Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Национальный исследовательский университет - Высшая школа
экономики»

Факультет экономических наук

Образовательная программа «Экономика»

Курсовая работа

Оценка риска по большим данным

Студент группы БЭК155

Лукьянов Андрей Андреевич

Научный руководитель

Лапшин Виктор Александрович

2018

Содержание

Введение

В данной курсовой работе сравниваются оценки метрики риска Value-at-Risk, полученные на данных разной частотности. В качестве датасета выступает журнал торговых сделок с биржи MOEX за 250 торговых дней, начиная со 2го марта 2015 года по 29 февраля 2016 года.

В разделе «Выбор ПО» я подробно расскажу, какими решениями я воспользовался для того, чтобы сделать работу с «Big Data» возможной.

В разделе «Ход работы» я подробно опишу, что я делал с выбранным ПО, чтобы сделать работу с «Big Data» возможной.

В разделе «Анализ рисков» я распишу, как я вычислял метрики, и проведу их сравнение для данных разной частотности.

Выбор ПО

Журнал торговых сделок за 250 дней – это примерно 103 миллиона записей. Очевидно, в Excel это не обработаешь, да и одного Python или R будет тоже недостаточно, поэтому было принято решение воспользоваться базой данных.

MongoDB

MongoDB [3] – это система управления базами данных (СУБД), которая обеспечила мне удобный доступ к датасету.

MongoDB запускается на виртуальной машине через loopback подобно Jupyter lab/notebook с помощью одной консольной команды.

Выбор пал на MongoDB по нескольким причинам. Во-первых, MongoDB очень удобно интегрирована с Python через библиотеку pymongo. Во-вторых, MongoDB бесплатна. В-третьих, с MongoDB легко работать на своем личном компьютере: она хорошо оптимизирована и нетрудна в освоении.

Studio3T

Одна особенность MongoDB – это отсутствие нативного интерфейса. Без стороннего GUI с MongoDB можно работать только через консоль, поэтому я воспользовался Studio3T: реализацией API для MongoDB [4].

Теперь немного более подробно о преимуществах работы с Studio3T. Во-первых, Studio3T имеет удобный графический интерфейс для построения запросов в базу данных MongoDB. Во-вторых, в Studio3T сама переводит запрос в нужный синтаксис: можно написать запрос в SQL, а Studio3T переведет этот же запрос, например, в pymongo или mongo shell самостоятельно. В-третьих, разработчики предоставляют бесплатную лицензию на 30 дней, но, несмотря на то, что мне ее не хватило, я легко получил бесплатную лицензию на год как студент, предоставив справку из учебной части.

Python

Выбор Python не требует детального объяснения: высокоуровневый удобный язык с большим числом библиотек под разные задачи. В моем случае, к стандартному набору для статистического исследования из pandas, numpy, statsmodels и matplotlib добавился еще pymongo. В качестве API я работал и в PyCharm, и в Jupyter lab. Приятным бонусом является то, что для IPython в GitHub есть приятное оформление.

Ход работы

Запись журнала торговых операций в базу данных

Для того, чтобы начать работать с базой данных, сначала нужно в нее записать имеющиеся данные. Как я уже упомянул выше, датасет включает 250 торговых дней, по .txt файлу со сделками на каждый день.

Я написал скрипт date_reader.py [2], который поочередно считывает записи из каждого .txt файла и делает записи в коллекцию trade_logs базы данных MICEX в MongoDB. Скрипт не только считывает, но и правильно форматирует данные: даты записываются в формате datetime, числа в float64, тикеры, как и должно быть, в string.

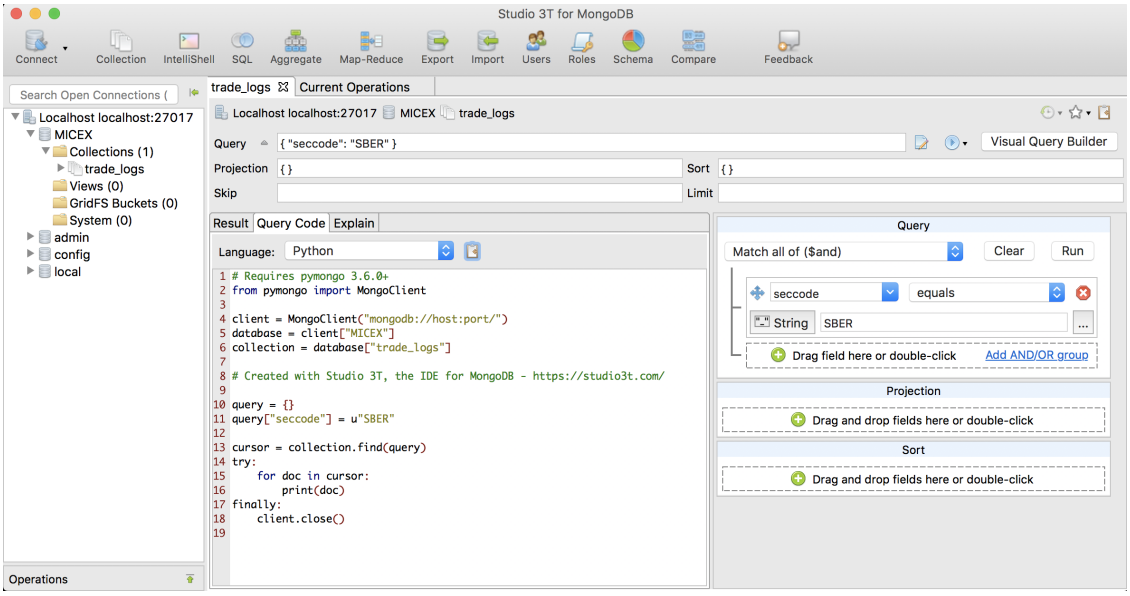
Данных действительно много, поэтому процесс записи длился около 14 часов.

Формирование запросов

Теперь, когда база данных уже записана в MongoDB, для более быстрого выполнения запросов нужно было проиндексировать записи. Это было выполнено с помощью Studio3T.

Далее я, используя GUI Studio3T, строил нужные мне запросы. После этого уже можно было брать готовый код запроса в python и копировать его в jupyter lab.

Ниже скриншот с примером построения запроса на все сделки по Сбербанку. Справа, под заголовком Query, нужный нам вариант запроса. Во вкладке Query Code Studio3T выдает готовый код в Python.



Работа в Jupyter lab

Мы получили код запроса в python в Studio3T, осталось только его скопировать в Jupyter lab и подождать пока выполнится запрос. MongoDB ведет историю запросов, поэтому один и тот же запрос, отправленный во второй раз выполнится быстрее, чем в первый.

В действительности, время выполнения первого запроса в MongoDB занимает примерно 1/3 времени записи полученных данных в DataFrame, поэтому в данном случае IPython в некоторой степени bottleneck.

Теперь, когда данные загружены, можно пользоваться любыми инструментами для анализа. Все расчеты находятся в файле Thesis.ipynb в репозитории курсовой [2].

Анализ рисков

Метод оценки

Теперь можно приступить непосредственно к методу оценки рисоков.

Для оценки рисков воспользуемся метрикой Value-at-Risk. Value-at-Risk – это нижняя граница доверительного интервала доходности портфеля [1]. В данном случае мы будем рассматривать только портфели состоящие из одного тикера.

Перед подсчетом метрики Value-at-Risk нужно сделать предположение о распределении данных. Тест Харке-Бера на нормальность позволит нам установить, нормально ли распределены доходности. В качестве оценок параметров распределений будем использовать среднее и дисперсию доходностей на заданной частоте, каждый раз увеличивая выборку.

Для анализа я выбрал 3 акции разной ликвидности (LKOH, TATN, YNDX) и одну наиболее ликвидную облигацию (SU24018RMFS2). P-value теста Харке-Бера для всех тикеров был равен 0, значит целесообразно было использовать нормальное распределение для оценки VaR.
Для всех тикеров я вывел графики VaR, оцененой с интервалом в неделю, день, час, минуту и секунду. Для того чтобы сравнить частотности между собой я также вывел сравнительную диаграмму, которая показывает оценки за одинаковый период одним цветом.

Для наиболее ликвидных облигаций подобные графики считаются довольно долго: вывов графиков для LKOH занят больше часа. Графики для других тикеров посчитались гораздо быстрее просто потому что у них было на порядок меньше сделок за год.

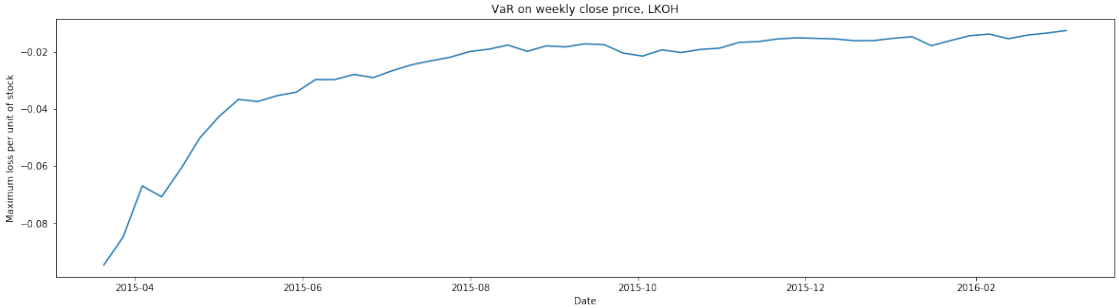
В файле Thesis.ipynb в репозитории курсовой [2] можно также просмотреть количество сделок для всех 516 тикеров торговавшихся за тот период.

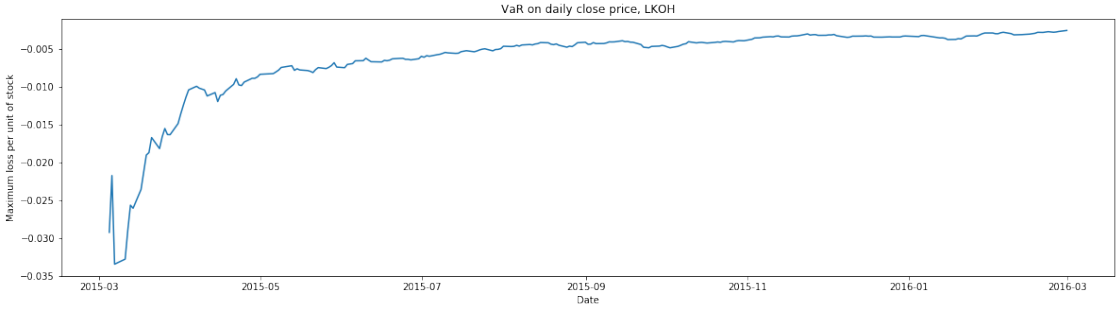
Алгоритм

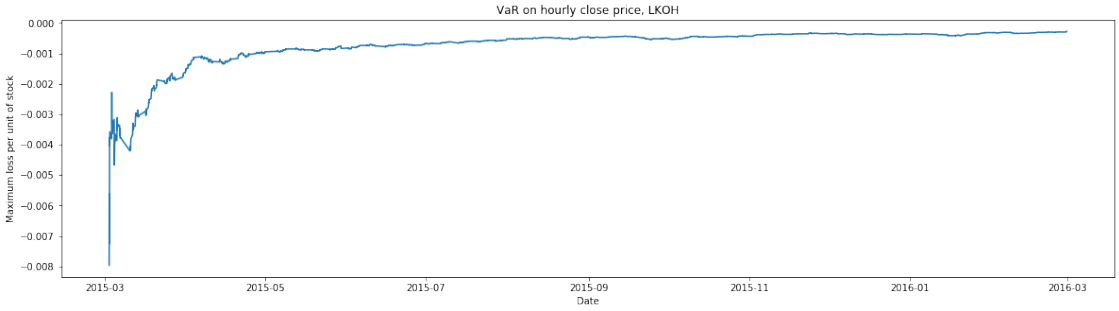
Алгоритм построен на том, что отправляет правильный запрос базе данных, который группирует данные по временным промежуткам, после чего из каждой группы база данных отдает самое последнее значение как цену закрытия. Далее все это это подается функции, считающей VaR.

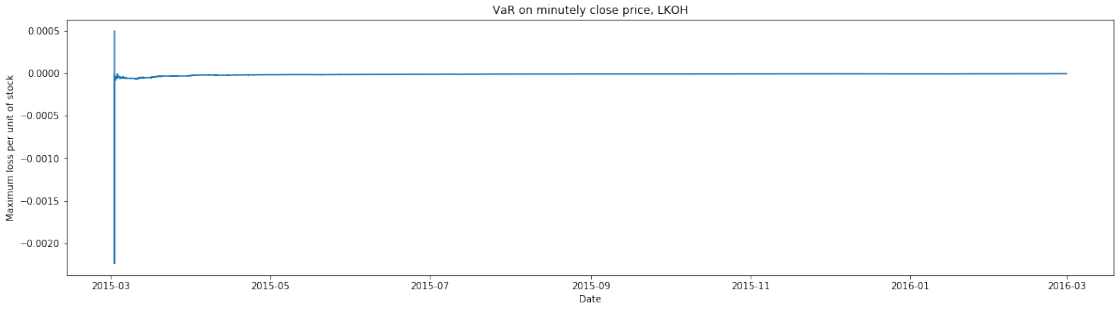
Для подсчета более высокочастного VaR этот алгоритм работает быстрее, чем если сначала ипортировать все данные по тикеру из базы данных в датафрейм, а потом выбирать оттуда цены закрытия. Алгоритм не требователен к многопоточности процессора: одного потока достаточноно и для базы данных, и для подсчета VaR. Возможно алгоритм можно ускорить на компьютере с более быстрой памятью.

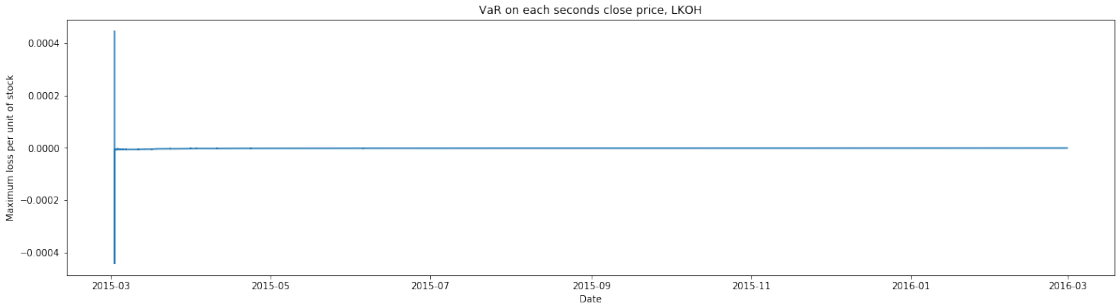
LKOH

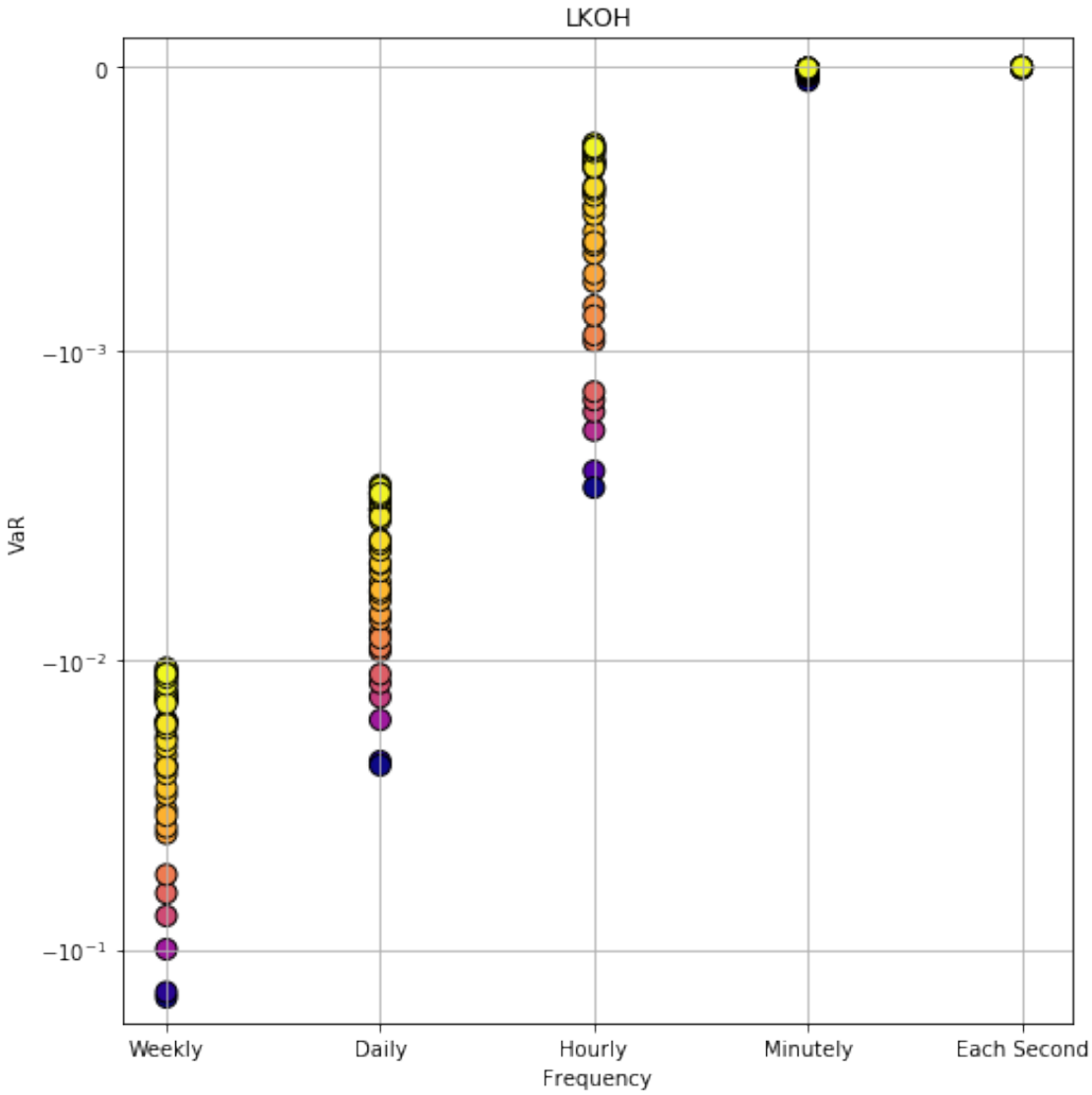

VaR on weekly close price, LKOH
Maximum loss per unit of stock
−0.02
−0.04
−0.06
−0.08
2015-04
2015-06
2015-08
2015-10
2015-12
2016-02
Date


VaR on daily close price, LKOH
Maximum loss per unit of stock
−0.005
−0.010
−0.015
−0.020
−0.025
−0.030
−0.035
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

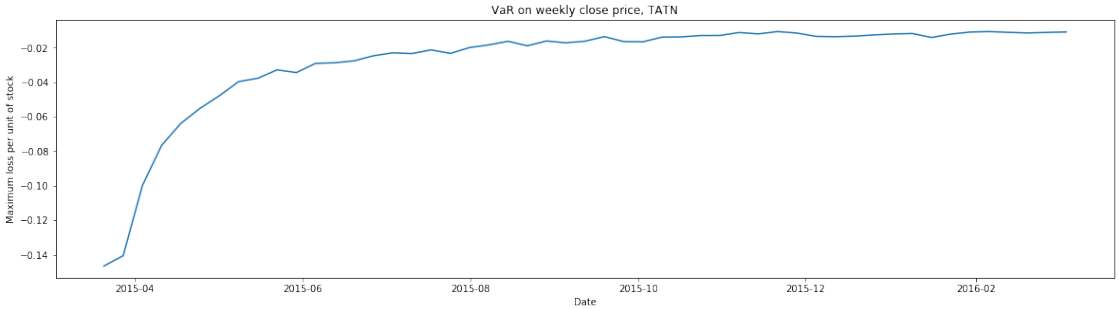

VaR on hourly close price, LKOH
Maximum loss per unit of stock
0.000
−0.001
−0.002
−0.003
−0.004
−0.005
−0.006
−0.007
−0.008
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

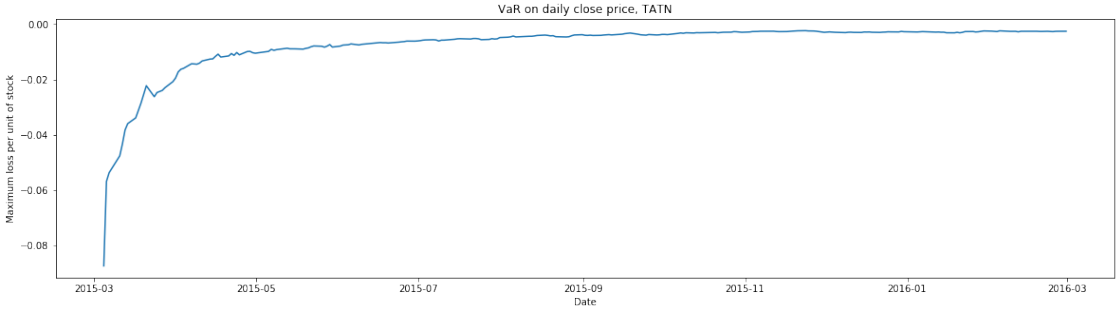

VaR on minutely close price, LKOH
Maximum loss per unit of stock
0.0005
0.0000
−0.0005
−0.0010
−0.0015
−0.0020
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

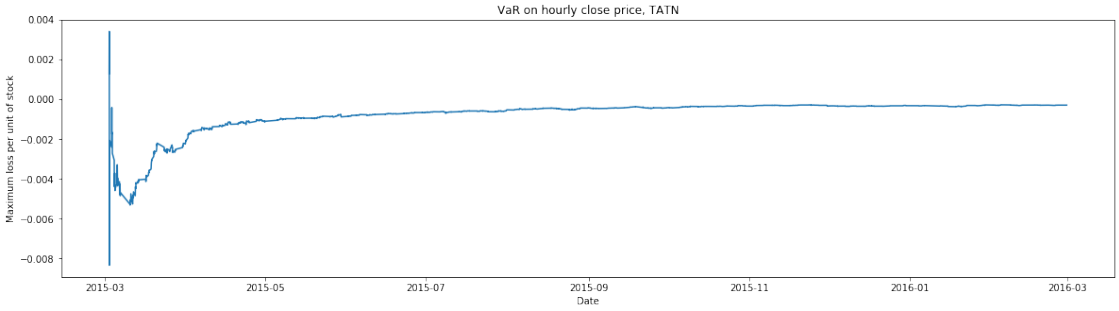

VaR on each seconds close price, LKOH
Maximum loss per unit of stock
0.0004
0.0002
0.0000
−0.0002
−0.0004
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

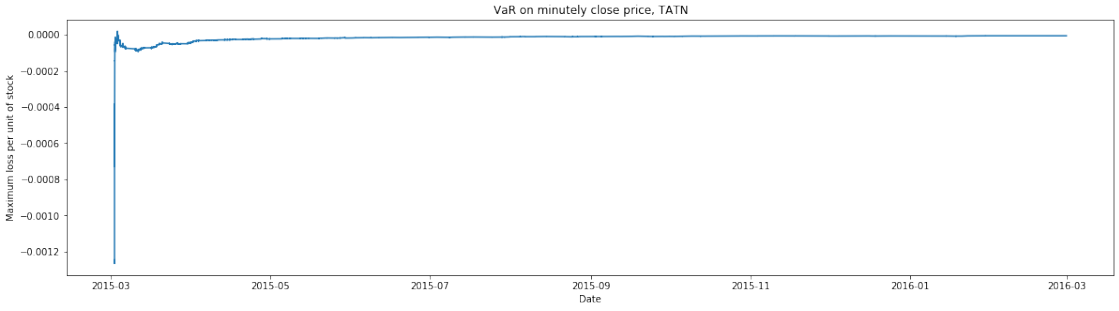

LKOH
0
-10^{-3}
-10^{-2}
-10^{-1}
VaR
Weekly
Daily
Hourly
Minutely
Each Second
Frequency

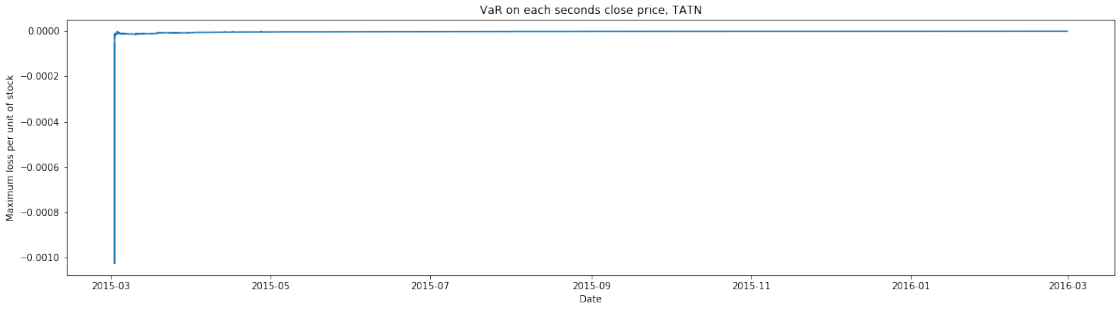
TATN

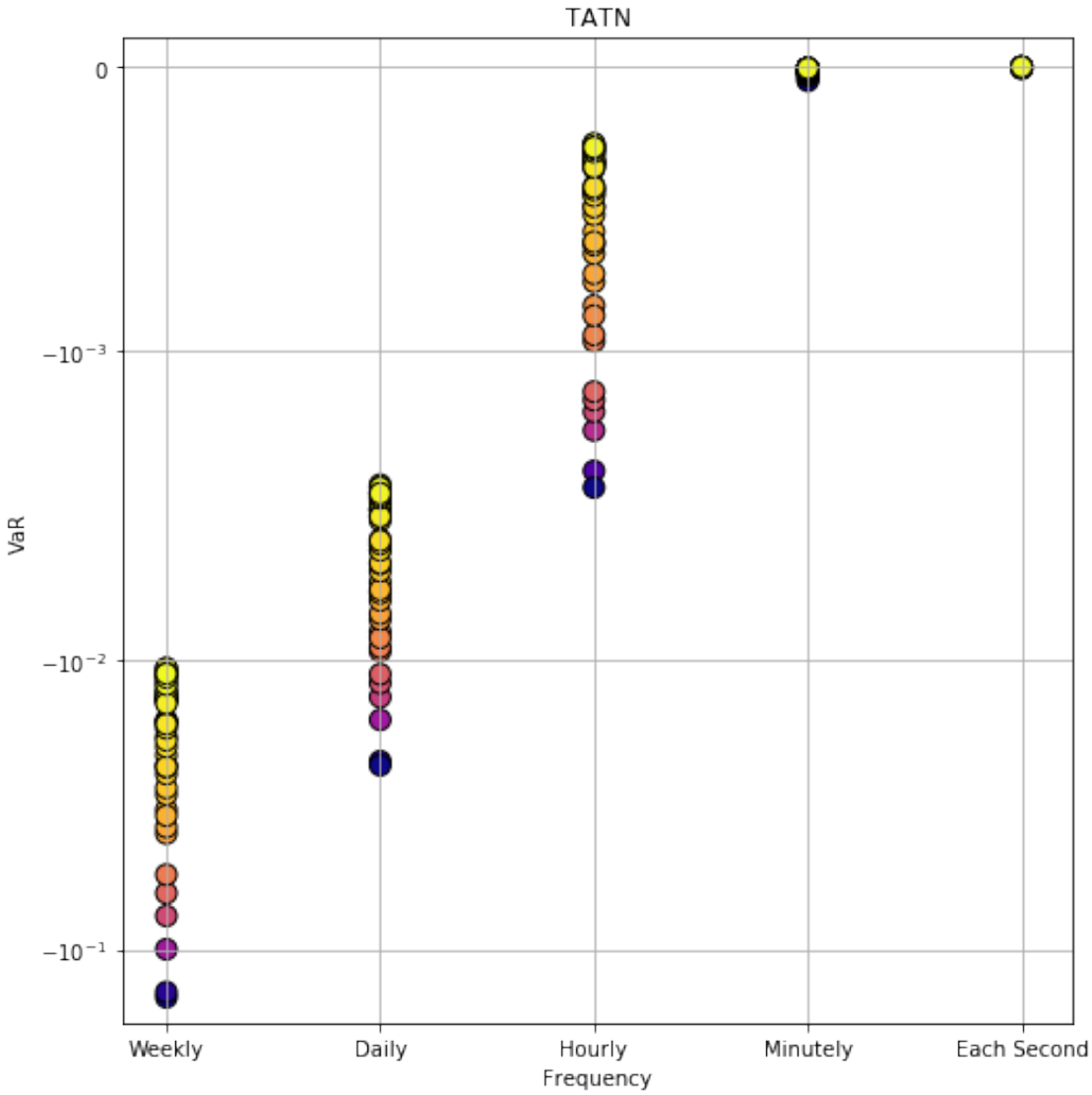

VaR on weekly close price, TATN
Maximum loss per unit of stock
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
−0.14
2015-04
2015-06
2015-08
2015-10
2015-12
2016-02
Date


VaR on daily close price, TATN
Maximum loss per unit of stock
0.00
−0.02
−0.04
−0.06
−0.08
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

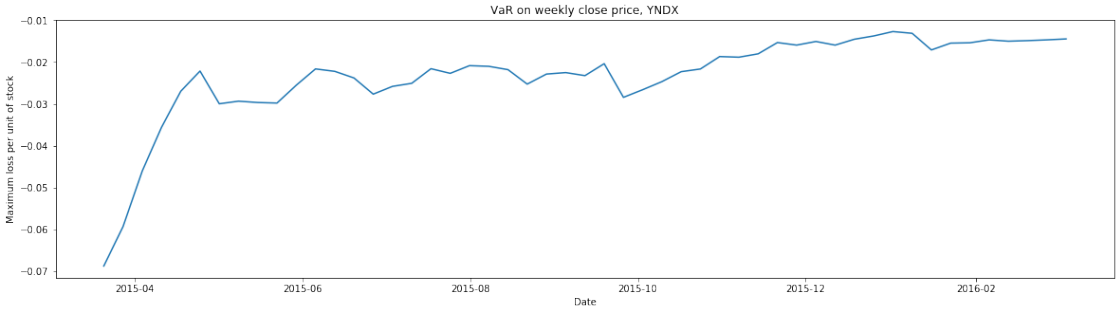

VaR on hourly close price, TATN
Maximum loss per unit of stock
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

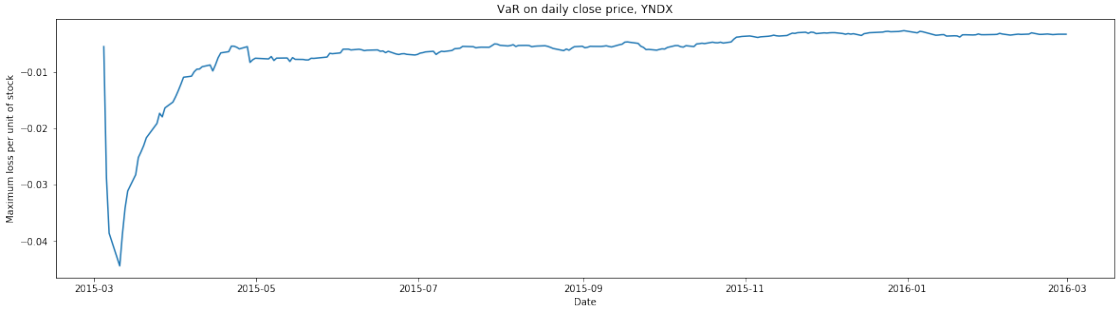

VaR on minutely close price, TATN
Maximum loss per unit of stock
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
−0.0012
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

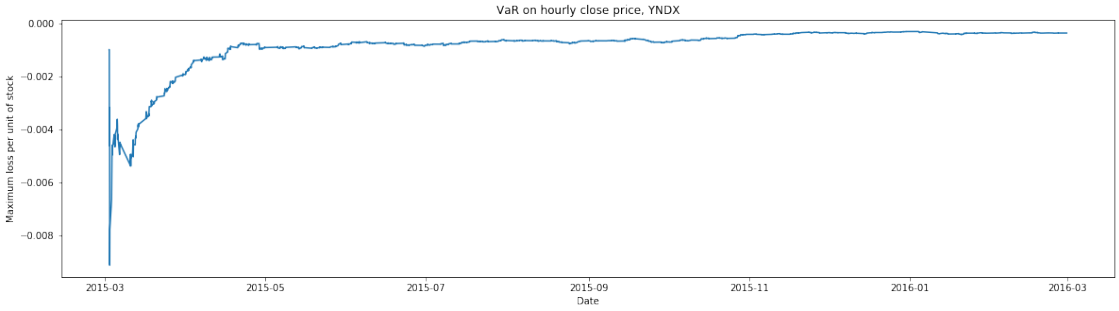

VaR on each seconds close price, TATN
Maximum loss per unit of stock
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

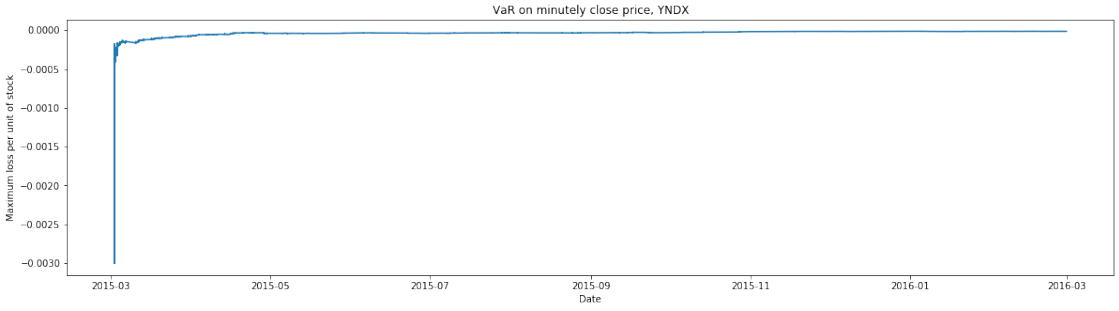

TATN
0
-10^{-3}
-10^{-2}
-10^{-1}
VaR
Weekly
Daily
Hourly
Minutely
Each Second
Frequency

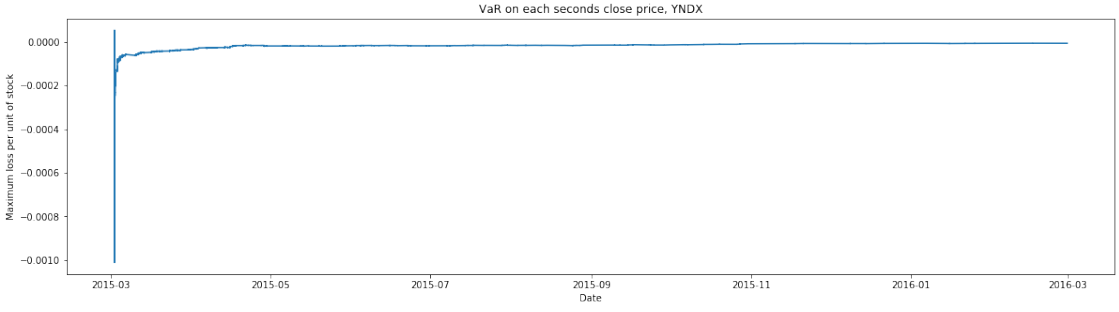
YNDX

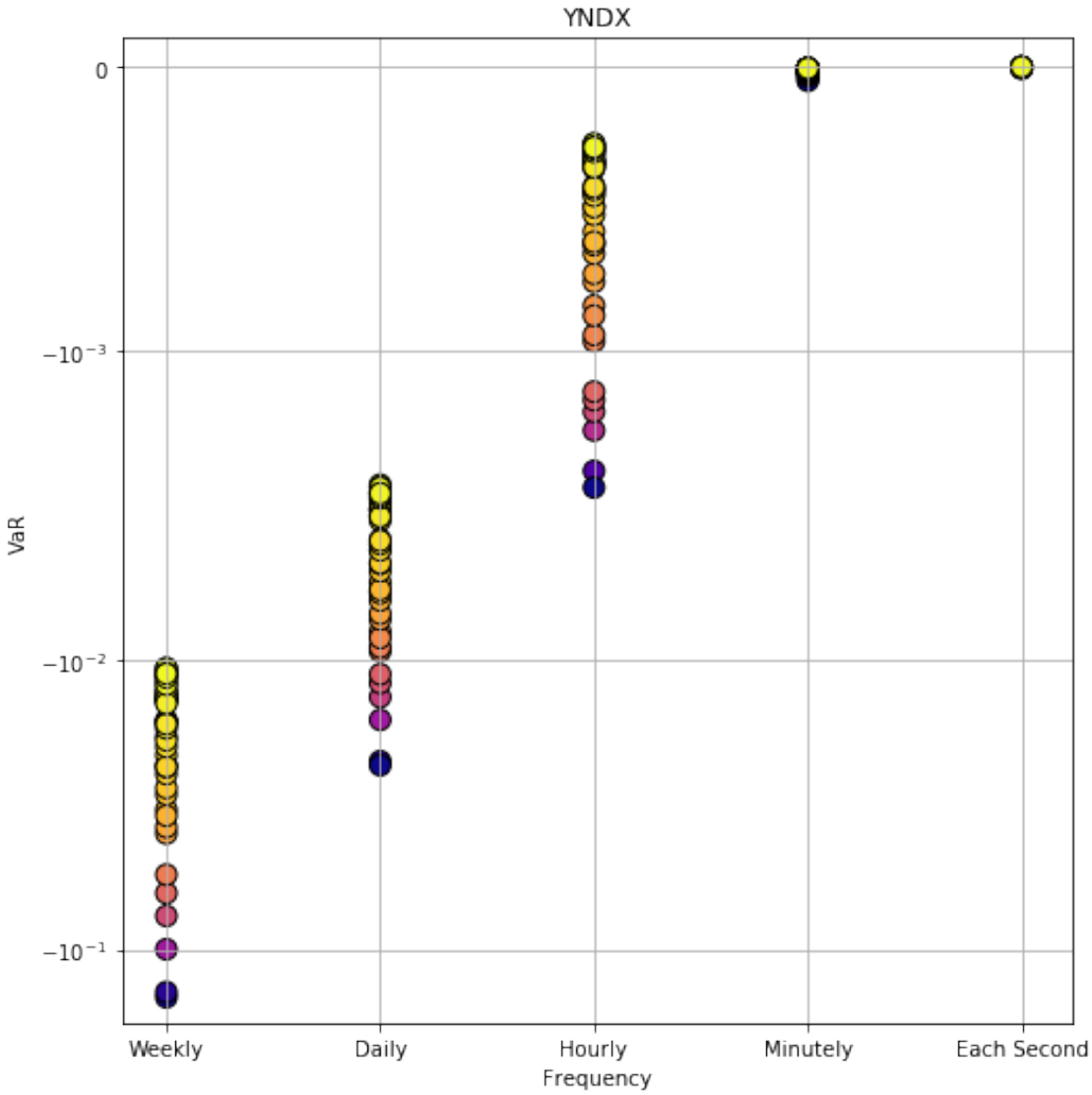

VaR on weekly close price, YNDX
Maximum loss per unit of stock
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
2015-04
2015-06
2015-08
2015-10
2015-12
2016-02
Date


VaR on daily close price, YNDX
Maximum loss per unit of stock
−0.01
−0.02
−0.03
−0.04
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

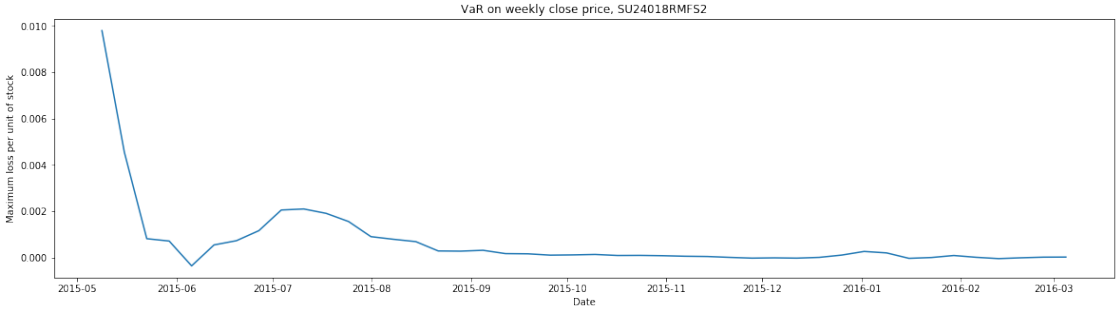

VaR on hourly close price, YNDX
Maximum loss per unit of stock
0.000
−0.002
−0.004
−0.006
−0.008
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

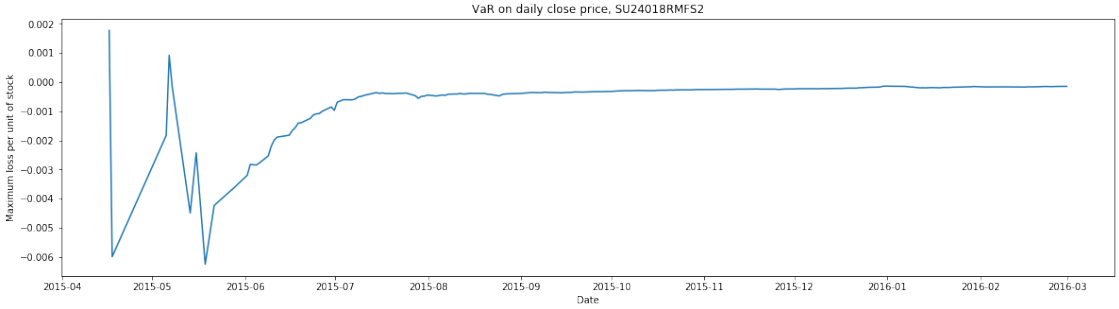

VaR on minutely close price, YNDX
Maximum loss per unit of stock
0.0000
−0.0005
−0.0010
−0.0015
−0.0020
−0.0025
−0.0030
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

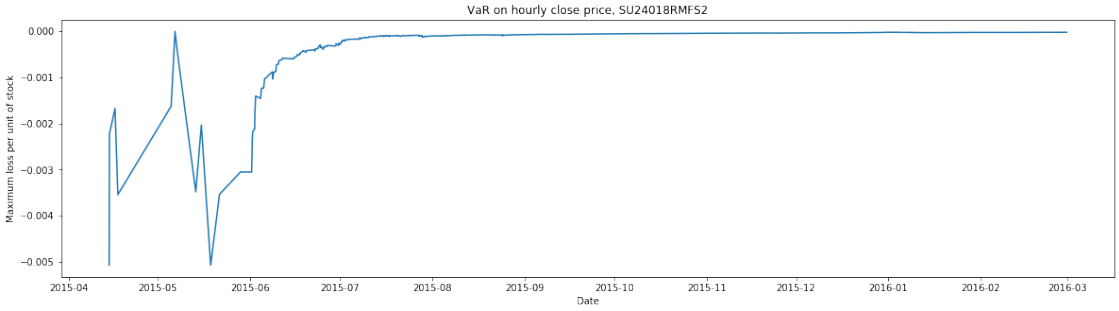

VaR on each seconds close price, YNDX
Maximum loss per unit of stock
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
Date

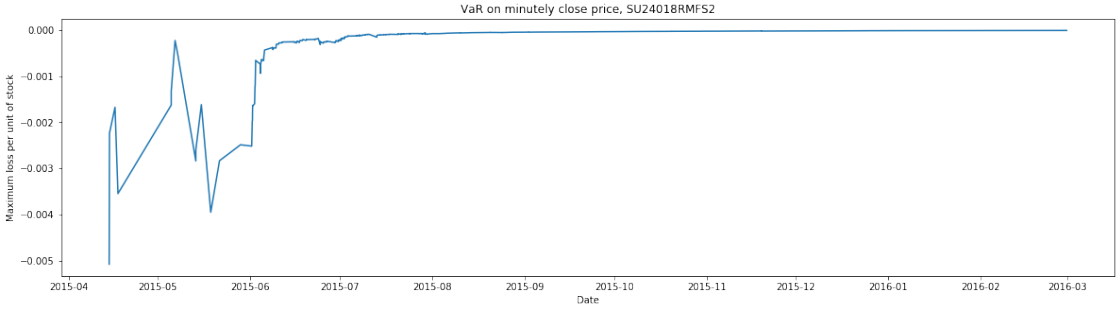

YNDX
0
-10^{-3}
-10^{-2}
-10^{-1}
VaR
Weekly
Daily
Hourly
Minutely
Each Second
Frequency

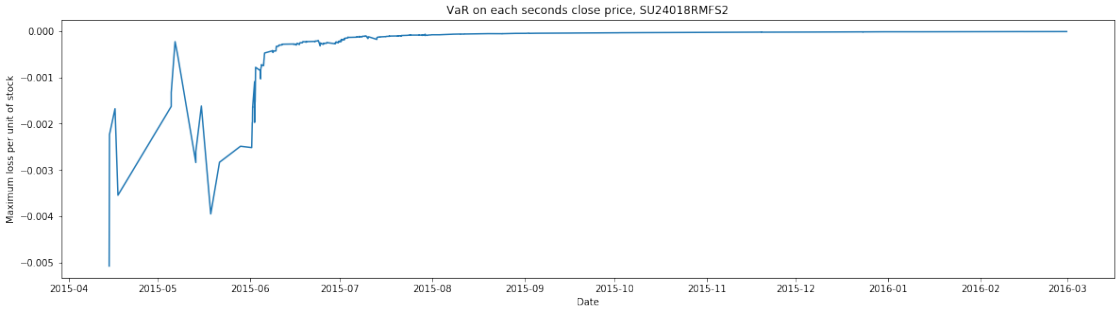
SU24018RMFS2

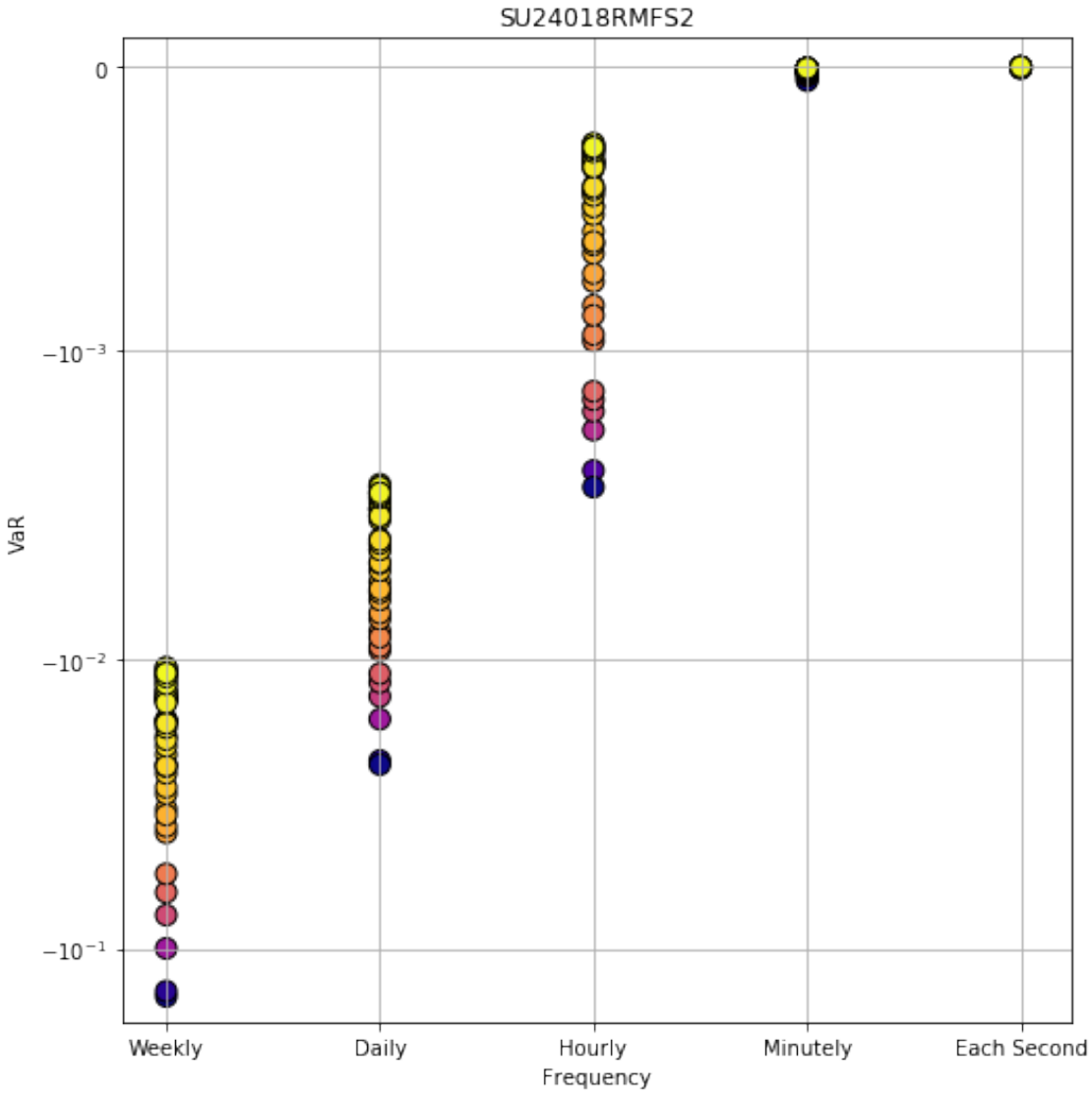











SU24018RMFS2
0
-10^{-3}
-10^{-2}
-10^{-1}
VaR
Weekly
Daily
Hourly
Minutely
Each Second
Frequency

Вывод

Для Лукойла общий паттерн виден отчетливо: чем выше частота оценки, тем менее стабилен VaR в самом начале и тем быстрее он увеличивается. В случае с посекундной оценкой есть относительно большие колебания в самом начале, но потом VaR становится прямой линией. Сравнительная диаграмма только подтверждает вышесказанное.

То же самое повторяется для Татнефти и Яндекса. Отличие только в том, что по мере уменьшения ликвидности вышеупомянутые эффекты происходят медленее.

В случае с облигацией SU24018RMFS2 мы наблюдаем случай с очень маленькой выборкой. Учащение оценки VaR больше, чем на день ничего не меняет, что заметно на графиках. В отличии от предыдущих случаев, где в качестве оценки VaR можно брать посекундную оценку за 2-4 недели, здесь можно, взять, например, поминутную за полгода. Можно также наблюдать, что VaR у облигации изначально находится в более узком интервале, что является показателем того, что облигации являются менее рисковым активом, чем акции.

Все что мы наблюдали выше похоже на закон больших чисел. Бо̀льшая выборка лучше приближает оценку параметра.

Итого, ничего необычного. Большая часть времени ушла на то, чтобы суметь вообще как-то поработать с таким объемом данных, а потом это как-то оптимизировать. С другой стороны, теперь у меня есть готовый фреймворк для работы с такими гигантскими датасетами.

Литература и ссылки

[1] Glyn A. Holton. *Value-at-Risk*, Second Edition
https://www.value-at-risk.net

[2] Репозиторий курсовой на GitHub
https://github.com/andrey-lukyanov/Thesis_2018

[3] MongoDB
https://www.mongodb.com

[4] Studio3T
https://studio3t.com