\*\*\*This is the user guide for the R package "research"\*\*\*

AUTHOR: Javier Castillo ISC 2012 - 5708 PUCMM

INDEX:

1. DESCRIPTION
2. TARGET OS
3. REQUIREMENTS/DEPENDENCIES
   1. NOTE
4. INSTALLATION
   1. IF PLACED IN CUSTOM DIRECTORY
5. SCRIPTS
6. CSV

DESCRIPTION:

This package is intended to provide support for scientific investigation concerning the way cane leaves capture water when it's raining.

TARGET OS:

This package was developed in Windows 10 and it has only been tested in this platform.

REQUIREMENTS/DEPENDENCIES:

These are this package dependencies. It is required that all dependencies are installed in order for the package to work properly.

- RStudio 1.0.136 or higher: This is the recommended working environment for using the "research" package.

- MongoDB 3.4 of higher: This package uses MongoDB as its database manager to permanently stored the data from the scientific research and its results.

- devtools 1.12.0: This R package enables the "research" package to be loaded into the R session without it being installed.

- roxygen2 6.0.1: This R package enables a special in-line documentation for R scripts.

- mongolite 1.1: This R package enables the connection between the "research" package and the MongoDB database.

- deSolve 1.14: This R package is used to calculate the experimental model using the ODE function.

NOTE:

It is advice to install MongoDB and the configure it as a Windows Service in order for it to start when the operating system boots up. For information in how to do follow the instructions in this page listed as "Configure a Windows Service for MongoDB Community Edition". (https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/)

INSTALLATION:

This package doesn't require installation. This are the steps on how to set it up. These steps assume that all the packages and programs listed above are already installed in the working machine.

1. Place the folder "research" into the Documents directory in windows. Alternatively, you can place the folder within another directory of your preference, but it will require another step (see IF PLACED IN CUSTOM DIRECTORY).

2. Load the project using the reseach.Rproj file, placed within the research folder. This will open a new session on RStudio and load all dependencies including the package itself.

IF PLACED IN CUSTOM DIRECTORY:

As it is said in step 2, when the project is open with RStudio it loads all dependencies and the package itself. The way it does it is by sourcing the .Rprofile when the project is opening. If you place the "research" folder within another directory other than "Documents" (or My Documents), then will have to either source() the package from within the R session or change the .Rprofile file.

These are the steps on how to change the .Rprofile file.

1. Look for the “. Rprofile" file in the "research" folder.

2. Open it with a code editor like Sublime Text, VS Code, etc.

3. Look for the function ".first". This is the function responsible for loading the libraries that the package is going to use.

4. Within the ".first" function, look for the statement "load\_all("~\\research\\R")". This is what you are going to change.

5. Change the path inside to where the folder actually is. Be sure to keep the [\\research\\R](file:///\\research\\R) part because this is what it's going to be loaded.

6. After changing the file, save it and load the project again in RStudio.

SCRIPTS

1. export.R: This script contains functions related to the export of data to external CSV files.
   1. exportDatabaseToCSV: Exports the whole experimental collection from the research database into a CSV file.

**Note**: The CSV file will be saved by default on the working directory of the project.

* 1. exportDataFrameToCSV: Exports a data frame into a specified CSV file.

**Params**:

* + - *df*: data frame to be exported.
    - *filenam*e: Name of the file in which the data frame will be inserted.

**Note:** *filename* can also receive a relative path to where the CSV file will be save. Keep in mind that the path should and with <name-of-file>.csv for the function to recognize it as a valid file name. In case the just <name-of-file>.csv is provided with no explicit path to where the file should be saved, the function will save it into the project's working directory.

1. filter.R: This script contains all functions related to database queries and data filtering. All these functions assume that the data stored in the database has the format and field names as specified in the CSV section of this document.
   1. connect: Connects to the experimental collection in the research database.

**Returns**: A function to query from the connected collection.

* 1. connectTo: Connects to the specified collection in the specified database.

**Params**:

* + - *from:* (string) Database name.
    - *to:* (string) Collection name, must exist in the database to be able to query it.

**Returns:** A function to query from the connected collection.

* 1. getSessionByLote: Returns all experimental sessions related to the loteID provided

**Params**:

* + - *loteID:* Identifier of the lote as specified in the CSV file.

**Returns:** values Returned: dataID, windDirection, windSpeed, temperature, humidity, startDate, endDate

* 1. getSessionByDate: Returns all experimental sessions within the date range provided.

**Params**:

* + - *sd:* Starting date of the search range.
    - *Ed:* Ending date of the search range.

**Note**: The date range is inclusive. If any of the parameters ISN't going to be used, specify

it as NULL. Otherwise there'll be an error.

**Returns:** Values Returned: dataID, windDirection, windSpeed, temperature, humidity, startDate, endDate

* 1. getSessionByID: Returns the experimental session associated with the ID provided.

**Params:**

* + - *sessionID*: Identifier of the session as specified in the CSV as the dataID field

**Returns:** Values Returned: dataID, windDirection, windSpeed, temperature, humidity, startDate, endDate

* 1. getSessionByTemp: Returns all experimental sessions within the temperature range specified.

**Params:**

* + - *st*: Minimum value of the search range.
    - *et*: Maximum value of the search range.

**Note:** The temperature range is inclusive. If any of the parameters ISN't going to be used, specify it as NULL. Otherwise there'll be an error.

**Returns:** Values Returned: dataID, windDirection, windSpeed, temperature, humidity, startDate, endDate

* 1. getSessionByHum: Returns all experimental sessions within the humidity range specified.

**Params:**

* + - *sh*: Minimum value of the search range.
    - *eh*: Maximum value of the search range.

**Note:** The humidity range is inclusive. If any of the parameters ISN't going to be used, specify it as NULL. Otherwise there'll be an error.

**Returns:** Values Returned: dataID, windDirection, windSpeed, temperature, humidity, startDate, endDate

* 1. getPlantByLote: Returns all the plantones associated with the specified loteID.

**Params:**

* + - *loteID*: Identifier of the lote as specified in the CSV.

**Returns:** Returned Values: plantonID, plantonLongitude, plantonLatitude, plantonBordeBit

* 1. getPlantCapPoint: Returns all the capture points associated with the loteID and plantonID specified.

**Params:**

* + - *loteID*: Identifier of the lote as specified in the CSV.
    - *platonID*: Identifier of the planton associated to the loteID

**Returns:** Returned Values: puntoCaptID, puntoCaptLatitud, puntoCaptLongitud, puntoCaptAlturaSurco, puntoCaptAmplitudSurco, puntoCaptRadioEfectivo

* 1. getCapPointByLote: Returns all capture points within the specified lote.

**Params:**

* + - *loteID*: Identifier of the lote as specified in the CSV.

**Returns:** Returned Values: puntoCaptID, puntoCaptLatitud, puntoCaptLongitud, puntoCaptAlturaSurco, puntoCaptAmplitudSurco, puntoCaptRadioEfectivo

* 1. getCapPointByExp: Returns all capture points associated with the specified sessionID.

**Params:**

* + - *sessionID*: Identifier of the session as specified in the CSV as the dataID field

**Returns:** Returned Values: puntoCaptID, puntoCaptLatitud, puntoCaptLongitud, puntoCaptAlturaSurco, puntoCaptAmplitudSurco, puntoCaptRadioEfectivo

* 1. getLoteByDate: Returns all lotes within the specified plantation date range.

**Params:**

* + - *sd*: Starting date of the search range.
    - *ed*: Ending date of the search range.

**Note:** The date range is inclusive. If any of the parameters ISN't going to be used, specify it as NULL. Otherwise there'll be an error.

**Returns:** values Returned: loteID, lotePlantationDate, loteLatitude, loteLongitude

* 1. getLoteByExp: Returns the specific lotes associated with the specified sessionID.

**Params**:

* + - *sessionID:* Identifier of the session as specified in the CSV as the dataID field

**Returns**: Values Returned: loteID, lotePlantationDate, loteLatitude, loteLongitude

* 1. getLamByCapPoint: Returns all laminas capture in the specified capture point in different experimental sessions.

**Params**:

* + - *capPointID:* Identifier of the capture point the laminas are associated to.

**Returns**: Returned Values: laminaID, laminaCaptureMoment, laminaPrecipitacion, fechaCaptLamina

* 1. getLamByCapHour: Returns all capture points within a specified hour-of-capture range from different experimental sessions.

**Params**:

* + - *sh*: Minimum value of the search range.
    - *eh*: Maximum value of the search range.

**Note**: The hour range is inclusive. If any of the parameters ISN't going to be used, specify it as NULL. Otherwise there'll be an error.

**Return**: Returned Values: laminaID, laminaCaptureMoment, laminaPrecipitacion, fechaCaptLamina

1. import.R: This script contains all functions to insert data into the MongoDB database from CSV files. Although these functions can receive and insert any kind of CSV, with any kind of information, into the database; it is advice that the CSV inserted follows the guideline from the CSV section of this document.
   1. importIntoDatabase: Imports from a CSV file to the mongo database.

**Params**:

* + - *filepath:* Absolute path of for the CSV file containing the data to be inserted.
  1. importIntoCollection: Import a dataset into a specific collection.

**Params**:

* + - *colname*: Name of the collection.
    - *filepath:* Absolute path of the CSV file containing the data.
  1. importAll: Imports all CSV files from a given directory into the database.

**Params**:

* + - *dirpath*: Absolute path to the directory containing the CSV files.

1. model.R: This script contains all functions related to the experimental model as specified here: <http://doc.orizondo.org/isc581/modelo.html>.
   1. model: Represents the functional dependency of the experimental model. It uses the 'euler' method and the ODE function from the DeSolve package to generate the output.

**Params:**

* + - *df*: Data frame containing the data capture during the experimental session.
    - *Aef:* Effective section of the discharge.
    - *Aint:* Effective interception area.
    - *Hm*: Maximum lamina the produces a discharge.
    - *Hd*: Value of the lamina that triggers the discharge in the container.

**Note**: The parameter *df* must contain the columns "Tiempo" and "Descarga", each containing time values and their respective water discharge quantities.

**Returns**: A DeSolve object containing the columns 't', 'h(t)', 'Qd(t)', which correspond to the times, behavior of the laminas in time t, and functional dependency of the discharge time t, respectively.

* 1. optimizeQ: This function is intended to optimize the functional dependency of the discharge and the time t using the squared deviation.

**Params**:

* + - *Aef:* Effective section of the discharge.
    - *Aint:* Effective interception area.
    - *Hm*: Maximum lamina the produces a discharge.
    - *Hd*: Value of the lamina that triggers the discharge in the container.

**Note**: The function returned automatically makes a plot comparing the observed data with the estimate data calculated by the model function.

**Returns**: A function with the entry parameter being a data frame as specified in the model function.

CSV

These are the guidelines on how the data must be structured inside the CSV files so that the filter.R functions work properly and maintain consistency within the database.

First there’s the CSV containing all the data concerning the experiment itself, this CSV file can have any name, but it is advice to put it a representative name. The following is the first line of the CSV file which R will interpret as the column names for the data inside:

dataID,loteID,lotePlantationDate,loteLatitude,loteLongitude,plantonID,plantonLongitude,\

plantonLatitude,plantonBordeBit,puntoCaptID,puntoCaptLatitud,puntoCaptLongitud,\

puntoCaptAlturaSurco,puntoCaptAmplitudSurco,puntoCaptRadioEfectivo,laminaID,\

laminaCaptureMoment,laminaPrecipitacion,fechaCaptLamina,windDirection,windSpeed,\

temperature,humidity,startDate,endDate

It is intended to be a single line, but due to its length it had to be cut out with the character ‘\’ representing it continuation in the line bellow.

These are the details of every field in the first line of the CSV containing the experiment’s data:

1. **dataID**: This is the identifier of the experimental session. Each line being that belongs to the same experimental session MUST contain the same value for this field.
2. **loteID**: This is the identifier of the lotes. Each line that contains information about the same lote MUST have the same value for this field.
3. **lotePlantationDate**: This is the date the lote was formed. The expected date format is (mm/dd/yyy).
4. **loteLatitude**: This is the latitude of the GPS coordinates designating the position of the lote.
5. **loteLongitude**: This is the longitude of the GPS coordinates designating the position of the lote.
6. **plantonID**: This is the identifier of the planton. Each line that contains information about the same planton MUST have the same value for this field.
7. **plantonLongitude:** This is the longitude of the GPS coordinates designating the position of the planton.
8. **plantonLatitude:** This is the latitude of the GPS coordinates designating the position of the planton.
9. **plantonBordeBit:** This field specifies if the planton on the edge of the lote, thus delimiting the lote. The values expected in this field are TRUE or FALSE.
10. **puntoCaptID:** This is the identifier of the capture point. Each line that contains information about the same capture point MUST have the same value for this field.
11. **puntoCaptLatitud:** This is the latitude of the GPS coordinates designating the position of the capture point.
12. **puntoCaptLongitud:** This is the longitude of the GPS coordinates designating the position of the capture point.
13. **puntoCaptAlturaSurco:** This is the height of the at which the capture point is placed.
14. **puntoCaptAmplitudSurco:** This is the width of the space between planton at which the capture point is placed.
15. **puntoCaptRadioEfectivo:** This is the effective radius of the capture point.
16. **laminaID:** This is the identifier of the captured lamina from a capture point during the experimental session.
17. **laminaCaptureMoment:** This is the instant at which the lamina was captured during the experimental session. The expected format is (hh:mm:ss).
18. **laminaPrecipitacion:** This is the captured value of the lamina.
19. **fechaCaptLamina:** This is the date the lamina was captured. The expected date format is (mm/dd/yyy).
20. **windDirection:** This is the direction of the wind during a certain experimental session. The values expected are: N, S, E, W, NE, NW, SE, SW, NNW, Etc.
21. **windSpeed:** This is the value of the speed of the wind during a certain experimental session.
22. **temperature:** This is the value of the temperature during a certain experimental session.
23. **humidity:** This is the value of the humidity during a certain experimental session.
24. **startDate:** This is the starting date of the experiment. The expected date format is (mm/dd/yyy).
25. **endDate:** This is the finishing date of the experiment. The expected date format is (mm/dd/yyy).

Lastly, there’s the CSV file containing the data that is going to be processed using the experimental model. This is the structure that the data must have for the model to work:

Tiempo, Descarga

These are the details of every field in the first line of the CSV containing the data to processed:

1. **Tiempo:** This is the value in ***seconds*** *of the moment the lamina was captured in the capture point.*
2. **Descarga:** This is the value of the lamina or discharge captured.

**Note:** Each line of this CSV is expected to be a pair of the capture time (in seconds) and its respective capture value or discharge.