Quant Insight

Qi Documentation

Release 0.9.0

May 14, 2019

CONTENTS

CHAPTER ONE

OVERVIEW

This document introduces the Quant Insight API wrapper which allows a Data Analyst with a Python environment to query the Quant Insight Web API using an intuitive Python interface.

This module provides an easy to use, object-oriented API which supports both synchronous and asynchronous access.

CHAPTER TWO

QUICK START GUIDE

Our Rest API connects your applications to real-time quant macro analytics that seamlessly integrates with **your** systems. From a quantitative macro risk management stress-testing tool, to integrating Qi's trading signals, there are a multitude of use-cases of Qi data integration.

This Python package is automatically generated by the Swagger Codegen project:

- API version: 0.9.0
- Package version: 0.9.0
- Build package: io.swagger.codegen.languages.PythonClientCodegen

2.1 Requirements.

Python 2.7 and 3.4+

2.2 Installing

To install your package, please contact Qi support contact to retrieve an **API download token**, then install using the following command, replacing `YOUR-TOKEN-HERE` with the provided download token:

```
pip install \
  --extra-index-url=https://dl.cloudsmith.io/YOUR-TOKEN-HERE/quant-insight/python/
↪python/index/ \
  qi-client
```

Note that this token allows access to download the python API, but a separate token (the **Qi API Key**) is required to access Quant Insight data

Then that's it! You're ready to import the package:

```
import qi_client
```

2.3 Upgrading

To upgrade your package at a later time, you can then run:

```
pip install \
  --upgrade \
  --extra-index-url=https://dl.cloudsmith.io/DLTOKEN/quant-insight/python/python/
↪index/ \
  qi-client
```

If that fails, it may be due to an expired token in which case you should request a new one from the Qi support team.

2.4 Getting Started

Please follow the *installation procedure* and then run the following:

```
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))
bucket = 'bucket_example' # str | Numeric ID or name of the bucket to retrieve
numeric_id = false # bool | If set to true, will consider identifier as a numeric ID -
↪ not as a name. (optional) (default to false)

try:
    # Get bucket details
    api_response = api_instance.get_bucket(bucket, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_bucket: %s\n" % e)
```

2.5 List of Methods Available

Class	Method	Description
DefaultApi	**get_bucket**	Get bucket details
DefaultApi	**get_bucket_drivers**	Get drivers for a given bucket
DefaultApi	**get_buckets**	Get all available buckets
DefaultApi	**get_driver**	Get driver details
DefaultApi	**get_instrument**	Get instrument definition
DefaultApi	**get_instruments**	Get all instruments defined
DefaultApi	**get_model**	Get a model definition
DefaultApi	**get_model_history**	Get model definition history
DefaultApi	**get_model_sensitivities**	Get sensitivities for a model
DefaultApi	**get_model_timeseries**	Get model timeseries data
DefaultApi	**get_models**	Get list of all defined models on the system
DefaultApi	**get_tags**	Get list of all defined tags on the system

2.6 Documentation For Authorization

2.6.1 Qi API Key

- **Type**: API key
- **API key parameter name**: X-API-KEY
- **Location**: HTTP header

CHAPTER

THREE

API REFERENCE

3.1 get_bucket

Bucket get_bucket(bucket, numeric_id=numeric_id)

Get bucket details

3.1.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the bucket to retrieve
# Datatype: str
bucket = 'bucket_example'

# If set to true, will consider identifier as a numeric ID - not as a name.␣
↪(optional) (default to false)
# Datatype: bool
numeric_id = false


try:
    # Get bucket details
    api_response = api_instance.get_bucket(bucket, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_bucket: %s\n" % e)
```

3.1.2 Parameters

Name	Description
bucket [str]	Numeric ID or name of the bucket to retrieve
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.1.3 Return type

Bucket

3.1.4 Authorization

Qi API Key

3.2 get_bucket_drivers

list[Driver] get_bucket_drivers(bucket, numeric_id=numeric_id)

Get drivers for a given bucket

3.2.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the bucket to examine
# Datatype: str
bucket = 'bucket_example'

# If set to true, will consider identifier as a numeric ID - not as a name.
↪(optional) (default to false)
# Datatype: bool
numeric_id = false


try:
    # Get drivers for a given bucket
```

(continues on next page)

(continued from previous page)

```
    api_response = api_instance.get_bucket_drivers(bucket, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_bucket_drivers: %s\n" % e)
```

3.2.2 Parameters

Name	Description
bucket [str]	Numeric ID or name of the bucket to examine
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.2.3 Return type

list[Driver]

3.2.4 Authorization

Qi API Key

3.3 get_buckets

list[Bucket] get_buckets()

Get all available buckets

3.3.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))


try:
    # Get all available buckets
    api_response = api_instance.get_buckets()
```

(continues on next page)

(continued from previous page)

```
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_buckets: %s\n" % e)
```

3.3.2 Parameters

This endpoint does not need any parameter.

3.3.3 Return type

list[Bucket]

3.3.4 Authorization

Qi API Key

3.4 get_driver

Driver get_driver(driver, numeric_id=numeric_id)

Get driver details

3.4.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the driver to examine
# Datatype: str
driver = 'driver_example'

# If set to true, will consider identifier as a numeric ID - not as a name.␣
↪(optional) (default to false)
# Datatype: bool
numeric_id = false
```

(continues on next page)

(continued from previous page)

```
try:
    # Get driver details
    api_response = api_instance.get_driver(driver, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_driver: %s\n" % e)
```

3.4.2 Parameters

Name	Description
driver [str]	Numeric ID or name of the driver to examine
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.4.3 Return type

Driver

3.4.4 Authorization

Qi API Key

3.5 get_instrument

Instrument get_instrument(instrument, mnemonic=mnemonic, numeric_id=numeric_id)

Get instrument definition

3.5.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the instrument to retrieve
```

(continues on next page)

(continued from previous page)

```
# Datatype: str
instrument = 'instrument_example'

# Where a string is provided as an instrument, this is the associated field name (e.g.
↪ BEST_EPS / PX_LAST) (optional) (default to PX_LAST)
# Datatype: str
mnemonic = 'PX_LAST'

# If set to true, will consider identifier as a numeric ID - not as a name.␣
↪(optional) (default to false)
# Datatype: bool
numeric_id = false


try:
    # Get instrument definition
    api_response = api_instance.get_instrument(instrument, mnemonic=mnemonic, numeric_
↪id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_instrument: %s\n" % e)
```

3.5.2 Parameters

Name	Description
instrument [str]	Numeric ID or name of the instrument to retrieve
mnemonic [str]	Where a string is provided as an instrument, this is the associated field name (e.g. BEST_EPS / PX_LAST)
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.5.3 Return type

Instrument

3.5.4 Authorization

Qi API Key

3.6 get_instruments

list[Instrument] get_instruments()

Get all instruments defined

3.6.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))


try:
    # Get all instruments defined
    api_response = api_instance.get_instruments()
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_instruments: %s\n" % e)
```

3.6.2 Parameters

This endpoint does not need any parameter.

3.6.3 Return type

list[Instrument]

3.6.4 Authorization

Qi API Key

3.7 get_model

Model get_model(model, term=term, version=version, numeric_id=numeric_id)

Get a model definition

3.7.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
```

(continues on next page)

(continued from previous page)

```
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve
# Datatype: str
model = 'model_example'

# Which model term to retrieve (optional)
# Datatype: str
term = 'term_example'

# Which version of the model to retrieve data for. (optional)
# Datatype: int
version = 56

# If set to true, will consider identifier as a numeric ID - not as a name.␣
↪(optional) (default to false)
# Datatype: bool
numeric_id = false


try:
    # Get a model definition
    api_response = api_instance.get_model(model, term=term, version=version, numeric_
↪id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_model: %s\n" % e)
```

3.7.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve
term [str]	Which model term to retrieve
version [int]	Which version of the model to retrieve data for.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.7.3 Return type

Model

3.7.4 Authorization

Qi API Key

3.8 get_model_history

list[ModelRevisionSchema] get_model_history(model, term=term, numeric_id=numeric_id)

Get model definition history

3.8.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for
# Datatype: str
model = 'model_example'

# Which model term to retrieve (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name.
↪(optional) (default to false)
# Datatype: bool
numeric_id = false


try:
    # Get model definition history
    api_response = api_instance.get_model_history(model, term=term, numeric_
↪id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_model_history: %s\n" % e)
```

3.8.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for
term [str]	Which model term to retrieve
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.8.3 Return type

list[ModelRevisionSchema]

3.8.4 Authorization

Qi API Key

3.9 get_model_sensitivities

object get_model_sensitivities(model, date_from=date_from, date_to=date_to, term=term, numeric_id=numeric_id, version=version)

Get sensitivities for a model

3.9.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve sensitivities for
# Datatype: str
model = 'model_example'

# Date in YYYY-MM-DD format from which to retrieve data (optional)
# Datatype: date
date_from = '2013-10-20'

# Date in YYYY-MM-DD format until which to retrieve data (optional)
# Datatype: date
```

(continues on next page)

(continued from previous page)

```
date_to = '2013-10-20'

# Which model term to retrieve (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name.␣
↪(optional) (default to false)
# Datatype: bool
numeric_id = false

# Which version of the model to retrieve data for. (optional)
# Datatype: int
version = 56


try:
    # Get sensitivities for a model
    api_response = api_instance.get_model_sensitivities(model, date_from=date_from,␣
↪date_to=date_to, term=term, numeric_id=numeric_id, version=version)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_model_sensitivities: %s\n" % e)
```

3.9.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve sensitivities for
date_from [date]	Date in YYYY-MM-DD format from which to retrieve data
date_to [date]	Date in YYYY-MM-DD format until which to retrieve data
term [str]	Which model term to retrieve
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.
version [int]	Which version of the model to retrieve data for.

3.9.3 Return type

object

3.9.4 Authorization

Qi API Key

3.10 get_model_timeseries

list[RegressionEntrySchema] get_model_timeseries(model, date_from=date_from, date_to=date_to, term=term, numeric_id=numeric_id, version=version)

Get model timeseries data

3.10.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for
# Datatype: str
model = 'model_example'

# Date in YYYY-MM-DD format from which to retrieve data (optional)
# Datatype: date
date_from = '2013-10-20'

# Date in YYYY-MM-DD format until which to retrieve data (optional)
# Datatype: date
date_to = '2013-10-20'

# Which model term to retrieve (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name.␣
↪(optional) (default to false)
# Datatype: bool
numeric_id = false

# Which version of the model to retrieve data for. (optional)
# Datatype: int
version = 56


try:
    # Get model timeseries data
    api_response = api_instance.get_model_timeseries(model, date_from=date_from, date_
↪to=date_to, term=term, numeric_id=numeric_id, version=version)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_model_timeseries: %s\n" % e)
```

3.10.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for
date_from [date]	Date in YYYY-MM-DD format from which to retrieve data
date_to [date]	Date in YYYY-MM-DD format until which to retrieve data
term [str]	Which model term to retrieve
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.
version [int]	Which version of the model to retrieve data for.

3.10.3 Return type

list[RegressionEntrySchema]

3.10.4 Authorization

Qi API Key

3.11 get_models

list[Model] get_models(tags=tags, asset_classes=asset_classes)

Get list of all defined models on the system

3.11.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited list of tags to filter results with. Results must contain *all*␣
↪tags specified. (optional)
# Datatype: str
tags = 'tags_example'

# Comma delimited list of asset classes to filter results with. Results must contain␣
↪*all* asset classes specified. (optional)
# Datatype: str
```

(continues on next page)

(continued from previous page)

```
asset_classes = 'asset_classes_example'


try:
    # Get list of all defined models on the system
    api_response = api_instance.get_models(tags=tags, asset_classes=asset_classes)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_models: %s\n" % e)
```

3.11.2 Parameters

Name	Description
tags [str]	Comma delimited list of tags to filter results with. Results must contain *all* tags specified.
asset_classes [str]	Comma delimited list of asset classes to filter results with. Results must contain *all* asset classes specified.

3.11.3 Return type

list[Model]

3.11.4 Authorization

Qi API Key

3.12 get_tags

list[TagSchema] get_tags()

Get list of all defined tags on the system

3.12.1 Examples

```
from __future__ import print_function
import time
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to set up a proxy
# configuration.proxy = 'http://localhost:3128'

# create an instance of the API class
```

(continues on next page)

(continued from previous page)

```
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))


try:
    # Get list of all defined tags on the system
    api_response = api_instance.get_tags()
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->get_tags: %s\n" % e)
```

3.12.2 Parameters

This endpoint does not need any parameter.

3.12.3 Return type

list[TagSchema]

3.12.4 Authorization

Qi API Key

Model List

3.13 Bucket

The model **Bucket** has the following properties:

Name	Description
id [int]	System-internal numeric identifier for this bucket / driver group.
name [str]	The name of this bucket

3.14 Driver

The model **Driver** has the following properties:

Name	Description
buckets [list[str]]	List of buckets which this driver belongs to.
expression [str]	Definition of the expression which defines this driver's value (e.g. 'a' or 'a + b').
id [int]	System-internal numeric identifier for this driver definition.
instrument1 [Instrument]	First instrument (instrument 'a') in this driver definition.
instrument2 [Instrument]	Second instrument (instrument 'b') in this driver definition if applicable.
instrument3 [Instrument]	Third instrument (instrument 'c') in this driver definition if applicable.
instrument4 [Instrument]	Fourth instrument (instrument 'd') in this driver definition if applicable.
name [str]	The unique name for this driver.
short_name [str]	A non-unique display name for this driver.

3.15 Instrument

The model **Instrument** has the following properties:

Name	Description
coverage [str]	Type of coverage for market data for this instrument.
id [int]	System-internal numeric identifier for this ticker / field combination.
mnemonic [str]	Field name (e.g. PX_LAST).
ticker [str]	Underlying ticker (e.g. AAPL UW Equity).

3.16 Model

The model **Model** has the following properties:

Name	Description
asset_class [str]	This is the name of the asset class to which this model belongs (e.g. Equity, FX etc).
created [datetime]	This is the date when this model was created.
definition [ModelDefinition]	
drivers [list[Driver]]	This is the list of drivers which are expected to influence this model.
id [int]	System-internal numeric identifier for this Qi Model. This will change on update so please don't use this as an external identifier.
model_parameter [str]	Term for this parameter (e.g. 'short term' or 'long term').
name [str]	A unique name for this model. Any lookups for this model should be made according to this name.
notes [str]	This is a field which may be populated in the model to give a better description of its purpose, function and constitution.
status [str]	This is the current state of the Model.
tags [list[str]]	List of tags associated with this model to assist with filtering.
version [int]	This is a numeric value for the current model revision.

3.17 ModelDefinition

The model **ModelDefinition** has the following properties:

Name	Description
bucket [str]	The driver bucket which this combination belongs to.
expression [str]	Definition of the expression which defines this combination's value (e.g. 'a' or 'a + b').
id [int]	System-internal numeric identifier for this instrument combination and expression.
instrument1 [Instrument]	First instrument (instrument 'a') in this instrument combination.
instrument2 [Instrument]	Second instrument (instrument 'b') in this instrument combination if applicable.
instrument3 [Instrument]	Third instrument (instrument 'c') in this instrument combination if applicable.
instrument4 [Instrument]	Fourth instrument (instrument 'd') in this instrument combination if applicable.

3.18 RegressionEntrySchema

The model **RegressionEntrySchema** has the following properties:

Name	Description
absolute_gap [float]	The absolute gap for actual vs projected for this model.
constant [float]	The constant applied to the model (for Qi debug only).
_date [datetime]	This is the date which the regression data has been calculated for.
fair_value [float]	The fair value for the model for the given date.
percentage_gap [float]	The percentage gap for actual vs projected for this model.
rsquare [float]	The rsquare value of the model.
sensitivities [list[SensitivitySchema]]	Optional list of sensitivities for this model in the given date.
sigma [float]	The value of sigma for the current model.
target_mean [float]	The mean of the actual model across its sample window.
target_stdev [float]	The standard deviation of the actual model across its sample window.
target_zscore [float]	The zscore of the model's actual value across its sample window.
zscore [float]	The zscore of the model's projected value across its sample window.

3.19 SensitivitySchema

The model **SensitivitySchema** has the following properties:

Name	Description
bucket_name [str]	This is the bucket which the driver which this sensitivity result is referring to is a member of.
coefficient [float]	This is a Qi-internal field used during calculation. It is provided for debug purposes only.
driver [Driver]	This is the driver which this sensitivity result is referring to.
driver_contribution [float]	This is the magnitude of this driver's influence on the model in question.
driver_name [str]	This is the name of the driver which this sensitivity result is referring to.
driver_short_name [str]	This is the short (display) name of the driver which this sensitivity result is referring to.
driver_zscore [float]	This is the zscore of this driver for this sensitivity across this model's zscore window.
driver_zscore_window_mean [float]	This is the mean of this driver for this sensitivity across this model's zscore window.
driver_zscore_window_stdev [float]	This is the standard deviation of this driver for this sensitivity across this model's zscore window.
sensitivity [float]	This is the magnitude of this driver's sensitivity to the model in question.

3.20 TimeSeriesEntrySchema

The model **TimeSeriesEntrySchema** has the following properties:

Name	Description
adjusted_value [float]	This is the adjusted value of the metric being queried.
adjustment [float]	This is the adjustment factor of the metric being queried.
_date [datetime]	This is the date which this value refers to.
stdev [float]	This is the standard deviation of the metric being queried.
volume [float]	This is not currently used.

CHAPTER

FOUR

CODE EXAMPLES

This section provides code examples that demonstrate common scenarios using Qi's Module for Python. Note that each example assumes that code similar to the following is at the top of each script:

```
import qi_client
import pandas
from datetime import datetime

configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))
```

The examples also make use of several standard, publicly available APIs. These APIs will need to be installed prior to running any of the examples. They may be installed using pip:

```
$ pip install matplotlib pandas
```

4.1 Historical R-Squared Data

Define a function to get a specified Model's R-Squared data for a given date range and model term.

```
def get_rsq(model, start, end, term):
    # Note that this may be  more than 1 year of data, so need to split requests
    year_start = int(start[:4])
    year_end = int(end[:4])
    time_series = []
    for year in range(year_start, year_end + 1):
        query_start = start
        if year != year_start:
            date_from = '%d-01-01' % year
        else:
            date_from = start
        if year != year_end:
            date_to = '%d-12-31' % year
        else:
            date_to = end

        print("Gathering data for %s from %s to %s..." % (model,
                                                            date_from,
                                                            date_to))
        time_series += api_instance.get_model_timeseries(
            model,
            date_from=date_from,
            date_to=date_to,
            term=term)

    rsq = [data.rsquare for data in time_series]
    dates = [data._date for data in time_series]

    df = pandas.DataFrame({'Dates': dates, 'Rsq': rsq})
    df.set_index('Dates', inplace=True)

    return df
```

Having defined a function, we can use it to graph the R-Squared timeseries for a specified model. In this example we plot:

SPX Index LT model

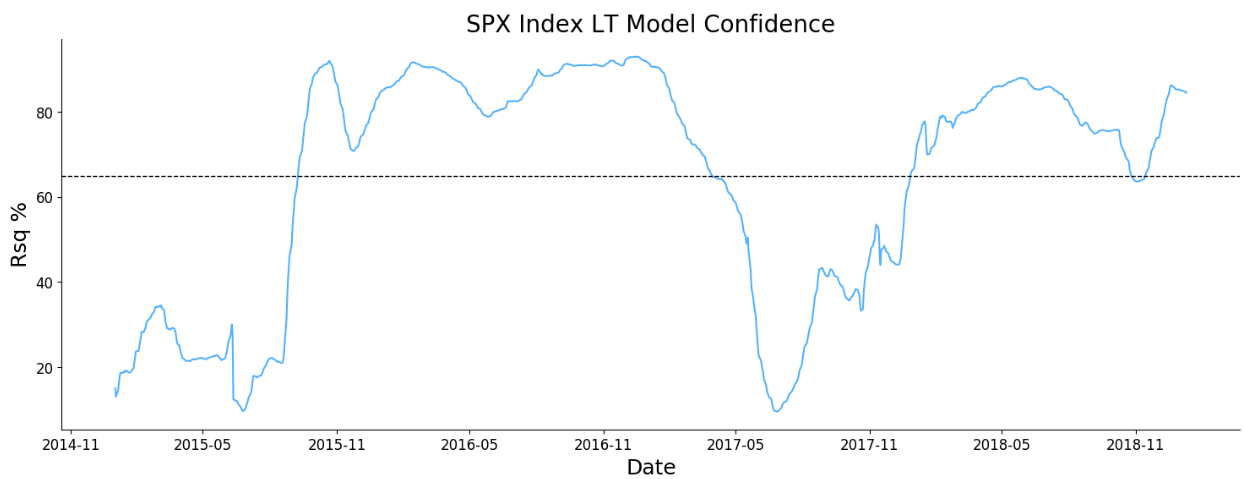
```
def example_historical_rsq():
    import matplotlib.pyplot as plt

    df = get_rsq('SPX',
                 '2015-01-01',
                 '2019-01-10',
                 'Long Term')

    fig = plt.figure(figsize=(18, 6))

    ax = plt.subplot(111)
    fig.patch.set_facecolor('#FFFFFF')

    plt.plot(df.index.values, df['Rsq'], color='#53B2FF')

```

(continues on next page)

(continued from previous page)

```
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)

    plt.ylabel("Rsq %", fontsize=18)
    plt.xlabel('Date', fontsize=18)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    plt.title('SPX Index LT Model Confidence', fontsize=20)

    ax.axhline(65, lw=1, linestyle='--', color='k')

    plt.show()

    fig.savefig('Qi_API_Rsq_Graph_Example_(SPX_Index)',
                bbox_inches="tight",
                facecolor=fig.get_facecolor())
```

R-Squared can be interpreted as a measure of model confidence. From the end of 2015 to the middle of 2017 macro had strong explanatory power for US equities. The sharp fall in R-Squared in the middle of 2017 suggests other non-macro factors became more significant; these could potentially include momentum, sentiment, positioning, geopolitics etc. More recently the power of macro has re-asserted itself.



4.2 Historical Factor Sensitivities

Define a function to get a specified Model's factor sensitivity data for a given date range, factor and model term.

```
def get_sensitivities(model, factors, start, end, term):
    # Note that this may be  more than 1 year of data, so need to split requests
    year_start = int(start[:4])
    year_end = int(end[:4])
    time_series_sensitivities = {}
    for year in range(year_start, year_end + 1):
        query_start = start
        if year != year_start:
            date_from = '%d-01-01' % year
        else:
            date_from = start
        if year != year_end:
            date_to = '%d-12-31' % year
        else:
            date_to = end

        print("Gathering data for %s from %s to %s..." % (model,
                                                          date_from,
                                                          date_to))
        time_series_sensitivities.update(
            api_instance.get_model_sensitivities(
                model,
                date_from=date_from,
                date_to=date_to,
                term=term
            )
        )

    dates = []
    factor_results = {}
    # Intialize factor lists for results
    for factor in factors:
        factor_results[factor] = []

    # Iterate over each time series result
    for date in sorted(time_series_sensitivities.keys()):
        # Use the real date for the index
        dates.append(datetime.strptime(date, '%Y-%m-%d'))
        # This will be the sensitivities for this particular day
        daily_sensitivities = time_series_sensitivities[date]
        # Only consider factors requested
        for factor in factors:
            factor_result = [sensitivity['sensitivity']
                             for sensitivity in daily_sensitivities
                             if sensitivity['driver_short_name'] == factor]
            if len(factor_result) > 0:
                factor_results[factor].append(factor_result[0])
            else:
                # Pyplot will ignore nan values rather than shift datapoint left
                factor_results[factor].append(float('nan'))

    # Add date range for X axis first
    dataframe_columns = {
```

(continues on next page)

(continued from previous page)

```
        'Dates': dates
    }

    # Add factor specific columns
    for factor in factors:
        dataframe_columns[factor] = factor_results[factor]

    # Initialize dataframe
    df = pandas.DataFrame(dataframe_columns)
    df.set_index('Dates', inplace=True)

    return df
```

Having defined a function, we can use it to graph a comparison across a number of model factor sensitivities. In this example we plot:

USDJPY LT Model - Interest Rate Differential Sensitivities

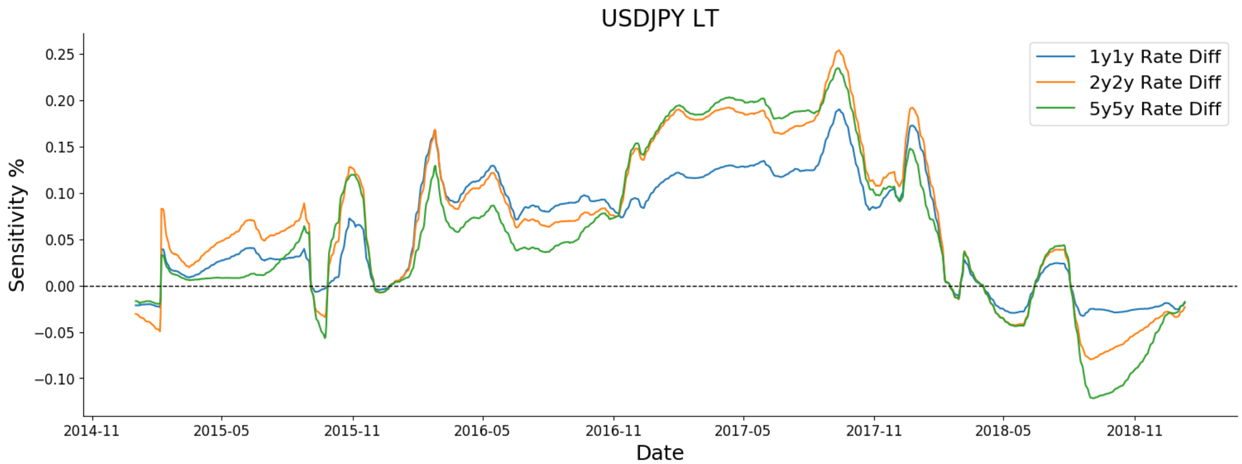
```
def example_historical_factor_sensitivities():
    import matplotlib.pyplot as plt

    df = get_sensitivities('USDJPY',
                           [
                               '1y1y Rate Diff.',
                               '2y2y Rate Diff.',
                               '5y5y Rate Diff.',
                           ],
                           '2015-01-01',
                           '2019-01-10',
                           'Long Term')

    fig = plt.figure(figsize=(18, 6))

    ax = plt.subplot(111)
    fig.patch.set_facecolor('#FFFFFF')

    plt.plot(df.index.values, df['1y1y Rate Diff.'], label='1y1y Rate Diff')
    plt.plot(df.index.values, df['2y2y Rate Diff.'], label='2y2y Rate Diff')
    plt.plot(df.index.values, df['5y5y Rate Diff.'], label='5y5y Rate Diff')

    plt.legend(loc='upper right', prop={'size': 16})

    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)

    plt.ylabel("Sensitivity %", fontsize=18)
    plt.xlabel('Date', fontsize=18)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    plt.title('USDJPY LT', fontsize=20)

    ax.axhline(0, lw=1, linestyle='--', color='k')

    plt.show()

    fig.savefig('Qi_API Factor_Sensitivity_Graph_Example_(USDJPY)',
```

(continues on next page)

(continued from previous page)

```
                bbox_inches="tight",
                facecolor=fig.get_facecolor())
```

Tracking USDJPY's sensitivity to interest rate differentials across different tenors. Notice that for most of 2017, FX was highly sensitive to changes in interest rates. Since the start of 2018 however, this relationship has deteriorated, and spot FX is currently indifferent to interest rate differentials.



4.3 Bucket Driver Sensitivities

Define a function to get a specified Model's bucket sensitivity data for a given date range and model term.

```
def get_bucket_drivers(model, date, term):
    sensitivity = api_instance.get_model_sensitivities(
                        model=model,
                        date_from=date,
                        date_to=date,
                        term=term
                    )

    df_sensitivities = pandas.DataFrame()
    for data in sensitivity[date]:
        if data['bucket_name'] in df_sensitivities.columns:
            df_sensitivities[str(data['bucket_name'])][0] = (
                df_sensitivities[str(data['bucket_name'])][0] +
                [data['sensitivity']]
            )
        else:
            df_sensitivities[str(data['bucket_name'])] = [data['sensitivity']]

    # Column heading
    df_sensitivities.index = ['Sensitivity']

    return df_sensitivities
```

Having defined a function, we can use it to graph the bucket driver sensitivities for a given model. In this example we plot:

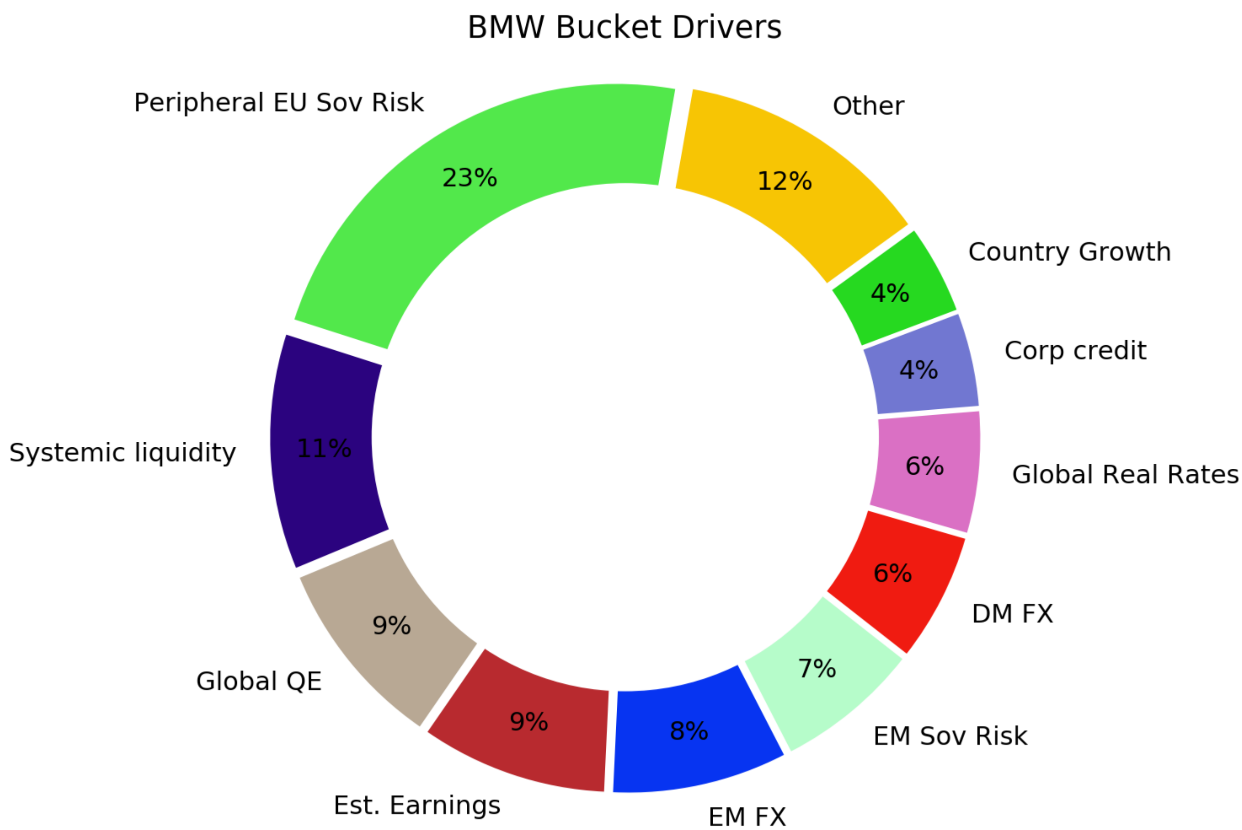
BMW LT model - bucket drivers

```
def example_bucket_driver_sensitivities():
    bucket_drivers = get_bucket_drivers('BMW',
                                        '2019-01-14',
                                        'Long Term')

    # Create Pie Data
    other = abs(bucket_drivers).transpose().nsmallest(5, 'Sensitivity').sum()
    df_other = pandas.DataFrame(other)
    df_other = df_other.rename(index={'Sensitivity': 'Other'},
                               columns={0: 'Sensitivity'})

    pie_bucket_drivers = abs(bucket_drivers).transpose().nlargest(10,
                                                              'Sensitivity')
    pie_bucket_drivers = pie_bucket_drivers.append(df_other)

    # Create Colours
    import random

    number_of_colors = 11

    color = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
             for i in range(number_of_colors)]

    df_color = pandas.DataFrame({'Color': color})
    df_color.index = pie_bucket_drivers.index.tolist()
```

(continues on next page)

(continued from previous page)

```
# Set up the plotting
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rcParams['font.size'] = 22

fig = plt.figure(figsize=(12, 12))

ax = plt.subplot(111)

# Data to plot
labels = pie_bucket_drivers.index.tolist()
sizes = [100*float(x) for x in pie_bucket_drivers['Sensitivity']]
colors = df_color.loc[pie_bucket_drivers.index.tolist()]['Color'].tolist()

# Plot
plt.pie(sizes,
        labels=labels,
        colors=colors,
        autopct='%1.f%%',
        pctdistance=0.84,
        shadow=False,
        explode=(
            0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05
        ),
        startangle=80)

centre_circle = plt.Circle((0, 0), 0.75, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.title('BMW Bucket Drivers')
plt.show()
fig.savefig('Qi_API_Bucket_Drivers_Pie_Chart_Example_(BMW).png',
            bbox_inches="tight",
            facecolor=fig.get_facecolor())
```


BMW Bucket Drivers
Peripheral EU Sov Risk
23%
Other
12%
Country Growth
4%
Corp credit
4%
Global Real Rates
6%
DM FX
6%
EM Sov Risk
7%
EM FX
8%
Est. Earnings
9%
Global QE
9%
Systemic liquidity
11%

4.4 Valuation Gaps

Define functions to get specified Model data for a given date and find the largest valuation gaps across a given set of assets.

```
def get_vals(model, start, end):
    year_start = int(start[:4])
    year_end = int(end[:4])
    time_series = []
    for year in range(year_start, year_end + 1):
        query_start = start
        if year != year_start:
            date_from = '%d-01-01' % year
        else:
            date_from = start
        if year != year_end:
            date_to = '%d-12-31' % year
        else:
            date_to = end

        print("Gathering data for %s from %s to %s..." % (model,
                                                          date_from,
                                                          date_to))
        time_series += api_instance.get_model_timeseries(
            model,
            date_from=date_from,
            date_to=date_to)

    Rsq = [data.rsquare for data in time_series]
    FVG = [data.sigma for data in time_series]
    dates = [data._date for data in time_series]

    df = pandas.DataFrame({'Dates': dates, 'Rsq': Rsq, 'FVG': FVG})
    df.set_index('Dates', inplace=True)

    return df
```

```
def get_top_valuation_gaps(date, models):
    FVG_s, ID, Names, Rsq_s = ([] for i in range(4))
    for asset in models:
        try:
            if float(get_vals(asset, date, date)['Rsq']) > 65:
                FVG_s.append(float(get_vals(asset, date, date)['FVG']))
                ID.append(asset)
                Names.append(
                    api_instance.get_model(
                        model=asset
                    ).name
                )
                Rsq_s.append(float(get_vals(asset, date, date)['Rsq']))
        # Skip if data not available
        except TypeError:
            continue

    df_FVG = pandas.DataFrame({'Name': Names, 'FVG': FVG_s, 'Rsq': Rsq_s})
    df_FVG.index = ID
```

(continues on next page)

(continued from previous page)

```
    top_gaps = pandas.concat([
        df_FVG.nsmallest(5, 'FVG'), df_FVG.nlargest(5, 'FVG')
    ])
    MAX = []
    MIN = []

    for asset in top_gaps['Name']:
        MAX.append(max(get_vals(asset, '2015-01-02', date)['FVG']))
        MIN.append(min(get_vals(asset, '2015-01-02', date)['FVG']))

    top_gaps['Max'] = MAX
    top_gaps['Min'] = MIN

    return top_gaps
```

Having defined these functions, we can use them to graph these top valuation gaps for a given date. In this example we plot:

Top Valuation Gaps

```
def example_valuation_gaps():

    stoxx_600 = list(set([
        model.name for model in api_instance.get_models(tags='STOXX Europe 600')
    ]))

    df = get_top_valuation_gaps('2019-01-14', sorted(stoxx_600))

    import matplotlib
    import matplotlib.pyplot as plt
    matplotlib.rcParams.update({'errorbar.capsize': 15})

    fig = plt.figure(figsize=(10, 7))

    ax = plt.subplot()

    plt.yticks(fontsize=12)
    plt.xticks(fontsize=12)

    ax.axhline(0, color='k', lw=1)
    fig.patch.set_facecolor('#FFFFFF')

    ax.errorbar(df['Name'],
                df['FVG'],
                [df['FVG'] - df['Min'], df['Max'] - df['FVG']],
                fmt='o',
                ms='20',
                color='#C3423F',
                ecolor='#53B2FF',
                lw=10,
                capthick=8,
                solid_capstyle='round',
                solid_joinstyle='round')

    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
```

(continues on next page)

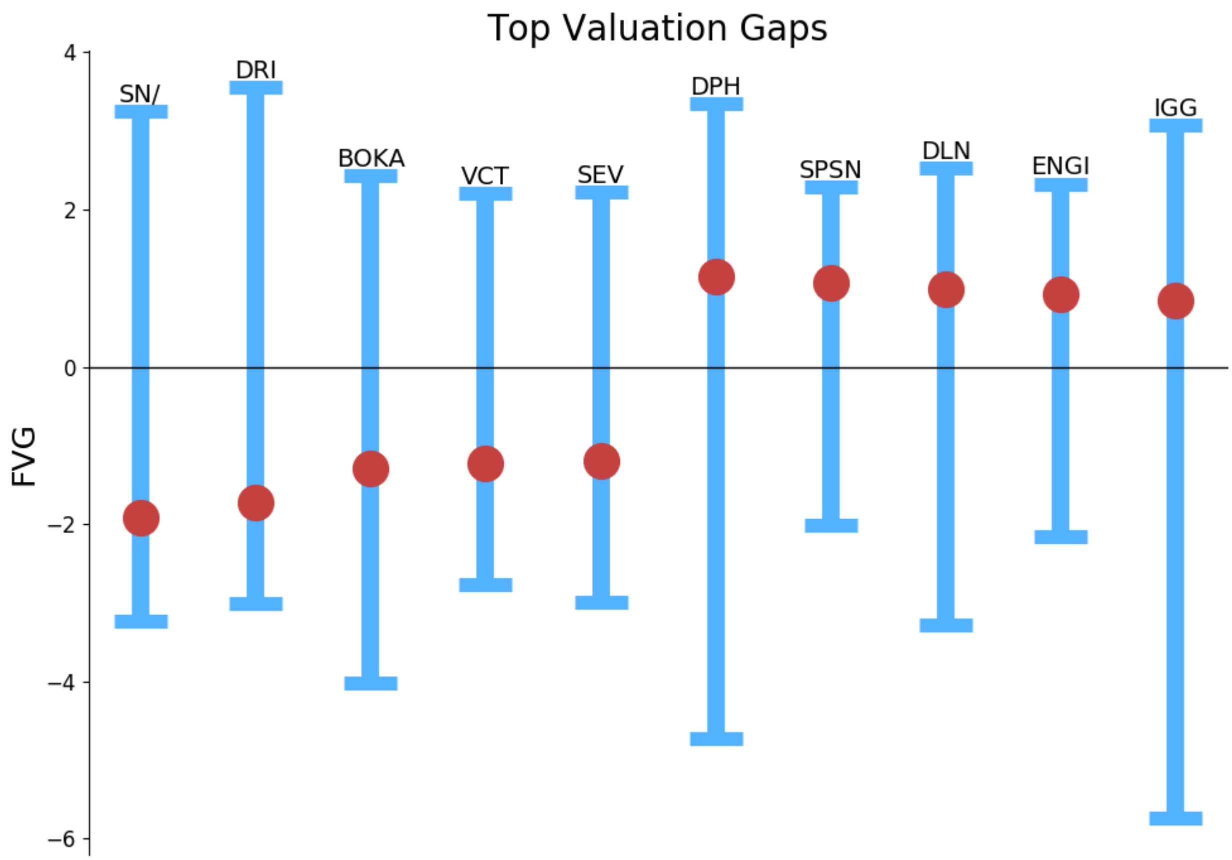
(continued from previous page)

```
    ax.tick_params(axis='x', bottom=False, labelbottom=False)

    plt.ylabel('FVG', fontsize=18)
    plt.title('Top Valuation Gaps', fontsize=20)

    for x in range(0, len(df['Name'][:10])):
        ax.annotate(df['Name'][x], (df['Name'][x], df['Max'][x]+0.12),
                    ha='center',
                    color='k',
                    fontsize=14)

    plt.tight_layout()

    df.to_csv(path_or_buf='Qi_API_Top_10_Valuation_Gaps.csv',
              float_format='%.5f',
              index_label='Model Name',
              columns=[
                  'Min',
                  'Max',
                  'FVG',
              ])

    fig.savefig('Qi_API_Top_10_Valuation_Gaps.png',
                bbox_inches="tight",
                facecolor=fig.get_facecolor())
    plt.show()
```

Displaying the results in a table and candle chart:

Table 1: Valuation Gap Summary

Model Name	Min	Max	FVG
SN/	-3.23550	3.24739	-1.91751
DRI	-3.00432	3.55386	-1.72134
BOKA	-4.02201	2.42945	-1.29859
VCT	-2.77558	2.20676	-1.23452
SEV	-3.00034	2.22265	-1.19572
DPH	-4.73037	3.34520	1.15388
SPSN	-2.01844	2.28753	1.06253
DLN	-3.28653	2.52369	0.98228
ENGI	-2.15225	2.32524	0.91743
IGG	-5.73275	3.06874	0.84491


Top Valuation Gaps
FVG
4
2
0
−2
−4
−6
SN/
DRI
BOKA
VCT
SEV
DPH
SPSN
DLN
ENGI
IGG

4.5 Qi Risk Factor Sensitivity

Define a function to create a portfolio sensitive to a given factor.

```python
def get_portfolio(factor, models, size, funds, date):
    FACTOR_sensitivity = []
    names = []
    POSITION = []

    for model in models:
        print("Getting model sensitivities for %s on %s" % (model, date))
        sensitivity = api_instance.get_model_sensitivities(
                            model=model,
                            date_from=date,
                            date_to=date
                        )
        df_sensitivities = pandas.DataFrame()
        if date not in sensitivity:
            continue
        for data in sensitivity[date]:
            df_sensitivities[str(data['driver_short_name'])] = [
                data['sensitivity']
            ]

        top10 = df_sensitivities.transpose().nlargest(10, 0)
        Factor_sens = float(df_sensitivities[factor])

        if factor in top10.index and Factor_sens > 0:
            FACTOR_sensitivity.append(Factor_sens)
            names.append(api_instance.get_model(model=model).name)
            position = top10.index.tolist().index(factor) + 1
            POSITION.append(position)

    portfolio_size = size
    df_factor_sensit = pandas.DataFrame({
                            'Name': names,
                            'Position': POSITION,
                            factor + ' Sensitivity': FACTOR_sensitivity
                        })
    portfolio = df_factor_sensit.nlargest(portfolio_size,
                                          str(factor) + ' Sensitivity')
    sw = 1 / (portfolio['Position'] + 2)
    summ = sum(sw)
    Weights = sw/summ
    position_values = [funds*x for x in Weights]
    factor_exposure = [a*b/100 for a, b in zip(
                            portfolio[factor + ' Sensitivity'],
                            position_values)]

    portfolio = portfolio.drop(['Position'], axis=1)
    portfolio.insert(1, 'Weight', Weights)
    portfolio.insert(2, 'Position Value', position_values)
    portfolio[factor + ' Exposure'] = factor_exposure

    return portfolio
```

Having defined a function we can use it to create a portfolio sensitive to a global trade war.

Portfolio sensitive to ADXY

```
def example_qi_risk_factor_sensitivity():

    stoxx_600 = list(set([
        model.name for model in api_instance.get_models(tags='STOXX Europe 600')
    ]))

    portfolio = get_portfolio('ADXY',
                              sorted(stoxx_600),
                              10,
                              1000000,
                              '2018-06-29')
    portfolio.to_csv(path_or_buf='Qi_API_Risk_Factor_Sensitivity.csv',
                     float_format='%.5f',
                     index=False,
                     columns=[
                         'Name',
                         'Weight',
                         'Position Value',
                         'ADXY Sensitivity',
                         'ADXY Exposure'
                     ])
    print(portfolio)
```

Displaying the results in a table:

Table 2: Risk Factor Sensitivity for ADXY

Name	Weight	Position Value	ADXY Sensitivity	ADXY Exposure
RNO	0.13013	130131.68087	0.09669	125.82432
EBS	0.10844	108443.06739	0.08983	97.41441
ENX	0.09295	92951.20062	0.08433	78.38575
ASM	0.09295	92951.20062	0.08348	77.59566
MTRO	0.09295	92951.20062	0.08055	74.87219
VACN	0.13013	130131.68087	0.07887	102.63486
TIT	0.10844	108443.06739	0.07807	84.66150
GLE	0.10844	108443.06739	0.07398	80.22618
MB	0.05422	54221.53369	0.07004	37.97676
SREN	0.08133	81332.30054	0.07002	56.94888

4.6 Qi Risk Factor Exposure

Define a function to find a portfolio's exposures to the top 10 macro factors.

```
def get_portfolio_exposures(portfolio, date, size):
    stock_names = portfolio['Name']
    df_tot = pandas.DataFrame()
    for stock in stock_names:
        sensitivity = api_instance.get_model_sensitivities(
                            model=stock,
                            date_from=date,
                            date_to=date
                        )
        df_sensitivities = pandas.DataFrame()
        for data in sensitivity[date]:
            df_sensitivities[str(data['driver_short_name'])] = [
                data['sensitivity']
            ]

        df_sensitivities = df_sensitivities.rename(index={0: stock})
        df_sensitivities = df_sensitivities.sort_index(axis=1)

        if df_tot.empty:
            df_tot = df_sensitivities
        else:
            df_tot = df_tot.append(df_sensitivities)

    SUMS = [sum(df_tot[x]) for x in df_tot.columns]
    df_tot.loc['Sum'] = SUMS
    portfolio_sensitivities = df_tot[
        df_tot.transpose().nlargest(size, 'Sum').index
    ].transpose()

    for stock in stock_names:
        portfolio_sensitivities[stock] = [
            a * float(
                portfolio.loc[portfolio['Name'] == stock]['Position Value']
            ) / 100
            for a in portfolio_sensitivities[stock]
        ]

    portfolio_sensitivities = portfolio_sensitivities.drop(columns='Sum')
    ex_SUMS = [
        sum(
            portfolio_sensitivities.loc[x]
        ) for x in portfolio_sensitivities.index
    ]
    portfolio_sensitivities['TOTAL'] = ex_SUMS
    portfolio_exposures = portfolio_sensitivities.transpose()

    return portfolio_exposures
```

Having defined a function we can use it determine the portfolio exposures to the top macro factors.

Portfolio exposure to each factor

```
def example_qi_risk_factor_exposure():
```

(continues on next page)

(continued from previous page)

```
stoxx_600 = list(set([
    model.name for model in api_instance.get_models(tags='STOXX Europe 600')
]))

portfolio = get_portfolio('ADXY',
                          sorted(stoxx_600),
                          10,
                          1000000,
                          '2018-06-29')

portfolio_exposures = get_portfolio_exposures(portfolio,
                                              '2018-06-29',
                                              5)

portfolio_exposures.to_csv(path_or_buf='Qi_API_Risk_Factor_Exposure.csv',
                           float_format='%.5f',
                           index_label='Name')
print(portfolio_exposures)
```

Displaying the results in a table:

Table 3: Risk Factor Exposure for ADXY

Name	Italian Sov. Confidence	Country TWI	Spain Sov. Confidence	ADXY	EUR 10y Real Rate
RNO	150.70550	138.00465	107.99628	125.82432	86.58962
EBS	173.30287	123.89620	124.19985	97.41441	78.39349
ENX	109.61735	126.46940	78.56235	78.38575	52.98218
ASM	-40.54531	109.28273	-29.05655	77.59566	19.71495
MTRO	4.21069	134.70488	3.02091	74.87219	26.17506
VACN	118.68009	139.48815	85.05407	102.63486	66.40620
TIT	199.74129	94.50813	143.14485	84.66150	81.14795
GLE	163.48877	124.56855	117.16189	80.22618	66.31294
MB	198.85205	65.98218	142.51046	37.97676	58.92796
SREN	158.61425	63.10573	113.67002	56.94888	60.11270
TOTAL	1236.66754	1120.01061	886.26414	816.54051	596.76305

CHAPTER

FIVE

GLOSSARY

5.1 Macro Factors

Below is a glossary for all Qi macro factors. In the first column, under 'Macro Factors', when there is a discrepancy between the Factor name and how it's labelled in the API, the appropriate label is shown in quotation marks

Table 1: Glossary for Qi Macro Factors

Macro Factors	Description	Definition	Interpretation
US High Yield 'US HY'	US High Yield Credit Spreads	An index based on a basket of 100 US single-name high yield Credit Default Swaps (CDS).	The CDS spread identifies concerns over default risk; it is the price the market is willing to pay for insurance against corporates defaulting. It can be thought of as a fear indicator. Spreads rise due to credit stress as the market pays more for credit protection. Up = higher credit risk in US High Yield corporates. Typically equities have a negative sensitivity to High Yield; i.e. a 1 SD move higher in CDS spreads would be negative for equity markets as they fear rising corporate defaults.
Itraxx Japan	Japanese Credit Spreads	An index based on a basket of 40 equally-weighted CDS on investment grade Japanese corporate credit.	Up = higher credit risk in Japan. Negative relationship with equities. Domestic equities suffer on the fear of rising corporate defaults. Note international equities benefited from tight spreads during the BoJ QE as it prompted capital flight from Japanese investors seeking higher yield.
Itraxx Crossover 'Itraxx Xover'	EUR High Yield Credit Spreads	An index of 75 equally weighted CDS on the most liquid sub investment grade European corporate entities.	Up = higher credit risk in Europe. Again, the typical relationship is wider CDS are negative for European equities. Greater demand for insurance against defaults is bad news for EU stocks.

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
FinSub Credit	EUR Financial Credit	An index of 30 equally weighted CDS on investment grade European financial entities.	Up = higher credit risk in EU banks. As above, greater demand for insurance against banks potentially defaulting would typically be negative for European financials and the broader equity market.
Brent	Crude Oil (Brent)	Energy commodity	Assets can have either a positive or negative relationship. Yields and energy stocks would expect to be positive: higher crude fuels higher inflation fears and helps the bottom line of energy stocks. Conversely, for other equities or even countries which are net importers, higher crude prices may be seen as a tax on consumers, businesses, sovereigns.
Western Texas Intermediate 'WTI'	Crude Oil (WTI)	Energy commodity	Assets can have either a positive or negative relationship. Yields and energy stocks would expect to be positive: higher crude fuels higher inflation fears and helps the bottom line of energy stocks. Conversely, for other equities or even countries which are net importers, higher crude prices may be seen as a tax on consumers, businesses, sovereigns.
Natural Gas	Natural Gas	Energy commodity	Assets can have either a positive or negative relationship. Yields and energy stocks would expect to be positive: higher crude fuels higher inflation fears and helps the bottom line of energy stocks. Conversely, for other equities or even countries which are net importers, higher crude prices may be seen as a tax on consumers, businesses, sovereigns.
EUR Swaption Rates Nvol 1y5y 'EUR 1y5y Rate Nvol'	ECB Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
USD Swaption Rates Nvol 1y5y 'USD 1y5y Rate Nvol'	Fed Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.
JPY Swaption Rates Nvol 1y5y 'JPY 1y5y Rate Nvol'	BoJ Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.
GBP Swaption Rates Nvol 1y5y 'GBP 1y5y Rate Nvol'	BoE Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.
CRB Food	Food prices	Commodity Research Bureau index of foodstuffs and components	More often than not, soft commodity prices are viewed as inflation proxies. A positive relationship implies the asset performs during a reflationary environment.

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
Wheat	Wheat	Wheat	More often that not, soft commodity prices are viewed as inflation proxies. A positive relationship implies the asset performs during a reflationary environment.
Copper	Copper	Industrial metal - futures contract	Can be 'spurious' but the typical pattern is higher industrial metal prices are positively associated with risky assets and bond yields, as they speak to a growing global economy.
Iron Ore	Iron Ore	Industrial metal - futures contract	Can be 'spurious' but the typical pattern is higher industrial metal prices are positively associated with risky assets and bond yields, as they speak to a growing global economy.
CRB Rind	Industrial Metals	Commodity Research Bureau Raw industrials Index	Can be 'spurious' but the typical pattern is higher industrial metal prices are positively associated with risky assets and bond yields, as they speak to a growing global economy.
Gold Silver Ratio	Gold Silver Ratio	Gold to Silver Ratio - futures contracts	Gold is widely perceived as a safe haven proxy. Higher gold prices relative to silver prices suggest uncertainty / fear is higher and there's a "flight-to-quality" dynamic at work.
VIX	VIX	A measure of the implied volatility of S&P 500 index options. Often referred to as the fear index or the fear gauge, it represents one measure of the market's expectation of stock market volatility over the next 30-day period.	Up = higher equity risk in US, more fear
VXEEM	VIX EEM	CBOE Emerging Markets ETF Volatility Index	Up = higher equity risk in EM, more fear
VDAX New	VDAX	Deutsche Boerse volatility index	Up = higher equity risk in EU, more fear
China GDP	Chinese GDP	China Current Quarter tracking GDP forecast (QoQ %) from Nowcast	Up = higher growth
Euro GDP	European GDP	Europe Current Quarter tracking GDP forecast (QoQ %) from Nowcast	Up = higher growth
US GDP	US GDP	US Current Quarter tracking GDP forecast (QoQ %) from Nowcast	Up = higher growth
Baltic Dry	Baltic Freight	An index used as a proxy for dry shipping stocks, often seen as a bellwether for industrial activity	Up = higher activity (implied stronger global growth)

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
EM CDS	EM sovereign risk	Generic 5y CDS of the sovereign(s) . The cost to insure against a country defaulting.	Up = higher default risk. US federal shutdowns, hard Brexit, fears about Chinese WMPs would all be examples of negative credit events that might cause the demand for protection to rise. Risky assets typically have a negative relationship, i.e. want sovereign risk to remain low.
China 5y CDS	China sovereign risk	Generic 5y CDS of the country. The cost to insure against a country defaulting.	Up = higher default risk. US federal shutdowns, hard Brexit, fears about Chinese WMPs would all be examples of negative credit events that might cause the demand for protection to rise. Risky assets typically have a negative relationship, i.e. want sovereign risk to remain low.
Greece 10y ASW 'Greek Sov. Confidence'	Greek Sovereign Risk	EUR 10y Swap rate minus Greece 10y Generic Bond Yield	Positive relationship means the financial asset performs when GGB yields stay comparatively low, i.e. there is less sovereign stress.
Italy 5y ASW 'Italian Sov. Confidence'	Italian Sovereign Risk	EUR 5y Swap rate minus Italy 5y Generic Bond Yield	In periods of increased political stress (like in the aftermath of the M5S / Lega Nord coalition winning the May 2018 election & in their spat with Brussels over the Italian budget) BTP yields rise relative to swaps. Any model that has a positive relationship at that time implies Italian politics needs to be benign for that asset to perform.
Spain 5y ASW 'Spain Sov. Confidence'	Spanish Sovereign Risk	EUR 5y Swap rate minus Spain 5y Generic Bond Yield	Negative relationship means the financial asset performs when SPGB yields spike versus swaps, i.e. there is increased sovereign stress.
Inflation expectations 2y EU '2y Infl. Expec.'	2y Inflation Expectations	2y inflation expectation for the country as measured by Zero Coupon inflation swaps	Up = higher inflation. Typically positive for risky assets (although not always) and negative for Fixed Income instruments.
Inflation expectations 5y EU '5y Infl. Expec.'	5y Inflation Expectations	5y inflation expectation for the country as measured by Zero Coupon inflation swaps	Up = higher inflation. Typically positive for risky assets (although not always) and negative for Fixed Income instruments.
Inflation expectations 10y EU '10y Infl. Expec.'	10y Inflation Expectations	10y inflation expectation for the country as measured by Zero Coupon inflation swaps	Up = higher inflation. Typically positive for risky assets (although not always) and negative for Fixed Income instruments.

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
EUR 1y CCY Basis Swap 'EUR 1y Basis Swap'	USD liquidity (EUR)	The cost for a European bank to fund themselves in USD	Up = lower risk in EUR currency. In general, cross currency basis is a measure of any Dollar shortage in financial markets. The more negative the basis becomes, the more severe the shortage. It is the additional hedging cost added to the interest differential of the two currencies. The most important drivers of the cross-currency basis spreads appear to be short and medium term EU financial sector credit risk and, to a slightly lesser extent, the equivalent US indicators. Calendar distortions like the year-end turn can impact but, in general, periods of more negative basis reflect worries about the EuroZone: peripheral stress, the sovereign-bank feedback loop etc.
JPY 1y CCY Basis Swap 'Jpy 1y Basis Swap'	USD liquidity (JPY)	The cost for a Japanese bank to fund themselves in USD	Up = lower risk in JPY currency. Given negative interest rates, Japanese investors typically will look to pick up yield overseas. Japan has huge savings pot that is looking for yield. USDJPY is a 'risk on' / 'risk off' metric. During periods of 'risk on' they will fund themselves using cheap currency which is Yen; so buy USD/sell JPY, and in 'risk off' it is usually sell USD/buy JPY.
5s30s Swapcurve '5s30s Swap Ccy1' '5s30s Swap Ccy2' 'Country 5s30s Swap'	Forward growth expectations	The spread between the 5y yield and the 30y yield of the relevant currency	Interest Rate curve proxy for medium term growth expectations. A bullish growth scenario implies higher 30yr yields versus 5yrs. Conversely, a recession would typically see the curve flatten / invert.
Country Economic Data 'Economic Data Ccy1' 'Economic Data Ccy2'	Economic Data	Country's Current Quarter GDP forecast (QoQ %) from Now-Cast. If it is not available, this factor represents the Surprise Index of that country (if NA then proxy: EM Surprise Index)	Up = stronger economic growth

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
Country TWI	Currency - Trade Weighted index	An index showing the value of a country's currency in relation to the currencies of a group of countries with which it trades. In the index, each country's currency is given an importance in relation to the amount of trade it does.	Up = Stronger Currency
Dollar Index 'DXY'	DXY Dollar Index	The U.S. Dollar Index (USDX, DXY) is an index of the value of the dollar (USD) in comparison to selected other currencies. Since the dollar index reflects the value of a basket of currencies relative to the dollar, it gives a more representative picture of the dollar's strength or weakness than a single currency pair like EUR/USD.	Up = USD higher vs DM FX.
Asia Dollar Index 'ADXY'	Asian EM FX	JPMorgan Asia Currency Index (ADXY), the first U.S. dollar tradable index of emerging Asian currencies. The ADXY is a spot index of emerging Asia's most actively traded currency pairs valued against the U.S. Dollar.	Higher means weaker Dollar, i.e. Up = USD lower vs Asia FX
EUR 10y Real Rate	EUR 10y Real Rate	EUR 10y Generic Nominal Yield minus 10y Expected Inflation	Reflect the real cost of capital so are often seen as the best single indicator for overall financial conditions in an economy. Can have a positive or negative relationship with equities. It was negative during large parts of the Great Financial Crisis - risky assets needed the ECB to keep monetary policy easy. A positive relationship suggests equities are sufficiently confident in a self-sustaining cyclical upswing that higher rates / tighter financial conditions reflect a strong economy which is good for earnings.
USD 10y Real Rate	USD 10y Real Rate	US 10y Generic Nominal Yield minus 10y Expected Inflation	Reflect the real cost of capital so are often seen as the best single indicator for overall financial conditions in an economy. Can have a positive or negative relationship with equities. It was negative during large parts of the Great Financial Crisis - risky assets needed the Fed to keep monetary policy easy. A positive relationship suggests equities are sufficiently confident in a self-sustaining cyclical upswing that higher rates / tighter financial conditions reflect a strong economy which is good for earnings/margins.

Continued on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
JPY 10y Real Rate	JPY 10y Real Rate	JPY 10y Generic Nominal Yield minus 10y Expected Inflation	Reflect the real cost of capital so are often seen as the best single indicator for overall financial conditions in an economy. Can have a positive or negative relationship with equities. It was negative during large parts of the Great Financial Crisis - risky assets needed the BoJ to keep monetary policy easy. A positive relationship suggests equities are sufficiently confident in a self-sustaining cyclical upswing that higher rates / tighter financial conditions reflect a strong economy which is good for earnings.
Index 1y Forward Earnings (eg EU) 'Index 1y Fwd Earnings'	1yr Forward Earnings	Market expectations for future earnings.	Up = better earnings ahead

5.2 Confidence

(Also known as R Squared or RSq) is a gauge of how sensitive the asset is to macroeconomic forces. Typically values above 65% are considered to show strong explanatory power or 'confidence'.

5.3 Qi Model Value

Represents the macro warranted fair value of the asset based on the Qi methodology.

5.4 Valuation Gap (VG) or Fair Valuation Gap (FVG)

The difference between the actual price of an asset and the Qi Model Value. This is quoted both as an absolute (can be percentage, USD or basis points depending on the instrument in question) and in standard deviation terms.

5.5 Rich

Represented by a positive 'FVG' and occurs when the market value of an asset is overvalued relative to the Qi macro warranted fair value.

5.6 Cheap

Represented by a negative 'FVG' and occurs when the market value of an asset is undervalued relative to the Qi macro warranted fair value.

5.7 Sensitivity

The impact on the price of the asset in %(basis points for rates) for a 1 standard deviation move up in the macro driver. This tells us the influence a driver has on the price of an asset, allowing us to rank drivers and also to identify the directional impact.

5.8 Historical Sensitivity

A historical time series of how the asset's sensitivity to a macro factor evolves.

5.9 Standard Deviation

Measures the dispersion of a set of data from its mean. It measures the absolute variability of a distribution; the higher the dispersion or variability, the greater is the standard deviation and greater will be the magnitude of the deviation of the value from their mean.

5.10 Driver

The individual macro factor we use as an explanatory variable in the Qi methodology e.g. US GDP

5.11 Driver Group

(Previously known as a bucket) The aggregation of related individual macro drivers into segments. Driver Group: Global Growth comprises US GDP, EU GDP and China GDP.

5.12 Driver Attribution

A breakdown of how much the movement in a driver has contributed to changes in the price of the underlying asset.

5.13 Custom Driver

A driver selected outside of the standard Qi driver set.

5.14 Model Type

Identifies whether the underlying explanatory variables have been curated by 'Qi' or whether they are a customised set, in which case they are 'Custom' models.

5.15 Custom Model

An asset modelled using a custom set of explanatory macro drivers.

5.16 Timeframe

The period over which the model is run can either be ST or LT.

- **ST** - Refers to a Short term model which is defined as 83 day lookback period.
- **LT** - Refers to a Long Term model and is defined as 250 day rolling lookback period.

5.17 Z-Score

The number of standard deviations from the mean a data point is.

5.18 Macro Regime

The broad macroeconomic dynamics impacting the asset. Within Qi, the Regime is defined as the set of most important macro drivers.

5.19 Macro Exposure

The overall level of sensitivity to macro drivers that the asset or portfolio is exhibiting.

5.20 Asset Class

The broad grouping of assets by similarities in market behaviour , laws and regulations.

5.21 Ticker

The corresponding bloomberg reference code for a given asset.