

DejaVu Serif DejaVu Sans DejaVu Sans Mono

lgimapy

Jason R.Becker

Created on : May, 2019

Last updated : June, 2019

Table of contents

Table of contents	i
List of figures	ii
List of tables	iii
1 Documentation for the Code	1
2 lgimapy index	2
3 lgimapy models	7
4 lgimapy data	10
5 lgimapy vis	11
6 lgimapy bloomberg	12
7 lgimapy daily_scripts	13
8 Indices and tables	14
Python Module Index	15

List of figures

List of tables

Chapter 1

Documentation for the Code

Chapter 2

lgimapy index

```
class lgimapy.index.Index(index_df, name="")
```

Class for indexes built form *IndexBuilder*

Parameters

- `index_df` (*pd.DataFrame*) – Index DataFrame from *IndexBuilder.build()*.
- `name` (*str*, *default=''*) – Optional name of index.

`df`

Full index DataFrame.

Type *pd.DataFrame*

`cusips`

List of all unique cusips in Index.

Type *List[str]*

`dates`

Sorted list of all dates in index.

Type *List[datetime object]*

`day(date)`: Return Index or DataFrame of single speciifed date.

`clean_treasuries()`: Clean treasuries to on the run T bonds only.

`market_value_weight()`: Market value weight a specified column
vs entire index market value.

`clean_treasuries()`

Clean treasury index for building model curve.

`cusips`

Memoized unique cusips in Index.

`dates`

Memoized unique sorted dates in Index.

`day(date, as_index=False)`

Memoized call to a dict of single day DataFrame with date as key.

Parameters

- `date` (*datetime object*) – Date of daily DataFrame to return.
- `as_index` (*bool*, *default=False*) – If true, return an Index for specified day instead of a DataFrame.

Returns df or Index – DataFrame or Index for specified date.

Return type `pd.DataFrame` or *Index*

`day_persistent_constituents(date, as_index=False)`

Creates Index or DataFrames containing only the intersection of constituents which existed in the index both the day before and the specified day.

Parameters

- `date` (*datetime object*) – Date of daily DataFrame to return.
- `as_index` (*bool, default=False*) – If true, return an Index for specified day instead of a DataFrame.

Returns df or Index – DataFrame or Index for specified date.

Return type `pd.DataFrame` or *Index*

`find_rating_changes(rating_agency)`

Find rating changes of persistent index.

Parameters `rating_agency` (`{'SP', 'Moody', 'Fitch'}`.) – Rating agency to find changes for.

Returns change_df – DataFrame of all rating changes, with no index, columns are ['date', 'cusip', 'change'].

Return type `pd.DataFrame`

`get_cusip_history(cusip)`

Get full history for specified cusip.

Parameters `cusip` (*str*) – Specified cusip.

Returns hist_df – DataFrame with datetime index and Index.df columns for specified cusip.

Return type `pd.DataFrame`

`get_value_history(col)`

Get history of any column for all cusips in Index.

Parameters `col` (*str*) – Column from *Index.df* to build history for (e.g., 'OAS').

Returns hist_df – DataFrame with datetime index, CUSIP columns, and price values.

Return type `pd.DataFrame`

`market_value_weight(col, df=None)`

Market value weight a specified column vs entire index market value.

Parameters

- `col` (*str*) – Column name to weight, (e.g., 'OAS').
- `df` (*pd.DataFrame, default=None*) – DataFrame to use for analysis, use full Index.df if None.

Returns mvw – Market value weighting of specified column.

Return type `float`

`subset_value_by_rating(col, save=False, fid=None, path=None)`

Find the market value weighted value of specified column by rating.

Parameters

- `col` (*str*) – Column from Index.df to market value weight.
- `save` (*bool*) – If true, store resulting DataFrame to fid.
- `fid` (*str, default=None*) – File name to store file, by default is input `col`.
- `path` (*Path, default=None*) – Path to store file, by default is `./data/subset_by_rating`.

Returns subset_df – if save is false, return DataFrame of results with datetime index and numeric rating columns.

Return type `pd.DataFrame`


```
class lgimapy.index.IndexBuilder
```

Class to pull and clean index data from SQL database.

df

Full DataFrame of all cusips over loaded period.

Type pd.DataFrame

trade_dates

List of dates with bond data.

Type list[datetime]

loaded_dates

List of dates currently loaded by builder.

Type list[datetime]

build(name="", start=None, end=None, rating='IG', currency=None, cusip=None, issuer=None, ticker=None, sector=None, subsector=None, municipals=True, treasuries=False, maturity=(None, None), price=(None, None), country_of_domicile=None, country_of_risk=None, amount_outstanding=(300, None), issue_years=(None, None), collateral_type=None, OAD=(None, None), OAS=(None, None), OASD=(None, None), liquidity_score=(None, None), credit_stats_only=False, credit_returns_only=False, financial_flag=None, special_rules=None)
Build index with customized rules from [IndexBuilder.df](#).

Parameters

- **name** (*str*, *default*='') – Optional name for returned index.
- **start** (*datetime*, *default*=None) – Start date for index, if None the start date from load is used.
- **end** (*datetime*, *default*=None) – End date for index, if None the end date from load is used.
- **rating** (*str*, *Tuple*[*str*, *str*], *default*='IG') – Bond rating/rating range for index.
Examples:
 - *str*: 'HY', 'IG', 'AAA', 'Aa1', etc.
 - *Tuple*[*str*, *str*]: (AAA, BB) uses all bonds in specified inclusive range.
- **currency** (*str*, *List*[*str*], *default*=None) – Currency or list of currencies to include.
- **cusip** (*str*, *List*[*str*], *default*=None) – Cusip or list of cusips to include.
- **issuer** (*str*, *List*[*str*], *default*=None) – Issuer, or list of issuers to include.
- **ticker** (*str*, *List*[*str*], *default*=None) – Ticker or list of tickers to include in index, default is all.
- **sector** (*str*, *List*[*str*], *default*=None) – Sector or list of sectors to include in index.
- **subsector** (*str*, *List*[*str*], *default*=None) – Subsector or list of subsectors to include in index.
- **municipals** (*bool*, *default*=False) – If True, include municipal bonds in index.
- **treasuries** (*bool*, *default*=False) – If True, include treasury bonds in index.
- **maturity** (*Tuple*[*float*, *float*], {5, 10, 20, 30}, *default*=None) – Maturities to include, if int is specified the following ranges are used:
 - 5: 4-6
 - 10: 6-11
 - 20: 11-25
 - 30: 25 - 31
- **price** (*Tuple*[*float*, *float*], *default*=(None, None)) – Price range of bonds to include, default is all.
- **country_of_domicile** (*str*, *List*[*str*], *default*=None) – Country or list of countries of domicile to include, default is all.

- `country_of_risk` (*str*, *List[str]*, *default=None*) – Country or list of countries wherer risk is centered to include in index, default is all.
- `amount_outstanding` (*Tuple[float, float]*, *default=(300, None)*) – Range of amount outstanding to include in index (Millions).
- `issue_years` (*Tuple[float, float]*, *default=(None, None)*) – Range of years since issue to include in index, default is all.
- `collateral_type` (*str*, *List[str]*, *default=None*) – Collateral type or list of types to include, default is all.
- `OAD` (*Tuple[float, float]*, *default=(None, None)*) – Range of option adjusted durations to include, default is all.
- `OAS` (*Tuple[float, float]*, *default=(None, None)*) – Range of option adjusted spreads to include, default is all.
- `OASD` (*Tuple[float, float]*, *default=(None, None)*) – Range of option adjusted spread durations, default is all.
- `liquidity_score` (*Tuple[float, float]*, *default=(None, None)*) – Range of liquidity scores to use, default is all.
- `credit_stats_only` (*bool*, *default=False*) – If True, only include bonds with credit stats in index.
- `credit_returns_only` (*bool*, *default=False*) – If True, only include bonds with credit returns in index.
- `financial_flag` (*{'financial', 'non-financial', 'other'}*, *default=None*) – Financial flag setting to identify fin and non-fin credits.
- `special_rules` (*str*, *List[str]*, *default=None*) – Special rule(s) for subsetting index using bitwise operators. If None, all specified inputs are applied independently of eachother as bitwise &. All rules can be stacked using paranthesis to create more complex rules.

Examples:

- Include specified sectors or subsectors: ‘Sector | Subsector’
- Include all but specified sectors: ‘~Sector’
- Include either (all but specified currency or specified sectors) xor specified maturities: ‘(~Curnecy | Sector) ^ MaturityYears’

Returns *Index* with specified rules.

Return type *Index*

`load(start=None, end=None, cusips=None, clean=True, dev=False, ret_df=False, local=False, data=None)`

Load data from SQL server. If end is not specified data is scraped through previous day. If neither start nor end are given only the data from previous day is scraped. Optionally load from local compressed format for increased performance or feed a DataFrame directly.

Parameters

- `start` (*str*, *datetime*, *default=None*) – Starting date for scrape.
- `end` (*str*, *datetime*, *default=None*) – Ending date for scrape.
- `cusips` (*List[str]*, *default=None*) – List of cusips to specify for the load, by default load all.
- `clean` (*bool*, *default=True*) – If true, apply standard cleaning rules to loaded DataFrame.
- `dev` (*bool*, *default=False*) – If True, use development SQL query to load all fields. **DO NOT USE IN PRODUCTION CODE**
- `ret_df` (*bool*, *default=False*) – If True, return loaded DataFrame.
- `local` (*bool*, *default=False*) – Load index from local binary file.

Returns *df* – DataFrame for specified date’s index data if *ret_df* is true.

Return type *pd.DataFrame*

`nearest_date(date)`

Return trade date nearest to input date.

Parameters *date (datetime object)* – Input date.

Returns `t_date` – Trade date nearest to input date.

Return type datetime object

`trade_dates`

Datetime index array of dates with credit data.

`lgimapy.index.spread_diff(df1, df2)`

Calculate spread difference of index values from df1 to df2.

Parameters

- `df1` (*pd.DataFrame*) – Older index DataFrame.
- `df2` (*pd.DataFrame*) – Newer index DataFrame.

Returns `df`

Return type `pd.DataFrame`

`lgimapy.index.standardize_cusips(df)`

Standardize CUSIPs, converting cusips which changed name to most recent cusip value for full history.

Parameters `df` (*pd.DataFrame*) – DataFrame of selected index cusips.

Returns `df` – DataFrame with all CUSIPs updated to current values.

Return type `pd.DataFrame`

Chapter 3

lgimapy models

```
class lgimapy.models.TreasuryCurve
```

Class for loading treasury yield curve and calculating yields for specified dates and maturities.

`trade_dates`

Type List of traded dates.

`get_yield(date, t)`: Get yield for specified date and maturities.

`get_KRD_total_returns(original_date, new_date, t)`

Reprice hypothetical par bond used for KRDs with a new yield curve.

`get_KRD_yields`

Memoized yields for specified date and maturities of [0.5, 2, 5, 10, 20, 30] years.

Parameters `date` (*datetime object*) – Date of yield curve.

Returns `yields` – Yields for [0.5, 2, 5, 10, 20, 30] years.

Return type [1 x 6] Array[float]

`get_KRDs_and_coupons`

Memoized key rate durations for specified date and maturities of [0.5, 2, 5, 10, 20, 30] years.

Parameters `date` (*datetime object*) – Date of yield curve.

Returns `krds` – KRDs for [0.5, 2, 5, 10, 20, 30] years.

Return type [1 x 6] Array[float]

`get_yield(date, t)`

Vectorized implementation to get yield(s) for a given date and maturities.

Parameters

- `date` (*datetime object*) – Date of yield curve.

- `t` (*float, nd.array[float]*) – Maturity or maturities (yrs) to return yields for.

Returns `y` – Yields for specified maturities.

Return type float, ndarray[float]

```
class lgimapy.models.TreasuryCurveBuilder(ix)
```

Class for fitting treasury curve and visualizing curve.

Parameters `ix` ([Index](#)) – Index class form IndexBuilder containing treasuries for a single date.

`fit()`: Fit curve to input data.

`plot()`: Plot fitted treasury curve.

`fit(method='price', n=25, n_drop=10, threshold=12, solver='SLSQP', verbose=0)`

Solve zero coupon bond curve given all bonds by minimizing the total squared error between the theoretical yield curve and market observed yield to maturities.

Methodolgy and notation taken from - <https://www.jstatsoft.org/article/view/v036i01/v36i01.pdf>

Parameters

- `init_params` (`{'price', 'yield'}`, `default='price'`) – Minimize error in inversion duration weighted price or yield.
- `n` (`int`, `default=20`) – Number of minimizations to run for final yield curve.
- `n_drop` (`int`, `default=10`) – Number of minimizations to run before dropping bonds with large errors.
- `theshold` (`float`, `default=12`) – Threshold (bp) that ytm errors must be within to not drop bonds from fitting routine.
- `solver` (`str`, `default='SLSQP'`) – Scipy solver to use when minimizing errors.
- `verbose` (`int`, `default=0`) –
Verbosity level for optimization.
 - 0: No results.
 - 1: Updated each time bonds are dropped.
 - 2: Update each completed optimization simulation.
 - 3: Update each optimization iteration.

`plot(trange=(0.1, 30), indv_bonds=False, strips=False, ax=None, figsize=(8, 6))`

Plot yield curve.

Parameters

- `trange` (`Tuple(float, float)`, `default=(0.1, 30)`) – Range (min, max) in years to show.
- `indv_bonds` (`bool`, `default=False`) – If true, show individual bond yields as data points.
- `ax` (`matplotlib axis`, `default=None`) – Matplotlib axis to plot figure, if None one is created.
- `figsize` (`list or tuple`, `default=(6, 6)`) – Figure size.

`save()`

Save beta values to `../data/treasury_curve_params.csv`.

`class lgimapy.models.TBond(s)`

Class for treasury bond math and manipulation given current state of the bond.

Parameters `series` (`pd.Series`) – Single bond row from `index_builder` DataFrame.

`coupon_dates`

Type All coupon dates for given treasury.

`coupon_years`

Type Time to all coupons in years.

`calculate_price(rfr)`: Calculate price of bond with given risk free rate.

`calculate_price(rfr)`

Calculate theoreticla price of the bond with given risk free rate.

Parameters `rfr` (`float`) – Continuously compounded risk free rate used to discount cash flows.

Returns `price` – Theoretical price (\$) of bond.

Return type `float`

`cash_flows`

Return `np.array` of cash flows to be acquired on `coupon_dates`.

coupon_dates

Return list[datetime object] of timestamps for all coupons. Note that all coupons are assumed to be on either the 15th or last day of the month, business days and holidays are ignored.

coupon_days

Return np.array of time in years for all coupons.

coupon_years

Return np.array of time in years for all coupons.

theoretical_ytm(*price*)

Calculate yield to maturity of the bond using true coupon values and dates and specified price.

Parameters *price* (*float*) – Price of the bond to use when calculating yield to maturity.

Returns *ytm* – Yield to maturity of the bond at specified price.

Return type float

ytm

Memoized yield to maturity of the bond using true coupon values, dates, and dirty price.

Parameters *price* (*float*, *default=None*) – Price of the bond to use when calculating yield to maturity. If no price is given, the market value dirty price is used.

Returns *ytm* – Yield to maturity of the bond at specified price.

Return type float

Chapter 4

lgimapy data

Chapter 5

lgimapy vis

Chapter 6

lgimapy bloomberg

`lgimapy.bloomberg.get_bloomberg_subsector(cusips)`

Get bloomberg subsector for list of cusips. First attempts to use saved *bloomberg_subsectors.json* file, updating the file for any cusip which is unsuccessful.

Parameters *cusips* (*str*, *List[str]*) – Cusip(s) to search bloomberg for subsectors.

Returns *subsectors* – List of bloomberg subsectors matching input cusips.

Return type *List[str]*

`lgimapy.bloomberg.update_subsector_json()`

Update bloomberg subsector json file.

Chapter 7

lgimapy daily_scripts

Chapter 8

Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

Python Module Index

|

lgimapy.bloomberg, [12](#)
lgimapy.daily_scripts, [13](#)
lgimapy.data, [10](#)
lgimapy.index, [2](#)
lgimapy.models, [7](#)
lgimapy.vis, [11](#)