MadingleyR vignette

Contents

MadingleyR Installation

The MadingleyR package can be directly installed from R using the `devtools` or `remotes` R package. The following command installs the package using the remotes R package:

```r
# Load the remotes package
library('remotes') # or use library('devtools')

# Install the MadingleyR package
install_github('MadingleyR/MadingleyR', subdir='Package')
```

In addition to installing the MadingleyR dependencies (`rgdal`, `sp`, `data.table` and `raster`), the installation process also downloads the precompiled C++ executable, default spatio-temporal input layers and all other default input parameters and includes them in the installation folder. The following code can be used to make sure a previous version of MadingleyR in uninstalled:

```r
# Uninstall MadingleyR package, this code is not executed within the vignette
detach('package:MadingleyR', unload=TRUE)
remove.packages('MadingleyR')
```

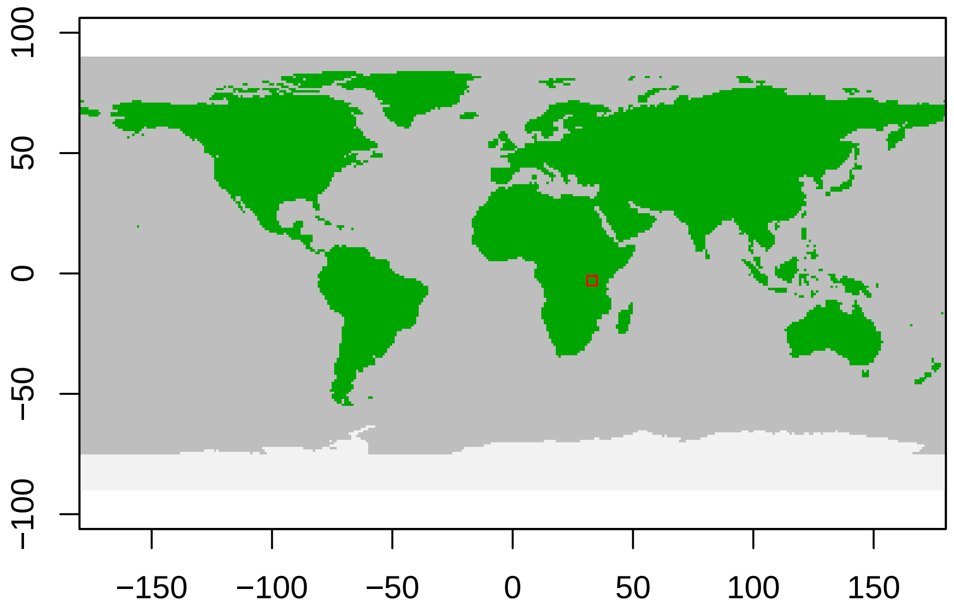
Model initialisation

The function `madingley_init()` initialises a model run by generating a cohort and stock data set. Both data sets are returned as data frames in a list object (here named: `mdata`) after the `madingley_init()` finishes. The cohort data set contains functional information for all cohorts (i.e. heterotrophs) needed to run a Madingley simulation (`mdata$cohorts`). The stock data set holds the functional information concerning the stocks (i.e. photo-autotrophs) (`mdata$stocks`). The generated data sets are based on the functional definitions defined in `cohort_def` (cohort definitions) and `stock_def` (stock definitions). `spatial_window` defines the boundaries of the spatial location, formatted as a vector containing four coordinates in the following order: 1) minimum longitude, 2) maximum longitude, 3) minimum latitude and 4) maximum latitude. The R code shown below illustrates the use of the `madingley_init()` function for an area that includes the Serengeti.

```r
# Load package
library(MadingleyR)

# Spatial model domain = c(min_long, max_long, min_lat, max_lat)
spatial_window = c(31, 35, -5, -1)

# plot the spatial window to check selection
plot_spatialwindow(spatial_window)
```



```r
# Prints possible input options to the R console
madingley_inputs( )
```

```
## possible input arguments are: input_type = "spatial inputs" OR "cohort definition" OR
"stock definition" OR "model parameters" OR "print options"
```

```
## Loading required package: rgdal
```

```
## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.0.4, released 2020/01/28
## Path to GDAL shared files: /usr/share/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 6.3.1, February 10th, 2020, [PJ_VERSION: 631]
## Path to PROJ shared files: /usr/share/proj
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading rgdal.
```

After checking which inputs are available, they can be loaded manually as shown below. However, if the default inputs suffice, it is possible to initialise the model without providing these inputs manually.

```r
# Load inputs manually
sptl_inp = madingley_inputs('spatial inputs')
chrt_def = madingley_inputs('cohort definition')
stck_def = madingley_inputs('stock definition')
mdl_prms = madingley_inputs('model parameters') # useful later for running the model
```

```r
# Initialise model the model using the pre-loaded inputs
mdata = madingley_init(spatial_window = spatial_window,
                       cohort_def = chrt_def,
                       stock_def = stck_def,
                       spatial_inputs = sptl_inp)
```

```
## Processing: realm_classification, land_mask, hanpp, available_water_capacity,
Ecto_max, Endo_C_max, Endo_H_max, Endo_O_max
## Processing: terrestrial_net_primary_productivity_1-12
## Processing: near-surface_temperature_1-12
## Processing: precipitation_1-12
## Processing: ground_frost_frequency_1-12
## Processing: diurnal_temperature_range_1-12
##
```

The returned `mdata` object will contain all cohorts and stocks (`data.frame`). In addition, the spatial window will be attached, making sure any consecutive model run will use the same spatial window.

Running the Madingley model

After generating cohorts and stocks, a simulation can be started using the madingley_run() function. The madingley_run() function requires the initialisation data set produced by the madingley_init() function. A typical Madingley simulation first requires a spin-up phase that allows ecosystem components to reach a stable state. This phase usually consists of a 100 to 1000-year model simulation without any model user induced changes. The code below runs the Madingley model for 100 years (years = 100) using the previously generated mdata object. The standard model input variables (e.g. cohort definitions, stock definitions, spatial inputs and/or model parameters) can be changed for madingley_run() via the following input parameters: cohort_def, stock_def, spatial_inputs, model_parameters.

```
# Run the Madingley model for 10 years
mdata2 = madingley_run(madingley_data = mdata,
                       years = 10,
                       cohort_def = chrt_def,
                       stock_def = stck_def,
                       spatial_inputs = sptl_inp,
                       model_parameters = mdl_prms)
```

Creating plots

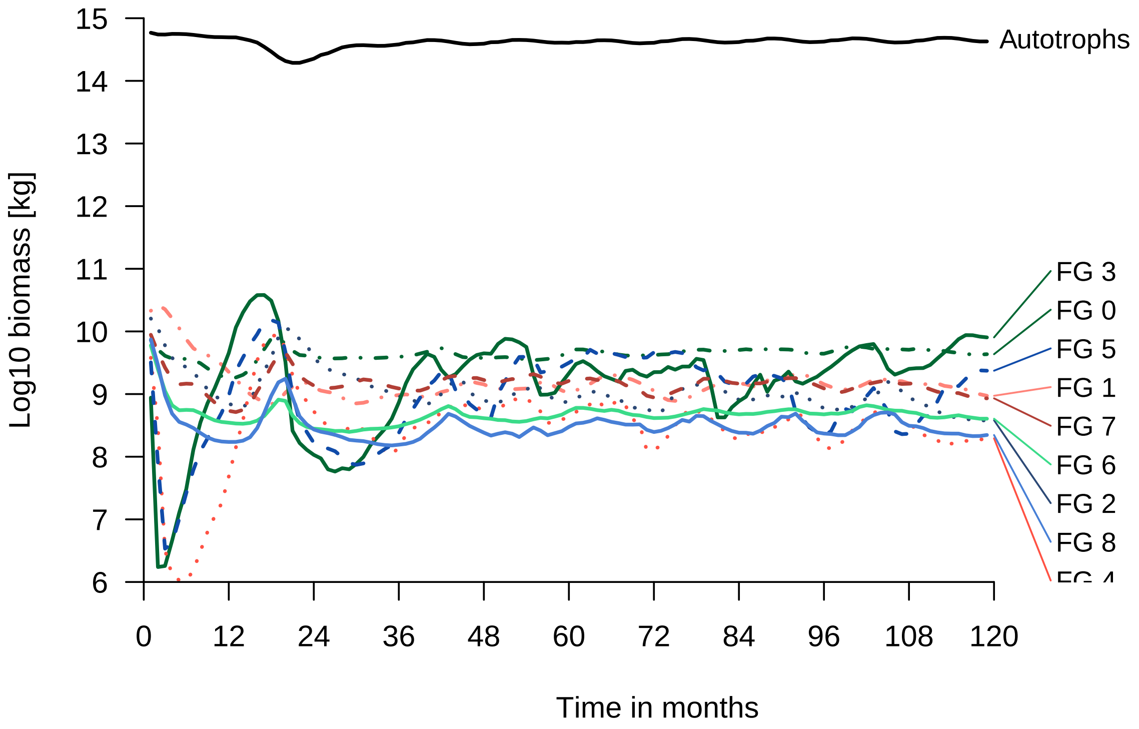
The `madingley_plot()` function creates several plots (see figures below) from the outputs generated by `madingley_run()`.

```r
# Create all MadingleyR plots
madingley_plot(mdata2)

# Check documentation to make individual plots
?madingley_plot
```

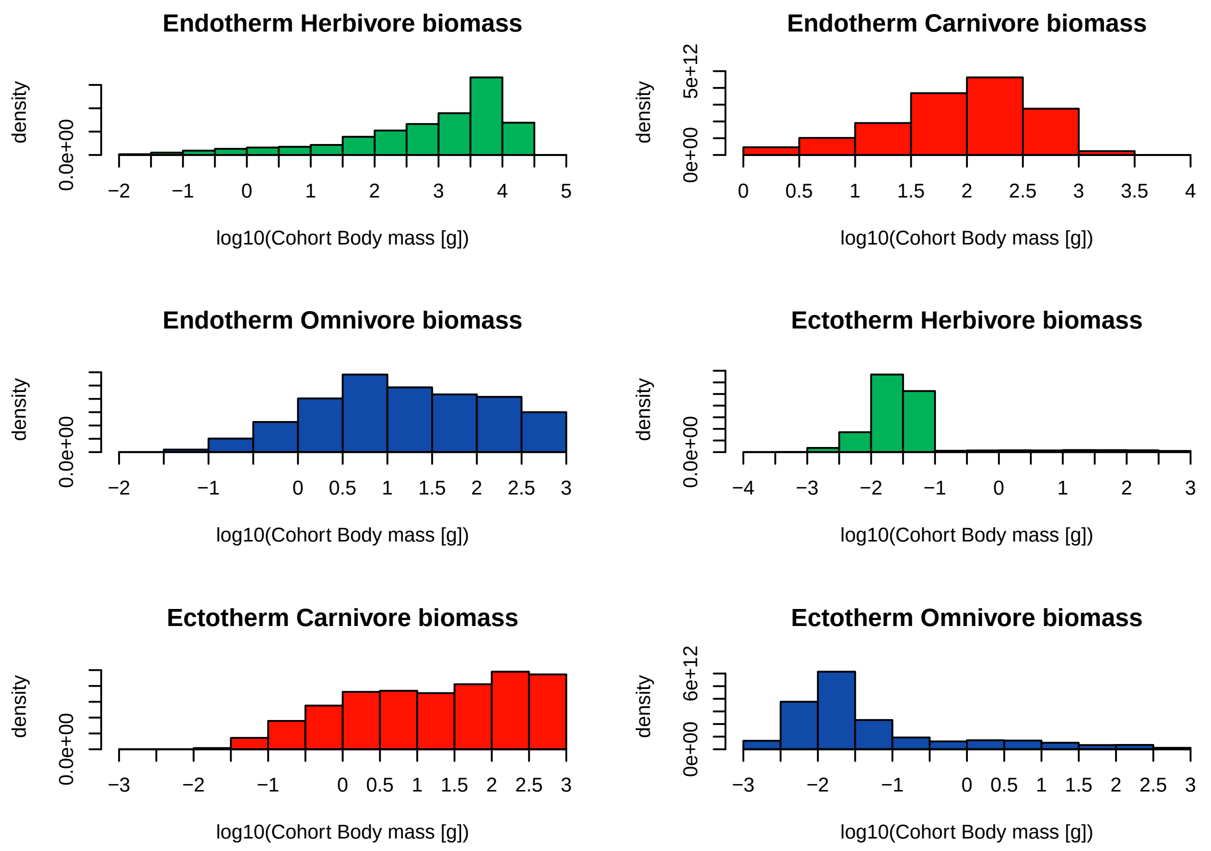
The individuals plots can also be produced one by one using the functions shown below.

```r
# Plot MadingleyR time lines
plot_timelines(mdata2)
```



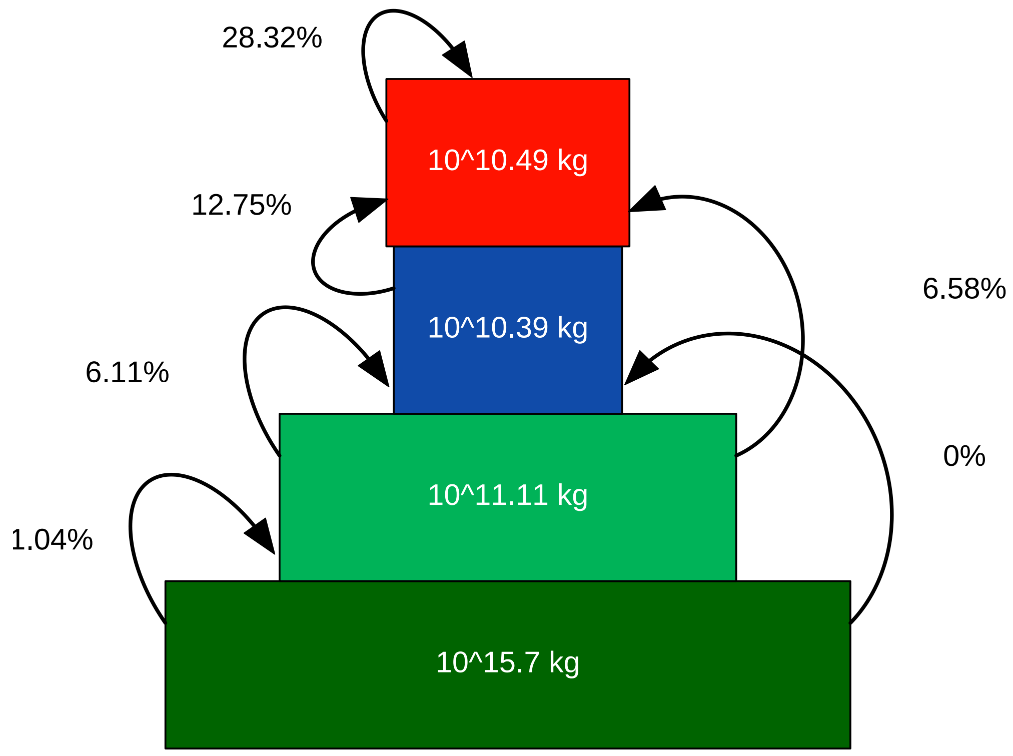
```r
# Plot MadingleyR body mass density
plot_densities(mdata2)
```

```
## loading inputs from: /tmp/RtmpBS7uze/madingley_outs_23_02_21_20_48_45/
```



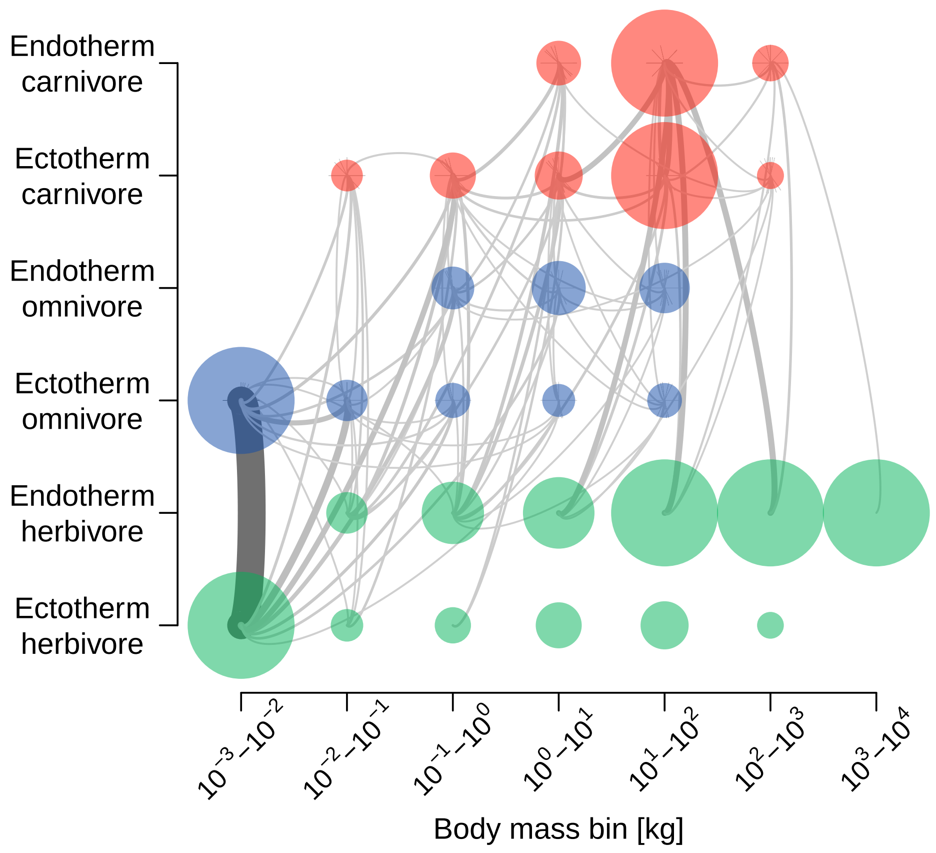
```
# Plot MadingleyR trophic pyramid
plot_trophicpyramid(mdata2)
```

```
## loading inputs from: /tmp/RtmpBS7uze/madingley_outs_23_02_21_20_48_45/
```



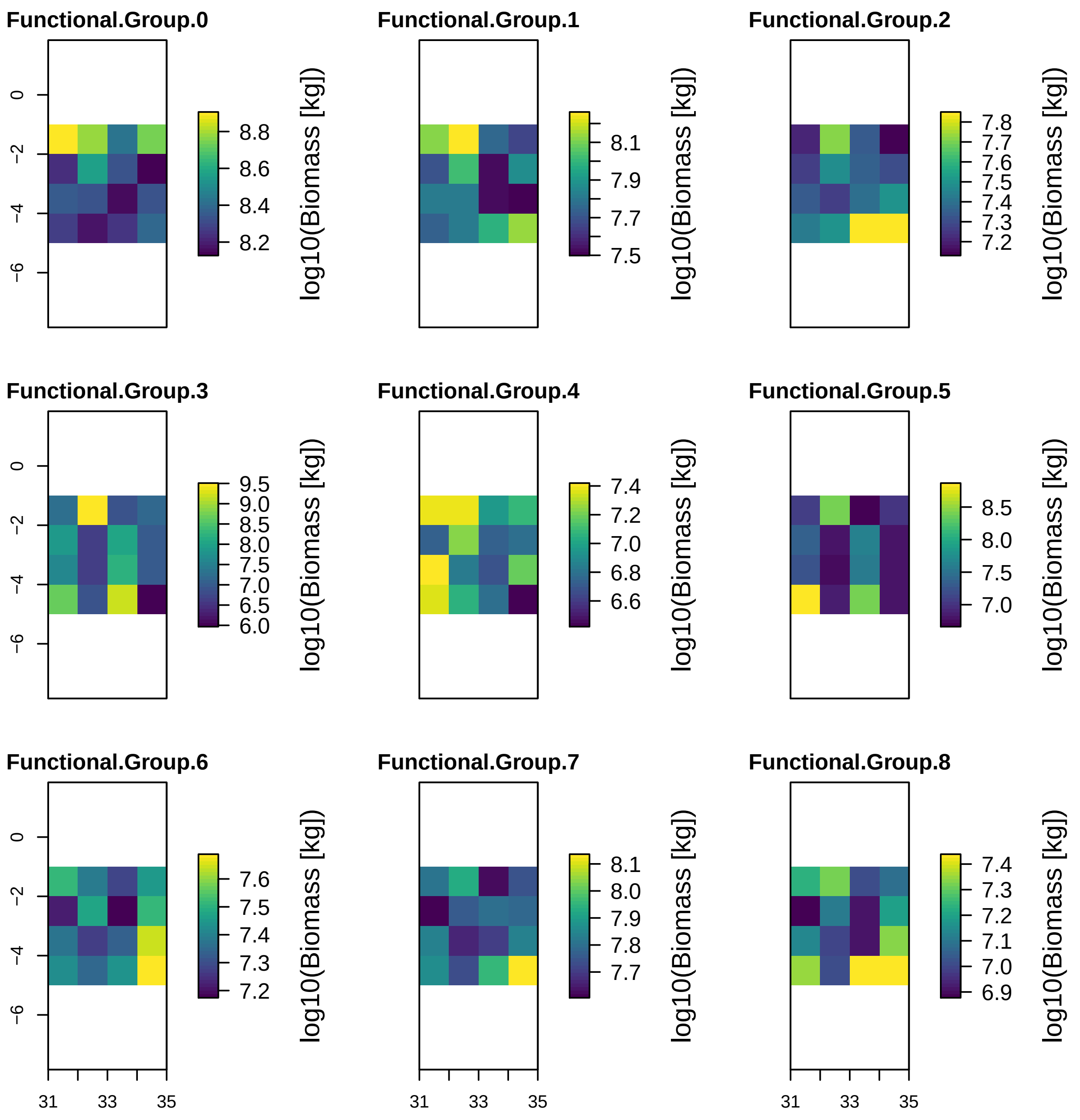
```
# Create MadingleyR log10-binned food-web plot
plot_foodweb(mdata2, max_flows = 5)
```

```
## loading inputs from: /tmp/RtmpBS7uze/madingley_outs_23_02_21_20_48_45/
```



```
# Plot MadingleyR spatial biomass
plot_spatialbiomass(mdata2, functional_filter = TRUE)
```

```
## loading inputs from: /tmp/RtmpBS7uze/madingley_outs_23_02_21_20_48_45/
```


Functional.Group.0
Functional.Group.1
Functional.Group.2
Functional.Group.3
Functional.Group.4
Functional.Group.5
Functional.Group.6
Functional.Group.7
Functional.Group.8
log10(Biomass [kg])
0
−2
−4
−6
31
33
35