Welcome to Advanced Stochastic Processes 1401-02's documentation!

Indices and tables

- Index
- Module Index
- Search Page

Data module

class data.**Option**(*tag*, *stock_symbol*, *option_symbol*, *strike*, *maturity_date*, *call*)

call

Alias for field number 5

maturity_date

Alias for field number 4

option_symbol

Alias for field number 2

stock_symbol

Alias for field number 1

strike

Alias for field number 3

tag

Alias for field number 0

data.**add_rf**(*data: DataFrame*) → DataFrame

Add risk-free rate to dataframe Args:

data (pd.DataFrame): dataframe to add rf to

Returns:

pd.DataFrame: dataframe with rf column

data.**add_std**(*data: DataFrame*, *rolling_window: int = 90*, *annualized: bool = True*) → DataFrame

Add rolling standard deviation to dataframe Args:

data (pd.DataFrame): dataframe to add std to rolling_window (int, optional): rolling window for std. Defaults to 90. annualized (bool, optional): annualize std. Defaults to True.

Returns:

pd.DataFrame: dataframe with std column

data.**fetch_data**(*option: Option*) → DataFrame

Fetch historical data for a given option along with underlying stock Args:

option (Option): option to fetch data for

Returns:

pd.DataFrame: historical data

data.**fetch_stoch_history**(*symbol: str*, *days: int = 365*) → DataFrame

Fetch historical data for a given stock Args:

symbol (str): stock symbol start_date (str): start date end_date (str): end date

Returns:

pd.DataFrame: historical data

`data.`**`read_bonds`**`() → DataFrame`
Read bonds data from excel file

`data.`**`read_fund_portfolio_options`**`(`*`fund: str`*`)`
Read fund portfolio options from excel file

Option pricing models

`class` `options.`**`BLS`**
Black-Scholes pricing model for European options

`static` **`add_price`**`(`*`option: src.data.Option`*`, `*`data: DataFrame`*`) → DataFrame`
Add option price to dataframe Args:

data (pd.Dataframe): dataframe to add option price to

Returns:
pd.DataFrame: dataframe with option price

`static` **`price`**`(`*`S0: float`*`, `*`K: float`*`, `*`T: float`*`, `*`r: float`*`, `*`v: float`*`, `*`call: bool = True`*`) → float`
Price an option using the Black-Scholes pricing model

Args:
S0 (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate v (float): annualized volatility call (bool, optional): True for call option, False for put option. Defaults to True.

Returns:
float: option price

`class` `options.`**`Bionomial`**
Binomial pricing model for European options

`static` **`add_price`**`(`*`option: src.data.Option`*`, `*`data: DataFrame`*`) → DataFrame`
Add option price to dataframe Args:

data (pd.Dataframe): dataframe to add option price to

Returns:
pd.DataFrame: dataframe with option price

`static` **`price`**`(`*`S0: float`*`, `*`K: float`*`, `*`T: float`*`, `*`r: float`*`, `*`v: float`*`, `*`N: int`*`, `*`call: bool = True`*`) → float`
Price an option using the binomial pricing model Args:

S0 (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate v (float): annualized volatility N (int): number of time steps call (bool, optional): True for call option, False for put option. Defaults to True.

Returns:
float: option price

`class` `options.`**`MontCarlo`**
Monte Carlo pricing model for European options

`static` **`add_price`**`(`*`option: src.data.Option`*`, `*`data: DataFrame`*`) → DataFrame`
Add option price to dataframe Args:

data (pd.Dataframe): dataframe to add option price to

Returns:
pd.DataFrame: dataframe with option price

`static` **`price`**`(`*`S0: float`*`, `*`K: float`*`, `*`T: float`*`, `*`r: float`*`, `*`v: float`*`, `*`N: int`*`, `*`num_simulations: int = 10000`*`) → float`
Price an option using the Monte Carlo pricing model

Args:

S0 (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate v (float): annualized volatility num_simulations (int, optional): number of simulations. Defaults to 10000.

Returns:

float: option price

class options.**Trinomial**

Trinomial pricing model for European options

static **add_price**(*option: src.data.Option*, *data: DataFrame*) → DataFrame

Add option price to dataframe Args:

data (pd.Dataframe): dataframe to add option price to

Returns:

pd.DataFrame: dataframe with option price

static **price**(*S0: float*, *K: float*, *T: float*, *r: float*, *v: float*, *N: int*, *call: bool = True*) → float

Price an option using the trinomial pricing model Args:

s0 (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate v (float): annualized volatility N (int): number of time steps call (bool, optional): True for call option, False for put option. Defaults to True.

Returns:

float: option price

class options.**Volatility**

Volatility class

static **black_scholes_model**(*S*, *K*, *T*, *r*, *sigma*)

Black-Scholes model for European options

Args:

S (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate

Returns:

float: option price

static **implied_volatility**(*S*, *K*, *T*, *r*, *C0*, *sigma_est*, *it=100*) → float

Calculate implied volatility

Args:

S (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate C0 (float): option price sigma_est (float): estimated volatility

Returns:

float: implied volatility

static **vega**(*S*, *K*, *T*, *r*, *sigma*)

Calculate vega

Args:

S (float): initial stock price K (float): strike price T (float): time to maturity as a fraction of one year r (float): risk-free interest rate sigma (float): annualized volatility

Returns:

float: vega

Statistical analysis of the results of the option pricing models.

analysis.**get_coint_df**(*results: list[pandas.core.frame.DataFrame]*, *option_tags: list[str]*) → DataFrame

Calculate the cointegration test results for each option

Args:

results (list[pd.DataFrame]): list of results for each option option_tags (list[str]): list of option tags

Returns:

pd.DataFrame: DataFrame containing the cointegration test results for each option

analysis.**get_errs**(*results: list[pandas.core.frame.DataFrame]*, *option_tags: list[str]*) → DataFrame

Calculate the error metrics for each option

Args:

results (list[pd.DataFrame]): list of results for each option option_tags (list[str]): list of option tags

Returns:

pd.DataFrame: DataFrame containing the error metrics for each option

analysis.**get_is_diff_correlated**(*results: list[pandas.core.frame.DataFrame]*, *option_tags: list[str]*) → DataFrame

Calculate the correlation between the difference of the actual option price and the model price and the time to maturity

Args:

results (list[pd.DataFrame]): list of results for each option option_tags (list[str]): list of option tags

Returns:

pd.DataFrame: DataFrame containing the correlation results for each option

Interest rate models

class interest.**HullWhite**

Hull-White model class

static **generate_paths**(*mean_reversion*, *volatility*, *initial_rate*, *maturity*, *num_paths*, *num_steps*)

Generate paths using the Hull-White model

Parameters

mean_reversion : *float*

Mean reversion parameter

volatility : *float*

Volatility parameter

initial_rate : *float*

Initial short rate

maturity : *float*

Maturity of the short rate

num_paths : *int*

Number of paths to generate

num_steps : *int*

Number of time steps per path

Returns

array_like

Array of paths

static **objective**(*params*, **args*) → float

Objective function for calibration of Hull-White model to market

Parameters

params : *tuple(float, float)*

a, sigma

args : *tuple(array_like, array_like, array_like)*

prices, maturities, ytm

Returns

float

Sum of squared errors between model prices and market prices

static **plot_paths**(*paths*, *num_steps*, *maturity*)

Plot the paths generated by the Hull-White model

Args:

paths (array_like): Array of paths num_steps (int): Number of time steps per path maturity (float): Maturity of the short rate

Returns:

None

Plotting functions for the project

plot.**plot_errs**(*errors: DataFrame*, *option_tags: list[str]*)

Plot the errors for each option

Args:

errors (pd.DataFrame): errors for each option option_tags (list[str]): list of option tags

Returns:

None

plot.**plot_methods**(*results: list[pandas.core.frame.DataFrame]*, *option_tags: list[str]*) → None

Plot the results of different pricing methods

Args:

results (list[pd.DataFrame]): list of results for each option option_tags (list[str]): list of option tags

Returns:

None