Spatial mapping functions

Emma Cavan

08/10/2018

Making maps of ocean satellite/model data using raster images

First install libraries

```
library(squash)
library(maps)
library(mapdata)
library(mapproj)
library(ggmap)
library(viridis)
library(lattice)
library(latticeExtra)
library(ggplot2)
library(rasterVis)
library(raster)
library(sp)
library(maptools)
library(magrittr)
library(stringr)
library(rworldmap)
library(gridExtra)
```

Function 1. Simple map

Make the function called raster_plot. The function has 3 input terms. 1) *x*, your data frame which must have columns ordered as *value*, *longitude*, *latitude*, 2) main *title* for the plot and 3) the *key* for the colour scale. Within the function *xy* is a data frame containing just the lon and lat, which is the last 2 columns in *x*. The next 4 lines convert from a data frame to a raster image, the final structure needed to make the levelplot. Here the viridis colour pallette is used and the whole globe (between 80 °N and -80 °S) plotted. You can set the area you want to plot using xlim and ylim in first line of the levelplot function.

```
raster_plot<- function(x,title,key){
  xy<-x[,c(2,3)]  # select lon and lat and make new df
  spdf.2 <- SpatialPointsDataFrame(coords = xy, data = x,
                  proj4string = CRS("+proj=longlat +datum=WGS84
                                    +ellps=WGS84 +towgs84=0,0,0"))
  pixels.2 <- SpatialPixelsDataFrame(spdf.2, tolerance = 0.914959, spdf.2@data)
  gridded(pixels.2) = TRUE
  ipcc.t.100<-raster(pixels.2) # turn into raster
  levelplot(ipcc.t.100, xlim=c(-180,180), ylim=c(-80,80),margin=FALSE,
            xlab=list("",cex=1),ylab=list("",cex=1),
            col.regions=viridis(300) ,
            main=title,
            at=seq(min(x[,1]), max(x[,1]), length=70),
            colorkey=list(at=seq(min(x[,1]), max(x[,1]), length=70),
                          labels=list(at=key),
```

```
                          labels=key),
            panel = function(x, y, ...) {
              panel.levelplot(x, y,  ...)
              mp <- map("world", plot = FALSE, fill=TRUE,interior = FALSE,bg="yellow")
              lpolygon(mp$x, mp$y, fill="black", col="black")}
  )
}
```

Now the function is setup, we need to create the data frame (x), the key and the title of the plot.

First read in your data. I have provided the data (mean primary production, PP, 2003 - 2016) used in this example on my Github site (www.github.com/e-cavan). The orginial satellite PP data used to make this data set was downloaded from https://www.science.oregonstate.edu/ocean.productivity/.

```
PP <- read.csv('~/PP_example.csv', header=T)
```

Make sure the variable you want to plot is in the first column (here PP), then the longitude and latitude. In the example data set I've provided columns are in the correct order, but here is some code you could use to reorder it. Under select=c() you list the columns in the order that you want, by column name or number.

```
x <- subset(PP, select=c(1,2,3)) # or x <- subset(PP,select=c('PP','lon','lat))
```

Write the title. VGPM is the algorithm used to convert raw satellite data (chlorophyll, temperature etc. to primary production) by Oregon State University.

```
title <- expression(paste('PP (g C m'^'-2'*' yr'^'-1'*') 2003-2016 (VGPM)'))
```

Show the min and max values for the key

```
summary(x[,1])
```

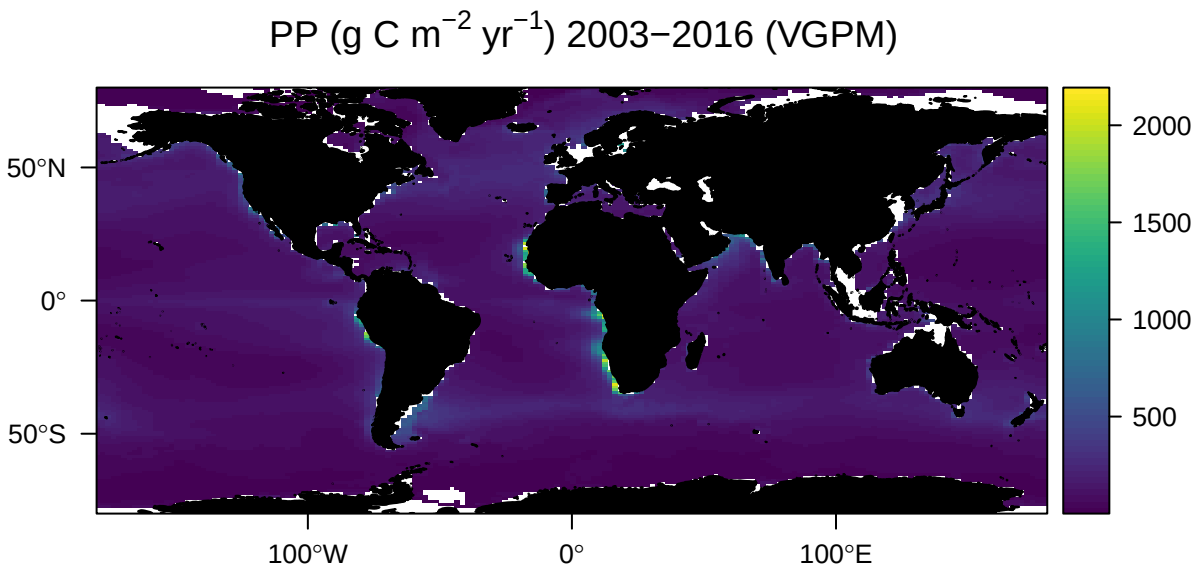
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    0.4329   61.6892  102.7806  124.3098  158.4481 2194.1682
```

Form key (legend) for the colour scale

```
key <- c(500, 1000, 1500, 2000)
```

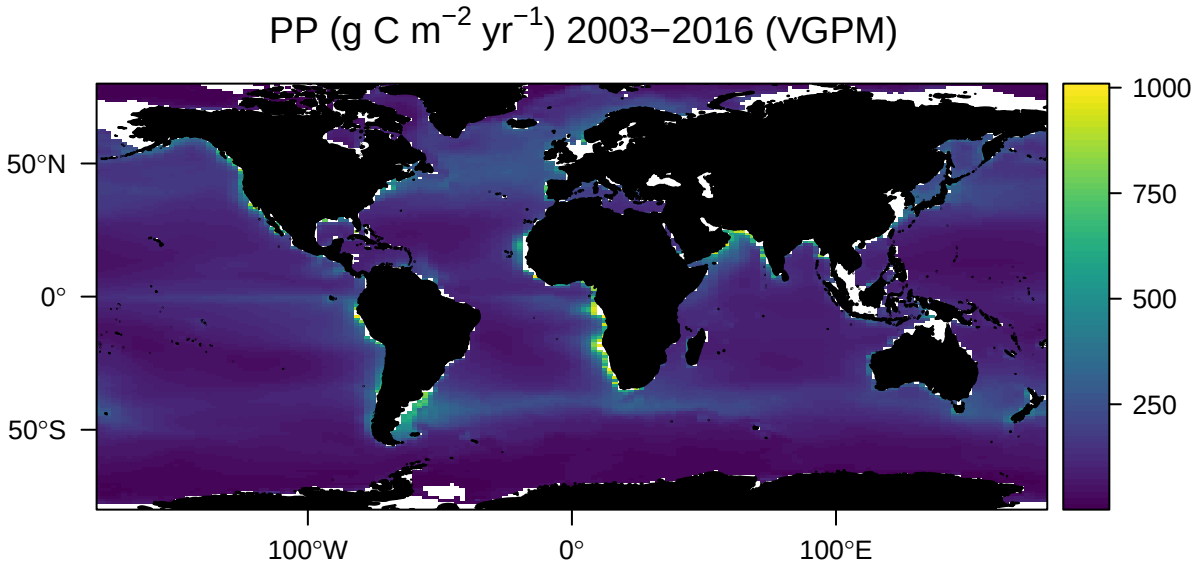
Finally run the function to make your map!

```
raster_plot(x, title, key)
```



Let's examine the plot. There are few very high values (> 1000 gC m-2 yr-1) around the coasts, these cloud the other data and here I'm interested in ocean open PP. Lets remove them.

```
x <- x[x[,1] < 1010,]
key <- c(250, 500, 750, 1000)
raster_plot(x, title, key)
```



Now there are still some very high values in this plot, so to help the viewer see detail in the lower values we can plot the log10(x) of primary production. This will involve another term in our function to produce a different key.

Function 2. Log plot.

```
raster_plot_log<- function(x,title, logkey, key){
  xy<-x[,c(2,3)]
  spdf.2 <- SpatialPointsDataFrame(coords = xy, data = x,
                proj4string = CRS("+proj=longlat +datum=WGS84 +
                                  ellps=WGS84 +towgs84=0,0,0"))
  pixels.2 <- SpatialPixelsDataFrame(spdf.2, tolerance = 0.914959, spdf.2@data)
  gridded(pixels.2) = TRUE
  ipcc.t.100<-raster(pixels.2)
   levelplot(ipcc.t.100, xlim=c(-180,180), ylim=c(-80,80),margin=FALSE,
             xlab=list("",cex=1),ylab=list("",cex=1),
             col.regions=viridis(100) , zscaleLog=10,
             main=title,
             at=seq(min(log10(x[,1])), max(log10(x[,1])), length=70),
             colorkey=list(at=seq(min(log10(x[,1])), max(log10(x[,1])), length=70),
                           labels=list(at=logkey,
                                       labels=key)),
             panel = function(x, y, ...) {
               panel.levelplot(x, y,  ...)
               mp <- map("world", plot = FALSE, fill=TRUE,interior = FALSE,bg="yellow")
               lpolygon(mp$x, mp$y, fill="black", col="black")}
  )
}
```

Determine the min and max values for the key

```
summary(x[,1])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    0.4329   61.6170  102.5979  121.2398  157.8941 1009.5559
```

Set the key for the colour scale. This sets the values to be shown on the color scale.

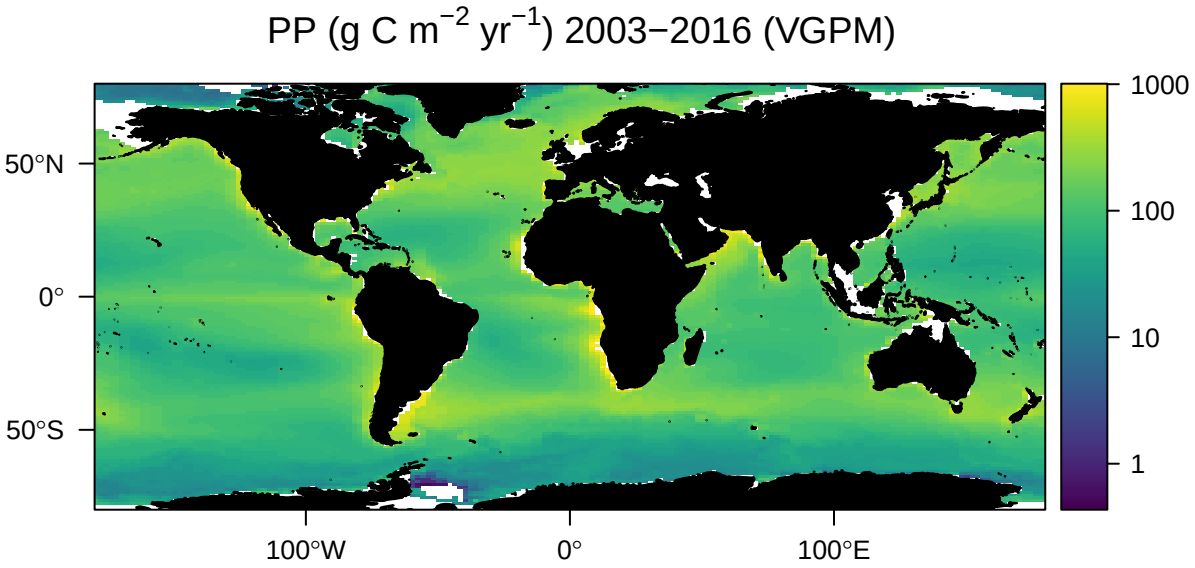
```
key <- c(1,10,100, 1000)
```

Set the logkey term. This sets the *actual* values in the colour scale.

```
logkey <- log10(c(1,10,100, 1000))
```

Finally run the function!

```
raster_plot_log(x, title,logkey, key)
```



That looks much better!

Function 3. Relative change

One final thing we might need to do is to plot relative increases or decreases (e.g. changes in temperature). For this we might want where there is no change in a value over time to be 0 (white), increases positive (red) and decreases negative (blue).

I will make some dummy data from this primary production data. To do this I shall just normalise the data, which will give us both postive and negative values.

```
tmp <- x[,1]             # make temporary vector
mean_tmp <- mean(tmp)    # compute mean
sd_tmp <- sd(tmp)        # compute standard deviation
norm <- (tmp - mean_tmp) / sd_tmp    # normalise, mean = 0
```

Now we have the data, let's look at it and see the min and max values

```
summary(norm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.3116 -0.6473 -0.2024  0.0000  0.3980  9.6447
```

Build function. The changes to this one is the *cols* function used to set *cols.regions*. This is created based on the min and max values and where they change from positive to negative. Here I have made a colour pallette which follows from blue - white - red.

```
raster_plot_delta<- function(x_new,title,key){
  xy<-x_new[,c(2,3)]
  spdf.2 <- SpatialPointsDataFrame(coords = xy, data = x_new,
                 proj4string = CRS("+proj=longlat +datum=WGS84 +
                                   ellps=WGS84 +towgs84=0,0,0"))
  pixels.2 <- SpatialPixelsDataFrame(spdf.2, tolerance = 0.914959, spdf.2@data)
  gridded(pixels.2) = TRUE
  ipcc.t.100<-raster(pixels.2)
  intervals <- seq(min(x_new[,1]),max(x_new[,1]),length=100)
  cols <- c(colorRampPalette(c( "dodgerblue","white"))(length(intervals[intervals < 0])),
            colorRampPalette(c("white", "firebrick"))(length(intervals[intervals > 0])))
   levelplot(ipcc.t.100, xlim=c(-180,180), ylim=c(-80,80),margin=FALSE,
            xlab=list("",cex=1),ylab=list("",cex=1),
            col.regions=cols ,
            main=title,
            at=seq(min(x_new[,1]), max(x_new[,1]), length=100),
            colorkey=list(at=seq(min(x_new[,1]), max(x_new[,1]), length=100),
                          labels=list(at=key,
                                      labels=key)),
            panel = function(x, y, ...) {
              panel.levelplot(x, y,  ...)
              mp <- map("world", plot = FALSE, fill=TRUE,interior = FALSE,bg="yellow")
              lpolygon(mp$x, mp$y, fill="black", col="black")}
  )
}
```

Make data frame with longitude and latitude

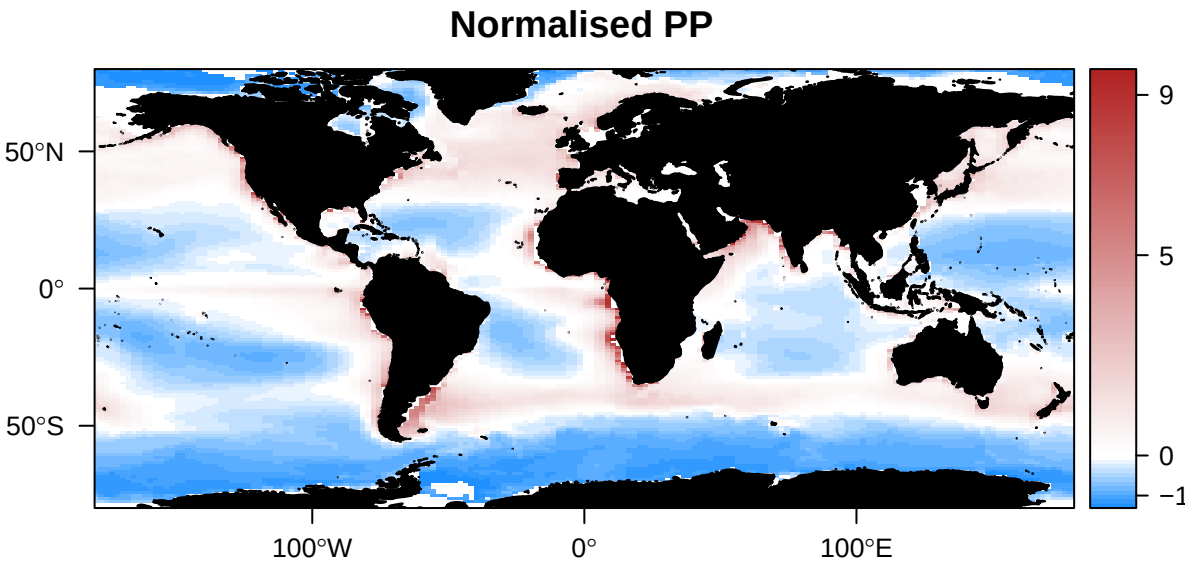
```
x_new <- as.data.frame(cbind(norm, x[,2], x[,3]))
```

Set the title and key.

```
title <- 'Normalised PP'
key <- c(-1, 0, 5, 9)
```

Run the function!

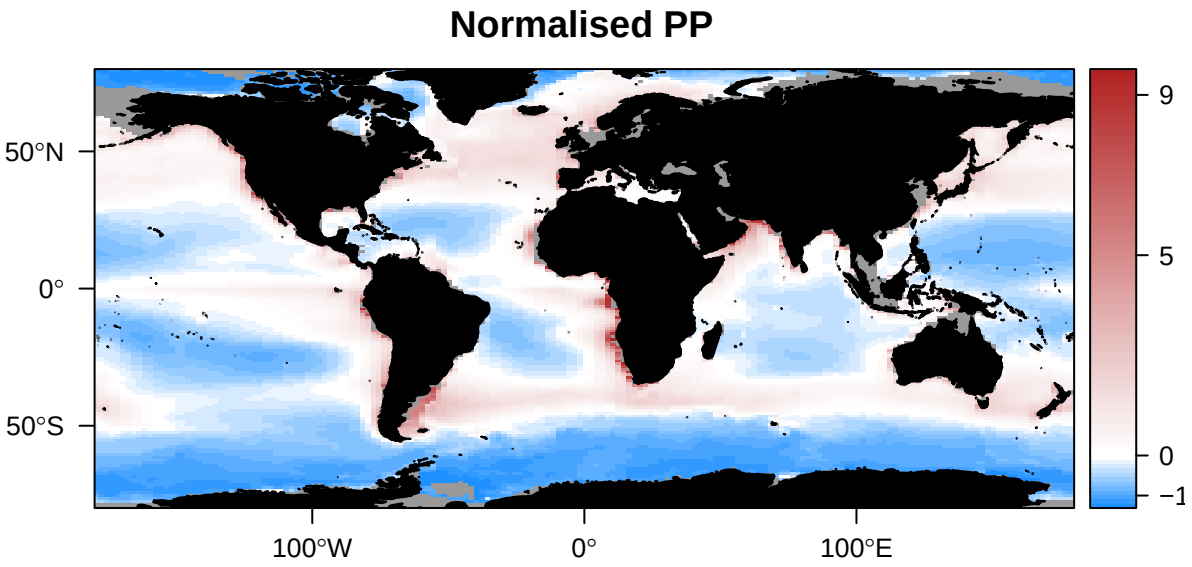
```
raster_plot_delta(x_new, title, key)
```



As we removed high values there are pixels around the coasts where there is no data that appears white, the same colour as the mean PP, 0. Where there is no data, the colour should be changed (e.g. to grey), otherwise the map can be interpreted that around the coasts normalised PP = 0 or the mean, when actually PP is extremely high. If we were using real directional change data this would be interpreted as there is no change in PP around the coasts. This can be altered in the function by adding panel.fill, changing the background (no data) to grey. Alternatively you could change the colour pallette to blue - grey - red.

```
raster_plot_delta<- function(x_new,title,key){
  xy<-x_new[,c(2,3)]
  spdf.2 <- SpatialPointsDataFrame(coords = xy, data = x_new,
                  proj4string = CRS("+proj=longlat +datum=WGS84 +
                                    ellps=WGS84 +towgs84=0,0,0"))
  pixels.2 <- SpatialPixelsDataFrame(spdf.2, tolerance = 0.914959, spdf.2@data)
  gridded(pixels.2) = TRUE
  ipcc.t.100<-raster(pixels.2)
  intervals <- seq(min(x_new[,1]),max(x_new[,1]),length=100)
  cols <- c(colorRampPalette(c( "dodgerblue","white"))(length(intervals[intervals < 0])),
            colorRampPalette(c("white", "firebrick"))(length(intervals[intervals > 0])))
   levelplot(ipcc.t.100, xlim=c(-180,180), ylim=c(-80,80),margin=FALSE,
            xlab=list("",cex=1),ylab=list("",cex=1),
            col.regions=cols ,
            main=title,
            at=seq(min(x_new[,1]), max(x_new[,1]), length=100),
            colorkey=list(at=seq(min(x_new[,1]), max(x_new[,1]), length=100),
                          labels=list(at=key,
                                      labels=key)),
            panel = function(x, y, ...) {
              panel.fill(col='grey60')
              panel.levelplot(x, y,  ...)
              mp <- map("world", plot = FALSE, fill=TRUE,interior = FALSE,bg="yellow")
              lpolygon(mp$x, mp$y, fill="black", col="black")}
  )
}


raster_plot_delta(x_new, title, key)
```


Normalised PP
50°N
0°
50°S
100°W
0°
100°E
9
5
0
−1