Dac-Man Documentation

Creating Plug-ins: version 1.0.0

July 8, 2019

Contents

1

Introduction

1.1 Overview

Scientific datasets come in various formats and data types. It is not a sustainable solution to compare files and datasets of different types in a single way. For example image files are a lot different than text files and hence, need to be compared differently. Additionally, different domain sciences have different structures to their files.

The Dac-Man plug-in framework allows users to implement their own plug-ins for comparing files and datasets of different types and formats. Additionally, Dac-Man also provides default plug-ins for a variety of different scientific data formats including hierarchical datasets (e.g., HDF5), image files (e.g., FITS, EDF) and tabular data (e.g., csv, Excel).

1.2 Usage

The plug-ins in Dac-Man are generally placed inside the `plugins/` directory. Dac-Man automatically selects the specific plug-in at runtime based on the file types. However, users can also specify explicitly which plug-in to use for data comparisons. Generally, the plug-ins in Dac-Man can be specified in three different ways.

1.2.1 Plug-ins configuration

When Dac-Man is installed, a plug-ins configuration file is created in user's `HOME` directory.

```
$HOME/.dacman/config/plugins.yaml
```

This file may contain all the registered internal plug-ins in Dac-Man. Users can edit this file to specify the selection of a plug-in for specific file-type comparisons.

1.2.2 CLI

Dac-Man also allows users to use their own change analysis scripts as plug-ins. To use this, users need to specify `-p/--plugin` option through the Dac-Man CLI.

```
dacman diff file1 file2 –p /path/to/myscript
```

1.2.3 Dac-Man API

Users can write their own change analysis pipelines using the Dac-Man API. The API also provides the necessary module for users to specify a plugin in their codes.

```
from dacman import Executor, DataDiffer
from dacman.plugins.default import DefaultPlugin
def diff1(file1, file2):
  comparisons = [(file1, file2)]
  differ = DataDiffer(comparisons, Executor.DEFAULT)
  differ.use_plugin(DefaultPlugin)
  differ.start()
```

Users can also create and register their own plug-ins in Dac-Man. The next chapter describes the Dac-Man plug-in API that can be used to create user-defined plug-ins in Dac-Man.

2
The Plug-in API

The plug-ins in Dac-Man are different 'comparators' that capture changes in files and datasets. All the comparator plug-ins, registered to Dac-Man are derived from the base `Comparator` class.

2.1 class Comparator

The `Comparator` class is an abstract class that provides the underlying methods for defining the metadata and algorithm for capturing data changes. All the methods in this class are abstract and need to be implemented by the deriving plug-in class.

2.1.1 supports()

A static method that sets up all the file/data types that are supported by the specific plug-in.

2.1.2 description()

A static method that describes the specific plug-in.

2.1.3 compare(a, b, *args)

The core method for implementing the algorithm to compare two files/datasets.

2.1.4 percent_change()

Method to summarize the amount of change in two files with respect to the comparison algorithm.

2.1.5 stats(changes)

Method for providing statistics on calculated changes from the `compare` method.

2.2 Creating Plug-ins

In order to create and register plug-ins in Dac-Man, users need to implement the methods in the `Comparator` class. Users can either choose to add multiple plug-ins in a single module, or create a module for each plug-in.

The following example shows a module for each plug-in (single_plugin.py):

```
class MyPlugin(Comparator):
  def supports():
    ...
  ...
  def compare(a, b, *args):
    ...
  ...
```

The following example shows multiple plug-ins in a module (multiple_plugins.py):

```
class TxtPlugin(Comparator):
  def supports():
    return ['txt']
  ...
  def compare(a, b, *args):
    ...
  ...


class CsvPlugin(Comparator):
  def supports():
    return ['csv']
  ...
  def compare(a, b, *args):
    ...
  ...
```

3 Default Plug-in

The default plug-in in Dac-Man uses a data abstraction, called the Dac-Man records to compare changes for different file formats and data types. Dac-Man records transforms the header and data of a file into an array. The records from two files are then compared using a linear algorithm.

3.1 Adaptors

The default plug-in uses several adaptors to transform data from different file formats into Dac-Man records. Currently, the plug-in uses adaptors for the following file formats:

- h5: HDF5 file format
- fits: image file format consisting of n-dimensional arrays or tables
- edf: time-series data
- tif: high-quality graphics image format

In order to use the default plug-in, users need to install the required libraries/-packages. Following is the list of all the packages required based for the corresponding file format support.

- h5: h5py
- fits: astropy
- edf: fabio
- tif: fabio

3.2 Change Stats

The default plug-in captures the following data change metrics:

- added: number of data values added
- deleted: number of data values deleted
- modified: number of data values modified
- unchanged: number of unchanged data values

In addition, the default plug-in also calculates % change between two files based on the Dac-Man records comparison.

4

HDF5 Plug-in

5 Tabular Data Plug-in