Fair-IRT_Law

June 13, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.colors as mcolors
import seaborn as sns

from irt import Beta3

from matplotlib import rcParams
import matplotlib.pyplot as plt
```

[2]:
```
pij = pd.read_csv('./Law_Pij.csv')
pij.set_index(pij.columns[0], inplace=True)

random_seed = 42
pij = pij.sample(n=1000, random_state=random_seed)
```

[3]:
```
original_shape = pij.values.shape
array = pij.values.flatten()

data = np.where(array > 1, 1, array)
normalized_array = data.reshape(original_shape)
normalized_df = pd.DataFrame(normalized_array, index=pij.index, columns=pij.
 ↪columns)
```

[4]:
```
normalized_df = 1 - normalized_df
```

[5]:
```
def ICC_function(abilities, difficulties, discriminations):
    a = ((1-abilities)/ abilities)
    b = (difficulties / (1-difficulties))
    c = a*b
    d = c**discriminations
    return (1 / (d+1))
```

[6]:
```
b4 = Beta3(
        learning_rate=100,
        epochs=10000,
        n_respondents=normalized_df.shape[1],
        n_items=normalized_df.shape[0],
        n_inits=1000,
        n_workers=-1,
        random_seed=1,
    )
b4.fit(normalized_df.values)
```

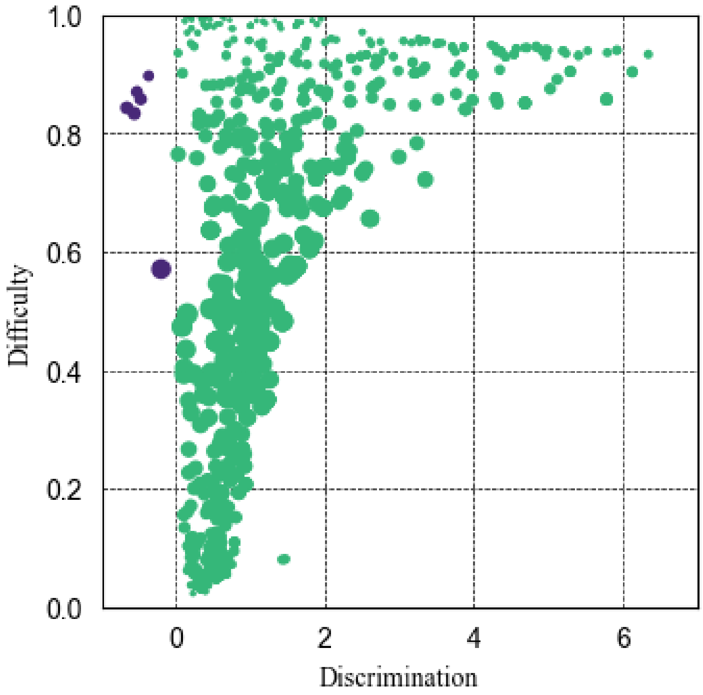
```
100%|      | 10000/10000 [00:16<00:00, 607.58it/s]
```

[6]:
```
<irt.Beta3 at 0x103e863d0>
```

[7]:
```
new_pij = pd.DataFrame(index=range(1000), columns=range(48))

for i in range(1000):
    for j in range(15):
        alpha = (b4.abilities[j] / b4.difficulties[i]) ** b4.discriminations[i]
        beta_val = ((1 - b4.abilities[j]) / (1 - b4.difficulties[i])) ** b4.
↪discriminations[i]
        new_pij.iloc[i, j] = (alpha)/(alpha+beta_val)
```

1 Figure 10(a)

[8]:
```
def plot_discriminations_difficulties(discriminations, difficulties,␣
↪normalized_df, font_size=10, font_ann_size=5, base_point_size=500):
    rcParams['font.family'] = 'serif'
    rcParams['font.serif'] = ['Times New Roman']

    sns.set_style('whitegrid')
    fig, ax = plt.subplots(figsize=(4, 4))
    ax.grid(color='black', linestyle='--', linewidth=0.5)

    ax.spines['bottom'].set_color('black')
    ax.spines['left'].set_color('black')
    ax.spines['right'].set_color('black')
    ax.spines['top'].set_color('black')
    ax.xaxis.label.set_color('black')
    ax.yaxis.label.set_color('black')
    ax.tick_params(axis='x', colors='black')
    ax.tick_params(axis='y', colors='black')

    point_sizes = base_point_size * (1 - np.abs(difficulties - 0.5) * 2)

    colors = []
    for disc, diff in zip(discriminations, difficulties):
        base_color = '#482878' if disc < 0 else '#35b779'
        intensity = 1
        color = mcolors.to_rgba(base_color, intensity)
        colors.append(color)
    scatter = ax.scatter(discriminations, difficulties, s=point_sizes, c=colors)

    plt.xlabel(r'Discrimination', fontsize=font_size, family='Times New Roman', color='black')
    plt.ylabel(r'Difficulty', fontsize=font_size, family='Times New Roman', color='black')

    ax.set_ylim(0, 1)
    plt.xlim(0, 9)
    ticks = np.arange(0, 9, 2)
    plt.xticks(ticks)

    if discriminations.min() < 0:
        x_min = int(discriminations.min()) - 1
    else:
        x_min = int(discriminations.min())

    if discriminations.max() > 0:
        x_max = int(discriminations.max()) + 1
    else:
        x_max = int(discriminations.max())

    ax.set_xlim(x_min, x_max)

plot_discriminations_difficulties(b4.discriminations, b4.difficulties, new_pij, font_size=10, font_ann_size=9, base_point_size=50)
```

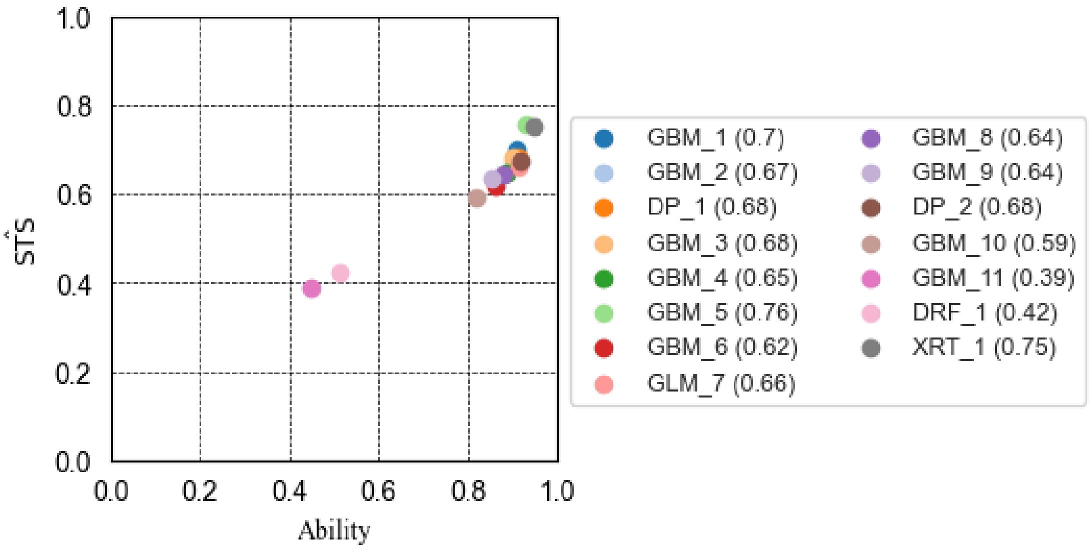


```
[9]: fairness_model = normalized_df.apply(np.mean,axis=0).to_numpy()
```

2 Figure 9

```
[10]: def plot_abilities_fairness(abilities, fairness_model, font_size=10):
          plt.rcParams["font.family"] = "Times New Roman"

          sns.set_style('whitegrid')
          fig, ax = plt.subplots(figsize=(3, 3))
          ax.grid(color='black', linestyle='--', linewidth=0.5)

          ax.spines['bottom'].set_color('black')
          ax.spines['left'].set_color('black')
          ax.spines['right'].set_color('black')
          ax.spines['top'].set_color('black')
          ax.xaxis.label.set_color('black')
          ax.yaxis.label.set_color('black')
          ax.tick_params(axis='x', colors='black')
          ax.tick_params(axis='y', colors='black')

          colors = sns.color_palette('tab20', n_colors=48)

          for i in range(fairness_model.shape[0]):
```

```
        plt.scatter(abilities[i], fairness_model[i], label=f'{pij.columns[i]}␣
 ↪({round(fairness_model[i], 2)})', color=colors[i])

        plt.xlabel(r'Ability', fontsize=font_size, family='Times New Roman',␣
 ↪color='black')
    plt.ylabel(r'$\hat{\text{STS}}$', fontsize=font_size, family='Times New␣
 ↪Roman', color='black')
    plt.legend(title='', fontsize=9, bbox_to_anchor=(1, 0.8), loc='upper left',␣
 ↪ncol=2)

    ticks = np.arange(0, 1.1, 0.2)
    plt.xlim(0, 1)
    plt.ylim(0, 1)
    plt.xticks(ticks)
    plt.yticks(ticks)

    plt.show()

plot_abilities_fairness(b4.abilities, fairness_model, font_size=10)
```



[11]:
```
def f(theta,delta_j,a_j):
    term1 = (delta_j / (1 - delta_j)) ** a_j
    term2 = (theta / (1 - theta)) ** (-a_j - 1)
    numerator = a_j * term1 * term2
    denominator = (1 + term1 * (theta / (1 - theta)) ** -a_j) ** 2
    return numerator / denominator * (1 / (1 - theta) ** 2)
```

[12]:

```
abilities = np.linspace(0.001, 0.999, 1000)
```

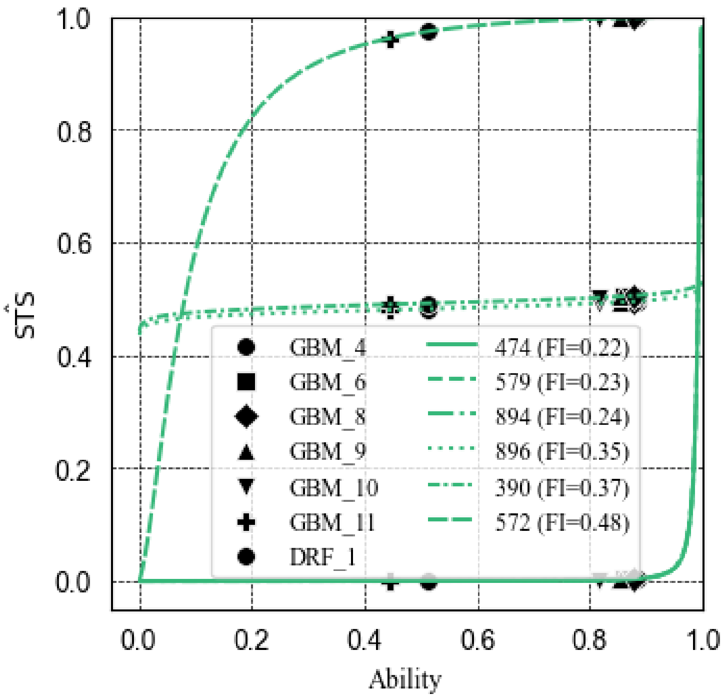
3 Figure 10(b)

[13]:

```
plt.figure(figsize=(4, 4))
plt.rcParams["font.family"] = "Times New Roman"

sns.set_style('whitegrid')
fig, ax = plt.subplots(figsize=(4, 4))
ax.grid(color='black', linestyle='--', linewidth=0.5)

ax.spines['bottom'].set_color('black')
ax.spines['left'].set_color('black')
ax.spines['right'].set_color('black')
ax.spines['top'].set_color('black')
ax.xaxis.label.set_color('black')
ax.yaxis.label.set_color('black')
ax.tick_params(axis='x', colors='black')
ax.tick_params(axis='y', colors='black')

markers = ['o', 's', 'D', '^', 'v', 'P']
num_markers = len(markers)
colors = ['red', 'blue', 'green', 'orange', 'purple', 'brown']
num_colors = len(colors)
linestyles = ['-', '--', '-.', ':', (0, (3, 1, 1, 1)),
              (0, (5, 1)), (0, (5, 10)), (0, (1, 1)),
              (0, (3, 5, 1, 5)), (0, (3, 10, 1, 10))]
num_linestyles = len(linestyles)
i = 0
j = 0
id = 0
list = [0.25,0.11,0.02,0.76,0.86]

added_labels = set()

for index in [473,578,893,895,389,571]:
    total_f_theta = 0
    for x_idx, x in enumerate([4,6,8,9,11,12,13]):
        marker = markers[x_idx % num_markers]
        fairness = ICC_function(b4.abilities[x], b4.difficulties[index], b4.
 ↪discriminations[index])
        f_theta = abs(f(b4.abilities[x], b4.difficulties[index], b4.
 ↪discriminations[index]))
        total_f_theta += f_theta
        total_f_theta = round(total_f_theta, 2)
```

```python
        label = f'{pij.columns[x]}'
        if label not in added_labels:
            sns.scatterplot(x=[b4.abilities[x]], y=[fairness], marker=marker, s=50, color='black', label=label)
            added_labels.add(label)
        else:
            sns.scatterplot(x=[b4.abilities[x]], y=[fairness], marker=marker, s=50, color='black')


        i += 1
    linestyle = linestyles[j % num_linestyles]
    fairness_2 = ICC_function(abilities, b4.difficulties[index], b4.discriminations[index])
    if b4.discriminations[index]>0:
        plt.plot(abilities, fairness_2, label=f'{index+1} (FI={total_f_theta})',color='#35b779', linestyle=linestyle)
    else:
        plt.plot(abilities, fairness_2, label=f'{index+1} (FI={total_f_theta})',color='#482878', linestyle=linestyle)
    j += 1
    id += 1

plt.xlim(-0.05, 1)
plt.ylim(-0.05, 1)

plt.gca().set_aspect('equal', adjustable='box')

legend = plt.legend(loc='upper left', fontsize=9, bbox_to_anchor=(0.15, 0.5), ncol=2)

for text in legend.get_texts():
    text.set_fontname('Times New Roman')
    text.set_color('black')

plt.xlabel(r'Ability', fontsize=10, family='Times New Roman')
plt.ylabel(r'$\hat{\text{STS}}$', fontsize=10, family='Times New Roman')

plt.grid(True)
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```



[14]:

```
from irt_special import Beta3
```

[15]:

```
def ICC_function_special(abilities, difficulties):
    a = ((1 - abilities) / abilities)
    b = (difficulties / (1 - difficulties))
    c = a * b
    d = c
    return (1 / (d + 1))
```

[16]:

```
b4_special = Beta3(
        learning_rate=100,
        epochs=10000,
        n_respondents=normalized_df.shape[1],
        n_items=normalized_df.shape[0],
        n_workers=-1,
        random_seed=1,
    )
b4_special.fit(normalized_df.values)
```

```
100%|      | 10000/10000 [00:12<00:00, 800.86it/s]
```

[16]:

```
<irt_special.Beta3 at 0x33f9b64f0>
```

[17]:

```
new_pij = pd.DataFrame(index=range(1000), columns=range(15))

for i in range(1000):
    for j in range(15):
        alpha = (b4_special.abilities[j] / b4_special.difficulties[i])
        beta_val = ((1 - b4_special.abilities[j]) / (1 - b4_special.
 ↪difficulties[i]))
        new_pij.iloc[i, j] = (alpha)/(alpha+beta_val)
```

4 Table 4

[18]:

```
# Individual 572
delta_j_values = b4_special.difficulties[571]
Delta_j_values = delta_j_values / (1 - delta_j_values)

log_Delta_j_values = np.log(Delta_j_values)

print(round(log_Delta_j_values,2))
```

```
-3.05
```

[19]:

```
# GBM_4, GBM_6, GBM_8, GBM_9, GBM_10, GBM_11, DRF_1
for i in [4,6,8,9,11,12,13]:
    theta_i_values = b4_special.abilities[i]
    Theta_i_values = (1 - theta_i_values) / theta_i_values

    log_Telta_j_values = np.log(Theta_i_values)

    res = np.log(1-new_pij[i][571])-np.log(new_pij[i][571])

    print(round(log_Telta_j_values,2), round(res,2))
```

```
-1.41 -4.46
-1.11 -4.15
-1.31 -4.36
-1.24 -4.29
-0.9 -3.95
0.47 -2.58
0.3 -2.75
```

[ ]:

```

```