Getting Started

New to *soccerdata*? Well, you came to the right place: this tutorial will walk you through installing, configuring, and using the library. By the end of this tutorial, you will be able to scrape data from the top-5 European leagues and use it to create your own data-driven analyses.

Installation

SoccerData can be easily installed via pip:

```
python3 -m pip install soccerdata
```

Scraping data

Each of the supported data sources has its corresponding class for fetching data with a uniform API. For example, the `FBref` class is used to fetch data from fbref.com.

```
import soccerdata as sd

# Create scraper class instance
fbref = sd.FBref()
```

Once you have a scraper class instance, you can use it to fetch data. See the the examples and API reference for the full list of options available for each scraper. For example, to fetch aggregated shooting stats for all teams:

```
# Create dataframes
season_stats = fbref.read_team_season_stats(stat_type='shooting')
```

The data is always returned as a convenient Pandas DataFrame.

Skip to content latest

league	season	team	#Pl	90s	Gls	Sh	SoT	SoT%	Sh/90	SoT/90	G/Sh
ENG-Premier League	2021	Arsenal	29	38.0	53	455	141	31.0	11.97	3.71	0.1
		Aston Villa	24	38.0	52	518	179	34.6	13.63	4.71	0.09
		Brighton	27	38.0	39	476	129	27.1	12.53	3.39	0.07
		Burnley	25	38.0	32	383	125	32.6	10.08	3.29	0.08
		Chelsea	27	38.0	56	553	194	35.1	14.55	5.11	0.09

By default, the data for all available leagues and the five most recent seasons will be retrieved. However, in most cases, you would want to limit the data to specific leagues and / or seasons. This can be done by passing a list of leagues and seasons to the constructor of the scraper class. For example:

```
# Create scraper class instance filtering on specific leagues and seasons
fbref = sd.FBref(leagues=['ENG-Premier League'], seasons=['1718', '1819'])
# Retrieve data for the specified leagues and seasons
season_stats = fbref.read_team_season_stats(stat_type='shooting')
```

Note that only a limited number of leagues are supported out-of-the-box. The leagues available for each source can be listed with the `available_leagues()` class method.

```
sd.FBref.available_leagues()
>>> ['Big 5 European Leagues Combined', 'ENG-Premier League', 'ESP-La Liga', 'FRA-Ligue 1
```

You can add more leagues but there are no guarantees that they will be scraped correctly.

Data caching

Data caching is used to speed up the runtime and to prevent exceeding the rate limit of web servers. By default, all downloaded data is cached to `~/soccerdata` on Linux and Mac OS, and to `C:\Users\yourusername\soccerdata` on Windows. A custom location can be set if desired. You can

Skip to content ng environment variables (see below) or on the level of an individual scraper by

`lir` parameter when creating the scraper class instance: latest

```
# Create scraper class instance with custom caching directory
fbref = sd.FBref(data_dir="/tmp/FBref")
```

This directory can be deleted at any time to reclaim disk space. However, this also means you will have to redownload the same data again if you need it, which will lead to reduced performance.

SoccerData has no knowledge of when the data on the server changes, so it is up to the user to decide when to refresh the cache. This can be done by deleting the cache directory or by setting the `no_cache` option to `True` when creating the scraper class instance:

```
# Create scraper class instance which always re-downloads the latest data
fbref = sd.FBref(no_cache=True)
```

Some methods will assume the cache is always out-of-date (for example, when scraping the fixture of the current season). Typically, these methods will have a `force_cache` option that can be set to `True` to force the cached data to be used. For example:

```
fbref = sd.FBref(leagues=['ENG-Premier League'], seasons=['2324'])
fbref.read_schedule(force_cache=True)
```

Caching can also be disabled entirely by setting the `no_store` option to `True` when creating the scraper class instance. However, it should almost always be left enabled.

```
# Create scraper class instance with caching disabled
fbref = sd.FBref(no_store=True)
```

Global configuration

Several settings can be configured globally using the following environment variables:

`SOCCERDATA_DIR`
: The directory where the downloaded data is cached and where logs are stored. By default, all data is stored to `~/soccerdata` on Linux / Mac OS and `C:\Users\yourusername\soccerdata` on Windows.

`SOCCERDATA_NOCACHE`
: If set to "true", no cached data is returned. Note that no-cache does not mean "don't cache". All downloaded data is still cached and overwrites existing caches. If the sense of "don't cache" that you want is actually "don't store", then `SOCCERDATA_NOSTORE` is the option to use. By default, data is retrieved from the cache.

`SOCCERDATA_NOSTORE`
: If set to "true", no data is stored. By default, data is cached.

`SOCCERDATA_MAXAGE`
: The maximum age of cached data in seconds. If the cached data is older than this, it will be re-downloaded. By default, this is set to infinity.

`SOCCERDATA_LOGLEVEL`
: The level of logging to use. By default, this is set to "INFO".

Example:

```bash
# bash
export SOCCERDATA_DIR = "~/soccerdata"
export SOCCERDATA_NOCACHE = "False"
export SOCCERDATA_NOSTORE = "False"
export SOCCERDATA_LOGLEVEL = "INFO"
```

Uniform team names

Each data source uses a different set of team names, which makes it difficult to combine data from multiple sources. To mitigate this, SoccerData allows translating the team names to uniform names. This is done by providing a `SOCCERDATA_DIR/config/teamname_replacements.json` file. This file should contain a mapping between a generic name for each team and the team name used by that you want to support. The example below will map "Tottenham Hotspur" Skip to content ur FC" and "Spurs" to "Tottenham" in all scraped data. latest

```
{
  "Tottenham": ["Tottenham Hotspur", "Tottenham Hotspur FC", "Spurs"],
}
```

Additional setup for scraping WhoScored data

WhoScored implements strong protection against scraping using Incapsula. To circumvent this, this scraper uses Selenium with the ChromeDriver extension to emulate a real user. Before using this scraper, you will have to install Chrome. A Selenium driver matching your Chrome version will be downloaded automatically when you run the scraper.

Next steps

Look at you! You're now basically an expert at SoccerData! ✨

From this point you can:

- Look at the example notebooks for each Data source.
- Take a deep dive into the API.
- Give us feedback or contribute, see Contributing.

Have fun! 🎉

latest