A basic workflow for submtting a new version of GMACS

Matthieu VERON

Last compiled on 16 January, 2023

[ADpaths_Windows.txt]: [ADpaths_MacOS.txt]: [Assessments]: https://github.com/GMACS-project/GMACS_Assessment_code/tree/main/Assessments [Run_GMACS]: https://github.com/GMACS-project/GMACS_Assessment_code/blob/main/GMACS/Run_GMACS.Rmd [Compare_Version_VS_Last_Assessment]: https://github.com/GMACS-project/GMACS_Assessment_code/blob/main/GMACS/Compare_Version_VS_Last_Assessment.Rmd

1 Introduction

This document is intended to give you "guidelines and generic steps" to follow when working on GMACS and releasing a new version. Its aims are to show you how to use the GMACS Github repository to modify GMACS while tracking code changes. The idea is to make this workflow easy to follow so we will work with Github without the command-line interface using the graphical interface for Git *Github Desktop*.

We assume you already know how to use Rstudio with projects. If you have never used Github (and/or Githhub Desktop) with Rstudio, you can find out more from the following online workshops: - the "Happy Git and Github for the useR" from *Jenny Brian*, and - the R Workflow workshop from *Elizabeth Holmes*.

Throughout this document, we will use `R` code to update to a new version of GMACS. The functions used in this document are available in the `gmr` package, which is has been designed to work with GMACS within R. This package is available on the GMACS-project GitHub organisation. To get it installed you will need the `devtools` package.

In the course of this document, we will also need various packages required to i) call AD Model Builder (ADMB) in `R` (to compile and build the GMACS executable). These packages are automatically loaded when loading the `gmr` package into R.

The following code proceeds the installation/update of the `[gmr]` package on your machine and load the various libraries.

```
rm(list = ls())     # Clean your R session

# Set the working directory as the directory of this document----
# setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# check your directory
getwd()

# Install and load the packages----

# 1.Install devtools and gdata on your machine ----
if (!require("devtools")) {
```

```
  # install devtools
  install.packages("devtools")
}

if (!require("gdata")) {
  # needed to manipulate data
  install.packages("gdata")
}

# 2. Install / update gmr package ----
.Src <- "GMACS-project/gmr"
# Get the latest version of gmr? (0: no; 1: install for the first time; 2: update the package)
.Update <-
  0
# the library directory to remove the gmr package from
mylib <-
  "~/R/win-library/4.1"

#  remotes::install_github() will work to install gmr on your machine
if (.Update == 1)
  devtools::install_github(.Src)

# Updating to the latest version of gmr
if (.Update == 2) {
  remove.packages("gmr", lib = mylib)
  devtools::install_github(.Src)
}

# Load the gmr package
library(gmr)
```

`Codes` in this document are summarized in a clean and runnable `.R` script (UpdateGMACS.R) which you should use whenever you want to propose a new version of GMACS. This R script will guide you through all the steps of the procedure making the GMACS upgrade process easier and more transparent.

2 Set-up for using this workflow

To submit a new version of GMACS, you will need a Github account if you want to work with the GMACS-proect repositories and act as an active developer. You will also need to get installed R (or Rstudio). Below are the links to install these programs:

- Install R
- Install Rstudio

If you want to link your R/Rstudio to your Github account so that you can push your changes from Rstudio directly to Github, please refer to the two workshops listed above. I will not cover this topic in this document. In this course, you will be working with Github Desktop to make the link between your updates on your local repositories and the one on your GitHub account.

3 Let's get the *GMACS_Assessment_code* repository on your computer

The following parts shows you a workflow to get the GMACS_Assessment_code repository on your computer. Here, you have two options:

1. You want to contribute to the development of GMACS: you will need to **fork** the repository to be able to submit your changes to the organization owners (using a pull request);
2. You only want to **clone** the repository of the organization and modify it for your own purpose (without contributing to the development of GMACS).

3.1 Forking the GMACS_Assessment_code repository

By ***forking*** this repository, you will be able to submit contribution to the organization's repository. If you just want to copy it and then modify it for your own purpose on you computer, please follow the steps described in the Copy a GMACS-project repository section.

1. In Github Desktop, click *File > Clone Repository*
2. In the *"Clone a repository"* popup window, chose the URL tab and paste the GMACS_Assessment_code repository url.
3. Check the selected folder in the local path and click `Clone`.

You are now ready to make some changes to GMACS, commit them and push up them to your local repository. As soon as you are ready to submit a new version of GMACS, you will have to sumbit a Pull request on the GitHub organization.

3.2 Clone the GMACS_Assessment_code repository

Here I show you how to copy the GMACS_Assessment_code repository on your computer. By following these steps, you will not be able to contribute to the GMACS-project organisation but you will have the possibility to adapt this repository for your purpose. Below are the steps to follow:

1. Get and copy the url of the GMACS_Assessment_code repository;
2. Go to you Github account. It should be github.com/**your-name**;
3. In Github, click the `+` in top right and select `import repository`;
4. Paste the url in ***the Your old repository's clone URL*** section and give a name for this new repo. You have now the GMACS_Assessment_code on your own Github (i.e., you have an url looking like *github.com/your-name/GMACS_Assessment_code* if you kept the same name as the one of the organization);
5. You can now clone this repository to your computer following the same steps as those used to clone the GMACS_Assessment_code repository from the organization website.

4 Let's work on a new version of GMACS

In the GMACS_Assessment_code repository, two sub folders should attract your attention when you want to implement/develop a new version of GMACS:

1. The **Dvpt_Version** folder which contains the latest version of GMACS ***you want to modify***.

2. The **Latest_Version** folder which holds the latest stable and tested version of GMACS (similar to the one in the Dvpt_Version folder, before you made any changes).

These two sub folders contain all the hardware you need to run GMACS either in a "development way" or to get a "stable" stock assessment.

If you want to make new changes to the GMACS code, you should work in the *Dvpt_Version* sub folder to keep the latest version "clean" till you run the comparison before submitting your changes.

In the following, I give you some guidelines about how correctly update a GMACS version and make a pull request on Github. Obviously, this assumes that you have previously forked (and not cloned on your local machine) the organization's repo.

Let's take the example of incorporating a simulation module into GMACS. I will not go through the implementation of the code itself but simply give you the path to follow to:

i. keep track of these changes,
ii. check that they do not impact the obtainment of a new executable (i.e., that the code can be compiled),
iii. analyze the impact of these changes on the assessment for a stock (i.e., compare with the last stable version of GMACS the results of the assessment using this version under development) and,
iv. release this new stable version of GMACS to the community.

Reminder: **You will be working in the Dvpt_Version subfolder.**

4.1 Modifying the gmacsbase.tpl

In the Dvpt_Version sub folder, open the gmacsbase.tpl file in an editor and incorporate the new functions/variables to implement the simulation module. This operation does not require any "new entries" or variable declarations in the .ctl, .dat or .prj files so that the change you make will remain at the gmacsbase.tpl file level.

4.2 Check compilation and build the executable

Once you are done with the code implementation, you now need to check that you are still able to compile the model and build the executable. To build the source files of GMACS into an executable, you will need to call ADMB. The only requirement is to provide GMACS with specific directories so it will be able locate ADMB and a C/C++ compiler on your machine.

To provide such directories, you can use the [ADpaths_Windows.txt] (or [ADpaths_MacOS.txt] on linux) file located in the [GMACS_Versions] folder of the GMACS organization. *You will need to modify the paths accordingly with your setups on your computer.*

This text file holds the information of specific R variable names and the pathway to use them. **Please do not modify the R variable names in this file otherwise you will have troubles when building the GMACS executable.**

When building a new GMACS executable, you will be asked to provide a name for the version you are implementing. **Please check the name of the current GMACS version and provide one accordingly.**

The following code compiles and build the GMACS executable.

```
# Define the name of the file containing the different pathways needed to build
# the GMACS executable - (here we are working with windows)
ADMBpaths <- "ADpaths_Windows.txt"

# Run the GetGmacsExe function
.GetGmacsExe(ADMBpaths = ADMBpaths)
```

While the command is running, you will see on the console exactly what is going on. If the compilation worked, you should have a new executable named `gmacs.exe` in the Dvpt_Version directory (if that is not the case you should have a look at the "ADMBpaths" file you used).

You are now ready to test this new version.

4.3 Run the Development version:

You are now going to run the GMACS version you implemented. We are still working in the Dvpt_Version folder.

First define the characteristics of this analysis:

```
# II- Run the development version----

# Define the working directories
# Set the working directories:
Dir_Dvpt_Vers <- file.path(getwd(), "Dvpt_Version")
# Dir_Last_Vers <- file.path(getwd(), "Latest_Version")

# Stock of interest - Here we are going to test all stocks
Spc <-c(
  "all"
)

# Names of the GMACS version to consider - Here we simply call it "Dvpt_Version"
GMACS_version <- c(
  "Dvpt_Version"
)

# Define directories
VERSIONDIR <- Dir_Dvpt_Vers

# Use Last Assessment for comparison?
# If yes, you must provide the names of the model(s) you want to consider for
# comparison for each stock in the variable .ASSMOD_NAMES.
# Those models folders must have to be hold in the "Assessments" folder of
# the GMACS_Assessment_code repository.
ASS <- FALSE

# Do ou need to compile GMACS?
# vector of length(.GMACS_version)
# 0: GMACS does not need to be compiled. This assumes that an executable exists in the directory of the
# 1: GMACS needs to be compiled
COMPILE <- 0       # You already compile and build the executable
```

```
# Run GMACS - Yes we want to get an assessment for each stock
RUN_GMACS <- TRUE

# Use latest available data for the assessment?
# If TRUE, the model will be using the input files available in the "Assessment_data"
# folder of the GMACS_Assessment_code repository.
# Remind that if you implemented a version in which you made modifications on the
# input files (either the .ctl, .dat, or .prj files) you don't want to use the
# original files because it won't work.
LastAssDat <- FALSE

# Define the directories for ADMB (if not already declared before)
ADMBpaths <- "ADpaths_Windows.txt"

# Show Rterminal - Do you want to see what is going on? (similar to verbose <- TRUE)
VERBOSE <- TRUE

# Do comparison? - This is not the topic right now
MAKE_Comp <- FALSE
```

Then, run the model:

```
res <- GMACS(
  Spc = Spc,
  GMACS_version = GMACS_version,
  Dir = VERSIONDIR,
  ASS = ASS,
  compile = COMPILE,
  run = RUN_GMACS,
  LastAssDat = LastAssDat,
  ADMBpaths = ADMBpaths,
  make.comp = MAKE_Comp,
  verbose = VERBOSE
)
```

All the output files resulting from these runs are stored in the "build/stock" folder for each stock.

If everything went well, you are now ready to compare, for each stock, the outputs of this development version with the latest assessment.

4.4 Compare the results of this new version with the results from the last assessment

In the GMACS_Assessment_code repository, the required input files and specific outputs files from the latest assessment for all stocks are stored in the folder [Assessments]: for each stocks, this folder contains the models that have been tested, presented and selected by the Crab Plan Team and the Scientific and Statistical Committee to realize the assessment of each crab stock.

This comparison can be done using the [Compare_Version_VS_Last_Assessment] Rmarkdown document.

In the same way as previously, the comparison is done by calling the the `GMACS()` function. You should have been able to do the run and the comparison at the same time i.e., calling the `GMACS()` function only once but for the purpose of this document we split the steps. Obviously, we are not going again through the entire

run here: you need here to modify the `GMACS_version` and the `VERSIONDIR` variables to consider the last assessment and turn off the `RUN_GMACS` variable to avoid going again through the run. You will also need to specify the `ASS` and `ASSMOD_NAMES` variables that indicate that you want to consider the last assessment in the comparison and the name this model. Finally, you need to turn on the `MAKE_Comp` variable to indicate that you want to make comparison between the two versions.

Changes the parameters for the comparison:

```
# Names of the GMACS version to consider for run
GMACS_version <- c(
  "Last_Assessment",
  "Dvpt_Version"
)

# Define directory
Dir_Assess <- file.path(dirname(getwd()), "Assessments")
VERSIONDIR <- c(
  Dir_Assess,
  Dir_Dvpt_Vers
  )


# Need to compile the model?
# vector of length(.GMACS_version)
# 0: GMACS is not compiled. This assumes that an executable exists in the directory of the concerned ve
# 1: GMACS is compiles
COMPILE <- c(0,0)       # You already compile and build the executable

# Run GMACS - Runs have already been done
RUN_GMACS <- FALSE

# Species
Spc <- c('EAG', 'WAG',
          'BBRKC',
          'SMBKC',
          'SNOW_crab')



# Use Last Assessment for comparison?
# If yes, you must provide the names of the model for each species in the variable .ASSMOD_NAMES
# Those model folder must have to be hold in the folder Assessments
ASS <- TRUE

# names of the model for the last assessment - Only useful if comparison is made.
# if all stocks are considered they have to be ordered as follow:
# "AIGKC/EAG" / "AIGKC/WAG" / "BBRKC" / "SMBKC" / "SNOW"
ASSMOD_NAMES <- c('model_21_1e', 'model_21_1e',
                  'model_21_1',
                  'model_16_0',
                  'model_21_g')

# Do comparison?
MAKE_Comp <- TRUE
```

Call the GMACS() function:

```
# Call the GMACS() function:

tables <- GMACS(Spc = Spc,
                GMACS_version = GMACS_version,
                Dir = VERSIONDIR,
                compile = COMPILE,
                ASS = ASS,
                AssMod_names = ASSMOD_NAMES,
                run = RUN_GMACS,
                make.comp = MAKE_Comp)
```

If you are satisfied with the results of the comparison between these two versions of GMACS, you are now ready to formalize and submit this new version.

The first step before submitting your new version on GitHub is to update the Latest_Version folder with the new code of GMACS and the outputs of the assessment for each stock.

4.5 Copy the latest version to the Latest_Version folder

Luckily and for the sake of efficiency and transparency, you don't have to do anything by hand.

The `UpdateGMACS()` function allows you to:

i) Copy and paste all the files you used for the GMACS development version to the Latest_Version folder
ii) Compile this new release version in the Latest_Version folder and get everything ready to use it

```
# Use the UpdateGMACS function to copy and paste the last files in the Latest_Version
# directory
UpdateGMACS()
```

4.6 Push up changes to the organization's repository