Eskapade-ROOT Documentation

KPMG Advanced Analytics
Big Data team

Dec 05, 2018

Contents

- Version: 0.8.5
- Released: Dec 2018

Eskapade is a light-weight, python-based data analysis framework, meant for modularizing all sorts of data analysis problems into reusable analysis components. For documentation on Eskapade, please go to this link.

Eskapade-ROOT is the ROOT-based extension of Eskapade. For documentation on Eskapade-ROOT, please go here.

CHAPTER 1

Release notes

1.1 Version 0.8

Version 0.8 of Eskapade-ROOT (August 2018) is a split off of the `root-analysis` module of Eskapade v0.7 into a separate package.

This way, Eskapade v0.8 no longer depends on ROOT. This new package Eskapade-ROOT does require ROOT to install, clearly.

CHAPTER 2

Installation

2.1 requirements

Eskapade-ROOT requires `Python 3.5+`, `Eskapade v0.8+`, `root_numpy 4.7.1` and `ROOT v6.10+`. These are pre-installed in the Eskapade docker.

2.2 pypi

To install the package from pypi, do:

```
$ pip install Eskapade-ROOT
```

2.3 github

Alternatively, you can check out the repository from github and install it yourself:

```
$ git clone https://github.com/KaveIO/Eskapade-ROOT.git eskapade-root
```

To (re)install the python code from your local directory, type from the top directory:

```
$ pip install -e eskapade-root
```

To (re)compile the cxx library, execute the following commands from the top directory:

```
$ cd cxx
$ cmake esroofit
$ cmake --build . -- -j1
$ cd ../
$ pip install -e .
```

2.4 python

After installation, you can now do in Python:

```
import esroofit
```

To load the Eskapade ROOT library in python, do:

```
from esroofit import roofit_utils
roofit_utils.load_libesroofit()
```

Congratulations, you are now ready to use Eskapade-ROOT!

CHAPTER 3

Quick run

To see the available Eskapade example, do:

```
$ export TUTDIR=`pip show Eskapade-ROOT | grep Location | awk '{ print $2"/esroofit/
↪tutorials" }'`
$ ls -l $TUTDIR/
```

E.g. you can now run:

```
$ eskapade_run $TUTDIR/esk401_roothist_fill_plot_convert.py
```

For all available examples, please see the tutorials.

CHAPTER 4

Contact and support

Contact us at: kave [at] kpmg [dot] com

Please note that the KPMG Eskapade group provides support only on a best-effort basis.

CHAPTER 5

Contents

5.1 Tutorials

This section contains materials on how to use Eskapade-ROOT. All command examples can be run from any directory with write access. For more in depth explanations on the functionality of the code-base, try the API docs.

5.1.1 All ROOT Examples in Eskapade

All Eskapade-ROOT example macros can be found in the tutorials directory. For ease of use, let's make a shortcut to the directory containing the tutorials:

```
$ export TUTDIR=`pip show Eskapade-ROOT | grep Location | awk '{ print $2"/esroofit/
↪tutorials" }'`
$ ls -l $TUTDIR/
```

The numbering of the example macros follows the package structure:

- `esk400+`: macros for processing ROOT datasets and performing analysis with ROOT.

These macros are briefly described below. You are encouraged to run all examples to see what they can do for you!

Example esk401: root histogram fill, plot, and convert

This macro illustrates how to 1) fill 1-3 dimensional root histograms from a pandas dataframe. In turn, these histogram are: 2) plotted, 3) converted to a roofit histogram (roodatahist), and 4) converted to a roofit dataset (roodataset).

```
$ eskapade_run $TUTDIR/esk401_roothist_fill_plot_convert.py
```

Example esk402: roodatahist filling from a pandas dataframe

This macro illustrates how to fill a N-dimensional roodatahist from a pandas dataframe. (A roodatahist can be filled iteratively, while looping over multiple pandas dataframes.) The roodatahist can be used to create a roofit histogram-pdf (roohistpdf).

```
$ eskapade_run $TUTDIR/esk402_roodatahist_fill.py
```

Example esk403: roodataset conversion into dataframe and back

This macro illustrates how to convert a pandas dataframe to a roofit dataset (= roodataset), do something to it with roofit, and then convert the roodataset back again to a pandas dataframe.

```
$ eskapade_run $TUTDIR/esk403_roodataset_convert.py
```

Example esk404: workspace to create a pdf, simulate, fit, and plot

Macro illustrates how do basic statistical data analysis with roofit, by making use of the rooworkspace functionality.

The example shows how to define a pdf, simulate data, fit this data, and then plot the fit result.

The generated data is converted to a dataframe and the contents is plotted with a default plotter link.

```
$ eskapade_run $TUTDIR/esk404_workspace_createpdf_simulate_fit_plot.py
```

Example esk405: simulation based on binned data

Imagine the situation where you wish to simulate an existing dataset, where you want the simulated dataset to have the same features and characteristics as the input dataset, including all known correlations between observables, possibly non-linear. The input data can have both categorical and continuous (float) observables.

This macro shows how this simulation can be done with roofit, by building a (potentially large) n-dimensional roofit histogram of all requested input observables with the RooDataHistFiller link.

Be careful not to blow up the total number of bins, which grows exponentially with the number of input observables. We can control this by setting the number of bins per continuous observable, or by setting the maximum total number of bins allowed in the histogram, which scales down the number of allowed bins in each continuous observable. Realize that, the more bins one has, the more input data is needed to will all bins with decent statistics.

This macro has two settings, controlled with settings['high_num_dims']. When false, the roodatahist contains 3 observables, of which two continous and 1 categorical. When true, the roodatahist is 6 dimensional, with 4 continous observables and 2 categorical ones. The latter example is slower, but works fine!

```
$ eskapade_run $TUTDIR/esk405_simulation_based_on_binned_data.py
```

Example esk406: simulation based on unbinned data

Imagine the situation where you wish to simulate an existing dataset consisting of continuous (float) observables only, where you want the simulated dataset to have the same features and characteristics as the input dataset, including the all correlations between observables.

This macro shows how this simulation can be done with roofit, by building a smooth pdf of the input dataset with kernel estimatation techniques, the so-called KEYS pdf, which describes the input observables and their correlations.

The technique works very well to describe 1 and 2 dimensional distributions, but is very cpu intensive and becomes ever more slow for higher number of dimensions.

This macro has two settings, controlled with settings['high_num_dims']. When false, the keys pdf contains 2 continuous observables. When true, the keys pdf 3 dimensional.

```
$ eskapade_run $TUTDIR/esk406_simulation_based_on_unbinned_data.py
```

Example esk407: classification unbiased fit estimate

This macro illustrates how to get an unbiased estimate of the number of high risk clients, by doing a template fit to data.

Assume a classifier has been trained and optimized to separate high-risk from low risk clients. But the high- to low-risk ratio in data is very low and unknown, so the false-positive rate is non-negligible.

We can use templates of the score of the ML classifier of the high- and low-risk testing samples to (at least) get an unbiased estimate of the total number of high-risk clients. This is done by fitting the (unbiased) testing templates to the score distribution in the actual dataset. The shapes differentiate the number of high- and low-risk clients.

```
$ eskapade_run $TUTDIR/esk407_classification_unbiased_fit_estimate.py
```

Example esk408: classification error propagation after fit

This macro continues on the idea in esk407_classification_unbiased_fit_estimate. It illustrates how to assign statistically motivated probabilities to high risk clients, by doing a template fit to data, and - based on this - calculating the probability and uncertainty on this for each client.

Assume a classifier has been trained and optimized to separate high-risk from low risk clients. But the high- to low-risk ratio in data is very low and unknown, so the false-positive rate is non-negligible.

We can use templates of the score of the ML classifier of the high- and low-risk testing samples to (at least) score the probability that someone is a high risk client, in light of the fact that most clients with a high classifier score will in fact be false-positive low risk clients.

In addition to the probability, the algorithm assigns as statistical uncertainty to each probability. The total sum of these probabilities equals the number of estimated high-risk clients, as also obtained in example esk407.

```
$ eskapade_run $TUTDIR/esk408_classification_error_propagation_after_fit.py
```

Example esk409: unredeemed vouchers

This macro is an example of an application of the truncated exponential PDF that is provided by Eskapade. The redeem of gift vouchers by customers of a store is modelled.

Vouchers are given out to customers of the store and can be exchanged for goods sold in the store. All vouchers represent the same amount of money and can only be used once. They are given to customers in batches at different dates.

Not all released vouchers are actually spent. To estimate how many currently released vouchers will be spent, the voucher age at which the redeem takes place is modelled by a double-exponential decay model. The exponential PDF is truncated at the voucher age, beyond which there can have been no redeems yet. Once the parameters of the model have been fit to (generated) redeem-event data, the total number of redeems at infinite voucher ages is estimated by scaling to the surface of an untruncated PDF with identical parameter values.

```
$ eskapade_run $TUTDIR/esk409_unredeemed_vouchers.py
```

Example esk410: testing correlations between categories

This macro illustrates how to find correlations between categorical observables.

Based on the hypothesis of no correlation expected frequencies of observations * are calculated. The measured frequencies are compared to expected frequencies. * From these the (significance of the) p-value of the hypothesis that the observables in the input dataset are not correlated is determined. The normalized residuals (pull values) for each bin in the dataset are also calculated. A detailed description of the method can be found in ABCDutils.h. A description of the method to calculate the expected frequencies can be found * in RooABCDHistPDF.cxx.

```
$ eskapade_run $TUTDIR/esk410_testing_correlations_between_categories.py
```

Example esk411: weibull predictive maintenance

Macro illustrates how to fit several Weibull distributions to a falling time difference distribution, indicating times between maintenance. The Weibull probability distribution is provided by Eskapade.

```
$ eskapade_run $TUTDIR/esk411_weibull_predictive_maintenance.py
```

5.1.2 Tutorial5: using RooFit

This section provides a tutorial on how to use RooFit in Eskapade. RooFit is an advanced fitting library in ROOT, which is great for modelling all sorts of data sets. See this tutorial for a 20 min introduction into RooFit. ROOT (and RooFit) works 'out of the box' in the Eskapade docker/vagrant image.

In this tutorial we will illustrates how to define a new probability density function (pdf) in RooFit, how to compile it, and how to use it in Eskapade to simulate a dataset, fit it, and plot the results.

Note: There are many good RooFit tutorials. See the macros in the directory `$ROOTSYS/tutorials/roofit/` of your local ROOT installation. This tutorial is partially based on the RooFit tutorial `$ROOTSYS/tutorials/roofit/rf104_classfactory.C`.

Building a new probability density function

Before using a new model in Eskapade, we need to create, compile and load a probability density function model in RooFit.

Move to the directory:

```
$ cd cxx/esroofit/src/
```

Start an interactive python session and type:

```
import ROOT
ROOT.RooClassFactory.makePdf("MyPdfV2","x,A,B","","A*fabs(x)+pow(x-B,2)")
```

This command creates a RooFit skeleton probability density function class named `MyPdfV2`, with the variable `x, a, b` and the given formula expression.

Also type:

```
ROOT.RooClassFactory.makePdf("MyPdfV3","x,A,B","","A*fabs(x)+pow(x-B,2)",True,False,
↪"x:(A/2)*(pow(x.max(rangeName),2)+pow(x.min(rangeName),2))+(1./3)*(pow(x.
↪max(rangeName)-B,3)-pow(x.min(rangeName)-B,3))")
```

This creates the RooFit p.d.f. class `MyPdfV3`, with the variable `x, a, b` and the given formula expression, and the given expression for analytical integral over `x`.

Exit python `(Ctrl-D)` and type:

```
$ ls -l MyPdf*
```

You will see two cxx files and two header files. Open the file `MyPdfV2.cxx`. You should see an `evaluate()` method in terms of `x, a` and `b` with the formula expression we provided.

Now open the file `MyPdfV3.cxx`. This also contains the method `analyticalIntegral()` with the expresssion for the analytical integral over x that we provided.

If no analytical integral has been provided, as in `MyPdfV2`, RooFit will try to try to compute the integral itself. (Of course this is a costly operation.) If you wish, since we know the analytical integral for `MyPdfV2`, go ahead and edit `MyPdfV2.cxx` to add the expression of the analytical integral to the class.

As another example of a simple pdf class, take a look at the expressions in the file: `cxx/esroofit/src/RooWeibull.cxx`.

Now move the header files to their correct location:

```
$ mv MyPdfV*.h ../include/
```

To make sure that these classes get picked up in Eskapade roofit libary, open the file `cxx/esroofit/dict/esroofit/LinkDef.h` and add the lines:

```
#pragma link C++ class MyPdfV2+;
#pragma link C++ class MyPdfV3+;
```

Finally, let's compile the C++ code of these classes:

```
$ cd cxx
$ cmake esroofit
$ cmake --build . -- -j1
$ cd ../
$ pip install -e .
```

You should see the compiler churning away, processing several existing classes but also `MyPdfV2` and `MyPdfV3`.

We are now able to open the Eskapade roofit library, so we can use these classes in python:

```
from esroofit import roofit_utils
roofit_utils.load_libesroofit()
```

In fact, this last snippet of code is used in the tutorial macro right below.

Running the tutorial macro

Let's take a look at the steps in tutorial macro `$TUTDIR/tutorial_5.py`. The macro illustrates how do basic statistical data analysis with roofit, by making use of the `RooWorkspace` functionality. A `RooWorkspace` is a

persistable container for RooFit projects. A workspace can contain and own variables, p.d.f.s, functions and datasets. The example shows how to define a pdf, simulate data, fit this data, and then plot the fit result. There are 5 sections; they are detailed in the sections below.

The next step is to run the tutorial macro.

```
$ eskapade_run $TUTDIR/tutorial_5.py
```

Let's discuss what we are seeing on the screen.

Loading the Eskapade ROOT library

The macro first checks the existence of the class `MyPdfV3` that we just created in the previous section.

```
# --- 0. make sure Eskapade RooFit library is loaded

# --- load and compile the Eskapade roofit library
from esroofit import roofit_utils
roofit_utils.load_libesroofit()

# --- check existence of class MyPdfV3 in ROOT
pdf_name = 'MyPdfV3'
logger.info('Now checking existence of ROOT class {name}', name=pdf_name)
cl = ROOT.TClass.GetClass(pdf_name)
if not cl:
    logger.fatal('Could not find ROOT class {name}. Did you build and compile it␣
↪correctly?', name=pdf_name)
    sys.exit(1)
else:
    logger.info('Successfully found ROOT class {name}', name=pdf_name)
```

In the output on the screen, look for `Now checking existence of ROOT class MyPdfV3`. If this was successful, it should then say `Successfully found class MyPdfV3`.

Instantiating a pdf

The link `WsUtils`, which stands for `RooWorkspace` utils, allows us to instantiate a pdf. Technically, one defines a model by passing strings to the rooworkspace factory. For examples on using the rooworkspace factory see basic, operations and tools for more details. The entire rooworkspace factory syntax can be found at commands.

```
ch = Chain('WsOps')

# --- instantiate a pdf
wsu = WsUtils(name = 'modeller')
wsu.factory = ["MyPdfV3::testpdf(y[-10,10],A[10,0,100],B[2,-10,10])"]
ch.add(wsu)
```

Here we use the pdf class we just created (`MyPdfV3`) to create a pdf called `testpdf`, with observable `y` and parameter `A` and `B`, having ranges `(-10,10)`, `(0,100)` and `(-10,10)` respectively, and with initial values for `A` and `B` of `10` and `2` respectively.

Simulating data

The link `WsUtils` is then used to simulate records according to the shape of `testpdf`.

```
wsu = WsUtils(name = 'simulater')
wsu.add_simulate(pdf='testpdf', obs='y', num=400, key='simdata')
ch.add(wsu)
```

Here we simulate `400` records of observable `y` with pdf `testpdf` (which is of type MyPdfV3). The simulated data is stored in the datastore under key `simdata`.

Fitting the data

Another version of the link `WsUtils` is then used to fit the simulated records with the pdf `testpdf`.

```
wsu = WsUtils(name = 'fitter')
wsu.pages_key='report_pages'
wsu.add_fit(pdf='testpdf', data='simdata', key='fit_result')
ch.add(wsu)
```

The link performs a fit of pdf `testpdf` to dataset `simdata`. We store the fit result object in the datastore under key `fit_result`. The fit knows from the input dataset that the observable is `y`, so that the fit parameters are `A` and `B`.

Plotting the fit result

Finally, the last version of the link `WsUtils` is used to plot the result of the fit on top of simulated data.

```
wsu = WsUtils(name = 'plotter')
wsu.pages_key='report_pages'
wsu.add_plot(obs='y', data='simdata', pdf='testpdf', pdf_kwargs={'VisualizeError':
↪'fit_result', 'MoveToBack': ()}, key='simdata_plot')
wsu.add_plot(obs='y', pdf='testpdf', file='fit_of_simdata.pdf', key='simdata_plot')
ch.add(wsu)
```

This link is configured to do two things. First it plots the observable `y` of the the dataset `simdata` and then plots the fitted uncertainy band of the pdf `testpdf` on top of this. The plot is stored in the datastore under the key `simdata_plot`. Then it plots the fitted pdf `testpdf` without uncertainty band on top of the same frame `simdata_plot`. The resulting plot is stored in the file `fit_of_simdata.pdf`

Fit report

The link `WsUtils` produces a summary report of the fit it has just performed. The pages of this report are stored in the datastore under the key `report_pages`. At the end of the Eskapade session, the plots and latex files produced by this tutorial are written out to disk.

The fit report can be found at:

```
$ cd results/tutorial_5/data/v0/report/
$ pdflatex report.tex
```

Take a look at the resulting fit report: `report.pdf`. It contains pages summarizing: the status and quality of the fit (including the correlation matrix), summary tables of the floating and fixed parameters in the fit, as well as the plot we have produced.

5.2 Developing and Contributing

5.2.1 Working on Eskapade-ROOT

You have some cool feature and/or algorithm you want to add to Eskapade-ROOT. How do you go about it?

First clone Eskapade-ROOT.

```
git clone https://github.com:KaveIO/Eskapade-ROOT.git eskapade-root
```

then

```
pip install -e eskapade-root
```

this will install Eskapade in editable mode, which will allow you to edit the code and run it as you would with a normal installation of eskapade.

To make sure that everything works try executing eskapade without any arguments, e.g.

```
eskapade_run --help
```

or you could just execute the tests using either the eskapade test runner, e.g.

```
eskapade_trial .
```

That's it.

5.2.2 Contributing

When contributing to this repository, please first discuss the change you wish to make via issue, email, or any other method with the owners of this repository before making a change. You can find the contact information on the index page.

Note that when contributing that all tests should succeed.

5.3 References

- Web page: https://eskapade-root.readthedocs.io
- Repository: https://github.com/kaveio/eskapade-root
- Issues & Ideas: https://github.com/kaveio/eskapade-root/issues
- Eskapade: http://eskapade.kave.io
- Contact us at: kave [at] kpmg [dot] com

5.4 API

5.4.1 API Documentation

EskapadeROOT

esroofit package

Subpackages

esroofit.decorators package

Submodules

esroofit.decorators.histograms module

esroofit.decorators.roofit module

Module contents

esroofit.links package

Submodules

esroofit.links.add_propagated_error_to_roodataset module

esroofit.links.convert_dataframe_2_roodataset module

esroofit.links.convert_roodataset_2_dataframe module

esroofit.links.convert_roodataset_2_roodatahist module

esroofit.links.convert_root_hist_2_roodatahist module

esroofit.links.convert_root_hist_2_roodataset module

esroofit.links.print_ws module

esroofit.links.read_from_root_file module

esroofit.links.roodatahist_filler module

esroofit.links.roofit_percentile_binning module

esroofit.links.root_hist_filler module

esroofit.links.trunc_exp_fit module

esroofit.links.trunc_exp_gen module

esroofit.links.uncorrelation_hypothesis_tester module

- modindex