SUPPLEMENTARY MATERIALS

Detection, Discrimination, Quantification and Classification of Disease-Linked Exhaled-Breath Metabolite Mixtures from Laser Back-Scattering Signatures

Prof. Rao Tatavarti, rtatavarti@gmail.com, 23 August 2026

Contents: S1 — run-averaged data tables (decimated to 1 Hz) for all seven data sets; S2 — complete regression model comparison tables; S3 — full Python implementation of the analysis pipeline; S4 — classification detail tables.

S1. Run-Averaged Data Tables (1 Hz decimation)

Full-rate (10 Hz) individual-run CSV files are provided alongside this document. The tables below give the run-averaged records decimated to 1 Hz for compactness; they are the direct inputs to the regressions of the main text.

Table S1.1. Set 1 — CKD (NH3): run-averaged record, 1 Hz.

Time (s)	NH3 Concentration (ppm)	I (milli v a.u.)	x	y
0	6.61e-5	-0.359	0.328	0.121
1	1.526	-0.333	0.188	0.170
2	2.836	-0.531	0.512	0.016
3	3.972	-0.520	1.174	0.032
4	4.968	-0.088	1.817	0.653
5	5.847	-0.397	2.362	0.457
6	6.628	-0.312	2.889	0.423
7	7.328	-0.301	3.639	0.536
8	7.959	-0.572	4.371	0.789
9	8.529	-0.698	4.620	0.658
10	9.048	-0.687	5.256	0.715
11	9.522	-0.919	5.765	1.122
12	9.957	-1.030	6.328	1.205
13	10.357	-1.293	6.691	0.961
14	10.726	-1.307	7.164	0.947
15	11.068	-1.368	7.962	1.062
16	11.386	-1.456	8.337	1.064
17	11.682	-1.590	9.176	1.341
18	11.958	-1.458	9.652	1.352
19	12.217	-1.737	10.227	1.697
20	12.459	-1.694	10.821	1.821
21	12.687	-1.799	11.503	1.967
22	12.902	-1.981	12.044	1.935
23	13.104	-1.865	12.486	1.993
24	13.295	-2.178	13.074	2.134
25	13.476	-2.097	13.874	2.317
26	13.647	-2.516	14.082	2.200
27	13.809	-2.373	15.127	2.425
28	13.964	-2.983	15.476	2.761
29	14.110	-2.814	16.362	2.946
30	14.250	-3.351	16.662	2.944
31	14.384	-3.322	17.511	3.098
32	14.511	-3.849	17.727	3.232
33	14.633	-3.645	18.863	3.457
34	14.749	-3.984	19.307	3.645
35	14.861	-3.840	19.922	3.485
36	14.968	-4.065	20.559	3.776
37	15.070	-4.038	21.283	3.994
38	15.168	-4.084	21.807	4.217
39	15.263	-4.190	22.583	4.301
40	15.354	-4.536	23.229	4.564

Time (s)	NH3 Concentration (ppm)	I (milli v a.u.)	x	y
41	15.441	-4.603	23.848	4.794
42	15.526	-4.794	24.809	4.914
43	15.607	-5.370	25.310	5.207
44	15.685	-5.232	25.933	5.370
45	15.761	-5.464	26.395	5.315
46	15.834	-5.755	27.164	5.481
47	15.904	-5.828	27.992	5.992
48	15.972	-5.891	28.558	6.129
49	16.038	-6.279	29.446	6.144
50	16.102	-6.352	30.116	6.414
51	16.164	-6.456	30.841	6.670
52	16.224	-6.296	31.104	6.772
53	16.282	-6.570	32.275	7.305
54	16.338	-6.622	32.524	7.274
55	16.392	-6.706	33.452	7.513
56	16.445	-6.984	34.193	7.555
57	16.497	-6.842	35.087	7.816
58	16.547	-7.046	35.595	7.919
59	16.595	-7.678	36.663	8.350

Table S1.2. Set 2 — SIBO (H2+CH4): run-averaged record, 1 Hz.

Time (s)	CH4 Concentration (ppm)	H2 Concentration (ppm)	I (milli v a.u.)	x	y
0	1.21e-4	1.63e-4	-0.031	-9.59e-4	-0.034
1	2.785	3.770	0.132	0.128	-0.019
2	5.176	7.005	0.195	0.862	0.133
3	7.250	9.812	0.356	1.565	0.087
4	9.066	12.270	0.612	1.783	0.656
5	10.670	14.441	0.283	2.111	0.724
6	12.097	16.372	0.454	2.924	0.957
7	13.374	18.101	0.484	3.751	1.176
8	14.524	19.658	0.045	4.016	1.082
9	15.566	21.067	0.236	4.630	1.261
10	16.513	22.349	0.125	4.990	1.234
11	17.378	23.520	0.076	5.462	1.347
12	18.171	24.593	-0.076	6.087	1.589
13	18.901	25.581	-0.191	6.546	1.461
14	19.575	26.494	-0.114	7.154	1.451
15	20.200	27.339	-0.216	7.580	1.591
16	20.780	28.124	-0.469	8.067	1.871
17	21.320	28.855	-0.254	8.720	1.970
18	21.824	29.537	-0.246	9.286	2.133
19	22.296	30.176	-0.145	9.856	2.447
20	22.739	30.775	-0.056	10.549	2.653
21	23.154	31.337	-0.332	10.958	2.723
22	23.546	31.867	-0.430	11.681	2.877
23	23.915	32.367	-0.363	12.369	2.911
24	24.263	32.838	-0.669	12.861	3.108

Time (s)	CH4 Concentration (ppm)	H2 Concentration (ppm)	I (milli v a.u.)	x	y
25	24.593	33.285	-0.871	13.449	3.189
26	24.905	33.708	-1.018	14.170	3.236
27	25.202	34.109	-1.397	14.751	3.247
28	25.484	34.490	-1.364	15.275	3.429
29	25.752	34.853	-1.787	15.936	3.610
30	26.007	35.198	-1.631	16.660	3.817
31	26.250	35.528	-2.024	17.087	3.853
32	26.483	35.842	-1.849	18.046	4.220
33	26.705	36.143	-2.173	18.480	4.325
34	26.917	36.431	-2.178	19.166	4.518
35	27.121	36.706	-2.319	20.098	4.674
36	27.316	36.970	-2.491	20.621	4.781
37	27.503	37.223	-2.317	21.308	4.955
38	27.682	37.466	-2.727	21.948	5.179
39	27.855	37.699	-2.739	22.583	5.209
40	28.021	37.924	-3.322	23.386	5.233
41	28.180	38.140	-3.042	23.838	5.373
42	28.334	38.348	-3.444	24.418	5.552
43	28.482	38.549	-3.807	25.452	5.678
44	28.625	38.742	-3.775	26.051	5.763
45	28.763	38.929	-4.102	26.535	6.234
46	28.896	39.109	-3.837	27.596	6.281
47	29.025	39.283	-4.383	28.225	6.327
48	29.149	39.452	-4.184	28.917	6.591
49	29.270	39.614	-4.556	29.549	6.813
50	29.386	39.772	-4.728	30.395	7.009
51	29.499	39.924	-4.597	30.971	6.777
52	29.608	40.072	-4.829	31.615	7.103
53	29.714	40.216	-5.210	32.402	7.436
54	29.817	40.355	-5.213	33.224	7.450
55	29.916	40.489	-5.264	34.020	7.788
56	30.013	40.620	-5.473	35.247	8.094
57	30.107	40.747	-5.833	35.586	7.879
58	30.198	40.871	-6.192	36.502	8.075
59	30.287	40.991	-6.275	37.140	7.929
60	30.373	41.107	-6.199	38.167	8.287
61	30.457	41.221	-6.370	38.986	8.386
62	30.538	41.331	-6.602	39.852	8.422
63	30.617	41.438	-6.602	40.850	8.777
64	30.695	41.543	-6.679	41.614	8.952
65	30.770	41.645	-6.780	41.946	8.937
66	30.843	41.744	-6.950	42.791	8.990
67	30.915	41.841	-7.135	43.551	9.409
68	30.985	41.935	-7.255	44.608	9.754
69	31.053	42.027	-7.214	46.077	9.882
70	31.119	42.117	-7.415	46.536	9.823
71	31.184	42.205	-7.502	47.313	10.136
72	31.247	42.290	-7.865	48.114	9.977

Time (s)	CH4 Concentration (ppm)	H2 Concentration (ppm)	I (milli v a.u.)	x	y
73	31.309	42.374	-7.915	48.651	10.145
74	31.369	42.456	-8.095	49.551	10.447
75	31.428	42.535	-8.406	50.807	10.601
76	31.485	42.613	-8.733	51.715	10.724
77	31.542	42.689	-9.016	53.159	10.458
78	31.597	42.764	-8.920	52.920	11.099
79	31.651	42.837	-9.156	53.745	11.029
80	31.703	42.908	-9.392	54.503	11.163
81	31.755	42.978	-9.587	55.686	11.395
82	31.805	43.046	-9.741	56.343	11.424
83	31.855	43.113	-9.969	57.489	11.775
84	31.903	43.178	-9.925	58.257	11.877
85	31.950	43.242	-10.116	59.157	12.095
86	31.997	43.305	-10.536	60.888	12.404
87	32.042	43.366	-10.502	60.970	12.551
88	32.087	43.427	-10.722	62.033	12.734
89	32.130	43.486	-11.140	62.837	12.959
90	32.173	43.544	-11.195	63.098	13.413
91	32.215	43.601	-11.491	64.533	13.044
92	32.256	43.656	-11.568	65.690	13.380
93	32.297	43.711	-11.665	66.778	13.713
94	32.336	43.765	-11.909	67.495	13.801
95	32.375	43.817	-12.287	68.934	13.751
96	32.413	43.869	-12.411	69.904	14.069
97	32.451	43.920	-12.576	70.733	14.051
98	32.487	43.969	-12.816	71.635	14.120
99	32.524	44.018	-13.108	72.724	14.724

Table S1.3. Set 3 — Cystic Fibrosis (CO+NO): run-averaged record, 1 Hz.

Time (s)	CO Concentration (ppm)	NO Concentration (ppb)	I (milli v a.u.)	x	y
0	11.582	2.03e-5	-0.687	4.019	0.751
1	11.080	0.485	-0.913	4.479	1.000
2	10.621	0.929	-1.012	4.644	0.941
3	10.197	1.338	-1.255	5.229	1.007
4	9.807	1.716	-1.356	5.721	1.064
5	9.445	2.066	-1.432	5.755	1.254
6	9.109	2.391	-1.405	6.573	1.592
7	8.796	2.694	-1.608	6.900	1.715
8	8.504	2.976	-1.795	7.626	1.614
9	8.230	3.241	-1.710	7.784	1.708
10	7.974	3.489	-1.870	8.508	2.076
11	7.733	3.722	-2.015	8.922	1.816
12	7.506	3.941	-2.229	9.607	2.101
13	7.293	4.148	-2.422	10.033	2.299
14	7.091	4.343	-2.705	10.515	2.294
15	6.899	4.528	-2.942	11.188	2.628
16	6.718	4.703	-3.110	11.597	2.651

Time (s)	CO Concentration (ppm)	NO Concentration (ppb)	I (milli v a.u.)	x	y
17	6.547	4.869	-3.214	11.948	2.659
18	6.383	5.027	-3.319	12.575	2.791
19	6.228	5.177	-3.370	13.227	2.996
20	6.080	5.320	-3.772	13.696	3.115
21	5.939	5.457	-3.769	14.435	3.276
22	5.804	5.587	-4.105	14.890	3.462
23	5.676	5.711	-4.136	15.258	3.466
24	5.553	5.830	-4.358	15.811	3.823
25	5.435	5.944	-4.851	16.565	3.846
26	5.322	6.054	-4.954	17.176	4.063
27	5.213	6.158	-5.003	17.586	4.047
28	5.109	6.259	-5.091	18.322	4.256
29	5.009	6.356	-5.128	18.971	4.180
30	4.913	6.449	-5.434	19.510	4.696
31	4.821	6.538	-5.483	20.186	4.752
32	4.732	6.624	-5.487	20.534	4.675
33	4.646	6.707	-6.156	21.314	5.061
34	4.563	6.787	-5.940	22.132	5.341
35	4.483	6.865	-6.109	22.577	5.193
36	4.406	6.939	-6.737	23.054	5.216
37	4.331	7.011	-6.838	23.962	5.520
38	4.259	7.081	-6.667	24.514	5.675
39	4.190	7.148	-7.539	24.866	5.642
40	4.122	7.214	-7.744	25.476	5.765
41	4.057	7.277	-7.701	26.462	5.929
42	3.993	7.338	-7.915	27.019	6.425
43	3.932	7.397	-8.294	27.243	6.147
44	3.873	7.455	-8.309	28.267	6.205
45	3.815	7.511	-8.534	28.927	6.390
46	3.759	7.565	-8.539	29.386	6.566
47	3.705	7.617	-8.689	30.134	6.508
48	3.652	7.669	-8.914	31.035	6.870
49	3.600	7.718	-8.999	31.571	7.116
50	3.550	7.767	-9.260	32.142	6.944
51	3.502	7.814	-9.389	32.837	7.159
52	3.455	7.859	-9.495	33.449	7.258
53	3.409	7.904	-9.684	34.295	7.812
54	3.364	7.947	-10.143	34.734	7.484
55	3.320	7.989	-10.329	35.628	7.744
56	3.278	8.030	-10.451	36.042	8.022
57	3.236	8.070	-10.665	36.615	7.892
58	3.196	8.109	-10.821	37.630	8.177
59	3.156	8.148	-11.046	38.160	8.239

Table S1.4. Set 4 — COPD (CO+NO): run-averaged record, 1 Hz.

Time (s)	CO Concentration (ppm)	NO Concentration (ppb)	I (milli v a.u.)	x	y
0	20.611	1.44e-4	-4.897	21.169	4.933

Time (s)	CO Concentration (ppm)	NO Concentration (ppb)	I (milli v a.u.)	x	y
1	20.284	3.550	-4.839	21.256	4.852
2	19.967	6.989	-4.833	21.394	4.867
3	19.661	10.322	-5.003	21.292	4.886
4	19.363	13.554	-5.065	21.116	4.590
5	19.075	16.691	-5.101	21.326	4.748
6	18.795	19.735	-5.261	21.839	4.949
7	18.523	22.691	-5.434	22.496	5.151
8	18.259	25.562	-5.483	22.938	5.143
9	18.002	28.353	-5.449	22.957	5.221
10	17.752	31.066	-5.947	23.372	5.317
11	17.509	33.705	-6.193	24.194	5.264
12	17.273	36.273	-6.283	24.544	5.625
13	17.043	38.773	-6.654	25.011	5.567
14	16.819	41.207	-6.928	25.642	5.842
15	16.601	43.578	-6.870	26.201	5.657
16	16.388	45.888	-7.456	27.087	5.893
17	16.181	48.139	-7.572	27.524	6.101
18	15.979	50.335	-7.567	28.394	5.979
19	15.782	52.476	-7.552	29.160	6.258
20	15.590	54.565	-7.712	29.544	6.173
21	15.402	56.604	-8.039	30.222	6.356
22	15.219	58.594	-7.987	30.851	6.413
23	15.040	60.538	-8.126	31.398	6.749
24	14.866	62.436	-8.261	32.508	6.852

Table S1.5. Set 5 — Heart Disease (Acetone+n-Pentane): run-averaged record, 1 Hz.

Time (s)	Acetone Concentration (ppm)	n-Pentane Concentration (ppb)	I (milli v a.u.)	x	y
0	2.805	2.17e-5	0.414	4.307	1.064
1	2.683	0.519	0.458	4.367	0.949
2	2.572	0.996	0.573	4.378	1.052
3	2.470	1.434	0.257	4.420	0.785
4	2.375	1.839	0.256	4.931	1.120
5	2.287	2.214	0.027	4.888	1.381
6	2.206	2.562	-0.101	5.571	1.530
7	2.130	2.886	-0.185	6.065	1.568
8	2.060	3.189	-0.502	6.392	1.841
9	1.993	3.472	-0.156	7.031	2.106
10	1.931	3.738	-0.595	7.389	1.927
11	1.873	3.987	-0.283	8.188	1.910
12	1.818	4.222	-0.628	8.372	1.807
13	1.766	4.444	-0.448	9.162	2.022
14	1.717	4.653	-0.728	9.637	2.271
15	1.671	4.851	-0.744	10.035	2.405
16	1.627	5.039	-0.939	10.607	2.487
17	1.586	5.217	-0.682	11.237	2.799
18	1.546	5.386	-0.985	11.807	2.915

Time (s)	Acetone Concentration (ppm)	n-Pentane Concentration (ppb)	I (milli v a.u.)	x	y
19	1.508	5.547	-0.775	12.165	3.251
20	1.473	5.700	-1.447	13.030	3.156
21	1.438	5.846	-1.461	13.272	3.227
22	1.406	5.986	-2.096	13.750	3.273
23	1.375	6.119	-1.958	14.464	3.354
24	1.345	6.247	-2.550	14.852	3.445
25	1.316	6.369	-2.182	15.573	3.810
26	1.289	6.486	-2.751	15.962	3.970
27	1.263	6.598	-2.619	16.738	4.055
28	1.237	6.706	-3.095	17.100	4.189
29	1.213	6.810	-3.094	17.939	4.393
30	1.190	6.909	-3.237	18.479	4.321
31	1.168	7.005	-3.351	18.956	4.598
32	1.146	7.097	-3.305	19.885	4.402
33	1.125	7.186	-3.490	20.128	4.580
34	1.105	7.272	-3.469	20.776	4.826
35	1.086	7.355	-3.758	21.332	5.000
36	1.067	7.435	-4.098	22.030	5.476
37	1.049	7.512	-4.429	22.792	5.389
38	1.032	7.587	-4.481	23.304	5.218
39	1.015	7.659	-4.663	23.630	5.302

Table S1.6. Set 6 — Base Gas (healthy surrogate): run-averaged record, 1 Hz.

Time (s)	I (milli v a.u.)	x	y
0	-0.008	0.280	0.178
1	0.140	0.544	0.212
2	0.111	1.340	0.402
3	0.286	1.857	0.347
4	0.217	1.918	0.489
5	0.105	2.787	0.663
6	-0.156	3.438	0.939
7	-0.125	3.726	0.939
8	-0.198	4.004	0.915
9	-0.271	4.819	1.254
10	-0.293	5.514	1.641
11	-0.592	5.824	1.519
12	-0.724	6.376	1.316
13	-1.357	6.803	1.419
14	-0.846	7.726	1.679
15	-1.186	7.903	1.962
16	-1.148	8.466	2.057
17	-1.032	8.581	2.028
18	-1.567	9.442	2.299
19	-1.534	10.132	2.542
20	-1.966	10.793	2.994
21	-1.677	11.532	2.941

Time (s)	I (milli v a.u.)	x	y
22	-2.105	11.777	3.095
23	-1.860	12.521	3.030
24	-2.221	13.352	3.596
25	-2.130	13.782	3.508
26	-2.444	14.093	3.659
27	-2.652	14.835	3.812
28	-2.653	15.178	3.738
29	-2.909	16.030	3.894
30	-3.015	16.752	4.317
31	-3.402	17.082	4.344
32	-3.132	17.843	4.270
33	-3.796	18.412	4.739
34	-3.413	19.418	4.979
35	-4.038	19.811	4.959
36	-4.043	20.517	5.074
37	-4.135	21.271	5.295
38	-4.711	21.531	5.333
39	-4.466	22.416	5.532
40	-4.932	23.018	5.743
41	-4.936	23.800	5.959
42	-4.824	24.558	6.010
43	-5.222	25.046	6.229
44	-5.542	25.975	6.278
45	-5.598	26.845	6.587
46	-5.830	27.249	6.665
47	-6.245	28.001	7.050
48	-6.005	28.614	7.372
49	-6.336	29.277	7.175
50	-6.606	30.072	7.268
51	-6.815	30.456	7.711
52	-6.939	31.239	7.740
53	-6.945	31.844	7.713
54	-7.472	32.460	7.847
55	-7.664	33.466	8.841
56	-7.901	34.439	8.652
57	-7.755	35.148	8.643
58	-8.305	36.507	8.619
59	-8.354	36.614	8.986

Table S1.7. Set 7 — Ambient control: run-averaged record, 1 Hz.

Time (s)	I (milli v a.u.)	x	y
0	0	0	0
1	-0.108	0.155	0.105
2	-0.172	0.245	0.036
3	-0.186	0.219	0.037
4	-0.150	0.129	-0.055
5	-0.283	0.340	0.220

Time (s)	I (milli v a.u.)	x	y
6	-0.185	0.237	0.118
7	-0.078	0.349	0.135
8	-0.144	0.067	-0.170
9	-0.120	0.034	-0.100
10	-0.352	-0.024	-0.146
11	-0.151	0.388	0.252
12	-0.341	0.170	-0.046
13	-0.193	0.123	-0.047
14	-0.089	0.232	-0.005
15	-0.105	0.040	-0.066
16	-0.086	0.054	-0.020
17	0.005	0.061	-0.061
18	0.087	0.063	-0.233
19	-0.074	0.226	0.100
20	-0.209	0.154	-0.159
21	-0.247	0.214	0.040
22	-0.195	0.071	-0.227
23	-0.164	0.331	0.056
24	-0.134	0.141	-0.100
25	-0.066	0.267	-0.021
26	-0.204	0.254	-0.046
27	-0.062	0.273	0.013
28	-0.026	0.122	-0.183
29	0.155	-0.002	-0.337
30	0.048	0.312	-0.014
31	-0.120	0.106	-0.201
32	-0.105	0.081	-0.182
33	0.261	-0.223	-0.526
34	-0.040	0.210	-0.111
35	-0.073	0.146	-0.133
36	-0.172	0.247	-0.016
37	-0.206	0.221	-0.156
38	-0.100	0.152	-0.152
39	-0.132	0.216	-0.043
40	-0.083	0.200	-0.165
41	-0.071	0.218	-0.032
42	-0.060	0.086	-0.175
43	0.078	0.164	-0.063
44	0.115	-0.030	-0.351
45	-0.009	0.086	-0.180
46	0.160	0.136	-0.156
47	0.179	0.152	0.027
48	-0.149	0.227	0.032
49	-0.043	0.302	0.040
50	0.016	0.139	-0.092
51	-0.054	0.128	-0.146
52	0.006	0.215	0.005
53	-0.006	0.108	-0.146

Time (s)	I (milli v a.u.)	x	y
54	-0.033	0.065	-0.223
55	0.099	0.131	-0.056
56	0.114	0.185	0.006
57	0.121	0.180	-0.041
58	-0.072	0.129	-0.077
59	0.002	0.030	-0.244

S2. Complete Regression Model Comparisons

Table S2.1. y–x regressions: all candidate models with equations, AIC, R^2 and coefficient CIs.

Set	Model	Best	R2	adj_R2	AIC	RMSE	Equation
CKD	Linear		0.984	0.984	-1397.7	0.310	y = -0.5838 +0.2286·x
CKD	Quadratic		0.997	0.997	-2353.9	0.140	y = -0.03352 +0.1328·x +0.002665·x^2
CKD	Cubic	YES	0.997	0.997	-2430.8	0.131	y = +0.074 +0.09468·x +0.005344·x^2 -4.932e-05·x^3
SIBO	Linear		0.996	0.996	-2642.1	0.266	y = +0.4441 +0.1988·x
SIBO	Quadratic		0.998	0.998	-3485.4	0.174	y = +0.0687 +0.2334·x -0.0004956·x^2
SIBO	Cubic	YES	0.998	0.998	-3556.8	0.168	y = -0.03021 +0.2515·x -0.001144·x^2 +6.068e-06·x^3
CF	Linear		0.994	0.994	-2106.1	0.172	y = +0.04965 +0.2193·x
CF	Quadratic	YES	0.997	0.997	-2420.5	0.132	y = -0.3401 +0.2688·x -0.001189·x^2
CF	Cubic		0.997	0.997	-2419.8	0.132	y = -0.3004 +0.2604·x -0.0007248·x^2 -7.372e-06·x^3
COPD	Linear		0.962	0.962	-1015.0	0.130	y = +1.044 +0.1758·x
COPD	Quadratic		0.965	0.965	-1035.0	0.124	y = -1.226 +0.3525·x -0.003365·x^2
COPD	Cubic	YES	0.968	0.967	-1054.1	0.119	y = -20.02 +2.528·x -0.08632·x^2 +0.001043·x^3
Heart	Linear		0.986	0.986	-1437.3	0.165	y = +0.08204 +0.2279·x
Heart	Quadratic		0.988	0.988	-1473.6	0.157	y = -0.1401 +0.2693·x -0.001524·x^2
Heart	Cubic	YES	0.988	0.988	-1485.3	0.154	y = +0.1589 +0.181·x +0.005729·x^2 -0.0001755·x^3
Base	Linear		0.996	0.996	-2117.3	0.170	y = -0.04291 +0.2481·x
Base	Quadratic		0.996	0.996	-2136.5	0.167	y = -0.1091 +0.2593·x -0.000309·x^2
Base	Cubic	YES	0.996	0.996	-2137.5	0.167	y = -0.07952 +0.2495·x +0.0003622·x^2 -1.218e-05·x^3
Ambient	Linear	YES	0.589	0.588	-3062.2	0.078	y = -0.2372 +0.9442·x
Ambient	Quadratic		0.589	0.587	-3060.2	0.078	y = -0.2373 +0.9466·x -0.009824·x^2
Ambient	Cubic		0.589	0.587	-3058.3	0.078	y = -0.2381 +0.9508·x +0.04791·x^2 -0.2231·x^3

Table S2.2. I–C regressions: all candidate models with equations, AIC, R^2 and coefficient CIs.

Set	Gas	Model	Best	R2	adj_R2	AIC	RMSE	Equation
CKD	NH_3 (ppm)	Linear		0.711	0.710	248.549	1.224	I = +2.916 -0.4885·C
CKD	NH_3 (ppm)	Quadratic		0.941	0.940	-699.311	0.555	I = -2.192 +0.7985·C -0.06343·C^2

Set	Gas	Model	Best	R2	adj_R2	AIC	RMSE	Equation
CKD	NH_3 (ppm)	Cubic	YES	0.984	0.984	-1493.9	0.286	I = +0.4905 -0.68·C +0.122·C^2 - 0.006518·C^3
SIBO	H_2 (ppm)	Linear		0.600	0.599	1933.0	2.621	I = +7.449 - 0.3441·C
SIBO	H_2 (ppm)	Quadratic		0.883	0.883	707.174	1.418	I = -5.172 +0.7846·C - 0.02024·C^2
SIBO	H_2 (ppm)	Cubic	YES	0.966	0.966	-520.927	0.767	I = +3.146 - 0.8642·C +0.05523·C^2 - 0.0009761·C^3
SIBO	CH_4 (ppm)	Linear		0.600	0.599	1933.0	2.621	I = +7.449 - 0.4657·C
SIBO	CH_4 (ppm)	Quadratic		0.883	0.883	707.174	1.418	I = -5.172 +1.062·C -0.03708·C^2
SIBO	CH_4 (ppm)	Cubic	YES	0.966	0.966	-520.927	0.767	I = +3.146 -1.17·C +0.1012·C^2 - 0.00242·C^3
CF	CO (ppm)	Linear		0.856	0.856	208.382	1.184	I = -13.01 +1.321·C
CF	CO (ppm)	Quadratic		0.984	0.984	-1092.5	0.400	I = -21.84 +4.385·C -0.2305·C^2
CF	CO (ppm)	Cubic	YES	0.997	0.996	-2016.3	0.185	I = -30.59 +8.881·C -0.9359·C^2 +0.03407·C^3
CF	NO (ppb)	Linear		0.856	0.856	208.382	1.184	I = +2.289 -1.366·C
CF	NO (ppb)	Quadratic		0.984	0.984	-1092.5	0.400	I = -1.97 +0.986·C - 0.2465·C^2
CF	NO (ppb)	Cubic	YES	0.997	0.996	-2016.3	0.185	I = -0.3391 - 0.9425·C +0.2649·C^2 - 0.03767·C^3
COPD	CO (ppm)	Linear		0.965	0.965	-740.053	0.225	I = -18.47 +0.693·C
COPD	CO (ppm)	Quadratic		0.979	0.979	-868.971	0.173	I = -35.3 +2.632·C - 0.0553·C^2
COPD	CO (ppm)	Cubic	YES	0.992	0.992	-1107.8	0.107	I = +151.1 -29.54·C +1.785·C^2 - 0.0349·C^3
COPD	NO (ppb)	Linear		0.965	0.965	-740.053	0.225	I = -4.183 - 0.06376·C
COPD	NO (ppb)	Quadratic		0.979	0.979	-868.971	0.173	I = -4.546 - 0.03244·C - 0.0004681·C^2
COPD	NO (ppb)	Cubic	YES	0.992	0.992	-1107.8	0.107	I = -4.95 +0.03915·C - 0.003155·C^2 +2.719e-05·C^3
Heart	Acetone (ppm)	Linear		0.845	0.845	-377.451	0.619	I = -6.486 +2.966·C
Heart	Acetone (ppm)	Quadratic		0.970	0.970	-1029.0	0.274	I = -13.54 +11.61·C -2.423·C^2
Heart	Acetone (ppm)	Cubic	YES	0.986	0.986	-1335.1	0.186	I = -23.46 +29.72·C -12.86·C^2 +1.904·C^3
Heart	n-Pentane (ppb)	Linear		0.845	0.845	-377.451	0.619	I = +1.834 - 0.6933·C
Heart	n-Pentane (ppb)	Quadratic		0.970	0.970	-1029.0	0.274	I = -0.04284 +0.4644·C - 0.1324·C^2
Heart	n-Pentane (ppb)	Cubic	YES	0.986	0.986	-1335.1	0.186	I = +0.7655 - 0.5924·C +0.1729·C^2 - 0.02431·C^3

S4. Classification Details

Table S4.1. Run-level majority-vote nearest-centroid confusion matrix.

	CKD	SIBO	CF	COPD	Heart	Base	Ambient
CKD	2	0	1	0	0	0	0
SIBO	0	3	0	0	0	0	0
CF	0	0	3	0	0	0	0
COPD	1	0	2	0	0	0	0
Heart	0	1	0	0	1	1	0
Base	1	1	0	0	0	0	0
Ambient	2	0	0	0	0	0	0

Table S4.2. Sample-level nearest-centroid confusion matrix (counts).

	CKD	SIBO	CF	COPD	Heart	Base	Ambient
CKD	936	347	261	160	16	75	5
SIBO	419	2389	19	0	26	141	6
CF	148	0	1468	99	0	85	0
COPD	132	0	316	186	20	96	0
Heart	82	432	5	0	375	306	0
Base	468	329	124	0	79	189	11
Ambient	684	0	153	0	0	17	346

Table S4.3. Trajectory LOO details per run: distances to the true class, best and second-best classes, and decision margin.

Set	Run	Pred	d_true	d_best	d_second	margin
CKD	1	CF	3.766	2.089	2.299	0.210
CKD	2	CKD	2.394	2.394	3.133	0.739
CKD	3	CKD	1.924	1.924	2.743	0.819
SIBO	1	CKD	6.901	2.170	3.310	1.139
SIBO	2	SIBO	2.498	2.498	3.099	0.602
SIBO	3	Heart	7.016	3.957	4.062	0.105
CF	1	COPD	2.281	2.219	2.281	0.062
CF	2	CF	1.404	1.404	2.079	0.675
CF	3	CF	1.653	1.653	2.320	0.667
COPD	1	COPD	2.756	2.756	3.044	0.287
COPD	2	COPD	1.921	1.921	2.905	0.984
COPD	3	CF	3.631	2.304	2.799	0.495
Heart	1	SIBO	1.846	1.489	1.846	0.356
Heart	2	Heart	1.239	1.239	1.744	0.504
Heart	3	Base	1.545	1.319	1.545	0.226
Base	1	Heart	4.795	1.843	1.999	0.156
Base	2	Heart	4.795	3.239	3.469	0.230
Ambient	1	Ambient	1.079	1.079	2.200	1.121
Ambient	2	Ambient	1.079	1.079	2.104	1.025

Table S4.4. Per-run rate-fingerprint features used by the LOO 1-NN classifier.

Set	Run	dI_dt	dx_dt	dy_dt	slope_y_x	curv_I	curv_x	resid_sigma_I
CKD	1	-0.152	0.625	0.137	0.219	-2.35e-4	0.002	0.247
CKD	2	-0.116	0.621	0.152	0.245	-0.001	0.001	0.289
CKD	3	-0.122	0.617	0.136	0.221	-9.38e-4	0.002	0.333
SIBO	1	-0.129	0.699	0.182	0.262	-3.82e-4	0.002	0.354
SIBO	2	-0.159	0.736	0.152	0.206	-0.001	0.002	0.379
SIBO	3	-0.138	0.777	0.108	0.133	-5.04e-4	0.004	0.358
CF	1	-0.170	0.602	0.143	0.238	-7.01e-4	0.002	0.228
CF	2	-0.185	0.579	0.123	0.211	-4.78e-4	0.002	0.288
CF	3	-0.184	0.588	0.123	0.209	-4.37e-4	0.002	0.298
COPD	1	-0.183	0.494	0.100	0.202	-0.001	0.013	0.275
COPD	2	-0.164	0.521	0.096	0.183	1.65e-4	0.014	0.200
COPD	3	-0.145	0.497	0.071	0.143	2.96e-5	0.014	0.244
Heart	1	-0.118	0.526	0.124	0.236	-0.001	0.004	0.306
Heart	2	-0.146	0.538	0.119	0.219	-0.001	0.003	0.294
Heart	3	-0.138	0.535	0.123	0.229	-0.002	0.003	0.288
Base	1	-0.141	0.627	0.144	0.231	-0.001	0.002	0.341
Base	2	-0.160	0.608	0.163	0.266	-0.001	0.003	0.309
Ambient	1	0.002	-0.001	-0.003	0.943	1.24e-4	-5.93e-6	0.126
Ambient	2	0.005	-1.08e-6	-0.003	0.882	-1.66e-4	-2.90e-5	0.181

S3. Python Algorithm (complete listing, Instruction 34)

The following self-contained Python 3 program reproduces every table and figure of the main report from the 19 raw CSV files. Dependencies: numpy, pandas, scipy, matplotlib.

```
#!/usr/bin/env python3
"""
================================================================================
 Laser Back-Scattering Breath-Metabolite Discrimination — Full Analysis Pipeline
================================================================================
 Implements Instructions "Inputs for consideration Individual Runs
 Aug 14 2026": data loading, run-averaging, 3D scatter plots (common axes),
 best-fit regressions (y~x and I~C) with 95% confidence intervals and error
 bands, Mahalanobis cluster statistics (d_m mean/min/max), nearest-centroid
 Mahalanobis classification validation, and print-ready 330-dpi figures.


 Author: Analysis pipeline prepared by Prof. Rao Tatavarti (CATS, Nashik)
 Date  : 14 August 2026
================================================================================
"""
import os, json, glob, itertools
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import FancyBboxPatch, FancyArrowPatch
from mpl_toolkits.mplot3d import Axes3D  # noqa


# ------------------------------------------------------------------ constants
UP   = "/root/.claude/uploads/c0ea54a2-a3e7-5b99-b7e8-a8186726361b"
OUT  = "/mnt/user-data/working/results"
FIG  = os.path.join(OUT, "figures")
TAB  = os.path.join(OUT, "tables")
DPI  = 330
for d in (OUT, FIG, TAB):
    os.makedirs(d, exist_ok=True)


# Okabe-Ito colour-blind-safe palette, fixed order for the seven data sets
PAL = {
    "CKD":       "#E69F00",   # orange
    "SIBO":      "#56B4E9",   # sky blue
    "CF":        "#009E73",   # bluish green
    "COPD":      "#D55E00",   # vermillion
    "HEART":     "#CC79A7",   # reddish purple
    "BASE":      "#0072B2",   # blue
    "AMBIENT":   "#7F7F7F",   # gray (control)
}
ORDER = ["CKD", "SIBO", "CF", "COPD", "HEART", "BASE", "AMBIENT"]
LABEL = {
    "CKD": "Set 1: CKD (NH$_3$)",
    "SIBO": "Set 2: SIBO (H$_2$+CH$_4$)",
    "CF": "Set 3: Cystic Fibrosis (CO+NO)",
    "COPD": "Set 4: COPD (CO+NO)",
    "HEART": "Set 5: Heart Disease (Acetone+n-Pentane)",
    "BASE": "Set 6: Healthy / Base Gas (N$_2$+O$_2$)",
    "AMBIENT": "Set 7: Ambient Control (no gas)",
}
SHORT = {"CKD":"CKD","SIBO":"SIBO","CF":"CF","COPD":"COPD",
         "HEART":"Heart","BASE":"Base","AMBIENT":"Ambient"}


FILES = {
    "CKD": ["fab55914", "03ec0e41", "2abf5239"],
    "SIBO": ["108c663c", "93276ccd", "8c1a660b"],
    "CF":  ["397e4e2d", "a43f85e9", "d6583af4"],
    "COPD":["4bc8b157", "e0e697ce", "425a1996"],
    "HEART":["7e5f9ada", "8a72dc39", "16240ee1"],
    "BASE":["7905b900", "d49069da"],
    "AMBIENT":["4fece6d9", "98f1de65"],
}
CONC_COLS = {   # canonical concentration column names per set (as in files)
    "CKD":  [("NH3 Concentration (ppm)", "NH$_3$ (ppm)")],
    "SIBO": [("H2 Concentration (ppm)", "H$_2$ (ppm)"),
             ("CH4 Concentration (ppm)", "CH$_4$ (ppm)")],
    "CF":   [("CO Concentration (ppm)", "CO (ppm)"),
             ("NO Concentration (ppb)", "NO (ppb)")],
    "COPD": [("CO Concentration (ppm)", "CO (ppm)"),
             ("NO Concentration (ppb)", "NO (ppb)")],
    "HEART":[("Acetone Concentration (ppm)", "Acetone (ppm)"),
             ("n-Pentane Concentration (ppb)", "n-Pentane (ppb)")],
    "BASE": [], "AMBIENT": [],
}


# ------------------------------------------------------------------ loading
def find_file(prefix):
    hits = glob.glob(os.path.join(UP, prefix + "*"))
    assert len(hits) == 1, f"ambiguous or missing {prefix}"
    return hits[0]


def load_all():
    data = {}
    for key, prefixes in FILES.items():
        runs = []
        for p in prefixes:
            df = pd.read_csv(find_file(p))
```

```
            df.columns = [c.strip() for c in df.columns]
            runs.append(df)
        data[key] = runs
    return data


def run_average(runs):
    """Average I, x, y (and concentrations) across runs on the common time grid."""
    n = min(len(r) for r in runs)
    base = runs[0].iloc[:n].copy()
    for col in base.columns:
        if col.startswith("Time"):
            continue
        base[col] = np.mean([r[col].values[:n] for r in runs], axis=0)
    return base


# ------------------------------------------------- polynomial OLS with CIs
def poly_ols(x, y, deg):
    """OLS polynomial fit returning coefficients, their 95% CIs, R2, adjR2, AIC."""
    x = np.asarray(x, float); y = np.asarray(y, float)
    n = len(x)
    X = np.vander(x, deg + 1, increasing=True)   # [1, x, x^2, ...]
    beta, res, rank, sv = np.linalg.lstsq(X, y, rcond=None)
    yhat = X @ beta
    resid = y - yhat
    dof = n - (deg + 1)
    sse = float(resid @ resid)
    sst = float(((y - y.mean()) ** 2).sum())
    r2 = 1 - sse / sst if sst > 0 else np.nan
    adj = 1 - (1 - r2) * (n - 1) / dof if dof > 0 else np.nan
    s2 = sse / dof
    XtXinv = np.linalg.pinv(X.T @ X)
    se = np.sqrt(np.diag(XtXinv) * s2)
    tcrit = stats.t.ppf(0.975, dof)
    ci_lo, ci_hi = beta - tcrit * se, beta + tcrit * se
    aic = n * np.log(sse / n) + 2 * (deg + 2)
    rmse = np.sqrt(sse / n)
    return dict(deg=deg, beta=beta, se=se, ci_lo=ci_lo, ci_hi=ci_hi,
                r2=r2, adj_r2=adj, aic=aic, rmse=rmse, s2=s2,
                XtXinv=XtXinv, tcrit=tcrit, n=n, dof=dof)


def poly_bands(fit, xg):
    """Mean-response 95% confidence band and 95% prediction band on grid xg."""
    Xg = np.vander(xg, fit["deg"] + 1, increasing=True)
    yg = Xg @ fit["beta"]
    var_mean = np.einsum("ij,jk,ik->i", Xg, fit["XtXinv"], Xg) * fit["s2"]
    half_conf = fit["tcrit"] * np.sqrt(var_mean)
    half_pred = fit["tcrit"] * np.sqrt(var_mean + fit["s2"])
    return yg, half_conf, half_pred


def best_poly(x, y, degs=(1, 2, 3)):
    fits = [poly_ols(x, y, d) for d in degs]
    best = min(fits, key=lambda f: f["aic"])
    return best, fits


def eqn_string(fit, xname="x", yname="y"):
    terms = []
    for i, b in enumerate(fit["beta"]):
        if i == 0: terms.append(f"{b:+.4g}")
        elif i == 1: terms.append(f"{b:+.4g}·{xname}")
        else: terms.append(f"{b:+.4g}·{xname}^{i}")
    return f"{yname} = " + " ".join(terms)


# ------------------------------------------------------------ Mahalanobis
def maha_stats(features):
    """features: dict key -> (n_i,3) arrays (I,x,y). Returns cluster stats."""
    cent = {k: v.mean(axis=0) for k, v in features.items()}
    cov = {}
    for k, v in features.items():
        c = np.cov(v.T)
        c += np.eye(3) * 1e-9 * np.trace(c)      # regularise
        cov[k] = c
    covinv = {k: np.linalg.pinv(c) for k, c in cov.items()}


    def dmaha(pts, key):
        d = pts - cent[key]
        return np.sqrt(np.einsum("ij,jk,ik->i", d, covinv[key], d))


    within = {k: dmaha(features[k], k) for k in features}
    cross = {}   # (sample-set, cluster) -> distances
    for a in features:
        for b in features:
            cross[(a, b)] = dmaha(features[a], b)
    # centroid-to-centroid distance under pooled covariance
    pooled = np.zeros((3, 3)); ntot = 0
    for k, v in features.items():
        pooled += np.cov(v.T) * (len(v) - 1); ntot += len(v) - 1
    pooled /= ntot
    pinv = np.linalg.pinv(pooled)
    cc = {}
    for a in features:
        for b in features:
            d = cent[a] - cent[b]
            cc[(a, b)] = float(np.sqrt(d @ pinv @ d))
    return dict(cent=cent, cov=cov, covinv=covinv, within=within,
                cross=cross, centroid_dm=cc, pooled=pooled)


# ============================================================ run pipeline
print("Loading data ...")
data = load_all()
```

```
avg = {k: run_average(v) for k, v in data.items()}


# ---- basic summary table ------------------------------------------------------
summary_rows = []
DUR = {"CKD":60,"SIBO":100,"CF":60,"COPD":25,"HEART":40,"BASE":60,"AMBIENT":60}
PRESS = {"CKD":3.1,"SIBO":5.6,"CF":3.6,"COPD":3.2,"HEART":2.6,"BASE":3.4,"AMBIENT":0.0}
for k in ORDER:
    a = avg[k]
    row = dict(Set=SHORT[k], Label=LABEL[k].replace("$",""), Runs=len(data[k]),
               N=len(a), Duration_s=DUR[k], FinalPressure_bar=PRESS[k],
               I_mean=a["I (milli a.u.)"].mean(), I_final=a["I (milli a.u.)"].iloc[-5:].mean(),
               x_final=a["x"].iloc[-5:].mean(), y_final=a["y"].iloc[-5:].mean())
    for col, lab in CONC_COLS[k]:
        row[f"Cmax {lab.replace('$','')}"] = a[col].max()
    summary_rows.append(row)
pd.DataFrame(summary_rows).to_csv(os.path.join(TAB, "T1_dataset_summary.csv"), index=False)


# ---- repeatability: run-to-run RMS difference & Pearson r on I ----------------
rep_rows = []
for k in ORDER:
    runs = data[k]; n = min(len(r) for r in runs)
    for (i, j) in itertools.combinations(range(len(runs)), 2):
        Ii, Ij = runs[i]["I (milli a.u.)"].values[:n], runs[j]["I (milli a.u.)"].values[:n]
        xi, xj = runs[i]["x"].values[:n], runs[j]["x"].values[:n]
        yi, yj = runs[i]["y"].values[:n], runs[j]["y"].values[:n]
        rep_rows.append(dict(Set=SHORT[k], pair=f"R{i+1}-R{j+1}",
                             rms_I=float(np.sqrt(np.mean((Ii-Ij)**2))),
                             r_I=float(np.corrcoef(Ii, Ij)[0, 1]),
                             rms_x=float(np.sqrt(np.mean((xi-xj)**2))),
                             r_x=float(np.corrcoef(xi, xj)[0, 1]),
                             rms_y=float(np.sqrt(np.mean((yi-yj)**2))),
                             r_y=float(np.corrcoef(yi, yj)[0, 1])))
rep_df = pd.DataFrame(rep_rows)
rep_df.to_csv(os.path.join(TAB, "T2_repeatability.csv"), index=False)


# ---- regression y ~ x (run-averaged) ------------------------------------------
yx_rows, yx_fits = [], {}
for k in ORDER:
    a = avg[k]
    fit, allf = best_poly(a["x"].values, a["y"].values)
    yx_fits[k] = fit
    for f in allf:
        yx_rows.append(dict(
            Set=SHORT[k], Model=["Linear","Quadratic","Cubic"][f["deg"]-1],
            Best=("YES" if f["deg"] == fit["deg"] else ""),
            R2=f["r2"], adj_R2=f["adj_r2"], AIC=f["aic"], RMSE=f["rmse"],
            Equation=eqn_string(f, "x", "y"),
            Slope_b1=f["beta"][1], b1_CI_lo=f["ci_lo"][1], b1_CI_hi=f["ci_hi"][1],
            Intercept_b0=f["beta"][0], b0_CI_lo=f["ci_lo"][0], b0_CI_hi=f["ci_hi"][0]))
pd.DataFrame(yx_rows).to_csv(os.path.join(TAB, "T3_regression_y_vs_x.csv"), index=False)


# ---- regression I ~ C (run-averaged) per gas ----------------------------------
ic_rows, ic_fits = [], {}
for k in ORDER:
    a = avg[k]
    for col, lab in CONC_COLS[k]:
        C = a[col].values; I = a["I (milli a.u.)"].values
        fit, allf = best_poly(C, I)
        ic_fits[(k, lab)] = fit
        for f in allf:
            ic_rows.append(dict(
                Set=SHORT[k], Gas=lab.replace("$",""),
                Model=["Linear","Quadratic","Cubic"][f["deg"]-1],
                Best=("YES" if f["deg"] == fit["deg"] else ""),
                R2=f["r2"], adj_R2=f["adj_r2"], AIC=f["aic"], RMSE=f["rmse"],
                Equation=eqn_string(f, "C", "I"),
                Slope_b1=f["beta"][1], b1_CI_lo=f["ci_lo"][1], b1_CI_hi=f["ci_hi"][1],
                Intercept_b0=f["beta"][0], b0_CI_lo=f["ci_lo"][0], b0_CI_hi=f["ci_hi"][0]))
pd.DataFrame(ic_rows).to_csv(os.path.join(TAB, "T4_regression_I_vs_C.csv"), index=False)


# ---- Mahalanobis cluster statistics -------------------------------------------
feats = {k: avg[k][["I (milli a.u.)", "x", "y"]].values for k in ORDER}
M = maha_stats(feats)


dm_rows = []
for k in ORDER:
    w = M["within"][k]
    da = M["cross"][(k, "AMBIENT")]
    db = M["cross"][(k, "BASE")]
    dm_rows.append(dict(Set=SHORT[k],
        dm_within_mean=w.mean(), dm_within_min=w.min(), dm_within_max=w.max(),
        dm_within_CI95_lo=w.mean()-1.96*w.std()/np.sqrt(len(w)),
        dm_within_CI95_hi=w.mean()+1.96*w.std()/np.sqrt(len(w)),
        dm_toAmbient_mean=da.mean(), dm_toAmbient_min=da.min(), dm_toAmbient_max=da.max(),
        dm_toBase_mean=db.mean(), dm_toBase_min=db.min(), dm_toBase_max=db.max()))
pd.DataFrame(dm_rows).to_csv(os.path.join(TAB, "T5_mahalanobis_stats.csv"), index=False)


cc = pd.DataFrame([[M["centroid_dm"][(a, b)] for b in ORDER] for a in ORDER],
                  index=[SHORT[k] for k in ORDER], columns=[SHORT[k] for k in ORDER])
cc.to_csv(os.path.join(TAB, "T6_centroid_mahalanobis_matrix.csv"))


# ---- classification validation (all individual runs, nearest centroid) ------
conf = pd.DataFrame(0, index=[SHORT[k] for k in ORDER], columns=[SHORT[k] for k in ORDER])
for k in ORDER:
    for r in data[k]:
        pts = r[["I (milli a.u.)", "x", "y"]].values
        dall = np.stack([np.sqrt(np.einsum("ij,jk,ik->i",
                    pts - M["cent"][c], M["covinv"][c], pts - M["cent"][c]))
                    for c in ORDER])          # (7, n)
        # majority vote over the run's samples
```

```
        votes = np.argmin(dall, axis=0)
        conf.loc[SHORT[k], SHORT[ORDER[int(np.bincount(votes).argmax())]]] += 1
conf.to_csv(os.path.join(TAB, "T7_run_confusion_matrix.csv"))


# sample-level accuracy
sample_conf = pd.DataFrame(0, index=[SHORT[k] for k in ORDER], columns=[SHORT[k] for k in ORDER])
for k in ORDER:
    for r in data[k]:
        pts = r[["I (milli a.u.)", "x", "y"]].values
        dall = np.stack([np.sqrt(np.einsum("ij,jk,ik->i",
                    pts - M["cent"][c], M["covinv"][c], pts - M["cent"][c]))
                    for c in ORDER])
        votes = np.argmin(dall, axis=0)
        for v, cnt in zip(*np.unique(votes, return_counts=True)):
            sample_conf.loc[SHORT[k], SHORT[ORDER[int(v)]]] += int(cnt)
sample_conf.to_csv(os.path.join(TAB, "T8_sample_confusion_matrix.csv"))
sample_acc = np.trace(sample_conf.values) / sample_conf.values.sum()
run_acc = np.trace(conf.values) / conf.values.sum()


json.dump(dict(sample_accuracy=float(sample_acc), run_accuracy=float(run_acc)),
          open(os.path.join(TAB, "classification_accuracy.json"), "w"), indent=1)
print(f"run-level accuracy {run_acc:.3f}, sample-level {sample_acc:.3f}")


# ---- trajectory-aware leave-one-out classification --------------------------
# A run is a TRAJECTORY z(t)=(I,x,y). Distance between a test run and a class
# template (run-average EXCLUDING the test run) = mean pointwise Mahalanobis
# distance under the pooled within-class covariance, over the common time span.
pinv_pooled = np.linalg.pinv(M["pooled"])
def traj_dist(runA, tmplB):
    n = min(len(runA), len(tmplB))
    d = runA[:n] - tmplB[:n]
    return float(np.mean(np.sqrt(np.einsum("ij,jk,ik->i", d, pinv_pooled, d))))


traj_conf = pd.DataFrame(0, index=[SHORT[k] for k in ORDER], columns=[SHORT[k] for k in ORDER])
traj_margin = []
for k in ORDER:
    for ridx, r in enumerate(data[k]):
        test = r[["I (milli a.u.)", "x", "y"]].values
        dists = {}
        for c in ORDER:
            if c == k:                          # leave-one-out template
                others = [q for qi, q in enumerate(data[c]) if qi != ridx]
                nmin = min(len(q) for q in others)
                tmpl = np.mean([q[["I (milli a.u.)", "x", "y"]].values[:nmin]
                                for q in others], axis=0)
            else:
                tmpl = avg[c][["I (milli a.u.)", "x", "y"]].values
            dists[c] = traj_dist(test, tmpl)
        pred = min(dists, key=dists.get)
        traj_conf.loc[SHORT[k], SHORT[pred]] += 1
        srt = sorted(dists.values())
        traj_margin.append(dict(Set=SHORT[k], Run=ridx + 1, Pred=SHORT[pred],
                                d_true=dists[k], d_best=srt[0], d_second=srt[1],
                                margin=srt[1] - srt[0]))
traj_conf.to_csv(os.path.join(TAB, "T9_trajectory_LOO_confusion.csv"))
pd.DataFrame(traj_margin).to_csv(os.path.join(TAB, "T9b_trajectory_LOO_details.csv"), index=False)
traj_acc = np.trace(traj_conf.values) / traj_conf.values.sum()
print(f"trajectory LOO accuracy {traj_acc:.3f}")


# ---- resubstitution trajectory classification (fingerprint distinctness) ----
resub_conf = pd.DataFrame(0, index=[SHORT[k] for k in ORDER], columns=[SHORT[k] for k in ORDER])
for k in ORDER:
    for r in data[k]:
        test = r[["I (milli a.u.)", "x", "y"]].values
        dists = {c: traj_dist(test, avg[c][["I (milli a.u.)", "x", "y"]].values)
                 for c in ORDER}
        resub_conf.loc[SHORT[k], SHORT[min(dists, key=dists.get)]] += 1
resub_conf.to_csv(os.path.join(TAB, "T10_trajectory_resub_confusion.csv"))
resub_acc = np.trace(resub_conf.values) / resub_conf.values.sum()
print(f"trajectory resubstitution accuracy {resub_acc:.3f}")


# ---- rate-based fingerprint features per run + LOO 1-NN ---------------------
feat_rows, Xf, Lf = [], [], []
for k in ORDER:
    for ridx, r in enumerate(data[k]):
        t = r["Time (s)"].values; I = r["I (milli a.u.)"].values
        x = r["x"].values; y = r["y"].values
        bI = np.polyfit(t, I, 1)[0]; bx = np.polyfit(t, x, 1)[0]; by = np.polyfit(t, y, 1)[0]
        byx = np.polyfit(x, y, 1)[0] if x.std() > 1e-6 else 0.0
        cI = np.polyfit(t, I, 2)[0]; cx = np.polyfit(t, x, 2)[0]
        sI = float(np.std(I - np.polyval(np.polyfit(t, I, 2), t)))
        feat_rows.append(dict(Set=SHORT[k], Run=ridx + 1, dI_dt=bI, dx_dt=bx,
                              dy_dt=by, slope_y_x=byx, curv_I=cI, curv_x=cx,
                              resid_sigma_I=sI))
        Xf.append([bI, bx, by, byx, cI, cx, sI]); Lf.append(SHORT[k])
feat_df = pd.DataFrame(feat_rows)
feat_df.to_csv(os.path.join(TAB, "T11_run_rate_fingerprints.csv"), index=False)
Xf = np.array(Xf); Lf = np.array(Lf)
Xs = (Xf - Xf.mean(0)) / Xf.std(0)
nn_correct = 0
for i in range(len(Lf)):
    dd = np.linalg.norm(Xs - Xs[i], axis=1); dd[i] = np.inf
    nn_correct += (Lf[np.argmin(dd)] == Lf[i])
nn_acc = nn_correct / len(Lf)
print(f"rate-fingerprint LOO 1-NN accuracy {nn_acc:.3f}")


json.dump(dict(sample_accuracy=float(sample_acc), run_accuracy=float(run_acc),
               trajectory_LOO_accuracy=float(traj_acc),
               trajectory_resub_accuracy=float(resub_acc),
               rate_fingerprint_LOO_1NN_accuracy=float(nn_acc)),
          open(os.path.join(TAB, "classification_accuracy.json"), "w"), indent=1)
```

```
# ============================================================ FIGURES ======
plt.rcParams.update({
    "font.family": "DejaVu Sans", "font.size": 9,
    "axes.grid": True, "grid.alpha": 0.25, "grid.linewidth": 0.5,
    "axes.spines.top": False, "axes.spines.right": False,
    "figure.dpi": 110,
})

# common 3-D axis limits over ALL raw runs
allI = np.concatenate([r["I (milli a.u.)"].values for k in ORDER for r in data[k]])
allx = np.concatenate([r["x"].values for k in ORDER for r in data[k]])
ally = np.concatenate([r["y"].values for k in ORDER for r in data[k]])
LIM = dict(x=(allx.min(), allx.max()), y=(ally.min(), ally.max()),
           I=(allI.min(), allI.max()))


# ---------------- Figure 1 : 3-D scatter, 7 panels + combined ----------------
fig = plt.figure(figsize=(14, 8))
for i, k in enumerate(ORDER):
    ax = fig.add_subplot(2, 4, i + 1, projection="3d")
    for j, r in enumerate(data[k]):
        ax.scatter(r["x"], r["y"], r["I (milli a.u.)"], s=2.2,
                   color=PAL[k], alpha=[0.85, 0.5, 0.3][j % 3],
                   depthshade=False,
                   marker=["o", "^", "s"][j % 3], linewidths=0)
    ax.set_xlim(LIM["x"]); ax.set_ylim(LIM["y"]); ax.set_zlim(LIM["I"])
    ax.set_xlabel("x (µm)", labelpad=-4, fontsize=7.5)
    ax.set_ylabel("y (µm)", labelpad=-4, fontsize=7.5)
    ax.set_zlabel("I (milli a.u.)", labelpad=-4, fontsize=7.5)
    ax.tick_params(pad=-2, labelsize=6.5)
    ax.set_title(LABEL[k], fontsize=8.5, pad=1)
    ax.view_init(elev=18, azim=-60)
ax = fig.add_subplot(2, 4, 8, projection="3d")
for k in ORDER:
    a = avg[k]
    ax.scatter(a["x"], a["y"], a["I (milli a.u.)"], s=2.5, color=PAL[k],
               alpha=0.7, depthshade=False, linewidths=0, label=SHORT[k])
ax.set_xlim(LIM["x"]); ax.set_ylim(LIM["y"]); ax.set_zlim(LIM["I"])
ax.set_xlabel("x (µm)", labelpad=-4, fontsize=7.5)
ax.set_ylabel("y (µm)", labelpad=-4, fontsize=7.5)
ax.set_zlabel("I (milli a.u.)", labelpad=-4, fontsize=7.5)
ax.tick_params(pad=-2, labelsize=6.5)
ax.set_title("All sets (run-averaged)", fontsize=8.5, pad=1)
ax.view_init(elev=18, azim=-60)
ax.legend(loc="upper left", fontsize=6, frameon=False,
          bbox_to_anchor=(-0.05, 1.02), handletextpad=0.1, markerscale=2.2)
fig.suptitle("Figure 1. 3-D back-scatter signatures (x, y, I) of all seven data sets on identical axes "
             "(markers ●/▲/■ = Runs 1/2/3)", fontsize=11, y=0.99)
fig.tight_layout(rect=(0, 0, 1, 0.96))
fig.savefig(os.path.join(FIG, "Fig1_3D_scatter_all_sets.png"), dpi=DPI)
plt.close(fig)
print("Fig1 done")


# ---------------- Figure 2 : y vs x regressions ------------------------------
fig, axes = plt.subplots(2, 4, figsize=(14, 6.5))
for i, k in enumerate(ORDER):
    ax = axes.flat[i]
    a = avg[k]; f = yx_fits[k]
    xg = np.linspace(a["x"].min(), a["x"].max(), 300)
    yg, hc, hp = poly_bands(f, xg)
    ax.scatter(a["x"], a["y"], s=4, color=PAL[k], alpha=0.45, linewidths=0)
    ax.plot(xg, yg, color="k", lw=1.4)
    ax.fill_between(xg, yg - hp, yg + hp, color=PAL[k], alpha=0.12, label="95% prediction")
    ax.fill_between(xg, yg - hc, yg + hc, color=PAL[k], alpha=0.35, label="95% confidence")
    mdl = ["Linear", "Quadratic", "Cubic"][f["deg"] - 1]
    ax.set_title(f"{LABEL[k]}\n{mdl}: $R^2$={f['r2']:.4f}, slope $b_1$={f['beta'][1]:.3f} "
                 f"[{f['ci_lo'][1]:.3f}, {f['ci_hi'][1]:.3f}]", fontsize=7.6)
    ax.set_xlabel("x (µm)", fontsize=8); ax.set_ylabel("y (µm)", fontsize=8)
    ax.tick_params(labelsize=7)
    if i == 0:
        ax.legend(fontsize=6.5, frameon=False, loc="upper left")
axes.flat[7].axis("off")
axes.flat[7].text(0.05, 0.6,
    "Best model chosen by AIC among\nlinear / quadratic / cubic fits.\n\n"
    "Shaded bands:\n dark = 95% CI of mean response\n light = 95% prediction band\n\n"
    "The y–x slope encodes the beam-steering\ntrajectory on the 2-D PSD and is\n"
    "flow/pressure-signature specific.",
    fontsize=8.5, va="center")
fig.suptitle("Figure 2. Best-fit y–x regressions of the position-sensing-diode trajectories (run-averaged), "
             "with 95% confidence and prediction bands", fontsize=11)
fig.tight_layout(rect=(0, 0, 1, 0.94))
fig.savefig(os.path.join(FIG, "Fig2_yx_regressions.png"), dpi=DPI)
plt.close(fig)
print("Fig2 done")


# ---------------- Figure 3 : I vs C calibration curves -----------------------
gas_panels = [(k, col, lab) for k in ORDER for col, lab in CONC_COLS[k]]
fig, axes = plt.subplots(3, 3, figsize=(12.5, 10))
for i, (k, col, lab) in enumerate(gas_panels):
    ax = axes.flat[i]
    a = avg[k]; f = ic_fits[(k, lab)]
    C = a[col].values; I = a["I (milli a.u.)"].values
    cg = np.linspace(C.min(), C.max(), 300)
    yg, hc, hp = poly_bands(f, cg)
    ax.scatter(C, I, s=4, color=PAL[k], alpha=0.45, linewidths=0)
    ax.plot(cg, yg, color="k", lw=1.4)
    ax.fill_between(cg, yg - hp, yg + hp, color=PAL[k], alpha=0.12)
    ax.fill_between(cg, yg - hc, yg + hc, color=PAL[k], alpha=0.35)
    mdl = ["Linear", "Quadratic", "Cubic"][f["deg"] - 1]
    ax.set_title(f"{SHORT[k]} – {lab}\n{mdl}: $R^2$={f['r2']:.4f}, "
                 f"$b_1$={f['beta'][1]:.3f} [{f['ci_lo'][1]:.3f}, {f['ci_hi'][1]:.3f}]",
                 fontsize=8)
```

```
    ax.set_xlabel(f"{lab}", fontsize=8.5)
    ax.set_ylabel("I (milli a.u.)", fontsize=8.5)
    ax.tick_params(labelsize=7)
fig.suptitle("Figure 3. Back-scattered intensity I versus injected trace-gas concentration (run-averaged),\n"
             "with best-fit calibration models, 95% confidence (dark) and prediction (light) bands",
             fontsize=10.5)
fig.tight_layout(rect=(0, 0, 1, 0.95))
fig.savefig(os.path.join(FIG, "Fig3_I_vs_C_calibrations.png"), dpi=DPI)
plt.close(fig)
print("Fig3 done")


# ---------------- Figure 4 : Mahalanobis statistics --------------------------
fig = plt.figure(figsize=(13.5, 9.4))
gs = fig.add_gridspec(2, 2, width_ratios=[1.1, 1.0], wspace=0.28, hspace=0.42)


axh = fig.add_subplot(gs[0, 0])
im = axh.imshow(cc.values, cmap="viridis")
axh.set_xticks(range(7), cc.columns, rotation=45, ha="right", fontsize=7.5)
axh.set_yticks(range(7), cc.index, fontsize=7.5)
for r in range(7):
    for c in range(7):
        v = cc.values[r, c]
        axh.text(c, r, f"{v:.1f}", ha="center", va="center", fontsize=6.4,
                 color="white" if v < cc.values.max() * 0.6 else "black")
axh.set_title("(a) Centroid-to-centroid $d_M$\n(pooled covariance)", fontsize=9)
axh.grid(False)
fig.colorbar(im, ax=axh, shrink=0.85, label="$d_M$")


axb = fig.add_subplot(gs[0, 1])
xpos = np.arange(7)
means = [M["cross"][(k, "AMBIENT")].mean() for k in ORDER]
mins  = [M["cross"][(k, "AMBIENT")].min() for k in ORDER]
maxs  = [M["cross"][(k, "AMBIENT")].max() for k in ORDER]
axb.bar(xpos, means, width=0.62, color=[PAL[k] for k in ORDER], alpha=0.9)
axb.errorbar(xpos, means,
             yerr=[np.array(means) - np.array(mins), np.array(maxs) - np.array(means)],
             fmt="none", ecolor="k", elinewidth=1, capsize=3)
for xi, (mn, mx, me) in enumerate(zip(mins, maxs, means)):
    axb.annotate(f"{me:.1f}", (xi, me), textcoords="offset points",
                 xytext=(0, 4), ha="center", fontsize=7)
axb.set_xticks(xpos, [SHORT[k] for k in ORDER], fontsize=8)
axb.set_ylabel("$d_M$ to Ambient-control cluster", fontsize=8.5)
axb.set_title("(b) Sample $d_M$ to Ambient cluster\n(bar = mean; whiskers = min–max)", fontsize=9)


axc = fig.add_subplot(gs[1, 0])
imc = axc.imshow(sample_conf.values / sample_conf.values.sum(axis=1, keepdims=True),
                 cmap="Blues", vmin=0, vmax=1)
axc.set_xticks(range(7), sample_conf.columns, rotation=45, ha="right", fontsize=7.5)
axc.set_yticks(range(7), sample_conf.index, fontsize=7.5)
for r in range(7):
    for c in range(7):
        frac = sample_conf.values[r, c] / sample_conf.values[r].sum()
        if frac > 0.005:
            axc.text(c, r, f"{frac*100:.0f}", ha="center", va="center",
                     fontsize=6.4, color="white" if frac > 0.6 else "black")
axc.set_title(f"(c) Sample-level Mahalanobis classification (%)\n"
              f"overall accuracy = {sample_acc*100:.1f}%; run-level = {run_acc*100:.0f}%",
              fontsize=9)
axc.set_xlabel("Assigned cluster", fontsize=8); axc.set_ylabel("True set", fontsize=8)
axc.grid(False)


axd = fig.add_subplot(gs[1, 1])
imd = axd.imshow(traj_conf.values / traj_conf.values.sum(axis=1).reshape(-1, 1),
                 cmap="Greens", vmin=0, vmax=1)
for r in range(7):
    for c in range(7):
        v = traj_conf.values[r, c]
        if v > 0:
            frac = v / traj_conf.values[r].sum()
            axd.text(c, r, f"{v}", ha="center", va="center", fontsize=8,
                     color="white" if frac > 0.6 else "black")
axd.set_xticks(range(7), traj_conf.columns, rotation=45, ha="right", fontsize=7.5)
axd.set_yticks(range(7), traj_conf.index, fontsize=7.5)
axd.set_title(f"(d) Trajectory-aware leave-one-out run classification\n"
              f"(runs assigned; overall accuracy = {traj_acc*100:.0f}%)", fontsize=9)
axd.set_xlabel("Assigned set", fontsize=8); axd.set_ylabel("True set", fontsize=8)
axd.grid(False)
fig.suptitle("Figure 4. Mahalanobis cluster statistics of the seven (I, x, y) signatures", fontsize=11, y=0.995)
fig.savefig(os.path.join(FIG, "Fig4_mahalanobis.png"), dpi=DPI, bbox_inches="tight")
plt.close(fig)
print("Fig4 done")


# ---------------- Figure 5 : time-series repeatability -----------------------
fig, axes = plt.subplots(3, 1, figsize=(11, 8.5), sharex=True)
for k in ORDER:
    for j, r in enumerate(data[k]):
        for ax, col in zip(axes, ["I (milli a.u.)", "x", "y"]):
            ax.plot(r["Time (s)"], r[col], color=PAL[k], lw=0.7,
                    alpha=[0.95, 0.55, 0.35][j % 3],
                    ls=["-", "--", ":"][j % 3],
                    label=SHORT[k] if (j == 0 and col == "I (milli a.u.)") else None)
axes[0].set_ylabel("I (milli a.u.)"); axes[1].set_ylabel("x (µm)"); axes[2].set_ylabel("y (µm)")
axes[2].set_xlabel("Time (s)")
axes[0].legend(ncols=7, fontsize=7.5, frameon=False, loc="lower left")
fig.suptitle("Figure 5. Raw time histories of I, x, y for all runs of all data sets "
             "(line style –/--/·· = Runs 1/2/3): repeatability and beam-steering dynamics", fontsize=11)
fig.tight_layout(rect=(0, 0, 1, 0.96))
fig.savefig(os.path.join(FIG, "Fig5_time_histories.png"), dpi=DPI)
plt.close(fig)
print("Fig5 done")
```

```
# ---------------- Figure 6 : analysis flow-chart ----------------------------
fig, ax = plt.subplots(figsize=(8.6, 11))
ax.axis("off"); ax.set_xlim(0, 10); ax.set_ylim(0, 14)
steps = [
    ("1. Raw data ingestion\n19 CSV files · 7 data sets · 10 Hz sampling\n(Time, C(t), I, x, y)", "#0072B2"),
    ("2. Structural validation & QC\ncolumn checks · time-grid alignment ·\noutlier flagging (e.g. Set-1 R3 transient at t=5.4 s)",
"#0072B2"),
    ("3. Run averaging\nensemble mean of I, x, y over 3 (or 2) runs\non the common time grid", "#0072B2"),
    ("4. Repeatability audit\nrun-to-run RMS differences and Pearson r\nfor I, x, y (Table T2)", "#56B4E9"),
    ("5. Regression modelling\ny = f(x) and I = f(C): linear/quadratic/cubic OLS\nAIC model selection · 95% CIs on slopes & intercepts
·\nconfidence + prediction bands", "#009E73"),
    ("6. Feature-space construction\n3-D feature vectors (I, x, y) per set\ncentroids μ_k and covariances Σ_k", "#E69F00"),
    ("7. Mahalanobis cluster statistics\nd_M within/between clusters: mean, min, max, CIs\ncentroid separation matrix (pooled Σ)",
"#E69F00"),
    ("8. Classification\nnearest-centroid Mahalanobis rule ·\nsample-level and run-level (majority-vote) confusion matrices",
"#D55E00"),
    ("9. Decision\ndetect → discriminate → quantify (via I=f(C) inverse) →\nclassify unknown exhaled-breath sample", "#CC79A7"),
]
yy = 13.4
boxes = []
for txt, col in steps:
    h = 1.02
    b = FancyBboxPatch((1.0, yy - h), 8.0, h, boxstyle="round,pad=0.10",
                       fc="white", ec=col, lw=2)
    ax.add_patch(b)
    ax.text(5.0, yy - h / 2, txt, ha="center", va="center", fontsize=8.4)
    boxes.append((yy - h, yy))
    yy -= h + 0.52
for (b1, t1), (b2, t2) in zip(boxes[:-1], boxes[1:]):
    ax.add_patch(FancyArrowPatch((5.0, b1 - 0.10), (5.0, t2 + 0.10), arrowstyle="-|>",
                                 mutation_scale=16, color="#444444", lw=1.4))
ax.set_title("Figure 6. Analysis pipeline: from raw back-scatter records to disease-signature classification",
             fontsize=11, pad=12)
fig.tight_layout()
fig.savefig(os.path.join(FIG, "Fig6_flowchart.png"), dpi=DPI, bbox_inches="tight")
plt.close(fig)
print("Fig6 done")


# ---- persist fit summaries for document generation --------------------------
def fit_to_dict(f):
    return dict(deg=int(f["deg"]), beta=f["beta"].tolist(), se=f["se"].tolist(),
                ci_lo=f["ci_lo"].tolist(), ci_hi=f["ci_hi"].tolist(),
                r2=float(f["r2"]), adj_r2=float(f["adj_r2"]), aic=float(f["aic"]),
                rmse=float(f["rmse"]), n=int(f["n"]))
json.dump({k: fit_to_dict(v) for k, v in yx_fits.items()},
          open(os.path.join(TAB, "yx_best_fits.json"), "w"), indent=1)
json.dump({f"{k}|{lab}": fit_to_dict(v) for (k, lab), v in ic_fits.items()},
          open(os.path.join(TAB, "ic_best_fits.json"), "w"), indent=1)


# decimated (1 s) run-averaged data tables for supplementary
for k in ORDER:
    a = avg[k].iloc[::10]
    a.to_csv(os.path.join(TAB, f"S_{SHORT[k]}_run_averaged_1Hz.csv"), index=False)


print("ALL DONE")
```