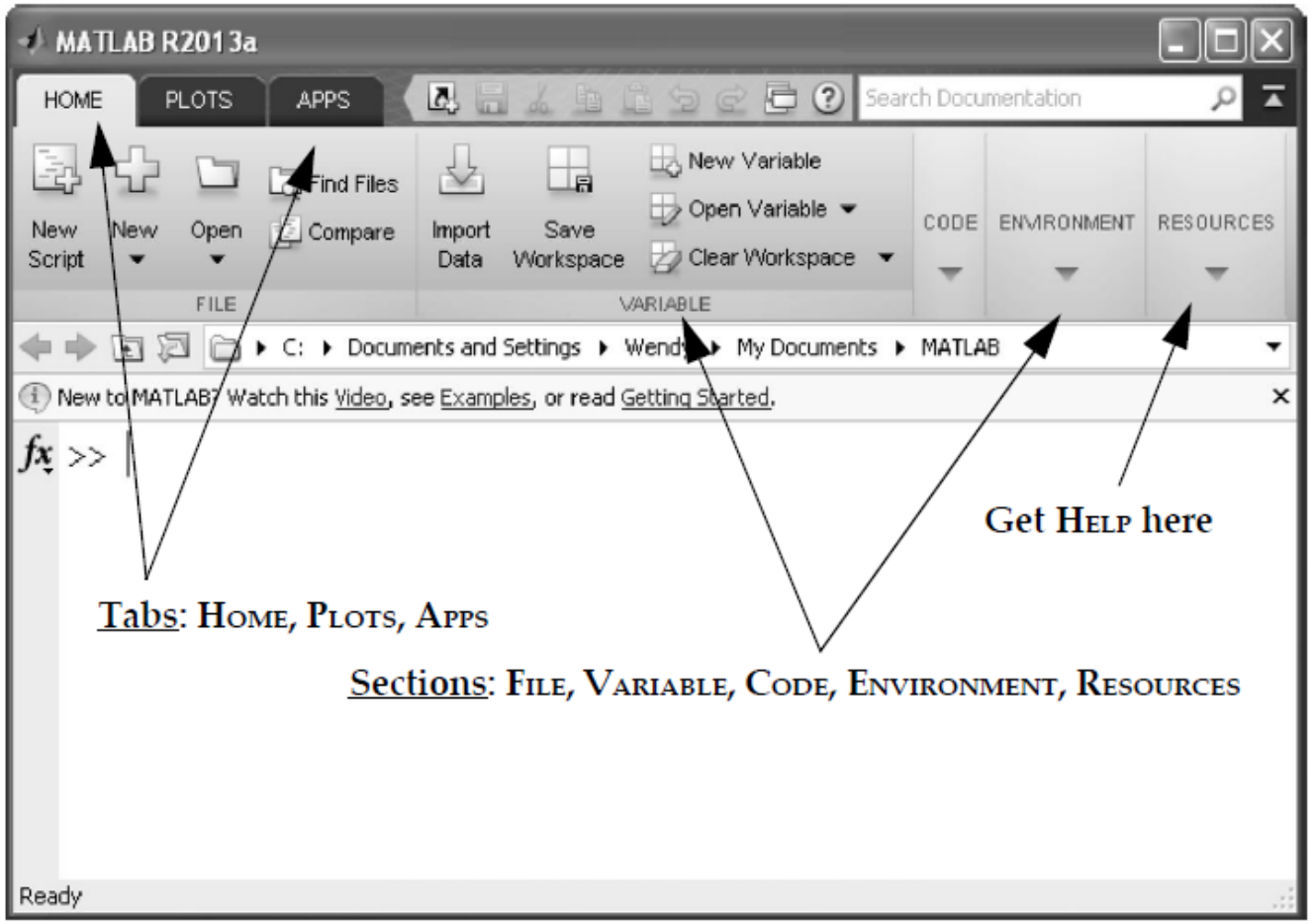
ELE 101

MATLAB

1. Giriş

MATLAB

- Command Window
- Current Folder
- Workspace
- Command History
- **`help functionname`**
- **`doc functionname`**
- ***`functionname`*`(`**
- **`lookfor parallel`**
- **`http://www.mathworks.com/matlabcentral/`**



Data Import Export

- .mat
  - load
  - save
- Comma-Separated Value (.csv) Files
  - Csvread
  - csvwrite
- Text
  - Save -ascii
- Dlmread
- .xls
  - spreadsheet
  - Xlsread
  - xlswrite

Dataset

- dir
  - Dosyalar
- **DensityEarth.txt**
  - **Dünya yoğunluğunun çeşitli yerlerde ölçümü**
- İris.mat
  - 150 tane Balık boyları (sepal length and with , petal length and width in cm)
    - 3 çeşit balık için (sınıflandırma amaçlı)
- USTemps.txt
  - Şehir , enlem, boylam ve ortalama sıcaklık (56 gözlem)

```
% First see what is in the workspace. This command
% lists all variables in the workspace.
who
% The workspace is empty and nothing is returned.
% Now load the iris.mat file.
load iris
% What is in the workspace now?
who
```

```
% Now save the variables in another file.
% We will save just the setosa variable object.
% Use save filename varname.
save setosa setosa
% See what files are in the current directory.
dir
% Remove objects from workspace to clean it up.
clear
% The directory should be empty. Load the earth data.
load DensityEarth.txt -ascii
% See what is in the workspace.
Who
```

- **save *filename varname* -ascii**

Import Wizard

```
% Remove objects for simplicity.
clear
% See what is in the workspace.
who
% This is displayed in the command window.
• Your variables are:
• City JanTemp Lat Long
```

Data Objects in Matlab

- Arrays
  - Scalar: Tek bir sayı
  - Vector: Bir boyutlu dizi
  - Matrix: İki boyutlu
  - Çok boyutlu

```
% Create a vector x.
x = [2, 4, 6];
% Delete the second element.
x(2) = [];
% Display the vector x.
disp(x)
% Find the elements that are negative.
ind = find(x < 0);
% Print the vector ind to the command window.
ind
```

Arrays

- Row
- Column
- Concetanate
  - Yan yana
  - Alt alta
- Sayı dizisi
  - Başlangıç: Aralık : Sonuç
  - İleri, geri
  - Sıfır dizisi
  - Bir dizisi
  - 3 boyutlu o `= ones(2,4,3)`
  - `strg = 'This is a string'`

Cells

- Array ve Cell farkı nedir?
- **`cell_arry = cell(2,4,3)`**

```
% Create a cell array, where one cell contains
% numbers and another cell element is a string.
cell_arry2 = {[1,2], 'This is a string'};
% Let’s check the size of the cell array
size(cell_arry2)
```

- Kıvrık parantez!

Structures

- Araba.{marka, model, cc, beygir gücü, silindir, renk}
- Öğrenci.{Ad, Soyad, NO, Yaş, Sınıf, GNO, Statü,Burs }
- Farklı data türleri (int, float, string vs. bir arada olabilir)
- **S = struct('*field1*',data1,'*field2*',data2,...).**

```
% Create a structure called employee with three fields.
employee = struct(...
    'name',{{'Wendy','MoonJung'}},...
    'area',{{'Visualization','Inference'}},...
    'deg',{{'PhD','PhD'}},...
    'score',[90 100])
all_names = employee.name
```

Table

- Cell ile benzerliği?

```
load USTemps
% Create a table using all four variables.
UTs_tab = table(City,JanTemp,Lat,Long)
% See what is in the workspace.
whos
```

- readtable function
  - import a file as a table object
    - delimited text files (.txt, .dat, or .csv)
    - Excel spreadsheet file with .xls or .xlsx extensions

Array

- Cell array
  - A{1,1}: içerik
  - A(1,1): hücre
  - A{1,1}(1:2)

```
% Extract the employee's area of interest.
e_area = employee.area;
% Display the contents in the window.
e_area
% Display Wendy's score.
employee.score(1)
% Get MoonJung's area.
employee.area{2}
```

TABLE 1.4

Examples of Accessing Elements of Arrays

Notation	Usage
`a(i)`	Access the *i*th element (cell) of a row or column vector array (cell array)
`a(3:5)`	Access elements 3 through 5 of a vector or cell array
`A(:,i)`	Access the *i*th column of a matrix or cell array. In this case, the colon in the row dimension tells MATLAB to access all rows.
`A(i,:)`	Access the *i*th row of a matrix or cell array. The colon tells MATLAB to gather all of the columns.
`A(2:4,1:2)`	Access the elements in the second, third, and fourth rows and the first two columns
`A(1,3,4)`	Access the element in the first row, third column on the fourth entry of dimension 3 (sometimes called the page).

Tablo

```
% Get a partial table by extracting the first
% three rows.
U1 = UTs_tab(1:3,:)
% Get the JanTemp variable.
jt = UTs_tab.JanTemp;
% See if it is equal to the JanTemp variable.
isequal(jt,JanTemp)
% We get an answer of 1, indicating they are the same.
% We can extract the Lat and Long data for the first
% three cities using the variable names.
U2 = UTs_tab{1:3,{'Lat','Long'}}
```

Stacking Arrays

- Bir datasetteki 3 değişkeni birleştirmek
  - İleride kmeans kullanılarak öbeklenecek
  - Alt alta

```
% First load the data back into the workspace.
load iris
% Now, we use the semicolon to stack
% setosa, versicolor, and virginica data objects.
irisAll = [setosa; versicolor; virginica];
% Now look at the workspace to see what is there now.
who
% Check on the size of irisAll.
size(irisAll)
```

Yan yana birleştirmek

```
% Load the data if not in the workspace.
load USTemps
% Use commas to concatenate as a row.
UStemps = [JanTemp, Lat, Long];
% Check the workspace.
who
% Verify the size of UStemps:
size(UStemps)
```

- **`X = [ [A, B] ; [C, D] ];`**
- **`TIP:` The semi-colon after a statement stops MATLAB from printing the results of the expression to the command line**

TABLE 1.5

File Management Commands

Command	Usage
`dir, ls`	Shows the files in the present directory
`delete filename`	Deletes *filename*
`pwd`	Shows the present directory
`cd dir`	Changes the directory. There is also a pop-up menu and button on the desktop that allows the user to change directory, as we show below.  .
`edit filename`	Brings up *filename* in the editor
`type filename`	Displays the contents of the file in the command window
`which filename`	Displays the path to *filename*. This can help determine whether a file is part of base MATLAB.
`what`	Lists the `.m` files and `.mat` files that are in the current directory

TABLE 1.6

Commands for Workspace Management

Command	Usage
`who`	Lists all variables in the workspace.
`whos`	Lists all variables in the workspace along with the size in bytes, array dimensions, and object type.
`clear`	Removes all variables from the workspace.
`clear x y`	Removes variables `x` and `y` from the workspace.

TABLE 1.7

List of MATLAB® Punctuation

Punctuation	Usage
`%`	A percent sign denotes a comment line. Information after the `%` is ignored.
`,`	When used to separate commands on a single line, a comma tells MATLAB to display the results of the preceding command. When used to combine elements or arrays, a comma or a blank space groups elements along a row. A comma also has other uses, including separating function arguments and array subscripts.
`;`	When used after a line of input or between commands on a single line, a semicolon tells MATLAB not to display the results of the preceding command. When used to combine elements or arrays, a semicolon stacks them in a column.
`...`	Three periods denote the continuation of a statement onto the next line.
`:`	The colon specifies a range of numbers. For example, `1:10` means the numbers 1 through 10. A colon in an array dimension accesses all elements in that dimension.

Aritmetik İşlemler

- +, -, *, /, ^
- A ve B matrislerinin çarpımı : * (boyutlara dikkat)
- A matrisinin tersi (eğer varsa): inv(A)
- A*x= b denkleminin çözümü
  - x=inv(A)*b
  - A/B
- Element bazında işlem
  - Toplam
  - Çarpım
  - Kuvvet

TABLE 1.8

List of Element-by-Element Operators

Operator	Usage
.*	Multiply element by element
./	Divide element by element
.^	Raise each element to a power

Fonksiyon

- Büyük bir kodun çeşitli yerlerinde sıklıkla yaptığımız bir işlem dizisi varsa
  - Aynı şeyleri her yere tekrar tekrar yazmak yerine, fonksiyon tanımlanır e gerektiği yerlerde çağrılır
    - Kod daha modüler hale gelir
    - Aynı fonskiyon başka projelerde de kullanılabilir
- ***Function syntax: functionname(arg1, ..., argk)***
- ***[out1,...,outm] = functionname(arg1,...,argk)***
- **Command syntax: *functionname arg1 ... Arg2***

```
% Save the data for the USTemps to a .mat file.
save USTemps City JanTemp Lat Long
% That used the command syntax to call a function.
% Now use the function syntax.
save('USt.mat','City','JanTemp','Lat','Long')
% See what is in our directory now.
Dir
```

- https://matlabacademy.mathworks.com/details/matlab-onramp/gettingstarted?s_tid=abt_train_b

ELE 101

MATLAB

2. Görselleştirme

Plot

- **`Help plot`**
- **`plot(x,y,'color_linestyle_marker')`**
- **`İki grafik birden:  plot(x1,y1,x2,y2)`**
- Style (Çizgi türü, renk, marker)
  - **`plot(x,y,':')`**
  - **`plot(x,y,'g'), plot(x,y,'.')`**
  - **`plot(x,y,'b--*')`**
  - **`plot(x,y,'r:',x2,v2,'k.-o')`**

TABLE 2.3
Marker Styles for Plots

Notation	Marker Style
`.`	point
`o`	circle
`x`	x-mark
`+`	plus
`*`	star
`s`	square
`d`	diamond
`v`	down triangle
`^`	up triangle
`<`	left triangle
`>`	right triangle
`p`	pentagram
`h`	hexagram

TABLE 2.1
Line Styles for Plots

Notation	Line Type
`-`	solid line
`:`	dotted line
`-.`	dash-dot line
`--`	dashed line

TABLE 2.2
Line Colors for Plots

Notation	Color
`b`	blue
`g`	green
`r`	red
`c`	cyan
`m`	magenta
`y`	yellow
`k`	black
`w`	white

Plot (devam)

- Xlabel, ylabel, title
  - **xlabel('*text*'),ylabel('*text*'),title('*text*')**
- İki defa plot çalıştırırsanız ikinciyi ilkinin üzerine çizer
  - Üst üste çizdirmek için «hold on» ve «hold off»
- Bir pencere içinde birden fazla grafik penceresi (ör. 2x2, 2x3 vb)
  - **% Create the left-most plot—the first one.**
  - **subplot(1,2,1)**
  - **plot(x,y)**
  - **% Add a title to the first plot.**
  - **title('My Plot')**
  - **% Create the right-most plot—the second one.**
  - **subplot(1,2,2)**
  - **plot(x2,y2)**

3D Data

- **`plot3(x,y,z)`**
  - Ör. Helix
  - t = 0:pi/50:10*pi;
  - plot3(sin(t),cos(t),t);
- **`surf(X,Y,Z)`**
  - [X,Y] = **meshgrid**(1:0.5:10,1:20);
  - Z = sin(X) + cos(Y);
  - C = X.*Y;
  - surf(X,Y,Z,C)
  - Colorbar
- **`mesh(X,Y,Z)`**
  - [X,Y] = meshgrid(-5:.5:5);
  - Z = Y.*sin(X) - X.*cos(Y);
  - s = mesh(X,Y,Z,'FaceAlpha','0.5')

2D Örnek: Gauss Dağılımı pdf Çizdirme

```
% To illustrate how to plot a curve, first
% generate the values for a normal distribution.
% This creates a standard normal probability
% distribution object.
stdnpd = makedist('Normal');
% Now, create one with different parameters.
npd = makedist('Normal','mu',0,'sigma',2);
% Define a vector for the domain.
x = -4:.01:4;
% Get the y values for both distributions.
y1 = pdf(stdnpd, x);
y2 = pdf(npd, x);
% Now, plot the curves on the same set of axes.
plot(x,y1,x,y2,'-.')
xlabel('X')
ylabel('PDF')
title('Plots of Normal Distributions')
legend('mu = 0, sigma = 1','mu = 0, sigma = 2')
```

3D Örnek: Çift Değişkenli Gauss Dağılımı pdf

```
%İki Gauss rastgele sayısının beraber dağılımı
% Get the vector for the mean.
mu = [0 0];
% Get the covariance matrix.
sigma = eye(2);
% Obtain the (x,y) pairs for the domain.
x = -3:.2:3; y = -3:.2:3;
[X,Y] = meshgrid(x,y);
% Evaluate the multivariate normal at
% the coordinates.
Z = mvnpdf([X(:), Y(:)],mu,sigma);
% Reshape to a matrix.
Z = reshape(Z,length(x),length(y));
% The surface plot is shown in Figure 2.3.
% Now, create the surface plot and add labels.
surf(X,Y,Z);
xlabel('X'), ylabel('Y'), zlabel('PDF')
title('Multivariate Normal Distribution')
axis tight
```

Scatter plot: Saçılım grafiği

- **2D: scatter(x,y,s,c)**
- **3D: scatter3(x,y,z,s,c)**

```
% Load the USTemps and iris data.
load USTemps
load iris
% Construct a 2-D scatter plot with plot,
% using temperature and latitude.
plot(Lat, JanTemp, '*')
% Adjust the axes to change white space.
axis([24 50 -2 70])
% Add labels.
xlabel('Latitude')
ylabel('Temperature (degs)')
title('Average January Temperature - US Cities')
```

3D Scatter plot örneği

```
% The next example shows how to construct a 3-D scatter plot using plot3.
% The scatter plot is shown in Figure 2.5.
% Construct a 3-D scatter plot using plot3.
plot3(Long, Lat, JanTemp, 'o')
% Add a box and grid lines to the plot.
box on
grid on
% Add labels.
xlabel('Longitude')
ylabel('Latitude')
zlabel('Temperature (degs)')
title('Average January Temperature - US Cities')
```

- ***TIP*: You can save, print, or export your plot using options in the FILE menu of the Figure window. Use the EDIT menu in the Figure window to copy the plot for pasting into documents.**
- **Tools → Edit Plot**
- **PLOTS Tab**

ELE 101

MATLAB

3. İstatistik

Giriş

- Veriyi indirdik, belli yöntemlerle görselleştirdik.
- Dağılım nasıl (görsel değil, matematiksel olarak)
- Elimizde bu verinin gerçek olasılıksal modeli yok (ELE 273)
  - Örneklem üzerinden bir takım parametreleri ölçebiliriz (Ort. Std. Vs.)
  - Olasılıksal dağılım için bir modele karar verebiliriz
- Tipik Değer: En basit istatistik
  - Mean (ortalama): $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
  - Median (median): Veriyi sıraladığımızda en ortadaki değer
  - Mode (mod): En sıklıkla karşılaştığımız değer
  - Trimmed mean : En yüksek ve en düşük %k'lık dilimi (outlier) atıp ortalama alma

Örnek:

```
% We will first load the variable.
load earth
% Now, we find the mean.
xbar = mean(earth)
% Find the median density of the Earth.
med = median(earth)

% Find the trimmed mean density of the Earth.
% We will use 20% for our trimming (trim the top and bottom 10%).
xbar_trim = trimmean(earth, 20)
% Now, find the mode.
mod_earth = mode(earth)
```

Dağınıklık

- Range (Aralık)
- $range = \max_i\{x_i\} - min_i\{x_i\}$
- Variance (değişinti)
- $\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
- Bunun karekökü daha faydalı bir istatistiktir
  - Standart sapma
  - $\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$

```
% First find the minimum, maximum and range
% of the earth data.
minearth = min(earth)
maxearth = max(earth)
% Find the range.
rngearth = range(earth)
% We can also find the range, as follows
maxearth - minearth
% Next, we find the variance and the
% standard deviation.
vearth = var(earth)
searth = std(earth)
% We can also find the standard deviation
% by taking the square root of the variance.
sqrt(var(earth))
```

İki verinin ilintisi (korelasyon)

- Covariance (Kovaryans)
- $cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$
  - Birinci veri ortalamasını aşınca, ikinci de aşıyor mu (pozitif korelasyon)
- İlinti katsayısı:
- $corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$
  - (-1 ve +1 arasında bir sayıdır)
  - Daha fazla fikir veren bir parametre

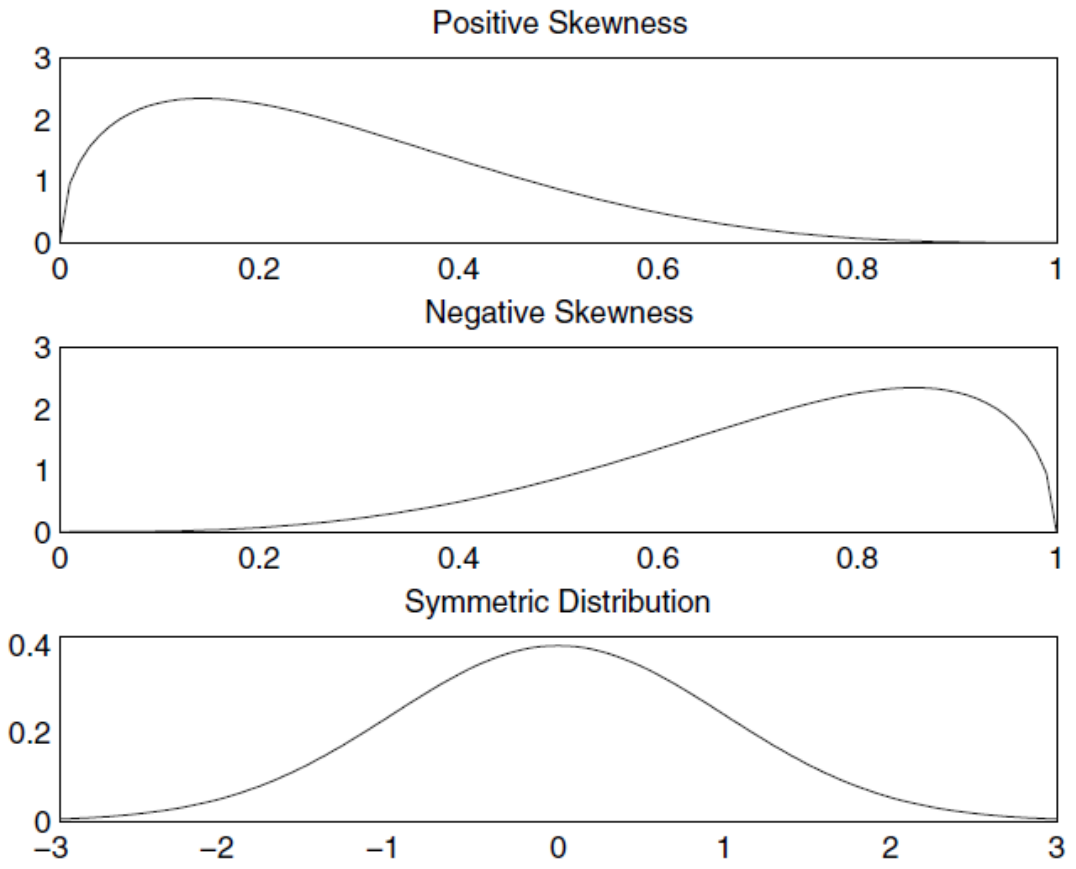
```
% Now, let's find the covariance matrix of
the

% setosa iris. First, we need to load it.

load iris

% Find the covariance matrix of the setosa
object.

cvsetosa = cov(setosa)

% Find the correlation coefficients.

crsetosa = corrcoef(setosa)

% If the argument to the var function is a
matrix,

% then it will return the variance of each
column

% or variable.

var(setosa)
```

Dağılım

- Dağılımın tipik değerini ve bu değer etrafında ne kadar saçılım gösterdiğini gördük
- Dağılımın şekli nasıl?
- Percentile: Veri aralığı 100 eşit parçaya bölünür
  - Ör. Çocukların boy ve kiloları (80 percentile: Aynı yaştaki çocukların %80'inden daha uzun)
- Quartile : Veri aralığı 4 eşit parçaya bölünür
  - $\hat{Q}_{0.25}, \hat{Q}_{0.50}\ (medyan), \hat{Q}_{0.75}$
  - Interquartile range (çeyrekler açıklığı) $\hat{Q}_{0.75} - \hat{Q}_{0.25}$ (verinin yüzde ellisinin bulunduğu aralık)
- Skewness (çarpıklık): Dağılımın asimetrikliği
  - $$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{\frac{3}{2}}}$$

Örnek

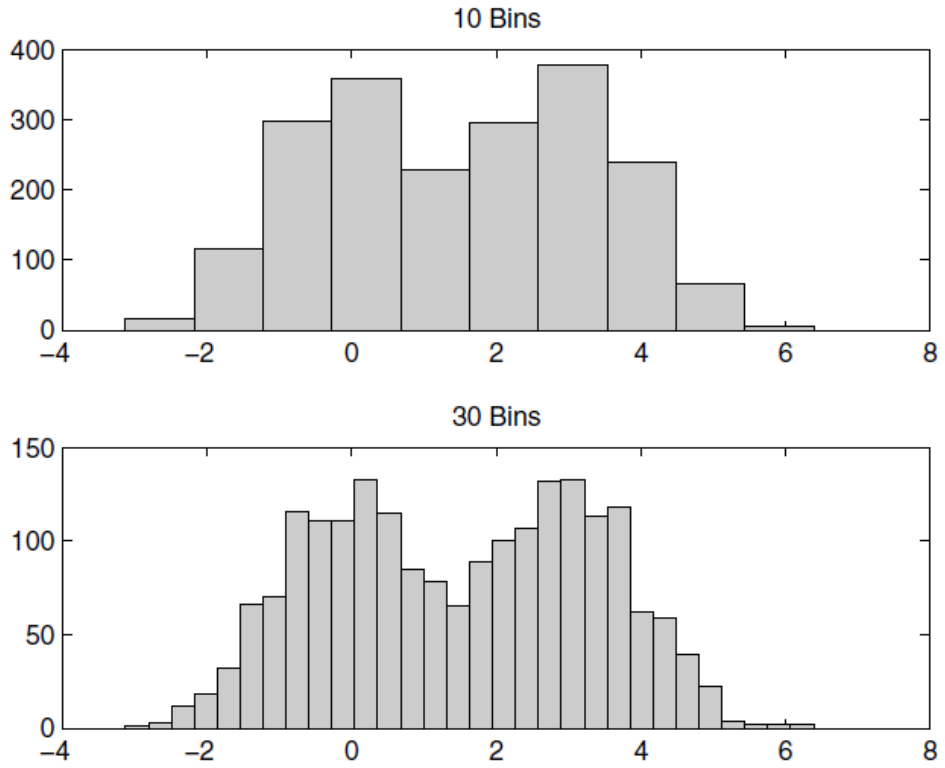
```
% First, we find the quartiles.
quart = quantile(earth,[.25 .50 .75])
% This is what we get from the median function.
median(earth) % This gives the same result.
% Next, find the IQR of the earth data.
iqrearth = iqr(earth)
% Get the quartiles using a different function.
pearth = prctile(earth,[25 50 75])
% Find the sample skewness of the earth data.
skearth = skewness(earth)
```



Dağılımı Görselleştirme

- Veriyi görselleştirmeyi görmüştük (plot, plot2, surf, scatter)
- Dağılımı görselleştirme (histogram)

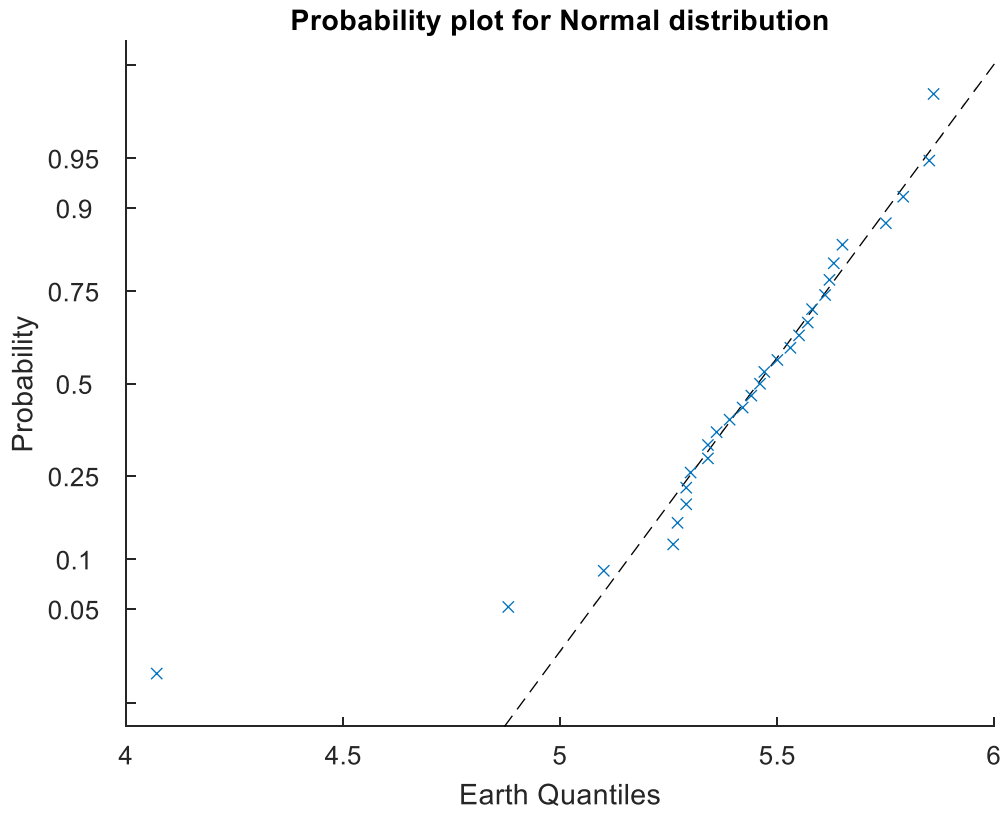
```
% Construct histogram using 10 bins.
% This is in base MATLAB.
hist(earth)
title('Histogram of the Earth Density Data')
xlabel('Multiple of the Density of Water')
ylabel('Frequency')
```



Örneklem Dağılımını Teorik bir Model ile karşılaştırma

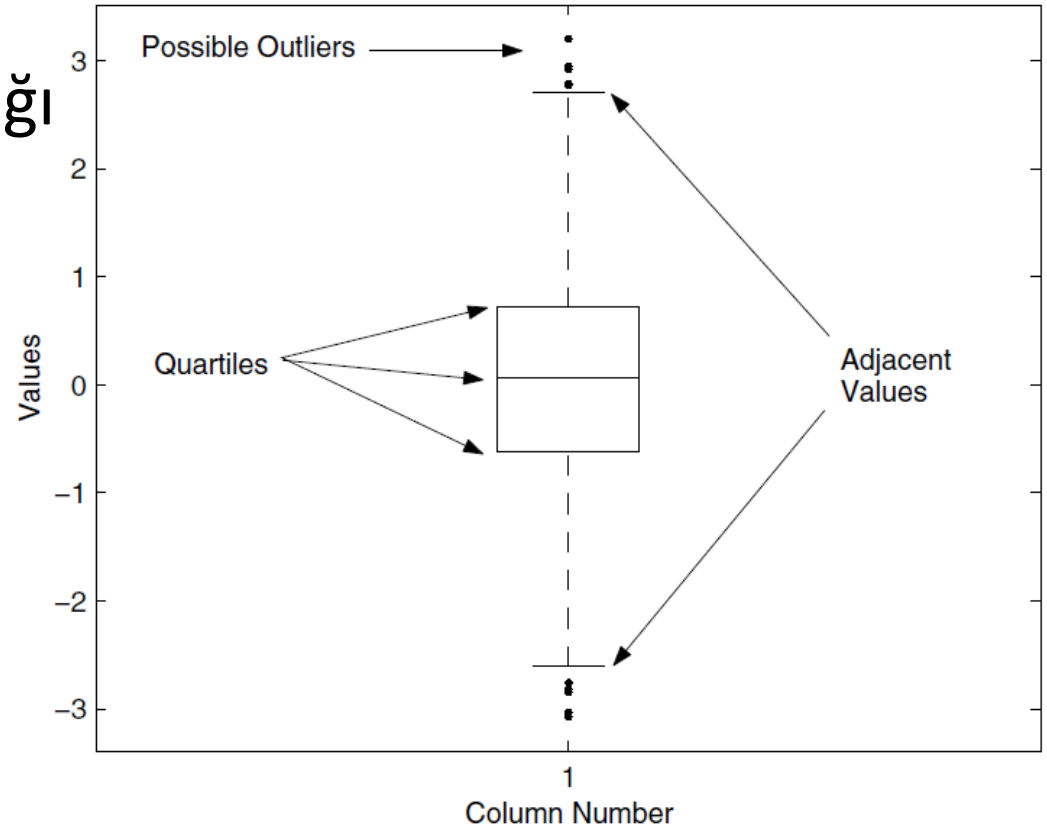
- Q-Q Plot: İki dağılım benzer mi (quantil-quantil karşılaştırması)
- Probability Plot: Örneklem dağılımını verilen teorik dağılım ile karşılaştırır

```
% Get a probability plot comparing the
% earth data to a normal distribution.
probplot(earth)
xlabel('Earth Quantiles')
% Construct a Q-Q plot of the sepal length
% for Virginica and Versicolor in the iris
data.
qqplot(virginica(:,1),versicolor(:,1))
title('Q-Q Plot of Sepal Length - Iris Data')
xlabel('Virginica Quantiles')
ylabel('Versicolor Quantiles')
```



Güven Aralığı – Box Plot

- Grafikte +-%25 quartile ve %95 güven aralığı bulunmaktadır.



ELE 101

MATLAB

4. Olasılıksal Dağılımlar

Giriş

- Veriyi görselleştirdik ve basit istatistiksel formüller üzerinden bazı dağılım parametrelerini bulduk (ortalama, std, range, percentile, çarpıklık, korelasyon (iki veri seti için)) Empirical
  - Bunlar bilinen olasılıksal dağılımlara ne kadar uyuyor?
  - Veriyi kendimiz rastgele üretmek istesek nasıl üreteceğiz?
  - İkisi için de olasılıksal modelleri bilmek gerekir (ELE 273) Theoretical
- Dağılımlar: Ayrık-Sürekli, Tek-Çok değişkenli, Parametrik
- Sürekli Dağılımlar
  - Bir aralıktaki bütün değerleri alabilirler (ör. Sıcaklık, nem, sinyal genliği)
  - Kümülatif dağılım fonksiyonu (CDF) $F_X(x)$: Bir değerden küçük eşit olma olasılığı
  - Olasılık yoğunluk fonksiyonu $f_X(x)$: Bir değeri alma olasılık yoğunluğu
    - Bir aralıkta integrali alındığında o aralıkta değer alma olasılığını verir
  - $F_X(a) = \int_{-\infty}^{a} f_X(x)dx$, $P(a \leq X \leq b) = \int_{a}^{b} f_X(x)dx$

TABLE 4.1

Some Continuous Distributions in the Statistics Toolbox

Distribution Name	Parameters[a]	Root Word	Add to Root Word for Operations
Beta	**a**, **b**	**beta**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**
Chi-Square	**df** (degrees of freedom)	**chi2**	**pdf**, **cdf**, **inv**, **stat**, **rnd**
Exponential	**mu** (mean)	**exp**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**
Extreme Value	**mu** (mean log value) **sigma** (spread log value)	**ev**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**
F	**df1**, **df2** (degrees of freedom)	**f**	**pdf**, **cdf**, **inv**, **stat**, **rnd**
Gamma	**a** (shape) **b** (scale)	**gam**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**
Lognormal	**mu** (mean) **sigma**(spread)	**logn**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**
Normal (Gaussian)	**mu** (mean) **sigma** (spread)	**norm**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**
Rayleigh	**b**	**rayl**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **rnd**
Student's *t*	**v** (degrees of freedom)	**t**	**pdf**, **cdf**, **inv**, **stat**, **rnd**
Uniform	**a**, **b** (defines interval)	**unif**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **rnd**
Weibull	**a** (scale) **b** (shape)	**wbl**	**pdf**, **cdf**, **inv**, **stat**, **fit**, **like**, **rnd**

Fonksiyonlar

- Drnd(N,1): İlgili dağılıma uygun N adet rastgele sayı üretmek
- Dpdf(x): İlgili olasılık yoğunluk fonksiyonunun x noktasındaki değeri
- Dcdf: (x): İlgili kümülatif fonksiyonunun x noktasındaki değeri
- Dinv(p): İlgili dağılıma p quantili
- Dstat(param): İlgili dağılımın parametreleri
- Dfit(data): Verinin (D dağılımı olduğu varsayımıyla) parametreleri
- Dlike(param, like): Verinin D dağılımında olma olabilirliği

Ayrık dağılımlar

- Ayrık rastgele değişkenler ayrık değerler alır (ör. Zar,)
- Olasılık ağırlık fonksiyonu (probability mass function)
  - $P_X(x) = P(X = x)$: Ayrık rastgele değişkenin x değeri alma olasılığı
- Kümülatif dağılım fonksiyonu $F_X(x)$: Bir değerden küçük eşit olma olasılığı
  - $F_X(a) = \sum_{x \le a} P_X(x)$

Some Discrete Distributions in the Statistics Toolbox

Distribution Name	Parameters[a]	Root Word	Add to Root for Operations
Binomial	`n`, `p`	`bino`	`pdf`, `cdf`, `inv`, `stat`, `fit`, `rnd`
Geometric	`p`	`geo`	`pdf`, `cdf`, `inv`, `stat`, `rnd`
Hypergeometric	`m`, `k`, `n`	`hyge`	`pdf`, `cdf`, `inv`, `stat`, `rnd`
Negative binomial	`r`, `p`	`nbin`	`pdf`, `cdf`, `inv`, `stat`, `fit`, `rnd`
Poisson	`lambda`	`poiss`	`pdf`, `cdf`, `inv`, `stat`, `fit`, `rnd`
Uniform (discrete)	`N`	`unid`	`pdf`, `cdf`, `inv`, `stat`, `rnd`

Çoklu dağılımlar

- Ör. İki rastgele değişken bir düzlemde yer alırlar
  - Bunun dağılımı üç boyutlu grafikle gösterilebilir
    - Ör. İki değişkenli Gauss (Çan şeklinde üç boyutlu bir grafik)
  - Multivariate Normal
    - Kovaryans matrisi (simetrik bir matris)

Additional Distributions in the Statistics Toolbox

mvnpdf	Multivariate normal PDF
mvncdf	Multivariate normal CDF
mvnrnd	Generate multivariate normal random variables
mvtpdf	Multivariate Student's *t* PDF
mvtcdf	Multivariate Student's *t* CDF
mvtrnd	Generate multivariate *t* random variables
ksdensity	Kernel density—arguments include options for PDF, CDF, inverse CDF, and fitting

Matlab Kodları: Kernel density estimation

```
% PAGE 85
% Generate a set of normal random variables.
% The mean is 0 and sigma is 1.
n = 10;
x = normrnd(0,1,1,n);
% Get a grid of points to evaluate density.
pts = linspace(-4,4);
% Set up a vector for our estimated PDF.
kpdf = zeros(size(pts));
% We need a bandwidth for the kernels.
% This is the normal reference rule [Scott, 1992].
h = 1.06*n^(-1/5);
% Hold the plot, because we will add
% a curve at each iteration of the loop.
hold on
for i = 1:n
    % Use the normal kernel, noting that
    % the mean is given by an observation
    % and the standard deviation is h.
    f = normpdf(pts, x(i), h);

    % Plot the kernel function for i-th point.
    plot(pts, f/n);

    % Keep adding the individual kernel function.
    kpdf = kpdf + f/n;
end
% Plot the kernel density estimate.
plot(pts, kpdf)
title('Kernel Density Estimate')
xlabel('X')
ylabel('PDF')
hold off
```

- *Kernel fonksiyonları*
  - h parametresi: bant genişliği : std sapma
  - $x(i)$: ortalama
- Her bir kernel fonksiyonu toplanır
- Gerçek olasılık dağılım fonksiyonunun bir kestirimi elde edilmiş olur
- Hazır fonksiyon
  - **`kpdf = ksdensity(x)`**
- Ters CDF
  - **`kicdf = ksdensity(x,pts,'function','cdf')`**
- ..
- ..

Matlab Kodları: Gauss ve Student's t dağılımı

```
% PAGE 87

% First, generate some points for the
domain.

pts = linspace(-4,4);

% Obtain a standard normal PDF.

nrmpdf = normpdf(pts,0,1);

% Obtain a Student's t with v = 1.

stpdf = tpdf(pts, 1);

% Plot the two PDFs.

plot(pts,nrmpdf,'-',pts,stpdf,'-.')

axis([-4 4, 0 0.42])

title('Normal and Student''s t
Distributions')

xlabel('x')

ylabel('PDF')

legend('Normal','Student''s t')
```

- Student's t : tek parametreli (v)
- v arttıkça Gauss'a benzer.

Ayrık rastgele sayılar: Ör. Poisson

```
% PAGE 88
% First specify the parameter for the Poisson.
lambda = 3;
% Get the points in the domain for plotting
pts = 0:10;
% Get the values of the PDF
ppdf = poisspdf(pts,lambda);
% Get the values of the CDF
pcdf = poisscdf(pts,lambda);
% Construct the plots
subplot(2,1,1)
plot(pts, ppdf, '+')
ylabel('Poisson PDF')
title('Poisson Distribution \lambda = 3')
subplot(2,1,2)
stairs(pts, pcdf)
ylabel('Poisson CDF')
xlabel('X')
```

- Poisson rastgele değişkenleri, negatif olmayan tamsayı değerleri alır
- Bir olayın olma sayısını modellemede kullanılır
- Tek parametrelidir (λ)
- Ortalaması : λ

Çok değişkenli student’s t dağılımı

```
% PAGE 90
% Example - Multivariate (2-D) Student's t
% First, set the parameters.
corrm = [1, 0.3 ; 0.3, 1];
df = 2;
% Get a grid for evaluation and plotting
x = linspace(-2,2,25);
[x1,x2] = meshgrid(x,x);
X = [x1(:), x2(:)];
% Evaluate the multivariate Student's t
PDF
% at points given by X.
pmvt = mvtpdf(X, corrm, df);
% Plot as a mesh surface.
mesh(x1,x2,reshape(pmvt,size(x1)));
xlabel('X1'), ylabel('X2'), zlabel('PDF')
title('2-D Student''s t Distribution')
```

- Çok değişkenli ve iki değişken birbiri ile ilintili
- Bu örnekte ilinti katsayısı 0.3
  - Pozitif ilintili
  - Biri arttıkça (azaldıkça) diğeri de artma (azalma) eğiliminde
- İlinti arttıkça çan eğrisi yanlardan bastırılır

Earth verisinin dağılım kestirimi

```
% PAGE 91
% First load the data into the workspace.
load earth
n = length(earth);
% Set two bandwidths.
% Normal reference rule - standard
deviation.
h1 = 1.06*std(earth)*n^(-1/5);
% Normal reference rule - IQR.
h2 = 0.786*iqr(earth)*n^(-1/5);
% Get a domain for the for the PDF.
[kd1,pts] =
ksdensity(earth,'bandwidth',h1);
[kd2,pts] =
ksdensity(earth,'bandwidth',h2);
plot(pts,kd1,pts,kd2,'-.')
legend('h1 = 0.1832','h2 = 0.1263')
title('Kernel Density Estimates - 2
Bandwidths')
ylabel('PDF'), xlabel('Density of Earth')
```

- Çok değişkenli ve iki değişken birbiri ile ilintili
- Bu örnekte ilinti katsayısı 0.3
  - Pozitif ilintili
  - Biri arttıkça (azaldıkça) diğeri de artma (azalma) eğiliminde
- İlinti arttıkça çan eğrisi yanlardan bastırılır

Parametre Kestirimi

```
% PAGE 95
% Load the earth data
load earth
% Estimate the parameters of a normal
distribution.
% Also ask for a 90% confidence interval
[mu1,sig1,muci1,sigci1] = normfit(earth,0.9);
% The following is the estimated mean.
display(mu1)
% Here is the estimated confidence interval.
display(muci1)
% The following estimated standard deviation is
shown.
display(sig1)
% This is the confidence interval.
display(sigci1)
% Alternative way to estimate mean and standard
deviation.
% Get the mean from the data.
mean(earth)
% Get the standard deviation
std(earth)
```

- Yandaki kod: Distribution fitting
  - Verinin dağılımı hangi ortalama ve std'li Gauss dağılımın grafiğine uyuyor?
- Alternatif:Maksimum olabilirlik
  - **`est = mle(X,'distribution','name'),`**
- **`[est,estci] = mle(x,'distribution','name','alpha',alpha)`**
- **`PDest = fitdist(x,'distname')`**
- **`estci = paramci(PDest)`**

- **`% Fit a normal distribution to the data.`**
- **`pdfit = fitdist(earth,'normal')`**

- **`paramci(pdfit)`**

- **`ksfit = fitdist(earth,'kernel');`**

Parametre Kestirimi

```
% PAGE 97
% Fit a normal distribution to the data.
pdfit = fitdist(earth,'normal')
% We can extract the confidence intervals
using the paramci function.
% This is one of the methods we can use
with a ProbDist object.
% Display the confidence intervals only.
% The intervals are given in the columns.
paramci(pdfit)
% Fit a kernel density to the earth data.
ksfit = fitdist(earth,'kernel')
```

```
% PAGE 98
% Get a set of values for the domain.
pts = linspace(min(earth)-1,max(earth)+1);
% Get the PDF of the different fits.
pdfn = pdf(pdfit,pts);
pdfk = pdf(ksfit,pts);
% Plot both curves
plot(pts,pdfn,pts,pdfk,'--')
% Plot the points on the horizontal axis.
hold on
plot(earth,zeros(1,n),'+')
hold off
legend('Normal Fit','Kernel Fit')
title('Estimated PDFs for the Earth Data')
xlabel('Density of the Earth')
ylabel('PDF')
```

Çoklu dağılım parametre kestirimi

```
% PAGEs 99 - 100
% Example of multivariate normal using the iris data.
load iris
% Construct a scatter plot matrix.
% This function is in base MATLAB.
plotmatrix(virginica)
title('Iris Virginica')
% Extract the variables from the Virginica species.
X = virginica(:,[1,3]);
% Estimate the mean.
mu = mean(X);
% Estimate the covariance.
cv = cov(X);
% Establish a grid for the PDF
x1 = linspace(min(X(:,1))-1,max(X(:,1))+1,25);
x2 = linspace(min(X(:,2))-1,max(X(:,2))+1,25);
[X1, X2] = meshgrid(x1,x2);
% Use the parameter estimates and generate the PDF.
pts = [X1(:),X2(:)];
Xpdf = mvnpdf(pts,mu,cv);
% Construct a mesh plot.
mesh(X1,X2,reshape(Xpdf,25,25))
title('Estimated PDF for Iris Virginica')
xlabel('Sepal Length'), ylabel('Petal Length')
zlabel('PDF')
axis tight
```

Rastgele Sayı Üretmek

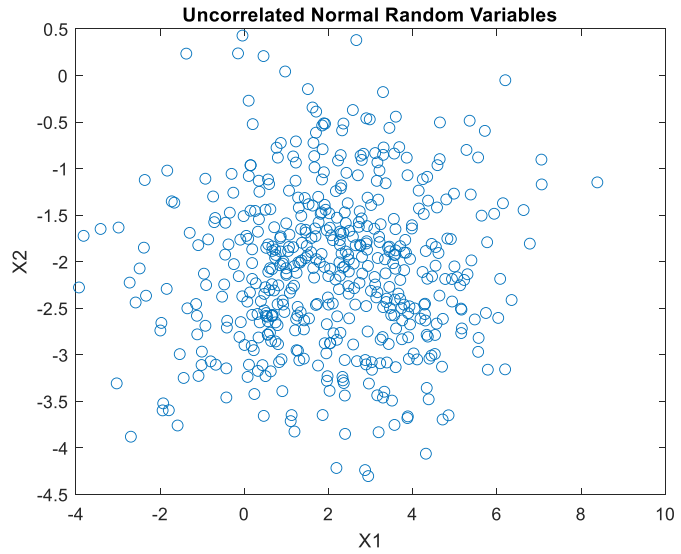
- Elimizde her zaman veri olmayabilir
- Kendimiz üretmemiz gerekebilir
- rand: 0-1 aralığında düzgün rastgele dağılımlı sayı
- randn : 0 ortalamalı ve 1 standart sapmalı Gauss rastgele sayısı
  - randn(n,p): $n \times p$ rastgele matris
- $a - b$ aralığında düzgün
  - $(b - a) * rand(n, p) + a$
- μ ortalamalı σ std sapmalı Gauss
  - $\mu + \sigma * randn(n, p)$
- Belli bir aralıkta düzgün ayrık rastgele tam sayı
  - $randi([a, b], n, p)$

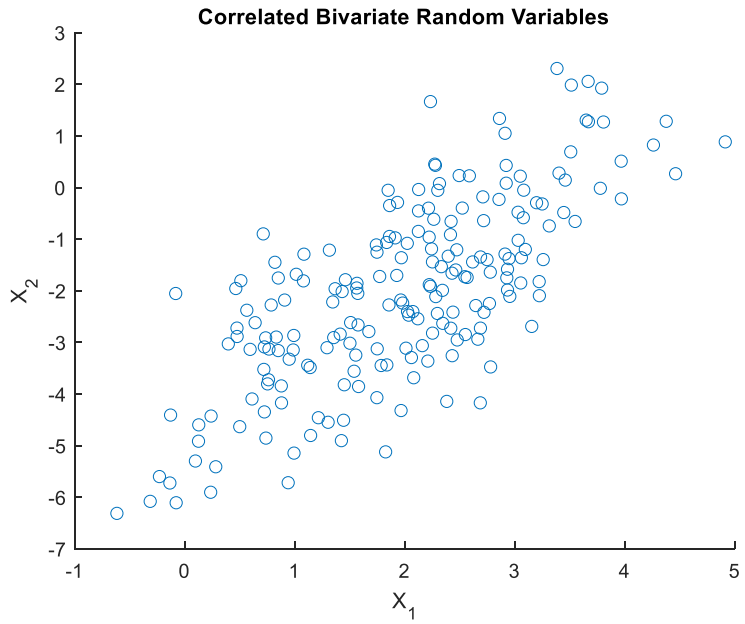
Matlab örnekleri

```
% PAGE 107
% Set the seed to 10.
rng(10)
% Generate 3 random variables.
r1 = rand(1,3)
% Generate 3 more random variables.
r2 = rand(1,3)
% Now, set the seed back to 10.
rng(10)
% Generate 6 random variables.
r3 = rand(1,6)
```

```
% PAGE 108
% Create a Q-Q plot of the earth data, where
% we compare it to an exponential distribution.
load earth
% Generate exponential variables.
% Use the mean of the data for the parameter.
rexp = exprnd(mean(earth),size(earth));
% Create the Q-Q plot.
plot(sort(earth),sort(rexp),'o')
xlabel('Data'),ylabel('Exponential')
title('Q-Q Plot of Earth Data and Exponential')
% Add a line that is estimated using quartiles.
% Get the first and third quartiles of the earth.
qeth = quantile(earth,[.25,.75]);
% Now, get the same for the exponential variables.
qexp = quantile(rexp,[.25,.75]);
% Fit a straight line. See Chapter 6 for more details.
p = polyfit(qeth,qexp,1);
pp = polyval(p,[5,max(earth)]);
hold on
plot([min(earth),max(earth)],pp)
```

Matlab örnekleri






```
% PAGE 110
% Generate two vectors of normal random variables
% with different means and variances.
% First is a variance of 4 and mean of 2.
x1 = randn(500,1)*sqrt(4) + 2;
% Next is a variance of 0.7 and a mean of -2.
x2 = randn(500,1)*sqrt(0.7) - 2;
% Construct a scatter plot.
plot(x1, x2, 'o')
title('Uncorrelated Normal Random Variables')
xlabel('X1'), ylabel('X2')
% Get the correlations
corrcoef([x1,x2])
% Generate bivariate correlated random variables.
mu = [2 -2];
covm = [1 1.25; 1.25 3];
X = mvnrnd(mu,covm,200);
% Show in a scatter plot.
scatter(X(:,1),X(:,2))
xlabel('X_1'),ylabel('X_2')
title('Correlated Bivariate Random Variables')
```

Tutorial on Matlab Basics

EECS 639

August 31, 2016

Arrays and Matrices

- **v = [-2 3 0 4.5 -1.5];** % length 5 row vector.
- **v = v';** % transposes v.
- **v(1);** % first element of v.
- **v(2:4);** % entries 2-4 of v.
- **v([3,5]);** % returns entries 3 & 5.
- **v=[4:-1:2];** % same as v=[4 3 2];
- **a=1:3; b=2:3; c=[a b];** → c = [1 2 3 2 3];

Arrays and Matrices (2)

- **x = linspace(-pi,pi,10);** % creates 10 linearly-spaced elements from –pi to pi.
- **logspace** is similar.
- **A = [1 2 3; 4 5 6];** % creates 2x3 matrix
- **A(1,2)** % the element in row 1, column 2.
- **A(:,2)** % the second column.
- **A(2,:)** % the second row.

Arrays and Matrices (3)

- **A+B**, **A-B**, **2*A**, **A*B** % matrix addition, matrix subtraction, scalar multiplication, matrix multiplication
- **A.*B** % element-by-element mult.
- **A'** % transpose of A (complex-conjugate transpose)
- **det(A)** % determinant of A

Creating special matrices

- **diag(v)** % change a vector v to a diagonal matrix.
- **diag(A)** % get diagonal of A.
- **eye(n)** % identity matrix of size n.
- **zeros(m,n)** % m-by-n zero matrix.
- **ones(m,n)** % m*n matrix with all ones.

Logical Conditions

- **==, <, >, <=, >=, ~=** (not equal), **~** (not)
- **&** (element-wise logical and), **|** (or)
- **find('condition')** – Return indices of A's elements that satisfies the condition.
- Example: **A = [7 6 5; 4 3 2];**

  find ('A == 3'); --> returns 5.

Solving Linear Equations

- **A = [1 2 3; 2 5 3; 1 0 8];**
- **b = [2; 1; 0];**

- **x = inv(A)*b;** % solves Ax=b if A is invertible.
  (Note: This is a BAD way to solve the equations!!! It's unstable and inefficient.)
- **x = A\b;** % solves Ax = b.
  (Note: This way is better.)

More matrix/vector operations

- **length(v)** % determine length of vector.
- **size(A)** % determine size of matrix.
- **rank(A)** % determine rank of matrix.
- **norm(A)**, **norm(A,1)**, **norm(A,inf)**
  % determine 2-norm, 1-norm,
  and infinity-norm of A.
- **norm(v)** % compute vector 2-norm.

For loops

- **x = 0;**
  for i=1:2:5 % start at 1, increment by 2
  x = x+i; % end with 5.
  end

  This computes x = 0+1+3+5=9.

While loops

- **x=7;**
  while (x > = 0)
  x = x-2;
  end;

  This computes x = 7-2-2-2-2 = -1.

If statements

- **if (x == 3)**
  disp(‘The value of x is 3.’);
  elseif (x == 5)
  disp(‘The value of x is 5.’);
  else
  disp(‘The value of x is not 3 or 5.’);
  end;

Switch statement

- **switch face**

  case {1}

  disp(‘Rolled a 1’);

  case {2}

  disp(‘Rolled a 2’);

  otherwise

  disp(‘Rolled a number >= 3’);

  end

- NOTE: Unlike C, ONLY the SWITCH statement between the matching case and the next case, otherwise, or end are executed. *(So breaks are unnecessary.)*

Break statements

- **break** – terminates execution of for and while loops. For nested loops, it exits the innermost loop only.

Vectorization

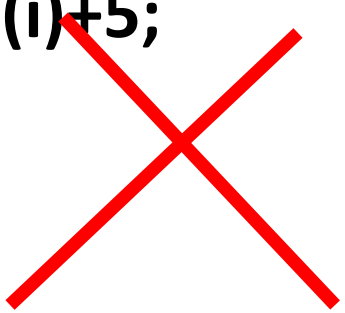
- Because Matlab is an interpreted language, i.e., it is not compiled before execution, loops run slowly.
- *Vectorized code runs faster* in Matlab.
- Example: **x=[1 2 3];**

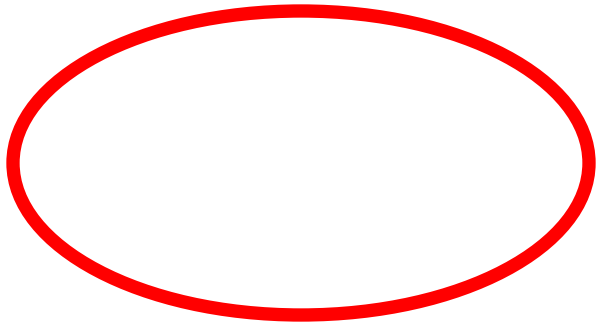
```
for i=1:3
   x(i) = x(i)+5;
end;
```

VS.

Vectorized:

```
x = x+5;
```





Scripts and Functions

- Two kinds of M-files:

  - **Scripts**, which do not accept input arguments or return output arguments. They operate on data in the workspace.

  - **Functions**, which can accept input arguments and return output arguments. Internal variables are local to the function.

M-file functions

- **function [area,circum] = circle(r)**

  % [area, circum] = circle(r) returns the

  % area and circumference of a circle

  % with radius r.

  area = pi*r^2;

  circum = 2*pi*r;

- Save function in circle.m.

M-file scripts

- **r = 7;**
  [area,circum] = circle(r);
  % call our circle function.
  disp([‘The area of a circle having…
  radius ‘ num2str(r) ‘ is ‘… num2str(area)]);

- Save the file as myscript.m.

Interactive Example (1)

- Write a Matlab program to compute the following sum

$$\sum 1/i^2, \text{ for } i=1, 2, \ldots, 10$$

  two different ways:

  1. 1/1+1/4+…+1/100
  2. 1/100+1/81+…+1/1.

Solution

```
% Forward summation
forwardsum = 0;
for  i=1:10
   forwardsum = forwardsum+1/(i^2);
end;

% Backward summation
backwardsum = 0;
for i=10:-1:1
   backwardsum = backwardsum+1/(i^2);
end;
```

Interactive Example (2)

- Write a Matlab function to multiply two
  n-by-n matrices A and B. (Do not use built-in functions.)

Solution

```
function [C] = matrix_multiply(A,B,n)
C = zeros(n,n);
for i=1:n
  for j=1:n
     for k=1:n
       C(i,j) = C(i,j) + A(i,k)*B(k,j);
    end;
  end;
end;
```

Can this code be written so that it runs faster?

Hint: Use vectorization.

Solution

- Script to use for testing:

```
n = 10;
A = rand(n,n);
B = rand(n,n);
C = matrix_multiply(A,B,n);
```

Exercises

- Write a MATLAB function for calculating the Euclidean distance between two points in the n dimensional space. The points are given as the input arguments a and b. Both should be arrays with n elements. It should not matter for your function whether either input is a row or a column. Demonstrate the work of your function by an example.
- Write a MATLAB function for calculating the Euclidean distance between two two-dimensional arrays A and B given as input arguments. A is of size N ×n, and B is of size M ×n. The function should return a matrix D of size N × M where element d(i, j) is the Euclidean distance between row i of A and row j of B.
- Write an inline function that will calculate $6x - 4y + xy + \cos^2(x - k)$. Assume that x and y may scalars, vectors or matrices and that all operations should be carried out element-wise.

- Write a MATLAB function for checking if a given point (x, y) is within the square with a bottom left corner at (p, q) and side s. The input arguments are x, y, p, q and s, and the output is either true (the point is in the square) or false (the point is not in the square).
- Write a non-recursive MATLAB function to calculate the Fibonacci sequence and return the number with a specified index, for example, the 4th number in the series (this number is 5).
- Write a recursive MATLAB function to calculate the Fibonacci sequence and return the number with a specified index.
- Write a short MATLAB function to find out whether a given number (up to 1,000,000) is a prime number. The function should return true or false. Bear in mind that 1 is not considered a prime number. (Hint: Use the brute force approach and divide the number (K) by all integers from 2 to K-1. Check the remainders for 0s.) Subsequently, apply this function to list all prime numbers between 1 and 100.
- Write your own function for the bubble sort algorithm and demonstrate its work.

Nesneye Yönelik Programlama

Giriş

- Procedural Programming
  - Bilgisayara bir iş yaptırmak için yazılan bir dizi kod
- Nesneye Yönelik Programlama
  - Objeler: Veri alanları ve ilgili metotlar
  - Veri ve metotlar birbiriyle ilişki içindedir
  - Objeler birbiriyle ilişki içindedir
  - C++, Objective-C, Smalltalk, Java, C#, Perl, Python, Ruby, PHP
- Modülerlik
- Soyutlama (abstraction)
- Kodun anlaşılırlığı, idamesi, gelişimi

Kavramlar

- Sınıf (Class): Nesne yaratmak, özellik ve yöntem tanımlamak için bir şablon
- Nesne (Object): Bir sınıf durumu: durum (properties) ve davranış (methods)
- Özellikler (Properties): Nesneyle birleşik olan veri
- Yöntemler (Methods): Bir sınıfta tanımlanan ve nesneyle ilişkili olan davranışlar (functions)
- Özellik (Attributes): modify behavior of classes and class components
- Kalıtım (Inheritance): Başka bir üst sınıfta (superclass) türetilen bir nesne (subclass)
- Çok biçimlilik (Polymorphism): single interface to entities of different types

Sınıf bileşenleri

- «classdef» block
  - Sınıf tanımı, attributes, ve üst sınıflar (superclasses)
- «properties» block
  - Sınıf örneği ile ilgili bütün özellikler
  - Bütün özellikler ve varsayılan değerler
- «methods» block
  - Sınıf ve parametrelerle ilgili yöntemler tanımlanır
  - İlk yöntem sınıfla aynı adı taşımalıdır.(constructor)
- «event» block
- «enumeration» block
- http://www.mathworks.com/help/matlab/matlab_oop/class-components.html

Matlab’da Nesneye Yönelik Programlama Nasıl Gerçekleştirilir?

- Nesneye yönelik programlama nesne (object) adındaki yapılara dayanır
- Bu nesneler, verinin ve o veri üzerinde çalışan çeşitli fonksiyonların özelliklerini birleştirmeye yarar
- Bu fonksiyonlara yöntem (method) denir.
- Ör. Hareket eden bir cismin katettiği mesafeyi kaydeden bir nesne yaratabilirsiniz.
  - Bu nesne daha karmaşık sistemlerin yapı taşı olarak kullanılabilir.
- Nesneye yönelik programlama kodlarınızı yönetmeye yarar.
- Kodları sınıf (class) ve fonksiyonlar halinde organize eder.
- Zaman içinde gerekli değişiklikleri yapmanızı kolaylaştırır
- Kodunuzdaki fazlalıkları, tekrarları engeller

Kapsülleme (Encapsulation)

- Kapsülleme nesneler , sınıflarla gerçekleşir
- Bir veri ve o veri üzerindeki işlemler kapsüllenir
- Oluşturulan sınıfın iç işleyişini dışarıdan gizler
- Dışarıdan müdahale için seçenekler vardır (Access modifiers)
  - Bir sınıfın özelliğine başka sınıflardan erişim
  - Bir yöntemi hangi yöntemler ve fonksiyonlar çağırabilir
- Üç tür erişim
  - Açık (Public) - unrestricted access (default).
  - Korumalı (Protected) – Aynı sınıfın yöntemlerinden veya alt sınıflardan erişim
  - Özel (Private) – Sadece aynı sınıfın metotlarından erişim mümkün

Matlab’da kapsülleme

```
classdef  Employees
properties
        name
        baseSalary
end
```

- Yukarıdaki kod employees adında temel bir sınıf oluşturur
- Özellikler: name ve baseSalary
  - Erişim parametresi yok: demek ki public (açık)
- Ör: özel olabilecek özellikler: doğum günü, adres, TC kimlik no
- Özelliğe erişimi kısıtlamak: GetAccess ve SetAccess.
  - GetAccess: Okumaya izin verir
  - SetAccess: Özelliğe değer atamaya izin verir

encapsulation

```
properties(Access=private)
dateOfBirth
address
end
properties(GetAccess = {?Engineer, ?Sales,
?TestEngineers})
dateOfJoining = 01/23/2020
department
end
```

```
methods
function obj = employees(name, baseSalary)
obj.name = name;
obj.baseSalary = baseSalary;
end
function r = getname(obj)
r = obj.name;
return;
end
end
end
```

- Katılım tarihi ve bölüm bilgisi Engineer, Sales ve TestEngineers alt sınıflarından erişilebilir.
- İki tane yöntem (method): constructor (employees) ve function getname.

- Nesneyi başlatmak için constructor'ı çağırırız. Nesneyi obj olarak adlandırırız ve employees fonksiyonunu çağırırız, ve name, baseSalary parametrelerini veririz.
- Aşağıdaki komutla:

```
obj = employees("neha", 1000)
```

- Bu komutu «command window»'dan çağırdığımızda objeyi elde etmiş oluruz.

```
>> obj = employees("neha", 1000)

obj =

  employees with properties:

          name: "neha"
    baseSalary: 1000
```

Kalıtım (Inheritance)

- Burada , sınıfı hiyerarşilere ayırırız.
  - Base class, superclass: Employees (param: name baseSalary)
  - Subclasses: Engineer, Sales (ekstra param: Product, commission)
- Kolaylık sağlar
- Superclass'ların açık özelliklerini miras alırlar

```
classdef Engineers< employees
properties
products
end
properties(Access=private)
team
end
```

Inheritance

- Küçük işareti (<) alt sınıf, üst sınıf ilişkisini ifade eder
- We can access the superclass constructor from the base class by the center below:

```
methods
function objE = Engineers(name, baseSalary,
products)
objE@employees(name, baseSalary)
objE.products = products;
return
end
function y = Salary(objE, noOfHours)
y = objE.baseSalary*noOfHours;
return end end end
```

Kalıtım

- Satış için de bir alt sınıf yapalım:

```
classdef Sales < employees
properties
commission
region
end
methods
function objS = Sales(name, baseSalary, region, commission)
objS@employees(name, baseSalary)
objS.region = region;
objS.commission = commission;
return
end
function S = Salary(objS, noOfHours)
S = (objS.baseSalary*noOfHours)*objS.commission;
end end end
```

Kalıtım

- The objE is the output of the subclass constructor and the read superclass name and any other argument required by the argument superclass.
- Matlab'da properties «sınıf» komutu ile bir sınıfın özelliklerini öğrenebilirsiniz. Bu sınıf «çalışanlar» üst sınıf`properties Engineers` özelliklerini devralmıştır.
-

```
>> properties Engineers

Properties for class Engineers:

    products
    name
    baseSalary
```

Kalıtım

- Üst sınıftan name ve base salary parametrelerini alıp onlara yeni değerler atayabiliriz.
- Employee ‘neha’. Ürün ve satışını belirtebiliriz
- İlk olarak objE and objS nesnelerini tanımlayalım.
- Tanım:

```
objE = Engineers(obj.name, obj.baseSalary, "simulink")
```

```
objE =

  Engineers with properties:

     products: "simulink"
         name: "neha"
   baseSalary: 1000
```

- Parametreler: name, salary, ve product (« Simulink»).

```
objS = Sales(obj.name, obj.baseSalary, "NA", 500)
```

- Parametreler: name, salary, region (NA), ve commission (500).

Çok biçimlilik (Polymorphism)

- İki alt sınıf (Mühendis ve Satış) aynı yöntemi (maaş hesabı) içerebilir ama içerikleri farklı olabilir.

```
% function for salary for an engineer
function y = Salary(objE, noOfHours)
y = objE.baseSalary*noOfHours;
return
end
% function for salary for sales
function S = Salary(objS, noOfHours)
S = (objS.baseSalary*noOfHours)*objS.commission;
end
```

- Her sınıfın nesnesini başlatalım (`objE` ve `objS`.)

Çok biçimlilik

- Komut penceresine aşağıdaki komutu yazalım:

```
SE = Salary(objE, 10) %Calculates the salary
SS = Salary(objS,10) %calculates the salary from sales
```

```
objS =

  Sales with properties:

    commission: 500
        region: "NA"
          name: "neha"
    baseSalary: 1000
```

- 10 değeri saat sayısıdır. Aynı isimde iki farklı fonksiyonun (yöntemin) farklı sonuç verdiğini görüyoruz

```
>> SE = Salary(objE, 10)

SE =

       10000

>> SS =  Salary(objS,10)   %calculates the sales

SS =

       10500
```

Soyutlama (Abstraction)

- Soyutlama kapsüllemenin bir varyasyonundur.
- Salary fonksiyonu sadece temel sınıfta tanımlanır ve alt sınıf aynı fonksiyona sahip olur.
- Soyutlama işlemini temel sınıfı «abstract» olarak tanımlayarak yaparız.

```
classdef(Abstract) Employees
properties
name
baseSalary
end
methods
Salary(obj)
getName(obj)
end
end
```

- A class is abstract when we either declare the attribute abstract, an abstract method, or property.
- Yöntem ve özellikler abstract sınıfta sadece tanımlanır, ama gerçekleştirilmesialt sınıfta olur.

Sonuç

- Nesneye yönelik programlama kodunuzu sınıflar ve alt sınıflar olarak organize eder, düzenli hale getirir ve anlaşılırlığını artırır
- Nesneye yönelik programlama aynı zamanda kodunuzunda zamanla yapacağınız değişikliklerin olumsuz etkilerini önler. Aynı kodun tekrar kullanılabilirliğini sağlar

- Employees: https://www.youtube.com/watch?v=kz4zYECb8AA
- Sensors: https://www.youtube.com/watch?v=XOkn-zxhwc4&t=1535s
- https://www.mathworks.com/company/newsletters/articles/introduction-to-object-oriented-programming-in-matlab.html

- İncrease senssorzeropad (128, 256, 2048 gets closer to -10 degrees)
- Make sensrsrnumdetector 32 then 2048 it doesntchange
- Should’nt change the speed of light
- Can change numdetector instead of numdetectors
- Very flexible but it can go wrong
- Change sensor.numDetector to 96 can find more baloons
- T= target; t.AoA= -10 ; t.range=1e8 ; t.signal=signal
- İsa(t,»struct»); isa(t.signal,»struct»)
- Balloon=blip; balloon(1)
- Balloon.identify(AoA) wrong; Balloon.identify(obj.AoA) wrong
- balloon(1).aoa=13 hata verir
- Problem: sensor’de her özelliğe erişmek iyi olmayabilir (bazı parametreler ve yöntemler içerdedir)
- Numdetector, minzeropad public