XMas Homework

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Exercise 1](#_8o1wyh30nnh6)

[Part 1](#_huagqvsmxet)

[Part 2](#_u0wxs1bswk52)

[1) Create the sampler xs](#_pax9m1tdnb6s)

[2) Generate {1, x, x2, f}](#_ia33g3bc46b1)

[3) Perform the regression of f onto 〈1, x, x2〉.](#_u7uqsqtfiygn)

[4) Plot the result](#_toetk27rw9dz)

[Exercise 2](#_whwqc6yxa3u7)

[1) Find the cumulative sells](#_90yaoqpwjdn)

[2) Fit this curve using a polynomial](#_dbnllqytri1w)

[Exercise 3](#_udlewapuqh6a)

[1) Read the data](#_7y30h8vudb8)

[2) For each column of winner finding the different outcomes and their relative frequency](#_ox59eg5sn190)

[3) Do the same with the column's stars individually and with its sum](#_dsljnc9jkwru)

[4) Finding the relative frequency of the sum of the 5 winners](#_l7obpfq57h1k)

[5) Draw a new random winner combination](#_3jn7hqrnrj3)

[6) Do the same with the stars](#_2viqbdyb6aml)

[7) Plot on the screen 10 different winning combinations](#_ne8fa69yd872)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Exercise 1

## Part 1

Perform the continuous regression of f(x) onto the subspace <1, x, x2> using the symbolic tools in Matlab. Plot the result (the function and the fitting).

x=sym('x','real');

f=log(x);

%[1,e]

a=1;

b=exp(1);

U=[1,x,x^2];

**% Fist, calculate the gram matrix:**

B=U'\*U;

G=int(B,a,b); %Gram matrix is the integral of B (of each term)

**% Calculate the comparison vector (the scalar product between f and each vector of the basis):**

c=U'\*f;

d=int(c,a,b);

% **Calculate the coordinates of the projection:**

coord=inv(G)\*d(:);

% **And now the projection(polynomial of degree 2)**

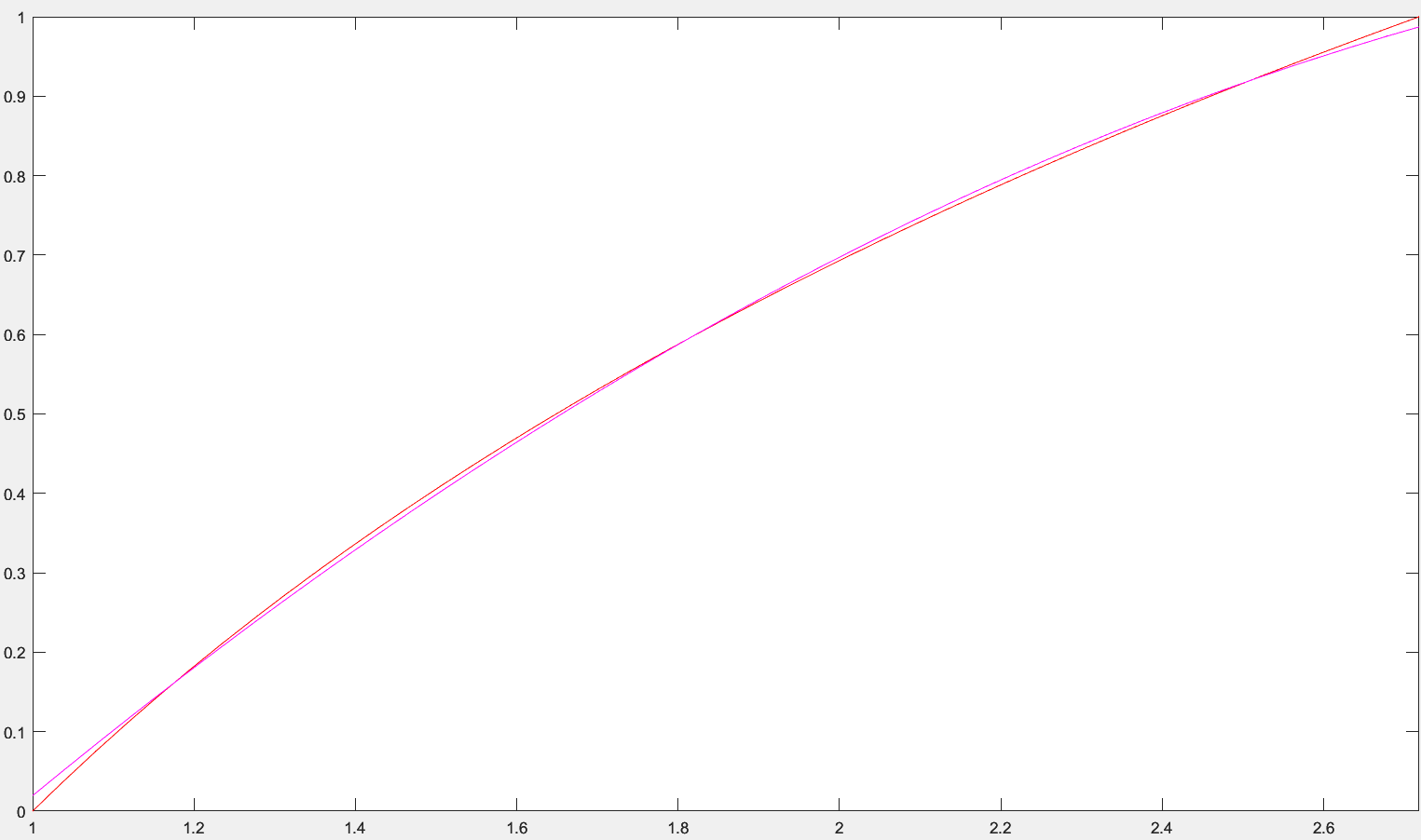
p2=U\*coord(:);

% **Plotting f(x) and the projection**

fplot(f,[a b],'r');

hold on

fplot(p2,[a b],'m')



## Part 2

Perform the discrete regression of f(x) onto <1, x, x2>. For that purpose:

1. Introduce as regular sampler of the interval [1, e] using 1000 points. Call this sampler xs.

2. Evaluate f(x) on xs, and the basis functions 1, x, x2 to generate {1, x, x2, f}.

3. Perform the regression of f onto 〈1, x, x2〉.

4. Plot the result (the function and the fitting).

### 1) Create the sampler xs

% The number of samples (as a test, I recommend putting c=100, because it’s faster

c=1000;

x=sym('x','real');

f=log(x);

**% Creating a vector (xs) with 1000 points of f in [1,e] uniformly distributed**

xs=[];

for index=(linspace(1,exp(1),c))

xs=[xs,subs(f,x,index)];

end

### 2) Generate {1, x, x2, f}

**% fitting=parabola=a0\*(1\_vector)+a1\*(x\_vector)+a2\*(x^2\_vector)**

**% The fitting lays is a hyperplane of space 1000 with dimension 3.**

**% fitting belongs to the span of <1,x,x^2>=U. U is a subspace of**

**% dimension 3 in R1000**

**% ||xs-fitting||=minimum. fitting= the orthogonal projection of xs onto U**

**% projection=[1,xi,(x^2)i](:)\*[a0,a1,a2](:)=U[a0,a1,a2](:)**

**% Creating U**

x=[1:1:c];

x2=x.^2;

U=[ones(c,1),x(:),x2(:)];

### 3) Perform the regression of f onto 〈1, x, x2〉.

**% U[a0,a1,a2](:)=projection**

**% Overdetermined linear system: #equations m=1000>> number of unknown=3**

**%<->least squares sense U'\*U[a0,a1,a2](:)=U'\*projection**

A=U'\*U; % Gram matrix

b=U'\*xs(:); % Comparison (U'\*y\_observe)

**% Calculate the coordinates of the projection**

coord=inv(A)\*b(:);

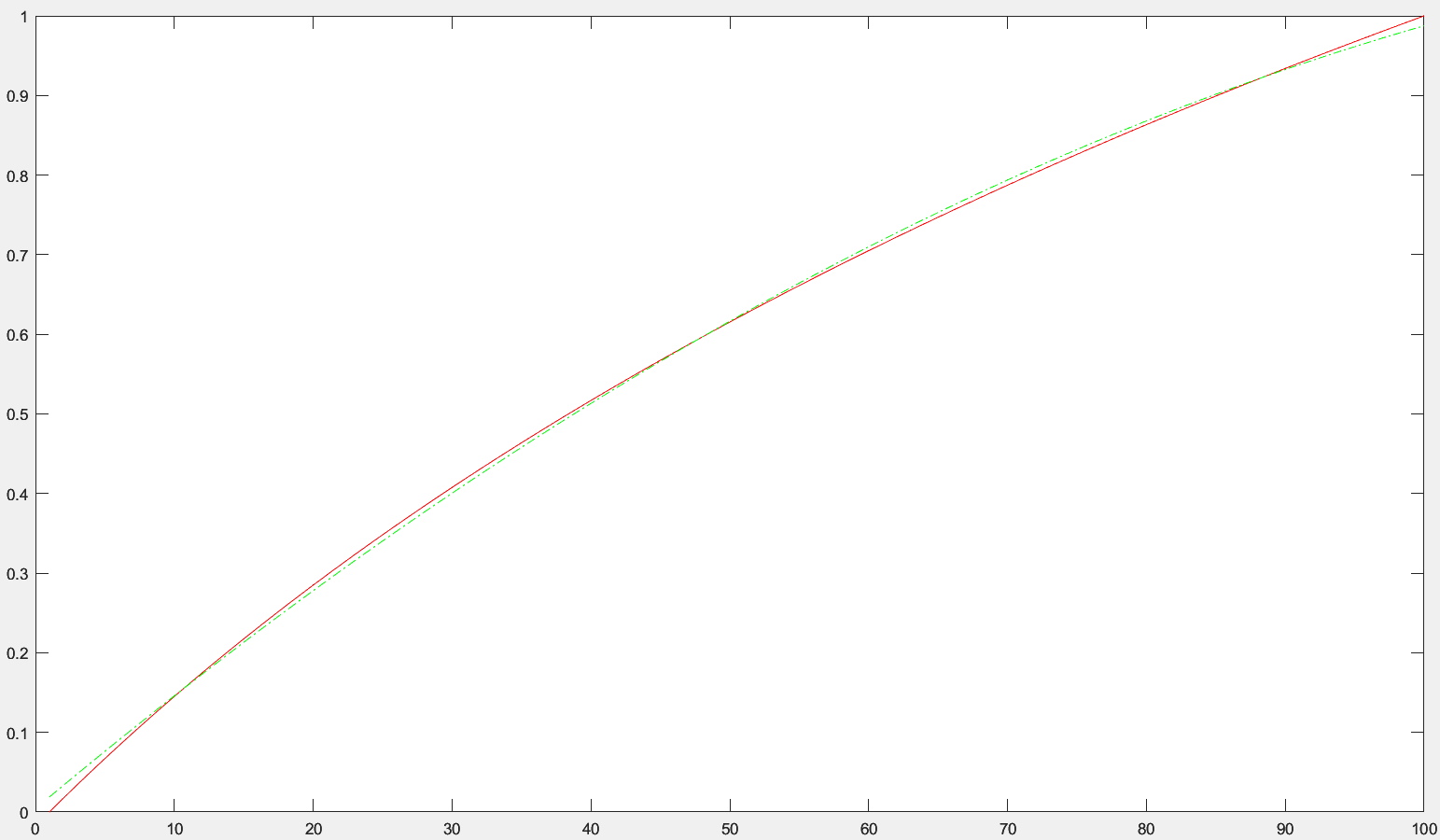
projection=U\*coord(:);

### 4) Plot the result

plot(xs,'r');

hold on

plot(projection,'g-.');



(I did it with c=100 samples)

# Exercise 2

A company invests 8000 dollars in marketing to acquire new clients. The company also has the following information:

1. The cost of getting a new client is 40 dollars
2. The average sales per customer (fee) is 30 dollars/month
3. The percentage of clients that stay after the first month (churn):

churn(%)=[100, 85, 60, 50, 30, 25, 20, 9, 5, 1, 0]

months=[1,2,3,4,5,6,7,8,9,10,11]=[1:11]

1. Write a program to find the cumulative sells curve for every month till all the clients churns:

sells(1), sells(1)+sells(2), sells(1)+sells(2)+sells(3)

1. Write a program to fit this curve using a polynomial. Define the degree.

## 1) Find the cumulative sells

months=[1:11];

churn=[100, 85, 60, 50, 30, 25, 20, 9, 5, 1, 0];

**% I calculate the total clients (clients at the first month): the firm invest 8000$ and each new client cost 40$**

client\_month=8000/40;

**% I calculate the total clients (clients at the first month): the firm invest 8000$ and each new client cost 40$**

sum\_sells=0;

sells=[];

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**% We can consider that in the 3rd month, that are 60% of the people stay in the 2nd month…**

for index=1:length(months)

% **Clients in the month**(I truncate): clients in that month multiply by the % of clients that stay that month

client\_month=fix(client\_month\*churn(index)/100);

% **Sells in the month**: The clients in that month multiplied by the fee

sells\_month=client\_month\*30;

% **Accumulated sells**: The sum of all the sells of the months before plus the sells of this month

sum\_sells=sum\_sells+sells\_month;

% **Sells**: It’s an array where in each month it’s all the accumulated sells until that month

sells(index)=sum\_sells;

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**% But for this exercise I would do it considering in the 3rd month that are 60% of the people of the 1st month**

for index=1:length(months)

sells\_month=client\_month\*churn(index)/100\*30; %sells in the month

sum\_sells=sum\_sells+sells\_month; %acumulative sells

sells(index)=sum\_sells; %put in each month the acumulative sells

end

## 2) Fit this curve using a polynomial

**% I do the same as the exercise before:**

x=sym('x','real');

le=length(months); % It’s the number of samples

degree=4;

x=[1:1:le];

U=[ones(le,1),x(:)];

**% Calculating U depending on the degree set (adding x^2 - degree=2; adding x^2 and x^3 - degree=3**

for index=2:degree

xn=x.^index;

U=[U,xn(:)];

end

A=U'\*U; % Gram matrix

b=U'\*sells(:); % Comparison between U and the sells (y\_observe)

**% Calculate the coordinates of the projection**

coord=inv(A)\*b(:); %coordinates of the projection

projection=U\*coord(:);

**% Plotting the sells and the projection**

plot(sells,'r');

hold on

plot(projection,'g-.');

I also, made a function to ask for the degree of the fitting and do the same as before:

function [projection] = Exercise\_2\_Function\_UO294067(degree,sells,months)

**% Calculates the discrete regression of the accumulative sells on a period**

**% of time (months) onto a polynomial with a given degree**

% -**degree**: the given degree of the fitting

% -**sells**: It’s an array where in each month it’s all the accumulated sells until that month

% -**months**: number of months of the sells

x=sym('x','real');

le=length(months); % It’s the number of samples

x=[1:1:le];

U=[ones(le,1),x(:)];

for index=2:degree

xn=x.^index;

U=[U,xn(:)];

end

A=U'\*U; % Gram matrix

b=U'\*sells(:); % Comparison between U and the sells(y\_observe)

coord=inv(A)\*b(:); % Coordinates of the projection

projection=U\*coord(:);

end

\_\_\_\_\_\_\_\_\_

Now the program done using de function

degree=input("Write the degree: "); **% Asking for the degree**

**% Calling the function** with the sells and month variables already defined before

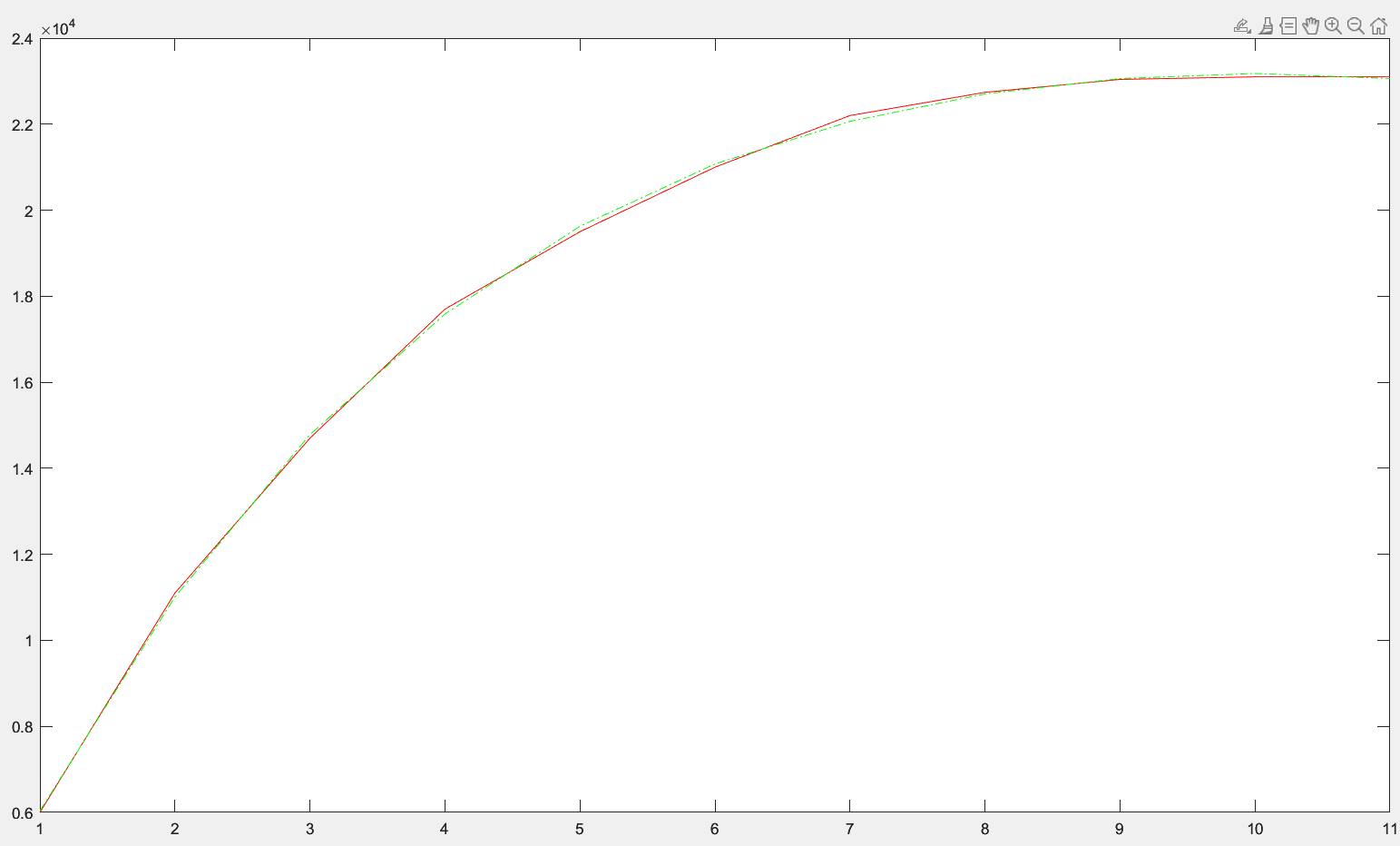
projection=Exercise\_2\_Function\_UO294067(degree,sells,months);

**% Plotting** the sells and the projection

plot(sells,'r');

hold on

plot(projection,'g-.');



(with degree 4)

# Exercise 3

<https://www.lotoideas.com/euromillones-resultados-historicos-de-todos-los-sorteos/>

Create an algorithm that generates new intelligent best on the Euromillones draw

1. Read the data using xlsread, and recover the different fields in different arrays: fecha(date), comb.ganadora(winners), estrellas(stars)
2. For each column of winner finding the different outcomes and their relative frequency, that is the percentage of times that they appear on the draw.
3. Do the same with the column's stars individually and with its sum
4. Also finding the relative frequency of the sum of the 5 winners (comb.Ganadora)
5. Considering the results 2, 3 and 4, draw a new random winner combination. Explain how you design the drawing algorithm. You are free on the design.
6. Do the same with the stars
7. Plot on the screen 10 different winning combinations

## 1) Read the data

% **Extracting the information** from the excel

T=xlsread('Euromillon.xlsx');

[~,data]=xlsread('Euromillon.xlsx'); % Extract the text part to then extract the dates

data=data(2:end,1); % Only taking the dates

winners=T(:,1:5);

stars=T(:,7:8);

## 2) For each column of winner finding the different outcomes and their relative frequency

First **count the numbers** (they are from 1 to 50) for each column

col1=winners(:,1); % This is the first column of the winners

% This is the array where in each position I will **store the number of times that the number appears**: position 1- number of time the 1 appears in the column 1

number\_col1=[];

% The same for the rest of columns:

col2=winners(:,2);

number\_col2=[];

col3=winners(:,3);

number\_col3=[];

col4=winners(:,4);

number\_col4=[];

col5=winners(:,5);

number\_col5=[];

% Now for each number j=[1,50], I **count with the function “numel” the number of times that j appears on in each column**, and store it in number\_col

for j=1:50

number\_col1=[number\_col1,numel(col1(col1==j))];

number\_col2=[number\_col2,numel(col2(col2==j))];

number\_col3=[number\_col3,numel(col3(col3==j))];

number\_col4=[number\_col4,numel(col4(col4==j))];

number\_col5=[number\_col5,numel(col5(col5==j))];

end

% Now, I found the **relative frequency** of each number in each column by dividing the number of times each number appears in that column, by the total amount of numbers stored in that column:

freq\_col1=number\_col1/length(col1);

freq\_col2=number\_col2/length(col2);

freq\_col3=number\_col3/length(col3);

freq\_col4=number\_col4/length(col4);

freq\_col5=number\_col5/length(col5);

## 3) Do the same with the column's stars individually and with its sum

First **count the stars** (they are from 1 to 12), and the sum

% I do **the same as before**

col\_s1=stars(:,1);

number\_col\_s1=[];

col\_s2=stars(:,2);

number\_col\_s2=[];

for j=1:12

number\_col\_s1=[number\_col\_s1,numel(col\_s1(col\_s1==j))];

number\_col\_s2=[number\_col\_s2,numel(col\_s2(col\_s2==j))];

end

% Now I sum the 2 columns and then I **calculate the number of times each sum appears**

sum\_stars=col\_s1+col\_s2;

number\_sum\_stars=[];

% Now I count from the minimum sum to the max sum the numbers of times it appears

for j=min(sum\_stars):max(sum\_stars) % min(sum\_stars)=3, max(sum\_stars)=23

number\_sum\_stars=[number\_sum\_stars,numel(sum\_stars(sum\_stars==j))];

end

% Now the **frequency of the columns and the sum**:

freq\_col\_s1=number\_col\_s1/length(col\_s1);

freq\_col\_s2=number\_col\_s2/length(col\_s2);

freq\_sum\_stars=number\_sum\_stars/length(sum\_stars);

## 4) Finding the relative frequency of the sum of the 5 winners

% I sum the 5 columns

sum\_winners=col1+col2+col3+col4+col5;

% First I eliminate the NaN elements

sum\_winners2=[]; % Store the non NaN elements in another array

for i=[1:length(sum\_winners)]

if ~(isnan(sum\_winners(i)))

sum\_winners2=[sum\_winners2;sum\_winners(i)];

end

end

sum\_winners=sum\_winners2; % Then updating the variable sum\_winners with the one without NaN elements

% Then I **calculate the number of times each sum appears** as the stars

number\_sum\_winners=[];

for j=min(sum\_winners):max(sum\_winners) % min(sum\_winners)=44, max(sum\_winners)=212

number\_sum\_winners=[number\_sum\_winners,numel(sum\_winners(sum\_winners==j))];

end

% And the **frequence**

freq\_sum\_winners=number\_sum\_winners/length(sum\_winners);

## 5) Draw a new random winner combination

The design for the algorithm is eliminating the elements that appear lower than the average of each column, so I will only have the numbers that are more repeated.

Then I will create an array with all these numbers for each column repeated as many times as the number\_col indicates, to take into account the frequency of each number.

Then I will pick randomly a number of the first column, then another one of the 2nd that is different from the first; then another one of the 3rd that is different from the ones before.

This 5 numbers will be the winner combination if their sum is below the average

% I **put as 0** in each column **the numbers that appear less times than the average**

% Calculate the **average** of times that a number appears in the first column

average1=sum(number\_col1,'all')/length(number\_col1);

% Set as 0 the times the number appears if they appear less than the average

number\_col1(number\_col1<average1)=0;

% I do the same for the rest of the columns

average2=sum(number\_col2,'all')/length(number\_col2);

number\_col2(number\_col2<average2)=0;

average3=sum(number\_col3,'all')/length(number\_col3);

number\_col3(number\_col3<average3)=0;

average4=sum(number\_col4,'all')/length(number\_col4);

number\_col4(number\_col4<average4)=0;

average5=sum(number\_col5,'all')/length(number\_col5);

number\_col5(number\_col5<average5)=0;

% I will call **correspondence, the matrix** with the corresponding **numbers** on

% the first column **and the number of times they appear** in the other one

% that is different from 0

numbers=[1:50];

correspondence1=[];

correspondence2=[];

correspondence3=[];

correspondence4=[];

correspondence5=[];

% I add to correspondence the numbers that have in the number of times is different from 0 (greater or equal to the average)

for j=numbers

if number\_col1(j)~=0

correspondence1=[correspondence1;j,number\_col1(j)];

end

if number\_col2(j)~=0

correspondence2=[correspondence2;j,number\_col2(j)];

end

if number\_col3(j)~=0

correspondence3=[correspondence3;j,number\_col3(j)];

end

if number\_col4(j)~=0

correspondence4=[correspondence4;j,number\_col4(j)];

end

if number\_col5(j)~=0

correspondence5=[correspondence5;j,number\_col5(j)];

end

end

% Create for each column **an array with each number repeated as the number\_col**

col1\_repeated=[];

for i=[1:length(correspondence1)]

for j=[1:correspondence1(i,2)]

col1\_repeated=[col1\_repeated; correspondence1(i,1)];

end

end

col2\_repeated=[];

for i=[1:length(correspondence2)]

for j=[1:correspondence2(i,2)]

col2\_repeated=[col2\_repeated; correspondence2(i,1)];

end

end

col3\_repeated=[];

for i=[1:length(correspondence3)]

for j=[1:correspondence3(i,2)]

col3\_repeated=[col3\_repeated; correspondence3(i,1)];

end

end

col4\_repeated=[];

for i=[1:length(correspondence4)]

for j=[1:correspondence4(i,2)]

col4\_repeated=[col4\_repeated; correspondence4(i,1)];

end

end

col5\_repeated=[];

for i=[1:length(correspondence5)]

for j=[1:correspondence5(i,2)]

col5\_repeated=[col5\_repeated; correspondence5(i,1)];

end

end

% Then, I **calculate a winner combination** by picking randomly a number from each column (all different) and their sum must be higher than or equal to the average:

while true

random=(length(col1\_repeated)-1).\*rand(1,1) + 1;

n1= col1\_repeated(round(random,0)); % Picked a random number from the 1st column

while true

random=(length(col2\_repeated)-1).\*rand(1,1) + 1;

n2= col2\_repeated(round(random,0)); % Picked a random number from the 2nd col

if n2~=n1 % A 2nd number is picked until is different from the 1st one

break

end

end

while true

random=(length(col3\_repeated)-1).\*rand(1,1) + 1;

n3=col3\_repeated(round(random,0)); % Picked a random number from the 3rd col

if n3~=n1 && n3~=n2 % A 3rd number is picked until is different from the previous ones

break

end

end

while true

random=(length(col4\_repeated)-1).\*rand(1,1) + 1;

n4=col4\_repeated(round(random,0)); % Picked a random number from the 4th col

if n4~=n1 && n4~=n2 && n4~=n3 % A 4th number is picked until is different from the previous ones

break

end

end

while true

random=(length(col5\_repeated)-1).\*rand(1,1) + 1;

n5=col5\_repeated(round(random,0)); % Picked a random number from the 5th col

if n5~=n1 && n5~=n2 && n5~=n3 && n5~=n4 % A 5th number is picked until is different from the previous ones

break

end

end

% Then, it’s check that their sum is greater or equal to the average of the sums

if n1+n2+n3+n4+n5>=sum(sum\_winners,'all')/length(sum\_winners)

break

end

end

% I print the combination

disp(""+n1+" "+n2+" "+n3+" "+n4+" "+n5)

## 6) Do the same with the stars

The design of the algorithm for the stars is the same as the one for the winners

% I **put as 0** in each col **the numbers that appear less times than the average**

% Calculate the **average** of times that a number appears in the first column

average\_s1=sum(number\_col\_s1,'all')/length(number\_col\_s1);

% Set as 0 the times the number appears if they appear less than the average

number\_col\_s1(number\_col\_s1<average\_s1)=0;

average\_s2=sum(number\_col\_s2,'all')/length(number\_col\_s2);

number\_col\_s2(number\_col\_s2<average\_s2)=0;

% I will call **correspondence, the matrix** with the corresponding **numbers** on

% the first column **and the number of times they appear** in the other one

% that is different from 0

numbers=[1:12];

correspondence\_s1=[];

correspondence\_s2=[];

% I add to correspondence the numbers that have in the number of times is different from 0 (greater or equal to the average)

for j=numbers

if number\_col\_s1(j)~=0

correspondence\_s1=[correspondence\_s1;j,number\_col\_s1(j)];

end

if number\_col\_s2(j)~=0

correspondence\_s2=[correspondence\_s2;j,number\_col\_s2(j)];

end

end

% Create for each column **an array with each number repeated as the col\_s**

col\_s1\_repeated=[];

for i=[1:length(correspondence\_s1)]

for j=[1:correspondence\_s1(i,2)]

col\_s1\_repeated=[col\_s1\_repeated; correspondence\_s1(i,1)];

end

end

col\_s2\_repeated=[];

for i=[1:length(correspondence\_s2)]

for j=[1:correspondence\_s2(i,2)]

col\_s2\_repeated=[col\_s2\_repeated; correspondence\_s2(i,1)];

end

end

% Then, I **calculate a star combination** by picking randomly a number from each column (all different) and their sum must be higher than or equal to the average:

while true

random=(length(col\_s1\_repeated)-1).\*rand(1,1) + 1;

s1= col\_s1\_repeated(round(random,0)); % Picked a random number from the 1st column

while true

random=(length(col\_s2\_repeated)-1).\*rand(1,1) + 1;

s2= col\_s2\_repeated(round(random,0)); % Picked a random number from the 2nd column

if s2~=s1 % A 2nd number is picked until is different from the 1st one

break

end

end

% Then, it’s check that their sum is greater or equal to the average of the sums

if s1+s2>=sum(sum\_stars,'all')/length(sum\_stars)

break

end

end

% I print the combination

disp(""+s1+" "+s2)

## 7) Plot on the screen 10 different winning combinations

I create 2 functions to do the same as 5) and 6)

function [n1,n2,n3,n4,n5] = calculate\_winners\_combination(col1\_repeated,col2\_repeated,col3\_repeated,col4\_repeated,col5\_repeated,sum\_winners)

% **Calculates a winning combination**

% -**col1\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col1

% -**col2\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col2

% -**col3\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col3

% -**col4\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col4

% -**col5\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col5

% -**sum\_winners**: It’s the sum of the 5 columns of the numbers

while true

random=(length(col1\_repeated)-1).\*rand(1,1) + 1;

n1= col1\_repeated(round(random,0));

while true

random=(length(col2\_repeated)-1).\*rand(1,1) + 1;

n2= col2\_repeated(round(random,0));

if n2~=n1

break

end

end

while true

random=(length(col3\_repeated)-1).\*rand(1,1) + 1;

n3=col3\_repeated(round(random,0));

if n3~=n1 && n3~=n2

break

end

end

while true

random=(length(col4\_repeated)-1).\*rand(1,1) + 1;

n4=col4\_repeated(round(random,0));

if n4~=n1 && n4~=n2 && n4~=n3

break

end

end

while true

random=(length(col5\_repeated)-1).\*rand(1,1) + 1;

n5=col5\_repeated(round(random,0));

if n5~=n1 && n5~=n2 && n5~=n3 && n5~=n4

break

end

end

if n1+n2+n3+n4+n5>=sum(sum\_winners,'all')/length(sum\_winners)

break

end

end

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function [s1,s2] = calculate\_stars\_combination(col\_s1\_repeated,col\_s2\_repeated,sum\_stars)

% **Calculates a winning combination for the stars**

% -**col\_s1\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col\_s1

% -**col\_s2\_repeated**: An array with each number (that is repeated more times than or equal to the average) repeated as the number\_col\_s2

% -**sum\_stars**: It’s the sum of the 2 columns of the stars

while true

random=(length(col\_s1\_repeated)-1).\*rand(1,1) + 1;

s1= col\_s1\_repeated(round(random,0));

while true

random=(length(col\_s2\_repeated)-1).\*rand(1,1) + 1;

s2= col\_s2\_repeated(round(random,0));

if s2~=s1

break

end

end

if s1+s2>=sum(sum\_stars,'all')/length(sum\_stars)

break

end

end

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now the code to print 10 number combinations:

for i=[1:10]

% Calculating a winner combination in each iteration

[n1,n2,n3,n4,n5]=calculate\_winners\_combination(col1\_repeated,col2\_repeated,col3\_repeated,col4\_repeated,col5\_repeated,sum\_winners);

% Calculating a star combination in each iteration

[s1,s2]=calculate\_stars\_combination(col\_s1\_repeated,col\_s2\_repeated,sum\_stars);

% Printing the final combination

disp(""+n1+" "+n2+" "+n3+" "+n4+" "+n5+" "+s1+" "+s2)

end