ROS Bag Python Controller Documentation

Release 0.1.2

Jean Nassar

January 09, 2017

Contents

Contents:

CHAPTER 1

ROS Bag Python Controller

Programmatically control ROS Bag files with Python. Look at `rosbag_pandas` on PyPI for an excellent package which allows you to work with the data directly.

- Free software: MIT license
- Documentation: https://pyrosbag.readthedocs.io.

1.1 Features

- General Bag class
- `rosbag play`

1.2 To do

- check
- compress
- decompress
- filter
- fix
- help
- info
- record
- reindex

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install ROS Bag Python Controller, run this command in your terminal:

```
$ pip install pyrosbag
```

This is the preferred method to install ROS Bag Python Controller, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for ROS Bag Python Controller can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/masasin/pyrosbag
```

Or download the tarball:

```
$ curl  -OL https://github.com/masasin/pyrosbag/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use ROS Bag Python Controller in a project, just import whatever components you need:

```
import pyrosbag as prb
```

For instance, to forward user input:

```
with prb.BagPlayer("example.bag") as example:
    example.play()
    while example.is_running():
        inputs = input()
        kotaro.send(inputs)
```

Or, to play the bag file intermittently:

```
import time

INTERVAL = 3  # seconds

with BagPlayer("example.bag") as example:
    example.play()
    while example.is_running():
        # Run for INTERVAL seconds.
        time.sleep(INTERVAL)

        # Pause for INTERVAL seconds.
        # While paused, step through at a rate of once a second.
        example.pause()
        for _ in range(INTERVAL - 1):
            time.sleep(1)
            example.step()
        time.sleep(1)

        # Resume playing the bag file.
        example.resume()
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/masasin/pyrosbag/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

ROS Bag Python Controller could always use more documentation, whether as part of the official ROS Bag Python Controller docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/masasin/pyrosbag/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *pyrosbag* for local development.

1. Fork the *pyrosbag* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pyrosbag.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pyrosbag
$ cd pyrosbag/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 pyrosbag tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4 and 3.5, and 3.6. Check https://travis-ci.org/masasin/pyrosbag/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests/test_pyrosbag.py::TestClassName::test_name
```

CHAPTER 5

Credits

5.1 Development Lead

- Jean Nassar <jeannassar5@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.1 (2017-01-09)

- Fix Continuous Integration.
- Fix Code Coverage metrics.

6.2 0.1.0 (2017-01-09)

- First release on PyPI.

CHAPTER 7

Technical Documentation

7.1 pyrosbag package

7.1.1 Module contents

Note that, in order to access the data within the bag file, the `rosbag_python` package is extremely convenient. It is available on PyPI.

7.1.2 pyrosbag.pyrosbag module

Programmatically control a ROS bag file.

This module implements the base class, and the various functions.

Currently implemented are:

- `rosbag play`

class `pyrosbag.pyrosbag.`**`Bag`**(*filenames*)
Bases: `object`

Open and manipulate a bag file programmatically.

Parameters **`filenames`** (*`StringTypes | List[StringTypes]`*) – The location of the bag files.

`filenames`
List[StringTypes]

The location of the bag files.

`process`
subprocess.Popen

The process containing the running bag file.

`__enter__`()
Context manager entry point.

`__exit__`(*exc_type*, *exc_value*, *traceback*)
Context manager exit point.

`is_running`
Check whether the bag file is running.

Returns The bag file is running.

Return type bool

send(*string*)
Write something to process stdin.

Parameters **string** (*str*) – The string to write.

Raises *BagNotRunningError* – If interaction is attempted when the bag file is not running.

stop()
Stop a running bag file.

Raises *BagNotRunningError* – If the bag file is not running.

wait()
Block until process is complete.

Raises *BagNotRunningError* – If the bag file is not running.

exception pyrosbag.pyrosbag.**BagError**
Bases: exceptions.Exception

Catch bag player exceptions.

exception pyrosbag.pyrosbag.**BagNotRunningError**(*action='talk to'*)
Bases: *pyrosbag.pyrosbag.BagError*

Raised when interaction is attempted with a bag file which is not running.

class pyrosbag.pyrosbag.**BagPlayer**(*filenames*)
Bases: *pyrosbag.pyrosbag.Bag*

Play Bag files.

pause()
Pause the bag file.

play(*wait=False*, *stdin=-1*, *stdout=None*, *stderr=None*, *quiet=None*, *immediate=None*, *start_paused=None*, *queue_size=None*, *publish_clock=None*, *clock_publish_freq=None*, *delay=None*, *publish_rate_multiplier=None*, *start_time=None*, *duration=None*, *loop=None*, *keep_alive=None*)
Play the bag file.

Parameters

- **wait** (*Optional[Bool]*) – Wait until completion.
- **stdin** (*Optional[file]*) – The stdin buffer. Default is subprocess.PIPE.
- **stdout** (*Optional[file]*) – The stdout buffer.
- **stderr** (*Optional[file]*) – The stderr buffer.
- **quiet** (*Optional[Bool]*) – Suppress console output.
- **immediate** (*Optional[Bool]*) – Play back all messages without waiting.
- **start_paused** (*Optional[Bool]*) – Start in paused mode.
- **queue_size** (*Optional[int]*) – Set outgoing queue size. Default is 100.
- **publish_clock** (*Optional[Bool]*) – Publish the clock time.
- **clock_publish_freq** (*Optional[float]*) – The frequency, in Hz, at which to publish the clock time. Default is 100.

- **delay** (*Optional[float]*) – The number of seconds to sleep afer every advertise call (e.g., to allow subscribers to connect).
- **publish_rate_multiplier** (*Optional[float]*) – The factor by which to multiply the publish rate.
- **start_time** (*Optional[float]*) – The number of seconds into the bag file at which to start.
- **duration** (*Optional[float]*) – The number of seconds from the start to play.
- **loop** (*Optional[Bool]*) – Loop playback.
- **keep_alive** (*Optional[Bool]*) – Keep alive past end of bag (e.g. for publishing latched topics).

resume()
Resume the bag file.

step()
Step through a paused bag file.

exception pyrosbag.pyrosbag.**MissingBagError**
Bases: *pyrosbag.pyrosbag.BagError*

Bag file was not specified.

msg = 'No Bag files were specified.'

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

p

Symbols

B

F

I

M

P

R

S

W