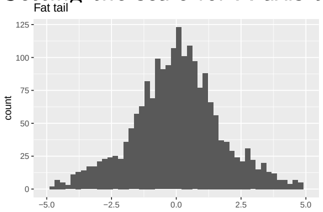
Thin tails distribution

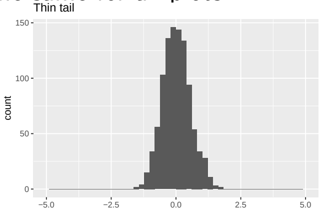
With thin tails distribution you have less data in the tails than it should be compared to the normal distribution

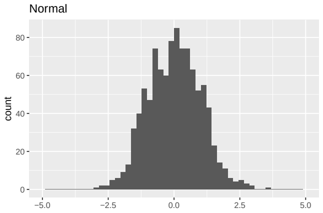


q-q plots

Setting the scale for X axis the same for all plots







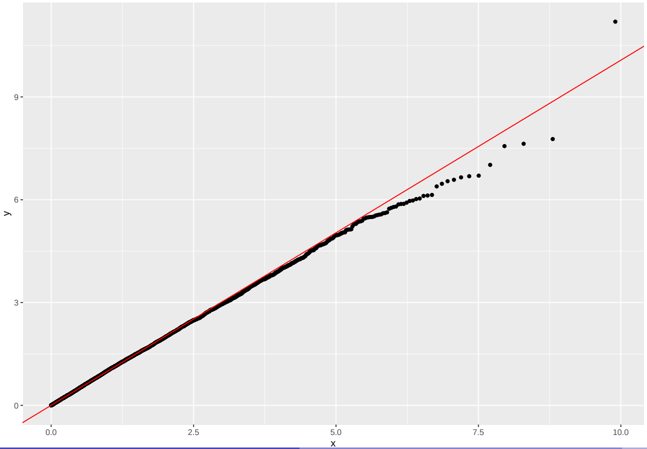
q-q plots

- By default the Q-Q plot is constructed with normal distribution as a theoretical distribution
- However you can use any distribution that is available in R
- Or provide your own distribution with parameters

q-q plots

q-q plot for exponential distribution

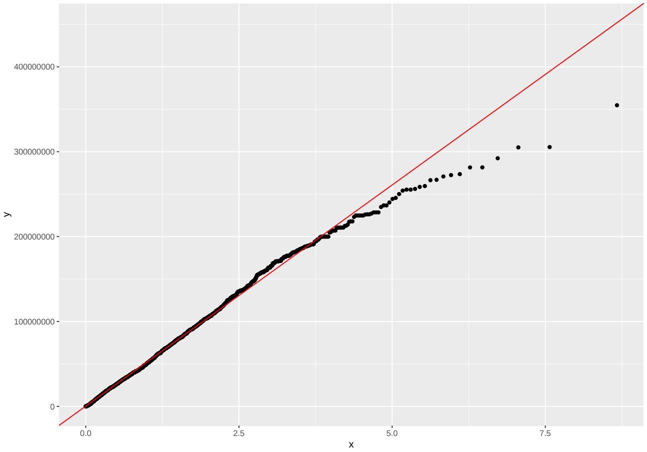
```
x_exp <- rexp(10000)
ggplot(mapping = aes(sample = x_exp)) + geom_qq(distribution = stats::qexp)
  geom_qq_line(color = 'red', distribution =stats::qexp)
```



q-q plots

Budget

```r
ggplot(movies, aes(sample = budget_adjusted)) +
  geom_qq(distribution = stats::qexp) +
  geom_qq_line(color = 'red', distribution = stats::qexp)
```


400000000
300000000
200000000
100000000
0
y
0.0
2.5
5.0
7.5
x