```
dataRaw = randi([5 10],1,400);
figure(1); rose(dataRaw,8)    % generate rose plot
title('angular histogram with 8 sectors')
[data,cent] = hist(dataRaw,unique(dataRaw))  % count the events

figure(2); stairs(cent,data)  % generate stairstep plot
xlabel('value'); ylabel('number of events');
```

4 Data Reduction Commands

In Table 3 find a few *Octave*/`MATLAB` commands to extract information from data sets.

`mean()`	mean value of a data set
`median()`	median value of a data set
`std()`	standard deviation of a data set
`var()`	variance of a data set
`quantile()`	determine arbitrary quantiles
`mode()`	determine the most frequently occurring value
`LinearRegression()`	perform a linear regression
`regress()`	perform a linear regression
`polyfit()`	perform a linear regression for a polynomial
`ols()`	perform an ordinary linear regression
`gls()`	perform an generalized linear regression
`cov()`	covariance matrix
`corr()`	linear correlation matrix
`corrcoef()`	correlation coefficient

Table 3: Commands for data reduction

4.1 Comands `mean()`, `std()`, `var()`, `median()`, `mode()`

- For a vector $x \in \mathbb{R}^N$ the command `mean(x)` determines the mean value by

$$\text{mean}(x) = \bar{x} = \frac{1}{N}\sum_{j=1}^{n} x_j$$

- For a sorted vector $x \in \mathbb{R}^N$ the command `median(x)` determines the median value by

$$\text{median}(x) = \begin{cases} x_{(N+1)/2} & \text{if } N \text{ is odd} \\ \frac{1}{2}(x_{N/2} + x_{N/2+1}) & \text{if } N \text{ is even} \end{cases}$$

- For a vector $x \in \mathbb{R}^N$ the command `std(x)` determines the standard deviation by the formula

$$\text{std}(x) = \sqrt{\text{var}(x)} = \left(\frac{1}{N-1} \sum_{j=1}^{n} (x_j - \bar{x})^2 \right)^{1/2}$$

By default cov() will normalize by $(N-1)$, but using an option you may divide by N, e.g. $\text{std}(x, 1)$.

- For a vector $x \in \mathbb{R}^N$ the command var(x) determines the variance by the formula

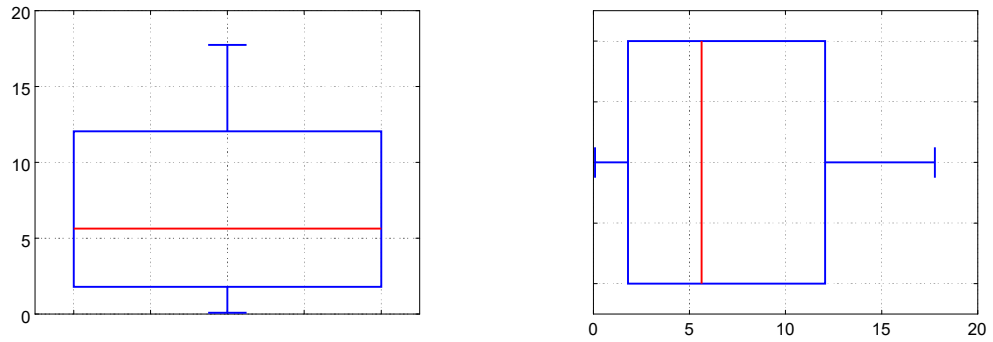
$$\mathrm{var}(x) = (\mathrm{std}(x))^2 = \frac{1}{N-1} \sum_{j=1}^{n} (x_j - \bar{x})^2$$

  By default var() will normalize by $(N-1)$, but using an option you may divide by N, e.g. `var(x,1)`.

- With the command `quantile()` you can compute arbitrary quantiles. Observe that there are different methods to determine the quantiles, leading to different results! Consult the built-in documentation in *Octave* by calling `help quantile` .

- Often quartile (division by four) or percentiles (division by 100) have to be determined. The command `quantile()` with a proper set of parameters does the job.

- With the command `boxplot()` you can generate a plot showing the median, the first and third quartile as a box and the extrem values. Observe that there are different ways to compute the positions of the quartiles and some implementations of `boxplot()` detect and mark outliers. Consult the documentation in *Octave*. Boxplots can also be displayed horizontally or vertically, as shown in Figure 7.



(a) vertical (b) horizontal

Figure 7: Boxplots

```
N = 10; % number of data points
data1 = 20*rand(N,1);
Mean = mean(data1)
Median = median(data1)
StdDev = std(data1)  % uses a division by (N-1)
Variance = StdDev^2
Variance2 = mean((data1-mean(data1)).^2)  % uses a division by N
Variance3 = sum((data1-mean(data1)).^2)/(N-1)

figure(1)
Quartile1 = boxplot(data1)'    % in Matlab slightly different
set(gca,'XTickLabel',{' '})    % remove labels on x axis
c_axis = axis(); axis([0.5 1.5 c_axis(3:4)])

figure(2)
```

```
%boxplot(data1,'orientation','horizontal')  % Matlab
boxplot(data1,0,'+',0)  % Octave
set(gca,'YTickLabel',{' '}) % remove labels on y axis
c_axis = axis(); axis([c_axis(1:2),0.5 1.5])

Quartile2 = quantile(data1,[0 0.25 0.5 0.75 1])'
Quantile10 = quantile(data1,0:0.1:1)'

data2 = randi(10,[100,1]);
ModalValue = mode(data2)  % determine the value occuring most often
```

It is possible to put multiple boxplots in one graph, and label the axis according to the data. In Figure 8 the weekdays are used to label the horizontal axis.

```
% generate the random data, with some structure
N = 20;  data = zeros(N,7);
for i = 1:7
  data(:,i) = 3+4*sin(i/4)+randn(N,1);
end%for

boxplot(data);
set(gca(),'xtick',[1:7],'xticklabel', {'Mo', 'Tu', 'We', 'Th','Fr','Sa','Su'});
```

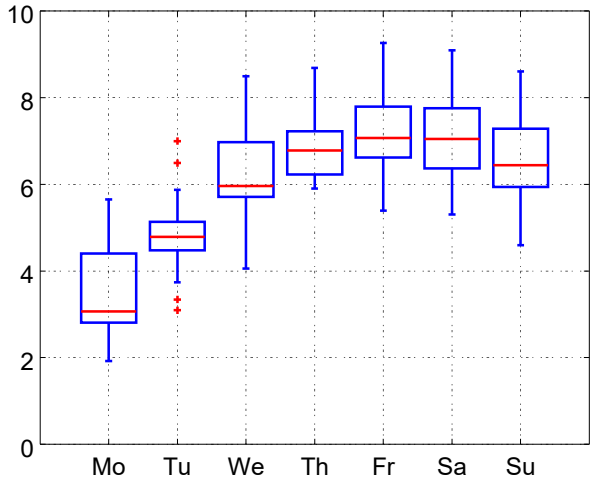


Figure 8: Multiple boxplots in one figure

4.2 For vectors: `cov()`, `corr()`, `corrcoef()`

For covariance and correlation coefficients first step is always to subtract the mean value of the components from a vector, i.e. then the new mean value is zero.

- Covariance of two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$

$$\mathrm{cov}(x,y) \quad = \quad \frac{1}{n-1} \sum_{j=1}^{n} (x_j - \mathrm{mean}(x)) \cdot (y_j - \mathrm{mean}(y))$$

$$= \frac{1}{n-1} \sum_{j=1}^{n} (x_j\, y_j - \text{mean}(x)\ \text{mean}(y))$$

By default `cov()` will normalize by $(n-1)$, but using an option you may divide by n, e.g. `cov(x,y,1)`. If $x = y$ we obtain

$$\text{cov}(x,x) = \frac{1}{n-1} \sum_{j=1}^{n} (x_j - \text{mean}(x))^2 = \text{var}(x)$$

Use `cov()` to determine the covariance.

- The correlation coefficient of two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$

$$\begin{aligned} \text{corr}(x,y) &= \frac{\text{cov}(x,y)}{\text{std}(x) \cdot \text{std}(y)} \\ &= \frac{\langle (\vec{x} - \text{mean}(\vec{x}))\,,\, (\vec{y} - \text{mean}(\vec{y})) \rangle}{\|\vec{x} - \text{mean}(\vec{x})\|\, \|\vec{y} - \text{mean}(\vec{y})\|} \\ &= \frac{\sum_{j=1}^{n} (x - \text{mean}(x))_j \cdot (y - \text{mean}(y))_j}{(\sum_{j=1}^{n} (x_j - \text{mean}(x))^2)^{1/2} (\sum_{j=1}^{n} (y_j - \text{mean}(y))^2)^{1/2}} \end{aligned}$$

Observe that if the average value of the components of both vectors are zero, then there is a geometric interpretation of the correlation coefficient as the angle between the two vectors.

$$\text{corr}(\vec{x}, \vec{y}) = \frac{\langle \vec{x}\,,\, \vec{y} \rangle}{\|\vec{x}\|\, \|\vec{y}\|} = \cos(\alpha)$$

This correlation coefficient can also be computed with by `corrcoef()`, which is currently not available in *Octave*.

```
x = linspace(0,pi/2,20)'; y = sin(x);  % generate some artificial data
MeanValues  = [mean(x), mean(y)]
Variances   = [var(x), var(y)]
StandardDev = [std(x), std(y)]
Covariance  = cov(x,y)
Correlation = corr(x,y)
-->
MeanValues  =   0.78540   0.62944
Variances   =   0.23922   0.10926
StandardDev =   0.48910   0.33055
Covariance  =   0.15794
Correlation =   0.97688
```

4.3 For matrices: `mean()`, `std()`, `var()`, `median()`, `cov()`, `corr()`, `corrcoef()`

Most of the above commands can be applied to matrices. Use each column as one data vector. Assume that $\mathbf{M} \in \mathbb{R}^{N \times m}$ is a matrix of m column vectors with N values in each column.

- `mean(M)` compute the average of each column. The result is a row vector with m components.
- `std(M)` compute the standard deviation of each column. The result is a row vector with m components.
- `var(M)` compute the variance of each column. The result is a row vector with m components.

- `median(M)` compute the median value of each column. The result is a row vector with m components.

To describe the effect of `cov()` and `corr()` the first step is to assure that the average of each column equals zero.

```
Mm = M - ones(N,1)*mean(M);
```

Observe that this operation does not change the variance of the column vectors.

- `cov(M)` determines the $m \times m$ covariance matrix

$$\mathrm{cov}(M) = \frac{1}{N-1}\,\mathbf{Mm}' \cdot \mathbf{Mm}$$

- The $m \times m$ correlation matrix contains all correlation coefficients of the m column vectors in the matrix $\mathbf{M}$. To compute this, first make sure that the norm of each column vector equals 1, i.e. the variance of the column vectors is normalized to 1 .

  ```
  Mm1 = Mm ./ diag(sqrt(sum(Mm.^2)));
  ```

  Determine the $m \times m$ (auto)correlation matrix `corr(M)` by

$$\mathrm{corr}(M) = \mathbf{Mm1}' \cdot \mathbf{Mm1}$$

  Observe that the diagonal entries are 1, since the each column vector correlates perfectly with itself.

5 Performing Linear Regression

5.1 Using `LinearRegression()`

The command `LinearRegression()` was written by the author of these notes.

- For *Octave* the command is contained in the optimization package `optim`. You may download the code at LinearRegression.m.
- The command can be used with `MATLAB` too, but you need a Matlab version.

With this command you can apply the method of least square to fit a curve to a given set of data points. The curve does **not** have to be a linear function, but a linear combination of (almost arbitrary) functions. In the code below a straight line is adapted to some points on a curve $y = \sin(x)$. Thus we try to find the optimal values for a and m such that

$$\chi^2 = \sum_j (a + m\,x_j - y_j)^2 \quad \text{is minimal}$$

The code to perform the this linear regression is given by