Chapter 4 - Quantitative assessment of the database

November 10, 2020

Jupyter notebook used to generate analysis of the database of morphological characters.

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[ ]:
```
sns.set()
```

[ ]:
```
data = pd.read_csv('characters.csv', index_col=0, delimiter=';')
```

[ ]:
```
statistics_grain = pd.DataFrame(index = ['Block', 'Neighbourhood', 'Urban␣
↪Area'], columns = ['Dimension', 'Shape', 'Distribution', 'Intensity',␣
↪'Connectivity', 'Diversity']).fillna(0)
```

[ ]:
```
for idx, r in data.iterrows():
    for scale in ['Block', 'Neighbourhood', 'Urban Area']:
        if scale in r['Scale of grain']:
            for cat in ['Dimension', 'Shape', 'Distribution', 'Intensity',␣
↪'Connectivity', 'Diversity']:
                if cat == idx:
                    statistics_grain.loc[scale, cat] = statistics_grain.
↪loc[scale, cat] + 1
```

[ ]:
```
statistics_grain['Total'] = statistics_grain.sum(axis=1)
```

[ ]:
```
statistics_grain.loc['Total']= statistics_grain.sum()
```

[ ]:
```
statistics_grain.rename(index={'Block': "Small", "Neighbourhood": "Medium",␣
↪"Urban Area": "Large"}, inplace=True)
```

[ ]:
```
sns.set_style("whitegrid")
sns.set_context("paper")
```

[ ]:
```
ax = statistics_grain.drop(columns='Total').drop('Total').plot(kind='bar',␣
↪figsize=(12, 8), cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
```

```
ax.yaxis.grid(False)
plt.title('Grain')
ax.set_ylim([0, 75])
plt.savefig('divided_bar.svg')
```

[ ]:
```
ax = statistics_grain.Total.drop('Total').plot(kind='bar', figsize=(6, 4),␣
↪cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.title('Grain')
ax.set_ylim([0, 240])
plt.savefig('total_grain.svg')
```

[ ]:
```
ax = statistics_grain.loc['Total'].drop('Total').plot(kind='bar', figsize=(12,␣
↪8), cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.title('Grain')
plt.savefig('total_categories.png')
```

[ ]:
```
statistics_extent = pd.DataFrame(index = ['Block', 'Neighbourhood', 'Urban␣
↪Area'], columns = ['Dimension', 'Shape', 'Distribution', 'Intensity',␣
↪'Connectivity', 'Diversity']).fillna(0)
```

[ ]:
```
for idx, r in data.iterrows():
    for scale in ['Block', 'Neighbourhood', 'Urban Area']:
        if scale in r['Scale of information extent']:
            for cat in ['Dimension', 'Shape', 'Distribution', 'Intensity',␣
↪'Connectivity', 'Diversity']:
                if cat == idx:
                    statistics_extent.loc[scale, cat] = statistics_extent.
↪loc[scale, cat] + 1
```

[ ]:
```
statistics_extent['Total'] = statistics_extent.sum(axis=1)
statistics_extent.loc['Total']= statistics_extent.sum()
statistics_extent.rename(index={'Block': "Small", "Neighbourhood": "Medium",␣
↪"Urban Area": "Large"}, inplace=True)
```

[ ]:
```
ax = statistics_extent.drop(columns='Total').drop('Total').plot(kind='bar',␣
↪figsize=(12, 8), cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.title('Extent')
```

```
ax.set_ylim([0, 75])
plt.savefig('divided_bar_ext.svg')
```

[ ]:
```
ax = statistics_extent.Total.drop('Total').plot(kind='bar', figsize=(6, 4),␣
↪cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.title('Extent')
ax.set_ylim([0, 240])
plt.savefig('total_ext.svg')
```

[ ]:
```
ax = statistics_extent.loc['Total'].drop('Total').plot(kind='bar', figsize=(12,␣
↪8), cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.title('Extent')
#plt.savefig('total_categories.png')
```

[ ]:
```
statistics_extent.to_csv("statistics_extent.csv")
statistics_grain.to_csv("statistics_grain.csv")
```

[ ]:
```
statistics_extent
```

[ ]:
```
statistics_grain
```

[ ]:
```
df = pd.DataFrame(data.index.value_counts(sort=False))

df['index2'] = pd.Categorical(
    df.index,
   ␣
↪categories=['Dimension','Shape','Distribution','Intensity','Connectivity','Diversity'],
    ordered=True)
```

[ ]:
```
df
```

[ ]:
```
ax = df.sort_values('index2').plot(kind='bar', figsize=(12, 8), cmap='Set2')
sns.despine(offset=5)
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.savefig('cats.svg')
```

[ ]:
```
pd.DataFrame(data.index.value_counts(sort=False)).reset_index()
```