Software for analysis of visual meteor data

Kristina Veljković, Ilija Ivanović

Petnica Meteor Group, Valjevo, Serbia
mackikac@gmail.com, ilija91ivanovic@gmail.com

In this paper, we will present new software for analysis of IMO data collected from visual observations. The software consists of a package of functions written in the statistical programming language *R*, as well as *Java* application which uses these functions in a user friendly environment. *R* code contains various filters for selection of data, methods for calculation of Zenithal Hourly Rate (ZHR), solar longitude, population index and graphical representation of ZHR and distribution of observed magnitudes. *Java* application allows everyone to use these functions without any knowledge of *R*. Both *R* code and *Java* application are open source and free with provided user manuals and examples.

1 Introduction

This paper presents results of software development for analysis of IMO visual meteor data. The software consists of a package of functions written in the statistical programming language *R*, as well as *Java* application. The purpose of *R* functions is to provide basic analysis, and *Java* application is developed with the aim to make use of this functions in a user friendly environment.

The *R* package *MetFns* contains data frames with visual meteor data (rate or magnitude data), as well as various filters for the selection of the data, methods for calculation of Zenithal Hourly Rate (ZHR), solar longitude, population index and a graphical representation of the ZHRs and the observed magnitude distributions.

The developed *Java* application allows users to call *R* functions without any knowledge about the *R* programming language. Although its purpose is to be proxy for these functions, the application contains a few extra features which can be useful to users. The application uses a standard graphical interface, and it contains help files from the *R* package.

All software is open source and free, with manuals and examples provided for both software packages.

The rest of the paper is organized as follows. In Section 2, we provide details about the installation of package *MetFns* and the application *MetRApp*. The description of the *R* package and the *Java* application is given in Sections 3 and 4, respectively. Finally, conclusions are drawn in Section 5.

2 Installation

In order to install the *R MetFns* package, follow the next steps:

1. (if not already installed) download and install the latest version of *R*[1]
2. Download the packages *astroFns*[2] and *plotrix*[3]
3. Download the package *MetFns*[4]

In order to install the *MetRApp* application:

1. (if not already installed) download and install latest *JRE*[5]
2. Download and install the *Runiversal* package from *CRAN*[6]
3. Download the application[7]

Please note: In our software we use many dependencies developed by third party organizations. We are not responsible for that content. Also, please note that as new versions of those dependencies are developed, and you use them, they may not be compatible with our software. We will try to keep up with new versions of these dependencies, but if you encounter difficulties, please contact us.

3 R package

The *R* package *MetFns* consists of data frames containing visual meteor data and functions which manipulate these data. Data frames can be divided into three sections, by their type:

a. Yearly rate data named *rateXX*, where XX represents the two last digits of the year

[1] cran.r-project.org
[2] cran.r-project.org/web/packages/astroFns/
[3] cran.r-project.org/web/packages/plotrix
[4] cran.r-project.org/web/packages/MetFns/
[5] http://www.oracle.com/technetwork/java/javase/downloads/index.html?ssSourceSiteId=ocomen
[6] http://cran.r-project.org/web/packages/Runiversal/
[7] https://bitbucket.org/ivail/metrapp (alternatively it may be hosted on www.meteori.rs)

b. Yearly magnitude data named *magnXX*

c. Accompanying data includes data frames *radiant* with coordinates of shower radiants throughout the year, *shw_list*, *vmdbpers* and *vmdbsite* with a list of observed meteor showers, observers and observing sites, respectively.

Functions that manipulate visual meteor data can be divided into four types:

- Functions that read rate or magnitude data from the IMO site or file saved on a computer, named *read.rate(data)* and *read.magn (data)*
- Functions that select (filter) data by one or more criteria
- Functions that perform some calculations over data
- Functions that draw graphics with the data

Next, we will cover the last three types of functions in more detail.

Filter functions

The Package *MetFns* contains 13 individual filter functions and a global filter. Some filters can be used only on rate data and it will be specified in the description of the filter.

In the following examples, we suppose that all rate and magnitude data are previously loaded (using *data* function, for example data (rate00))

- *filter.shw (data, shw)* selects data for a given visual meteor dataset and specified shower code. For example, if we want to select data for the Perseids from the rate data for the year 2000, we would call the function *filter.shw* in the following way

  ```
  filter.shw (rate00, shw="PER")
  ```

  If we wish to do the same selection for the magnitude data, we would type in the *R* console

  ```
  filter.shw (magn00, shw="PER")
  ```

- *filter.date (data, year, month, day.beg, day.end=day.beg)* selects data for a given visual meteor dataset and specified year, month and day (or days). By default, the argument *day.end* (ending day) is set to be equal to *day.beg* (beginning day). So, if the argument *day.end* is not provided, the function *filter.date* selects data for a given date, otherwise it selects data for a period of days, limited by *day.beg* and *day.end*.

  The day given in meteor datasets corresponds to the beginning of the observing time period. For the selection of the data, the day corresponding to the middle of the observing time period is used. For example, to select rate data for the period from 5–15 August 2007, we would type

  ```
  filter.date (rate07, year=2007,
  month=8, day.beg=5, day.end=15)
  ```

  In a similar way, we would select the magnitude data.

- *filter.time (data, time.low, time.up) s*elects data for a given visual meteor dataset and specified time period. Arguments *time.low* and *time.up* are in the format 0-2359 specifying, the lower and upper boundary of time in hours and minutes, respectively.
- *filter.imocode (data, imocode)* selects data for a given visual meteor dataset and specified IMO observer code.
- *filter.obsname (data, name, name)* selects data for a given visual meteor dataset and specified observer's first and last name. It can be used when one is not certain of the IMO observer code (due to possible non-uniqueness of the five letter combination).
- *filter.gc (data, long.low=0, long.up=180, ew=c ("E","W"), lat.low=0, lat.up=90, ns=c ("N","S"))* selects data for a given visual meteor dataset and specified geographical coordinates of the observing site or interval of geographical coordinates. The arguments *long.low* and *long.up* represent, respectively, the lower and upper boundary of longitude and *lat.low* and *lat.up* are, respectively, the lower and upper boundary of latitude.

  If the values of the arguments *long.low* and *long.up*, as well as *lat.low* and *lat.up*, are the same, *filter.gc* selects data for a particular observing site. This filter enables one to select data only by longitude or latitude, with the geographical coordinates being between given boundaries, less, greater or equal to a boundary.

  For example, if we wish to select magnitude data for the year 2004 for a site with longitude 19.7E and latitude 44.2N, we would type into the *R* console.

  ```
  filter.gc (magn04, long.low=19.7,
  long.up =19.7, ew="E", lat.low=44.2,
  lat.up=44.2, ns="N")
  ```

- *filter.site (data, site)* selects data for a given visual meteor dataset and specified observing site. In order to use this filter, the argument *data* has to consist of the column named *"sitecode"*.
- *filter.country (data, country)* selects data for a given visual meteor dataset and specified country. The data selection is performed using *filter.site* which filters data by codes of all sites belonging to the specified country. As for the function *filter.site*, data has to consist of the column named *"sitecode"*.
- *filter.sol (data, sol.low=0, sol.up=359.999)* selects data for a given visual meteor dataset and specified solar longitude or interval of solar longitudes.
- *filter.F (data, F.low=1, F.up=3)* selects data for a given visual meteor rate dataset and specified correction factor

or interval of correction factor for clouds. Arguments *F.low* and *F.up* represent, respectively, the lower and the upper boundary for the correction for clouds.

- *filter.mag (data, mag.low=2.0, mag.up=7.5)* selects data for a given visual meteor dataset and specified limiting magnitude or interval of magnitudes. The arguments *mag.low* and *mag.up* are, respectively, the lower and the upper boundary of the limiting magnitude.

- *filter.h (data, shw, Ralpha=NULL, Delta=NULL, h.low=10, h.up=90)* selects data for a given visual meteor dataset, specified shower and its radiant elevation or interval of radiant elevations. The arguments *h.low* and *h.up* specify, respectively, the lower and the upper boundary of the radiant elevation.

- *filter.totcor (data, shw, Ralpha=NULL, Delta=NULL, r, C=5)* selects data for a given visual meteor rate dataset, specified shower, population index and correction factor. The correction factor is equal to (Rendtel and Arlt, 2008)

$$C = \frac{r^{6.5-lmg} F}{\sin(h)},$$

where *r* is the population index, *lmg* the limiting magnitude, *F* the correction factor for clouds and *h* the radiant elevation. One needs to specify the maximum value of the correction factor C (default value of C is 5).

- The function *filter* performs various data selections for a given visual meteor data. It is a wrapper function for all previously mentioned filters.

  For example, if we want to select rate data for observations of the Perseids in Serbia, time period 5–15th August 2007, limiting magnitudes of 5.5 or higher and a total correction factor less than 5, we would use

```
filter (rate07, shw="PER", year=2007,
month=8, day.beg=5, day.end=15,
country= "Serbia", mag.low =5.5,
r=2.2)
```

Calculation functions

In our *R* package, we have three functions that perform different calculations on visual meteor data.

- *solar.long (year, month, day, time)* calculates the solar longitude with respect to the equinox of 2000.0 (Steyeart, 1991) for a given year, month, day and time in hours.

- *zhr (data,year, month, day.beg, day.end, shw, r=NULL, Ralpha=NULL, Delta=NULL, k, c=1)* calculates the average zenithal hourly rate (ZHR) of a meteor shower for a given rate data, specified shower, period of days, population index, length of time interval and ZHR correction. The average zenithal hourly rate is given by the formula

$$\overline{ZHR} = \frac{c + \sum_{i=1}^{k} n_i}{\sum_{i=1}^{k} \frac{T_{eff,i}}{C_i}}$$

where k is the number of observing periods, n_i - the number of meteors seen during the observing period *i*, $T_{eff,i}$ - the effective time or amount of time an observer actually scans the sky for meteors during the observing period *i*, and C_i – a correction factor.

In the numerator, c is included to correct for the asymmetric high and low end possibilities in a Poisson distribution (Bias, 2011.). By default, it is set to 1.

The standard error of the average zenithal rate is calculated by the formula

$$\sigma = \frac{\overline{ZHR}}{\sqrt{c + \sum_{i=1}^{k} n_i}}$$

The spatial number density of meteoroids producing meteors of magnitude at least 6.5 is (per 10^9 km^3) (Koschack and Rendtel, 1990a)

$$\rho = \frac{(10.65r - 12.15)\overline{ZHR}}{3600 \cdot 178700 r^{-1.82} V},$$

where V is the stream's geocentric velocity.

The standard error of the spatial number density is approximated with

$$\sigma_\rho \cong \frac{\sigma \cdot \rho}{\overline{ZHR}}$$

Day is divided in subintervals of *k* hours. For example, if *k=12*, subintervals are $[0,12)$ and $[12,24)$. Zenithal hourly rate is calculated for each subinterval in the following manner: If middle of observer's time period belongs to the subinterval, his/hers data values are used in calculation of ZHR.

For example, if we want to select visual meteor data for observation of Orionids, period 20-24th October 2006, 12 hours' time intervals, and calculate ZHR

```
rateOri <- filter (rate06, shw="ORI",
year =2006, month=10, day.beg=20,
day.end=24)

zhr (rateOri, year=2006, month=10,
day.beg =20, day.end=24, shw="ORI",
r=2.5, k=12)
```

- *pop.index (data, year, month, day.beg, day.end=day.beg, shw, mag=-6:7)* calculates population index of a meteor shower for a given magnitude data, specified period of days and magnitude values.

  Cumulative summarized magnitude distribution $\Phi(m)$ is formed by summing cumulative frequencies of all observers for each magnitude class *m*.

  Using the relationship $r = \frac{\Phi(m+1)}{\Phi(m)}$ and substituting 0, 1,...,m magnitudes, equation $\Phi(m) = \Phi(0) \cdot r^m$ (or $\ln \Phi(m) = \ln \Phi(0) + m \ln r$ in logarithmic form) can be written. Then, population index *r* is calculated by the method of least squares, for chosen range of magnitude values.

  Standard error of population index is approximated with

$$\sigma_r \cong r \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{(n-2)\sum_{i=1}^{n} m_i^2}}$$

  where n is number of magnitude values, e_i regression residuals, *i=1,2,...,n.*

  The interval for regression is chosen such that: there is at least 3 meteors per magnitude class, the faintest magnitude classes are not included ($m \le 4$ or in exceptional cases $m \le 5$) and there are at least 5 magnitude classes available (Koschack and Rendtel, 1990b). All these conditions are fulfilled for the range of magnitude values printed in results.

  To select magnitude data for observation of Perseids, time period 1-20th August 2007 and calculate population index using magnitudes -6 to +4, we would type

```
magnPer <-filter (magn07, shw="PER",
year=2007, month=8, day.beg=1,
day.end=20)

pop.index (magnPer, year=2007,
month=8,          day.beg=1,
day.end=20, shw="PER", mag=-6:4)
```

Drawing graphs functions

We have two functions of this type.

- *mag.distr (data, year, month, day.beg, day.end=day.beg, shw)* graphically represents magnitude distribution for a given magnitude dataset, specified meteor shower and period of days. It returns a plot of summarized magnitude distribution consisting of histogram and box-plot.

  For example, to select data for observations of Perseids, period 12-14th August 2007 and make a graphic of magnitude distribution, we would type into *R* console

```
magnPer <-filter (magn07, shw="PER",
year =2007, month=8, day.beg=12,
day.end=14)

popI.distrib (magnPer, year=2007,
month=8, day.beg=12, day.end=14,
shw="PER")
```

- *zhr.graph (data, year, month, day. beg, day.end=day.beg, shw, r=NULL, Ralpha=NULL, Delta=NULL, k, c=1, type=c ("UTC", "sol"))* represents graphically the average zenithal hourly rate of a meteor shower with error bars for a given rate dataset, specified shower, period of days, population index, length of time interval, ZHR correction and a type of x-axis display.

  For type="*UTC*", the tick marks on the x-axis represent coordinated universal time (UTC), set *k* distance apart, with labels specifying date (at 00:00 UTC). For type="*sol*", the tick marks and the labels on the x-axis represent the solar longitude, corresponding to the above mentioned time in UTC.

  Function *zhr.graph* returns the *xy* plot of the Zenithal Hourly Rate, with time (UTC) or solar longitude on the *x*-axis and the ZHR on the *y*-axis. The ZHR is represented with black filled circles with 68% confidence intervals/one sigma error bars.

  For example, to select data for observations of the Orionids, period 20–26th October 2006, 6hrs time intervals, and to generate a ZHR graph we would type:

```
rateOri <-filter (rate06, shw="ORI",
year =2006, month=10, day.beg=20,
day.end=26)

zhr.graph (rateOri, year=2006,
month=10, day.beg=20, day.end=26,
shw="ORI", r=2.5, k=6, type="UTC")
```

4 MetRApp – Java Application

The motivation for this application aims to provide a simple user interface to the *R* package and to allow all users to use its functions without any necessary knowledge of *R*. Currently, it is developed only as a desktop application, but with potential to be moved to the web.

User experience

This application uses a standard graphical user interface to communicate with users. The setup needed for runtime is only to provide the paths for the installation of *R* and other

resources needed to run the application (datasets, tables etc.).

Software architecture of MetRApp

This application is developed using standard three tier architecture. All compiled versions and the application are available at the link given in section 2 and it can be independently developed by other organizations. The application is developed using *Java* 7 and *Netbeans IDE*. The version control software is *git* and the repository host is *Bitbucket*. A short description of the software architecture of this application is provided too, as a starting point for any potential efforts.

Data tier

MetRApp does not maintain any data in the databases since its function is to be a proxy between the *R* package and the users. However, since the results of the execution of the *R* code are contained in *R* data structures, appropriate domain classes have to be implemented in *MetRApp*.

In this case, a domain class for the *R data frame* has to be implemented. Three *Java* classes are implemented to provide an appropriate representation of data from the *R data frame* – abstract class *DataFrame*, and two classes which extend the previous one – *StringDataFrame* and *DataFrameFromCSV*. More details about domain classes are provided in the documentation on the project's repository.

In addition to these domain classes, more of them had to be implemented for the IMO data. These new classes are a representation of, for example, persons or sites instances in the corresponding datasets. Also, the *R* code which is used to evaluate data is inserted in the class *InitialRCode.*

Logic tier

The logic part of this system is implemented in several packages. The first part contains filters for the selection of data which correspond to the previously explained *R* filters. These '*Java* filters' had to be implemented to avoid unnecessary parsing of data in the communication between *MetRApp* and *MetFns.* This approach allows very fast filtering of data which is done in *MetRApp* only, without calling *R* code. However, only 11 of the 13 filters could be fully implemented in *Java* because of the dependency to third – party functions available exclusively in *R* for some filters.

The architecture of this package is very basic – there is an abstract class *JavaFilter* which is then extended by concrete filters which implement the logic of the appropriate *R* filter. This abstract class has reference to the current dataset which is used in the application runtime, and also provides the abstract method which accepts *HashMap* of parameters which are needed for a concrete filter to be executed.

As stated before, *MetRApp* calls *R* functions to evaluate data and returns the result to the user. This communication is achieved using *RCaller*[8], a software library for calling *R* functions within *Java* programs. The idea behind *RCaller* is very simple – the *Java* program (in this case *MetRApp*) is the caller and it sends requests (containing *R* code and data) via *XML* and accepts responses, again in *XML* format.

The logic tier of the application also contains a few controllers which are responsible for dispatching calls between objects and for providing an essential backbone for all implemented functionalities. The controllers have also references to all data sources and they provide a control on their correct usage.

Presentation tier

As stated before, this application uses a standard graphical user interface platform. It is based on *Swing* components, without any additional customization (and dependencies). The structure of the presentation tier is not very well optimized, since new efforts were made to move this application to the web.

Future improvements

A very large part of the application is implemented using software patterns which provide a large flexibility and very much simplify the implementation of new functionalities. Six software patterns were implemented in the application, but not all of them are currently used.

Requests for new functionalities are welcome, as well as reviews of the current version of the software. Our group will continue to develop this software, but also support new initiatives by branching this code base.

5 Conclusion

The developed software covers a vast majority of use cases[9] specified by our meteor observation group. Due to its modular architecture, it is possible to expand the application specification and to provide additional features if needed. All resources including source code, test examples, documentation and other files are provided at public repositories, and everyone can develop their specific distribution of this software. We hope that other IMO observers will find our software useful. However, if new features are requested, we shall try to implement them in new releases of our software.

Acknowledgment

We would like to thank Branislav Savić whose suggestions and comments were invaluable in developing our software.

[8] https://code.google.com/p/rcaller/
[9] http://en.wikipedia.org/wiki/Use_case

References

Bias P. V. (2011). "A Note on Poisson inference and extrapolations under low raw data and short interval observation conditions". *WGN, Journal of the IMO*, **39**, 14–19.

Koschack R. and Rendtel J. (1990a). "Determination of spatial number density and mass index from visual meteor observations (I)". *WGN, Journal of the IMO*, **18**, 44–58.

Koschack R. and Rendtel J. (1990b). "Determination of spatial number density and mass index from visual meteor observations (II)". *WGN, Journal of the IMO*, **18**, 119–140.

Rendtel J. and Arlt R., editors (2008). IMO Handbook For Meteor Observers. IMO, Potsdam.

Steyaert C. (1991). "Calculating the Solar Longitude 2000.0". *WGN, Journal of the IMO*, **19**, 31–34.