LibreriaCVR

Código Fuente: Doménica Barreiro

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 cvr.models.GeoData Class Reference

Inherited by cvr.models.TemporalGeoData.

Public Member Functions

- **GeoData** (double data, double accuracy)

Public Attributes

- double **data**
- double **accuracy**

3.1.1 Detailed Description

Modelo del objeto GeoData, estos objetos se utilizan como dato fundamental en el análisis de las ventanas deslizantes.

Atributos:

data - valor tipo double que representa un dato geográfico. Se utiliza para los cálculos internos de las ventanas deslizantes.

accuracy - valor tipo double que representa la calidad o precisión del dato. Debe ser un valor entre 0 y 1, siendo 0 la precisión mínima y 1 la precisión máxima.

Author

Domenica Barreiro

3.2 cvr.utmconversion.LLPoint Class Reference

Public Member Functions

- **LLPoint** (double latitud, double longitud)

Public Attributes

- double **latitud**
- double **longitud**

3.2.1 Detailed Description

Modelo del objeto LLPoint, estos objetos se utilizan para la representación de un punto geográfico con coordenadas Latitud y Longitud.

Author

Domenica Barreiro

3.3 cvr.main.PruebaDataProcessing Class Reference

Static Public Member Functions

- static void **main** (String[ ] args)

3.3.1 Detailed Description

Author

Domenica Barreiro

3.4 cvr.dataprocessing.RollingMetric Enum Reference

Public Attributes

- **MEAN**
- **MEDIAN**

3.4.1 Detailed Description

Enum RollingMetric contiene los valores de las métricas que se pueden utilizar para calcular una ventana deslizante.

Author

Domenica Barreiro

3.5 cvr.dataprocessing.RollingWindow Class Reference

Static Public Member Functions

- static double[ ] rolling (int window_k, int window_x, double min_accuracy, RollingMetric metric, Collection< GeoData > data)
- static double[ ] rollingTemporal (int time_interval, ChronoUnit time_unit, int window_x, double min_accuracy, RollingMetric metric, Collection< TemporalGeoData > data)

3.5.1 Detailed Description

Clase con métodos estáticos que realizan el análisis de las ventanas deslizantes. Existen dos implementaciones:

1. rolling: Ventana deslizante por número de datos.
2. rollingTemporal: Ventana deslizante temporal (umbral de tiempo).

Author

Domenica Barreiro

3.5.2 Member Function Documentation

3.5.2.1 rolling()

```
static double [] cvr.dataprocessing.RollingWindow.rolling (
            int window_k,
            int window_x,
            double min_accuracy,
            RollingMetric metric,
            Collection< GeoData > data )  [static]
```

Calcula una métrica y el error de datos utilizando ventanas deslizantes dado un número de datos por ventana.

Parameters

window_k	valor entero que representa el número de datos que debe tener una ventana para calcular las métricas.
window_x	valor entero que representa el número de ventanas que se utilizan para el cálculo del valor final.
min_accuracy	valor flotante que representa la precisión mínima que debe tener un dato para ser considerado válido dentro del análisis. Debe estar entre 0 y 1, siendo 0 la precisión mínima y 1 la precisión máxima.
metric	valor tipo RollingMetric que define cuál métrica se va a calcular en las ventanas.
data	debe ser una colección que implemente la interfaz Collection cuyo contenido sean elementos tipo GeoData pertenecientes a esta librería con los atributos: data, accuracy donde el campo accuracy debe ser un valor decimal entre 0 y 1 que represente la calidad o precisión del dato, siendo 0 la precisión mínima y 1 la precisión máxima.

Returns

arreglo de 3 números decimales con los siguientes campos: [promedio_final, error, codigo_salida] donde promedio_final es el promedio de los resultados obtenidos de las {window_x} ventanas cuya desviación estándar sea mínima, error es el promedio de las desviaciones estándar de las {window_x} ventanas escogidas, el código_salida indica si el algoritmo culminó correctamente o tuvo algún error.

Códigos de salida:
1 Algoritmo culminado sin errores.
-1 Si la colección {data} está vacía.
-2 Si el número asignado de elementos a una ventana {window_k}, es mayor que el número total de datos en la colección {data}.
-3 Si el argumento {min_accuracy} no está en el rango establecido [0,1].
-4 Si el número de ventanas a considerar para calcular el resultado final {window_x}, es mayor que el número de ventanas que se pueden formar de la colección {data} con {window_k} elementos.

3.5.2.2 rollingTemporal()

```
static double [] cvr.dataprocessing.RollingWindow.rollingTemporal (
            int time_interval,
            ChronoUnit time_unit,
            int window_x,
            double min_accuracy,
            RollingMetric metric,
            Collection< TemporalGeoData > data )  [static]
```

Calcula una métrica y el error de los datos utilizando ventanas deslizantes dado un umbral temporal.

Parameters

time_interval	valor entero que representa el valor de tiempo utilizado como umbral para la ventana. Debe ser mayor a cero.
time_unit	valor tipo ChronoUnit que indica la unidad de tiempo en el que se encuentra time_interval. Sólo se aceptan HOURS, MINUTES y SECONDS como unidades de tiempo válidas para el análisis.
window_x	valor entero que representa el número de ventanas que se utilizan para el cálculo del valor final.
min_accuracy	valor flotante que representa la precisión mínima que debe tener un dato para ser considerado válido dentro del análisis. Debe estar entre 0 y 1, siendo 0 la precisión mínima y 1 la precisión máxima.
metric	valor tipo RollingMetric que define cuál métrica se va a calcular en las ventanas.
data	debe ser un objeto que implemente la interfaz Collection cuyo contenido sean elemento tipo TemporalData pertenecientes a esta librería con los atributos: timestamp, data, accuracy donde accuracy debe ser un valor decimal entre 0 y 1 que represente la calidad o precisión del dato, siendo 0 la precisión mínima y 1 la precisión máxima.

Returns

arreglo de 3 números decimales con los siguientes campos: [promedio_final, error, codigo_salida] donde promedio_final es el promedio de los resultados obtenidos de las {window_x} ventanas cuya desviación estándar sea mínima, error es el promedio de las desviaciones estándar de las {window_x} ventanas escogidas, el código_salida indica si el algoritmo culminó correctamente o tuvo algún error.

Códigos de salida:
1 Algoritmo culminado sin errores.
-1 Si la colección {data} está vacía.
-2 Si el argumento {time_interval} no es mayor que cero.
-3 Si el argumento {min_accuracy} no está en el rango establecido [0,1].
-4 Si el argumento {time_unit} no corresponde a los valores aceptados en el algoritmo (HOURS, MINUTES, SECONDS).
-5 Si el número de ventanas a considerar para calcular el resultado final {window_x}, es mayor que el número de ventanas que se forman de la colección {data} con el umbral de tiempo establecido.
-6 Si hubieron ventanas cuyos datos temporales no cumplían con el umbral de tiempo requerido.

3.6 cvr.models.TemporalGeoData Class Reference

Inherits cvr.models.GeoData.

Public Member Functions

- **TemporalGeoData** (LocalDateTime timestamp, double data, double accuracy)

Public Attributes

- LocalDateTime **timestamp**

3.6.1 Detailed Description

Modelo del objeto TemporalGeoData, estos objetos se utilizan como dato fundamental en el análisis de las ventanas deslizantes temporales. Hereda de la clase GeoData.

Atributos:

timestamp - objeto tipo LocalDateTime que representa la marca temporal en la cual se ha recolectado este dato.

Author

Doménica Barreiro

3.7 cvr.dataprocessing.UtilsRW Class Reference

Static Public Member Functions

- static double getMovingMean (Queue< GeoData > window)
- static double getMovingMedian (Queue< GeoData > window)
- static double getMovingStd (Queue< GeoData > window)
- static double calculateMean (double numArray[ ])
- static int meetsTimeThreshold (long[ ] time_difference, int time, ChronoUnit unit)
- static long[ ] timestampDifference (LocalDateTime first, LocalDateTime current)

3.7.1 Detailed Description

Clase de métodos estáticos auxiliares que se utilizan en el análisis de las ventanas deslizantes.

Author

Domenica Barreiro

3.7.2 Member Function Documentation

3.7.2.1 calculateMean()

```
static double cvr.dataprocessing.UtilsRW.calculateMean (
            double numArray[] )  [static]
```

Calcula la media aritmética de un arreglo de datos tipo double.

Parameters

numArray	arreglo de datos tipo double al que se le calculará la media.

Returns

valor tipo double que representa la media de los datos del arreglo.

3.7.2.2 getMovingMean()

```
static double cvr.dataprocessing.UtilsRW.getMovingMean (
            Queue< GeoData > window )  [static]
```

Calcula la media aritmética de los datos recibidos por parámetro.

Parameters

window	colección tipo Queue de java que contiene objetos tipo GeoData de los que se obtendrá la media aritmética.

Returns

valor tipo double que representa la media aritmética de los datos de la colección. Se utiliza el campo {data} del objeto GeoData para realizar el cálculo.

3.7.2.3 getMovingMedian()

```
static double cvr.dataprocessing.UtilsRW.getMovingMedian (
            Queue< GeoData > window )  [static]
```

Calcula la mediana de los datos recibidos por parámetro.

Parameters

window	colección tipo Queue de java que contiene objetos tipo GeoData de los que se obtendrá la mediana.

Returns

valor tipo double que representa la mediana de los datos de la colección. Se utiliza el campo {data} del objeto GeoData para realizar el cálculo.

3.7.2.4 getMovingStd()

```
static double cvr.dataprocessing.UtilsRW.getMovingStd (
            Queue< GeoData > window )  [static]
```

Calcula la desviación estándar de los datos recibidos por parámetro.

Parameters

window	colección tipo Queue de java que contiene objetos tipo GeoData de los que se obtendrá la desviación estándar.

Returns

valor tipo double que representa la desviación estándar de los datos de la colección. Se utiliza el campo {data} del objeto GeoData para realizar el cálculo.

3.7.2.5 meetsTimeThreshold()

```
static int cvr.dataprocessing.UtilsRW.meetsTimeThreshold (
            long[] time_difference,
            int time,
            ChronoUnit unit )  [static]
```

Verifica que el intervalo de tiempo almacenado en el arreglo time_difference cumple con el umbral establecido en base a los parámetros time y unit.

Parameters

time_difference	arreglo de cuatro datos tipo long que representan un intervalo de tiempo de la siguiente forma: [dias, horas, minutos, segundos]
time	valor entero que representa el umbral de tiempo.
unit	objeto tipo ChronoUnit de la librería time.temporal de java, representa la unidad de tiempo del umbral (HOURS, MINUTES, SECONDS).

Returns

valor entero que indica si el intervalo de tiempo del arreglo time_difference cumple o no con el umbral de tiempo.
0: si el intervalo time_difference cumple con el umbral de tiempo en cantidad y unidad.
1: si el intervalo time_difference no cumple y supera el umbral de tiempo.
-1: si el intervalo time_difference no cumple y es menor que el umbral de tiempo.

3.7.2.6 timestampDifference()

```
static long [] cvr.dataprocessing.UtilsRW.timestampDifference (
            LocalDateTime first,
            LocalDateTime current )  [static]
```

Calcula la diferencia entre dos objetos LocalDateTime de la librería time de java, en cuatro unidades de tiempo: días, horas, minutos y segundos.

Parameters

first	objeto LocalDateTime utilizado como base para realizar el cálculo de la diferencia. Debe ser temporalmente menor que el parámetro current.
current	objeto LocalDateTime con el que se calcula la diferencia de tiempo en base al parámetro first. Debe ser temporalmente mayor que el parámetro first.

Returns

un arreglo de cuatro datos tipo long que representan la diferencia entre las dos fechas: [dias, horas, minutos, segundos].

3.8 cvr.utmconversion.UTMConversion Class Reference

Static Public Member Functions

- static UTMPoint **convertLatLonToUTM** (LLPoint point)
- static LLPoint **convertUTMToLatLon** (UTMPoint point)

3.8.1 Detailed Description

Clase con métodos estáticos que realizan la conversión de UTMPoint a LLPoint y viceversa.

Siendo UTMPoint una representación de coordenadas UTM y LLPoint la representación de coordenadas geográficas Latitud y Longitud.

Author

Domenica Barreiro

Index