Package OceanView - a short manual.

Karline Soetaert
NIOZ-Yerseke
The Netherlands

Abstract

The R (R Development Core Team 2013) package **OceanView** (Soetaert 2014a) is a compagnon to the packages **plot3D** (Soetaert 2014b) and **plot3Drgl** (Soetaert 2014c). These packages contain functions for visualising multidimensional data in base R graphics (**plot3D**) or in openGL (**plot3Drgl**).

OceanView has functions specifically designed for visualising complex oceanographic data.

In this vignette it is shown how to visualise flows, how to create movie sequences for depicting particle tracks in 2-D and 3-D, how to increase the resolution of multidimensional data or how to quickly produce plots of all columns in a data-frame or matrix.

Other examples of functions to visualise multi-dimensional data can be found in the help files or vignettes of the packages **plot3D** and **plot3Drgl**.

In another vignette in **OceanView**, (`vignette("Northsea")`) **OceanView** is used for plotting the output of a 3-D hydrodynamic model.

A graphical gallery using one of **plot3D**, **plot3Drgl** or **OceanView** is in http://www.rforscience.com/rpackages/visualisation/oceanview/.

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

1. Converting large data sets from long to cross-table format

This function was made to convert data from monitoring campaigns into a format suitable for creating images. Typically monitoring campaigns extend over a couple of sampling days, but when making images, these different sampling days should be treated as one campaign.

The long-term monitoring data from the NIOZ (Soetaert, Middelburg, Heip, Meire, Damme, and Maris 2006) for instance, in dataset `WSnioz`, contain a selection of the water quality data from the monthly sampling in the Westerschelde.

```
head (WSnioz, n = 2)
```

```
        SamplingDateTime SamplingDateTimeREAL Station Latitude Longitude
3798 1996-02-12 11:19:00             35107.47       1 51.41265   3.56628
3808 1996-02-12 11:19:00             35107.47       1 51.41265   3.56628
     VariableName                         VariableDesc VariableUnits
3798      SPMCHLA Chlorophyll-a in SPM (HPLC-FLU/DAD) microg Chla/l
3808      WCSALIN          Salinity measured with CTD           PSU
     DataValue
```

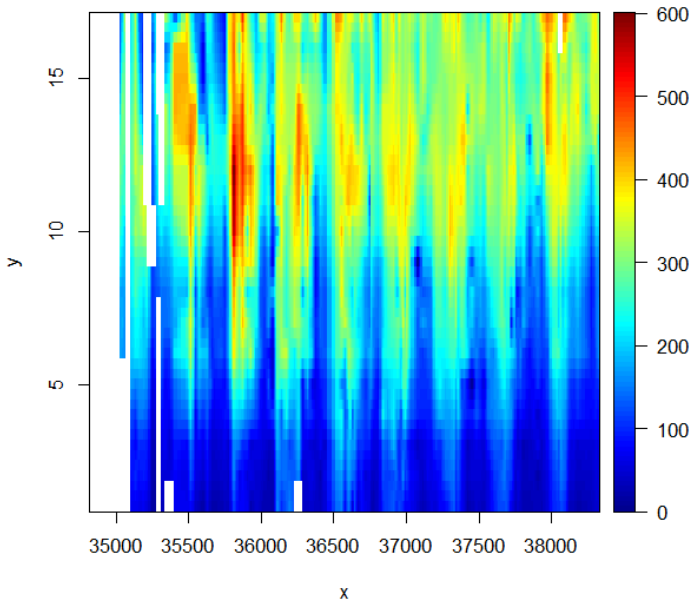

Figure 1: An image of spatio-temporal data

```
3798   2.30216
3808  31.70000
```

The data are in the format as extracted from the NIOZ database. To visualise its contents, it is easiest to put these data in cross-table format; here it is assumed that samplings that took place within 5 consecutive days belong to the same campaign (`df.row`).

```
NO3 <- db2cross(WSnioz, row = "SamplingDateTimeREAL",
         col = "Station", val = "DataValue",
         subset = (VariableName == "WNO3"), df.row = 5)
```

To create the image plot, the resolution is increased (`resfac`):

```
image2D(NO3, resfac = 3)
```

2. Quickly analysing and plotting several columns from a matrix

Oceanographers often have their data in a spreadsheet, where columns are different variables. The data set `WSnioz.table` contains the long-term monitoring data from the NIOZ in such a tabular format.

Function `Msummary` and `Mdescribe` create suitable summaries of the columns of tabular data sets.

```
  head(WSnioz.table, n = 2)
```

```
     SamplingDateTimeREAL Station   SPMCHLA WCSALIN WCTEMP     WNH4
[1,]             34822.02      11  4.163144     4.3  13.83 61.07143
[2,]             34822.47       9 10.078139     9.6  12.88 28.21429
         WNO2       WNO3 WO2     WPO4      WSi
[1,] 14.28571 0.42857143  NA 4.516129 181.1032
[2,]  8.50000 0.07142857  NA 3.709677 122.2064
```

```
  Msummary(WSnioz.table)
```

```
               variable factor_or_char     Min.     X1st.Qu.       Median
1  SamplingDateTimeREAL          FALSE 34822.02 35878.461806 36757.497569
2               Station          FALSE     1.00     5.000000     9.000000
3               SPMCHLA          FALSE     0.00     2.544327     4.938900
4               WCSALIN          FALSE     0.05     2.600000    13.100000
5                WCTEMP          FALSE     0.88     8.097500    12.566440
6                  WNH4          FALSE     0.00     3.297500     6.975000
7                  WNO2          FALSE     0.00     2.050000     3.870000
8                  WNO3          FALSE     0.00   132.207500   230.530000
9                   WO2          FALSE     0.28     4.422500     7.361092
10                 WPO4          FALSE     0.30     3.610000     4.785000
11                  WSi          FALSE     0.00    36.377500    86.030000
           Mean     X3rd.Qu.       Max.
1  36690.189719 37544.468594 38330.5729
2      9.139367    13.000000    17.0000
3     13.366919    12.215364   264.7097
4     13.266638    22.058010    33.1000
5     12.863026    17.720875    25.0500
6     47.090164    56.907500   798.6400
7      6.940766     8.385000    92.5500
8    225.648381   315.415000   600.9100
9      6.605326     9.148576    13.3800
10     5.208439     6.385000    25.0100
11   102.494002   165.737500   346.5200
```

Function `Mplot` is a quick way to visualise the contents of tabular data, while `Msplit` splits the data according to a factor. [1]

[1] Of course, there are many other functions in other packages that do similar things

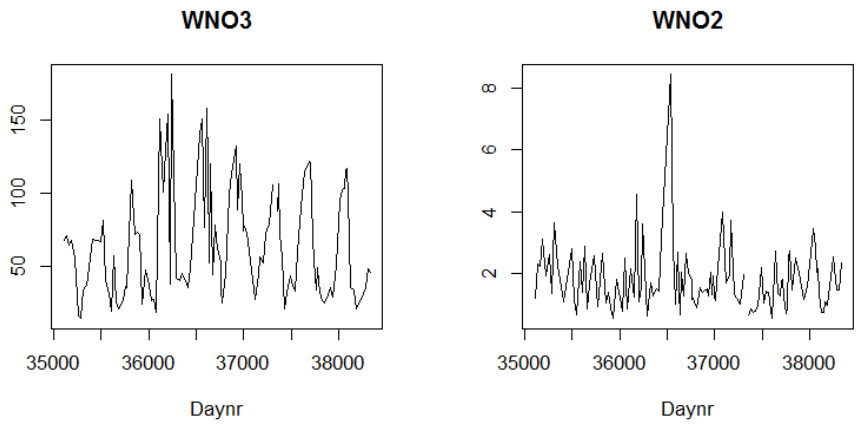

Figure 2: A quick plot of a (selection) of a tabular data set

As a first example, plot the contents of the tabular NIOZ monitoring data for station 1 and for two variables.

```
Mplot(WSnioz.table, subset = Station == 1,
  select = c("WNO3", "WNO2"), xlab = "Daynr")
```

We now plot the contents of the tabular NIOZ monitoring data for the stations 1 and 13. We first split the data set according to the station number, selecting these two stations (`Msplit`), then plot the timeseries for four variables.

```
Mplot(Msplit(WSnioz.table, "Station", subset = Station %in% c(1, 13)) ,
  select = c("WNO3", "WNO2", "WNH4", "WO2"), lty = 1, lwd = 2,
  xlab = "Daynr", log = c("y", "y", "y", ""),
  legend = list(x = "left", title = "Station"))
```

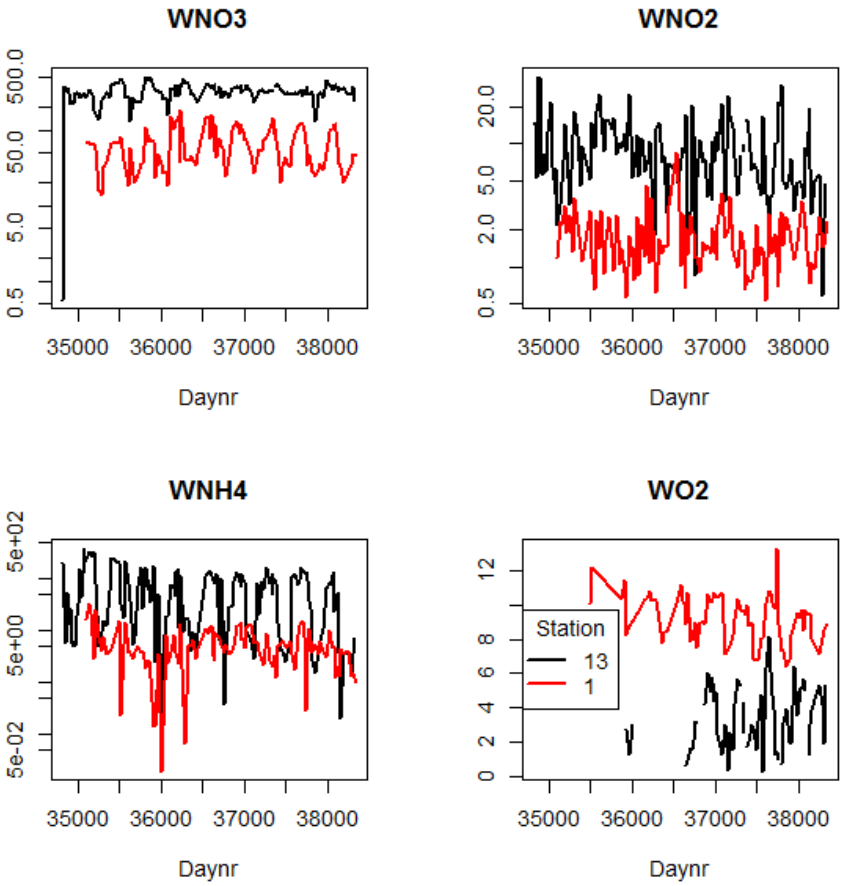

Figure 3: A quick plot of a (selection) of two data sets

3. Resolution and mapping to sigma coordinates

Sometimes, we may want to have data in higher or lower resolution. Package **OceanView** contains a quick-and-dirty, linear, interpolation method to increase (or decrease) the resolution. As it is written in R-code, it is not very fast.

Here we convert the dataset `volcano`, to very low resolution. (decreasing resolution is handy if you want to quickly visualise a very large dataset).

```
changeres(var = volcano, x = 1:nrow(volcano), y = 1:ncol(volcano), resfac = 0.1)
```

```
$var
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,]  100  101  101  108  108  107  103
 [2,]  109  112  141  162  149  119  106
 [3,]  122  140  179  184  187  149  106
 [4,]  113  136  165  163  173  153  110
 [5,]  110  125  154  161  138  120  107
 [6,]  113  129  154  150  139  111  104
 [7,]  115  131  150  134  126  103  100
 [8,]  102  109  110  116  100   96   96
 [9,]   97  100  100  100   96   94   94

$x
[1]  1 11 22 33 44 54 65 76 87

$y
[1]  1 11 21 31 41 51 61
```

The function `remap` is more flexible:

```
remap(var = volcano, x = 1:nrow(volcano), y = 1:ncol(volcano),
  xto = c(1, 20, 40), yto = c(2, 5))
```

```
$var
     [,1] [,2]
[1,]  100  101
[2,]  124  129
[3,]  108  113

$x
[1]  1 20 40

$y
[1] 2 5
```

The function `extract` interpolates to pairs of points

```
 extract(volcano, x = 1:nrow(volcano), y = 1:ncol(volcano),
   xyto = cbind(c(2, 5), c(5, 10)))

$var
[1] 102 103

$xy
     x  y
[1,] 2  5
[2,] 5 10
```

The mapping to sigma-coordinates is exemplified in the `vignette ("Northsea")`.

4. Plotting two-dimensional velocity data

Three functions were created to plot 2D velocity data: `quiver2D`, `flowpath` and `vectorplot`.

```
  par(mfrow = c(2, 2))
  x  <- seq(-1, 1, by = 0.2)
  y  <- seq(-1, 1, by = 0.2)
  dx <- outer(x, y , function(x, y) -y)
  dy <- outer(x, y , function(x, y) x)
 # velocity plot, different color for up/downward pointing arrows
  F <- quiver2D(u = dx, v = dy, x = x, y = y, colvar = dx > 0,
      col = c("red", "blue"), colkey = FALSE, arr.max = 0.3, arr.min = 0.1)
  legend("topright", bg = "white",
      legend = paste("max = ", format(F$speed.max, digits = 2)))
  names(F)
```

```
[1] "x0"        "y0"        "x1"        "y1"        "col"       "length"
[7] "speed.max"
```

```
  quiver2D(u = dx, v = dy, x = x, y = y, colvar = sqrt(dx^2 + dy^2),
      arr.max = 0.1, arr.min = 0.1, clab = "speed")
 # flow paths
  flowpath(u = dx, v = dy, x = x, y = y, numarr = 3,
    startx = 0.1, starty = 0.1)
  flowpath(u = dx, v = dy, x = x, y = y, col = "red", numarr = 2,
    startx = c(0.9, -0.9), starty = c(0.0, 0.0), add = TRUE)
 # vectorplots
  u <- rnorm(10)
  v <- rnorm(10)
  x <- y <- 1 : 10
  vectorplot(u = u, v = v, x = x, y = y, clim = c(0, 3),
    colvar = sqrt(u^2 + v^2), arr = TRUE)
  points(x, y)
```

```
[1] "x0"        "y0"        "x1"        "y1"        "col"       "length"
[7] "speed.max"
```

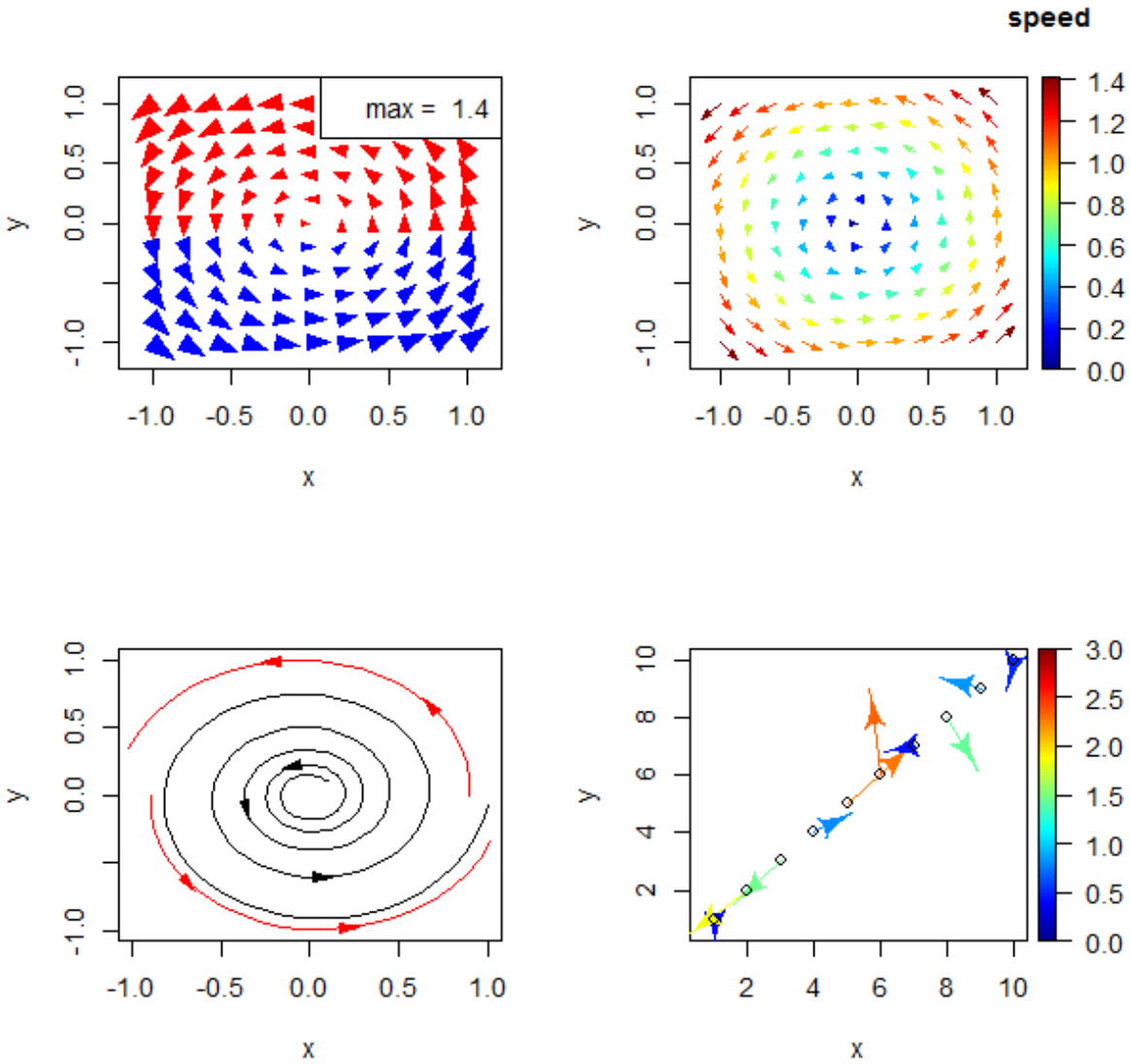

Figure 4: Several ways to visualise flows

5. Plotting temporally variable particle distributions

OceanView incorporates a number of functions to display the result of a particle transport (Lagrangian) model in two and three dimensions. It also comprises a data set with output from the Lagrangian Transport model (`Ltrans`) from Chesapeake Bay (Schlag and North 2012).

5.1. A quick view of particle distributions

`Ltrans` is an array of dimension (608 x 4 x 108) that contains for each of the 608 particles tracked, and at each of the 108 output steps the longitude, latitude, water depth and source region; the latter takes the values of 1 or 2.

```
dim(Ltrans)
```

```
[1] 608   4 108
```

We produce a quick view of the particle geographical position and water depth of all particles, on a bathymetric map of the area. We start by plotting the bathymetry, using grey scales. The color key is not drawn, but space for it is reserved (`plot = FALSE`). Then we add the particle positions using depth as the color variable.

```
image2D(Chesapeake$lon, Chesapeake$lat, z = Chesapeake$depth,
  col = grey(seq(1, 0., length.out = 100)), main = "Ltrans",
  colkey = list(plot = FALSE))
scatter2D(x = Ltrans[,1,], y = Ltrans[,2,], colvar = Ltrans[,3,],
  pch = ".", cex = 2, add = TRUE, clab = "depth, m")
```

5.2. Particle distributions in 2D

We can plot the temporal evolution of the particles in more detail, either using the traditional device (slow), or using open GL (fast).

We start by plotting the geographical position at selected time points, ignoring the depth (2-D output) using traditional graphics; the colors `green` and `orange` represent the source area of the particles. Note that we specify the bathymetric map of the area through the `image` argument of the function.

```
lon <- Chesapeake$lon
lat <- Chesapeake$lat
depth <- Chesapeake$depth
par(mfrow = c(2, 2))
for (i in seq(10, 106, length.out = 4))
   tracers2D(Ltrans[, 1, i], Ltrans[, 2, i],
             colvar = Ltrans[ ,4, i], col = c("green", "orange"),
             pch = 16, cex = 0.5,
             image = list(x = lon, y = lat, z = depth), colkey = FALSE,
             main = paste("time ", i))
```

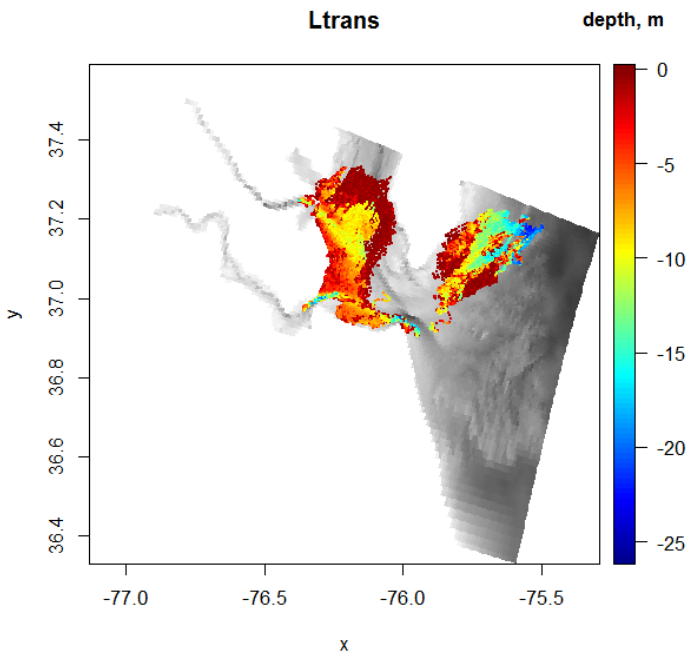

Figure 5: Distribution of particles in Chesapeake (all time instances)

In open GL, it works slightly different: first we create a 2D bathymetric map, on which we add the tracer positions. The output of this code is not shown, but the particles move very fast (on my computer), so you will probably want to slow it down. When using openGL, you can zoom in into specific regions of the plot, or cut slices (`cutrgl`).

```
image2Drgl (x = lon, y = lat, z = depth)
 for (i in seq(1, 108, by = 3)) {
   tracers2Drgl(Ltrans[, 1, i], Ltrans[, 2, i],
              colvar = Ltrans[ ,4, i], col = c("green", "orange"))
# remove # to slow down
#   Sys.sleep(0.1)
 }
```

5.3. Particle distributions in 3D

In a similar way, we can plot the temporal evolution of the 3-D positions (including depth) of particles using traditional or open GL graphics.

We start by plotting the geographical position and the depth, i.e. 3-D output and using traditional graphics. Note that we specify the drawing of the bathymetry of the area through the `surf` argument of the function (see e.g. `?persp3D` for its arguments).

```
lon <- Chesapeake$lon
lat <- Chesapeake$lat
```

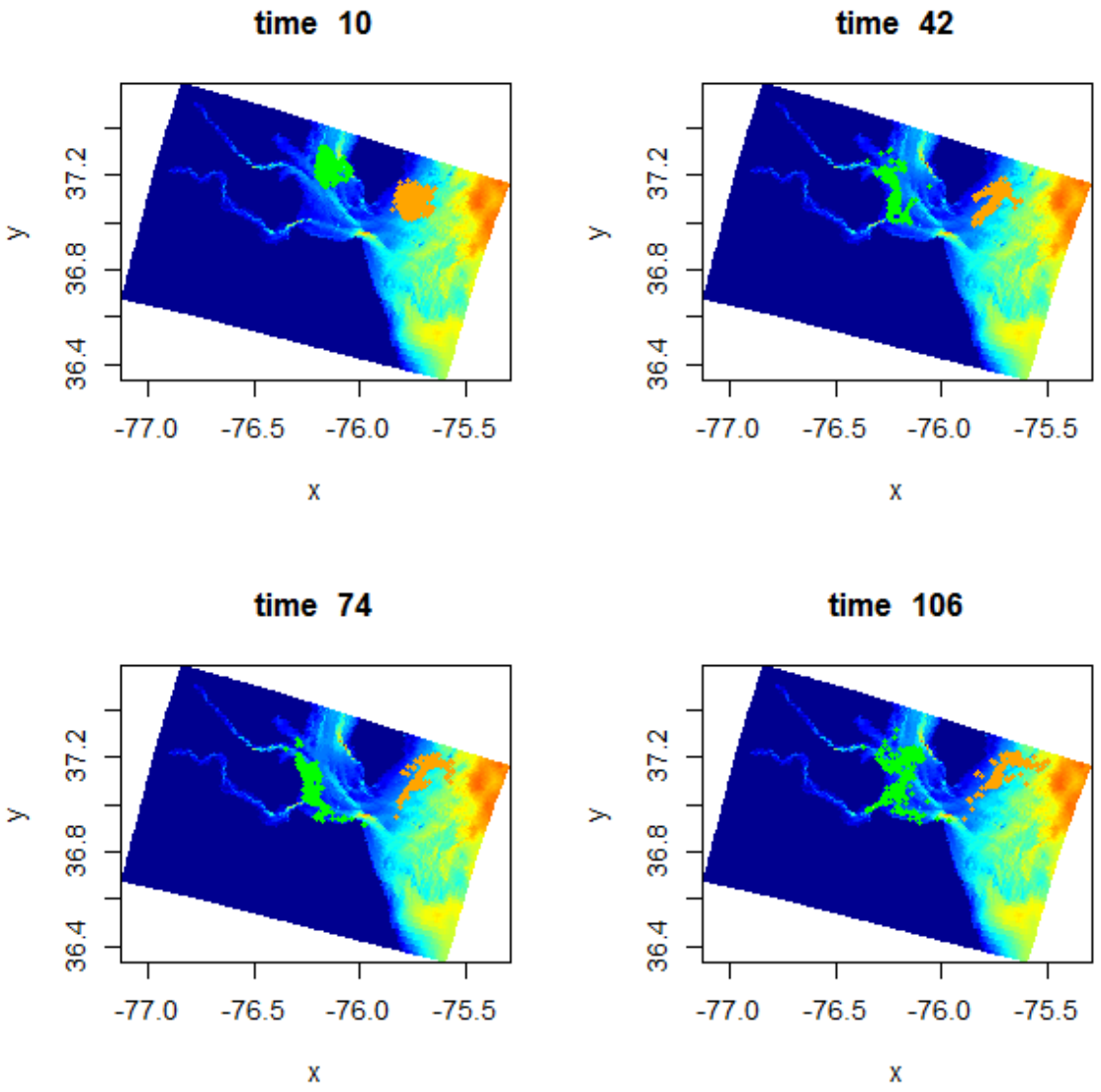

Figure 6: 2D distribution of particles in Chesapeake at selected time points using traditional graphics

```
depth <- Chesapeake$depth
par(mfrow = c(1, 2), mar = c(0, 0, 2, 0))
for (i in c(20, 100))
tracers3D(Ltrans[, 1, i], Ltrans[, 2, i], Ltrans[, 3, i],
          colvar = Ltrans[ ,4, i], col = c("green", "orange"),
          pch = 16, cex = 0.5,
          surf = list(x = lon, y = lat, z = -depth, scale = FALSE,
           expand = 0.02, colkey = FALSE, shade = 0.3, colvar = depth),
          colkey = FALSE, main = paste("time ", i))
```

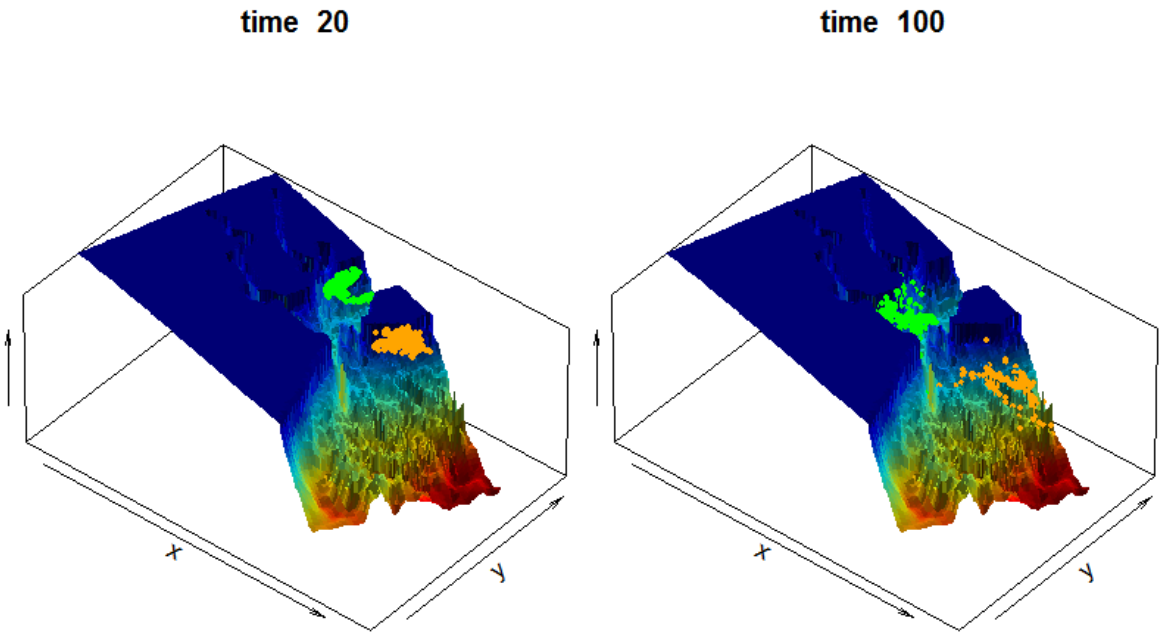

Figure 7: 3D distribution of particles in Chesapeake at two time points, traditional graphics

To do the same in open GL, we can use function `tracers3Drgl` (see help file of `Ltrans`), or use function `moviepoints3D`; the former requires to loop over the time points that we want to display, the latter requires input of the times, which should have the same length as x, y, z.

```
persp3Drgl(x = lon, y = lat, z = -depth, colvar = depth, scale = FALSE,
  expand = 0.02, main = "particle distribution",
  lighting = TRUE, smooth = TRUE)

nt <- dim(Ltrans)[3]  # number of time points
np <- dim(Ltrans)[1]  # number of particles
```

```
times <- rep(1:nt, each = np)

moviepoints3D(x = Ltrans[, 1, ], y = Ltrans[, 2, ], z = Ltrans[, 3, ],
              t = times, colvar = Ltrans[ ,4, ], col = c("green", "orange"),
              cex = 5, ask = TRUE)

}
```

The figure shows only one time point, after I have rotated the bathymetry a bit:

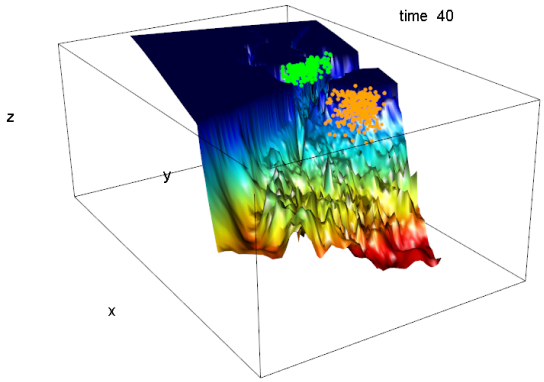

Figure 8: Screen capture of the 3D distribution of particles in Chesapeake in openGL

Note: a comparable function `movieslice3D` creates a sequence of 2-D slices along an axis of a full 3-D data set. See `example(movieslice3D)`.

6. Finally

This vignette was made with Sweave (Leisch 2002).

References

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.

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.

Schlag ZR, North EW (2012). *Lagrangian TRANSport model (LTRANS v.2) Users Guide*. Cambridge, MD.

Soetaert K (2014a). *OceanView: Visualisation of Oceanographic Data and Model Output*. R package version 1.0.

Soetaert K (2014b). *plot3D: Plotting multi-dimensional data*. R package version 1.0, URL http://cran.at.r-project.org/web/packages/plot3D/vignettes/plot3D.pdf.

Soetaert K (2014c). *plot3Drgl: Plotting multi-dimensional data - using rgl*. R package version 1.0, URL http://cran.at.r-project.org/web/packages/plot3Drgl/vignettes/plot3Drgl.pdf.

Soetaert K, Middelburg J, Heip C, Meire P, Damme SV, Maris T (2006). "Long-term change in dissolved inorganic nutrients in the heterotrophic Scheldt estuary (Belgium, the Netherlands)." *Limnology and Oceanography*, **51**. DOI: 10.4319/lo.2006.51.1_part_2.0409, URL http://aslo.org/lo/toc/vol_51/issue_1_part_2/0409.pdf.

Affiliation:

Karline Soetaert
Royal Netherlands Institute of Sea Research (NIOZ)
4401 NT Yerseke, Netherlands
E-mail: karline.soetaert@nioz.nl
URL: http://www.nioz.nl/staff-detail?id=784400