Package FPOPapprox2D

Description

FPOPapprox2D is an R package developed in Rcpp/C++ performing parametric changepoint detection in p-variate time series with the Gaussian cost function. Changepoint detection uses the Functional Pruning Optimal Partitioning method (FPOP) based on an exact dynamic programming algorithm.
The package implements the following geometry types for FPOP-pruning:

- **Geometry 1**: "$\cap$", approximating set - *rectangle*;
- **Geometry 2**: "$\cap\backslash\cup$", approximating set - *rectangle*;
- **Geometry 3**: "$\backslash\cup$", approximating set - *disk*.

Package structure

- Part R
  - FPOPapprox2D.R
    The file contains the implementation of the function **data_genDp** - generation of data of dimension p with a given values of means and changepoints
  - RcppExports.R
    The file contains the implementation of the function **FPOPDp** that calls the function **FPOPDp** implemented in C++.
- Part C++
  - GausseCostDp.h, GausseCostDp.cpp
    The files contain the implementation code for the class **GausseCostDp**.
  - DiskDp.h, DiskDp.cpp
    The files contain the implementation code for the class **DiskDp**.
  - RectDp.h, RectDp.cpp
    The files contain the implementation code for the class **RectDp**.
  - Geom1Dp.h, Geom1Dp.cpp
    The files contain the implementation code for the class **Geom1Dp**.
  - Geom2Dp.h, Geom2Dp.cpp
    The files contain the implementation code for the class **Geom2Dp**.
  - Geom3Dp.h, Geom3Dp.cpp
    The files contain the implementation code for the class **Geom3Dp**.
  - OPDp.h, OPDp.cpp
    The files contain the implementation code for the class **OPDp**.
  - main.cpp
    The file contains the code of the function **FPOPDp** that implements the change-point detection in p-variate time-series using the Functional Pruning Optimal Partitioning algorithm.
  - RcppExports.cpp
    The file contains the code that exports data R/C ++ .

Class GausseCostDp

We consider $(x^0, .., x^p)$ - p-variate time-series when $x^i = (x^i_0, .., x^i_{n-1}), \quad i = 0 : (p-1)$ - the vectors of univariate data size n.
We use the Gaussian cost of the segmented p-variate data when m_t is the value of the optimal cost, $m_0 = -\beta$. We introduce the notations:

$$\begin{aligned} mu_k &= E(x^k_{i:t}); \\ coef &= (t - i + 1); \\ mi\_1\_p &= m_{i-1} + \beta; \\ coef\_Var &= (t - i + 1) \cdot \sum_{k=0}^{p-1} Var(x^k_{i:t}). \end{aligned} \tag{1}$$

The Gaussian cost function takes the form:

$$q^i_t(\theta) = mi\_1\_p + coef \cdot (sum^{p-1}_{k=0}(\theta_k - mu - k)^2 + coef\_Var, \quad i = 1 : t. \tag{2}$$

The class characteristics (1):

- **p, coef, coef_Var, mi_1_p, mu**

The class implements:

- the constructors:
  - **GausseCostDp(unsigned int dim)** The Gaussian cost function of dimension $p = dim$ at the initial moment. All parameters except p are equal to zero.
  - **GausseCostDp(unsigned int dim, unsigned int i, unsigned int t, double* si_1, double* st, double mi_1pen)** The Gaussian cost function of dimension $p = dim$ at the time $[i, t]$.
- the constructor copy **GausseCostDp(const GausseCostDp &cost)**
- the destructor **~GausseCostDp()**
- the class methods for accessing characteristics, for getting the minimum value and initializing the Gaussian cost function:
  - **get_p(), get_coef(), get_coef_Var(), get_mi_1_p(), get_mu()**
  - **get_min()**
  - **InitialGausseCostDp(unsigned int dim, unsigned int i, unsigned int t, double* si_1, double* st, double mi_1pen)**

Class DiskDp

The class characteristics:

- A class element is a circle in dimension **p** that is defined by the center coordinates **center** and the radius **radius**.

The class implements:

- the constructors:
  - **DiskDp(unsigned int dim)** The disk of dimension **p** as dim. All parameters except **p**are equal to zero.

  - **DiskDp(unsigned int dim, double* c, double r)** The disk of dimension **p** as *dim* when vector c is the center coordinates and r is the radius of the circle.

- the constructor copy **DiskDp(const DiskDp &disk)**
- the destructor ~**DiskDp()**
- the class methods for accessing characteristics and initializing the disk:
  - **get_p(), get_center(), get_radius()**
  - **InitialDiskDp(unsigned int dim, double* c, double r)**

Class RectDp

The class characteristics:

- A class element is a rectangle in dimension **p**. The coordinates of the rectangle are defined using the matrix of contrains **coordinates**. Each row contains two constraint values for each axis.

The class implements:

- the constructors:
  - **RectDp(unsigned int dim)**
    The rectangle of dimension $p = dim$ with the constrains:
    $$coordinates_{i,0} = -\inf, \quad coordinates_{i,1} = \inf, \quad i = 0 : p - 1.$$
  - **RectDp(unsigned int dim, double** coords)** The rectangle of dimension $p = dim$ with the matrix of contrains $coordinates = coords$.
- the constructor copy **RectDp(const RectDp &rect)**
- the destructor ~**RectDp()**
- the class methods for accessing characteristics of rectangle:
  - **get_p(), get_coordinates()**
- the methods to find the minimum and maximum of two numbers:
  - **min_ab(double a, double b), max_ab(double a, double b)**
- **bool IsEmpty_rect()** The function checks the correctness of the coordinates of the rectangle and returns a value *true* if the coordinates are not correct.
- **Intersection_disk(DiskDp disk)** The function approximates a rectangle and a circle intersection area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the intersection area of the rectangle and the circle.
- **Exclusion_disk(DiskDp disk)** The function approximates a rectangle and a circle difference area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the difference area of the rectangle and the circle.

Class Geom1Dp

The class characteristics:

- This class implements the geometry **"Geometry 1"** for FPOP-pruning.
  - **p** is the value of dimension.
  - **label_t** is the time moment.
  - **rect_t** is the pointer to rectangle in dimension **p**, the rectangle is the element of class **RectDp**.

The class implements:

- the constructors:
  - **Geom1Dp(unsigned int dim)** Initializing the dimension **p** as *dim* and a pointer **rect_t** to the rectangle in the dimension **p**.
  - **Geom1Dp(unsigned int dim, unsigned int t)** Initializing the dimension **p**, the time moment **label_t** as *dim*, *t* and the pointer to the rectangle in the dimension **p**.
- the constructor copy **Geom1Dp(const Geom1Dp & geom1)**
- the destructor **~Geom1Dp()**
- the class methods for accessing characteristics of the geometry:
  - **get_p(), get_label_t()**
- **get_disks_t_1(), CleanGeometry()** The methods don't have a body (empty) and are written only for the correct operation of the FPOP-algorithm template.
- **InitialGeometry(unsigned int dim, unsigned int t,std::list<DiskDp> disks)** Initializing the time moment **label_t** as *t* in the **Geometry 1**.
- **UpdateGeometry(DiskDp disk)** The function **Intersection_disk(DiskDp disk)** implemented in the **RectDp** class is applied to the rectangle at the **rect_t** pointer.
- **EmptyGeometry()** The function checks the parameters of rectangle at the **rect_t** pointer. If the parameters are not correct, this rectangle is empty.

Class Geom2Dp

The class characteristics:

- This class implements the geometry type **"Geometry 2"** for FPOP-pruning.
  - **p** is the value of dimension.
  - **label_t** is the time moment.
  - **disks_t_1** is the list of active circles for the moment $t-1$.
  - **rect_t** is the pointer to rectangle in dimension **p**, the rectangle is the element of class **RectDp**.

The class implements:

- the constructors:

  - **Geom2Dp(unsigned int dim)** Initializing the dimension **p** as *dim* and a pointer **rect_t** to the rectangle in the dimension **p**.
  - **Geom2Dp(unsigned int dim, unsigned int t)** Initializing the dimension **p**, the time moment **label_t** as *dim*, *t* and the pointer **rect_t** to the rectangle in the dimension **p**.
- the constructor copy **Geom2Dp(const Geom2Dp & geom2)**
- the destructor ~**Geom2Dp()**
- the class methods for accessing characteristics of the geometry:
  - **get_p(), get_label_t()**
- **get_disks_t_1(), CleanGeometry()** These methods don't have a body (empty) and are written only for the correct operation of the FPOP-algorithm template.
- **InitialGeometry(unsigned int dim, unsigned int t,std::list<DiskDp> disks)** Initializing the time moment **label_t** as *t* in the **Geometry 2**.
- **UpdateGeometry(DiskDp disk)** The function **Intersection_disk(DiskDp disk)** implemented in the **RectDp** class is applied to the rectangle at the **rect_t** pointer.
- **EmptyGeometry()** The function checks the parameters of rectangle at the **rect_t** pointer. If the parameters are not correct, this rectangle is empty.

Class Geom3Dp

The class characteristics:

- This class implements the geometry type **"Geometry 3** for FPOP-pruning.
  - **p** is the value of dimension.
  - **label_t** is the time moment.
  - **fl_empty** is $false$ if geometry exists, otherwise $true$.
  - **disks_t_1** is the list of active circles for the moment $t-1$.

The class implements:

- the constructors:
  - **Geom3Dp(unsigned int dim)** Initializing the dimension **p** as *dim*.
  - **Geom3Dp(unsigned int dim, unsigned int t)** Initializing the dimension **p** and the time moment **label_t** as *dim* and *t*.
- the constructor copy **Geom3Dp(const Geom3Dp & geom3)**
- the class methods for accessing characteristics of the geometry:
  - **get_p(), get_label_t(), get_disks_t_1()**
- **CleanGeometry()** The function clears the list **disks_t_1()**.
- **InitialGeometry(unsigned int dim, unsigned int t, std::list<DiskDp> disks)** Initializing the time moment **label_t** as *t* and **disks_t_1()** as *disks* in the **Geometry 3**.
- **UpdateGeometry(DiskDp disk)** The function **Exclusion_disk(DiskDp disk)** implemented in the **RectDp** class is applied to the rectangle at the **rect_t** pointer for each disk of the list **disks_t_1()**.
- **EmptyGeometry()** The function checks the parameter **fl_empty**. If **fl_empty** is $false$ the geometry exists, otherwise - the geometry is empty.

Template <Class GeomX> Class OPDp

The class implements the FPOP-algorithm for different types of geometry **GeomX**.
Note:The geometry **GeomX** must have the following functions:

- **get_p(), get_label_t(), get_disks_t_1()**
- **CleanGeometry()**
- **EmptyGeometry()**
- **InitialGeometry(unsigned int dim, unsigned int t, std::list<DiskDp> disks)**
- **UpdateGeometry(DiskDp disk_t)**

The class characteristics:

- **p** is the value of dimension.
- **n** is the number of data points.
- **penalty** is a value of penalty (a non-negative real number).
- **sx12** are sum vectors $\sum_{i=0}^{t-1} x_i^k$, $\sum_{i=0}^{t-1}(x_i^k)^2$, $t = 0 : n-1$, $k = 0 : p-1$.
- **chpts** is the vector of changepoints.
- **means** is the list of successive means for data x.
- **globalCost** is the global cost.

The class implements:

- the constructor:
  - **OPDp<GeomX>(Rcpp::NumericMatrix x, double beta)**

$$penalty = beta;$$
$$p = (unsigned\ int)x.nrow();$$
$$n = (unsigned\ int)x.ncol();$$
$$memory\ for\ sx12.$$

- the constructor copy **OPDp<GeomX> (const OPDp<GeomX> &geomX)**
- the destructor **OPDp<GeomX>()**
- the class methods for accessing characteristics of class:
  - **get_p(), get_n(), get_penalty(), get_chpts(), get_means(), get_globalCost()**
- **vect_sx12(Rcpp::NumericMatrix x)** The method to find the sum vectors:
  $\sum_{i=0}^{t-1} x_i^k$, $\sum_{i=0}^{t-1}(x_i^k)^2$, $t = 0 : n-1$, $k = 0 : p-1$.
- **algoFPOP(Rcpp::NumericMatrix x, int type, bool test_mode)**
  The function implements the FPOP-algorithm with the different types of geometry <**GeomX**>.
  Currently we implemented the following types:
  - *type* = 1: Class **Geom1Dp** (Geometry 1);
  - *type* = 2: Class **Geom2Dp** (Geometry 2);
  - *type* = 3: Class **Geom3Dp** (Geometry 3).

Intersection_disk(DiskDp disk)

Description

The function approximates a rectangle and a circle intersection area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the intersection area of the rectangle and the circle.
If there is no intersection, the function makes at least one of the rectangle parameters satisfies the condition:

$$coordinates_{i,0} \geq coordinates_{i,1}, \quad i = 0 : p-1. \tag{3}$$

Input parameters:

- **disk** is the circle, the element of class **DiskDp**.

Algorithm:

Preprocessing

We define:

- the parameters of the circle **disk**:

$$\begin{aligned} c &= disk.get\_center(); \\ r &= disk.get\_radius(). \end{aligned} \tag{4}$$

- the point pnt_min is the point of the rectangle that is minimally distant from the center of the **disk**.
- vector dx_inter^2(6) is the values of a discriminant devided by 4 of the system (5) for all $k = 0 : p-1$.

$$\begin{cases} (x\_inter_k - c_k)^2 + \sum_{i=0, i\neq k}^{p-1} (x\_inter_i - c_i)^2 = r^2; \\ x\_inter_i = pnt\_min_i, \quad i = 0 : p-1, \quad i \neq k. \end{cases} \tag{5}$$

$$dx\_inter_k^2 = r^2 - \sum_{i=0, i\neq k}^{p-1} (pnt\_min_i - c_i)^2, \quad k = 0 : p-1. \tag{6}$$

If for each $k = 0 : p-1$ $dx\_inter_k^2$ is positive we define the characteristics of rectangle as:

$$\begin{cases} dx\_inter_k^2 > 0, \quad k = 0 : p-1; \\ x\_inter_{k0} = c_k - \sqrt{dx\_inter_k^2}; \\ x\_inter_{k1} = c_k + \sqrt{dx\_inter_k^2}. \end{cases} \tag{7}$$

$$\begin{aligned} coordinates_{k,0} &= \max\{coordinates_{k,0}, x\_inter_{k0}\}; \\ coordinates_{k,1} &= \min\{coordinates_{k,1}, x\_inter_{k1}\}. \end{aligned} \tag{8}$$

else (isn't intersection) we define the characteristics of rectangle as:

$$coordinates_{0,0} = coordinates_{0,1}. \tag{9}$$

Exclusion_disk(DiskDp disk)

Description

The function approximates a rectangle and a circle difference area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the difference area of the rectangle and the circle.
If the difference is the empty set, the function makes the rectangle with parameters that correspond to the condition (13).

Input parameters:

The input of this function consists:

- **disk** is the circle, the element of class **DiskDp**.

Algorithm:

Preprocessing

We define:

- the parameters of the circle **disk** as(4)
- the point pnt_max is the vertex of the rectangle that are maximally distant from the center of the **disk**.

For each $k = 0 : p - 1$:

- we calculate $dx\_excl_k^2$ (11)(the value of a discriminant devided by 4 of the system (10)):

$$\begin{cases} (x\_excl_k - c_k)^2 + \sum_{i=0, i\neq k}^{p-1} (x\_excl_i - c_i)^2 = r^2; \\ x\_excl_i = pnt\_max_i, \quad i = 0 : p-1, \quad i \neq k. \end{cases} \tag{10}$$

$$dx\_excl_k^2 = r^2 - \sum_{i=0, i\neq k}^{p-1} (pnt\_max_i - c_i)^2. \tag{11}$$

  If $dx\_excl_k^2$ is positive we find the intersection points with $p-1$-planes:

$$\begin{cases} dx\_excl_k^2 > 0; \\ x\_excl_{k0} = c_k + \sqrt{dx\_excl_k^2}; \\ x\_excl_{k1} = c_k - \sqrt{dx\_excl_k^2}. \end{cases} \tag{12}$$

  We define the characteristics of rectangle as:

  – if $pnt\_max_k = coordinates_{k,1}$:

$$coordinates_{k,0} = \max\{coordinates_{k,0}, x\_excl_{k0}\}. \tag{13}$$

  – if $pnt\_max_k = coordinates_{k,0}$:

$$coordinates_{k,1} = \min\{coordinates_{k,1}, x\_excl_{k1}\}. \tag{14}$$

IsEmpty_rect()

Description

The function checks the parameters of the rectangle. If the parameters are not correct, this rectangle is empty.

Output parameters:

The function returns a boolean value **true** if the rectangle is empty, and **false** if it is not empty.

Algorithm:

If at least one of the rectangle parameters satisfies the condition (13) this rectangle is empty and the function returns a boolean value **true**, else **false**.

algoFPOP(Rcpp::NumericMatrix x, int type, bool test_mode)

Description

The function implements the FPOP-algorithm with 3 types of geometry <**GeomX**>:

- *type* = 1: Class **Geom1Dp** (Geometry 1);
- *type* = 2: Class **Geom2Dp** (Geometry 2);
- *type* = 3: Class **Geom3Dp** (Geometry 3).

Input parameters:

- **x** is the matrix of data;
- **type** is the value defined the type of geometry;
- **test_mode** is the parameter for the test of candidates (by default, $false$).

Output parameters:

The function forms the vectors **chpts**, list of **means** and the value of **globalCost**.

Algorithm:

Preprocessing

We allocate the memory for:

- the vector *last_chpt* of best last changepoints;
- the matrix *last_mean* matrix of means for the best last changepoints;
- the vector m is the value of the sum optimal cost and penalty at the moment t, $t = 0 : n - 1$;
- the vector *mus* is the values of temporary means.

We define:

- $sx12 = vect\_sx12(x)$;

- $m[0] = 0$;
- $test\_file$ is the file for test results;
- $geom = GeomX(p)$;
- $disk = DiskDp(p)$;
- $cost = GausseCostDp(p)$;
- $list\_disk$ is a list of active disks for $t-1$ (for initial moment as $NULL$) .
- $list\_geom$ is a list of active geometries for t (for initial moment as $NULL$).

Processing

For each $t = 0 : n-1$:

- By default, the value of *cost* is the value of Gaussian cost for the time period $(t-1, t)$:

$$cost.InitialGausseCostDp(p, t, t, sx12[t], sx12[t+1], m[t]).$$

  We define:

  - $min\_val = cost.get\_min()$ is a minimum value for the *cost*.
  - $lbl = t$ is a best last position for t.
  - $mus = cost.get\_mu()$ vector temporary means of the interval (lbl, t).

  The first run: Searching of $m[t+1]$

  For each element of the list $list\_geom$:

  - We define:
    - u is the $label\_t$ of the current list element.
    - the value min_val for the interval (u, t).
    - the active disk for $t-1$ and add this disk to the $list\_disk$.
  - We choose the minimum among all found values min_val and define the values lbl and mus that correspond this minimum.
  - We put the value $min\_val + penalty$ to the vector m by the position $t+1$ and the corresponding lbl to the vector $last\_chpt$ by the position t and mus to the matrix $last\_mean$ by the row t.

  New geometry

  We clear the variable *geom* (if necessary), initialize it according to the values p, t, $list\_disk$ and clear the $list\_disk$. After that we add this element to the list $list\_geom$ (15).

$$\begin{gathered} geom.CleanGeometry(); \\ geom.InitialGeometry(p, t, list\_disk); \\ list\_disk.clear(); \\ list\_geom.push_back(geom). \end{gathered} \tag{15}$$

  The second run: Pruning

  For each element of the list $list\_geom$:

– We define lbl as $label\_t$ of the current list element.

– We initialize the Gaussien cost function $cost$ for the interval (lbl, t) as 16 and $r2$ is the radius to the second power of the new disk as 17.

$$cost.InitialGausseCostDp(p, lbl, t, sx12[lbl], sx12[t+1], m[lbl]); \tag{16}$$

$$r2 = \frac{m[t+1] - m[lbl] - cost.get\_coef\_Var()}{cost.get\_coef()}. \tag{17}$$

– PELT-pruning:
If $r2 \leq 0$ we remove this element of the $list\_geom$, else we initialize $disk$ as:

$$disk.InitialDiskDp(p, cost.get\_mu(), sqrt(r2)). \tag{18}$$

– FPOP-pruning:
We update the current list element using the function $UpdateGeometry(disk)$. If after updating current list element is empty we remove this element of $list\_geom$.

Output:

Knowing the values of vector $last\_chpt$, the matrix $last\_mean$ and vector m we forme the vector of $chpts$, the list of $means$ and the value of $globalCost$.