FairIRT_Adult_es_new

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_ES.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))
```

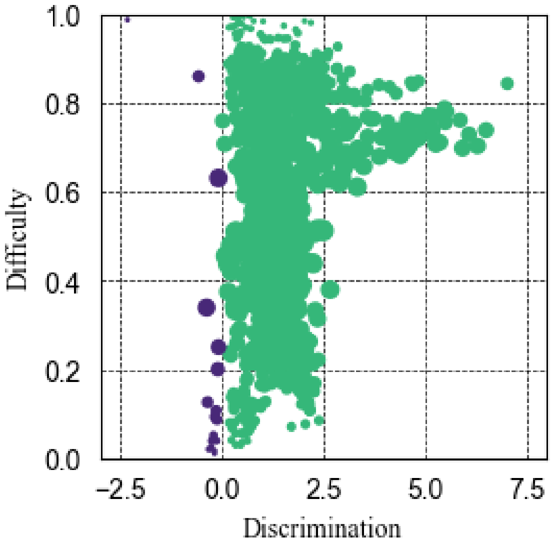
[5]:
```
b4 = Beta3(
        learning_rate=100,
        epochs=10000,
        n_respondents=normalized_df.shape[1],
        n_items=normalized_df.shape[0],
        n_workers=-1,
        random_seed=1,
    )
b4.fit(normalized_df.values)
```

```
100%|      | 10000/10000 [00:18<00:00, 548.75it/s]
```

[5]:
```
<irt.Beta3 at 0x3071c0df0>
```

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

for i in range(1000):
    for j in range(14):
        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("Figure7a.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)
df
```

[9]:

```
     Model  Ability       STS
0    GBM_3  0.821726  0.690290
1    GBM_2  0.814382  0.684904
2    GBM_5  0.808179  0.670759
3    GBM_4  0.875418  0.751162
4   GBM_g4  0.815006  0.669351
5   GBM_g2  0.757357  0.586212
6    GBM_1  0.861996  0.739535
7   GBM_g3  0.785758  0.643618
8   GBM_g1  0.786056  0.639942
9   GBM_g5  0.940807  0.841832
10   XRT_1  0.930655  0.824125
11   DRF_1  0.728466  0.568885
12    DL_1  0.680123  0.552305
13   GLM_1  0.283573  0.245493
```

[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)
```

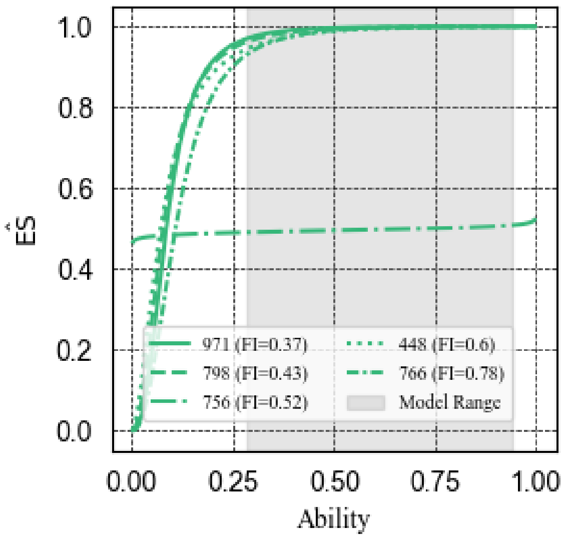
[12]:

```
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 [970,797,755,447,765]:
    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'$\hat{\mathrm{ES}}$', fontsize=10, family='Times New Roman')

plt.grid(True)
plt.savefig("Figure7b.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

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



[13]:
```
from irt_special import Beta3
```

[14]:

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

[15]:

```
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, 727.32it/s]
```

[15]:

```
<irt_special.Beta3 at 0x33fbaf490>
```

[16]:

```
new_pij = pd.DataFrame(index=range(1000), columns=range(14))

for i in range(1000):
    for j in range(14):
        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, 970)
```

```
log_Delta_j_values (for difficulty at index 970): -3.768
```

[17]:

```
    log_Telta_j_values    res
0               -1.681 -5.449
1               -1.641 -5.410
2               -1.562 -5.331
3               -2.113 -5.881
4               -1.569 -5.337
5               -1.048 -4.817
6               -2.026 -5.795
7               -1.396 -5.164
8               -1.375 -5.144
9               -2.946 -6.714
10              -2.869 -6.637
11              -0.937 -4.705
12              -0.799 -4.568
13               1.225 -2.543
```

[17]:

```

```