Geodf

Mar 28, 2020

Contents

CHAPTER 1

Introduction

This library provides a `GeoDataFrame` object to manipulate geographical series, with a time dimension or not.

The dataframe behaves like the pandas ones in python. In addition the geodf dataframes have feature columns: columns that the function and units are know: geometry, time, speed. This make it possible for the dataframe to provide more information out of the box from these columns.

CHAPTER 2

Columns

A dataframe has columns. At minimum a geometry column.

2.1 The column object

2.1.1 GeoDataFrameColumn

Properties:

name `String`: the name of the column

dtype `GeoDataFrameColumnType`: the internal type of the column: see the types section for details

type `Type`: the dart type the column's data

indice `int`: the column indice for the data. Mostly for internal usage

2.2 Constructors

2.2.1 GeoDataFrameColumn.fromGeoJsonGeometry

Build a column from parsed geojson objects

Positional parameters:

geometry `dynamic`: a geojson geometry: either a GeoJsonPoint , a GeoJsonMultiPoint or a GeoJsonLine

columnName `String`: the name of the column

```
final col = GeoDataFrameColumn.fromGeoJsonGeometry(someGeoJsonObject, "myline");
print(col.dtype);
print(col.type);
```

2.2.2 GeoDataFrameColumn.inferFromDataPoint

Build a column with types infered from a datapoint

Positional parameters:

dataPoint `dynamic`: a sample of the data to check to infer types

columnName `String`: the name of the column

CHAPTER 3

Feature columns

There are special columns representing features. They are declared at dataframe creation time. They are used internaly to make calculations and provide info in computed properties or methods. Feature columns:

- **geometry** declares a geometry. Data types: GeoPoint or GeoSerie
- **time** the date column. Data type: `DateTime`. Note: this column is created from timestamps
- **speed** the speed column: unit is meters per second. Data type: `double`

Only the geometry column is required. The others are optional. Example:

```
final data = <Map<String, dynamic>>[
   <String,dynamic>{
      "geometry": GeoPoint(latitude: 51.0, longitude: 0.0),
      "timestamp": 125855222,
      "speed": 0.0
   },
];
final df = GeoDataFrame.fromRecords(data,
     geometryCol: "geometry",
     speedCol: "speed",
     timestampCol: "timestamp");
```

3.1 `featureCols`

Get feature columns info:

```
final fcols = df.featureCols();
final timeCol = df.fcols.timeCol;
```

Shortcuts to get feature columns info:

```
final timeCol = df.timeCol;
final geomCol = df.geometryCol;
final speedCol = df.speedCol;
// internal type
print(speedCol.dtype);
// dart type
print(speedCol.type);
```

CHAPTER 4

Types

4.1 GeoDataFrameColumnType

Represents the internal type of a column:

- **categorical** A `String` for data representing categorical information
- **numeric** Either a `double` of an `int`
- **time** A `DateTime` object
- **geometry** GeoPoint or GeoSerie

4.2 TimestampType

The time unit of a timestamp

- **seconds** Seconds since epoch
- **milliseconds** Milliseconds since epoch
- **microseconds** Microseconds since epoch

CHAPTER 5

Create

5.1 From a list of dictionnaries

Create a dataframe from a list of dictionnaries

`GeoDataFrame.fromRecords`

Named parameters:

geometryCol **required** `String`: the geometry column name. The data must be of type GeoPoint or GeoSerie

speedCol `String`: the speed column name. The data must be of type `double`

timestampCol `String`: the timestamp column name. The data must be of type `int`

timestampFormat `TimestampType`: the type of timestamp **default** `TimestampType.milliseconds`

verbose `bool`: verbosity **default** `false`

```
final data = <Map<String, dynamic>>[
   <String,dynamic>{
      "geometry": GeoPoint(latitude: 51.0, longitude: 0.0),
      "timestamp": 125855222,
      "speed": 0.0,
      "altitude": 100.0,
      "foo": 30,
      "bar": 10.0
   },
];
final df = GeoDataFrame.fromRecords(data,
     geometryCol: "geometry",
     speedCol: "speed",
     timestampCol: "timestamp");
```

5.2 From a geojson file

Create a dataframe from a geojson file

Important: supported geojson features are: Point, Line, MultiPoint

`GeoDataFrame.fromGeoJsonFile`

Parameters:

path **required** `String`: the path to the geojson file

Named parameters:

timestampProperty `String`: the geojson property with a timestamp **default**: `timestamp`

speedProperty `String`: the geojson property for speed **default**: `speed`

timestampFormat `TimestampType`: the type of the timestamp **default**: `TimestampType.milliseconds`

verbose `bool`: verbosity **default** `false`

```
final df = await GeoDataFrame.fromGeoJsonFile("data/positions.geojson");
```

5.3 With random data

Create a dataframe filled with random data

`GeoDataFrame.random`

Named parameters:

distance `double`: the distance between points **default** `10.0`

speed `double` speed of each point. If not provided it will be randomized

timeInterval *Duration* time between each point **default** `Duration(seconds: 10)`

bearing `double` bearing of each point **default** `142.0`

startLatitude `double` latitude of the first geopoint **default** `51.0`

startLongitude `double` longitude of the first geopoint **default** `0.0`

numRecords `int` number of records **default** `100`

verbose `bool`: verbosity **default** `false`

```
final df = GeoDataFrame.random();
```

CHAPTER 6

Operations

For all standard operations like select, count, sort check the documentation of the Df package

CHAPTER 7

Computed properties

These properties are calculated from the feature columns

7.1 Speed

avgSpeed `double`: the average speed

avgSpeedWhenMoving `double`: the average speed when moving

avgSpeedKmhRounded `double`: the average speed in kilometers per hour rounded

avgMovingSpeedKmhRounded `double`: the average speed when moving in kilometers per hour rounded

7.2 Distance

distance `double`: the total distance in meters

distanceKmRounded `double`: the total distance in kilometers