8 July 2018

TO: QAtools File
FROM: Al Cooper
SUBJECT: R script to produce data-review plots as in QAtools

General Structure

The objective documented in this memo is to provide an Rscript that produces the same set of plots as QAtools but that can be included in the routine processing chain so that the plots go to the field catalog automatically. To ensure compatibility with QAtools, the script should reference the same plot functions and configuration as QAtools and should share global variables like VRPlot.

The following is some reference documentation describing how this is handled in QAtools:

1. A set of functions for generating the plots in maintained in the subdirectory *PlotFunctions.* Those functions have names like Rplot1.R ... RPlot30.R .

2. A configuration routine *Configuration.R* defines variables to be plotted for each project, and it has a separate section for each project. When it is read at initialization, it sets up a list named *VRPlot* with a section for each plot. There may be multiple plots generated by a function in *PlotFunctions;* the correspondence is not 1:1. For example, multiple plots are generated by RPlot3.R that display information regarding humidity in various ways, but the specification of veriables is all in the list specified by VRPlot$PV5. There are some special rules to follow when setting up *Configuration.R:*

   (a) In sequences of temperature variables like "ATH1", 'ATH2", "ATF1", "AT_A" the variable specified last will be used as the reference when differences are plotted.

   (b) The reverse convention applied to humidity (Sorry!); the variable specified first is the reference dewpoint for comparisons.

   (c) There is a set of offsets called "pitch_offset", "roll_offset" and "thdg_offset" that are used for comparisons among IRU/INS variables, to account for the case where there is a project-dependent offset resulting from variability in installation. These can be used to produce normally zero differences to facilitate detection of changes.

   (d) Variables with wing-position suffixes can be specified as follows: "CONCD_". In this case the variable list will be searched for the first match, so you don't have to keep changing the variable names when the probe is moved. However, if there are two variables with the same names except for the suffix, you must specify the full names even if only one is desired in the plot; otherwise the wrong choice may be selected.

   (e) There are usually some items at the end of the list, e.g., VRPlot$PV30, that are included only as dummy placeholders in case new plots are defined. They are not used now.

3. The PDF plots then are generated in the function *savePDF(Data, inp).* In QAtools the configuration file is loaded first and the input data file and default user inputs are provided. In addition, any limits currently specified (like time limits or minimum TAS) are enforced when the plots are generated. This therefore requires some special handling in the Rscript. *savePDF()* loops through the finctions in *PlotFunctions* and generates the plots as specified there. It saves the output in a single output file named *ProjectTFnnPlots.pdf* where *Project* is the project name (e.g., WECAN), TF is the type of flight (e.g., *rf* or *tf* or *ff*), and *nn* is the flight number. The plots for the first research flight in WECAN will be saved in *WECANrf01Plots.pdf* .

Structure of DataReview.R

This script re-uses significant parts of the existing code in QAtools so when plots are changed they are changed in both QAtools and the new script, called here *DataReview.R* . Then the steps required to construct the plots for the latest flight in the active-project directory are these:

1. *DataReview.R* will set up some run arguments and some global variables to match those generated by the *global.R* routine in QAtools:
   (a) *DataReview.R* will determine some aspects of the desired run as follows:
      i. It will find the active project from the first entry in *Configuration.R* . This can be over-ridden by specifying the first run-time argument to *DataReview.R* to be the active project desired for the plots.
      ii. The script will then search for the latest netCDF file in the appropriate project directory. Here also, the desired flight can be over-ridden by supplying a second run-time argument as the flight name, for example “ff03”.
      iii. The next two run-time arguments, if present, specify the time limits. The command to run the script might be, for example, *Rscript DataReview.R WECAN ff03 150000 160000* . If these arguments are omitted or if they are “0 400000” the plots are shown from the first and last times that the airspeed exceeds 65 m/s.
      iv. The fifth run-time argument can be used to produce a single plot, for example by using the argument 1 to get a plot of the plan-view flight track.

   (b) If the plot file for the latest flight with a netCDF file in the specified project already exists, the script will exit. (Purge the file if you want to re-create it.) However, if the flight is specifically supplied via run-time arguments, the plot file will be regenerated.

2. Next, *DataReview.R* must set up the desired plot variables as specified in *Configuration.R* for this project. In *QAtools* this is done in the function *loadVRPlot()*, which formerly was located in *global.R.* To ensure consistency, this is now changed so that *loadVRPlot.R* is included by “sourcing” in both *QAtools* and *DataReview.R.* Once the project and flight are determined as above, *loadVRPlot()* is called to construct the list VRPlot, which contains the project-specific variables used to generate the plots.

3. The required variable list is then constructed from the *VRPlot* variables, and the data.frame to be used for the plots is constructed using that variable list.

4. Sourcing of functions is also used to maintain consistency in *transferAttributes.R* and *savePDF.R*, both now “sourced” in both QAtools and *DataReview.R* . The plot routines themselves are also loaded from the *PlotFunctions* directory by both.

5. Finally, *savePDF()* is called to construct and save the file with the plots.

Instructions:

Run the script using one of these calls:

1. *Rscript ~/RStudio/QAtools/DataReview.R* . With no arguments, this will search for the latest file in the project defined first in *Configure.R* and, if there is no matching plot file, will produce it. It will not replace as existing plotfile. (Plotfiles have names like SOCRATESrf01Plots.pdf.)

2. *Rscript ~/RStudio/QAtools/DataReview.R Project* . This will perform the same function for the specified project.

3. *Rscript ~/RStudio/QAtools/DataReview.R Project Flight* . This will construct plots for the specified project and flight if the plot file does not already exist. *Flight* should be specified in a format like rf15 . In this case, the *Project* argument cannot be omitted. If *Flight* is specified to be ALL, all plot files not already present for the specified project will be generated.

4. *Rscript ~/RStudio/QAtools/DataReview.R Project Flight Start End* . This will restrict the plot to span only the specified times, which should be specified in HHMMSS format (e.g, 121506 or 23555). In this case, the output file will have “Plot” replaced by “PlotTR” (e.g., SOCRATESrf15PlotsTR.pdf) and an existing file of the same name will be overwritten.

5. *Rscript ~/RStudio/QAtools/DataReview.R Project Flight Start End PlotNumber* . This wil function like item 4 above except that it will produce only a single plot. In this case, *Start* and *End* can be supplied as 0 and 400000 to generate the plot for the entire flight.

Alternately, the script can be run from the R or RStudio console in interactive mode. In that case, it will ask for user input for the five arguments above, with corresponding defaults that would represent leaving the run-time arguments above missing.

– End of Memo –