Meteorological Data Quality Checks Software

USER MANUAL

European Commission

Date	November 2018
Location	JRC-Directorate Sustainable Resources
	D5 – Food Security Unit, Ispra, Italy
Distribution	MARS4CAST

Annotations table

Annotation	Description
AMDAC	**A**ctual **M**eteorological **DA**tabase **C**onstruction
QUACKME	**Qu**ality **C**hec**k**s of **Me**teorological Data
SG	**S**pecialists **G**roup
MOS	**M**odel **O**utput **S**tatistics
WMO	**W**orld **M**eteorological **O**rganization
ECMWF	**E**uropean **C**entre for **M**edium-Range **W**eather **F**orecasts

Version summary

Version	Description
1.1.0	New parameter for WeackChecks command line New parameter for ThresholdChecks command line
1.1.1	Change the value of option –m for WeakChecks
1.2.0	Added command line for RRRGenerator
1.2.1	1.10 Software error messages
1.3.0	MOS options for WeakChecks, Aggregation

1. Modules

QUACKME package contains the following modules:

- WeakChecks module
- Aggregation module
- HeavyChecks module
- ThresholdChecks module
- TGenerator module
- SConverter module
- GUI module
- Quackme module

ATTENTION!

To run the various steps need to satisfy the following prerequisites:

- Have installed the R version 3.4.3
- Open an Windows Command
- Select the current path equal to the path of QUACKME installation (In this manual we use, like sample, the installation path for QUACKME = “d:/QUACKME/R”)

To run the QUACKEM R modules need to build the following directory structure, starting from a root directory (in our case “d:/QUACKME”):

- Config directory (that will contains all configuration files
- Output (will contains Log files and output files)
- Input (will contains original input files
- R (will contains R sources)
- History (will contains all history files)
- Scripts (contains the scripts used to generate some XML configuration file)

1.1 **WeakChecks** module

WeakChecks module represents the start of the chain for meteorological data elaboration. His role consists in execution of the weak checks on hourly meteorological observations.

To run the weak checks is necessary to run the following command line:

```
     "c:/Program Files/R/R-3.4.3/bin/RScript.exe"
WeakChecks.Parallel.R
     -c "d:/QUACKME/Config/"
     -i "d:/QUACKME/Input/"
     -o "d:/QUACKME/Output/"
     -f "I.WMO.20180114.csv"
     -t "d:/QUACKME/History/"
     -d 20180114
     -l "d:/QUACK<E/Log/"
     -r
"O.KO.WeakChecks.20180114.xml;O.KO.WeakChecks.20180115.xml"
     -n "number of cores"
     -m "hour"
     -s "MOS path"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **WeakChecks.Parallel.R** represents the application to be executed inside the **R** engine (need to specify the absolute path where the file resides)
- **-c (**mandatory option) specify the path for configuration files (*Workflow.xml*, *Messages.xml*)
- **-i** (mandatory option) specify the path for the input files
- **-o (**mandatory option) specify the path for the output files
- **-f (**mandatory option) specify the name of the file to be processed. The file must be present into the path specified by the option "**-i**"
- **-t** (optional) specify the path where to create the history file. If this is missing the history file are not created
- **-d** (mandatory) specify the reference date on the format YYYYMMDD
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists
- **-r** (optional) specify the name of the files that contains the errors corrected from the GUI. The file must be present on the input path (specified by the parameter **–i**). If you want to specify more than one file, the file names must be separated by "**;**".

- **-n** (optional) specify the name of cores to use on parallel run. If that parameter missing the application will use the number of detected cores – 2, otherwise the application will use the number of cores determined like minimum between the number of cores for the machine and the value specified on the parameter.
- **-m** (optional) allow the possibility to indicate to WeakChecks module to append the H.<currentdate>.hist file (if exists) to the input csv file. From the history file are read only observation of day in elaboration (option –d) with time <= hour.
- **-s** (optional) allow the possibility to indicate the path of MOS file

The input of the **WeakChecks** can be:

- A text file with meteorological hourly observations from the meteorological station
- An XML file (only if present) with the corrected errors
- An XML file with the workflow configuration (Workflow.xml) present into configuration path
- An XML file with the alert messages configuration (Messages.WeakChecks.xml) present into the configuration path
- A text file with MOS data

The output of **WeakChecks** module are only files:

- File (text) with all correct observations. Convention name :
  `O.OK.WeakChecks.YYYYMMDD.dat`
- File (xml) with all wrong or suspicious observations. Convention name:
  `O.KO.WeakChecks.YYYYMMDD.xml`
- History file (text file) if the –t option was specified in the command line. Convention name:
  `H.YYYYMMDD.hist`
- Log file (text) with log details. Convetion name:
  `WeakChecksYYYYMMDDHHmm.log`
- Debug file (text) with specific details during the run. Convention name:
  `debug.WeakChecks.YYYYMMDDHHmm.txt`

1.2 **Aggregation** module

Aggregation module has the role to convert hourly observations into daily observations. For that mission the module will use only the correct observations produced by the **WeakChecks** module.

To run the **Aggregation** module is necessary to run the following command line:

```
     c:/Program Files/R/R-3.4.3/bin/RScript.exe
Aggregation.Parallel.R
     -c "d:/QUACKME/Config/"
     -i "d:/QUACKME/Input/"
     -o "d:/QUACKME/Output/"
     -f "I.WMO.20180114.csv"
     -t "d:/QUACKME/History/"
     -l "d:/QUACKME/Log/"
     -n "number of cores"
     -s "MOS path"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **Aggregation.Parallel.R** represents the application to be executed inside the **R** engine (need to specify the absolute path where the file resides)
- **-c (**mandatory option) specify the path for configuration files (*Aggregation.xml*)
- **-i** (mandatory option) specify the path for the input files
- **-o (**mandatory option) specify the path for the output files
- **-f (**mandatory option) specify the name of the file to be processed. The file must be present into the path specified by the option "**-i**"
- **-t** (mandatory) specify the path where resides the file for hourly observations data of the past
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists.
- **-n** (optional) specify the name of cores to use on parallel run. If that parameter missing the application will use the number of detected cores – 2, otherwise the application will use the number of cores determined like minimum between the number of cores for the machine and the value specified on the parameter.
- **-s** (optional) allow the possibility to indicate the path of MOS file

The input of the **Aggregation** can be:

- A text file with meteorological hourly observations from the meteo station. Naming convetion of the input file:

O.OK.WeakChecks.YYYYMMDD.dat

- An XML file with the configuration of the daily aggregation to generate (Aggregation.xml) present into the configuration path
- A text file with MOS data

The output of **Aggregation** module are only files:

- File (text) with all correct observations. Convention name :
  O.OK.Aggregation.YYYYMMDD.dat
- File (xml) with all wrong or suspicious observations. Convetion name:
  O.KO.Aggregation.YYYYMMDD.xml
- History file (text file) if the –t option was specified in the command line. Convention name:
  D.YYYYMMDD.hist
- Log file (text) with log details. Convetion name:
  AggregationYYYYMMDDHHmm.log
- Debug file (text) with specific details during the run. Convention name:
  debug.Aggregation.YYYYMMDDHHmm.txt

1.3 **HeavyChecks** module

HeavyChecks module has the role to verify the daily observations. For that mission the module will use the output and some configurations of Aggregation module.

To run the **HeavyChecks** module is necessary to run the following command line:

```
     "c:/Program Files/R/R-3.4.3/bin/RScript.exe"
HeavyChecks.Parallel.R
     -c "d:/QUACKME/Config/"
     -i "d:/QUACKME/Output/"
     -o "d:/QUACKME/Output/"
     -t "d:/QUACKME/History/"
     -f "O.Aggregation.20180114.dat"
     -m "Model.2018.dat"
     -l "d:/QUACKME/Log/"
     -n "number of cores"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **HeavyChecks.Parallel.R** represents the application to be executed inside the **R** engine
- **-c (**mandatory option) specify the path for configuration files (*Aggregation.xml*)
- **-i** (mandatory option) specify the path for the input files
- **-o (**mandatory option) specify the path for the output files
- **-f (**mandatory option) specify the name of the file to be processed. The file must be present into the path specified by the option **"-i"**
- ~~**-m** (optional) specify the name of an Model Output Statistics file, that must be present into the configuration path, specified though the option **"-c"**~~
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists.
- **-t** (mandatory) specify the path where resides the file for hourly observations data of the past. If present, the module will make a copy of the output file to this directory. The file name respect the convention name: D.<YYYYMMDD>.hist.
- **-n** (optional) specify the name of cores to use on parallel run. If that parameter missing the application will use the number of detected cores – 2, otherwise the application will use the number of cores determined like minimum between the number of cores for the machine and the value specified on the parameter.

The input of the **HeavyChecks** can be:

- A text file with meteorological daily observations for all stations. Naming

convention of the input file:
O.Aggregation.YYYYMMDD.dat

- An XML file with aggregation properties to manage (Aggregation.xml) present into the configuration path
- An XML file with alert messages configuration (Messages.HeavyChecks.xml) present into the configuration path
- ~~A text file with Model Output Statistics data, having the same format of the input file.~~

The output of **HeavyChecks** module are only files:

- File (text) with all correct observations. Convention name :
  O.OK.HeavyChecks.YYYYMMDD.dat
- File (xml) with all wrong or suspicious observations. Convention name:
  O.KO.HeavyChecks.YYYYMMDD.xml
- Log file (text) with log details. Convetion name:
  HeavyChecksYYYYMMDDHHmm.log
- Debug file (text) with specific details during the run. Convention name:
  debug.HeavyChecks.YYYYMMDDHHmm.txt

1.4 **ThresholdChecks** module

ThresholdChecks module has the role to execute checks between the daily observations and threshold data (daily and seasons). For that mission the module will use the output and some configurations of HeavyChecks and Aggregaton modules.

To run the **ThresholdChecks** module is necessary to execute the following command line:

```
     "c:/Program Files/R/R-3.4.3/bin/RScript.exe"
HeavyChecks.Parallel.R
     -c "d:/QUACKME/Config/"
     -d "20190715"
     -i "d:/QUACKME/Input/"
     -f "O.OK.HeavyChecks.20180114.dat"
     -o "d:/QUACKME/Output/"
     -t "d:/QUACKME/History/"
     -l "d:/QUACKME/Log/"
     -n "number of cores"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **Threshold.Parallel.R** represents the application to be executed inside the **R** engine
- **-c (**mandatory option) specify the path for configuration files (*Aggregation.xml, Messages.Daily.ThresholdChecks.xml, Messages.Seasons.ThresholdChecks.xml*)
- **-d** (mandatory) specify the reference date on the format YYYYMMDD
- **-i** (mandatory option) specify the input path
- **-f (**mandatory option) specify the name of the file to be processed. The file must be present into the path specified by the option **"-i"**
- **-o (**mandatory option) specify the path for the output files
- **-t** (mandatory) specify the path where resides the file for hourly observations data of the past. The module will search for files with name: D.<YYYYMMDD>.hist, where YYYYMMDD is previous to the current elaboration date
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists.
- **-n** (optional) specify the name of cores to use on parallel run. If that parameter missing the application will use the number of detected cores – 2, otherwise the application will use the number of cores determined like minimum between the number of cores for the machine and the value specified on the parameter.

The input of the **ThresholdChecks** can be:

- A text file with meteorological daily observations for all stations. Naming Convention of the input file:
  `O.OK.HeavyChecks.YYYYMMDD.dat`
- An XML file with aggregation property to manage (Aggregation.xml) present into the configuration path
- Two XML file with alert messages configuration (Messages.Daily.ThresholdChecks.xml and Messages.Seasons.ThresholdChecks.xml) presents into the configuration path
- Two texts files with daily and seasons thresholds.

The output of **ThresholdChecks** module are only files:

- File (text) with all correct observations. Convention name :
  `O.OK.ThresholdChecks.YYYYMMDD.dat`
- File (xml) with all wrong or suspicious observations. Convention name:
  `O.KO.ThresholdChecks.YYYYMMDD.xml`
- Log file (text) with log details. Convetion name:
  `ThresholdChecksYYYYMMDDHHmm.log`
- Debug file (text) with specific details during the run. Convention name:
  `debug.ThresholdChecks.YYYYMMDDHHmm.txt`

1.5 **TGenerator** module

The **TGenerator** module has the role to create the threshold files (daily and seasons).

To run the **TGenerator** module is necessary to lucnh the following command line:

```
     "c:/Program Files/R/R-3.4.3/bin/RScript.exe"
TGenerator.Parallel.R
     -m [Daily|Season]
     -o "d:/QUACKME/Output/"
     -l "d:/QUACKME/Log/"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **TGeneratore.Parallel.R** represents the application to be executed inside the **R** engine
- **-m (**mandatory option) specify the mode to run: **Daily** if want to create the daily threshold file, **Season** if want to create the seasons threshold file
- **-o (**mandatory option) specify the path for the output files
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists.

For each mode, the application will generate (into the Log Path) a specific debug file. Is possible to run in parallel two instance of **TGenerator**, one with Daily mode and other with Season mode, in the way to use at maximum the parallelism.

TGenerator open a connection to an Oracle database of which the parameters are inside TGenerator.Parallel.R file. It is necessary to create a local DSN to use for Oracle connection.

The output of **TGenerator** module are only files:

- File (text) with daily threshold. Convention name :
  `Threshold.Daily.dat`
- File (text) with season's threshold. Convention name :
  `Threshold.Seasons.dat`
- Log file (text) with log details. Convention name:
  `TGenerator.YYYYMMDDHHmm.log`
- Debug file (text) with specific details during the run. Convention name:
  `debug.daily.TGenerator.YYYYMMDDHHmm.txt`
  `debug.seasons.TGenerator.YYYYMMDDHHmm.txt`

1.6 **SConverter** module

SConverter module has the role to convert the JRC format of daily data into S-File format, specific of Alterra.

The command line to run **SConverter** is the following:

```
"c:/Program Files/R/R-3.4.3/bin/RScript.exe" SConverter.R
-c "d:/QUACKME/Config/"
-f "d:/QUACKME/Output/OK.Threshold.20180114.dat"
-o "d:/QUACKME/Output/"
-l "d:/QUACKME/Log/"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **SConverter.R** represents the application to be executed inside the **R** engine
- **-c (**mandatory option) specify the path for configuration files (*Aggregation.xml*)
- **-f (**mandatory option) specify the name of the file to be processed, including the path
- **-o (**mandatory option) specify the path for the output files
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists.

The output of **Sconverter** module are only files:

- File (text) with data. Convention name :
  `SYYYYMMDD.dat`
- Log file (text) with log details. Convention name:
  `SConverter.YYYYMMDDHHmm.log`

1.7 **RRRGenerator** module

RRRGenerator module has the role to generate the 3/6 hour precipitation files for a specif day.

To run the RRRGenerator is necessary to run the following command line:

```
     "c:/Program Files/R/R-3.4.3/bin/RScript.exe"
WeakChecks.Parallel.R
     -i "d:/QUACKME/History/"
     -o "d:/QUACKME/Output/"
     -d <reference date>
     -l "d:/QUACK<E/Log/"
     -r <hour interval [3/6]>
     -n "number of cores"
```

, where:

- **Rscript** represents the program that will run the **R** engine
- **RRRGenerator.R** represents the application to be executed inside the **R** engine (need to specify the absolute path where the file resides)
- **-i** (mandatory option) specify the path for the hourly history files
- **-o (**mandatory option) specify the path for the output files
- **-d** (mandatory) specify the reference date on the format YYYYMMDD
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists
- **-r** (madnatory) specify the time interval to generate the precipitation file. Two values are accepted: 3 or 6
- **-n** (optional) specify the name of cores to use on parallel run. If that parameter missing the application will use the number of detected cores – 2, otherwise the application will use the number of cores determined like minimum between the number of cores for the machine and the value specified on the parameter.

The input of the **RRRGenerator** are the hourly history files with the name lke H.<date>.hist. The application are searching for the history file relative to the reference date (parameter –d) and to the <reference date – 1 day>.

The output of **RRRGenerator** module are text files (4 precipitation files for 6 hour interval and 8 preicpitation files for 3 hour interval) :

- Precipitation files: **`rrr_YYYYMMDDHH.txt`**
- Log file (text) with log details. Convetion name: **`WeakChecksYYYYMMDDHHmm.log`**
- Debug file (text) with specific details during the run. Convention name:

debug.WeakChecks.YYYYMMDDHHmm.txt

1.8 **QUACKME module**

The QUACKME module is an R based application with the scope to generate the bat file to run all software for quality checks of meteorological data.

QUACKME module accept the following options for the command line:

```
"c:/Program Files/R/R-3.4.3/bin/RScript.exe" Quackme.R
-c "d:/QUACKME/Config/"
-i "d:/QUACKME/Input/"
-o "d:/QUACKME/Output/"
-f "I.WMO.20180114.csv"
-t "d:/QUACKME/History/"
-d "20180114"
-r "O.KO.HeavyChecks.xml"
-l "d:/QUACKME/Log/"
```

where:

- **Rscript** represents the program that will run the **R** engine
- **Quackme.R** represents the application to be executed inside the **R** engine (need to specify the absolute path where the file resides)
- **-c (**mandatory option) specify the path for configuration files
- **-i** (mandatory option) specify the path for the input files
- **-o (**mandatory option) specify the path for the output files
- **-f (**mandatory option) specify the name of the file to be processed. The file must be present into the path specified by the option "**-i**"
- **-t** (optional) specify the path where to create the history file. If this is missing the history file are not created
- **-d** (mandatory) specify the reference date on the format YYYYMMDD
- **-l** (optional) specify the path for log and debug files. If this option is missing the log and debug files will be moved into Log directory inside the output path (specified by the option "**-o**"). The Log directory will be create if not exists
- **-r** (optional) specify the name of the file that contains the WeakChecks errors corrected through the GUI module. The file must be present on the input path (specified by the parameter **–i**).

The QUACKME module product a .bat file that will execute the necessary modules in base of input parameters.

Quackme.20190715.bat

1.8 GUI module

The GUI module is an Web application, based on PHP, HTML, Javascript, JQuery and Bootstrap.

Prerequisites to run the web application:

- PHP 7 installed
- An web server that can run PHP (preferable Apache2)
- Create anywhere on the Web Server a virtual directory with all the structure of the web project

The URL to run the application is:

<web server name or web server ip>:<http port if mapped>/MainPage.html

1.9 Sample

The attached document are containing command line sample for all R modules listed before.

SampleCommandLine.txt

1.10 Software error messages

1.10.1 WeakChecks errors

Error message	Severity
Input file contains duplicate rows for the combination <Station, DayTime>	Fatal
Error detecting cores: <error message>	Fatal
The '-r' option has an invalid value!	Fatal

Missing error file <error file name> - skip elaboration of date.	Warning

1.10.2 Aggregation errors

Error message	Severity
Error detecting cores: <error message>	Fatal
Missing History file: <history file name>	Warning

1.10.3 HeavyChecks errors

Error message	Severity
Missing workflow active configuration for HeavyChecks	Fatal
Model file <model file name> does not exists!	Fatal
Error detecting cores: <error message>	Fatal

1.10.4 ThresholdChecks errors

Error message	Severity
Missing workflow active configuration for ThresholdChecks	Fatal
Error detecting cores: <error message>	Fatal
Missing daily threshold file: <threshold daily filename>	Fatal
Missing season threshold file: <threshold seasons filename>	Fatal
Error loading daily threshold file: <error message>	Fatal
Error loading seasons threshold file: <error message>	Fatal

1.10.5 Common errors

Error message	Severity
The config path is mandatory !	Fatal
The config path <config path> does not exists !	Fatal
The input path is mandatory!	Fatal

The input path <input path> does not exists!	Fatal
The input file is mandatory !	Fatal
The input file <input file name> does not exists on the path <input path>!	Fatal
The output directory is mandatory !	Fatal
Output directory <output path> does not exists!	Fatal
Log directory <log path> does not exists!	Fatal
The history directory is mandatory!	Fatal
History directory <history path> does not exists!	Fatal

Each module can report unmanaged error, if the various methods are generating it. In the log file you can find a message preceded by “ERROR”, method name and error description.