Twiser Documentation

Twiser Team

Apr 19, 2022

CONTENTS

Contents:

CHAPTER

ONE

GETTING STARTED

The Twiser package implements a Python library for variance reduction in A/B tests using pre-experiment covariates supporting publication[1]. These functions extend the idea of using pre-experiment data for variance reduction previously proposed in publication[2].

1.1 Installation

Only `Python>=3.7` is officially supported, but older versions of Python likely work as well.

The package is installed with:

```
pip install twiser
```

See GitHub, PyPI, and Read the Docs.

1.2 Example Usage

A full demo notebook of the package is given in `demo/survey_loan.ipynb`. Here is a snippet of the different methods from the notebook:

1.2.1 Setup a predictor as a control variate

First, we need to define a regression model. We can use anything that fits the sklearn idiom of `fit` and `predict` methods. This predictor is used to take the `n x d` array of treatment unit covariates `x_covariates` and predict the treatment outcomes `n`-length outcome array `x`. Likewise, it makes predictions from the `m x d` array of control unit covariates `y_covariates` to the control `m`-length outcome array `y`.

```
predictor = RandomForestRegressor(criterion="squared_error", random_state=0)
```

[1] R. Turner, U. Pavalanathan, S. Webb, N. Hammerla, B. Cohn, and A. Fu. Isotonic regression adjustment for variance reduction. In CODE@MIT, 2021.

[2] A. Deng, Y. Xu, R. Kohavi, and T. Walker. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pages 123–132, 2013.

1.2.2 Basic z-test

First, we apply the basic two-sample z-test included in Twiser. This works basically the same as `scipy.stats.ttest_ind`.

```
estimate, (lb, ub), pval = twiser.ztest(x, y, alpha=0.05)
show_output(estimate, (lb, ub), pval)
```

```
ATE estimate: 0.80 in (-0.14, 1.75), CI width of 1.89, p = 0.0954
```

1.2.3 Variance reduction with held out data

Next, we apply variance reduction where the predictor was trained on a held out 30% of the data. This is the easiest to show validity, but some of the added power is lost because not all data is used in the test.

```
estimate, (lb, ub), pval = twiser.ztest_held_out_train(
  x,
  x_covariates,
  y,
  y_covariates,
  alpha=0.05,
  train_frac=0.3,
  predictor=predictor,
  random=np.random.RandomState(123),
)
show_output(estimate, (lb, ub), pval)
```

```
ATE estimate: 1.40 in (0.20, 2.59), CI width of 2.39, p = 0.0217*
```

1.2.4 Variance reduction with cross validation

To be more statistically efficient we train and predict using 10-fold cross validation. Here, no data is wasted. As we can see it is a more significant result.

```
estimate, (lb, ub), pval = twiser.ztest_cross_val_train(
  x,
  x_covariates,
  y,
  y_covariates,
  alpha=0.05,
  k_fold=10,
  predictor=predictor,
  random=np.random.RandomState(123),
)
show_output(estimate, (lb, ub), pval)
```

```
ATE estimate: 1.38 in (0.51, 2.25), CI width of 1.74, p = 0.0019*
```

1.2.5 Variance reduction in-sample

In the literature it is popular to train the predictor in the same sample as the test. This often gives the most power. However, any overfitting in the predictor can also invalidate the results.

```
estimate, (lb, ub), pval = twiser.ztest_in_sample_train(
  x,
  x_covariates,
  y,
  y_covariates,
  alpha=0.05,
  predictor=predictor,
  random=np.random.RandomState(123),
)
show_output(estimate, (lb, ub), pval)
```

```
ATE estimate: 0.86 in (0.24, 1.49), CI width of 1.24, p = 0.0065*
```

1.2.6 Other interfaces

It is also possible to call these methods using raw control predictions instead of training the predictor in the Twiser method. It also supports a sufficient statistics interface for working with large datasets. See the documentation for details.

1.3 Support

Create a new issue.

1.4 Links

The source is hosted on GitHub.

The documentation is hosted at Read the Docs.

Installable from PyPI.

1.5 References

1.6 License

CHAPTER

TWO

USER GUIDE

The goal of this package is to make hypothesis testing using variance reduction methods as easy as using `scipy.stats.ttest_ind()` and `scipy.stats.ttest_ind_from_stats()`. A lot of the API is designed to match that simplicity as much as possible.

The publication in[1] was implented using this package. The variance reduction ideas here are built on top of the CUPED method ideas in[2] and[3].

The package currently supports three kinds of tests:

- basic z-test: This is the one from the intro stats textbooks.
- held out: This is a held out control variate method (train the predictor on a held out set).
- cross val: This is a k-fold cross validation type setup when training the predictor.

The distinction between basic, held out (aka cv), and cross val (aka stacked) is discussed in[4].

Each method has a few different ways to call it:

- basic: Call the method using the raw data and the control variate predictions.
- from stats: Call the method using sufficient statistics of the data and predictions only.
- train: Pass in a predictor object to train and evaluate the predictor in the routine.
  - For lack of a better choice, I assume the model has a sklearn-style *fit()* and *predict()* API.

Every statistical test in this package returns the same set of variables:

- A best estimate (of the difference of means)
- A confidence interval (on the difference of means)
- A p-value under the H0 that the two means are equal
  - The p-value and confidence interval are tested to be consistent with each under inversion.

[1] R. Turner, U. Pavalanathan, S. Webb, N. Hammerla, B. Cohn, and A. Fu. Isotonic regression adjustment for variance reduction. In CODE@MIT, 2021.

[2] A. Deng, Y. Xu, R. Kohavi, and T. Walker. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pages 123–132, 2013.

[3] A. Poyarkov, A. Drutsa, A. Khalyavin, G. Gusev, and P. Serdyukov. Boosted decision tree regression adjustment for variance reduction in online controlled experiments. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 235–244, 2016.

[4] I. Barr. Reducing the variance of A/B tests using prior information. Degenerate State, Jun 2018.

References

twiser.twiser.**ztest_from_stats**(*mean1*, *std1*, *nobs1*, *mean2*, *std2*, *nobs2*, ***, *alpha=0.05*)
Version of *ztest()* that works off the sufficient statistics of the data.

Parameters

- **mean1** (*float*) – The sample mean of the treatment group outcome x.
- **std1** (*float*) – The sample standard deviation of the treatment group outcome.
- **nobs1** (*int*) – The number of samples in the treatment group.
- **mean2** (*float*) – The sample mean of the control group outcome y.
- **std2** (*float*) – The sample standard deviation of the control group outcome.
- **nobs2** (*int*) – The number of samples in the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest**(*x*, *y*, ***, *alpha=0.05*, *ddof=1*)
Standard two-sample unpaired z-test. It does not assume equal sample sizes or variances.

Parameters

- **x** (numpy.ndarray of shape of shape (n,)) – Outcomes for the treatment group.
- **y** (numpy.ndarray of shape (m,)) – Outcomes for the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **ddof** (*int*) – The "Delta Degrees of Freedom" argument for computing sample variances.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_held_out_from_stats**(*mean1*, *cov1*, *nobs1*, *mean2*, *cov2*, *nobs2*, ***, *alpha=0.05*)
Version of *ztest_held_out()* that works off the sufficient statistics of the data.

Parameters

- **mean1** (numpy.ndarray of shape (2,)) – The sample mean of the treatment group outcome and its prediction: `[mean(x), mean(xp)]`.
- **cov1** (numpy.ndarray of shape (2, 2)) – The sample covariance matrix of the treatment group outcome and its prediction: `cov([x, xp])`.
- **nobs1** (*int*) – The number of samples in the treatment group.

- **mean2** (numpy.ndarray of shape (2,)) – The sample mean of the control group outcome and its prediction: `[mean(y), mean(yp)]`.
- **cov2** (numpy.ndarray of shape (2, 2)) – The sample covariance matrix of the control group outcome and its prediction: `cov([y, yp])`.
- **nobs2** (*int*) – The number of samples in the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_held_out**(*x*, *xp*, *y*, *yp*, ***, *alpha=0.05*, *health_check_output=True*, *ddof=1*)

Two-sample unpaired z-test with variance reduction using control variarates. It does not assume equal sample sizes or variances.

The predictions (control variates) must be derived from features that are independent of assignment to treatment or control. If the predictions in treatment and control have a different distribution then the test may be invalid.

Parameters

- **x** (numpy.ndarray of shape (n,)) – Outcomes for the treatment group.
- **xp** (numpy.ndarray of shape (n,)) – Predicted outcomes for the treatment group.
- **y** (numpy.ndarray of shape (m,)) – Outcomes for the control group.
- **yp** (numpy.ndarray of shape (m,)) – Predicted outcomes for the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **health_check_output** (*bool*) – If `True` perform a health check that ensures the predictions have the same distribution in treatment and control. If not, issue a warning.
- **ddof** (*int*) – The "Delta Degrees of Freedom" argument for computing sample variances.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_held_out_train**(*x*, *x_covariates*, *y*, *y_covariates*, ***, *alpha=0.05*, *train_frac=0.2*, *health_check_input=False*, *health_check_output=True*, *predictor=None*, *random=None*, *ddof=1*)

Version of *ztest_held_out()* that also trains the control variate predictor.

The covariates/features must be independent of assignment to treatment or control. If the features in treatment and control have a different distribution then the test may be invalid.

Parameters

- **x** (numpy.ndarray of shape (n,)) – Outcomes for the treatment group.

- **x_covariates** (numpy.ndarray of shape (n, d)) – Covariates/features for the treatment group.
- **y** (numpy.ndarray of shape (m,)) – Outcomes for the control group.
- **y_covariates** (numpy.ndarray of shape (m, d)) – Covariates/features for the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **train_frac** (*float*) – The fraction of data to hold out for training the predictors. To ensure test validity, we do not use the same data for training the predictors and performing the test. This must be inside the interval range `[0, 1]`.
- **health_check_input** (*bool*) – If `True` perform a health check that ensures the features have the same distribution in treatment and control. If not, issue a warning. It works by training a classifier to predict if a data point is in training or control. This can be slow for a large data set since it requires training a classifier.
- **health_check_output** (*bool*) – If `True` perform a health check that ensures the predictions have the same distribution in treatment and control. If not, issue a warning.
- **predictor** (*sklearn-like regression object*) – An object that has a *fit* and *predict* routine to make predictions. The object does not need to be fit yet. It will be fit in this method.
- **random** (numpy.random.RandomState) – An optional numpy random stream can be passed in for reproducibility.
- **ddof** (*int*) – The "Delta Degrees of Freedom" argument for computing sample variances.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_in_sample_train**(*x*, *x_covariates*, *y*, *y_covariates*, ***, *alpha=0.05*, *health_check_input=False*, *health_check_output=False*, *predictor=None*, *random=None*, *ddof=1*)

Version of *ztest_held_out()* that also trains the control variate predictor.

The covariates/features must be independent of assignment to treatment or control. If the features in treatment and control have a different distribution then the test may be invalid.

Parameters

- **x** (numpy.ndarray of shape (n,)) – Outcomes for the treatment group.
- **x_covariates** (numpy.ndarray of shape (n, d)) – Covariates/features for the treatment group.
- **y** (numpy.ndarray of shape (m,)) – Outcomes for the control group.
- **y_covariates** (numpy.ndarray of shape (m, d)) – Covariates/features for the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.

- **health_check_input** (*bool*) – If `True` perform a health check that ensures the features have the same distribution in treatment and control. If not, issue a warning. It works by training a classifier to predict if a data point is in training or control. This can be slow for a large data set since it requires training a classifier.
- **health_check_output** (*bool*) – If `True` perform a health check that ensures the predictions have the same distribution in treatment and control. If not, issue a warning.
- **predictor** (*sklearn-like regression object*) – An object that has a *fit* and *predict* routine to make predictions. The object does not need to be fit yet. It will be fit in this method.
- **random** (`numpy.random.RandomState`) – An optional numpy random stream can be passed in for reproducibility.
- **ddof** (*int*) – The "Delta Degrees of Freedom" argument for computing sample variances.

Return type `Tuple[float, Tuple[float, float], float]`

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_cross_val_from_stats**(*mean1*, *cov1*, *nobs1*, *mean2*, *cov2*, *nobs2*, *, *alpha=0.05*)

Version of *ztest_cross_val()* that works off the sufficient statistics of the data.

Parameters

- **mean1** (`numpy.ndarray` of shape (k, 2)) – The sample mean of the treatment group outcome and its prediction: `[mean(x), mean(xp)]`, for each fold in the k-fold cross validation.
- **cov1** (`numpy.ndarray` of shape (k, 2, 2)) – The sample covariance matrix of the treatment group outcome and its prediction: `cov([x, xp])`, for each fold in the k-fold cross validation.
- **nobs1** (`numpy.ndarray` of shape (k,)) – The number of samples in the treatment group, for each fold in the k-fold cross validation.
- **mean2** (`numpy.ndarray` of shape (k, 2)) – The sample mean of the control group outcome and its prediction: `[mean(y), mean(yp)]`, for each fold in the k-fold cross validation.
- **cov2** (`numpy.ndarray` of shape (k, 2, 2)) – The sample covariance matrix of the control group outcome and its prediction: `cov([y, yp])`, for each fold in the k-fold cross validation.
- **nobs2** (`numpy.ndarray` of shape (k,)) – The number of samples in the control group, for each fold in the k-fold cross validation.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.

Return type `Tuple[float, Tuple[float, float], float]`

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_cross_val**(*x*, *xp*, *x_fold*, *y*, *yp*, *y_fold*, ***, *alpha=0.05*, *health_check_output=True*)

Two-sample unpaired z-test with variance reduction using the cross validated control variarates method. It does not assume equal sample sizes or variances.

The predictions (control variates) must be derived from features that are independent of assignment to treatment or control. If the predictions in treatment and control have a different distribution then the test may be invalid.

Parameters

- **x** (numpy.ndarray of shape (n,)) – Outcomes for the treatment group.
- **xp** (numpy.ndarray of shape (n,)) – Predicted outcomes for the treatment group derived from a cross-validation routine.
- **x_fold** (numpy.ndarray of shape (n,)) – The cross validation fold assignment for each data point in treatment (of *dtype int*).
- **y** (numpy.ndarray of shape (m,)) – Outcomes for the control group.
- **yp** (numpy.ndarray of shape (m,)) – Predicted outcomes for the control group derived from a cross-validation routine.
- **y_fold** (numpy.ndarray of shape (n,)) – The cross validation fold assignment for each data point in control (of *dtype int*).
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **health_check_output** (*bool*) – If `True` perform a health check that ensures the predictions have the same distribution in treatment and control. If not, issue a warning.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_cross_val_train**(*x*, *x_covariates*, *y*, *y_covariates*, ***, *alpha=0.05*, *k_fold=5*, *health_check_input=False*, *health_check_output=True*, *predictor=None*, *random=None*)

Version of *ztest_cross_val()* that also trains the control variate predictor.

The covariates/features must be independent of assignment to treatment or control. If the features in treatment and control have a different distribution then the test may be invalid.

Parameters

- **x** (numpy.ndarray of shape (n,)) – Outcomes for the treatment group.
- **x_covariates** (numpy.ndarray of shape (n, d)) – Covariates/features for the treatment group.
- **y** (numpy.ndarray of shape (m,)) – Outcomes for the control group.
- **y_covariates** (numpy.ndarray of shape (m, d)) – Covariates/features for the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **k_fold** (*int*) – The number of folds in the cross validation: k.

- **health_check_input** (*bool*) – If `True` perform a health check that ensures the features have the same distribution in treatment and control. If not, issue a warning. It works by training a classifier to predict if a data point is in training or control. This can be slow for a large data set since it requires training a classifier.
- **health_check_output** (*bool*) – If `True` perform a health check that ensures the predictions have the same distribution in treatment and control. If not, issue a warning.
- **predictor** (*sklearn-like regression object*) – An object that has a *fit* and *predict* routine to make predictions. The object does not need to be fit yet. It will be fit in this method.
- **random** (`numpy.random.RandomState`) – An optional numpy random stream can be passed in for reproducibility.

Return type `Tuple[float, Tuple[float, float], float]`

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_cross_val_train_blockwise**(*x*, *x_covariates*, *y*, *y_covariates*, ***, *alpha=0.05*, *k_fold=5*, *health_check_input=False*, *health_check_output=True*, *predictor=None*, *random=None*)

Version of *ztest_cross_val_train()* that is more efficient if the fit routine scales worse than O(N), otherwise this will not be more efficient.

Parameters

- **x** (`numpy.ndarray` of shape (n,)) – Outcomes for the treatment group.
- **x_covariates** (`numpy.ndarray` of shape (n, d)) – Covariates/features for the treatment group.
- **y** (`numpy.ndarray` of shape (m,)) – Outcomes for the control group.
- **y_covariates** (`numpy.ndarray` of shape (m, d)) – Covariates/features for the control group.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **k_fold** (*int*) – The number of folds in the cross validation: k.
- **health_check_input** (*bool*) – If `True` perform a health check that ensures the features have the same distribution in treatment and control. If not, issue a warning. It works by training a classifier to predict if a data point is in training or control. This can be slow for a large data set since it requires training a classifier.
- **health_check_output** (*bool*) – If `True` perform a health check that ensures the predictions have the same distribution in treatment and control. If not, issue a warning.
- **predictor** (*sklearn-like regression object*) – An object that has a *fit* and *predict* routine to make predictions. The object does not need to be fit yet. It will be fit in this method.
- **random** (`numpy.random.RandomState`) – An optional numpy random stream can be passed in for reproducibility.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

twiser.twiser.**ztest_cross_val_train_load_blockwise**(*data_iter*, *, *alpha=0.05*, *predictor=None*, *callback=None*)

Version of *ztest_cross_val_train_blockwise()* that loads the data in blocks to avoid overflowing memory. Using *ztest_cross_val_train_blockwise()* is faster if all the data fits in memory.

Parameters

- **data_iter** (*Sequence[Callable[[], Tuple[ndarray, ndarray, ndarray, ndarray]]]*) – An iterable of functions, where each function returns a different cross validation fold. The functions should return data in the format of a tuple: `(x, x_covariates, y, y_covariates)`. See the parameters of *ztest_cross_val_train_blockwise()* for details on the shapes of these variables.
- **alpha** (*float*) – Required confidence level, typically this should be 0.05, and must be inside the interval range $[0, 1)$.
- **predictor** (*sklearn-like regression object*) – An object that has a *fit* and *predict* routine to make predictions. The object does not need to be fit yet. It will be fit in this method.
- **callback** (Optional[Callable[[Any], None]]) – An optional callback that gets called for each cross validation fold in the format `callback(predictor)`. This is sometimes useful for logging.

Return type Tuple[float, Tuple[float, float], float]

Returns

- *estimate* – Estimate of the difference in means: $\mathbb{E}[x] - \mathbb{E}[y]$.
- *ci* – Confidence interval (with coverage $1 - \alpha$) for the estimate.
- *pval* – The p-value under the null hypothesis H0 that $\mathbb{E}[x] = \mathbb{E}[y]$.

CHAPTER

THREE

HOW TO CONTRIBUTE

We'd love to get patches from you!

3.1 Getting Started

The following instructions have been tested with Python 3.7.4 on Mac OS (11.5.2).

3.2 Building dependencies

First, define the variables for the paths we will use:

```
GIT=/path/to/where/you/put/repos
ENVS=/path/to/where/you/put/virtualenvs
```

Then clone the repo in your git directory `$GIT`:

```
cd $GIT
git clone https://github.com/twitter/twiser.git
```

Inside your virtual environments folder `$ENVS`, make the environment:

```
cd $ENVS
virtualenv twiser --python=python3.7
source $ENVS/twiser/bin/activate
```

Now we can install the pip dependencies. Move back into your git directory and run

```
cd $GIT/twiser
pip install -r requirements/base.txt
pip install -e .  # Install the package itself
```

3.2.1 Install in editable mode

First, define the variables for the paths we will use:

```
GIT=/path/to/where/you/put/repos
ENVS=/path/to/where/you/put/virtualenvs
```

Then clone the repo in your git directory `$GIT`:

```
cd $GIT
git clone https://github.com/twitter/twiser.git
```

Inside your virtual environments folder `$ENVS`, make the environment:

```
cd $ENVS
virtualenv twiser --python=python3.7
source $ENVS/twiser/bin/activate
```

Now we can install the pip dependencies. Move back into your git directory and run

```
cd $GIT/twiser
pip install -r requirements/base.txt
pip install -e .  # Install the package itself
```

3.2.2 Contributor tools

First, we need to setup some needed tools:

```
cd $ENVS
virtualenv twiser_tools --python=python3.7
source $ENVS/twiser_tools/bin/activate
pip install -r $GIT/twiser/requirements/tools.txt
```

To install the pre-commit hooks for contributing run (in the `twiser_tools` environment):

```
cd $GIT/twiser
pre-commit install
```

To rebuild the requirements, we can run:

```
cd $GIT/twiser

# Check if there any discrepancies in the .in files
pipreqs twiser/ --diff requirements/base.in
pipreqs tests/ --diff requirements/tests.in
pipreqs docs/ --diff requirements/docs.in

# Regenerate the .txt files from .in files
pip-compile-multi --no-upgrade
```

3.3 Building the Project

The wheel (tar ball) for deployment as a pip installable package can be built using the script:

```
cd $GIT/twiser/
./build_wheel.sh
```

This script will only run if the git repo is clean, i.e., first run `git clean -x -ff -d`.

3.3.1 Building the documentation

First setup the environment for building with `Sphinx`:

```
cd $ENVS
virtualenv twiser_docs --python=python3.7
source $ENVS/twiser_docs/bin/activate
pip install -r $GIT/twiser/requirements/docs.txt
```

Then we can do the build:

```
cd $GIT/twiser/docs
make all
open _build/html/index.html
```

Documentation will be available in all formats in `Makefile`. Use `make html` to only generate the HTML documentation.

3.4 Workflow

We follow the GitHub Flow Workflow, which typically involves forking the project into your GitHub account, adding your changes to a feature branch, and opening a Pull Request to contribute those changes back to the project.

3.5 Testing

The tests for this package can be run with:

```
cd $GIT/twiser
./local_test.sh
```

The script creates an environment using the requirements found in `requirements/test.txt`. A code coverage report will also be produced in `$GIT/twiser/htmlcov/index.html`.

3.6 Style

The coding style is enforced the the pre-commit hooks in `.pre-commit-config.yaml`. Most importantly, just use black. That takes care of most of the formatting.

3.7 Issues

When filing an issue, try to adhere to the provided template if applicable. In general, the more information you can provide about exactly how to reproduce a problem, the easier it will be to help fix it.

CHAPTER

FOUR

CODE OF CONDUCT

We expect all contributors to abide by our Code of Conduct.

CHAPTER

FIVE

CREDITS

5.1 Development lead

Ryan Turner - Twitter (rdturnermtl)

5.2 Contributors

- Umashanthi Pavalanathan - Twitter

PYTHON MODULE INDEX

t

INDEX

M

T

Z