tickerplant.ai

Single Console Tickerplant : q/Kdb+/pyQ/python Stack

Rohan Shiloh Shah
rohanshilohshah@outlook.com

June 2, 2018

www.cs.McGill.ca/~rshah3

Table of contents

1. Temporal Big Data

2. Architecture

3. Installation

4. Conclusion

Outline of Presentation

'Dispel any notion of tick being a black box product which cannot be modified according(ly)' -N.Perrem

Sections

- What is tick data?

Outline of Presentation

‘Dispel any notion of tick being a black box product which cannot be modified according(ly)’ -N.Perrem

Sections

- What is tick data?
- Why do traditional RDBMS fail spectacularly when trying to capture this data in real-time for subsequent analysis?

Outline of Presentation

‘Dispel any notion of tick being a black box product which cannot be modified according(ly)’ -N.Perrem

Sections

- What is tick data?
- Why do traditional RDBMS fail spectacularly when trying to capture this data in real-time for subsequent analysis?
- Overview of q/Kdb+ tickerplant architecture

Outline of Presentation

‘Dispel any notion of tick being a black box product which cannot be modified according(ly)’ -N.Perrem

Sections

- What is tick data?
- Why do traditional RDBMS fail spectacularly when trying to capture this data in real-time for subsequent analysis?
- Overview of q/Kdb+ tickerplant architecture
- Machine Learning and Order Book Analysis

Temporal Big Data

Order Book

The real-time fx/equity/derivative order book is generally available at one of 4 levels.

- Level 1 : best bid/ask prices with size and traded volume
- Level 2 : market depth 5
- Level 3 : market depth 20
- Tick by Tick : yeverything

Bombay & National Stock Exchanges

The NSE provides feeds for all 4 levels whereas the BSE only provides 1 minute snapshots of the first 2 levels? Derivatives data from the BSE is currently free.

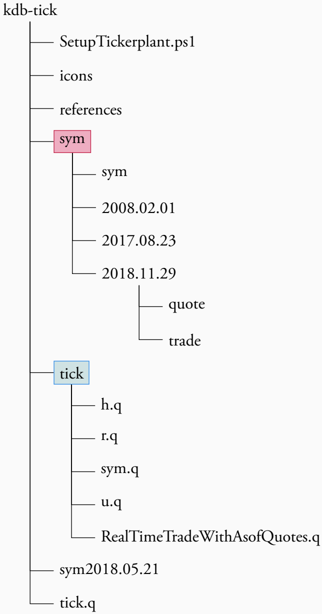
Pricing

`www.nseindia.com/supra_global/content/dotex/data_products.htm`

`www.bseindia.com/products_and_services/productinformation.aspx`

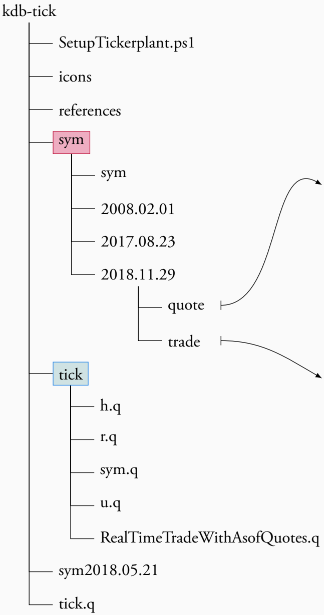
Order Book Size on Disk

Files : Date Partitioned Column Database & q Code



- kdb-tick
  - SetupTickerplant.ps1
  - icons
  - references
  - sym
    - sym
    - 2008.02.01
    - 2017.08.23
    - 2018.11.29
      - quote
      - trade
  - tick
    - h.q
    - r.q
    - sym.q
    - u.q
    - RealTimeTradeWithAsofQuotes.q
  - sym2018.05.21
  - tick.q

Tables : Quote & Trade



- kdb-tick
  - SetupTickerplant.ps1
  - icons
  - references
  - sym
    - sym
    - 2008.02.01
    - 2017.08.23
    - 2018.11.29
      - quote
      - trade
  - tick
    - h.q
    - r.q
    - sym.q
    - u.q
    - RealTimeTradeWithAsofQuotes.q
  - sym2018.05.21
  - tick.q

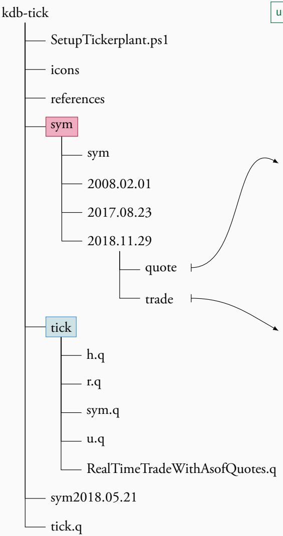
c	t	f	a
date	d		
time	n		(s)
sym	s		g
bid	f		
ask	f		
bsize	j		
asize	j		

c	t	f	a
date	d		
time	n		(s)
sym	s		g
price	f		
size	j		

Joins: Latest Trade

```
upsert select (last) by sym from quote
```

Sym	Time	Bid	Ask	Bsize	Asize
IBM.N	0D08:04:45.262855000	191.17	191.19	422	572
MSFT.O	0D08:04:45.162728000	45.15	45.16	609	884

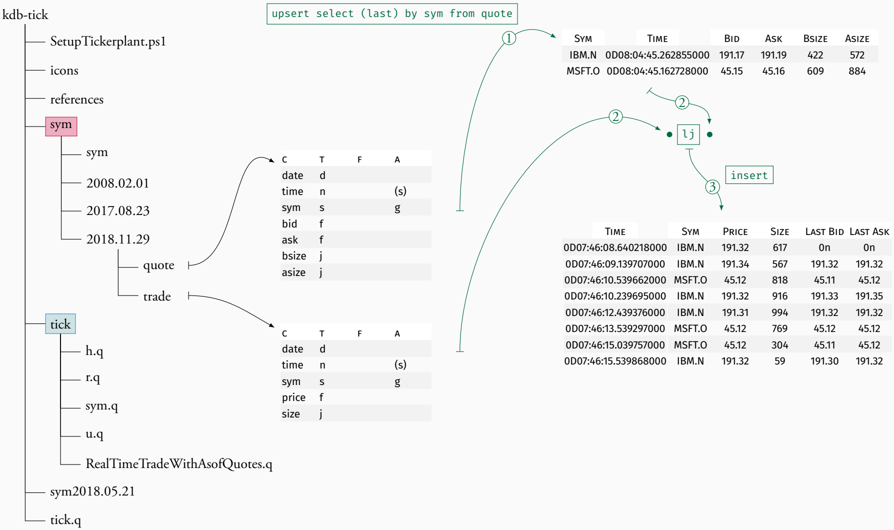


- kdb-tick
  - SetupTickerplant.ps1
  - icons
  - references
  - sym
    - sym
    - 2008.02.01
    - 2017.08.23
    - 2018.11.29
      - quote
      - trade
  - tick
    - h.q
    - r.q
    - sym.q
    - u.q
    - RealTimeTradeWithAsofQuotes.q
  - sym2018.05.21
  - tick.q

c	t	f	a
date	d		
time	n		(s)
sym	s		g
bid	f		
ask	f		
bsize	j		
asize	j		

c	t	f	a
date	d		
time	n		(s)
sym	s		g
price	f		
size	j		

Temporal Joins : Latest Bid/Ask Asof Trade : Method 1



```
kdb-tick
├── SetupTickerplant.ps1
├── icons
├── references
├── sym
│   ├── sym
│   ├── 2008.02.01
│   ├── 2017.08.23
│   └── 2018.11.29
│       ├── quote
│       └── trade
├── tick
│   ├── h.q
│   ├── r.q
│   ├── sym.q
│   ├── u.q
│   └── RealTimeTradeWithAsofQuotes.q
├── sym2018.05.21
└── tick.q
```

c	t	f	a
date	d		
time	n		(s)
sym	s		g
bid	f		
ask	f		
bsize	j		
asize	j		

c	t	f	a
date	d		
time	n		(s)
sym	s		g
price	f		
size	j		

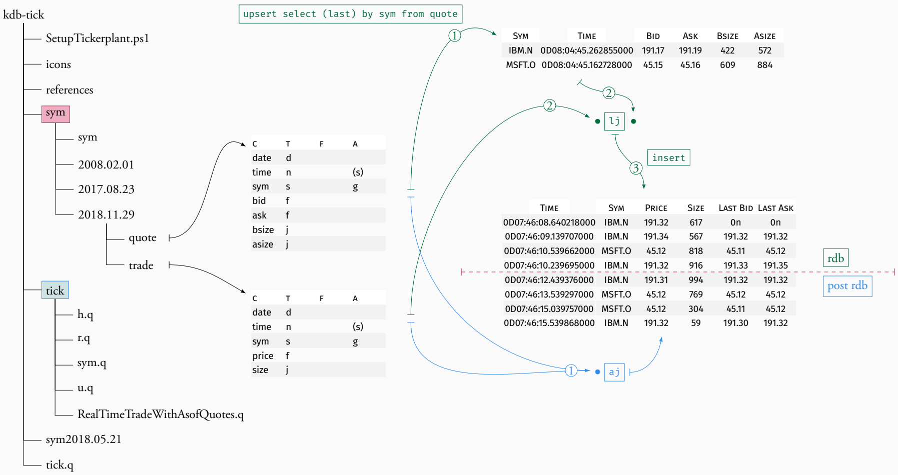
```
upsert select (last) by sym from quote
```

```
insert
```

Sym	Time	Bid	Ask	Bsize	Asize
IBM.N	0D08:04:45.262855000	191.17	191.19	422	572
MSFT.O	0D08:04:45.162728000	45.15	45.16	609	884

Time	Sym	Price	Size	Last Bid	Last Ask
0D07:46:08.640218000	IBM.N	191.32	617	0n	0n
0D07:46:09.139707000	IBM.N	191.34	567	191.32	191.32
0D07:46:10.539662000	MSFT.O	45.12	818	45.11	45.12
0D07:46:10.239695000	IBM.N	191.32	916	191.33	191.35
0D07:46:12.439376000	IBM.N	191.31	994	191.32	191.32
0D07:46:13.539297000	MSFT.O	45.12	769	45.12	45.12
0D07:46:15.039757000	MSFT.O	45.12	304	45.11	45.12
0D07:46:15.539868000	IBM.N	191.32	59	191.30	191.32

Temporal Joins : Latest Bid/Ask Asof Trade : Method 2



- kdb-tick
  - SetupTickerplant.ps1
  - icons
  - references
  - sym
    - sym
    - 2008.02.01
    - 2017.08.23
    - 2018.11.29
      - quote
      - trade
  - tick
    - h.q
    - r.q
    - sym.q
    - u.q
    - RealTimeTradeWithAsofQuotes.q
  - sym2018.05.21
  - tick.q

C	T	F	A
date	d		
time	n		(s)
sym	s		g
bid	f		
ask	f		
bsize	j		
asize	j		

C	T	F	A
date	d		
time	n		(s)
sym	s		g
price	f		
size	j		

```
upsert select (last) by sym from quote
```

Sym	Time	Bid	Ask	Bsize	Asize
IBM.N	0D08:04:45.262855000	191.17	191.19	422	572
MSFT.O	0D08:04:45.162728000	45.15	45.16	609	884

Time	Sym	Price	Size	Last Bid	Last Ask
0D07:46:08.640218000	IBM.N	191.32	617	0n	0n
0D07:46:09.139707000	IBM.N	191.34	567	191.32	191.32
0D07:46:10.539662000	MSFT.O	45.12	818	45.11	45.12
0D07:46:10.239695000	IBM.N	191.32	916	191.33	191.35
0D07:46:12.439376000	IBM.N	191.31	994	191.32	191.32
0D07:46:13.539297000	MSFT.O	45.12	769	45.12	45.12
0D07:46:15.039757000	MSFT.O	45.12	304	45.11	45.12
0D07:46:15.539868000	IBM.N	191.32	59	191.30	191.32

Temporal Joins

function	description
aj/aj0	get the value of a field in one table asof the value in another
asof	like aj except can be passed a dictionary as left argument of the sym/times to return data asof
wj	generalization of aj with an additional argument list of pairs of start/end times that specify windows
raj	reverse aj for most recent future looking data points; simplest implementation negate the date

TEMPORAL ARITHMETIC

function	description
date ± integers	integers representing day counts can be added/subtracted from dates and as a result dates themselves can be added to/subtracted from to give day counts
time ± integers	integers representing milliseconds can be added/subtracted from times
datetime ± floats	the integer part represents a day count and the fractional part represents a fraction of a day `2003.03.23T08:31:53.000 + 2.5%24`
timestamp ± floats	does not work
timestamp ± integers	integers representing milliseconds can be added/subtracted
timestamp ± 1D	integers with trailing ”D” representing days can be added/subtracted

timestamp × integers	multiply minutes/hours/days etc by integer
	`0D00:12*3`
	`0D01:00*2`
	`2012.02.28D00:18:00+(0D00:10+til 115)`

.Q.addmonths	`.Q.addmonths[.z.P;12]`
addyear	`addyear[.z.P;1]`

EXAMPLE: AJ VERSUS WJ

Listing 1: q/Kdb+ code for aj vs. wj

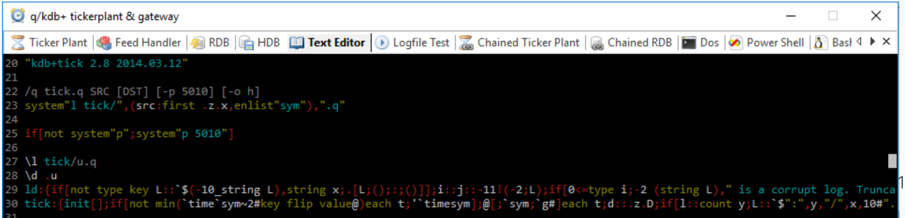
```
t:([]sym:3#`ibm;time:10:01:01 10:01:04 10:01:08;price:100 101 105)
q:([]sym:9#`ibm;time:10:01:01+til 9;ask:101 103 103 104 104 107 108 107 108;
     bid:98 99 102 103 103 104 106 106 107)
w:-2 1+\:t.time
w:0 10+\:t.time
wj[w;`sym`time;t;(q;(max;`ask);(min;`bid))]
aj[`sym`time;t;q]
```

asks and bids				trades		start/end times of bins		wj		aj	
sym	time	ask	bid	time	price	w1	w2	ask	bid	ask	bid
‘ibm	10:01:01	101	98	10:01:01	100	10:01:01	10:01:11	108	98	101	98
‘ibm	10:01:02	103	99								
‘ibm	10:01:03	103	102								
‘ibm	10:01:04	104	103	10:01:04	101	10:01:04	10:01:14	108	103	104	103
‘ibm	10:01:05	104	103								
‘ibm	10:01:06	107	104								
‘ibm	10:01:07	108	106								
‘ibm	10:01:08	107	106	10:01:08	105	10:01:08	10:01:18	108	106	107	106
‘ibm	10:01:09	108	107								
‘ibm	10:01:10	108	107								

Architecture

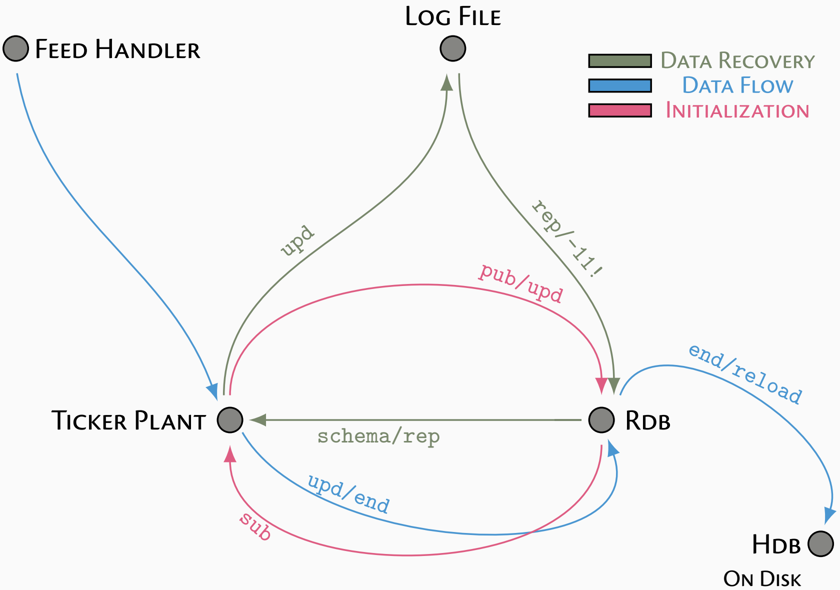
Single Console Tickerplant

process	description
Tickerplant	writes to logfile in addition to publishing data to RDB and other subscribers
Feedhandler	connects, recieves & normalizes exchange/market data before feeding it to the tickerplant
Rdb	real-time in-memory store for current day's data
Hdb	partitioned on-disk stores for historical data
Logfile	on-disk file that stores alll intra-day updates for potential re-play during recovery
Chained TP	subscribes to main (zero-latency) tickerplant and recieves delayed updates and keeps no log file
Chained Rdb	subscribes to either the main or chained tickerplant

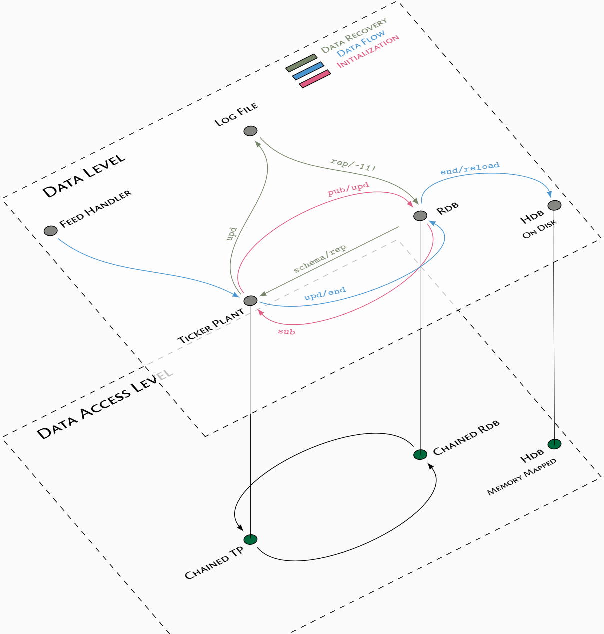


```
"kdb+tick 2.8 2014.03.12"

/q tick.q SRC [DST] [-p 5010] [-o h]
system"l tick/",(src:first .z.x,enlist"sym"),".q"

if[not system"p";system"p 5010"]

\l tick/u.q
\d .u
ld:{if[not type key L::`$(-10_string L),string x;.[L;();:;()]];i::j::-11!(-2;L);if[0<=type i;-2 (string L)," is a corrupt log. Trunca
tick:{init[];if[not min(`time`sym~2#key flip value@)each t;'`timesym];@[;`sym;`g#]each t;d::.z.D;if[l::count y;L::`$":",y,"/",x,10#".
```

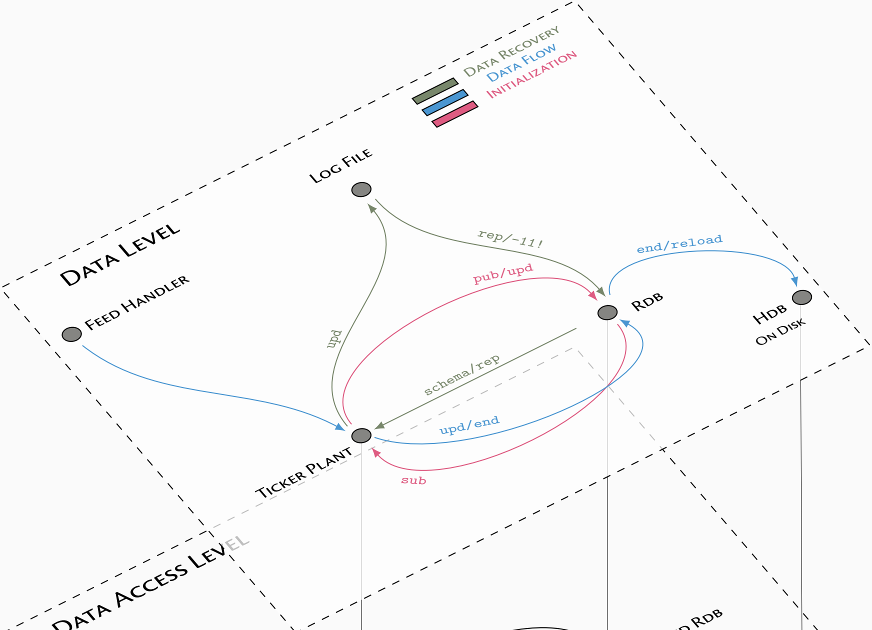
Data Layer


Log File
Feed Handler
Data Recovery
Data Flow
Initialization
upd
rep/-11!
pub/upd
end/reload
Ticker Plant
Rdb
schema/rep
upd/end
sub
Hdb
On Disk

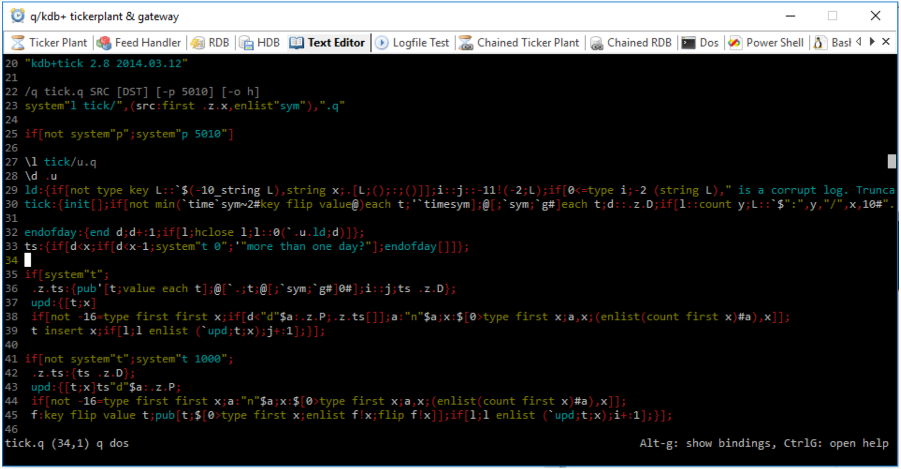
Data Access Layer



Application Layer



Single Console Tickerplant Screenshot



```
"kdb+tick 2.8 2014.03.12"

/q tick.q SRC [DST] [-p 5010] [-o h]
system"l tick/",(src:first .z.x,enlist"sym"),".q"

if[not system"p";system"p 5010"]

\l tick/u.q
\d .u
ld:{if[not type key L::`$(-10_string L),string x;.[L;();:;()]];i::j::-11!(-2;L);if[0<=type i;-2 (string L)," is a corrupt log. Trunca
tick:{init[];if[not min(`time`sym~2#key flip value@)each t;'`timesym];@[;`sym;`g#]each t;d::.z.D;if[l::count y;L::`$":",y,"/",x,10#".

endofday:{end d;d+:1;if[l;hclose l;l::0(`.u.ld;d)]};
ts:{if[d<x;if[d<x-1;system"t 0";'"more than one day?"];endofday[]]};

if[system"t";
 .z.ts:{pub'[t;value each t];@[`.;t;@[;`sym;`g#]0#];i::j;ts .z.D};
 upd:{[t;x]
 if[not -16=type first first x;if[d<"d"$a:.z.P;.z.ts[]];a:"n"$a;x:$[0>type first x;a,x;(enlist(count first x)#a),x]];
 t insert x;if[l;l enlist (`upd;t;x);j+:1];}];

if[not system"t";system"t 1000";
 .z.ts:{ts .z.D};
 upd:{[t;x]ts"d"$a:.z.P;
 if[not -16=type first first x;a:"n"$a;x:$[0>type first x;a,x;(enlist(count first x)#a),x]];
 f:key flip value t;pub[t;$[0>type first x;enlist f!x;flip f!x]];if[l;l enlist (`upd;t;x);i+:1];}];

tick.q (34,1) q dos                                                              Alt-g: show bindings, CtrlG: open help
```

Installation

Install Using Powershell

Download powershell script from :

HTTPS://GITHUB.COM/ROHANSHILOH/TICKERPLANT.AI

PS1 SCRIPT

- Downloads 64-bt console2 and extracts it
- Downloads all required q files from code.kx.com
- Downloads all reference white-papers from code.kx.com
- Copies console2 config file into appropriate folder
- Checks which verson of python and powershell are installed
- Starts single console tickerplant

Setup Python, Bash & VC Compiler

PYTHON SETUP

- Install Anaconda and iPython
- Identify your python path and add it to the PATH environment
- follow instructions for experimental setup of pyQ here `https://pyq.enlnt.com/install/install.html#experimental-support-for-windows`
- Set console2's shell value :

```
cmd.exe /k C:\Anaconda\IPython.exe
```

BASH SETUP

- Install the Linux Subsystem on Windows
- Set console2's shell value :

```
%SystemRoot%\system32\bash.exe
```

Conclusion

Summary

The application of REAL-TIME ITERATIVE machine learning algorithms to stock market order book data can most *efficiently* be done using the Q/KDB+/PYQ/PYTHON stack.

`github.com/rohanshiloh/tickerplant.ai`

Next Steps

- Tick PCA & Data Visualization of the Order Book at Depth
- Frequentist, Bayesian, Generalized & Regularized (Ridge L_2 & Lasso L_1) Linear Regression Models
- Bayesian (Multi-Strategy & Multi-Period) Portfolio Optimization
- Structural (Regularized) Covariance

References i

- N. Coulter, *Order book: a kdb+ intraday storage and access methodology*, 2014.
  Coulter covers the fine balance between processing/storage of an order book and it's subsequent retrieval and analysis. Different trade table schemas and accessor functions are presented and dicussed.
- D. Easley and M. O'Hara, *Price, trade, size and information in securities markets*, 1987.

References ii

- C. M. C. Lee and M. J. Ready, *Inferring trade direction from intraday data*, 1991.
  Lee and Ready's seminal paper on tick data anaysis came out in 1991. They did their research on a supercomputer at Cornell; likely not running q/kdb. Oddly enough Canada supported their research. The paper outlines various 'tick tests' for an uptick, downtick, a zero-uptick and a zero-downtick. The additional function tradeclass in the file tickaddendum.q provides defintions for each of these classes and appends them as a column to the trade table.
- C. McCarthy, *Intraday writedown solutions*, 2014.
  This paper will be incredibly useful to independent researchers running a tickerplant at home on 4core/4gb.

References iii

N. Perrem, *Building real-time tick subscribers*, 2014.
Perrem's excellent paper is a good introduction to standard ticker plant architecture. Chapter 4 covers how to build a custom temporally joined trade aj quote architecture with it's own binary logfile.