Soundscape analysis for 'Sensitive aerial hearing within a noisy nesting soundscape in a deep-diving seabird, the common murre *Uria aalge*'

2022-11-15

Overview

This analysis follows the following steps:

1. Import the data
2. Smooth the noisy soundscape data
3. Difference the data to make it stationary
4. Simple correlation
5. Cross correlation (correlation at all lags)

The following packages are required:

```
library(tidyverse)
library(readxl)
library(janitor)
library(zoo)
library(tseries)
```

S1 Import the data

The primary data used here are the sound level data, which is provided with a sampling interval of 5 minutes. This data is joined by environmental data on light levels, and wind speed which are both provided with a 1 hour sampling interval and swell (wave height) which is provided with a 10 minute interval. This difference between sampling intervals necessitates interpolation of the low-frequency data to ensure that the time series are harmonized with the sound level data.

Sound levels

```
soundLevels <- read_xlsx("data/data_soundlevels_forOwen.xlsx", sheet = 2, skip = 1) %>%
  mutate(time_value = lubridate::parse_date_time(gsub(
    pattern = "'", replacement = "", Timestamp
  ), "d-b-Y H:M:S")) %>%
  select(-Timestamp) %>%
  select(time_value, everything()) %>%
  clean_names()
```

Wave height (swell)

```
waveHeight <- read_xlsx("data/data_offshorewaveheight_forOwen.xlsx") %>%
  mutate(time_value = lubridate::parse_date_time(gsub(
```

```
    pattern = "'", replacement = "", Timestamps
  ), "d-b-Y H:M:S")) %>%
  select(time_value, waveHeight = "Offshore swell height") %>%
  mutate(waveHeight = as.numeric(waveHeight))
```

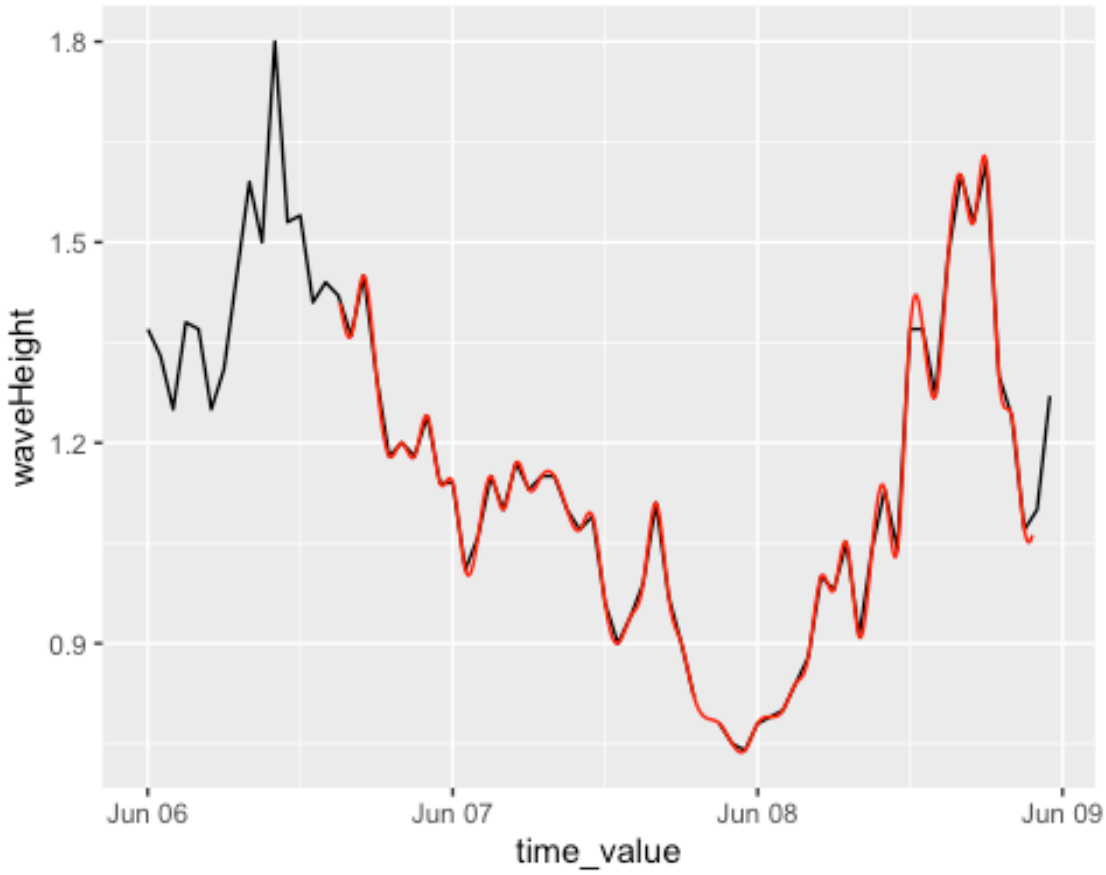
After importing, the data are interpolated using cubic splines in order to predict points on the same time interval as the sound data. To check visually the efficacy of the interpretation the raw data are then plotted alongside the interpolated values.

Interpolation:

```
SplineFun <- splinefun(x = waveHeight$time_value, y = waveHeight$waveHeight)
SplineFit <- SplineFun(soundLevels$time_value)
SplineFit_df <- data.frame(time_value = soundLevels$time_value, waveHeight =
SplineFit)
```

Plotting:

```
ggplot(waveHeight, aes(x = time_value, y = waveHeight)) +
  geom_line() +
  geom_line(data = SplineFit_df, inherit.aes = TRUE, colour = "red")
```



The interpolated value for swell is now added to the `soundLevels` data frame.

```
soundLevels <- left_join(soundLevels, SplineFit_df)

## Joining, by = "time_value"
```

Sealevels

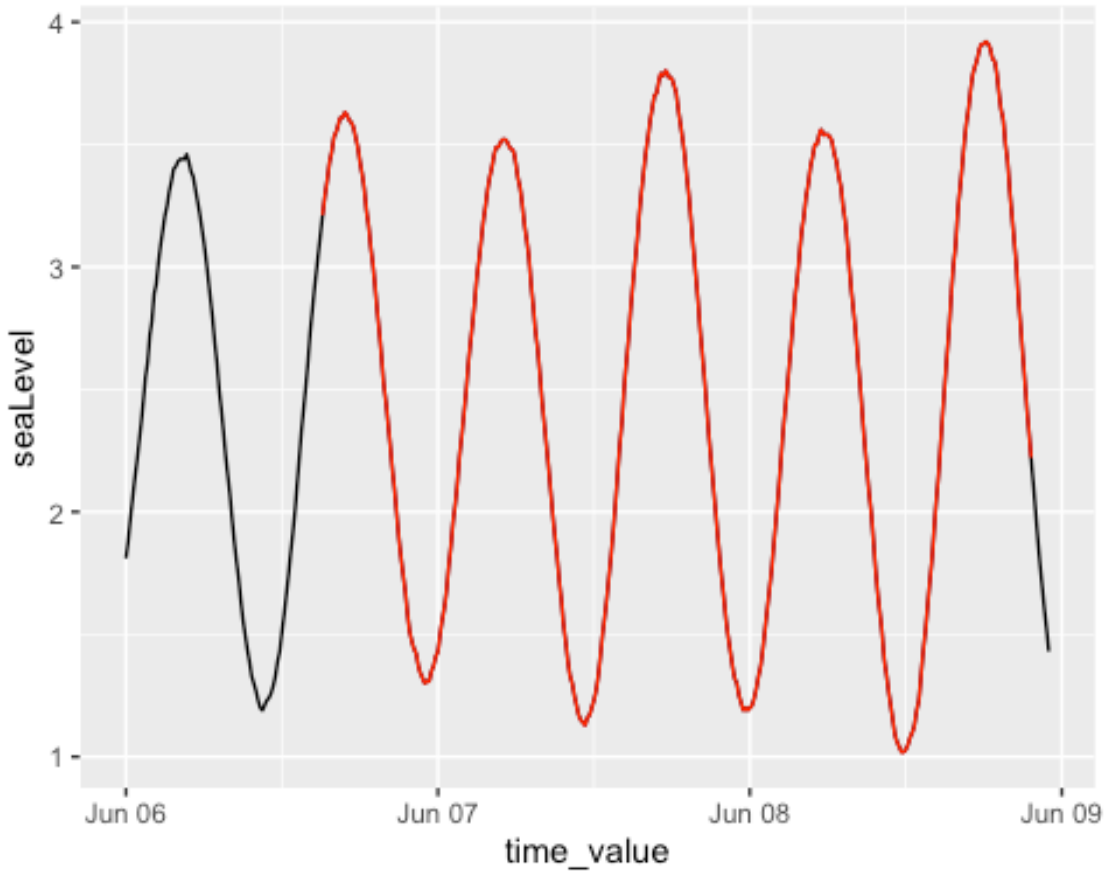
```
seaLevels <- read_xlsx("data/data_sealevels_forOwen.xlsx", trim_ws = TRUE, sk
ip = 1) %>%
  mutate(time_value = lubridate::parse_date_time(gsub(
    pattern = "'", replacement = "", Timestamps
  ), "d-b-Y H:M:S")) %>%
  select(time_value, seaLevel = SeaLevel)
```

Interpolation:

```
SplineFun <- splinefun(x = seaLevels$time_value, y = seaLevels$seaLevel)
SplineFit <- SplineFun(soundLevels$time_value)
SplineFit_df <- data.frame(time_value = soundLevels$time_value, seaLevel = Sp
lineFit)
```

Fit check:

```
ggplot(seaLevels, aes(x = time_value, y = seaLevel)) +
  geom_line() +
  geom_line(data = SplineFit_df, inherit.aes = TRUE, colour = "red")
```



Add to the soundLevels data frame:

```
soundLevels <- left_join(soundLevels, SplineFit_df)

## Joining, by = "time_value"
```

Windspeed

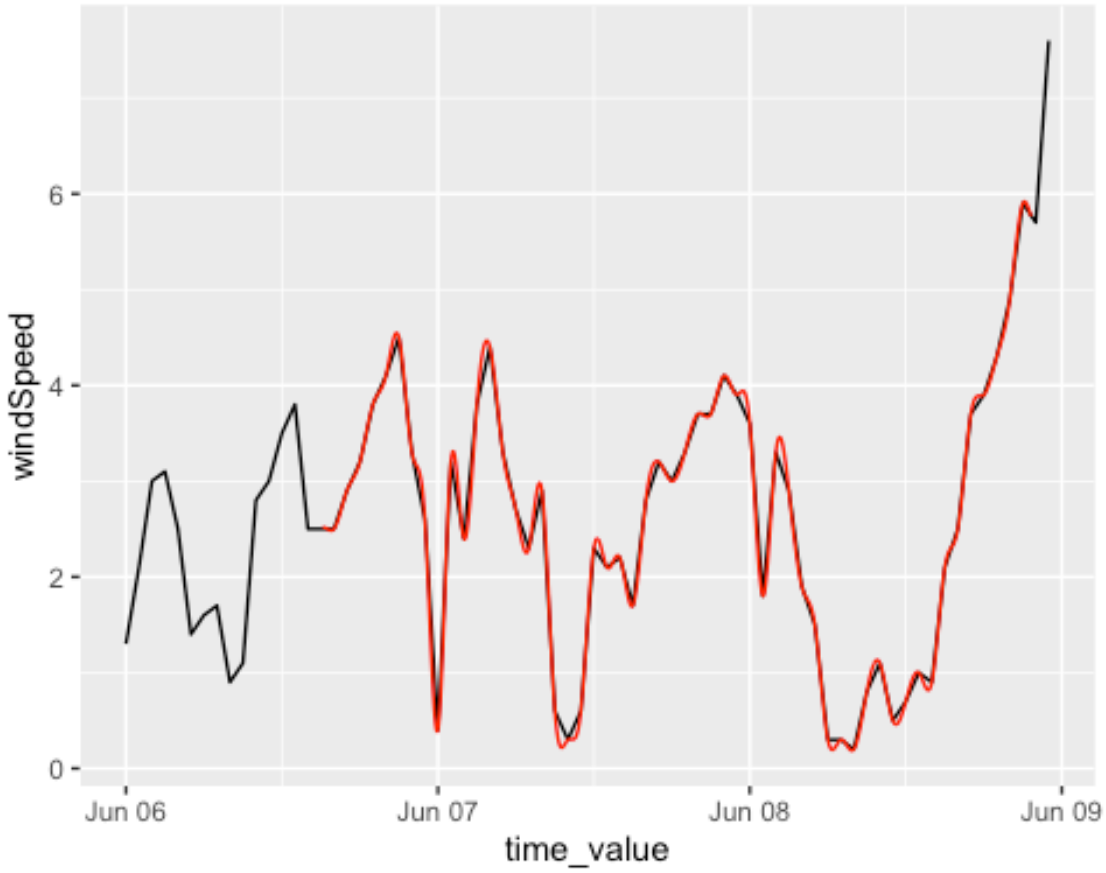
```
windSpeed <- read_xlsx("data/data_windspeed_forOwen.xlsx") %>%
  mutate(time_value = lubridate::parse_date_time(gsub(
    pattern = "'", replacement = "", Timestamps
  ), "d-b-Y H:M:S")) %>%
  select(-Timestamps) %>%
  select(time_value, windSpeed = "wind speed")
```

Interpolate:

```
SplineFun <- splinefun(x = windSpeed$time_value, y = windSpeed$windSpeed)
SplineFit <- SplineFun(soundLevels$time_value)
SplineFit_df <- data.frame(time_value = soundLevels$time_value, windSpeed = S
plineFit)
```

Check the fit:

```
ggplot(windSpeed, aes(x = time_value, y = windSpeed)) +
  geom_line() +
  geom_line(data = SplineFit_df, inherit.aes = TRUE, colour = "red")
```



Add the interpolated values to the soundLevels data frame:

```
soundLevels <- left_join(soundLevels, SplineFit_df)

## Joining, by = "time_value"
```

Light Levels

There are several measures available in this data set. However, the focus of this analysis will be the total average amount of sunlight which is labeled "RADGL" in the original data file.

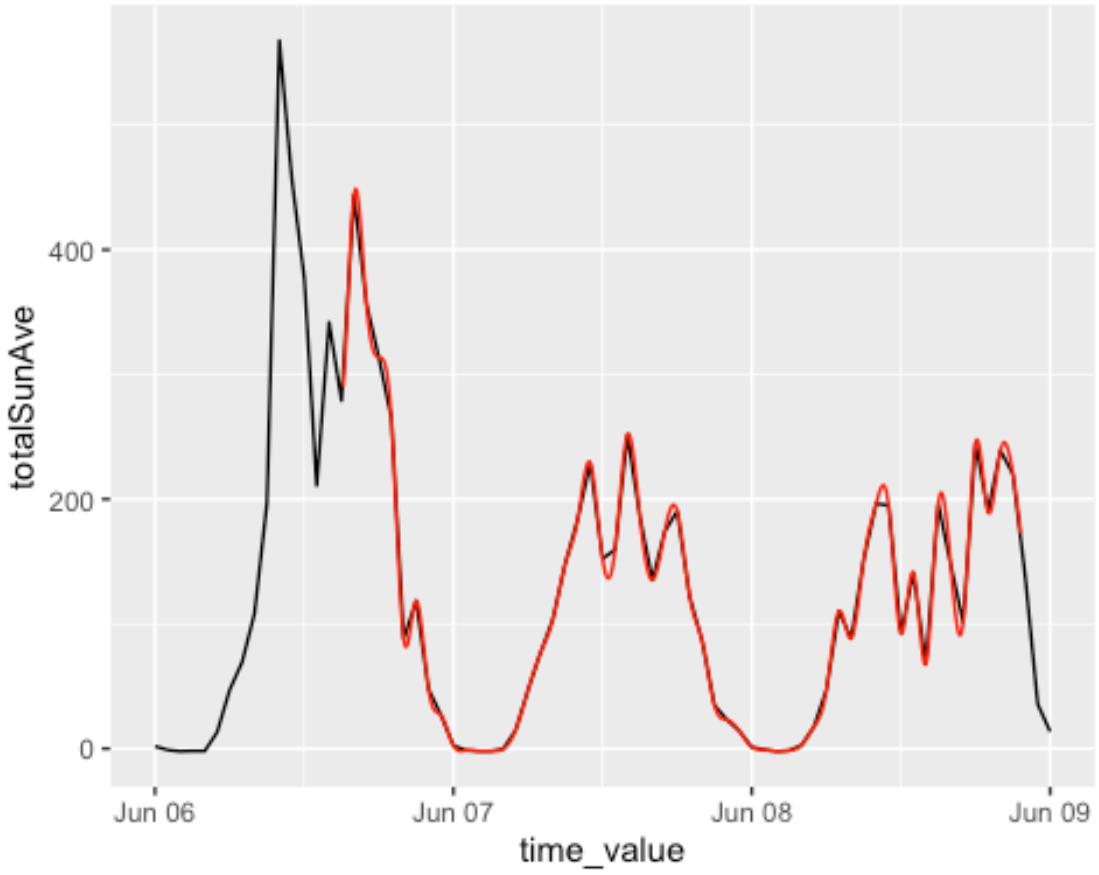
```
lightLevels <- read_xlsx("data/light_Afgreidsla_nr_20220824_67588_svl.xlsx",
sheet = 3) %>%
  select(
    time_value = TIMI, totalSunAve = RADGL, totalSunMax = RADGLX, sunRadRefle
ctAve = RADSWS,
    sunRadReflectMax = RADSWSX, IRfromSkyAve = RADLWI, IRfromSkyMax = RADLWIX
,
    IRfromGroundAve = RADLWS, IRfromGroundMax = RADLWSX
  ) %>%
  select(time_value, totalSunAve)
```

Interpolate:

```
SplineFun <- splinefun(x = lightLevels$time_value, y = lightLevels$totalSunAv
e)
SplineFit <- SplineFun(soundLevels$time_value)
SplineFit_df <- data.frame(time_value = soundLevels$time_value, totalSunAve =
SplineFit)
```

Fit check:

```
ggplot(lightLevels, aes(x = time_value, y = totalSunAve)) +
  geom_line() +
  geom_line(data = SplineFit_df, inherit.aes = TRUE, colour = "red")
```



Add the interpolated value to the `soundLevels` data frame:

```
soundLevels <- left_join(soundLevels, SplineFit_df)

## Joining, by = "time_value"
```

The code above produced several objects which are unnecessary for the subsequent analysis. These unnecessary objects are now removed and the `soundLevels` dataset is retained for analysis. At this point, the dataset `soundLevels` contains all the sound level data alongside the other environmental variables.

```
rm(list = ls()[which(!ls() == "soundLevels")])
```

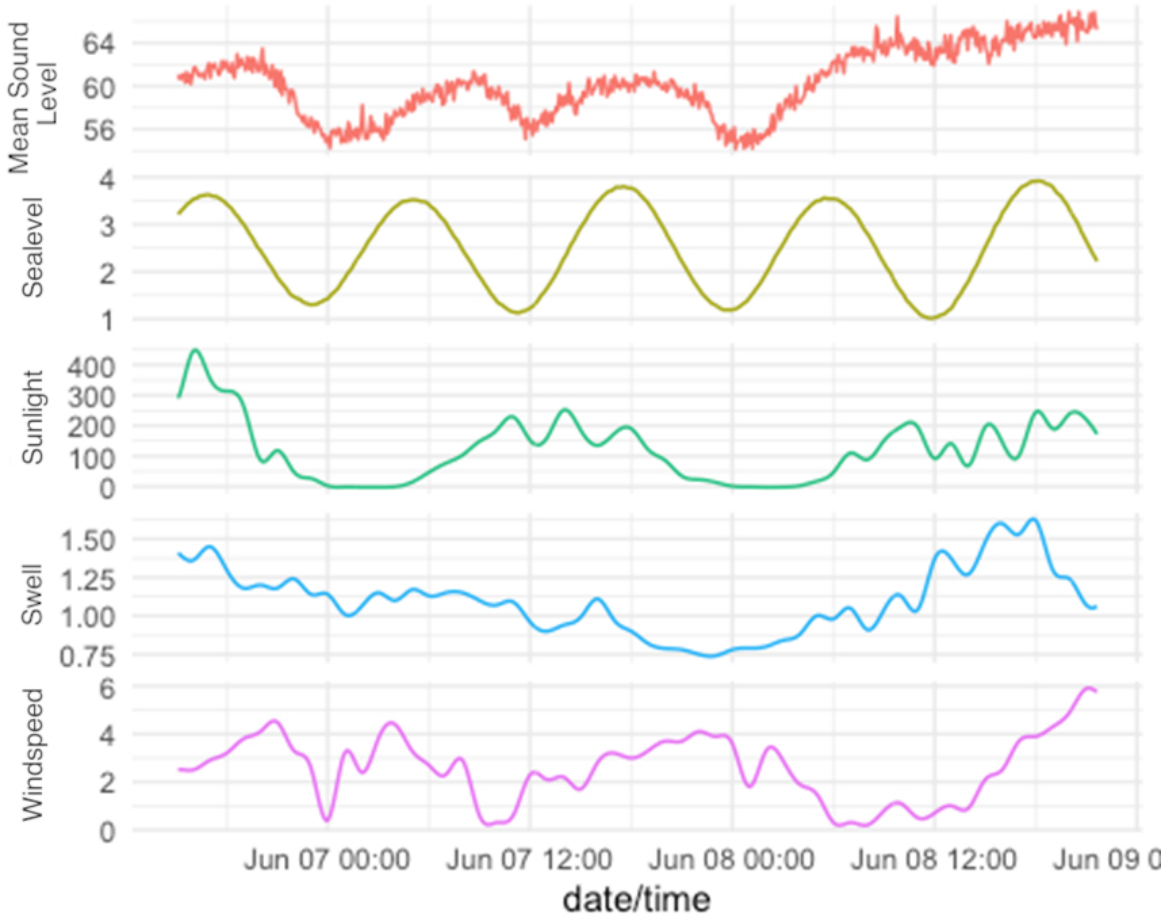
To avoid the potential for confusion, only the columns necessary for the subsequent analyses are retained using `select`.

```
soundLevels <- soundLevels %>%
  select(time_value, mean_a_weighted, seaLevel, windSpeed, waveHeight, totalS
unAve)
```

S2 Plotting the data

The sound level data are now plotted alongside the environmental variables.

```
ggplot(
  soundLevels %>%
    pivot_longer(
      cols = -time_value,
      names_to = "measurement",
      values_to = "value"
    ),
  aes(x = time_value, y = value, colour = measurement)
) +
  geom_line() +
  facet_wrap(
    measurement ~ .,
    nrow = 5,
    scales = "free_y",
    strip.position = "left",
    labeller = as_labeller(
      c(
        mean_a_weighted = "Mean Sound Level",
        seaLevel = "Sea level (m)",
        windSpeed = "Wind speed (m/s)",
        waveHeight = "Swell (m)",
        totalSunAve = "Sunlight"
      )
    )
  ) +
  theme_minimal() +
  theme(legend.position = "none", strip.placement = "outside") +
  ylab(NULL) +
  xlab("date/time")
```



Smoothing the sound level data.

The sound level data are very "noisy". This noise will tend to mess up the differencing method because the broader pattern of low frequency increases/decreases will be obscured by high-frequency noise.

The smoothing function to do this, `zoo::rollmean`, creates a rolling mean of values in the time series, with a window of size `k`. The function requires the data to be in a specific format (`zoo`), so it is first necessary to create a new vector `meanA`, which is a copy of the `mean_a_weighted` data, then convert this new vector into a `zoo` object using `zoo::zoo`.

```
soundLevels <- soundLevels %>%
  mutate(meanA = zoo::zoo(mean_a_weighted, time_value))
```

Then a smoothed version of the data is created using the `zoo::rollmean` function, with a `k` value of 13 Since the interval for the data is 5 minutes, this represents a window of 65 minutes. Note that the `rep(NA...)...` parts of the code below is to place `NA` values before and after the rolling mean values and ensure that the value provided represents the middle of the observation window. Note that if this is not done, the values will be offset from the other (environmental) variables.

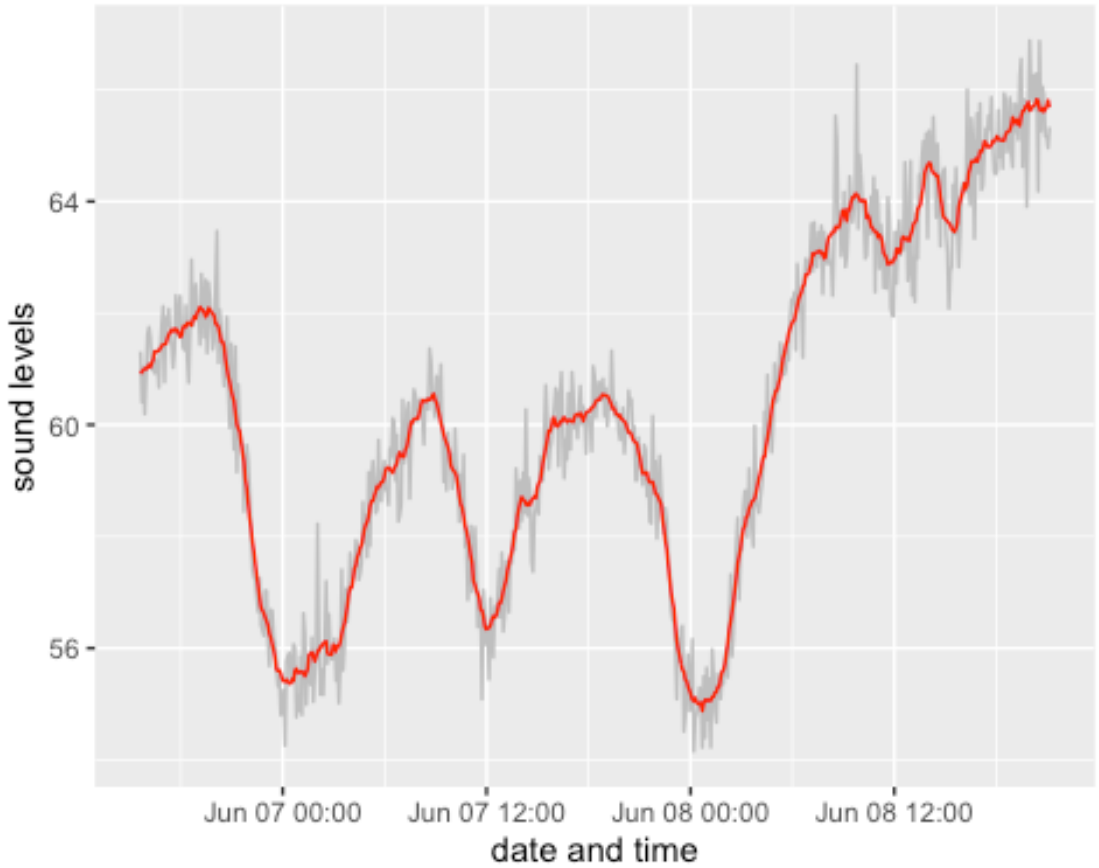
```
k <- 13
soundLevels$meanA_smoothed <- c(rep(NA, (k - 1) / 2),
                          as.numeric(zoo::rollmean(soundLevels$meanA, k = k)),
                          rep(NA, (k - 1) / 2))
```

After doing this, there are some NA values at the beginning and end of the data frame and these are removed using na.omit before proceeding. It is also necessary to ensure that the new column is of a simple numeric data type (rather than a zoo time series object). The copy of meanA is no longer needed and is also removed.

```
soundLevels <- soundLevels %>%
  na.omit() %>%
  mutate(meanA_smoothed = as.numeric(meanA_smoothed)) %>%
  select(-meanA)
```

Before proceeding the efficacy of the smoothing can be assessed visually:

```
ggplot(soundLevels, aes(x = time_value, meanA_smoothed)) +
  geom_line(aes(x = time_value, y = mean_a_weighted), colour = "grey75") +
  geom_line(colour = "red") +
  ylab("sound levels")+
  xlab("date and time")
```



S3 Differencing

Dickey-Fuller tests for stationarity are carried out on the time series to determine which ones are non-stationary and should detrended using differencing.

```
adf.test(soundLevels$meanA_smoothed)

##
##  Augmented Dickey-Fuller Test
##
## data:  soundLevels$meanA_smoothed
## Dickey-Fuller = -3.4047, Lag order = 8, p-value = 0.05264
## alternative hypothesis: stationary

adf.test(soundLevels$waveHeight)# p>0.05

##
##  Augmented Dickey-Fuller Test
##
## data:  soundLevels$waveHeight
## Dickey-Fuller = -2.3388, Lag order = 8, p-value = 0.4349
## alternative hypothesis: stationary

adf.test(soundLevels$windSpeed)# p>0.05

##
##  Augmented Dickey-Fuller Test
##
## data:  soundLevels$windSpeed
## Dickey-Fuller = -2.81, Lag order = 8, p-value = 0.2354
## alternative hypothesis: stationary

adf.test(soundLevels$seaLevel)

## Warning in adf.test(soundLevels$seaLevel): p-value smaller than printed p-
value

##
##  Augmented Dickey-Fuller Test
##
## data:  soundLevels$seaLevel
## Dickey-Fuller = -15.252, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary

adf.test(soundLevels$totalSunAve) # p>0.05

##
##  Augmented Dickey-Fuller Test
##
## data:  soundLevels$totalSunAve
## Dickey-Fuller = -3.4823, Lag order = 8, p-value = 0.04401
## alternative hypothesis: stationary
```

Three of the time series are non-stationary. These are detrended as follows:

```
soundLevels <- soundLevels %>%
  mutate(waveHeight_diff = waveHeight - lag(waveHeight)) %>%
  mutate(windSpeed_diff = windSpeed - lag(windSpeed)) %>%
  mutate(seaLevel_diff = seaLevel - lag(seaLevel)) %>%
  na.omit()
```

S4 Simple correlation

After differencing, the simple correlation among the variables can be assessed more appropriately.

```
cor(soundLevels %>%
  select(meanA_smoothed, seaLevel, waveHeight_diff, windSpeed_diff, totalSunA
ve))

##                 meanA_smoothed    seaLevel waveHeight_diff windSpeed_diff
## meanA_smoothed      1.00000000  0.36484247     -0.04749991     0.15378172
## seaLevel            0.36484247  1.00000000     -0.15541150     0.09416371
## waveHeight_diff    -0.04749991 -0.15541150      1.00000000    -0.07291020
## windSpeed_diff      0.15378172  0.09416371     -0.07291020     1.00000000
## totalSunAve         0.53277750  0.25699947     -0.15421668     0.18879352
##                 totalSunAve
## meanA_smoothed    0.5327775
## seaLevel          0.2569995
## waveHeight_diff  -0.1542167
## windSpeed_diff    0.1887935
## totalSunAve       1.0000000
```

These are pairwise correlations and shows that the correlation with noise levels is greatest for `totalSunAve`, then `seaLevel`, then `windspeed`. These three correlations are statistically significant. The correlation with `waveHeight` is not statistically significant.

```
cor.test(soundLevels$meanA_smoothed, soundLevels$totalSunAve)

##
##  Pearson's product-moment correlation
##
## data:  soundLevels$meanA_smoothed and soundLevels$totalSunAve
## t = 15.915, df = 639, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4749321 0.5860388
## sample estimates:
##       cor
## 0.5327775

cor.test(soundLevels$meanA_smoothed, soundLevels$seaLevel)
```

```
##
##  Pearson's product-moment correlation
##
## data:  soundLevels$meanA_smoothed and soundLevels$seaLevel
## t = 9.9054, df = 639, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2957583 0.4301301
## sample estimates:
##       cor
## 0.3648425
```

```
cor.test(soundLevels$meanA_smoothed, soundLevels$windSpeed_diff)
```

```
##
##  Pearson's product-moment correlation
##
## data:  soundLevels$meanA_smoothed and soundLevels$windSpeed_diff
## t = 3.9342, df = 639, p-value = 9.263e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.07726147 0.22850088
## sample estimates:
##       cor
## 0.1537817
```

```
cor.test(soundLevels$meanA_smoothed, soundLevels$waveHeight_diff)
```

```
##
##  Pearson's product-moment correlation
##
## data:  soundLevels$meanA_smoothed and soundLevels$waveHeight_diff
## t = -1.2021, df = 639, p-value = 0.2298
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.12448236  0.03005099
## sample estimates:
##         cor
## -0.04749991
```

S5 Cross correlations

The correlations above are straightforward correlations where the sound levels at time *t* were correlated with the environmental variables at time *t*. However, it is likely that there are lags in these effects. Therefore it is sensible to examine the correlations at other time lags. Cross-correlation calculates the correlation coefficients at a range of lags and can be informative about the system under study.

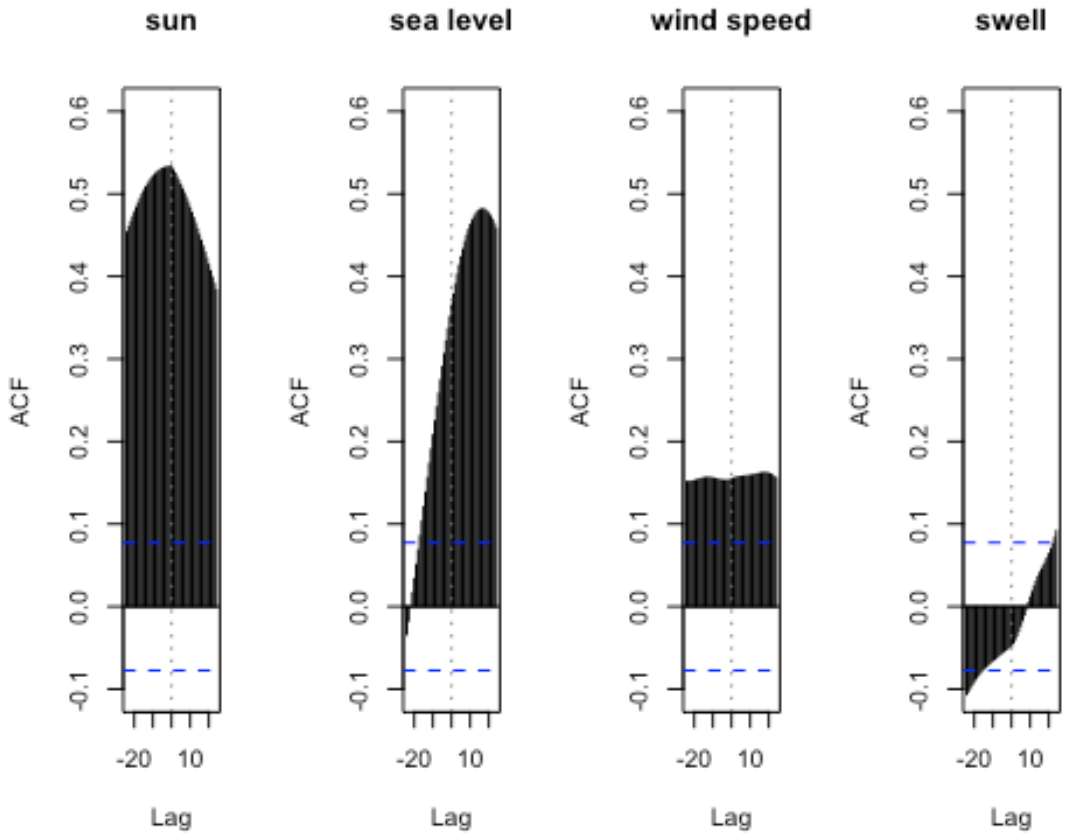
```
par(mfrow = c(1, 4))
maxLag <- 120/5
ccf(
```

```
  soundLevels$meanA_smoothed,
  soundLevels$totalSunAve,
  lag.max = maxLag,
  ylim = c(-0.1, 0.6),
  main = "sun"
)
abline(v = 0, col = "grey50", lty = "dotted")

ccf(
  soundLevels$meanA_smoothed,
  soundLevels$seaLevel,
  lag.max = maxLag,
  ylim = c(-0.1, 0.6),
  main = "sea level"
)
abline(v = 0, col = "grey50", lty = "dotted")

ccf(
  soundLevels$meanA_smoothed,
  soundLevels$windSpeed_diff,
  lag.max = maxLag,
  ylim = c(-0.1, 0.6),
  main = "wind speed"
)
abline(v = 0, col = "grey50", lty = "dotted")

ccf(
  soundLevels$meanA_smoothed,
  soundLevels$waveHeight_diff,
  lag.max = maxLag,
  ylim = c(-0.1, 0.6),
  main = "swell"
)
abline(v = 0, col = "grey50", lty = "dotted")
```



S6 Summary

- The sound level is positively and significantly associated, at zero lag, with sunlight, sea level and wind speed (in order of strength of correlation).
- There is only a weak association with swell (wave height), and this is non-significant at zero lag.
- The association with sunlight is strongest at zero lag, implying that there is a simple relationship: when it is sunny the birds are more actively.
- The lag with sea level is such that the correlation is strongest with a 12 unit lag (i.e. 12 * 5 = 1 hour).
- The positive association with wind speed could well be spurious (i.e. higher wind speeds occur when it is sunny, and the birds are more active when it is sunny? or similar)
- These associations are hard to tease apart because the time frame is fairly short in relation to the range of conditions. For example, to tease apart the effect of wind speed from other factors, you would need recordings that include incidences of high and low wind speeds at high and low sunlight, high and low swell, and high and low tide, several times.