ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

How to Develop a Reinforcement Learning Trading System

Learning Objectives

- Identify the components of an RL trading system
- Understand the steps required to develop a trading strategy using deep reinforcement learning strategies
- Identify the final strategy checks required to go live with an RL trading system

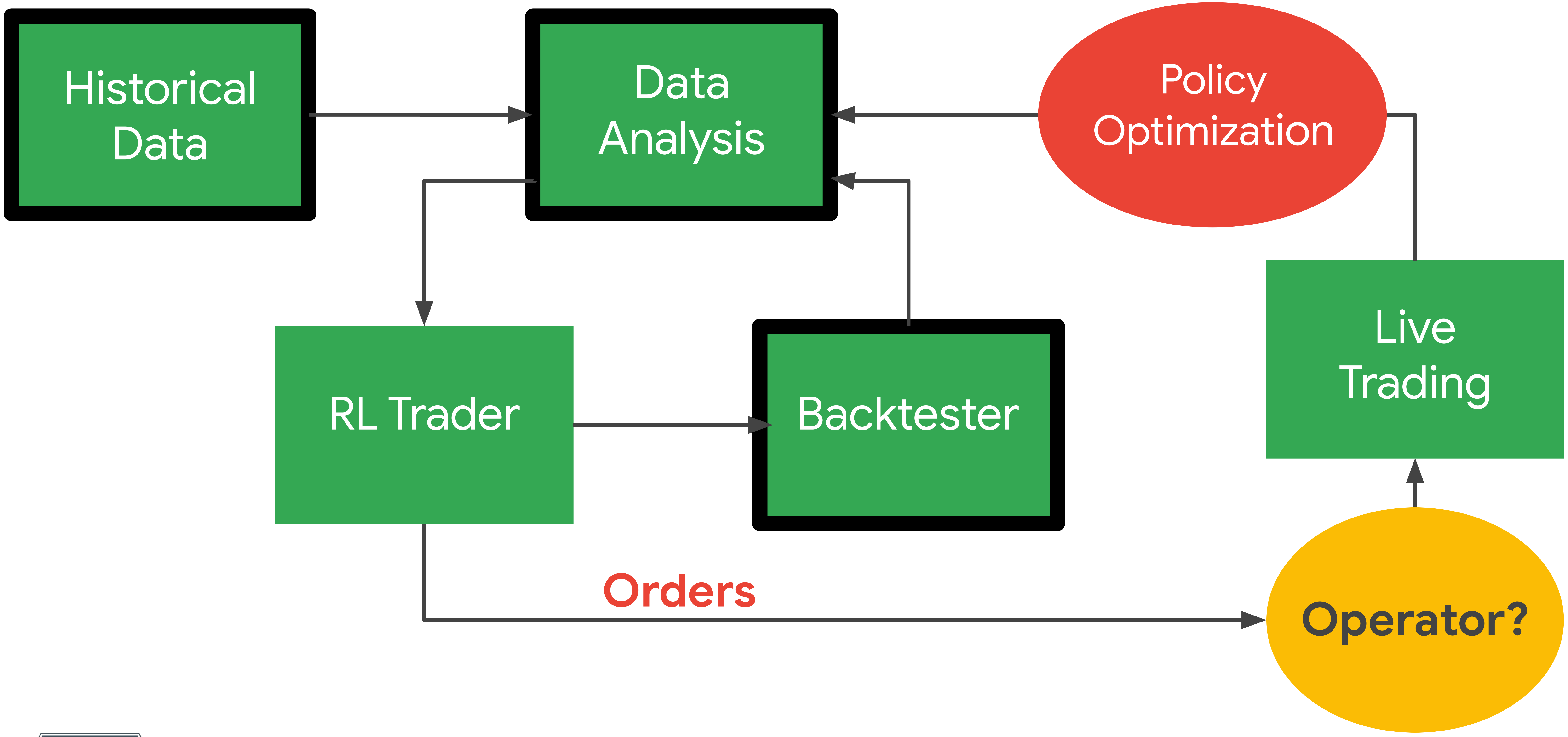
Agenda

Components of a Reinforcement Learning Trading System

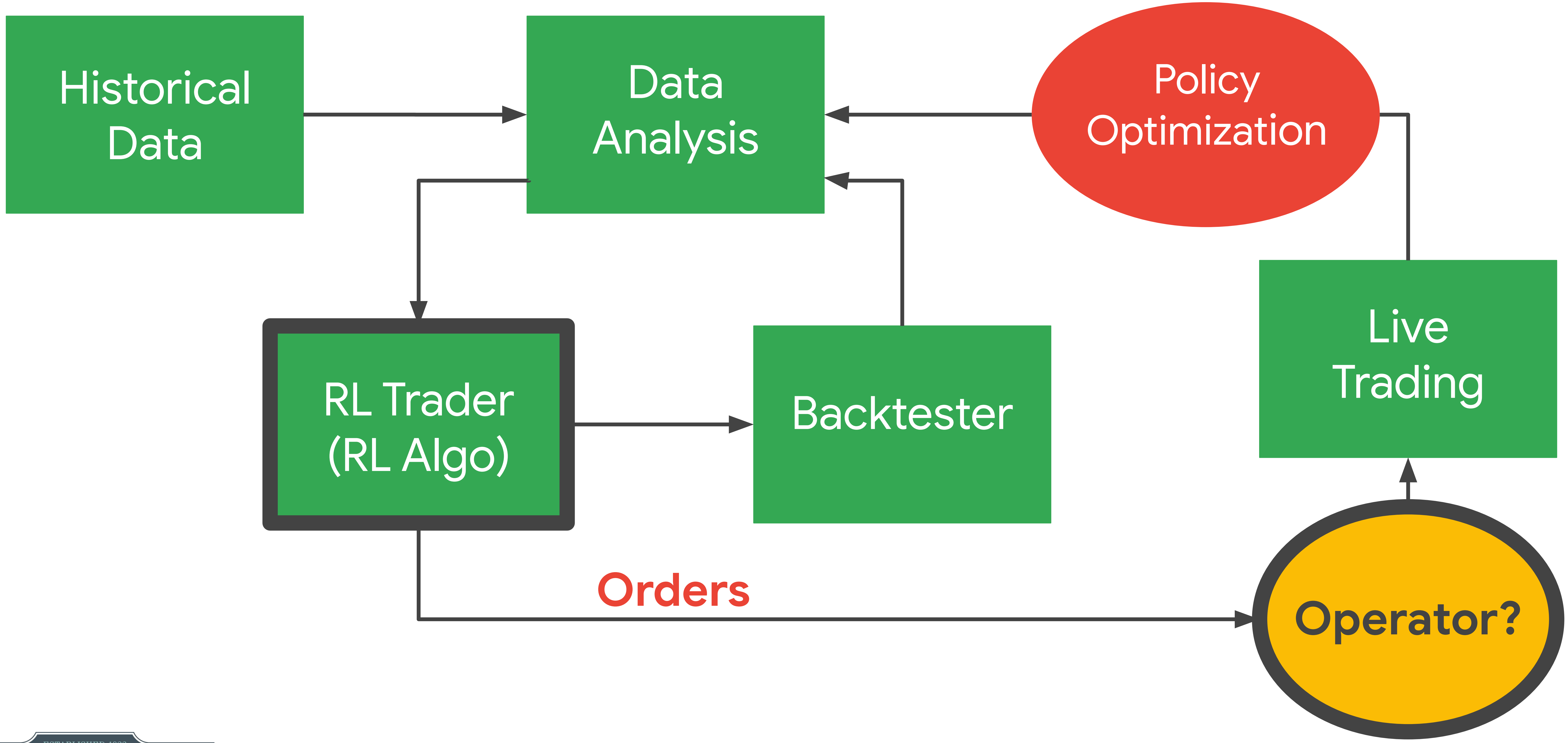
Steps Required to Develop a Deep Reinforcement Learning Strategy

Final Checks Before Going Live with Your Strategy

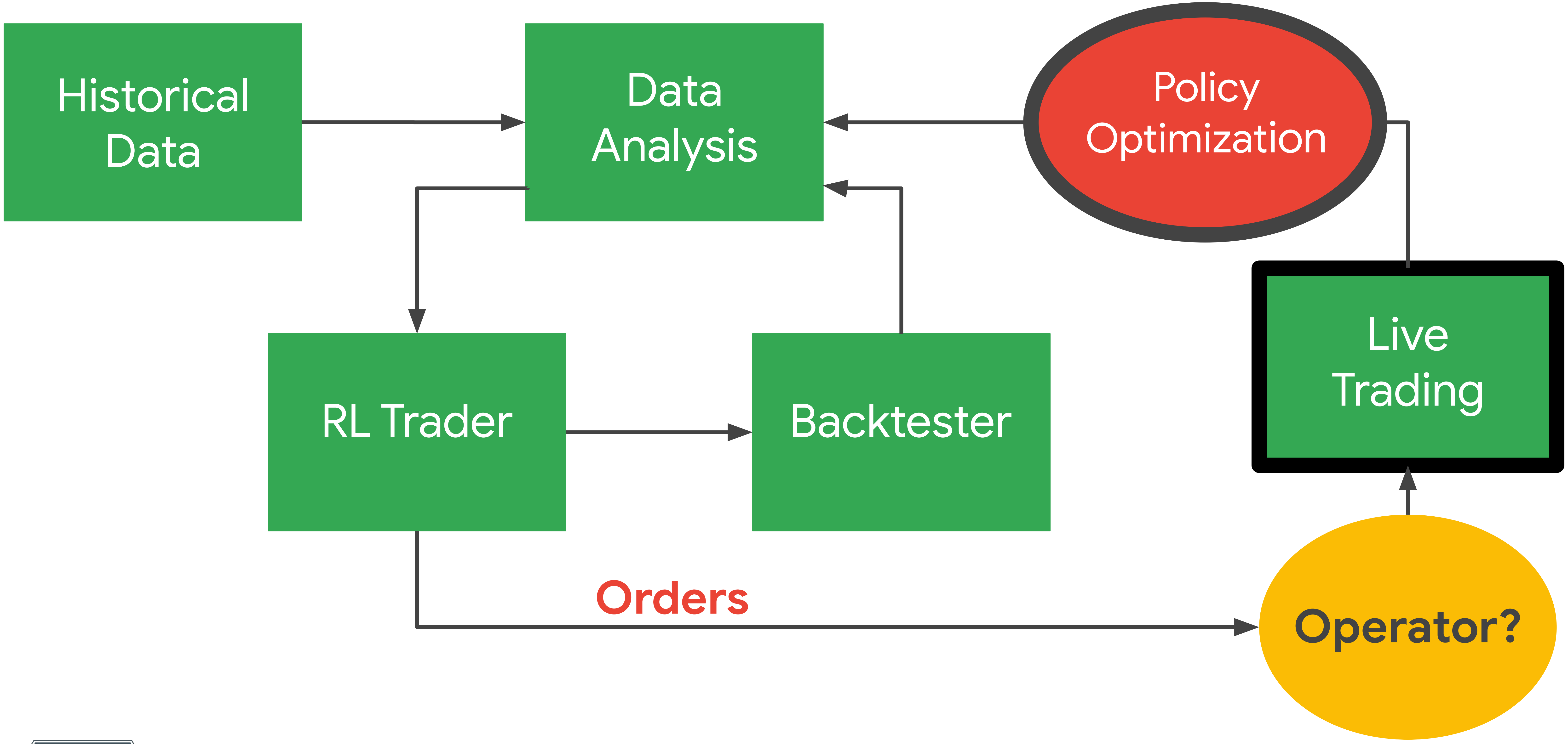
RL Trading System



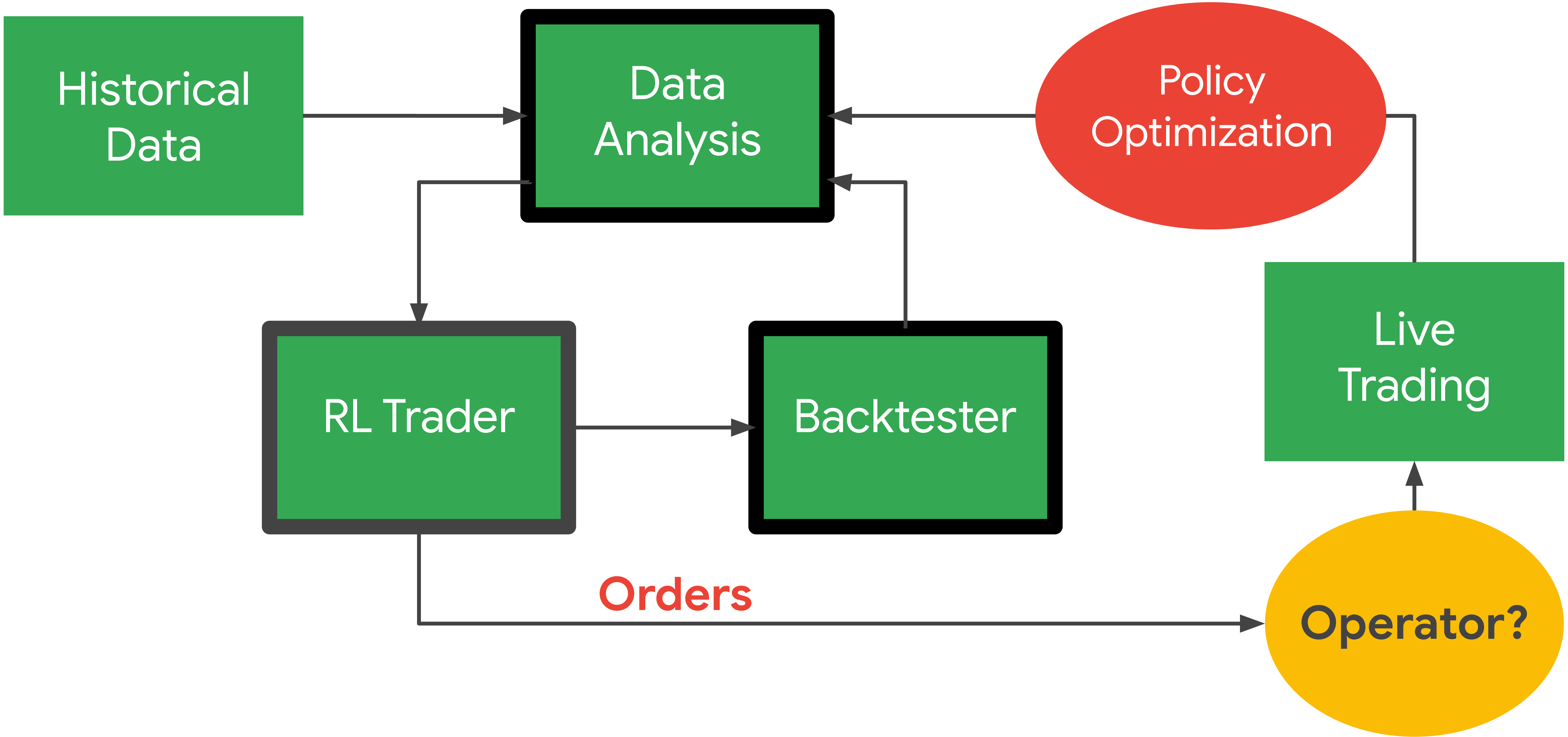
RL Trading System



RL Trading System



RL Trading System



Agenda

Components of a Reinforcement Learning Trading System

Steps Required to Develop a Deep Reinforcement Learning Strategy

Final Checks Before Going Live with Your Strategy

DRL Algorithm Development

1. Choose your instrument(s)
2. Model trading costs and other potential drags on performance
3. Obtain and cleanse a sufficient amount of historical data
4. Create an ensemble of algos and experiment with different inputs
5. Define your action space and decide on a training method
6. Train and backtest your RL Trader
7. Go live or retrain your RL Trader

Choose the Instrument You Will Trade

- Make sure sufficient data is available
- Low-volume / low-liquidity markets can be more profitable
- High volatility instruments such as Crypto can also have higher performance potential
- Need to be sure you have the risk appetite and budget needed to trade effectively

Selection also depends on your risk appetite. Crypto is high risk, equities, less so, and FX, even less. Also remember that risk is not just dependent on the instrument or market you are trading. It can be increased or decreased dramatically by the amount of leverage you add to your trading capital. Low volume, low liquidity markets can be more profitable, but you need to have the risk capacity and appetite to trade them.

Model Trading Costs

- Slippage is the difference between the expected price of a trade and the price at which it is actually executed
- Slippage is highest during periods of high volatility when market orders are used
- You need bid-ask price data at a minimum to model slippage in backtesting (visibility into the order book is better though)

You also need to understand the effect your trades have on the market you are trading. This is where you need to model market impact costs such as slippage, which is the difference between the expected price of the trade and the actual price at which the trade is executed.

Model Trading Costs

- Slippage is the difference between the expected price of a trade and the price at which it is actually executed
- Slippage is highest during periods of high volatility when market orders are used
- You need bid-ask price data at a minimum to model slippage in backtesting (visibility into the order book is better though)

Slippage can occur at any time, but is most prevalent and severe during periods of higher volatility, particularly when market orders are used. You can eliminate slippage by using limit orders with a fixed price, but you run the risk of missing a trading opportunity if your order is not filled.

Model Trading Costs

- Slippage is the difference between the expected price of a trade and the price at which it is actually executed
- Slippage is highest during periods of high volatility when market orders are used
- You need bid-ask price data at a minimum to model slippage in backtesting (visibility into the order book is better though)

You need to make sure you calculate slippage in your backtesting. This can only be done if you have bid-ask price data or better yet, visibility into the order book of the security you are backtesting. This will allow you to model market impact costs more realistically. If you do not calculate these cost, your algo may look profitable in testing but will disappoint when deployed in live trading. If this data is not obtainable, you can infer or estimate slippage from other indicators such as trade size, volume, and volatility.

Collect/Cleanse Data

- Training an RL algo is very sensitive to gaps or outliers in the data
- Need to interpolate missing values and “correct” outliers
- If data gaps are too big, remove the data from the training set

Each time you get access to raw financial data, you have an asset whose potential value can only be unlocked if it is cleaned up. It will take some time, but it is worth the effort. Investing the time needed for this tedious job is important and require strict attention to detail. Almost every raw dataset has missing values.

How do you handle the missing values? Simpler is better. Use interpolated means for filling in small amounts of missing data, but remove the data completely if the proportion of missing data is too high for a certain period.

Ensemble of Algorithms

- You need a variety of algos as each will perform differently in different trading environments
- Give algos that perform better a higher weight for trading decisions
- Keep backtesting on new data and re-weight algos immediately

You should create many algos that behave differently in different time frames and trading environments. Then use the ones that behave best in the most recent few months to trade live. Or if you're using an ensemble of algos, give higher weights to those that have behaved better in the recent past when making the decision whether to trade or not. Keep on backtesting as new data come in and if the behavior of the market and/or the performance of an algo changes, switch or reweight the algos immediately.

Use Different Inputs

- Can you use both price and sentiment data to train your RL algorithm?
- Better to use separate neural networks for each type of data and feed their representations to your RL Trader
- Stepwise structure with different NNs at each step works best

Can you blend numerical data with classification data? For example, numerical values with sentiment over the news. You can but it works much better if you separate neural networks for each of these tasks and then use the representations of these neural networks to feed your RL trader.
You can think of your neural networks like the intellect of a small child. It needs to be guided and taught and given clear examples without blending lots of stuff together and confusing the inputs. Splitting similar data into different neural networks will improve your algo's performance immediately. Instead of using one neural network to master all of your data, think of the whole process and break it into steps. Then build a separate neural network for each step.

RL Algorithm

- Responsible for taking the inputs from the deep recurrent neural network (DRNN) and making a decision
- Buy, Sell, or Hold
- Needs to learn actions based on analysis of a continuous data feed (like a human trader)
- Many RL methods but literature and experience support actor-based methods

This part of our RL trading system is responsible for taking the inputs from the deep recurrent neural network or DRNN and making a decision. This is the command center of our RL trader.
What kind of decisions should it be able to make? It will choose among three possible decisions; buy, sell, or hold. There are also many ways to train your RL algo. You need one that can take near continuous values, not just snapshots of reality. It will need to learn actions based on data analysis in a near continuous feed of sensory data just like a trader in front of a terminal. In my experience and also in the literature, the actor base reinforcement learning methods tend to work better.

Training Your RL Trader

- Present the RL Trader with a window of data that you have normalized
- Allow it to experiment with different actionable scenarios
- Calculate PnL for initial training then add Sharpe ratios for each scenario
- Your RL Trader will eventually develop a trading style that will maximize PnL and/or Sharpe ratio

Now that you have designed your system and gathered your data, you are ready to train your RL trader or agent. We need to present the RL trader with a window of data and let it experiment with different actionable scenarios while calculating PnL and also Sharpe ratio on later stages in every case scenario.
You need to punish it when it does badly and reward it if it does well. This will happen in many iterations, and over time, the RL trader will learn to trade in a style that maximizes this reward.

Backtest Your RL Trader

- Use more recent out-of-sample and unseen data set
- Data set should include a range of market situations (uptrending, downtrending, volatile, steady)
- If RL Trader maintains or improves performance then you are ready to go live or paper trade

Backtesting in RL trader is critical and needs to happen on out-of-sample data, which the RL trader has never seen, either in training or during testing. Only if the algorithm performs well on this kind of data should you even consider testing it in a live trading environment. Keep a good amount of recent data for backtesting. Data that your algo has not been training or testing on and make sure that you have all of the representative market situations covered such as volatile markets, steady, down trending, up trending, etc. Also, make sure that you are representing the market accurately. For example, splits of stocks, spin-offs, companies that were acquired or failed, so you don't have survivor bias.

Agenda

Components of a Reinforcement Learning Trading System

Steps Required to Develop a Deep Reinforcement Learning Strategy

Final Checks Before Going Live with Your Strategy

Transition to Live Trading

- Don’t be hasty
- It took you a lot of time and effort to get to this stage
- You have already spent heavily in time and costs
- The rewards can be substantial if your trader is successful and keeps its edge

Now you are ready to make the final checks before allowing your RL trader to trade a live market. You spent a lot of time and experimented with different RL models. You've managed to achieve a good sharp ratio for your RL trader after lots of trial and error. How do you know if your RL trader is ready to go live? What are the indications of a well-trained RL trader? Don't be in a hurry. It's going to take you a while to create a profitable algorithm trader, and you'll never know if it works until you test it with real money.

In a medium size trading firm, it takes about a $100,000 to $250,000 to develop and test several strategies until you get one that works out. But then the return can be 10 to 30 times the development costs depending on how long you're able to keep your trading edge.

Go-Live Checklist

- Your RL Trader seems to perform well for a long period in the past and in different market conditions
- Your RL Trader seems to perform well on another security that you haven't trained it on
- Your algorithm does get decent performance in the test set and has done well in training and development sets

Next are a few of the key criteria that must be satisfied before you can even think of deploying.
- First, your RL Trader seems to perform well over a long historical period and under different market conditions. This shows the trader has versatility and can be profitable under a variety of conditions.
- Second, your trader also performs well with another security that you haven't trained it on at the same time series timestep. This is a very positive signal. Timesteps are the number of lagged observations that you use to predict your next value in a time series.
- Third, your algorithm gets decent performance in the test set and has done well in the training and development sets.

Go-Live Checklist

- If your algo gets destroyed in the testing data set it means that it has been overfitted and you need to either:
  - Bring in more data
  - Look for less deep neural network architectures
  - Increase your dropout percentage

If your algo makes big losses in the test set, it means that it has been over-fitted and that you need to either bring in more data, and/or look for a less deep neural network architecture or you need to increase your dropout percentage.
Recall that dropout is a technique where randomly selected neurons are ignored during training. They are dropped out randomly. This means that their contribution to the activation of downstream neurons is temporarily removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. With a higher dropout percentage, it is less likely that your algo has been over-fitted to the training data.

Dropout Rate

- As a neural network learns, neuron weights settle into their context within the network
- If neurons are dropped this results in the network learning multiple independent representations
- The network becomes less sensitive to the specific weights of neurons and is better able to generalize and not overfit

As your neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing for some specialization. Neighboring neurons come to rely on this specialization, which if taken too far can result in a fragile model that is to specialized to the training data.

This reliance on context for a neuron during training is referred to as complex co-adaptations. You can imagine that if neurons are randomly dropped out of the network during training, that other neurons will have to step in and handle the representations required to make predictions for the missing neurons. This results in multiple independent internal representations being learned by the network. The effect of this is that the network becomes less sensitive to specific weights of the neurons. This in turn results in a network that is capable of better generalization and is less likely to over fit the training data.

Go-Live Checklist

- Your algorithm performs very badly in the training set and cannot be trained
  - Use more inputs for your NN
  - Increase depth of NN
  - Adjust hyperparameters
  - Make sure you initialize the weights before training

If your algorithm performs very badly in the training set and cannot be trained. You should use more inputs for your neural network, or increase the depth of your neural network, or adjust the hyper parameters of the neural network and RL Agent. Also make sure that you have initialized the weights for the deep learning part before starting the training.

Go-Live Checklist

- Your algorithm performs well in all train and test sets and you want to try live data?
- You need to keep on training with smaller learning rates for your deep neural network. There is more potential performance to squeeze out of it before live trading

Your algorithm performs well in all train and test sets and you are ready to try live trading, but you need to keep on training despite having smaller learning rates for your deep neural network. This incremental learning, despite happening at a lower rate, will allow you to squeeze some more dollars of performance out of your algo before live trading.

Go-Live Checklist

- Your live environment is extremely volatile
- Use a demo account to paper trade for a while and see what happens
- Try live trading but make micro trades until you gain confidence in the strategy

You need to scale the implementation of your algo to reflect market conditions. You should not deploy immediately into a live environment with super high margins and extremely volatile conditions. Instead, you start with a demo account and trade in demo for a while to see what happens. Alternatively, you can trade live, but use Microtrace to gain some confidence, and see where you might need to intervene.

ESTABLISHED 1922

NEW YORK INSTITUTE OF FINANCE

Investment and Trading Risk Management

Learning Objectives

- Distinguish between risk management in trading vs investment portfolios

- Understand how diversification is achieved in a trading portfolio

- Identify optimization criteria for managing strategy risk and portfolio risk

Agenda

Investment Risk Management

Trading Strategy Risk Management

Trading Portfolio Risk Management

Maximizing Risk-Adjusted Return

- Identify undervalued assets
- Reduce risk by combining assets in a portfolio

Asset Price Volatility: Systematic Risk

- Systematic risk influences the value of financial assets in general
- A significant change in the performance of the economy would affect most of the assets in an investor's portfolio

Asset Price Volatility: Systematic Risk

This risk is generally measured by:

- Beta
- Other indicators of an investment's correlation with the overall economy
- Measures of the volatility of market indexes such as the VIX

Asset Price Volatility: Systematic Risk

- This kind of risk affects a single asset
- An example is news that affects a specific stock such as a corporate credit rating downgrade
- Diversification is the most effective strategy for investors to limit their exposure to unsystematic risk
- This risk is generally measured by an estimate of the volatility of a specific asset

Diversification and Risk

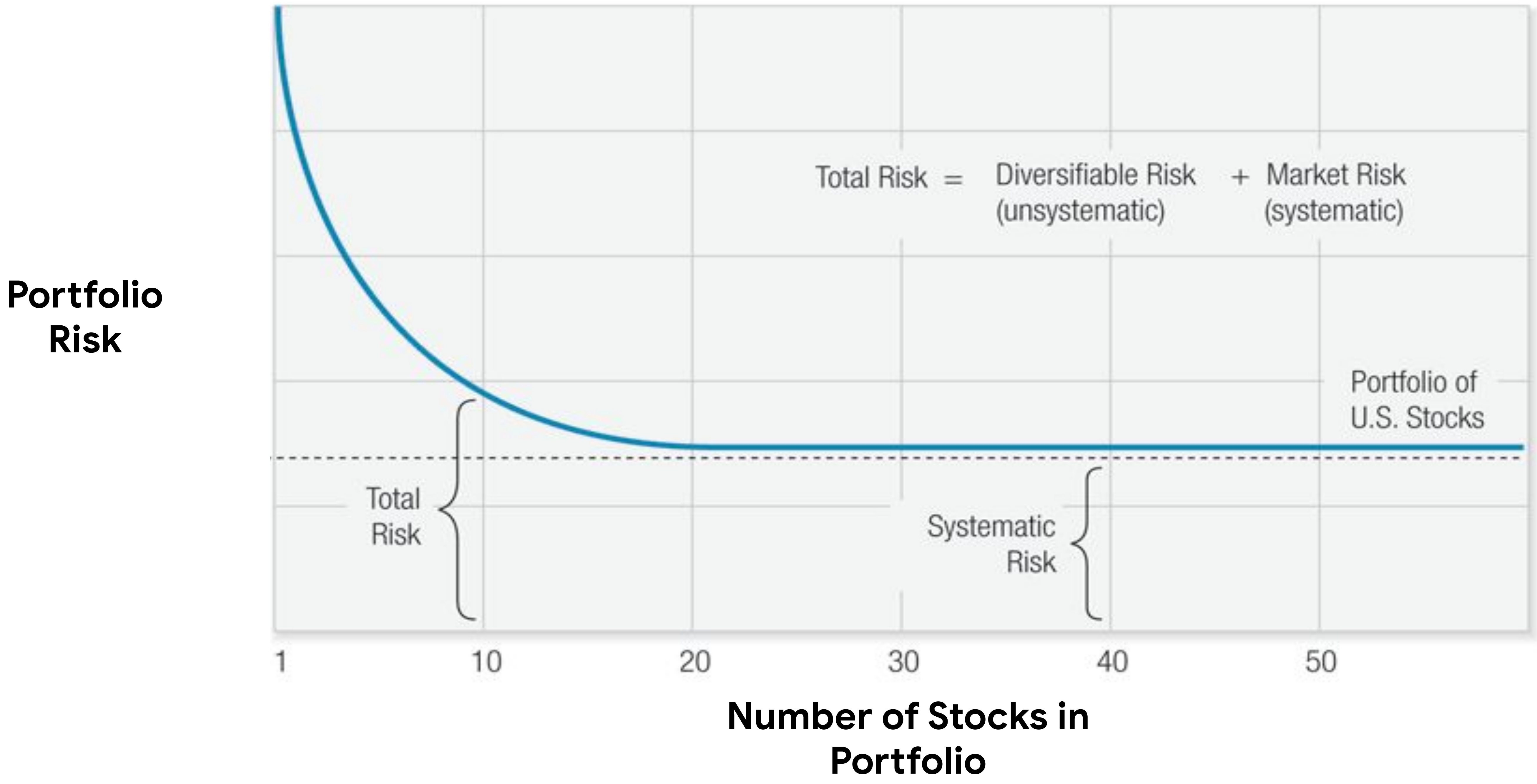
- The diversification effect on portfolios can be decomposed into two components:
  - Risk reduction as a result of holding imperfectly correlated securities
  - Risk reduction as number of securities in a portfolio increases

Diversification and Risk

- The risk of a portfolio is measured by the ratio of the covariance of a portfolio's return to the variance of the market return (portfolio beta)

- A domestic portfolio that is fully diversified would have a beta of 1

Portfolio Risk Reduction Through Diversification



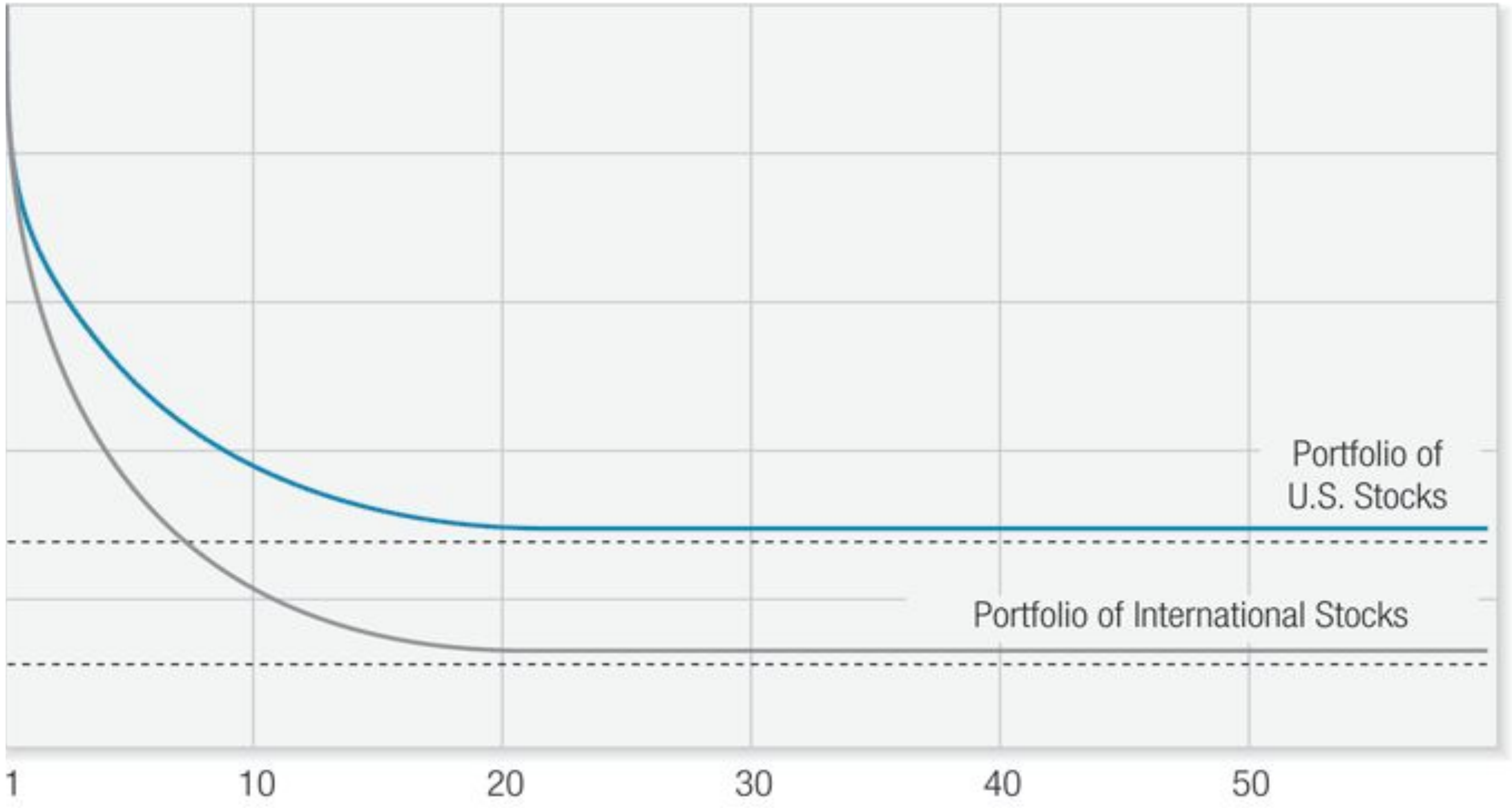
International Diversification and Risk

- The total risk of any portfolio is therefore composed of systematic risk and unsystematic risk

- Increasing the number of securities in the portfolio reduces the unsystematic risk component leaving the systematic risk component unchanged

Risk Reduction with International Diversification

Portfolio Risk



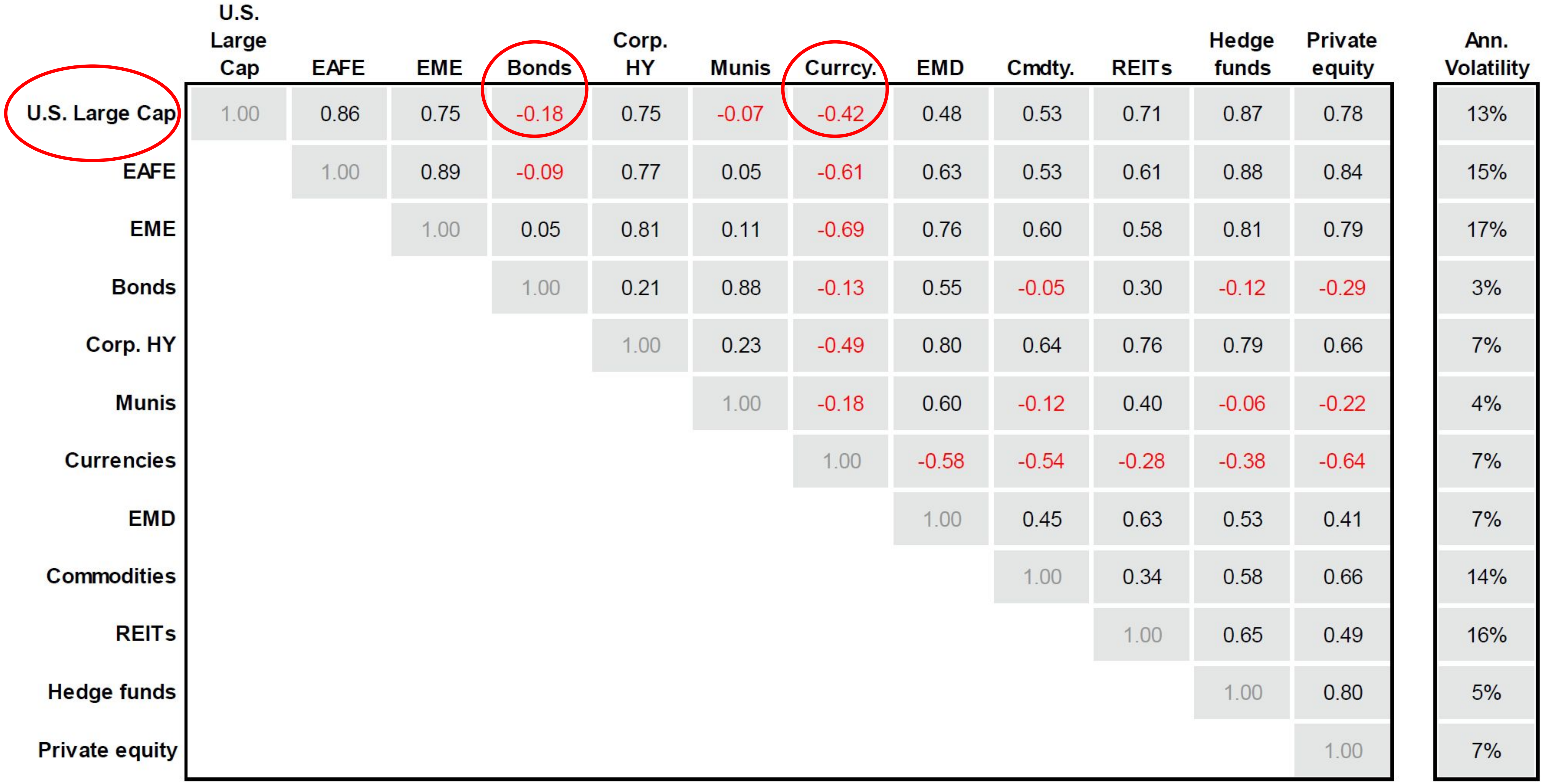
Number of Stocks in Portfolio

Market Correlations and Volatility

	U.S. Large Cap	EAFE	EME	Bonds	Corp. HY	Munis	Currcy.	EMD	Cmdty.	REITs	Hedge funds	Private equity	Ann. Volatility
U.S. Large Cap	1.00	0.86	0.75	-0.18	0.75	-0.07	-0.42	0.48	0.53	0.71	0.87	0.78	13%
EAFE		1.00	0.89	-0.09	0.77	0.05	-0.61	0.63	0.53	0.61	0.88	0.84	15%
EME			1.00	0.05	0.81	0.11	-0.69	0.76	0.60	0.58	0.81	0.79	17%
Bonds				1.00	0.21	0.88	-0.13	0.55	-0.05	0.30	-0.12	-0.29	3%
Corp. HY					1.00	0.23	-0.49	0.80	0.64	0.76	0.79	0.66	7%
Munis						1.00	-0.18	0.60	-0.12	0.40	-0.06	-0.22	4%
Currencies							1.00	-0.58	-0.54	-0.28	-0.38	-0.64	7%
EMD								1.00	0.45	0.63	0.53	0.41	7%
Commodities									1.00	0.34	0.58	0.66	14%
REITs										1.00	0.65	0.49	16%
Hedge funds											1.00	0.80	5%
Private equity												1.00	7%

Source: Barclays Inc., Bloomberg, Cambridge Associates, Credit Suisse/Tremont, FactSet, Federal Reserve, MSCI, Standard & Poor's, J.P. Morgan Asset Management.

Market Correlations and Volatility



	U.S. Large Cap	EAFE	EME	Bonds	Corp. HY	Munis	Currcy.	EMD	Cmdty.	REITs	Hedge funds	Private equity	Ann. Volatility
U.S. Large Cap	1.00	0.86	0.75	-0.18	0.75	-0.07	-0.42	0.48	0.53	0.71	0.87	0.78	13%
EAFE		1.00	0.89	-0.09	0.77	0.05	-0.61	0.63	0.53	0.61	0.88	0.84	15%
EME			1.00	0.05	0.81	0.11	-0.69	0.76	0.60	0.58	0.81	0.79	17%
Bonds				1.00	0.21	0.88	-0.13	0.55	-0.05	0.30	-0.12	-0.29	3%
Corp. HY					1.00	0.23	-0.49	0.80	0.64	0.76	0.79	0.66	7%
Munis						1.00	-0.18	0.60	-0.12	0.40	-0.06	-0.22	4%
Currencies							1.00	-0.58	-0.54	-0.28	-0.38	-0.64	7%
EMD								1.00	0.45	0.63	0.53	0.41	7%
Commodities									1.00	0.34	0.58	0.66	14%
REITs										1.00	0.65	0.49	16%
Hedge funds											1.00	0.80	5%
Private equity												1.00	7%

Source: Barclays Inc., Bloomberg, Cambridge Associates, Credit Suisse/Tremont, FactSet, Federal Reserve, MSCI, Standard & Poor's, J.P. Morgan Asset Management.

International Diversification and Risk

- The FX risks of a portfolio are reduced through international diversification
- FX as an asset category also provides diversification benefits

Investment Diversification and Risk

- Most asset managers are constrained by their mandate to invest in a single asset category.

- Diversification with other asset categories is usually accomplished by the investor

Agenda

Investment Risk Management

Trading Strategy Risk Management

Trading Portfolio Risk Management

Trading Capital and Risk Management

- Zero or minimal operator intervention means risk strategy design is key
- Risk management usually handled at strategy level and portfolio level

Strategy Level Risk Management

- Stop Losses
  - Static
  - Dynamic
  - Variable
- Set a maximum loss or risk on each trade or strategy

Strategy Level Risk Management

- Return metrics
  - Win/Loss %
  - Average Win
  - Average Loss
- Risk-to-Reward Ratio set by your RL Algorithm to maximize strategy return

Strategy Level Risk Management

- RL Algorithm will optimize:
  - Profit target
  - Stop loss level(s)
  - Time outs
- Too high a profit target or loose stop loss levels can dramatically lengthen the average trade completion time

Strategy Level Risk Management

- Long trade completion time:
  - Increases capital needs
  - Reduces number of trades
  - Reduces period returns
- RL reward design must incorporate all of these factors

Portfolio Level Risk Management

- Global Stop Losses:
  - Strategy or firm level risk limits
  - Usually a fixed percentage of trading capital
  - Closes out all open trades
- Key factor in survival of a trading firm

Portfolio Level Risk Management

- Strategy Diversification
  - Multiple strategies
  - Multiple asset classes
  - Multiple assets within each asset class

Portfolio Level Risk Management

- Pairs Trading Diversification
  - Pairs chosen using PCA analysis, fundamental analysis, and momentum
  - Multiple pairs chosen for each strategy
  - Strategies applied to different asset classes (Stocks, FX, Crypto...)

Agenda

Investment Risk Management

Trading Strategy Risk Management

Trading Portfolio Risk Management

Portfolio Level Risk Management

- Measuring risk reduction in a portfolio of trading assets
  - Level of correlation between trading assets
  - Number of trading assets held in portfolio

Notebook: Measuring Risk Reduction in a Portfolio*

From Principles: Life and Work by Ray Dalio

“From my earlier failures, I knew that no matter how confident I was in making any one bet I could still be wrong—and that proper diversification was the key to reducing risks without reducing returns. If I could build a portfolio filled with high-quality return streams that were properly diversified (they zigged and zagged in ways that balanced each other out), I could offer clients an overall portfolio return much more consistent and reliable than what they could get elsewhere.”

*Based on https://lambdaclass.com/finance_playground/diversification-dalio-holy-grail.html

Portfolio Risk Reduction

In [1]:

```
%config InlineBackend.figure_format = "retina"

import numpy as np
import pandas as pd
import altair as alt

np.random.seed(42) # Set seed for reproducibility
```

Portfolio Risk Reduction

```
In [2]:
def correlated_streams(n, mean, risk, corr):
   """Generates `n` return streams with given average `mean` and
`risk`,
   and with an average correlation level `corr`.
   """
   num_samples = 10_000
   means = np.full(n, mean)

   corr_mat = np.full((n, n), corr, dtype=np.dtype("d"))
   np.fill_diagonal(corr_mat, 1,)
   cov_mat = corr_mat * risk**2

   streams = np.random.multivariate_normal(means, cov_mat,
size=num_samples)

return streams.T
```

Portfolio Risk Reduction

In [3]:

```
n = 5
mean, std, corr = 10, 15, 0.6
streams = correlated_streams(n, mean, std, corr)
```

In [4]:

```
streams.mean(axis=1)
```

Out[4]:

```
array([10.12229747,  9.92797016,  9.98877207, 10.05103342,
9.90978558])
```

In [5]:

```
streams.std(axis=1)
```

Portfolio Risk Reduction

Out[5]:

```
array([15.07254044, 15.05168254, 15.17926238, 15.2192544 ,
15.14908131])
```

In [6]:

```
np.corrcoef(streams)
```

Out[6]:

```
array([[1.        , 0.60676484, 0.61222918, 0.61179636, 0.60301561],
       [0.60676484, 1.        , 0.61036834, 0.61049393, 0.61073826],
       [0.61222918, 0.61036834, 1.        , 0.61526424, 0.61265281],
       [0.61179636, 0.61049393, 0.61526424, 1.        , 0.605607  ],
       [0.60301561, 0.61073826, 0.61265281, 0.605607  , 1.        ]])
```

Portfolio Risk Reduction

In [7]:

```
def aggregate_risk(return_streams, n):
    """Returns the pooled risk (std) of the `n` first streams
    in `return_streams`
    """
    assert len(return_streams) >= n

    aggregate_returns = np.sum(return_streams[:n], axis=0) / n
    return aggregate_returns.std()
```

Portfolio Risk Reduction

In [8]:

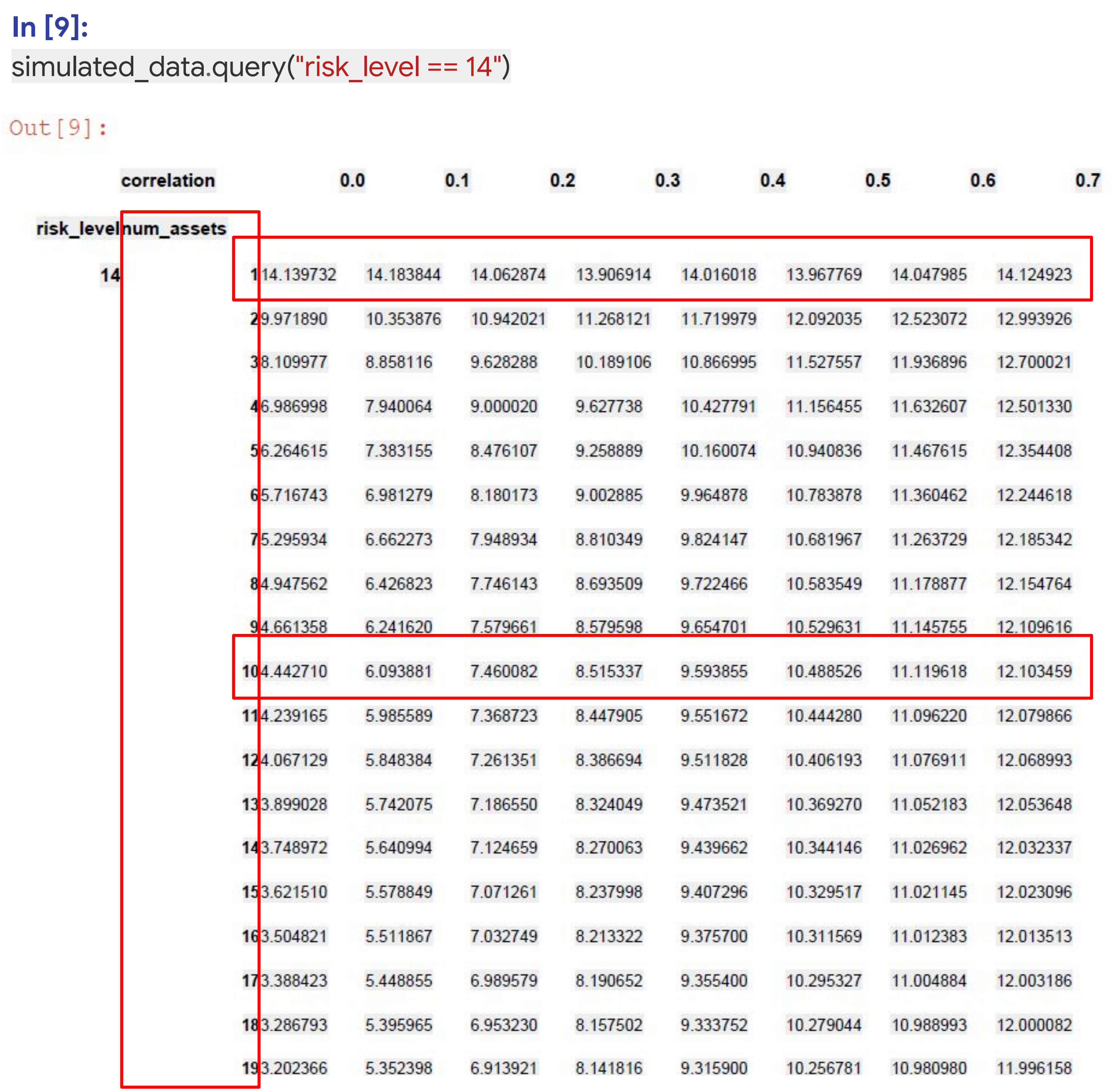
```
max_assets = 20
assets = range(1, max_assets+1)

mean = 10 # Avg mean return of 10%
risk_levels = range(1, 15)

index = pd.MultiIndex.from_product([risk_levels, assets],
names=["risk_level", "num_assets"])
simulated_data = pd.DataFrame(index=index)

for risk in risk_levels:
    for corr in np.arange(0.0, .8, 0.1):
        return_streams = correlated_streams(max_assets, mean, risk,
corr)
        risk_level = np.zeros(max_assets)
        for num_assets in assets:
            risk_level[num_assets-1] = aggregate_risk(return_streams,
num_assets)
        simulated_data.loc[(risk, ), round(corr, 1)] = risk_level
simulated_data.columns.names = ["correlation"]
```

Portfolio Risk Reduction



In [9]:

```
simulated_data.query("risk_level == 14")
```

Out[9]:

	correlation	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
risk_level	num_assets								
14	1	14.139732	14.183844	14.062874	13.906914	14.016018	13.967769	14.047985	14.124923
	2	9.971890	10.353876	10.942021	11.268121	11.719979	12.092035	12.523072	12.993926
	3	8.109977	8.858116	9.628288	10.189106	10.866995	11.527557	11.936896	12.700021
	4	6.986998	7.940064	9.000020	9.627738	10.427791	11.156455	11.632607	12.501330
	5	6.264615	7.383155	8.476107	9.258889	10.160074	10.940836	11.467615	12.354408
	6	5.716743	6.981279	8.180173	9.002885	9.964878	10.783878	11.360462	12.244618
	7	5.295934	6.662273	7.948934	8.810349	9.824147	10.681967	11.263729	12.185342
	8	4.947562	6.426823	7.746143	8.693509	9.722466	10.583549	11.178877	12.154764
	9	4.661358	6.241620	7.579661	8.579598	9.654701	10.529631	11.145755	12.109616
	10	4.442710	6.093881	7.460082	8.515337	9.593855	10.488526	11.119618	12.103459
	11	4.239165	5.985589	7.368723	8.447905	9.551672	10.444280	11.096220	12.079866
	12	4.067129	5.848384	7.261351	8.386694	9.511828	10.406193	11.076911	12.068993
	13	3.899028	5.742075	7.186550	8.324049	9.473521	10.369270	11.052183	12.053648
	14	3.748972	5.640994	7.124659	8.270063	9.439662	10.344146	11.026962	12.032337
	15	3.621510	5.578849	7.071261	8.237998	9.407296	10.329517	11.021145	12.023096
	16	3.504821	5.511867	7.032749	8.213322	9.375700	10.311569	11.012383	12.013513
	17	3.388423	5.448855	6.989579	8.190652	9.355400	10.295327	11.004884	12.003186
	18	3.286793	5.395965	6.953230	8.157502	9.333752	10.279044	10.988993	12.000082
	19	3.202366	5.352398	6.913921	8.141816	9.315900	10.256781	10.980980	11.996158

Portfolio Risk Reduction

In [10]:

```
def plot_risk_level(data, risk_level):
    subset = data.query("risk_level == @risk_level")
    stacked = subset.stack().reset_index(name="risk")
    stacked.head()

    chart = alt.Chart(data=stacked)

    highlight = alt.selection(type="single", on="mouseover",
                              fields=["correlation"], nearest=True)

    base = chart.encode(
            alt.X("num_assets", axis=alt.Axis(title="Number of
Assets")),
            alt.Y("risk", axis=alt.Axis(title="Risk %")),
            alt.Color("correlation:N", scale=alt.Scale(scheme="set2")))
```

In [10]: (continued)

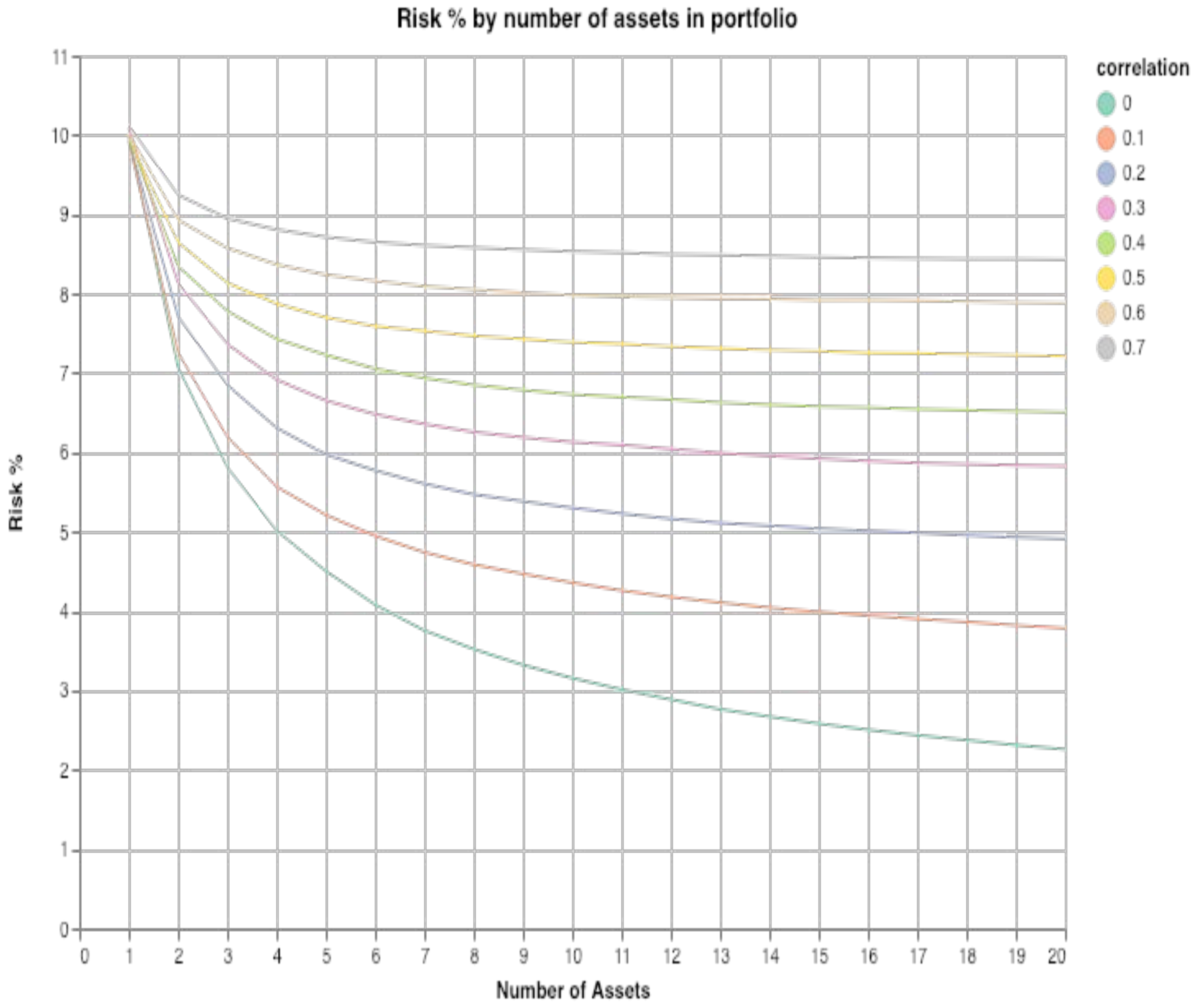
```
    points = base.mark_circle().encode(
        opacity=alt.value(0)
    ).add_selection(
        highlight
    ).properties(
        height=400,
        width=600,
        title="Risk % by number of assets in portfolio"
    )
    lines = base.mark_line().encode(
        size=alt.condition(~highlight, alt.value(1), alt.value(3)),
        tooltip=["correlation"]
    )
    return points + lines
```

In [11]:

```
plot_risk_level(simulated_data, 10)
```

Out[11]:

Portfolio Risk Reduction



Portfolio Level Risk Management

- Reduced risk through exposure to different sources of income
    - Combine uncorrelated revenue streams
    - From a number of trading assets
    - Capture “true” alpha and use leverage to increase returns in a “Risk Parity” strategy*

* https://www.bridgewater.com/resources/risk-parity-is-about-balance.pdf