LAMPIRAN

Lampiran 1. Hasil Tuning Parameter per Replikasi ($\rho = 0{,}2$)

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
1.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.833413	0.211644	3.756256
2.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.054262	0.136849	3.026150
3.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	2.806529	0.136849	2.779173
4.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.412574	0.136849	4.365436
5.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.487731	0.136849	4.446757
6.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	2.915068	0.142192	2.884985
7.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.445172	0.120822	3.423566
8.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.430135	0.126164	3.405400
9.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.102720	0.136849	4.064050
10.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.104204	0.163562	4.046337
11.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.010324	0.110137	2.995031

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
12.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.487318	0.318493	3.379245
13.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.452908	0.131507	4.416473
14.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.326072	0.206301	4.243766
15.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.246594	0.286438	4.091635
16.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.262591	0.136849	3.233098
17.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.049609	0.120822	5.016059
18.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.754747	0.195616	3.671129
19.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.175422	0.249041	5.043988
20.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.864705	0.120822	3.839972
21.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.888557	0.254384	3.785792
22.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.340137	0.136849	3.304734
23.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.084653	0.174247	3.031503
24.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.717329	0.136849	3.685329

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
25.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.022193	0.115479	3.001841
26.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.902265	0.147534	3.856918
27.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.373724	0.152877	3.332085
28.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.000579	0.136849	2.970212
29.	K = 4 *Knot* = {0, 0.3, 0.75}	3.644587	0.249041	3.593410
30.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	2.927238	0.104795	2.915377
31.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.218491	0.120822	3.197986
32.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.650037	0.126164	4.612812
33.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.697018	0.152877	4.641402
34.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	2.350193	0.115479	2.336623
35.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.550895	0.152877	4.491653
36.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.821640	0.136849	3.780081
37.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.630110	0.168904	3.577461
38.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.605348	0.136849	3.573129

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
39.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.104320	0.142192	3.073204
40.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.440483	0.227671	4.305572
41.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.359411	0.168904	3.307490
42.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.374779	0.200959	4.275435
43.	K = 3 *Knot* = {0.25, 0.5, 0.75}	2.776236	0.115479	2.760547
44.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.514068	0.152877	3.468108
45.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.309954	0.136849	4.264547
46.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.882398	0.227671	3.781448
47.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.064854	0.163562	3.022442
48.	K = 3 *Knot* = {0.25, 0.5, 0.75}	2.987229	0.168904	2.939514
49.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.694000	0.142192	3.655576
50.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.363829	0.131507	4.324775

Lampiran 2. Hasil Tuning Parameter per Replikasi ($\rho = 0{,}5$)

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
1.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.743457	0.110137	3.728363
2.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.509637	0.158219	3.461616
3.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.097590	0.174247	5.004549
4.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.690657	0.158219	3.639157
5.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	2.433702	0.131507	2.411191
6.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.419563	0.142192	3.384020
7.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.174771	0.329178	5.002486
8.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.048648	0.243699	4.907068
9.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.769935	0.152877	3.722200
10.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.380810	0.179589	4.311920
11.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.461246	0.136849	4.419437
12.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.054802	0.136849	3.025768

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
13.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.756038	0.131507	4.716840
14.	K = 3 *Knot* = {0.25, 0.5, 0.75}	2.899578	0.200959	2.841348
15.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.883157	0.195616	4.796314
16.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.289086	0.158219	3.244366
17.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.350492	0.126164	3.325335
18.	K = 3 *Knot* = {0.25, 0.5, 0.75}	2.903607	0.126164	2.879804
19.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.361994	0.206301	5.240448
20.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.287390	0.142192	5.232678
21.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.771931	0.233014	5.627981
22.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.647000	0.318493	5.556348
23.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.346098	0.147534	5.284452
24.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.669188	0.142192	3.633687
25.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.798341	0.120822	3.776206
26.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.441853	0.131507	4.400961

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
27.	K = 3 *Knot* = {0.25, 0.5, 0.75}	2.536252	0.126164	2.514579
28.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.106933	0.147534	4.059461
29.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.026752	0.152877	2.986698
30.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.878445	0.126164	3.843347
31.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.855702	0.115479	3.832098
32.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.185628	0.136849	4.141990
33.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.766991	0.254384	5.589631
34.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.618948	0.216986	3.527702
35.	K = 3 *Knot* = {0.25, 0.5, 0.75}	2.546388	0.152877	2.509362
36.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.122822	0.184932	5.013420
37.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.174178	0.168904	3.122488
38.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.042247	0.200959	3.958526
39.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.804075	0.216986	3.722319

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
40.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.533844	0.158219	4.471668
41.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	2.692868	0.120822	2.673284
42.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.252539	0.174247	4.187265
43.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.289142	0.168904	5.210855
44.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.816141	0.142192	3.775365
45.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.775088	0.152877	3.728319
46.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.417495	0.120822	3.393223
47.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.488322	0.179589	3.431079
48.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.006816	0.136849	3.965658
49.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	6.261021	0.286438	6.064026
50.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	3.883152	0.158219	3.831234

Lampiran 3. Hasil Tuning Parameter per Replikasi ($\rho = 0{,}8$)

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
1.	K = 3 *Knot* = {0.25, 0.5, 0.75}	6.262470	0.233014	6.097604
2.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	9.031600	0.222329	8.854412
3.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.175017	0.142192	5.124438
4.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	6.250444	0.131507	6.198566
5.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.278627	0.184932	5.188859
6.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.400866	0.168904	4.332381
7.	K = 3 *Knot* = {0.25, 0.5, 0.75}	3.990240	0.163562	3.937335
8.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.294781	0.147534	5.230840
9.	K = 3 *Knot* = {0.25, 0.5, 0.75}	6.135220	0.168904	6.039523
10.	K = 5 *Knot* = {0, 0.25, 0.5, 0.6667, 0.75}	7.116005	0.190274	6.975054
11.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	6.076860	0.168904	5.979317
12.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.873747	0.243699	5.713179

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
13.	K = 3 *Knot* = {0.25, 0.5, 0.75}	6.430461	1.200000	6.159235
14.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.670174	0.179589	5.570171
15.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.450551	0.179589	5.337158
16.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.421191	0.233014	4.311284
17.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.908656	0.281096	5.724443
18.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.515999	0.147534	5.444843
19.	K = 4 *Knot* = {0.15, 0.3, 0.5, 0.75}	6.182042	0.275753	6.029693
20.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.242166	0.190274	5.145674
21.	K = 3 *Knot* = {0.25, 0.5, 0.75}	7.200296	0.366575	6.979010
22.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.612964	0.163562	5.526451
23.	K = 3 *Knot* = {0.25, 0.5, 0.75}	6.375752	0.163562	6.282344
24.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.403820	0.190274	5.306259
25.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.447419	0.158219	5.381149

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
26.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	6.253338	0.163562	6.156620
27.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.498211	0.142192	4.444809
28.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.298305	0.323836	5.132318
29.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.513574	0.254384	5.376199
30.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.332899	0.206301	5.217333
31.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	4.107175	0.233014	4.019341
32.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.442136	0.152877	4.388023
33.	K = 3 *Knot* = {0.25, 0.5, 0.75}	4.719668	0.152877	4.664765
34.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.190894	0.200959	5.086434
35.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	7.185215	0.243699	7.046904
36.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.065172	1.200000	4.925646
37.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.909450	0.249041	5.740204
38.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	6.319718	0.184932	6.213660

Replikasi (r)	Parameter Tuning Optimal Spline Truncated	GCV Spline Truncated	Parameter Tuning Optimal *Kernel* Regression	GCV *Kernel* Regression
39.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.268213	0.238356	5.107457
40.	K = 3 *Knot* = {0.25, 0.5, 0.75}	6.320615	0.147534	6.245707
41.	K = 3 *Knot* = {0.25, 0.5, 0.75}	5.196067	0.216986	5.070068
42.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	4.262221	0.115479	4.239449
43.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	6.351646	0.200959	6.221899
44.	K = 3 *Knot* = {0.25, 0.5, 0.75}	7.066778	0.136849	7.007252
45.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	6.087006	0.216986	5.954758
46.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.976065	0.254384	5.828988
47.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.137338	0.190274	5.023239
48.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	3.968810	0.142192	3.925254
49.	K = 4 *Knot* = {0.2, 0.4, 0.6, 0.8}	5.732669	0.142192	5.671564
50.	K = 5 *Knot* = {0.1667, 0.3333, 0.5, 0.6667, 0.8333}	5.866369	0.398630	5.666802

Lampiran 4. Data Persentase Penduduk Miskin dan Variabel yang Diduga Berpengaruh per Provinsi di Indonesia Tahun 2016 – 2024

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
Aceh	2016	16.73	8.13	9.36
	2017	16.89	7.39	9.42
	2018	15.97	6.54	9.46
	2019	15.32	5.48	9.59
	2020	14.99	5.4	9.71
	2021	15.33	6.3	9.77
	2022	14.64	5.97	9.79
	2023	14.45	5.75	9.89
	2024	14.23	5.56	9.95
Sumatera Utara	2016	10.35	6.49	9.46
	2017	10.22	6.41	9.55
	2018	9.22	5.61	9.61
	2019	8.83	5.57	9.71
	2020	8.75	4.71	9.83
	2021	9.01	6.01	9.88
	2022	8.42	5.47	9.99
	2023	8.15	5.24	10.07
	2024	7.99	5.1	10.18
Sumatera Barat	2016	7.09	5.81	8.97
	2017	6.87	5.8	9.02
	2018	6.65	5.68	9.1
	2019	6.42	5.38	9.22
	2020	6.28	5.25	9.34
	2021	6.63	6.67	9.46
	2022	5.92	6.17	9.51
	2023	5.95	5.9	9.59
	2024	5.97	5.79	9.72
Riau	2016	7.98	5.94	8.97
	2017	7.78	5.76	9.06
	2018	7.39	5.55	9.11
	2019	7.08	5.36	9.35
	2020	6.82	4.92	9.47
	2021	7.12	4.96	9.52
	2022	6.78	4.4	9.54
	2023	6.68	4.25	9.6
	2024	6.67	3.85	9.69
Jambi	2016	8.41	4.66	8.55
	2017	8.19	3.67	8.61
	2018	7.92	3.56	8.7
	2019	7.6	3.52	8.86
	2020	7.58	4.26	8.97

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2021	8.09	4.76	9.03
	2022	7.62	4.7	9.07
	2023	7.58	4.5	9.16
	2024	7.1	4.45	9.26
Sumatera Selatan	2016	13.54	3.94	8.32
	2017	13.19	3.8	8.41
	2018	12.8	4.08	8.48
	2019	12.71	4.02	8.6
	2020	12.66	3.9	8.68
	2021	12.84	5.17	8.78
	2022	11.9	4.74	8.82
	2023	11.78	4.53	8.9
	2024	10.97	3.97	8.98
Bengkulu	2016	17.32	3.84	8.82
	2017	16.45	2.81	8.91
	2018	15.43	2.63	8.94
	2019	15.23	2.41	9.08
	2020	15.03	3.08	9.2
	2021	15.22	3.72	9.26
	2022	14.62	3.39	9.28
	2023	14.04	3.21	9.35
	2024	13.56	3.17	9.4
Lampung	2016	14.29	4.54	8.1
	2017	13.69	4.43	8.19
	2018	13.14	4.32	8.29
	2019	12.62	3.95	8.36
	2020	12.34	4.26	8.51
	2021	12.62	4.54	8.56
	2022	11.57	4.31	8.61
	2023	11.11	4.18	8.72
	2024	10.69	4.12	8.8
Kep. Bangka Belitung	2016	5.22	6.17	8.04
	2017	5.2	4.46	8.13
	2018	5.25	3.59	8.24
	2019	4.62	3.32	8.35
	2020	4.53	3.35	8.49
	2021	4.9	5.04	8.54
	2022	4.45	4.18	8.57
	2023	4.52	3.89	8.66
	2024	4.55	3.85	8.78
Kep. Riau	2016	5.98	9.03	9.9
	2017	6.06	6.44	10
	2018	6.2	7.3	10.01

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2019	5.9	7.02	10.13
	2020	5.92	5.98	10.22
	2021	6.12	10.12	10.38
	2022	6.24	8.02	10.46
	2023	5.69	7.61	10.52
	2024	5.37	6.94	10.65
DKI Jakarta	2016	3.75	5.77	10.92
	2017	3.77	5.36	10.97
	2018	3.57	5.73	11.06
	2019	3.47	5.5	11.11
	2020	4.53	5.15	11.17
	2021	4.72	8.51	11.2
	2022	4.69	8	11.3
	2023	4.44	7.57	11.42
	2024	4.3	6.03	11.49
Jawa Barat	2016	8.95	8.57	8.41
	2017	8.71	8.49	8.46
	2018	7.45	8.22	8.61
	2019	6.91	7.78	8.79
	2020	7.88	7.71	8.96
	2021	8.4	8.92	9.03
	2022	8.06	8.35	9.14
	2023	7.62	7.89	9.16
	2024	7.46	6.91	9.24
Jawa Tengah	2016	13.27	4.2	7.7
	2017	13.01	4.15	7.77
	2018	11.32	4.19	7.84
	2019	10.8	4.19	8.03
	2020	11.41	4.2	8.19
	2021	11.79	5.96	8.26
	2022	10.93	5.75	8.38
	2023	10.77	5.24	8.44
	2024	10.47	4.39	8.47
DI Yogyakarta	2016	13.34	2.81	9.62
	2017	13.02	2.84	9.68
	2018	12.13	3	9.73
	2019	11.7	2.89	9.83
	2020	12.28	3.38	9.95
	2021	12.8	4.28	8.26
	2022	11.34	3.73	10.07
	2023	11.04	3.58	10.16
	2024	10.83	3.24	10.23
Jawa Timur	2016	12.05	4.14	7.78

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2017	11.77	4.1	7.87
	2018	10.98	3.77	7.93
	2019	10.37	3.77	8.11
	2020	11.09	3.6	8.31
	2021	11.4	5.17	8.37
	2022	10.38	4.81	8.5
	2023	10.35	4.33	8.53
	2024	9.79	3.74	8.69
Banten	2016	5.42	7.95	8.79
	2017	5.45	7.75	8.87
	2018	5.24	7.72	8.93
	2019	5.09	7.55	9.07
	2020	5.92	7.99	9.22
	2021	6.66	9.01	9.29
	2022	6.16	8.53	9.46
	2023	6.17	7.97	9.48
	2024	5.84	7.02	9.55
Bali	2016	4.25	2.12	8.84
	2017	4.25	1.28	8.93
	2018	4.01	0.88	9
	2019	3.79	1.22	9.19
	2020	3.78	1.25	9.31
	2021	4.53	5.42	9.45
	2022	4.57	4.84	9.74
	2023	4.25	3.73	9.74
	2024	4	1.87	9.87
Nusa Tenggara Barat	2016	16.07	3.86	7.64
	2017	14.75	3.28	7.69
	2018	14.56	3.15	7.98
	2019	13.97	3.04	8.08
	2020	14.14	3.97	8.13
	2021	13.68	3.92	8.31
	2022	13.85	3.73	8.39
	2023	12.91	3.3	8.5
	2024	16.07	3.86	7.64
Nusa Tenggara Timur	2016	22.19	3.59	7.54
	2017	21.85	3.21	7.62
	2018	21.35	2.82	7.7
	2019	21.09	2.98	7.98
	2020	20.9	2.64	8.09
	2021	20.99	3.38	8.2
	2022	20.05	3.3	8.25
	2023	19.96	3.1	8.31

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2024	19.48	3.17	8.48
Kalimantan Barat	2016	7.87	4.58	7.49
	2017	7.88	4.22	7.57
	2018	7.77	4.09	7.65
	2019	7.49	4.06	7.8
	2020	7.17	4.47	7.9
	2021	7.15	5.73	8
	2022	6.73	4.86	8.1
	2023	6.71	4.52	8.17
	2024	6.32	4.2	8.28
Kalimantan Tengah	2016	5.66	3.67	8.52
	2017	5.37	3.13	8.59
	2018	5.17	3.14	8.66
	2019	4.98	3.21	8.83
	2020	4.82	3.33	8.95
	2021	5.16	4.25	9.03
	2022	5.28	4.2	9.03
	2023	5.11	3.84	9.07
	2024	5.17	3.67	9.17
Kalimantan Selatan	2016	4.85	3.63	8.28
	2017	4.73	3.53	8.37
	2018	4.54	3.72	8.45
	2019	4.55	3.41	8.59
	2020	4.38	3.67	8.69
	2021	4.83	4.33	8.74
	2022	4.49	4.2	8.89
	2023	4.29	3.95	8.95
	2024	4.11	3.89	9.02
Kalimantan Timur	2016	6.11	8.86	9.55
	2017	6.19	8.55	9.62
	2018	6.03	6.79	9.63
	2019	5.94	6.65	9.88
	2020	6.1	6.72	9.99
	2021	6.54	6.81	10.09
	2022	6.31	6.77	10.13
	2023	6.11	6.37	10.17
	2024	5.78	5.75	10.22
Kalimantan Utara	2016	6.23	3.92	9.01
	2017	7.22	5.17	9.1
	2018	7.09	4.7	9.18
	2019	6.63	5.84	9.24
	2020	6.8	5.71	9.3
	2021	7.36	4.67	9.4

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2022	6.77	4.62	9.45
	2023	6.45	4.1	9.53
	2024	6.32	4.01	9.58
Sulawesi Utara	2016	8.34	7.82	9.31
	2017	8.1	6.12	9.4
	2018	7.8	5.86	9.51
	2019	7.66	5.17	9.63
	2020	7.62	5.34	9.74
	2021	7.77	7.28	9.83
	2022	7.28	6.51	9.87
	2023	7.38	6.19	9.94
	2024	7.25	5.98	10.02
Sulawesi Tengah	2016	14.45	3.46	8.56
	2017	14.14	2.97	8.64
	2018	14.01	3.12	8.74
	2019	13.48	3.46	8.98
	2020	12.92	2.93	9.09
	2021	13	3.73	9.18
	2022	12.33	3.67	9.17
	2023	12.41	3.49	9.22
	2024	11.77	3.15	9.28
Sulawesi Selatan	2016	9.4	5.11	8.31
	2017	9.38	4.77	8.42
	2018	9.06	5.04	8.45
	2019	8.69	3.46	8.73
	2020	8.72	5.7	8.86
	2021	8.78	5.79	8.95
	2022	8.63	5.75	9.09
	2023	8.7	5.26	9.12
	2024	8.06	4.9	9.22
Sulawesi Tenggara	2016	12.88	3.78	8.86
	2017	12.81	3.14	8.93
	2018	11.63	2.77	9.03
	2019	11.24	2.88	9.25
	2020	11	3.1	9.41
	2021	8.78	4.22	9.52
	2022	11.17	3.86	9.59
	2023	11.43	3.66	9.62
	2024	11.21	3.22	9.74
Gorontalo	2016	17.72	3.88	7.71
	2017	17.65	3.65	7.77
	2018	16.81	3.38	7.83
	2019	15.52	3.25	8.11

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2020	15.22	3.29	8.26
	2021	15.61	3.41	8.32
	2022	15.42	3.25	8.39
	2023	15.15	3.07	8.48
	2024	14.57	3.05	8.64
Sulawesi Barat	2016	11.74	2.72	7.76
	2017	11.3	2.98	7.84
	2018	11.25	2.33	7.94
	2019	11.02	1.29	8.22
	2020	10.87	2.39	8.33
	2021	11.29	3.28	8.39
	2022	11.75	3.11	8.47
	2023	11.49	3.04	8.48
	2024	11.21	3.02	8.56
Maluku	2016	19.18	6.98	9.69
	2017	18.45	7.77	9.74
	2018	18.12	7.07	9.78
	2019	17.69	6.61	10.03
	2020	17.44	6.71	10.2
	2021	17.87	6.73	10.25
	2022	15.97	6.44	10.37
	2023	16.42	6.08	10.38
	2024	16.05	5.96	10.44
Maluku Utara	2016	6.33	3.43	8.96
	2017	6.35	4.82	9
	2018	6.64	4.56	9.07
	2019	6.77	4.96	9.32
	2020	6.78	4.09	9.42
	2021	6.89	5.06	9.51
	2022	6.23	4.98	9.6
	2023	6.46	4.6	9.61
	2024	6.32	4.16	9.7
Papua Barat	2016	25.43	5.73	9.57
	2017	25.1	7.52	9.67
	2018	23.01	6.27	9.73
	2019	22.17	5.81	9.92
	2020	21.37	6.78	10
	2021	21.84	6.18	10.03
	2022	21.33	5.78	10.14
	2023	20.49	5.53	10.14
	2024	21.66	4.31	9.87
Papua	2016	28.54	2.97	6.48
	2017	27.62	3.96	6.58

Provinsi (i)	Tahun Pengamatan (t)	Variabel Respon (y_{it})	Variabel Prediktor (x_{itq})	
			x_{it1}	x_{it2}
	2018	27.74	2.75	6.66
	2019	22.17	3.22	6.85
	2020	26.64	3.42	6.96
	2021	26.86	3.77	7.05
	2022	26.56	3.6	7.31
	2023	26.03	3.49	7.34
	2024	17.26	5.81	10.7

Keterangan:

Variabel	Deskripsi
y_{it}	Persentase Penduduk Miskin
x_{it1}	Tingkat Pengangguran Terbuka
x_{it2}	Rata-Rata Lama Sekolah

Lampiran 5. Statistik Deskriptif P0 per Provinsi (Lintas Tahun 2016–2024)

Provinsi	Mean P0	SD (lintas tahun)	Min–Max (lintas tahun)
Papua	25.491	3.576	17.26–28.54
Papua Barat	22.489	1.715	20.49–25.43
Nusa Tenggara Timur	20.873	0.896	19.48–22.19
Maluku	17.466	1.112	15.97–19.18
Gorontalo	15.963	1.141	14.57–17.72
Aceh	15.394	0.958	14.23–16.89
Bengkulu	15.211	1.146	13.56–17.32
Nusa Tenggara barat	14.490	1.145	12.91–16.48
Sulawesi Tengah	13.168	0.916	11.77–14.45
Sumatera Selatan	12.488	0.795	10.97–13.54
Lampung	12.452	1.180	10.69–14.29
DI Yogyakarta	12.053	0.892	10.83–13.34
Jawa Tengah	11.530	0.996	10.47–13.27
Sulawesi Tenggara	11.350	1.191	8.78–12.88
Sulawesi Barat	11.324	0.296	10.87–11.75
Jawa Timur	10.909	0.746	9.79–12.05
Sumatera Utara	8.993	0.831	7.99–10.35
Sulawesi Selatan	8.824	0.413	8.06–9.40
Jawa Barat	7.938	0.659	6.91–8.95
Jambi	7.788	0.400	7.10–8.41
Sulawesi Utara	7.689	0.366	7.25–8.34
Kalimantan Barat	7.232	0.565	6.32–7.88
Riau	7.144	0.480	6.67–7.98
Kalimantan Utara	6.763	0.398	6.23–7.36
Maluku Utara	6.530	0.243	6.23–6.89
Sumatera Barat	6.420	0.424	5.92–7.09
Kalimantan Timur	6.123	0.216	5.78–6.54

Provinsi	Mean P0	SD (lintas tahun)	Min–Max (lintas tahun)
Kep. Riau	5.942	0.273	5.37–6.24
Banten	5.772	0.512	5.09–6.66
Kalimantan Tengah	5.191	0.237	4.82–5.66
Kep. Bangka Belitung	4.804	0.339	4.45–5.25
Kalimantan Selatan	4.530	0.248	4.11–4.85
Bali	4.159	0.287	3.78–4.57
DKI Jakarta	4.138	0.496	3.47–4.72

Lampiran 6. Hasil Cross-Validation Berbasis Tahun (2016–2024) pada Data Riil.

Tahun	Metode	Parameter tuning optimal	RMSE (within)	MAE (within)
2016	*Spline Truncated*	K = 3	0.985377	0.806675
	Kernel	h = 0,06	0.8915145	0.715713
2017	*Spline Truncated*	K = 3	0.713176	0.558638
	Kernel	h = 0,06	0.700278	0.552767
2018	*Spline Truncated*	K = 5	0.488064	0.314961
	Kernel	h = 0,07	0.487623	0.317177
2019	*Spline Truncated*	K = 4	0.717481	0.414504
	Kernel	h = 0,06	0.695703	0.388116
2020	*Spline Truncated*	K = 4	0.365456	0.248046
	Kernel	h = 0,07	0.36862	0.251442
2021	*Spline Truncated*	K = 4	0.695124	0.530043
	Kernel	h = 0,07	0.691247	0.524132
2022	*Spline Truncated*	K = 3	0.652864	0.550304
	Kernel	h = 0,07	0.545589	0.451924
2023	*Spline Truncated*	K = 3	0.545753	0.443842
	Kernel	h = 0,06	0.558618	0.454328
2024	*Spline Truncated*	K = 5	1.421806	0.693839
	Kernel	h = 0,07	1.469831	0.711887

Lampiran 7. Kode Simulasi Data Longitudinal (Pembangkitan Data dan Estimasi Model)

```
SIMULASI DATA LONGITUDINAL
## ========================
need_pkgs <- c("dplyr")
for (p in need_pkgs) {
  if (!requireNamespace(p, quietly = TRUE)) install.packages(p)
}
library(dplyr)
## =========SETTING UTAMA================
R_REPL <- 50
N_SUBJ <- 30
T_TIME <- 5
SIGMA  <- 1
RHO_VEC <- c(0.2, 0.5, 0.8)
SEED <- 2025
OUT_METRICS <- "hasil_simulasi_metrics.csv"
OUT_PREDS   <- "hasil_simulasi_predictions.csv"
OUT_DETAIL  <- "hasil_simulasi_detail_per_replikasi.csv"
OUT_LONG    <- "hasil_simulasi_longdata.csv"
OUT_LONG_FULL <- "hasil_simulasi_longdata_full.csv"
set.seed(SEED)
## ===========SETTING "CIRI LONGITUDINAL============
TAU_ALPHA <- 1.0
TAU_BETA  <- 0.6
TAU_AMP   <- 0.35
PHASE_RANGE <- 0.08
SIGMA_TV_STRENGTH <- 0.70
SIGMA_SD_INDIV    <- 0.25
RHO_TV_STRENGTH <- 0.20
```

```
UNBALANCED <- TRUE
MISS_TARGET <- 0.18
MISS_MIN_OBS_PER_ID <- 2
MISS_B1_TIME <- 1.0
MISS_B2_ALPHA <- 0.7
MISS_B3_LAST  <- 0.6
DO_CV_TIME <- TRUE
## ==========f(u) baseline================
f_true <- function(u) {
  2 + 1.2*u - 2*u^2 + 0.8*sin(2*pi*u)
}
clip <- function(x, lo, hi) pmin(pmax(x, lo), hi)
## ===========Error AR(1)===============
## eps_t = rho_t * eps_{t-1} + e_t
generate_ar1_tv <- function(T, rho_base, sigma_vec, u_vec, rho_tv_strength = 0) {
  eps <- numeric(T)
  rho_vec <- rep(rho_base, T)
  if (rho_tv_strength > 0) {
    rho_vec <- rho_base * (1 - rho_tv_strength * u_vec)
  }
  rho_vec <- clip(rho_vec, -0.95, 0.95)
  denom <- max(1e-6, 1 - rho_base^2)
  eps[1] <- rnorm(1, 0, sigma_vec[1] / sqrt(denom))
  if (T >= 2) {
    for (t in 2:T) {
      eps[t] <- rho_vec[t] * eps[t-1] + rnorm(1, 0, sigma_vec[t])
    }
  }
  eps
}
## ===========Missingness berpola (MAR-ish)==============
```

```
calibrate_logit_intercept <- function(lp, target) {
  f <- function(b0) mean(plogis(b0 + lp)) - target
  out <- tryCatch(
    uniroot(f, lower = -20, upper = 20)$root,
    error = function(e) qlogis(clip(target, 1e-6, 1 - 1e-6)) - mean(lp)
  )
  out
}
apply_missingness <- function(dat,
                              target = 0.15,
                              min_obs_per_id = 2,
                              b1_time = 1.0,
                              b2_alpha = 0.6,
                              b3_last = 0.6) {
  lp <- b1_time * dat$u +
    b2_alpha * abs(dat$alpha_i) +
    b3_last * as.numeric(dat$time == max(dat$time))
  b0 <- calibrate_logit_intercept(lp, target)
  p_miss <- plogis(b0 + lp)
  miss <- rbinom(nrow(dat), 1, p_miss)
  ids <- unique(dat$id)
  for (idv in ids) {
    idx <- which(dat$id == idv)
    n_keep <- sum(miss[idx] == 0)
    if (n_keep < min_obs_per_id) {
      need <- min_obs_per_id - n_keep
      cand <- idx[miss[idx] == 1]
      if (length(cand) > 0) {
        unmiss <- sample(cand, size = min(need, length(cand)))
        miss[unmiss] <- 0
      }
```

```
    if (sum(miss[idx] == 0) < min_obs_per_id) {
      ord <- idx[order(dat$time[idx])]
      keep <- ord[1:min_obs_per_id]
      miss[idx] <- 1
      miss[keep] <- 0
    }
  }
 }
 dat$miss <- miss
 list(dat_full = dat,
      dat_obs = dat[dat$miss == 0, , drop = FALSE],
      miss_rate = mean(miss))
}
## ============= Generate data longitudinal=============
## f_i(u) = alpha_i + (1+amp_i) f_true(u_shift) + beta_i u
generate_longitudinal_data <- function(rho, n = 30, T = 5, sigma = 1,
                                       unbalanced = TRUE) {
 time_vec <- 1:T
 u_vec <- (time_vec - 1) / (T - 1)
 out <- vector("list", n)
 for (i in 1:n) {
  alpha_i <- rnorm(1, 0, TAU_ALPHA)
  beta_i  <- rnorm(1, 0, TAU_BETA)
  amp_i   <- rnorm(1, 0, TAU_AMP)
  phase_i <- runif(1, -PHASE_RANGE, PHASE_RANGE)
  u_shift <- clip(u_vec + phase_i, 0, 1)
  f_i_vec <- alpha_i + (1 + amp_i) * f_true(u_shift) + beta_i * u_vec
  sigma_i <- sigma * exp(rnorm(1, 0, SIGMA_SD_INDIV))
  sigma_t <- sigma_i * (1 + SIGMA_TV_STRENGTH * u_vec)
  sigma_t <- pmax(sigma_t, 1e-6)
```

```
  eps_i <- generate_ar1_tv(
    T = T,
    rho_base = rho,
    sigma_vec = sigma_t,
    u_vec = u_vec,
    rho_tv_strength = RHO_TV_STRENGTH
  )
  y_i <- f_i_vec + eps_i
  out[[i]] <- data.frame(
    id = i,
    time = time_vec,
    u = u_vec,
    alpha_i = alpha_i,
    beta_i  = beta_i,
    amp_i   = amp_i,
    phase_i = phase_i,
    f = f_i_vec,
    eps = eps_i,
    y = y_i
  )
}
dat_full <- do.call(rbind, out)
rownames(dat_full) <- NULL
if (!unbalanced) {
  dat_full$miss <- 0
  return(list(dat = dat_full, dat_full = dat_full, miss_rate = 0))
}
miss_out <- apply_missingness(
  dat_full,
  target = MISS_TARGET,
```

```
  min_obs_per_id = MISS_MIN_OBS_PER_ID,
  b1_time = MISS_B1_TIME,
  b2_alpha = MISS_B2_ALPHA,
  b3_last = MISS_B3_LAST
 )
 list(dat = miss_out$dat_obs, dat_full = miss_out$dat_full, miss_rate = miss_out$miss_rate)
}
## ==========Spline Truncated Linear + GCV (K = 3..5)===============
make_trunc_basis_linear <- function(u, knots) {
 u <- as.numeric(u)
 Bt <- sapply(knots, function(k) pmax(u - k, 0))
 Bt <- as.matrix(Bt)
 colnames(Bt) <- paste0("k", seq_along(knots))
 data.frame(u = u, Bt)
}
gcv_ols <- function(fit) {
 n <- length(fit$fitted.values)
 sse <- sum(resid(fit)^2)
 p <- fit$rank
 (sse/n) / (1 - p/n)^2
}
knot_candidates <- function(u, K, type = c("equal", "quantile")) {
 type <- match.arg(type)
 if (type == "equal") {
  ks <- seq(min(u), max(u), length.out = K + 2)[2:(K+1)]
 } else {
  probs <- seq(0, 1, length.out = K + 2)[2:(K+1)]
  ks <- as.numeric(quantile(u, probs = probs, names = FALSE, type = 7))
 }
 sort(unique(ks))
```

```
}
fit_trunc_spline_gcv <- function(dat, K_set = 3:5) {
  u <- dat$u
  y <- dat$y
  best <- list(gcv = Inf)
  for (K in K_set) {
    for (typ in c("equal", "quantile")) {
      knots <- knot_candidates(u, K, typ)
      B <- make_trunc_basis_linear(u, knots)
      fit <- lm(y ~ ., data = cbind(y = y, B))
      gcv <- gcv_ols(fit)
      if (is.finite(gcv) && gcv < best$gcv) {
        best <- list(
          K = K, type = typ, knots = knots,
          fit_obj = fit, gcv = gcv
        )
      }
    }
  }
  best$y_hat <- as.numeric(fitted(best$fit_obj))
  best$trS <- best$fit_obj$rank
  best
}
predict_trunc_spline <- function(sp_obj, u_new) {
  Bn <- make_trunc_basis_linear(u_new, sp_obj$knots)
  as.numeric(predict(sp_obj$fit_obj, newdata = Bn))
}
## ============Kernel NW Gaussian + GCV=============
K_gauss <- function(z) exp(-0.5*z^2)
nw_fit_train <- function(u_train, y_train, h) {
  z <- outer(u_train, u_train, function(a,b) (a-b)/h)
```

```
  W <- K_gauss(z)
  denom <- rowSums(W)
  denom[denom == 0] <- 1e-12
  S <- W / denom
  y_hat <- as.numeric(S %*% y_train)
  list(y_hat = y_hat, S = S)
}
fit_Kernel_gcv <- function(dat) {
  u <- as.numeric(dat$u)
  y <- as.numeric(dat$y)
  n <- length(y)
  h_grid <- seq(0.03, 1.20, length.out = 220)
  best <- list(gcv = Inf)
  for (h in h_grid) {
    out <- nw_fit_train(u, y, h)
    y_hat <- out$y_hat
    S <- out$S
    sse <- sum((y - y_hat)^2)
    trS <- sum(diag(S))
    gcv <- (sse/n) / (1 - trS/n)^2
    if (is.finite(gcv) && gcv < best$gcv) {
      best <- list(h = h, gcv = gcv, trS = trS, y_hat = y_hat)
    }
  }
  best
}
nw_predict <- function(u_train, y_train, u_new, h) {
  z <- outer(u_new, u_train, function(a,b) (a-b)/h)
  W <- K_gauss(z)
  denom <- rowSums(W)
  denom[denom == 0] <- 1e-12
```

```
  as.numeric((W / denom) %*% y_train)
}
## ============METRICS + Diagnostik residual longitudinal per-id=============
metrics_basic <- function(y, yhat) {
  e <- y - yhat
  mse <- mean(e^2)
  rmse <- sqrt(mse)
  mae <- mean(abs(e))
  denom <- sum((y - mean(y))^2)
  r2 <- if (denom <= 1e-12) NA_real_ else 1 - sum(e^2) / denom
  c(mse = mse, rmse = rmse, mae = mae, r2 = r2)
}
dw_stat <- function(e) sum(diff(e)^2) / max(1e-12, sum(e^2))
acf_lag1 <- function(e) {
  if (length(e) < 3) return(NA_real_)
  a <- stats::acf(e, plot = FALSE, lag.max = 1)$acf
  as.numeric(a[2])
}
diag_by_id <- function(dat, e) {
  dat2 <- dat
  dat2$e <- e
  ids <- unique(dat2$id)
  dw_vec <- numeric(0)
  acf1_vec <- numeric(0)
  w_vec <- integer(0)
  for (idv in ids) {
    sub <- dat2[dat2$id == idv, ]
    sub <- sub[order(sub$time), ]
    if (nrow(sub) >= 2) {
      dw_vec <- c(dw_vec, dw_stat(sub$e))
```

```
    w_vec  <- c(w_vec, nrow(sub))
   } else {
    dw_vec <- c(dw_vec, NA_real_)
    w_vec  <- c(w_vec, nrow(sub))
   }
   acf1_vec <- c(acf1_vec, acf_lag1(sub$e))
  }
  w_dw <- w_vec[!is.na(dw_vec)]
  w_ac <- w_vec[!is.na(acf1_vec)]
  dw_mean <- if (length(w_dw) == 0) NA_real_ else sum(dw_vec[!is.na(dw_vec)]
* w_dw) / sum(w_dw)
  acf1_mean <- if (length(w_ac) == 0) NA_real_ else
sum(acf1_vec[!is.na(acf1_vec)] * w_ac) / sum(w_ac)
  c(dw_mean = dw_mean, acf1_mean = acf1_mean)
}
metrics_f <- function(f, yhat) {
  e <- f - yhat
  mse <- mean(e^2)
  rmse <- sqrt(mse)
  mae <- mean(abs(e))
  denom <- sum((f - mean(f))^2)
  r2 <- if (denom <= 1e-12) NA_real_ else 1 - sum(e^2) / denom
  c(mse = mse, rmse = rmse, mae = mae, r2 = r2)
}
## ==========RUN SIMULATION + OUTPUT FILE DETAIL PER
REPLIKASI==========
results <- data.frame(
  rho = numeric(0),
  repl = integer(0),
  n_obs = integer(0),
  miss_rate = numeric(0),
```

```
  mse_y_spline = numeric(0),
  rmse_y_spline = numeric(0),
  mae_y_spline = numeric(0),
  r2_y_spline  = numeric(0),
  mse_y_Kernel = numeric(0),
  rmse_y_Kernel = numeric(0),
  mae_y_Kernel = numeric(0),
  r2_y_Kernel  = numeric(0),
  dw_y_spline = numeric(0),
  acf1_y_spline = numeric(0),
  dw_y_Kernel = numeric(0),
  acf1_y_Kernel = numeric(0),
  cv_rmse_spline = numeric(0),
  cv_mae_spline  = numeric(0),
  cv_rmse_Kernel = numeric(0),
  cv_mae_Kernel  = numeric(0),
  mse_f_spline = numeric(0),
  rmse_f_spline = numeric(0),
  mae_f_spline = numeric(0),
  r2_f_spline  = numeric(0),
  mse_f_Kernel = numeric(0),
  rmse_f_Kernel = numeric(0),
  mae_f_Kernel = numeric(0),
  r2_f_Kernel  = numeric(0),
  spline_K = integer(0),
  spline_knots = character(0),
  trS_spline = numeric(0),
  Kernel_h = numeric(0),
  trS_Kernel = numeric(0)
)
```

```
preds_store <- list()
detail_store <- list()
full_store <- list()
u_grid_full <- (0:(T_TIME-1)) / (T_TIME - 1)
for (rho in RHO_VEC) {
  for (r in 1:R_REPL) {
    gen <- generate_longitudinal_data(
      rho = rho,
      n = N_SUBJ,
      T = T_TIME,
      sigma = SIGMA,
      unbalanced = UNBALANCED
    )
    dat <- gen$dat
    dat_full <- gen$dat_full
    miss_rate <- gen$miss_rate
    ## Fit spline
    sp <- fit_trunc_spline_gcv(dat, K_set = 3:5)
    yhat_sp <- sp$y_hat
    ## Fit Kernel
    ke <- fit_Kernel_gcv(dat)
    yhat_ke <- ke$y_hat
    ## Metrics in-sample y
    my_sp <- metrics_basic(dat$y, yhat_sp)
    my_ke <- metrics_basic(dat$y, yhat_ke)
    ## Residual diag per-id
    dg_sp <- diag_by_id(dat, dat$y - yhat_sp)
    dg_ke <- diag_by_id(dat, dat$y - yhat_ke)
    ## Oracle metrics (f)
    mf_sp <- metrics_f(dat$f, yhat_sp)
    mf_ke <- metrics_f(dat$f, yhat_ke)
```

```
## Append metrics
results <- rbind(results, data.frame(
  rho = rho,
  repl = r,
  n_obs = nrow(dat),
  miss_rate = miss_rate,
  mse_y_spline = my_sp["mse"],
  rmse_y_spline = my_sp["rmse"],
  mae_y_spline = my_sp["mae"],
  r2_y_spline  = my_sp["r2"],
  mse_y_Kernel = my_ke["mse"],
  rmse_y_Kernel = my_ke["rmse"],
  mae_y_Kernel = my_ke["mae"],
  r2_y_Kernel  = my_ke["r2"],
  acf1_y_spline = dg_sp["acf1_mean"],
  acf1_y_Kernel = dg_ke["acf1_mean"],
  cv_rmse_spline = cv_sp["cv_rmse"],
  cv_mae_spline  = cv_sp["cv_mae"],
  cv_rmse_Kernel = cv_ke["cv_rmse"],
  cv_mae_Kernel  = cv_ke["cv_mae"],
  mse_f_spline = mf_sp["mse"],
  rmse_f_spline = mf_sp["rmse"],
  mae_f_spline = mf_sp["mae"],
  r2_f_spline  = mf_sp["r2"],
  mse_f_Kernel = mf_ke["mse"],
  rmse_f_Kernel = mf_ke["rmse"],
  mae_f_Kernel = mf_ke["mae"],
  r2_f_Kernel  = mf_ke["r2"],
  spline_K = sp$K,
  spline_knots = paste(round(sp$knots, 4), collapse = ","),
  trS_spline = sp$trS,
```

```
  Kernel_h = ke$h,
  trS_Kernel = ke$trS
 ))
 ## ==========SIMPAN DETAIL PER REPLIKASI (SEMUA BARIS)==========
 detail <- dat
 detail$rho <- rho
 detail$repl <- r
 detail$miss_rate <- miss_rate
 detail$x <- detail$u
 detail$f_true <- detail$f
 detail$yhat_spline <- as.numeric(yhat_sp)
 detail$res_spline  <- as.numeric(detail$y - yhat_sp)
 detail$yhat_Kernel <- as.numeric(yhat_ke)
 detail$res_Kernel  <- as.numeric(detail$y - yhat_ke)
 detail$spline_K <- sp$K
 detail$spline_knots <- paste(round(sp$knots, 4), collapse = ",")
 detail$Kernel_h <- ke$h
 detail$trS_spline <- sp$trS
 detail$trS_Kernel <- ke$trS
 detail <- detail[, c(
  "id","time","x","f_true","y","eps","miss","rho","repl","miss_rate",
  "yhat_spline","res_spline","yhat_Kernel","res_Kernel",
  "spline_K","spline_knots","trS_spline","Kernel_h","trS_Kernel"
 )]
 detail_store[[length(detail_store) + 1]] <- detail
 ## Simpan FULL panel
 full <- dat_full
 full$rho <- rho
 full$repl <- r
 full$miss_rate <- miss_rate
```

```
    full$x <- full$u
    full$f_true <- full$f
    full <- full[,
c("id","time","x","f_true","y","eps","miss","rho","repl","miss_rate")]
    full_store[[length(full_store) + 1]] <- full
    ## Simpan mean per u untuk repl=1 tiap rho
    if (r == 1) {
     tmp_sum <- dat %>%
      mutate(yhat_spline = yhat_sp, yhat_Kernel = yhat_ke) %>%
      group_by(u) %>%
      summarise(
       n_obs_u = n(),
       f_true_mean = mean(f),
       y_mean = mean(y),
       yhat_spline_mean = mean(yhat_spline),
       yhat_Kernel_mean = mean(yhat_Kernel),
       .groups = "drop"
      )
     tmp_full <- data.frame(u = u_grid_full) %>%
      left_join(tmp_sum, by = "u")
     tmp_full$rho <- rho
     tmp_full$repl <- r
     tmp_full$miss_rate <- miss_rate
     preds_store[[paste0("rho", rho)]] <- tmp_full
    }
  }
}
```

Lampiran 8. Kode Analisis Hasil Simulasi (Ringkasan Metrik, Visualisasi, dan Diagnostik Residual)

```
## ANALISIS HASIL SIMULASI
rm(list = ls())
## ========================
DATA_DIR <- "D:/Document/SKRIPSI/Data/Data Hasil"
setwd(DATA_DIR)
cat("Working directory:", getwd(), "\n")
cat("Files in folder:\n")
print(list.files())
## ========================
pkgs <- c("readr", "dplyr", "tidyr", "ggplot2")
need <- pkgs[!pkgs %in% installed.packages()[, "Package"]]
if (length(need) > 0) install.packages(need)
invisible(lapply(pkgs, library, character.only = TRUE))
## ========================
MET_FILE  <- "hasil_simulasi_metrics.csv"
PRED_FILE <- "hasil_simulasi_predictions.csv"
LONG_FILE <- "hasil_simulasi_longdata.csv"
LONG_FULL_FILE <- "hasil_simulasi_longdata_full.csv"
stopifnot(file.exists(MET_FILE))
stopifnot(file.exists(PRED_FILE))
stopifnot(file.exists(LONG_FILE))
metrics <- readr::read_csv(MET_FILE, show_col_types = FALSE)
pred    <- readr::read_csv(PRED_FILE, show_col_types = FALSE)
longdat <- readr::read_csv(LONG_FILE, show_col_types = FALSE)
longfull <- NULL
if (file.exists(LONG_FULL_FILE)) {
  longfull <- readr::read_csv(LONG_FULL_FILE, show_col_types = FALSE)
}
## enforce numeric rho
```

```
metrics$rho <- as.numeric(metrics$rho)
pred$rho    <- as.numeric(pred$rho)
longdat$rho <- as.numeric(longdat$rho)
if (!is.null(longfull)) longfull$rho <- as.numeric(longfull$rho)
## ========================
## u
if (!"u" %in% names(pred)    && "x" %in% names(pred))    pred    <- pred
%>% dplyr::mutate(u = x)
if (!"u" %in% names(longdat) && "x" %in% names(longdat)) longdat <- longdat
%>% dplyr::mutate(u = x)
## f
if (!"f" %in% names(longdat) && "f_true" %in% names(longdat)) {
  longdat <- longdat %>% dplyr::mutate(f = f_true)
}
if (!is.null(longfull)) {
  if (!"u" %in% names(longfull) && "x" %in% names(longfull)) longfull <-
longfull %>% dplyr::mutate(u = x)
  if (!"f" %in% names(longfull) && "f_true" %in% names(longfull)) longfull <-
longfull %>% dplyr::mutate(f = f_true)
}
## ========================
OUT_DIR <- file.path(DATA_DIR, "output_fase3")
if (!dir.exists(OUT_DIR)) dir.create(OUT_DIR, recursive = TRUE)
cat("Output folder:", OUT_DIR, "\n")
theme_skripsi <- function(base_size = 12) {
  ggplot2::theme_bw(base_size = base_size) +
    ggplot2::theme(
      panel.grid.major = ggplot2::element_line(linewidth = 0.25, color = "grey85"),
      panel.grid.minor = ggplot2::element_line(linewidth = 0.15, color = "grey92"),
      legend.position = "bottom",
      legend.title = ggplot2::element_blank(),
      plot.title = ggplot2::element_text(face = "bold")
```

```
  )
}
## ==========VALIDASI KOLOM METRICS==========
is_fix <- all(c("mse_y_spline","r2_y_spline","mse_y_Kernel","r2_y_Kernel") %in% names(metrics))
is_old <- all(c("mse_spline","r2_spline","mse_Kernel","r2_Kernel") %in% names(metrics))
if (!is_fix && !is_old) {
  stop("Kolom metrics tidak dikenali. Pastikan memakai output FASE 2 fix (mse_y_*) atau versi lama (mse_*).")
}
get_col <- function(df, nm_fix, nm_old = NULL) {
  if (nm_fix %in% names(df)) return(df[[nm_fix]])
  if (!is.null(nm_old) && nm_old %in% names(df)) return(df[[nm_old]])
  return(rep(NA_real_, nrow(df)))
}
## ==========IN-SAMPLE y==========
ringkas_y <- dplyr::bind_rows(
  metrics %>%
    dplyr::transmute(
      rho,
      Metode = "Spline Truncated",
      MSE  = get_col(metrics, "mse_y_spline", "mse_spline"),
      RMSE = get_col(metrics, "rmse_y_spline", NULL),
      MAE  = get_col(metrics, "mae_y_spline", NULL),
      R2   = get_col(metrics, "r2_y_spline", "r2_spline")
    ),
  metrics %>%
    dplyr::transmute(
      rho,
      Metode = "Kernel",
      MSE  = get_col(metrics, "mse_y_Kernel", "mse_Kernel"),
```

```
      RMSE = get_col(metrics, "rmse_y_Kernel", NULL),
      MAE  = get_col(metrics, "mae_y_Kernel", NULL),
      R2   = get_col(metrics, "r2_y_Kernel", "r2_Kernel")
    )
) %>%
  dplyr::group_by(rho, Metode) %>%
  dplyr::summarise(
    Mean_MSE = mean(MSE, na.rm = TRUE),
    SD_MSE   = sd(MSE, na.rm = TRUE),
    Mean_R2  = mean(R2, na.rm = TRUE),
    SD_R2    = sd(R2, na.rm = TRUE),
    Mean_RMSE = mean(RMSE, na.rm = TRUE),
    SD_RMSE   = sd(RMSE, na.rm = TRUE),
    Mean_MAE  = mean(MAE, na.rm = TRUE),
    SD_MAE    = sd(MAE, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  dplyr::arrange(rho, Metode)
write.csv(ringkas_y, file.path(OUT_DIR, "tabel_ringkasan_inSample_y.csv"),
row.names = FALSE)
## ==========GAMBAR DATA MENTAH SIMULASI==============
## 8B.1 Spaghetti plot
set.seed(2026)
N_SHOW <- 15
df_spag <- longdat %>%
  dplyr::filter(repl == 1) %>%
  dplyr::group_by(rho) %>%
  dplyr::group_modify(~{
    ids <- unique(.x$id)
    ids_s <- sample(ids, size = min(N_SHOW, length(ids)), replace = FALSE)
    .x %>% dplyr::filter(id %in% ids_s)
```

```
  }) %>%
  dplyr::ungroup()
mean_line <- longdat %>%
  dplyr::filter(repl == 1) %>%
  dplyr::group_by(rho, time, u) %>%
  dplyr::summarise(mean_y = mean(y, na.rm = TRUE), .groups = "drop")
p_spag <- ggplot2::ggplot(df_spag, ggplot2::aes(x = u, y = y, group = id, color =
factor(id))) +
  ggplot2::geom_line(alpha = 0.60, linewidth = 0.60) +
  ggplot2::geom_point(alpha = 0.55, size = 0.9) +
  ggplot2::geom_line(data = mean_line,
                     ggplot2::aes(x = u, y = mean_y, group = 1),
                     inherit.aes = FALSE,
                     color = "black", linewidth = 1.45) +
  ggplot2::facet_wrap(~rho, nrow = 1,
                      labeller = ggplot2::labeller(rho = function(x) paste0("rho = ", x))) +
  ggplot2::labs(
    x = "Waktu ternormalisasi (u)",
    y = "y (data mentah simulasi)",
    title = "Spaghetti Plot Data Mentah Simulasi + Garis Rata-rata",
    subtitle = "Menampilkan heterogenitas antar individu sebelum fitting model
(contoh 15 individu per rho; repl=1)"
  ) +
  ggplot2::guides(color = "none") +
  theme_skripsi()
ggplot2::ggsave(file.path(OUT_DIR,
"Gambar_Spaghetti_DataMentah_vs_rho.png"),
                p_spag, width = 11, height = 4.6, dpi = 300)
## 8B.2 lag-1 scatter dari error AR(1) murni (eps)
if (is.null(longfull) || !"eps" %in% names(longfull)) {
  stop("Butuh file hasil_simulasi_longdata_full.csv (kolom eps). Jalankan ulang
FASE 2 versi final agar eps tersimpan.")
```

```
}
eps_pairs <- longfull %>%
  dplyr::group_by(rho, repl, id) %>%
  dplyr::arrange(time, .by_group = TRUE) %>%
  dplyr::mutate(eps_lag1 = dplyr::lag(eps)) %>%
  dplyr::ungroup() %>%
  dplyr::filter(is.finite(eps_lag1), is.finite(eps))
## sampling agar plot tidak terlalu padat
set.seed(2026)
eps_pairs_s <- eps_pairs %>%
  dplyr::group_by(rho) %>%
  dplyr::group_modify(~{
    n_take <- min(2000, nrow(.x))
    dplyr::slice_sample(.x, n = n_take, replace = FALSE)
  }) %>%
  dplyr::ungroup()
p_lag1 <- ggplot2::ggplot(eps_pairs_s, ggplot2::aes(x = eps_lag1, y = eps)) +
  ggplot2::geom_point(alpha = 0.18, size = 0.9, color = "#1f77b4") +
  ggplot2::geom_smooth(method = "lm", se = FALSE, linewidth = 1.15, color =
"black") +
  ggplot2::facet_wrap(~rho, nrow = 1,
                      labeller = ggplot2::labeller(rho = function(x) paste0("rho = ", x))) +
  ggplot2::labs(
    x = expression(e[t-1]),
    y = expression(e[t]),
    title = "Lag-1 Scatter Error AR(1) Murni (eps)",
    subtitle = "Semakin besar rho, pola hubungan eps_t terhadap eps_{t-1} makin
kuat"
  ) +
  theme_skripsi()
```

```
ggplot2::ggsave(file.path(OUT_DIR, "Gambar_Lag1Scatter_Eps_vs_rho.png"),
                p_lag1, width = 11, height = 4.2, dpi = 300)

## 8B.3 Ringkasan ACF(1) eps per id (dirata-rata)
acf_lag1_safe <- function(x) {
  x <- as.numeric(x)
  if (length(x) < 3) return(NA_real_)
  if (sd(x, na.rm = TRUE) == 0) return(NA_real_)
  as.numeric(stats::acf(x, plot = FALSE, lag.max = 1)$acf[2])
}
acf_eps_raw <- longfull %>%
  dplyr::group_by(rho, repl, id) %>%
  dplyr::arrange(time, .by_group = TRUE) %>%
  dplyr::summarise(acf1_eps = acf_lag1_safe(eps), .groups = "drop")
acf_eps_sum <- acf_eps_raw %>%
  dplyr::group_by(rho) %>%
  dplyr::summarise(
    Mean_ACF1_eps = mean(acf1_eps, na.rm = TRUE),
    Median_ACF1_eps = median(acf1_eps, na.rm = TRUE),
    SD_ACF1_eps = sd(acf1_eps, na.rm = TRUE),
    .groups = "drop"
  ) %>% dplyr::arrange(rho)
readr::write_csv(acf_eps_sum, file.path(OUT_DIR, "tabel_acf1_eps_murni.csv"))
p_acf_eps <- ggplot2::ggplot(acf_eps_sum, ggplot2::aes(x = rho, y =
Mean_ACF1_eps)) +
  ggplot2::geom_line(linewidth = 1.10, color = "#1f77b4") +
  ggplot2::geom_point(size = 3.0, color = "#1f77b4") +
  ggplot2::geom_errorbar(ggplot2::aes(ymin = Mean_ACF1_eps - SD_ACF1_eps,
                                      ymax = Mean_ACF1_eps + SD_ACF1_eps),
                         width = 0.03, color = "grey35") +
  ggplot2::labs(
```

```
    x = "rho",
    y = "Mean ACF(1) eps (± SD)",
    title = "Ringkasan Autokorelasi: ACF(1) Error AR(1) Murni",
    subtitle = "Dihitung per id lalu dirata-rata; menjadi dasar bridging simulasi vs data riil"
  ) +
  theme_skripsi()
ggplot2::ggsave(file.path(OUT_DIR, "Gambar_ACF1_Eps_vs_rho.png"),
                p_acf_eps, width = 7.2, height = 4.3, dpi = 300)
## ===========PLOT MEAN MSE & R2 vs rho============
p_mse <- ggplot(ringkas_y, aes(x=rho, y=Mean_MSE, group=Metode, color=Metode)) +
  geom_line(linewidth=0.9) + geom_point(size=2.7) +
  geom_errorbar(aes(ymin=Mean_MSE-SD_MSE, ymax=Mean_MSE+SD_MSE), width=0.03) +
  labs(x=expression(rho), y="Mean MSE (± SD)",
       title="Perbandingan Mean MSE (in-sample y) vs Tingkat Autokorelasi") +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_MeanMSE_vs_rho.png"), p_mse, width=7, height=4, dpi=300)


p_r2 <- ggplot(ringkas_y, aes(x=rho, y=Mean_R2, group=Metode, color=Metode)) +
  geom_line(linewidth=0.9) + geom_point(size=2.7) +
  geom_errorbar(aes(ymin=Mean_R2-SD_R2, ymax=Mean_R2+SD_R2), width=0.03) +
  labs(x=expression(rho), y="Mean R² (± SD)",
       title="Perbandingan Mean R² (in-sample y) vs Tingkat Autokorelasi") +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_MeanR2_vs_rho.png"), p_r2, width=7, height=4, dpi=300)
## ============REPLIKASI “REPRESENTATIF” (per rho)============
mse_spl_col <- if (is_fix) "mse_y_spline" else "mse_spline"
```

```
mean_mse_spline <- metrics %>%
  dplyr::group_by(rho) %>%
  dplyr::summarise(mean_mse_spline = mean(.data[[mse_spl_col]],
na.rm=TRUE), .groups="drop")
rep_tbl <- metrics %>%
  dplyr::left_join(mean_mse_spline, by="rho") %>%
  dplyr::mutate(abs_diff = abs(.data[[mse_spl_col]] - mean_mse_spline)) %>%
  dplyr::group_by(rho) %>%
  dplyr::slice_min(order_by=abs_diff, n=1, with_ties=FALSE) %>%
  dplyr::ungroup() %>%
  dplyr::select(rho, repl,
                mse_spline = all_of(mse_spl_col),
                mean_mse_spline,
                spline_K = any_of("spline_K"),
                spline_knots = any_of("spline_knots"),
                Kernel_h = any_of("Kernel_h"),
                trS_spline = any_of("trS_spline"),
                trS_Kernel = any_of("trS_Kernel"))
write.csv(rep_tbl, file.path(OUT_DIR, "tabel_replikasi_representatif.csv"),
row.names=FALSE)
## ==========VISUALISASI PER-REPL
REPRESENTATIF===============
avg_acf_by_id <- function(df, res_col, lag.max) {
  ids <- sort(unique(df$id))
  mat <- sapply(ids, function(i) {
    s <- df[df$id == i, ]
    s <- s[order(s$time), , drop=FALSE]
    n <- nrow(s)
    out <- rep(NA_real_, lag.max + 1)
    if (n >= 2) {
      lm_i <- min(lag.max, n - 1)
```

```
    acv <- as.numeric(stats::acf(s[[res_col]], plot=FALSE, lag.max=lm_i)$acf)
    out[1:(lm_i+1)] <- acv
   }
   out
  })
  if (is.vector(mat)) mat <- matrix(mat, ncol=1)
  data.frame(lag=0:lag.max, val=rowMeans(mat, na.rm=TRUE),
             se=apply(mat, 1, sd, na.rm=TRUE) / sqrt(ncol(mat)))
}
avg_pacf_by_id <- function(df, res_col, lag.max) {
  ids <- sort(unique(df$id))
  mat <- sapply(ids, function(i) {
   s <- df[df$id == i, ]
   s <- s[order(s$time), , drop=FALSE]
   n <- nrow(s)
   out <- rep(NA_real_, lag.max)
   if (n >= 3 && lag.max >= 1) {
    lm_i <- min(lag.max, n - 1)
    pcv <- as.numeric(stats::pacf(s[[res_col]], plot=FALSE, lag.max=lm_i)$acf)
    out[1:lm_i] <- pcv
   }
   out
  })
  if (is.vector(mat)) mat <- matrix(mat, ncol=1)
  data.frame(lag=1:lag.max, val=rowMeans(mat, na.rm=TRUE),
             se=apply(mat, 1, sd, na.rm=TRUE) / sqrt(ncol(mat)))
}
plot_fit_and_acf <- function(rho_val, repl_val) {
  df <- longdat %>% dplyr::filter(rho == rho_val, repl == repl_val)
  if (nrow(df) == 0) return(invisible(FALSE))
```

```
  df_u <- df %>%
    dplyr::group_by(u) %>%
    dplyr::summarise(
      y_mean  = mean(y),
      f_mean  = mean(f),
      yhat_spline_mean = mean(yhat_spline),
      yhat_Kernel_mean = mean(yhat_Kernel),
      .groups="drop"
    ) %>% dplyr::arrange(u)
  p_fit <- ggplot() +
    geom_point(data=df, aes(x=u, y=y),
               position=position_jitter(width=0.01, height=0),
               alpha=0.35) +
    geom_line(data=df_u, aes(x=u, y=f_mean, linetype="f (true, mean antar
individu)"), linewidth=1.0) +
    geom_line(data=df_u, aes(x=u, y=yhat_spline_mean, linetype="Spline
Truncated"), linewidth=1.0) +
    geom_line(data=df_u, aes(x=u, y=yhat_Kernel_mean, linetype="Kernel"),
linewidth=1.0) +
    scale_linetype_manual(values=c(
      "f (true, mean antar individu)"="solid",
      "Spline Truncated"="dashed",
      "Kernel"="dotdash"
    )) +
    labs(x="u (waktu ternormalisasi)", y="y",
         title=paste0("Data Representatif: ρ=", rho_val, ", repl=", repl_val),
         linetype="Kurva") +
    theme_skripsi()
  ggsave(file.path(OUT_DIR, paste0("Gambar_Fit_rho", rho_val, "_repl",
repl_val, ".png")),
         p_fit, width=7, height=4, dpi=300)
  Tobs <- max(df$time, na.rm=TRUE)
```

```
  lag.max <- max(1, min(4, Tobs - 1))
  acf_spl  <- avg_acf_by_id(df, "res_spline", lag.max)
  pacf_spl <- avg_pacf_by_id(df, "res_spline", lag.max)
  acf_ker  <- avg_acf_by_id(df, "res_Kernel", lag.max)
  pacf_ker <- avg_pacf_by_id(df, "res_Kernel", lag.max)
  png(file.path(OUT_DIR, paste0("Gambar_ACF_PACF_Spline_rho", rho_val,
"_repl", repl_val, ".png")),
      width=900, height=700)
  par(mfrow=c(2,1), mar=c(4,4,3,1))
  plot(acf_spl$lag, acf_spl$val, type="h", lwd=2,
       main=paste0("ACF Residual (avg per subjek) – Spline | ρ=", rho_val),
       xlab="Lag", ylab="ACF")
  abline(h=0); lines(acf_spl$lag, acf_spl$val + 1.96*acf_spl$se, lty=2)
  lines(acf_spl$lag, acf_spl$val - 1.96*acf_spl$se, lty=2)
  plot(pacf_spl$lag, pacf_spl$val, type="h", lwd=2,
       main=paste0("PACF Residual (avg per subjek) – Spline | ρ=", rho_val),
       xlab="Lag", ylab="PACF")
  abline(h=0); lines(pacf_spl$lag, pacf_spl$val + 1.96*pacf_spl$se, lty=2)
  lines(pacf_spl$lag, pacf_spl$val - 1.96*pacf_spl$se, lty=2)
  dev.off()
  png(file.path(OUT_DIR, paste0("Gambar_ACF_PACF_Kernel_rho", rho_val,
"_repl", repl_val, ".png")),
      width=900, height=700)
  par(mfrow=c(2,1), mar=c(4,4,3,1))
  plot(acf_ker$lag, acf_ker$val, type="h", lwd=2,
       main=paste0("ACF Residual (avg per subjek) – Kernel | ρ=", rho_val),
       xlab="Lag", ylab="ACF")
  abline(h=0); lines(acf_ker$lag, acf_ker$val + 1.96*acf_ker$se, lty=2)
  lines(acf_ker$lag, acf_ker$val - 1.96*acf_ker$se, lty=2)
  plot(pacf_ker$lag, pacf_ker$val, type="h", lwd=2,
       main=paste0("PACF Residual (avg per subjek) – Kernel | ρ=", rho_val),
```

```
      xlab="Lag", ylab="PACF")
  abline(h=0); lines(pacf_ker$lag, pacf_ker$val + 1.96*pacf_ker$se, lty=2)
  lines(pacf_ker$lag, pacf_ker$val - 1.96*pacf_ker$se, lty=2)
  dev.off()
  invisible(TRUE)
}
apply(rep_tbl[, c("rho","repl")], 1, function(v) {
  plot_fit_and_acf(rho_val=as.numeric(v[1]), repl_val=as.integer(v[2]))
})
## ===========ACF(1) & Ljung–Box (lag 1)==============
acf_at_lag <- function(x, lag_k=1) {
  x <- as.numeric(x)
  if (length(x) <= lag_k) return(NA_real_)
  if (sd(x, na.rm=TRUE) == 0) return(NA_real_)
  as.numeric(stats::acf(x, plot=FALSE, lag.max=lag_k)$acf[lag_k+1])
}
ljung_box_p <- function(x, lag_k=1) {
  x <- as.numeric(x)
  if (length(x) <= lag_k + 1) return(NA_real_)
  if (sd(x, na.rm=TRUE) == 0) return(NA_real_)
  out <- try(stats::Box.test(x, lag=lag_k, type="Ljung-Box"), silent=TRUE)
  if (inherits(out,"try-error")) return(NA_real_)
  as.numeric(out$p.value)
}
Tobs <- max(longdat$time, na.rm=TRUE)
LAG_ACF <- min(1, Tobs - 1)
LAG_LB  <- min(1, Tobs - 1)
acf_tbl <- longdat %>%
  dplyr::select(rho, repl, id, time, res_spline, res_Kernel) %>%
  tidyr::pivot_longer(cols=c(res_spline, res_Kernel),
                      names_to="MetodeRaw", values_to="res") %>%
```

```
  dplyr::mutate(Metode = ifelse(MetodeRaw=="res_spline","Spline
Truncated","Kernel")) %>%
  dplyr::group_by(rho, repl, Metode, id) %>%
  dplyr::arrange(time, .by_group=TRUE) %>%
  dplyr::summarise(acf1 = acf_at_lag(res, LAG_ACF), .groups="drop") %>%
  dplyr::group_by(rho, Metode) %>%
  dplyr::summarise(Mean_ACF1 = mean(acf1, na.rm=TRUE),
                   SD_ACF1   = sd(acf1, na.rm=TRUE),
                   .groups="drop") %>%
  dplyr::arrange(rho, Metode)
write.csv(acf_tbl, file.path(OUT_DIR, "tabel_acf_lag1.csv"), row.names=FALSE)
p_acf1 <- ggplot(acf_tbl, aes(x=rho, y=Mean_ACF1, group=Metode,
color=Metode)) +
  geom_line(linewidth=0.9) + geom_point(size=2.7) +
  geom_errorbar(aes(ymin=Mean_ACF1-SD_ACF1,
ymax=Mean_ACF1+SD_ACF1), width=0.03) +
  labs(x=expression(rho), y="Mean ACF(1) residual (± SD)",
       title="Diagnostik Residual: Rata-rata ACF Lag-1 per Subjek") +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_ACF1_vs_rho.png"), p_acf1, width=7,
height=4, dpi=300)
lb_tbl <- longdat %>%
  dplyr::select(rho, repl, id, time, res_spline, res_Kernel) %>%
  tidyr::pivot_longer(cols=c(res_spline, res_Kernel),
                      names_to="MetodeRaw", values_to="res") %>%
  dplyr::mutate(Metode = ifelse(MetodeRaw=="res_spline","Spline
Truncated","Kernel")) %>%
  dplyr::group_by(rho, repl, Metode, id) %>%
  dplyr::arrange(time, .by_group=TRUE) %>%
  dplyr::summarise(p_lb = ljung_box_p(res, LAG_LB), .groups="drop") %>%
  dplyr::group_by(rho, Metode) %>%
  dplyr::summarise(
    Mean_pvalue = mean(p_lb, na.rm=TRUE),
```

```
    RejectRate_5pct = mean(p_lb < 0.05, na.rm=TRUE),
    .groups="drop"
  ) %>% dplyr::arrange(rho, Metode)
write.csv(lb_tbl, file.path(OUT_DIR, "tabel_ljungbox.csv"), row.names=FALSE)
p_lb <- ggplot(lb_tbl, aes(x=rho, y=RejectRate_5pct, group=Metode,
color=Metode)) +
  geom_line(linewidth=0.9) + geom_point(size=2.7) +
  scale_y_continuous(limits=c(0,1)) +
  labs(x=expression(rho), y="Proporsi penolakan H0 (α=0.05)",
       title="Uji Ljung–Box Residual per Subjek (lag 1): Reject Rate") +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_LjungBox_RejectRate_vs_rho.png"), p_lb,
width=7, height=4, dpi=300)
```

Lampiran 9. Kode Penerapan pada Data Longitudinal Riil Periode 2016–2024 (Estimasi, Evaluasi, dan Diagnostik)

```
## PENERAPAN PADA DATA RIIL (2016–2024)
rm(list = ls())
if (!exists("theme_skripsi")) {
  theme_skripsi <- function(base_size = 12) {
    ggplot2::theme_bw(base_size = base_size) +
      ggplot2::theme(
        panel.grid.major = ggplot2::element_line(linewidth = 0.25, color = "grey85"),
        panel.grid.minor = ggplot2::element_line(linewidth = 0.15, color = "grey92"),
        legend.position = "bottom",
        legend.title = ggplot2::element_blank(),
        plot.title = ggplot2::element_text(face = "bold")
      )
  }
}
## ==========================
DATA_DIR  <- "D:/Document/SKRIPSI/Data/Data Hasil"
DATA_FILE <- "data P0.xlsx"
setwd(DATA_DIR)
cat("Working directory:", getwd(), "\n")
cat("Files in folder:\n")
print(list.files())
stopifnot(file.exists(DATA_FILE))
## ==========================
pkgs <- c("readxl", "readr", "dplyr", "tidyr", "ggplot2", "rlang")
need <- pkgs[!pkgs %in% installed.packages()[, "Package"]]
if (length(need) > 0) install.packages(need)
invisible(lapply(pkgs, library, character.only = TRUE))
```

```
## ==========================
COL_SPL  <- "#2E86DE"  # Spline (biru)
COL_KER  <- "#E74C3C"  # Kernel (merah)
COL_MEAN <- "#2ECC71"  # Mean aktual (hijau)
COL_PT   <- "#7F8C8D"  # titik data (abu gelap)
COL_THIN <- "#9B59B6"  # garis provinsi (ungu muda)
COL_GRID <- "#BDC3C7"  # grid/garis bantu
## ===========BACA DATA=============
raw <- readxl::read_excel(DATA_FILE)
find_col <- function(df, patterns) {
 nms <- names(df)
 for (p in patterns) {
  idx <- which(grepl(p, nms, ignore.case = TRUE))
  if (length(idx) > 0) return(nms[idx[1]])
 }
 NA_character_
}
prov_col <- find_col(raw, c("^prov$", "provinsi", "province", "nama.*prov",
"wilayah"))
year_col <- find_col(raw, c("^year$", "tahun", "thn", "periode"))
y_col    <- find_col(raw, c("^y$", "\\bp0\\b", "p0", "persen.*miskin",
"penduduk.*miskin", "poverty"))
x1_col   <- find_col(raw, c("^x1$", "x_?1", "pred1", "cov1"))
x2_col   <- find_col(raw, c("^x2$", "x_?2", "pred2", "cov2"))
if (is.na(prov_col) || is.na(year_col) || is.na(y_col)) {
 cat("\nNAMA KOLOM DI EXCEL:\n"); print(names(raw))
 stop("\nKolom wajib tidak ditemukan. Minimal harus ada: Provinsi, Tahun, dan
P0/y.\n")
}
dat <- raw %>%
 transmute(
  prov = as.character(.data[[prov_col]]),
```

```
    year = as.integer(.data[[year_col]]),
    y    = suppressWarnings(as.numeric(.data[[y_col]])),
    x1   = if (!is.na(x1_col)) suppressWarnings(as.numeric(.data[[x1_col]])) else NA_real_,
    x2   = if (!is.na(x2_col)) suppressWarnings(as.numeric(.data[[x2_col]])) else NA_real_
  ) %>%
  filter(!is.na(prov), !is.na(year), !is.na(y)) %>%
  filter(year >= 2016, year <= 2024) %>%
  arrange(prov, year)
stopifnot(all(c("prov","year","y") %in% names(dat)))
cat("\nRingkas data (2016–2024):\n")
cat("- N obs:", nrow(dat), "\n")
cat("- N prov:", dplyr::n_distinct(dat$prov), "\n")
cat("- Tahun:", paste(range(dat$year), collapse="–"), "\n")
## index time & u (0..1)
ymin <- min(dat$year)
ymax <- max(dat$year)
dat <- dat %>%
  mutate(
    time = year - ymin + 1L,
    u    = (year - ymin) / (ymax - ymin)
  )
## ===========FE (alpha_i) via within==============
alpha_tbl <- dat %>%
  group_by(prov) %>%
  summarise(alpha_i = mean(y, na.rm = TRUE), .groups="drop")
dat <- dat %>%
  left_join(alpha_tbl, by="prov") %>%
  mutate(y_within = y - alpha_i)
```

```
## ============METRIK & DIAGNOSTIK==============
calc_metrics <- function(y, yhat) {
  y <- as.numeric(y); yhat <- as.numeric(yhat)
  ok <- is.finite(y) & is.finite(yhat)
  y <- y[ok]; yhat <- yhat[ok]
  sse <- sum((y - yhat)^2)
  sst <- sum((y - mean(y))^2)
  mse <- mean((y - yhat)^2)
  rmse <- sqrt(mse)
  mae <- mean(abs(y - yhat))
  r2 <- ifelse(sst > 0, 1 - sse/sst, NA_real_)
  list(mse=mse, rmse=rmse, mae=mae, r2=r2)
}
dw_stat <- function(e) {
  e <- as.numeric(e)
  if (length(e) < 2) return(NA_real_)
  den <- sum(e^2)
  if (!is.finite(den) || den <= 1e-12) return(NA_real_)
  sum(diff(e)^2) / den
}
acf_at_lag <- function(x, lag_k=1) {
  x <- as.numeric(x)
  if (length(x) <= lag_k) return(NA_real_)
  if (sd(x, na.rm=TRUE) == 0) return(NA_real_)
  as.numeric(stats::acf(x, plot=FALSE, lag.max=lag_k)$acf[lag_k+1])
}
ljung_box_p <- function(x, lag_k=1) {
  x <- as.numeric(x)
  if (length(x) <= lag_k + 1) return(NA_real_)
  if (sd(x, na.rm=TRUE) == 0) return(NA_real_)
  out <- try(stats::Box.test(x, lag=lag_k, type="Ljung-Box"), silent=TRUE)
```

```
  if (inherits(out,"try-error")) return(NA_real_)
  as.numeric(out$p.value)
}
## ==========SPLINE TRUNCATED (linear + hinge) + GCV==========
build_trunc_basis <- function(u, knots) {
  u <- as.numeric(u)
  X <- cbind(1, u)
  if (length(knots) > 0) {
    for (k in knots) X <- cbind(X, pmax(0, u - k))
  }
  colnames(X) <- c("Intercept","u",
                   if (length(knots)>0) paste0("tp_", seq_along(knots)) else NULL)
  X
}
fit_spline_gcv <- function(u, y, K_grid = 1:6) {
  u <- as.numeric(u); y <- as.numeric(y)
  best <- list(gcv=Inf, K=NA_integer_, knots=numeric(0), beta=NULL)
  n <- length(y)
  for (K in K_grid) {
    knots <- if (K > 0) as.numeric(stats::quantile(u, probs = seq(1, K)/(K+1), type=7)) else numeric(0)
    X <- build_trunc_basis(u, knots)
    XtX <- crossprod(X)
    if (qr(XtX)$rank < ncol(X)) next
    beta <- solve(XtX, crossprod(X, y))
    yhat <- as.numeric(X %*% beta)
    sse  <- sum((y - yhat)^2)
    p <- ncol(X)
    gcv <- (sse / n) / ((1 - p/n)^2)
```

```
    if (is.finite(gcv) && gcv < best$gcv) {
      best <- list(gcv=gcv, K=K, knots=knots, beta=beta)
    }
  }
  best
}
predict_spline <- function(u_new, fit) {
  Xn <- build_trunc_basis(u_new, fit$knots)
  as.numeric(Xn %*% fit$beta)
}
## ==========KERNEL NW (Gaussian) + bandwidth==========
gaussK <- function(z) exp(-0.5*z^2) / sqrt(2*pi)
nw_predict <- function(u_train, y_train, u_new, h) {
  u_train <- as.numeric(u_train); y_train <- as.numeric(y_train); u_new <-
as.numeric(u_new)
  h <- as.numeric(h)
  sapply(u_new, function(uu) {
    w <- gaussK((uu - u_train)/h)
    sw <- sum(w)
    if (!is.finite(sw) || sw <= 0) return(NA_real_)
    sum(w * y_train) / sw
  })
}
select_h_loocv <- function(u, y, h_grid) {
  u <- as.numeric(u); y <- as.numeric(y)
  n <- length(y)
  best <- list(cv=Inf, h=NA_real_)
  D <- outer(u, u, "-")
  for (h in h_grid) {
    W <- gaussK(D / h)
    diag(W) <- 0
```

```
    denom <- rowSums(W)
    numer <- rowSums(W * matrix(y, n, n, byrow=TRUE))
    yhat <- numer / denom
    ok <- is.finite(yhat) & is.finite(denom) & denom > 0
    if (!any(ok)) next
    cv <- mean((y[ok] - yhat[ok])^2)
    if (is.finite(cv) && cv < best$cv) best <- list(cv=cv, h=h)
  }
  best
}
## ==========FIT FULL DATA (within)==============
u_all  <- dat$u
yw_all <- dat$y_within
fit_spl <- fit_spline_gcv(u_all, yw_all, K_grid = 1:6)
ywhat_spline <- predict_spline(u_all, fit_spl)
h_grid <- seq(0.03, 0.35, by=0.01)
fit_h  <- select_h_loocv(u_all, yw_all, h_grid)
ywhat_Kernel <- nw_predict(u_all, yw_all, u_all, fit_h$h)
yhat_spline <- ywhat_spline + dat$alpha_i
yhat_Kernel <- ywhat_Kernel + dat$alpha_i
dat <- dat %>%
  mutate(
    ywhat_spline = ywhat_spline,
    ywhat_Kernel = ywhat_Kernel,
    yhat_spline  = yhat_spline,
    yhat_Kernel  = yhat_Kernel,
    res_spline   = y_within - ywhat_spline,
    res_Kernel   = y_within - ywhat_Kernel
  )
## ===========METRIK + TUNING=============
m_within_spl <- calc_metrics(dat$y_within, dat$ywhat_spline)
```

```
m_within_ker <- calc_metrics(dat$y_within, dat$ywhat_Kernel)
m_y_spl <- calc_metrics(dat$y, dat$yhat_spline)
m_y_ker <- calc_metrics(dat$y, dat$yhat_Kernel)
metrics_out <- data.frame(
  Metode = c("Spline Truncated", "Kernel"),
  mse_within  = c(m_within_spl$mse,  m_within_ker$mse),
  rmse_within = c(m_within_spl$rmse, m_within_ker$rmse),
  mae_within  = c(m_within_spl$mae,  m_within_ker$mae),
  r2_within   = c(m_within_spl$r2,   m_within_ker$r2),
  mse_y       = c(m_y_spl$mse,       m_y_ker$mse),
  rmse_y      = c(m_y_spl$rmse,      m_y_ker$rmse),
  mae_y       = c(m_y_spl$mae,       m_y_ker$mae),
  r2_y        = c(m_y_spl$r2,        m_y_ker$r2)
)
tuning_out <- data.frame(
  spline_K = fit_spl$K,
  spline_knots = paste(round(fit_spl$knots, 5), collapse=", "),
  spline_gcv = fit_spl$gcv,
  Kernel_h = fit_h$h,
  Kernel_loocv_mse = fit_h$cv
)
## ==========RINGKAS DESKRIPTIF DATA RIIL==============
## Ringkasan statistik per tahun (antar provinsi)
year_stats <- dat %>%
  group_by(year) %>%
  summarise(
    N_prov = n_distinct(prov),
    mean_y = mean(y, na.rm = TRUE),
    median_y = median(y, na.rm = TRUE),
    min_y = min(y, na.rm = TRUE),
    max_y = max(y, na.rm = TRUE),
```

```
    sd_y  = sd(y, na.rm = TRUE),
    q25_y = as.numeric(quantile(y, 0.25, na.rm = TRUE, type = 7)),
    q75_y = as.numeric(quantile(y, 0.75, na.rm = TRUE, type = 7)),
    iqr_y = q75_y - q25_y,
    .groups = "drop"
  )
## SD keseluruhan (untuk heterogenitas skala-bebas)
real_sd_overall <- sd(dat$y, na.rm = TRUE)
year_stats <- year_stats %>% dplyr::arrange(year)
write.csv(year_stats, file.path(OUT_DIR, "tabel_ringkasan_stat_per_tahun.csv"),
row.names = FALSE)
## Ringkasan antar provinsi: ekstrem (mean tinggi/rendah) & stabilitas (sd
sepanjang waktu)
prov_stats <- dat %>%
  group_by(prov) %>%
  summarise(
    mean_y = mean(y, na.rm = TRUE),
    sd_y_time = sd(y, na.rm = TRUE),
    min_y = min(y, na.rm = TRUE),
    max_y = max(y, na.rm = TRUE),
    range_y = max_y - min_y,
    .groups = "drop"
  ) %>% arrange(desc(mean_y))
write.csv(prov_stats, file.path(OUT_DIR,
"tabel_ringkasan_stat_per_provinsi.csv"), row.names = FALSE)
prov_top5 <- prov_stats %>% slice_max(order_by = mean_y, n = 5)
prov_bot5 <- prov_stats %>% slice_min(order_by = mean_y, n = 5)
prov_stable5 <- prov_stats %>% slice_min(order_by = sd_y_time, n = 5)
prov_unstable5 <- prov_stats %>% slice_max(order_by = sd_y_time, n = 5)
write.csv(bind_rows(
  prov_top5 %>% mutate(kategori="Tertinggi (mean)"),
  prov_bot5 %>% mutate(kategori="Terendah (mean)"),
```

```
  prov_stable5 %>% mutate(kategori="Paling Stabil (SD terkecil)"),
  prov_unstable5 %>% mutate(kategori="Paling Fluktuatif (SD terbesar)")
), file.path(OUT_DIR, "tabel_prov_ekstrem_dan_stabilitas.csv"), row.names =
FALSE)
## Plot 1: Mean nasional per tahun + band min-max
p_mean_band <- ggplot(year_stats, aes(x = year, y = mean_y)) +
  geom_ribbon(aes(ymin = min_y, ymax = max_y), alpha = 0.18) +
  geom_line(linewidth = 1.15) +
  geom_point(size = 2.6) +
  labs(
    x = "Tahun",
    y = "P0 (mean antar provinsi)",
    title = "Rata-rata Nasional per Tahun dengan Band Min–Max (antar Provinsi)",
    subtitle = "Band menunjukkan heterogenitas antar provinsi pada setiap tahun"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_MeanNasional_Band_MinMax.png"),
       p_mean_band, width = 9, height = 4.8, dpi = 300)
## Plot 2: Boxplot per tahun (heterogenitas antar provinsi)
p_box_year <- ggplot(dat, aes(x = factor(year), y = y)) +
  geom_boxplot(alpha = 0.65, outlier_alpha = 0.25) +
  labs(
    x = "Tahun",
    y = "P0 (antar provinsi)",
    title = "Boxplot P0 per Tahun (Heterogenitas Antar Provinsi)"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_Boxplot_P0_PerTahun.png"),
       p_box_year, width = 9, height = 4.8, dpi = 300)
## ==========SIMULASI vs DATA RIIL===============
## ---------- Helper metrik ----------
```

```
rho_pooled <- function(df, id_col, time_col, x_col) {
  id_col   <- rlang::ensym(id_col)
  time_col <- rlang::ensym(time_col)
  x_col    <- rlang::ensym(x_col)
  tmp <- df %>%
    arrange(!!id_col, !!time_col) %>%
    group_by(!!id_col) %>%
    mutate(x_lag = dplyr::lag(!!x_col)) %>%
    ungroup() %>%
    filter(is.finite(x_lag), is.finite(.data[[rlang::as_name(x_col)]]))
  if (nrow(tmp) == 0) return(NA_real_)
  x_now <- tmp[[rlang::as_name(x_col)]]
  num <- sum(tmp$x_lag * x_now, na.rm = TRUE)
  den <- sum(tmp$x_lag^2, na.rm = TRUE)
  if (!is.finite(den) || den <= 1e-12) return(NA_real_)
  num / den
}
hetero_scale_free <- function(df, unit_col, time_col, y_col) {
  # heterogenitas skala-bebas: mean_t [ SD antar unit pada t / SD keseluruhan ]
  unit_col <- rlang::ensym(unit_col)
  time_col <- rlang::ensym(time_col)
  y_col    <- rlang::ensym(y_col)
  y_all <- df[[rlang::as_name(y_col)]]
  sd_all <- sd(y_all, na.rm = TRUE)
  if (!is.finite(sd_all) || sd_all <= 1e-12) return(NA_real_)
  sd_by_time <- df %>%
    group_by(!!time_col) %>%
    summarise(sd_unit = sd(.data[[rlang::as_name(y_col)]], na.rm = TRUE),
.groups = "drop")
  mean(sd_by_time$sd_unit / sd_all, na.rm = TRUE)
}
```

```
boot_ci_cluster <- function(df, cluster_col, stat_fun, B = 2000, conf = 0.95, seed
= 2026) {
  set.seed(seed)
  cluster_col <- rlang::ensym(cluster_col)
  ids <- unique(df[[rlang::as_name(cluster_col)]])
  ids <- ids[!is.na(ids)]
  if (length(ids) == 0) return(c(mean = NA_real_, lo = NA_real_, hi = NA_real_))
  alpha <- (1 - conf) / 2
  boot <- replicate(B, {
    samp <- sample(ids, size = length(ids), replace = TRUE)
    df_b <- dplyr::bind_rows(lapply(samp, function(v) df %>%
filter(.data[[rlang::as_name(cluster_col)]] == v)))
    stat_fun(df_b)
  })
  c(
    mean = mean(boot, na.rm = TRUE),
    lo   = as.numeric(quantile(boot, probs = alpha,     na.rm = TRUE, type = 7)),
    hi   = as.numeric(quantile(boot, probs = 1 - alpha, na.rm = TRUE, type = 7))
  )
}
## ---------- (1) DATA RIIL: ρ̂ pooled pada y_within + heterogenitas skala-bebas +
CI ----------
real_rho_ci <- boot_ci_cluster(
  dat, cluster_col = prov,
  stat_fun = function(d) rho_pooled(d, prov, time, y_within),
  B = 2000, seed = 2026
)
real_het_ci <- boot_ci_cluster(
  dat, cluster_col = prov,
  stat_fun = function(d) hetero_scale_free(d, prov, year, y),
  B = 2000, seed = 2026
```

```
)
## Indikasi shock sekitar 2020–2021: perubahan mean nasional antar tahun (tetap dipertahankan)
deltas <- year_stats %>%
  arrange(year) %>%
  mutate(delta_mean = c(NA_real_, diff(mean_y)),
         abs_delta  = abs(delta_mean))
delta_2019_2020 <- deltas$delta_mean[deltas$year == 2020]
delta_2020_2021 <- deltas$delta_mean[deltas$year == 2021]
thr_shock <- as.numeric(quantile(deltas$abs_delta, 0.75, na.rm = TRUE, type = 7))
shock_flag <- any(c(abs(delta_2019_2020), abs(delta_2020_2021)) >= thr_shock, na.rm = TRUE)
kar_riil <- data.frame(
  Sumber = "Data riil (2016–2024)",
  rho = NA_real_,
  N_unit = n_distinct(dat$prov),
  T_time = n_distinct(dat$time),
  RhoHat_pooled_mean = real_rho_ci["mean"],
  RhoHat_pooled_lo   = real_rho_ci["lo"],
  RhoHat_pooled_hi   = real_rho_ci["hi"],
  HetZ_mean = real_het_ci["mean"],
  HetZ_lo   = real_het_ci["lo"],
  HetZ_hi   = real_het_ci["hi"],
  MaxAbsDeltaMean = max(deltas$abs_delta, na.rm = TRUE),
  Time_of_MaxAbsDelta = deltas$year[which.max(deltas$abs_delta)],
  Shock_2020_2021 = ifelse(shock_flag, "YA", "TIDAK"),
  stringsAsFactors = FALSE
)
## ---------- (2) SIMULASI: hitung per replikasi lalu ringkas per rho + CI ----------
SIM_LONG_FILE      <- file.path(DATA_DIR, "hasil_simulasi_longdata.csv")
```

```
SIM_LONG_FULL_FILE <- file.path(DATA_DIR,
"hasil_simulasi_longdata_full.csv")
kar_sim <- NULL
if (file.exists(SIM_LONG_FULL_FILE) || file.exists(SIM_LONG_FILE)) {
  sim_long <- if (file.exists(SIM_LONG_FULL_FILE)) {
    readr::read_csv(SIM_LONG_FULL_FILE, show_col_types = FALSE)
  } else {
    readr::read_csv(SIM_LONG_FILE, show_col_types = FALSE)
  }
  if ("rho" %in% names(sim_long)) sim_long$rho <- as.numeric(sim_long$rho)
  need_cols <- c("rho","repl","id","time","y","eps")
  miss <- setdiff(need_cols, names(sim_long))
  if (length(miss) > 0) {
    stop("Bridging ABC membutuhkan kolom simulasi: ", paste(miss, collapse=",
"),
         ". Pastikan menghasilkan hasil_simulasi_longdata_full.csv (punya eps).")
  }
  ## Metrik per replikasi:
  ## - Autokorelasi: ρ̂ pooled dihitung dari eps (galat AR(1) murni)
  ## - Heterogenitas: hetero skala-bebas dihitung dari y (antar id per time / sd
keseluruhan)
  sim_rep <- sim_long %>%
    group_by(rho, repl) %>%
    summarise(
      RhoHat_pooled = rho_pooled(dplyr::pick(everything()), id, time, eps),
      HetZ          = hetero_scale_free(dplyr::pick(everything()), id, time, y),
      N_unit        = n_distinct(id),
      T_time        = n_distinct(time),
      .groups = "drop"
    )
  ## Ringkas per rho + CI (percentile antar replikasi)
  sim_sum <- sim_rep %>%
```

```
  group_by(rho) %>%
  summarise(
    N_unit = round(mean(N_unit, na.rm = TRUE)),
    T_time = round(mean(T_time, na.rm = TRUE)),
    RhoHat_pooled_mean = mean(RhoHat_pooled, na.rm = TRUE),
    RhoHat_pooled_lo   = quantile(RhoHat_pooled, 0.025, na.rm = TRUE, type =
7),
    RhoHat_pooled_hi   = quantile(RhoHat_pooled, 0.975, na.rm = TRUE, type =
7),
    HetZ_mean = mean(HetZ, na.rm = TRUE),
    HetZ_lo   = quantile(HetZ, 0.025, na.rm = TRUE, type = 7),
    HetZ_hi   = quantile(HetZ, 0.975, na.rm = TRUE, type = 7),
    .groups = "drop"
  ) %>%
  arrange(rho)
trend_sum <- sim_long %>%
  group_by(rho, time) %>%
  summarise(mean_y = mean(y, na.rm = TRUE), .groups = "drop") %>%
  arrange(rho, time) %>%
  group_by(rho) %>%
  mutate(delta_mean = c(NA_real_, diff(mean_y))) %>%
  summarise(
    MaxAbsDeltaMean = max(abs(delta_mean), na.rm = TRUE),
    Time_of_MaxAbsDelta = time[which.max(abs(delta_mean))],
    .groups = "drop"
  )
kar_sim <- sim_sum %>%
  left_join(trend_sum, by = "rho") %>%
  mutate(
    Sumber = paste0("Simulasi (rho=", rho, ")"),
    Shock_2020_2021 = NA_character_
```

```
  ) %>%
  select(Sumber, rho, N_unit, T_time,
         RhoHat_pooled_mean, RhoHat_pooled_lo, RhoHat_pooled_hi,
         HetZ_mean, HetZ_lo, HetZ_hi,
         MaxAbsDeltaMean, Time_of_MaxAbsDelta, Shock_2020_2021) %>%
  arrange(rho)
}
## ---------- (3) Gabungkan + Δ + D(ρ) ----------
if (!is.null(kar_sim)) {


 bridging_tbl <- bind_rows(
  kar_sim,
  kar_riil
 )
 real_rho_hat <- kar_riil$RhoHat_pooled_mean
 real_het_z   <- kar_riil$HetZ_mean
 bridging_tbl <- bridging_tbl %>%
  mutate(
   Delta_Rho = ifelse(is.na(rho), NA_real_, abs(RhoHat_pooled_mean -
real_rho_hat)),
   Delta_Het = ifelse(is.na(rho), NA_real_, abs(HetZ_mean - real_het_z))
  )
 ## jarak gabungan terstandar D(ρ)
 s_rho <- sd(bridging_tbl$RhoHat_pooled_mean[!is.na(bridging_tbl$rho)], na.rm
= TRUE)
 s_het <- sd(bridging_tbl$HetZ_mean[!is.na(bridging_tbl$rho)], na.rm = TRUE)
 valid_rho_scale <- is.finite(s_rho) && s_rho > 1e-12
 valid_het_scale <- is.finite(s_het) && s_het > 1e-12
 bridging_tbl <- bridging_tbl %>%
  mutate(
   D_rho = ifelse(!is.na(rho) & valid_rho_scale, (Delta_Rho / s_rho)^2,
NA_real_),
```

```
    D_het = ifelse(!is.na(rho) & valid_het_scale, (Delta_Het / s_het)^2,
NA_real_),
    D     = ifelse(!is.na(rho), D_rho + D_het, NA_real_)
  )
 best_row <- bridging_tbl %>% filter(!is.na(rho)) %>% arrange(D) %>% slice(1)
 cat("\n[BRIDGING ABC] Skenario simulasi terdekat (min D): rho =",
best_row$rho, "\n")
} else {
 bridging_tbl <- kar_riil
 cat("\n[BRIDGING ABC] Data simulasi tidak ditemukan → hanya menyimpan
karakteristik data riil.\n")
}
## Simpan tabel
write.csv(bridging_tbl, file.path(OUT_DIR,
"tabel_bridging_simulasi_vs_riil.csv"), row.names = FALSE)
## ---------- (4) Plot bridging dengan interval (C) ----------
if (!is.null(kar_sim)) {
 ## Plot 1: Autokorelasi (ρ̂ pooled)
 p_bridge_rho <- ggplot(bridging_tbl %>% filter(!is.na(rho)),
                        aes(x = rho, y = RhoHat_pooled_mean, group = 1)) +
   geom_line(linewidth = 1.0) +
   geom_point(size = 2.8) +
   geom_errorbar(aes(ymin = RhoHat_pooled_lo, ymax = RhoHat_pooled_hi),
width = 0.02) +
   geom_hline(yintercept = kar_riil$RhoHat_pooled_mean, linetype = 2) +
   annotate("rect",
            xmin = -Inf, xmax = Inf,
            ymin = kar_riil$RhoHat_pooled_lo, ymax = kar_riil$RhoHat_pooled_hi,
            alpha = 0.10) +
   labs(
     x = expression(rho),
     y = expression(hat(rho)[pooled]),
     title = "Bridging Autokorelasi (ρ̂ pooled): Simulasi vs Data Riil",
```

```
    subtitle = "Titik+error bar = simulasi (CI antar replikasi); garis putus-putus+pita = data riil (CI bootstrap antar provinsi)"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_Bridging_RhoHatPooled_Sim_vs_Riil.png"),
       p_bridge_rho, width = 7.5, height = 4.6, dpi = 300)
## Plot 2: Heterogenitas skala-bebas (HetZ)
p_bridge_hetz <- ggplot(bridging_tbl %>% filter(!is.na(rho)),
                        aes(x = rho, y = HetZ_mean, group = 1)) +
  geom_line(linewidth = 1.0, color = "#1f77b4") +
  geom_point(size = 2.8, color = "#1f77b4") +
  geom_errorbar(aes(ymin = HetZ_lo, ymax = HetZ_hi), width = 0.02, color = "#1f77b4") +
  geom_hline(yintercept = kar_riil$HetZ_mean, linetype = 2, color = "black") +
  annotate("rect",
           xmin = -Inf, xmax = Inf,
           ymin = kar_riil$HetZ_lo, ymax = kar_riil$HetZ_hi,
           alpha = 0.10) +
  labs(
    x = expression(rho),
    y = "Heterogenitas skala-bebas (Mean SD antar unit / SD keseluruhan)",
    title = "Bridging Heterogenitas (Skala-bebas): Simulasi vs Data Riil",
    subtitle = "Titik+error bar = simulasi (CI antar replikasi); garis putus-putus+pita = data riil (CI bootstrap antar provinsi)"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_Bridging_HeterogenitasZ_Sim_vs_Riil.png"),
       p_bridge_hetz, width = 7.5, height = 4.6, dpi = 300)
}
write.csv(
  dat %>% select(prov, year, time, u, y, x1, x2, alpha_i, y_within,
```

```
            ywhat_spline, ywhat_Kernel, yhat_spline, yhat_Kernel,
            res_spline, res_Kernel),
  file.path(OUT_DIR, "hasil_riil_longdata.csv"),
  row.names = FALSE
)
write.csv(metrics_out, file.path(OUT_DIR, "hasil_riil_metrics.csv"), row.names =
FALSE)
write.csv(tuning_out,  file.path(OUT_DIR, "hasil_riil_tuning.csv"),  row.names =
FALSE)
cat("\n=== TUNING TERPILIH ===\n"); print(tuning_out)
cat("\n=== METRIK IN-SAMPLE ===\n"); print(metrics_out)
## ============ OUTPUT FOLDER============
OUT_DIR <- file.path(DATA_DIR, "output_fase4")
if (!dir.exists(OUT_DIR)) dir.create(OUT_DIR, recursive = TRUE)
## ============POLA DATA AKTUAL==============
mean_actual <- dat %>%
  group_by(year) %>%
  summarise(mean_y = mean(y, na.rm=TRUE), .groups="drop") %>%
  arrange(year)
p_actual <- ggplot() +
  geom_line(data = dat, aes(x=year, y=y, group=prov),
            color = COL_THIN, alpha=0.25, linewidth=0.7) +
  geom_point(data = dat, aes(x=year, y=y),
             color = COL_PT, alpha=0.25, size=1.1) +
  geom_line(data = mean_actual, aes(x=year, y=mean_y),
            color = COL_MEAN, linewidth=1.3) +
  geom_point(data = mean_actual, aes(x=year, y=mean_y),
             color = COL_MEAN, size=2.6) +
  labs(
    x="Tahun", y="P0 (Persentase Penduduk Miskin)",
    title="Pola Persentase Penduduk Miskin terhadap Waktu (2016–2024) – Data
Aktual",
```

```
    subtitle="Garis tipis: tiap provinsi | Garis tebal: rata-rata nasional (mean antar provinsi)"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_Pola_P0_Aktual_2016_2024.png"),
       p_actual, width = 9, height = 4.8, dpi = 300)
## ============GAMBAR FIT (AKTUAL + KURVA METODE)==============
mean_by_year <- dat %>%
  group_by(year) %>%
  summarise(
    y_mean = mean(y, na.rm=TRUE),
    yhat_spline_mean = mean(yhat_spline, na.rm=TRUE),
    yhat_Kernel_mean = mean(yhat_Kernel, na.rm=TRUE),
    .groups="drop"
  ) %>% arrange(year)
fit_long <- mean_by_year %>%
  tidyr::pivot_longer(cols = c(y_mean, yhat_spline_mean, yhat_Kernel_mean),
                      names_to = "kurva", values_to = "val") %>%
  mutate(
    kurva_label = dplyr::case_when(
      kurva == "y_mean" ~ "Rata-rata y",
      kurva == "yhat_spline_mean" ~ "Spline Truncated",
      TRUE ~ "Kernel"
    )
  )
yr  <- range(fit_long$val, na.rm = TRUE)
pad <- 0.6
ymin_zoom <- floor((yr[1] - pad) * 2) / 2
ymax_zoom <- ceiling((yr[2] + pad) * 2) / 2
```

```
p_fit <- ggplot() +
  geom_point(data = dat, aes(x=year, y=y),
             color=COL_PT, alpha=0.18, size=1.4) +
  geom_line(data = fit_long, aes(x=year, y=val, color=kurva_label),
            linewidth=1.25) +
  scale_color_manual(values = c(
    "Rata-rata y" = COL_MEAN,
    "Spline Truncated" = COL_SPL,
    "Kernel" = COL_KER
  )) +
  scale_y_continuous(
    breaks = seq(ymin_zoom, ymax_zoom, by = 0.5),
    minor_breaks = seq(ymin_zoom, ymax_zoom, by = 0.25)
  ) +
  coord_cartesian(ylim = c(ymin_zoom, ymax_zoom)) +
  labs(
    x="Tahun", y="P0", color="Kurva",
    title="Penerapan Data Riil 2016–2024: Fit Spline Truncated vs Kernel (Mean
per Tahun)"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_Fit_DataRiil_ZoomY.png"),
       p_fit, width = 9, height = 4.8, dpi = 300)
## ============ACF/PACF RESIDUAL==============
avg_acf_by_prov <- function(df, res_col, lag.max) {
  ids <- sort(unique(df$prov))
  mat <- sapply(ids, function(pv) {
    s <- df[df$prov == pv, ]
    s <- s[order(s$time), , drop=FALSE]
    n <- nrow(s)
    out <- rep(NA_real_, lag.max + 1)
```

```
    if (n >= 2) {
      lm_i <- min(lag.max, n - 1)
      acv <- as.numeric(stats::acf(s[[res_col]], plot=FALSE, lag.max=lm_i)$acf)
      out[1:(lm_i+1)] <- acv
    }
    out
  })
  if (is.vector(mat)) mat <- matrix(mat, ncol=1)
  data.frame(
    lag = 0:lag.max,
    val = rowMeans(mat, na.rm=TRUE),
    se  = apply(mat, 1, sd, na.rm=TRUE) / sqrt(ncol(mat))
  )
}
avg_pacf_by_prov <- function(df, res_col, lag.max) {
  ids <- sort(unique(df$prov))
  mat <- sapply(ids, function(pv) {
    s <- df[df$prov == pv, ]
    s <- s[order(s$time), , drop=FALSE]
    n <- nrow(s)
    out <- rep(NA_real_, lag.max)
    if (n >= 3 && lag.max >= 1) {
      lm_i <- min(lag.max, n - 1)
      pcv <- as.numeric(stats::pacf(s[[res_col]], plot=FALSE, lag.max=lm_i)$acf)
      out[1:lm_i] <- pcv
    }
    out
  })
  if (is.vector(mat)) mat <- matrix(mat, ncol=1)
  data.frame(
    lag = 1:lag.max,
```

```
    val = rowMeans(mat, na.rm=TRUE),
    se  = apply(mat, 1, sd, na.rm=TRUE) / sqrt(ncol(mat))
  )
}
lag.max <- max(1, min(4, max(dat$time) - 1))
acf_spl  <- avg_acf_by_prov(dat, "res_spline", lag.max)
pacf_spl <- avg_pacf_by_prov(dat, "res_spline", lag.max)
acf_ker  <- avg_acf_by_prov(dat, "res_Kernel", lag.max)
pacf_ker <- avg_pacf_by_prov(dat, "res_Kernel", lag.max)
plot_acf_pacf_colored <- function(acf_df, pacf_df, main_prefix, col_bar) {
  par(mfrow=c(2,1), mar=c(4,4,3,1))
  plot(acf_df$lag, acf_df$val, type="h", lwd=3, col=col_bar,
       main=paste0("ACF Residual (rata-rata per prov) – ", main_prefix),
       xlab="Lag", ylab="ACF")
  abline(h=0, col=COL_GRID)
  lines(acf_df$lag, acf_df$val + 1.96*acf_df$se, lty=2, col=COL_GRID)
  lines(acf_df$lag, acf_df$val - 1.96*acf_df$se, lty=2, col=COL_GRID)
  plot(pacf_df$lag, pacf_df$val, type="h", lwd=3, col=col_bar,
       main=paste0("PACF Residual (rata-rata per prov) – ", main_prefix),
       xlab="Lag", ylab="PACF")
  abline(h=0, col=COL_GRID)
  lines(pacf_df$lag, pacf_df$val + 1.96*pacf_df$se, lty=2, col=COL_GRID)
  lines(pacf_df$lag, pacf_df$val - 1.96*pacf_df$se, lty=2, col=COL_GRID)
}
png(file.path(OUT_DIR, "Gambar_ACF_PACF_Spline_DataRiil.png"),
width=900, height=700)
plot_acf_pacf_colored(acf_spl, pacf_spl, "Spline Truncated", COL_SPL)
dev.off()
png(file.path(OUT_DIR, "Gambar_ACF_PACF_Kernel_DataRiil.png"),
width=900, height=700)
plot_acf_pacf_colored(acf_ker, pacf_ker, "Kernel", COL_KER)
```

```
dev.off()
## ===========ACF(1), Ljung–Box==============
LAG1 <- 1
diag_tbl <- dat %>%
  select(prov, time, res_spline, res_Kernel) %>%
  pivot_longer(cols=c(res_spline, res_Kernel),
               names_to="MetodeRaw", values_to="res") %>%
  mutate(Metode = ifelse(MetodeRaw=="res_spline","Spline
Truncated","Kernel")) %>%
  group_by(Metode, prov) %>%
  arrange(time, .by_group=TRUE) %>%
  summarise(
    acf1 = acf_at_lag(res, LAG1),
    p_lb = ljung_box_p(res, LAG1),
    .groups="drop"
  )
acf_tbl <- diag_tbl %>%
  group_by(Metode) %>%
  summarise(Mean_ACF1 = mean(acf1, na.rm=TRUE),
            SD_ACF1   = sd(acf1, na.rm=TRUE),
            .groups="drop")
lb_tbl <- diag_tbl %>%
  group_by(Metode) %>%
  summarise(Mean_pvalue = mean(p_lb, na.rm=TRUE),
            RejectRate_5pct = mean(p_lb < 0.05, na.rm=TRUE),
            .groups="drop")
write.csv(acf_tbl, file.path(OUT_DIR, "tabel_acf_lag1_residual.csv"),
row.names=FALSE)
write.csv(lb_tbl,  file.path(OUT_DIR, "tabel_ljungbox_residual.csv"),
row.names=FALSE)
meth_cols <- c("Spline Truncated"=COL_SPL, "Kernel"=COL_KER)
```

```
p_acf1 <- ggplot(acf_tbl, aes(x=Metode, y=Mean_ACF1, color=Metode)) +
  geom_point(size=3.2) +
  geom_errorbar(aes(ymin=Mean_ACF1-SD_ACF1,
ymax=Mean_ACF1+SD_ACF1), width=0.15) +
  scale_color_manual(values=meth_cols) +
  labs(x="", y="Mean ACF(1) residual (± SD)",
       title="Diagnostik Residual Data Riil: ACF Lag-1 (dirata-ratakan per
provinsi)") +
  theme_skripsi() +
  theme(legend.position="none")
ggsave(file.path(OUT_DIR, "Gambar_ACF1_Residual_DataRiil.png"),
       p_acf1, width=8, height=4.2, dpi=300)
p_lb <- ggplot(lb_tbl, aes(x=Metode, y=RejectRate_5pct, color=Metode)) +
  geom_point(size=3.2) +
  scale_y_continuous(limits=c(0,1)) +
  scale_color_manual(values=meth_cols) +
  labs(x="", y="Proporsi penolakan H0 (α=0.05)",
       title="Uji Ljung–Box Residual Data Riil (lag 1): Reject Rate") +
  theme_skripsi() +
  theme(legend.position="none")
ggsave(file.path(OUT_DIR, "Gambar_LjungBox_RejectRate_DataRiil.png"),
       p_lb, width=8, height=4.2, dpi=300)
## ===========KATEGORI AUTOKORELASI
(LOW/MED/HIGH)==============
acf_y_tbl <- dat %>%
  group_by(prov) %>%
  arrange(time, .by_group=TRUE) %>%
  summarise(acf1_y_within = acf_at_lag(y_within, 1), .groups="drop") %>%
  mutate(
    kategori_autokorelasi = case_when(
      abs(acf1_y_within) < 0.3 ~ "Rendah",
      abs(acf1_y_within) < 0.7 ~ "Sedang",
```

```
    TRUE ~ "Tinggi"
  )
 )
write.csv(acf_y_tbl, file.path(OUT_DIR, "tabel_kategori_autokorelasi_y.csv"),
row.names=FALSE)
cat_cols <- c("Rendah"="#27AE60", "Sedang"="#F1C40F",
"Tinggi"="#C0392B")
p_cat <- ggplot(acf_y_tbl %>% arrange(acf1_y_within) %>% mutate(prov =
factor(prov, levels=prov)),
            aes(x=prov, y=acf1_y_within, color=kategori_autokorelasi)) +
 geom_point(size=2.4) +
 geom_hline(yintercept=c(-0.7,-0.3,0.3,0.7), linetype=2, color=COL_GRID) +
 coord_flip() +
 scale_color_manual(values=cat_cols) +
 labs(
  x="Provinsi", y="ACF(1) dari y_within",
  title="Klasifikasi Tingkat Autokorelasi Data (per Provinsi) berdasarkan ACF(1)
y_within",
  subtitle="Batas: |ACF1|<0.3 (Rendah), 0.3–<0.7 (Sedang), ≥0.7 (Tinggi)",
  color="Kategori"
 ) +
 theme_skripsi()
ggsave(file.path(OUT_DIR, "Gambar_KategoriAutokorelasi_YWithin.png"),
    p_cat, width=9, height=10, dpi=300)
p_count <- ggplot(acf_y_tbl, aes(x=kategori_autokorelasi,
fill=kategori_autokorelasi)) +
 geom_bar() +
 scale_fill_manual(values=cat_cols) +
 labs(x="Kategori", y="Jumlah Provinsi",
    title="Jumlah Provinsi per Kategori Autokorelasi (berdasarkan ACF(1)
y_within)",
    fill="Kategori") +
 theme_skripsi()
```

```
ggsave(file.path(OUT_DIR, "Gambar_Count_KategoriAutokorelasi.png"),
       p_count, width=7.5, height=4.2, dpi=300)
metrik_per_group <- dat %>%
  left_join(acf_y_tbl %>% select(prov, kategori_autokorelasi), by="prov") %>%
  group_by(kategori_autokorelasi) %>%
  summarise(
    mse_within_spline = mean((y_within - ywhat_spline)^2, na.rm=TRUE),
    mse_within_Kernel = mean((y_within - ywhat_Kernel)^2, na.rm=TRUE),
    .groups="drop"
  )
write.csv(metrik_per_group, file.path(OUT_DIR,
"tabel_metrik_per_kategori.csv"), row.names=FALSE)
## ==========DISTRIBUSI ACF(1) y_within per provinsi==============
p_dist_acf1 <- ggplot(acf_y_tbl, aes(x = acf1_y_within)) +
  geom_histogram(aes(y = after_stat(density)), bins = 15,
                 fill = "#3498DB", color = "#1F2D3D", alpha = 0.55) +
  geom_density(color = "#E74C3C", linewidth = 1.15) +
  geom_vline(xintercept = c(-0.7, -0.3, 0.3, 0.7),
             linetype = 2, color = "#7F8C8D") +
  labs(
    x = "ACF(1) y_within (per provinsi)",
    y = "Density",
    title = "Distribusi ACF Lag-1 Data Riil per Provinsi",
    subtitle = "ACF(1) dihitung dari y_within (demean per provinsi). Garis putus-
putus = batas kategori |ACF1|"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR,
"Gambar_Distribusi_ACF1_YWithin_PerProvinsi.png"),
       p_dist_acf1, width = 8.5, height = 4.8, dpi = 300)
```

```
p_box_acf1 <- ggplot(acf_y_tbl, aes(x = "", y = acf1_y_within)) +
  geom_boxplot(fill = "#2ECC71", color = "#1F2D3D", alpha = 0.65, width =
0.25) +
  geom_jitter(width = 0.08, alpha = 0.35, color = "#34495E") +
  geom_hline(yintercept = c(-0.7, -0.3, 0.3, 0.7),
             linetype = 2, color = "#7F8C8D") +
  labs(
    x = "",
    y = "ACF(1) y_within (per provinsi)",
    title = "Ringkasan Distribusi ACF Lag-1 per Provinsi (Boxplot + Titik)"
  ) +
  theme_skripsi()
ggsave(file.path(OUT_DIR,
"Gambar_Boxplot_ACF1_YWithin_PerProvinsi.png"),
       p_box_acf1, width = 7.5, height = 4.8, dpi = 300)
```