Módulo IX: Estadística y tratamiento de datos

- Análisis estadístico básico.
- Objetos de tipo “Conjunto de datos”.
- Histograma y distribución de probabilidad.
- Ajuste de curvas.

Dataset / Conjunto de Datos

```
% Leer archivo de datos. La hoja .csv tiene datos separados por comas (,)
% entonces se asigna ',' de delimitador
clear;
clc;
ds = dataset('file', 'energy-consumption.csv', 'delimiter', ',');
```

```
Warning: Variable names were modified to make them valid MATLAB identifiers.
```

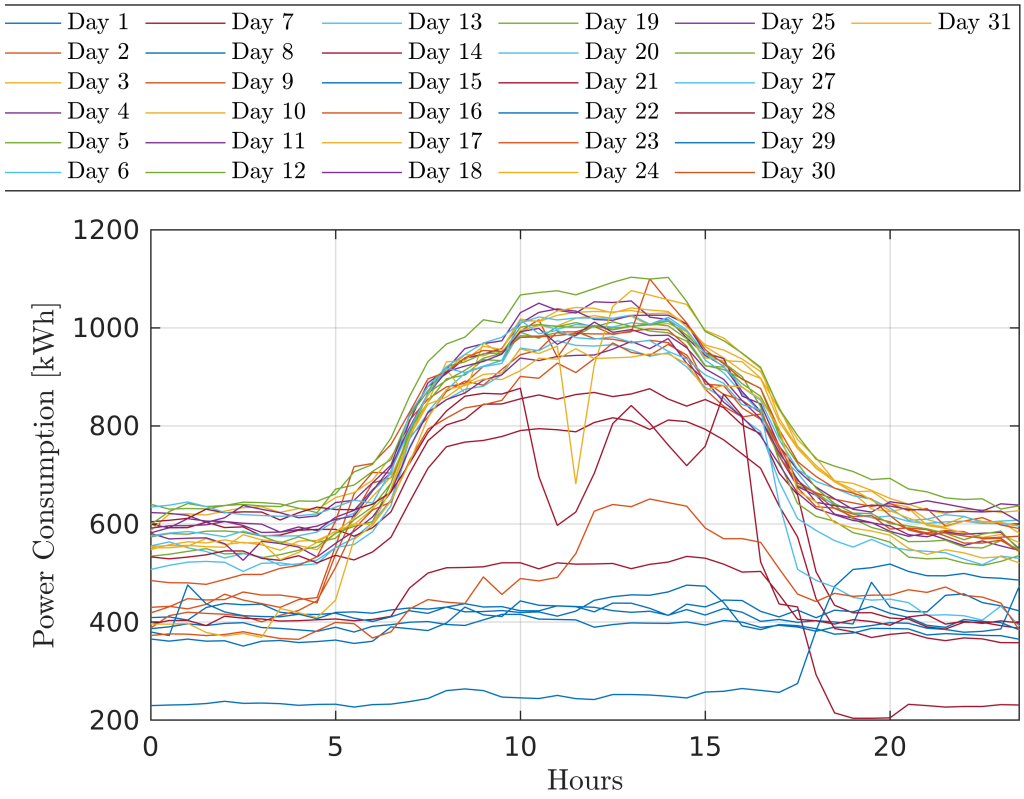
```
% Formato Latex en gráficas
set(0, 'defaultTextInterpreter', 'latex');
set(0, 'defaultLegendInterpreter', 'latex');
```

Graficar series de datos

Ex. 1. Graficar consumo de energía cada media hora para todos los días de agosto.

```
consumo = ds(:, 5 : end-1); % Consumo del primer mes

figure(1)
hold on
hours = 0 : 0.5 : 23.5; % vector de tiempo
labels = {};
% Generar múltiples strings para la legend:
for i = 1 : 31
    plot(hours, consumo(i, :))
    labels{i} = ['Day ', num2str(i)];
end

xlabel('Hours')
ylabel('Power Consumption [kWh]')
legend(labels,'Location', 'northoutside', 'NumColumns', 6)
grid on
xlim([0, 23.5])
```

```
box on
```

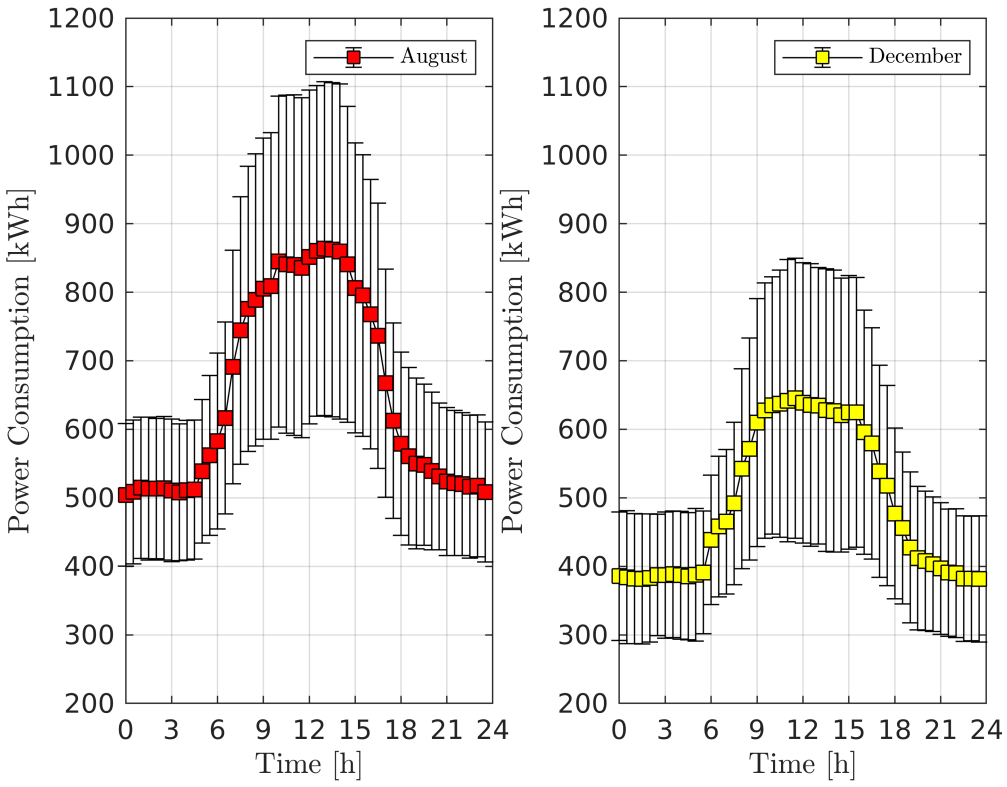


Promedio y desviación estándar

Ex. 2. Graficar la media y desviación estándar del consumo en agosto y diciembre usando barras de error.

```
aug_mean = mean( double(ds(1:31, 5:end-1)) ); % Calcular la media de cada día de agosto
aug_std = std( double(ds(1:31, 5:end-1)) ); % Calcular la desviación estándar de cada d
figure(2)
subplot(1,2,1)
% Graficar media con barras de error:
errorbar( hours, aug_mean, aug_std, '-s','MarkerSize',7,'Color', 'k', 'MarkerFaceColor
legend('August')
xlim([0, 24])
ylim([200,1200])
xticks(0 : 3 : 24)
grid on
box on
ylabel('Power Consumption $\rm{[kWh]}$')
xlabel('Time $\rm{[h]}$')

dec_mean = mean( double(ds(127:158, 5:end-1)) );
dec_std = std( double(ds(127:158, 5:end-1)) );
```

```
subplot(1,2,2)
errorbar( hours, dec_mean, dec_std, '-s','MarkerSize',7,'Color', 'k', 'MarkerFaceColor
legend('December')
xlim([0, 24])
ylim([200,1200])
xticks(0 : 3 : 24)
grid on
box on
ylabel('Power Consumption $\rm{[kWh]}$')
xlabel('Time $\rm{[h]}$')
```



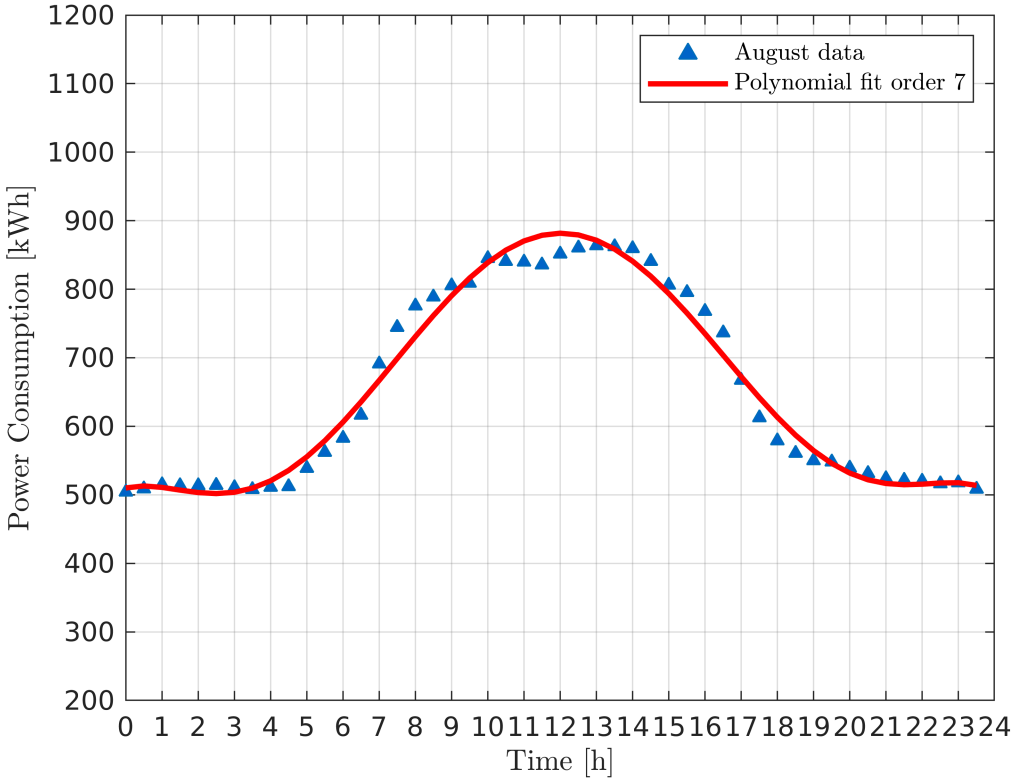
Ajuste de curvas

Ex. 3. Encuentre un polinomio que ajuste los datos de consumo promedio en enero.

```
% Datos a ajustar:
mean_consumption_august = mean(double(consumo(1:31, :)));
standard_deviation_august = std(double(consumo(1:31, :)));

% Ajuste polinomial:
order_polinomial = 7;
parameters_august = polyfit(hours, mean_consumption_august, order_polinomial);
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X values, reduce the degree of
the polynomial, or try centering and scaling as described in HELP POLYFIT.
```

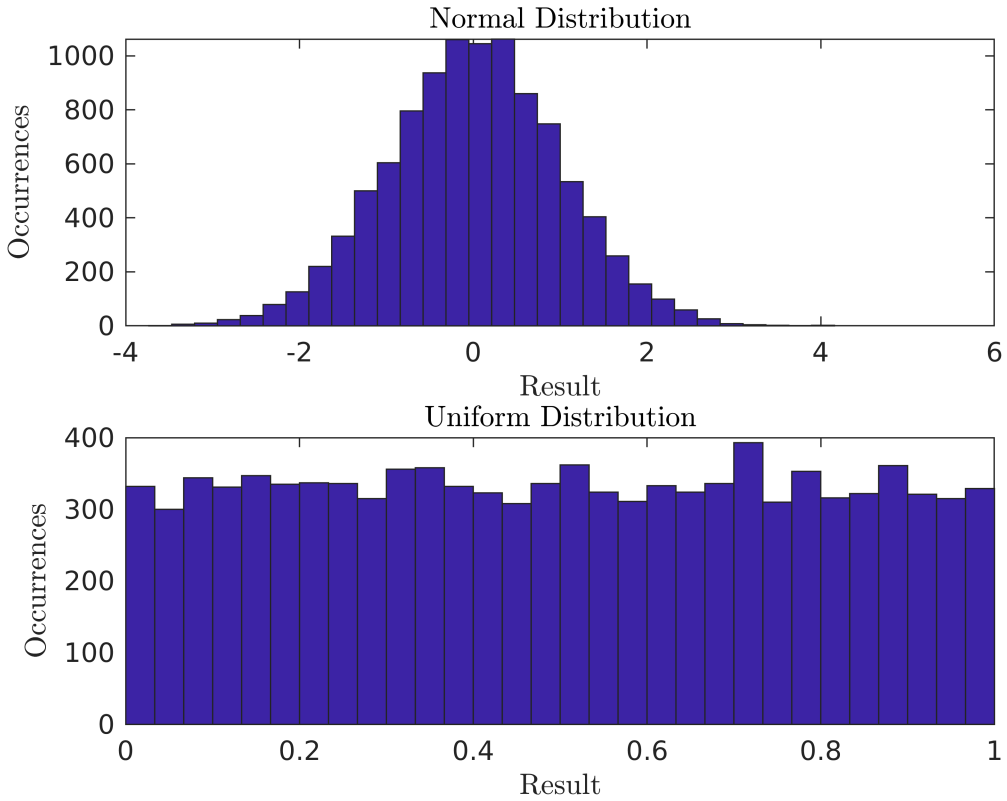
```
yfit_august = polyval(parameters_august, hours);



figure(3)
% Dibujar datos muestreados:
scatter(hours, mean_consumption_august, 20, ...
    'Marker', '^', 'MarkerFaceColor', [0.0, 0.4, 0.8]);
hold on
% Dibujar línea de ajuste:
plot(hours, yfit_august, 'r', 'Linewidth', 2)
legend({'August data', ['Polynomial fit order ', num2str(order_polinomial)]})
xlim([0 24])
ylim([200 1200])
xticks(0:24)
grid on
box on
ylabel('Power Consumption [kWh]')
xlabel('Time [h]')
```



Histograma y distribuciones de probabilidad

Ex. 4. Lance muchos números aleatorios desde las distribuciones normal y uniforme y observe el histograma de los valores arrojados.

```
clear; clc;
% Aleatorios con distribución normal (media = 0, std = 1)
data_normal = randn(10000, 1);
% Aleatorios con distribución uniforme (valores entre 0 y 1 equiprobables)
data_uniform = rand(10000, 1);

figure(4)
subplot(2,1,1)
hist(data_normal, 30) % Graficar histograma con 30 separaciones
title('Normal Distribution')
xlabel('Result')
ylabel('Occurrences')
subplot(2,1,2)
hist(data_uniform, 30)
title('Uniform Distribution')
xlabel('Result')
ylabel('Occurrences')
```



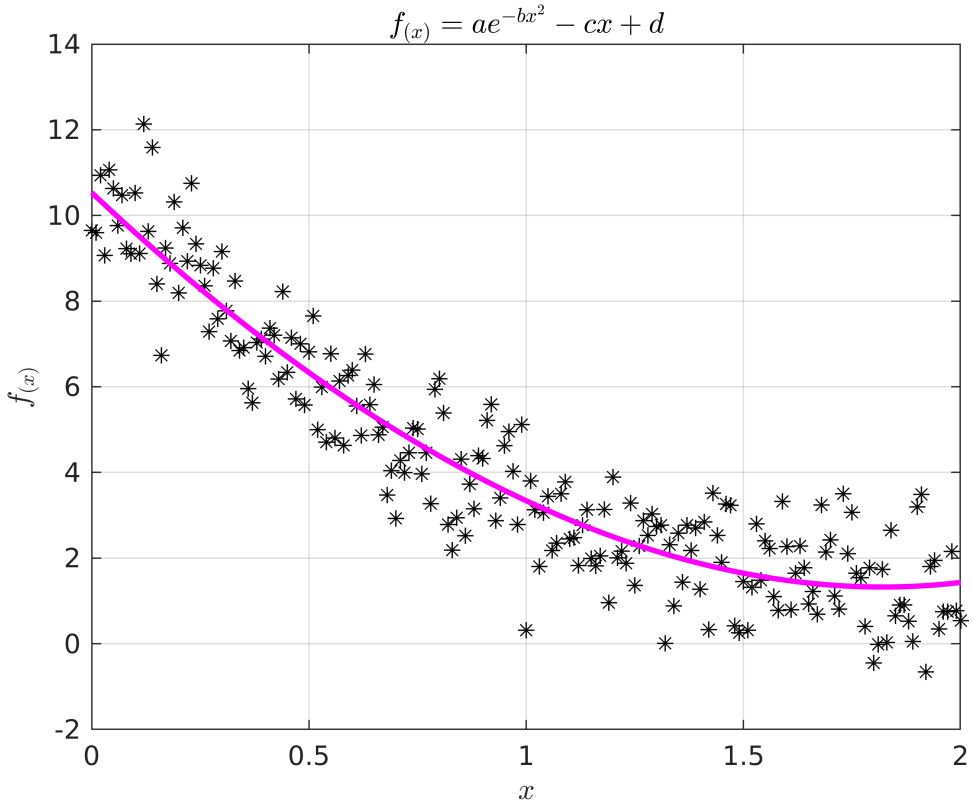
Ajuste con distintas funciones

Ex. 5. Encuentre los parámetros que mejor ajustan una curva a unos datos muestreados con ruido.

```
xs = 0 : 0.01 : 2;
params = [5, 3, 2, 5];
y = kind_of_func( params, xs );

y = y + randn( 1, length(y) ); % Agregar ruido aleatorio

params_guess = [-5, -3, 1, 0]; % Una primera estimación de los parámetros

% Hacer ajuste:
params_fitted = lsqcurvefit(@kind_of_func, params_guess, xs, y);
```

```
Solver stopped prematurely.

lsqcurvefit stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 400 (the default value).
```

```
% Verificar la curva obtenida con los parámetros estimados:
y_fitted = kind_of_func(params_fitted, xs);

figure(5)
clf
scatter( xs, y, '*k' )
hold on
plot( xs, y_fitted, 'm', 'Linewidth', 2)
grid on
box on
title('$f_{(x)} = ae^{-bx^2} - cx + d$')
xlabel('$x$')
ylabel('$f_{(x)}$')
```




```
function y = kind_of_func( params, x )
% La función que se desea usar como base del ajuste
    a = params(1);
    b = params(2);
    c = params(3);
    d = params(4);
    y = a.*exp(-b*x.^2) - c.*x + d;
end
```