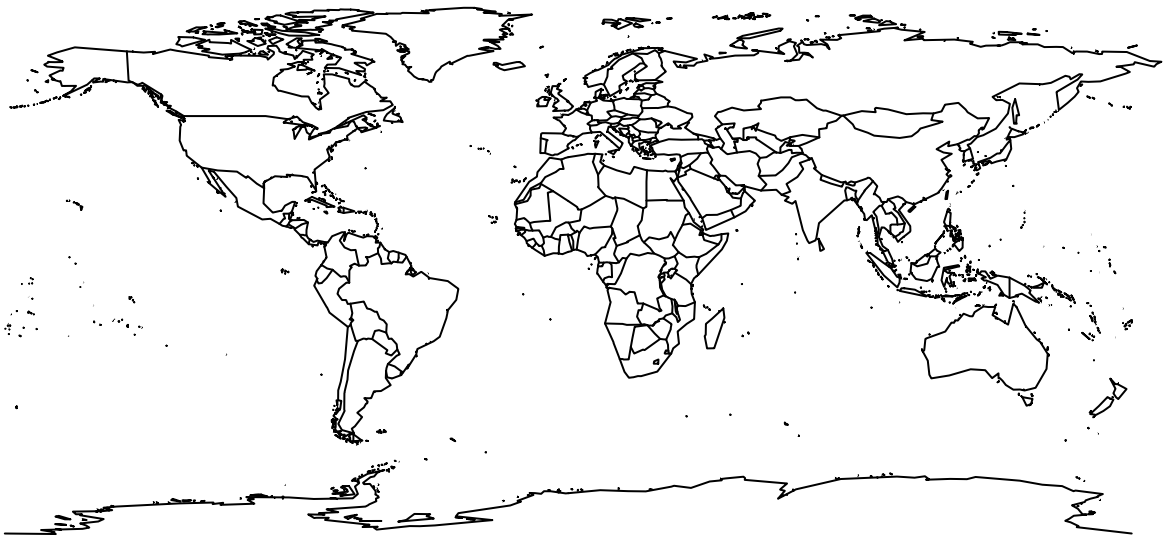
Geospatial Data and Maps: points, lines, polygons, gridded, shapfiles.

Mengqian LU

Start with a simple map

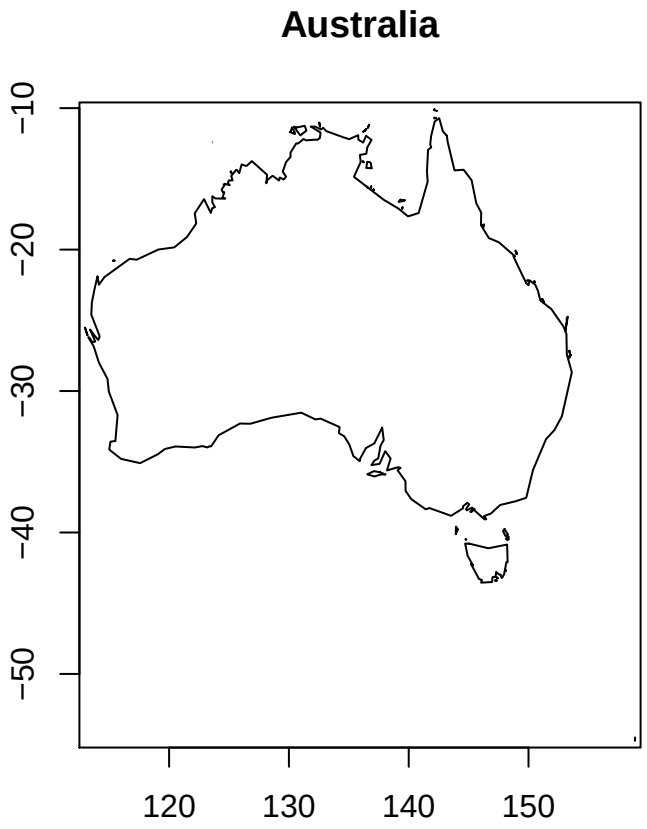
low resolution map of the world

```
library(maps)
library(mapdata)
map()
```



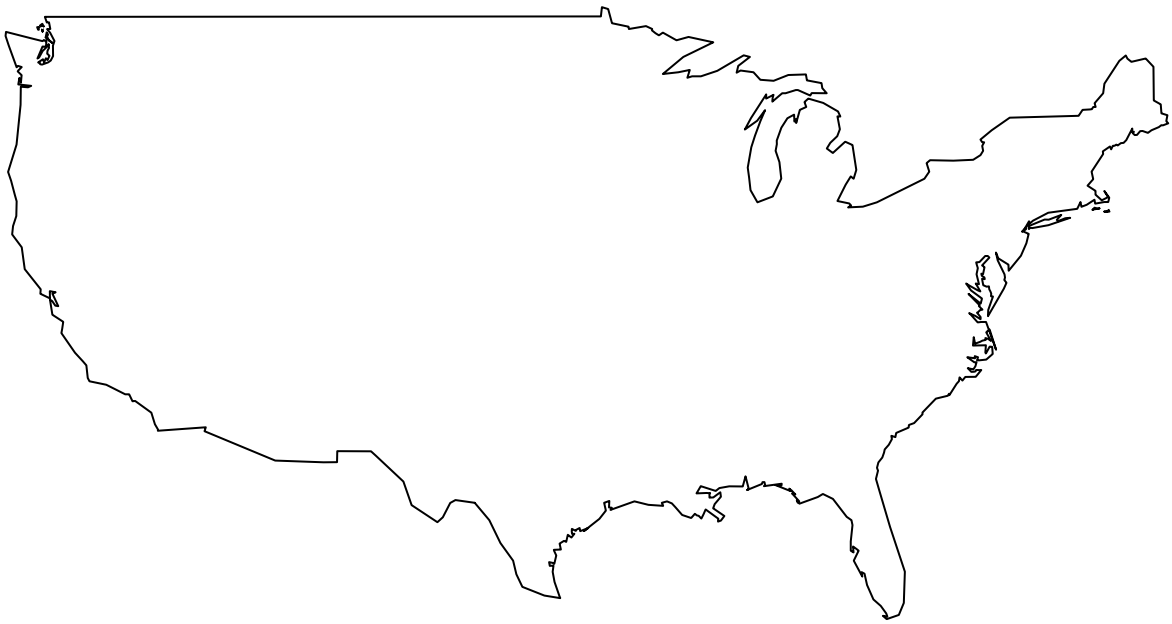
more

```
map('world',regions='australia')
title('Australia')
map.axes()
```



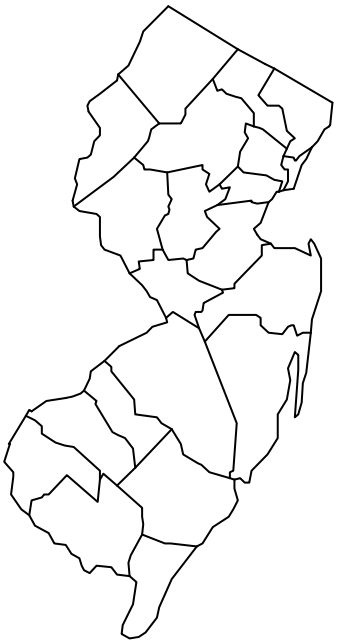
more

```
map('usa')  # national boundaries
```



more

```
map('county', 'new jersey') # county map of New Jersey
```

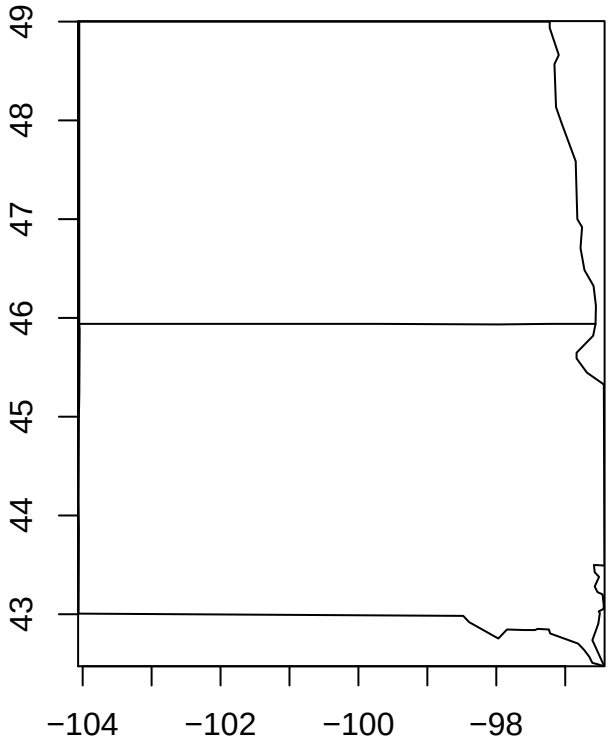


more

```
map('state', region = c('new york', 'new jersey', 'penn'))  # map of three states
```

more

```
map("state", ".*dakota", myborder = 0)  # map of the dakotas
map.axes()                # show the effect of myborder = 0
```



more

```
map('state', proj = 'bonne', param = 45)     # Bonne equal-area projection of states
```

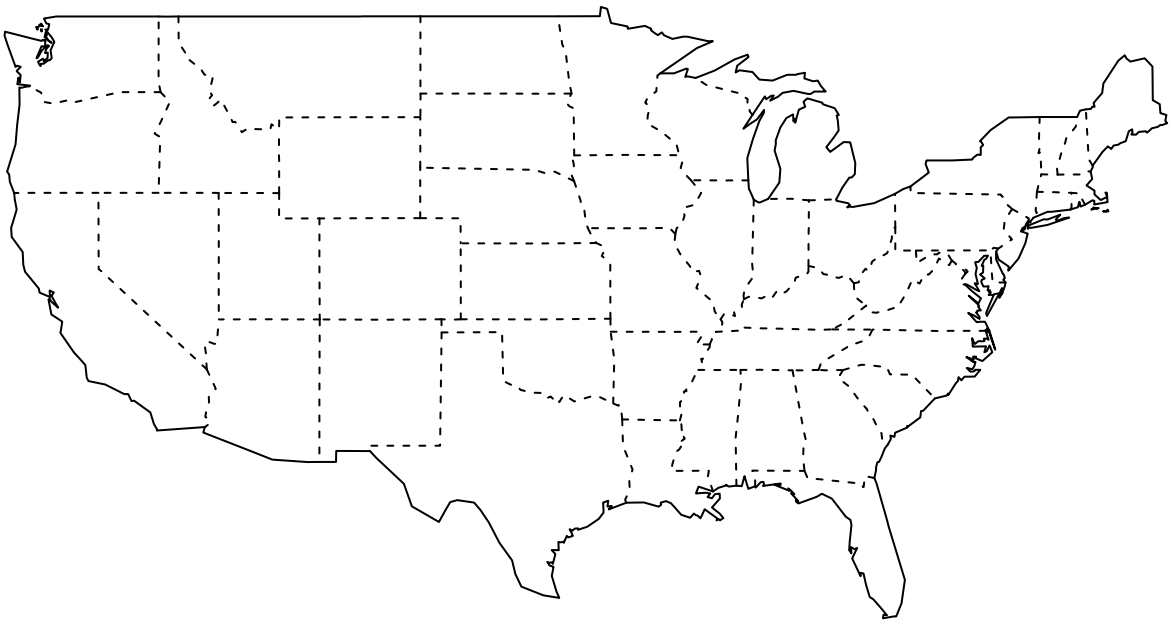
more

```
# names of the San Juan islands in Washington state
map('county', 'washington,san', names = TRUE, plot = FALSE)
```

```
## [1] "washington,san juan:lopez island"
## [2] "washington,san juan:orcas island"
## [3] "washington,san juan:san juan island"
```

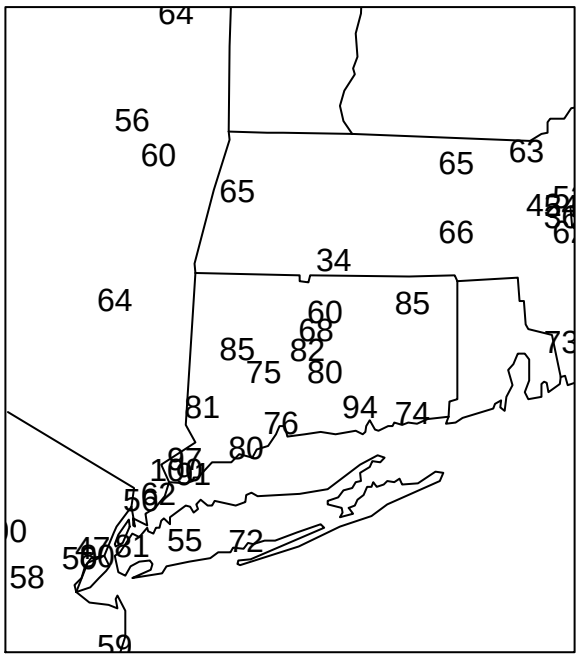
more

```
# national boundaries in one linetype, states in another
map("state", interior = FALSE)
map("state", boundary = FALSE, lty = 2, add = TRUE)
```



more

```
# plot the ozone data on a base map
data(ozone)
map("state", xlim = range(ozone$x), ylim = range(ozone$y))
text(ozone$x, ozone$y, ozone$median)
box()
```



2009 unemployment rate

```
# mapproj is used for  projection="polyconic"
# color US county map by 2009 unemployment rate
# match counties to map using FIPS county codes
# Based on J's solution to the "Choropleth Challenge"
# http://blog.revolutionanalytics.com/2009/11/choropleth-challenge-result.html

# load data
# unemp includes data for some counties not on the "lower 48 states" county
# map, such as those in Alaska, Hawaii, Puerto Rico, and some tiny Virginia
#  cities
data(unemp)
data(county.fips)
```

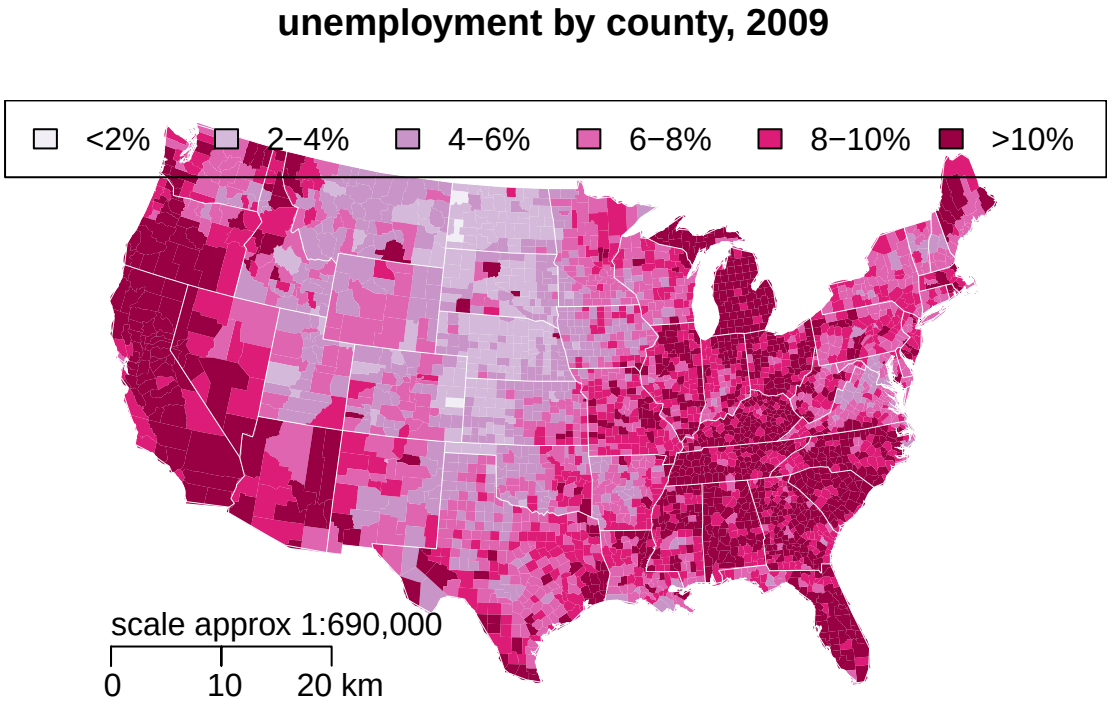
Some preparations

The FIPS county code is a five-digit Federal Information Processing Standard (FIPS) code (FIPS 6-4) which uniquely identifies counties and county equivalents in the United States, certain U.S. possessions, and certain freely associated states.

```
# define color buckets
colors = c("#F1EEF6", "#D4B9DA", "#C994C7", "#DF65B0", "#DD1C77", "#980043")
# use cut() to convert numeric to factor
unemp$colorBuckets = as.numeric(cut(unemp$unemp, c(0, 2, 4, 6, 8, 10, 100)))
leg.txt = c("<2%", "2-4%", "4-6%", "6-8%", "8-10%", ">10%")
# align data with map definitions by (partial) matching state,county
# names, which include multiple polygons for some counties
cnty.fips = county.fips$fips[match(map("county", plot=FALSE)$names,
                                   county.fips$polyname)]
colorsmatched = unemp$colorBuckets[match(cnty.fips, unemp$fips)]
```

draw map

```
library(fields)
map("county", col = colors[colorsmatched], fill = TRUE, resolution = 0,
    lty = 0, projection = "polyconic")
map("state", col = "white", fill = FALSE, add = TRUE, lty = 1, lwd = 0.2,
    projection="polyconic")
map.scale(relwidth = 0.15)
title("unemployment by county, 2009")
legend("topright", leg.txt, horiz = TRUE, fill = colors)
zr = rbind( c(1,6), c(1,6)) # separate ranges for columns of y.
colorbar.plot(x=1,y=5,strip=colorsmatched, adj.x=1, zrange= zr)
```




```
# image.plot(z=colorsmatched, breaks=c(0, 2, 4, 6, 8, 10, 100),col = colors,legend.only=TRUE)
```

Plot GPS Data and Shapefiles

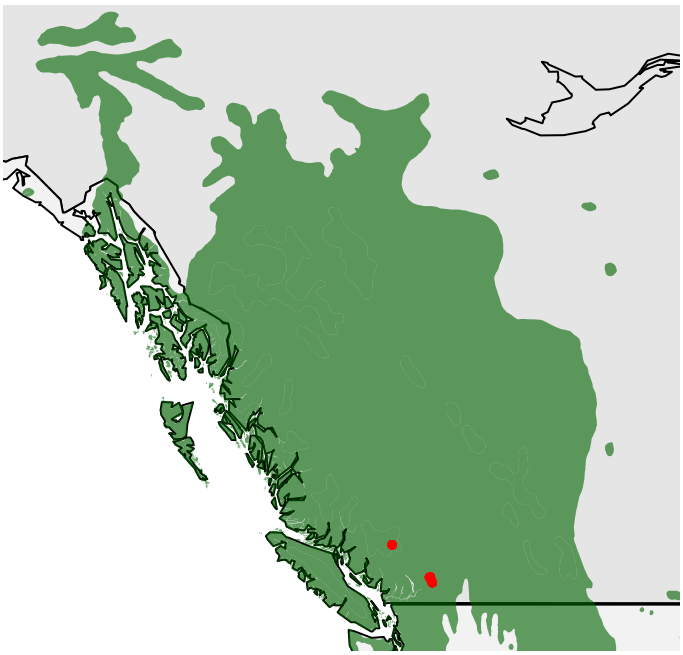
Shapefile is a popular geospatial vector data format for geographic information system (GIS) software, However GIS software is expensive and R now can do as well as GIS, if not better.

```
library(maptools)  #for shapefiles
library(scales)  #for transparency
# First read in shapefiles that contain layers of data for species range
setwd('/Users/MengqianLU/Dropbox/Spring 2016/Exploratory Data Analysis and Visualization/R Codes/Data')
# Pinus occidentalis, also called "West Indian pine"
pinuocci = readShapePoly("./pinuocci/pinuocci.shp")
# Pinus serotina, also called "pond pine"
pinusero = readShapePoly("./pinusero/pinusero.shp")
# Pinus contorta, also called "lodgepole pine"
pinucont= readShapePoly("./pinucont/pinucont.shp")
```

Add a ecologist's field samples in .csv file

```
setwd('/Users/MengqianLU/Dropbox/Spring 2016/Exploratory Data Analysis and Visualization/R Codes/Data')
hersamps = read.csv('herFieldSamples.csv')
# Her data for sampling sites, contains a column of "lat" and a column of "lon" with GPS points in deci
map("worldHires","Canada", xlim=c(-140,-110),ylim=c(48,64), col="gray90", fill=TRUE)  #plot the lodgepo
map("worldHires","usa", xlim=c(-140,-110),ylim=c(48,64), col="gray95", fill=TRUE, add=TRUE)  #add the a
```

```
#plot the lodgepole pine's range
plot(pinucont, add=TRUE, xlim=c(-140,-110),ylim=c(48,64), col=alpha("darkgreen", 0.6), border=FALSE)
# to preserve the coastlines and borders behind the species range -- using the function 'alpha("darkgre
points(hersamps$lon, hersamps$lat, pch=19, col="red", cex=0.5)  #plot the ecologist's sample sites
```



Distortion problem comes when you show a large area that has a wide north-south extent.

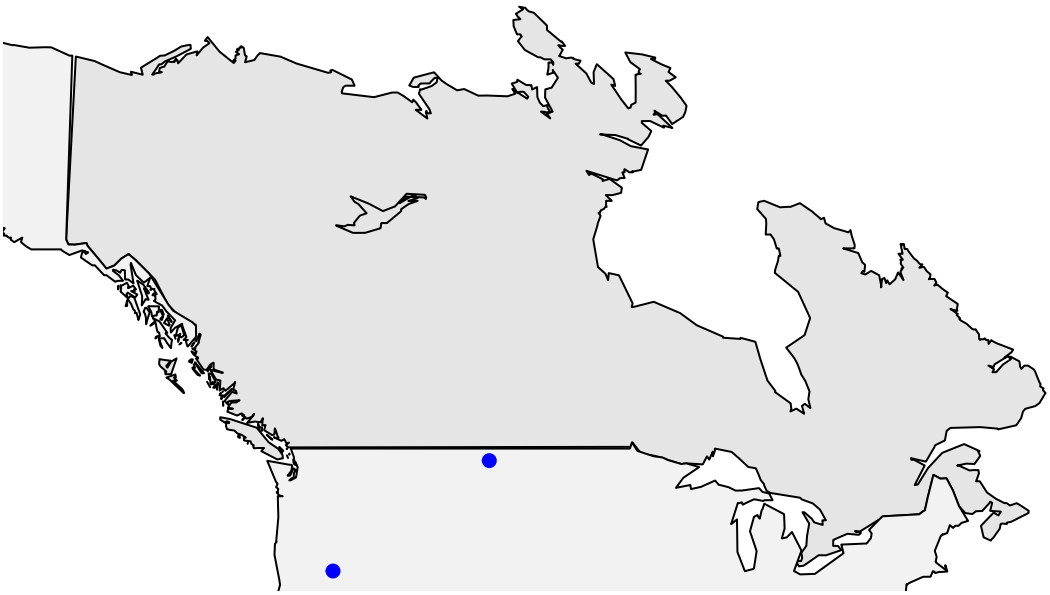
- ** Map Projections **

```
library(mapproj) # package to make projected maps
# with different projections
map("worldHires","Canada", xlim=c(-140,-110),ylim=c(48,64), col="gray90", fill=TRUE, projection="gilber
map("worldHires","usa", xlim=c(-140,-110),ylim=c(48,64), col="gray95", fill=TRUE, add=TRUE, projection=
```

With reference points

```
# now make some reference points of my interests
lon = c(-120, -86, -107, -100)  #longitude vector
lat = c(41.7, 34.6, 48.3, 40)  #latitude vector

coord = mapproject(lon, lat, proj="gilbert", orientation=c(90, 0, 225))  #convert points to projected l
map("worldHires","Canada", xlim=c(-140,-110),ylim=c(48,64), col="gray90", fill=TRUE, projection="gilber
map("worldHires","usa", xlim=c(-140,-110),ylim=c(48,64), col="gray95", fill=TRUE, add=TRUE, projection=
points(coord, pch=19, cex=0.9, col="blue")  #plot converted points
```



Recreate the John Snow plot Data

```
library(HistData)
data(Snow.deaths); data(Snow.pumps); data(Snow.streets); data(Snow.polygons)
```

draw a rough approximation to Snow's map and data

define some funtions to make the pieces re-usable

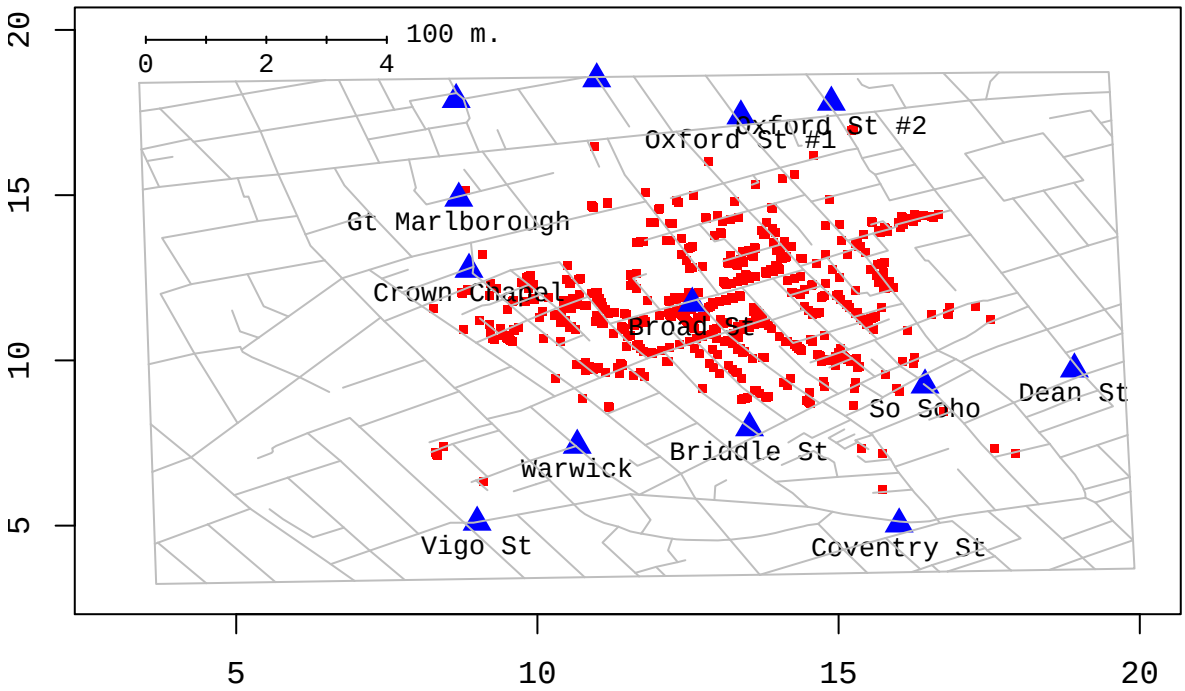
```
Sdeaths = function(col="red", pch=15, cex=0.6) {
        # make sure that the plot limits include all the other stuff
   plot(Snow.deaths[,c("x","y")], col=col, pch=pch, cex=cex,
    xlab="", ylab="", xlim=c(3,20), ylim=c(3,20),
    main="Snow's Cholera Map of London",family='mono')
    }
# function to plot and label the pump locations
Spumps = function(col="blue", pch=17, cex=1.5)  {
   points(Snow.pumps[,c("x","y")], col=col, pch=pch, cex=cex)
   text(Snow.pumps[,c("x","y")], labels=Snow.pumps$label, pos=1, cex=0.8,family='mono')
}

# function to draw the streets
Sstreets = function(col="gray") {
    slist = split(Snow.streets[,c("x","y")],as.factor(Snow.streets[,"street"]))
    invisible(lapply(slist, lines, col=col))
}
```

draw a scale showing distance in meters in upper left

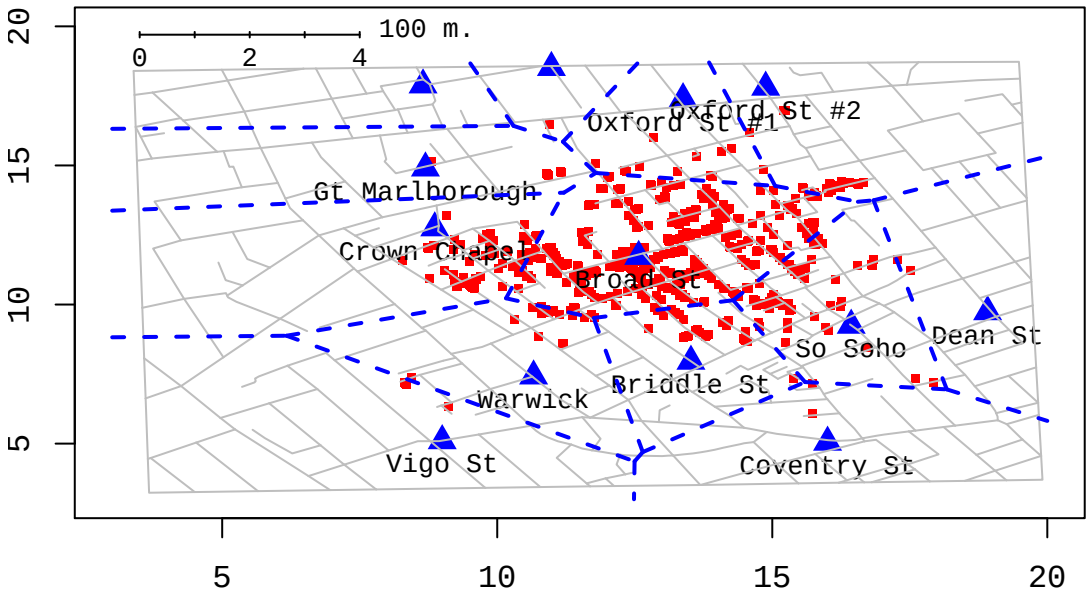
```
mapscale = function(xs=3.5, ys=19.7) {
   scale = matrix(c(0,0, 4,0, NA, NA), nrow=3, ncol=2, byrow=TRUE)
   colnames(scale)= c("x","y")
    # tick marks
   scale = rbind(scale, expand.grid(y=c(-.1, .1, NA), x=0:4)[,2:1])
   lines(xs+scale[,1], ys+scale[,2])
    # value and axis labels
   stext = matrix(c(0,0, 2,0, 4,0, 4, 0.1), nrow=4, ncol=2, byrow=TRUE)
   text(xs+stext[,1], ys+stext[,2], labels=c("0", "2", "4", "100 m."), pos=c(1,1,1,4), cex=0.8,family='
}

options(family='mono')
# draw the map with the pieces
Sdeaths()
Spumps()
Sstreets()
mapscale()
```



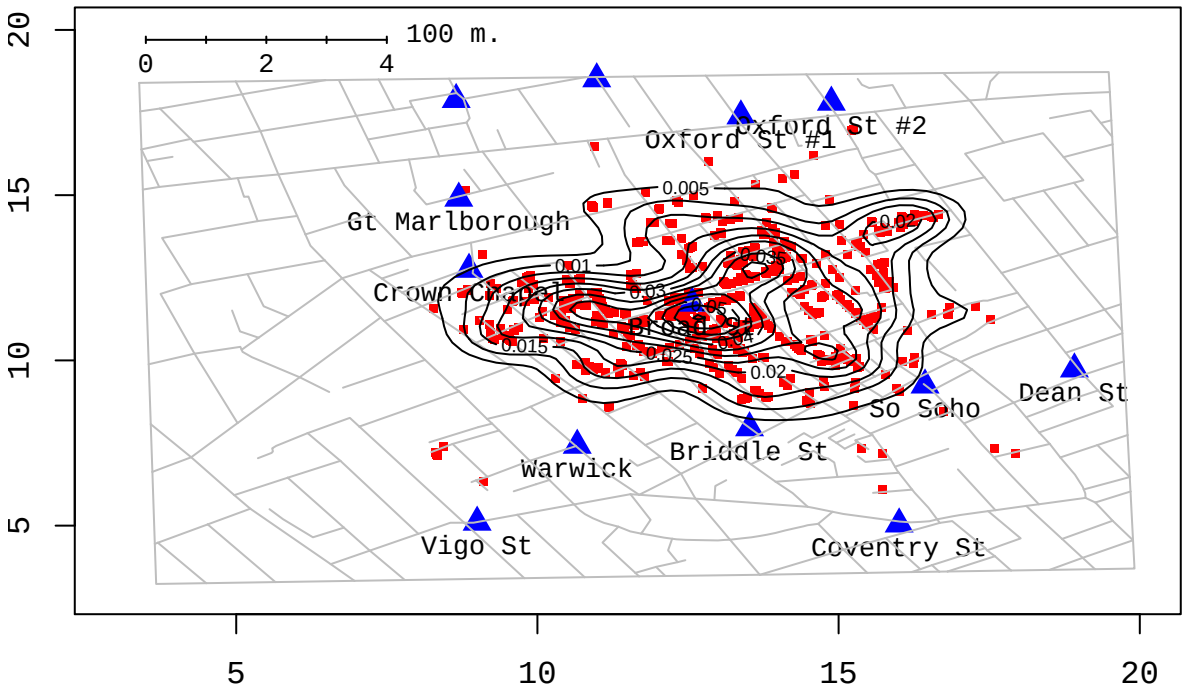
draw the Thiessen polygon boundaries

```
Sdeaths();Spumps();Sstreets();mapscale();
starts = which(Snow.polygons$start==0)
for(i in 1:length(starts)) {
    this = starts[i]:(starts[i]+1)
    lines(Snow.polygons[this,2:3], col="blue", lwd=2, lty=2)
    }
```



overlay bivariate kernel density contours of deaths

```
Sdeaths()
Spumps()
Sstreets()
mapscale()
require(KernSmooth)
kde2d = bkde2D(Snow.deaths[,2:3], bandwidth=c(0.5,0.5))
contour(x=kde2d$x1, y=kde2d$x2,z=kde2d$fhat, add=TRUE)
```



Using sp instead

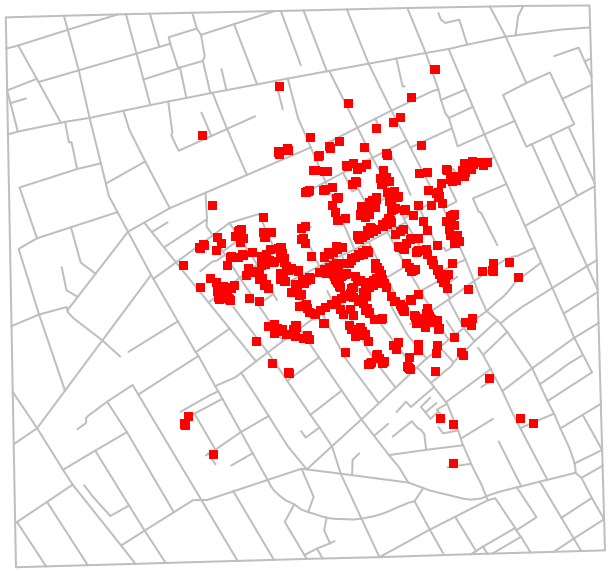
```
library(sp)
# streets
slist = split(Snow.streets[,c("x","y")],as.factor(Snow.streets[,"street"]))
Ll1 = lapply(slist,Line)
Lsl1 = Lines(Ll1,"Street")
Snow.streets.sp = SpatialLines(list(Lsl1))
plot(Snow.streets.sp, col="gray")
title(main="Snow's Cholera Map of London (sp)")
```

Snow's Cholera Map of London (sp)

deaths

```
Snow.deaths.sp = SpatialPoints(Snow.deaths[,c("x","y")])
plot(Snow.streets.sp, col="gray");title(main="Snow's Cholera Map of London (sp)");plot(Snow.deaths.sp, a
```

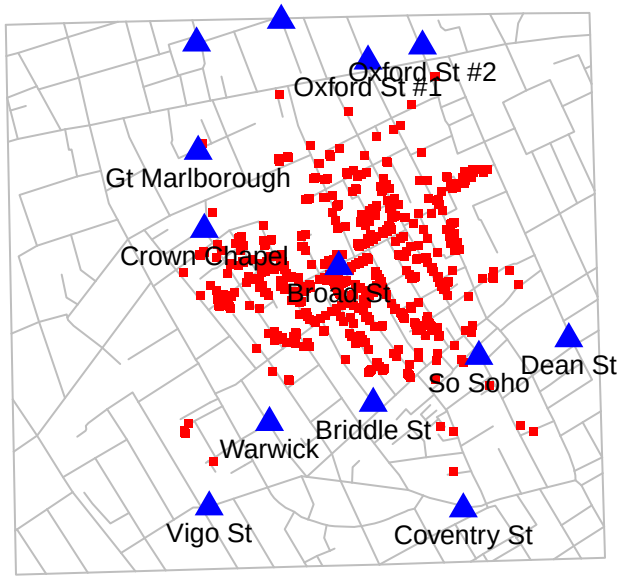
Snow's Cholera Map of London (sp)



pumps

```
spp = SpatialPoints(Snow.pumps[,c("x","y")])
Snow.pumps.sp = SpatialPointsDataFrame(spp,Snow.pumps[,c("x","y")])
plot(Snow.streets.sp, col="gray");title(main="Snow's Cholera Map of London (sp)");plot(Snow.deaths.sp, a
text(Snow.pumps[,c("x","y")], labels=Snow.pumps$label, pos=1, cex=0.8)
```

Snow's Cholera Map of London (sp)



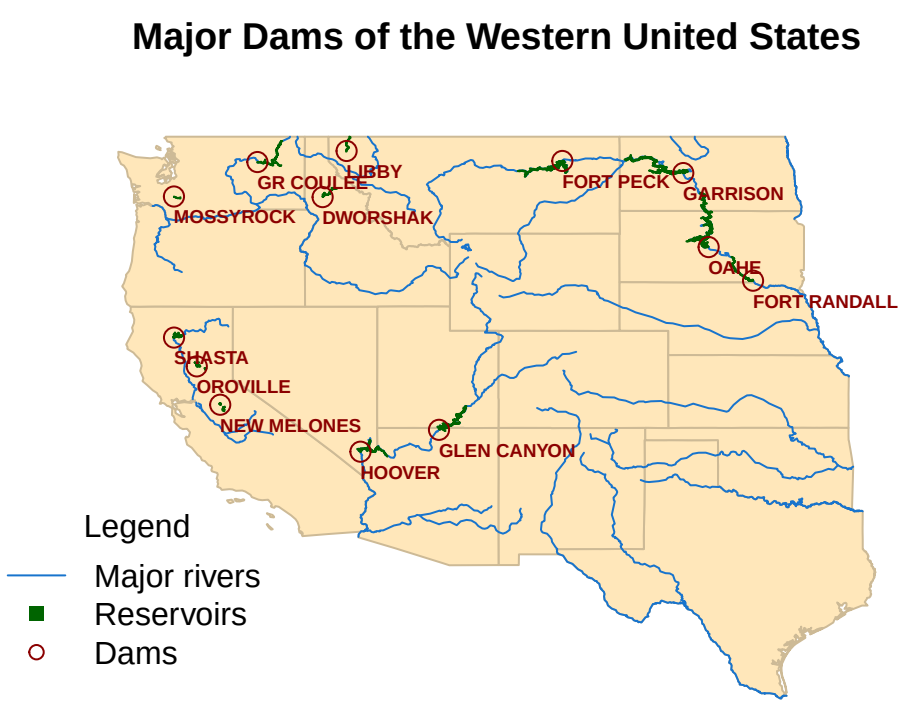
Example 1: Multi-layers – Thematic map with points, lines, and polygons A simple thematic map showing the location of major dams in the western United States. The data layers are available from the USGS as ESRI shapefiles, a commonly used file format for storing vector geospatial data. The map includes point, line, and polygon data layers along with dam labels, a legend, and optional coordinates along the axes.

```
library(sp)
library(maptools) # used here to read shapefiles
setwd('/Users/MengqianLU/Dropbox/Spring 2016/Exploratory Data Analysis and Visualization/R Codes/Data/Sp
# read in the spatial data
# ...western US state outlines
states = readShapePoly('western-states')
# ...major western US reservoirs
reservoirs = readShapePoly('western-reservoirs')
# ...major western US rivers
rivers = readShapeLines('western-rivers')
# ...locations of several western US dams
dams = readShapePoints('western-dams')
```

start by plotting the states

```
plot(states, border = "wheat3", col = "wheat1")
# add the river lines
lines(rivers, col = "dodgerblue3")
# add the reservoirs
plot(reservoirs, col = "darkgreen", border = "darkgreen", add = TRUE)
# add dams (circled)
points(dams, cex = 1.4, col = "darkred")
# add dam labels (using trial and error for placement)
text(dams, labels = as.character(dams$DAM_NAME), col = "darkred", cex = 0.6,
    font = 2, offset = 0.5, adj = c(0, 2))
```

```
# add a plot title and legend
title("Major Dams of the Western United States")
legend("bottomleft", legend = c("Major rivers", "Reservoirs", "Dams"), title = "Legend",
    bty = "n", inset = 0.05, lty = c(1, -1, -1), pch = c(-1, 15, 1), col = c("dodgerblue3",
        "darkgreen", "darkred"))
```



USGS http://viewer.nationalmap.gov/launch/

```
library(rgdal)
library(maptools)
setwd('/Users/MengqianLU/Dropbox/Spring 2016/Exploratory Data Analysis and Visualization/R Codes/Data/Sp
# read in the counties and their centroids
centroids = readShapePoints('op-county-centroids')
counties = readShapePoly('op-counties')

# read raster in as a SpatialGridDataFrame object
psImg = readGDAL('op-landsat.img')
```

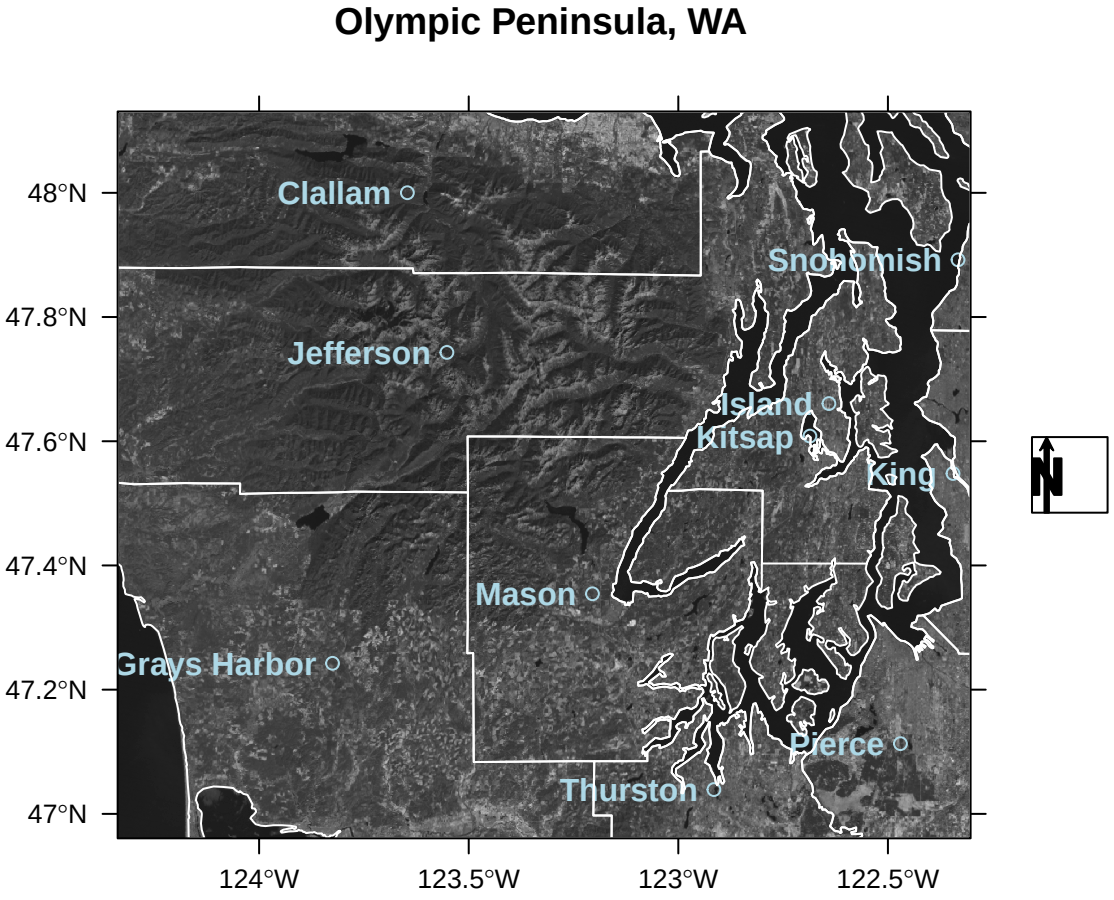
```
## op-landsat.img has GDAL driver HFA
## and has 868 rows and 1509 columns
```

specify overplot layers for use by spplot; in order for polygons to be plotted atop the raster, we need to convert them to SpatialLines

```
polys = list("sp.lines", as(counties, "SpatialLines"), col = "white")
points = list("sp.points", centroids, col = "lightblue", pch = 1)
labels = list("panel.text", coordinates(centroids)[, 1], coordinates(centroids)[,
    2], labels = sub(" County", "", centroids$COUNTY), col = "lightblue", font = 2,
    pos = 2)
```

plot the raster with polygons, points, and labels

```
# png('map-olympic-spplot.png', height=600, width=600)
spplot(psImg, "band1", col.regions = grey(0:256/256), sp.layout = list(points,
    labels, polys), cuts = 256, colorkey = FALSE, scales = list(draw = TRUE),
    main = "Olympic Peninsula, WA", legend = list(right = list(fun = mapLegendGrob(layout.north.arrow()
```




```
# dev.off()
```

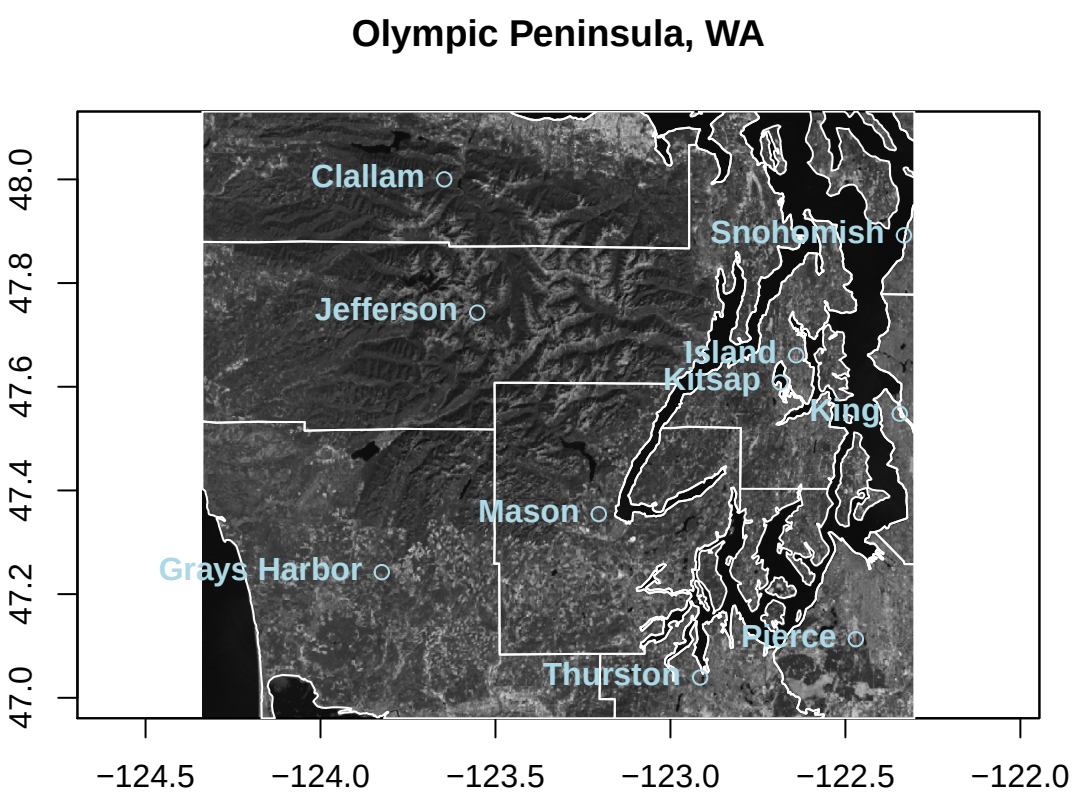
Faster codes

Raster package offers considerably more capabilities, including an extension of the base plot function for producing raster maps upon which other figure elements can be superimposed.

```
library(raster)
# read in the counties and their centroids
setwd('/Users/MengqianLU/Dropbox/Spring 2016/Exploratory Data Analysis and Visualization/R Codes/Data/Sp
centroids = readShapePoints("op-county-centroids")
counties = readShapePoly("op-counties")
 # read raster in as a Raster object
ps = raster("op-landsat.img")
```

plot the raster, then polygons, points, and labels

```
plot(ps, col = grey(0:256/256), legend = FALSE, main = "Olympic Peninsula, WA")
par(bg = "transparent")
plot(counties, border = "white", col = "transparent", add = TRUE)
points(centroids, col = "lightblue")
text(centroids, labels = sub(" County", "", centroids$COUNTY), col = "lightblue",
    font = 2, pos = 2)
```



Example of Australia's coastal climate information

```
library(latticeExtra)
data(EastAuClimate)

## Compare the climates of state capital cities
EastAuClimate[c("Hobart", "Melbourne", "Sydney", "Brisbane"), ]
```

```
##           SummerMaxTemp SummerMinTemp WinterMaxTemp WinterMinTemp
## Hobart             22.0          12.7          12.2           4.7
## Melbourne          26.5          15.8          13.9           6.8
## Sydney             26.7          19.3          17.0           7.4
## Brisbane           28.8          20.9          20.6           9.5
##           SummerRain WinterRain MeanAnnRain RainDays ClearDays CloudyDays
## Hobart          28.1       44.1       576.4     90.8      41.1      177.1
## Melbourne       32.3       46.8       654.4     99.2      48.9      178.0
## Sydney          80.6       54.5      1129.2     95.8     104.1      126.3
## Brisbane       107.5       40.0      1192.1     89.4     123.5      105.9
##              ID Latitude Longitude Elevation State
```

```
## Hobart    94029   -42.89    147.33        51   TAS
## Melbourne 86071   -37.81    144.97        31   VIC
## Sydney    66037   -33.94    151.17         6   NSW
## Brisbane  40223   -27.42    153.11         4   QLD
```

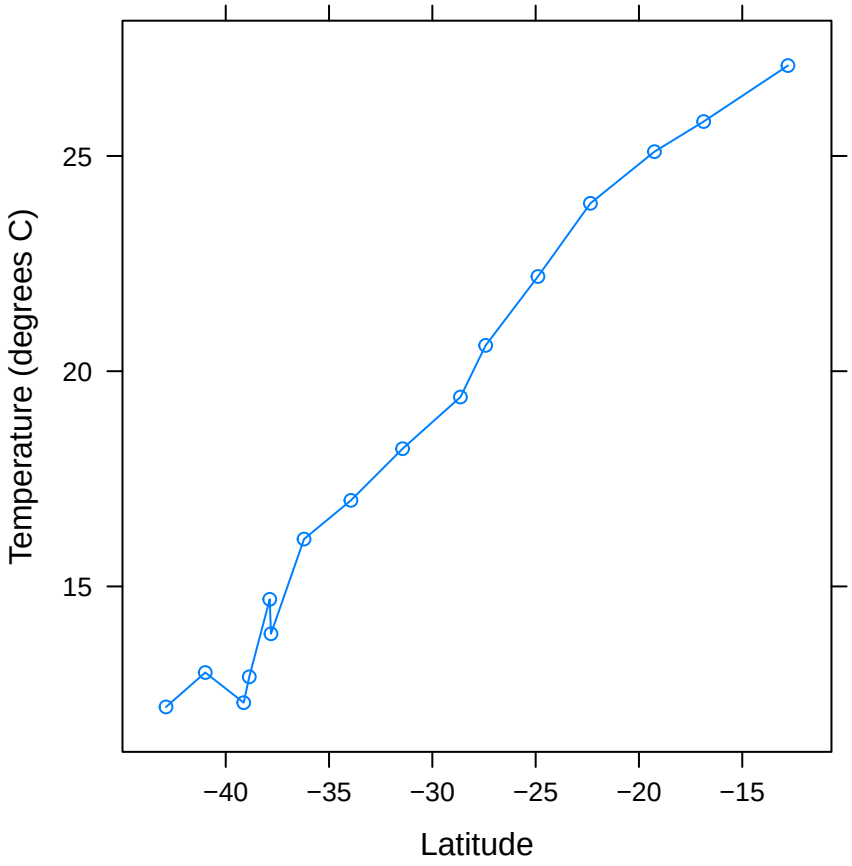
```
## A function to plot maps (a Lattice version of maps::map)
lmap <- function(database = "world", regions = ".", exact = FALSE, boundary = TRUE,
    interior = TRUE, projection = "", parameters = NULL, orientation = NULL,
    aspect = "iso", type = "l", par.settings = list(axis.line = list(col = "transparent")),
    xlab = NULL, ylab = NULL, ...) {
    theMap <- map(database, regions, exact = exact, boundary = boundary, interior = interior,
        projection = projection, parameters = parameters, orientation = orientation,
        plot = FALSE)
    xyplot(y ~ x, theMap, type = type, aspect = aspect, par.settings = par.settings,
        xlab = xlab, ylab = ylab, default.scales = list(draw = FALSE), ...)
}
```

Plot the sites on a map of Australia

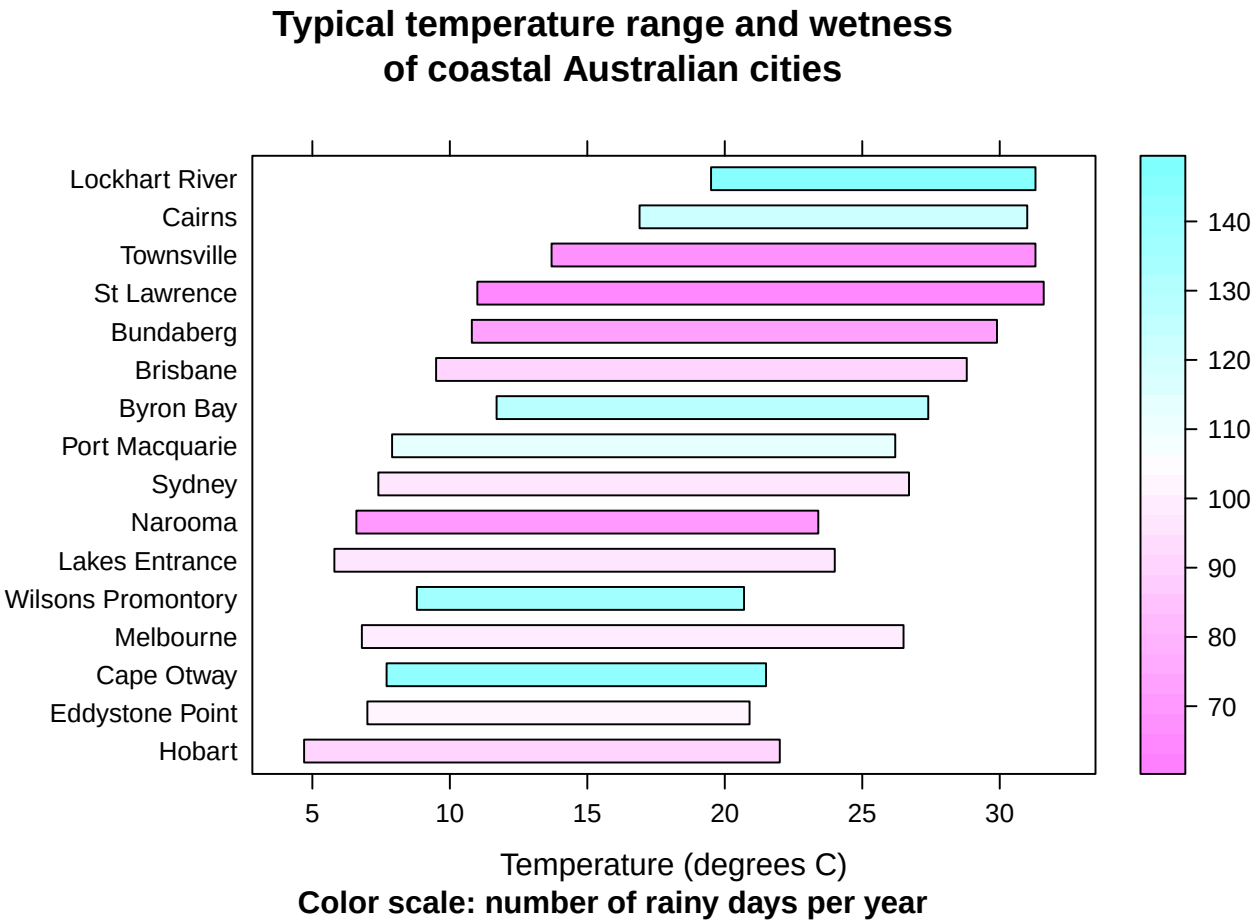
```
if (require("maps")) {
  lmap(regions = c("Australia", "Australia:Tasmania"),
       exact = TRUE, projection = "rectangular",
       parameters = 150, xlim = c(130, 170),
       panel = function(...) {
         panel.xyplot(...)
         with(EastAuClimate, {
           panel.points(Longitude, Latitude, pch = 16)
           txt <- row.names(EastAuClimate)
           i <- c(3, 4)
           panel.text(Longitude[ i], Latitude[ i], txt[ i], pos = 2)
           panel.text(Longitude[-i], Latitude[-i], txt[-i], pos = 4)
         })
       })
}
```

Average daily maximum temperature in July (Winter).

```
xyplot(WinterMaxTemp ~ Latitude, EastAuClimate, aspect = "xy", type = c("p",
    "a"), ylab = "Temperature (degrees C)")
```

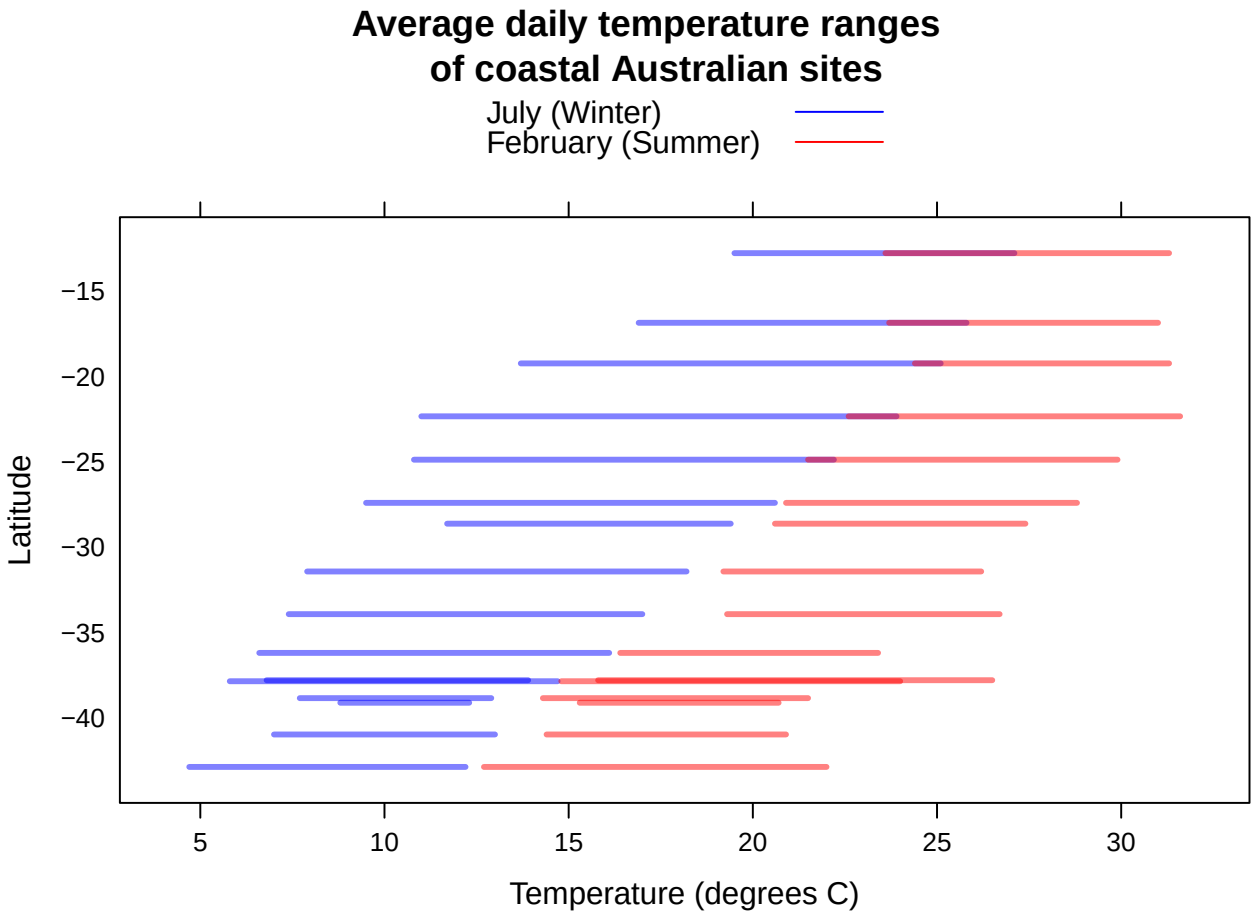



```
## (Make a factor with levels in order - by coastal location)
siteNames <- factor(row.names(EastAuClimate), levels = row.names(EastAuClimate))
## Plot temperature ranges (as bars), color-coded by RainDays
segplot(siteNames ~ WinterMinTemp + SummerMaxTemp, EastAuClimate, level = RainDays,
    sub = "Color scale: number of rainy days per year", xlab = "Temperature (degrees C)",
    main = paste("Typical temperature range and wetness", "of coastal Australian cities",
        sep = "\n"))
```



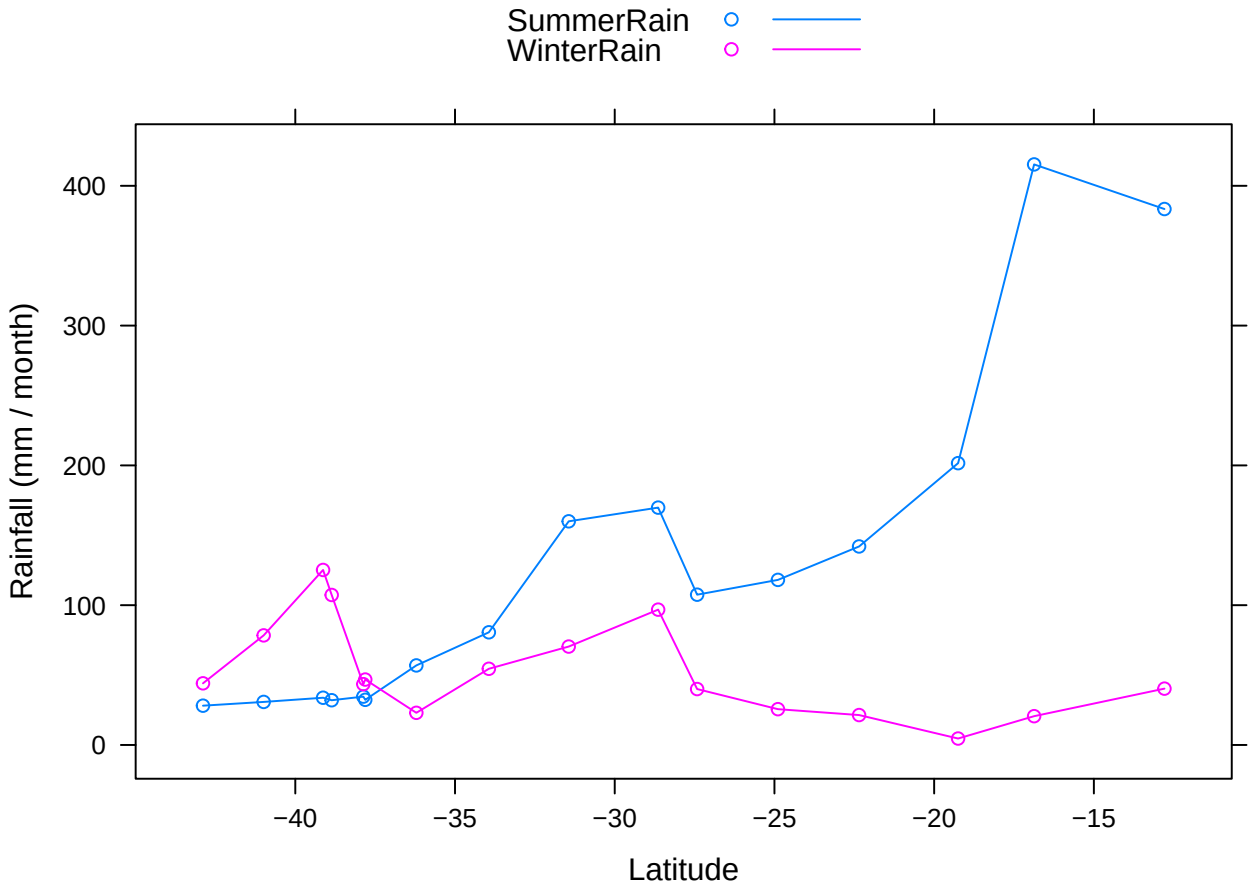
Show Winter and Summer temperature ranges separately

```
segplot(Latitude ~ WinterMinTemp + SummerMaxTemp, EastAuClimate, main = "Average daily temperature rang
    ylab = "Latitude", xlab = "Temperature (degrees C)", par.settings = simpleTheme(lwd = 3,
        alpha = 0.5), key = list(text = list(c("July (Winter)", "February (Summer)")),
        lines = list(col = c("blue", "red"))), panel = function(x, y, z, ...,
        col) {
        with(EastAuClimate, {
            panel.segplot(WinterMinTemp, WinterMaxTemp, z, ..., col = "blue")
            panel.segplot(SummerMinTemp, SummerMaxTemp, z, ..., col = "red")
        })
    })
```



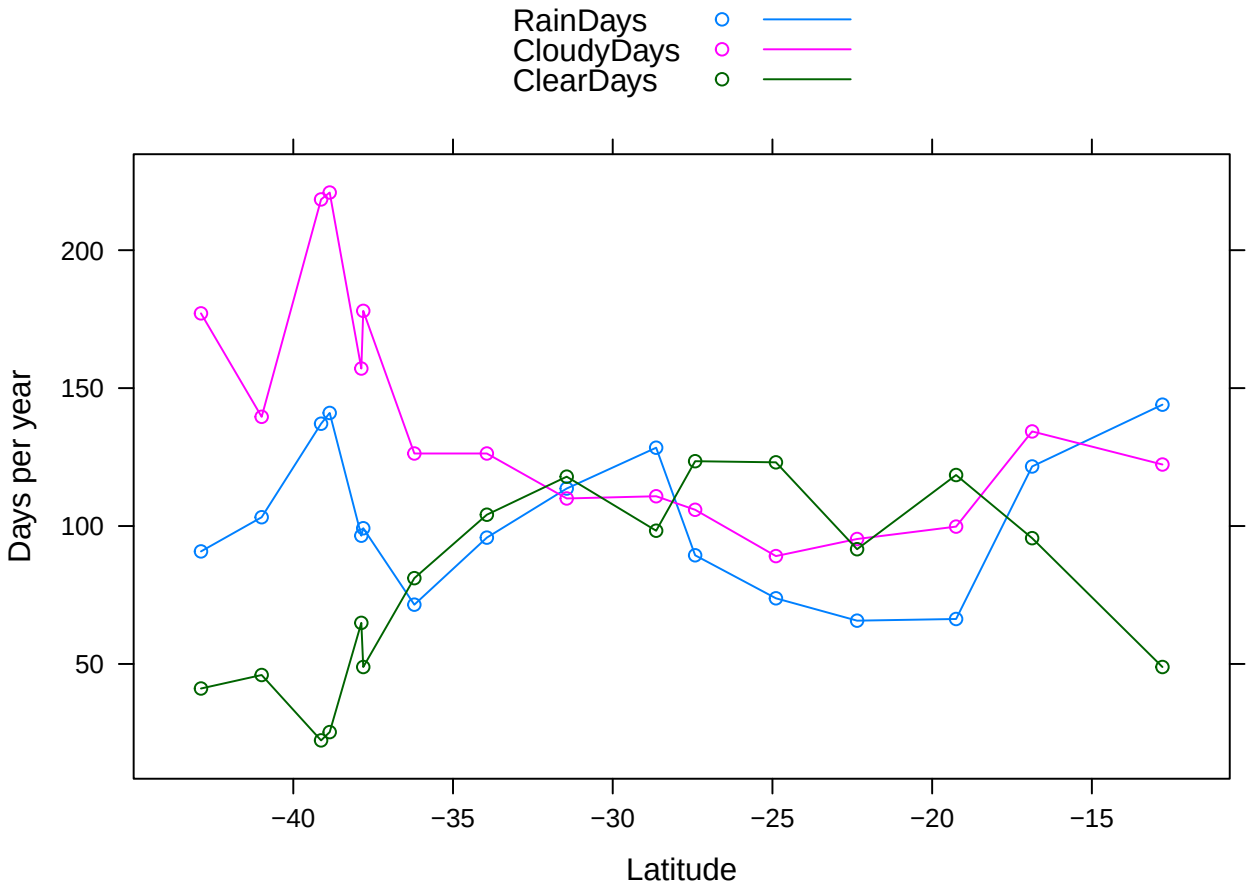
Northern sites have Summer-dominated rainfall; Southern sites have Winter-dominated rainfall.

```
xyplot(SummerRain + WinterRain ~ Latitude, EastAuClimate, type = c("p", "a"),
    auto.key = list(lines = TRUE), ylab = "Rainfall (mm / month)")
```



Clear days are most frequent in the mid latitudes.

```
xyplot(RainDays + CloudyDays + ClearDays ~ Latitude, EastAuClimate, type = c("p",
    "a"), auto.key = list(lines = TRUE), ylab = "Days per year")
```


RainDays
CloudyDays
ClearDays
Days per year
200
150
100
50
−40
−35
−30
−25
−20
−15
Latitude