ExpAn Documentation

Release 0.6.0

Zalando SE

Sep 14, 2017

Contents

1 ExpAn: Experiment Analysis **3**
- 1.1 Installation . . . 3
- 1.2 Usage . . . 3
- 1.3 Documentation . . . 4
- 1.4 License . . . 4

2 Installation **5**
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5

3 Usage **7**
- 3.1 Some mock-up data . . . 7
- 3.2 Per-entity ratio vs. ratio of totals . . . 7

4 Contributing **9**
- 4.1 Style guide . . . 9
- 4.2 Testing . . . 9
- 4.3 Branching / Release . . . 9
- 4.4 Versioning . . . 9
- 4.5 Bumping Version . . . 10
- 4.6 Travis CI and PyPI deployment . . . 10
- 4.7 TODOs . . . 11

5 Related Projects **13**

6 Credits **15**
- 6.1 Development Lead . . . 15
- 6.2 Contributors . . . 15

7 Change Log **17**
- 7.1 v0.5.3 (2017-06-26) . . . 17
- 7.2 v0.5.2 (2017-05-11) . . . 18
- 7.3 v0.5.1 (2017-04-20) . . . 18
- 7.4 v0.5.0 (2017-04-05) . . . 18
- 7.5 v0.4.5 (2017-02-10) . . . 19
- 7.6 v0.4.4 (2017-02-09) . . . 19
- 7.7 v0.4.3 (2017-02-07) . . . 19

7.8 v0.4.2 (2016-12-08) . . . 19
7.9 v0.4.1 (2016-10-18) . . . 19
7.10 v0.4.0 (2016-08-19) . . . 19
7.11 v0.3.4 (2016-08-08) . . . 20
7.12 v0.3.3 (2016-08-02) . . . 20
7.13 v0.3.2 (2016-08-02) . . . 20
7.14 v0.3.1 (2016-07-15) . . . 20
7.15 v0.3.0 (2016-06-23) . . . 20
7.16 v0.2.5 (2016-05-30) . . . 20
7.17 v0.2.4 (2016-05-16) . . . 20
7.18 v0.2.3 (2016-05-06) . . . 21
7.19 v0.2.2 (2016-05-06) . . . 21
7.20 v0.2.1 (2016-05-06) . . . 21
7.21 v0.2.0 (2016-05-06) . . . 21

8 Indices and tables **23**

Contents:

CHAPTER 1

ExpAn: Experiment Analysis

A/B tests (a.k.a. Randomized Controlled Trials or Experiments) have been widely applied in different industries to optimize business processes and user experience. ExpAn (**Exp**eriment **An**alysis) is a Python library developed for the statistical analysis of such experiments and to standardise the data structures used.

The data structures and functionality of ExpAn are generic such that they can be used by both data scientists optimizing a user interface and biologists running wet-lab experiments. The library is also standalone and can be imported and used from within other projects and from the command line.

Major statistical functionalities include:

- **feature check**
- **delta**
- **subgroup analysis**
- **trend**

Installation

To install ExpAn, run this command in your terminal:

```
$ pip install expan
```

Usage

To use ExpAn in a project:

```
import expan
```

Some mock-up data:

```
from expan.core.experiment import Experiment
from expan.core.util import generate_random_data

exp = Experiment('B', *generate_random_data())
exp.delta()
```

Documentation

The latest stable version is 0.6.0.

ExpAn main documentation

ExpAn Description - details about the concept of the library and data structures.

ExpAn Introduction - a full jupyter (iPython) notebook. You can view it as slides with jupyter:

```
sh serve_intro_slides
```

License

CHAPTER 2

Installation

Stable release

To install ExpAn, run this command in your terminal:

```
$ pip install expan
```

This is the preferred method to install ExpAn, 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.

From sources

The sources for ExpAn can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/zalando/expan
```

Or download the tarball:

```
$ curl  -OL https://github.com/zalando/expan/tarball/master
```

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

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

CHAPTER 3

Usage

To use ExpAn in a project:

```
import expan
```

Some mock-up data

```
from expan.core.experiment import Experiment
from expan.core.util import generate_random_data

exp = Experiment('B', *generate_random_data())
exp.delta()
```

Per-entity ratio vs. ratio of totals

There are two different definitions of a ratio metric (think of e.g. conversion rate, which is the ratio between the number of orders and the number of visits): 1) one that is based on the entity level or 2) ratio between the total sums, and **ExpAn** supports both of them.

In a nutshell, one can reweight the individual **per-entity ratio** to calculate the **ratio of totals**. This enables to use the existing `statistics.delta()` function to calculate both ratio statistics (either using normal assumtion or bootstraping).

Calculating the conversion rate

As an example let's look at how to calculate the conversion rate, which might be typically defined per-entity as the average ratio between the number of orders and the number of visits:

$$\overline{CR}^{(pe)} = \frac{1}{n}\sum_{i=1}^{n} CR_i = \frac{1}{n}\sum_{i=1}^{n} \frac{O_i}{V_i}$$

The ratio of totals is a reweighted version of CR_i to reflect not the entities' contributions (e.g. contribution per custormer) but overall equal contributions to the conversion rate, which can be formulated as:

$$CR^{(rt)} = \frac{\sum_{i=1}^{n} O_i}{\sum_{i=1}^{n} V_i}$$

Overall as reweighted Individual

One can calculate the $CR^{(rt)}$ from the $\overline{CR}^{(pe)}$ using the following weighting factor (easily proved by paper and pencile):

$$CR^{(rt)} = \frac{1}{n} \sum_{i=1}^{n} \alpha_i \frac{O_i}{V_i}$$

with

$$\alpha_i = n \frac{V_i}{\sum_{i=1}^{n} V_i}$$

Weighted delta function

To have such functionality as a more generic approach in **ExpAn**, we can introduce a *weighted delta* function. Its input are

- The per-entity metric, e.g. O_i/V_i
- A reference metric, on which the weighting factor is based, e.g. V_i

With this input it calculates α as described above and outputs the result of `statistics.delta()`.

CHAPTER 4

Contributing

Style guide

We follow PEP8 standards with the following exceptions:

- Use *tabs instead of spaces* - this allows all individuals to have visual depth of indentation they prefer, without changing the source code at all, and it is simply smaller

Testing

Easiest way to run tests is by running the command `tox` from the terminal. The default Python environments for testing with are py27 and py34, but you can specify your own by running e.g. `tox -e py35`.

Branching / Release

We currently use the gitflow workflow. Feature branches are created from and merged back to the `dev` branch, and the `master` branch stores snapshots/releases of the `dev` branch.

See also the much simpler github flow here

Versioning

For the sake of reproducibility, always be sure to work with a release when doing the analysis!

We use semantic versioning (http://semver.org), and the current version of ExpAn is: v0.4.0.

The version is maintained in `setup.cfg`, and propagated from there to various files by the `bumpversion` program. The most important propagation destination is in `version.py` where it is held in the string `__version__` with the form:

```
'{major}.{minor}.{patch}'
```

The __version__ string and a version() function is imported by core.__init__ and so is accessible to imported functions in expan.

The version(format_str) function generates version strings of any form. It can use git's commit count and revision number to generate a long version string which may be useful for pip versioning? Examples: NB: caution using this... it won't work if not in the original git repository.

```
>>> import core.binning
>>> core.version()
'v0.4.0'
>>> core.version('{major}.{minor}..{commits}')
'0.0..176'
>>> core.version('{commit}')
'a24730a42a4b5ae01bbdb05f6556dedd453c1767'
```

See: StackExchange 151558

Bumping Version

Can use bumpversion to maintain the __version__ in version.py:

```
$ bumpversion patch
```

or

```
$ bumpversion minor
```

This will update the version number, create a new tag in git, and commit the changes with a standard commit message.

When you have done this, you must push the commit and new tag to the repository with:

```
$ git push --tags
```

Travis CI and PyPI deployment

We use Travis CI for testing builds and deploying our PyPI package.

A **build** and **test** is triggered when a commit is pushed to either

- **dev**,
- **master**
- or a **pull request branch to dev or master**.

If you want to **deploy to PyPI**, then follow these steps:

- assuming you have a dev branch that is up to date, create a pull request from dev to master (a travis job will be started for the pull request)
- once the pull request is approved, merge it (another travis job will be started because a push to master happened)
- checkout master

- push **tags** to **master** (a third travis job will be started, but this time it will also push to PyPI because tags were pushed)

If you wish to skip triggering a CI task (for example when you change documentation), please include `[ci skip]` in your commit message.

The flow would then look like follows:

1. `git fetch`
2. `git checkout dev`
3. `git pull`
4. `bumpversion (patch|minor)`
5. `make docs`
6. `git push --tags`
7. `git push`
8. create pull request from dev to master
9. merge pull request

TODOs

- parallelization, eg. for the bootstrapping code
- Bayesian updating/early stopping
- multiple comparison correction, definitely relevant for delta and SGA, have to think about how to correct for time dependency in the trend analysis
- implement from_json and to_json methods in the Binning class, in order to convert the Python object to a json format for persisting in the Results metadata and reloading from a script

CHAPTER 5

Related Projects

There may be alternative libraries providing similar functionality, and these should be collected here. Very incomplete list so far...

- **abba** (https://github.com/thumbtack/abba)

  Mainly handles binomial distributions.

- **bootstrapped** (https://github.com/facebookincubator/bootstrapped)

  Calculates bootstrapped confidence intervals, with A/B test as an example.

CHAPTER 6

Credits

Development Lead

- Octopus McSquid <team-octopus@zalando.de>

Contributors

- Grigory Bordyugov <grigory.bordyugov@zalando.de>
- Dominic Heger <dominic.heger@zalando.de>
- Jie Bao <jie.bao@zalando.de>
- Marko Kolarek <marko.kolarek@zalando.de>
- Robert Muil <robert.muil@zalando.de>
- Till Riffert <till.riffert@zalando.de>

CHAPTER 7

Change Log

v0.5.3 (2017-06-26)

Full Changelog

Implemented enhancements:

- Weighted KPIs is only implemented in regular delta #114

Fixed bugs:

- Assumption of nan when computing weighted KPIs #119
- Weighted KPIs is only implemented in regular delta #114
- Percentiles value is lost during computing group_sequential_delta #108

Closed issues:

- Failing early stopping unit tests #85

Merged pull requests:

- OCTO-1804: Optimize the loading of .stan model in expan. #126 (daryadedik)
- Test travis python version #125 (shansfolder)
- OCTO-1619 Cleanup ExpAn code #124 (shansfolder)
- OCTO-1748: Make number of iterations as a method argument in _bayes_sampling #123 (daryadedik)
- OCTO-1615 Use Python builtin logging instead of our own debugging.py #122 (shansfolder)
- OCTO-1711 Support weighted KPIs in early stopping #121 (shansfolder)
- Fixed a few bugs #120 (shansfolder)
- OCTO-1614 cleanup module structure #115 (shansfolder)
- OCTO-1677 : fix missing .stan files #113 (gbordyugov)
- Bump version 0.5.1 -> 0.5.2 #112 (mkolarek)

v0.5.2 (2017-05-11)

Full Changelog

Merged pull requests:

- OCTO-1502 support **kwargs for four delta functions #111 (shansfolder)
- new version 0.5.1 #107 (mkolarek)

v0.5.1 (2017-04-20)

Full Changelog

Implemented enhancements:

- Derived KPIs are passed to Experiment.fixed_horizon_delta() but never used in there #96

Merged pull requests:

- OCTO-1501: bugfix in Results.to_json() #105 (gbordyugov)
- OCTO-1502 removed variant_subset parameter... #104 (gbordyugov)
- OCTO-1540 cleanup handling of derived kpis #102 (shansfolder)
- OCTO-1540: cleanup of derived kpi handling in Experiment.delta() and ... #97 (gbordyugov)
- Merge dev to master for v0.5.0 #94 (mkolarek)

v0.5.0 (2017-04-05)

Full Changelog

Implemented enhancements:

- Bad code duplication in experiment.py #81
- pip == 8.1.0 requirement #76

Fixed bugs:

- Experiment.sga() assumes features and KPIs are merged in self.metrics #87
- pctile can be undefined in `Results.to\_json\(\)` #78

Closed issues:

- Results.to_json() => TypeError: Object of type 'UserWarning' is not JSON serializable #77
- Rethink Results structure #66

Merged pull requests:

- new dataframe tree traverser in to_json() #92 (gbordyugov)
- updated requirements.txt to have 'greater than' dependencies instead ... #89 (mkolarek)
- pip version requirement #88 (gbordyugov)
- Test #86 (s4826)
- merging in categorical binning #84 (gbordyugov)

v0.4.5 (2017-02-10)

Full Changelog

Fixed bugs:

- Numbers cannot appear in variable names for derived metrics #58

v0.4.4 (2017-02-09)

Full Changelog

Implemented enhancements:

- Add argument assume_normal and treatment_cost to calculate_prob_uplift_over_zero() and prob_uplift_over_zero_single_metric() #26
- host intro slides (from the ipython notebook) somewhere for public viewing #10

Closed issues:

- migrate issues from github enterprise #20

v0.4.3 (2017-02-07)

Full Changelog

Closed issues:

- coverage % is misleading #23

v0.4.2 (2016-12-08)

Full Changelog

Fixed bugs:

- frequency table in the chi square test doesn't respect the order of categories #56

v0.4.1 (2016-10-18)

Full Changelog

v0.4.0 (2016-08-19)

Full Changelog

Closed issues:

- Support 'overall ratio' metrics (e.g. conversion rate/return rate) as opposed to per-entity ratios #44

v0.3.4 (2016-08-08)

Full Changelog

Closed issues:

- perform trend analysis cumulatively #31
- Python3 #21

v0.3.3 (2016-08-02)

Full Changelog

v0.3.2 (2016-08-02)

Full Changelog

v0.3.1 (2016-07-15)

Full Changelog

v0.3.0 (2016-06-23)

Full Changelog

Implemented enhancements:

- Add P(uplift>0) as a statistic #2

v0.2.5 (2016-05-30)

Full Changelog

Implemented enhancements:

- Implement __version__ #14

Closed issues:

- upload full documentation! #1

v0.2.4 (2016-05-16)

Full Changelog

Closed issues:

- No module named experiment and test_data #13

v0.2.3 (2016-05-06)

Full Changelog

v0.2.2 (2016-05-06)

Full Changelog

v0.2.1 (2016-05-06)

Full Changelog

v0.2.0 (2016-05-06)

** This Change Log was automatically generated by 'github_changelog_generator <https://github.com/skywinder/Github-Changelog-Generator>'__*

CHAPTER 8

Indices and tables

- genindex
- modindex
- search