Notebook_BaseMV

May 11, 2022

0.0.1 ANÁLISE EXPLORATÓRIA DE DADOS

Trabalho para obtenção de nota no âmbito da disciplina obrigatória de "Estatística Econômica Aplicada" do curso de Mestrado Profissional em Economia – Área de Concentração em Economia – do Programa de Pós-Graduação Profissional em Economia (PPECO) da Faculdade de Ciências Econômicas (FCE) da Universidade Federal do Rio Grande do Sul (UFRGS)

Aluno: Marcus Vinicius Rossetti Guerra

Professor: Fernando Augusto Boeira Sabino da Silva

Abril/2022

0.0.2 1. Descrição da base de dados

A base de dados tem como fonte o Sistema Gerenciador de Séries Temporais (SGS) do Banco Central do Brasil (Bacen) e contém séries temporais relativamente aos seguintes indicadores:

a) SGS 21424 - Índice da Basileia (B1B2) dos chamados Consolidado Bancário I e Consolidado Bancário II (conceito internacional de medida de solvência do sistema financeiro);

b) SGS 13645 - Percentual do total de provisões em relação à carteira de crédito do sistema financeiro (% de provisionamento de créditos de liquidação duvidosa da carteira de crédito do sistema financeiro);

c) SGS 21082 - Inadimplência da carteira de crédito (% da carteira de crédito do sistema financeiro com pelo menos uma parcela com atraso superior a 90 dias);

0.0.3 2. Importação da base de dados ora nomeada "BaseMV"

[60]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('C:/Users/mvrgu/OneDrive/Documentos/Scanned Documents/Mestrado/
 ↪Estatística Econômica Aplicada/BaseMV.csv', sep=';')
df.columns =['Data', 'Índice da Basileia', 'Provisionamento', 'Inadimplência␣
 ↪Total']
display(df)
```

	Data	Índice da Basileia	Provisionamento	Inadimplência Total
0	mar/11	16.99	5.5	3.17
1	abr/11	16.67	5.6	3.24

```
2    mai/11              16.65              5.6                3.37
3    jun/11              16.92              5.6                3.32
4    jul/11              16.34              5.7                3.42
..      …                …                  …                  …
127  out/21              16.29              5.8                2.28
128  nov/21              16.53              5.8                2.32
129  dez/21              16.51              5.8                2.31
130  jan/22              16.11              5.8                2.46
131  fev/22              16.13              5.8                2.52

[132 rows x 4 columns]
```

0.0.4 3. Verificação dos dados da "BaseMV"

3.1 Informações gerais da "BaseMV"

[61]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Data                 132 non-null    object
 1   Índice da Basileia   132 non-null    float64
 2   Provisionamento      132 non-null    float64
 3   Inadimplência Total  132 non-null    float64
dtypes: float64(3), object(1)
memory usage: 4.2+ KB
```

[62]:
```
print('    A "BaseMV" possui', df.shape[0], "linhas e", df.shape[1], "colunas.")
```

```
    A "BaseMV" possui 132 linhas e 4 colunas.
```

[63]:
```
df.values
```

[63]:
```
array([['mar/11', 16.99, 5.5, 3.17],
       ['abr/11', 16.67, 5.6, 3.24],
       ['mai/11', 16.65, 5.6, 3.37],
       ['jun/11', 16.92, 5.6, 3.32],
       ['jul/11', 16.34, 5.7, 3.42],
       ['ago/11', 16.18, 5.7, 3.45],
       ['set/11', 16.08, 5.6, 3.46],
       ['out/11', 16.17, 5.7, 3.58],
       ['nov/11', 16.52, 5.7, 3.58],
       ['dez/11', 16.33, 5.7, 3.54],
       ['jan/12', 16.0, 5.8, 3.66],
       ['fev/12', 16.07, 5.8, 3.71],
       ['mar/12', 16.0, 5.7, 3.67],
```

```
['abr/12', 15.94, 5.7, 3.76],
['mai/12', 16.0, 5.7, 3.76],
['jun/12', 16.43, 5.7, 3.71],
['jul/12', 16.64, 5.7, 3.72],
['ago/12', 16.38, 5.8, 3.75],
['set/12', 16.41, 5.7, 3.72],
['out/12', 16.32, 5.7, 3.76],
['nov/12', 16.4, 5.7, 3.67],
['dez/12', 16.43, 5.5, 3.55],
['jan/13', 16.59, 5.5, 3.54],
['fev/13', 16.51, 5.5, 3.51],
['mar/13', 17.02, 5.4, 3.45],
['abr/13', 17.62, 5.5, 3.47],
['mai/13', 17.21, 5.4, 3.47],
['jun/13', 16.87, 5.2, 3.25],
['jul/13', 17.3, 5.2, 3.21],
['ago/13', 16.98, 5.1, 3.13],
['set/13', 17.06, 5.1, 3.14],
['out/13', 16.76, 5.1, 3.06],
['nov/13', 16.59, 5.1, 2.97],
['dez/13', 16.12, 5.0, 2.86],
['jan/14', 14.96, 5.0, 2.84],
['fev/14', 15.19, 4.9, 2.86],
['mar/14', 15.42, 4.9, 2.88],
['abr/14', 15.45, 4.9, 2.92],
['mai/14', 15.36, 4.8, 3.0],
['jun/14', 15.48, 4.8, 2.9],
['jul/14', 15.81, 4.8, 2.96],
['ago/14', 16.59, 4.9, 2.98],
['set/14', 16.5, 4.8, 2.91],
['out/14', 16.7, 4.9, 2.96],
['nov/14', 16.58, 4.8, 2.85],
['dez/14', 16.67, 4.9, 2.73],
['jan/15', 15.98, 4.9, 2.82],
['fev/15', 15.63, 4.9, 2.85],
['mar/15', 15.53, 4.9, 2.82],
['abr/15', 15.81, 4.9, 2.96],
['mai/15', 15.97, 5.0, 3.02],
['jun/15', 16.33, 5.0, 2.92],
['jul/15', 16.13, 5.1, 3.03],
['ago/15', 15.88, 5.2, 3.12],
['set/15', 15.47, 5.5, 3.12],
['out/15', 15.7, 5.6, 3.24],
['nov/15', 15.83, 5.7, 3.38],
['dez/15', 16.33, 5.7, 3.37],
['jan/16', 15.3, 5.8, 3.47],
['fev/16', 15.58, 6.0, 3.5],
```

```
['mar/16', 16.25, 6.0, 3.53],
['abr/16', 16.19, 6.1, 3.65],
['mai/16', 16.29, 6.2, 3.73],
['jun/16', 16.53, 6.2, 3.51],
['jul/16', 16.27, 6.4, 3.56],
['ago/16', 16.59, 6.4, 3.64],
['set/16', 16.74, 6.5, 3.68],
['out/16', 16.89, 6.6, 3.84],
['nov/16', 17.08, 6.5, 3.8],
['dez/16', 17.15, 6.6, 3.7],
['jan/17', 16.56, 6.8, 3.73],
['fev/17', 16.56, 6.8, 3.75],
['mar/17', 16.73, 6.8, 3.85],
['abr/17', 16.88, 6.8, 3.93],
['mai/17', 16.96, 6.9, 4.04],
['jun/17', 17.35, 6.8, 3.73],
['jul/17', 17.68, 6.6, 3.69],
['ago/17', 17.85, 6.6, 3.7],
['set/17', 17.96, 6.6, 3.6],
['out/17', 17.85, 6.7, 3.63],
['nov/17', 17.93, 6.6, 3.54],
['dez/17', 18.15, 6.5, 3.24],
['jan/18', 17.4, 6.5, 3.42],
['fev/18', 17.27, 6.5, 3.43],
['mar/18', 17.25, 6.4, 3.29],
['abr/18', 17.41, 6.4, 3.32],
['mai/18', 17.07, 6.4, 3.3],
['jun/18', 17.17, 6.3, 3.04],
['jul/18', 17.58, 6.4, 3.02],
['ago/18', 16.96, 6.4, 3.04],
['set/18', 17.66, 6.4, 3.04],
['out/18', 18.07, 6.5, 3.05],
['nov/18', 18.08, 6.4, 2.96],
['dez/18', 17.95, 6.4, 2.87],
['jan/19', 17.84, 6.4, 2.95],
['fev/19', 17.71, 6.3, 2.91],
['mar/19', 17.82, 6.3, 2.99],
['abr/19', 17.97, 6.3, 3.02],
['mai/19', 18.08, 6.3, 3.05],
['jun/19', 18.02, 6.1, 2.95],
['jul/19', 17.93, 6.1, 3.06],
['ago/19', 17.72, 6.2, 3.04],
['set/19', 17.74, 6.1, 3.06],
['out/19', 18.01, 6.1, 3.03],
['nov/19', 17.84, 6.2, 3.01],
['dez/19', 17.12, 6.4, 2.94],
['jan/20', 16.7, 6.4, 3.0],
```

```
['fev/20', 16.17, 6.4, 3.04],
['mar/20', 15.63, 6.6, 3.18],
['abr/20', 15.73, 6.7, 3.3],
['mai/20', 15.93, 6.8, 3.25],
['jun/20', 16.32, 6.9, 2.9],
['jul/20', 16.59, 6.8, 2.77],
['ago/20', 16.53, 6.7, 2.66],
['set/20', 16.66, 6.5, 2.44],
['out/20', 16.48, 6.5, 2.36],
['nov/20', 16.78, 6.6, 2.24],
['dez/20', 16.85, 6.4, 2.12],
['jan/21', 16.49, 6.4, 2.15],
['fev/21', 16.36, 6.3, 2.24],
['mar/21', 16.45, 6.2, 2.14],
['abr/21', 16.63, 6.2, 2.2],
['mai/21', 16.75, 6.1, 2.34],
['jun/21', 16.91, 6.1, 2.28],
['jul/21', 16.78, 6.1, 2.32],
['ago/21', 16.65, 6.0, 2.32],
['set/21', 16.76, 5.8, 2.29],
['out/21', 16.29, 5.8, 2.28],
['nov/21', 16.53, 5.8, 2.32],
['dez/21', 16.51, 5.8, 2.31],
['jan/22', 16.11, 5.8, 2.46],
['fev/22', 16.13, 5.8, 2.52]], dtype=object)
```

A "BaseMV é uma matriz de valores configurados em um Painel.

[64]:
```
df.columns
```

[64]:
```
Index(['Data', 'Índice da Basileia', 'Provisionamento', 'Inadimplência Total'],
dtype='object')
```

Os títulos das 4 (quatro) colunas da "BaseMV" estão retroevidenciados.

3.2 Averiguação da presença de dados nulos na "BaseMV"

[65]:
```
nulls = pd.DataFrame(df.isna().sum()/len(df))
nulls= nulls.reset_index()
nulls.columns = ['Nome da Coluna', 'Percentagem de Nulos']
nulls.sort_values(by='Percentagem de Nulos', ascending = False)
```

[65]:

	Nome da Coluna	Percentagem de Nulos
0	Data	0.0
1	Índice da Basileia	0.0
2	Provisionamento	0.0
3	Inadimplência Total	0.0

A "BaseMV" não contém quaisquer dados nulos.

0.0.5 4. Manipulação dos dados da "BaseMV"

Não é necessária qualquer manipulação dos dados da "BaseMV".

0.0.6 5. Análise exploratória dos dados da "BaseMV"

A análise exploratória de dados é usada para examinar e investigar conjuntos de dados e resumir suas principais características, antes de quaisquer suposições, muitas vezes usando métodos de visualização de dados e estatística descritiva. Ela permite descobrir padrões, detectar anomalias, além de encontrar relações interessantes entre as variáveis. Uma vez completa, seus resultados podem então ser usados para modelagens ou análises de dados mais sofisticadas.

5.1. Descrição dos dados

[66]:
```
df.describe().applymap(lambda x: f"{x:0.2f}")
```

[66]:

	Índice da Basileia	Provisionamento	Inadimplência Total
count	132.00	132.00	132.00
mean	16.67	5.91	3.15
std	0.74	0.61	0.46
min	14.96	4.80	2.12
25%	16.18	5.50	2.91
50%	16.59	6.00	3.12
75%	17.09	6.40	3.54
max	18.15	6.90	4.04

Legenda:

count = contagem dos registros não nulos de cada uma das séries de dados

mean = valor médio de cada uma das séries de dados

std = desvio padrão de cada uma das séries de dados

min = valor mínimo de cada uma das séries de dados

25% = primeiro quartil ou o valor que separa os 25% menores valores dos 75% maiores valores de cada uma das séries de dados

50% = segundo quartil (que é a própria mediana) ou o valor que separa os 50% menores dos 50% maiores valores de cada uma das séries de dados

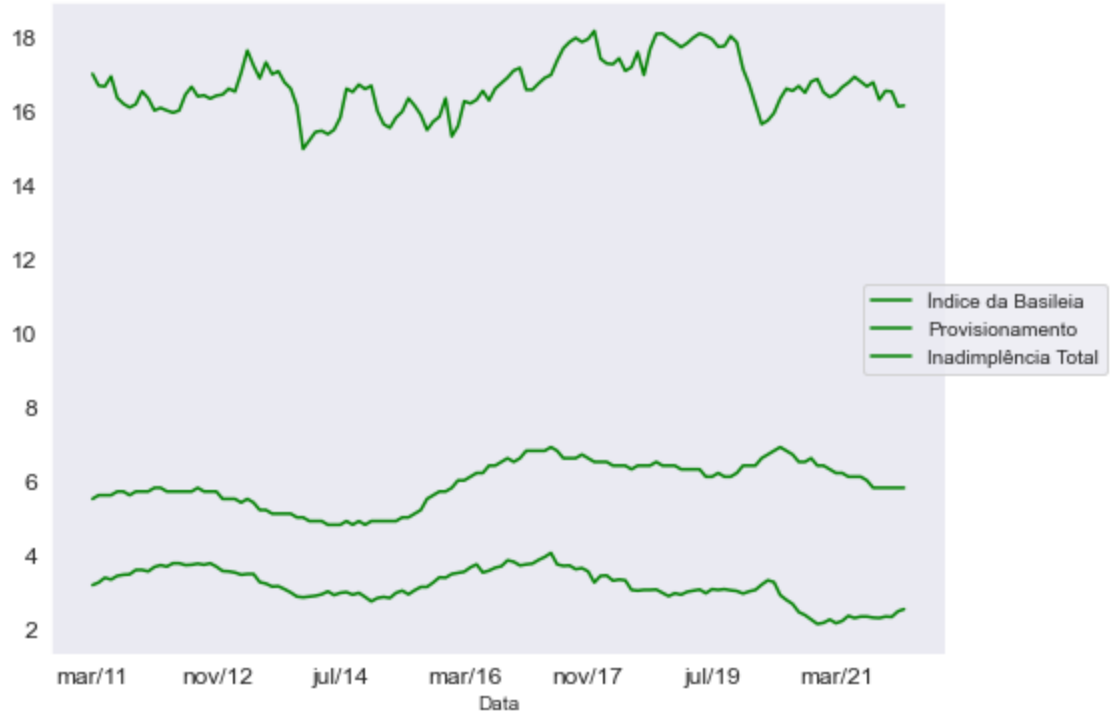
75% = terceiro quartil ou o valor que separa os 75% menores valores dos 25% maiores valores de cada uma das séries de dados

max = valor máximo de cada uma das séries de dados

5.2. Análise dos dados no tempo

- Gráfico de linhas (line plot) com todas as séries

[67]:
```
from pylab import rcParams
import matplotlib.pyplot as plt
import seaborn as sns

df.plot(x='Data', kind='line', fontsize=12)

plt.legend(loc='right',bbox_to_anchor = (0.05,0,1,1), bbox_transform = plt.
 ↪gcf().transFigure )

plt.grid()

rcParams['figure.figsize'] = 8, 4

plt.show()
```



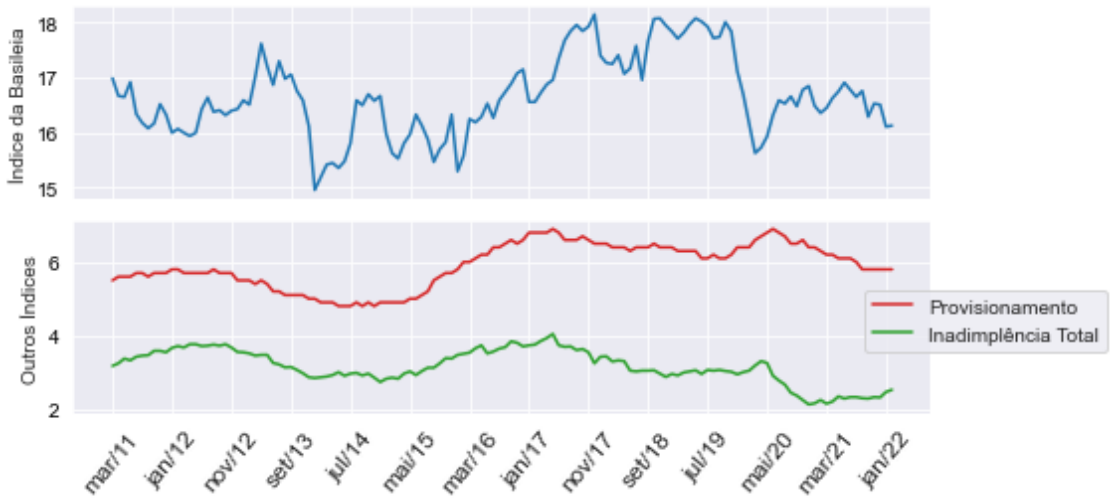
A escala não favorece a visualização.

Por isso, a seguir, as séries são apresentadas em gráficos de linhas distintos.

- Gráfico de linhas (multi line plot) com todas as séries

[68]:
```
# Parâmetros
x = df['Data']
y1 = df['Índice da Basileia']
y2 = df['Provisionamento']
```

```
y3 = df['Inadimplência Total']

# Gerais
fig, (ax1, ax2) = plt.subplots(2, 1,sharex=True)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', which='both', top=False, bottom=False,␣
 ↪left=False, right=False)
plt.grid()

# Gráfico 1
ax1.plot(x, y1, color='tab:blue')
ax1.set_ylabel('Índice da Basileia', fontsize=10)
ax1.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax1.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax1.tick_params(axis='x', rotation=0, labelsize=12)
ax1.tick_params(axis='y', rotation=0, labelcolor='black' )

# Gráfico 2
ax2.plot(x, y2, color='tab:red',label='Provisionamento')
ax2.plot(x, y3, color='tab:green',label='Inadimplência Total')
ax2.set_ylabel('Outros Índices', fontsize=10)
ax2.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax2.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax2.tick_params(axis='x', rotation=50, labelsize=12)
ax2.tick_params(axis='y', rotation=0, labelsize=10, labelcolor='black' )
ax2.legend( loc = 'center right', bbox_to_anchor = (0.02,-0.1,1,1),
            bbox_transform = plt.gcf().transFigure )

fig.tight_layout()

rcParams['figure.figsize'] = 8, 4

plt.show()
```



A escala do gráfico continua não favorecendo a visualização.

Por isso, na sequência, as séries são apresentadas duas a duas, em escala logarítimica, em gráficos de linhas distintos.

- Gráfico de linhas (multi line plot) com "Índice da Basileia" e "Provisionamento" Em escala logarítmica para facilitar a visualização.

[69]:
```
um_df = np.log(df[['Índice da Basileia', 'Provisionamento']])
um_df['Data'] = df['Data']

# Parâmetros
x = um_df['Data']
y1 = um_df['Índice da Basileia']
y2 = um_df['Provisionamento']

# Gerais
fig, (ax1, ax2) = plt.subplots(2, 1,sharex=True)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', which='both', top=False, bottom=False,␣
 ↪left=False, right=False)
plt.grid()

# Gráfico 1
ax1.plot(x, y1, color='tab:blue')
ax1.set_ylabel('Índice da Basileia', fontsize=10)
ax1.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax1.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax1.tick_params(axis='x', rotation=0, labelsize=12)
```

```
ax1.tick_params(axis='y', rotation=0, labelcolor='black' )

# Gráfico 2
ax2.plot(x, y2, color='tab:red')
ax2.set_ylabel('Provisionamento', fontsize=10)
ax2.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax2.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas
 ↪Meses
ax2.tick_params(axis='x', rotation=50, labelsize=12)
ax2.tick_params(axis='y', rotation=0, labelsize=10, labelcolor='black' )

fig.tight_layout()

rcParams['figure.figsize'] = 8, 4

plt.show()
```

É possível visualizar correlação positiva entre o "Índice da Basileia" e o "Provisionamento".

- Gráfico de linhas (multi line plot) com "Índice da Basileia" e "Inadimplência Total"
Em escala logarítmica para facilitar a visualização.

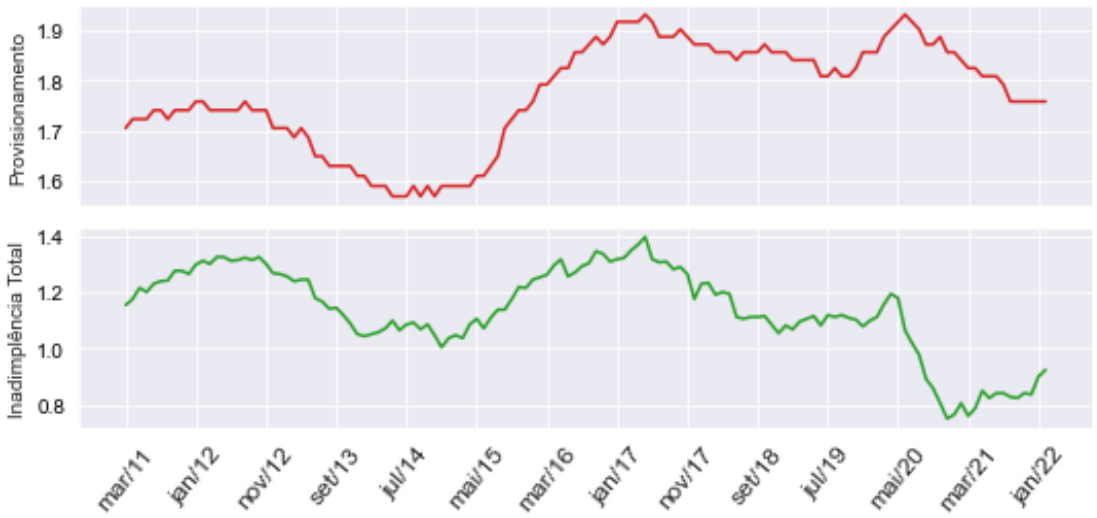
[70]:
```
dois_df = np.log(df[['Índice da Basileia', 'Inadimplência Total']])
dois_df['Data'] = df['Data']

# Parâmetros
x = dois_df['Data']
y1 = dois_df['Índice da Basileia']
y2 = dois_df['Inadimplência Total']
```

```
# Gerais
fig, (ax1, ax2) = plt.subplots(2, 1,sharex=True)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', which='both', top=False, bottom=False,␣
 ↪left=False, right=False)
plt.grid()

# Gráfico 1
ax1.plot(x, y1, color='tab:blue')
ax1.set_ylabel('Índice da Basileia', fontsize=10)
ax1.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax1.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax1.tick_params(axis='x', rotation=0, labelsize=12)
ax1.tick_params(axis='y', rotation=0, labelcolor='black' )

# Gráfico 2
ax2.plot(x, y2, color='tab:green')
ax2.set_ylabel('Inadimplência Total', fontsize=10)
ax2.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax2.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax2.tick_params(axis='x', rotation=50, labelsize=12)
ax2.tick_params(axis='y', rotation=0, labelsize=10, labelcolor='black' )

fig.tight_layout()

rcParams['figure.figsize'] = 8, 4

plt.show()
```

É possível visualizar correlação positiva entre o "Índice da Basileia" e o "Provisionamento", pelo menos, até 2017.

- Gráfico de linhas (multi line plot) com "Provisionamento" e "Inadimplência Total"
Em escala logarítmica para facilitar a visualização.

[71]:
```
tres_df = np.log(df[['Provisionamento', 'Inadimplência Total']])
tres_df['Data'] = df['Data']

# Parâmetros
x = tres_df['Data']
y1 = tres_df['Provisionamento']
y2 = tres_df['Inadimplência Total']

# Gerais
fig, (ax1, ax2) = plt.subplots(2, 1,sharex=True)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', which='both', top=False, bottom=False,␣
 ↪left=False, right=False)
plt.grid()

# Gráfico 1
ax1.plot(x, y1, color='tab:red')
ax1.set_ylabel('Provisionamento', fontsize=10)
ax1.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax1.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax1.tick_params(axis='x', rotation=0, labelsize=12)
ax1.tick_params(axis='y', rotation=0, labelcolor='black' )

# Gráfico 2
ax2.plot(x, y2, color='tab:green')
ax2.set_ylabel('Inadimplência Total', fontsize=10)
ax2.set_xticks(np.arange(0, len(x), 10)) # Legendas Meses
ax2.set_xticklabels(x[::10], rotation=0, fontdict={'fontsize':10}) # Legendas␣
 ↪Meses
ax2.tick_params(axis='x', rotation=50, labelsize=12)
ax2.tick_params(axis='y', rotation=0, labelsize=10, labelcolor='black' )

fig.tight_layout()

rcParams['figure.figsize'] = 8, 4

plt.show()
```

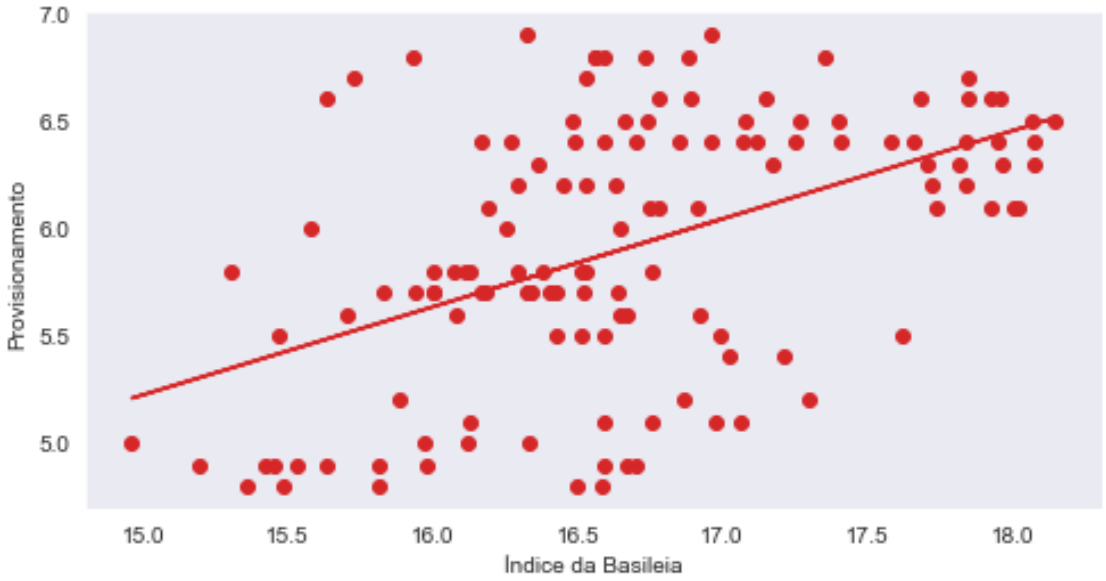


É possível visualizar correlação positiva entre o "Provisionamento" e a "Inadimplência Total", pelo menos, até 2017.

5.3. Análise da dispersão e correlação dos dados

- Gráfico de dispersão (scatter plot) com "Índice da Basileia" e "Provisionamento"

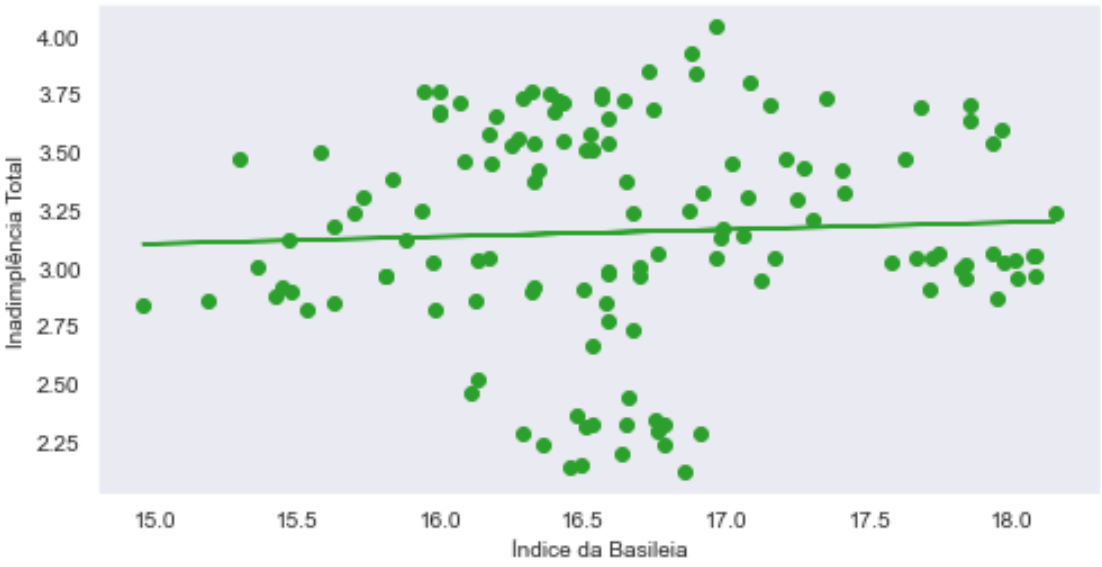
[72]:
```
fig,ax=plt.subplots()

ax.scatter(df['Índice da Basileia'], df['Provisionamento'], marker="o", label␣
 ↪='Provisionamento', color = 'tab:red')
m, b = np.polyfit(df['Índice da Basileia'], df['Provisionamento'], 1)
plt.plot(df['Índice da Basileia'], m*df['Índice da Basileia'] + b, color = 'tab:
 ↪red')

plt.xlabel("Índice da Basileia")
plt.ylabel("Provisionamento")

plt.grid()

rcParams['figure.figsize'] = 8, 4

plt.show()
```



É possível visualizar correlação linear positiva entre o "Índice da Basileia" e o "Provisionamento".

- Gráfico de dispersão (scatter plot) com "Índice da Basileia" e "Inadimplência Total"

[73]:
```
fig,ax=plt.subplots()

ax.scatter(df['Índice da Basileia'], df['Inadimplência Total'], marker="o",␣
 ↪label = 'Inadimplência Total', color = 'tab:green')
m, b = np.polyfit(df['Índice da Basileia'], df['Inadimplência Total'], 1)
plt.plot(df['Índice da Basileia'], m*df['Índice da Basileia'] + b,color = 'tab:
 ↪green')

plt.xlabel("Índice da Basileia")
plt.ylabel("Inadimplência Total")

plt.grid()

rcParams['figure.figsize'] = 8, 4

plt.show()
```

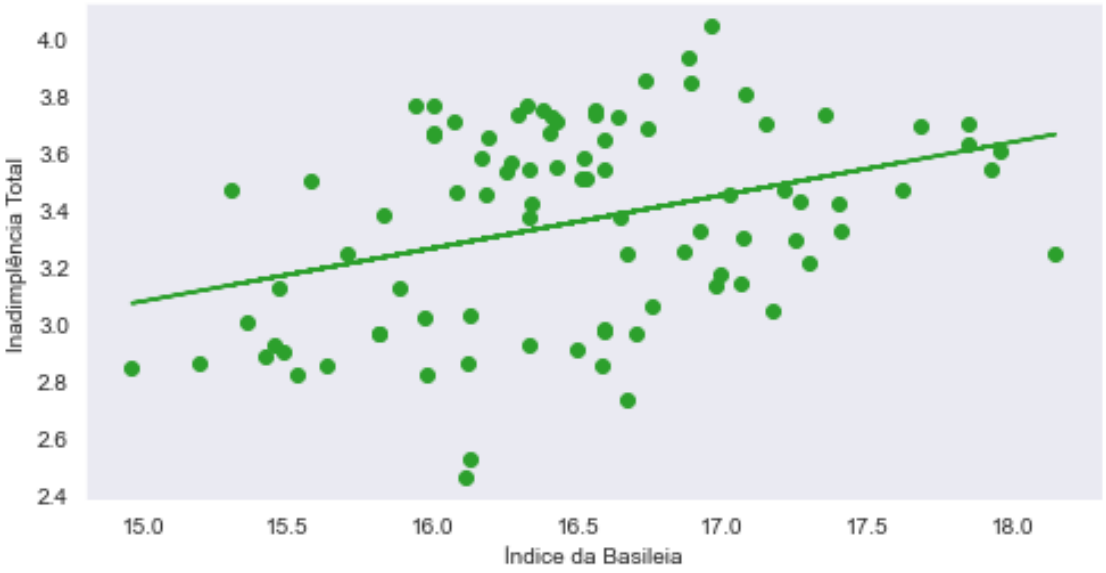


Não é possível visualizar correlação entre o "Índice da Basileia" e a "Inadimplência Total".

Como, da visualização dos gráficos de linha anteriores, havia sugestão de correlação até 2017, a seguir, a apresentação do mesmo gráfico de dispersão com as duas séries até dez/2017.

- Gráfico de dispersão (scatter plot) com "Índice da Basileia" e "Inadimplência Total"
Com as séries até Dez/2017 para confirmar a correlação sugerida anteriormente.

[74]:
```
# Séries até Dez/2017
quatro_df = df.drop(labels=range(88, 130), axis=0)

fig,ax=plt.subplots()

ax.scatter(quatro_df['Índice da Basileia'], quatro_df['Inadimplência Total'],␣
 ↪marker="o", label = 'Inadimplência Total', color = 'tab:green')
m, b = np.polyfit(quatro_df['Índice da Basileia'], quatro_df['Inadimplência␣
 ↪Total'], 1)
plt.plot(quatro_df['Índice da Basileia'], m*quatro_df['Índice da Basileia'] +␣
 ↪b, color = 'tab:green')

plt.xlabel("Índice da Basileia")
plt.ylabel("Inadimplência Total")

plt.grid()

rcParams['figure.figsize'] = 8, 4

plt.show()
```

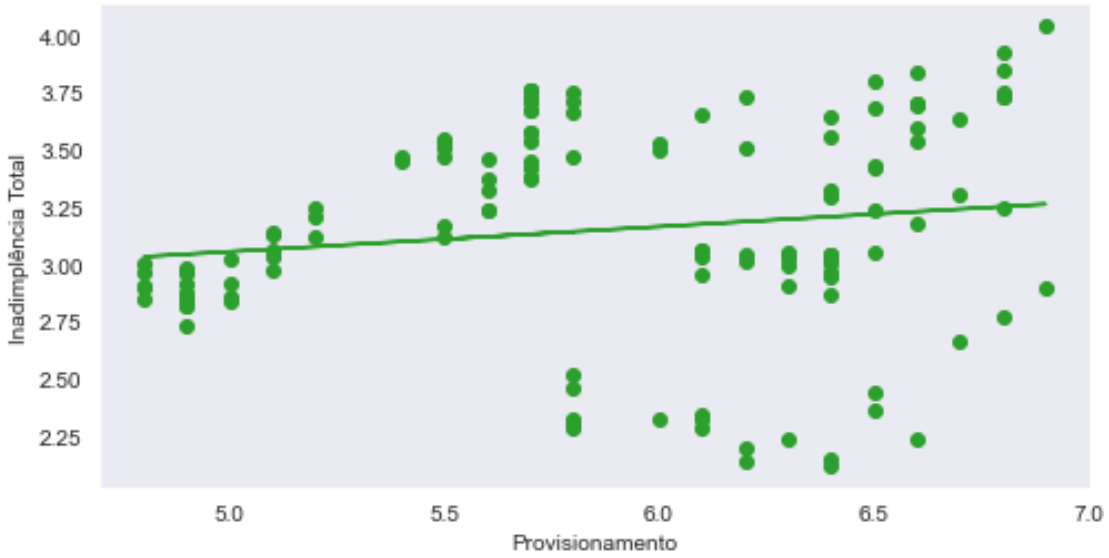


Assim, de fato, até dez/2017, é possível visualizar correlação linear positiva entre o "Índice da Basileia" e a "Inadimplência Total".

Depois, porém, não é mais possível visualizar essa ocorrência - fato passível de sustentar uma eventual inflexão entre os dois indicadores.

- Gráfico de dispersão (scatter plot) com "Provisionamento" e "Inadimplência Total"

[75]:
```
fig,ax=plt.subplots()

ax.scatter(df['Provisionamento'], df['Inadimplência Total'], marker="o", label␣
 ↪= 'Inadimplência Total', color = 'tab:green')
m, b = np.polyfit(df['Provisionamento'], df['Inadimplência Total'], 1)
plt.plot(df['Provisionamento'], m*df['Provisionamento'] + b, color = 'tab:
 ↪green')

plt.xlabel("Provisionamento")
plt.ylabel("Inadimplência Total")

plt.grid()

rcParams['figure.figsize'] = 8, 4

plt.show()
```

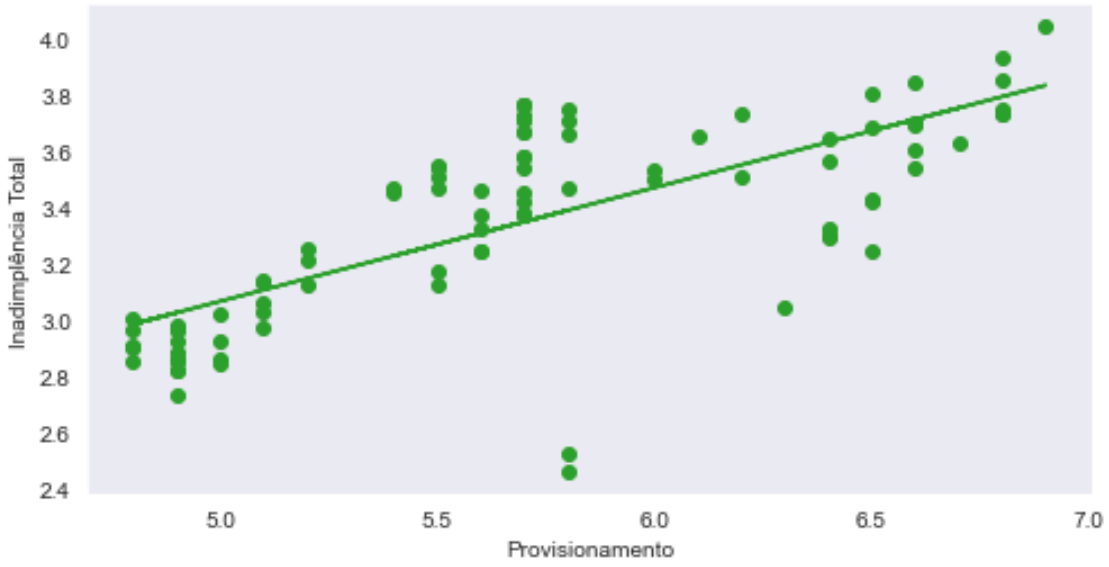


Não é possível visualizar correlação entre o "Provisionamento" e a "Inadimplência Total".

Como, da visualização dos gráficos de linha anteriores, havia sugestão de correlação até 2017, a seguir, a apresentação do mesmo gráfico de dispersão com as duas séries até dez/2017.

- Gráfico de dispersão (scatter plot) com "Provisionamento" e "Inadimplência Total"
Com as séries até Dez/2017 para confirmar a correlação sugerida anteriormente.

[76]:
```
# Séries até Dez/2017
quatro_df = df.drop(labels=range(88, 130), axis=0)

fig,ax=plt.subplots()

ax.scatter(quatro_df['Provisionamento'], quatro_df['Inadimplência Total'],␣
 ↪marker="o", label = 'Inadimplência Total', color = 'tab:green')
m, b = np.polyfit(quatro_df['Provisionamento'], quatro_df['Inadimplência␣
 ↪Total'], 1)
plt.plot(quatro_df['Provisionamento'], m*quatro_df['Provisionamento'] + b,␣
 ↪color = 'tab:green')

plt.xlabel("Provisionamento")
plt.ylabel("Inadimplência Total")

plt.grid()

rcParams['figure.figsize'] = 8, 4

plt.show()
```



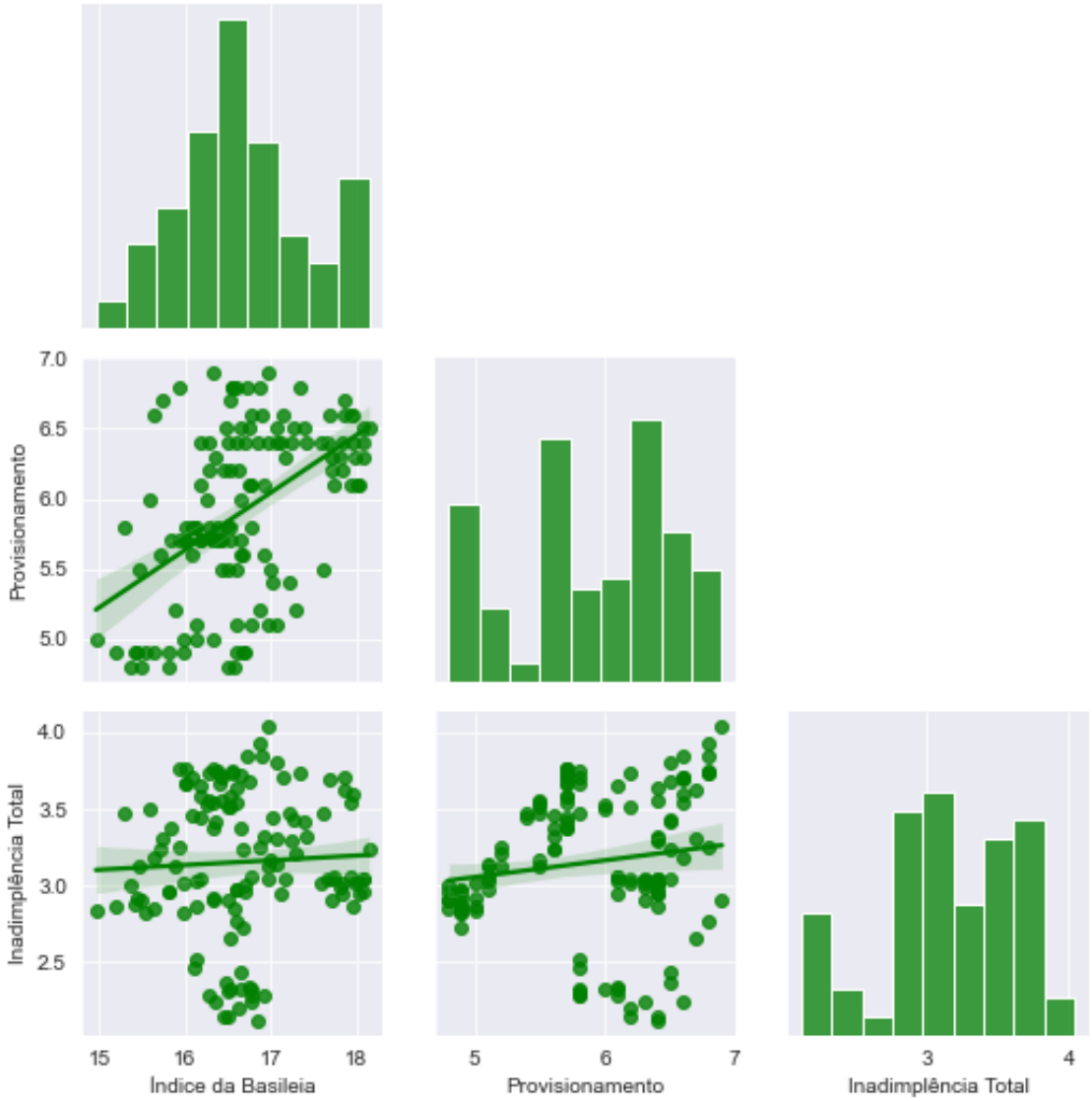
Assim, de fato, até dez/2017, é possível visualizar correlação linear positiva entre o "Provisionamento" e a "Inadimplência Total".

Depois, porém, não é mais possível visualizar essa ocorrência - fato passível de sustentar uma eventual inflexão entre os dois indicadores.

- Matriz com os gráficos de dispersão (scatter plot matrix) para todas as séries Para resumir em uma figura os gráficos de dispersão isolados apresentados anteriormente.

[77]:
```
g = sns.PairGrid(df, corner=True, height=2.5)
g.map_diag(sns.histplot)
g.map_offdiag(sns.regplot)
```

[77]:
```
<seaborn.axisgrid.PairGrid at 0x23ead429940>
```

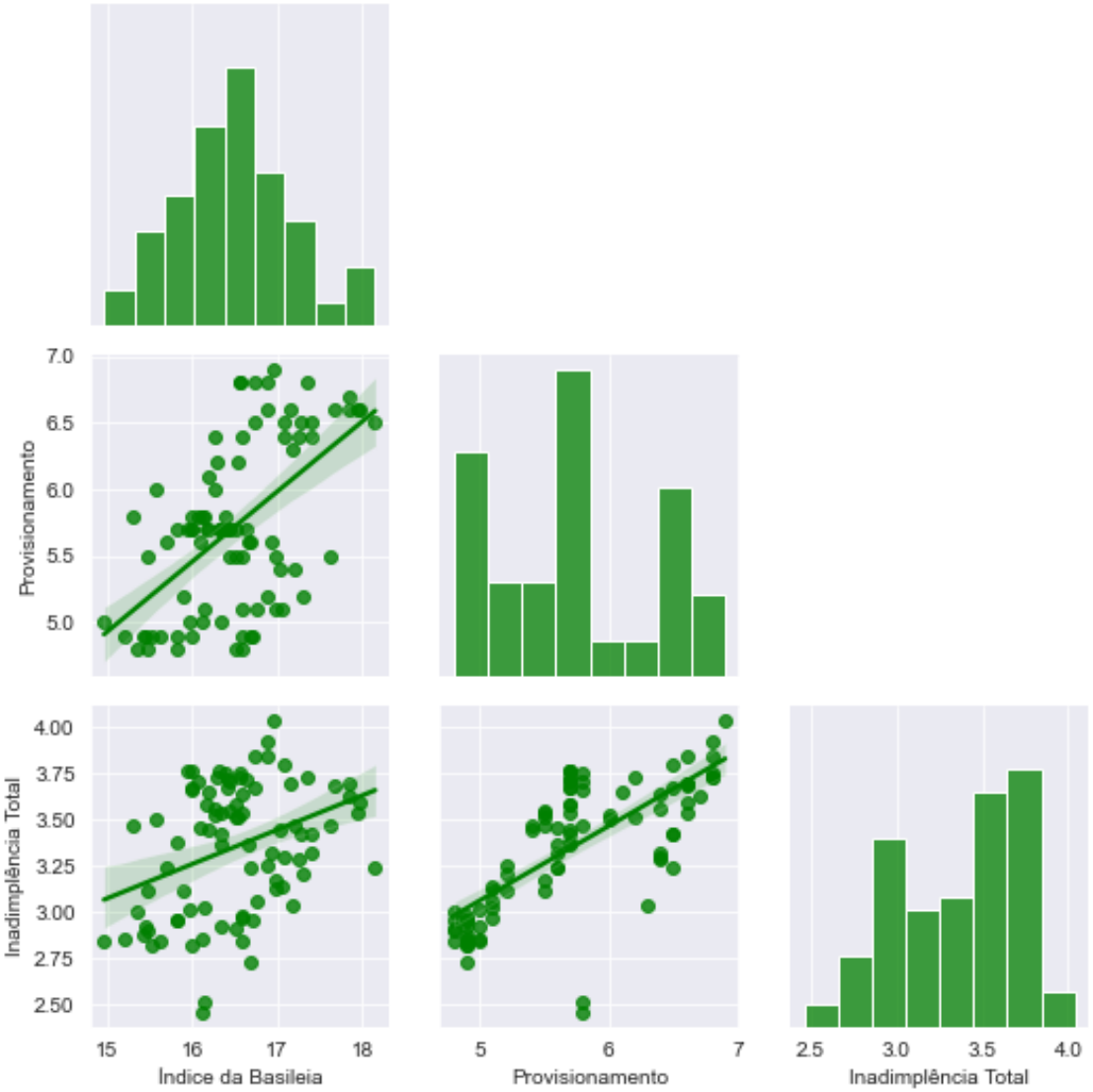


Reafirmando as sugestões anteriores.

- Matriz com os gráficos de dispersão (scatter plot matrix) para todas as séries Com as séries até Dez/2017 para confirmar a correlação retrosugerida e resumir em uma figura os gráficos de dispersão isolados apresentados anteriormente.

[78]:
```
g = sns.PairGrid(quatro_df, corner=True, height=2.5)
g.map_diag(sns.histplot)
g.map_offdiag(sns.regplot)
```

[78]:
```
<seaborn.axisgrid.PairGrid at 0x23ead389ee0>
```



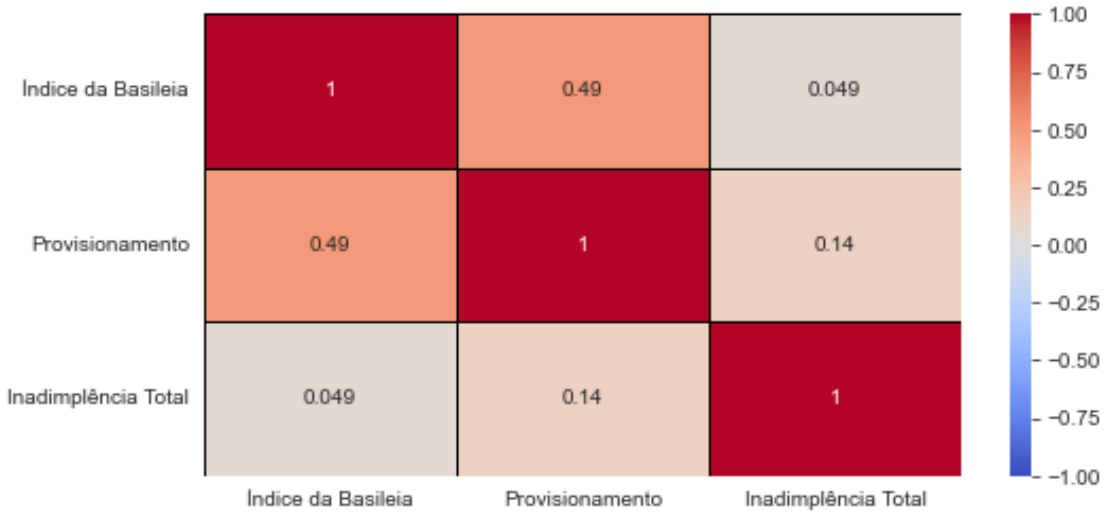
Reafirmando as sugestões anteriores.

- Matriz de correlação (correlation matrix) e mapa de calor (heat map) do coeficiente de correlação produto-momento (Pearson) para todas as séries

[79]:
```
df_corr = df.corr(method='pearson')
display(df_corr)
sns.heatmap(df.corr(), annot=True, mask=False, vmin=-1, vmax=1, center= 0,␣
↪cmap= 'coolwarm', linewidths=1, linecolor="black")
```

	Índice da Basileia	Provisionamento	Inadimplência Total
Índice da Basileia	1.000000	0.493856	0.048902
Provisionamento	0.493856	1.000000	0.143570
Inadimplência Total	0.048902	0.143570	1.000000

[79]: <AxesSubplot:>



Fica reforçada a sugestão anterior de correlação entre "Índice da Basileia" e "Provisionamento". O coeficiente de correlação produto-momento é 0.49 - indicando baixa a moderada correlação linear positiva.

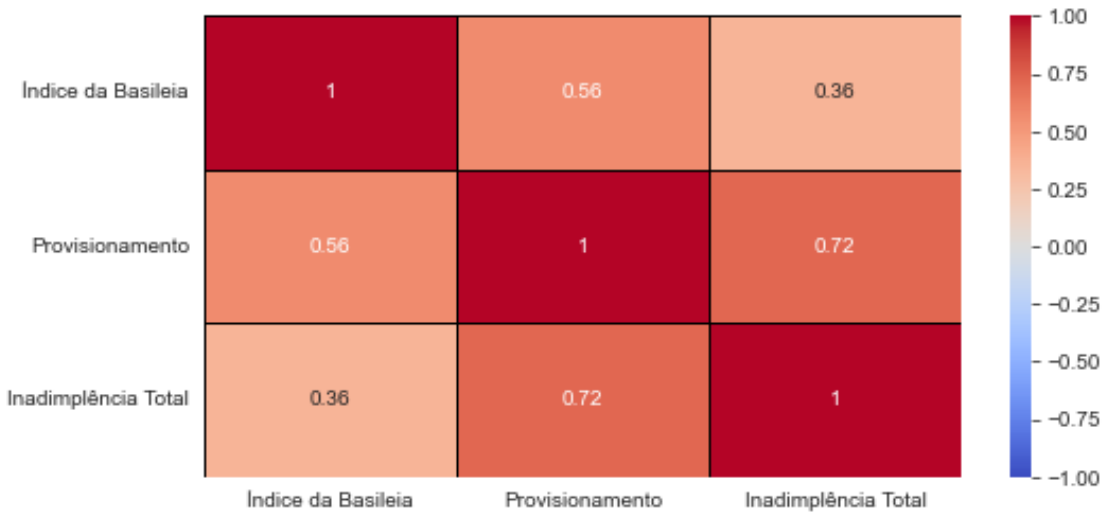
Fica reforçada a sugestão anterior de pouca correlação entre "Índice da Basileia" e "Inadimplência Total (o coeficiente de correlação produto-momento é 0.049) e entre"Provisionamento" e "Inadimplência Total (o coeficiente de correlação produto-momento é 0.14). Como, para esses casos, da visualização dos gráficos de linha e dispersão anteriores, havia sugestão de correlação até 2017, a seguir, a apresentação do mesmo mapa de calor para todas as séries até dez/2017.

- Matriz de correlação (correlation matrix) e mapa de calor (heat map) do coeficiente de correlação produto-momento (Pearson) para todas as séries Com as séries até Dez/2017 para confirmar a correlação sugerida anteriormente.

[80]:
```
quatro_df_corr = quatro_df.corr(method='pearson')
display(quatro_df_corr)
sns.heatmap(quatro_df.corr(), annot=True, mask=False, vmin=-1, vmax=1, center=␣
↪0, cmap= 'coolwarm', linewidths=1, linecolor="black")
```

	Índice da Basileia	Provisionamento	Inadimplência Total
Índice da Basileia	1.000000	0.564637	0.356915
Provisionamento	0.564637	1.000000	0.724307
Inadimplência Total	0.356915	0.724307	1.000000

[80]: <AxesSubplot:>



Fica, mais uma vez, reforçada a sugestão anterior de correlação entre "Índice da Basileia" e "Provisionamento". O coeficiente de correlação produto-momento é 0.56 - indicando moderada correlação linear positiva.

Fica, além disso, reforçada a sugestão anterior de correlação (até dez/2017) entre "Índice da Basileia" e "Inadimplência Total (o coeficiente de correlação produto-momento é 0.36) - no caso, indicação de baixa correlação linear positiva. Depois, porém, não é mais possível visualizar essa ocorrência - fato passível de sustentar uma eventual inflexão entre os dois indicadores.

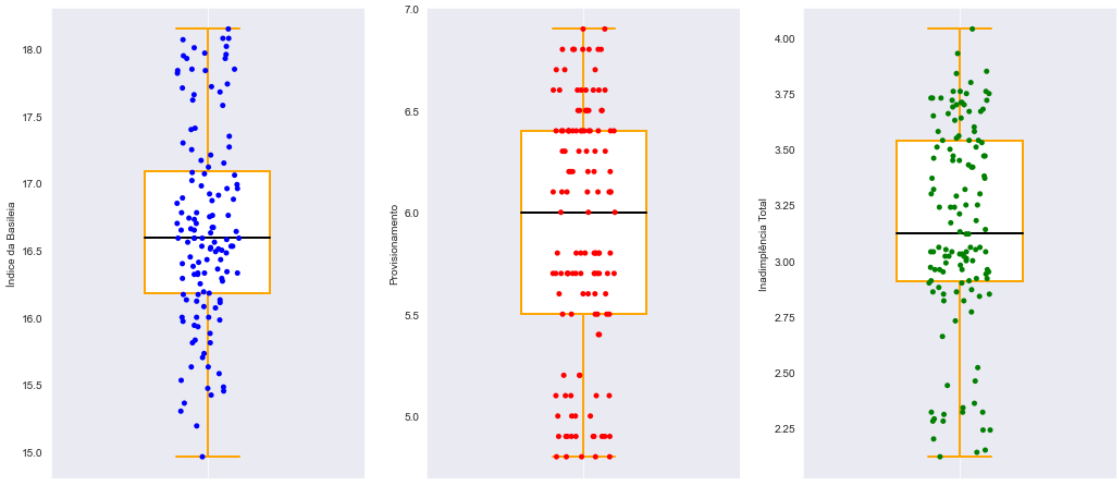
Fica, ainda, reforçada a sugestão anterior de correlação (até dez/2017) entre "Provisionamento" e "Inadimplência Total (o coeficiente de correlação produto-momento é 0.72) - no caso, indicação de alta correlação linear positiva. Depois, porém, não é mais possível visualizar essa ocorrência - o que também é passível de sustentar uma eventual inflexão entre os dois indicadores.

5.4. Análise da simetria e do achatamento dos dados

- Diagrama de caixa (box plot) com todas as séries

[81]:
```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(4,6))

ax1 = fig.add_axes([0, 0, 1, 1])
ax1 = sns.boxplot(y=df['Índice da Basileia'], whis=np.inf, width=0.4,␣
↪capprops=dict(color="orange", linewidth=2),␣
↪whiskerprops=dict(color="orange", linewidth=2), boxprops=dict(color=␣
↪"orange", facecolor="white", linewidth=2), medianprops=dict(color="black",␣
↪linewidth=2), flierprops=dict(marker='o', color="black", linewidth=5, alpha=.
↪5))
ax1 = sns.stripplot(y=df['Índice da Basileia'], color="blue")
plt.grid()

ax2 = fig.add_axes([1.2, 0, 1, 1])
```

```
ax2 = sns.boxplot(y=df['Provisionamento'], whis=np.inf, width=0.4,␣
 ↪cappropps=dict(color="orange", linewidth=2),␣
 ↪whiskerprops=dict(color="orange", linewidth=2), boxprops=dict(color=␣
 ↪"orange", facecolor="white", linewidth=2), medianprops=dict(color="black",␣
 ↪linewidth=2), flierprops=dict(marker='o', color="black", linewidth=5, alpha=.
 ↪5))
ax2 = sns.stripplot(y=df['Provisionamento'], color="red")
plt.grid()

ax3 = fig.add_axes([2.4, 0, 1, 1])
ax3 = sns.boxplot(y=df['Inadimplência Total'], whis=np.inf, width=0.4,␣
 ↪cappropps=dict(color="orange", linewidth=2),␣
 ↪whiskerprops=dict(color="orange", linewidth=2), boxprops=dict(color=␣
 ↪"orange", facecolor="white", linewidth=2), medianprops=dict(color="black",␣
 ↪linewidth=2), flierprops=dict(marker='o', color="black", linewidth=5, alpha=.
 ↪5))
ax3 = sns.stripplot(y=df['Inadimplência Total'], color="green")
plt.grid()

plt.show()
```



Na série "Provisionamento" existe forte indicação de assimetria. O segundo quartil (Q2) menos o primeiro quartil (Q1) é maior que o terceiro quartil (Q3) menos o segundo quartil (Q2). No mesmo sentido, o primeiro quartil (Q1) menos o valor mínimo é maior que o valor máximo menos o terceiro quartil (Q3). Logo, existe sugestão de assimetria para esquerda (enviesada negativamente).

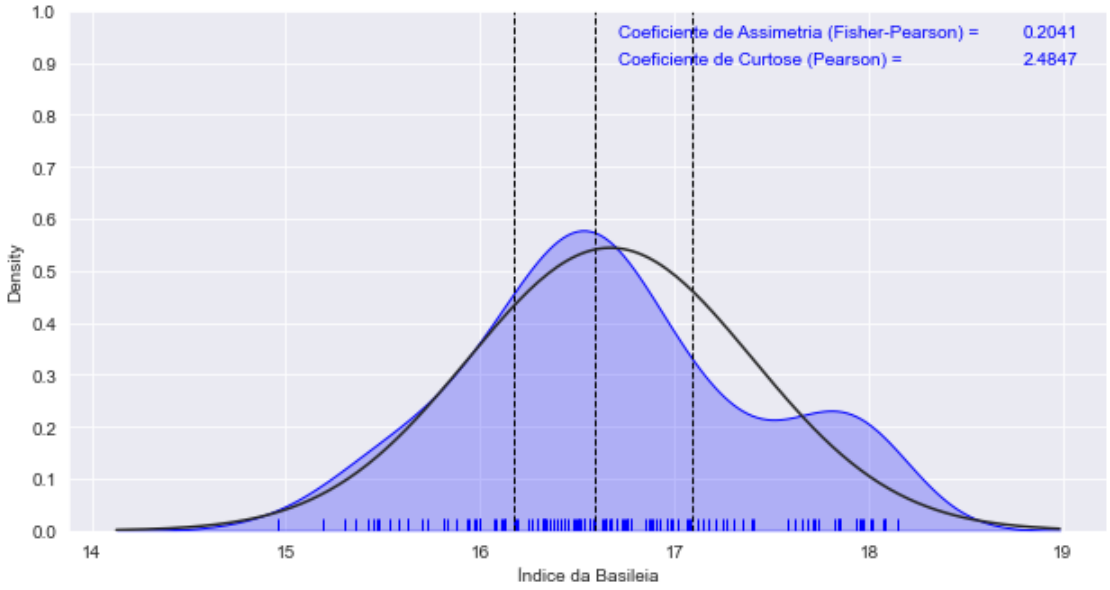
Nas séries "Índice da Basileia" e "Inadimplência Total" a mesma indicação é menos robusta. O segundo quartil (Q2) menos o primeiro quartil (Q1) é menor que o terceiro quartil (Q3) menos o segundo quartil (Q2) - o que pode sugerir assimetrias para direita (enviesadas positivamente). Por outro lado, o primeiro quartil (Q1) menos o valor mínimo é maior que o valor máximo menos o terceiro quartil (Q3) - o que pode sinalizar assimetrias para esquerda (enviesadas negativamente).

- Gráfico da estimação de densidade por kernel (kernel density estimation plot) de todas as séries

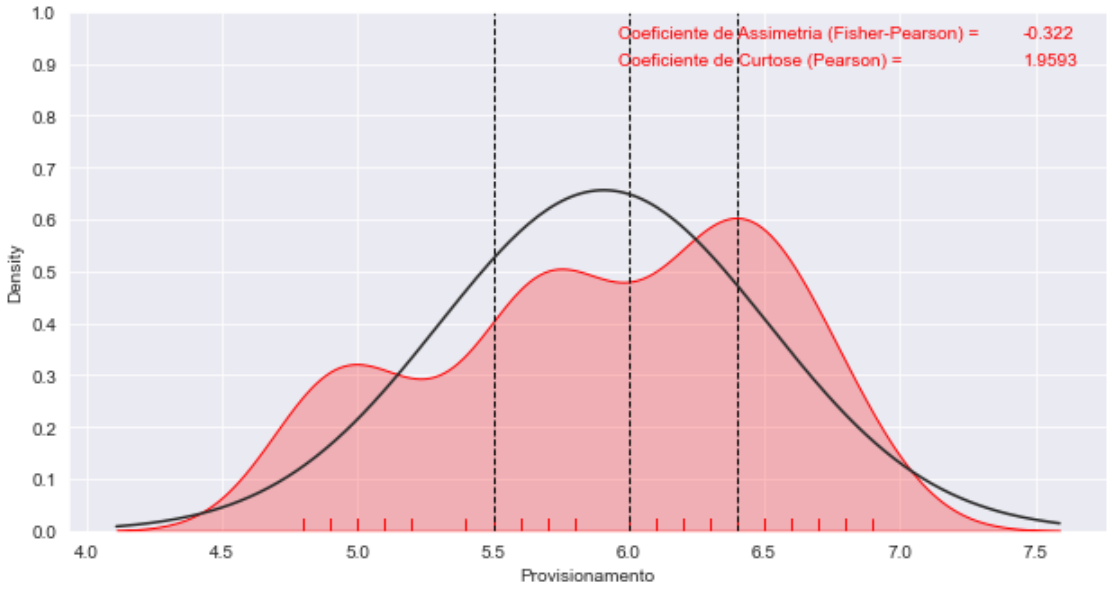
[82]:
```
from scipy import stats
from scipy.stats import skew
from scipy.stats import kurtosis

fig, (ax1, ax2) = plt.subplots(1,2, figsize=(8,4))

ax1 = fig.add_axes([0, 0, 1, 1])
ax1 = sns.kdeplot(x=df['Índice da Basileia'], shade=True, color = 'blue')
ax1 = plt.gca()
ax1.set_ylim([0, 1])
ax1.set_yticks(np.arange(0, 1.1, 0.1))
xmedian = np.median(df['Índice da Basileia'])
plt.axvline(xmedian, c='black',ls='--',linewidth=1)
xq1 = np.percentile(df['Índice da Basileia'],25)
plt.axvline(xq1, c='black',ls='--',linewidth=1)
xq3 = np.percentile(df['Índice da Basileia'],75)
plt.axvline(xq3, c='black',ls='--',linewidth=1)
ax2 = sns.distplot(df['Índice da Basileia'], hist=False, kde=False, fit=stats.
 ↪norm, rug=True, color = 'blue')
fig.text(0.53, .95, 'Coeficiente de Assimetria (Fisher-Pearson) =',␣
 ↪fontsize=10, c="blue")
fig.text(0.92, .95, round(skew (df['Índice da Basileia']),4), fontsize=10,␣
 ↪c="blue")
fig.text(0.53, .90, 'Coeficiente de Curtose (Pearson) =', fontsize=10, c="blue")
fig.text(0.92, .90, round(kurtosis (df['Índice da Basileia'],fisher=False),4),␣
 ↪fontsize=10, c="blue")

plt.show()

from scipy import stats
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(8,4))

ax1 = fig.add_axes([0, 0, 1, 1])
ax1 = sns.kdeplot(x=df['Provisionamento'], shade=True, color = 'red')
ax1 = plt.gca()
ax1.set_ylim([0, 1])
ax1.set_yticks(np.arange(0, 1.1, 0.1))
xmedian = np.median(df['Provisionamento'])
plt.axvline(xmedian, c='black',ls='--',linewidth=1)
xq1 = np.percentile(df['Provisionamento'],25)
plt.axvline(xq1, c='black',ls='--',linewidth=1)
xq3 = np.percentile(df['Provisionamento'],75)
plt.axvline(xq3, c='black',ls='--',linewidth=1)
ax2 = sns.distplot(df['Provisionamento'], hist=False, kde=False, fit=stats.
 ↪norm, rug=True, color = 'red')
```

```
fig.text(0.53, .95, 'Coeficiente de Assimetria (Fisher-Pearson) =',␣
 ↪fontsize=10, c="red")
fig.text(0.92, .95, round(skew (df['Provisionamento']),4), fontsize=10, c="red")
fig.text(0.53, .90, 'Coeficiente de Curtose (Pearson) =', fontsize=10, c="red")
fig.text(0.92, .90, round(kurtosis (df['Provisionamento'],fisher=False),4),␣
 ↪fontsize=10, c="red")

plt.show()

fig, (ax1, ax2) = plt.subplots(1,2, figsize=(8,4))

ax1 = fig.add_axes([0, 0, 1, 1])
ax1 = sns.kdeplot(x=df['Inadimplência Total'], shade=True, color = 'green')
ax1 = plt.gca()
ax1.set_ylim([0, 1])
ax1.set_yticks(np.arange(0, 1.1, 0.1))
xmedian = np.median(df['Inadimplência Total'])
plt.axvline(xmedian, c='black',ls='--',linewidth=1)
xq1 = np.percentile(df['Inadimplência Total'],25)
plt.axvline(xq1, c='black',ls='--',linewidth=1)
xq3 = np.percentile(df['Inadimplência Total'],75)
plt.axvline(xq3, c='black',ls='--',linewidth=1)
ax2 = sns.distplot(df['Inadimplência Total'], hist=False, kde=False, fit=stats.
 ↪norm, rug=True, color = 'green')
fig.text(0.53, .95, 'Coeficiente de Assimetria (Fisher-Pearson) =',␣
 ↪fontsize=10, c="green")
fig.text(0.92, .95, round(skew (df['Inadimplência Total']),4), fontsize=10,␣
 ↪c="green")
fig.text(0.53, .90, 'Coeficiente de Curtose (Pearson) =', fontsize=10,␣
 ↪c="green")
fig.text(0.92, .90, round(kurtosis (df['Inadimplência Total'],fisher=False),4),␣
 ↪fontsize=10, c="green")
plt.show()
```

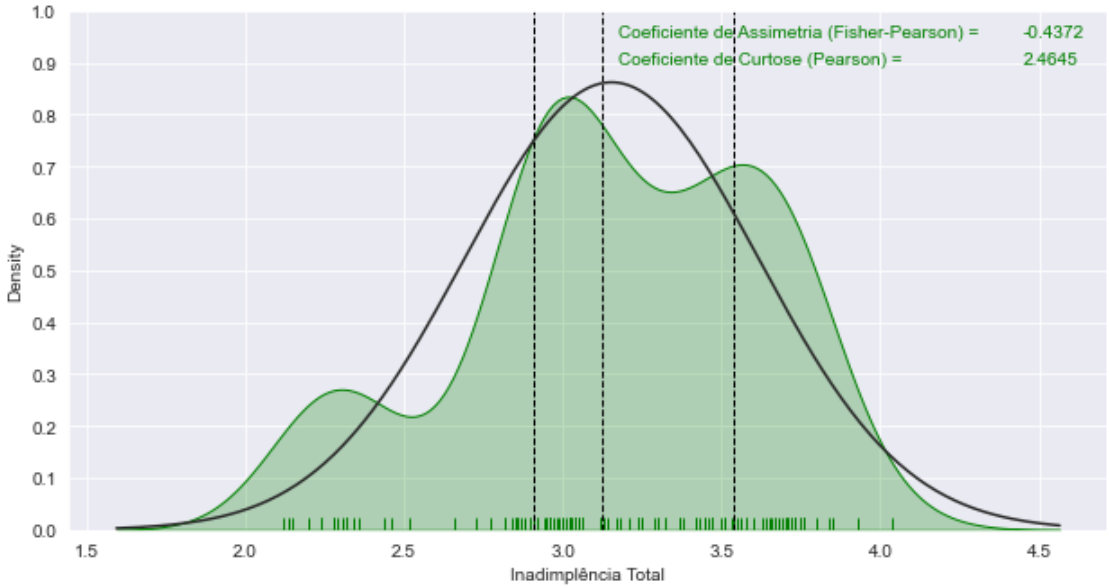
```
C:\Users\mvrgu\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\mvrgu\anaconda3\lib\site-packages\seaborn\distributions.py:2103:
FutureWarning: The `axis` variable is no longer used and will be removed.
Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```




```
C:\Users\mvrgu\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\mvrgu\anaconda3\lib\site-packages\seaborn\distributions.py:2103:
FutureWarning: The `axis` variable is no longer used and will be removed.
Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```



```
C:\Users\mvrgu\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\mvrgu\anaconda3\lib\site-packages\seaborn\distributions.py:2103:
FutureWarning: The `axis` variable is no longer used and will be removed.
Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```



Na série "Índice da Basileia" fica atenuada a ambiguidade anterior. O coeficiente de assimetria (Fisher-Pearson) está situado entre 0 (zero) e +0,5 (mais meio) - indicando uma muito leve assimetria para direita (enviesada positivamente).

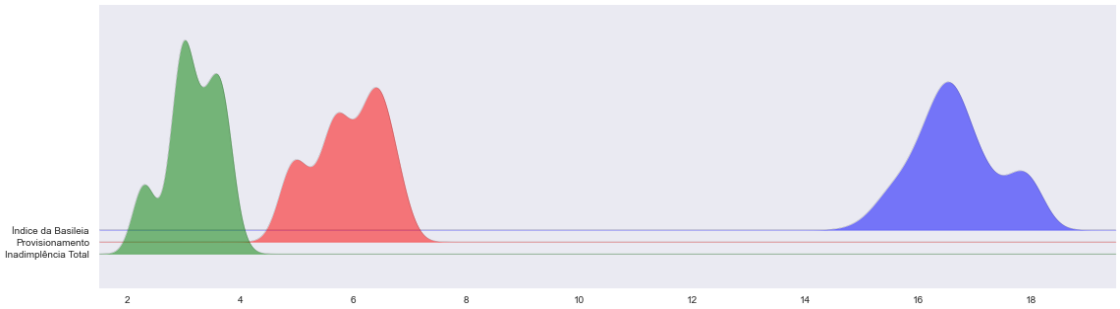
Na série "Provisionamento" fica reforçada a sugestão anterior. O coeficiente de assimetria (Fisher-Pearson) está situado entre -0.5 (menos meio) e 0 (zero), indicando uma leve assimetria para esquerda (enviesada negativamente).

Na série "Inadimplência Total" fica atenuada a ambiguidade anterior. O coeficiente de assimetria (Fisher-Pearson) está situado entre -0.5 (menos meio) e 0 (zero), indicando uma leve assimetria para esquerda (enviesada negativamente).

Já, em termos do achatamento dos dados, todas as séries ("Índice da Basileia", "Provisionamento" e "Inadimplência Total") se mostram do tipo platicúrticas - uma vez que possuem coeficientes de curtose (Pearson) menores que (3) três.

- Gráfico de cumes (ridgeline plot ou joy plot) com os gráficos de densidade de todas as séries sobrepostos

[83]:
```
import joypy
fig, axes = joypy.joyplot(df, overlap = 5,color=["b","r","g",], figsize = (16,4.
↪5), linewidth=.2, alpha=0.5, x_range=(1.5,19.5))
```

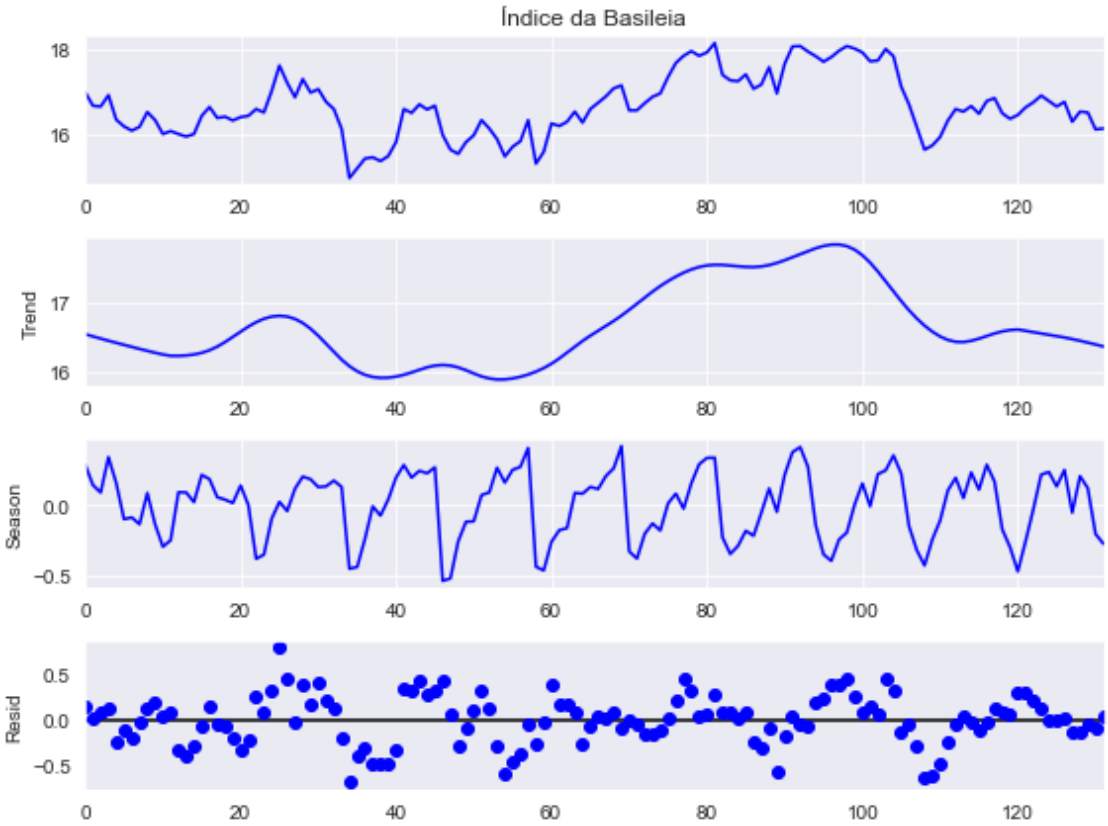


Reafirmando as sugestões anteriores.

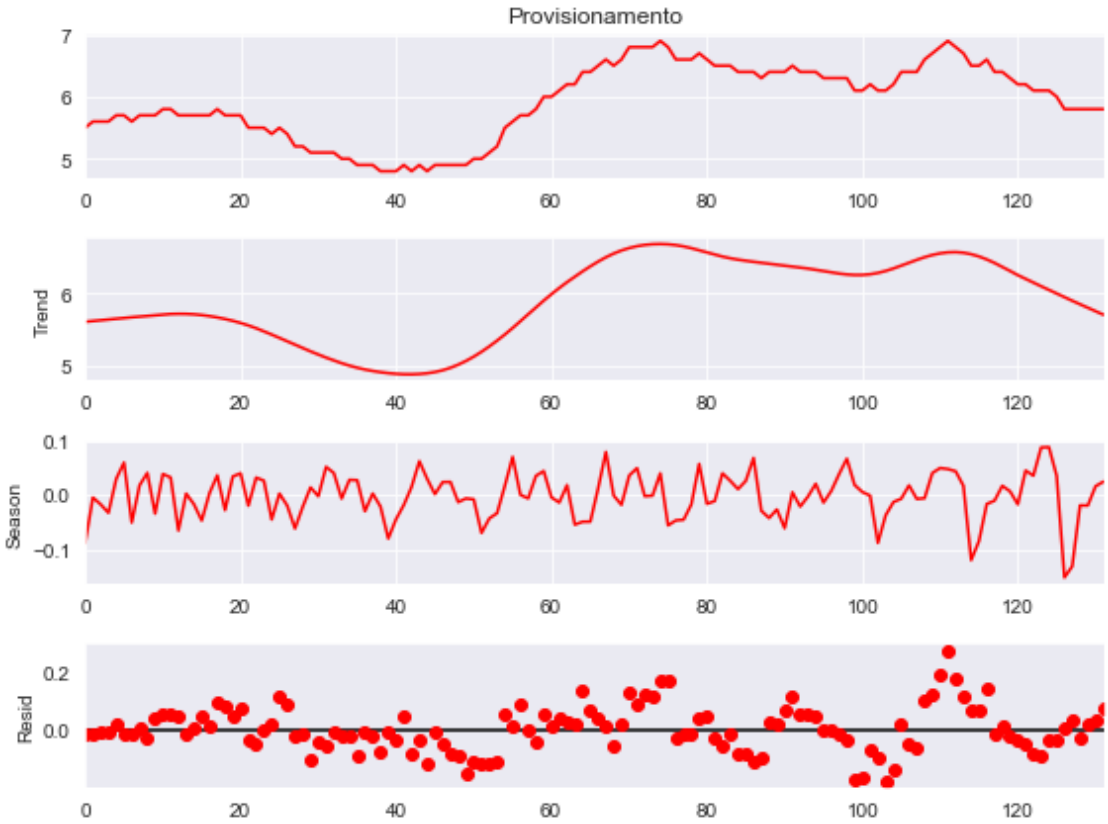
5.5. Decomposição dos dados

- STL using LOESS ou decomposição da sazonalidade e tendência (decomposition plot) de todas as séries

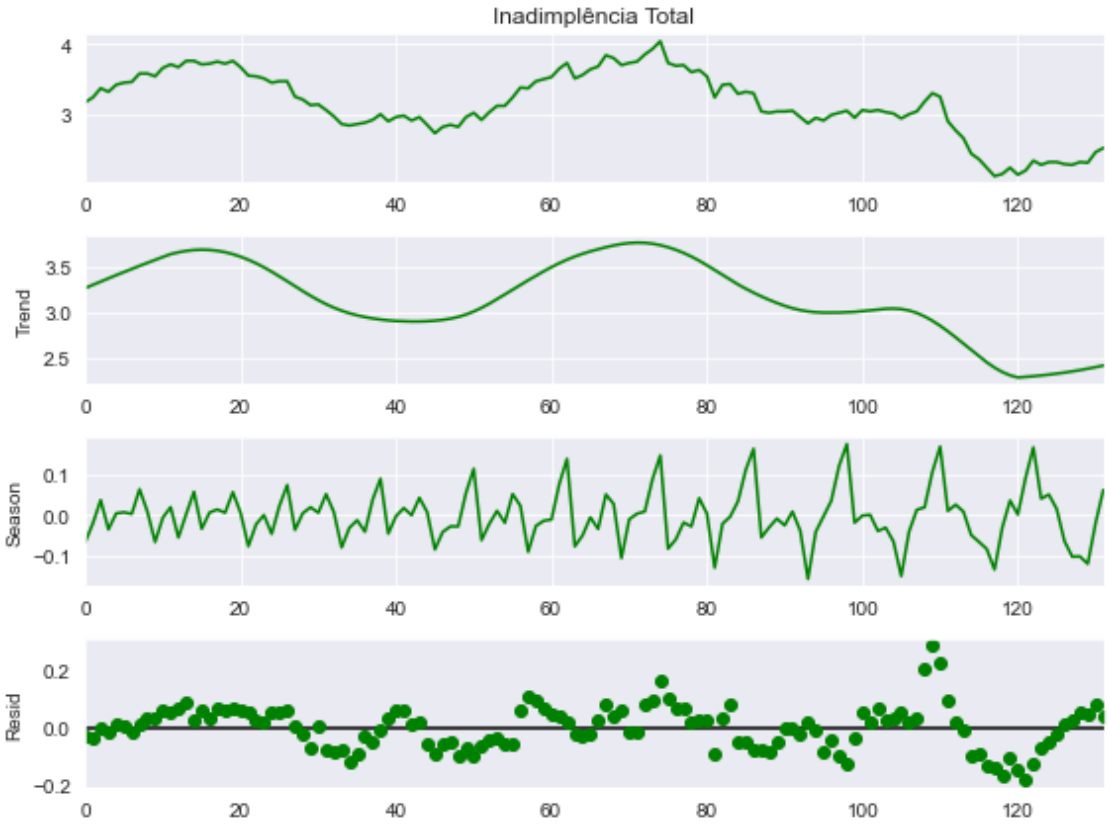
[84]:
```
from pandas.plotting import register_matplotlib_converters
from statsmodels.tsa.seasonal import STL
from cycler import cycler
register_matplotlib_converters()
sns.set_style("darkgrid")
plt.rc("figure", figsize=(8, 6))
plt.rc("font", size=10)
plt.rcParams['axes.prop_cycle'] = cycler(color=['b'])
plt.rcParams['lines.linewidth'] = 1.5
rcParams['lines.linestyle'] = '-'
stl = STL(df['Índice da Basileia'], seasonal=7, period=12)
res = stl.fit()
fig = res.plot()
plt.grid()
```



[85]:
```
sns.set_style("darkgrid")
plt.rc("figure", figsize=(8, 6))
plt.rc("font", size=10)
plt.rcParams['axes.prop_cycle'] = cycler(color=['r'])
plt.rcParams['lines.linewidth'] = 1.5
rcParams['lines.linestyle'] = '-'
stl = STL(df['Provisionamento'], seasonal=7, period=12)
res = stl.fit()
fig = res.plot()
plt.grid()
```



[86]:
```
sns.set_style("darkgrid")
plt.rc("figure", figsize=(8, 6))
plt.rc("font", size=10)
plt.rcParams['axes.prop_cycle'] = cycler(color=['g'])
plt.rcParams['lines.linewidth'] = 1.5
rcParams['lines.linestyle'] = '-'
stl = STL(df['Inadimplência Total'], seasonal=7, period=12)
res = stl.fit()
fig = res.plot()
plt.grid()
```



Todas as séries ("Índice da Basileia", "Provisionamento" e "Inadimplência Total") sugerem a presença de padrão de sazonalidade e ausência de padrão dos resíduos.

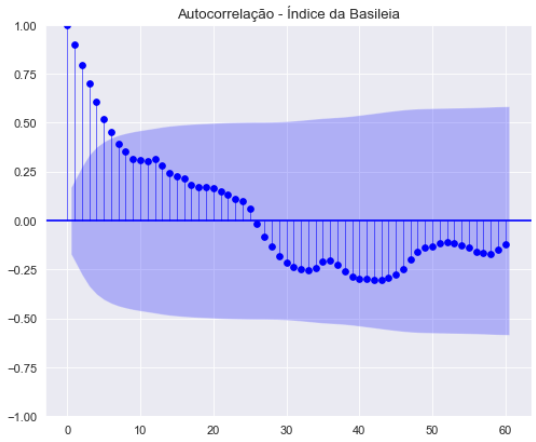
5.6. Autocorrelação dos dados

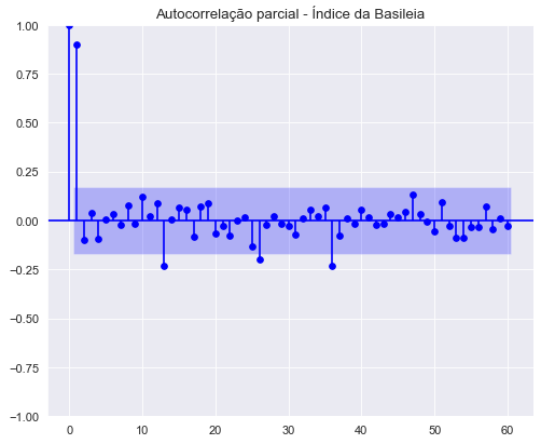
- Função de autocorrelação (ACF plot) e autocorrelação parcial (PACF plot) de todas as séries

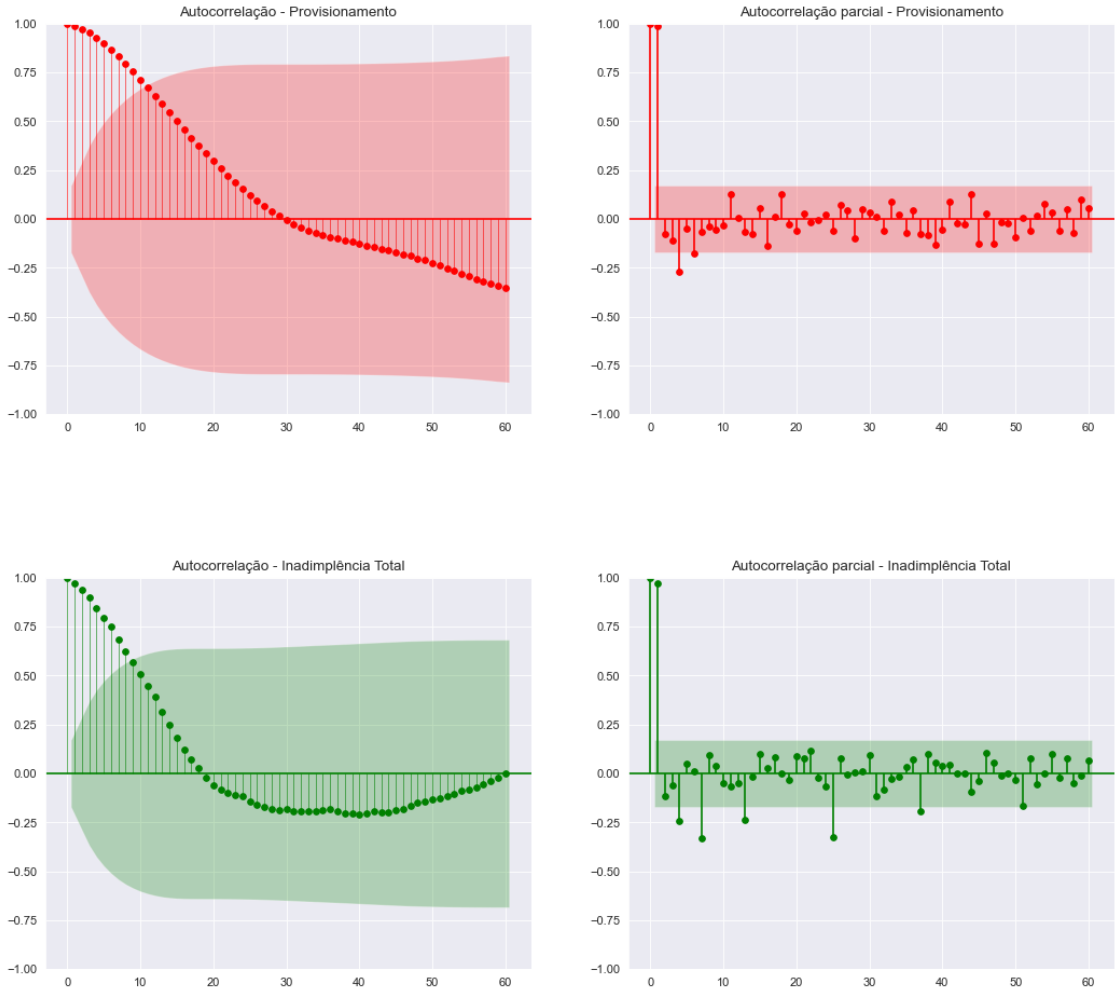
[87]:
```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
from matplotlib.collections import PolyCollection, LineCollection

first_diff = df.shift(1)
first_diff = first_diff.dropna(inplace = False)

fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(16,6), dpi= 80)
plot_acf(df["Índice da Basileia"].values.squeeze(), lags=60, ax=ax1, title=␣
 ↪"Autocorrelação - Índice da Basileia", alpha=0.05,␣
 ↪vlines_kwargs=dict(linewidth=.5))
my_color="blue"
```

```
for item in ax1.collections:
    if type(item)==PolyCollection:
        item.set_facecolor(my_color)
    if type(item)==LineCollection:
        item.set_color(my_color)
for item in ax1.lines:
    item.set_color(my_color)

plot_pacf(df["Índice da Basileia"].values.squeeze(), lags=60,␣
 ↪ax=ax2,method='ywmle', title= "Autocorrelação parcial - Índice da Basileia",␣
 ↪alpha=0.05)
my_color="blue"
for item in ax2.collections:
    if type(item)==PolyCollection:
        item.set_facecolor(my_color)
    if type(item)==LineCollection:
        item.set_color(my_color)
for item in ax2.lines:
    item.set_color(my_color)

plt.show()

fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(16,6), dpi= 80)
plot_acf(df["Provisionamento"].values.squeeze(), lags=60, ax=ax1, title=␣
 ↪"Autocorrelação - Provisionamento", alpha=0.05,␣
 ↪vlines_kwargs=dict(linewidth=.5))
my_color="red"
for item in ax1.collections:
    if type(item)==PolyCollection:
        item.set_facecolor(my_color)
    if type(item)==LineCollection:
        item.set_color(my_color)
for item in ax1.lines:
    item.set_color(my_color)

plot_pacf(df["Provisionamento"].values.squeeze(), lags=60,␣
 ↪ax=ax2,method='ywmle', title= "Autocorrelação parcial - Provisionamento",␣
 ↪alpha=0.05)
my_color="red"
for item in ax2.collections:
    if type(item)==PolyCollection:
        item.set_facecolor(my_color)
    if type(item)==LineCollection:
        item.set_color(my_color)
for item in ax2.lines:
    item.set_color(my_color)
```

```
plt.show()

fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(16,6), dpi= 80)
plot_acf(df["Inadimplência Total"].values.squeeze(), lags=60, ax=ax1, title=␣
 ↪"Autocorrelação - Inadimplência Total", alpha=0.05,␣
 ↪vlines_kwargs=dict(linewidth=.5))
my_color="green"
for item in ax1.collections:
    if type(item)==PolyCollection:
        item.set_facecolor(my_color)
    if type(item)==LineCollection:
        item.set_color(my_color)
for item in ax1.lines:
    item.set_color(my_color)

plot_pacf(df["Inadimplência Total"].values.squeeze(), lags=60,␣
 ↪ax=ax2,method='ywmle', title= "Autocorrelação parcial - Inadimplência␣
 ↪Total", alpha=0.05)
my_color="green"
for item in ax2.collections:
    if type(item)==PolyCollection:
        item.set_facecolor(my_color)
    if type(item)==LineCollection:
        item.set_color(my_color)
for item in ax2.lines:
    item.set_color(my_color)

plt.show()
```







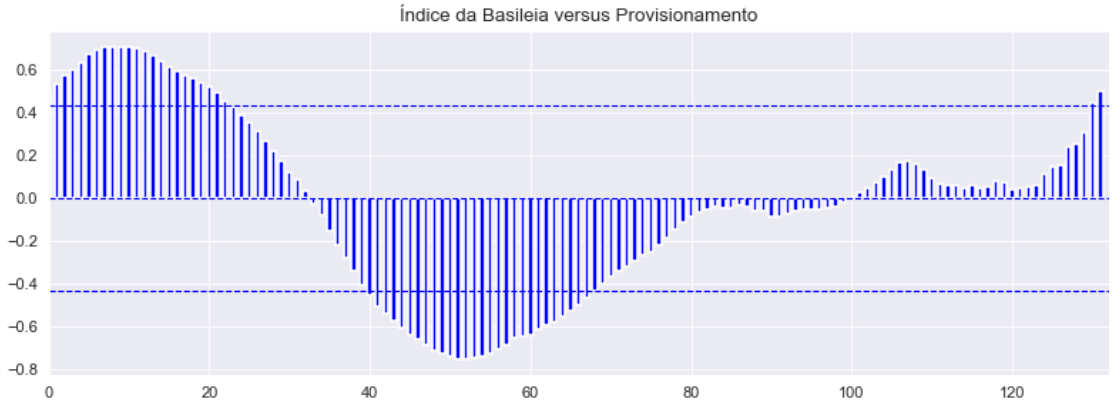
Todas as séries ("Índice da Basileia", "Provisionamento" e "Inadimplência Total") sugerem não aleatoriedade (autocorrelações com pelo menos um ponto significativo) e não estacionariedade (autocorrelações que declinam lentamente para a região não significativa). Ademais, em nenhuma delas existe indicação de padrão de sazonalidade.

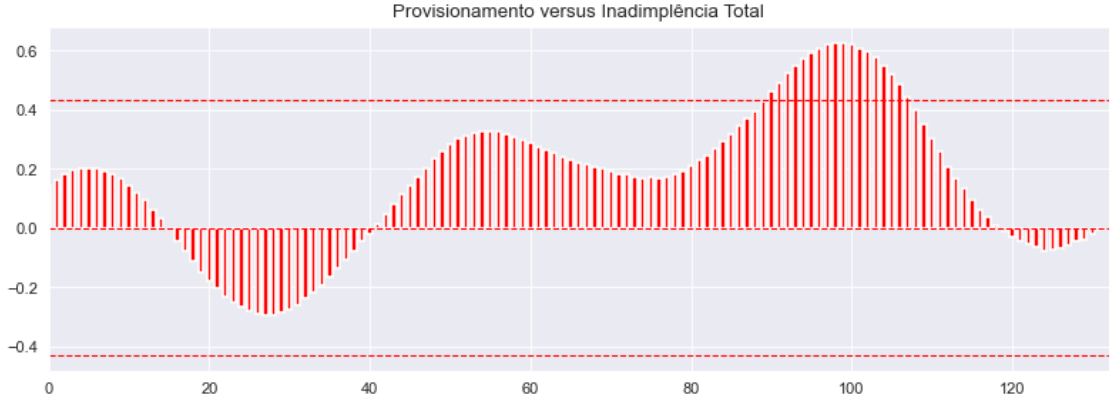
5.7. Correlação cruzada dos dados

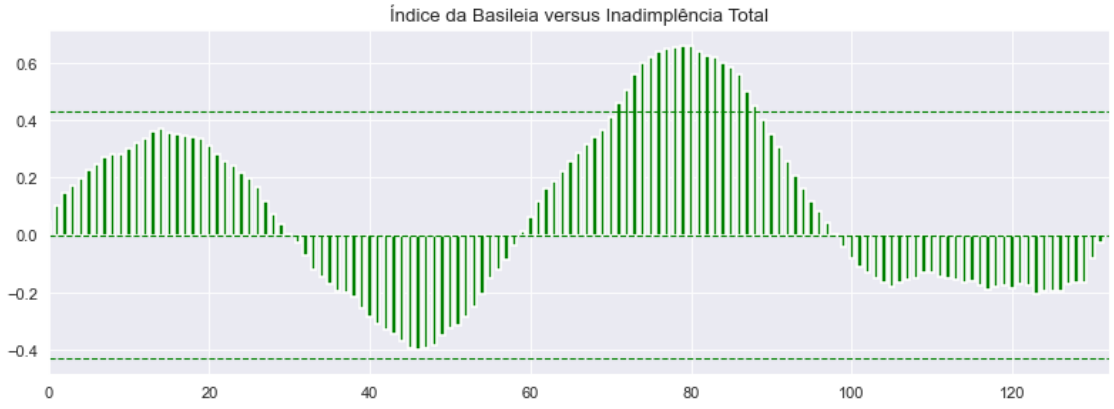
- Função de correlação cruzada (cross correlation ou CCF plot) de todas as séries - duas a duas

[88]:
```
import statsmodels.tsa.stattools as stattools
import math

ccs = stattools.ccf(df["Índice da Basileia"], df["Provisionamento"])
nlags = math.sqrt(len(ccs))+10
conf_level = 2 / np.sqrt(nlags)
```

```
plt.figure(figsize=(12,4), dpi= 80)

plt.hlines(0, xmin=0, xmax=len(ccs), color='b', ls='--', lw=1)
plt.hlines(conf_level, xmin=0, xmax=len(ccs), color='b', ls='--', lw=1)
plt.hlines(-conf_level, xmin=0, xmax=len(ccs), color='b', ls='--', lw=1)

plt.bar(x=np.arange(len(ccs)), height=ccs, width=.6, color='b', lw=1.5)

plt.title('Índice da Basileia versus Provisionamento', fontsize=12)
plt.xlim(0,len(ccs))
plt.show()

ccs = stattools.ccf(df["Provisionamento"], df["Inadimplência Total"])
nlags = math.sqrt(len(ccs))+10
conf_level = 2 / np.sqrt(nlags)

plt.figure(figsize=(12,4), dpi= 80)

plt.hlines(0, xmin=0, xmax=len(ccs), color='r', ls='--', lw=1)
plt.hlines(conf_level, xmin=0, xmax=len(ccs), color='r', ls='--', lw=1)
plt.hlines(-conf_level, xmin=0, xmax=len(ccs), color='r', ls='--', lw=1)

plt.bar(x=np.arange(len(ccs)), height=ccs, width=.6, color='r', lw=1.5)

plt.title('Provisionamento versus Inadimplência Total', fontsize=12)
plt.xlim(0,len(ccs))
plt.show()

ccs = stattools.ccf(df["Índice da Basileia"], df["Inadimplência Total"])
nlags = math.sqrt(len(ccs))+10
conf_level = 2 / np.sqrt(nlags)

plt.figure(figsize=(12,4), dpi= 80)

plt.hlines(0, xmin=0, xmax=len(ccs), color='g', ls='--', lw=1)
plt.hlines(conf_level, xmin=0, xmax=len(ccs), color='g', ls='--', lw=1)
plt.hlines(-conf_level, xmin=0, xmax=len(ccs), color='g', ls='--', lw=1)

plt.bar(x=np.arange(len(ccs)), height=ccs, width=.6, color='g', lw=1.5)

plt.title('Índice da Basileia versus Inadimplência Total', fontsize=12)
plt.xlim(0,len(ccs))
plt.show()
```







No caso de “Índice da Basileia” versus “Provisionamento”, há correlação positiva significativa entre os pontos 0 e 20 (aproximadamente) e correlação negativa significativa entre os pontos 40 e 70 (aproximadamente).

Já, no caso de “Provisionamento” versus “Inadimplência Total”, há correlação positiva significativa entre os pontos 90 e 110 (aproximadamente).

Finalmente, no caso de “Índice da Basileia” versus “Inadimplência Total”, há correlação positiva significativa entre os pontos 70 e 90 (aproximadamente).