Reports

Release 0.3.1

Thomas Cokelaer

February 16, 2017

Contents

Current version: 0.3.1, February 16, 2017

Python version Python 2.7, 3.4 and 3.5

documentation On readthedocs

Issues Please fill form on On github

Reports is a simple Python package that provides tools to create HTML documents. It is based on a set of JINJA templates and a class called **Report**. In addition tools such as HTMLTable can help in the creation of HTML table to be included in the report.

The package relies on Pandas for the HTML table creation, as shown in the example below.

We provide a simple JINJA example (stored with the pacakge in ./share/data/templates/generic directory) and we let the users design their own templates.

This is used in GDSCTools and Sequana packages.

CHAPTER 1

Usage

Example

Here below, we show the code used to create this example.

```
# We will create a Report and insert an HTML table in it
from reports import Report, HTMLTable

# Let us create some data. It will be a HTML table built using Pandas
# but you could create the HTML table code yourself.
import pandas as pd

# create a dataframe to play with. Note that there is a numeric column
# In addition, there is a column (Entry name) that will be transformed into URLs
df = pd.DataFrame({
    "Entry name":["ZAP70_HUMAN", "TBK1_HUMAN"],
    "Entry": ["P43403", "Q9UHD2"],
    "Frequency": [0.5,0.9]})

# From reports, we convert the dataframe into a HTMLTable
table = HTMLTable(df)

# a numeric column can be colorized
table.add_bgcolor('Frequency', cmap="copper")

# part of URLs can be added to the content of a column
table.add_href('Entry', url='http://uniprot.org/uniprot/', suffix="")
html = table.to_html()

# Create a generic report. It has a set of tags that can be filled
# using the **jinja** attribute.
r = Report("generic")

# set the **summary** tag with the HTML code of the table
r.jinja['summary'] = html

# Generate and show the report
r.create_report(onweb=True)
```

See the results in example

Using your own JINJA template

Create a directory called **test** and add a file called **index.html**

Add this code:

```
<h1> {{ title }} </h1>
<p> Number of reads : {{ reads }} </p>
```

Now, create your HTML files:

```
from reports import Report
report = Report("test")
report.jinja['title'] = 'Simple Example'
report.jinja['reads'] = "123456"
report.create_report(onweb=True)
```

CHAPTER 2

Issues

Please fill bug report in https://github.com/cokelaer/reports

CHAPTER 3

Contributions

Please join https://github.com/cokelaer/reports

References

Contents

Reports

Base classes to create HTML reports easily

class **Report** (*searchpath=None*, *filename='index.html'*, *directory='report'*, *overwrite=True*, *verbose=True*, *template_filename='index.html'*, *extra_css_list=[]*, *extra_js_list=[]*, *init_report=True*)

A base class to create HTML pages

The `Report` is used to

1. fetch Jinja templates and css from a user directory (by default a generic set of files is provided as an example
2. fetch the CSS and images
3. hold variables and contents within a dictionary (`jinja`)
4. Create the HTML document in a local directory.

```
from report import Report
r = Report()
r.create_report(onweb=True)
```

The next step is for you to copy the templates in a new directory, edit them and fill the `jinja` attribute to fulfil your needs:

```
from report import Report
r = Report("myreport_templates")
r.jinja["section1"] = "<h1></h1>"
r.create_report()
```

Constructor

Parameters

- **searchpath** – where to find the jina templates. If not provided, uses the generic template
- **filename** – output filename (default to **index.html**)
- **directory** – defaults to **report**
- **overwrite** – default to True
- **verbose** – default to True
- **template_filename** – entry point of the jinja code
- **extra_css_list** – where to find the extra css
- **extra_js_list** – where to find the extra css
- **init_report** (*bool*) – init the report that is create the directories to store css and JS.

abspath
The absolute path of the document (read only)

add_dependencies = None
flag to add dependencies

create_report (*onweb=True*)

directory
The directory where to save the HTML document

filename
The filename of the HTML document

get_table_dependencies (*package='reports'*)
Returns dependencies of the pipeline as an HTML/XML table

The dependencies are the python dependencies as returned by pkg_resource module.

get_time_now()
Returns a time stamp

onweb()
Open the HTML document in a browser

to_html()

write()

HTMLTable

Base classes to create HTML reports easily

class HTMLTable (*df*, *name=None*, ***kargs*)

Handler to export dataframe into HTML table.

Dataframe in Pandas already have a to_html method to export the dataframe into a HTML formatted table. However, we provide here a few handy features:

- Takes each cell in a given column and creates an HTML reference in each cell. See *add_href()* method.
- add an HTML background into cells (numeric content) of a given column using different methods (e.g., normalise). See *add_bgcolor()*

```
import pandas as pd
df = pd.DataFrame({'A':[1,2,10], 'B':[1,10,2]})
from gdsctools import HTMLTable
html = HTMLTable(df)
```

Note: similar project exists such as prettytable but could not do exactly what we wanted at the time gdsctools was developed.

Note: Could be moved to biokit or easydev package.

Constructor

Parameters

- **df** (*dataframe*) – a pandas dataframe to transform into a table
- **name** (*str*) – not used yet

There is an `pd_options` attribute to reduce the max column width or the precision of the numerical values.

add_bgcolor (*colname*, *cmap='copper'*, *mode='absmax'*, *threshold=2*)

Change column content into HTML paragraph with background color

Parameters

- **colname** –
- **cmap** – a colormap (matplotlib) or created using colormap package (from pypi).
- **mode** – type of normalisation in 'absmax', 'max', 'clip' (see details below)
- **threshold** – used if mode is set to 'clip'

Colormap have values between 0 and 1 so we need to normalised the data between 0 and 1. There are 3 mode to normalise the data so far.

If mode is set to 'absmax', negatives and positives values are expected to be found in a range from -inf to inf. Values are scaled in between [0,1] X' = (X / M +1) /2. where m is the absolute maximum. Ideally a colormap should be made of 3 colors, the first color used for negative values, the second for zeros and third color for positive values.

If mode is set to 'clip', values are clipped to a max value (parameter *threshold* and values are normalised by that same threshold.

If mode is set to 'max', values are normalised by the max.

add_href(*colname*, *url=None*, *newtab=False*, *suffix=None*)
default behaviour: takes column content and put into:

```
<a href={content}.html>content</a>
```

This is used to link to local files. If url is provided, you typically want to link to an external url where the content is an identifier:

```
<a href={url}{content}>content</a>
```

Note that in the first case, *.html* is appended but not in the second case, which means cell's content should already have the .html Also in the second case, a new tab is open whereas in the first case the url is open in the current tab.

Note: this api may change in the future.

---

sort(*name*, *ascending=True*)

to_html(*index=False*, *escape=False*, *header=True*, *collapse_table=True*, *class_outer='table_outer'*, ***kargs*)
Return HTML version of the table

This is a wrapper of the to_html method of the pandas dataframe.

Parameters

- **index** (*bool*) – do not include the index
- **escape** (*bool*) – do not escape special characters
- **header** (*bool*) – include header
- **collapse_table** (*bool*) – long tables are shorten with a scroll bar
- **kargs** – any parameter accepted by `pandas.DataFrame.to_html()`

CHAPTER 4

Changelog

changelog

version 0.3.1 Fix License and RTD

version 0.3.0 Fix call to easydev.precision for the case where data contains infinite/nan values

version 0.2.1 Fix warnings due to division by zero; add some tests

version 0.2.0 outer class of the table is always used (no check of a minimal size)

version 0.1.9 Add new option to not create the sub directories (css/js...)

version 0.1.8 fix another bug/typo in js_list

version 0.1.7 fix bug/typo in js_list

version 0.1.6 extra js can now be added (similarly to the CSS)

version 0.1.5 get_dependencies now accept an input package argument (defaults to reports)

version 0.1.4 HTMLTable sorting is confused with the content of scientific notation that are used as characters. The user should change te dataframe types but we change the type is we can from object to float.

Second fix is related to CSS. We were already including CSS from reports/resources/css, from a list of user-defined CSS. We now also include CSS found in the JINJA searchpath provided by the user.

version 0.1.3

- change css_path into extra_css_list parameter
- change parameter names to allow to get all set of jinja files instead of just one.

version 0.1.2

- refactoring: moved ./src/reports into ./reports and mv ./share into ./reports this was done to ease the modifications of setup.py so that the templates can be accessed to using python setupy install or develop mode AND pip install.

version 0.1.1

- added jquery javascript

version 0.1

- first functional release used in sequana project and gdsctools project

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

h

r

A

C

D

F

G

H

O

R

S

T

W