# A basic workflow for using the GMACS repository for developpers

Matthieu VERON

Last compiled on 21 December, 2022

## 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 are available in the `gmr` package, which is useful for working with GMACS. This package is available on the repository. 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), ii) .

As the repository is private, you will have to set your credentials in R to be able to download the package. The following code help you to do this. Please use the address of your Github account (the one that gives you access to the organization's repository in Github).

```
.pack <- "gmr"
.DirSrc <- "GMACS-project/gmr"
.Username <- ""                      # the name of your Github
.UserEmail<- ""        # Your email address associated to your Github account

# Remove the package if you already got it installed on your computer
remove.packages(.pack, lib="~/R/win-library/4.1")

# Set config
usethis::use_git_config(user.name = .Username, user.email = .UserEmail)

# Go to github page to generate token
usethis::create_github_token()

# Paste your PAT into pop-up that follows...
credentials::set_github_pat()

# Now remotes::install_github() will work
devtools::install_github(.DirSrc)
```

The following code proceeds the installation 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"
.Update <- 0                              # Get the latest version of gmr? (0: no; 1: install for the fir
                                          # 2: update the package)
mylib <- "~/R/win-library/4.1"            # the library directory to remove the
# gmr package from

#  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 release 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

As a member of the GMACS-project organization, you probably already have a Github account; if not, you will need one if you want to work with this repository and potentially act as an active developer. You will also need to get installed R (or Rstudio) and Github Desktop on your computer. Below are the links to install these programs:

- Install R

- Install Rstudio
- Install Github Desktop

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.

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

I am going to show 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 therefore need to **fork** the repository to be able to directly push up changes to GMACS to the organization's repository.
2. You only want to **clone** the repository of the organization and modify it to your liking without pushing up modifications to the repository of the organization. In this case, you will not contribute directly to the development of GMACS.

## 3.1 Fork the GMACS_Assessment_code repository

By ***forking*** this repository, you will be able to contribute 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.

**Warning:** *the following workflow will provide you with "real" access to the GMACS repository. Therefore, any changes you may make to this repository can become "permanent". Please, never delete an important file or folder if you are not sure. Thank you.*

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 the organization repository. You can also create a new RStudio project in the folder holding the GMACS_Assessment_code repository on your computer and start to work with GMACS outputs.

## 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 without fear of impacting the organization's repository. 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** folder of the GMACS_Assessment_code repository, two subfolders should attract your attention when you want to implement a new version of GMACS (I'm talking about development when you make a change (no matter how small) to the GMACS code):

1. The **Dvpt_Version** folder: this folder contains the version of GMACS ***currently in development***. This means that this version is still being tested after some modifications and has not been released as a stable version.
2. The **Latest_Version** folder: this folder holds the latest stable and tested version of GMACS.

These two subfolders 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, please work only in the subfolder *Dvpt_Version*.**

In the following, I give you some guidelines about how correctly update a GMACS version and push up those change on Github. Obviously, this assumes that you have previously forked (and not cloned) the organization's repo on your computer.

Let's take the example of incorporating time-varying natural mortality into GMACS with a focus on snow crab. 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 Modify the gmacsbase.tpl

In the Dvpt_Version subfolder, open the gmacsbase.tpl file in an editor and incorporate the new functions/variables to get time-varying natural mortality configuration. 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 (first) at the gmacsbase.tpl file level.

## 4.2 Check compilation and build the executable

Once you are done with the configuration of time-varying natural mortality in GMACS, 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. Gmacs can now be compile, build and run using `R`. The only requirement is to provide him with specific directories so it will be able locate ADMB and a C/C++ compiler.

To provide such directories, you can use the ADpaths.txt file available on Github. 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 some troubles and will not be able to build the GMACS executable.** By default, this file is locate in the current working directory (i.e., in the GMACS folder).

Running this procedure, you will be asked to provide a name for the new version of GMACS. This name will be the one used at the end of all your modifications for this new version so if you have to realize multiple compilation, please stay "consistent" in the name of the version you provide.

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
.ADMBpaths <- "ADpaths.txt"

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

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 root of the Dvpt_Version directory. You are now ready to test this new version. Let's first modify (if needed) the .dat, .ctl, .prj files.

## 4.3 Modify the .dat, .ctl, .prj files

In our case, we only need to modify the control parameters for the Time varying natural mortality rates. In the .ctl file, we therefore changed the *type* of model for natural mortality, the *phase of estimation*, the *standard deviation* of the deviations for the random walk, the number of *step-changes* as we used blocked changes, we define the specific years position of the knots and finally we specified the values for the initial parameters, their low bound, their high bound and their phase.

## 4.4 Run the Development version using a case study: Snow crab

You are now going to run the GMACS development version for a specific case study. Let's take the snow crab stock as example for this demonstration. GMACS can be run using the Run_GMACS Rmarkdown document which call the `GMACS()` function.

First define the characteristics of this analysis:

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

# Species of interest
.Spc <-c(
  "SNOW_crab"
)

# Names of the GMACS version to consider
.GMACS_version <- c(
  "Dvpt_Version"
)

# Define directories
.VERSIONDIR <- c(
```

```
  paste0(getwd(), "/Dvpt_Version/")
)

# 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 <- FALSE

# Need to conpile 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 <- 0       # You already compile and build the executable

# Run GMACS
.RUN_GMACS <- TRUE

# Use latest available data for the assessment?
.LastAssDat <- TRUE

# Define the directories for ADMB
.ADMBpaths <- "ADpaths.txt"

# Show Rterminal
.VERBOSE <- TRUE

# Do comparison?
.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 this run are stored in the **SNOW_M_time_varying** directory. You are now ready to compare the outputs of this GMACS development version with the ones coming from the latest assessment.

## 4.5 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 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. Here, we will compare the results of the stock assessment for snow crab between the latest assessment and the development version of GMACS. 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 have first 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 you want to consider the last assessment in the comparison and the name of the model used for this assessment and of course, you will have to turn on the `.MAKE_Comp` variable to inticate 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
.VERSIONDIR <- c(
  paste0(dirname(getwd()), "/Assessments/"),
  paste0(getwd(), "/Dvpt_Version/")
)

# Need to conpile 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
.RUN_GMACS <- FALSE

# Species
.Spc <- c(
          '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_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 release this new version and spread it to the community.

# 5 Spread the new GMACS version to the whole community

The first step before releasing the new version is to update the Latest_Version folder with the new code of GMACS.

## 5.1 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()
```

## 5.2 Push up changes to the organization's repository

## 5.3 Re-run the assessments with the latest version (optional)