scrapenhl2 Documentation

Release 0.4.1

Muneeb Alam

Feb 08, 2018

Contents

CHAPTER 1

Table of Contents

1.1 Scrape

The scrapenhl2.scrape module contains methods useful for scraping.

1.1.1 Useful examples

Updating data:

```
from scrapenhl2.scrape import autoupdate
autoupdate.autoupdate()
```

Get the season schedule:

```
from scrapenhl2.scrape import schedules
schedules.get_season_schedule(2017)
```

Convert between player ID and player name:

```
from scrapenhl2.scrape import players
pname = 'Alex Ovechkin'
players.player_as_id(pname)

pid = 8471214
players.player_as_str(pid)
```

There's much more, and feel free to submit pull requests with whatever you find useful.

1.1.2 Methods

The functions in these modules are organized pretty logically under the module names.

Autoupdate

This module contains methods for automatically scraping and parsing games.

`scrapenhl2.scrape.autoupdate.`**`autoupdate`**`(`*season=None*`)`

Run this method to update local data. It reads the schedule file for given season and scrapes and parses previously unscraped games that have gone final or are in progress. Use this for 2010 or later.

Parameters **`season`** – int, the season. If None (default), will do current season

Returns nothing

`scrapenhl2.scrape.autoupdate.`**`delete_game_html`**`(`*season*, *game*`)`

Deletes html files. HTML files are used for live game charts, but deleted in favor of JSONs when games go final.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns nothing

`scrapenhl2.scrape.autoupdate.`**`read_final_games`**`(`*games*, *season*`)`

Parameters

- **`games`** –
- **`season`** –

Returns

`scrapenhl2.scrape.autoupdate.`**`read_inprogress_games`**`(`*inprogressgames*, *season*`)`

Saves these games to file via html (for toi) and json (for pbp)

Parameters **`inprogressgames`** – list of int

Returns

Events

This module contains methods related to PBP events.

`scrapenhl2.scrape.events.`**`convert_event`**`(`*event*`)`

Converts to a more convenient, standardized name (see get_event_dictionary)

Parameters **`event`** – str, the event name

Returns str, shortened event name

`scrapenhl2.scrape.events.`**`event_setup`**`()`

Loads event dictionary into memory

Returns nothing

`scrapenhl2.scrape.events.`**`get_event_dictionary`**`()`

Returns the abbreviation: long name event mapping (in lowercase)

Returns dict of str:str

`scrapenhl2.scrape.events.`**`get_event_longname`**

A method for translating event abbreviations to full names (for pbp matching)

Parameters **`eventname`** – str, the event name

Returns the non-abbreviated event name

Games

This module contains methods related to scraping games.

`scrapenhl2.scrape.games.`**`find_recent_games`**(*team1*, *team2=None*, *limit=1*)

A convenience function that lists the most recent in progress or final games for specified team(s)

Parameters

- **`team1`** – str, a team
- **`team2`** – str, a team (optional)
- **`limit`** – How many games to return

Returns df with relevant rows

`scrapenhl2.scrape.games.`**`get_player_5v5_log_filename`**(*season*)

Gets the filename for the season's player log file. Includes 5v5 CF, CA, TOI, and more.

Parameters **`season`** – int, the season

Returns str, /scrape/data/other/[season]_player_log.feather

`scrapenhl2.scrape.games.`**`most_recent_game_id`**(*team1*, *team2*)

A convenience function to get the most recent game (this season) between two teams.

Parameters

- **`team1`** – str, a team
- **`team2`** – str, a team

Returns int, a game number

General helpers

This module contains general helper methods. None of these methods have dependencies on other scrapenhl2 modules.

`scrapenhl2.scrape.general_helpers.`**`add_sim_scores`**(*df*, *name*)

Adds fuzzywuzzy's token set similarity scores to provded dataframe

Parameters

- **`df`** – pandas dataframe with column Name
- **`name`** – str, name to compare to

Returns df with an additional column SimScore

`scrapenhl2.scrape.general_helpers.`**`anti_join`**(*df1*, *df2*, ***kwargs*)

Anti-joins two dataframes.

Parameters

- **`df1`** – dataframe
- **`df2`** – dataframe
- **`kwargs`** – keyword arguments as passed to pd.DataFrame.merge (except for 'how'). Specifically, need join keys.

Returns dataframe

`scrapenhl2.scrape.general_helpers.`**`check_number`**(*obj*)

A helper method to check if obj is int, float, np.int64, etc. This is frequently needed, so is helpful.

Parameters **`obj`** – the object to check the type

Returns bool

`scrapenhl2.scrape.general_helpers.`**`check_number_last_first_format`**(*name*)

Checks if specified name looks like "8 Ovechkin, Alex"

Parameters **`name`** – str

Returns bool

`scrapenhl2.scrape.general_helpers.`**`check_types`**(*obj*)

A helper method to check if obj is int, float, np.int64, or str. This is frequently needed, so is helpful.

Parameters **`obj`** – the object to check the type

Returns bool

`scrapenhl2.scrape.general_helpers.`**`fill_join`**(*df1*, *df2*, ***kwargs*)

Uses data from df2 to fill in missing values from df1. Helpful when you have to join using multiple data sources. Preserves data order. Won't work when joining introduces duplicates.

Parameters

- **`df1`** – dataframe
- **`df2`** – dataframe
- **`kwargs`** – keyword arguments as passed to pd.DataFrame.merge (except for 'how' and 'suffixes')

Returns dataframe

`scrapenhl2.scrape.general_helpers.`**`flip_first_last`**(*name*)

Changes Ovechkin, Alex to Alex Ovechkin. Also changes to title case.

Parameters **`name`** – str

Returns str, flipped if applicable

`scrapenhl2.scrape.general_helpers.`**`fuzzy_match_player`**(*name_provided*, *names*, *minimum_similarity=50*)

This method checks similarity between each entry in names and the name_provided using token set matching and returns the entry that matches best. Returns None if no similarity is greater than minimum_similarity. (See e.g. http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/)

Parameters

- **`name_provided`** – str, name to look for
- **`names`** – list (or ndarray, or similar) of
- **`minimum_similarity`** – int from 0 to 100, minimum similarity. If all are below this, returns None.

Returns str, string in names that best matches name_provided

`scrapenhl2.scrape.general_helpers.`**`get_initials`**(*pname*)

Splits name on spaces and returns first letter from each part.

Parameters **`pname`** – str, player name

Returns str, player initials

`scrapenhl2.scrape.general_helpers.`**`get_lastname`**`(`*pname*`)`

Splits name on first space and returns second part.

> **Parameters** **`pname`** – str, player name
>
> **Returns** str, player last name

`scrapenhl2.scrape.general_helpers.`**`infer_season_from_date`**

Looks at a date and infers the season based on that: Year-1 if month is Aug or before; returns year otherwise.

> **Parameters** **`date`** – str, YYYY-MM-DD
>
> **Returns** int, the season. 2007-08 would be 2007.

`scrapenhl2.scrape.general_helpers.`**`intervals`**`(`*lst*, *interval_pct=10*`)`

A method that divides list into intervals and returns tuples indicating each interval mark. Useful for giving updates when cycling through games.

> **Parameters**
>
> - **`lst`** – lst to divide
> - **`interval_pct`** – int, pct for each interval to represent. e.g. 10 means it will mark every 10%.
>
> **Returns** a list of tuples of (index, value)

`scrapenhl2.scrape.general_helpers.`**`log_exceptions`**`(`*fn*`)`

A decorator that wraps the passed in function and logs exceptions should one occur

> **Parameters** **`function`** – the function
>
> **Returns** nothing

`scrapenhl2.scrape.general_helpers.`**`melt_helper`**`(`*df*, ***kwargs*`)`

Earlier versions of pandas do not support pd.DataFrame.melt. This helps to bridge the gap. It first tries df.melt, and if that doesn't work, it uses pd.melt.

> **Parameters**
>
> - **`df`** – dataframe
> - **`kwargs`** – arguments to pd.melt or pd.DataFrame.melt.
>
> **Returns** melted dataframe

`scrapenhl2.scrape.general_helpers.`**`mmss_to_secs`**`(`*strtime*`)`

Converts time from mm:ss to seconds

> **Parameters** **`strtime`** – str, mm:ss
>
> **Returns** int

`scrapenhl2.scrape.general_helpers.`**`once_per_second`**`(`*fn*, *calls_per_second=1*`)`

A decorator that sleeps for one second after executing the function. Used when scraping NHL site. This also means all functions that access the internet sleep for a second.

> **Parameters** **`fn`** – the function
>
> **Returns** nothing

`scrapenhl2.scrape.general_helpers.`**`period_contribution`**`(`*x*`)`

Turns period–1, 2, 3, OT, etc–into # of seconds elapsed in game until start. :param x: str or int, 1, 2, 3, etc :return: int, number of seconds elapsed until start of specified period `scrapenhl2.scrape.general_helpers.`**`print_and_log`**`(`*message*, *level='info'*, *print_and_log=True*`)`

A helper method that prints message to console and also writes to log with specified level.

Parameters

- **`message`** – str, the message
- **`level`** – str, the level of log: info, warn, error, critical
- **`print_and_log`** – bool. If False, logs only.

Returns nothing

`scrapenhl2.scrape.general_helpers.`**`remove_leading_number`**`(`*string*`)`

Will convert 8 Alex Ovechkin to Alex Ovechkin, or Alex Ovechkin to Alex Ovechkin

Parameters **`string`** – a string

Returns string without leading numbers

`scrapenhl2.scrape.general_helpers.`**`start_logging`**`()`

Clears out logging folder, and starts the log in this folder

`scrapenhl2.scrape.general_helpers.`**`try_to_access_dict`**`(`*base_dct*, **keys*, ***kwargs*`)`

A helper method that accesses base_dct using keys, one-by-one. Returns None if a key does not exist.

Parameters

- **`base_dct`** – dict, a dictionary
- **`keys`** – str, int, or other valid dict keys
- **`kwargs`** – can specify default using kwarg default_return=0, for example.

Returns obj, base_dct[key1][key2][key3]… or None if a key is not in the dictionary

`scrapenhl2.scrape.general_helpers.`**`try_url_n_times`**`(`*url*, *timeout=5*, *n=5*`)`

A helper method that tries to access given url up to five times, returning the page.

Parameters

- **`url`** – str, the url to access
- **`timeout`** – int, number of secs to wait before timeout. Default 5.
- **`n`** – int, the max number of tries. Default 5.

Returns bytes

Organization

This module contains paths to folders.

`scrapenhl2.scrape.organization.`**`check_create_folder`**`(`**args*`)`

A helper method to create a folder if it doesn't exist already

Parameters **`args`** – list of str, the parts of the filepath. These are joined together with the base directory

Returns nothing

`scrapenhl2.scrape.organization.`**`get_base_dir`**`()`

Returns the base directory of this package (one directory up from this file)

Returns str, the base directory

`scrapenhl2.scrape.organization.`**`get_other_data_folder`**`()`

Returns the folder containing other data

Returns str, /scrape/data/other/

`scrapenhl2.scrape.organization.`**`get_parsed_data_folder`**`()`

Returns the folder containing parsed data

Returns str, /scrape/data/parsed/

`scrapenhl2.scrape.organization.`**`get_raw_data_folder`**`()`

Returns the folder containing raw data

Returns str, /scrape/data/raw/

`scrapenhl2.scrape.organization.`**`get_season_parsed_pbp_folder`**`(`*season*`)`

Returns the folder containing parsed pbp for given season

Parameters **`season`** – int, current season

Returns str, /scrape/data/parsed/pbp/[season]/

`scrapenhl2.scrape.organization.`**`get_season_parsed_toi_folder`**`(`*season*`)`

Returns the folder containing parsed toi for given season

Parameters **`season`** – int, current season

Returns str, /scrape/data/raw/toi/[season]/

`scrapenhl2.scrape.organization.`**`get_season_raw_pbp_folder`**`(`*season*`)`

Returns the folder containing raw pbp for given season

Parameters **`season`** – int, current season

Returns str, /scrape/data/raw/pbp/[season]/

`scrapenhl2.scrape.organization.`**`get_season_raw_toi_folder`**`(`*season*`)`

Returns the folder containing raw toi for given season

Parameters **`season`** – int, current season

Returns str, /scrape/data/raw/toi/[season]/

`scrapenhl2.scrape.organization.`**`get_season_team_pbp_folder`**`(`*season*`)`

Returns the folder containing team pbp logs for given season

Parameters **`season`** – int, current season

Returns str, /scrape/data/teams/pbp/[season]/

`scrapenhl2.scrape.organization.`**`get_season_team_toi_folder`**`(`*season*`)`

Returns the folder containing team toi logs for given season

Parameters **`season`** – int, current season

Returns str, /scrape/data/teams/toi/[season]/

`scrapenhl2.scrape.organization.`**`get_team_data_folder`**`()`

Returns the folder containing team log data

Returns str, /scrape/data/teams/

`scrapenhl2.scrape.organization.`**`organization_setup`**`()`

Creates other folder if need be

Returns nothing

Players

This module contains methods related to individual player info.

`scrapenhl2.scrape.players.`**`check_default_player_id`**`(`*playername*`)`

E.g. For Mike Green, I should automatically assume we mean 8471242 (WSH/DET), not 8468436. Returns None if not in dict. Ideally improve code so this isn't needed.

Parameters **`playername`** – str

Returns int, or None

`scrapenhl2.scrape.players.`**`generate_player_ids_file`**`()`

Creates a dataframe with these columns:

- ID: int, player ID
- Name: str, player name
- DOB: str, date of birth
- Hand: char, R or L
- Pos: char, one of C/R/L/D/G

It will be populated with Alex Ovechkin to start. :return: nothing

`scrapenhl2.scrape.players.`**`generate_player_log_file`**`()`

Run this when no player log file exists already. This is for getting the datatypes right. Adds Alex Ovechkin in Game 1 vs Pittsburgh in 2016-2017.

Returns nothing

`scrapenhl2.scrape.players.`**`get_player_handedness`**

Retrieves handedness of player

Parameters **`player`** – str or int, the player name or ID

Returns str, player hand (L or R)

`scrapenhl2.scrape.players.`**`get_player_ids_file`**`()`

Returns the player information file. This is stored as a feather file for fast read/write.

Returns /scrape/data/other/PLAYER_INFO.feather

`scrapenhl2.scrape.players.`**`get_player_info_from_url`**`(`*playerid*`)`

Gets ID, Name, Hand, Pos, DOB, Height, Weight, and Nationality from the NHL API.

Parameters **`playerid`** – int, the player id

Returns dict with player ID, name, handedness, position, etc

`scrapenhl2.scrape.players.`**`get_player_log_file`**`()`

Returns the player log file from memory.

Returns dataframe, the log

`scrapenhl2.scrape.players.`**`get_player_log_filename`**`()`

Returns the player log filename.

Returns str, /scrape/data/other/PLAYER_LOG.feather

`scrapenhl2.scrape.players.`**`get_player_position`**

Retrieves position of player

Parameters **`player`** – str or int, the player name or ID

Returns str, player position (e.g. C, D, R, L, G)

`scrapenhl2.scrape.players.`**`get_player_url`**(*playerid*)

Gets the url for a page containing information for specified player from NHL API.

Parameters **`playerid`** – int, the player ID

Returns str, https://statsapi.web.nhl.com/api/v1/people/{[}playerid]

`scrapenhl2.scrape.players.`**`player_as_id`**

A helper method. If player entered is int, returns that. If player is str, returns integer id of that player.

Parameters

- **`playername`** – int, or str, the player whose names you want to retrieve
- **`filterids`** – a tuple of players to choose from. Needs to be tuple else caching won't work.
- **`dob`** – yyyy-mm-dd, use to help when multiple players have the same name

Returns int, the player ID

`scrapenhl2.scrape.players.`**`player_as_str`**

A helper method. If player is int, returns string name of that player. Else returns standardized name.

Parameters

- **`playerid`** – int, or str, player whose name you want to retrieve
- **`filterids`** – a tuple of players to choose from. Needs to be tuple else caching won't work. Probably not needed but you can use this method to go from part of the name to full name, in which case it may be helpful.

Returns str, the player name

`scrapenhl2.scrape.players.`**`player_setup`**()

Loads team info file into memory.

Returns nothing

`scrapenhl2.scrape.players.`**`playerlst_as_id`**(*playerlst*, *exact=False*, *filterdf=None*)

Similar to player_as_id, but less robust against errors, and works on a list of players.

Parameters

- **`players`** – a list of int, or str, players whose IDs you want to retrieve.
- **`exact`** – bool. If True, looks for exact matches. If False, does not, using player_as_id (but will be slower)
- **`filterdf`** – df, a dataframe of players to choose from. Defaults to all.

Returns a list of int/float

`scrapenhl2.scrape.players.`**`playerlst_as_str`**(*players*, *filterdf=None*)

Similar to player_as_str, but less robust against errors, and works on a list of players

Parameters

- **`players`** – a list of int, or str, players whose names you want to retrieve
- **`filterdf`** – df, a dataframe of players to choose from. Defaults to all.

Returns a list of str

`scrapenhl2.scrape.players.`**`rescrape_player`**(*playerid*)

If you notice that a player name, position, etc, is outdated, call this method on their ID. It will re-scrape their data from the NHL API.

Parameters **`playerid`** – int, their ID. Also accepts str, their name.

Returns nothing

`scrapenhl2.scrape.players.`**`update_player_ids_file`**(*playerids*, *force_overwrite=False*)

Adds these entries to player IDs file if need be.

Parameters

- **`playerids`** – a list of IDs
- **`force_overwrite`** – bool. If True, will re-scrape data for all player ids. If False, only new ones.

Returns nothing

`scrapenhl2.scrape.players.`**`update_player_ids_from_page`**(*pbp*)

Reads the list of players listed in the game file and adds to the player IDs file if they are not there already.

Parameters **`pbp`** – json, the raw pbp

Returns nothing

`scrapenhl2.scrape.players.`**`update_player_log_file`**(*playerids*, *seasons*, *games*, *teams*, *statuses*)

Updates the player log file with given players. The player log file notes which players played in which games and whether they were scratched or played.

Parameters

- **`playerids`** – int or str or list of int
- **`seasons`** – int, the season, or list of int the same length as playerids
- **`games`** – int, the game, or list of int the same length as playerids
- **`teams`** – str or int, the team, or list of int the same length as playerids
- **`statuses`** – str, or list of str the same length as playerids

Returns nothing

`scrapenhl2.scrape.players.`**`update_player_logs_from_page`**(*pbp*, *season*, *game*)

Takes the game play by play and adds players to the master player log file, noting that they were on the roster for this game, which team they played for, and their status (P for played, S for scratch).

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`pbp`** – json, the pbp of the game

Returns nothing

`scrapenhl2.scrape.players.`**`write_player_ids_file`**(*df*)

Writes the given dataframe to disk as the player ids mapping.

Parameters **`df`** – pandas dataframe, player ids file

Returns nothing

`scrapenhl2.scrape.players.`**`write_player_log_file`**`(`*df*`)`

Writes the given dataframe to file as the player log filename

Parameters **`df`** – pandas dataframe

Returns nothing

Schedules

This module contains methods related to season schedules.

`scrapenhl2.scrape.schedules.`**`attach_game_dates_to_dateframe`**`(`*df*`)`

Takes dataframe with Season and Game columns and adds a Date column (for that game)

Parameters **`df`** – dataframe

Returns dataframe with one more column

`scrapenhl2.scrape.schedules.`**`generate_season_schedule_file`**`(`*season*, *force_overwrite=True*`)`

Reads season schedule from NHL API and writes to file.

The output contains the following columns:

- Season: int, the season
- Date: str, the dates
- Game: int, the game id
- Type: str, the game type (for preseason vs regular season, etc)
- Status: str, e.g. Final
- Road: int, the road team ID
- RoadScore: int, number of road team goals
- RoadCoach str, 'N/A' when this function is run (edited later with road coach name)
- Home: int, the home team ID
- HomeScore: int, number of home team goals
- HomeCoach: str, 'N/A' when this function is run (edited later with home coach name)
- Venue: str, the name of the arena
- Result: str, 'N/A' when this function is run (edited accordingly later from PoV of home team: W, OTW, SOL, etc)
- PBPStatus: str, 'Not scraped' when this function is run (edited accordingly later)
- TOIStatus: str, 'Not scraped' when this function is run (edited accordingly later)

Parameters

- **`season`** – int, the season
- **`force_overwrite`** – bool. If True, generates entire file from scratch. If False, only redoes when not Final previously.

Returns Nothing

`scrapenhl2.scrape.schedules.`**`get_current_season`**`()`

Returns the current season.

Returns The current season variable (generated at import from _get_current_season)

`scrapenhl2.scrape.schedules.`**`get_game_data_from_schedule`**

This is a helper method that uses the schedule file to isolate information for current game (e.g. teams involved, coaches, venue, score, etc.)

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns dict of game data

`scrapenhl2.scrape.schedules.`**`get_game_date`**`(`*season*, *game*`)`

Returns the date of this game

Parameters

- **`season`** – int, the game
- **`game`** – int, the season

Returns str

`scrapenhl2.scrape.schedules.`**`get_game_result`**`(`*season*, *game*`)`

Returns the result of this game for home team (e.g. W, SOL)

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns int, the score

`scrapenhl2.scrape.schedules.`**`get_game_status`**`(`*season*, *game*`)`

Returns the status of this game (e.g. Final, In Progress)

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns int, the score

`scrapenhl2.scrape.schedules.`**`get_home_score`**`(`*season*, *game*`)`

Returns the home score from this game

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns int, the score

`scrapenhl2.scrape.schedules.`**`get_home_team`**`(`*season*, *game*, *returntype='id'*`)`

Returns the home team from this game

Parameters

- **`season`** – int, the game
- **`game`** – int, the season
- **`returntype`** – str, 'id' or 'name'

Returns float or str, depending on returntype

`scrapenhl2.scrape.schedules.`**`get_road_score`**(*season*, *game*)

Returns the road score from this game

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns int, the score

`scrapenhl2.scrape.schedules.`**`get_road_team`**(*season*, *game*, *returntype='id'*)

Returns the road team from this game

Parameters

- **`season`** – int, the game
- **`game`** – int, the season
- **`returntype`** – str, 'id' or 'name'

Returns float or str, depending on returntype

`scrapenhl2.scrape.schedules.`**`get_season_schedule`**(*season*)

Gets the the season's schedule file from memory.

Parameters **`season`** – int, the season

Returns dataframe (originally from /scrape/data/other/[season]_schedule.feather)

`scrapenhl2.scrape.schedules.`**`get_season_schedule_filename`**(*season*)

Gets the filename for the season's schedule file

Parameters **`season`** – int, the season

Returns str, /scrape/data/other/[season]_schedule.feather

`scrapenhl2.scrape.schedules.`**`get_season_schedule_url`**(*season*)

Gets the url for a page containing all of this season's games (Sep 1 to Jun 26) from NHL API.

Parameters **`season`** – int, the season

Returns str, https://statsapi.web.nhl.com/api/v1/schedule?startDate={[}season{]}-09-01&endDate={[}season+1{]}-06-25

`scrapenhl2.scrape.schedules.`**`get_team_games`**(*season=None*, *team=None*, *startdate=None*, *enddate=None*)

Returns list of games played by team in season.

Just calls get_team_schedule with the provided arguments, returning the series of games from that dataframe.

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team
- **`startdate`** – str or None
- **`enddate`** – str or None

Returns series of games

`scrapenhl2.scrape.schedules.`**`get_team_schedule`**(*season=None*, *team=None*, *startdate=None*, *enddate=None*)

Gets the schedule for given team in given season. Or if startdate and enddate are specified, searches between those dates. If season and startdate (and/or enddate) are specified, searches that season between those dates.

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team
- **`startdate`** – str, YYYY-MM-DD
- **`enddate`** – str, YYYY-MM-DD

Returns dataframe

`scrapenhl2.scrape.schedules.`**`get_teams_in_season`**(*season*)

Returns all teams that have a game in the schedule for this season

Parameters **`season`** – int, the season

Returns set of team IDs

`scrapenhl2.scrape.schedules.`**`schedule_setup`**()

Reads current season and schedules into memory.

Returns nothing

`scrapenhl2.scrape.schedules.`**`write_season_schedule`**(*df*, *season*, *force_overwrite*)

A helper method that writes the season schedule file to disk (in feather format for fast read/write)

Parameters

- **`df`** – the season schedule datafraome
- **`season`** – the season
- **`force_overwrite`** – bool. If True, overwrites entire file. If False, only redoes when not Final previously.

Returns Nothing

Manipulate schedules

This module contains methods related to generating and manipulating schedules.

`scrapenhl2.scrape.manipulate_schedules.`**`update_schedule_with_coaches`**(*pbp*, *season*, *game*)

Uses the PbP to update coach info for this game.

Parameters

- **`pbp`** – json, the pbp for this game
- **`season`** – int, the season
- **`game`** – int, the game

Returns nothing

`scrapenhl2.scrape.manipulate_schedules.`**`update_schedule_with_pbp_scrape`**(*season*, *game*)

Updates the schedule file saying that specified game's pbp has been scraped.

Parameters

- **season** – int, the season
- **game** – int, the game, or list of ints

Returns updated schedule

`scrapenhl2.scrape.manipulate_schedules.`**`update_schedule_with_result`**(*season*, *game*, *result*)

Updates the season schedule file with game result (which are listed 'N/A' at schedule generation)

Parameters

- **season** – int, the season
- **game** – int, the game
- **result** – str, the result from home team perspective

Returns

`scrapenhl2.scrape.manipulate_schedules.`**`update_schedule_with_result_using_pbp`**(*pbp*, *season*, *game*)

Uses the PbP to update results for this game.

Parameters

- **pbp** – json, the pbp for this game
- **season** – int, the season
- **game** – int, the game

Returns nothing

`scrapenhl2.scrape.manipulate_schedules.`**`update_schedule_with_toi_scrape`**(*season*, *game*)

Updates the schedule file saying that specified game's toi has been scraped.

Parameters

- **season** – int, the season
- **game** – int, the game, or list of int

Returns nothing

Scrape play by play

This module contains methods for scraping pbp.

`scrapenhl2.scrape.scrape_pbp.`**`get_game_from_url`**(*season*, *game*)

Gets the page containing information for specified game from NHL API.

Parameters

- **season** – int, the season
- **game** – int, the game

Returns str, the page at the url

`scrapenhl2.scrape.scrape_pbp.`**`get_game_pbplog_filename`**`(`*season*, *game*`)`

Returns the filename of the parsed pbp html game pbp

Parameters

- **`season`** – int, current season
- **`game`** – int, game

Returns str, /scrape/data/raw/pbp/[season]/[game].html

`scrapenhl2.scrape.scrape_pbp.`**`get_game_pbplog_url`**`(`*season*, *game*`)`

Gets the url for a page containing pbp information for specified game from HTML tables.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

:return : str, e.g. http://www.nhl.com/scores/htmlreports/20072008/PL020001.HTM

`scrapenhl2.scrape.scrape_pbp.`**`get_game_raw_pbp_filename`**`(`*season*, *game*`)`

Returns the filename of the raw pbp folder

Parameters

- **`season`** – int, current season
- **`game`** – int, game

Returns str, /scrape/data/raw/pbp/[season]/[game].zlib

`scrapenhl2.scrape.scrape_pbp.`**`get_game_url`**`(`*season*, *game*`)`

Gets the url for a page containing information for specified game from NHL API.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns str, https://statsapi.web.nhl.com/api/v1/game/{[}season{]}0{[}game{]}/feed/live

`scrapenhl2.scrape.scrape_pbp.`**`get_raw_html_pbp`**`(`*season*, *game*`)`

Loads the html file containing this game's play by play from disk.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns str, the html pbp

`scrapenhl2.scrape.scrape_pbp.`**`get_raw_pbp`**`(`*season*, *game*`)`

Loads the compressed json file containing this game's play by play from disk.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns json, the json pbp

`scrapenhl2.scrape.scrape_pbp.`**`save_raw_html_pbp`**`(`*page*, *season*, *game*`)`

Takes the bytes page containing html pbp information and saves as such

Parameters

- **page** – bytes
- **season** – int, the season
- **game** – int, the game

Returns nothing

`scrapenhl2.scrape.scrape_pbp.`**`save_raw_pbp`**(*page*, *season*, *game*)

Takes the bytes page containing pbp information and saves to disk as a compressed zlib.

Parameters

- **page** – bytes. str(page) would yield a string version of the json pbp
- **season** – int, the season
- **game** – int, the game

Returns nothing

`scrapenhl2.scrape.scrape_pbp.`**`scrape_game_pbp`**(*season*, *game*, *force_overwrite=False*)

This method scrapes the pbp for the given game.

Parameters

- **season** – int, the season
- **game** – int, the game
- **force_overwrite** – bool. If file exists already, won't scrape again

Returns bool, False if not scraped, else True

`scrapenhl2.scrape.scrape_pbp.`**`scrape_game_pbp_from_html`**(*season*, *game*, *force_overwrite=True*)

This method scrapes the html pbp for the given game. Use for live games.

Parameters

- **season** – int, the season
- **game** – int, the game
- **force_overwrite** – bool. If file exists already, won't scrape again

Returns bool, False if not scraped, else True

`scrapenhl2.scrape.scrape_pbp.`**`scrape_pbp_setup`**()

Creates raw pbp folders if need be

Returns

`scrapenhl2.scrape.scrape_pbp.`**`scrape_season_pbp`**(*season*, *force_overwrite=False*)

Scrapes and parses pbp from the given season.

Parameters

- **season** – int, the season
- **force_overwrite** – bool. If true, rescrapes all games. If false, only previously unscraped ones

Returns nothing

Parse play by play

This module contains methods for parsing PBP.

`scrapenhl2.scrape.parse_pbp.`**`get_5v5_corsi_pm`**(*season*, *game*, *cfca=None*)

Returns a dataframe from home team perspective. Each row is a Corsi event, with time and note of whether it's positive or negative for home team.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`cfca`** – str, or None. If you specify 'cf', returns CF only. For CA, use 'ca'. None returns CF - CA.

Returns dataframe with columns Time and HomeCorsi

`scrapenhl2.scrape.parse_pbp.`**`get_game_parsed_pbp_filename`**(*season*, *game*)

Returns the filename of the parsed pbp folder

Parameters

- **`season`** – int, current season
- **`game`** – int, game

Returns str, /scrape/data/parsed/pbp/[season]/[game].zlib

`scrapenhl2.scrape.parse_pbp.`**`get_parsed_pbp`**(*season*, *game*)

Loads the compressed json file containing this game's play by play from disk.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns json, the json pbp

`scrapenhl2.scrape.parse_pbp.`**`parse_game_pbp`**(*season*, *game*, *force_overwrite=False*)

Reads the raw pbp from file, updates player IDs, updates player logs, and parses the JSON to a pandas DF and writes to file. Also updates team logs accordingly.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`force_overwrite`** – bool. If True, will execute. If False, executes only if file does not exist yet.

Returns True if parsed, False if not

`scrapenhl2.scrape.parse_pbp.`**`parse_game_pbp_from_html`**(*season*, *game*, *force_overwrite=False*)

Reads the raw pbp from file, updates player IDs, updates player logs, and parses the JSON to a pandas DF and writes to file. Also updates team logs accordingly.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

- **force_overwrite** – bool. If True, will execute. If False, executes only if file does not exist yet.

Returns True if parsed, False if not

`scrapenhl2.scrape.parse_pbp.`**`parse_pbp_setup`**`()`

Creates parsed pbp folders if need be

Returns nothing

`scrapenhl2.scrape.parse_pbp.`**`parse_season_pbp`**`(`*season*, *force_overwrite=False*`)`

Parses pbp from the given season.

Parameters

- **season** – int, the season
- **force_overwrite** – bool. If true, parses all games. If false, only previously unparsed ones

Returns nothing

`scrapenhl2.scrape.parse_pbp.`**`read_events_from_page`**`(`*rawpbp*, *season*, *game*`)`

This method takes the json pbp and returns a pandas dataframe with the following columns:

- Index: int, index of event
- Period: str, period of event. In regular season, could be 1, 2, 3, OT, or SO. In playoffs, 1, 2, 3, 4, 5…
- MinSec: str, m:ss, time elapsed in period
- Time: int, time elapsed in game
- Event: str, the event name
- Team: int, the team id. Note that this is switched to blocked team for blocked shots to ease Corsi calculations.
- Actor: int, the acting player id. Switched with recipient for blocks (see above)
- ActorRole: str, e.g. for faceoffs there is a "Winner" and "Loser". Switched with recipient for blocks (see above)
- Recipient: int, the receiving player id. Switched with actor for blocks (see above)
- RecipientRole: str, e.g. for faceoffs there is a "Winner" and "Loser". Switched with actor for blocks (see above)
- X: int, the x coordinate of event (or NaN)
- Y: int, the y coordinate of event (or NaN)
- Note: str, additional notes, which may include penalty duration, assists on a goal, etc.

Parameters

- **rawpbp** – json, the raw json pbp
- **season** – int, the season
- **game** – int, the game

Returns pandas dataframe, the pbp in a nicer format

`scrapenhl2.scrape.parse_pbp.`**`save_parsed_pbp`**`(`*pbp*, *season*, *game*`)`

Saves the pandas dataframe containing pbp information to disk as an HDF5.

Parameters

- **pbp** – df, a pandas dataframe with the pbp of the game
- **season** – int, the season
- **game** – int, the game

Returns nothing

Scrape TOI

This module contains methods for scraping TOI.

scrapenhl2.scrape.scrape_toi.**get_game_raw_toi_filename**(*season*, *game*)

Returns the filename of the raw toi folder

Parameters

- **season** – int, current season
- **game** – int, game

Returns str, /scrape/data/raw/toi/[season]/[game].zlib

scrapenhl2.scrape.scrape_toi.**get_home_shiftlog_filename**(*season*, *game*)

Returns the filename of the parsed toi html home shifts

Parameters

- **season** – int, the season
- **game** – int, the game

Returns str, /scrape/data/raw/pbp/[season]/[game]H.html

scrapenhl2.scrape.scrape_toi.**get_home_shiftlog_url**(*season*, *game*)

Gets the url for a page containing shift information for specified game from HTML tables for home team.

Parameters

- **season** – int, the season
- **game** – int, the game

:return : str, e.g. http://www.nhl.com/scores/htmlreports/20072008/TH020001.HTM

scrapenhl2.scrape.scrape_toi.**get_raw_html_toi**(*season*, *game*, *homeroad*)

Loads the html file containing this game's toi from disk.

Parameters

- **season** – int, the season
- **game** – int, the game
- **homeroad** – str, 'H' for home or 'R' for road

Returns str, the html toi

scrapenhl2.scrape.scrape_toi.**get_raw_toi**(*season*, *game*)

Loads the compressed json file containing this game's shifts from disk.

Parameters

- **season** – int, the season
- **game** – int, the game

Returns dict, the json shifts

`scrapenhl2.scrape.scrape_toi.`**`get_road_shiftlog_filename`**(*season*, *game*)

Returns the filename of the parsed toi html road shifts

Parameters

- **`season`** – int, current season
- **`game`** – int, game

Returns str, /scrape/data/raw/pbp/[season]/[game]H.html

`scrapenhl2.scrape.scrape_toi.`**`get_road_shiftlog_url`**(*season*, *game*)

Gets the url for a page containing shift information for specified game from HTML tables for road team.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

:return : str, e.g. http://www.nhl.com/scores/htmlreports/20072008/TV020001.HTM

`scrapenhl2.scrape.scrape_toi.`**`get_shift_url`**(*season*, *game*)

Gets the url for a page containing shift information for specified game from NHL API.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

:return : str, http://www.nhl.com/stats/rest/shiftcharts?cayenneExp=gameId={[}season{]}0{[}game]

`scrapenhl2.scrape.scrape_toi.`**`save_raw_toi`**(*page*, *season*, *game*)

Takes the bytes page containing shift information and saves to disk as a compressed zlib.

Parameters

- **`page`** – bytes. str(page) would yield a string version of the json shifts
- **`season`** – int, the season
- **`game`** – int, the game

Returns nothing

`scrapenhl2.scrape.scrape_toi.`**`save_raw_toi_from_html`**(*page*, *season*, *game*, *homeroad*)

Takes the bytes page containing shift information and saves to disk as html.

Parameters

- **`page`** – bytes. str(page) would yield a string version of the json shifts
- **`season`** – int, he season
- **`game`** – int, the game
- **`homeroad`** – str, ‘H’ or ‘R’

Returns nothing

`scrapenhl2.scrape.scrape_toi.`**`scrape_game_toi`**(*season*, *game*, *force_overwrite=False*)

This method scrapes the toi for the given game.

Parameters

- **`season`** – int, the season

- **game** – int, the game
- **force_overwrite** – bool. If file exists already, won't scrape again

Returns nothing

`scrapenhl2.scrape.scrape_toi.`**`scrape_game_toi_from_html`**(*season*, *game*, *force_overwrite=True*)

This method scrapes the toi html logs for the given game.

Parameters

- **season** – int, the season
- **game** – int, the game
- **force_overwrite** – bool. If file exists already, won't scrape again

Returns nothing

`scrapenhl2.scrape.scrape_toi.`**`scrape_season_toi`**(*season*, *force_overwrite=False*)

Scrapes and parses toi from the given season.

Parameters

- **season** – int, the season
- **force_overwrite** – bool. If true, rescrapes all games. If false, only previously unscraped ones

Returns nothing

`scrapenhl2.scrape.scrape_toi.`**`scrape_toi_setup`**()

Creates raw toi folders if need be

Returns

Parse TOI

This module contains methods for parsing TOI.

`scrapenhl2.scrape.parse_toi.`**`get_game_parsed_toi_filename`**(*season*, *game*)

Returns the filename of the parsed toi folder

Parameters

- **season** – int, current season
- **game** – int, game

Returns str, /scrape/data/parsed/toi/[season]/[game].zlib

`scrapenhl2.scrape.parse_toi.`**`get_melted_home_road_5v5_toi`**(*season*, *game*)

Reads parsed TOI for this game, filters for 5v5 TOI, and melts from wide to long on player

Parameters

- **season** – int, the season
- **game** – int, the game

Returns (home_df, road_df), each with columns Time, PlayerID, and Team (which will be H or R)

`scrapenhl2.scrape.parse_toi.`**`get_parsed_toi`**(*season*, *game*)

Loads the compressed json file containing this game's shifts from disk.

Parameters

- **season** – int, the season
- **game** – int, the game

Returns json, the json shifts

`scrapenhl2.scrape.parse_toi.`**`parse_game_toi`**(*season*, *game*, *force_overwrite=False*)

Parses TOI from json for this game

Parameters

- **season** – int, the season
- **game** – int, the game
- **force_overwrite** – bool. If True, will execute. If False, executes only if file does not exist yet.

Returns nothing

`scrapenhl2.scrape.parse_toi.`**`parse_game_toi_from_html`**(*season*, *game*, *force_overwrite=False*)

Parses TOI from the html shift log from this game.

Parameters

- **season** – int, the season
- **game** – int, the game
- **force_overwrite** – bool. If True, will execute. If False, executes only if file does not exist yet.

Returns nothing

`scrapenhl2.scrape.parse_toi.`**`parse_season_toi`**(*season*, *force_overwrite=False*)

Parses toi from the given season. Final games covered only.

Parameters

- **season** – int, the season
- **force_overwrite** – bool. If true, parses all games. If false, only previously unparsed ones

Returns

`scrapenhl2.scrape.parse_toi.`**`parse_toi_setup`**()

Creates parsed toi folders if need be

Returns

`scrapenhl2.scrape.parse_toi.`**`read_shifts_from_html_pages`**(*rawtoi1*, *rawtoi2*, *teamid1*, *teamid2*, *season*, *game*)

Aggregates information from two html pages given into a dataframe with one row per second and one col per player.

Parameters

- **rawtoi1** – str, html page of shift log for team id1
- **rawtoi2** – str, html page of shift log for teamid2
- **teamid1** – int, team id corresponding to rawtoi1
- **teamid2** – int, team id corresponding to rawtoi1
- **season** – int, the season

- **game** – int, the game

Returns dataframe

`scrapenhl2.scrape.parse_toi.`**`read_shifts_from_page`**(*rawtoi*, *season*, *game*)

Turns JSON shift start-ends into TOI matrix with one row per second and one col per player

Parameters

- **rawtoi** – dict, json from NHL API
- **season** – int, the season
- **game** – int, the game

Returns dataframe

`scrapenhl2.scrape.parse_toi.`**`save_parsed_toi`**(*toi*, *season*, *game*)

Saves the pandas dataframe containing shift information to disk as an HDF5.

Parameters

- **toi** – df, a pandas dataframe with the shifts of the game
- **season** – int, the season
- **game** – int, the game

Returns nothing

Team information

This module contains methods related to the team info files.

`scrapenhl2.scrape.team_info.`**`add_team_to_info_file`**(*teamid*)

In case we come across teams that are not in the default list (1-110), use this method to add them to the file.

Parameters **teamid** – int, the team ID

Returns (tid, tabbrev, tname)

`scrapenhl2.scrape.team_info.`**`generate_team_ids_file`**(*teamids=None*)

Reads all team id URLs and stores information to disk. Has the following information:

- ID: int
- Abbreviation: str (three letters)
- Name: str (full name)

Parameters **teamids** – iterable of int. Tries to access team ids as listed in teamids. If not, goes from 1-110.

Returns nothing

`scrapenhl2.scrape.team_info.`**`get_team_colordict`**()

Get the team color dictionary

Returns a dictionary of IDs to tuples of hex colors

`scrapenhl2.scrape.team_info.`**`get_team_colors`**(*team*)

Returns primary and secondary color for this team.

Parameters **team** – str or int, the team

Returns tuple of hex colors

`scrapenhl2.scrape.team_info.`**`get_team_info_file`**`()`

Returns the team information dataframe from memory. This is stored as a feather file for fast read/write.

Returns dataframe from /scrape/data/other/TEAM_INFO.feather

`scrapenhl2.scrape.team_info.`**`get_team_info_filename`**`()`

Returns the team information filename

Returns str, /scrape/data/other/TEAM_INFO.feather

`scrapenhl2.scrape.team_info.`**`get_team_info_from_url`**`(`*teamid*`)`

Pulls ID, abbreviation, and name from the NHL API.

Parameters **`teamid`** – int, the team ID

Returns (id, abbrev, name)

`scrapenhl2.scrape.team_info.`**`get_team_info_url`**`(`*teamid*`)`

Gets the team url from the NHL API.

Parameters **`teamid`** – int, the team ID

Returns str, http://statsapi.web.nhl.com/api/v1/teams/{[}teamid]

`scrapenhl2.scrape.team_info.`**`team_as_id`**

A helper method. If team entered is int, returns that. If team is str, returns integer id of that team.

Parameters **`team`** – int, or str

Returns int, the team ID

`scrapenhl2.scrape.team_info.`**`team_as_str`**

A helper method. If team entered is str, returns that. If team is int, returns string name of that team.

Parameters

- **`team`** – int, or str
- **`abbreviation`** – bool, whether to return 3-letter abbreviation or full name

Returns str, the team name

`scrapenhl2.scrape.team_info.`**`team_setup`**`()`

This method loads the team info df into memory

Returns nothing

`scrapenhl2.scrape.team_info.`**`write_team_info_file`**`(`*df*`)`

Writes the team information file. This is stored as a feather file for fast read/write.

Parameters **`df`** – the (team information) dataframe to write to file

Returns nothing

Teams

This module contains method related to team logs.

`scrapenhl2.scrape.teams.`**`get_team_pbp`**`(`*season*, *team*`)`

Returns the pbp of given team in given season across all games.

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team abbreviation.

Returns df, the pbp of given team in given season

`scrapenhl2.scrape.teams.`**`get_team_pbp_filename`**`(`*season*, *team*`)`

Returns filename of the PBP log for this team and season

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team abbreviation.

Returns

`scrapenhl2.scrape.teams.`**`get_team_toi`**`(`*season*, *team*`)`

Returns the toi of given team in given season across all games.

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team abbreviation.

Returns df, the toi of given team in given season

`scrapenhl2.scrape.teams.`**`get_team_toi_filename`**`(`*season*, *team*`)`

Returns filename of the TOI log for this team and season

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team abbreviation.

Returns

`scrapenhl2.scrape.teams.`**`team_setup`**`()`

Creates team log-related folders.

Returns nothing

`scrapenhl2.scrape.teams.`**`update_team_logs`**`(`*season*, *force_overwrite=False*, *force_games=None*`)`

This method looks at the schedule for the given season and writes pbp for scraped games to file. It also adds the strength at each pbp event to the log. It only includes games that have both PBP *and* TOI.

Parameters

- **`season`** – int, the season
- **`force_overwrite`** – bool, whether to generate from scratch
- **`force_games`** – None or iterable of games to force_overwrite specifically

Returns nothing

`scrapenhl2.scrape.teams.`**`write_team_pbp`**`(`*pbp*, *season*, *team*`)`

Writes the given pbp dataframe to file.

Parameters

- **`pbp`** – df, the pbp of given team in given season
- **`season`** – int, the season
- **`team`** – int or str, the team abbreviation.

Returns nothing

`scrapenhl2.scrape.teams.`**`write_team_toi`**(*toi*, *season*, *team*)

Writes team TOI log to file

Parameters

- **`toi`** – df, team toi for this season
- **`season`** – int, the season
- **`team`** – int or str, the team abbreviation.

Returns

1.2 Manipulate

The scrapenhl2.manipulate module contains methods useful for scraping.

1.2.1 Useful examples

Add on-ice players to a file:

```
from scrapenhl.manipulate import add_onice_players as onice
onice.add_players_to_file('/Users/muneebalam/Downloads/zone_entries.csv', 'WSH', time_
↪format='elapsed')
# Will output zone_entries_on-ice.csv in Downloads, with WSH players and opp players␣
↪on-ice listed.
```

See documentation below for more information and additional arguments to add_players_to_file.

1.2.2 Methods

General

`scrapenhl2.manipulate.manipulate.`**`add_score_adjustment_to_team_pbp`**(*df*)

Adds AdjFF and AdjFA

Parameters **`df`** – dataframe

Returns dataframe with extra columns

`scrapenhl2.manipulate.manipulate.`**`convert_to_all_combos`**(*df*, *fillval=0*, **args*)

This method takes a dataframe and makes sure all possible combinations of given arguments are present. For example, if you want df to have all combos of P1 and P2, it will create a dataframe with all possible combos, left join existing dataframe onto that, and return that df. Uses fillval to fill *all* non-key columns.

Parameters

- **`df`** – the pandas dataframe
- **`fillval`** – obj, the value with which to fill. Default fill is 0
- **`args`** – str, column names, or tuples of combinations of column names

Returns df with all combos of columns specified

`scrapenhl2.manipulate.manipulate.`**`count_by_keys`**(*df*, **args*)

A convenience method that isolates specified columns in the dataframe and gets counts. Drops when keys have NAs.

Parameters

- **`df`** – dataframe
- **`args`** – str, column names in dataframe

Returns df, dataframe with each of args and an additional column, Count

`scrapenhl2.manipulate.manipulate.`**`filter_for_corsi`**(*pbp*)
Filters given dataframe for goal, shot, miss, and block events

Parameters **`pbp`** – a dataframe with column Event

Returns pbp, filtered for corsi events

`scrapenhl2.manipulate.manipulate.`**`filter_for_event_types`**(*pbp*, *eventtype*)
Filters given dataframe for event type(s) specified only.

Parameters

- **`pbp`** – dataframe. Need a column titled Event
- **`eventtype`** – str or iterable of str, e.g. Goal, Shot, etc

Returns dataframe, filtered

`scrapenhl2.manipulate.manipulate.`**`filter_for_fenwick`**(*pbp*)
Filters given dataframe for SOG only.

Parameters **`pbp`** – dataframe. Need a column titled Event

Returns dataframe. Only rows where Event == 'Goal' or Event == 'Shot'

`scrapenhl2.manipulate.manipulate.`**`filter_for_five_on_five`**(*df*)
Filters given dataframe for 5v5 rows

Parameters **`df`** – dataframe, columns HomeStrength + RoadStrength or TeamStrength + OppStrength

Returns dataframe

`scrapenhl2.manipulate.manipulate.`**`filter_for_goals`**(*pbp*)
Filters given dataframe for goals only.

Parameters **`pbp`** – dataframe. Need a column titled Event

Returns dataframe. Only rows where Event == 'Goal'

`scrapenhl2.manipulate.manipulate.`**`filter_for_sog`**(*pbp*)
Filters given dataframe for SOG only.

Parameters **`pbp`** – dataframe. Need a column titled Event

Returns dataframe. Only rows where Event == 'Goal' or Event == 'Shot'

`scrapenhl2.manipulate.manipulate.`**`filter_for_team`**(*pbp*, *team*)
Filters dataframe for rows where Team == team

Parameters

- **`pbp`** – dataframe. Needs to have column Team
- **`team`** – int or str, team ID or name

Returns dataframe with rows filtered `scrapenhl2.manipulate.manipulate.`**`generate_5v5_player_log`**`(`*season*`)`

Takes the play by play and adds player 5v5 info to the master player log file, noting TOI, CF, etc. This takes awhile because it has to calculate TOICOMP. :param season: int, the season :return: nothing

`scrapenhl2.manipulate.manipulate.`**`generate_player_toion_toioff`**`(`*season*`)`

Generates TOION and TOIOFF at 5v5 for each player in this season. :param season: int, the season :return: df with columns Player, TOION, TOIOFF, and TOI60.

`scrapenhl2.manipulate.manipulate.`**`generate_toicomp`**`(`*season*`)`

Generates toicomp at a player-game level :param season: int, the season :return: df,

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_game_cfca`**`(`*season*, *team*`)`

Gets CFON, CAON, CFOFF, and CAOFF by game for given team in given season.

Parameters

- **`season`** – int, the season
- **`team`** – int, team id

Returns df with game, player, CFON, CAON, CFOFF, and CAOFF

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_game_gfga`**`(`*season*, *team*`)`

Gets GFON, GAON, GFOFF, and GAOFF by game for given team in given season.

Parameters

- **`season`** – int, the season
- **`team`** – int, team id

Returns df with game, player, GFON, GAON, GFOFF, and GAOFF

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_game_shift_startend`**`(`*season*, *team*`)`

Generates shift starts and ends for shifts that start and end at 5v5–OZ, DZ, NZ, OtF.

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team

Returns dataframe with shift starts and ends

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_game_toi`**`(`*season*, *team*`)`

Gets TOION and TOIOFF by game and player for given team in given season. :param season: int, the season :param team: int, team id :return: df with game, player, TOION, and TOIOFF

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_game_toicomp`**`(`*season*, *team*`)`

Calculates data for QoT and QoC at a player-game level for given team in given season. :param season: int, the season :param team: int, team id :return: df with game, player,

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_log`**`(`*season*, *force_create=False*`)`

Parameters

- **`season`** – int, the season
- **`force_create`** – bool, create from scratch even if it exists?

Returns

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_log_filename`**`(`*season*`)`

Parameters **`season`** – int, the season

Returns

`scrapenhl2.manipulate.manipulate.`**`get_5v5_player_season_toi`**(*season*, *team*)

Gets TOION and TOIOFF by player for given team in given season. :param season: int, the season :param team: int, team id :return: df with game, player, TOION, and TOIOFF

`scrapenhl2.manipulate.manipulate.`**`get_directions_for_xy_for_game`**(*season*, *game*)

It doesn't seem like there are rules for whether positive X in XY event locations corresponds to offensive zone events, for example. Best way is to use fields in the the json.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game

Returns dict indicating which direction home team is attacking by period

`scrapenhl2.manipulate.manipulate.`**`get_directions_for_xy_for_season`**(*season*, *team*)

Gets directions for team specified using get_directions_for_xy_for_game

Parameters

- **`season`** – int, the season
- **`team`** – int or str, the team

Returns dataframe

`scrapenhl2.manipulate.manipulate.`**`get_game_h2h_corsi`**(*season*, *games*, *cfca=None*)

This method gets H2H Corsi at 5v5 for the given game(s).

Parameters

- **`season`** – int, the season
- **`games`** – int, the game, or list of int, the games
- **`cfca`** – str, or None. If you specify 'cf', returns CF only. For CA, use 'ca'. None returns CF - CA.

Returns a df with [P1, P1Team, P2, P2Team, CF, CA, C+/-]. Entries will be duplicated, as with get_game_h2h_toi.

`scrapenhl2.manipulate.manipulate.`**`get_game_h2h_toi`**(*season*, *games*)

This method gets H2H TOI at 5v5 for the given game.

Parameters

- **`season`** – int, the season
- **`games`** – int, the game, or list of int, the games

Returns a df with [P1, P1Team, P2, P2Team, TOI]. Entries will be duplicated (one with given P as P1, another as P2)

`scrapenhl2.manipulate.manipulate.`**`get_line_combos`**(*season*, *game*, *homeroad='H'*)

Returns a df listing the 5v5 line combinations used in this game for specified team, and time they each played together

Parameters

- **`season`** – int, the game
- **`game`** – int, the season
- **`homeroad`** – str, 'H' for home or 'R' for road

Returns pandas dataframe with columns P1, P2, P3, Secs. May contain duplicates scrapenhl2.manipulate.manipulate.**get_micah_score_adjustment**()

See http://hockeyviz.com/txt/senstats

Returns dataframe: HomeScoreDiff, HomeFFWeight, and HomeFAWeight

scrapenhl2.manipulate.manipulate.**get_pairings**(*season*, *game*, *homeroad='H'*)

Returns a df listing the 5v5 pairs used in this game for specified team, and time they each played together

Parameters

- **season** – int, the game
- **game** – int, the season
- **homeroad** – str, 'H' for home or 'R' for road

Returns pandas dataframe with columns P1, P2, Secs. May contain duplicates

scrapenhl2.manipulate.manipulate.**get_pbp_events**(**args*, ***kwargs*)

A general method that yields a generator of dataframes of PBP events subject to given limitations.

Keyword arguments are applied as "or" conditions for each individual keyword (e.g. multiple teams) but as "and" conditions otherwise.

The non-keyword arguments are event types subject to "or" conditions:

- 'fac' or 'faceoff'
- 'shot' or 'sog' or 'save'
- 'hit'
- 'stop' or 'stoppage'
- 'block' or 'blocked shot'
- 'miss' or 'missed shot'
- 'give' or 'giveaway'
- 'take' or 'takeaway'
- 'penl' or 'penalty'
- 'goal'
- 'period end'
- 'period official'
- 'period ready'
- 'period start'
- 'game scheduled'
- 'gend' or 'game end'
- 'shootout complete'
- 'chal' or 'official challenge'
- 'post', which is not an officially designated event but will be searched for

Dataframes are returned season-by-season to save on memory. If you want to operate on all seasons, process this data before going to the next season.

Defaults to return all regular-season and playoff events by all teams.

Supported keyword arguments:

- add_on_ice: bool. If True, adds on-ice players for each time.
- players_on_ice: str or int, or list of them, player IDs or names of players on ice for event.
- players_on_ice_for: like players_on_ice, but players must be on ice for team that "did" event.
- players_on_ice_ag: like players_on_ice, but players must be on ice for opponent of team that "did" event.
- team, str or int, or list of them. Teams to filter for.
- team_for, str or int, or list of them. Team that committed event.
- team_ag, str or int, or list of them. Team that "received" event.
- home_team: str or int, or list of them. Home team.
- road_team: str or int, or list of them. Road team.
- start_date: str or date, will only return data on or after this date. YYYY-MM-DD
- end_date: str or date, will only return data on or before this date. YYYY-MM-DD
- start_season: int, will only return events in or after this season. Defaults to 2010-11.
- end_season: int, will only return events in or before this season. Defaults to current season.
- **season_type: int or list of int. 1 for preseason, 2 for regular, 3 for playoffs, 4 for ASG, 6 for Oly, 8 for WC.** Defaults to 2 and 3.
- start_game: int, start game. Applies only to start season. Game ID will be this, or greater.
- end_game: int, end game. Applies only to end season. Game ID will be this, or smaller.
- acting_player: str or int, or list of them, players who committed event (e.g. took a shot).
- receiving_player: str or int, or list of them, players who received event (e.g. took a hit).
- **strength_hr: tuples or list of them, e.g. (5, 5) or ((5, 5), (4, 4), (3, 3)). This is (Home, Road).** If neither strength_hr nor strength_to is specified, uses 5v5.
- **strength_to: tuples or list of them, e.g. (5, 5) or ((5, 5), (4, 4), (3, 3)). This is (Team, Opponent).** If neither strength_hr nor strength_to is specified, uses 5v5.
- score_diff: int or list of them, acceptable score differences (e.g. 0 for tied, (1, 2, 3) for up by 1-3 goals)
- start_time: int, seconds elapsed in game. Events returned will be after this.
- end_time: int, seconds elapsed in game. Events returned will be before this.

Parameters

- **`args`** – str, event types to search for (applied "OR", not "AND")
- **`kwargs`** – keyword arguments specifying filters (applied "AND", not "OR")

Returns df, a pandas dataframe

`scrapenhl2.manipulate.manipulate.`**`get_player_positions`**`()`

Use to get player positions :return: df with colnames ID and position

`scrapenhl2.manipulate.manipulate.`**`get_player_toi`**`(`*season*, *game*, *pos=None*, *homeroad='H'*`)`

Returns a df listing 5v5 ice time for each player for specified team.

Parameters

- **`season`** – int, the game
- **`game`** – int, the season

- **pos** – specify 'L', 'C', 'R', 'D' or None for all
- **homeroad** – str, 'H' for home or 'R' for road

Returns pandas df with columns Player, Secs

scrapenhl2.manipulate.manipulate.**get_player_toion_toioff_file**(*season*, *force_create=False*)

Parameters

- **season** – int, the season
- **force_create** – bool, should this be read from file if possible, or created from scratch

Returns

scrapenhl2.manipulate.manipulate.**get_player_toion_toioff_filename**(*season*)

Parameters **season** – int, the season

Returns

scrapenhl2.manipulate.manipulate.**get_toicomp_file**(*season*, *force_create=False*)

If you want to rewrite the TOI60 file, too, then run get_player_toion_toioff_file with force_create=True before running this method. :param season: int, the season :param force_create: bool, should this be read from file if possible, or created from scratch :return:

scrapenhl2.manipulate.manipulate.**get_toicomp_filename**(*season*)

Parameters **season** – int, the season

Returns

scrapenhl2.manipulate.manipulate.**infer_zones_for_faceoffs**(*df*, *directions*, *xcol='X'*, *ycol='Y'*, *timecol='Time'*, *focus_team=None*, *season=None*, *faceoffs=True*)

Inferring zones for events from XY is hard–this method takes are of that by referencing against the JSON's notes on which team went which direction in which period.

This method notes several different zones for faceoffs:

- OL (offensive zone, left)
- OR (offensive zone, right)
- NOL (neutral zone, near offensive blueline, left)
- NOR (neutral zone, near offensive blueline, right)
- NDL (neutral zone, near defensive blueline, left)
- NDR (neutral zone, near defensive blueline, right)
- DL (defensive zone, left)
- DR (defensive zone, right)
- N (center ice)

This method can also handle non-faceoff events, using three zones

- N (neutral)
- O (offensive)

- D (defensive)

  Parameters

  - **`df`** – dataframe with columns Game, specified xcol, and specified ycol
  - **`directions`** – dataframe with columns Game, Period, and Direction ('left' or 'right')
  - **`xcol`** – str, the column containing X coordinates in df
  - **`ycol`** – str, the column containing Y coordinates in df
  - **`timecol`** – str, the column containing the time in seconds.
  - **`focus_team`** – int, str, or None. Directions are stored with home perspective. So specify focus team and will flip when focus_team is on the road. If None, does not do the extra home/road flip. Necessitates Season column in df.
  - **`season`** – int, the season
  - **`faceoffs`** – bool. If True will use the nine zones above. If False, only the three.

  Returns dataframe with extra column EventLoc

`scrapenhl2.manipulate.manipulate.`**`merge_onto_all_team_games_and_zero_fill`**(*df*, *season*, *team*)

A method that gets all team games from this season and left joins df onto it on game, then zero fills NAs. Makes sure you didn't miss any games and get NAs later.

Parameters

- **`df`** – dataframe with columns Game and PlayerID or Player
- **`season`** – int, the season
- **`team`** – int or str, the team

Returns dataframe

`scrapenhl2.manipulate.manipulate.`**`player_columns_to_name`**(*df*, *columns=None*)

Takes a dataframe and transforms specified columns of player IDs into names. If no columns provided, searches for defaults: H1, H2, H3, H4, H5, H6, HG (and same seven with R)

Parameters

- **`df`** – A dataframe
- **`columns`** – a list of strings, or None

Returns df, dataframe with same column names, but columns now names instead of IDs

`scrapenhl2.manipulate.manipulate.`**`save_5v5_player_log`**(*df*, *season*)

Parameters **`season`** – int, the season

Returns nothing

`scrapenhl2.manipulate.manipulate.`**`save_player_toion_toioff_file`**(*df*, *season*)

Parameters

- **`df`** –
- **`season`** – int, the season

Returns

`scrapenhl2.manipulate.manipulate.`**`save_toicomp_file`**(*df*, *season*)

Parameters

- **`df`** –
- **`season`** – int, the season

Returns

`scrapenhl2.manipulate.manipulate.`**`team_5v5_score_state_summary_by_game`**(*season*)

Uses the team TOI log to group by team and game and score state for this season. 5v5 only.

Parameters **`season`** – int, the season

Returns dataframe, grouped by team, strength, and game

`scrapenhl2.manipulate.manipulate.`**`team_5v5_shot_rates_by_score`**(*season*)

Uses the team TOI and PBP logs to group by team and game and score state for this season. 5v5 only.

Parameters **`season`** – int, the season

Returns dataframe, grouped by team, strength, and game. Also columns for TOI, CF, and CA

`scrapenhl2.manipulate.manipulate.`**`time_to_mss`**(*sectime*)

Converts a number of seconds to m:ss format

Parameters **`sectime`** – int, a number of seconds

Returns str, sectime in m:ss

Add on-ice players

Add on-ice players to a file by specifying filename and columns from which to infer time elapsed in game.

`scrapenhl2.manipulate.add_onice_players.`**`add_onice_players_to_df`**(*df*, *focus_team*, *season*, *gamecol*, *player_output='ids'*)

Uses the _Secs column in df, the season, and the gamecol to join onto on-ice players.

Parameters

- **`df`** – dataframe
- **`focus_team`** – str or int, team to focus on. Its players will be listed in first in sheet.
- **`season`** – int, the season
- **`gamecol`** – str, the column with game IDs
- **`player_output`** – str, use 'names' or 'nums' or 'ids'. Currently 'nums' is not supported.

Returns dataframe with team and opponent players

`scrapenhl2.manipulate.add_onice_players.`**`add_players_to_file`**(*filename*, *focus_team*, *season=None*, *gamecol='Game'*, *periodcol='Period'*, *timecol='Time'*, *time_format='elapsed'*, *update_data=False*, *player_output='names'*)

Adds names of on-ice players to the end of each line, and writes to file in the same folder as input file. Specifically, adds 1 second to the time in the spreadsheet and adds players who were on the ice at that time.

You cannot necessarily trust results when times coincide with stoppages–and it's worth checking faceoffs as well.

Parameters

- **`filename`** – str, the file to read. Will save output as this filename but ending in "on-ice.csv"
- **`focus_team`** – str or int, e.g. 'WSH' or 'WPG'
- **`season`** – int. For 2007-08, use 2007. Defaults to current season.
- **`gamecol`** – str. The column holding game IDs (e.g. 20001). By default, looks for column called "Game"
- **`periodcol`** – str. The column holding period number/name (1, 2, 3, 4 or OT, etc). By default: "Period"
- **`timecol`** – str. The column holding time in period in M:SS format.
- **`time_format`** – str, how to interpret timecol. Use 'elapsed' or 'remaining'. E.g. the start of a period is 0:00 with elapsed and 20:00 in remaining.
- **`update_data`** – bool. If True, will autoupdate() data for given season. If not, will not update game data. Use when file includes data from games not already scraped.
- **`player_output`** – str, use 'names' or 'nums'. Currently only supports 'names'

Returns nothing

`scrapenhl2.manipulate.add_onice_players.`**`add_times_to_file`**(*df*, *periodcol*, *timecol*, *time_format*)

Uses specified periodcol, timecol, and time_format col to calculate _Secs, time elapsed in game.

Parameters

- **`df`** – dataframe
- **`periodcol`** – str, the column that holds period name/number (1, 2, 3, 4 or OT, etc)
- **`timecol`** – str, the column that holds time in m:ss format
- **`time_format`** – use 'elapsed' (preferred) or 'remaining'. This refers to timecol: e.g. 120 secs elapsed in the 2nd period might be listed as 2:00 in timecol, or as 18:00.

Returns dataframe with extra column _Secs, time elapsed in game.

TOI and Corsi for combinations of players

This module contains methods for generating H2H data for games

`scrapenhl2.manipulate.combos.`**`get_game_combo_corsi`**(*season*, *game*, *player_n=2*, *cfca=None*, **hrcodes*)

This method gets H2H Corsi at 5v5 for the given game.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`player_n`** – int. E.g. 1 gives you a list of players and TOI, 2 gives you h2h, 3 gives you groups of 3, etc.

- **`cfca`** – str, or None. If you specify 'cf', returns CF only. For CA, use 'ca'. None returns CF - CA.
- **`hrcodes`** – to limit exploding joins, specify strings containing 'H' and 'R' and 'A', each of length player_n For example, if player_n=3, specify 'HHH' to only get home team player combos. If this is left unspecified, will do all combos, which can be problematic when player_n > 3. 'R' for road, 'H' for home, 'A' for all (both)

Returns a df with [P1, P1Team, P2, P2Team, TOI, etc]. Entries will be duplicated.

`scrapenhl2.manipulate.combos.`**`get_game_combo_toi`**(*season*, *game*, *player_n=2*, **hrcodes*)

This method gets H2H TOI at 5v5 for the given game.

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`player_n`** – int. E.g. 1 gives you a list of players and TOI, 2 gives you h2h, 3 gives you groups of 3, etc.
- **`hrcodes`** – to limit exploding joins, specify strings containing 'H' and 'R' and 'A', each of length player_n For example, if player_n=3, specify 'HHH' to only get home team player combos. If this is left unspecified, will do all combos, which can be problematic when player_n > 3. 'R' for road, 'H' for home, 'A' for all (both)

Returns a df with [P1, P1Team, P2, P2Team, TOI, etc]. Entries will be duplicated.

`scrapenhl2.manipulate.combos.`**`get_team_combo_corsi`**(*season*, *team*, *games*, *n_players=2*)

Gets combo Corsi for team for specified games

Parameters

- **`season`** – int, the season
- **`team`** – int or str, team
- **`games`** – int or iterable of int, games
- **`n_players`** – int. E.g. 1 gives you player TOI, 2 gives you 2-player group TOI, 3 makes 3-player groups, etc

Returns dataframe

`scrapenhl2.manipulate.combos.`**`get_team_combo_toi`**(*season*, *team*, *games*, *n_players=2*)

Gets 5v5 combo TOI for team for specified games

Parameters

- **`season`** – int, the season
- **`team`** – int or str, team
- **`games`** – int or iterable of int, games
- **`n_players`** – int. E.g. 1 gives you player TOI, 2 gives you 2-player group TOI, 3 makes 3-player groups, etc

Returns dataframe

1.3 Plot

The scrapenhl2.plot module contains methods useful for plotting.

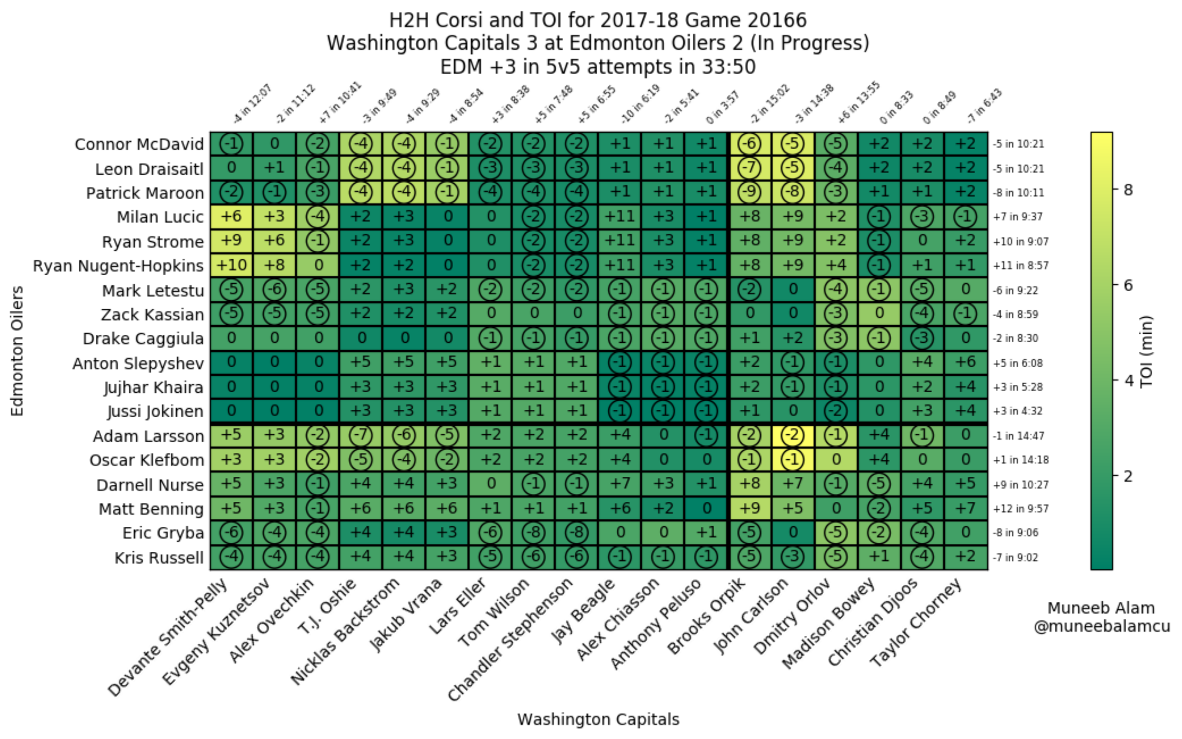
1.3.1 Useful examples

First, import:

```
from scrapenhl2.plot import *
```

Get the H2H for an in-progress game:

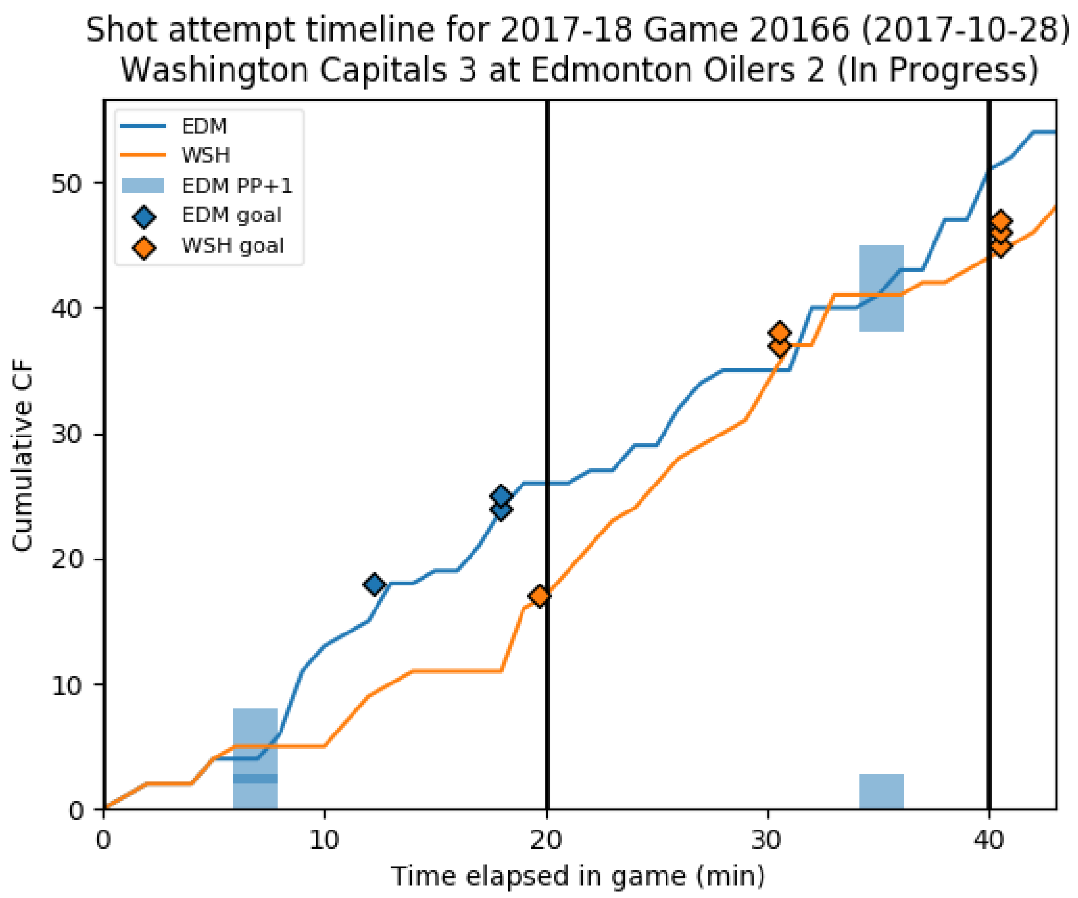
```
live_h2h('WSH', 'EDM')
```



The top left cell indicates Connor McDavid (row 1) faced Devante Smith-Pelly (column 1) for 1:50.
During that time, WSH out-attempted EDM by 1.

Get the Corsi timeline as well, but don't update data this time:

```
live_timeline('WSH', 'EDM', update=False)
```



Save the timeline of a memorable game to file:

```
game_timeline(2016, 30136, save_file='/Users/muneebalam/Desktop/WSH_TOR_G6.png')
```

More methods being added regularly.

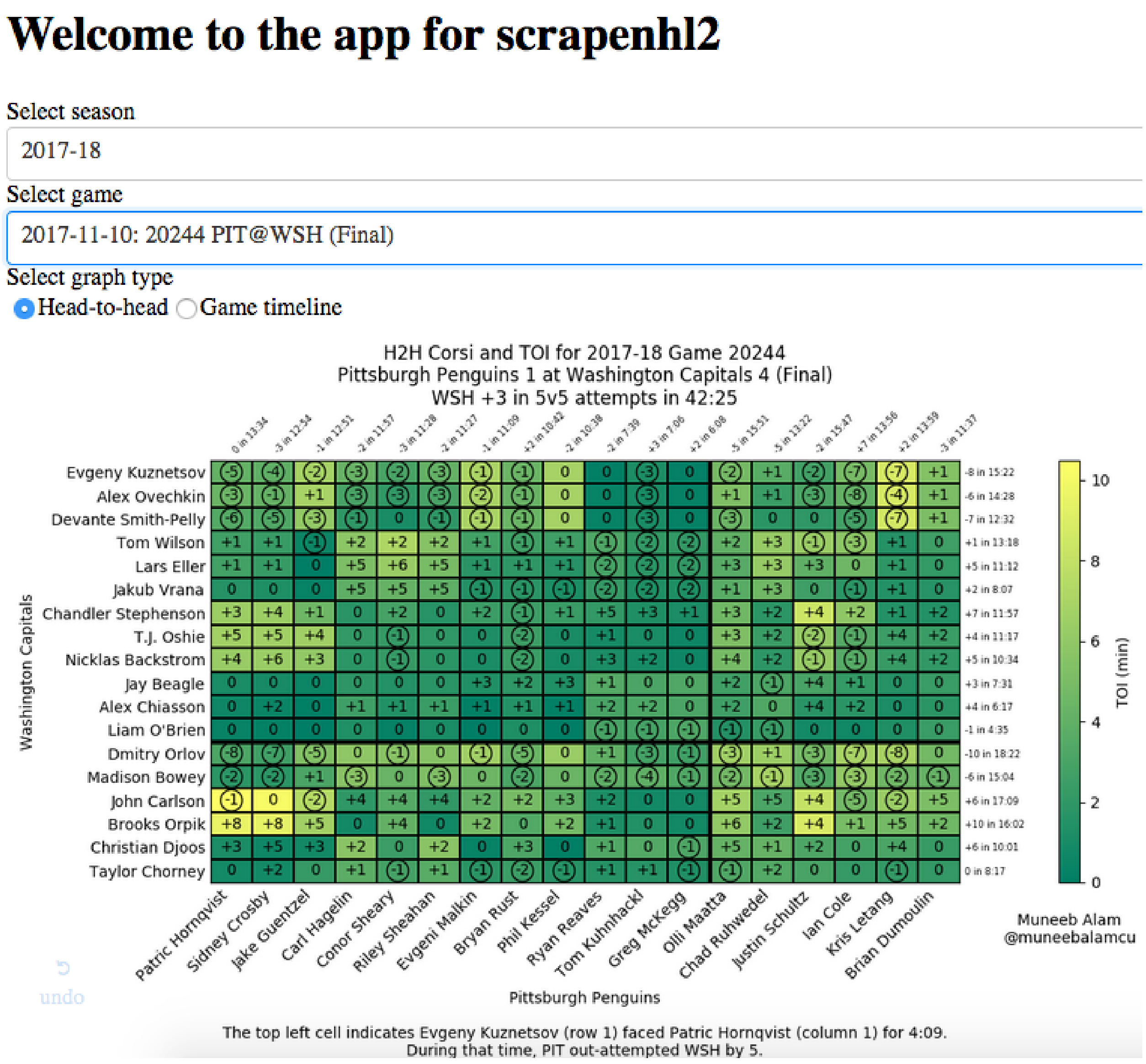
1.3.2 App

This package contains a lightweight app for browsing charts and doing some data manipulations.

Launch using:

```
import scrapenhl2.plot.app as app
app.browse_game_charts()
# app.browse_player_charts()
# app.browse_team_charts()
```

It will print a link in your terminal–follow it. The page looks something like this:



Welcome to the app for scrapenhl2

Select season

2017-18

Select game

2017-11-10: 20244 PIT@WSH (Final)

Select graph type

[✓ Head-to-head] Game timeline

H2H Corsi and TOI for 2017-18 Game 20244
Pittsburgh Penguins 1 at Washington Capitals 4 (Final)
WSH +3 in 5v5 attempts in 42:25

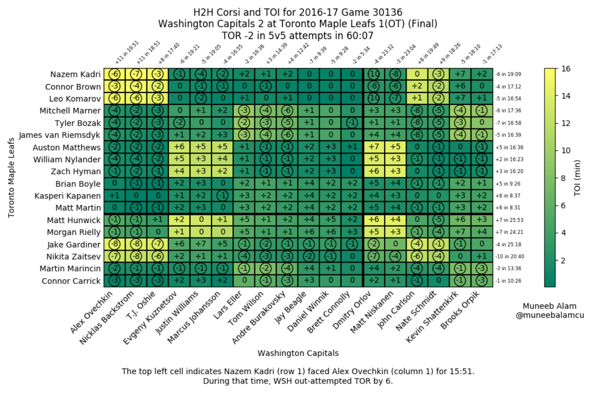
	Patric Hornqvist	Sidney Crosby	Jake Guentzel	Carl Hagelin	Conor Sheary	Riley Sheahan	Evgeni Malkin	Bryan Rust	Phil Kessel	Ryan Reaves	Tom Kuhnhackl	Greg McKegg	Olli Maatta	Chad Ruhwedel	Justin Schultz	Ian Cole	Kris Letang	Brian Dumoulin	
	0 in 13:34	-3 in 12:54	-1 in 12:51	-2 in 11:57	-3 in 11:28	-2 in 11:27	-1 in 11:09	+2 in 10:42	-2 in 10:38	-2 in 7:39	+3 in 7:06	+2 in 6:08	-5 in 15:51	-5 in 13:22	-2 in 15:47	+7 in 13:56	+2 in 13:59	-3 in 11:37	
Evgeny Kuznetsov	-5	-4	-2	-3	-2	-3	-1	-1	0	0	-3	0	-2	+1	-2	-7	-7	+1	-8 in 15:22
Alex Ovechkin	-3	-1	+1	-3	-3	-3	-2	-1	0	0	-3	0	+1	+1	-3	-8	-4	+1	-6 in 14:28
Devante Smith-Pelly	-6	-5	-3	-1	0	-1	-1	-1	0	0	-3	0	-3	0	0	-5	-7	+1	-7 in 12:32
Tom Wilson	+1	+1	-1	+2	+2	+2	+1	-1	+1	-1	-2	-2	+2	+3	-1	-3	+1	0	+1 in 13:18
Lars Eller	+1	+1	0	+5	+6	+5	+1	+1	+1	-2	-2	-2	+3	+3	+3	0	+1	0	+5 in 11:12
Jakub Vrana	0	0	0	+5	+5	+5	-1	-1	-1	-2	-2	-2	+1	+3	0	-1	+1	0	+2 in 8:07
Chandler Stephenson	+3	+4	+1	0	+2	0	+2	-1	+1	+5	+3	+1	+3	+2	+4	+2	+1	+2	+7 in 11:57
T.J. Oshie	+5	+5	+4	0	-1	0	0	-2	0	+1	0	0	+3	+2	-2	-1	+4	+2	+4 in 11:17
Nicklas Backstrom	+4	+6	+3	0	-1	0	0	-2	0	+3	+2	0	+4	+2	-1	-1	+4	+2	+5 in 10:34
Jay Beagle	0	0	0	0	0	0	+3	+2	+3	+1	0	0	+2	-1	+4	+1	0	0	+3 in 7:31
Alex Chiasson	0	+2	0	+1	+1	+1	+1	+1	+1	+2	+2	0	+2	0	+4	+2	0	0	+4 in 6:17
Liam O'Brien	0	0	0	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	-1 in 4:35
Dmitry Orlov	-8	-7	-5	0	-1	0	-1	-5	0	+1	-3	-1	-3	+1	-3	-7	-8	0	-10 in 18:22
Madison Bowey	-2	-2	+1	-3	0	-3	0	-2	0	-2	-4	-1	-2	-1	-3	-3	-2	-1	-6 in 15:04
John Carlson	-1	0	-2	+4	+4	+4	+2	+2	+3	+2	0	0	+5	+5	+4	-5	-2	+5	+6 in 17:09
Brooks Orpik	+8	+8	+5	0	+4	0	+2	0	+2	+1	0	0	+6	+2	+4	+1	+5	+2	+10 in 16:02
Christian Djoos	+3	+5	+3	+2	0	+2	0	+3	0	+1	0	-1	+5	+1	+2	0	+4	0	+6 in 10:01
Taylor Chorney	0	+2	0	+1	-1	+1	-1	-2	-1	+1	+1	-1	-1	+2	0	0	-1	0	0 in 8:17

The top left cell indicates Evgeny Kuznetsov (row 1) faced Patric Hornqvist (column 1) for 4:09.
During that time, PIT out-attempted WSH by 5.

The dropdowns also allow you to search–just start typing.

1.3.3 Methods (games)

Game H2H



This module contains methods for creating a game H2H chart.

scrapenhl2.plot.game_h2h.**game_h2h**(*season*, *game*, *save_file=None*)
: Creates the grid H2H charts seen on @muneebalamcu

  Parameters

  - **season** – int, the season
  - **game** – int, the game
  - **save_file** – str, specify a valid filepath to save to file. If None, merely shows on screen.

  Returns nothing

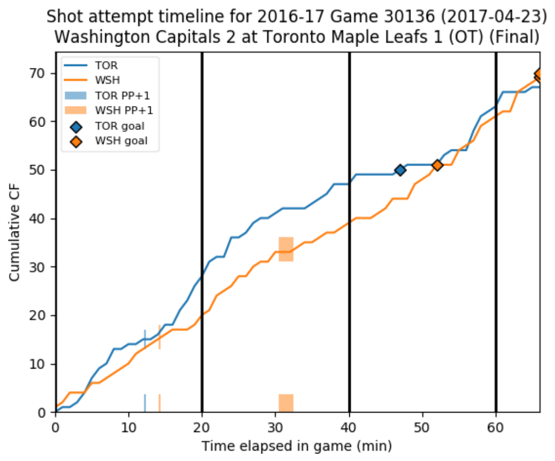
scrapenhl2.plot.game_h2h.**live_h2h**(*team1*, *team2*, *update=True*, *save_file=None*)
: A convenience method that updates data then displays h2h for most recent game between specified tams.

  Parameters

  - **team1** – str or int, team
  - **team2** – str or int, other team
  - **update** – bool, should data be updated first?
  - **save_file** – str, specify a valid filepath to save to file. If None, merely shows on screen.

  Returns nothing

Corsi timeline



This module has methods for creating a game corsi timeline.

`scrapenhl2.plot.game_timeline.`**`game_timeline`**(*season*, *game*, *save_file=None*)

Creates a shot attempt timeline as seen on @muneebalamcu

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`save_file`** – str, specify a valid filepath to save to file. If None, merely shows on screen. Specify 'fig' to return the figure

Returns nothing, or the figure

`scrapenhl2.plot.game_timeline.`**`get_goals_for_timeline`**(*season*, *game*, *homeroad*, *granularity='sec'*)

Returns a list of goal times

Parameters

- **`season`** – int, the season
- **`game`** – int, the game
- **`homeroad`** – str, 'H' for home and 'R' for road
- **`granularity`** – can respond in minutes ('min'), or seconds ('sec'), elapsed in game

Returns a list of int, seconds elapsed

`scrapenhl2.plot.game_timeline.`**`live_timeline`**(*team1*, *team2*, *update=True*, *save_file=None*)

A convenience method that updates data then displays timeline for most recent game between specified tams.

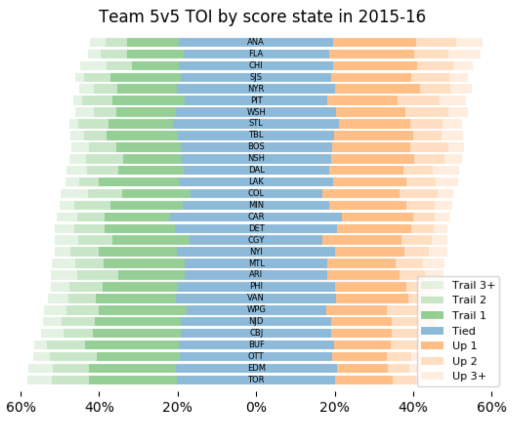
Parameters

- **`team1`** – str or int, team
- **`team2`** – str or int, other team
- **`update`** – bool, should data be updated first?
- **`save_file`** – str, specify a valid filepath to save to file. If None, merely shows on screen.

Returns nothing

1.3.4 Methods (teams)

Team TOI by score



This module contains methods for making a stacked bar graph indicating how much TOI each team spends in score states.

`scrapenhl2.plot.team_score_state_toi.`**`get_score_state_graph_title`**(*season*)

Parameters **`season`** – int, the season

Returns

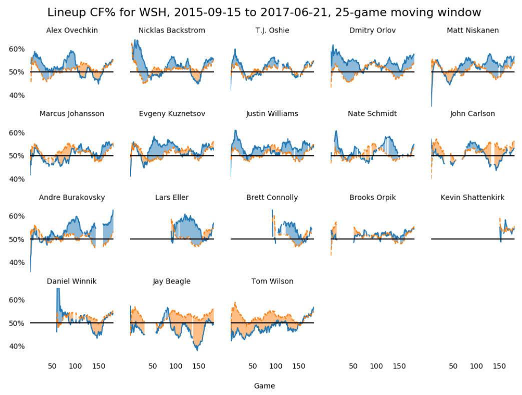
`scrapenhl2.plot.team_score_state_toi.`**`score_state_graph`**(*season*)

Generates a horizontal stacked bar graph showing how much 5v5 TOI each team has played in each score state for given season.

Parameters **`season`** – int, the season

Returns

Team lineup CF%



This module contains methods to generate a graph showing player CF%. 18 little graphs, 1 for each of 18 players.

`scrapenhl2.plot.team_lineup_cf.`**`team_lineup_cf_graph`**(*team*, *****kwargs***)

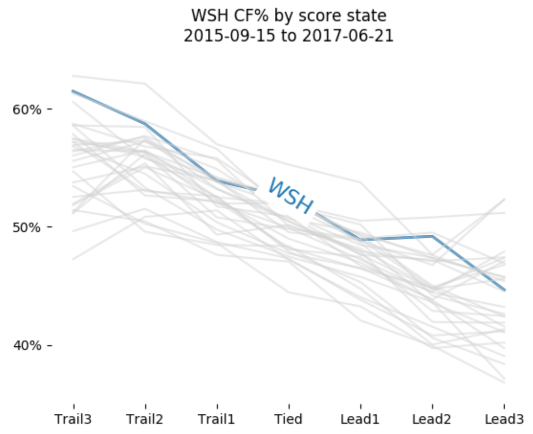
This method builds a 4x5 matrix of rolling CF% line graphs. The left 4x3 matrix are forward lines and the top-right 3x2 are defense pairs.

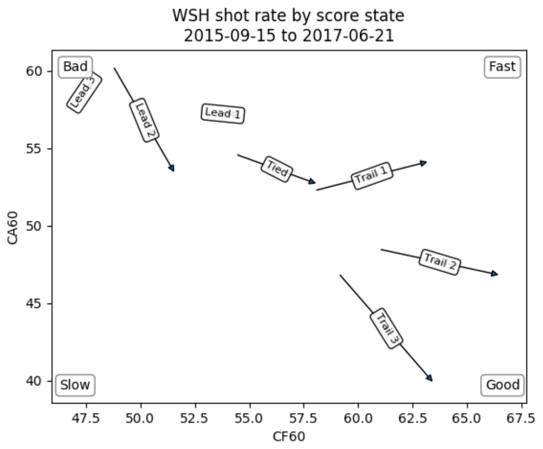
Parameters

- **team** – str or id, team to build this graph for
- **kwargs** – need to specify the following as iterables of names: l1, l2, l3, l4, p1, p2, p3. Three players for each of the 'l's and two for each of the 'p's.

Returns figure, or nothing

Team shot rates by score





This module creates a scatterplot for specified team with shot attempt rates versus league median from down 3 to up 3.

`scrapenhl2.plot.team_score_shot_rate.`**`team_score_shot_rate_parallel`**(*team*, *startseason*, *endseason=None*, *save_file=None*)

Parameters

- **team** –
- **startseason** –
- **endseason** –
- **save_file** –

Returns

scrapenhl2.plot.team_score_shot_rate.**team_score_shot_rate_scatter**(*team*, *startseason*, *endseason=None*, *save_file=None*)

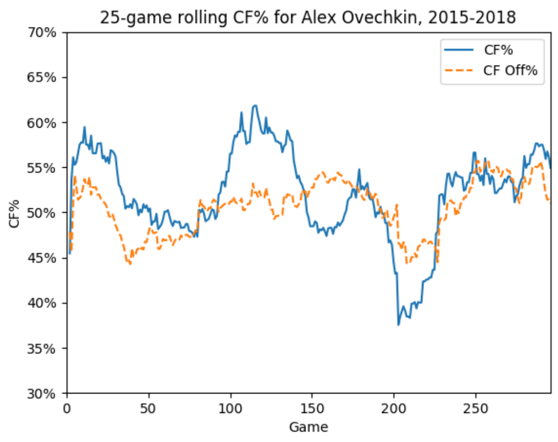
Parameters

- **team** – str or int, team
- **startseason** – int, the starting season (inclusive)
- **endseason** – int, the ending season (inclusive)

Returns nothing

1.3.5 Methods (individuals)

Player rolling CF and GF



This module creates rolling CF% and GF% charts

scrapenhl2.plot.rolling_cf_gf.**rolling_player_cf**(*player*, ***kwargs*)
Creates a graph with CF% and CF% off. Defaults to roll_len of 25.

Parameters

- **player** – str or int, player to generate for
- **kwargs** – other filters. See scrapenhl2.plot.visualization_helper.get_and_filter_5v5_log for more information.

Returns nothing, or figure

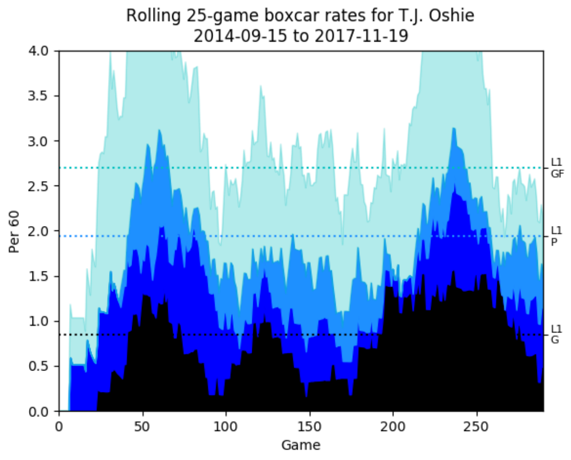
scrapenhl2.plot.rolling_cf_gf.**rolling_player_gf**(*player*, ***kwargs*)
Creates a graph with GF% and GF% off. Defaults to roll_len of 40.

Parameters

- **player** – str or int, player to generate for
- **kwargs** – other filters. See scrapenhl2.plot.visualization_helper.get_and_filter_5v5_log for more information.

Returns nothing, or figure

Player rolling boxcars



This module contains methods for creating the rolling boxcars stacked area graph.

scrapenhl2.plot.rolling_boxcars.**calculate_boxcar_rates**(*df*)
Takes the given dataframe and makes the following calculations:

- Divides col ending in GFON, iA2, iA1, and iG by one ending in TOI
- Adds iG to iA1, calls result iP1
- Adds iG and iA1 to iA2, calls result iP
- Adds /60 to ends of iG, iA1, iP1, iA2, iP, and GFON

Parameters `df` – dataframe

Returns dataframe with columns changed as specified, and only those mentioned above selected.

scrapenhl2.plot.rolling_boxcars.**rolling_player_boxcars**(*player*, ***kwargs*)
A method to generate the rolling boxcars graph.

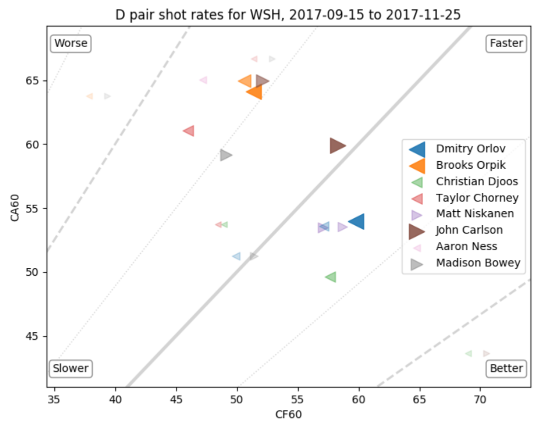
Parameters

- `player` – str or int, player to generate for
- `kwargs` – other filters. See scrapenhl2.plot.visualization_helper.get_and_filter_5v5_log for more information.

Returns nothing, or figure

1.3.6 Methods (individual comparisons)

Team D-pair shot rates



This module contains methods for creating a scatterplot of team defense pair shot rates.

`scrapenhl2.plot.defense_pairs.`**`drop_duplicate_pairs`**(*rates*)

The shot rates dataframe has duplicates–e.g. in one row Orlov is PlayerID1 and Niskanen PlayerID2, but in another Niskanen is PlayerID1 and Orlov is playerID2. This method will select only one, using the following rules:

- For mixed-hand pairs, pick the one where P1 is the lefty and P2 is the righty
- For other pairs, arrange by PlayerID. The one with the smaller ID is P1 and the larger, P2.

Parameters **`rates`** – dataframe as created by get_dpair_shot_rates

Returns dataframe, rates with half of rows dropped

`scrapenhl2.plot.defense_pairs.`**`get_dpair_shot_rates`**(*team*, *startdate*, *enddate*)

Gets CF/60 and CA/60 by defenseman duo (5v5 only) for this team between given range of dates

Parameters

- **`team`** – int or str, team
- **`startdate`** – str, start date
- **`enddate`** – str, end date (inclusive)

Returns dataframe with PlayerID1, PlayerID2, CF, CA, TOI (in secs), CF/60 and CA/60

`scrapenhl2.plot.defense_pairs.`**`team_dpair_shot_rates_scatter`**(*team*, *min_pair_toi=50*, ***kwargs*)

Creates a scatterplot of team defense pair shot attempr rates.

Parameters

- **`team`** – int or str, team
- **`min_pair_toi`** – int, number of minutes for pair to qualify
- **`kwargs`** – Use season- or date-range-related kwargs only.

Returns

Usage

This module creates static and animated usage charts.

`scrapenhl2.plot.usage.`**`animated_usage_chart`**`(`*`**kwargs`*`)`

> **Parameters** **`kwargs`** –
>
> **Returns**

`scrapenhl2.plot.usage.`**`parallel_coords_team_comparison`**`(`*`**kwargs`*`)`

> **Parameters** **`kwargs`** –
>
> **Returns** nothing, or figure

`scrapenhl2.plot.usage.`**`parallel_usage_chart`**`(`*`**kwargs`*`)`

> **Parameters** **`kwargs`** – Defaults to take last month of games for all teams.
>
> **Returns** nothing, or figure

1.3.7 Helper methods

This method contains utilities for visualization.

`scrapenhl2.plot.visualization_helper.`**`add_cfpct_ref_lines_to_plot`**`(`*`ax, refs=None`*`)`

Adds reference lines to specified axes. For example, it could add 50%, 55%, and 45% CF% lines.

50% has the largest width and is solid. 40%, 60%, etc will be dashed with medium width. Other numbers will be dotted and have the lowest width.

Also adds little labels in center of pictured range.

Parameters

- **`ax`** – axes. CF should be on the X axis and CA on the Y axis.
- **`refs`** – None, or a list of percentages (e.g. [45, 50, 55]). Defaults to every 5% from 35% to 65%

Returns nothing

`scrapenhl2.plot.visualization_helper.`**`add_good_bad_fast_slow`**`(`*`margin=0.05, bottomleft='Slower', bottomright='Better', topleft='Worse', topright='Faster'`*`)`

Adds better, worse, faster, slower, to current matplotlib plot. CF60 should be on the x-axis and CA60 on the y-axis. Also expands figure limits by margin (default 5%). That means you should use this before using, say, add_cfpct_ref_lines_to_plot.

Parameters

- **`margin`** – expand figure limits by margin. Defaults to 5%.
- **`bottomleft`** – label to put in bottom left corner
- **`bottomright`** – label to put in bottom right corner
- **`topleft`** – label to put in top left corner
- **`topright`** – label to put in top right corner

Returns nothing

`scrapenhl2.plot.visualization_helper.`**`filter_5v5_for_player`**(*df*, ***kwargs*)

This method filters the given dataframe for given player(s), if specified

Parameters

- **`df`** – dataframe
- **`kwargs`** – relevant one is player

Returns dataframe, filtered for specified players

`scrapenhl2.plot.visualization_helper.`**`filter_5v5_for_team`**(*df*, ***kwargs*)

This method filters the given dataframe for given team(s), if specified

Parameters

- **`df`** – dataframe
- **`kwargs`** – relevant one is team

Returns dataframe, filtered for specified players

`scrapenhl2.plot.visualization_helper.`**`filter_5v5_for_toi`**(*df*, ***kwargs*)

This method filters the given dataframe for minimum or max TOI or TOI60.

This method groups at the player level. So if a player hits the minimum total but not for one or more teams they played for over the the relevant time period, they will be included.

Parameters

- **`df`** – dataframe
- **`kwargs`** – relevant ones are min_toi, max_toi, min_toi60, and max_toi60

Returns dataframe, filtered for specified players

`scrapenhl2.plot.visualization_helper.`**`format_number_with_plus`**(*stringnum*)

Converts 0 to 0, -1 to -1, and 1 to +1 (for presentation purposes).

Parameters **`stringnum`** – int

Returns str, transformed as specified above.

`scrapenhl2.plot.visualization_helper.`**`generic_5v5_log_graph_title`**(*figtype*, ***kwargs*)

Generates a figure title incorporating parameters from kwargs:

[Fig type] for [player, or multiple players, or team] [date range] [rolling window, if applicable] [TOI range, if applicable] [TOI60 range, if applicable]

Methods for individual graphs can take this list and arrange as necessary.

Parameters

- **`figtype`** – str brief description, e.g. Rolling CF% or Lineup CF%
- **`kwargs`** – See get_and_filter_5v5_log

Returns list of strings

`scrapenhl2.plot.visualization_helper.`**`get_5v5_df_start_end`**(***kwargs*)

This method retrieves the correct years of the 5v5 player log and concatenates them.

Parameters **`kwargs`** – the relevant ones here are startseason and endseason

Returns dataframe

`scrapenhl2.plot.visualization_helper.`**`get_and_filter_5v5_log`**(***kwargs*)

This method retrieves the 5v5 log and filters for keyword arguments provided to the original method. For example, rolling_player_cf calls this method first.

Currently supported keyword arguments:

- startseason: int, the season to start with. Defaults to current - 3.
- startdate: str, yyyy-mm-dd. Defaults to Sep 15 of startseason
- endseason: int, the season to end with (inclusive). Defaults to current
- enddate: str, yyyy-mm-dd. Defaults to June 21 of endseason + 1
- roll_len: int, calculates rolling sums over this variable.
- roll_len_days: int, calculates rolling sum over this time window
- player: int or str, player ID or name
- players: list of int or str, player IDs or names
- min_toi: float, minimum TOI for a player for inclusion in minutes.
- max_toi: float, maximum TOI for a player for inclusion in minutes.
- min_toi60: float, minimum TOI60 for a player for inclusion in minutes.
- max_toi60: float, maximum TOI60 for a player for inclusion in minutes.
- team: int or str, filter data for this team only
- add_missing_games: bool. If True will add in missing rows for missing games. Must also specify team.

Developer's note: when adding support for new kwargs, also add support in _generic_graph_title

Parameters **`kwargs`** – e.g. startseason, endseason.

Returns df, filtered

`scrapenhl2.plot.visualization_helper.`**`get_enddate_from_kwargs`**(***kwargs*)

Returns 6/21 of endseason + 1, or enddate

`scrapenhl2.plot.visualization_helper.`**`get_line_slope_intercept`**(*x1*, *y1*, *x2*, *y2*)

Returns slope and intercept of lines defined by given coordinates

`scrapenhl2.plot.visualization_helper.`**`get_startdate_enddate_from_kwargs`**(***kwargs*)

Returns startseason and endseason kwargs. Defaults to current - 3 and current

`scrapenhl2.plot.visualization_helper.`**`hex_to_rgb`**(*value*, *maxval=256*)

Return (red, green, blue) for the hex color given as #rrggbb.

`scrapenhl2.plot.visualization_helper.`**`insert_missing_team_games`**(*df*, ***kwargs*)

Parameters

- **`df`** – dataframe, 5v5 player log or part of it
- **`kwargs`** – relevant ones are 'team' and 'add_missing_games'

Returns dataframe with added rows

`scrapenhl2.plot.visualization_helper.`**`make_5v5_rolling_days`**(*df*, ***kwargs*)

Takes rolling sums based on roll_len_days kwarg. E.g. 30 for a ~monthly rolling sum.

Parameters

- **`df`** – dataframe

- **kwargs** – the relevant one is roll_len_days, int

Returns dataframe with extra columns

`scrapenhl2.plot.visualization_helper.`**`make_5v5_rolling_gp`**(*df*, ***kwargs*)

Takes rolling sums of numeric columns and concatenates onto the dataframe. Will exclude season, game, player, and team.

Parameters

- **df** – dataframe
- **kwargs** – the relevant one is roll_len

Returns dataframe with extra columns

`scrapenhl2.plot.visualization_helper.`**`make_color_darker`**(*hex=None*, *rgb=None*, *returntype='hex'*)

Makes specified color darker. This is done by converting to rgb and multiplying by 50%.

Parameters

- **hex** – str. Specify either this or rgb.
- **rgb** – 3-tuple of floats 0-255. Specify either this or hex
- **returntype** – str, 'hex' or 'rgb'

Returns a hex or rgb color, input color but darker

`scrapenhl2.plot.visualization_helper.`**`make_color_lighter`**(*hex=None*, *rgb=None*, *returntype='hex'*)

Makes specified color lighter. This is done by converting to rgb getting closer to 255 by 50%.

Parameters

- **hex** – str. Specify either this or rgb.
- **rgb** – 3-tuple of floats 0-255. Specify either this or hex
- **returntype** – str, 'hex' or 'rgb'

Returns a hex or rgb color, input color but lighter

`scrapenhl2.plot.visualization_helper.`**`parallel_coords`**(*backgrounddf*, *foregrounddf*, *groupcol*, *legendcol=None*, *axis=None*)

Parameters

- **backgrounddf** –
- **foregrounddf** –
- **groupcol** – For inline labels (e.g. initials)
- **legendcol** – So you can provide another groupcol for legend (e.g. name)
- **axis** –

Returns

`scrapenhl2.plot.visualization_helper.`**`parallel_coords_background`**(*dataframe*, *groupcol*, *axis=None*)

Parameters

- **dataframe** –

- **groupcol** –
- **axis** –
- **zorder** –
- **alpha** –
- **color** –
- **label** –

Returns

`scrapenhl2.plot.visualization_helper.`**`parallel_coords_foreground`**(*dataframe*, *groupcol*, *axis=None*)

Parameters

- **dataframe** –
- **groupcol** –
- **axis** –
- **zorder** –
- **alpha** –
- **color** –
- **label** –

Returns

`scrapenhl2.plot.visualization_helper.`**`parallel_coords_xy`**(*dataframe*, *groupcol*)

Parameters

- **dataframe** – data in wide format
- **groupcol** – column to use as index (e.g. playername)

Returns column dictionary, dataframe in long format

`scrapenhl2.plot.visualization_helper.`**`rgb_to_hex`**(*red*, *green*, *blue*)

Return color as #rrggbb for the given RGB color values.

`scrapenhl2.plot.visualization_helper.`**`savefilehelper`**(***kwargs*)

Saves current matplotlib figure, or saves to file, or displays

Parameters kwargs – searches for 'save_file'. If not found or None, displays figure. If 'fig', returns figure. If a filepath, saves.

Returns nothing, or a figure

This module is from SO. It adds labels for lines on the lines themselves.

`scrapenhl2.plot.label_lines.`**`labelLine`**(*line*, *x*, *label=None*, *align=True*, ***kwargs*)

Labels line with line2D label data

`scrapenhl2.plot.label_lines.`**`labelLines`**(*lines*, *align=True*, *xvals=None*, ***kwargs*)

Labels lines in a line graph

1.4 Support

Feel free to contact me with questions or suggestions.

1.4.1 Docs

Read the Docs

1.4.2 Github

Link.

1.4.3 Contact

Twitter, or create an issue on GitHub.

1.4.4 Collaboration

I'm happy to partner with you in development efforts–just shoot me a message, or just get started by resolving a GitHub issue or suggested enhancement. Please also let me know if you'd like to alpha- or beta-test my code.

1.4.5 Donations

If you would like to support my work, please donate money to a charity of your choice. Many large charities do great work all around the world (e.g. Médecins Sans Frontières), but don't forget that your support is often more critical for local/small charities. Also consider that small regular donations are sometimes better than one large donation.

You can vet a charity you're targeting using a charity rating website.

If you do make a donation, make me happy and leave a record here.. (It's anonymous.)

CHAPTER 2

Introduction

scrapenhl2 is a python package for scraping and manipulating NHL data pulled from the NHL website.

CHAPTER 3

Installation

You need python3 and the python scientific stack (e.g. numpy, matplotlib, pandas, etc). Easiest way is to simply use Anaconda. To be safe, make sure you have python 3.5+, matplotlib 2.0+, and pandas 0.20+.

Next, if you are on Windows, you need to get python-Levenshtein. You can find it here. Download the appropriate .whl file–connect your version of python with the "cp" you see and use the one with "amd64" if you have an AMD 64-bit processor–and navigate to your downloads folder in command line. For example:

```
cd
cd muneebalam
cd Downloads
```

Next, install the whl file using pip:

```
pip install [insert filename here].whl
```

(Sometimes, this errors out and says you need Visual Studio C++ tools. You can download and install the 2015 version from here.)

Now, all users can open up terminal or command line and enter:

```
pip install scrapenhl2
```

(If you have multiple versions of python installed, you may need to alter that command slightly.)

For now, installation should be pretty quick, but in the future it may take awhile (depending on how many past years' files I make part of the package).

As far as coding environments go, I recommend jupyter notebook or Pycharm Community. Some folks also like the PyDev plugin in Eclipse. The latter two are full-scale applications, while the former launches in your browser. Open up terminal or command line and run:

```
jupyter notebook
```

Then navigate to your coding folder, start a new Python file, and you're good to go.

CHAPTER 4

Use

Note that because this is in pre-alpha/alpha, syntax and use may be buggy and subject to change.

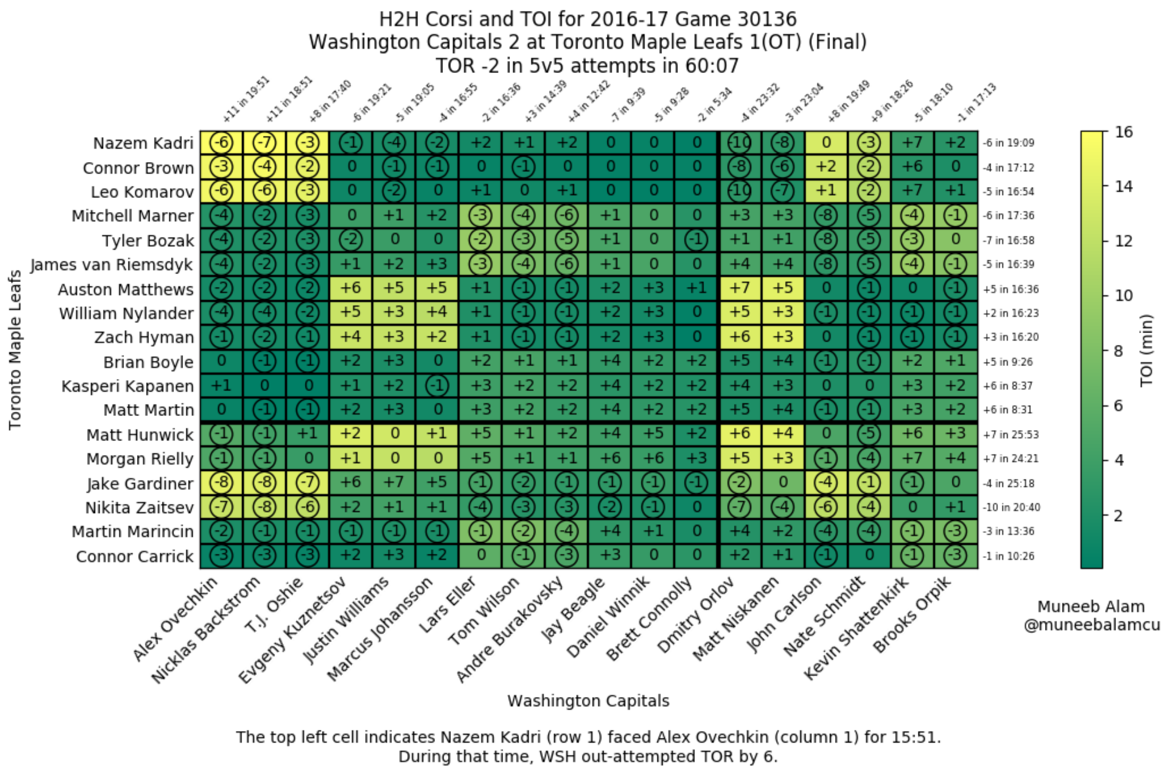
On startup, when you have an internet connection and some games have gone final since you last used the package, open up your python environment and update:

```
from scrapenhl2.scrape import autoupdate
autoupdate.autoupdate()
```

Autoupdate should update you regularly on its progress; be patient.

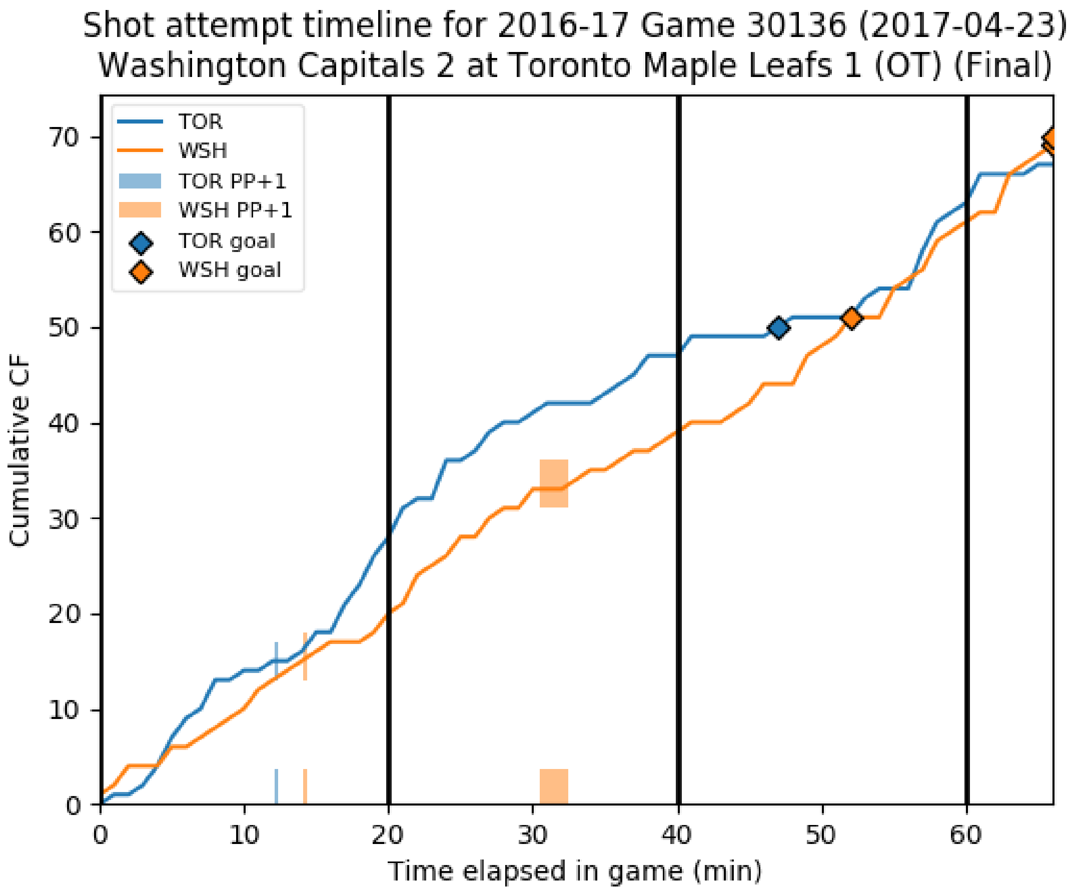
To get a game H2H, use:

```
from scrapenhl2.plot import game_h2h
season = 2016
game = 30136
game_h2h.game_h2h(season, game)
```



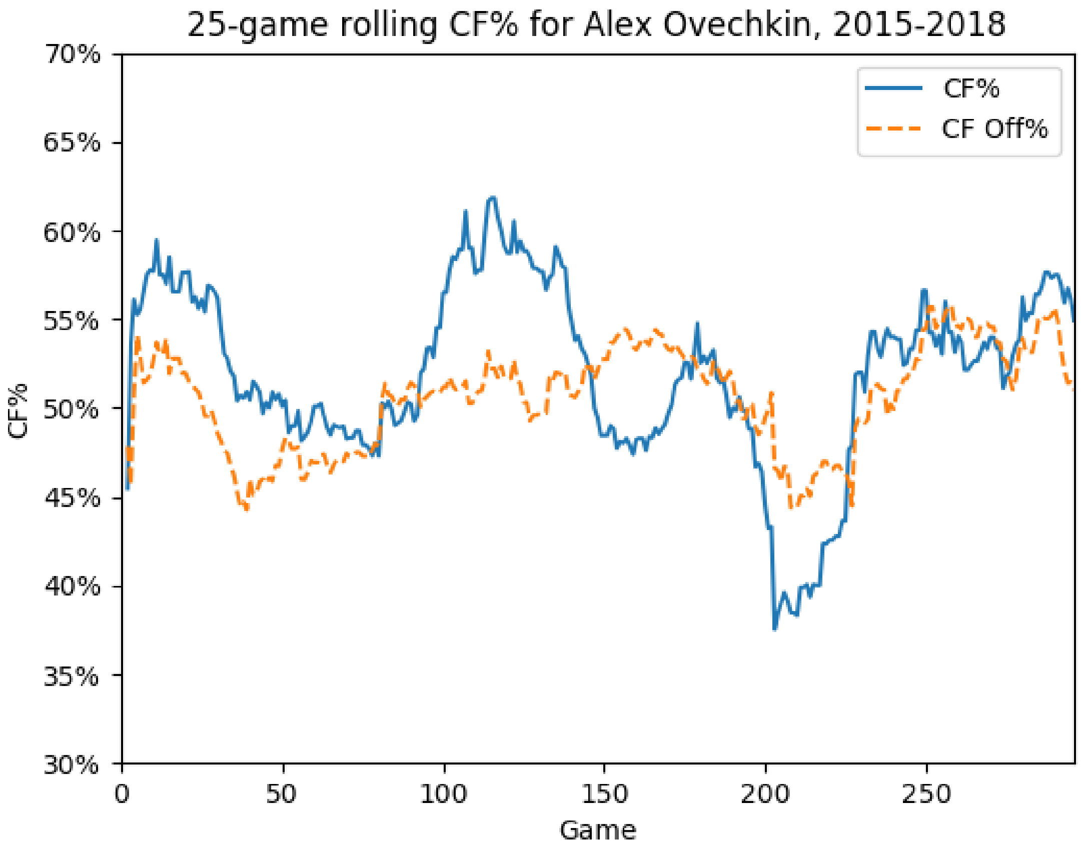
To get a game timeline, use:

```
from scrapenhl2.plot import game_timeline
season = 2016
game = 30136
game_timeline.game_timeline(season, game)
```



To get a player rolling CF% graph, use:

```
from scrapenhl2.plot import rolling_cf_gf
player = 'Ovechkin'
rolling_games = 25
start_year = 2015
end_year = 2017
rolling_cf_gf.rolling_player_cf(player, rolling_games, start_year, end_year)
```



This package is targeted for script use, so I recommend familiarizing yourself with python. (This is not intended to be a replacement for a site like Corsica.)

Look through the documentation at Read the Docs and the examples on Github. Also always feel free to contact me with questions or suggestions.

CHAPTER 5

Contact

Twitter.

CHAPTER 6

Collaboration

I’m happy to partner with you in development efforts–just shoot me a message or submit a pull request. Please also let me know if you’d like to alpha- or beta-test my code.

CHAPTER 7

Donations

If you would like to support my work, please donate money to a charity of your choice. Many large charities do great work all around the world (e.g. Médecins Sans Frontières), but don't forget that your support is often more critical for local/small charities. Also consider that small regular donations are sometimes better than one large donation.

You can vet a charity you're targeting using a charity rating website.

If you do make a donation, make me happy and leave a record here.. (It's anonymous.)

CHAPTER 8

Change log

1/13/18: Various bug fixes, some charts added.

11/10/17: Switched from Flask to Dash, bug fixes.

11/5/17: Bug fixes and method to add on-ice players to file. More refactoring.

10/28/17: Major refactoring. Docs up and running.

10/21/17: Added basic front end. Committed early versions of 2017 logs.

10/16/17: Added initial versions of game timelines, player rolling corsi, and game H2H graphs.

10/10/17: Bug fixes on scraping and team logs. Started methods to aggregate 5v5 game-by-game data for players.

10/7/17: Committed code to scrape 2010 onward and create team logs; still bugs to fix.

9/24/17: Committed minimal structure.

CHAPTER 9

Major outstanding to-dos

- Bring in old play by play and shifts from HTML
- More examples
- More graphs
- More graphs in Dash app

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index