
nbdime Documentation

Release 0.1.0

Martin Sandve Alnæs and Project Jupyter

December 05, 2016

1	Abstract	3
2	Quickstart	7
2.1	Git integration quickstart	7
3	Contents	11
3.1	Installation	11
3.2	nbdime commands	13
3.3	Version control integration	19
3.4	Testing	22
3.5	Glossary	23
3.6	Use cases	23
3.7	diff format	24
3.8	Merge details	28
3.9	REST API draft for nbdime server v0.1	30

Version: 0.1

nbtime provides tools for diffing and merging Jupyter notebooks.

Loading Matplotlib demos with %load

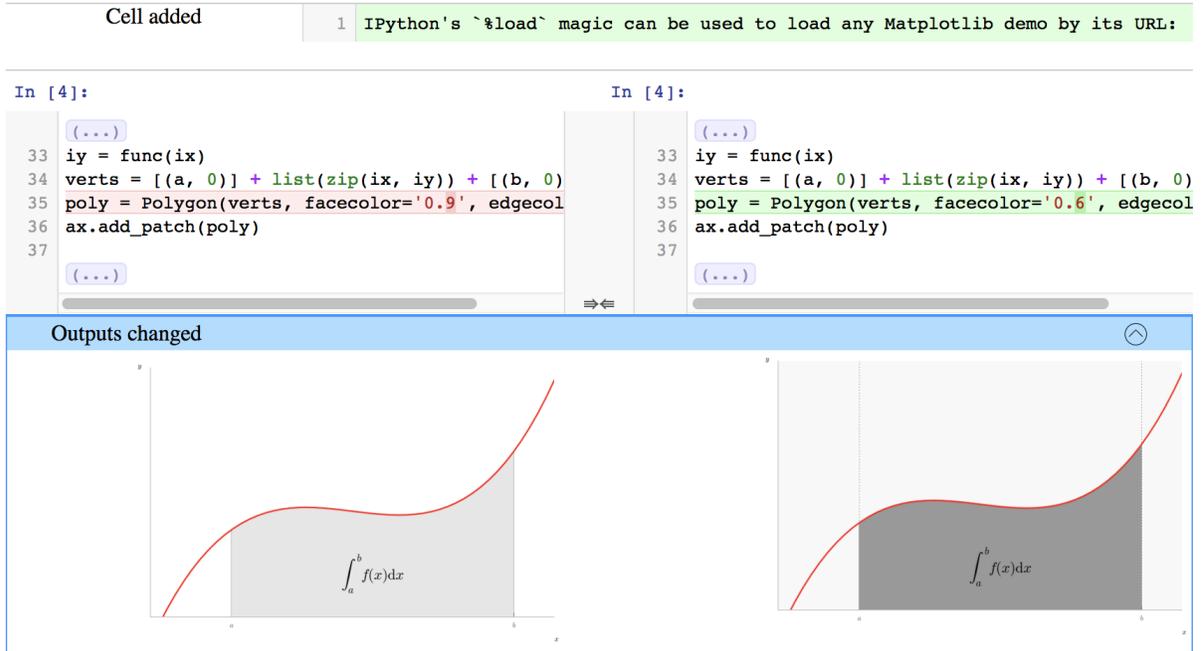


Fig. 1: Figure: nbtime example

Abstract

Jupyter notebooks are useful, rich media documents stored in a plain text JSON format. This format is relatively easy to parse. However, primitive line-based diff and merge tools do not handle well the logical structure of notebook documents. These tools yield diffs like this:

nbdime, on the other hand, provides “content-aware” diffing and merging of Jupyter notebooks. It understands the structure of notebook documents. Therefore, it can make intelligent decisions when diffing and merging notebooks, such as:

- eliding base64-encoded images for terminal output
- using existing diff tools for inputs and outputs
- rendering image diffs in a web view
- auto-resolving conflicts on generated values such as execution counters

nbdime yields diffs like this:

```

$ diff a.ipynb b.ipynb
76,77d75
<     "plt.rc('axes', grid=False)\n",
<     "plt.rc('axes', facecolor='white')\n",
90c88
<     "image/png": "iVBORw0KGgoAAAANSUHEUgAABLkAAAMQCAyAAADLj7dLAAAABHNCSVQICAgIFAhki
AAAAAlwSFLz\nAAAWJQAFAFiUBSVIk8AAAIABJREFUeJzsvXeYZFd57b12h0maPNJII2lG0aCAKEBCFgozIxBAp
lY\n1waDyDZg8MX+zMU2F4Mx1x8PwAwxBjg4yNi2BFQMa20iiAQFKIjXKWRtJIE3tSz3TXuX+8vV2n\nnqyucv
N+9z/o9zzynprvq1D6nqqqr1prbRNFEQghhBBCCGEEEEII8Zkh1wMghBBCCGEEEEIIISQv\nnFlkIIYQQQgghhB
BCiPdQ5CKEEEEIIYQQQggh3kORixBCCCGEEEEIIYR4D0UuQgghhBBCCGEEEEII9\nnFlkIIYQQQgghhBBCiPdQ5CK
EEEEIIYQQQggh3kORixBCCCGEEEEIIYR4D0UuQgghhBBCCGEEEEII9\nnFlkIIYQQQgghhBBCiPdQ5CKEEEEIIYQQ
Qggh3kORixBCCCGEEEEIIYR4D0UuQgghhBBCCGEEEEII9\nnFlkIIYQQQjzEGH0JMaZljPmo67EkZwq8D7keByGEE
ELChCIXIYQQQirDGPOMkaFj3BhzkMnx/H/G\nnmG3GmP/pagwFEbkeQJUY75gjNljHmD67EQgghRB8UuQgghhB
BSJe+DCDMjAH7L4TjeAmA+gLc5\nnHEMRGncDqj3AVGI4Ddd4DQQQggh+qDIRQghhJBKMMacCuBMAFsg4sy7jTH
DjobzZwBuBvBxR/dP\nnsVERADcC+LTrgRBCCCFEHxS5CCGEEFIVH4C4uP4SIlQcBOD1LgYSRVEziqIXR1H0frf3
T7IRRDff\nnRlH0K1EUXe96LIQQQgjRB0UuQgghhJSOMWYpgP8BoAXg7wH8HcTN9Tsxu0UIIYQQQsKBIhchhBBC\
nquBdA0YAuDyKoscBfBvALgBnGW0e73RkhBBCCEkCCChyEUIIIaRUjDEGUjI fQRxc iKJoDMB3p65C\nnNxchhBBC
CMkNRS5CCCGELM3FAA4HsAnAD2I/t5HFNxpjFuW5A2PMXGPMh4wxNxpjxowx04wxdxpj\nnPmGMmd/l+pcYY1rGm
I92/HzB1M8np/5/mDHmr40xjxpjdhtj7jbGfMwYMy92m3ONMT80xmw0xuyx\nnus6njDFLeozV3veHpv5/LDHm88
aY9V0332yMulwnqfg4o4Lz8njHmp1P73WmMuxfq/g7Ls++p/c8x\nnrxzFGH0lMeYZY8weY8xjxphvGmPOGHDbQ4w
xf2GMeWdQdk8ZY75vjDmzz20+NnXu3tLld48YY/ZO\nnXT7KGPMPxpHjDETphvT/38rVwde0IIIIYRU0UuQggh
hJTN70BcXJdGUBTP/jCKousA3A9gLoB3\nnZn35LEB2I4D/H8ApU/u8D8BxAP4YwG09RLSoy892xPb7EgC/APBWA
E8DuBfAUQA+CuByY8yoMeZ/\nArgKwDkAHgRwD4AjAPwBgJ/2Ee+ift4L4BfAng/gEUAbgfwKICTp+7nIwPMG5
Och06MMadNjfkz\nnAE4H8PjU/hd03d/dxpg3Zdn31P7PhJzrLwBYDvk18xaIcPkmADcYY/68x23PhZzb3wEwBuA
OAPMB\nnvBrAtcaYV/e42wjdHzf70xhjXghZ0fPNAMYB/BTALZ23L/PcE0IIicQNFkIIYQQUhrGmGMBRj36\nn79
e7XOXvIaLIE3LczUcBnAhgHYBDoyg6LYqi0wA8DyI6rYSIMDOG1/mDKIriAsp3AFwH4JcPlRhF\nnA0BoiHB0JoC
vAfg0gK8CWDG16t8LAawC8DMAxwL4kx5jNgAaL4E4DYAL4mi6JAoiL4SRdGpAA4A\nn8HEAswFcaoz5HwnPhezC
mGMg4tshkP6zLVEUnTK1/xUAXglgK4BvGGPOT7Pvqf2fB0ByAAcDaAI4\nnIoqi46f2vxLARwF4FsC7psYSZxEkq
norgC0jKDo1iqIXT3z3wIYAfAVY8xo2nFBzuvXATwG4JQo\nnip4XRdG5URR9reM6pZ17QgghhLiDIhchhBBCyu
T9EFHhxiK7ury+28AmARwpDHm5Rnv43yIO+fp\nnoyh61v4wiqIHic6g06Mo+r8Z9jsG4LVRF6207fMxiHBlALw
FwPVRFL03iqLdsetsBPB7U9f59T77\nnPxxAZQD0jaLoZ/FfRFE0FkXRxyEuMgPgb4wxy1KM/w8ALADw4Sik3hhf
0dMd+/8xgJdDztuXU+zX\nn8jUA8wB8PYqiX586L/H9/wjASwG8MIqi+ztuuxjAkwBeMbUIgb3NHgDvhQhUBWBYk
2Fcw/a2URT9\nnss/1yjz3hBBCHEERS5CCCGELIIXZj9IZ0y/C+c7iaLoKQA/hogJWQvorfvq2C77vy+Kojysz7v
fT\nnURTt7fJz2yswafjLrgMS4WQjgAP7Rba3A7gkiqKJXg0IouiFAPwIilj9dpJBG2N0gtjXbo+i6NN9\nn9n0H5
Fi0nsac12TFU/s/HcCLAWwD8ME++/9FFEW9vJ1R+PR1dhtJtA+v89L0qb4LgB8Ji5M9qCU\nnc08IIYQQt1DkIo
QQQkhZvBXS/7QLEpnrhRXALjTGHJnhfr4NEck+Z4z5G2PMGmPMrAz76eSKbj+c\nnchxtmfrvbX1u/+TudKGP338
ngRgDAN+EHN8rElwXEAcVIE6lQdw8te9zEu4baMdPL4uiaEffa3an\nnBeA/+vz+vqkx9Tpvq7ghwXXK0veEEEEII
cQhFlkIIISUXSgzprvDhBDLgPwDERMeF+G+/k0pPwc\nnAN4JKRnfZoy51hjzQWPMwz7BKRsvhfbp7ZPJrhOr
89bSR1m66e2hye8vl018SNTkwn2/Afgk5DH\nn60CE+wak4yyCdJNLVXM3F1cM+1zJ+jm132NiKevcE0IIicQhI6
4HQAgghJDwmCozPwEihlxijLkk\nn4U3faoz5SLzjahBRFLUAFNAY8wUARwdwHoCzALwEwNkaFscYc0EURQ+n0YY
px9ag6/QTa1xzH4AN\nnCa/7SIr92sL+XqscDmLNgN9n3S8hhBBCag5FLkIIISUge3XGkfbmTOIZZBS8jdBvtLL

```

Fig. 1.1: Figure: diff using traditional line-based diff tool

Loading Matplotlib demos with %load

Cell added

```
1 IPython's `%load` magic can be used to load any Matplotlib demo by its URL:
```

In [4]:

```
(...)
```

```
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.9', edgecolor='0.9')
36 ax.add_patch(poly)
37
```

(...)

In [4]:

```
(...)
```

```
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.6', edgecolor='0.6')
36 ax.add_patch(poly)
37
```

(...)

⇒⇐

Outputs changed

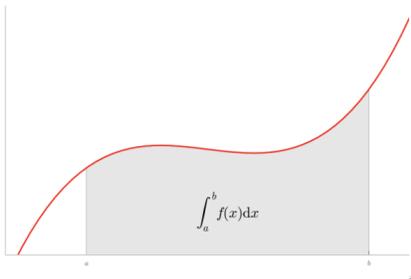
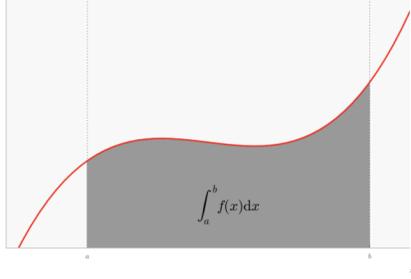



Fig. 1.2: Figure: nbdime's content-aware diff

Quickstart

To get started with nbdime, install with pip:

```
pip install nbdime
```

And you can be off to the races by diffing notebooks in your terminal with **nbdiff**:

```
nbdiff notebook_1.ipynb notebook_2.ipynb
```

or viewing a rich web-based rendering of the diff with **nbdiff-web**:

```
nbdiff-web notebook_1.ipynb notebook_2.ipynb
```

For more information about nbdime's commands, see [nbdime commands](#).

2.1 Git integration quickstart

Many of us who are writing and sharing notebooks do so with git and GitHub. Git doesn't handle diffing and merging notebooks very well by default, but you can configure git to use nbdime and it will get a lot better.

To configure git to use nbdime to as a command-line driver to diff and merge notebooks:

```
git-nbdiffdriver config --enable --global  
git-nbmergedriver config --enable --global
```

Now when you do **git diff** or **git merge** with notebooks, you should see a nice diff view, like this:

To configure git to use the web-based GUI viewers of notebook diffs and merges:

```
git-nbdiffftool config --enable --global  
git-nbmergetool config --enable --global
```

With these, you can trigger the **tools** with:

```
git diffftool --tool nbdime [ref [ref]]
```

and:

```
git mergetool --tool nbdime
```

Note: Using **git-nbdiffdriver config** overrides the ability to call **git diffftool** with notebooks.

You can still call **nbdiff-web** to diff files directly, but getting the files from git refs is still on our TODO list.

```
$ nbdiff c.ipynb b.ipynb
nbdiff c.ipynb b.ipynb
--- c.ipynb 2016-11-30 15:12:21
+++ b.ipynb 2016-11-30 15:12:30
## modified /cells/9/outputs/0/data/text/plain:
- <matplotlib.figure.Figure at 0x10ea05940>
+ <matplotlib.figure.Figure at 0x10eb21860>

## replaced /cells/14/outputs/0/data/image/png:
- iVBORw0K...<snip base64, md5=3f7d4e61ee33aaae...>
+ iVBORw0K...<snip base64, md5=1d6960ad89e9de61...>

## modified /cells/14/outputs/0/data/text/plain:
- <matplotlib.figure.Figure at 0x1110200b8>
+ <matplotlib.figure.Figure at 0x11112bf28>

## modified /cells/14/source:
@@ -25,14 +25,14 @@ x = np.linspace(0, 10)
y = func(x)

fig, ax = plt.subplots()
plt.plot(x, y, 'g','r', linewidth=2)
plt.ylim(ymin=0)

# Make the shaded region
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
poly = Polygon(verts, facecolor='0.9', edgecolor='0')facecolor='0.6', edgecolor='0.5')
ax.add_patch(poly)
```

Fig. 2.1: Figure: nbdime’s ‘content-aware’ command-line diff

Loading Matplotlib demos with %load



Fig. 2.2: Figure: nbtime's content-aware diff

For more detailed information on integrating nbtime with version control, see [Version control integration](#).

Loading Matplotlib demos with %load

Cell deleted locally

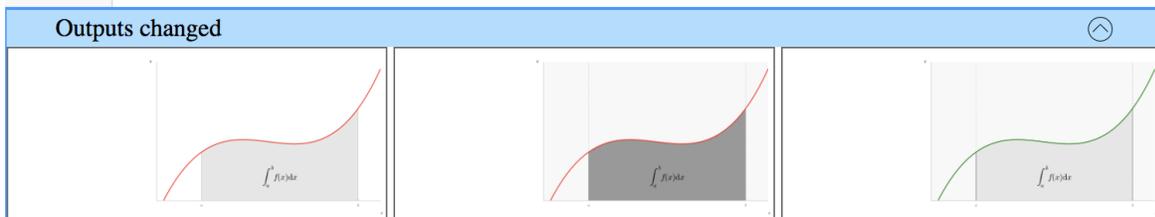
Delete cell

1 IPython's `%load` magic can be used to load any Matplotlib demo by

		<input type="checkbox"/> Delete cell	
<pre> 26 (...) 27 fig, ax = plt.subplots 28 plt.plot(x, y, 'r', li 29 plt.ylim(ymin=0) 30 31 # Make the shaded regi 32 ix = np.linspace(a, b) 33 iy = func(ix) 34 verts = [(a, 0)] + lis 35 poly = Polygon(verts, : 36 ax.add_patch(poly) 37 38 (...) 39 40 (...) 41 plt.figtext(0.9, 0.05, 42 plt.figtext(0.1, 0.9, 43 44 ax.spines['right'].set_ 45 ax.spines['top'].set_v 46 (...) 47 48 (...) 49 50 (...) 51 plt.show() 52 53 </pre>	<pre> 26 (...) 27 fig, ax = plt.subplots 28 plt.plot(x, y, 'r', li 29 plt.ylim(ymin=0) 30 31 # Make the shaded regi 32 ix = np.linspace(a, b) 33 iy = func(ix) 34 verts = [(a, 0)] + lis 35 poly = Polygon(verts, : 36 ax.add_patch(poly) 37 38 (...) 39 40 (...) 41 plt.figtext(0.9, 0.05, 42 plt.figtext(0.1, 0.9, 43 44 ax.spines['right'].set_ 45 ax.spines['top'].set_v 46 (...) 47 48 (...) 49 50 (...) 51 plt.show() 52 53 </pre>	<pre> 26 (...) 27 fig, ax = plt.subplots 28 plt.plot(x, y, 'g', li 29 plt.ylim(ymin=0) 30 31 # Make the shaded regi 32 ix = np.linspace(a, b) 33 iy = func(ix) 34 verts = [(a, 0)] + lis 35 poly = Polygon(verts, : 36 ax.add_patch(poly) 37 38 (...) 39 40 (...) 41 plt.figtext(0.9, 0.05, 42 plt.figtext(0.1, 0.9, 43 44 ax.spines['right'].set_ 45 ax.spines['top'].set_v 46 (...) 47 48 (...) 49 50 (...) 51 plt.show() 52 53 </pre>	

```

26 (...)
27 fig, ax = plt.subplots()
28 plt.plot(x, y, 'g', linewidth=2)
29 plt.ylim(ymin=0)
30
31 # Make the shaded region
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.6', edgecolor='0.5')
36 ax.add_patch(poly)
37
38 (...)
39
40 (...)
41 plt.figtext(0.9, 0.05, '$x$')
42 plt.figtext(0.1, 0.9, '$y$')
43
44 ax.spines['right'].set_visible(False)
45 ax.spines['top'].set_visible(False)
46 (...)
47
48 (...)
49
50 (...)
51 plt.show()
52
53
                    
```



3.1 Installation

3.1.1 Installing nbdime

To install the latest stable release using **pip**:

```
pip install --upgrade nbdime
```

Dependencies

nbdime requires Python version 3.3 or higher. If you are using Python 2, nbdime requires 2.7.1 or higher.

nbdime depends on the following Python packages, which will be installed by **pip**:

- six
- nbformat
- tornado
- colorama
- backports.shutil_which (on python 2.7)

and nbdime's web-based viewers depend on the following Node.js packages:

- codemirror
- json-stable-stringify
- jupyter-js-services
- jupyterlab
- phosphor

3.1.2 Installing latest development version

Installing a development version of nbdime requires [Node.js](#).

Installing nbdime using **pip** will install the Python package dependencies and will automatically run `npm` to install the required Node.js packages.

Setting up a virtualenv with Node.js

The following steps will: create a virtualenv, named `myenv`, in the current directory; activate the virtualenv; and install `npm` inside the virtualenv using `nodeenv`:

```
python3 -m venv myenv          # For Python 2: python2 -m virtualenv myenv
source myenv/bin/activate
pip install nodeenv
nodeenv -p
```

With this environment active, you can now install `nbdime` and its dependencies using `pip`.

For example with Python 3.5, the steps with output are:

```
$ python3 -m venv myenv
$ source myenv/bin/activate
(myenv) $ pip install nodeenv
Collecting nodeenv
  Downloading nodeenv-1.0.0.tar.gz
Installing collected packages: nodeenv
  Running setup.py install for nodeenv ... done
Successfully installed nodeenv-1.0.0
(myenv) $ nodeenv -p
* Install prebuilt node (7.2.0) ..... done.
* Appending data to /Users/username/myenv/bin/activate
(myenv) $
```

Using Python 2.7, the steps with output are (note: you may need to install `virtualenv` as shown here):

```
$ python2 -m pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
    100% || 1.8MB 600kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.1.0
$ python2 -m virtualenv myenv
New python executable in /Users/username/myenv/bin/python
Installing setuptools, pip, wheel...done.
$ source myenv/bin/activate
(myenv) $ pip install nodeenv
Collecting nodeenv
  Downloading nodeenv-1.0.0.tar.gz
Installing collected packages: nodeenv
  Running setup.py install for nodeenv ... done
Successfully installed nodeenv-1.0.0
(myenv) $ nodeenv -p
* Install prebuilt node (7.2.0) ..... done.
* Appending data to /Users/username/myenv/bin/activate
(myenv) $
```

Install the development version

Download and install directly from source:

```
pip install -e git+https://github.com/jupyter/nbdime
```

Or clone the [nbdime repository](https://github.com/jupyter/nbdime) and use `pip` to install:

```
git clone https://github.com/jupyter/nbdime
cd nbdime
pip install -e .
```

3.2 nbdime commands

nbdime provides the following CLI commands:

```
nbshow
nbdiff
nbdiff-web
nbmerge
nbmerge-web
```

Pass `--help` to each command to see help text for the command's usage.

Additional commands are available for *Git integration*.

3.2.1 nbshow

nbshow gives you a nice, terminal-optimized summary view of a notebook. You can use it to quickly peek at notebooks without launching the full notebook web application.

```
$ nbshow -s -o c.ipynb
markdown cell 0:
source:
  # Plotting with Matplotlib

  IPython works with the [Matplotlib](http://matplotlib.org/) plotting library,
  which integrates Matplotlib with IPython's display system and event loop
  handling.

  ## matplotlib mode

  To make plots using Matplotlib, you must first enable IPython's matplotlib
  mode.

  To do this, run the `%matplotlib` magic command to enable plotting in the
  current Notebook.

  This magic takes an optional argument that specifies which Matplotlib backend
  should be used.
  Most of the time, in the Notebook,
  you will want to use the `inline` backend, which will embed plots inside
  the Notebook:
```

code cell 1:

```
source:
  %matplotlib inline
  import matplotlib.pyplot as plt
  import numpy as np
```

code cell 2:

```
source:
  x = np.linspace(0, 3*np.pi, 500)
  plt.plot(x, np.sin(x**2))
  plt.title('A simple chirp');
```

outputs:

output 0:

```
output_type: display_data
data:
  image/png: iVBORw0K...<snip base64, md5=7665fcc01cfdaa71...>
  text/plain: <matplotlib.figure.Figure at 0x10ea05940>
metadata (unknown keys):
  image/png:
    height: 392
    width: 604
```

markdown cell 3:

3.2.2 Diffing

nbdime offers two commands for viewing the diff between two notebooks:

- **nbdiff** for command-line diffing
- **nbdiff-web** for rich web-based diffing of notebooks

See also:

For more technical details on how nbdime compares notebooks, see [diff format](#).

nbdiff

nbdiff does a terminal-optimized rendering of notebook diffs. Pass it the two notebooks you would like to compare, and it returns a nice, readable presentation of the changes in the notebook.

```
$ nbdiff c.ipynb b.ipynb
nbdiff c.ipynb b.ipynb
--- c.ipynb 2016-11-30 15:12:21
+++ b.ipynb 2016-11-30 15:12:30
## modified /cells/9/outputs/0/data/text/plain:
- <matplotlib.figure.Figure at 0x10ea05940>
+ <matplotlib.figure.Figure at 0x10eb21860>

## replaced /cells/14/outputs/0/data/image/png:
- iVBORw0K...<snip base64, md5=3f7d4e61ee33aaae...>
+ iVBORw0K...<snip base64, md5=1d6960ad89e9de61...>

## modified /cells/14/outputs/0/data/text/plain:
- <matplotlib.figure.Figure at 0x1110200b8>
+ <matplotlib.figure.Figure at 0x11112bf28>

## modified /cells/14/source:
@@ -25,14 +25,14 @@ x = np.linspace(0, 10)
y = func(x)

fig, ax = plt.subplots()
plt.plot(x, y, 'g','r', linewidth=2)
plt.ylim(ymin=0)

# Make the shaded region
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
poly = Polygon(verts, facecolor='0.9', edgecolor='0')facecolor='0.6', edgecolor='0.5')
ax.add_patch(poly)
```

nbdiff-web

Like **nbdiff**, **nbdiff-web** compares two notebooks.

Instead of a terminal rendering, **nbdiff-web** opens a web browser, compares the two notebooks, and displays the rich rendered diff of images and other outputs.

Loading Matplotlib demos with %load

Cell added

```
1 IPython's `%load` magic can be used to load any Matplotlib demo by its URL:
```

In [4]:

```
(...)
```

```
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.9', edgecolor='0.9')
36 ax.add_patch(poly)
37
```

(...)

In [4]:

```
(...)
```

```
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.6', edgecolor='0.6')
36 ax.add_patch(poly)
37
```

(...)

⇒⇐

Outputs changed

3.2.3 Merging

Merging notebook changes and dealing with merge conflicts are important parts of a development workflow. With notebooks, merging changes is a non-trivial technical task. Traditional, line-based tools can produce invalid notebooks that you have to fix by hand, which is no fun at all, or can risk unintended data loss.

nbtime provides some improved tools for merging notebooks, taking into account knowledge of the notebook file format to ensure that a valid notebook is always produced. Further, by understanding details of the notebook format, nbtime can automatically resolve conflicts on generated fields.

See also:

For more details on how nbtime merges notebooks, see [Merge details](#).

nbmerge

nbmerge merges two notebooks with a common parent. If there are conflicts, they are stored in metadata of the destination file. **nbmerge** will exit with nonzero status if there are any unresolved conflicts.

nbmerge writes the output to `stdout` by default, so you can use pipes to send the result to a file, or the `-o`, `--output` argument to specify a file in which to save the merged notebook.

Because there are several categories of data in a notebook (such as input, output, and metadata), **nbmerge** has several ways to deal with conflicts, and can take different actions based on the type of data with the conflict.

Important: Conflict-resolution in **nbmerge** is under active development and is subject to change.

The `-m`, `--merge-strategy` option lets you select a global strategy to use. The following options are currently implemented:

inline This is the default. Conflicts in input and output are recorded with conflict markers, while conflicts on metadata are stored in the appropriate metadata (actual values are kept as their base values).

This gives you a valid notebook that you can open in your usual notebook editor and resolve conflicts by hand, just like you might for a regular source file in your text editor.

use-base When a conflict is encountered, use the value from the base notebook.

use-local When a conflict is encountered, use the value from the local notebook.

use-remote When a conflict is encountered, use the value from the remote notebook.

union When a conflict is encountered, include both the local and the remote value, in that order (local then remote). Conflicts on non-sequence types (anything not list or string) are left unresolved.

Note: The union strategy might resolve to nonsensical values, while still marking conflicts as resolved, so use this carefully.

The `--input-strategy` and `--output-strategy` options lets you specify a strategy to use for conflicts on inputs and outputs, respectively. They accept the same values as the `--merge-strategy` option. If these are set, they will take precedence over `--merge-strategy` for inputs and/or outputs. `--output-strategy` takes two additional options: `remove` and `clear-all`:

remove When a conflict is encountered on a single output, remove that output.

clear-all When a conflict is encountered on any output in a given code cell, clear all outputs for that cell.

To use `nbmerge`, pass it three notebooks:

- `base`: the base, common parent notebook
- `local`: your local changes to base
- `remote`: other changes to base that you want to merge with yours

For example:

```
nbmerge base.ipynb local.ipynb remote.ipynb > merged.ipynb
$ nbmerge v1.ipynb v2.ipynb v3.ipynb -o merged.ipynb
[W autoresolve:162] autoresolving conflict at /cells/0/outputs with inline-outputs
[W autoresolve:162] autoresolving conflict at /cells/0/execution_count with clear
[W autoresolve:162] autoresolving conflict at /cells/1/execution_count with clear
[W nbmergeapp:47] Conflicts occurred during merge operation.
[I nbmergeapp:60] Merge result written to merged.ipynb
```

nbmerge-web

nbmerge-web is just like **nbmerge** above, but instead of automatically resolving or failing on conflicts, a webapp for manually resolving conflicts is displayed:

```
nbmerge-web base.ipynb local.ipynb remote.ipynb -o merged.ipynb
```

Loading Matplotlib demos with %load

Cell deleted locally

Delete cell

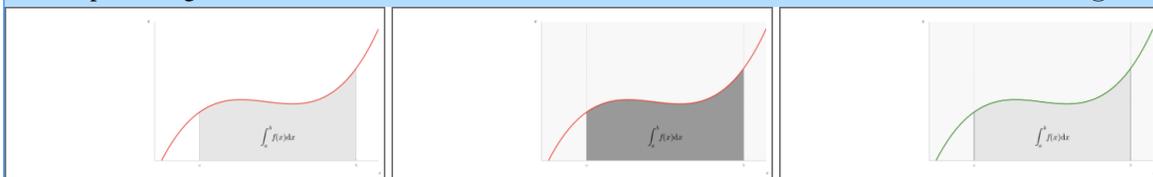
1 IPython's `%load` magic can be used to load any Matplotlib demo by

<pre> 26 (...) 27 fig, ax = plt.subplots 28 plt.plot(x, y, 'r', li 29 plt.ylim(ymin=0) 30 31 # Make the shaded regi 32 ix = np.linspace(a, b) 33 iy = func(ix) 34 verts = [(a, 0)] + lis 35 poly = Polygon(verts, : 36 ax.add_patch(poly) 37 38 (...) 39 40 (...) 41 plt.figtext(0.9, 0.05, 42 plt.figtext(0.1, 0.9, 43 44 ax.spines['right'].set_ 45 ax.spines['top'].set_v 46 (...) 47 48 (...) 49 50 (...) 51 plt.show() 52 53 </pre>	<pre> 26 (...) 27 fig, ax = plt.subplots 28 plt.plot(x, y, 'r', li 29 plt.ylim(ymin=0) 30 31 # Make the shaded regi 32 ix = np.linspace(a, b) 33 iy = func(ix) 34 verts = [(a, 0)] + lis 35 poly = Polygon(verts, : 36 ax.add_patch(poly) 37 38 (...) 39 40 (...) 41 plt.figtext(0.9, 0.05, 42 plt.figtext(0.1, 0.9, 43 44 ax.spines['right'].set_ 45 ax.spines['top'].set_v 46 (...) 47 48 (...) 49 50 (...) 51 plt.show() 52 53 </pre>	<pre> 26 (...) 27 fig, ax = plt.subplots 28 plt.plot(x, y, 'g', li 29 plt.ylim(ymin=0) 30 31 # Make the shaded regi 32 ix = np.linspace(a, b) 33 iy = func(ix) 34 verts = [(a, 0)] + lis 35 poly = Polygon(verts, : 36 ax.add_patch(poly) 37 38 (...) 39 40 (...) 41 plt.figtext(0.9, 0.05, 42 plt.figtext(0.1, 0.9, 43 44 ax.spines['right'].set_ 45 ax.spines['top'].set_v 46 (...) 47 48 (...) 49 50 (...) 51 plt.show() 52 53 </pre>
--	--	--

```

26 (...)
27 fig, ax = plt.subplots()
28 plt.plot(x, y, 'g', linewidth=2)
29 plt.ylim(ymin=0)
30
31 # Make the shaded region
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.6', edgecolor='0.5')
36 ax.add_patch(poly)
37
38 (...)
39
40 (...)
41 plt.figtext(0.9, 0.05, '$x$')
42 plt.figtext(0.1, 0.9, '$y$')
43
44 ax.spines['right'].set_visible(False)
45 ax.spines['top'].set_visible(False)
46 (...)
47
48 (...)
49
50 (...)
51 plt.show()
52
53
                    
```

Outputs changed



3.3 Version control integration

Note: Currently only integration with git is supported out of the box.

Integration with other version control software should be possible if the version control software allows for external drivers and/or tools. For integration, follow the same patterns as outlined in the manual registration sections.

3.3.1 Git integration

Git integration of nbdime is supported in two ways:

- through **drivers** for diff and merge operations, where nbdime takes on the responsibility for performing the diff/merge:
 - *Diff driver*
 - *Merge driver*
- through defining nbdime as diff and merge **tools**, which allow nbdime to display the diff/merge to the user without having to actually depend on git:
 - *Diff web tool*
 - *Merge web tool*

Configure git integration by editing the `.gitconfig` (or `.git/config`) and `.gitattributes` in each git repository or in the home directory for global effect. Read on for commands that edit these files and execute nbdime through git.

Diff driver

Registering an external diff driver with git tells git to call that application to calculate and display diffs to the user. The driver will be called for commands such as `git diff`, but will not be used for all git commands (e.g. `git add --patch` will not use the driver). Consult the git documentation for further details.

Registration can be done in two ways – at the command line or manually.

Command line registration

nbdime supplies an entry point for registering its driver with git:

```
git-nbdiffdriver config --enable [--global]
```

This command will register the nbdime diff driver with git on the project (repository) or global (user) level when the `--global` option is used. Additionally, this command will associate the diff driver with the `.ipynb` file extension, again either on the project or global level.

Manual registration

Alternatively, the diff driver can be registered manually with the following steps:

- To register the driver with git under the name "jupyternotebook", add the following entries to the appropriate `.gitconfig` file:

```
[diff "jupyternotebook"]  
command = git-nbdiffdriver diff
```

- To associate the diff driver with a file type, add the following entry to the appropriate `.gitattributes` file:

```
*.ipynb diff=jupyternotebook
```

Merge driver

Registering an external merge driver with git tells git to call that driver application to calculate merges of certain files. This allows nbtime to become responsible for merging all notebooks.

Registration can be done in two ways – at the command line or manually.

Command line registration

nbtime supplies an entry point for registering its merge driver with git:

```
git-nbmergedriver config --enable [--global]
```

This command will register the nbtime merge driver with git on the project or global level. Additionally, the command will associate the merge driver with the `.ipynb` file extension, again either on the project or global level.

Manual registration

Alternatively, the diff driver can be registered manually with the following steps:

- To register the driver with git under the name “jupyternotebook”, add the following entries to the appropriate `.gitconfig` file:

```
[merge "jupyternotebook"]  
command = git-nbmergedriver merge %O %A %B %L %P
```

- To associate the diff driver with a file type, add the following entry to the appropriate `.gitattributes` file:

```
*.ipynb diff=jupyternotebook
```

Diff web tool

The rich, web-based diff view can be installed as a git *diff tool*. This enables the diff viewer to display diffs of repository history instead of just files.

Command line registration

To register nbtime as a git diff tool, run the command:

```
git-nbdiffstool config --enable [--global]
```

Once registered, the diff tool can be started by running the git command:

```
git diffstool --tool=nbtime [<commit> [<commit>]] [--] [<path>...]
```

If you want to avoid specifying the tool each time, nbdime can be set as the default tool by adding the `--set-default` flag to the registration command:

```
git-nbdifftool config --enable [--global] --set-default
```

This command will set the CLI's diff tool as the default diff tool, and the web based diff tool as the default GUI diff tool. To launch the web view with this configuration, run the git command as follows:

```
git difftool -g [<commit> [<commit>]] [--] [<path>...]
```

Note: Git does not allow selection of different tools per file type. If you set nbdime as the default tool it will be called for **all** changed files. This includes non-notebook files, which nbdime will fail to process.

Manual registration

Alternatively, the diff tool can be registered manually with the following steps:

- To register both the CLI and web diff tools with git under the names “nbdime” and “nbdime”, add the following entries to the appropriate `.gitconfig` file:

```
[difftool "nbdime"]
cmd = git-nbdifftool diff "$LOCAL" "$REMOTE"

[difftool "nbdime"]
cmd = git-nbdifftool "$LOCAL" "$REMOTE"
```

- To set the diff tools as the default tools, add or modify the following entries in the appropriate “.gitconfig” file:

```
[diff]
tool = nbdime
guitool = nbdime
```

Merge web tool

The rich, web-based merge view can be installed as a git *merge tool*. This enables nbdime to process merge conflicts during merging in git.

Command line registration

To register nbdime as a git merge tool, run the command:

```
git-nbmergetool config --enable [--global]
```

Once registered, the merge tool can be started by running the git command:

```
git mergetool --tool=nbdime [<file>...]
```

If you want to avoid specifying the tool each time, nbdime can be set as the default tool by adding the `--set-default` flag to the registration command:

```
git-nbmergetool config --enable --set-default [--global]
```

This will allow the merge tool to be launched simply by:

```
git mergetool [<file>...]
```

Note: Git does not allow to select different tools per file type, so if you set nbtime as the default tool it will be called for *all merge conflicts*. This includes non-notebooks, which nbtime will fail to process. For most repositories, it will therefore not make sense to have nbtime as the default, but rather to call it selectively

Manual registration

Alternatively, the merge tool can be registered manually with the following steps:

- To register both the merge tool with git under the name “nbtime”, add the following entry to the appropriate `.gitconfig` file:

```
[mergetool "nbtime"]
cmd = git-nbmergetool "$BASE" "$LOCAL" "$REMOTE" "$MERGED"
```

- To set nbtime as the default merge tool, add or modify the following entry in the appropriate `.gitconfig` file:

```
[merge]
tool = nbtime
```

3.4 Testing

See the latest automated build, test, and coverage status at:

- [Build and test on Travis](#)
- [Coverage on codecov](#)

3.4.1 Dependencies

Install the test dependencies:

```
pip install "nbtime[test]"
```

3.4.2 Running tests locally

To run python tests, locally, enter:

```
pytest
```

from the project root. If you have Python 2 and Python 3 installed, you may need to enter:

```
python3 -m pytest
```

to run the tests with Python 3. See the [pytest documentation](#) for more options.

To run javascript/typescript tests, enter:

```
npm test
```

from the `nbtime-web` folder.

3.4.3 Submitting test cases

If you have notebooks with interesting merge challenges, please consider [contributing them](#) to nbdime as test cases!

3.5 Glossary

diff object A diff object represents the difference $B-A$ between two objects, A and B , as a list of operations (ops) to apply to A to obtain B .

merge decision An object describing a part of the merge operation between two objects with a common base. Contains both the information about local and remote changes, and the decision taken to resolve the merge.

JSONPatch JSON Patch defines a JSON document structure for expressing a sequence of operations to apply to a JavaScript Object Notation (JSON) document; it is suitable for use with the `HTTP PATCH` method. See [RFC 6902 JavaScript Object Notation \(JSON\) Patch](#).

3.6 Use cases

Use cases for nbdime are envisioned to be mainly in the categories of a merge command for version control integration and diff command for inspecting changes and automated regression testing. At the core of nbdime is the diff algorithms, which must handle not only text in source cells but also a number of data formats based on mime types in output cells.

3.6.1 Basic diffing use cases

While developing basic correct diffing is fairly straightforward, there are still some issues to discuss.

Other tasks (issues will be created for these):

- Plugin framework for mime type specific diffing.
- Diffing of common output types (png, svg, etc.)
- Improve fundamental sequence diff algorithm. Current algorithm is based on a brute force $O(N^2)$ longest common subsequence (LCS) algorithm. This will be rewritten in terms of a faster algorithm such as Myers $O(ND)$ LCS based diff algorithm, optionally using Python's `difflib` for some use cases where it makes sense.

3.6.2 Version control use cases

Most commonly, cell source is the primary content, and output can presumably be regenerated. Indeed, it is not possible to guarantee that merged sources and merged output is consistent or makes any kind of sense.

Some tasks:

- Merge of output cell content is not planned.
- Is it important to track source lines moving between cells?

Loading Matplotlib demos with %load

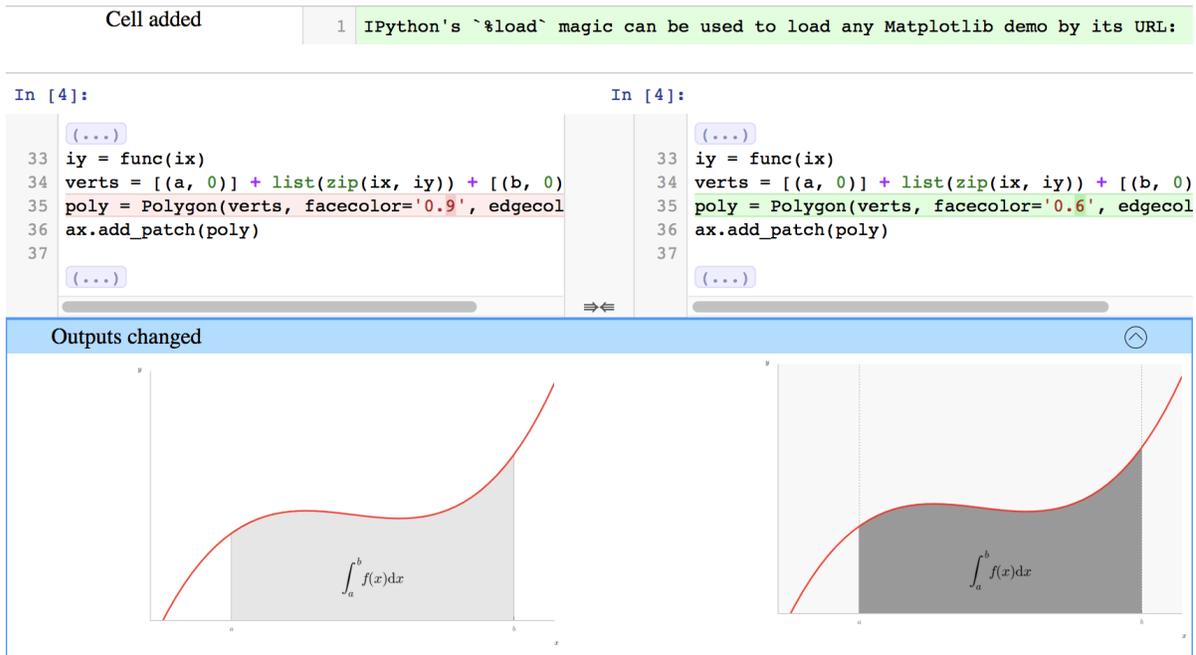


Fig. 3.1: Figure: nbtime's content-aware diff

3.6.3 Regression testing use cases

3.7 diff format

3.7.1 Basics

A *diff object* represents the difference B-A between two objects, A and B, as a list of operations (ops) to apply to A to obtain B. Each operation is represented as a dict with at least two items:

```
{ "op": <opname>, "key": <key> }
```

The objects A and B are either mappings (dicts) or sequences (lists or strings). A different set of ops are legal for mappings and sequences. Depending on the op, the operation dict usually contains an additional argument, as documented below.

The diff objects in nbtime are:

- json-compatible nested structures of dicts (with string keys) and
- lists of values with heterogeneous datatypes (strings, ints, floats).

The difference between these input objects is represented by a json-compatible results object. A JSON schema for validating diff entries is available in `diff_format.schema.json`.

3.7.2 Diff format for mappings

For mappings, the key is always a **string**.

Valid operations (ops) are:

- **remove** - delete existing value at key:

```
{ "op": "remove", "key": <string> }
```

- **add** - insert new value at key not previously existing:

```
{ "op": "add", "key": <string>, "value": <value> }
```

- **replace** - replace existing value at key with new value:

```
{ "op": "replace", "key": <string>, "value": <value> }
```

- **patch** - patch existing value at key with another diffobject:

```
{ "op": "patch", "key": <string>, "diff": <diffobject> }
```

3.7.3 Diff format for sequences

For sequences (list and string) the key is always an **integer index**. This index is relative to object A of length N.

Valid operations (ops) are:

- **removerange** - delete the values A[key:key+length]:

```
{ "op": "removerange", "key": <string>, "length": <n> }
```

- **addrange** - insert new items from valuelist before A[key], at end if key=len(A):

```
{ "op": "addrange", "key": <string>, "valuelist": <values> }
```

- **patch** - patch existing value at key with another diffobject:

```
{ "op": "patch", "key": <string>, "diff": <diffobject> }
```

3.7.4 Relation to JSONPatch

The above described diff representation format has similarities with the *JSONPatch* standard but is also different in a few ways:

operations

- JSONPatch contains operations `move`, `copy`, `test` not used by nbtime.
- nbtime contains operations `addrange`, `removerange`, and `patch` not in JSONPatch.

patch

- JSONPatch uses a deep JSON pointer based `path` item in each operation instead of providing a recursive `patch op`.
- nbtime uses a `key` item in its `patch op`.

diff object

- JSONPatch can represent the diff object as a *single list*.
- nbtime uses a *tree of lists*.

To convert a nbtime diff object to the JSONPatch format, use the `to_json_patch` function:

```
from nbtime.diff_format import to_json_patch
jp = to_json_patch(diff_obj)
```

Note: This function `to_json_patch` is currently a draft, subject to change, and not yet covered by tests.

3.7.5 Examples

For examples of diffs using nbtime, see [test_patch.py](#).

3.8 Merge details

Loading Matplotlib demos with %load

Cell deleted locally
 Delete cell

```

1 IPython's `%load` magic can be used to load any Matplotlib demo by
    
```

(...)
 Delete cell

```

26
27 fig, ax = plt.subplots
28 plt.plot(x, y, 'r', li
29 plt.ylim(ymin=0)
30
31 # Make the shaded regi
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + lis
35 poly = Polygon(verts,
36 ax.add_patch(poly)
37
38
39
40
41 plt.figtext(0.9, 0.05,
42 plt.figtext(0.1, 0.9,
43
44 ax.spines['right'].set
45 ax.spines['top'].set_v
46
47
48
49
50
51 plt.show()
52
53
                    
```

(...)
 Delete cell

```

26
27 fig, ax = plt.subplots
28 plt.plot(x, y, 'r', li
29 plt.ylim(ymin=0)
30
31 # Make the shaded regi
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + lis
35 poly = Polygon(verts,
36 ax.add_patch(poly)
37
38
39
40
41 plt.figtext(0.9, 0.05,
42 plt.figtext(0.1, 0.9,
43
44 ax.spines['right'].set
45 ax.spines['top'].set_v
46
47
48
49
50
51 plt.show()
52
53
                    
```

(...)
 Delete cell

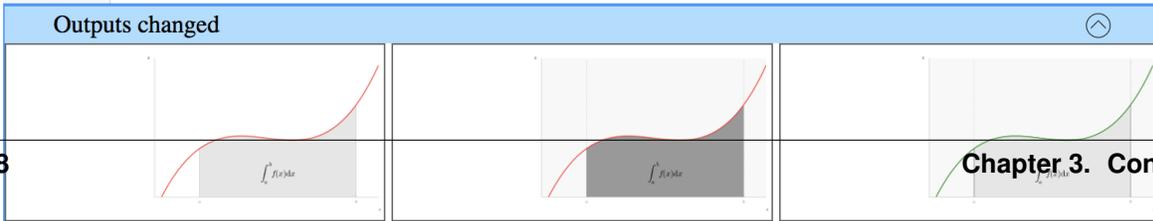
```

26
27 fig, ax = plt.subplots
28 plt.plot(x, y, 'g', li
29 plt.ylim(ymin=0)
30
31 # Make the shaded regi
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + lis
35 poly = Polygon(verts,
36 ax.add_patch(poly)
37
38
39
40
41 plt.figtext(0.9, 0.05,
42 plt.figtext(0.1, 0.9,
43
44 ax.spines['right'].set
45 ax.spines['top'].set_v
46
47
48
49
50
51 plt.show()
52
53
                    
```

(...)
 Delete cell

```

26
27 fig, ax = plt.subplots()
28 plt.plot(x, y, 'g', linewidth=2)
29 plt.ylim(ymin=0)
30
31 # Make the shaded region
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
35 poly = Polygon(verts, facecolor='0.6', edgecolor='0.5')
36 ax.add_patch(poly)
37
38
39
40
41 plt.figtext(0.9, 0.05, '$x$')
42 plt.figtext(0.1, 0.9, '$y$')
43
44 ax.spines['right'].set_visible(False)
45 ax.spines['top'].set_visible(False)
46
47
48
49
50
51 plt.show()
52
53
                    
```



nbdime implements a three-way merge of Jupyter notebooks and a subset of generic JSON objects.

3.8.1 Merge Results

A merge operation with a shared origin object `base` and modified objects, `local` and `remote`, outputs these merge results:

- a fully or partially merged object
- a set of *merge decision* objects that describe the merge operation

3.8.2 Merge decision format

Each three-way notebook merge is based on the differences between the `base` version and the two changed versions – `local` and `remote`. These differences, “base“ with `local` and `base` with `remote`, are then compared, and for each change a set of decisions are made. A *merge decision* object represents such a decision, and is represented as a dict with the following entries:

```
{
  "local_diff": <diff object>,
  "remote_diff": <diff object>,
  "conflict": <boolean>,
  "action": <action taken/suggested>,
  "common_path": <JSON path>,
  "custom_diff": <diff object>
}
```

Merge conflicts

Merge conflicts are indicated with the `conflict` field on the decision object, and if true, indicates that the given differences could not be automatically reconciled.

Note: Even when conflicted, the `action` field might indicate a suggested or “best guess” resolution of the decision. If no such suggestion can be inferred, the base value will be used as the default resolution.

Merge actions

Each *merge decision* has an entry `action` which describes the resolution of the merge. It can take the following values:

- **local:** Use the `local` changes, as described by `local_diff`.
- **remote:** Use the `remote` changes as described by `remote_diff`.
- **base:** Use the original value, that is, do not apply any changes.
- **either:** Indicates that the `local` and `remote` changes are interchangeable, and that either can be used.
- **local_then_remote** - First apply the `local` changes, then the `remote` changes. This is only applicable for certain subset of merges, like insertions in the same location (for example two cells added in the same location).
- **remote_then_local** - Similar to **local_then_remote**, but `remote` changes are taken before `local` ones.
- **clear** - Remove the value(s) on the object. Can, for example, be used to clear the outputs of a cell.

- **custom** - Use the changes as described by `custom_diff`. This can be used for more complex resolutions than those described by the other actions above. A simple example would be for the case of multiple cells (or alternatively, multiple lines of text) inserted both locally and remotely in the same location. Here, the correct resolution might be to take the first element from `local`, then the `remote` changes, and finally the rest of the `local` changes.

Common path

The `common_path` entry of a merge decision describes the path in which the local and remote changes diverge. For example if the local changes are specified as:

```
patch "cells"
  patch index 0
    patch "source"
      addrange <some lines of source to add>
    patch "outputs"
      addrange <a new output added>
```

and the remote changes are specified as:

```
patch "cells"
  patch index 0
    patch "outputs"
      removerange <all outputs removed>
```

then the common path will be `["cells", 0]`, and the *diff object* will omit the patch "cells" and patch 0 operations.

3.9 REST API draft for nbtime server v0.1

The following is a **draft** of the REST API for nbtime. It is not yet frozen but is guided on preliminary work and likely close to the final result. It is also not implemented in this form yet.

The Python package, commandline, and web API should cover the same functionality using the same names but different methods of passing input/output data. Thus consider the request to be the input arguments and response to be the output arguments for all APIs.

3.9.1 Definitions

`json_*` always a JSON object
`json_notebook` a full Jupyter notebook
`json_diff_args` arguments to control `nbdiff` behaviour
`json_merge_args` arguments to control `nbmerge` behaviour
`json_diff_object` diff result in nbtime diff format
`**json_merge_object` merge result in nbtime merge format

3.9.2 /diff

Compute diff of two notebooks provided in full JSON format.

Request:

```
{
  "base": json_notebook,
  "remote": json_notebook,
  "args": json_diff_args
}
```

Response:

```
{
  "diff": json_diff_object
}
```

3.9.3 /merge

Compute merge of three notebooks provided in full JSON format.

Request:

```
{
  "base": json_notebook,
  "local": json_notebook,
  "remote": json_notebook,
  "args": json_merge_args
}
```

Response:

```
{
  "merged": json_notebook,
  "localconflicts": json_diff_object,
  "remoteconflicts": json_diff_object,
}
```

3.9.4 /localdiff

Compute diff of notebooks known to the server by name.

Request:

```
{
  "base": "filename.ipynb",
  "remote": "filename.ipynb",
  "args": json_diff_args
}
```

Response:

```
{
  "base": json_notebook,
  "diff": json_diff_object
}
```

3.9.5 /localmerge

Compute merge of notebooks known to the server by name.

Request:

```
{
  "base": "filename.ipynb",
  "local": "filename.ipynb",
  "remote": "filename.ipynb",
  "args": json_merge_args
}
```

Response:

```
{
  "merged": json_notebook,
  "localconflicts": json_diff_object,
  "remoteconflicts": json_diff_object,
}
```

D

diff object, [23](#)

J

JSONPatch, [23](#)

M

merge decision, [23](#)