

---

# Prophet Documentation

*Release 0.1.0*

**Michael Su**

May 11, 2018



<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>User Guide</b>	<b>5</b>
2.1	Quickstart . . . . .	5
2.2	Tutorial . . . . .	6
2.3	Advanced . . . . .	6
2.4	Best Practices . . . . .	7
2.5	Registry . . . . .	7
<b>3</b>	<b>API Reference</b>	<b>9</b>
3.1	API . . . . .	9
<b>4</b>	<b>Contributor Guide</b>	<b>13</b>
4.1	Getting Started . . . . .	13
<b>5</b>	<b>Additional Notes</b>	<b>15</b>
5.1	Roadmap . . . . .	15
5.2	Changelog . . . . .	15
5.3	License . . . . .	15
	<b>Python Module Index</b>	<b>17</b>



Prophet is a Python microframework for financial markets. Prophet strives to let the programmer focus on modeling financial strategies, portfolio management, and analyzing backtests. It achieves this by having few functions to learn to hit the ground running, yet being flexible enough to accomodate sophistication.

Go to [Quickstart](#) to get started or see the [Tutorial](#) for a more thorough introduction.

Prophet's data generators, order generators, and analyzers are all executed sequentially which is conducive to allowing individuals to build off of others work. If you have a package you would like to share, please open an issue on the github repository to register it in the [Registry](#).



---

### Features

---

- Flexible market backtester
- Convenient order generator
- See [Roadmap](#) for upcoming features





## Quickstart

To install Prophet, run:

```
pip install prophet
```

Here's a quick simulation with a `OrderGenerator` that uses avoids leverage and buys 100 shares of AAPL stock a day as long as it has enough cash.

```
import datetime as dt

from prophet import Prophet
from prophet.data import YahooCloseData
from prophet.analyze import default_analyzers
from prophet.orders import Orders

class OrderGenerator(object):

    def __init__(self):
        super(OrderGenerator, self).__init__()
        self._data = dict()

    def run(self, prices, timestamp, cash, **kwargs):
        # Lets buy lots of Apple!
        symbol = "AAPL"
        orders = Orders()
        if (prices.loc[timestamp, symbol] * 100) < cash:
            orders.add_order(symbol, 100)

        return orders

prophet = Prophet()
prophet.set_universe(["AAPL", "XOM"])
prophet.register_data_generators(YahooCloseData())
prophet.set_order_generator(OrderGenerator())
prophet.register_portfolio_analyzers(default_analyzers)

backtest = prophet.run_backtest(start=dt.datetime(2010, 1, 1))
analysis = prophet.analyze_backtest(backtest)
print analysis
```

```
# +-----+
# | sharpe           | 1.09754359611 |
# | average_return   | 0.00105478425027 |
# | cumulative_return | 2.168833 |
# | volatility       | 0.0152560508189 |
# +-----+

# Generate orders for you to execute today
# Using Nov, 10 2014 as the date because there might be no data for today's
# date (Market might not be open) and we don't want examples to fail.
today = dt.datetime(2014, 11, 10)
print prophet.generate_orders(today)
# Orders[Order(symbol='AAPL', shares=100)]
```

## Tutorial

### Introduction

You can get the full source code of the tutorial [here](#)

The tutorial is based off of the last homework in QSTK. There are some differences that will be explained. This tutorial is currently unfinished so please read the source code linked above. Hopefully I'll get it done tomorrow.

### Data Generation

First you need to initialize the object and setup the stock universe:

```
prophet = Prophet()
prophet.set_universe(["AAPL", "XOM"])
```

Then you register any data generators. Please see the source code of `prophet.data` for an example of a data generator. Data generators don't have to just pull raw data though like `prophet.data.YahooCloseData` does. For instance, you can generate correlation data based off the price data. Prophet encourages you to

```
# Registering data generators
prophet.register_data_generators(YahooCloseData())
prophet.set_order_generator(OrderGenerator())
prophet.register_portfolio_analyzers(default_analyzers)
```

### Order Generation

### Portfolio Analysis

## Advanced

### Slippage & Commissions

The `run_backtest` method on the `Prophet` object contains a commission and slippage option for you to make the backtest more realistic. Slippage is how much of the price increases (when buying) or decreases (when selling) from the price data. Commission represents the fees you pay per trade.

Please open an issue if those parameters aren't sufficient for your needs. See the [API](#) for more details.

## Best Practices

Try to keep as much of your code as possible in the pandas (or numpy) space. Lots of smart folk have spent a considerable amount of time optimizing those libraries. Since most of the code in pandas and numpy is executed in C, it will be much more performant.

For the data generators, please pass around pandas Dataframes as much as possible. (Which then means that your order generator will have to operate on pandas Dataframes)

## Registry

**There are currently no packages in the registry**

If you have a package you would like to share, please open an issue on the github repository to register it in the *Registry*.



---

## API Reference

---

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

### API

This part of the documentation covers all the interfaces of Prophet. For parts where Flask depends on external libraries, we document the most important right here and provide links to the canonical documentation.

### Prophet Object

**class** `prophet.Prophet`

The application object. Serves as the primary interface for using the Prophet library.

**config**

*dict*

Dictionary of settings to make available to other functions. Useful for things like `RISK_FREE_RATE`.

**analyze\_backtest** (*backtest*)

Analyzes a backtest with the registered portfolio analyzers.

**Parameters** `backtest` (*prophet.backtest.BackTest*) – a backtest object

**Returns** `prophet.backtest.BackTest`

**generate\_orders** (*target\_datetime*, *lookback=0*, *cash=1000000*, *buffer\_days=0*, *portfolio=Portfolio()*)

Generates orders for a given day. Useful for generating trade orders for a your personal account.

**Parameters**

- **target\_datetime** (*datetime*) – The datetime you want to generate orders for.
- **lookback** (*int*) – Number of trading days you want data for before the (*target\_datetime* - *buffer\_days*)
- **cash** (*int*) – Amount of starting cash
- **buffer\_days** (*int*) – number of trading days you want extra data generated for. Acts as a data start date.
- **portfolio** (*prophet.portfolio.Portfolio*) – Starting portfolio

**register\_data\_generators** (\**functions*)

Registers functions that generate data to be assessed in the order generator.

**Parameters** *functions* (*list*) – List of functions.

**register\_portfolio\_analyzers** (*functions*)

Registers a list of functions that are sequentially executed to generate data. This list is appended to list of existing data generators.

**Parameters** *functions* (*list of function*) – Each function in the list of args is executed in sequential order.

**run\_backtest** (*start*, *end=None*, *lookback=0*, *slippage=0.0*, *commission=0.0*, *cash=1000000*, *initial\_portfolio=Portfolio()*)

Runs a backtest over a given time period.

**Parameters**

- **start** (*datetime*) – The start of the backtest window
- **end** (*datetime*) – The end of the backtest windows
- **lookback** (*int*) – Number of trading days you want data for before the start date
- **slippage** (*float*) – Percent price slippage when executing order
- **commission** (*float*) – Amount of commission paid per order
- **cash** (*int*) – Amount of starting cash
- **portfolio** (*prophet.portfolio.Portfolio*) – Starting portfolio

**Returns** `pandas.backtest.BackTest`

**set\_order\_generator** (*order\_generator*)

Sets the order generator for backtests.

**Parameters** *order\_generator* – Instance of class with a run method that generates

**set\_universe** (*symbols*)

Sets the list of all tickers symbols that will be used.

**Parameters** *symbols* (*list of str*) – Ticker symbols to be used by Prophet.

## Order Objects

**class** `prophet.orders.Order`

`Order(symbol, shares)`

**class** `prophet.orders.Orders` (\**args*)

Orders object that an OrderGenerator should return.

**add\_order** (*symbol*, *shares*)

Add an order to the orders list.

**Parameters**

- **symbol** (*str*) – Stock symbol to purchase
- **shares** (*int*) – Number of shares to purchase. Can be negative.

## Backtest Object

`prophet.backtest.BackTest`

## Portfolio Objects

`class prophet.portfolio.Portfolio`

Portfolio object where keys are stock symbols and values are share counts. You can pass these into a backtest to start with an initial basket of stocks.

---

**Note:** Subclasses dict in v0.1

---

## Analyzer Objects

`class prophet.analyze.Analyzer`

`class prophet.analyze.Volatility`

`class prophet.analyze.AverageReturn`

`class prophet.analyze.Sharpe`

`class prophet.analyze.CumulativeReturn`

`prophet.analyze.default_analyzers = [volatility, average_return, sharpe, cumulative_return]`

`list()` -> new empty list `list(iterable)` -> new list initialized from iterable's items

## Data Objects

`class prophet.data.DataGenerator (cache_path=None, data_path=None)`

`class prophet.data.PandasDataGenerator (cache_path=None, data_path=None)`

`class prophet.data.YahooCloseData (cache_path=None, data_path=None)`

`class prophet.data.YahooVolumeData (cache_path=None, data_path=None)`





---

## Contributor Guide

---

### Getting Started

Setup your dev environment with the following commands.

```
git clone git@github.com:Emsu/prophet.git
cd prophet
virtualenv env
pip install dev-requirements.txt
python setup.py develop
```

If you want to help with longer term development, please open an issue [here](#)



---

## **Additional Notes**

---

### **Roadmap**

#### **v0.2**

- Ability to handle more frequent timeseries data
- Stress Tester and scenario builder
- More data sources

### **Changelog**

#### **Version 0.1.1**

- Added Python 3 support

#### **Version 0.1**

- First public release

### **License**

Copyright (c) 2014, Michael Su All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## p

- `prophet`, [9](#)
- `prophet.analyze`, [11](#)
- `prophet.backtest`, [11](#)
- `prophet.data`, [11](#)
- `prophet.orders`, [10](#)
- `prophet.portfolio`, [11](#)



## A

`add_order()` (`prophet.orders.Orders` method), 10  
`analyze_backtest()` (`prophet.Prophet` method), 9  
`Analyzer` (class in `prophet.analyze`), 11  
`AverageReturn` (class in `prophet.analyze`), 11

## B

`BackTest` (in module `prophet.backtest`), 11

## C

`config` (`prophet.Prophet` attribute), 9  
`CumulativeReturn` (class in `prophet.analyze`), 11

## D

`DataGenerator` (class in `prophet.data`), 11  
`default_analyzers` (in module `prophet.analyze`), 11

## G

`generate_orders()` (`prophet.Prophet` method), 9

## O

`Order` (class in `prophet.orders`), 10  
`Orders` (class in `prophet.orders`), 10

## P

`PandasDataGenerator` (class in `prophet.data`), 11  
`Portfolio` (class in `prophet.portfolio`), 11  
`Prophet` (class in `prophet`), 9  
`prophet` (module), 9  
`prophet.analyze` (module), 11  
`prophet.backtest` (module), 11  
`prophet.data` (module), 11  
`prophet.orders` (module), 10  
`prophet.portfolio` (module), 11

## R

`register_data_generators()` (`prophet.Prophet` method), 9  
`register_portfolio_analyzers()` (`prophet.Prophet` method),

10

`run_backtest()` (`prophet.Prophet` method), 10

## S

`set_order_generator()` (`prophet.Prophet` method), 10  
`set_universe()` (`prophet.Prophet` method), 10  
`Sharpe` (class in `prophet.analyze`), 11

## V

`Volatility` (class in `prophet.analyze`), 11

## Y

`YahooCloseData` (class in `prophet.data`), 11  
`YahooVolumeData` (class in `prophet.data`), 11