In [1]:

```
import pandas as pd
from utils.gerar_score import Score
```

In [2]:

```
df_features = pd.read_excel("C:\\Users\\DELL\\Google Drive\\2021-2\\TCC\\Dataset\\DADOS_SCORE_21_09_refatoracao.xlsx",sheet_name="Planilha2")
```

In [3]:

```
df_features_dh = pd.read_excel("C:\\Users\\DELL\\Google Drive\\2021-2\\TCC\\Dataset\\DADOS_SCORE_21_09_Pablo.xlsx",sheet_name="Planilha2")
```

In [4]:

```
df = pd.read_excel("C:\\Users\\DELL\\Google Drive\\2021-2\\TCC\\Dataset\\dados tratados missing 5.xlsx")
```

In [5]:

```
df.head()
```

Out[5]:

	IN002	IN031	IN101	IN049	IN019	IN023	IN024	IN055	IN056	IN057	...	Código do Prestador
0	573.25	29.49	129.76	27.52	505.21	91.53	14.56	57.19	14.12	100.0	...	31062000
1	396.04	42.86	92.39	28.12	268.67	96.83	3.47	84.09	3.02	100.0	...	31062000
2	248.83	49.62	102.38	30.76	162.33	98.98	98.98	69.54	69.54	100.0	...	31003011
3	532.22	170.89	26.44	89.86	453.06	99.88	99.42	99.70	65.20	0.0	...	31004011
4	365.25	34.90	102.62	22.28	319.54	73.69	15.01	34.46	6.22	100.0	...	31062000

5 rows × 41 columns

In [6]:

```
df_features.head()
```

Out[6]:

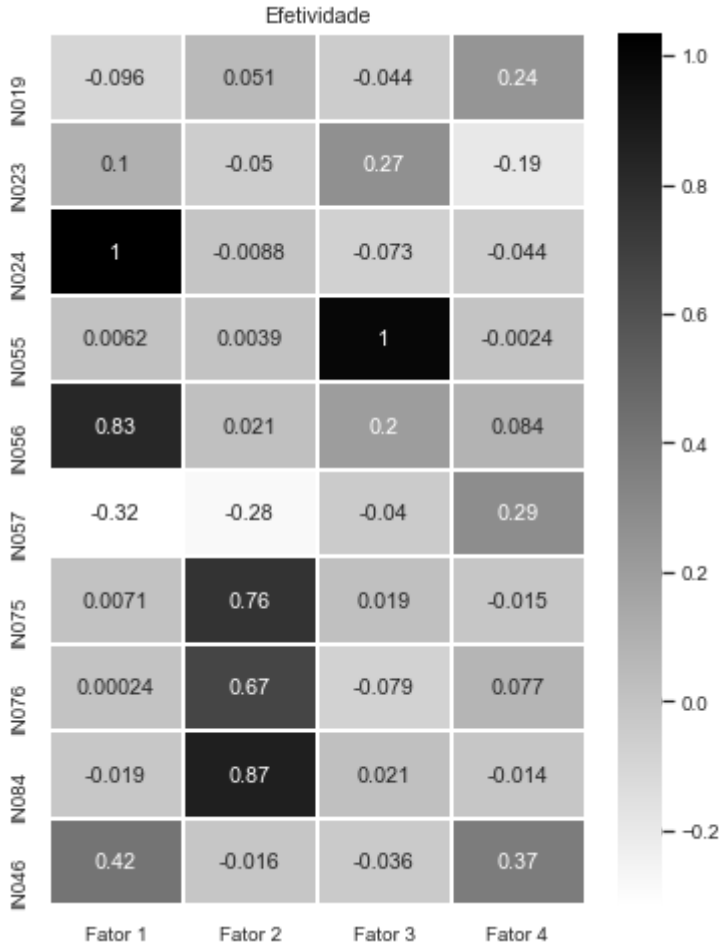
	Variavel	Grupo	Sentido
0	IN002	Eficiencia	0
1	Tarifa	Eficiencia	0
2	IN031	Eficiencia	1
3	IN101	Eficiencia	0
4	IN049	Eficiencia	1

In [7]:

```
import os
if os.path.exists("Scores.xlsx") == True:
    os.remove("Scores.xlsx")
```

In [8]:

```
score=Score(df,df_features,delta=0.05,normalizar_score=True,save_excel=True,rotacao='ob
limin',verbose=True)
```

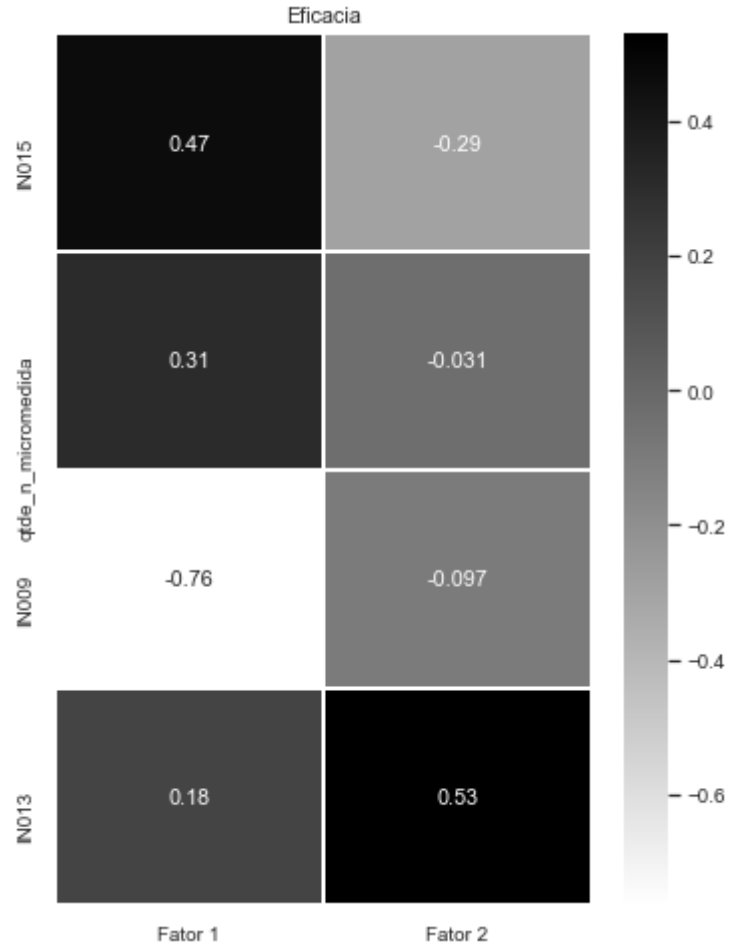


Fatores sem inverter:

	0	1	2	3
IN019	-0.072208	-0.029852	-0.067511	0.218481
IN023	0.176908	0.061442	0.299937	-0.162899
IN024	0.996467	0.184250	0.337369	0.089012
IN055	0.405126	0.181441	0.999115	0.022015
IN056	0.921024	0.193546	0.537581	0.190905
IN057	-0.348934	-0.413651	-0.209399	0.308624
IN075	0.157036	0.769318	0.155459	-0.186445
IN076	0.105186	0.635669	0.040958	-0.076182
IN084	0.151533	0.868852	0.165677	-0.212363
IN046	0.450530	-0.027558	0.138461	0.431661

Fatores invertidos:

	0	1	2	3
IN019	-0.072208	-0.029852	-0.067511	0.218481
IN023	0.176908	0.061442	0.299937	-0.162899
IN024	0.996467	0.184250	0.337369	0.089012
IN055	0.405126	0.181441	0.999115	0.022015
IN056	0.921024	0.193546	0.537581	0.190905
IN057	0.348934	0.413651	0.209399	-0.308624
IN075	-0.157036	-0.769318	-0.155459	0.186445
IN076	-0.105186	-0.635669	-0.040958	0.076182
IN084	-0.151533	-0.868852	-0.165677	0.212363
IN046	0.450530	-0.027558	0.138461	0.431661

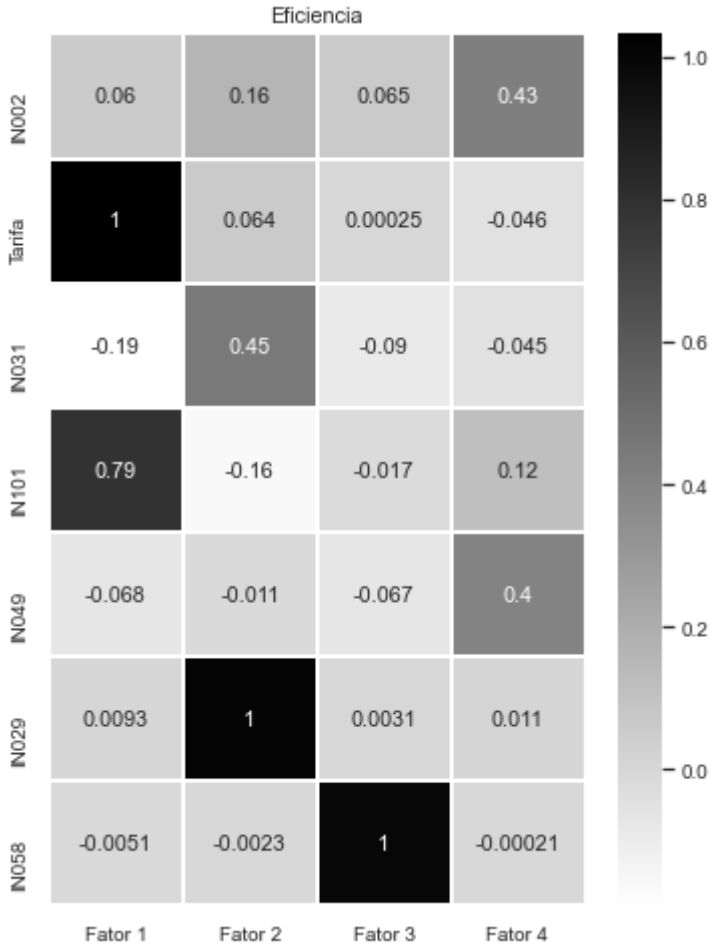


Fatores sem inverter:

	0	1
IN015	0.328208	-0.067787
qtde_n_micromedida	0.295635	0.117749
IN009	-0.811023	-0.462929
IN013	0.434437	0.615631

Fatores invertidos:

	0	1
IN015	0.328208	-0.067787
qtde_n_micromedida	-0.295635	-0.117749
IN009	0.811023	0.462929
IN013	-0.434437	-0.615631


Eficiencia
IN002
Tarifa
IN031
IN101
IN049
IN029
IN058
0.06
0.16
0.065
0.43
1
0.064
0.00025
-0.046
-0.19
0.45
-0.09
-0.045
0.79
-0.16
-0.017
0.12
-0.068
-0.011
-0.067
0.4
0.0093
1
0.0031
0.011
-0.0051
-0.0023
1
-0.00021
Fator 1
Fator 2
Fator 3
Fator 4
1.0
0.8
0.6
0.4
0.2
0.0

```
Fatores sem inverter:
              0         1         2         3
IN002   0.135409  0.067393  0.057510  0.415375
Tarifa  0.997474 -0.085826 -0.262117  0.264884
IN031  -0.344991 -0.109734  0.529849 -0.196519
IN101   0.882679 -0.061726 -0.438679  0.396572
IN049   0.067729 -0.042709 -0.063119  0.382322
IN029  -0.312895 -0.068800  0.998782 -0.180534
IN058  -0.081143  0.997662 -0.071810  0.043477

Fatores invertidos:
              0         1         2         3
IN002   0.135409  0.067393  0.057510  0.415375
Tarifa  0.997474 -0.085826 -0.262117  0.264884
IN031   0.344991  0.109734 -0.529849  0.196519
IN101   0.882679 -0.061726 -0.438679  0.396572
IN049  -0.067729  0.042709  0.063119 -0.382322
IN029   0.312895  0.068800 -0.998782  0.180534
IN058   0.081143 -0.997662  0.071810 -0.043477
------------------------------------------------- Scores ---------------
---------------------------------
     Efetividade  Eficacia  Eficiencia  Score_Medio
0       0.278768  0.479107    0.748444        0.502
1       0.434799  0.481388    0.415645        0.444
2       0.869300  0.570610    0.396414        0.612
3       0.448986  0.001145    0.068606        0.173
4       0.133093  0.515938    0.270161        0.306
..           ...       ...         ...          ...
807     0.492997  0.739941    0.154548        0.462
808     0.476946  0.690504    0.517146        0.562
809     0.789010  0.628349    0.679416        0.699
810     0.784191  0.755610    0.073622        0.538
811     0.156908  0.502135    0.206158        0.288

[812 rows x 4 columns]
```

Comparação com anterior

In [9]:

```
cidades = ['Rio Doce','Bom Sucesso','Uberaba','Mantena','Papagaios','Lagoa da Prata','C
armópolis de Minas','Patrocínio','Monte Carmelo',
'Machado','Itaguara','São José da Varginha','Sacramento','Japaraíba','Arantina','Caraí'
,'São Sebastião do Maranhão','Setubinha',
'São João da Ponte','Presidente Bernardes','São José do Jacuri','Guaraciaba','Luisburg
o','Serra Azul de Minas','Icaraí de Minas',
'Gonçalves','Santo Antônio do Retiro','Ladainha']

cidades1=["Uberlândia",
"Araporã",
"Divinópolis",
"Pará de Minas",
"Itabirito",
"Caeté",
"Cabeceira Grande",
"Florestal",
"Monjolos",
"Pratinha"]

cidades2=["Nova Serrana"]
munic = []

for i in cidades1:
    munic.append(df[df['Nome_Município']==i].index[0])

#munic
```

In [10]:

```
df_score=pd.read_excel("C:\\Users\\DELL\\Google Drive\\2021-2\\TCC\\Codigos\\Versão Fin
al\\Scores.xlsx")
```

In [11]:

```
score_filtred=df_score.iloc[munic]
score_filtred.insert(0, 'Ranking', range(1, 1 + len(score_filtred)))

###POR CONCEITO - TABELA SCORE FINAL E MUNICIPIO
ordenado = score_filtred.sort_values('Score_Medio', ascending=False)
ordenado.insert(0, 'Rank_media', range(1, 1 + len(ordenado)))
ordenado.sort_index(ascending=True, inplace=True)
#ordenado
```

In [12]:

```
 pd.set_option('display.float_format', '{:.8f}'.format)
```

In [13]:

```
##Só pra ajudar na visualização
score_mun=df["Nome_Município"]
a=score_mun.iloc[munic]
show=ordenado

ordenado = score_filtred.sort_values('Score_Medio', ascending=False)
ordenado.insert(1, 'Rank_calculado', range(1, 1 + len(ordenado)))
ordenado.sort_index(ascending=True, inplace=True)


difs = list(score_filtred.Ranking-ordenado.Rank_calculado)
show.insert(0,"Cidade",a)
show.insert(3,"Diferença",difs)
show.sort_values('Ranking')
```

Out[13]:

	Cidade	Rank_media	Ranking	Diferença	Efetividade	Eficacia	Eficiencia	Score
750	Uberlândia	1	1	0	0.98127489	0.84754721	0.97129324	0.93
35	Araporã	6	2	-4	0.76076715	0.15953985	0.94607451	0.62
229	Divinópolis	3	3	0	0.96180584	0.64732484	0.80028762	0.80
501	Pará de Minas	2	4	2	0.97968370	0.86364084	0.70032103	0.84
333	Itabirito	4	5	1	0.94576406	0.71639026	0.49714648	0.72
100	Caeté	5	6	1	0.78065648	0.85795361	0.52225679	0.72
94	Cabeceira Grande	9	7	-2	0.13964795	0.67031711	0.21439685	0.34
268	Florestal	7	8	1	0.52342766	0.44099085	0.49658500	0.48
446	Monjolos	8	9	1	0.36740781	0.35840833	0.32238180	0.34
565	Pratinha	10	10	0	0.20887140	0.02779472	0.29973652	0.17

Análise Descritiva

In [14]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from matplotlib import pyplot
from numpy import median
import numpy as np
```

In [15]:

```
## Adicionando scores no dataset
df['Score']=df_score["Score_Medio"]
df['Score_Eficacia']=df_score["Eficacia"]
df['Score_Efetividade']=df_score["Efetividade"]
df['Score_Eficiencia']=df_score["Eficiencia"]
#print(df.columns.values)
# #df
```

Melhores e piores colocados

In [16]:

```
df_aux= df.loc[:, ['Nome_Município','Score_Efetividade','Score_Eficiencia','Score_Eficacia','Score','Natureza jurídica']]
df_aux.nlargest(10,'Score')
```

Out[16]:

	Nome_Município	Score_Efetividade	Score_Eficiencia	Score_Eficacia	Score	N
361	Jacutinga	0.98302145	1.00000000	0.92590327	0.97000000	Admin públic
750	Uberlândia	0.98127489	0.97129324	0.84754721	0.93300000	A
191	Coqueiral	0.96857120	0.82915670	0.92657718	0.90800000	A
243	Elói Mendes	0.92898565	0.85619021	0.93891614	0.90800000	A
415	Mantena	0.97563544	0.87506393	0.87236005	0.90800000	A
449	Monte Alegre de Minas	0.97487513	0.85263853	0.89341999	0.90700000	Admin públic
542	Pirajuba	0.89342264	0.98439806	0.83976216	0.90600000	Socie e mi admin
114	Campo do Meio	0.95617319	0.81306161	0.93723755	0.90200000	A
603	Sacramento	0.97139919	0.91684370	0.79340073	0.89400000	A
175	Conceição das Alagoas	0.97572384	0.92077929	0.77997203	0.89200000	Admin públic

In [17]:

```
pd.set_option('display.float_format', lambda x: '%.6f' % x)
df_aux.nsmallest (10,'Score')
```

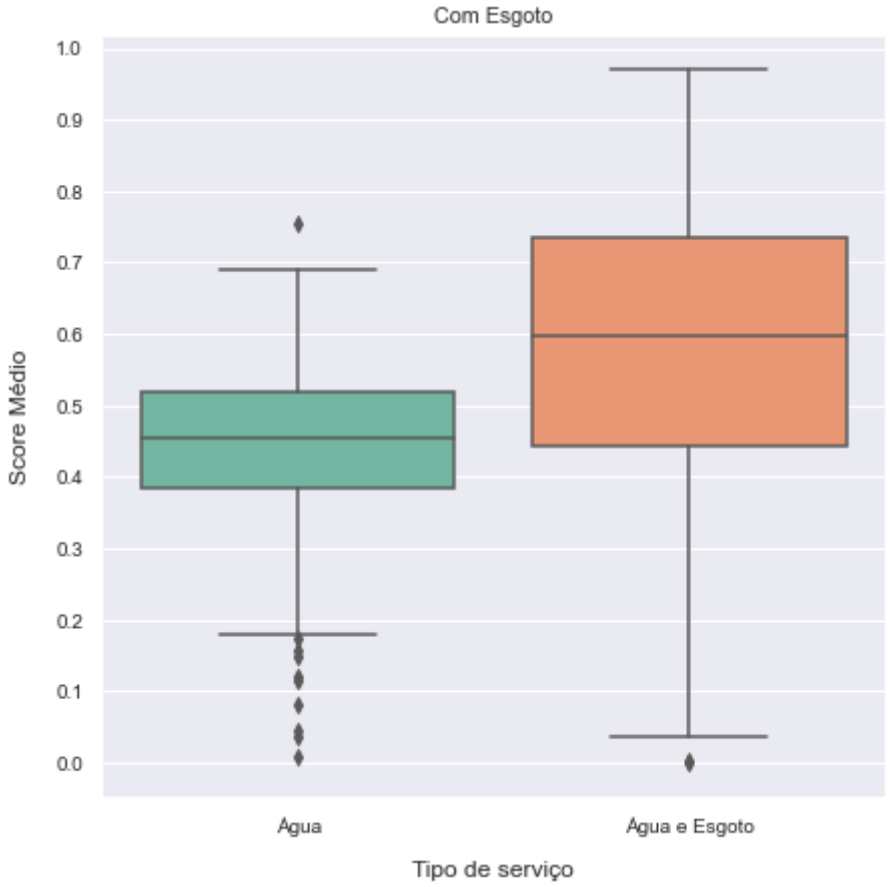
Out[17]:

	Nome_Município	Score_Efetividade	Score_Eficiencia	Score_Eficacia	Score	Nat ju
341	Itambé do Mato Dentro	0.000000	0.000000	0.000673	0.000000	Administ pública
133	Caranaíba	0.000161	0.010882	0.000040	0.004000	Administ pública
33	Arantina	0.023654	0.004695	0.000206	0.010000	Socieda eco mist administ
573	Queluzito	0.051924	0.056053	0.000038	0.036000	Administ pública
396	Lamim	0.019293	0.092817	0.000012	0.037000	Administ pública
594	Rio Preto	0.133437	0.002824	0.000010	0.045000	Administ pública
9	Aiuruoca	0.246633	0.000005	0.000669	0.082000	Administ pública
662	São João da Lagoa	0.120434	0.114405	0.011283	0.082000	Administ pública
199	Coronel Pacheco	0.248631	0.001214	0.000671	0.084000	Administ pública
153	Catas Altas	0.119297	0.134142	0.003248	0.086000	Administ pública

Boxplot Tipo de serviço

In [18]:

```
df_aux=df.loc[:,["Tipo de serviço",'Score']]
fig, b2 = plt.subplots(1,1, figsize=(7,7))
b2 = sns.boxplot(x="Tipo de serviço", y="Score", palette='Set2', data=df_aux)

b2.set_xlabel("Tipo de serviço", fontsize=12, labelpad=12)
b2.set_ylabel("Score Médio", fontsize=12, labelpad=12)
b2.set_yticks(np.arange(0.0, 1.1, 0.1))

b2.set_title("Com Esgoto")
plt.show()
```



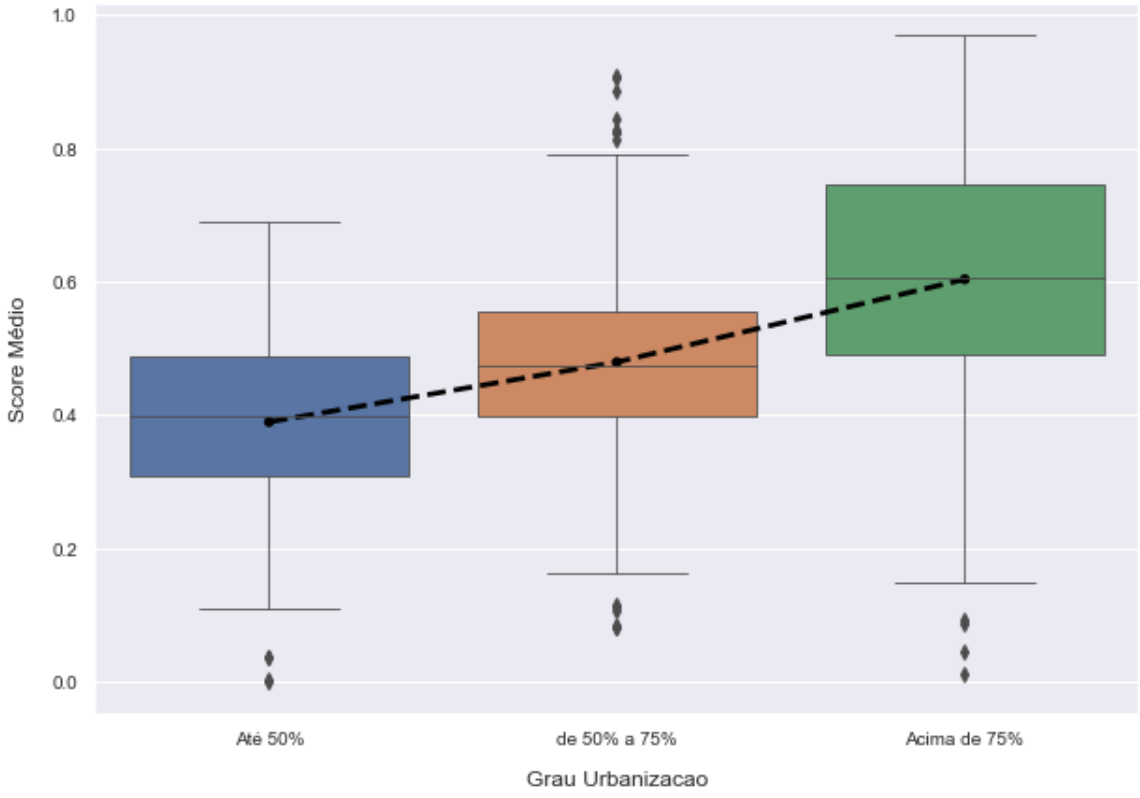
Boxplot Grau de urbanização

In [19]:

```
df_aux=df.loc[:, ['Score','grau_urbanizacao','Nome_Município']]
df_aux['Niveis_GU'] = np.where(df_aux.grau_urbanizacao<=0.50, "Até 50%",
                               np.where((df_aux.grau_urbanizacao>0.50)&(df_aux.grau_ur
banizacao<=0.75), "de 50% a 75%",
                                        "Acima de 75%") )
fig, ax = plt.subplots(1,1, figsize=(10, 7))

order=["Até 50%","de 50% a 75%","Acima de 75%"]

sns.set_style("ticks")
ax = sns.boxplot(x='Niveis_GU', y='Score', order=order, showfliers=True, linewidth=0.8,
showmeans=False, data=df_aux)
ax = sns.pointplot(x='Niveis_GU', y='Score', order=order, data=df_aux, ci=None, color=
'black', linestyles='--',markers = '.')

ax.set_xlabel("Grau Urbanizacao", fontsize=12, labelpad=12)
ax.set_ylabel("Score Médio", fontsize=12, labelpad=12)
```

Out[19]:

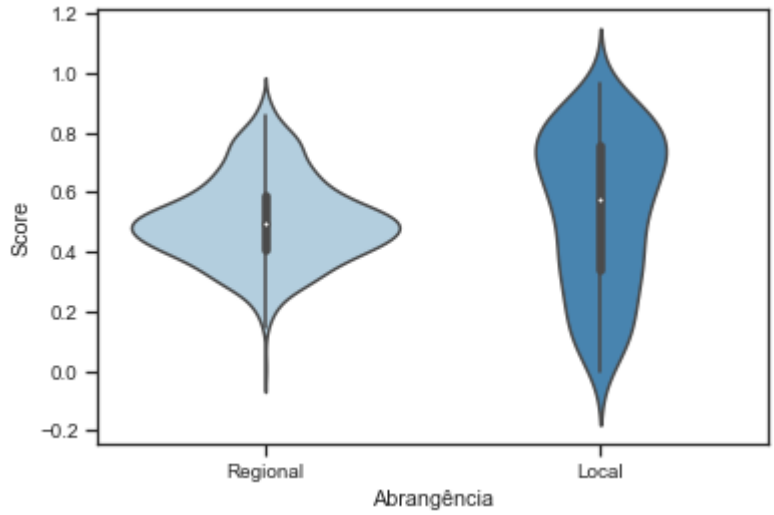
```
Text(0, 0.5, 'Score Médio')
```



Boxplot Abrangência

In [20]:

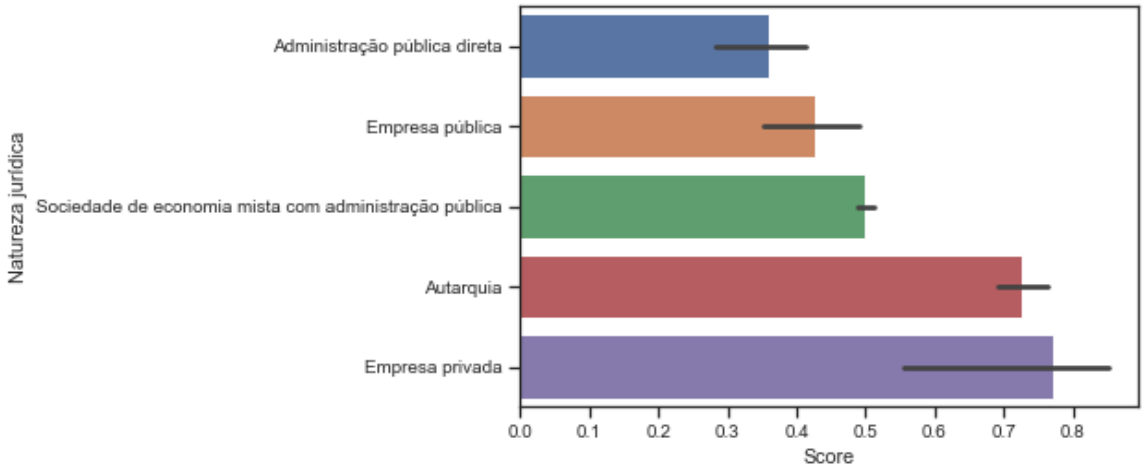
```
df_aux=df.loc[:, ['Score','Abrangência']]
violins = sns.violinplot(x="Abrangência", y="Score", data=df_aux,widths=0.45, palette=
'Blues')
```



Natureza juridica (Mediana)

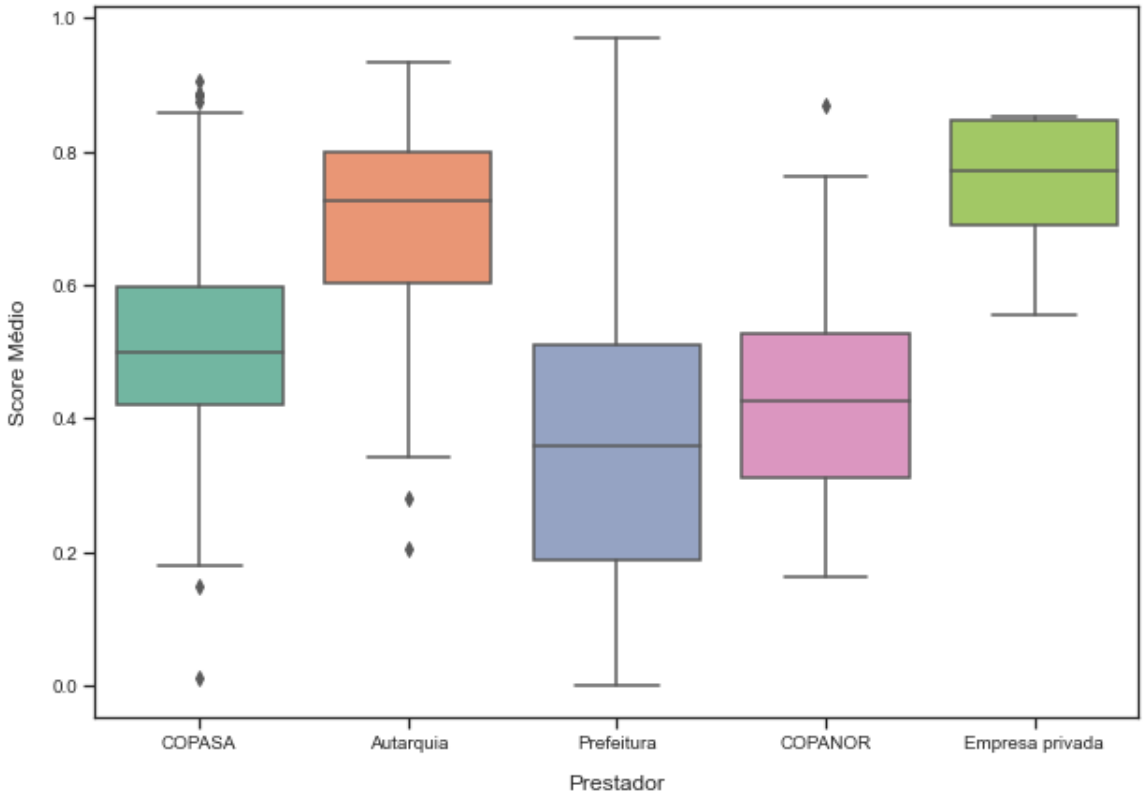
In [21]:

```
df_aux=df.loc[:, ['Score','Natureza jurídica']]
order=df_aux.groupby(["Natureza jurídica"])["Score"].median().sort_values().index
p=sns.barplot(x='Score', y='Natureza jurídica', data=df_aux, estimator=np.median, order
=order)
```



In [22]:

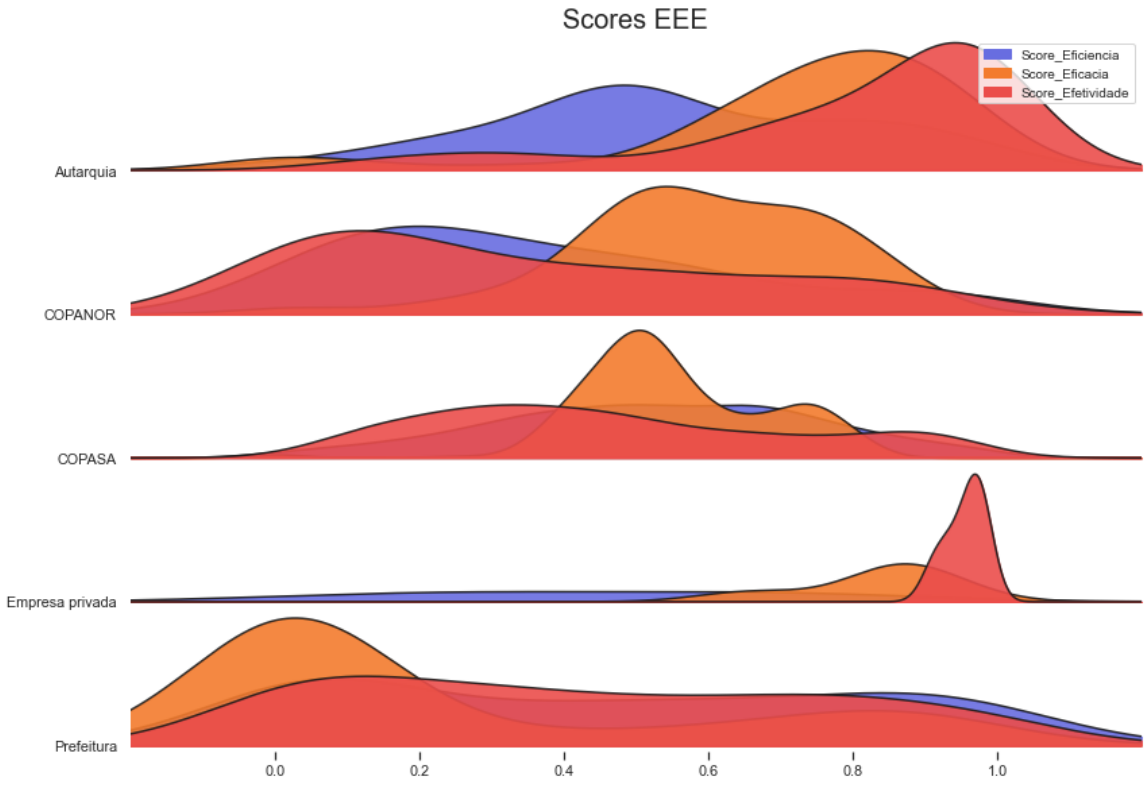
```
df_aux=df.loc[:, ['Score','Prestador2']]

fig, b2 = plt.subplots(1,1, figsize=(10, 7))
b2 = sns.boxplot(x="Prestador2", y="Score", palette='Set2', data=df_aux)

b2.set_xlabel("Prestador", fontsize=12, labelpad=12)
b2.set_ylabel("Score Médio", fontsize=12, labelpad=12)

plt.show()
```



In [23]:

```
from joypy import joyplot

df_aux=df.loc[:, ['Score_Efetividade','Score_Eficiencia','Score_Eficacia','Prestador2'
]]
# df_aux=df_aux.melt(id_vars=["prestador2"],
#         var_name="Tipo_Score",
#         value_name="Scores")
# df_aux["Merge"]=df_aux["prestador2"]+" "+df_aux["Tipo_Score"]

plt.figure()

ax, fig = joyplot(
    data=df_aux,
    by='Prestador2',
    column=['Score_Eficiencia','Score_Eficacia','Score_Efetividade'],
    color=['#686de0'#azul
           ,"#f37b2d"#laranja
           , '#eb4d4b' #vermelho
          ],
    legend=True,
    alpha=0.9,
    figsize=(12, 8),
    ylim='own',
    overlap=0
)
plt.title('Scores EEE', fontsize=20)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

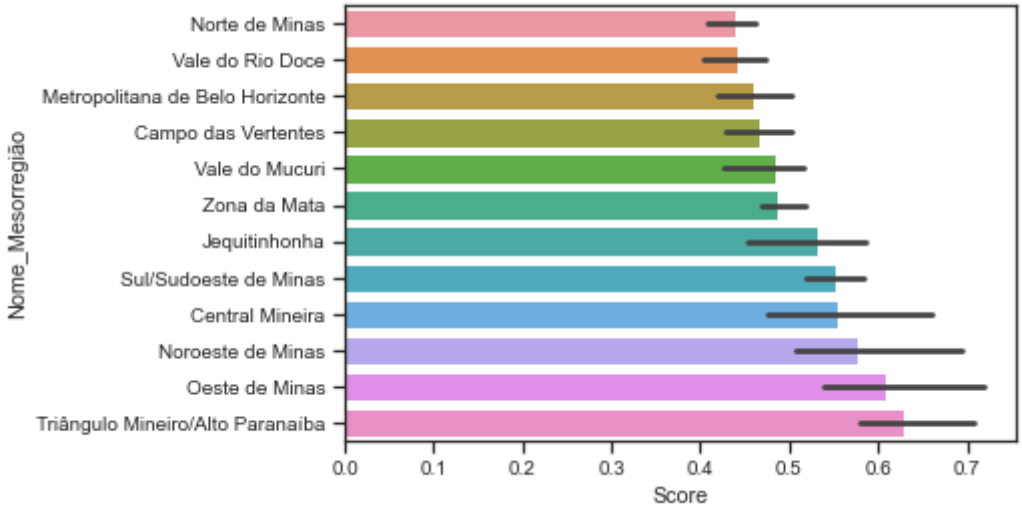



```
<Figure size 432x288 with 0 Axes>
```

Mesoregião (Mediana)

In [24]:

```
df_aux=df.loc[:, ['Score','Nome_Mesorregião']]
order=df_aux.groupby(["Nome_Mesorregião"])["Score"].median().sort_values().index
p=sns.barplot(x='Score', y='Nome_Mesorregião', data=df_aux, estimator=np.median, order=
order)
```



In [25]:

```
df_aux=df.loc[:, ['Score','Nome_Mesorregião']]
#df_aux.groupby('NV08Meso')['Score'].median()
df_aux=df_aux.groupby(df_aux.Nome_Mesorregião)[['Score']].median()
df_aux.sort_values('Score')
```

Out[25]:

Nome_Mesorregião	Score
Norte de Minas	0.439000
Vale do Rio Doce	0.442000
Metropolitana de Belo Horizonte	0.460500
Campo das Vertentes	0.467500
Vale do Mucuri	0.484000
Zona da Mata	0.487000
Jequitinhonha	0.532000
Sul/Sudoeste de Minas	0.552500
Central Mineira	0.555000
Noroeste de Minas	0.576500
Oeste de Minas	0.609000
Triângulo Mineiro/Alto Paranaíba	0.630000

In [26]:

```
import shapefile as shp
from unidecode import unidecode
import geopandas as gpd
import folium
from folium import plugins
import json
import branca.colormap as cmp
import shapely.geometry
import plotly.express as px
```

In [27]:

```
with open("C:/Users/DELL/Google Drive/2021-2/TCC/geojs-31-mun.json", encoding="utf8") a
s file:
    geo_json_data = json.load(file)
```

In [28]:

```
df_media = pd.DataFrame( {'name': df['Nome_Município'], 'Score':df['Score'],'Score_Efic
acia':df['Score_Eficiencia'],
                          'Score_Efetividade':df['Score_Efetividade'],'Score_Eficienci
a':df['Score_Eficiencia'],
                         'Score_Eficiencia':df['Score_Eficiencia']})

df_MG = df_media[~df_media.duplicated(subset=['name'], keep='first')]
#df_MG.loc[:, 'Cidade'] = df_MG.loc[:, 'Cidade'].str.upper()

df_MG['Quantil'] = np.where(df_MG.Score<=0.25, "(0-25)%",
                            np.where((df_MG.Score>0.25)&(df_MG.Score<=0.5), "(25-50)%",
                                     np.where((df_MG.Score>0.5)&(df_MG.Score<=0.75), "
(50-75)%", "(75-100)%")
                                    )
                           )
with open("C:/Users/DELL/Google Drive/2021-2/TCC/geojs-31-mun.json", encoding="utf8") a
s file:
    geo_json_data = json.load(file)

geo_df = gpd.GeoDataFrame.from_features(geo_json_data["features"]).merge(df_MG, on="nam
e").set_index("name")
#geo_df
```

In [29]:

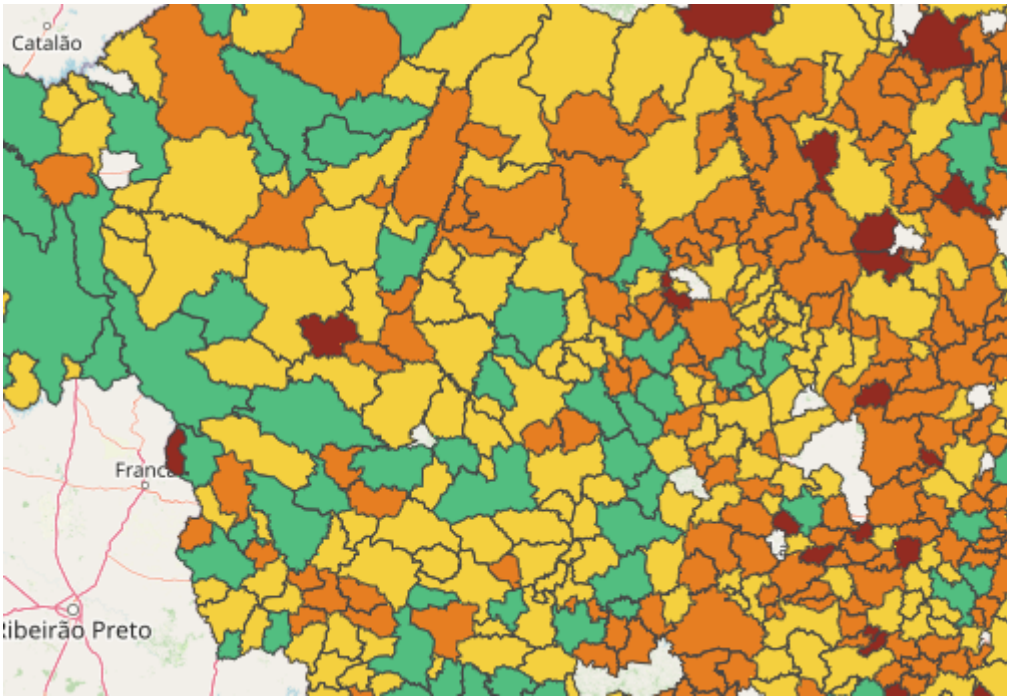
```
# import plotly.io as pio
# pio.renderers.default = 'browser'
```

Mapa Score

In [30]:

```
df_media = pd.DataFrame( {'name': df['Nome_Município'], 'Score':round(df['Score'],4),'S
core_Eficacia':round(df['Score_Eficacia'],4),
                          'Score_Efetividade':round(df['Score_Efetividade'],4),'Score_E
ficiencia':round(df['Score_Eficiencia'],4),
                          'Nome_Mesorregião':df['Nome_Mesorregião']})
df_MG = df_media[~df_media.duplicated(subset=['name'], keep='first')]
#df_MG.loc[:, 'Cidade'] = df_MG.loc[:, 'Cidade'].str.upper()

df_MG['Quantil'] = np.where(df_MG.Score<=0.25, "(0-25)%",
                            np.where((df_MG.Score>0.25)&(df_MG.Score<=0.5), "(25-50)%",
                                     np.where((df_MG.Score>0.5)&(df_MG.Score<=0.75), "
(50-75)%", "(75-100)%")
                                    )
                           )
with open("C:/Users/DELL/Google Drive/2021-2/TCC/geojs-31-mun.json", encoding="utf8") a
s file:
    geo_json_data = json.load(file)

geo_df = gpd.GeoDataFrame.from_features(geo_json_data["features"]).merge(df_MG, on="nam
e").set_index("name")
fig = px.choropleth_mapbox(geo_df,
                           geojson=geo_df.geometry,
                           locations=geo_df.index,
                           color="Quantil",
                           category_orders= {'Quantil':["(0-25)%","(25-50)%","(50-75)%"
, "(75-100)%"]},
                           color_discrete_sequence=["#922B21", "#E67E22", "#F4D03F"," #
52BE80"],
                           center={"lat": -19.84164, "lon": -43.98651},
                           mapbox_style="open-street-map",
                           zoom=6,
                           #hover_name='name',
                           hover_data=['Quantil','Score','Score_Eficacia','Score_Efetiv
idade','Score_Eficiencia'],
                           title="Score"
                          )
fig.show()
```

Score

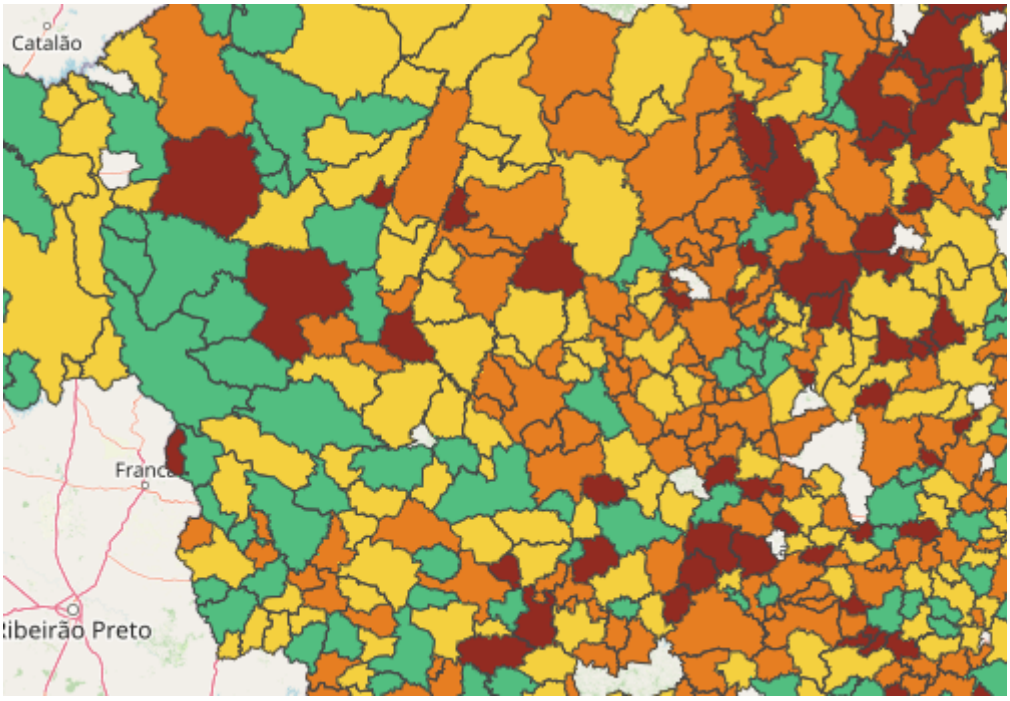


Mapa eficiencia

In [31]:

```
df_media = pd.DataFrame( {'name': df['Nome_Município'], 'Score':round(df['Score'],4),'S
core_Eficacia':round(df['Score_Eficacia'],4),
                          'Score_Efetividade':round(df['Score_Efetividade'],4),'Score_E
ficiencia':round(df['Score_Eficiencia'],4),
                          'Nome_Mesorregião':df['Nome_Mesorregião']})
df_MG = df_media[~df_media.duplicated(subset=['name'], keep='first')]
#df_MG.loc[:, 'Cidade'] = df_MG.loc[:, 'Cidade'].str.upper()

df_MG['Quantil'] = np.where(df_MG.Score_Eficiencia<=0.25, "(0-25)%",
                            np.where((df_MG.Score_Eficiencia>0.25)&(df_MG.Score_Eficien
cia<=0.5), "(25-50)%",
                                     np.where((df_MG.Score_Eficiencia>0.5)&(df_MG.Score
_Eficiencia<=0.75), "(50-75)%", "(75-100)%")
                                    )
                           )
with open("C:/Users/DELL/Google Drive/2021-2/TCC/geojs-31-mun.json", encoding="utf8") a
s file:
    geo_json_data = json.load(file)

geo_df = gpd.GeoDataFrame.from_features(geo_json_data["features"]).merge(df_MG, on="nam
e").set_index("name")
fig = px.choropleth_mapbox(geo_df,
                           geojson=geo_df.geometry,
                           locations=geo_df.index,
                           color="Quantil",
                           category_orders= {'Quantil':["(0-25)%","(25-50)%","(50-75)%"
, "(75-100)%"]},
                           color_discrete_sequence=["#922B21", "#E67E22", "#F4D03F"," #
52BE80"],
                           center={"lat": -19.84164, "lon": -43.98651},
                           mapbox_style="open-street-map",
                           zoom=6,
                           #hover_name='name',
                           hover_data=['Quantil','Score','Score_Eficacia','Score_Efetiv
idade','Score_Eficiencia'],
                           title="Score_Eficiencia"
                          )
fig.show()
```

Score_Eficiencia

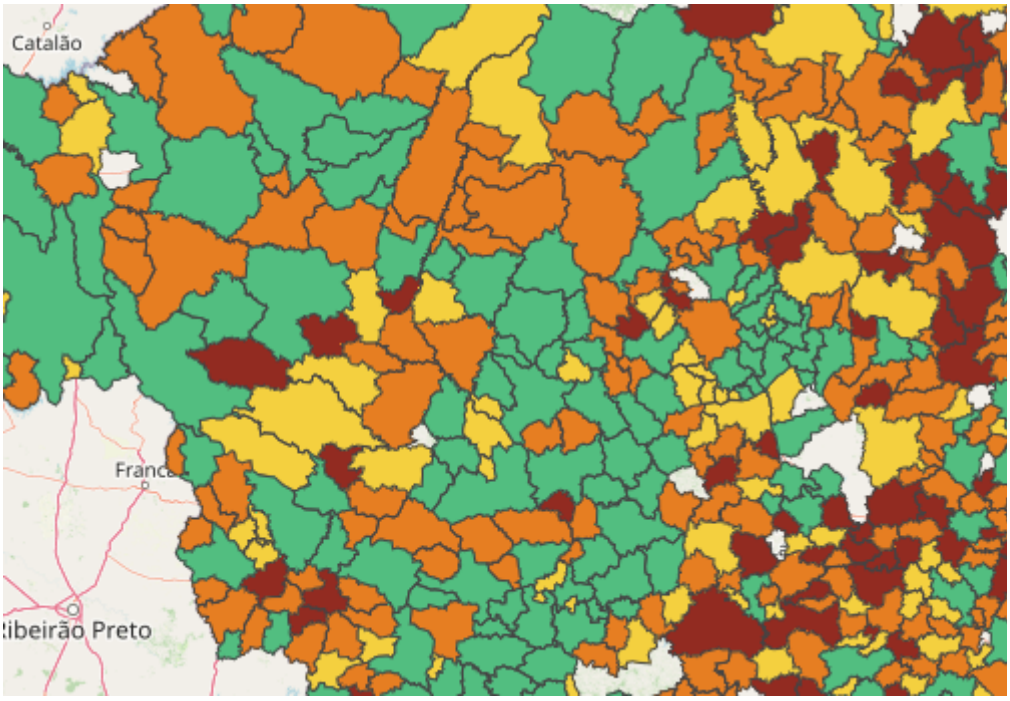


Mapa Efetividade

In [32]:

```
df_media = pd.DataFrame( {'name': df['Nome_Município'], 'Score':round(df['Score'],4),'S
core_Eficacia':round(df['Score_Eficacia'],4),
                          'Score_Efetividade':round(df['Score_Efetividade'],4),'Score_E
ficiencia':round(df['Score_Eficiencia'],4),
                          'Nome_Mesorregião':df['Nome_Mesorregião']})
df_MG = df_media[~df_media.duplicated(subset=['name'], keep='first')]
#df_MG.loc[:, 'Cidade'] = df_MG.loc[:, 'Cidade'].str.upper()

df_MG['Quantil'] = np.where(df_MG.Score_Efetividade<=0.25, "(0-25)%",
                            np.where((df_MG.Score_Efetividade>0.25)&(df_MG.Score_Efetiv
idade<=0.5), "(25-50)%",
                                     np.where((df_MG.Score_Efetividade>0.5)&(df_MG.Scor
e_Efetividade<=0.75), "(50-75)%", "(75-100)%")
                                    )
                           )
with open("C:/Users/DELL/Google Drive/2021-2/TCC/geojs-31-mun.json", encoding="utf8") a
s file:
    geo_json_data = json.load(file)

geo_df = gpd.GeoDataFrame.from_features(geo_json_data["features"]).merge(df_MG, on="nam
e").set_index("name")
fig = px.choropleth_mapbox(geo_df,
                           geojson=geo_df.geometry,
                           locations=geo_df.index,
                           color="Quantil",
                           category_orders= {'Quantil':["(0-25)%","(25-50)%","(50-75)%"
, "(75-100)%"]},
                           color_discrete_sequence=["#922B21", "#E67E22", "#F4D03F"," #
52BE80"],
                           center={"lat": -19.84164, "lon": -43.98651},
                           mapbox_style="open-street-map",
                           zoom=6,
                           #hover_name='name',
                           hover_data=['Quantil','Score','Score_Eficacia','Score_Efetiv
idade','Score_Eficiencia'],
                           title="Score_Efetividade"
                          )
fig.show()
```

Score_Efetividade



Mapa Eficacia

In [33]:

```
df_media = pd.DataFrame( {'name': df['Nome_Município'], 'Score':round(df['Score'],4),'S
core_Eficacia':round(df['Score_Eficacia'],4),
                          'Score_Efetividade':round(df['Score_Efetividade'],4),'Score_E
ficiencia':round(df['Score_Eficiencia'],4),
                          'Nome_Mesorregião':df['Nome_Mesorregião']})
df_MG = df_media[~df_media.duplicated(subset=['name'], keep='first')]
#df_MG.loc[:, 'Cidade'] = df_MG.loc[:, 'Cidade'].str.upper()

df_MG['Quantil'] = np.where(df_MG.Score_Efetividade<=0.25, "(0-25)%",
                            np.where((df_MG.Score_Efetividade>0.25)&(df_MG.Score_Efetiv
idade<=0.5), "(25-50)%",
                                     np.where((df_MG.Score_Efetividade>0.5)&(df_MG.Scor
e_Efetividade<=0.75), "(50-75)%", "(75-100)%")
                                    )
                           )
with open("C:/Users/DELL/Google Drive/2021-2/TCC/geojs-31-mun.json", encoding="utf8") a
s file:
    geo_json_data = json.load(file)

geo_df = gpd.GeoDataFrame.from_features(geo_json_data["features"]).merge(df_MG, on="nam
e").set_index("name")
fig = px.choropleth_mapbox(geo_df,
                           geojson=geo_df.geometry,
                           locations=geo_df.index,
                           color="Quantil",
                           category_orders= {'Quantil':["(0-25)%","(25-50)%","(50-75)%"
, "(75-100)%"]},
                           color_discrete_sequence=["#922B21", "#E67E22", "#F4D03F"," #
52BE80"],
                           center={"lat": -19.84164, "lon": -43.98651},
                           mapbox_style="open-street-map",
                           zoom=6,
                           #hover_name='name',
                           hover_data=['Quantil','Score','Score_Eficacia','Score_Efetiv
idade','Score_Eficiencia'],
                           title="Score_Eficacia"
                          )
fig.show()
```

Score_Eficacia