Pynoramix Documentation

Release 0.1

Diego Prada-Gracia

June 25, 2012

CONTENTS

Contents:

CHAPTER

ONE

GETTING STARTED

1.1 Getting Pynoramix

1.1.1 Last stable version (0.1)

-Adding the project in easy_install or setup.py (http://packages.python.org/an_example_pypi_project/setuptools.html#registering-your-project)

-Links to raolab.com or GitHub from raolab.

1.1.2 Source Code

Document the project in GitHub and add the link.

1.2 Installing

Pynoramix depends on some packages:

- NumPy

CHAPTER

TWO

GETTING STARTED

2.1 Installing your doc directory

You may already have sphinx sphinx installed – you can check by doing:

```
python -c 'import sphinx'
```

If that fails grab the latest version of and install it with:

```
> sudo easy_install -U Sphinx
```

Now you are ready to build a template for your docs, using sphinx-quickstart:

```
> sphinx-quickstart
```

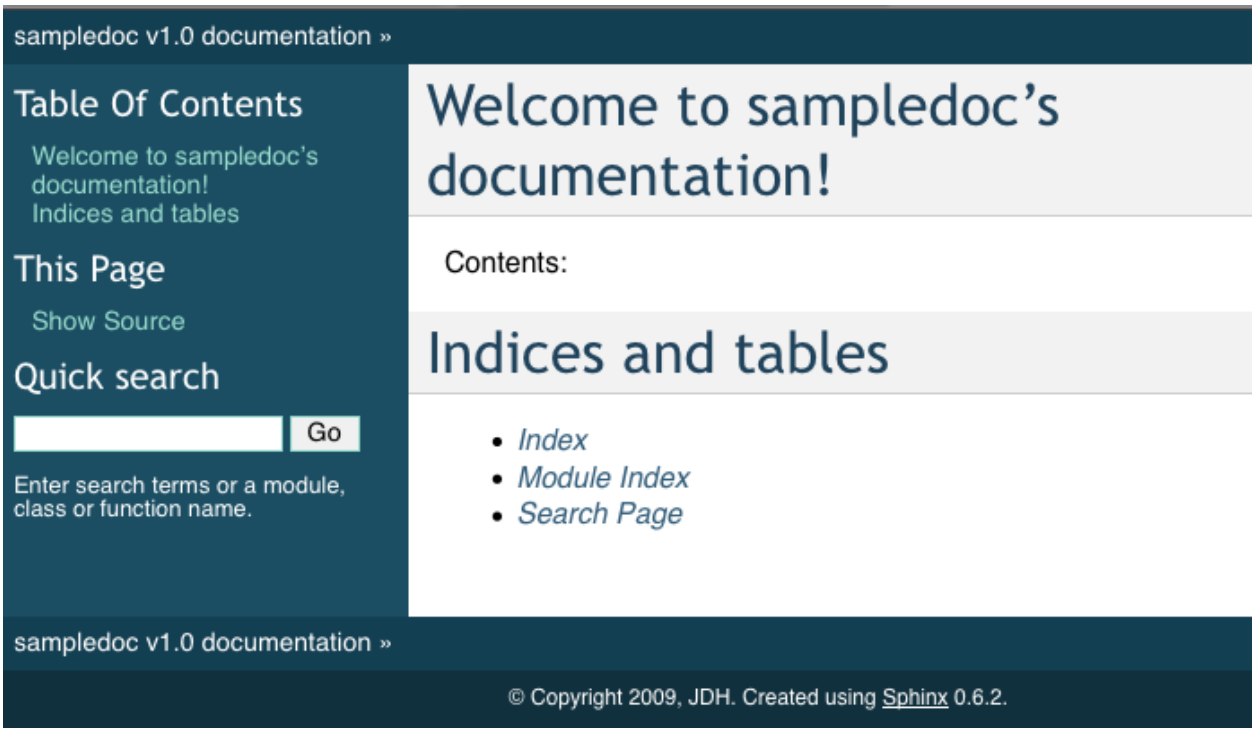
accepting most of the defaults. I choose "sampledoc" as the name of my project. cd into your new directory and check the contents:

```
home:~/tmp/sampledoc> ls
Makefile      _static         conf.py
_build                _templates      index.rst
```

The index.rst is the master ReST for your project, but before adding anything, let's see if we can build some html:

```
make html
```

If you now point your browser to `_build/html/index.html`, you should see a basic sphinx site.



2.1.1 Fetching the data

Now we will start to customize out docs. Grab a couple of files from the web site or svn. You will need `getting_started.rst` and `_static/basic_screenshot.png`. All of the files live in the "completed" version of this tutorial, but since this is a tutorial, we'll just grab them one at a time, so you can learn what needs to be changed where. Since we have more files to come, I'm going to grab the whole svn directory and just copy the files I need over for now. First, I'll cd up back into the directory containing my project, check out the "finished" product from svn, and then copy in just the files I need into my `sampledoc` directory:

```
home:~/tmp/sampledoc> pwd
/Users/jdhunter/tmp/sampledoc
home:~/tmp/sampledoc> cd ..
home:~/tmp> svn co https://matplotlib.svn.sourceforge.net/svnroot/\
matplotlib/trunk/sampledoc_tut
A    sampledoc_tut/cheatsheet.rst
A    sampledoc_tut/_static
A    sampledoc_tut/_static/basic_screenshot.png
A    sampledoc_tut/conf.py
A    sampledoc_tut/Makefile
A    sampledoc_tut/_templates
A    sampledoc_tut/_build
A    sampledoc_tut/getting_started.rst
A    sampledoc_tut/index.rst
Checked out revision 7449.
home:~/tmp> cp sampledoc_tut/getting_started.rst sampledoc/
home:~/tmp> cp sampledoc_tut/_static/basic_screenshot.png \
sampledoc/_static/
```

The last step is to modify `index.rst` to include the `getting_started.rst` file (be careful with the indentation, the "g" in "getting_started" should line up with the ':' in `:maxdepth`:

```
Contents:

.. toctree::
   :maxdepth: 2

   getting_started.rst
```

and then rebuild the docs:

```
cd sampledoc
make html
```

When you reload the page by refreshing your browser pointing to `_build/html/index.html`, you should see a link to the "Getting Started" docs, and in there this page with the screenshot. *Voila!*

Note we used the image directive to include to the screenshot above with:

```
.. image:: _static/basic_screenshot.png
```

Next we'll customize the look and feel of our site to give it a logo, some custom css, and update the navigation panels to look more like the sphinx site itself – see *custom_look*.

CHAPTER

THREE

TUTORIALS AND EXAMPLES

First of all, lets load Pynoramix in our script or in a ipython session:

```
In [1]: from pynoramix_beta import *
```

Some basic notions on python will be assumed along this tutorial. If you just landed here without any idea on python, have a look to the section *First steps on python*.

Todo

Make a short tutorial on python, enough to run pynoramix.

3.1 Networks

How to create, load and handle a network.

3.1.1 A network from scratch

Lets create as simple example a network of cities:

```
In [2]: cities=net()
# Network:
# 0 nodes
# 0 links out
# 0 total weight nodes
```

Nodes can be added in two ways, along or inferred by the links addition:

```
In [3]: cities.add_node('Zaragoza')
In [4]: cities.add_link('Rome','Turin',446)
In [5]: cities.add_link('Rome','Zaragoza',1112)
In [6]: cities.add_link('Zaragoza','Kiev',2561)
In [7]: cities.info()
# Network:
# 4 nodes
# 3 links out
# 0 total weight nodes
```

The nodes are attached to the network in order of creation:

```
In [7]: cities.node[0].label
Out[7]: 'Zaragoza'
In [8]: cities.node[3].label
Out[8]: 'Kiev'
```

Links are stored as a dictionary for each node (see ref). Nodes are from now on referred because of their indexes.

```
In [9]: cities.labels['Rome']
Out[9]: 1
In [10]: cities.node[1].link.keys()
Out[10]: [0, 2]
In [11]: print 'From Rome to Zaragoza: ', cities.node[1].link[0], 'km.'
From Rome to Zaragoza:  1112 km.
```

See Also:

include here a link to the class definition and attributes.

3.1.2 Loading a network from a file

A network can be loaded together with their labels. Pynoramix uses its own compact format for the network, while the labels can be readed with many formats. This way a network can be initialized with the files or a posteriori:

CHAPTER

FOUR

NETWORKS

Description of the network object and attributes. This is a test to include the comments from the code.

nada

4.1 Network class

class pyn_cl_net.**cl_node**

Fundamental unit to constitute a network together with links.

Variables

- **label** – label or key
- **weight** – weight
- **link** – linked nodes, {node_index, weight_link}.
- **k_out** – degree or connectivity with directed links out.
- **k_in** – degree or connectivity with directed links in.
- **k** – total degree or connectivity.
- **cluster** – index of cluster the node belongs to
- **component** – index of disconnected component the node belongs to
- **coors** – spatial coordinates for representation

most_weighted_links(*length=1*)

Ranked indexes of connected nodes according to the weight of the links.

Parameters **length** (*int*) – N number of links requested

Returns ranked node indexes

Return type N-dim list [int]

class pyn_cl_net.**net**(*file_net=None*, *file_keys=None*, *directed=True*, *labels_format='2columns'*, *verbose=True*)

Supra-structure composed by nodes

add_node(*new_node*, *weight=0*)

Adding a new node to the network.

Parameters

- **new_node** (*str,int,float...*) – key or label of the new node

- **weight** (int, *opt*) – weight of new node

info()
Print network general variables: num_nodes, k_total, weight.

..:param arg1: description ..:param arg2: description ..:type arg1: type description ..:type arg1: type description ..:return: return description ..:rtype: the return type description ..:Example: (followed by a blank line)

CHAPTER

FIVE

EJEMPLO DE TITULO DE CAPITULO

There should only be one of these per page and this will also – when converting to pdf – be used for the chapters.

5.1 Como itemizo

Ejemplos de itemizes.

5.1.1 Ejemplos:

Aqui los ejemplos.

Ejemplos 1

- A thing.
- Another thing.

Ejemplo 2

1. Item 1.
2. Item 2.
3. Item 3.

Ejemplo 2

- Some.
- Thing.
- Different.

5.2 Page Sections

Ejemplos de formato:

bold and *italics*

5.3 Marking paragraphs

This is a statement.

> **Warning:** Never, ever, use this code!

New in version 0.0.1. It's okay to use this code.

Here is something I want to talk about:

```
def my_fn(foo, bar=True):
    """A really useful function.

    Returns None
    """
```

bla,bla

This is inline `if __name__ == '__main__':`

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

p

INDEX