FishMaps: producing maps for fishery data in batch mode

Fernando P. Mayer
fernandomayer@gmail.com

August, 2007

1 Introduction

Fisheries scientists often need to display their information in a spatially structured way. Since fleets quickly moves searching for target species according to the season of the year, it is a fundamental part of every fishery analysis to plot fishery data into maps, in order to show the dynamics of the fishing fleet. Quantitative fishery data are represented here by the catch (the weight of fish caught), the effort (*e.g.* days fishing or number of hooks), and the Catch Per Unity of Effort or CPUE (the ratio between catch and effort, which can be assumed to be proportional to the population abundance). For those interested in more details about these terms, a basic reference is the text of Gulland (1983) [1]. All this information, specially the CPUE plotted in georeferenced maps by some unit of time (*e.g.* year and/or quarter) may be usefull to identify areas of large abundance or vulnerability of some species. Furthermore, maps can help in the study of the behaviour and movements of fisherman across time and space.

The tuna and tuna-like fisheries in south of Brazil are among the more profitable. The long-line fleet targets the swordfish (*Xiphias gladius*), which reache a high price in the international market, while the bait-boat fleet targets the skipjack tuna (*Katsuwonus pelamis*), which is less valuable but the large catches compensate the effort. Fisheries Study Group (GEP) monitore those fisheries and publishes a series of annual reports about them. The main purposes are to show the dynamics of fishing fleets and the high catch spots across time and space.

Because reports have to be published in the same way every year, it was decided to implement such a functionality of producing fisheries maps in R. Actually, there is a great number of commercial specialized GIS softwares

available to make this maps. Although these programs does their jobs very well, producing maps in batch mode, *e.g.* by year and quarter for a data base from 1990 to 2005, may take you some time[1]. Moreover, the cost and limitation of (usually) running in only one operating system is also a disadvantage. The task of producing maps in batch mode and in the same environment of the statistical analysis was the main motivations to start a new package to provide this functionality in R. Furthermore, the possibility of using the package in any operating system (that only depends where R is running, and, as known, R runs in a wide variety of operating systems) was also of great interest.

Code to produce proportional symbol mapping in R has already been available (see [2]). However, that code is only usefull for data that will be displayed on land because it uses the centroids of the polygons (used to draw pieces of land) as the reference to plot the symbols. For oceanographic georeferenced data this doesn't work because there are no polygons for the oceans. On CRAN there is also the PBSmapping package [3], which, among other things, has the functionality of producing high-quality maps for fishey data. Although this package is intended for this purpose, users who want a simple solution and with no or little experience on some GIS concepts (such as polylines and polygons) may find it difficult to achieve their results, despite the final quality.

The FishMaps package is a much simpler interface for those users that need a simple display of quantitative fishery information and have little experience with GIS concepts. FishMaps is simple but may be used to quickly produce a great number of maps in batch mode and save them automatically in your working directory in a file format suitable for inclusion in documents. Despite the original motivation to build the package was the tuna and tuna-like fishery, users have the possibility to make their own maps and plot any kind of georeferenced data.

2 Package design

Besides functions from the R base packages, FishMaps uses several functions from other packages. For example, it uses the `map()` (which is the kernel producing the maps) from the maps package [4] (code originally designed for S by [5]), the `worldHires` database (which provides hi-resolution maps) from the mapdata package [6], and `degAxis()` (which makes the map's axes with the degree symbol) from the sp package [7].

[1] Supposing you have to make maps one by one.

FishMaps itself consists of four datasets and four functions. Two of this functions (`mapBB()` and `mapLL()`) can be considered only "transitional", since they are only used inside the other two main functions (`mapq()` and `mapy()`). Although been transitional, the first two functions are explicitly shown because they generate a standard map for bait-boat (`mapBB()`) and long-line (`mapLL()`) fishery data. This is a way users who will utilize these standard maps can see and manipulate previously its arguments.

Functions `mapq()` and `mapy()` are similar, but the first is used to produce maps by year and quarter, and the second only by year. Both functions can use pre-defined maps, such as the ones produced by `mapBB()` or `mapLL()`, or users can also define their own map boundaries. Note that using the pre-defined maps, for now, is only usefull for data in the southwest of the Atlantic Ocean, since the functions were originally designed for that. With the development of the package, more pre-defined maps can be built, if users request this. While this don't happen, users can also define their own map functions in the same way.

Internally, in both functions (`mapq()` and `mapy()`), data are properly separated to form classes using the standard `cut()` function (with the argument `include.lowest = TRUE` fixed). As in this function, one can use the argument `breaks`, that is directly passed to `cut()`. This allows to specify the number of desired classes or the break points for the classes. Each class will then receive an index from $1, \ldots, n$ where n is the number of classes. This index will be used as the `cex` argument to the function `points()`, which will make the symbols (solid circles) proportional to the index of each class. The symbols will then be plotted in the map according to their latitude and longitude coordinates.

Optional legends for classes are also provided. The legend contains the proportional symbol and the respective class in the form `(a,b]` (except the first class, which will be `[a,b]`), extracted from `cut()` with the argument `labels = TRUE`.

Cathes of fish can be equal to zero (and therefore, CPUE), specially in the case of by-catch species[2] [8]. So, it's important to show in the map where these zero catches occured, instead of just occult them. For this, both functions (`mapq()` and `mapy()`) will handle data with zeros properly, so that there will be a class only for the zero with a different symbol in the map. If this is the case for your data, the best (easiest) way to specify the `breaks` argument is to choose the number of intervals. If you wish to specify a numeric vector with the break points, start with a number different of zero because the function will handle that for you.

[2]Which are rare and/or not aimed by fisherman.

Resulting maps can be displayed on the screen (the default), or automatically saved as PNG or PDF file format. Note that if you have a large amount of maps (*i.e.* a long time series), displaying them on the screen may only be usefull if you want to have a previous view of the final output, since for this is used the argument `ask = TRUE` in the `par()` function. Therefore, you will need to press the Enter key for each map to be shown. Alternatively, one can specify `fig = TRUE` to produce one file for each map. Besides the file format, users can also specify the width and height, a name for the figure, and other graphical parameters in the same way as in `par()`. The unit of measure for width and height will depend on the file format you chose (pixels for PNG and inches for PDF). The name for the figure can be any arbitrary character string. This string will be used to compose a name that, for one of the maps using `mapq()`, will be like `map_cpue_2001_1.png` if you made the string `"cpue"` as the name. The string `"map"` is fixed and the numbers are the year and the quarter, respectively.

3 Example usage

In this session, some examples of how functions `mapq()` and `mapy()` can be used are shown. The first example illustrates the use of `mapy()` with the `LL.data.y` database, which contains information of swordfish CPUE landed in the south of Brazil between 2001 and 2005 (only untill 2004 is shown here). The following code was used to produce the maps, and the result is shown in Figure 1.

```
> mapy(year, lat, lon, cpue, breaks=4, type="LL",
      ident.cex=1.5, leg.title="CPUE", leg.cex=1.2,
      fig=T, fig.type="pdf", fig.w=7, fig.h=7,
      fig.name="CPUE_LL", fig.par=list(mar=c(4,2,1,0.5),
      cex.axis=1.4), map.data="worldHires", grid=FALSE,
      resolution=0.5)
```

Note in this code the argument `type = "LL"` was used, what means that the function will use a pre-defined map, which is, in this case, that made by the `mapLL()` function. The last three arguments, `map.data`, `grid` and `resolution`, are arguments of other functions, possible due the presence of the '`...`' argument. This allows the use of arguments from `map()` and `mapLL()` (or `mapBB()`) functions. For example, `map.data` is an argument used in `mapLL()` to specify the use of a high resolution database, and `resolution` is an argument from `maps()` that specifies the refinement to be used (the default value is 1, and the full resolution is 0).

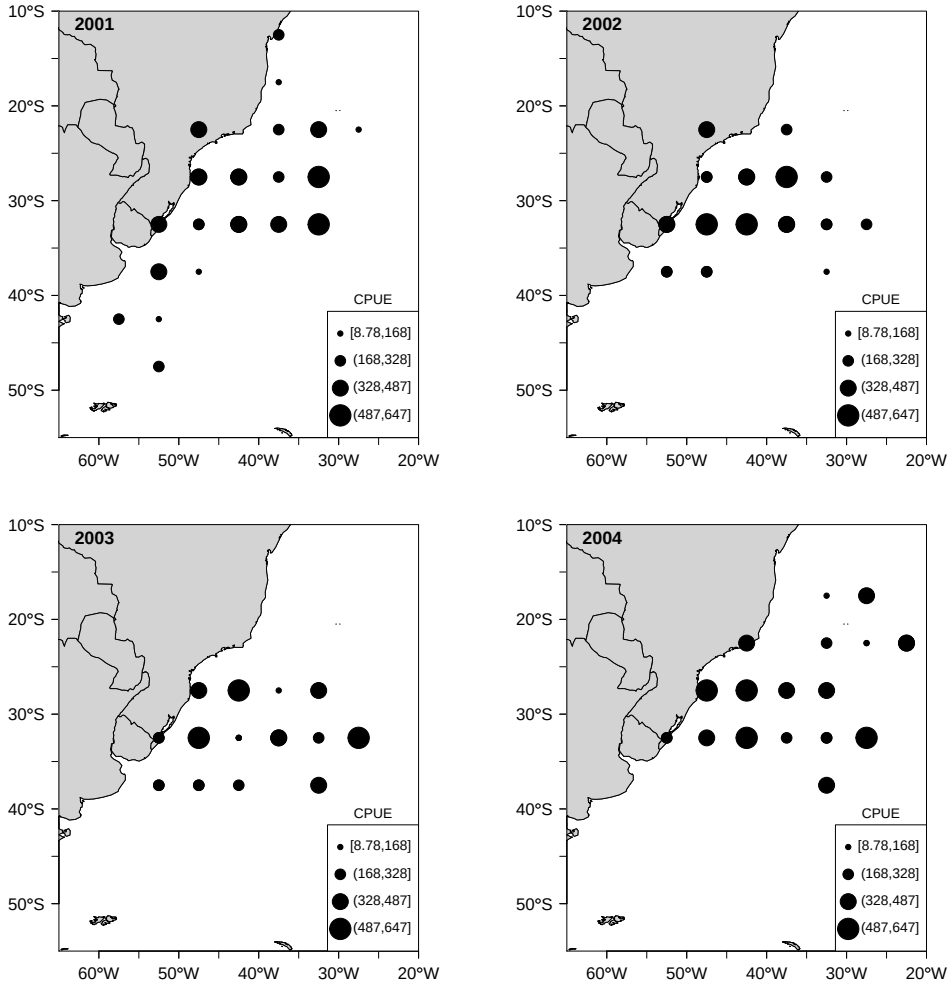

Figure 1: Swordfish CPUE (kg/1000 hooks) for years 2001–2004. Maps were drawn with the `mapy()` function.

The second example shows how users can define their own map boundaries, instead of using a pre-defined map. For this, the `BB.data.yq` database was used, which contains information of skipjack tuna CPUE by year (2001–2002) anq quarter (only 2001 is presented here). Figure 2 shows the result of the code below.

```
> mapq(year, quarter, lat, lon, cpue, breaks=4, ident.cex=1.5,
      xlim=c(-52,-40), ylim=c(-34,-22), majortick=2,
      minortick=1, leg.cex=1.2, fig=T, fig.type="pdf",
      fig.w=7, fig.h=7, fig.name="CPUE_BB",
      fig.par=list(mar=c(4,2,1,0.5),cex.axis=1.4),
```

```
database="worldHires", resolution=0.5, border=0)
```

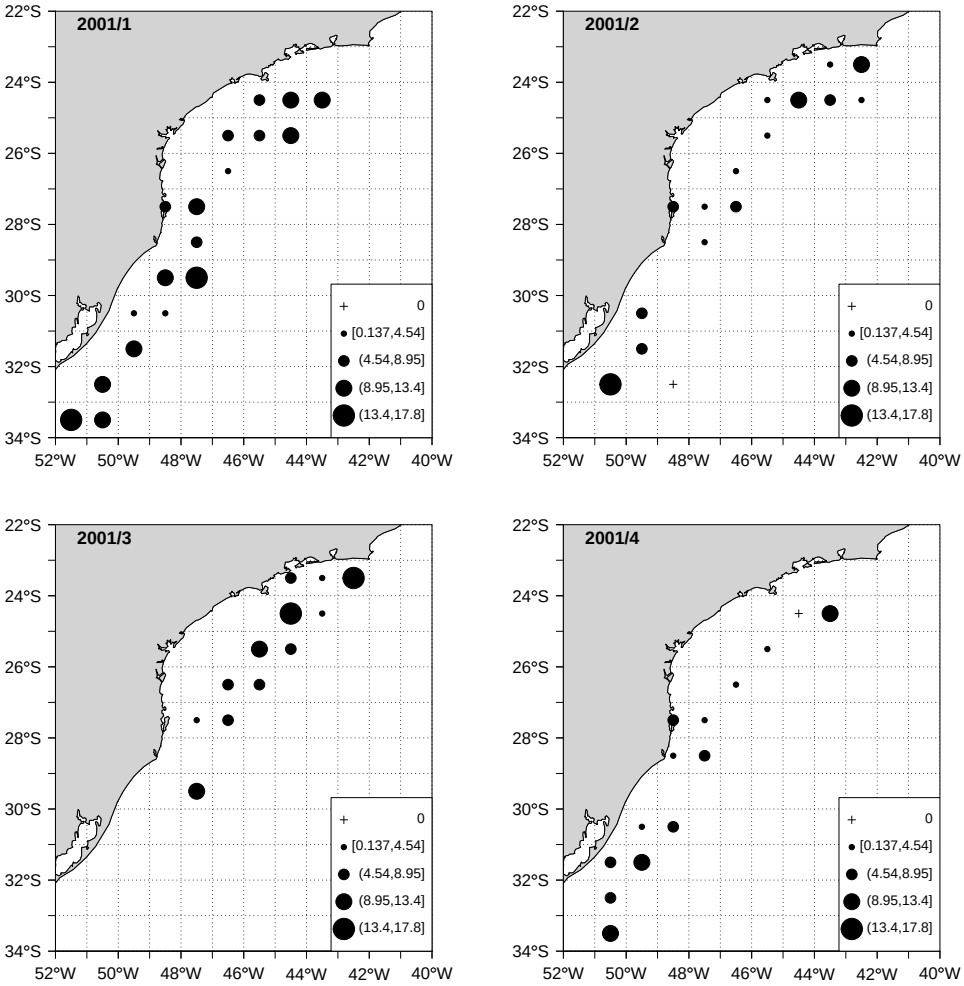

Figure 2: Skipjack tuna CPUE (t/fishing day) for the quarters of 2001. Maps were drawn with the `mapq()` function.

To specify their own map boundaries, users can use the arguments `xlim` (for the extremes of longitude) and `ylim` (for the extremes of latitude). The `majortick` and `minortick` arguments are used to specify the intervals at which the axes ticks should be drawn. Argument `majortick` says at what interval the numbers of latitude/longitude should be displayed, while `minortick` serves to refine the grid of the map. This informations are also used to effectively draw the grid. Note that in a user-defined map, the argument that controls the visualization of the grid is `mapgrid`. In a pre-defined

map, this argument is invalid because functions `mapLL()` and `mapBB()` have their own `grid` argument, which is used for the same purpose (and can be seen in the code for the first example). In a similar way, to specify the use of the `worldHires` database in a user-defined map, one have to use the argument `database`, which is passed to the `map()` function internally. In a pre-defined map, this argument must be the `map.data`, which is used by `mapLL()` and `mapBB()` functions.

Note that in both codes above, the necessary data structure to build the maps is very simple. One only needs a data frame with year (represented as `year`), quarter (only for `mapq()` – `quarter`), latitude (`lat`), longitude (`lon`), and the variable of interest (in this case the `cpue`).

4 Further development

Package FishMaps is fairly recent, so it is in an early stage of development (its current version is 0.2-0). In many points it can be improved. Although effort was made to give users flexibility to build their maps, there are still many things which are internally fixed in the code. This causes that users have no options to control some details. One example is the identifier for the maps, which is still restricted to the upper left corner. Other kinds of annotations to the plots are also not allowed yet.

Solutions for this kind of problems and improvements to the package may be found by different manners. One of them is try to use some of the generic sp class objects. Other is by using the grid graphics model, in a similar way to that explained in chapter 7 of "R Graphics" [9]. For now on, the search to give more flexibility to users and for code improvement will be the objective of development for package FishMaps.

References

[1] J. A. Gulland. *Fish stock assessment: a manual of basic methods.* FAO/Wiley series on food and agriculture. John Wiley & Sons, 1983.

[2] S. Tanimura, C. Kuroiwa, and T. Mizota. Proportional symbol mapping in R. *J. Stat. Soft.*, 15(5):1–7, 2006.

[3] J. T. Schnute, N. M. Boers, and R. Haigh. PBS mapping 2: user's guide. *Can. Tech. Rep. Fish. Aquat. Sci.*, 2549:126 p., 2004.

[4] R. Brownrigg and T. P. Minka. *maps: Draw Geographical Maps*, 2007. R package version 2.0-35.

[5] R. A. Becker and A. R. Wilks. Maps in S. *AT&T Bell Laboratories Statistics Research Report [93.2]*, 1993.

[6] R. Brownrigg. *mapdata: Extra Map Databases*, 2007. R package version 2.0-19.

[7] E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, 2005.

[8] H. A. Andrade. Estimation of the relative abundance of atlantic billfish: effects of three approaches to cope with catches equal to zero. *Col. Vol. Sci. Pap. ICCAT*, 2007.

[9] P. Murrel. *R Graphics*. Computer science and data analysis series. Chapman & Hall/CRC, 2006.