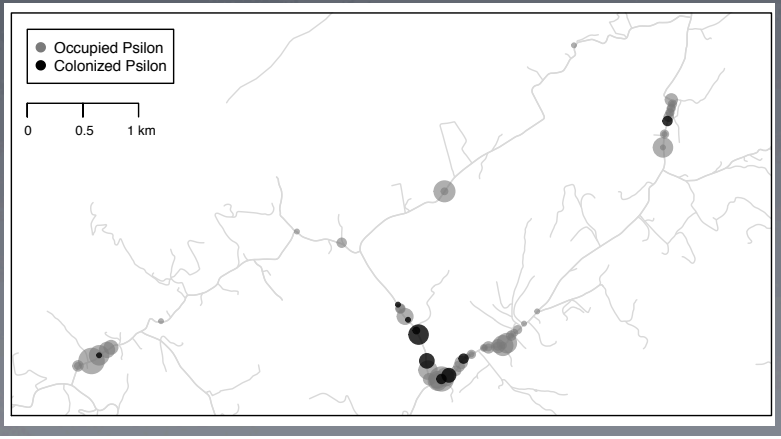
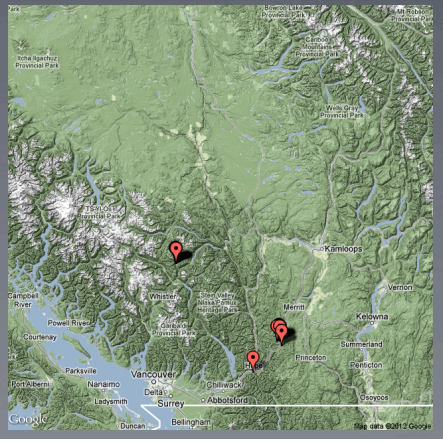
How to Make Maps in R

Kim Gilbert

27 August 2012

UBC





Why use R?

- Simplicity
  - ArcGIS is hard to learn
  - The simplest map requires no data files
  - More "control" over features
  - R help is amazing

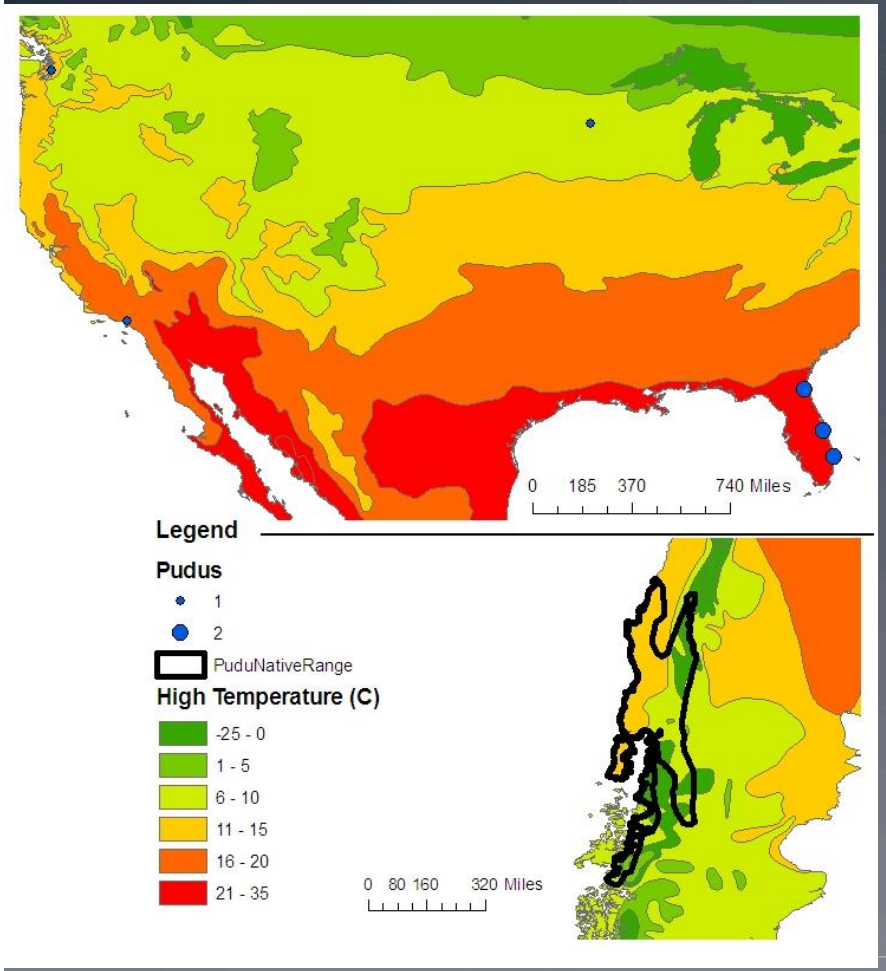
Why use R?

- Simplicity
  - ArcGIS is hard to learn
  - The simplest map requires no data files
  - More "control" over features
  - R help is amazing
- Repeatability
  - Run the same script and produce the same map
  - ArcGIS version incompatibilities

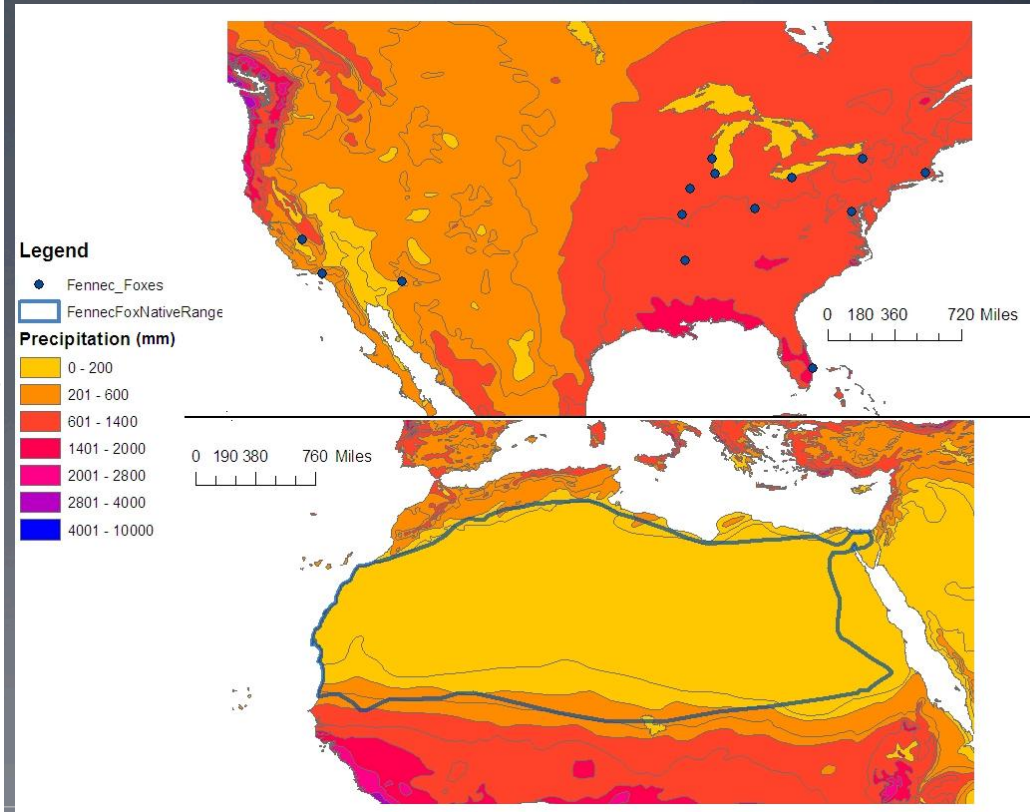
Why use R?

- Simplicity
  - ArcGIS is hard to learn
  - The simplest map requires no data files
  - More "control" over features
  - R help is amazing
- Repeatability
  - Run the same script and produce the same map
  - ArcGIS version incompatibilities
- FREE!
  - No licenses or complications, available wherever you are

Created with ArcMap

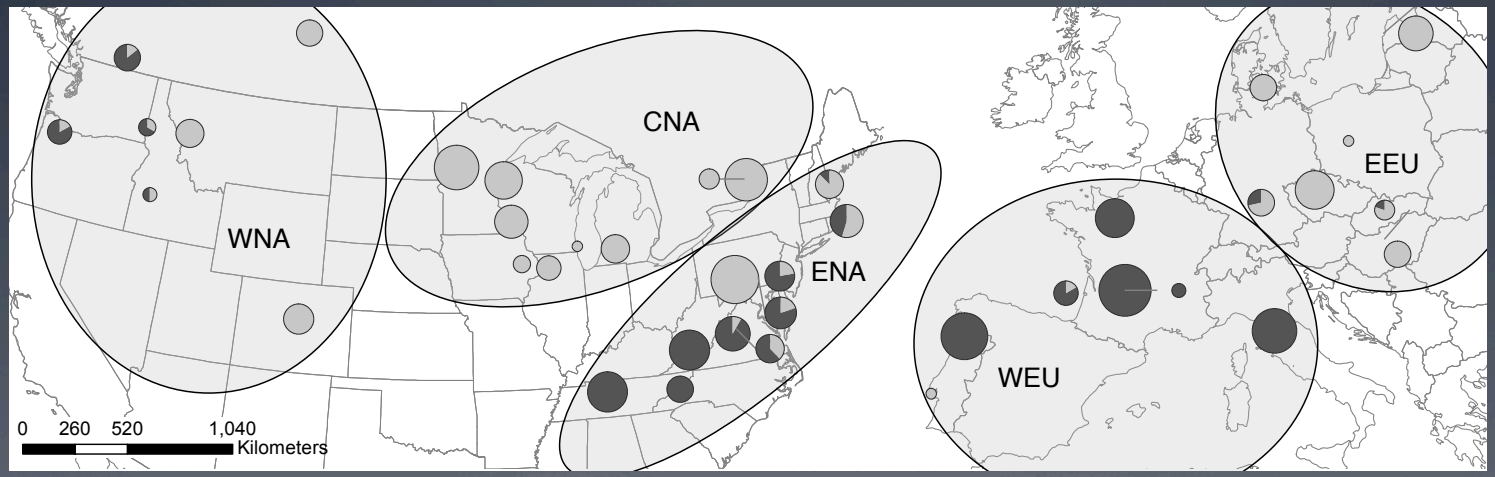


Powerful, but messy if you don't know what you are doing



Made with ArcGIS

Difficult to control what appears in the legend
Annoying lines if you tell points to not overlap

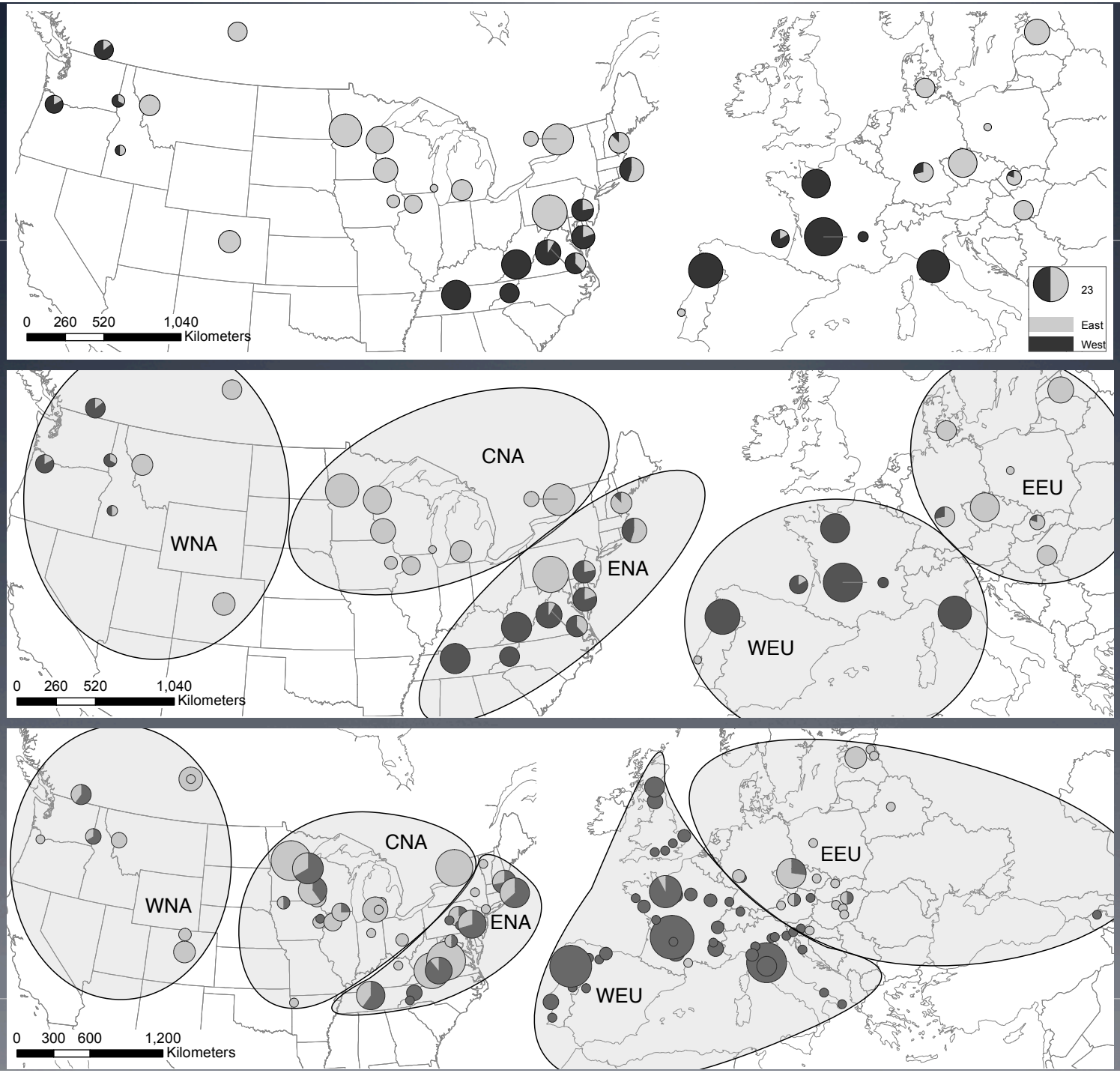


Not easily reproducible

Bits of Canada above Maine now missing

Legend forgotten

Text and shading added via photoshop


0 260 520 1,040
Kilometers
23
East
West
WNA
CNA
ENA
WEU
EEU
0 260 520 1,040
Kilometers
WNA
CNA
ENA
WEU
EEU
0 300 600 1,200
Kilometers

Created with R

```
library(maps)

library(mapdata)

map("worldHires","Canada”,
    xlim=c(-141,-53),
    ylim=c(40,85),
    col="gray90",
    fill=TRUE)
```

Only three lines of code!

Created with R

```
library(maps)

library(mapdata)

library(mapproj)

map("worldHires","Canada”,
    xlim=c(-141,-53),
    ylim=c(40,85),
    col="gray90",
    fill=TRUE,
    projection="conic",
    param=35))
```

Tools of the Trade

```
library(sp)          #classes and methods for spatial
     data

library(maptools)    #tools for reading and handling
     spatial objects

gpclibPermit()       #allow general polygon clipping
     library for R - library(gpclib)

library(maps)        #for creating geographical maps

library(mapdata)     #contains basic data to go along
     with 'maps' (topographic and geologic)

library(sfsmisc)     #utilities from Seminar fuer
     Statistik ETH Zurich

library(mapproj)     #for creating projected maps

library(raster)      #tools to deal with raster maps

library(rgeos)       #interface to geometry engine - open
     source (GEOS)

library(rgdal)       #bindings for the geospatial data
     abstraction library
```

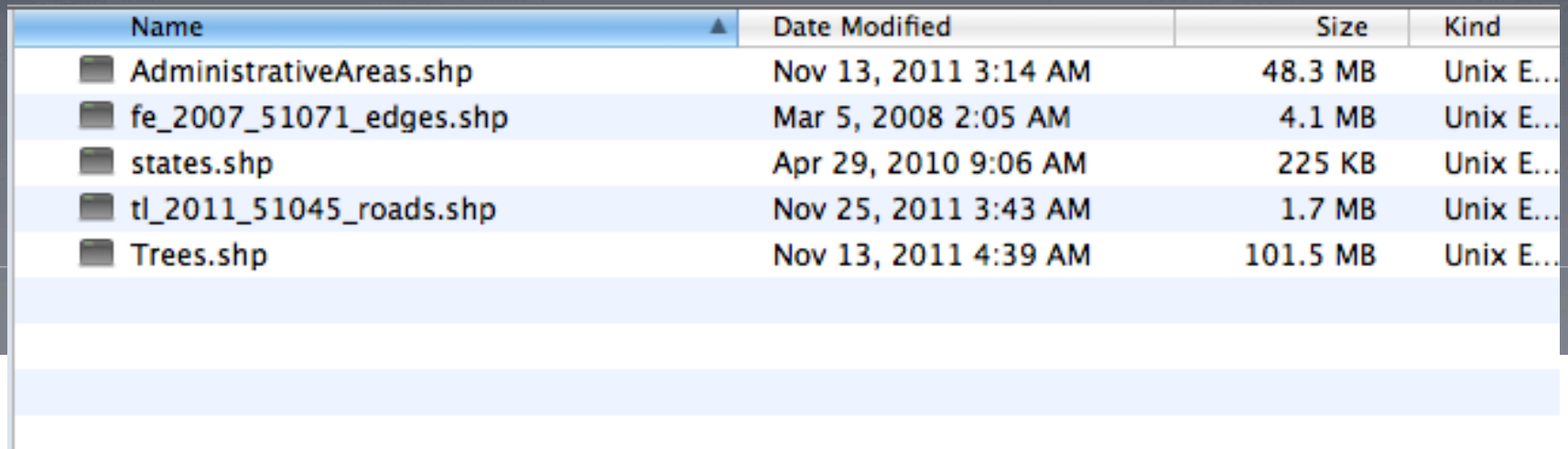
```
library(scales)          #for transparency

library(mapplots)        #data visualization on maps

library(RgoogleMaps)     #overlays on Google map
 tiles in R

library(plotGoogleMaps) # plot SP data as HTML map
 mashup over Google Maps

library(ggmap)           #a package for spatial
 visualization with Google Maps and OpenStreetMap

library(GEOmap)          #topographic and geologic
 mapping

library(plotrix)         #various plotting functions
```

Tools of the Trade

<>	A	B	C	D	E	F	G	H	I	J	K	L
1	id	Section	Psilon	Year	GenoPop	Colonist	Long	Lat	MPWFst	Ho	Hs	AvgPWDist
2	178	2	2.1.a_0.075_L	2008	44	N	-80.66686	37.37894	0.097046535	0.421373	0.538433	2227.115275
3	177	2	2.1.a_0.15_R	2008	29	N	-80.66777	37.37812	0.085401767	0.411518	0.553356	2109.211584
4	175	2	2.1.a_0.175_R	2008	44	N	-80.66808	37.37784	0.070897837	0.46801	0.609394	2072.205611
5	174	2	2.1.a_0.2_R	2008	15	N	-80.66831	37.37753	0.068583628	0.471678	0.574605	2037.643722
6	173	2	2.1.a_0.225_R	2008	4	N	-80.66855	37.37722	0.082717837	0.458333	0.614583	2004.630927
7	172	2	2.1.a_0.25_R	2008	8	N	-80.6688	37.37693	0.073491023	0.499479	0.643266	1972.887567
8	171	2	2.1.a_0.3_L	2008	5	N	-80.66945	37.37642	0.049795023	0.440625	0.559896	1912.836065
9	169	2	2.1.a_0.475_L	2008	21	N	-80.6714	37.37444	0.09771807	0.407688	0.454828	1732.448993
10	168	2	2.1.a_0.5_L	2008	8	N	-80.67168	37.37415	0.121656419	0.355655	0.43125	1709.509788
11	167	2	2.1.a_0.675_R	2008	27	N	-80.67275	37.37152	0.098669698	0.407995	0.529872	1542.57989
12	170	2	2.1.a.d_0.4_L	2008	1	N	-80.6706	37.37532	na	nan	nan	1807.950671
13	165	2	2.1.b_0.1_L	2008	6	N	-80.67546	37.36725	0.08828314	0.383333	0.519792	1329.111698
14	166	2	2.1.b_0.1_R	2008	36	N	-80.67546	37.36725	0.107982791	0.407894	0.587555	1329.111698
15	176	2	2.1.c_0.05_L	2008	2	N	-80.67655	37.3664	0.099811667	0.466667	0.55	1315.394062
16	163	2	2.1.c_0.125_O	2008	24	N	-80.67752	37.36545	0.090377209	0.452093	0.588277	1308.273359
17	162	2	2.1.c_0.125_R	2008	10	N	-80.67752	37.36545	0.069233977	0.408705	0.517758	1308.273359
18	161	2	2.1.c_0.175_O	2008	1	N	-80.67805	37.36481	na	nan	nan	1308.273359
19	160	2	2.1.c_0.2_L	2008	16	N	-80.67831	37.36449	0.071778605	0.422897	0.554038	1310.485077
20	159	2	2.1.c_0.225_L	2008	3	N	-80.67858	37.36417	0.079142	0.395833	0.578125	1314.940699
21	157	2	2.1.c_0.225_O	2008	10	N	-80.67858	37.36417	0.102910116	0.37567	0.482651	1314.940699
22	158	2	2.1.c_0.3_L	2008	8	N	-80.67937	37.36321	0.080975907	0.410714	0.555022	1334.599972
23	155	2	2.1.c_0.3_O	2008	19	N	-80.67937	37.36321	0.106547395	0.449052	0.516264	1334.599972
24	154	2	2.1.c_0.375_O	2008	44	N	-80.68017	37.36225	0.050051465	0.406705	0.549239	1365.074004
25	151	2	2.1.c_0.4_O	2008	20	N	-80.68021	37.36188	0.053549535	0.452222	0.556016	1377.773431
26	148	2	2.1.c_0.425_O-A	2008	47	N	-80.68014	37.36152	0.061754256	0.439633	0.560699	1391.845722
27	140	2	2.1.c_0.475_O	2008	39	N	-80.68055	37.36081	0.062428256	0.42769	0.563793	1434.515955
28	142	2	2.1.c_0.55_O-A	2008	25	N	-80.68139	37.35983	0.05426314	0.39767	0.535209	1509.410238
29	138	2	2.1.c_0.55_O-B	2008	22	N	-80.68139	37.35983	0.052365837	0.424191	0.556784	1509.410238
30	137	2	2.1.c_0.625_L	2008	1	N	-80.68229	37.35891	na	nan	nan	1599.317256
31	136	2	2.1.c_0.65_R	2008	10	N	-80.68264	37.35865	0.063571163	0.361607	0.541282	1630.025902
32	237	2	2.1.c.d_0.325_L	2008	11	N	-80.67964	37.36289	0.109763023	0.379025	0.445636	1343.407778
33	153	2	2.1.c.d_0.4_L	2008	2	N	-80.68021	37.36188	0.096545023	0.678571	0.589286	1377.773431
34	238	2	2.1.c.d_0.425_L	2008	8	N	-80.68014	37.36152	0.052602558	0.357292	0.50372	1391.845722
35	242	2	2.1.c.d_0.45_R	2008	18	N	-80.68027	37.36114	0.053395744	0.428128	0.573907	1414.284078
36	243	2	2.1.c.d_0.475_L	2008	21	N	-80.68055	37.36081	0.053427256	0.465774	0.570125	1434.515955
37	146	2	2.1.c.d_0.5_L	2008	2	N	-80.68083	37.36049	0.068968581	0.466667	0.6	1456.448179
38	141	2	2.1.c.d_0.525_R	2008	24	N	-80.68111	37.36016	0.063013279	0.467219	0.569954	1481.818521
39	139	2	2.1.c.d_0.55_L	2008	19	N	-80.68139	37.35983	0.062658395	0.377843	0.528138	1509.410238
40	143	2	2.1.c.d_0.575_R	2008	7	N	-80.68167	37.35951	0.05934	0.368304	0.532887	1538.512167
41	245	2	2.1.c.d_0.6_L	2008	50	N	-80.68195	37.35918	0.068127233	0.426269	0.528401	1569.54942

(Optional) Tools of the Trade

- Shapefiles
  - Used for adding additional layers of data: roads, cities, waterways, species ranges, temperature or precipitation gradients…..
- These do not come with R
  - A good amount of GIS data is freely available online
    - ESRI, USGS, GeoBC, GeoGratis
  - Google is your best friend



Name	Date Modified	Size	Kind
AdministrativeAreas.shp	Nov 13, 2011 3:14 AM	48.3 MB	Unix E…
fe_2007_51071_edges.shp	Mar 5, 2008 2:05 AM	4.1 MB	Unix E…
states.shp	Apr 29, 2010 9:06 AM	225 KB	Unix E…
tl_2011_51045_roads.shp	Nov 25, 2011 3:43 AM	1.7 MB	Unix E…
Trees.shp	Nov 13, 2011 4:39 AM	101.5 MB	Unix E…

Getting started

1. Load libraries
2. Read in data
3. Plot data

Getting started

1. Load libraries
2. Read in data
3. Plot data

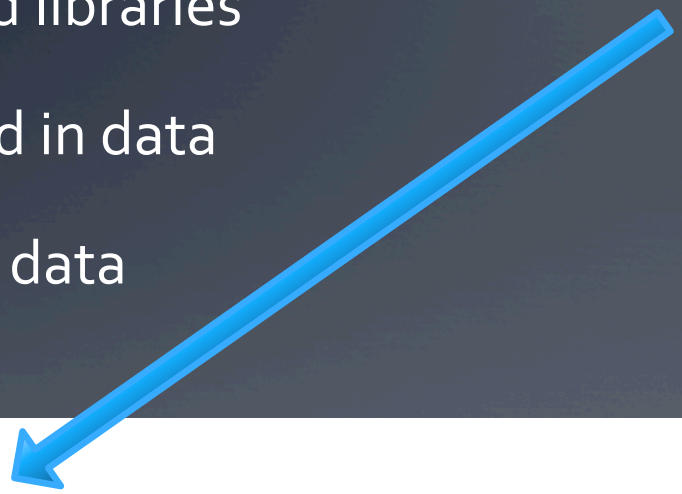
```
library(maps)

library(mapdata)
```

```
map("worldHires","Canada”, xlim=c(-141,-53), ylim=c(40,85), col="gray90”, fill=TRUE)
```

Getting started

1. Load libraries
2. Read in data
3. Plot data

Load libraries
(in this case that loads our data as well)



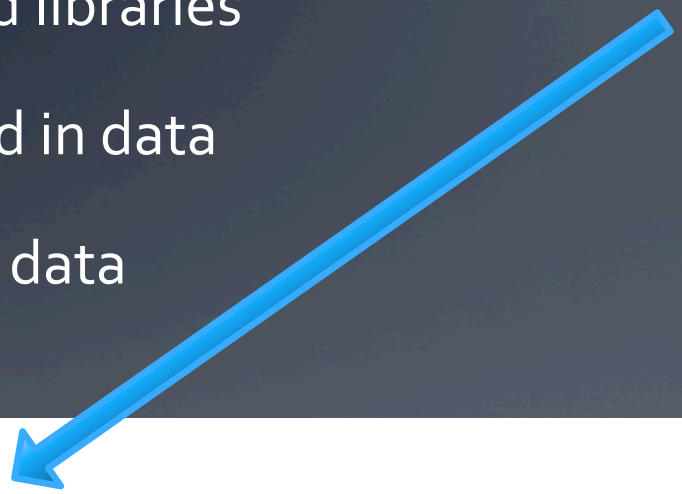
```
library(maps)

library(mapdata)
```

```
map("worldHires","Canada”, xlim=c(-141,-53), ylim=c(40,85), col="gray90”, fill=TRUE)
```

Getting started

1. Load libraries
2. Read in data
3. Plot data

Load libraries
(in this case that loads our data as well)



```
library(maps)

library(mapdata)
```

The 'map' function draws the map

```
map("worldHires","Canada", xlim=c(-141,-53), ylim=c(40,85), col="gray90", fill=TRUE)
```

Getting started

1. Load libraries
2. Read in data
3. Plot data

Load libraries
(in this case that loads our data as well)

Draw this data: "draw the country of Canada, out of the database 'worldHires'"

```
library(maps)

library(mapdata)
```

The 'map' function draws the map

```
map("worldHires","Canada", xlim=c(-141,-53), ylim=c(40,85), col="gray90", fill=TRUE)
```

Getting started

1. Load libraries
2. Read in data
3. Plot data

Load libraries
(in this case that loads our data as well)

Draw this data: "draw the country of Canada, out of the database 'worldHires'"

Draw the map to cover this area: from longitude -53 to -141 and latitude 40 to 85

```
library(maps)

library(mapdata)
```

The 'map' function draws the map

```
map("worldHires","Canada", xlim=c(-141,-53), ylim=c(40,85), col="gray90", fill=TRUE)
```

*Note the order!
xlim is for LONGITUDE
ylim is for LATITUDE

Getting started

1. Load libraries
2. Read in data
3. Plot data

Load libraries
(in this case that loads our data as well)

Draw this data: "draw the country of Canada, out of the database 'worldHires'"

Draw the map to cover this area: from longitude -53 to -141 and latitude 40 to 85

```
library(maps)

library(mapdata)
```

The 'map' function draws the map

Make the map gray

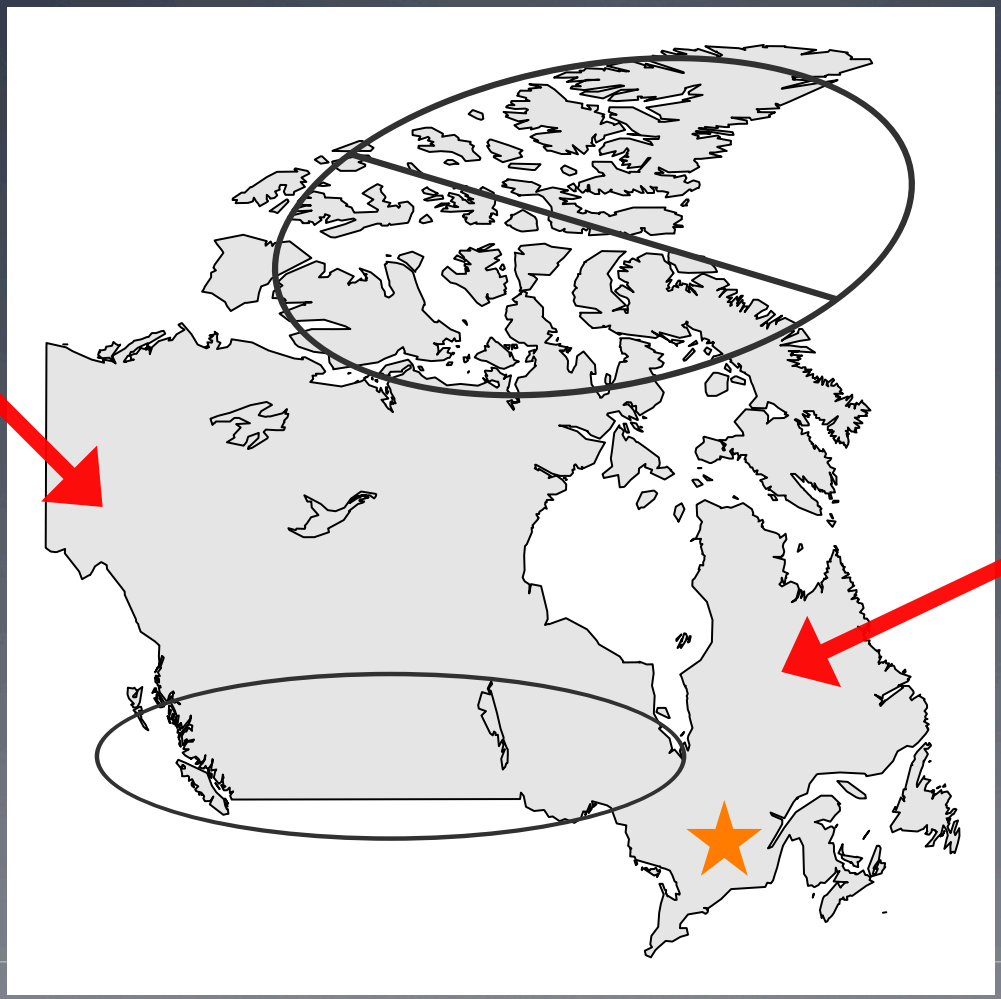
Fill the map with gray (rather than an empty outline of gray)

```
map("worldHires","Canada", xlim=c(-141,-53), ylim=c(40,85), col="gray90", fill=TRUE)
```

```
library(maps)

library(mapdata)

map("worldHires","Canada”, xlim=c(-141,-53), ylim=c(40,85), col="gray90”, fill=TRUE)
```

You could stop here and throw everything else on with powerpoint…

But we won't do that, onward with R!



A more involved example: 1. Load libraries

```
library(sp)          #classes and methods for spatial data

library(maptools)    #tools for reading and handling  spatial objects

gpclibPermit()       #allow general polygon clipping library for R - library(gpclib)

library(maps)        #for creating geographical maps

library(mapdata)     #contains basic data to go along with 'maps' (topographic and geologic)

library(sfsmisc)     #utilities from Seminar fuer Statistik ETH Zurich

library(mapproj)     #for creating projected maps

library(raster)      #tools to deal with raster maps

library(rgeos)       #interface to geometry engine - open source (GEOS)

library(rgdal)       #bindings for the geospatial data abstraction library

library(scales)      #for transparency
```

2. Read in data: shapefiles and GPS points

- 'readShapePoly' – read in a polygon shape layer
  - This means the layer is of type "polygon"
  - i.e. not lines, such as roads or rivers
- 'readShapeLines' ; 'readShapePoints' – read in a line or point shape layer

```
pcontorta <- readShapePoly("~/Documents/My Documents/UBC/Research/LodgepoleData/pinucont/pinucont.shp")

roads2011 <- readShapeLines("/Users/Kim/Documents/My Documents/Taylor Lab/SideProject/R_Analyses/Maps/
    MiscGISdata/VA_2011_PriSecRoads/tl_2011_51_prisecroads.shp")
```

2. Read in data: shapefiles and GPS points

- 'readShapePoly' – read in a polygon shape layer
  - This means the layer is of type "polygon"
  - i.e. not lines, such as roads or rivers
- 'readShapeLines' ; 'readShapePoints' – read in a line or point shape layer

```
pcontorta <- readShapePoly("~/Documents/My Documents/UBC/Research/LodgepoleData/pinucont/pinucont.shp")

roads2011 <- readShapeLines("/Users/Kim/Documents/My Documents/Taylor Lab/SideProject/R_Analyses/Maps/
    MiscGISdata/VA_2011_PriSecRoads/tl_2011_51_prisecroads.shp")
```

- Read in your data, as you would any other .csv file
  - For GPS points, must have columns individually for latitude and longitude
  - **Must be in decimal degrees**

```
samps <- read.csv("~/Documents/My Documents/UBC/Research/FieldWork/Summer2012/
    FieldData_2012.csv", header=TRUE, stringsAsFactors=T, strip.white=TRUE, na.strings=c("NA",
    "na", "nan", "inf", "", "."))
```

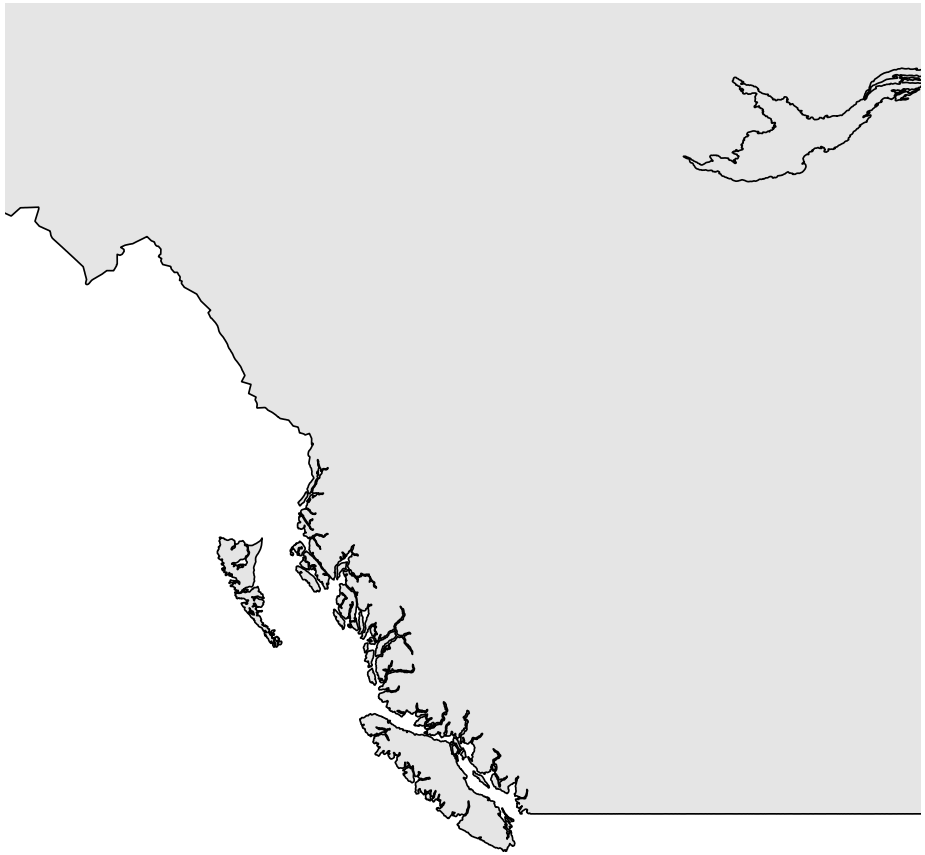
3. Create the Map

- Several steps:
  1. Plot the base map
     - This will be the final shape and size of your full map
  2. Plot layers or data onto this map
     - Add your shape files or data points
     - Plotting occurs in order, unless you use transparency, you will cover things up
  3. Add any final touches you'd like
- *Remember to use "add=TRUE" for every step after the first

Let's map the range of lodgepole pine and some sample sites:

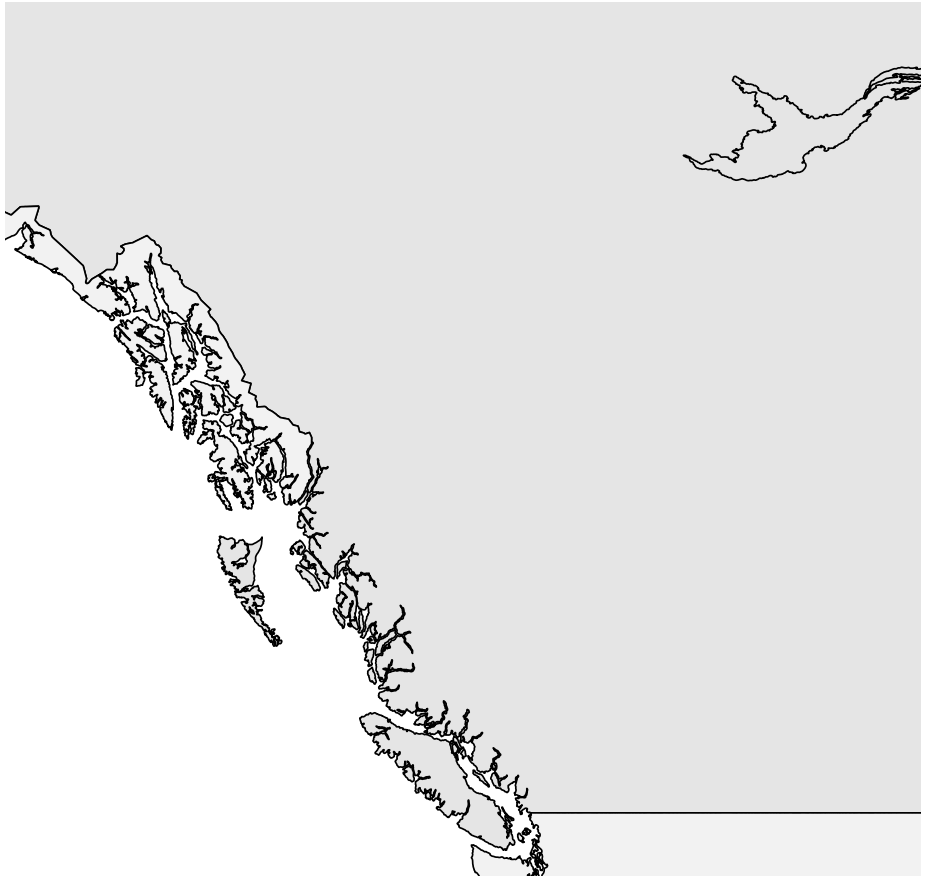
1. Plot Map

```
map("worldHires",
   "Canada",
   xlim=c(-140,-110),
   ylim=c(48,64),
   col="gray90",
   fill=TRUE)
```



1. Plot Map

```
map("worldHires",
   "usa",
   xlim=c(-140,-110),
   ylim=c(48,64),
   col="gray95",
   fill=TRUE,
   add=TRUE)
```



2. Add layers

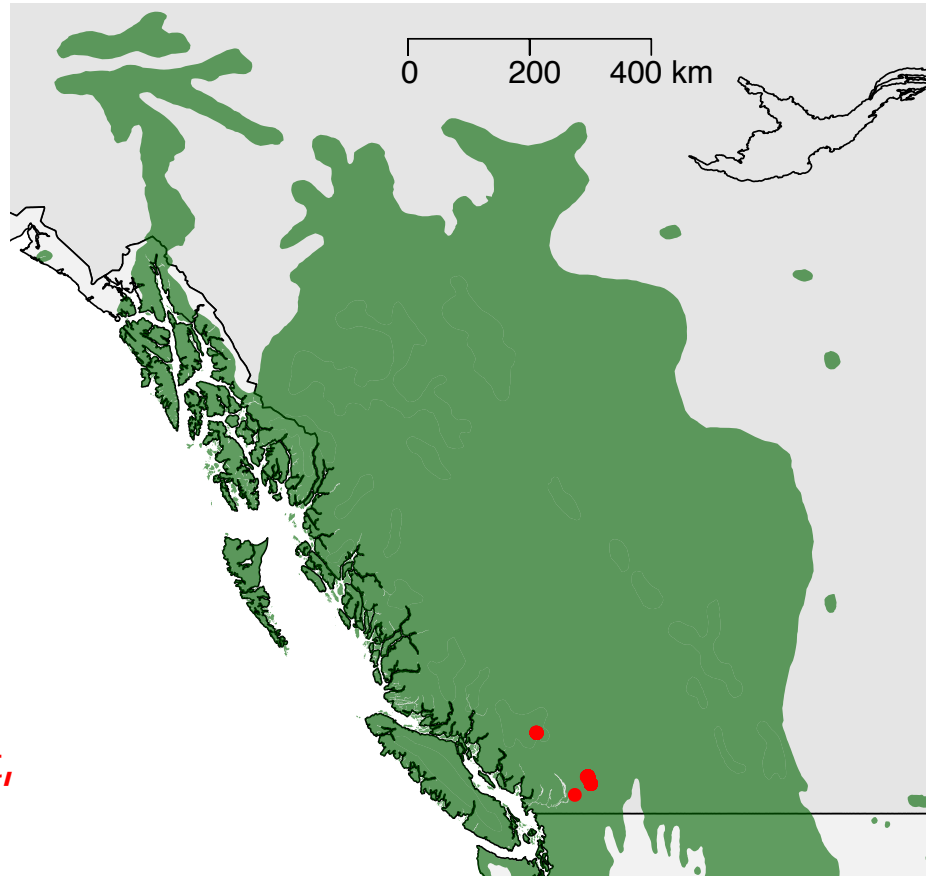
```
plot(pcontorta,
   add=TRUE,
   xlim=c(-140,-110),
   ylim=c(48,64),
   col=alpha("darkgreen",
    0.6),
   border=FALSE)
```

2. Add layers

```
points(samps$Long,
    samps$Lat,
    pch=19,
    col="red",
    cex=1)
```

3. Add extras

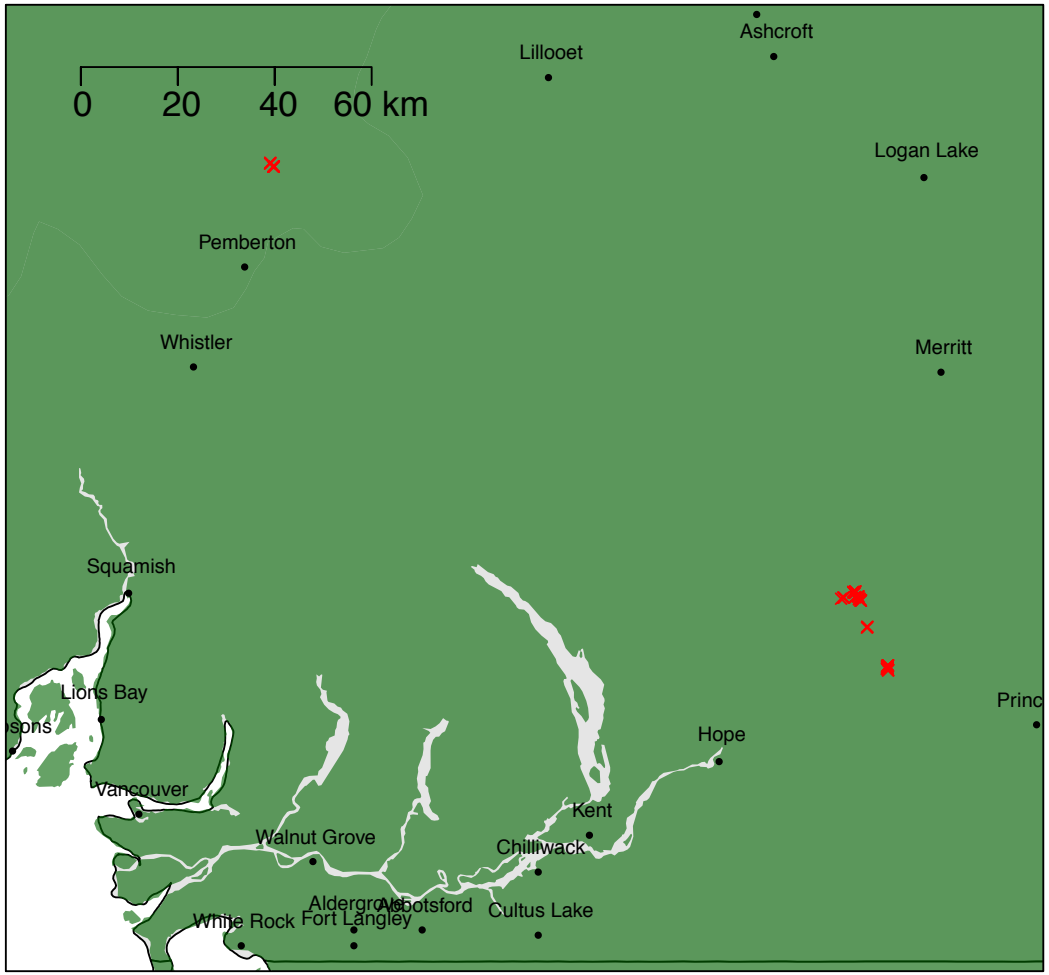
```
map.scale(-127, 63.5,
    ratio=FALSE,
    relwidth=0.2,
    cex=1.2)
```



*Note: This is an unprojected map, placement of the scale bar is important, as it will change size north to south

Zoomed In

```
#plot map
map("worldHires","Canada", xlim=c(-123.5,-120.5),ylim=c(49,50.8), col="gray90", fill=TRUE)
map("worldHires","usa", xlim=c(-140,-110),ylim=c(48,64), col="gray95", fill=TRUE, add=TRUE)

#plot species range
plot(pcontorta, add=TRUE,
xlim=c(-140,-110),
ylim=c(48,64), col=alpha("darkgreen",
0.6), border=FALSE)

#map cities
map.cities(country="Canada",
label=TRUE, cex=1, xlim=c(-140,-110),
ylim=c(48,64),
pch=20)

#plot field data
points(samps$Long, samps$Lat, pch=4,
col="red", cex=0.8)

#put a scale on the map
map.scale(-123.3, 50.7, ratio=FALSE,
relwidth=0.2, cex=1.2)

box()
```



Create a Projected Map – More Realistic

Unprojected

```
map(database="world", xlim=c
    (-170,-10),ylim=c
    (45,83),resolution=0,
    col="grey80",
    fill=TRUE)
```

```
library(mapproj)

map(database="world",xlim=c
    (-170,-20), ylim=c(65,83),
    projection="gilbert",
    fill=T,col="grey80",
    orientation=c(90,0,225),
    resolution=0)
```

Projected

Plotting points on a projected map

```
#plotting the map, note the 'orientation' which appears to be necessary for
    points and map to match up.
    map(database="world",ylim=c(45,90),xlim=c(-160,-50),
    projection="gilbert", fill=T,col="grey80",  orientation=c(90,0,225),
    resolution=0, lwd=0.5)


    lon<-c(-72,-66,-107,-154) #fake longitude vector
    lat<-c(81.7,64.6,68.3,60) #fake latitude vector
```

Plotting points on a projected map

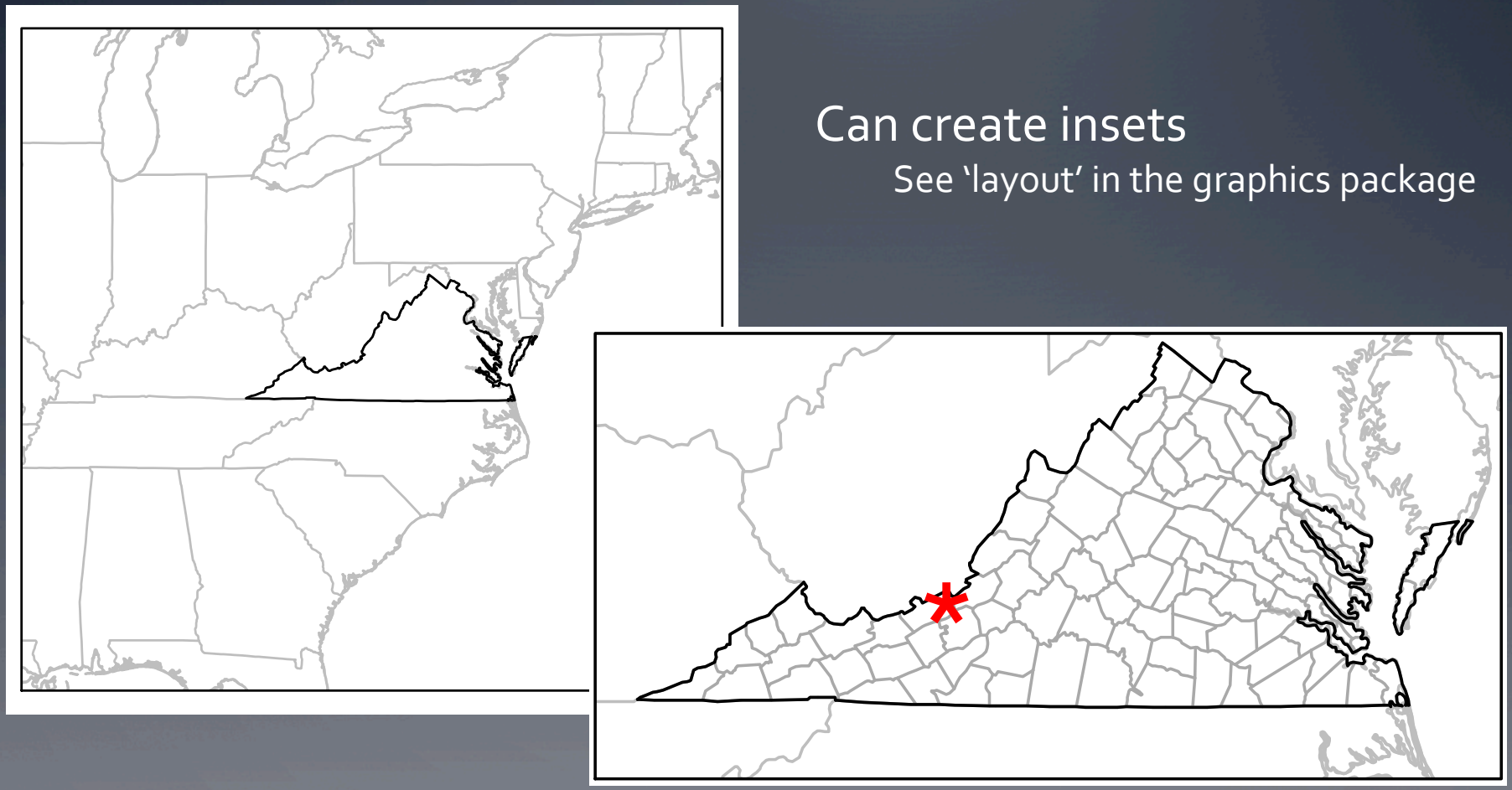
```
#plotting the map, note the 'orientation' which appears to be necessary for
    points and map to match up.
    map(database="world",ylim=c(45,90),xlim=c(-160,-50),
    projection="gilbert", fill=T,col="grey80",  orientation=c(90,0,225),
    resolution=0, lwd=0.5)


    lon<-c(-72,-66,-107,-154) #fake longitude vector
    lat<-c(81.7,64.6,68.3,60) #fake latitude vector

#converting your points
    to projected lat/lon
    coord<-mapproject(
    lon,lat,
    proj="gilbert",
    orientation=c(90,0,225))

    points(coord,
    pch=21,cex=1.2)
    #plot converted points
    onto map
```

Plotting points on a projected map

```
#plotting the map, note the 'orientation' which appears to be necessary for
    points and map to match up.
    map(database="world",ylim=c(45,90),xlim=c(-160,-50),
    projection="gilbert", fill=T,col="grey80",  orientation=c(90,0,225),
    resolution=0, lwd=0.5)


    lon<-c(-72,-66,-107,-154) #fake longitude vector
    lat<-c(81.7,64.6,68.3,60) #fake latitude vector

#converting your points
    to projected lat/lon
    coord<-mapproject(
    lon,lat,
    proj="gilbert",
    orientation=c(90,0,225))

    points(coord,
    pch=21,cex=1.2)
    #plot converted points
    onto map
```

*Note: there is currently no capability to plot a scale bar on a projected map, but grid lines can be plotted

```
map("state", add=TRUE, col="gray")

map("state","Virginia", add=TRUE, col="black")

points(-80.523326, 37.375409, pch="*", col="red", cex=3)
```



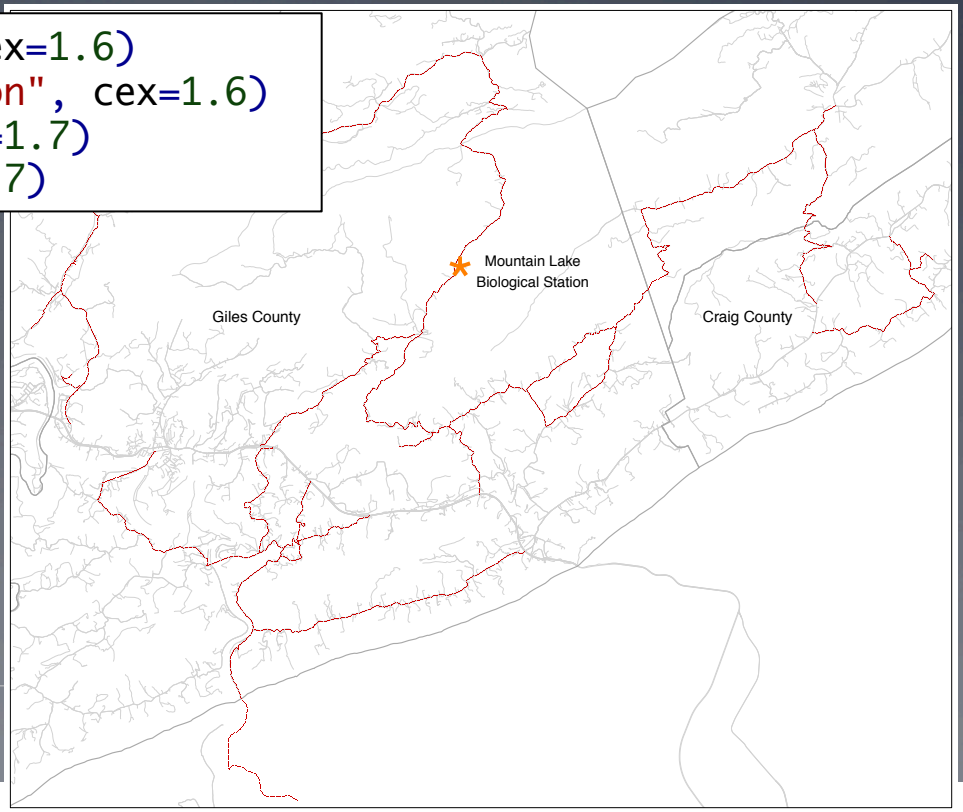
Can create insets

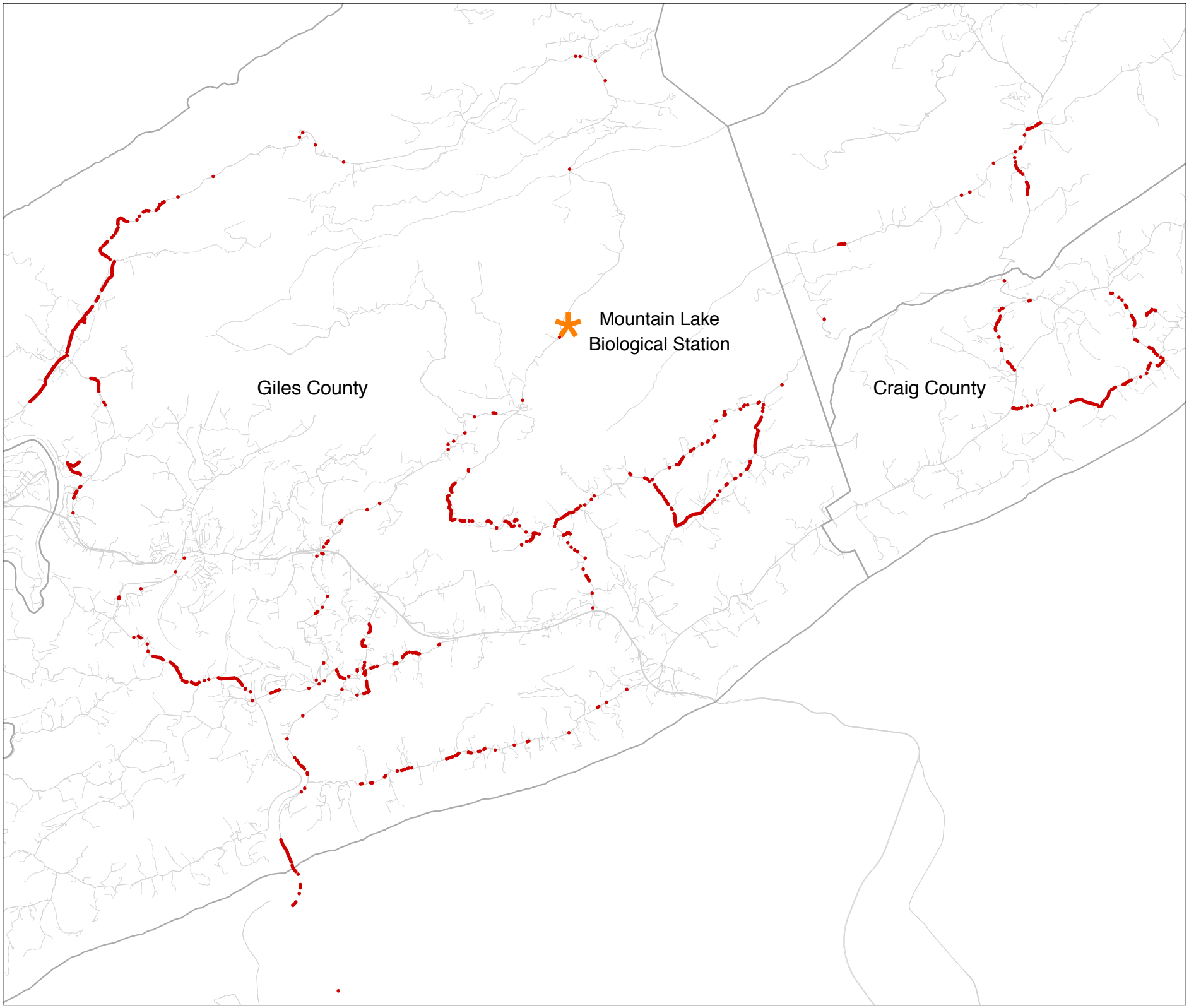
See 'layout' in the graphics package

Adding Text and Detail

```
#main roads, Virginia
plot(roads2011, add=TRUE, xlim=c(-80.69,-80.341),ylim=c(37.215,37.45), col="light gray", lwd=0.8)

#Giles county roads
plot(gilesdetail2, add=TRUE, xlim=c(-80.69,-80.341),ylim=c(37.215,37.45), col="light gray", lwd=0.8)
#Craig county roads
plot(craigdetail, add=TRUE, xlim=c(-80.69,-80.341),ylim=c(37.215,37.45), col="light gray", lwd=0.8)

#Giles county line
plot(gilessub, add=TRUE, xlim=c(-80.69,-80.341),ylim=c(37.215,37.45), col="dark gray", lwd=2)
#Craig county line
plot(craigsub, add=TRUE, xlim=c(-80.69,-80.341),ylim=c(37.215,37.45), col="dark gray", lwd=2)
```

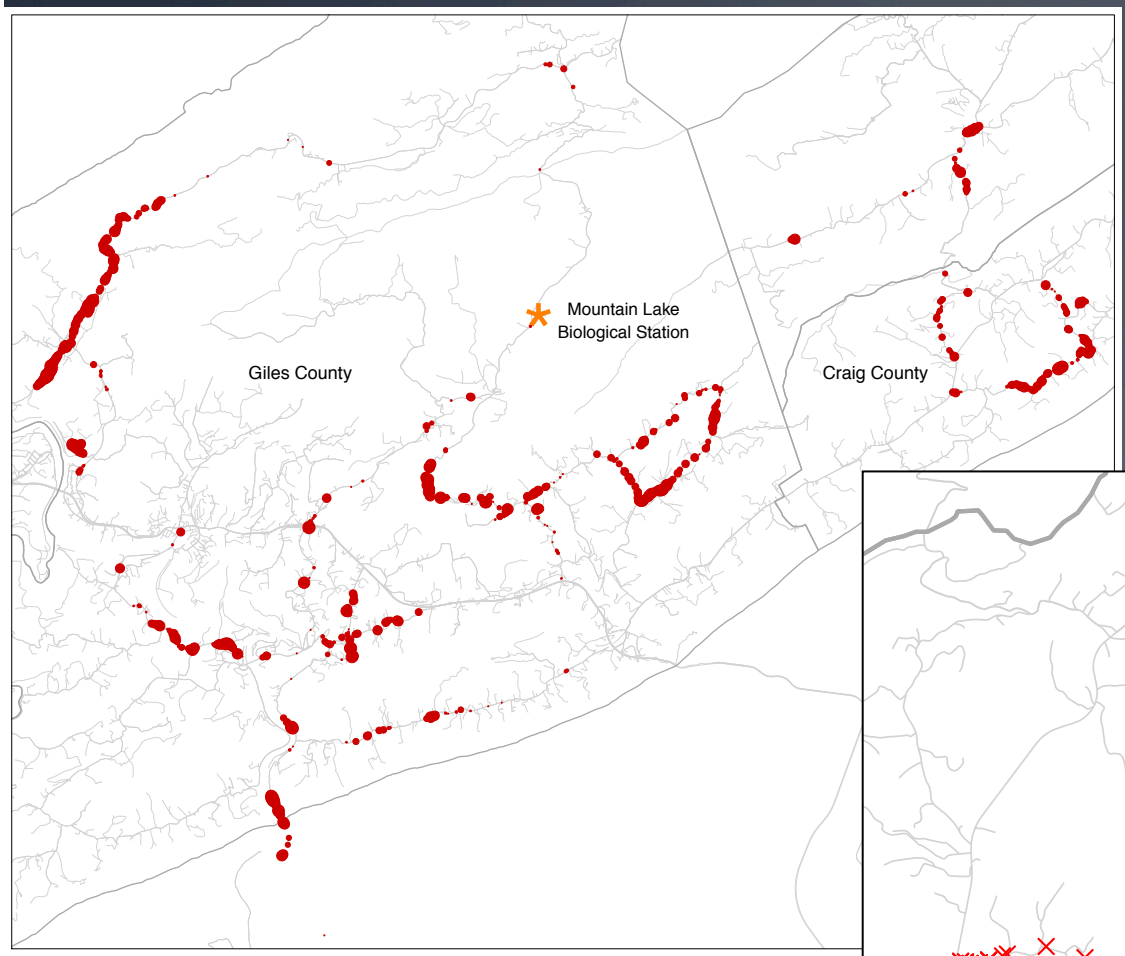
```
text(-80.496,37.377, "Mountain Lake", cex=1.6)
text(-80.496,37.3705, "Biological Station", cex=1.6)
text(-80.415,37.36, "Craig County", cex=1.7)
text(-80.6,37.36, "Giles County", cex=1.7)
```

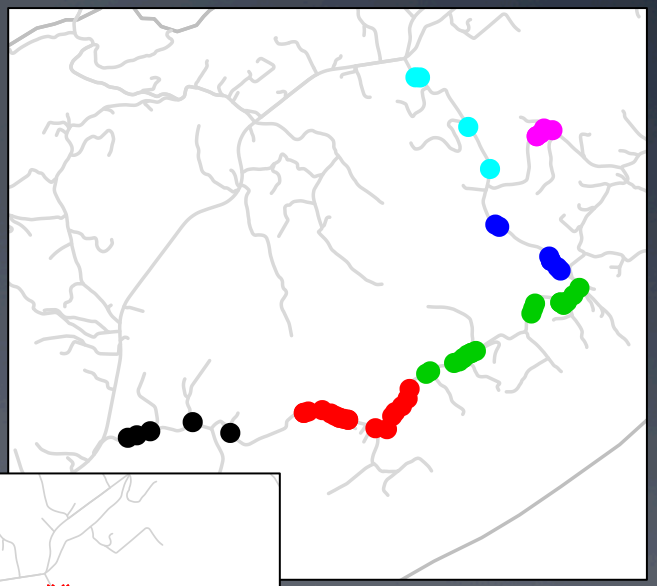


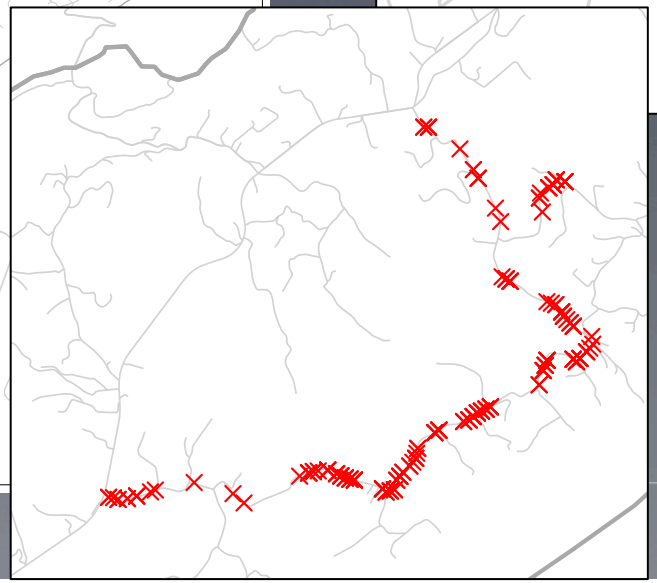

Mountain Lake
Biological Station
Giles County
Craig County

Scaled Point Sizes and Types

```
points(pops2$W, pops2$N, pch=20, col="red3", cex=sqrt(pops2$Plants))

points(pops2$W, pops2$N, pch=20, col=as.numeric(pops2$Subsection),
cex=sqrt(pops2$Plants))
```





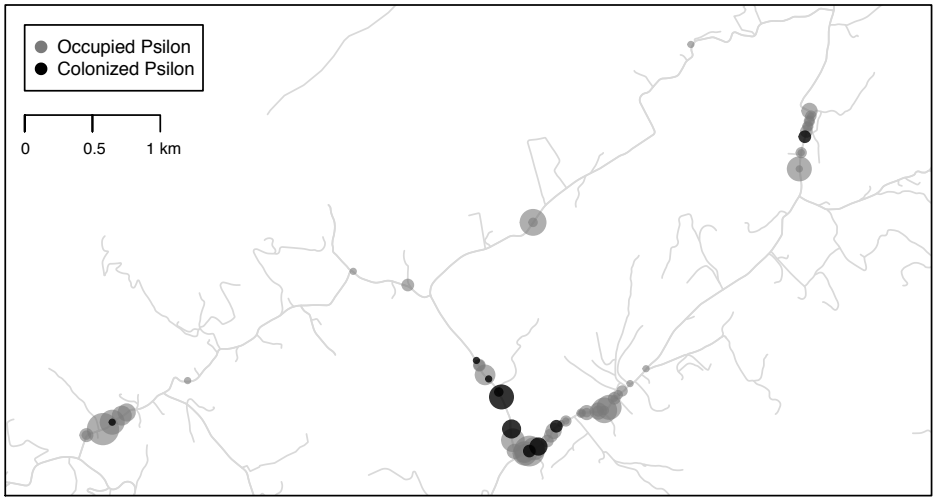


Using Transparency

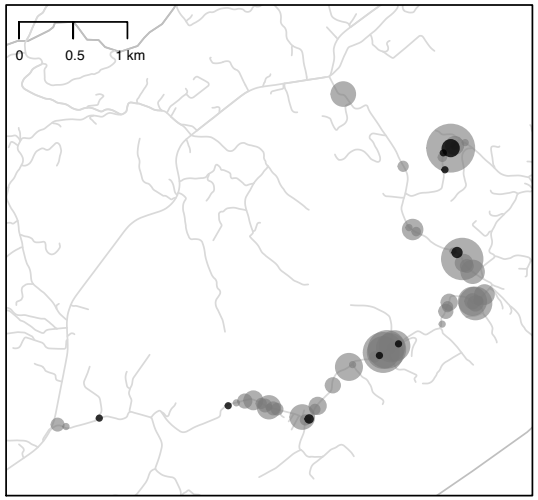
```
library(scales) #for transparency

points(slat08$Long, slat08$Lat, pch=pch20, col=c(alpha(slat08$col2,0.6), alpha
(slat08$col2,0.8)), cex=sqrt(slat08$GenoPop))
```

Section 2

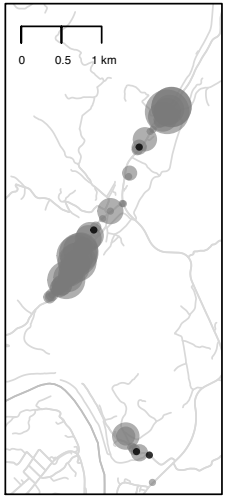
2008



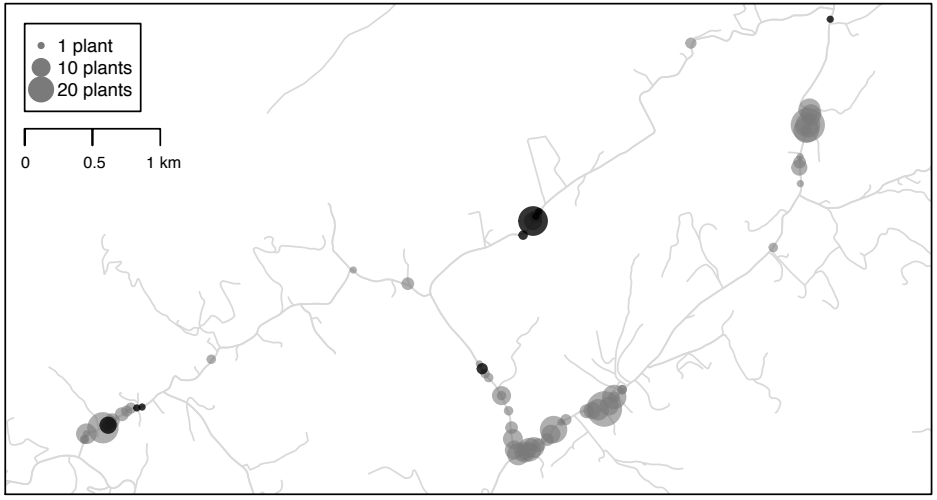
Section 6

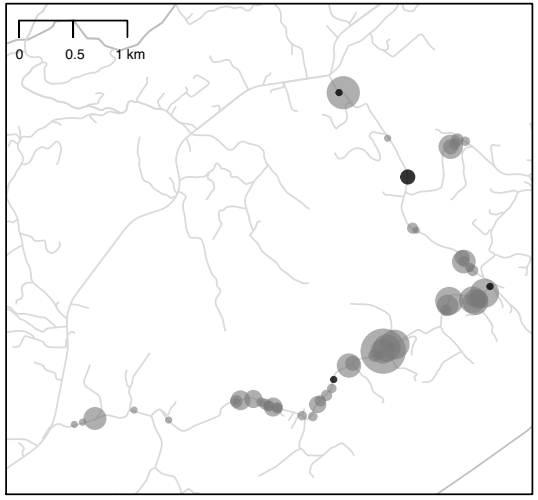


Section 9



2010





Plotting Pie Charts

Libraries: ‘mapplots’ or ‘plotrix’

Functions: ‘add.pie’ or ‘floating.pie’ (respectively)

Plotting Pie Charts

Libraries: 'plotrix' or 'mapplots'

Functions: 'floating.pie' or 'add.pie' (respectively)

```
floating.pie(5, 48, x=c(0.001,5), radius=5, col=c("orange", "blue"))
```

x position (longitude)

y position (latitude)

values for sectors of the pie chart

radius of the pie chart (i.e. scale by this size)

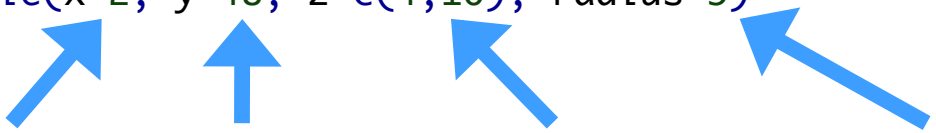
*Note: floating.pie cannot accept zeros (when one pie chart is of only one type)

Plotting Pie Charts

Libraries: ‘mapplots’ or ‘plotrix’

Functions: ‘add.pie’ or ‘floating.pie’ (respectively)

```
add.pie(x=2, y=48, z=c(4,10), radius=5)
```



x position (longitude)

y position (latitude)

values for sectors of the pie chart

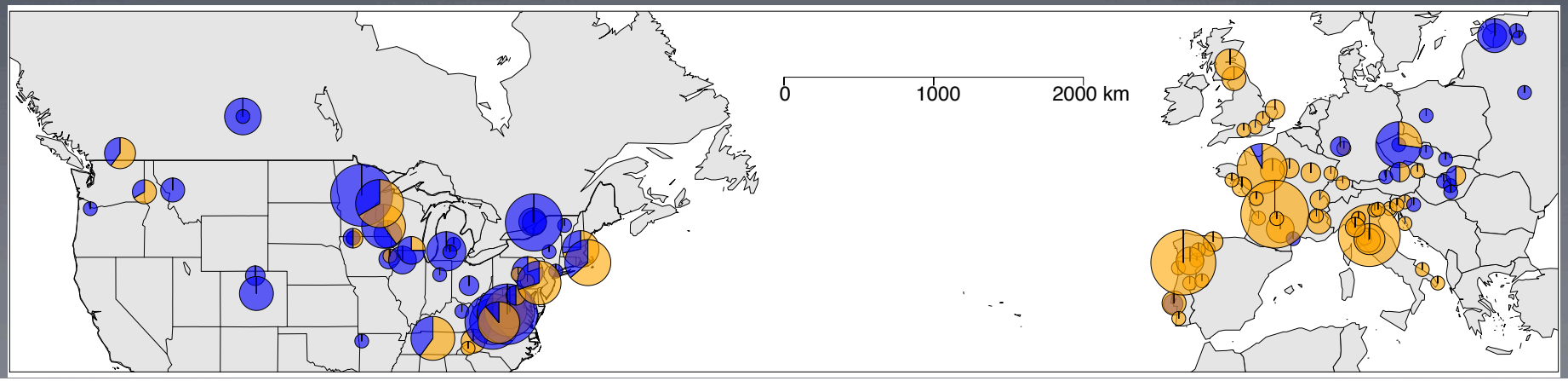
radius of the pie chart (i.e. scale by this size)

Plotting Pie Charts

Libraries: ‘mapplots’ or ‘plotrix’

Functions: ‘add.pie’ or ‘floating.pie’ (respectively)

```
floating.pie(5, 48, x=c(0.001,5), radius=5, col=c("orange", "blue"))

add.pie(x=2, y=48, z=c(4,10), radius=5)
```



Final Touches

- Print to a .pdf, arrange layout, create a border edge

```
pdf("~/Documents/My_Documents/UBC/Research/FieldWork/Summer2012/
SamplingMap_PcontRange.pdf")
```

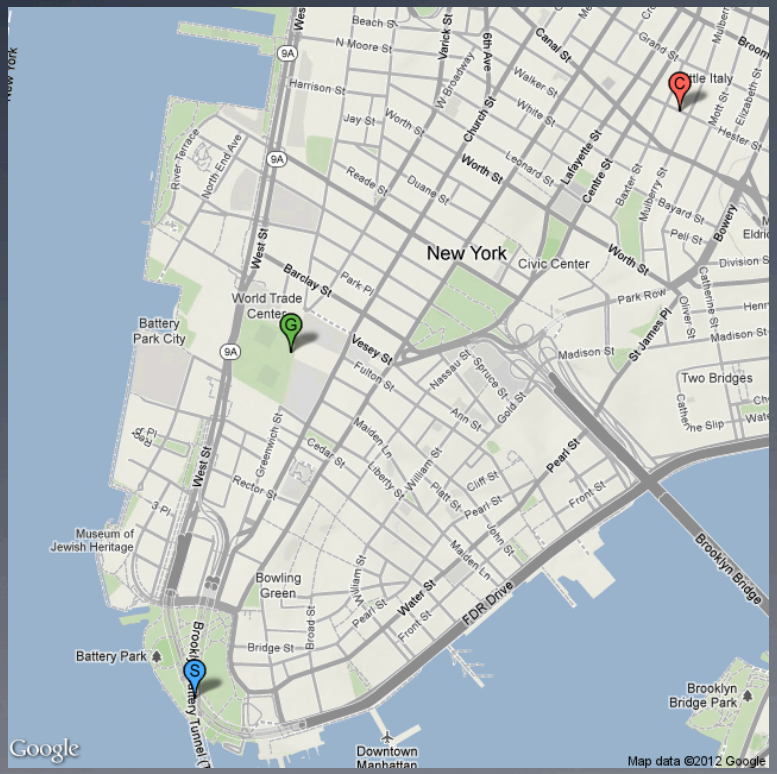
Make the map here

Can use "layout" settings to plot multiple maps together

```
box() #put a box around the map

dev.off()
```

‘RgoogleMaps’

A package in R to plot data onto maps from Google



I'm still new at RgoogleMaps...

Two main functions:

- GetMap()
- GetMap.bbox()

The second one takes care of some of the overhead

Plots straight to an output file

Not sure about shapefiles yet

Another map of BC

```
#define our map's range
lat <- c(48,64)
lon <- c(-140,-110)

#tell what point to center on
center = c(mean(lat), mean(lon))

#zoom: 1 = furthest out (entire globe),
larger numbers = closer in
zoom <- 5

terrmap <- GetMap(
center=center,
zoom=zoom,
maptype= "terrain",
destfile = "terrain.png")


#lots of visual options for maptype, just like
google maps: "roadmap", "mobile", "satellite",
"terrain", "hybrid", "mapmaker-roadmap",
"mapmaker-hybrid”
```

Add our data

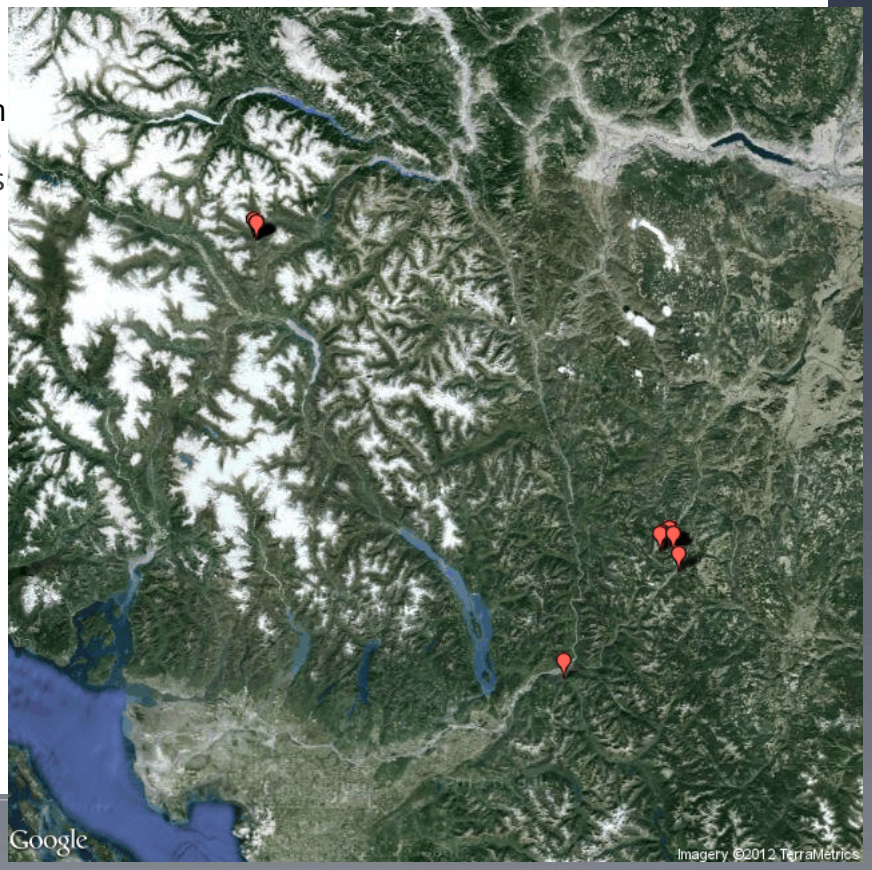
```
samps <- read.csv("~/Documents/My Documents/UBC/Research/FieldWork/Summer2012/
FieldData_2012.csv", header=TRUE, stringsAsFactors=T, strip.white=TRUE, na.strings=c("NA", "na",
"nan", "inf", "", "."))

#want to make a data frame of just points and their visualizations
samps$size <- "small”
samps$col <- "red”
samps$char <- ""
mymarkers <- cbind.data.frame(samps$lat, samps$lon, samps$size, samps$col, samps$char)
names(mymarkers) <- c("lat", "lon", "size", "col", "char")
mymarkers <- mymarkers[c(1:18,50:64),] #taking a subset because it craps out when I use all of
the data
```

Plot our data

```
samps <- read.csv("~/Documents/My Documents/UBC/Research/FieldWork/Summer2012/
FieldData_2012.csv", header=TRUE, stringsAsFactors=T, strip.white=TRUE, na.strings=c("NA", "na",
"nan", "inf", "", "."))

#want to make a data frame of just points and their visualizations
samps$size <- "small”
samps$col <- "red”
samps$char <- ""
mymarkers <- cbind.data.frame(samps$lat, samps$lon, samps$size, samps$col, samps$char)
names(mymarkers) <- c("lat", "lon", "size", "col", "char")
mymarkers <- mymarkers[c(1:18,50:64),] #taking a subset because it craps out when I use all of
the data


lat <- c(48,60)
lon <- c(-140,-110)
center = c(mean(lat), mean(lon))
zoom <- 5
satclose <- GetMap.bbox(lonR= range(lon),
latR= range(lat),
center= c(50, -122),
destfile= "satclose.png",
markers= mymarkers,
zoom=8,
maptype="satellite")
```

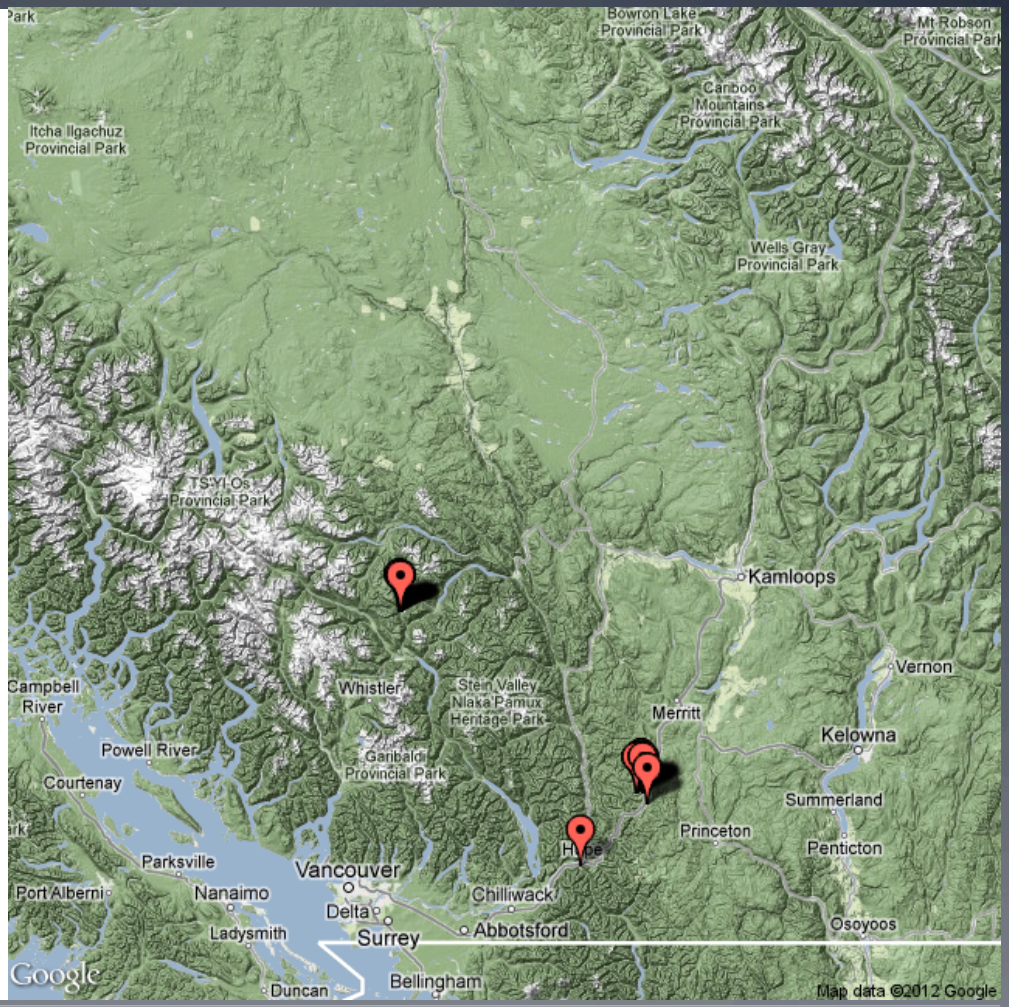
Plot our data

```
samps <- read.csv("~/Documents/My Documents/UBC/Research/FieldWork/Summer2012/
FieldData_2012.csv", header=TRUE, stringsAsFactors=T, strip.white=TRUE, na.strings=c("NA", "na",
"nan", "inf", "", "."))

#want to make a data frame of just points and their visualizations
samps$size <- "small"
samps$col <- "red"
samps$char <- ""
mymarkers <- cbind.data.frame(samps$lat, samps$lon
names(mymarkers) <- c("lat", "lon", "size", "col",
mymarkers <- mymarkers[c(1:18,50:64),] #taking a s
the data


lat <- c(48,60)
lon <- c(-140,-110)
center = c(mean(lat), mean(lon))
zoom <- 5
satclose <- GetMap.bbox(lonR= range(lon),
latR= range(lat),
center= c(50, -122),
destfile= "satclose.png",
markers= mymarkers,
zoom=8,
maptype="satellite")
```



Another version

```
samps$size <- "large"



lat <- c(48,60)
lon <- c(-140,-110)
center = c(mean(lat), mean(lon))
zoom <- 5



terrfar <- GetMap.bbox(
lonR= range(lon),
latR= range(lat),
center= c(51, -122),
destfile= "terrfar.png",
markers= mymarkers2,
zoom=7,
maptype="terrain")
```



And the best part about R, there will be more every day!