Package fpop2D

Description

fpop2D is an R package developed in Rcpp/C++ performing parametric changepoint detection in bivariate 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 FPOP method with two types of pruning: "intersection of sets" and "difference of intersection and union of sets". Also, the package implements the Pruned Exact Linear Time(PELT) method that is a direct part of the FPOP method.

Package structure

- Part R
  - fpop2D.R
    The file contains the implementation of the following functions:
    data_gen2D - generation of data of dimension 2 with a given values of means and changepoints
    plotFPOP2D - plot of data with a values of means and changepoints.
  - RcppExports.R
    The file contains the implementation of the function **FPOP2D** that calls the function **FPOP2D** implemented in C++.
- Part C++
  - Cost.h, Cost.cpp
    The files contain the implementation code for the class**Cost**.
  - Disk.h, Disk.cpp
    The files contain the implementation code for the class **Disk**.
  - Rect.h, Rect.cpp
    The files contain the implementation code for the class**Rect**.
  - Geom.h, Geom.cpp
    The files contain the implementation code for the class **Geom**.
  - OP.h, OP.cpp
    The files contain the implementation code for the class **OP**.
  - fpop2D.cpp
    The file contains the code of the function **FPOP2D** that implements the change-point detection in bivariate time-series using the Functional Pruning Optimal Partitioning algorithm.
  - RcppExports.cpp
    The file contains the code that exports data R/C ++ .

We consider $(y_1, y_2) \in (R^2)^n$ - bivariate time-series when $y_1 = (y_1^1, .., y_n^1)$ and $y_2 = (y_1^2, .., y_n^2)$ - two vectors of univariate data size n.
We use the Gaussian cost of the segmented bivariate data when m_t is the value of the optimal cost, $m_0 = -\beta$. The cost function has the form:

$$q_t^i(\theta_1, \theta_2) = m_{i-1} + \beta + (t-i+1)((\theta_1 - E[y_{i:t}^1])^2 + (\theta_2 - E[y_{i:t}^2])^2) + (t-i+1)(V(y_{i:t}^1) + V(y_{i:t}^2)), \qquad i = 1:t. \tag{1}$$

$$E[y_{i:t}^k] = \frac{1}{t-i+1} \sum_{l=i}^{t} (y_{i:t}^k), \qquad k = 1, 2. \tag{2}$$

$$V(y_{i:t}^k) = \frac{1}{t-i+1} \sum_{l=i}^{t} (y_{i:t}^k)^2 - \left(\frac{1}{t-i+1} \sum_{l=i}^{t} (y_{i:t}^k) \right)^2, \qquad k = 1, 2. \tag{3}$$

In the case of a Gaussian cost $q_t^i(\theta_1, \theta_2)$ is a paraboloid so that:

$$m_t = min_{i=1:t} \left\{ m_{i-1} + (t-i+1)(V(y_{i:t}^1) + V(y_{i:t}^2)) + \beta \right\}. \tag{4}$$

We introduce the notations:

$$\begin{aligned} mu1 &= E[y_{i:t}^1], \\ mu2 &= E[y_{i:t}^2], \\ coef &= (t-i+1), \\ mi\_1\_p &= m_{i-1} + \beta, \\ coef\_Var &= (t-i+1)(V(y_{i:t}^1) + V(y_{i:t}^2)). \end{aligned} \tag{5}$$

Then the cost function takes the form:

$$q_t^i(\theta_1, \theta_2) = mi\_1\_p + coef \cdot ((\theta_1 - mu1)^2 + (\theta_2 - mu1)^2) + coef\_Var, \qquad i = 1:t. \tag{6}$$

Class Cost

We define the class characteristics (5):

- **coef**, **coef_Var**, **mi_1_p**, **mu1**, **mu2**

The class implements constructor:

- **Cost(unsigned int i, unsigned int t, double** vectS, double mi_1pen)** is the cost function at the time $[i, t]$.

We define the class methods:

- **get_coef()**, **get_coef_Var()**, **get_mi_1_p()**, **get_mu1()**, **get_mu2()**

  We use these methods to access the characteristics of the class.

- **get_min()**

  We use this method to get the minimum value of the cost function (4).

Class Disk

A class element is a circle that is defined by the center coordinates and the radius.

- We define the class characteristics:
  - **center1, center2** is the center of the circle.
  - **radius** is the radius of the circle.
- The class implements two constructors:
  - **Disk()** All characteristics are equal zero by default.
  - **Disk(double c1, double c2, double r)**
    $(c1, c2)$ are the center coordinates and r is the radius of the circle.
- We define the class methods:
  - **get_center1()**, **get_center2()**, **get_radius()**
    We use these methods to access the class characteristics.

Class Rect

A class element is a rectangle that is defined by the coordinates of the lower left-hand corner and the upper-right hand corner.

- We define the class characteristics:
  - **rectx0,recty0** are coordinates of the lower left-hand corner.
  - **rectx1, recty1** are coordinates of the upper-right hand corner.
- The class implements two constructors:
  - **Rect()**:
    - $rectx0 = recty0 = -\inf$);
    - $rectx1 = recty1 = \inf$).
  - **Rect(double x0, double y0, double x1, double y1)**
    - $(x0, y0)$ is the the lower left-hand corner;
    - $(x1, y1)$ is the upper-right hand corner.
- We define the class methods:
  - **get_rectx0()**, **get_recty0()**, **get_rectx1()**, **get_recty1()**
    We use these methods to access the class characteristics.
  - **min_ab(double a, double b)**, **max_ab(double a, double b)**
    We use these methods to find the minimum and maximum of two numbers.
  - **intersection(Disk 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.
  - **difference(Disk 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 Geom

- We define the class characteristics:
  - **label_t** is the time moment.
  - **rect_t** is the rectangle, the element of class **Rect**.
  - **disks_t_1** is the list of active circles for the moment $t-1$.
- The class implements two constructors:
  - **Geom(double c1, double c2, double r, double t, std::list¡unsigned int¿ l_disk)**
    - $label\_t = t$;
    - $disks\_t\_1 = l\_disk$;
    - $rect\_t = Rect(c1 - r, c2 - r, c1 + r, c2 + r)$.
  - **Geom(double t, std::list¡unsigned int¿ l_disk)**
    - $label\_t = t$;
    - $disks\_t\_1 = l\_disk$;
    - $rect\_t = Rect()$.
- We define the class methods:
  - **get_label_t()**, **get_rect_t()**, **get_disks_t_1()**
  - **bool empty_set(Rect R)**
    The function checks the parameters of the rectangle **R**. If the parameters are not correct, this rectangle is empty.
  - **intersection_rect_disk(Disk disk)**
    The function calls the function **intersection(Disk disk)** implemented in the Rect class.
  - **difference_rect_disk(Disk disk)**
    The function calls the function **difference(Disk disk)** implemented in the Rect class.

Class OP

- We define the class characteristics:
  - **n** is the number of data points.
  - **penalty** is a value of penalty (a non-negative real number).
  - **sy12** are sum vectors $\sum_{k=1}^{t} y_k^1$, $\sum_{k=1}^{t} y_k^2$, $\sum_{k=1}^{t}(y_k^1)^2$, $\sum_{k=1}^{t}(y_k^2)^2$, $t = 1 : n$.
  - **m** is the vector of the optimal cost value + *penalty*.
  - **changepoints** is the vector of changepoints.
  - **means1** is the vector of successive means for data1.
  - **means2** is the vector of successive means for data2.
  - **globalCost** is the global cost.
  - **geom_activ** is the element of Geom class.
  - **list_geom** is the list of Geom class element.
  - **it_disk** is the iterator for **list_geom**.
  - **list_disk** is the list of active labels for the moment $t - 1$.
  - **it_disk** is the iterator for **list_disk**.
- The class implements the constructor:
  - **OP(std::vector<double> y1, std::vector<double> y2, double beta)**
    - $penalty = beta$;
    - $n = y1.size()$;
    - we allocate memory for $sy12, m$.
- We define the class methods:
  - **get_changepoints()**, **get_means1()**, **get_means2()**, **get_globalCost()**, **get_n()**, **get_sy12()**, **get_geom_activ()**, **get_it_list()**
    We use these methods to access the class characteristics.
  - **vect_sy12(std::vector<double> y1, std::vector<double> y2)**
    We use this method to find the sum vectors $\sum_{k=1}^{t} y_k^1$, $\sum_{k=1}^{t} y_k^2$, $\sum_{k=1}^{t}(y_k^1)^2$, $\sum_{k=1}^{t}(y_k^2)^2$, $t = 1 : n$.
  - **algoFPOP(std::vector<double> y1, std::vector<double> y2, int type)**
    The function implements the Functional Pruning Optimal Partitioning algorithm with two types of pruning: "intersection of sets"($type = 1$) and "difference of intersection and union of sets"($type = 2$). Also, the package implements the Pruned Exact Linear Time(PELT) method ($type = 0$).

void Rect::intersection(Disk 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 the rectangle with parameters that correspond to the condition:

$$(rectx0 \geq rectx1)||(recty0 \geq recty1). \tag{7}$$

Input parameters:

The input of this function:

- **disk** is the circle, the element of class **Disk** with characteristics:
  - **center1, center2** is the center of the circle;
  - **radius** is the radius of the circle.

  To access these characteristics we use the methods **get_center1()**, **get_center2()**, **get_radius()** implemented in the class **Disk**.

Algorithm:

Preprocessing

Using the methods **get_center1()**, **get_center2()**, **get_radius()** implemented in the class **Disk** we define the parameters of the circle **disk**:

$$\begin{aligned} c1 &= disk.get\_center1(), \\ c2 &= disk.get\_center2(), \\ r &= disk.get\_radius(). \end{aligned} \tag{8}$$

Approximation

We consider two directions and update the characteristics of rectangle:

- horizontal direction(the characteristics $recty0, recty1$):
  If $rectx0 \leq c1 \leq rectx1$ then

$$\begin{aligned} recty0 &= \max\{recty0, c2 - r\}, \\ recty1 &= \min\{recty1, c2 + r\}. \end{aligned} \tag{9}$$

  Otherwise, we consider the points of intersection of the circle with straight lines $x = rectx0$ and $x = rectx1$. The circle has two intersection points with a straight line if the discriminant is positive. We define the values dl, dr (12) as the value of a discriminant devided by 4 of each system (10, 11):

$$\begin{cases} (x - c1)^2 + (y - c2)^2 = r^2, \\ x = rectx0. \end{cases} \tag{10}$$

$$\begin{cases}(x-c1)^2+(y-c2)^2=r^2,\\ x=rectx1.\end{cases} \tag{11}$$

$$\begin{aligned} dl &= r^2-(rectx0-c1)^2,\\ dr &= r^2-(rectx1-c1)^2, \end{aligned} \tag{12}$$

Note: we define the default intersection points for the algorithm to work correctly as:

$$\begin{aligned} l1 = r1 &= \infty,\\ l2 = r2 &= -\infty. \end{aligned} \tag{13}$$

We check the sign of dl, dr and find the intersection points:

$$\begin{cases} dl > 0,\\ l1 = c2-\sqrt{dl},\\ l2 = c2+\sqrt{dl}. \end{cases} \tag{14}$$

$$\begin{cases} dr > 0,\\ r1 = c2-\sqrt{dr},\\ r2 = c2+\sqrt{dr}. \end{cases} \tag{15}$$

We define the characteristics of rectangle as:

$$\begin{aligned} recty0 &= \max\{recty0, \min\{l1, r1\}\},\\ recty1 &= \min\{recty1, \max\{l2, r2\}\}. \end{aligned} \tag{16}$$

- vertical direction (the characteristics $rectx0, rectx1$)

  If $recty0 \le c2 \le recty1$ then

$$\begin{aligned} rectx0 &= \max\{rectx0, c1-r\},\\ rectx1 &= \min\{rectx1, c1+r\}. \end{aligned} \tag{17}$$

Otherwise, we consider the points of intersection of the circle with straight lines $y = recty0$ and $y = recty1$. The circle has two intersection points with a straight line if the discriminant is positive. We define the values db, dt (20) as the value of a discriminant devided by 4 of each system (18, 19):

$$\begin{cases}(x-c1)^2+(y-c2)^2=r^2,\\ y=recty0.\end{cases} \tag{18}$$

$$\begin{cases}(x-c1)^2+(y-c2)^2=r^2,\\ y=recty1.\end{cases} \tag{19}$$

$$\begin{aligned} db &= r^2-(recty0-c2)^2,\\ dt &= r^2-(recty1-c2)^2. \end{aligned} \tag{20}$$

Note: we define the default intersection points for the algorithm to work correctly as:

$$\begin{aligned} b1 = t1 = \infty, \\ b2 = t2 = -\infty. \end{aligned} \tag{21}$$

We check the sign of db, dt and find the intersection points:

$$\begin{cases} db > 0, \\ b1 = c1 - \sqrt{db}, \\ b2 = c1 + \sqrt{db}. \end{cases} \tag{22}$$

$$\begin{cases} dt > 0, \\ t1 = c1 - \sqrt{dt}, \\ t2 = c1 + \sqrt{dt}. \end{cases} \tag{23}$$

We define the characteristics of rectangle as:

$$\begin{aligned} rectx0 = \max\{rectx0, \min\{b1, t1\}\}, \\ rectx1 = \min\{rectx1, \max\{b2, t2\}\}. \end{aligned} \tag{24}$$

- If all the values dl, dr, db, dt are non-positive, the rectangle has no intersections with the circle and we define the characteristics of rectangle as:

$$rectx0 = rectx1. \tag{25}$$

bool Geom::empty_set(Rect rect)

Description

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

Input parameters:

- **rect** is the rectangle, the element of class **Rect** with characteristics:
  - **rectx0,recty0** are coordinates of the bottom left corner;
  - **rectx1, recty1** are coordinates of the top right corner.

To access these characteristics we use the methods **get_rectx0()**, **get_recty0()**, **get_rectx1()**, **get_recty1()**, implemented in the class **Rect**.checks the parameters of the rectangle.

Output parameters:

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

Algorithm:

If the parameters of the rectangle correspond to the condition (7) this rectangle is empty and the function returns a boolean value **true**, else **false**.

void Rect::difference(Disk 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 (7).

Input parameters:

The input of this function consists:

- **disk** is the circle, the element of class **Disk** with characteristics:
  - **center1, center2** is the center and **radius** is the radius of the circle.

  To access these characteristics we use the methods **get_center1()**, **get_center2()**, **get_radius()** implemented in the class **Disk**.

Algorithm:

Preprocessing

We define:

- the parameters of the circle **disk** (8),
- the values dl, dr, db, dt (12,20).

Approximation

We consider two directions and update the characteristics of rectangle:

- horizontal direction(the characteristics $recty0, recty1$):

  We consider the points of intersection of the circle with straight lines $x = rectx0$ and $x = rectx1$. We update the characteristics $recty0, recty1$ if dl and dr (12) is positive.

  Note: we define the default intersection points for the algorithm to work correctly as (13).

  We check the sign of dl, dr and find the intersection points (14) and (15).

  We define the characteristics of rectangle as:

$$\begin{aligned} recty0 &= \max\{recty0, \min\{l2, r2\}\}, \\ recty1 &= \min\{recty1, \max\{l1, r1\}\}. \end{aligned} \tag{26}$$

- vertical direction (the characteristics $rectx0, rectx1$)

  We consider the points of intersection of the circle with straight lines $y = recty0$ and $y = recty1$. We update the characteristics $rectx0, rectx1$ if db and dt (20) is positive.

  Note: we define the default intersection points for the algorithm to work correctly as (21).

  We check the sign of db, dt and find the intersection points (22) and (23).

  We define the characteristics of rectangle as:

$$\begin{aligned} rectx0 &= \max\{rectx0, \min\{b2, t2\}\}, \\ rectx1 &= \min\{rectx1, \max\{b1, t1\}\}. \end{aligned} \tag{27}$$

void OP::algoFPOP(std::vector<double> y1, std::vector<double> y2, int type)

Description

The function implements FPOP method with two types of pruning: "intersection of sets"($type = 1$) and "difference of intersection and union of sets"($type = 2$) for bivariate time series. Also, the function implements PELT method ($type = 0$).

Input parameters:

- **y1** is the vector of data1 (a univariate time series).
- **y2** is the vector of data2 (a univariate time series).
- **type** is the value defined the type of pruning (0 = PELT, 1 = FPOP(intersection of sets), 2 = FPOP (difference of intersection and union sets)).

Output parameters:

The function forms the vectors **changepoints**, **means1, means2** and the value **globalCost**.

Algorithm:

Preprocessing

- We define:
  - $n = y1.size()$.
  - $sy12 = vect\_sy12(y1, y2)$.
  - $m[0] = 0$.
- We allocate memory for the matrix $last\_chpt\_mean$:
  - the vector of best last changepoints.
  - the vector of means for best last changepoints $y1$.
  - the vector of means for best last changepoints $y2$.

Processing

For each $t = 0, .., n-1$ we do:

- We define:
  - $min\_val = \inf$ is a value for searching of the cost function minimum.
  - lbl is a best last position for t.
  - $mean1$ is a value of mean of the interval (lbl, t) for $y1$.
  - $mean2$ is a value of mean of the interval (lbl, t) for $y2$.
- New D_t^t

  We create $geom\_activ$ for the point (y_t^1, y_t^2) of the bivariate time series as (28) and we add this element to the list $list\_geom$.

$$geom\_activ = Geom(t, list\_disk). \tag{28}$$

- We define $list\_activ\_disk$ is a list of active disks for t.
- The first run: Searching of $m[t+1]$

  For each element of the list $list\_geom$ we do:
  - We define u is the $label\_t$ of the current element.
  - We create $cost\_activ$ is the cost function for the interval (u, t).
  - We find the minimum value min_val of the cost function using the method **get_min()**.
- We choose the minimum among all found values min_val and we define the value lbl that correspond this minimum.
- We put the value $\min min\_val + penalty$ to the vector m by the position $t+1$.
- We put the values lbl, $mean1$ and $mean2$ that corresponds $m[t+1]$ to the matrix $last\_chpts$.
- The second run: Pruning

  For each element of the list $list\_geom$ we do:
  - We define the current element and his $label\_t$ as $geom\_activ$ and lbl .
  - We define and calculate the cost function for the interval (lbl, t) as 29 and $r2\_inter$ is the radius to the second power of the disk D^t_{lbl} as 30.

$$cost\_inter = Cost(lbl, t, sy12[lbl], sy12[t+1], m[lbl]); \tag{29}$$

$$r2\_inter = \frac{m[t+1] - m[lbl] - cost\_inter.get\_coef\_Var()}{cost\_inter.get\_coef()}. \tag{30}$$

  - If $type \geq 0$:

    PELT-pruning:

    If $r2\_inter \leq 0$ we delete this element of list $list\_geom$.
  - If $type \geq 1$ and $r2\_inter > 0$:

    We define $disk\_inter = Disk(cost\_inter.get\_mu1(), cost\_inter.get\_mu2(), \sqrt{r2\_inter})$.

    We update the rectangle $rect\_t$ of $geom\_activ$ using the function $intersection\_geom\_disk(disk\_inter)$.

    FPOP-pruning(intersection of sets):
    - If new rectangle $rect\_t$ is empty we delete this element of list $list\_geom$.
    - Else we add $label\_t$ of $geom\_activ$ to the list $list\_activ\_disk$.
  - If $type = 2$ and $rect\_t$ is not empty:

    We define the list $list\_disk$ as the list $disks\_t\_1$ of $geom\_activ$.

    For each element of $list\_disk$:
    - We define his value $*it\_disk$.
    - We define and calculate the cost function for the interval $(*it\_disk, lbl - 1)$ as 31 and $r2\_dif$ is the radius to the second power of the disk $D^{lbl-1}_{*it\_disk}$ as 32.

$$cost\_dif = Cost(*it\_disk, lbl - 1, sy12[*it\_disk], sy12[lbl], m[*it\_disk]); \tag{31}$$

$$r2\_dif = \frac{m[lbl] - m[*it\_disk] - cost\_dif.get\_coef\_Var()}{cost\_dif.get\_coef()}. \tag{32}$$

* If $r2\_dif > 0$:
  We define $disk\_dif = Disk(cost\_dif.get\_mu1(), cost\_dif.get\_mu2(), \sqrt{r2\_dif})$.
  We update the rectangle $rect\_t$ of $geom\_activ$ using the function $difference\_geom\_disk(disk\_dif)$.
  FPOP-pruning (difference of intersection and union of sets):
  If new rectangle $rect\_t$ is empty we delete this element of list $list\_geom$ and we finish to treat the $list\_disk$ elements.

– We define the elements of list $list\_disk$ as $list\_activ\_disk$. We will use this list $list\_disk$ as the value of list $disks\_t\_1$ to generate a new element of list $list\_geom$ in the next iteration.

Output:

Knowing the values of the matrix $last\_chpts$ and vector m we forme the vectors $changepoints$, $means1$, $means2$ and the value $globalCost$.