Cod and Capelin Abundance in Barents Sea

Janelle Morano

2/28/2020

last updated 3/6/2020

Geostatiscal Procedure

When you have data that are spatially correlated, often, points that are close together are going to be more similar than points that are farther apart. This is autocorrelation. When you have spatial data, the goal is to find trends in the data that you are interested in understanding while accounting for the autocorrelation. You don't want it to muddle your interpretations.

Some terms to remember… **Variance**: how far a set of random numbers are spread out (deviate) from the average value; the square of the standard deviation.

Covariance: the joint variability of two random variables; the tendency in the linear. Random variable whose covariance is zero are uncorrelated, and independent.

Variance-Covariance Matrix: square matrix that contains the variances and covariances associated with variables. The diagonal elements contain the variances of the variables; the off-diagonal elements contain the covariances between all possible pairs of the variables.

Residual: the difference between each observations and the sample mean.

Empirical Variogram: description of the spatial dependence of the data

Actual Variogram: model of the spatial dependence of the data.

Basic Approach

The basic approach to working with geospatial data follows these steps:

1.Plot the locations a. Identify x and y and put it in geographic context b. Plot the measurements at locations 1. Plot the empirical variogram and find the best model and parameters 1. Fit the actual variogram model 1. Make predictions

Not sure when these happen:

- Calculate the variance (matrix of variance)
- Calculate the co-variance (variance-covariance matrix)
- Decompose into upper-triangular matrix

Libraries

You need these.

```
knitr::opts_chunk$set(echo = TRUE)
library(sp)
library(gstat)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
library(geoR)
```

```
## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.7-5.2.1 (built on 2016-05-02) is now loaded
## ----------------------------------------------------------
```

```
library(RColorBrewer)
library(classInt)
```

Plot Data Locations

Step 1: Plot the acoustic survey locations in the Barents Sea.

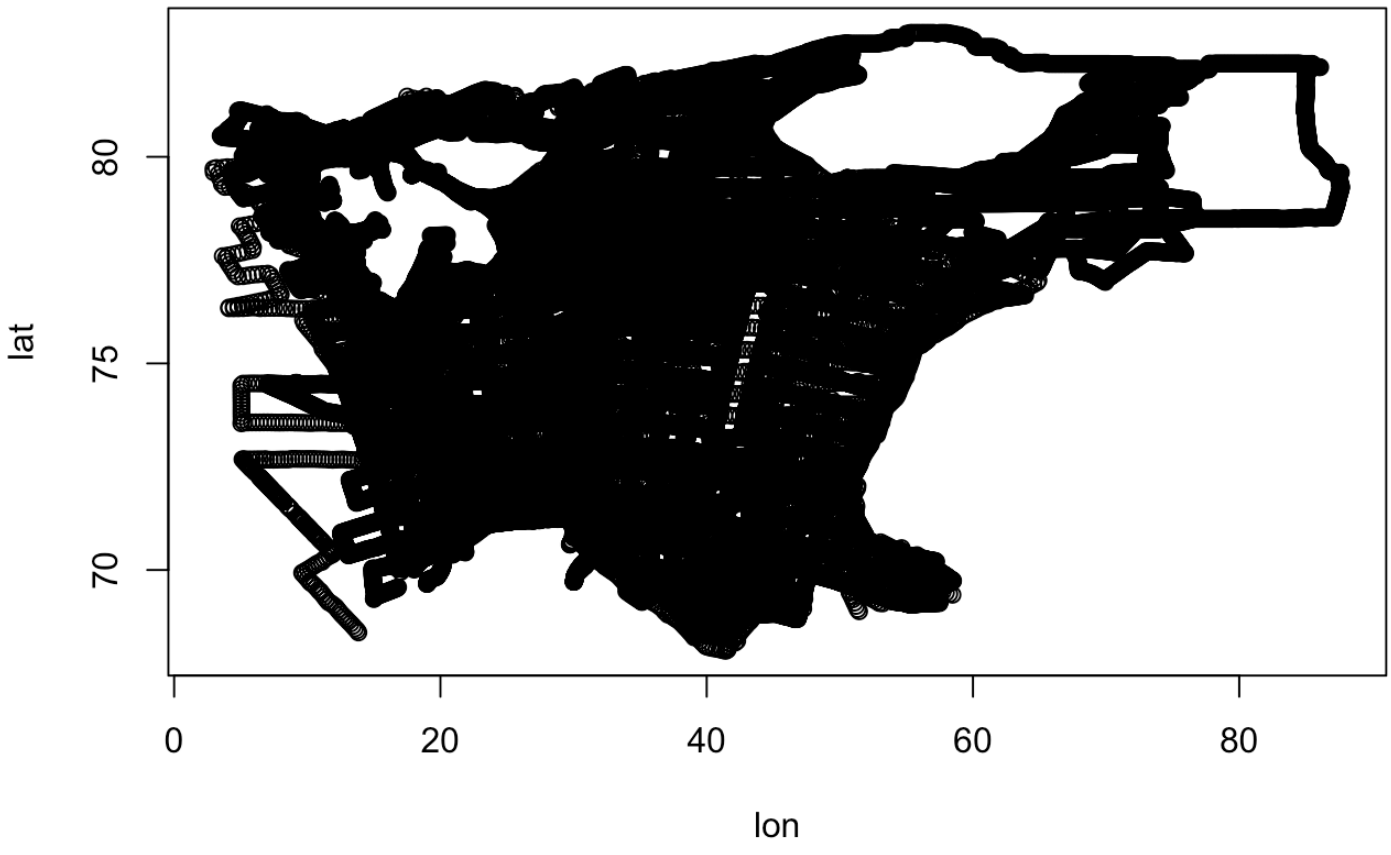
```
data = read.csv("AllData.csv", head = TRUE)
#head(data)
#dim(data)

#make each year a unique number for reference
(iyear = unique(data$year))
```

```
##  [1] 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
```

```
#assign the lat and lon
data$Xloc = data$lon
data$Yloc = data$lat
coordinates(data)=c("Xloc","Yloc")

## Plot survey locations
plot(lat~lon,data=data)
```



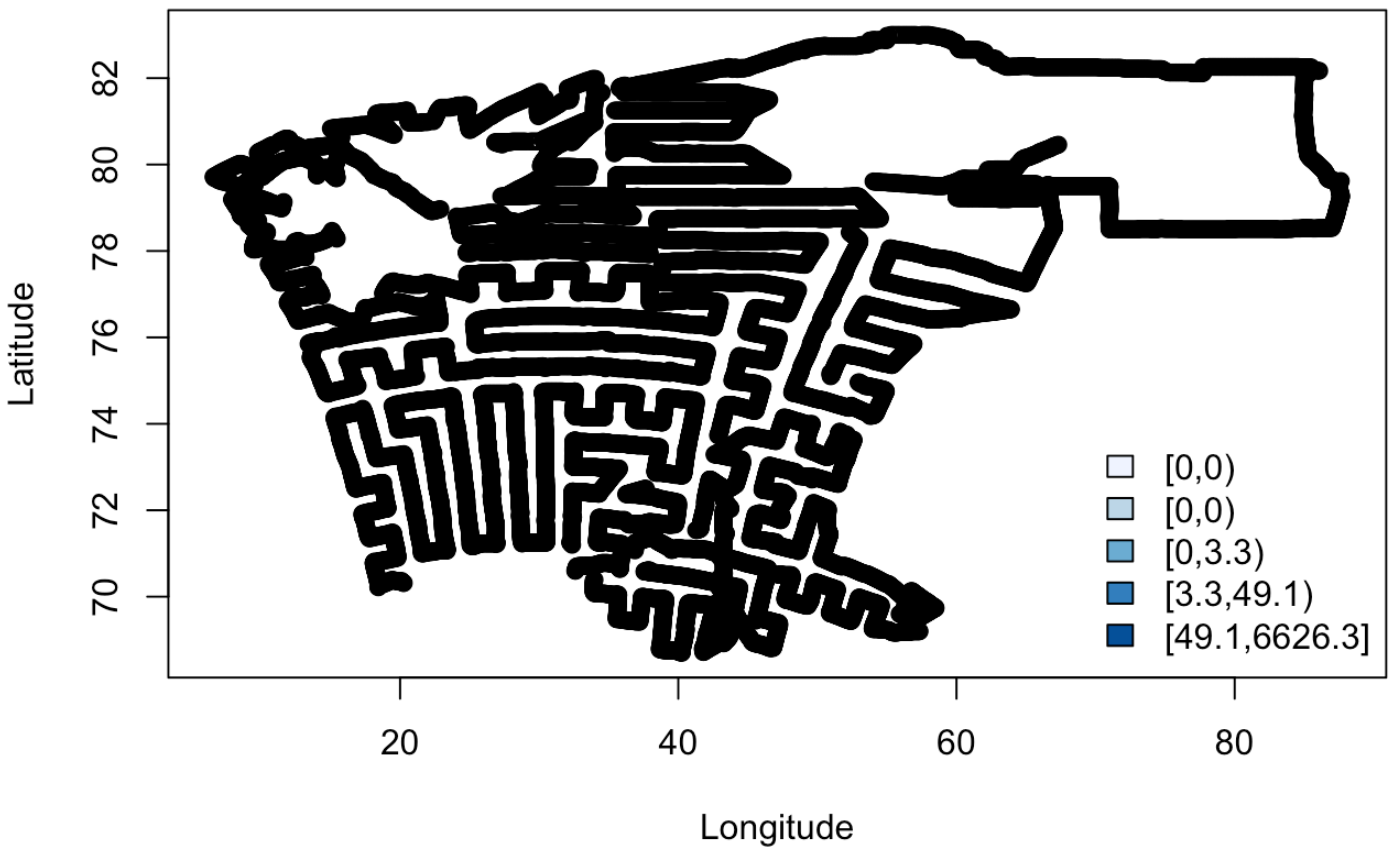
Ok, gotta correct the projection. In the meantime…

I'm just going to look at 2013. Let's look at capelin and cod abundance in 2013.

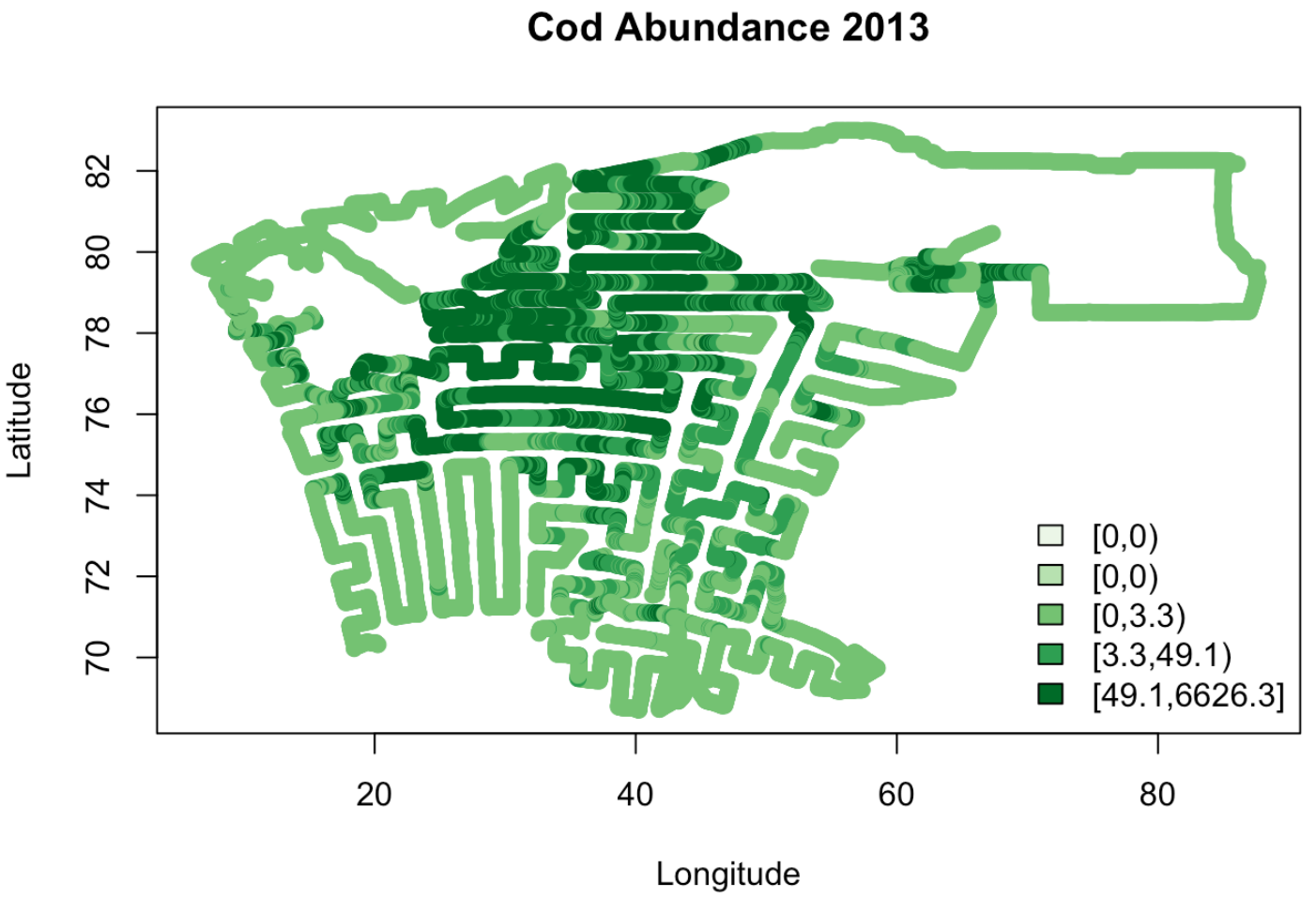
```
# Make a dataset for just 2013 to simply things.
yy = 2013
datai = data[data$year==yy,]

# Set colors
pal = brewer.pal(5,"Blues")
q5 = classIntervals(datai$capelin, n=5, style="quantile")
q5Colors = findColours(q5,pal)

# Plot relative capelin densities for 2013
plot(c(min(datai$Xloc),max(datai$Xloc)),
       c(min(datai$Yloc),max(datai$Yloc)),
       xlab="Longitude",ylab="Latitude",type="n")
points(datai)#,col=q5Colors,pch=19,add=T) #look at q5$brks
legend("bottomright",fill=attr(q5Colors,"palette"), legend = names(a
ttr(q5Colors,"table")),bty="n")
  title(paste("Capelin Abundance",yy))
```




```
# Plot relative cod densities for 2013
# Set colors
pal2 = brewer.pal(5,"Greens")
q52 = classIntervals(datai$cod, n=5, style="quantile")
q5Colors2 = findColours(q5,pal2)
plot(c(min(datai$Xloc),max(datai$Xloc)),
       c(min(datai$Yloc),max(datai$Yloc)),
       xlab="Longitude",ylab="Latitude",type="n")
plot(datai,col=q5Colors2,pch=19,add=T)
legend("bottomright",fill=attr(q5Colors2,"palette"), legend = names(
attr(q5Colors2,"table")),bty="n")
title(paste("Cod Abundance",yy))
```



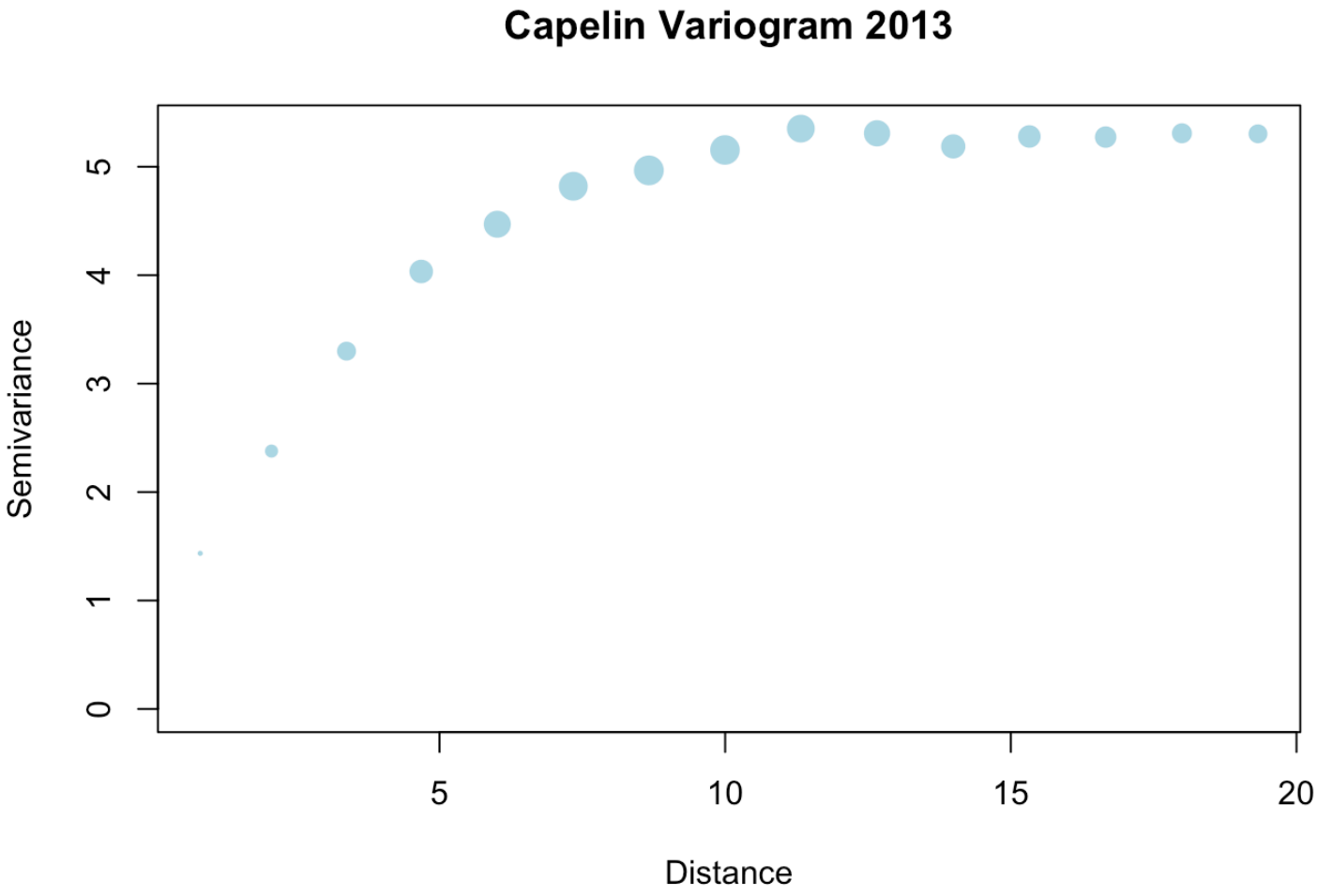
Ahhh, but these data are the same. I need to correct this and assign the colors to the data.

Well, we can continue with what to do and figure out the details later.

Empirical Variogram

Variogram of the log of capelin + 1–why?

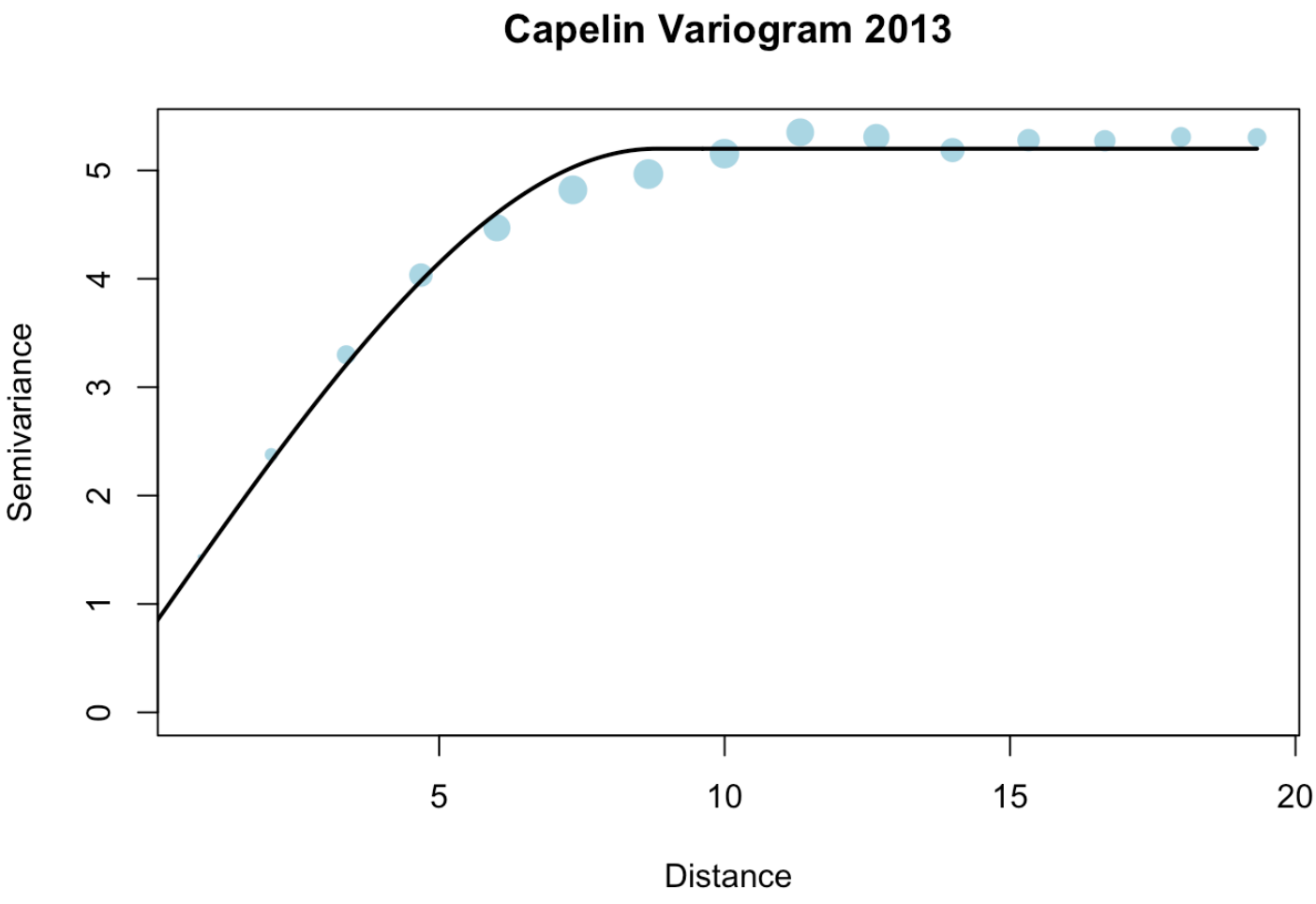
```
# Calculate the empirical variogram
capelin.vario = variogram(log(capelin+1)~1,datai,cutoff=20)
#
# Plot the empirical variogram
#
plot(gamma~dist,capelin.vario,
  ylim=c(0,max(gamma)),type='n',
  xlab="Distance",ylab="Semivariance",
  main=paste("Capelin Variogram",yy))
points(gamma~dist,capelin.vario,cex=2*np/max(np),pch=16,col="lightbl
ue")
```



Now we can identify the parameters and fit the variogram by eye. range = 4 nugget = 0.8 sill = 5

Fit the model first by eye

```
my.range = 8.8
my.nugget = 0.8
my.psill = 5.2-my.nugget
#
capelin.eye = vgm(model="Sph",psill=my.psill,range=my.range,nugget=m
y.nugget)
plot(gamma~dist,capelin.vario,
  ylim=c(0,max(gamma)),type='n',
  xlab="Distance",ylab="Semivariance",
  main=paste("Capelin Variogram",yy))
points(gamma~dist,capelin.vario,cex=2*np/max(np),pch=16,col="lightbl
ue")
vgmline = variogramLine(capelin.eye,max(capelin.vario$dist))
lines(gamma~dist,vgmline,lwd=2)
```



Fit the Actual Model

Now use these eye parameters to start the fit of model.

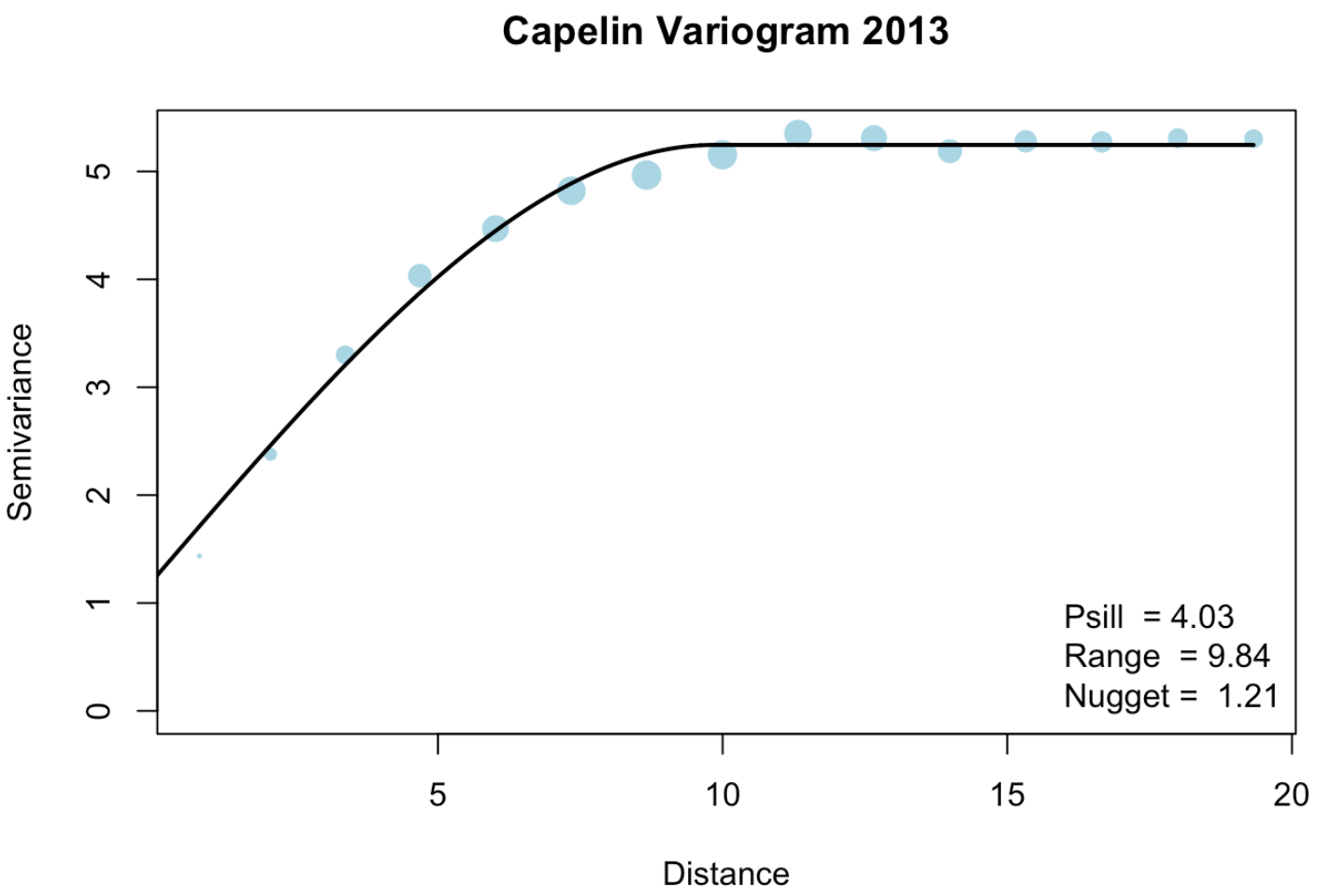
```
capelin.fit=fit.variogram(capelin.vario,
    vgm(model="Sph",psill=my.psill,range=my.range,nugget=my.nugget),
    fit.method=1)
#
# Look at estimates
capelin.fit
```

```
##   model    psill    range
## 1   Nug 1.213995 0.000000
## 2   Sph 4.031394 9.839629
```

```
capelin.psill=capelin.fit$psill[2]
capelin.range=capelin.fit$range[2]
capelin.nugget=capelin.fit$psill[1]
```

Plot the data, model and parameter estimates

```
plot(gamma~dist,capelin.vario,
  ylim=c(0,max(gamma)),type='n',
  xlab="Distance",ylab="Semivariance",
  main=paste("Capelin Variogram",yy))
points(gamma~dist,capelin.vario,cex=2*np/max(np),pch=16,col="lightbl
ue")
vgmline = variogramLine(capelin.fit,max(capelin.vario$dist))
lines(gamma~dist,vgmline,lwd=2)
#
legend("bottomright",legend = c(
        paste("Psill  =",round(capelin.psill,2)),
        paste("Range  =",round(capelin.range,2)),
        paste("Nugget = ",round(capelin.nugget,2))),
        bty="n")
```

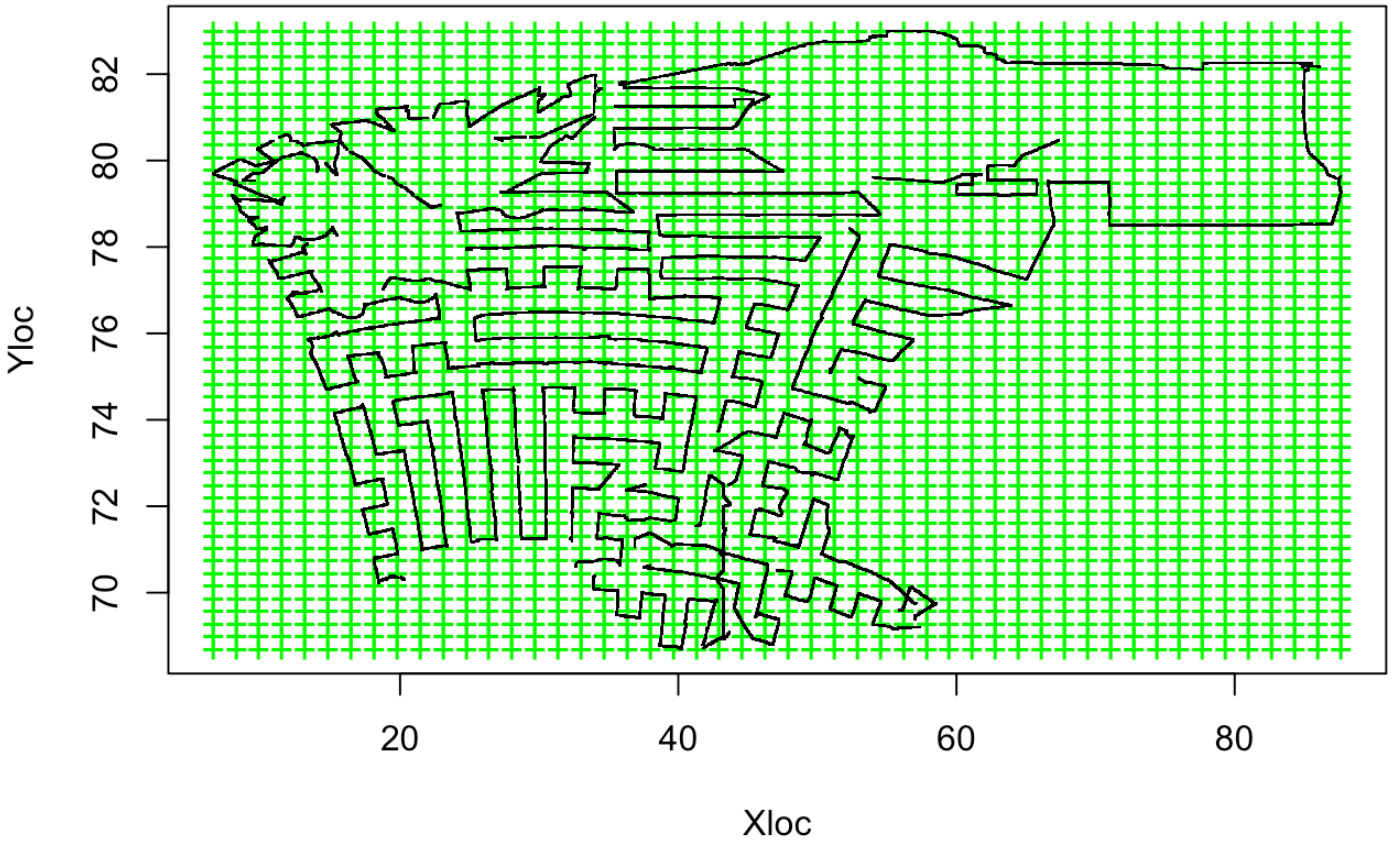


Predict

Let's make some predictions about the data, given the fitted model. So, we start with a grid to make predictions on.

```
# Create a grid of points to predict over
capelin.grid = expand.grid(
      Xloc=seq(min(datai$Xloc),max(datai$Xloc),length=50),
      Yloc=seq(min(datai$Yloc),max(datai$Yloc),length=50))
names(capelin.grid)=c("Xloc","Yloc")
coordinates(capelin.grid)=c("Xloc","Yloc")
capelin.grid = as(capelin.grid, "SpatialPixels")

# Now plot the data and overlay the prediction grid
plot(Yloc~Xloc,capelin.grid,cex=1.2,pch='+',col="green")
points(Yloc~Xloc,datai,pch=".")
```



The next thing is to get kriging working just to make sure I understand how it works, less about this being the correct approach for these data.