Community Modelling Workflow

Generalised Dissimilarity Modelling (GDM)

About this document

Generalised Dissimilarity Modelling (GDM) has become a widely-used method to link differences in composition between samples and explanatory environmental variables ("covariates"). GDMs may be fitted using any assemblage of covariates thought to be important in explaining the differences in composition between pairs of samples. Geographical distance can be optionally included as a covariate to account for "isolation by distance" (IBD) effects.

This document provides an outline of a prototype R-package, *cmGDM*, which is designed to implement GDM modelling as the first method within a new EcoCommons Community Modelling module. Fitting a GDM is performed by the R-package *gdm* available from the CRAN repository. *cmGDM* has been designed to implement a simple, robust workflow for basic fitting, review and reporting of GDMs.

This document describes the state of development and testing of the R-package *cmGDM* as at 30 November 2021.

Design concepts and principles

1. One streamlined protocol for data collation & upload

The R-package *gdm* provides a preparatory function, *formatsitepair()*, which allows users to present data components in four formats which are then compiled into a data table used by the function *gdm()* to fit the model. Although this flexibility is suitable for experienced R users, it can be very challenging for casual users to make appropriate choices among numerous possible combinations, and prepare data components in the required formats.

The path to data importation chosen during the development of this R-package is to follow procedures referred to as bioformat = 3 in the *gdm* package.

There are four versions or "bioformats" but they represent different combinations of pre-processed fundamental data tables needed to fit a GDM. The concept implemented in *cmGDM* is to begin with basic table formats and generate pre-processed or derived tables by asking users to supply necessary information. For example, users will be asked to load community data as either a presence-absence table, an abundance table, or a dissimilarity table. In the first two table types, they are standard tables which may be assembled by hand (in spreadsheets for example) or output by functions in other R-packages. For dissimilarity tables, many functions in other R-packages can produce dissimilarity tables including the packages *vegan* and several packages for analysing phylogenetic and population genetic data.

The expected advantages of this approach include:

- Users can easily assemble necessary data tables for each data component in which ever application they feel comfortable using (typically, a spreadsheet application)
- More robust data checking is possible enabling better guidance and support to users who need additional assistance
- No loss of modelling flexibility even though data input preparations are streamlined into a simplified protocol

2. Staged data entry

Implementation of the data entry steps in the prototype workflow is based on the idea of staged data entry. That is, users are asked to step through a sequence of uploading each data component. Feedback on errors encountered at each step is provided and progress to the next step is only possible when data quality checks have been passed for the current step.

3. Intercept data format errors

As mentioned above, data entry problems are intercepted at each step through the data upload sequence. These data quality checks duplicate many corresponding checks and error messages generated by the function *formatsitepair()*. However, the error messages returned by that function are not easily interpreted by less-experienced *R*-users. A key design decision was taken to perform data quality checks independent of *formatsitepair()* and generate error messages designed to be more informative for all users. This is particularly important given the design choice described in the next section.

4. Flexible data file formats

EcoCommons users have a wide range of skills and experience in preparing and manipulating data required for model fitting. More experienced users, particularly those users with good *R* skills, will be able to load data into an *R* session from a range of formats (e.g. Excel spreadsheet, comma-separated value or csv file, etc), re-organise and export data to a variety of formats suitable for use in fitting a GDM. Existing modelling modules in EcoCommons permit only the uploading of csv-formatted text files as data tables. A design decision taken in developing the prototype *R*-package *cmGDM* was allowing users to upload data tables in a wide selection of standard data file types. This is easily implemented in *R* because of the very extensive list of *R*-packages available in the CRAN repo which cater for many file formats.

The file formats currently implemented include:

- comma-separated value (csv) text files
- TAB-separated text files
- space-separated (tsv) text files
- openDocument (ods) format spreadsheets
- Excel spreadsheets (xls & xlsx) spreadsheets

For both spreadsheet types, users can select the sheet within a multi-sheet document (more correctly referred to as a 'workbook') by sheet number or name as this is a parameter available in the functions called from relevant packages to import spreadsheets.

5. Loosely OO design

A major design choice in the prototype package *cmGDM* was to use an object-oriented approach as much as possible. The package implements a simple S3 object of class "cm_experiment" which embodies the staged approach to data collation and quality checking.

Test Data set and testing process

An important aspect of implementing a flexible data entry front-end is rigorous testing of the many combinations or data states likely to be encountered in use. A test data suite was developed to facilitate comprehensive testing during development.

Data files Sets of small data files within each data component required to fit a GDM have been prepared to provide a cross-section of the errors which must be trapped. The files and the conditions they test are listed in the following table. The files are available in the *test_data* folder of the GitLab repository.

File names reflect the error condition contained within each file (where *error condition* includes the state *No error* as indicated by "OK" within the file name).

Test procedure A set of R-scripts is provided in the folder *test_scripts* to run through a tests for each data entry script. Naturally, they assume that package *cmGDM* has been downloaded and installed.

Because a trapped error invokes the **stop()** function, running each test involves highlighting the appropriate lines and then clicking on the "Run" option in R-Studio. An error-checking test is expected to emit the appropriate error message to the R-console.

Running the "OK" test should emit no message - just a return to the command prompt. Typing `thisExperiment$status` on the command line will show that the status flag for the data component under

test is **TRUE**. For example, immediately after instantiating a new experiment object we will see the following:

```
> library(cmGDM)
Loading required package: gdm
> thisExperiment <- cm_create_new_experiment("user123", "Blah")
> thisExperiment$status
      siteData_OK biologicalData_OK      covarData_OK predictionData_OK       modelFit_OK
            FALSE             FALSE             FALSE             FALSE             FALSE
>
```

After successfully loading site/sample data, we expect to see the following:

```
> thisExperiment$status
      siteData_OK biologicalData_OK      covarData_OK predictionData_OK       modelFit_OK
             TRUE             FALSE             FALSE             FALSE             FALSE
```

The workflow follows the sequence left to right along the *status* vector. Progression depends on successful completion of the preceding step with the exception that a GDM model can be fitted (by a call to *cm_run_gdm_experiment()*) immediately after a successful call to *cm_load_covars()*. Prediction covariates are only needed when a fitted GDM is being "projected" (i.e. used to make predictions). At the time of writing, that function has not been written although the function *cm_load_prediction_data()* has been written and tested.

A worked example

A small set of data files has also been included in the folder *worked_example*. An R-script is included to run the complete workflow, but of course each step can be run from the command line if desired.

The worked example illustrates the workflow to fit a very basic GDM.

Successful completion of the worked example should result in the following output in the console when the function *cm_gdm_summary()* is called:

```
> cm_gdm_summary(dataTestExp2)
------------------------------------------------
EcoCommons Community Modelling Module
GDM Model Summary
------------------------------------------------

Experiment name: Test with RF refugia data
Run date: 2021-11-29

Number of sites/samples: 703

Covariates:
  Geographical distance included: Yes
  Covariates: Number used = 20
    CHELSA_bio01
    CHELSA_bio02
    CHELSA_bio03
    CHELSA_bio04
    CHELSA_bio05
    CHELSA_bio06
    CHELSA_bio07
    CHELSA_bio08
    CHELSA_bio09
    CHELSA_bio10
```

```
    CHELSA_bio11
    CHELSA_bio12
    CHELSA_bio13
    CHELSA_bio14
    CHELSA_bio15
    CHELSA_bio16
    CHELSA_bio17
    CHELSA_bio18
    CHELSA_bio19
    Geographic

Model performance:

  Model deviance: 19.39
  Explained deviance: 6.83 (35.21%)
  NULL deviance: 20.81

Covariate importance:
    Covariate       % Contrib.
    ------------    -----------
      Geographic     0.00
    CHELSA_bio01     0.00
    CHELSA_bio02     0.00
    CHELSA_bio03     0.00
    CHELSA_bio04    28.44
    CHELSA_bio05     0.00
    CHELSA_bio06     7.74
    CHELSA_bio07     0.00
    CHELSA_bio08     2.86
    CHELSA_bio09    36.61
    CHELSA_bio10     0.00
    CHELSA_bio11     0.00
    CHELSA_bio12    12.36
    CHELSA_bio13     0.00
    CHELSA_bio14     0.00
    CHELSA_bio15     0.00
    CHELSA_bio16     0.00
    CHELSA_bio17     0.00
    CHELSA_bio18    11.99
    CHELSA_bio19     0.00
>
```