SMDC$_{perftestsDocumentation}$

Release unknown

Christoph Paulik

June 09, 2015

Contents

This is the documentation of **SMDC_perftests**, a small python module that provides a decorator for measuring the time a function needs to execute. It then stores the results in a `SMDC_perftests.performance_tests.test_runner.TestResults` object. These objects can be compared to each other to quickly find out if the measured time was significantly different using a 95% confidence interval.

The objects can also be stored to and restored from netCDF4 files on disk. There are also plotting functions for the TestResults object.

CHAPTER 1

Requirements

This package was tested using python2.7 and requires the packages

```
netCDF4
pytest
matplotlib
pygeogrids

# optional
# seaborn for pretty plots
```

CHAPTER 2

Contents

2.1 Examples

2.1.1 Basic Example

```
import smdc_perftests.performance_tests.test_cases as test_cases
import time
import numpy as np
```

```
# use measure decorator to run function multiple times
# and measure execution time of each run
# the returned results gets the name given in
# the decorator but can be changed later if necessary

@test_cases.measure('experiment', runs=50)
def experiment(sleeptime=0.01):
    time.sleep(sleeptime+np.random.rand(1)*sleeptime)

result1 = experiment()
result2 = experiment(0.05)
result2.name = "sleep 0.05"
result3 = experiment(0.011)
result3.name = "sleep 0.011"

# the results can be printed
print result1
print result3
```

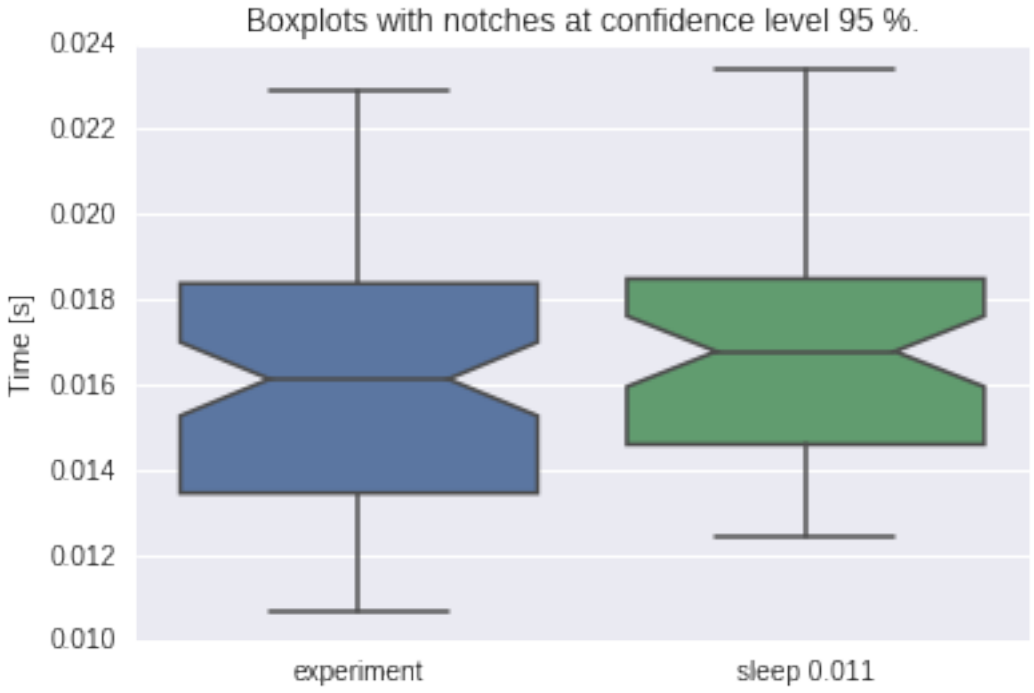
```
Results experiment
50 runs
median 0.0158 mean 0.0157 stdev 0.0029
sum 0.7859
95%% confidence interval of the mean
upper 0.0165
       |
mean  0.0157
       |
lower 0.0149

Results sleep 0.011
50 runs
median 0.0158 mean 0.0163 stdev 0.0034
```

```
sum 0.8168
95%% confidence interval of the mean
upper 0.0173
       |
mean  0.0163
       |
lower 0.0154
```

```
# the results can also be compared based on the 95% confidence intervals.

print result1 < result2
print result2 < result1
print result1 < result3
```

```
True
False
False
```

```
# or then plotted as boxplots
import smdc_perftests.visual as vis
import matplotlib.pyplot as plt
%matplotlib inline

fig, axis = vis.plot_boxplots(result1, result3)
plt.show()
```



2.1.2 Example with Dataset

```
import smdc_perftests.performance_tests.test_runner as test_runner
import time
import datetime as dt
import numpy as np
```

```
# define a fake Dataset class that implements the methods
# get_timeseries, get_avg_image and get_data

class FakeDataset(object):

    """
    Fake Dataset that provides routines for reading
    time series and images
    that do nothing
    """

    def __init__(self):
        pass
        self.ts_read = 0
        self.img_read = 0
        self.cells_read = 0

    def get_timeseries(self, gpi, date_start=None, date_end=None):
        time.sleep(0.01*np.random.rand(1))
        self.ts_read += 1
        return None

    def get_avg_image(self, date_start, date_end=None, cell_id=None):
        """
        Image readers generally return more than one
        variable. This should not matter for these tests.
        """
        assert type(date_start) == dt.datetime
        self.img_read += 1
        time.sleep(0.01*np.random.rand(1))
        return None, None, None, None, None

    def get_data(self, date_start, date_end, cell_id):
        """
        Image readers generally return more than one
        variable. This should not matter for these tests.
        """
        assert type(date_start) == dt.datetime
        assert type(date_end) == dt.datetime
        self.cells_read += 1
        time.sleep(0.01*np.random.rand(1))
        return None, None, None, None, None
```

```
fd = FakeDataset()
# setup grid point index list, must come from grid object or
# sciDB
# this test dataset has 10000 gpis of which 1 percent will be read
gpi_list = range(10000)

@test_runner.measure('test_rand_gpi', runs=100)
def test_ts():
    test_runner.read_rand_ts_by_gpi_list(fd, gpi_list)

result_ts = test_ts()

print result_ts
```

```
Results test_rand_gpi
100 runs
median 0.5642 mean 0.5591 stdev 0.0334
sum 55.9069
95%% confidence interval of the mean
upper 0.5657
       |
mean  0.5591
       |
lower 0.5524
```

```
# setup datetime list
# this test dataset has 10000 days of dates of which 1 percent will be read
date_list = []
for days in range(10000):
    date_list.append(dt.datetime(2007, 1, 1) + dt.timedelta(days=days))

@test_runner.measure('test_rand_date', runs=100)
def test_img():
    test_runner.read_rand_img_by_date_list(fd, date_list)

result_img = test_img()
print result_img
```

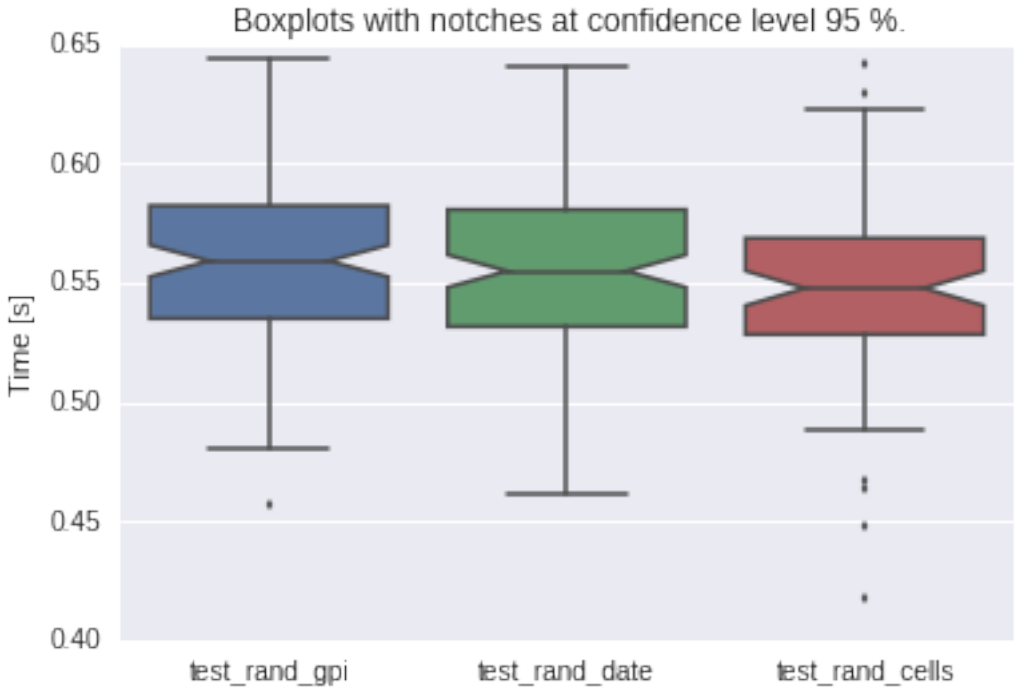
```
Results test_rand_date
100 runs
median 0.5530 mean 0.5548 stdev 0.0343
sum 55.4800
95%% confidence interval of the mean
upper 0.5616
       |
mean  0.5548
       |
lower 0.5480
```

```
"""
Read data by cell list using fixed start and end date
1 percent of the cells are read with a minimum of 1 cell.
"""
fd = FakeDataset()
cell_list = range(10000)

@test_runner.measure('test_rand_cells', runs=100)
def test():
    test_runner.read_rand_cells_by_cell_list(fd,
                                             dt.datetime(2007, 1, 1), dt.datetime(2008, 1, 1), cell_l

results_cells = test()
print results_cells
```

```
Results test_rand_cells
100 runs
median 0.5510 mean 0.5476 stdev 0.0368
sum 54.7624
95%% confidence interval of the mean
upper 0.5549
       |
mean  0.5476
```

```
       |
lower 0.5403
```

```
import smdc_perftests.visual as vis
import matplotlib.pyplot as plt
%matplotlib inline

fig, axis = vis.plot_boxplots(result_ts, result_img, results_cells)
plt.show()
```



2.1.3 Example of running the test suite and analyzing the results

```
import os
from datetime import datetime
from smdc_perftests.performance_tests import test_scripts
# the test_scripts module contains the function
# run performance tests which runs all the performance tests on a dataset

# in this example we will use the esa cci dataset class
from smdc_perftests.datasets.esa_cci import ESACCI_netcdf
from smdc_perftests import helper
```

```
#init the esa cci dataset
fname = os.path.join("/media", "sf_H", "Development", "python",
                     "workspace",
                     "SMDC", "SMDC_perftests", "tests", "test_data",
                     "ESACCI-2Images.nc")
# only read the sm variable for this testrun
ds = ESACCI_netcdf(fname, variables=['sm'])
# get the testname from the filename
testname = os.path.splitext(os.path.split(fname)[1])[0]
```

```
# generate a date range list using the helper function
# in this example this does not make a lot of sense
date_range_list = helper.generate_date_list(datetime(2013, 11, 30),
                                            datetime(2013, 12, 1),
                                            n=50)

# set a directory into which to save the results
# in this case the the tests folder in the home directory
res_dir = "/home/pydev/tests/"
# run the performance tests using the grid point indices from
# the dataset grid, the generated date_range_list and gpi read percentage
# of 0.1 percent and only one repeat
test_scripts.run_performance_tests(testname, ds, res_dir,
                                   gpi_list=ds.grid.land_ind,
                                   date_range_list=date_range_list,
                                   gpi_read_perc=0.1,
                                   repeats=1)
```

```
reading 245 out of 244243 time series
reading 1 out of 50 dates
reading 1 out of 50 dates
```

This creates the following files named using the name given to the test and the name of the test function that was run.

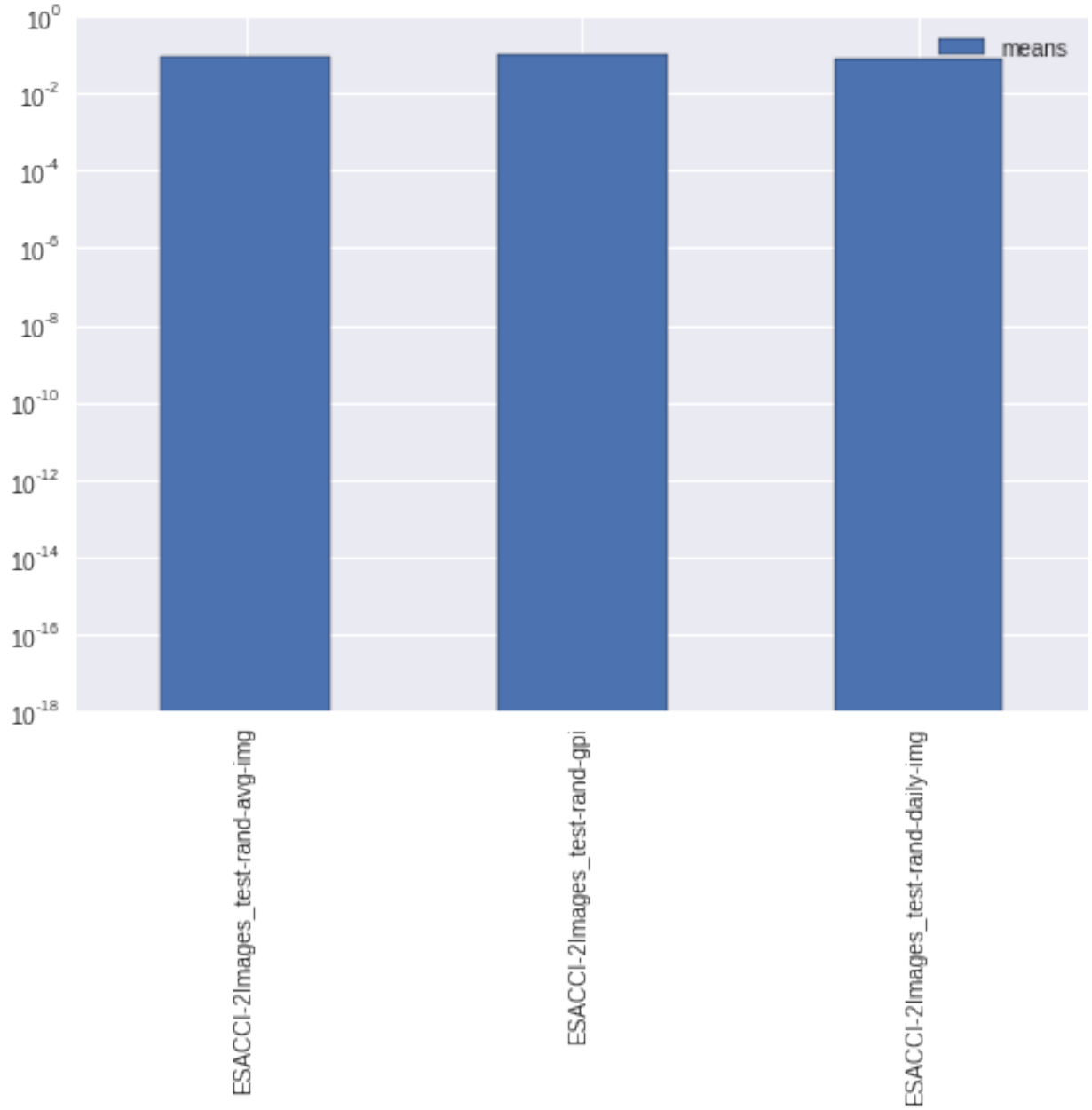
```
!ls /home/pydev/tests
```

```
ESACCI-2Images_test-rand-avg-img.nc    ESACCI-2Images_test-rand-gpi.nc
ESACCI-2Images_test-rand-daily-img.nc
```

Visualization of the results

```
%matplotlib inline
import glob
import smdc_perftests.performance_tests.analyze as analyze

# get all the files in the results folder
fs = glob.glob(os.path.join(res_dir, "*.nc"))
df = analyze.prep_results(fs)
# this returnes the mean times at the moment
print df
# and makes a very simple bar plot
ax = analyze.bar_plot(df)
```

```
                                    means
ESACCI-2Images_test-rand-avg-img    0.085946
ESACCI-2Images_test-rand-gpi        0.098265
ESACCI-2Images_test-rand-daily-img  0.059122
```



2.2 License

```
# Copyright (c) 2014,Vienna University of Technology,
# Department of Geodesy and Geoinformation
# All rights reserved.

# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#    * Redistributions of source code must retain the above copyright
#      notice, this list of conditions and the following disclaimer.
#    * Redistributions in binary form must reproduce the above copyright
#      notice, this list of conditions and the following disclaimer in the
```

```
#       documentation and/or other materials provided with the distribution.
#     * Neither the name of the Vienna University of Technology,
#       Department of Geodesy and Geoinformation nor the
#       names of its contributors may be used to endorse or promote products
#       derived from this software without specific prior written permission.

# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
# ARE DISCLAIMED. IN NO EVENT SHALL VIENNA UNIVERSITY OF TECHNOLOGY,
# DEPARTMENT OF GEODESY AND GEOINFORMATION BE LIABLE FOR ANY
# DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
# ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
# (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

2.3 smdc_perftests

2.3.1 smdc_perftests package

Subpackages

smdc_perftests.datasets package

Submodules

smdc_perftests.datasets.EQUI_7 module

smdc_perftests.datasets.ascat module

smdc_perftests.datasets.esa_cci module

Module contents

smdc_perftests.performance_tests package

Submodules

smdc_perftests.performance_tests.analyze module

smdc_perftests.performance_tests.test_cases module

smdc_perftests.performance_tests.test_scripts module

Module contents

Submodules

smdc_perftests.helper module

Helper functions Created on Wed Apr 1 14:50:18 2015

@author: christoph.paulik@geo.tuwien.ac.at

smdc_perftests.helper.**generate_date_list**(*minimum*, *maximum*, *n=500*, *max_spread=30*, *min_spread=None*)

Parameters **minimum: datetime**

minimum datetime

maximum: datetime

maximum datetime

n: int

number of dates to generate

max_spread: int, optional

maximum spread between dates

min_spread: int, optional

minimum spread between dates

Returns date_list: list

list of start, end lists The format is a list of lists e.g. [[datetime(2007,1,1), datetime(2007,1,1)],

[datetime(2007,1,1), datetime(2007,12,31)]]

smdc_perftests.visual module

Module for visualizing the test results Created on Tue Nov 25 13:44:56 2014

@author: Christoph.Paulik@geo.tuwien.ac.at

smdc_perftests.visual.**plot_boxplots**(**args*, ***kwargs*)

plots means and confidence intervals of given TestResults objects

Parameters ***args: TestResults instances**

any Number of TestResults instances that should be plotted side by side

conf_level: int, optional

confidence level to use for the computed confidence intervals

****kwargs: varied**

all other keyword arguments will be passed on to the plt.subplots function

Returns fig: matplotlib.Figure

ax1: matplotlib.axes

Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

S

G

P

S