jelinekstat Documentation

Release 0.1.1

Exneyder A. Montoya-Araque
Ludger O. Suarez-Burgoa

Jul 22, 2018

Contents:

Application software in **Python 3** to apply the statistical model for a sample of n 2^{nd}-order tensors Jelínek (1978) in order to obtain the mean tensor $\boldsymbol{k}$ of the sample, the $\boldsymbol{k}$'s principal values k_1, k_2, k_3, with their confidence intervals, and the $\boldsymbol{k}$'s principal directions $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ with their confidence regions.

This application program is able to plot the summary of the statistical model described above in a stereographic projection for a better understanding of the outcomes.

CHAPTER 1

Installation

1.1 Stable release

To install `jelinekstat`, run this command in your terminal:

```
$ pip install jelinekstat
```

This is the preferred method to install jelinekstat, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for `jelinekstat` can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/eamontoyaa/jelinekstat
```

Or download the tarball:

```
$ curl  -OL https://github.com/eamontoyaa/jelinekstat/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

Modules

This application software is divided into two modules. The first one is called `jelinekstat.py` and is the guideline of the second-order tensors statistical proposal of Jelínek (1978). The second module is called `tools.py`, and contains extra supportive tools for the functions contained in the first module and that are not included implicitly in the original reference.

2.1 jelinekstat

Module which contains the required functions to apply the statistical model for a sample of n second-order tensors Jelínek (1978) in order to obtain the mean tensor $\boldsymbol{k}$ of the sample, the $\boldsymbol{k}$'s principal values k_1, k_2 & k_3, with their confidence intervals, and the $\boldsymbol{k}$'s principal directions $\boldsymbol{p}_1, \boldsymbol{p}_2$ & $\boldsymbol{p}_3$ with their confidence regions.

This application program is able to plot the summary of the statistical model described above in a stereographic projection for a better understanding of the outcomes.

Note:

- The packages numpy, scipy, matplotlib and mplstereonet are required for using the `jelinekstat.py` module. All of them are downloadable from the PyPI repository.
- The mathematical notation in this documentation is taken from the original reference Jelínek (1978).
- Copyright (c) 2018, Universidad Nacional de Colombia, Medellín. Copyright (c) 2018, Exneyder A. Monotoya-Araque and Ludger O. Suarez-Burgoa. BSD-2-Clause or higher.

`jelinekstat.`**`normalizeTensors`**(*sample*)

Divides all the tensor's elements by the mean susceptibility k, *i.e.* gets $\boldsymbol{k}_{\text{norm}}$ using the equations (8) of Jelínek (1978).

Parameters **`sample`** (*numpy.ndarray*) – $(n \times 6)$ array that contains the values obtained from the `extractdata` fuction.

Returns $(n \times 6)$ array that contains the tensors $\boldsymbol{k}_{\text{norm}}$ with the same format and structure of the `extractdata` function's output.

Return type (*numpy.ndarray*)

Examples

```
>>> from jelinekstat.tools import dataFromFile
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> normalizeTensors(sample)
array(
    [[ 1.02327,  1.02946,  0.94727, -0.01495, -0.03599, -0.05574],
     [1.02315,  1.01803,  0.95882, -0.00924, -0.02058, -0.03151],
     [1.02801,  1.03572,  0.93627, -0.03029, -0.03491, -0.06088],
     [1.02775343,  1.00633335,  0.96591322, -0.01635005, -0.04148014,
      -0.02006007],
     [1.02143,  1.01775,  0.96082, -0.02798, -0.04727, -0.02384],
     [1.01822661,  1.01202663,  0.96974677, -0.01125996, -0.02832991,
      -0.03648988],
     [1.01486338,  1.0206734 ,  0.96446321, -0.01046003, -0.01913006,
      -0.03864013],
     [1.04596,  1.01133,  0.94271, -0.0166 , -0.04711, -0.03636]])
```

jelinekstat.**meantensor**(*sample*, *normalized=False*)

Estimates the mean tensor $\boldsymbol{k}$ of a randomly chosen sample of n specimens by using the equation (11) of Jelínek (1978) after being normalized the specimens through the `normalizeTensors` function.

Parameters

- **sample** (*numpy.ndarray*) – $(n \times 6)$ array that contains the values of the tensors after being imported with the `extractdata` function.
- **normalize** (*bool*) – Logical variable to indicate if the tensors in the `sample` variable are already normalized by using the equation (11) of (Jelínek (1978)). `False` is the default value. In the case they are not normalized, they will be.

Returns

Three elements are returned; they are described below.

- **meanTensorVect** (*numpy.ndarray*): Mean tensor in vector form.
- **meanTensorMtx** (*numpy.ndarray*): Mean tensor in matrix form.
- **numTensors** (*int*): Number of tensors.

Examples

```
>>> from jelinekstat.tools import dataFromFile
>>> from jelinekstat.jelinekstat import meantensor
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> meanTensorVect, meanTensorMtx, numTensors = meantensor(
        sample, normalized=False)
>>> meanTensorVect
array([ 1.02533293,  1.01891542,  0.95575165, -0.01714126, -0.03435001,
>>>    -0.03794001])
>>> meanTensorMtx
array([[ 1.02533293, -0.01714126, -0.03794001],
       [-0.01714126,  1.01891542, -0.03435001],
       [-0.03794001, -0.03435001,  0.95575165]])
```

(continues on next page)

(continued from previous page)

```
>>> numTensors
8
```

jelinekstat.**covMtx2PPlane**(*covMtx*, *meanTensor*, *numTensors*, *tensorVectForm=True*)

Obtains the covariance matrix $\mathbf{V^P}$ of the $\boldsymbol{k}^{\mathbf{P}}$'s elements (*i.e.* going over to a Cartesian system determined by $\boldsymbol{p}_1, \boldsymbol{p}_2$ & $\boldsymbol{p}_3$ as is shown equation (19) of Jelínek (1978)), by using the equation (20) and (21) of the same reference.

Parameters

- **covMtx** (*numpy.ndarray*) – (6×6) array that estimates the unbiased covariance matrix $\mathbf{V}$ of the tensors in the sample. It can be obtained by using the equation (13) of Jelínek (1978)) or the `cov` numpy function (as is shown in the example).
- **meanTensor** (*numpy.ndarray*) – mean tensor $\boldsymbol{k}$ of he sample either in vector or matrix form.
- **numTensors** (*int*) – Number of tensors in the sample.
- **tensorVectForm** (*bool*) – Logical variable to indicate if the input mean tensor is in vector form. `True` is the default value.

Returns (6×6) covariance matrix $\mathbf{V^P}$ of $\boldsymbol{k}^{\mathbf{P}}$'s elements.

Return type (*numpy.ndarray*)

Examples

```
>>> from numpy import cov
>>> from jelinekstat.tools import dataFromFile
>>> from jelinekstat.jelinekstat import meantensor, covMtx2PPlane
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> normTensors = normalizeTensors(sample)
>>> meanTensorVect, meanTensorMtx, numTensors = meantensor(
>>>     normTensors, normalized=True)
>>> covMtx = cov(normTensors.T, bias=False)
>>> covMtx2PPlane(
>>>     covMtx, meanTensorVect, numTensors)
array([[  1.02065332e-04,   4.74951816e-05,  -1.49560514e-04,
         -1.65998575e-06,   3.81988803e-05,  -2.24249042e-05],
       [  4.74951816e-05,   5.55684188e-05,  -1.03063600e-04,
         -2.03320518e-05,  -5.50208910e-06,  -1.06377394e-05],
       [ -1.49560514e-04,  -1.03063600e-04,   2.52624114e-04,
          2.19920375e-05,  -3.26967912e-05,   3.30626436e-05],
       [ -1.65998575e-06,  -2.03320518e-05,   2.19920375e-05,
          2.95042484e-05,   2.05384181e-05,  -7.98602106e-06],
       [  3.81988803e-05,  -5.50208910e-06,  -3.26967912e-05,
          2.05384181e-05,   9.00227092e-05,  -7.40633382e-05],
       [ -2.24249042e-05,  -1.06377394e-05,   3.30626436e-05,
         -7.98602106e-06,  -7.40633382e-05,   9.59108639e-05]])
```

jelinekstat.**localCovMtxs**(*meanTensor*, *pCovMtx*, *tensorVectForm=True*)

Determines the covariance matrix $\mathbf{W_i}$ of the random variables $(\mathrm{d}p_{ji}, \mathrm{d}p_{ki})$ from the local Cartesian System $\mathrm{d}\boldsymbol{p}_i$ that define the $P-$ plane where each confidence area of the mean tensor's princial direcions are drawn by using the equation (27) of Jelínek (1978).

Parameters

- **meanTensor** (*numpy.ndarray*) – mean tensor $\boldsymbol{k}$ of the sample either in vector or matrix form.
- **pCovMtx** (*numpy.ndarray*) – (6×6) covariance matrix $\mathbf{V^P}$ of $\boldsymbol{k^P}$'s elements. It is obtained with the `covMtx2PPlane` function.
- **tensorVectForm** (*bool*) – Logical variable to indicate if the input $\boldsymbol{k}$ is in vector form. `True` is the default value.

Returns

Three elements are returned; they are described below.

- **W** (*list*): list of three (2×2) arrayes with the covariance matrix $\mathbf{W}_i$ described above.
- **eigVal_W** (*list*): list with the three couples of $\mathbf{W}_i$'s eigenvalues obtained with the `getEigSorted` function.
- **eigVec_W** (*list*): list with the three (2×2) arrayes of $\mathbf{W}_i$'s eigevectors obtained with the `getEigSorted` function.

Examples

```
>>> from numpy import cov
>>> from jelinekstat.tools import dataFromFile
>>> from jelinekstat.jelinekstat import meantensor, covMtx2PPlane
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> normTensors = normalizeTensors(sample)
>>> meanTensorVect, meanTensorMtx, numTensors = meantensor(
>>>     normTensors, normalized=True)
>>> covMtx = cov(normTensors.T, bias=False)
>>> pCovMtx = covMtx2PPlane(
>>>     covMtx, meanTensorVect, numTensors)
>>> W, eigVal_W, eigVec_W = localCovMtxs(
>>>     meanTensorVect, pCovMtx)
>>> W
[array([[ 0.41635005, -0.00798796],
        [-0.00798796,  0.00679994]]),
 array([[ 0.00739345, -0.02211065],
        [-0.02211065,  0.41635005]]),
array([[ 0.00679994, -0.00565157],
       [-0.00565157,  0.00739345]])]
>>> eigVal_W
[array([ 0.41650579,  0.0066442 ]),
 array([ 0.41754201,  0.00620149]),
 array([ 0.01275605,  0.00143733])]
>>> eigVec_W
[array([[ 0.99980999,  0.01949312],
        [-0.01949312,  0.99980999]]),
 array([[ 0.05383071, -0.99855008],
        [-0.99855008, -0.05383071]]),
 array([[ 0.68831829, -0.7254088 ],
        [-0.7254088 , -0.68831829]])]
```

`jelinekstat.`**eigValsIntervals**(*pCovMtx*, *numTensors*, *confLvl=0.95*, *estimate=True*)

Determines the limits of the variabilities of $\boldsymbol{k}$'s principal values for a confidence level given. Ther are obtained by using the equation (29) of Jelínek (1978) or their estimate values by using the equation (35) of Jelínek (1978).

Parameters

- **pCovMtx** (*numpy.ndarray*) – (6×6) covariance matrix $\mathbf{V^P}$ of $\boldsymbol{k^P}$'s elements. It is obtained with the `covMtx2PPlane` function.
- **numTensors** (*int*) – Number of tensors in the sample.
- **confLvl** (*float*) – Confidence level of the limits of the variabilities of $\boldsymbol{k}$'s principal values. `0.95` is the default value.
- **estimate** (*bool*) – Logical variable to indicate if the output is based whether on the real or estimate covariance matrix $\mathbf{V^P}$. `True` is the default value.

Returns Array with the three limits of the variabilities of $\boldsymbol{k}$'s principal values.

Return type (*numpy.ndarray*)

Examples

```
>>> from numpy import cov
>>> from jelinekstat.tools import dataFromFile
>>> from jelinekstat.jelinekstat import *
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> normTensors = normalizeTensors(sample)
>>> meanTensorVect, meanTensorMtx, numTensors = meantensor(
>>>     normTensors, normalized=True)
>>> covMtx = cov(normTensors.T, bias=False)
>>> pCovMtx = covMtx2PPlane(
>>>     covMtx, meanTensorVect, numTensors)
>>> eigValsIntervals(
>>>     pCovMtx, numTensors, confLvl=0.95, estimate=True)
array([ 0.0084461 ,  0.00623205,  0.01328784])
```

jelinekstat.**eigVectsRegions**(*W*, *eigVal_W*, *eigVec_W*, *numTensors*, *confLvl=0.95*, *estimate=True*)

Determines the ellipses' geometric parameters of the confidence regions that define the limits of the variabilities of the $\boldsymbol{k}$'s principal vectors.

The axes lenghts ara obtained by using the equation (32) of Jelínek (1978) or their estimated values by using the equation (35) of Jelínek (1978) and the inclination angles by using the equation (41) of the same reference.

Parameters

- **W** (*list*) – list of three (2×2) arrayes with the covariance matrix $\mathbf{W_i}$ described above. It is obtained from the `eigValsIntervals` function.
- **eigVal_W** (*list*) – list with the three couples of $\mathbf{W_i}$'s eigenvalues obtained with the `getEigSorted` function. It is obtained from the `eigValsIntervals` function.
- **eigVec_W** (*list*) – list with the three (2×2) arrayes of $\mathbf{W_i}$'s eigevectors obtained with the `getEigSorted` function. It is obtained from the `eigValsIntervals` function.
- **numTensors** (*int*) – Number of tensors in the sample.
- **confLvl** (*float*) – Confidence level of the limits of the variabilities of $\boldsymbol{k}$'s principal vectors. `0.95` is the default value.
- **estimate** (*bool*) – Logical variable to indicate if the output is based whether on the real or estimate covariance matrix $\mathbf{V^P}$. `True` is the default value.

Returns

Three elements are returned; they are described below.

- **majorAxis** (*numpy.ndarray*): Array with the three lengths of the ellipses' major axis that define the confidence region. The order is acording to the principal values returned from the `getEigSorted` function.
- **minorAxis** (*list*): list with the three couples of $\mathbf{W_i}$'s eigenvalues obtained with the `getEigSorted` function. The order is acording to the principal values returned from the `getEigSorted` function.
- **theta** (*list*): list with the three ellipse inclinations in radians measured from the horizontal axis of the local Cartesian System of each ellipse to the respective major axis counter clockwise.

Examples:

```
>>> from numpy import cov
>>> from jelinekstat.tools import dataFromFile
>>> from jelinekstat.jelinekstat import *
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> normTensors = normalizeTensors(sample)
>>> meanTensorVect, meanTensorMtx, numTensors = meantensor(
>>>     normTensors, normalized=True)
>>> covMtx = cov(normTensors.T, bias=False)
>>> pCovMtx = covMtx2PPlane(
>>>     covMtx, meanTensorVect, numTensors)
>>> W, eigVal_W, eigVec_W = localCovMtxs(
>>>     meanTensorVect, pCovMtx)
>>> majorAxis, minorAxis, theta = eigVectsRegions(
>>>     W, eigVal_W, eigVec_W, numTensors, confLvl=0.95,
>>>     estimate=True)
>>> majorAxis
array([ 0.66888885,  0.66949335,  0.13745895])
>>> minorAxis
array([ 0.09950548,  0.09615434,  0.04640122])
>>> theta
array([-0.01949436, -1.51693959, -0.81162812])
```

`jelinekstat.`**`tensorStat`**(*sample*, *confLevel=0.95*, *want2plot=True*, *plotName='001'*, *ext='pdf'*)

Summarizes the Jelínek (1978) statisctic proposal for 2nd-order tensors and plots it if is wanted.

Parameters

- **`sample`** (*numpy.ndarray*) – $(nimes6)$ array that contains the values obtained from the `extractdata` fuction.
- **`confLvl`** (*float*) – Confidence level of the limits of the variabilities of $oldsymbolk$'s principal vectors and values. `0.95` is the default value.
- **`want2plot`** (*bool*) – Logical variable to indicate if is wanted to plot the summary. `True` is the default value.
- **`plotName`** (*str*) – Sample name for saving the final plot. '01' is the default value.
- **`ext`** (*str*) – File extension for saving the final plot. 'pdf' is the default value.

Returns

Summary of the Jelínek (1978) statisctic proposal for 2nd-order tensors where is stored the data related to the mean tensor and its variability expressed as the variability of their principal values and vectors.

Return type (*dict*)

Examples

```
>>> from jelinekstat.tools import dataFromFile
>>> from jelinekstat.jelinekstat import tensorStat
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> jelinekStatsSummary, stereonetPlot = tensorStat(
>>>     sample, confLevel=0.95, want2plot=True, plotName='example',
>>>     ext='pdf')
>>> jelinekStatsSummary
{'K': array([[ 1.02533293, -0.01714126, -0.03794001],
             [-0.01714126,  1.01891542, -0.03435001],
             [-0.03794001, -0.03435001,  0.95575165]]),
 'k': array([1.02533, 1.01891, 0.95575, -0.01714, -0.03435, -0.03794]),
 'n': 8,
 'k1': {'value': 1.042393712466853, 'variability': 0.008446101031382},
 'k2': {'value': 1.033975634080194, 'variability': 0.006232053391831},
 'k3': {'value': 0.92363065345295237, 'variability': 0.0132878448184},
 'p1': {'coords': array([-0.92438729,  0.196842  ,  0.32674357]),
  'plg': 19.071242330117354,
  'trd': 167.97880567817268,
  'majAx': 0.66888885495049721,
  'minAx': 0.099505476893979108,
  'incl': -1.1169443953398852},
 'p2': {'coords': array([-0.04467227, -0.90653975,  0.41975002]),
  'plg': 24.818806179014864,
  'trd': 267.17887306022931,
  'majAx': 0.66949335258868026,
  'minAx': 0.096154338416730253,
  'incl': -86.914236097035555},
 'p3': {'coords': array([-0.37883047, -0.37341521, -0.8467872 ]),
  'plg': 57.863927991299327,
  'trd': 44.587546531270618,
  'majAx': 0.13745894751107776,
  'minAx': 0.0464012210689351,
  'incl': -46.502865813326189}}
>>> stereonetPlot.show()
```

2.2 tools

Module which contains the functions that are supportive tools for the functions contained in the `jelinekstat.py` module which is the guideline of the second-order tensors statistical proposal of Jelínek (1978).

Note:

- The packages numpy, matplotlib and mplstereonet are required for using the `jelinekstat.py` module. All of them are downloadable from the PyPI repository.
- The mathematical notation in this documentation is taken from the original reference Jelínek (1978).
- Copyright (c) 2018, Universidad Nacional de Colombia, Medellín. Copyright (c) 2018, Exneyder A. Monotoya-Araque and Ludger O. Suarez-Burgoa. BSD-2-Clause or higher.

---

tools.**dataFromFile**(*file*)

Loads the `.txt` file with all second-order tensors components.

Parameters **file** (*str*) – `txt` file tabulated with tabular spaces as delimiter. The file is structured as a $(n \times 6)$ array, where n is the number of tensors and each row contains the vector form with the 6 components of a tensor in the following order $t_{11}, t_{22}, t_{33}, t_{12}, t_{23}, t_{13}$.

Returns

Two elements are returned; they are described below.

- **sample** (*numpy.ndarray*): $(n \times 6)$ array that contains the same values than the `.txt` file.
- **numTensors** (*int*): Number of tensors.

Examples

```
>>> from jelinekstat.tools import dataFromFile
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> sample
array([[ 1.02327,  1.02946,  0.94727, -0.01495, -0.03599, -0.05574],
       [ 1.02315,  1.01803,  0.95882, -0.00924, -0.02058, -0.03151],
       [ 1.02801,  1.03572,  0.93627, -0.03029, -0.03491, -0.06088],
       [ 1.02775,  1.00633,  0.96591, -0.01635, -0.04148, -0.02006],
       [ 1.02143,  1.01775,  0.96082, -0.02798, -0.04727, -0.02384],
       [ 1.01823,  1.01203,  0.96975, -0.01126, -0.02833, -0.03649],
       [ 1.01486,  1.02067,  0.96446, -0.01046, -0.01913, -0.03864],
       [ 1.04596,  1.01133,  0.94271, -0.0166 , -0.04711, -0.03636]])
>>> numTensors
8
```

tools.**tensorvect2matrixform**(*tensorVect*)

Converts a second order tensor from the vector form of to its matricial form.

Parameters **tensorVect** (*list* or *numpy.ndarray*) – $(n \times 6)$ tensor components written as column vector with the following order $t_{11}, t_{22}, t_{33}, t_{12}, t_{23}, t_{13}$.

Returns (3×3) second-order tensor expressed as a 3×3 matrix.

Return type (*numpy.ndarray*)

Examples

```
>>> from jelinekstat.tools import tensorvect2matrixform
>>> tensorVect = [11, 22, 33, 12, 23, 13]
>>> tensorvect2matrixform(tensorVect)
array([[11, 12, 13],
       [12, 22, 23],
       [13, 23, 33]])
```

tools.**vector2plungetrend**(*vector*)

Converts a $\mathbb{R}^3$ vector to the **plunge** δ, **trend** δ_{dir} notation used in Structural Geology and Rock Mechanics.

The $\mathbb{R}^3$ notation is assumed to be coincident with the **NED** notation (*i.e* North, East, Nadir).

Parameters **vector** (*list* or *numpy.ndarray*) – (x, y, z) vector.

Returns $(\delta, \delta_{\text{dir}})$ of the input vector in degrees.

Return type (*tuple*)

Examples

```
>>> from jelinekstat.tools import vector2plungetrend
>>> vector = [1, 0, 0]
>>> vector2plungetrend(vector)
(0.0, 0.0)
```

```
>>> from jelinekstat.tools import vector2plungetrend
>>> vector = [0, 1, 0]
>>> vector2plungetrend(vector)
(0.0, 90.0)
```

```
>>> from jelinekstat.tools import vector2plungetrend
>>> vector = [1, 1, 1]
>>> vector2plungetrend(vector)
(35.264389682754654, 45.0)
```

tools.**getEigSorted**(*matrix*)

Obtains eigenvalues and eigenvectors of a diagonalizable matrix. The eigenvalues are sorted descending.

Parameters **matrix** (*numpy.ndarray*) – (3×3) diagonalizable matrix.

Returns

Two elements are returned; they are described below.

- **sortedEigVal** (*numpy.ndarray*): (3×1) array with the eigenvalues ordered descending.
- **sortedEigVec** (*numpy.ndarray*): (3×3) array with the eigenvectors, such that the column `sortedEigVec[:, i]` is the eigenvector corresponding to the eigenvalue `sortedEigVal[i]`

Examples

```
>>> from numpy import array
>>> from jelinekstat.tools import getEigSorted
>>> matrix = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> sortedEigVal, sortedEigVec = getEigSorted(matrix)
>>> sortedEigVal
array([  1.61168440e+01,  -9.75918483e-16,  -1.11684397e+00])
>>> sortedEigVec
array([[-0.23197069,  0.40824829, -0.78583024],
       [-0.52532209, -0.81649658, -0.08675134],
       [-0.8186735 ,  0.40824829,  0.61232756]])
```

tools.**confRegions2PPlanes**(*majorAxis*, *minorAxis*, *theta*, *want2plot=True*, *confLvl=0.95*)

Determines the $\mathbb{R}^2$ coordinates of each confidence ellipse in the local Cartesyan System of the $P-$ planes that contain them. It is donde from their geometric parameters obtained from the `eigVectsRegions` function. If it is wanted, plots them too.

Parameters

- **majorAxis** (*numpy.ndarray*) – Array with the three lengths of the ellipses' major axis that define the confidence region. The order is acording to the principal values returned from the `getEigSorted` function.
- **minorAxis** (*list*) – list with the three couples of $\mathbf{W}_i$'s eigenvalues obtained with the `getEigSorted` function. The order is acording to the principal values returned from the `getEigSorted` function.
- **theta** (*list*) – list with the three ellipse inclinations in radians measured from the horizontal axis of the local Cartesian System of each ellipse to the respective major axis counter clockwise.
- **want2plot** (*bool*) – Logical variable to indicate if is wanted to plot the ellipes. `True` is the default value.
- **confLvl** (*float*) – Confidence level of the limits of the variabilities of $\boldsymbol{k}$'s principal vectors and values. `0.95` is the default value.

Returns

Three elements are returned; they are described below.

- **x** (*list*): List of three arrays that contain the abscises of the each ellipse.
- **y** (*list*): List of three arrays that contain the ordinates of the each ellipse.
- **fig** (*list*): `matplotlib` object. use `fig.show()` for displaying the plot

Examples

```
>>> from numpy import array
>>> from jelinekstat.tools import confRegions2PPlanes
>>> majorAxis = array([ 0.66888885,  0.66949335,  0.13745895])
>>> minorAxis = array([ 0.09950548,  0.09615434,  0.04640122])
>>> theta = array([-0.01949436, -1.51693959, -0.81162812])
>>> x, y, fig = confRegions2PPlanes(majorAxis, minorAxis, theta,
>>>                                 want2plot=True, confLvl=0.95)
>>> fig.show()
```

`tools.`**rotateaxis2proyectellipses**(*axisN*, *axisE*, *axisD*)

Since it is easier, the projection of an ellpise (confidence region) on the stereogrpahic net is thought as a serie of rotations from the bottom of the semi-sphere, *i.e.*, the *nadir*.

This function determines the axes names around which is necesary to rotate a confidence ellipse once it is placed at nadir of a semi-sphere to poject her from the nadir to the real position on the semi-sphere. Besides, it determines the angles to rotate at each axis name.

The `mplstereonet` reference system has the x, y and z vectors as its base, and they correspond to the *nadir*, *east* and *north* vectors in the **NED** reference system of the semi-spherical space of the Stereographic Projection.

It is implicit that the three input vectors are orthogonal to each other due to they correspond to the principal vectors of $\boldsymbol{k}$.

Parameters

- **axisN** (*numpy.ndarray*) – Array with the coordinates x, y, z of the eigenvector that will point to the north-axis once the ellipse is placed at nadir of the semi-sphere, *i.e.*, its othogonal eigenvector associated points downward.

- **axisE** (*numpy.ndarray*) – Array with the coordinates x, y, z of the eigenvector that will point to the east-axis once the ellipse is placed at nadir of the semi-sphere, *i.e.*, its othogonal eigenvector associated points downward.
- **axisD** (*numpy.ndarray*) – Array with the coordinates x, y, z of the eigenvector that is othogonal to the ellipse.

Returns

Two elements are returned; they are described below.

- **axis2rot** (*list*): Strings with the axis-names of the **NED** system around which will be done the rotatios to project the confidence ellipse.
- **angles2rot** (*list*): List of the angles in degrees for rotating a ellipse once it is placed orthogonal to the nadir.

Examples

```
>>> from numpy import array
>>> from jelinekstat.tools import rotateaxis2proyectellipses
>>> axis2rot, angles2rot = rotateaxis2proyectellipses(
>>>     array([2, 2, 1]), array([-2, 1, 2]), array([1, -2, 2]))
>>> axis2rot
['E', 'D', 'N', 'D']
>>> angles2rot
[-180, 26.565051177077976, 48.189685104221404, 206.56505117707798]
```

tools.**proyAnEllipse2LongLat**(*x*, *y*, *axis2rot*, *angles2rot*)

Pojects just an ellipse from the $\mathbb{R}^2$ coordinates of the $P-$ plane that contains it to the real position on the stereographic projection through some rotations from an initial position at the nadir of the semi-sphere.

Parameters

- **x** (*numpy.ndarray* or *list*) – Abscises of just one ellipse's boundary on the $P-$ plane. It is obtained from the `confRegions2PPlanes` function.
- **y** (*numpy.ndarray* or *list*) – Ordinates of just one ellipse's boundary on the $P-$ plane. It is obtained from the `confRegions2PPlanes` function.
- **axis2rot** (*list*) – Strings with the axis-names of the **NED** system around which will be done the rotatios to project the confidence ellipse. It is obtained from the `rotateaxis2proyectellipses` function.
- **angles2rot** (*list*) – List of the angles in degrees for rotating a ellipse once it is placed orthogonal to the nadir. It is obtained from the `rotateaxis2proyectellipses` function.

Returns

Two elements are returned; they are described below.

- **ellipLong** (*numpy.ndarray*): Longitudes of the ellipse's boundary after being rotated to its right position in the stereographic projection.
- **ellipLat** (*numpy.ndarray*): Latitudes of the ellipse's boundary after being rotated to its right position in the stereographic projection.

Examples

```
>>> from numpy import array
>>> from jelinekstat.tools import confRegions2PPlanes
>>> majorAxis = array([ 0.66888885,  0.66949335,  0.13745895])
>>> minorAxis = array([ 0.09950548,  0.09615434,  0.04640122])
>>> theta = array([-0.01949436, -1.51693959, -0.81162812])
>>> x, y = confRegions2PPlanes(majorAxis, minorAxis, theta, False,
>>>                            0.95)
>>> ellipLong, ellipLat = proyAnEllipse2LongLat(
>>>     x[0], y[0], ['E', 'D', 'N', 'D'], [-180, 116.37, 70.93, 77.98])
```

tools.**eigVects2PlgTrd**(*tensor*, *tensorVectForm=True*)

Obtains the principal vectors of a second-order tensor and returns them in the the **plunge**, **trend** $(\delta, \delta_{\text{dir}})$ notation used in Structural Geology and Rock Mechanics.

Parameters

- **tensor** (*numpy.ndarray*) – A secon-order tensor.
- **tensorVectForm** (*bool*) – Logical variable to indicate if the input tensor is in vector form. `True` is the default value.

Returns

Two elements are returned; they are described below.

- **eigVecPlg** (*list*): Plunges of the three principal vectors of the input tensor. The order is acording to the principal values returned from the `getEigSorted` function.
- **eigVecTrd** (*list*): Trends of the three principal vectors of the input tensor. The order is acording to the principal values returned from the `getEigSorted` function.

Examples

```
>>> from numpy import array
>>> from jelinekstat.tools import eigVects2PlgTrd
>>> tensor = array([1.023, 1.0295,  0.9473, -0.0150, -0.0360, -0.056])
>>> eigVecPlg, eigVecTrd = eigVects2PlgTrd(
>>>     tensor, tensorVectForm=True)
>>> eigVecPlg
[31.176002127688509, 7.2363353791762837, 57.806971200122995]
>>> eigVecTrd
[198.07283722120425, 292.47894708173089, 34.114473250861067]
```

tools.**proyAllEllipses2LongLat**(*x*, *y*, *meanTensor*, *tensorVectForm=True*)

Pojects all the three confidence ellipses from the $mathbbR^2$ coordinates of the $mathscrP-$ plane that contain them to the real position on the stereographic projection through some rotations from an initial position at the nadir of the semi-sphere.

Parameters

- **x** (*numpy.ndarray* or *list*) – Arrangement of the three lists each one with the abscises of an ellipse's boundary on the $mathscrP-$ plane. They are obtained from the `confRegions2PPlanes` function.
- **y** (*numpy.ndarray* or *list*) – Arrangement of the three lists each one with the ordinates of an ellipse's boundary on the $mathscrP-$ plane. They are obtained from the `confRegions2PPlanes` function.

- **meanTensor** (*numpy.ndarray*) – mean tensor $\boldsymbol{k}$ of the sample either in vector or matrix form.
- **tensorVectForm** (*bool*) – Logical variable to indicate if the input $\boldsymbol{k}$ is in vector form. `True` is the default value.

Returns

Two elements are returned; they are described below.

- **long** (*numpy.ndarray*): Array of the three lists each one with the longitudes (in radians) of all the ellipse's boundary after being rotated to its right position in the stereographic projection.
- **lat** (*numpy.ndarray*): Array of the three lists each one with the latitudes (in radians) of all the ellipse's boundary after being rotated to its right position in the stereographic projection.

Examples

```
>>> from numpy import array
>>> from jelinekstat.tools import *
>>> from jelinekstat.jelinekstat import meantensor
>>> sample, numTensors = dataFromFile('inputDataExample.txt')
>>> meanTensorVect, meanTensorMtx, numTensors = meantensor(
>>>     sample, normalized=True)
>>> majorAxis = array([ 0.66888885,  0.66949335,  0.13745895])
>>> minorAxis = array([ 0.09950548,  0.09615434,  0.04640122])
>>> theta = array([-0.01949436, -1.51693959, -0.81162812])
>>> x, y = confRegions2PPlanes(
>>>     majorAxis, minorAxis, theta, False, 0.95)
>>> long, lat = proyAllEllipses2LongLat(x, y, meanTensorVect)
```

`tools.`**splitIterables**(*iter1*, *iter2*)

Splits two iterable elements which are paired by selecting the math:*n* common indexes where there are sign changes in both inputs at the same time. If there is any index to split the inputs, it returns the same inputs within a list.

Parameters

- **iter1** (*numpy.ndarray* or *list*) – An iterable element which is paired to `iter2`.
- **iter2** (*numpy.ndarray* or *list*) – An iterable element which is paired to `iter1`.

Returns

Two elements are returned; they are described below.

- **iter1Splitted** (*list*): Segments of the original `iter1` input after being splitted.
- **iter2splitted** (*list*): Segments of the original `iter2` input after being splitted.

Examples

```
>>> from jelinekstat.tools import splitIterables
>>> iter1, iter2 = [1, -2, -3, 4, 5, 6], [-3, -2, -1, 0, 1, 2]
>>> iter1Splitted, iter2splitted = splitIterables(iter1, iter2)
>>> iter1Splitted
[[1, -2, -3], [4, 5, 6]]
>>> iter2splitted
[[-3, -2, -1], [0, 1, 2]]
```

CHAPTER 3

Use and Examples

3.1 Importing data from file

In some cases the data to process is stored in external files. Due to those cases, this application software is able to import the data from an external tabulate `.txt` which contains the data of the randomly selected sample. It is done by using the function `dataFromFile` from the `tools.py` module.

Some conditions are required in the file format for importing it. It has to be structured as a $(n \times 6)$ array, where math:n is the number of tensors and each row contains a tensor in the vector form with the 6 components sorted like this order $t_{11}, t_{22}, t_{33}, t_{12}, t_{23}, t_{13}$.

The `exampledata.txt` file has the following content:

```
1.02327    1.02946    0.94727    -0.01495    -0.03599    -0.05574
1.02315    1.01803    0.95882    -0.00924    -0.02058    -0.03151
1.02801    1.03572    0.93627    -0.03029    -0.03491    -0.06088
1.02775    1.00633    0.96591    -0.01635    -0.04148    -0.02006
1.02143    1.01775    0.96082    -0.02798    -0.04727    -0.02384
1.01823    1.01203    0.96975    -0.01126    -0.02833    -0.03649
1.01486    1.02067    0.96446    -0.01046    -0.01913    -0.03864
1.04596    1.01133    0.94271    -0.01660    -0.04711    -0.03636
```

And the minimum script for importing `exampledata.txt` is the following:

```
# import the function
from jelinekstat.tools import dataFromFile

sample, numTensors = dataFromFile('exampledata.txt')
```

3.2 Examples

Two ways for executing the application software via script are presented below.

3.2.1 Short script

The first way is by using the function `tensorStat` from the `jelinekstat.py` module in a short script as follow.

```
# import the functions
from jelinekstat.jelinekstat import tensorStat

# Input data
sample = [[1.02327, 1.02946, 0.94727, -0.01495, -0.03599, -0.05574],
          [1.02315, 1.01803, 0.95882, -0.00924, -0.02058, -0.03151],
          [1.02801, 1.03572, 0.93627, -0.03029, -0.03491, -0.06088],
          [1.02775, 1.00633, 0.96591, -0.01635, -0.04148, -0.02006],
          [1.02143, 1.01775, 0.96082, -0.02798, -0.04727, -0.02384],
          [1.01823, 1.01203, 0.96975, -0.01126, -0.02833, -0.03649],
          [1.01486, 1.02067, 0.96446, -0.01046, -0.01913, -0.03864],
          [1.04596, 1.01133, 0.94271, -0.01660, -0.04711, -0.03636]]
confLevel = 0.95

jelinekStatsSummary, stereonetPlot = tensorStat(
        sample, confLevel=0.95, want2plot=True, plotName='shortSCRoutcome',
        ext='pdf')
stereonetPlot.show()
```

3.2.2 Long script

The second way is by using all the code lines inside the same function used above in a much longer script as follow.

```
import numpy as np
import matplotlib.pyplot as plt

from jelinekstat.jelinekstat import normalizeTensors, meantensor, \
    covMtx2PPlane, eigValsIntervals, localCovMtxs, eigVectsRegions
from jelinekstat.tools import getEigSorted, confRegions2PPlanes, \
    eigVects2PlgTrd, proyAllEllipses2LongLat, splitIterables

# Input data.
sample = [[1.02327, 1.02946, 0.94727, -0.01495, -0.03599, -0.05574],
          [1.02315, 1.01803, 0.95882, -0.00924, -0.02058, -0.03151],
          [1.02801, 1.03572, 0.93627, -0.03029, -0.03491, -0.06088],
          [1.02775, 1.00633, 0.96591, -0.01635, -0.04148, -0.02006],
          [1.02143, 1.01775, 0.96082, -0.02798, -0.04727, -0.02384],
          [1.01823, 1.01203, 0.96975, -0.01126, -0.02833, -0.03649],
          [1.01486, 1.02067, 0.96446, -0.01046, -0.01913, -0.03864],
          [1.04596, 1.01133, 0.94271, -0.01660, -0.04711, -0.03636]]
confLevel = 0.95

# PERFORMING THE CALCULATION STEP BY STEP.

# Normalization of the imput sample
sample = normalizeTensors(sample)

# Mean tensor from normalized data and sample size (n)
k, K, n = meantensor(sample, True)  # k (vector); K (matrix)

# Eigenvalues (kK) and eigenvectors (pK) of the mean tensor
```

(continues on next page)

(continued from previous page)

```
kK, pK = getEigSorted(K)

# Unbiased covariance matrix (V).
V = np.cov(sample.T, bias=False)

# Covariance matrix (V) in the system of the k's principal vectors (pV).
pV = covMtx2PPlane(V, k, n)

# Confidence intervals of eigenvalues of mean tensor (kIntervals).
kIntervals = eigValsIntervals(pV, n, confLevel)

# Local covariance matrices (W) in each P-plane of each confidence region.
W, eigValW, eigVectW = localCovMtxs(k, pV)

# Length and orientation of ellipses semi-axis.
majorAxis, minorAxis, theta = eigVectsRegions(
        W, eigValW, eigVectW, n, confLevel)

# Coordiantes of the three ellipses in each P-plane.
x, y, PPlanePlots = confRegions2PPlanes(
        majorAxis, minorAxis, theta, True, confLevel)

# Stereographic notation to plot the mean tensor's principal vectors (pK).
pKPlg, pKTrd = eigVects2PlgTrd(k)  # Plg (plunge); Trd (trend)

# (plunge,trend) notation to plot principal axis of all tensors.
samplePlgTrd = list(map(eigVects2PlgTrd, sample))

# (lon, lat) notation of each confidence region.
kRegionsLong, kRegionsLat = proyAllEllipses2LongLat(x, y, k)

# Summary of the Jelinek (1978) statistic proposal for 2nd-order tensors.
jelinekStatSummary = {
        'k': k,
        'n': n,
        'k1': {'mean': kK[0], 'variability': kIntervals[0]},
        'k2': {'mean': kK[1], 'variability': kIntervals[1]},
        'k3': {'mean': kK[2], 'variability': kIntervals[2]},
        'p1': {'coords': pK[:, 0], 'plg': pKPlg[0], 'trd': pKTrd[0],
               'majAx': majorAxis[0], 'minAx': minorAxis[0],
               'incl': np.degrees(theta[0])},
        'p2': {'coords': pK[:, 1], 'plg': pKPlg[1], 'trd': pKTrd[1],
               'majAx': majorAxis[1], 'minAx': minorAxis[1],
               'incl': np.degrees(theta[1])},
        'p3': {'coords': pK[:, 2], 'plg': pKPlg[2], 'trd': pKTrd[2],
               'majAx': majorAxis[2], 'minAx': minorAxis[2],
               'incl': np.degrees(theta[2])}
        }

# Plotting.
fig = plt.figure(num='Jelinek plot summary')
plt.ioff()
markers = ['s', '^', 'o']
labels = ['$k_1 = ' + str(round(kK[0], 3)) + '\pm' +
          str(round(kIntervals[0], 3)) + '$',
          '$k_2 = ' + str(round(kK[1], 3)) + '\pm' +
          str(round(kIntervals[1], 3)) + '$',
```

(continues on next page)

(continued from previous page)

```
          '$k_3 = ' + str(round(kK[2], 3)) + '\\pm' +
          str(round(kIntervals[2], 3)) + '$']
ax = fig.add_subplot(111, projection='stereonet')
# Eigenvectors of all tensors
for tensor in samplePlgTrd:
    for i in range(3):
        ax.line(tensor[0][i], tensor[1][i], markers[i], color='0.3',
                ms=5, fillstyle='none')
# Eigenvectors of mean tensor
for i in range(3):
    ax.line(pKPlg[i], pKTrd[i], markers[i], color='k', ms=7,
            label=labels[i])
# Confidence regions
for i in range(3):
    kRegionsLongSplitted, kRegionsLatSplitted = splitIterables(
            kRegionsLong[i], kRegionsLat[i])
    for i in range(len(kRegionsLongSplitted)):
        ax.plot(kRegionsLongSplitted[i], kRegionsLatSplitted[i], ':k',
                lw=1)
# Empty plot to add the confidence region legends.
confLvl = str(round(confLevel * 100, 1))
ax.line(0, 0, ':k', lw=1, label='$'+confLvl + '\%$ conf. regions')
ax.legend(loc=tuple(np.radians([45, -7])), fontsize='x-small')
ax.grid(True, ls='--', lw=0.5)
fig.savefig('longSCRoutcome.pdf', bbox_inches='tight')
fig.show()
```

Since it is the same picture than the obtained with the **short script**, it is not displayed again.

CHAPTER 4

Authors

- Exneyder A. Montoya-Araque <eamontoyaa@gmail.com>
- Ludger O. Suarez-Burgoa <losuarezb@unal.edu.co>

CHAPTER 5

License

BSD-2-Clause

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER 6

History

6.1 0.1.0 (2018-07-08)

- First release on PyPI.

6.2 0.1.1 (2018-07-12)

- Documentation and the examples directory were completely adjusted.

CHAPTER 7

References

Jelínek, V (1978). Statistical processing of anisotropy of magnetic susceptibility measured on group of specimens. Studia Geophysica et Geodaetica, 22 (1), pp. 50-62.

CHAPTER 8

Links

- Documentation
- PyPI
- GitHub

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

CHAPTER 10

License and Copyright

Python Module Index

j

t

Index