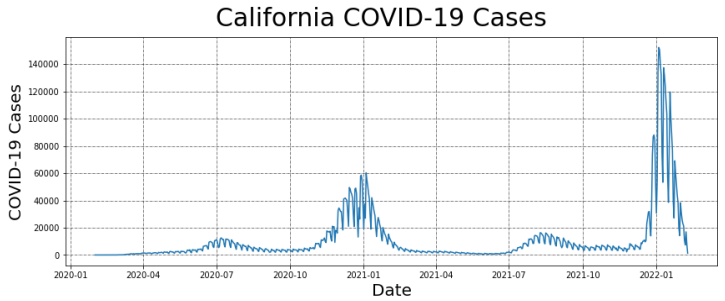
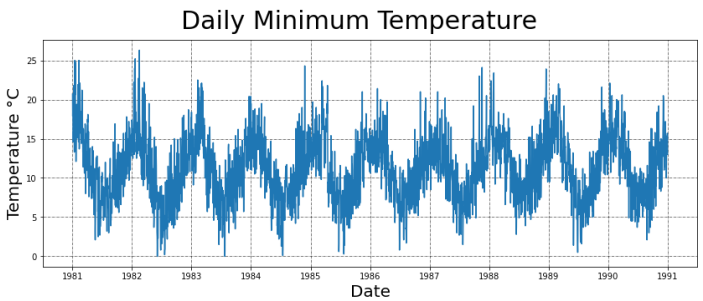
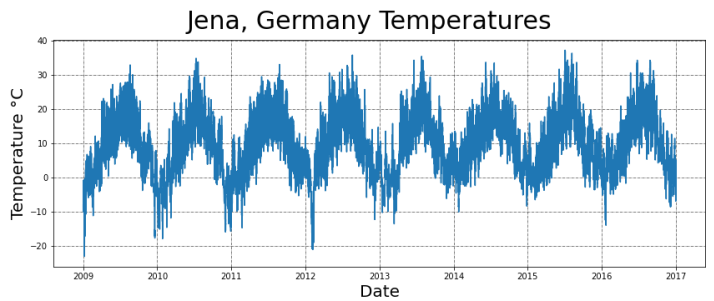
Matrix Profile MPI Implementation

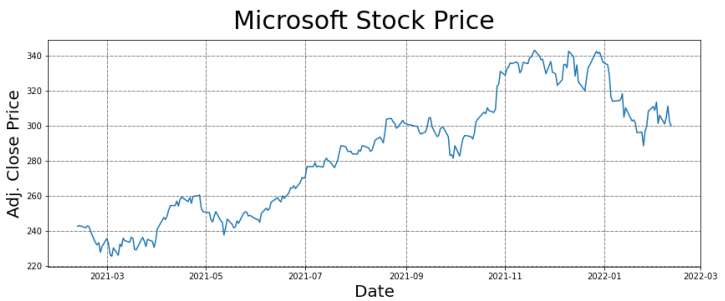
Brody Larsen* and Richard McNew†
Department of Computer Science
Utah State University
Logan, Utah, USA
* a01977457@usu.edu, † a02077329@usu.edu

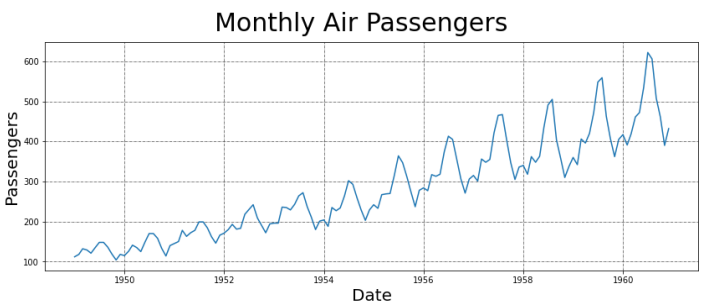
What is Time Series Data?

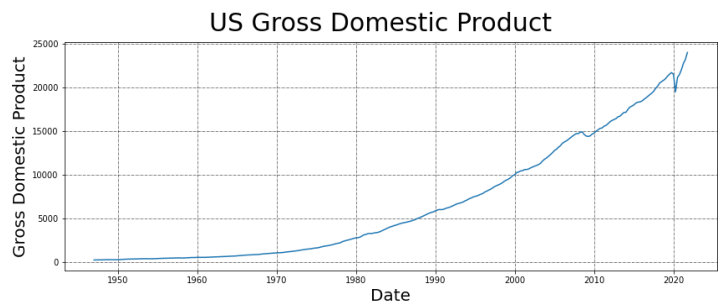






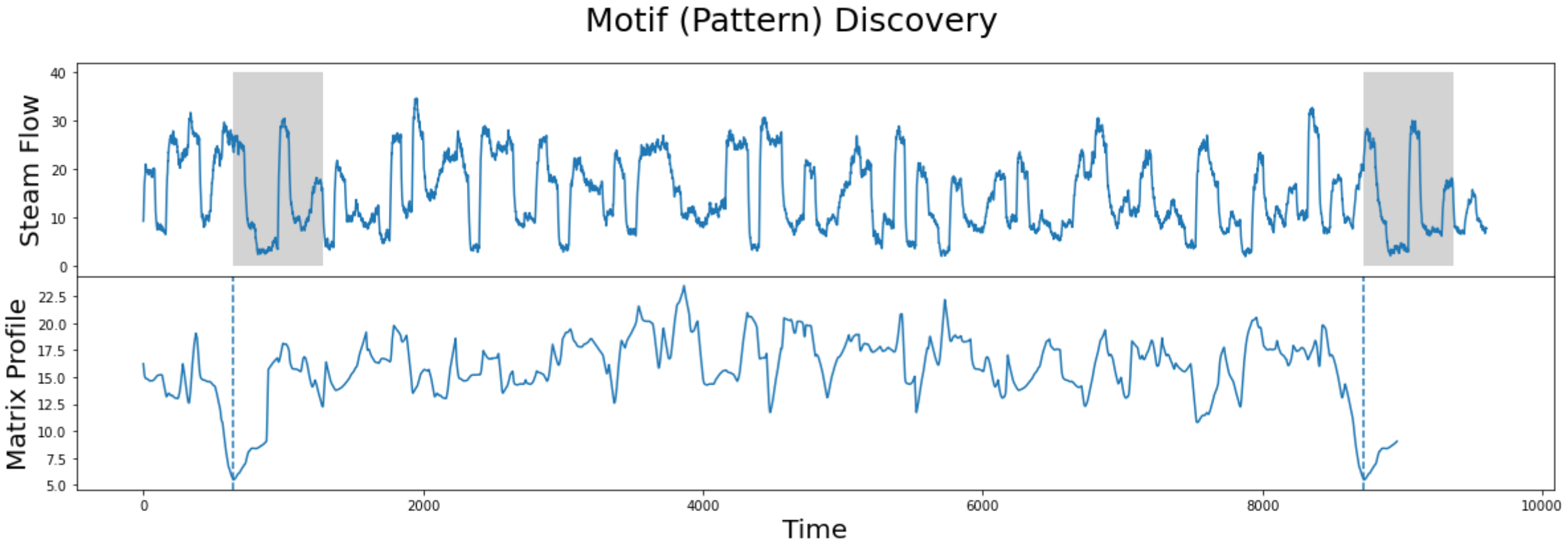






What is the Matrix Profile?

The Matrix Profile is a data structure and set of accompanying algorithms that annotate a time series and make most time series datamining easy to solve.



Matrix Profile vs. Other Tools?

1) exact - it allows for time series analysis without false positives or false negatives

2) parameter-free - unlike many time series data analysis tools, no hyperparameter tuning is needed

3) space efficient - a matrix profile data structure does not require much space, enabling large datasets to be processed in memory

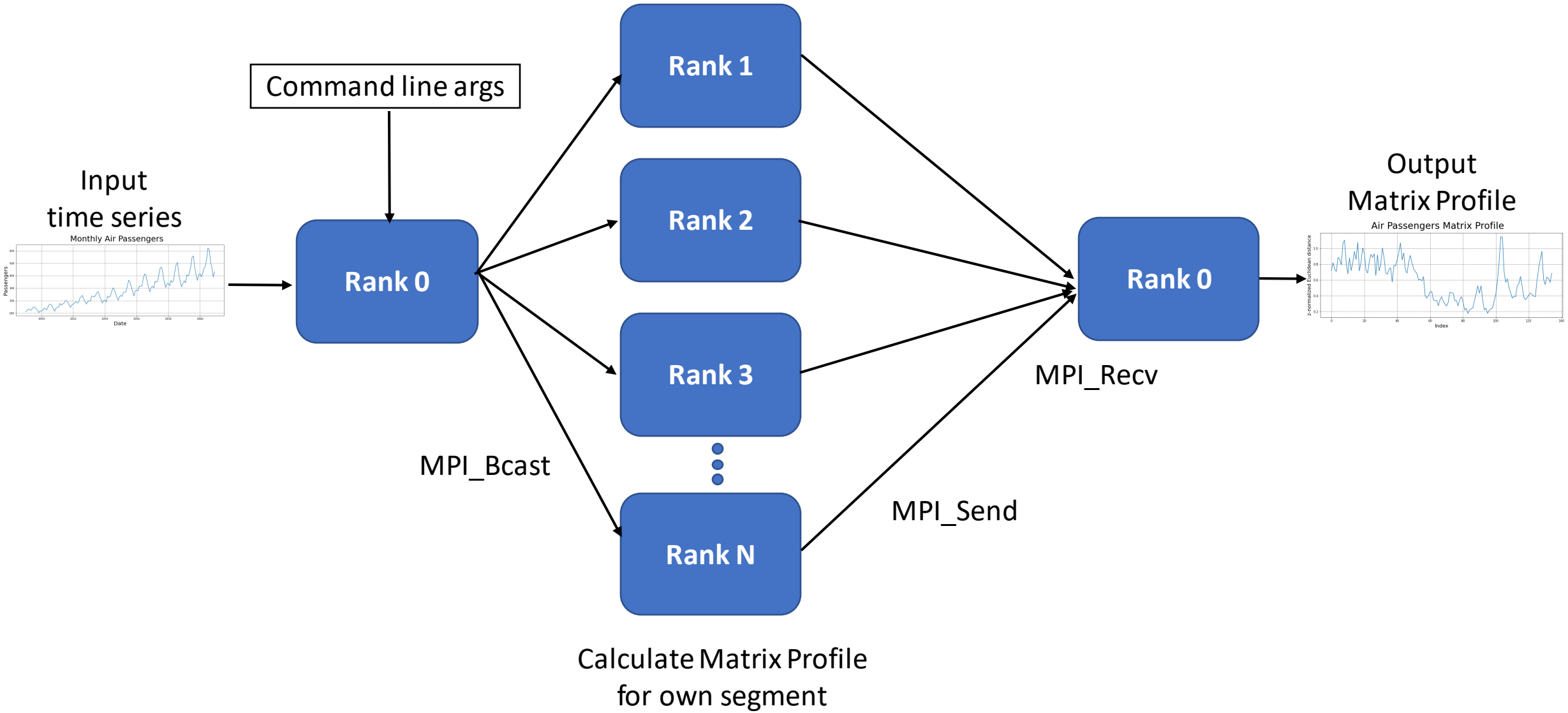
4) parallelizable - it is fast to compute on modern hardware

5) simple - it is easy to use and easy to understand.

Thesis

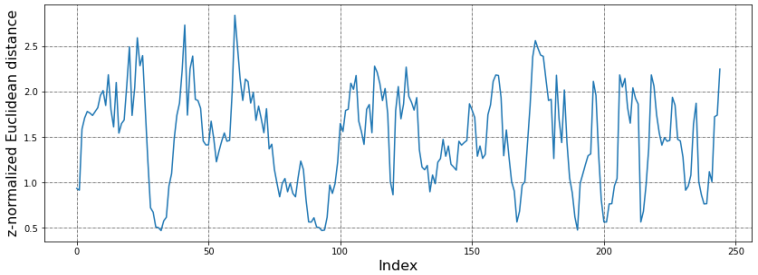
In this project we created a minimal MPI implementation of the Matrix Profile in C++.

Approach

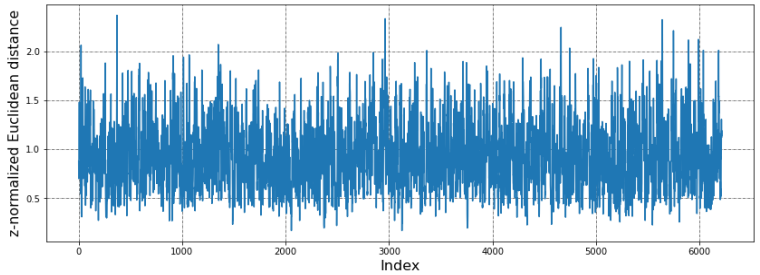

Command line args
Input
time series
Monthly Air Passengers
Passengers
Date
Rank 0
Rank 1
Rank 2
Rank 3
Rank N
MPI_Bcast
MPI_Send
Rank 0
MPI_Recv
Output
Matrix Profile
Air Passengers Matrix Profile
z-normalized Euclidean distance
Index
Calculate Matrix Profile
for own segment

Results

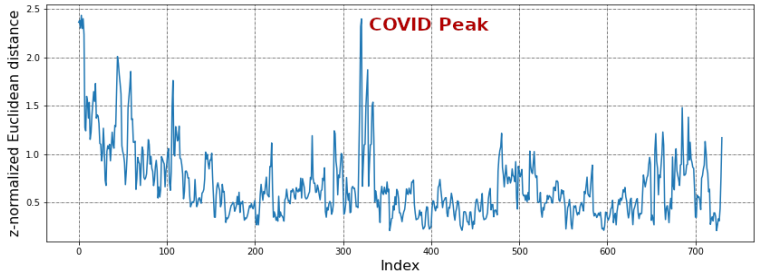
Apple Stock Price Matrix Profile



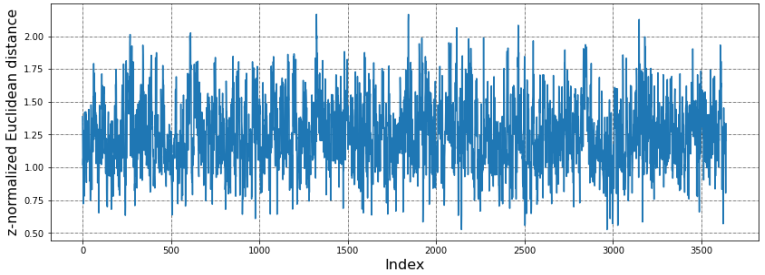
Amazon Stock Price Matrix Profile



California COVID-19 Cases Matrix Profile

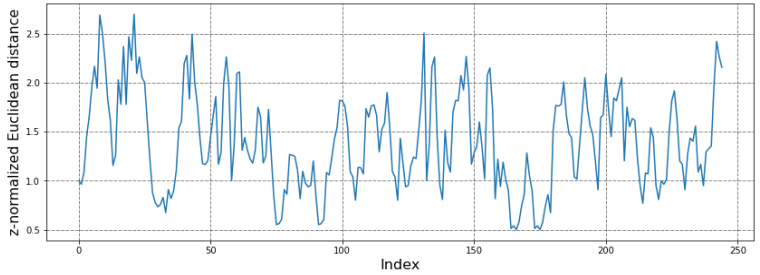


Daily Minimum Temperature Matrix Profile

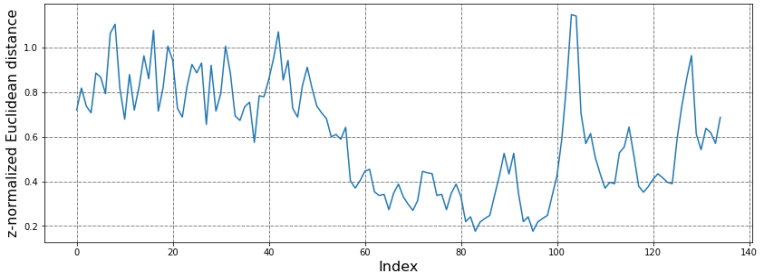


Jena Climate Matrix Profile did not finish in time

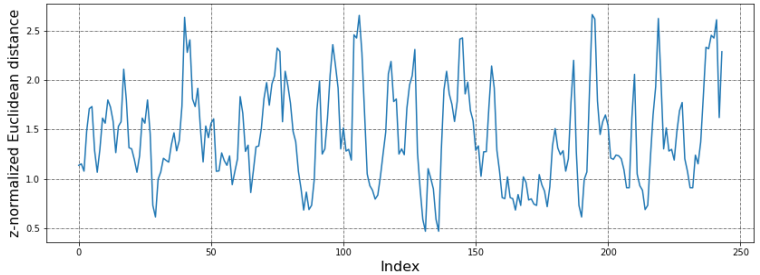
Microsoft Stock Price Matrix Profile



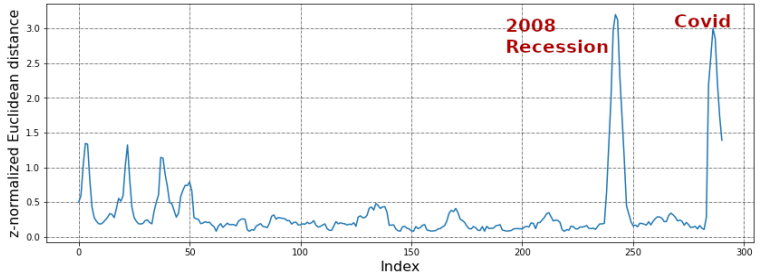
Air Passengers Matrix Profile



Tesla Stock Price Matrix Profile

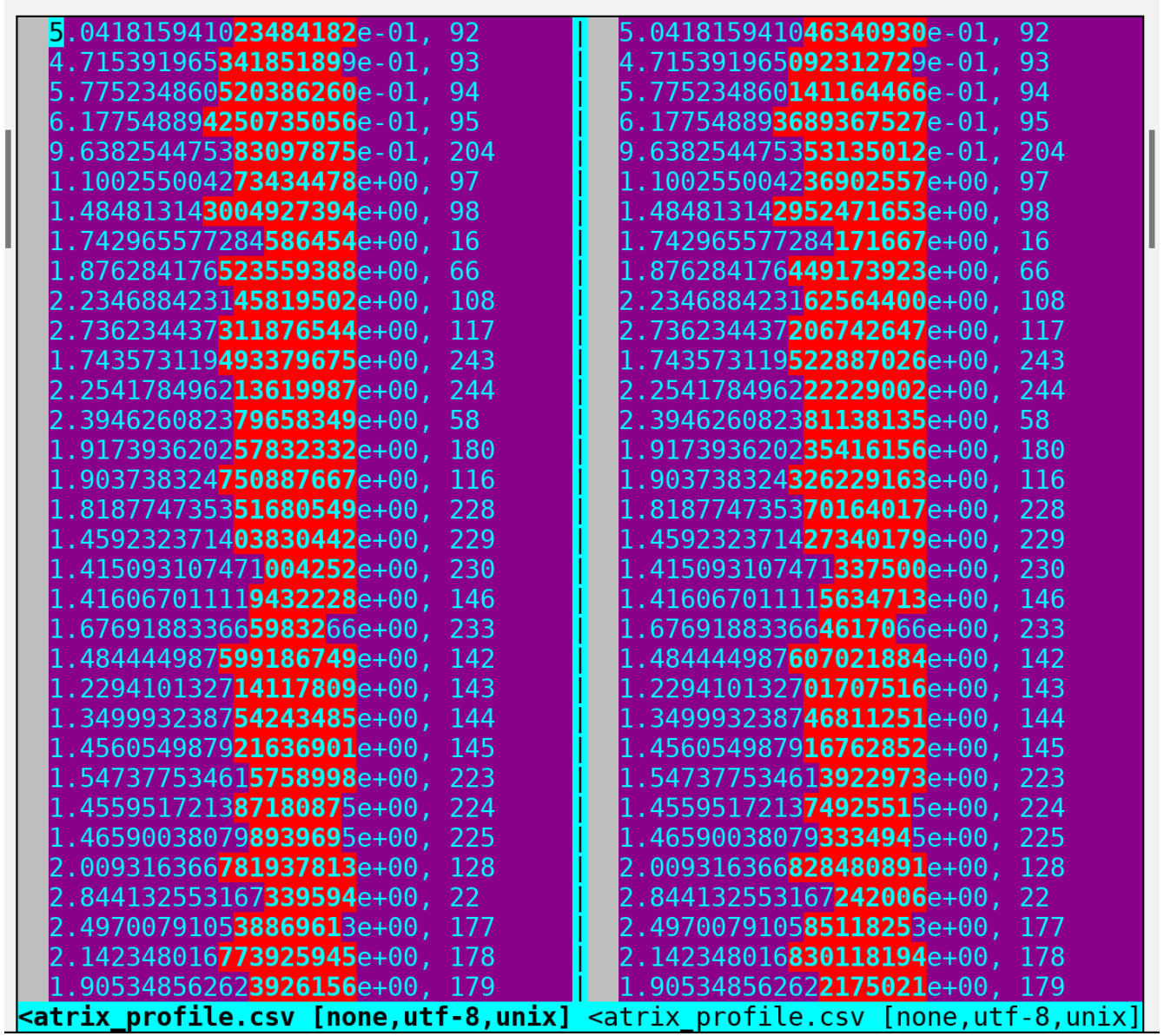


US Gross Domestic Product Matrix Profile

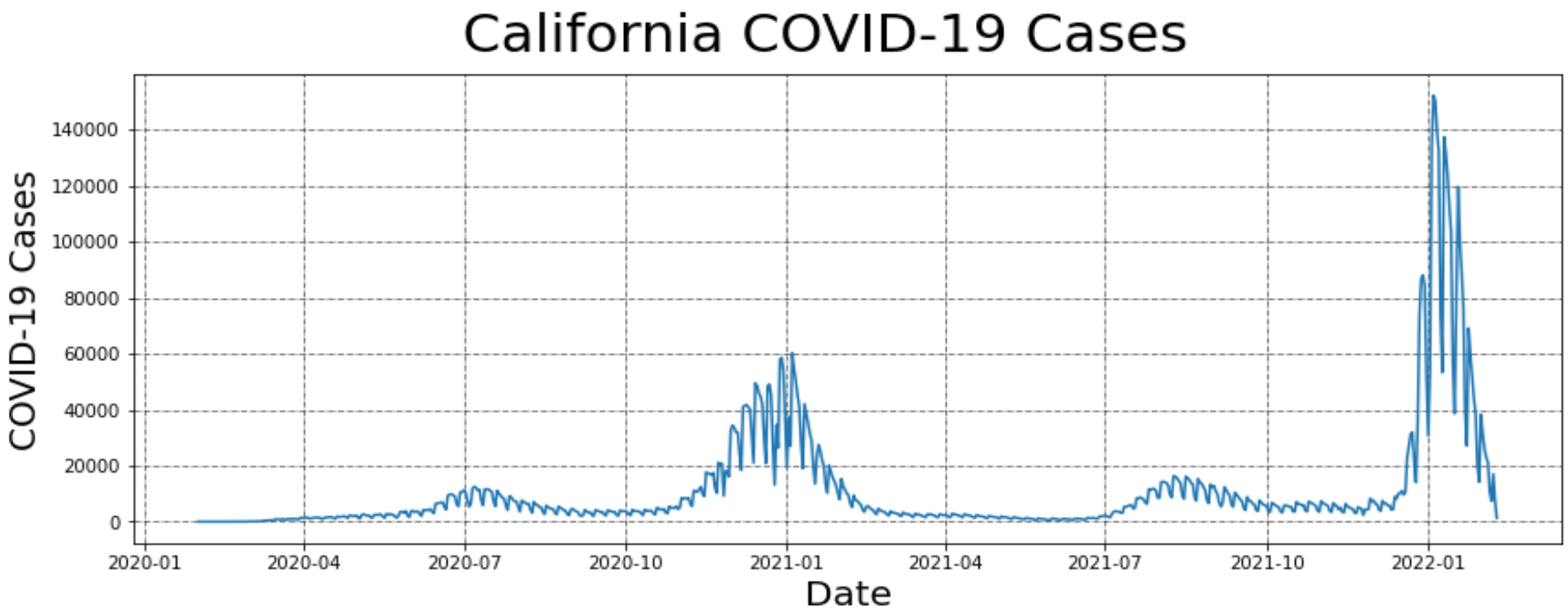


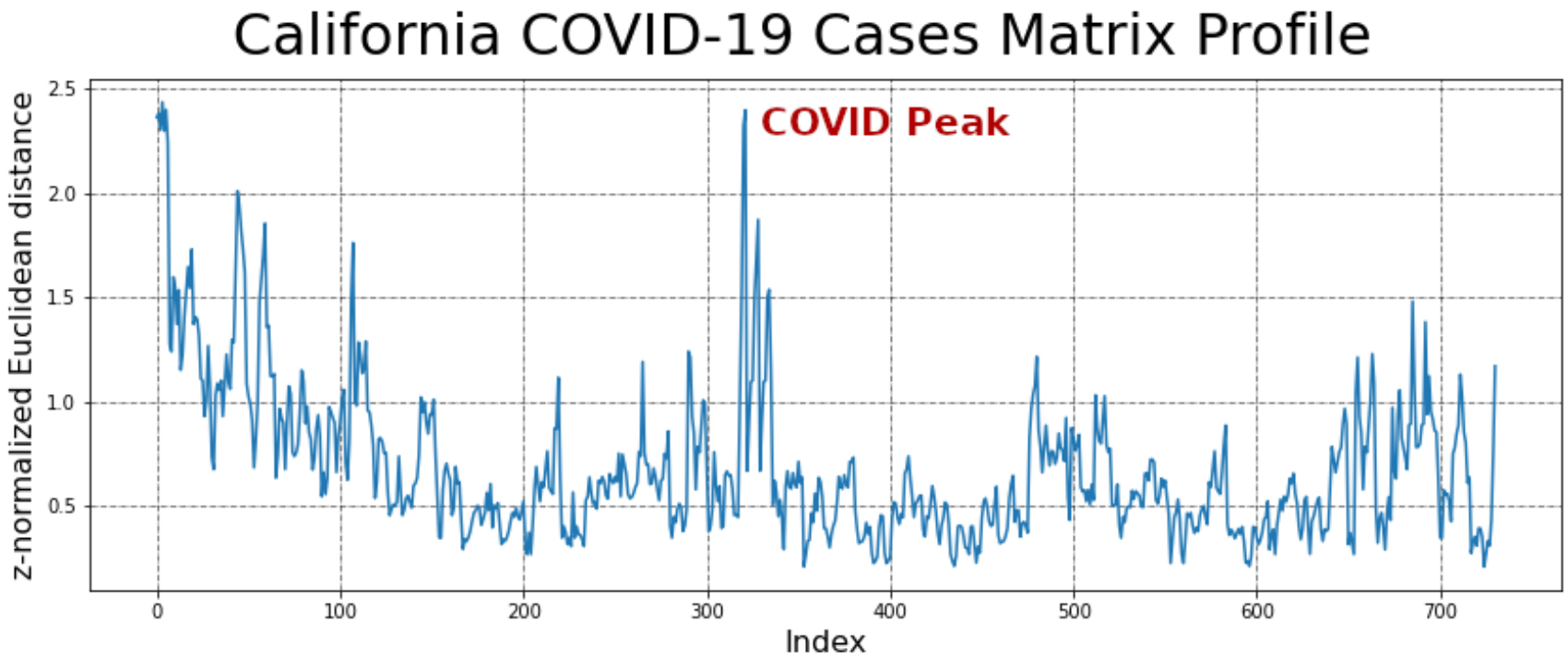
Side-by-Side Diff of Output Matrix Profiles

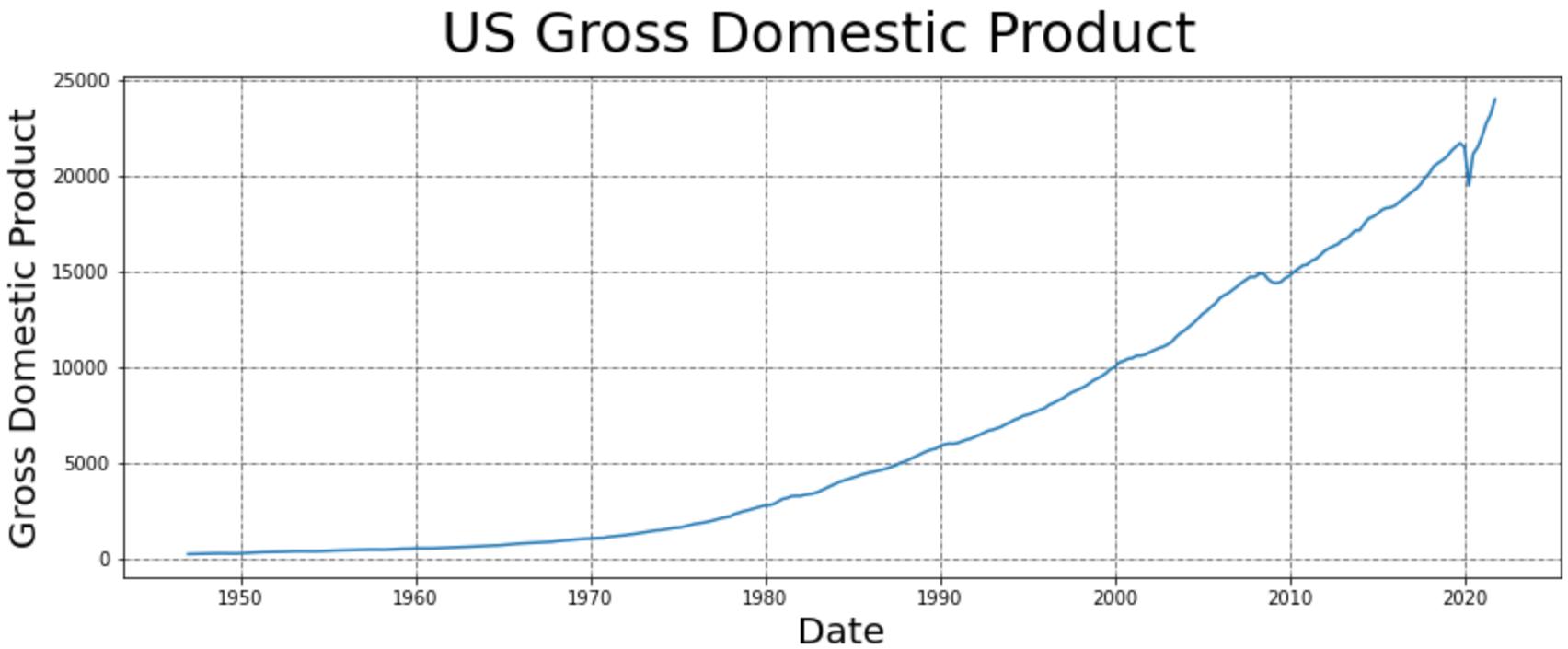
99.95 % Similarity

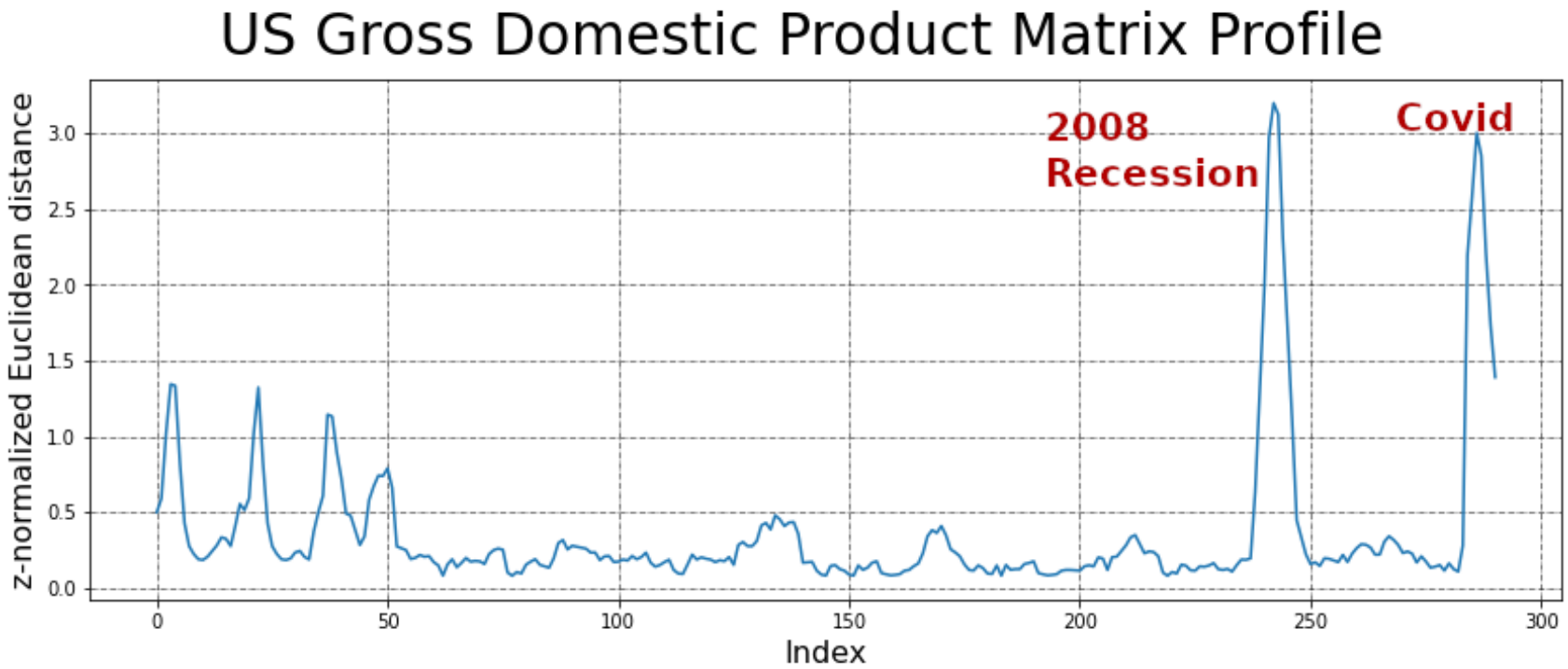


5.041815941023484182e-01, 92	5.041815941046340930e-01, 92
4.715391965341851899e-01, 93	4.715391965092312729e-01, 93
5.775234860520386260e-01, 94	5.775234860141164466e-01, 94
6.177548894250735056e-01, 95	6.177548893689367527e-01, 95
9.638254475383097875e-01, 204	9.638254475353135012e-01, 204
1.100255004273434478e+00, 97	1.100255004236902557e+00, 97
1.484813143004927394e+00, 98	1.484813142952471653e+00, 98
1.742965577284586454e+00, 16	1.742965577284171667e+00, 16
1.876284176523559388e+00, 66	1.876284176449173923e+00, 66
2.234688423145819502e+00, 108	2.234688423162564400e+00, 108
2.736234437311876544e+00, 117	2.736234437206742647e+00, 117
1.743573119493379675e+00, 243	1.743573119522887026e+00, 243
2.254178496213619987e+00, 244	2.254178496222229002e+00, 244
2.394626082379658349e+00, 58	2.394626082381138135e+00, 58
1.917393620257832332e+00, 180	1.917393620235416156e+00, 180
1.903738324750887667e+00, 116	1.903738324326229163e+00, 116
1.818774735351680549e+00, 228	1.818774735370164017e+00, 228
1.459232371403830442e+00, 229	1.459232371427340179e+00, 229
1.415093107471004252e+00, 230	1.415093107471337500e+00, 230
1.416067011119432228e+00, 146	1.416067011115634713e+00, 146
1.676918833665983266e+00, 233	1.676918833664617066e+00, 233
1.484444987599186749e+00, 142	1.484444987607021884e+00, 142
1.229410132714117809e+00, 143	1.229410132701707516e+00, 143
1.349993238754243485e+00, 144	1.349993238746811251e+00, 144
1.456054987921636901e+00, 145	1.456054987916762852e+00, 145
1.547377534615758998e+00, 223	1.547377534613922973e+00, 223
1.455951721387180875e+00, 224	1.455951721374925515e+00, 224
1.465900380798939695e+00, 225	1.465900380793334945e+00, 225
2.009316366781937813e+00, 128	2.009316366828480891e+00, 128
2.844132553167339594e+00, 22	2.844132553167242006e+00, 22
2.497007910538869613e+00, 177	2.497007910585118253e+00, 177
2.142348016773925945e+00, 178	2.142348016830118194e+00, 178
1.905348562623926156e+00, 179	1.905348562622175021e+00, 179
<atrix_profile.csv [none,utf-8,unix]	<atrix_profile.csv [none,utf-8,unix]


California COVID-19 Cases
COVID-19 Cases
140000
120000
100000
80000
60000
40000
20000
0
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07
2021-10
2022-01
Date


California COVID-19 Cases Matrix Profile
COVID Peak
z-normalized Euclidean distance
2.5
2.0
1.5
1.0
0.5
0
100
200
300
400
500
600
700
Index

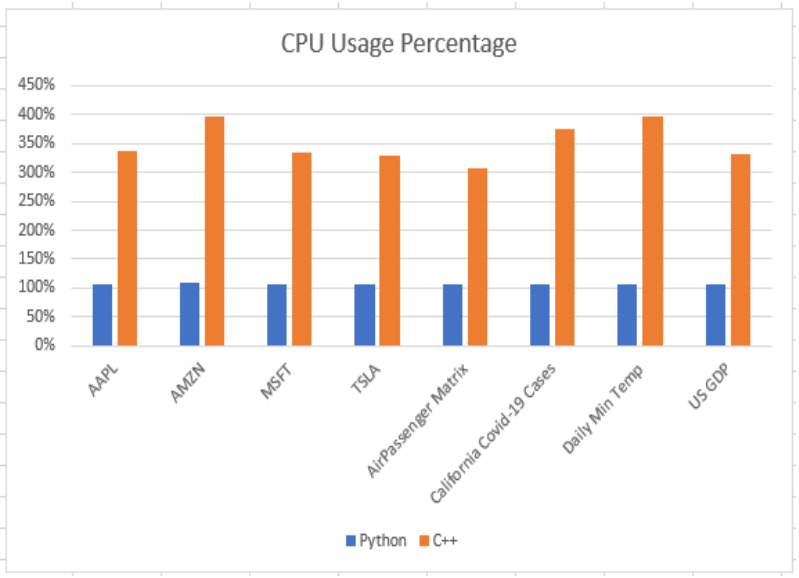

US Gross Domestic Product
Gross Domestic Product
25000
20000
15000
10000
5000
0
1950
1960
1970
1980
1990
2000
2010
2020
Date

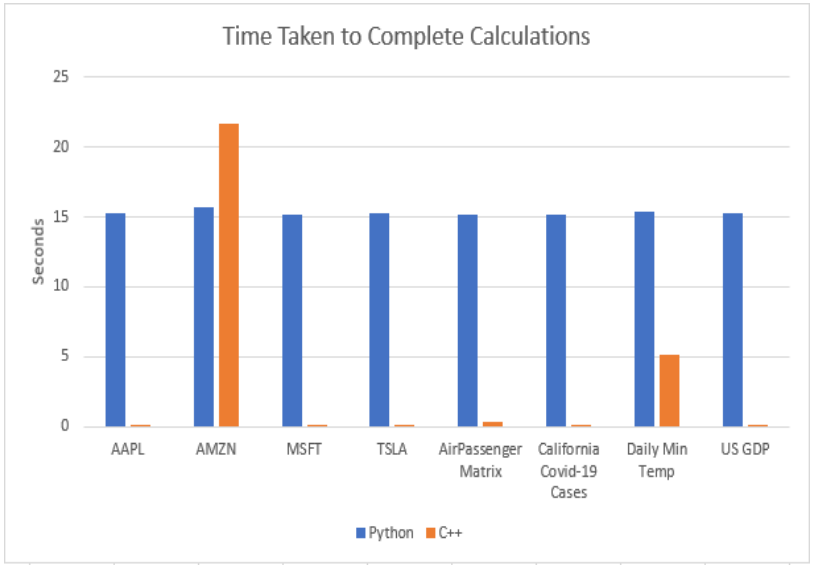

US Gross Domestic Product Matrix Profile
z-normalized Euclidean distance
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2008
Recession
Covid
0
50
100
150
200
250
300
Index

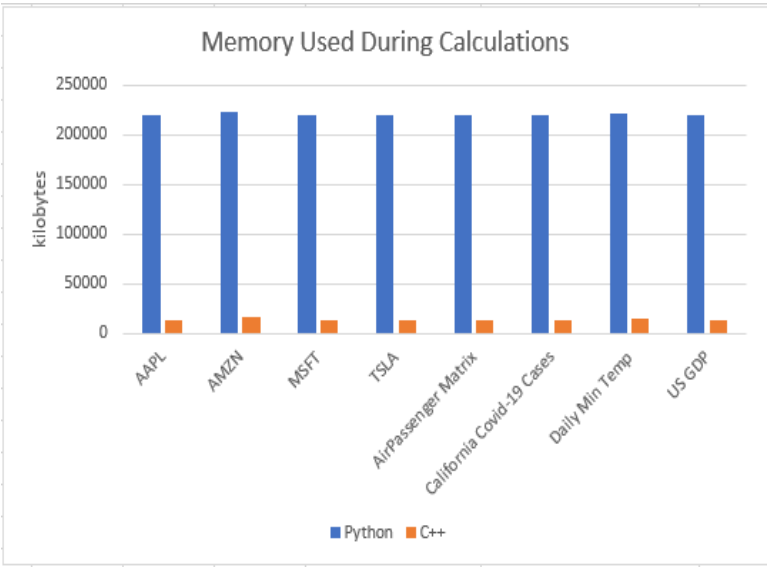
Conclusion

Our MPI C++ implementation worked faster than the Python implementation on smaller datasets.

It used more CPU and less memory than Python. However, since we used a slow algorithm, it took more time when the data set grew in size.





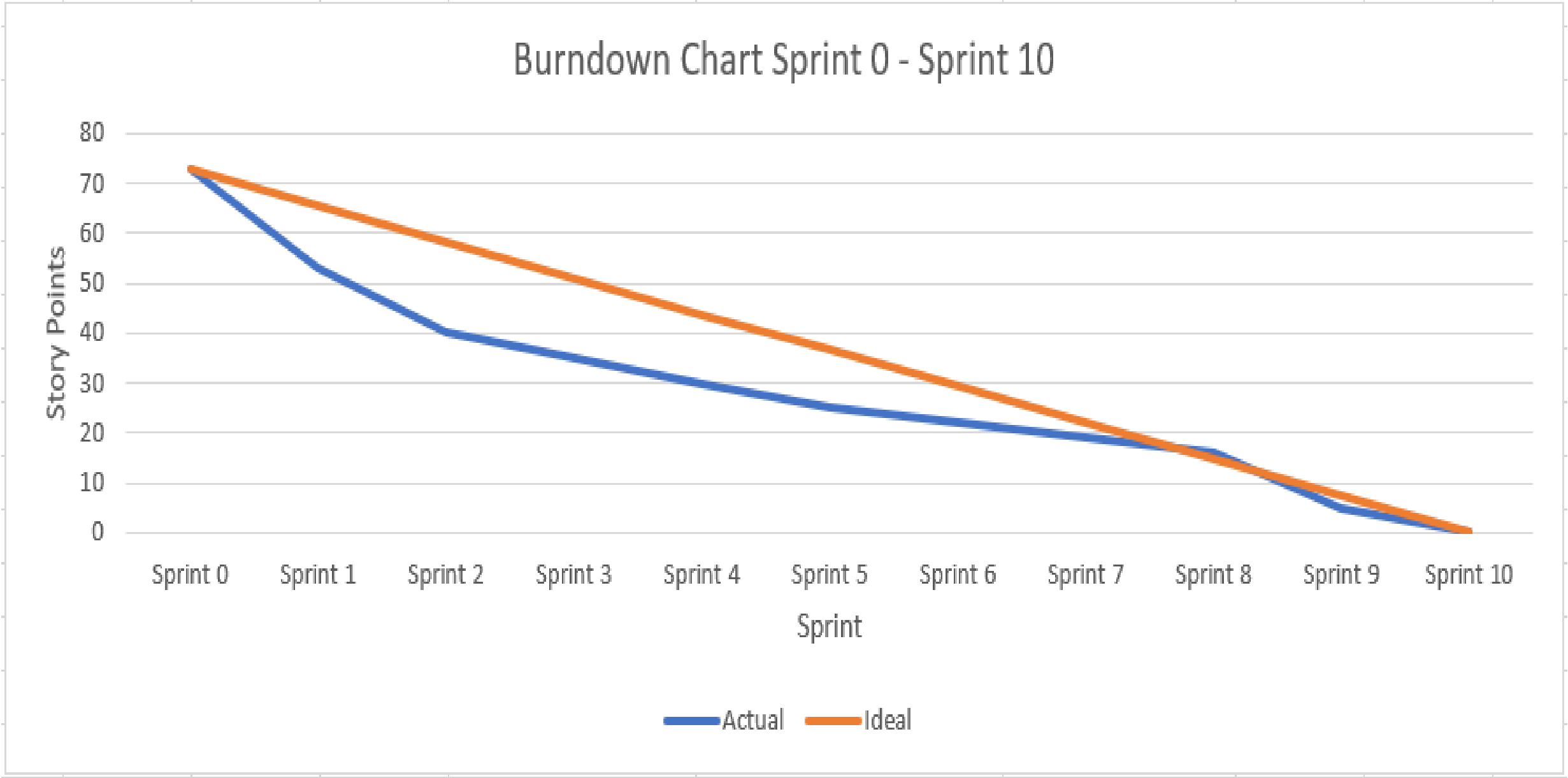


Task List

Tasks Part 1	Worked on	Story Points
Get contact information and availability shared to all group members	All	2
Decide on thesis for the project	Richard and Brody	2
Decide on a Meeting Plan	Richard and Brody	1
Create a draft proposal document	Richard and Brody	2
Review and revise draft proposal document	Richard and Brody	2
Turn in completed proposal document	Richard	1
Determine team member operating systems	Richard and Brody	1
Decide on project build system	Richard and Brody	2
Create a MPI C++ stub project and ensure everyone can build it	Richard and Brody	3
Choose a unit test framework and ensure it works for everyone	Richard and Brody	1
Ensure everyone can use version control, pull changes, push changes, etc.	Richard and Brody	1
Setup Continuous Integration on GitHub Actions	Richard	2
Collect time series data to use as test input data	Richard and Brody	3
Create a Python script to run the STUMPY Matrix Profile implementation against the test input time series data to generate Matrix Profile output data	Richard	2
Capture output Matrix Profile data for each of the test input time series	Richard and Brody	3
Study STUMPY Matrix Profile implementation and learn about Matrix Profile	Richard and Brody	5
Study Matrix Profile papers and slides	Richard and Brody	5
Research and find good MPI libraries to use	Richard and Brody	5
Incorporate MPI libraries found in Sprint_4 into CMake build	Richard	2
Create Work Breakdown Structure	Brody	3
Create Midpoint Report	Richard and Brody	3
Total		38

Task List

Tasks Part 2	Worked on	Story Points
Implement Mean and Standard Deviation function	Richard	2
Implement timing unit test	Richard	1
Implement Fourier Transformation and Inverse Fourier Transform Wrapper Functions	Richard	1
Parse and Validate command line arguments	Brody	1
Implement Unit Test for Logging	Richard	1
Implement Distance Profile	Richard	2
Implement Sliding Dot Product	Richard	3
Implement STAMP	Richard	3
Read Input Time Series from csv file	Richard and Brody	1
Write Matrix Profile to csv output file	Richard	2
Write Final Report	Richard and Brody	3
Create Presentation	Richard and Brody	2
Total		35


Burndown Chart Sprint 0 - Sprint 10
Story Points
80
70
60
50
40
30
20
10
0
Sprint 0
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5
Sprint 6
Sprint 7
Sprint 8
Sprint 9
Sprint 10
Sprint
Actual
Ideal

Questions?