Andrea Venuta

Software Developer, Quantitative Analyst

Personal information

Place of birth	Prato, Italy
Date of birth	September 2nd, 1989
Mobile	+39 3394971144
Web/blog	https://veeenu.github.io
E-Mail	andrea.venuta1989@gmail.com
GitHub	https://github.com/veeenu
LinkedIn	https://linkedin.com/in/andreavenuta

Education

MSc Finance & Risk Management — 110L/110
Università degli Studi di Firenze — *2018*

Informatica SS. MM. FF. NN. — 103/110
Università degli Studi di Firenze — *2015*

Diploma di Perito Informatico — 100/100
I.T.I.S. "T. Buzzi" — *2008*

Skills

Quantitative Finance

Backtesting of algorithmic trading strategies: Trend-following/mean-reverting on equities, futures relative volatility trading. Knowledge of econometrics, statistics and quantitative methods of derivatives pricing. Experienced in Bloomberg Terminal API. Experienced in data-science oriented Python (Pandas, NumPy, Jupyter). Experienced in working with large amounts of time series data (Apache Parquet, HDF5, RDBMS). Experienced in low-level algorithm optimization and memory management in C/C++.

Front-end Web Development

Strong knowledge of the Javascript ecosystem. Experienced in interactive data visualization (D3.js/WebGL/Canvas), web application design (Vue.js). Well-versed in ECMAScript6 and proficient in architecting maintainable workflows.

Back-end Web Development / DevOps

Experienced in RESTful web services development with the Java EE/Swing/Hibernate stack, Node.js, Python, .NET Core. Experienced in microservices architecture and Docker Compose-based deployment.

Graphics Programming

Real-time rendering algorithms on the OpenGL pipeline. Computational geometry, algorithms for 3D model procedural generation, representation and animation; shading algorithms.

Machine Learning

Experienced in working with neural networks for AI-based decision making and pattern recognition. Interested in deep learning and modern ML techniques.

Other

Experienced in developing desktop applications in C++/Qt, Java/JavaFX and Electron. Experienced in developing native Android apps. Knowledge of LaTeX.

Professional experience

Azimut Capital Management sgr — *10/2017 - present day*

- Developed "Suzaku", a proprietary backtesting platform in the form of a C++ CPython extension, supporting Jupyter widgets via Vue.js/Web Components. Mixed vector/event-driven architecture, allowing limit orders, margin trading, portfolio weight targeting, integrating with the Pandas/NumPy/SciPy stack.
- Implemented an automated signal generation pipeline for the AZ MA Algo Equity Strategies fund: relative vol trading of VIX ETNs replicated via futures, long-short momentum and mean reversion on index futures, with parametric diversification.

Interfase s.r.l. — *08/2012 - 02/2018*

- Developed a multi-monitor interactive data visualization installation, controlled from a terminal with a local ad-hoc network
- Developed a distributed multi-device Virtual Reality immersive video player, with playback controlled by a narrator with a remote application
- Developed a GIS tool for visually designing domain-specific scenarios, feeding the result to a mathematical model engine and reporting upon its output

GWC World — *08/2011 - 06/2013*

Standouter.com — *08/2011 - 08/2012*

Academic and personal projects

Deep Learning models for High-Frequency Cryptocurrency Forecasting — *2018*

MSc thesis. I applied a long short-term memory based recurrent deep neural network to intraday pricing data for a number of cryptocurrencies listed on the GDAX exchange. Findings showed that a simple model on 5-minute data is unable to forecast returns, but may forecast realized volatility (squared returns) with significant precision.

Procedural Content Generation and Real-time Rendering — *2015*

BSc thesis. I researched and designed a mathematical formalism by merging parametric open L-systems and shape grammars to define urban architectural elements and procedurally generate 3D models of cities. The models were textured with signal function-based, anti-aliased procedural textures and rendered in the context of a deferred, multi-pass shading renderer. The thesis can be read on my personal website.

Neural Network exam project: Gesture Recognition — *2014*

In this project I developed, along with a colleague, a sign language "typewriter" by training an artificial neural network with data sourced from a Leap Motion controller. Every time a threshold on the classification for a sign is passed, the corresponding letter is output.

JS1K competition — *2014, 2015*

I competed twice in the yearly JS1K, a competition based on writing web-based visualizations in at most 1024 bytes of Javascript code. I submitted a minimal tunnel visualization in 2014 and a Perlin noise implementation in 2015.

Goals and interests

Researching the viability of machine learning techniques, such as artificial neural networks and deep learning, in the context of trading strategy planning and automation.

Studying functional programming techniques and languages, such as Haskell and Erlang, to develop fault-tolerant, mathematically testable automated trading systems.

More on quantitative development

During my professional experience in quantitative portfolio management, being responsible for the entirety of the technological stack our fund's models were built and tested upon at Azimut, I've had the chance to work with a sizable number of different technologies; I will briefly describe my day-to-day experience with some of those, as I believe it can give a prospective employer some insight on my ways of reasoning.

Backtesting iterations: switching from a Python to a C++ backend

At first, I used to develop trading strategies using Quantopian's ZipLine event-based backtesting engine in Jupyter Notebook. Perhaps due to the combination of the event-based model and the entirety of the codebase being Python, the backtests ran unacceptably slow at scale. Scale is fundamental for any stock based analysis: any trading system on the S&P500 is bound to have either thousands of assets times tens of thousands of data points, or 500 assets times a few hundreds data points times a crippling survivorship bias. Running a backtest on a survivorship-bias-free dataset would require a complex system and anywhere between 15 minutes to 1 hour; even incremental changes to the parameters would have led to whole days wasted with little to show for it, optimizations would've been outright unthinkable of, and CPython's GIL coupled with the heavy interprocess communication required to move such large amounts of data would render both threading and multiprocessing useless approaches for the task at hand. I then decided it was worth the effort to look into building a custom backtester in the form of a CPython C++ extension with Jupyter widgets visualizations. That way, I could still keep developing the strategies in a high level language, relying upon the exceptional high level data science tooling of the Python community (Pandas, NumPy, SciPy, ...) while minimizing the execution time of the parts that mattered. My first approach relied upon the Boost::Python library and it was an iterative backtester masked as an event-driven backtester -- it relied upon lightweight accessors to shadow the actual data and simulate a feed of prices to a function to avoid look-ahead bias without actually moving or copying any data, sourced its content from NumPy arrays, and asked the user to extend a class and implement a handful of methods. The lengthy backtests sped up by more than one order of magnitude, not one of them clocking in at more than a few seconds. This proved invaluable in rapidly evaluating a number of models and exploring their parameter space to ascertain stability. As a second iteration, I decided to migrate towards pybind11, as it provided an interface much leaner than Boost::Python, improve the support to Pandas and NumPy by implementing a rudimentary "typechecking" system (i.e. tolerate either Pandas objects, or NumPy arrays, or even scalars, depending on the context; a concept similar to broadcasting, if you like) and increment the internal complexity while allowing for much simpler systems to be developed by the user. Now it is asked of the user to simply write "signal" functions, i.e. real functions of the prices and compositions of other signals, and a "money management" function that does asset allocation with the signals; all of this is implemented via function decorators, which allow for very concise systems to be written and even faster performance to be achieved as the signals are intended to be vector functions (and can be implemented in Cython!).

HDF5 as time series data storage

As I have a background in web development, I've always believed databases would be the way to go to store large amounts of structured data. While that may be the case when the requirement is networking scalability, concurrent read/writes and high availability, it is much less so when high reading speed of contiguous, homogeneous data (such as time series data) is a priority. I learned that the hard way by benchmarking SQL SELECT statements executed by Pandas against reading roughly the same amount of data sourced via plain CSV files. CSV files have another obnoxious defect, though: they need to be parsed, which is a relatively expensive operation. Enter HDF5: a relatively old, very stable, hierarchical file format designed for storing a number of large arrays on a single file. Its library, with numerous bindings (Pandas natively supports it) provides some interesting features such as lazy loading, which made opening the 10GB file I built after sourcing the pricing data from our data provider an instantaneous operation. There are alternatives out there (i.e. TeaFiles); the bottom line is that, while RDBMS aren't going to lose any momentum anytime soon, they're probably ill suited for the particular use case of permanently unchanging information such as "yesterday's VIX closing price".

My approach to quantitative trading strategies

I'm a firm believer in rationality, and focusing one's efforts largely on making sure one is not committing any obvious mistakes, and as many non-obvious mistakes as we can. There is plenty of information concerning how biases work; at a minimum, one should strive towards minimizing those, i.e. reject a stock strategy if it is ridden by survivorship bias, be wary of the data mining bias when adjusting parameters, and so on. Once one has, in good faith, done the best they can and found the models they genuinely believes the most in, the process should be executed with the utmost diligence, if not fully automated, as operational misalignments are the source of most of the tracking error. Quantitative analysis is a deceitful tool, and as such it is not to be arrogantly treated as a bulletproof solution, but its results are to be explored with humility and skepticism until it is hard to find counterevidence; and after that point, it is important to remember that we are experiencing an ever-changing financial landscape in which models can and do stop working more often than not. Being an avid reader of Nassim Nicholas Taleb even before studying finance, I soon took a liking to trading strategies based on convexity and rare events and grew to believe in the "barbell strategy" approach as the one having the most viable long term payoff on the basis of both sustainability and returns. I'm thus interested in strategies on derivatives which can supply nonlinearity, such as VIX futures (by virtue of their underlying) or equity options. Those strategies may certainly exhibit jarring features such as disproportionately high expected compounded annualized growth rate and nearly unsustainable volatility if considered as stand-alone, but would be much more manageable if included as a much smaller portion of a fixed-income portfolio, and would oftentimes provide both useful decorrelation and alpha.