BAYESIAN MODEL ASSESSMENT FOR JOINTLY MODELING MULTIDIMENSIONAL RESPONSE DATA WITH APPLICATION TO COMPUTERIZED TESTING
Supplementary Materials

Fang Liu, Xiaojing Wang, Roeland Hancock, and Ming-Hui Chen

January 11, 2022

In the Supplementary Materials, we include additional tables and figures, ranking item difficulty, diagnostics for model adequacy, trace, autocorrelation, and density plots, calibration of the DIC and LPML assessment criteria, and codes for MCMC sampling and decomposition of DIC and LPML.

S.1 Additional Tables and Figures

S.1.1 Additional Tables

Table S.1 provides the SE values for all item and latent attribute parameters in four different scenarios in the simulation for Subsection 5.1. Notice we have grouped parameters in their corresponding category according to $\boldsymbol{a} = (a_1, \cdots, a_J)'$, $\boldsymbol{b} = (b_1, \cdots, b_J)'$, $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_J)'$, $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_N)'$ and $\boldsymbol{\tau} = (\tau_1, \cdots, \tau_N)'$ for summarizing their results. For example, in Table S.1, the content in the intersection of Column 'SE' and Row '$\boldsymbol{a}$ (Min, Max)' presents the median value of the SE among all a_js and the bracket shows

Table S.1: SE Summary of parameters in the simulation

	$N = 125, J = 20$	$N = 125, J = 40$	$N = 250, J = 20$	$N = 250, J = 40$
$\boldsymbol{a}$ (Min, Max)	.253 (.151 , .292)	.248 (.139 , .295)	.193 (.117 , .274)	.188 (.104 , .267)
$\boldsymbol{b}$ (Min, Max)	.255 (.157 , .376)	.236 (.141 , .400)	.166 (.091 , .349)	.153 (.098 , .325)
$\boldsymbol{\lambda}$ (Min, Max)	.056 (.039 , .078)	.056 (.034 , .076)	.040 (.026 , .056)	.039 (.026 , .055)
$\boldsymbol{\theta}$ (Min, Max)	.323 (.300 , .360)	.273 (.246 , .327)	.316 (.287 , .362)	.260 (.231 , .315)
$\boldsymbol{\tau}$ (Min, Max)	.350 (.326 , .392)	.268 (.247 , .298)	.338 (.311 , .362)	.259 (.239 , .284)
β_0	.688	.670	.468	.494
β_1	.838	.689	.552	.492
β_2	1.068	.848	.666	.602
σ^2_{PDE}	13.300	10.260	9.044	6.832

the range of the SE values for all a_js. A similar interpretation can be applied to Bias, MSE, CP and SD for all other columns and rows in the table. Table S.2 to Table S.8 show the posterior estimates of all unknown parameters of the proposed joint model for the AppRISE data. In these tables, the abbreviation 'PARM' denotes the name of the unknown parameter in the model, 'EAP' means posterior estimate of that parameter, 'SD' is the sample standard deviation of our MCMC draws for the parameter and 'HPD' presents the 95% highest probability density interval.

Table S.2: The Discrimination estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
a_1	0.469	0.233	[0.042 , 0.902]	a_2	0.337	0.204	[0.000 , 0.701]
a_3	1.906	0.509	[0.967 , 2.922]	a_4	1.210	0.394	[0.458 , 1.993]
a_5	0.994	0.323	[0.428 , 1.630]	a_6	1.695	0.429	[0.912 , 2.543]
a_7	1.314	0.419	[0.569 , 2.141]	a_8	1.370	0.464	[0.532 , 2.260]
a_9	1.529	0.439	[0.699 , 2.377]	a_{10}	1.562	0.428	[0.773 , 2.416]
a_{11}	0.835	0.413	[0.155 , 1.656]	a_{12}	1.499	0.525	[0.608 , 2.544]
a_{13}	1.020	0.732	[0.000 , 2.291]	a_{14}	0.374	0.373	[0.000 , 1.138]
a_{15}	0.760	0.257	[0.284 , 1.277]	a_{16}	1.280	0.441	[0.504 , 2.130]
a_{17}	1.038	0.415	[0.360 , 1.866]	a_{18}	0.908	0.455	[0.196 , 1.829]
a_{19}	0.562	0.342	[0.000 , 1.185]	a_{20}	1.474	0.489	[0.643 , 2.454]
a_{21}	0.448	0.358	[0.000 , 1.172]	a_{22}	0.596	0.293	[0.136 , 1.186]
a_{23}	1.261	0.425	[0.521 , 2.096]	a_{24}	0.662	0.262	[0.172 , 1.160]
a_{25}	1.685	0.446	[0.889 , 2.577]	a_{26}	0.328	0.338	[0.000 , 1.030]
a_{27}	0.798	0.511	[0.049 , 1.832]	a_{28}	1.069	0.585	[0.140 , 2.208]
a_{29}	0.402	0.335	[0.000 , 1.067]	a_{30}	1.147	0.448	[0.377 , 2.014]
a_{31}	0.312	0.302	[0.000 , 0.928]	a_{32}	0.373	0.322	[0.000 , 0.995]
a_{33}	1.153	0.540	[0.256 , 2.249]	a_{34}	1.100	0.431	[0.400 , 1.989]
a_{35}	1.560	0.415	[0.805 , 2.413]	a_{36}	0.607	0.426	[0.000 , 1.452]
a_{37}	1.165	0.337	[0.540 , 1.821]	a_{38}	0.831	0.292	[0.260 , 1.396]
a_{39}	0.304	0.226	[0.000 , 0.748]	a_{40}	1.255	0.373	[0.557 , 1.954]
a_{41}	0.536	0.315	[0.007 , 1.108]	a_{42}	1.791	0.510	[0.853 , 2.813]
a_{43}	0.800	0.496	[0.003 , 1.717]				

Table S.3: The Difficulty estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
b_1	-0.558	0.670	[-1.981 , 0.564]	b_2	0.434	0.836	[-1.370 , 2.110]
b_3	-1.927	0.386	[-2.706 , -1.249]	b_4	0.087	0.267	[-0.480 , 0.547]
b_5	-2.690	0.854	[-4.342 , -1.231]	b_6	-0.334	0.193	[-0.706 , 0.036]
b_7	-2.814	0.855	[-4.541 , -1.420]	b_8	-2.371	0.766	[-3.927 , -1.102]
b_9	-2.017	0.484	[-2.981 , -1.219]	b_{10}	-1.761	0.406	[-2.602 , -1.087]
b_{11}	-1.474	1.204	[-3.868 , 0.742]	b_{12}	-2.461	0.935	[-4.344 , -0.902]
b_{13}	2.106	1.557	[-1.677 , 4.348]	b_{14}	-0.200	1.488	[-3.128 , 2.499]
b_{15}	-0.672	0.427	[-1.513 , 0.035]	b_{16}	-2.187	0.744	[-3.704 , -0.993]
b_{17}	-1.982	1.033	[-4.027 , -0.210]	b_{18}	-0.916	0.961	[-2.783 , 0.606]
b_{19}	0.303	0.806	[-1.541 , 1.656]	b_{20}	-2.530	0.791	[-4.103 , -1.227]
b_{21}	-0.143	1.154	[-2.630 , 1.690]	b_{22}	-1.139	0.908	[-3.069 , 0.265]
b_{23}	-2.369	0.768	[-3.942 , -1.147]	b_{24}	-0.931	0.598	[-2.139 , -0.027]
b_{25}	-1.739	0.370	[-2.481 , -1.088]	b_{26}	-0.185	1.515	[-3.149 , 2.565]
b_{27}	-0.491	1.161	[-2.798 , 1.442]	b_{28}	-0.259	0.930	[-2.219 , 1.062]
b_{29}	-0.524	1.327	[-3.118 , 1.816]	b_{30}	-1.758	0.917	[-3.603 , -0.285]
b_{31}	-0.221	1.395	[-2.984 , 2.259]	b_{32}	-0.611	1.452	[-3.445 , 2.217]
b_{33}	-1.170	1.121	[-3.352 , 0.827]	b_{34}	-1.537	0.780	[-3.126 , -0.300]
b_{35}	-1.681	0.384	[-2.444 , -1.016]	b_{36}	-0.413	1.158	[-2.798 , 1.456]
b_{37}	-2.046	0.561	[-3.174 , -1.121]	b_{38}	0.890	0.367	[0.223 , 1.625]
b_{39}	-0.939	1.297	[-3.432 , 1.488]	b_{40}	-2.379	0.596	[-3.634 , -1.445]
b_{41}	0.037	0.752	[-1.635 , 1.311]	b_{42}	2.151	0.360	[1.539 , 2.899]
b_{43}	0.105	0.877	[-1.803 , 1.434]				

We note that the low frequency of subjects responding to some items may be a possible reason for the high posterior standard deviations for those items' difficulty levels. For example, 8 subjects responded item 26 and the posterior standard deviation of b_{26} is 1.515; on the contrary, 108 subjects responded item 6 and the posterior standard deviation of b_6 is 0.193.

Table S.4: The λ estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
λ_1	1.219	0.056	[1.114 , 1.331]	λ_2	1.341	0.067	[1.212 , 1.472]
λ_3	0.828	0.052	[0.724 , 0.926]	λ_4	1.260	0.071	[1.121 , 1.402]
λ_5	1.276	0.058	[1.163 , 1.389]	λ_6	1.268	0.065	[1.138 , 1.392]
λ_7	1.053	0.062	[0.936 , 1.176]	λ_8	1.216	0.061	[1.098 , 1.339]
λ_9	1.219	0.049	[1.117 , 1.311]	λ_{10}	1.019	0.053	[0.915 , 1.121]
λ_{11}	1.558	0.115	[1.335 , 1.794]	λ_{12}	1.212	0.066	[1.078 , 1.340]
λ_{13}	1.260	0.198	[0.882 , 1.659]	λ_{14}	1.758	0.289	[1.199 , 2.343]
λ_{15}	0.980	0.057	[0.872 , 1.093]	λ_{16}	1.098	0.059	[0.986 , 1.215]
λ_{17}	1.196	0.080	[1.043 , 1.356]	λ_{18}	1.397	0.119	[1.170 , 1.637]
λ_{19}	1.412	0.072	[1.266 , 1.547]	λ_{20}	0.963	0.052	[0.867 , 1.068]
λ_{21}	1.631	0.140	[1.362 , 1.915]	λ_{22}	1.498	0.068	[1.366 , 1.633]
λ_{23}	1.172	0.052	[1.072 , 1.276]	λ_{24}	1.223	0.063	[1.095 , 1.341]
λ_{25}	0.989	0.054	[0.884 , 1.099]	λ_{26}	1.873	0.281	[1.328 , 2.446]
λ_{27}	1.165	0.142	[0.880 , 1.440]	λ_{28}	1.579	0.114	[1.346 , 1.796]
λ_{29}	1.525	0.150	[1.226 , 1.819]	λ_{30}	1.230	0.070	[1.095 , 1.367]
λ_{31}	1.874	0.105	[1.667 , 2.082]	λ_{32}	1.500	0.301	[0.921 , 2.118]
λ_{33}	1.376	0.119	[1.139 , 1.610]	λ_{34}	1.318	0.066	[1.190 , 1.446]
λ_{35}	0.823	0.051	[0.719 , 0.919]	λ_{36}	1.295	0.115	[1.071 , 1.519]
λ_{37}	1.020	0.056	[0.908 , 1.128]	λ_{38}	1.469	0.070	[1.330 , 1.605]
λ_{39}	1.474	0.120	[1.236 , 1.708]	λ_{40}	0.931	0.047	[0.835 , 1.020]
λ_{41}	1.408	0.071	[1.268 , 1.547]	λ_{42}	1.250	0.079	[1.088 , 1.397]
λ_{43}	1.387	0.091	[1.211 , 1.569]				

Table S.5: The σ_j^2 estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
σ_1^2	0.239	0.035	[0.176 , 0.308]	σ_2^2	0.384	0.053	[0.289 , 0.494]
σ_3^2	0.196	0.028	[0.146 , 0.252]	σ_4^2	0.374	0.059	[0.262 , 0.489]
σ_5^2	0.230	0.034	[0.167 , 0.299]	σ_6^2	0.350	0.050	[0.259 , 0.453]
σ_7^2	0.269	0.040	[0.196 , 0.350]	σ_8^2	0.272	0.041	[0.196 , 0.354]
σ_9^2	0.160	0.023	[0.116 , 0.207]	σ_{10}^2	0.213	0.030	[0.159 , 0.273]
σ_{11}^2	0.163	0.079	[0.053 , 0.310]	σ_{12}^2	0.189	0.038	[0.123 , 0.266]
σ_{13}^2	0.480	0.242	[0.169 , 0.941]	σ_{14}^2	0.747	0.521	[0.174 , 1.625]
σ_{15}^2	0.252	0.035	[0.188 , 0.324]	σ_{16}^2	0.238	0.036	[0.172 , 0.310]
σ_{17}^2	0.188	0.049	[0.106 , 0.286]	σ_{18}^2	0.457	0.117	[0.259 , 0.689]
σ_{19}^2	0.292	0.051	[0.199 , 0.394]	σ_{20}^2	0.160	0.025	[0.115 , 0.209]
σ_{21}^2	0.634	0.170	[0.353 , 0.979]	σ_{22}^2	0.214	0.042	[0.139 , 0.297]
σ_{23}^2	0.159	0.025	[0.115 , 0.210]	σ_{24}^2	0.326	0.046	[0.239 , 0.418]
σ_{25}^2	0.229	0.032	[0.171 , 0.291]	σ_{26}^2	0.603	0.503	[0.136 , 1.401]
σ_{27}^2	0.318	0.129	[0.124 , 0.567]	σ_{28}^2	0.327	0.097	[0.164 , 0.520]
σ_{29}^2	0.355	0.147	[0.148 , 0.650]	σ_{30}^2	0.149	0.036	[0.087 , 0.221]
σ_{31}^2	0.196	0.074	[0.087 , 0.343]	σ_{32}^2	0.518	0.636	[0.076 , 1.378]
σ_{33}^2	0.191	0.084	[0.072 , 0.360]	σ_{34}^2	0.202	0.039	[0.131 , 0.280]
σ_{35}^2	0.192	0.027	[0.144 , 0.246]	σ_{36}^2	0.297	0.097	[0.144 , 0.490]
σ_{37}^2	0.251	0.035	[0.186 , 0.321]	σ_{38}^2	0.357	0.055	[0.261 , 0.472]
σ_{39}^2	0.284	0.096	[0.135 , 0.478]	σ_{40}^2	0.144	0.021	[0.106 , 0.186]
σ_{41}^2	0.318	0.053	[0.219 , 0.422]	σ_{42}^2	0.289	0.058	[0.186 , 0.405]
σ_{43}^2	0.287	0.071	[0.164 , 0.428]	μ_b	-0.958	0.301	[-1.555 , -0.373]
σ_b^2	2.295	0.809	[0.993 , 3.904]				

Table S.6: The ability estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
θ_1	1.363	0.472	[0.432 , 2.268]	θ_2	-1.907	0.403	[-2.689 , -1.124]
θ_3	0.188	0.395	[-0.567 , 0.963]	θ_4	-1.668	0.381	[-2.428 , -0.931]
θ_5	-0.072	0.384	[-0.843 , 0.658]	θ_6	-0.183	0.369	[-0.891 , 0.565]
θ_7	1.900	0.411	[1.057 , 2.670]	θ_8	0.864	0.388	[0.119 , 1.637]
θ_9	-1.176	0.374	[-1.926 , -0.456]	θ_{10}	1.239	0.390	[0.489 , 2.014]
θ_{11}	2.763	0.441	[1.905 , 3.640]	θ_{12}	1.507	0.393	[0.713 , 2.256]
θ_{13}	-0.239	0.375	[-0.969 , 0.511]	θ_{14}	0.584	0.396	[-0.224 , 1.325]
θ_{15}	-0.583	0.357	[-1.249 , 0.138]	θ_{16}	0.487	0.412	[-0.312 , 1.297]
θ_{17}	-0.082	0.417	[-0.926 , 0.708]	θ_{18}	1.773	0.407	[0.976 , 2.564]
θ_{19}	-0.452	0.363	[-1.149 , 0.271]	θ_{20}	0.806	0.383	[0.048 , 1.558]
θ_{21}	-1.724	0.390	[-2.521 , -0.984]	θ_{22}	-0.578	0.361	[-1.271 , 0.153]
θ_{23}	0.230	0.395	[-0.545 , 1.002]	θ_{24}	0.491	0.413	[-0.322 , 1.290]
θ_{25}	0.010	0.387	[-0.771 , 0.758]	θ_{26}	-0.497	0.372	[-1.203 , 0.245]
θ_{27}	-0.298	0.369	[-1.041 , 0.414]	θ_{28}	-1.040	0.376	[-1.778 , -0.312]
θ_{29}	-0.023	0.383	[-0.740 , 0.772]	θ_{30}	1.085	0.389	[0.341 , 1.851]
θ_{31}	0.460	0.372	[-0.272 , 1.169]	θ_{32}	0.010	0.380	[-0.731 , 0.757]
θ_{33}	2.164	0.459	[1.257 , 3.049]	θ_{34}	0.177	0.371	[-0.541 , 0.892]
θ_{35}	-1.460	0.381	[-2.190 , -0.698]	θ_{36}	-1.106	0.383	[-1.878 , -0.382]
θ_{37}	0.866	0.389	[0.109 , 1.635]	θ_{38}	2.454	0.467	[1.493 , 3.301]
θ_{39}	0.593	0.373	[-0.137 , 1.325]	θ_{40}	-0.716	0.351	[-1.425 , -0.039]
θ_{41}	-0.358	0.380	[-1.091 , 0.410]	θ_{42}	-0.318	0.373	[-1.053 , 0.399]
θ_{43}	-0.138	0.380	[-0.911 , 0.565]	θ_{44}	-1.096	0.346	[-1.753 , -0.402]
θ_{45}	0.296	0.374	[-0.421 , 1.025]	θ_{46}	0.604	0.393	[-0.153 , 1.379]
θ_{47}	0.617	0.414	[-0.187 , 1.421]	θ_{48}	0.189	0.396	[-0.590 , 0.966]
θ_{49}	0.111	0.389	[-0.630 , 0.892]	θ_{50}	1.907	0.409	[1.107 , 2.717]
θ_{51}	-0.329	0.368	[-1.045 , 0.393]	θ_{52}	0.392	0.393	[-0.383 , 1.168]
θ_{53}	-0.480	0.366	[-1.166 , 0.266]	θ_{54}	1.111	0.437	[0.240 , 1.954]
θ_{55}	-1.183	0.366	[-1.932 , -0.497]	θ_{56}	0.440	0.397	[-0.319 , 1.237]
θ_{57}	0.197	0.401	[-0.577 , 0.978]	θ_{58}	0.542	0.403	[-0.230 , 1.361]
θ_{59}	-0.001	0.371	[-0.723 , 0.713]	θ_{60}	1.019	0.414	[0.210 , 1.822]
θ_{61}	-0.916	0.360	[-1.605 , -0.185]	θ_{62}	2.345	0.442	[1.507 , 3.247]
θ_{63}	-1.778	0.399	[-2.516 , -0.958]	θ_{64}	1.109	0.393	[0.373 , 1.917]
θ_{65}	0.320	0.385	[-0.413 , 1.104]	θ_{66}	-0.535	0.356	[-1.230 , 0.158]
θ_{67}	0.952	0.386	[0.181 , 1.690]	θ_{68}	2.058	0.420	[1.262 , 2.899]
θ_{69}	-0.171	0.391	[-0.918 , 0.611]	θ_{70}	-0.023	0.435	[-0.858 , 0.845]
θ_{71}	0.719	0.384	[-0.010 , 1.507]	θ_{72}	0.544	0.384	[-0.189 , 1.330]
θ_{73}	0.099	0.382	[-0.634 , 0.863]	θ_{74}	1.306	0.387	[0.561 , 2.084]
θ_{75}	0.999	0.394	[0.238 , 1.793]	θ_{76}	-1.128	0.373	[-1.860 , -0.392]
θ_{77}	-0.667	0.361	[-1.398 , 0.018]	θ_{78}	0.851	0.386	[0.081 , 1.605]

Table S.7: The ability and speed estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
θ_{79}	0.080	0.378	[-0.651 , 0.823]	θ_{80}	0.532	0.393	[-0.237 , 1.307]
θ_{81}	-0.397	0.361	[-1.097 , 0.308]	θ_{82}	0.599	0.396	[-0.162 , 1.389]
θ_{83}	-0.563	0.394	[-1.317 , 0.208]	θ_{84}	0.770	0.400	[-0.005 , 1.566]
θ_{85}	-0.207	0.405	[-0.995 , 0.573]	θ_{86}	-0.088	0.369	[-0.800 , 0.631]
θ_{87}	-0.319	0.362	[-1.061 , 0.360]	θ_{88}	-0.392	0.364	[-1.108 , 0.311]
θ_{89}	-0.690	0.355	[-1.399 , -0.014]	θ_{90}	-0.011	0.382	[-0.756 , 0.736]
θ_{91}	-0.261	0.361	[-0.954 , 0.447]	θ_{92}	2.450	0.457	[1.559 , 3.333]
θ_{93}	1.618	0.406	[0.822 , 2.403]	θ_{94}	0.648	0.387	[-0.083 , 1.430]
θ_{95}	0.417	0.388	[-0.347 , 1.175]	θ_{96}	-0.314	0.383	[-1.081 , 0.421]
θ_{97}	0.824	0.387	[0.084 , 1.596]	θ_{98}	0.236	0.362	[-0.448 , 0.979]
θ_{99}	0.062	0.392	[-0.673 , 0.856]	θ_{100}	-0.517	0.409	[-1.322 , 0.287]
θ_{101}	-0.510	0.398	[-1.277 , 0.266]	θ_{102}	-1.389	0.385	[-2.142 , -0.624]
θ_{103}	-1.877	0.402	[-2.662 , -1.102]	θ_{104}	-1.657	0.391	[-2.440 , -0.900]
θ_{105}	-0.698	0.356	[-1.376 , 0.010]	θ_{106}	-0.652	0.350	[-1.328 , 0.048]
θ_{107}	-0.724	0.353	[-1.381 , -0.005]	θ_{108}	1.373	0.396	[0.630 , 2.174]
θ_{109}	-1.019	0.373	[-1.727 , -0.264]	θ_{110}	-0.392	0.377	[-1.122 , 0.342]
θ_{111}	-0.542	0.403	[-1.318 , 0.250]	θ_{112}	1.286	0.396	[0.554 , 2.085]
θ_{113}	-0.337	0.367	[-1.062 , 0.370]	θ_{114}	-0.642	0.361	[-1.340 , 0.070]
θ_{115}	0.219	0.397	[-0.556 , 0.988]	θ_{116}	-0.795	0.341	[-1.446 , -0.119]
θ_{117}	-0.011	0.378	[-0.719 , 0.766]				
τ_1	-0.156	0.404	[-0.963 , 0.615]	τ_2	1.667	0.541	[0.654 , 2.781]
τ_3	-2.272	0.388	[-3.043 , -1.544]	τ_4	1.558	0.518	[0.553 , 2.557]
τ_5	-0.496	0.390	[-1.234 , 0.277]	τ_6	0.302	0.357	[-0.381 , 1.016]
τ_7	0.339	0.367	[-0.408 , 1.027]	τ_8	-1.200	0.340	[-1.868 , -0.553]
τ_9	-0.385	0.470	[-1.287 , 0.536]	τ_{10}	-0.866	0.336	[-1.525 , -0.209]
τ_{11}	1.044	0.430	[0.197 , 1.886]	τ_{12}	0.785	0.344	[0.101 , 1.451]
τ_{13}	0.824	0.362	[0.138 , 1.563]	τ_{14}	-0.085	0.353	[-0.765 , 0.628]
τ_{15}	-0.523	0.368	[-1.255 , 0.210]	τ_{16}	-0.945	0.392	[-1.739 , -0.204]
τ_{17}	-0.431	0.398	[-1.200 , 0.360]	τ_{18}	0.910	0.364	[0.187 , 1.616]
τ_{19}	-0.065	0.383	[-0.796 , 0.696]	τ_{20}	0.691	0.343	[0.018 , 1.350]
τ_{21}	0.184	0.523	[-0.835 , 1.196]	τ_{22}	-0.155	0.388	[-0.911 , 0.616]
τ_{23}	-0.409	0.376	[-1.161 , 0.302]	τ_{24}	0.351	0.384	[-0.403 , 1.104]
τ_{25}	0.068	0.379	[-0.687 , 0.792]	τ_{26}	-1.022	0.393	[-1.785 , -0.244]
τ_{27}	-0.225	0.354	[-0.971 , 0.433]	τ_{28}	0.444	0.468	[-0.416 , 1.424]
τ_{29}	0.634	0.385	[-0.121 , 1.386]	τ_{30}	1.157	0.340	[0.491 , 1.828]
τ_{31}	0.278	0.341	[-0.387 , 0.944]	τ_{32}	0.359	0.380	[-0.399 , 1.094]
τ_{33}	1.165	0.405	[0.337 , 1.924]	τ_{34}	-0.991	0.342	[-1.675 , -0.334]
τ_{35}	0.943	0.490	[-0.005 , 1.913]	τ_{36}	-1.221	0.477	[-2.188 , -0.316]
τ_{37}	0.531	0.347	[-0.150 , 1.227]	τ_{38}	1.027	0.429	[0.214 , 1.911]

Table S.8: The speed estimation results of AppRISE data

PARM	EAP	SD	HPD	PARM	EAP	SD	HPD
τ_{39}	0.042	0.334	[-0.624 , 0.666]	τ_{40}	-0.020	0.392	[-0.775 , 0.745]
τ_{41}	-1.766	0.403	[-2.539 , -0.959]	τ_{42}	1.273	0.387	[0.527 , 2.056]
τ_{43}	1.094	0.393	[0.328 , 1.895]	τ_{44}	-1.041	0.419	[-1.855 , -0.232]
τ_{45}	1.318	0.344	[0.665 , 1.997]	τ_{46}	-1.184	0.339	[-1.866 , -0.552]
τ_{47}	-2.828	0.381	[-3.568 , -2.098]	τ_{48}	-1.199	0.384	[-1.935 , -0.430]
τ_{49}	0.340	0.386	[-0.403 , 1.121]	τ_{50}	1.215	0.366	[0.477 , 1.910]
τ_{51}	-0.853	0.368	[-1.578 , -0.146]	τ_{52}	-2.644	0.377	[-3.398 , -1.920]
τ_{53}	-0.574	0.383	[-1.325 , 0.180]	τ_{54}	-0.108	0.389	[-0.871 , 0.668]
τ_{55}	-0.304	0.471	[-1.224 , 0.629]	τ_{56}	0.681	0.360	[-0.037 , 1.376]
τ_{57}	-0.652	0.383	[-1.398 , 0.096]	τ_{58}	-0.997	0.364	[-1.694 , -0.282]
τ_{59}	-0.355	0.349	[-1.029 , 0.321]	τ_{60}	-0.985	0.365	[-1.728 , -0.308]
τ_{61}	0.634	0.415	[-0.181 , 1.442]	τ_{62}	1.074	0.404	[0.299 , 1.890]
τ_{63}	2.389	0.536	[1.374 , 3.461]	τ_{64}	1.260	0.355	[0.573 , 1.960]
τ_{65}	-0.659	0.359	[-1.380 , 0.031]	τ_{66}	-0.402	0.384	[-1.164 , 0.342]
τ_{67}	0.796	0.350	[0.107 , 1.474]	τ_{68}	0.284	0.380	[-0.471 , 1.007]
τ_{69}	-1.027	0.398	[-1.835 , -0.269]	τ_{70}	1.364	0.423	[0.524 , 2.175]
τ_{71}	0.239	0.343	[-0.428 , 0.925]	τ_{72}	0.181	0.351	[-0.502 , 0.864]
τ_{73}	0.654	0.341	[0.002 , 1.336]	τ_{74}	0.591	0.337	[-0.071 , 1.249]
τ_{75}	-0.909	0.347	[-1.615 , -0.254]	τ_{76}	0.485	0.461	[-0.404 , 1.406]
τ_{77}	-0.387	0.392	[-1.154 , 0.365]	τ_{78}	0.112	0.338	[-0.596 , 0.747]
τ_{79}	0.238	0.353	[-0.449 , 0.923]	τ_{80}	0.515	0.363	[-0.202 , 1.216]
τ_{81}	0.176	0.378	[-0.546 , 0.929]	τ_{82}	-1.286	0.375	[-2.052 , -0.583]
τ_{83}	0.907	0.455	[0.022 , 1.817]	τ_{84}	-0.933	0.357	[-1.632 , -0.227]
τ_{85}	-0.505	0.412	[-1.307 , 0.302]	τ_{86}	-0.689	0.353	[-1.397 , -0.019]
τ_{87}	-0.043	0.375	[-0.788 , 0.677]	τ_{88}	-0.546	0.382	[-1.305 , 0.198]
τ_{89}	-0.561	0.379	[-1.293 , 0.19]	τ_{90}	-0.002	0.382	[-0.767 , 0.742]
τ_{91}	-0.28	0.35	[-0.968 , 0.409]	τ_{92}	1.43	0.419	[0.605 , 2.237]
τ_{93}	1.182	0.359	[0.467 , 1.866]	τ_{94}	0.37	0.348	[-0.291 , 1.077]
τ_{95}	-0.134	0.356	[-0.802 , 0.591]	τ_{96}	0.743	0.393	[-0.029 , 1.504]
τ_{97}	-0.194	0.329	[-0.816 , 0.482]	τ_{98}	-0.572	0.33	[-1.204 , 0.086]
τ_{99}	-1.541	0.366	[-2.25 , -0.82]	τ_{100}	1.48	0.457	[0.609 , 2.383]
τ_{101}	-0.076	0.457	[-0.959 , 0.82]	τ_{102}	0.92	0.489	[-0.073 , 1.853]
τ_{103}	0.049	0.528	[-0.986 , 1.071]	τ_{104}	-0.937	0.525	[-1.965 , 0.112]
τ_{105}	-0.715	0.386	[-1.478 , 0.037]	τ_{106}	-1.078	0.394	[-1.844 , -0.322]
τ_{107}	0.54	0.388	[-0.256 , 1.265]	τ_{108}	0.382	0.344	[-0.295 , 1.063]
τ_{109}	1.054	0.474	[0.102 , 1.97]	τ_{110}	0.364	0.386	[-0.378 , 1.135]
τ_{111}	-0.384	0.451	[-1.267 , 0.508]	τ_{112}	0.105	0.336	[-0.584 , 0.742]
τ_{113}	0.811	0.379	[0.043 , 1.531]	τ_{114}	-0.241	0.394	[-0.992 , 0.545]
τ_{115}	0.588	0.389	[-0.204 , 1.323]	τ_{116}	-0.094	0.356	[-0.785 , 0.612]
τ_{117}	1.6	0.367	[0.873 , 2.309]				

Table S.9: Posterior estimates of the covariance and correlation parameters in equation 2.4 for the AppRISE data.

Parameter	EAP	SD	95% HPD Interval
$\sigma_{12} = \sigma_\tau \sin\varphi$	0.134	0.035	[0.062, 0.199]
$\sigma_{13} = \beta_1$	14.563	1.264	[12.052, 17.065]
$\sigma_{23} = \sigma_\tau(\beta_1 \sin\varphi + \beta_2 \cos\varphi)$	3.056	0.601	[1.889, 4.220]
$\rho_{12} = \frac{\sigma_{12}}{\sigma_\tau} = \sin\varphi$	0.398	0.097	[0.196, 0.570]
$\rho_{13} = \frac{\sigma_{13}}{\sqrt{\sigma_{33}}} = \frac{\beta_1}{\sqrt{\beta_1^2+\beta_2^2+\sigma_{\text{PDE}}^2}}$	0.864	0.043	[0.773, 0.941]
$\rho_{23} = \frac{\sigma_{23}}{\sigma_\tau\sqrt{\sigma_{33}}} = \frac{\beta_1 \sin\varphi+\beta_2 \cos\varphi}{\sqrt{\beta_1^2+\beta_2^2+\sigma_{\text{PDE}}^2}}$	0.538	0.070	[0.393, 0.666]

Table S.10: Difference measures of model assessment criteria for the AppRISE data.

Difference Measure	Value
$\Delta\text{DIC}_{[\text{AC,RT}]}$	132.900
$\Delta\text{LPML}_{[\text{AC,RT}]}$	71.702
$\Delta\text{DIC}_{[\text{AC,PDE}]}$	38.109
$\Delta\text{LPML}_{[\text{AC,PDE}]}$	21.410
$\Delta\text{DIC}_{[\text{RT,PDE}]}$	109.067
$\Delta\text{LPML}_{[\text{RT,PDE}]}$	60.188

We note that (i) $\Delta\text{DIC}_{[\text{AC,RT}]} = \text{DIC}_{[\text{AC,RT}]} - \text{DIC}_{[\text{AC,RT|PDE}]}$ and $\Delta\text{LPML}_{[\text{AC,RT}]} = \text{LPML}_{[\text{AC,RT|PDE}]} - \text{LPML}_{[\text{AC,RT}]}$, which quantify the gain in the fit of the AC and RT data by incorporating the PDE data; (ii) $\Delta\text{DIC}_{[\text{AC,PDE}]} = \text{DIC}_{[\text{AC,PDE}]} - \text{DIC}_{[\text{AC,PDE|RT}]}$ and $\Delta\text{LPML}_{[\text{AC,PDE}]} = \text{LPML}_{[\text{AC,PDE|RT}]} - \text{LPML}_{[\text{AC,PDE}]}$, which quantify the gain in the fit of the AC and PDE data by incorporating the RT data; and (iii) $\Delta\text{DIC}_{[\text{RT,PDE}]} = \text{DIC}_{[\text{RT,PDE}]} - \text{DIC}_{[\text{RT,PDE|AC}]}$ and $\Delta\text{LPML}_{[\text{RT,PDE}]} = \text{LPML}_{[\text{RT,PDE|AC}]} - \text{LPML}_{[\text{RT,PDE}]}$, which quantify the gain in the fit of the RT and PDE data by incorporating the AC data.

S.1.2 Additional Figures

Figures S.1, S.2, and S.3 show the frequency plots in the left panel are based on simulation design $N = 125$ and $J = 40$ and the frequency plots in the right panel are based on AppRISE data for the θ_i and τ_i parameters, for the a_j and b_j parameters, and for the λ_j and σ_j^2 parameters, respectively, while Figure S.4 shows the distributions of item accuracies, response times, and PDE scores of the AppRISE data.

From Figures S.1, S.2, and S.3, we see that (i) the distributions of the true θ_i and τ_i parameters in the simulation study are similar to those of the posterior means of the θ_i and τ_i parameters from the empirical data; (ii) there are more true a_j parameters that are greater than 1 and more true b_j parameters that are greater than 0 compared to the posterior means of the a_j parameters as well as the b_j parameters, implying that the items in our simulation study are more discriminable and more difficult than those items in the AppRISE data; and (iii) the distribution of the true λ_j parameters in the simulation study is similar to the distribution of the posterior means of λ_j from the empirical data but there are more true σ_j^2 parameters that are greater than 0.4 in the simulation study compared to those posterior means of the σ_j^2 parameters from the empirical data, implying that the response times in the simulation study are more variable than those in the AppRISE data.

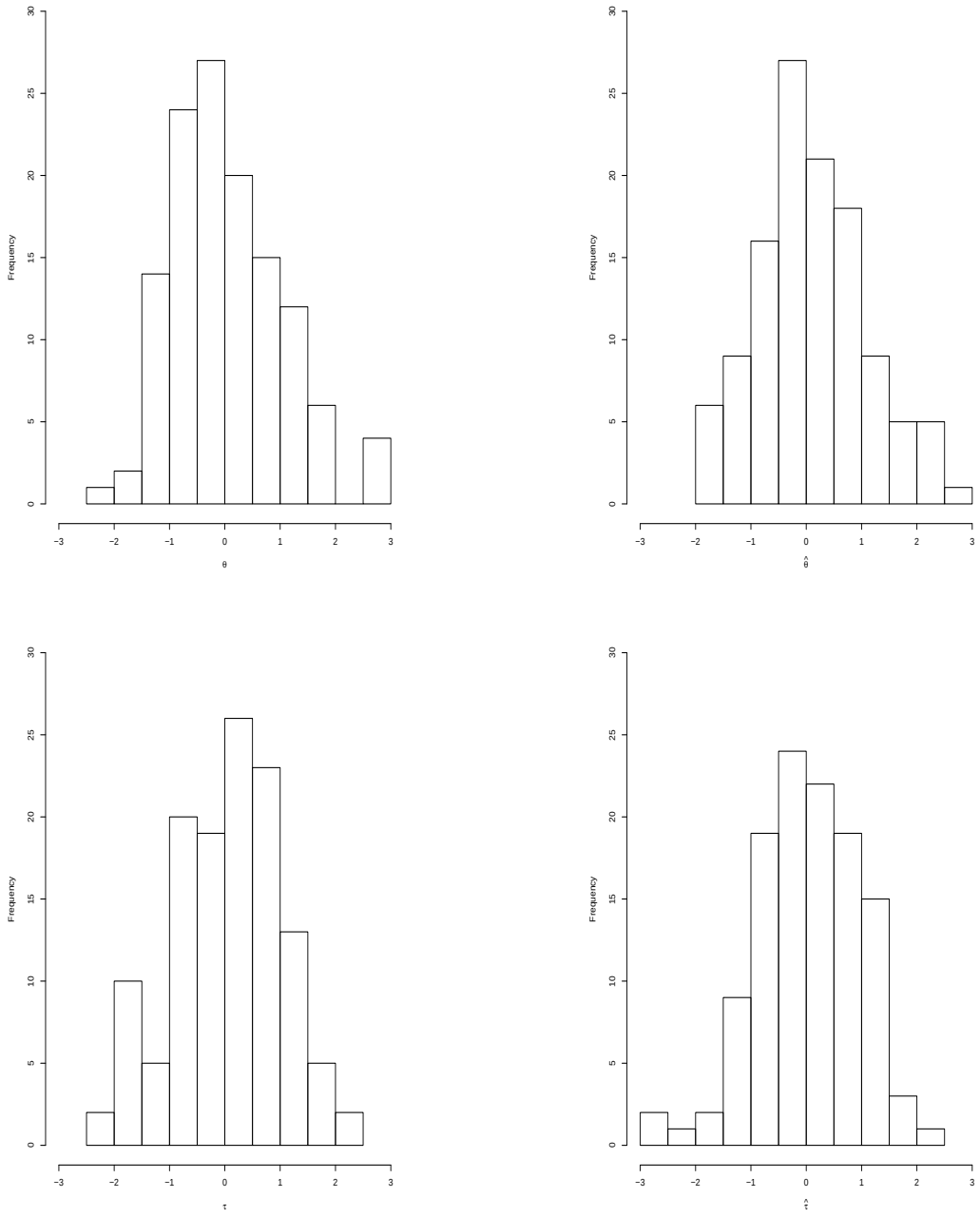


Figure S.1: The frequency plots in the left panel are based on simulation design $N = 125$ and $J = 40$ and the frequency plots in the right panel are based on AppRISE data for the θ_i and τ_i parameters.

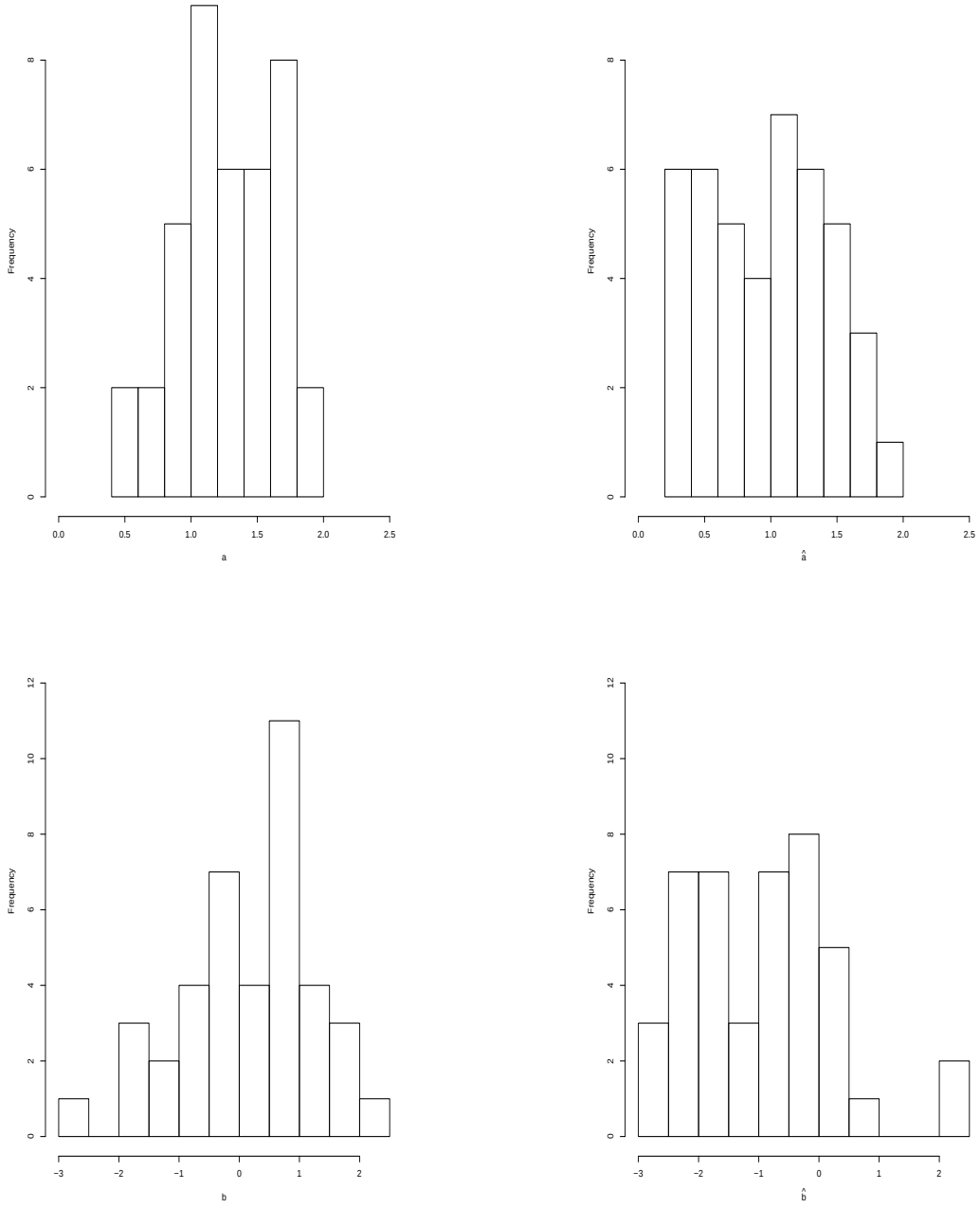


Figure S.2: The frequency plots in the left panel are based on simulation design $N = 125$ and $J = 40$ and the frequency plots in the right panel are based on AppRISE data for the a_j and b_j parameters.

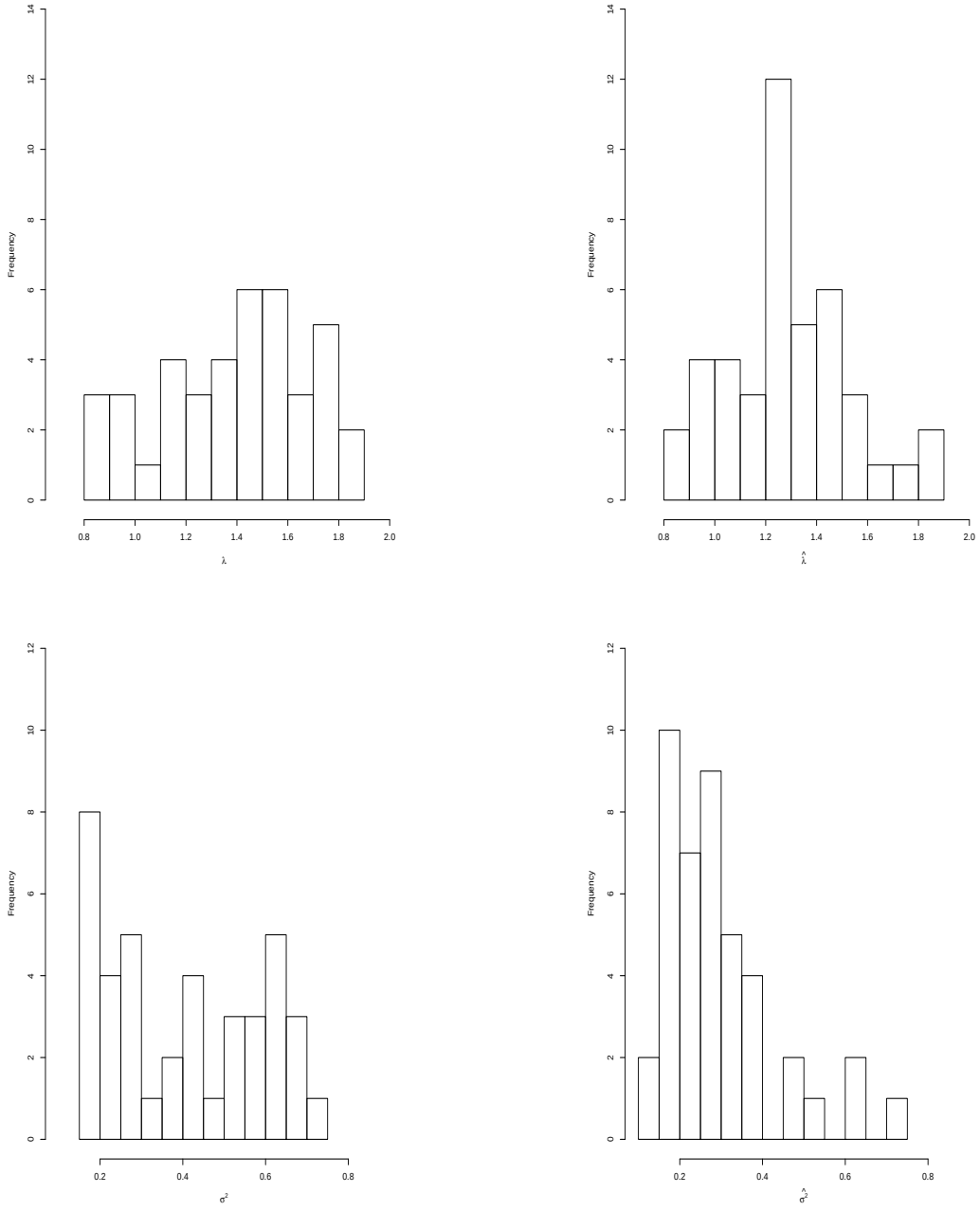
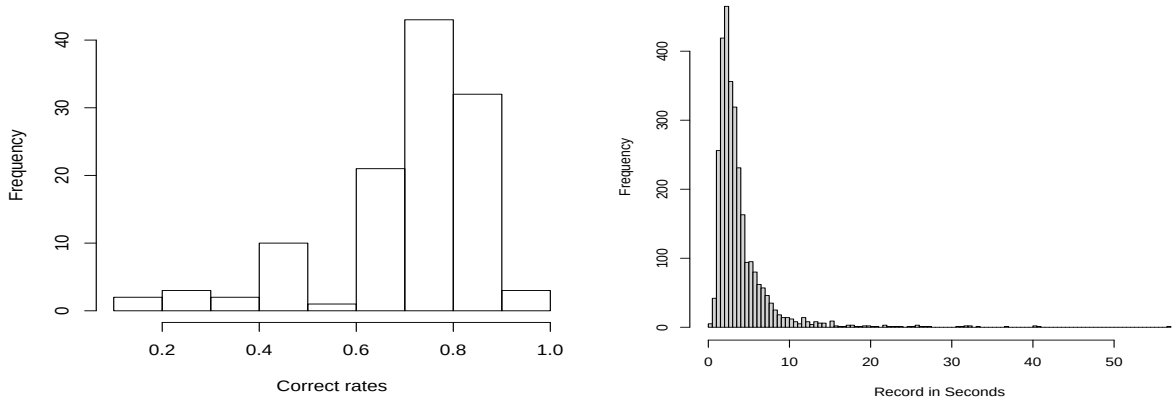
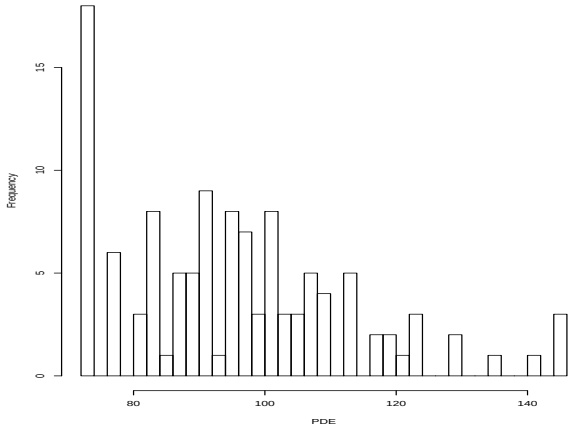


Figure S.3: The frequency plots in the left panel are based on simulation design $N = 125$ and $J = 40$ and the frequency plots in the right panel are based on AppRISE data for the λ_j and σ_j^2 parameters.



(a) Distribution of item accuracy of participants. (b) supplfig/Distribution of item response time.



(c) PDE standard scores

Figure S.4: Frequency histograms of the AppRISE data.

S.2 Ranking Item Difficulty

Researchers are often interested in comparing and ordering item difficulties. In our Bayesian framework, one way to rank the difficulty is to compare the posterior means of b_js in the IRT model equation 2.1, which can be computed via averaging the MCMC samples from their posterior distributions. However, this approach does not account for uncertainty of the b_js. A better approach, often used in network meta-analysis to compare different treatments (Rücker & Schwarzer, 2015), is to rank the estimated probability for b_js being the maximum among all items using the MCMC samples. To illustrate the idea of how to compute this probability, we use Item 1 as an example. Assume the probability of Item 1 being ranked as the most difficult item among all J items, namely, $\mathrm{P}(b_1 = \max_{1\leq j\leq J} b_j \mid \mathcal{D}_{obs})$ can be estimated using the MCMC samples as follows.

- Step 1: Generate MCMC samples $b_j^{[m]}$ from the posterior distribution, for $m = 1, \cdots, M$
- Step 2: Then, $\sum_{m=1}^{M} I\{b_1^{[m]} = \max_{1\leq j\leq J} b_j^{[m]}\}/M$ is used to estimate $\mathrm{P}(b_1 = \max_{1\leq j\leq J} b_j \mid \mathcal{D}_{obs})$.

Here $I\{b_1^{[m]} = \max_{1\leq j\leq J} b_j^{[m]}\}$ is the indicator function, being one when b_1 is the maximum and zero if otherwise. The above probability can be displayed using a cumulative ranking curve and summarized by the surface under the cumulative ranking curve (SUCRA) (Salanti et al., 2011; Chaimani et al., 2013). The most difficult item will have the highest SUCRA value. Thus, when the value of SUCRA is one, the corresponding item is the most difficult, while the value of SUCRA is zero, it indicates that item is the easiest one.

Figure S.5 presents the results of SUCRA values under the joint model and IRT model alone for all 43 items in AppRISE data, where the blue triangle indicates the SUCRA value for the joint model, while the red dot represents the SUCRA value from the IRT model alone. We have marked out the item numbers and their corresponding SUCRA values for the five most difficult one and the five most easiest one for the two models in

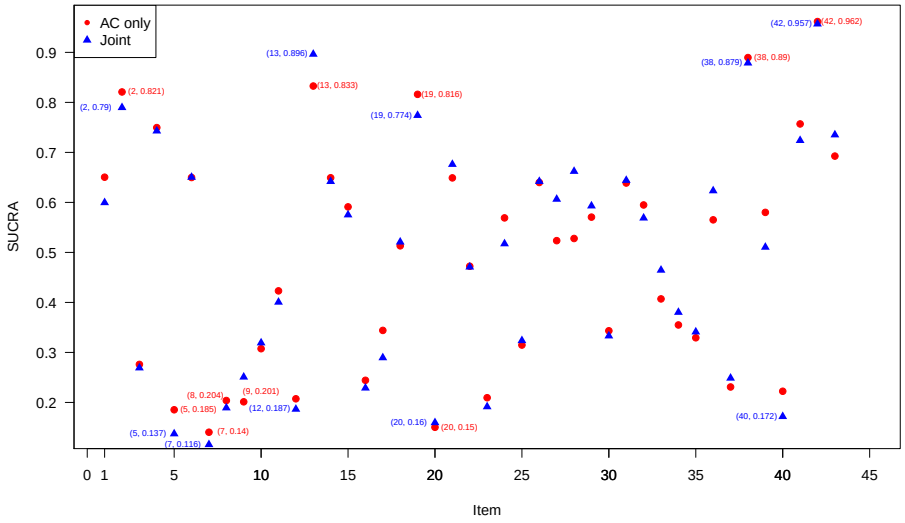


Figure S.5: The results of SUCRA values under the joint model and IRT model alone.

the comparison, which are shown as a pair in the brackets near the triangles or dots of the plot. It is clearly seen that the trend of SUCRA values are similar for both model, but with slightly changes of values on some specific items. From Figure S.5, we see the most five difficult items are the same for the two models we consider, which are Item 2, Item 13, Item 19, Item 38 and Item 42 based on their SUCRA values, while the set of the five most easiest items for these two models is slightly different. The Item 13 and Item 42 have the biggest SUCRA value for the joint model, which is concordant with that they have lowest correct rate, 0.154 and 0.158, respectively in the exploratory data analysis. In the IRT model, the SUCRA values are changed a little bit for Item 13 from 0.896 to 0.833, which make Item 42 becomes the most difficult one in the IRT model and then Item 38 becomes the second. Similar phenomenon happens for Item 27 and Item 28, where they also have similar correct rate, 0.765 and 0.750, respectively. The changes of SUCRA values are comparatively large for these two items between the two models in the comparison. In addition, we have found that the number of participants who took Item 13, Item 42, Item 27 and Item 28 are 13, 57, 28 and 17. It seems if based only on item

response data to rank the item difficulty of the items that have similar correct rates, there is less support to distinguish the difficulty very well when the number of participants is small.

S.3 Diagnostics for Model Adequacy

Following the criteria of $\Delta\text{DIC}_{\text{AC}}$ and $\Delta\text{LPML}_{\text{AC}}$, it supports the joint model is favorable over the AC model alone in the analysis of AppRISE data. But in the joint model, we are still not sure whether the two-parameter logistic AC model is adequate in fitting the item response data. How about the log-normal model for the RT data? Is it reasonable for the PDE data with normal model? In this section, we focus on the development of Bayesian residuals for assessing the model adequacy of the joint model in fitting the item response, RT and PDE data. As the item responses are dichotomous but the RT and PDE data are continuous, we employ two different methods to evaluate the residuals of AC, PDE models, and RT model, separately.

S.3.1 The Residuals for Item Response Data

There is a rich literature on Bayesian model adequacy such as Box (1980); Geisser (1987); Gelfand et al. (1992); Dey et al. (1995); Chen et al. (1998) and many others. Particularly, Chen et al. (1998) and Gelfand et al. (1992) discussed a cross-validation based approach, in which the predictive distribution is used to assess model adequacy. We employ this approach to our joint model as follows.

Denote $\mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}$ to indicate the data with only the ith participant's responses removed from our multidimensional response data. Then, define a CPO using the posterior predictive density of $\boldsymbol{y}_i$ given $\mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}$, i.e.,

$$\text{CPO}_i^{(-\boldsymbol{y}_i)} = \pi\left(\boldsymbol{y}_i|\mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right) = \int\int f(\boldsymbol{y}_i|\boldsymbol{\gamma},\theta_i)\pi(\boldsymbol{\gamma},\theta_i|\mathcal{D}_{obs}^{(-\boldsymbol{y}_i)})d\boldsymbol{\gamma}d\theta_i,$$

which is called the cross-validated predictive density. To save the space, we omit the detailed derivation of $f(\boldsymbol{y}_i|\boldsymbol{\gamma},\theta_i)$ and $\pi(\boldsymbol{\gamma},\theta_i|\mathcal{D}_{obs}^{(-\boldsymbol{y}_i)})$, which are not hard to obtain. Then, the Bayesian standardized residual for the ith participant is computed through

$$\boldsymbol{d}_i = (d_{i1}, d_{i2}, ...d_{iJ_i})' = \left[\text{Cov}\left(\boldsymbol{y}_i \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)\right]^{-1/2} \times \left[\boldsymbol{y}_i - \text{E}\left(\boldsymbol{y}_i \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)\right], \qquad (1)$$

where $\text{E}\left(\boldsymbol{y}_i \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)$ is a J_i-dimensional vector of expected values and $\text{Cov}\left(\boldsymbol{y}_i \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)$ is a $J_i \times J_i$ covariance matrix of the posterior predictive distribution $\pi\left(\boldsymbol{y}_i \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)$. Although the key quantities in Equation (1) are not in closed form, we can use a posterior sampling-based method to approximate them. Recall that $\boldsymbol{\gamma}^{[m]}$ and $\theta_i^{[m]}$ for $m = 1, \cdots, M$ are MCMC draws from the joint posterior of the joint model given $\mathcal{D}_{obs}$ and denote $\boldsymbol{p}_i = (p_{i1}, \cdots, p_{iJ_i})'$ with p_{ij} defined in equation 2.7, then, for $j = 1, \cdots, J_i$, $k = 1, \cdots, J_i$ and $j \neq k$,

$$\begin{aligned}
\widehat{\text{E}}(\boldsymbol{y}_i \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}) &= \widehat{\text{CPO}}_i^{(-\boldsymbol{y}_i)} \times \frac{1}{M}\sum_{m=1}^{M} \frac{\boldsymbol{p}_i^{[m]}}{f(\boldsymbol{y}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]})}, \\
\widehat{\text{Var}}(y_{ij} \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}) &= \widehat{\text{CPO}}_i^{(-\boldsymbol{y}_i)} \times \frac{1}{M}\sum_{m=1}^{M} \frac{p_{ij}^{[m]}}{f(\boldsymbol{y}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]})} - \left[\widehat{\text{E}}\left(y_{ij} \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)\right]^2,
\end{aligned}$$

$$\widehat{\text{Cov}}(y_{ij}, y_{ik}|\mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}) = \widehat{\text{CPO}}_i^{(-\boldsymbol{y}_i)} \times \frac{1}{M}\sum_{m=1}^{M} \frac{p_{ij}^{[m]} p_{ik}^{[m]}}{f(\boldsymbol{y}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]})} - \widehat{\text{E}}\left(y_{ij} \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right)\widehat{\text{E}}\left(y_{ik} \mid \mathcal{D}_{obs}^{(-\boldsymbol{y}_i)}\right),$$

where $\widehat{\text{CPO}}_i^{(-\boldsymbol{y}_i)} = M/\sum_{m=1}^{M}\left[f(\boldsymbol{y}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]})\right]^{-1}$ and $\boldsymbol{p}_i^{[m]} = (p_{i1}^{[m]}, \cdots, p_{iJ_i}^{[m]})'$.

Though we can plug in these estimates above into Equation (1), the estimate $\boldsymbol{d}_i$ is a vector. To better present in a graph, we average the elements of $\boldsymbol{d}_i$ to define $s_i = \sum_{j=1}^{J_i} d_{ij}/J_i$, where the idea of taking averages specific to a participant's level was enlightened by Wright and Stone (1979) and Wang et al. (2018). Then, a substantially large absolute value of s_i, i.e., $|s_i|$, indicates an aberrant. We use a simple scatter plot of the residual s_i versus the participant in Figure S.6 to show any potential outliers in the item response data. From this figure, most points are uniformly spread out above and below zero, which indicates the two-parameter logistic IRT model used in our analysis fit the item response data well.

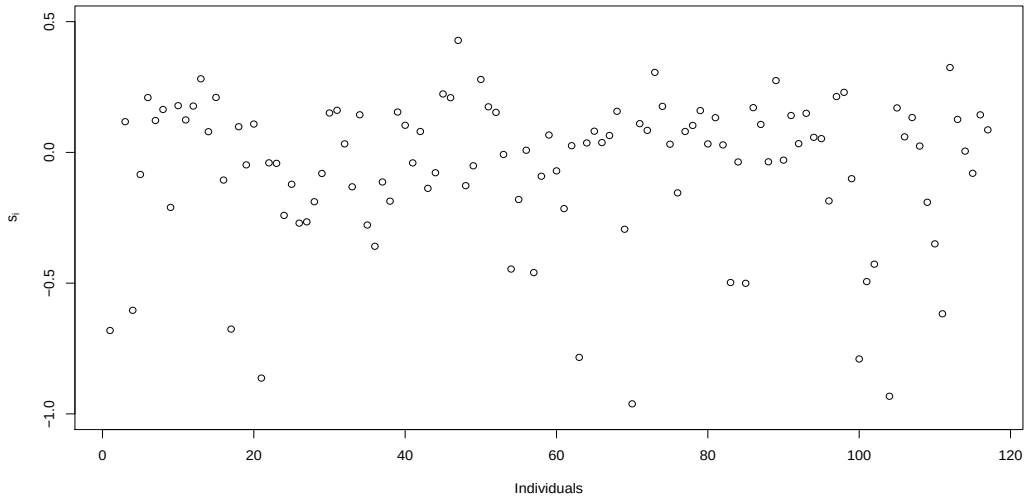


Figure S.6: Bayesian residual plots of s_i versus participants for item response data

S.3.2 The Residuals for the RT Data

Following the RT model 2.2, given the jth item, for any $i \in \{1, \cdots, N\}$, we have

$$\epsilon_{ij}/\sigma_j = (\log t_{ij} - \lambda_j + \sigma_\tau \tau_i^*)/\sigma_j \sim \mathcal{N}(0, 1). \tag{2}$$

Since the RT is continuous, define $e_{ij} = (\log t_{ij} - \widehat{\lambda}_j + \widehat{\sigma_\tau}\widehat{\tau}_i^*)/\widehat{\sigma}_j$, as a standardized residual, where $\widehat{\lambda}_j$, $\widehat{\tau}_i^*$ and $\widehat{\sigma}_j^2$ denote the posterior means of λ_j, τ_i^* and σ_j^2 in the model 2.2. Then, for a given item j, we order the values of e_{ij} for $i = 1, \cdots, N$ and suppose there are $K(\leq N)$ distinct residuals in total, which yields $e_{(1)j} < \cdots \leq e_{(k)j} < \cdots \leq e_{(K)j}$ with $e_{(k)j}$ indicating the kth order statistic of residuals for the jth item, $e_{(1)j}$ being the minimum value and $e_{(K)j}$ being the maximum. Define a probability $S_j(e) = \mathrm{P}(Z_j \geq e)$, where Z_j represents a random event that the residual of a participant for the RT model will be larger than a real value e for the jth item, where $j = 1, \cdots, J$. Then, from Formula (2), we have $Z_j \sim \mathcal{N}(0, 1)$ and thus, $1 - S_j(e)$ is a cdf of the standard normal distribution.

We estimate $S_j(e)$ by $\widehat{S}_j(e) = \prod_{k:e_{(k)j} \leq e} (1 - c_{kj}/h_{kj})$ according to Kaplan and Meier (1958) (KM), where h_{kj} is the number of e_{ij}s satisfying $e \geq e_{(k)j}$ and c_{kj} equals to the

number of e_{ij}s with $e = e_{(k)j}$, for $i \in \{1, \cdots, N\}$. Hence, if a log-normal model for the RT data is appropriate, then $\Phi^{-1}(1 - S(e_{ij}))$ versus e_{ij} should be roughly linear with slope equal to one, where $\Phi^{-1}(\cdot)$ is the inverse of standard normal cdf. To derive the frequentist confidence intervals with better small sample properties, we follow the idea of Borgan and Liestøl (1990) to use the log-minus-log transformation of the standard $100(1-\alpha)\%$ confidence interval for $S(e_{ij})$. Then, the transformed $100(1-\alpha)\%$ confidence interval is $([\widehat{S}(e_{ij})]^{\rho}, \widehat{S}(e_{ij})]^{1/\rho})$,where $\widehat{S}(e_{ij})$ is the KM estimate, $\rho = e^{z_{\alpha/2}\widehat{\sigma}_S(e_{ij})/\log \widehat{S}(e_{ij})}$, $z_{\alpha/2}$ is the upper $(\alpha/2)\%$ percentile of the standard normal pdf, $\widehat{\sigma}^2_S(e_{ij}) = \widehat{\mathrm{Var}}(\widehat{S}(e_{ij}))/\widehat{S}^2(e_{ij})$, and $\widehat{\mathrm{Var}}(\widehat{S}(e_{ij})) = [\widehat{S}(e_{ij})]^2 \sum_{k:e_{(k)j} \le e_{ij}} c_{kj}/(h_{kj}(h_{kj} - c_{kj}))$ is the variance of the KM estimator calculated by Greenwood's formula (Greenwood, 1926).

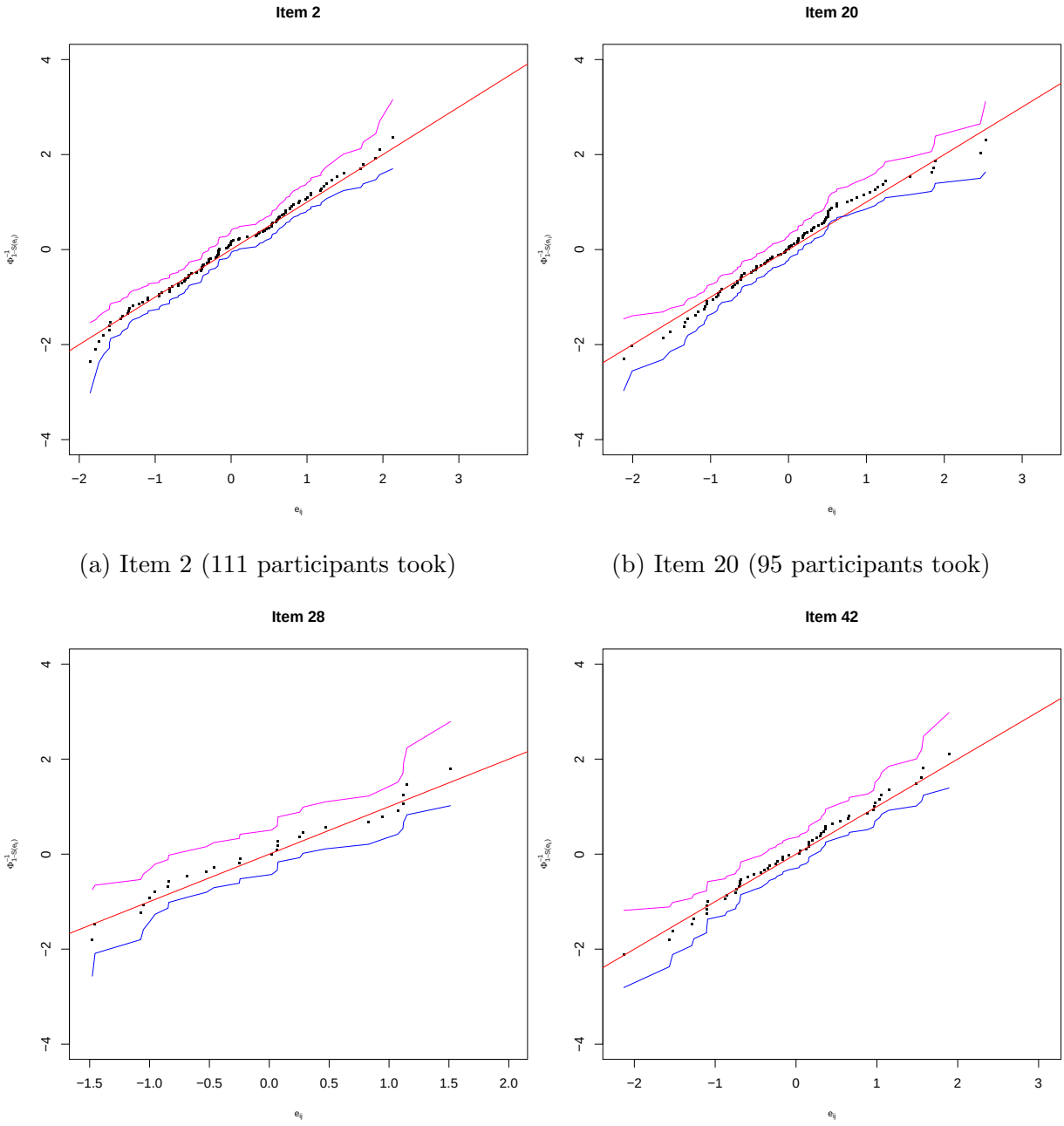
Figure S.7 presents the plot of item specific residuals for the RT data, where to save the space, we only select the graphs for a few items. The subfigures in Figure S.7 illustrate three different types of the number of participants (small, medium and comparatively large) for item specific residuals. When the number of participants increases, the residuals in black dots are much closer to the red line with slope one, which suggests the log-normal RT model fits the data very well. In addition, the 95% confidence bands of the estimation of $\Phi^{-1}(1 - S(e_{ij}))$ are much tighter when the sample size of participants increase, which implies there are less uncertainty about our conclusion for log-normal model in fitting the RT data.

S.3.3 The Residuals for the PDE Data

Following the PDE model 2.9, denote $\mathcal{D}_{obs}^{(-\mathrm{PDE}_i)}$ the data with only the ith participant's PDE score removed from our multidimensional response data. Then, define a CPO using the posterior predictive density of $\mathrm{PDE_i}$ given $\mathcal{D}_{obs}^{(-\mathrm{PDE}_i)}$, i.e.,

$$\mathrm{CPO}_i^{(-\mathrm{PDE}_i)} = \pi\left(\mathrm{PDE}_i | \mathcal{D}_{obs}^{(-\mathrm{PDE}_i)}\right) = \int\int\int f(\mathrm{PDE}_i|\boldsymbol{\gamma}, \theta_i, \tau_i)\pi(\boldsymbol{\gamma}, \theta_i, \tau_i|\mathcal{D}_{obs}^{(-\mathrm{PDE}_i)})d\boldsymbol{\gamma}d\theta_i d\tau_i,$$

Bayesian standardized residual for ith subject is defined as



(a) Item 2 (111 participants took)

(b) Item 20 (95 participants took)

(c) Item 28 (28 participants took)

(d) Item 42 (57 participants took)

Figure S.7: The Item-based residuals for the RT data

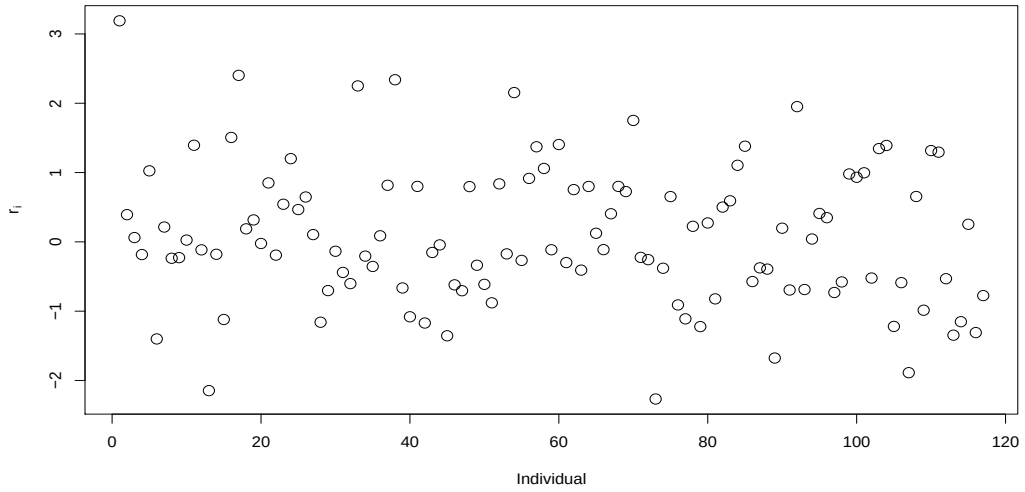


Figure S.8: Bayesian residual plots of r_i versus participants for PDE data

$$r_i = \frac{\text{PDE}_i - \text{E}(\text{PDE}_i \mid \mathcal{D}_{obs}^{(-\text{PDE}_i)})}{\sqrt{\text{Var}(\text{PDE}_i \mid \mathcal{D}_{obs}^{(-\text{PDE}_i)})}} \tag{3}$$

the key quantities in equation (3) can be approximated by using a posterior sampling-based method. Notice that $\boldsymbol{\gamma}^{[m]}$, $\theta_i^{[m]}$ and $\tau_i^{[m]}$ for $m = 1, \cdots, M$ are MCMC draws from the joint posterior of the joint model given $\mathcal{D}_{obs}$. Then,

$$\begin{aligned}
\widehat{\text{E}}(\text{PDE}_i \mid \mathcal{D}_{obs}^{(-\text{PDE}_i)}) &= \widehat{\text{CPO}}_i^{(-\text{PDE}_i)} \times \frac{1}{M} \sum_{m=1}^{M} \frac{\beta_0^{[m]} + \beta_1^{[m]}\theta_i^{[m]} + \beta_2^{[m]}\tau_i^{[m]}}{f(\text{PDE}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]}, \tau_i^{[m]})}, \\
\widehat{\text{Var}}(\text{PDE}_i \mid \mathcal{D}_{obs}^{(-\text{PDE}_i)}) &= \widehat{\text{CPO}}_i^{(-\text{PDE}_i)} \times \frac{1}{M} \sum_{m=1}^{M} \frac{\sigma_{\text{PDE}}^{2[m]} + (\beta_0^{[m]} + \beta_1^{[m]}\theta_i^{[m]} + \beta_2^{[m]}\tau_i^{[m]})^2}{f(\text{PDE}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]}, \tau_i^{[m]})} \\
&\quad - \left[\widehat{\text{E}}\left(\text{PDE}_i \mid \mathcal{D}_{obs}^{(-\text{PDE}_i)}\right)\right]^2,
\end{aligned}$$

where $\widehat{\text{CPO}}_i^{(-\text{PDE}_i)} = M / \sum_{m=1}^{M} \left[f(\text{PDE}_i \mid \boldsymbol{\gamma}^{[m]}, \theta_i^{[m]}, \tau_i^{[m]})\right]^{-1}$.

We present a simple scatter plot of the residual r_i by plugging in those quantites into equation 3 versus the participant in Figure S.8 to show any potential outliers in the PDE data. From this figure, we see that most points are uniformly spread out above and below zero, which indicates the normal model used in our analysis fit the PDE data well.

S.4 The Trace, Autocorrelation Function, and Density Plots

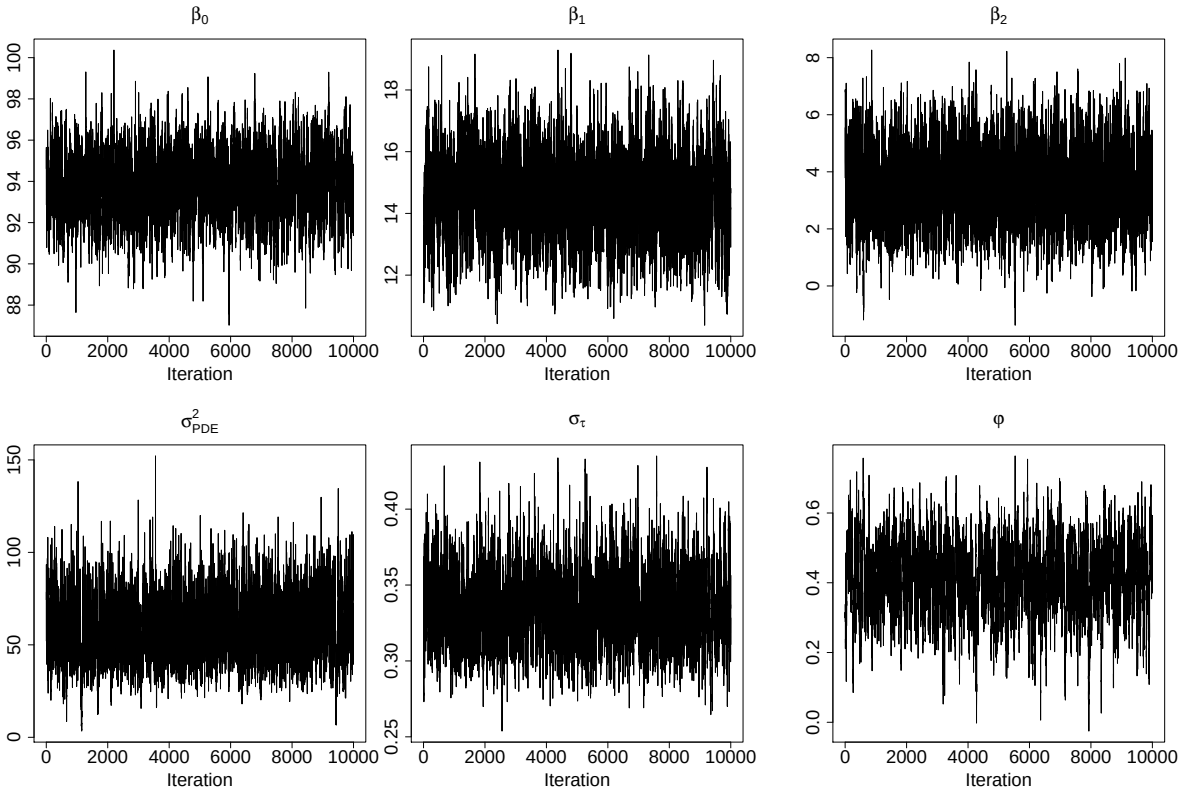


Figure S.9: The trace plots for $\beta_0, \beta_1, \beta_2, \sigma^2_{\text{PDE}}, \sigma_\tau, \varphi$.

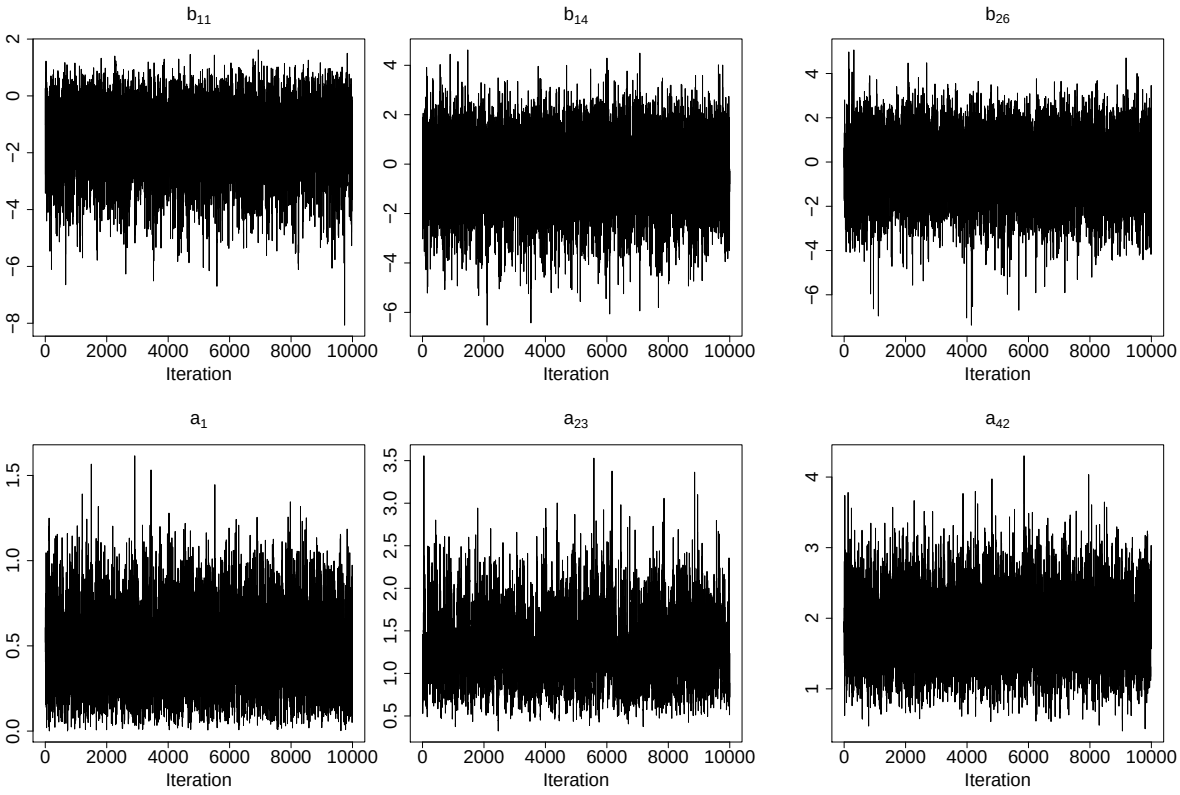


Figure S.10: The trace plots of a_j and b_j for some j.

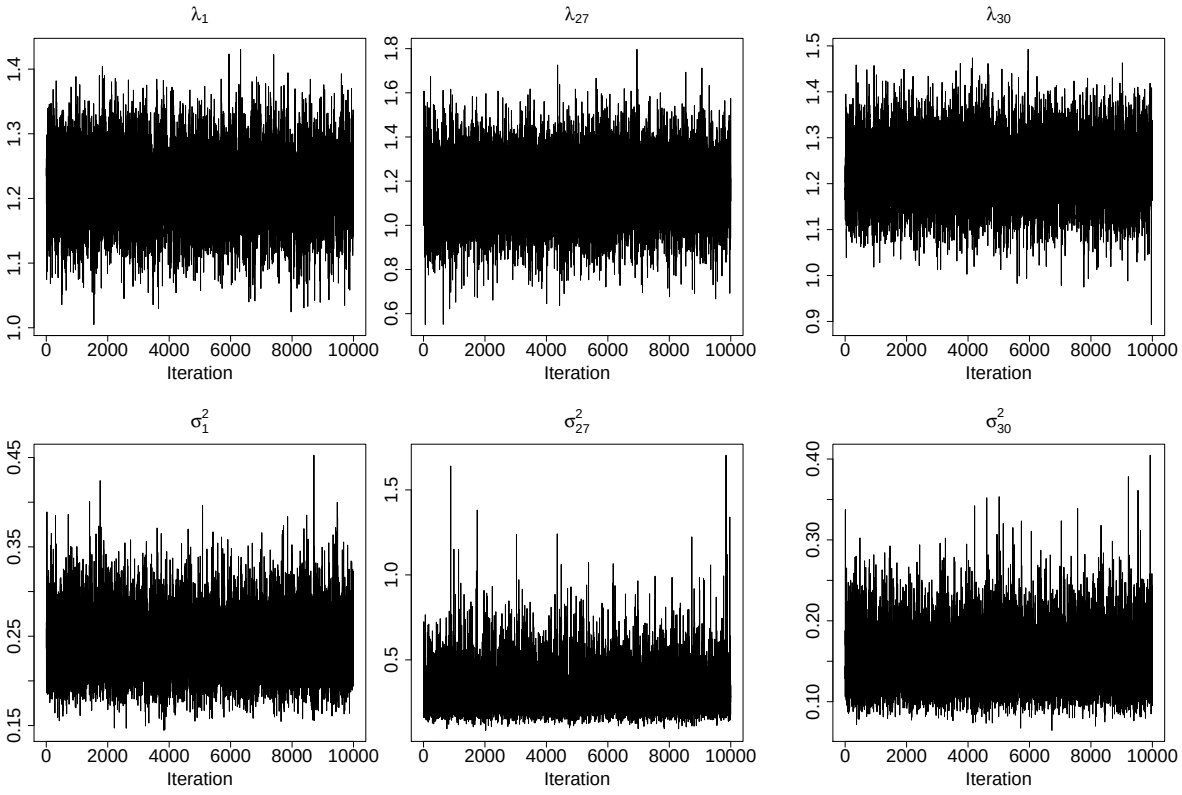


Figure S.11: The trace plots of λ_j and σ_j^2 for some j.

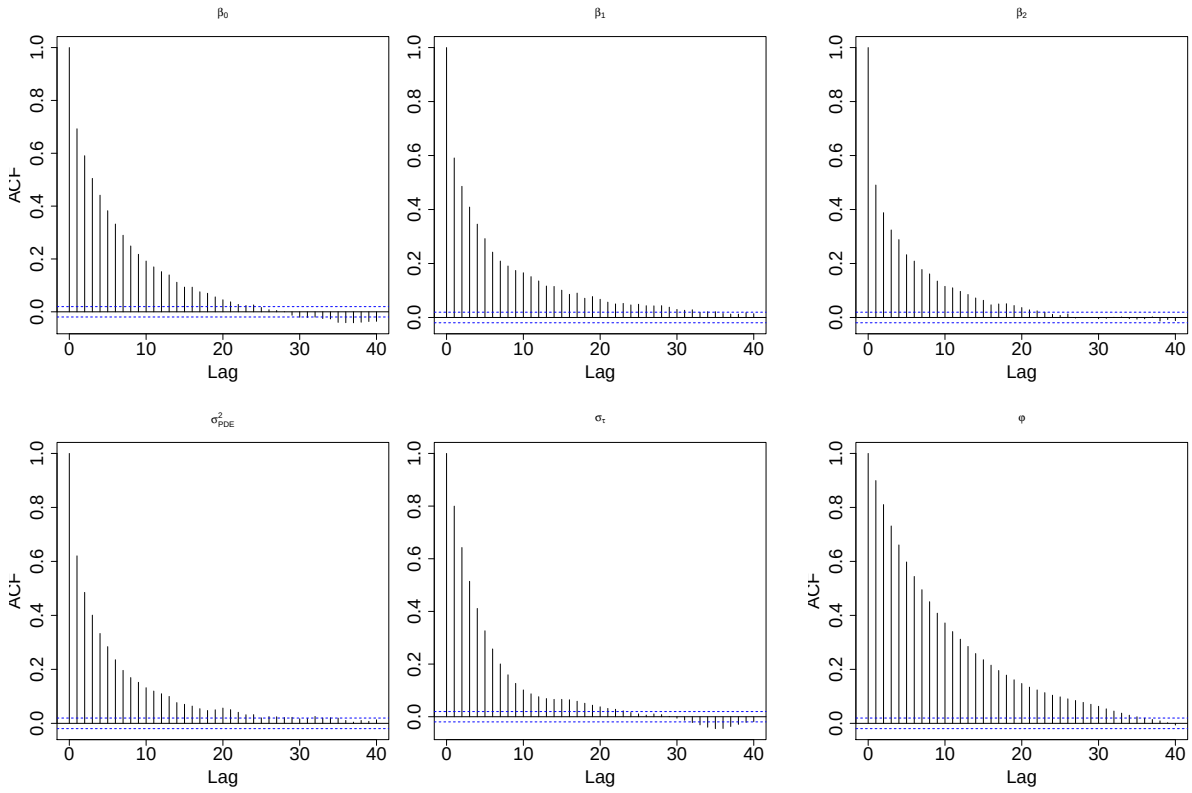


Figure S.12: The Acf plots for $\beta_0, \beta_1, \beta_2, \sigma^2_{\mathrm{PDE}}, \sigma_\tau, \varphi$.

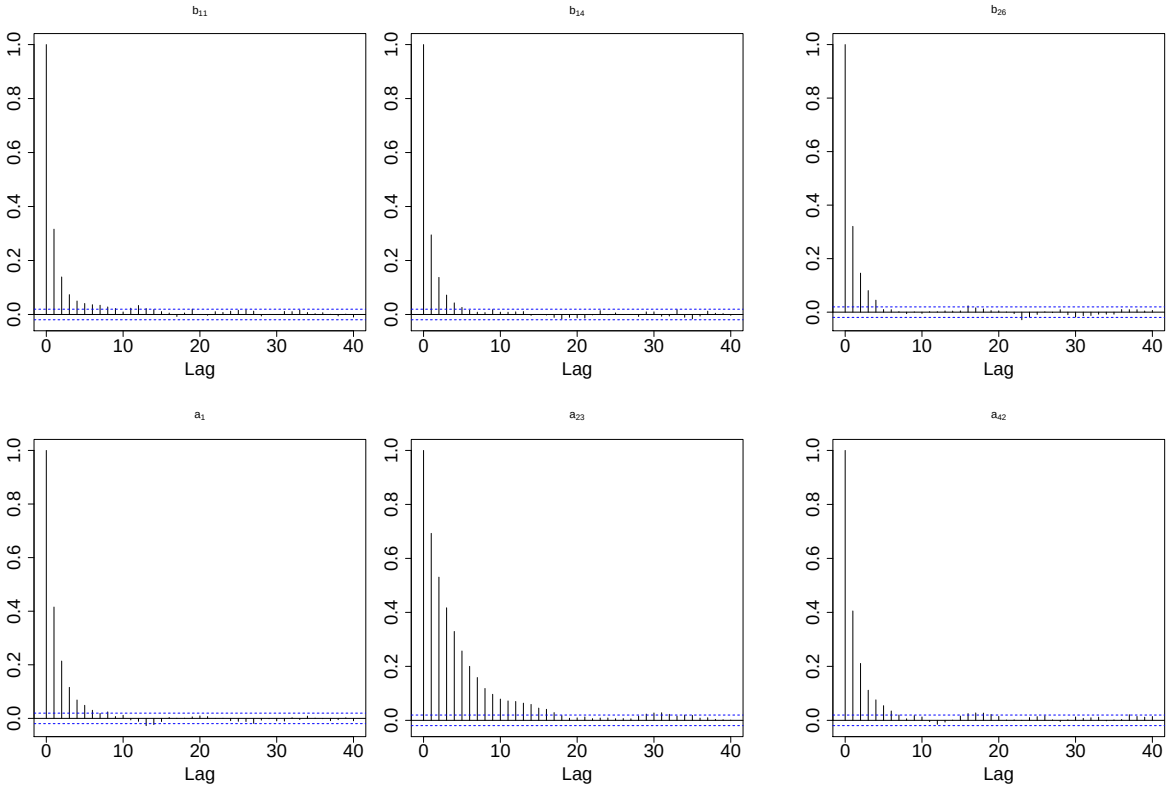


Figure S.13: The Acf plots of a_j and b_j for some j.

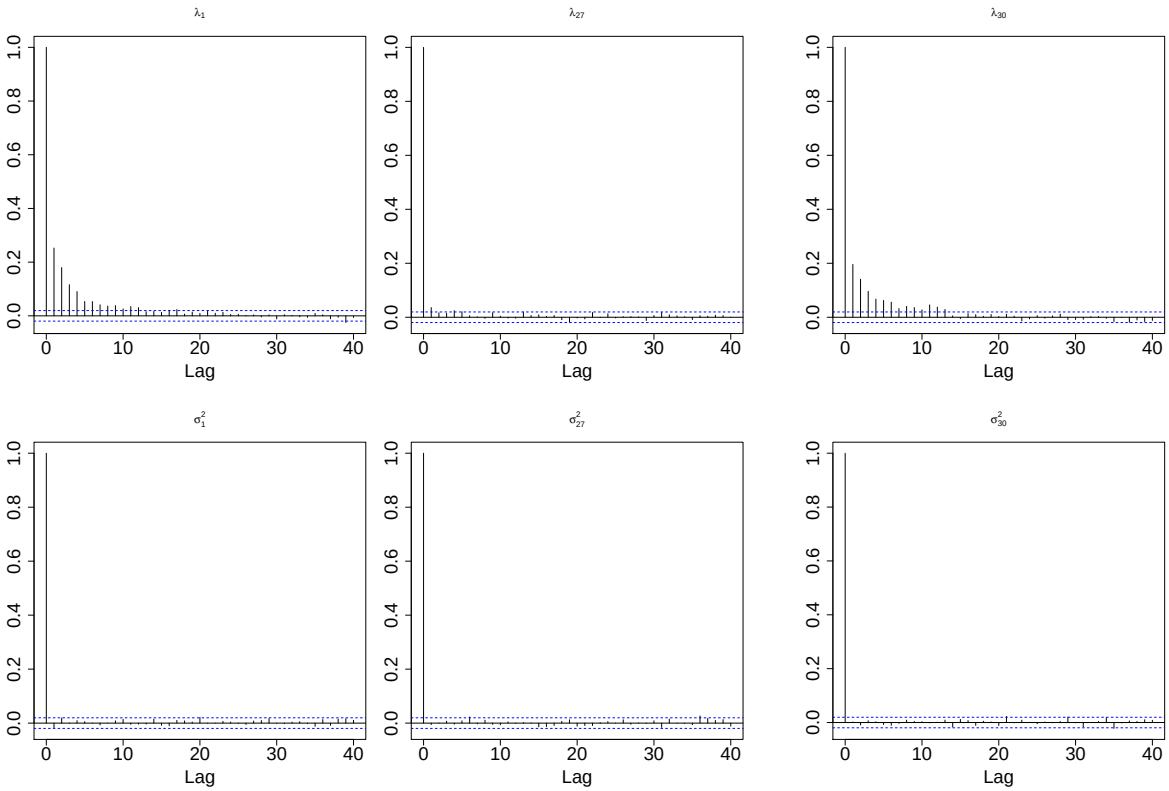


Figure S.14: The Acf plots of λ_j and σ_j^2 for some j.

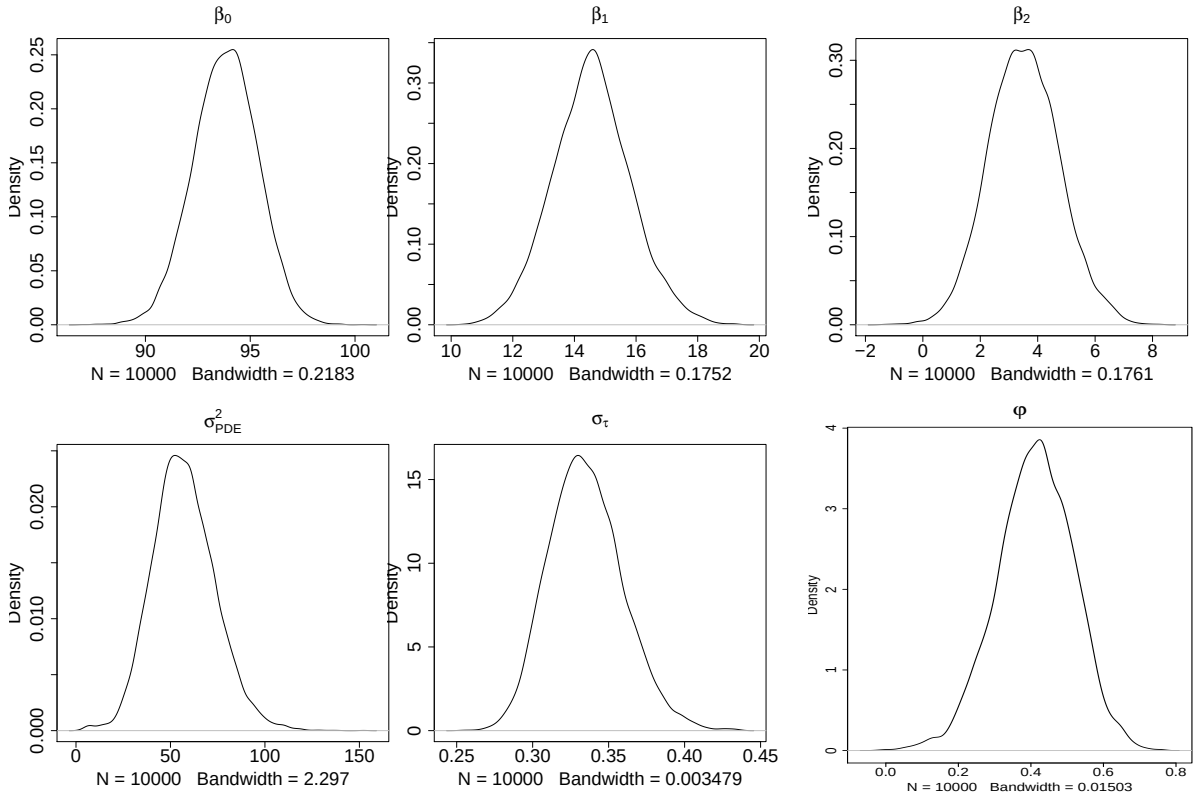


Figure S.15: The Density plots for $\beta_0, \beta_1, \beta_2, \sigma^2_{\mathrm{PDE}}, \sigma_\tau, \varphi$.

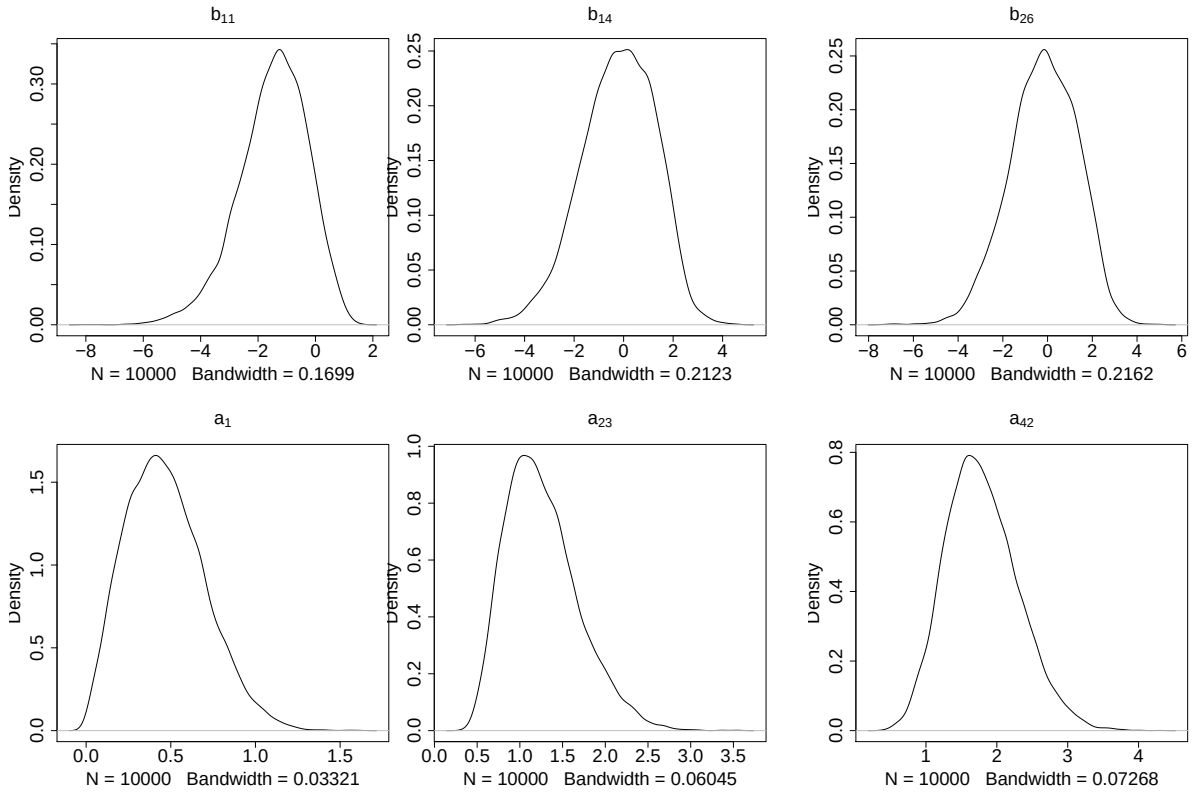


Figure S.16: The Density plots of a_j and b_j for some j.

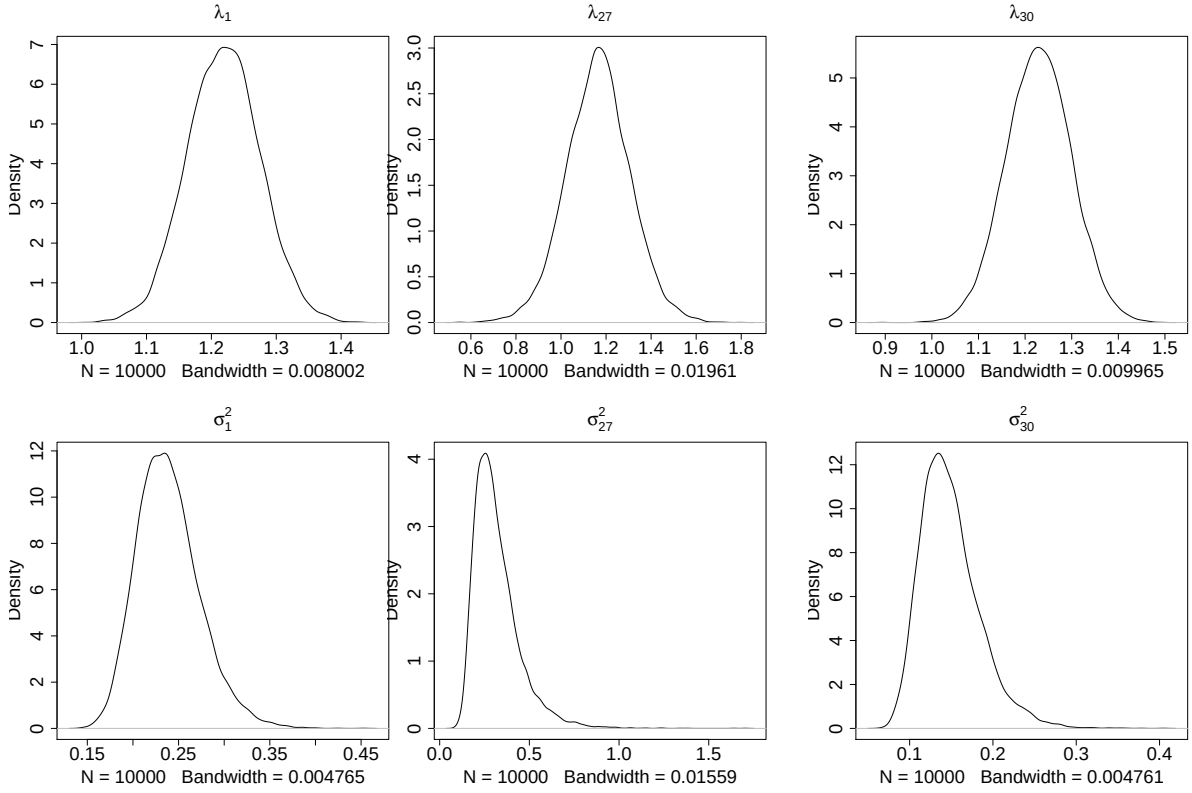


Figure S.17: The Density plots of λ_j and σ_j^2 for some j.

S.5 Calibration of the DIC and LPML Assessment Criteria

In this section, we develop a posterior-predictive approach to obtain a calibration distribution. Let $\pi(\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R \mid \mathcal{D}_{obs})$ denote the augmented joint posterior distribution of $\boldsymbol{\gamma}$, $\boldsymbol{\theta}^R$ and $\boldsymbol{\tau}^R$ given in Section 3. Also, let $\mathcal{D}_{\text{new}}$ denote the future data, which has the same structure as $\mathcal{D}_{obs}$. Furthermore, we let $f(\mathcal{D}_{\text{new}}|\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R)$ denote the density of $\mathcal{D}_{\text{new}}$ under the proposed joint model given in equations 2.7-2.9. Then the calibration distribution of $\mathcal{D}_{\text{new}}$ is defined as

$$f(\mathcal{D}_{\text{new}}|\mathcal{D}_{obs}) = \int f(\mathcal{D}_{\text{new}}|\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R)\pi(\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R \mid \mathcal{D}_{obs})d\boldsymbol{\gamma}d\boldsymbol{\theta}^R d\boldsymbol{\tau}^R.$$

Let DIC_{new} and LPML_{new} denote two DIC and LPML assessment criteria of interest calculated based on $\mathcal{D}_{\text{new}}$. Also let DIC_{obs} and LPML_{obs} denote the corresponding two DIC and LPML assessment criteria of interest calculated based on $\mathcal{D}_{obs}$. The calibration algorithm for quantifying uncertainty of the model assessment criteria is given as follows:

Step 1. Generate $(\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R)$ from $\pi(\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R \mid \mathcal{D}_{obs})$;

Step 2. Sample $\mathcal{D}_{\text{new}}$ from $f(\mathcal{D}_{\text{new}}|\boldsymbol{\gamma}, \boldsymbol{\theta}^R, \boldsymbol{\tau}^R)$;

Step 3. Compute DIC_{new} and LPML_{new}; and

Step 4. Repeat Step 1 - Step 4 independently B times to obtain $\{(\text{DIC}_{\text{new},b}, \text{LPML}_{\text{new},b}), \ b = 1, \ldots, B\}$.

Step 5. Use the predictive samples from Step 4 to calculate the sample standard deviations (SDs), denoted by $\text{SD}(\text{DIC}_{obs})$ and $\text{SD}(\text{LPML}_{obs})$, or the 95% predictive intervals for qualifying the uncertainty of DIC_{obs} and LPML_{obs}.

We apply the above algorithm to compute the SDs of the model assessment criteria in the simulation study as well as in the empirical study. For the simulation study, we

only consider $N = 125$ and $J = 20$. The results are reported in Table S.11. From this table, we see that all SEs of the model assessment criteria fall within the respective ranges of SDs. For the AppRISE data, the values of SDs for these difference assessment criteria are shown in Table S.12.

Table S.11: Mean, SE, and SD values for all variations of ΔDIC and ΔLPML discussed in Section 4 for simulation of Section 5.

	Mean	SE	Range (Min, Max) of SD
$\Delta\text{DIC}_{\text{AC}}$	121.961	14.548	(11.654, 27.772)
$\Delta\text{DIC}^{*}_{\text{AC}}$	16.876	4.893	(4.105, 9.774)
$\Delta\text{LPML}_{\text{AC}}$	63.126	7.302	(5.807, 13.953)
$\Delta\text{DIC}_{\text{RT}}$	38.533	8.762	(7.054, 17.509)
$\Delta\text{DIC}^{*}_{\text{RT}}$	15.358	4.760	(4.272, 9.488)
$\Delta\text{LPML}_{\text{RT}}$	20.612	4.335	(3.574, 9.141)
$\Delta\text{DIC}_{\text{PDE}}$	145.017	15.176	(13.367, 26.925)
$\Delta\text{LPML}_{\text{PDE}}$	74.211	7.860	(6.588, 18.859)

Table S.12: Values (SDs) of Model Assessment Criteria for the AppRISE Data

$\Delta\text{DIC}_{\text{AC}}$	$\Delta\text{DIC}^{*}_{\text{AC}}$	$\Delta\text{LPML}_{\text{AC}}$	$\Delta\text{DIC}_{\text{PDE}}$
106.021 (20.251)	11.824 (7.173)	55.020 (9.998)	133.210 (22.098)
$\Delta\text{DIC}_{\text{RT}}$	$\Delta\text{DIC}^{*}_{\text{RT}}$	$\Delta\text{LPML}_{\text{RT}}$	$\Delta\text{LPML}_{\text{PDE}}$
35.615 (10.469)	12.376 (6.393)	19.523 (5.406)	68.687 (11.946)

S.6 The Codes for MCMC Sampling and Decomposition of DIC and LPML

S.6.1 R Codes for MCMC Sampling

First, we present the R simulation codes, which include setting the true values for the parameters, generating data and fitting data in the 'nimble' language.

(i) **SetTruePara_Simulated.R**

```
## Set true values for parameters in Simulation
# index: seed for generating seed of fitting and generating
# simulation data under different N and J
rm(list = ls())
repli = 500
N = 125
J = 20
index = (N==125)*(J==20)+2*(N==125)*(J==40)+3*(N==250)*(J==20)+4*(N==250)*(J==40)
set.seed(index)
repli = 500
seed.mcmcjoint = sample(1:10000, repli)
seed.data.ac = sample(1:10000, repli)
seed.data.rt = sample(1:10000, repli)
seed.data.pde = sample(1:10000, repli)
seed.mcmcaconly = sample(1:10000, repli)
seed.mcmcrtonly = sample(1:10000, repli)
seed.mcmcpdeonly = sample(1:10000, repli)
seed.mcmcac_rtpde = sample(1:10000, repli)
seed.mcmcrt_acpde = sample(1:10000, repli)
seed.mcmcpde_acrt = sample(1:10000, repli)
```

```
# Nit[i]: the numbers of items that subject i take
Nit = rep(J, N)
# each row of stu_item represents the item index for each subject
stu_item = matrix(rep(1:J, N), N, J, byrow = TRUE)
# set true value
varphi = 0.411
beta0 = 93.8
beta1 = 14.5
beta2 = 3.5
sigma2_pde = 57
w = -1
sigma_tau = exp(w)
mub=0
sigma2b=1
set.seed(3705)
b=rnorm(J, mean = mub, sd = sqrt(sigma2b))
set.seed(290)
a  = runif(J, 0.3, 1.9)
set.seed(369879)
th = rnorm(N, mean = 0, sd = 1)
stu_p=matrix(NA,N,J)
for (i in 1:N) {
  for (j in 1:J) {
    ep=a[j]*(th[i]-b[j])
    p=ifelse(ep<=0,exp(ep)/(1+exp(ep)),1/(1+exp(-ep)))
    stu_p[i,j]=p }}
set.seed(234578)
lambda = runif(J, 0.8, 1.9)
```

```
set.seed(54633)
sigma2 = runif(J, 0.14, 0.75)
set.seed(4032)
tau = rnorm(N, 0, 1)
truePara = c(a, b, beta0, beta1, beta2, lambda, mub,
             sigma2, sigma2b, sigma2_pde, tau, th,
             varphi, w)
##.....basic setting for nimble
Path0 = Sys.getenv('PATH')
Pathrtools = paste( "C:\\Rtools\\bin;C:\\Rtools\\mingw_64\\bin;",Path0 ,sep="" )
Sys.setenv( PATH = Pathrtools)
```

Then, we introduce nimble fitting under joint modelling as follow,

(ii) **joint.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    mu_PDE[i] <- beta0 + beta1*theta[i] + beta2*tau[i]
    PDE[i] ~ dnorm(mean = mu_PDE[i], var = sigma2_pde)
    for (j in 1:Nit[i]){
      ind[i, j] <- a[stu_item[i, j]]*(theta[i] - b[stu_item[i, j]])
      p[i, j] <- 1/(1+exp(-ind[i, j]))*(ind[i, j] > 0)
                 + exp(ind[i, j])/(1+exp(ind[i, j]))*(ind[i, j] <= 0)
      stu_resp[i, j] ~ dbern(prob = p[i, j])
      mu_rt[i, j] <- lambda[stu_item[i, j]]
                   - exp(w)*(sin(varphi)*theta[i]+cos(varphi)*tau[i])
```

```
      rt[i, j] ~ dnorm(mean = mu_rt[i,j], var = sigma2[stu_item[i, j]])
    }
  }
  for(i in 1:N)
  {
    theta[i] ~ dnorm(mean = 0, sd = 1)
    tau[i] ~ dnorm(mean = 0, sd = 1)
  }
  for(j in 1:J)
  {
    a[j] ~ T(dnorm(mean = 0, sd = 1), 0, )
    b[j] ~ dnorm(mean = mu_b, var = sigma2_b)
    sigma2[j] ~ dinvgamma(shape = 0.1, scale = 0.1)
    lambda[j] ~ dnorm(mean = 0, sd = 10)
  }
  mu_b ~ dnorm(mean = 0, var = 10*sigma2_b)
  sigma2_b ~ dinvgamma(shape = 0.1, scale = 0.1)
  w ~ dnorm(mean = 0, sd = 10)
  varphi ~ dunif(-pi/2, pi/2)
  sigma2_pde ~ dinvgamma(shape = 0.1, scale = 0.1)
  beta0 ~ dnorm(mean = 0, sd = 10)
  beta1 ~ dnorm(mean = 0, sd = 10)
  beta2 ~ dnorm(mean = 0, sd = 10)
})
constants <- list(N = N, J = J, Nit = Nit, stu_item = stu_item, pi = pi)
jointModel <- nimbleModel(joint,
                          constants = constants,
                          check = TRUE)
```

```
for (k in 1:500) {
  set.seed(seed.data.ac[k])
  stu_resp = matrix(,N, J)
  for (i in 1:N) {
    for (j in 1:J) {
      stu_resp[i,j]=ifelse(runif(1)<=stu_p[i, j], 1, 0)
    }
  }
  write.csv(stu_resp,paste0('stu_resp',N,'_',J,'_',k,'.csv'))
  set.seed(seed.data.rt[k])
  rt=matrix(NA,N,J)
  tau_star = sigma_tau*(th*sin(varphi)+tau*cos(varphi))
  for (i in 1:N) {
    for (j in 1:J) {
      mu=lambda[j]-tau_star[i]
      rt[i,j]=rnorm(1,mu,sd=sqrt(sigma2[j]))
    }
  }
  write.csv(rt,paste0('rt',N,'_',J,'_',k,'.csv'))
  set.seed(seed.data.pde[k])
  PDE=rep(0,N)
  for(i in 1:N){
      PDE[i]=rnorm(1,beta0+beta1*th[i]+beta2*tau[i],sd=sqrt(sigma2_pde))
  }
  write.csv(PDE,paste0('PDE',N,'_',J,'_',k,'.csv'))
  set.seed(seed.mcmcjoint[k])
  data <- list( rt = rt, stu_resp = stu_resp, PDE = PDE)
  inits <- list(beta0 = 0, beta1 = 2, beta2 = 2, sigma2_pde = 1,
```

```
              mu_b = 0, sigma2_b = 2,
              varphi = 0, w = 2, sigma2 = rep(1, J), theta = rnorm(N),
              a = rep(1, J), b = rnorm(J), tau = rnorm(N), lambda = rnorm(J))
jointModel$setData(data)
jointModel$setInits(inits)
samples <- nimbleMCMC(model = jointModel,
                      niter = 25000, nchains = 1, nburnin = 5000,thin=2,
                      monitors = c("sigma2_pde","beta0","beta1","beta2",
                                   "mu_b","sigma2_b","w","varphi","sigma2",
                                   "a", "b","lambda","theta","tau")
)
write.csv(samples,paste0('samples_resprtpde_',N,'_',J,'_',k,'.csv'))
summary <- matrix(0, 4*J + 8 + 2*N, 7)
colnames(summary) <- c('true', 'eap', 'bias', 'sd', 'lower', 'Upper', 'cp')
rownames(summary) <- colnames(samples)
eap = colMeans(samples)
sd = apply(samples, 2, sd)
require(boa)
hpd = apply(samples, 2, boa.hpd, alpha = 0.05)
summary[, 'true'] = truePara
summary[, 'eap'] = eap
summary[, 'bias'] = summary[, 'eap'] - summary[, 'true']
summary[, 'sd'] = sd
summary[, c('lower', 'Upper')] = t(hpd)
gh1 = truePara-hpd[1, ]
gh2 = truePara-hpd[2, ]
summary[, 'cp']=ifelse(gh1*gh2<=0,1,0)
write.csv(summary, paste0('summ_resprtpde_',N,'_',J,'_',k,'.csv'))
```

```
}
```

The response only, response time only and PDE only model fitting codes are given as follows.

(iii) **onlyresp.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    for (j in 1:Nit[i]){
      ind[i, j] <- a[stu_item[i, j]]*(theta[i] - b[stu_item[i, j]])
      p[i, j] <- 1/(1+exp(-ind[i, j]))*(ind[i, j]>0) +
                 exp(ind[i, j])/(1+exp(ind[i, j]))*(ind[i, j]<=0)
      stu_resp[i, j] ~ dbern(prob = p[i, j])
      }
  }
  for(i in 1:N)
  {
    theta[i] ~ dnorm(mean = 0, sd = 1)
  }
  for(j in 1:J)
  {
    a[j] ~ T(dnorm(mean = 0, sd = 1), 0, )
    b[j] ~ dnorm(mean = mu_b, var = sigma2_b)
  }
  mu_b ~ dnorm(mean = 0, var = 10*sigma2_b)
  sigma2_b ~ dinvgamma(shape = 0.1, scale = 0.1)
```

```
})
constants <- list(N = N, J = J, Nit = Nit, stu_item = stu_item)
jointModel <- nimbleModel(joint,
                          constants = constants,
                          check = TRUE)
for (k in 1:500) {
  set.seed(seed.data.ac[k])
  stu_resp = matrix(,N, J)
  for (i in 1:N) {
    for (j in 1:J) {
      stu_resp[i,j]=ifelse(runif(1)<=stu_p[i, j], 1, 0)
    }
  }
   set.seed(seed.mcmcaconly[k])
   data <- list(stu_resp = stu_resp)
   inits <- list(mu_b = 0, sigma2_b = 2, theta = rnorm(N),
                 a = rep(1, J), b = rnorm(J))
   jointModel$setData(data)
   jointModel$setInits(inits)
   samples <- nimbleMCMC(model = jointModel,
                          niter = 25000, nchains = 1, nburnin = 5000,thin=2,
                          monitors = c("mu_b","sigma2_b","a", "b","theta"))
   write.csv(samples,paste0('samples_responly_',N,'_',J,'_',k,'.csv'))
}
```

(iv) **onlyrt.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
```

```
DICLPML_rtonly=function(samples, N,J, rt, Nit, stu_item ){
  mvdnorm = function(x, mu, Sigma){
    K = length(x)
    logf = -K/2*log(2*pi)-0.5*log(det(Sigma))-0.5*t(x-mu)%*%solve(Sigma)%*%(x-mu)
    return(exp(logf))
  }
  logf = matrix(, nrow(samples), N)
  Dev = vector(length = nrow(samples))
  mcmc_lambda=samples[,paste0('lambda[',1:J,']')]
  mcmc_sigma2=samples[,paste0('sigma2[',1:J,']')]
  mcmc_w=samples[,'w']
  require(MASS)
  for (k in 1:nrow(samples)) {
    lambda = mcmc_lambda[k,]
    sigma2 = mcmc_sigma2[k,]
    sigma_tau = exp(mcmc_w[k])
    for (i in 1:N) {
      t = rt[i, 1:Nit[i]]
      item = stu_item[i, 1:Nit[i]]
      mu = lambda[item]
      Sigma = matrix(sigma_tau^2, Nit[i], Nit[i])+diag(sigma2[item])
      logf[k, i] = log(mvdnorm(t, mu, Sigma))
    }
    Dev[k] = -2*sum(logf[k, ])
  }
  Dev_bar = mean(Dev)
  lambda = colMeans(mcmc_lambda)
  sigma2 = colMeans(mcmc_sigma2)
```

```
  sigma_tau = mean(exp(mcmc_w))
  Dev_hat = 0
  for (i in 1:N) {
    t = rt[i, 1:Nit[i]]
    item = stu_item[i, 1:Nit[i]]
    mu = lambda[item]
    Sigma = matrix(sigma_tau^2, Nit[i], Nit[i])+diag(sigma2[item])
    Dev_hat =Dev_hat -2*log(mvdnorm(t, mu, Sigma))
  }
  PD = Dev_bar - Dev_hat
  DIC = Dev_hat + 2*PD
  Umax = apply(-logf, 2, max)
  H = exp(-logf - outer(rep(1, nrow(samples)), Umax))
  logCPO=-log(colMeans(H))-Umax
  LPML=sum(logCPO)
  Cri=c(DIC,PD,LPML)
}
require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    for (j in 1:Nit[i]){
      mu_rt[i, j] <- lambda[stu_item[i, j]] - exp(w)*tau[i]
      rt[i, j] ~ dnorm(mean = mu_rt[i, j], var = sigma2[stu_item[i, j]])
    }
  }
  for(i in 1:N)
  {
    tau[i] ~ dnorm(mean = 0, sd = 1)
```

```
  }
  for(j in 1:J)
  {
    sigma2[j] ~ dinvgamma(shape = 0.1, scale = 0.1)
    lambda[j] ~ dnorm(mean = 0, sd = 10)
  }
  w ~ dnorm(mean = 0, sd = 10)
})
constants <- list(N = N, J = J, Nit = Nit, stu_item = stu_item)
jointModel <- nimbleModel(joint,
                          constants = constants,
                          check = TRUE)
diclpml=matrix(,repli, 3)
for (k in 1:500) {
  set.seed(seed.data.rt[k])
  rt=matrix(NA,N,J)
  tau_star = sigma_tau*(th*sin(varphi)+tau*cos(varphi))
  for (i in 1:N) {
    for (j in 1:J) {
      mu=lambda[j]-tau_star[i]
      rt[i,j]=rnorm(1,mu,sd=sqrt(sigma2[j]))
    }
  }
  set.seed(seed.mcmcrtonly[k])
  data <- list( rt = rt)
  inits <- list(w = 2, sigma2 = rep(1, J), lambda=rnorm(J), tau = rnorm(N))
  jointModel$setData(data)
  jointModel$setInits(inits)
```

```
  samples <- nimbleMCMC(model = jointModel,
                        niter = 25000, nchains = 1, nburnin = 5000, thin = 2,
                        monitors = c("w", "sigma2", "lambda", "tau"))
  diclpml[k,]=DICLPML_rtonly(samples, N, J,rt, Nit, stu_item )
  write.csv(diclpml, paste0('diclpml_rtonly_',N,'_',J,'.csv'))
}
```

(v) **onlypde.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
DICLPML_pdeonly=function(samples,N){
  nlen=nrow(samples)
  u=matrix(, nlen, N)
  dev=rep(0, nlen)
  for (m in 1:nlen) {
    mu_PDE = samples[m, 'mu_PDE']
    sigma2_pde = samples[m, 'sigma2_pde']
    u[m,] = dnorm(PDE, mu_PDE, sd = sqrt(sigma2_pde), log=TRUE)
    dev[m] = -2*sum(u[m,])
  }
  eap = colMeans(samples)
  mu_PDE_hat = eap['mu_PDE']
  sigma2_pde_hat = eap['sigma2_pde']
  dev_hat = -2*sum(dnorm(PDE, mu_PDE_hat, sd = sqrt(sigma2_pde_hat), log=TRUE))
  dev_bar = mean(dev)
  pd = dev_bar - dev_hat
  dic = dev_hat + 2*pd
  cpo=1/(colMeans(1/exp(u)))
```

```
  lpml=sum(log(cpo))
  h=c(dic,pd,lpml)
}

require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    PDE[i] ~ dnorm(mean = mu_PDE, var = sigma2_pde)
  }
  mu_PDE ~ dnorm(mean = 0, sd = 10)
  sigma2_pde ~ dinvgamma(shape = 0.1, scale = 0.1)
})
constants <- list(N = N)
jointModel <- nimbleModel(joint, constants = constants,check = TRUE)
diclpml=matrix(,repli, 3)
colnames(diclpml) = c('dic', 'pd', 'lpml')
for (k in 1:500) {
  set.seed(seed.data.pde[k])
  PDE=rep(0,N)
  for(i in 1:N){
    PDE[i]=rnorm(1,beta0+beta1*th[i]+beta2*tau[i],sd=sqrt(sigma2_pde))
  }
  set.seed(seed.mcmcpdeonly[k])
  data <- list( PDE = PDE)
  inits <- list(sigma2_pde = 58, mu_PDE = 0)
  jointModel$setData(data)
  jointModel$setInits(inits)
  samples <- nimbleMCMC(model = jointModel,
```

```
                        niter = 25000, nchains = 1, nburnin = 5000, thin = 2,
                        monitors = c("sigma2_pde", "mu_PDE"))
  diclpml[k,]=DICLPML_pdeonly(samples,N)
  write.csv(diclpml,paste0('DICLPML_pdeonly_',N,'_',J,'.csv'))
}
```

Below are the MCMC codes for sampling from the conditional distributions.

(vi) **cond_resp_rtpde.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    mu_PDE[i] <- beta0 + beta1*theta[i] + beta2*tau[i]
    PDE[i] ~ dnorm(mean = mu_PDE[i], var = sigma2_pde)
    for (j in 1:Nit[i]){
      ind[i, j] <- a[stu_item[i, j]]*(theta[i] - b[stu_item[i, j]])
      p[i, j] <- 1/(1+exp(-ind[i, j]))*(ind[i, j] > 0)
             + exp(ind[i, j])/(1+exp(ind[i, j]))*(ind[i, j] <= 0)
      stu_resp[i, j] ~ dbern(prob = p[i, j])
      mu_rt[i, j] <- lambda[stu_item[i, j]]
                  -exp(w)*(sin(varphi)*theta[i]+cos(varphi)*tau[i])
      rt[i, j] ~ dnorm(mean = mu_rt[i,j], var = sigma2[stu_item[i, j]])
    }
  }
  for(i in 1:N)
  {
    theta[i] ~ dnorm(mean = 0, sd = 1)
```

```
    tau[i] ~ dnorm(mean = 0, sd = 1)
  }
  for(j in 1:J)
  {
    a[j] ~ T(dnorm(mean = 0, sd = 1), 0, )
    b[j] ~ dnorm(mean = mu_b, var = sigma2_b)
  }
  mu_b ~ dnorm(mean = 0, var = 10*sigma2_b)
  sigma2_b ~ dinvgamma(shape = 0.1, scale = 0.1)
})
for (k in 1:500) {
  set.seed(seed.data.ac[k])
  stu_resp = matrix(,N, J)
  for (i in 1:N) {
    for (j in 1:J) {
      stu_resp[i,j]=ifelse(runif(1)<=stu_p[i, j], 1, 0)
    }
  }
  set.seed(seed.data.rt[k])
  rt=matrix(NA,N,J)
  tau_star = sigma_tau*(th*sin(varphi)+tau*cos(varphi))
  for (i in 1:N) {
    for (j in 1:J) {
      mu=lambda[j]-tau_star[i]
      rt[i,j]=rnorm(1,mu,sd=sqrt(sigma2[j]))
    }
  }
  set.seed(seed.data.pde[k])
```

```
PDE=rep(0,N)
for(i in 1:N){
  PDE[i]=rnorm(1,beta0+beta1*th[i]+beta2*tau[i],sd=sqrt(sigma2_pde))
}
summ = read.csv(paste0('summ_resprtpde_',N,'_',J,'_',k,'.csv'))
eap = summ$eap
names(eap) = summ$X
eap_beta0 = eap['beta0']
eap_beta1 = eap['beta1']
eap_beta2 = eap['beta2']
eap_sigma2_pde = eap['sigma2_pde']
eap_varphi = eap['varphi']
eap_w = eap['w']
eap_sigma2 = eap[paste0('sigma2[', 1:J, ']')]
eap_lambda = eap[paste0('lambda[', 1:J, ']')]
set.seed(seed.mcmcac_rtpde[k])
data <- list( rt = rt, stu_resp = stu_resp, PDE = PDE)
constants <- list(N = N, J = J, Nit = Nit, stu_item = stu_item, pi = pi,
                  beta0 = eap_beta0, beta1 = eap_beta1, beta2 = eap_beta2,
                  sigma2_pde = eap_sigma2_pde, varphi = eap_varphi,
                  w = eap_w, sigma2 = eap_sigma2, lambda = eap_lambda)
inits <- list( mu_b = 0, sigma2_b = 2, theta = rnorm(N), a = rep(1, J),
               b = rnorm(J), tau = rnorm(N))
jointmodel <- nimbleModel(joint,
                          data = data,
                          constants = constants,
                          inits = inits,
                          check = TRUE)
```

```
  samples <- nimbleMCMC(model = jointmodel,
                        niter = 25000, nchains = 1, nburnin = 5000, thin = 2,
                        monitors = c( "mu_b", "sigma2_b", "a", "b", "theta"))
  write.csv(samples,paste0('samplescond_resp_rtpde_',N,'_',J,'_',k,'.csv'))
}
```

(vii) **cond_rt_resppde.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    mu_PDE[i] <- beta0 + beta1*theta[i] + beta2*tau[i]
    PDE[i] ~ dnorm(mean = mu_PDE[i], var = sigma2_pde)
    for (j in 1:Nit[i]){
      ind[i, j] <- a[stu_item[i, j]]*(theta[i] - b[stu_item[i, j]])
      p[i, j] <- 1/(1+exp(-ind[i, j]))*(ind[i, j] > 0)
                + exp(ind[i, j])/(1+exp(ind[i, j]))*(ind[i, j] <= 0)
      stu_resp[i, j] ~ dbern(prob = p[i, j])
      mu_rt[i, j] <- lambda[stu_item[i, j]]
                   - exp(w)*(sin(varphi)*theta[i]+cos(varphi)*tau[i])
      rt[i, j] ~ dnorm(mean = mu_rt[i,j], var = sigma2[stu_item[i, j]])
    }
  }
  for(i in 1:N)
  {
    theta[i] ~ dnorm(mean = 0, sd = 1)
    tau[i] ~ dnorm(mean = 0, sd = 1)
```

```
  }
  for(j in 1:J)
  {
    sigma2[j] ~ dinvgamma(shape = 0.1, scale = 0.1)
    lambda[j] ~ dnorm(mean = 0, sd = 10)
  }
  w ~ dnorm(mean = 0, sd = 10)
  varphi ~ dunif(-pi/2, pi/2)
})
for (k in 1:500) {
  set.seed(seed.data.ac[k])
  stu_resp = matrix(,N, J)
  for (i in 1:N) {
    for (j in 1:J) {
      stu_resp[i,j]=ifelse(runif(1)<=stu_p[i, j], 1, 0)
    }
  }
  set.seed(seed.data.rt[k])
  rt=matrix(NA,N,J)
  tau_star = sigma_tau*(th*sin(varphi)+tau*cos(varphi))
  for (i in 1:N) {
    for (j in 1:J) {
      mu=lambda[j]-tau_star[i]
      rt[i,j]=rnorm(1,mu,sd=sqrt(sigma2[j]))
    }
  }
  set.seed(seed.data.pde[k])
  PDE=rep(0,N)
```

```
for(i in 1:N){
  PDE[i]=rnorm(1,beta0+beta1*th[i]+beta2*tau[i],sd=sqrt(sigma2_pde))
}
summ = read.csv(paste0('summ_resprtpde_',N,'_',J,'_',k,'.csv'))
eap = summ$eap
names(eap) = summ$X
eap_a = eap[paste0('a[', 1:J, ']')]
eap_b = eap[paste0('b[', 1:J, ']')]
eap_beta0 = eap['beta0']
eap_beta1 = eap['beta1']
eap_beta2 = eap['beta2']
eap_sigma2_pde = eap['sigma2_pde']
set.seed(seed.mcmcrt_acpde[k])
data <- list( rt = rt, stu_resp = stu_resp, PDE = PDE)
constants <- list(N = N, J = J, Nit = Nit, stu_item = stu_item, pi = pi,
              beta0 = eap_beta0, beta1 = eap_beta1, beta2 = eap_beta2,
              sigma2_pde = eap_sigma2_pde, a = eap_a, b = eap_b)
inits <- list(varphi = 0, w = 2, sigma2 = rep(1, J), theta = rnorm(N),
              tau = rnorm(N), lambda = rnorm(J))
jointmodel <- nimbleModel(joint,
                          data = data,
                          constants = constants,
                          inits = inits,
                          check = TRUE)
 samples <- nimbleMCMC(model = jointmodel,
                   niter = 25000, nchains = 1, nburnin = 5000, thin = 2,
                   monitors = c( "w", "varphi", "sigma2", "lambda", "theta"))
write.csv(samples,paste0('samplescond_rt_resppde_',N,'_',J,'_',k,'.csv'))
```

```
}
```

(viii) **cond_pde_acrt.R**

```
rm(list = ls())
source('SetTruePara_Simulated.R')
require(nimble)
joint <- nimbleCode({
  for(i in 1:N){
    mu_PDE[i] <- beta0 + beta1*theta[i] + beta2*tau[i]
    PDE[i] ~ dnorm(mean = mu_PDE[i], var = sigma2_pde)
    for (j in 1:Nit[i]){
      ind[i, j] <- a[stu_item[i, j]]*(theta[i] - b[stu_item[i, j]])
      p[i, j] <- 1/(1+exp(-ind[i, j]))*(ind[i, j] > 0)
                 + exp(ind[i, j])/(1+exp(ind[i, j]))*(ind[i, j] <= 0)
      stu_resp[i, j] ~ dbern(prob = p[i, j])
      mu_rt[i, j] <- lambda[stu_item[i, j]]
                     - exp(w)*(sin(varphi)*theta[i]+cos(varphi)*tau[i])
      rt[i, j] ~ dnorm(mean = mu_rt[i,j], var = sigma2[stu_item[i, j]])
    }
  }
  for(i in 1:N)
  {
    theta[i] ~ dnorm(mean = 0, sd = 1)
    tau[i] ~ dnorm(mean = 0, sd = 1)
  }
  sigma2_pde ~ dinvgamma(shape = 0.1, scale = 0.1)
  beta0 ~ dnorm(mean = 0, sd = 10)
  beta1 ~ dnorm(mean = 0, sd = 10)
```

```
  beta2 ~ dnorm(mean = 0, sd = 10)
})
for (k in 1:500) {
  set.seed(seed.data.ac[k])
  stu_resp = matrix(,N, J)
  for (i in 1:N) {
    for (j in 1:J) {
      stu_resp[i,j]=ifelse(runif(1)<=stu_p[i, j], 1, 0)
    }
  }
  set.seed(seed.data.rt[k])
  rt=matrix(NA,N,J)
  tau_star = sigma_tau*(th*sin(varphi)+tau*cos(varphi))
  for (i in 1:N) {
    for (j in 1:J) {
      mu=lambda[j]-tau_star[i]
      rt[i,j]=rnorm(1,mu,sd=sqrt(sigma2[j]))
    }
  }
  set.seed(seed.data.pde[k])
  PDE=rep(0,N)
  for(i in 1:N){
    PDE[i]=rnorm(1,beta0+beta1*th[i]+beta2*tau[i],sd=sqrt(sigma2_pde))
  }
  summ = read.csv(paste0('summ_resprtpde_',N,'_',J,'_',k,'.csv'))
  eap = summ$eap
  names(eap) = summ$X
  eap_a = eap[paste0('a[', 1:J, ']')]
```

```
  eap_b = eap[paste0('b[', 1:J, ']')]
  eap_varphi = eap['varphi']
  eap_w = eap['w']
  eap_sigma2 = eap[paste0('sigma2[', 1:J, ']')]
  eap_lambda = eap[paste0('lambda[', 1:J, ']')]
  set.seed(seed.mcmcpde_acrt[k])
  data <- list( rt = rt, stu_resp = stu_resp, PDE = PDE)
  constants <- list(N = N, J = J, Nit = Nit, stu_item = stu_item, pi = pi,
                    a = eap_a, b = eap_b, varphi = eap_varphi,
                    w = eap_w, sigma2 = eap_sigma2, lambda = eap_lambda)
  inits <- list(beta0 = 0, beta1 = 2, beta2 = 2, sigma2_pde = 1,
                theta = rnorm(N), tau = rnorm(N))
  jointmodel <- nimbleModel(joint,
                            data = data,
                            constants = constants,
                            inits = inits,
                            check = TRUE)
  samples <- nimbleMCMC(model = jointmodel,
                      niter = 25000, nchains = 1, nburnin = 5000, thin = 2,
                      monitors = c("sigma2_pde", "beta0", "beta1", "beta2",
                                   "theta"))
  write.csv(samples,paste0('samplescond_pde_resprt_',N,'_',J,'_',k,'.csv'))
}
```

With all the MCMC samples available, the decomposition codes include two parts (Matlab and FORTRAN).

S.6.2 Matlab Codes for Decomposition of DIC

Matlab codes shown below are only for computing $\mathrm{DIC}_{[\mathrm{RT,PDE}]}$ and $\mathrm{DIC}_{[\mathrm{RT}]}$.

```
function logf=LOGF_RTPDE_ACRTPDE(varphi,lambda,sigma2,sigma2_tau,
                                 beta0,beta1,beta2,sigma2_pde,
                                 PDE,rt,stu_item,Nit)
    dim=size(rt);
    N=dim(1);
    logf=zeros(N,1);
    for i=1:N
        t=rt(i,1:Nit(i));
        item=stu_item(i,1:Nit(i));
        mu=[lambda(item),beta0];
        Sigma=diag([sigma2(item)+sigma2_tau,beta1^2+beta2^2+sigma2_pde]);
        for j=1:(Nit(i)-1)
            for k=(j+1):(Nit(i))
                Sigma(j,k)=sigma2_tau;
                Sigma(k,j)=Sigma(j,k);
            end
        end
        Sigma(Nit(i)+1,1:Nit(i))=-sin(varphi)*beta1*sqrt(sigma2_tau)...
                                 -cos(varphi)*beta2*sqrt(sigma2_tau);
        Sigma(1:Nit(i),Nit(i)+1)=Sigma(Nit(i)+1,1:Nit(i));
        logf(i)=log(mvnpdf([t,PDE(i)],mu,Sigma));
    end
end

function logf=LOGF_RT_PDEACRT(lambda,sigma2,sigma2_tau,rt,stu_item,Nit)
```

```
    dim=size(rt);
    N=dim(1);
    logf=zeros(N,1);
    for i=1:N
        t=rt(i,1:Nit(i));
        item=stu_item(i,1:Nit(i));
        mu=lambda(item);
        Sigma=diag(sigma2(item)+sigma2_tau);
        for j=1:(Nit(i)-1)
            for k=(j+1):(Nit(i))
                Sigma(j,k)=sigma2_tau;
                Sigma(k,j)=Sigma(j,k);
            end
        end
        logf(i)=log(mvnpdf(t,mu,Sigma));
    end
end
```

S.6.3 FORTRAN Codes for Decomposition of DIC and LPML

We develop the FORTRAN codes to compute all of the other quantities in the decomposition of DIC and LPML except for $\text{DIC}_{[\text{RT,PDE}]}$ and $\text{DIC}_{[\text{RT}]}$. Note that the data should be transformed into a correct format in order to import into FORTRAN. FORTRAN codes mainly focus on the calculation of quantities that involve one-dimensional integrals over θ_i. The key subroutines and user-defined functions are **DICPDcalculate.f**, **floatingControl.f**, and **function.f**.

(ix) **function.f**

```
c.....function for [ACRT]
      real*8 function fnacrt(x)
              implicit real*8 (a-h,o-z)
              parameter(nitem=20)
              parameter(pi=3.141592741012573)
              real*8 x,Ln,yi
              integer irespi(nitem)
              integer indi(nitem),Niti
              real*8  Ln_hati,t(nitem)
              real*8  ak(nitem),bk(nitem)
              real*8  lambdak(nitem),sigma2k(nitem)
              real*8  varphik,sigma2_tauk
              common /va/ak
              common /vb/bk
              common /vsigma2_tau/sigma2_tauk
              common /vlambda/lambdak
              common /vsigma2/sigma2k
              common /vvarphi/varphik
              common /vLn_hati/Ln_hati
              common /virespi/irespi
              common /vNiti/Niti
              common /vindi/indi
              common /vt/t
              Ln=-0.5d0*x**2-0.5d0*(Niti+1)*dlog(2*pi)
              h=1.0d0
              h0=0.0d0
              h1=0.0d0
              yi=0.0d0
```

```
              h2=0.0d0
              do k=1,Niti
                 j=indi(k)
                 h2=h2+dlog(sigma2k(j))
                 ep=ak(j)*(x-bk(j))
                 if(ep.le.0.0d0)then
                         yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
                 else
                         yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
                 endif
                 h=h+dcos(varphik)**2*sigma2_tauk/sigma2k(j)
                 ep1=t(k)-lambdak(j)+dsqrt(sigma2_tauk)*dsin(varphik)*x
                 h0=h0+ep1**2/sigma2k(j)
                 h1=h1+dcos(varphik)*dsqrt(sigma2_tauk)*ep1/sigma2k(j)
              enddo
              Ln=Ln+yi-0.5d0*h2-0.5d0*dlog(h)-0.5d0*h0
     1           +0.5d0/h*h1**2
              fnacrt=dexp(Ln-Ln_hati)
       end
c.....................................................................
c.....function for [ACRTPDE]
      real*8 function fnacrtpde(x)
              implicit real*8 (a-h,o-z)
              parameter(nitem=20)
              parameter(pi=3.141592741012573)
              real*8 x,Ln,yi
              integer irespi(nitem)
              integer indi(nitem),Niti
```

```
      real*8  Ln_hati,t(nitem),PDEi
      real*8  ak(nitem),bk(nitem)
      real*8  lambdak(nitem),sigma2k(nitem)
      real*8  varphik,sigma2_tauk
      real*8  beta0k,beta1k,beta2k,sigma2_pdek
      common /va/ak
      common /vb/bk
      common /vsigma2_tau/sigma2_tauk
      common /vlambda/lambdak
      common /vsigma2/sigma2k
      common /vvarphi/varphik
      common /vLn_hati/Ln_hati
      common /virespi/irespi
      common /vNiti/Niti
      common /vindi/indi
      common /vt/t
      common /vbeta0/beta0k
      common /vbeta1/beta1k
      common /vbeta2/beta2k
      common /vsigma2_pde/sigma2_pdek
      common /vPDEi/PDEi
      Ln=-0.5d0*x**2-0.5d0*(Niti+2)*dlog(2*pi)
     1    -0.5*dlog(sigma2_pdek)
     1    -0.5d0/(sigma2_pdek)*(PDEi-beta0k-beta1k*x)**2
      h=1.0d0+beta2k**2/sigma2_pdek
      h0=0.0d0
      h1=-beta2k*(PDEi-beta0k-beta1k*x)/sigma2_pdek
      yi=0.0d0
```

```
              h2=0.0d0
              do k=1,Niti
                 j=indi(k)
                 h2=h2+dlog(sigma2k(j))
                 ep=ak(j)*(x-bk(j))
                 if(ep.le.0.0d0)then
                         yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
                 else
                         yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
                 endif
                 h=h+dcos(varphik)**2*sigma2_tauk/sigma2k(j)
                 ep1=t(k)-lambdak(j)+dsqrt(sigma2_tauk)*dsin(varphik)*x
                 h0=h0+ep1**2/sigma2k(j)
                 h1=h1+dcos(varphik)*dsqrt(sigma2_tauk)*ep1/sigma2k(j)
              enddo
              Ln=Ln+yi-0.5d0*h2-0.5d0*dlog(h)-0.5d0*h0+0.5d0/h*h1**2
              fnacrtpde=dexp(Ln-Ln_hati)
          end
c.....................................................................
c.....function for [AC|RTPDE]
      real*8 function fnac_rtpde(x)
              implicit real*8 (a-h,o-z)
              parameter(nitem=20)
              parameter(pi=3.141592741012573)
              real*8 x,Ln,yi
              integer irespi(nitem)
              integer indi(nitem),Niti
              real*8  Ln_hati,AA,BB,PDEi
```

```
real*8  t(nitem)
real*8  ak(nitem),bk(nitem)
real*8  sigma2k(nitem)
real*8  lambdak(nitem)
real*8  sigma2_tauk,varphik
real*8  beta0k,beta1k,beta2k,sigma2_pdek
common /va/ak
common /vb/bk
common /virespi/irespi
common /vNiti/Niti
common /vindi/indi
common /vvarphi/varphik
common /vlambda/lambdak
common /vsigma2/sigma2k
common /vsigma2_tau/sigma2_tauk
common /vLn_hati/Ln_hati
common /vt/t
common /vbeta0/beta0k
common /vbeta1/beta1k
common /vbeta2/beta2k
common /vsigma2_pde/sigma2_pdek
common /vPDEi/PDEi
h1=0.0d0
h2=0.0d0
h3=0.0d0
h4=0.0d0
h7=0.0d0
do k=1,Niti
```

```
            j=indi(k)
            h1=h1+1.0d0/sigma2k(j)
            h2=h2+1.0d0/(sigma2k(j)**2)
            h3=h3+(t(k)-lambdak(j))/(sigma2k(j)**2)
            h4=h4+(dsin(varphik)*dsqrt(sigma2_tauk)*(PDEi-beta0k)
     1           -beta1k*(t(k)-lambdak(j)))/sigma2k(j)
            h7=h7+(t(k)-lambdak(j))/sigma2k(j)
         enddo
         h5=0.0d0
         h6=0.0d0
         do it=1,(Niti-1)
            j=indi(it)
            do it1=(it+1),Niti
               k=indi(it1)
               h5=h5+1.0d0/sigma2k(j)/sigma2k(k)
               ep=t(it)-lambdak(j)+t(it1)-lambdak(k)
               h6=h6+ep/sigma2k(j)/sigma2k(k)
            enddo
          enddo
          h0=dsin(varphik)*dcos(varphik)
            deno=1.0d0+beta2k**2/sigma2_pdek
     1           +dcos(varphik)**2*sigma2_tauk*h1
            anume_AA=(h0*sigma2_tauk)**2*(h2+2*h5)
     1              +beta1k*beta2k/sigma2_pdek*(beta1k*beta2k
     1              +2*h0*sigma2_tauk*sigma2_pdek*h1)/sigma2_pdek
            AA=1.0d0+beta1k**2/sigma2_pdek
     1          +dsin(varphik)**2*sigma2_tauk*h1-anume_AA/deno
            bnume_BB=(beta1k*beta2k**2*(PDEi-beta0k)
```

```
     1                     +dcos(varphik)*dsqrt(sigma2_tauk)*beta2k
     1                     *sigma2_pdek*h4)/(sigma2_pdek**2)
     1                  -h0*dcos(varphik)*(dsqrt(sigma2_tauk))**3*(h3+h6)
                 BB=dsin(varphik)*dsqrt(sigma2_tauk)*h7
     1                -beta1k*(PDEi-beta0k)/sigma2_pdek+(bnume_BB)/deno
               Ln=-0.5d0*dlog(2*pi)+0.5d0*dlog(AA)-0.5d0*AA*(x+BB/AA)**2
               yi=0.0d0
               do k=1,Niti
                  j=indi(k)
                  ep=ak(j)*(x-bk(j))
                  if(ep.le.0.0d0)then
                          yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
                  else
                          yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
                  endif
               enddo
               Ln=Ln+yi
               fnac_rtpde=dexp(Ln-Ln_hati)
           end
c.............................................................
c.....function for [ACPDE]
      real*8 function fnacpde(x)
               implicit real*8 (a-h,o-z)
               parameter(nitem=20)
               parameter(pi=3.141592741012573)
               real*8 x,Ln,yi
               integer irespi(nitem)
               integer indi(nitem),Niti
```

```
      real*8  Ln_hati,PDEi
      real*8  ak(nitem),bk(nitem)
      real*8  beta0k,beta1k,beta2k,sigma2_pdek
      common /va/ak
      common /vb/bk
      common /vLn_hati/Ln_hati
      common /virespi/irespi
      common /vNiti/Niti
      common /vindi/indi
      common /vbeta0/beta0k
      common /vbeta1/beta1k
      common /vbeta2/beta2k
      common /vsigma2_pde/sigma2_pdek
      common /vPDEi/PDEi
      h=1.0d0+beta2k**2/sigma2_pdek
      Ln=-0.5d0*x**2-dlog(2*pi)-0.5*dlog(sigma2_pdek)
     1   -0.5d0/(sigma2_pdek)*(PDEi-beta0k-beta1k*x)**2
     1   -0.5d0*dlog(h)
     1   +0.5d0/h*(beta2k*(PDEi-beta0k-beta1k*x)/sigma2_pdek)**2
      yi=0.0d0
      do k=1,Niti
         j=indi(k)
         ep=ak(j)*(x-bk(j))
         if(ep.le.0.0d0)then
                 yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
         else
                 yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
         endif
```

```
              enddo
              Ln=Ln+yi
              fnacpde=dexp(Ln-Ln_hati)
          end
c.....................................................................
c.....function for [AC|RT]
      real*8 function fnac_rt(x)
              implicit real*8 (a-h,o-z)
              parameter(nitem=20)
              parameter(pi=3.141592741012573)
              real*8 x,Ln,yi
              integer irespi(nitem)
              integer indi(nitem),Niti
              real*8  Ln_hati,AA,BB
              real*8  t(nitem)
              real*8  ak(nitem),bk(nitem)
              real*8  sigma2k(nitem)
              real*8  lambdak(nitem)
              real*8  sigma2_tauk,varphik
              common /va/ak
              common /vb/bk
              common /virespi/irespi
              common /vNiti/Niti
              common /vindi/indi
              common /vvarphi/varphik
              common /vlambda/lambdak
              common /vsigma2/sigma2k
              common /vsigma2_tau/sigma2_tauk
```

```
common /vLn_hati/Ln_hati
common /vt/t
h1=0.0d0
h2=0.0d0
h3=0.0d0
h7=0.0d0
do k=1,Niti
     j=indi(k)
     h1=h1+1.0d0/sigma2k(j)
     h2=h2+1.0d0/(sigma2k(j)**2)
     h3=h3+(t(k)-lambdak(j))/(sigma2k(j)**2)
     h7=h7+(t(k)-lambdak(j))/sigma2k(j)
enddo
h5=0.0d0
h6=0.0d0
do it=1,(Niti-1)
   j=indi(it)
   do it1=(it+1),Niti
      k=indi(it1)
      h5=h5+1.0d0/sigma2k(j)/sigma2k(k)
      ep=t(it)-lambdak(j)+t(it1)-lambdak(k)
      h6=h6+ep/sigma2k(j)/sigma2k(k)
   enddo
 enddo
 h0=dsin(varphik)*dcos(varphik)
   deno=1.0d0+dcos(varphik)**2*sigma2_tauk*h1
   anume_AA=(h0*sigma2_tauk)**2*(h2+2*h5)
   AA=1.0d0+dsin(varphik)**2*sigma2_tauk*h1-anume_AA/deno
```

```
             bnume_BB=h0*dcos(varphik)*(dsqrt(sigma2_tauk))**3*(h3+h6)
               BB=dsin(varphik)*dsqrt(sigma2_tauk)*h7-(bnume_BB)/deno
            Ln=-0.5d0*dlog(2*pi)+0.5d0*dlog(AA)-0.5d0*AA*(x+BB/AA)**2
            yi=0.0d0
            do k=1,Niti
               j=indi(k)
               ep=ak(j)*(x-bk(j))
               if(ep.le.0.0d0)then
                       yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
               else
                       yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
               endif
            enddo
            Ln=Ln+yi
            fnac_rt=dexp(Ln-Ln_hati)
         end
c.............................................................
c.....function for [AC]
      real*8 function fnac(x)
            implicit real*8 (a-h,o-z)
            parameter(nitem=20)
            parameter(pi=3.141592741012573)
            real*8 x,Ln,yi
            integer irespi(nitem)
            integer indi(nitem),Niti
            real*8  Ln_hati
            real*8  ak(nitem),bk(nitem)
            common /va/ak
```

```
      common /vb/bk
      common /virespi/irespi
      common /vNiti/Niti
      common /vindi/indi
      common /vLn_hati/Ln_hati
      Ln=-0.5d0*dlog(2*pi)-0.5d0*x**2
      yi=0.0d0
      do k=1,Niti
         j=indi(k)
         ep=ak(j)*(x-bk(j))
         if(ep.le.0.0d0)then
                 yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
         else
                 yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
         endif
      enddo
      Ln=Ln+yi
      fnac=dexp(Ln-Ln_hati)
    end
```

(x) **floatingControl.f**

```
c.....Ln_hat_acrtpde for [ACRTPDE]
      subroutine Ln_hat_ACRTPDEjoint(Ln_hat,th,sigma2,
     1                               sigma2_tau,lambda,varphi,a,b,
     1                               beta0,beta1,beta2,sigma2_pde,
     1                               PDE,Nit,ind,iresp,rt,nstu,nitem)
      implicit real*8 (a-h,o-z)
      parameter(pi=3.141592741012573)
```

```
      real*8 rt(nstu,nitem),PDE(nstu)
      integer nstu,nitem
      real*8 Ln_hat(nstu),th(nstu)
      real*8 sigma2(nitem),sigma2_tau
      real*8 lambda(nitem),varphi
      real*8 a(nitem),b(nitem)
      integer Nit(nstu),iresp(nstu,nitem),ind(nstu,nitem)
      real*8 beta0,beta1,beta2,sigma2_pde
      do i=1,nstu
         thi=th(i)
         Ln_hat(i)=-0.5d0*(Nit(i)+2)*dlog(2*pi)-0.5d0*thi**2
     1             -0.5d0*dlog(sigma2_pde)-0.5d0/sigma2_pde
     1             *(PDE(i)-beta0-beta1*thi)**2
         h=1.0d0+beta2**2/sigma2_pde
         h0=0.0d0
         h1=-beta2/sigma2_pde*(PDE(i)-beta0-beta1*thi)
         yi=0.0d0
         h2=0.0d0
         do k=1,Nit(i)
            j=ind(i,k)
            t=rt(i,k)
            iu=iresp(i,k)
            h2=h2+dlog(sigma2(j))
            h=h+dcos(varphi)**2*sigma2_tau/sigma2(j)
            ep=t-lambda(j)+dsqrt(sigma2_tau)*dsin(varphi)*thi
            h0=h0+ep**2/sigma2(j)
            h1=h1+ep/sigma2(j)*dcos(varphi)*dsqrt(sigma2_tau)
            ep1=a(j)*(thi-b(j))
```

```
                    if(ep1.le.0.0d0)then
                           yi=yi+ep1*iu-dlog(1.0d0+dexp(ep1))
                    else
                           yi=yi+ep1*(iu-1)-dlog(1.0d0+dexp(-ep1))
                    endif
                 enddo
                 Ln_hat(i)=Ln_hat(i)+yi-0.5d0*dlog(h)
     1                     -0.5d0*h0-0.5d0*h2+0.5d0/h*h1**2
              enddo
           end
c.............................................................
c.....Ln_hat_acrt for [ACRT]
      subroutine Ln_hat_ACRT_PDEACRT(Ln_hat,th,sigma2,
     1                               sigma2_tau,lambda,varphi,
     1                              a,b,Nit,ind,iresp,rt,nstu,nitem)
              implicit real*8 (a-h,o-z)
              parameter(pi=3.141592741012573)
              real*8 rt(nstu,nitem)
              integer nstu,nitem
              real*8 Ln_hat(nstu),th(nstu)
              real*8 sigma2(nitem),sigma2_tau
              real*8 lambda(nitem),varphi
              real*8 a(nitem),b(nitem)
              integer Nit(nstu),iresp(nstu,nitem),ind(nstu,nitem)
              do i=1,nstu
                 thi=th(i)
                 Ln_hat(i)=-0.5d0*(Nit(i)+1)*dlog(2*pi)-0.5d0*thi**2
                 h0=0.0d0
```

```
                h=1.0d0
                h1=0.0d0
                yi=0.0d0
                h2=0.0d0
                do k=1,Nit(i)
                   j=ind(i,k)
                   t=rt(i,k)
                   iu=iresp(i,k)
                   h2=h2+dlog(sigma2(j))
                   h=h+dcos(varphi)**2*sigma2_tau/sigma2(j)
                   ep=t-lambda(j)+dsqrt(sigma2_tau)*dsin(varphi)*thi
                   h0=h0+ep**2/sigma2(j)
                   h1=h1+ep/sigma2(j)*dcos(varphi)*dsqrt(sigma2_tau)
                   ep=a(j)*(thi-b(j))
                   if(ep.le.0.0d0)then
                           yi=yi+ep*iu-dlog(1.0d0+dexp(ep))
                   else
                           yi=yi+ep*(iu-1)-dlog(1.0d0+dexp(-ep))
                   endif
                enddo
                Ln_hat(i)=Ln_hat(i)+yi-0.5d0*dlog(h)
     1               -0.5d0*h0-0.5d0*h2+0.5d0/h*h1**2
             enddo
          end
c.....................................................
c.....Ln_hat_ac_rtpde for [AC|RTPDE]
      subroutine Ln_hatAC_RTPDE(Ln_hat,th,sigma2,
     1                          sigma2_tau,lambda,varphi,a,b,
```

```
1                              beta0,beta1,beta2,sigma2_pde,
1                              PDE,Nit,ind,iresp,rt,nstu,nitem)
      implicit real*8 (a-h,o-z)
      parameter(pi=3.141592741012573)
      real*8 rt(nstu,nitem),PDE(nstu)
      integer nstu,nitem
      real*8 Ln_hat(nstu),th(nstu)
      real*8 sigma2(nitem),sigma2_tau
      real*8 lambda(nitem),varphi
      real*8 a(nitem),b(nitem)
      integer Nit(nstu),iresp(nstu,nitem),ind(nstu,nitem)
      real*8 beta0,beta1,beta2,sigma2_pde
      real*8 AA,BB
      Ln_hat=-0.5d0*dlog(2*pi)
      do i=1,nstu
         thi=th(i)
         yi=0.0d0
         do k=1,Nit(i)
            j=ind(i,k)
            ep1=a(j)*(thi-b(j))
            iu=iresp(i,k)
            if(ep1.le.0.0d0)then
                   yi=yi+ep1*iu-dlog(1.0d0+dexp(ep1))
            else
                   yi=yi+ep1*(iu-1)-dlog(1.0d0+dexp(-ep1))
            endif
         enddo
         h1=0.0d0
```

```
            h2=0.0d0
            h3=0.0d0
            h4=0.0d0
            h7=0.0d0
            do k=1,Nit(i)
               j=ind(i,k)
               h1=h1+1.0d0/sigma2(j)
               h2=h2+1.0d0/(sigma2(j)**2)
               h3=h3+(rt(i,k)-lambda(j))/(sigma2(j)**2)
               h4=h4+(dsin(varphi)*dsqrt(sigma2_tau)*(PDE(i)-beta0)
     1              -beta1*(rt(i,k)-lambda(j)))/sigma2(j)
               h7=h7+(rt(i,k)-lambda(j))/sigma2(j)
            enddo
            h5=0.0d0
            h6=0.0d0
            do it=1,(Nit(i)-1)
               j=ind(i,it)
               do it1=(it+1),Nit(i)
                  k=ind(i,it1)
                  h5=h5+1.0d0/sigma2(j)/sigma2(k)
                  ep=rt(i,it)-lambda(j)+rt(i,it1)-lambda(k)
                  h6=h6+ep/sigma2(j)/sigma2(k)
               enddo
            enddo
            h0=dsin(varphi)*dcos(varphi)
            deno=1.0d0+beta2**2/sigma2_pde
     1           +dcos(varphi)**2*sigma2_tau*h1
            anume_AA=(h0*sigma2_tau)**2*(h2+2*h5)
```

```
     1                    +beta1*beta2/sigma2_pde*(beta1*beta2
     1                    +2*h0*sigma2_tau*sigma2_pde*h1)/sigma2_pde
                 AA=1.0d0+beta1**2/sigma2_pde
     1               +dsin(varphi)**2*sigma2_tau*h1-anume_AA/deno
                 bnume_BB=(beta1*beta2**2*(PDE(i)-beta0)
     1                    +dcos(varphi)*dsqrt(sigma2_tau)*beta2
     1                    *sigma2_pde*h4)/(sigma2_pde**2)
     1                  -h0*dcos(varphi)*(dsqrt(sigma2_tau))**3*(h3+h6)
                 BB=dsin(varphi)*dsqrt(sigma2_tau)*h7
     1               -beta1*(PDE(i)-beta0)/sigma2_pde+(bnume_BB)/deno
                 Ln_hat(i)=Ln_hat(i)+yi+0.5d0*dlog(AA)
     1                    -0.5d0*AA*(thi+BB/AA)**2
               enddo
          end
c.....................................................................
c.....[ACPDE]
       subroutine Ln_hat_ACPDE_RTACPDE(Ln_hat,th,a,b,
     1                                 beta0,beta1,beta2,sigma2_pde,
     1                                 PDE,Nit,ind,iresp,nstu,nitem)
               implicit real*8 (a-h,o-z)
               parameter(pi=3.141592741012573)
               real*8 PDE(nstu)
               integer nstu,nitem
               real*8 Ln_hat(nstu),th(nstu)
               real*8 a(nitem),b(nitem)
               integer Nit(nstu),iresp(nstu,nitem),ind(nstu,nitem)
               real*8 beta0,beta1,beta2,sigma2_pde
               h=1.0d0+beta2**2/sigma2_pde
```

```
            do i=1,nstu
               thi=th(i)
               Ln_hat(i)=-dlog(2*pi)-0.5d0*thi**2
     1                   -0.5d0*dlog(sigma2_pde)-0.5d0/sigma2_pde
     1                   *(PDE(i)-beta0-beta1*thi)**2-0.5d0*dlog(h)
     1          +0.5d0/h*((PDE(i)-beta0-beta1*thi)*beta2/sigma2_pde)**2
               yi=0.0d0
               do k=1,Nit(i)
                  j=ind(i,k)
                  iu=iresp(i,k)
                  ep1=a(j)*(thi-b(j))
                  if(ep1.le.0.0d0)then
                         yi=yi+ep1*iu-dlog(1.0d0+dexp(ep1))
                  else
                         yi=yi+ep1*(iu-1)-dlog(1.0d0+dexp(-ep1))
                  endif
               enddo
               Ln_hat(i)=Ln_hat(i)+yi
            enddo
         end
c.............................................................
c.....Ln_hat_ac_rt for [AC|RT]
      subroutine Ln_hatAC_RT(Ln_hat,th,sigma2,sigma2_tau,lambda,varphi,
     1                              a,b,Nit,ind,iresp,rt,nstu,nitem)
            implicit real*8 (a-h,o-z)
            parameter(pi=3.141592741012573)
            real*8 rt(nstu,nitem)
            integer nstu,nitem
```

```
real*8 Ln_hat(nstu),th(nstu)
real*8 sigma2(nitem),sigma2_tau
real*8 lambda(nitem),varphi
real*8 a(nitem),b(nitem)
integer Nit(nstu),iresp(nstu,nitem),ind(nstu,nitem)
real*8 AA,BB
Ln_hat=-0.5d0*dlog(2*pi)
do i=1,nstu
   thi=th(i)
   yi=0.0d0
   do k=1,Nit(i)
      j=ind(i,k)
      ep1=a(j)*(thi-b(j))
      iu=iresp(i,k)
      if(ep1.le.0.0d0)then
             yi=yi+ep1*iu-dlog(1.0d0+dexp(ep1))
      else
             yi=yi+ep1*(iu-1)-dlog(1.0d0+dexp(-ep1))
      endif
   enddo
   h1=0.0d0
   h2=0.0d0
   h3=0.0d0
   h7=0.0d0
   do k=1,Nit(i)
      j=ind(i,k)
      h1=h1+1.0d0/sigma2(j)
      h2=h2+1.0d0/(sigma2(j)**2)
```

```
                  h3=h3+(rt(i,k)-lambda(j))/(sigma2(j)**2)
                  h7=h7+(rt(i,k)-lambda(j))/sigma2(j)
               enddo
               h5=0.0d0
               h6=0.0d0
               do it=1,(Nit(i)-1)
                  j=ind(i,it)
                  do it1=(it+1),Nit(i)
                     k=ind(i,it1)
                     h5=h5+1.0d0/sigma2(j)/sigma2(k)
                     ep=rt(i,it)-lambda(j)+rt(i,it1)-lambda(k)
                     h6=h6+ep/sigma2(j)/sigma2(k)
                  enddo
               enddo
               h0=dsin(varphi)*dcos(varphi)
               deno=1.0d0+dcos(varphi)**2*sigma2_tau*h1
               anume_AA=(h0*sigma2_tau)**2*(h2+2*h5)
               AA=1.0d0+dsin(varphi)**2*sigma2_tau*h1-anume_AA/deno
               bnume_BB=h0*dcos(varphi)*(dsqrt(sigma2_tau))**3*(h3+h6)
               BB=dsin(varphi)*dsqrt(sigma2_tau)*h7-(bnume_BB)/deno
               Ln_hat(i)=Ln_hat(i)+yi+0.5d0*dlog(AA)
     1                     -0.5d0*AA*(thi+BB/AA)**2
            enddo
         end
c.............................................................
c......[AC]
      subroutine Ln_hat_AC_PDEACRT(Ln_hat,th,a,b,
     1                             Nit,ind,iresp,nstu,nitem)
```

```
      implicit real*8 (a-h,o-z)
      parameter(pi=3.141592741012573)
      integer nstu,nitem
      real*8 Ln_hat(nstu),th(nstu)
      real*8 a(nitem),b(nitem)
      integer Nit(nstu),iresp(nstu,nitem),ind(nstu,nitem)
      do i=1,nstu
         thi=th(i)
         Ln_hat(i)=-0.5*dlog(2*pi)-0.5d0*thi**2
         yi=0.0d0
         do k=1,Nit(i)
            j=ind(i,k)
            iu=iresp(i,k)
            ep1=a(j)*(thi-b(j))
            if(ep1.le.0.0d0)then
                   yi=yi+ep1*iu-dlog(1.0d0+dexp(ep1))
            else
                   yi=yi+ep1*(iu-1)-dlog(1.0d0+dexp(-ep1))
            endif
         enddo
         Ln_hat(i)=Ln_hat(i)+yi
      enddo
   end
```

(xi) **DICPDcalculate.f**

```
subroutine DICPDcalculate(DIC,PD,Dev,niter)
 implicit real*8 (a-h,o-z)
 integer niter
```

```
      real*8  DIC,PD,Dev(1+niter)
      real*8  Dev_hat,Dev_bar
      Dev_hat=Dev(1+niter)
      Dev_bar=sum(Dev(1:niter))/niter
      PD=Dev_bar-Dev_hat
      DIC=Dev_hat+2*PD
     end
```

Below is the main program for DIC and LPML decomposition.

(xii) **dic.f**

```
      program DIC_Decompositions
c.....................................................................
c          DIC decomposition: [ACRTPDE]=[AC|RTPDE]+[RTPDE]
c                                      =[PDE|ACRT]+[ACRT]
c                                      =[RT|ACPDE]+[ACPDE]
c.....................................................................
c
c      nstu:         no. of subjects
c      nitem:        no. of items
c      npara:        no. of parameters from MCMC samples of joint model
c      niter:        length of MCMC
c      iresp:        nstu-by-nitem matrix, which save the response data
c      ind:          nstu-by-nitem matrix, which save the item index
c      Nit:          vector with length nstu, which save the total no.
c                                  of response items for each subjects.
c      samples:      (1+niter)-by-npara matrix, which save the MCMC
c                    samples from joint model.
c      rt:           nstu-by-nitem matrix,
```

```
c                   which save the response time data
c      PDE:         vector with length nstu, which save the PDE data.
       implicit real*8 (a-h,o-z)
       parameter(nstu=125,nitem=20)
       parameter(npara=4*nitem+2*nstu+8,niter=10000)
       parameter(nrep0=270,nrep=270)
       parameter(pi=3.141592741012573)
       character(3)sequ
       character(26)respfile
       character(20)rtfile
       character(21)PDEfile
       character(36)mcmcfile
       character(36)resultsfile
       character(3)sequi
       character(2)sequj
       integer iresp(nstu,nitem)
       integer ind(nstu,nitem)
       integer Nit(nstu),istu
       real*8  samples(1+niter,npara)
       real*8  rt(nstu,nitem),PDE(nstu)
c......indicate the begainning time
       integer now(3),ntoday(3)
c......mcmc variables
       real*8  a(1+niter,nitem),a_hat(nitem)
       real*8  b(1+niter,nitem),b_hat(nitem)
       real*8  theta(1+niter,nstu),theta_hat(nstu)
       real*8  sigma2(1+niter,nitem),sigma2_hat(nitem)
       real*8  lambda(1+niter,nitem),lambda_hat(nitem)
```

```
      real*8  tau(1+niter,nstu),w(1+niter)
      real*8  varphi(1+niter),varphi_hat
      real*8  sigma2_tau(1+niter),sigma2_tau_hat
      real*8  beta0(1+niter),beta0_hat
      real*8  beta1(1+niter),beta1_hat
      real*8  beta2(1+niter),beta2_hat
      real*8  sigma2_pde(1+niter),sigma2_pde_hat
c.....Ln_hat
      real*8  Ln_hat_acrtpde(nstu)
      real*8  Ln_hat_acrt(nstu)
      real*8  Ln_hat_acpde(nstu)
      real*8  Ln_hat_ac(nstu)
c.....dic decomposition lpml
      real*8  uacrt(1+niter,nstu),Devacrt(1+niter)
      real*8  DICacrt,PDacrt
      real*8  uac(1+niter,nstu),Devac(1+niter)
      real*8  urt_ac(1+niter,nstu),Devrt_ac(1+niter)
      real*8  DICac,PDac,DICrt_ac,PDrt_ac
      real*8  upde_acrt(1+niter,nstu),Devpde_acrt(1+niter)
      real*8  DICpde_acrt,PDpde_acrt
      real*8  uacrtpde(1+niter,nstu),Devacrtpde(1+niter)
      real*8  DICacrtpde,PDacrtpde,LPMLacrtpde
      real*8  uacpde(1+niter,nstu),Devacpde(1+niter)
      real*8  DICacpde,PDacpde,DICrt_acpde,PDrt_acpde
      real*8  urt_acpde(1+niter,nstu),Devrt_acpde(1+niter)
c.....integral variable
      integer interv
      real*8  result
```

```
      real*8  bound,errabs,errrel,delta,errest
c.....common
      real*8  ak(nitem),bk(nitem),sigma2k(nitem),lambdak(nitem)
      real*8  sigma2_tauk,varphik,thetak(nstu)
      real*8  beta0k,beta1k,beta2k,sigma2_pdek
      integer irespi(nitem),Niti,indi(nitem)
      real*8  t(nitem),Ln_hati
      external fnacrt,fnacrtpde,fnac_rtpde,fnacpde
      external fnac_rt,fnac
      common /va/ak
      common /vb/bk
      common /vvarphi/varphik
      common /vlambda/lambdak
      common /vsigma2/sigma2k
      common /vsigma2_tau/sigma2_tauk
      common /vLn_hati/Ln_hati
      common /virespi/irespi
      common /vNiti/Niti
      common /vindi/indi
      common /vt/t
      common /vbeta0/beta0k
      common /vbeta1/beta1k
      common /vbeta2/beta2k
      common /vsigma2_pde/sigma2_pdek
      common /vPDEi/PDEi
      common /vistu/istu
c.....show beginning time.
      call tdate(iday,month,iyear)
```

```
        call timdy(ihour,imin,isec)
        write(*,*)'starting time is'
        write(*,*)iyear,'-',month,'-',iday,' ',
     1           ihour,':',imin,':',isec
c......read data
        write(sequi,'(i3.3)')nstu
        write(sequj,'(i2.2)')nitem
        do nsim=nrep0,nrep
        write(*,*)'nsim=',nsim
        write(sequ,'(i3.3)')nsim
        PDEfile='For_PDE'//sequi//'_'//sequj//'_'//sequ//'.csv'
        rtfile='For_rt'//sequi//'_'//sequj//'_'//sequ//'.csv'
        respfile='For_stu_resp'//sequi//'_'//sequj//'_'//sequ//'.csv'
        mcmcfile='For_samples_resprtpde_'//sequi//'_'//sequj//'_'
     1                                                 //sequ//'.csv'
        do i=1,nstu
          Nit(i)=nitem
          do j=1,nitem
             ind(i,j)=j
          enddo
        enddo
        open(unit=21,file=respfile,status='old')
        do i=1,nstu
          read(21,*)iresp(i,1:Nit(i))
        enddo
        close(21)
        open(unit=231,file=rtfile,status='old')
        do i=1,nstu
```

```
        read(231,*)rt(i,1:Nit(i))
      enddo
      close(231)
      open(unit=231,file=PDEfile,status='old')
      do i=1,nstu
        read(231,*)PDE(i)
      enddo
      close(231)
      open(unit=22,file=mcmcfile,status='old')
      do i=1,niter
        read(22,*)samples(i,1:npara)
      enddo
      close(22)
c.....separate mcmc
      a(1:niter,1:nitem)=samples(1:niter,1:nitem)
      b(1:niter,1:nitem)=samples(1:niter,(1+nitem):(2*nitem))
      beta0(1:niter)=samples(1:niter,1+2*nitem)
      beta1(1:niter)=samples(1:niter,2+2*nitem)
      beta2(1:niter)=samples(1:niter,3+2*nitem)
      lambda(1:niter,1:nitem)=samples(1:niter,(4+2*nitem):(3+3*nitem))
      sigma2(1:niter,1:nitem)=samples(1:niter,
     1                                (1+4+3*nitem):(4+4*nitem))
      sigma2_pde(1:niter)=samples(1:niter,4*nitem+6)
      tau(1:niter,1:nstu)=samples(1:niter,
     1                             (4*nitem+6+1):(4*nitem+6+nstu))
      theta(1:niter,1:nstu)=samples(1:niter,
     1                        (4*nitem+6+nstu+1):(4*nitem+6+2*nstu))
      varphi(1:niter)=samples(1:niter,4*nitem+6+2*nstu+1)
```

```
      w(1:niter)=samples(1:niter,4*nitem+6+2*nstu+2)
      do i=1,niter
         sigma2_tau(i)=dexp(2*w(i))
      enddo
c.....eap
      do j=1,nitem
         a(1+niter,j)=sum(a(1:niter,j))/niter
         b(1+niter,j)=sum(b(1:niter,j))/niter
         lambda(1+niter,j)=sum(lambda(1:niter,j))/niter
         sigma2(1+niter,j)=sum(sigma2(1:niter,j))/niter
      enddo
      sigma2_pde(1+niter)=sum(sigma2_pde(1:niter))/niter
      sigma2_tau(1+niter)=sum(sigma2_tau(1:niter))/niter
      varphi(1+niter)=sum(varphi(1:niter))/niter
      beta0(1+niter)=sum(beta0(1:niter))/niter
      beta1(1+niter)=sum(beta1(1:niter))/niter
      beta2(1+niter)=sum(beta2(1:niter))/niter
      do i=1,nstu
         theta(1+niter,i)=sum(theta(1:niter,i))/niter
         tau(1+niter,i)=sum(tau(1:niter,i))/niter
      enddo
c.....hat
      a_hat=a(1+niter,1:nitem)
      b_hat=b(1+niter,1:nitem)
      beta0_hat=beta0(1+niter)
      beta1_hat=beta1(1+niter)
      beta2_hat=beta2(1+niter)
      lambda_hat=lambda(1+niter,1:nitem)
```

```
        sigma2_hat=sigma2(1+niter,1:nitem)
        sigma2_pde_hat=sigma2_pde(1+niter)
        theta_hat=theta(1+niter,1:nstu)
        varphi_hat=varphi(1+niter)
        sigma2_tau_hat=sigma2_tau(1+niter)
c......start decomposition.........................................
c......Ln_hat
        ak=a_hat
        bk=b_hat
        beta0k=beta0_hat
        beta1k=beta1_hat
        beta2k=beta2_hat
        lambdak=lambda_hat
        sigma2k=sigma2_hat
        sigma2_pdek=sigma2_pde_hat
        thetak=theta_hat
        varphik=varphi_hat
        sigma2_tauk=sigma2_tau_hat
        ![ACRTPDE]
        call Ln_hat_ACRTPDEjoint(Ln_hat_acrtpde,thetak,sigma2k,
     1                                sigma2_tauk,lambdak,varphik,ak,bk,
     1                                beta0k,beta1k,beta2k,sigma2_pdek,
     1                                PDE,Nit,ind,iresp,rt,nstu,nitem)
        ![ACRT]
        call Ln_hat_ACRT_PDEACRT(Ln_hat_acrt,thetak,sigma2k,
     1                                sigma2_tauk,lambdak,varphik,
     1                               ak,bk,Nit,ind,iresp,rt,nstu,nitem)
        ![ACPDE]
```

```
        call Ln_hat_ACPDE_RTACPDE(Ln_hat_acpde,thetak,ak,bk,
     1                                beta0k,beta1k,beta2k,sigma2_pdek,
     1                                PDE,Nit,ind,iresp,nstu,nitem)
        ![AC]
        call  Ln_hat_AC_PDEACRT(Ln_hat_ac,thetak,ak,bk,
     1                               Nit,ind,iresp,nstu,nitem)
c......integral
        interv=2
        bound=0.0d0
        errabs=0.0d0
        errrel=0.00001d0
        do it=1,(1+niter)
           if(mod(it,2000).eq.0)then
                  write(*,*)it
           endif
           ak=a(it,1:nitem)
           bk=b(it,1:nitem)
           lambdak=lambda(it,1:nitem)
           sigma2k=sigma2(it,1:nitem)
           sigma2_tauk=sigma2_tau(it)
           varphik=varphi(it)
           beta0k=beta0(it)
           beta1k=beta1(it)
           beta2k=beta2(it)
           sigma2_pdek=sigma2_pde(it)
           do istu=1,nstu
              t=rt(istu,1:nitem)
              irespi=iresp(istu,1:nitem)
```

```
        indi=ind(istu,1:nitem)
        Niti=Nit(istu)
        PDEi=PDE(istu)
        ![ACRTPDE]
        Ln_hati=Ln_hat_acrtpde(istu)
        call DQDAGI(fnacrtpde,bound,interv,errabs,errrel,
     1                                          result,errest)
        uacrtpde(it,istu)=dlog(result)+Ln_hat_acrtpde(istu)
       ![ACRT]+[PDE|ACRT]
        Ln_hati=Ln_hat_acrt(istu)
        call DQDAGI(fnacrt,bound,interv,errabs,errrel,
     1                                          result,errest)
        uacrt(it,istu)=dlog(result)+Ln_hat_acrt(istu)
        upde_acrt(it,istu)=uacrtpde(it,istu)-uacrt(it,istu)
        ![AC] & [RT|AC]
        Ln_hati=Ln_hat_ac(istu)
        call DQDAGI(fnac,bound,interv,errabs,errrel,
     1                                          result,errest)
        uac(it,istu)=dlog(result)+Ln_hat_ac(istu)
        urt_ac(it,istu)=uacrt(it,istu)-uac(it,istu)
       ![ACPDE]+[RT|ACPDE]
        Ln_hati=Ln_hat_acpde(istu)
        call DQDAGI(fnacpde,bound,interv,errabs,errrel,
     1                                          result,errest)
        uacpde(it,istu)=dlog(result)+Ln_hat_acpde(istu)
        urt_acpde(it,istu)=uacrtpde(it,istu)-uacpde(it,istu)
     enddo
     Devacrt(it)=-2*sum(uacrt(it,1:nstu))
```

```
   Devacrtpde(it)=-2*sum(uacrtpde(it,1:nstu))
   Devpde_acrt(it)=-2*sum(upde_acrt(it,1:nstu))
   Devacpde(it)=-2*sum(uacpde(it,1:nstu))
   Devrt_acpde(it)=-2*sum(urt_acpde(it,1:nstu))
   Devrt_ac(it)=-2*sum(urt_ac(it,1:nstu))
   Devac(it)=-2*sum(uac(it,1:nstu))
enddo
call DICPDcalculate(DICacrt,PDacrt,Devacrt,niter)
call DICPDcalculate(DICacrtpde,PDacrtpde,Devacrtpde,niter)
call DICPDcalculate(DICpde_acrt,PDpde_acrt,Devpde_acrt,niter)
call DICPDcalculate(DICacpde,PDacpde,Devacpde,niter)
call DICPDcalculate(DICrt_acpde,PDrt_acpde,Devrt_acpde,niter)
call DICPDcalculate(DICac,PDac,Devac,niter)
call DICPDcalculate(DICrt_ac,PDrt_ac,Devrt_ac,niter)
write(*,*)'dicacrt=',DICacrt
write(*,*)'pdacrt=',PDacrt
write(*,*)'dicacrtpde=',DICacrtpde
write(*,*)'pdacrtpde=',PDacrtpde
write(*,*)'dicpde_acrt=',DICpde_acrt
write(*,*)'pdpde_acrt=',PDpde_acrt
write(*,*)'DICacpde=',DICacpde
write(*,*)'pdacpde=',PDacpde
write(*,*)'DICrt_acpde=',DICrt_acpde
write(*,*)'pdrt_acpde=',PDrt_acpde
write(*,*)'dicac=',DICac
write(*,*)'pdac=',PDac
write(*,*)'dicrt_ac=',DICrt_ac
write(*,*)'pdrt_ac=',PDrt_ac
```

```
        LPMLacrtpde=0.0d0
        do i=1,nstu
           umax=maxval(-uacrtpde(1:niter,i))
           yi=0.0d0
           do k=1,niter
              yi=yi+dexp(-uacrtpde(k,i)-umax)
           enddo
           yi=yi/niter
           LPMLacrtpde=LPMLacrtpde-umax-dlog(yi)
         enddo
        write(*,*)'LPMLacrtpde=',LPMLacrtpde
c......save to file
      resultsfile='DICdecomResults_simple'//sequi//'_'//sequj//'_'
     1                                              //sequ//'.dat'
      open(unit=16,file=resultsfile,access='sequential',
     1             status='unknown')
      write(16,983)DICacrt
      write(16,984)PDacrt
      write(16,985)DICpde_acrt
      write(16,986)PDpde_acrt
      write(16,987)DICacrtpde
      write(16,988)PDacrtpde
      write(16,989)LPMLacrtpde
      write(16,880)DICac
      write(16,881)PDac
      write(16,886)DICrt_ac
      write(16,887)PDrt_ac
      write(16,888)DICacpde
```

```
      write(16,889)PDacpde
      write(16,890)DICrt_acpde
      write(16,891)PDrt_acpde
      close(16)
 983     format('DICacrt=',f12.6)
 984     format('PDacrt=',f12.6)
 985     format('DICpde_acrt=',f12.6)
 986     format('PDpde_acrt=',f12.6)
 987     format('DICacrtpde=',f12.6)
 988     format('PDacrtpde=',f12.6)
 989     format('LPMLacrtpde=',f12.6)
 880     format('DICac=',f12.6)
 881     format('PDac=',f12.6)
 886     format('DICrt_ac=',f12.6)
 887     format('PDrt_ac=',f12.6)
 888     format('DICacpde=',f12.6)
 889     format('PDacpde=',f12.6)
 890     format('DICrt_acpde=',f12.6)
 891     format('PDrt_acpde=',f12.6)
c......ending time.
       call tdate(iday,month,iyear)
       call timdy(ihour,imin,isec)
       write(*,*)'ending time is'
       write(*,*)iyear,'-',month,'-',iday,' ',
     1           ihour,':',imin,':',isec
       enddo
      end
      include 'DICPDcalculate.f'
```

```
      include 'floatingControl.f'
      include 'function.f'
```

(xiii) **lpml.f**

```
      program LPML_Decompositions
c.....................................................................
c          LPML decomposition: [ACRTPDE]=[AC|RTPDE]+[RTPDE]
c                                       =[PDE|ACRT]+[ACRT]
c                                       =[RT|ACPDE]+[ACPDE]
c.....................................................................
       implicit real*8 (a-h,o-z)
       parameter(nstu=125,nitem=20)
       parameter(npara=4*nitem+2*nstu+8,niter=10000)
       parameter(nac_rtpde=2*nitem+nstu+2)
       parameter(npde_acrt=4+nstu)
       parameter(nrt_acpde=2*nitem+nstu+2)
       parameter(nrep0=270,nrep=270)
       parameter(pi=3.141592741012573)
       character(3)sequ
       character(26)respfile
       character(20)rtfile
       character(21)PDEfile
       character(36)mcmcfile
       character(41)mcmcac_rtpdefile
       character(41)mcmcpde_acrtfile
       character(41)mcmcrt_acpdefile
       character(30)resultsfile
       character(3)sequi
```

```
      character(2)sequj
      integer iresp(nstu,nitem)
      integer ind(nstu,nitem)
      integer Nit(nstu),istu
      real*8  samples(1+niter,npara)
      real*8  rt(nstu,nitem),PDE(nstu)
      real*8  sac_rtpde(1+niter,nac_rtpde)
      real*8  spde_acrt(1+niter,npde_acrt)
      real*8  srt_acpde(1+niter,nrt_acpde)
c.....indicate the begainning time
      integer now(3),ntoday(3)
c.....mcmc variables
      real*8  a(1+niter,nitem),a_hat(nitem)
      real*8  b(1+niter,nitem),b_hat(nitem)
      real*8  theta(1+niter,nstu),theta_hat(nstu)
      real*8  sigma2(1+niter,nitem),sigma2_hat(nitem)
      real*8  lambda(1+niter,nitem),lambda_hat(nitem)
      real*8  tau(1+niter,nstu),w(1+niter)
      real*8  varphi(1+niter),varphi_hat
      real*8  sigma2_tau(1+niter),sigma2_tau_hat
      real*8  beta0(1+niter),beta0_hat
      real*8  beta1(1+niter),beta1_hat
      real*8  beta2(1+niter),beta2_hat
      real*8  sigma2_pde(1+niter),sigma2_pde_hat
c.....Ln_hat
      real*8  Ln_hat_acrtpde(nstu)
      real*8  Ln_hat_acrt(nstu)
      real*8  Ln_hat_ac_rtpde(nstu)
```

```
      real*8  Ln_hat_acpde(nstu)
c.....lpml
      real*8  uacrt(1+niter,nstu)
      real*8  upde_acrt(1+niter,nstu)
      real*8  LPMLpde_acrt
      real*8  uacrtpde(1+niter,nstu)
      real*8  uac_rtpde(1+niter,nstu)
      real*8  LPMLac_rtpde
      real*8  uacpde(1+niter,nstu)
      real*8  LPMLrt_acpde
      real*8  urt_acpde(1+niter,nstu)
c.....integral variable
      integer interv
      real*8  result
      real*8  bound,errabs,errrel,delta,errest
c.....common
      real*8  ak(nitem),bk(nitem),sigma2k(nitem),lambdak(nitem)
      real*8  sigma2_tauk,varphik,thetak(nstu)
      real*8  beta0k,beta1k,beta2k,sigma2_pdek
      integer irespi(nitem),Niti,indi(nitem)
      real*8  t(nitem),Ln_hati
      external fnacrt,fnacrtpde,fnac_rtpde,fnacpde
      common /va/ak
      common /vb/bk
      common /vvarphi/varphik
      common /vlambda/lambdak
      common /vsigma2/sigma2k
      common /vsigma2_tau/sigma2_tauk
```

```
      common /vLn_hati/Ln_hati
      common /virespi/irespi
      common /vNiti/Niti
      common /vindi/indi
      common /vt/t
      common /vbeta0/beta0k
      common /vbeta1/beta1k
      common /vbeta2/beta2k
      common /vsigma2_pde/sigma2_pdek
      common /vPDEi/PDEi
      common /vistu/istu
c.....show beginning time.
      call tdate(iday,month,iyear)
      call timdy(ihour,imin,isec)
      write(*,*)'starting time is'
      write(*,*)iyear,'-',month,'-',iday,' ',
     1          ihour,':',imin,':',isec
c.....read data
      write(sequi,'(i3.3)')nstu
      write(sequj,'(i2.2)')nitem
      do nsim=nrep0,nrep
      write(*,*)'nsim=',nsim
      write(sequ,'(i3.3)')nsim
      PDEfile='For_PDE'//sequi//'_'//sequj//'_'//sequ//'.csv'
      rtfile='For_rt'//sequi//'_'//sequj//'_'//sequ//'.csv'
      respfile='For_stu_resp'//sequi//'_'//sequj//'_'//sequ//'.csv'
      mcmcfile='For_samples_resprtpde_'//sequi//'_'//sequj//'_'
     1                                          //sequ//'.csv'
```

```
      do i=1,nstu
        Nit(i)=nitem
        do j=1,nitem
           ind(i,j)=j
        enddo
      enddo
      open(unit=21,file=respfile,status='old')
      do i=1,nstu
        read(21,*)iresp(i,1:Nit(i))
      enddo
      close(21)
      open(unit=231,file=rtfile,status='old')
      do i=1,nstu
        read(231,*)rt(i,1:Nit(i))
      enddo
      close(231)
      open(unit=231,file=PDEfile,status='old')
      do i=1,nstu
        read(231,*)PDE(i)
      enddo
      close(231)
      open(unit=22,file=mcmcfile,status='old')
      do i=1,niter
        read(22,*)samples(i,1:npara)
      enddo
      close(22)
c.....separate mcmc
      a(1:niter,1:nitem)=samples(1:niter,1:nitem)
```

```
      b(1:niter,1:nitem)=samples(1:niter,(1+nitem):(2*nitem))
      beta0(1:niter)=samples(1:niter,1+2*nitem)
      beta1(1:niter)=samples(1:niter,2+2*nitem)
      beta2(1:niter)=samples(1:niter,3+2*nitem)
      lambda(1:niter,1:nitem)=samples(1:niter,(4+2*nitem):(3+3*nitem))
      sigma2(1:niter,1:nitem)=samples(1:niter,
     1                            (1+4+3*nitem):(4+4*nitem))
      sigma2_pde(1:niter)=samples(1:niter,4*nitem+6)
      tau(1:niter,1:nstu)=samples(1:niter,
     1                          (4*nitem+6+1):(4*nitem+6+nstu))
      theta(1:niter,1:nstu)=samples(1:niter,
     1                       (4*nitem+6+nstu+1):(4*nitem+6+2*nstu))
      varphi(1:niter)=samples(1:niter,4*nitem+6+2*nstu+1)
      w(1:niter)=samples(1:niter,4*nitem+6+2*nstu+2)
      do i=1,niter
         sigma2_tau(i)=dexp(2*w(i))
      enddo
c.....eap
      do j=1,nitem
         a(1+niter,j)=sum(a(1:niter,j))/niter
         b(1+niter,j)=sum(b(1:niter,j))/niter
         lambda(1+niter,j)=sum(lambda(1:niter,j))/niter
         sigma2(1+niter,j)=sum(sigma2(1:niter,j))/niter
      enddo
      sigma2_pde(1+niter)=sum(sigma2_pde(1:niter))/niter
      sigma2_tau(1+niter)=sum(sigma2_tau(1:niter))/niter
      varphi(1+niter)=sum(varphi(1:niter))/niter
      beta0(1+niter)=sum(beta0(1:niter))/niter
```

```
        beta1(1+niter)=sum(beta1(1:niter))/niter
        beta2(1+niter)=sum(beta2(1:niter))/niter
        do i=1,nstu
           theta(1+niter,i)=sum(theta(1:niter,i))/niter
           tau(1+niter,i)=sum(tau(1:niter,i))/niter
        enddo
c......hat
        a_hat=a(1+niter,1:nitem)
        b_hat=b(1+niter,1:nitem)
        beta0_hat=beta0(1+niter)
        beta1_hat=beta1(1+niter)
        beta2_hat=beta2(1+niter)
        lambda_hat=lambda(1+niter,1:nitem)
        sigma2_hat=sigma2(1+niter,1:nitem)
        sigma2_pde_hat=sigma2_pde(1+niter)
        theta_hat=theta(1+niter,1:nstu)
        varphi_hat=varphi(1+niter)
        sigma2_tau_hat=sigma2_tau(1+niter)
c......integral
        interv=2
        bound=0.0d0
        errabs=0.0d0
        errrel=0.00001d0
c.............................................................
c......[AC|RTPDE]
        mcmcac_rtpdefile='For_samplescond_resp_rtpde_'//sequi//'_'//
     1                   sequj//'_'//sequ//'.csv'
        write(*,*)mcmcac_rtpdefile
```

```
  open(unit=2,file=mcmcac_rtpdefile,status='old')
  do i=1,niter
    read(2,*)sac_rtpde(i,1:nac_rtpde)
  enddo
  close(2)
  a(1:niter,1:nitem)=sac_rtpde(1:niter,1:nitem)
  b(1:niter,1:nitem)=sac_rtpde(1:niter,(1+nitem):(2*nitem))
  theta(1:niter,1:nstu)=sac_rtpde(1:niter,
1                               (1+2+2*nitem):(2*nitem+2+nstu))
  do j=1,nitem
     a(1+niter,j)=sum(a(1:niter,j))/niter
     b(1+niter,j)=sum(b(1:niter,j))/niter
  enddo
  do i=1,nstu
     theta(1+niter,i)=sum(theta(1:niter,i))/niter
  enddo
  ak=a(1+niter,1:nitem)
  bk=b(1+niter,1:nitem)
  beta0k=beta0_hat
  beta1k=beta1_hat
  beta2k=beta2_hat
  lambdak=lambda_hat
  sigma2k=sigma2_hat
  sigma2_pdek=sigma2_pde_hat
  thetak=theta(1+niter,1:nstu)
  varphik=varphi_hat
  sigma2_tauk=sigma2_tau_hat
  ![AC|RTPDE]
```

```
  call Ln_hatAC_RTPDE(Ln_hat_ac_rtpde,thetak,sigma2k,
1                                sigma2_tauk,lambdak,varphik,ak,bk,
1                                beta0k,beta1k,beta2k,sigma2_pdek,
1                                PDE,Nit,ind,iresp,rt,nstu,nitem)
  uac_rtpde=0.0d0
  do it=1,(1+niter)
     if(mod(it,2000).eq.0)then
            write(*,*)it
     endif
     ak=a(it,1:nitem)
     bk=b(it,1:nitem)
    do istu=1,nstu
        t=rt(istu,1:nitem)
        irespi=iresp(istu,1:nitem)
        indi=ind(istu,1:nitem)
        Niti=Nit(istu)
        PDEi=PDE(istu)
        Ln_hati=Ln_hat_ac_rtpde(istu)
       call DQDAGI(fnac_rtpde,bound,interv,errabs,errrel,
1                                            result,errest)
       uac_rtpde(it,istu)=dlog(result)+Ln_hat_ac_rtpde(istu)
     enddo
  enddo
  LPMLac_rtpde=0.0d0
  do i=1,nstu
     umax=maxval(-uac_rtpde(1:niter,i))
     !write(*,*)'umaix(',i,')=',umax
     yi=0.0d0
```

```
          do k=1,niter
             yi=yi+dexp(-uac_rtpde(k,i)-umax)
          enddo
          yi=yi/niter
          LPMLac_rtpde=LPMLac_rtpde-umax-dlog(yi)
        enddo
       write(*,992)LPMLac_rtpde
 992     format('LPMLac_rtpde=',f12.6)

c.....................................................................
c.......[PDE|ACRT]
       mcmcpde_acrtfile='For_samplescond_pde_resprt_'//sequi//'_'//
     1                  sequj//'_'//sequ//'.csv'
       open(unit=2,file=mcmcpde_acrtfile,status='old')
       do i=1,niter
         read(2,*)spde_acrt(i,1:npde_acrt)
       enddo
       close(2)
       beta0(1:niter)=spde_acrt(1:niter,1)
       beta1(1:niter)=spde_acrt(1:niter,2)
       beta2(1:niter)=spde_acrt(1:niter,3)
       sigma2_pde(1:niter)=spde_acrt(1:niter,4)
       theta(1:niter,1:nstu)=spde_acrt(1:niter,5:(4+nstu))
       sigma2_pde(1+niter)=sum(sigma2_pde(1:niter))/niter
       beta0(1+niter)=sum(beta0(1:niter))/niter
       beta1(1+niter)=sum(beta1(1:niter))/niter
       beta2(1+niter)=sum(beta2(1:niter))/niter
       do i=1,nstu
```

```
      theta(1+niter,i)=sum(theta(1:niter,i))/niter
    enddo
    ak=a_hat
    bk=b_hat
    beta0k=beta0(1+niter)
    beta1k=beta1(1+niter)
    beta2k=beta2(1+niter)
    lambdak=lambda_hat
    sigma2k=sigma2_hat
    sigma2_pdek=sigma2_pde(1+niter)
    thetak=theta(1+niter,1:nstu)
    varphik=varphi_hat
    sigma2_tauk=sigma2_tau_hat
    ![ACRT]
    call Ln_hat_ACRT_PDEACRT(Ln_hat_acrt,thetak,sigma2k,
   1                                sigma2_tauk,lambdak,varphik,
   1                               ak,bk,Nit,ind,iresp,rt,nstu,nitem)
    ![ACRTPDE]
    call Ln_hat_ACRTPDEjoint(Ln_hat_acrtpde,thetak,sigma2k,
   1                                sigma2_tauk,lambdak,varphik,ak,bk,
   1                                beta0k,beta1k,beta2k,sigma2_pdek,
   1                                PDE,Nit,ind,iresp,rt,nstu,nitem)
    upde_acrt=0.0d0
    do it=1,(1+niter)
       if(mod(it,2000).eq.0)then
              write(*,*)it
       endif
       beta0k=beta0(it)
```

```
      beta1k=beta1(it)
      beta2k=beta2(it)
      sigma2_pdek=sigma2_pde(it)
      do istu=1,nstu
         t=rt(istu,1:nitem)
         irespi=iresp(istu,1:nitem)
         indi=ind(istu,1:nitem)
         Niti=Nit(istu)
         PDEi=PDE(istu)
         ![ACRTPDE]
         Ln_hati=Ln_hat_acrtpde(istu)
         call DQDAGI(fnacrtpde,bound,interv,errabs,errrel,
     1                                              result,errest)
         uacrtpde(it,istu)=dlog(result)+Ln_hat_acrtpde(istu)
        ![ACRT]
         Ln_hati=Ln_hat_acrt(istu)
        call DQDAGI(fnacrt,bound,interv,errabs,errrel,result,errest)
         uacrt(it,istu)=dlog(result)+Ln_hat_acrt(istu)
         upde_acrt(it,istu)=uacrtpde(it,istu)-uacrt(it,istu)
      enddo
   enddo
   LPMLpde_acrt=0.0d0
   do i=1,nstu
      umax=maxval(-upde_acrt(1:niter,i))
      !write(*,*)'umaix(',i,')=',umax
      yi=0.0d0
      do k=1,niter
         yi=yi+dexp(-upde_acrt(k,i)-umax)
```

```
           enddo
           yi=yi/niter
           LPMLpde_acrt=LPMLpde_acrt-umax-dlog(yi)
         enddo
        write(*,993)LPMLpde_acrt
 993     format('LPMLpde_acrt=',f12.6)

c.....................................................................
c.......[RT|ACPDE]
        mcmcrt_acpdefile='For_samplescond_rt_resppde_'//sequi//'_'//
     1                   sequj//'_'//sequ//'.csv'

        open(unit=2,file=mcmcrt_acpdefile,status='old')
        do i=1,niter
          read(2,*)srt_acpde(i,1:nrt_acpde)
        enddo
        close(2)
        lambda(1:niter,1:nitem)=srt_acpde(1:niter,1:nitem)
        sigma2(1:niter,1:nitem)=srt_acpde(1:niter,(1+nitem):(2*nitem))
        theta(1:niter,1:nstu)=srt_acpde(1:niter,
     1                                    (1+2*nitem):(2*nitem+nstu))
        varphi(1:niter)=srt_acpde(1:niter,2*nitem+nstu+1)
        w(1:niter)=srt_acpde(1:niter,2*nitem+nstu+2)
        do i=1,niter
           sigma2_tau(i)=dexp(2*w(i))
        enddo
c......eap
        do j=1,nitem
```

```
      lambda(1+niter,j)=sum(lambda(1:niter,j))/niter
      sigma2(1+niter,j)=sum(sigma2(1:niter,j))/niter
    enddo
    sigma2_tau(1+niter)=sum(sigma2_tau(1:niter))/niter
    varphi(1+niter)=sum(varphi(1:niter))/niter
    do i=1,nstu
      theta(1+niter,i)=sum(theta(1:niter,i))/niter
    enddo
    ak=a_hat
    bk=b_hat
    beta0k=beta0_hat
    beta1k=beta1_hat
    beta2k=beta2_hat
    lambdak=lambda(1+niter,1:nitem)
    sigma2k=sigma2(1+niter,1:nitem)
    sigma2_pdek=sigma2_pde_hat
    thetak=theta(1+niter,1:nstu)
    varphik=varphi(1+niter)
    sigma2_tauk=sigma2_tau(1+niter)
    ![ACPDE]
    call Ln_hat_ACPDE_RTACPDE(Ln_hat_acpde,thetak,ak,bk,
   1                          beta0k,beta1k,beta2k,sigma2_pdek,
   1                          PDE,Nit,ind,iresp,nstu,nitem)
    ![ACRTPDE]
    call Ln_hat_ACRTPDEjoint(Ln_hat_acrtpde,thetak,sigma2k,
   1                          sigma2_tauk,lambdak,varphik,ak,bk,
   1                          beta0k,beta1k,beta2k,sigma2_pdek,
   1                          PDE,Nit,ind,iresp,rt,nstu,nitem)
```

```
  urt_acpde=0.0d0
  do it=1,(1+niter)
     if(mod(it,2000).eq.0)then
            write(*,*)it
     endif
     lambdak=lambda(it,1:nitem)
     sigma2k=sigma2(it,1:nitem)
     varphik=varphi(it)
     sigma2_tauk=sigma2_tau(it)
     do istu=1,nstu
        t=rt(istu,1:nitem)
        irespi=iresp(istu,1:nitem)
        indi=ind(istu,1:nitem)
        Niti=Nit(istu)
        PDEi=PDE(istu)
        ![ACRTPDE]
        Ln_hati=Ln_hat_acrtpde(istu)
        call DQDAGI(fnacrtpde,bound,interv,errabs,errrel,
1                                              result,errest)
        uacrtpde(it,istu)=dlog(result)+Ln_hat_acrtpde(istu)
       ![ACPDE]
        Ln_hati=Ln_hat_acpde(istu)
      call DQDAGI(fnacpde,bound,interv,errabs,errrel,result,errest)
        uacpde(it,istu)=dlog(result)+Ln_hat_acpde(istu)
        urt_acpde(it,istu)=uacrtpde(it,istu)-uacpde(it,istu)
     enddo
  enddo
  LPMLrt_acpde=0.0d0
```

```
       do i=1,nstu
          umax=maxval(-urt_acpde(1:niter,i))
          !write(*,*)'umaix(',i,')=',umax
          yi=0.0d0
          do k=1,niter
             yi=yi+dexp(-urt_acpde(k,i)-umax)
          enddo
          yi=yi/niter
          LPMLrt_acpde=LPMLrt_acpde-umax-dlog(yi)
        enddo
       write(*,*)LPMLrt_acpde
 994     format('LPMLrt_acpde=',f12.6)
       resultsfile='LPMLdecomResults'//sequi//'_'//sequj//'_'
     1                                                //sequ//'.dat'
       open(unit=16,file=resultsfile,access='sequential',
     1             status='unknown')
       write(16,993)LPMLpde_acrt
       write(16,992)LPMLac_rtpde
       write(16,994)LPMLrt_acpde
       close(16)
c......ending time.
       call tdate(iday,month,iyear)
       call timdy(ihour,imin,isec)
       write(*,*)'ending time is'
       write(*,*)iyear,'-',month,'-',iday,' ',
     1           ihour,':',imin,':',isec
      enddo
      end
```

```
      include 'floatingControl.f'
      include 'function.f'
```

Under AC model only, the DIC and LPML are calculated as follow.

(xiv) **margresp_only.f**

```
     program responly
       implicit real*8 (a-h,o-z)
       parameter(nstu=125,nitem=20)
       parameter(npara=nstu+2+2*nitem)
       parameter(niter=10000)
       parameter(nrep0=270,nrep=270)
       parameter(pi=3.141592741012573)
       character(3)sequ
       character(26)respfile
       character(30)mcmcfile
       character(22)resultsfile
       character(3)sequi
       character(2)sequj
       integer iresp(nstu,nitem),ind(nstu,nitem)
       integer Nit(nstu),istu
       real*8  samples(1+niter,npara)
c......indicate the begainning time
       integer now(3),ntoday(3)
c......mcmc variables
       real*8  a(1+niter,nitem),b(1+niter,nitem)
       real*8  theta(1+niter,nstu)
c......dic lpml
       real*8 u(1+niter,nstu),Dev(1+niter),Dev_hat,Dev_bar,PD,umax,lpml
```

```
c......integral variable
       integer interv
       real*8  result
       real*8  bound,errabs,errrel,delta,errest
c......common
       real*8  Ln_hati,Ln_hat(nstu)
       real*8  ak(nitem),bk(nitem)
       integer irespi(nitem),Niti,indi(nitem)
       external fnaconly
       common /va/ak
       common /vb/bk
       common /vLn_hati/Ln_hati
       common /virespi/irespi
       common /vNiti/Niti
       common /vindi/indi
c......show beginning time.
       call tdate(iday,month,iyear)
       call timdy(ihour,imin,isec)
       write(*,*)'starting time is'
       write(*,*)iyear,'-',month,'-',iday,' ',
     1           ihour,':',imin,':',isec
c......read data
       write(sequi,'(i3.3)')nstu
       write(sequj,'(i2.2)')nitem
       do nsim=nrep0,nrep
       write(*,*)'nsim=',nsim
       write(sequ,'(i3.3)')nsim
       respfile='For_stu_resp'//sequi//'_'//sequj//'_'//sequ//'.csv'
```

```
      mcmcfile='For_samples_resp'//sequi//'_'//sequj//'_'//sequ//'.csv'
      do i=1,nstu
        Nit(i)=nitem
        do j=1,nitem
           ind(i,j)=j
        enddo
      enddo
      open(unit=21,file=respfile,status='old')
      do i=1,nstu
        read(21,*)iresp(i,1:Nit(i))
      enddo
      close(21)
      open(unit=22,file=mcmcfile,status='old')
      do i=1,niter
        read(22,*)samples(i,1:npara)
      enddo
      close(22)
c.....separate different parameters
      a(1:niter,1:nitem)=samples(1:niter,1:nitem)
      b(1:niter,1:nitem)=samples(1:niter,(1+nitem):(2*nitem))
      theta(1:niter,1:nstu)=samples(1:niter,
     1                              (2*nitem+2+1):(2*nitem+2+nstu))
      do j=1,nitem
         a(1+niter,j)=sum(a(1:niter,j))/niter
         b(1+niter,j)=sum(b(1:niter,j))/niter
      enddo
      do i=1,nstu
         theta(1+niter,i)=sum(theta(1:niter,i))/niter
```

```
        enddo
c......Ln_hat
        Ln_hat=-0.5d0*dlog(2*pi)
        do i=1,nstu
           yi=0.0d0
           do k=1,Nit(i)
              j=ind(i,k)
              ep=a(1+niter,j)*(theta(1+niter,i)-b(1+niter,j))
              if(ep.le.0.0d0)then
                      yi=yi+ep*iresp(i,k)-dlog(1.0d0+dexp(ep))
              else
                      yi=yi+ep*(iresp(i,k)-1)-dlog(1.0d0+dexp(-ep))
              endif
           enddo
           Ln_hat(i)=Ln_hat(i)+yi-0.5d0*theta(1+niter,i)**2
        enddo
c......integral
        interv=2
        bound=0.0d0
        errabs=0.0d0
        errrel=0.00001d0
        u=0.0d0
        Dev=0.0d0
        do k=1,(niter+1)
          if(mod(k,2000).eq.0)then
                  write(*,*)k
          endif
          ak=a(k,1:nitem)
```

```
    bk=b(k,1:nitem)
    do istu=1,nstu
       Ln_hati=Ln_hat(istu)
       Niti=Nit(istu)
       indi=ind(istu,1:nitem)
       irespi=iresp(istu,1:nitem)
       call DQDAGI(fnaconly,bound,interv,errabs,errrel,
1                  result,errest)
       u(k,istu)= -dlog(result)-Ln_hati
    enddo
    Dev(k)=2*sum(u(k,1:nstu))
  enddo
  Dev_hat=Dev(1+niter)
  Dev_bar=sum(Dev(1:niter))/niter
  PD=Dev_bar-Dev_hat
  DIC=Dev_hat+2*PD
  lpml=0.0d0
  do i=1,nstu
     umax=maxval(u(1:niter,i))
     yi=0.0d0
     do k=1,niter
        yi=yi+dexp(u(k,i)-umax)
     enddo
     yi=yi/niter
     lpml=lpml-umax-dlog(yi)
   enddo
  write(*,*)'dic=',DIC
  write(*,*)'pd=',PD
```

```
       write(*,*)'lpml=',lpml
c......save marginal DIC, LPML to file
       resultsfile='responly'//sequi//'_'//sequj//'_'//sequ//'.dat'
       open(unit=16,file=resultsfile,access='sequential',
     1             status='unknown')
       write(16,12)DIC
       write(16,13)PD
       write(16,14)lpml
 12       format(f12.6)
 13       format(f12.6)
 14       format(f12.6)
c......ending time.
       call tdate(iday,month,iyear)
       call timdy(ihour,imin,isec)
       write(*,*)'ending time is'
       write(*,*)iyear,'-',month,'-',iday,' ',
     1           ihour,':',imin,':',isec
      enddo
      end
c.....function for DQDAGI
      real*8 function fnaconly(x)
              implicit real*8 (a-h,o-z)
              parameter(nitem=20)
              parameter(pi=3.141592741012573)
              real*8 x,Ln,yi
              integer irespi(nitem)
              integer indi(nitem),Niti
              real*8  Ln_hati
```

```
      real*8  ak(nitem),bk(nitem)
      common /va/ak
      common /vb/bk
      common /vLn_hati/Ln_hati
      common /virespi/irespi
      common /vNiti/Niti
      common /vindi/indi
      Ln=-0.5d0*dlog(2*pi)-0.5d0*x**2
      yi=0.0d0
      do k=1,Niti
         j=indi(k)
         ep=ak(j)*(x-bk(j))
         if(ep.le.0.0d0)then
                 yi=yi+ep*(irespi(k))-dlog(1.0d0+dexp(ep))
         else
                 yi=yi+ep*(irespi(k)-1)-dlog(1.d0+dexp(-ep))
         endif
      enddo
      Ln=Ln+yi
      fnaconly=dexp(Ln-Ln_hati)
end
```

Additional References

Borgan, Ø., & Liestøl, K. (1990). A note on confidence intervals and bands for the survival function based on transformations. *Scandinavian Journal of Statistics*, 35–41.

Box, G. E. P. (1980). Sampling and Bayes' inference in scientific modelling and robustness (with discussion). *Journal of the Royal Statistical Society, Series A*, *143*(4), 383–404.

Chaimani, A., Higgins, J. P., Mavridis, D., Spyridonos, P., & Salanti, G. (2013). Graphical tools for network meta-analysis in stata. *PloS One*, *8*(10), e76654.

Chen, M.-H., Manatunga, A. K., & Williams, C. J. (1998). Heritability estimates from human twin data by incorporating historical prior information. *Biometrics*, 1348–1362.

Dey, D. K., Kuo, L., & Sahu, S. K. (1995). A Bayesian predictive approach to determining the number of components in a mixture distribution. *Statistics and Computing*, *5*(4), 297–305.

Geisser, S. (1987). Influential observations, diagnostics and discovery tests. *Journal of Applied Statistics*, *14*(2), 133–142.

Gelfand, A. E., Dey, D. K., & Chang, H. (1992). Model determination using predictive distributions with implementation via sampling-based-methods (with discussion). In A. P. D. J.M. Bernado J.O. Berger & A. Smith (Eds.), *In bayesian statistics 4.* Oxford: Oxford University Press.

Greenwood, M. (1926). The natural duration of cancer. *Reports on Public Health and Medical Subjects*(33).

Kaplan, E. L., & Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, *53*(282), 457–481.

Rücker, G., & Schwarzer, G. (2015). Ranking treatments in frequentist network meta-analysis works without resampling methods. *BMC Medical Research Methodology*, *15*(1), 58.

Salanti, G., Ades, A., & Ioannidis, J. P. (2011). Graphical methods and numerical summaries for presenting results from multiple-treatment meta-analysis: an overview and tutorial. *Journal of Clinical Epidemiology*, *64*(2), 163 - 171.

Wang, C., Xu, G., & Shang, Z. (2018). A two-stage approach to differentiating normal and aberrant behavior in computer based testing. *Psychometrika*, *83*(1), 223–254.

Wright, B. D., & Stone, M. H. (1979). *Best test design.* MESA Press: Chicago, IL.