**Proposal for community development of RDBES package (“icesRDBES”)**

WKRDB-EST2, 15/9/2020

The aim of these guidelines has been not to make package development too onerous for the wider ICES community whilst also not putting too large a workload on the package maintainers – we also still need to ensure a minimum standard is met (e.g. the code is valid and in a consistent style, verifiably does what it’s supposed to, and can be submitted to CRAN).

Once the guidelines are agreed we should create a simple guideline document that can be given to potential contributors describing what they need to do. These contribution guidelines can be included in the repo e.g. <https://github.com/tidyverse/dplyr/blob/master/.github/CONTRIBUTING.md>

* The package source code is hosted within its own repo by ICES e.g. in ices-tools-dev / ices-tools-prod
* A small group of maintainers will need to be volunteered, including someone from the Secretariat
* There are two branches within the repo: master and dev
  + The master branch is protected so that only the maintainers can commit to it
  + The dev branch is used for all development work – contributors can commit directly to it
  + A pull request needs to be created when we want to merge the development branch into the master branch – the maintainers will need to approve the pull request
  + Release labels can be applied to the master branch to keep track of releases
  + A “lint” tool is configured that will compare committed code to a defined style and warn if there any problems – contributors should endeavor to resolve any issues flagged
  + Contributors need to be given commit access to the repo – ICES have a work-flow for this.
  + Contributors should pull the dev branch, make and commit changes on their local machine, and only push changes back to GitHub once their work is consistent (e.g. a new function is created and documented)
  + Contributors should only use base R / approved packages
* Before any development is started an “issue” should be raised on the GitHub repo – this can be to point out a bug in existing code, improve existing functions, or describe new functions that are required.
  + The proposed changes can then be discussed and agreed with the maintainers and other relevant people – this should also act as peer-review system for the statistical content of the proposed development
  + All contributed developments should be linked to an issue – they will not be included in a future pull request if they are not.
  + If a bug has been identified, then working code to demonstrate that bug should be provided in the issue – this can then be converted into a test within the package. The package code should then develop to a point where the test can be passed.
* The fastest way for contributors to get their code include in the package is to provide code that fully meets our package standards. These are:
  + The contributor is using the latest version of R, roxygen and any dependencies
  + For each function an R file exists in the R directory
  + Full roxygen2 documentation has been generated for that function and any data included. A good example of comprehensive document can be provided by the “gam” function (<https://www.rdocumentation.org/packages/mgcv/versions/1.8-33/topics/gam>). This level of documentation might not be appropriate or feasible in all cases but does show some important features to bear in mind.
  + The contributor has defined tests for all new functionality
  + Devtools::check has been run successfully on their local machine
  + The code passes the automatic lint checks
* We recognize that not every contributor will be able to meet this standard so have also defined a minimum standard – only supplying code at this level will result in it taking longer to be included in the package. The minimum standard of contribution is:
  + The contributor is using the latest version of R, roxygen and any dependencies
  + For each function an R file exists in the R directory
  + The Roxygen2 documentation comments have been generated for that function and any data – the descriptions of functions and parameters should be written. The Roxygen2 function should have a short description – see the documentation for “exampleFunction” for an example.
  + Some simple examples of using the functions and its expected outputs are supplied
  + The code doesn’t have any major issues raised by the lint checks
* Periodically the maintainers will update the package in CRAN. If people want/need the latest version of the package it can always be installed directly from GitHub.