R Notebook

# Specializing Noise

A principle benefit of the drifting array is it’s ability to cover a considerable distance over the short deployment duration. In this portion of the report we ask whether it’s possible to create a noise map of the area.

The primary challenge in addressing this question is decoupling the space and time components as the units drift. Consider that the low frequency domain is dominated by atmospheric signals, shipping, and the calls of very loud animals including fin and blue whales. These sounds propagate tens to hundreds of miles resulting in highly correlated ambient noise metrics in time.

Lets first load the data and plot what we see.

rm(list = ls())  
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-41. For overview type 'help("mgcv-package")'.

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.2.3

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:nlme':  
##   
## collapse

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(viridis)

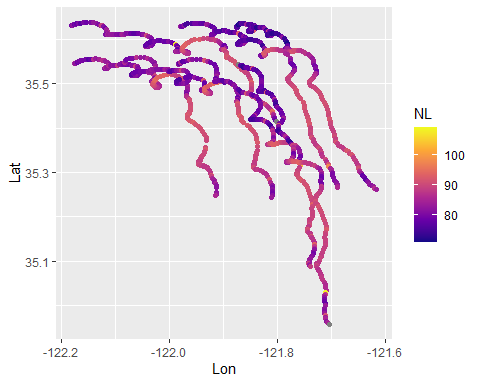
## Loading required package: viridisLite

## Warning: package 'viridisLite' was built under R version 4.2.3

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.2.3

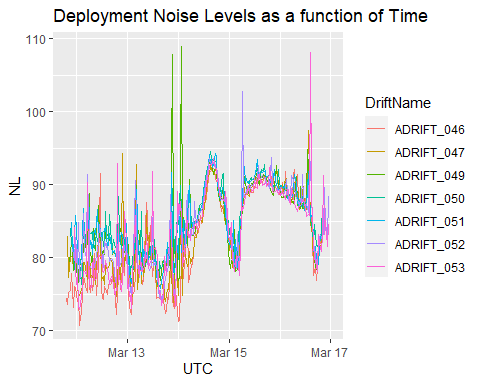
load('Adrift\_GPS\_2023.rds')  
load('Adrift\_NL\_2023\_500Hz.rds')  
  
# Link the gps and the noise levels  
noiseDf$Lon = NaN  
noiseDf$Lat = NaN  
  
DriftNames = unique(GPSdf$DriftName)  
GPSdf$NL<- NaN  
  
for(drift in DriftNames){  
 GPSsub = subset(GPSdf, DriftName == drift)  
 NLsub = subset(noiseDf, DriftName==drift)  
   
 UTMflon <- approxfun(GPSsub$UTC, GPSsub$Longitude)  
 UTMflat <- approxfun(GPSsub$UTC, GPSsub$Latitude)  
 NLf<-approxfun(NLsub$datetime\_posix, NLsub$TOL\_500, na.rm = TRUE)  
   
 ##############################################################  
 # Calculate the range from the whale to the GPS, TDOA and RL  
 ############################################################  
   
 # Lat/lon/ of the drift when the call was produced  
 noiseDf$Lon[noiseDf$DriftName==drift] =UTMflon(NLsub$datetime\_posix)  
 noiseDf$Lat[noiseDf$DriftName==drift] = UTMflat(NLsub$datetime\_posix)  
 GPSdf$NL[GPSdf$DriftName==drift] <-NLf(GPSsub$UTC)  
}  
  
colnames(noiseDf)[2]<-'NL'  
  
  
# Clear out any NA values  
noiseDf = noiseDf[!is.na(noiseDf$Lat),]  
  
# add seconds since start as an integer for the gam  
noiseDf$seconds = as.numeric(noiseDf$datetime\_posix-median(noiseDf$datetime\_posix))  
  
# there are some duplicated values for some reason.  
duplicatedIdx = which(duplicated(noiseDf$Lon))  
noiseDf$Lon[duplicatedIdx]=noiseDf$Lon[duplicatedIdx]+  
 rnorm(n = length(duplicatedIdx))/1000  
noiseDf$Lat[duplicatedIdx]=noiseDf$Lat[duplicatedIdx]+  
 rnorm(n = length(duplicatedIdx))/1000  
  
  
duplicatedIdx = which(duplicated(GPSdf$Lon))  
GPSdf$Lon[duplicatedIdx]=GPSdf$Lon[duplicatedIdx]+  
 rnorm(n = length(duplicatedIdx))/1000  
GPSdf$Lat[duplicatedIdx]=GPSdf$Lat[duplicatedIdx]+  
 rnorm(n = length(duplicatedIdx))/1000  
  
  
ggplot(GPSdf)+  
 geom\_point(aes(x=Lon, y=Lat, color=NL))+  
 scale\_color\_viridis\_c(option='plasma')



In this figure, it appears that as the units go south, the ambient noise level increases. This could lead us to think that the southern bit of the survey area is louder than the northern bit. However, recall that a large storm went through the area on March 14th. This could be influencing our background noise and as the buoys were fairly well behaved, spatial and temporal covariates could be confounded. To investigate this, lets plot the noise levels from all units as a function of time.

ggplot(GPSdf)+  
 geom\_line(aes(x= UTC, y=NL, color= DriftName))+  
 ggtitle('Deployment Noise Levels as a function of Time')

## Warning: Removed 4 rows containing missing values (`geom\_line()`).



This figure demonstrates the varying timescales acting on the ambient noise levels. Large-scale events such as storms impact the entire array (regardless of location) and smaller scale events effect locations differently. We cannot then assume that background noise was stationary in time, we must account for temporal variaotion in some way.

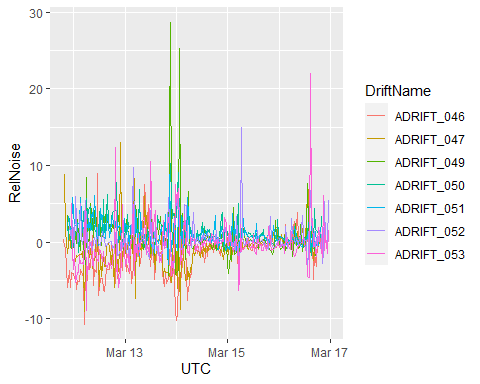
The simplest method of addressing the large scale events is to simply subtract the hourly median noise level from all the events. This will then leave us with relative noisiness. The below code calculates the median and plots the normalized noise levels.

# Calculate the median hourly noise level for each DriftName, day, and hour  
median\_hourly\_noise<- noiseDf %>%  
 group\_by(Date = date(datetime\_posix), Hour = hour(datetime\_posix)) %>%  
 summarise(medianNL = median(NL, na.rm = TRUE))

## `summarise()` has grouped output by 'Date'. You can override using the  
## `.groups` argument.

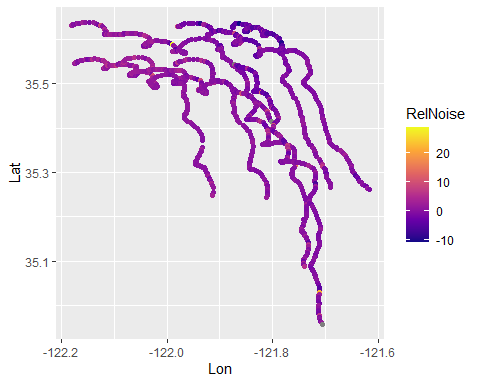
noiseDf$Date = date(noiseDf$datetime\_posix)  
noiseDf$Hour = hour(noiseDf$datetime\_posix)  
  
# Merge the median hourly noise back to the original data frame  
noiseDf <- noiseDf %>%  
 left\_join(median\_hourly\_noise, by = c( "Date", "Hour"))  
  
# Find differences in noise levels and replot  
noiseDf$RelNoise = noiseDf$NL- noiseDf$medianNL  
  
# Larger scale, crashes R, modelling will be done using the smaller grid  
#ggplot(noiseDf)+ geom\_line(aes(x= datetime\_posix, y=RelNoise, color= DriftName))  
   
# But also this dataframe is huge so lets look at the same using only the GPS locations which will making the modelling tennable  
GPSdf$Date = date(GPSdf$UTC)  
GPSdf$Hour = hour(GPSdf$UTC)  
GPSdf <- GPSdf %>%  
 left\_join(median\_hourly\_noise, by = c( "Date", "Hour"))  
  
GPSdf$RelNoise = GPSdf$NL- GPSdf$medianNL  
  
ggplot(GPSdf)+   
 geom\_line(aes(x= UTC, y=RelNoise, color= DriftName))

## Warning: Removed 4 rows containing missing values (`geom\_line()`).



That looks much more reasonable. Now we can see that there was actually considerable variation in noise level at the start of the study, when the units were more north, than later in the deployment. We can, and should, no plot this on our deployment map.

ggplot(GPSdf)+  
 geom\_point(aes(x=Lon, y=Lat, color=RelNoise))+  
 scale\_color\_viridis\_c(option='plasma')



In this figure we start seeing something more like what the GAM was showing, slightly elevated noise levels in the northwest corner and generally a lot more consistency. We would not expect large swings in median noise levels on the half hour scale. This is a better representation of the spatial element of noise in the 500Hz bin.

We can now use the fields package to fit both the raw data and the normalized noise levels to a surface.

library(fields)

## Warning: package 'fields' was built under R version 4.2.3

## Loading required package: spam

## Warning: package 'spam' was built under R version 4.2.3

## Spam version 2.9-1 (2022-08-07) is loaded.  
## Type 'help( Spam)' or 'demo( spam)' for a short introduction   
## and overview of this package.  
## Help for individual functions is also obtained by adding the  
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##   
## Attaching package: 'spam'

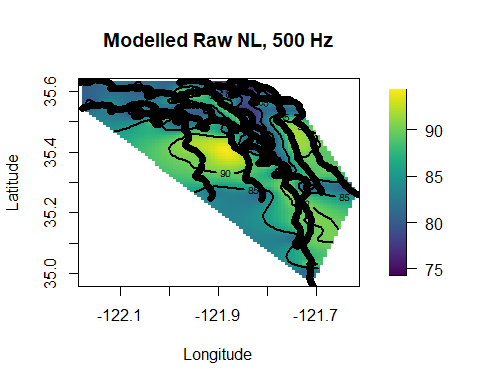
## The following objects are masked from 'package:base':  
##   
## backsolve, forwardsolve

##   
## Try help(fields) to get started.

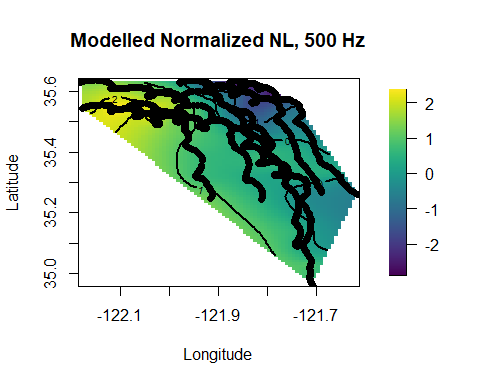
loc<-cbind(GPSdf$Lon,GPSdf$Lat) #locations  
obj<-spatialProcess(loc,GPSdf$NL, Distance = "rdist.earth",  
 cov.args = list(Covariance ="Exponential") )

## Warning in spatialProcess(loc, GPSdf$NL, Distance = "rdist.earth", cov.args = list(Covariance = "Exponential")): Numerical hessian from optim indicates  
## MLE is not a maximum

surface(obj,xlab = 'Longitude', ylab='Latitude',   
 main = 'Modelled Raw NL, 500 Hz')  
points(x = GPSdf$Lon, y= GPSdf$Lat, col = "black", pch = 16)



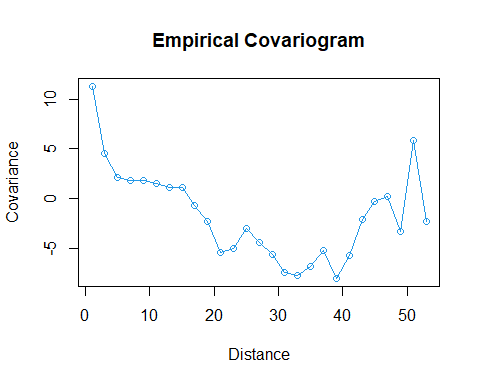
obj.norm<-spatialProcess(loc,  
 GPSdf$NL-GPSdf$medianNL,   
 Distance = "rdist.earth",  
 cov.args = list(Covariance ="Exponential"),  
 REML = TRUE,  
 cov.function = 'stationary.cov')  
surface(obj.norm,  
 xlab = 'Longitude', ylab='Latitude',   
 main = 'Modelled Normalized NL, 500 Hz')  
points(x = GPSdf$Lon, y= GPSdf$Lat, col = "black", pch = 16)



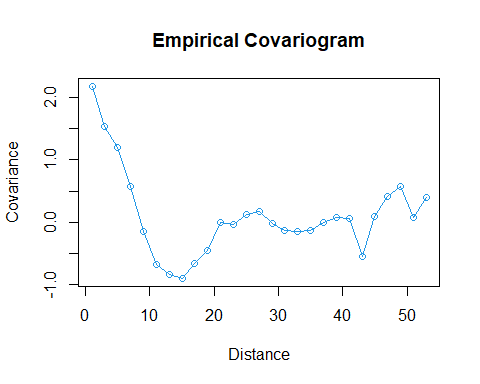
We can evaluate spatial and temporal autocorrelation using a ‘[Variogram](https://vsp.pnnl.gov/help/vsample/Kriging_Variogram.htm#:~:text=Variogram%20Parameters&text=The%20default%20and%20recommended%20first,for%20log%2Dnormally%20distributed%20data.)’ in the same package.

“A variogram is a description of the spatial continuity of the data. The experimental variogram is a discrete function calculated using a measure of variability between pairs of points at various distances.”

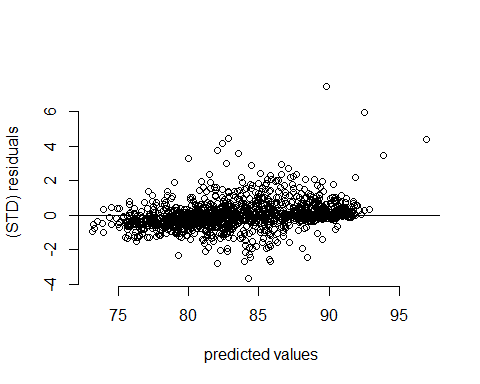
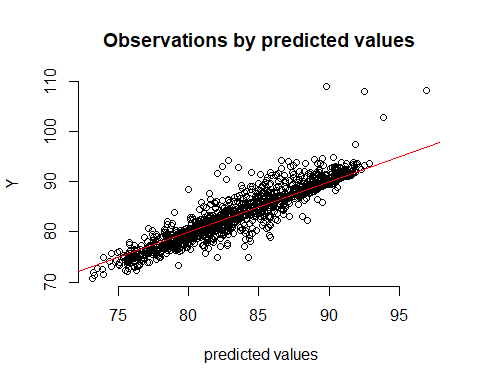
look1<-vgram( loc, GPSdf$NL, N=15, lon.lat=TRUE, type="covariogram")  
look2<-vgram( loc, GPSdf$medianNL-GPSdf$NL,   
 N=15, lon.lat=TRUE, type="covariogram")  
  
brk<- seq( 0, 600,(1 + 1) ) # will give 25 bins.  
plot(look1, breaks=brk, col=4)



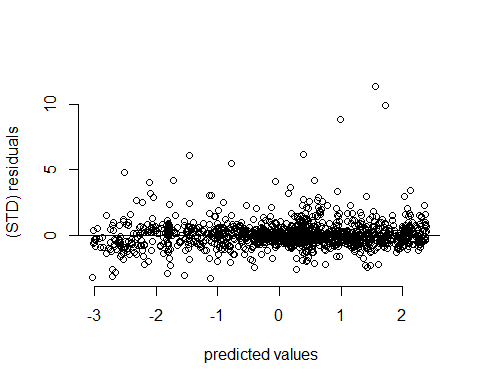
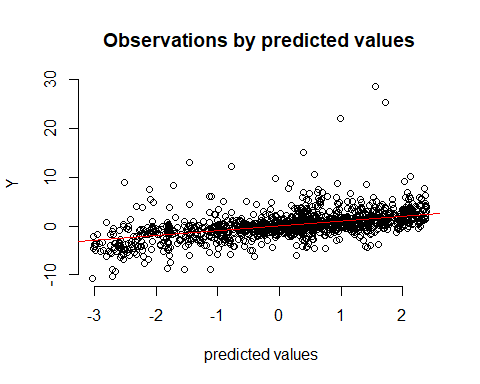
plot(look2, breaks=brk, col=4)



# We can also plot the residuals  
plot(obj)



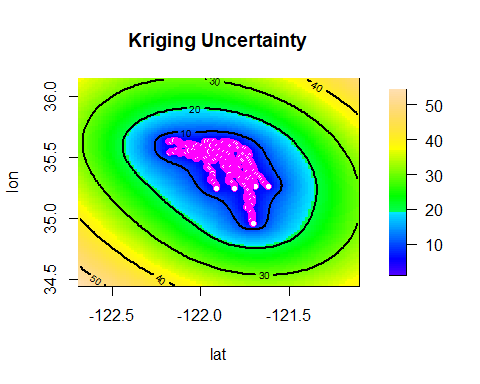
plot(obj.norm)



By normalizing the noise levels, we’ve brought down the covariance, but it’s somewhat unclear the units of the Y axis. This will need further digging. We can also look at the residuals to help define our fit. Here we can again see that normalizng the noise levels seems to bring down the variance considerably; there seems to be a relationship in the raw data between the noise level and the standard deviation of the residuals-which isn’t good if memory recalls. This is also worth some discussion.

Last we can use the fields package to look at model uncertainty.

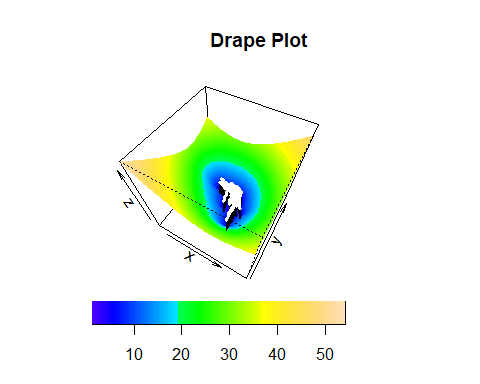
grid.list <- list(lat=seq(min(GPSdf$Lon)-.5,max(GPSdf$Lon)+.5, length.out =100),   
 lon=seq(min(GPSdf$Lat)-.5,max(GPSdf$Lat)+.5, length.out =100))  
full.grid <- make.surface.grid(grid.list)  
  
SEobs <- predictSE.mKrig(obj)  
SEout <- predictSE(obj, xnew=full.grid)  
  
lookSE <- as.surface(full.grid, SEout)  
{surface(lookSE,col=topo.colors(100))  
title("Kriging Uncertainty")  
points(loc[,1],loc[,2],col="magenta",bg="white",pch=21)}



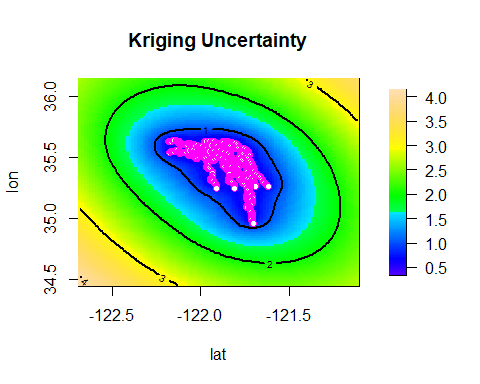
{drape.plot(lookSE, border=NA, aRange=160, phi=55,col=topo.colors(100)) -> dp2  
title("Drape Plot")  
pushpin(loc[,1],loc[,2],SEobs,dp2, cex=0.4, col="white")}

## Warning in persp.default(x, y, z, theta = theta, phi = phi, col =  
## drape.info$color.index, : "aRange" is not a graphical parameter

## Warning in cbind(x, y, z, 1, deparse.level = 0L): number of rows of result is  
## not a multiple of vector length (arg 3)

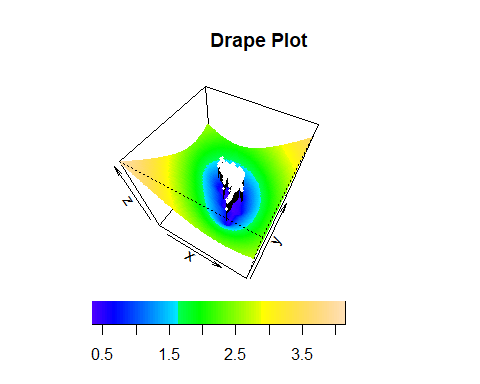


SEobs.norm <- predictSE.mKrig(obj.norm)  
SEout.norm <- predictSE(obj.norm, xnew=full.grid)  
  
lookSE.norm <- as.surface(full.grid, SEout.norm)  
{surface(lookSE.norm,col=topo.colors(100))  
title("Kriging Uncertainty")  
points(loc[,1],loc[,2],col="magenta",bg="white",pch=21)}



{drape.plot(lookSE.norm, border=NA, phi=55,col=topo.colors(100)) -> dp2  
title("Drape Plot")  
pushpin(loc[,1],loc[,2],SEobs,dp2, cex=0.4, col="white")}

## Warning in cbind(x, y, z, 1, deparse.level = 0L): number of rows of result is  
## not a multiple of vector length (arg 3)



This has been a relatively quick look at how we can specialize our noise metrics. There is much more than can and should be done. However, for the purpose of the report I think there is maybe one or two more things worth investigating including estimating the predictive power of the method.

Predictive power

1. Knock out one or more of the sensors
2. Fit the field models
3. Used the field models to predict the levels at the knocked-out location
4. Estimate the error
5. Do this for one or more knockouts