Fair-IRT_Simulated

October 14, 2024

[1]:
```python
from irt import Beta3
import numpy as np
import pandas as pd
import seaborn as sns
```

[2]:
```python
n_rows = 50 # number of individual
n_cols = 20 # number of prediction models

np.random.seed(0)
abil = np.random.rand(n_cols)
diff = np.random.rand(n_rows)

discr = np.random.normal(1,1,size = n_rows)

pij = pd.DataFrame(index=range(n_rows), columns=range(n_cols))

for i in range(n_rows):
    for j in range(n_cols):
        alpha = (abil[j] / diff[i]) ** discr[i]
        beta_val = ((1 - abil[j]) / (1 - diff[i])) ** discr[i]
        pij.iloc[i, j] = (alpha)/(alpha+beta_val)
```

[3]:
```python
pij
```

[3]:

	0	1	2	3	4	5	6 \
0	0.028596	0.056196	0.035209	0.02817	0.017724	0.041816	0.018711
1	0.155407	0.341303	0.201494	0.152456	0.082246	0.247106	0.088584
2	0.59232	0.758894	0.647801	0.588229	0.45919	0.691188	0.474274
3	0.198231	0.388577	0.247984	0.194968	0.113709	0.295321	0.121385
4	0.691305	0.744842	0.708288	0.690066	0.650685	0.721997	0.655393
5	0.439889	0.554854	0.474845	0.4374	0.362951	0.504145	0.371365
6	0.656944	0.708295	0.673076	0.655772	0.618865	0.6862	0.623246
7	0.155537	0.226603	0.175056	0.154207	0.1177	0.192735	0.121533
8	0.505065	0.538847	0.515389	0.504324	0.481546	0.523973	0.484197
9	0.403084	0.285147	0.365117	0.405859	0.493294	0.334725	0.482981
10	0.807595	0.907871	0.844855	0.804677	0.698766	0.871188	0.712661
11	0.34977	0.453526	0.380418	0.347615	0.284682	0.406688	0.291655
12	0.441701	0.333798	0.407669	0.444167	0.520775	0.379989	0.511831

13	0.470928	0.719881	0.551572	0.465152	0.298768	0.616861	0.316469
14	0.595039	0.611122	0.599969	0.594685	0.58374	0.604056	0.58502
15	0.425953	0.61399	0.483561	0.421879	0.304024	0.531913	0.316819
16	0.389361	0.690685	0.483088	0.382861	0.210671	0.562315	0.227475
17	0.421436	0.622802	0.483195	0.417077	0.292037	0.535078	0.305477
18	0.003633	0.016901	0.005818	0.003512	0.00124	0.008601	0.0014
19	0.533195	0.490692	0.520252	0.534122	0.562484	0.509457	0.559197
20	0.747352	0.891002	0.801329	0.74313	0.593446	0.839258	0.612521
21	0.535329	0.591462	0.552615	0.534085	0.4957	0.566909	0.500178
22	0.479334	0.502735	0.486468	0.478823	0.463121	0.49241	0.464947
23	0.775498	0.824175	0.791294	0.774333	0.736432	0.803807	0.741051
24	0.415165	0.539012	0.452536	0.412516	0.334166	0.484075	0.342927
25	0.367227	0.555132	0.422969	0.363344	0.254248	0.47098	0.265795
26	0.437632	0.408423	0.428666	0.438276	0.458198	0.421238	0.455869
27	0.982081	0.995421	0.988157	0.981543	0.954629	0.991624	0.959085
28	0.805773	0.923095	0.851556	0.802109	0.664759	0.88253	0.683058
29	0.400122	0.311386	0.372028	0.402165	0.466334	0.349276	0.458758
30	0.446229	0.830289	0.582798	0.436498	0.187069	0.688309	0.209529
31	0.7828	0.96711	0.872404	0.77489	0.455989	0.920953	0.497017
32	0.000096	0.000465	0.000155	0.000093	0.000032	0.000232	0.000036
33	0.874797	0.926796	0.893347	0.873366	0.822155	0.906897	0.828867
34	0.473006	0.460243	0.469108	0.473286	0.481897	0.465866	0.480893
35	0.977887	0.994925	0.985846	0.977171	0.940165	0.990259	0.946405
36	0.435186	0.542865	0.467855	0.432861	0.363241	0.495272	0.371122
37	0.9447	0.988445	0.965427	0.942829	0.847964	0.976764	0.863541
38	0.598626	0.781693	0.660811	0.594013	0.447983	0.708744	0.465018
39	0.932008	0.982888	0.954999	0.929994	0.835124	0.968287	0.850014
40	0.925895	0.970927	0.944022	0.924405	0.863199	0.955842	0.872049
41	0.980077	0.994134	0.986253	0.979541	0.954183	0.989921	0.958241
42	0.387973	0.568717	0.44216	0.384175	0.27592	0.488396	0.287539
43	0.996478	0.999531	0.998095	0.996319	0.985825	0.998858	0.987875
44	0.859023	0.932392	0.88659	0.856844	0.775496	0.905832	0.786449
45	0.736616	0.885409	0.792236	0.732279	0.57989	0.831527	0.599166
46	0.02132	0.14973	0.03958	0.020387	0.005073	0.065472	0.005971
47	0.300783	0.250557	0.284863	0.301945	0.338851	0.272006	0.334438
48	0.596627	0.548688	0.582154	0.597659	0.628938	0.569997	0.625344
49	0.991836	0.998029	0.994703	0.991579	0.978288	0.996308	0.980542

	7	8	9	10	11	12	13 \
0	0.158917	0.37044	0.015106	0.081801	0.026509	0.030778	0.21995
1	0.738799	0.937575	0.065928	0.483691	0.140963	0.170597	0.835916
2	0.917883	0.974928	0.415348	0.830452	0.571522	0.612244	0.945488
3	0.749204	0.931939	0.093541	0.521673	0.182156	0.214873	0.836298
4	0.818439	0.873609	0.636664	0.772575	0.685014	0.697359	0.839761
5	0.726618	0.848442	0.338695	0.618817	0.427343	0.45217	0.775573
6	0.781823	0.840129	0.605868	0.735543	0.651002	0.662678	0.803964
7	0.385427	0.570049	0.107029	0.276469	0.148912	0.162212	0.449488

8	0.593326	0.644774	0.473752	0.558031	0.50132	0.508708	0.611758
9	0.144046	0.067167	0.52363	0.227899	0.417169	0.389532	0.11093
10	0.975565	0.993715	0.65579	0.941371	0.792501	0.821464	0.984814
11	0.628177	0.772709	0.264762	0.515397	0.338943	0.360455	0.683703
12	0.191561	0.101876	0.546986	0.278411	0.454194	0.429616	0.154529
13	0.935958	0.987774	0.250168	0.82497	0.441842	0.49942	0.963898
14	0.636958	0.661625	0.57997	0.620217	0.593247	0.59678	0.645742
15	0.847367	0.949967	0.268142	0.711009	0.405472	0.446128	0.895431
16	0.945727	0.992453	0.166596	0.820577	0.356952	0.421863	0.972622
17	0.863301	0.959405	0.254596	0.725034	0.399538	0.443043	0.909543
18	0.179285	0.727026	0.000867	0.040068	0.003062	0.00429	0.345279
19	0.421585	0.35646	0.57212	0.4664	0.537878	0.528633	0.398191
20	0.977409	0.995533	0.535885	0.935985	0.725541	0.767442	0.987195
21	0.677988	0.752683	0.482534	0.622624	0.529038	0.541438	0.705657
22	0.541024	0.578226	0.457754	0.516123	0.476749	0.48185	0.554209
23	0.885475	0.926737	0.72254	0.848066	0.769566	0.781166	0.901927
24	0.725999	0.855593	0.309065	0.608692	0.40183	0.42826	0.778686
25	0.814032	0.937672	0.222241	0.659114	0.347782	0.386571	0.87119
26	0.361992	0.318695	0.46506	0.391988	0.440893	0.434464	0.346431
27	0.99952	0.999948	0.938698	0.997913	0.979193	0.984512	0.99978
28	0.98561	0.997375	0.608151	0.956597	0.786722	0.823042	0.992081
29	0.192967	0.11328	0.488697	0.265809	0.410484	0.390126	0.160872
30	0.989485	0.999421	0.131798	0.932099	0.397617	0.494541	0.996202
31	0.998912	0.999963	0.340598	0.989868	0.740756	0.818756	0.99967
32	0.006152	0.073232	0.000022	0.001146	0.00008	0.000114	0.01496
33	0.971054	0.9887	0.801217	0.946774	0.867417	0.881637	0.97918
34	0.439439	0.41918	0.48485	0.452964	0.47442	0.471631	0.432273
35	0.999556	0.99996	0.917663	0.997829	0.97403	0.981099	0.99981
36	0.707042	0.82902	0.340493	0.603326	0.423464	0.446661	0.755271
37	0.999167	0.999938	0.793667	0.99537	0.93462	0.953088	0.999666
38	0.93769	0.984073	0.398716	0.85527	0.575146	0.621046	0.961174
39	0.99834	0.999835	0.784474	0.992387	0.921252	0.941151	0.999264
40	0.99406	0.998777	0.834155	0.983223	0.918079	0.932835	0.996594
41	0.999224	0.999894	0.939965	0.997101	0.977222	0.982517	0.999616
42	0.814172	0.934536	0.243474	0.667212	0.368921	0.406846	0.869169
43	0.999983	0.999999	0.977604	0.999852	0.9956	0.997164	0.999995
44	0.98135	0.994935	0.741008	0.956532	0.847723	0.869335	0.988195
45	0.976121	0.995272	0.52197	0.932516	0.714238	0.757277	0.986456
46	0.84412	0.993691	0.003139	0.367892	0.016995	0.026554	0.946695
47	0.181106	0.128379	0.351996	0.224465	0.306679	0.29511	0.160821
48	0.468533	0.391214	0.639428	0.520792	0.601834	0.591539	0.440911
49	0.99981	0.999981	0.970076	0.999128	0.990453	0.992991	0.999916

	14	15	16	17	18	19
0	0.001997	0.002475	0.00056	0.103702	0.076121	0.133732
1	0.003524	0.00483	0.000544	0.579056	0.455098	0.6776
2	0.070618	0.087799	0.018447	0.867114	0.818035	0.899545

3	0.006686	0.008904	0.001221	0.607557	0.495508	0.694994
4	0.448802	0.468928	0.335258	0.789391	0.767368	0.806867
5	0.118705	0.134297	0.055228	0.658344	0.60666	0.699524
6	0.43622	0.454152	0.335024	0.75232	0.730389	0.769984
7	0.030361	0.034831	0.013352	0.312213	0.266289	0.354404
8	0.378312	0.388102	0.322683	0.570387	0.554323	0.583919
9	0.834356	0.810892	0.928783	0.19536	0.238336	0.163682
10	0.139051	0.173334	0.033393	0.956594	0.935923	0.968997
11	0.0932	0.105028	0.044841	0.555396	0.503377	0.598694
12	0.818964	0.797322	0.91172	0.245668	0.288703	0.212676
13	0.015314	0.021055	0.002284	0.87477	0.807335	0.915127
14	0.532044	0.537157	0.501732	0.626072	0.61846	0.632489
15	0.038834	0.048537	0.010083	0.765525	0.693326	0.816885
16	0.005304	0.007762	0.000553	0.879246	0.799191	0.924031
17	0.031053	0.039535	0.007251	0.781293	0.706574	0.833168
18	0.00001	0.000016	0.000001	0.069081	0.033951	0.122818
19	0.686258	0.674943	0.748432	0.45072	0.471101	0.433534
20	0.057573	0.07695	0.009639	0.955197	0.928888	0.969991
21	0.325084	0.340597	0.241704	0.642326	0.616652	0.663517
22	0.391697	0.398542	0.352191	0.524795	0.513528	0.534347
23	0.518592	0.541843	0.382903	0.862093	0.843652	0.876291
24	0.095572	0.109603	0.041123	0.651804	0.595432	0.696625
25	0.030287	0.037973	0.00777	0.719787	0.639785	0.77848
26	0.551356	0.542284	0.604075	0.381454	0.395157	0.369959
27	0.221784	0.302816	0.023088	0.998747	0.997569	0.999287
28	0.067085	0.090508	0.010434	0.970317	0.951433	0.980583
29	0.746248	0.723087	0.855833	0.238628	0.274307	0.210926
30	0.000824	0.00143	0.000032	0.964036	0.91829	0.982485
31	0.001193	0.002264	0.000027	0.995324	0.987246	0.998015
32	0.0	0.0	0.0	0.002057	0.000962	0.003919
33	0.419484	0.464287	0.197963	0.956855	0.943352	0.965856
34	0.521905	0.51799	0.544966	0.448253	0.454374	0.443066
35	0.13068	0.191619	0.010091	0.998749	0.99744	0.99932
36	0.128757	0.144342	0.063344	0.641035	0.591787	0.680696
37	0.0352	0.056348	0.001981	0.997448	0.994469	0.998679
38	0.050056	0.06444	0.010709	0.891051	0.842826	0.921296
39	0.054633	0.082205	0.004321	0.995512	0.991088	0.997499
40	0.226625	0.283577	0.047241	0.988291	0.981326	0.99214
41	0.304507	0.389929	0.044733	0.998166	0.996676	0.998894
42	0.037268	0.046193	0.010199	0.724597	0.648935	0.780272
43	0.112775	0.190752	0.003296	0.99993	0.999815	0.999969
44	0.21237	0.257141	0.058042	0.967528	0.952605	0.976542
45	0.054655	0.07312	0.009131	0.952712	0.925067	0.968296
46	0.000007	0.000014	0.0	0.558275	0.315815	0.748336
47	0.529614	0.510374	0.639711	0.208603	0.229369	0.192078
48	0.757642	0.746463	0.816692	0.502617	0.526217	0.482557
49	0.343218	0.447097	0.038103	0.999486	0.998979	0.999714

[4]:
```python
normalized_df = pij.astype(np.float32)
```

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

def loss_function(b4, normalized_df):
        loss = []
        for i in range(normalized_df.shape[0]):
            for j in range(normalized_df.shape[1]):
                pij_predicted = ICC_function(b4.abilities[j], b4.
 ↪difficulties[i], b4.discriminations[i])
                res = np.mean(abs(pij_predicted-normalized_df.iloc[i, j]))
                loss.append(res)
        return np.mean(loss)

def evaluate_model(learning_rate, epochs, normalized_df):
    b4 = Beta3(
        learning_rate=learning_rate,
        epochs=epochs,
        n_respondents=normalized_df.shape[1],
        n_items=normalized_df.shape[0],
        n_workers=-1,
        random_seed=1
    )
    b4.fit(normalized_df.values)
    loss = loss_function(b4, normalized_df)
    return loss
```

[6]:
```python
b4 = Beta3(
        learning_rate=1,
        epochs=5000,
        n_respondents=normalized_df.shape[1],
        n_items=normalized_df.shape[0],
        n_workers=-1,
        random_seed=1,
    )
b4.fit(normalized_df.values)
```

```
100%|      | 5000/5000 [00:07<00:00, 673.41it/s]
```

[6]:
```
<irt.Beta3 at 0x343387220>
```

[7]:
```
loss = loss_function(b4, normalized_df)
loss
```

[7]: 0.019228961147294065

[8]:
```
new_pij = pd.DataFrame(index=range(n_rows), columns=range(n_cols))

for i in range(n_rows):
    for j in range(n_cols):
        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)
```

[9]:
```
new_pij
```

[9]:

	0	1	2	3	4	5	6	\
0	0.021762	0.050046	0.02845	0.02134	0.011658	0.035239	0.012496	
1	0.158693	0.34661	0.207785	0.155519	0.080993	0.254781	0.087495	
2	0.594922	0.759211	0.652019	0.590646	0.455641	0.695036	0.471131	
3	0.205793	0.39409	0.257872	0.202328	0.116319	0.30552	0.124258	
4	0.679829	0.744208	0.70128	0.678237	0.627427	0.717914	0.63341	
5	0.440452	0.559364	0.478273	0.437723	0.356398	0.508884	0.365386	
6	0.643549	0.705241	0.663867	0.64205	0.594736	0.679778	0.600258	
7	0.128466	0.21561	0.152397	0.126857	0.08524	0.174195	0.089287	
8	0.500935	0.536774	0.51238	0.500103	0.474533	0.521573	0.477455	
9	0.40231	0.281219	0.361507	0.405338	0.50086	0.330061	0.489801	
10	0.819193	0.92323	0.860819	0.815817	0.688367	0.888186	0.705182	
11	0.349139	0.458888	0.383006	0.34673	0.276902	0.411104	0.284441	
12	0.441652	0.329966	0.404792	0.444363	0.528538	0.375894	0.5189	
13	0.479456	0.704786	0.55513	0.473947	0.313646	0.614332	0.330601	
14	0.543428	0.575849	0.553817	0.542671	0.519315	0.562138	0.521993	
15	0.41726	0.630023	0.485547	0.412384	0.274562	0.54097	0.288854	
16	0.403846	0.668775	0.489681	0.397765	0.232691	0.559385	0.248968	
17	0.426904	0.621856	0.489497	0.422418	0.293739	0.540159	0.307312	
18	0.00909	0.033761	0.013856	0.008815	0.003417	0.019412	0.003809	
19	0.519001	0.474142	0.504698	0.520041	0.55186	0.493192	0.548236	
20	0.746557	0.883471	0.79927	0.742388	0.594983	0.835356	0.613416	
21	0.532521	0.591979	0.551661	0.531125	0.488068	0.566943	0.492997	
22	0.487199	0.512319	0.495206	0.486617	0.468747	0.501647	0.470788	
23	0.768826	0.829249	0.789582	0.767262	0.715665	0.805254	0.721905	
24	0.411976	0.545339	0.454033	0.40896	0.3205	0.48836	0.330122	
25	0.368988	0.55862	0.427907	0.36483	0.248844	0.476898	0.260798	
26	0.505857	0.475731	0.496247	0.506556	0.528013	0.488521	0.525562	
27	0.965745	0.989325	0.976304	0.96482	0.921526	0.982425	0.928263	
28	0.802713	0.915388	0.847499	0.799098	0.664555	0.877159	0.682092	
29	0.402886	0.308745	0.371689	0.405189	0.477551	0.347346	0.469165	

30	0.465598	0.751615	0.56442	0.458423	0.258439	0.640671	0.278367
31	0.733909	0.894915	0.798023	0.728773	0.546084	0.840637	0.568759
32	0.001998	0.006371	0.002893	0.001945	0.000849	0.003895	0.000934
33	0.895435	0.946632	0.915263	0.893843	0.833633	0.928662	0.841706
34	0.50588	0.494675	0.502308	0.50614	0.514123	0.499436	0.51321
35	0.956432	0.986848	0.970135	0.955229	0.89884	0.978028	0.907596
36	0.43531	0.547508	0.47091	0.432742	0.356178	0.499768	0.364651
37	0.916406	0.97235	0.940816	0.914312	0.822982	0.955421	0.836407
38	0.60213	0.772974	0.662142	0.597618	0.454536	0.706967	0.470972
39	0.918108	0.974218	0.942907	0.915964	0.82094	0.957537	0.835044
40	0.922508	0.969344	0.942044	0.920873	0.85296	0.954295	0.862672
41	0.970697	0.990515	0.979495	0.96993	0.934336	0.984643	0.939852
42	0.390185	0.571925	0.447245	0.386131	0.271238	0.494229	0.283267
43	0.961403	0.988422	0.973607	0.960329	0.909596	0.980615	0.917523
44	0.864301	0.937958	0.893506	0.861933	0.771241	0.912827	0.783439
45	0.737755	0.882019	0.793463	0.733347	0.578122	0.831523	0.597429
46	0.072776	0.236505	0.108407	0.070657	0.027783	0.147392	0.030973
47	0.375217	0.402856	0.38395	0.374585	0.355372	0.391027	0.357548
48	0.552144	0.496921	0.534613	0.553415	0.592053	0.520452	0.587681
49	0.970965	0.990056	0.979318	0.970243	0.937432	0.984284	0.942451

	7	8	9	10	11	12	13 \
0	0.172438	0.391558	0.009544	0.078183	0.019707	0.023946	0.237362
1	0.73403	0.914327	0.064677	0.484287	0.143142	0.174998	0.817037
2	0.914267	0.966632	0.411658	0.827832	0.573125	0.615638	0.938339
3	0.7384	0.903911	0.0959	0.51953	0.188686	0.223393	0.812448
4	0.827826	0.878924	0.61	0.775897	0.671723	0.68756	0.847727
5	0.731216	0.835608	0.331117	0.623056	0.426667	0.453862	0.77263
6	0.789427	0.844196	0.578722	0.736509	0.635927	0.650847	0.810387
7	0.426155	0.626573	0.074512	0.279438	0.120491	0.136596	0.498117
8	0.593011	0.637443	0.466141	0.55643	0.496718	0.505011	0.609038
9	0.141393	0.07492	0.532632	0.224768	0.417725	0.387593	0.113419
10	0.982774	0.995135	0.637265	0.953745	0.801585	0.835029	0.988995
11	0.63776	0.76219	0.255914	0.522022	0.337016	0.361044	0.685424
12	0.187837	0.110666	0.556121	0.274867	0.455418	0.428432	0.156478
13	0.916001	0.974294	0.267658	0.801585	0.451629	0.506472	0.944406
14	0.62613	0.665475	0.511605	0.593497	0.53959	0.547132	0.640354
15	0.871475	0.954664	0.235985	0.73318	0.392744	0.441349	0.910297
16	0.920156	0.979627	0.190101	0.786817	0.373381	0.434014	0.950393
17	0.85327	0.942477	0.256726	0.718054	0.404316	0.449027	0.893695
18	0.227746	0.629071	0.002499	0.068152	0.007779	0.010564	0.354653
19	0.403739	0.348695	0.562252	0.449498	0.524264	0.513911	0.383792
20	0.971627	0.991592	0.540622	0.927431	0.724979	0.76632	0.98154
21	0.680905	0.745418	0.473904	0.62381	0.525443	0.53935	0.704852
22	0.552202	0.58444	0.462881	0.526174	0.48425	0.490049	0.563738
23	0.89882	0.935801	0.697252	0.856858	0.760828	0.776377	0.913757
24	0.738727	0.851019	0.293677	0.617467	0.396767	0.426837	0.784078

25	0.812605	0.922512	0.216547	0.659715	0.348134	0.389619	0.86133
26	0.428133	0.390028	0.535056	0.459153	0.509397	0.502436	0.414452
27	0.998386	0.999659	0.898814	0.994439	0.960807	0.969943	0.999071
28	0.980941	0.994622	0.611654	0.948894	0.783889	0.819705	0.987825
29	0.187864	0.11883	0.501728	0.262339	0.414596	0.391674	0.160306
30	0.956603	0.991177	0.2065	0.857529	0.429457	0.500877	0.975216
31	0.980924	0.995569	0.480111	0.940735	0.707263	0.758143	0.988624
32	0.039428	0.158365	0.000646	0.012048	0.001743	0.002279	0.065909
33	0.982658	0.993238	0.808576	0.963663	0.887151	0.902925	0.987581
34	0.47682	0.462219	0.516749	0.488488	0.507196	0.504609	0.471619
35	0.998127	0.999624	0.86945	0.993286	0.950005	0.961889	0.998942
36	0.712924	0.817472	0.332313	0.608165	0.42234	0.447928	0.753886
37	0.995589	0.999036	0.779588	0.985282	0.905302	0.925999	0.997431
38	0.92542	0.972852	0.407923	0.841978	0.579108	0.62396	0.947636
39	0.996203	0.999223	0.775167	0.986653	0.906717	0.927897	0.99784
40	0.993385	0.998148	0.821839	0.981896	0.913897	0.930073	0.995792
41	0.998468	0.999659	0.915749	0.994947	0.966607	0.974185	0.999102
42	0.812134	0.918859	0.238477	0.667377	0.369815	0.410243	0.85895
43	0.99836	0.999672	0.882858	0.994103	0.955661	0.966269	0.999075
44	0.98361	0.99465	0.733522	0.960583	0.851946	0.875403	0.988979
45	0.972628	0.992224	0.521439	0.927675	0.714947	0.758647	0.982456
46	0.734517	0.943395	0.020289	0.398032	0.062617	0.084018	0.839782
47	0.448173	0.486028	0.349146	0.418404	0.372019	0.378317	0.461597
48	0.409548	0.341693	0.604548	0.466366	0.558573	0.545915	0.384854
49	0.99824	0.999577	0.920633	0.994532	0.967125	0.97426	0.99894

	14	15	16	17	18	19
0	0.000855	0.001101	0.00029	0.105289	0.071594	0.143116
1	0.003773	0.005107	0.001034	0.581738	0.455863	0.679149
2	0.075627	0.092905	0.03043	0.865362	0.815478	0.897602
3	0.007932	0.010371	0.002515	0.605339	0.494144	0.690304
4	0.392344	0.414628	0.303125	0.796014	0.769794	0.816001
5	0.114494	0.12953	0.0663	0.664649	0.610589	0.706445
6	0.383797	0.403555	0.304368	0.756736	0.730433	0.777166
7	0.01379	0.016518	0.00636	0.329481	0.2658	0.387593
8	0.368398	0.378277	0.327568	0.569851	0.552514	0.584087
9	0.839546	0.816877	0.911757	0.190896	0.235375	0.159151
10	0.101681	0.131054	0.032205	0.967612	0.948752	0.97798
11	0.086756	0.098036	0.050773	0.565159	0.509396	0.610197
12	0.825866	0.804911	0.895751	0.240475	0.285414	0.207093
13	0.024555	0.032234	0.007567	0.852962	0.784334	0.895186
14	0.419881	0.429334	0.380293	0.6055	0.589988	0.618193
15	0.026378	0.033772	0.009058	0.792476	0.714061	0.844458
16	0.010801	0.01483	0.002768	0.84822	0.765899	0.896994
17	0.035997	0.045009	0.013643	0.774807	0.700024	0.825964
18	0.000059	0.000087	0.000011	0.10838	0.059354	0.173041
19	0.680207	0.668616	0.727091	0.432684	0.454407	0.414877

20	0.076361	0.098422	0.02464	0.948191	0.920086	0.964117
21	0.313467	0.328972	0.252127	0.645115	0.617524	0.667272
22	0.393871	0.400947	0.364218	0.535682	0.523408	0.545818
23	0.442914	0.470528	0.33089	0.87361	0.851655	0.889637
24	0.084098	0.097102	0.044694	0.664419	0.603355	0.71124
25	0.030623	0.03813	0.011854	0.722186	0.64033	0.780558
26	0.617556	0.609148	0.652582	0.447801	0.46246	0.435724
27	0.238872	0.308108	0.06591	0.996454	0.993663	0.99781
28	0.091718	0.118647	0.02877	0.964184	0.943398	0.975638
29	0.762382	0.739715	0.843319	0.23316	0.271246	0.204548
30	0.007816	0.011233	0.00165	0.906182	0.8399	0.94123
31	0.037449	0.05141	0.009373	0.960567	0.933405	0.974687
32	0.000025	0.000035	0.000006	0.018624	0.010611	0.029548
33	0.353242	0.403545	0.179751	0.9722	0.960741	0.979169
34	0.548084	0.544818	0.561966	0.484235	0.489724	0.47969
35	0.174157	0.23235	0.043199	0.99578	0.992318	0.997434
36	0.123017	0.138046	0.073814	0.648137	0.596245	0.68868
37	0.118046	0.158643	0.030095	0.990498	0.983295	0.99406
38	0.065972	0.082272	0.024914	0.87879	0.829702	0.90973
39	0.101976	0.139647	0.024146	0.991546	0.984767	0.994822
40	0.228762	0.283311	0.080143	0.987438	0.979876	0.991517
41	0.301613	0.377064	0.092825	0.996728	0.994268	0.997945
42	0.03808	0.046852	0.015481	0.726318	0.649099	0.781604
43	0.191481	0.253836	0.048101	0.996298	0.993249	0.997752
44	0.19534	0.23839	0.075878	0.971299	0.956807	0.979623
45	0.065393	0.085306	0.020126	0.948961	0.920094	0.965099
46	0.000438	0.000655	0.000078	0.526933	0.362342	0.660476
47	0.278974	0.285905	0.250596	0.429193	0.415282	0.440804
48	0.740219	0.727492	0.790025	0.445483	0.472459	0.423368
49	0.351207	0.427277	0.121115	0.99638	0.993836	0.997673

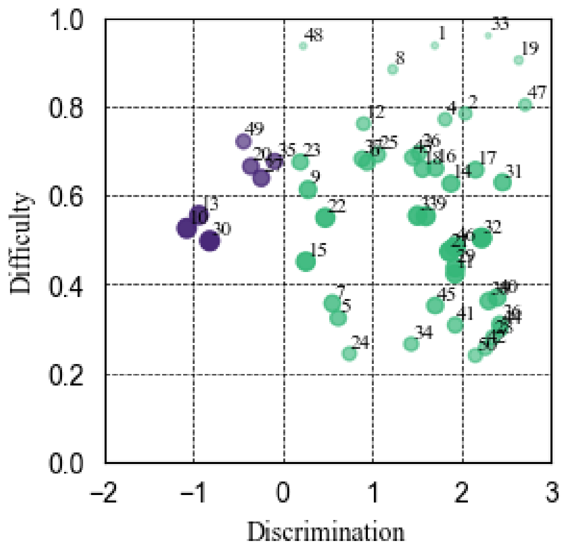
[10]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(pij, new_pij)
print("(MSE):", mse)
```

```
(MSE): 0.0014751571022834531
```

[11]:
```
from matplotlib import rcParams
import numpy as np
import matplotlib.colors as mcolors
import matplotlib.pyplot as plt

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']
```

```python
    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 - np.abs(diff - 0.5) * 1.5
        color = mcolors.to_rgba(base_color, intensity)
        colors.append(color)

    scatter = ax.scatter(discriminations, difficulties, s=point_sizes, c=colors)

    for i in range(normalized_df.shape[0]):
        ax.text(x=discriminations[i]+0.015, y=difficulties[i]+0.015, s=f'{normalized_df.index[i]+1}', fontsize=font_ann_size, family='Times New Roman', color='black')

    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("Figure2a.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)
```

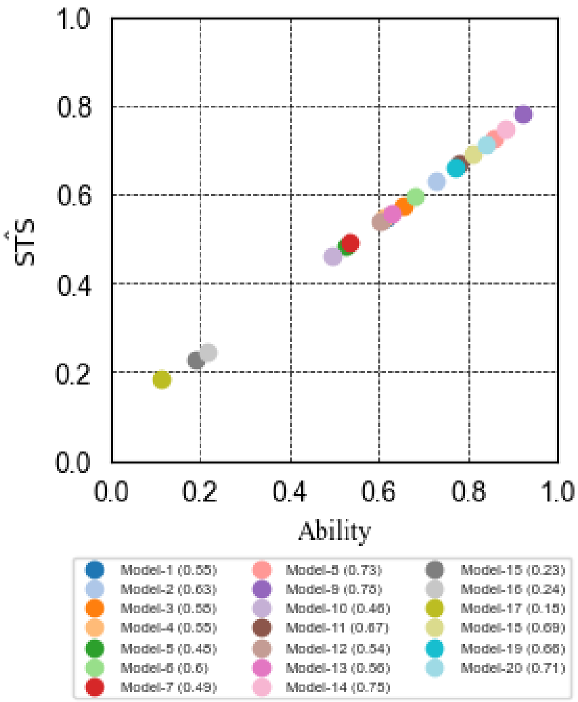


[12]:
```
fairness_model = new_pij.apply(np.mean,axis=0).to_numpy()
fairness_model
```

[12]:
```
array([0.54856206, 0.62876889, 0.57543071, 0.54656731, 0.48304063,
       0.59619941, 0.49049716, 0.7271804 , 0.7806597 , 0.46137368,
       0.66730743, 0.53840426, 0.55825125, 0.74860302, 0.22771779,
       0.24449113, 0.18270387, 0.69112036, 0.6599672 , 0.7140374 ])
```

[13]:
```
import numpy as np
import matplotlib.pyplot as plt

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=20)

    for i in range(fairness_model.shape[0]):
        plt.scatter(abilities[i], fairness_model[i], label=f'Model-{i+1} ({round(fairness_model[i], 2)})', color=colors[i])

    plt.xlabel(r'Ability', fontsize=font_size, family='Times New Roman', color='black')
    plt.ylabel(r'$\hat{\mathrm{STS}}$', fontsize=font_size, family='Times New Roman', color='black')
    plt.legend(title='', fontsize=5.5, bbox_to_anchor=(-0.1, -0.2), loc='upper left', ncol=3)

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

    plt.savefig("Figure3a.pdf", format="pdf", bbox_inches='tight')

    plt.show()

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

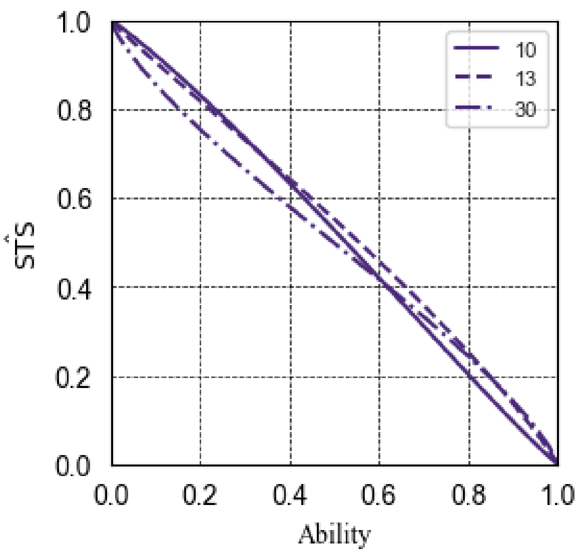


[14]:
```
abilities = np.linspace(0.001, 0.999, 1000)
```

[15]:
```
import matplotlib.pyplot as plt

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')

linestyles = ['-', '--', '-.', ':']
num_linestyles = len(linestyles)
i =0
```

```
for index in range (50):
    if b4.difficulties[index] < 0.6 and b4.difficulties[index] > 0.4:
        if b4.discriminations[index] < 0:
            fairness = ICC_function(abilities, b4.difficulties[index], b4.
↪discriminations[index])
            linestyle = linestyles[i % num_linestyles]
            plt.plot(abilities, fairness, label=f'{index+1}', color='#482878',␣
↪linestyle=linestyle)
            i = i+1

plt.xlim(0, 1)
plt.ylim(0, 1)

plt.legend(loc='upper right', fontsize=8)

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("Figure2b.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

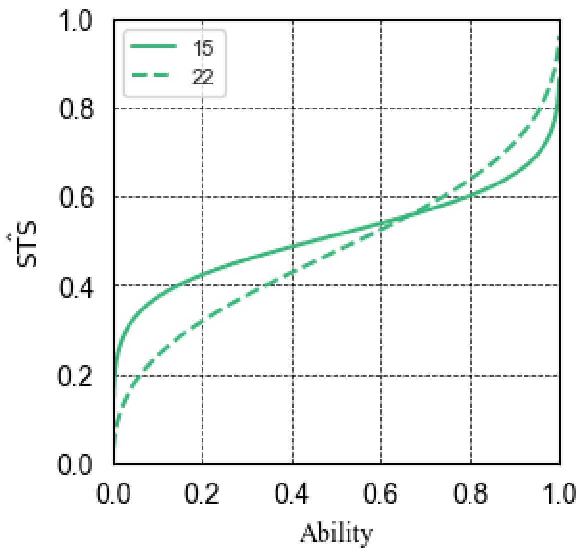
```
<Figure size 300x300 with 0 Axes>
```



[16]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(3, 3))
plt.rcParams["font.family"] = "Times New Roman"
```

```python
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')

linestyles = ['-', '--', '-.', ':']
num_linestyles = len(linestyles)
i = 0

for index in range (50):
    if b4.difficulties[index] < 0.6 and b4.difficulties[index] > 0.4:
        if b4.discriminations[index] < 1 and b4.discriminations[index] > 0:
            fairness = ICC_function(abilities, b4.difficulties[index], b4.discriminations[index])
            linestyle = linestyles[i % num_linestyles]
            plt.plot(abilities, fairness, label=f'{index+1}', color='#35b779', linestyle=linestyle)
            i = i+1

plt.xlim(0, 1)
plt.ylim(0, 1)

plt.legend(loc='upper left', fontsize=8)

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("Figure2c.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

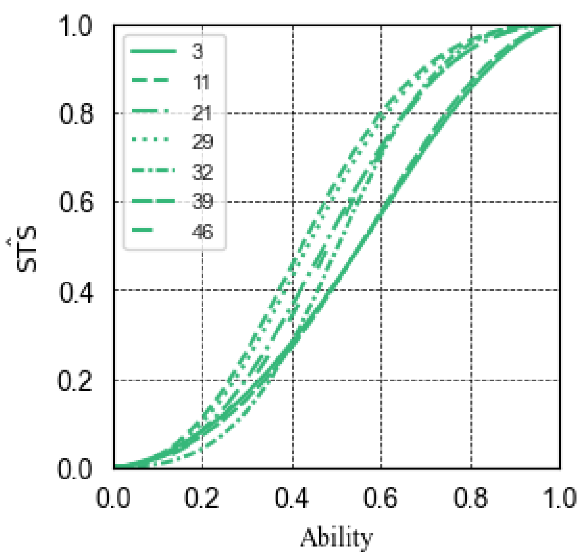
```
<Figure size 300x300 with 0 Axes>
```



[17]:
```
import matplotlib.pyplot as plt

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')

linestyles = ['-', '--', '-.', ':', (0, (3, 1, 1, 1)), (0, (5, 1)), (0, (5,␣
↪10))]
num_linestyles = len(linestyles)
i =0

for index in range (50):
    if b4.difficulties[index] < 0.6 and b4.difficulties[index] > 0.4:
        if b4.discriminations[index] > 1:
            fairness = ICC_function(abilities, b4.difficulties[index], b4.
↪discriminations[index])
```

```
            linestyle = linestyles[i % num_linestyles]
            plt.plot(abilities, fairness, label=f'{index+1}', color='#35b779',␣
↪linestyle=linestyle)
            i =i+1

plt.xlim(0, 1)
plt.ylim(0, 1)

plt.legend(loc='upper left', fontsize=8)

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("Figure2d.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

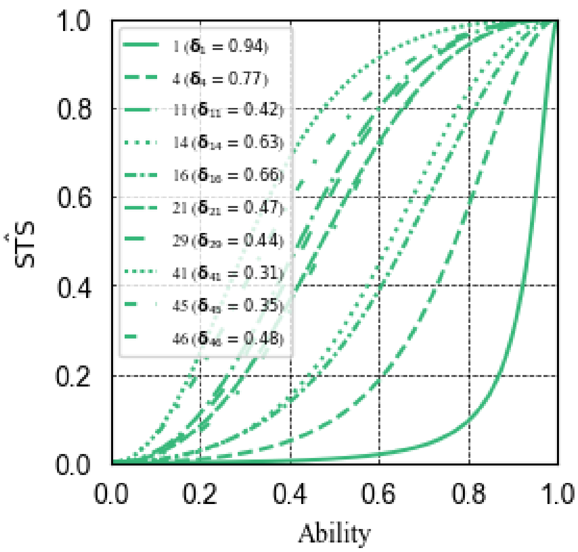
```
<Figure size 300x300 with 0 Axes>
```



[18]:
```
import matplotlib.pyplot as plt

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')
```

```python
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')

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

for index in range (50):
    if b4.discriminations[index] > 1.7 and b4.discriminations[index]<2:
        fairness = ICC_function(abilities, b4.difficulties[index], b4.discriminations[index])
        linestyle = linestyles[i % num_linestyles]
        plt.plot(abilities, fairness, label=f'{index+1} ($\mathbf{{\delta}}_{{{index+1}}}={b4.difficulties[index]:.2f}$)', color='#35b779', linestyle=linestyle)
        i += 1

plt.xlim(0, 1)
plt.ylim(0, 1)

legend = plt.legend(loc='upper left', fontsize=6, ncol=1)
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("Figure2e.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

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



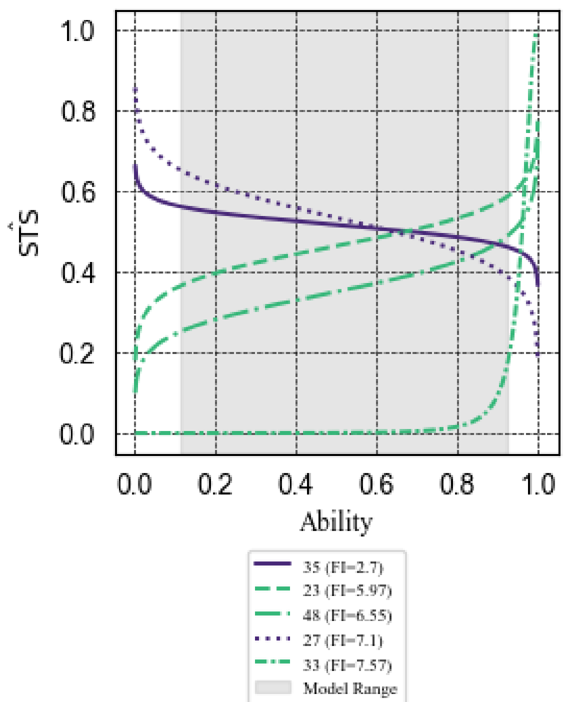
[19]:
```python
import numpy as np
import matplotlib.pyplot as plt

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)
```

[20]:
```python
import matplotlib.pyplot as plt
import numpy as np

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 [34,22,47,26,32]:
    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.xticks(np.arange(0, 1.2, 0.2))
```

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

legend = plt.legend(loc='upper left', fontsize=6.5, bbox_to_anchor=(0.28, -0.
 ↪2), ncol=1)

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("Figure3b.pdf", format="pdf", bbox_inches='tight')
plt.show()
```

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




```
[ ]:
```