Laboratory of data visualization: Python project

Student: Marc Palomer

NIUB: 20118394

First of all, we are going to read all the information from different hospitals, which of course will be received in different format data sheets. We will need to read this formats, homogenize them into pandas' DataFrames, give them variable names and categorize the data depending on its procedence.

In [ ]:
```
#Import the needed libraries
import tabula as t
import pandas as p

# Read pdf, excel and csv into a DataFrame
LB = t.read_pdf("LongBeachData.pdf", pages='all',multiple_tables= False)
L = p.DataFrame(LB[0])
Clev = p.read_csv("processed.cleveland.data")
H = p.read_excel("Hungarian_Switzerland.xlsx",sheet_name = "Hungarian Data")
S = p.read_excel("Hungarian_Switzerland.xlsx",sheet_name = "Switzerland Data")


#We add a parient number to identify which patient we are talking about inside every hospital.
#After prior observation we can see Cleveland is the only dataset without this info.
num = []
for element in range (1,len(Clev)+1):
    num.append("Patient "+ str(element))
if Clev.columns[0] != "Patient num":
    Clev.insert(0,"Patient num",num)

#We change the names of the variables in order to homogenise them (prior checking all follow the same order in the datasets)
variables = ['Patient num','age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak',
          'slope','ca','thal','num']
Clev.columns = variables
H.columns = variables
S.columns = variables
L.columns = variables

#We insert the Hospital column at the last variable position with its categorical value
Clev['Hospital']='CLE'
H['Hospital']='Hun'
S['Hospital']='Switz'
L['Hospital']='Long'

#We reorganize the columns as we wish
Clev=Clev.reindex(columns=['Patient num','age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak',
          'slope','ca','thal','Hospital','num'])
L=L.reindex(columns=['Patient num','age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak',
          'slope','ca','thal','Hospital','num'])
S=S.reindex(columns=['Patient num','age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak',
          'slope','ca','thal','Hospital','num'])
H=H.reindex(columns=['Patient num','age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak',
          'slope','ca','thal','Hospital','num'])
```

Concatenating the datasets from the four hospitals

In [ ]:
```
#We concatenate the four Dataframes into a big one
C = p.concat([H, S, L, Clev], axis = 0, ignore_index=True)


print("\n if we don't take into consideration the 'Patient num' and the 'num', we have "+str(len(C.columns)-2)+" variables")
```

```
 if we don't take into consideration the 'Patient num' and the 'num', we have 14 variables
```

Handling 'incorrect' or 'incomplete' data

If hospital is not CLE the ca variable is registered esporadically. All elements of 'ca' not in CLE will be converted to NaN

In [ ]:
```
from numpy import *
for idx, ele in enumerate(C['Hospital']):
    if ele != 'CLE':
        C.loc[[idx],['ca']] = nan #We look up the dataframe as it it was a matrix but using loc() property
```

Cleaning missing values

Now we must change all the uninformative symbols for others we can handle, *? --> Nan*

In [ ]:
```
C = C.replace('?', nan)
m = input("If you wish to check if the replacement was correctly performed write Y")
if m == "Y":
    for element in C.columns:
        print("Variable: "+element)
        print(C[element].value_counts(dropna=False))
        print("\n")
#Here we can observe the amount of each observation we have divided by variables
#We can see the interrogants have been correctly replaced by NANs
```

Data type handling

Now we are going to check the different types of data we are working with:

In [ ]:
```
def Data_Type(array):
    data_type = []
    for element in array:
        if type(element) not in data_type:
            data_type.append(type(element))
    return data_type

for element in C:
    data=Data_Type(C[str(element)])
    print(element, "; ", data)
```

```
Patient num ;  [<class 'str'>]
age ;  [<class 'float'>]
sex ;  [<class 'float'>]
cp ;  [<class 'float'>]
trestbps ;  [<class 'int'>, <class 'float'>, <class 'str'>]
chol ;  [<class 'int'>, <class 'float'>, <class 'str'>]
fbs ;  [<class 'int'>, <class 'float'>, <class 'str'>]
restecg ;  [<class 'float'>]
thalach ;  [<class 'int'>, <class 'float'>, <class 'str'>]
exang ;  [<class 'int'>, <class 'float'>, <class 'str'>]
oldpeak ;  [<class 'float'>, <class 'str'>, <class 'int'>]
slope ;  [<class 'float'>, <class 'int'>, <class 'str'>]
ca ;  [<class 'float'>, <class 'str'>]
thal ;  [<class 'float'>, <class 'int'>, <class 'str'>]
Hospital ;  [<class 'str'>]
num ;  [<class 'int'>]
```

As seen above, we have some datatypes inside some of our variables that don't match with what it should be, it is necessary to homogenize the data types into the correct ones.

In [ ]:
```
#Function to_numeric transforms the fata into float in our case
for el in variables:
    C[el]=p.to_numeric(C[el], downcast = 'float', errors = 'ignore')

#We check if now the datatypes are correct. They are.
for element in C:
    data=Data_Type(C[str(element)])
    print(element, "; ", data)
```

```
Patient num ;  [<class 'str'>]
age ;  [<class 'float'>]
sex ;  [<class 'float'>]
cp ;  [<class 'float'>]
trestbps ;  [<class 'float'>]
chol ;  [<class 'float'>]
fbs ;  [<class 'float'>]
restecg ;  [<class 'float'>]
thalach ;  [<class 'float'>]
exang ;  [<class 'float'>]
oldpeak ;  [<class 'float'>]
slope ;  [<class 'float'>]
ca ;  [<class 'float'>]
thal ;  [<class 'float'>]
Hospital ;  [<class 'str'>]
num ;  [<class 'float'>]
```

Preliminary correlation visualization

In [ ]:
```
(C.corr()).style.background_gradient(cmap='coolwarm')
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
age	1.000000	0.056362	0.169530	0.243648	-0.086633	0.231853	0.211280	-0.366646	0.202802	0.257062	0.152616	0.365732	0.136450	0.340880
sex	0.056362	1.000000	0.172326	0.000666	-0.197499	0.088036	-0.016620	-0.179671	0.182011	0.103230	0.123866	0.095014	0.373528	0.259977
cp	0.169530	0.172326	1.000000	0.027071	-0.132187	0.046494	0.035859	-0.349075	0.418722	0.249027	0.213648	0.229952	0.318350	0.397031
trestbps	0.243648	0.000666	0.027071	1.000000	0.092641	0.158838	0.097695	-0.105315	0.153050	0.161074	0.061430	0.100775	0.106890	0.123051
chol	-0.086633	-0.197499	-0.132187	0.092641	1.000000	0.024479	0.116103	0.235992	-0.034279	0.047327	-0.060602	0.118462	-0.180299	-0.231357
fbs	0.231853	0.088036	0.046494	0.158838	0.024479	1.000000	0.128458	-0.054939	0.031773	0.051465	0.084172	0.152906	0.103808	0.158310
restecg	0.211280	-0.016620	0.035859	0.097695	0.116103	0.128458	1.000000	0.052253	0.034009	0.115988	-0.019782	0.131116	-0.041551	0.142026
thalach	-0.366646	-0.179671	-0.349075	-0.105315	0.235992	-0.054939	0.052253	1.000000	-0.356174	-0.152076	-0.363476	-0.264423	-0.327854	-0.365992
exang	0.202802	0.182011	0.418722	0.153050	-0.034279	0.031773	0.034009	-0.356174	1.000000	0.394245	0.323877	0.144129	0.341977	0.387921
oldpeak	0.257062	0.103230	0.249027	0.161074	0.047327	0.051465	0.115988	-0.152076	0.394245	1.000000	0.419669	0.299260	0.253504	0.445021
slope	0.152616	0.123866	0.213648	0.061430	-0.060602	0.084172	-0.019782	-0.363476	0.323877	0.419669	1.000000	0.116697	0.289363	0.309374
ca	0.365732	0.095014	0.229952	0.100775	0.118462	0.152906	0.131116	-0.264423	0.144129	0.299260	0.116697	1.000000	0.258398	0.518018
thal	0.136450	0.373528	0.318350	0.106890	-0.180299	0.103808	-0.041551	-0.327854	0.341977	0.253504	0.289363	0.258398	1.000000	0.441990
num	0.340880	0.259977	0.397031	0.123051	-0.231357	0.158310	0.142026	-0.365992	0.387921	0.445021	0.309374	0.518018	0.441990	1.000000

Filtering patients by different factors

Use factors to filter patients (rows): Hospital, sex, age. Prior preliminary checking of correlations has been done, but as it occupies a lot of visual space it won't be shown explicitly.

In [ ]:
```
#Filtering by age

above_65=C[C['age']>=65]
corr_above_65=above_65.corr()
corr_above_65.style.background_gradient(cmap='coolwarm')
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
age	1.000000	0.033325	-0.074838	-0.011196	-0.048274	0.026610	0.000537	-0.126190	0.089905	0.109849	0.104141	0.084174	-0.134184	0.100739
sex	0.033325	1.000000	0.221611	-0.033602	-0.210002	0.160525	-0.051217	-0.233665	0.338841	0.193855	0.261174	0.161065	0.442694	0.394314
cp	-0.074838	0.221611	1.000000	-0.133561	-0.110416	-0.062330	-0.076772	-0.211765	0.344061	0.118175	0.164758	0.141578	0.475168	0.339720
trestbps	-0.011196	-0.033602	-0.133561	1.000000	-0.016634	0.256778	-0.118766	0.063094	0.033354	0.095522	0.129808	-0.020129	0.148726	0.054447
chol	-0.048274	-0.210002	-0.110416	-0.016634	1.000000	0.059616	0.234163	0.278992	0.098509	0.062056	-0.014021	0.032136	-0.313221	-0.087734
fbs	0.026610	0.160525	-0.062330	0.256778	0.059616	1.000000	-0.045962	0.040322	-0.002564	-0.054152	0.165247	0.047607	0.189778	0.072373
restecg	0.000537	-0.051217	-0.076772	-0.118766	0.234163	-0.045962	1.000000	0.096373	-0.133937	-0.103108	-0.097562	0.042981	-0.221748	-0.068336
thalach	-0.126190	-0.233665	-0.211765	0.063094	0.278992	0.040322	0.096373	1.000000	-0.269099	-0.017796	-0.300428	0.032491	-0.175721	-0.162900
exang	0.089905	0.338841	0.344061	0.033354	0.098509	-0.002564	-0.133937	-0.269099	1.000000	0.152470	0.302040	0.276778	0.262371	0.446374
oldpeak	0.109849	0.193855	0.118175	0.095522	0.062056	-0.054152	-0.103108	-0.017796	0.152470	1.000000	0.525121	-0.021557	0.268021	0.353450
slope	0.104141	0.261174	0.164758	0.129808	-0.014021	0.165247	-0.097562	-0.300428	0.302040	0.525121	1.000000	-0.038112	0.327551	0.395138
ca	0.084174	0.161065	0.141578	-0.020129	0.032136	0.047607	0.042981	0.032491	0.276778	-0.021557	-0.038112	1.000000	0.101654	0.443672
thal	-0.134184	0.442694	0.475168	0.148726	-0.313221	0.189778	-0.221748	-0.175721	0.262371	0.268021	0.327551	0.101654	1.000000	0.479300
num	0.100739	0.394314	0.339720	0.054447	-0.087734	0.072373	-0.068336	-0.162900	0.446374	0.353450	0.395138	0.443672	0.479300	1.000000

As seen above, while in general population "ca" and "thal" was the most correlated factor with the output, in people higher than 65 "thal" and "exang" are the most correlated ones.

In [ ]:
```
C_CLE=C[C['Hospital'] == "CLE"]
C_S=C[C['Hospital'] == "Switz"]
C_L=C[C['Hospital'] == "Long"]
C_H=C[C['Hospital'] == "Hun"]

#We observe the maximum and minimum ages for further segmentation
maxi = (C['age'].max())
mini = (C['age'].min())
print('The ages are between '+str(mini)+' and '+str(maxi))
```

```
The ages are between 28.0 and 77.0
```

As shown below, filtering by sex we get that for males the most correlated factors with the output are "ca" and "oldpeak", while for female are "ca"and "thal"

In [ ]:
```
#Filtering by sex
n = 0
while n == 0:
    gender = input('Insert 0 for male and 1 for female correlation table')
    if gender == str(1) or gender == str(0):
        n = n+1

corr = C[C['sex']== int(gender)]
(corr.corr()).style.background_gradient(cmap='coolwarm')
```

```
/opt/anaconda3/envs/my_env/lib/python3.7/site-packages/pandas/io/formats/style.py:2813: RuntimeWarning: All-NaN slice encountered
  smin = np.nanmin(gmap) if vmin is None else vmin
/opt/anaconda3/envs/my_env/lib/python3.7/site-packages/pandas/io/formats/style.py:2814: RuntimeWarning: All-NaN slice encountered
  smax = np.nanmax(gmap) if vmax is None else vmax
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
age	1.000000	nan	0.155178	0.317277	0.134045	0.158147	0.181860	-0.302742	0.138253	0.237970	0.067830	0.388285	0.099994	0.253078
sex	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
cp	0.155178	nan	1.000000	0.097650	-0.023830	0.043408	0.168247	-0.217160	0.400194	0.313222	0.270557	0.141782	0.237156	0.400117
trestbps	0.317277	nan	0.097650	1.000000	0.079986	0.103380	0.063181	-0.095607	0.277986	0.239406	0.172446	0.276674	0.295658	0.355919
chol	0.134045	nan	-0.023830	0.079986	1.000000	0.168783	0.274548	0.018629	0.077453	0.067027	-0.108465	0.094507	0.203160	-0.080600
fbs	0.158147	nan	0.043408	0.103380	0.168783	1.000000	0.179897	-0.101728	0.116099	0.080346	-0.069890	0.266756	0.150283	0.223980
restecg	0.181860	nan	0.168247	0.063181	0.274548	0.179897	1.000000	0.068474	0.027939	0.194209	0.046251	0.098204	0.044779	0.175387
thalach	-0.302742	nan	-0.217160	-0.095607	0.018629	-0.101728	0.068474	1.000000	-0.223801	-0.151526	-0.323480	-0.147917	-0.339408	-0.166787
exang	0.138253	nan	0.400194	0.277986	0.077453	0.116099	0.027939	-0.223801	1.000000	0.284109	0.315257	0.027597	0.382908	0.332382
oldpeak	0.237970	nan	0.313222	0.239406	0.067027	0.080346	0.194209	-0.151526	0.284109	1.000000	0.494234	0.464330	0.383266	0.559184
slope	0.067830	nan	0.270557	0.172446	-0.108465	-0.069890	0.046251	-0.323480	0.315257	0.494234	1.000000	0.198823	0.385770	0.392011
ca	0.388285	nan	0.141782	0.276674	0.094507	0.266756	0.098204	-0.147917	0.027597	0.464330	0.198823	1.000000	0.381585	0.644481
thal	0.099994	nan	0.237156	0.295658	0.203160	0.150283	0.044779	-0.339408	0.382908	0.383266	0.385770	0.381585	1.000000	0.565168
num	0.253078	nan	0.400117	0.355919	-0.080600	0.223980	0.175387	-0.166787	0.332382	0.559184	0.392011	0.644481	0.565168	1.000000

Correlation table by hospital (Thalach vs output or num)

Now we must generate a correlation table between thalach and num, divided for the different hospitals and in different age groups. This could be performed for an arbitrary pair of variables, but the ones with higher apparent correlation coeffitient in the global correlation chart would be logically used

In [ ]:
```
import numpy as np
#Fem els grups d'edat en grups de 8 anys perque hi hagi més homogeneitat de mostra a cada classe
labs=np.linspace(1,30,10)
frames = [C_S,C_L,C_H,C_CLE]
for element in frames:
    bins=p.cut(x=element['age'], bins=np.linspace(26,80,11), labels=labs)
    element.insert(len(element.columns),'Bins',bins)
```

In [ ]:
```
#Aquí conseguirem els factors de correlació entre 'thalch' i 'num' per cada un dels bins d'edat ascendentment
correlations = p.DataFrame()
hospitals = ['Switzerland','Long Beach','Hungary','Cleveland']
for j,k in zip(frames,hospitals):
    factors =[]
    for i in labs:
        C_num = j[j['Bins']==i]
        m =C_num.filter(['thalach','num'])
        corr_factor = m.corr()
        factor =corr_factor['num']
        factors.append(factor[0])
    correlations.insert(0,column = k, value = factors)
correlations =abs(correlations)
correlations
```

Out[ ]:

	Cleveland	Hungary	Long Beach	Switzerland
0	NaN	0.440086	NaN	NaN
1	0.878049	0.543539	NaN	0.197666
2	0.352447	0.112779	0.089616	0.137606
3	0.623660	0.234090	0.131306	0.177363
4	0.370659	0.351050	0.373180	0.250819
5	0.443044	0.325104	0.020371	0.390251
6	0.338190	0.154918	0.258306	0.439467
7	0.255657	NaN	0.045236	0.320750
8	0.425026	NaN	0.319759	0.766760
9	1.000000	NaN	0.587137	NaN

Ara hem de fer una taula de correlació entre thalach i num dividida entre hospitals i grups d'edat, igual que surt al pwp. Això ho podem fer amb aquelles dues variables o amb les que vulguessim, veient les que fan millor pinta a la taula de correlacions global.

Generate a line plot of the correlation DF

- Include title, axis labels, annotations
- Generate a scatter plot of ‘age’ vs. ‘thalach’ colored by ‘num’
- Generate a bar plot of ‘age’

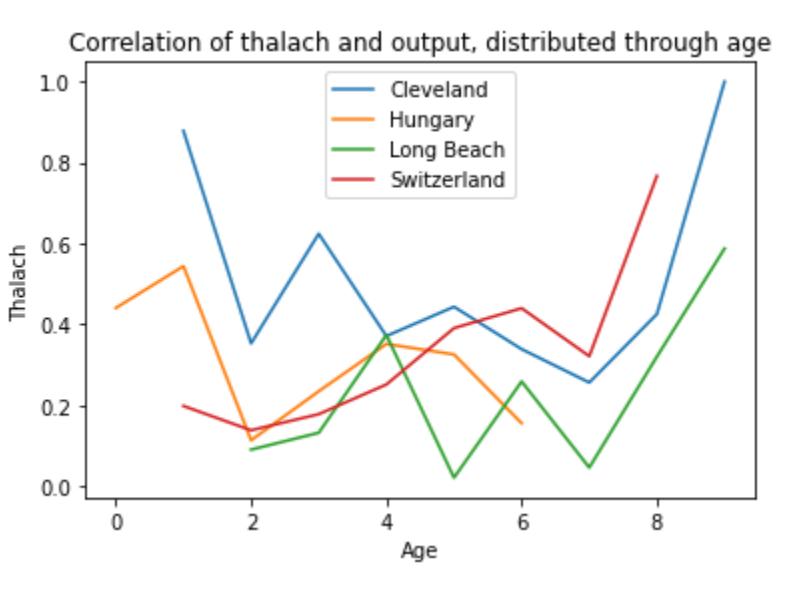
Line Plot

In [ ]:
```
import matplotlib.pyplot as plt

#Line Plot
ax =correlations.plot.line(title = 'Correlation of thalach and output, distributed through age' )
ax.set_xlabel("Age")
ax.set_ylabel("Thalach")
```

Out[ ]:

```
Text(0, 0.5, 'Thalach')
```

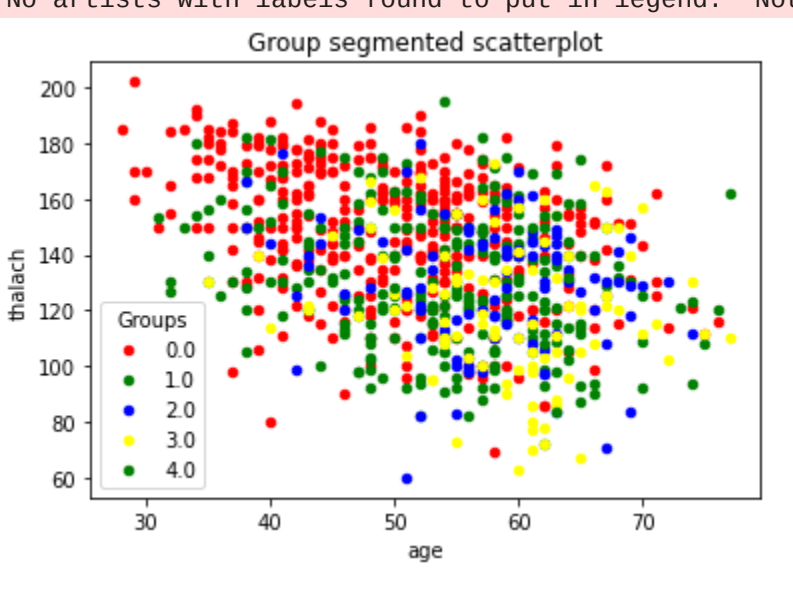


Scatter plot

In this scatter plot we can observe the different aglomerations of thalach values for the different ages and segmented by groups

In [ ]:
```
#Scatter plot
groups = C.groupby('num')
colors = {0:'red', 1:'green', 2:'blue', 3:'yellow', 4: 'green'}
fig, ax = plt.subplots()
ax.margins(0.05) # Optional, just adds 5% padding to the autoscaling
ax.legend(title ='Groups')
for name, group in groups:
    group.plot(ax=ax, kind = 'scatter',x = 'age',y ='thalach',label=str(name), color = colors[name], title ='Group segmented scatterplot')
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
```



Barplot

We will need the number of people in each age group and the age representing each bin in order to construct a bar plot of the age distribution

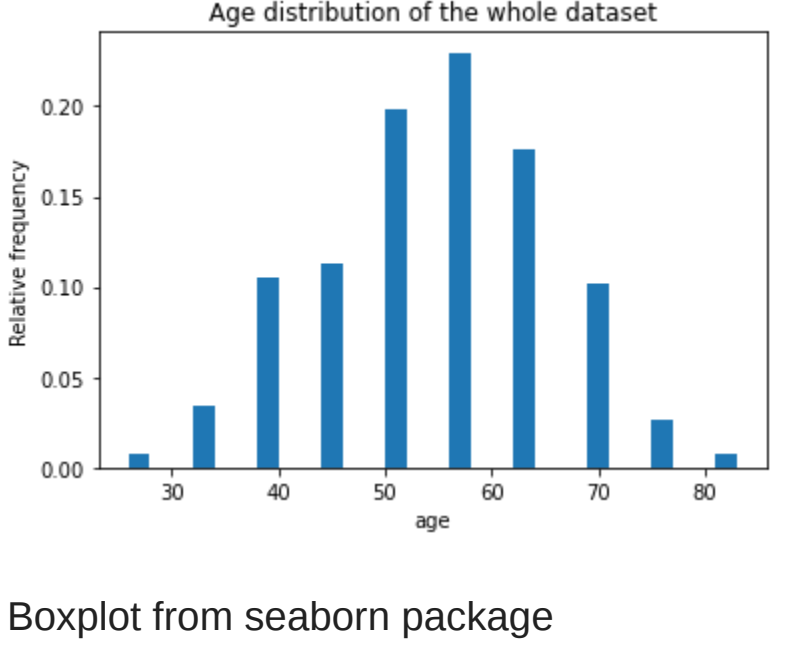
In [ ]:
```
#labs or labels is the age groups

bins = p.cut(x=C['age'], bins=np.linspace(26,80,11), labels=labs)
C.insert(len(C.columns),'Bins',bins)
```

In [ ]:
```
x = np.round(labs*1.89+25,0)
gr=C.groupby('Bins')
frequency = []
for name, group in gr:
    frequency.append(group.shape[0])
frequency[:] = [x / C.shape[0] for x in frequency]
width = 2
plt.bar(x,frequency, width)
plt.xlabel('age')
plt.ylabel("Relative frequency")
plt.title("Age distribution of the whole dataset")
```

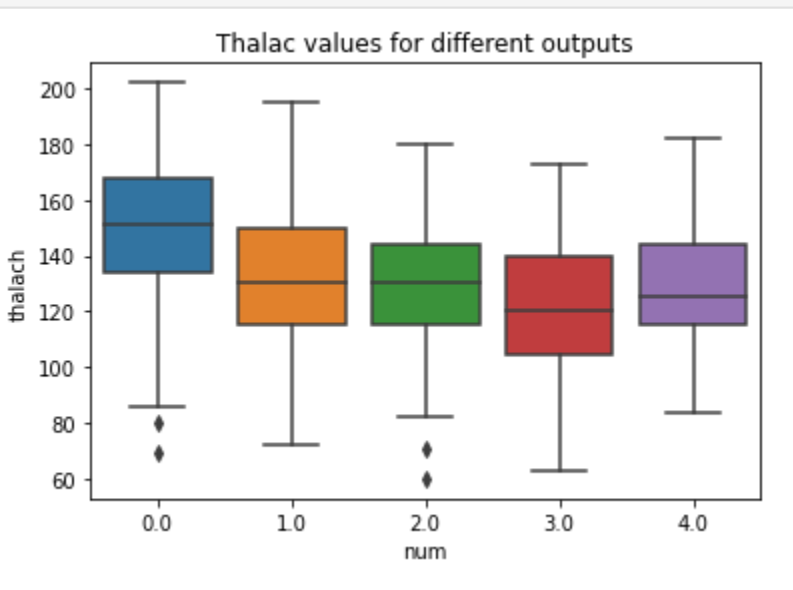
Out[ ]:

```
Text(0.5, 1.0, 'Age distribution of the whole dataset')
```



Boxplot from seaborn package

In [ ]:
```
#Seaborn boxplot
import seaborn as sns
ax = sns.boxplot(x="num", y='thalach', data=C).set_title("Thalac values for different outputs")
```



In the graph above we have related the values of the patients' thalach parameter with its cardiac output "diagnostic".