

# PSCF demo

---

## Contents

<b>1</b>	<b>Preliminaries</b>	<b>1</b>
<b>2</b>	<b>Three main objects: trajectories, map, and grid</b>	<b>2</b>
2.1	Trajectories . . . . .	2
2.2	Map . . . . .	2
2.3	Grid . . . . .	3
<b>3</b>	<b>Evaluate map and grid</b>	<b>3</b>
<b>4</b>	<b>Prepare the trajectories</b>	<b>3</b>
<b>5</b>	<b>Plotting trajectories</b>	<b>4</b>
5.1	Examples - VOCALS . . . . .	4
5.2	Additional examples: shiptracks, other projections (not run) . . . . .	6
5.3	The showmap() function . . . . .	7
<b>6</b>	<b>Misc.</b>	<b>7</b>
6.1	Image resolution . . . . .	7
6.2	Data extraction for export/ customization . . . . .	8

---

## 1 Preliminaries

From this point it will be assumed that you have generated trajectory files with HYSPLIT (following the instructions in Readme.pdf) and combine them into a `coords.rda` file. The example `coords.rda` file used here was generated for the VOCALS-REx campaign.

First, load libraries and functions. Be sure that you have the R packages `chron`, `fields`, `maps`, `mapproj`, and `akima` installed - this can be done very easily. For example, while connected to the internet, type at the R prompt:

```
> install.packages("chron", repos = "http://cran.r-project.org")
```

After all libraries have been installed, Begin the program:

```
> invisible(capture.output({  
+   library(chron)  
+   library(fields)
```

```

+   library(maps)
+   library(mapproj)
+   library(akima)
+ })))
> mapf.env <- (if (all(regexpr("mapfunctions", search()) < 0)) attach(NULL,
+   2, name = "mapfunctions") else pos.to.env(grep("mapfunctions",
+   search()))))
> sys.source("functions/pscf_functions.r", mapf.env)
> source("functions/classdef.r")
> options(stringsAsFactors = FALSE)

```

Functions are located in a folder called `functions/`, in `functions.r`. Also located in the same folder is a file called `classdef.r`, which contains definitions for object classes used here.

User inputs - tell us where your files are:

```

> coordsfile <- "outputs/coords_vocals.rda"
> groupfile <- "userinputs/groupfile-example_alcf.txt"

```

Your group file should look like this:

```

> head(read.delim(groupfile, row.names = 1))

```

	Start	End	Group
VX0021	10/21/08 12:03:00	10/21/08 17:32:00	low
VX0022	10/21/08 23:33:00	10/22/08 10:30:00	low
VX0024	10/22/08 11:40:00	10/22/08 21:54:00	high
VX0025	10/22/08 23:51:00	10/23/08 11:16:00	high
VX0026	10/23/08 12:28:00	10/24/08 11:09:00	high
VX0030	10/24/08 12:17:00	10/24/08 23:30:00	high

## 2 Three main objects: trajectories, map, and grid

### 2.1 Trajectories

Read trajectories; shorten to 3 days (optional):

```

> trajectories <- readtrajectories(coordsfile)
> trajectories <- shorten(trajectories, ndays = 3)

```

Randomly sample 1/2 of trajectories for this example (remove this line for production run).

```

> trajectories <- random(trajectories, fraction = 0.5)

```

### 2.2 Map

Define map [longitude (xlim) and latitude (ylim) arguments are optional].

```

> mp <- definemap("world", xlim = c(-110, -50), ylim = c(-60, 5))

```

If map database is "world2", the longitudes have to be transformed (otherwise, leave unchanged).

```

> trajectories <- transform(trajectories, mp)

```

## 2.3 Grid

Define grid. The following line will divide the box containing trajectory endpoints (both latitude and longitude) into 40 even-sized boxes.

```
> xygrid <- definegrid(traj = trajectories, len = 40)
```

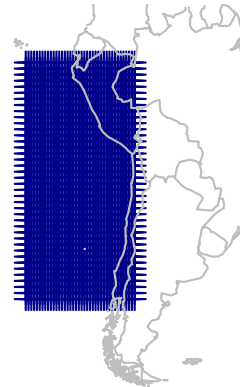
Alternatively, you can specify the grid coordinates directly (not run in this example).

```
> xygrid <- definegrid(longrid = seq(-95, 50, 5), latgrid = seq(35,  
+      93, 3))
```

## 3 Evaluate map and grid

Look at the map boundaries and spacing of grid points overlayed on map; redefine if necessary.

```
> par(mfrow = c(1, 2), mar = rep(1, 4))  
> showmap(mp, gridlines = TRUE)  
> showmap(mp, xygrid)
```



## 4 Prepare the trajectories

Overlay trajectories on the grid [the last argument can be either `identity` to count number of trajectory points (default), or `unique` to count unique trajectories over each grid cell]:

```
> trajectories <- overlay(trajectories, xygrid, identity)
```

(See [reports/identity-unique/summary.pdf](#) for comparison between the two options.)

Read in group file and prepare trajectory object for visualization; attach data to 'trajectories' object. If new groups are desired, rerun from this point on (do not have to reload trajectories).

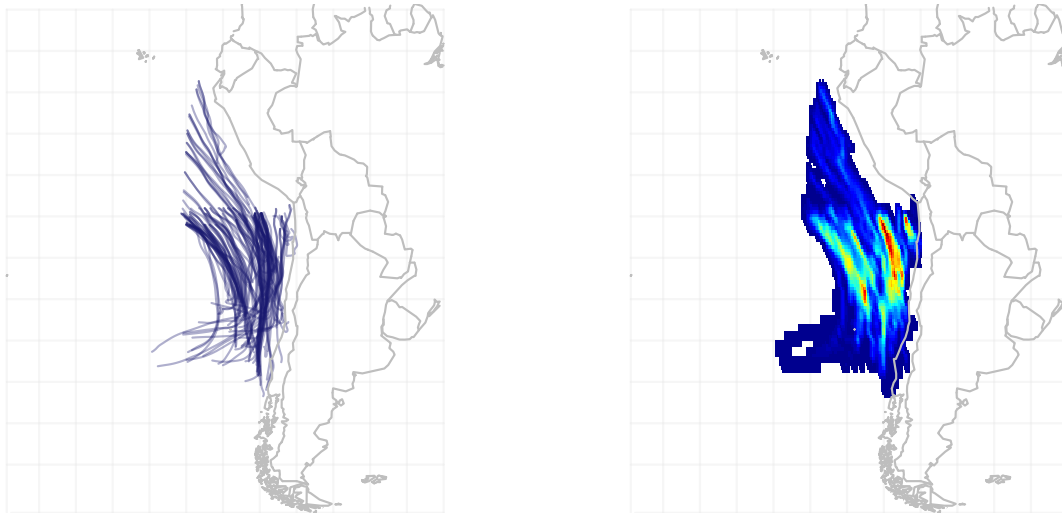
```
> groups(trajectories) <- readgroups(groupfile)
> trajectories <- prepareforvis(trajectories, xygrid)
```

## 5 Plotting trajectories

### 5.1 Examples - VOCALS

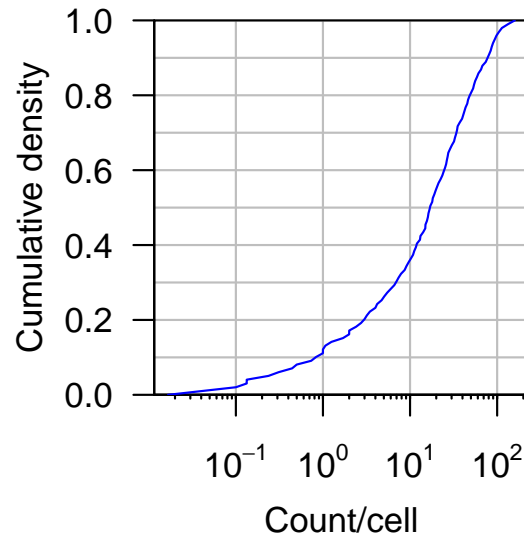
Show all trajectories (as 'spaghetti' and 'density'):

```
> par(mfrow = c(1, 2), mar = rep(1, 4))
> showmap(mp, trajectories, type = "spaghetti", groupindex = 0,
+       gridlines = TRUE)
> showmap(mp, trajectories, type = "density", groupindex = 0, gridlines = TRUE)
```



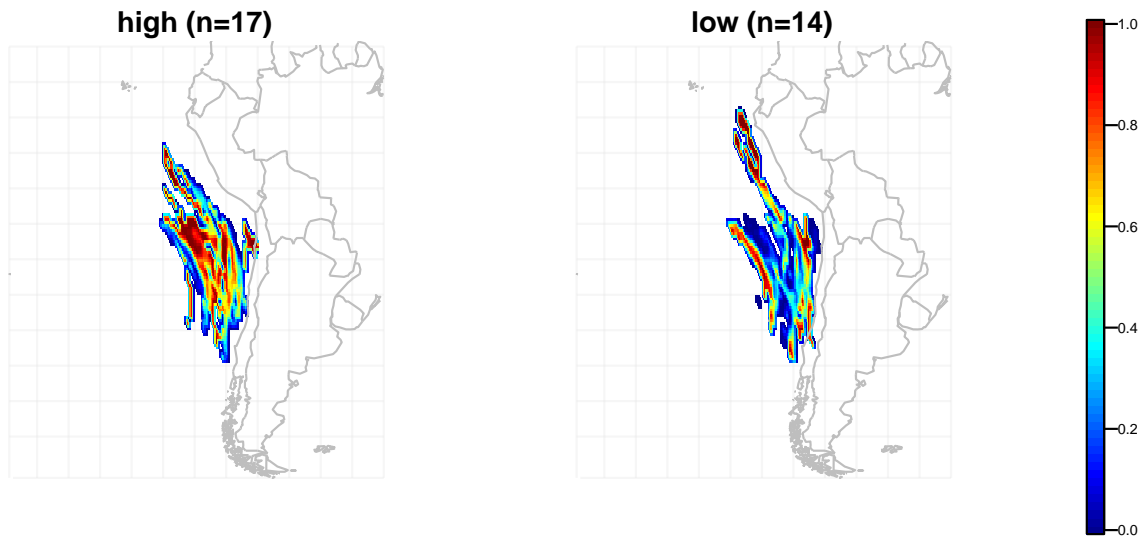
Next, visualize number of trajectories/ points per cell.

```
> par(mfrow = c(1, 1), mar = c(4.5, 4.5, 1.5, 1.5), mgp = c(2.5,
+       1, 0), pty = "s")
> cumuldensp(trajectories)
```



We will only include grid cells for which the number of trajectories/points (count/cell) are above 10, approximately (values  $<1$  indicate cells in which weights ( $\rightarrow$  fraction of hour) were  $<1$ ), so we pass the argument `threshold=0.4` to the `showmap()` function. The PSCF plot is created with the following code. Export with graphics desired device (e.g., `pdf()`, `png()`).

```
> ngr <- length(groups(trjectories))
> layout(matrix(1:(ngr + 1), nrow = 1), width = c(rep(5, ngr),
+ 1))
> par(mar = c(1, 1, 1.5, 1), mgp = c(1, 1, 0), lend = 3, pty = "s")
> for (i in 1:ngr) {
+   showmap(mp, trajectories, threshold = 0.4, type = "pscf",
+     groupindex = i, gridlines = TRUE)
+   title(main = grpname(trjectories, i), cex.main = 1.2)
+ }
> addlegend(m1 = 2, m2 = 1.5, m3 = 2, m4 = 2, mgp = c(2, 0.5, 0),
+   cex.axis = 0.6)
```



## 5.2 Additional examples: shiptracks, other projections (not run)

We can also add ship tracks. Calculate it from the originating point for each of the back trajectories:

```
> shiptrack <- lapply(colnames(coords(trajectories))[2:1], function(x,
+   y) sapply(y[, x], `[`, 1), coords(trajectories))
```

Plot ship tracks. For an orthographic projection, we don't need x- and y-limits. To get rid of them, redefine the map without passing values to `xlim` and `ylim` parameters.

```
> mp <- definemap("world")
```

Make the plot:

```
> showmap(mp, shiptrack = shiptrack, projection = "orthographic",
+   orientation = c(90, 0, -12.5))
```

Map in stereographic projection:

```
> par(mfrow = c(1, 2), mar = rep(1, 4))
> showmap(mp, gridlines = TRUE)
> showmap(mp, xygrid)
```

PSCF map with shiptracks in stereographic projection:

```
> ngr <- length(groups(trajectories))
> layout(matrix(1:(ngr + 1), , nrow = 1), width = c(rep(5, ngr),
+   1))
> par(mar = c(1, 1, 1.5, 1), mgp = c(1, 1, 0), lend = 3, pty = "s")
> for (i in 1:ngr) {
+   showmap(mp, trajectories, shiptrack = shiptrack, type = "pscf",
```

```
+      gridlines = TRUE, groupindex = i, projection = "stereographic")
+      title(main = grpname(trajectories, i), cex.main = 1.2)
+    }
> addlegend(m1 = 2, m2 = 1.5, m3 = 2, m4 = 2, mgp = c(2, 0.5, 0),
+      cex.axis = 0.6)
```

### 5.3 The showmap() function

Summary of arguments to `showmap`:

Argument	Possible values
<code>mobj</code>	'Map' object (*required*)
<code>obj1</code>	'XYGrid' or 'Traj' object
<code>type</code>	'diagnose', 'spaghetti', 'density', or 'pscf' (character)
<code>gridlines</code>	TRUE or FALSE (logical)
<code>groupindex</code>	1,2,...n or 0 for 'spaghetti' or 'density' (integer)
<code>shiptrack</code>	list with longitude and latitude components (list)
<code>threshold</code>	exclude grid cells containing less than <b>threshold</b> quantile (percentile /100) of trajectory counts.
<code>...</code>	projection and parameters to be passed to <code>mapproject</code>

The 'Map' object is the only required argument. Not specifying a value for `projection` will give you a rectangular projection.

`showmap()` is intended to be an exploratory tool. Edit function `mpj()` in `functions/classdefs.r` if further customizations are desired.

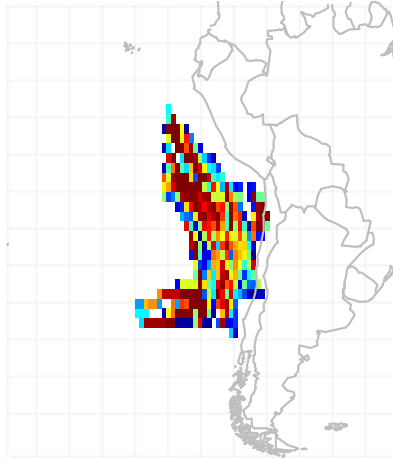
## 6 Misc.

### 6.1 Image resolution

Increasing/decreasing image resolution using the `ninterp` argument to `showmap()` - compare with left PSCF figure - 'high' case. In this case I decreased the resolution to contrast the visual difference in appearance.

```
> showmap(mp, trajectories, type = "pscf", gridlines = TRUE, groupindex = 1,
+      ninterp = 30)
> title(main = grpname(trajectories, 1), cex.main = 1.2)
```

## high (n=17)



### 6.2 Data extraction for export/ customization

Use `extract()` on object `trajectories`. (You can also pass a `threshold` argument to `extract()`). Note: matrix resolution is defined by `xygrid` using `definegrid()`; `ninterp` is only used for controlling resolution for visualization using `showmap()`.

```
> output <- extract(trajectories, type = "pscf", groupindex = 1,  
+   threshold = 0.4)
```

You can export the data using `write()`:

```
> write(output$x, file = "xvalues.txt", ncol = 1)  
> write(output$y, file = "yvalues.txt", ncol = 1)  
> write(t(output$z), file = "zvalues.txt", ncol = ncol(output$z))
```

Or plot it in R using `image()` (which is called internally by `showmap()`).

```
> image(output, col = grey.colors(64), asp = 1)
```



