matchbox_api_utils Documentation

Release 0.15.3

Author

Sep 13, 2017

Contents:

CHAPTER 1

matchbox_api_utils package

1.1 Submodules

1.2 matchbox_api_utils.matchbox module

class matchbox_api_utils.matchbox.**Matchbox**(*url*, *creds*, *make_raw=None*)
Bases: object

MATCHBox API Connector Class.

Basic connector class to make a call to the API and load the raw data. From here we pass data, current MatchData or TreatmentArm data to appropiate calling classes.

1.3 matchbox_api_utils.matchbox_conf module

class matchbox_api_utils.matchbox_conf.**Config**(*mb_config_file=None*, *mb_json_data=None*, *ta_json_data=None*, *amois_lookup=None*)
Bases: object

classmethod read_config(*config_file*)

1.4 Module contents

class matchbox_api_utils.**Matchbox**(*url*, *creds*, *make_raw=None*)
Bases: object

MATCHBox API Connector Class.

Basic connector class to make a call to the API and load the raw data. From here we pass data, current MatchData or TreatmentArm data to appropiate calling classes.

class matchbox_api_utils.**MatchData**(*config_file=None*, *url=None*, *creds=None*, *patient=None*, *json_db='sys_default'*, *load_raw=None*, *make_raw=None*)

Bases: object

MatchboxData class

Parsed MATCHBox Data from the API as collected from the Matchbox class above. This class has methods to generate queries, further filtering, and heuristics on the dataset.

find_variant_frequency(*query*, *query_patients=None*)

Find and return variant hit rates.

Based on an input query in the form of a variant_type : gene dict, where the gene value can be a list of genes, output a list of patients that had hits in those gene with some disease and variant information.

Args:

query (dict): Dictionary of variant_type: gene mappings where:

- variant type is one or more of 'snvs','indels','fusions','cnvs'
- gene is a list of genes to query.

query_patients (list): List of patients for which we want to obtain data.

Returns: Will return a dict of matching data with disease and MOI information

Example: >>> query={'snvs' : ['BRAF','MTOR'], 'indels' : ['BRAF', 'MTOR']} find_variant_frequency(query)

gen_patients_list(*matchbox_data*, *patient*)

Process the MATCHBox API data.

Process the MATCHBox API data (usually in JSON format from MongoDB) into a much more concise and easily parsable dict of data. This dict will be the main dataset used for later data analysis and queries and is the main structure for the MatchboxData class below.

Returns: patients (dict): Dict of parsed MATCHBox API data.

get_biopsy_summary(*category=None*)

Return dict of patients registered in MATCHBox with biopsy and sequencing information.

Categories returned are total PSNs issued (including outside assay patients), total passed biopsies, total failed biopsies (per MDACC message), total MSNs (only counting latest MSN if more than one issued to a biopsy due to a failure) as a method of figuring out how many NAs were prepared, and total with sequencing data.

Can filter output based on any one criteria by leveraging the "category" variable

Args:

catetory (str): biopsy category to return. Valid categories are 'psn','passed_biopsy', 'failed_biopsy','no_biopsy','msn','sequenced','outside'.

Returns: dict: whole set of category:count or single category:count data.

get_bsn(*psn=None*, *msn=None*)

Retrieve a patient BSN from either an input PSN or MSN.

Args: psn (str): A PSN number to query. msn (str): A MSN number to query.

Returns: bsn (str): A BSN that maps to the PSN or MSN input.

```
>>> print(get_bsn(psn='14420'))
T-17-000550
```

get_disease_summary(*disease=None*)

Return a summary of registered diseases and counts. Despite a MEDRA Code and other bits of disease related data, we'll rely on output from CTEP Term value only.

Args: query_disease (str): Disease or comma separated list of diseases to filter on.

Returns: Dictionary of disease(s) and counts.

get_msn(*psn=None*, *bsn=None*)

Retrieve a patient BSN from either an input PSN or MSN.

Args: psn (str): A MSN number to query. bsn (str): A BSN number to query.

Returns: msn (str): A string of comma separated MSNs that map to an input BSN or PSN.

```
>>> print(get_msn(bsn='T-17-000550'))
MSN44180
```

get_patient_ta_status(*psn=None*)

Input a list of PSNs and return information about the treatment arm(s) to which they were assigned, if they were assigned to any arms. If no PSN list is passed to the function, return results for every PSN in the study.

Args: psn (list): Optional list of PSNs to query.

Returns: Dict of tuple of Status, Arm ID, and drug name.

Example:

```
>>> get_patient_ta_status(psn='10005')
{'10005' : (u'OFF_TRIAL_NO_TA_AVAILABLE', '---', '---')}
```

```
>>> get_patient_ta_status(psn='10837')
{'10837': (u'ON_TREATMENT_ARM', u'EAY131-Z1A', u'Binimetinib')}
```

get_patients_and_disease(*psn=None*, *msn=None*, *bsn=None*, *outside=False*, *no_disease=False*)

Return dict of PSN:Disease for valid biopsies. Valid biopsies can are defined as being only Passed and can not be Failed, No Biopsy or outside assay biopsies at this time.

Args: psn (str): Optional PSN or comma separated list of PSNs on which to filter data. bsn (str): Optional BSN or comma separated list of BSNs on which to filter data. msn (str): Optional MSN or comma separated list of MSNs on which to filter data. outside (bool): Also include outside assay data. False by default. no_disease (bool): Return all data, even if there is no disease indicated for the

patient specimen. Default: False

Returns: Dict of PSN : Disease mappings. If no match for input ID, returns None.

Example:

```
>>> print(get_disease(psn='11352'))
'Serous endometrial adenocarcinoma'
```

get_patients_by_arm(*armid*)

Input an arm ID and return a list of patients that are on the treatment arm

get_psn(*msn=None*, *bsn=None*)
Retrieve a patient PSN from either an input MSN or BSN.

Args: msn (str): A MSN number to query. bsn (str): A BSN number to query.

Returns: psn (str): A PSN that maps to the MSN or BSN input.

```
>>> print(get_psn(bsn='T-17-000550'))
PSN14420
```

get_seq_datafile(*dtype=None*, *msn=None*, *psn=None*)
Get path of VCF file from MB Obj and return the VCF file from either the MB mirror or the source.

get_variant_report(*psn=None*, *msn=None*)
Input a PSN or MSN and return a tab delimited set of variant data for the patient return var dict psn, msn, bsn disease

map_msn_psn(*pt_id*, *id_type*)
Map a MSN to PSN or PSN to MSN

Note: This function is going to be deprecated in favor of individual calls.

NOTE: This function is deprecated in favor of individual get_bsn, get_psn, get_msn class of functions. Given a patient ID (either MSN or PSN) and a type val, output corresponding MSN / PSN mapping.

Note: If requesting an MSN as output, you will recieve an array of data since there can be more than one MSN / PSN. When requesting a PSN from an MSN, you will recieve only one value.

Args: pt_id (str): Patient ID as either a MSN or PSN id_type (str): Type of ID input ('msn' | 'psn').

Returns: result (str): Corresponding MSN or PSN that maps to the input MSN or PSN.

```
>>> print(map_msn_psn('14420','psn'))
[u'MSN44180']
```

matchbox_dump(*filename=None*)
Dump a parsed MATCHBox dataset.

Call to the API and make a JSON file that can later be loaded in, rather than making an API call and reprocessing. Useful for quicker look ups as the API call can be very, very slow with such a large DB.

Note: This is a different dataset than the raw dump.

Args: filename (str): Filename to use for output. Default filename is: 'mb_obj_<date_generated>.json'

Returns: file: MATCHBox API JSON file.

class matchbox_api_utils.**TreatmentArms**(*config_file=None*, *url=None*, *creds=None*, *json_db='sys_default'*, *load_raw=None*, *make_raw=False*)

Bases: object

NCI-MATCH Treatment Arms and aMOIs Class

get_exclusion_disease(*armid*)
Input an arm ID and return a list of exclusionary diseases for the arm, if there are any. Otherwise return None.

Args: armid (str): Full identifier of the arm to be queried.

Returns: List of exclusionary diseases for the arm, or None if there aren't any.

Example:

```
>>> get_exclusion_disease('EAY131-Z1A')
[u'Melanoma', u'Colorectal Cancer']
```

```
>>> get_exclusion_disease('EAY131-Y')
None
```

make_match_arms_db (*api_data*)

Make a database of MATCH Treatment Arms.

Read in raw API data and create pared down JSON structure that can be easily parsed later one.

map_amoi (*variant*)

Input a variant dict derived from some kind and return either an aMOI id in the form of Arm(ile). If variant is not an aMOI, returns 'None'.

Args:

variant (dict): Variant dict to annotate. Dict must have the following keys in order to be

valid:

- type : [snvs_indels, cnvs, fusions]
- oncomineVariantClass
- gene
- identifier (i.e. variant ID (COSM476))
- exon
- function

Not all variant types will have meaningful data for these fields, and so fields may be padded with a null char (e.g. '.', '-', 'NA', etc.).

Returns results (list): Arm ID(s) with (i)nclusion or (e)xclusion information.

Example:

```
>>> variant = { 'type' : 'snvs_indels', 'gene' : 'BRAF', 'identifier' :
↪'COSM476', 'exon' : '15', 'function' : 'missense' ,
↪'oncominevariantclass' : 'Hotspot' }
['EAY131-Y(e)', 'EAY131-P(e)', 'EAY131-N(e)', 'EAY131-H(i)']
```

map_drug_arm (*armid=None*, *drugname=None*)

Input an Arm ID or a drug name, and retun a tuple of arm, drugname, and ID. If no arm ID or drug name is input, will return a whole table of all arm data.

Args: armid (str): Offcial NCI-MATCH Arm ID in the form of EAY131-xxx (e.g. EAY131-Z1A). drugname (str): Drug name as registered in the NCI-MATCH subprotocols. Right now,

required to have the full string (e.g. 'MLN0128(TAK-228)' or, unfortunately, 'Sunitinib malate (SU011248 L-malate)'), but will work on a regex to help later on.

Returns: List of tuples or None.

Example:

```
>>> map_drug_arm(armid='EAY131-Z1A')
(u'EAY131-Z1A', 'Binimetinib', u'788187')
```

ta_json_dump(*amois_filename=None*, *ta_filename=None*)

Dump the TreatmentArms data to a JSON file that can be easily loaded downstream. We will make both the treatment arms object, as well as the amois lookup table object.

Args: amois_filename (str): Name of aMOI lookup JSON file. Default: amoi_lookup_<datestring>.json ta_filename (str): Name of TA object JSON file Default: ta_obj_<datestring>.json

Returns: ta_obj_<date>.json amois_lookup_<date>.json

matchbox_api_utils

Description of this package coming soon. For now consult the individual files in the 'bin' directory.

- Note: Don't use pip to install... won't run the post installer script!

v0.1dev, 6/13/2017 – Initial release

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index