PhiK Documentation

KPMG Advanced Analytics
Big Data team

May 17, 2020

Contents

- Version: 0.9.11. Released: April 2020
- Documentation: https://phik.readthedocs.io
- Repository: https://github.com/kaveio/phik
- Publication: https://arxiv.org/abs/1811.11440

Phi_K is a new and practical correlation coefficient based on several refinements to Pearson's hypothesis test of independence of two variables.

The combined features of Phi_K form an advantage over existing coefficients. First, it works consistently between categorical, ordinal and interval variables. Second, it captures non-linear dependency. Third, it reverts to the Pearson correlation coefficient in case of a bi-variate normal input distribution. These are useful features when studying the correlation matrix of variables with mixed types.

The presented algorithms are easy to use and available through this public Python library: the correlation analyzer package. Emphasis is paid to the proper evaluation of statistical significance of correlations and to the interpretation of variable relationships in a contingency table, in particular in case of low statistics samples.

For example, the Phi_K correlation analyzer package has been used to study surveys, insurance claims, correlograms, etc. For details on the methodology behind the calculations, please see our publication.

CHAPTER 1

Documentation

The entire Phi_K documentation including tutorials can be found at read-the-docs. See the tutorials for detailed examples on how to run the code with pandas. We also have one example on how calculate the Phi_K correlation matrix for a spark dataframe.

CHAPTER 2

Check it out

The Phi_K library requires Python 3 and is pip friendly. To get started, simply do:

```
$ pip install phik
```

or check out the code from out GitHub repository:

```
$ git clone https://github.com/KaveIO/PhiK.git
$ pip install -e PhiK/
```

where in this example the code is installed in edit mode (option -e).

You can now use the package in Python with:

```python
import phik
```

Congratulations, you are now ready to use the PhiK correlation analyzer library!

CHAPTER 3

Quick run

As a quick example, you can do:

```python
import pandas as pd
import phik
from phik import resources, report

# open fake car insurance data
df = pd.read_csv( resources.fixture('fake_insurance_data.csv.gz') )
df.head()

# Pearson's correlation matrix between numeric variables (pandas functionality)
df.corr()

# get the phi_k correlation matrix between all variables
df.phik_matrix()

# get global correlations based on phi_k correlation matrix
df.global_phik()

# get the significance matrix (expressed as one-sided Z)
# of the hypothesis test of each variable-pair dependency
df.significance_matrix()

# contingency table of two columns
cols = ['mileage','car_size']
df[cols].hist2d()

# normalized residuals of contingency test applied to cols
df[cols].outlier_significance_matrix()

# show the normalized residuals of each variable-pair
df.outlier_significance_matrices()

# generate a phik correlation report and save as test.pdf
report.correlation_report(df, pdf_file_name='test.pdf')
```

For all available examples, please see the tutorials at read-the-docs.

CHAPTER 4

Contact and support

- Issues & Ideas: https://github.com/kaveio/phik/issues
- Q&A Support: contact us at: kave [at] kpmg [dot] com

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

CHAPTER 5

Contents

5.1 Why did we build this?

When exploring a data set, for example to model one variable in terms of the others, it is useful to summarize the dependencies between the variables, assess their significances, and visualize the individual variable dependencies. The `PhiK` correlation analyzer library contains several useful functions to help one do so.

- This library implements a novel correlation coefficient, ϕ_K, with properties that - taken together - form an advantage over existing methods.

  The calculation of correlation coefficients between paired data variables is a standard tool of analysis for every data analyst. Pearson's correlation coefficient is a de facto standard in most fields, but by construction only works for interval variables (sometimes called continuous variables). Pearson is unsuitable for data sets with mixed variable types, e.g. where some variables are ordinal or categorical.

  While many correlation coefficients exist, each with different features, we have not been able to find a correlation coefficient with Pearson-like characteristics and a sound statistical interpretation that works for interval, ordinal and categorical variable types alike.

  The correlation coefficient ϕ_K follows a uniform treatment for interval, ordinal and categorical variables, captures non-linear dependencies, and is similar to Pearson's correlation coefficient in case of a bivariate normal input distribution.

- We found that, by default, popular analysis libraries such `R` and `scipy` make incorrect ("asymptotic") assumptions when assessing the statistical significance of the χ^2 contingency test of variable independence. In particular, the actual number of degrees of freedom and the shape of the test statistic distribution can differ significantly from their theoretical predictions in case of low to medium statistics data samples. This leads to incorrect p-values for the hypothesis test of variable independence. A prescription has been implemented to fix these two mistakes.

- Visualizing the dependency between variables can be tricky, especially when dealing with (unordered) categorical variables. To help interpret any variable relationship found, we provide a method for the detection of significant excesses or deficits of records with respect to the expected values in a contingency table, so-called outliers, using a statistically independent evaluation for expected frequency of records, accouncting for the uncertainty on the expectation. We evaluate the significance of each outlier frequency in a table, and normalize and visualize these accordingly. The resulting plots we find to be very valuable to help interpret variable dependencies, and work alike for interval, ordinal and categorical variables.

The `PhiK` analysis library is particularly useful in modern-day analysis when studying the dependencies between a set of variables with mixed types, where often some variables are categorical. The package has been used by us to study surveys, insurance claims, correlograms, etc.

For details on the methodology behind the calculations, please see our publication. For the available examples on how to use the methods, please see the tutorials section.

5.2 Tutorials

This section contains materials on how to use the Phi_K correlation analysis code. There are additional side notes on how certain aspects work and where to find parts of the code. For more in depth explanations on the functionality of the code-base, try the API docs.

The tutorials are available in the `python/phik/notebooks` directory. We have:

- A basic tutorial: this covers the basics of calculating Phi_K, the statistical significance, and interpreting the correlation.
- An advanced tutorial: this shows how to use the advanced features of the `PhiK` library.
- A spark tutorial: this shows how to calculate the Phi_K correlation matrix for a spark dataframe.

You can open these notebooks directly:

- Run them interactively at MyBinder.
- View them statically: basic tutorial and the advanced tutorial and the spark tutorial.

5.3 Publication & Talks

5.3.1 Publication

- arXiv: https://arxiv.org/abs/1811.11440

5.3.2 Talks

- Coming soon.

5.3.3 References

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

5.4 Developing and Contributing

5.4.1 Working on the package

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

First clone the package.

```
git clone https://github.com/KaveIO/PhiK.git
```

then

```
pip install -e PhiK/
```

this will install `PhiK` in editable mode, which will allow you to edit the code and run it as you would with a normal installation of the `PhiK` correlation analyzer package.

To make sure that everything works try executing the tests, e.g.

```
cd PhiK/
phik_trial .
```

or

```
cd PhiK/
python setup.py test
```

That's it.

5.4.2 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.

5.4.3 Tips and Tricks

- Enable auto reload in `jupyter`:

```
%load_ext autoreload
```

this will reload modules before executing any user code.

5.5 API

5.5.1 API Documentation

PhiK

phik package

Subpackages

phik.decorators package

Submodules

phik.decorators.pandas module

Project: PhiK - correlation analyzer library

Module: phik.decorators.pandas

Created: 2018/11/14

Description: Decorators for pandas DataFrame objects

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Module contents

Submodules

phik.betainc module

Project: PhiK - correlation analyzer library

Created: 2018/09/05

Description: Implementation of incomplete beta function

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

`phik.betainc.`**`contfractbeta`**`(`*a: float*, *b: float*, *x: float*, *ITMAX: int = 5000*, *EPS: float = 1e-07*`)` → float

Continued fraction form of the incomplete Beta function.

Code translated from: Numerical Recipes in C.

Example kindly taken from blog: https://malishoaib.wordpress.com/2014/04/15/the-beautiful-beta-functions-in-raw-python/

Parameters

- **a** (*float*) – a
- **b** (*float*) – b
- **x** (*float*) – x
- **ITMAX** (*int*) – max number of iterations, default is 5000.
- **EPS** (*float*) – epsilon precision parameter, default is 1e-7.

Returns continued fraction form

Return type float

`phik.betainc.`**`incompbeta`**(*a: float*, *b: float*, *x: float*) → float

Evaluation of incomplete beta function.

Code translated from: Numerical Recipes in C.

Here a, b > 0 and 0 <= x <= 1. This function requires contfractbeta(a,b,x, ITMAX = 200)

Example kindly taken from blog: https://malishoaib.wordpress.com/2014/04/15/the-beautiful-beta-functions-in-raw-python/

Parameters

- **a** (*float*) – a
- **b** (*float*) – b
- **x** (*float*) – x

Returns incomplete beta function

Return type float

`phik.betainc.`**`log_incompbeta`**(*a: float*, *b: float*, *x: float*) → float

Evaluation of logarithm of incomplete beta function

Logarithm of incomplete beta function is implemented to ensure sufficient precision for values very close to zero and one.

Code translated from: Numerical Recipes in C.

Here a, b > 0 and 0 <= x <= 1. This function requires contfractbeta(a,b,x, ITMAX = 200)

Example kindly taken from blog: https://malishoaib.wordpress.com/2014/04/15/the-beautiful-beta-functions-in-raw-python/

Parameters

- **a** (*float*) – a
- **b** (*float*) – b
- **x** (*float*) – x

Returns tuple of log(incb) and log(1-incb)

Return type tuple

phik.binning module

Project: PhiK - correlation analyzer library

Created: 2018/09/06

Description: A set of rebinning functions, to help rebin two lists into a 2d histogram.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

`phik.binning.`**`bin_array`**(*arr*, *bin_edges*)

Index the data given the bin_edges.

Underflow and overflow values are indicated.

Parameters

- **`arr`** – array like object with input data
- **`bin_edges`** – list with bin edges.

Returns indexed data

`phik.binning.`**`bin_data`**(*data*, *cols: list = []*, *bins=10*, *quantile: bool = False*, *retbins: bool = False*)

Index the input dataframe given the bin_edges for the columns specified in cols.

Parameters

- **`data`** (*`DataFrame`*) – input data
- **`cols`** (*`list`*) – list of columns with numeric data which needs to be indexed
- **`bins`** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **`quantile`** – when bins is an integer, uniform bins (False) or bins based on quantiles (True)

Returns rebinned dataframe

Return type pandas.DataFrame

`phik.binning.`**`bin_edges`**(*arr*, *nbins: int*, *quantile: bool = False*) → numpy.ndarray

Create uniform or quantile bin-edges for the input array.

Parameters

- **`arr`** – array like object with input data
- **`nbins`** (*`int`*) – the number of bin
- **`quantile`** (*`bool`*) – uniform bins (False) or bins based on quantiles (True)

Returns array with bin edges

`phik.binning.`**`create_correlation_overview_table`**(*vals: dict*) → pandas.core.frame.DataFrame

Create overview table of phik/significance data.

Parameters **`vals`** (*`dict`*) – dictionary holding data for each variable pair formatted as {'var1:var2' : value}

Returns symmetric table with phik/significances of all variable pairs

Return type pandas.DataFrame

`phik.binning.`**`hist2d`**(*df*, *interval_cols=None*, *bins=10*, *quantile: bool = False*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*, *retbins: bool = False*) → pandas.core.frame.DataFrame

Give binned 2d dataframe of two colums of input dataframe

Parameters

- **`df`** – input data. Dataframe must contain exactly two columns
- **`interval_cols`** – columns with interval variables which need to be binned

- **bins** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **quantile** (*bool*) – when the number of bins is specified, use uniform binning (False) or quantile binning (True)
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns histogram dataframe

`phik.binning.`**`hist2d_from_rebinned_df`**(*data_binned: pandas.core.frame.DataFrame, dropna: bool = True, drop_underflow: bool = True, drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Give binned 2d dataframe of two colums of rebinned input dataframe

Parameters

- **df** – input data. Dataframe must contain exactly two columns
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns histogram dataframe

phik.bivariate module

Project: PhiK - correlation analyzer library

Created: 2019/11/23

Description: Convert Pearson correlation value into a chi2 value of a contingency test matrix of a bivariate gaussion, and vice-versa. Calculation uses scipy's mvn library.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.bivariate.`**`bivariate_normal_theory`**(*rho: float, nx: int = -1, ny: int = -1, n: int = 1, sx: numpy.ndarray = None, sy: numpy.ndarray = None*) → numpy.ndarray

Return binned pdf of bivariate normal distribution.

This function returns a "perfect" binned bivariate normal distribution.

Parameters

- **rho** (*float*) – tilt parameter
- **nx** (*int*) – number of uniform bins on x-axis. alternative to sx.

- **ny** (*int*) – number of uniform bins on y-axis. alternative to sy.
- **sx** (*np.ndarray*) – bin edges array of x-axis. default is None.
- **sy** (*np.ndarray*) – bin edges array of y-axis. default is None.
- **n** (*int*) – number of entries. default is one.

Returns np.ndarray of binned bivariate normal pdf

`phik.bivariate.`**`chi2_from_phik`**(*rho: float*, *n: int*, *subtract_from_chi2: float = 0*, *corr0: list = None*, *scale: float = None*, *sx: numpy.ndarray = None*, *sy: numpy.ndarray = None*, *pedestal: float = 0*, *nx: int = -1*, *ny: int = -1*) → float

Calculate chi2-value of bivariate gauss having correlation value rho

Calculate no-noise chi2 value of bivar gauss with correlation rho, with respect to bivariate gauss without any correlation.

Parameters

- **rho** (*float*) – tilt parameter
- **n** (*int*) – number of records
- **subtract_from_chi2** (*float*) – value subtracted from chi2 calculation. default is 0.
- **corr0** (*list*) – mvn_array result for rho=0. Default is None.
- **scale** (*float*) – scale is multiplied with the chi2 if set.
- **sx** (*np.ndarray*) – bin edges array of x-axis. default is None.
- **sy** (*np.ndarray*) – bin edges array of y-axis. default is None.
- **pedestal** (*float*) – pedestal is added to the chi2 if set.
- **nx** (*int*) – number of uniform bins on x-axis. alternative to sx.
- **ny** (*int*) – number of uniform bins on y-axis. alternative to sy.

Returns float chi2 value

`phik.bivariate.`**`phik_from_chi2`**(*chi2: float*, *n: int*, *nx: int*, *ny: int*, *sx: numpy.ndarray = None*, *sy: numpy.ndarray = None*, *pedestal: float = 0*) → float

Correlation coefficient of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **chi2** (*float*) – input chi2 value
- **n** (*int*) – number of records
- **nx** (*int*) – number of uniform bins on x-axis. alternative to sx.
- **ny** (*int*) – number of uniform bins on y-axis. alternative to sy.
- **sx** (*np.ndarray*) – bin edges array of x-axis. default is None.
- **sy** (*np.ndarray*) – bin edges array of y-axis. default is None.
- **pedestal** (*float*) – pedestal is added to the chi2 if set.

Returns float correlation coefficient

phik.data_quality module

Project: PhiK - correlation analyzer library

Created: 2018/12/28

Description: A set of functions to check for data quality issues in input data.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.data_quality.`**`dq_check_hist2d`**(*hist2d*)

Basic data quality checks for a contingency table

The Following checks are done:

1. There must be at least two bins in both the x and y direction.
2. If the number of bins in the x and/or y direction is larger than 100 a warning is printed.

Parameters **`hist2d`** – contigency table

Returns bool passed_check

`phik.data_quality.`**`dq_check_nunique_values`**(*df*, *interval_cols*, *dropna=True*)

Basic data quality checks per column in a dataframe.

The following checks are done:

1. For all non-interval variables, if the number of unique values per variable is larger than 100 a warning is printed. When the number of unique values is large, the variable is likely to be an interval variable. Calculation of phik will be slow(ish) for pairs of variables where one (or two) have many different values (i.e. many bins).

2. For all interval variables, the number of unique values must be at least two. If the number of unique values is zero (i.e. all NaN) the column is removed. If the number of unique values is one, it is not possible to automatically create a binning for this variable (as min and max are the same). The variable is therefore dropped, irrespective of whether dropna is True or False.

3. For all non-interval variables, the number of unique values must be at least either a) 1 if dropna=False (NaN is now also considered a valid category), or b) 2 if dropna=True

The function returns a dataframe where all columns with invalid data are removed. Also the list of interval_cols is updated and returned.

Parameters

- **`df`** (*`pd.DataFrame`*) – input data
- **`interval_cols`** (*`list`*) – column names of columns with interval variables.
- **`dropna`** (*`bool`*) – remove NaN values when True

Returns cleaned data, updated list of interval columns

phik.definitions module

Project: PhiK - correlation analyzer library

Created: 2018/09/05

Description: Definitions used throughout the phik package

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

phik.entry_points module

Project: PhiK - correlation analyzer library

Created: 2018/11/13

Description: Collection of phik entry points

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.entry_points.`**`phik_trial`**`()`

Run Phi_K tests.

We will keep this here until we've completed switch to pytest or nose and tox. We could also keep it, but I don't like the fact that packages etc. are hard coded. Gotta come up with a better solution.

phik.outliers module

Project: PhiK - correlation analyzer library

Created: 2018/09/05

Description: Functions for calculating the statistical significance of outliers in a contingency table.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.outliers.`**`get_exact_poisson_uncertainty`**(*x: float*, *nsigmas: float = 1*) → float

Calculate the uncerainty on x using an exact poisson confidence interval.

Calculate the uncerainty on x using an exact poisson confidence interval. The width of the confidence interval can be specified using the number of sigmas. The default number of sigmas is set to 1, resulting in an error that is approximated by the standard poisson error sqrt(x).

Exact poisson uncertainty is described here: https://ms.mcmaster.ca/peter/s743/poissonalpha.html https://www.statsdirect.com/help/rates/poisson_rate_ci.htm https://www.ncbi.nlm.nih.gov/pubmed/2296988

Parameters **x** (*float*) – value

Return xerr the uncertainty on x (1 sigma)

Return type float

`phik.outliers.`**`get_independent_frequency_estimates`**(*values: numpy.ndarray*, *CI_method: str = 'poisson'*) → numpy.ndarray

Calculation of expected frequencies, based on the ABCD-method, i.e. independent frequency estimates.

Parameters

- **values** – The contingency table. The table contains the observed number of occurrences in each category.

- **CI_method** (*string*) – method to be used for uncertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval

Returns exp, experr expected frequencies, error on the expected frequencies

phik.outliers.**get_outlier_significances**(*obs: numpy.ndarray*, *exp: numpy.ndarray*, *experr: numpy.ndarray*) → numpy.ndarray

Evaluation of significance of observation

Evaluation of the significance of the difference between the observed number of occurences and the expected number of occurences, taking into account the uncertainty on the expectednumber of occurences. When the uncertainty is not zero, the Linnemann method is used to calculate the pvalues.

Parameters

- **obs** – observed numbers
- **exp** – expected numbers
- **experr** – uncertainty on the expected numbers

Returns pvalues, zvalues

phik.outliers.**get_poisson_uncertainty**(*x: float*) → float

Calculate the uncerainty on x using standard poisson error. In case x=0 the error=1 is assigned.

Parameters **x** (*float*) – value

Return xerr the uncertainty on x (1 sigma)

Return type float

phik.outliers.**get_uncertainty**(*x: float*, *CI_method: str = 'poisson'*) → float

Calculate the uncertainty on a random number x taken from the poisson distribution

The uncertainty on the x is calculated using either the standard poisson error (poisson) or using the asymmetric exact poisson interval (exact_poisson). https://www.ncbi.nlm.nih.gov/pubmed/2296988 #FIXME: check ref

Parameters

- **x** (*float*) – value
- **CI_method** (*string*) – method to be used for uncertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval

Return xerr the uncertainty on x (1 sigma)

phik.outliers.**log_poisson_obs_mid_p**(*nobs: int*, *nexp: float*, *nexperr: float*) → Tuple[float, float]

Calculate the logarithm of the p-value for measuring nobs observations given the expected value.

The Lancaster mid-P correction is applied to take into account the effects of discrete statistics. If the uncertainty on the expected value is known the Linnemann method is used for the p-value calcuation. Otherwise the Poisson distribution is used to estimate the p-value.

Parameters

- **nobs** (*int*) – observed count
- **nexp** (*float*) – expected number
- **nexperr** (*float*) – uncertainty on the expected number

Returns tuple of log(p) and log(1-p)

Return type tuple

`phik.outliers.`**`log_poisson_obs_p`**(*nobs: int*, *nexp: float*, *nexperr: float*) → Tuple[float, float]

Calculate logarithm of p-value for nobs observations given the expected value and its uncertainty using the Linnemann method.

If the uncertainty on the expected value is known the Linnemann method is used. Otherwise the Poisson distribution is used to estimate the p-value.

Measures of Significance in HEP and Astrophysics Authors: J. T. Linnemann http://arxiv.org/abs/physics/0312059

Code inspired by: https://root.cern.ch/doc/master/NumberCountingUtils_8cxx_source.html#l00086

Three fixes are added for:

- nobs = 0, when - by construction - p should be 1.
- uncertainty of zero, for which Linnemann's function does not work, but one can simply revert to regular Poisson.
- when nexp=0, betainc always returns 1. Here we set nexp = nexperr.

Parameters

- **`nobs`** (*`int`*) – observed count
- **`nexp`** (*`float`*) – expected number
- **`nexperr`** (*`float`*) – uncertainty on the expected number

Returns tuple containing pvalue and 1 - pvalue

Return type tuple

`phik.outliers.`**`outlier_significance_from_array`**(*x*, *y*, *num_vars: list = None*, *bins=10*, *quantile: bool = False*, *ndecimals: int = 1*, *CI_method: str = 'poisson'*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Calculate the significance matrix of excesses or deficits of input x and input y. x and y can contain interval, ordinal or categorical data. Use the num_vars variable to indicate whether x and/or y contain interval data.

Parameters

- **`x`** (*`list`*) – array-like input
- **`y`** (*`list`*) – array-like input
- **`num_vars`** (*`list`*) – list of variables which are numeric and need to be binned, either ['x'],['y'],or['x','y']
- **`bins`** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **`quantile`** (*`bool`*) – when the number of bins is specified, use uniform binning (False) or quantile binning (True)
- **`ndecimals`** – number of decimals to use in labels of binned interval variables to specify bin edges (default=1)
- **`CI_method`** (*`string`*) – method to be used for undertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval
- **`dropna`** (*`bool`*) – remove NaN values with True

- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns outlier significance matrix (pd.DataFrame)

phik.outliers.**outlier_significance_from_binned_array**(*x*, *y*, *CI_method: str = 'poisson'*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Calculate the significance matrix of excesses or deficits of input x and input y. x and y can contain binned interval, ordinal or categorical data.

Parameters

- **x** (*list*) – array-like input
- **y** (*list*) – array-like input
- **CI_method** (*string*) – method to be used for undertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns outlier significance matrix (pd.DataFrame)

phik.outliers.**outlier_significance_matrices**(*df: pandas.core.frame.DataFrame*, *interval_cols: list = None*, *CI_method: str = 'poisson'*, *ndecimals: int = 1*, *bins=10*, *quantile: bool = False*, *combinations: list = []*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*, *retbins: bool = False*)

Calculate the significance matrix of excesses or deficits for all possible combinations of variables, or for those combinations specified using combinations

Parameters

- **df** – input data
- **interval_cols** – columns with interval variables which need to be binned
- **CI_method** (*string*) – method to be used for undertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval
- **ndecimals** – number of decimals to use in labels of binned interval variables to specify bin edges (default=1)
- **bins** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **quantile** (*bool*) – when the number of bins is specified, use uniform binning (False) or quantile binning (True)

- **`combinations`** – in case you do not want to calculate an outlier significance matrix for all permutations of the available variables, you can specify a list of the required permutations here, in the format [(var1, var2), (var2, var4), etc]
- **`dropna`** (*`bool`*) – remove NaN values with True
- **`drop_underflow`** (*`bool`*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **`drop_overflow`** (*`bool`*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)
- **`retbins`** (*`bool`*) – if true, function also returns dict with bin_edges of rebinned variables.

Returns dictionary with outlier significance matrices (pd.DataFrame)

`phik.outliers.`**`outlier_significance_matrices_from_rebinned_df`**(*data_binned, binning_dict={}, CI_method='poisson', ndecimals=1, combinations=[], dropna=True, drop_underflow=True, drop_overflow=True*)

Calculate the significance matrix of excesses or deficits for all possible combinations of variables, or for those combinations specified using combinations. This functions could also be used instead of outlier_significance_matrices in case all variables are either categorical or ordinal, so no binning is required.

Parameters

- **`data_binned`** – input data. Interval variables need to be binned. Dataframe must contain exactly two columns
- **`binning_dict`** (*`dict`*) – dictionary with bin edges for each binned interval variable. When no bin_edges are provided values are used as bin label. Otherwise, bin labels are constructed based on provided bin edge information.
- **`CI_method`** (*`string`*) – method to be used for uncertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval
- **`bins`** – specify the binning, either by proving the number of bins, a list of bin edges, or a dictionary with bin specifications per variable. (default=10)
- **`ndecimals`** – number of decimals to use in labels of binned interval variables to specify bin edges (default=1)
- **`quantile`** (*`bool`*) – when the number of bins is specified, use uniform binning (False) or quantile binning (True)
- **`combinations`** – in case you do not want to calculate an outlier significance matrix for all permutations of the available variables, you can specify a list of the required permutations here, in the format [(var1, var2), (var2, var4), etc]
- **`dropna`** (*`bool`*) – remove NaN values with True
- **`drop_underflow`** (*`bool`*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **`drop_overflow`** (*`bool`*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns dictionary with outlier significance matrices (pd.DataFrame) phik.outliers.**outlier_significance_matrix**(*df: pandas.core.frame.DataFrame*, *interval_cols: list = None*, *CI_method: str = 'poisson'*, *ndecimals: int = 1*, *bins=10*, *quantile: bool = False*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*, *retbins: bool = False*)

Calculate the significance matrix of excesses or deficits

Parameters

- **df** – input data. Dataframe must contain exactly two columns
- **interval_cols** – columns with interval variables which need to be binned
- **CI_method** (*string*) – method to be used for undertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval
- **bins** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **ndecimals** – number of decimals to use in labels of binned interval variables to specify bin edges (default=1)
- **quantile** (*bool*) – when the number of bins is specified, use uniform binning (False) or quantile binning (True)
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)
- **retbins** (*bool*) – if true, function also returns dict with bin_edges of rebinned variables.

Returns outlier significance matrix (pd.DataFrame)

phik.outliers.**outlier_significance_matrix_from_hist2d**(*data: numpy.ndarray*, *CI_method: str = 'poisson'*) → numpy.ndarray

Calculate the significance matrix of excesses or deficits in a contingency table

Parameters

- **data** – numpy array contingency table
- **CI_method** (*string*) – method to be used for undertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval

Returns pvalue matrix, outlier significance matrix phik.outliers.**outlier_significance_matrix_from_rebinned_df**(*data_binned: pandas.core.frame.DataFrame*, *binning_dict: dict*, *CI_method: str = 'poisson'*, *ndecimals: int = 1*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Calculate the significance matrix of excesses or deficits

Parameters

- **data_binned** – input data. Dataframe must contain exactly two columns
- **binning_dict** (*dict*) – dictionary with bin edges for each binned interval variable. When no bin_edges are provided values are used as bin label. Otherwise, bin labels are constructed based on provided bin edge information.
- **CI_method** (*string*) – method to be used for undertainty calculation. poisson: normal poisson error. exact_poisson: error calculated from the asymmetric exact poisson interval
- **ndecimals** – number of decimals to use in labels of binned interval variables to specify bin edges (default=1)
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns outlier significance matrix (pd.DataFrame)

phik.outliers.**poisson_obs_mid_p**(*nobs: int*, *nexp: float*, *nexperr: float*) → float

Calculate the p-value for measuring nobs observations given the expected value.

The Lancaster mid-P correction is applied to take into account the effects of discrete statistics. If the uncertainty on the expected value is known the Linnemann method is used for the p-value calcuation. Otherwise the Poisson distribution is used to estimate the p-value.

Parameters

- **nobs** (*int*) – observed count
- **nexp** (*float*) – expected number
- **nexperr** (*float*) – uncertainty on the expected number

Returns mid p-value

Return type float

phik.outliers.**poisson_obs_mid_z**(*nobs: int*, *nexp: float*, *nexperr: float*) → float

Calculate the Z-value for measuring nobs observations given the expected value.

The Z-value express the number of sigmas the observed value diviates from the expected value, and is based on the p-value calculation. The Lancaster midP correction is applied to take into account the effects of low statistics. If the uncertainty on the expected value is known the Linnemann method is used for the p-value calcuation. Otherwise the Poisson distribution is used to estimate the p-value.

Parameters

- **nobs** (*int*) – observed count
- **nexp** (*float*) – expected number
- **nexperr** (*float*) – uncertainty on the expected number

Returns Z-value

Return type tuple

`phik.outliers.`**`poisson_obs_p`**(*nobs: int*, *nexp: float*, *nexperr: float*) → float

Calculate p-value for nobs observations given the expected value and its uncertainty using the Linnemann method.

If the uncertainty on the expected value is known the Linnemann method is used. Otherwise the Poisson distribution is used to estimate the p-value.

Measures of Significance in HEP and Astrophysics Authors: J. T. Linnemann http://arxiv.org/abs/physics/0312059

Code inspired by: https://root.cern.ch/doc/master/NumberCountingUtils_8cxx_source.html#l00086

Three fixes are added for:

- nobs = 0, when - by construction - p should be 1.
- uncertainty of zero, for which Linnemann's function does not work, but one can simply revert to regular Poisson.
- when nexp=0, betainc always returns 1. Here we set nexp = nexperr.

Parameters

- **nobs** (*int*) – observed count
- **nexp** (*float*) – expected number
- **nexperr** (*float*) – uncertainty on the expected number

Returns p-value

Return type float

`phik.outliers.`**`poisson_obs_z`**(*nobs: int*, *nexp: float*, *nexperr: float*) → float

Calculate the Z-value for measuring nobs observations given the expected value.

The Z-value express the number of sigmas the observed value diviates from the expected value, and is based on the p-value calculation. If the uncertainty on the expected value is known the Linnemann method is used. Otherwise the Poisson distribution is used to estimate the p-value.

Parameters

- **nobs** (*int*) – observed count
- **nexp** (*float*) – expected number
- **nexperr** (*float*) – uncertainty on the expected number

Returns Z-value

Return type float

phik.phik module

Project: PhiK - correlation analyzer library

Created: 2018/09/05

Description: Functions for the Phik correlation calculation

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.phik.`**`global_phik_array`**(*df: pandas.core.frame.DataFrame*, *interval_cols: list = None*, *bins=10*, *quantile: bool = False*, *noise_correction: bool = True*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Global correlation values of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **`data_binned`** (*`pd.DataFrame`*) – input data
- **`interval_cols`** (*`list`*) – column names of columns with interval variables.
- **`bins`** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **`quantile`** – when bins is an integer, uniform bins (False) or bins based on quantiles (True)
- **`noise_correction`** (*`bool`*) – apply noise correction in phik calculation
- **`dropna`** (*`bool`*) – remove NaN values with True
- **`drop_underflow`** (*`bool`*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **`drop_overflow`** (*`bool`*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns global correlations array

`phik.phik.`**`global_phik_from_rebinned_df`**(*data_binned: pandas.core.frame.DataFrame*, *noise_correction: bool = True*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Global correlation values of bivariate gaussian derived from chi2-value from rebinned df

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **`data_binned`** (*`pd.DataFrame`*) – rebinned input data
- **`noise_correction`** (*`bool`*) – apply noise correction in phik calculation
- **`dropna`** (*`bool`*) – remove NaN values with True

- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns global correlations array

`phik.phik.`**`phik_from_array`**(*x*, *y*, *num_vars: list = []*, *bins=10*, *quantile: bool = False*, *noise_correction: bool = True*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → float

Correlation matrix of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **x** – array-like input
- **y** – array-like input
- **num_vars** – list of numeric variables which need to be binned, e.g. ['x'] or ['x','y']
- **bins** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **quantile** – when bins is an integer, uniform bins (False) or bins based on quantiles (True)
- **noise_correction** (*bool*) – apply noise correction in phik calculation
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns phik correlation coefficient

`phik.phik.`**`phik_from_binned_array`**(*x*, *y*, *noise_correction: bool = True*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → float

Correlation matrix of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **x** – array-like input. Interval variables need to be binned beforehand.
- **y** – array-like input. Interval variables need to be binned beforehand.
- **noise_correction** (*bool*) – apply noise correction in phik calculation
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)

- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns phik correlation coefficient

`phik.phik.`**`phik_from_hist2d`**(*observed: numpy.ndarray*, *noise_correction: bool = True*) → float

correlation coefficient of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **observed** – 2d-array observed values
- **noise_correction** (*bool*) – apply noise correction in phik calculation

Returns float correlation coefficient phik

`phik.phik.`**`phik_from_rebinned_df`**(*data_binned: pandas.core.frame.DataFrame*, *noise_correction: bool = True*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Correlation matrix of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **data_binned** (*pd.DataFrame*) – input data where interval variables have been binned
- **noise_correction** (*bool*) – apply noise correction in phik calculation
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns phik correlation matrix

`phik.phik.`**`phik_matrix`**(*df: pandas.core.frame.DataFrame*, *interval_cols: list = None*, *bins=10*, *quantile: bool = False*, *noise_correction: bool = True*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Correlation matrix of bivariate gaussian derived from chi2-value

Chi2-value gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Bivariate gaussian's range is set to [-5,5] by construction.

Parameters

- **data_binned** (*pd.DataFrame*) – input data
- **interval_cols** (*list*) – column names of columns with interval variables.

- **bins** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **quantile** – when bins is an integer, uniform bins (False) or bins based on quantiles (True)
- **noise_correction** (*bool*) – apply noise correction in phik calculation
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns phik correlation matrix

phik.phik.**spark_phik_matrix_from_hist2d_dict**(*spark_context*, *hist_dict*)

Correlation matrix of bivariate gaussian using spark parallellization over variable-pair 2d histograms

See spark notebook phik_tutorial_spark.ipynb as example.

Each input histogram gets converted into correlation coefficient of bivariate gauss with correlation value rho, assuming giving binning and number of records. Correlation coefficient value is between 0 and 1.

Parameters

- **spark_context** – spark context
- **hist_dict** – dict of 2d numpy grids with value-counts. keys are histogram names.

Returns phik correlation matrix

phik.report module

Project: PhiK - correlation analyzer library

Created: 2018/09/06

Description: Functions to create nice correlation overview and matrix plots

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

phik.report.**correlation_report**(*data: pandas.core.frame.DataFrame*, *interval_cols: list = None*, *bins=10*, *quantile: bool = False*, *do_outliers: bool = True*, *pdf_file_name: str = ''*, *significance_threshold: float = 3*, *correlation_threshold: float = 0.5*, *noise_correction: bool = True*, *store_each_plot: bool = False*, *lambda_significance: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*, *nsim_chi2: int = 1000*, *significance_method: str = 'asymptotic'*, *CI_method: str = 'poisson'*) → Union[pandas.core.frame.DataFrame, dict]

Create a correlation report for the given dataset.

The following quantities are calculated:

- The phik correlation matrix
- The significance matrix

- The outlier significances measured in pairs of variables. (optional)

Parameters

- **`data`** – input dataframe
- **`interval_cols`** – list of columns names of columns containing interval data
- **`bins`** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **`quantile`** – when bins is an integer, uniform bins (False) or bins based on quantiles (True)
- **`do_outliers`** – Evaluate outlier significances of variable pairs (when True)
- **`pdf_file_name`** – file name of the pdf where the results are stored
- **`store_each_plot`** – store each plot in folder derived from pdf_file_name. If true, single pdf is no longer stored. Default is false.
- **`significance_threshold`** – evaluate outlier significance for all variable pairs with a significance of uncorrelation higher than this threshold
- **`correlation_threshold`** – evaluate outlier significance for all variable pairs with a phik correlation higher than this threshold
- **`noise_correction`** – Apply noise correction in phik calculation
- **`lambda_significance`** – test statistic used in significance calculation. Options: [pearson, log-likelihood]
- **`simulation_method`** – sampling method using in significance calculation. Options: [mutlinominal, row_product_multinominal, col_product_multinominal, hypergeometric]
- **`nsim_chi2`** – number of simulated datasets in significance calculation.
- **`significance_method`** – method for significance calculation. Options: [asymptotic, MC, hybrid]
- **`CI_method`** – method for uncertainty calculation for outlier significance calculation. Options: [poisson, exact_poisson]

Returns phik_matrix (pd.DataFrame), global_phik (np.array), significance_matrix (pd.DataFrame), outliers_overview (dictionary), output_files (dictionary)

`phik.report.`**`plot_correlation_matrix`**(*matrix_colors: numpy.ndarray*, *x_labels: list*, *y_labels: list*, *pdf_file_name: str = ”*, *title: str = 'correlation'*, *vmin: float = -1*, *vmax: float = 1*, *color_map: str = 'RdYlGn'*, *x_label: str = ”*, *y_label: str = ”*, *top: int = 20*, *matrix_numbers: numpy.ndarray = None*, *print_both_numbers: bool = True*, *figsize: tuple = (7, 5)*, *usetex: bool = False*, *identity_layout: bool = True*, *fontsize_factor: float = 1*) → None

Create and plot correlation matrix.

Copied with permission from the eskapade package (pip install eskapade)

Parameters

- **`matrix_colors`** – input correlation matrix
- **`x_labels`** (*`list`*) – Labels for histogram x-axis bins
- **`y_labels`** (*`list`*) – Labels for histogram y-axis bins

- **pdf_file_name** (*str*) – if set, will store the plot in a pdf file
- **title** (*str*) – if set, title of the plot
- **vmin** (*float*) – minimum value of color legend (default is -1)
- **vmax** (*float*) – maximum value of color legend (default is +1)
- **x_label** (*str*) – Label for histogram x-axis
- **y_label** (*str*) – Label for histogram y-axis
- **color_map** (*str*) – color map passed to matplotlib pcolormesh. (default is 'RdYlGn')
- **top** (*int*) – only print the top 20 characters of x-labels and y-labels. (default is 20)
- **matrix_numbers** – input matrix used for plotting numbers. (default it matrix_colors)
- **identity_layout** – Plot diagonal from right top to bottom left (True) or bottom left to top right (False)

phik.report.**plot_hist_and_func**(*data, func, funcparams, xbins=False, labels=['', ''], xlabel='', ylabel='', title='', xlimit=None, alpha=1*)

Create a histogram of the provided data and overlay with a function.

Parameters

- **data** (*list*) – data
- **func** (*function*) – function of the type f(x, a, b, c) where parameters a, b, c are optional
- **funcparams** (*list*) – parameter values to be given to the function, to be specified as [a, b, c]
- **xbins** – specify binning of histogram, either by giving the number of bins or a list of bin edges
- **labels** – labels of histogram and function to be used in the legend
- **xlabel** – figure xlabel
- **ylabel** – figure ylabel
- **title** – figure title
- **xlimit** – x limits figure
- **alpha** – alpha histogram

Returns

phik.resources module

Project: PhiK - correlation analyzer library

Created: 2018/11/13

Description: Collection of helper functions to get fixtures, i.e. for test data and notebooks. These are mostly used by the (integration) tests and example notebooks.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

phik.resources.**fixture**(*name: str*) → str

Return the full path filename of a fixture data set.

Parameters **`name`** (*`str`*) – The name of the fixture.

Returns The full path filename of the fixture data set.

Return type str

Raises **`FileNotFoundError`** – If the fixture cannot be found.

`phik.resources.`**`notebook`**(*name: str*) → str

Return the full path filename of a tutorial notebook.

Parameters **`name`** (*`str`*) – The name of the notebook.

Returns The full path filename of the notebook.

Return type str

Raises **`FileNotFoundError`** – If the notebook cannot be found.

phik.significance module

Project: PhiK - correlation analyzer library

Created: 2018/09/05

Description: Functions for doing the significance evaluation of an hypothesis test of variable independence using a contingency table.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.significance.`**`fit_test_statistic_distribution`**(*chi2s: list*, *nbins: int = 50*) → float

Fit the hybrid chi2-distribution to the data to find f.

Perform a binned likelihood fit to the data to find the optimal value for the fraction f in

h(x|f) = N * (f * chi2(x, ndof) + (1-f) * gauss(x, ndof, sqrt(ndof))

The parameter ndof is fixed in the fit using ndof = mean(x). The total number of datapoints N is also fixed.

Parameters

- **`chi2s`** (*`list`*) – input data - a list of chi2 values
- **`nbins`** (*`int`*) – in order to fit the data a histogram is created with nbins number of bins

Returns f, ndof, sigma (width of gauss), bw (bin width)

`phik.significance.`**`hfunc`**(*x: float*, *N: float*, *f: float*, *k: float*, *sigma: float*) → float

Definition of the combined probability density function h(x)

h(x|f) = N * (f * chi2(x, k) + (1-f) * gauss(x, k, sigma)

Parameters

- **`x`** (*`float`*) – x
- **`N`** (*`float`*) – normalisation
- **`f`** (*`float`*) – fraction [0,1]
- **`k`** (*`float`*) – ndof of chi2 function and mean of gauss
- **`sigma`** (*`float`*) – width of gauss

Returns h(x|f)

`phik.significance.`**`significance_from_array`**(*x*, *y*, *num_vars: list = []*, *bins=10*, *quantile: bool = False*, *lambda_: str = 'log-likelihood'*, *nsim: int = 1000*, *significance_method: str = 'hybrid'*, *simulation_method: str = 'multinominal'*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → float

Calculate the significance of correlation

Calculate the significance of correlation for two variables which can be of interval, oridnal or categorical type. Interval variables will be binned.

Parameters

- **x** – array-like input
- **y** – array-like input
- **num_vars** – list of numeric variables which need to be binned, e.g. ['x'] or ['x','y']
- **bins** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **quantile** – when bins is an integer, uniform bins (False) or bins based on quantiles (True)
- **lambda** (*str*) – test statistic. Available options are [pearson, log-likelihood]
- **nsim** (*int*) – number of simulated datasets
- **simulation_method** (*str*) – simulation method. Options: [mutlinominal, row_product_multinominal, col_product_multinominal, hypergeometric].
- **significance_method** (*str*) – significance_method. Options: [asymptotic, MC, hybrid]
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns p-value, significance

`phik.significance.`**`significance_from_binned_array`**(*x*, *y*, *lambda_: str = 'log-likelihood'*, *significance_method: str = 'hybrid'*, *nsim: int = 1000*, *simulation_method: str = 'multinominal'*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → float

Calculate the significance of correlation

Calculate the significance of correlation for two variables which can be of interval, oridnal or categorical type. Interval variables need to be binned.

Parameters

- **x** – array-like input
- **y** – array-like input

- **lambda** (*str*) – test statistic. Available options are [pearson, log-likelihood]
- **simulation_method** (*str*) – simulation method. Options: [multinominal, row_product_multinominal, col_product_multinominal, hypergeometric].
- **nsim** (*int*) – number of simulated datasets
- **significance_method** (*str*) – significance_method. Options: [asymptotic, MC, hybrid]
- **dropna** (*bool*) – remove NaN values with True
- **drop_underflow** (*bool*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **drop_overflow** (*bool*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns p-value, significance

`phik.significance.`**`significance_from_chi2_MC`**(*chi2: float*, *values: numpy.ndarray*, *nsim: int = 1000*, *lambda_: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*) → float

Convert a chi2 into significance using knowledge about the shape of the chi2 distribution of simulated data

Calculate significance based on simulation (MC method), using a simple percentile.

Parameters

- **chi2** (*float*) – chi2 value
- **chi2s** (*list*) – chi2s values

Returns pvalue, significance

`phik.significance.`**`significance_from_chi2_asymptotic`**(*values: numpy.ndarray*, *chi2: float*) → float

Convert a chi2 into significance using knowledge about the number of degrees of freedom

Convertions is done using asymptotic approximation.

Parameters

- **chi2** (*float*) – chi2 value
- **ndof** (*float*) – number of degrees of freedom

Returns pvalue, significance

`phik.significance.`**`significance_from_chi2_hybrid`**(*chi2: float*, *values: numpy.ndarray*, *nsim: int = 1000*, *lambda_: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*) → float

Convert a chi2 into significance using a hybrid method

This method combines the asymptotic method with the MC method, but applies several corrections:

- use effective number of degrees of freedom instead of number of degrees of freedom. The effective number of degrees of freedom is measured as mean(chi2s), with chi2s a list of simulated chi2 values.
- for low statistics data sets, with on average less than 4 data points per bin, the distribution of chi2-values is better described by h(x|f) then by the usual chi2-distribution. Use h(x|f) to convert the chi2 value to the pvalue and significance.

h(x|f) = N * (f * chi2(x, ndof) + (1-f) * gauss(x, ndof, sqrt(ndof))

Parameters

- **`chi2`** (*`float`*) – chi2 value
- **`chi2s`** (*`list`*) – chi2s values
- **`avg_per_bin`** (*`float`*) – average number of data points per bin

Returns pvalue, significance

`phik.significance.`**`significance_from_chi2_ndof`**(*chi2: float*, *ndof: float*) → float

Convert a chi2 into significance using knowledge about the number of degrees of freedom

Convertions is done using asymptotic approximation.

Parameters

- **`chi2`** (*`float`*) – chi2 value
- **`ndof`** (*`float`*) – number of degrees of freedom

Returns pvalue, significance

`phik.significance.`**`significance_from_hist2d`**(*values: numpy.ndarray*, *nsim: int = 1000*, *lambda_: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*, *significance_method: str = 'hybrid'*) → float

Calculate the significance of correlation of two variables based on the contingency table

Parameters

- **`values`** – contingency table
- **`nsim`** (*`int`*) – number of simulations
- **`lambda`** (*`str`*) – test statistic. Available options are [pearson, log-likelihood]
- **`simulation_method`** (*`str`*) – simulation method. Options: [mutlinominal, row_product_multinominal, col_product_multinominal, hypergeometric].
- **`significance_method`** (*`str`*) – significance_method. Options: [asymptotic, MC, hybrid]

Returns pvalue, significance

`phik.significance.`**`significance_from_rebinned_df`**(*data_binned: pandas.core.frame.DataFrame*, *lambda_: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*, *nsim: int = 1000*, *significance_method: str = 'hybrid'*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Calculate significance of correlation of all variable combinations in the dataframe

Parameters

- **`data_binned`** – input binned dataframe
- **`nsim`** (*`int`*) – number of simulations
- **`lambda`** (*`str`*) – test statistic. Available options are [pearson, log-likelihood]
- **`simulation_method`** (*`str`*) – simulation method. Options: [mutlinominal, row_product_multinominal, col_product_multinominal, hypergeometric].

- **`significance_method`** (*`str`*) – significance_method. Options: [asymptotic, MC, hybrid]
- **`dropna`** (*`bool`*) – remove NaN values with True
- **`drop_underflow`** (*`bool`*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **`drop_overflow`** (*`bool`*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns significance matrix

`phik.significance.`**`significance_matrix`**(*df: pandas.core.frame.DataFrame*, *interval_cols: list = None*, *lambda_: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*, *nsim: int = 1000*, *significance_method: str = 'hybrid'*, *bins=10*, *dropna: bool = True*, *drop_underflow: bool = True*, *drop_overflow: bool = True*) → pandas.core.frame.DataFrame

Calculate significance of correlation of all variable combinations in the dataframe

Parameters

- **`df`** (*`pd.DataFrame`*) – input data
- **`interval_cols`** (*`list`*) – column names of columns with interval variables.
- **`nsim`** (*`int`*) – number of simulations
- **`lambda`** (*`str`*) – test statistic. Available options are [pearson, log-likelihood]
- **`simulation_method`** (*`str`*) – simulation method. Options: [mutlinominal, row_product_multinominal, col_product_multinominal, hypergeometric].
- **`nsim`** – number of simulated datasets
- **`significance_method`** (*`str`*) – significance_method. Options: [asymptotic, MC, hybrid] :param bool dropna: remove NaN values with True
- **`bins`** – number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10) E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
- **`dropna`** (*`bool`*) – remove NaN values with True
- **`drop_underflow`** (*`bool`*) – do not take into account records in underflow bin when True (relevant when binning a numeric variable)
- **`drop_overflow`** (*`bool`*) – do not take into account records in overflow bin when True (relevant when binning a numeric variable)

Returns significance matrix

phik.simulation module

Project: PhiK - correlation analyzer library

Created: 2018/09/05

Description: Helper functions to simulate 2D datasets

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

Redistribution and use in source and binary forms, with or without modification, are permitted according to the terms listed in the file LICENSE.

`phik.simulation.`**`sim_2d_data`**

Simulate a 2 dimensional dataset given a 2 dimensional pdf

Parameters

- **`hist`** (*`array-like`*) – contingency table, which contains the observed number of occurrences in each category. This table is used as probability density function.
- **`ndata`** (*`int`*) – number of simulations

Returns simulated data

`phik.simulation.`**`sim_2d_data_patefield`**(*data: numpy.ndarray*) → numpy.ndarray

Simulate a two dimensional dataset with fixed row and column totals.

Simulation algorithm by Patefield: W. M. Patefield, Applied Statistics 30, 91 (1981) Python implementation inspired by (C version): https://people.sc.fsu.edu/~jburkardt/c_src/asa159/asa159.html

Parameters **`data`** – contingency table, which contains the observed number of occurrences in each category. This table is used as probability density function.

Returns simulated data

`phik.simulation.`**`sim_2d_product_multinominal`**(*data: numpy.ndarray*, *axis: str*) → numpy.ndarray

Simulate 2 dimensional data with either row or column totals fixed.

Parameters

- **`data`** – contingency table, which contains the observed number of occurrences in each category. This table is used as probability density function.
- **`axis`** – fix row totals (rows) or column totals (cols).

Returns simulated data

`phik.simulation.`**`sim_chi2_distribution`**(*values*, *nsim: int = 1000*, *lambda_: str = 'log-likelihood'*, *simulation_method: str = 'multinominal'*) → list

Simulate 2D data and calculate the chi-square statistic for each simulated dataset.

Parameters

- **`values`** – The contingency table. The table contains the observed number of occurrences in each category
- **`nsim`** (*`int`*) – number of simulations (optional, default=1000)
- **`simulation_method`** (*`str`*) – sampling method. Options: [multinominal, hypergeometric, row_product_multinominal, col_product_multinominal]
- **`lambda`** (*`str`*) – test statistic. Available options are [pearson, log-likelihood].

Returns chi2s list of chi2 values for each simulated dataset

`phik.simulation.`**`sim_data`**

Simulate a 2 dimenstional dataset given a 2 dimensional pdf

Several simulation methods are provided:

- multinominal: Only the total number of records is fixed.
- row_product_multinominal: The row totals fixed in the sampling.

- col_product_multinominal: The column totals fixed in the sampling.
- hypergeometric: Both the row or column totals are fixed in the sampling. Note that this type of sampling is only available when row and column totals are integers.

Parameters

- **data** – contingency table
- **method** (*str*) – sampling method. Options: [multinominal, hypergeometric, row_product_multinominal, col_product_multinominal]

Returns simulated data

phik.simulation.**simulate**

split off simulate function to allow for parallellization

phik.statistics module

Project: PhiK - correlation coefficient package

Created: 2018/09/05

Description: Statistics helper functions, for the calculation of phik and significance of a contingency table.

Authors: KPMG Advanced Analytics & Big Data team, Amstelveen, The Netherlands

phik.statistics.**estimate_ndof**(*chi2values: list*) → float

Estimation of the effective number of degrees of freedom.

A good approximation of endof is the average value. Alternatively a fit to the chi2 distribution can be make. Both values are returned.

Parameters **chi2values** (*list*) – list of chi2 values

Returns endof0, endof

phik.statistics.**estimate_simple_ndof**(*observed: numpy.ndarray*) → int

Simple estimation of the effective number of degrees of freedom.

This equals the nominal calculation for ndof minus the number of empty bins in the expected contingency table.

Parameters **observed** – numpy array of observed cell counts

Returns endof

phik.statistics.**get_chi2_using_dependent_frequency_estimates**(*vals: numpy.ndarray*, *lambda_: str = 'log-likelihood'*) → float

Chi-square test of independence of variables in a contingency table.

The expected frequencies are based on the marginal sums of the table, i.e. dependent frequency estimates.

Parameters **values** – The contingency table. The table contains the observed number of occurrences in each category

Returns chi2

`phik.statistics.`**`get_dependent_frequency_estimates`**(*vals: numpy.ndarray*) → numpy.ndarray

Calculation of dependent expected frequencies.

Calculation is based on the marginal sums of the table, i.e. dependent frequency estimates. :param values: The contingency table. The table contains the observed number of occurrences in each category

Returns exp expected frequencies

`phik.statistics.`**`theoretical_ndof`**(*observed: numpy.ndarray*) → int

Simple estimation of the effective number of degrees of freedom.

This equals the nominal calculation for ndof minus the number of empty bins in the expected contingency table.

Parameters **`observed`** – numpy array of observed cell counts

Returns theoretical ndof

`phik.statistics.`**`z_from_logp`**(*logp: float, flip_sign: bool = False*) → float

Convert logarithm of p-value into one-sided Z-value

Parameters

- **`logp`** (*`float`*) – logarithm of p-value
- **`flip_sign`** (*`bool`*) – flip sign of Z-value, e.g. use for input log(1-p). Default is false.

Returns statistical significance Z-value

Return type float

phik.version module

THIS FILE IS AUTO-GENERATED BY PHIK SETUP.PY.

Module contents

5.6 Indices and tables

- genindex
- modindex

Python Module Index

Index