Statistics with MATLAB/*Octave*

Andreas Stahel
Bern University of Applied Sciences

Version of 20th December 2021

There is no such thing as "*the perfect document*" and improvements are always possible. I welcome feedback and constructive criticism. Please let me know if you use/like/dislike these notes. Please send your observations and remarks to Andreas.Stahel@bfh.ch .

Contents

List of Figures

List of Tables

1 Introduction

In this document a few `MATLAB`/*Octave* commands for statistics are listed and elementary sample codes are given. This should help you to get started using *Octave*/`MATLAB` for statistical problems. The notes can obviously **not** replace a regular formation in statistics and probability. Find the current version of this document at web.sha1.bfh.science/StatisticsWithMatlabOctave.pdf . Some of the data used in the notes are available at web.sha1.bfh.science/Data_Files/ .

- The short notes you are looking at right now should serve as a starting point and will **not** replace reading and understanding the built-in documentation of *Octave* and/or `MATLAB`.
- There are many good basic introductions to *Octave* and `MATLAB`. Use you favourite document and the built-in help. The author of these notes maintains a set of lecture notes at web.sha1.bfh.science/Labs/PWF/Documentation/OctaveAtBFH.pdf and most of the codes can be found at web.sha1.bfh.science/Labs/PWF/Codes/ or as a single, compressed file at web.sha1.bfh.science/Labs/PWF/Codes.tgz .
- For users of `MATLAB` is is assumed that the statistics toolbox is available.
- For users of *Octave* is is assumed that the statistics package is available and loaded.
  - Use `pkg list` to display the list of available packages. Packages already loaded are marked by `*` .
  - If the package `statistics` is installed, but not loaded, then type `pkg load statistics` .
  - On Linux system you can use your package manager to install *Octave* packages, e.g. by `apt-get install octave-statistics` .
  - If the package statistics is not installed on a Unix system you can download, compile and install packages on your system. In *Octave* type `pkg install -forge statistics`. It is possible that you have to install the package `io` first by `pkg install -forge io` .
  - On a typical Win* system all packages are usually installed, since it is very difficult to install. Consult the documentation provided with your installation of *Octave*.

2 Commands to Load Data from Files

It is a common task that data is available in a file. Depending on the format of the data there are a few commands that help to load the data into `MATLAB` or *Octave*. They will be used often in these notes for the short sample codes. A list is shown in Table 1. Consult the built-in help to find more information on those commands.

3 Commands to Generate Graphics

In Table 2 find a few *Octave*/`MATLAB` commands to generate pictures used in statistics. Consult the built-in help to learn about the exact syntax.

3.1 Histograms

With the command `hist()` you can generate histograms, as seen in Figure 1.

`load()`	loading data in text format or binary formats
`dlmread()`	loading data in (comma) separated format
`dlmwrite()`	writing data in (comma) separated format
`textread()`	read data in text format
`strread()`	read data from a string
`fopen()`	open a file for reading or writing
`fclose()`	close a file for reading
`fread()`	read fron an open file
`fgetl()`	read one line from a file
`sscanf()`	formated reading from a string
`sprintf()`	formated writing to a string

Table 1: Commands to load data from a file

`hist()`	generate a histogram
`[heights,centers] = hist()`	generate data for a histogram
`histc()`	compute histogram count
`histfit()`	generate histogram and fitted normal density
`bar(centers,heights)`	generate a bar chart
`barh()`	generate a horizontal bar chart
`bar3()`	generate a 3D bar chart
`pie()`	generate a pie chart
`stem()`	generate a 2D stem plot
`stem3()`	generate a 3D stem plot
`rose()`	generate an angular histogram
`stairs()`	generate a stairs plot
`boxplot()`	generate a boxplot

Table 2: Commands to generate statistical graphs

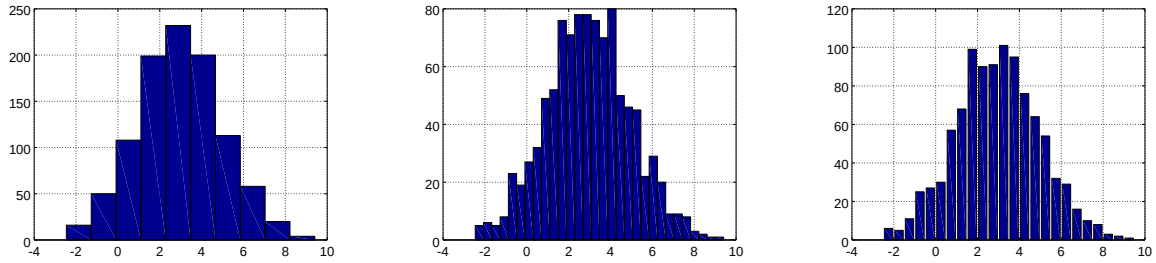
```
data = 3+2*randn(1000,1); % generate the random data
figure(1); hist(data)     % hisogram with default values
figure(2); hist(data,30)  % histogram with 30 classes
```

It is possible to specify the centers of the classes and then compute the number of elements in the class by giving the command `hist()` two return arguments. Then use `bar()` to generate the histogram. The code below chooses classes of width 0.5 between -2.5 and $+9.5$. Thus the first center is at -2.25, the last center at 9.25 and the centers are 0.5 apart.

```
[heights,centers] = hist(data,[-2.25:0.5:9.25]);
figure(3); bar(centers,heights)
```

The number of elements on the above classes can also be computed with the command `heights = histc(data,[-2.5:0.5:9.5])`. Here we specify the limits of the classes.

With a combination of the commands `unique()` and `hist()` one can also count the number of



(a) default values (b) with 30 classes (c) with selected centers

Figure 1: Histograms

entries in a vector.

```
 a = randi(10,100,1) % generate 100 random integres between 1 and 10
[count, elem] = hist(a,unique(a)) % determine the entries (elem) and
                                  % their number (count)
bar(elem,count)  % display the bar chart
```

With the command `histfit()` generate a histogram and the best matching normal distribution as a graph. Find the result of the code below in Figure 2.

```
data = round( 50+20*randn(1,2000));
histfit(data)
xlabel('values'); ylabel('number of events')
```

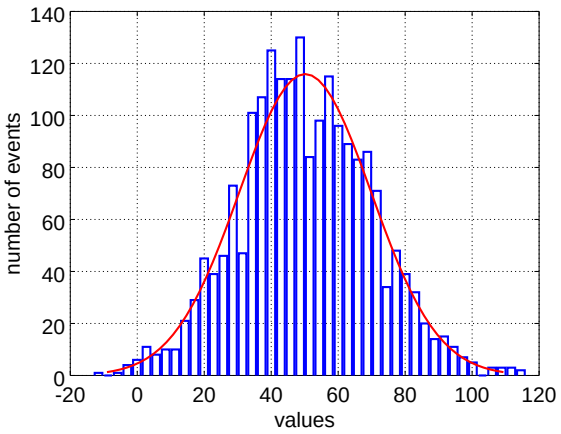


Figure 2: A histogram with matching normal distribution

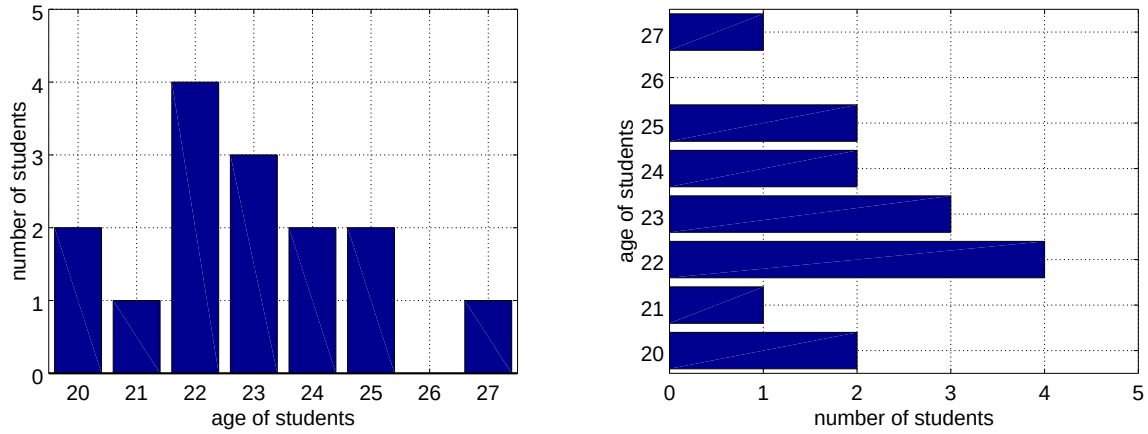
3.2 Bar Diagrams and Pie Charts

Using the commands `bar()` and `barh()` one can generate vertical and horizontal bar charts. The results of the code below is shown in Figure 3.

```
ages = 20:27;     students = [2 1 4 3 2 2 0 1];

figure(1); bar(ages,students)
           axis([19.5 27.5 0 5])
           xlabel('age of students'); ylabel('number of students')

figure(2); barh(ages,students)
           axis([0 5 19.5 27.5])
           ylabel('age of students'); xlabel('number of students')
```

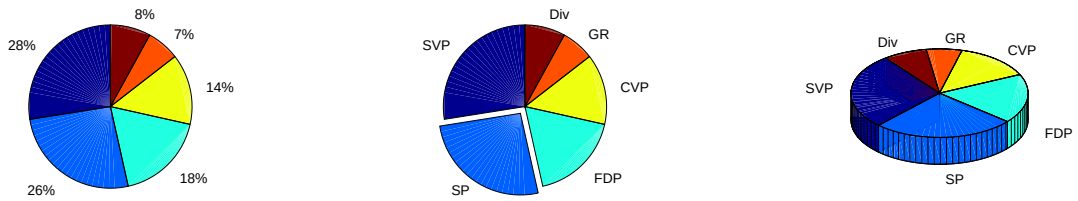


(a) vertical (b) horizontal

Figure 3: Bar diagram

Using the commands `pie()` and `pie3()` one can generate pie charts. With the correct options set labels and some of the slices can be drawn slightly removed from the pie. The result of the code below is shown in Figure 4.

```
strength = [55 52  36 28 13 16];
Labels = {'SVP','SP','FDP','CVP','GR','Div'}
figure(1); pie(strength)
figure(2); pie(strength,[0 1 0 0 0 0],Labels)
figure(3); pie3(strength,Labels)
```



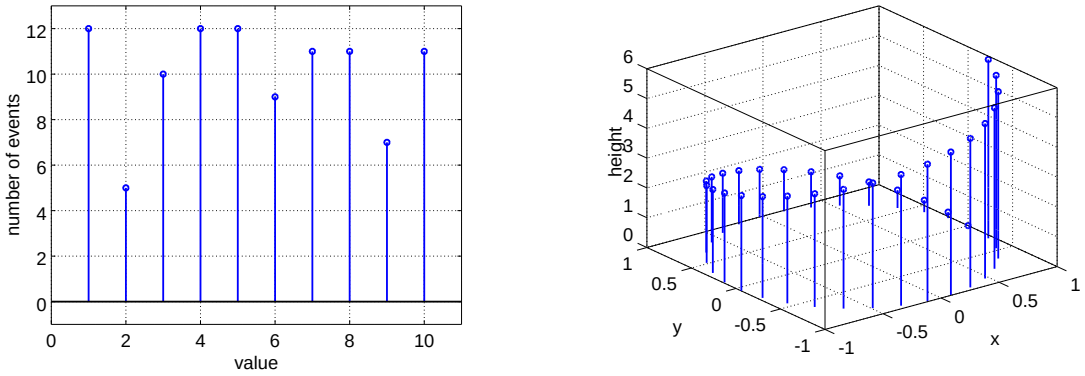
(a) default values (b) with labels (c) 3D

Figure 4: Pie charts

3.3 Plots: `stem()`, `stem3()`, `rose()`, `stairs()`

With a stem plot a vertical line with a small marker at the top can be used to visualize data. The code below first generates a set of random integers, and then uses a combination of `unique()` and `hist()` to determine the frequency (number of occurrences) of those numbers.

```
ii = randi(10,100,1);   % generate 100 random integers between 1 and 10
[anz,cent] = hist(ii,unique(ii))  % count the events
stem(cent,anz)          % generate a 2D stem graph
xlabel('value'); ylabel('number of events'); axis([0 11, -1 max(anz)+1])
-->
anz =   12    5   10   12   12    9   11   11    7   11
cent =   1    2    3    4    5    6    7    8    9   10
```



(a) 2D (b) 3D

Figure 5: Stem plots

With `stem3()` a 3D stem plot can be generated.

```
theta = 0:0.2:6;
stem3 (cos (theta), sin (theta), theta);
xlabel('x'); ylabel('y'); zlabel('height')
```

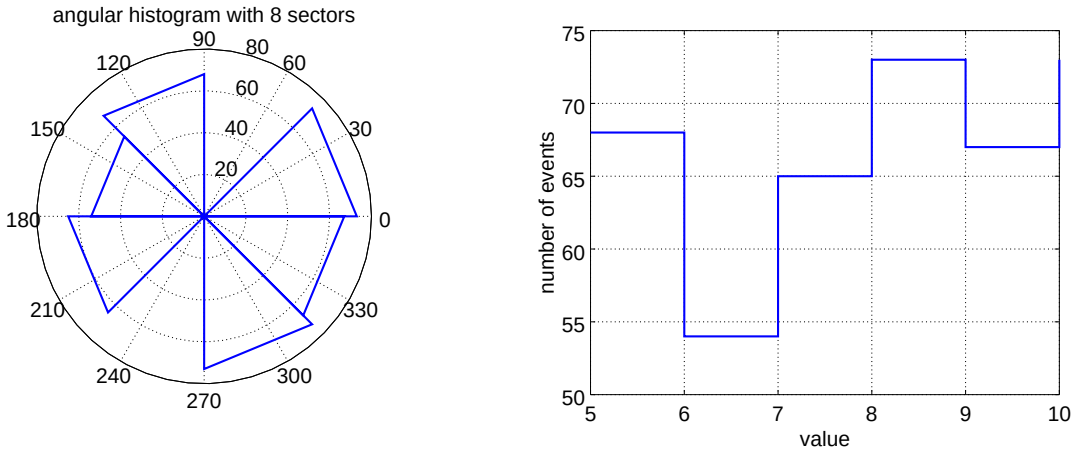
MATLAB/Octave provide commands to generate angular histograms and stairstep plots.

```
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.



(a) rose plot

(b) stairs plot

Figure 6: Rose plot and stairs plot

4.1 Commands `mean()`, `std()`, `var()`, `median()`, `mode()`, `quantile()`, `boxplot()`

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

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

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

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

- For a vector $x \in \mathbb{R}^n$ the command `median(x)` determines the median value. For a sorted vector is is given 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 `mode(x)` determines the most often occurring value in x. The commands `mode(randi(10,20,1))` generate 20 random integer values between 1 and 10, and then the most often generated value.

`mean()`	mean value of a data set
`std()`	standard deviation of a data set
`var()`	variance of a data set
`median()`	median value of a data set
`mode()`	determine the most frequently occurring value
`quantile()`	determine arbitrary quantiles
`boxplot()`	generate a boxplot
`LinearRegression()`	perform a linear regression
`regress()`	perform a linear regression
`lscov()`	generalized least square estimation, with weights
`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

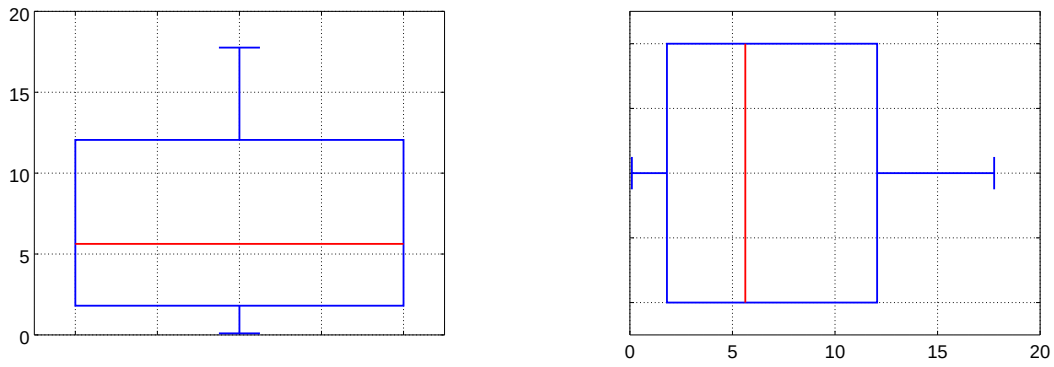
Table 3: Commands for data reduction

- 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. You may also use `prctile()` to determine the values for the quartile (default) or other values for the divisions.
- With the command `boxplot()` you can generate a plot showing the median, the first and third quartile as a box and the extreme values. Observe that there are different ways to compute the positions of the quartiles and some implementations of `boxplot()` detect and mark outliers. By using an optional argument you can select which points are considered as outliers. Consult the documentation in *Octave*. Boxplots can also be displayed horizontally or vertically, as shown in Figure 7.

```
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(data1,0,'+',0)  % Octave
```



(a) vertical (b) horizontal

Figure 7: Boxplots

```
            %boxplot(data1,'orientation','horizontal')  % Matlab
            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 abreviated names of 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'});
```

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 $\vec{x}, \vec{y} \in \mathbb{R}^n$

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

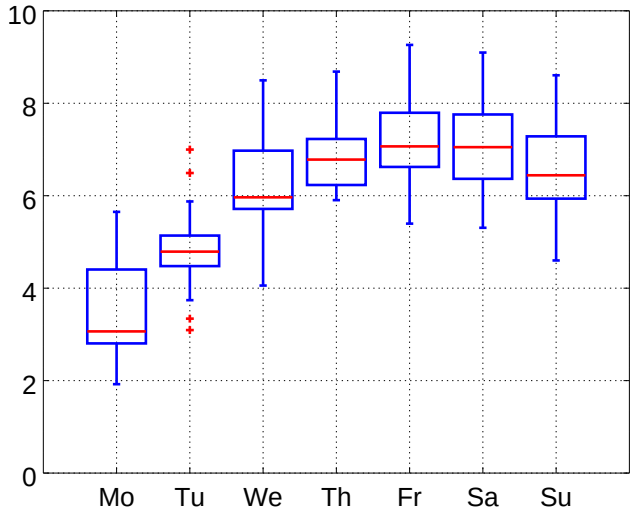


Figure 8: Multiple boxplots in one figure

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

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

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

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

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.

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

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

The method of least squares, linear or nonlinear regression is one of the most often used tools in science and engineering. `MATLAB`/*Octave* provide multiple commands to perform these algorithms.

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

```
% generate the artificial data
x = linspace(0,2,10)'; y = sin(x);

% perform the linear regression, aiming for a straight line
F = [ones(size(x)),x];
[p,e_var,r,p_var] = LinearRegression(F,y);
Parameters_and_StandardDeviation = [p sqrt(p_var)]
estimated_std = sqrt(mean(e_var))
-->
Parameters_and_StandardDeviation =   0.202243   0.091758
                                     0.477863   0.077345
estimated_std =  0.15612
```

The above result implies that the best fitting straight line is given by

$$y = a + m\,x = 0.202243 + 0.477863\,x$$

Assuming that the data is normally distributed one can show that the values of a and m normally distributed. For this example the estimated standard deviation of a is given by 0.09 and the standard deviation of m is 0.08. The standard deviation of the residuals $r_j = a + m\,x_j - y_j$ is estimated by 0.16 . This is visually confirmed by Figure 9(a), generated by the following code.

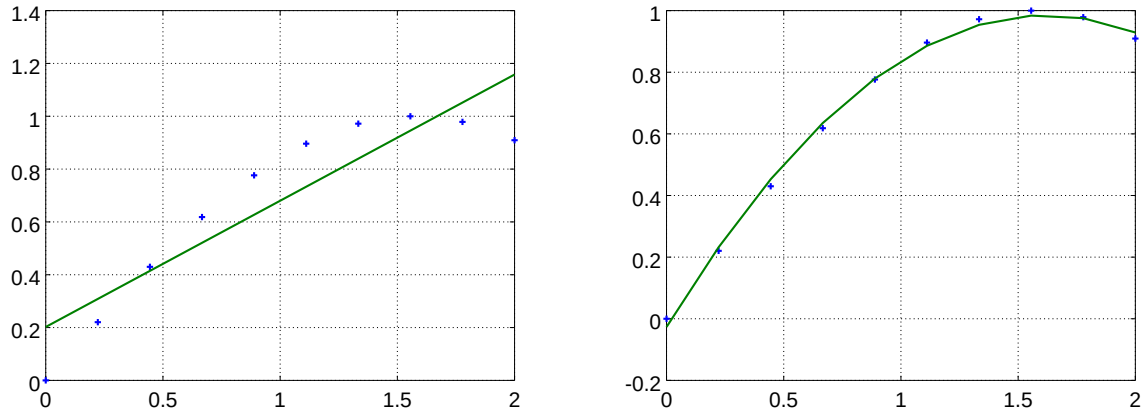
```
y_reg = F*p;
plot(x,y,'+', x , y_reg)
```

With linear regression one may fit different curves to the given data. The code below generates the best matching parabola and the resulting Figure 9(b).

```
% perform the linear regression, aiming for a parabola
F = [ones(size(x)),x, x.^2];
[p,e_var,r,p_var] = LinearRegression(F,y);

Parameters_and_StandardDeviation = [p sqrt(p_var)]
estimated_std = sqrt(mean(e_var))

y_reg = F*p;
```



(a) straight line regression

(b) parabola regression

Figure 9: Results of two linear regressions

```
plot(x,y,'+', x , y_reg)
-->
Parameters_and_StandardDeviation =  -0.026717   0.015619
                                     1.250604   0.036370
                                    -0.386371   0.017506
estimated_std =  0.019865
```

Since the parabola is a better match for the points on the curve $y = \sin(x)$ we find smaller estimates for the standard deviations of the parameters and residuals.

It is possible perform linear regression with functions of multiple variables. The function

$$z = p_1 \cdot 1 + p_2 \cdot x + p_3 \cdot y$$

describes a plane in 3D space. A surface of this type is fit to a set of given points (x_j, y_j, z_j) by the code below, resulting in Figure 10. The columns of the matrix $\mathbf{F}$ have to contain the values of the basis functions 1, x and y at the given data points.

```
N = 100; x = 2*rand(N,1); y = 3*rand(N,1); z =  2 + 2*x- 1.5*y + 0.5*randn(N,1);

F = [ones(size(x)), x , y];
p = LinearRegression(F,z)

[x_grid, y_grid] = meshgrid([0:0.1:2],[0:0.2:3]);
z_grid = p(1) + p(2)*x_grid + p(3)*y_grid;

figure(1); plot3(x,y,z,'*')
           hold on
           mesh(x_grid,y_grid,z_grid)
           xlabel('x'); ylabel('y'); zlabel('z');
           hold off
-->
p =   1.7689   2.0606   -1.4396
```

Since only very few (`N=100`) points were used the exact parameter values $\vec{p} = (+2, +2, -1.5)$ are note very accurately reproduced. Increasing N will lead to more accurate results for this simulation, or decrease the size of the random noise in `+0.5*randn(N,1)`.

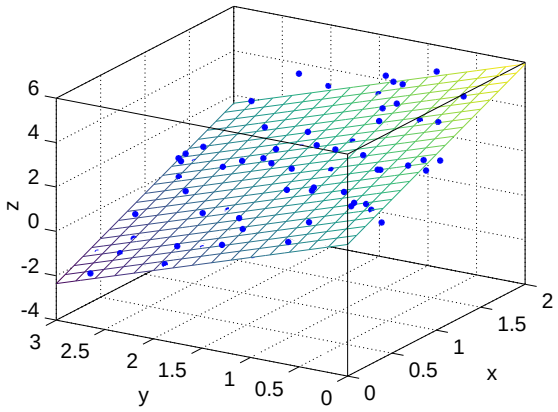


Figure 10: Result of a 3D linear regression

The command `LinearRegression()` does not determine the confidence intervals for the parameters, but it returns the estimated standard deviations, resp. the variances. With these the confidence intervals can be computed, using the Student-t distribution. To determine the CI modify the above code slightly.

```
[p,~,~,p_var] = LinearRegression(F,z);
alpha = 0.05;
p_CI = p + tinv(1-alpha/2,N-3)*[-sqrt(p_var) +sqrt(p_var)]
-->
p_CI =  +1.6944  +2.2357
        +1.8490  +2.2222
        -1.5869  -1.3495
```

The result implies that the 95% confidence intervals for the parameters p_i are given by

$$\begin{aligned} +1.6944 &< p_1 < +2.2357 \\ +1.8490 &< p_2 < +2.2222 \quad \text{with a confidence level of } 95\% \text{ .} \\ -1.5869 &< p_3 < -1.3495 \end{aligned}$$

5.2 Using `regress()`

`MATLAB` and *Octave* provide the command `regress()` to perform linear regressions. The following code determines the best matching straight line to the given data points.

```
x = linspace(0,2,10)'; y = sin(x);
F = [ones(size(x)),x];
[p, p_int, r, r_int, stats] = regress(y,F);
parameters = p
parameter_intervals = p_int
estimated_std = std(r)
-->
parameters =   0.20224
               0.47786
parameter_intervals =  -0.0093515   +0.4138380
                       +0.2995040   +0.6562220
estimated_std =  0.14719
```

The values of the optimal parameters (obviously) have to coincide with the result generated by `LinearRegression()`. Instead of the standard deviations for the parameters `regress()` returns the confidence intervals for the parameters. The above numbers imply for the straight line $y = a + m\,x$

$$\begin{array}{rcccl} -0.0093 & < & a & < & 0.4138 \\ 0.300 & < & m & < & 0.656 \end{array} \quad \text{with a confidence level of } 95\% \ .$$

The value of the confidence level can be adjusted by calling `regress()` with a third argument, see `help regress` .

5.3 Using `lscov()`, `polyfit()` or `ols()`

The command `lscov()` is rather similar to the above, but can return the covariance matrix for the determined parameters.

If your aim is to fit a polynomial to the data, you may use `polyfit()`. The above example for a parabola (polynomial of degree 2) is solved by

```
[p,s] = polyfit(x,y,2);
p
-->
p =  -0.386371   1.250604  -0.026717
```

Observe that the coefficient of the polynomial are returned in deceasing order. Since `regress()` and `LinearRegression()` are more flexible and provide more information your author's advice is to use those, even if `polyfit()` would work.

With *Octave* one may also use the command `ols()`, short for Ordinary Least Square. But as above, there is no advantage over using `LinearRegression()`.

```
p = ols(y,F)
-->
p =  -0.026717  1.250604 -0.386371
```

6 Generating Random Number

In Table 4 find commands to generate random numbers, given by different distributions.

As an example we generate $N = 1000$ random numbers given by a binomial distribution with $n = 9$ trials and $p = 0.8$. Thus each of the 100 random number will be an integer between 0 and 9. Find the result of the code below in Figure 11 and compare with Figure 12(d), showing the result for the exact (non random) distribution.

```
N = 1000;  data = binornd(9, 0.8, N,1);     % generate the random numbers
[height,centers] = hist(data,unique(data))  % data for the histogram
bar(centers,height/sum(height))
xlabel('value'); ylabel('experimental probability')
title('Binomial distribution with n=9, p=0.8')
```

`rand()`	uniform distribution
`randi()`	random integers
`randn()`	normal distribution
`rande()`	exponentially distributed
`randp()`	Poisson distribution
`randg()`	gamma distribution
`normrnd()`	normal distribution
`binornd()`	binomial distribution
`exprnd()`	exponential distribution
`trnd()`	Student-t distribution
`discrete_rnd()`	discrete distribution

Table 4: Commands for generating random numbers

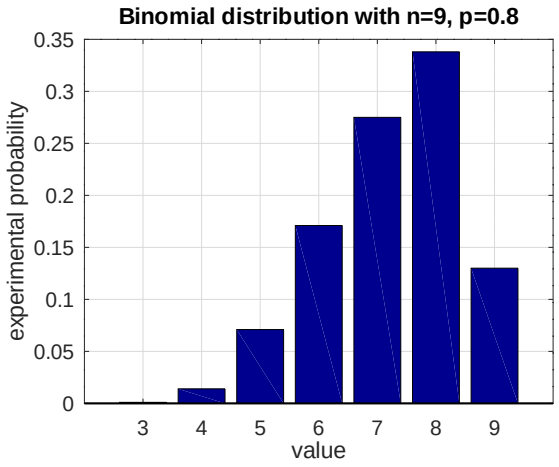


Figure 11: Histogram of random numbers, generated by a binomial distribution with $n = 9$, $p = 0.8$

7 Commands to Work with Probability Distributions

`MATLAB`/*Octave* provides functions to compute the values of probability density functions (PDF), and the cumulative distribution functions (CDF). In addition the inverse of the CDF are provided, i.e. solve CDF$(x) = y$ for x . As examples examine the following short code segments, using the normal distribution.

- To determine the values of the PDF for a normal distribution with mean 3 and standard deviation 2 for x values between -1 and 7 use

```
x = linspace(-1,7);    p = normpdf(x,3,2);
plot(x,p)
```

- To determine the corresponding values of the CDF use `cp = normcdf(x,3,2)` .
- To determine for what values of x the value of the CDF equals 0.025 and 0.975 use

```
norminv([0.025,0.975],3,2)
-->
-0.91993    6.91993
```

The result implies that 95% of all values are between $-0.91 \approx -1$ and $+6.91 \approx 7$. This is consistent with the approximative rule of thumb $\mu \pm 2\,\sigma$.

For most distributions `MATLAB` and *Octave* provide multiple commands to work with the density functions and cumulative distribution functions, see Table 7. The first part of the name of the command consists of an abbreviated name of the distribution and the second part spells out the operation to be applied. As an example consider the normal distribution and then use the command `normpdf()`, `normcdf()`, `norminv()`, `normrnd()` and `normstat()` to work with the normal distribution.

Name of command	Function
`*pdf()`	probability density function
`*cdf()`	cumulative density function
`*inv()`	inverse of the cumulative density function
`*rnd()`	generate random numbers
`*stat()`	compute mean and variance
`*test()`	testing of hypothesis

Table 5: Functions for distributions

In addition one may use the commands `cdf()` and `pdf()` to compute values of the probability density function. As example consider the result of `cdf('normal',0,0,1)`, leading to 0.500 .

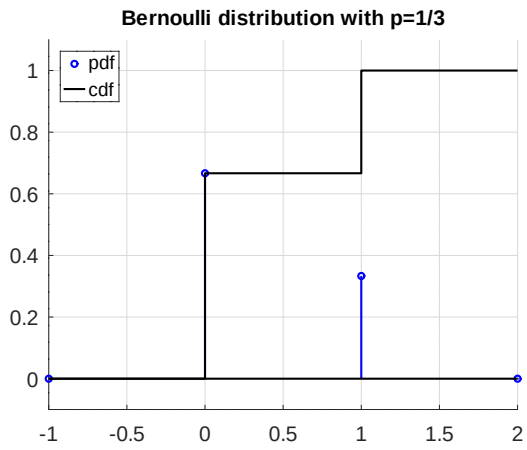
7.1 Discrete distributions

For any discrete distribution the mean value μ and the variance σ^2 are determined by

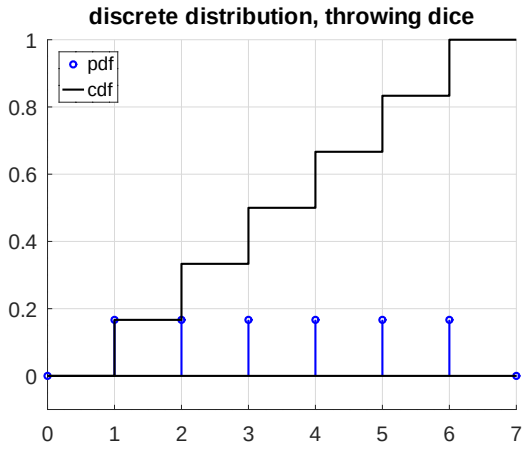
$$\mu = \sum_j \mathrm{pdf}(x_j) \cdot x_j \quad \text{and} \quad \sigma^2 = \sum_j \mathrm{pdf}(x_j) \cdot (x_j - \mu)^2$$

Name of distribution	Function	μ	σ
Discrete	`discrete_pdf(x,v,p)`		
Bernoulli	`discrete_pdf(x,[0 1],[1-p,p])`	p	$\sqrt{p\,(1-p)}$
Binomial	`binopdf(x,n,p)`	$n\,p$	$\sqrt{n\,p\,(1-p)}$
Poisson	`poisspdf(x,lambda)`	λ	$\sqrt{\lambda}$
Hypergeometric	`hygepdf(x,T,M,n)`	$n\,\frac{m}{T}$	

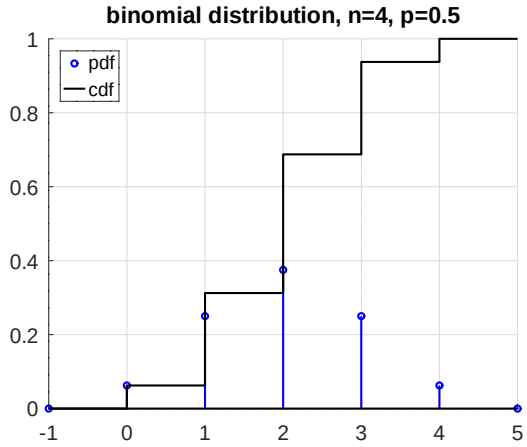
Table 6: Discrete distributions, mean value μ and standard deviation σ



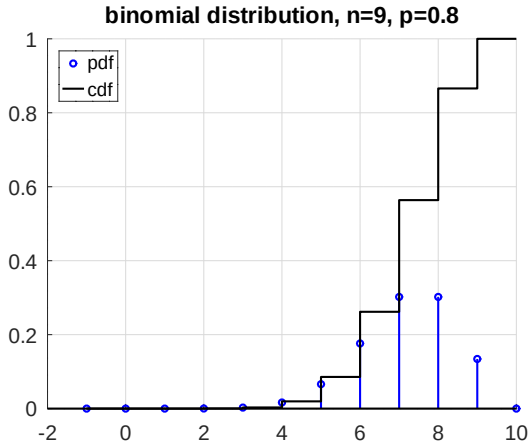
(a) Bernoulli distribution



(b) discrete distribution, dice



(c) binomial distribution



(d) binomial distribution

Figure 12: Graphs of some discrete distributions

7.1.1 Bernoulli distribution and general discrete distributions

With the functions `discrete_pdf()` and `discrete_cdf()`[1] you can generate discrete probability distributions. To generate a Bernoulli distribution with probability 1/3

$$P(X=0)=\frac{2}{3} \quad \text{and} \quad P(X=1)=\frac{1}{3}$$

use the code below, leading to Figure 12(a). There is no need to normalize the total probability to 1, i.e. in the code below `discrete_pdf(x,[0 1],[2 1])` would work just as well.

```
x  = -1:2;
p  = discrete_pdf(x,[0 1],[2/3 1/3]);  cp = discrete_cdf(x,[0 1],[2/3 1/3]);

figure(1); stem(x,p,'b');    hold on
           stairs(x,cp,'k'); hold off
           title('Bernoulli distribution with p=1/3')
           axis([-1 2 -0.1 1.1]); legend('pdf','cdf','location','northwest');
```

The Bernoulli distribution can also be considered a special case of the binomial distribution, with $n=1$.

Throwing a regular dice also leads to a discrete distribution, each of the possible results 1, 2, 3, 4, 5 and 6 will show with probability 1/6 . Find the result in Figure 12(b).

```
x  = 0:7;
p  = discrete_pdf (x,1:6,ones(6,1)/6);
cp = discrete_cdf (x,1:6,ones(6,1)/6);
figure(2); stem(x,p,'b');    hold on
           stairs(x,cp,'k'); hold off
           title('discrete distribution, throwing dice')
           axis([0 7 -0.1 1])
           legend('pdf','cdf','location','northwest');
```

7.1.2 Binomial distribution

The binomial distribution is generated by n independent Bernoulli trials with identical probability p, i.e. for each of the n independent events you obtain the result 1 with probability p . Then add up the results. This leads to

$$\begin{aligned} X &= \sum_{j=1}^{n} X_j \\ P(X=i) = p(i) &= \binom{n}{i} p^i \cdot (1-p)^{n-i} = \frac{n!}{i!\cdot(n-i)!}\, p^i \cdot (1-p)^{n-i} \end{aligned}$$

The code below generates the PDF and CDF for a binomial distribution with 4 events and the individual probability $p = 0.5$. Find the results in Figure 12(c). The resulting distribution is symmetric about the mean value 2 .

[1] The current version of MATLAB does not provide these two commands. Ask this author for a MATLAB compatible versions.

```
x  = -1:5;
p  = binopdf(x,4,0.5);
cp = binocdf(x,4,0.5);

figure(3); stem(x,p,'b');    hold on
           stairs(x,cp,'k'); hold off
           title('binomial distribution, n=4, p=0.5')
           legend('pdf','cdf','location','northwest');
```

Similarly you may examine the distribution function of 9 draws with an individual probability of $p = 0.8$. In the result in Figure 12(d) it is clearly visible that the result is skewed towards higher values, since they occur with a higher probability..

```
x  = -1:10;
p  = binopdf(x,9,0.8);
cp = binocdf(x,9,0.8);

figure(4); stem(x,p,'b');    hold on
           stairs(x,cp,'k'); hold off
           title('binomial distribution, n=9, p=0.8')
           legend('pdf','cdf','location','northwest');
```

The command `binostat` determines mean and standard deviation of a binomial distribution.

7.1.3 Poisson distribution

A Poisson distribution with parameter $\lambda > 0$ is given by

$$P(X = i) = p(i) = \frac{\lambda^i}{i!}\, e^{-\lambda}$$

Find the graph of the Poisson distribution with $\lambda = 2.5$ in Figure 13, generated by

```
x  = -1:10; lambda = 2.5;
p  = poisspdf(x,lambda);
cp = poisscdf(x,lambda);

figure(5); stem(x,p,'b');    hold on
           stairs(x,cp,'k'); hold off
           title('Poisson distribution, \lambda=2.5')
           legend('pdf','cdf','location','northwest');
```

7.2 Continuous distributions

For all continuous distributions one may compute the average value μ, the standard deviation σ and the cumulative distribution function by integrals. The mode and median of the distributions are characterized as special values of the cumulative distribution function.

$$\begin{aligned} \mu &= \int_{-\infty}^{+\infty} \mathrm{pdf}(x) \cdot x \; dx \\ \sigma^2 &= \int_{-\infty}^{+\infty} \mathrm{pdf}(x) \cdot (x-\mu)^2 \; dx \\ \mathrm{cdf}(x) &= \int_{-\infty}^{x} \mathrm{pdf}(s) \; ds \end{aligned}$$

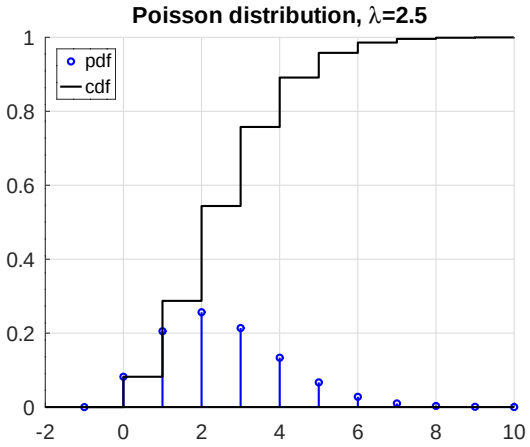


Figure 13: A Poisson distribution with $\lambda = 2.5$

$$\begin{aligned} \text{cdf(median)} &= 0.5 \\ \text{cdf(mode)} &= \max_{x\in\mathbb{R}}\{\text{pdf}(x)\} \end{aligned}$$

Name	Function	μ	σ	median
Uniform	`unifpdf(x,0,1)`	$1/2$	$1/\sqrt{12}$	$1/2$
Uniform	`unifpdf(x,A,B)`	$\frac{A+B}{2}$	$\frac{B-A}{\sqrt{12}}$	$\frac{A+B}{2}$
Normal	`normpdf(x,mu,sigma)`	μ	σ	μ
Standard Normal	`stdnormal_pdf(x)`	0	1	0
Exponential	`exppdf(x,lambda)`	λ	λ	$\lambda \ln 2$
Student-t	`tpdf(x,n)`	0	$\sqrt{\frac{n}{n-2}}$ if $n > 2$	0
χ distribution		$\frac{\sqrt{2}\,\Gamma((n+1)/2)}{\Gamma(n/2)}$	$\sqrt{n-\mu^2}$	
χ^2 distribution	`chi2pdf(x,n)`	n	$\sqrt{2\,n}$	$\approx n\,(1-\frac{2}{9\,n})^3$

Table 7: Continuous distributions, mean value μ, standard deviation σ and median

7.2.1 Uniform distribution

The uniform distribution on the interval $[A\,,\,B]$ is characterized by

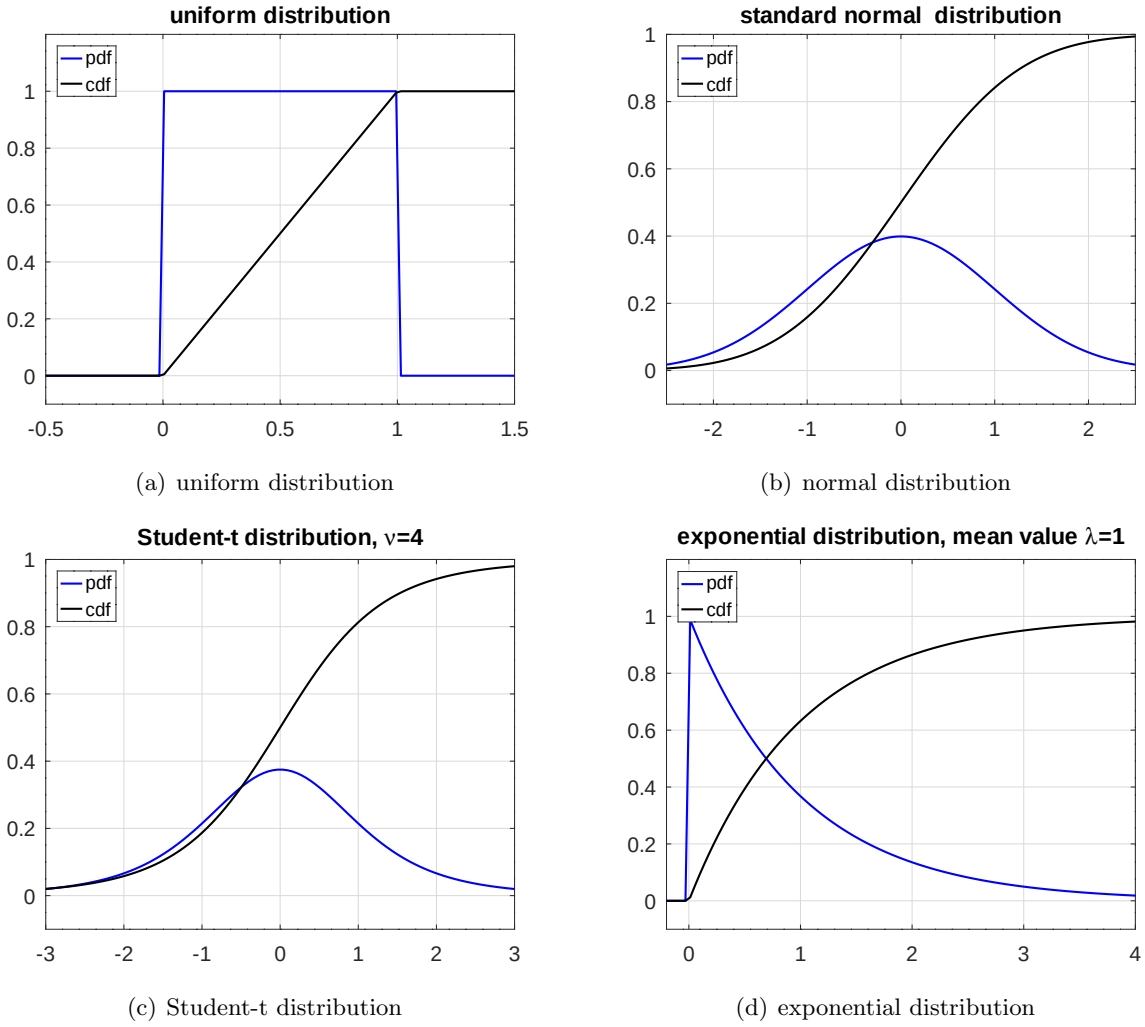
$$\text{pdf}(x) = \begin{cases} \frac{1}{B-A} & \text{for} \quad A \le x \le B \\ 0 & \text{otherwise} \end{cases}$$

$$\text{cdf}(x) = \begin{cases} 0 & \text{for} \quad x \le A \\ \frac{x-A}{B-A} & \text{for} \quad A \le x \le B \\ 1 & \text{for} \quad B \le x \end{cases}$$

Find the result for a uniform distribution on the interval $[0\,,\,1]$ in Figure 14(a), generated by

```
x  = linspace(-0.5,1.5);
p  = unifpdf(x,0,1);
cp = unifcdf(x,0,1);

figure(1); plot(x,p,'b',x,cp,'k')
           title('uniform distribution')
           axis([-0.5 1.5 -0.1 1.2])
           legend('pdf','cdf','location','northwest')
```



(a) uniform distribution

(b) normal distribution

(c) Student-t distribution

(d) exponential distribution

Figure 14: Graphs of some continous distributions

7.2.2 Normal distribution

The normal distribution with mean μ and standard devition σ is given by

$$\begin{aligned} \text{pdf}(x) &= \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\,\sigma^2}) \\ \text{cdf}(x) &= \int_{-\infty}^{x} \text{pdf}(s)\,ds = \frac{1}{2}\left(1+\text{erf}(\frac{x-\mu}{\sqrt{2}\,\sigma})\right) \end{aligned}$$

Find the result for a normal distribution with mean $\mu = 1$ and standard deviation $\sigma = \frac{1}{2}$ in Figure 14(b), generated by

```
x  = linspace(-0.5,3.0);
p  = normpdf(x,1,0.5);
cp = normcdf(x,1,0.5);

figure(1); plot(x,p,'b',x,cp,'k')
           title('normal  distribution, \mu= 1, \sigma=0.5')
           axis([-0.5 3.0 -0.1 1.])
           legend('pdf','cdf','location','northwest')
```

7.2.3 Student-t distribution

The Student-t or Student's t distribution arises when estimating the variances of small samples of normally distributed numbers. It can be expressed on terms of the Gamma function[2] Γ by

$$\text{pdf}(x,\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

where $\nu \in \mathbb{N}$ is the number of degrees of freedom. The corresponding cumulative density function is given by the general formula

$$\text{cdf}(x,\nu) = \int_{-\infty}^{x} \text{pdf}(s,\nu)\; ds$$

and there is no elementary expression for it, for most values of ν.

- The probability density function resembles a normal distribution with mean 0 and standard deviation 1, but it is wider and and a lower maximal value. The standard deviation is not equal to 1 .
- As the number of degrees of freedom ν increases it converges to a standard normal distribution, see Figure 15.
- For some small values of ν there are explicit formulas, shown in Table 8.

Figure 15 was generated by

```
x  = linspace(-3,3);
p1 = tpdf(x,1); p2 = tpdf(x,2); p10 = tpdf(x,10); pn = normpdf(x,0,1);

figure(1); plot(x,p1,x,p2,x,p10,x,pn)
           title('Student-t distribution'); axis([-3 3 -0.1 0.5])
           legend('\nu=1','\nu=2','\nu=10','normal','location','northwest')
```

7.2.4 χ^2 distribution

The χ^2–distribution with parameter n (degrees of freedom) is defined for $x > 0$ and given by

$$\begin{aligned}\text{pdf}(x) &= \frac{1}{2^{n/2}\,\Gamma(\frac{n}{2})}\, x^{\frac{n}{2}-1}\, \exp(-\frac{x}{2}) \\ \text{cdf}(x) &= \int_0^x \text{pdf}(s)\; ds \;=\; \frac{1}{\Gamma(\frac{n}{2})}\,\gamma(\frac{n}{2}\,,\,\frac{x}{2})\end{aligned}$$

where $\gamma(s,t)$ is the lower incomplete gamma function. The mode (maximal value) is attained at $\max\{n-2\,,\,0\}$. Find the result for a a few χ^2 distributions in Figure 16, generated by

```
```

[2]The Gamma function is an extension of the well known factorial function, $\Gamma(n+1) = n! = \prod_{i=1}^{n} i$.

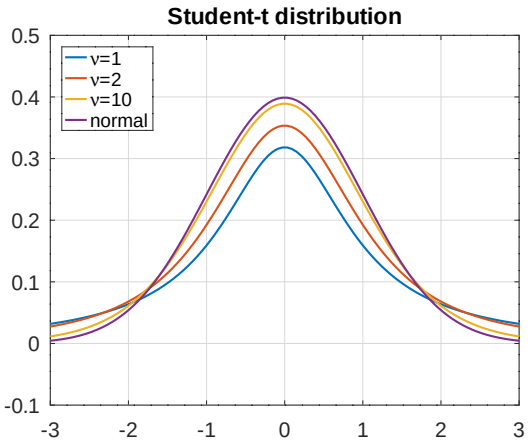


Figure 15: Student-t distributions and a normal distribution

ν	$\text{pdf}(x,\nu)$	$\text{cdf}(x,\nu)$
1	$\frac{1}{\pi\,(1+x^2)}$	$\frac{1}{2}+\frac{1}{\pi}\,\arctan(x)$
2	$\frac{1}{(2+x^2)^{3/2}}$	$\frac{1}{2}+\frac{x}{2\sqrt{2+x^2}}$
3	$\frac{6\sqrt{3}}{\pi\,(3+x^2)^2}$	
∞	$\frac{1}{\sqrt{2\pi}}\,\exp(-\frac{x^2}{2})$	$\frac{1}{2}\,(1+\text{erf}(\frac{x}{\sqrt{2}}))$

Table 8: Student-t distribution for some small values of ν

```
x = linspace(0,4);
pdf1= chi2pdf(x,1); pdf2= chi2pdf(x,2); pdf3= chi2pdf(x,3); pdf5= chi2pdf(x,5);
cdf1= chi2cdf(x,1); cdf2= chi2cdf(x,2); cdf3= chi2cdf(x,3); cdf5= chi2cdf(x,5);

figure(1); plot(x,pdf1,x,pdf2,x,pdf3,x,pdf5)
           ylabel('pdf(x)'); title('\chi^2 pdf');
           legend('n=1','n=2','n=3','n=5'); axis([0,4,0,1])

figure(2); plot(x,cdf1,x,cdf2,x,cdf3,x,cdf5)
           ylabel('cdf(x)'); title('\chi^2 cdf'); axis([0,4,0,1.1]);
           legend('n=1','n=2','n=3','n=5','location','northwest');
```

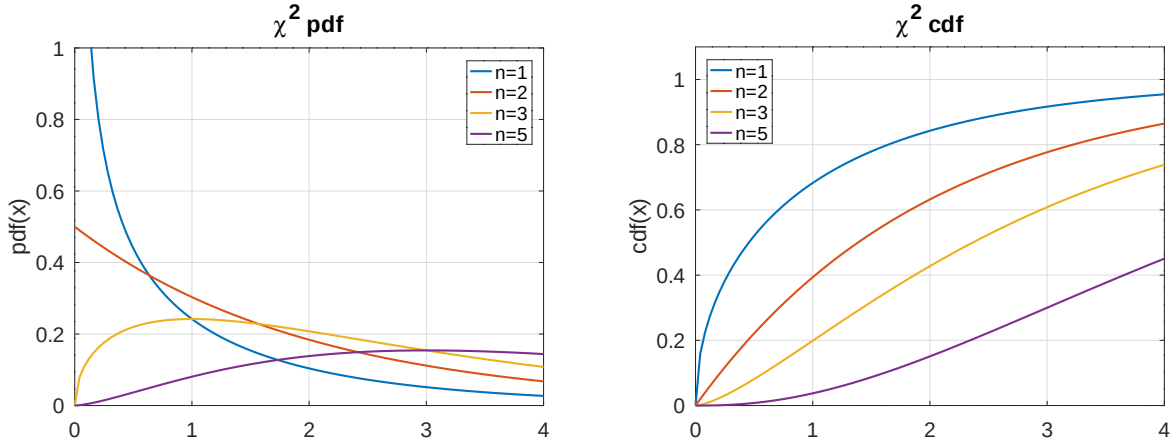
7.2.5 Exponential distribution

The exponential distribution[3] with mean λ in `MATLAB` and *Octave* is given by

$$\text{pdf}(x) \;=\; \frac{1}{\lambda}\,\exp(-x/\lambda) \quad \text{and} \quad \text{cdf}(x) \;=\; 1-\exp(-x/\lambda)$$

and are computed by `exppdf(x,lambda)` and `expcdf(x,lambda)`. Find the graphs for $\lambda = 1,\ 0.5$ and 0.2 in Figure 17. Some typical application of the exponential distribution are the length of phone calls, length of wait in a line, lifetime of components, ...

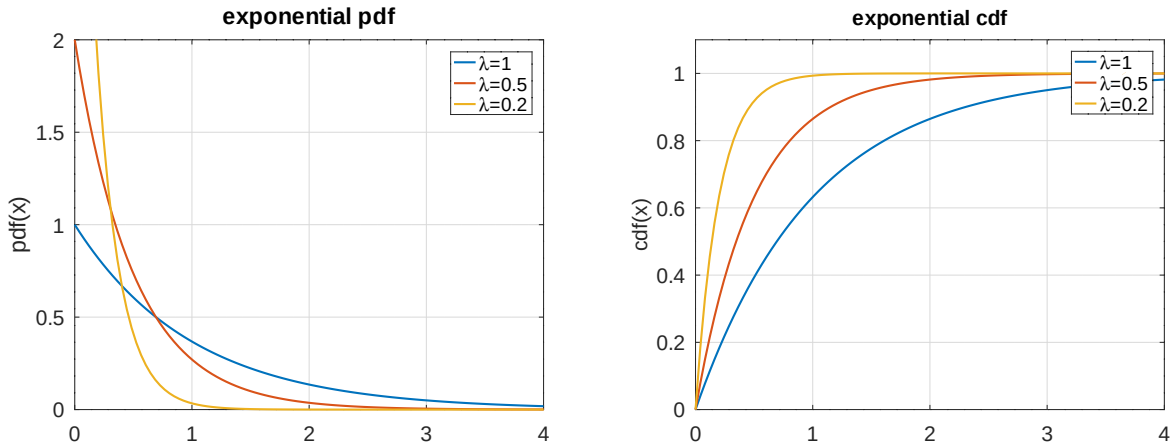
[3]Some references use the factor $1/\lambda$ instead of λ, i.e. $\text{pdf}(x) = \lambda\,\exp(-x\,\lambda)$ and $\text{cdf}(x) = 1-\exp(-x\,\lambda)$.



(a) χ^2 distribution functions

(b) χ^2 cumulative distribution functions

Figure 16: χ^2 distributions, PDF and CDF



(a) probability distribution functions

(b) cumulative distribution functions

Figure 17: Exponential distributions

8 Commands for Confidence Intervals and Hypothesis Testing

In this section a few command to determine confidence intervals and testing of hypothesis are shown. My personal preference is clearly to work with confidence intervals. Is is (too) easy to abuse the commands for hypothesis testing and computing P values.

8.1 Confidence Intervals

A confidence interval contains the true parameter μ to be examined with a certain level of confidence, as illustrated in Figure 18. One has to chose a level of significance α. Then the confidence interval has to contain the true parameter μ with a level of confidence (probability) of $p = 1 - \alpha$. Often used values for α are $0.05 = 5\%$ and $0.01 = 1\%$.

$$\begin{aligned} 0 < \alpha < 1 \quad &: \quad \text{level of significance} \\ p = 1 - \alpha \quad &: \quad \text{level of confidence} \end{aligned}$$

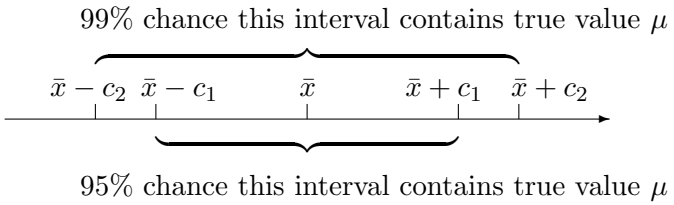


Figure 18: Confidence intervals at levels of significance $\alpha = 0.05$ and $\alpha = 0.01$

For the following examples we repeatedly use a data set to illustrate the commands. The numbers may represent the number of defects detected on a sample of 17 silicon wafers, selected at random from a large production.

WaferDefects.txt

```
7 16 19 12 15 9 6 16 14 7 2 15 23 15 12 18 9
```

8.1.1 Estimating the mean value μ, with (supposedly) known standard deviation σ

Assume to have data with an unknown mean value μ, but a known standard deviation σ. This is a rather unusual situation, in most cases the standard deviation is not known and one has to use the similar computations in the next section 8.1.2. For sake of simplicity start with a known value of σ.

A sampling of n data points leads to x_i and you want to determine the mean value μ. According to the central limit theorem the random variable

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

is approximated by a normal distribution with mean μ and standard deviation $\sigma/\sqrt{n}$. Now we seek a value u such that the green area in Figure 19(a) equals $(1 - \alpha)$, where α is a small positive value. This implies that the true (unknown) value of μ is with a high probability in the green section in Figure 19(a). This is a two–sided confidence interval. If we want to know whether the true value of μ is below (or above) a certain threshold we use a similar argument to determine the one–sided confidence interval in Figure 19(b) .

- To determine a two–sided confidence interval for the significance level α examine Figure 19(a) and proceed as follows:

1. Determine u such that

$$\begin{aligned}(1-\alpha) &= P(-u < U < u) = \frac{1}{\sqrt{2\pi}} \int_{-u}^{u} e^{-x^2/2}\, dx \\ \frac{\alpha}{2} &= P(U < -u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-u} e^{-x^2/2}\, dx = \mathrm{cdf}(-u) \\ &= P(+u < U) = \frac{1}{\sqrt{2\pi}} \int_{+u}^{+\infty} e^{-x^2/2}\, dx = 1 - \mathrm{cdf}(+u)\end{aligned}$$

   With `MATLAB`/*Octave* this value may be determined by `u = - norminv(alpha/2)` or by `u = norminv(1-alpha/2)`.

2. Then determine the estimator $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the two–sided confidence interval is given by $[\bar{x} - \frac{u\,\sigma}{\sqrt{n}}\,,\, \bar{x} + \frac{u\,\sigma}{\sqrt{n}}]$, i.e.

$$P(\bar{x} - \frac{u\,\sigma}{\sqrt{n}} < x < \bar{x} + \frac{u\,\sigma}{\sqrt{n}}) = 1 - \alpha$$

- For the above example we may use the code below.

```
data = load('WaferDefects.txt');
N = length(data)
% we use the estimated variance as the supposedly given value
sigma = std(data); % this is NOT a realistic situation
alpha = 0.05 % choose the significance level
u = -norminv(alpha/2)
x_bar = mean(data);
ConfidenceInterval = [x_bar - u*sigma/sqrt(N) , x_bar + u*sigma/sqrt(N)]
-->
ConfidenceInterval = [10.082   15.212]
```

You may also use the command `ztest()` from Section 8.2.2 to compute the confidence interval.

The above code can be reused with a smaller significance level `alpha = 0.01`, leading to a wider confidence interval of $[9.2760\,,\, 16.0182]$.

- To determine the one–sided confidence interval for a significance level α examine Figure 19(b) and proceed as follows:

1. Determine u such that

$$\begin{aligned}(1-\alpha) &= P(U < u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-x^2/2}\, dx = \mathrm{cdf}(u) \\ \alpha &= P(u < U) = \frac{1}{\sqrt{2\pi}} \int_{u}^{+\infty} e^{-x^2/2}\, dx = 1 - \mathrm{cdf}(u)\end{aligned}$$

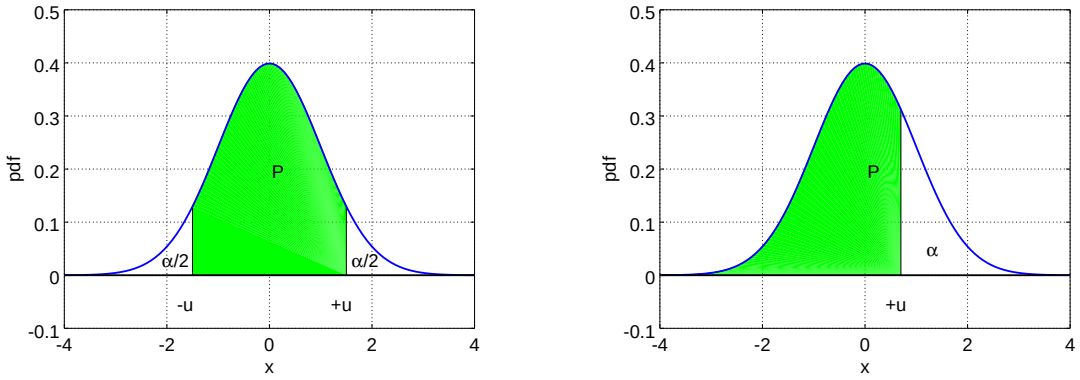
   With `MATLAB`/*Octave* this value may be determined by `u = norminv(1-alpha)`.

2. Determine the estimator $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Now one–sided confidence interval is given by $[-\infty\,,\, \bar{x} + \frac{u\,\sigma}{\sqrt{n}}]$, i.e.

$$P(x < \bar{x} + \frac{u\,\sigma}{\sqrt{n}}) = 1 - \alpha$$

- For the above example we may use the following code.

```
data = load('WaferDefects.txt');
N = length(data)
% we use the estimated variance as the supposedly given value
sigma = std(data); % this is NOT a realistic situation
alpha = 0.05 % choose the significance level
u = norminv(1-alpha)
x_bar = mean(data);
UpperLimit =  x_bar + u*sigma/sqrt(N)
-->
UpperLimit =  14.800
```



(a) two–sided confidence interval (b) one–sided confidence interval

Figure 19: Two- and one–sided confidence intervals

8.1.2 Estimating the mean value μ, with unknown standard deviation σ

Assuming you have data X_i all given by the same normal distribution $N(\mu, \sigma)$ with mean μ and standard deviation σ. Now the unbiased estimators

$$\bar{X}\frac{1}{n} = \sum_{i=1}^{n} X_i \quad \text{and} \quad S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

are **not independent**. The distribution of the random variable

$$Z = \frac{\mu - \bar{X}}{S/\sqrt{n}}$$

is a Student-t distribution with $n-1$ degrees of freedom, see Section 7.2.3.

- To determine a two–sided confidence interval with significance level α examine Figure 19(a) and use

$$\begin{aligned}(1-\alpha) &= P(-u < Z < +u) = P(-u < \frac{\mu - \bar{X}}{S/\sqrt{n}} < +u) \\ &= P(-u\frac{S}{\sqrt{n}} < \mu - \bar{X} < +u\frac{S}{\sqrt{n}}) = P(\bar{X} - u\frac{S}{\sqrt{n}} < \mu < \bar{X} + u\frac{S}{\sqrt{n}})\end{aligned}$$

  The value of u is determined by

$$\frac{\alpha}{2} = \int_{-\infty}^{-u} \mathrm{pdf}(x)\, dx = \mathrm{cdf}(-u)$$

and computed by `u = - tinv(alpha/2,n-1)`. With the estimators

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

the two–sided confidence interval is given by

$$\left[\bar{x} - u\,\frac{\sigma}{\sqrt{n}}\,,\, \bar{x} + u\,\frac{\sigma}{\sqrt{n}}\right]$$

- For the above example we may use the code below.

```
data = load('WaferDefects.txt');
N = length(data)
alpha = 0.05 % choose the significance level
u = -tinv(alpha/2,N-1)
x_bar = mean(data);
sigma = std(data);
ConfidenceInterval = [x_bar - u*sigma/sqrt(N) , x_bar + u*sigma/sqrt(N)]
-->
ConfidenceInterval = [9.8727   15.4215]
```

Observe that this confidence interval is slightly wider that the one with (supposedly) known standard deviation σ. This is reasonable, since we have less information at our disposition.

- To determine the one–sided confidence interval for the significance level α examine Figure 19(b) and proceed as follows:

  1. Determine u such that

  $$(1-\alpha) = P(U < u) = \int_{-\infty}^{u} \mathrm{pdf}(x)\,dx = \mathrm{cdf}(u)$$

  With `MATLAB`/*Octave* this value may be determined by `u = tinv(1-alpha,n-1)`.

  2. With the estimators

  $$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

  the one–sided confidence interval is given by $[-\infty\,,\,\bar{x} + \frac{u\,\sigma}{\sqrt{n}}]$, i.e.

  $$P(x < \bar{x} + \frac{u\,\sigma}{\sqrt{n}}) = 1 - \alpha$$

- For the above example we may use the following code.

```
data = load('WaferDefects.txt');
N = length(data)
alpha = 0.05 % choose the significance level
u = tinv(1-alpha,N-1)
x_bar = mean(data);
sigma = std(data);
UpperLimit =  x_bar + u*sigma/sqrt(N)
-->
UpperLimit =  14.932
```

Observe that for large samples ($n \gg 1$) the Student-t distribution is close to the standard normal distribution and the results for the confidence intervals for known or unknown standard deviation σ differ very little.

8.1.3 Estimating the variance for nomaly distributed random variables

Assume that X_i are random variables with a normal distribution $N(\mu, \sigma)$, i.e. with mean μ and standard deviation σ. An unbiased estimator for the variance σ^2 is given by

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$$

This is a random variable whose distribution is related to the χ^2 distribution. The modified variable

$$Y = \frac{(n-1)\, S^2}{\sigma^2} = \frac{1}{\sigma^2} \sum_{i=1}^{n} (X_i - \bar{X})^2$$

follows a χ^2 distribution with $n-1$ degrees of freedom. To determine confidence intervals we have to observe that this distribution is not symmetric, see Section 7.2.4. Obviously we can not obtain negative values. The values of $\chi^2_{\alpha/2,n-1}$ and $\chi^2_{1-\alpha/2,n-1}$ may be characterized by

$$\begin{aligned}
(1-\alpha) &= P(\chi^2_{\alpha/2,n-1} < Y < \chi^2_{1-\alpha/2,n-1}) = \int_{\chi^2_{\alpha/2,n-1}}^{\chi^2_{1-\alpha/2,n-1}} \mathrm{pdf}(x)\, dx \\
\frac{\alpha}{2} &= \int_{0}^{\chi^2_{\alpha/2,n-1}} \mathrm{pdf}(x)\, dx \;=\; \mathrm{cdf}(\chi^2_{\alpha/2,n-1}) \\
\frac{\alpha}{2} &= \int_{\chi^2_{1-\alpha/2,n-1}}^{+\infty} \mathrm{pdf}(x)\, dx \;=\; 1 - \mathrm{cdf}(\chi^2_{1\alpha/2,n-1})
\end{aligned}$$

and thus can be computed by `chi2inv(alpha/2,n-1)` resp. `chi2inv(1-alpha/2,n-1)`. Since

$$\begin{aligned}
(1-\alpha) &= P\left(\chi^2_{\alpha/2,n-1} < \frac{(n-1)\, S^2}{\sigma^2} < \chi^2_{1-\alpha/2,n-1}\right) \\
&= P\left(\frac{\chi^2_{\alpha/2,n-1}}{(n-1)\, S^2} < \frac{1}{\sigma^2} < \frac{\chi^2_{1-\alpha/2,n-1}}{(n-1)\, S^2}\right) \\
&= P\left(\frac{(n-1)\, S^2}{\chi^2_{1-\alpha/2,n-1}} < \sigma^2 < \frac{(n-1)\, S^2}{\chi^2_{\alpha/2,n-1}}\right)
\end{aligned}$$

and we have a confidence interval for the variance σ^2

$$\left[\frac{(n-1)\, S^2}{\chi^2_{1-\alpha/2,n-1}}, \frac{(n-1)\, S^2}{\chi^2_{\alpha/2,n-1}}\right]$$

or for the standard deviation σ

$$\left[\frac{\sqrt{n-1}\, S}{\sqrt{\chi^2_{1-\alpha/2,n-1}}}, \frac{\sqrt{n-1}\, S}{\sqrt{\chi^2_{\alpha/2,n-1}}}\right]$$

For the above example we use the code below.

```
data = load('WaferDefects.txt');
N = length(data);
alpha = 0.05; % choose the significance level
chi2_low  = chi2inv(alpha/2,N-1);
chi2_high = chi2inv(1-alpha/2,N-1);
```

```
sigma = std(data)
ConfidenceIntervalVariance = sigma^2*(N-1)*[1/chi2_high , 1/chi2_low]
ConfidenceIntervalStd = sqrt(ConfidenceIntervalVariance)
-->
sigma = 5.3961
ConfidenceIntervalVariance = [16.151   67.444]
ConfidenceIntervalStd      = [4.0188   8.2124]
```

Observe that the confidence interval for the standard deviation is not symmetric about the estimated value of 5.3961 .

8.1.4 Estimating the parameter p for a binomial distribution

For random variable X_i with a binomial distribution with parameter $0 < p < 1$ we use

$$\bar{P} = \frac{\bar{k}}{n} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

as an unbiased estimator for the parameter p. To construct a confidence interval for p we seek a lower limit p_l and an upper limit p_u such that

$$P(p_l < p < p_u) = 1 - \alpha$$

For this to happen we need to solve the equations

$$\begin{aligned} P(S_n > \bar{k}) &= \sum_{i=\bar{k}+1}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) p_l^i \, (1-p_l)^{n-i} = 1 - \mathrm{cdf}(\bar{k}, p_l) = \frac{\alpha}{2} \\ P(S_n \leq \bar{k}) &= \sum_{i=1}^{\bar{k}-1} \left(\begin{array}{c} n \\ i \end{array} \right) p_u^i \, (1-p_u)^{n-i} = \mathrm{cdf}(\bar{k}, p_u) = \frac{\alpha}{2} \end{aligned}$$

Thus we have to solve the equations $1 - \mathrm{cdf}(\bar{k}, p_l) = \alpha/2$ and $\mathrm{cdf}(\bar{k}, p_u) = \alpha/2$ for the unknowns p_l and p_u. This can be done using the command `fzero()`. Use `help fzero` to obtain information about this command, used to solve a single equation. For the command `fzero()` we have to provide the interval in which p is to be found, e.g. $0 \leq p \leq 1$.

Assuming that out of 1000 samples only 320 objects satisfy the desired property. The estimator for p is $\bar{p} = \frac{320}{1000} = 0.32$. Then use the code below to determine the two–sided confidence interval $[p_l \, , \, p_u]$ at significance level $\alpha = 0.05$.

```
N = 1000; Yes = 320; alpha = 0.05;
p_low = fzero(@(p)1-binocdf(Yes-1,N,p)-alpha/2,[0,1]);
p_up  = fzero(@(p)binocdf(Yes,N,p)-alpha/2,[0,1]);
Interval_Binom = [p_low p_up]
-->
Interval_Binom = [0.29115  0.34991]
```

Since $N = 1000$ is large and p is neither close to 1 or 0 we can approximate the binomial distribution by a normal distribution with mean 0.32 and standard deviation $\sigma = \sqrt{\frac{p\,(1-p)}{N}}$. The resulting confidence interval has to be similar to the above, as confirmed by the code below.

```
```

```
N = 1000; Yes = 320; alpha = 0.05;
% use an approximative normal distribution
p = Yes/N;
u = -norminv(alpha/2);
sigma = sqrt(p*(1-p)/N)
Interval_Normal = [ p-u*sigma  p+u*sigma ]
-->
Interval_Normal =  [0.29109   0.34891]
```

In the above example a two–sided interval is constructed. There are applications when a one–sided interval is required. Examine a test of 100 samples, with only 2 samples failing. Now determine an upper limit for the fail rate, using a confidence level of $\alpha = 0.01$. The parameter p (the fail rate) is too large if the probability to detect 2 or less fails is smaller than α. Thus the upper limit p_u satisfies the equation

$$P(S_n \leq 2) = \sum_{i=0}^{2} \left(\begin{array}{c} 100 \\ i \end{array} \right) p_u^i (1-p_u)^{100-i} = \mathrm{cdf}(2, p_u) = \alpha$$

The code below shows that the fail rate is smaller than $\approx 8\%$, with a probability of $1 - \alpha = 99\%$.

```
N = 100; Fail = 2; alpha = 0.01;
p_up  = fzero(@(p)binocdf(Fail,N,p)-alpha,[0,1]);
-->
p_up = 0.081412
```

Rerunning the above code with $\alpha = 5\%$ leads to a fail rate smaller than $\approx 6\%$, with a probability of 95% . The maximal fail rate is smaller now, since we accept a lower probability. An approximation by a normal distribution is not justified in this example, since p is rather close to 0 . The (wrong) result for the fail rate would be $\approx 5.3\%$ for $\alpha = 1\%$.

8.2 Hypothesis Testing, P Value

`MATLAB` and *Octave* provide a set of command to use the method of testing a hypothesis, see Table 9. The commands can apply one–sided and two–sided tests, and may determine a confidence interval.

`ztest()`	testing for the mean, with known σ
`ttest()`	testing for the mean, with unknown σ
`binotest()`	testing for p, using a binomial distribution

Table 9: Commands for testing a hypothesis

8.2.1 A coin flipping example

When flipping a coin you (usually) assume that the coin is fair, i.e. "head" and "tail" are equally likely to show, or the probability p for "head" is $p = 0.5$. This is a **Null Hypothesis**.

$$\text{Null hypothesis} \quad H_0 \quad : \quad p = \frac{1}{2}$$

The corresponding **alternative Hypothesis** is

$$\text{Alternative hypothesis} \quad H_1 \quad : \quad p \neq \frac{1}{2}$$

By flipping a coin 20 times you want to decide whether the coin is fair. Choose a level of significance α and determine the resulting **domain of acceptance** A. In this example the domain of acceptance is an interval containing 10 . If the actual number of heads is in A you accept the hypothesis $p = \frac{1}{2}$, otherwise you reject it. The probability of rejecting H_0, even if is is true, should be α, i.e. α is the probability of committing a type 1 error. You might also commit a type 2 error, i.e. accept H_0 even if it is not true. The probability of committing a type 2 error is β and $1 - \beta$ is called the **power of the test**. The relations are shown in Table 10.

	H_0 is true	H_1 is true
H_0 accepted	$1 - \alpha$ correct	β type 2 error
H_0 rejected	α type 1 error	$1 - \beta$ correct

Table 10: Errors when testing a hypothesis with level of significance α

Obviously the choice of the level of significance has an influence on the result.

- If $0 < \alpha < 1$ is very small:
  - The domain of acceptance A will be large, and we are more likely to accept the hypothesis, even if it is wrong. Thus we might make a type 2 error.
  - The probability to reject a true hypothesis is small, given by α. Thus we are not very likely to make a type 1 error.
- If $0 < \alpha < 1$ is large:
  - The domain of acceptance A will be small, and we are more likely to reject the hypothesis, even if it is true. Thus we might make a type 1 error.
  - The probability to accept a false hypothesis is small. Thus we are not very likely to make a type 2 error.
- The smaller the value α of the level of significance, the more likely the hypothesis H_0 is accepted. Thus there is a smallest value of α, leading to rejection of H_0 .

The smallest value of α for which the null hypothesis H_0 is rejected, based on the given sampling, is called the **P value**.

8.2.2 Testing for the mean value μ, with (supposedly) known standard deviation σ, `ztest()`

Assume to have normally distributed data with an unknown mean value μ, but known standard deviation σ, just as in Section 8.1.1. The null hypothesis to be tested is that the mean value equals a given value μ_0. For a given level of significance α the domain of acceptance A is characterized

Null hypothesis H_0	mean $\mu = \mu_0$
Alternative hypothesis H_1	mean $\mu \neq \mu_0$

by

$$1 - \alpha = P(\bar{X} \in A) = P(\mu_0 - a \leq \bar{X} \leq \mu_0 + a)$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ is an unbiased estimator of the true mean value μ. Using the central limit theorem we know that the random variable $U = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$ follows a standard normal distribution. Using this we compute a by `a = -norminv(alpha/2)`, i.e.

$$\frac{\alpha}{2} = \int_{-\infty}^{-a} \mathrm{pdf}(x)\, dx$$

and then the domain of acceptance is given by

$$A = [\mu_0 - a\,\frac{\sigma}{\sqrt{n}}\,,\ \mu_0 + a\,\frac{\sigma}{\sqrt{n}}]$$

The P value is characterized by

$$\begin{aligned} 1-\alpha_{min} &= P(|\bar{X}-\mu_0| \geq |\bar{x}-\mu_0|) \\ P = \alpha_{min} &= P(\bar{X} < \mu_0 - |\bar{x}-\mu_0|) + P(\bar{X} > \mu_0 + |\bar{x}-\mu_0|) \\ &= 2\,P(\bar{X} < \mu_0 - |\bar{x}-\mu_0|) \end{aligned}$$

and thus can be computed by $P = 2{*}\mathrm{normcdf}(-|\bar{x}-\mu_0|\,\frac{\sqrt{n}}{\sigma})$. This is easily coded in `MATLAB`/*Octave*. Observe that P also gives the probability that $\bar{X}$ is further away from the mean value μ_0 than the already observed $\bar{x}$, which is used for the test. This is equal to the the total probability of all events less likely than the observed $\bar{x}$.

As example we want to test the hypothesis that the average number of defects in the wafers for the data set introduced on page 27 is given by 14.

```
data = load('WaferDefects.txt');
mu = mean(data) % the mean of the sampling
n = length(data);
mu0 = 14;               % test against this mean value
sigma = sigma(data);    % assumed value of the standard deviation
alpha = 0.05;           % choose level of significance

a = -norminv(alpha/2);
if abs(mu-mu0)<a*sigma/sqrt(n)
   disp('H_0 not rejected, might be true')
else
   disp('H_0 rejected, probably false')
end%if
P_value = 2*normcdf(-abs(mu-mu0)*sqrt(n)/sigma)
-->
mu =  12.647
H_0 not rejected, might be true
P_value =  0.30124
```

Observe that the average μ of the sample is well below the tested value of $\mu_0 = 14$. Since the size of the sample is rather small, we do not have enough evidence to reject the hypothesis. This **does not** imply that the hypothesis is true. The computed P value is larger than the chosen level of significance $\alpha = 0.05$, which also indicated that the hypothesis can not be rejected.

With the command `ztest()` you can test this hypothesis. It will in addition determine the confidence interval using the results from Section 8.1.1.

```
[H, PVAL, CI] = ztest (data, mu0, sigma)
-->
H = 0
PVAL =  0.30124
CI =   10.082   15.212
```

In the documentation of `ztest` find the explanation for the results.

- Since $H = 0$ the null hypothesis not rejected, i.e. it might be true.
- Since the P value of 0.3 is larger than $\alpha = 0.05$ the null hypothesis not rejected.
- The confidence interval $[10.082\,,\,15.212]$ is determined by the method in Section 8.1.1.

The default value for the level of significance is $\alpha = 0.05 = 5\%$. By providing more arguments you can used different values, e.g. `[H, PVAL, CI] = ztest(data,mu0,sigma,'alpha',0.01)` .

8.2.3 Testing for the mean value μ, with unknown standard deviation σ, `ttest()`

Assume to have normally distributed data with an unknown mean value μ and standard deviation σ, just as in Section 8.1.2. The null hypothesis to be tested is that the mean value equals a given value μ_0. For a given level of significance α the domain of acceptance A is characterized by

Null hypothesis H_0	mean $\mu = \mu_0$
Alternative hypothesis H_1	mean $\mu \neq \mu_0$

$$1 - \alpha = P(\bar{X} \in A) = P(\mu_0 - a \leq \bar{X} \leq \mu_0 + a)$$

where $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$ is an unbiased estimator of the true mean value μ an $S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$ is an estimator of the variance σ^2. Now use that the random variable $Z = \frac{\bar{X}-\mu}{S/\sqrt{n}}$ follows a Student-t distribution with $n-1$ degrees of freedom. Using this we compute a by `a = -tinv(alpha/2,n-1)`, i.e.

$$\frac{\alpha}{2} = \int_{-\infty}^{-a} \mathrm{pdf}(x)\, dx$$

and then the domain of acceptance is given by

$$A = [\mu_0 - a\,\frac{\sigma}{\sqrt{n}}\,,\ \mu_0 + a\,\frac{\sigma}{\sqrt{n}}]$$

The P value is characterized by

$$\begin{aligned} 1 - \alpha_{min} &= P(|\bar{X} - \mu_0| \geq |\bar{x} - \mu_0|) \\ P = \alpha_{min} &= P(\bar{X} < \mu_0 - |\bar{x} - \mu_0|) + P(\bar{X} > \mu_0 + |\bar{x} - \mu_0|) \\ &= 2\,P(\bar{X} < \mu_0 - |\bar{x} - \mu_0|) \end{aligned}$$

and thus can be computed by $P = 2{*}\mathrm{tcdf}(-|\bar{x}-\mu_0|\,\frac{\sqrt{n}}{\sigma}, n-1)$. This is easily coded in `MATLAB`/*Octave*.

As example we want to test the hypothesis that the average number of defects in the wafers for the data set introduced on page 27 is given by 14.

```
data = load('WaferDefects.txt');
mu = mean(data) % the mean of the sampling
n = length(data);
mu0 = 14;      % test against this mean value
sigma = std(data);    % assumed value of the standard deviation
alpha = 0.05; % choose level of significance

a = -tinv(alpha/2,n-1);
if abs(mu-mu0)<a*sigma/sqrt(n)   disp('H_0 not rejected, might be true')
else                             disp('H_0 rejected, probably false')
end%if
P_value = 2*tcdf(-abs(mu-mu0)*sqrt(n)/sigma,n-1)
-->
mu =  12.647
H_0 not rejected, might be true
P_value =  0.31662
```

Observe that the average μ of the sample is well below the tested value of $\mu_0 = 14$. Since the size of the sample is rather small, we do not have enough evidence to reject the hypothesis. This **does not** imply that the hypothesis is true. The computed P value is larger then the chosen level of significance $\alpha = 0.05$, which also indicated that the hypothesis can not be rejected.

With the command `ttest()` you can test this hypothesis. It will also compute the confidence interval[4] by the methods from Section 8.1.2.

```
[H, PVAL, CI] = ttest(data,mu0)
-->
H    =  0
PVAL =  0.31662
CI   =  9.8727   15.4215
```

In the documentation of `ttest` find the explanation for the results.

- Since $H = 0$ the null hypothesis not rejected, i.e. it might be true.
- Since the P value of 0.32 is larger than $\alpha = 0.05$ the null hypothesis is not rejected.
- The two–sided confidence interval $[9.8727\,,\,15.4215]$ is determined by the method in Section 8.1.2.

The default value for the level of significance is $\alpha = 0.05 = 5\%$. By providing more arguments you can used different values, e.g. `[H, PVAL, CI] = ttest(data,mu0,'alpha',0.01)` .

8.2.4 One–sided testing for the mean value μ, with unknown standard deviation σ

One can also apply one sided tests. Assume that we claim the the actual mean value μ is below a given value μ_0. For a given level of significance α the domain of acceptance A is characterized by

Null hypothesis H_0	mean $\mu \leq \mu_0$
Alternative hypothesis H_1	mean $\mu > \mu_0$

$$1 - \alpha = P(\bar{X} \in A) = P(\bar{X} \leq \mu_0 + a)$$

[4]The version of `ttest()` from the package statistics 1.2.4 in *Octave* does not produce the correct confidence interval. The bug is reported and fixed in version 1.3.0 .

Using this we compute a by `a = tinv(1-alpha,n-1)`, i.e.

$$1-\alpha = \int_{-\infty}^{a} \text{pdf}(x)\,dx \quad \text{or} \quad \alpha = \int_{a}^{\infty} \text{pdf}(x)\,dx$$

and then the domain of acceptance is given by

$$A = [-\infty\,,\ \mu_0 + a\,\frac{\sigma}{\sqrt{n}}]$$

The $P = \alpha_{min}$ value is characterized by

$$\alpha_{min} = P(\bar{X} \geq \bar{x}) = 1 - P(\bar{X} \leq \mu_0 + (\bar{x} - \mu_0))$$

and thus can be computed by $P = 1 - \text{tcdf}((\bar{x}-\mu_0)\,\frac{\sqrt{n}}{\sigma}, n-1)$. This is easily coded in `MATLAB`/*Octave*.

As example we want to test the hypothesis that the average number of defects in the wafers for the data set introduced on page 27 is smaller than 14.

```
data = load('WaferDefects.txt');
mu = mean(data)    % the mean of the sampling
n = length(data);
mu0 = 14;          % test against this mean value
sigma = std(data); % value of the standard deviation
alpha = 0.05;      % choose level of significance

a = tinv(1-alpha,n-1);
if mu < mu0+a*sigma/sqrt(n)   disp('H_0 not rejected, might be true')
else                          disp('H_0 rejected, probably false')
end%if
P_value = 1 - tcdf((mu-mu0)*sqrt(n)/sigma,n-1)
-->
mu =  12.647
H_0 not rejected, might be true
P_value =  0.84169
```

Observe that the average μ of the sample is well below the tested value of $\mu_0 = 14$. Thus the hypothesis is very likely to be correct, which is confirmed by the above result.

With the command `ttest()` you can test this hypothesis. It will also compute the one–sided confidence interval.

```
[H, PVAL, CI] = ttest (data,mu0,'tail','right')
-->
H    =  0
PVAL =  0.84169
CI   =  10.362  -Inf
```

- Since $H = 0$ the null hypothesis is not rejected, i.e. it might be true.
- Since the P value of 0.84 is larger than $\alpha = 0.05$ the null hypothesis is not rejected.
- The two–sided confidence interval $[10.362\,,\ +\infty]$ is determined by the method in Section 8.1.2.

The default value for the level of significance is $\alpha = 0.05 = 5\%$. By providing more arguments you can used different values, e.g. `[H, PVAL, CI] = ttest(data,mu0,'tail','right','alpha',0.01)` .

8.2.5 Testing the variance for normally distributed random variables

8.2.6 A two–sided test for the parameter p of a binomial distribution

By flipping a coin 1000 times you observe 475 "heads". Now there are different hypothesis that you might want to test for the parameter p, the ratio of "heads" and total number of flips.

Situation	Hypothesis	Test
"head" and "tail" are equally likely	$p = \frac{1}{2}$	two–sided
"head" is less likely than "tail"	$p \leq \frac{1}{2}$	one–sided
"head" is more likely than "tail"	$p \geq \frac{1}{2}$	one–sided

The methods and commands to be examined below will lead to statistical answers to the above questions.

Assume to have data given by a binomial distribution with parameter p, i.e. we have N data points and each point has value 1 with probability p and 0 otherwise. The null hypothesis to be tested is that the parameter p equals a given value p_0. Let $k = \sum_{i=1}^{N} x_i$ and $\bar{x} = \frac{k}{N} = \frac{1}{N} \sum_{i=1}^{N} x_i$ be the result of a sample, then $\bar{x}$ is an estimator of p. For a given level of significance α the domain

Null hypothesis H_0	$p = p_0$
Alternative hypothesis H_1	$p \neq p_0$

of acceptance A is characterized by

$$\begin{aligned} 1-\alpha &\leq P(\bar{x} \in A) = P(A_{low} \leq \bar{x} \leq A_{high}) \\ \alpha &\geq P(\bar{x} \notin A) = P(\bar{x} < A_{low}) + P(A_{high} < \bar{x}) \; = \; \mathrm{cdf}(A_{low}) + 1 - \mathrm{cdf}(A_{high}) \end{aligned}$$

where `cdf()` is the cumulative density function for the binomial distribution with parameter p_0. This condition translates to

$$\mathrm{cdf}(A_{low}) \leq \frac{\alpha}{2} \quad \text{and} \quad 1 - \mathrm{cdf}(A_{high}) \leq \frac{\alpha}{2}$$

Since the binomial distribution is a discrete distribution we can not insist on the limiting equality, but have to work with inequalities, leading to

$$\texttt{binocdf}(N \cdot A_{low}, N, p_0) \leq \frac{\alpha}{2} \quad \text{and} \quad 1 - \texttt{binocdf}(N \cdot A_{high}, N, p_0) \leq \frac{\alpha}{2}$$

Using `MATLAB`/*Octave* commands this can be solved by

$$A_{low} = \texttt{binoinv}(\frac{\alpha}{2}, N, p_0)/N \quad \text{and} \quad A_{high} = \texttt{binoinv}(1 - \frac{\alpha}{2}, N, p_0)/N$$

The null hypothesis $H_0 : p = p_0$ is accepted if

$$A_{low} \leq \frac{k}{N} \leq A_{high}$$

Since the P value is also given by the total probability of all events less likely than the observed k, we have an algorithm to determine P .

1. Compute the probability that the observed number k of "heads" shows by $p_k = \mathrm{binopdf}(k, N, p_0)$.
2. Of all $p_j = \mathrm{binopdf}(j, N, p_0)$ add those that are smaller or equal to p_k.
3. This can be packed into a few lines of code.

```
p_k = binopdf(k,n,p0);
p_all = binopdf([0:n],n,p0);
p = sum(p_all(find(p_all <= p_k)));
```

As an example consider flipping a coin 1000 times and observe 475 "heads". To test whether this coin is fair make the hypothesis $p = 0.5$ and test with a significance level of $\alpha = 0.05$.

```
N = 1000      % number of coins flipped
p0 = 0.5;     % hypothesis to be tested
alpha = 0.05; % level of sigificance
Heads = 475   % number of observed heads

A_low  = (binoinv(alpha/2,N,p0))/N;
A_high = binoinv(1-alpha/2,N,p0)/N;
DomainOfAcceptance = [A_low,A_high]
if (Heads/N >= A_low)&&(A_high >= Heads/N)  disp('H_0 not rejected, might be true')
else                                        disp('H_0 rejected, probably false')
end%if
p_k = binopdf(Heads,N,p0);  p_all = binopdf([0:N],N,p0);
P_value = sum(p_all(find(p_all <= p_k)))
-->
DomainOfAcceptance = [0.469  0.531]
H_0 not rejected, might be true
P_value =  0.12121
```

The above can be compared with the confidence interval determined in Section 8.1.4, see page 32.

```
p_low = fzero(@(p)1-binocdf(Heads-1,N,p)-alpha/2,[0,1]);
p_up  = fzero(@(p)binocdf(Heads,N,p)-alpha/2,[0,1]);
Interval_Binom = [p_low p_up]
-->
Interval_Binom = [0.44366   0.50649]
```

Since the value of $p = \frac{1}{2}$ is inside the interval of confidence we conclude that the coin might be fair. Observe that the confidence interval is built around the estimated expected value $p = 0.475$, while the domain of acceptance is built around the tested value $p_0 = \frac{1}{2}$.

The above result can be generated by the *Octave* command `binotest()`[5].

```
[h,p_val,ci] = binotest(475,1000,0.5)
-->
h = 0
p_val =  0.12121
ci =   [0.44366  0.50649]
```

With `MATLAB` the command `binofit()` determines the estimator for the parameter p and the confidence interval. Observe that the hypothesis is not tested and the P value not computed. Thus the command `binofit()` uses results from Section 8.1.4 on page 32.

[5]The code in `binotest.m` is available in the statistics package with version 1.3.0 or newer. There is now also a version of the code for `MATLAB`. Contact this author.

```
[p,ci] = binofit(475,1000,0.05)
-->
p  =  0.4750
ci =  0.4437    0.5065
```

8.2.7 One–sided test for the parameter p for a binomial distribution

The above method can also be used for one–sided tests. For a given level of significance α the

Null hypothesis H_0	$p \leq p_0$
Alternative hypothesis H_1	$p > p_0$

domain of acceptance A is characterized by

$$\begin{aligned} 1-\alpha &\leq P(\bar{x} \in A) = P(\bar{x} \leq A_{high}) \\ \alpha &\geq P(\bar{x} \notin A) = P(A_{high} < \bar{x}) \; = \; 1 - \mathrm{cdf}(A_{high}) \end{aligned}$$

This condition translates to $1-\mathrm{cdf}(A_{high}) \leq \alpha$, leading to $1-\texttt{binocdf}(N \cdot A_{high}, N, p_0) \leq \alpha$. Using `MATLAB`/*Octave* commands to can be solved by

$$A_{high} \;=\; \texttt{binoinv}(1-\alpha, N, p_0)/N$$

The null hypothesis $H_0 \,:\, p \leq p_0$ is accepted if $\frac{k}{N} \leq A_{high}$. Since the P value is defined as the smallest value of the level of significance α for which the null hypothesis is rejected we use

$$\mathrm{P} = \alpha_{min} = 1 - \texttt{binocdf}(k-1, N, p_0)$$

For the above coin flipping example we claim that the coin is less likely to show "heads" than "tail".

```
N = 1000      % number of coins flipped
p0 = 0.5;     % hypothesis to be tested
alpha = 0.05; % level of sigificance
Heads = 475   % number of observed heads

A_high = binoinv(1-alpha,N,p0)/N;
DomainOfAcceptance = [0,A_high]
if (A_high >= Heads/N)  disp('H_0 not rejected, might be true')
else                    disp('H_0 rejected, probably false')
end%if
P_value = 1-binocdf(Heads-1,N,p0)
-->
DomainOfAcceptance = [0  0.52600]
H_0 not rejected, might be true
P_value =  0.94663
```

The result states that the coin might be more likely to show "heads". The observed value of $p \approx 0.475$ is well within the domain of acceptance $A = [0\,,\, 0.526]$.

The above result can be generated by the *Octave* command `binotest`.

Null hypothesis H_0	$p \geq p_0$
Alternative hypothesis H_1	$p < p_0$

```
[h,p_val,ci] = binotest(475,1000,0.5,'tail','left')
-->
h = 0
p_val =  0.94663
ci =   [0 0.50150]
```

Obviously we can also test whether the coin is less likely to show "heads". Since the arguments are very similar to the above we just show the resulting code.

```
A_low = binoinv(alpha,N,p0)/N;
DomainOfAcceptance = [A_low,1]
if (A_low <= Heads/N)   disp('H_0 not rejected, might be true')
else                    disp('H_0 rejected, probably false')
end%if
P_value = binocdf(Heads,N,p0)
-->
DomainOfAcceptance = [0.474 1]
H_0 not rejected, might be true
P_value =  0.060607
```

The result states that the coin might be less likely to show "heads". But the observed value of $p \approx 0.475$ is barely within the domain of acceptance $A = [0.474\,,\,1]$. The P value of $P \approx 0.06$ is just above $\alpha = 5\%$. If we increase α slightly, i.e. be more tolerant towards errors of the first type, the hypothesis would be rejected.

The above result can be generated by the *Octave* command `binotest`.

```
[h,p_val,ci] = binotest(475,1000,0.5,'tail','right')
-->
h = 0
p_val =  0.060607
ci =   [0.44860  1]
```

Observe that in the previous examples for 475 "heads" on 1000 coin flips none of the three null hypothesis $p = \frac{1}{2}$, $p \leq \frac{1}{2}$ or $p \geq \frac{1}{2}$ is rejected. This is clearly illustrating that we do **not** prove that one of the hypothesis is correct. All we know is that they are not very likely to be false.

8.2.8 Testing for the parameter p for a binomial distribution for large N

If N and $N\,p_0\,(1-p_0)$ are large (e.g. $N > 30$ and $N\,p_0\,(1-p_0) > 10$) the binomial distribution of $Y = \frac{1}{N}\sum_{i=1}^{N} X_i$ with parameter p_0 can be approximated by a normal distribution with mean p_0 and standard deviation $\sigma = \sqrt{\frac{p_0\,(1-p_0)}{N}}$. Thus we can replace the binomial distribution in the above section by this normal distribution and recompute the domains of acceptance and the P values. The formulas to be used are identical to the ones in Section 8.2.2. For the confidence intervals use the tools from Section 8.1.1.

- Two–sided test with null hypothesis $H_0 \;:\; p = p_0$.

```
N = 1000      % number of coins flipped
p0 = 0.5;     % hypothesis to be tested
alpha = 0.05; % level of sigificance
Heads = 475   % number of observed heads
sigma = sqrt(p0*(1-p0)/N);  % standard deviation for p

u = -norminv(alpha/2);
DomainOfAcceptance = [p0-u*sigma,p0+u*sigma]
if (abs(Heads/N-p0)<u*sigma)  disp('H_0 not rejected, might be true')
else                          disp('H_0 rejected, probably false')
end%if
P_value = 2*normcdf(-abs(Heads/N-p0)/sigma)
-->
DomainOfAcceptance = [0.469  0.531]
H_0 not rejected, might be true
P_value =  0.11385
```

- One–sided test with null hypothesis $H_0 \ : \ p \leq p_0$.

```
u = -norminv(alpha);
DomainOfAcceptance = [0 p0+u*sigma]
if (Heads/N<p0+u*sigma)  disp('H_0 not rejected, might be true')
else                     disp('H_0 rejected, probably false')
end%if
P_value = 1-normcdf((Heads/N-p0)/sigma)
-->
DomainOfAcceptance = [0  0.52601]
H_0 not rejected, might be true
P_value =  0.94308
```

- One–sided test with null hypothesis $H_0 \ : \ p \geq p_0$.

```
u = -norminv(alpha);
DomainOfAcceptance = [p0-u*sigma 1]
if (Heads/N>p0-u*sigma)  disp('H_0 not rejected, might be true')
else                     disp('H_0 rejected, probably false')
end%if
P_value = normcdf((Heads/N-p0)/sigma)
-->
DomainOfAcceptance = [0.47399 1]
H_0 not rejected, might be true
P_value =  0.056923
```

- All of the above results are very close to the numbers obtained by the binomial distribution in Sections 8.2.6 and 8.2.7. This is no surprise, since $N = 1000$ is large enough and $N\,p_0\,(1 - p_0) = 250 \gg 10$ and thus the normal distribution is a good approximation of the binomial distribution.

Index