8CC00_1 Documentation

Rebecca Kuepper

Feb 27, 2021

CONTENTS:

CHAPTER

ONE

SOURCE FILES

1.1 PCA module

Principal Component Analysis tools

class `AssignmentPCA.`**`AssignmentPCA`**

Bases: `object`

Class for principal component analysis of the CellLineRMAExpression data.

`calcLoads`(*n: int*, *eigpairs: list*, *varNames: list*) → list

Calculate the loads of the variables on given PC.

Assumptions: * PC number is in range * eigpairs and varNames have the same length * eigpairs and varNames are not empty

Parameters

- **`n`** -- PC number (starting at 1).
- **`eigpairs`** -- Sorted list (high-low) containing tuples of (eigVal, eigVec).
- **`varNames`** -- List containing strings of the variable names in the same order as eigpairs.

Returns List of (load, varName) tuples, sorted with highest load first.

`covariance`(*param1: list*, *param2: list*) → float

Return the covariance of parameter lists param1 and param2.

Assumption: param1 and param2 contain numbers and are of equal length.

Parameters

- **`param1`** -- List of parameters to be compared.
- **`param2`** -- List of parameters to compare with .

Returns covariance of param1 and param2.

```
>>> self.covariance([1, 3, 5, 11, 0, 4], [2, 6, 2, 78, 1, 4])
106.4
>>> self.covariance([1], [1, 2])
Traceback (most recent call last):
    ...
AssertionError: Parameter lists must be of the same length.
```

`cumulativeMovingAverage`(*x: list*) → list

Return a list of the cumulative moving average of input parameterlist x.

Assumption: x contains numbers or is empty.

Parameters **`x`** -- list of numbers.

Returns list of float cumulative moving averages.

```
>>> self.cumulativeMovingAverage([1, 3, 5, 11, 0, 4])
[1.0, 2.0, 3.0, 5.0, 4.0, 4.0]
>>> self.cumulativeMovingAverage([])
[]
```

`listOfCellLineNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18`

`plotCumulativeMovingAverage`(*x: list*, *title: str = 'Cumulative moving average'*) → None
Plot the cumulative moving average of a list x

Assumption: x contains numbers or is empty.

Parameters

- **`x`** -- List of parameters to be calculated moving average of and plotted.
- **`title`** -- string containing a title for the graph.

`readRMAExpressionAssigned`() → list
Return a list of RMA expression values of assigned cell lines stored in self.listOfCellLineNumbers.

1.2 cell line module

Analysis of dataset. (RMA = Rombust Multi-array Averages)

class `CellLineRMAExpression.`**`CellLineRMAExpression`**(*type: str*)
Bases: `object`

Class for analysis of cancer cell data.

`cancerType`(*cellLine: str*) → str
Return the name of the cancer type for a given cell line.

Assumption: cellLine exists.

Parameters **`cellLine`** – String containing the name of a cell line.

Returns String containing the name of the type of cancer with which the cell line is associated, or None if cellLine doesn't exist.

```
>>> self.cancerType('AU565')
BRCA
>>> self.cancerType('')
None
```

`readRMAExpression`(*cellLine: str*) → list
Read the RMA expression of a single cell line.

Assumption: cellLine in dataset.

Parameters **`cellLine`** – String, cell line from which the RMA expression is to be read.

Returns list of RMA expressions of all 244 genes of cellLine, or None if cellLine does not exist.

```
>>> self.readRMAExpression('')
None
```

(continues on next page)

(continued from previous page)

```
>>> len(self.readRMAExpression('AU565'))
244
```

1.3 main module

Main file for PCA Analysis of cancer cell RNA expression of different genes.

In this modules, the following steps were taken to perform PCA:

1. The data was read and converted to a np array.
2. The covariance of the genes was calculated using the `covariance()` function from the *AssignmentPCA* module.
3. From this matrix, the eigenvalues and eigenvectors were calculated and sorted by relevance.
4. The cumulative overall variance was then calculated.
5. A bar graph of principal component contribution and a scree plot were created.
6. It was decided to reduce to 3 dimensions.
7. A new subspace with only 3 PC's was created.
8. A 2D and 3D principal component plot were generated.
9. Loads for the three major PC's were calculated using `calcLoads()` from *AssignmentPCA*.
10. For each of the three major PC's, the 50 highest loads of the genes has been shown in a bar graph.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX