MareFrame

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Deliverable No. 3.2

Project acronym:
MareFrame

Project title:
„Co-creating Ecosystem-based Fisheries Management Solutions"

Grant agreement No: **613571**
Project co-funded by the European Commission within the
Seventh Framework Programme

Start date of project: **1st January 2014**
Duration: **48 months**

Due date of deliverable:	31/01/2015
Submission date:	26/03/2015
File Name:	D3.2 MAREFRAME_Finalised version of database system with upload routines available for main data sources and initial extraction routines for some assessment methods
Revision number:	02
Document status:	Final[1]
Dissemination Level:	PU[2]

Revision Control

Role	Name	Organisation	Date	File suffix[3]
Authors	Jamie Lentin	STL	27/01/2015	JL
Authors				
WP leader	Jamie Lentin	STL	27/01/2015	JL
Coordinator	Anna K. Daníelsdóttir	Matis		AKD
Administrative Coordinator	Oddur M. Gunnarsson	Matis		OMG
Scientific Coordinator	Gunnar Stefánsson	UI		GS
WP leader	Jamie Lentin	STL	25/11/2015	JL

[1] Document will be a draft until it was approved by the coordinator
[2] PU: Public, PP: Restricted to other programme participants (including the Commission Services), RE: Restricted to a group specified by the consortium (including the Commission Services), CO: Confidential, only for members of the consortium (including the Commission Services)
[3] The initials of the revising individual in capital letters

MareFrame

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Deliverable D3.2

Finalised version of database system with upload routines available for main data sources and initial extraction routines for some assessment methods

28/01/2014

(rev 25/11/2015)

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Executive Summary

In this report we describe the MFDB R package, the core product of the WP3 database system. Version 1.0 has been released as open source, and is available for modellers to use to help automate generation of GADGET models.

We then go into details of installation and give a brief outline of usage.

There was a workshop on using the system in December. Feedback from that was gathered and is presented later in this report, and finally an outline for further development is given.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Contents

Introduction: Description of MFDB system 5
Installation & Usage 6
Database workshop & feedback 6
Quick bugs that were fixed in the following weeks: 6
Changes that will require database changes and an MFDB 2.0 7
Conclusions 7
References 8
Annex: MFDB Package documentation 8
Introduction & Schema description 8
Importing data 8
Connect to database 8
Define areas & divisions 8
Define sampling types 9
Import temperature data 9
Import survey data 9
Querying data 9
Creating GADGET files 10
Appendix: Database Schema 11

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Introduction: Description of MFDB system

The database system developed in WP3 is not a system installed on a server, nor is it a program that you install and run locally. Instead, MFDB is an R [1] package, or "toolbox". The aim of this toolbox is to make it easy for a modeller to make an R [1] program to automatically download, process and create model input files based on input data.

This 'inside-out' toolbox approach has a number of advantages:

- Existing R packages can be used for access to institutional data, e.g. the DATRAS package [2]. Data files can be loaded using R's built-in file reading tools.
- The model file generation process can be infinitely customised, as case-study specific R code can be inserted at any point to modify the results. This also means that MFDB does not have to support every special case.
- MFDB can be integrated with any other useful steps in model generation, even running GADGET.

The R [1] programming language was chosen as the dominant language in the area, both in terms of existing tools and modeller familiarity.

In essence, an R [1] package defines a collection of functions. There are 2 groups of functions in MFDB at this point. The mfdb functions manage and query the database. In detail:

- **mfdb**: Connects to (& creates) a PostgreSQL database
- **mfdb_import_**: Import data into a PostgreSQL database
- **mfdb_sample_**, **mfdb_temperature_**, **mfdb_area_**: Query the database, restricting and grouping data
- **mfdb_unaggregated**, **mfdb_group**, **mfdb_interval**, ...: Functions to control grouping in query functions

The result of these functions can be then used with R's built-in abilities to export tables to spreadsheets, and then imported into any other system.

The gadget functions take the results of the MFDB queries and populate a GADGET model directory. In detail:

- **gadget_directory**: Creates a GADGET model directory
- **gadget_areafile**: Turn the results of some mfdb queries into a GADGET areafile
- **gadget_likelihood_component**: Turn the results of mfdb queries into a GADGET likelihood component
- **gadget_file**: Low-level GADGET model file input/output

One uses the mfdb_import functions to load your data into the database, then mfdb_sample to transform and aggregate this data. Then gadget_directory and gadget_likelihood_component can write this data out into the GADGET-specific file formats.

In December, MFDB 1.0 was released. This can be used to generate the GADGET areafile, as well as any likelihood component based on stock sampling. Any number of bootstrap samples of stocks can also be generated. There are examples within the package of using these features with the ICES DATRAS database, and MRI data.

In the description of work we aimed to have basic output for 2 modelling packages at this stage, decided to be GADGET and an R-based EwE in D3.1. However, the latter has not been released at the

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

time of writing. This not only means that we cannot provide support for it, but active modelling work is focused on GADGET models. As such, GADGET support is currently more advanced than it would have otherwise been, and MFDB's GADGET capabilities will be expanded faster over the coming months to keep pace with modelling activity and ensure Modellers are supported. EwE support can be revisited when software is available.

Regardless, for modelling tools without built-in support, all MFDB outputs can be exported to spreadsheets, to allow for manual imports.

Installation & Usage

Firstly, you will need to have installed R [1] onto your computer. If you have not done this already, instructions are available on the R [1] project home page.

Secondly, you will need to install the PostgreSQL [3] database. Instructions for specific operating systems are available on the MFDB Github page [4]. Most commonly, under Debian/Ubuntu you will have to run the following:

```
sudo apt-get install libpq-dev postgresql
```

Then create a user and database:

```
su - postgres psql
CREATE USER your_username
CREATE DATABASE mf
OWNER your_username;
```

Finally, you can install the package at the R [1] prompt using devtools [5]. The database schema will be created automatically the first time you connect.

```
# install.packages("devtools") devtools::install_github("mareframe/mfdb")
```

All functions have R help pages. For an overview of using MFDB, read the package help page using the package?mfdb command. This gives an overview of the concepts behind MFDB and examples of usage. It is reproduced in the annex "MFDB Package documentation".

There are example scripts that can import form ICES and MRI databases included in the MFDB package. These can be either run using the demo(package = 'mfdb') command, or you can look at the demo source code [6] on GitHub.

Database workshop & feedback

As part of the annual meeting in Aberdeen, there was a workshop on installing and using MFDB. The feedback was generally positive, the following comments were received

Quick bugs that were fixed in the following weeks:

As long as there were no changes to the database schema required, then the changes could be quickly made and added to a MFDB 1.1:

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

- When connecting to the database, it wasn't possible to change the database name from "mf" to something else
- There was no example of how the bootstrap sampling functions worked
- Temperature should be reported to 1 decimal place

MFDB 1.1 was released after the workshop, mid January, and addressed these concerns.

Changes that will require database changes and an MFDB 2.0

Changes in the database schema need to be handled with care, since there is potential for data loss. A user may not have the permission or desire to change it. Because of this, schema changes are only allowed between major versions, so it is clear which versions of the R package will work with your database.

MFDB 2.0 addresses the following concerns brought up in the workshop:

- You cannot vary gear and vessel types within a survey, this should be possible.
- Allow any indices to be stored, not just temperature. In particular, store acoustic data and then use it as an abundance index when querying stock samples.
- Support importing and querying of predator/prey stomach data, generating the ratio of stomachs present / stomachs as required by GADGET.

MFDB 2.0 is nearly finished, and should be released February 2015.

Conclusions

At this point MFDB demonstrates a database system which works end-to-end, i.e. can import data from at least 2 key data sources and transform this data into GADGET model files. At this point there is enough for it to be useful for a modeller working with GADGET.

MFDB 2.0 is nearly finished, at which point all concerns in the database workshop should be addressed.

As modellers are now using this tool, it is anticipated that many more feature requests will come in over 2015, which will be incorporated into future releases. There are several tasks for future versions of MFDB already known though. Namely:

- Generating GADGET refweight files
- Generating weighted standard deviation for data

This collaborative approach should ensure that the MFDB provides the features modellers require, and provide more GADGET support as their models become more intricate.

Support for other models will be started as soon as applicable, in particular:

- Reading of Atlantis data into MFDB, once stable Atlantis model output is available
- EwE model output, once the R version of EwE is available

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

References

[1]	http://www.r-project.org/
[2]	http://www.rforge.net/DATRAS/Tutorial.htm
[3]	http://postgresql.org/
[4]	http://github.com/mareframe/mfdb/
[5]	cran.r-project.org/web/packages/devtools/index.html
[6]	https://github.com/mareframe/mfdb/tree/1.x/demo

Annex: MFDB Package documentation

Introduction & Schema description

Before doing anything with **mfdb**, it is worth knowing a bit about how data is stored. Broadly, there are 2 basic types of table in **mfdb**, *taxonomy* and *measurement* tables.

The measurement tables store all forms of sample data supported, at the finest available detail. These are then aggregated when using any of the mfdb query functions. All measurement data is separated by case study, so multiple case studies can be loaded into a database without conflicts.

Taxonomy tables store all possible values for terms and their meaning, to ensure consistency in the data. A particularly important table is case_study, which lists the possible case studies that the database can store. When connecting to mfdb, you need to specify which case study you will be working with, so you access the correct data.

Most Taxonomies are the same across any installation, and their definitions are stored as data attached to this package. See mfdb-data for more information on these. Others, such as areacell and sampling_type are case study specific, and you will need to define your terms before you can import data.

Importing data

Unless you are working with a remote database, you will need to populate the database at least once before you are able to do any querying. The steps your script needs to do are:

Connect to database

Use the mfdb() function. This will create tables / populate taxonomies if necessary.

Define areas & divisions

mfdb models space in the following way:

areacell

> The finest level of detail stored in the database. Every measurement (e.g. temperature, length sample) is assigned to an areacell. This will generally correspond to ICES gridcells, however there is no requirement to do so. You might augment gridcell information with depth, or include divisions when the measurement doesn't correlate to a specific areacell.

division

> Collections of areacells, e.g. ICES subdivisions, or whatever is appropriate.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Finally, when querying, divisions are grouped together into named collections, for instance mfdb_group(north = 1:3, south = 4:6) will put anything in divisions 1–3 under an area named "north", 4–5 under an area named "south".

areacells and divisions are defined using mfdb_import_area() and mfdb_import_division() respectively. Before you can upload any measurements, you have to define the areacells that they will use.

Define sampling types

Any survey data can have a sampling type defined, which then can be used when querying data. If you want to use a sampling type, then define it using mfdb_import_sampling_type().

Import temperature data

At this point, you can start uploading actual measurements. The easiest of which is temperature. Upload a table of areacell/month/temperature data using mfdb_import_temperature()

Import survey data

Finally, import any survey data using mfdb_import_survey(). Ideally upload your data in separate chunks. For example, if you have length and age-length data, don't combine them in R, upload them separately and both will be used when querying for length data. This keeps the process simple, and allows you to swap out data as necessary

See mfdb_import_survey for more information or the demo directory for concrete examples.

Querying data

There are a selection of querying functions available, all of which work same way. You give a set of parameters, each of which can be a vector of data you wish returned, for instance year = 1998:2000 or species = c('COD').

If also grouping by this column (i.e. 'year', 'timestep', 'area' and any other columns given, e.g. 'age'), then the parameter will control how this grouping works, e.g. maturity_stage = mfdb_group(imm = 1, mat = 2:5) will result in the maturity_stage column having either 'imm' or 'mat'. These will also be used to generate GADGET aggregation files later.

For example, the following queries the temperature table:

```
defaults <- list(
        area = mfdb_group("101" = c(1011, 1012, 1013)),
        # Group months to create 2 timesteps for each year
        timestep = mfdb_timestep_quarterly,
        year = 1996:2005)
agg_data <- mfdb_temperature(mdb, defaults)
```

All functions will result in a list of data.frame result tables (generally only one, unless you requested bootstrapping). Each are suitable for feeding into a gadget function to output into model files. See mfdb_sample_count for more information or the demo directory for concrete examples.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Creating GADGET files

Finally, there are a set of functions that turn the output of queries into GADGET model files. These work on a gadget_directory object, which can either be an existing GADGET model to alter, or an empty / nonexistant directory.

Generally, the result of an mfdb query will be enough to create a corresponding GADGET file, for instance, the following will create a GADGET area file in your gadget directory:

```
gadget_dir_write(gd,gadget_areafile( size = mfdb_area_size(mdb, defaults)[[1]], temperature = mfdb_temperature(mdb, defaults)[[1]]))
```

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 613571

Appendix: Database Schema

Below is the database schema as will be implemented for MFDB 2.0. This includes the predator and prey relationship tables, as well as the existing sample table and taxonomy tables.