pytest-regressions Documentation

ESSS

Nov 20, 2018

Contents:

CHAPTER 1

Installation

Install `pytest-regressions` using `pip`:

```
$ pip install pytest-regressions
```

Or if you are using `conda`:

```
$ conda install -c conda-forge pytest-regressions
```

CHAPTER 2

Overview

pytest-regressions provides some fixtures that make it easy to maintain tests that generate lots of data or specific data files like images.

This plugin uses a *data directory* (courtesy of pytest-datadir) to store expected data files, which are stored and used as baseline for future test runs.

2.1 Example

Let's use data_regression as an example, but the workflow is the same for the other *_regression fixtures.

Suppose we have a summary_grids function which outputs a dictionary containing information about discrete grids for simulation. Of course your function would actually return some computed/read value, but here it is using an inline result for this example:

```
def summary_grids():
    return {
        "Main Grid": {
            "id": 0,
            "cell_count": 1000,
            "active_cells": 300,
            "properties": [
                {"name": "Temperature", "min": 75, "max": 85},
                {"name": "Porosity", "min": 0.3, "max": 0.4},
            ],
        },
        "Refin1": {
            "id": 1,
            "cell_count": 48,
            "active_cells": 44,
            "properties": [
                {"name": "Temperature", "min": 78, "max": 81},
                {"name": "Porosity", "min": 0.36, "max": 0.39},
```

(continues on next page)

(continued from previous page)

```
            ],
        },
    }
```

We could test the results of this function like this:

```
def test_grids():
    data = summary_grids()
    assert data["Main Grid"]["id"] == 0
    assert data["Main Grid"]["cell_count"] == 1000
    assert data["Main Grid"]["active_cells"] == 300
    assert data["Main Grid"]["properties"] == [
        {"name": "Temperature", "min": 75, "max": 85},
        {"name": "Porosity", "min": 0.3, "max": 0.4},
    ]
    ...
```

But this presents a number of problems:

- Gets old quickly.
- Error-prone.
- If a check fails, we don't know what else might be wrong with the obtained data.
- Does not scale for large data.
- **Maintenance burden**: if the data changes in the future (and it will) it will be a major head-ache to update the values, specially if there are a lot of similar tests like this one.

2.2 Using data_regression

The `data_regression` fixture provides a method to check general dictionary data like the one in the previous example.

Just declare the `data_regression` fixture and call the `check` method:

```
def test_grids2(data_regression):
    data = summary_grids()
    data_regression.check(data)
```

The first time your run this test, it will *fail* with a message like this:

```
>           pytest.fail(msg)
E           Failed: File not found in data directory, created:
E           - C:\Users\bruno\pytest-regressions\tests\test_grids\test_grids2.yml
```

The fixture will generate a `test_grids2.yml` file (same name as the test) in the *data directory* with the contents of the dictionary:

```
Main Grid:
  active_cells: 300
  cell_count: 1000
  id: 0
  properties:
  - max: 85
```

(continues on next page)

(continued from previous page)

```
    min: 75
    name: Temperature
  - max: 0.4
    min: 0.3
    name: Porosity
Refin1:
  active_cells: 44
  cell_count: 48
  id: 1
  properties:
  - max: 81
    min: 78
    name: Temperature
  - max: 0.39
    min: 0.36
    name: Porosity
```

This file should be committed to version control.

The next time you run this test, it will compare the results of `summary_grids()` with the contents of the YAML file. If they match, the test passes. If they don't match the test will fail, showing a nice diff of the text differences.

If the test fails because the new data is correct (the implementation might be returning more information about the grids for example), then you can use the `--force-regen` flag to update the expected file:

```
$ pytest --force-regen
```

and commit the updated file.

This workflow makes it very simple to keep the files up to date and to check all the information we need.

CHAPTER 3

API Reference

Note: Any references to `str` in the documentation below can also be considered `unicode` in Python 2.7.

3.1 data_regression

`DataRegressionFixture.`**`check`**(*data_dict*, *basename=None*, *fullpath=None*)

Checks the given dict against a previously recorded version, or generate a new file.

Parameters

- **`data_dict`** (*`dict`*) – any yaml serializable dict.
- **`basename`** (*`str`*) – basename of the file to test/record. If not given the name of the test is used. Use either *basename* or *fullpath*.
- **`fullpath`** (*`str`*) – complete path to use as a reference file. This option will ignore `datadir` fixture when reading *expected* files but will still use it to write *obtained* files. Useful if a reference file is located in the session data dir for example.

`basename` and `fullpath` are exclusive.

3.2 file_regression

`FileRegressionFixture.`**`check`**(*contents*, *encoding=None*, *extension='.txt'*, *newline=None*, *basename=None*, *fullpath=None*, *binary=False*, *obtained_filename=None*, *check_fn=None*)

Checks the contents against a previously recorded version, or generate a new file.

Parameters

- **`contents`** (*`str`*) – content to be verified.

- **encoding** (*str | None*) – Encoding used to write file, if any.
- **extension** (*str*) – Extension of file.
- **newline** (*str | None*) – See *io.open* docs.
- **binary** (*bool*) – If the file is binary or text.
- **obtained_filename** – ..see:: FileRegressionCheck
- **check_fn** – a function with signature `(obtained_filename, expected_filename)` that should raise AssertionError if both files differ. If not given, use internal function which compares text using `difflib`.

3.3 num_regression

`NumericRegressionFixture.`**check**(*data_dict*, *basename=None*, *fullpath=None*, *tolerances=None*, *default_tolerance=None*, *data_index=None*, *fill_different_shape_with_nan=True*)

Checks the given dict against a previously recorded version, or generate a new file. The dict must map from user-defined keys to 1d numpy arrays.

Example:

```
num_regression.check({
    'U_gas': U[0][positions],
    'U_liquid': U[1][positions],
    'gas_vol_frac [-]': vol_frac[0][positions],
    'liquid_vol_frac [-]': vol_frac[1][positions],
    'P': Pa_to_bar(P)[positions],
})
```

Parameters

- **data_dict** (*dict*) – dict mapping keys to numpy arrays.
- **basename** (*str*) – basename of the file to test/record. If not given the name of the test is used.
- **fullpath** (*str*) – complete path to use as a reference file. This option will ignore embed_data completely, being useful if a reference file is located in the session data dir for example.
- **tolerances** (*dict*) – dict mapping keys from the data_dict to tolerance settings for the given data. Example:

  ```
  tolerances={'U': Tolerance(atol=1e-2)}
  ```

- **default_tolerance** (*dict*) – dict mapping the default tolerance for the current check call. Example:

  ```
  default_tolerance=dict(atol=1e-7, rtol=1e-18).
  ```

  If not provided, will use defaults from numpy's `isclose` function.
- **data_index** (*list*) – If set, will override the indexes shown in the outputs. Default is panda's default, which is `range(0, len(data))`.

- **`fill_different_shape_with_nan`** (*`bool`*) – If set, all the data provided in the data_dict that has size lower than the bigger size will be filled with `np.NaN`, in order to save the data in a CSV file.

`basename` and `fullpath` are exclusive.

3.4 image_regression

`ImageRegressionFixture.`**`check`**(*image_data*, *diff_threshold=0.1*, *expect_equal=True*, *basename=None*)

Checks that the given image contents are comparable with the ones stored in the data directory.

Parameters

- **`image_data`** (*`bytes`*) – image data
- **`basename`** (*`str | None`*) – basename to store the information in the data directory. If none, use the name of the test function.
- **`expect_equal`** (*`bool`*) – if the image should considered equal below of the given threshold. If False, the image should be considered different at least above the threshold.
- **`diff_threshold`** (*`float`*) – Tolerage as a percentage (1 to 100) on how the images are allowed to differ.

CHAPTER 4

License

Index