otpod Documentation

Release 0.0.1

Antoine Dumas

April 13, 2016

CONTENTS:

1.1 Documentation of the API

This is the user manual for the Python bindings to the otpod library.

1.1.1 Data analysis

UnivariateLinearModelAnalysis	Linear regression analysis with residuals hypothesis tests.

UnivariateLinearModelAnalysis

class **UnivariateLinearModelAnalysis**(**args*)

   Linear regression analysis with residuals hypothesis tests.

   Available constructors:

   UnivariateLinearModelAnalysis(*inputSample, outputSample*)

   UnivariateLinearModelAnalysis(*inputSample, outputSample, noiseThres, saturationThres, resDistFact, boxCox*)

> **Parameters** **inputSample** : 2-d sequence of float
>
>> Vector of the defect sizes, of dimension 1.
>
> **outputSample** : 2-d sequence of float
>
>> Vector of the signals, of dimension 1.
>
> **noiseThres** : float
>
>> Value for low censored data. Default is None.
>
> **saturationThres** : float
>
>> Value for high censored data. Default is None
>
> **resDistFact** : `openturns.DistributionFactory`
>
>> Distribution hypothesis followed by the residuals. Default is `openturns.NormalFactory`.
>
> **boxCox** : bool or float
>
>> Enable or not the Box Cox transformation. If boxCox is a float, the Box Cox transformation is enabled with the given value. Default is False.

Notes

This method automatically :

- computes the Box Cox parameter if *boxCox* is True,

- computes the transformed signals if *boxCox* is True or a float,

- builds the univariate linear regression model on the data,

- computes the linear regression parameters for censored data if needed,

- computes the residuals,

- runs all hypothesis tests.

Methods

drawBoxCoxLikelihood([name])	Draw the loglikelihood versus the Box Cox parameter.
drawLinearModel([model, name])	Draw the linear regression prediction versus the true data.
drawResiduals([model, name])	Draw the residuals versus the defect values.
drawResidualsDistribution([model, name])	Draw the residuals histogram with the fitted distribution.
drawResidualsQQplot([model, name])	Draw the residuals QQ plot with the fitted distribution.
getAndersonDarlingPValue()	Accessor to the Anderson Darling test p-value.
getBoxCoxParameter()	Accessor to the Box Cox parameter.
getBreuschPaganPValue()	Accessor to the Breusch Pagan test p-value.
getCramerVonMisesPValue()	Accessor to the Cramer Von Mises test p-value.
getDurbinWatsonPValue()	Accessor to the Durbin Watson test p-value.
getHarrisonMcCabePValue()	Accessor to the Harrison McCabe test p-value.
getInputSample()	Accessor to the input sample.
getIntercept()	Accessor to the intercept of the linear regression model.
getKolmogorovPValue()	Accessor to the Kolmogorov test p-value.
getNoiseThreshold()	Accessor to the noise threshold.
getOutputSample()	Accessor to the output sample.
getR2()	Accessor to the R2 value.
getResiduals()	Accessor to the residuals.
getResidualsDistribution()	Accessor to the residuals distribution.
getSaturationThreshold()	Accessor to the saturation threshold.
getSlope()	Accessor to the slope of the linear regression model.
getStandardError()	Accessor to the standard error of the estimate.
getZeroMeanPValue()	Accessor to the Zero Mean test p-value.
printResults()	Print results of the linear analysis in the terminal.
saveResults(name)	Save all analysis test results in a file.

drawBoxCoxLikelihood(*name=None*)

Draw the loglikelihood versus the Box Cox parameter.

Parameters name : string

name of the figure to be saved with *transparent* option sets to True and *bbox_inches='tight'*. It can be only the file name or the full path name. Default is None.

Returns fig : matplotlib.figure

Matplotlib figure object.

> **ax** : matplotlib.axes
>
> > Matplotlib axes object.

> #### Notes

> This method is available only when the parameter *boxCox* is set to True.

drawLinearModel(*model='uncensored'*, *name=None*)

> Draw the linear regression prediction versus the true data.

> > **Parameters model** : string
> >
> > > The linear regression model to be used, either *uncensored* or *censored* if censored threshold were given. Default is *uncensored*.
> >
> > **name** : string
> >
> > > name of the figure to be saved with *transparent* option sets to True and *bbox_inches='tight'*. It can be only the file name or the full path name. Default is None.
> >
> > **Returns fig** : matplotlib.figure
> >
> > > Matplotlib figure object.
> >
> > **ax** : matplotlib.axes
> >
> > > Matplotlib axes object.

drawResiduals(*model='uncensored'*, *name=None*)

> Draw the residuals versus the defect values.

> > **Parameters model** : string
> >
> > > The residuals to be used, either *uncensored* or *censored* if censored threshold were given. Default is *uncensored*.
> >
> > **name** : string
> >
> > > name of the figure to be saved with *transparent* option sets to True and *bbox_inches='tight'*. It can be only the file name or the full path name. Default is None.
> >
> > **Returns fig** : matplotlib.figure
> >
> > > Matplotlib figure object.
> >
> > **ax** : matplotlib.axes
> >
> > > Matplotlib axes object.

drawResidualsDistribution(*model='uncensored'*, *name=None*)

> Draw the residuals histogram with the fitted distribution.

> > **Parameters model** : string
> >
> > > The residuals to be used, either *uncensored* or *censored* if censored threshold were given. Default is *uncensored*.
> >
> > **name** : string
> >
> > > name of the figure to be saved with *transparent* option sets to True and *bbox_inches='tight'*. It can be only the file name or the full path name. Default is None.

> > **Returns fig** : matplotlib.figure
> >
> > > Matplotlib figure object.
> >
> > > **ax** : matplotlib.axes
> >
> > > Matplotlib axes object.

drawResidualsQQplot(*model='uncensored'*, *name=None*)
> Draw the residuals QQ plot with the fitted distribution.

> > **Parameters model** : string
> >
> > > The residuals to be used, either *uncensored* or *censored* if censored threshold were given. Default is *uncensored*.
> >
> > > **name** : string
> >
> > > name of the figure to be saved with *transparent* option sets to True and *bbox_inches='tight'*. It can be only the file name or the full path name. Default is None.
> >
> > **Returns fig** : matplotlib.figure
> >
> > > Matplotlib figure object.
> >
> > > **ax** : matplotlib.axes
> >
> > > Matplotlib axes object.

getAndersonDarlingPValue()
> Accessor to the Anderson Darling test p-value.

> > **Returns pValue** : `openturns.NumericalPoint`
> >
> > > Either the p-value for the uncensored case or for both cases.

getBoxCoxParameter()
> Accessor to the Box Cox parameter.

> > **Returns lambdaBoxCox** : float
> >
> > > The Box Cox parameter used to transform the data. If the transformation is not enabled None is returned.

getBreuschPaganPValue()
> Accessor to the Breusch Pagan test p-value.

> > **Returns pValue** : `openturns.NumericalPoint`
> >
> > > Either the p-value for the uncensored case or for both cases.

getCramerVonMisesPValue()
> Accessor to the Cramer Von Mises test p-value.

> > **Returns pValue** : `openturns.NumericalPoint`
> >
> > > Either the p-value for the uncensored case or for both cases.

getDurbinWatsonPValue()
> Accessor to the Durbin Watson test p-value.

> > **Returns pValue** : `openturns.NumericalPoint`
> >
> > > Either the p-value for the uncensored case or for both cases.

getHarrisonMcCabePValue()
> Accessor to the Harrison McCabe test p-value.

> > **Returns pValue** : `openturns.NumericalPoint`
> >
> > Either the p-value for the uncensored case or for both cases.

getInputSample()
Accessor to the input sample.

> **Returns defects** : `openturns.NumericalSample`
>
> The input sample which is the defect values.

getIntercept()
Accessor to the intercept of the linear regression model.

> **Returns intercept** : `openturns.NumericalPoint`
>
> The intercept parameter for the uncensored and censored (if so) linear regression model.

getKolmogorovPValue()
Accessor to the Kolmogorov test p-value.

> **Returns pValue** : `openturns.NumericalPoint`
>
> Either the p-value for the uncensored case or for both cases.

getNoiseThreshold()
Accessor to the noise threshold.

> **Returns noiseThres** : float
>
> The noise threshold if it exists, if not it returns *None*.

getOutputSample()
Accessor to the output sample.

> **Returns signals** : `openturns.NumericalSample`
>
> The input sample which is the signal values.

getR2()
Accessor to the R2 value.

> **Returns R2** : `openturns.NumericalPoint`
>
> Either the R2 for the uncensored case or for both cases.

getResiduals()
Accessor to the residuals.

> **Returns residuals** : `openturns.NumericalSample`
>
> The residuals computed from the uncensored and censored linear regression model. The first column corresponds with the uncensored case.

getResidualsDistribution()
Accessor to the residuals distribution.

> **Returns distribution** : list of `openturns.Distribution`
>
> The fitted distribution on the residuals, computed in the uncensored and censored (if so) case.

getSaturationThreshold()
Accessor to the saturation threshold.

> **Returns saturationThres** : float
>
> The saturation threshold if it exists, if not it returns *None*.

---

getSlope()
> Accessor to the slope of the linear regression model.

> > **Returns** **slope** : `openturns.NumericalPoint`

> > > The slope parameter for the uncensored and censored (if so) linear regression model.

getStandardError()
> Accessor to the standard error of the estimate.

> > **Returns** **stderr** : `openturns.NumericalPoint`

> > > The standard error of the estimate for the uncensored and censored (if so) linear regression model.

getZeroMeanPValue()
> Accessor to the Zero Mean test p-value.

> > **Returns** **pValue** : `openturns.NumericalPoint`

> > > Either the p-value for the uncensored case or for both cases.

printResults()
> Print results of the linear analysis in the terminal.

saveResults(*name*)
> Save all analysis test results in a file.

> > **Parameters** **name** : string

> > > Name of the file or full path name.

> > ### Notes

> > The file can be saved as a csv file. Separations are made with tabulations.

> > If *name* is the file name, then it is saved in the current working directory.

1.1.2 POD model

[*UnivariateLinearModelPOD*](#)	Linear regression based POD.

UnivariateLinearModelPOD

class **UnivariateLinearModelPOD**(**args*)
> Linear regression based POD.

> **Available constructors:**

> UnivariateLinearModelPOD(*analysis=analysis, detection=detection*)

> UnivariateLinearModelPOD(*inputSample, outputSample, detection, noiseThres, saturationThres, resDistFact, boxCox*)

> > **Parameters** **analysis** : [*UnivariateLinearModelAnalysis*](#)

> > > Linear analysis object.

> > **inputSample** : 2-d sequence of float

> > > Vector of the defect sizes, of dimension 1.

> > **outputSample** : 2-d sequence of float
> >
> > > Vector of the signals, of dimension 1.
> >
> > **detection** : float
> >
> > > Detection value of the signal.
> >
> > **noiseThres** : float
> >
> > > Value for low censored data. Default is None.
> >
> > **saturationThres** : float
> >
> > > Value for high censored data. Default is None
> >
> > **resDistFact** : `openturns.DistributionFactory`
> >
> > > Distribution hypothesis followed by the residuals. Default is None.
> >
> > **boxCox** : bool or float
> >
> > > Enable or not the Box Cox transformation. If boxCox is a float, the Box Cox transformation is enabled with the given value. Default is False.

Notes

This class aims at building the POD based on a linear regression model. If a linear analysis has been launched, it can be used as prescribed in the first constructor. Otherwise, parameters must be given as in the second constructor.

Following the given distribution in *resDistFact*, the POD model is built different hypothesis:

- •if *resDistFact* = *None*, it corresponds with Berens-Binomial. This is the default case.

- •if *resDistFact* = `openturns.NormalFactory`, it corresponds with Berens-Gauss.

- •if *resDistFact* = {`openturns.KernelSmoothing`, `openturns.WeibullFactory`, ...}, the confidence interval is built by bootstrap.

Methods

getPODCLModel([model, confLevel])	Accessor to the POD model at a given confidence level.
getPODModel([model])	Accessor to the POD model.
getSimulationSize()	Accessor to the simulation size.
run()	Bla bla bla
setSimulationSize(size)	Accessor to the simulation size

getPODCLModel (*model='uncensored'*, *confLevel=0.95*)
> Accessor to the POD model at a given confidence level.
>
> > **Parameters model** : string
> >
> > > The linear regression model to be used, either *uncensored* or *censored* if censored threshold were given. Default is *uncensored*.
> >
> > **confLevel** : float
> >
> > > The confidence level the POD must be computed. Default is 0.95
> >
> > **Returns PODModelCl** : `openturns.NumericalMathFunction`

The function which computes the probability of detection for a given defect value at the confidence level given as parameter.

getPODModel(*model='uncensored'*)
    Accessor to the POD model.

        Parameters model : string

            The linear regression model to be used, either *uncensored* or *censored* if censored threshold were given. Default is *uncensored*.

        Returns PODModel : `openturns.NumericalMathFunction`

            The function which computes the probability of detection for a given defect value.

getSimulationSize()
    Accessor to the simulation size.

run()
    Bla bla bla

setSimulationSize(*size*)
    Accessor to the simulation size

        Parameters size : int

            The size of the simulation used to compute the confidence interval.

1.2 Examples

INDICES AND TABLES

- genindex
- modindex
- search