R package rodeoFABM: Basic Use and Sample Applications

johannes.feldbauer@tu-dresden.de

2020-03-17

Contents

1 Main features of `rodeoFABM`

The package `rodeoFABM` is a colletion of small tools to help create water quality models that can be coupled to physical host models using the `FABM` interface (Bruggeman and Bolding (2014)). As the name suggests it is heaviely influenced by the R package `rodeo` (Kneis, Petzoldt, and Berendonk (2017)). The principle idea is to have a system that:

- Helps users that don't have the technical know how
- make model adaptation, communikation, and maintanace easy

Therfore the water quality model is written in the standard Peterson matrix notation and stored in text files or spread sheets. The package `rodeoFABM` automatically generates `FABM` specific FORTRAN code from these files and can automatically compile `GOTM` coupled with the newly created model, as well as .yaml control files for the water quality model.

2 Installation and requirements

In order to fully use `rodeoFABM` and run the examples some tools are neede:

- The GNU compilers
- GNU Make

- GNU CMake
- Rdevtools
- R packages: `readODs`, `gotmtools`

The package `rodeoFABM` can be installed from github using:

```
library("devtools")
install_github("JFeldbauer/rodeoFABM")
```

3 Basic use

A simple example that is extended along the way

3.1 first example (how it works)

To demonstrate the workflow we will use a very simple model that is provided in the package. The files are contained in the package and can be loaded using:

```
# copy example ods file
example_model <- system.file("extdata/simple_model.ods", package = "rodeoFABM")
file.copy(from = example_model, to = ".", recursive = TRUE)
```

Now we can read in the tables with the declaration of the state variables, model parameters, used functions and external dependencies, process rate descriptions, and the stoichiometry matrix.

```
library(readODS)

# read in example ods file
odf_file <- "simple_model.ods"
vars <- read_ods(odf_file, 1)
pars <- read_ods(odf_file, 2)
funs <- read_ods(odf_file, 3)
pros <- read_ods(odf_file, 4)
stoi <- read_ods(odf_file, 5)
```

From these we can now generate FORTRAN files

```
library(rodeoFABM)

# generate fabm code
gen_fabm_code(vars,pars,funs,pros,stoi,"simple_model.f90",diags = TRUE)
```

And compile GOTM-FABM with them. Therfore first we need to clone the lake branche of GOTM-FABM from github and prepare the build process using cmake. This needs only to be done once, using the function `clone_GOTM()`:

```
# clone github repo
clone_GOTM(build_dir = "build", src_dir = "gotm_src")
```

Now we can build GOTM-FABM with our own model using:

```
# build GOTM
build_GOTM(build_dir = "build",fabm_file = "simple_model.f90",
           src_dir = "gotm_src/extern/fabm/src/models/tuddhyb/rodeo")
```

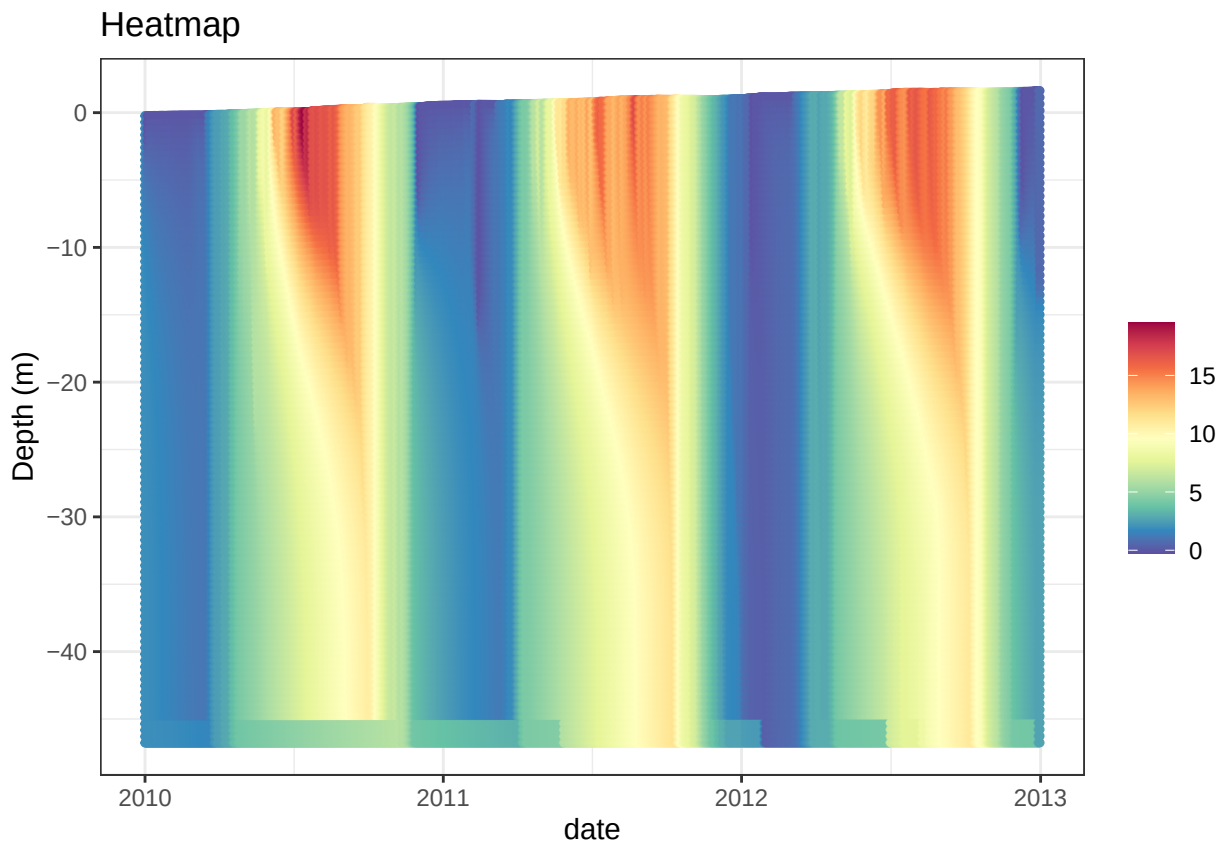
After copying the example gotm.yaml (the GOTM controll fille), the exymple hypsograph, and the example forcing data, we finally can run GOTM-FABM with our own small model using:

```
# copy example gotm.yaml
yaml <- system.file("extdata/gotm.yaml", package = "rodeoFABM")
file.copy(from = yaml, to = ".", recursive = TRUE)
# write hypsograph
write.table(hypsograph, "hypsograph.dat", sep = "\t", row.names = FALSE,
            quote = FALSE)
# write meteo data
write.table(meteo_file, "meteo_file.dat", sep = "\t", row.names = FALSE,
            quote = FALSE)
# run gotm
system2("./gotm")
```

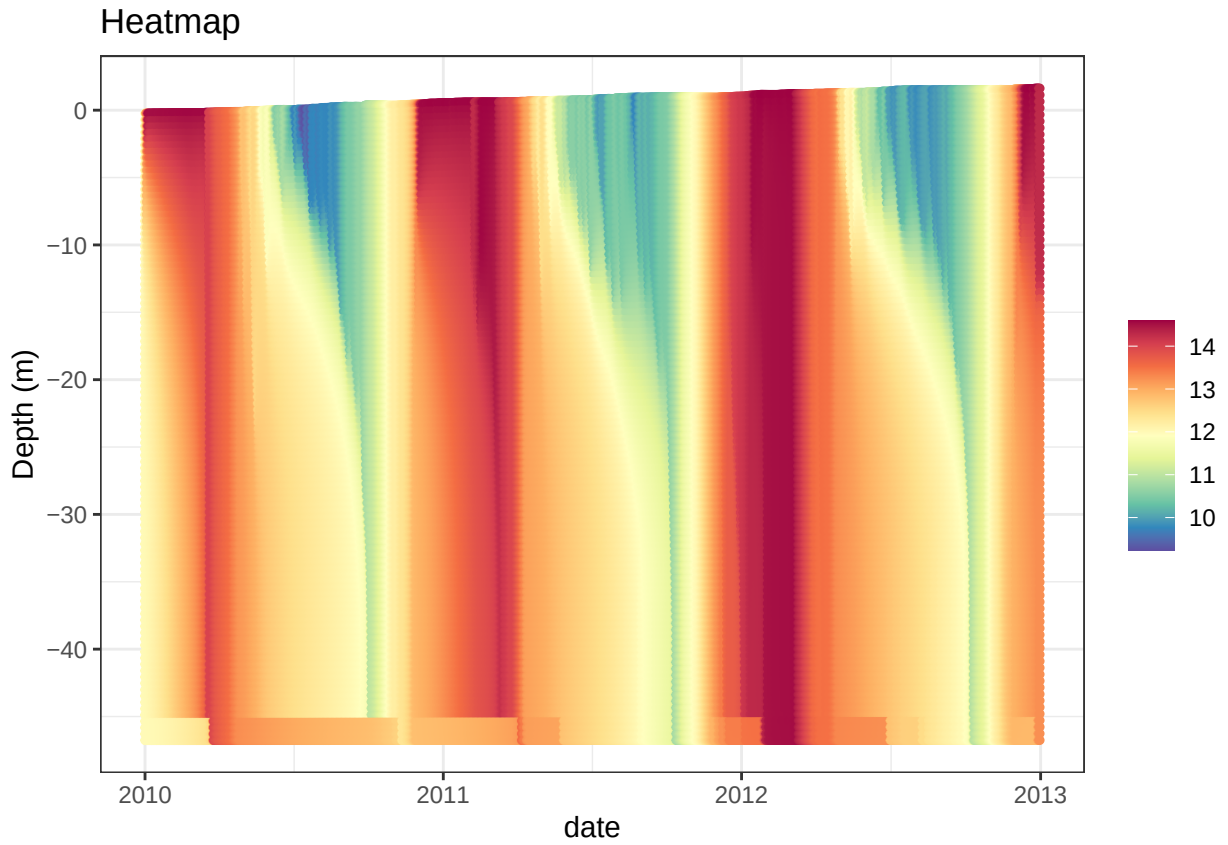
We can plot the results e.g. using the `gotmtools` library, which can be installed from the AEMON-J github using:

```
devtools:::install_github("aemon-j/gotmtools")
```

```
library(gotmtools)
# plot temperature
plot_vari("output.nc", "temp")
```




```
# plot oxygen
plot_vari("output.nc", "rodeo_C_O2")
```



These are the essential steps used. In the next section we will go into the details of building a model using the `rodeoFABM` package step by step.

3.2 create own model

In order to demonstrate the necessary steps we will create a very simple phytoplankton-nutrients model and step by step add more processes.

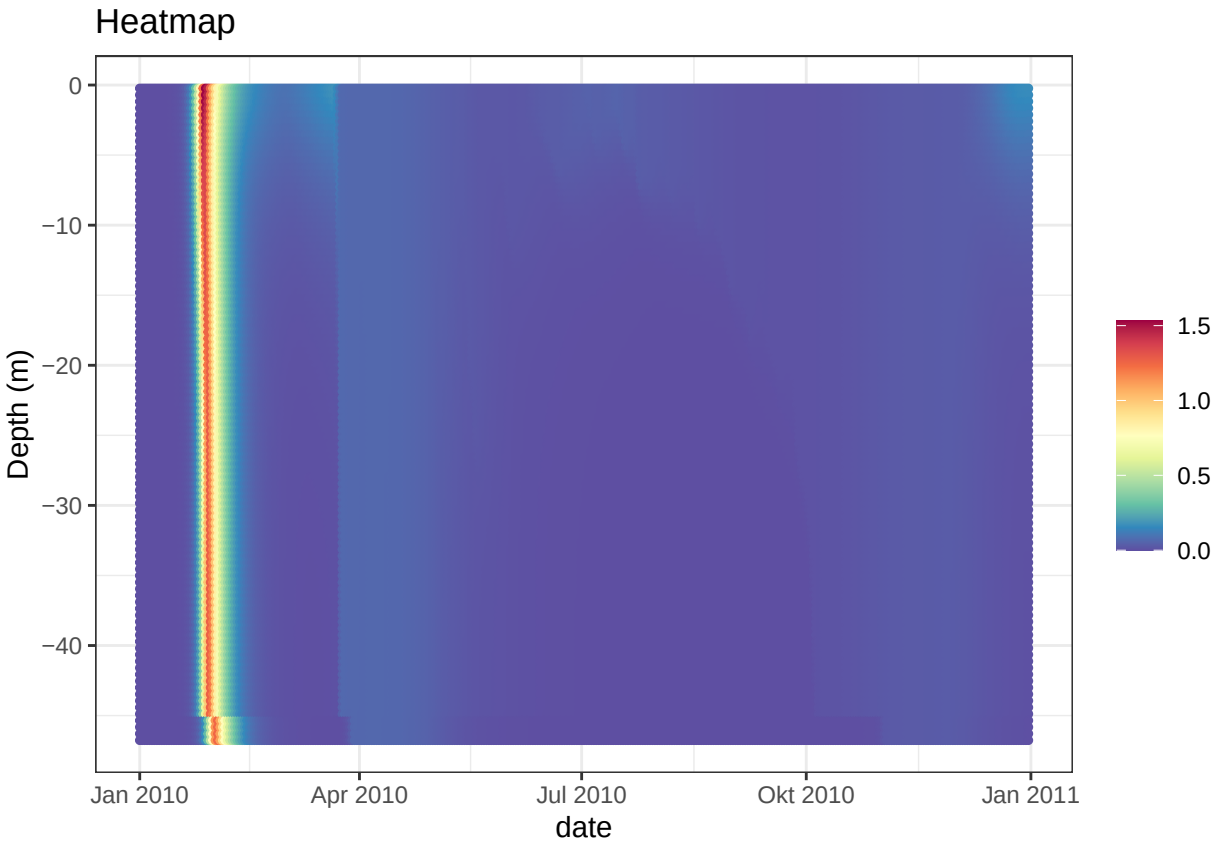
copy all necessary files to run GOTM. We use the same meteo and hypso files as in the first example

```
# GOTM controll fille
yaml <- system.file("extdata/examples/gotm.yaml",
                    package = "rodeoFABM")
file.copy(from = yaml, to = ".", recursive = TRUE)
# inflow hydrological data
infl <- system.file("extdata/examples/inflow_m.dat",
                    package = "rodeoFABM")
file.copy(from = infl, to = ".", recursive = TRUE)
# inflow nutient data
nut <- system.file("extdata/examples/inflow_wq_m.dat",
                    package = "rodeoFABM")
file.copy(from = nut, to = ".", recursive = TRUE)
# outflow data
out <- system.file("extdata/examples/outflow.dat",
                    package = "rodeoFABM")
file.copy(from = out, to = ".", recursive = TRUE)
```

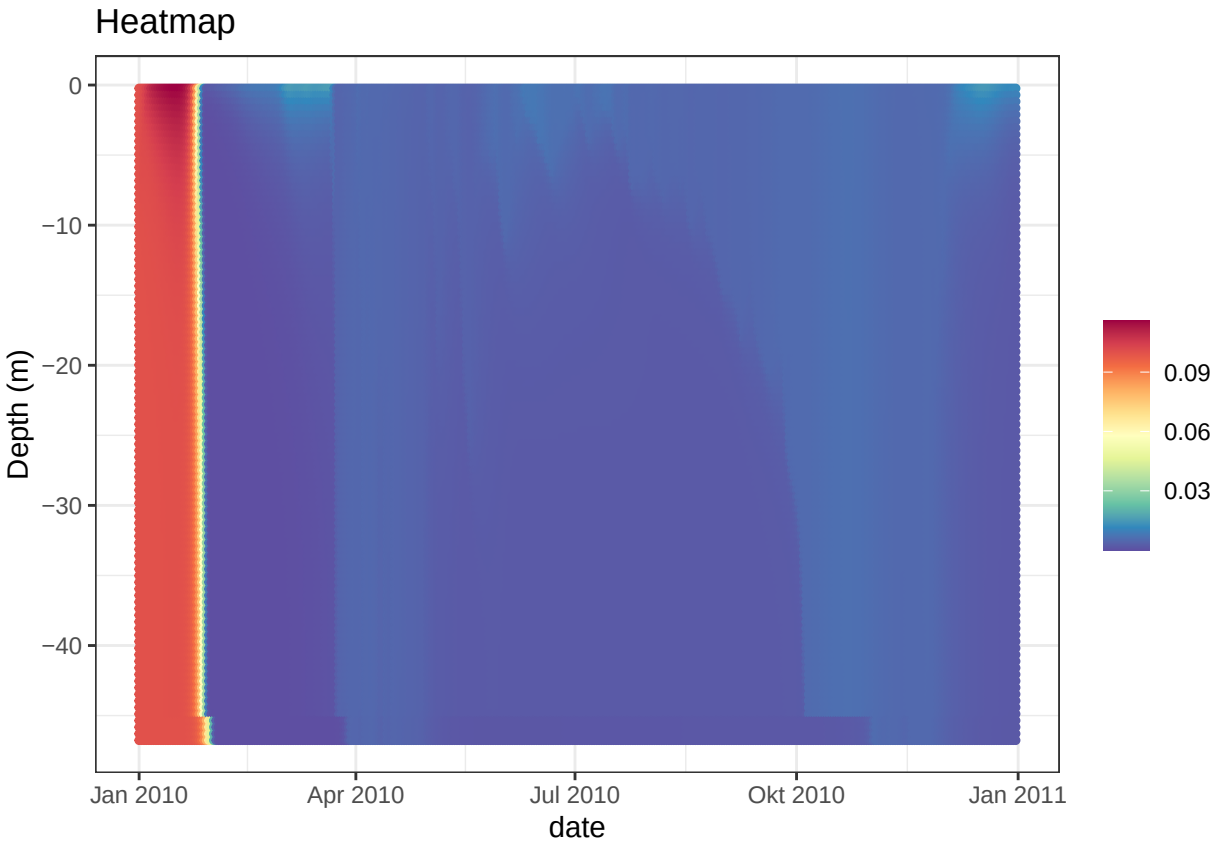
3.2.1 Inflow

Add Substrat if comming in from the inflow

```
# copy the spread sheet
ods <- system.file("extdata/examples/simple_alg.ods",
                   package = "rodeoFABM")
file.copy(from = ods, to = ".", recursive = TRUE)

# read in first simple model
vars <- read_ods("simple_alg.ods", sheet = "vars")
pars <- read_ods("simple_alg.ods", sheet = "pars")
funs <- NULL
pros <- read_ods("simple_alg.ods", sheet = "pros")
stoi <- read_ods("simple_alg.ods", sheet = "stoi")

# create the fabm code
gen_fabm_code(vars, pars, funs, pros, stoi, "model_1.f90")

# build GOTM with the model
build_GOTM(build_dir = "build", src_dir = "gotm_src/extern/fabm/src/models/tuddhyb/rodeo/",
           fabm_file = "model_1.f90")

# run the model
system2("./gotm")

# plot the variables
plot_vari("output.nc", "rodeo_C")
```




```
plot_vari("output.nc", "rodeo_HPO4")
```

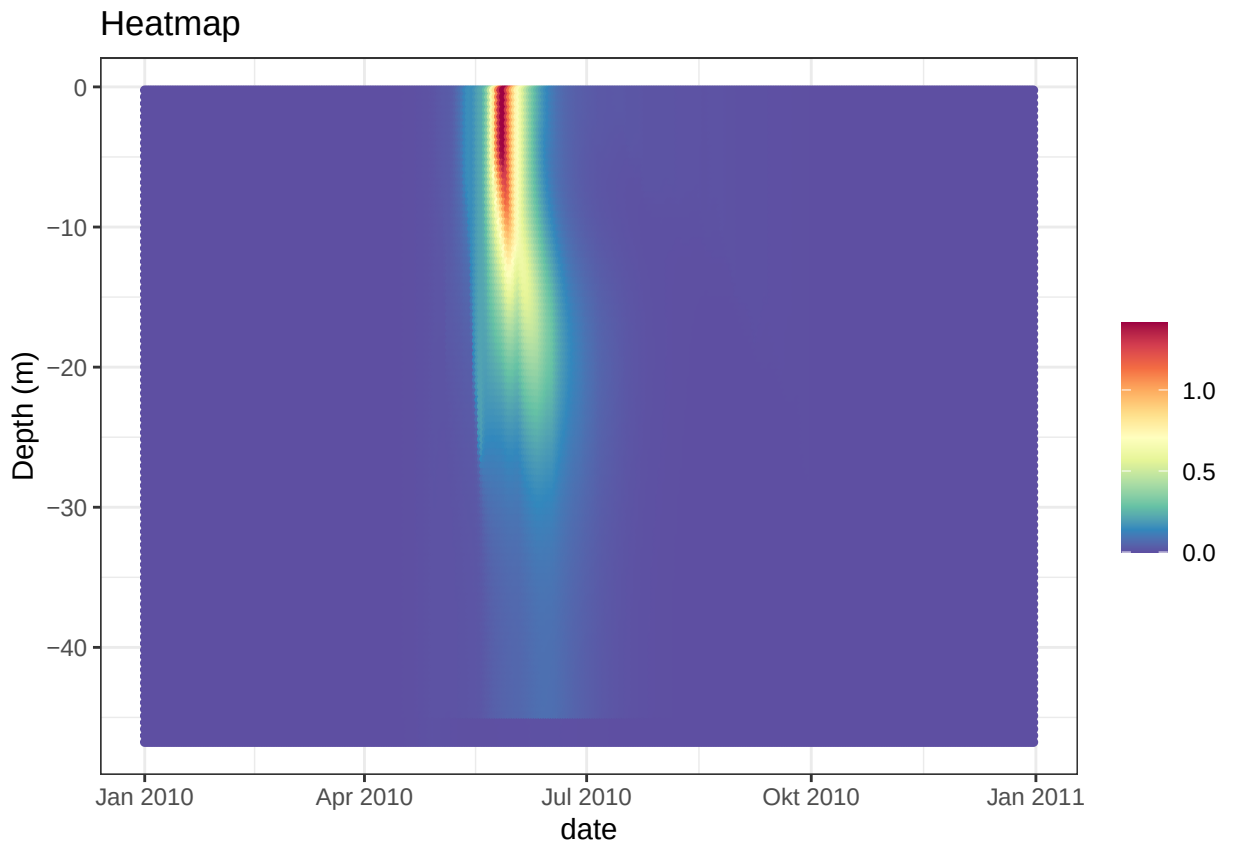


3.2.2 Getting forcing data from the host model

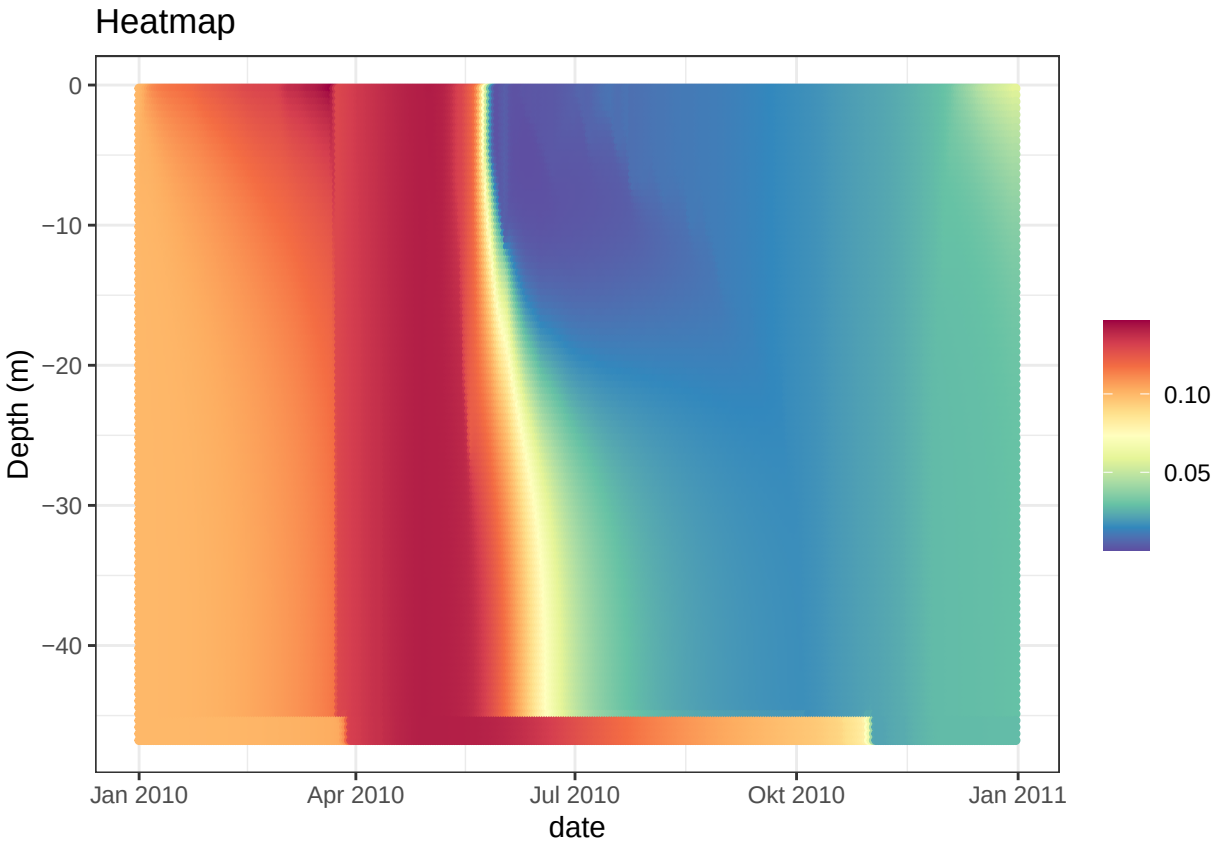
PAR dependency of growth

```
ods <- system.file("extdata/examples/simple_alg_par.ods",
                   package = "rodeoFABM")
file.copy(from = ods, to = ".", recursive = TRUE)

# read in first simple model
vars <- read_ods("simple_alg_par.ods", sheet = "vars")
pars <- read_ods("simple_alg_par.ods", sheet = "pars")
funs <- read_ods("simple_alg_par.ods", sheet = "funs")
pros <- read_ods("simple_alg_par.ods", sheet = "pros")
stoi <- read_ods("simple_alg_par.ods", sheet = "stoi")

# create the fabm code
gen_fabm_code(vars, pars, funs, pros, stoi, "model_2.f90")

# build GOTM with the model
build_GOTM(build_dir = "build", src_dir = "gotm_src/extern/fabm/src/models/tuddhyb/rodeo/",
           fabm_file = "model_2.f90")

# run the model
system2("./gotm")
```

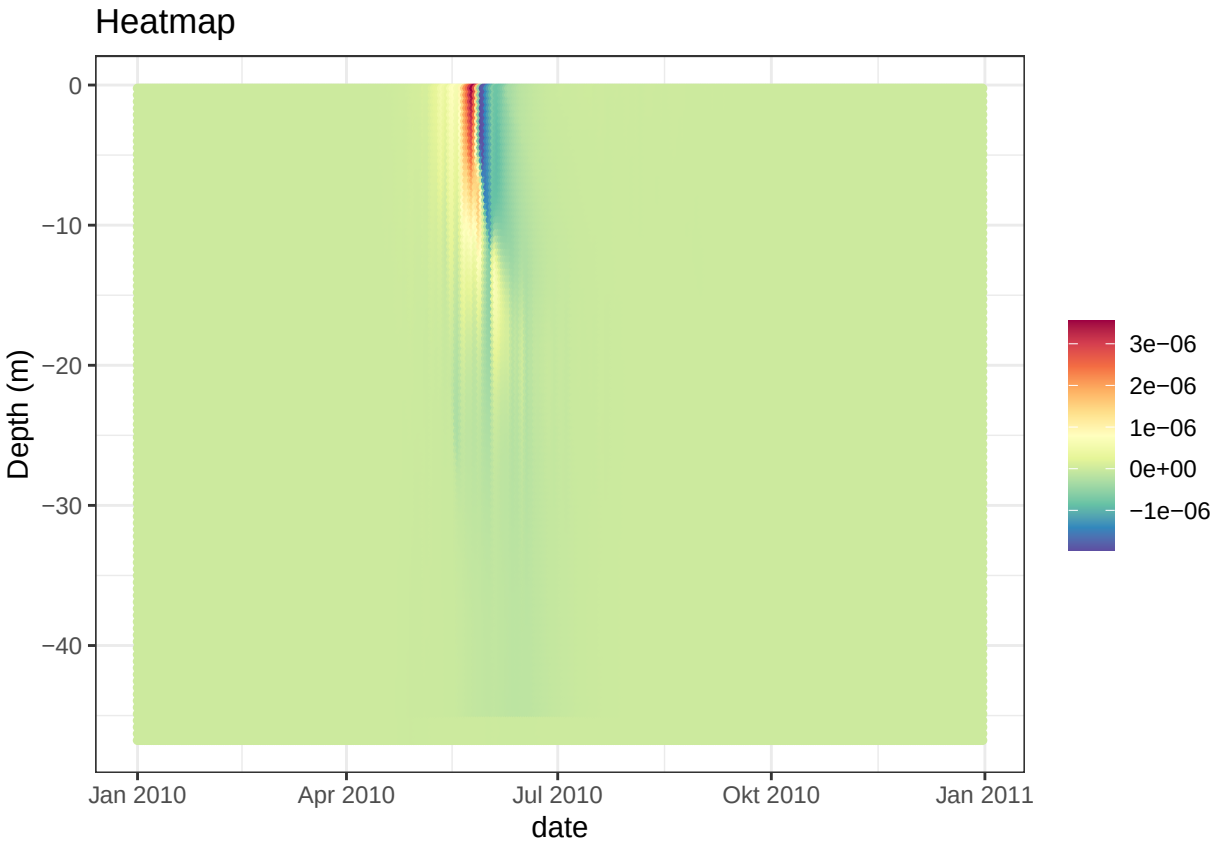
```
# plot the variables
plot_vari("output.nc", "rodeo_C")
```




```
plot_vari("output.nc", "rodeo_HPO4")
```




```
# also plot net. growth
growth <- get_vari("output.nc", "rodeo_growth")
death <- get_vari("output.nc", "rodeo_death")
net_growth <- cbind(growth$Datetime, growth[, -1] - death[, -1])
z  <-  get_vari("output.nc", var = "z", incl_time = TRUE)
long_heatmap(wide2long(net_growth, z))
```

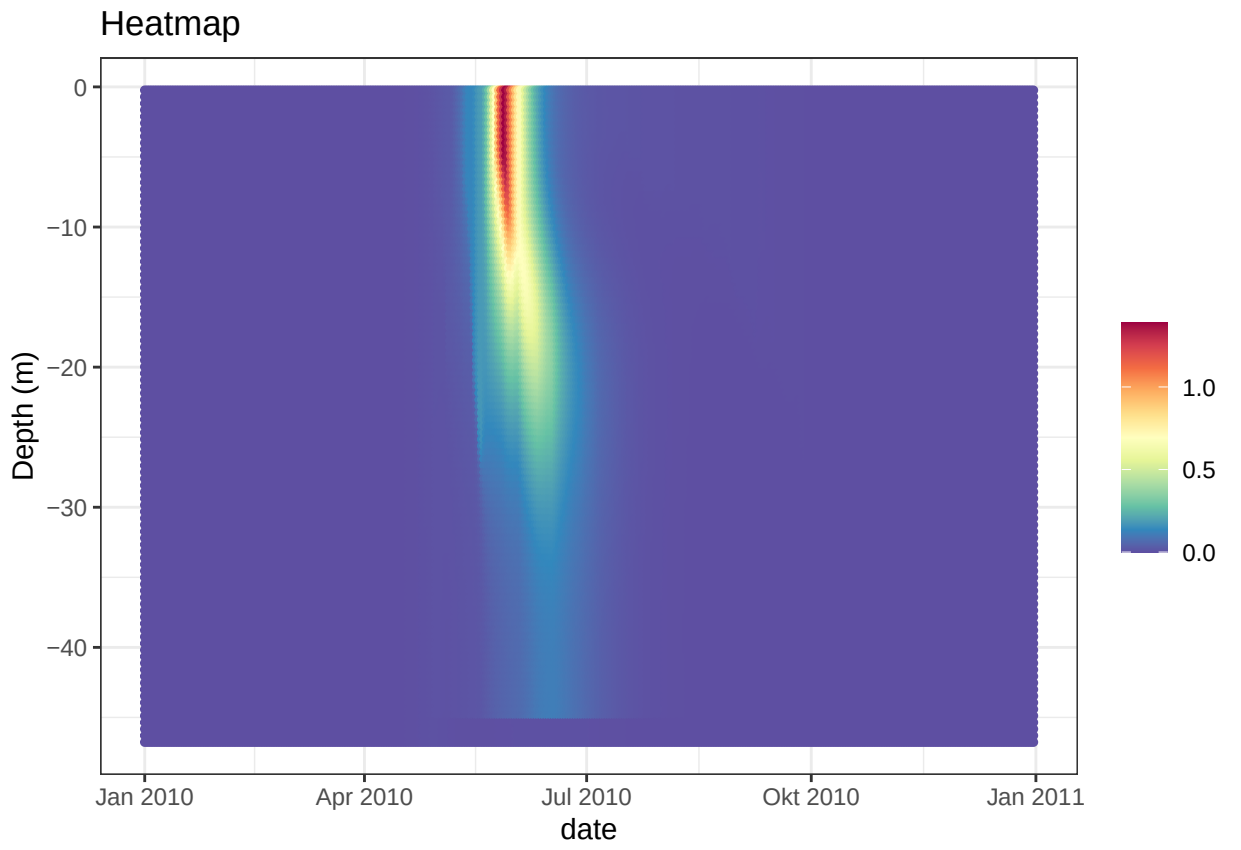


3.3 sedimentation

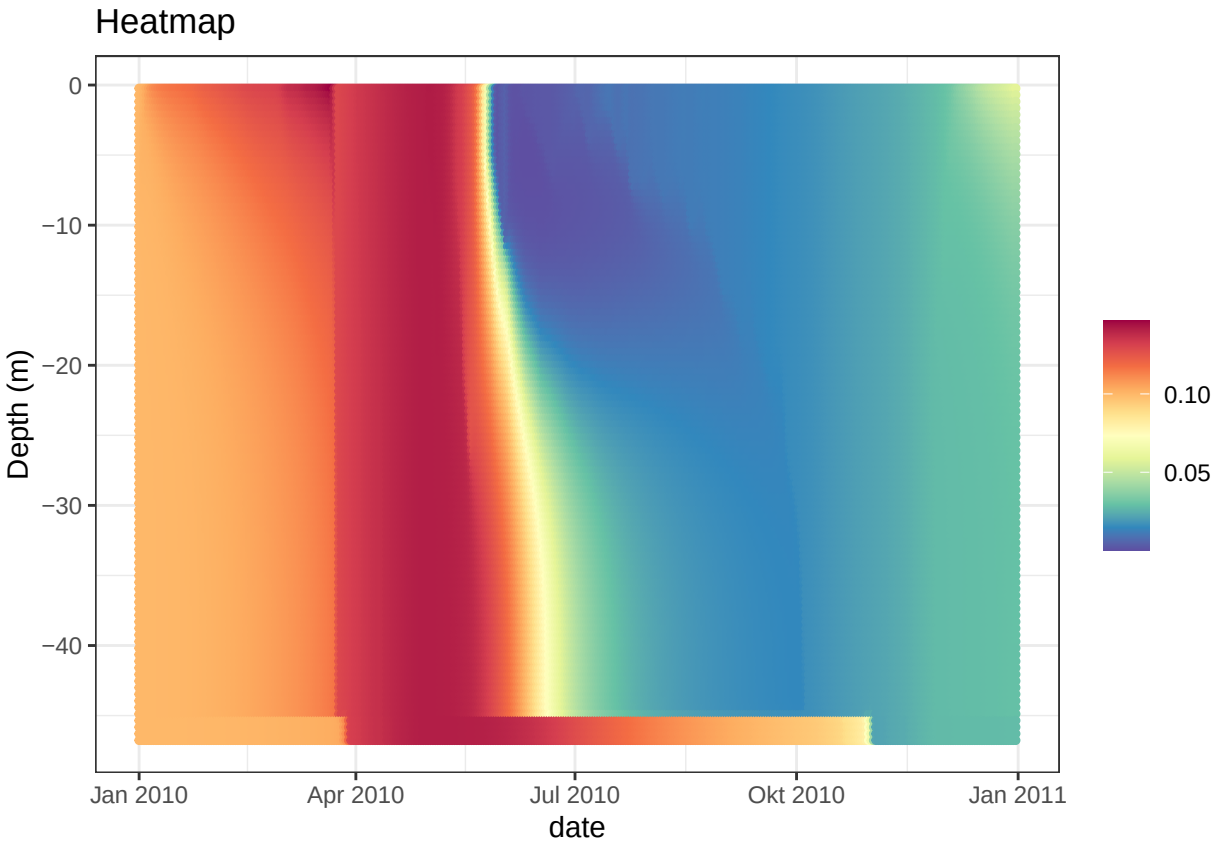
algae are settling

```
ods <- system.file("extdata/examples/simple_alg_par_sed.ods",
                   package = "rodeoFABM")
file.copy(from = ods, to = ".", recursive = TRUE)

# read in first simple model
vars <- read_ods("simple_alg_par_sed.ods", sheet = "vars")
pars <- read_ods("simple_alg_par_sed.ods", sheet = "pars")
funs <- read_ods("simple_alg_par_sed.ods", sheet = "funs")
pros <- read_ods("simple_alg_par_sed.ods", sheet = "pros")
stoi <- read_ods("simple_alg_par_sed.ods", sheet = "stoi")

# create the fabm code
gen_fabm_code(vars, pars, funs, pros, stoi, "model_3.f90")

# build GOTM with the model
build_GOTM(build_dir = "build", src_dir = "gotm_src/extern/fabm/src/models/tuddhyb/rodeo/",
           fabm_file = "model_3.f90")

# run the model
system2("./gotm")
```

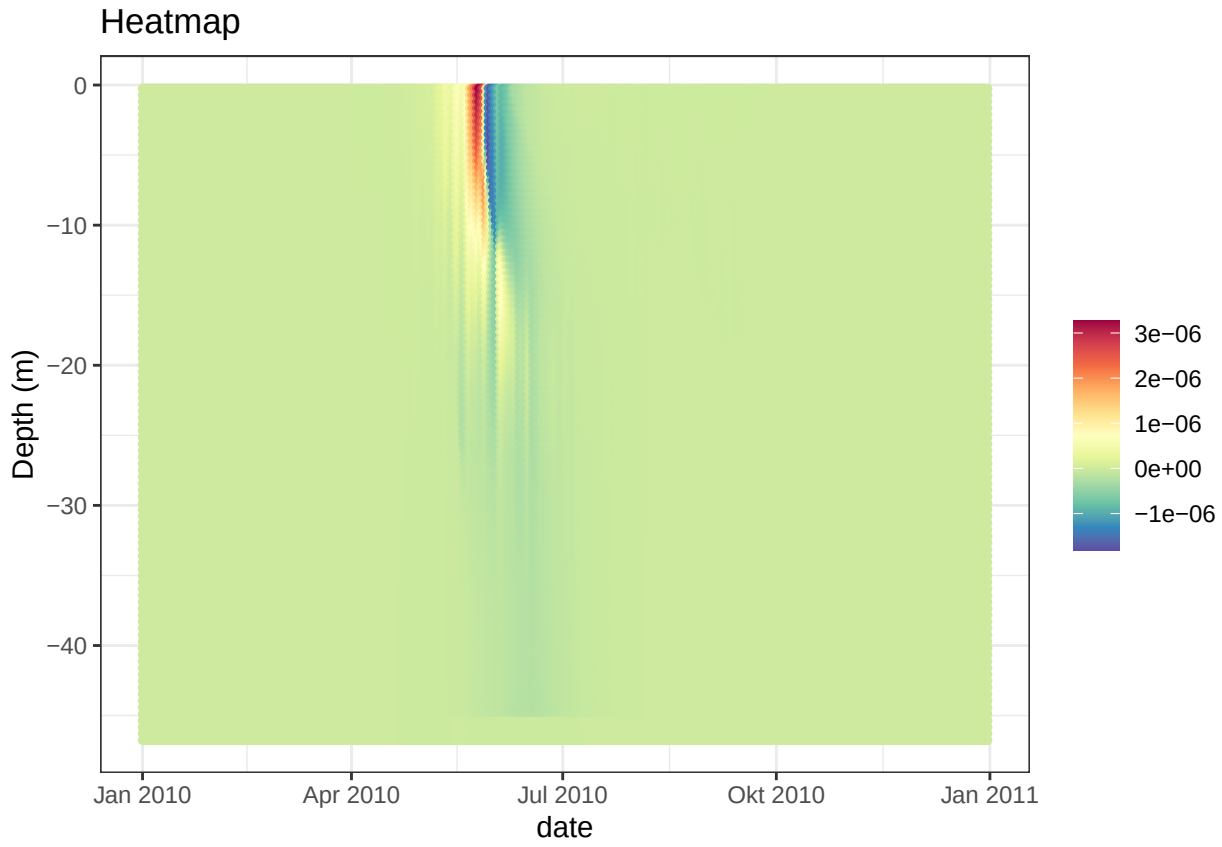
```
# plot the variables
plot_vari("output.nc", "rodeo_C")
```




```
plot_vari("output.nc", "rodeo_HPO4")
```




```
# also plot net. growth
growth <- get_vari("output.nc", "rodeo_growth")
death <- get_vari("output.nc", "rodeo_death")
net_growth <- cbind(growth$Datetime, growth[, -1] - death[, -1])
z  <-  get_vari("output.nc", var = "z", incl_time = TRUE)
long_heatmap(wide2long(net_growth, z))
```

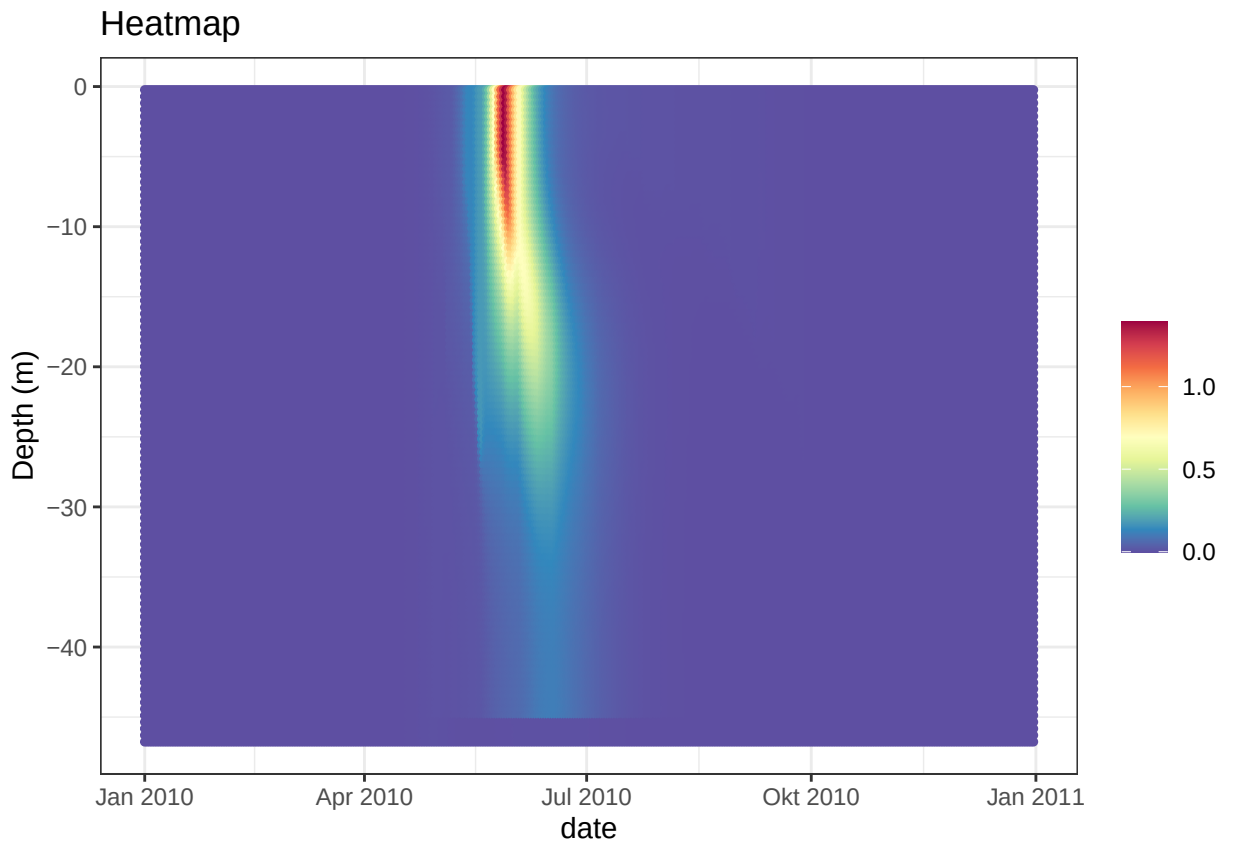


3.3.1 Processes at the upper and lower boundaries (surface and sediment)

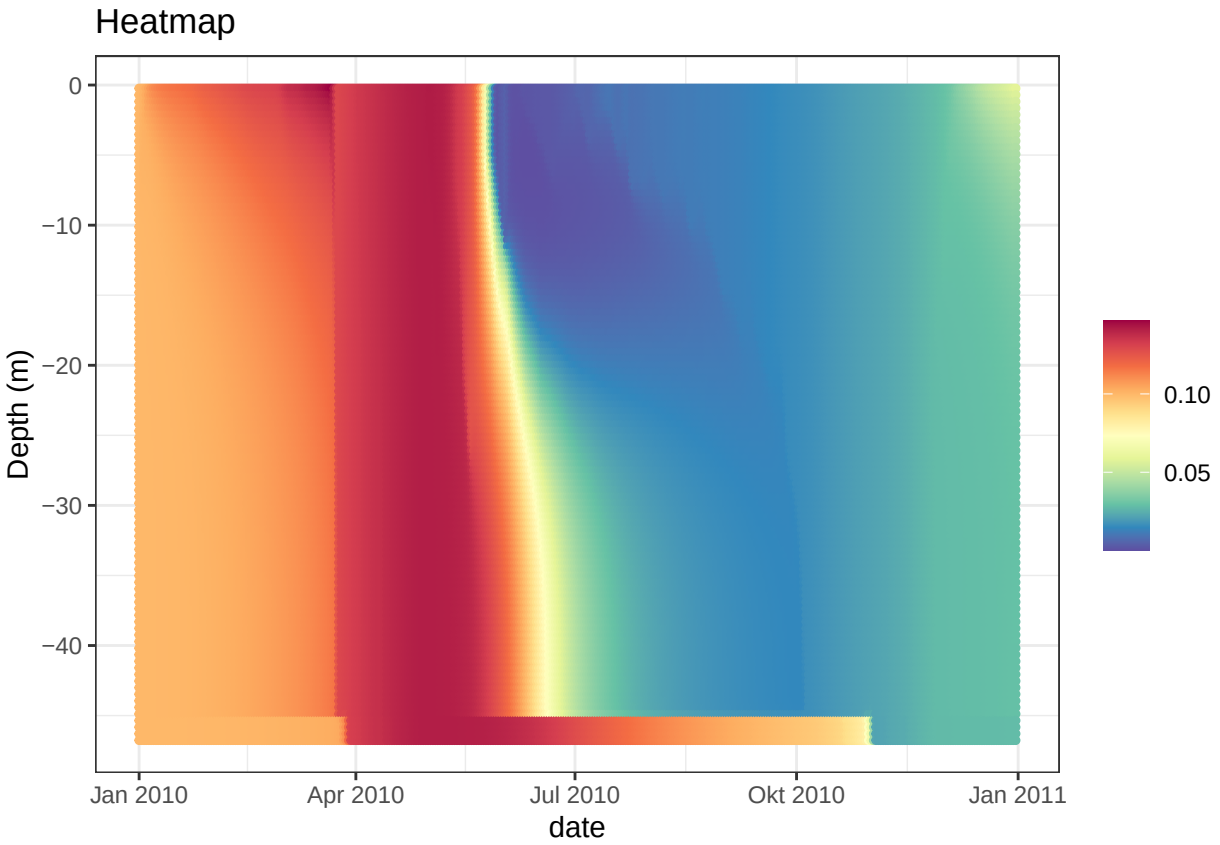
Add oxygen along with surface exchange and a constant oxygen consumption

```
ods <- system.file("extdata/examples/simple_alg_O2.ods",
                   package = "rodeoFABM")
file.copy(from = ods, to = ".", recursive = TRUE)
# read in first simple model
vars <- read_ods("simple_alg_O2.ods", sheet = "vars")
pars <- read_ods("simple_alg_O2.ods", sheet = "pars")
funs <- read_ods("simple_alg_O2.ods", sheet = "funs")
pros <- read_ods("simple_alg_O2.ods", sheet = "pros")
stoi <- read_ods("simple_alg_O2.ods", sheet = "stoi")

# create the fabm code
gen_fabm_code(vars, pars, funs, pros, stoi, "model_4.f90")

# build GOTM with the model
build_GOTM(build_dir = "build", src_dir = "gotm_src/extern/fabm/src/models/tuddhyb/rodeo/",
           fabm_file = "model_4.f90")

# run the model
system2("./gotm")

# plot the variables
```

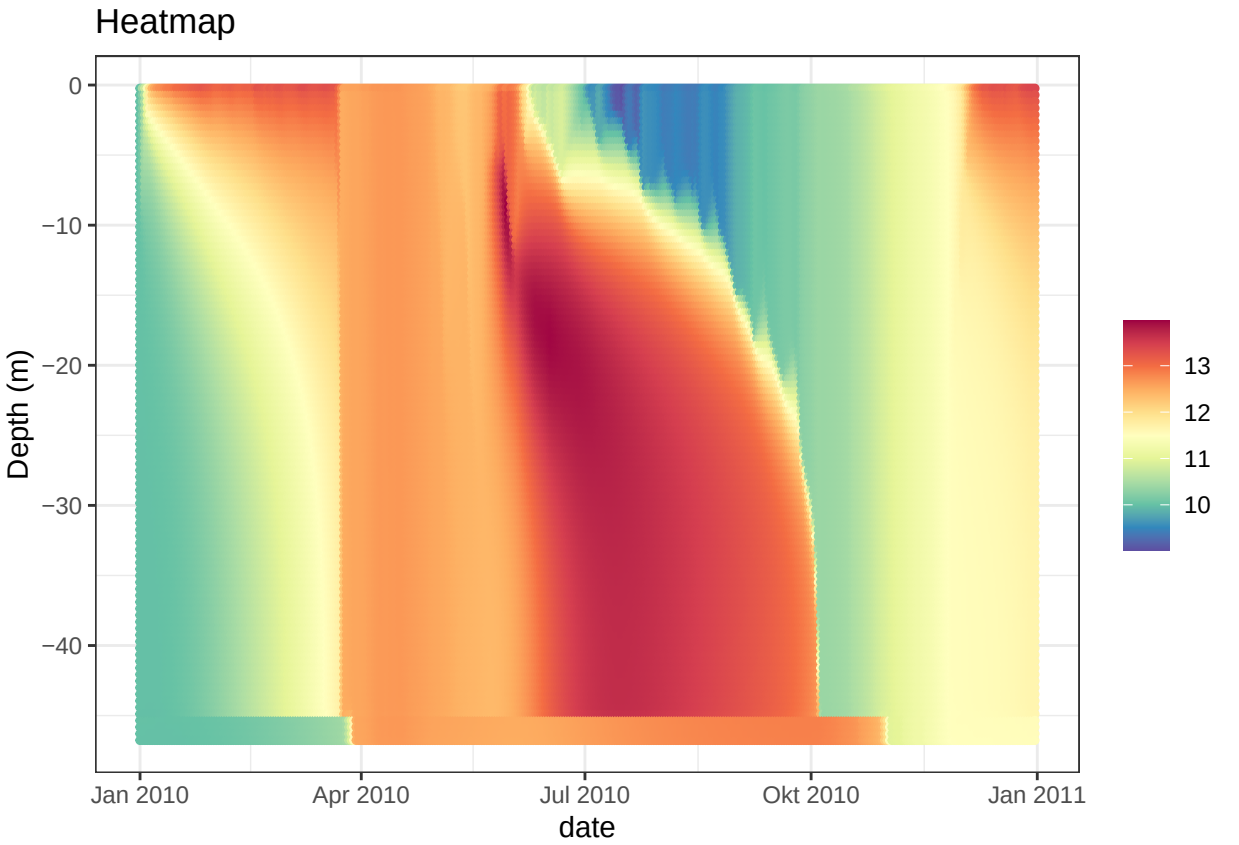
```
plot_vari("output.nc", "rodeo_C")
```




```
plot_vari("output.nc", "rodeo_HPO4")
```




```
plot_vari("output.nc", "rodeo_O2")
```

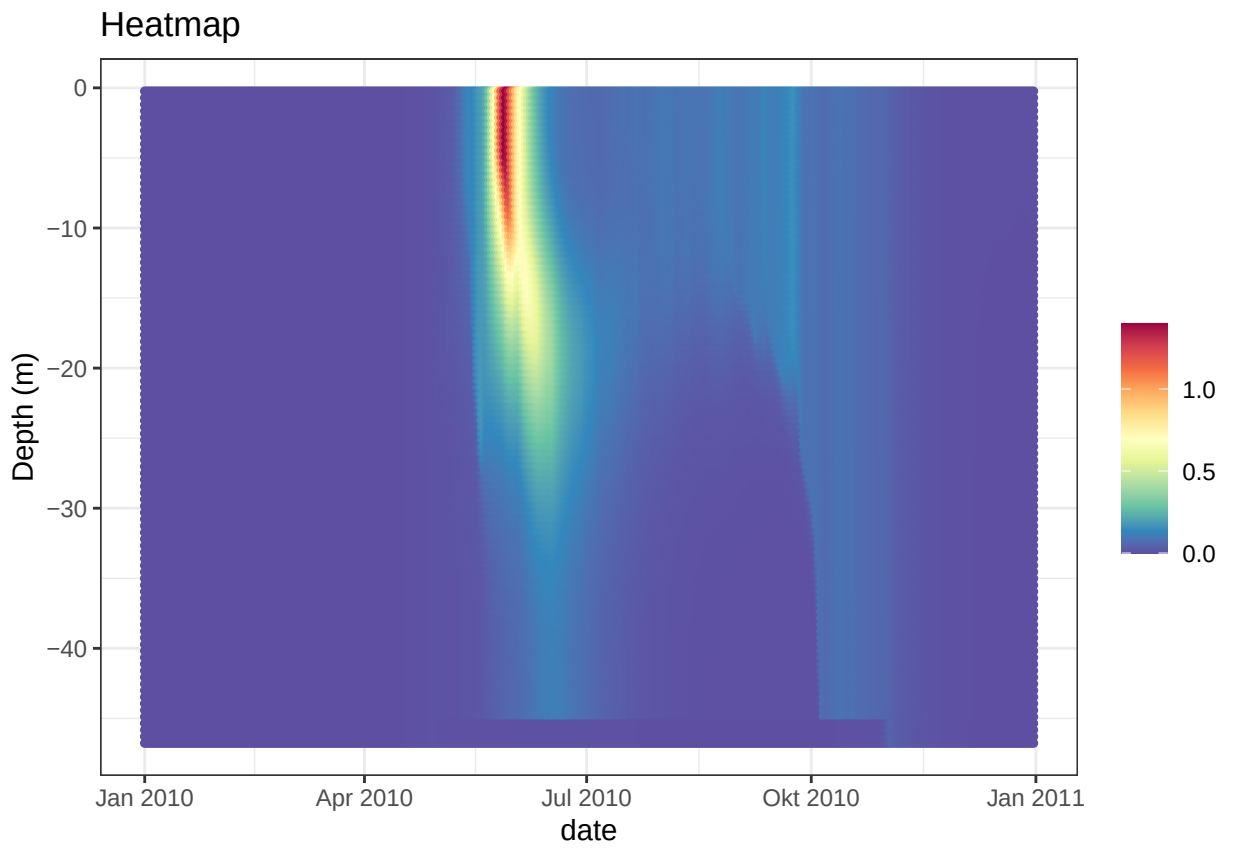


3.3.2 Sediment or Surface attached state variables

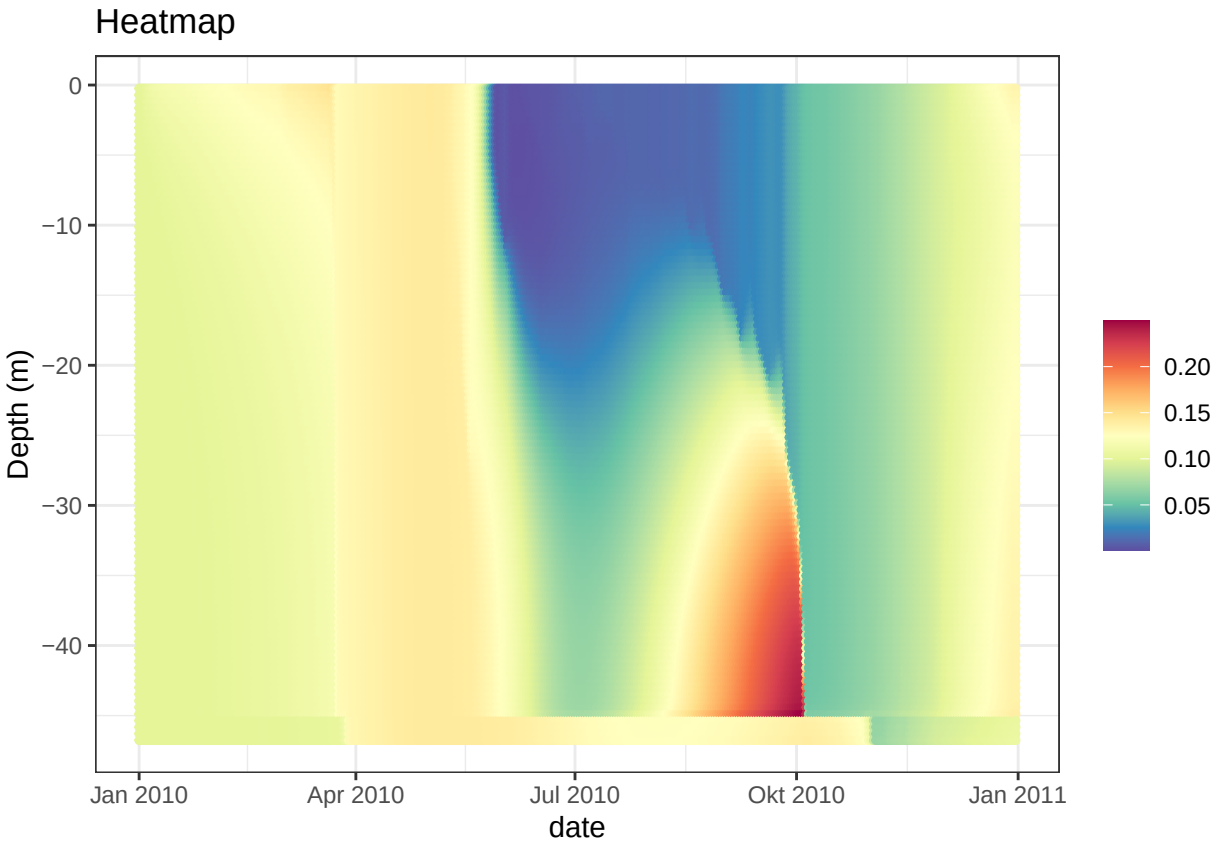
death of algae generates POM which sediments faster and can become SPOM and be mineralized at the sediment faster

```
ods <- system.file("extdata/examples/simple_alg_O2_POM.ods",
                   package = "rodeoFABM")
file.copy(from = ods, to = ".", recursive = TRUE)

# read in first simple model
vars <- read_ods("simple_alg_O2_POM.ods", sheet = "vars")
pars <- read_ods("simple_alg_O2_POM.ods", sheet = "pars")
funs <- read_ods("simple_alg_O2_POM.ods", sheet = "funs")
pros <- read_ods("simple_alg_O2_POM.ods", sheet = "pros")
stoi <- read_ods("simple_alg_O2_POM.ods", sheet = "stoi")

# create the fabm code
gen_fabm_code(vars, pars, funs, pros, stoi, "model_5.f90")

# build GOTM with the model
build_GOTM(build_dir = "build", src_dir = "gotm_src/extern/fabm/src/models/tuddhyb/rodeo/",
           fabm_file = "model_5.f90")

# run the model
system2("./gotm")
```

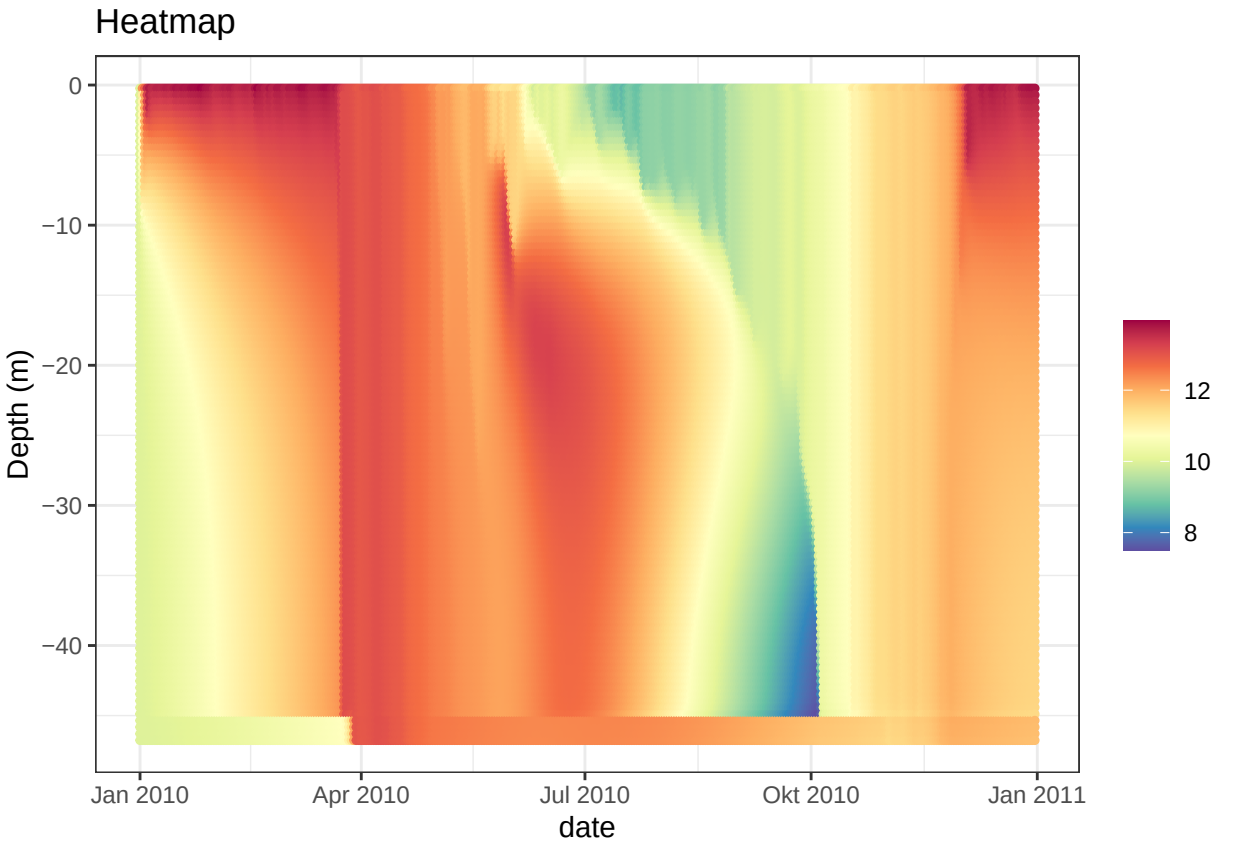
```
# plot the variables
plot_vari("output.nc", "rodeo_C")
```



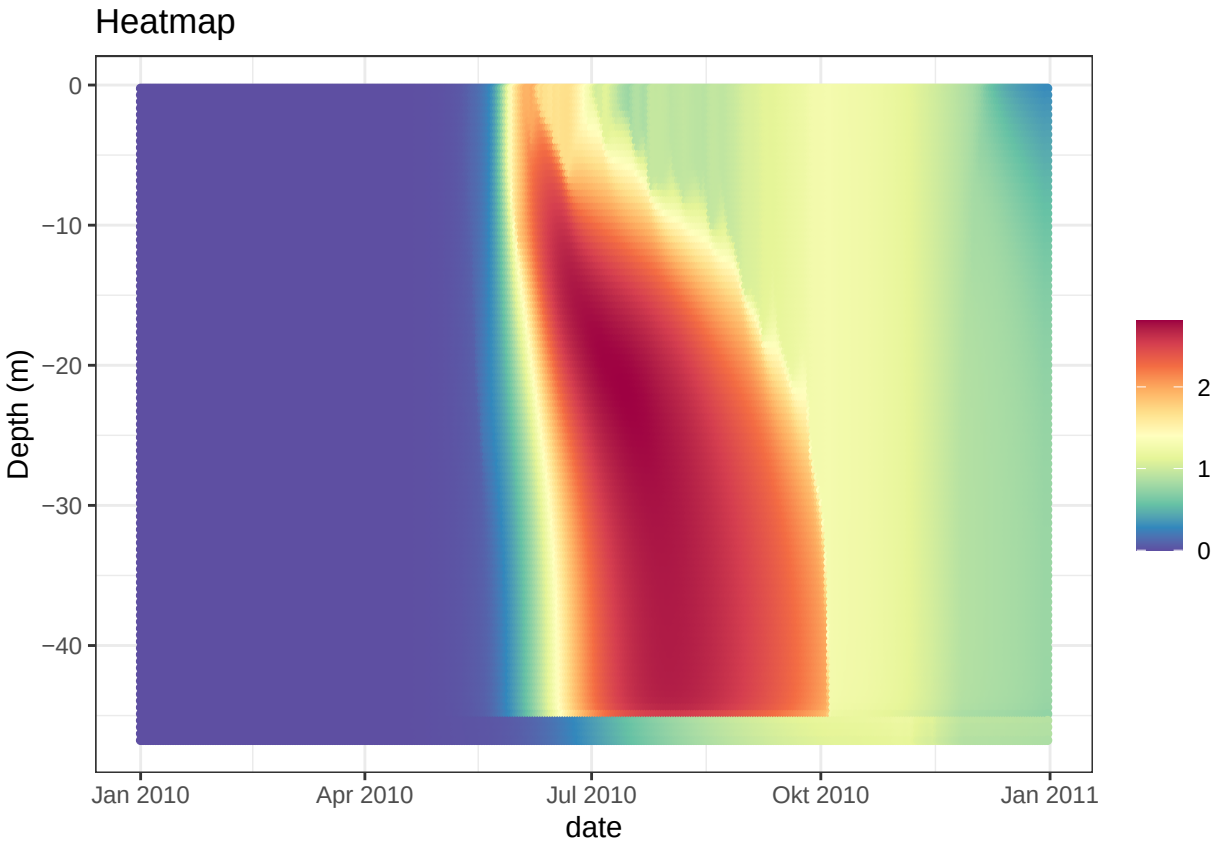

```
plot_vari("output.nc", "rodeo_HPO4")
```



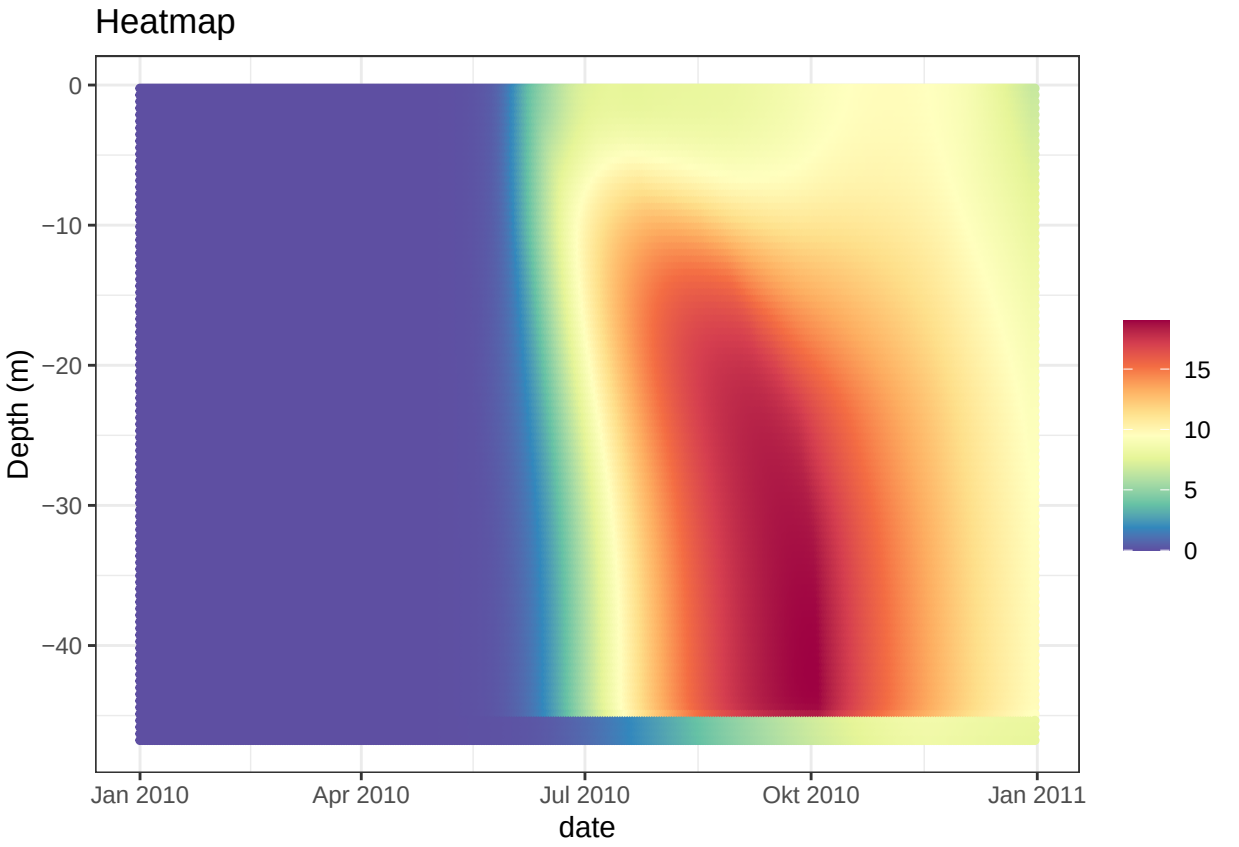

```
plot_vari("output.nc", "rodeo_O2")
```




```
plot_vari("output.nc", "rodeo_POM")
```




```
plot_vari("output.nc", "rodeo_SPOM")
```



3.4 Additional features

3.4.1 Additional state variable argumenhts

There are a few additional arguments for state variables that can be defined in FABM. In order to use them a new column in the state variable data frame needs to be added with exactly the name.

- minimum: minnimum allowed value for the state variable
- maximum: maximum allowed value for the state variable
- specific_light_extinction: specific light extinction coefficient of this variable
- no_precipitation_dilution: the variable is not diluted by precipitation
- no_river_dilution: the variable is not diluted by river inflows

3.4.2 automatic model documentation

if wanted `rodeoFABM` can automatically generate LaTeX documentation of the state variables, parameter, processes and stoichometry. still under developement!

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

References

Bruggeman, Jorn, and Karsten Bolding. 2014. "A General Framework for Aquatic Biogeochemical Models." *Environmental Modelling & Software* 61 (November): 249–65. https://doi.org/10.1016/j.envsoft.2014.04.002.

Kneis, David, Thomas Petzoldt, and Thomas U. Berendonk. 2017. "An R-Package to Boost Fitness and Life Expectancy of Environmental Models." *Environmental Modelling & Software* 96 (October): 123–27. https://doi.org/10.1016/j.envsoft.2017.06.036.