Data-driven Model Falsification in Python

Subhayan De, Ph.D.
Postdoctoral Associate
University of Colorado Boulder
Email:Subhayan.De@colorado.edu
Website:www.subhayande.com

1 Data-driven Model Falsification Module

For a theoretical overview on model falsification please see:

1. De, Subhayan, et al. "Investigation of model falsification using error and likelihood bounds with application to a structural system." *Journal of Engineering Mechanics* 144.9 (2018): 04018078.

2. De, Subhayan, et al. "A hybrid probabilistic framework for model validation with application to structural dynamics modeling." *Mechanical Systems and Signal Processing* 121 (2019): 961-980.

Download the module from https://github.com/subhayande/Model_Falsification. See the demo (`fals_test1.py`) for an example of the implementation.

Required packages: numpy, scipy, time

NOTE: Currently, only Gaussian distributions for residual errors are allowed and two-sided hypothesis tests are implemented.

Report any bugs to Subhayan.De@colorado.edu

1.1 Procedures implemented

Error-bound and Likelihood-bound model falsification using error criteria:

- Familywise error rate (FWER)
  - Bonferroni
  - Sidák
- False discovery rate (FDR)
  - BH procedure

1.2 EBBonferroni

```
===========================================================================
|        Error-bound Falsification class using Bonferroni criterion      |
|              (derived from Error-bound Falsification class)            |
===========================================================================
Initialization:
Fals = EBBonferroni(nModels, nMeas, alpha, paramDist, distVar, meas, func,
residualSD)
===========================================================================
Attributes:
models:         Model instances
nModels:        Total number of models
nParam:         Number of uncertain parameter in a model
paramDist:      Names of the probability distributions of the uncertain
parameters.
(Currently supports normal/gaussian, uniform, lognormal)
distVar:        Variables of the uncertain parameter distribution
nMeas:          Total number of measurements
predictions:    Predictions by the models
residualSD:     Standard deviation of the residual error
pValues:        p values for predictions from all the models
alphaValues:    alpha values for predictions from all the models
func:           prediction function
version:        version of the code
===========================================================================
Methods:
doFalsification:        performs falsification
using Bonferroni correction
inherited:
genModels:          generates all the model instances
funEval:            predicts
getAlphapValues:    calculates alpha and p values
===========================================================================
Reference: De, S., Brewick, P.T., Johnson, E.A. and Wojtkiewicz, S.F., 2018
"Investigation of Model Falsification Using Error and Likelihood Bounds
with Application to a Structural System."
Journal of Engineering Mechanics, 144(9), p.04018078.
===========================================================================
written by Subhayan De (email:Subhayan.De@colorado.edu)
===========================================================================
```

1.3 EBSidak

```
===========================================================================
|           Error-bound Falsification class using Sidak criterion        |
|              (derived from Error-bound Falsification class)            |
===========================================================================
```

```
Initialization:
Fals = EBSidak(nModels, nMeas, alpha, paramDist, distVar, meas, func,
residualSD)
===========================================================================
Attributes:
models:         Model instances
nModels:        Total number of models
nParam:         Number of uncertain parameter in a model
paramDist:      Names of the probability distributions of the uncertain
parameters.
(Currently supports normal/gaussian, uniform, lognormal)
distVar:        Variables of the uncertain parameter distribution
nMeas:          Total number of measurements
predictions:    Predictions by the models
residualSD:     Standard deviation of the residual error
pValues:        p values for predictions from all the models
alphaValues:    alpha values for predictions from all the models
func:           prediction function
version:        version of the code
===========================================================================
Methods:
doFalsification:        performs falsification using Sidak correction
inherited:
genModels:          generates all the model instances
funEval:            predicts
getAlphapValues:    calculates alpha and p values
===========================================================================
Reference: De, S., Brewick, P.T., Johnson, E.A. and Wojtkiewicz, S.F., 2018
"Investigation of Model Falsification Using Error and Likelihood Bounds
with Application to a Structural System."
Journal of Engineering Mechanics, 144(9), p.04018078.
===========================================================================
written by Subhayan De (email:Subhayan.De@colorado.edu)
===========================================================================
```

1.4 EBBH

```
===========================================================================
|            Error-bound Falsification class using BH procedure           |
|              (derived from Error-bound Falsification class)             |
===========================================================================
Initialization:
Fals = EBBH(nModels, nMeas, alpha, paramDist, distVar, meas, func,
residualSD)
===========================================================================
Attributes:
models:         Model instances
nModels:        Total number of models
```

```
nParam:         Number of uncertain parameter in a model
paramDist:      Names of the probability distributions of the uncertain
parameters.
(Currently supports normal/gaussian, uniform, lognormal)
distVar:        Variables of the uncertain parameter distribution
nMeas:          Total number of measurements
predictions:    Predictions by the models
residualSD:     Standard deviation of the residual error
pValues:        p values for predictions from all the models
alphaValues:    alpha values for predictions from all the models
func:           prediction function
version:        version of the code
===========================================================================
Methods:
doFalsification:        performs falsification using BH procedure
inherited:
genModels:          generates all the model instances
funEval:            predicts
getAlphapValues:    calculates alpha and p values
===========================================================================
Reference: De, S., Brewick, P.T., Johnson, E.A. and Wojtkiewicz, S.F., 2018
"Investigation of Model Falsification Using Error and Likelihood Bounds
with Application to a Structural System."
Journal of Engineering Mechanics, 144(9), p.04018078.
===========================================================================
written by Subhayan De (email:Subhayan.De@colorado.edu)
===========================================================================
```

1.5 LBBonferroni

```
===========================================================================
|     Likelihood-bound Falsification class using Bonferroni criterion     |
|           (derived from Likelihood-bound Falsification class)           |
===========================================================================
Initialization:
Fals = LBBonferroni(nModels, nMeas, alpha, paramDist, distVar, meas, func,
residualSD)
===========================================================================
Attributes:
models:         Model instances
nModels:        Total number of models
nParam:         Number of uncertain parameter in a model
paramDist:      Names of the probability distributions of the uncertain
parameters.
(Currently supports normal/gaussian, uniform, lognormal)
distVar:        Variables of the uncertain parameter distribution
nMeas:          Total number of measurements
predictions:    Predictions by the models
```

```
residualSD:     Standard deviation of the residual error
pValues:        p values for predictions from all the models
alphaValues:    alpha values for predictions from all the models
func:           prediction function
version:        version of the code
===========================================================================
Methods:
doFalsification:        performs falsification using
Bonferroni criterion
inherited:
genModels:          generates all the model instances
funEval:            predicts
getAlphapValues:    calculates alpha and p values
===========================================================================
Reference: De, S., Brewick, P.T., Johnson, E.A. and Wojtkiewicz, S.F., 2018
"Investigation of Model Falsification Using Error and Likelihood Bounds
with Application to a Structural System."
Journal of Engineering Mechanics, 144(9), p.04018078.
===========================================================================
written by Subhayan De (email:Subhayan.De@colorado.edu)
===========================================================================
```

1.6 LBSidak

```
===========================================================================
|        Likelihood-bound Falsification class using Sidak criterion       |
|           (derived from Likelihood-bound Falsification class)           |
===========================================================================
Initialization:
Fals = LBSidak(nModels, nMeas, alpha, paramDist, distVar, meas, func,
residualSD)
===========================================================================
Attributes:
models:         Model instances
nModels:        Total number of models
nParam:         Number of uncertain parameter in a model
paramDist:      Names of the probability distributions of the uncertain
parameters.
(Currently supports normal/gaussian, uniform, lognormal)
distVar:        Variables of the uncertain parameter distribution
nMeas:          Total number of measurements
predictions:    Predictions by the models
residualSD:     Standard deviation of the residual error
pValues:        p values for predictions from all the models
alphaValues:    alpha values for predictions from all the models
func:           prediction function
version:        version of the code
===========================================================================
```

```
Methods:
doFalsification:        performs falsification using
Bonferroni criterion
inherited:
genModels:          generates all the model instances
funEval:            predicts
getAlphapValues:    calculates alpha and p values
===========================================================================
Reference: De, S., Brewick, P.T., Johnson, E.A. and Wojtkiewicz, S.F., 2018
"Investigation of Model Falsification Using Error and Likelihood Bounds
with Application to a Structural System."
Journal of Engineering Mechanics, 144(9), p.04018078.
===========================================================================
written by Subhayan De (email:Subhayan.De@colorado.edu)
===========================================================================
```

1.7 LBBH

```
===========================================================================
|         Likelihood-bound Falsification class using BH procedure         |
|           (derived from Likelihood-bound Falsification class)           |
===========================================================================
Initialization:
Fals = LBBH(nModels, nMeas, alpha, paramDist, distVar, meas, func,
residualSD)
===========================================================================
Attributes:
models:         Model instances
nModels:        Total number of models
nParam:         Number of uncertain parameter in a model
paramDist:      Names of the probability distributions of the uncertain
parameters.
(Currently supports normal/gaussian, uniform, lognormal)
distVar:        Variables of the uncertain parameter distribution
nMeas:          Total number of measurements
predictions:    Predictions by the models
residualSD:     Standard deviation of the residual error
pValues:        p values for predictions from all the models
alphaValues:    alpha values for predictions from all the models
func:           prediction function
version:        version of the code
===========================================================================
Methods:
doFalsification:        performs falsification using
Bonferroni criterion
inherited:
genModels:          generates all the model instances
funEval:            predicts
```

```
getAlphapValues:    calculates alpha and p values
============================================================================
Reference: De, S., Brewick, P.T., Johnson, E.A. and Wojtkiewicz, S.F., 2018
"Investigation of Model Falsification Using Error and Likelihood Bounds
with Application to a Structural System."
Journal of Engineering Mechanics, 144(9), p.04018078.
============================================================================
written by Subhayan De (email:Subhayan.De@colorado.edu)
============================================================================
```

2 Example

NOTE: Implementation of this example is in:

```
fals_test1.py
```

Consider the following linear regression problem:

$$y = 3 + 4.5x + \text{noise} \tag{1}$$

where the regression parameters are $\theta = [3, 4.5]^T$. Using 200 measurements the above methods are implemented. The priors for the coefficients are assumed as $\mathcal{U}(2,5)$ and $\mathcal{N}(4, 0.5^2)$, *i.e.*, uniform and Gaussian, respectively. The problem is initialized using

```
# number of measurements
nMeas = 200

# parameters
w1 = 3.0
w2 = 4.5

# noisy data
X = 2.0*np.random.rand(nMeas,1)
np.savetxt('fals1_data.txt',X)
y = w1 + w2 * X + np.random.randn(nMeas,1)

# Model definition
thetaDist = np.array(['uniform','normal'])
modelDistVar = np.array([[2,5],[4.0,0.5]])

def fun(param):
    """
    function to calculate model predictions
    """
    X = np.loadtxt('fals1_data.txt')
    pred = param[0] + param[1]*X
    return pred
```

2.1 EBBonferroni

```
from Falsification import EBBonferroni

fls = EBBonferroni(nModels =4000, nMeas=nMeas, alpha=0.05,
                   paramDist=thetaDist, distVar=modelDistVar,
                   meas = y, func = fun, residualSD = 1.0) # initialize

fls.doFalsification() # perform falsification
```

The output is shown below:

```
Using Error-bound model falsification with Bonferroni criterion


Total number of candidate models =  1000


---------------------------
STEP I: Generating candidate models

Error-bound Bonferroni |█████████████████████████████████| 100.0% Complete:
Time Elapsed = 0.0s

---------------------------
STEP II: Predicting using candidate models

Error-bound Bonferroni |█████████████████████████████████| 100.0% Complete:
Time Elapsed = 1.83s

---------------------------
STEP III: Calculating alpha and p values

Error-bound Bonferroni |█████████████████████████████████| 100.0% Complete:
Time Elapsed = 77.26s

---------------------------
STEP IV: Falsifying models

Error-bound Bonferroni |█████████████████████████████████| 100.0% Complete:
Time Elapsed = 77.57s

Number of unfalsified models =  573


Percent unfalsified = 57.30 %
```

2.2 EBSidak

```
from Falsification import EBSidak
```

```
fls = EBSidak(nModels =1000, nMeas=nMeas, alpha=0.05,
              paramDist=thetaDist, distVar=modelDistVar,
              meas = y, func = fun, residualSD = 1.0) # initialize

fls.doFalsification() # perform falsification
```

The output is the following:

```
Using Error-bound model falsification with Sidak criterion


Total number of candidate models =  1000



---------------------------
STEP I: Generating candidate models

Error-bound Sidak |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 0.0s

---------------------------
STEP II: Predicting using candidate models

Error-bound Sidak |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 1.82s

---------------------------
STEP III: Calculating alpha and p values

Error-bound Sidak |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 76.23s

---------------------------
STEP IV: Falsifying models

Error-bound Sidak |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 76.53s

Number of unfalsified models =  572


Percent unfalsified = 57.20 %
```

2.3 EBBH

```
from Falsification import EBBH

fls = EBBH(nModels =1000, nMeas=nMeas, alpha=0.05,
```

```
           paramDist=thetaDist, distVar=modelDistVar,
           meas = y, func = fun, residualSD = 1.0) # initialize

fls.doFalsification() # perform falsification
```

The output is the following:

```
Using Error-bound model falsification with BH procedure


Total number of candidate models =  1000


---------------------------
STEP I: Generating candidate models

Error-bound BH procedure |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 0.0s

---------------------------
STEP II: Predicting using candidate models

Error-bound BH procedure |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 1.92s

---------------------------
STEP III: Calculating alpha and p values

Error-bound BH procedure |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 79.53s

---------------------------
STEP IV: Falsifying models

Error-bound BH procedure |██████████████████████████████████| 100.0% Complete:
Time Elapsed = 79.86s

Number of unfalsified models =  507


Percent unfalsified = 50.70 %
```

2.4 LBBonferroni

```
from Falsification import LBBonferroni

fls = LBBonferroni(nModels =1000, nMeas=nMeas, alpha=0.05,
                   paramDist=thetaDist, distVar=modelDistVar,
                   meas = y, func = fun, residualSD = 0.4) # initialize
```

```
fls.doFalsification() # perform falsification
```

The output is the following:

```
Using Likelihood-bound model falsification with Bonferroni criterion


Total number of models =  1000


---------------------------
STEP I: Generating candidate models

Likelihood-bound Bonferroni criterion
|█████████████████████████████████| 100.0% Complete:
Time Elapsed = 0.0s

---------------------------
STEP II: Predicting using candidate models

Likelihood-bound Bonferroni criterion
|█████████████████████████████████| 100.0% Complete:
Time Elapsed = 1.96s

---------------------------
STEP III: Calculating alpha and p values

Likelihood-bound Bonferroni criterion
|█████████████████████████████████| 100.0% Complete:
Time Elapsed = 79.68s

---------------------------
STEP IV: Calculating likelihood of the candidate models

Likelihood-bound Bonferroni criterion
|█████████████████████████████████| 100.0% Complete:
Time Elapsed = 111.81s

---------------------------
STEP V: Falsifying models

Likelihood-bound Bonferroni criterion
|█████████████████████████████████| 100.0% Complete:
Time Elapsed = 112.08s

Number of unfalsified models =  595
```

```
Percent unfalsified = 59.50 %
```

2.5 LBSidak

```
from Falsification import LBSidak

fls = LBSidak(nModels =1000, nMeas=nMeas, alpha=0.05,
              paramDist=thetaDist, distVar=modelDistVar,
              meas = y, func = fun, residualSD = 0.4) # initialize

fls.doFalsification() # perform falsification
```

The output is the following:

```
Using Likelihood-bound model falsification with Sidak criterion


Total number of models =  1000


---------------------------
STEP I: Generating candidate models

Likelihood-bound Sidak criterion |██████████████████████████|
100.0% Complete:
Time Elapsed = 0.0s

---------------------------
STEP II: Predicting using candidate models

Likelihood-bound Sidak criterion |██████████████████████████|
100.0% Complete:
Time Elapsed = 1.98s

---------------------------
STEP III: Calculating alpha and p values

Likelihood-bound Sidak criterion |██████████████████████████|
100.0% Complete:
Time Elapsed = 79.37s

---------------------------
STEP IV: Calculating likelihood of the candidate models

Likelihood-bound Sidak criterion |██████████████████████████|
100.0% Complete:
Time Elapsed = 111.25s
```

```
---------------------------
STEP V: Falsifying models

Likelihood-bound Sidak criterion |██████████████████████████████████████████████████|
100.0% Complete:
Time Elapsed = 111.52s

Number of unfalsified models =  592


Percent unfalsified = 59.20 %
```

2.6 LBBH

```
from Falsification import LBBH

fls = LBBH(nModels =1000, nMeas=nMeas, alpha=0.05,
           paramDist=thetaDist, distVar=modelDistVar,
           meas = y, func = fun, residualSD = 0.4) # initialize

fls.doFalsification() # perform falsification
```

The output is the following:

```
Using Likelihood-bound model falsification with BH procedure


Total number of models =  1000


---------------------------
STEP I: Generating candidate models

Likelihood-bound BH procedure |██████████████████████████████████████████████████| 100.0% Complete:
Time Elapsed = 0.0s

---------------------------
STEP II: Predicting using candidate models

Likelihood-bound BH procedure |██████████████████████████████████████████████████| 100.0% Complete:
Time Elapsed = 1.97s

---------------------------
STEP III: Calculating alpha and p values

Likelihood-bound BH procedure |██████████████████████████████████████████████████| 100.0% Complete:
Time Elapsed = 80.2s
```

```
---------------------------
STEP IV: Calculating likelihood of the candidate models

Likelihood-bound BH procedure |██████████████████████████| 100.0% Complete:
Time Elapsed = 112.59s

---------------------------
STEP V: Falsifying models

Likelihood-bound BH procedure |██████████████████████████| 100.0% Complete:
Time Elapsed = 112.85s

Number of unfalsified models =  185


Percent unfalsified = 18.50 %
```