

# Package **OceanView**, visualisation of the output of a 3D hydrodynamic model.

Karline Soetaert

NIOZ-Yerseke  
The Netherlands

---

## Abstract

The R (R Development Core Team 2013) package **OceanView** (Soetaert 2021a) is a companion package to the packages **plot3D** (Soetaert 2021b) and **plot3Drgl** (Soetaert 2021c). These packages contain functions for visualising multidimensional data in base R graphics (**plot3D**) or in OpenGL (**plot3Drgl**).

**OceanView** is specifically designed for visualising complex oceanographic data.

It can produce a.o. quiver plots, vector plots for visualising flows, it can create movie sequences for depicting particle tracks in 2-D and 3-D, and so on ...

Here we apply **OceanView** for plotting the output of a 3-D hydrodynamic model of the North Sea, as produced by the GETM software (Burchard and Bolding 2002).

*Keywords:* marine science, hydrodynamic models, 3-D data, 4-D data, quiver, image2D, R .

---

## 1. Preamble

This 'vignette' contains scripts to make figures of the output of the Northsea Model as created by the 3-D hydrodynamical model GETM (Burchard and Bolding 2002).

As the netcdf file with model output is very large (> 150 MB), it is not part of the package nor can it be downloaded. Hence you will not be able to recreate this vignette.

However, a similar (but significantly smaller) dataset, **Sylt3D** is part of the **OceanView** package:

```
example(Sylt3D)
```

## 2. Reading data

We use two packages here, **RNetCDF** (Michna and Woods 2020) for reading netcdf files, **OceanView** (Soetaert 2021a) for visualising output. Load the required packages:

```
library(RNetCDF)
library(OceanView)
```

Open the NETCDF file with the output of the Northsea simulation, and print its content: <sup>1</sup>

```
d.nc <- open.nc( "ns_06nm_sept_1997_3d.nc")
print.nc(d.nc)

dimensions:
  xc = 111 ;
  yc = 87 ;
  level = 26 ;
  time = UNLIMITED ; // (30 currently)
variables:
  int grid_type ;
  int vert_cord ;
  int ioff ;
      ioff:long_name = "index offset (i)" ;
  int joff ;
      joff:long_name = "index offset (j)" ;
  double dx ;
      dx:units = "m" ;
      dx:long_name = "grid spacing (x)" ;
  double dy ;
      dy:units = "m" ;
      dy:long_name = "grid spacing (y)" ;
  double xc(xc) ;
      xc:units = "m" ;
  double yc(yc) ;
      yc:units = "m" ;
  double lonc(xc, yc) ;
      lonc:units = "degrees_east" ;
      lonc:long_name = "longitude" ;
      lonc:valid_range = -180180 ;
      lonc:_FillValue = -999 ;
      lonc:missing_value = -999 ;
  double latc(xc, yc) ;
      latc:units = "degrees_north" ;
      latc:long_name = "latitude" ;
      latc:valid_range = -9090 ;
      latc:_FillValue = -999 ;
      latc:missing_value = -999 ;
  double convc(xc, yc) ;
      convc:units = "degrees" ;
      convc:long_name = "grid rotation" ;
      convc:valid_range = -180180 ;
      convc:_FillValue = -999 ;
      convc:missing_value = -999 ;
```

---

<sup>1</sup>Note: the text in boxes is the R-code; the text below boxes in this format (if any) is output, produced by R.

```
double latu(xc, yc) ;
    latu:units = "degrees" ;
    latu:long_name = "latu" ;
    latu:valid_range = -9090 ;
    latu:_FillValue = -999 ;
    latu:missing_value = -999 ;
double latv(xc, yc) ;
    latv:units = "degrees" ;
    latv:long_name = "latv" ;
    latv:valid_range = -9090 ;
    latv:_FillValue = -999 ;
    latv:missing_value = -999 ;
double level(level) ;
    level:units = "level" ;
double bathymetry(xc, yc) ;
    bathymetry:units = "m" ;
    bathymetry:long_name = "bathymetry" ;
    bathymetry:valid_range = -54000 ;
    bathymetry:_FillValue = -10 ;
    bathymetry:missing_value = -10 ;
float time(time) ;
    time:units = "seconds since 1997-09-01 00:00:00" ;
    time:long_name = "time" ;
float elev(xc, yc, time) ;
    elev:units = "m" ;
    elev:long_name = "elevation" ;
    elev:valid_range = -1515 ;
    elev:_FillValue = -9999 ;
    elev:missing_value = -9999 ;
float u(xc, yc, time) ;
    u:units = "m/s" ;
    u:long_name = "int. zonal vel." ;
    u:valid_range = -33 ;
    u:_FillValue = -9999 ;
    u:missing_value = -9999 ;
float v(xc, yc, time) ;
    v:units = "m/s" ;
    v:long_name = "int. meridional vel." ;
    v:valid_range = -33 ;
    v:_FillValue = -9999 ;
    v:missing_value = -9999 ;
float h(xc, yc, level, time) ;
    h:units = "m" ;
    h:long_name = "layer thickness" ;
    h:_FillValue = -9999 ;
    h:missing_value = -9999 ;
float hcc(xc, yc, level) ;
```

```

        hcc:units = "" ;
        hcc:long_name = "hcc" ;
        hcc:valid_range = 01 ;
        hcc:_FillValue = -1 ;
        hcc:missing_value = -1 ;
float uu(xc, yc, level, time) ;
        uu:units = "m/s" ;
        uu:long_name = "zonal vel." ;
        uu:valid_range = -33 ;
        uu:_FillValue = -9999 ;
        uu:missing_value = -9999 ;
float vv(xc, yc, level, time) ;
        vv:units = "m/s" ;
        vv:long_name = "meridional vel." ;
        vv:valid_range = -33 ;
        vv:_FillValue = -9999 ;
        vv:missing_value = -9999 ;
float w(xc, yc, level, time) ;
        w:units = "m/s" ;
        w:long_name = "vertical vel." ;
        w:valid_range = -33 ;
        w:_FillValue = -9999 ;
        w:missing_value = -9999 ;
float salt(xc, yc, level, time) ;
        salt:units = "PSU" ;
        salt:long_name = "salinity" ;
        salt:valid_range = 040 ;
        salt:_FillValue = -9999 ;
        salt:missing_value = -9999 ;
float temp(xc, yc, level, time) ;
        temp:units = "degC" ;
        temp:long_name = "temperature" ;
        temp:valid_range = 040 ;
        temp:_FillValue = -9999 ;
        temp:missing_value = -9999 ;

// global attributes:
        :title = "North Sea - 6nm" ;
        :history = "Generated by getm, ver. 1.8.0" ;

```

## 2.1. Reading the data

The data are read:

```

lat_c <- var.get.nc(d.nc, "latc")
lon_c <- var.get.nc(d.nc, "lonc")
time <- var.get.nc(d.nc, "time")

```

```

level <- var.get.nc(d.nc, "level")
convc <- var.get.nc(d.nc, "convc")
xc    <- var.get.nc(d.nc, "xc") / 1000 # in km
yc    <- var.get.nc(d.nc, "yc") / 1000
dx    <- var.get.nc(d.nc, "dx") / 1000
dy    <- var.get.nc(d.nc, "dy") / 1000
hcc   <- var.get.nc(d.nc, "hcc")
h     <- var.get.nc(d.nc, "h")
bathy <- var.get.nc(d.nc, "bathymetry")
# get 2-D data
u     <- var.get.nc(d.nc, "u")
v     <- var.get.nc(d.nc, "v")
elev  <- var.get.nc(d.nc, "elev")
temp  <- var.get.nc(d.nc, "temp")
ww    <- var.get.nc(d.nc, "w")
uu    <- var.get.nc(d.nc, "uu")
vv    <- var.get.nc(d.nc, "vv")

```

Show some values: the grid spacing, and the ranges of the velocities: :

```
c(dx, dy)
```

```
[1] 11.112 11.112
```

```
range(uu, na.rm = TRUE)
```

```
[1] -1.217723 1.751082
```

```
range(vv, na.rm = TRUE)
```

```
[1] -1.303300 2.244228
```

```
range(ww, na.rm = TRUE)
```

```
[1] -0.01644286 0.02704342
```

## 2.2. Box depths

The output of the model lacks an output variable that has the depth, in the middle of each compartment. It is created from the thickness of each box ( $h$ ). First we take the temporal mean of the box thicknesses:

```
hh <- apply(h, MARGIN = 1:3, FUN = mean)
```

Then we take the cumulative sum of the box thicknesses. This gives the depth at the box interfaces:

6      *Package **OceanView**, visualisation of the output of a 3D hydrodynamic model.*

```
depth <- hh
depth[ , ,1] <- 0
for (i in 26:2)
  depth[ , , i-1] <- depth[ , , i-1] + depth[ , , i]
```

Then we take the mean of the interfaces, to get the depth at the centre of a box.

```
for (i in 2:26)
  depth[ , , i] <- 0.5*(depth[ , , i-1] + depth[ , , i])
```

### 3. Plot bathymetry and grid

jet2.col from the package plot3D is a suitable color scheme, not too dark.

```

par(mfrow = c(2, 2))
col <- jet2.col(100)
#
# Plot every fourth grid point
plot(x = lon_c[seq(1, length(lon_c), by = 4)],
     y = lat_c[seq(1, length(lat_c), by = 4)],
     xlab = "longitude", ylab = "latitude",
     pch = ".", main = "grid points")
legend("bottom", legend = paste (c("x: ", "y: "), dim(lon_c)))
#
# Plot the bathymetry on the grid (distorted)
image2D(bathy, x = xc, y = yc, NAcol = "black",
        col = col, xlab = "km", ylab = "km",
        main = "Bathymetry on grid", clab = c("", "", "m"))
#
# The bathymetry on regular coordinates - shade emphasises bathymetry
image2D(bathy, x = lon_c, y = lat_c, NAcol = "black", shade = 0.15,
        col = col, main = "Bathymetry on coordinates",
        xlab = "longitude", ylab = "latitude", clab = c("", "", "m"))
#
# add transect line
lines(lon_c[30,], lat_c[30,]) # from where the sigma grid is plotted
#
# plot the sigma grid on transect line
matplot(x = yc, depth[30,,], type = "l", col = "black",
        main = "sigma grid", ylab = "depth, m", xlab = "y",
        lty = 1, ylim = c(140, 0))
lines(yc, bathy[30, ], lwd = 2, col = "red")

```

#### 3.1. Another bathymetric view

Bathymetric data can also be plotted as perspective plots.

```

par(mfrow = c(1, 1))
# make bathymetry that does not have NAs (for z-values)
D <- bathy ; D[is.na(D)] <- 0
persp3D(x = xc, y = yc, z = -D, colvar = -bathy,
        col = "grey", shade = 0.5, scale = FALSE, phi = 80,
        NAcol = grey(0.2), theta = 0, box = FALSE)

```

To zoom, rotate, cut, try:

```

plotrgl()
plotdev(xlim = c(-100, 500), ylim = c(5667, 6300), zlim = c(-50,0), phi = 50)

```

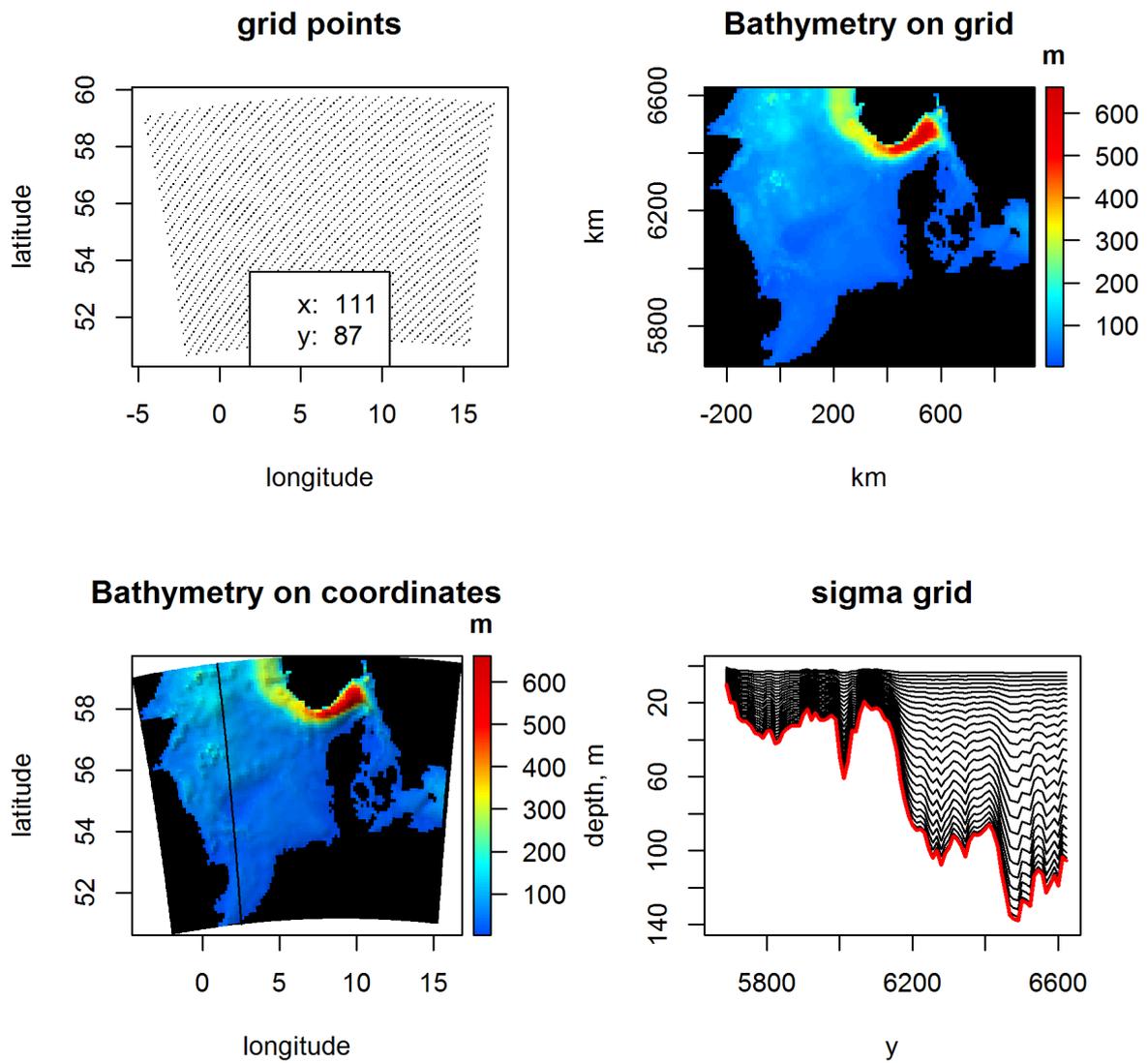


Figure 1: Bathymetry and grid

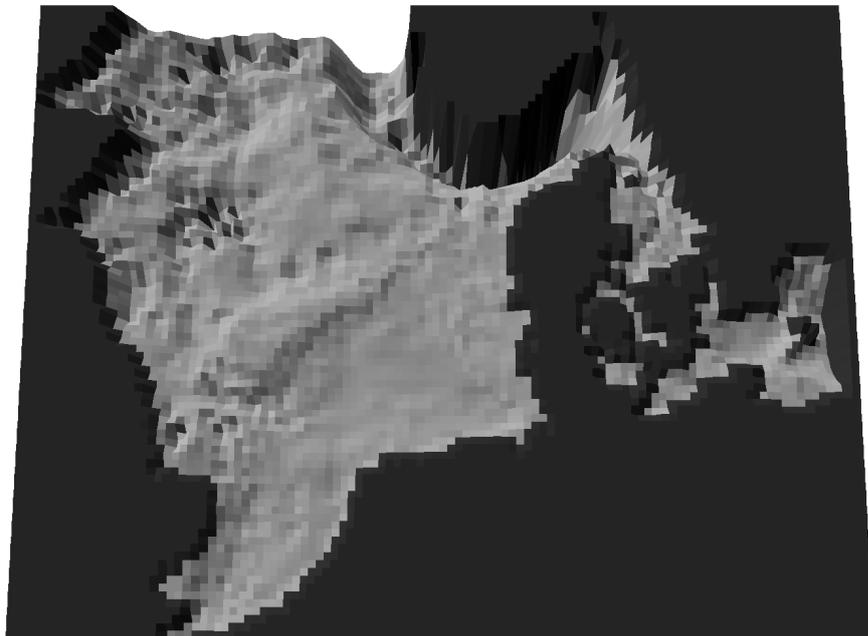


Figure 2: Bathymetry in perspective

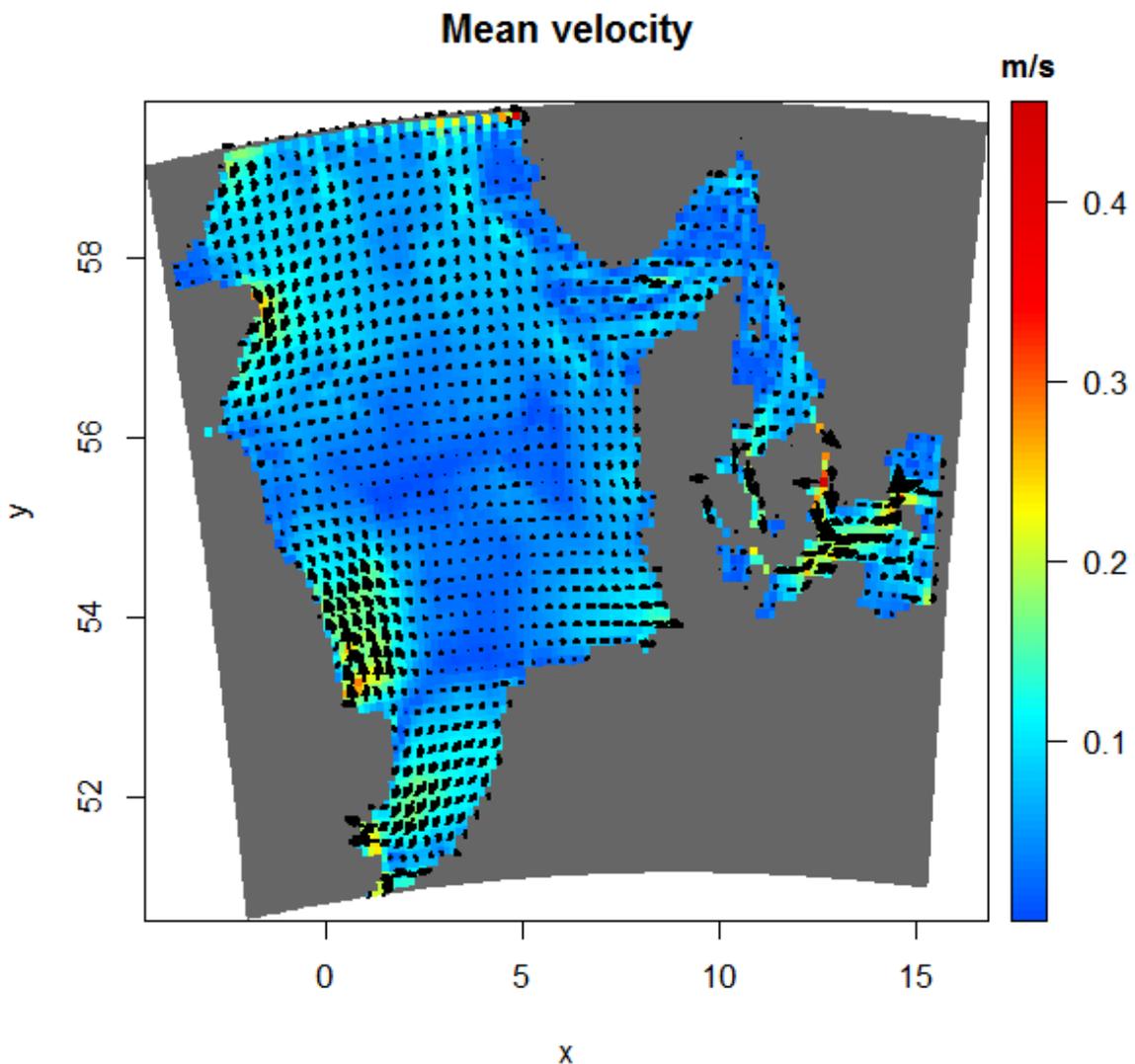
## 4. integrated velocities

The arrays `u`, `v` contain depth-average velocities, as they vary in time. We calculate the average velocity over the 30 days of the simulation:

```
U <- apply (u, FUN = mean, MARGIN = 1:2)
V <- apply (v, FUN = mean, MARGIN = 1:2)
meanV <- sqrt(U^2 + V^2)
```

We plot the average velocities as quivers on an image:

```
par(mfrow = c(1, 1))
image2D(z = meanV, x = lon_c, y = lat_c,
        col = col, NAcol = grey(0.4),
        main = "Mean velocity", clab = c("", "", "m/s"))
quiver2D(U, V, x = lon_c, y = lat_c, add = TRUE,
         by = 2, scale = 3, lwd = 2, arr.max = 0.3)
```



Try this, and use left and right mousekey:

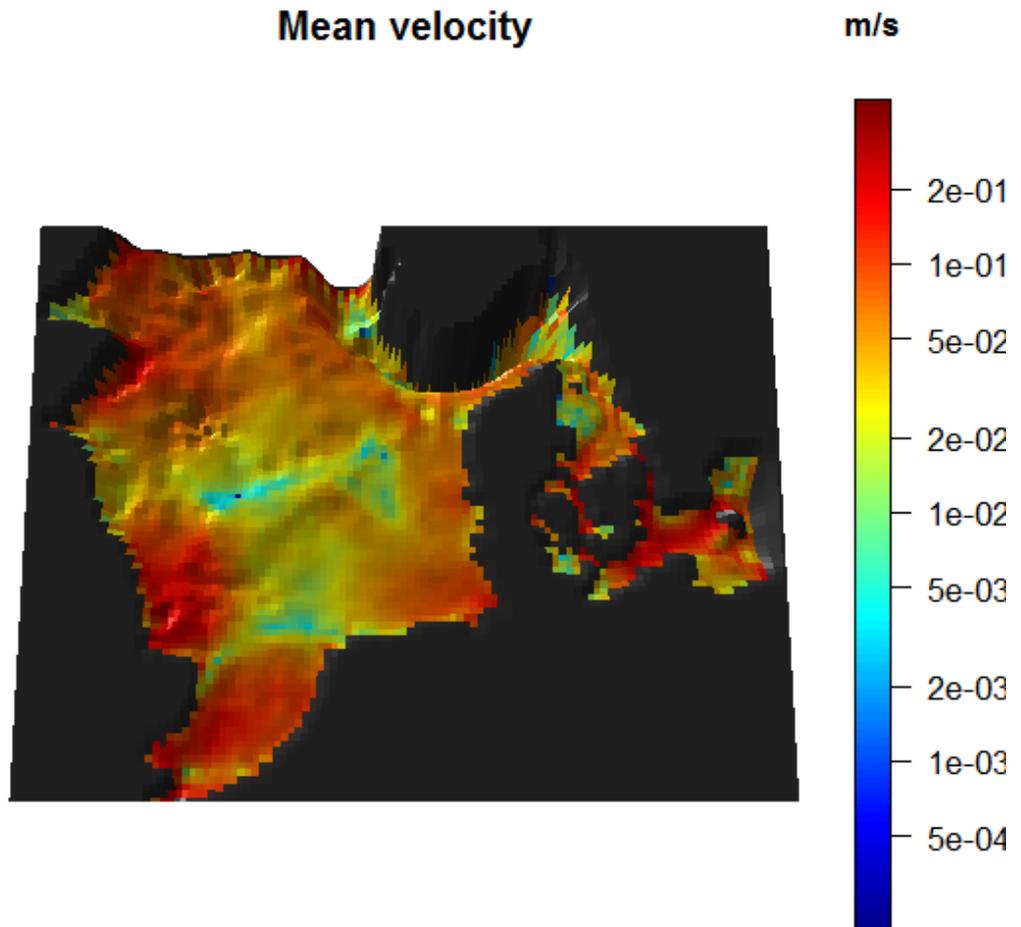
```
image2Drgl(z = meanV, x = lon_c, y = lat_c,
  col = col, NAcol = grey(0.4),
  main = "Mean velocity")
quiver2Drgl(U, V, x = lon_c, y = lat_c, add = TRUE,
  by = 2, scale = 3, lwd = 2, arr.max = 0.2)
# select a rectangular region
cutrgl()
```

Velocity (log scale) draped on the bathymetry.

```
persp3D(x = xc, y = yc, z = -D, colvar = meanV,
  lighting = TRUE, scale = FALSE, phi = 80, box = FALSE,
```

12      Package **OceanView**, visualisation of the output of a 3D hydrodynamic model.

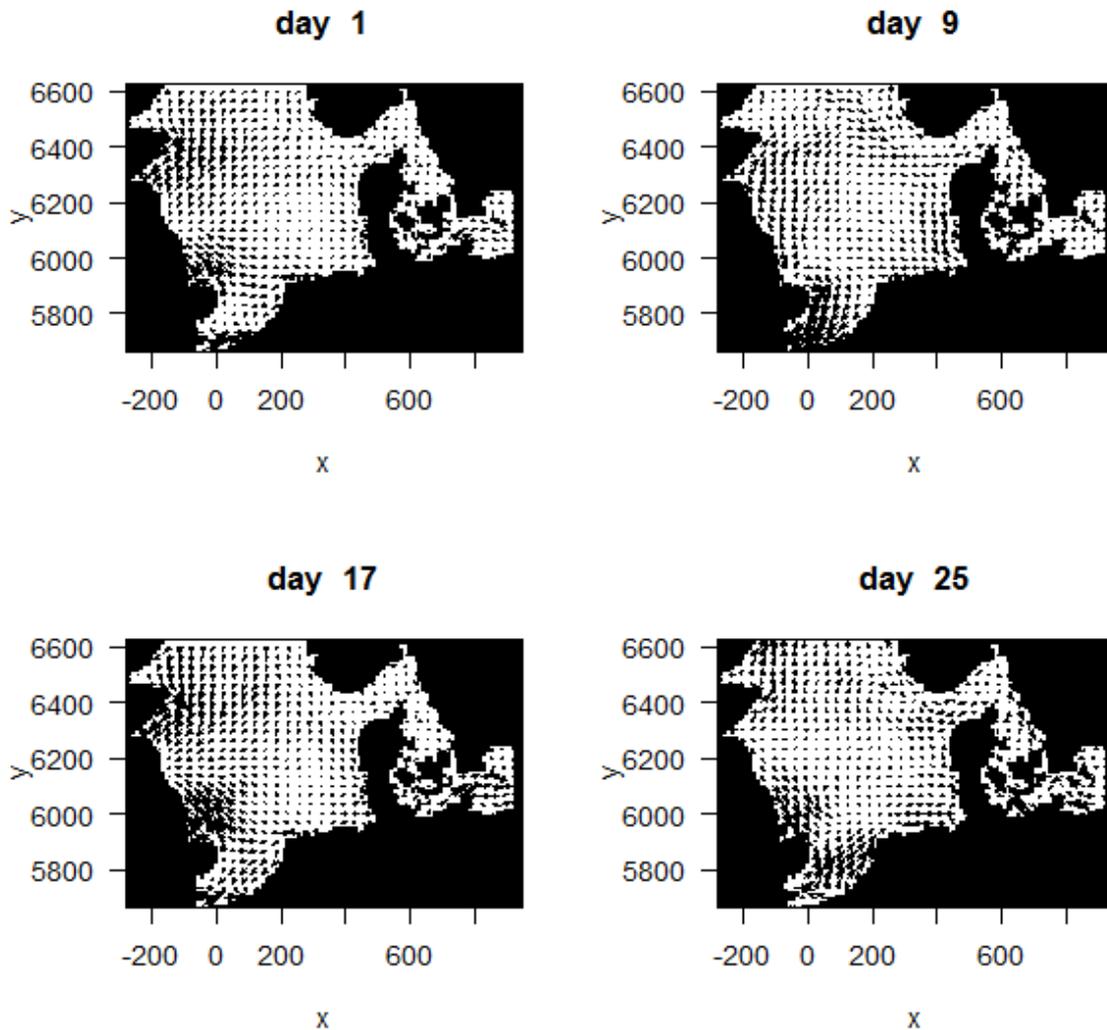
```
log = "c", clab= "m/s", main = "Mean velocity",  
NAcol = grey(0.2), theta = 0, inttype = 2)
```



## 5. Quivers at selected time points

We can also easily visualise quivers at selected time points:

```
tselect <- time[seq(from = 1, to = 30, by = 8)]
par(oma = c(0, 0, 2, 0), las = 1)
quiver2D(u, v, x = xc, y = yc,
  subset = (time %in% tselect),
  scale = 3, arr.max = 0.2, by = 3, mask = bathy,
  NAcol = "darkgrey", main = paste("day ", tselect/86400))
```

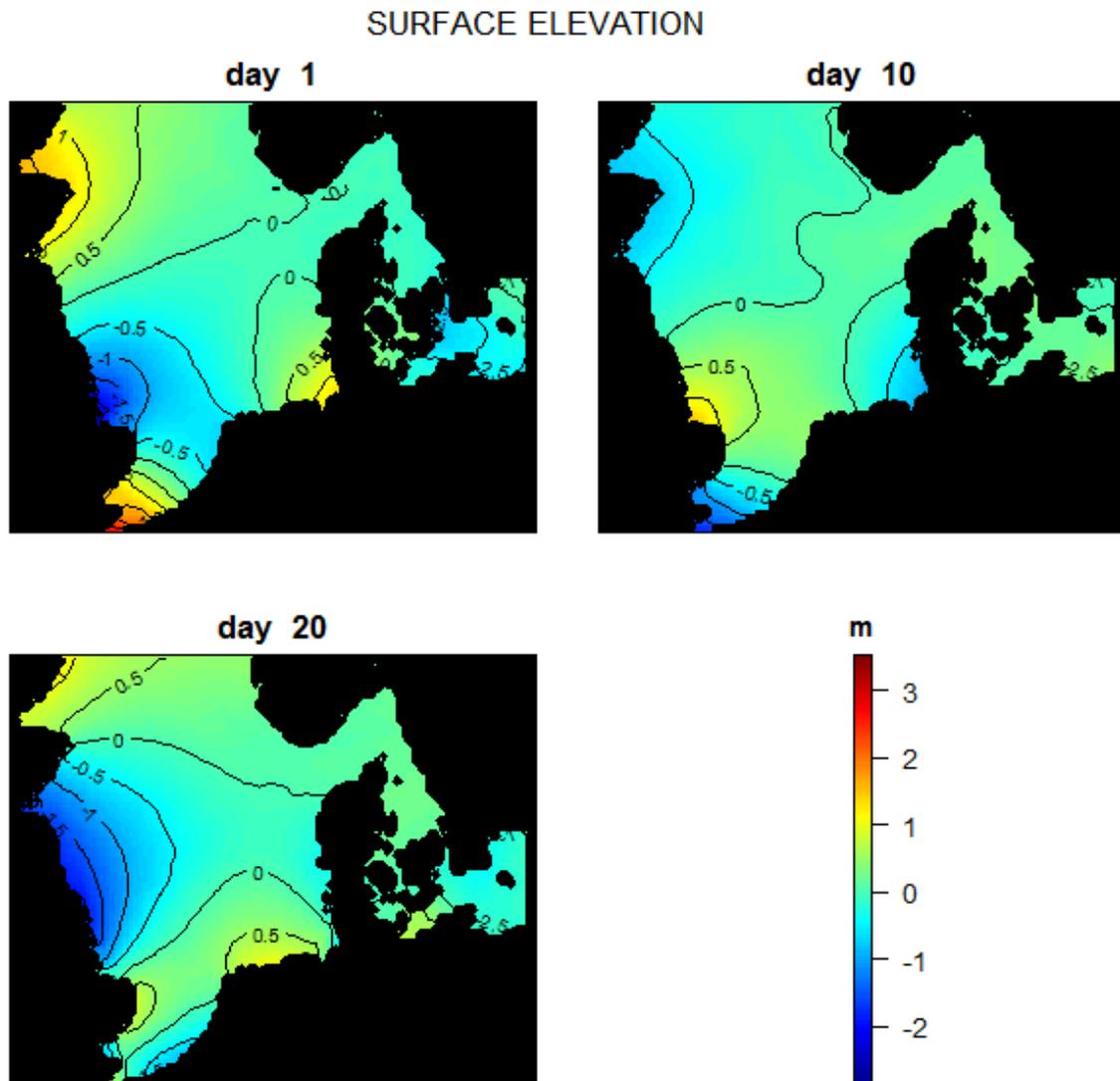


## 6. surface elevation, for selected time steps

To depict the surface elevation at 3 time points, we use `image2D`.

```
zlim <- range(elev, na.rm = TRUE)
tselect <- time[c(1, 10, 20)]
```

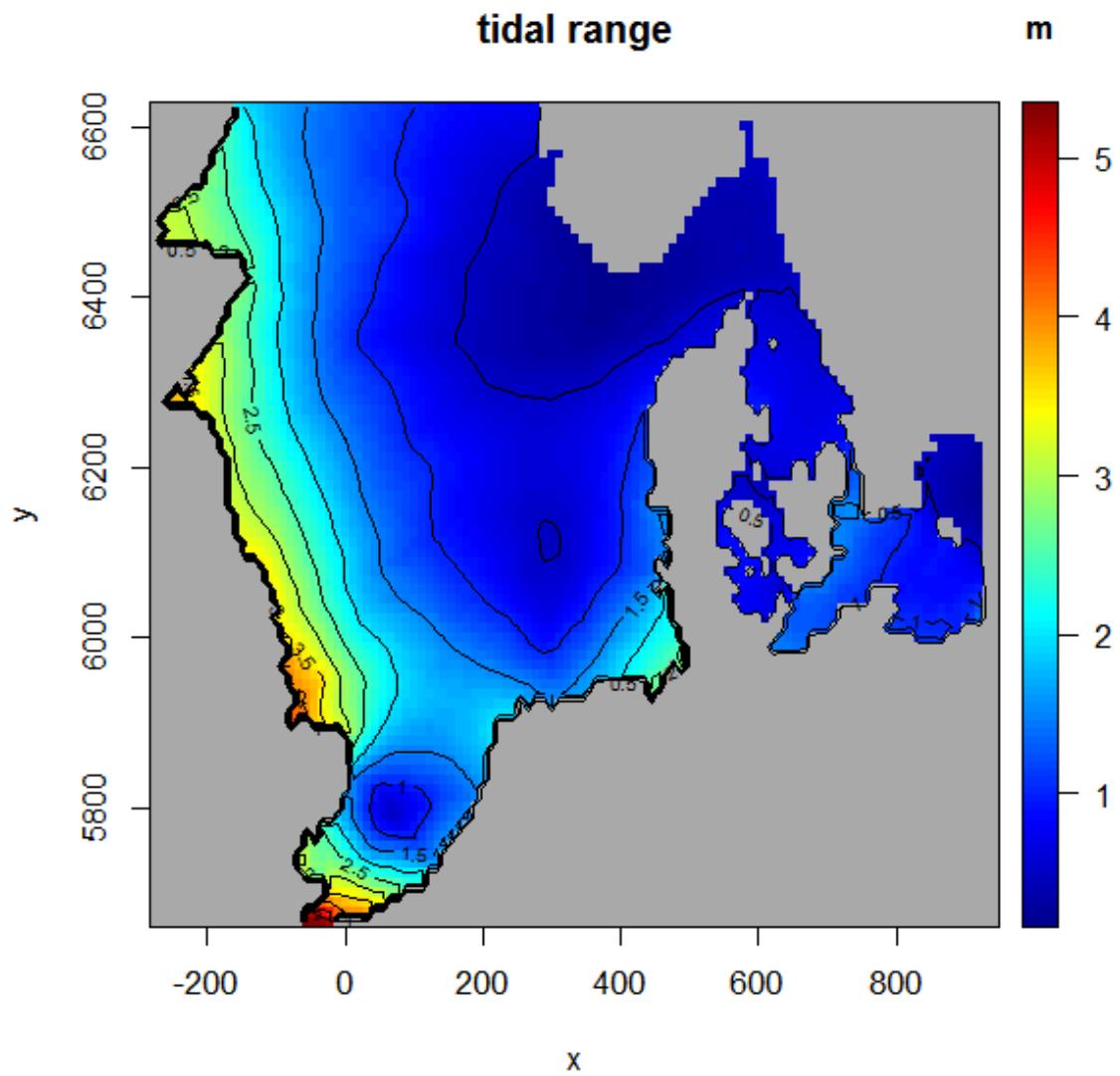
```
par(oma = c(0, 0, 2, 0))
pm <- par(mar = c(2, 2, 2, 0))
image2D(elev, subset = (time %in% tselect), mfrow = c(2, 2),
        NAcol = "black", axes = FALSE, xlab = "", ylab = "",
        contour = TRUE, main = paste("day ", tselect/3600/24),
        zlim = zlim, colkey = FALSE)
colkey(clim = zlim, clab = "m")
mtext(outer = TRUE, side = 3, "SURFACE ELEVATION", line = 0)
par(mar = pm)
```



The range of surface elevation, averaged over time is plotted, and contours are added:

```
tidal.range <- apply(elev, MARGIN = 1:2,
                    FUN = function(x) diff(range(x)))

tselect <- time[seq(from = 1, to = 30, length.out = 9)]
par(mfrow = c(1, 1))
image2D(x = xc, y = yc, z = tidal.range, contour = TRUE,
        NAcol = "darkgrey", clab = "m", main = "tidal range")
```

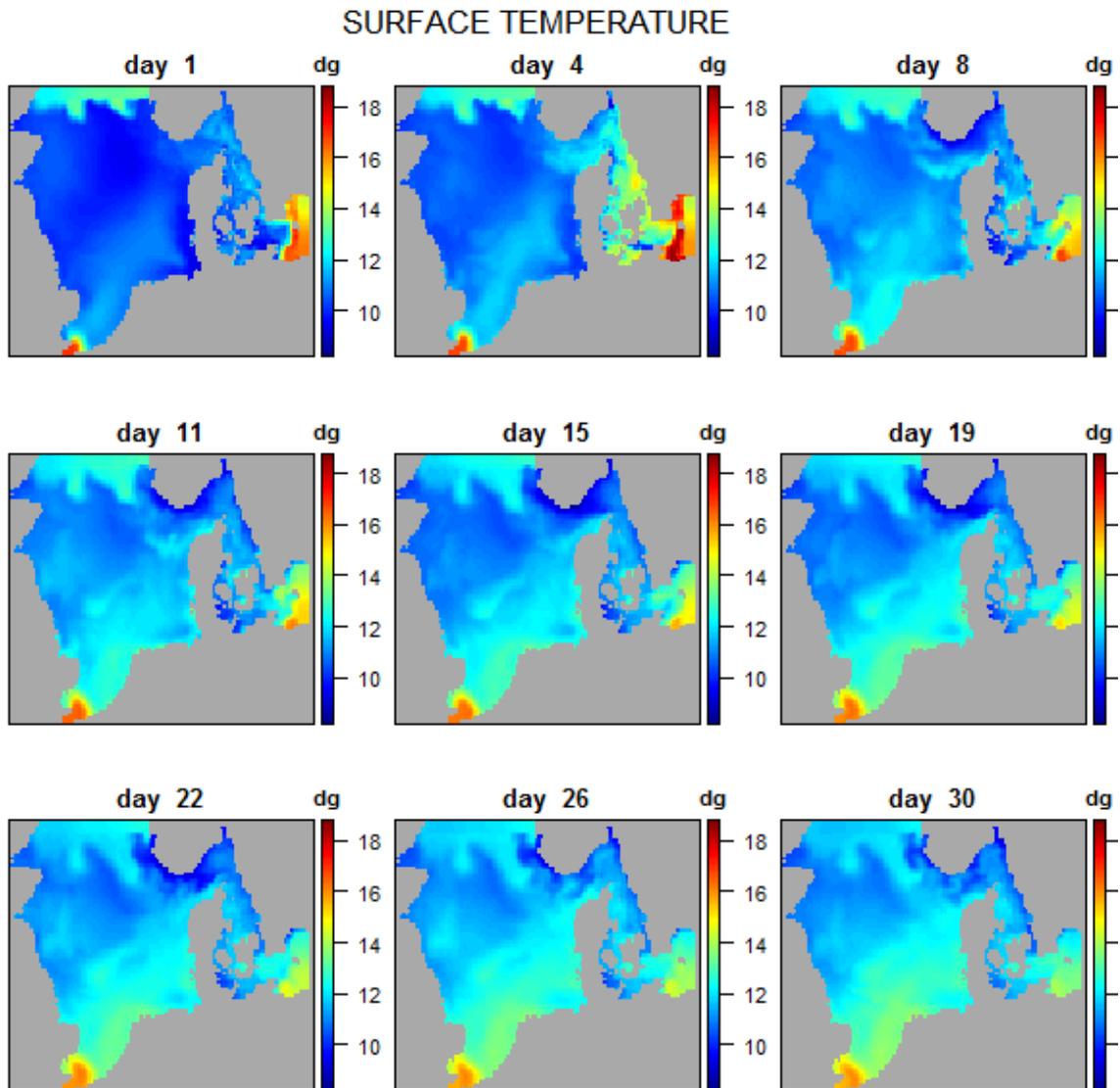


## 7. surface temperature, for selected time steps

`temp` contains 3D temperature data at all time steps. The surface data are selected first, and then plotted for a selection of time steps.

```
surfTemp <- temp[ , , 26, ]  
zlim <- range(surfTemp, na.rm = TRUE)
```

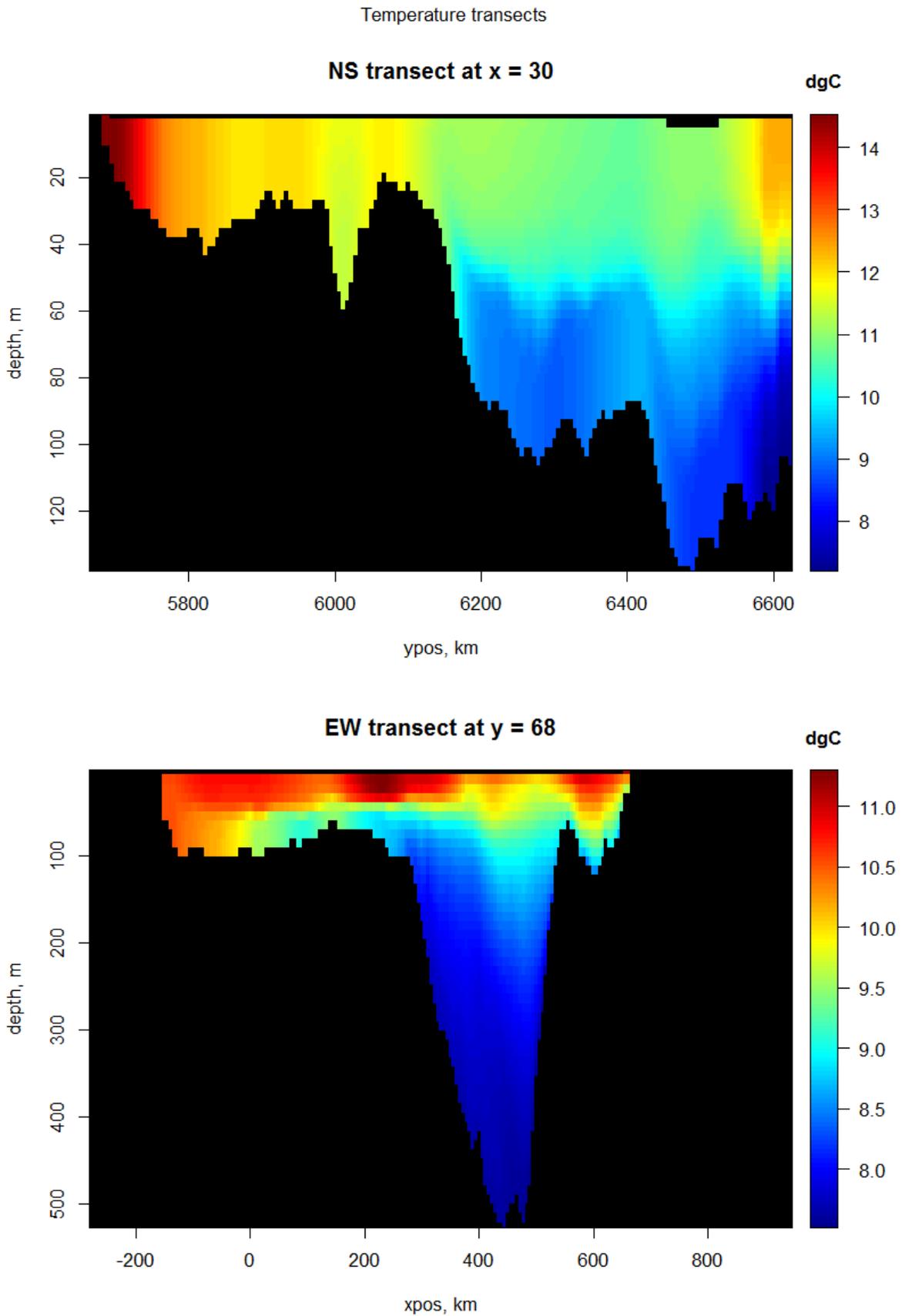
```
par(oma = c(0, 0, 2, 0))  
pm <- par(mar = c(2, 2, 2, 0))  
image2D(surfTemp, subset = (time %in% tselect), mfrow = c(3, 3),  
        NAcol = "darkgrey", axes = FALSE, xlab = "", ylab = "",  
        main = paste("day ", tselect/3600/24), zlim = zlim, clab = "dg")  
mtext (outer = TRUE, side = 3, "SURFACE TEMPERATURE")  
par(mar = pm)
```



## 8. mean temperature along a NS-depth and a SW-depth transect

For two transects, first the time-mean temperature is estimated (`apply`). The temperature matrix is then converted from sigma coordinates to depth values (`mapsigma`), increasing the resolution (`resfac`). Then the `image` is generated.

```
par(oma = c(0, 0, 2, 0))
par(mfrow = c(2, 1))
TranTemp <- apply(temp[30 , , ], MARGIN = 1:2, FUN = mean)
MS <- mapsigma(TranTemp, sigma = depth[30, , ], x = yc, resfac = 2)
image2D(MS$var, y = MS$depth, x = MS$x, ylim = rev(range(MS$depth)),
        NAcol = "black", ylab = "depth, m", xlab = "ypos, km",
        main = "NS transect at x = 30", clab = c("", "dgC"))
TranTemp <- apply(temp[ , 68, , ], MARGIN = 1:2, FUN = mean)
MS <- mapsigma(TranTemp, sigma = depth[ , 68, ], x = xc, resfac = 2)
image2D(MS$var, y = MS$depth, x = MS$x, ylim = rev(range(MS$depth)),
        NAcol = "black", ylab = "depth, m", xlab = "xpos, km",
        main = "EW transect at y = 68", clab = c("", "dgC"))
mtext (outer = TRUE, side = 3, "Temperature transects")
```

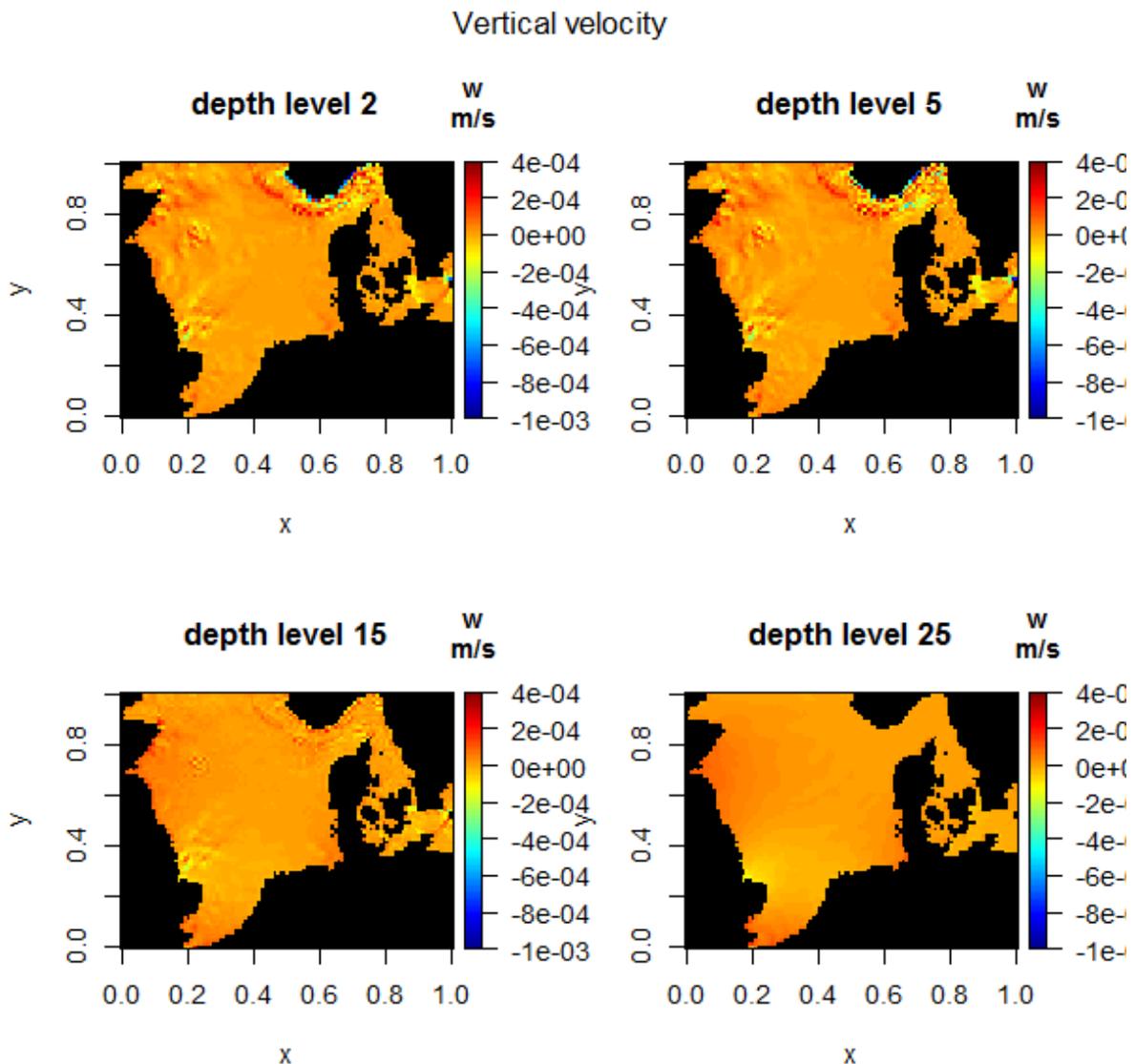


## 9. time-averaged vertical velocities at several depths

Here we create the temporally averaged vertical velocities, at selected depth levels and remove the N and E points, as they are unstable:

```
wmean <- apply(wv, MARGIN = 1:3, FUN = mean)
subset.level <- c(2, 5, 15, 25)
x.remove <- 105:111 ; y.remove <- 82:87

par(oma = c(0, 0, 2, 0))
image2D(wmean[-x.remove, -y.remove, ], NAcol = "black",
        main = paste("depth level", subset.level), zlim = c(-1e-3, 4e-4),
        subset = (level %in% subset.level), clab = c("w", "m/s"))
mtext (outer = TRUE, side = 3, "Vertical velocity")
```



## 10. time-averaged horizontal velocity vectors at several depths

`uu` and `vv` contain 3D output of horizontal velocities at all time points. We first create temporally averaged velocities:

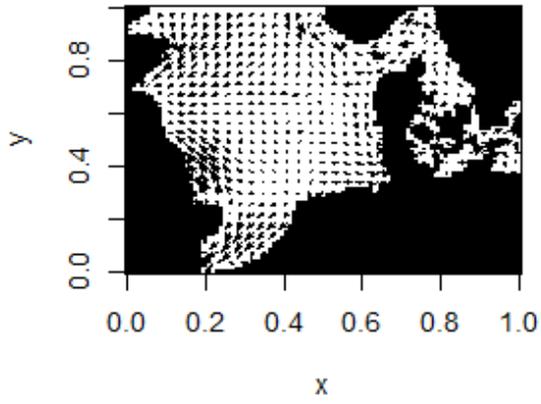
```
umean <- apply(uu, MARGIN = 1:3, FUN = mean)
vmean <- apply(vv, MARGIN = 1:3, FUN = mean)
```

The unstable N and E points are removed and a quiver produced:

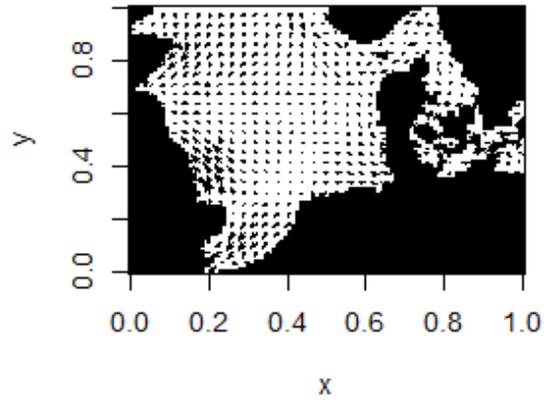
```
par(oma = c(0, 0, 2, 0))
quiver2D(umean[-x.remove, -y.remove, ], vmean[-x.remove, -y.remove, ],
         NAcol = "grey", mask = bathy[-x.remove, -y.remove],
         main = paste("depth level", subset.level), by = 3,
         scale = 3, arr.max = 0.2, subset = (level %in% subset.level))
mtext (outer = TRUE, side = 3, "Time-average flows")
```

Time-average flows

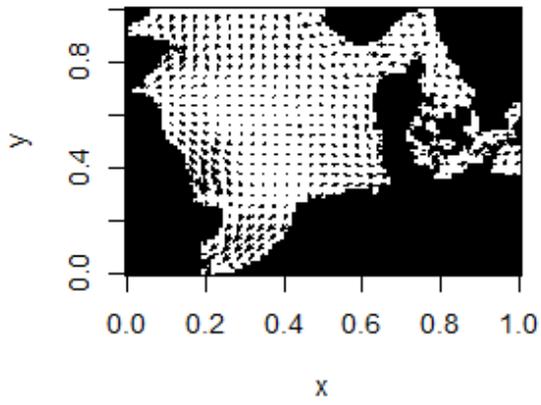
**depth level 2**



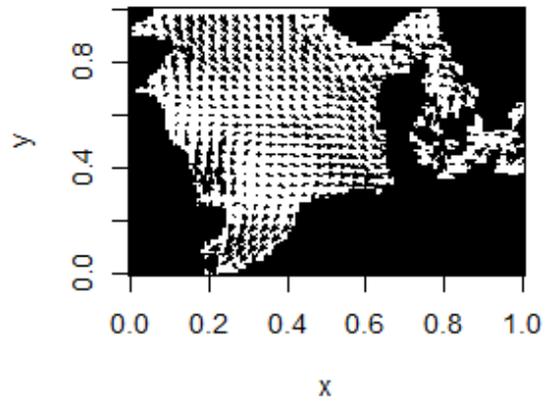
**depth level 5**



**depth level 15**



**depth level 25**



## 11. NE and SW transects of time-averaged flows

The vertical velocities are 1000 times smaller than the horizontal ones; so to produce the quiver plots, they are multiplied with 1000:

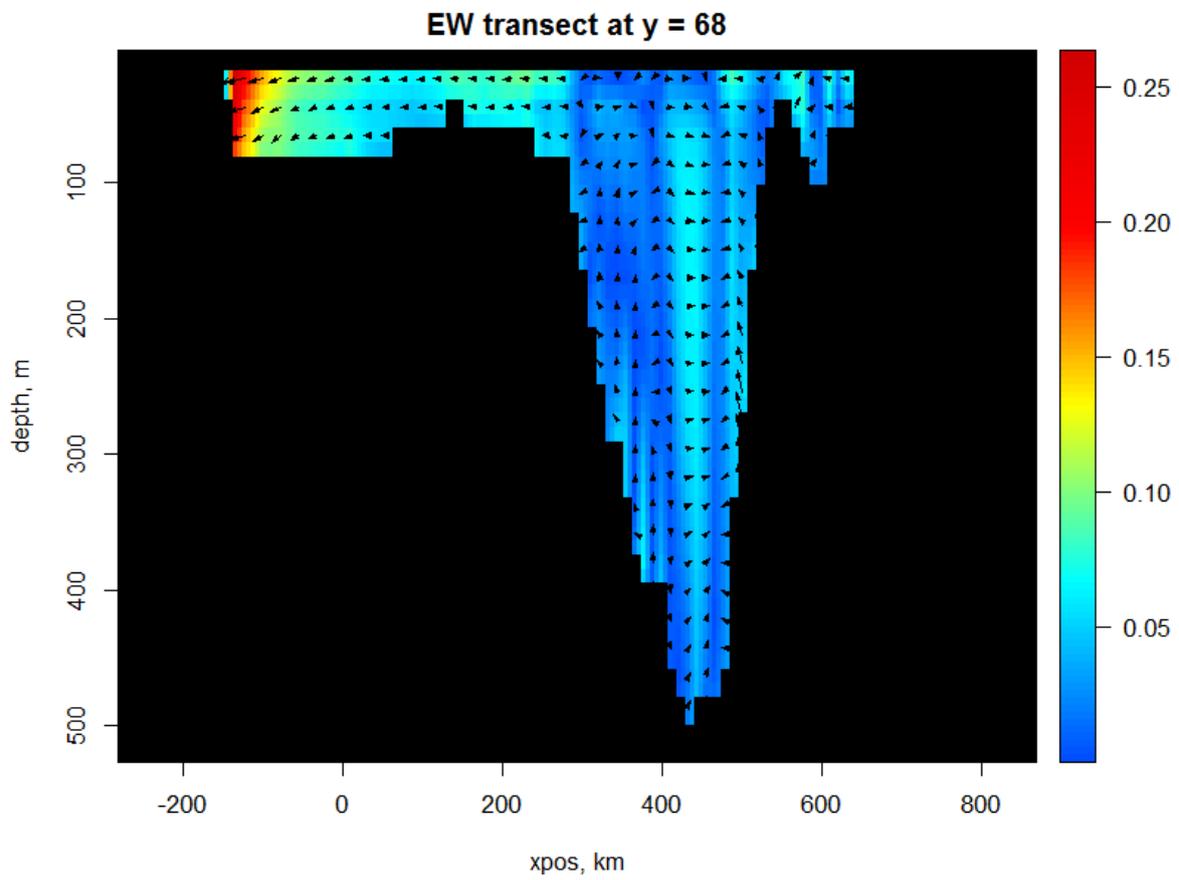
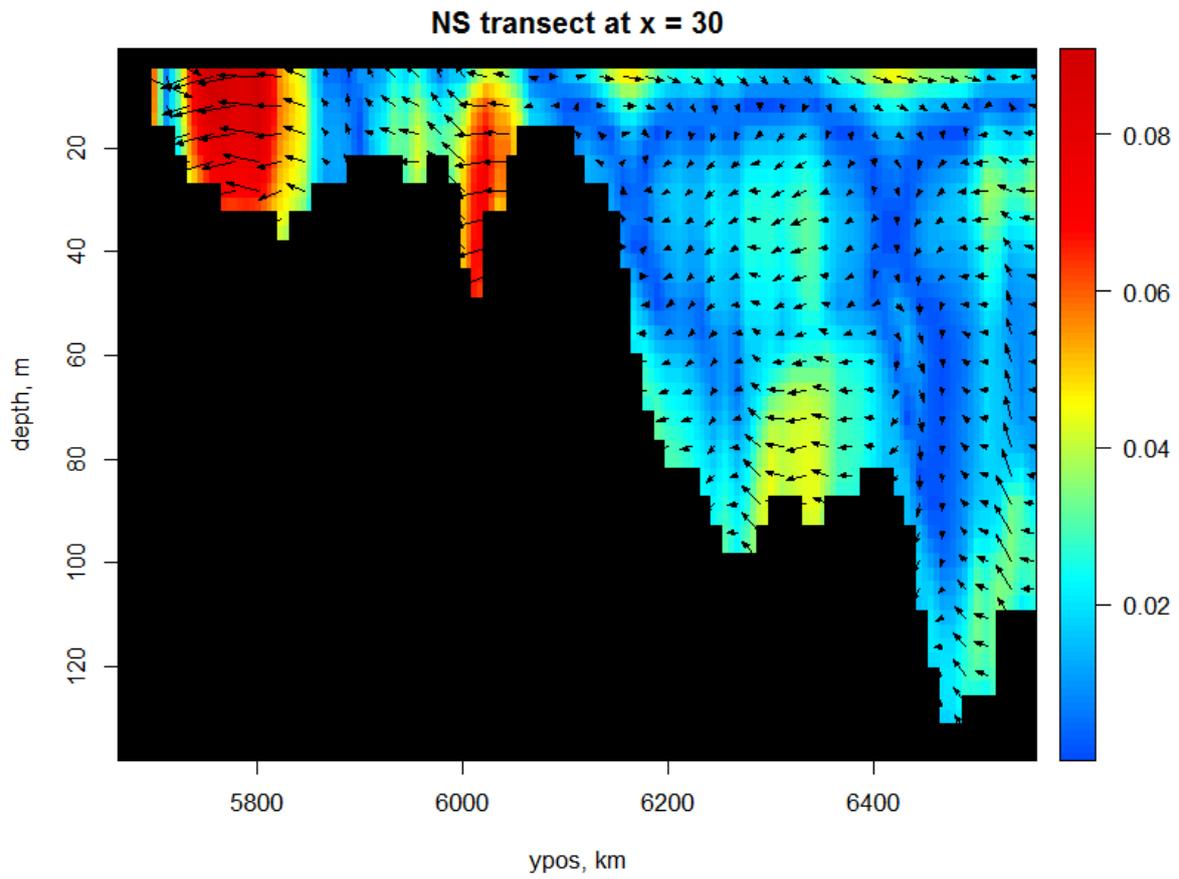
```
par(mfrow = c(2, 1), mar = c(4, 4, 2, 2))
# NS transect
Tran.u <- mapsigma(umean[30 , -y.remove, ], sigma = depth[30, -y.remove,])
Tran.w <- mapsigma(wmean[30 , -y.remove, ], sigma = depth[30, -y.remove,])
TotVelo <- sqrt(Tran.u$var^2 + Tran.w$var^2 )
DD <- Tran.w$depth
# ranges of u and v
range(Tran.u$var, na.rm = TRUE)
```

```
[1] -0.09098173  0.12890817
```

```
range(Tran.w$var, na.rm = TRUE)
```

```
[1] -7.923670e-05  7.631142e-05
```

```
#
image2D(TotVelo, y = DD, x = yc[ -y.remove], col = col,
        ylim = rev(range(DD)), NAcol = "black", resfac = 2,
        ylab = "depth, m", xlab = "ypos, km",
        main = "NS transect at x = 30")
quiver2D(Tran.u$var, Tran.w$var*1000, arr.min = 0.1, arr.max = 0.1,
        y = DD, x = yc[ -y.remove],
        by = c(2,1), scale = 3, add = TRUE)
#
# EW transect
Tran.v <- mapsigma(vmean[-x.remove , 68, ], sigma = depth[-x.remove, 68,])
Tran.w <- mapsigma(wmean[-x.remove , 68, ], sigma = depth[-x.remove, 68,])
TotVelo <- sqrt(Tran.v$var^2 + Tran.w$var^2 + 1e-8)
DD <- Tran.w$depth
image2D(TotVelo, y = DD, x = xc[ -x.remove], col = col,
        ylim = rev(range(DD)), NAcol = "black", resfac = 2,
        ylab = "depth, m", xlab = "xpos, km",
        main = "EW transect at y = 68")
quiver2D(Tran.v$var, Tran.w$var*1000, arr.min = 0.1, arr.max = 0.1,
        y = DD, x = xc[ -x.remove],
        by = c(2, 1), scale = 3, add = TRUE)
mtext (outer = TRUE, side = 3, "Flow transects")
```



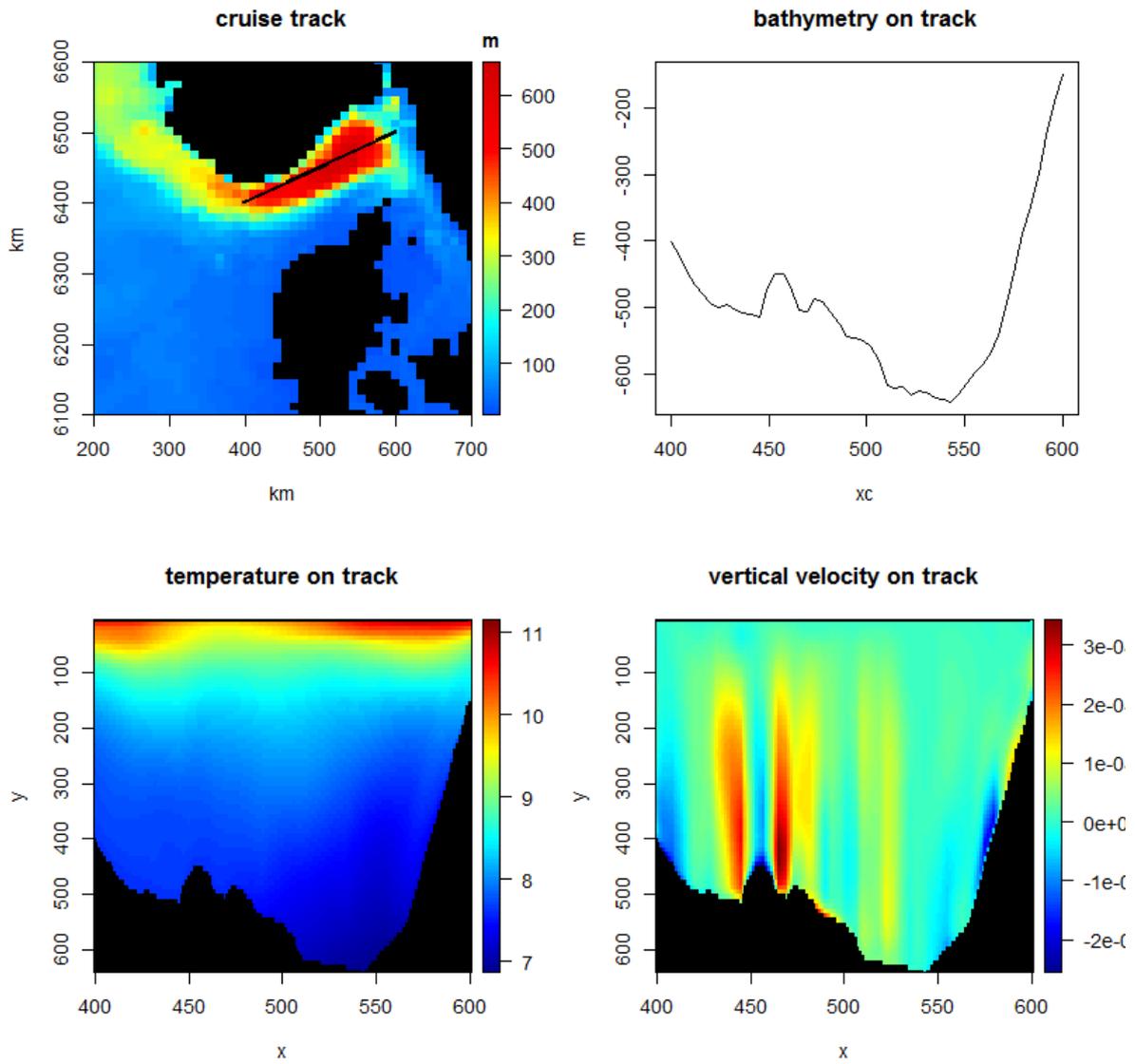
## 12. plotting irregular transects, along a ship track

For this figure, we will extract data along a (hypothetical) ship track:

```

track <- cbind(seq(400, 600, length.out = 50),
              seq(6400, 6500, length.out = 50))
# Draw the bathymetry on km grid
par(mfrow = c(2, 2))
image2D(bathy, x = xc, y = yc, NAcol = "black",
        xlim = c(200, 700), ylim = c(6100, 6600),
        col = col, xlab = "km", ylab = "km",
        clab = c("", "", "m"), main = "cruise track")
# add track
points(track, pch = ".", cex=3)
# Depth on track
trackBat <- extract(var = bathy, x = xc, y = yc, xyto = track)
plot(x = trackBat$xy[,1], y = -trackBat$var, type = "l", xlab = "xc",
     ylab = "m", main = "bathymetry on track")
# temperature
Avgtemp <- apply(temp, MARGIN = 1:3, FUN = mean, na.rm = TRUE)
tempSigma <- transectsigma(var = Avgtemp, sigma = depth,
                          x = as.vector(xc), y = as.vector(yc), to = track, resfac = 3)
image2D(tempSigma$var, x = tempSigma$x, y = tempSigma$depth,
        ylim = rev(range(tempSigma$depth)), NAcol = "black",
        main = "temperature on track")
# vertical velocity
wSigma <- transectsigma(var = wmean, sigma = depth,
                      x = as.vector(xc), y = as.vector(yc), to = track, resfac = 3)
image2D(wSigma$var, x = wSigma$x, y = wSigma$depth,
        ylim = rev(range(tempSigma$depth)), NAcol = "black",
        main = "vertical velocity on track")

```



### 13. Finally

This vignette was made with Sweave ([Leisch 2002](#)).

## References

- Burchard H, Bolding K (2002). *GETM, A General Estuarine Transport Model*. Scientific Documentation. EUR 20253 EN, URL <http://www.getm.eu>.
- Leisch F (2002). “Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis.” In W Härdle, B Rönz (eds.), *Compstat 2002 - Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL <http://www.stat.uni-muenchen.de/~leisch/Sweave>.
- Michna P, Woods M (2020). *RNetCDF: Interface to NetCDF Datasets*. R package version 2.4-2, URL <http://CRAN.R-project.org/package=RNetCDF>.
- R Development Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
- Soetaert K (2021a). *OceanView: Visualisation of Oceanographic Data and Model Output*. R package version 1.0.6, URL <http://CRAN.R-project.org/package=OceanView>.
- Soetaert K (2021b). *plot3D: Plotting multi-dimensional data*. R package version 1.4, URL <http://CRAN.R-project.org/package=plot3D>.
- Soetaert K (2021c). *plot3Drgl: Plotting multi-dimensional data - using rgl*. R package version 1.0.2, URL <http://CRAN.R-project.org/package=plot3Drgl>.

### Affiliation:

Karline Soetaert  
Royal Netherlands Institute of Sea Research (NIOZ)  
4401 NT Yerseke, Netherlands  
E-mail: [karline.soetaert@nioz.nl](mailto:karline.soetaert@nioz.nl)  
URL: <http://http://www.nioz.nl/>