Descriptive Statistics /Описателни статистики/

Population /Популация/
Total set of observations that can be made.

Population sample /Извадка/
A sample is a random selection of population members. It is a small group drawn from the population that has the characteristics of the entire population. It can be with or without replacement.

Example:
Let's say we want to measure the content of vitamin C in the kiwi. We can't take all the kiwis and measure there vitamin C, so we make a **population sample** of 30 kiwis and measure their vitamin C.

Let's denote with: X a **random variable** measuring the vitamin C in 100g of kiwi. Let's make 30 experiments, so our **sample size** n will be 30. And every one of our experiments will consist of taking 100g of kiwi and measuring the vitamin C in it, so this will be our sample $x_1, x_2, ..., x_{30}$.
Let's say our sample looks like this:

```
> vitamin <- c(91.66, 92.34, 92.69, 92.97, 93.40, 92.94, 92.76, 92.39, 93.74, 91.77, 92.09,
+              92.99, 93.13, 92.73, 92.81, 93.39, 93.29, 93.03, 93.13, 92.08, 93.27, 92.82,
+              92.99, 92.60, 93.41, 92.89, 92.76, 92.19, 92.34, 92.68)
```

Order statistics /Вариационен ред/
Order statistics present the sample data in a sorted way. They are denoted by $x_{(1)} \leq x_{(2)} \leq \ldots \leq x(n)$, where $x_{(1)}$ is the smallest observation in the sample and $x_{(n)}$ is the biggest observation in the sample.

```
> sort(vitamin)
 [1] 91.66 91.77 92.08 92.09 92.19 92.34 92.34 92.39 92.60 92.68 92.69 92.73
[13] 92.76 92.76 92.81 92.82 92.89 92.94 92.97 92.99 92.99 93.03 93.13 93.13
[25] 93.27 93.29 93.39 93.40 93.41 93.74
```

One more example:

```
> x = c(10, 3, 4, 7, 34, 23); x
[1] 10  3  4  7 34 23
```

sort sorts the vector in ascending order by default

```
> sort(x)
[1]  3  4  7 10 23 34
> sort(x, decreasing = TRUE)
[1] 34 23 10  7  4  3
```

Rank /Ранг/
Ranks are the places of the elements in the order statistics.
$Rank(x_{(i)}) = i$

We can take the ranks of the elements using the rank function, which gives the respective rank of the number present in the vector, where the smallest number receives the rank 1.

```
> rank(x)
[1] 4 1 2 3 6 5
```

When we have equal elements in the vector by default we are taking the average of their ranks. You can change this to take the smallest, biggest or the rank by occurrence of the equal elements using ties.method.

```
> y = c(x, 10)
> rank(y)
[1] 4.5 1.0 2.0 3.0 7.0 6.0 4.5
> y = c(x, 10, 10)
> rank(y)
[1] 5 1 2 3 8 7 5 5
> rank(y, ties.method = "average")
[1] 5 1 2 3 8 7 5 5
> rank(y, ties.method = "min")
[1] 4 1 2 3 8 7 4 4
> rank(y, ties.method = "max")
[1] 6 1 2 3 8 7 6 6
> rank(y, ties.method = "first")
[1] 4 1 2 3 8 7 5 6
```

Order
order returns the indices of the vector in a sorted order

```
> order(x)
[1] 2 3 4 1 6 5
```

This means that if you take the elements of the vector in this order
$x[2] = 3,\ x[3] = 4,\ x[4] = 7,\ x[1] = 10,\ x[6] = 23,\ x[5] = 34$
you will receive the sorted array.

```
> x[order(x)]
[1]  3  4  7 10 23 34
```

Minimal observation
min returns the smallest observation $x_{(1)}$

```
> min(vitamin)
[1] 91.66
```

Maximal observation

max returns the largest observation $x_{(n)}$

```
> max(vitamin)
[1] 93.74
```

Sample range /Обхват/

Sample range is the spread (distance) from the biggest to the smallest value in the sample $x_{(n)} - x_{(1)}$. We can make it using directly the formula

```
> max(vitamin) - min(vitamin)
[1] 2.08
```

or we can use function range to give us the smallest and the biggest observation and then to compute their difference with diff

```
> range(vitamin)
[1] 91.66 93.74
> diff(range(vitamin))
[1] 2.08
```

Sample mean /Извадково средно/

Sample mean is the arithmetic average of all of the sample values.

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

```
> mean(vitamin)
[1] 92.776
```

Sample mean estimates the mean of the population.
Another way to calculate the sample mean is by using the formula

```
> sum(vitamin) / length(vitamin)
[1] 92.776
```

Another example when we have highly distinctive observation.

```
> vitamin.outlirs <- c(vitamin, 320)
> mean(vitamin.outlirs)
[1] 100.1058
```

As we see the mean is highly sensitive to distinctive observations.
We can have NA values in the sample. We handle them with na.rm = TRUE

```
> vitamin.missing <- vitamin
> vitamin.missing[c(3, 7, 21)] <- NA
> vitamin.missing
[1] 91.66 92.34    NA 92.97 93.40 92.94    NA 92.39 93.74 91.77 92.09 92.99
[13] 93.13 92.73 92.81 93.39 93.29 93.03 93.13 92.08    NA 92.82 92.99 92.60
[25] 93.41 92.89 92.76 92.19 92.34 92.68
```

```
> mean(vitamin.missing)
[1] NA
> mean(vitamin.missing, na.rm = TRUE)
[1] 92.76148
> mean(vitamin.missing[-c(3, 7, 21)])
[1] 92.76148
> mean(vitamin.missing[!is.na(vitamin.missing)])
[1] 92.76148
```

or we can calculate it using the formula, but in these case we need to subtract the number of the NA from the sample size.

```
> sum(vitamin.missing)
[1] NA
> sum(vitamin.missing, na.rm = TRUE)
[1] 2504.56
> sum(vitamin.missing, na.rm = TRUE) / (length(vitamin.missing) -
sum(is.na(vitamin.missing)))
[1] 92.76148
```

Trimmed mean /Орязано средно/

It is useful when the data contains many highly distinctive observations.

```
> vitamin.outlirs <- c(vitamin.outlirs, 50, 405, 31)
> vitamin.outlirs
 [1]  91.66  92.34  92.69  92.97  93.40  92.94  92.76  92.39  93.74  91.77
[11]  92.09  92.99  93.13  92.73  92.81  93.39  93.29  93.03  93.13  92.08
[21]  93.27  92.82  92.99  92.60  93.41  92.89  92.76  92.19  92.34  92.68
[31] 320.00  50.00 405.00  31.00
> length(vitamin.outlirs)
[1] 34
> mean(vitamin.outlirs)
[1] 105.5671
> mean(vitamin.outlirs, trim = 1/17)
[1] 92.776
> vitamin.trim <- vitamin.outlirs[c(-31, -32, -33, -34)]; vitamin.trim
 [1] 91.66 92.34 92.69 92.97 93.40 92.94 92.76 92.39 93.74 91.77 92.09 92.99
[13] 93.13 92.73 92.81 93.39 93.29 93.03 93.13 92.08 93.27 92.82 92.99 92.60
[25] 93.41 92.89 92.76 92.19 92.34 92.68
> mean(vitamin.trim)
[1] 92.776
```

In the above example we trim off $\frac{1}{17}$ (or 2 observations) from the top and the bottom and then take the average.

```
> length(vitamin.outlirs) / 18
[1] 1.888889
> mean(vitamin.outlirs, trim = 1/18)
[1] 98.54
> vitamin.trim.2 <- vitamin.trim[c( -33, -34)]; vitamin.trim.2
 [1] 91.66 92.34 92.69 92.97 93.40 92.94 92.76 92.39 93.74 91.77 92.09 92.99
```

```
[13] 93.13 92.73 92.81 93.39 93.29 93.03 93.13 92.08 93.27 92.82 92.99 92.60
[25] 93.41 92.89 92.76 92.19 92.34 92.68
> mean(vitamin.trim.2)
[1] 92.776
```

If the number of observations to be trimmed is not integer, we trim off $\left\lfloor \frac{1}{18} \right\rfloor$ (or 1 observation) from the top and the bottom and then take the average.
As we trim more and more, the value of the mean gets closer to the median which is when trim = 1/2.

Sample median /Извадкова медиана/

$$Me(X) = \frac{1}{2}\left(x_{(\lfloor (n+1)/2 \rfloor)} + x_{(\lceil (n+1)/2 \rceil)}\right)$$

Sample median is the number from which half of the observations are less than or equal to the median and half of the observations are bigger than or equal to the median. There for it is called also empirical 0.5-quantile or Q_2.

```
> median(vitamin)
[1] 92.815
> median(vitamin.outlirs)
[1] 92.815
```

As we can see the median is not sensitive to highly distinctive data.
Another way to calculate the sample median is to sort the elements and then to use the above formula

```
> vitamin.sort = sort(vitamin)
> n = length(vitamin)
> 1/2 * (vitamin.sort[floor((n + 1) / 2)] + vitamin.sort[ceiling((n + 1) / 2)])
[1] 92.815
```

We can handle NA in the data with na.rm = TRUE

```
> median(vitamin.missing)
[1] NA
> median(vitamin.missing, na.rm = TRUE)
[1] 92.82
```

Or

```
> t = vitamin.missing[!is.na(vitamin.missing)]
> t = sort(t)
> n = length(t)
> 1/2 * (t[floor((n + 1) / 2)] + t[ceiling((n + 1) / 2)])
[1] 92.82
```

Sample mode /Извадкова мода/
Sample mode is the most probable value in the sample for a discrete data or the place where the histogram has a maxima for a continuous data.

```
> x = c(10, 12, 10, 4, 7, 10, 0, 3, 10, 9, 7, 3, 5, 0, 12, 5, 7, 3, 7, 5, 10)
```

For a **discrete data** we can take the **frequency table /Честотна таблица/** with the function table. It presents how many times has an observation occurred in the sample.

```
> t = table(x); t
x
 0  3  4  5  7  9 10 12
 2  3  1  3  4  1  5  2
> names(t[which.max(t)])
[1] "10"
```

We can also sort the table, by the frequencies of the observations, so it is more easy to see the mode

```
> sort(table(x), decreasing = TRUE)
x
10  7  3  5  0 12  4  9
 5  4  3  3  2  2  1  1
```

The mode is not obligatory unique.
For a **continuous data**

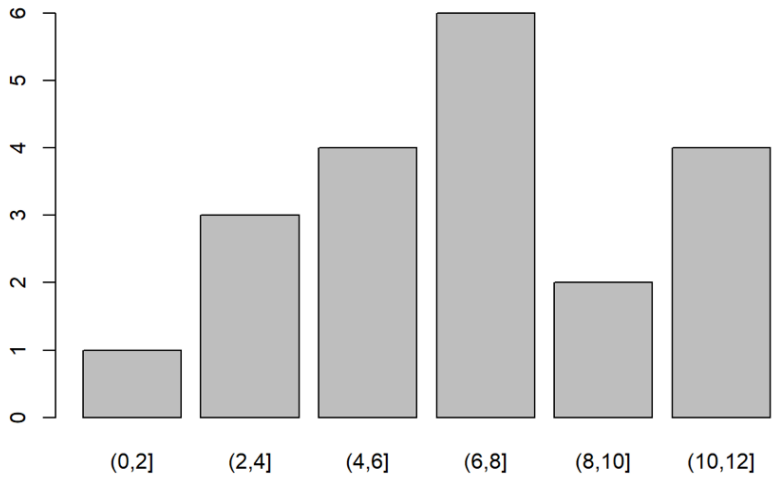
```
> x = c(10.5, 6.2, 10.4, 4.7, 7.3, 10.9, 0, 3.5, 7.3, 9.6, 7.1, 3.8, 5.3, 0.2, 12, 5.6, 7.8, 3.6, 7.2, 5.1, 8.3)
```

We can slice the data using the cut function in equally long intervals.

```
> x.group = cut(x, breaks = c(0, 2, 4, 6, 8, 10, 12))
> class(x.group)
[1] "factor"
> x.group
[1] (10,12] (6,8]   (10,12] (4,6]   (6,8]   (10,12] <NA>    (2,4]   (6,8]
[10] (8,10]  (6,8]   (2,4]   (4,6]   (0,2]   (10,12] (4,6]   (6,8]   (2,4]
[19] (6,8]   (4,6]   (8,10]
Levels: (0,2] (2,4] (4,6] (6,8] (8,10] (10,12]
```

This function returns a factor variable in which every observation is presented as the interval it is in.
We calculate the **frequency table** and present the results on a plot.

```
> table(x.group)
x.group
  (0,2]   (2,4]   (4,6]   (6,8]  (8,10] (10,12]
      1       3       4       6       2       4
> barplot(table(x.group))
```



One way to estimate the mode of continuous variable is

$$Mo = L + \frac{f_{Mo} - f_{Mo-1}}{2f_{Mo} - (f_{Mo-1} + f_{Mo+1})} w$$

where:

- L is the left end of the interval with max frequency
- f_{Mo-1} is the frequency of the interval before the interval with max
- f_{Mo} is the frequency of the interval with max frequency
- f_{Mo+1} is the frequency of the interval after the interval with max frequency
- w length of the interval with max frequency

In our example:

$$L = 6, f_{Mo-1} = 4, f_{Mo} = 6, f_{Mo+1} = 2, w = 8 - 6 = 2,$$

$$Mo = 6 + \frac{6-4}{2*6-(4+2)} * 2 = 6 + \frac{2}{12-6} * 2 = 6 + \frac{2}{3} \approx 6.67$$

There are a lot of options presented on how to estimate the mode in modeest package with the most likely value function mlv.

Sample quantile /Извадкови квантили/

P-quantile is the number from which $p * 100\,\%$ of the observations are less than or equal to the p-quantile and $(1-p) * 100\,\%$ of the observations are bigger than or equal to the p-quantile.

```
> p = 0.35
> quantile(vitamin, p)
   35%
92.696
```

First quartile /Първи квартил/

The **first quartile** Q is the 0.25-quantile.

```
> p = 0.25
> quantile(vitamin, p)
    25%
```

```
92.4425
```

Third quartile /Трети квартил/

The third quartile Q_3 is the 0.75-quantile.

```
> p = 0.75
> quantile(vitamin, p)
  75%
93.105
```

IQR /Между квантилен размах/

The inter quartile range (IQR) is the difference between the 3rd and 1st quartile $IQR = Q_3 - Q_1$.

```
> IQR(vitamin)
[1] 0.6625
> quantile(vitamin, 0.75) - quantile(vitamin, 0.25)
  75%
0.6625
```

Lower hinge

The lower hinge H_L is the median of the first half of the ordered observations.

```
> fivenum(vitamin)[2]
[1] 92.39
```

Upper hinge

The upper hinge H_U is the median of the second half of the ordered observations.

```
> fivenum(vitamin)[4]
[1] 93.13
```

Standard deviation /Стандартно отклонение/

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Sample standard deviation measures the deviation of the observations around their average.

```
> sd(vitamin)
[1] 0.4992588
> sd(vitamin.outlirs)
[1] 67.2411
```

Sample variance /Извадкова дисперсия/

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Sample variance is the second power of the standard deviation.

```
> var(vitamin)
[1] 0.2492593
> var(vitamin.outlirs)
[1] 4521.366
```

Differences

```
> x = c(10, 3, 4, 7, 34, 23); x
[1] 10  3  4  7 34 23
```

diff returns the first differences

```
> diff(x)
[1]  -7   1   3  27 -11
```

with lag = 2 returns the second differences

```
> diff(x, lag = 2)
[1] -6  4 30 16
```

Cummulative sums /Кумулативни суми/
cumsum returns the cumulative sums

```
> cumsum(x)
[1] 10 13 17 24 58 81
```

Cummulative maximum /Кумулативни максимуми/
cummax returns the cumulative maximum

```
> cummax(x)
[1] 10 10 10 10 34 34
```

Cummulative minimum /Кумулативни минимуми/
cummin returns the cumulative minimum

```
> cummin(x)
[1] 10  3  3  3  3  3
```

Sources
[1] Monika Petkova's notes on R programming language @ FMI, Sofia University