CS8803-MDS

Project Update

Lecture 19
10/31/22

Logistics

Evaluation plan assignment is online

- Due 11/18 (Friday) 5PM, together with the final progress report
- 5% of grade
- Start early on parts 1-3

Presentation guidelines

Total time: 5min per group (timed)

What to cover:

- A quick reminder of your project
- What have you tried since the proposal? What worked and what failed?
- What are your next steps

Project group 12 - Sampling for Scatterplot Trend Exploration

- Scatterplots are a fundamental tool for data exploration tasks

- Scatterplotting large datasets has the potential to be both
  - Hard to interpret
  - Computationally intense

- There have been user studies of effectiveness of sampling techniques on outlier identification, shape examination, and density detection, **but not trend analysis!**

Project group 12 - Progress Update - Data

Synthetic Dataset using numpy

- Easy to generate, but…
- Not interesting, linear trends too easy to detect even with perturbation

Datasets in the wild (UCI Machine Learning Repository)

- Hard to find “correct” datasets, but..
- Linear trends between features are harder to detect but still surfaceable (i.e. more interesting)

Project Group 12 - Progress Update - Data

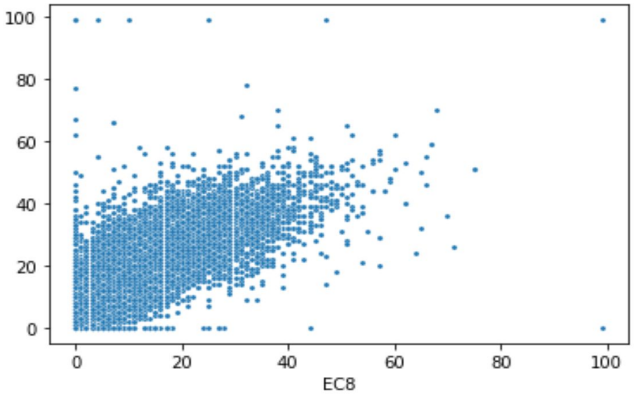
Dataset - KDD ‘98 Challenge Cup

- 114 MB
- ~95K donor records for a charity organization
- Number of features: 479
- Number of highly correlated pairs of features: 734

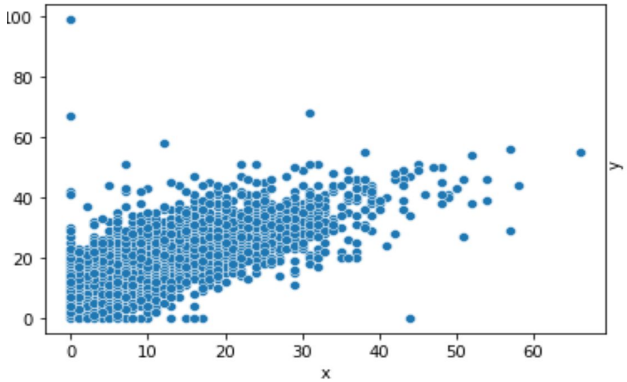
Scatterplot Sampling technique explored:

- Random Sampling
- Density-based Sampling
- Blue Noise Sampling
- Outlier-biased Random Sampling
- Outlier Biased Blue Noise Sampling
- Recursive Subdivision Based Sampling

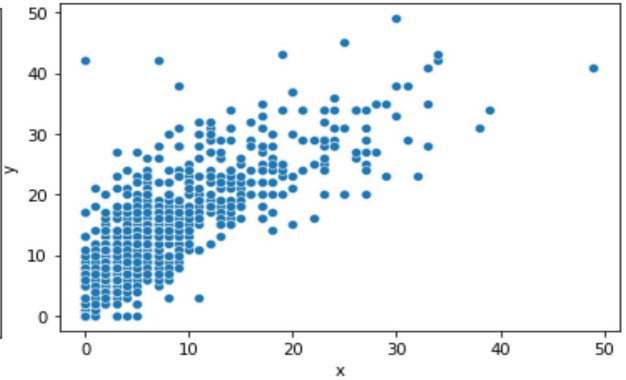
Project Group 12 - Progress Update



Original



Blue Noise



Random

Project Group 12 - Next Steps

- Explore more natural datasets. Ideally we should be generating plots from 2-3 different datasets to ensure external validity
- Explore metrics to evaluate user perception against the line-of-best fit (RMSE?)
- Come up with an user study design to compare the effectiveness of the scatterplot sampling techniques

Project group 11 - Multi-class NLP Generation

Is Dataset viable? Can it be classified?

BoW: 84%, CNN: 80%, Transformers (BERT): 83-84%

- 84% pretty good for BoW
- Unsure why CNN performs worst, expected BERT to perform better
  - CNN might be bad due to the fact that relationships between relevant words are not locally bounded (e.g. ‘you are the dumbest person I know’ can not convolve over ‘you’ and ‘dumbest’ with a size 2 kernel)
- “Other Cyberbullying” category is weird

index	tweet_text	cyberbullying_type
23916	@ikralla fyi, it looks like I was caught by it. I'm not a botter, so...	other_cyberbullying
23917	I need to just switch to an organization-based github, but I don't want to pay $25/month because I'm cheap. :\	other_cyberbullying
23918	RMAed my monoprice. Shoddy power bricks on those. Getting a refund and picking up another ASUS VG278HE. It's cheaper, anyways.	other_cyberbullying

Project group 11 - Next Steps

- Maybe don’t use other cyberbullying data labels, test out accuracies without ‘other cyberbullying’
- Start implementing natural language generation
- Might not have time for topic modelling
- Find an alternative to google colab

Project group 10

Evaluation of Data Models for Query Efficiency in Time Series

- Dataset - Hourly Stock Prices
  - 2.29GB
  - ~7500 stocks traded on NASDAQ
  - Date range: 2021-01-01 to 2021-12-30.
  - 500 - 5000 data points per stock
- Databases used
  - PostgreSQL
  - TimescaleDB
  - Redis
  - InfluxDB

ticker	name	date	open	high	low	close	adjusted cl	volume
A	Agilent Tec	2021-01-04	119.01	120.09	118.8	120.02	120.02	0
A	Agilent Tec	2021-01-04	120.02	120.03	118.41	118.86	118.86	341376
A	Agilent Tec	2021-01-04	118.84	118.868	117.27	117.57	117.57	108887
A	Agilent Tec	2021-01-04	117.5	118.26	117.11	118.1	118.1	90501
A	Agilent Tec	2021-01-04	118.12	118.17	117.75	118	118	114713
A	Agilent Tec	2021-01-04	118	118.22	117.7	118.2	118.2	132395
A	Agilent Tec	2021-01-04	118.2	118.76	118.015	118.66	118.66	250850
A	Agilent Tec	2021-01-04	118.64	118.64	118.6347	118.6347	118.6347	0
A	Agilent Tec	2021-01-05	118.48	120.32	118.065	119.59	119.59	284854
A	Agilent Tec	2021-01-05	119.59	119.64	118.9	119.595	119.595	172554

Project group 10

Evaluation of Data Models for Query Efficiency in Time Series

- Installed the 4 databases
- Preprocessed data to fit our pipeline
- Designed schemas mentioned in the proposal to store data
- Developed code to perform CRUD operations for data and record time taken
- Identified the visualizations we want to focus on

Project group 10

Evaluation of Data Models for Query Efficiency in Time Series

Next Steps:

- Perform data compression
- Design & write queries to prepare data for visualizations
  - Line Chart
  - Pie Chart
  - Histogram
- Collect metrics on data storage and performance
- Stretch Goal: Package this as a benchmark system for comparing RDBMS, NoSQL and Time Series DB.

Project group 9

Kernelized density estimation (KDE) from approximate nearest neighbor (ANN) candidates.

$$Y(q) = \sum_{x \in D} w(x) K(x, q)$$

The goal is to estimate Y(q) from ANNS candidates (some samples of x that are close to q). This approach is more efficient than existing ones, since we already know values of K(x,q) for the ANN query.

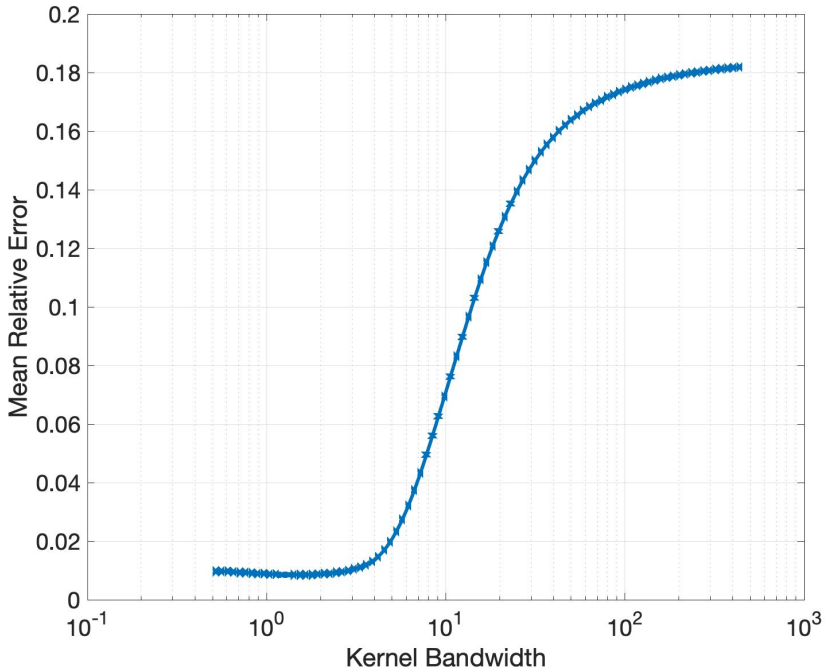
The question is: How accurate are the estimates given only a small fraction of ANN candidates?

Project group 9

Plain KDE setting: w(x) = 1 for all data points.

MNIST dataset (784-dim)

Our KDE is very accurate when bandwidth is smaller than some threshold.



Project group 9

Kernelized SVM setting: w(x) is the dual coefficient of x.

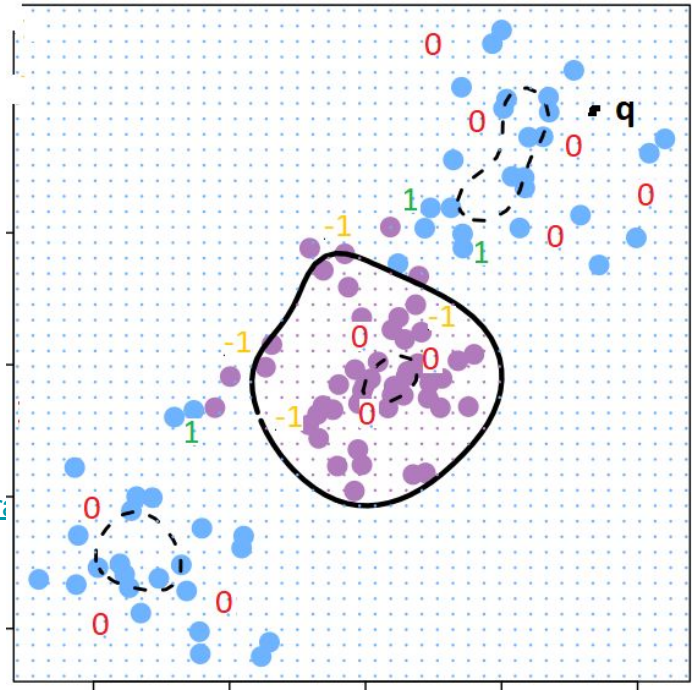
Our error is linear to the bandwidth (unbounded).

Although the two settings look similar,
in SVM, the weights are correlated with spatial position (whether a point is ANN candidate).

As a result, our method has systematic bias.

Figure adapted from the website:

6 Support vector machines | An Introduction to Machine Learning (bioinformatics-tr



Project group 9

Next steps:

1. Try kernelized ridge regression: the weights may be not as correlated to location as SVM.

2. Test other types of kernels (other than Gaussian RBF).

3. Try modelling the correlation between weight and location in our scheme.

Project group 8: Querying blockchain data

Summary

- Querying blockchain data is hard
  - It is inefficient (Linked list structure)
  - Query results need to be ‘verified’ for correctness before they can be accepted.

Project group 8: Querying blockchain data

Summary

- Querying blockchain data is hard
  - It is inefficient (Linked list structure)
  - Query results need to be ‘verified’ for correctness before they can be accepted.

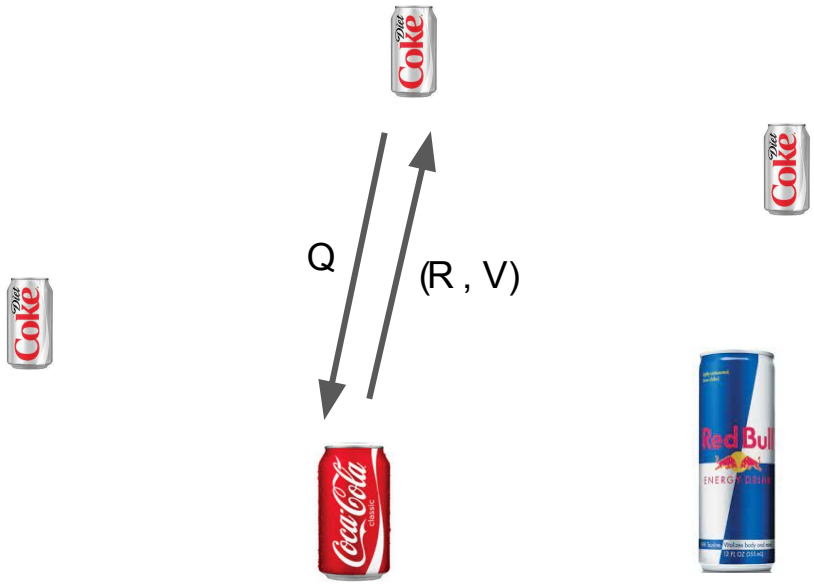
Query Result Verification

Lite nodes

Full nodes

Miner nodes

Query Result Verification



What does it mean to “verify” a query result?

1. Soundness
2. Completeness

Vector: Is this even possible without V being extremely large?

- Answer: Yes!
- vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases by C Xu et.al (SIGMOD 2019)

Next steps

- The vchain paper focused more on showing that verifiability was possible.
- Not much emphasis placed on efficiency
- Efficiency metrics
  - Time taken for the full node to generate result set and proof
  - Time taken for the client node to verify the result
  - Size of the proof
- It also supports very specific types of queries: boolean range queries.

OpenSea
Created by @rchen8 2 years ago

DEX metrics #Defi #DEX
Created by @hagaetc 3 years ago

DeFi users over time #DeFi
Created by @rchen8 3 years ago

NFT Project Dashboard #N
Created by @rantum 1 year ago

Olympus (OHM) #DeFi #Stab
Created by @shadow 2 years ago

Gas Prices by @alex_kr
Created by @kroeger0x 2 years ag

@rchen8 / OpenSea monthly volume (Ethereum)
forked from @hagaetc/OpenSea USD volume

75 ☆

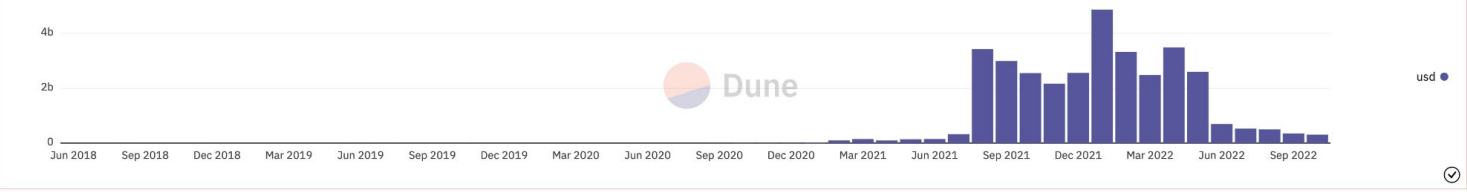
```
ITH token AS
(SELECT DISTINCT call_tx_hash AS tx_hash,
                 CASE
                     WHEN addrs[7] = '\x0000000000000000000000000000000000000000' THEN '\xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2'
                     ELSE addrs[7]
                 END AS token_address
 FROM opensea."WyvernExchange_call_atomicMatch_" a
 WHERE (addrs[4] = '\x5b3256965e7c3cf26e11fcaf296dfc8807c01073'
        OR addrs[11] = '\x5b3256965e7c3cf26e11fcaf296dfc8807c01073')
   AND call_success
   AND call_block_time > '2022-06-01'),
   excluded_txns AS
(SELECT call_tx_hash
 FROM opensea."WyvernExchange_call_atomicMatch_" a
 WHERE (addrs[4] = '\x5b3256965e7c3cf26e11fcaf296dfc8807c01073'
        OR addrs[11] = '\x5b3256965e7c3cf26e11fcaf296dfc8807c01073')
   AND call_success
   AND call_block_time >= '2022-06-01'
 GROUP BY 1
 HAVING count(DISTINCT CASE
                       WHEN addrs[7] = '\x0000000000000000000000000000000000000000' THEN '\xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2'
                       ELSE addrs[7]
                   END) > 1)
ELECT MONTH,
     sum(usd) AS usd
ROM
```

Table | Bar chart

Last run **2 hours ago**
Last run took **1 minute**

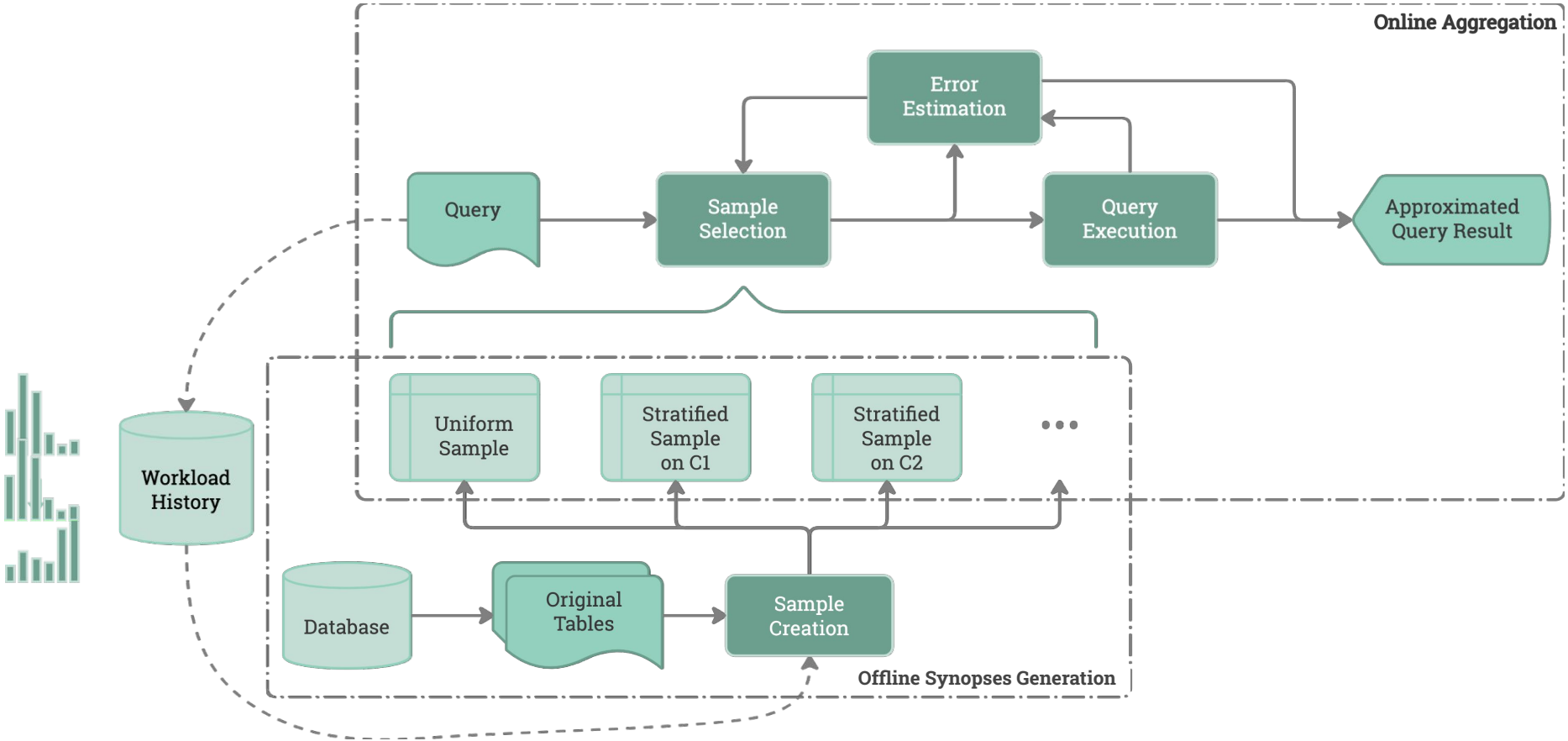
OpenSea monthly volume (Ethereum)

@rchen8



Project Group 7

- Original Topic: Adaptive Sampling and Aggregation Precomputation on Distributed Databases
  - Too broad
  - Lack technical innovation (research components)
  - Infeasible within one semester (many technical blockers anticipated)
- New Topic: Workload-Aware Adaptive On-Disk Sample Update for Online AQP


Online Aggregation
Error Estimation
Query
Sample Selection
Query Execution
Approximated Query Result
Uniform Sample
Stratified Sample on C1
Stratified Sample on C2
...
Workload History
Database
Original Tables
Sample Creation
Offline Synopses Generation

Project Overview

- Original Topic: Adaptive Sampling and Aggregation Precomputation on Distributed Databases
  - Too broad
  - Lack technical innovation (research components)
  - Infeasible within one semester (many technical blockers anticipated)
- New Topic: Workload-Aware Adaptive On-Disk Sample Update for Online AQP
  - Inspiration: BlinkDB creates sample at system launch time and never updates them afterwards
  - Question: Can the performance be maintained if the workload changes dramatically (e.g., user querying entirely different subsets of columns)? How can we optimize the offline-created sample storage for such scenario?
  - Proposed Solution: Periodically update on-disk samples by observing the workload change

What We've Tried: Benchmarks

> *Is there such scenario where attribute query frequency in the workload may change over time?*

- Investigated popular analytical benchmarks such as TPC-H/TPC-DS
  - Template-based, designed for traditional performance reporting scenarios where all plots (targeted query columns) are predefined
- In workloads for **interactive data exploration (IDE)**, workload might keep changing.
  - Queries are built incrementally[1]; target attribute sets might shift between each exploration stage

> *How to find / synthesize benchmarks that reflects typical workload of IDE?*

- **IDEBench[1]**: a customizable benchmark for evaluating db engines based on common IDE workflows and metrics
  - workload generator + data generator with scaler, given a real-world dataset

[1] Eichmann, Philipp, et al. "Idebench: A benchmark for interactive data exploration." *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020.

What We've Tried: Baseline AQP System

> *Which AQP engine/system should we build upon?*

- BlinkDB[1]
  - Samples created when system starts, given some "historical workload" (workload-driven)
  - Challenging to get it work:
    - Out-of-date
    - Many hard-coded dependencies
- VerdictDB[2]
  - Samples created when system starts, based on "column cardinality" (database-driven, workload-agnostic)
  - Successfully built and passed tests

[1] Agarwal, Sameer, et al. "BlinkDB: queries with bounded errors and bounded response times on very large data." Proceedings of the 8th ACM European Conference on Computer Systems. 2013.
[2] Park, Yongjoo, et al. "VerdictDB: Universalizing approximate query processing." Proceedings of the 2018 International Conference on Management of Data. 2018.

Next Steps

> *How to update the offline samples? How often?*

- Design and implement some sample update mechanisms
- Integrate the sample update module into the VerdictDB workflow

> *How does introducing workload-aware sample updates affect query efficiency and accuracy? What about storage and pre-computation cost?*

- Configure IDEBench and generate data and workloads based on real-world dataset
- Evaluate the performance of the proposed mechanisms on IDEBench, compare with baselines (VerdictDB, VerdictDB + vanilla workload-aware sample initialization, …)
- Experiment with different hyperparameters (e.g. update period, time limit, storage budgets, etc.)

> *How to optimize the on-disk storage layout on replacement?*

Team 6 - SQL Auto-Completion with Reinforcement Learning

Cangdi, Ting, Yiheng

Major updates

- **Construct a basic Q-learning model**
  - We implemented a basic Q-learning model that runs on a vector of integers, and expect to also run a synthetic sequence of related queries to see how it performs. We will decide whether we need a DAG later.
- **Prepare a query training dataset**
  - Sloan Digital Sky Survey (SDSS), but need to apply further processing
  - Found a paper by Microsoft Research that introduced methods for cleaning and preparing the Sloan Digital Sky Survey query dataset.

Major challenges

- **Cos-Similarity might not be a good evaluator**
- **Cleaning the SDSS dataset**

Q Learning - The Model Performance So far

With a training set of 30 randomly generated integer vectors that of various length, the model predicts the next vector with an accuracy of about 93% (training result). This basic tests shows that such a Q-Learning model might be able to be applied to Query-prediction.

It took just under 3 minutes (about 152 seconds) to train the model in 10M steps.

Q Learning - Cosine Similarity

Pros:

- It is very **efficient**. For average vector length n, finding the cos-similarity takes an amortized cost of O(n). This is significantly better than other popular evaluators (e.g., recall and precision each take O(n^2)).

Cons:

- Cos-similarity can only take in query vectors of the **same length.**
- It is **not very intuitive**. Identical vectors (1,1) and (1,1) have cos-similarity of 1 while vectors (1,1) and (3,3) also have cos-similarity 1. (1,1) and (3,3) could represent completely different queries in our context.

Q Learning - Cosine Similarity

Possible solutions that we are investigating

- An alternative to cos-similarity that measures the similarity of two vectors.

- Encoding (vectorize) the queries in a way that the resulting vectors are of the same length.

Encoding

One-hot encoding

Pro: Vectors are always of the same length

Con: Will significantly increase dimensionality

Sentence encoder from Tensorflow

Pro: May encode to the same length while keeping dimensionality low (independent from the number of SQL query fragments)

Con: Some may not have decoders available

Training Data - SDSS Pros & Cons

Sloan Digital Sky Survey (SDSS)

Pros

- A lot of data. Almost 20 years of web/sql queries.
- One MSR paper discussing cleaning this data.
- Consistent with prior research.

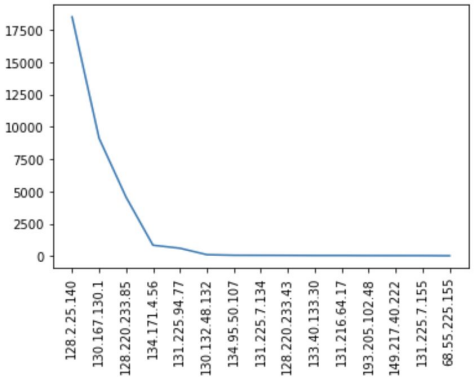
Cons

- A lot of bots.
- No session separation (but timestamp exists).
- The majority queries are not handwritten queries.

SDSS - Distribution of one week of data

	clientIP	count
15	128.2.25.140	18498
39	130.167.130.1	9122
18	128.220.233.85	4568
74	134.171.4.56	848
64	131.225.94.77	613
..	...	...
62	131.225.94.150	1
130	152.163.252.65	1
213	216.109.41.120	1
212	213.74.173.1	1
239	64.12.116.12	1

[299 rows x 2 columns]

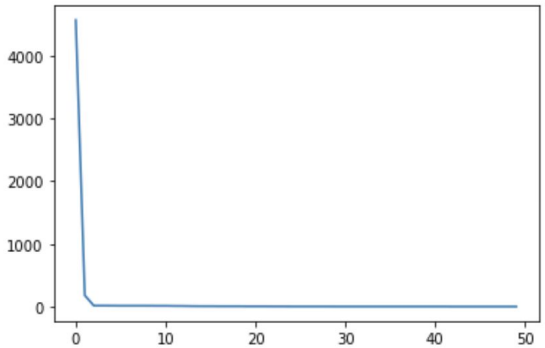


Client IP

	count
0	4568
1	184
2	20
3	20
4	19
...	...
29507	1
29508	1
29509	1
29510	1
29511	1

[29512 rows x 3 columns]

[<matplotlib.lines.Line2D at 0x7fdd8d78d100>]

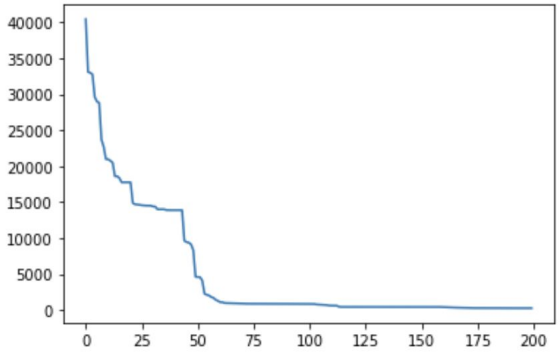


Query

	fragment	count
0	as	40419
1	n	33092
2	n.distance	32970
3	p	32752
4	p.ra	29611
...	...	...
36261	57.460815	1
36262	57.460796	1
36263	57.460560	1
36264	57.456852	1
36265	95.481767	1

[36266 rows x 2 columns]

[<matplotlib.lines.Line2D at 0x7fdd8fde55e0>]



Query Fragment

SDSS - Next step

Pull a lot of data.

Throw away a lot of data.

Training Data - Alternative Choice

SqlShare

https://uwescience.github.io/sqlshare/data_release.html

Pros:

- Handwritten queries.

Cons:

- No timestamp. (But there is a paper introducing a session separation method on this dataset)

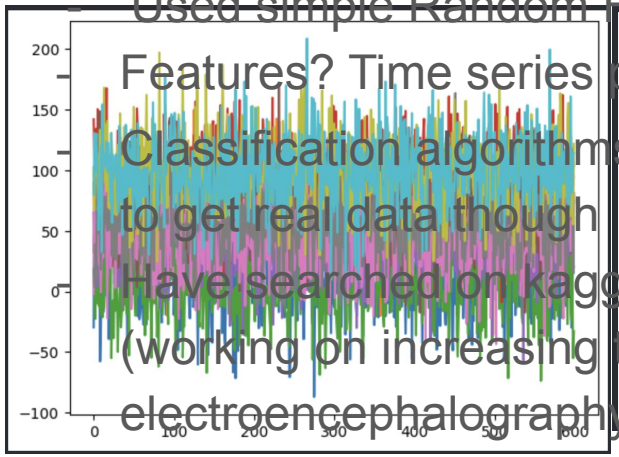
Project group 5

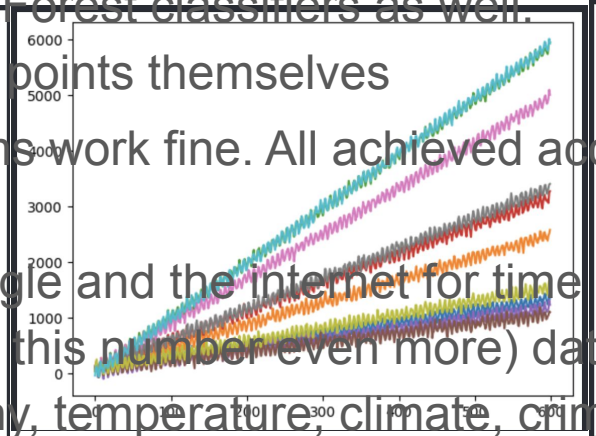
Today’s visualization systems require processing before its visualization to make it easily inferable. This is typically done by smoothing the raw data. Smoothing techniques, while minimizing noise, can also hide the structure of data (oversmoothing). No technique has yet been developed that is fully versatile and works well for every possible time-series by minimizing noise and preserving structure.

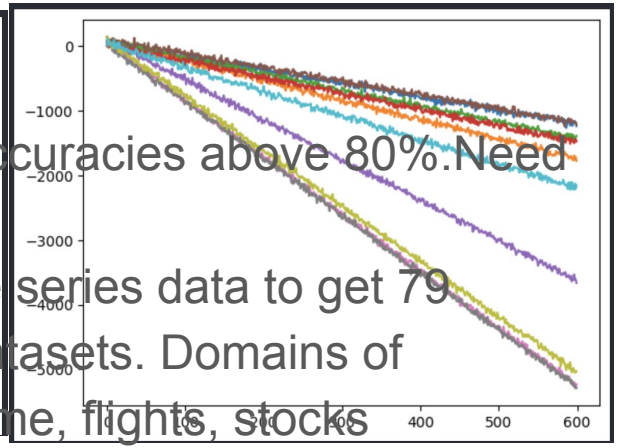
Therefore, the main question to ask here is: Can a generalized framework be developed for the effective smoothing of different categories of time series?

Current progress made

- We started with understanding time series classification. For a quick sanity check, generated time series data for categories we thought are reasonable
- Looked at time series classification algorithms (sktime library has common implementations)
- Used simple Random Forest classifiers as well.





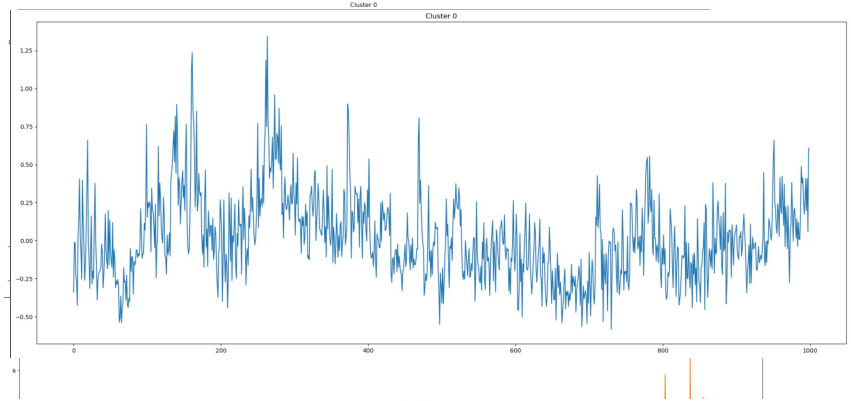


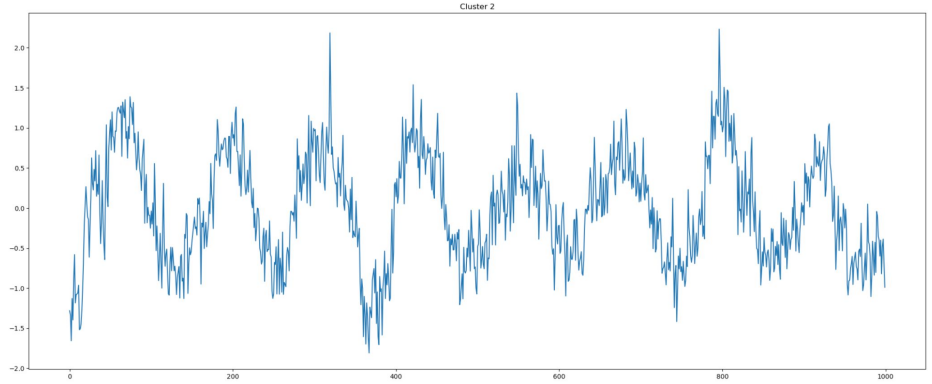
Features? Time series points themselves

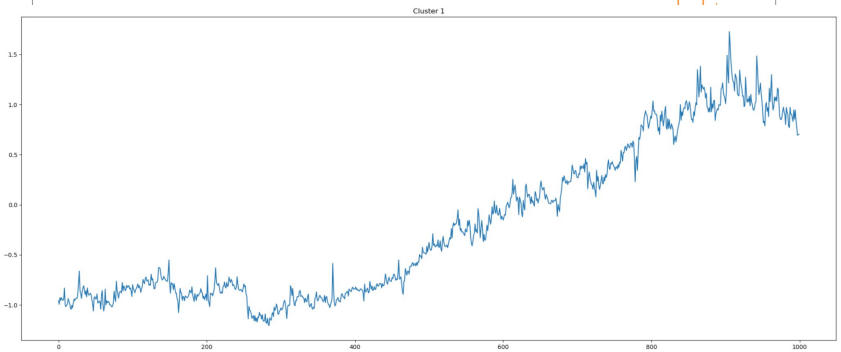
Classification algorithms work fine. All achieved accuracies above 80%.Need to get real data though

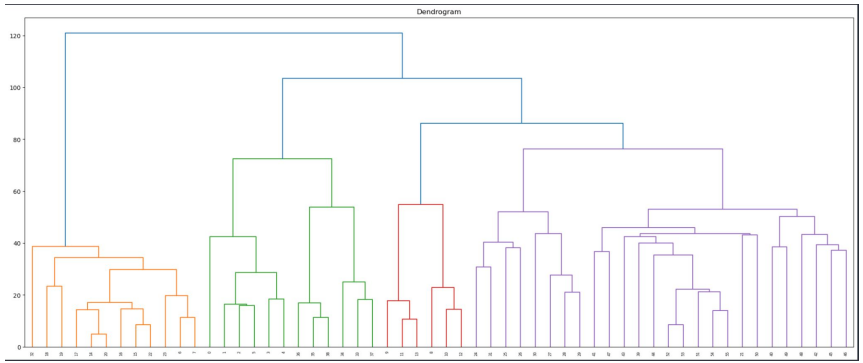
Have searched on kaggle and the internet for time series data to get 79 (working on increasing this number even more) datasets. Domains of electroencephalography, temperature, climate, crime, flights, stocks

What categories? Applying Hierarchical Agglomerative Clustering on datasets





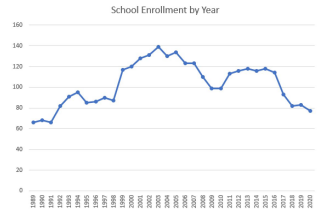




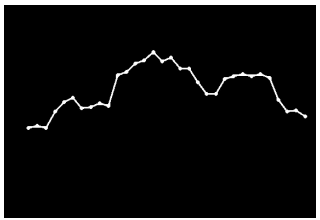
Next Steps

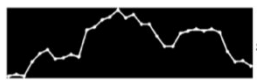
- ➔ Shortlist statistical measures and smoothing techniques which will be applied on the collected datasets.
- ➔ Create a user-study to understand which technique and measure works the best on a particular dataset.
- ➔ Analyse the results from the study and make inferences about how a category affects the choice of the smoothing technique.

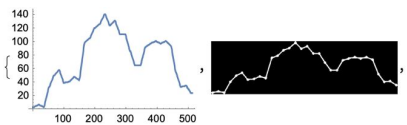
Project group 4 - **Line charts similarity search**




```
giveFinalCurve[imagePath_] := Module[{},
  graph = Import[imagePath];
  graph = RemoveAlphaChannel@ImageCompose[ImageResize[Image[{{1}}], ImageDimensions@graph], graph];

  hFilter = Closing[graph, BoxMatrix[{100, 0}]];
  vFilter = Closing[graph, BoxMatrix[{0, 100}]];
  grid = ImageApply[Min, {hFilter, vFilter}];

  cleanGraph = DeleteSmallComponents[Erosion[Dilation[Binarize[ImageDifference[graph, grid]], 2], 2]];

  largestComponent = Sort[ComponentMeasurements[cleanGraph, "OuterPerimeterCount"], #1[[2]] > #2[[2]] &][[1, 1]];
  finalCurve = ComponentMeasurements[cleanGraph, "Image"][[largestComponent]][[2]];

  finalCurve2 = finalCurve;
  While[True,
   If[
    Length[ComponentMeasurements[Erosion[finalCurve2, 1], "Area"]] == 1,
    finalCurve2 = Erosion[finalCurve2, 1];
    (*Print["erosion repeat"];*)
    ,
    (*Print["Breaking due to more than 1 component"];*)
    Break[];
   ]
  ];

  finalCurve3 =
   N /@ Function[{col}, Mean[First /@ Select[MapIndexed[{#2[[1]], #1} &, Reverse[col]], #[[2]] == 1 &]] /@
     Transpose[ImageData[finalCurve2]];

  {ListLinePlot[finalCurve3, PlotRange → All], finalCurve2, finalCurve3}
]
```

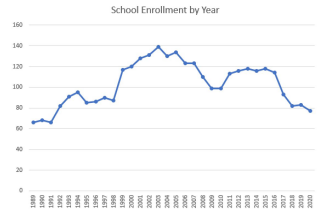




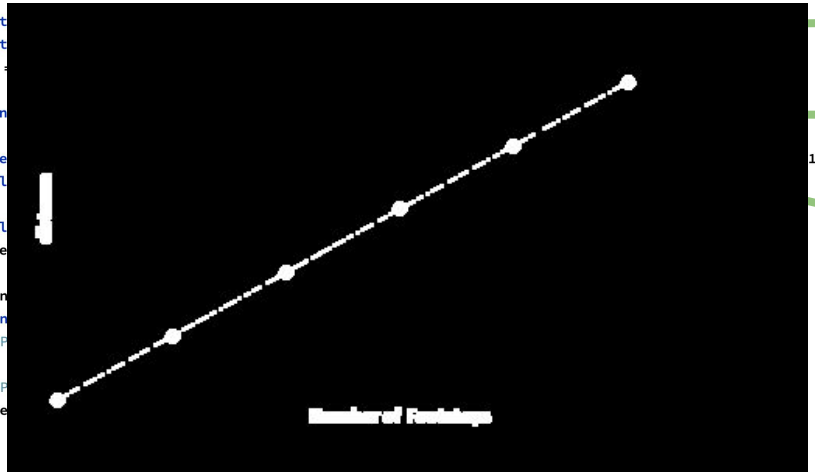


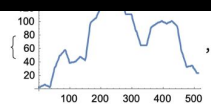

```
{4., 4., 4., 4., 4., 4.5, 5., 5.5, 5.5, 5.5, 6., 6., 6., 6.5, 6.5, 6.5, 7., 7.5, 8., 8., 8., 7.5, 7.,
 6.5, 6.5, 6.5, 6., 6., 6., 5.5, 5.5, 5.5, 5., 4.5, 4., 4., 4.5, 6., 7.5, 11., 12.5, 14.5, 16.5,
 18., 20., 21.5, 23.5, 25.5, 26.5, 30.5, 32., 33.5, 34., 34., 35., 36., 38., 39.5, 40.5, 41.5, 42.5,
 43.5, 44.5, 45.5, 47., 49., 50., 51., 51., 51., 51.5, 52., 53., 53.5, 54., 54.5, 55., 55.5, 56.,
 56.5, 57., 58., 58.5, 59., 59., 59., 58., 57., 54., 53., 52., 51., 49.5, 48., 47., 46., 45., 42.,
 41., 40., 40., 40., 40., 40.5, 40.5, 41., 41., 41., 41., 41., 41., 41., 41., 41.5, 41.5, 42., 42.,
 42., 42., 42.5, 43., 44., 44., 44.5, 45., 45.5, 46., 46.5, 47., 47., 48., 48.5, 49., 49., 49., 48.5,
 48., 47.5, 47.5, 47., 47., 46.5, 46., 46., 45.5, 45.5, 45., 44.5, 44., 45., 46.5, 48.5, 51., 57.,
```

Project group 4



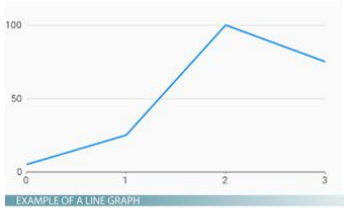

```
giveFinalCurve[imagePath_] := Module[{},
  graph = Import[imagePath];
  graph = RemoveAlphaChannel@ImageCompose[ImageResize[Image[{{1}}], ImageDimensions@graph], graph];

  hFilt
  vFilt
  grid =

  clean

  large
  final                                                                    1]];

  final
  While
   If[
    Len
    fin
    (*P
    ,
    (*P
    Bre
   ]
  ];

  finalCurve3 =
   N /@ Function[{col}, Mean[First /@ Select[MapIndexed[{#2[[1]], #1} &, Reverse[col]], #[[2]] == 1 &]]] /@
     Transpose[ImageData[finalCurve2]];

  {ListLinePlot[finalCurve3, PlotRange → All], finalCurve2, finalCurve3}
]
```

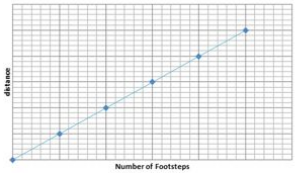


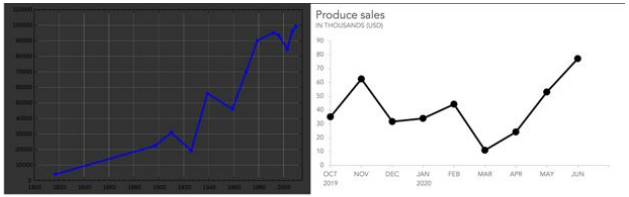


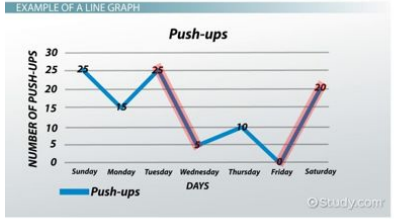
```
{4., 4., 4., 4., 4., 4.5, 5., 5.5, 5.5, 5.5, 6., 6., 6., 6.5, 6.5, 6.5, 7., 7.5, 8., 8., 8., 7.5, 7.,
 6.5, 6.5, 6.5, 6., 6., 6., 5.5, 5.5, 5.5, 5., 4.5, 4., 4., 4.5, 6., 7.5, 11., 12.5, 14.5, 16.5,
 18., 20., 21.5, 23.5, 25.5, 26.5, 30.5, 32., 33.5, 34., 34., 35., 36., 38., 39.5, 40.5, 41.5, 42.5,
 43.5, 44.5, 45.5, 47., 49., 50., 51., 51., 51., 51.5, 52., 53., 53.5, 54., 54.5, 55., 55.5, 56.,
 56.5, 57., 58., 58.5, 59., 59., 59., 58., 57., 54., 53., 52., 51., 49.5, 48., 47., 46., 45., 42.,
 41., 40., 40., 40., 40., 40.5, 40.5, 41., 41., 41., 41., 41., 41., 41., 41., 41.5, 41.5, 42., 42.,
 42., 42., 42.5, 43., 44., 44., 44.5, 45., 45.5, 46., 46.5, 47., 47., 48., 48.5, 49., 49., 49., 48.5,
 48., 47.5, 47.5, 47., 47., 46.5, 46., 46., 45.5, 45.5, 45., 44.5, 44., 45., 46.5, 48.5, 51., 57.,
```

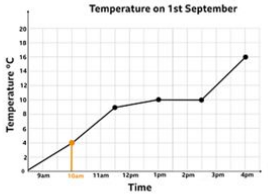
Project group 4

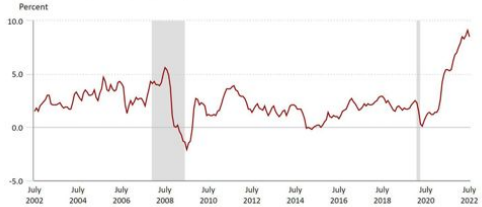


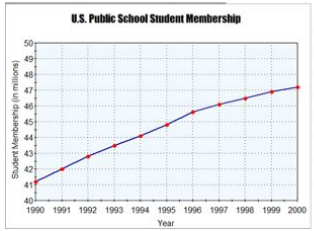


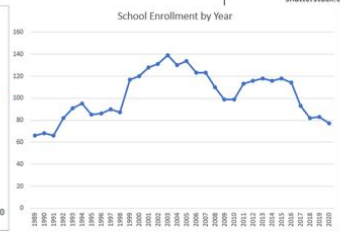


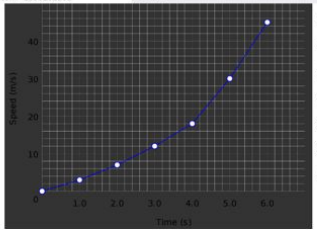


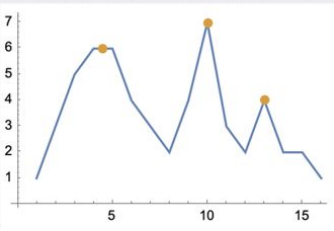








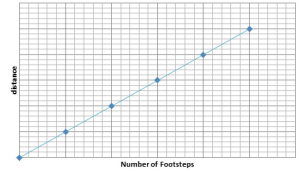


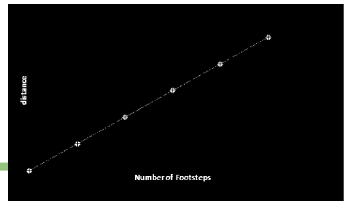


1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	0
11	1
12	1
13	1
15	1
17	1
20	1

Project group 4

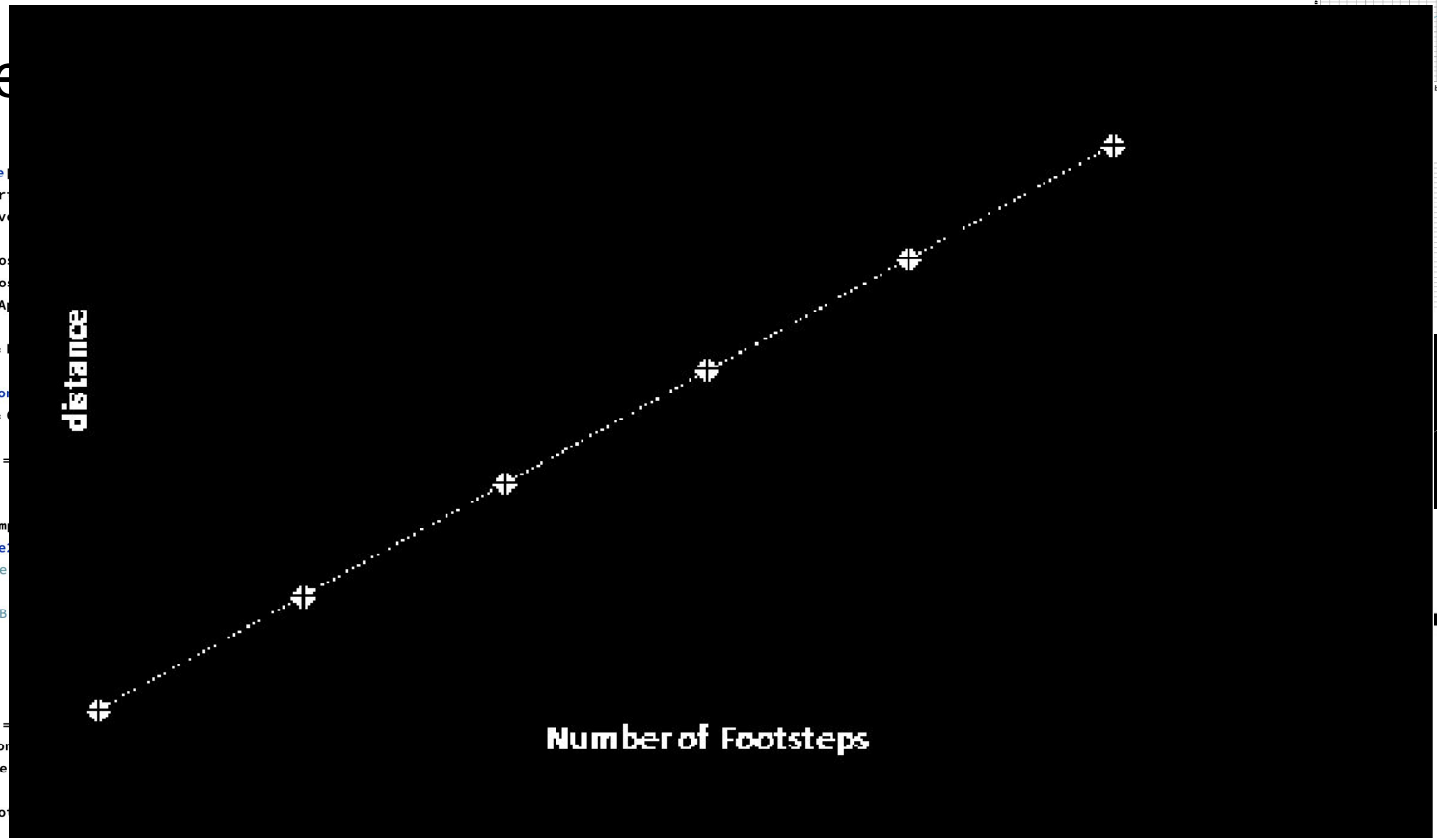
```
giveFinalCurve[imagePath_] := Module[{},
  graph = Import[imagePath];
  graph = RemoveAlphaChannel@ImageCompose[ImageResize[Image[{{1}}], ImageDimensions@graph], graph];

  hFilter = Closing[graph, BoxMatrix[{100, 0}]];
  vFilter = Closing[graph, BoxMatrix[{0, 100}]];
  grid = ImageApply[Min, {hFilter, vFilter}];

  cleanGraph = DeleteSmallComponents[Erosion[Dilation[Binarize[ImageDifference[graph, grid]], 2], 2]];

  largestComponent = Sort[ComponentMeasurements[cleanGraph, "OuterPerimeterCount"], #1[[2]] > #2[[2]] &][[1, 1]];
  finalCurve = ComponentMeasurements[cleanGraph, "Image"][[largestComponent]][[2]];

  finalCurve2 = finalCurve;
  While[True,
   If[
    Length[ComponentMeasurements[Erosion[finalCurve2, 1], "Area"]] == 1,
    finalCurve2 = Erosion[finalCurve2, 1];
    (*Print["erosion repeat"];*)
    ,
    (*Print["Breaking due to more than 1 component"];*)
    Break[];
   ]
  ];

  finalCurve3 =
   N /@ Function[{col}, Mean[First /@ Select[MapIndexed[{#2[[1]], #1} &, Reverse[col]], #[[2]] == 1 &]]] /@
     Transpose[ImageData[finalCurve2]];

  {ListLinePlot[finalCurve3, PlotRange → All], finalCurve2, finalCurve3}
 ]
```

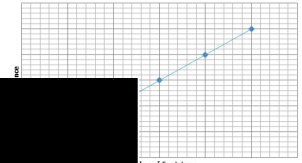


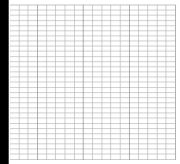


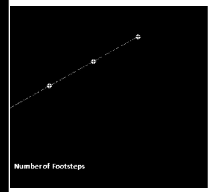
Proje

```
giveFinalCurve[
  graph = Impor
  graph = Remov

  hFilter = Clo
  vFilter = Clo
  grid = ImageA

  cleanGraph =

  largestCompo
  finalCurve =

  finalCurve2 =
  While[True,
   If[
    Length[Com
    finalCurve
    (*Print["e
    ,
    (*Print["B
    Break[];
   ]
  ];

  finalCurve3 =
   N /@ Functio
     Transpose

  {ListLinePlo
]
```



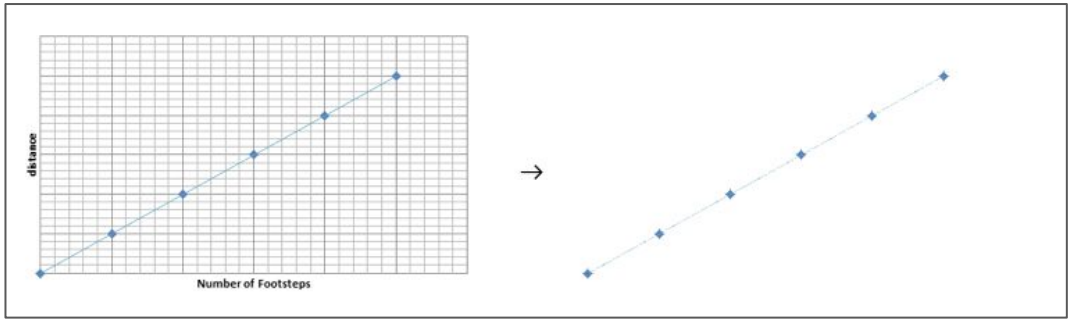


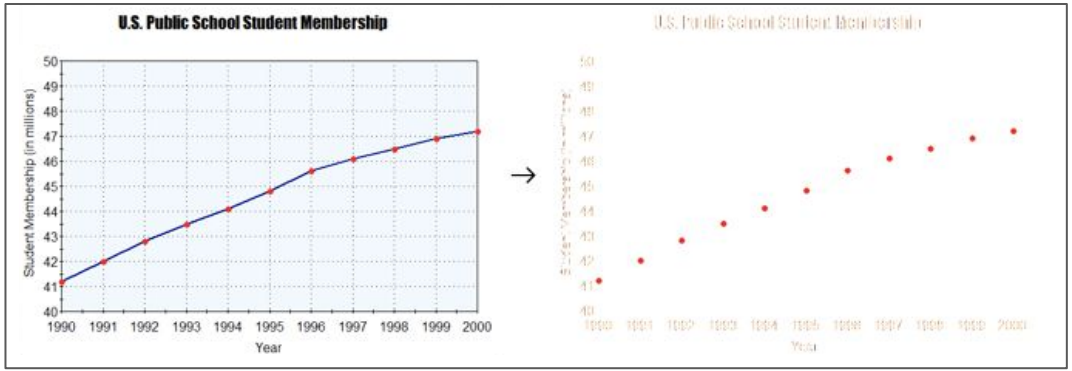




Project group 4

We need to look at colors

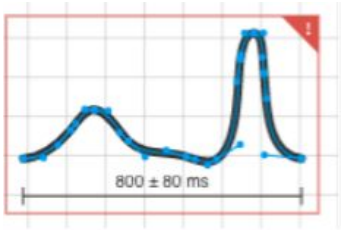


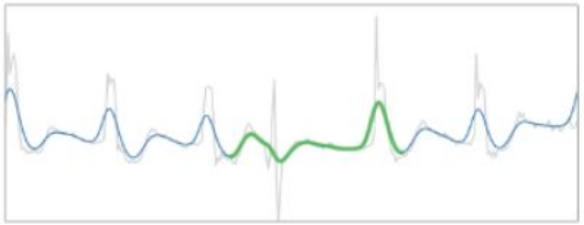


Project Group 3 - QetchPlus

Problem: Expressive time series queries using human sketches with complex pattern matching techniques.

Bit - Existing implementations utilize only distance metrics for their pattern matching methods such as Dynamic time warping, Euclidean distance and their own original distance metrics.

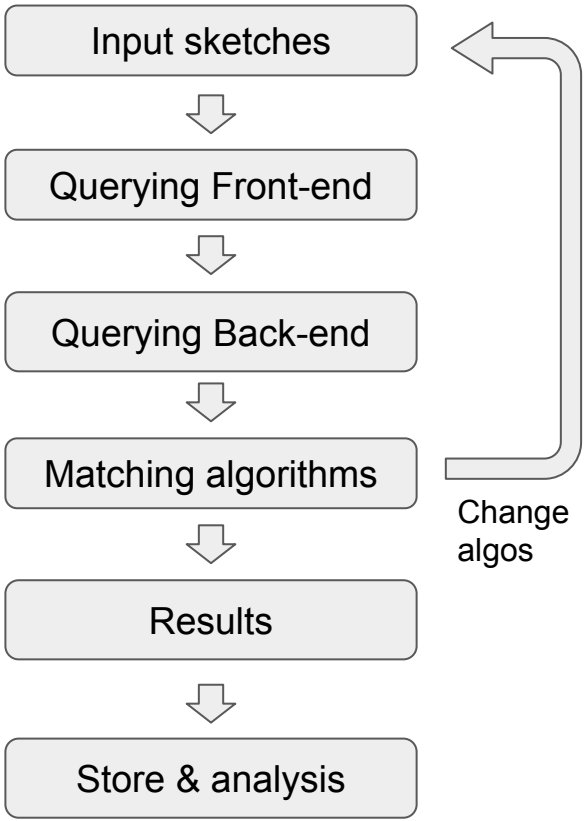




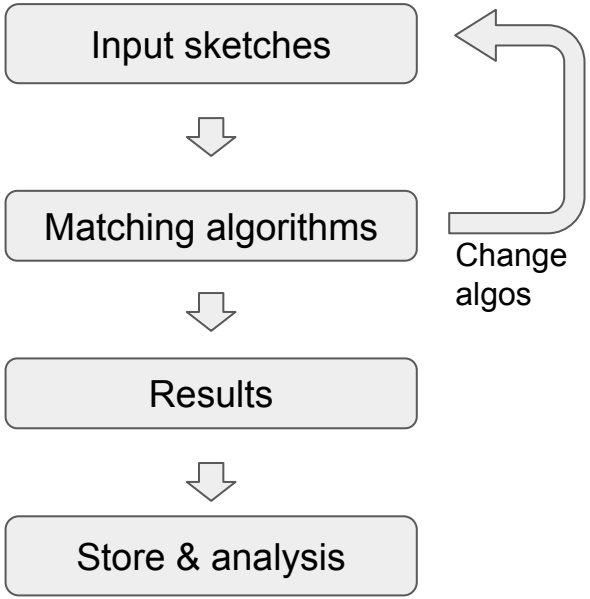
Flip - Improve performance in time series matching based on usage of more complex matching methods and compare with DTW, ED and qetch.

Challenge - Finding and testing algorithms that perform better than qetch

Vectoring - Quick Evaluation



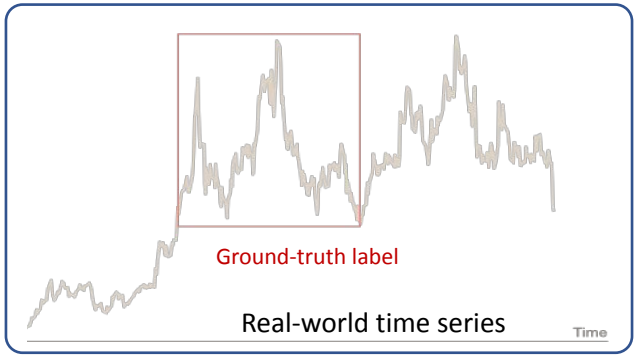
OR

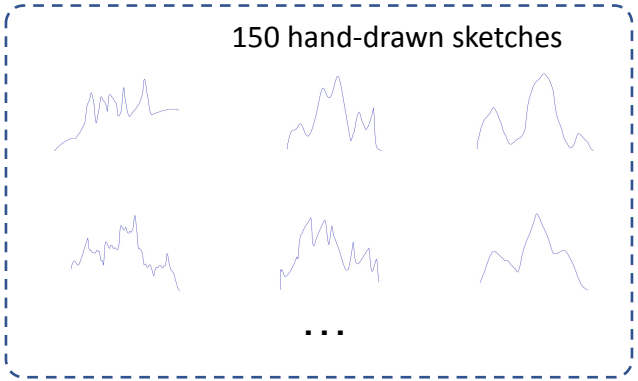


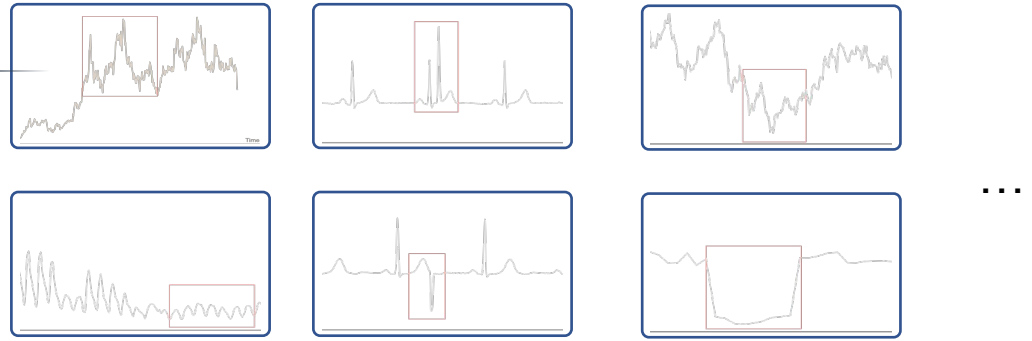
- Get rid of onerous implementations of front/backend
- Quick evaluation and analysis
- Only need to incorporate the algo with the best performance

What we have done since our last proposal

Crowd-study Dataset







Crowd-study dataset

- 8 real-world time series from various domains
- 150 labeled sketches for each time series

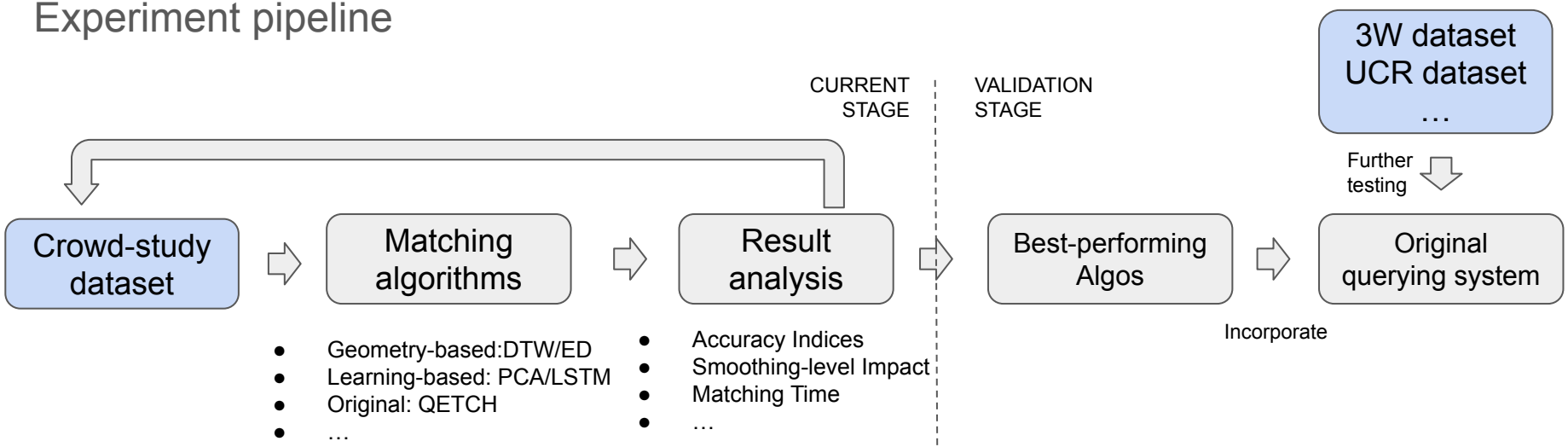
- Contains some duplicates, corrupted sketches
- Stored in png
  → need data preprocessing & cleaning

What we have done since our last proposal

Data Preprocessing and Data Cleaning

- extracted corresponding one-dimensional data and ground-truth labels from the image files
- remove the corrupted and deformed data samples
- detected and eliminated the duplicate sketch samples

Experiment pipeline



Next Steps -

Short term goal:

1. Reimplement qetch algorithm outside their framework.
2. Implementation and testing of algorithms on the data pipeline using the processed dataset (Plug in and test using the concept of vectoring).

Long term goal:

1. Re-integration of the identified “best” algorithms with existing Qetch application.

Project group 2: QuickScatter

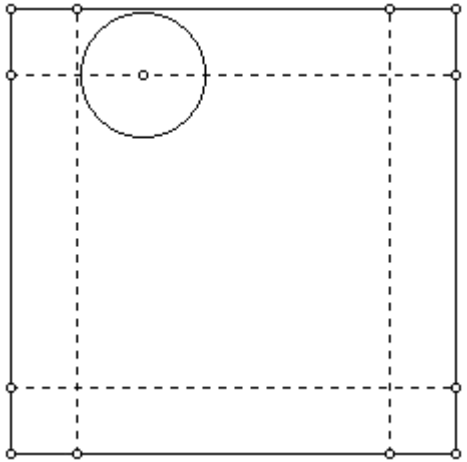
Scatter plots or an invaluable tool for data analysis since they can capture correlations between ostensibly randoms variables and visualize clusters within arbitrary data sets. For large datasets, however, they are prone to overplotting. To accommodate large datasets, scatter plots should adopt statistical sampling strategies based on the dimensions of the scatter plot to reduce latency and avoid overloading the user with redundant information.

Central Research Question: Can scatter plot be adapted to detect relations and/or represent large datasets for interactive environments?

Project group 2: QuickScatter Initial Exploration

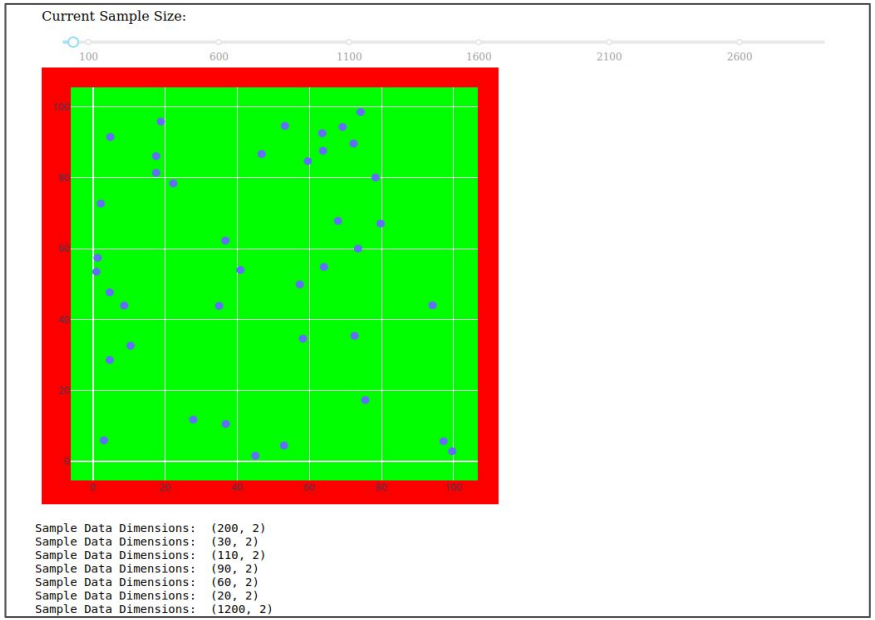
- Developed a sampling upper bound dependent on the dimensions of the scatter plot.
- Explicit derivation based on Buffon's Coin Problem.
  - L = Length Of Square Canvas
  - R = Radius of Marker
  - B = User Defined Probability Threshold
- Assumes all previous (n-1) coins did not overlap. Joint probability leads to low samples and sparse scatter plots.

$$\frac{(L-2r)^2-\beta L^2}{\pi r^2}=n$$

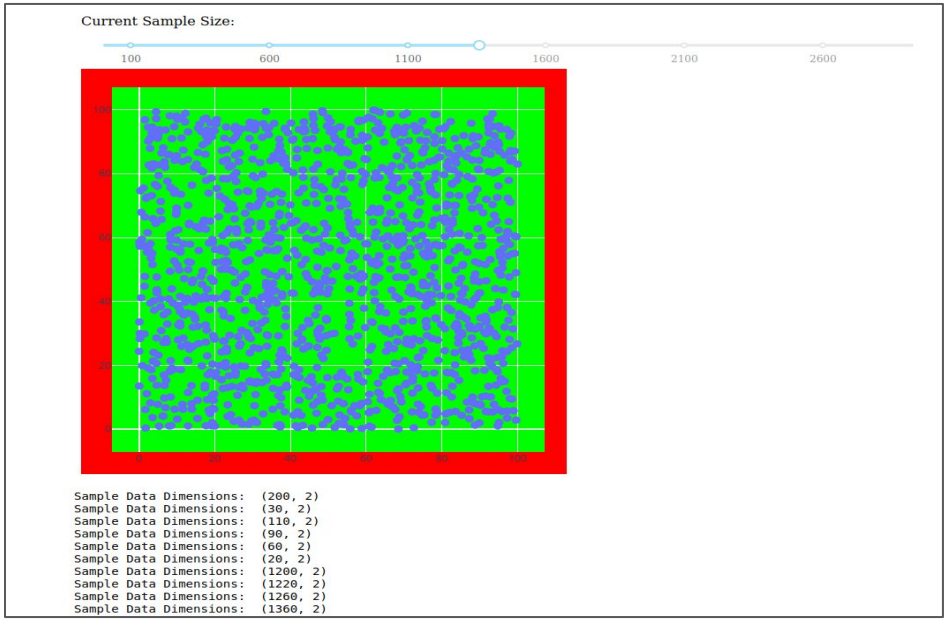


Project Update 2: Empirical Evaluation Of Upper Bound

Joint Probability: ~30 Points



Previous n-1 coins don't intersect: ~1400 Points



Project group 2: QuickScatter

- Explored approach: seam-carving
- DP algorithm from CV used to resize images by removing unchanging horizontal/vertical paths in image matrix.
- Pros: seam-carving was able to maintain correlation and remove clutter in dense areas
- Cons: Runtime varies significantly with input image => not feasible for high-res images

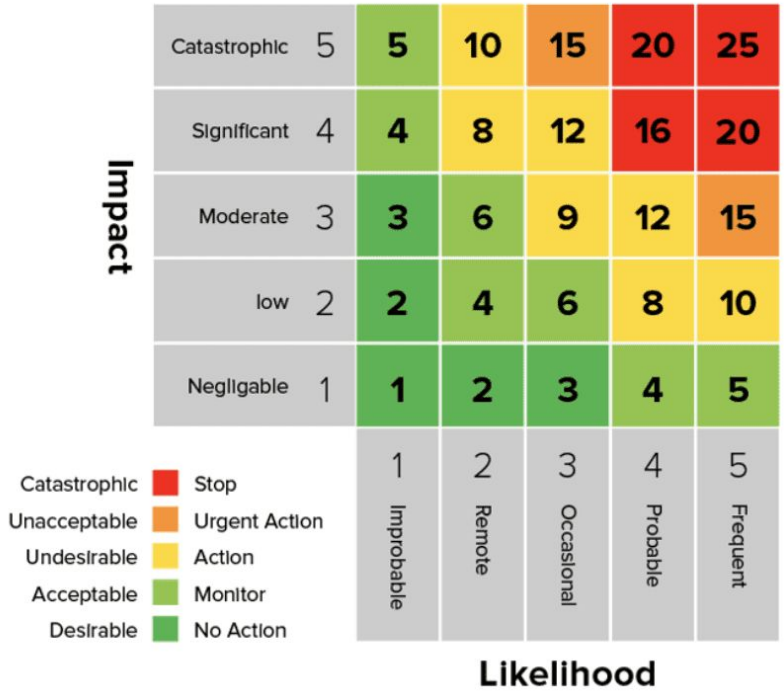
Project group 2: QuickScatter

- Next goal: Read more sampling literature, and understand what is a reasonable sample size for creating heat maps
- Intermediate steps:

  1.) Simulate a multi-Gaussian population to run sampling experiments on

  2.) Determine a good upper-bound for sampling on a real-world MLB dataset

  3.) Determine the edge-cases where over-sampling is likely to occur



Group 1 : Exploratory Video Analytics

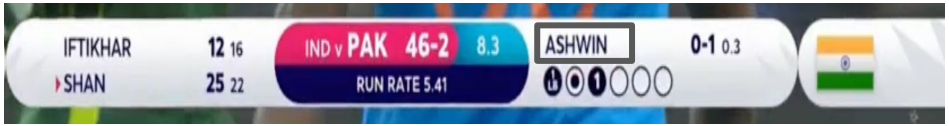
Support Exploratory Video Analytics using domain hints

- Eg: Find where player (Ashwin) is bowling in a cricket game
- Approach 1: Run face detection
- Approach 2: Look for Ashwin's jersey number

Group 1 : Exploratory Video Analytics

Support Exploratory Video Analytics using domain hints

- Eg: Find where player (Ashwin) is bowling in a cricket game
- Approach 1: Run face detection
- Approach 2: Look for Ashwin's jersey number
- **Alternate Approach : Use scoreboard and apply OCR – domain hint**



Project Updates

- Dataset Finalization
  - Open sourced cricket dataset : Fine-Grain Annotation of Cricket Videos
  - Youtube Clips and Screen Recording of Cricket Games
- Enhancing label-studio-frontend to meet our application requirements.

System Overview

Talk is cheap. Show me demo!