## Alternative visualization tools: Monsoon in the Indian subcontinent

So far we have shown plotting examples using the trellis plots generated by the spplot `method`. In this examples we show alternative plotting options using more standard R plotting functions for gridded data. To this aim, we load the precipitation data of 1997 for the lead month 1 forecast over the Indian subcontinent, considering the monsoon season from June to September:

```
monsoon <- loadSeasonalForecast("CFS", var="tp", members=16, lonLim=c(65,92), latLim=c(5,37), season=6:9, years=1997)
```

The georeferencing of the data is stored as a `SpatialGrid`, which has several convenient attributes for an effective description of a gridded field, including the possibility of defining a coordinate reference system, highly useful -sometimes indispensable- for many geospatial operations. Plotting objects inheriting from this class or related classes (e.g. `SpatialGridDataFrame`) is straightforward using the `spplot` methods, as described in the previous example?, but it is not directly usable as input for other plotting methods in R. Next, we present some typical plotting functions and how to extract the spatial coordinates in a suitable format for plotting.

The total precipitation field is calculated as the total accumulated precipitation during the selected period for each model grid cell:

```
tp <- colSums(monsoon$MemberData$Member_16)
```

And this is the classical way of displaying the data using the `spplot` method:

```
sgdf <- SpatialGridDataFrame(monsoon$LonLatCoords, as.data.frame(tp))
data(world_map)
wl <- as(world_map, "SpatialLines")
wlines <- list("sp.lines", wl)
spplot(sgdf, scales = list(draw = TRUE), sp.layout = list(wlines), col.regions = rev(topo.colors(41)))
```

It is also straightforward to represent the precipitation using contour lines:

```
spplot(sgdf, scales = list(draw = TRUE), contour = TRUE, col = "red", col.regions = rev(topo.colors(41)))
```

The help files of functions `image` and `contour` for instance, indicate the type of data structure required for displaying three-dimensional or spatial data (*images*) by many R standard functions. Essentially, this is a list of elements specifying the x and y positions of the elements to be displayed as a grid, being the field z a matrix of values whose positions coincide with those of the x and y elements.

As an illustration, the following function produces such a list from an input `SpatialGridDataFrame`, by indicating also the corresponding data column to be represented, as in the argument `zcol` of `spplot`. This function takes care of the appropriate ordering of the data for spatial consistency. Note that by default, if `zcol` is omitted, the function will represent the first column of the data slot.

```
sgdf2xyz <- function(sgdf, zcol = 1) {
     coords <- coordinates(sgdf)
     z <- slot(sgdf, "data")[ ,zcol]
     aux <- cbind(coords, z)
     aux.ordered <- aux[order(aux[ ,2], aux[ ,1]), ]
     x <- unique(aux.ordered[ ,1])
     y <- unique(aux.ordered[ ,2])
     z <- t(matrix(aux.ordered[ ,3], nrow = length(y), ncol = length(x), byrow = TRUE))
     xyz.list <- list("x" = x, "y" = y, "z" = z)
     return(xyz.list)
}
```

We create the xyz object for data visualization using several basic R functions:

```
xyz <- sgdf2xyz(sgdf)
```

### `image` function

```
image(xyz, asp = 1, col = rev(topo.colors(21)))
lines(wl)
```

### `contour` function

`contour` can be used alone or in combination with other plots by setting the argument `add = TRUE`

```
par(mfrow = c(1,2))
contour(xyz, levels=seq(0,4000,200), labcex = 1.2)
image(xyz, col = terrain.colors(21))
contour(xyz, col = "blue", add = TRUE)
# Use dev.off() To restore the original par settings
```