MONTE-CARLO SIMULATION

CALCULATION OF VAR (VALUE-AT-RISK) & CVAR (CONDITIONAL VALUE-AT-RISK)

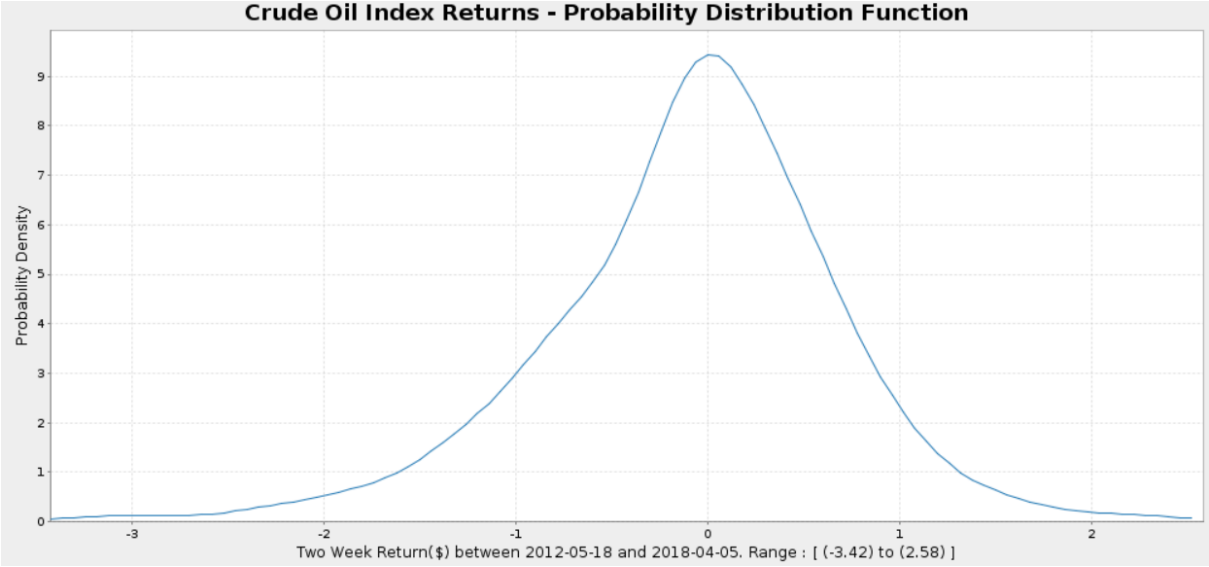
PRESENTER: SANJOY ROY

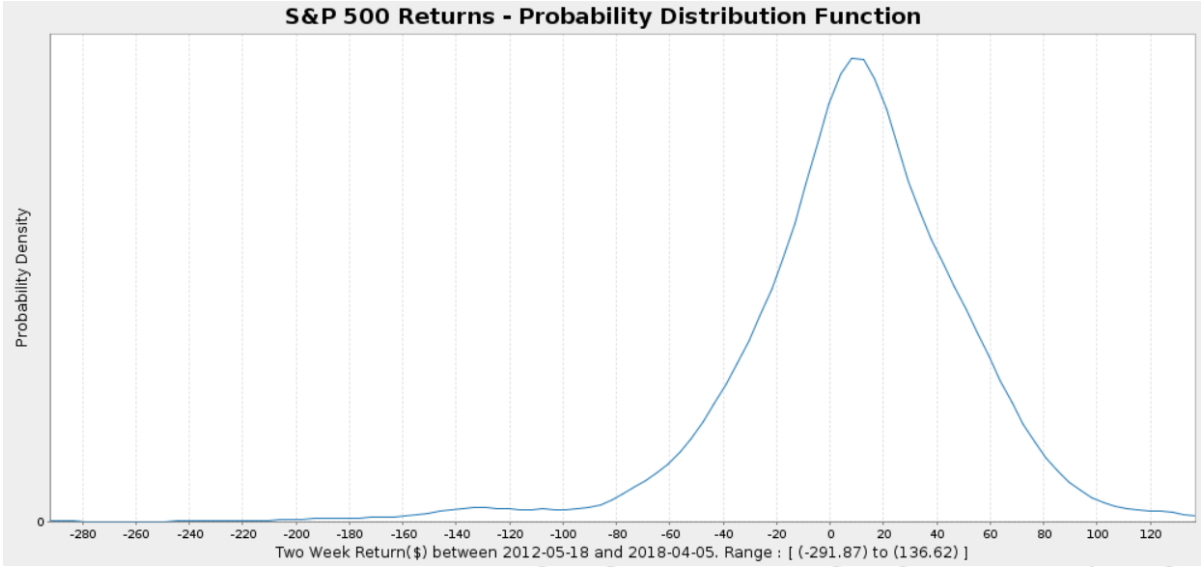
15-APR-2018

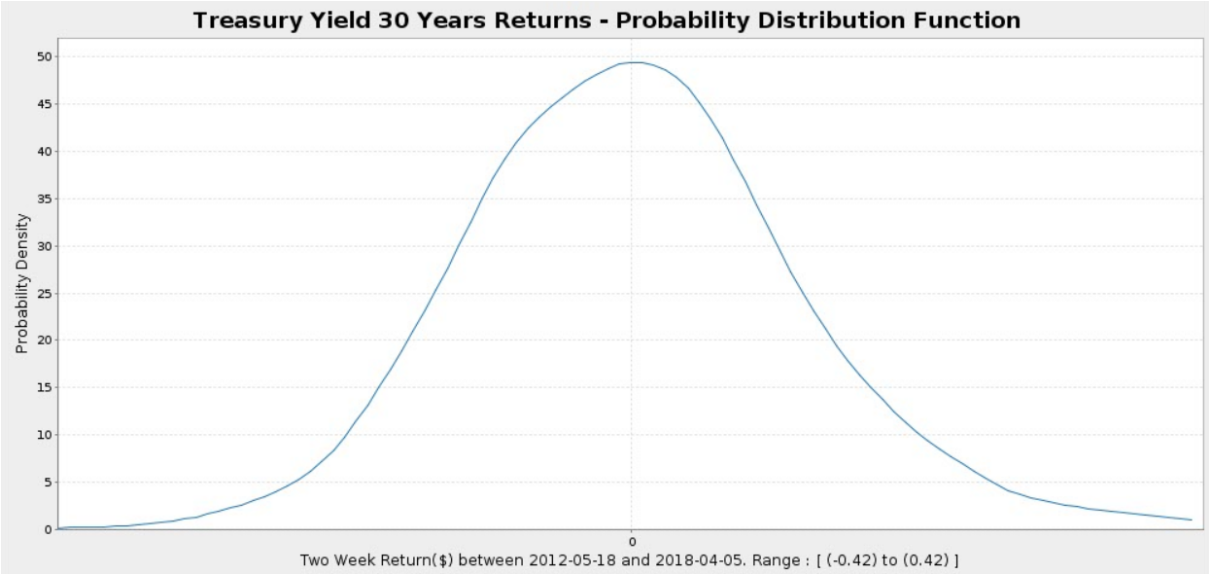
TERMINOLOGY

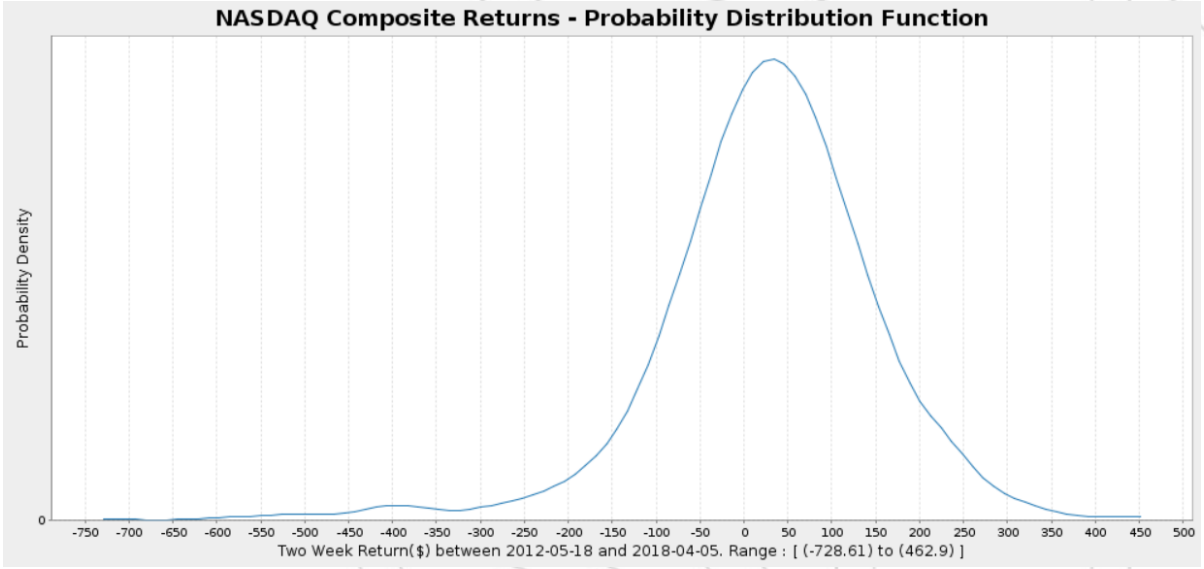
- V-a-R (Value-At-Risk) – How much can one expect to lose
- Parameters defining VaR – Portfolio, Time Period and p-value.
  - A VaR of 1 million dollars with a 5% p-value and two weeks implies that the portfolio stands only a 5% chance of losing more than 1 million dollars over two weeks
- C-VaR (conditional-V-a-R) aka Expected Shortfall:
  - same three parameters as a VaR statistic, but considers the expected loss instead of the cut-off value.
  - A CVaR of 5 million dollars with a 5% p-value and two weeks indicates the belief that the average loss in the worst 5% of outcomes is 5 million dollars.
- A portfolio is a bunch of instruments which we are trying interested to gauge the V-a-R or expected loss.
- Market factors are the key-indices which are considered as imaginary super-set of instruments traded in a particular market/bourse – eg, S&P500, US Treasury,
- Return – change in an instrument or portfolio's value over a time period.

RESULTS – PDF OF INDEXES USED

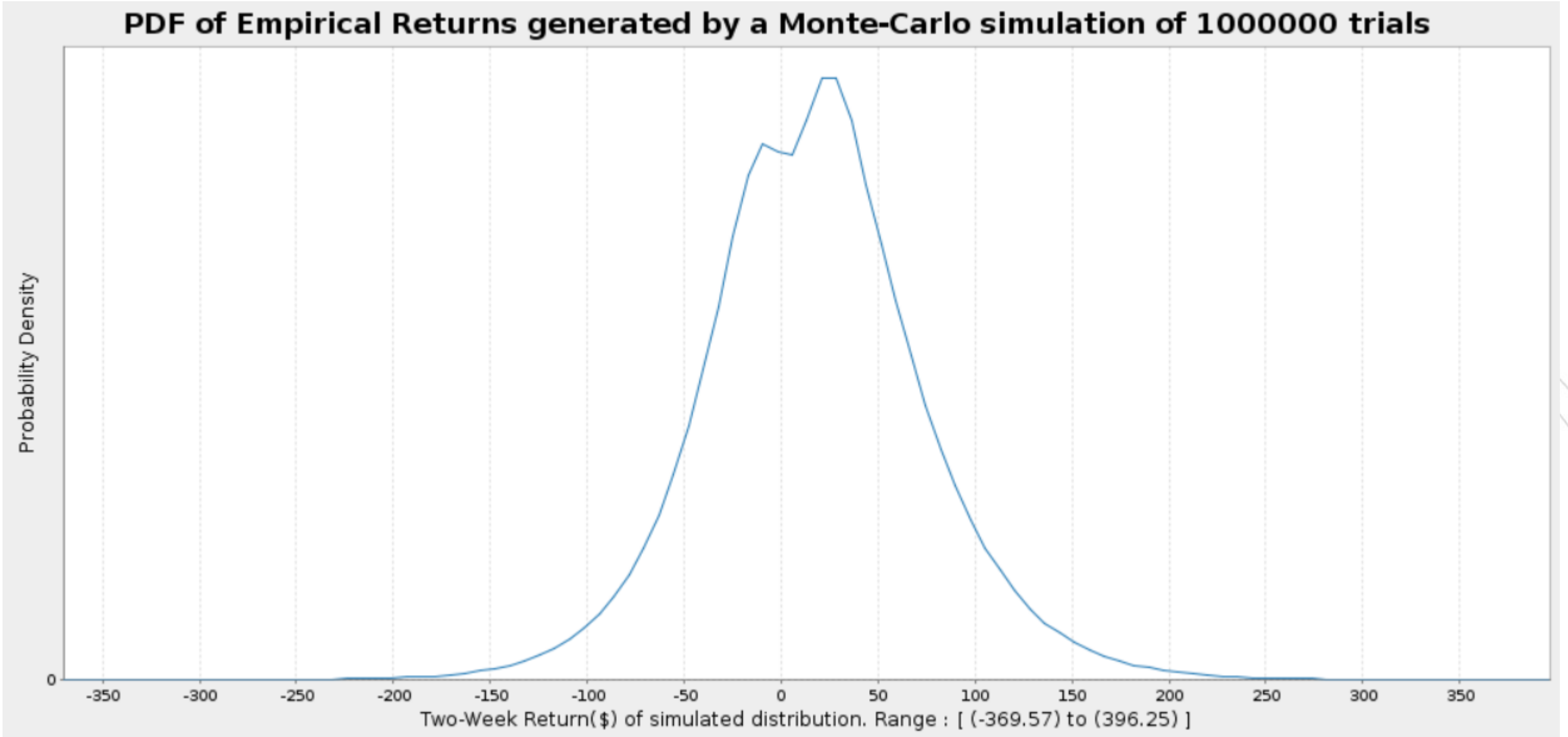








RESULTS – THE SIMULATED RETURN OF PORTFOLIO

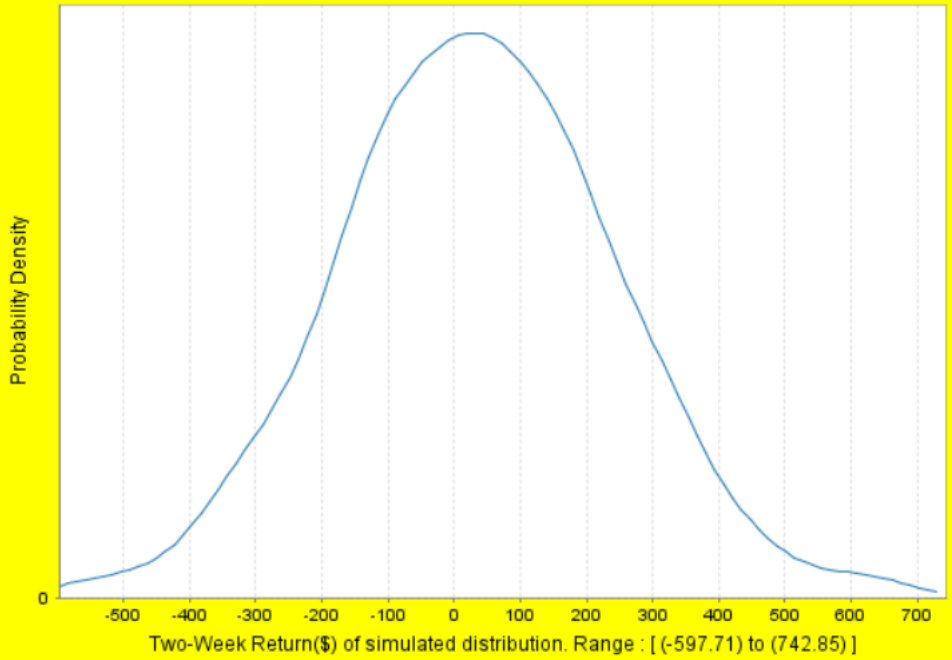


SUMMARY OF RESULTS

Monte Carlo Simulation

http://localhost:8000/results.html

PDF of Empirical Returns generated by a Monte-Carlo simulation of 1000 trials




```
################ Execution Stats as shown below: ################
Monte-Carlo Execution Completed for the entire portfolio of stocks which consists of below:
hdfs://localhost:8020/user/vmuser/mcdata/stocks/AAPL.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/AMZN.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/CSCO.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/FB.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/GOOGL.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/IBM.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/MSFT.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/NFLX.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/ORCL.csv
hdfs://localhost:8020/user/vmuser/mcdata/stocks/TCEHY.csv
```

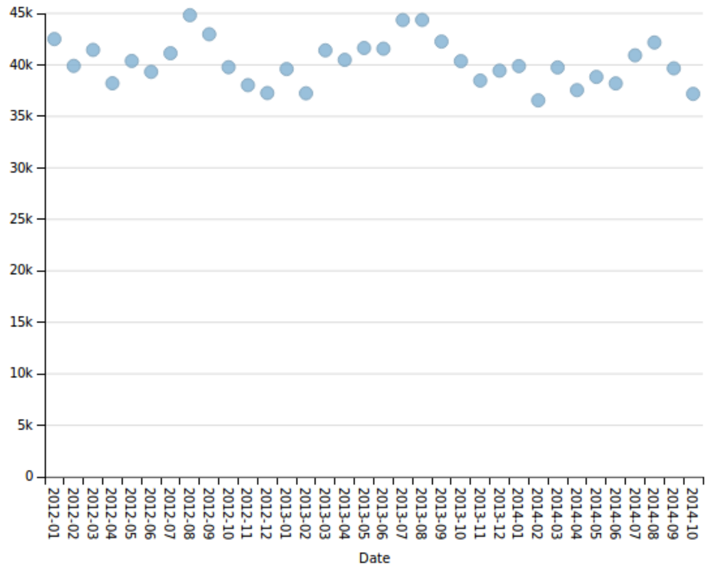
EXTENDED RESULTS AS A WEBLOG

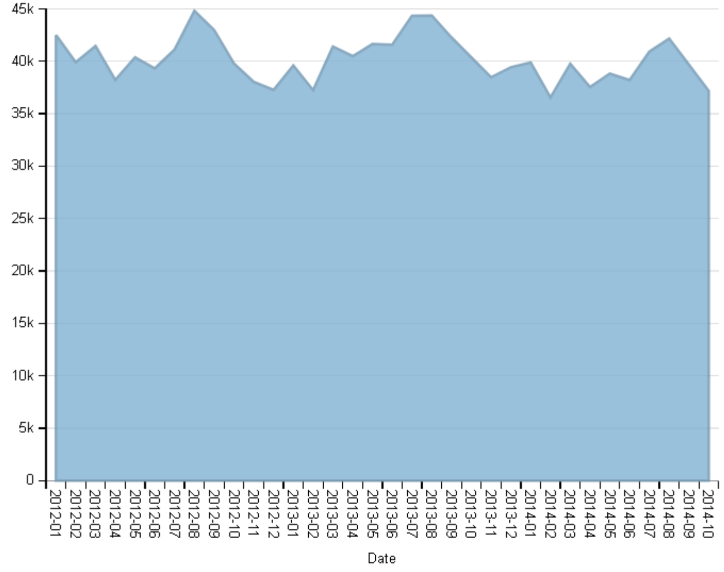
localhost:8000/results.txt

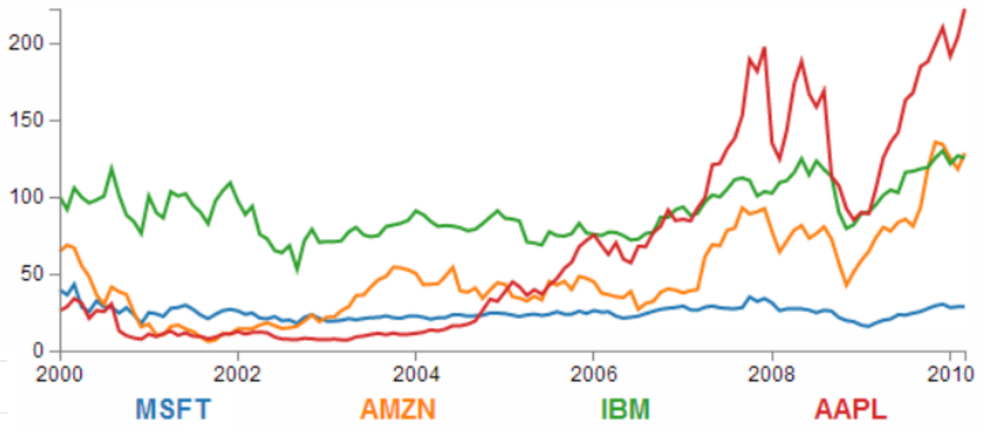
localhost:8000/results.txt

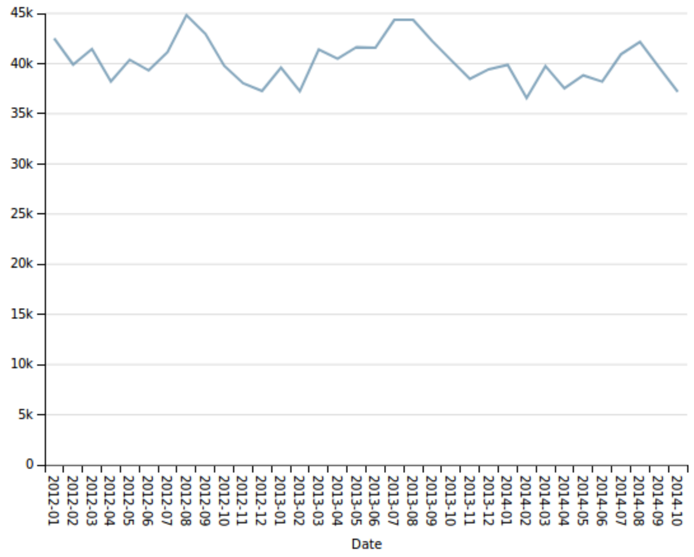
```
################ Execution Stats as shown below: ################
Monte-Carlo Execution Completed for the entire portfolio of stocks which consists of below:
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/AAPL.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/AMZN.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/CSCO.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/FB.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/GOOGL.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/IBM.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/MSFT.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/NFLX.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/ORCL.csv
 hdfs://localhost:8020/user/vmuser/mcdata/stocks/TCEHY.csv
Following files in the factors dir:
 hdfs://localhost:8020/user/vmuser/mcdata/factors/GSPC.csv
 hdfs://localhost:8020/user/vmuser/mcdata/factors/IXIC.csv
 hdfs://localhost:8020/user/vmuser/mcdata/factors/OIL.csv
 hdfs://localhost:8020/user/vmuser/mcdata/factors/TYX.csv
-----------------------------------------------------------------------
Started Execution at : 2018/05/02 18:14:35
Ended Execution at : 2018/05/02 18:23:43
Total Execution Time (ms): 548059
Time Horizon: 2012-05-18 to 2018-04-05
Number of trials : 1000000
Number of parallel threads executed: 100
No of instruments in the portfolio: 10
No of indices used : 4
Base Data Size (records) of all instruments combined : 42379
Base Data Size (records) of all stocks combined : 63081
Trimmed Data Size (records) of two-week returns of entire portfolio of stocks and indexes : 80850
Size of Empirical Distribution (trials) : 1000000
-----------------------------------------------------------------------
THE RESULTS:
VaR for entire portfolio: 73.49967946168957
CVaR (expected shortfall) for entire portfolio: 103.57125055224739
confidenceLevel : 0.8
Computed bootstrapped confidence interval of VaR for entire portfolio at 5%: (-73.85345941681665,-73.24173410738808)
Computed bootstrapped confidence interval of CVaR for entire portfolio at 5%: (-103.96452877440151,-103.1128666323168)
Accuracy of prediction using Backtesting on historical data for a sample instrument in the portfolio : 93.74149
-----------------------------------------------------------------------
BREAKDOWN of TIME CONSUMED BY STAGES:
Breakdown of total Execution Time (ms): 548059
STAGE 01: Total Data Preparation Time (ms): 9017
STAGE 02: Total Polynomial Regression model fitting Execution Time (ms): 46
STAGE 03: Total Monte Carlo Simulation Execution Time (ms): 11779
STAGE 04: Total Validation Execution Time (ms): 520903
-----------------------------------------------------------------------
BREAKDOWN of TIME CONSUMED BY STAGE 04:
Breakdown of Evaluation time (ms): 520903
Total time taken to compute bootstrapped confidence intervals (ms): 520886
Total time taken Backtesting on historical data (ms): 14
```

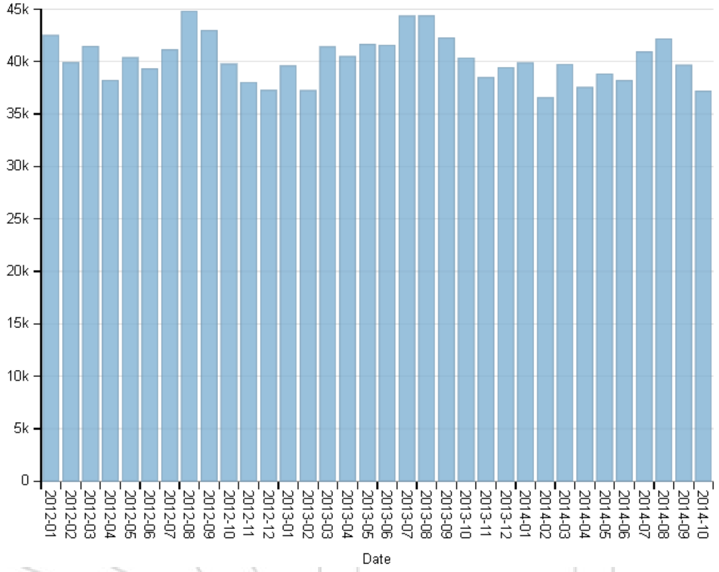
D3 VISUALISATIONS – ILLUSTRATING TRENDS



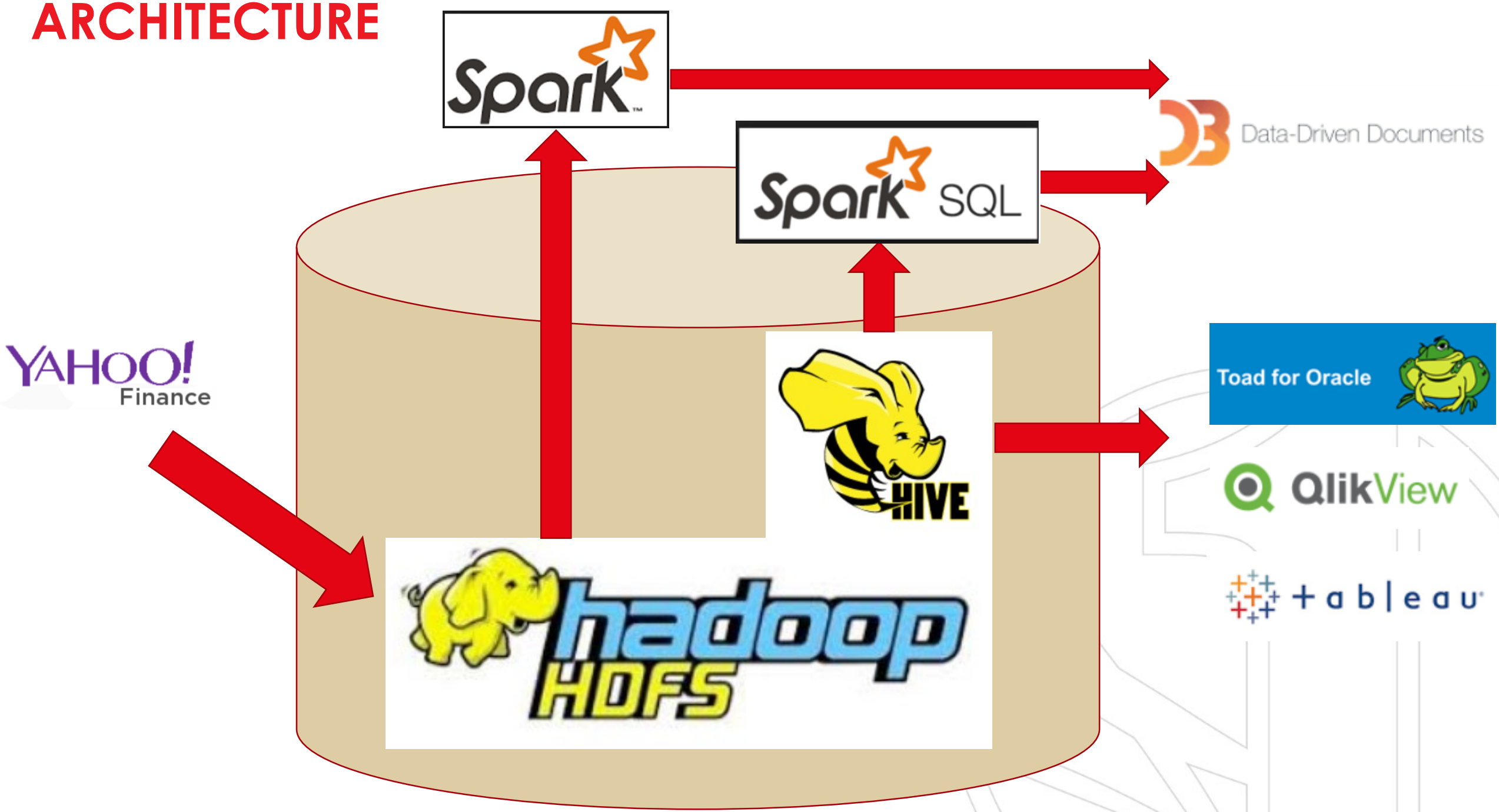








ARCHITECTURE



ASSUMPTIONS

- All financial data for instruments and market factors are extracted from YAHOO! Finance.
- The window of return is stipulated to 2-weeks – for both instruments and factors, and the p-value is set to 5%.
- We are interested in the daily closing value of an instrument/index (factor)
- We will be using OLS (Ordinary Least Squares) Regression algorithm to model the relationship between market conditions (factors) and each instrument's returns.
- We will employ Multi-Variate Normal distribution to simulate the trial data.
- We will be using breeze library for plotting the probability density functions
- For a given distribution, we will choose a sample from the distribution, derive the standard deviation of the sample, and use Gaussian kernel on the sample (with the std-dev) to derive the probability density of the sample.
- The trials will be split into specific number of threads (which is parameterized) to be executed in parallel.
- For every trial, we will deploy Mersenne-Twister algorithm to generate a random number which will eventually help to build the simulated data for values of an instrument in each trial.

MONTE-CARLO SIMULATIONS OVERVIEW

- Step 1: We will model the relationship between market conditions (factors) and each instrument's returns.
  - Using vector algebra, we will generalize the total return of an entire portfolio.
  - Employ OLS (Ordinary Least Squares) Regression algorithm – with polynomial features.
  - Output of the model is the coefficients/weightage vector
- Step 2: Choosing a distribution function (MVND) with the parameters of the weightage vector - covariance (cov) and means (m) and a random number, or, MVND = f(random number, cov,m)
- Step 3: Generate the trial data based on for parameterized number of trials, and in how many parallel threads we want to generate the workload.
  - For every iteration in the trial the MVND = f(random number, cov,m) function provides a simulated distribution of instrument prices.
  - Each vector of simulated distribution of instrument return is applied the weightage vector to provide the instrument return (as mandated by the model in Step 1).
  - All such instrument returns constitutes the simulated vector of returns of a portfolio.

MONTE-CARLO SIMULATIONS OVERVIEW

- Step 04: Calculate the Risk Measures from the simulated returns
  - V-a-R (at p-value) – which essentially means pick the return, from the simulated return vector generated by the MC Trial, which is in the bottom p% of the distribution.
  - C-V-a-R (at p-value) – calculate the average return of the bottom p% of the distribution

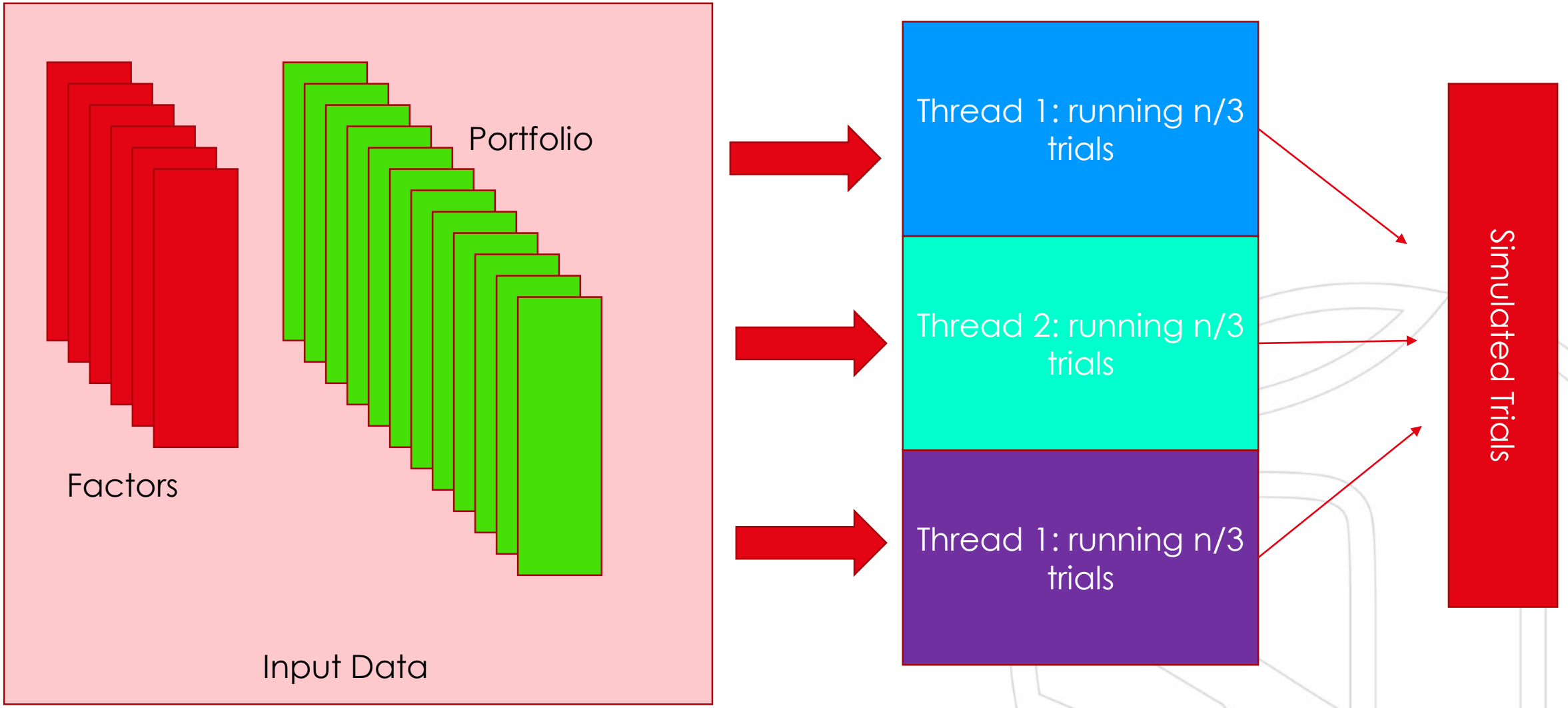
- Step 05: Evaluating the results
  - Boot-Strapping the confidence interval
  - Back-testing on historical data

DATA FLOW – HOW WE ARE DOING IT

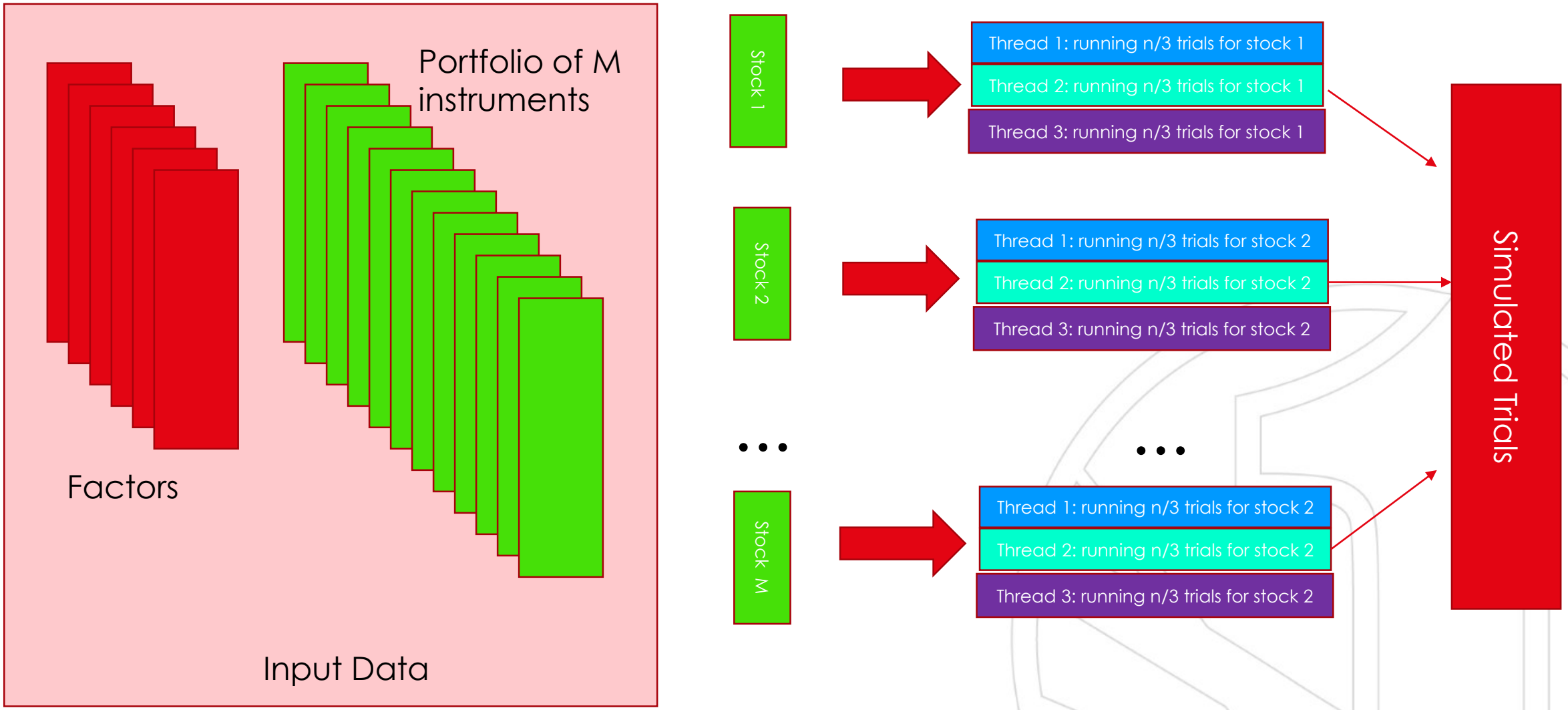
- Provisions made to run the monte-carlo simulation to calculate the V-a-R and C-V-a-R for
  - a particular stock (which can be parameterized) or
  - an entire set of portfolio.
- Extract publicly traded instrument and indices from YAHOO! finance
  - Stage the instruments in an HDFS directory – which indicates a portfolio.
  - Stage the factors in another HDFS directory – which indicates the factor.
- Filter the portfolio of instruments and the indices on the same time-window of 2-weeks– to generate the instrument-return and factor matrix.
- Model the relationship between market conditions (factors) and each instrument's returns to derive the weightage vector.
- Hitherto everything is being setup using normal scala collection in a single thread – to ensure the integrity of the data specially applicable for the linear algebra operations underneath the OLS Regression algorithm.
- Apply Apache Spark's distributed framework to split up the trials in threads and execute the threads in parallel: (partitioning-by-trial).
  - In each thread - Use the parameters of the weightage vector (covariance and means) to generate simulated return of vectors for each instrument.
- Additional parallelism can be achieved by executing the trials for individual stocks in parallel.

PROCESSING PARALLEL WORKLOADS – METHOD 1

Executing n trials with 3 threads



PROCESSING PARALLEL WORKLOADS – METHOD 2



WHERE TO GO FROM HERE

- Parameterizing the OLS Regression with further regularization parameters
- Option to choose a different regression model like Decision Tree.
- Enable SQL on Big Data – Make Spark SQL Framework function via HiveContext and process data from HDFS via Hive Metastore
- Extend the algorithm so that it can be leveraged for online scoring as well.
- Read/Write the data in HDFS in binary(efficient) file formats, and use AVRO serialization/de-serialization techniques to read/write the data
  - Store data in ORC format to query in HIVE
  - Store data in PARQUET format to query from HBASE
- Data Visualizations – Intuitive Dashboards on streaming data


Q
A