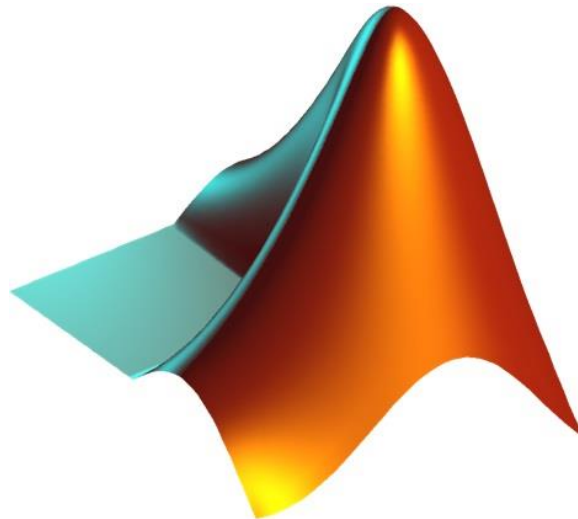




# MATLAB / Simulink Lab Course

## Optimization & Statistics Toolbox



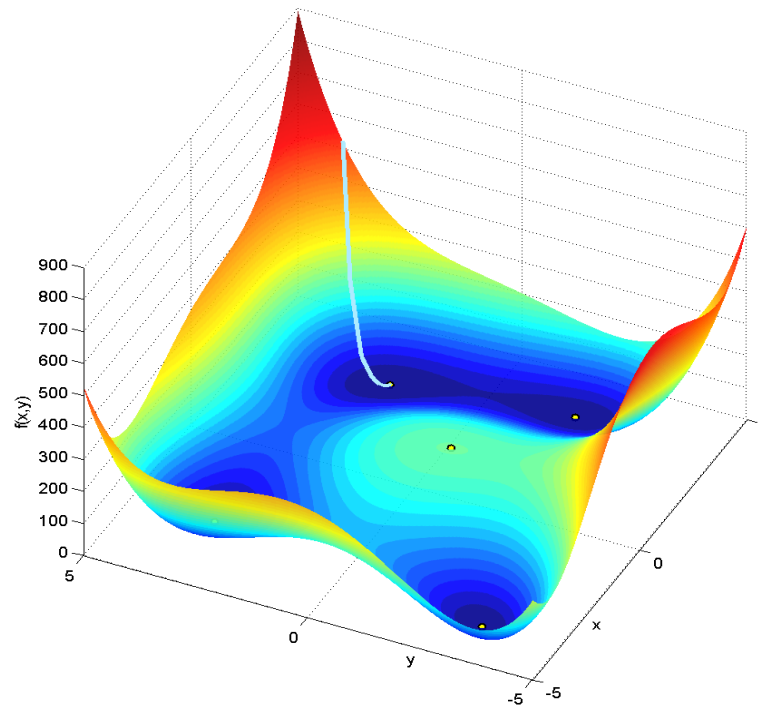
# Objectives & Preparation "Optimization & Statistics Toolbox"

- Which MathWorks products are covered?
  - ⇒ Optimization Toolbox
  - ⇒ Statistics and Machine Learning Toolbox
- What skills are learnt?
  - ⇒ Finding optimal solutions to continuous and discrete problems
  - ⇒ Incorporating constraints into an optimization problem
  - ⇒ Describe, analyzing, and modeling data
  - ⇒ Generating random numbers
- How to prepare for the session?
  - ⇒ MathWorks Webinars:
    - <https://de.mathworks.com/videos/tips-and-tricks-getting-started-using-optimization-with-matlab-81594.html>  

    - <https://de.mathworks.com/videos/data-driven-fitting-with-matlab-81809.html>  


# Outline

- Optimization Toolbox
  - Introduction
  - Linear programming
  - Quadratic programming
  - Nonlinear optimization
  - Nonlinear least squares
  - Outlook
  
- Statistics Toolbox
  - Introduction
  - Data Import
  - Exploratory Data Analysis
  - Fitting Distribution Objects
  - Generate Random Numbers
  - Hypothesis Tests
  - Parametric Regression Analysis
  
- List of Commands

# Optimization Toolbox



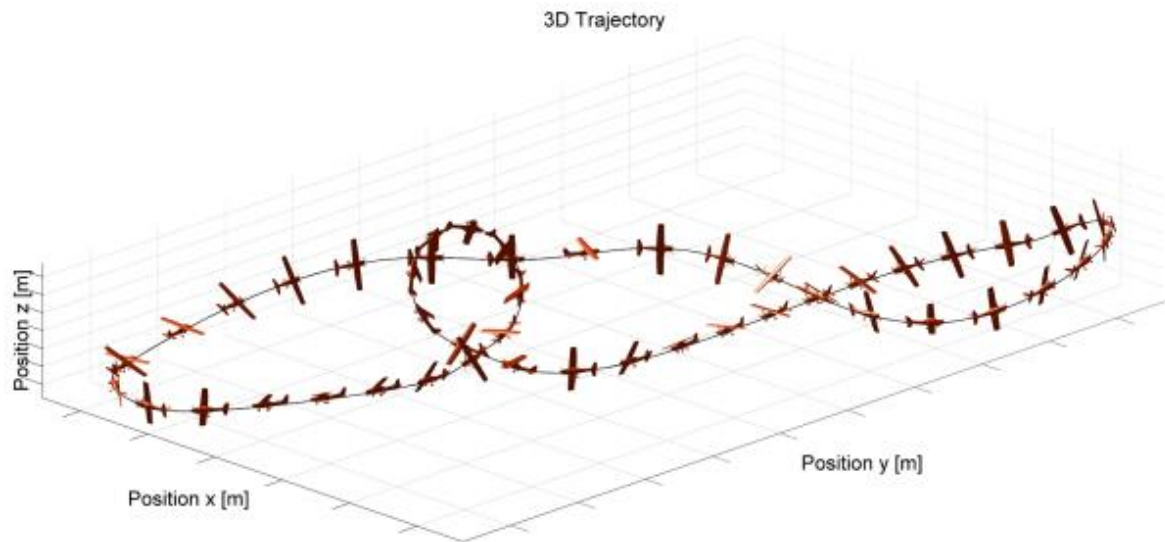
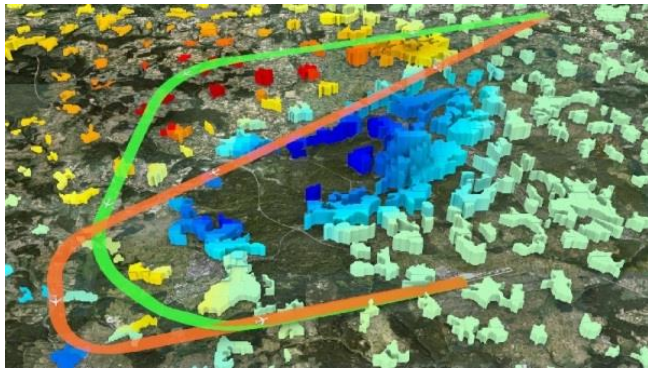
# Optimization Toolbox – Motivation

## Trajectory Optimization

How can we calculate noise minimal approaches and departure routes?

How can an air race track be designed so that it is fair for two different aircraft types?

How can fuel minimal trajectories be calculated?



## Optimization Toolbox – Introduction

- The **Optimization Toolbox** provides functions for **finding parameters** that minimize or maximize objectives while satisfying constraints.
- Available **solvers** for different kind of optimization problems: **linear programming, mixed-integer linear programming, quadratic programming, nonlinear optimization** and **nonlinear least-squares**.
- You can use these solvers to **find optimal solutions** to continuous and discrete problems, **perform tradeoff analyses**, and **incorporate optimization** methods into algorithms and applications.
- In general, optimization is an **iterative process** (compare Newton's method)
- Considering minimization problems is sufficient (**maximization** of  $f$  is minimization of  $-f$ )

Source: <http://www.mathworks.com/products/optimization/>

# Optimization Toolbox – Introduction

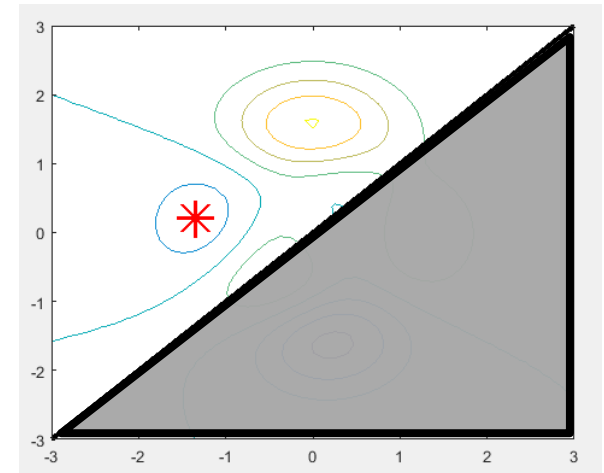
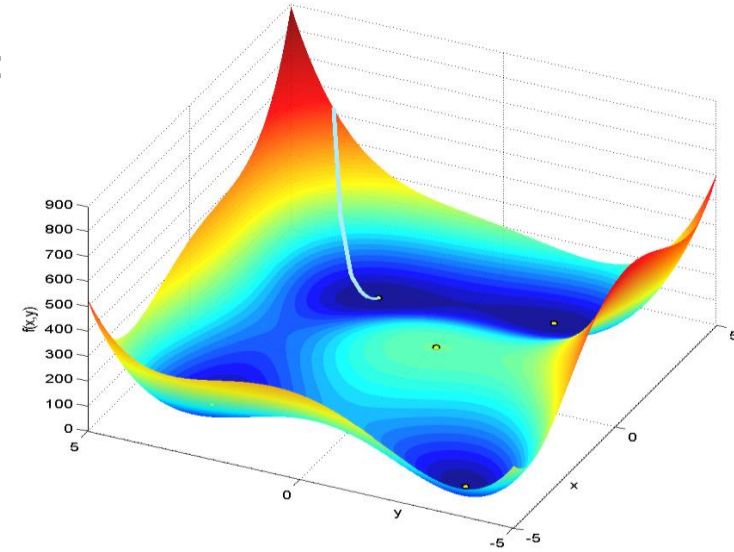
An optimization problem consists of two key components:

- Objective function (also called **cost function**)

The goal is to **find a (local) minimum** of this function.

- Constraints (optional)

The solution (i.e. local minimum) is restricted to be in a certain area of the domain.



## Optimization Toolbox – (Mixed-integer) Linear programming

- **Goal:** Find the minimum of a problem specified by

$$\min_x f^T \cdot x$$

such that

- $A \cdot x \leq b$
- $A_{eq} \cdot x = b_{eq}$
- $lb \leq x \leq ub$

where  $f$ ,  $x$ ,  $b$ ,  $b_{eq}$ ,  $lb$  and  $ub$  are vectors with suitable lengths and  $A$  and  $A_{eq}$  are matrices of suitable dimensions.

- MATLAB function: **linprog**

```
x = linprog(f,A,b,Aeq,beq,lb,ub);
```

- MATLAB function to find the **optimal solution** in the integers: **intlinprog**

```
x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub);
```

`intcon` describes which components of  $x$  are restricted to be integers.

Source: <http://www.mathworks.com/help/optim/ug/linprog.html>



# Optimization Toolbox – (Mixed-integer) Linear programming

## Example **linprog**:

$$\min_x -x_1 - x_2$$

such that

- $x_2 \leq -0.5 * x_1 + 5$
- $x_2 \leq -2 * x_1 + 10$
- $-x_1 \leq 0$
- $-x_2 \leq 0$

```
A = [0.5  1
      2    1
      -1   0
      0   -1];
```



```
b = [5 10 0 0];
```

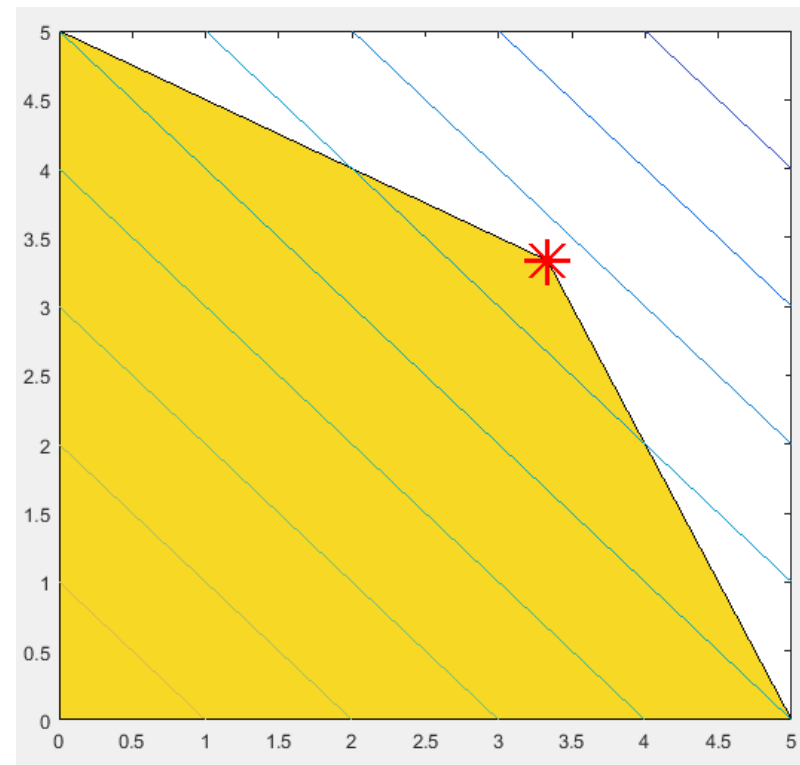
```
f = [-1 -1];
```

```
x = linprog(f,A,b)
```

x =

3.3333

3.3333



Observe that the conditions for the lower and upper bound can also be integrated in the matrix conditions with  $A$ .

## Optimization Toolbox – Quadratic programming

- **Goal:** Find the minimum of a problem specified by

$$\min_x \frac{1}{2} \cdot x^T \cdot H \cdot x + f^T \cdot x$$

such that

- $A \cdot x \leq b$
- $A_{eq} \cdot x = b_{eq}$
- $lb \leq x \leq ub$

where  $f$ ,  $x$ ,  $b$ ,  $b_{eq}$ ,  $lb$  and  $ub$  are vectors with suitable lengths and  $H$ ,  $A$  and  $A_{eq}$  are matrices of suitable dimensions.

- MATLAB function: **quadprog**

```
[x,fval] = quadprog(H,f,A,b,Aeq,beq,lb,ub);
```

The optional output variable **fval** is equal to the value of the objective function at  $x$ :

$$fval = \frac{1}{2} \cdot x^T \cdot H \cdot x + f^T \cdot x$$

Source: <http://www.mathworks.com/help/optim/ug/quadprog.html>

# Optimization Toolbox – Quadratic programming

- Example **quadprog**:

$$\min_x \quad \frac{3}{2} * x_1^2 + \frac{1}{2} x_1 * x_2 + \frac{3}{2} * x_2^2 + 6 * x_1 + 4 * x_2$$

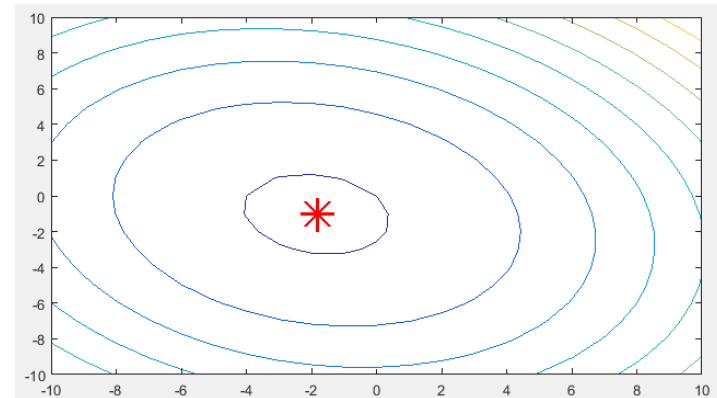
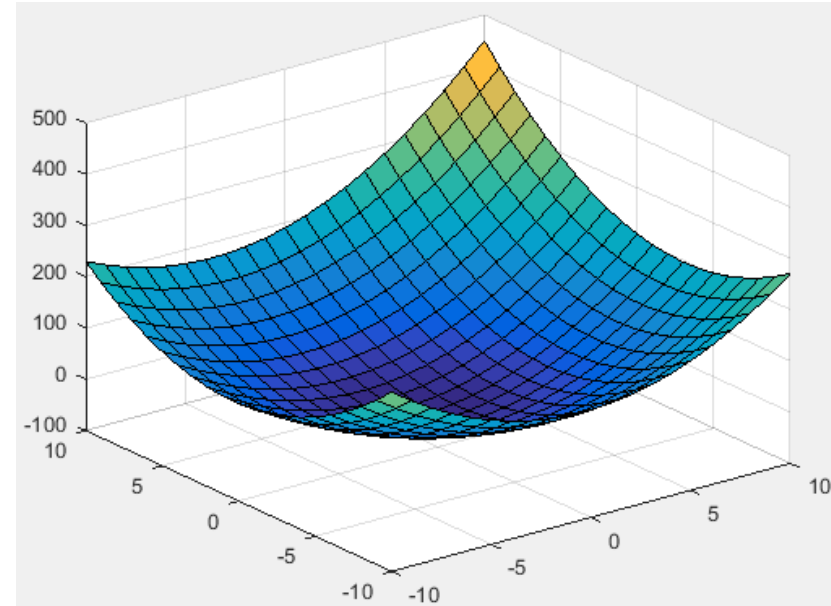
without constraints.

```
H = [3    0.5  
     0.5  3];
```

```
f = [6; 4];
```

```
x = quadprog(H,f)
```

```
x =  
-1.8286  
-1.0286
```



## Optimization Toolbox – Nonlinear optimization (unconstrained)

- **Goal:** Find the minimum of a problem specified by

$$\min_x f(x)$$

with  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ .

The problem setting is “unconstrained”, i.e. there are no restrictions for  $x$ .

As already stated, optimization is an iterative process and in the given scenario, a starting point  $x_0$  has to be predetermined.

- MATLAB function: **fminunc**

```
[x,fval] = fminunc(fun,x0,options);
```

In the optional variable `options`, further settings about the optimization (such as maximal number of iterations can be set). To obtain options use the `optimoptions` function.

Source: <http://www.mathworks.com/help/optim/ug/fminunc.html>

# Optimization Toolbox – Nonlinear optimization (unconstrained)

- Example **fminunc**:  
minimize MATLAB's Peaks function  
  
without constraints.

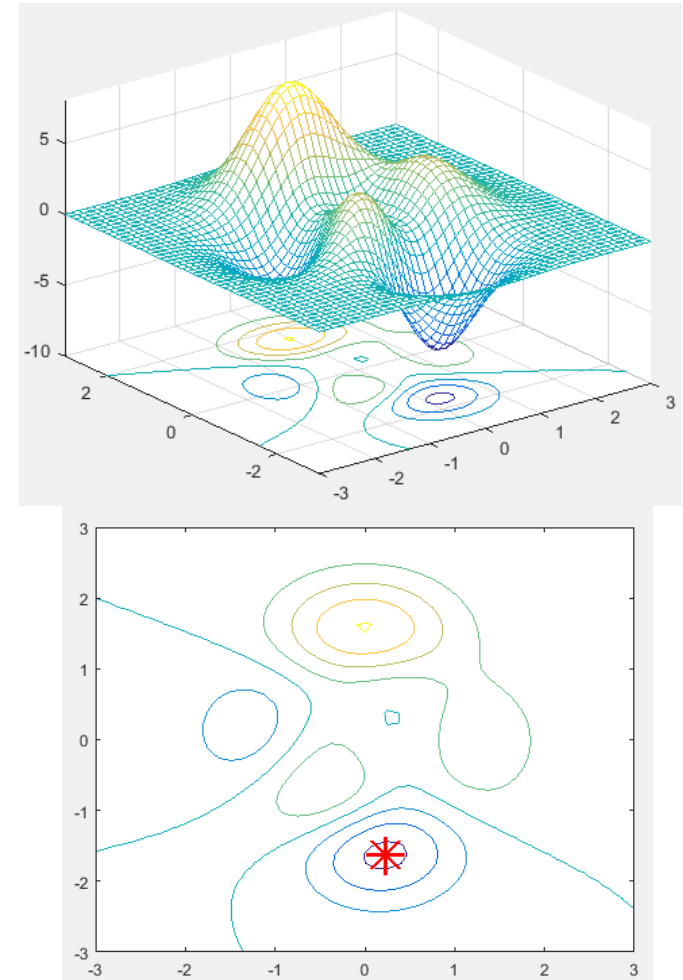
```
myPeaks = @(x) 3*(1-x(1))^2.*exp(-(x(1)^2) - (x(2)+1)^2)...  
- 10*(x(1)/5 - x(1)^3 - x(2)^5).*exp(-x(1)^2-x(2)^2)...  
- 1/3*exp(-(x(1)+1)^2 - x(2)^2);
```

```
x0 = [-1,-2];
```

```
x = fminunc(myPeaks,x0)
```

```
x =
```

```
0.2283    -1.6255
```



Source: <http://www.mathworks.com/help/matlab/ref/peaks.html>

## Optimization Toolbox – Nonlinear optimization (constrained)

- **Goal:** Find the minimum of a problem specified by

$$\min_x f(x)$$

such that

- $c(x) \leq 0$
- $c_{eq}(x) = 0$
- $A \cdot x \leq b$
- $A_{eq} \cdot x = b_{eq}$
- $lb \leq x \leq ub$

with  $f, c, c_{eq}: \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $b, b_{eq}, lb$  and  $ub$  are vectors with suitable lengths and  $A$  and  $A_{eq}$  are matrices of suitable dimensions.

- MATLAB function: **fmincon**

```
[x,fval] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options);
```

In the optional variable **nonlcon**, the functions  $c$  and  $c_{eq}$  are defined.

Source: <http://www.mathworks.com/help/optim/ug/fmincon.html>

# Optimization Toolbox – Nonlinear optimization (constrained)

- Example **fmincon**:  
minimize MATLAB's Peaks function

such that:

- $x_1 \leq x_2$

```
myPeaks = @(x) 3*(1-x(1))^2.*exp(-(x(1)^2) - (x(2)+1)^2)...
    - 10*(x(1)/5 - x(1)^3 - x(2)^5).*exp(-x(1)^2-x(2)^2)...
    - 1/3*exp(-(x(1)+1)^2 - x(2)^2);
```

```
x0 = [-3,-1];
```

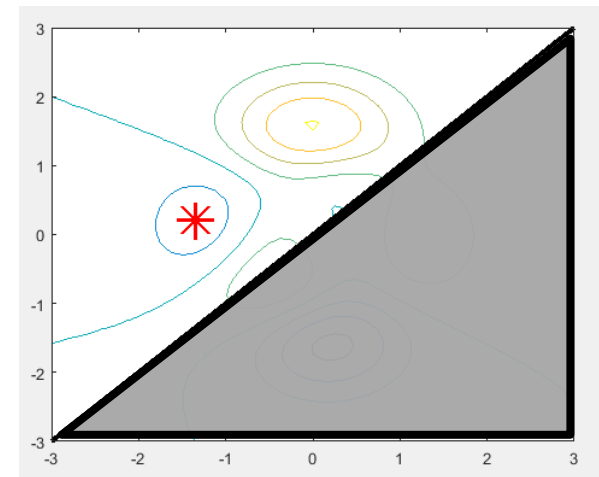
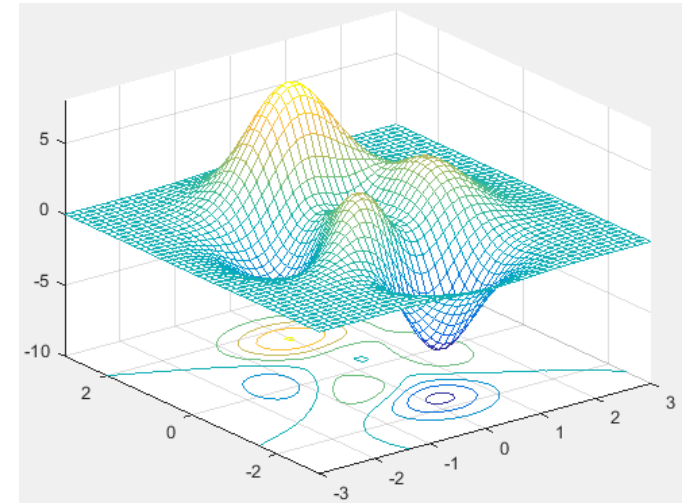
```
A = [1,-1];
```

```
b = 0;
```

```
x = fmincon(myPeaks,x0,A,b)
```

```
x =
```

```
-1.3474    0.2045
```



Source: <http://www.mathworks.com/help/matlab/ref/peaks.html>

## Optimization Toolbox – Nonlinear least-squares

- **Goal:** Find the minimum of a problem specified by

$$\min_x \|f(x)\|_2^2 = \min_x (f_1(x)^2 + \dots + f_n(x)^2)$$

with  $f: \mathbb{R}^m \rightarrow \mathbb{R}^n$ .

- MATLAB function: **lsqnonlin**

```
x = lsqnonlin(fun,x0);
```

Source: <http://www.mathworks.com/help/optim/ug/lsqnonlin.html>



# Optimization Toolbox – Nonlinear least-squares

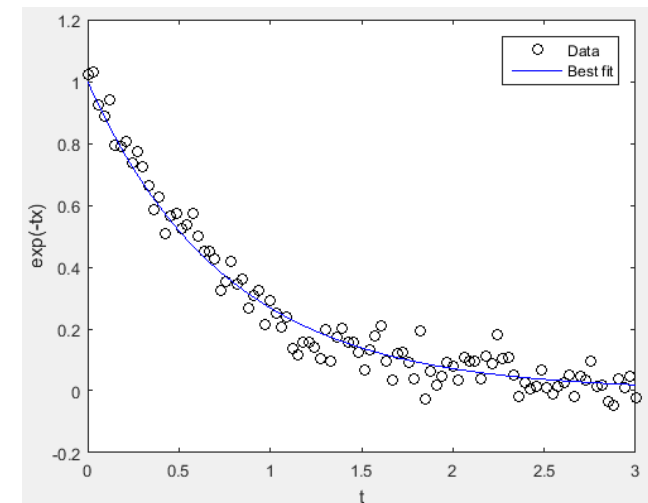
## ▪ Example **lsqnonlin**:

- Fit a simple exponential decay curve to data.
- **Goal:** Find optimal exponential decay rate based on sample data.
- Generate data from an exponential decay model plus noise. The model is:  $y = e^{-1.3*t} + \varepsilon$ .
- $t$  is ranging from 0 to 3 and  $\varepsilon$  is normally distributed noise with mean 0 and standard deviation 0.05.

```
d = linspace(0,3);  
y = exp(-1.3*d) + 0.05*randn(size(d));  
  
fun = @(r)exp(-r*d)-y;  
  
x0 = 4;  
x = lsqnonlin(fun,x0)
```

x =

1.3169

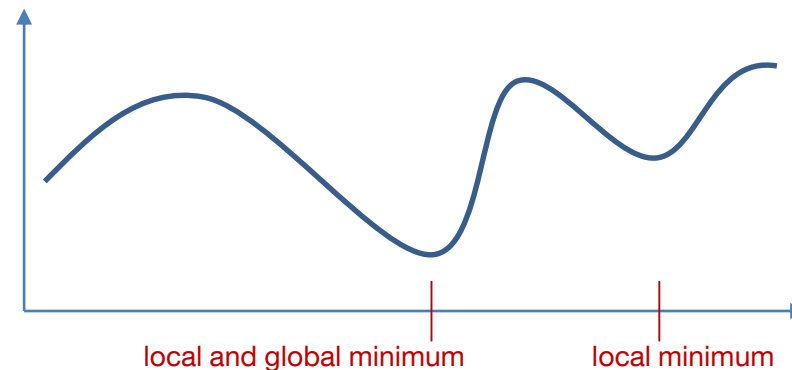


Source: <http://www.mathworks.com/help/optim/ug/lsqnonlin.html>

## Optimization Toolbox – Outlook: Global and non-smooth optimization

So far we talked about finding local minima but a local minimum is not always a **global minimum**. MATLAB offers the following concepts:

- Multistart
- Globalsearch



For **non-smooth optimization problems** (e.g. non-continuous objective function) there are the following methods:

- Pattern search
- Simulated Annealing
- Genetic Algorithm

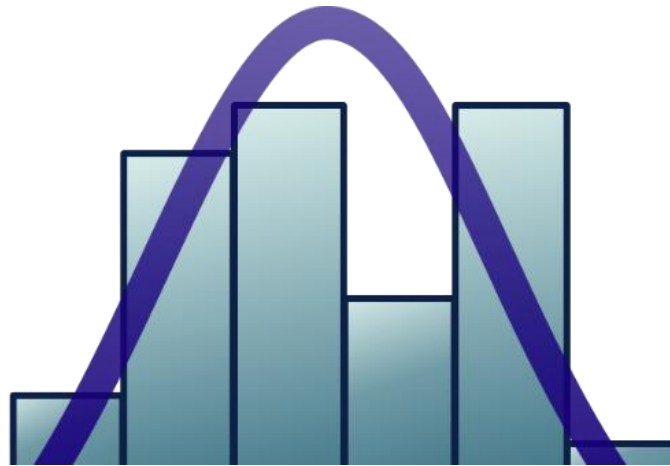
## Optimization Toolbox – Outlook: Optimization Decision Table

- To find a suitable solver for a given problem, the **Optimization Decision Table** can be useful:

Constraint Type	Objective Type				
	Linear	Quadratic	Least Squares	Smooth nonlinear	Nonsmooth
None	n/a ( $f = \text{const}$ , or $\min = -\infty$ )	quadprog, Information	\, lsqcurvefit, lsqnonlin, Information	fminsearch, fminunc, Information	fminsearch, *
Bound	linprog, Information	quadprog, Information	lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg, Information	fminbnd, fmincon, fseminf, Information	fminbnd, *
Linear	linprog, Information	quadprog, Information	lsqlin, Information	fmincon, fseminf, Information	*
General smooth	fmincon, Information	fmincon, Information	fmincon, Information	fmincon, fseminf, Information	*
Discrete	intlinprog, Information	*	*	*	*

Source: <http://www.mathworks.com/help/optim/optimization-decision-table.html>

# Statistics Toolbox



# Statistics Toolbox – Motivation

## Flight Safety

What is the **probability** of certain incidents (e.g. runway overrun) within my flight operation?

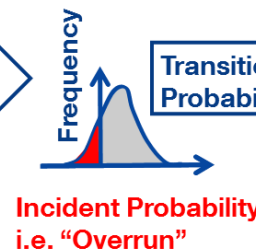
To answer this question, a lot of statistics is necessary and probability distributions have to be fitted to recorded data.

### Contributing Factors (Model Input)

Weight  
Wind  
Speed  
Flaps  
Start of Braking  
...



Model  
Output



Transition  
Probabilities

### Potential Outcomes

Outcome 1  
(e.g. hull loss)

Outcome 2

Outcome 3

⋮

Outcome n

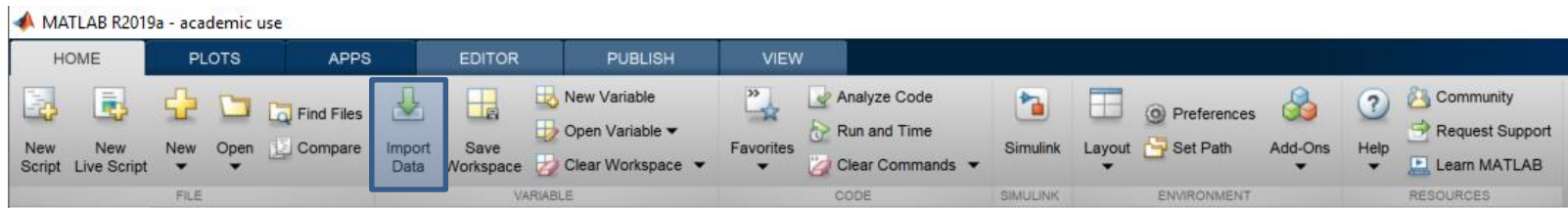
## Statistics Toolbox – Introduction

- The **Statistics and Machine Learning Toolbox** provides functions and apps to describe, analyze, and model data using statistics and machine learning.
- You can use descriptive statistics and plots for **exploratory data analysis**, **fit probability distributions to data**, **generate random numbers** for **Monte Carlo simulations**, and perform **hypothesis tests**.
- **Regression and classification algorithms** let you draw **inferences** from data and build predictive models.

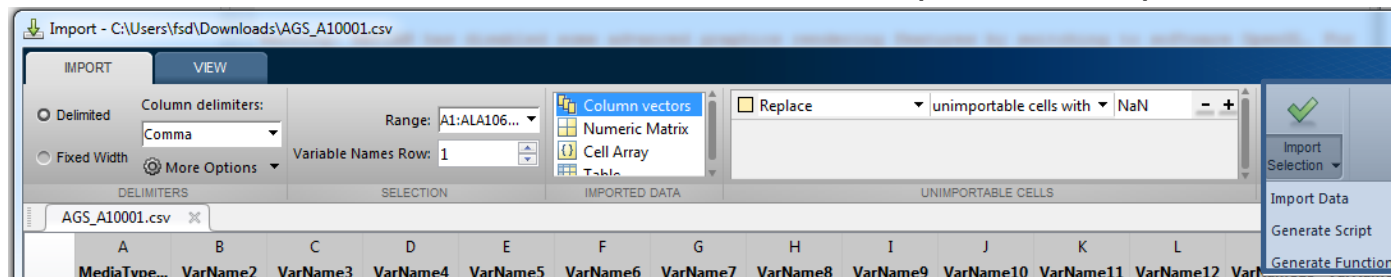
Source: <http://www.mathworks.com/products/statistics/>

## Statistics Toolbox – Data Import

- Every statistical analysis works with data which has to be imported into MATLAB first.
- A very convenient way to import data is the “**Import Data**” button in the “HOME” tab.



- After choosing the file, MATLAB looks for the suitable commands automatically.
- The chosen commands can be considered with the help of the “Import Selection”.



- Alternatively, the right command for the specific file can be applied directly:  
**textscan, readtable, xlsread, csvread, dlmread,...**

## Statistics Toolbox – Exploratory Data Analysis

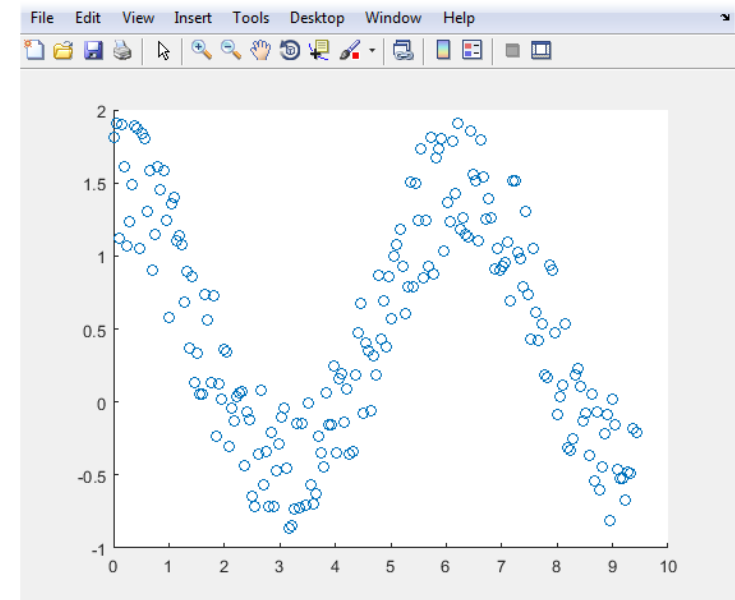
- **Scatterplots** are a very basic tool exploratory data analysis, i.e. graphical representation of data

```
scatter(x,y)
```

- Example:

```
x = linspace(0,3*pi,200);  
y = cos(x) + rand(1,200);  
scatter(x,y)
```

- `linspace(0,3*pi,200)` returns a row vector of 200 evenly spaced points between 0 and  $3\pi$
- `rand(1,200)` returns a row vector of 200 single uniformly distributed random numbers in the interval (0,1)

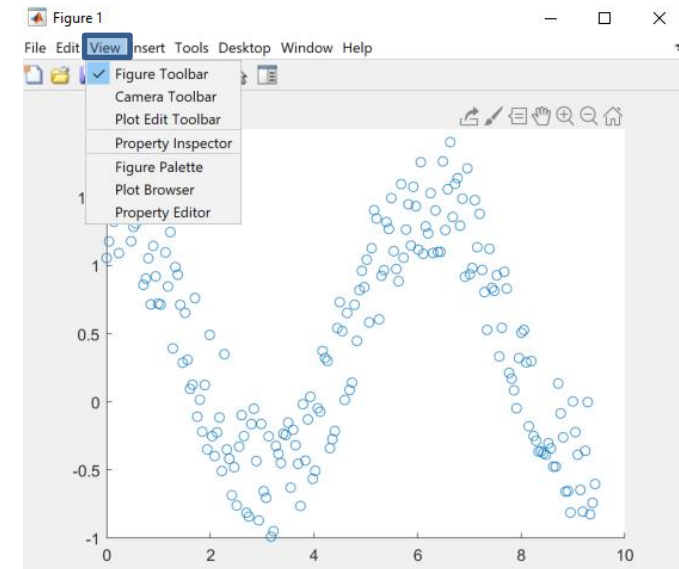
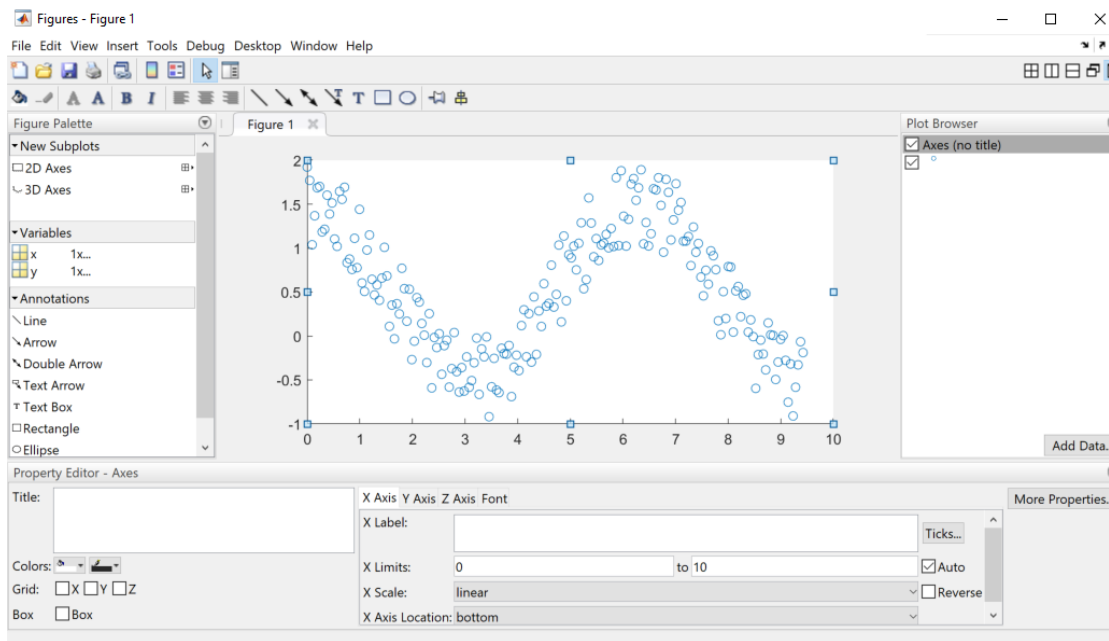


Source: <http://www.mathworks.com/help/matlab/ref/scatter.html>



# Statistics Toolbox – Exploratory Data Analysis

- **Changing properties** of the plot is easily possible:  
Hit “**View**” to show plot tools and dock figures
- Label, colors and further options can be set



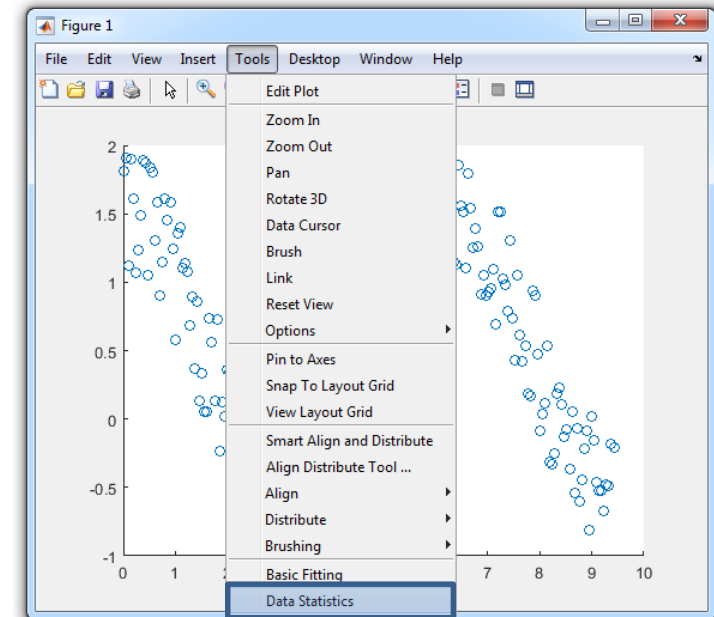
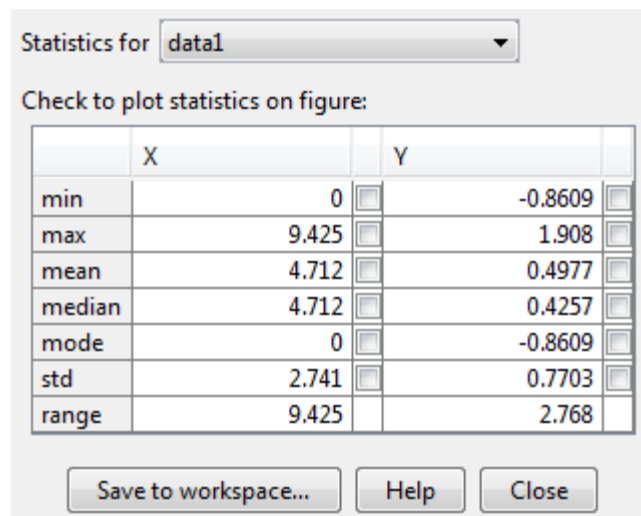
- After finishing the modifications, the used MATLAB code can be observed:  
**File → Generate Code**  
Alternatively the functions to adapt the plot (**axes**, **xlabel**,...) can be used directly.

## Statistics Toolbox – Exploratory Data Analysis

- To get a **basic statistical overview** of the considered data, the “**Data Statistics**” command is useful.

- The occurring windows gives the overview for the individual dimensions. The following values are clear:

- min
- max
- range



- mean**, **median**, **mode** and **std** are described in the following.

## Statistics Toolbox – Exploratory Data Analysis

- **mean** calculates the average or mean value of arrays.

```
M = mean(A);
```

If  $A = (a_1, \dots, a_n)$  is a vector, then  $M$  is the mean of its elements:

$$M = \frac{1}{n} \sum_{i=1}^n a_i$$

It is also possible to calculate the mean for matrices and for multidimensional arrays (see documentation).

Source: <http://www.mathworks.com/help/matlab/ref/mean.html>

- **median** returns a number separating the higher half of a data sample from the lower half. If there is an even number of observations, then the median is defined as the mean of the two middle values.

```
M = median(A);
```

If  $A = (a_0, \dots, a_n)$  is a (sorted!) vector, then  $M$  is the median of its elements:

$$M = \begin{cases} a_{n/2} & \text{for } n \text{ even} \\ \frac{a_{n-1/2} + a_{n+1/2}}{2} & \text{for } n \text{ odd} \end{cases}$$

Versions for matrices and multidimensional arrays are again available (see documentation).

Source: <http://www.mathworks.com/help/matlab/ref/median.html>

## Statistics Toolbox – Exploratory Data Analysis

- **mode** returns the **most frequent value** in an array.

```
M = mode(A);
```

It is also possible to calculate the mode for matrices and for multidimensional arrays (see documentation).

Source: <http://www.mathworks.com/help/matlab/ref/mode.html>

- **std** returns the **standard deviation** of an array, which is a measure of its dispersion or spread.

```
S = std(A);
```

If  $A = (a_1, \dots, a_n)$  is a vector, then the (empirical) standard deviation is defined as:

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (a_i - \mathbf{mean}(a))^2}$$

Versions for matrices and multidimensional arrays are again available (see documentation).

<http://www.mathworks.com/help/matlab/ref/std.html>

- **var** calculates the **variance** of an array and is also widely used in statistics.  $\text{var} = \text{std}^2$

```
V = var(A);
```

<http://www.mathworks.com/help/matlab/ref/var.html>

## Statistics Toolbox – Exploratory Data Analysis

- **cov** calculates the **covariance** matrix of / between arrays.

```
C = cov(A,B);
```

If  $A = (a_1, \dots, a_n)$ ,  $B = (b_1, \dots, b_n)$  are vectors, then  $C$  is the covariance matrix of  $A$  and  $B$ .

This is the following 2x2 matrix:

$$C = \begin{pmatrix} \frac{1}{n-1} \sum_{i=1}^n (a_i - \mathbf{mean}(A)) \cdot (a_i - \mathbf{mean}(A)) & \frac{1}{n-1} \sum_{i=1}^n (b_i - \mathbf{mean}(B)) \cdot (a_i - \mathbf{mean}(A)) \\ \frac{1}{n-1} \sum_{i=1}^n (a_i - \mathbf{mean}(A)) \cdot (b_i - \mathbf{mean}(B)) & \frac{1}{n-1} \sum_{i=1}^n (b_i - \mathbf{mean}(B)) \cdot (b_i - \mathbf{mean}(B)) \end{pmatrix}$$

Observe that this matrix is symmetric along its main diagonal. Furthermore,  $C(1,1) = \mathbf{var}(A)$  and  $C(2,2) = \mathbf{var}(B)$ .

It is also possible to calculate the covariance for matrices and further objects (see documentation). For a detailed ANalysis Of Variance and Covariance the so called **ANOVA methods** can be useful.

Source: <http://www.mathworks.com/help/matlab/ref/cov.html>, <http://www.mathworks.com/help/stats/analysis-of-variance-and-covariance.html>

## Statistics Toolbox – Exploratory Data Analysis

- **corrcoef** calculates the **correlation coefficients** of / between arrays.

```
R = corrcoef(A,B);
```

If  $A = (a_1, \dots, a_n)$ ,  $B = (b_1, \dots, b_n)$  are vectors, then  $R$  is the correlation matrix of  $A$  and  $B$ .

$$R = \begin{pmatrix} \frac{\frac{1}{n-1} \sum_{i=1}^n (a_i - \text{mean}(A)) \cdot (a_i - \text{mean}(A))}{\text{std}(A) \cdot \text{std}(A)} & \frac{\frac{1}{n-1} \sum_{i=1}^n (b_i - \text{mean}(B)) \cdot (a_i - \text{mean}(A))}{\text{std}(B) \cdot \text{std}(A)} \\ \frac{\frac{1}{n-1} \sum_{i=1}^n (a_i - \text{mean}(A)) \cdot (b_i - \text{mean}(B))}{\text{std}(A) \cdot \text{std}(B)} & \frac{\frac{1}{n-1} \sum_{i=1}^n (b_i - \text{mean}(B)) \cdot (b_i - \text{mean}(B))}{\text{std}(B) \cdot \text{std}(B)} \end{pmatrix}$$

Observe that this matrix is again symmetric along its main diagonal. Furthermore,  $R(1,1) = R(2,2) = 1$ .

It is also possible to calculate the covariance for matrices and further objects (see documentation).

Source: <http://www.mathworks.com/help/matlab/ref/corrcoef.html>

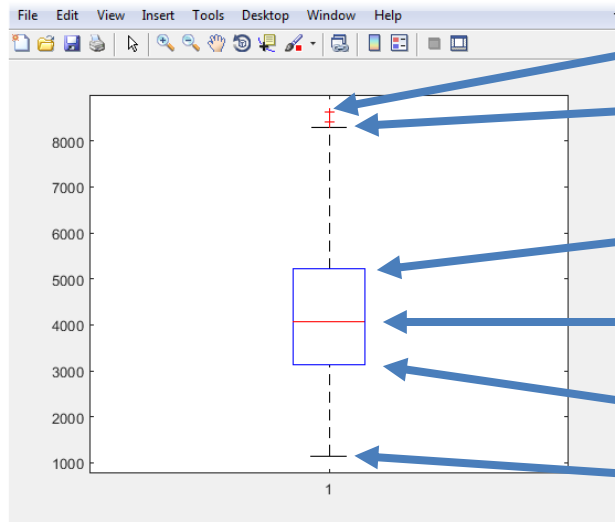
## Statistics Toolbox – Exploratory Data Analysis

- **boxplot** is a further useful and widely used tool.

```
boxplot(X);
```

If X is a matrix, there is one box per column; if X is a vector, there is just one box.

```
load cities.mat  
boxplot(ratings(:,5))
```



single data points, “outliers”

upper “whisker”, by default this is 75 % quantile +  $1.5 * (75 \% \text{ quantile} - 25 \% \text{ quantile})$

75 % quantile, i.e. 75 % of the data below that value

median (= 50 % quantile)

25 % quantile

lower “whisker”

Source: <http://www.mathworks.com/help/stats/boxplot.html>

# Statistics Toolbox – Exploratory Data Analysis

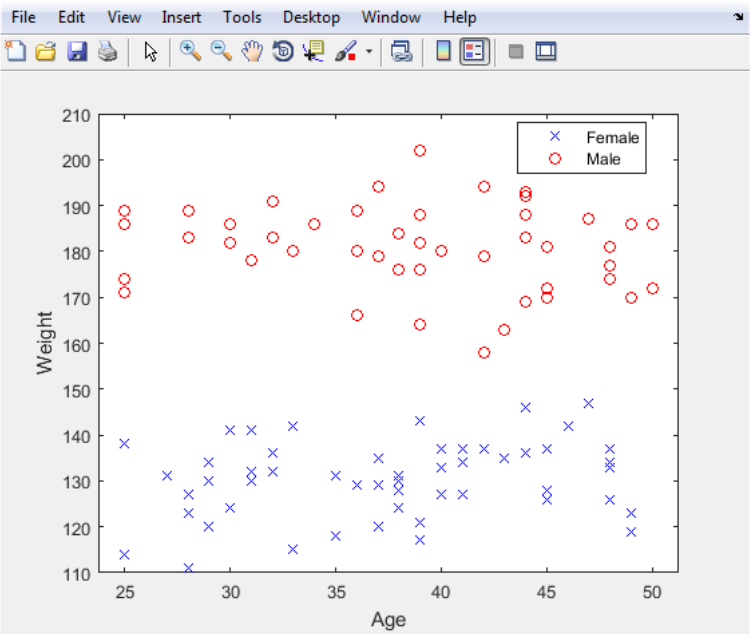
- **grpstats** and **gscatter** are tools to analyze data categorized in different groups.

```
load hospital.mat
ds = hospital(:,{'Sex','Age','Weight','Smoker'});
statarray = grpstats(ds,'Sex')
```

statarray =

	Sex	GroupCount	mean_Age	mean_Weight	mean_Smoker
Female	Female	53	37.717	130.47	0.24528
Male	Male	47	38.915	180.53	0.44681

```
figure;
gscatter(ds.Age,ds.Weight,ds.Sex,'br','xo')
xlabel('Age');
ylabel('Weight');
```

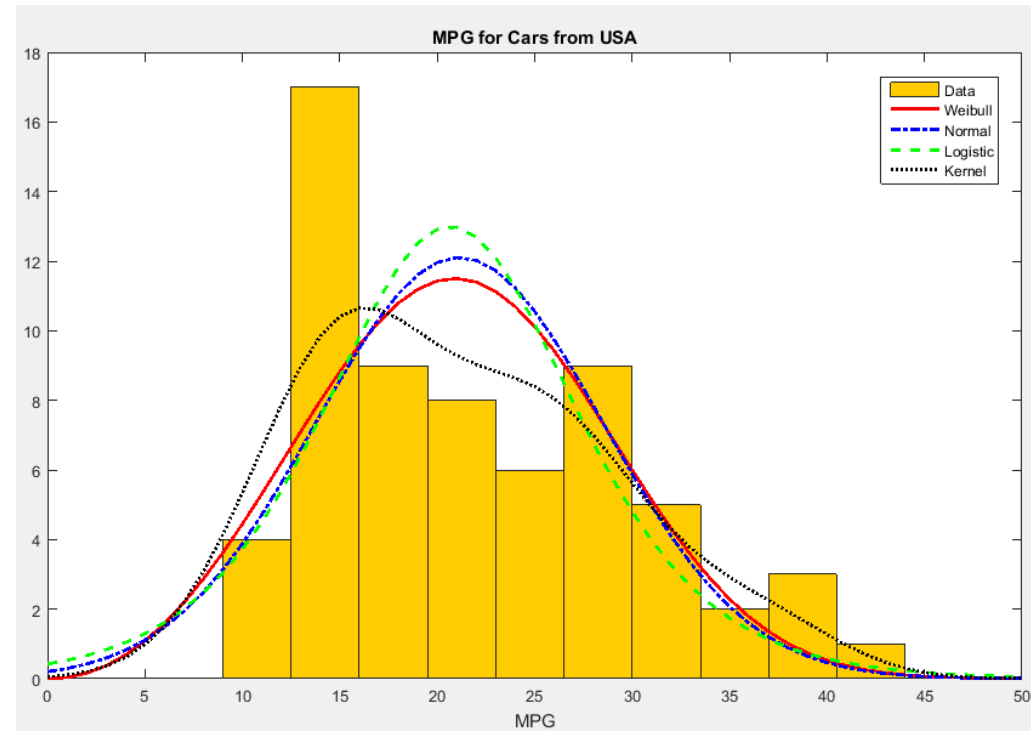


Source: <https://de.mathworks.com/help/stats/gscatter.html>



## Statistics Toolbox – Fitting Distribution Objects

- Fit probability distributions to sample data
- Evaluate probability functions such as pdf and cdf
- Calculate summary statistics such as mean and median
- Visualize sample data
- Generate random numbers
- To generate MATLAB Distribution Objects **makedist** and **fitdist** can be used



Source: <http://www.mathworks.com/help/stats/probability-distributions-1.html>,  
<http://www.mathworks.com/help/stats/compare-multiple-distribution-fits.html>

# Statistics Toolbox – Fitting Distribution Objects

- **makedist** can be used to create probability distribution objects.

```
pd = makedist(distname,Name,Value);
```

- This command creates a probability distribution object for the **distribution** distname with one or more distribution parameter values specified by the name-value pair arguments Name,Value.
- Available distributions can be seen in the table on the right.
- The command can be used to generate both, **discrete** and **continuous** distributions.
- The output pd is a MATLAB **class object** for the specific distribution, e.g. prob.BetaDistribution class for the Beta distribution.

Source: <http://www.mathworks.com/help/stats/makedist.html>

Distribution Name	Description
'Beta'	Beta distribution
'Binomial'	Binomial distribution
'BirnbaumSaunders'	Birnbaum-Saunders distribution
'Burr'	Burr distribution
'Exponential'	Exponential distribution
'ExtremeValue'	Extreme Value distribution
'Gamma'	Gamma distribution
'GeneralizedExtremeValue'	Generalized Extreme Value distribution
'GeneralizedPareto'	Generalized Pareto distribution
'InverseGaussian'	Inverse Gaussian distribution
'Logistic'	Logistic distribution
'Loglogistic'	Loglogistic distribution
'Lognormal'	Lognormal distribution
'Multinomial'	Multinomial distribution
'Nakagami'	Nakagami distribution
'NegativeBinomial'	Negative Binomial distribution
'Normal'	Normal distribution
'PiecewiseLinear'	Piecewise Linear distribution
'Poisson'	Poisson distribution
'Rayleigh'	Rayleigh distribution
'Rician'	Rician distribution
'tLocationScale'	t Location-Scale distribution
'Triangular'	Triangular distribution
'Uniform'	Uniform distribution
'Weibull'	Weibull distribution

## Statistics Toolbox – Fitting Distribution Objects

- **fitdist** fits a probability distribution object to data.

```
pd = fitdist(x,distname,Name,Value);
```

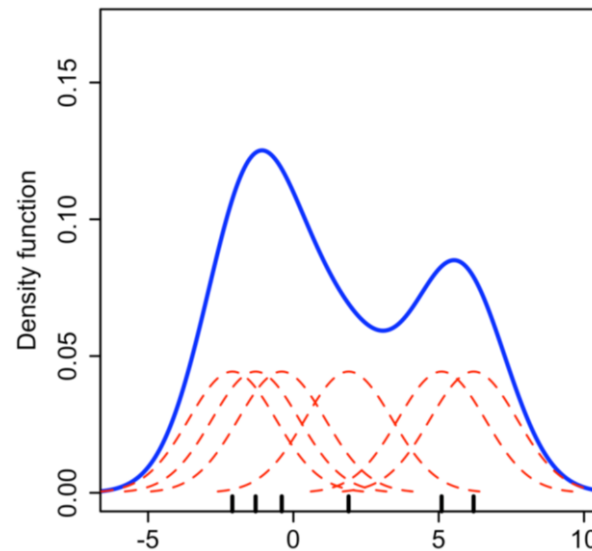
- This command creates a probability distribution object `pd` by fitting the distribution specified by `distname` to the data in column vector `x`.
- Additional options can be specified by one or more name-value pair arguments `Name,Value`. For example, you can indicate censored data or specify control parameters for the iterative fitting algorithm.
- For a given distribution object, the following commands are very useful:
  - `cdf`
  - `icdf`
  - `iqr`
  - `median`
  - `pdf`
  - `random`
  - `truncate`
  - `mean`
  - `negloglik`
  - `paramci`
  - `proflik`
  - `std`
  - `var`

Source: <http://www.mathworks.com/help/stats/fitdist.html>

## Statistics Toolbox – Fitting Distribution Objects

- **fitdist** used for **Kernel Density Estimation (KDE)**.
- KDE is a non-parametric way to estimate the probability distribution. Thereby, a so called kernel function is **centered** at every data point. Subsequently, all the collected kernel functions are summed up and normalized suitably.

```
pd = fitdist(x, 'Kernel');
```



Source: <http://www.mathworks.com/help/stats/fitdist.html>, [https://en.wikipedia.org/wiki/Kernel\\_density\\_estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation)

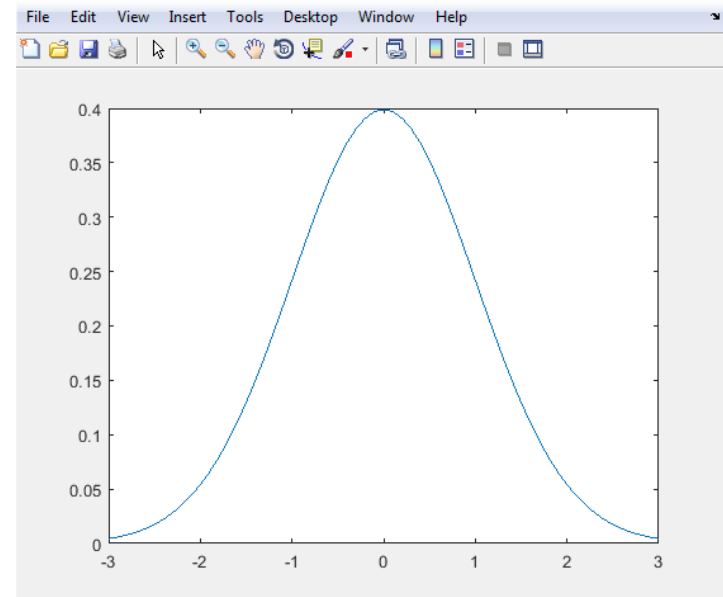
## Statistics Toolbox – Fitting Distribution Objects

- **pdf** returns the probability density function at the specific value.

```
y = pdf(pd,x);
```

- The pdf describes the relative likelihood for the associated random variable to take at a given value.

```
pd = makedist('Normal',0,1);  
x = -3:0.1:3;  
y = pdf(pd,x);  
plot(x,y)
```



Source: <https://de.mathworks.com/help/stats/prob.normaldistribution.pdf.html>

## Statistics Toolbox – Fitting Distribution Objects

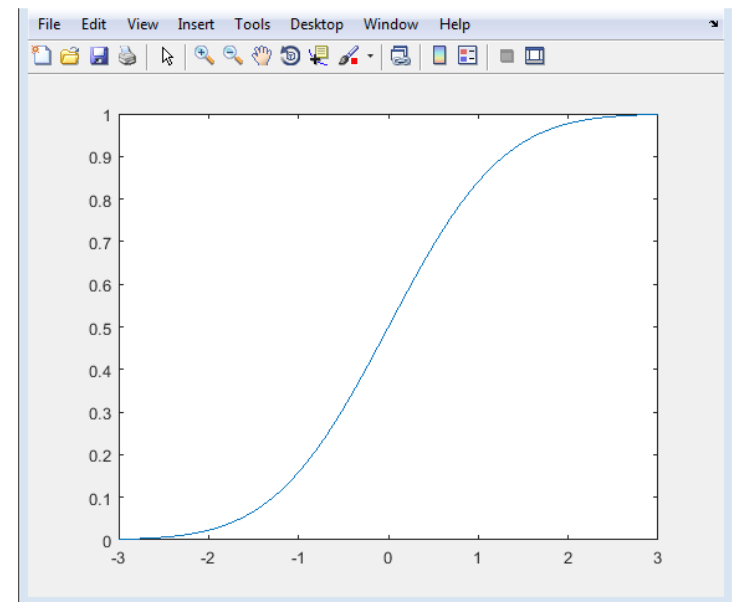
- **cdf** returns the cumulative distribution function at the specific value and is closely related to the pdf.

```
y = cdf(pd,x);
```

- The **cdf** describes the probability that values less or equal the given value occur.

$$\text{cdf}(x) = \int_{-\infty}^x \text{pdf}(z) dz$$

```
pd = makedist('Normal',0,1);  
x = -3:0.1:3;  
y = cdf(pd,x);  
plot(x,y)
```



Source: <https://de.mathworks.com/help/stats/prob.normaldistribution.cdf.html>

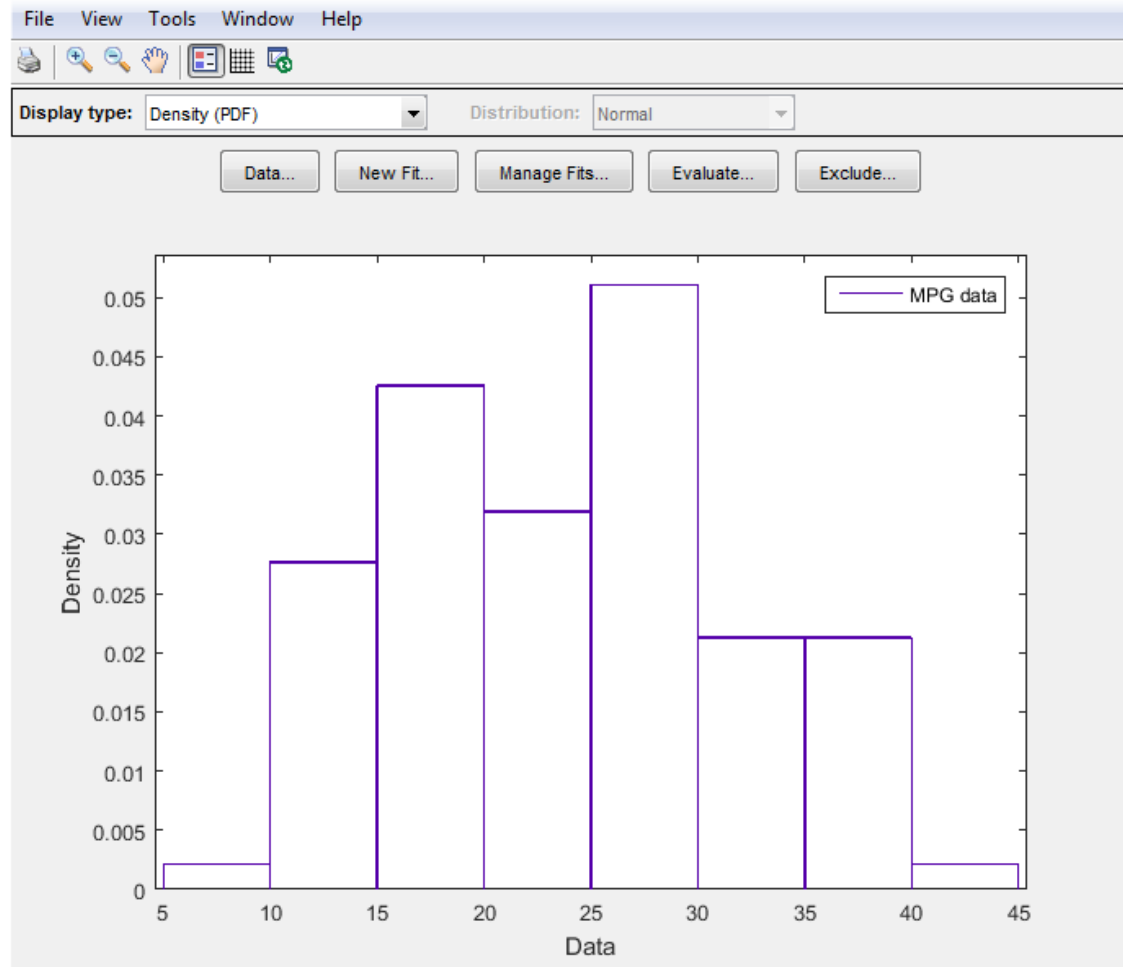
## Statistics Toolbox – Fitting Distribution Objects

- The **Distribution Fitting Tool** is a nice tool to fit distributions to data

dfittool

- This command opens the Distribution Fitting app

Source:  
<http://www.mathworks.com/help/stats/dfittool.html>



## Statistics Toolbox – Generate Random Numbers

- In MATLAB, there are various methods to generate random numbers.
- **random** returns a random number  $Y$  from the distribution specified by the probability distribution object  $pd$ .

```
Y = random(pd);
```

- Applying the `cdf` function always transfers the random numbers of any distribution to random numbers of the uniform distribution on the interval  $[0,1]$ .

```
pd = makedist('Normal',0,1);  
Y_normal = random(pd)  
Y_uniform = cdf(pd,Y_normal)
```

```
Y_normal =  
-0.7873
```

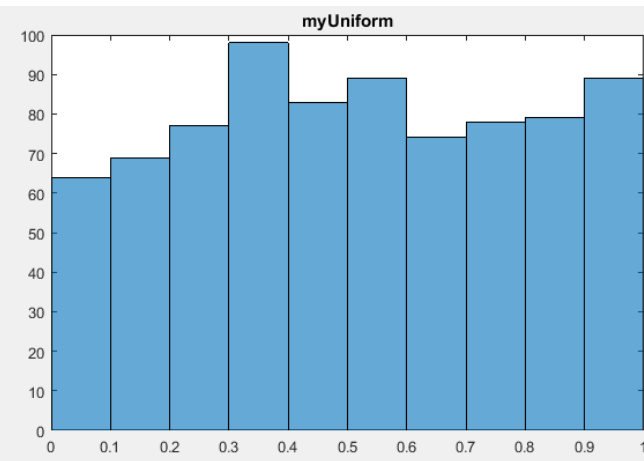
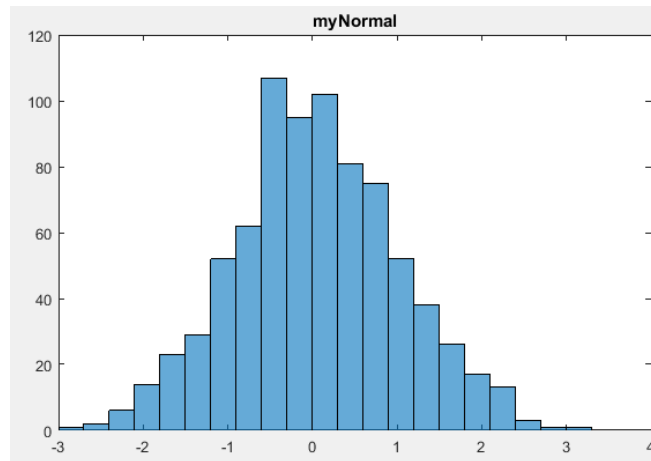
```
Y_uniform =  
0.2156
```

Source: <https://de.mathworks.com/help/stats/prob.normaldistribution.random.html>



## Statistics Toolbox – Generate Random Numbers

```
pd = makedist('Normal',0,1);  
myNormal = random(pd,800,1);  
myUniform = cdf(pd,myNormal);  
subplot(1,2,1);  
histogram(myNormal);  
title('myNormal')  
subplot(1,2,2);  
histogram(myUniform);  
title('myUniform')
```



## Statistics Toolbox – Hypothesis Tests

- A statistical **hypothesis test** is a tool to verify certain statistical assumptions.
- A **null hypothesis**  $H_0$  is compared to the statistical **alternative**  $H_1$ .
- Statistical hypothesis tests are performed with respect to a **significance level** which can be considered as a threshold probability.
- The **result of the test** is given by accepting or rejecting the null hypothesis for a pre-specified significance level.
- Various hypothesis tests for different situations are available in MATLAB:
  - One-sample and paired-sample t-test**
  - z-test**
  - Lilliefors test**
  - One-sample Kolmogorov-Smirnov test**

Source: <http://www.mathworks.com/help/stats/hypothesis-testing.html>

## Statistics Toolbox – Hypothesis Tests

### ▪ One-sample Kolmogorov-Smirnov test

```
[h,p] = kstest(x,Name,Value);
```

- The command returns a test decision for the **null hypothesis** that the data in vector  $x$  comes from a standard normal distribution, against the **alternative** that it does not come from such a distribution, using the one-sample Kolmogorov-Smirnov test. The result  $h$  is 1 if the test rejects the null hypothesis at the 5% significance level, or 0 otherwise.
- With the `Name,Value` arguments, **additional options** specified by one or more name-value pair arguments can be given. For example, you can test for a distribution other than standard normal, change the significance level, or conduct a one-sided test.
- An optional output is the  **$p$ -value**. It indicates the probability under the null hypothesis of observing a value as extreme or more extreme of the  $ks$ -statistic computed from the sample.

Source: <http://www.mathworks.com/help/stats/hypothesis-testing.html>, <http://www.mathworks.com/help/stats/kstest.html>

## Statistics Toolbox – Hypothesis Tests

```
pdNormal = makedist('Normal');  
samplesNormal = random(pdNormal,500,1);  
  
pdEV = makedist('ExtremeValue', -1, 2);  
samplesEV = random(pdEV,500,1);  
  
result1 = kstest(samplesNormal, 'CDF',pdNormal)  
result2 = kstest(samplesNormal, 'CDF',pdEV)  
result3 = kstest(samplesEV, 'CDF',pdNormal)  
result4 = kstest(samplesEV, 'CDF',pdEV)
```

```
result1 = 0
```

```
result2 = 1
```

```
result3 = 1
```

```
result4 = 0
```

## Statistics Toolbox – Parametric Regression Analysis

- Regression is the process of **fitting models to data**.
- If a model is parametric, **regression estimates the parameters** from the data.
- For a **linear model**, estimation of the model parameters is based on linear algebra.
- For a **nonlinear model**, estimation is based on search methods from optimization that minimize the norm of a residual vector.
- Models of data with categorical response (e.g. “Safe Flights”, “Critical Flights”) are called **classifiers**. The related problem is called “Parametric Classification”.
- Depending on the regression model and on the result of interest, several methods and algorithms are available in MATLAB. They are summarized on the next slide.

Source: <http://www.mathworks.com/help/stats/introduction-to-parametric-regression-analysis.html>,  
<http://www.mathworks.com/help/stats/introduction-to-parametric-classification.html>

# Statistics Toolbox – Parametric Regression Analysis

## Choose a Regression Function

You have:	You want:	Use this:
Continuous or categorical predictors, continuous response, linear model	Fitted model coefficients	<code>fitlm</code> . See <a href="#">Linear Regression</a> .
Continuous or categorical predictors, continuous response, linear model of unknown complexity	Fitted model and fitted coefficients	<code>stepwiselm</code> . See <a href="#">Stepwise Regression</a> .
Continuous or categorical predictors, response possibly with restrictions such as nonnegative or integer-valued, generalized linear model	Fitted generalized linear model coefficients	<code>fitglm</code> or <code>stepwiseglm</code> . See <a href="#">Generalized Linear Models</a> .
Continuous predictors with a continuous nonlinear response, parametrized nonlinear model	Fitted nonlinear model coefficients	<code>fitnlm</code> . See <a href="#">Nonlinear Regression</a> .
Continuous predictors, continuous response, linear model	Set of models from ridge, lasso, or elastic net regression	<code>lasso</code> or <code>ridge</code> See <a href="#">Lasso and Elastic Net or Ridge Regression</a> .
Correlated continuous predictors, continuous response, linear model	Fitted model and fitted coefficients	<code>plsregress</code> See <a href="#">Partial Least Squares</a> .
Continuous or categorical predictors, continuous response, unknown model	Nonparametric model	<code>fitrtree</code> or <code>fitensemble</code> See <a href="#">Decision Trees or Ensemble Methods</a> .
Categorical predictors only	ANOVA	<code>anova</code> , <code>anova1</code> , <code>anova2</code> , <code>anovan</code>
Continuous predictors, multivariable response, linear model	Fitted multivariate regression model coefficients	<code>mvregress</code>
Continuous predictors, continuous response, mixed-effects model	Fitted mixed-effects model coefficients	<code>nlmefit</code> or <code>nlmefitsa</code> See <a href="#">Mixed-Effects Models</a> .

Source: <http://www.mathworks.com/help/stats/introduction-to-parametric-regression-analysis.html>,  
<http://www.mathworks.com/help/stats/introduction-to-parametric-classification.html>

# Statistics Toolbox – Parametric Regression Analysis

## Example: Fitting a linear model using `fitlm`

- **Goal:** Fit a linear model to the following data (only the first 20 of 100 lines illustrated):

- In the sample data set “carsmall”, the following data are given
  - Weight
  - Acceleration
  - MPG (Miles per Gallon)

- Considered model structure:  
$$\text{MPG} = C_1 + C_2 * \text{Weight} + C_3 * \text{Acceleration}$$
- Unknown parameters:  $C_1, C_2, C_3$

ans =

Weight	Acceleration	MPG
3504	12	18
3693	11.5	15
3436	11	18
3433	12	16
3449	10.5	17
4341	10	15
4354	9	14
4312	8.5	14
4425	10	14
3850	8.5	15
3090	17.5	NaN
4142	11.5	NaN
4034	11	NaN
4166	10.5	NaN
3850	11	NaN
3563	10	15
3609	8	14
3353	8	NaN
3761	9.5	15
3086	10	14

```
load carsmall

tbl = table(Weight,Acceleration,MPG,'VariableNames',...
            {'Weight','Acceleration','MPG'});

lm = fitlm(tbl,'MPG~Weight+Acceleration')
```

# Statistics Toolbox – Parametric Regression Analysis

Generated Output of **fitlm**:

lm =

Linear regression model:

$MPG \sim 1 + Weight + Acceleration$

Estimated Coefficients:

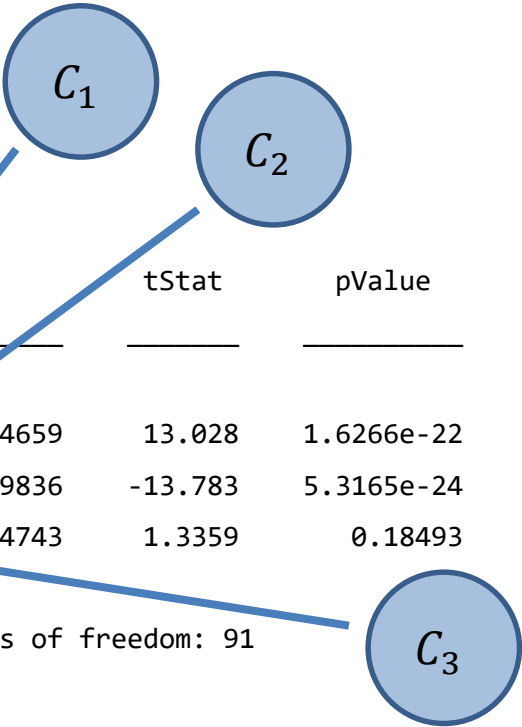
	Estimate	SE	tStat	pValue
(Intercept)	45.155	3.4659	13.028	1.6266e-22
Weight	-0.0082475	0.00059836	-13.783	5.3165e-24
Acceleration	0.19694	0.14743	1.3359	0.18493

Number of observations: 94, Error degrees of freedom: 91

Root Mean Squared Error: 4.12

R-squared: 0.743, Adjusted R-Squared 0.738

F-statistic vs. constant model: 132, p-value = 1.38e-27

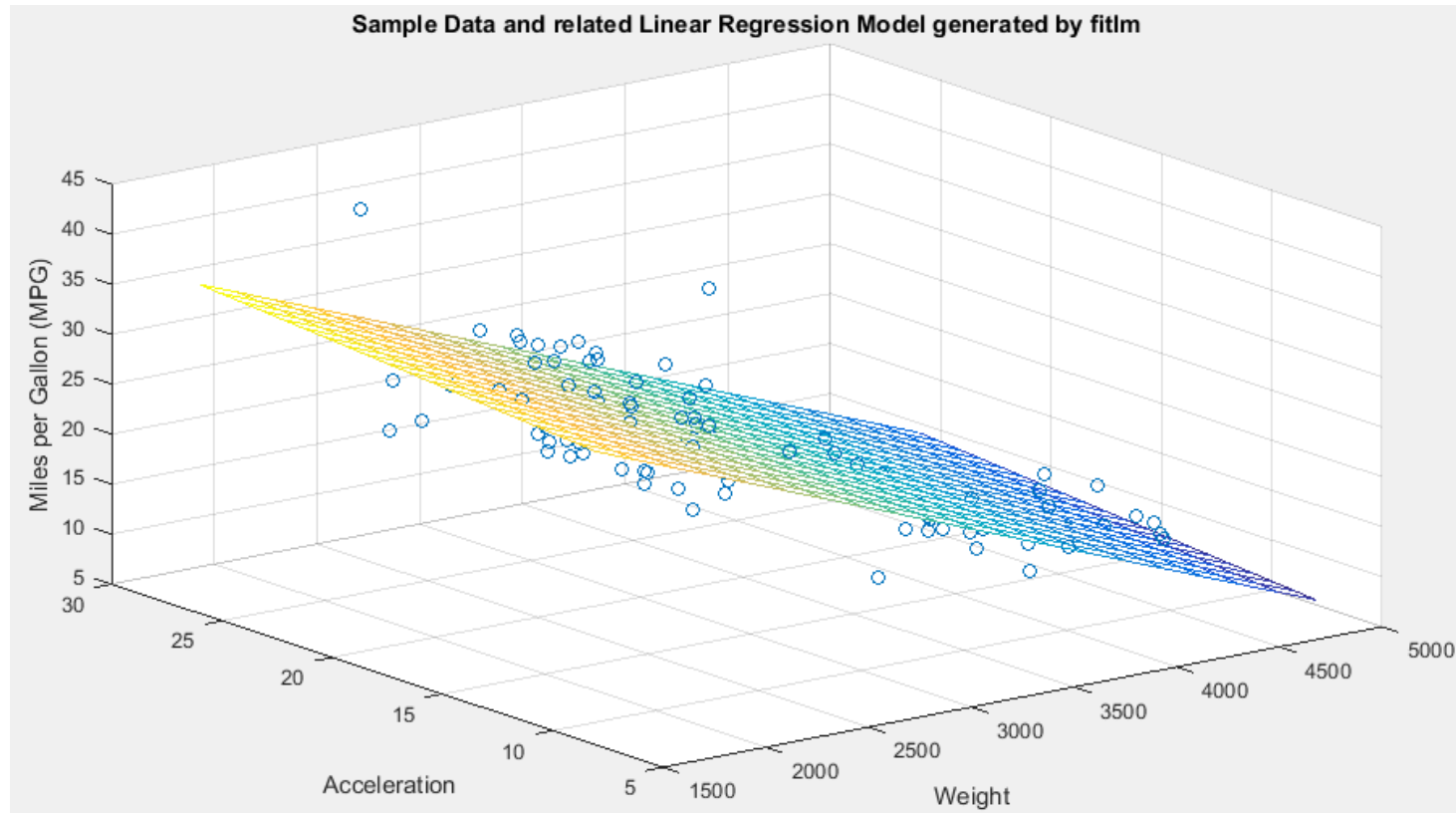


Source: <http://www.mathworks.com/help/stats/fitlm.html>



## Statistics Toolbox – Parametric Regression Analysis

- Illustration of **fitlm**-output:



Source: <http://www.mathworks.com/help/stats/fitlm.html>

# List of Commands

Command	Explanation
linprog	Solve linear programming problems
intlinprog	Mixed-integer linear programming
quadprog	Solve quadratic programming problems
fminunc	Minimization of unconstrained nonlinear multivariable function
optimoptions	Create optimization options
fmincon	Minimization of constrained nonlinear multivariable function
lsqnonlin	Solve nonlinear least-squares problems
scatter	Generate Scatter plot
linspace	Generate linearly spaced vector
rand	Returns uniformly distributed random number
mean	Returns average value of array

Command	Explanation
median	Returns median of an array
mode	Returns most frequent value of an array
std	Returns the standard deviation of an array
var	Returns the variance
cov	Returns the covariance
corrcoef	Returns the correlation coefficients
boxplot	Creates a box plot
grpstats	Summary statistics organized by group
gscatter	Scatter plot by group
makedist	Create probability distribution object
fitdist	Fit probability distribution object to data

# List of Commands continued

Command	Explanation
pdf	Probability density function
cdf	Cumulative distribution function
random	Generate random numbers
dfittool	Open Distribution Fitting app
kstest	One-sample Kolmogorov-Smirnov test
fitlm	Create a linear regression model
table	Create a table from workspace variables