ภาคผนวก

ภาคผนวก ก

ผลการวิเคราะห์ข้อมูล

ตารางที่ 4 ความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักเป็นจริง ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X และขนาดตัวอย่าง

ค่าผิดปกติจากกลุ่ม (%)	n	r_{xy}	r_s	τ_a	r_b
0	20	0.074(+)	0.052	0.052	0.062
	50	0.051	0.051	0.050	0.046
	100	0.055	0.050	0.050	0.053
5	20	0.063	0.056	0.053	0.059
	50	0.055	0.052	0.044	0.048
	100	0.057	0.053	0.055	0.054
10	20	0.065(+)	0.057	0.056	0.057
	50	0.062	0.060	0.054	0.052
	100	0.056	0.056	0.052	0.041
20	20	0.056	0.060	0.041	0.049
	50	0.061	0.057	0.051	0.054
	100	0.056	0.046	0.057	0.055
30	20	0.044	0.048	0.043	0.056
	50	0.055	0.056	0.053	0.060
	100	0.041	0.040	0.050	0.047

หมายเหตุ (+) หมายถึง ค่าความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักเป็นจริง มีค่ามากกว่าขอบเขตบนของ α.

ตารางที่ 5 ความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักเป็นจริง ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y และขนาดตัวอย่าง

ค่าผิดปกติจากกลุ่ม (%)	n	r_{xy}	r_s	τ_a	r_b
5	20	0.063	0.049	0.050	0.058
	50	0.054	0.051	0.057	0.045
	100	0.044	0.049	0.053	0.048
10	20	0.046	0.048	0.047	0.063
	50	0.058	0.053	0.051	0.053
	100	0.044	0.048	0.050	0.040
20	20	0.045	0.046	0.031(-)	0.044
	50	0.062	0.052	0.050	0.053
	100	0.047	0.038	0.037	0.036
30	20	0.045	0.052	0.045	0.050
	50	0.049	0.041	0.060	0.055
	100	0.041	0.040	0.044	0.043

หมายเหตุ (-) หมายถึง ค่าความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักเป็นจริง มีค่าน้อยกว่าขอบเขตล่างของ α^*

ตารางที่ 6 ความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักไม่เป็นจริง ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X

n	ρ	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)				
			0	5	10	20	30
20	0.20	r_{xy}	0.121	0.1	0.108	0.077	0.089
		r_s	0.118	0.115	0.101	0.117	0.097
		τ_a	0.102	0.099	0.091	0.104	0.084
		r_b	0.113	0.11	0.106	0.093	0.077
	0.50	r_{xy}	0.645	0.439	0.368	0.353	0.363
		r_s	0.582	0.546	0.525	0.481	0.443
		τ_a	0.57	0.53	0.512	0.466	0.428
		r_b	0.599	0.564	0.523	0.474	0.433
	0.80	r_{xy}	1	0.968	0.92	0.873	0.886
		r_s	0.992	0.989	0.984	0.961	0.951
		τ_a	0.99	0.988	0.978	0.966	0.938
		r_b	0.995	0.989	0.985	0.96	0.906
	1.00	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
50	0.20	r_{xy}	0.318	0.233	0.194	0.185	0.182
		r_s	0.273	0.267	0.262	0.22	0.22
		τ_a	0.27	0.268	0.25	0.216	0.206
		r_b	0.29	0.271	0.25	0.233	0.202
	0.50	r_{xy}	0.971	0.887	0.817	0.785	0.781
		r_s	0.949	0.938	0.918	0.904	0.876
		τ_a	0.946	0.939	0.919	0.89	0.871
		r_b	0.959	0.948	0.933	0.89	0.851

ตารางที่ 6 ความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักไม่เป็นจริง ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X (ต่อ)

n	ρ	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)				
			0	5	10	20	30
50	0.80	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
	1.00	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
100	0.20	r_{xy}	0.543	0.392	0.333	0.312	0.334
		r_s	0.485	0.45	0.431	0.396	0.391
		τ_a	0.483	0.465	0.435	0.418	0.372
		r_b	0.513	0.48	0.452	0.406	0.375
	0.50	r_{xy}	1	0.993	0.987	0.972	0.981
		r_s	1	1	0.999	0.998	0.996
		τ_a	1	0.999	0.998	0.996	0.995
		r_b	1	1	0.999	0.996	0.993
	0.80	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
	1.00	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1

ตารางที่ 7 ความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักไม่เป็นจริง ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y

n	ρ	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)				
			0	5	10	20	30
20	0.20	r_{xy}	0.121	0.109	0.097	0.102	0.082
		r_s	0.118	0.121	0.115	0.102	0.096
		τ_a	0.102	0.11	0.102	0.091	0.094
		r_b	0.113	0.108	0.109	0.102	0.08
	0.50	r_{xy}	0.645	0.457	0.363	0.352	0.335
		r_s	0.582	0.561	0.524	0.465	0.45
		τ_a	0.57	0.539	0.501	0.443	0.458
		r_b	0.599	0.565	0.516	0.451	0.395
	0.80	r_{xy}	1	0.954	0.913	0.87	0.855
		r_s	0.992	0.986	0.977	0.964	0.938
		τ_a	0.99	0.985	0.976	0.967	0.936
		r_b	0.995	0.991	0.985	0.959	0.905
	1.00	r_{xy}	1	1	1	0.999	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
50	0.20	r_{xy}	0.318	0.229	0.181	0.172	0.178
		r_s	0.273	0.266	0.248	0.231	0.211
		τ_a	0.27	0.246	0.243	0.233	0.21
		r_b	0.29	0.276	0.251	0.222	0.196
	0.50	r_{xy}	0.971	0.895	0.804	0.774	0.756
		r_s	0.949	0.938	0.925	0.891	0.855
		τ_a	0.946	0.932	0.92	0.891	0.854
		r_b	0.959	0.946	0.993	0.893	0.852

ตารางที่ 7 ความถี่สัมพัทธ์ของการปฏิเสธสมมติฐานหลักเมื่อสมมติฐานหลักไม่เป็นจริง ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y (ต่อ)

n	ρ	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)				
			0	5	10	20	30
50	0.80	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
	1.00	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
100	0.20	r_{xy}	0.543	0.381	0.364	0.282	0.302
		r_s	0.485	0.451	0.429	0.402	0.371
		τ_a	0.483	0.453	0.432	0.398	0.355
		r_b	0.513	0.475	0.451	0.385	0.346
	0.50	r_{xy}	1	0.998	0.987	0.975	0.971
		r_s	1	1	0.998	0.999	0.996
		τ_a	1	1	1	0.997	0.991
		r_b	1	0.999	0.999	1	0.988
	0.80	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1
	1.00	r_{xy}	1	1	1	1	1
		r_s	1	1	1	1	1
		τ_a	1	1	1	1	1
		r_b	1	1	1	1	1

ตารางที่ 8 ค่าประสิทธิภาพสัมพัทธ์ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X

ρ	n	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)			
			5	10	20	30
0.20	20	r_{xy}	0.8264	0.8926	0.6364	0.7355
		r_s	0.9746	0.8559	0.9915	0.822
		τ_a	0.9712	0.875	1.0096	0.8077
		r_b	0.9708	0.9343	0.8467	0.7591
	50	r_{xy}	0.7327	0.6101	0.5818	0.5723
		r_s	0.978	0.9597	0.8059	0.8059
		τ_a	0.9926	0.9259	0.8	0.763
		r_b	0.9211	0.8618	0.8322	0.6908
	100	r_{xy}	0.7219	0.6133	0.5746	0.6151
		r_s	0.9278	0.8887	0.8165	0.8062
		τ_a	0.9627	0.9006	0.8654	0.7702
		r_b	0.9308	0.8769	0.7981	0.7442
0.50	20	r_{xy}	0.6806	0.5705	0.5473	0.5628
		r_s	0.9381	0.9021	0.8265	0.7612
		τ_a	0.9298	0.8982	0.8175	0.7509
		r_b	0.9779	0.8927	0.8186	0.7461
	50	r_{xy}	0.9135	0.8414	0.8084	0.8043
		r_s	0.9884	0.9673	0.9526	0.9231
		τ_a	0.9926	0.9715	0.9408	0.9207
		r_b	0.9896	0.9699	0.9335	0.8941
	100	r_{xy}	0.993	0.987	0.972	0.981
		r_s	1	0.999	0.998	0.996
		τ_a	0.999	0.998	0.996	0.995
		r_b	1	0.999	0.996	0.993

ตารางที่ 8 ค่าประสิทธิภาพสัมพัทธ์ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X (ต่อ)

ρ	n	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)			
			5	10	20	30
0.80	20	r_{xy}	0.968	0.92	0.873	0.886
		r_s	0.996	0.991	0.9709	0.9299
		τ_a	0.997	0.9919	0.9688	0.9587
		r_b	0.998	0.9879	0.9758	0.9475
	50	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
	100	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
1.00	20	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
	50	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
	100	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1

ตารางที่ 9 ค่าประสิทธิภาพสัมพัทธ์ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y

ρ	n	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)			
			5	10	20	30
0.20	20	r_{xy}	0.9008	0.8017	0.843	0.6777
		r_s	1.0254	0.9746	0.8644	0.8136
		τ_a	1.0865	0.9808	0.8846	0.9231
		r_b	0.9635	0.927	0.8759	0.7226
	50	r_{xy}	0.7201	0.5692	0.5409	0.5597
		r_s	0.9744	0.9084	0.8462	0.7729
		τ_a	0.9111	0.9	0.8667	0.7778
		r_b	0.9441	0.8914	0.7697	0.6842
	100	r_{xy}	0.7017	0.6703	0.5193	0.5562
		r_s	0.9299	0.8845	0.8289	0.7649
		τ_a	0.9379	0.8944	0.824	0.735
		r_b	0.9269	0.8808	0.75	0.6731
0.50	20	r_{xy}	0.7085	0.5628	0.5457	0.5194
		r_s	0.9639	0.9003	0.799	0.7732
		τ_a	0.9456	0.8789	0.7772	0.8035
		r_b	0.9527	0.9038	0.8091	0.6987
	50	r_{xy}	0.9217	0.828	0.7971	0.7786
		r_s	0.9884	0.9747	0.9389	0.9009
		τ_a	0.9852	0.9725	0.9419	0.9027
		r_b	0.9844	0.973	0.9356	0.8982
	100	r_{xy}	0.998	0.987	0.975	0.971
		r_s	1	0.998	0.999	0.996
		τ_a	1	1	0.997	0.991
		r_b	0.999	0.999	1	0.988

ตารางที่ 9 ค่าประสิทธิภาพสัมพัทธ์ของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์และเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y (ต่อ)

ρ	n	ตัวประมาณค่า	ค่าผิดปกติจากกลุ่ม(%)			
			5	10	20	30
0.80	20	r_{xy}	0.954	0.913	0.87	0.855
		r_s	0.994	0.9849	0.9718	0.9456
		τ_a	0.9949	0.9859	0.9768	0.9455
		r_b	0.996	0.992	0.9689	0.9279
	50	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
	100	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
1.00	20	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
	50	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1
	100	r_{xy}	1	1	1	1
		r_s	1	1	1	1
		τ_a	1	1	1	1
		r_b	1	1	1	1

ตารางที่ 10 ค่าเฉลี่ยความคลาดเคลื่อนกำลังสองของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์ ขนาดตัวอย่างและเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X

ρ	n	ค่าผิดปกติจากกลุ่ม(%)	ตัวประมาณค่า			
			r_{xy}	r_s	τ_a	r_b
0.00	20	5	0.0201	0.0105	0.0145	0.0072
		10	0.0256	0.0123	0.0154	0.0111
		20	0.029	0.0178	0.0198	0.0176
		30	0.0311	0.0203	0.0208	0.0259
	50	5	0.0067	0.0028	0.0044	0.0016
		10	0.0091	0.0039	0.0056	0.0033
		20	0.0117	0.0056	0.0064	0.0062
		30	0.0113	0.0075	0.0077	0.0086
	100	5	0.0033	0.0013	0.0021	0.0009
		10	0.005	0.0018	0.0024	0.0016
		20	0.0062	0.0026	0.003	0.0025
		30	0.0052	0.0029	0.0033	0.0036
0.20	20	5	0.0186	0.0087	0.016	0.0063
		10	0.0268	0.0108	0.0179	0.0098
		20	0.0298	0.0158	0.0216	0.0165
		30	0.0289	0.0182	0.0225	0.022
	50	5	0.0072	0.0027	0.0092	0.0018
		10	0.0109	0.0041	0.0106	0.0037
		20	0.0132	0.0062	0.0128	0.0066
		30	0.0124	0.0067	0.0141	0.0093
	100	5	0.0044	0.0014	0.0074	0.001
		10	0.007	0.0022	0.0081	0.0017
		20	0.0085	0.0035	0.0094	0.0034
		30	0.0077	0.0042	0.0108	0.0049

ตารางที่ 10 ค่าเฉลี่ยความคลาดเคลื่อนกำลังสองของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์ ขนาดตัวอย่างและเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X (ต่อ)

ρ	n	ค่าผิดปกติจากกลุ่ม(%)	ตัวประมาณค่า			
			r_{xy}	r_s	τ_a	r_b
0.50	20	5	0.0234	0.009	0.0346	0.0064
		10	0.0353	0.012	0.0395	0.0111
		20	0.0449	0.0198	0.0483	0.019
		30	0.0392	0.0232	0.0545	0.0314
	50	5	0.0115	0.0033	0.0314	0.0017
		10	0.0213	0.0057	0.0364	0.005
		20	0.0263	0.0086	0.0427	0.0096
		30	0.0262	0.0115	0.0478	0.0159
	100	5	0.0104	0.0025	0.0332	0.0015
		10	0.017	0.004	0.0367	0.0032
		20	0.0232	0.0068	0.0426	0.0075
		30	0.0223	0.009	0.0477	0.0121
0.80	20	5	0.0293	0.0069	0.0543	0.0067
		10	0.0471	0.0106	0.0633	0.011
		20	0.0612	0.0195	0.082	0.0244
		30	0.0577	0.0251	0.0945	0.0409
	50	5	0.0177	0.0025	0.0504	0.0017
		10	0.0392	0.0053	0.0608	0.0053
		20	0.0517	0.0105	0.0761	0.0136
		30	0.0497	0.0152	0.0893	0.0258
	100	5	0.0209	0.0018	0.0515	0.0013
		10	0.0375	0.0036	0.0602	0.0038
		20	0.0491	0.0079	0.0756	0.0109
		30	0.049	0.0124	0.0876	0.0211

ตารางที่ 10 ค่าเฉลี่ยความคลาดเคลื่อนกำลังสองของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์ ขนาดตัวอย่างและเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร X (ต่อ)

ρ	n	ค่าผิดปกติจากกลุ่ม(%)	ตัวประมาณค่า			
			r_{xy}	r_s	τ_a	r_b
1.00	20	5	0.0351	0.0015	0.0039	0.0025
		10	0.0622	0.0046	0.0115	0.0067
		20	0.079	0.0118	0.0322	0.0203
		30	0.0774	0.0192	0.0523	0.043
	50	5	0.0244	0.0006	0.0018	0.0009
		10	0.0563	0.0028	0.0091	0.0048
		20	0.0738	0.0083	0.0273	0.0155
		30	0.0739	0.0141	0.0479	0.0322
	100	5	0.0308	0.0007	0.0025	0.0011
		10	0.0559	0.0025	0.0087	0.0039
		20	0.0738	0.0073	0.0267	0.0137
		30	0.0726	0.0123	0.0457	0.0277

ตารางที่ 11 ค่าเฉลี่ยความคลาดเคลื่อนกำลังสองของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์ ขนาดตัวอย่างและเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y

ρ	n	ค่าผิดปกติจากกลุ่ม(%)	ตัวประมาณค่า			
			r_{xy}	r_s	τ_a	r_b
0.00	20	5	0.02	0.0103	0.014	0.0071
		10	0.0272	0.0136	0.0163	0.0103
		20	0.0306	0.0177	0.0196	0.0198
		30	0.0317	0.0218	0.0217	0.024
	50	5	0.0069	0.0027	0.0046	0.0019
		10	0.0101	0.0039	0.0053	0.0032
		20	0.0109	0.0055	0.0068	0.0059
		30	0.0109	0.0066	0.0075	0.0078
	100	5	0.0038	0.0013	0.0021	0.0009
		10	0.0047	0.0018	0.0024	0.0015
		20	0.0056	0.0025	0.003	0.0026
		30	0.0053	0.0033	0.0034	0.0042
0.20	20	5	0.0197	0.0087	0.015	0.0061
		10	0.0292	0.0116	0.0176	0.009
		20	0.0309	0.0161	0.0213	0.016
		30	0.0303	0.0205	0.0251	0.0235
	50	5	0.0074	0.0027	0.0092	0.0017
		10	0.0127	0.0045	0.0107	0.0037
		20	0.0139	0.0057	0.0127	0.0063
		30	0.0134	0.0076	0.0144	0.0097
	100	5	0.0052	0.0016	0.0074	0.0009
		10	0.007	0.0022	0.0086	0.0019
		20	0.0093	0.0037	0.01	0.0038
		30	0.0088	0.0043	0.0111	0.0059

ตารางที่ 11 ค่าเฉลี่ยความคลาดเคลื่อนกำลังสองของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์ ขนาดตัวอย่างและเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y (ต่อ)

ρ	n	ค่าผิดปกติจากกลุ่ม(%)	ตัวประมาณค่า			
			r_{xy}	r_s	τ_a	r_b
0.50	20	5	0.0234	0.0091	0.0346	0.0071
		10	0.0356	0.0122	0.0393	0.0109
		20	0.0427	0.02	0.0478	0.0194
		30	0.0416	0.0235	0.052	0.0319
	50	5	0.0101	0.0034	0.0317	0.0022
		10	0.0216	0.0054	0.0367	0.0046
		20	0.0273	0.0092	0.0434	0.0103
		30	0.029	0.0127	0.0489	0.0171
	100	5	0.011	0.0025	0.0335	0.0017
		10	0.0185	0.004	0.0366	0.0033
		20	0.0236	0.0071	0.0431	0.008
		30	0.024	0.0097	0.0471	0.0144
0.80	20	5	0.0301	0.008	0.0534	0.006
		10	0.0487	0.0117	0.0641	0.0112
		20	0.0616	0.0196	0.0803	0.0247
		30	0.0631	0.0293	0.0955	0.0431
	50	5	0.018	0.0027	0.05	0.0019
		10	0.0404	0.0056	0.0613	0.0057
		20	0.052	0.011	0.0779	0.0139
		30	0.0481	0.0146	0.0889	0.0258
	100	5	0.0212	0.0019	0.052	0.0014
		10	0.0377	0.0038	0.0607	0.0039
		20	0.0485	0.0084	0.0763	0.0118
		30	0.0497	0.0129	0.0891	0.0224

ตารางที่ 11 ค่าเฉลี่ยความคลาดเคลื่อนกำลังสองของตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์ทั้ง 4 แบบ จำแนกตามค่าสัมประสิทธิ์สหสัมพันธ์ ขนาดตัวอย่างและเปอร์เซนต์ของค่าผิดปกติจากกลุ่มที่เกิดในตัวแปร Y (ต่อ)

ρ	n	ค่าผิดปกติจากกลุ่ม(%)	ตัวประมาณค่า			
			r_{xy}	r_s	τ_a	r_b
1.00	20	5	0.0358	0.0018	0.0038	0.0021
		10	0.0618	0.0049	0.0113	0.0066
		20	0.0804	0.0122	0.0328	0.0213
		30	0.0769	0.0199	0.0538	0.0428
	50	5	0.0245	0.0006	0.0019	0.0009
		10	0.0566	0.0029	0.0093	0.0046
		20	0.0751	0.0082	0.0287	0.0155
		30	0.0735	0.0139	0.0474	0.0322
	100	5	0.0307	0.0007	0.0025	0.0012
		10	0.0558	0.0024	0.0087	0.0039
		20	0.0738	0.0074	0.0269	0.0137
		30	0.0724	0.0122	0.046	0.0275

ภาคผนวก ข

เทคนิคการผลิตเลขสุ่มที่ใช้ในงานวิจัย

เทคนิคสำหรับการผลิตเลขสุ่มที่ใช้ในการวิจัย

วิธีการทางคณิตศาสตร์ ในการจำลองเลขสุ่ม (เทียม) มีหลายวิธีการ ในการวิจัยนี้เลือกใช้วิธีตัวแบบจำลองสมภาคแบบผสม(Mixed Congruential Method) ซึ่งเป็นวิธีจำลองเลขสุ่มที่ปรับปรุงจากวิธีการจำลองแบบ "วิธีสมภาค" (Congruential Method) ซึ่งมีสูตรหรือตัวแบบ คือ

$$X_i = (aX_{i-1} + c) \bmod m \; ; \; i = 0,1,2,... \qquad (a)$$

โดยทั่วไปค่า c,a และ m เป็นค่าคงที่จำนวนเต็มไม่เป็นลบ คำว่า mod คือ modulus และความหมายของตัวแบบคือ X_i เป็นเศษเหลือที่เป็นจำนวนเต็มที่ได้จากการหาร $(c + aX_{i-1})$ ด้วย m นั่นคือ $X_i = (c + aX_{i-1} - mk_i)$ ซึ่ง $k_i = [\,(c + aX_{i-1})/m\,]$ (หมายถึง จำนวนเต็มใหญ่ที่สุดที่น้อยกว่าหรือเท่ากับผลหาร $(c + aX_{i-1})/m$ ดังนั้นค่าที่เป็นไปได้ของ X_i คือ 1,2,...,m-1 และก่อนที่จะได้ค่าของ $X_1, X_2, ...$ ต้องกำหนดค่าของ c,a ,m และ X_0 เราเรียก X_0 ว่า "ซีด" (Seed) หรือ "ค่าเริ่มต้น" (Starting Value) จาก X_i ที่ได้จากการคำนวณนำมาหาค่า R_i ซึ่ง

$$R_i = \frac{X_i}{m} \quad i = 1,2,... \qquad (b)$$

จะได้ R_i มีค่าอยู่ระหว่าง (0,1) เรียก $R_1, R_2, R_3, ...$ ว่าเป็น "เลขสุ่มเทียม" หรือ "เลขสุ่มคล้าย" ทั้งนี้เพราะเมื่อทราบค่าเริ่มต้น X_0 ค่าต่อไปจะมีค่าที่แน่นอนตามสูตร และทุกครั้งที่เริ่มด้วย X_0 ค่าเดิม (ขณะที่ค่าของ c,a และ m ไม่เปลี่ยนแปลง) จะได้ X_i เป็นเลขชุดเดิม นอกจากนี้ได้ว่า

$$X_i = aX_0 + c - mk_1, k_1 = [(aX_0 + c)/m]$$

$$X_2 = a^2X_0 + ac - amk_1 + c - mk_2$$

$$= a^2X_0 + c(1 + a) - m(k_2 + k_1a)$$

$$X_3 = a^3X_0 + a^2c - a^2mk_1 + ac - amk_2 + c - mk_3$$

$$= a^3X_0 + c(1 + a + a^2) - m(k_3 + k_2a + k_1a^2)$$

$$X_i = a^iX_0 + c(1 + a + a^2 + ... + a^{i-1}) - m(k_i + k_{i-1} + ... + k_1a^{i-1})$$

$$= \left(a^iX_0 + c\frac{1 - a^i}{1 - a}\right) \bmod m$$

ซึ่งแสดงให้เห็นว่าแต่ละค่า X_i คำนวณมาจากค่าคงที่ c,a ,m และ X_0 เท่านั้น ด้วยลักษณะดังกล่าวจึงไม่ใช่การผลิตเลขสุ่มอย่างแท้จริง อย่างไรก็ตามการเลือกกำหนดค่า c,a ,m และ X_0 อย่างเหมาะสมจะได้ค่า $X_1, X_2, ...$ เสมือนได้มาจากการแจกแจง U(0,1) และเป็นอิสระกัน

ถ้ากำหนด c>0 เราเรียกตัวแบบ (a) ว่า "ตัวแบบจำลองสมภาคแบบผสม" (Mixed Congruential Method) ในงานวิจัยนี้กำหนดให้

$$a = 4t + 1 ; t = 2$$

c = ค่าส่วนเพิ่มที่กำหนดเป็นเลขจำนวนเต็มคี่ ในงานวิจัยนี้กำหนดให้ c = 5

$$m = 2,147,483,648$$

X_0 = seed number มีค่าอยู่ระหว่าง 0 ถึง (m-1)

ตัวแบบจำลองสมภาคแบบผสมนิยมใช้กันมากตัวแบบหนึ่ง ซึ่งได้ผ่านการตรวจสอบคุณสมบัติแล้วอย่างมาก คือ สำหรับคอมพิวเตอร์ 32 บิตต่อ 1 คำ (32 -bit word) กำหนด $m = 2^{31} - 1 = 2147483647$, $a = 7^5 = 16807$ และ X_0 เป็นจำนวนเต็มบวกไม่เกิน m ตัวแบบตามข้อกำหนด m และ a นี้ จะได้ความยาวคาบสูงสุดที่เป็นไปได้ m-1 = 2147483646 และด้วยค่า m และ a นี้มีใช้ในหลายซอฟต์แวร์ (Software) เช่นในโปรแกรม สำเร็จรูป IBM SL/MATH, LLRANDOM และ IMSL (International Mathematics and Statistics Library) ใช้ในภาษาจำลอง SIMPL/I และใช้ในภาษาโปรแกรมคอมพิวเตอร์ฟอร์แทรน (FORTRAN) ตัวคูณ a อีกตัวหนึ่งที่ใช้ได้ดีคู่กับ $m = 2^{31}-1$ คือ a = 630360016 สำหรับคอมพิวเตอร์ 36 บิตต่อ 1 คำ ค่า m และ a ที่ใช้ได้ดีคู่หนึ่งคือ $m = 2^{35}-31$ และ $a = 5^5 = 3125$

การผลิตตัวเลขสุ่มที่มีรูปแบบการแจกแจงแบบต่าง ๆ ที่ใช้ในการวิจัยนี้ต้องใช้เลขสุ่มได้มาจากการแจกแจง U(0,1) มาช่วยในการสร้างตัวผลิตเลขสุ่มปกติมาตรฐาน (Standard Normal Process Generator) ซึ่งในการวิจัยครั้งนี้ใช้วิธีการผลิตเลขสุ่มปกติด้วยวิธีรับ-ปฏิเสธ ซึ่งมีรายละเอียดดังนี้

การจำลองตัวแปรสุ่มปกติด้วยวิธีรับ-ปฏิเสธ

ให้ W เป็นตัวแปรสุ่มมีฟังก์ชันความหนาแน่น

$$f_W(w) = \sqrt{\frac{2}{\pi}} e^{\frac{-w^2}{2}} \quad , w \geq 0$$

เราเรียกการแจกแจงของ W ว่า "การแจกแจงปกติมาตรฐานบวก" (Positive Standard Normal Distribution) เพราะ $W = |Z|$ เราจะจำลอง W ด้วยวิธีรับ-ปฏิเสธ จากนั้น ให้ Z เท่ากับ W หรือ -W ด้วยความน่าจะเป็นเท่ากัน

เลือก $Y \sim Ex(\lambda = 1)$ เป็นตัวแปรสุ่มที่จะจำลองเป็นค่าของ W ได้ว่า Y มีฟังก์ชันความหนาแน่น

$$f_y(y) = e^{-y}, y \geq 0$$

ให้

$$m = \max_{y \geq 0} \left\{ \frac{f_W(y)}{f_y(y)} \right\} = \max_{y \geq 0} \left\{ \sqrt{\frac{2}{\pi}} e^{y - y^2/2} \right\}$$

ซึ่ง $\sqrt{\frac{2}{\pi}} e^{y-y^2/2}$ มีค่ามากที่สุดเมื่อ $y - \frac{1}{2}y^2$ มีค่ามากที่สุด โดยใช้แคลคูลัส หาค่า y จากสมการ

$$\frac{d}{dy}\left(y - \frac{1}{2}y^2 \right) = 1 - y = 0$$

ได้ $y = 1$ ให้ $y - \frac{1}{2}y^2$ มีค่ามากที่สุดเท่ากับ $\frac{1}{2}$ ดังนั้นได้

$$m = \sqrt{\frac{2e}{\pi}}$$

และ

$$\frac{f_w(w)}{mf_y(y)} = \exp\left(y - \frac{1}{2}y^2 - \frac{1}{2}\right) = \exp\left\{-\frac{1}{2}\left(y^2 - 2y + 1\right)\right\}$$

$$= \exp\left(-\frac{1}{2}(y - 1)^2\right)$$

เพราะฉะนั้นขั้นตอนวิธีสำหรับจำลอง $Z \sim N(0,1)$ ด้วยวิธีรับ-ปฏิเสธ มีดังนี้

(1) จำลองเลขสุ่ม R_i และให้ $Y = \ell nR_1$

(2) จำลองเลขสุ่ม R_2

(3) ถ้า $R_2 > = \exp\left(-\frac{1}{2}(Y - 1)^2\right)$ กลับไปขั้นตอน (1)

(4) จำลองเลขสุ่ม R_3

(5) ถ้า $R_3 \leq 0.5$ ให้ $Z = Y$ มิฉะนั้นให้ $Z = -Y$

เนื่องจากเงื่อนไข $R_2 > = \exp\left(-\frac{1}{2}(Y - 1)^2\right)$ เขียนได้ใหม่เป็น $-\ell nR_2 < \left(\frac{1}{2}(Y - 1)^2\right)$ ซึ่ง $-\ell nR_2$ ก็คือตัวแปรสุ่มเลขชี้กำลังที่มีค่าเฉลี่ยเท่ากับ 1 เพราะฉะนั้น สามารถเขียนขั้นตอน (2) และ(3) ได้ใหม่เป็น

(2) จำลองเลขสุ่ม R_2 และให้ $V = -\ell nR_2$

(3) ถ้า $V < \left(\frac{1}{2}(Y - 1)^2\right)$ กลับไปขั้นตอน (1)

ขั้นตอนวิธีข้างต้น มีจำนวนครั้งทำซ้ำจนกว่าจะเข้าขั้นตอน (4) (ได้ค่า Z) เท่ากับ $m = \sqrt{\frac{2e}{\pi}} \approx 1.315$ ครั้งโดยเฉลี่ย และมีประสิทธิภาพ $= \frac{1}{m} \approx 0.7602$ หรือประมาณ 76.02 %

ภาคผนวก ค

โปรแกรมจำลองแบบ

ลักษณะการเขียนโปรแกรม

การเขียนโปรแกรมใช้เทคนิค Object Oriented Programming โดยใช้ภาษาซีชาร์บ (C#) เป็นเครื่องมือในการเขียนโปรแกรม โดยในโปรแกรมนั้นจะเขียนแยกตัวผลิตและตัวทดสอบแยกกันเป็นคนละ Class เมื่อเขียนเสร็จแล้ว นำเอา Class เหล่านั้นมาทำเป็น Object เรียกใช้ตัวโปรแกรมอีกที

//ตัวจำลองข้อมูลสุ่ม

```
// FILENAME : MCSRandom.cs
using System;
using System.Collections.Generic;
using System.Text;
namespace RCC
{
    /// <summary>
    /// ตัวจำลองข้อมูลสุ่ม
    /// </summary>
    public class MCSRandom
    {
        private int t = 2;
        private int k = 9;              // 4t+1
        private int c = 5;
        private uint m = 2147483648;
        private int seed = int.Parse(Properties.Settings.Default.seed);
        private float random = 0;
        /// <summary>
        /// สร้าง instance สำหรับการสุ่มจาก seed ที่กำหนดโดย configuration
        /// </summary>
        public MCSRandom(){
            random = seed;
        }
        /// <summary>
        /// สร้าง instance สำหรับการสุ่มจาก seed ที่กำหนด
        /// </summary>
        /// <param name="seed">ค่าเริ่มต้นการสุ่ม</param>
        public MCSRandom( int seed )
        {
            this.random = this.seed = seed;
        }
        /// <summary>
```

```
/// สุ่มตัวเลขตัวถัดไป
/// </summary>
public float Next()
{
   random = (k * random + c) % m;
   return random / (m - 1);
}

public int T
{
   get { return t; }
   set {
      t = value;
      k = 4 * t + 1;
   }
}
public int K
{
   get { return k; }
   set { k = value; }
}
public int Seed {
   get{ return seed; }
}
/// <summary>
/// Set and save new seed
/// </summary>
public static int SSeed
{
   get { return int.Parse(Properties.Settings.Default.seed); }
   set {
      Properties.Settings.Default.seed = value + "";
      Properties.Settings.Default.Save();
   }
}
public int C {
   get { return c; }
   set { c = value;
```

```
            if (c % 2 == 0)
                throw new ValueOutOfRangeException( "The C value must be odd number." );
        }
    }
    public uint M{
        get { return m; }
        set { m = value; }
    }
    public int N {
        set { m = (uint)Math.Pow(2, N); }
        get { return (int)Math.Log(m, 2); }
    }
  }
}
```

//การสุ่มตัวอย่าง

// FILENAME : RandomData.cs

```
using System;
using System.Data;
using System.IO;
using System.Text;
using System.Diagnostics;
namespace RCC
{
  /// <summary>
  /// ชุดตัวอย่างที่สุ่มขึ้นมาสำหรับการจำลอง
  /// </summary>
  class RandomData : Statistic
  {
    private DataTable  data = new DataTable();
    private float [ ]   rhoConstants    = null;
    private MCSRandom   random  = null;
    private MCSRandom   rho_0_random = null;
    private int   count     = 0;
    private Outlier outlier  = null;
```

```
/// <summary>
/// สร้าง instance สำหรับการสุ่มตัวอย่าง
/// </summary>
/// <param name="count">จำนวนตัวอย่างที่จะสุ่ม</param>
/// <param name="rhoConstants">ชุดของค่าความสัมพันธ์ ตัวแรกต้องมีค่าเป็น 0 เสมอ</param>
public RandomData(int count, float [ ] rhoConstants )
{
   this.data.TableName = "(Count="+count+")";
   this.rhoConstants = rhoConstants;
   this.random = new MCSRandom( );
   this.rho_0_random = new MCSRandom(random.Seed + 1);

   this.count = count;

   DataColumn c = new DataColumn("X");
   c.DataType = typeof(float);
   this.data.Columns.Add(c);

   foreach (float rho in rhoConstants)
   {
      c = new DataColumn( "Y(p="+ rho + ")");
      c.DataType = typeof(float);
      this.data.Columns.Add( c );
   }
}

// destructor
~RandomData()
{
   data.Dispose();
}

/// <summary>
/// เริ่มการสุ่มตัวอย่าง
/// </summary>
public void SimulateData( )
{
   this.data.Clear();
```

```
    this.Generate();
    base.Source = this.data;
}

/// <summary>
/// สุ่มตัวอย่างทีละ 1 ตัวอย่าง
/// </summary>
/// <returns>ค่าที่สุ่มได้ตามเงื่อนไขของการจำลอง</returns>
private float GenerateValue( )
{
    float exp = 0;
    float r1 = 0;
    float r2 = 0;
    float r3 = 0;
    float y = 0;
    float v = 0;
    do
    {
        r1 = this.random.Next();
        y = (float)-Math.Log(r1);

        r2 = this.random.Next();
        v = (float)-Math.Log(r2);

        exp = ((y - 1) * (y - 1)) * 0.5f;
    } while (v < exp);

    r3 = this.random.Next();
    return (r3 <= 0.5f) ? y : -y;
}

/// <summary>
/// สุ่มตัวอย่างทีละ 1 ตัวอย่าง ใช้สำหรับการสุ่มเมื่อค่า rho เป็น 0
/// </summary>
/// <param name="psudoNext">สุ่มตัวอย่างสำหรับค่า rho เป็น 0</param>
/// <returns>ค่าที่สุ่มได้ตามเงื่อนไขของการจำลอง</returns>
private float GenerateValue(bool psudoNext)
{
```

```
    float exp = 0;
    float r1 = 0;
    float r2 = 0;
    float r3 = 0;
    float y = 0;
    float v = 0;
    do
    {
        r1 = this.rho_0_random.Next();
        y = (float)-Math.Log(r1);

        r2 = this.rho_0_random.Next();
        v = (float)-Math.Log(r2);

        exp = ((y - 1) * (y - 1)) * 0.5f;
    } while (v < exp);

    r3 = this.rho_0_random.Next();
    return (r3 <= 0.5f) ? y : -y;
}

/// <summary>
/// เริ่มการสุ่มตัวอย่าง
/// </summary>
private void Generate()
{
    for (int i = 0; i < count; i++)
    {
        float x = 0;
        float[] y = new float[this.rhoConstants.Length];
        x = GenerateValue();

        // คำนวณ yi ตามจำนวนของค่า rho
        // คำนวณ yi เมื่อ rho เป็น 0
        y[0] = GenerateValue(true);
        for (int ri = 1; ri < rhoConstants.Length; ri++)
        {
            if (rhoConstants[ri] == 1)
```

```
            {
                // คำนวณ yi เมื่อ rho เป็น 1
                y[ri] = x;
            }
            else
            {

                // คำนวณ yi ณ rho ใดๆ
                float d = GenerateValue();
                y[ri] = x * rhoConstants[ri] + d *(float)Math.Sqrt(1.0f - (rhoConstants[ri] *
rhoConstants[ri]));
            }
        }

        // เพิ่มข้อมูลลงในตาราง
        string[ ] row = new string[rhoConstants.Length + 1];
        row[0] = x + "";
        for (int vi = 0; vi < rhoConstants.Length; vi++)
            row[vi + 1] = y[vi] + "";

        this.data.Rows.Add(row);
    }
}

/// <summary>
/// บันทึกตัวอย่างที่ได้
/// </summary>
/// <param name="filename">ชื่อและที่อยู่ของไฟล์ที่จะบันทึก</param>
public void SaveXml(string filename)
{
    StreamWriter w = null;
    try
    {
        w = new StreamWriter(filename);
        string header = "<?xml version=\"1.0\"?>"
                + "\r\n<Workbook xmlns=\"urn:schemas-microsoft-com:office:spreadsheet\""
                + "\r\nxmlns:o=\"urn:schemas-microsoft-com:office:office\""
                + "\r\nxmlns:x=\"urn:schemas-microsoft-com:office:excel\""
```

```
                    + "\r\nxmlns:ss=\"urn:schemas-microsoft-com:office:spreadsheet\""
                    + "\r\nxmlns:html=\"http://www.w3.org/TR/REC-html40\">";
            w.WriteLine(header);
            w.WriteLine(" <Worksheet ss:Name=\"" + data.TableName + "\">");
            w.WriteLine("  <Table>");
            w.WriteLine("   <Row>");
            for (int c = 0; c < data.Columns.Count; c++)
            {
               w.WriteLine("    <Cell><Data ss:Type=\"String\">" + data.Columns[c] +
"</Data></Cell>");
            }
            w.WriteLine("   </Row>");
            for (int r = 0; r < data.Rows.Count; r++)
            {
               w.WriteLine("   <Row>");
               for (int c = 0; c < data.Columns.Count; c++)
                  w.WriteLine("    <Cell><Data ss:Type=\"Number\">" + data.Rows[r][c] +
"</Data></Cell>");
               w.WriteLine("   </Row>");
            }
            w.WriteLine("  </Table>");
            w.WriteLine(" </Worksheet>");
            w.WriteLine("</Workbook>");
         }
         catch (Exception ex)
         {
            throw ex;
         }
         finally
         {
            if( w != null )
               w.Close();
         }
      }
      /// <summary>
      /// ตัวคำนวณ outlier สำหรับชุดตัวอย่างนี้
      /// </summary>
      public Outlier Outlier
```

```
        {
            get
            {
                if( this.outlier == null )
                    return (this.outlier = new Outlier(this));
                return this.outlier;
            }
        }
        /// <summary>
        /// ชุดของค่าความสัมพันธ์ ตัวแรกต้องเป็รค่า 0 เสมอ
        /// </summary>
        public float[ ] RhoConstants {
            get { return this.rhoConstants; }
        }
        /// <summary>
        /// ชุดตัวอย่างที่สุ่มได้
        /// </summary>
        public DataTable Data
        {
            get { return this.data; }
        }
    }
}
```

//การคำนวณค่าควอไทล์

// FILENAME : Quatile.cs

```
using System;
using System.Data;

namespace RCC
{
    /// <summary>
    /// ตัวคำนวณค่า quatile
    /// </summary>
    class Quatile
    {
        private float value = 0;
```

```
private float position = 0;
private int quatile = 0;
private int column = 0;
private DataView data = null;
/// <summary>
/// สร้างตัวคำนวณค่า quatile จากตารางข้อมูลของตัวอย่างที่กำหนด
/// </summary>
/// <param name="data">ตารางข้อมูลของตัวอย่าง</param>
public Quatile( DataTable data ) {
   this.data = new DataView( data.Copy() );

   this.data.AllowEdit = false;
   this.data.AllowNew = false;
   this.data.AllowDelete = false;
}

/// <summary>
/// สร้างตัวคำนวณค่า quatile จากตารางข้อมูลของตัวอย่างที่กำหนด
/// </summary>
/// <param name="data">ตารางข้อมูลของตัวอย่าง</param>
/// <param name="column">ค่าเริ่มต้นของคอลัมน์ที่จะคำนวณหาค่า quatile </param>
public Quatile(DataTable data, int column )
{
   this.data = new DataView(data.Copy());

   this.data.AllowEdit = false;
   this.data.AllowNew = false;
   this.data.AllowDelete = false;
   this.column = column;
}
/// <summary>
/// คำนวณค่า quatile
/// </summary>
/// <param name="quatile">ตำแหน่ง quatile ที่ต้องการ</param>
/// <param name="column">คอลัมน์ที่จะคำนวณค่า quatile</param>
/// <returns>ค่า quatile ณ ตำแหน่งที่ระบุ</returns>
private float QuatileValue(int quatile, int column)
{
```

```
    this.quatile = quatile;
    this.column  = column;

    string cName = this.data.Table.Columns[column].ColumnName;
    this.data.Sort = cName + " ASC";
    this.position = (quatile * (data.Count + 1)) / 4.0f;

    float qa = (float)this.data[(int)position - 1][column];
    float qb = (float)this.data[(int)position][column];
    float diffPoint = (position - (int)position);
    this.value =  qa + (diffPoint * ( qb - qa));
    //this.data.Sort = "";
    return this.value;
}

/// <summary>
/// คอลัมน์ปัจจุบันที่จะคำนวณค่า quatile
/// </summary>
public int Column
{
    get { return this.column; }
    set { this.column = value; }
}

/// <summary>
/// ค่า quatile ที่ 1
/// </summary>
public float Quatile1
{
    get { return QuatileValue(1, this.column); }
}
/// <summary>
/// ค่า quatile ที่ 1
/// </summary>
public float Quatile2
{
    get { return QuatileValue(2, this.column); }
}
```

```
        /// <summary>
        /// ค่า quatile ที่ 1
        /// </summary>
        public float Quatile3
        {
            get { return QuatileValue(3, this.column); }
        }

        /// <summary>
        /// ตำแหน่งของ quatile แต่ละค่า
        /// </summary>
        public float[] QuatilePosition
        {
            get {
                float[] q = new float[3];
                for (int i = 0; i < 3; i++)
                {
                    q[i] = ((i+1) * (data.Count + 1)) / 4.0f;
                }

                return q;
            }
        }
    }
}
```

//การสร้างค่าผิดปกติจากกลุ่ม

// FILENAME : Outlier.cs

```
using System;
using System.Collections;
using System.Data;
using System.IO;
namespace RCC
{
    /// <summary>
    /// แกนที่จะจำลองการเกิด outliers
    /// </summary>
```

```
enum OutlierAxis { X, Y }

/// <summary>
/// จำลองการเกิด outliers
/// </summary>
class Outlier
{
   private RandomData source = null;
   private OutlierAxis axis = OutlierAxis.Y;
   private int  percent = -1;

   /// <summary>
   /// สร้าง instance สำหรับการจำลองการเกิด outliers
   /// </summary>
   /// <param name="source">ตัวอย่างที่จะจำลองการเกิด outliers</param>
   public Outlier(RandomData source)
   {
      this.source = source;
   }

   /// <summary>
   /// เลือกแกนที่จะจำลองการเกิด outliers
   /// </summary>
   public OutlierAxis Axis
   {
      get { return this.axis; }
      set { this.axis = value; }
   }

   /// <summary>
   /// Simulate outliers on X or Y axis.
   /// </summary>
   /// <param name="percent">Percent of outlier to simulate</param>
   /// <param name="axis">Axis to get outliers</param>
   /// <returns>Table of outliers</returns>
   public DataTable GetOulier(int percent, OutlierAxis axis )
   {
      if (percent == 0) return source.Data.Copy();
```

```
this.axis = axis;
this.percent = percent;
DataTable dt = source.Data.Copy();

// calculate count of data
int n = (int)(percent * 0.01 * dt.Rows.Count);

// calculate outlier
Quatile q = new Quatile(source.Data);

int ax = 0;
int count = 0;
// ถ้ากำหนด outliers บนแกน X จะคำนวณเพียงคอลัมน์แรกเท่านั้น
if (axis == OutlierAxis.X)
{
   count = 1;
}
else
{
   // คอลัมน์เริ่มต้น
   ax = 1;
   count = source.Data.Columns.Count;
}
Random rand = new Random();
ArrayList arr = new ArrayList();
for (int columnIndex = ax ; columnIndex < count; columnIndex++)
{
   arr.Clear();

   q.Column = columnIndex;
   for (int i = 0; i < n; i++)
   {
      int item;
      do{
         item = rand.Next(source.Data.Rows.Count);
      } while (arr.Contains(item)) ;
      arr.Add(item);
      float d = (float)dt.DefaultView[item][columnIndex];
```

```
            if( d < 0 )
                d = d - (q.Quatile1 + (2 * (q.Quatile3 - q.Quatile1)));
            else
                d = d + (q.Quatile3 + (2 * (q.Quatile3 - q.Quatile1)));
            dt.DefaultView[item][columnIndex] = d;
        }
    }

    //dt.DefaultView.Sort = "";
    DataTable dtr = dt.DefaultView.ToTable();
    dt.Dispose();
    return dtr;
}

/// <summary>
/// บันทึกตัวอย่างที่จำลองการเกิด outliers แล้ว
/// </summary>
/// <param name="filename">ชื่อและที่อยู่ของไฟล์ที่จะทำการบันทึก</param>
/// <param name="data">ชุดตัวอย่างที่จะบันทึก</param>
public void SaveXml(string filename, DataTable data)
{
    StreamWriter w = null;
    try
    {
        w = new StreamWriter(filename);
        string header = "<?xml version=\"1.0\"?>"
                    + "\r\n<Workbook xmlns=\"urn:schemas-microsoft-com:office:spreadsheet\""
                    + "\r\nxmlns:o=\"urn:schemas-microsoft-com:office:office\""
                    + "\r\nxmlns:x=\"urn:schemas-microsoft-com:office:excel\""
                    + "\r\nxmlns:ss=\"urn:schemas-microsoft-com:office:spreadsheet\""
                    + "\r\nxmlns:html=\"http://www.w3.org/TR/REC-html40\">";
        w.WriteLine(header);
        w.WriteLine(" <Worksheet ss:Name=\"Outliers\">");
        w.WriteLine("  <Table>");

        w.WriteLine("   <Row>");
        for (int c = 0; c < data.Columns.Count; c++)
        {
```

```
                w.WriteLine("    <Cell><Data ss:Type=\"String\">" + data.Columns[c] +
"</Data></Cell>");
            }
            w.WriteLine("  </Row>");

            for (int r = 0; r < data.Rows.Count; r++)
            {
                w.WriteLine("  <Row>");
                for (int c = 0; c < data.Columns.Count; c++)
                    w.WriteLine("    <Cell><Data ss:Type=\"Number\">" + data.Rows[r][c] +
"</Data></Cell>");
                w.WriteLine("  </Row>");
            }
            w.WriteLine("  </Table>");
            w.WriteLine(" </Worksheet>");
            w.WriteLine("</Workbook>");
        }
        catch (Exception ex)
        {
            throw ex;
        }
        finally
        {
            if (w != null)
                w.Close();
        }
    }
  }
}
```

```
//ตัวประมาณค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สัน
// FILENAME : PearsonT.cs

using System;
using System.Data;
namespace RCC
{
   /// <summary>

   /// ตัวประมาณค่าความสัมพันธ์แบบเพียร์สัน
   /// </summary>
   class PearsonT
   {
      private RandomData source = null;
      private float [ ] rxy = null;
      private string[ ] srxy = null;
      private int percent = -1;
      private string [ ] t = null;
      private float xMean = 0;
      private OutlierAxis axis = OutlierAxis.Y;
      /// <summary>
      /// สร้าง instace สำหรับตัวประมาณค่านี้
      /// </summary>
      /// <param name="source">ตัวอย่างที่จะทำการประมาณค่า</param>
      public PearsonT(RandomData source)
      {
         this.source = source;
      }
      /// <summary>
      /// คำนวณค่าความสัมพันธ์ของตัวอย่าง
      /// </summary>
      /// <param name="percent">ร้อยละของตัวอย่างที่จะจำลองการเกิด outliers </param>
      /// <returns>ค่าความสัมพันธ์ของตัวอย่างที่ประมาณได้</returns>
      public string[] GetSRxy(int percent)
      {
         GetRxy(percent);
         return this.srxy;
      }
```

```
/// <summary>
/// คำนวณค่าความสัมพันธ์ของตัวอย่าง
/// </summary>
/// <param name="percent">ร้อยละของตัวอย่างที่จะจำลองการเกิด outliers </param>
/// <returns>ค่าความสัมพันธ์ของตัวอย่างที่ประมาณได้</returns>
public float[ ] GetRxy(int percent)
{
   if (percent == this.percent && this.rxy != null)
      return this.rxy;
   this.rxy = new float[source.RhoConstants.Length];
   this.srxy = new string[source.RhoConstants.Length];
   this.percent = percent;
   float s = 0;
   float sx = 0, sy = 0;
   Statistic outlier = new Statistic( source.Outlier.GetOulier(percent, axis) );
   this.xMean = outlier.Mean(0);
   for (int column = 1; column <= source.RhoConstants.Length; column++)
   {
      float yMean = outlier.Mean(column);
      s = sx = sy = 0;
      for (int i = 0; i < source.Data.Rows.Count; i++)
      {
         float xi = (float)outlier.Source.Rows[i][0];
         float yi = (float)outlier.Source.Rows[i][column];
         s  += (xi - xMean) * (yi - yMean);
         sx += (xi - xMean) * (xi - xMean);
         sy += (yi - yMean) * (yi - yMean);
      }

      this.rxy[column-1] = s / (float)Math.Sqrt(sx * sy);
      this.srxy[column - 1] = this.rxy[column - 1].ToString();
   }
   return this.rxy;
}
```

```
        /// <summary>
        /// คำนวณค่าความสัมพันธ์ของตัวอย่าง ( ค่า T )
        /// </summary>
        /// <param name="percent">ร้อยละของตัวอย่างที่จะจำลองการเกิด outliers</param>
        /// <returns>ค่าความสัมพันธ์ของตัวอย่างที่ประมาณได้ ( ค่า T )</returns>
        public string[ ] GetT(int percent)
        {
            if (percent == this.percent && this.rxy != null)
                return this.t;
            this.t = new string[source.RhoConstants.Length];
            GetRxy(percent);
            for (int i = 0; i < t.Length; i++)
            {
                float sqrt  = (float)Math.Sqrt(source.Data.Rows.Count - 2);
                float sqrtx = (float)Math.Sqrt(1 - (rxy[i] * rxy[i]));
                t[i] = ((rxy[i] * sqrt) / sqrtx) + "";
            }
            return t;
        }
        /// <summary>
        /// เลือกแกนที่จำลองการเกิด outliers
        /// </summary>
        public OutlierAxis SelectedAxis
        {
            get { return this.axis; }
            set { this.axis = value; this.rxy = null; }
        }

    }
}
```

```
//การจัดอันดับข้อมูล
// FILENAME : Rankable.cs

using System;
using System.Data;
namespace RCC
{
   /// <summary>
   /// คลาสที่สามารถสร้างลำดับของข้อมูลได้
   /// </summary>
   abstract class Rankable
   {
      /// <summary>
      /// คำนวณค่าลำดับของข้อมูล
      /// </summary>
      /// <param name="source">ตารางข้อมูลที่จะทำการคำนวณลำดับ</param>
      /// <returns>ตารางลำดับของข้อมูล</returns>
      protected DataTable GetRankTable(DataTable source)
      {
         DataTable dtr = source.Copy();
         int n = dtr.Rows.Count;

         for (int column = 0; column < dtr.Columns.Count; column++)
         {
            dtr.DefaultView.Sort = dtr.Columns[column].ColumnName + " ASC";

            dtr = dtr.DefaultView.ToTable();
            for (int i = 0; i < n; i++)
            {
               dtr.Rows[i][column] = i + 1;
            }
         }
         //dtr.DefaultView.Sort = "";
         return dtr;
      }

   }
}
```

```
//สัมประสิทธิ์สหสัมพันธ์แบบสเปียร์แมน
// FILENAME : SpearmanT.cs

using System;
using System.Data;
namespace RCC
{
   /// <summary>
   /// ตัวประมาณค่าความสัมพันธ์แบบ Spearman
   /// </summary>
   class SpearmanT : Rankable
   {
      private RandomData source = null;
      private int percent  = -1;
      private float [] rs  = null;
      private string[] t   = null;
      private string[] srs = null;
      private OutlierAxis axis = OutlierAxis.Y;

      /// <summary>
      /// สร้าง instance สำหรับตัวประมาณค่านี้
      /// </summary>
      /// <param name="source"></param>
      public SpearmanT(RandomData source)
      {
         this.source = source;
      }

      /// <summary>
      /// คำนวณค่าความสัมพันธ์
      /// </summary>
      /// <param name="percent">ร้อยละที่จะจำลองการเกิด outlier</param>
      /// <returns>ค่าความสัมพันธ์ที่คำนวณได้</returns>
      public string[] GetSRs(int percent)
      {
         GetRs(percent);
         return srs;
      }
```

```
/// <summary>
/// คำนวณค่าความสัมพันธ์
/// </summary>
/// <param name="percent">ร้อยละที่จะจำลองการเกิด outlier</param>
/// <returns>ค่าความสัมพันธ์ที่คำนวน</returns>
public float [] GetRs(int percent)
{
    if (this.percent == percent && rs != null)
        return rs;

    this.percent = percent;

    rs = new float[this.source.RhoConstants.Length];
    srs = new string[this.source.RhoConstants.Length];
    float s = 0;

    DataTable dtr = this.source.Outlier.GetOulier(percent, axis);
    int n = dtr.Rows.Count;
    dtr = this.GetRankTable( dtr );
    dtr.DefaultView.Sort = dtr.Columns[0].ColumnName + " ASC";

    for (int column = 1; column <= rs.Length; column++)
    {
        s = 0;
        for (int i = 0; i < n; i++)
        {
            float d = (float)dtr.DefaultView[i][column] - (float)dtr.DefaultView[i][0];
            d = d * d;
            s += d;
        }

        rs[column - 1] = 1 - ((6 * s) / (n * (n * n - 1)));
        srs[column - 1] = rs[column - 1].ToString();
    }
    dtr.Dispose();
    return rs;
}
/// <summary>
```

```
        /// ค่า Degree of freedom สำหรับตัวอย่างที่กำลังทำการประมาณค่านี้
        /// </summary>
        public int DegreeOfFreedom
        {
            get { return this.source.Data.Rows.Count - 2; }
        }
        /// <summary>
        /// คำนวณค่าความสัมพันธ์และแปลงเป็นค่า T
        /// </summary>
        /// <param name="percent">ร้อยละที่จะจำลองการเกิด outlier</param>
        /// <returns>ค่าความสัมพันธ์ที่คำนวนได้เมื่อแปลงเป็นค่า T</returns>
        public string [] GetT(int percent)
        {
            if (this.percent == percent && rs != null)
                return this.t;
            GetRs(percent);
            this.t  = new string[rs.Length];
            for (int column = 0; column < this.source.RhoConstants.Length; column++)
            {
                float sqrt = rs[column] * (float)Math.Sqrt(DegreeOfFreedom);
                float sqrtx = (float)Math.Sqrt(1 - (rs[column] * rs[column]));
                t[column]  = (sqrt / sqrtx) + "";
            }

            return t;
        }

        /// <summary>
        /// กำหนดแกนที่จะจำลองการเกิด outlier
        /// </summary>
        public OutlierAxis SelectedAxis
        {
            get { return this.axis; }
            set { this.axis = value; this.rs = null; }
        }
    }
}
```

```
//สัมประสิทธิ์สหสัมพันธ์แบบเคนดอลล์
// FILENAME : KendallT.cs

using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
namespace RCC
{
    /// <summary>
    /// ตัวประมาณค่าความสัมพันธ์แบบ Kendall
    /// </summary>
    class KendallT : Rankable
    {
        private RandomData source = null;       // ตัวอย่างที่จะทำการประมาณค่า
        private Outlier outlier = null;         // ตัวอย่างที่มี outliers
        private DataView dv = null;
        private float[] tau = null;             // ชุดของค่า Tau-a ที่ประมาณค่าแล้ว
        private int percent = -1;               // ค่าร้อยละที่จะจำลองการเกิด outliers
        private string[] z = null;              // ค่าที่ประมาณแล้ว
        private string[] stau = null;           // ค่า Tau-a ที่ประมาณความสัมพันธ์แล้ว ( string )
        private OutlierAxis axis = OutlierAxis.Y;  // แกนที่จะจำลองการเกิด outliers

        /// <summary>
        /// สร้าง instace ของตัวประมาณค่าความสัมพันธ์แบบ Kendall
        /// </summary>
        /// <param name="source">ตัวอย่างที่จะทำการประมาณค่า</param>
        public KendallT(RandomData source)
        {
            this.source = source;
            this.outlier = source.Outlier;
        }

        /// <summary>
        /// คำนวณค่าความสัมพันธ์
        /// </summary>
        /// <param name="percent">ร้อยละที่จะจำลองการเกิด outliers</param>
        /// <returns>ค่าความสัมพันธ์ที่ประมาณได้</returns>
```

```
public string[] GetSTau(int percent)
{
   GetTau(percent);
   return this.stau;
}

/// <summary>
/// คำนวณค่าความสัมพันธ์
/// </summary>
/// <param name="percent">ร้อยละที่จะจำลองการเกิด outliers</param>
/// <returns>ค่าความสัมพันธ์ที่ประมาณได้</returns>
public float[] GetTau(int percent)
{
   this.percent = percent;

   this.tau = new float[source.RhoConstants.Length];
   this.stau = new string[source.RhoConstants.Length];
   this.dv = new DataView(GetRankTable(outlier.GetOulier(percent, axis)));

   dv.Sort = dv.Table.Columns[0].ColumnName + " ASC";

   int p = 0, q = 0;
   int n = dv.Count;
   for (int column = 1; column <= source.RhoConstants.Length; column++)
   {
      p = q = 0;
      for (int i = 0; i < n; i++)
      {
         int yi = dv.Find( dv[i][column] )+1;
         for (int j = i + 1; j < n; j++)
         {
            if ((float)dv[j][column] > yi)
               p++;
            else if ((float)dv[j][column] < yi)
               q++;
         }
      }
```

```
            tau[column-1] = (2.0f * (p - q)) / (n * (n - 1));
            stau[column - 1] = tau[column - 1].ToString();
        }
        dv.Dispose();
        return tau;
    }

    /// <summary>
    /// คำนวณค่าความสัมพันธ์ ( แปลงเป็นค่า Z )
    /// </summary>
    /// <param name="percent">ร้อยละที่จะจำลองการเกิด outliers</param>
    /// <returns>ค่าความสัมพันธ์ที่ประมาณได้ (แปลงเป็นค่า Z)</returns>
    public string[] GetZ( int percent )
    {
        GetTau(percent);
        int n = source.Data.Rows.Count;
        this.z = new string[tau.Length];

        for (int i = 0; i < z.Length; i++ )
         z[i] = "" + (float)(tau[i] * Math.Sqrt(9 * n * (n - 1))) / (float)Math.Sqrt(2 * (2 * n + 5));

        return z;
    }

    /// <summary>
    /// เลือกแกนที่จะจำลองการเกิด outliers
    /// </summary>
    public OutlierAxis SelectedAxis
    {
        get { return this.axis; }
        set { this.axis = value; this.tau = null; }
    }

  }
}
```

//สัมประสิทธิ์สหสัมพันธ์แบบถ่วงน้ำหนัก

```
// FILENAME: Biweight.cs

using System;
using System.Data;
namespace RCC
{
   /// <summary>
   /// ตัวประมาณค่าความสัมพันธ์แบบถ่วงน้ำหนัก
   /// </summary>
   class Biweight
   {
      private RandomData source = null;
      private int percent = -1;
      private float[] rb = null;
      private string[] t = null;
      private string[] srb = null;
      private OutlierAxis axis = OutlierAxis.Y;

      /// <summary>
      /// สร้าง instance สำหรับตัวประมาณค่านี้
      /// </summary>
      /// <param name="source">ตัวอย่างที่จะทำการประมาณค่า</param>
      public Biweight(RandomData source)
      {
         this.source = source;
      }

      /// <summary>
      /// คำนวณค่าความสัมพันธ์จากตัวอย่าง
      /// </summary>
      /// <param name="percent">ร้อยละที่จะจำลองการเกิด outliers </param>
      /// <returns>ค่าความสัมพันธ์ที่ประมาณได้</returns>
      public string[] GetSRb(int percent)
      {
         GetRb(percent);
         return this.srb;
      }
```

```
/// <summary>
/// คำนวณค่าความสัมพันธ์จากตัวอย่าง
/// </summary>
/// <param name="percent">ร้อยละที่จะจำลองการเกิด outliers </param>
/// <returns>ค่าความสัมพันธ์ที่ประมาณได้</returns>
public float[] GetRb(int percent)
{
   if (this.percent == percent && this.rb != null )
      return this.rb;
   this.percent = percent;

   this.rb = new float[source.RhoConstants.Length];
   this.srb = new string[source.RhoConstants.Length];
   DataTable outlier = source.Outlier.GetOulier(percent, axis);
   Quatile q = new Quatile(outlier);

   int n = this.source.Data.Rows.Count;

   for (int column = 1; column <= source.RhoConstants.Length; column++)
   {
      q.Column = 0;
      float madX = MadX(q);
      float mx = q.Quatile2;

      q.Column = column;
      float madY = MadY(q);
      float my = q.Quatile2;
      float ui = 0;
      float vi = 0;
      float sxx = 0;
      float syy = 0;
      float Sbxy = 0;
      float Sbxx = 0;
      float Sbyy = 0;

      for (int i = 0; i < n; i++)
      {
         float xi = (float)outlier.Rows[i][0];
```

```
            float yi = (float)outlier.Rows[i][column];
            float x = 0;
            float y = 0;

            ui = (xi - mx) / (9 * madX);
            vi = (yi - my) / (9 * madY);

            float ai = (-1 <= ui && ui <= 1) ? 1 : 0;
            float bi = (-1 <= vi && vi <= 1) ? 1 : 0;
            float sx = 1 - (ui * ui);
            float sy = 1 - (vi * vi);

            x = ai * (xi - mx) * (sx * sx);
            y = bi * (yi - my) * (sy * sy);

            Sbxy += (x * y);
            Sbxx += (x * x);
            Sbyy += (y * y);
            sxx += ai * sx * (1 - (5 * (ui * ui)));
            syy += bi * sy * (1 - (5 * (vi * vi)));
        }

        Sbxy = (n * Sbxy) / (sxx * syy);
        Sbxx = (n * Sbxx) / (sxx * sxx);
        Sbyy = (n * Sbyy) / (syy * syy);

        this.rb[column - 1] = Sbxy / (float)Math.Sqrt(Sbxx * Sbyy);
        this.srb[column - 1] = this.rb[column - 1].ToString();
    }

    return this.rb;
}

/// <summary>
/// ค่า MAD บนแกน X สำหรับการคำนวณค่าความสัมพันธ์
/// </summary>
/// <param name="q">ตัวคำนวณค่า quatile สำหรับตัวอย่างนี้</param>
/// <returns></returns>
```

```
private float MadX(Quatile q)
{
   int n = this.source.Data.Rows.Count;
   float meX = q.Quatile2;
   DataTable data = new DataTable();
   data.Columns.Add("x", typeof(float));

   for (int i = 0; i < n; i++)
   {
      data.Rows.Add(Math.Abs((float)source.Data.Rows[i][0] - meX));
   }

   return new Quatile(data).Quatile2;
}

/// <summary>
/// ค่า MAD บนแกน Y สำหรับการคำนวณค่าความสัมพันธ์
/// </summary>
/// <param name="q">ตัวคำนวณค่า quatile สำหรับตัวอย่างนี้</param>
/// <returns></returns>
private float MadY(Quatile q)
{
   int n = this.source.Data.Rows.Count;
   float meY = q.Quatile2;
   DataTable data = new DataTable();
   data.Columns.Add("y", typeof(float));

   for (int i = 0; i < n; i++)
   {
      data.Rows.Add( Math.Abs((float)source.Data.Rows[i][q.Column] - meY) );
   }

   return new Quatile(data).Quatile2;
}

/// <summary>
/// ค่า Degree of Freedom สำหรับตัวอย่างนี้
/// </summary>
```

```
        public int DegreeOfFreedom
        {
            get { return this.source.Data.Rows.Count - 2; }
        }

        /// <summary>
        /// คำนวณค่าความสัมพันธ์และทำการแปลงเป็นค่า T
        /// </summary>
        /// <param name="percent">ร้อยละของตัวอย่างที่จะจำลองการเกิด outliers </param>
        /// <returns>ค่า T ของค่าความสัมพันธ์ที่ประมาณได้</returns>
        public string[] GetT(int percent)
        {
            if (this.percent == percent && this.rb != null)
                return this.t;

            GetRb(percent);
            this.t = new string[rb.Length];

            for (int column = 0; column < this.source.RhoConstants.Length; column++)
            {
                float sqrt = rb[column] * (float)Math.Sqrt(DegreeOfFreedom);
                float sqrtx = (float)Math.Sqrt(1 - (rb[column] * rb[column]));
                t[column] = (sqrt / sqrtx) + "";
            }

            return t;
        }
        /// <summary>
        /// เลือกแกนที่จะจำลองการเกิด outliers
        /// </summary>
        public OutlierAxis SelectedAxis
        {
            get { return this.axis; }
            set { this.axis = value; this.rb = null; }
        }
    }
}
```

```
// Type I Error
// FILENAME : TypeIError.cs

using System;
using System.Data;
namespace RCC
{
    /// <summary>
    /// คลาสสำหรับการคำนวณหาค่าความผิดพลาดประเภทที่ 1
    /// </summary>
    class TypeIError
    {
        private DataSet[] resultSet = null;
        private Controler controler = null;
        private DataTable dt = null;

        /// <summary>
        /// สร้าง instance สำหรับคลาสนี้
        /// </summary>
        /// <param name="controler">controler</param>
        public TypeIError(Controler controler)
        {
            this.controler = controler;
            this.resultSet = controler.ResultSet;
            dt = new DataTable("Type I Error");
            dt.Columns.Add("Outlier", typeof(string));
            dt.Columns.Add("n", typeof(float));
            dt.Columns.Add("Rxy", typeof(float));
            dt.Columns.Add("Rs", typeof(float));
            dt.Columns.Add("Tau-a", typeof(float));
            dt.Columns.Add("Rb", typeof(float));

            for (int i = 0; i < controler.PercentOutliers.Length; i++)
            {
                for (int set = 0; set < resultSet.Length; set++)
                {
                    DataView psT =
```

```
                new DataView(resultSet[set].Tables["Pearson T " + controler.PercentOutliers[i] + "%
outliers"]);
            DataView spT =
                new DataView(resultSet[set].Tables["Spearman T " + controler.PercentOutliers[i] +
"% outliers"]);
            DataView kdZ =
                new DataView(resultSet[set].Tables["Kendall Z " + controler.PercentOutliers[i] + "%
outliers"]);
            DataView biT =
                new DataView(resultSet[set].Tables["Biweight T " + controler.PercentOutliers[i] + "%
outliers"]);

            string[] row = new string[6];

            row[0] = controler.PercentOutliers[i].ToString();
            row[1] = controler.SampleSet[set] + "";
            float t = t_value(controler.SampleSet[set]);
            float z = Properties.Settings.Default.z;
            float n = controler.TestRound;

            string cName = psT.Table.Columns[0].ColumnName;
            psT.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
            row[2] = psT.Count / n + "";
            psT.RowFilter = "";

            cName = spT.Table.Columns[0].ColumnName;
            spT.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
            row[3] = spT.Count / n + "";
            spT.RowFilter = "";

            cName = kdZ.Table.Columns[0].ColumnName;
            kdZ.RowFilter = "[" + cName + "] > " + z + " OR [" + cName + "] < -" + t;
            row[4] = kdZ.Count / n + "";
            kdZ.RowFilter = "";

            cName = biT.Table.Columns[0].ColumnName;
            biT.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
            row[5] = biT.Count / n + "";
```

```
                biT.RowFilter = "";

                dt.Rows.Add(row);
            }
        }
    }

    /// <summary>
    /// ค่าวิกฤติ
    /// </summary>
    /// <param name="count">ขนาดตัวอย่าง</param>
    /// <returns></returns>
    private float t_value(int count)
    {
        switch (count)
        {
            case 20: return 2.101f;
            case 50: return 2.011f;
            case 100:return 1.984f;
            default: return 1.984f;
        }
    }

    /// <summary>
    /// ค่าความถี่สัมพัทธ์ที่คำนวณได้
    /// </summary>
    public DataTable RelativeFrequency

    {
        get { return this.dt; }
    }
  }
}
```

```
//Power Of Test
// FILENAME : PowerOfTest.cs

using System;
using System.Data;

namespace RCC
{
   /// <summary>
   /// อำนาจการทดสอบ
   /// </summary>
   class PowerOfTest
   {
      private Controler control = null;
      private DataSet[] resultSet = null;
      private DataSet frq = null;

      /// <param name="resultSet">ชุดของค่าความสัมพันธ์ที่ประมาณค่าได้</param>
      public PowerOfTest(DataSet [] resultSet )
      {
         this.control = Controler.Instance;
         this.resultSet = resultSet;
         this.frq = new DataSet("Power of test");
         DataTable dt = new DataTable("Power of test");
         dt.Columns.Add("Outliers", typeof(int));
         dt.Columns.Add("n", typeof(int));
         dt.Columns.Add("Rho", typeof(float));
         dt.Columns.Add("Rxy", typeof(float));
         dt.Columns.Add("Rs", typeof(float));
         dt.Columns.Add("Tau a", typeof(float));
         dt.Columns.Add("Rb", typeof(float));

         frq.Tables.Add(dt);
         for (int i = 0; i < control.PercentOutliers.Length; i++)
         {
            for (int set = 0; set < resultSet.Length; set++)
            {
               for (int rho = 0; rho < control.RhoConstants.Length; rho++)
```

```
{
    DataView psT =
        new DataView(resultSet[set].Tables["Pearson T " + control.PercentOutliers[i] + "%
outliers"]);
    DataView spT =
        new DataView(resultSet[set].Tables["Spearman T " + control.PercentOutliers[i] +
"% outliers"]);
    DataView kdZ =
        new DataView(resultSet[set].Tables["Kendall Z " + control.PercentOutliers[i] + "%
outliers"]);
    DataView biT =
        new DataView(resultSet[set].Tables["Biweight T " + control.PercentOutliers[i] +
"% outliers"]);

    string[] row = new string[7];
    row[0] = control.PercentOutliers[i] + "";
    row[1] = resultSet[set].DataSetName;
    float t = t_value(control.SampleSet[set]);
    float z = Properties.Settings.Default.z;
    float n = control.TestRound;

    row[2] = control.RhoConstants[rho].ToString();

    string cName = psT.Table.Columns[rho].ColumnName;
    psT.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
    row[3] = (psT.Count / n) + "";

    cName = spT.Table.Columns[rho].ColumnName;
    spT.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
    row[4] = (spT.Count / n) + "";

    cName = kdZ.Table.Columns[rho].ColumnName;
    kdZ.RowFilter = "[" + cName + "] > " + z + " OR [" + cName + "] < -" + t;
    row[5] = (kdZ.Count / n) + "";

    cName = biT.Table.Columns[rho].ColumnName;
    biT.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
    row[6] = (biT.Count / n) + "";
```

```
                frq.Tables[0].Rows.Add(row);
            }// end for
        }// end for
    }// end for
}// end contructor

/// <summary>
/// ค่าวิกฤติ
/// </summary>
/// <param name="count">ขนาดตัวอย่าง</param>
/// <returns></returns>
private float t_value(int count)
{
    switch (count)
    {
        case 20: return 2.101f;
        case 50: return 2.011f;
        case 100: return 1.984f;
        default: return 1.984f;
    }
}

/// <summary>
/// ค่าความถี่สัมพัทธ์ที่คำนวณได้
/// </summary>
public DataSet RelativeFrequency
{
    get { return this.frq; }
}
}// end class
}// end  RCC
```

```
// Relative Efficiency
// FILENAME : RelativeEfficiency.cs

using System;
using System.Data;
namespace RCC
{
   /// <summary>
   /// ตัวคำนวณค่าประสิทธิภาพสัมพัทธ์
   /// </summary>
   class RelativeEfficiency
   {
      private DataSet re = null;
      private DataSet[] result = null;
      private string[,] estimater = { { "Rxy", "Rs", "Tau-a", "Rb" }, { "Pearson T ", "Spearman T ",
"Kendall Z ", "Biweight T " } };

      /// <summary>
      /// สร้าง instance สำหรับการคำนวณค่าประสิทธิภาพสัมพัทธ์
      /// </summary>
      /// <param name="control">controler</param>
      public RelativeEfficiency(Controler control)
      {
         result = control.ResultSet;

         this.re = new DataSet();

         for (int rho = 1; rho < control.RhoConstants.Length; rho++)
         {
            DataTable dt = new DataTable("Relative Efficiency Rho = " + control.RhoConstants[rho]);
            dt.Columns.Add("n", typeof(int));
            dt.Columns.Add("Estimater", typeof(string));
            foreach( int outlier in control.PercentOutliers )
               dt.Columns.Add( outlier + "%", typeof(float));

            for (int n = 0; n < control.SampleSet.Length; n++)
            {
               float t = t_value(control.SampleSet[n]);
```

```
                for (int es = 0; es < estimater.GetLength(1); es++)
                {
                    string[] row = new string[control.PercentOutliers.Length + 1];
                    row[0] = control.SampleSet[n] + "";
                    row[1] = estimater[0, es];

                    DataView table0 =
                        new DataView(control.ResultSet[n].Tables[estimater[1, es] + "0% outliers"]);

                    if (estimater[0, es] == "Tau-a") t = Properties.Settings.Default.z;

                    string cName = table0.Table.Columns[rho].ColumnName;
                    table0.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;

                    for (int ol = 1; ol < control.PercentOutliers.Length; ol++)
                    {
                        DataView table =
                            new DataView(control.ResultSet[n].Tables[estimater[1, es] +
control.PercentOutliers[ol] + "% outliers"]);

                        table.RowFilter = "[" + cName + "] > " + t + " OR [" + cName + "] < -" + t;
                        row[ol + 1] = ((float)table.Count / table0.Count) + "";

                        table.Dispose();
                        table = null;
                    }
                    table0.Dispose();
                    dt.Rows.Add(row);
                }
            }

            re.Tables.Add(dt);
        }
    }
```

```
        /// <summary>
        /// ค่าวิกฤติ
        /// </summary>

        /// <param name="count">จำนวนตัวอย่าง</param>
        private float t_value(int count)
        {
            switch (count)
            {
                case 20: return 2.101f;
                case 50: return 2.011f;
                case 100: return 1.984f;
                default: return 1.984f;
            }
        }

        /// <summary>
        /// ค่าประสิทธิภาพสัมพัทธ์ที่คำนวณได้
        /// </summary>
        public DataSet RE
        {
            get { return this.re; }
        }

    }
}
```

// MeanSquareError

```
// FILENAME: MeanSquareError.cs

using System;
using System.Data;
namespace RCC
{
    /// <summary>
    /// ตัวแปลความหมายแบบ ค่าเฉลี่ยความคลาดเคลื่อนกำลังสอง
    /// </summary>
    class MeanSquareError
```

```
{
    private Controler control   = null;
    private DataSet[] result    = null;
    private DataSet   mse       = new DataSet("Mean Square Error");

    /// <summary>
    /// สร้าง instance สำหรับตัวแปลความหมายนี้
    /// </summary>
    /// <param name="control">controler</param>
    public MeanSquareError(Controler control)
    {
        this.control = control;
        this.result  = control.ResultSet;

        for (int rho = 0; rho < control.RhoConstants.Length; rho++)
        {
            DataTable dt = new DataTable("MSE Rho=" + control.RhoConstants[rho]);
            dt.Columns.Add("n", typeof(int));
            dt.Columns.Add("Outliers", typeof(string));
            dt.Columns.Add("Rxy", typeof(float));
            dt.Columns.Add("Rs", typeof(float));
            dt.Columns.Add("Tau-a", typeof(float));
            dt.Columns.Add("Rb", typeof(float));

            for (int n = 0; n < control.SampleSet.Length; n++)
            {
                DataTable PsNoOutlier = result[n].Tables["Pearson R 0% outliers"];

                for (int ol = 1; ol < control.PercentOutliers.Length; ol++)
                {
                    DataTable Ps = result[n].Tables["Pearson R " + control.PercentOutliers[ol] + "%
outliers"];
                    DataTable Sp = result[n].Tables["Spearman R " + control.PercentOutliers[ol] + "%
outliers"];
                    DataTable Kt = result[n].Tables["Kendall Tau " + control.PercentOutliers[ol] + "%
outliers"];
                    DataTable Bi = result[n].Tables["Biweight R " + control.PercentOutliers[ol] + "%
outliers"];
```

```
                string [] row = new string[6];
                row[0] = control.SampleSet[n].ToString();
                row[1] = control.PercentOutliers[ol].ToString();

                float psSum = 0;
                float spSum = 0;
                float ktSum = 0;
                float biSum = 0;
                for (int r = 0; r < control.TestRound; r++)
                {
                   float diff = ((float)Ps.Rows[r][rho]-(float)PsNoOutlier.Rows[r][rho] );
                   psSum += diff * diff;

                   diff = ((float)Sp.Rows[r][rho] - (float)PsNoOutlier.Rows[r][rho]);
                   spSum += diff * diff;

                   diff = ((float)Kt.Rows[r][rho] - (float)PsNoOutlier.Rows[r][rho]);
                   ktSum += diff * diff;

                   diff = ((float)Bi.Rows[r][rho] - (float)PsNoOutlier.Rows[r][rho]);
                   biSum += diff * diff;
                }

                row[2] = (psSum / control.TestRound) + "";
                row[3] = (spSum / control.TestRound) + "";
                row[4] = (ktSum / control.TestRound) + "";
                row[5] = (biSum / control.TestRound) + "";

                dt.Rows.Add( row );
              }// end for
            }// end for

            this.mse.Tables.Add( dt );
          }// end for
        }// end constructor

        /// <summary>
```

```
        /// ค่าที่ผ่านการแปลความหมายแล้ว
        /// </summary>
        public DataSet MSE
        {
            get { return this.mse; }
        }
    }// end mse
}

//การควบคุมการจำลองข้อมูลและการประมาณค่า และเก็บผลลัพธ์
// FILENAME: Controler.cs

using System;
using System.Data;
using System.Collections;
using System.IO;

namespace RCC
{
    /// <summary>
    /// ควบคุมการจำลองข้อมูลและการประมาณค่า  รวมทั้งเก็บผลลัพธ์ที่ได้
    /// </summary>
    class Controler
    {
        // event delegates
        public delegate void ProgessTickEventHandler( object sender, ProgressEventArgs e );
        public delegate void SimulationFinish();
        public delegate void CancelEventHandler();

        // events
        public event ProgessTickEventHandler ProgressTick;
        public event SimulationFinish Finish;
        public event CancelEventHandler Cancel;

        private int testRound   = 0;                    // จำนวนรอบที่ทำการจำลอง
        private int progress    = 0;                    // ความก้าวหน้าในการจำลอง
        private float[] rhoConstants = null;            // ชุดค่าความสัมพันธ์
        private int [] percentOutliers = null;          // ชุดของค่าร้อยละที่จะจำลองการเกิด Outlier
```

```
        private int [] setCount = null;                 // ชุดของตัวจำนวนตัวอย่าง
        private RandomData [] data = null;              // ชุดของตัวอย่างที่สุ่มขึ้นมา
        private bool cancel = false;                    // flag แสดงการทำงาน เมื่อเป็น true จะหลุดออกจาก
การจำลอง
        private DataSet[] resultSetX = null;            // ชุดค่าที่ประมาณแล้ว และเกิด Outlier บนแกน X
        private DataSet[] resultSetY = null;            // ชุดค่าที่ประมาณแล้ว และเกิด Outlier บนแกน Y
        private OutlierAxis selectedAxis = OutlierAxis.Y;  // กำลังทำงานกับผลการประมาณค่าที่เกิด
outlier บนแกนใด
        private int roundToSave = 0;                    // เลือกบันทึกข้อมูลสุ่มจากรอบที่กำหนด
        private string filename = "samples.xls";        // ชื่อแฟ้มที่จะบันทึกข้อมูลสุ่ม

        private static Controler instance = null;       // instance สำหรับ controler

        // private constructor
        private Controler(int testRound, float[] rhoConstants, int[] percentOutliers, int [] setCount)
        {
            this.rhoConstants = rhoConstants;
            this.percentOutliers = percentOutliers;
            this.testRound = testRound;
            this.setCount  = setCount;
            instance = this;
            Initialize();
        }
        // singleton constructor
        /// <summary>
        /// สร้าง instance และเตรียมการจำลองข้อมูล
        /// </summary>
        /// <param name="testRound">รอบที่จะทำการจำลอง</param>
        /// <param name="rhoConstants">ชุดของค่าความสัมพันธ์</param>
        /// <param name="percentOutliers">ชุดของร้อยละที่จะจำลองการเกิด outlier</param>
        /// <param name="sampleSet">ชุดของขนาดตัวอย่างที่จะจำลอง</param>
        /// <returns>instance ที่พร้อมสำหรับการเริ่มจำลอง</returns>
        public static Controler GetControler(int testRound, float[] rhoConstants, int[] percentOutliers, int[]
sampleSet)
        {
            instance = new Controler(testRound, rhoConstants, percentOutliers, sampleSet);
            return instance;
        }
```

```
/// <summary>
/// สร้างตารางสำหรับเก็บข้อมูล ต้องเรียกก่อนใช้ Simulate()เสมอ
/// </summary>
private void Initialize()
{
   this.data = new RandomData[setCount.Length];
   this.resultSetY = new DataSet[setCount.Length];
   this.resultSetX = new DataSet[setCount.Length];

   for( int i =0; i < this.setCount.Length; i++ )
   {
      this.resultSetY[i] = new DataSet(setCount[i] + "");
      this.resultSetX[i] = new DataSet(setCount[i] + "");
      for( int ic = 0; ic < percentOutliers.Length; ic++ )
      {
         // y axis ////////////////////////////////////////
         DataTable psR = new DataTable( "Pearson R " + percentOutliers[ic] + "% outliers" );
         DataTable psT = new DataTable( "Pearson T " + percentOutliers[ic] + "% outliers" );

         DataTable spR = new DataTable( "Spearman R " + percentOutliers[ic] + "% outliers" );
         DataTable spT = new DataTable( "Spearman T " + percentOutliers[ic] + "% outliers" );

         DataTable kdT = new DataTable( "Kendall Tau " + percentOutliers[ic] + "% outliers");
         DataTable kdZ = new DataTable( "Kendall Z " + percentOutliers[ic] + "% outliers" );

         DataTable biR = new DataTable( "Biweight R " + percentOutliers[ic] + "% outliers" );
         DataTable biT = new DataTable( "Biweight T " + percentOutliers[ic] + "% outliers" );
         for( int rho = 0; rho < rhoConstants.Length; rho++)
         {
            string col = "ρ = " + rhoConstants[rho];

            psR.Columns.Add(col, typeof(float));
            psT.Columns.Add(col, typeof(float));

            spR.Columns.Add(col, typeof(float));
            spT.Columns.Add(col, typeof(float));

            kdZ.Columns.Add(col, typeof(float));
```

```
    kdT.Columns.Add(col, typeof(float));

    biR.Columns.Add(col, typeof(float));
    biT.Columns.Add(col, typeof(float));
}

resultSetY[i].Tables.Add(psR);
resultSetY[i].Tables.Add(psT);

resultSetY[i].Tables.Add(spR);
resultSetY[i].Tables.Add(spT);

resultSetY[i].Tables.Add(kdZ);
resultSetY[i].Tables.Add(kdT);

resultSetY[i].Tables.Add(biR);
resultSetY[i].Tables.Add(biT);

// x axis ////////////////////////////////////////
psR = new DataTable("Pearson R " + percentOutliers[ic] + "% outliers");
psT = new DataTable("Pearson T " + percentOutliers[ic] + "% outliers");

spR = new DataTable("Spearman R " + percentOutliers[ic] + "% outliers");
spT = new DataTable("Spearman T " + percentOutliers[ic] + "% outliers");

kdT = new DataTable("Kendall Tau " + percentOutliers[ic] + "% outliers");
kdZ = new DataTable("Kendall Z " + percentOutliers[ic] + "% outliers");

biR = new DataTable("Biweight R " + percentOutliers[ic] + "% outliers");
biT = new DataTable("Biweight T " + percentOutliers[ic] + "% outliers");
for (int rho = 0; rho < rhoConstants.Length; rho++)
{
    string col = "ρ = " + rhoConstants[rho];

    psR.Columns.Add(col, typeof(float));
    psT.Columns.Add(col, typeof(float));

    spR.Columns.Add(col, typeof(float));
```

```
            spT.Columns.Add(col, typeof(float));

            kdZ.Columns.Add(col, typeof(float));
            kdT.Columns.Add(col, typeof(float));

            biR.Columns.Add(col, typeof(float));
            biT.Columns.Add(col, typeof(float));
        }

        resultSetX[i].Tables.Add(psR);
        resultSetX[i].Tables.Add(psT);

        resultSetX[i].Tables.Add(spR);
        resultSetX[i].Tables.Add(spT);

        resultSetX[i].Tables.Add(kdZ);
        resultSetX[i].Tables.Add(kdT);

        resultSetX[i].Tables.Add(biR);
        resultSetX[i].Tables.Add(biT);
      }
   }
}

/// <summary>
/// เริ่มจำลองข้อมูลและทำการประมาณค่า ต้องเรียก Initialize() ก่อนเสมอ
/// </summary>
public void Simulate()
{
   ProgressEventArgs e = null;
   for (int i = 0; i < this.testRound; i++)
   {
      if (this.cancel == true)
      {
         this.Clear();
         OnCancel();
         return;
      }
```

```
        progress = (int)(((float)i / testRound) * 100);
        e = new ProgressEventArgs( progress );
        e.StepDescription = "Simulating for " + testRound + " rounds...";
        OnProgessTick(e);
        GetRoundResult(e);

        if (this.roundToSave == i+1)
            SaveSample(filename, data);
        OnProgessTick(e);
    }
    e.Progress = 100;
    OnProgessTick(e);
    OnFinish();
}

/// <summary>
/// จำลองข้อมูลและทำการประมาณค่าสำหรับแต่ละรอบ
/// </summary>
/// <param name="e">ค่าความคืบหน้า</param>
private void GetRoundResult(ProgressEventArgs e)
{
    for (int i = 0; i < setCount.Length; i++)
    {
        this.data[i] = new RandomData( setCount[i], this.rhoConstants );
        this.data[i].SimulateData();

        SpearmanT sp = new SpearmanT( data[i] );
        PearsonT  pt = new PearsonT( data[i] );
        Biweight  bw = new Biweight( data[i] );
        KendallT  kt = new KendallT( data[i] );

        for (int p = 0; p < percentOutliers.Length; p++)
        {
            ////////////// จำลองการเกิด outlier บนแกน Y

            // ดึงเอา instance ของแต่ละตารางออกมา
            DataTable psr = resultSetY[i].Tables["Pearson R " + percentOutliers[p] + "% outliers"];
            DataTable pst = resultSetY[i].Tables["Pearson T " + percentOutliers[p] + "% outliers"];
```

```
DataTable spr = resultSetY[i].Tables["Spearman R " + percentOutliers[p] + "% outliers"];
DataTable spt = resultSetY[i].Tables["Spearman T " + percentOutliers[p] + "% outliers"];
DataTable kdt = resultSetY[i].Tables["Kendall Tau " + percentOutliers[p] + "% outliers"];
DataTable kdz = resultSetY[i].Tables["Kendall Z " + percentOutliers[p] + "% outliers"];
DataTable bir = resultSetY[i].Tables["Biweight R " + percentOutliers[p] + "% outliers"];
DataTable bit = resultSetY[i].Tables["Biweight T " + percentOutliers[p] + "% outliers"];


pt.SelectedAxis = sp.SelectedAxis = kt.SelectedAxis = bw.SelectedAxis = OutlierAxis.Y;
pst.Rows.Add(pt.GetT(percentOutliers[p]));
psr.Rows.Add(pt.GetSRxy(percentOutliers[p]));

// -- spearman
spt.Rows.Add(sp.GetT(percentOutliers[p]));
spr.Rows.Add(sp.GetSRs(percentOutliers[p]));

//- kendall
kdz.Rows.Add(kt.GetZ(percentOutliers[p]));
kdt.Rows.Add(kt.GetSTau(percentOutliers[p]));

//- biweight
bit.Rows.Add(bw.GetT(percentOutliers[p]));
bir.Rows.Add(bw.GetSRb(percentOutliers[p]));

OnProgessTick(e);
////////////// จำลองการเกิด outlier บนแกน X
// ดึงเอา instance ของแต่ละตารางออกมา
psr = resultSetX[i].Tables["Pearson R " + percentOutliers[p] + "% outliers"];
pst = resultSetX[i].Tables["Pearson T " + percentOutliers[p] + "% outliers"];
spr = resultSetX[i].Tables["Spearman R " + percentOutliers[p] + "% outliers"];
spt = resultSetX[i].Tables["Spearman T " + percentOutliers[p] + "% outliers"];
kdt = resultSetX[i].Tables["Kendall Tau " + percentOutliers[p] + "% outliers"];
kdz = resultSetX[i].Tables["Kendall Z " + percentOutliers[p] + "% outliers"];
bir = resultSetX[i].Tables["Biweight R " + percentOutliers[p] + "% outliers"];
bit = resultSetX[i].Tables["Biweight T " + percentOutliers[p] + "% outliers"];

pt.SelectedAxis = sp.SelectedAxis = kt.SelectedAxis = bw.SelectedAxis = OutlierAxis.X;
pst.Rows.Add(pt.GetT(percentOutliers[p]));
```

```
            psr.Rows.Add(pt.GetSRxy(percentOutliers[p]));

            // -- spearman
            spt.Rows.Add(sp.GetT(percentOutliers[p]));
            spr.Rows.Add(sp.GetSRs(percentOutliers[p]));

            //- kendall
            kdz.Rows.Add(kt.GetZ(percentOutliers[p]));
            kdt.Rows.Add(kt.GetSTau(percentOutliers[p]));

            //- biweight
            bit.Rows.Add(bw.GetT(percentOutliers[p]));
            bir.Rows.Add(bw.GetSRb(percentOutliers[p]));
        }// end for
        MCSRandom.SSeed += 2;
    }

}

/// <summary>
/// บันทึกตัวอย่างที่จำลองขึ้นมา
/// </summary>
/// <param name="filename">ชื่อและที่อยู่ที่จะทำการบันทึก</param>
/// <param name="data">ชุดข้อมูลที่จะบันทึก</param>
private void SaveSample(string filename, RandomData[] data)
{
    StreamWriter w = null;
    try
    {
        w = new StreamWriter(filename);
        string header = "<?xml version=\"1.0\"?>"
                    + "\r\n<Workbook xmlns=\"urn:schemas-microsoft-com:office:spreadsheet\""
                    + "\r\nxmlns:o=\"urn:schemas-microsoft-com:office:office\""
                    + "\r\nxmlns:x=\"urn:schemas-microsoft-com:office:excel\""
                    + "\r\nxmlns:ss=\"urn:schemas-microsoft-com:office:spreadsheet\""
                    + "\r\nxmlns:html=\"http://www.w3.org/TR/REC-html40\">";
        w.WriteLine(header);
```

```
            for (int i = 0; i < data.Length; i++)
            {
                w.WriteLine(" <Worksheet ss:Name=\"" + data[i].Data.TableName + "\">");
                w.WriteLine("  <Table>");

                w.WriteLine("   <Row>");
                for (int c = 0; c < data[i].Data.Columns.Count; c++)
                {
                    w.WriteLine("    <Cell><Data ss:Type=\"String\">" + data[i].Data.Columns[c] +
"</Data></Cell>");
                }
                w.WriteLine("   </Row>");

                for (int r = 0; r < data[i].Data.Rows.Count; r++)
                {
                    w.WriteLine("   <Row>");
                    for (int c = 0; c < data[i].Data.Columns.Count; c++)
                        w.WriteLine("    <Cell><Data ss:Type=\"String\">" + data[i].Data.Rows[r][c] +
"</Data></Cell>");
                    w.WriteLine("   </Row>");
                }
                w.WriteLine("  </Table>");
                w.WriteLine(" </Worksheet>");
            }
            w.WriteLine("</Workbook>");
        }//  end for
        catch (Exception ex)
        {
            throw ex;
        }
        finally
        {
            if (w != null)
                w.Close();
        }
    }

    /// <summary>
```

```
/// ยกเลิกการจำลอง
/// </summary>
public void CancelSimulation()
{
   this.cancel = true;
}

/// <summary>
/// ล้างข้อมูลและค่าต่าง ๆ ที่ตั้งไว้
/// </summary>
private void Clear()
{
   if (data == null)
      return;
   foreach (RandomData d in this.data)
   {
      if( d != null )
      d.Data.Dispose();
   }
   foreach (DataSet result in resultSetX)
   {
      if( result != null )
      result.Dispose();
   }
   foreach (DataSet result in resultSetY)
   {
      if( result != null )
      result.Dispose();
   }

   data = null;
   rhoConstants = null;
   percentOutliers = null;
}

/// <summary>
/// ล้างข้อมูลและค่าที่ตั้งไว้ รวมไปถึง Events handler ทั้งหมด
/// </summary>
```

```
        public void Reset()
        {
            Clear();

            this.ProgressTick = null;
            this.Finish = null;
            this.Cancel = null;
        }
#region Properties
        /// <summary>
        /// จำนวนรอบที่จะทำการจำลอง
        /// </summary>
        public int TestRound
        {
            get { return this.testRound; }
            set { this.testRound = value; }
        }
        /// <summary>
        /// instance ปัจจุบันของ controler
        /// </summary>
        public static Controler Instance
        {
            get { return instance; }
        }
        /// <summary>
        /// ตั้งค่ารอบที่จะบันทึกข้อมูลที่จำลองขึ้น
        /// </summary>
        public int RoundToSave
        {
            get { return this.roundToSave; }
            set { this.roundToSave = value; }
        }
        /// <summary>
        /// ชื่อและที่อยู่ของไฟล์ที่จะบันทึกข้อมูลที่จำลองขึ้น
        /// </summary>
        public string Filename
        {
            get { return this.filename; }
```

```
    set { this.filename = value; }
}

/// <summary>
/// ชุดของขนาดตัวอย่าง
/// </summary>
public int[] SampleSet
{
    get { return this.setCount; }
    set { this.setCount = value; }
}

/// <summary>
/// ชุดของร้อยละที่จะทำการจำลองการเกิด outliers ต้องมี 0 เป็นค่าตัวแรกเสมอ
/// </summary>
public int[] PercentOutliers
{
    get { return this.percentOutliers; }
}

/// <summary>
///  ชุดของค่าความสัมพันธ์  ต้องมี 0 เป็นค่าตัวแรกเสมอ
/// </summary>
public float[] RhoConstants
{
    get { return this.rhoConstants; }
    set { this.rhoConstants = value; }
}

/// <summary>
/// เลือกแกนของ outlier ที่กำลังแสดงผลอยู่
/// </summary>
public OutlierAxis SelectedAxis
{
    get { return this.selectedAxis; }
    set { this.selectedAxis = value; }
}
```

```
/// <summary>
/// ชุดข้อมูลที่ประมาณค่าแล้ว กำหนดแกนโดยการตั้งค่าที่ SelectedAxis
/// </summary>
public DataSet[] ResultSet
{
   get {
      if (this.selectedAxis == OutlierAxis.X)
         return this.resultSetX;
      return this.resultSetY;
   }
}

/// <summary>
/// ชุดข้อมูลที่ประมาณค่าแล้ว และจำลองการเกิด outlier บนแกน X
/// </summary>
public DataSet[] ResultSetX
{
   get { return this.resultSetX; }
}

/// <summary>
/// ชุดข้อมูลที่ประมาณค่าแล้ว และจำลองการเกิด outlier บนแกน X
/// </summary>
public DataSet[] ResultSetY
{
   get { return this.resultSetY; }
}

/// <summary>
/// เหตุการณ์เมื่อมีการยกเลิกการจำลอง
/// </summary>
protected virtual void OnCancel()
{
   if (Cancel != null)
      Cancel();
}

/// <summary>
```

```
        /// เหตุการณ์เมื่อการจำลองเสร็จสมบูรณ์
        /// </summary>
        protected virtual void OnFinish()
        {
            if (Finish != null && cancel == false)
                Finish();
        }

        /// <summary>
        /// เหตุการณ์เมื่อการจำลองมีความคืบหน้า
        /// </summary>
        /// <param name="e">ข้อมูลความคืบหน้า</param>
        protected virtual void OnProgessTick(ProgressEventArgs e)
        {
            if (ProgressTick != null)
                ProgressTick(this, e);
        }
        #endregion
    }
}

//แสดงความคืบหน้าในการคำนวณ
// FILENAME : ProgressEventArgs.cs

using System;

namespace RCC
{
    /// <summary>
    /// ความคืบหน้าการคำนวณ
    /// </summary>
    class ProgressEventArgs : System.EventArgs
    {
        private int progress = 0;
        private string description = "";
        public ProgressEventArgs( int progress )
        {
```

```
            this.progress = progress;
        }

        public ProgressEventArgs(int progress, string description)
        {
            this.progress = progress;
        }

        public string StepDescription
        {
            get { return this.description; }
            set { this.description = value; }
        }

        public int Progress {
            get { return progress; }
            set { this.progress = value; }
        }
    }
}

// FILENAME : Settings.cs

namespace RCC.Properties {


    // This class allows you to handle specific events on the settings class:
    //  The SettingChanging event is raised before a setting's value is changed.
    //  The PropertyChanged event is raised after a setting's value is changed.
    //  The SettingsLoaded event is raised after the setting values are loaded.
    //  The SettingsSaving event is raised before the setting values are saved.
    internal sealed partial class Settings {

        public Settings() {
            // // To add event handlers for saving and changing settings, uncomment the lines below:
            //
            // this.SettingChanging += this.SettingChangingEventHandler;
            //
```

```
        // this.SettingsSaving += this.SettingsSavingEventHandler;
        //
    }

    private void SettingChangingEventHandler(object sender,
System.Configuration.SettingChangingEventArgs e) {
        // Add code to handle the SettingChangingEvent event here.

    }

    private void SettingsSavingEventHandler(object sender,
System.ComponentModel.CancelEventArgs e) {
        // Add code to handle the SettingsSaving event here.
    }
  }
}

// FILENAME : ValueOutOfRange.cs

using System;

namespace RCC
{
   /// <summary>
   /// Exception เมื่อค่าที่ระบุไม่อยู่ในช่วงที่กำหนด
   /// </summary>
   class ValueOutOfRangeException : System.Exception
   {
      public ValueOutOfRangeException(string msg) : base( msg )
      {
      }
   }
}
```