Metódos Quantitativos

O objetivo desse trabalho é examinar a estrutura e configuração dos dados, além de aprender sobre o relacionamento entre as variáveis da base de dados.

A $Análise\ Exploratória\ de\ Dados$ inclui um conjunto de ferramentas descritivas e gráficas para a buscar padrões e tendências que desempenharam o papel de hipóteses para uma análise completa.

Empregando técnicas $estatísticas\ descritivas\ e\ gráficas$ para estudar o conjunto de dados, detectando $outliers\ e\ anomalias$, e $comunicando\ de\ forma\ eficaz\ os\ resultados\ do\ modelo.$

In [226]:
```
import pandas as pd
import requests
import statsmodels.api as sm
import urllib
from urllib import request, response, error, parse
from urllib.request import urlopen
from bs4 import BeautifulSoup, element
```

1. Extraíndo os Dados.

In [227]:
```
req = requests.get('https://projects.fivethirtyeight.com/global-club-soccer-ranking
s/')
if req.status_code == 200:
    print ('Requisição bem sucedida!')
```

```
Requisição bem sucedida!
```

In [228]:
```
url = "https://projects.fivethirtyeight.com/global-club-soccer-rankings/"
html = urlopen(url)
soup = BeautifulSoup(html,"lxml")
title = soup.title
titleText = title.get_text()
print(titleText)
```

```
Global Club Soccer Rankings | FiveThirtyEight
```

In [229]:
```
table = soup.find('table', {'class':'all-teams'})
```

In [230]:
```
table_str = str(table)
```

In [231]:

```
df = pd.read_html(table_str)[0]
df
```

Out[231]:

	Unnamed: 0_level_0	Unnamed: 1_level_0	Unnamed: 2_level_0	Unnamed: 3_level_0	Unnamed: 4_level_0	Team rating		
	Rank	1-week change	team	League	League country	off.	def.	spi
0	1	NaN	Man. City	Premier League	England	3.2	0.2	94.8
1	2	NaN	Bayern Munich	Bundesliga	Germany	3.4	0.4	93.6
2	3	NaN	Liverpool	Premier League	England	2.9	0.3	91.9
3	4	1.0	Barcelona	La Liga	Spain	2.8	0.4	89.6
4	5	-1.0	PSG	Ligue 1	France	2.9	0.5	89.2
5	6	NaN	Real Madrid	La Liga	Spain	2.9	0.5	89.1
6	7	1.0	Atlético Madrid	La Liga	Spain	2.2	0.3	86.5
7	8	-1.0	Chelsea	Premier League	England	2.6	0.5	86.2
8	9	NaN	Juventus	Serie A	Italy	2.6	0.5	85.9
9	10	NaN	RB Leipzig	Bundesliga	Germany	2.6	0.6	84.0
10	11	NaN	Leicester	Premier League	England	2.3	0.5	83.7
11	12	NaN	Ajax	Eredivisie	Netherlands	3.1	0.9	83.7
12	13	NaN	Tottenham	Premier League	England	2.5	0.7	82.9
13	14	NaN	RB Salzburg	Bundesliga	Austria	3.0	1.0	81.5
14	15	3.0	Leverkusen	Bundesliga	Germany	2.3	0.6	81.0
15	16	NaN	Dortmund	Bundesliga	Germany	2.4	0.7	80.5
16	17	2.0	Sevilla	La Liga	Spain	2.1	0.5	80.2
17	18	-3.0	Man. United	Premier League	England	2.1	0.6	79.8
18	19	1.0	Napoli	Serie A	Italy	2.2	0.7	79.4
19	20	-3.0	Inter Milan	Serie A	Italy	2.4	0.7	79.4
20	21	6.0	Getafe	La Liga	Spain	1.8	0.4	77.7
21	22	1.0	Porto	Primeira Liga	Portugal	2.0	0.6	77.4
22	23	-2.0	Everton	Premier League	England	2.0	0.6	77.0
23	24	-2.0	Zenit	Premier League	Russia	2.0	0.6	77.0
24	25	5.0	Roma	Serie A	Italy	2.2	0.7	76.8
25	26	-2.0	Real Sociedad	La Liga	Spain	2.1	0.7	76.7
26	27	-2.0	Benfica	Primeira Liga	Portugal	2.2	0.8	75.8
27	28	5.0	Atalanta	Serie A	Italy	2.3	0.9	75.8
28	29	-3.0	Arsenal	Premier League	England	2.3	0.9	75.7
29	30	-1.0	Athletic Bilbao	La Liga	Spain	1.8	0.5	75.4
...	...	...	...	...	...	...	...	...
599	600	NaN	Mansfield Town	League Two	England	0.5	2.1	13.5
600	601	NaN	Falkenbergs	Allsvenskan	Sweden	0.6	2.2	13.4
601	602	NaN	Colchester	League Two	England	0.4	2.0	13.2
602	603	NaN	Las Vegas Lights	USL Championship	USA	0.6	2.4	12.1
603	604	NaN	Oldham Athletic	League Two	England	0.5	2.2	11.7
604	605	NaN	Rochdale	League One	England	0.5	2.2	11.5
605	606	NaN	Salford City	League Two	England	0.5	2.2	11.4

	Unnamed: 0_level_0	Unnamed: 1_level_0	Unnamed: 2_level_0	Unnamed: 3_level_0	Unnamed: 4_level_0	Team rating		
	Rank	1-week change	team	League	League country	off.	def.	spi
606	607	NaN	Hartford Athletic	USL Championship	USA	0.8	2.9	11.4
607	608	NaN	Stellenbosch	Premier Division	South Africa	0.3	2.0	10.6
608	609	NaN	Bolton	League One	England	0.5	2.4	10.3
609	610	NaN	S.P. Rangers	USL Championship	USA	0.7	2.9	10.0
610	611	NaN	Birmingham	USL Championship	USA	0.4	2.3	9.9
611	612	NaN	Bethlehem Steel	USL Championship	USA	0.7	3.0	9.6
612	613	NaN	Atlanta United 2	USL Championship	USA	0.6	2.9	9.1
613	614	NaN	Tacoma Defiance	USL Championship	USA	0.6	2.8	9.1
614	615	NaN	Newport County	League Two	England	0.2	1.9	9.0
615	616	NaN	Memphis 901	USL Championship	USA	0.3	2.2	8.9
616	617	NaN	Southend United	League One	England	0.5	2.7	8.9
617	618	NaN	Tulsa	USL Championship	USA	0.5	2.6	8.6
618	619	NaN	Leyton Orient	League Two	England	0.4	2.5	8.5
619	620	NaN	Cambridge	League Two	England	0.2	2.1	8.0
620	621	NaN	Stevenage	League Two	England	0.2	2.1	7.8
621	622	1.0	Port Vale	League Two	England	0.2	2.2	7.6
622	623	1.0	Crawley Town	League Two	England	0.3	2.5	7.4
623	624	1.0	Carlisle United	League Two	England	0.3	2.4	7.3
624	625	-3.0	Grimsby Town	League Two	England	0.2	2.2	7.3
625	626	NaN	Macclesfield	League Two	England	0.2	2.2	7.0
626	627	NaN	Walsall	League Two	England	0.2	2.2	6.7
627	628	NaN	C.S. Switchbacks	USL Championship	USA	0.2	2.4	5.6
628	629	NaN	Morecambe	League Two	England	0.2	2.5	5.3

629 rows × 8 columns

Utilizamos a biblioteca $BeautifulSoup$ para análise dos dados HTML e XML extraídos do website. Porém, para fazermos isso, é necessário a utilização da ferramenta $urllib$ que faz a conexão com a pagína web. Pela $urllib$ também é feita a verificação da conexão com a pagína web.

Criamos o objeto $BeautifulSoup$, onde, 'lxml' é o analisador de html. Usando a ferramenta $Inspecionar$ do navegador procuramos pela tag 'table' e sua classe 'all-teams', e usando o $BeautifulSoup$ extraimos a tabela.

2. Outro formato para ler os dados.

Quando o banco de dados é disponibilizado, podemos fazer o download dos dados e ler o arquivo csv diretamente.

In [485]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as scs
import statsmodels.api as sm
import statistics as stats
import math
```

In [233]:
```
Dados = pd.read_csv('spiglobalrankings.csv')
Dados.head(20)
```

Out[233]:

	rank	prev_rank	name	league	off	def	spi
0	1	1	Manchester City	Barclays Premier League	3.24	0.25	94.74
1	2	3	Bayern Munich	German Bundesliga	3.46	0.41	93.98
2	3	2	Liverpool	Barclays Premier League	2.90	0.29	92.43
3	4	4	Paris Saint-Germain	French Ligue 1	2.90	0.50	89.22
4	5	6	Barcelona	Spanish Primera Division	2.83	0.47	89.17
5	6	5	Real Madrid	Spanish Primera Division	2.87	0.49	89.14
6	7	9	Atletico Madrid	Spanish Primera Division	2.25	0.31	86.90
7	8	7	Chelsea	Barclays Premier League	2.62	0.51	86.72
8	9	8	Juventus	Italy Serie A	2.51	0.49	85.99
9	10	10	RB Leipzig	German Bundesliga	2.55	0.59	84.48
10	11	13	Leicester City	Barclays Premier League	2.23	0.45	83.73
11	12	12	Ajax	Dutch Eredivisie	3.02	0.90	83.71
12	13	14	Tottenham Hotspur	Barclays Premier League	2.50	0.64	82.95
13	14	11	FC Salzburg	Austrian T-Mobile Bundesliga	3.04	1.00	82.39
14	15	15	Borussia Dortmund	German Bundesliga	2.45	0.73	80.41
15	16	20	Internazionale	Italy Serie A	2.38	0.70	80.31
16	17	17	Bayer Leverkusen	German Bundesliga	2.29	0.64	80.31
17	18	18	Sevilla FC	Spanish Primera Division	2.12	0.54	80.29
18	19	16	Manchester United	Barclays Premier League	2.13	0.55	80.16
19	20	19	Napoli	Italy Serie A	2.25	0.62	80.09

In [234]:
```
Dados.isnull().sum()
```

Out[234]:
```
rank         0
prev_rank    0
name         0
league       0
off          0
def          0
spi          0
dtype: int64
```

Podemos observar que obtemos uma tabela sem a coluna dos países, porém nessa segunda tabela os dados já estão tratados pois não observamos dados faltantes NaN, confirmado pelo método isnull(). Logo iremos utliza-lá.

3. Tratamento de dados:

Para que seja melhor a interação com os dados, é necessário o tratamento e a limpeza adequada.

In [235]:
```
dados = Dados.rename(columns={"rank": "Rank", "prev_rank": "Variação Semanal", "name": "Time", "league": "Liga", "off": "Rating Ofensivo", "def": "Rating Defensivo", "spi": "Rating de Força Futebolística"})
```

In [236]:
```
dados.head(10)
```

Out[236]:

	Rank	Variação Semanal	Time	Liga	Rating Ofensivo	Rating Defensivo	Rating de Força Futebolística
0	1	1	Manchester City	Barclays Premier League	3.24	0.25	94.74
1	2	3	Bayern Munich	German Bundesliga	3.46	0.41	93.98
2	3	2	Liverpool	Barclays Premier League	2.90	0.29	92.43
3	4	4	Paris Saint-Germain	French Ligue 1	2.90	0.50	89.22
4	5	6	Barcelona	Spanish Primera Division	2.83	0.47	89.17
5	6	5	Real Madrid	Spanish Primera Division	2.87	0.49	89.14
6	7	9	Atletico Madrid	Spanish Primera Division	2.25	0.31	86.90
7	8	7	Chelsea	Barclays Premier League	2.62	0.51	86.72
8	9	8	Juventus	Italy Serie A	2.51	0.49	85.99
9	10	10	RB Leipzig	German Bundesliga	2.55	0.59	84.48

In [237]:
```
print(dados.shape)
```

```
(629, 7)
```

In [238]:
```
dados.describe()
```

Out[238]:

	Rank	Variação Semanal	Rating Ofensivo	Rating Defensivo	Rating de Força Futebolística
count	629.00000	629.00000	629.000000	629.000000	629.000000
mean	315.00000	315.00000	1.246789	1.452019	41.551574
std	181.72094	181.72094	0.502927	0.460630	18.636075
min	1.00000	1.00000	0.200000	0.250000	5.300000
25%	158.00000	158.00000	0.920000	1.120000	28.170000
50%	315.00000	315.00000	1.190000	1.450000	39.950000
75%	472.00000	472.00000	1.540000	1.760000	54.030000
max	629.00000	629.00000	3.460000	2.980000	94.740000

In [239]:
```
dados.dtypes
```

Out[239]:
```
Rank                               int64
Variação Semanal                   int64
Time                              object
Liga                              object
Rating Ofensivo                  float64
Rating Defensivo                 float64
Rating de Força Futebolística    float64
dtype: object
```

4. Descrevendo os Dados estatísticamente:

Após ler a base de dados, vamos descrever algumas técnicas estatísticas.

In [240]:
```
qualitativa = ['Time', 'Liga']
discreta = ['Rank', 'Variação Semanal']
continua = dados.columns[4:]
```

Tabela de Frequência para Variáveis Discretas:

In [241]:

```
tab_freq_liga = dados.Liga.value_counts(sort=True)
tab_freq_liga = pd.DataFrame({"Freq.": tab_freq_liga})

tab_freq_liga['Acum.'] = tab_freq_liga['Freq.'].cumsum()
tab_freq_liga['Rel. (%)'] = ((tab_freq_liga['Freq.']/629)*100)
tab_freq_liga['Rel. Acum. (%)'] = tab_freq_liga['Rel. (%)'].cumsum()

tab_freq_liga
```

Out[241]:

	Freq.	Acum.	Rel. (%)	Rel. Acum. (%)
United Soccer League	36	36	5.723370	5.723370
English League Championship	24	60	3.815580	9.538951
English League Two	24	84	3.815580	13.354531
Argentina Primera Division	24	108	3.815580	17.170111
Major League Soccer	24	132	3.815580	20.985692
English League One	23	155	3.656598	24.642289
Spanish Segunda Division	22	177	3.497615	28.139905
Italy Serie A	20	197	3.179650	31.319555
Barclays Premier League	20	217	3.179650	34.499205
Spanish Primera Division	20	237	3.179650	37.678855
French Ligue 1	20	257	3.179650	40.858506
Italy Serie B	20	277	3.179650	44.038156
Brasileiro Série A	20	297	3.179650	47.217806
French Ligue 2	20	317	3.179650	50.397456
Mexican Primera Division Torneo Apertura	19	336	3.020668	53.418124
German 2. Bundesliga	18	354	2.861685	56.279809
German Bundesliga	18	372	2.861685	59.141494
Turkish Turkcell Super Lig	18	390	2.861685	62.003180
Dutch Eredivisie	18	408	2.861685	64.864865
Japanese J League	18	426	2.861685	67.726550
Portuguese Liga	18	444	2.861685	70.588235
Chinese Super League	16	460	2.543720	73.131955
Norwegian Tippeligaen	16	476	2.543720	75.675676
Russian Premier Liga	16	492	2.543720	78.219396
Swedish Allsvenskan	16	508	2.543720	80.763116
South African ABSA Premier League	16	524	2.543720	83.306836
Belgian Jupiler League	16	540	2.543720	85.850556
Greek Super League	14	554	2.225755	88.076312
Danish SAS-Ligaen	14	568	2.225755	90.302067
Austrian T-Mobile Bundesliga	12	580	1.907790	92.209857
Scottish Premiership	12	592	1.907790	94.117647
UEFA Europa League	11	603	1.748808	95.866455
Australian A-League	11	614	1.748808	97.615262
Swiss Raiffeisen Super League	10	624	1.589825	99.205087
UEFA Champions League	4	628	0.635930	99.841017
Mexican Primera Division Torneo Clausura	1	629	0.158983	100.000000

In [242]:
```
tab_freq_liga.dtypes
```

Out[242]:
```
Freq.                int64
Acum.                int64
Rel. (%)           float64
Rel. Acum. (%)     float64
dtype: object
```

Tabela de Frequência para Variáveis Contínuas:

In [243]:
```
spi = dados['Rating de Força Futebolística']
bins_range = range(0, int(round(spi.max()))+10,10)
tab_freq_spi = pd.cut(spi, bins=bins_range, include_lowest=True, right=False)
tab_freq_spi = tab_freq_spi.value_counts(sort=False)

tab_freq_spi = pd.DataFrame({"Freq.": tab_freq_spi})
tab_freq_spi['Acum.'] = tab_freq_spi['Freq.'].cumsum()
tab_freq_spi['Rel. (%)'] = ((tab_freq_spi['Freq.']/tab_freq_spi['Freq.'].sum())*100)
tab_freq_spi['Rel. Acum. (%)'] = tab_freq_spi['Rel. (%)'].cumsum()

tab_freq_spi
```

Out[243]:

	Freq.	Acum.	Rel. (%)	Rel. Acum. (%)
[0, 10)	20	20	3.179650	3.179650
[10, 20)	58	78	9.220986	12.400636
[20, 30)	105	183	16.693164	29.093800
[30, 40)	133	316	21.144674	50.238474
[40, 50)	115	431	18.282989	68.521463
[50, 60)	81	512	12.877583	81.399046
[60, 70)	68	580	10.810811	92.209857
[70, 80)	29	609	4.610493	96.820350
[80, 90)	17	626	2.702703	99.523052
[90, 100)	3	629	0.476948	100.000000

In [244]:
```
tab_freq_spi.dtypes
```

Out[244]:
```
Freq.                int64
Acum.                int64
Rel. (%)           float64
Rel. Acum. (%)     float64
dtype: object
```

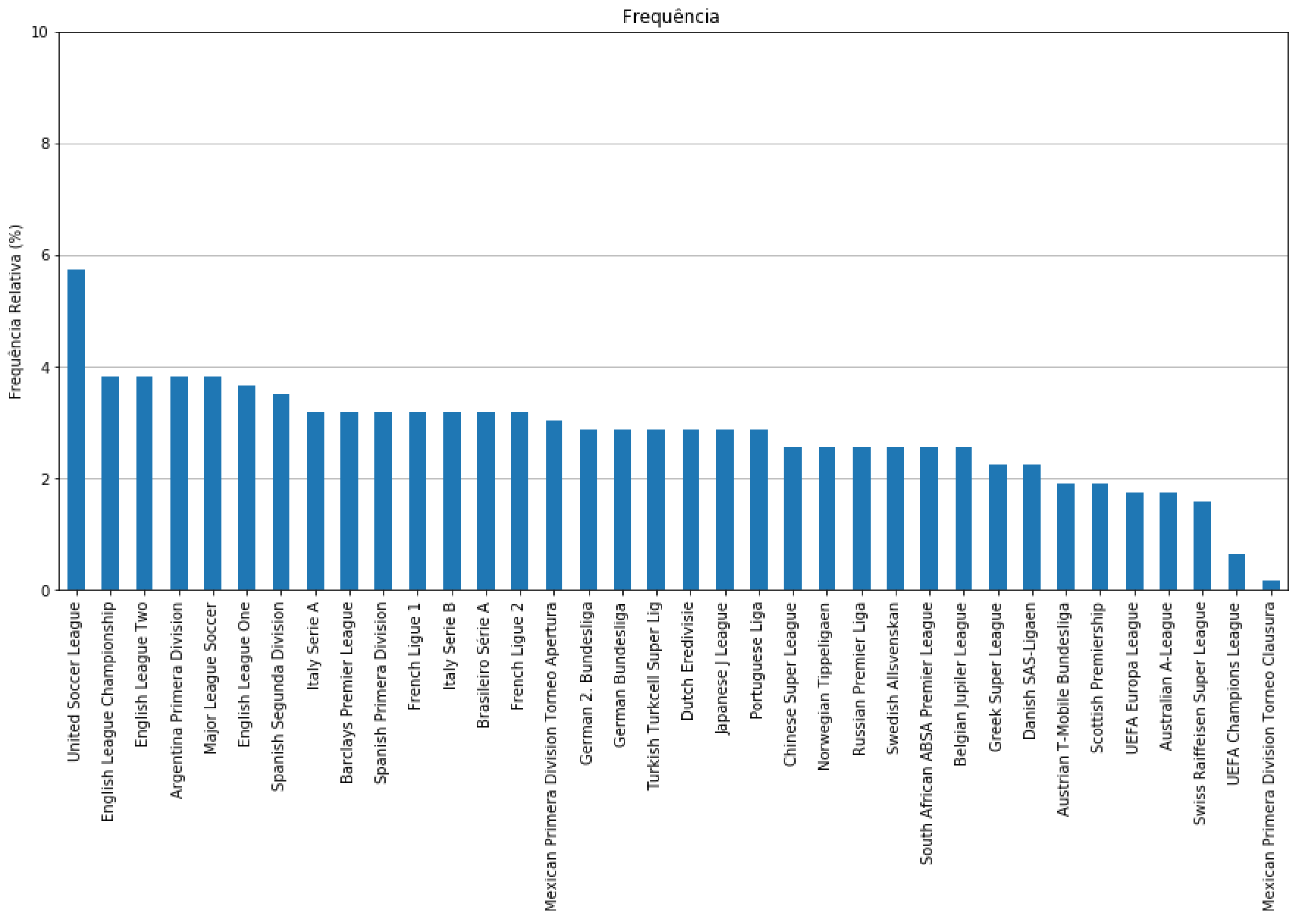
In [245]:
```
dados[continua].describe()
```
Out[245]:

	Rating Ofensivo	Rating Defensivo	Rating de Força Futebolística
count	629.000000	629.000000	629.000000
mean	1.246789	1.452019	41.551574
std	0.502927	0.460630	18.636075
min	0.200000	0.250000	5.300000
25%	0.920000	1.120000	28.170000
50%	1.190000	1.450000	39.950000
75%	1.540000	1.760000	54.030000
max	3.460000	2.980000	94.740000

5. Visualizando as informações estátisticas

Gráfico Massa de Probabilidade para Variáveis Discretas:

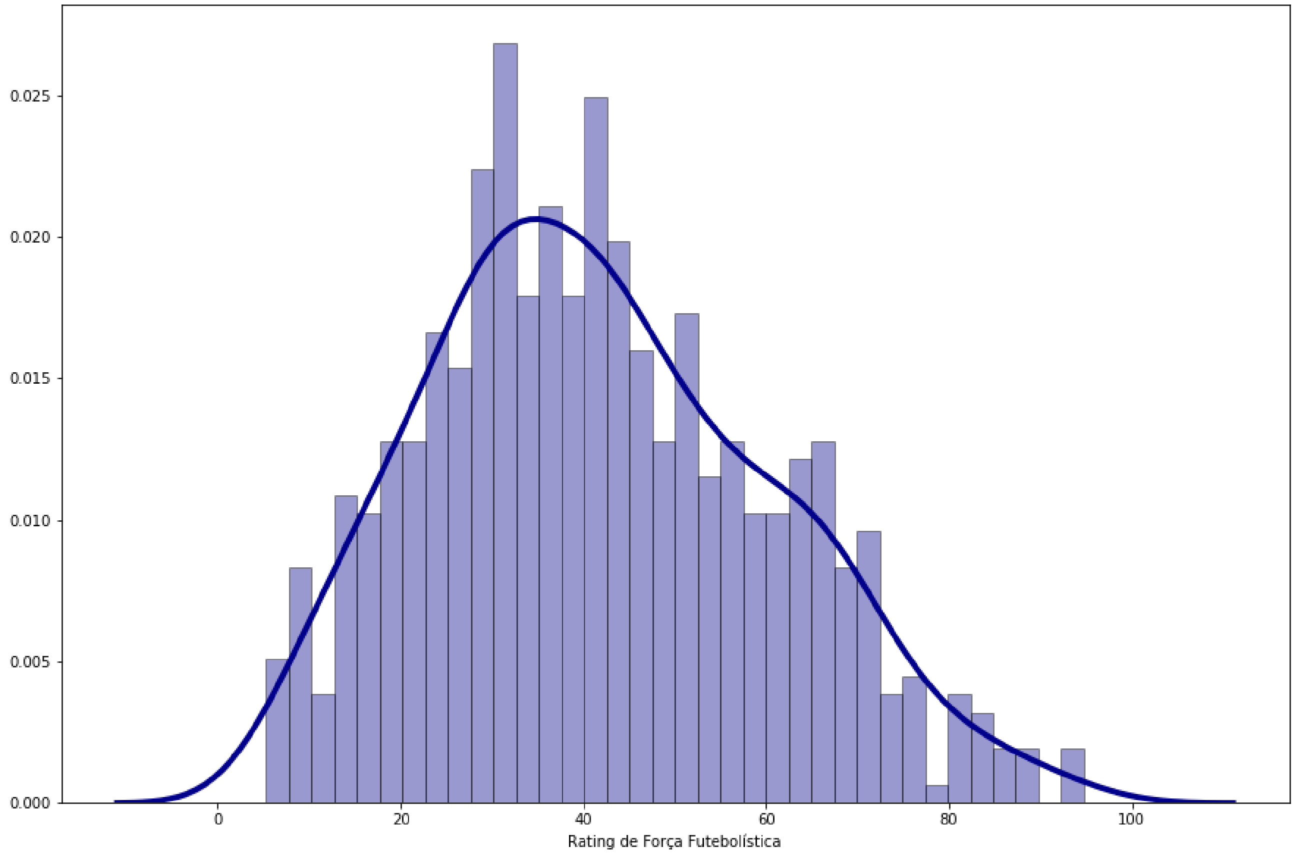
In [246]:
```
tab_freq_liga['Rel. (%)'].plot(kind='bar',zorder=2)
plt.grid(axis='y', zorder=1)
plt.rcParams['figure.figsize'] = (15,10)
plt.xticks(fontsize = 10, rotation=90)
plt.ylim(0,10)
plt.ylabel(r'Frequência Relativa (%)')
plt.title('Frequência ')

plt.plot()
plt.show()
```



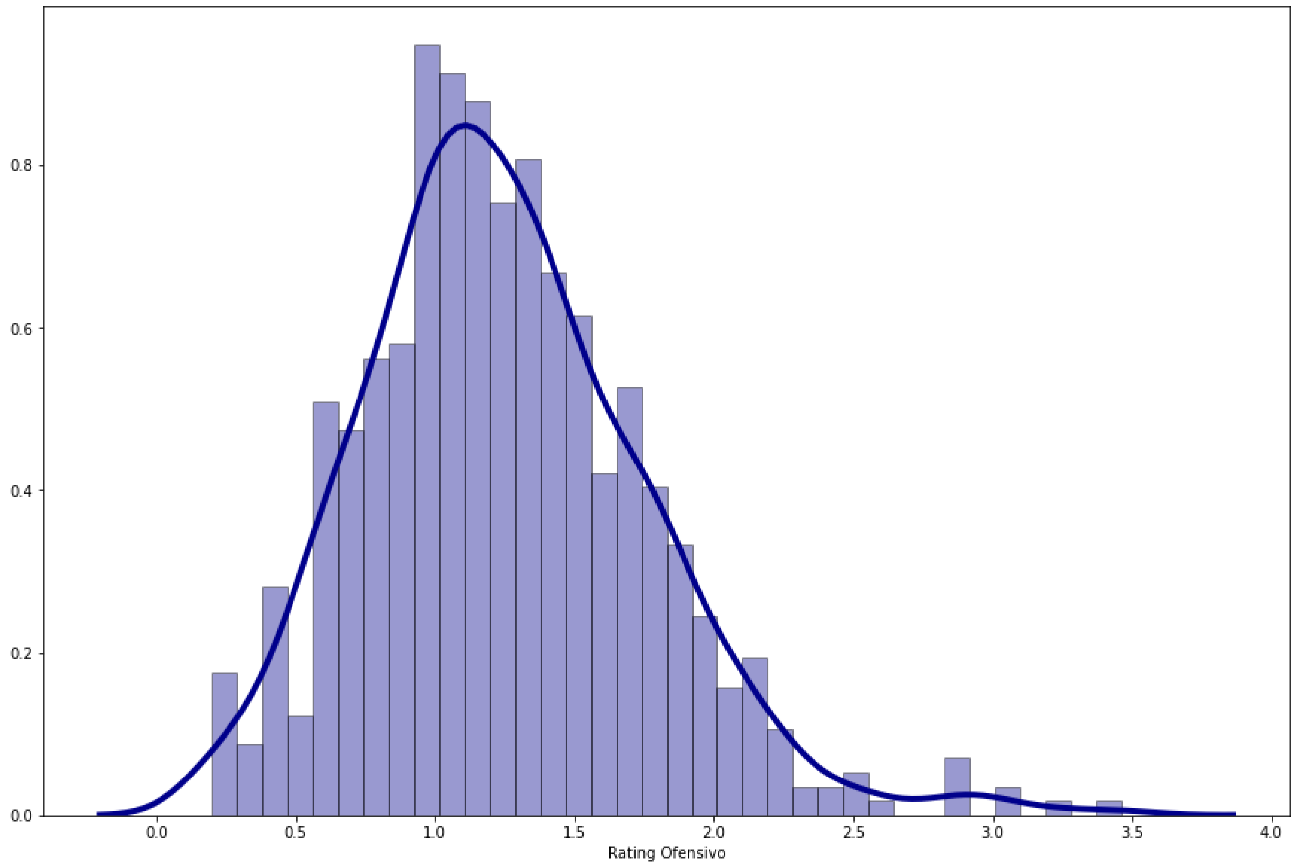
Esse gráfico nos dá a frequência relativa (probabilidade) de algum clube da lista participar de alguma dessas ligas.

Gráficos Densidade de Kernel para Variáveis Contínuas:

In [247]:
```
sns.distplot(dados['Rating de Força Futebolística'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
plt.plot()
plt.show()
```



In [248]:
```
sns.distplot(dados['Rating Ofensivo'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
plt.plot()
plt.show()
```



In [249]:
```
sns.distplot(dados['Rating Defensivo'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
plt.plot()
plt.show()
```

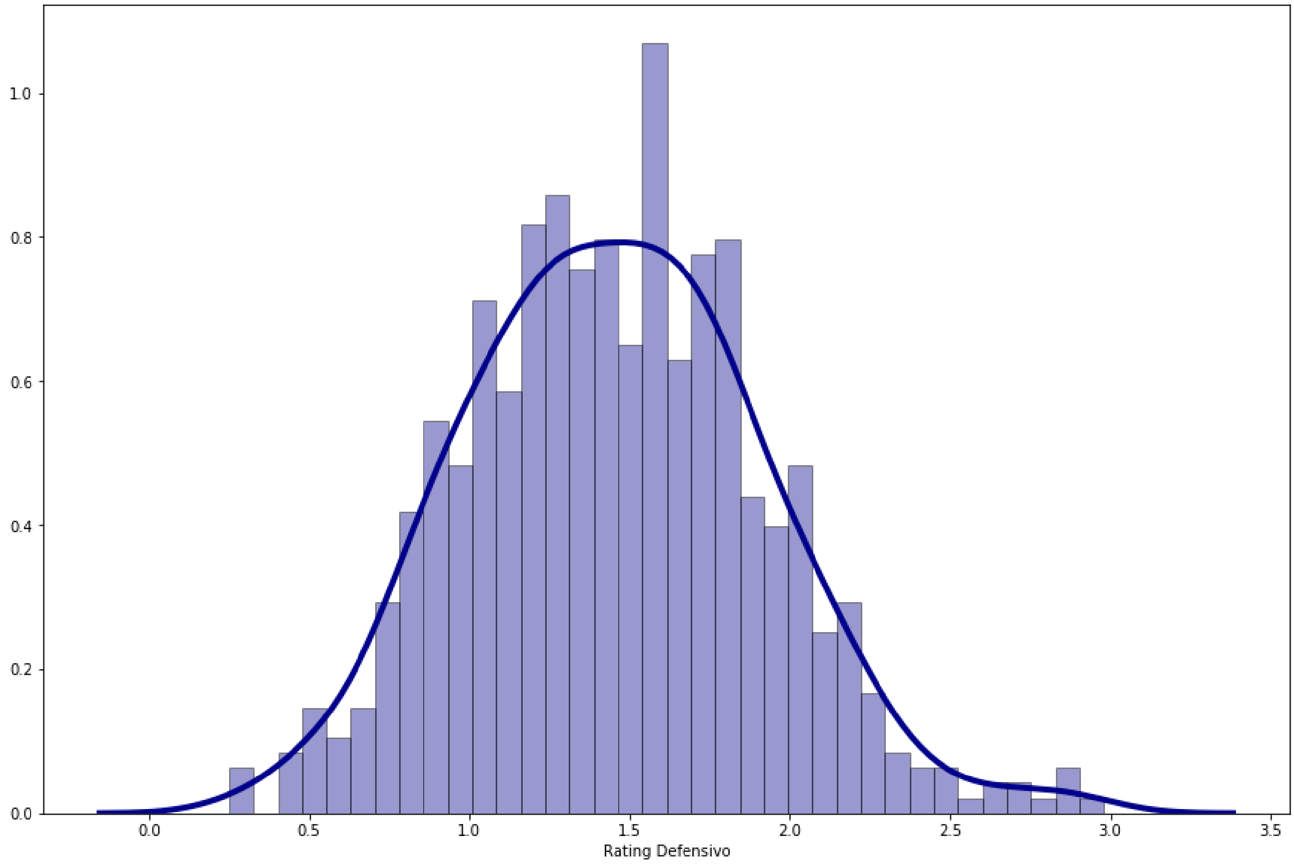
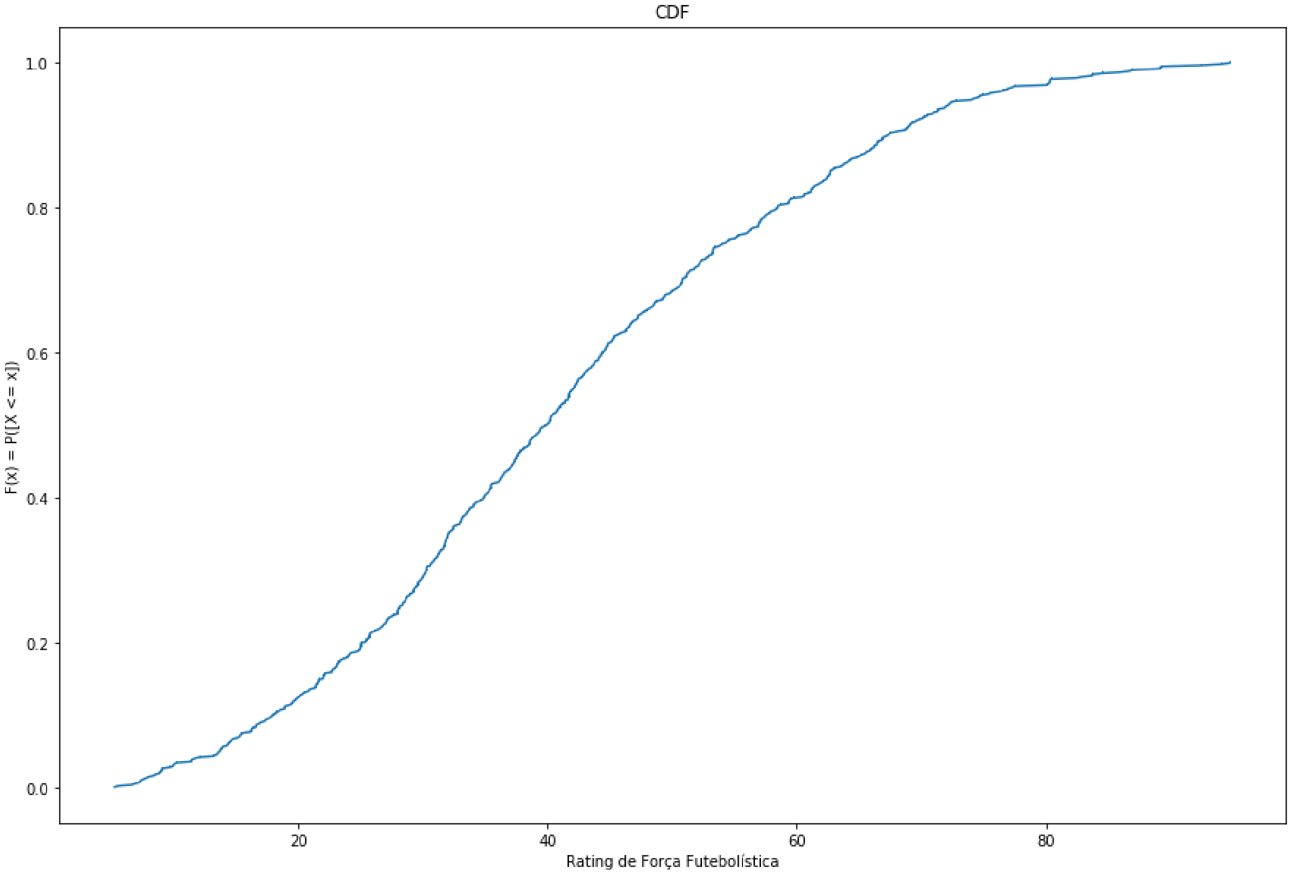


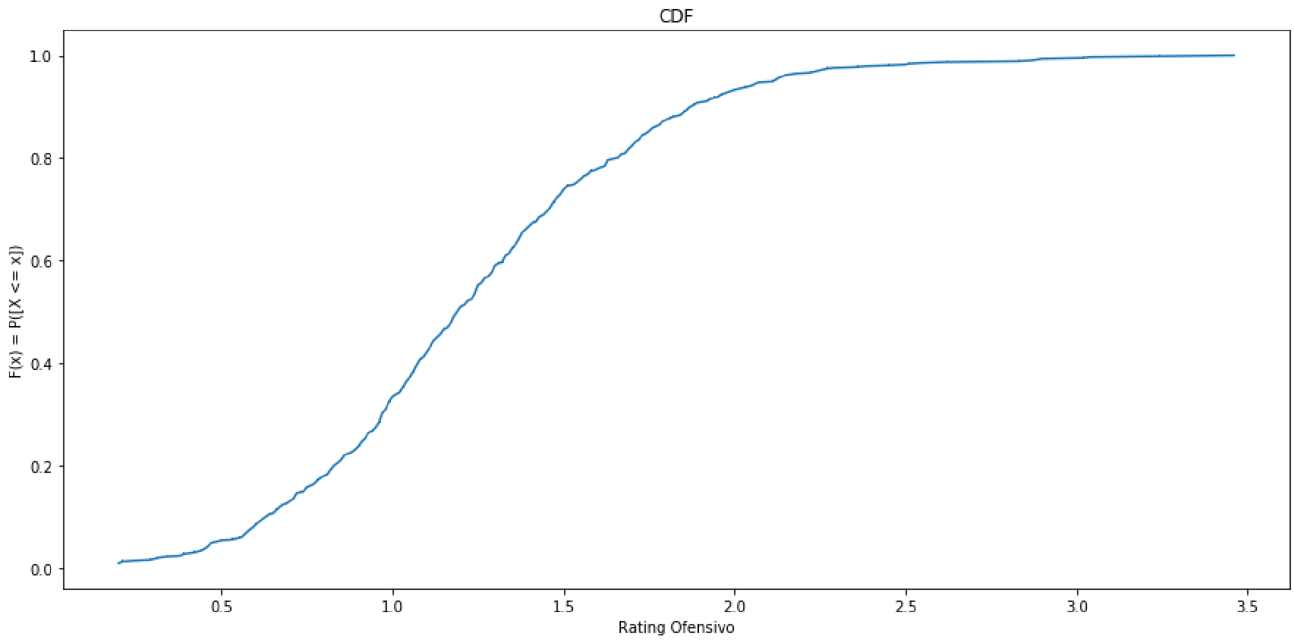
Gráfico de densidade pela estimativa da densidade de Kernel de uma versão contínua e suave de um histograma estimado a partir dos dados. Nesse método, uma curva contínua (o kernel) é desenhada em todos os pontos de dados individuais e todas essas curvas são adicionadas juntas para fazer uma única estimativa de densidade suave. O kernel mais usado é um gaussiano (que produz uma curva de sino gaussiano em cada ponto de dados). O eixo x é o valor da variável como em um histograma. O eixo y é a função de densidade de probabilidade para a estimativa da densidade do kernel.

Gráficos de Distribuição Acumulativa para Variáveis Contínuas:

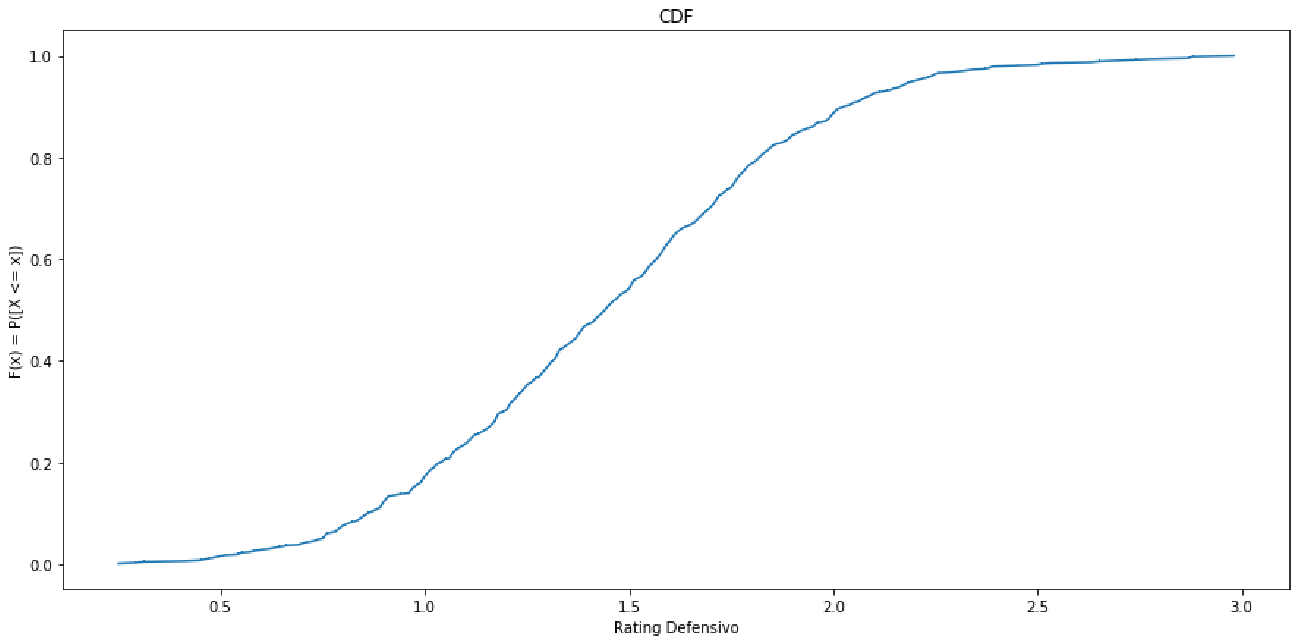
In [250]:
```
x = np.sort(dados['Rating de Força Futebolística'])
func = sm.distributions.empirical_distribution.ECDF(x)
y = func(x)
plt.title('CDF')
plt.xlabel('Rating de Força Futebolística')
plt.ylabel('F(x) = P([X <= x])')
plt.plot(x, y, marker = ',')
plt.rcParams['figure.figsize'] = (15,7)
plt.show()
```



In [251]:
```
x = np.sort(dados['Rating Ofensivo'])
func = sm.distributions.empirical_distribution.ECDF(x)
y = func(x)
plt.title('CDF')
plt.xlabel('Rating Ofensivo')
plt.ylabel('F(x) = P([X <= x])')
plt.plot(x, y, marker = ',')
plt.rcParams['figure.figsize'] = (15,7)
plt.show()
```

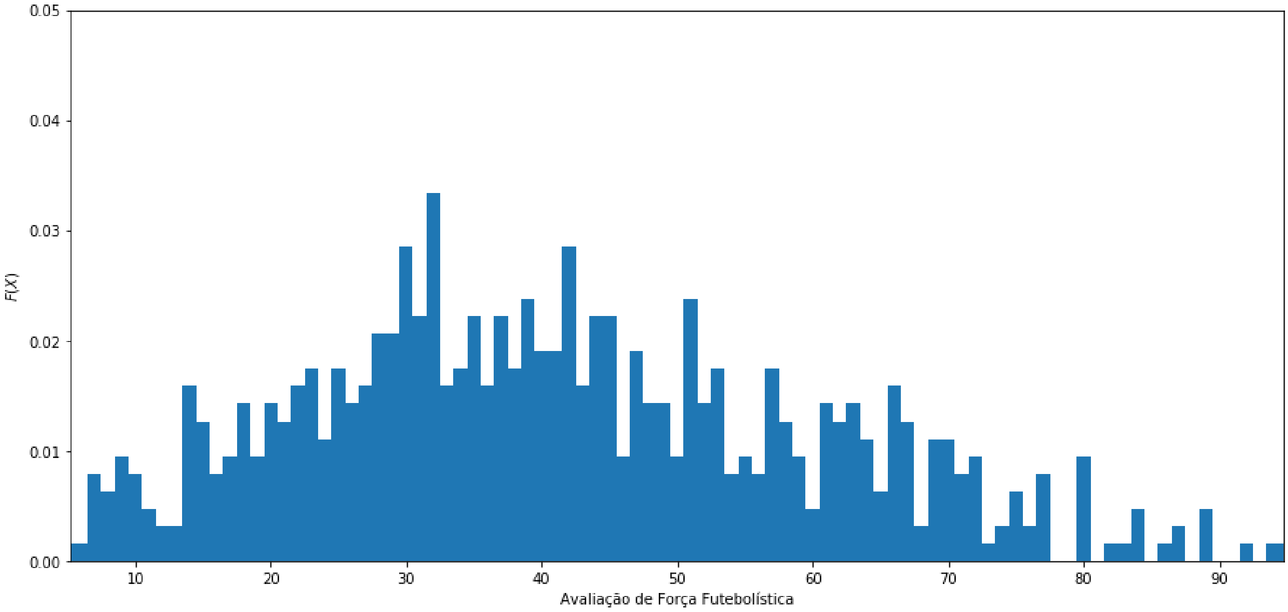


In [252]:
```
x = np.sort(dados['Rating Defensivo'])
func = sm.distributions.empirical_distribution.ECDF(x)
y = func(x)
plt.title('CDF')
plt.xlabel('Rating Defensivo')
plt.ylabel('F(x) = P([X <= x])')
plt.plot(x, y, marker = ',')
plt.rcParams['figure.figsize'] = (15,7)
plt.show()
```

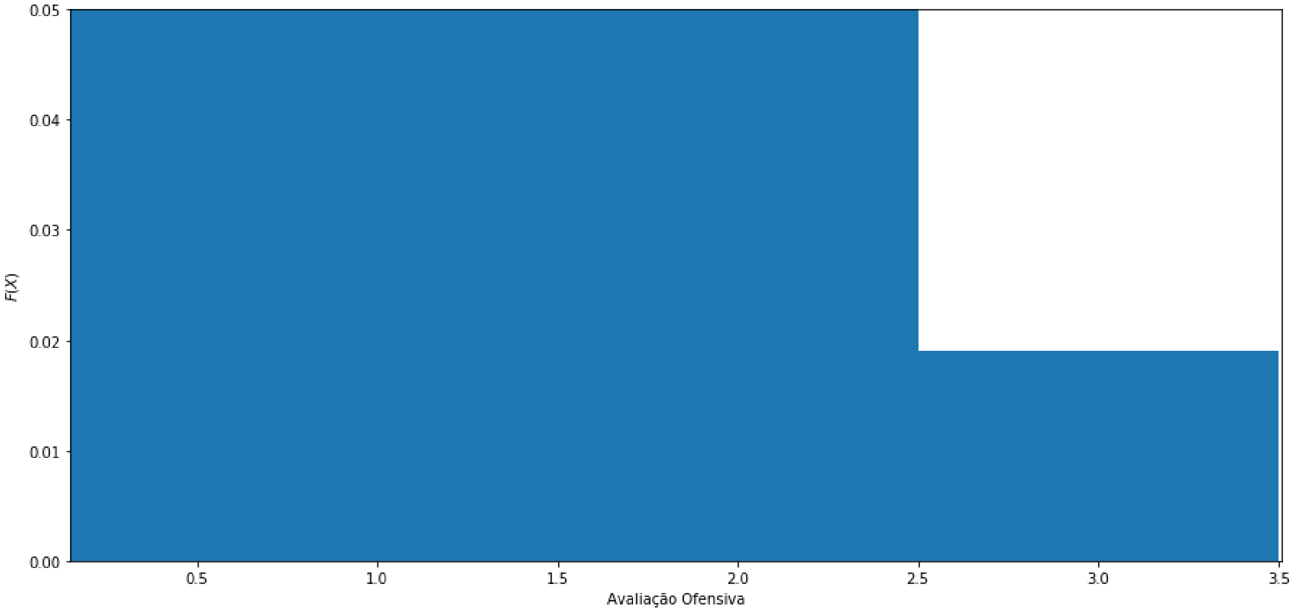


Gráficos de Distribuição de Densidade para Variáveis Contínuas:

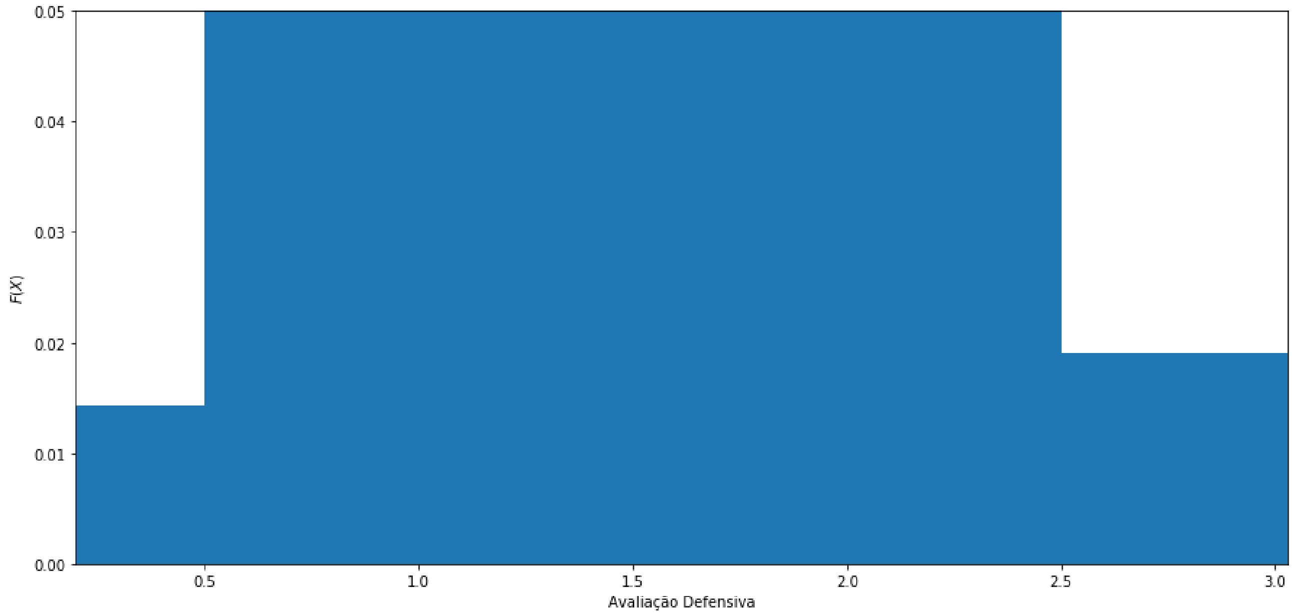
In [253]:
```
ax = dados['Rating de Força Futebolística'].plot(kind='hist', density=True, histtype=
'bar', rwidth=1,
xlim=(min(dados['Rating de Força Futebolística'])-0.05, max(dados['Rating de Força Fu
tebolística'])+0.05), ylim=(0,0.05),
 legend=False, bins=np.arange(len(dados['Rating de Força Futebolística']))-0.5)
ax.set_xlabel(r'Avaliação de Força Futebolística')
ax.set_ylabel(r'$F(X)$')
plt.rcParams['figure.figsize'] = (15,7)
```



In [254]:
```
ax = dados['Rating Ofensivo'].plot(kind='hist', density=True, histtype='bar', rwidth=
1,
                     xlim=(min(dados['Rating Ofensivo'])-0.05, max(dados['Rating Ofen
sivo'])+0.05), ylim=(0,0.05),
                     legend=False, bins=np.arange(len(dados['Rating Ofensivo']))-0.5)
ax.set_xlabel(r'Avaliação Ofensiva')
ax.set_ylabel(r'$F(X)$')
plt.rcParams['figure.figsize'] = (15,7)
```



In [255]:
```
ax = dados['Rating Defensivo'].plot(kind='hist', density=True, histtype='bar', rwidth
=1,
                     xlim=(min(dados['Rating Defensivo'])-0.05, max(dados['Rating Def
ensivo'])+0.05), ylim=(0,0.05),
                     legend=False, bins=np.arange(len(dados['Rating Defensivo']))-0.5
)
ax.set_xlabel(r'Avaliação Defensiva')
ax.set_ylabel(r'$F(X)$')
plt.rcParams['figure.figsize'] = (15,7)
```



6. Teste Kolmogorov-Smirnov

Aplica-se o teste Kolmogorov-Smirnov com o objetivo de comprovação da hipótese nula (H_0), que afirma a normalidade da distribuição amostral. Para tal, procuram-se p-values superiores a 0.8, conforme o especificado. Nesse contexto, duas amostras são selecionadas.

In [256]:
```
x1 = dados['Rating Ofensivo']
y1 = dados['Rating Defensivo']
z1 = dados['Rating de Força Futebolística']
x1.to_numpy()
y1.to_numpy()
z1.to_numpy()
```

```
Out[256]: array([94.74, 93.98, 92.43, 89.22, 89.17, 89.14, 86.9 , 86.72, 85.99,
           84.48, 83.73, 83.71, 82.95, 82.39, 80.41, 80.31, 80.31, 80.29,
           80.16, 80.09, 77.45, 77.3 , 77.  , 76.83, 76.57, 76.14, 75.53,
           75.38, 74.84, 74.68, 74.54, 74.15, 73.99, 72.72, 72.35, 72.35,
           72.2 , 72.04, 71.92, 71.82, 71.33, 71.31, 71.3 , 70.99, 70.92,
           70.46, 70.43, 70.21, 70.1 , 69.87, 69.62, 69.51, 69.2 , 69.08,
           69.07, 68.93, 68.92, 68.75, 68.72, 68.14, 67.77, 67.45, 67.41,
           67.24, 67.06, 66.86, 66.85, 66.82, 66.51, 66.48, 66.44, 66.42,
           66.23, 66.21, 66.09, 65.92, 65.89, 65.58, 65.51, 65.31, 65.13,
           64.98, 64.81, 64.41, 64.35, 64.21, 64.07, 63.85, 63.74, 63.65,
           63.46, 62.99, 62.95, 62.75, 62.71, 62.69, 62.66, 62.65, 62.54,
           62.47, 62.45, 62.37, 62.17, 62.07, 61.92, 61.77, 61.59, 61.39,
           61.35, 61.22, 61.18, 61.18, 61.13, 60.84, 60.59, 60.58, 60.4 ,
           59.71, 59.52, 59.43, 59.41, 59.36, 59.35, 58.69, 58.5 , 58.45,
           58.41, 58.32, 58.13, 57.9 , 57.85, 57.68, 57.54, 57.47, 57.36,
           57.14, 57.12, 57.11, 57.01, 56.96, 56.93, 56.93, 56.89, 56.5 ,
           56.33, 56.27, 56.17, 56.09, 55.92, 55.44, 55.27, 55.16, 55.06,
           54.57, 54.51, 54.4 , 54.32, 54.03, 53.92, 53.73, 53.41, 53.31,
           53.3 , 53.28, 53.28, 53.22, 53.21, 52.96, 52.92, 52.74, 52.69,
           52.38, 52.38, 52.22, 52.21, 52.15, 52.1 , 51.91, 51.81, 51.74,
           51.44, 51.39, 51.26, 51.24, 51.16, 51.14, 51.07, 50.87, 50.84,
           50.83, 50.8 , 50.77, 50.7 , 50.64, 50.56, 50.41, 50.41, 50.13,
           49.96, 49.9 , 49.79, 49.47, 49.43, 49.43, 49.29, 49.28, 48.96,
           48.7 , 48.6 , 48.56, 48.48, 48.47, 48.28, 48.19, 47.98, 47.92,
           47.74, 47.59, 47.51, 47.29, 47.27, 47.27, 47.21, 47.  , 46.81,
           46.79, 46.67, 46.62, 46.58, 46.52, 46.52, 46.32, 46.29, 46.26,
           45.96, 45.79, 45.61, 45.34, 45.31, 45.3 , 45.21, 45.17, 45.08,
           44.88, 44.88, 44.82, 44.76, 44.7 , 44.66, 44.65, 44.55, 44.41,
           44.37, 44.31, 44.3 , 44.19, 44.13, 44.11, 44.06, 43.8 , 43.78,
           43.76, 43.66, 43.56, 43.52, 43.34, 43.25, 43.13, 43.08, 42.98,
           42.95, 42.87, 42.82, 42.72, 42.51, 42.47, 42.42, 42.41, 42.34,
           42.29, 42.2 , 42.18, 42.12, 41.98, 41.9 , 41.81, 41.76, 41.76,
           41.71, 41.68, 41.67, 41.67, 41.64, 41.42, 41.41, 41.32, 41.18,
           41.08, 40.98, 40.97, 40.86, 40.8 , 40.79, 40.71, 40.56, 40.44,
           40.33, 40.26, 40.24, 40.23, 40.2 , 40.17, 40.13, 39.99, 39.95,
           39.73, 39.56, 39.45, 39.42, 39.38, 39.35, 39.24, 39.16, 39.03,
           38.99, 38.77, 38.75, 38.61, 38.6 , 38.56, 38.55, 38.54, 38.44,
           38.32, 38.07, 38.06, 37.84, 37.79, 37.78, 37.65, 37.61, 37.58,
           37.53, 37.46, 37.45, 37.38, 37.36, 37.32, 37.25, 37.15, 37.11,
           37.02, 36.98, 36.83, 36.7 , 36.56, 36.56, 36.45, 36.45, 36.34,
           36.25, 36.24, 36.18, 36.12, 36.01, 35.57, 35.51, 35.49, 35.49,
           35.45, 35.35, 35.32, 35.3 , 35.18, 35.05, 35.05, 34.92, 34.91,
           34.82, 34.72, 34.53, 34.27, 34.23, 34.09, 34.08, 34.07, 33.83,
           33.73, 33.68, 33.65, 33.51, 33.5 , 33.39, 33.28, 33.18, 33.13,
           33.13, 33.08, 33.01, 32.99, 32.95, 32.68, 32.48, 32.47, 32.43,
           32.4 , 32.26, 32.13, 32.1 , 32.05, 32.01, 31.99, 31.95, 31.95,
           31.84, 31.83, 31.78, 31.77, 31.74, 31.7 , 31.69, 31.69, 31.6 ,
           31.4 , 31.39, 31.3 , 31.26, 31.26, 31.17, 31.11, 30.98, 30.93,
           30.87, 30.77, 30.68, 30.68, 30.58, 30.32, 30.32, 30.32, 30.3 ,
           30.24, 30.23, 30.15, 30.1 , 30.02, 29.99, 29.94, 29.86, 29.79,
           29.65, 29.65, 29.58, 29.56, 29.52, 29.38, 29.35, 29.28, 29.25,
           29.21, 29.14, 28.96, 28.91, 28.71, 28.67, 28.67, 28.61, 28.6 ,
           28.41, 28.39, 28.35, 28.17, 28.11, 28.06, 28.03, 28.03, 27.96,
           27.95, 27.93, 27.64, 27.53, 27.33, 27.21, 27.21, 27.12, 27.08,
           27.05, 26.85, 26.82, 26.68, 26.58, 26.45, 26.22, 26.08, 25.84,
           25.77, 25.75, 25.75, 25.69, 25.68, 25.48, 25.47, 25.45, 25.04,
           25.02, 25.01, 25.01, 25.  , 24.93, 24.89, 24.75, 24.39, 24.18,
           24.18, 24.08, 24.03, 23.88, 23.58, 23.4 , 23.23, 23.22, 23.15,
           23.11, 23.09, 23.01, 22.93, 22.77, 22.73, 22.72, 22.26, 22.1 ,
           22.08, 22.06, 22.06, 22.02, 21.65, 21.63, 21.61, 21.61, 21.45,
           21.44, 21.38, 21.35, 20.95, 20.9 , 20.76, 20.5 , 20.38, 20.27,
           20.15, 20.  , 19.91, 19.79, 19.74, 19.57, 19.57, 19.47, 19.25,
           18.94, 18.94, 18.93, 18.57, 18.46, 18.22, 18.21, 17.98, 17.93,
           17.79, 17.74, 17.5 , 17.5 , 17.29, 17.01, 16.89, 16.64, 16.6 ,
           16.59, 16.3 , 16.28, 16.21, 16.19, 15.57, 15.45, 15.38, 15.32,
```

```
15.17, 14.79, 14.66, 14.62, 14.51, 14.44, 14.35, 14.3 , 13.98,
13.98, 13.81, 13.81, 13.73, 13.61, 13.54, 13.42, 13.16, 12.05,
11.72, 11.46, 11.43, 11.42, 10.27, 10.1 ,  9.98,  9.95,  9.62,
 9.09,  9.08,  9.03,  8.92,  8.89,  8.56,  8.45,  8.  ,  7.84,
 7.63,  7.42,  7.33,  7.26,  6.92,  6.68,  5.6 ,  5.3 ])
```

In [257]:

```
ks1 = scs.kstest(x1, 'norm')
ks2 = scs.kstest(y1, 'norm')
ks3 = scs.kstest(z1, 'norm')

print("Rating Ofensivo, \np-value=", ks1)
print("Rating Defensivo, \np-value=", ks2)
print("Rating de Força Futebolística, \np-value=", ks3)
```

```
Rating Ofensivo,
p-value= KstestResult(statistic=0.6552477964226743, pvalue=5.190505330200898e-265)
Rating Defensivo,
p-value= KstestResult(statistic=0.7254983037468203, pvalue=0.0)
Rating de Força Futebolística,
p-value= KstestResult(statistic=0.9999999420986596, pvalue=0.0)
```

Os baixíssimos valores do p-value < 0.001 rejeitam a chance de constatação da Hipótese Nula (H_0), que afirma a normalidade da distribuição amostral.

É importante frisar que os valores p-value=0 não é uma verdade, mas uma aproximação. Pois sempre há uma chance de obter os resultados da Hipótese Nula (H_0), por menor ou improvável que seja a chance. É provavel que a hipótese nula tenha sido rejeitada pelo tamanho da amostra.

Vamos tentar provar que há diferença entre os alvos da comparação estatística, confirmando a Hipótese Alternativa (H_1).

6.1 Avaliação pelo Teste de Shapiro

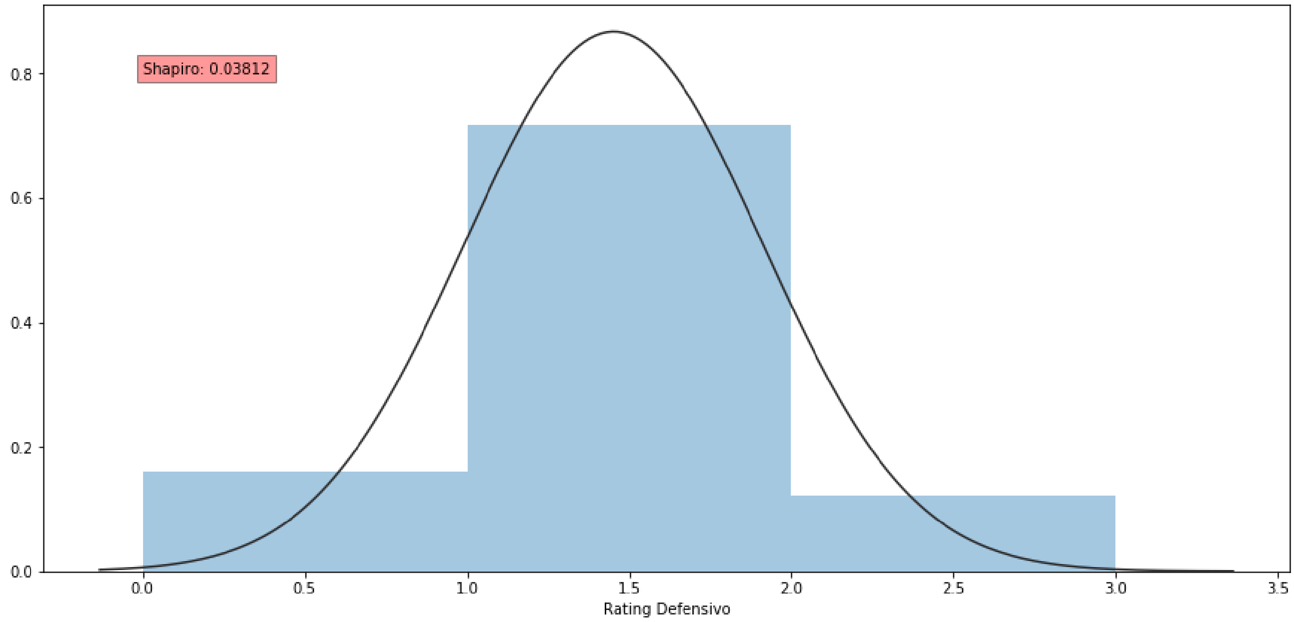
In [258]:

```
s1 = scs.shapiro(x1)
s2 = scs.shapiro(y1)
s3 = scs.shapiro(z1)

print("Rating Ofensivo, \np-value=", s1)
print("Rating Defensivo, \np-value=", s2)
print("Rating de Força Futebolística, \np-value=", ks3)
```

```
Rating Ofensivo,
p-value= (0.9746729135513306, 5.779883682777154e-09)
Rating Defensivo,
p-value= (0.9949859380722046, 0.0381171777844429)
Rating de Força Futebolística,
p-value= KstestResult(statistic=0.9999999420986596, pvalue=0.0)
```

In [259]:
```
sns.distplot(y1, bins=range(0,4,1), fit=scs.norm, kde=False)
print(s2)

plt.text(0, 0.8, 'Shapiro: '+str(round(s2[1], 5) ), bbox=dict(facecolor='red', alpha=
0.4), zorder=4 )
plt.show()
```

```
(0.9949859380722046, 0.0381171777844429)
```



O teste de Shapiro é independente do tamanho da amostra. Por este metódo a única mudança significativa analisando os resultados foram os dados de Rating Defensivo. Porém, o p-value continuou abaixo de 5%. Logo, rejeitamos a hipótese nula novamente.

7. Regressão e Predição

In [260]:
```
Y = dados[continua]
g=sns.pairplot(Y)
```

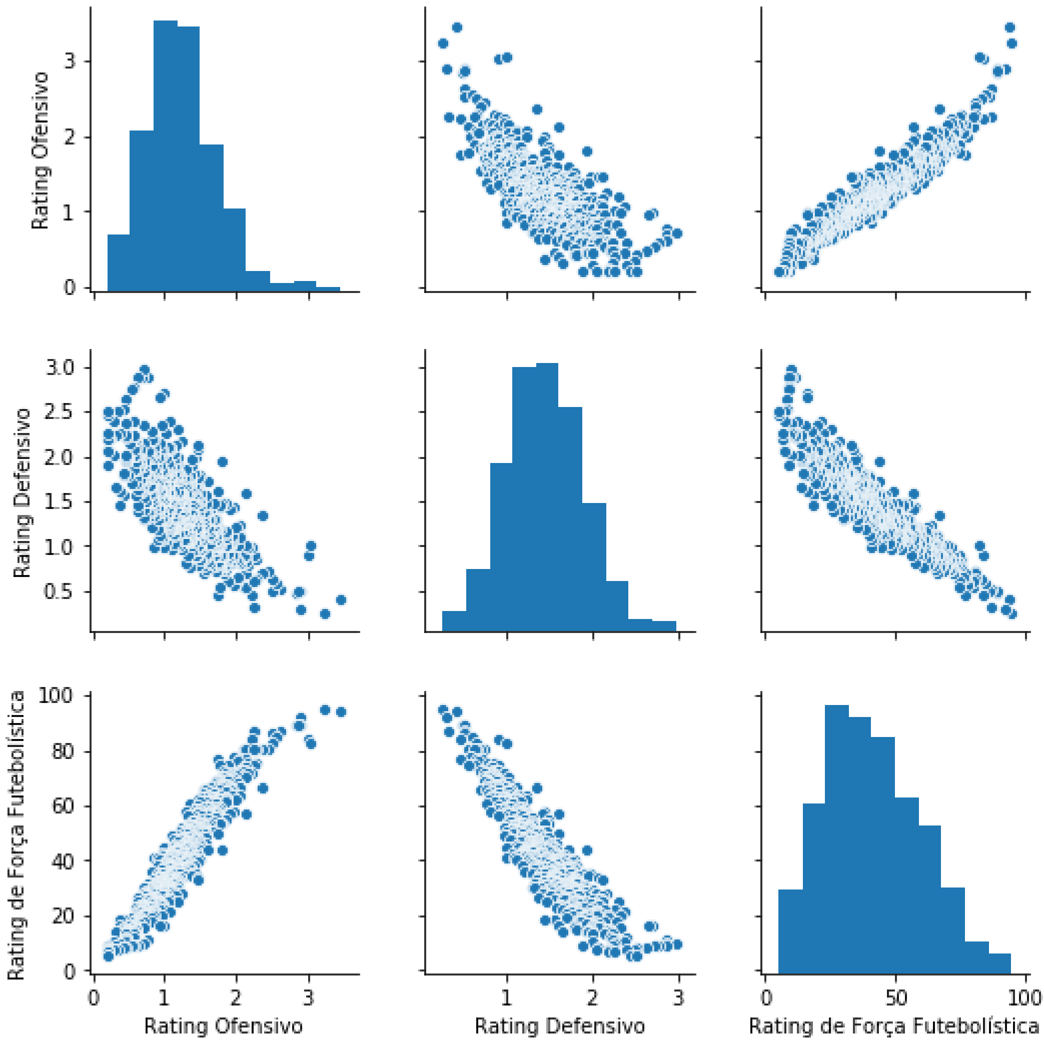
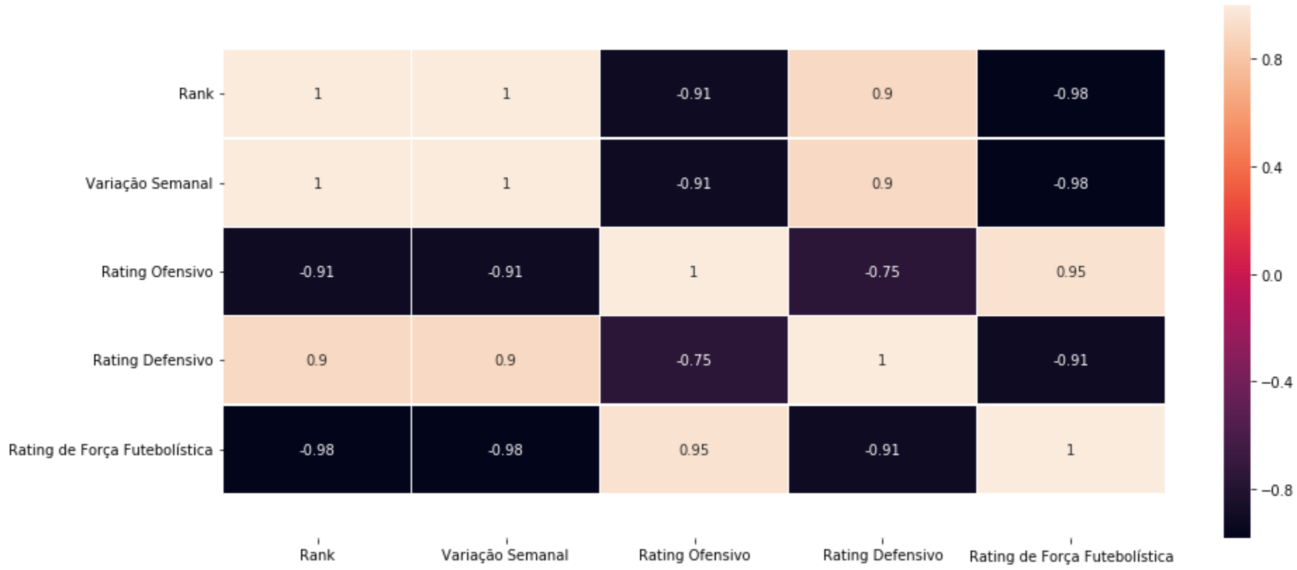


Gráfico matriz de dispersão, onde as diagonais nos mostram o a distribuição de uma única variável em formato de histograma. Enquanto as matrizes triangular inferior e superior mostram a relação entre duas variáveis.

In [273]:
```
corr = dados.corr()
```

In [271]:
```
plt.gcf()
sns.heatmap(corr, linewidths=.5, annot=True)
lim_y = plt.ylim()
plt.ylim(lim_y[0]+0.5, lim_y[1]-0.5)
plt.yticks(rotation=0,va="center")
plt.show()
```



In [272]:
```
cor = Y.corr()
cor
```

Out[272]:

	Rating Ofensivo	Rating Defensivo	Rating de Força Futebolística
Rating Ofensivo	1.000000	-0.753668	0.946446
Rating Defensivo	-0.753668	1.000000	-0.913229
Rating de Força Futebolística	0.946446	-0.913229	1.000000

Como observado, a correlação para Rating Ofensivo e Defensivo com o Rating de Força Futebolística é muito forte. Pois os melhores times fazem muito gols e levam poucos gols. Logo, nos melhores times, o Rating Ofensivo é alto e por isso correlação positiva e o Rating Defensivo é baixo e por isso sua correlação negativa.

Podemos perceber que a correlação entre Rating Ofensivo(Gols feitos) x Rating Defensivo(Gols levados) é forte. Pois também é possível existirem times que fazem muitos gols e levam muitos gols.

7.1 Qualificando a Regressão por Mínimos Quadrados

Calculamos a regressão de mínimos quadrados para dois conjuntos de medidas.

A função scipy.stats.linregress calcula uma regressão de mínimos quadrados para dois conjuntos de medidas: E retorna: a inclinação, interceptação, rvalue, pvalue, erro padrão da estimativa.

In [447]:
```
z2 = z1.values
```

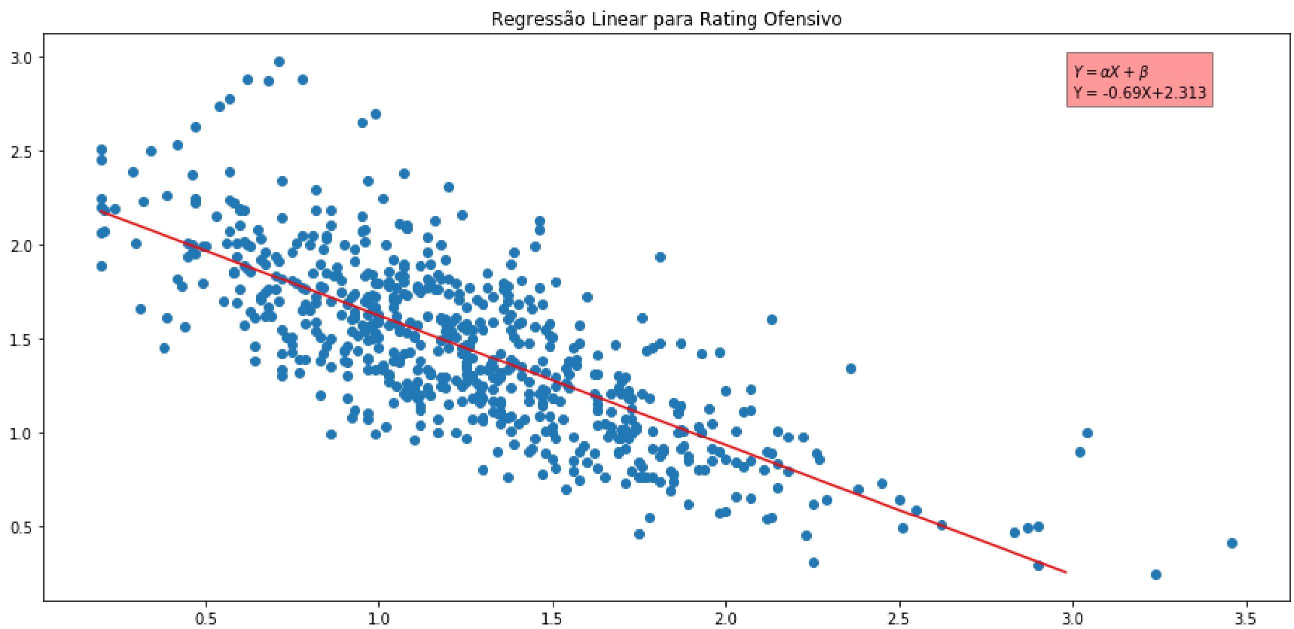
Conjunto Rating Ofensivo x Rating Defensivo

In [496]:
```
b0, b1, r, pvalue, stder  = scs.stats.linregress(x1,y1)
print('Parâmetros: b0 =', b0, 'b1 =', b1)
print('Valor para o teste de hipótese que a inclinação é nula', pvalue)
print("Coeficiente de Determininação:", r**2)
print("Desvio padrão da estimativa:", stder)
```

```
Parâmetros: b0 = -0.6902839119853853 b1 = 2.3126571578637503
Valor para o teste de hipótese que a inclinação é nula 2.210723813121036e-116
Coeficiente de Determininação: 0.5680160212911687
Desvio padrão da estimativa: 0.02404072212028617
```

Coeficiente de Determinação não indica uma boa qualidade de regressão, porém o desvio padrão dos erros indicam uma baixa variabilidade. Pelo teste de hipótese não podemos considerar a inclinação nula.

In [497]:
```
plt.scatter(x1, y1)
b0, b1, r, pvalue, stder  = scs.stats.linregress(x1,y1)
x_1 = np.linspace(x1.min(), y1.max(), 100)
plt.plot(x_1, b0*x_1+b1, 'red')
plt.text(3,2.8, r'$Y = \alpha X + \beta$' +'\nY = '+str(round(b0, 3))+'X+'+str(round(
b1, 3)), bbox=dict(facecolor='red', alpha=0.4))
plt.title('Regressão Linear para Rating Ofensivo')
plt.show()
```



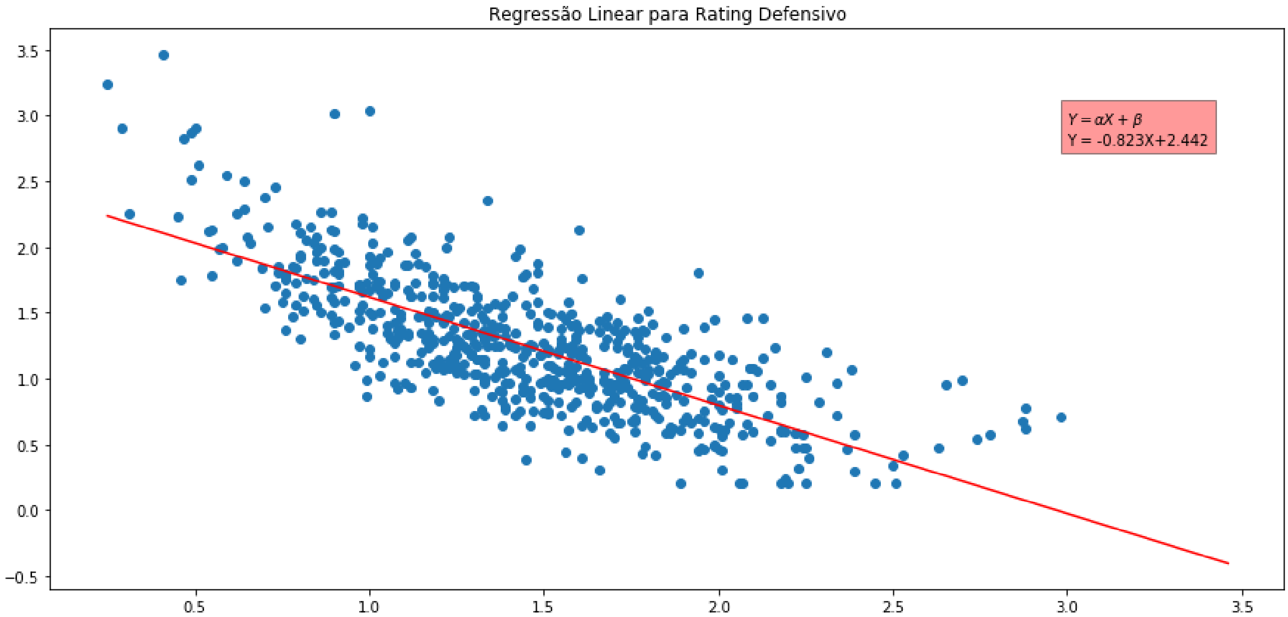
Conjunto Rating Defensivo x Rating Ofensivo

In [498]:
```
b0, b1, r, pvalue, stder = scs.stats.linregress(y1,x1)
print('Parâmetros: b0 =', b0, 'b1 =', b1)
print('Valor para o teste de hipótese que a inclinação é nula', pvalue)
print("Coeficiente de Determininação:", r**2)
print("Desvio padrão da estimativa:", stder)
```

```
Parâmetros: b0 = -0.8228730402501332 b1 = 2.4416159063930873
Valor para o teste de hipótese que a inclinação é nula 2.2107238131206588e-116
Coeficiente de Determininação: 0.568016021291169
Desvio padrão da estimativa: 0.02865844293550236
```

Coeficiente de Determinação não indica uma boa qualidade de regressão, porém o desvio padrão dos erros indicam uma baixa variabilidade. Pelo teste de hipótese não podemos considerar a inclinação nula.

```
In [500]: plt.scatter(y1, x1)
b0, b1, r, pvalue, stder  = scs.stats.linregress(y1,x1)
x_1 = np.linspace(y1.min(), x1.max(), 100)
plt.plot(x_1, b0*x_1+b1, 'red')
plt.text(3,2.8, r'$Y = \alpha X + \beta$' +'\nY = '+str(round(b0, 3))+'X+'+str(round(
b1, 3)), bbox=dict(facecolor='red', alpha=0.4))
plt.title('Regressão Linear para Rating Defensivo')
plt.show()
```



Conjunto Rating Ofensivo x Rating de Força Futebolística

```
In [501]: b0, b1, r, pvalue, stder = scs.stats.linregress(x1,z2)
print('Parâmetros: b0 =', b0, 'b1 =', b1)
print('Valor para o teste de hipótese que a inclinação é nula', pvalue)
print("Coeficiente de Determininação:", r**2)
print("Desvio padrão da estimativa:", stder)
```

```
Parâmetros: b0 = 35.07077738595165 b1 = -2.1742698718360316
Valor para o teste de hipótese que a inclinação é nula 4.79236033130055e-310
Coeficiente de Determininação: 0.8957601475585099
Desvio padrão da estimativa: 0.4777851505213695
```

Coeficiente de Determinação indica uma boa qualidade de regressão, e o desvio padrão dos erros indicam uma baixa variabilidade. Pelo teste de hipótese não podemos considerar a inclinação nula.

Conjunto Rating Defensivo x Rating de Força Futebolística

```
In [502]: b0, b1, r, pvalue, stder = scs.stats.linregress(y1,z2)
print('Parâmetros: b0 =', b0, 'b1 =', b1)
print('Valor para o teste de hipótese que a inclinação é nula', pvalue)
print("Coeficiente de Determininação:", r**2)
print("Desvio padrão da estimativa:", stder)
```

```
Parâmetros: b0 = -36.94722498872811 b1 = 95.19964948601775
Valor para o teste de hipótese que a inclinação é nula 1.1391676904182119e-246
Coeficiente de Determininação: 0.8339874584812345
Desvio padrão da estimativa: 0.6583225689883281
```

Coeficiente de Determinação indica uma boa qualidade de regressão, e o desvio padrão dos erros indicam uma baixa variabilidade. Pelo teste de hipótese não podemos considerar a inclinação nula.

A partir dessa análise prêvia podemos fazer um estudo mais detalhado do conjunto Rating Ofensivo x Rating de Força Futebolística pois foi o que apresentou melhor coeficiente de determinação.

In [504]:

```
xm = stats.mean(x1)
ym = stats.mean(z2)
xy = np.multiply(x1,z2)
sxy = sum(xy)
xx = np.multiply(x1,x1)
sxx = sum(xx)
n = len(x1)

b1 = (sxy-n*xm*ym)/(sxx-n*xm*xm)
b0 = ym-b1*xm

est=[]
for i in range(len(x1)):
    est.append(b0+(b1*x[i]))
print(est)
```

[6.593424474651769, 7.99625557008984, 8.697671117808875, 12.204748856404052, 13.6075
79951842125, 13.958287725701641, 14.308995499561156, 15.010411047280193, 15.01041104
7280193, 15.361118821139712, 15.7118265949923, 16.763949916577783, 17.114657690437
3, 17.1146576904373, 17.816073238156335, 18.166781012015853, 18.517488785875372, 19.
569612107453924, 19.569612107453924, 20.27102765517296, 20.27102765517296, 20.621735
42903248, 20.972443202891995, 22.024566524470547, 22.375274298330066, 22.37527429833
0066, 22.725982072189584, 23.427397619908618, 23.427397619908618, 23.77810539376813
6, 23.778105393768136, 24.128813167627655, 24.479520941487174, 24.479520941487174, 2
4.479520941487174, 24.479520941487174, 24.479520941487174, 24.479520941487174, 25.18
093648920621, 25.18093648920621, 25.531644263065726, 25.531644263065726, 25.53164426
3065726, 25.531644263065726, 25.882352036925244, 25.882352036925244, 25.882352036925
244, 25.882352036925244, 26.233059810784763, 26.233059810784763, 26.583767584644278,
26.583767584644278, 26.934475358503796, 27.285183132363315, 27.285183132363315, 27.2
85183132363315, 27.63589090622283, 27.63589090622283, 27.63589090622283, 27.63589090
622283, 27.98659868008235, 27.98659868008235, 27.98659868008235, 28.337306453941867,
28.337306453941867, 29.038722001660904, 29.038722001660904, 29.038722001660904, 29.0
38722001660904, 29.038722001660904, 29.389429775520423, 29.389429775520423, 29.38942
9775520423, 29.389429775520423, 29.389429775520423, 29.389429775520423, 29.389429775
520423, 29.389429775520423, 29.740137549379938, 29.740137549379938, 29.7401375493799
38, 29.740137549379938, 29.740137549379938, 29.740137549379938, 30.441553097098975,
30.441553097098975, 30.792260870958486, 31.493676418677524, 31.844384192537042, 31.8
44384192537042, 31.844384192537042, 31.844384192537042, 31.844384192537042, 31.84438
4192537042, 32.19509196639656, 32.19509196639656, 32.19509196639656, 32.195091966396
56, 32.54579974025608, 32.54579974025608, 32.54579974025608, 32.8965075141156, 32.89
65075141156, 32.8965075141156, 32.8965075141156, 32.8965075141156, 32.8965075141156,
32.8965075141156, 32.8965075141156, 33.247215287975116, 33.247215287975116, 33.24721
5287975116, 33.247215287975116, 33.247215287975116, 33.247215287975116, 33.597923061
834635, 33.597923061834635, 33.597923061834635, 33.597923061834635, 33.9486308356941
5, 33.94863083569415, 33.94863083569415, 33.94863083569415, 33.94863083569415, 34.29
933860955367, 34.29933860955367, 34.65004638341319, 34.65004638341319, 34.6500463834
1319, 34.65004638341319, 35.00075415727271, 35.35146193113222, 35.35146193113222, 3
5.35146193113222, 35.35146193113222, 35.35146193113222, 35.35146193113222, 35.351461
93113222, 35.35146193113222, 35.70216970499174, 35.70216970499174, 35.7021697049917
4, 35.70216970499174, 36.05287747885126, 36.05287747885126, 36.05287747885126, 36.40
3585252710776, 36.403585252710776, 36.403585252710776, 36.754293026570295, 36.754293
026570295, 36.754293026570295, 36.754293026570295, 36.754293026570295, 37.1050008004
2981, 37.10500080042981, 37.10500080042981, 37.10500080042981, 37.10500080042981, 3
7.455708574289325, 37.455708574289325, 37.806416348148844, 37.806416348148844, 37.80
6416348148844, 38.15712412200836, 38.15712412200836, 38.15712412200836, 38.507831895
86788, 38.50783189586788, 38.50783189586788, 38.50783189586788, 38.8585396697274, 3
8.8585396697274, 38.8585396697274, 38.8585396697274, 38.8585396697274, 39.2092474435
8692, 39.20924744358692, 39.20924744358692, 39.20924744358692, 39.20924744358692, 3
9.20924744358692, 39.20924744358692, 39.20924744358692, 39.20924744358692, 39.209247
44358692, 39.55995521744643, 39.55995521744643, 39.91066299130595, 39.9106629913059
5, 39.91066299130595, 40.261370765165466, 40.261370765165466, 40.261370765165466, 4
0.261370765165466, 40.261370765165466, 40.261370765165466, 40.261370765165466, 40.26
1370765165466, 40.261370765165466, 40.612078539024985, 40.612078539024985, 40.612078
539024985, 40.612078539024985, 40.962786312884504, 40.962786312884504, 40.9627863128
84504, 40.962786312884504, 40.962786312884504, 40.962786312884504, 41.3134940867440
2, 41.31349408674402, 41.31349408674402, 41.31349408674402, 41.31349408674402, 41.66
420186060354, 41.66420186060354, 41.66420186060354, 41.66420186060354, 41.6642018606
0354, 41.66420186060354, 41.66420186060354, 42.01490963446306, 42.01490963446306, 4
2.01490963446306, 42.36561740832258, 42.36561740832258, 42.36561740832258, 42.365617
40832258, 42.36561740832258, 42.716325182182096, 42.716325182182096, 43.067032956041
615, 43.067032956041615, 43.067032956041615, 43.067032956041615, 43.067032956041615,
43.067032956041615, 43.41774072990113, 43.41774072990113, 43.41774072990113, 43.4177
4072990113, 43.41774072990113, 43.41774072990113, 43.76844850376065, 43.768448503760
65, 43.76844850376065, 43.76844850376065, 43.76844850376065, 43.76844850376065, 43.7
6844850376065, 44.119156277620164, 44.119156277620164, 44.119156277620164, 44.119156
277620164, 44.46986405147968, 44.46986405147968, 44.46986405147968, 44.4698640514796
8, 44.46986405147968, 44.46986405147968, 44.46986405147968, 44.46986405147968, 44.46
986405147968, 44.46986405147968, 44.8205718253392, 44.8205718253392, 44.820571825339
2, 45.17127959919872, 45.17127959919872, 45.17127959919872, 45.17127959919872, 45.52
198737305824, 45.52198737305824, 45.52198737305824, 45.52198737305824, 45.8726951469
17756, 45.872695146917756, 45.872695146917756, 45.872695146917756, 46.2234029207772

7, 46.22340292077727, 46.22340292077727, 46.22340292077727, 46.22340292077727, 46.22
340292077727, 46.22340292077727, 46.574110694636786, 46.574110694636786, 46.57411069
4636786, 46.574110694636786, 46.574110694636786, 46.574110694636786, 46.574110694636
786, 46.924818468496305, 46.924818468496305, 46.924818468496305, 47.275526242355824,
47.275526242355824, 47.62623401621534, 47.62623401621534, 47.62623401621534, 47.6262
3401621534, 47.62623401621534, 47.62623401621534, 47.97694179007486, 47.976941790074
86, 47.97694179007486, 47.97694179007486, 48.32764956393437, 48.32764956393437, 48.3
2764956393437, 48.32764956393437, 48.32764956393437, 48.67835733779389, 48.678357337
79389, 48.67835733779389, 48.67835733779389, 48.67835733779389, 48.67835733779389, 4
9.02906511165341, 49.02906511165341, 49.02906511165341, 49.02906511165341, 49.029065
11165341, 49.37977288551293, 49.37977288551293, 49.37977288551293, 49.3797728855129
3, 49.730480659372446, 49.730480659372446, 49.730480659372446, 49.730480659372446, 4
9.730480659372446, 50.081188433231965, 50.081188433231965, 50.081188433231965, 50.43
1896207091484, 50.431896207091484, 50.431896207091484, 50.431896207091484, 50.782603
980951, 50.782603980951, 50.782603980951, 50.782603980951, 50.782603980951, 50.78260
3980951, 50.782603980951, 50.782603980951, 50.782603980951, 51.13331175481052, 51.13
331175481052, 51.13331175481052, 51.13331175481052, 51.48401952867004, 51.4840195286
7004, 51.83472730252956, 51.83472730252956, 51.83472730252956, 51.83472730252956, 5
1.83472730252956, 51.83472730252956, 52.185435076389076, 52.185435076389076, 52.1854
35076389076, 52.185435076389076, 52.185435076389076, 52.185435076389076, 52.18543507
6389076, 52.536142850248595, 52.536142850248595, 52.536142850248595, 52.536142850248
595, 52.536142850248595, 52.886850624108106, 52.886850624108106, 52.886850624108106,
52.886850624108106, 52.886850624108106, 53.237558397967625, 53.237558397967625, 53.2
37558397967625, 53.237558397967625, 53.237558397967625, 53.237558397967625, 53.23755
8397967625, 53.588266171827144, 53.588266171827144, 53.588266171827144, 53.588266171
827144, 53.588266171827144, 53.588266171827144, 53.588266171827144, 53.5882661718271
44, 53.93897394568666, 53.93897394568666, 53.93897394568666, 53.93897394568666, 53.9
3897394568666, 53.93897394568666, 54.28968171954618, 54.28968171954618, 54.289681719
54618, 54.28968171954618, 54.28968171954618, 54.28968171954618, 54.28968171954618, 5
4.6403894934057, 54.6403894934057, 54.6403894934057, 54.6403894934057, 54.6403894934
057, 54.99109726726521, 54.99109726726521, 54.99109726726521, 54.99109726726521, 55.
34180504112473, 55.34180504112473, 55.69251281498425, 55.69251281498425, 56.04322058
884377, 56.04322058884377, 56.04322058884377, 56.393928362703285, 56.39392836270328
5, 56.393928362703285, 56.393928362703285, 56.393928362703285, 56.744636136562804, 5
6.744636136562804, 56.744636136562804, 56.744636136562804, 56.744636136562804, 57.09
5343910422315, 57.095343910422315, 57.095343910422315, 57.095343910422315, 57.446051
684281834, 57.446051684281834, 57.446051684281834, 57.446051684281834, 57.4460516842
81834, 57.79675945814135, 57.79675945814135, 57.79675945814135, 57.79675945814135, 5
7.79675945814135, 57.79675945814135, 58.14746723200087, 58.14746723200087, 58.147467
23200087, 58.14746723200087, 58.14746723200087, 58.14746723200087, 58.1474672320008
7, 58.14746723200087, 58.49817500586039, 58.49817500586039, 58.49817500586039, 58.84
888277971991, 58.84888277971991, 58.84888277971991, 58.84888277971991, 59.1995905535
7943, 59.19959055357943, 59.19959055357943, 59.550298327438945, 59.550298327438945,
59.550298327438945, 59.550298327438945, 59.550298327438945, 59.550298327438945, 59.5
50298327438945, 59.550298327438945, 59.901006101298464, 59.901006101298464, 59.90100
6101298464, 59.901006101298464, 59.901006101298464, 59.901006101298464, 59.901006101
298464, 60.25171387515798, 60.25171387515798, 60.25171387515798, 60.25171387515798,
60.25171387515798, 60.6024216490175, 60.6024216490175, 60.6024216490175, 60.60242164
90175, 60.6024216490175, 60.6024216490175, 60.95312942287702, 60.95312942287702, 60.
95312942287702, 60.95312942287702, 61.30383719673654, 61.30383719673654, 61.30383719
673654, 61.65454497059605, 61.65454497059605, 61.65454497059605, 61.65454497059605,
61.65454497059605, 62.005252744455575, 62.005252744455575, 62.005252744455575, 62.00
5252744455575, 62.005252744455575, 62.35596051831509, 62.35596051831509, 62.35596051
831509, 62.70666829217461, 62.70666829217461, 62.70666829217461, 62.70666829217461,
62.70666829217461, 63.057376066034124, 63.057376066034124, 63.057376066034124, 63.40
808383989365, 63.75879161375315, 63.75879161375315, 64.10949938761267, 64.1094993876
1267, 64.10949938761267, 64.46020716147218, 64.46020716147218, 64.46020716147218, 6
4.46020716147218, 64.46020716147218, 64.81091493533171, 64.81091493533171, 65.161622
70919122, 65.16162270919122, 65.16162270919122, 65.86303825691026, 65.8630382569102
6, 65.86303825691026, 65.86303825691026, 66.21374603076978, 66.5644538046293, 66.564
4538046293, 66.5644538046293, 66.5644538046293, 66.5644538046293, 67.26586935234833,
67.26586935234833, 67.61657712620786, 67.61657712620786, 67.61657712620786, 67.61657
712620786, 67.96728490006737, 67.96728490006737, 67.96728490006737, 67.9672849000673
7, 67.96728490006737, 67.96728490006737, 68.31799267392688, 68.31799267392688, 68.31
799267392688, 68.31799267392688, 68.31799267392688, 68.6687004477864, 68.66870044778
64, 69.01940822164592, 69.01940822164592, 69.37011599550544, 69.72082376936496, 69.7

```
2082376936496, 69.72082376936496, 70.07153154322448, 70.42223931708399, 70.422239317
08399, 70.42223931708399, 70.77294709094352, 70.77294709094352, 70.77294709094352, 7
1.12365486480303, 71.12365486480303, 71.47436263866256, 71.47436263866256, 71.474362
63866256, 71.82507041252207, 72.52648596024109, 72.52648596024109, 72.8771937341006
2, 73.22790150796013, 73.22790150796013, 73.57860928181965, 74.28002482953869, 74.28
002482953869, 74.28002482953869, 74.28002482953869, 74.28002482953869, 74.6307326033
982, 74.6307326033982, 74.98144037725773, 75.68285592497676, 75.68285592497676, 75.6
8285592497676, 76.03356369883628, 76.3842714726958, 76.3842714726958, 76.73497924655
531, 76.73497924655531, 76.73497924655531, 77.08568702041482, 78.13781034199339, 78.
83922588971242, 79.89134921129097, 79.89134921129097, 80.94347253286954, 81.29418030
672903, 81.64488808058856, 81.64488808058856, 83.74913472374567, 85.50267359304326,
85.85338136690277, 86.5547969146218, 90.06187465321698, 90.76329020093601, 92.516829
07023361, 93.91966016567169, 95.32249126110975, 98.47886122584542, 98.8295689997049
2, 98.82956899970492, 102.3366467383001]
```

In [505]:

```
sse=0
for i in range(len(x1)):
    sse+=(z2[i]-b0-b1*x1[i])*(z2[i]-b0-b1*x1[i])
sst=0
for i in range(len(y)):
    sst+=(z2[i]-ym)*(z2[i]-ym)
r=(sst-sse)/sst
print(r)
```

```
0.8957601475585101
```

Confirmando o R da Função scipy.stats.linregress. Podemos concluir que a qualidade da regressão é alta.

In [506]:

```
ey=[]
for i in range(len(z2)):
    ey.append(z2[i]-est[i])
print(ey)
```

[88.14657552534823, 85.98374442991016, 83.73232888219113, 77.01525114359595, 75.5624
2004815788, 75.18171227429836, 72.59100450043886, 71.7095889527198, 70.9795889527198
1, 69.1188817886029, 68.01817340500077, 66.94605008342221, 65.8353423095627, 65.275
3423095627, 62.5939267618436, 62.14321898798415, 61.7925121412463, 60.720387892546
086, 60.590387892546076, 59.818972344827046, 57.178972344827045, 56.67826457096751 4,
56.02755679710805, 54.805433475529455, 54.194725701669924, 53.7647257016699 3, 52.80
401792781042, 51.95260238009138, 51.412602380091386, 50.90189460623187, 50.761894606
23187, 50.02118683237235, 49.51047905851282, 48.240479058512825, 47.87047905851282,
47.87047905851282, 47.72047905851283, 47.56047905851283, 46.73906351079379, 46.63906
351079378, 45.79835573693427, 45.77835573693427, 45.76835573693427, 45.4583557369342
66, 45.03764796307476, 44.57764796307475, 44.547647963074766, 44.32764796307475, 43.
86694018921523, 43.63694018921524, 43.03623241535573, 42.92623241535573, 42.26552464
149621, 41.79481686763668, 41.78481686763668, 41.64481686763669, 41.284109093777175,
41.114109093777174, 41.08410909377717, 40.504109093777174, 39.783401319917644, 39.46
340131991765, 39.423401319917645, 38.90269354605813, 38.72269354605814, 37.821277998
3391, 37.811277998339094, 37.78127799833909, 37.471277998339104, 37.4412779983391, 3
7.05057022447957, 37.030570224479575, 36.84057022447958, 36.82057022447957, 36.70057
022447958, 36.530570224479575, 36.500570224479574, 36.19057022447957, 35.7698624506 2
007, 35.569862450620064, 35.38986245062006, 35.239862450620066, 35.069862450620064,
34.66986245062006, 33.90844690290102, 33.76844690290102, 33.27773912904151, 32.35632
358132248, 31.89561580746296, 31.805615807462956, 31.61561580746296, 31.145615807462
96, 31.10561580746296, 30.905615807462958, 30.51490803360344, 30.494908033603437, 3
0.464908033603436, 30.454908033603438, 29.99420025974392, 29.92420025974392, 29.9042
00259743924, 29.4734924858844, 29.273492485884404, 29.173492485884402, 29.0234924858
84404, 28.873492485884405, 28.693492485884406, 28.493492485884403, 28.4534924858844 0
4, 27.972784712024882, 27.932784712024883, 27.932784712024883, 27.882784712024886, 2
7.592784712024887, 27.342784712024887, 26.982076938165363, 26.802076938165364, 26.11
2076938165366, 25.922076938165368, 25.481369164305846, 25.461369164305843, 25.411369
164305846, 25.401369164305848, 24.741369164305844, 24.200661390446328, 24.1506613904
4633, 23.759953616586806, 23.66995361658681, 23.479953616586812, 23.249953616586808,
22.849245842727292, 22.32853806886778, 22.18853806886778, 22.118538068867778, 22.008
53806886778, 21.78853806886778, 21.768538068867777, 21.75853806886778, 21.6585380688
67777, 21.25783029500826, 21.22783029500826, 21.22783029500826, 21.18783029500826, 2
0.447122521148742, 20.27712252114874, 20.217122521148745, 19.766414747289225, 19.686
414747289227, 19.516414747289225, 18.685706973429703, 18.515706973429708, 18.4057069
734297, 18.305706973429707, 17.815706973429705, 17.404999199570184, 17.2949991995701
85, 17.214999199570187, 16.924999199570188, 16.814999199570188, 16.274291425710672,
15.954291425710672, 15.503583651851159, 15.493583651851154, 15.473583651851158, 15.1
22875877991639, 15.062875877991637, 15.052875877991639, 14.45216810413212, 14.412168
104132121, 14.232168104132121, 14.182168104132117, 13.521460330272603, 13.5214603302
72603, 13.3614603302726, 13.351460330272602, 13.2914603302726, 12.890752556413084, 1
2.700752556413079, 12.600752556413084, 12.530752556413084, 12.23075255641308, 12.180
752556413083, 12.05075255641308, 12.030752556413084, 11.950752556413079, 11.93075255
6413083, 11.510044782553571, 11.310044782553568, 10.929337008694056, 10.919337008694
05, 10.88933700869405, 10.508629234834537, 10.438629234834536, 10.378629234834534, 1
0.298629234834536, 10.14862923483453, 10.14862923483453, 9.868629234834536, 9.698629
234834534, 9.638629234834532, 9.177921460975014, 8.857921460975014, 8.81792146097501
5, 8.817921460975015, 8.327213687115496, 8.317213687115498, 7.997213687115497, 7.737
213687115499, 7.637213687115498, 7.597213687115499, 7.166505913255975, 7.15650591325
5977, 6.966505913255979, 6.876505913255976, 6.666505913255975, 6.255798139396461, 6.
075798139396461, 5.925798139396463, 5.845798139396457, 5.6257981393964585, 5.6057981
39396462, 5.605798139396462, 5.195090365536942, 4.985090365536941, 4.79509036553694
3, 4.424382591677421, 4.304382591677424, 4.25438259167742, 4.2143825916774205, 4.154
382591677425, 3.8036748178179067, 3.603674817817904, 3.222967043958384, 3.1929670439
58383, 2.892967043958386, 2.722967043958384, 2.5429670439583845, 2.2729670439583884,
1.8922592700988687, 1.8822592700988636, 1.7922592700988673, 1.7522592700988682, 1.66
22592700988648, 1.462259270098869, 1.1115514962393505, 1.0515514962393482, 0.9915514
962393459, 0.9315514962393507, 0.8915514962393445, 0.8815514962393465, 0.78155149623
93451, 0.290843722379833, 0.2508437223798339, 0.1908437223798387, 0.180843722379833
6, -0.2798640514796844, -0.3398640514796796, -0.3598640514796827, -0.409864051479679
86, -0.669864051479685, -0.689864051479681, -0.7098640514796841, -0.809864051479685
5, -0.9098640514796799, -0.949864051479679, -1.4805718253391973, -1.570571825339200
7, -1.6905718253391981, -2.091279599198721, -2.1912795991987224, -2.221279599198716
4, -2.301279599198722, -2.7019873730582376, -2.801987373058239, -3.01198737305824, -
3.051987373058239, -3.4526951469177547, -3.46269514691776, -3.532695146917753, -3.58

26951469177573, -4.023402920777265, -4.043402920777268, -4.10340292077727, -4.243402
920777271, -4.323402920777269, -4.413402920777266, -4.46340292077727, -4.81411069463
6788, -4.864110694636786, -4.894110694636787, -4.904110694636785, -4.90411069463678
5, -4.934110694636786, -5.154110694636785, -5.514818468496308, -5.604818468496305, -
5.744818468496305, -6.195526242355825, -6.295526242355827, -6.656234016215343, -6.76
6234016215343, -6.826234016215345, -6.836234016215343, -6.916234016215341, -7.066234
01621534, -7.536941790074863, -7.646941790074862, -7.716941790074863, -7.73694179007
4859, -8.097649563934375, -8.12764956393437, -8.15764956393437, -8.19764956393437, -
8.33764956393437, -8.728357337793888, -8.948357337793894, -9.118357337793888, -9.228
357337793888, -9.258357337793889, -9.298357337793888, -9.679065111653408, -9.7890651
11653407, -9.869065111653413, -9.999065111653408, -10.039065111653407, -10.609772885
512925, -10.629772885512928, -10.769772885512928, -10.779772885512926, -11.170480659
372444, -11.18048065937245, -11.190480659372447, -11.290480659372449, -11.4104806593
72446, -12.011188433231965, -12.021188433231963, -12.241188433231962, -12.6418962070
91484, -12.651896207091482, -12.781896207091485, -12.821896207091484, -13.2026039809
51004, -13.252603980951001, -13.322603980951001, -13.332603980951, -13.402603980951,
-13.422603980951003, -13.462603980951002, -13.532603980951002, -13.632603980951004,
-14.023311754810521, -14.113311754810518, -14.153311754810524, -14.303311754810522,
-14.784019528670036, -14.924019528670037, -15.274727302529556, -15.384727302529555,
-15.384727302529555, -15.494727302529554, -15.584727302529558, -15.594727302529556,
-16.005435076389077, -16.06543507638908, -16.17543507638908, -16.615435076389076, -1
6.67543507638908, -16.695435076389074, -16.695435076389074, -17.086142850248592, -1
7.186142850248594, -17.216142850248595, -17.236142850248598, -17.356142850248595, -1
7.83685062410811, -17.83685062410811, -17.966850624108105, -17.97685062410811, -18.0
66850624108106, -18.517558397967626, -18.707558397967624, -18.967558397967622, -19.0
07558397967628, -19.14755839796762, -19.157558397967627, -19.167558397967625, -19.75
8266171827145, -19.858266171827147, -19.908266171827144, -19.938266171827145, -20.07
8266171827146, -20.088266171827144, -20.198266171827143, -20.308266171827142, -20.75
8973945686662, -20.80897394568666, -20.80897394568666, -20.858973945686664, -20.9289
73945686664, -20.94897394568666, -21.339681719546178, -21.60968171954618, -21.809681
719546184, -21.819681719546182, -21.85968171954618, -21.889681719546182, -22.0296817
19546183, -22.510389493405697, -22.540389493405698, -22.590389493405702, -22.6303894
934057, -22.6503894934057, -23.04109726726521, -23.04109726726521, -23.1510972672652
1, -23.161097267265212, -23.561805041124728, -23.57180504112473, -23.95251281498425,
-23.99251281498425, -24.353220588843765, -24.353220588843765, -24.443220588843765, -
24.993928362703286, -25.003928362703284, -25.093928362703284, -25.133928362703283, -
25.133928362703283, -25.574636136562802, -25.634636136562804, -25.764636136562803, -
25.814636136562804, -25.874636136562803, -26.325343910422315, -26.415343910422315, -
26.415343910422315, -26.515343910422317, -27.126051684281833, -27.126051684281833, -
27.126051684281833, -27.146051684281833, -27.206051684281835, -27.56675945814135, -2
7.646759458141354, -27.69675945814135, -27.776759458141353, -27.806759458141354, -2
7.85675945814135, -28.28746723200087, -28.35746723200087, -28.497467232000872, -28.4
97467232000872, -28.567467232000872, -28.587467232000872, -28.62746723200087, -28.76
746723200087, -29.148175005860388, -29.218175005860388, -29.24817500586039, -29.6388
82779719907, -29.708882779719907, -29.888882779719907, -29.938882779719908, -30.4895
90553579426, -30.529590553579425, -30.529590553579425, -30.940298327438946, -30.9502
98327438944, -31.140298327438945, -31.160298327438944, -31.200298327438944, -31.3802
98327438943, -31.440298327438946, -31.490298327438946, -31.871006101298462, -31.8710
06101298462, -31.941006101298463, -31.951006101298464, -31.971006101298464, -32.2610
0610129846, -32.37100610129846, -32.921713875157984, -33.04171387515798, -33.0417138
7515798, -33.131713875157985, -33.171713875157984, -33.552421649017504, -33.75242164
90175, -33.7824216490175, -33.9224216490175, -34.0224216490175, -34.1524216490175, -
34.73312942287702, -34.87312942287702, -35.11312942287702, -35.183129422877016, -35.
55383719673654, -35.55383719673654, -35.61383719673654, -35.97454497059605, -36.1745
44970596045, -36.18454497059605, -36.20454497059605, -36.61454497059605, -36.9852527
4445558, -36.99525274445557, -36.99525274445557, -37.005252744455575, -37.0752527444
55575, -37.465960518315086, -37.60596051831509, -37.965960518315086, -38.52666829217
461, -38.52666829217461, -38.626668292174614, -38.67666829217461, -38.8266682921746
2, -39.477376066034125, -39.657376066034125, -39.82737606603412, -40.18808383989365,
-40.60879161375315, -40.64879161375315, -41.01949938761267, -41.09949938761267, -41.
17949938761267, -41.69020716147219, -41.73020716147218, -41.740207161472185, -42.200
20716147218, -42.36020716147218, -42.73091493533171, -42.75091493533171, -43.1016227
0919122, -43.141622709191225, -43.51162270919122, -44.23303825691026, -44.2530382569
1026, -44.25303825691026, -44.413038256910255, -44.773746030769786, -45.184453804629
3, -45.214453804629294, -45.61445380462929, -45.6644538046293, -45.80445380462929, -
46.76586935234833, -46.88586935234834, -47.34657712620786, -47.46657712620786, -47.6

```
1657712620786, -47.70657712620786, -48.17728490006737, -48.227284900067374, -48.3972
8490006737, -48.39728490006737, -48.49728490006737, -48.71728490006737, -49.37799267
392688, -49.37799267392688, -49.38799267392688, -49.74799267392688, -49.857992673926
88, -50.44870044778641, -50.458700447786406, -51.039408221645914, -51.0894082216459
2, -51.580115995505444, -51.98082376936496, -52.220823769364955, -52.22082376936495
5, -52.78153154322448, -53.41223931708399, -53.53223931708399, -53.78223931708399, -
54.17294709094352, -54.182947090943514, -54.47294709094352, -54.84365486480303, -54.
91365486480303, -55.28436263866256, -55.904362638662555, -56.02436263866255, -56.445
070412522064, -57.20648596024109, -57.35648596024109, -58.087193734100616, -58.56790
150796013, -58.60790150796013, -59.068609281819654, -59.84002482953869, -59.93002482
953869, -59.98002482953869, -60.300024829538685, -60.300024829538685, -60.8207326033
982, -60.8207326033982, -61.25144037725772, -62.072855924976764, -62.14285592497676
4, -62.26285592497676, -62.87356369883628, -64.3342714726958, -64.6642714726958, -6
5.27497924655532, -65.30497924655532, -65.31497924655531, -66.81568702041483, -68.03
781034199339, -68.85922588971242, -69.94134921129097, -70.27134921129097, -71.853472
53286953, -72.21418030672903, -72.61488808058856, -72.72488808058856, -74.8591347237
4567, -76.94267359304325, -77.40338136690276, -78.5547969146218, -82.22187465321697,
-83.13329020093602, -85.09682907023361, -86.58966016567169, -88.06249126110974, -91.
55886122584542, -92.14956899970491, -93.22956899970492, -97.0366467383001]
```

In [507]:
```
qme=sse/(n-2)
serro=np.sqrt(qme)
print('Desvio Padrão  dos erros:', serro)
```

```
Desvio Padrão  dos erros: 6.021676069856358
```

In [508]:
```
sb0 = serro * math.sqrt(1/n+(xm*xm/(sxx-(n*(xm*xm)))))
print('Desvio Padrão do parâmetro b0', sb0)
```

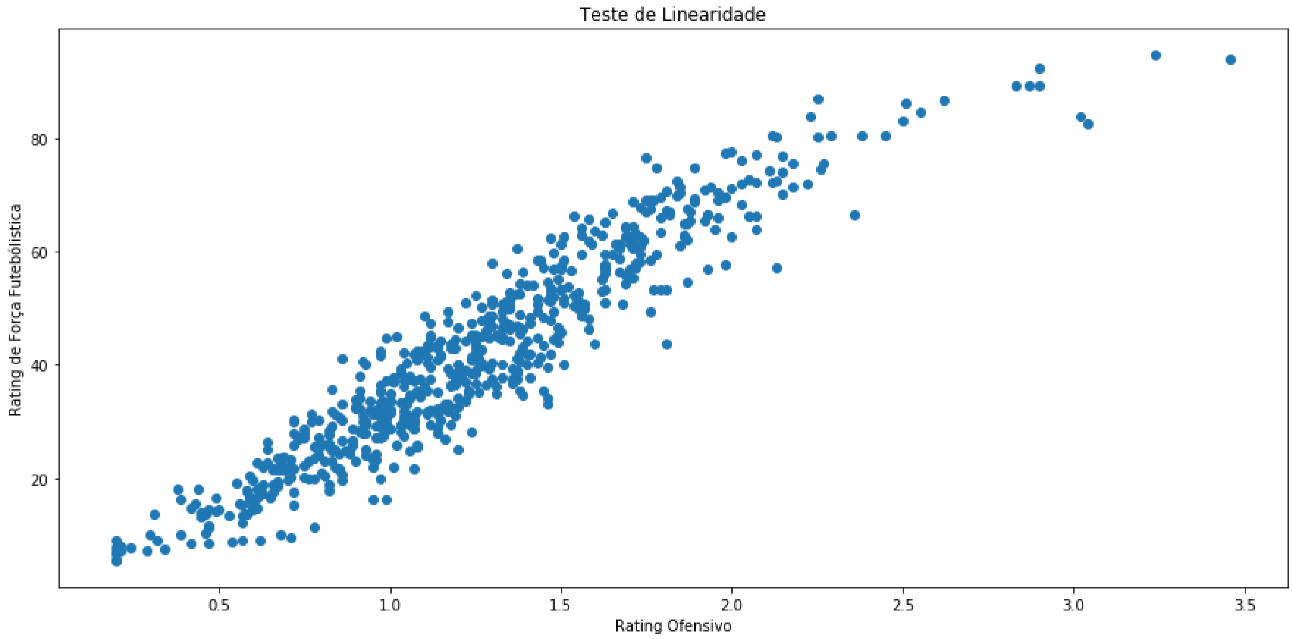
```
Desvio Padrão do parâmetro b0 0.6422639398455074
```

In [509]:
```
sb1 = serro / math.sqrt(sxx-(n*(xm*xm)))
print('Desvio Padrão do parâmetro b1', sb1)
```

```
Desvio Padrão do parâmetro b1 0.4777851505213685
```

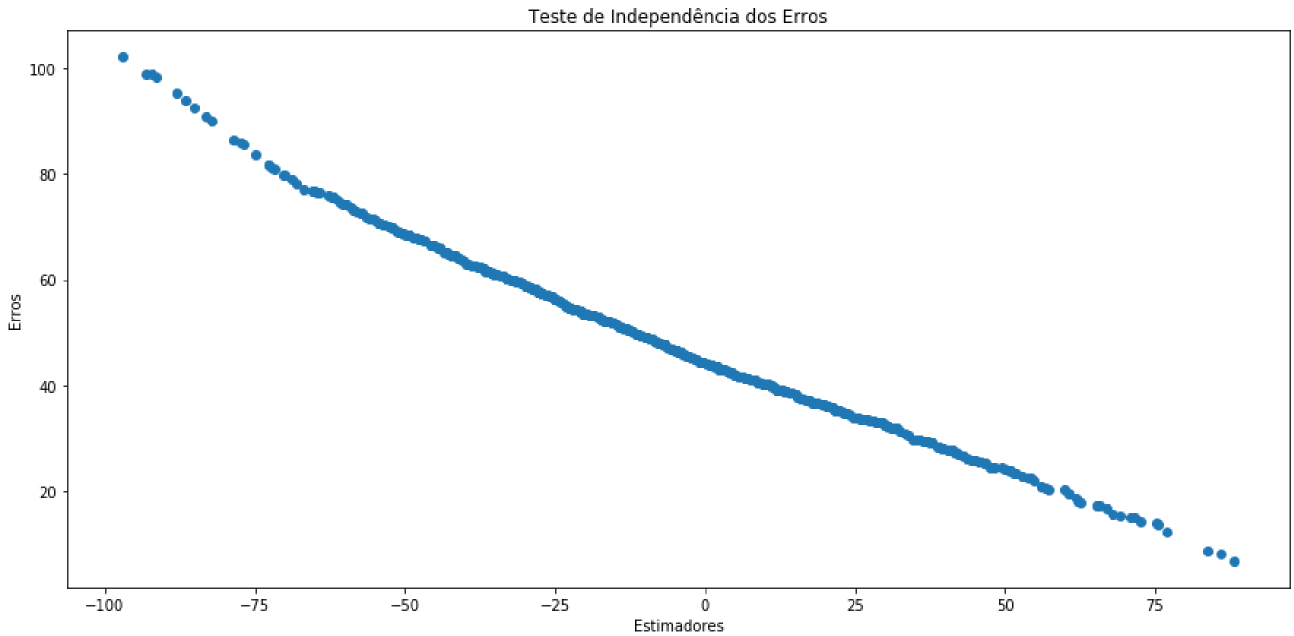
7.2 Testes Visual de Pressuposto

In [518]:
```
plt.scatter(x1,z2)
plt.xlabel("Rating Ofensivo")
plt.ylabel("Rating de Força Futebólistica")
plt.title("Teste de Linearidade")
plt.show()
```



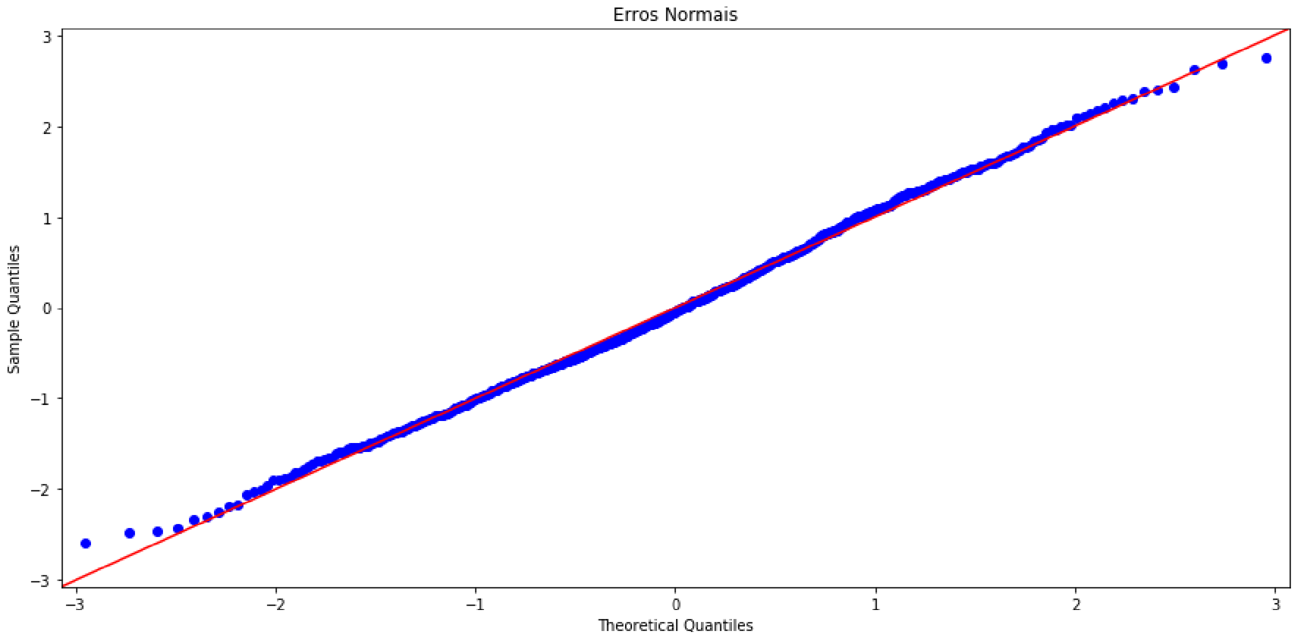
O teste de linearidade é bem válidado se pensarmos em uma reta ajustada aos dados.

In [513]:
```
plt.scatter(ey,est)
plt.xlabel("Estimadores")
plt.ylabel("Erros")
plt.title("Teste de Independência dos Erros")
plt.rcParams['figure.figsize'] = (15,7)
plt.show()
```



O erros aparentam seguir um padrão decrescente mostrando uma tendência visível, evidênciando uma dependência dos resíduos. Indicado que um modelo de regressão não linear sobre a amostra pode apresentar melhores resultados.

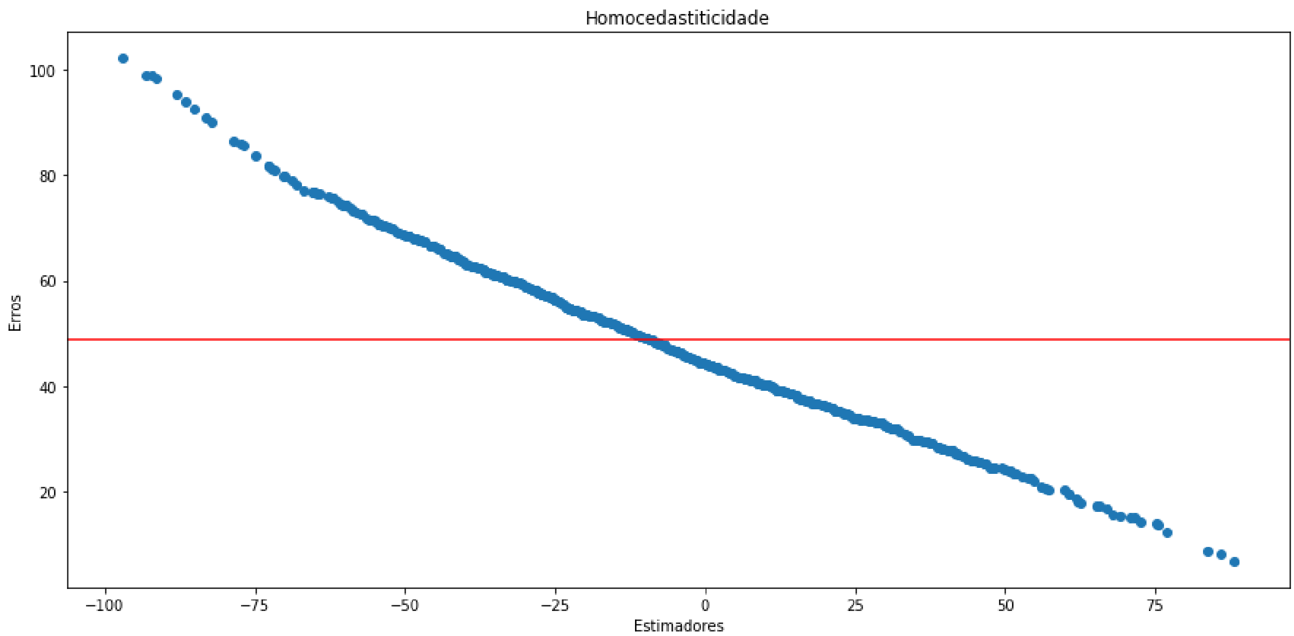
```
In [525]: sm.qqplot(np.array(ey), scs.t, fit=True, line='45')
          plt.title("Erros Normais")

          plt.show()
```



Os erros estão seguindo uma distribuição normal. Então podemos predizer e estimar em nosso modelo.

```
In [527]: mediaerros= np.mean(est)
          print(mediaerros)
          plt.scatter(ey,est)
          plt.axhline(y=mediaerros, color='r', linestyle='-')
          plt.xlabel("Estimadores")
          plt.ylabel("Erros")
          plt.title("Homocedastiticidade")
          plt.show()
```

```
48.74916796939987
```



A distribuição dos dados em torno da média dos resíduos está com uma tendência visivel, temos indícios que a variância dos resíduos não são homogêneas existindo heterocedasticidade. Essa tendência é um bom índicio para uso de regressão não-linear.