nbdime Documentation

Release 0.4.0.dev

Martin Sandve Alnæs and Project Jupyter

Aug 18, 2017

Installation and usage

Version: 0.4.0.dev

nbdime provides tools for diffing and merging Jupyter notebooks.

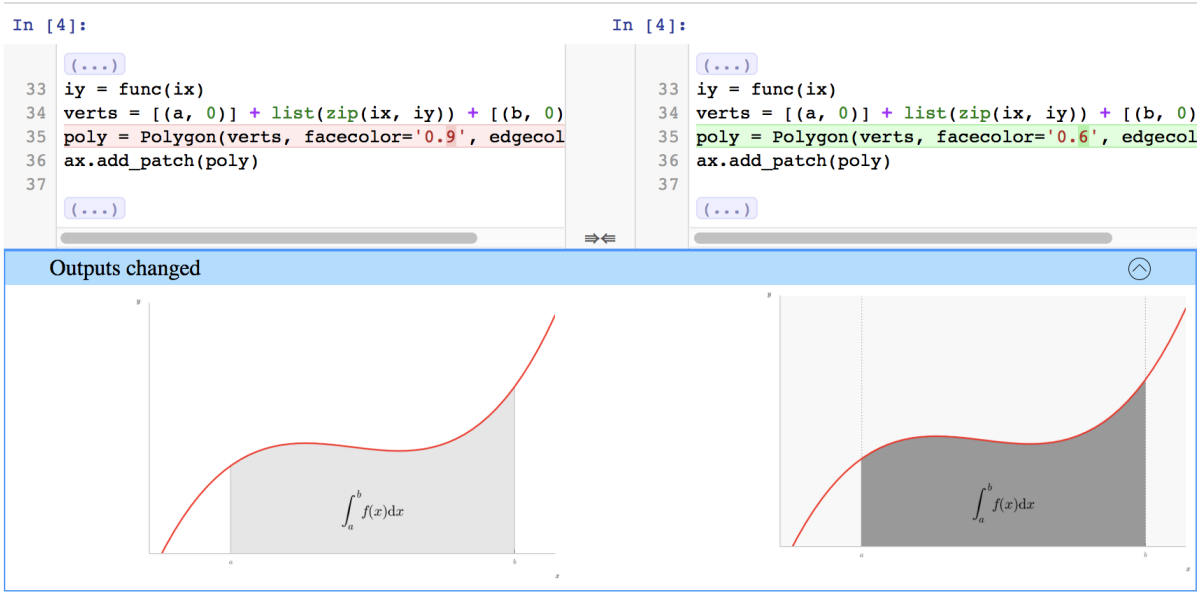


Fig. 1: Figure: nbdime example

CHAPTER 1

Why is nbdime needed?

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:

```
$ diff a.ipynb b.ipynb
76,77d75
<     "plt.rc('axes', grid=False)\n",
<     "plt.rc('axes', facecolor='white')\n",
90c88
<       "image/png": "iVBORw0KGgoAAAANSUhEUgAABLkAAAMQCAYAAADLj7dlAAAABHNCSVQICAgIfAhki
AAAAAlwSFlz\nAAAWJQAAFiUBSVIk8AAAIABJREFUeJzsvXeYZFd57b12h0maPNJII2lGOaCAkEBCFgozIxkBAp
lY\n1waDyDZg8MX+zMU2F4Mx1x8PwWAwxmBjg4yNi2BfQMa20iiAQFkIjXKWRtJIE3tSz3TXuX+8vV2n\nqyucv
N+9z/o9zzynprvq1D6nqqtqr1prbRNFEQghhBBCCCGEEEII8Zkh1wMghBBCCCGEEEIIISQv\nFLkIIYQQQgghhB
BCiPdQ5CKEEEIIIYQQQggh3kORixBCCCGEEEIIIYR4D0UuQgghhBBCCCGEEOI9\nFLkIIYQQQgghhBBCiPdQ5CK
EEEIIIYQQQggh3kORixBCCCGEEEIIIYR4D0UuQgghhBBCCCGEEOI9\nFLkIIYQQQgghhBBCiPdQ5CKEEEIIIYQQ
Qggh3kORixBCCCGEEEIIIYR4D0UuQgghhBBCCCGEEOI9\nFLkIIYQQQjzEGHOJMaZljPmo67EkZWq8D7keByGEE
ELChCIXIYQQQirDGPOmKaFj3BhzkMNx/H/G\nmG3GmP/pagwFEbkeQJUYY75gjNlijHmD67EQQgghRB8UuQghhB
BSJe+DCDMjAH7L4TjeAmA+gLc5\nHEMRGNcDqJi3AVgI4DddQ4QQgh+qDIRQghhJBKMMacCuBMAFsg4sy7jTH
DjobzZwBuBvBxR/dP\nsvERADcC+LTrgRBCCCFEHxS5CCGEEFIVH4C4uP4SIlQcBOD1LgYSRVEziqIXR1H0fRf3
T7IRRdFf\nRlH0K1EUXe96LIQQQgjRB0UuQgghhJSOMWYpgP8BoAXg7wH8HcN9Tsux0UIIYQQQsKBIhchhBBC\
```

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

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:

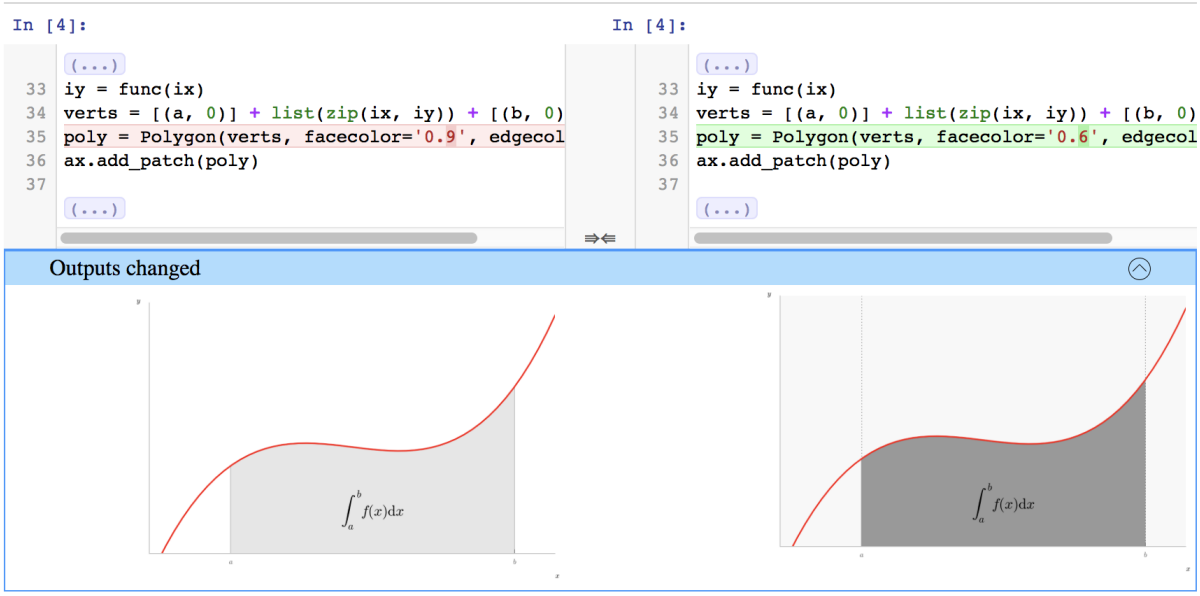


Fig. 1.2: Figure: nbdime's content-aware diff

CHAPTER 2

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 *Console commands*.

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.

The quickest way to get set up for git integration is to call:

```
nbdime config-git --enable --global
```

New in version 0.3: `nbdime config-git`. Prior to 0.3, each nbdime entrypoint had to enable git integration separately.

This will enable the both the drivers and the tools for both diff and merge.

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

To use the web-based GUI viewers of notebook diffs, call:

```
nbdime-web [ref [ref]]
```

```
$ 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

New in version 0.3: support for passing git refs to nbdime commands

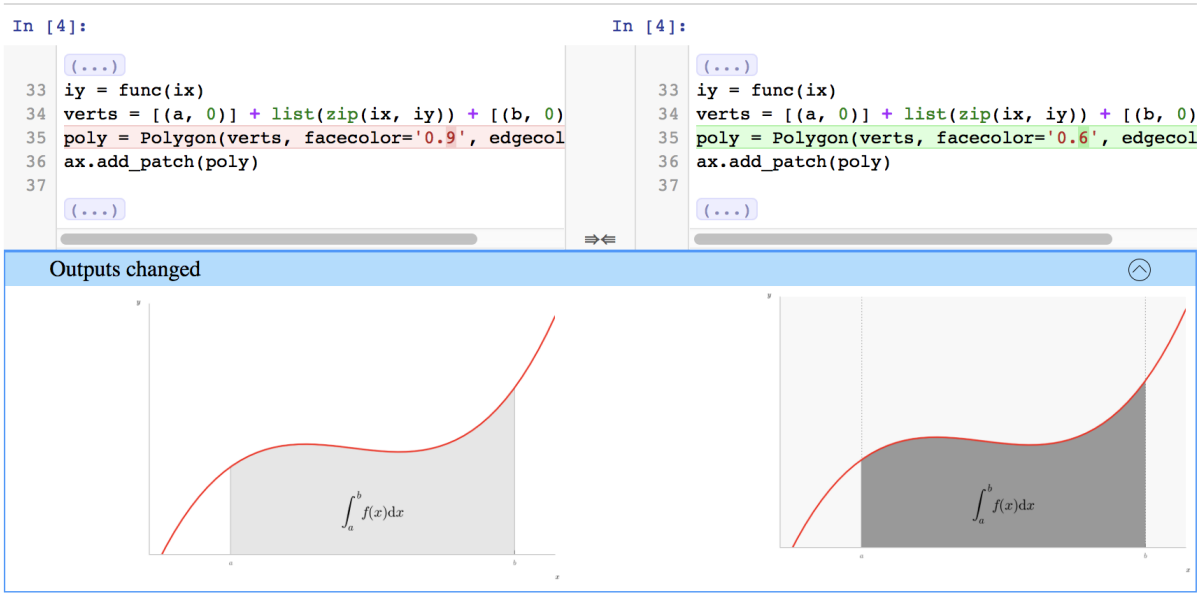


Fig. 2.2: Figure: nbdime's content-aware diff

If you have a merge conflict in a notebook, the merge driver will ensure that the conflicted notebook is a valid notebook that can be viewed in the normal notebook viewer. In it, the conflicts will be marked similarly to how git would normally indicate conflicts, and they can be resolved manually. Alternatively, nbdime provides a web-base mergetool for visualizing and resolving merge conflicts, and it can be launched by calling:

```
nbdime mergetool
```

For more detailed information on integrating nbdime with version control, see *Version control integration*.

Loading Matplotlib demos with %load

Cell deleted locally — Delete cell

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

Delete cell

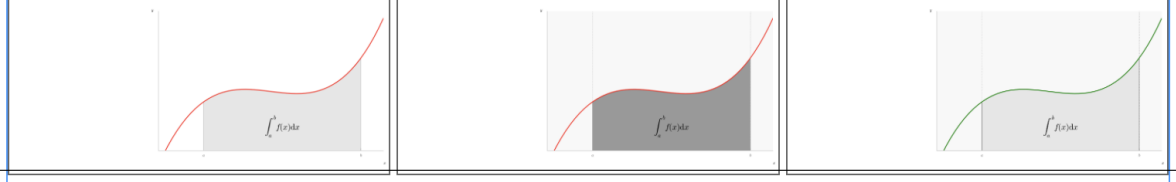
```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts,
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts,
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts,
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'g', 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='0.6', edgecolor='0.5')
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05, '$x$')
plt.figtext(0.1, 0.9, '$y$')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
(...)

plt.show()
```

Outputs changed



CHAPTER 3

Contents

Installation

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

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#egg=nbdime
```

Or clone the nbdime repository and use `pip` to install:

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

Console commands

nbdime provides the following CLI commands:

```
nbshow
nbdiff
nbdiff-web
nbmerge
nbmerge-web
mergetool
config-git
```

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

Additional commands are available for *Git integration*.

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:
```

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

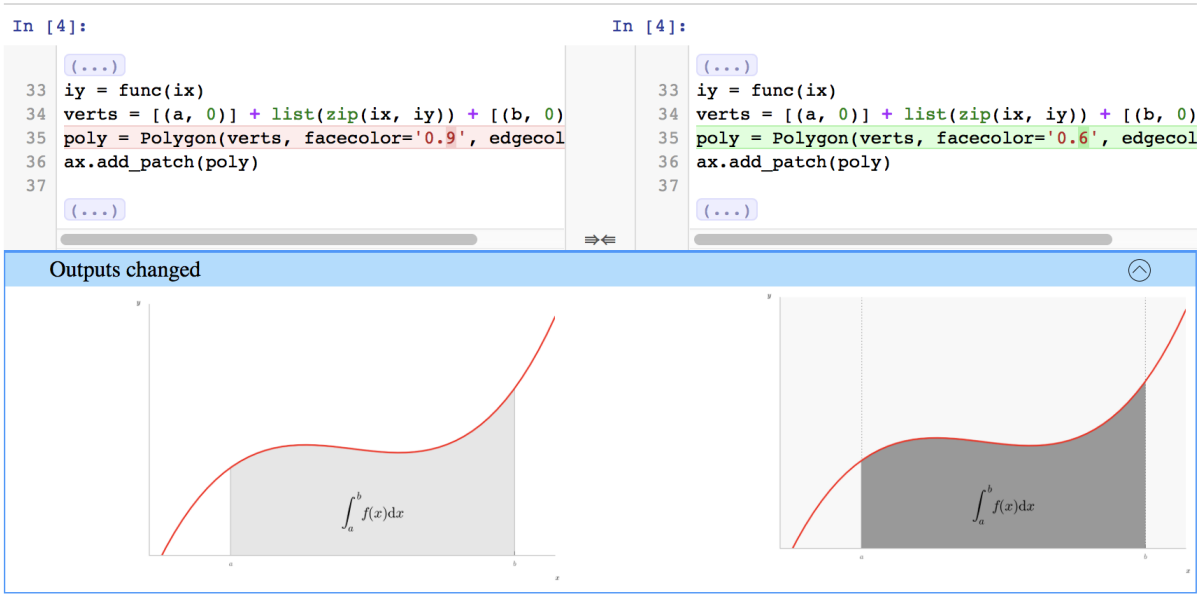
nbdiff does a terminal-optimized rendering of notebook diffs. Pass it the two notebooks you would like to compare, and it returns a nice, readable presentation of the changes in the notebook.

```
$ 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.



Common diff options

You can specify which parts of the notebooks to compare for the diff, by supplying the following flags to any of the diff commands:

- `--sources` / `-s`
- `--outputs` / `-o`
- `--metadata` / `-m`
- `--attachments` / `-a`

These flags can be combined, e.g. *-sm* will only process source and metadata. Alternatively, you can supply some arguments to process everything *except* some parts:

- `--ignore-sources` / `-S`
- `--ignore-outputs` / `-O`
- `--ignore-metadata` / `-M`
- `--ignore-attachments` / `-A`

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.

nbdime 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, nbdime can automatically resolve conflicts on generated fields.

See also:

For more details on how nbdime 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.

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, respecively. 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 occured 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

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

Delete cell

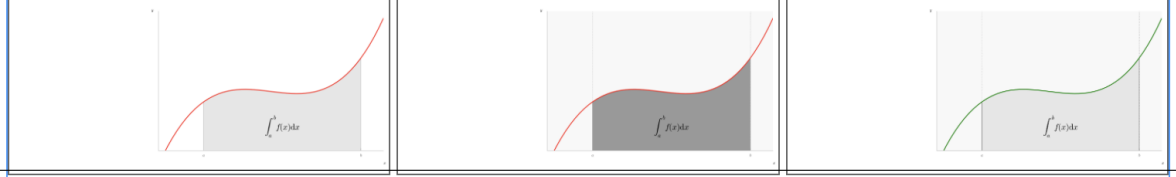
```
(...)
fig, ax = plt.subplots
plt.plot(x, y, 'r', li
plt.ylim(ymin=0)

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts, 
ax.add_patch(poly)

(...)
plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

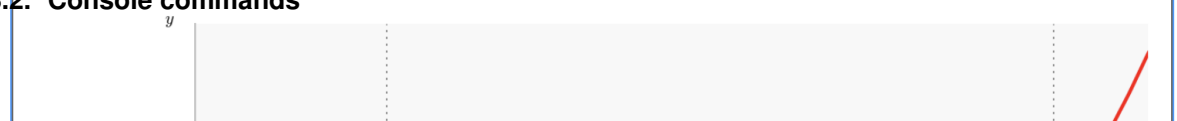
```
(...)
fig, ax = plt.subplots
plt.plot(x, y, 'r', li
plt.ylim(ymin=0)

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts, 
ax.add_patch(poly)

(...)
plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)
fig, ax = plt.subplots
plt.plot(x, y, 'g', li
plt.ylim(ymin=0)

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts, 
ax.add_patch(poly)

(...)
plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'g', 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='0.6', edgecolor='0.5')
ax.add_patch(poly)

(...)
plt.figtext(0.9, 0.05, '$x$')
plt.figtext(0.1, 0.9, '$y$')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
(...)

plt.show()
```

Outputs changed





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.

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/etc directory for global effect. Details for how to individually configure the different drivers/tools are given below.

To configure all diff/merge drivers and tools, simply call:

```
nbdime config-git (--enable | --disable) [--global | --system]
```

This command will register nbdime with git for the current project (repository), or on the global (user), or sytem level according to the `--global` or `--system` options.

New in version 0.3: `nbdime config-git`. Prior to 0.3, each nbdime entrypoint had to enable git integration separately.

Note: When neither the global or system flag is given, the configuration is only applied to the current project (repository). The command will therefore need to be run from within a git repository.

Usage

Once configured, the diff/merge drivers should simply work out of the box. For example, the normal git diff command:

```
git diff [<commit> [<commit>]] [--] [<path>...]
```

should give you the standard diff for any non-notebook files, but use nbdime's command-line diff for all *.ipynb* files. Nbdime will also be used for all merges on notebook files (no specific commands needed). To launch the rich, web-based tools (for diff visualization and merge conflict visualization/resolution), the following commands will need to be executed:

```
nbdiff-web [ref [ref]]
```

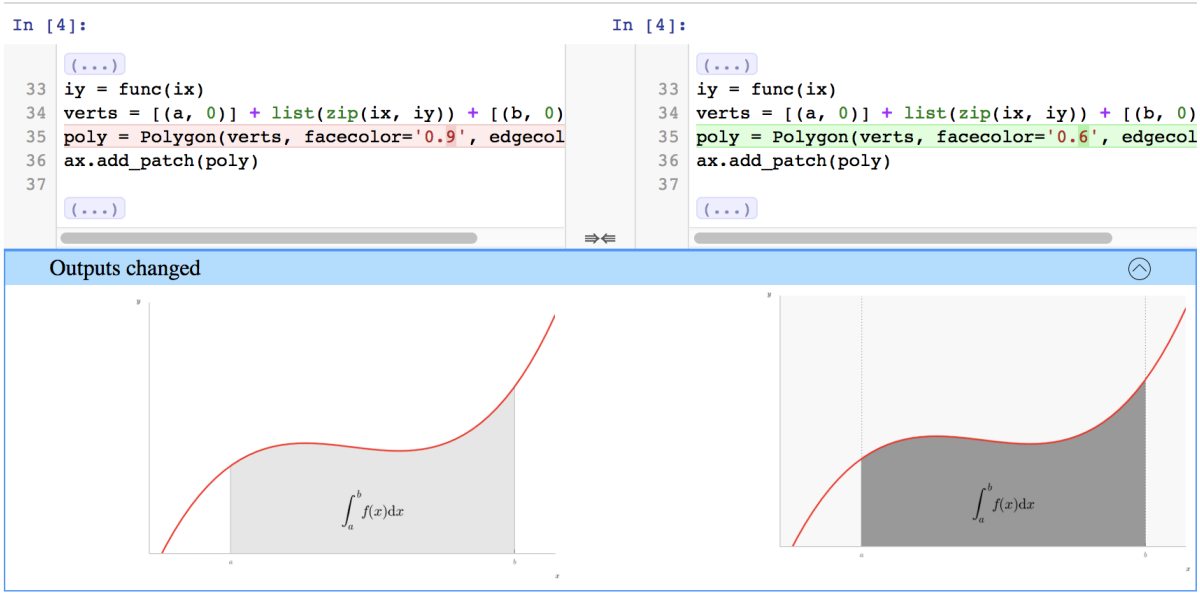


Fig. 3.1: Figure: nbdime's content-aware diff

and:

```
git mergetool --tool nbdime -- *.ipynb
```

Note: Using **`git-nbdiffdriver config`** overrides the ability to call **`git difftool`** 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.

Note: If you simply call *git mergetool –tool nbdime*, it will be called for all merge conflicts, even on filetypes that it cannot handle. To only call on notebooks, add a filter on file paths, e.g. *git mergetool –tool nbdime – *.ipynb*. **This command has also been aliased as 'nbdime mergetool' for easy access**, and you can also add your own git alias for this command.

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:

Loading Matplotlib demos with %load

Cell deleted locally — Delete cell

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

Delete cell

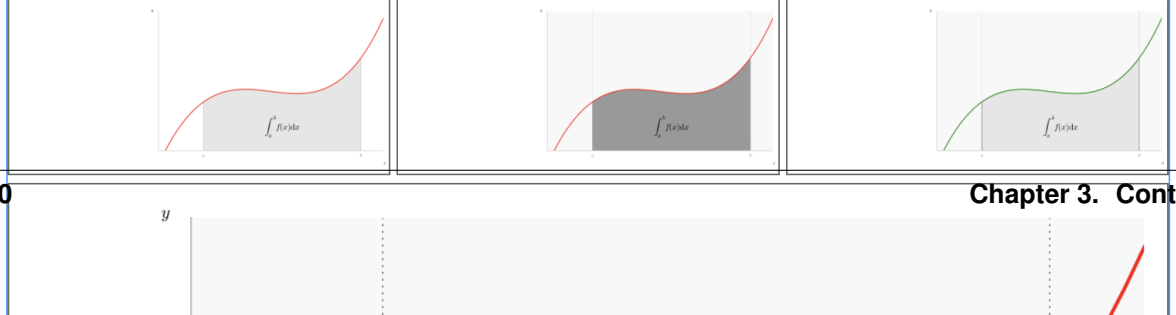
```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts, 
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts, 
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts, 
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'g', 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='0.6', edgecolor='0.5')
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05, '$x$')
plt.figtext(0.1, 0.9, '$y$')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
(...)

plt.show()
```

Outputs changed



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

This command will register the nbdime diff driver with git, and associate the diff driver with the `.ipynb` file extension. The *–global | –system* flags work as explained above.

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 (*git config [–global | –system] -e* to edit):

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

  or if you prefer to use webdiff:: [diff "jupyternotebook"] command = git-nbdiffdriver webdiff [–ip IP]

- 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 nbdime to become responsible for merging all notebooks.

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

Command line registration

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

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

This command will register the nbdime merge driver with git, and associate the merge driver with the `.ipynb` file extension. The *–global | –system* flags work as explained above.

Manual registration

Alternatively, the merge 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 (*git config [–global | –system] -e* to edit):

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

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

```
*.ipynb merge=jupyternotebook
```

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, and present them for resolution.

Command line registration

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

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

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 | --system]
```

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 nbdime as the default tool it will be called for *all merge conflicts*. This includes non-notebooks, which nbdime will fail to process. For most repositories, it will therefore not make sense to have nbdime 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 "nbdime", add the following entry to the appropriate `.gitconfig` file (*git config [–global | –system] -e* to edit):

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

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

  ```
  [merge]
  tool = nbdime
  ```

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.

Changes in nbdime

0.3

- Handle git refs directly in nbdime, so you can `nbdiff HEAD notebook.ipynb`, etc. in git repos. This replaces the never-quite-working git difftool
- Support filtering options on all entrypoints, e.g. `nbdiff -s` to only show diff of sources. See `nbdiff -h` for details
- Fix MathJax CDN URL, now that cdn.mathjax.org has shutdown
- Use `jupyter-packaging` to build javascript sources in `setup.py`
- Various fixes and performance improvements

0.2

- Support ip, base-url arguments to web entrypoints
- Enable export of web diff to static HTML
- Various fixes and improvements

0.1

0.1.2 - 2017-01

- Fix inclusion of webapp sources in wheels

0.1.1 - 2017-01

- Fix default location of `--global` git attributes file
- Support `--system` argument for git configuration, allowing easy setup of nbdime system-wide
- Render tracebacks and colors in stream output in web view
- Render output as long as one MIME-type is safe in web view
- Improve styling of web view
- Fix a bug in inline-merge

0.1.0 - 2016-12

First release of nbdime!

Testing

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

- Build and test on Travis
- Coverage on codecov

Dependencies

Install the test dependencies:

```
pip install "nbdime[test]"
```

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:

```
cd nbdime-web/
npm test
```

Submitting test cases

If you have notebooks with interesting merge challenges, please consider contributing them to nbdime as test cases!

diff format

This sections provide details on how nbdime represents diffs, and will mostly be relevant for those who want to use nbdime as a library, or that want to contribute to nbdime.

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> }
```

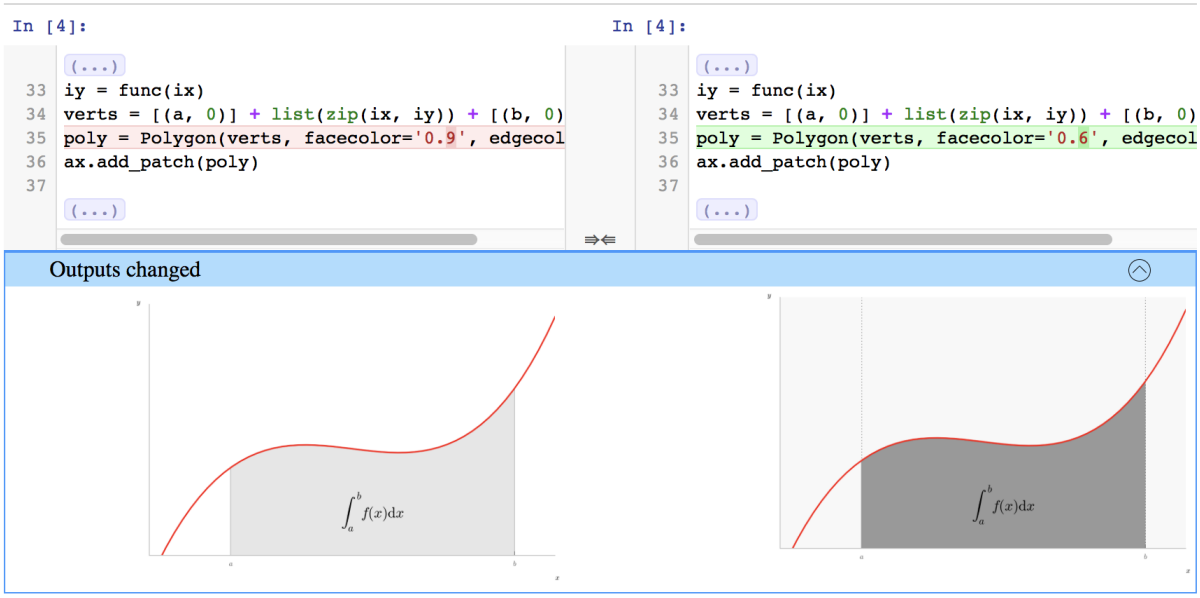


Fig. 3.3: Figure: nbdime's content-aware diff

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 nbdime 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.

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> }
```

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> }
```

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 nbdime.
- nbdime 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.
- nbdime uses a `key` item in its `patch` op.

diff object

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

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

```
from nbdime.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.

Examples

For examples of diffs using nbdime, see test_patch.py.

Merge details

This sections provide details on how nbdime handles merges, and will mostly be relevant for those who want to use nbdime as a library, or that want to contribute to nbdime.

Loading Matplotlib demos with %load

Cell deleted locally — Delete cell

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

Delete cell

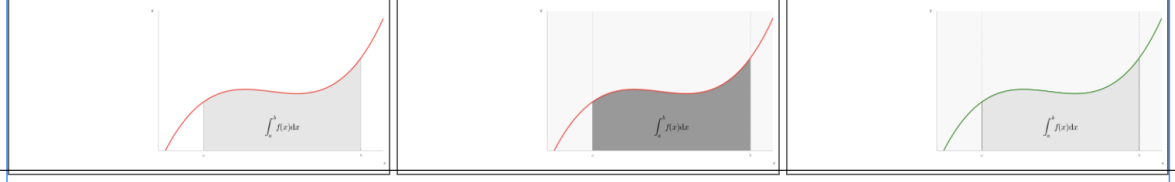
```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts,
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

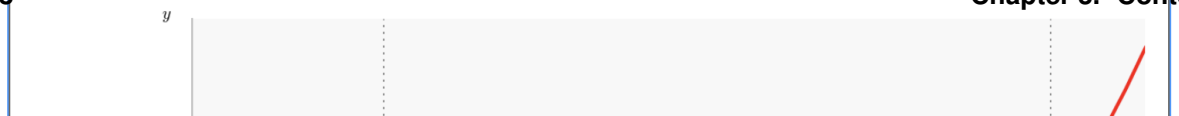
```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts,
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

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

# Make the shaded regi
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + lis
poly = Polygon(verts,
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05,
plt.figtext(0.1, 0.9,

ax.spines['right'].set_
ax.spines['top'].set_v
(...)

plt.show()
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'g', 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='0.6', edgecolor='0.5')
ax.add_patch(poly)

(...)

plt.figtext(0.9, 0.05, '$x$')
plt.figtext(0.1, 0.9, '$y$')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
(...)

plt.show()
```

Outputs changed





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

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

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.

REST API

The following is a preliminary REST API for nbdime. It is not yet frozen but is guided on preliminary work and likely close to the final result.

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.

Definitions

*json_** : always a JSON object

json_notebook : a full Jupyter notebook

json_diff_object : diff result in nbdime diff format as specified in *diff format*

json_merge_decisions : merge decisions as specified in *Merge details*

/api/diff

Compute diff of two notebooks provided as filenames local to the server working directory, and/or as URLs.

Request:

```
{
  "base":   "filename.ipynb" | "http://your-domain/url/path",
  "remote": "filename.ipynb" | "http://your-domain/url/path"
}
```

Response:

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

/api/merge

Compute merge of three notebooks provided as filenames local to the server working directory, and/or as URLs.

Request:

```
{
  "base":   "filename.ipynb" | "http://your-domain/url/path",
  "local":  "filename.ipynb" | "http://your-domain/url/path",
  "remote": "filename.ipynb" | "http://your-domain/url/path"
}
```

Response:

```
{
  "base": json_notebook,
  "merge_decisions": json_merge_decisions
}
```

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.

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.

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?

Regression testing use cases

CHAPTER 4

Acknowledgements

nbdime is developed with financial support from:

- OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541), http://opendreamkit.org .
- The Gordon and Betty Moore Foundation through Grant GBMF #4856,by the Alfred P. Sloan Foundation and by the Helmsley Trust.

Index