SampleSmartSea dataset

Ilaria Pia

16/03/2020

SampleSmartSea dataset

The data are stored in two folders. The first one contains data collected in years 1975-2005, the second one contains prediction data for years 2006-2059. In each of the two folders, we have NETcdf files containing data collected at a specific survey site in a specific year. The survey sites consists of 18 different intensive stations. In addiction we have 5 NETcdf files containing Gulf of Bothnia (GoB) layers that accounts values (measurements or predictions) of variables such as: 'Effective.ocean.transport.along.i.axis', 'Effective.ocean.transport.along.j.axis', 'Ice.concentration.for.categories', 'Length.of.events.of..oxyc' , 'potential_temperature'. These variables are stored using different grid-types. According to the file denomination: grid_T files have scalar values, representative of the center of each cell. Here, such grid is used for temperature an hypoxia. U, V and W grids store vector components, they represent movement from one grid-cell to another, by storing the speed of the point between the two cells: U showing the east-west velocities, V the north-south and W the up-down ones. Such grids are used to store 'Effective.ocean.transport' variables. For the ice concentration we have daily average (1d) values, while for all the other variables we have monthly average (1m) values. Data are taken in year 2000 in the first folder and are predicted for year 2058 in the second folder. In the latter, we have a sixth raster-layer variable

First we create our function to open and store `.nc` data as a list of dataframes.

```
library(ncdf4)

open_nc = function(file, verbose = F) {
  data = nc_open(file)
  variables = names(data$var)
  if(verbose) print(data)

  l = NULL
  l[variables] = list(NULL)
  for (i in 1:data$nvars) {
    l[[i]] = ncvar_get(data,variables[i])
    nna=sum(is.na(l[[i]]))
    if(nna>0) cat(paste("Warning: variable", variables[i], "contains", nna, "NA values", "\n"))
  }
  nc_close(data)
  return(l)
}

summary_nc = function(nc, varlist = F) {
  if(varlist)  print(summary(nc))
  par(mfrow = c(2,2))
  for (i in 3:length(nc)) {
    cat(paste( "\n", names(nc)[i], "\n", sep = ""))
    print( summary(as.vector(nc[[i]])) )
```

```
    hist(nc[[i]], main = names(nc)[i], xlab ="" )
  }
}
```

Now we can have a look at the files, we consider the file '2055_sharkUS5B', that contains predictions for year 2055, at the station sharkUSB.

```
nc = open_nc("/home/piailari/Documents/thesis/SampleSmartSea/A005/2055_sharkUS5B.nc")
str(nc)
```

```
## List of 17
##  $ nav_lat             : num [1(1d)] 62.6
##  $ nav_lon             : num [1(1d)] 20
##  $ deptht_bounds       : num [1:2, 1:36] 0 3.01 3.01 6.03 6.03 ...
##  $ votemper            : num [1:36, 1:365] 5.74 5.74 5.74 5.74 5.74 ...
##  $ time_centered       : num [1:365(1d)] 3.31e+09 3.31e+09 3.31e+09 3.31e+09 3.31e+09 ...
##  $ time_centered_bounds: num [1:2, 1:365] 3.31e+09 3.31e+09 3.31e+09 3.31e+09 3.31e+09 ...
##  $ time_counter_bounds : num [1:2, 1:365] 3.31e+09 3.31e+09 3.31e+09 3.31e+09 3.31e+09 ...
##  $ vosaline            : num [1:36, 1:365] 5.25 5.25 5.25 5.25 5.25 ...
##  $ sossh               : num [1:365(1d)] 0.145 0.166 0.182 0.175 0.188 ...
##  $ nitrate             : num [1:36, 1:365] 1.72 1.72 1.72 1.72 1.72 ...
##  $ phosphate           : num [1:36, 1:365] 0.0626 0.0626 0.0626 0.0626 0.0626 ...
##  $ oxygen              : num [1:36, 1:365] 8.22 8.22 8.22 8.22 8.22 ...
##  $ ammonium            : num [1:36, 1:365] 0.0369 0.0369 0.0369 0.0369 0.0369 ...
##  $ zooplankton         : num [1:36, 1:365] 0.438 0.438 0.438 0.438 0.438 ...
##  $ phytoplankton1      : num [1:36, 1:365] 0.0152 0.0152 0.0152 0.0152 0.0152 ...
##  $ phytoplankton2      : num [1:36, 1:365] 0.0776 0.0776 0.0776 0.0776 0.0776 ...
##  $ phytoplankton3      : num [1:36, 1:365] 0.00127 0.00127 0.00127 0.00127 0.00127 ...
```

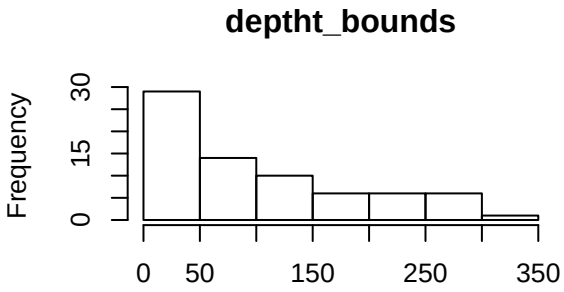
```
summary_nc(nc)
```

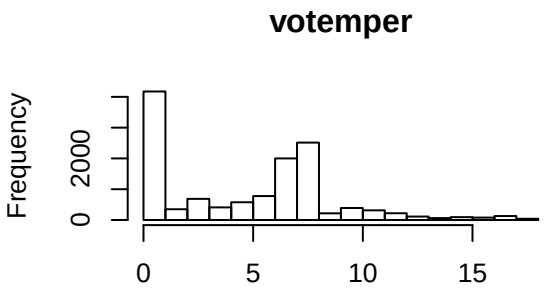
```
##
## deptht_bounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   28.26   69.15  102.34  164.27  318.42
```

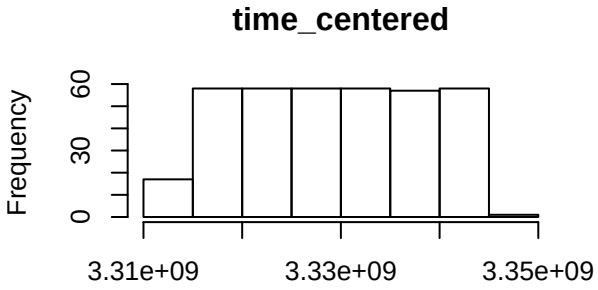
```
##
## votemper
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   5.548   4.660   7.208  17.793
```

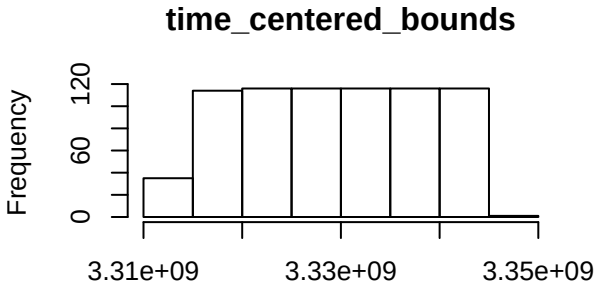
```
##
## time_centered
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 3.314e+09 3.321e+09 3.329e+09 3.329e+09 3.337e+09 3.345e+09
```

```
##
## time_centered_bounds
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 3.314e+09 3.321e+09 3.329e+09 3.329e+09 3.337e+09 3.345e+09
```







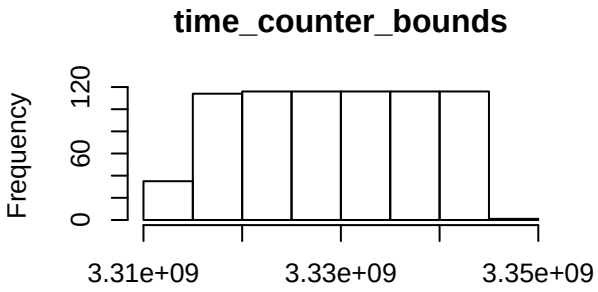


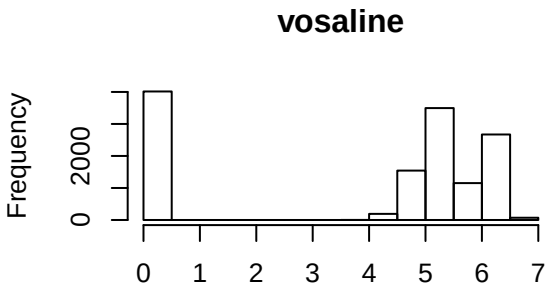

```
##
## time_counter_bounds
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 3.314e+09 3.321e+09 3.329e+09 3.329e+09 3.337e+09 3.345e+09
```

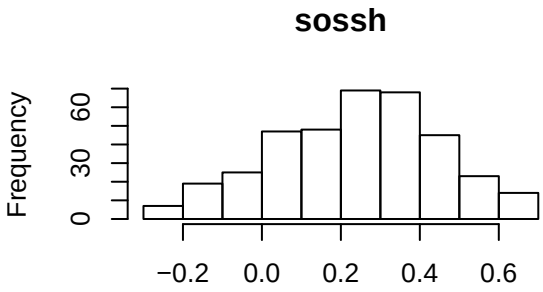
```
##
## vosaline
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   5.111   3.823   5.790   6.535
```

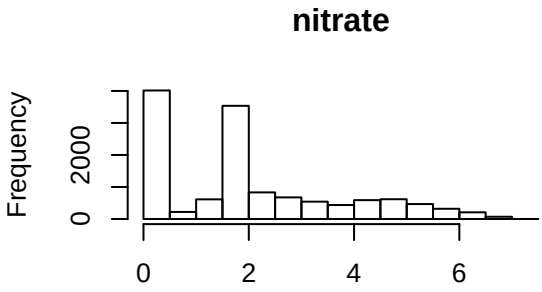
```
##
## sossh
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.28636  0.08844  0.26204  0.23814  0.38191  0.66919
```

```
##
## nitrate
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.750   1.962   2.963   7.016
```

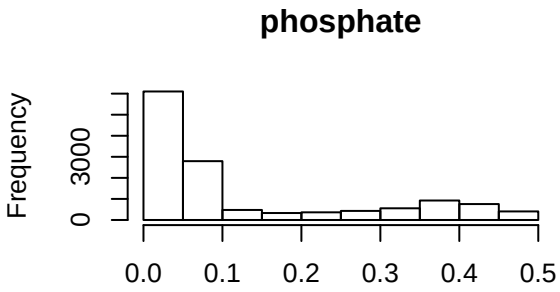


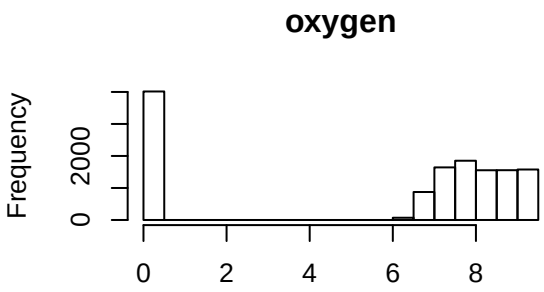


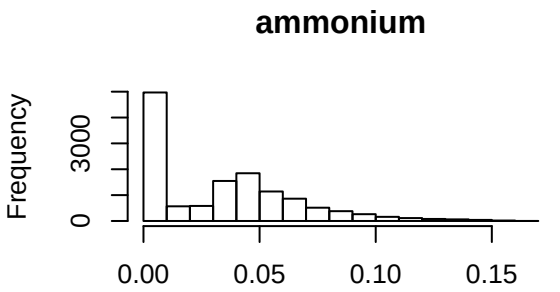


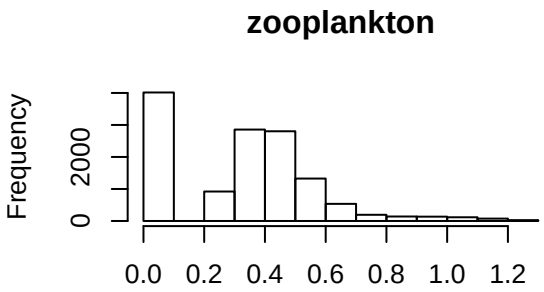



```
##
## phosphate
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.05573 0.12024 0.21942 0.48861

##
## oxygen
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   7.487   5.606   8.430   9.446

##
## ammonium
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.03320 0.03326 0.05238 0.16059

##
## zooplankton
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.3578  0.3195  0.4579  1.2393
```





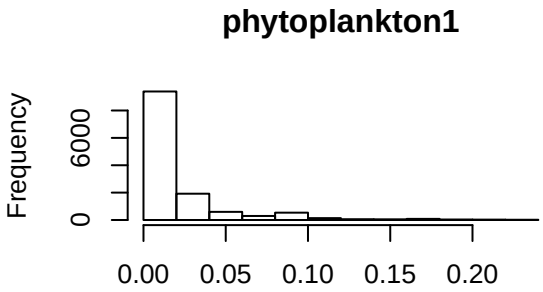


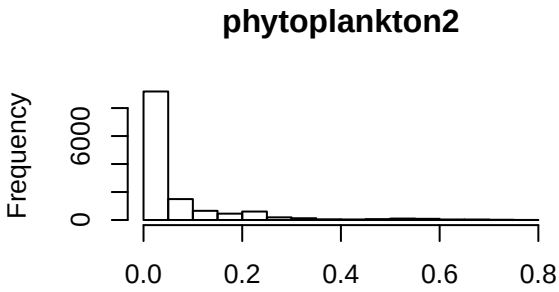


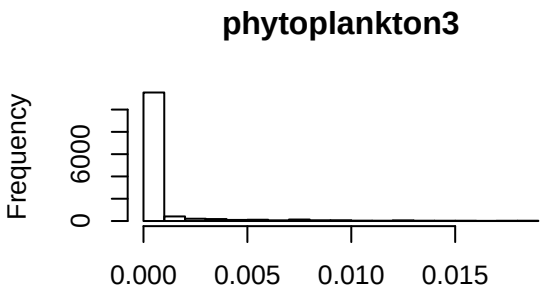

```
##
## phytoplankton1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00359 0.01880 0.02344 0.23204
```

```
##
## phytoplankton2
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000000 0.001121 0.058696 0.066628 0.759647
```

```
##
## phytoplankton3
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000e+00 0.000e+00 2.093e-06 7.563e-04 1.186e-04 1.889e-02
```







The file is a list of 17 variables. The variables 'nav_lat' and 'nav_lon' gives latitude and longitude of the station. There are three variables concerning the time. The remaining variables consist of environmental quantities, measured or predicted according to the year (smaller or greater than 2005). From the dimension of such variables we can observe that we have 36 measurements each day, taken at different depths. The variable 'depth_bounds' gives the depth boundaries at which each measurement was taken, in increasing order, that is from the most superficial, to the deepest values. The only variable with just one measurement per day is 'sossh'.

Now we consider one of the files containing Gob raster layers: 'GOB_1m_20580101_20581231_grid_T.nc'. We can observe that it contains layers for 4 different environmental variables: temperature, salinity, River runoffs, total precipitation. For each variable we have 12 different bands, each of them contains the monthly mean measurements in the Gulf of Bothnia area for the year 2058. The variables of interest are salinity and temperature, for them we also have 36 different levels, that correspond to 36 different depth bounds of the measurement. In addiction we have latitude, longitude and three time variables.

The file 'NORDIC-GOB_1d_20580101_20581231_grid_T.nc' has 365 different bands, containing the daily ice concentration and the daily ice thickness for the year 2058. Both variables are measured at 5 different levels. We are plotting the data for the first of April.

```
library("raster")
```

```
## Loading required package: sp
```

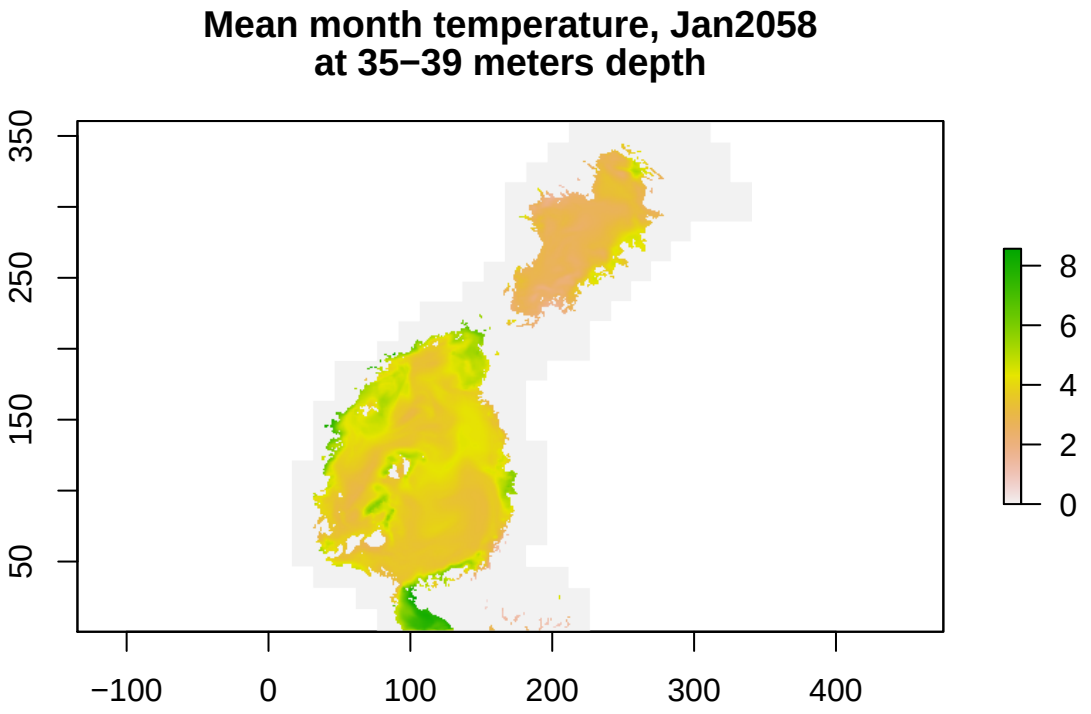
```
setwd("/home/piailari/Documents/thesis/SampleSmartSea/A005/")

#G = open_nc("NORDIC-GOB_1m_20580101_20581231_grid_T.nc")
#str(G)

Gob = raster("NORDIC-GOB_1m_20580101_20581231_grid_T.nc", varname="votemper", band =1, level=12, xmx=ma
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
Gob
```

```
## class      : RasterLayer
## band       : 1  (of  12  bands)
## dimensions : 360, 340, 122400  (nrow, ncol, ncell)
## resolution : 1, 1  (x, y)
## extent     : 0.5, 340.5, 0.5, 360.5  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : /home/local/piailari/Documents/thesis/SampleSmartSea/A005/NORDIC-GOB_1m_20580101_205812
## names      : potential_temperature
## z-value    : 3409560000
## zvar       : votemper
## level      : 12
```

```
plot(Gob, main = c("Mean month temperature, Jan2058", "at 35-39 meters depth"))
```




```
Gob_d = raster("NORDIC-GOB_1d_20580101_20581231_grid_T.nc", band =91, varname = "iceconccat" , level =2
```

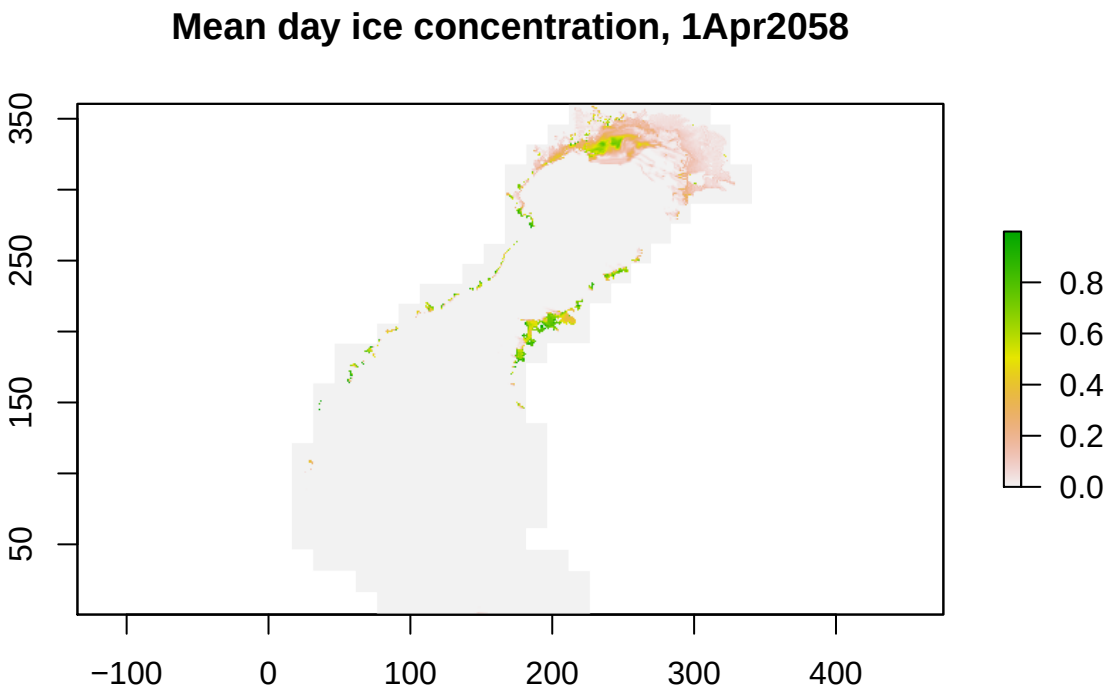
```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
Gob_d
```

```
## class      : RasterLayer
## band       : 91  (of  365  bands)
## dimensions : 360, 340, 122400  (nrow, ncol, ncell)
## resolution : 1, 1  (x, y)
## extent     : 0.5, 340.5, 0.5, 360.5  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : /home/local/piailari/Documents/thesis/SampleSmartSea/A005/NORDIC-GOB_1d_20580101_205812
## names      : Ice.concentration.for.categories
## z-value    : 3416040000
## zvar       : iceconccat
```

```
## level      : 2
```

```
plot(Gob_d, main = "Mean day ice concentration, 1Apr2058")
```



Note the warning message: 'I was asked to get a varid for dimension named x BUT this dimension HAS NO DIMVAR!' and 'I was asked to get a varid for dimension named y BUT this dimension HAS NO DIMVAR!'. The plot is made using different coordinates, than latitude and longitude.

We now plot the raster layer containing temperature measurements, made for the most superficial waters (0-3 meters depth) in January and June for year 2058 and 2000. We add the stations point locations.

```
### Future
# set path and filename
ncpath <- "/home/piailari/Documents/thesis/SampleSmartSea/A005/"

ncname = c()
year = sort(rep(2006:2059,18))
type = rep(c("B7","BO3","BO5","C3","F3","F9","F16","F64","Forsmark", "Furuogrund", "Kalix","MS4", "NB1"
for (i in 1:length(year)) {
  ncname[i] = paste(year[i], "_shark", type[i],  sep="")
}
ncfname.fut <- paste(ncpath, ncname, ".nc", sep="")
```

```
setwd("/home/piailari/Documents/thesis/SampleSmartSea/A005/")

## January 2058
r = raster("NORDIC-GOB_1m_20580101_20581231_grid_T.nc")
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
r
```

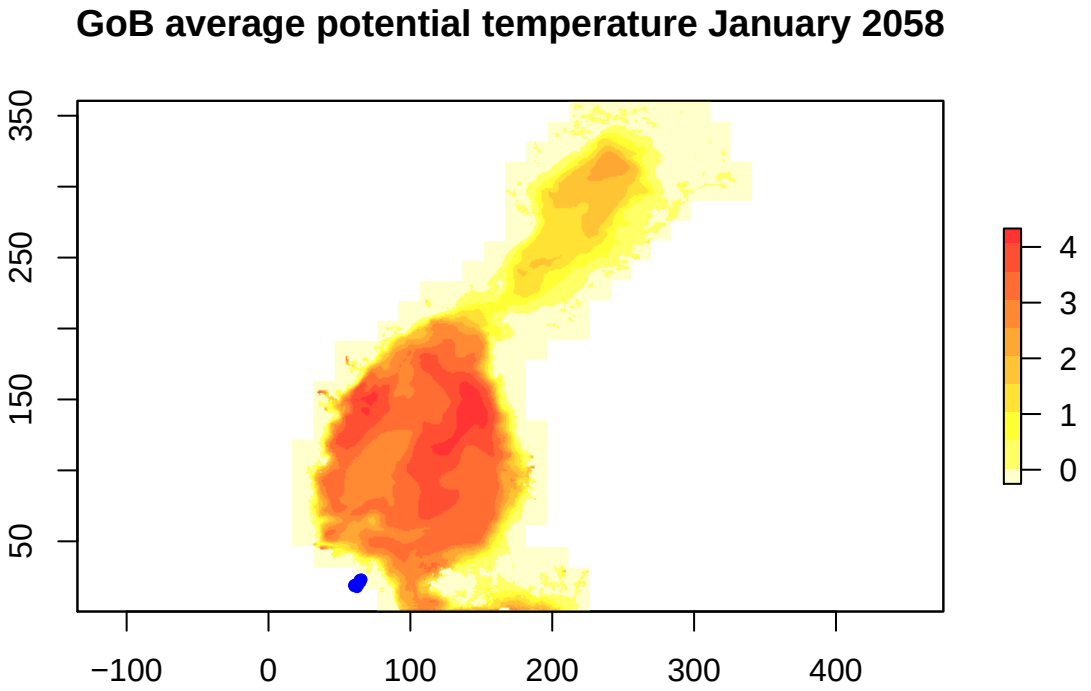
```
## class      : RasterLayer
## band       : 1  (of  12  bands)
## dimensions : 360, 340, 122400  (nrow, ncol, ncell)
```

```
## resolution : 1, 1  (x, y)
## extent     : 0.5, 340.5, 0.5, 360.5  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : /home/local/piailari/Documents/thesis/SampleSmartSea/A005/NORDIC-GOB_1m_20580101_205812
## names      : potential_temperature
## z-value    : 3409560000
## zvar       : votemper
## level      : 1
```

```
summary(r$potential_temperature)
```

```
##         potential_temperature
## Min.                -0.2532143
## 1st Qu.              0.0000000
## Median               0.7608339
## 3rd Qu.              2.9112134
## Max.                 4.3311162
## NA's                 0.0000000
```

```
plot(r, ylim = c(-50,400), main = "GoB average potential temperature January 2058", col = rev(heat.colo

# we add the 18 stations location on GoB
for (i in 1:18) {
  nc = open_nc(ncfname.fut[i])
  points(nc$nav_lat,nc$nav_lon, col =4, pch =20)

}
```




```
## June 2058
r = raster("NORDIC-GOB_1m_20580101_20581231_grid_T.nc", band =6)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

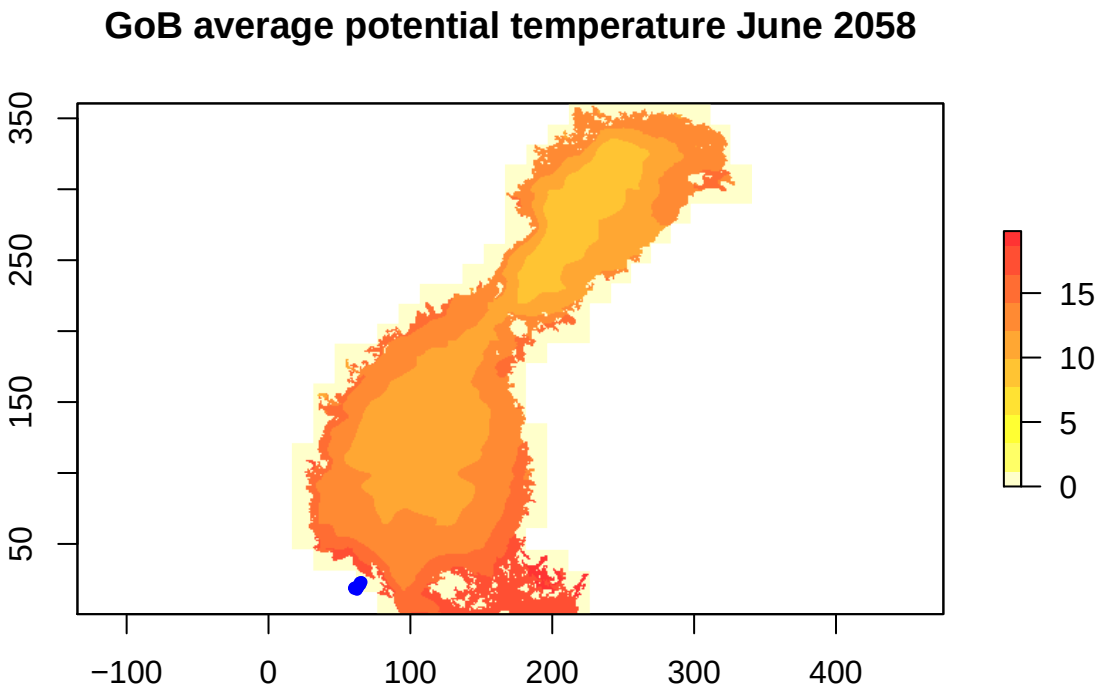
```
r
```

```
## class      : RasterLayer
## band       : 6  (of  12  bands)
## dimensions : 360, 340, 122400  (nrow, ncol, ncell)
## resolution : 1, 1  (x, y)
## extent     : 0.5, 340.5, 0.5, 360.5  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : /home/local/piailari/Documents/thesis/SampleSmartSea/A005/NORDIC-GOB_1m_20580101_205812
## names      : potential_temperature
## z-value    : 3422563200
## zvar       : votemper
## level      : 1
```

```
#print(r)
summary(r$potential_temperature)
```

```
##         potential_temperature
## Min.                0.000000
## 1st Qu.             8.910741
## Median             11.846814
## 3rd Qu.            13.191478
## Max.               19.810968
## NA's                0.000000
```

```
plot(r, ylim = c(-50,400), main = "GoB average potential temperature June 2058", col = rev(heat.colors(

# we add the 18 stations location on GoB
for (i in 1:18) {
  nc = open_nc(ncfname.fut[i])
  points(nc$nav_lat,nc$nav_lon, col =4, pch =20)

}
```



```
### Past
# set path and filename
ncpath <- "/home/piailari/Documents/thesis/SampleSmartSea/A001/"

ncname = c()
year = sort(rep(1975:2005,18))
type = rep(c("B7","BO3","BO5","C3","F3","F9","F16","F64","Forsmark", "Furuogrund", "Kalix","MS4", "NB1"
for (i in 1:length(year)) {
  ncname[i] = paste(year[i], "_shark", type[i],  sep="")
}
ncfname <- paste(ncpath, ncname, ".nc", sep="")
```

```
setwd("/home/piailari/Documents/thesis/SampleSmartSea/A001")

## January 2000
r = raster("NORDIC-GOB_1m_20000101_20001231_grid_T.nc")
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

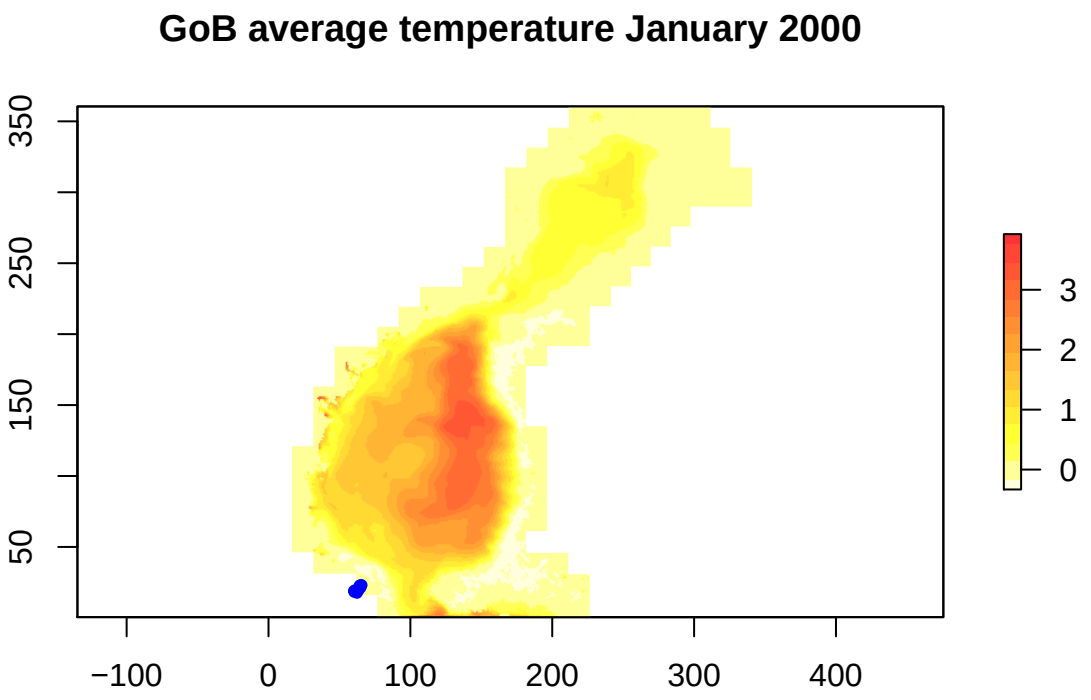
```
r
```

```
## class      : RasterLayer
## band       : 1  (of  12  bands)
## dimensions : 360, 340, 122400  (nrow, ncol, ncell)
## resolution : 1, 1  (x, y)
## extent     : 0.5, 340.5, 0.5, 360.5  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : /home/local/piailari/Documents/thesis/SampleSmartSea/A001/NORDIC-GOB_1m_20000101_200012
## names      : potential_temperature
## z-value    : 1579176000
## zvar       : votemper
## level      : 1
```

```
summary(r$potential_temperature)
```

```
##         potential_temperature
## Min.               -0.3320562
## 1st Qu.             0.0000000
## Median              0.2694537
## 3rd Qu.             1.3517581
## Max.                3.9283943
## NA's                0.0000000
```

```
plot(r, ylim = c(-50,400),  main = "GoB average temperature January 2000", col = rev(heat.colors(15)), a

for (i in 1:18) {
  nc = open_nc(ncfname[i])
  points(nc$nav_lat,nc$nav_lon, col =4, pch =20)
}
```




```
## June 2000
r = raster("NORDIC-GOB_1m_20000101_20001231_grid_T.nc", band=6) #other files has less interesting varia
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
r
```

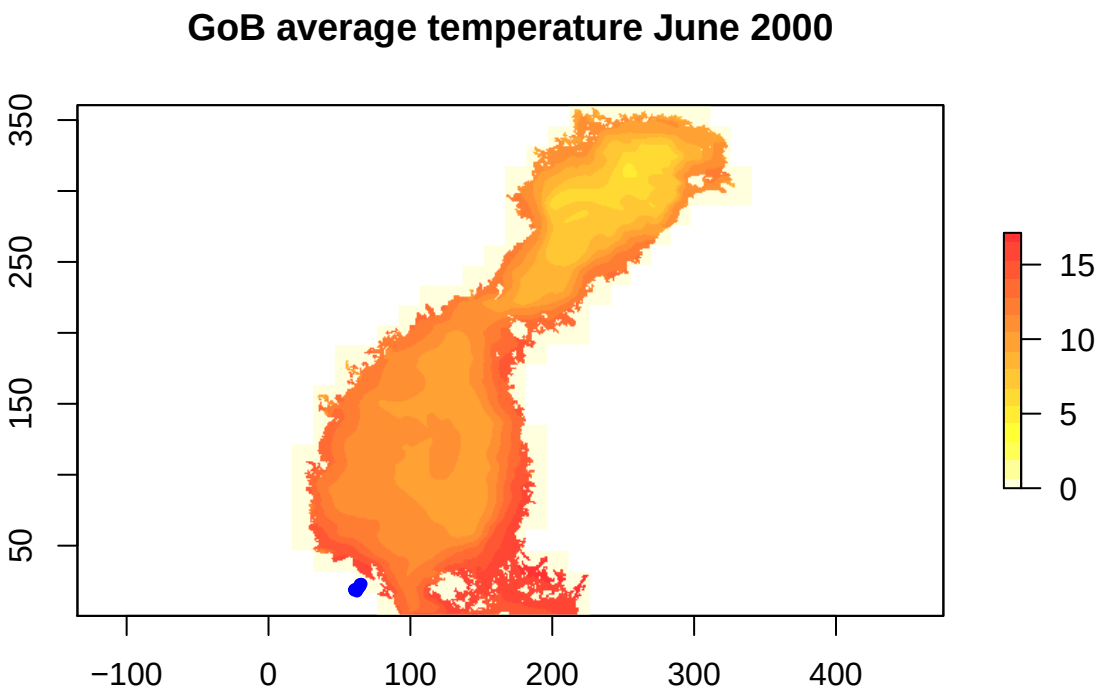
```
## class      : RasterLayer
## band       : 6  (of  12  bands)
## dimensions : 360, 340, 122400  (nrow, ncol, ncell)
## resolution : 1, 1  (x, y)
## extent     : 0.5, 340.5, 0.5, 360.5  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : /home/local/piailari/Documents/thesis/SampleSmartSea/A001/NORDIC-GOB_1m_20000101_200012
## names      : potential_temperature
## z-value    : 1592265600
## zvar       : votemper
## level      : 1
```

```
#print(r)
summary(r$potential_temperature)
```

```
##         potential_temperature
## Min.                 0.000000
## 1st Qu.              6.537467
## Median              10.337451
## 3rd Qu.             11.759617
## Max.                17.130022
## NA's                 0.000000
```

```
plot(r, ylim = c(-50,400),  main = "GoB average temperature June 2000", col = rev(heat.colors(15)), alp

for (i in 1:18) {
  nc = open_nc(ncfname[i])
```

```
  points(nc$nav_lat,nc$nav_lon, col =4, pch =20)
}
```



We plotted the Gob average temperatures in January and June of years 2000 and 2058. We added (blue spots) the survey sites. We can observe that they are all in the same area, and out of the sea boundary, hence there is something wrong with the coordinates obtained either from the stations or from the raster layers. If we extract longitude and latitude from the GoB files, we can observe they are different from the one reported on the map: the raster function used the data-point indexes instead of lat and lon variables from the netCDF file. We need to fix this:

```
library(raster)
setwd("/home/piailari/Documents/thesis/SampleSmartSea/A005/")
inputfile <- "NORDIC-GOB_1m_20580101_20581231_grid_T.nc"


lat <- raster(inputfile, varname="nav_lat")
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
lon <- raster(inputfile, varname="nav_lon")
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp <- raster(inputfile, varname="votemper",  band =6)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
## values drops the "raster" wrapper, just returns values in order as a vector
d <- cbind(values(lon), values(lat), values(temp))
dd =na.exclude(d)
## remap arbitrary values to [0,1] for a colour table
scl <- function(x) 1-(x - min(x, na.rm = TRUE))/diff(range(x, na.rm = TRUE))
```

```
n <- 56
plot(d[,1:2], ylim=c(60,66), xlim=c(10,32), pch = 16, col = (heat.colors(n)[scl(d[,3]) * (n-1) + 1]), x

# colorbar
library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.5-1 (2019-12-12) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: maps
```

```
## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```
my.colors = rev(heat.colors(56))
z=matrix(dd[,3],nrow=1)
x=0.1
y=seq(min(dd[,3]), max(dd[,3]),len=nrow(dd))
image.plot(x,y,z, legend.only=TRUE, col=my.colors,axes=FALSE,xlab="",ylab="",   smallplot= c(.80,.82,0.
#colorbar.plot(28,65, horizontal = F, strip.width = 0.02, strip.length = 0.1, col=rev(heat.colors(64)),

#North arrow
library(GISTools)
```

```
## Loading required package: maptools
```

```
## Checking rgeos availability: TRUE
```

```
## Loading required package: RColorBrewer
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following objects are masked from 'package:raster':
##
##     area, select
```

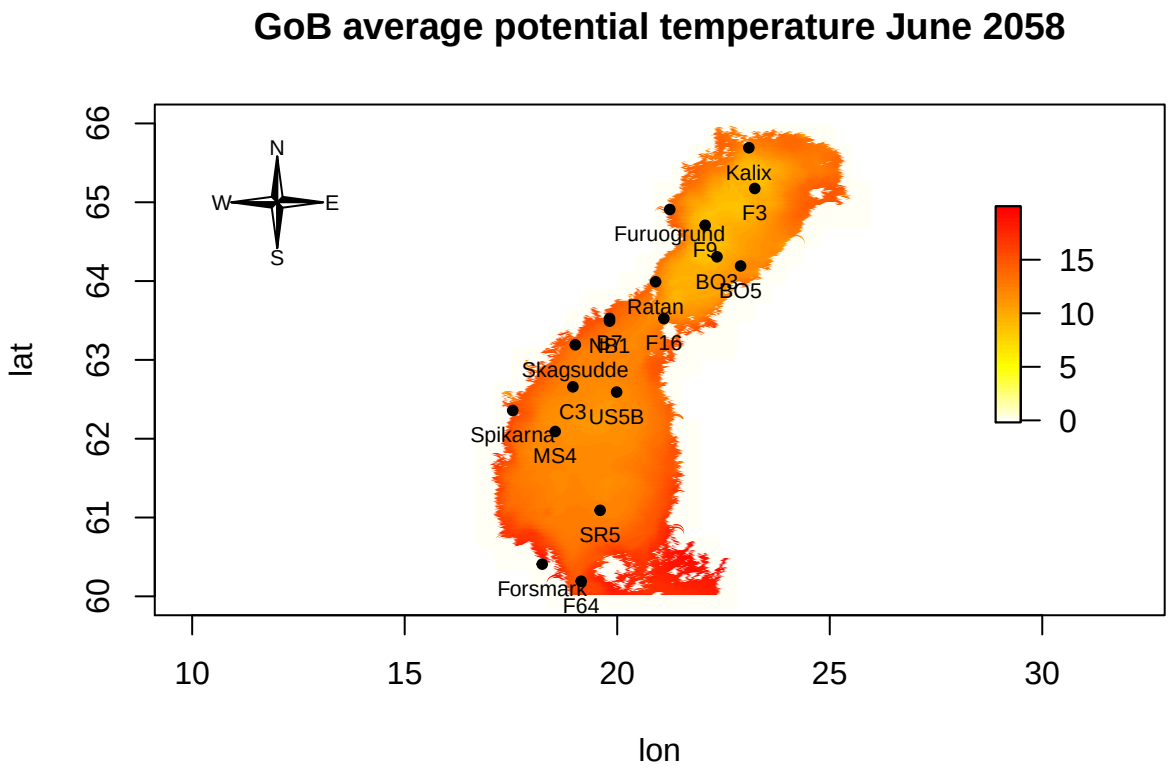
```
## Loading required package: rgeos
```

```
## rgeos version: 0.5-2, (SVN revision 621)
##  GEOS runtime version: 3.6.2-CAPI-1.10.2
##  Linking to sp version: 1.4-1
##  Polygon checking: TRUE
```

```
##
## Attaching package: 'GISTools'
```

```
## The following object is masked from 'package:maps':
##
##     map.scale
```

```
compassRose(12,65, cex = 0.7)
#north.arrow(28,60,len=0.2,lab='NORTH',cex.lab=0.7,tcol='black')

#Stations locations
for (i in 1:18) {
  nc = open_nc(ncfname.fut[i])
  points(nc$nav_lon,nc$nav_lat, col =1, pch =20)
  text(nc$nav_lon,nc$nav_lat, labels=type[i], col =1, pos=1, cex=0.7)

}
```



Temperature and salinity

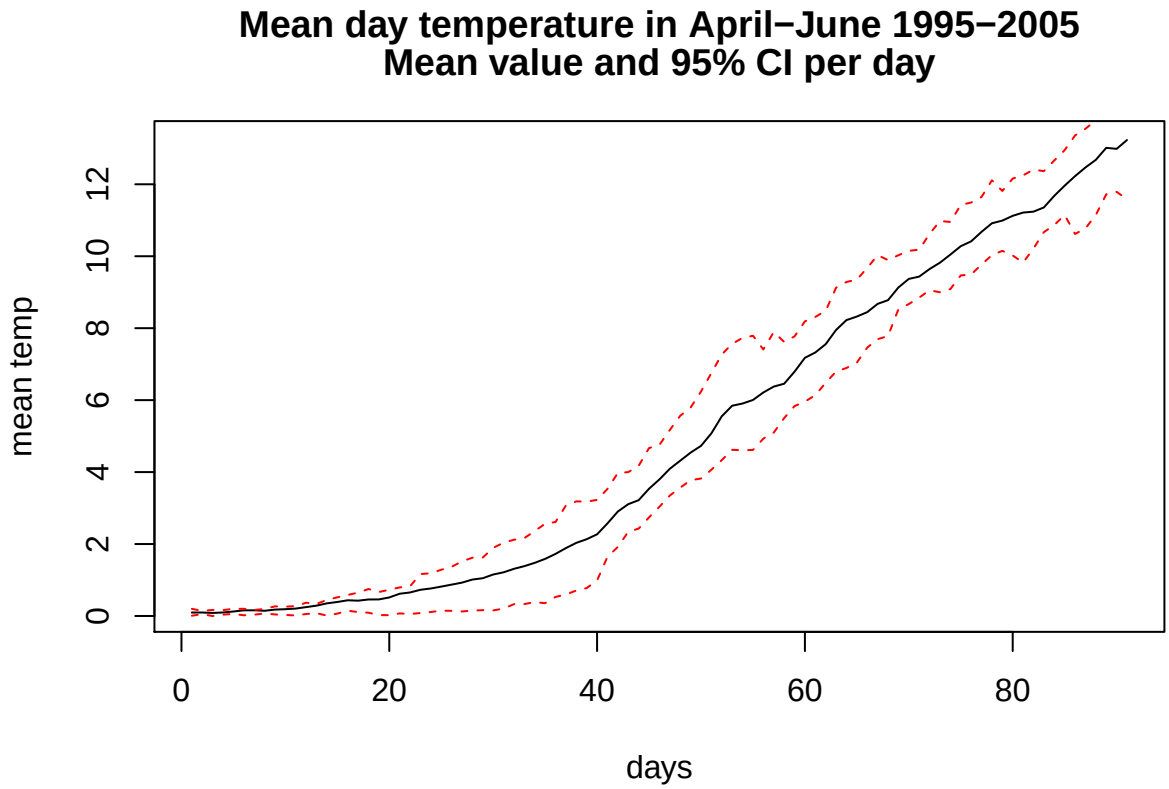
We are interested in studying how temperature and salinity impact on white fishes reproduction and on larvae number. We calculate and visualize the following:

- the average sea surface (0-10m depths) temperature of April-June for years 1995-2005 and 2030-2059
- the average sea surface (0-10m depths) salinity of April-June for years 1995-2005 and 2030-2059

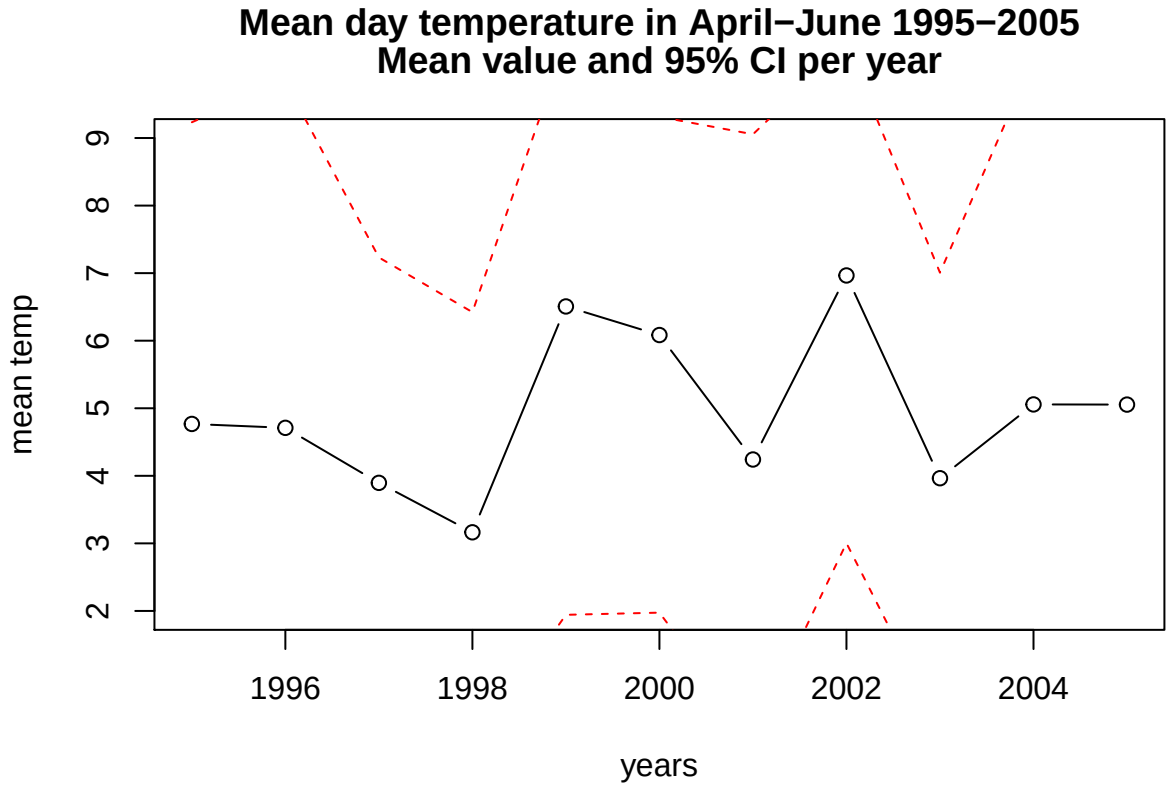
We are interested in values from April to June, from day 91 to day 181 (for leap years we will consider days 92-182), concerning the most superficial measurements, up to 9 meters depth: we consider the mean value of the first three daily measurements, made at a depth range of 0-3, 3-6 and 6-9 meters respectively. Indeed white fishes spawns hatch in spring, in superficial waters.

We consider the data files corresponding to the station 'sharkB7'. We start with data collected in years 1995 - 2005.

```
### Past - temperature

ncfnameB7 = ncfname[seq(1, length(ncfname), by = 18)]

temp = matrix(,nrow = length(ncfnameB7), ncol = 91*36)  #April - June
## mean.temp
# 31 rows : years 1995-2005
# 91 columns : days April-June
mean.temp = matrix(,nrow = length(ncfnameB7), ncol = length(91:181))
for(i in 1:length(ncfnameB7)) {
  nc = open_nc(ncfnameB7[i])
  if(length(nc$time_centered)==366) nc$votemper = nc$votemper[,-60] # we exclude the 29th of february f

  temp[i,] = as.vector(nc$votemper[,91:181])
  mean.temp[i,] = apply(nc$votemper[1:3,91:181], 2, mean) # mean daily temperature, for 3 different dep
}

# mean temperature in each day
par(mfrow= c(1,1))
# per day
#boxplot(mean.temp[21:31,],  main = c("Mean day temperature in April-June 1995-2005", "Distributions pe
day.mean.temp = apply(mean.temp[21:31,], 2, mean)
plot(day.mean.temp, main = c("Mean day temperature in April-June 1995-2005","Mean value and 95% CI per
day.q = apply(mean.temp[21:31,], 2, quantile)
lines(1:91, day.q[2,], col=2, lty=2)
lines(1:91, day.q[4,], col=2, lty=2)
```



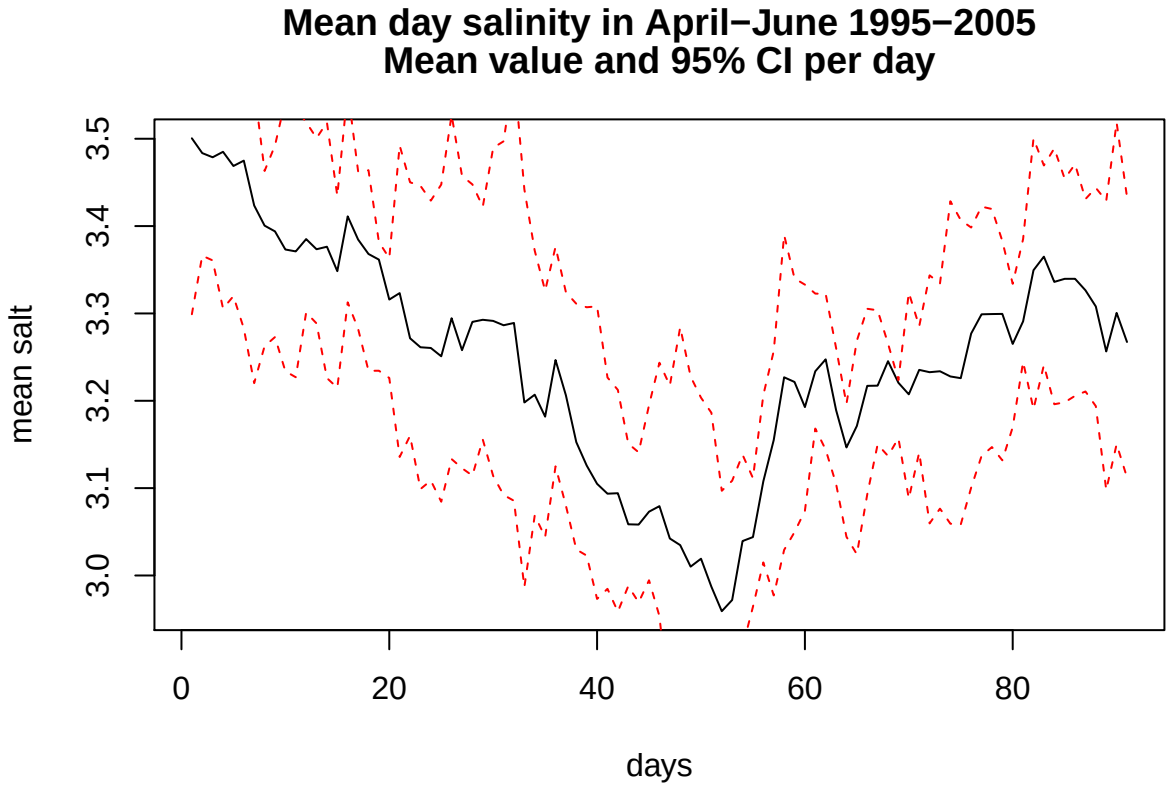

```
#per year
#boxplot(t(mean.temp[21:31,]),  main = c( "Mean day temperature in April-June 1995-2005", "Distribution
#axis(1, at = 1995:2005,  las=2,tick = T, labels = 1995:2005)
year.mean.temp = apply(mean.temp, 1, mean)
plot(1995:2005, year.mean.temp[21:31], main = c( "Mean day temperature in April-June 1995-2005", "Mean
year.q = apply(mean.temp, 1, quantile)
lines(1995:2005, year.q[2,21:31], col=2, lty=2)
lines(1995:2005, year.q[4,21:31], col=2, lty=2)
```



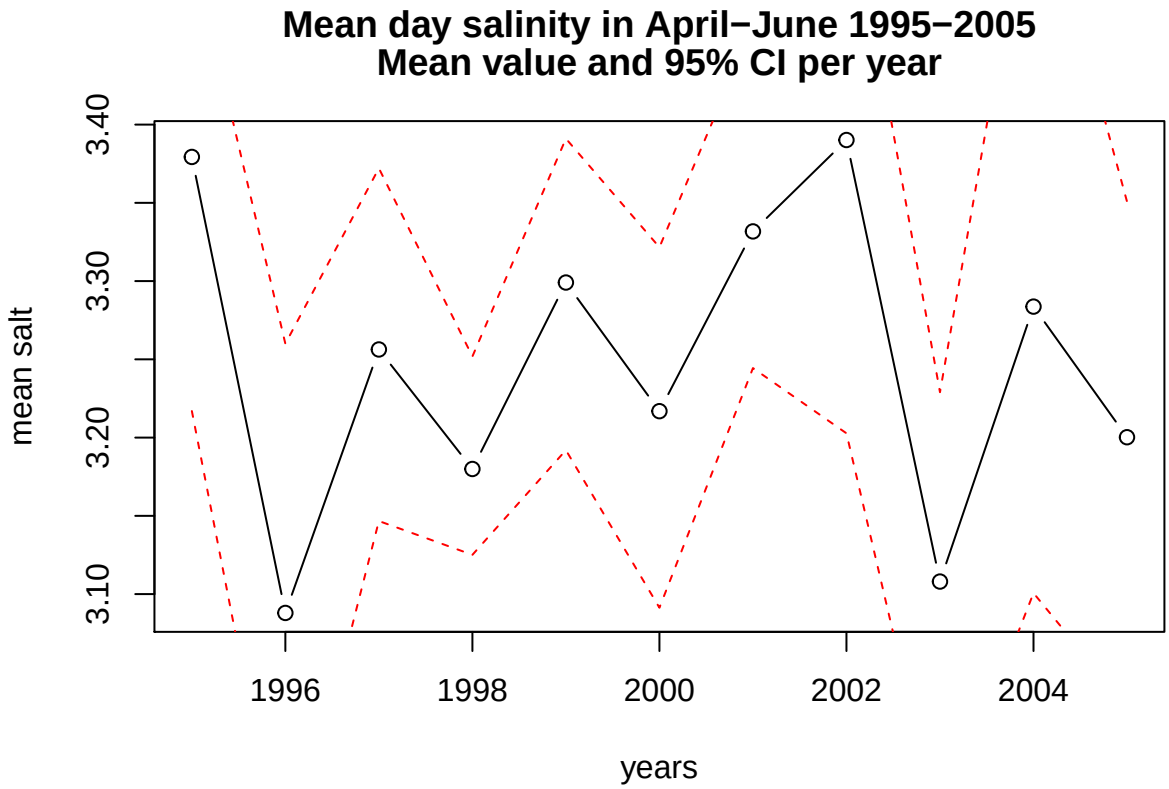
From the above plots we can observe how temperature generally increases from April to June, from 0 to 12 degrees. The increasing rate is low in the first month, it then grows in May. In the considered years (1995-2005) we can't observe a clear trend of the daily temperature: years 1999 and 2002 were the warmest ones, with a mean temperature around 7, year 1998 was the coldest, with a mean temperature around 3 degrees.

```
### Past - salinity

ncfnameB7 = ncfname[seq(1, length(ncfname), by = 18)]

salt = matrix(,nrow = length(ncfnameB7), ncol = 91*36)  #April - June
## mean.salt
# 31 rows : years 1995-2005
# 91 columns : days April-June
mean.salt = matrix(,nrow = length(ncfnameB7), ncol = length(91:181))
for(i in 1:length(ncfnameB7)) {
  nc = open_nc(ncfnameB7[i])
  if(length(nc$time_centered)==366) nc$vosaline = nc$vosaline[,-60] # we exclude the 29th of february f

  salt[i,] = as.vector(nc$vosaline[,91:181])
  mean.salt[i,] = apply(nc$vosaline[1:3,91:181], 2, mean)
}

# mean salinity in each day
par(mfrow= c(1,1))
# per day
#boxplot(mean.salt[21:31,],  main = c("Mean day salinity in April-June 1995-2005", "Distributions per d
day.mean.salt = apply(mean.salt[21:31,], 2, mean)
plot(day.mean.salt, main = c("Mean day salinity in April-June 1995-2005","Mean value and 95% CI per day
```

```
day.q = apply(mean.salt[21:31,], 2, quantile)
lines(1:91, day.q[2,], col=2, lty=2)
lines(1:91, day.q[4,], col=2, lty=2)
```




```
#per year
#boxplot(t(mean.salt[21:31,]),  main = c( "Mean day salinity in April-June 1995-2005", "Distributions p
#axis(1, at = 1995:2005,  las=2,tick = T, labels = 1995:2005)
year.mean.salt = apply(mean.salt, 1, mean)
plot(1995:2005, year.mean.salt[21:31], main = c( "Mean day salinity in April-June 1995-2005", "Mean valu
year.q = apply(mean.salt, 1, quantile)
lines(1995:2005, year.q[2,21:31], col=2, lty=2)
lines(1995:2005, year.q[4,21:31], col=2, lty=2)
```

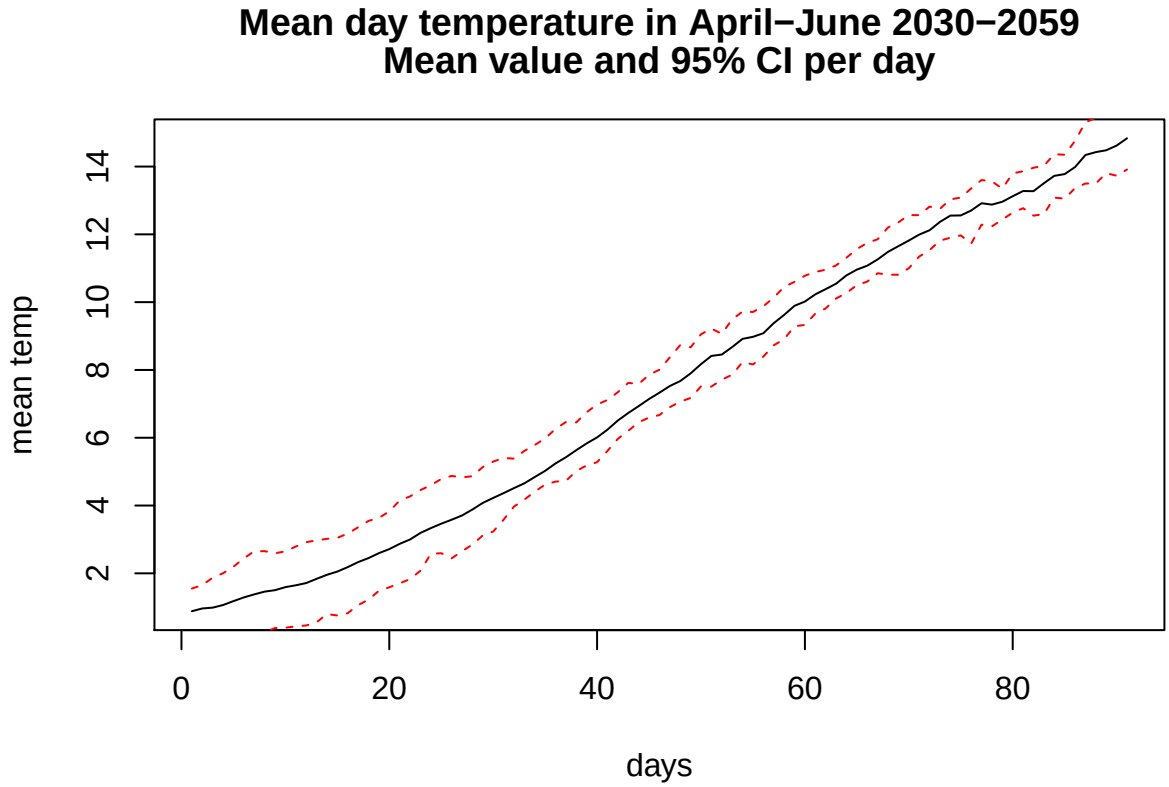


From the above plots we can observe how salinity varies very little in the years 1995-2005, from a minimum around 3.10 to a maximum around 3.4, there is no evident trend. If we consider the the daily salinity , averaged in the 31 years, we can observe how salinity is constantly decreases from 3.5 to a minimum around 3 reached around mid of May, to increase again up to 3.3 in the end of June.

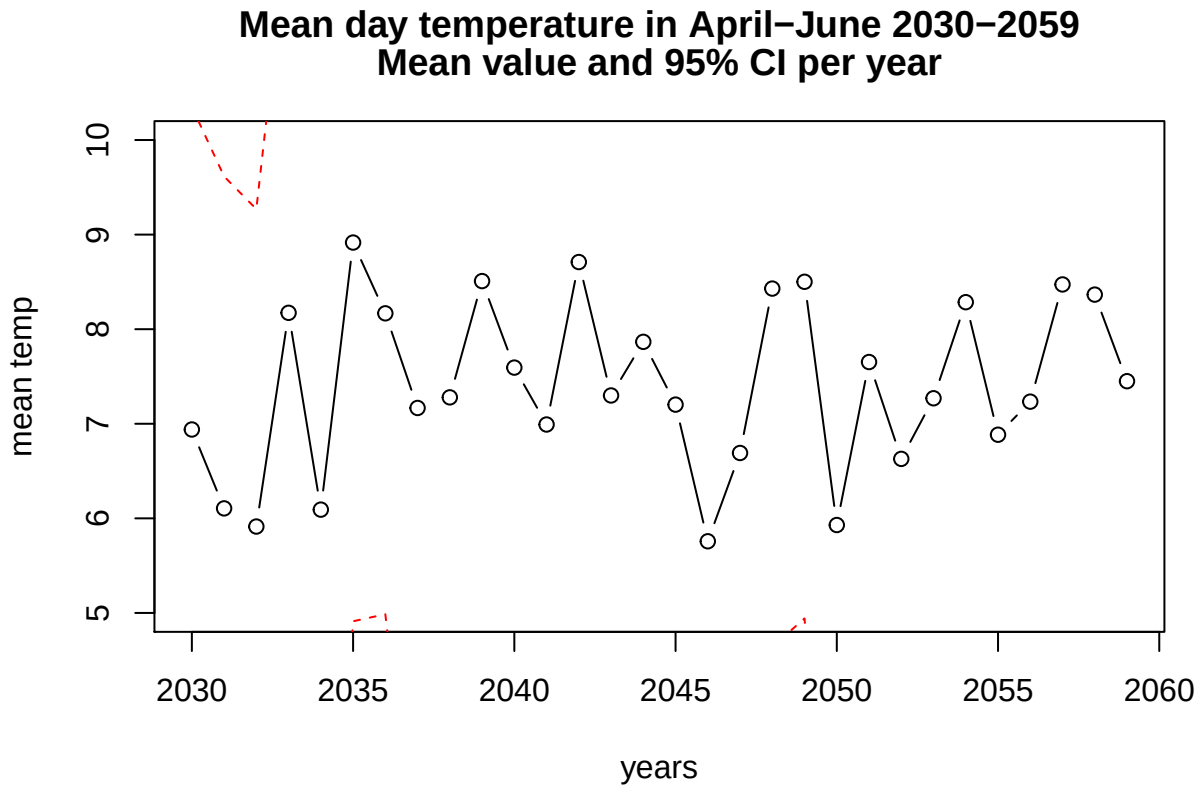
We consider now the data predicted for years 2030 - 2059.

```
### Future - temperature

ncfname.futB7 = ncfname.fut[seq(1, length(ncfname.fut), by = 18)]

temp = matrix(,nrow = length(ncfname.futB7), ncol = 91*36)  #April - June
## mean.temp
# 54 rows : years 2006-2059
# 91 columns : days April-June
mean.temp = matrix(,nrow = length(ncfname.futB7), ncol = length(91:181))
for(i in 1:length(ncfname.futB7)) {
  nc = open_nc(ncfname.futB7[i])
  if(length(nc$time_centered)==366) nc$votemper = nc$votemper[,-60] # we exclude the 29th of february f

  temp[i,] = as.vector(nc$votemper[,91:181])
  mean.temp[i,] = apply(nc$votemper[1:3,91:181], 2, mean)
}

# mean temperature in each day
par(mfrow= c(1,1))
# per day
#boxplot(mean.temp[25:54,],  main = c("Mean day temperature in April-June 2030-2059", "Distributions pe
day.mean.temp.fut = apply(mean.temp[25:54,], 2, mean)
plot(day.mean.temp.fut, main = c("Mean day temperature in April-June 2030-2059", "Mean value and 95% CI
```

```
day.q = apply(mean.temp[25:54,], 2, quantile)
lines(1:91, day.q[2,], col=2, lty=2)
lines(1:91, day.q[4,], col=2, lty=2)
```



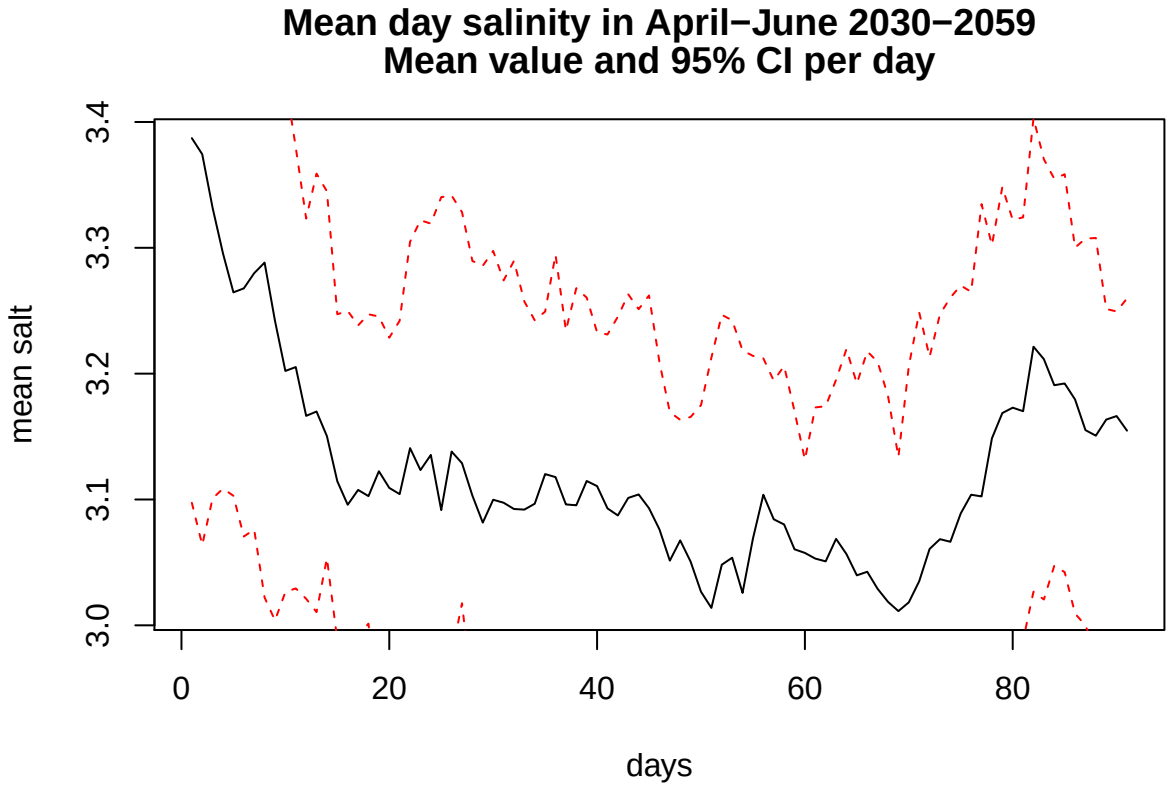

```
#per year
#boxplot(t(mean.temp[25:54,]),  main = c( "Mean day temperature in April-June 2030-2059", "Distribution
#axis(1, at = 2030:2059,  las=2,tick = T, labels = 2030:2059)
year.mean.temp.fut = apply(mean.temp, 1, mean)
plot(2030:2059, year.mean.temp.fut[25:54], main = c("Mean day temperature in April-June 2030-2059","Mea
year.q = apply(mean.temp, 1, quantile)
lines(2030:2059, year.q[2,25:54], col=2, lty=2)
lines(2030:2059, year.q[4,25:54], col=2, lty=2)
```



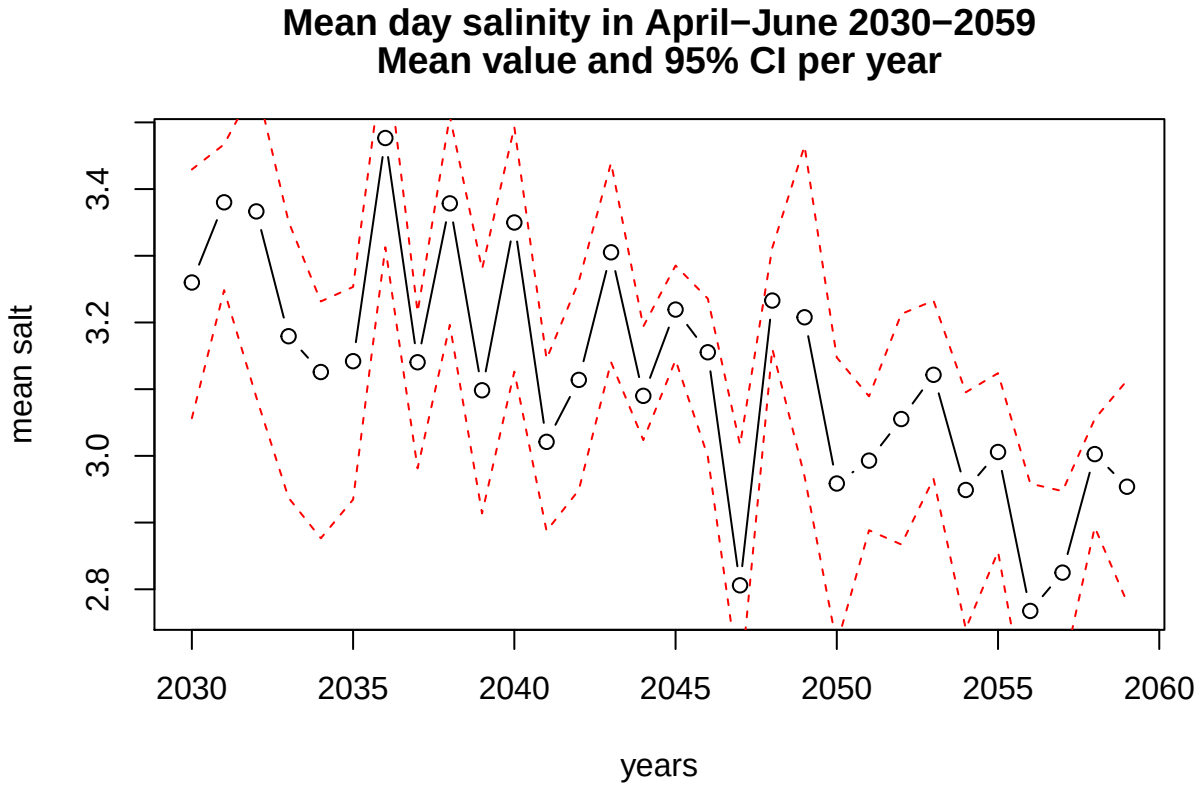
From the above plots we can observe how temperature increases with a linear trend from April to June, from 1 to 14 degrees. We can notice how the maximum temperature increased of 2 degrees with respect to years 1995-2005, while the minimum has increased of about 1 degree. The increasing rate is just a bit lower rate in the first month. We can't observe a clear trend of the daily temperature predictions through the years: the warmest value is around 9 degrees for 2035, while the maximum registered in the past was around 7 degrees. The coldest prediction is around 6 degrees for years 2046 and 2050.

```
### Future - salinity

ncfname.futB7 = ncfname.fut[seq(1, length(ncfname.fut), by = 18)]

salt = matrix(,nrow = length(ncfname.futB7), ncol = 91*36)  #April - June
## mean.salt
# 54 rows : years 2006-2059
# 91 columns : days April-June
mean.salt = matrix(,nrow = length(ncfname.futB7), ncol = length(91:181))
for(i in 1:length(ncfname.futB7)) {
  nc = open_nc(ncfname.futB7[i])
  if(length(nc$time_centered)==366) nc$vosaline = nc$vosaline[,-60] # we exclude the 29th of february f

  salt[i,] = as.vector(nc$vosaline[,91:181])
  mean.salt[i,] = apply(nc$vosaline[1:3,91:181], 2, mean)
}

# mean salinity in each day
par(mfrow= c(1,1))
# per day
#boxplot(mean.salt[25:54,],  main = c("Mean day salinity in April-June 2030-2059", "Distributions per d
day.mean.salt.fut = apply(mean.salt[25:54,], 2, mean)
```

```r
plot(day.mean.salt.fut, main = c("Mean day salinity in April-June 2030-2059", "Mean value and 95% CI pe
day.q = apply(mean.salt[25:54,], 2, quantile)
lines(1:91, day.q[2,], col=2, lty=2)
lines(1:91, day.q[4,], col=2, lty=2)
```



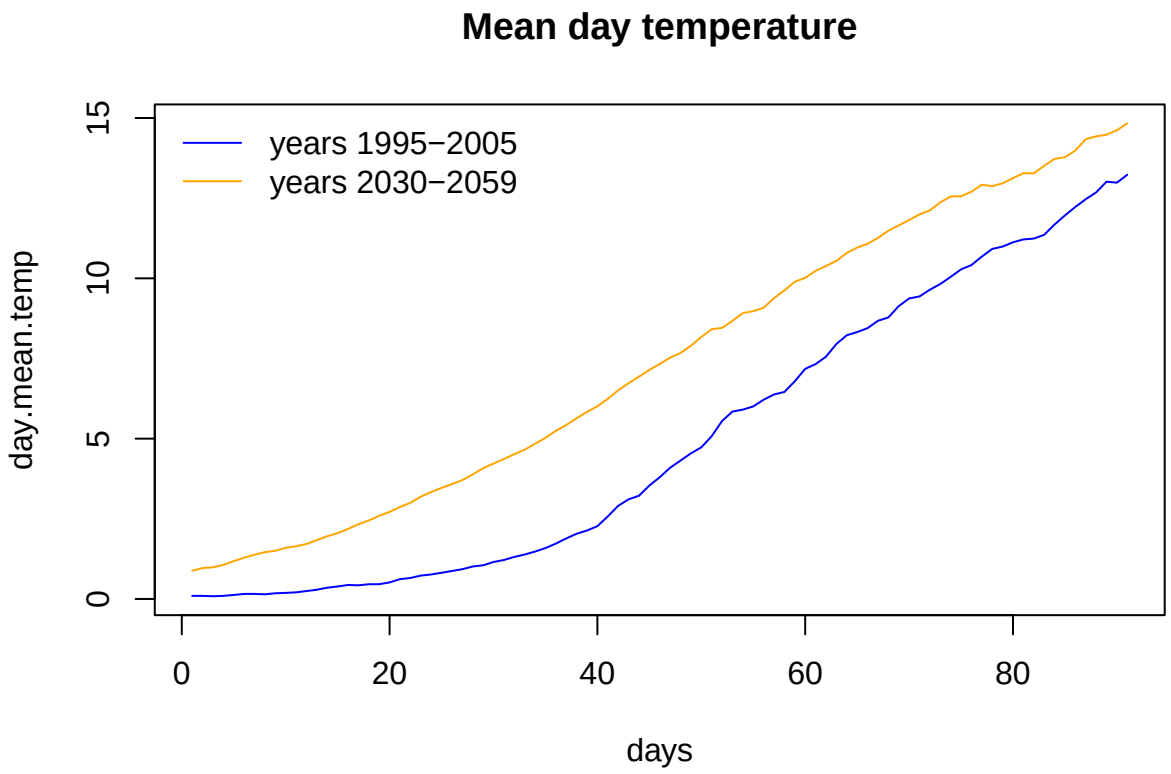

```r
#per year
#boxplot(t(mean.salt[25:54,]),  main = c( "Mean day salinity in April-June 2030-2059", "Distributions p
#axis(1, at = 2030:2059,  las=2,tick = T, labels = 2030:2059)
year.mean.salt.fut = apply(mean.salt, 1, mean)
plot(2030:2059, year.mean.salt.fut[25:54], main = c("Mean day salinity in April-June 2030-2059","Mean va
year.q = apply(mean.salt, 1, quantile)
lines(2030:2059, year.q[2,25:54], col=2, lty=2)
lines(2030:2059, year.q[4,25:54], col=2, lty=2)
```



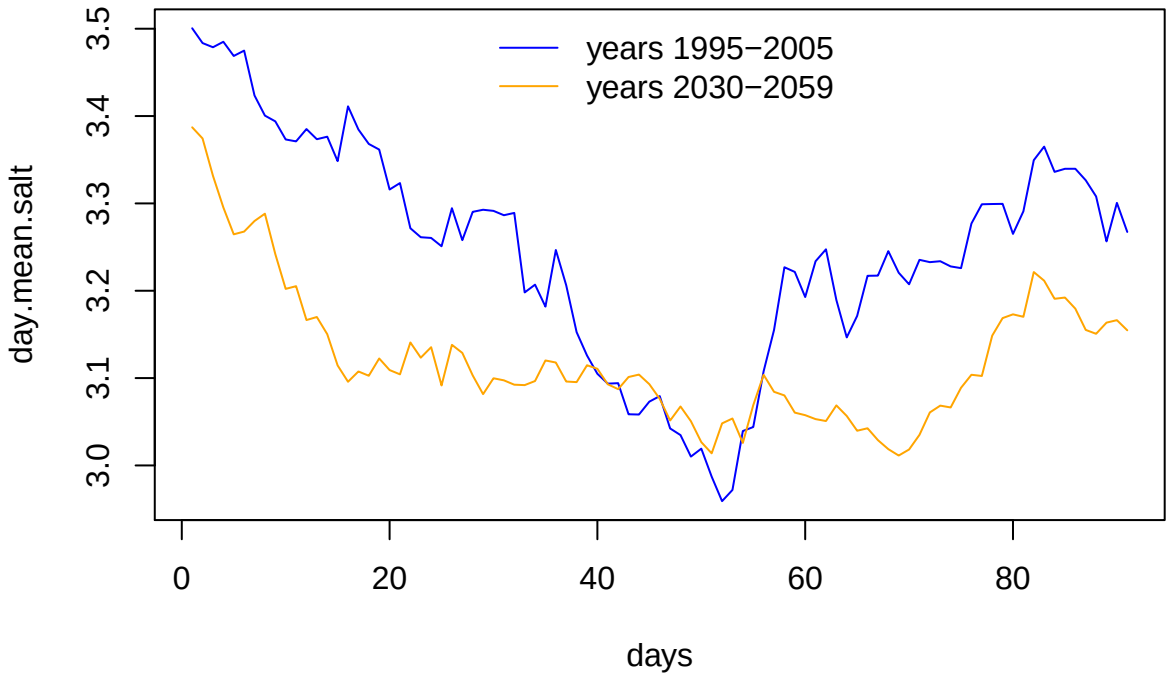
Again, if we consider daily salinity it remains constantly around 3, decreasing for the first months from 3.4 to a minimum of 3 in mid May, and then increases up to 3.2 in June. Year mean salinity instead varies between 2.8 in 2056 and 3.5 in 2030, a decreasing trend is visible from the plot.

Let's compare the predicted values with the measured ones for the mean daily temperature and salinity in April-June: we can observe how in the future, daily temperature increases of about 2 degrees on average, while daily salinity generally decreases of about 0.1.

```
plot(day.mean.temp, col="blue", main = "Mean day temperature", type="l", ylim=c(min(day.mean.temp), max
lines(1:91, day.mean.temp.fut, col= "orange")
legend("topleft", legend=c("years 1995-2005", "years 2030-2059"), col=c("blue", "orange"), lty=1, bty="
```




```
plot(day.mean.salt, col="blue", main = "Mean day salinity", type="l",xlab="days")
lines(1:91, day.mean.salt.fut, col= "orange")
legend("top", legend=c("years 1995-2005", "years 2030-2059"), col=c("blue", "orange"), lty=1, bty="n" )
```



Temperature as a covariate

We now focus on the temperature values (measured in year 2000 and predicted temperature in year 2058), retrieved from the GoB files. We want to extract this quantity in order to use it as a covariate in the hierarchical species distribution model we developed for white fishes.
To do so, we need to conform the latter variable to the one in 'whitefish.raster' data. Such variables cover raster cells of $300 \times 300 m^2$, while the temperature extracted from the NETcdf files have a wider scale, they cover a $1 \times 1 M^2$ nautical miles , that is equivalent to $1852 \times 1852 m^2$. Hence we need to upscale our raster data related to the temperature. We will consider the mean temperature evaluated in the 3 most superficial layers, saved in different levels of the raster file.

```
library(raster)
setwd("/home/piailari/Documents/thesis/SampleSmartSea/A005/")
gob <- "NORDIC-GOB_1m_20580101_20581231_grid_T.nc"

#extract April, May and June temperature
temp.apr1 = raster(gob, varname="votemper",  band =4, level =1) #0-3m depth
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.may1 = raster(gob, varname="votemper",  band =5, level =1)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.jun1 = raster(gob, varname="votemper",  band =6, level =1)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.apr2 = raster(gob, varname="votemper",  band =4, level =2) #3-6m depth
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.may2 = raster(gob, varname="votemper",  band =5, level =2)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.jun2 = raster(gob, varname="votemper",  band =6, level =2)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.apr3 = raster(gob, varname="votemper",  band =4, level =3) #6-9m depth
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

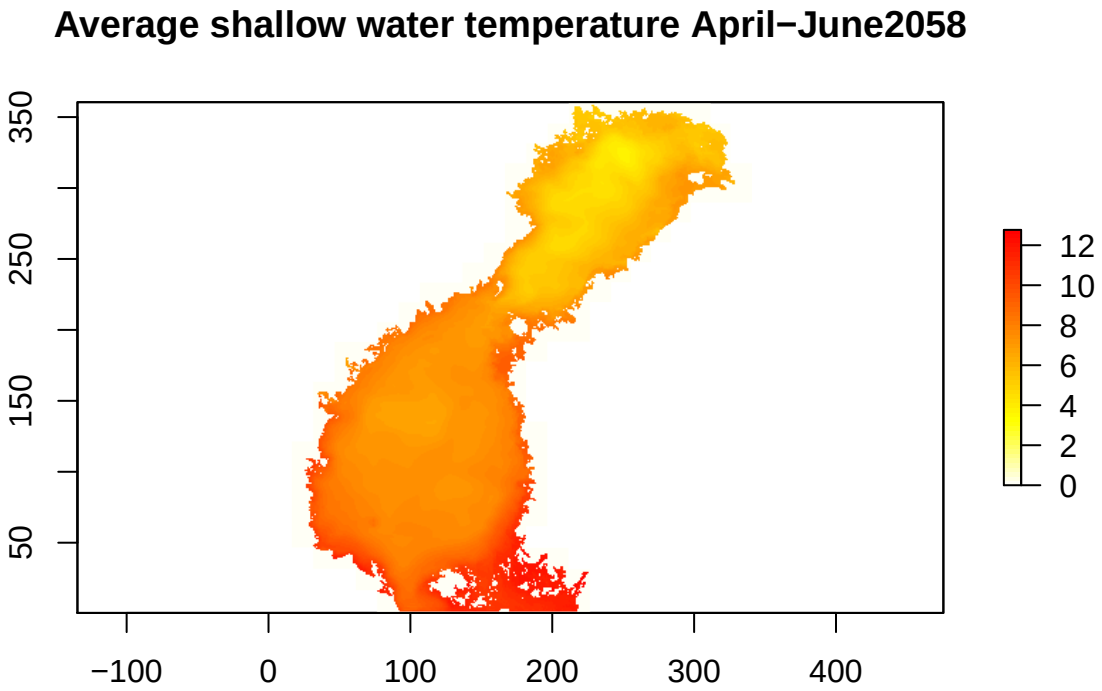
```
temp.may3 = raster(gob, varname="votemper",  band =5, level =3)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
temp.jun3 = raster(gob, varname="votemper",  band =6, level =3)
```

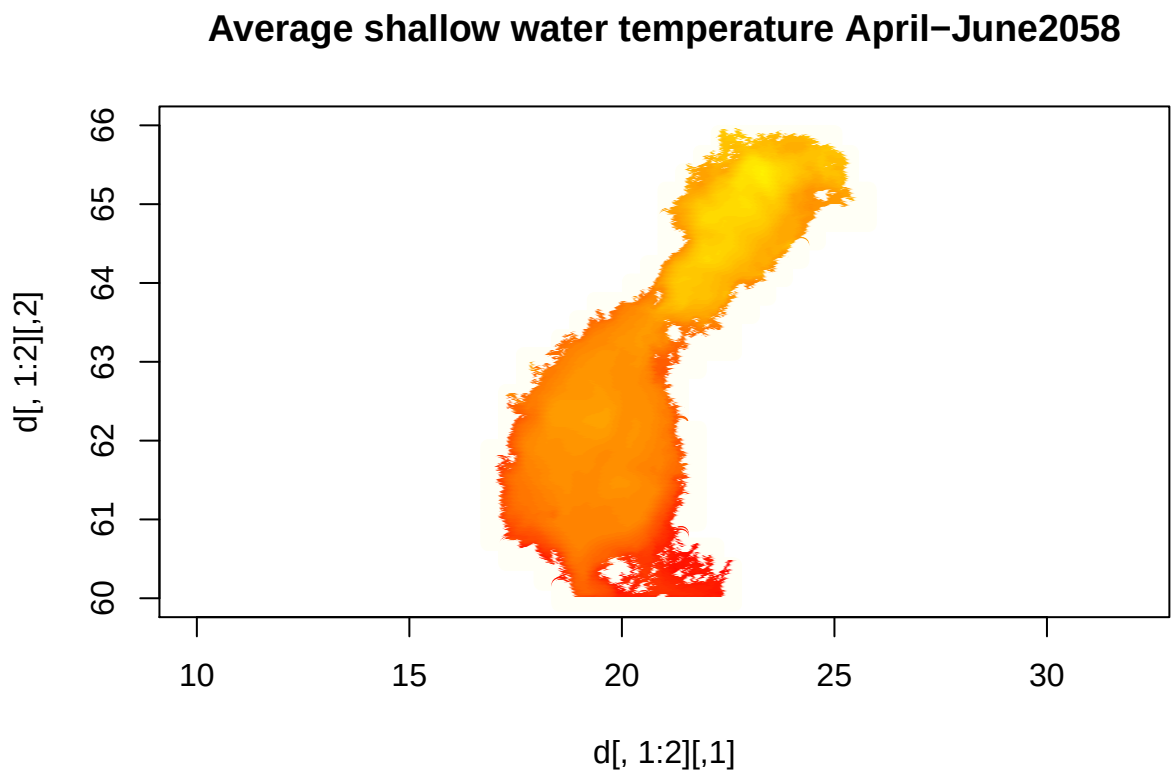
```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
# layer with everage temperature in Apr-Jun at depth 0-9m
mean.temp.layer = overlay(temp.apr1, temp.may1, temp.jun1, temp.apr2, temp.may2, temp.jun2, temp.apr3,
plot(mean.temp.layer, main="Average shallow water temperature April-June2058", col = rev(heat.colors(n))
```

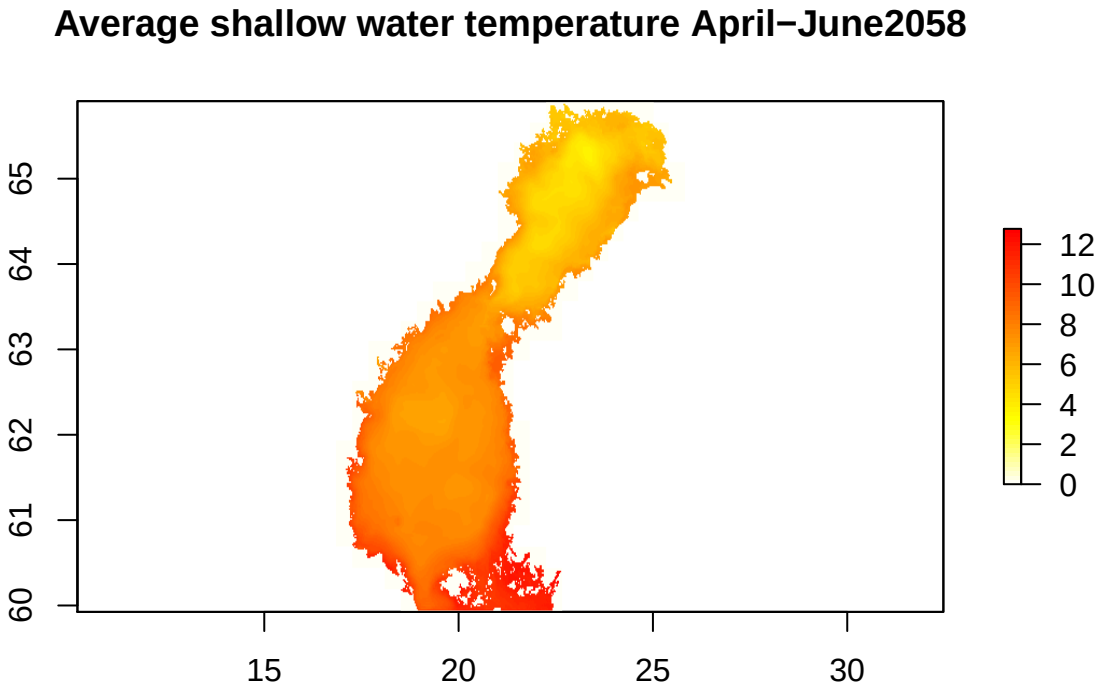


Again the coordinate system of our mean temperature layer are reduced to its indexes, we change them to latitude and longitude:

```
d =  cbind(values(lon), values(lat), values(mean.temp.layer))
d = na.exclude(d)
plot(d[,1:2], ylim=c(60,66), xlim=c(10,32),pch = 16, col = (heat.colors(n)[scl(d[,3]) * (n-1) + 1]), ma
```



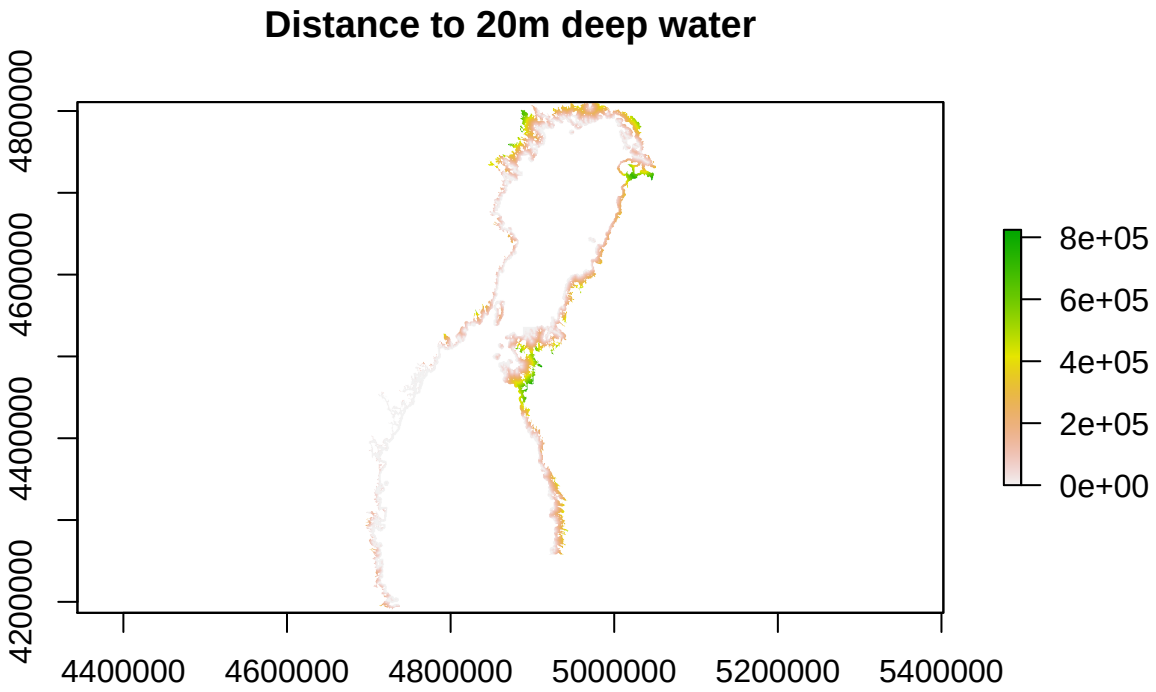

```
# from df to raster
e.temp = extent(cbind(d[,1],d[,2]))
r.temp = raster(e.temp, ncol=length(unique(d[,1])), nrow=length(unique(d[,2])))
mean.temp.r = rasterize(d[,1:2],r.temp, d[,3], fun=mean)

plot(mean.temp.r, main="Average shallow water temperature April-June2058",  col=rev(heat.colors(n)) )
```



If we plot one of the whitefish raster variables, we can observe that the coordinate reference system (CRS)

for the two files is different, let's display the ice coverage:

```
### whitefish raster data
setwd("/home/piailari/Documents/thesis")
whitefish.raster = read.table("predraster_whitefish.txt", header=TRUE, sep="\t")

e = extent(cbind(whitefish.raster$X,whitefish.raster$Y))
r = raster(e, ncol=length(unique(whitefish.raster$X)), nrow=length(unique(whitefish.raster$Y)))

# white fish covariate
z <- rasterize(cbind(whitefish.raster$X,whitefish.raster$Y), r, whitefish.raster$DIST20M, fun=mean)
plot(z, xlim=cbind(min(whitefish.raster$X),max(whitefish.raster$X)),
     ylim=cbind(min(whitefish.raster$Y),max(whitefish.raster$Y)), main="Distance to 20m deep water")
```



A planar CRS is defined by a projection, datum, and a set of parameters. The parameters determine things like where the center of the map is. The number of parameters depends on the projection. We know the coordinate system in the whitefish data is ETRS89-extended / LAEA Europe, that is a UTM (easting-northing), with datum European Terrestrial Reference System 1989 (https://en.wikipedia.org/wiki/European_Terrestrial_Reference_System_1989) while the Gob data has latitude and longitude as coordinates, so it has a geographic coordinate system: WGS84, based on the World Geodetic System 1984 datum. We need to recover the corresponding PROJ.4 notation for ETRS89, assign it to the respective raster file and then project the Gob raster.

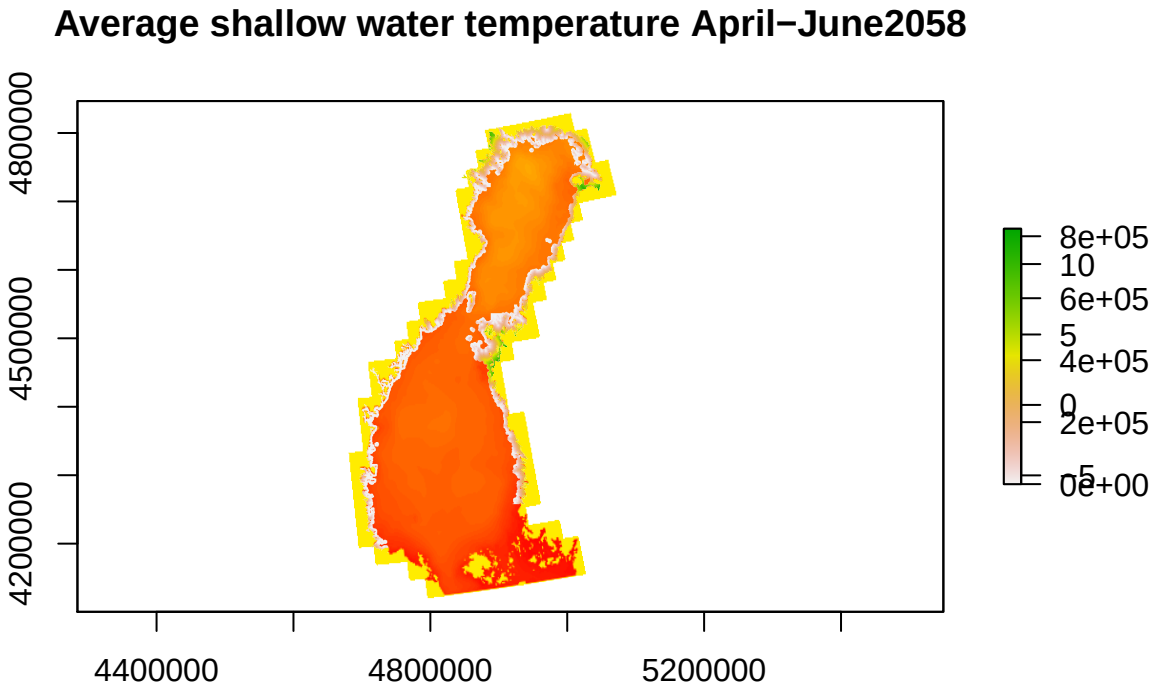
We can also specify the EPSG code, that in our case is EPSG:3035 for the whitefish raster and 7663 for temperature raster (http://epsg.io/4326).

```
#https://mgimond.github.io/Spatial/coordinate-systems-in-r.html
library(rgdal)
```

```
## rgdal: version: 1.4-8, (SVN revision 845)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: /usr/share/gdal/2.2
##  GDAL binary built with GEOS: TRUE
```

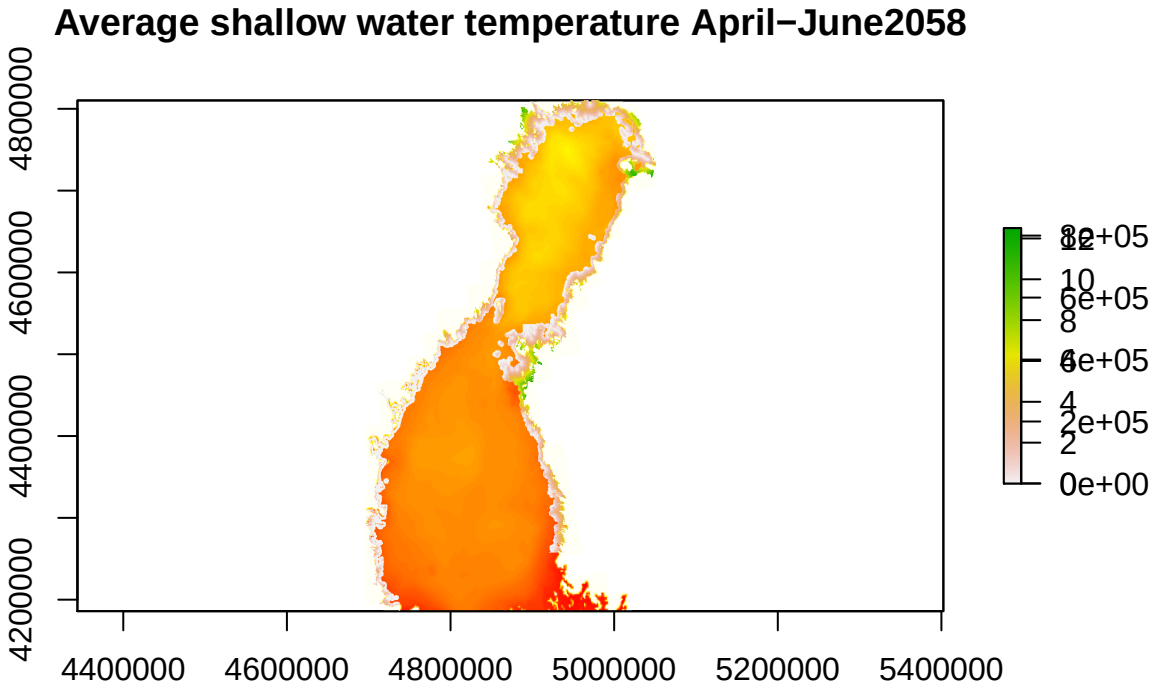
```
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: (autodetected)
##  Linking to sp version: 1.4-1
```

```r
# set original crs
crs(r) = CRS("+init=epsg:3035") #https://epsg.io/3035    +proj=utm +zone=35 +ellps=GRS80 +towgs84=0,0,0
crs(mean.temp.r) = CRS("+proj=longlat +ellps=WGS84") # lat lon, geo.cs : CRS("+init=epsg:4326")

# project
mean.temp.pr <- projectRaster(mean.temp.r, crs=crs(r))
plot(mean.temp.pr, main="Average shallow water temperature April-June2058", col= rev(heat.colors(56)))
plot(z, add=T)
```



From the plot we can see how the temperature layer is actually bigger, we need to crop it.

```r
mean.temp.pr = crop(mean.temp.pr, e)
plot(mean.temp.pr, main="Average shallow water temperature April-June2058", col= rev(heat.colors(56)))
plot(z, add=T)
```



Now we need to upgrade the resolution of the raster layer. The R function `disaggregate`, in raster package disaggregate a RasterLayer to create a new RasterLayer with a higher resolution (smaller cells). The values in the new RasterLayer are the same as in the larger original cells. By specifying method="bilinear" values are locally interpolated.
If we use a disaggregate factor equal to $\frac{res_{temp}}{res_{wf.layer}}$ we notice that the final resolution of the temperature layer is not exactly equal to the one of the whitefish raster, this is because the factor is automatically rounded to an integer. We will use another function: `resample` , that transfers values between non matching Raster objects, using bilinear interpolation.

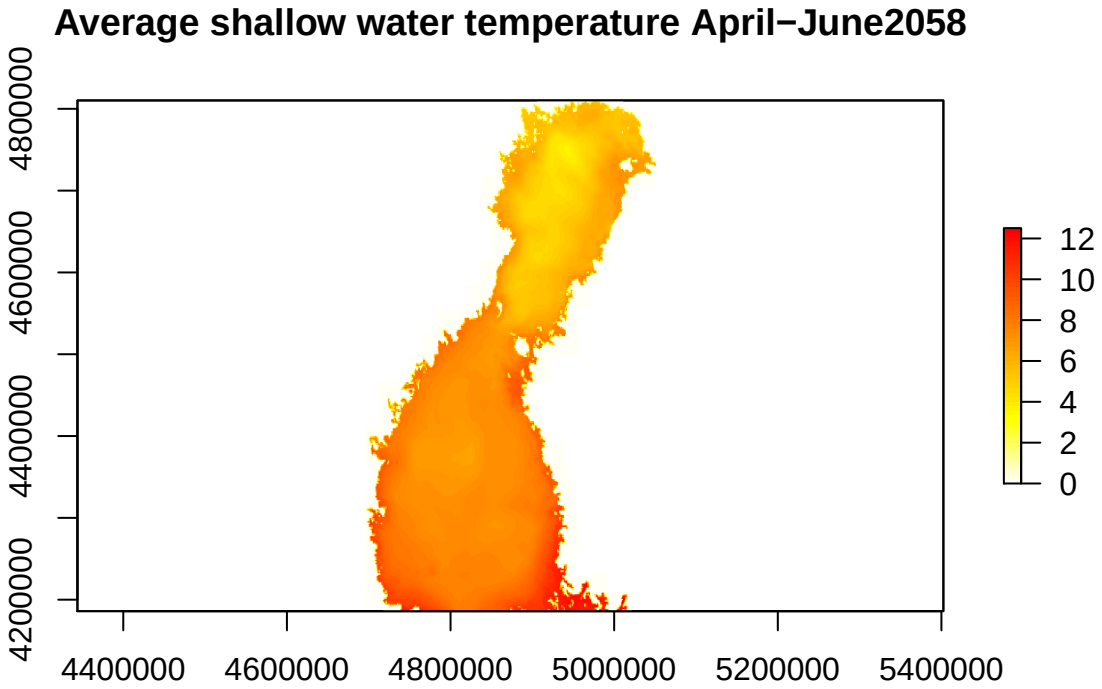
```
# Look at the resolution
mean.temp.pr
```

```
## class      : RasterLayer
## dimensions : 343, 255, 87465  (nrow, ncol, ncell)
## resolution : 1390, 1820  (x, y)
## extent     : 4695959, 5050409, 4185962, 4810222  (xmin, xmax, ymin, ymax)
## crs        : +init=epsg:3035 +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +
## source     : memory
## names      : layer
## values     : 0, 12.50618  (min, max)
```

```
r
```

```
## class      : RasterLayer
## dimensions : 2078, 1183, 2458274  (nrow, ncol, ncell)
## resolution : 299.7464, 300.4331  (x, y)
## extent     : 4695791, 5050391, 4186575, 4810875  (xmin, xmax, ymin, ymax)
## crs        : +init=epsg:3035 +proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +
```

```
# upscale
mean.temp.upsc = disaggregate(mean.temp.pr, fact = res(mean.temp.pr)/res(r))
plot(mean.temp.upsc,  main="Average shallow water temperature April-June2058", col= rev(heat.colors(56))
```




```
res(mean.temp.upsc)    #  = 278.0000, 303.3333
```

```
## [1] 278.0000 303.3333
```

```
mean.temp.upsc=resample(mean.temp.upsc,r, method="bilinear") #with categorical variable use ngb for nea
res(mean.temp.upsc)    #  = 299.7464, 300.4331
```

```
## [1] 299.7464 300.4331
```

We have the converted raster for our variable of interest, finally we recover the dataset containing the projected coordinates and the corresponding temperature vale, so that it can be merged with the covariates from whitefish.dat file , and try to merge them.

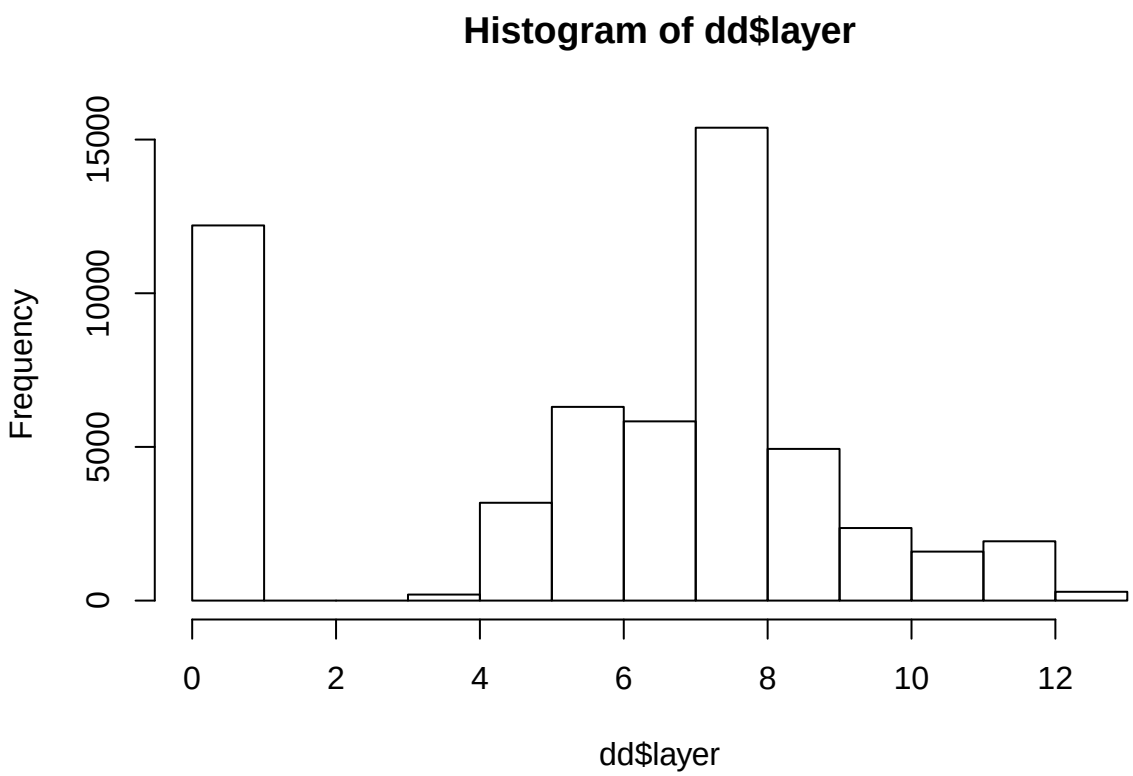
```
whitefish.dat.new=as.data.frame(mean.temp.upsc, xy=T)

# extract white fish covariates of interest
# BOTTOMCLS - Bottom type classification, a categorical variable with classes 0:5
# DIS_SAND  - distance to sandy shore, continuous variable
# FE300ME   - The average fetch (openness/exposure) over all directions, continuous variable
# ICELAST09 - The last ice cover date in winter 2009-10, continuous variable
# RIVERS    - Influence of rivers (~weighted average distance to river mouths), continuous variable
# SAL910WIN - Winter salinity in 2009-2010, continuous variable
# DIST20M - distance to deep
# CHL_A
names = c("FE300W" , "BOTTOMCLS", "DIS_SAND", "FE300ME", "ICELAST09", "RIVERS", "SAL910WIN", "DIST20M",
for (var in names) {
  z = rasterize(cbind(whitefish.raster$X,whitefish.raster$Y), r, whitefish.raster[,var], fun=mean)
  data1= as.data.frame(z, xy=T)
  # merge
  whitefish.dat.new = merge.data.frame(whitefish.dat.new, data1, by= c("x", "y" ))
}

colnames(whitefish.dat.new) = c("E_etrs89", "N_etrs89", "TEMP", names)
# remove NA
```

```
whitefish.dat.new = whitefish.dat.new[complete.cases(whitefish.dat.new ),]

head(whitefish.dat.new)
```

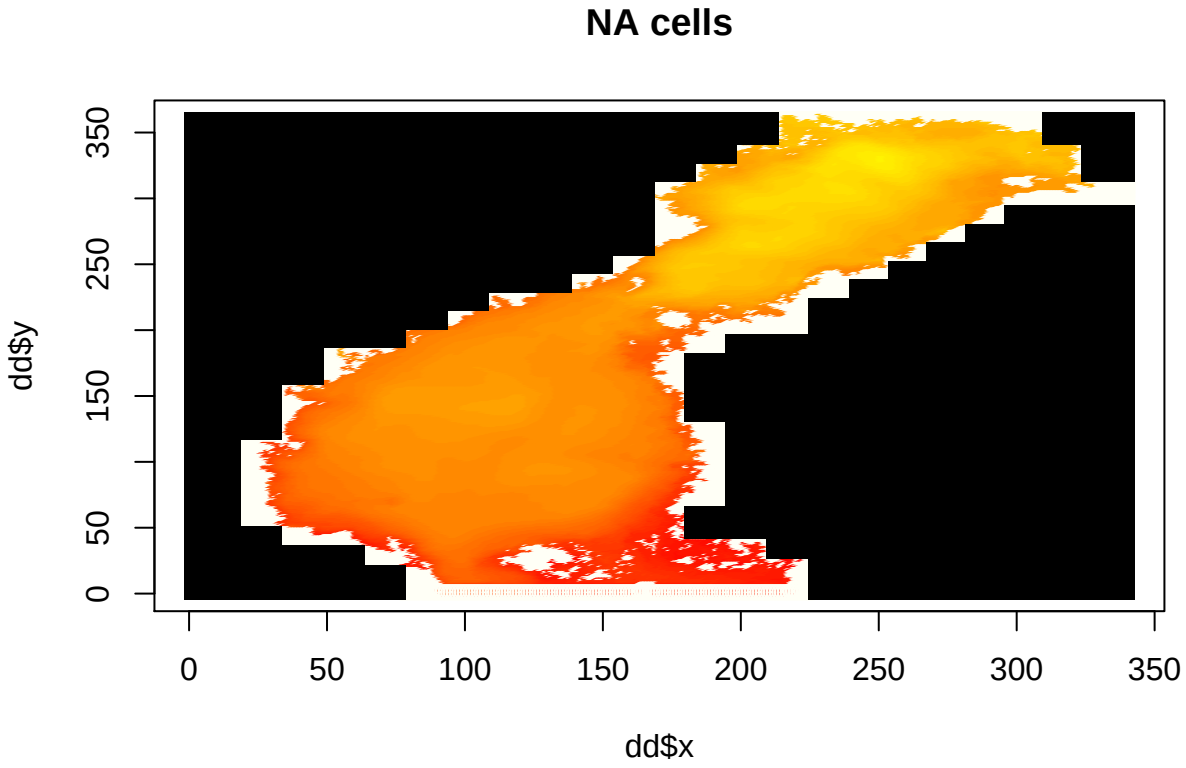
```
##      E_etrs89 N_etrs89 TEMP FE300W BOTTOMCLS DIS_SAND FE300ME ICELAST09 RIVERS
## 2434  4696241  4293379    0    289         2       49    7828         15  11790
## 2435  4696241  4293679    0    289         2       49    7828         15  11790
## 2436  4696241  4293980    0    289         4       49    7828         15  11728
## 4512  4696540  4293379    0    859         2       49     234         15  11790
## 4513  4696540  4293679    0    859         2       49     234         15  11790
## 4514  4696540  4293980    0   5725         5       49    7819         15  11728
##      SAL910WIN DIST20M CHL_A
## 2434         5  243338    40
## 2435         5  240852    40
## 2436         5  243338    40
## 4512         5  237338    40
## 4513         5  231852    40
## 4514         5  243338    40
```

Let's have a look at our mean temperature variable, we consider it before the coordinate and scaling transformations.

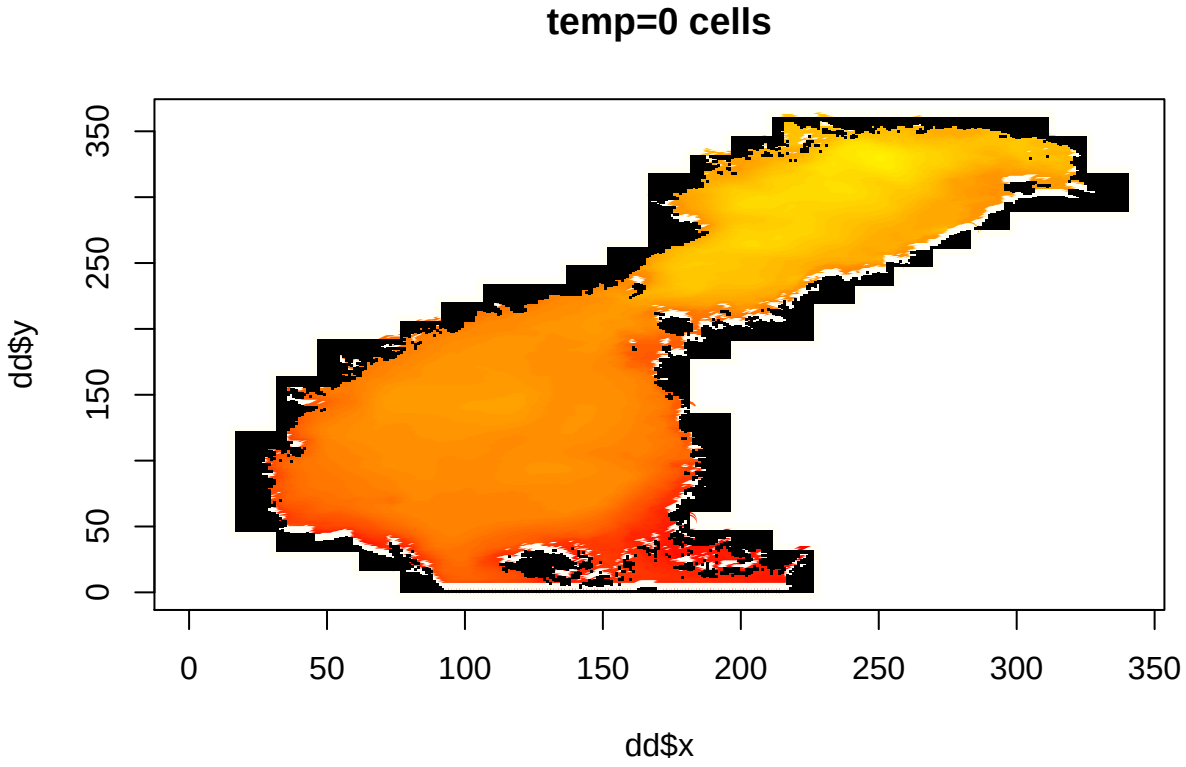
```
dd=as.data.frame(mean.temp.layer, xy=T, na.rm=F)
hist(dd$layer)
```



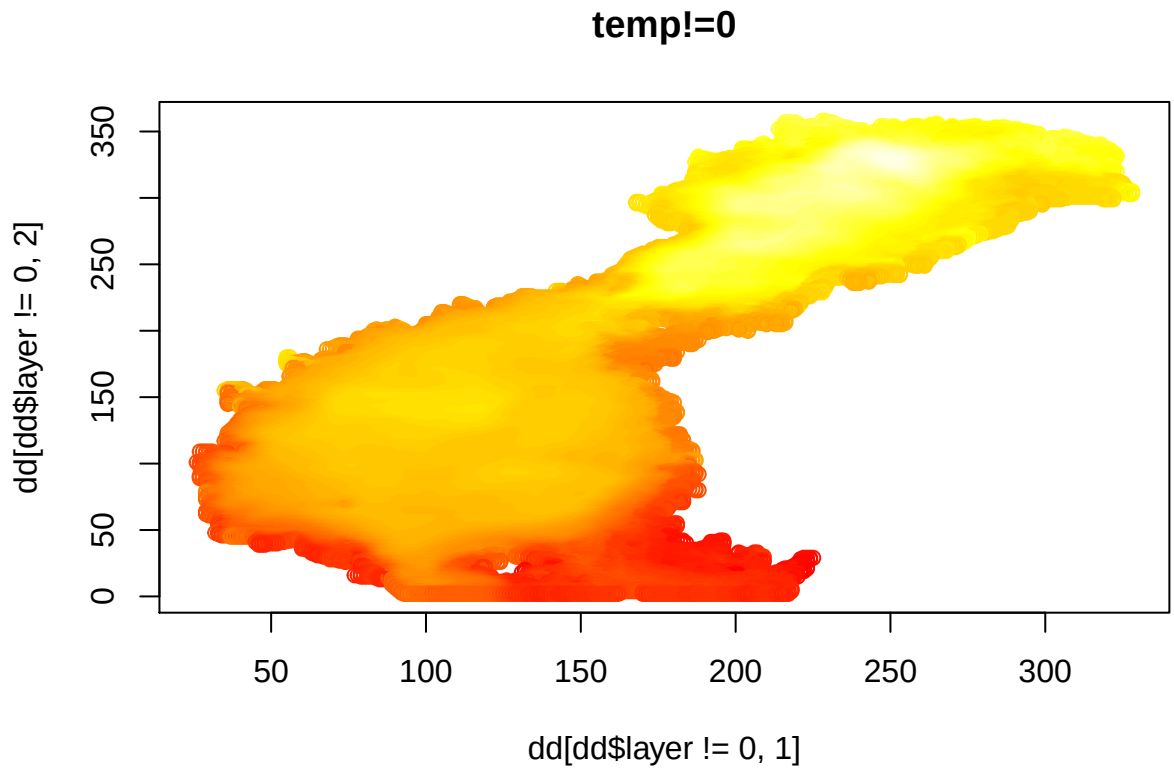

```
# NA cells
plot(dd$x,dd$y, col=(heat.colors(n)[scl(dd[,3]) * (n-1) + 1]), main ="NA cells")
points(dd[is.na(dd$layer),1], dd[is.na(dd$layer),2], pch=15, cex=1)
```




```
# temp=0 cells
plot(dd$x,dd$y, col=(heat.colors(n)[scl(dd[,3]) * (n-1) + 1]), main ="temp=0 cells")
points(dd[dd$layer==0,1], dd[dd$layer==0,2], pch=15, cex=0.2 )
```

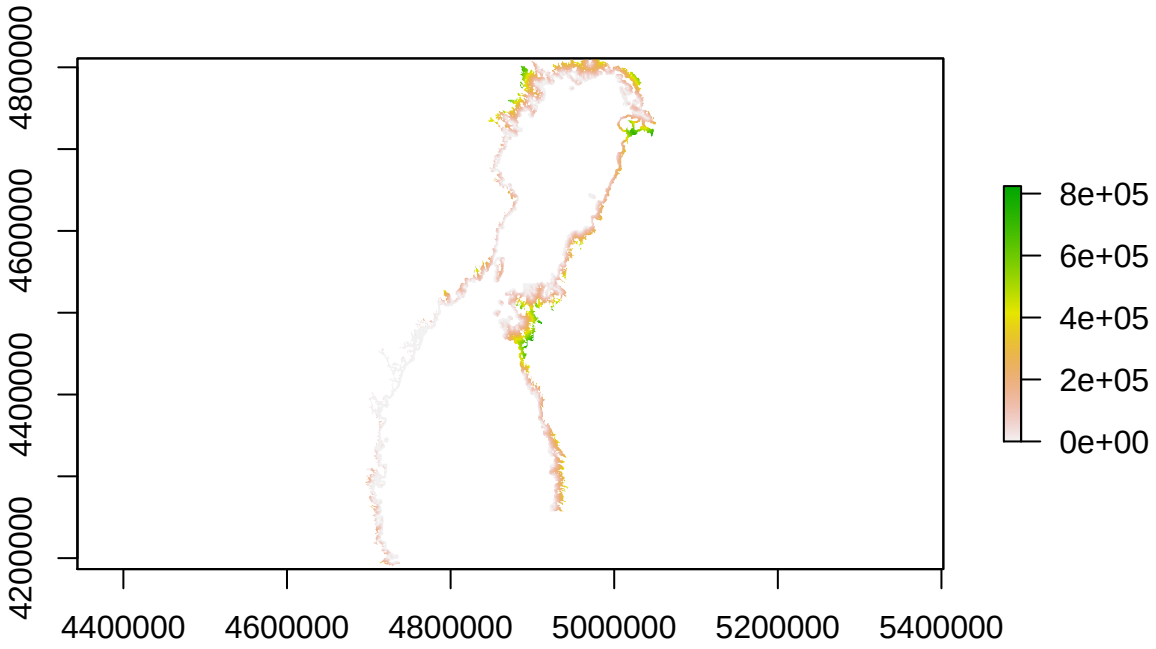


```
plot(dd[dd$layer!=0,1], dd[dd$layer!=0,2], col=(heat.colors(n)[scl(dd[dd$layer!=0,3]) * (n-1) + 1]) , ma
```

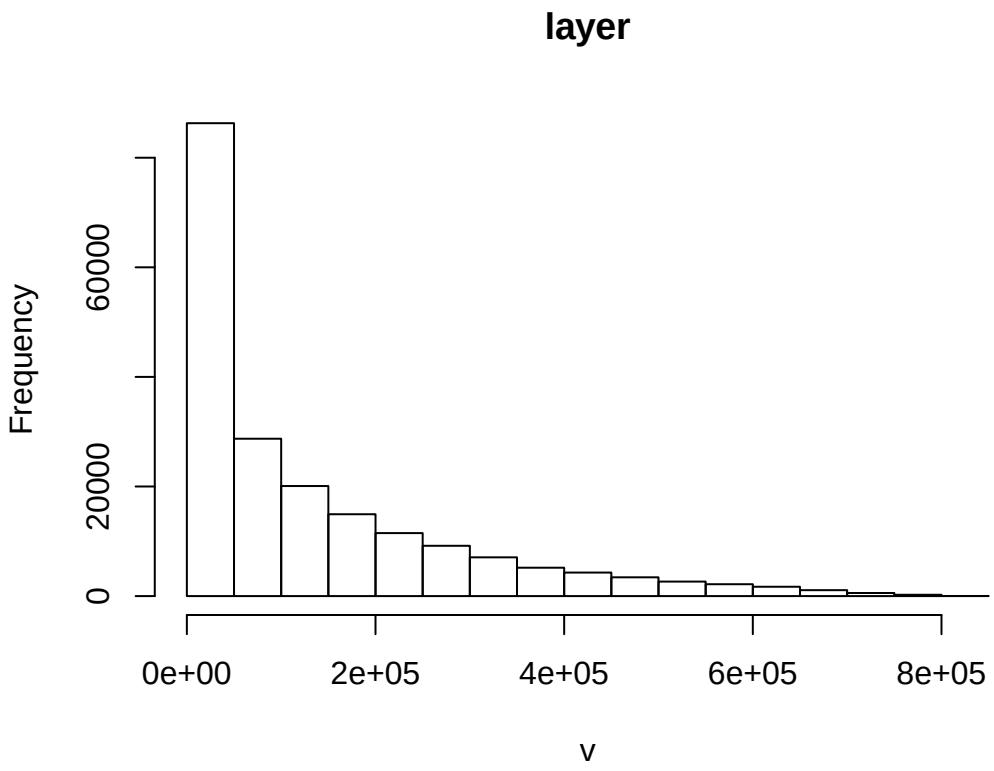


The variables from whitefish raster file have no land values, but a lot of NA, we will discard.

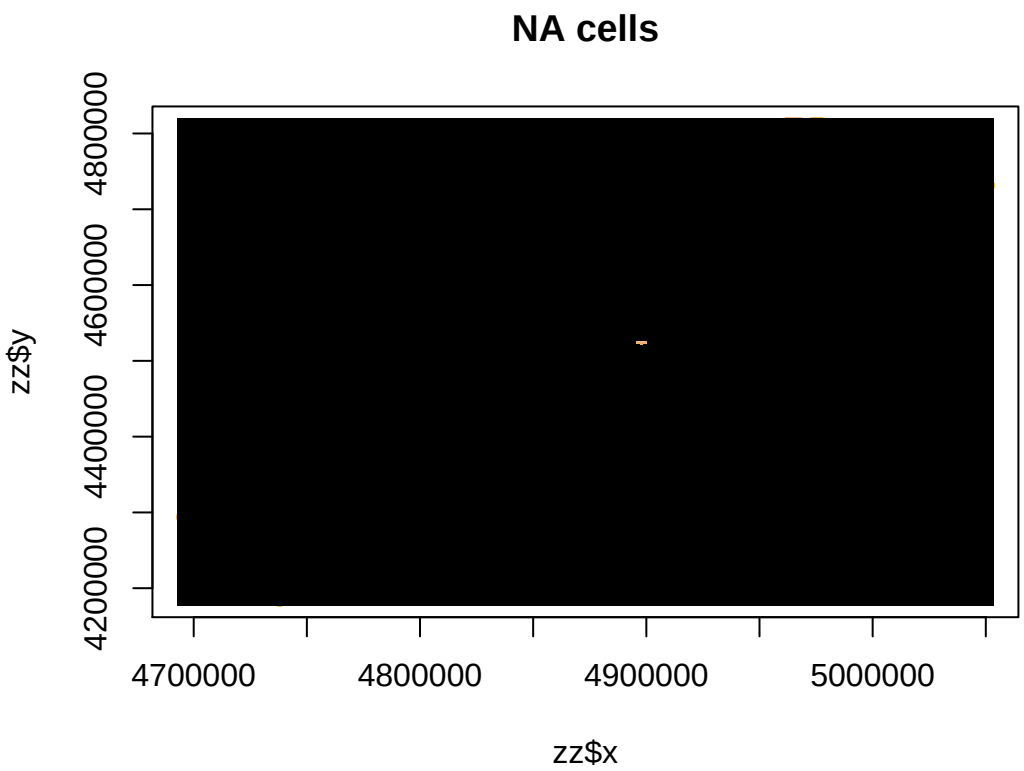
```
z <- rasterize(cbind(whitefish.raster$X,whitefish.raster$Y), r, whitefish.raster$DIST20M, fun=mean)
plot(z)
```



```r
hist(z)
```




```r
zz=as.data.frame(z,xy=T)
# NA cells
plot(zz$x,zz$y, col=(terrain.colors(n)[scl(zz[,3]) * (n-1) + 1]), main ="NA cells")
points(zz[is.na(zz$layer),1], zz[is.na(zz$layer),2], pch=15, cex=1)
```



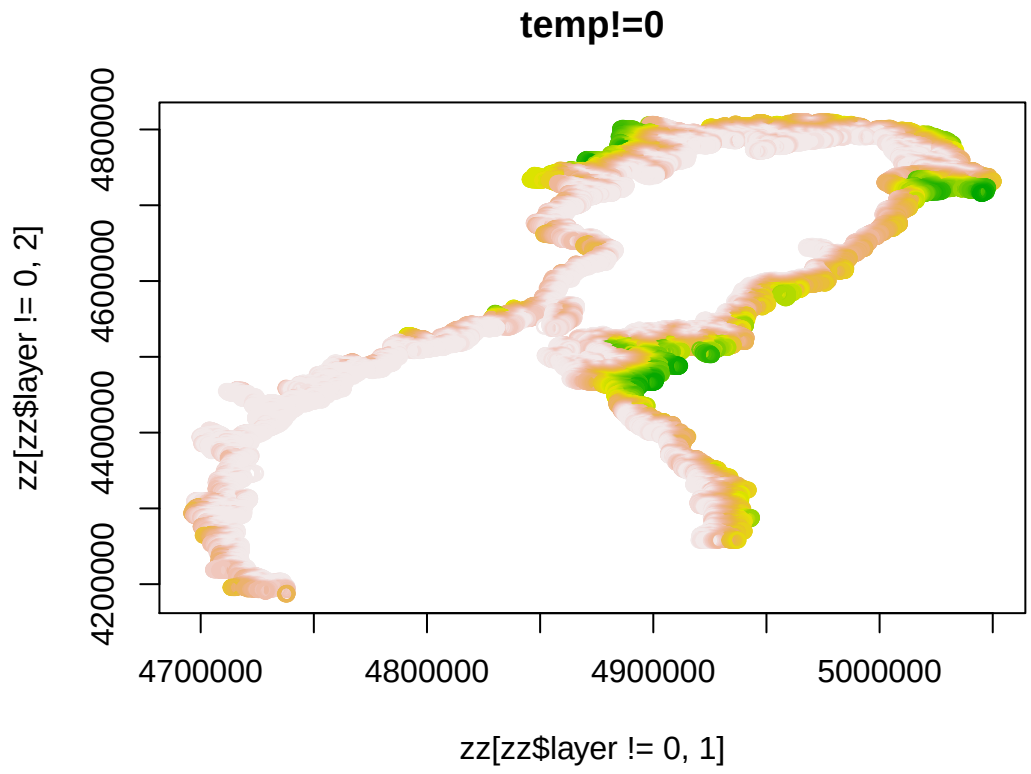

```
length(which(is.na(zz$layer))) # 2260000
```

```
## [1] 2259170
```

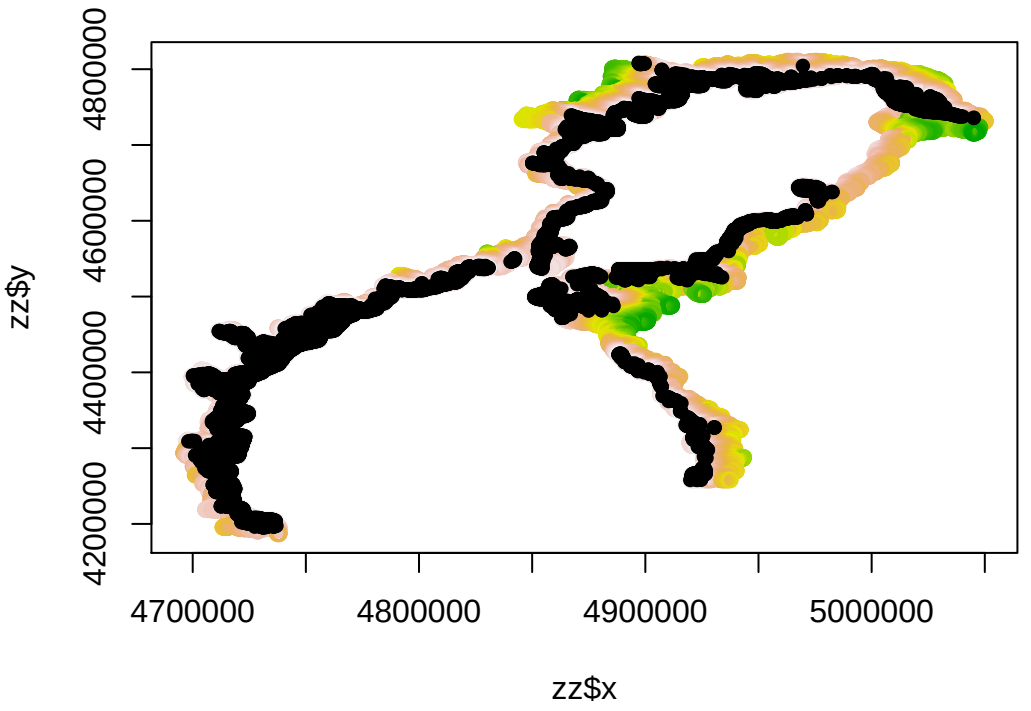
```
length(which(zz$layer==0)) # 32000
```

```
## [1] 31932
```

```
plot(zz[zz$layer!=0,1], zz[zz$layer!=0,2], col=(terrain.colors(n)[scl(zz[zz$layer!=0,3]) * (n-1) + 1])
```

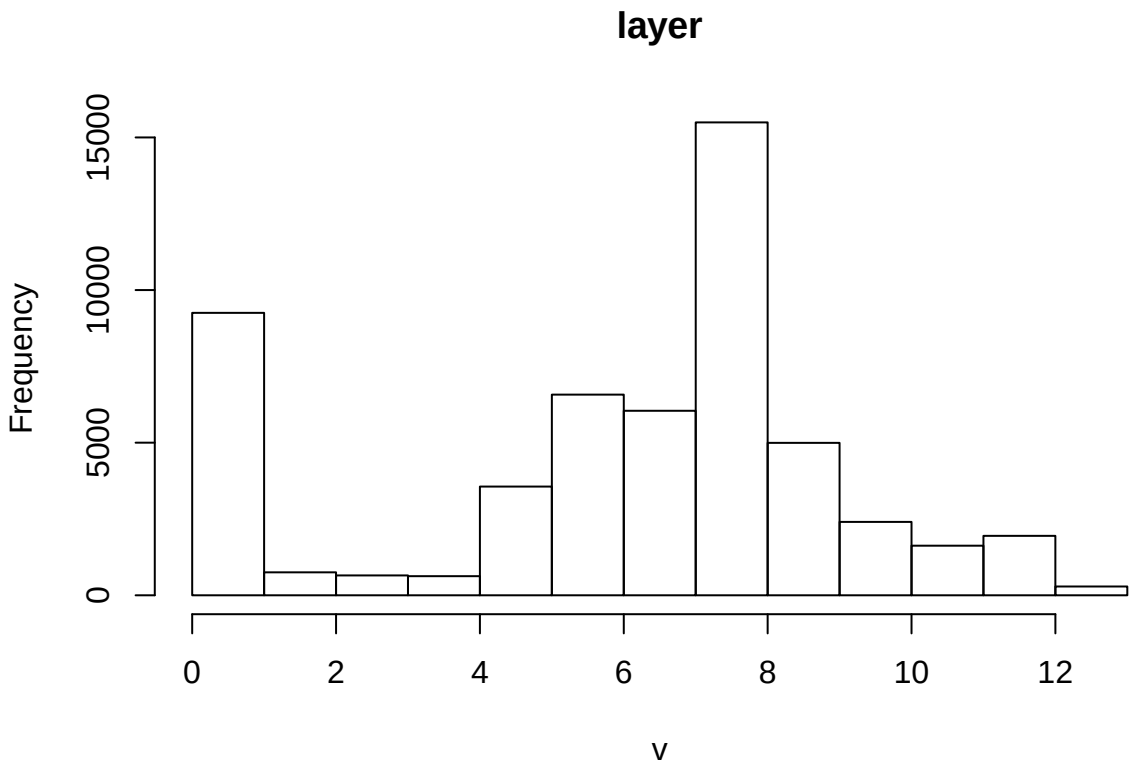



```
plot(zz$x,zz$y, col=(terrain.colors(n)[scl(zz[,3]) * (n-1) + 1]), main ="layer =0 cells")
points(zz[zz$layer==0,1], zz[zz$layer==0,2], col=1, pch=16)
```

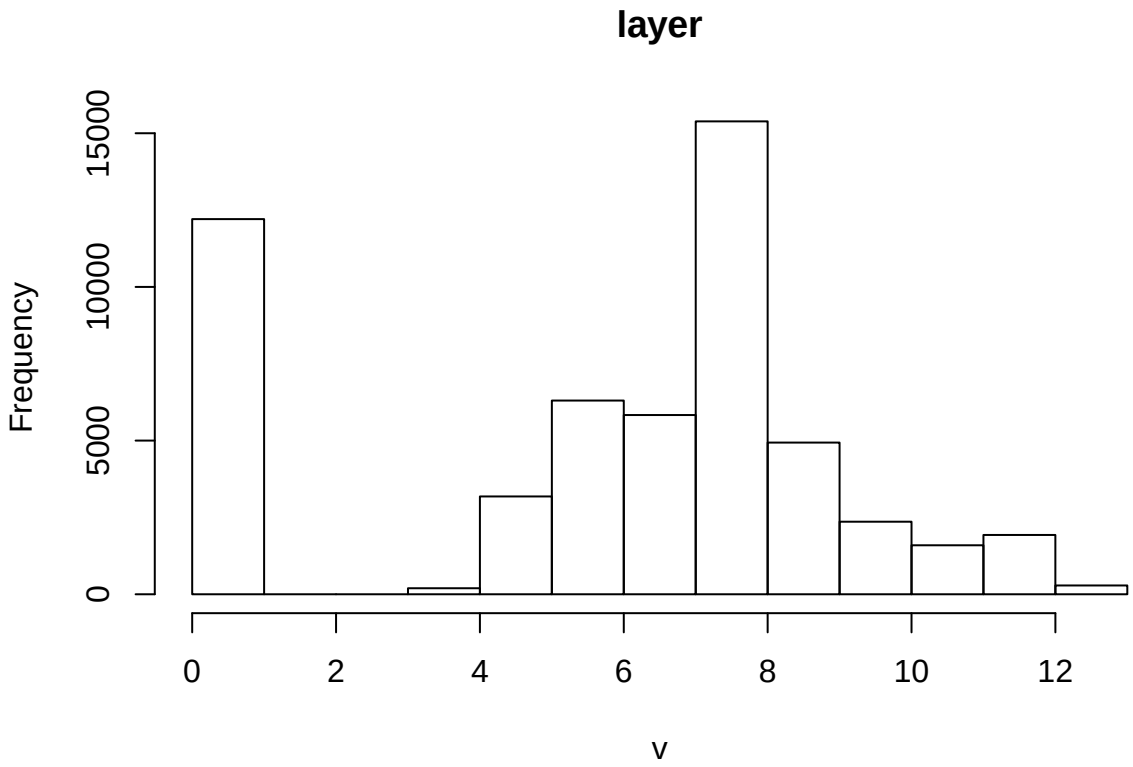


We substitute each 0 temperature values in mean.temp with the average values of their 8 closest cells. We still get some 0 zero cells, as they are probably surrounded by other 0 cells.

```r
#subsitute 0 cells with mean values of surrounding cells
fill.0 <- function(x, i=5) {
  if(is.na(x[i]) | x[i] !=0 )  return(x[i])
  else    return( mean(x[-i], na.rm=T ))

}

mean.temp.layer2 <- focal(mean.temp.layer, w = matrix(1,3,3), fun = fill.0,pad=T)
hist(mean.temp.layer2 )
```




```r
hist(mean.temp.layer)
```



We want to get rid of all zero cells. We try by removing all land cells from the data, then we will look at the closest raster cell for each data-point in the whitefish.dat file.

```
# remove land cells
mean.temp.r[mean.temp.r==0]=NA
```

Let's create a function that automatize the steps above:

```
merge_raster.variable_df0 = function(file, var, band, level, final.raster , var.names =c(), lon.name="l
  "
  This function takes as arguments
  file : string, path of raster file we want to extract a layer from   (EX Smartsea Gob file )
  var : name of the raster.layer variable we want to extract from file  (EX temperature)
  band :  number or vector, bands to consider (averaged)
  level : number or vector, levels to consider (averaged)
  final.raster : dataframe from which we extra    ct raster file we want to conform our layer with
  var.names : variables names of final.raster file we want to include in the output dataframe (optional
  lon.name : name of the longitude variable from file, default lon
  lat.name : name of the latitude variable from file, default lat
  lon.final : name of longitude variable from final.raster, default X
  lat.final : name of latitude variable from final.raster, default Y

  and returns a list with 2 objects
  data.frame : df with  longitude, latitude (in the same crs and resolution of r.final), the  variable
  raster.layer : raster layer of the variable of interest in the same crs and resolution of r.final
  "

  library(raster)
  library(rgdal)

  # layer with average band and level var
```

```
rast0 = raster(file, varname=var,  band = band[1], level = level[1]) # raster with proper extension
mean.layer = overlay(-rast0, rast0, fun ="sum") # create a 0 layer to start with
for (f in file) {
  for (i in band) {
  for (j in level) {
    rast = raster(f, varname=var,  band = i, level = j)
    mean.layer = overlay(rast, mean.layer, fun =sum)
  }
}
}
mean.layer = mean.layer/(length(band)*length(level)*length(file))

# remove land cells
if(remove.land) mean.layer[mean.layer==0]=NA
else {
  # subsitute 0 cells with mean values of surrounding cells
  fill.0 = function(x, i=5) {
    if(is.na(x[i]) | x[i] !=0 )  return(x[i])
    else    return( mean(x[-i], na.rm=T ))
  }
 mean.layer = focal(mean.layer, w = matrix(1,3,3), fun = fill.0,pad=T)
}

# raster file
lon = raster(file, varname=lon.name)
lat = raster(file, varname=lat.name)
d =  cbind(values(lon), values(lat), values(mean.layer))
d = na.exclude(d)
# from df to raster
e = extent(cbind(d[,1],d[,2]))
r = raster(e, ncol=length(unique(d[,1])), nrow=length(unique(d[,2])))
mean.r = rasterize(d[,1:2],r, d[,3], fun=mean)

# extrat quantities from final raster file
e.final = extent(cbind(final.raster[, lon.final],final.raster[, lat.final]))
r.final = raster(e.final, ncol=length(unique(final.raster[, lon.final])), nrow=length(unique(final.ras
res.final = res(r.final)
crs.final = crs(r.final)
if(is.na(crs.final)) crs.final=CRS("+init=epsg:3035")

# set original crs
if(is.na(crs(mean.r))) crs(mean.r) = CRS("+proj=longlat +ellps=WGS84") # lat lon, geo.cs : CRS("+init
# project
mean.pr = projectRaster(mean.r, crs=crs.final)

# crop
mean.pr = crop(mean.pr, e.final)

# upscale
mean.upsc = disaggregate(mean.pr, fact = res(mean.pr)/res.final)
mean.upsc = resample(mean.upsc,r.final, method="bilinear")

# merge the two data as a new data frame
```

```
  dat.new=as.data.frame(mean.upsc, xy=T)
  for (variable in var.names) {
    z = rasterize(cbind(final.raster[, lon.final], final.raster[,lat.final]), r.final, final.raster[,va
    data1= as.data.frame(z, xy=T)
    # merge : only the obs from final.raster variables are keeped
    dat.new = merge.data.frame(dat.new, data1, by= c("x", "y" ), all.x = F, all.y = T)
  }

  colnames(dat.new) = c("E_etrs89", "N_etrs89", var, var.names)
  # remove NA
  dat.new = dat.new[complete.cases(dat.new ),]

  return(list("data.frame" = dat.new, "raster.layer" = mean.upsc))
}
```

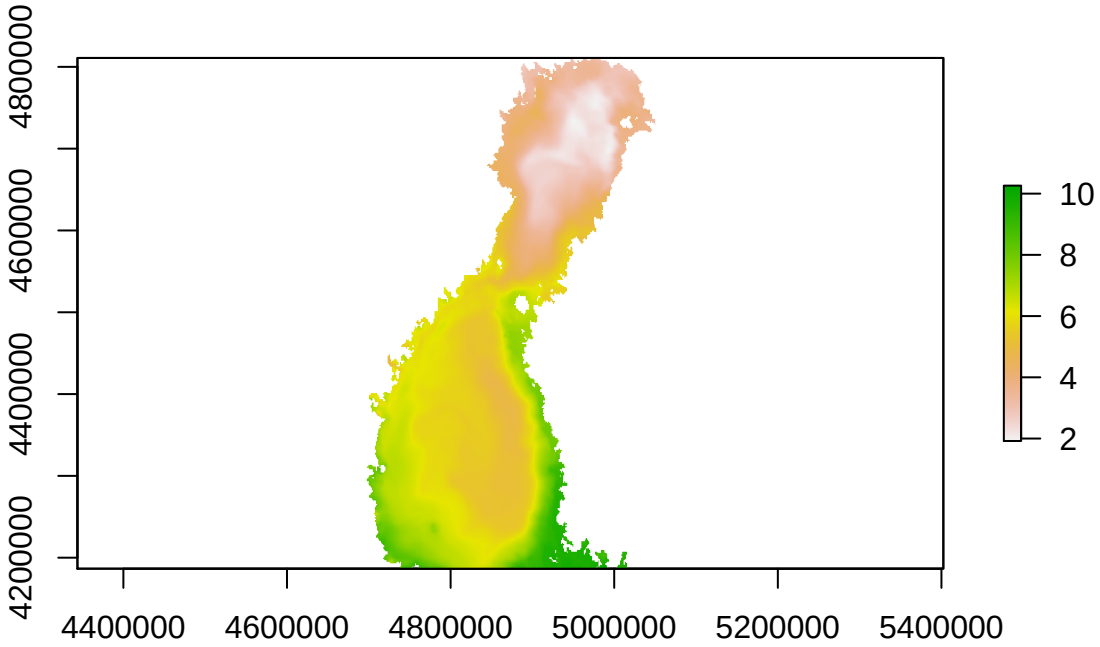
```
#try the function
lista = merge_raster.variable_df0("/home/piailari/Documents/thesis/SampleSmartSea/A001/NORDIC-GOB_1m_20
                          "votemper", band = 4:6, level = 1:3, final.raster = whitefish.raster, var.names =
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
head(lista$data.frame)
```

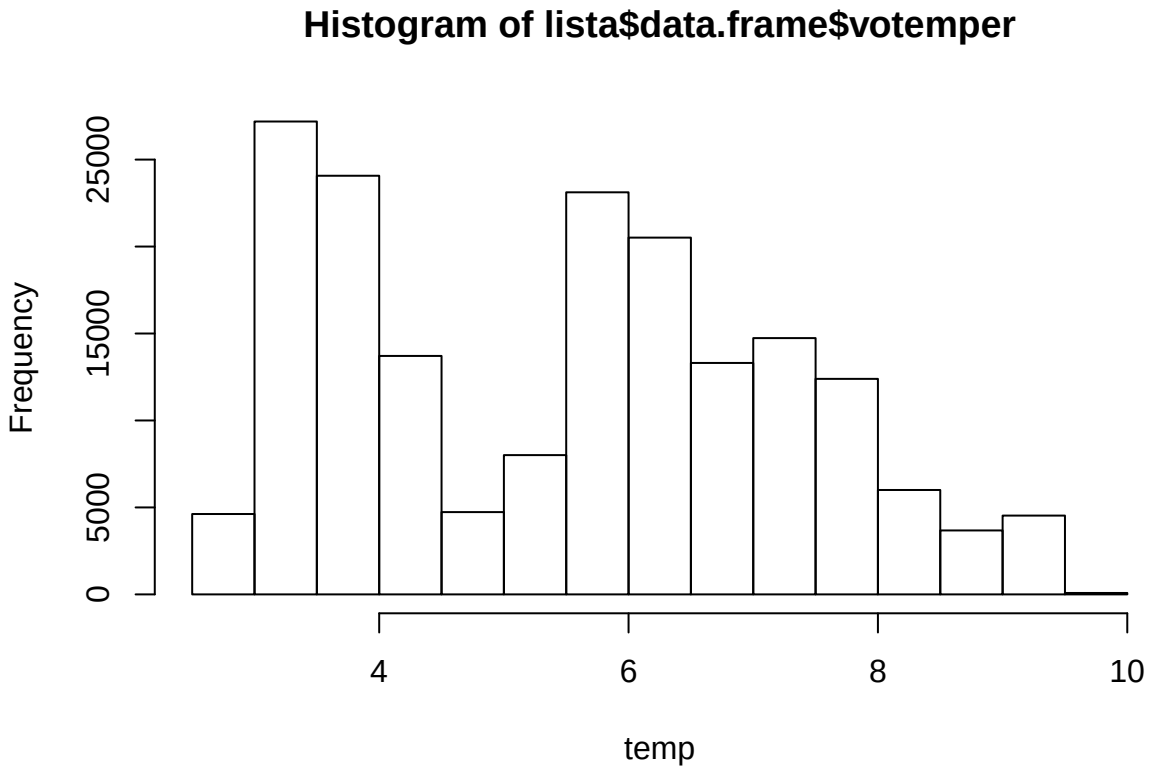
```
##       E_etrs89 N_etrs89 votemper FE300ME
## 14904  4698039  4293980 8.325198     881
## 14905  4698039  4294280 8.325198    8325
## 14906  4698039  4294581 8.325198    8522
## 14958  4698039  4310203 7.876825     272
## 14959  4698039  4310504 7.876825     900
## 14960  4698039  4310804 7.876825      66
```

```
plot(lista$raster.layer)
```



Note: we removed all land raster cells , ie the one where temperature is equal to 0. Another approach could be to fill such cells with the average value from their nearest neighbors, if 'remove.land'=F, such method is applied, considering the 8 nearest cells. In this way, though we still have have 15000 raster cells with 0 temperature, as they are surrounded just by land cells.

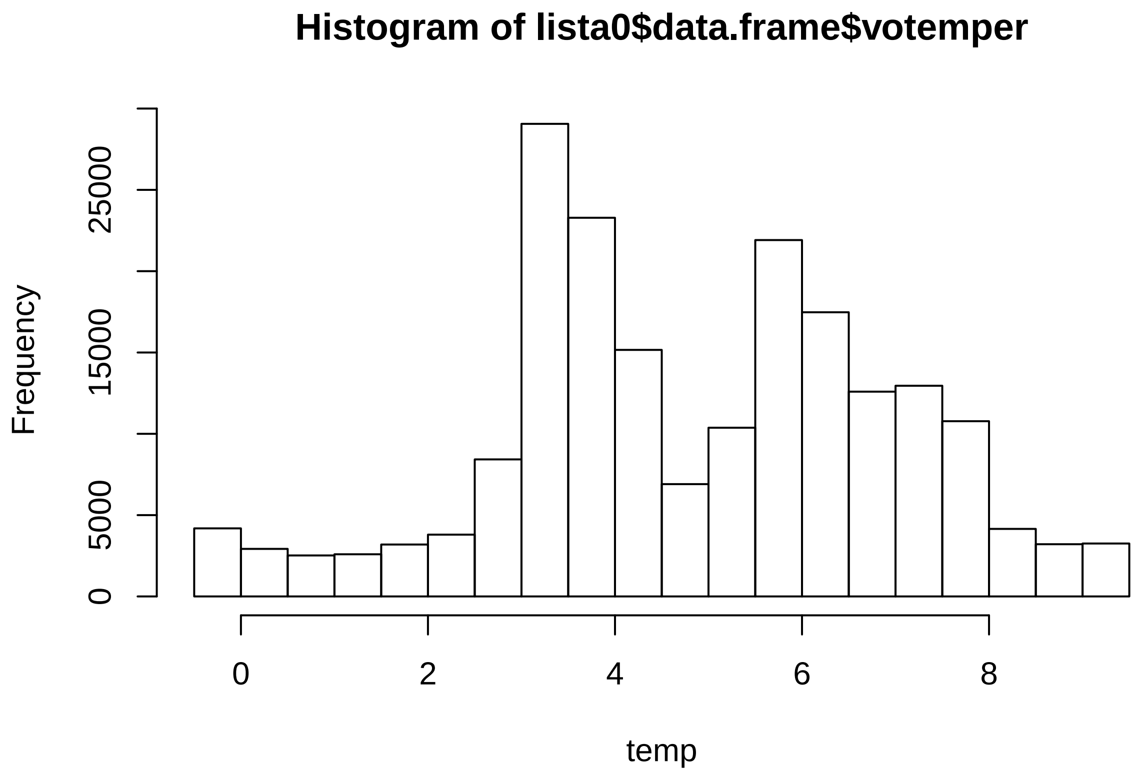
```
hist(lista$data.frame$votemper, xlab = "temp")
```



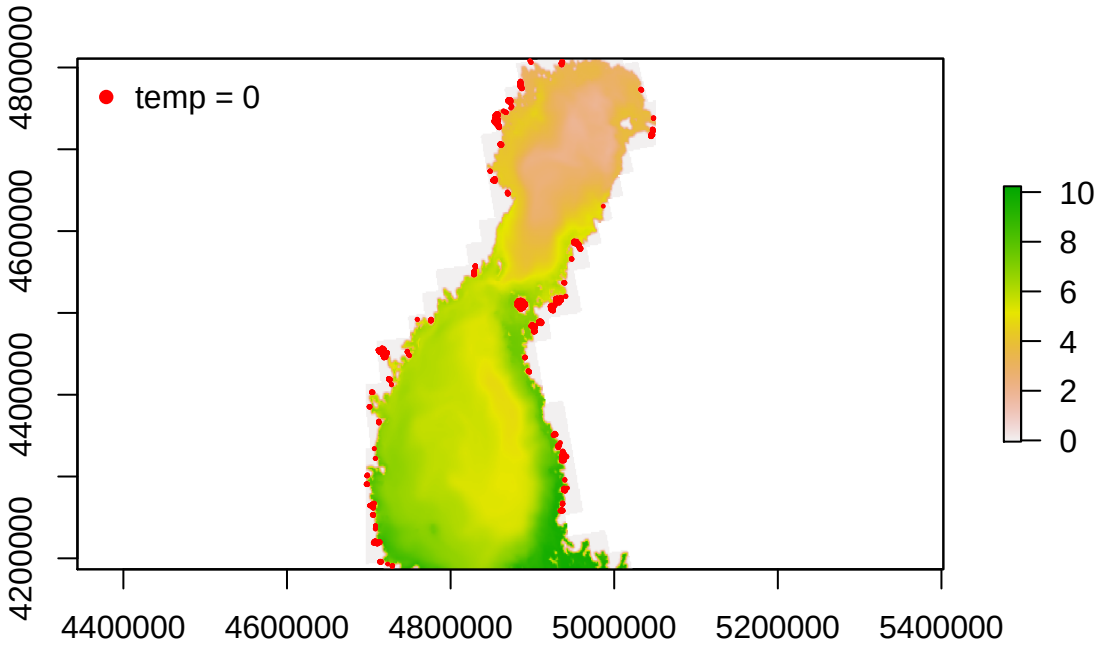
```
lista0 = merge_raster.variable_df0("/home/piailari/Documents/thesis/SampleSmartSea/A001/NORDIC-GOB_1m_20
                    "votemper", band = 4:6, level = 1:3, final.raster = whitefish.raster, var.names = c
                    remove.land=F)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
hist(lista0$data.frame$votemper, xlab = "temp")
```



```
temp0=lista0$data.frame[lista0$data.frame$votemp==0, ]
plot(lista0$raster.layer)
points(temp0[,1], temp0[,2], col=2, pch=20, cex=0.3)
legend("topleft", legend = "temp = 0", pch=16, col=2, bty = "n")
```



The next task, is to recover the raster cell to which each observation contained in the whitefish.dat file belongs. We will then select the corresponding temperature values and merge such values with the latter dataset

```
whitefish.dat = read.table("data_whitefish.txt", header=TRUE, sep="\t")
head(whitefish.dat)
```

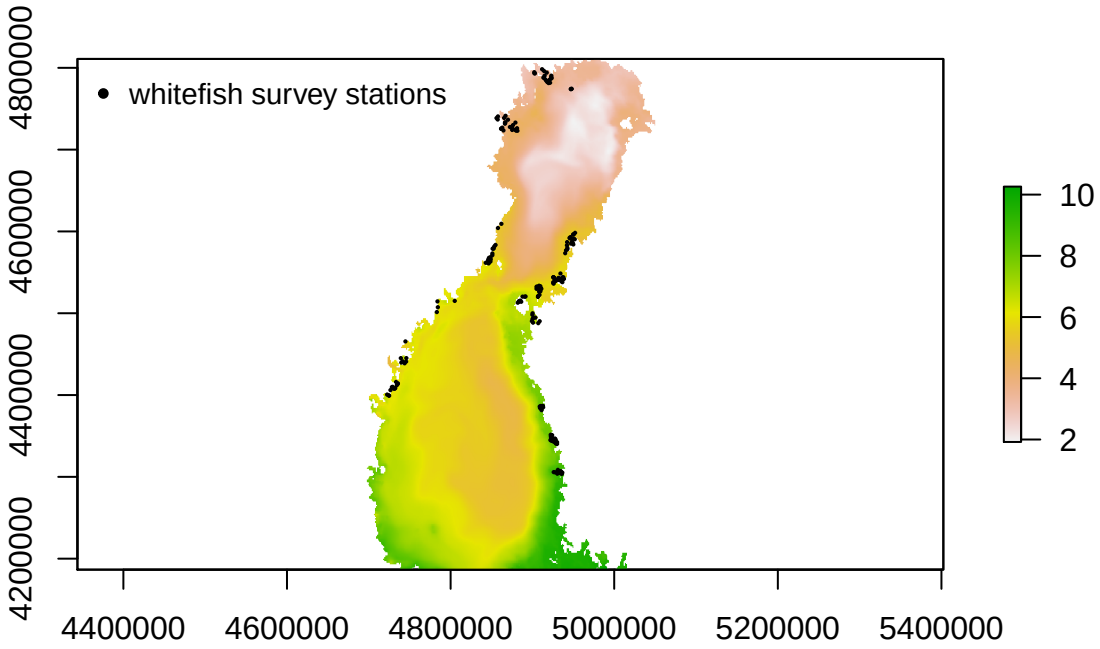
```
##   N_etrs89 E_etrs89 YEAR VOLUME WHISUM SHOPROFILE DEPTH BOTTOM BOTTOMCOV
## 1  4609326  4862079 2010    3.0      4          5   0.2      6         5
## 2  4604207  4857880 2010    7.5      1          1   0.5      6         4
## 3  4583479  4854910 2010    6.0     13          4   0.4      6         4
## 4  4581746  4853481 2010    7.5      2          1   0.5      6         4
## 5  4579606  4852488 2010    3.0      1          5   0.2      1         5
## 6  4578870  4852426 2010    9.0      0          4   0.3      6         4
##   FETCH300W FE300ME DIST20M LINED ISLANDNUM DIS_SAND DIS_SHALLO PE900 PE3000
## 1     63765    7162   65396    57        28       27       2480    10     10
## 2     61996   15065   42000    49        15       26       2519   110     29
## 3    103947    7059   34970    35        20       30       2574   123     54
## 4     65248   19087   51941    43        28       30       2588    71     48
## 5     44448    6440   62485    46        51       31       2599   101     22
## 6    105618    4462   85455    46        47       31       2602    24      9
##   SAL09SPRS SAL10SPRS SAL910WIN ICEFIRST09 ICEFIRST10 ICELAST09 ICELAST10
## 1    3.5672    4.0111    3.5318          1         -1        15        17
## 2    3.5433    3.9958    3.5177          1         -1        16        18
## 3    3.4977    3.9593    3.5123          1         -1        15        18
## 4    3.4977    3.9593    3.5123          1         -1        16        18
## 5    3.4879    3.9468    3.5340          1         -1        16        18
## 6    3.4879    3.9468    3.5340          1         -1        16        18
##   ICEWIN09 ICEWIN10 EUTSTAT EKOSTAT PHOSP NITROG CHL_A SECCHI RIVERS BOTTOMCLS
## 1       12       18   24.45       3 36.77  26.23 35.71  22.67   4958         4
```

```
## 2       14       19   24.13       3 36.78  27.41 35.41  22.84   4277         4
## 3       12       19   25.35       3 38.20  28.54 34.98  23.24   3391         0
## 4       14       19   25.01       3 38.05  29.30 34.69  23.00    589         4
## 5       14       19   25.11       3 38.20  29.41 34.57  23.00   2877         4
## 6       14       19   25.13       3 38.41  29.90 34.35  23.02   3458         4
##   SHAREA
## 1    101
## 2    100
## 3     71
## 4    161
## 5     67
## 6     44
```

```
head(lista$data.frame)
```

```
##       E_etrs89 N_etrs89 votemper FE300ME
## 14904  4698039  4293980 8.325198     881
## 14905  4698039  4294280 8.325198    8325
## 14906  4698039  4294581 8.325198    8522
## 14958  4698039  4310203 7.876825     272
## 14959  4698039  4310504 7.876825     900
## 14960  4698039  4310804 7.876825      66
```

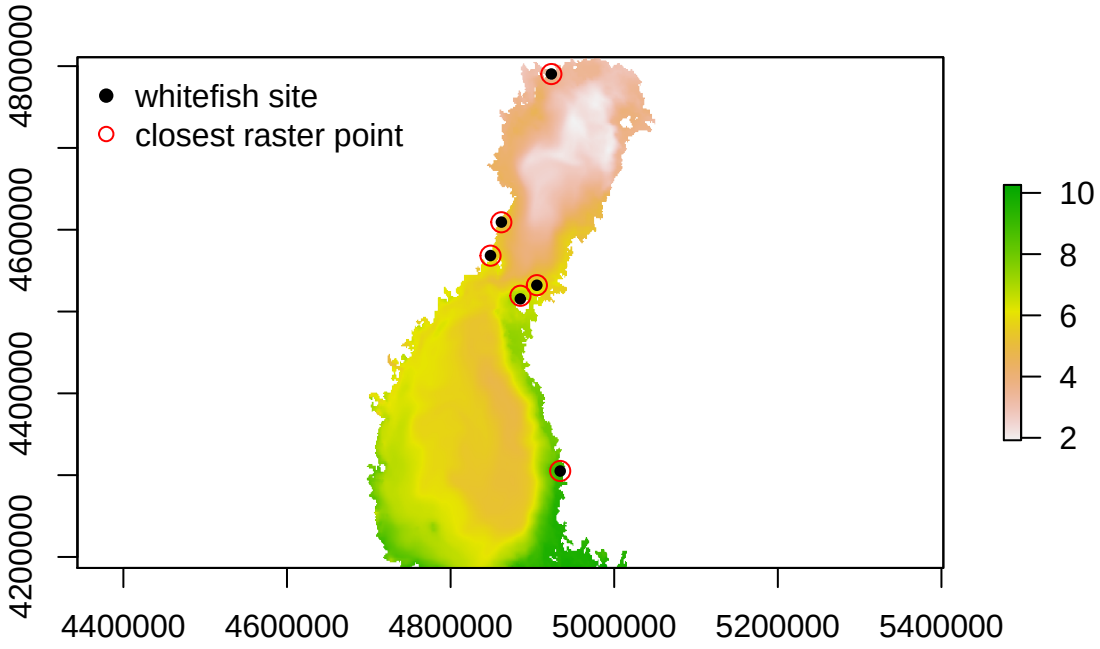
```
plot(lista$raster.layer)
points(whitefish.dat$E_etrs89, whitefish.dat$N_etrs89, col=1, pch=16, cex=0.3)
legend("topleft", legend = "whitefish survey stations", pch=20, col=1, bty = "n", cex=0.9)
```



For each whitefish data-point, we look for the cell in our raster, such that is the closest to it, by computing the square euclidean distances among the data from the two files.

```
rast.index=c()
for (i in 1:nrow(whitefish.dat)) {
  dist.E=(abs(lista$data.frame$E_etrs89- whitefish.dat$E_etrs89[i]))**2
  dist.N=(abs(lista$data.frame$N_etrs89- whitefish.dat$N_etrs89[i]))**2
  # square distance
```

```
  d =dist.E+dist.N
  # closest cell
  rast.index[i] = which.min(d)
}

# plot some of the sites
plot(lista$raster.layer)
pos =c(1,10,50,100,150,200)
points(whitefish.dat$E_etrs89[pos], whitefish.dat$N_etrs89[pos], col=1, pch=16, cex=0.8)
points(lista$data.frame$E_etrs89[rast.index[pos]], lista$data.frame$N_etrs89[rast.index[pos]], col=2, p
legend("topleft", legend = c("whitefish site", "closest raster point"), pch = c(16,1), col=c(1,2), bty =
```




```
TEMP = lista$data.frame$votemper[rast.index]
whitefish.dat.temp =data.frame(whitefish.dat, TEMP)
head(whitefish.dat.temp)
```

```
##   N_etrs89 E_etrs89 YEAR VOLUME WHISUM SHOPROFILE DEPTH BOTTOM BOTTOMCOV
## 1  4609326  4862079 2010    3.0      4          5   0.2      6         5
## 2  4604207  4857880 2010    7.5      1          1   0.5      6         4
## 3  4583479  4854910 2010    6.0     13          4   0.4      6         4
## 4  4581746  4853481 2010    7.5      2          1   0.5      6         4
## 5  4579606  4852488 2010    3.0      1          5   0.2      1         5
## 6  4578870  4852426 2010    9.0      0          4   0.3      6         4
##   FETCH300W FE300ME DIST20M LINED ISLANDNUM DIS_SAND DIS_SHALLO PE900 PE3000
## 1     63765    7162   65396    57        28       27       2480    10     10
## 2     61996   15065   42000    49        15       26       2519   110     29
## 3    103947    7059   34970    35        20       30       2574   123     54
## 4     65248   19087   51941    43        28       30       2588    71     48
## 5     44448    6440   62485    46        51       31       2599   101     22
## 6    105618    4462   85455    46        47       31       2602    24      9
##   SAL09SPRS SAL10SPRS SAL910WIN ICEFIRST09 ICEFIRST10 ICELAST09 ICELAST10
## 1    3.5672    4.0111    3.5318          1         -1        15        17
## 2    3.5433    3.9958    3.5177          1         -1        16        18
```

```
## 3    3.4977    3.9593    3.5123          1         -1        15        18
## 4    3.4977    3.9593    3.5123          1         -1        16        18
## 5    3.4879    3.9468    3.5340          1         -1        16        18
## 6    3.4879    3.9468    3.5340          1         -1        16        18
##   ICEWIN09 ICEWIN10 EUTSTAT EKOSTAT PHOSP NITROG CHL_A SECCHI RIVERS BOTTOMCLS
## 1       12       18   24.45       3 36.77  26.23 35.71  22.67   4958         4
## 2       14       19   24.13       3 36.78  27.41 35.41  22.84   4277         4
## 3       12       19   25.35       3 38.20  28.54 34.98  23.24   3391         0
## 4       14       19   25.01       3 38.05  29.30 34.69  23.00    589         4
## 5       14       19   25.11       3 38.20  29.41 34.57  23.00   2877         4
## 6       14       19   25.13       3 38.41  29.90 34.35  23.02   3458         4
##   SHAREA     TEMP
## 1    101 5.529678
## 2    100 5.480217
## 3     71 5.685743
## 4    161 5.758716
## 5     67 5.775214
## 6     44 5.798412
```

```
length(which(TEMP==0)) #0 obs with 0 value, ie temperature here was not measured:land spot
```

```
## [1] 0
```

Final data frame construction

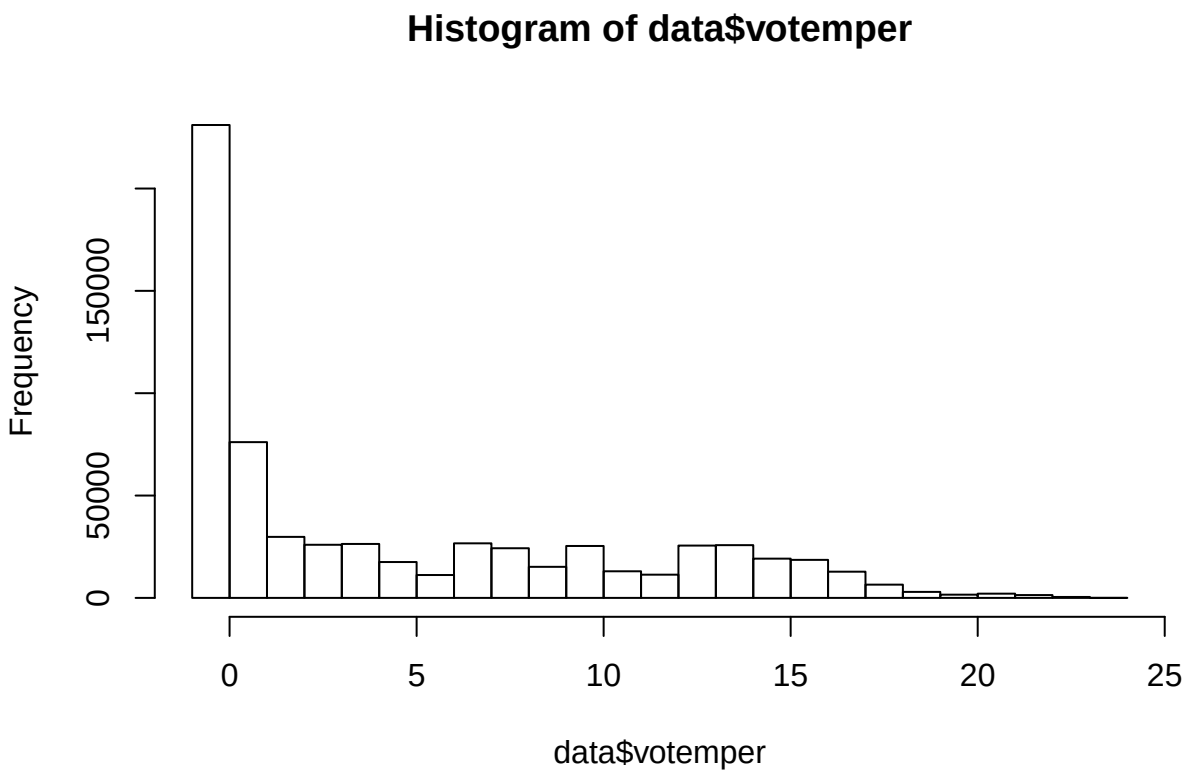
```
library(ncdf4)
library(raster)
```

Let's have a look at one of the file, from the latest SeaSmart data repositories and test the function we built. We took the 2010 monthly temperature

```
setwd("/home/piailari/Documents/thesis/SmartSeaTemperature_monthly_0-9m")

data= open_nc("votemper_0-9_A002_2010_mean.nc")
```

```
## Warning: variable votemper contains 818232 NA values
```

```
hist(data$votemper)
```




```
apr.t=raster("votemper_0-9_A001_1979_mean.nc", varname="votemper", band =4)
```

```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
may.t=raster("votemper_0-9_A001_1979_mean.nc", varname="votemper", band =5)
```

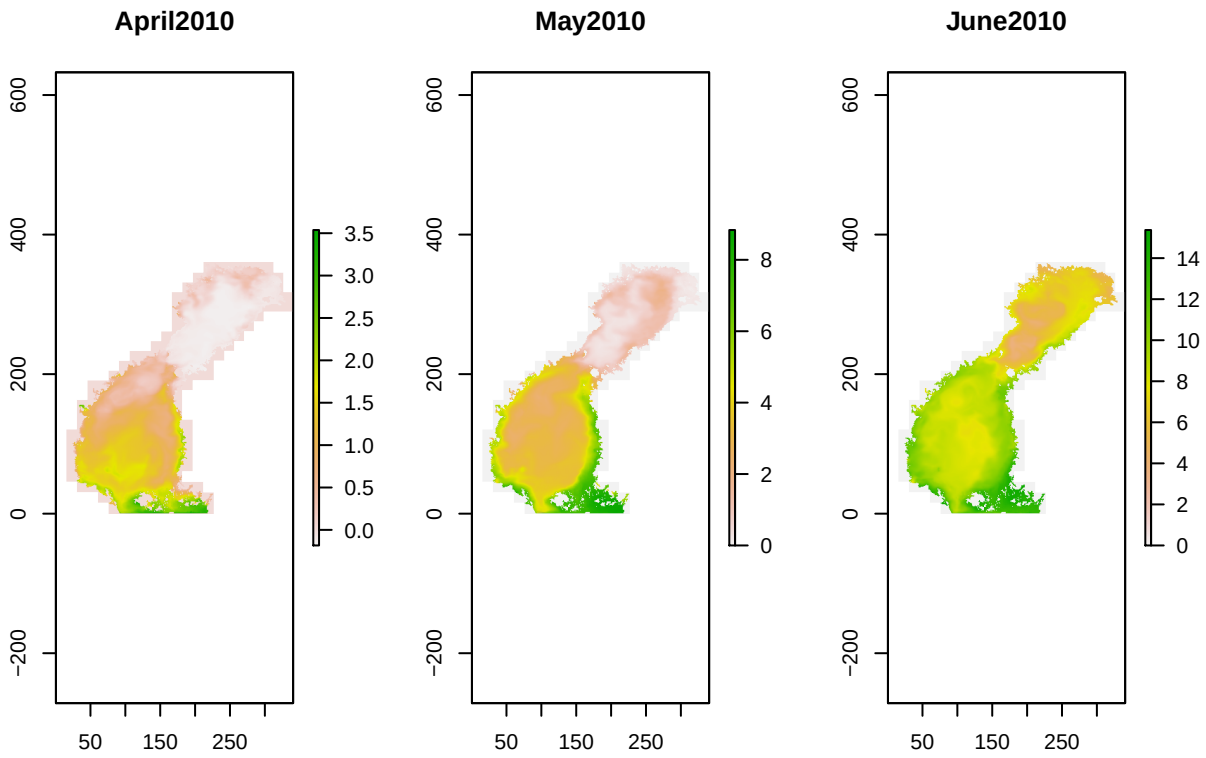
```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
june.t=raster("votemper_0-9_A001_1979_mean.nc", varname="votemper", band =6)
```

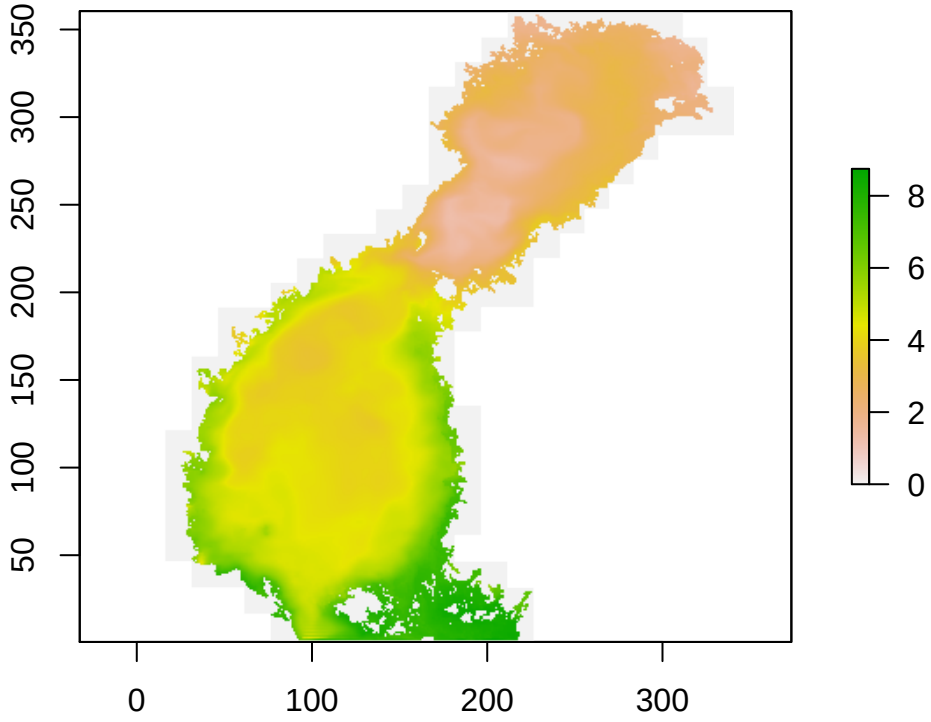
```
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named x BUT this dimens
## [1] "vobjtovarid4: **** WARNING **** I was asked to get a varid for dimension named y BUT this dimens
```

```
par(mfrow=c(1,3))
plot(apr.t, main ="April2010")
```

```
plot(may.t, main ="May2010")
plot(june.t, main ="June2010")
```

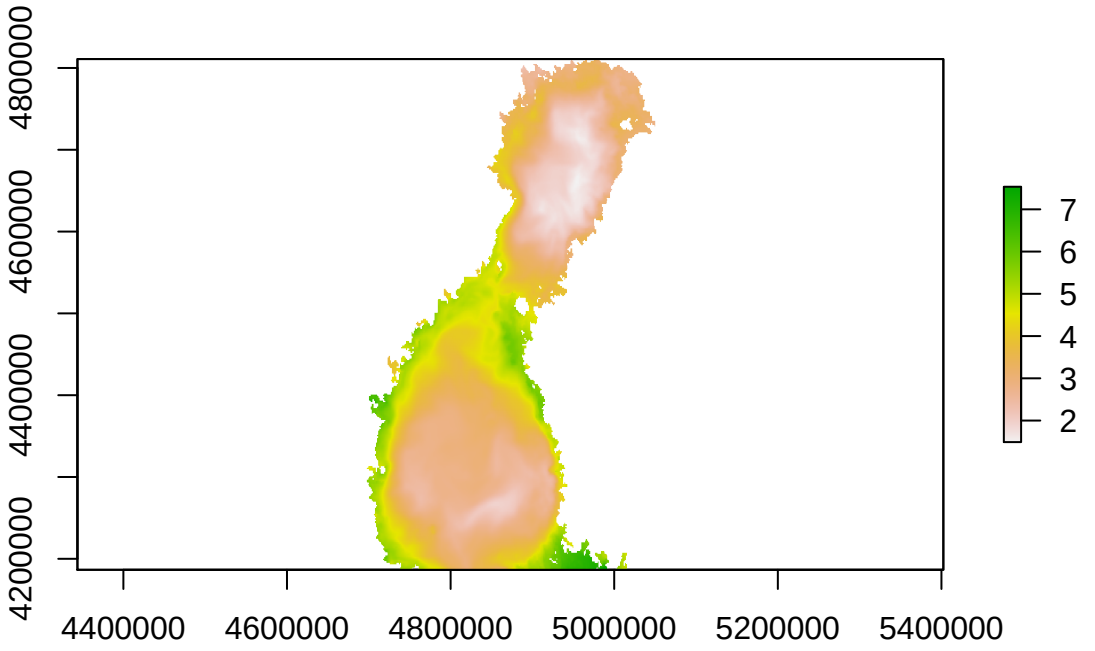



```
mean.t=overlay(apr.t, may.t, june.t, fun="mean")
par(mfrow=c(1,1))
plot(mean.t)
```



```
path ="/home/piailari/Documents/thesis/SmartSeaTemperature_monthly_0-9m/votemper_0-9_A002_2010_mean.nc"
lista10 = merge_raster.variable_df0(path, var = "votemper", band = 4:6, level = 1:3, final.raster = whi
```

```
plot(lista10$raster.layer)
```




```
head(lista10$data.frame)
```

```
##      E_etrs89 N_etrs89 votemper
## 2021  4946829  4810424 2.615301
## 2022  4947128  4810424 2.615301
## 2023  4947428  4810424 2.615301
## 2024  4947728  4810424 2.615301
## 2025  4948028  4810424 2.615301
## 2026  4948327  4810424 2.615301
```

We merge it to whitefishdata, as done previously. First we load the full whitefish dataframe, and select the covariates of interest.

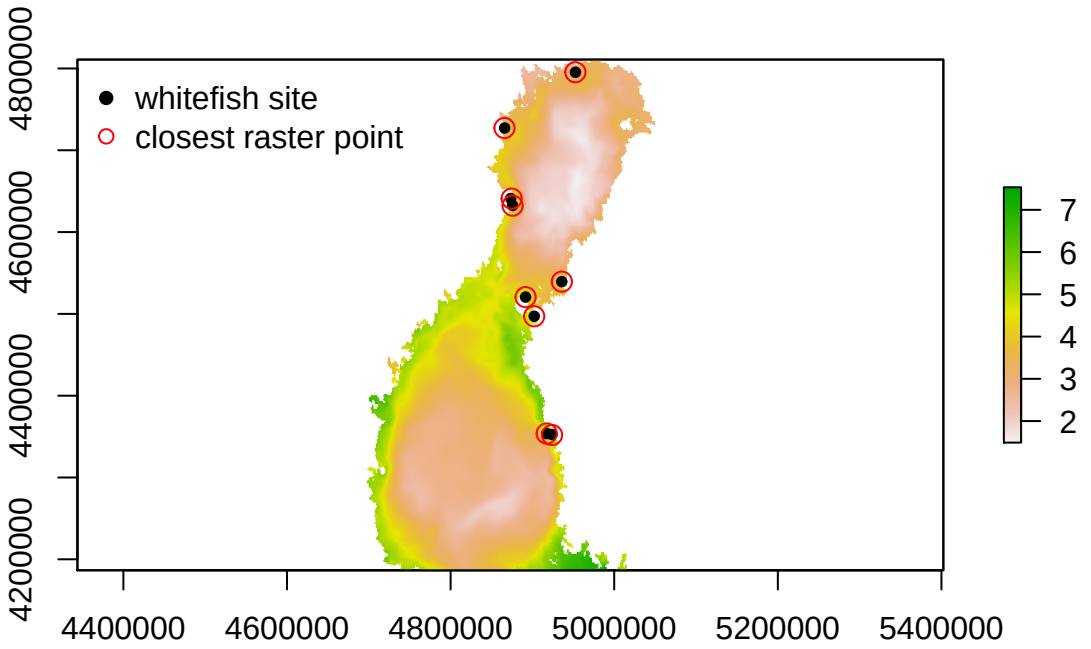
```
library(readxl)
setwd("/home/piailari/Documents/BayesianDataAnalysis/exercises/week3/exercise2b")
data.full = read_xlsx("bsg653_3.xlsx")

colnames(data.full)
```

```
##  [1] "DATA"        "FUNDAREA"    "YEAR"        "AREA"        "AREANAME"
##  [6] "EFFN"        "METHOD"      "SITE"        "SITE1"       "DAY"
## [11] "MONTH"       "HOUR"        "MINUTE"      "SECOND"      "N"
## [16] "E"           "SPECIES"     "GACMAX40"    "GACMIN40"    "PUNG"
## [21] "GACMAX40BIN" "GACMIN40BIN" "PUNGBIN"     "VOLUME"      "WHIVOL"
## [26] "VENVOL"      "WHISUM"      "VENSUM"      "WHIBIN"      "VENBIN"
## [31] "WTEMP"       "SHOPROFILE"  "DEPTH"       "BOTTOM"      "BOTTOMCOV"
## [36] "FETCH300W"   "FE300ME"     "DIST20M"     "LINED"       "ISLANDNUM"
## [41] "DIS_SAND"    "DIS_SHALLO"  "PE900"       "PE3000"      "SAL09SPRS"
## [46] "SAL10SPRS"   "SAL910WIN"   "ICEFIRST09"  "ICEFIRST10"  "ICELAST09"
## [51] "ICELAST10"   "ICEWIN09"    "ICEWIN10"    "EUTSTAT"     "EKOSTAT"
```

```
## [56] "PHOSP"       "NITROG"      "CHL_A"       "SECCHI"      "RIVERS"
## [61] "BOTTOMCLS"   "SHAREA"      "SEA_AREA"
```

```
whitefish.dat = data.full[, c("N", "E", "FE300ME" , "BOTTOMCLS", "DIS_SAND", "FE300ME", "ICELAST09", "R
colnames(whitefish.dat)[1:2]= c("N_etrs89", "E_etrs89")
```

```
rast.index=c()
for (i in 1:nrow(whitefish.dat)) {
  dist.E=(abs(lista10$data.frame$E_etrs89- whitefish.dat$E_etrs89[i]))**2
  dist.N=(abs(lista10$data.frame$N_etrs89- whitefish.dat$N_etrs89[i]))**2
  # square distance
  d =dist.E+dist.N
  # closest cell
  rast.index[i] = which.min(d)
}

# plot some of the sites
plot(lista10$raster.layer)
pos =c(1,10,50,100,150,200,300,420,550)
points(whitefish.dat$E_etrs89[pos], whitefish.dat$N_etrs89[pos], col=1, pch=16, cex=0.8)
points(lista10$data.frame$E_etrs89[rast.index[pos]], lista10$data.frame$N_etrs89[rast.index[pos]], col=
legend("topleft", legend = c("whitefish site", "closest raster point"), pch = c(16,1), col=c(1,2), bty =
```




```
TEMP = lista10$data.frame$votemper[rast.index]
whitefish.dat.temp =data.frame(whitefish.dat, TEMP)
head(whitefish.dat.temp)
```

```
##   N_etrs89 E_etrs89 FE300ME BOTTOMCLS DIS_SAND FE300ME.1 ICELAST09 RIVERS
## 1  4632492  4875736    7443         1       17      7443        16  12053
## 2  4632150  4876171    7378         5       17      7378        16  12002
## 3  4633801  4875794   30121         1       17     30121        16  14062
## 4  4637652  4878980    7631         5       15      7631        16  18521
## 5  4637717  4878414    7537         1       15      7537        16  18458
## 6  4638320  4879176    3290         1       14      3290        16  18515
##   SAL910WIN DIST20M CHL_A     TEMP
```

```
## 1    3.5246  152308 36.31 4.531885
## 2    3.5246  148793 36.40 4.450592
## 3    3.5348  160970 36.31 4.443753
## 4    3.5348   71396 36.31 4.272337
## 5    3.5348   91882 36.31 4.361594
## 6    3.5348   79882 36.31 4.270304
```

```
length(which(TEMP==0)) #0 obs with 0 value, ie temperature here was not measured:land spot
```

```
## [1] 0
```

Since all of the results are model based and the last "historical year" in those model runs is 2005 we would use years 1995-2005 to represent the "current values" for temperature and salinity. We build the training temp and salinity data so that we take
* temp: the average of the temperature values in April-June over all years in 1995-2005 in water depths 0-9
* salinity: the average of the monthly salinity values over all months in years 1995-2005 in water depths 0-9

We first focus on the temperature, measured in April-May. this time the depth is already set to 0-9m, so we don't need to worry about the layer levels.

We modify the function, so that it fits our new data

```
merge_raster.variable_df = function(file, var, band, level, final.raster , final.dat , var.names =c(), l
  "
  This function takes as arguments
  file : vector of strings, paths of raster files we want to extract a layer from   (EX Smartsea Gob fil
  var : name of the raster.layer variable we want to extract from file  (EX temperature)
  band :  number or vector, bands to consider (averaged)
  level : number or vector, levels to consider (averaged)
  final.raster : dataframe from which we extra the raster file we want to conform our layer with
  final.dat : dataframe to which we add our final variable (with coordinates: E_etrs89, N_etrs89 !!!)
  var.names : variables names of final.raster file we want to include in the output dataframe (optional)
  lon.name : name of the longitude variable from file, default lon
  lat.name : name of the latitude variable from file, default lat
  lon.final : name of longitude variable from final.raster, default X
  lat.final : name of latitude variable from final.raster, default Y

  and returns a list with 2 objects
  data.frame : df with  longitude, latitude (in the same crs and resolution of r.final), the  variable o
  raster.layer : raster layer of the variable of interest in the same crs and resolution of r.final
  "

  library(raster)
  library(rgdal)

  # layer with average band and level var
  rast0 = raster(file, varname=var,  band = band[1], level = level[1]) # raster with proper extension
  mean.layer = overlay(-rast0, rast0, fun ="sum") # create a 0 layer to start with
  for (f in file) {
    for (i in band) {
    for (j in level) {
      rast = raster(f, varname=var,  band = i, level = j)
      mean.layer = overlay(rast, mean.layer, fun =sum)
    }
  }
  }
  mean.layer = mean.layer/(length(band)*length(level)*length(file))
```

```
# remove land cells
mean.layer[mean.layer==0]=NA

# raster file
lon = raster(file, varname=lon.name)
lat = raster(file, varname=lat.name)
d =  cbind(values(lon), values(lat), values(mean.layer))
d = na.exclude(d)
# from df to raster
e = extent(cbind(d[,1],d[,2]))
r = raster(e, ncol=length(unique(d[,1])), nrow=length(unique(d[,2])))
mean.r = rasterize(d[,1:2],r, d[,3], fun=mean)

# extrat quantities from final raster file
e.final = extent(cbind(final.raster[, lon.final],final.raster[, lat.final]))
r.final = raster(e.final, ncol=length(unique(final.raster[, lon.final])), nrow=length(unique(final.ras
res.final = res(r.final)
crs.final = crs(r.final)
if(is.na(crs.final)) crs.final=CRS("+init=epsg:3035")

# set original crs
if(is.na(crs(mean.r))) crs(mean.r) = CRS("+proj=longlat +ellps=WGS84") # lat lon, geo.cs : CRS("+init
# project
mean.pr = projectRaster(mean.r, crs=crs.final)

# crop
mean.pr = crop(mean.pr, e.final)

# upscale
mean.upsc = disaggregate(mean.pr, fact = res(mean.pr)/res.final)
mean.upsc = resample(mean.upsc,r.final, method="bilinear")

# create a new data frame
dat.new=as.data.frame(mean.upsc, xy=T)
colnames(dat.new) = c("E_etrs89", "N_etrs89", var)
# remove NA
dat.new = dat.new[complete.cases(dat.new ),]

# merge it with the final data frame
rast.index=c()
for (i in 1:nrow(final.dat)) {
  dist.E=(abs(dat.new$E_etrs89- final.dat$E_etrs89[i]))**2
  dist.N=(abs(dat.new$N_etrs89- final.dat$N_etrs89[i]))**2
  # square distance
  d =dist.E+dist.N
  # closest cell
  rast.index[i] = which.min(d)
}

var.f = dat.new[rast.index,3]
final.dat = final.dat[,var.names]
df =data.frame(final.dat, var =var.f)
```

```
  return(list("data.frame" = df, "raster.layer" = mean.upsc))
}
```

We first load the final raster file and dataframe

```
### whitefish raster data
setwd("/home/piailari/Documents/thesis")
whitefish.raster = read.table("predraster_whitefish.txt", header=TRUE, sep="\t")

### whitefish data frame
library(readxl)
setwd("/home/piailari/Documents/BayesianDataAnalysis/exercises/week3/exercise2b")
whitefish.dat = read_xlsx("bsg653_3.xlsx")
colnames(whitefish.dat)[15:16]= c("N_etrs89", "E_etrs89")
# covariates of intererst
variables = c("N_etrs89", "E_etrs89", "FE300ME" , "BOTTOMCLS", "DIS_SAND", "ICELAST09", "RIVERS", "SAL9
```

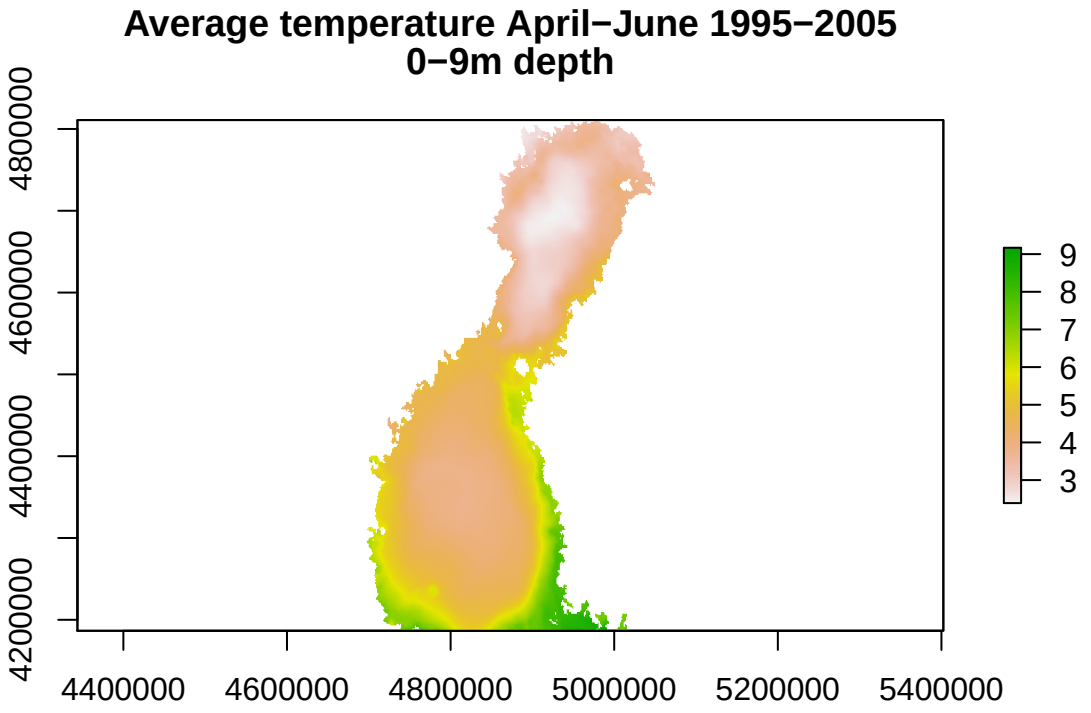
Now we select the files from years 1995-2005

```
path ="/home/piailari/Documents/thesis/SmartSeaTemperature_monthly_0-9m/"
years=1995:2005
names.t=c()
for(i in 1:length(years)) names.t[i]= paste(path, "votemper_0-9_A001_",years[i],"_mean.nc", sep = "")
```

and apply the function

```
l = merge_raster.variable_df(names.t, var = "votemper", band = 4:6, level = 1, final.raster = whitefish
                             final.dat = whitefish.dat , var.names = variables , lon.name = "nav_lon", l
```

```
plot(l$raster.layer, main=c("Average temperature April-June 1995-2005", "0-9m depth"))
```




```
colnames(l$data.frame)[ncol(l$data.frame)] ="TEMP09M"
head(l$data.frame)
```

```
##   N_etrs89 E_etrs89 FE300ME BOTTOMCLS DIS_SAND ICELAST09 RIVERS SAL910WIN
## 1  4632492  4875736    7443         1       17         16  12053    3.5246
## 2  4632150  4876171    7378         5       17         16  12002    3.5246
## 3  4633801  4875794   30121         1       17         16  14062    3.5348
## 4  4637652  4878980    7631         5       15         16  18521    3.5348
## 5  4637717  4878414    7537         1       15         16  18458    3.5348
## 6  4638320  4879176    3290         1       14         16  18515    3.5348
##   DIST20M CHL_A  TEMP09M
## 1  152308 36.31 4.063705
## 2  148793 36.40 4.016508
## 3  160970 36.31 4.012287
## 4   71396 36.31 3.960890
## 5   91882 36.31 4.022866
## 6   79882 36.31 3.962358
```

We repeat the procedure for salinity

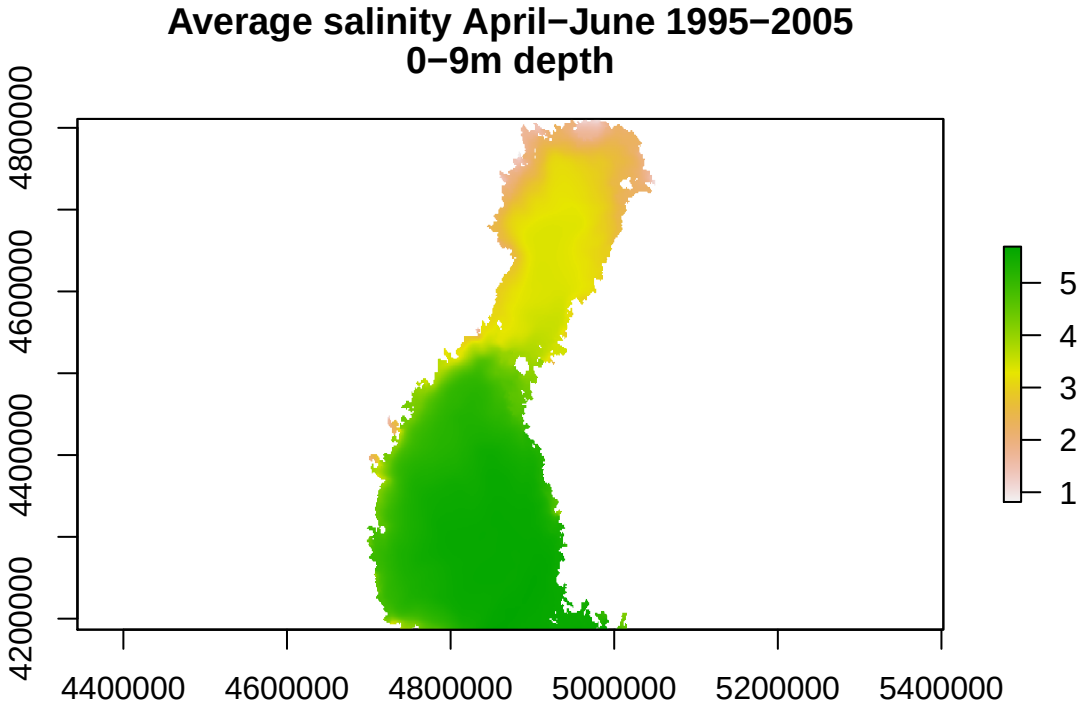
```
path ="/home/piailari/Documents/thesis/SmartSeaSalinity_monthly_0-9m/"
years=1995:2005
names.s=c()
for(i in 1:length(years)) names.s[i]= paste(path, "vosaline_0-9_A001_",years[i],"_mean.nc", sep = "")
```

and apply the function

```
ls = merge_raster.variable_df(names.s, var = "vosaline", band = 4:6, level = 1, final.raster = whitefis
                              final.dat = whitefish.dat , var.names = variables , lon.name = "nav_lon",
```

We get the final dataset.

```
plot(ls$raster.layer, main=c("Average salinity April-June 1995-2005", "0-9m depth"))
```




```
# final data
data.whitefish.final = data.frame(l$data.frame, "SALT09M" =ls$data.frame$var)
head(data.whitefish.final)
```

```
##   N_etrs89 E_etrs89 FE300ME BOTTOMCLS DIS_SAND ICELAST09 RIVERS SAL910WIN
## 1  4632492  4875736    7443         1       17        16  12053    3.5246
## 2  4632150  4876171    7378         5       17        16  12002    3.5246
## 3  4633801  4875794   30121         1       17        16  14062    3.5348
## 4  4637652  4878980    7631         5       15        16  18521    3.5348
## 5  4637717  4878414    7537         1       15        16  18458    3.5348
## 6  4638320  4879176    3290         1       14        16  18515    3.5348
##   DIST20M CHL_A  TEMP09M  SALT09M
## 1  152308 36.31 4.063705 2.793786
## 2  148793 36.40 4.016508 2.780895
## 3  160970 36.31 4.012287 2.779378
## 4   71396 36.31 3.960890 2.719728
## 5   91882 36.31 4.022866 2.724489
## 6   79882 36.31 3.962358 2.717036
```

```
#     write.table(data.whitefish.final, file="white_fishes_final.txt", row.names=FALSE, col.names=TRUE)
```

Finally we createt the raster data file with all the covariates of interest

```
df.temp = as.data.frame(l$raster.layer, xy=T)
df.salt = as.data.frame(ls$raster.layer, xy=T)
#raster.whitefish.final = merge(df.temp, df.salt, by= c("x", "y"))
#colnames(raster.whitefish.final) = c("x", "y", "TEMP09M", "SALT09M")

covariates = c( "FE300ME" ,  "BOTTOMCLS", "DIS_SAND"  ,"ICELAST09", "RIVERS"    ,"SAL910WIN" ,"DIST20M"
l.rast = merge_raster.variable_df0(names.t, var = "votemper", band = 4:6, level = 1, final.raster = whi
                                   var.names = covariates , lon.name = "nav_lon", lat.name = "nav_lat")
ls.rast = merge_raster.variable_df0(names.s, var = "vosaline", band = 4:6, level = 1, final.raster = wh
                                   var.names = c( "FE300ME") , lon.name = "nav_lon", lat.name = "nav_lat")

raster.whitefish.final = l.rast$data.frame
colnames(raster.whitefish.final)[3] ="TEMP09M"
raster.whitefish.final =data.frame(raster.whitefish.final, "SALT09M" =ls.rast$data.frame[,"vosaline"])
head(raster.whitefish.final)

# write.table(raster.whitefish.final, file="white_fishes_final_raster.txt", row.names=FALSE, col.names=
```