CIROH Baseflow Python Package

Release 0.1

CIROH-GW Research Assistants

May 14, 2024

TABLE OF CONTENTS

The baseflow Python package is designed to . . . .

CHAPTER ONE

OVERVIEW

How to use

How to install

Etc

Etc

1.1 CIROH

Something here acknowledging our project, objectives, etc

include links to CIROH, NOAA

https://ciroh.ua.edu/ CIROH, a partnership between NOAA and The University of Alabama, is a national consortium committed to advancing water prediction – the forecasting of streamflow entering water systems, extreme events such as floods and droughts, and water quality – and building community resilience to water-related challenges. CIROH scientists, from 28 different institutions—academic, government, and private, work to improve the understanding of hydrologic processes, operational hydrologic forecasting techniques and workflows, community water modeling, translation of forecasts to actionable products, and use of water predictions in decision making.

1.2 BYU Hydroinformatics Lab

short description

link

CHAPTER TWO

API DOCUMENTATION

`baseflow.models.`**`lyne_hollick`**(*streamflow_list*, *alpha*)

Calculates baseflow approximations using the Lyne and Hollick equation.

Parameters

- **`streamflow_list`** (*`list`*) – A list of streamflow values
- **`alpha`** (*`float`*) – Catchment constant between 0 and 1

Returns

A timeseries list of baseflow values

Return type

list

Example

```
import pandas as pd
discharge_time_series = pd.read_csv("/my/sample/file.csv")
alpha = 0.925
baseflow = lyne_hollick(discharge_time_series['Discharge'], alpha)
```

`baseflow.models.`**`chapman`**(*streamflow_list*, *alpha*, *beta*)

Calculates baseflow approximations using the Chapman equation.

Parameters

- **`streamflow_list`** (*`float series`*) – A list of streamflow values
- **`alpha`** (*`float`*) – Hydrological recession constant between 0 and 1

Returns

A timeseries list of baseflow values

Return type

list

Example

```
import pandas as pd
discharge_time_series = pd.read_csv("/my/sample/file.csv")
alpha = 0.925
baseflow = chapman(discharge_time_series['Discharge'], alpha)
```

`baseflow.models.`**`eckhardt`**`(`*streamflow_list*, *alpha*, *bfi_max*`)`

Calculates baseflow approximations using the Eckhardt equation.

Parameters

- **`streamflow_list`** (*`float series`*) – A list of streamflow values
- **`alpha`** (*`float`*) – Hydrological recession constant between 0 and 1
- **`bfi_max`** – BFImax is the maximum attainable value of the baseflow index, indicating the long-term ratio of baseflow to total streamflow computed using a filtering algorithm. It's always less than 1, implying the absence of direct runoff in a catchment. This suggests either highly permeable soil or flat terrain.

Returns
A timeseries list of baseflow values

Return type
list

Example

```
import pandas as pd
discharge_time_series = pd.read_csv("/my/sample/file.csv")
alpha = 0.925
bfi_max = 0.8
baseflow = eckhardt(discharge_time_series['Discharge'], alpha, bfi_max)
```

`baseflow.models.`**`chapman_maxwell`**`(`*streamflow_list*, *k*`)`

Separates baseflow from a streamflow hydrograph using the Chapman & Maxwell method.

Parameters

- **`streamflow_list`** (*`list`*) – A list of streamflow values in chronological order.
- **`k`** (*`float`*) – A smoothing parameter between 0 and 1.

Returns
A timeseries list of baseflow values.

Return type
list

Example

```
import pandas as pd
discharge_time_series = pd.read_csv("/my/sample/file.csv")
k = 0.9
baseflow = chapman_maxwell(discharge_time_series['Discharge'], k)
```

`baseflow.models.`**`hyd_run`**(*streamflow_list*, *k*, *passes*)

Separates baseflow from a streamflow hydrograph using a digital filter method.

Parameters

- **`streamflow_list`** (*`pandas.Series`*) – A pandas Series of streamflow values in chronological order.
- **`k`** (*`float`*) – A filter coefficient between 0 and 1 (typically 0.9).
- **`passes`** (*`int`*) – Number of times the filter passes through the data (typically 4).

Returns
A list of baseflow values.

Return type
list

Example

```
import pandas as pd
discharge_time_series = pd.read_csv("/my/sample/file.csv")
k = 0.9
passes = 4
baseflow_list = hyd_run(discharge_time_series['Discharge'], k, passes)
```

CHAPTER

THREE

CONTRIBUTORS GUIDE

Welcome to the contributors guide for our project! This guide is designed to help you get started with contributing to our open-source project. Whether you're a seasoned developer or just getting started, there are many ways you can contribute and make a difference.

This guide will walk you through the process of cloning the repository, making changes, and submitting a pull request. By following these steps, you'll be able to contribute your ideas, fixes, and improvements to the project.

We value your contributions and appreciate the time and effort you put into making our project better for everyone. Thank you for considering contributing, and we look forward to seeing your contributions!

Now, let's get started!

3.1 Cloning the Repo

To contribute to our project, you'll first need to clone the repository to your local machine. Follow these steps:

1. **Fork the Repository**: Go to our project's GitHub page and click on the "Fork" button in the top-right corner. This creates a copy of the repository in your GitHub account.
2. **Clone Your Fork**: Open a terminal and use the *git clone* command to clone your forked repository to your local machine. Replace <*your_username*> with your GitHub username.

   ```
   git clone https://github.com/<your_username>/your-repo.git
   ```

3. **Set Up Upstream**: Navigate to the directory of your cloned repository and set up an upstream remote pointing to the original repository.

   ```
   cd your-repo
   git remote add upstream https://github.com/BYU-Hydroinformatics/baseflow.git
   ```

4. **Verify Remotes**: Ensure that your remotes are set up correctly by running:

   ```
   git remote -v
   ```

3.2 Submitting a Pull Request

Once you've made changes to the code or documentation, you can submit a pull request to have your contributions reviewed and merged into the main project. Here's how:

1. **Create a Branch**: Before making changes, create a new branch to work in. This keeps your changes isolated from the main codebase.

   ```
   git checkout -b feature-branch
   ```

2. **Make Changes**: Make your desired changes to the code or documentation within your branch.
3. **Commit Changes**: Once you're satisfied with your changes, commit them to your branch.

   ```
   git add .
   git commit -m "Your commit message here"
   ```

4. **Push Changes**: Push your changes to your forked repository on GitHub.

   ```
   git push origin feature-branch
   ```

5. **Submit Pull Request**: Visit your forked repository on GitHub and click on the "New Pull Request" button. Select the branch you just pushed from the dropdown menu and provide a descriptive title and comment for your pull request.
6. **Review and Merge**: A project maintainer will review your pull request, possibly requesting changes. Once approved, your changes will be merged into the main project.

PYTHON MODULE INDEX

INDEX