Esse aqui é o trabalho da Cadeira de Introdução a Modelagem. Nesse trabalho foi necessário resolver 2 questões no âmbito de Regressão Linear. Primeiro será importado as bibliotecas do Python para realizar os cálculos.

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
%load_ext autoreload
%autoreload 2
```

Agora criaremos a função do cálculo de Mínimos Quadrados. Essa função retorna os coeficientes, o Coeficiente de Determinação e o Coeficiente de Determinação Ajustado.

In [ ]:

```
def MMQ(x,y):

    try:
        x = np.vstack((np.ones(x.shape[1]),x)).T
    except:
        x = np.vstack((np.ones(x.shape[0]),x)).T

    b = np.dot(np.linalg.inv(np.dot(x.T,x)),np.dot(x.T,y))

    #print('Os coeficientes da sua regressão são: ', b)

    y_ = np.dot(b,x.T)
    r2 = 1 - (np.dot(y - y_,y - y_)/np.dot(y - np.mean(y),y - np.mean(y)))

    #print('O Coeficiente de Determinação foi calculado como:', r2)

    r2A = 1 - ((x.shape[0] - 1)*np.dot(y - y_,y - y_))/((x.shape[0] - x.shape[1] -

    #print('O Coeficiente de Determinação Ajustado foi calculado como:', r2A)
    return b,r2,r2A
```

Questão 4

In exercise physiology, an objective measure of aerobic fitness is the oxygen consumption in volume per unit body weight per unit time by individual. To determine if it is feasible to predict this fitness measure, an experiment was conducted in which 31 individuals were tested. The following factors were studied:

- X1: age in years
- X2: weight in kilograms
- X3: time to run 1 ½ miles
- X4: resting pulse rate
- X5: pulse rate at the end of run
- X6: maximum pulse rate during run
- Y: oxygen consumption in millilitres (ml) per kilogram (kg) body weight per minute

Será carregado os dados da questão:

In [ ]:

```
r = np.loadtxt('Q4.txt',dtype=str)
r = np.array([[i.replace(',','.')for i in j] for j in r]).astype(float)
r = r[:,0:]
Y = r[:,1]
X = r[:,2:]
```

Seguem aqui os gráficos de cada variável em função do Consumo de Oxigênio. Percebe-se que o Tempo para percorrer uma 1 milha e a Pulsação em repouso são as melhores variáveis.

In [ ]:

```
label = [
    'Idade (Anos)',
    'Peso (kg)',
    'Tempo para percorrer 1 milha',
    'Pulsação em repouso',
    'Pulsação em repouso no final',
    'Máxima Pulsação',
]
for i in range(X.shape[1]):
    x = X[:,i]
    plt.figure(figsize=(8,8))
    plt.scatter(x,Y)
    plt.xlabel(label[i])
    plt.ylabel("Consumo de Oxigênio (ml/kgmin)")
    plt.title("Consumo de Oxigênio (ml/kgmin) em função de " +label[i])
    plt.show()
```

Agora quais serão as melhores variáveis representam a variável y? Para isso criaremos a seguinte função `best_MMQ` . Da coleção de variáveis x_i escolhemos o x_i com o melhor r^2, que chamaremos de x'_1. Sabendo que essa é a melhor variável vamos agora escolher a melhor variável x_i, baseado no r^2, para entrar no modelo, ou seja calcular x_i|x'_1. Se todos os r^2 forem menores que o r^2 com apenas x'_1 é retornado o modelo apenas com x'_1, caso contrário será escolhido a melhor variável desse grupo, x'_2, e será adicionado ao modelo e assim sucessivamente. Ao final, retornará quais as melhores variáveis para o modelo e será printado também o histórico do r^2 e quanto de adesão de uma nova variável foi signifcante no r^2.

In [ ]:

```
def best_MMQ(Y,X):
    vetor = []
    historico = []
    r_historico = []
    while(True):
        r_max = 0
        max_ = 0
        for i in range(X.shape[1]):
            if(len(vetor)==0):
                x = X[:,i]
            else:
                v_ = np.copy(vetor)
                v_ = list(v_)
                if(i in vetor):
                    continue
                else:
                    v_.append(i)
                x = X[:,v_]
            _,r2,__ = MMQ(x.T,Y)
            if(r2 > r_max):
                r_max = r2
                max_ = i

        if((len(vetor)==0) or (r_max > r_historico[len(vetor)-1])):
            vetor.append(max_)
            historico.append(vetor)
            r_historico.append(r_max)
            if(len(r_historico)>1):
                print(r_historico[-1],"{:.1f}".format((r_historico[-1] - r_historic
        else:
            break
    return historico[-1],r_historico[-1]
```

In [ ]:

```
variaveis,r2 = best_MMQ(Y,X)
print(variaveis)
```

Podemos perceber que a adesão da última variável foi pouco significativa no modelo e poderíamos até retirá-la. Como foi previsto as variáveis 2 foram as que tiveram maior significância no modelo.

In [ ]:

```
coeficientes,r2,r2A = MMQ(X[:,variaveis[:-1]].T,Y)
print('Os coeficientes da sua regressão são: ', coeficientes)
print('O Coeficiente de Determinação foi calculado como:', r2)
print('O Coeficiente de Determinação Ajustado foi calculado como:', r2A)
```

Questão 5

In an effort to develop a preliminary personnel equation for estimation of worker-hours per month expended in surgical services at Naval hospitals, the U.S. Navy collected data on y (worker-hours per month) and x (surgical cases) from 15 hospitals. The data (taken from the Navy's Procedures and Analyses for Staffing Standards

Development: Data/Regression Analysis Handbook) are shown in the Table 12 below.

Fit the following models to these data.

- $y = \beta o + \beta 1 X + \varepsilon$
- $ln(y) = \beta o + \beta 1 X^{-1} + \varepsilon$
- $y = \beta o + \beta 1 X + \beta 2 X^2 + \varepsilon$
- Comment on the adequacy of each of these models

Vamos carregar os dados:

In [ ]:

```
y = np.array([1275,
1350,
1650,
2000,
3750,
4222,
5018,
6125,
6200,
8150,
9975,
12200,
12750,
13014,
13275])
x = np.array([
    230,
235,
250,
277,
522,
545,
625,
713,
735,
820,
992,
1322,
1900,
2022,
2155
])
```

Primeiro vamos analisar graficamente como se comporta a variável preditora em função dos dados. Tanto no caso liner quanto no caso log.

In [ ]:

```
plt.figure(figsize=(8,8))
plt.ylabel("Horas-Trabalho por Mês.")
plt.xlabel("Número de Casos Cirúrgicos.")
plt.plot(x,y)
plt.show()
```

In [ ]:

```
plt.figure(figsize=(8,8))
plt.ylabel("Logaritmo das Horas-Trabalho por Mês.")
plt.xlabel("Logaritmo dos Números de Casos Cirúrgicos.")
plt.plot(np.log(x),np.log(y))
plt.show()
```

Percebemos que o gráfico logy vs logx apresenta a melhor configuração de uma reta. Agora iremos aplicar os Mínimos Quadrados r ver como fica.

In [ ]:

```
coeficientes,r2,r2A = MMQ(x,y)
print('Os coeficientes da sua regressão são: ', coeficientes)
print('O Coeficiente de Determinação foi calculado como:', r2)
print('O Coeficiente de Determinação Ajustado foi calculado como:', r2A)
plt.figure(figsize=(8,8))
plt.ylabel("Horas-Trabalho por Mês.")
plt.xlabel("Número de Casos Cirúrgicos.")
plt.scatter(x,y,label = 'dados')
plt.plot(x,np.dot(np.vstack((np.ones(x.shape[0]),x)).T,coeficientes),label = 'Regre
plt.legend()
plt.show()
```

In [ ]:

```
coeficientes,r2,r2A = MMQ(np.log(x),np.log(y))
print('Os coeficientes da sua regressão são: ', coeficientes)
print('O Coeficiente de Determinação foi calculado como:', r2)
print('O Coeficiente de Determinação Ajustado foi calculado como:', r2A)
plt.figure(figsize=(8,8))
plt.ylabel("Horas-Trabalho por Mês.")
plt.xlabel("Número de Casos Cirúrgicos.")
plt.scatter(np.log(x),np.log(y),label = 'dados')
plt.plot(np.log(x),np.dot(np.vstack((np.ones(np.log(x).shape[0]),np.log(x))).T,coef
plt.legend()
plt.show()
```

In [ ]:

```
x_ = np.vstack((x,x**2))
coeficientes,r2,r2A = MMQ(x_,y)
print('Os coeficientes da sua regressão são: ', coeficientes)
print('O Coeficiente de Determinação foi calculado como:', r2)
print('O Coeficiente de Determinação Ajustado foi calculado como:', r2A)
plt.figure(figsize=(8,8))
plt.ylabel("Horas-Trabalho por Mês.")
plt.xlabel("Número de Casos Cirúrgicos.")
plt.scatter(x,y,label = 'dados')
plt.plot(x,np.dot( np.vstack((np.ones(x_.shape[1]),x_)).T ,coeficientes),label = 'R
plt.legend()
plt.show()
```

Portanto, não só visualmente mas vendo os valores de R^2 e do R^2 Ajustado, vemos que o melhr fit é usando o x^2.