Fair-IRT_Adult_show

October 14, 2024

[1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.colors as mcolors

from scipy import stats
from matplotlib import rcParams

from irt import Beta3
import matplotlib.pyplot as plt
```

[2]:

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

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

[3]:

```
array = pij.values.flatten()

transformed_data, best_lambda = stats.boxcox(array)
transformed_array = transformed_data.reshape(pij.shape)

res = pd.DataFrame(transformed_array, index=pij.index, columns=pij.columns)

array = res.values

min_val = np.min(array)
max_val = np.max(array)
normalized_array = (array - min_val) / (max_val - min_val)

normalized_df = pd.DataFrame(normalized_array, index=pij.index, columns=pij.
 ↪columns)
```

[4]:

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

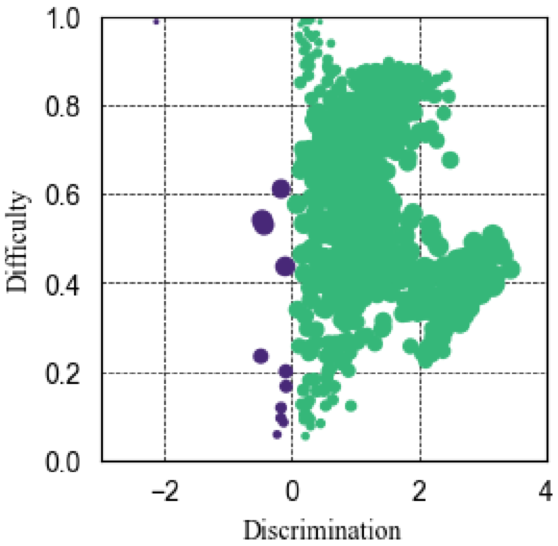
[5]:
```
b4 = Beta3(
        learning_rate=100,
        epochs=10000,
        n_respondents=normalized_df.shape[1],
        n_items=normalized_df.shape[0],
        n_workers=-1,
        random_seed=1,
    )
b4.fit(normalized_df.values)
```

```
100%|      | 10000/10000 [00:19<00:00, 522.46it/s]
```

[5]:
```
<irt.Beta3 at 0x3440f8880>
```

[6]:
```
new_pij = pd.DataFrame(index=range(1000), columns=range(24))

for i in range(1000):
    for j in range(24):
        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)
```

[7]:
```
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=(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')

    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)

    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)
    plt.savefig("Figure4a.pdf", format="pdf", bbox_inches='tight')
    plt.show()

plot_discriminations_difficulties(b4.discriminations, b4.difficulties, new_pij, font_size=10, font_ann_size=8, base_point_size=50)
```



[8]:

```
fairness_model = normalized_df.apply(np.mean,axis=0).to_numpy()
```

[9]:

```
def create_abilities_fairness_table(name, abilities, fairness_model):
    df = pd.DataFrame({
        'Model': name,
        'Ability': abilities,
        'STS': fairness_model
    })

    return df

df = create_abilities_fairness_table(pij.columns, b4.abilities, fairness_model)
print(df)
```

```
    Model   Ability       STS
0    SE_1  0.883245  0.793072
1   GBM_1  0.864788  0.764232
2   XRT_1  0.935142  0.864057
3   DRF_1  0.830784  0.694447
4    DL_1  0.776728  0.675320
5   GLM_1  0.473357  0.415491
6    SE_2  0.850524  0.752932
7   GBM_2  0.809756  0.701128
8   XRT_2  0.945343  0.864091
9   DRF_2  0.791173  0.653765
10   DL_2  0.714843  0.624717
11  GLM_2  0.481589  0.437398
12   SE_3  0.809707  0.693970
13  GBM_3  0.845318  0.731784
14   DL_3  0.541319  0.451491
15  XRT_3  0.838186  0.714750
16  DRF_3  0.613332  0.491743
17  GLM_3  0.435998  0.380779
18   SE_4  0.850991  0.719303
19  GBM_4  0.912202  0.771445
20  XRT_4  0.821233  0.690483
21  DRF_4  0.545751  0.442608
22   DL_4  0.618056  0.550603
23  GLM_4  0.434619  0.416322
```

[10]:

```
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)
```

[11]:

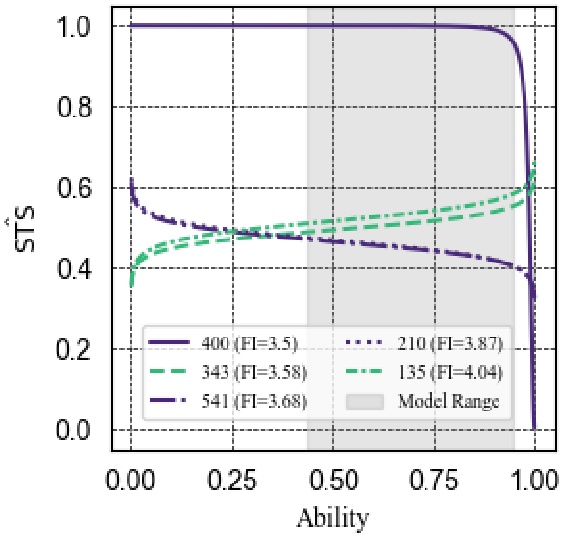
```
abilities = np.linspace(0.001, 0.999, 1000)
```

[12]:

```
min_ability = np.min(b4.abilities)
max_ability = np.max(b4.abilities)

plt.figure(figsize=(3, 3))
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')

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 [399,342,540,209,134]:
    total_f_theta = 0

    for i in range(b4.abilities.shape[0]):
        f_theta = abs(f(b4.abilities[i], b4.difficulties[index], b4.
 ↪discriminations[index]))
        total_f_theta += f_theta

    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={round(total_f_theta, 2)})',color='#35b779', linestyle=linestyle)
    else:
        plt.plot(abilities, fairness_2, label=f'{index+1}␣
 ↪(FI={round(total_f_theta, 2)})',color='#482878', linestyle=linestyle)
    j += 1
    id += 1


plt.fill_betweenx(np.arange(-0.05, 1.15, 0.1), min_ability, max_ability,␣
 ↪color='gray', alpha=0.2, label='Model Range')

plt.xlim(-0.05, 1.05)
plt.ylim(-0.05, 1.05)


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

legend = plt.legend(loc='upper left', fontsize=7, bbox_to_anchor=(0.05, 0.3),␣
 ↪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'$\hat{\mathrm{STS}}$', fontsize=10, family='Times New Roman')

plt.grid(True)
plt.savefig("Figure4b.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

```
<Figure size 300x300 with 0 Axes>
```



[13]:

```python
from irt_special import Beta3
```

[14]:

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

[15]:

```python
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 [00:13<00:00, 724.11it/s]
```

[15]:

```
<irt_special.Beta3 at 0x343fb4f10>
```

[16]:

```python
new_pij = pd.DataFrame(index=range(1000), columns=range(24))

for i in range(1000):
    for j in range(24):
        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)
```

[17]:

```
def generate_table(b4_special, new_pij, index):
    delta_j_values = b4_special.difficulties[index]
    Delta_j_values = delta_j_values / (1 - delta_j_values)
    log_Delta_j_values = np.log(Delta_j_values)

    print(f"log_Delta_j_values (for difficulty at index {index}):␣
 ↪{round(log_Delta_j_values, 3)}")

    results = []

    for i in range(b4_special.abilities.shape[0]):
        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][index]) - np.log(new_pij[i][index])

        results.append({
            "log_Telta_j_values": round(log_Telta_j_values, 3),
            "res": round(res, 3)
        })

    df = pd.DataFrame(results)
    return df

generate_table(b4_special, new_pij, 342)
```

```
log_Delta_j_values (for difficulty at index 342): 1.236
```

[17]:

```
    log_Telta_j_values    res
0               -2.163 -0.928
1               -1.977 -0.742
2               -2.863 -1.627
3               -1.599 -0.364
4               -1.398 -0.162
5               -0.049  1.187
6               -1.863 -0.628
7               -1.549 -0.314
8               -2.922 -1.687
9               -1.327 -0.091
10              -1.131  0.105
11              -0.173  1.063
12              -1.541 -0.305
13              -1.769 -0.534
```

```
14              -0.273  0.962
15              -1.705 -0.469
16              -0.467  0.768
17               0.200  1.435
18              -1.776 -0.540
19              -2.286 -1.050
20              -1.545 -0.310
21              -0.236  1.000
22              -0.748  0.488
23              -0.020  1.216
```

```
[ ]:
```