Bio Stats II : Lab 7

Gavin Fay

03/09/2016

Lab schedule

1/27: Introduction to R and R Studio, working with data
2/03: Intro to plotting, manipulating data
2/10: Probability, linear modeling
2/17: Programming practices, conditional statements
2/24: Creating functions, debugging
3/02: Permutation analysis
3/09: Advanced plotting

Recommended reading

TBD

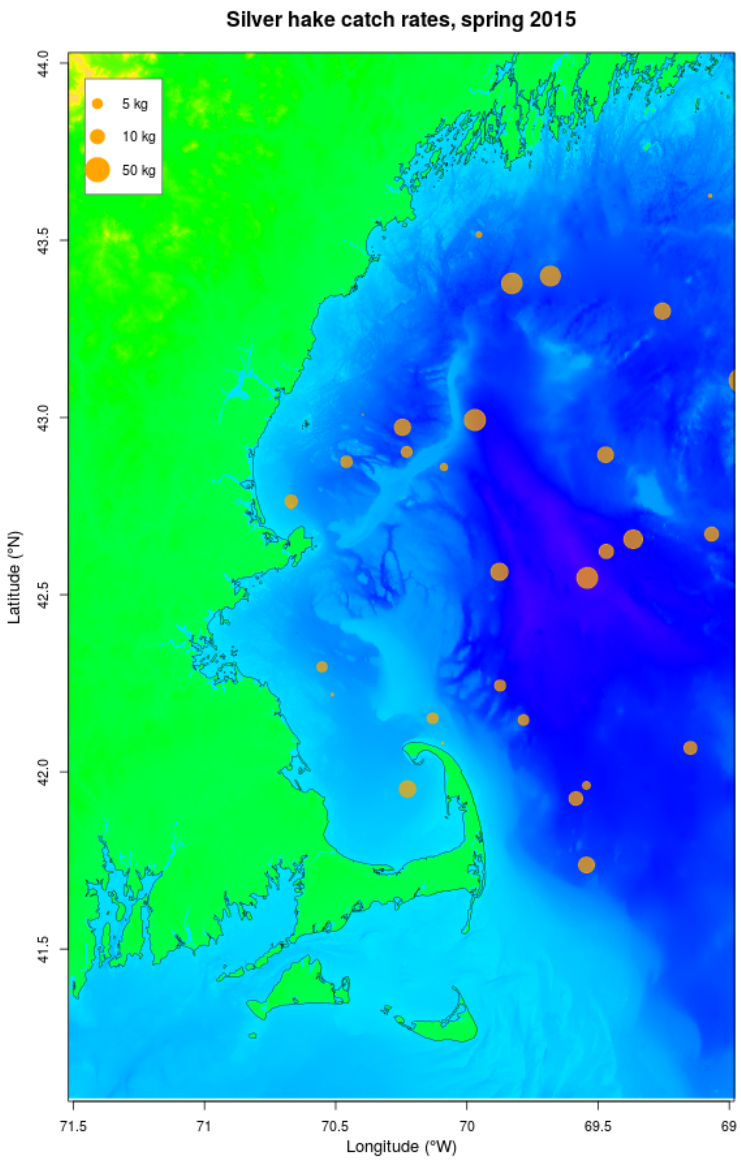
Some great tips on plotting with `R`.
`https://www.datacamp.com/community/tutorials/15-questions-about-r-plots`

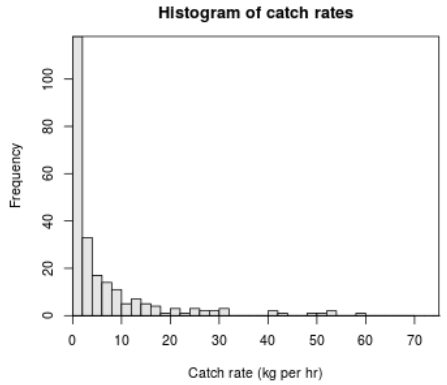
Tufte ER (2001) The visual display of quantitative information. Second edition. Graphics Press, Cheshire, Connecticut

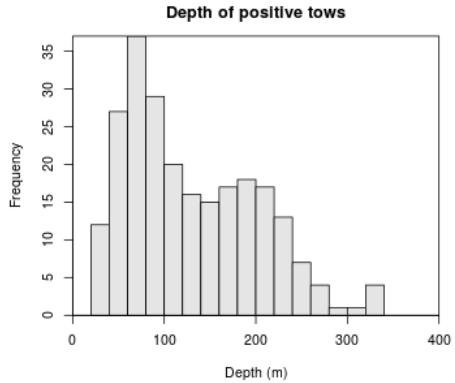
Additional reading: Wilkinson, The Grammar of Graphics. Springer. (grammar of graphics is the gg in `ggplot`...)
`ggplot2` visual user guide

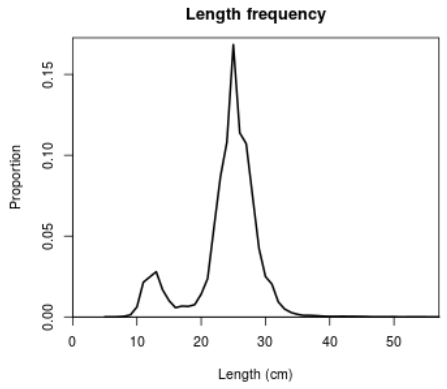
Tutorial to create interactive R maps using `leaflets`.

Todays exercise





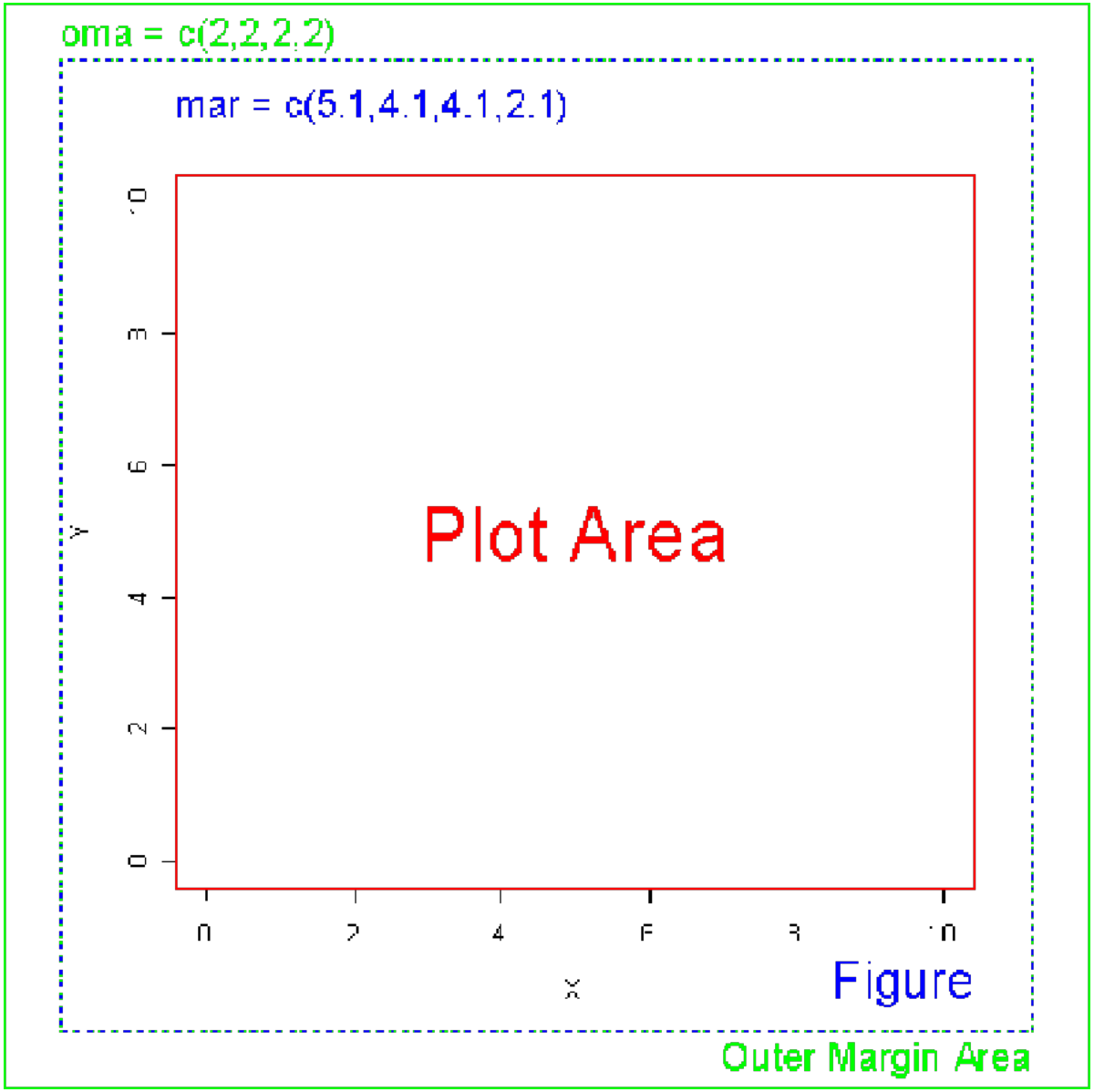




Outline

- Multipanel plots
- Plotting 3D data, images & contours
- NetCDF files
- Maps
- Adding polygons to plots

Plot margins



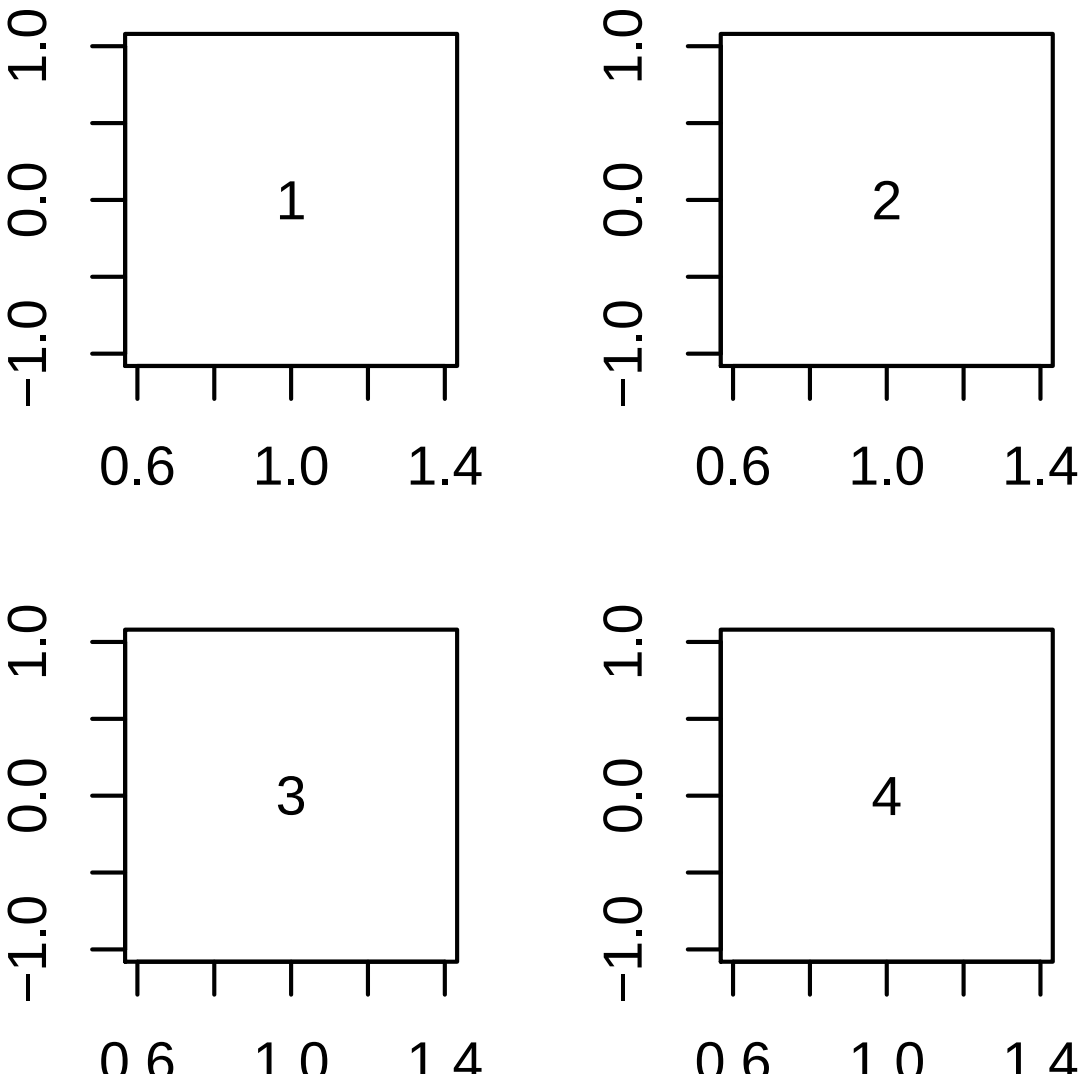
Multipanel plots

`par()` options `mfrow` and `mfcol`.

```
>   par(mfrow=c(2,2))  #creates a 2 x 2 panel layout.
>   #plots are added by row.
>   par(mfcol=c(2,2))  #creates a 2 x 2 panel layout.
>   #plots are added by column.
>   for (i in 1:4) plot(0,pch=as.character(i))
```

Note we can (and should) remove unwanted whitespace and redundant axes with the options we learned in lab 2.

```
>   par(mfrow=c(2,2), mar=c(2,2,2,2))
>   #creates a 2 x 2 panel layout.
>   #plots are added by row.
>   for (i in 1:4) plot(0,pch=as.character(i))
```

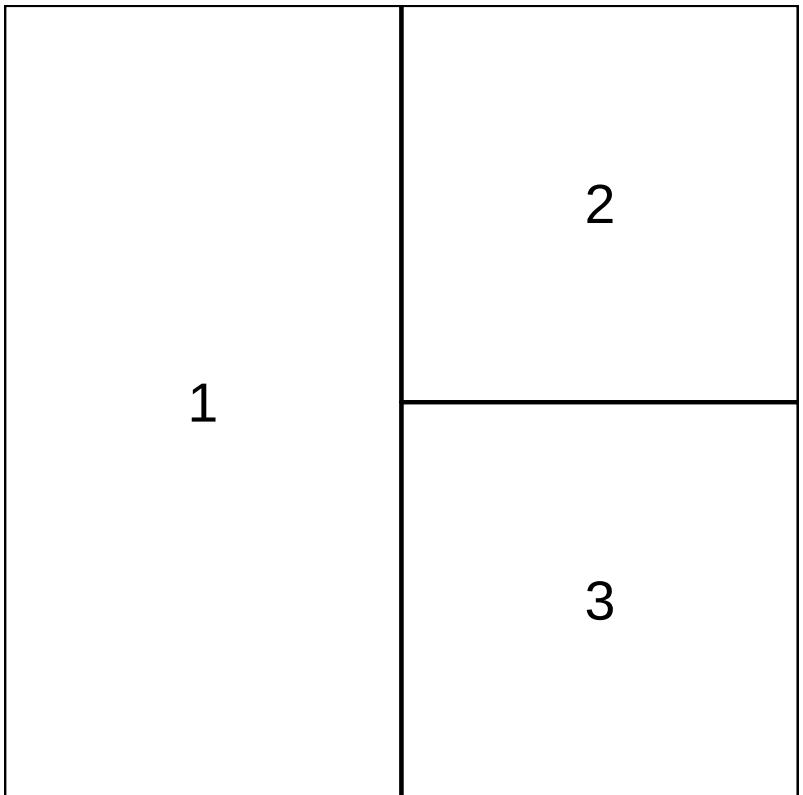


Multipanel plots using `layout()`

The function `layout()` can also be used to arrange plots.
- allows for considerable flexibility.

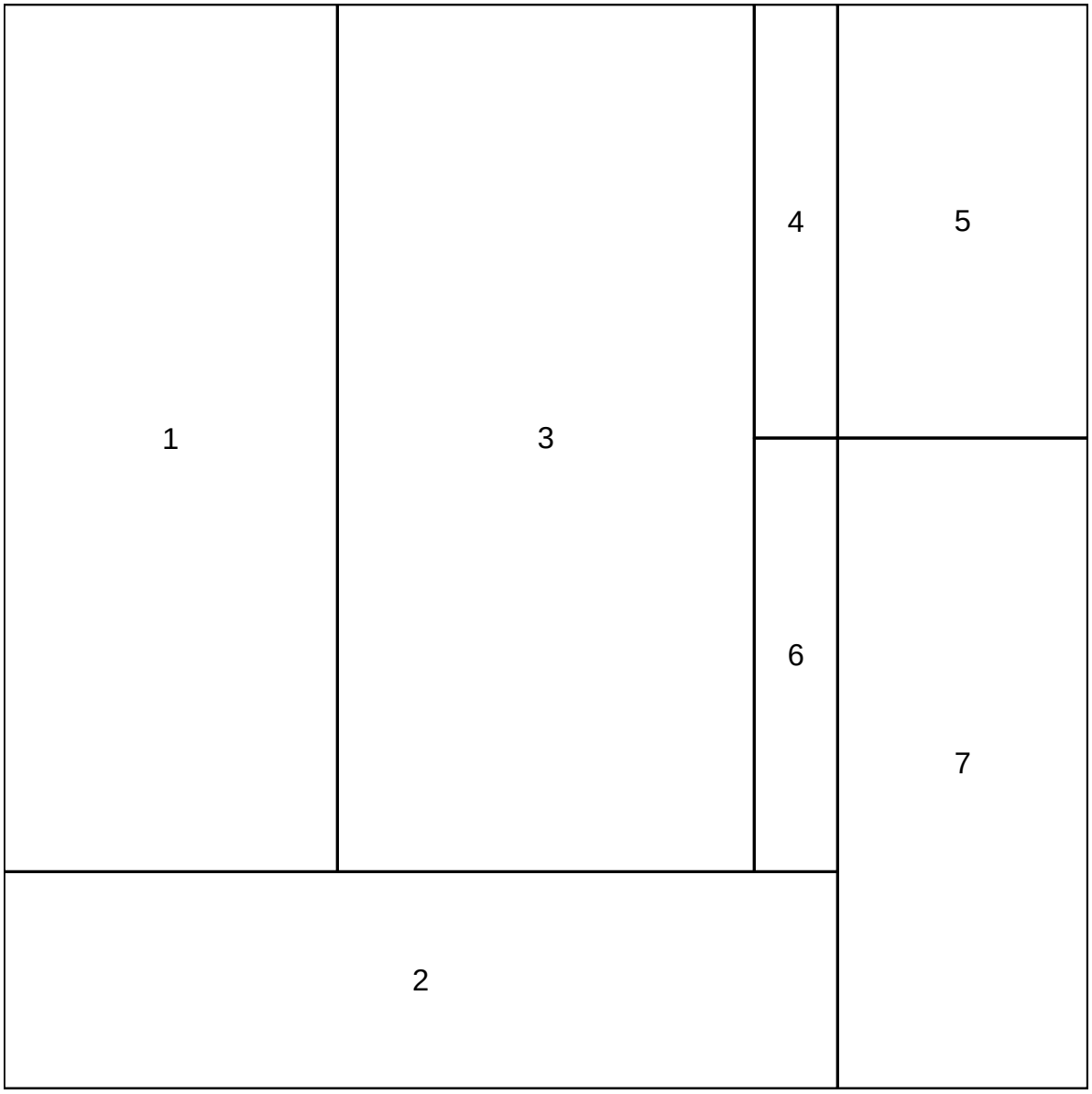
e.g. create plot layout with two columns, with 1 plot in the left column and 2 plots in the right.

```
>     layout(matrix(c(1,2,1,3), 2, 2, byrow = TRUE))
>     layout.show(3)
```

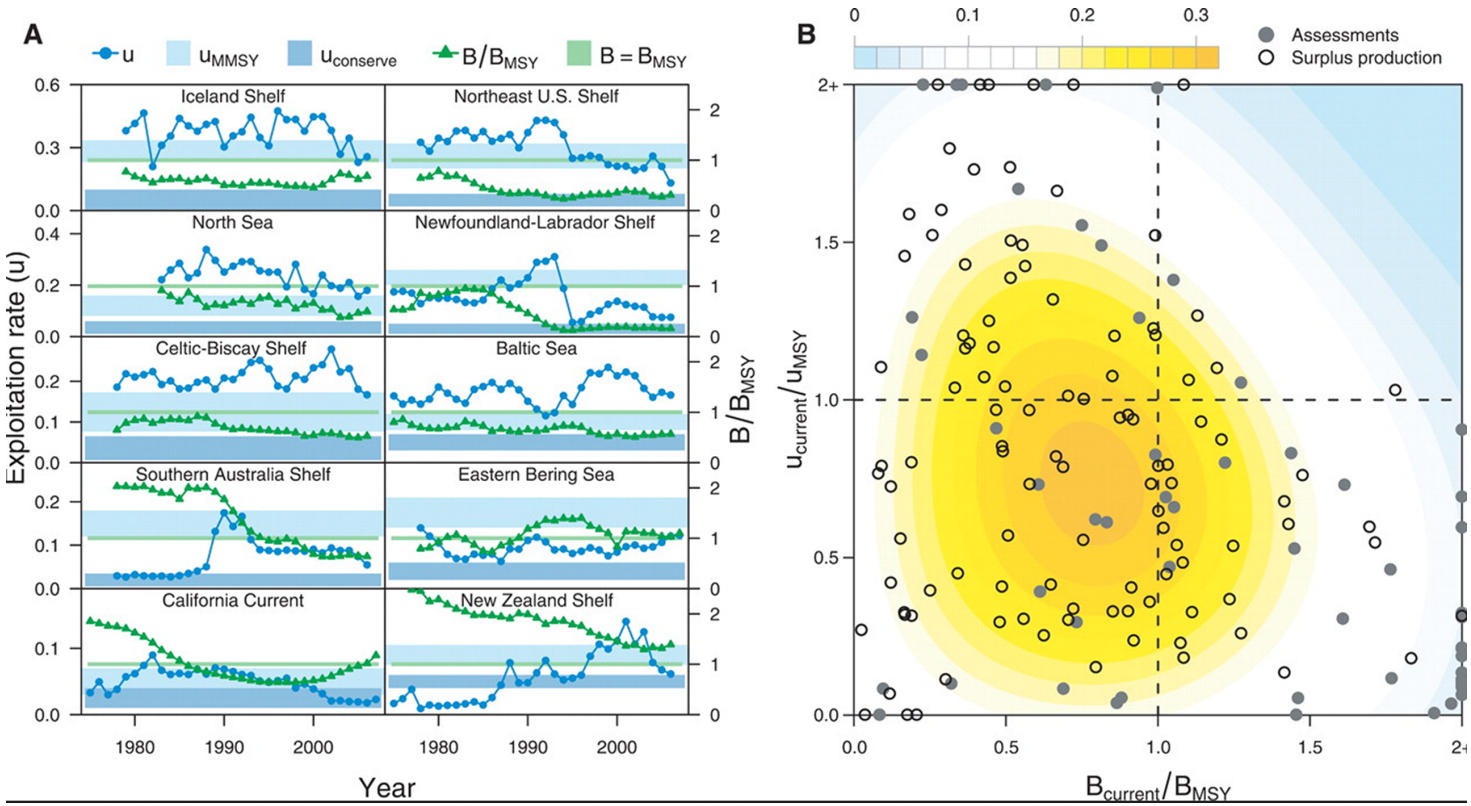


Complex example

```
> mat <- matrix(c( 1, 1, 3, 4, 5,
+                  1, 1, 3, 6, 7,
+                  2, 2, 2, 2, 7),
+                nrow=3, ncol=5, byrow=T)
>
> layout(mat=mat, widths = c(1,3,5,1,3),
+        heights = c(2,2,1) )
>
> layout.show(n=7)
```


1
3
4
5
6
7
2

Example: Worm et al. (2009) *Science*



Example: Worm et al. (2009) *Science*

11	
1	6
2	7
3	8
4	9
5	10

13
12

Example: Worm et al. (2009) *Science*

```
>    mat <- matrix(c(11,  11,   0,  13,
+                     1,   6,   0,  12,
+                     2,   7,   0,  12,
+                     3,   8,   0,  12,
+                     4,   9,   0,  12,
+                     5,  10,   0,  12),
+                   nrow=6, ncol=4, byrow = TRUE)
>
>    layout(mat=mat, widths = c(2,2,1,4),
+           heights = c(1,2,2,2,2,2) )
>
>    layout.show(13)
```

Lab exercise

Use `layout()` to create a structure for 4 plots that has 2 columns and 3 rows.
The first column should be twice as wide as the second column.
The first plot should appear in the entire first column, with the remaining plots filling the 2nd column, top to bottom.
Verify that you achieved your objective with a call to `layout.show()` detailing the location of the four plots.

Plotting 3D data (`image()`)

We often want to display 3D or spatial data in a plot. *images*

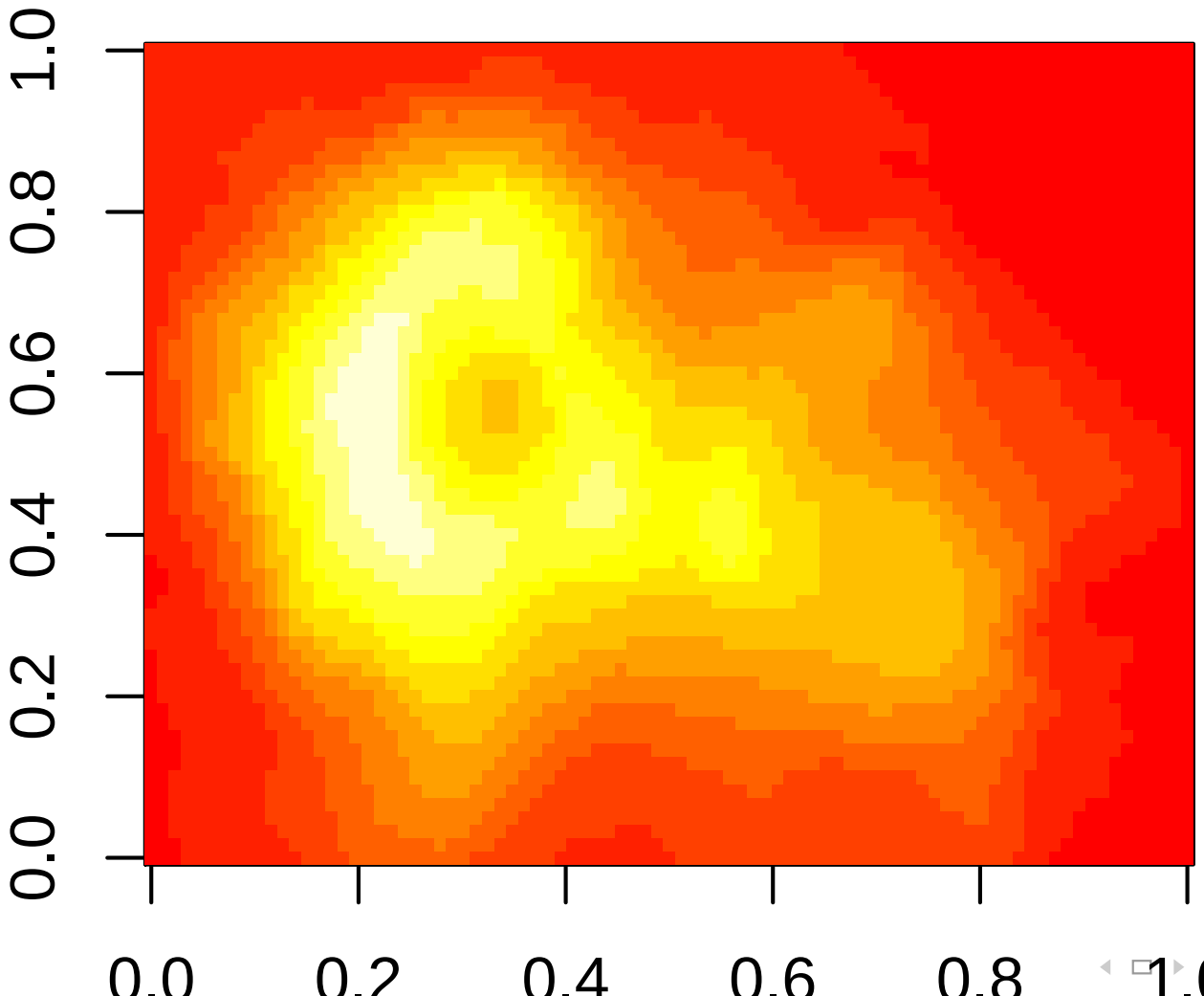
```
>   image(x, y, # locations of grid lines,
+               # in order & ascending
+         z,    # matrix containing values to be plotted
+         col = heat.colors(12), #vector of colors
+         add = FALSE, # add to current plot?
+         useRaster, # use TRUE when have big matrix
+                    # and on regular grid.
+         ...)
```

Needing `x` and `y` to be in order and increasing is a common trip up for use of this and related functions.

image()

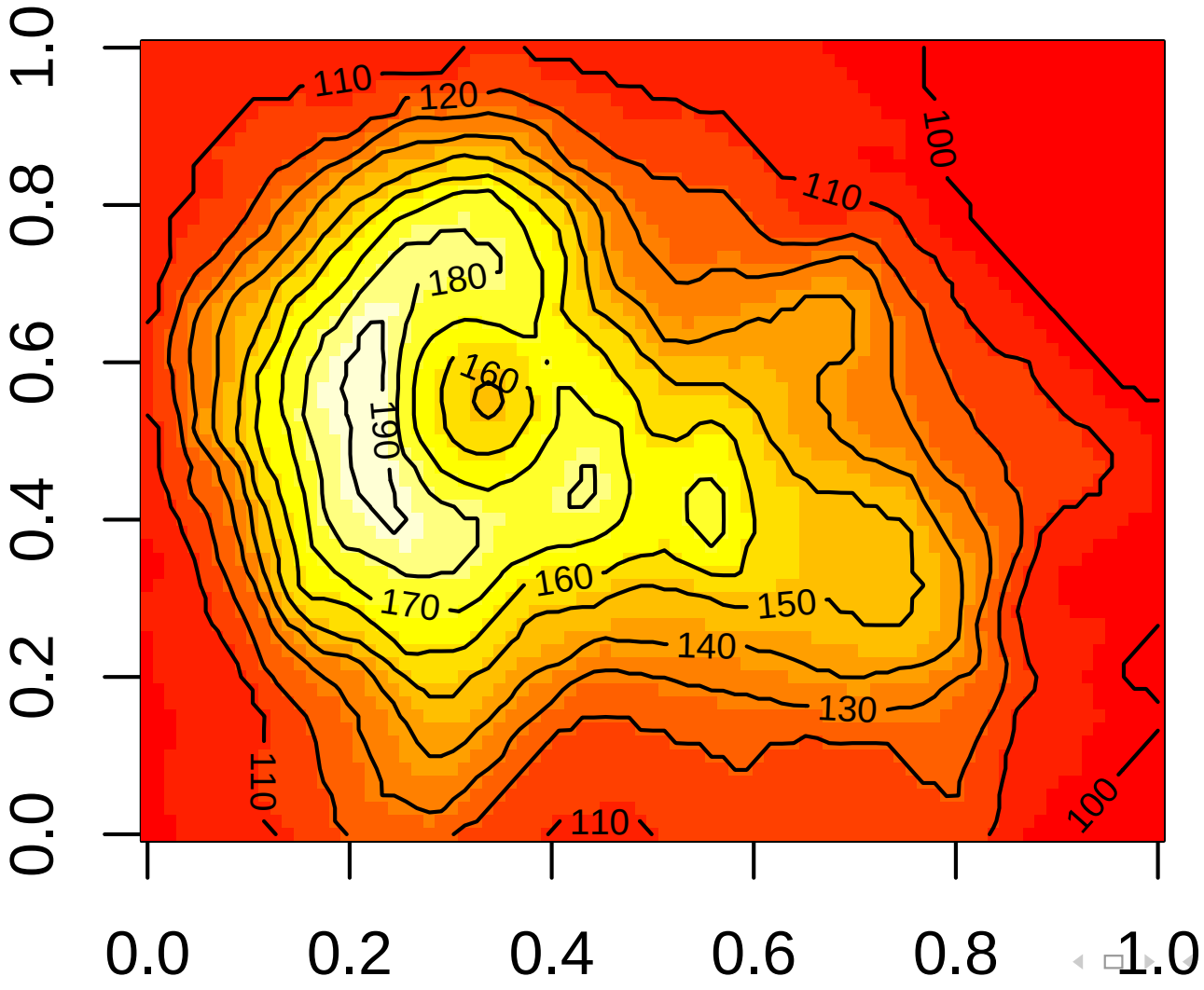
e.g. volcano dataset

```
>    image(volcano,useRaster=TRUE)
```



Adding contours with `contour()`

```
>    image(volcano, useRaster=TRUE)
>    contour(volcano, add=TRUE)
```



NetCDF files

Large data sets can have complex structures, and often are not amenable to text files.

"NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data."

Netcdf files are very common for atmospheric and oceanographic data.

As data are stored in arrays, these can be relatively easy to use once you 'crack' them open.
Packages in `R` for dealing with NetCDF
- `RNetCDF` ; the new standard
- `ncdf` ; older, not supported, but you will see much code (e.g. mine) that uses this.

NetCDF structure

NetCDF files contain sets of array objects (variables), each variable also has associated metadata.

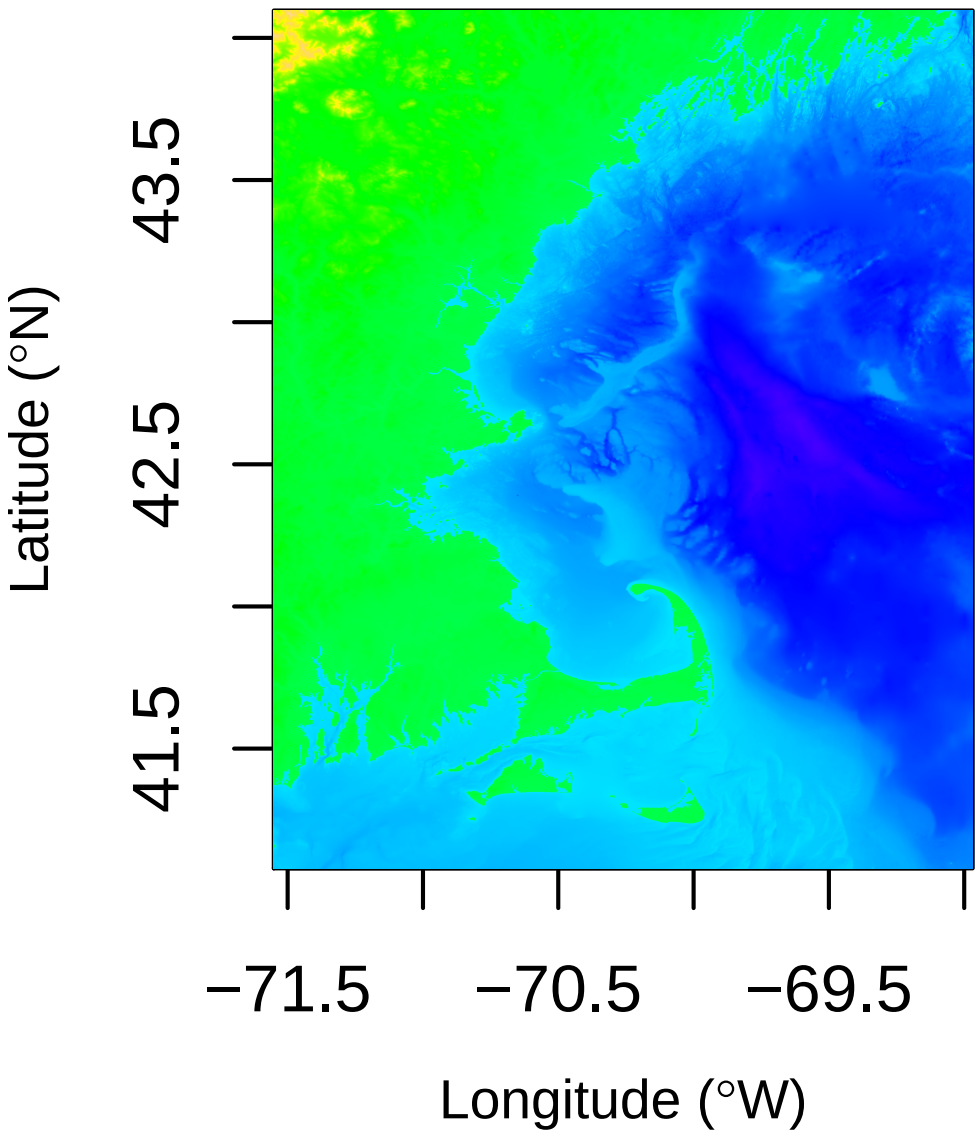
```
> library(RNetCDF)
> bath_file <- 'crm.nc'
> #open ncdf file
> my.nc <- open.nc(bath_file)
> #Use print.nc(my.nc) to print summary info
> #read netcdf dataset into a list object
> ncdat <- read.nc(my.nc)
> head(ncdat$lat)
[1] 41.07917 41.08000 41.08083 41.08167 41.08250 41.08333
> #alternatively, extract a single variable
> latvec <- var.get.nc(my.nc,'lat')
> head(latvec)
[1] 41.07917 41.08000 41.08083 41.08167 41.08250 41.08333
> #close the ncdf file
> close.nc(my.nc)
```

Example: Coastal Relief Model (Bathymetry)

3 second resolution data downloaded from NOAA via:
http://maps.ngdc.noaa.gov/viewers/wcs-client/

```
> bath_file <- 'crm.nc'
> #open ncdf file
> my.nc <- open.nc(bath_file)
> #read netcdf dataset into a list object
> ncdat <- read.nc(my.nc)
> #close the ncdf file
> close.nc(my.nc)
> lonvec <- ncdat$lon
> latvec <- ncdat$lat
> d2 <- ncdat$Band1
```

```
> #create a color scheme
> dmin=min(d2)
> dmax=max(d2)
> NwaterColors = round(abs(dmin))
> NlandColors = round(abs(dmax))
> #taking first 1/3 of topo.colors of length NwaterColors for water
> waterColorVec =
+   topo.colors(NwaterColors*3)[1:NwaterColors]
> #taking second 2/3 of topo.colors of length NlandColors for land
> landColorVec =
+   topo.colors(NlandColors*3/2)[round(NlandColors/
+               2+1):round(1.5*NlandColors)]
> #combine
> allColorVec = c(waterColorVec,landColorVec)
>
> #make an image plot
> image(lonvec,latvec,as.matrix(d2), col=allColorVec,
+       useRaster=TRUE)
> mtext(side=1,expression(paste("Longitude (",degree,
+                         "W)",sep="")),line=2.5,cex=0.8)
> mtext(side=2,expression(paste("Latitude (",degree,
+                         "N)",sep="")),line=2.5,cex=0.8)
```


Latitude (°N)
43.5
42.5
41.5
−71.5
−70.5
−69.5
Longitude (°W)

Maps in R

There are `n` (where `n` is large) ways to produce maps using `R`.

Google is your best friend.

Many of these are beyond this course and delve into the dark abstractedly projected rabbithole that is cartography.

However, we can produce reasonable looking maps quite easily. (because learning GIS can be hard)

Packages of interest:

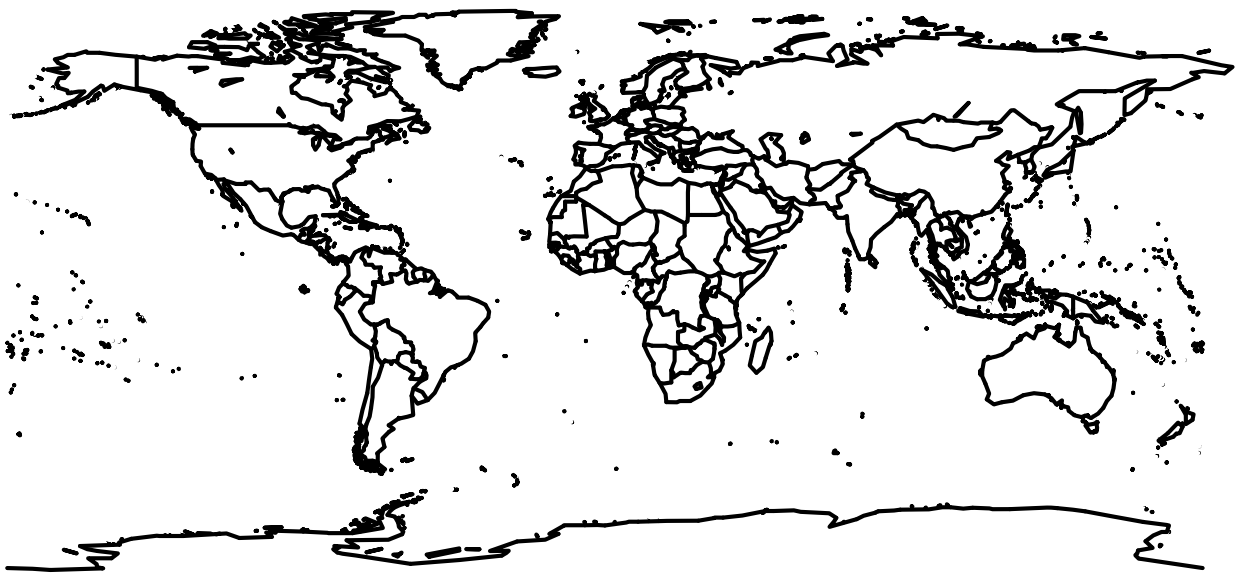
```
> library(maps)     #contains functions to make maps
> library(mapdata) # mapping data
> library(mapproj) # functions to deal with projections
> library(maptools) # more useful spatial functions
> library(scales) # for transparent colors
> library(RgoogleMaps) # not used today but very easy!
```

Making maps

Function `map()` is the workhorse.

e.g. map of the world

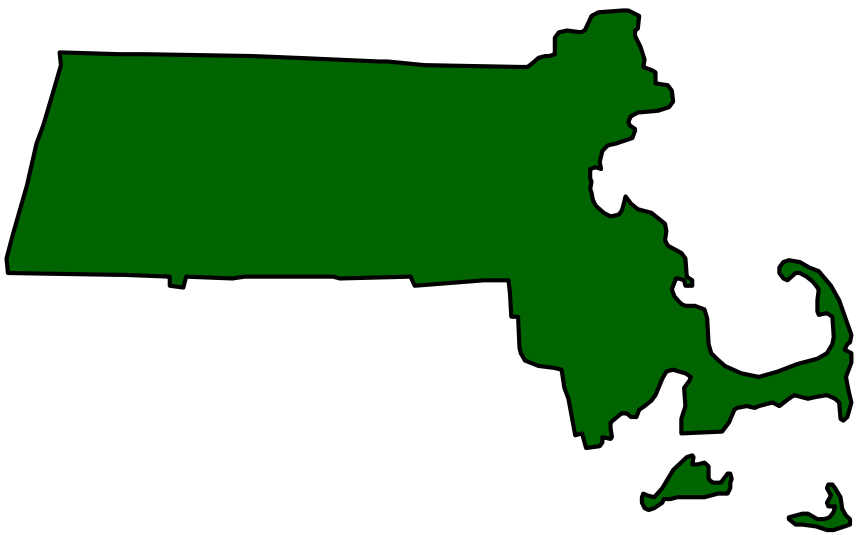
```
> map("worldHires")
```



More with maps

First argument is to a maps database, within which there are many objects.
3 main (US-centric) databases: "world", "state", "county"
`worldHires` better for detailed coastlines (but still low-res).
Coordinates are decimal degrees. Use `xlim` & `ylim` as for `plot()`.
Explore more at help(package='maps')

```
> map("state","Massachusetts",fill=TRUE,col="darkgreen")
```



Lab exercise

Use `map()` to create a map of part of the Gulf of Maine.

- The range for the Longitudes should be 71.5°W to 69°W.
- The range for the Latitudes should be 41.1°N to 44°N. (Note that longitudes in the maps database are centred on the Prime Meridian, and range from -180 to 180)
- Add axes, and color in the land.
- Add points indicating the locations of New Bedford, Boston, and Portland (Lat/Lons for US cities can be found in data frame `us.cities`. Use `points()` to add as with `plot()`).
- Experiment with the map projection.
- *BONUS* Re-add the cities (you will have to account for the new projection).

Additional mapping

As with `plot()`, additional layers can be added to maps using `add=TRUE`.
These overlay what has come before, so instructions should be sequential.

You can use `alpha()` to easily make colors transparent.

Far too many options to even crack the surface.

Other packages to check out include:

- `sp`
- `PBSmapping`
- `ggmap`

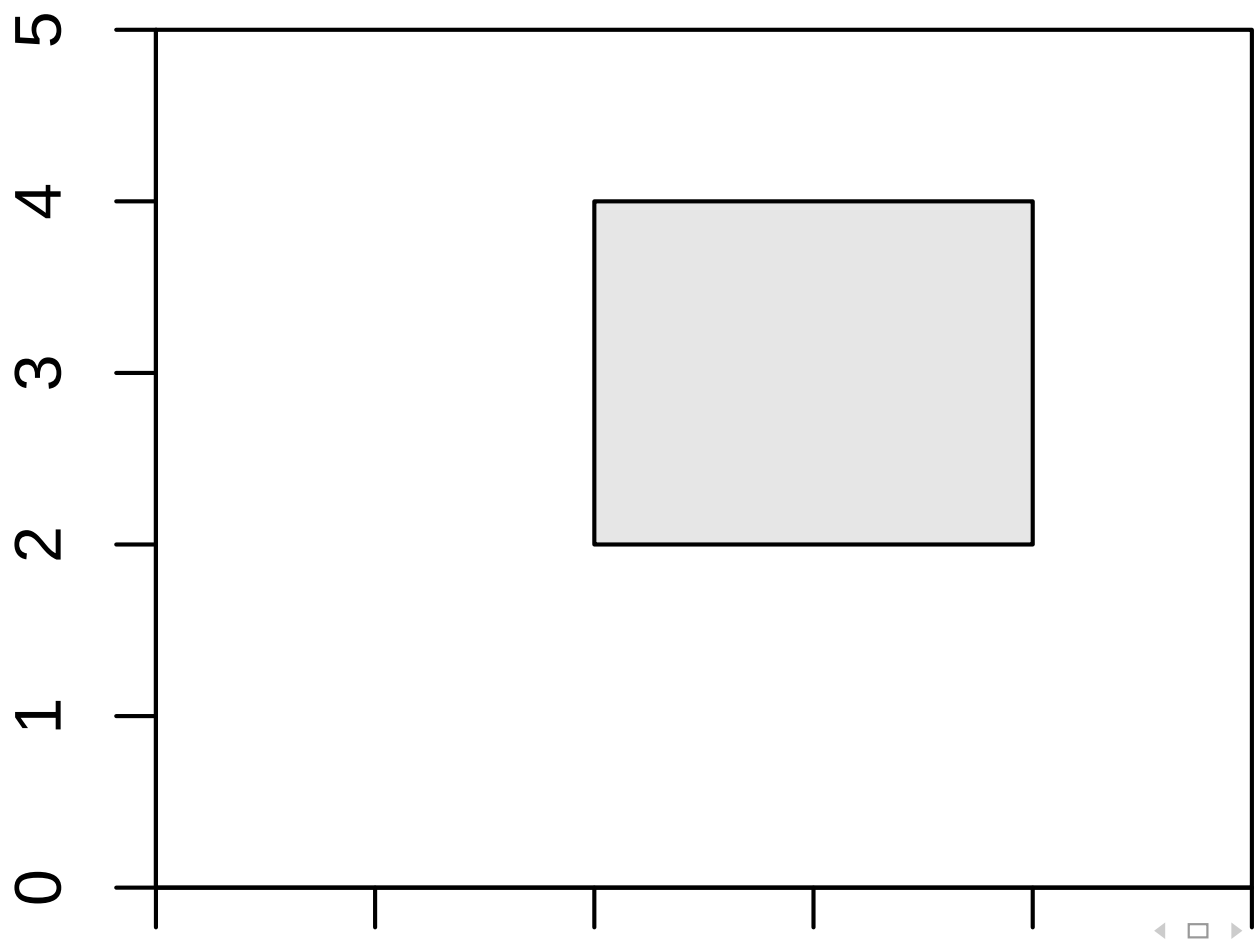
Adding polygons to plots

A lot of spatial data are defined as polygons (e.g. shapefiles).

We can also add polygons to regular plots.
Can be useful for plotting confidence intervals/regions, etc.

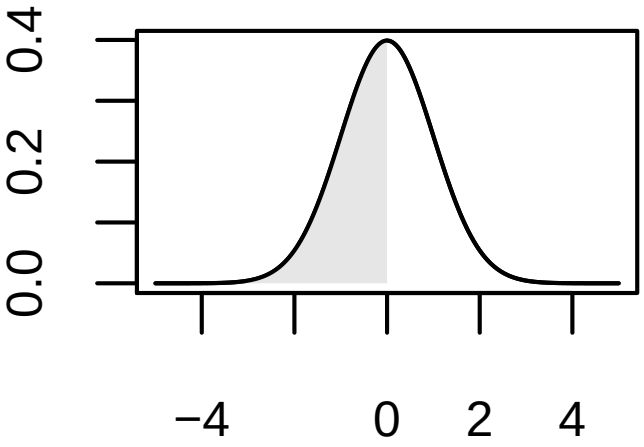
```
> polygon(x, y = NULL, #vectors of vertex coordinates
+         density = NULL, angle = 45,
+         border = NULL, col = NA, lty = par("lty"),
+         ..., fillOddEven = FALSE)
```

```
> plot(0,type='n',xlim=c(0,5),ylim=c(0,5),
+      xaxs="i",yaxs="i",ann=FALSE)
> #draw a square
> xvec <- c(2,4,4,2)
> yvec <- c(2,2,4,4)
> polygon(xvec,yvec,col=gray(0.9))
```



Lab exercise

Use a polygon to shade the lower 50% probability density for a standard normal distribution. e.g.



`gomLLhigh.Rdata` contains high-resolution coastline data for the area of the Gulf of Maine we plotted earlier.
(obtained from `http://www.soest.hawaii.edu/pwessel/gshhg/` and extracted using `importGSHHS()` from `PBSmapping`)
Coordinates for polygons making up the coast are found in vectors `X` and `Y`, with polygons identified in vector `PID`.
Plot polygon 2. Color and label it appropriately.

Lab exercise - pulling it all together

Plot a map of the spring 2015 NMFS bottom trawl survey catch rates for silver hake in the Gulf of Maine. Add histograms of the catch rates and depths in tows that caught silver hake, and the total length frequency (weighted by tow).

Steps:
1. Read in the survey data. (in `neus_data.csv`) 2. Identify the data for silver hake (use `neus_svspp.csv`), in the spring, in 2015.
3. Set up the plot layout.
4. For the map, use `map()` to set up the plot area to ensure a reasonable projection. Map the area.
5. Add the bathymetry layer to the plot.
6. Add points corresponding to the locations of positive tows. Make the size of the plotting character proportional to the log(Biomass).
7. Add axis labels, title, etc.
8. Add a legend.
9. Re-plot the coastline to make things clear. Use polygons and the high resolution data from `gomLLhigh.Rdata` to ensure Newport OR is not left underwater.
10. Use experience from previous labs to plot the histograms and the silver hake length frequency.

Hints
1. We have already created most of the (plotting) code to do this.
2. The coastal relief layer takes time to render. Once you know this is working, comment out this code while you work on the rest of the plot.