Tutorial 14: Spectral Analysis (II)

The data for this tutorial is found in the course directory under the file `nino3data.asc`. The data points are the monthly average sea surface temperatures over the East Equatorial Pacific.

Firstly, prepare the data; read it into a table and name the three columns 'Year', 'SST' and 'SSA'. SST denotes the average over the month, while A means 'anomaly' and gives the difference from the monthly average.

You'll need the package **ggplot2** and **gridExtra**; other packages are listed lower down. Install them when necessary.

```
> install.packages("ggplot2")
> install.packages("gridExtra")
```

Activate these packages.

For the data:

```
> www<-"https://www.mimuw.edu.pl/~noble/courses/TimeSeries/Data/nino3data.asc"
> nino<-read.table(www,skip=3,header=F)
> names(nino) <- c("Year", "SST", "SSA")
> plot(nino$Year, nino$SST, type = "l")
```

To plot the data:

```
> plot1 <- ggplot(data = nino) + geom_line(aes(y = SST, x = Year))
> plot2 <- ggplot(data = nino) + geom_line(aes(y = SSA, x = Year))
> grid.arrange(plot1, plot2)
```

This should give you some idea of how the 'averages' and 'anomalies' are evolving in time.

You can get autocorrelation plots as follows:

```
> acf1 <- acf(nino$SST, lag.max = 12 * 20, plot = FALSE)
> acf2 <- acf(nino$SSA, lag.max = 12 * 20, plot = FALSE)
> plot1 <- ggplot() + geom_line(aes(x = acf1$lag/12, y = acf1$acf))
> plot2 <- ggplot() + geom_line(aes(x = acf2$lag/12, y = acf2$acf))
> grid.arrange(plot1, plot2)
```

These would suggest that some frequencies are important here, so that a study in the frequency domain may prove fruitful with this data set.

Harmonic Regression The purpose of spectral analysis is to decompose a time series into periodic components. Linear regression is a useful tool for this, where the time series is regressed on a set of sine and cosine waves. For a dataset with annual variation, we might expect that the sine and cosine waves with one year might be important, but there may be others.

The El Nino/La Nina cycle is around 3-6 years; the following analysis regresses the data against suitable cycles.

First, we create a data frame with sines and cosines.

```
# Create dataframe with different harmonics
X <- data.frame(Year=nino$Year,
       y = nino$SST,
       sin(2*pi*1*nino$Year), cos(2*pi*1* nino$Year), # sine and cos for
frequency = 1
       sin(2*pi*2*nino$Year), cos(2*pi*2*nino$Year), # freq. equals 2 (i.e.
period= 6 months)
       sin(2*pi*1/3*nino$Year), cos(2*pi*1/3*nino$Year), # freq = 1/3 (period=3
years)
       sin(2*pi*1/3.5*nino$Year), cos(2*pi*1/3.5*nino$Year), # freq=3.5
(period=3.5 years)
       sin(2*pi*1/6*nino$Year), cos(2*pi*1/6*nino$Year),   # freq=6 (period=6
years)
       sin(2*pi*1.01*nino$Year), cos(2*pi*1.01*nino$Year) # freq=1.01
(period=.99 years)
)
```

Note In the above, as throughout, this is not how it appears in R; you have to remove some of the line breaks.

The following plots illustrate the contents of the data frame.

```
ggplot(data=subset(X, Year>1980)) + geom_line(aes(x=Year, y=X[X$Year>1980,3]))
ggplot(data=subset(X, Year>1980)) + geom_line(aes(x=Year, y=X[X$Year>1980,5]))
ggplot(data=subset(X, Year>1980)) + geom_line(aes(x=Year, y=X[X$Year>1980,7]))
ggplot(data=subset(X, Year>1980)) + geom_line(aes(x=Year, y=X[X$Year>1980,9]))
ggplot(data=subset(X, Year>1980)) + geom_line(aes(x=Year, y=X[X$Year>1980,11]))
```

Now let us see how well the series can be predicted using sines and cosines. Regress SST agains these. Linear regression is (of course) done using the `lm` (linear model) command:

```
mod <- lm(y ~ . - Year, data = X)  # Regress y on everything (but Year)
summary(mod)
```

Which frequencies are significant?

```
X$resid <- residuals(mod)
X$pred <- predict(mod)
ggplot(data = subset(X, Year > 1970)) + geom_line(aes(x = Year, y = y)) +
    geom_line(aes(x = Year, y = pred), color = "red")
```

Frequency Analysis The call `spec.pgram` calculates the periodogram using a fast Fourier transform.

```
raw.spec <- spec.pgram(nino$SST, taper = 0)
plot(raw.spec)
plot(raw.spec, log = "no")

# spec.df <- as.data.frame(raw.spec)
spec.df <- data.frame(freq = raw.spec$freq, spec = raw.spec$spec)
# Create a vector of periods to label on the graph, units are in years
yrs.period <- rev(c(1/6, 1/5, 1/4, 1/3, 0.5, 1, 3, 5, 10, 100))
yrs.labels <- rev(c("1/6", "1/5", "1/4", "1/3", "1/2", "1", "3", "5", "10",
    "100"))
yrs.freqs <- 1/yrs.period * 1/12  #Convert annual period to annual freq, and
then to monthly freq
spec.df$period <- 1/spec.df$freq
ggplot(data = subset(spec.df)) + geom_line(aes(x = freq, y = spec)) +
    scale_x_continuous("Period (years)",
    breaks = yrs.freqs, labels = yrs.labels) + scale_y_continuous()
```

The log scaling of the spectrum can be useful:

```
ggplot(data = subset(spec.df)) + geom_line(aes(x = freq, y = spec)) +
    scale_x_continuous("Period (years)",
    breaks = yrs.freqs, labels = yrs.labels) + scale_y_log10()

ggplot(data = subset(spec.df)) + geom_line(aes(x = freq, y = spec)) +
    scale_x_log10("Period (years)",
    breaks = yrs.freqs, labels = yrs.labels) + scale_y_log10()
```

Smoothing the Periodogram One way to smooth the periodogram is to integrate against a smoothing kernel. The Daniell kernel is one example. Look it up to find out what it is and try plotting it with the following parameter values:

```
plot(kernel("daniell", m = 10))  # A short moving average
plot(kernel("daniell", m = 50))  # A long moving average
plot(kernel("daniell", c(5, 5)))  # m=5 moving average of a m=5 moving average
plot(kernel("daniell", c(5, 5, 5)))  # a m=5 moving average of that!
```

Now try using it:

```
k = kernel("daniell", c(9, 9, 9))
smooth.spec <- spec.pgram(nino$SST, kernel = k, taper = 0)


# Note how the confidence interval got much narrower

spec.df <- data.frame(freq = smooth.spec$freq, ‘c(9,9,9)‘ = smooth.spec$spec)
names(spec.df) <- c("freq", "c(9,9,9)")
# Add other smooths
k <- kernel("daniell", c(9, 9))
spec.df[, "c(9,9)"] <- spec.pgram(nino$SST, kernel = k, taper = 0, plot =
FALSE)$spec
k <- kernel("daniell", c(9))
spec.df[, "c(9)"] <- spec.pgram(nino$SST, kernel = k, taper = 0, plot =
FALSE)$spec



# melt from wide format into long format
library(reshape2)
spec.df <- melt(spec.df, id.vars = "freq", value.name = "spec", variable.name =
"kernel")
plot1 <- ggplot(data = subset(spec.df)) + geom_path(aes(x = freq, y = spec,
    color = kernel)) + scale_x_continuous("Period (years)", breaks = yrs.freqs,
    labels = yrs.labels) + scale_y_log10()

plot2 <- ggplot(data = subset(spec.df)) + geom_path(aes(x = freq, y = spec,
    color = kernel)) + scale_x_log10("Period (years)", breaks = yrs.freqs,
labels = yrs.labels) +
    scale_y_log10()

grid.arrange(plot1, plot2)
```

Now try this with the SSA (anomaly) series and see what happens:

```
k = kernel("daniell", c(9, 9, 9))
smooth.spec <- spec.pgram(nino$SSA, kernel = k, taper = 0, plot = FALSE)
# spec.df <- as.data.frame(smooth.spec)
spec.df <- data.frame(freq = smooth.spec$freq, spec = smooth.spec$spec)
ggplot(data = subset(spec.df)) + geom_line(aes(x = freq, y = spec))
    + scale_x_continuous("Period (years)",
    breaks = yrs.freqs, labels = yrs.labels) + scale_y_continuous()
```

Tapering Besides windowing, another technique commonly applied is *tapering.*

When estimating a periodogram, there is an implicit assumption that the time series is circular, i.e. that it could be wrapped around, so that the time goes to $\pm\infty$. This is clearly a false assumption; if the series is wrapped around, there will be a jump where the end meets. This jump is spurious, but it will propagate itself through all the frequencies, contaminating them.

The solution is to downweight the beginning and end of the data. When the periodogram is calculated, more weight is given to the middle, and less weight to the ends. There is still the jump at the end, but it has very little weight, so its effect is diminished. This downweighting is called tapering. The question is, how much of the series should be downweighted. 5% at each end? 10%? 50% (i.e. the whole thing)? This can be determined empirically.

```
k = kernel("daniell", c(9, 9))
smooth.spec <- spec.pgram(nino$SSA, kernel = k, taper = 0, plot = FALSE)

spec.df <- data.frame(freq = smooth.spec$freq, ‘0%‘ = smooth.spec$spec)
names(spec.df) <- c("freq", "0%")
# Add other tapers
spec.df[, "10%"] <- spec.pgram(nino$SSA, kernel = k, taper = 0.1, plot =
FALSE)$spec
spec.df[, "30%"] <- spec.pgram(nino$SSA, kernel = k, taper = 0.3, plot =
FALSE)$spec

spec.df <- melt(spec.df, id.vars = "freq", value.name = "spec", variable.name =
"taper")
plot1 <- ggplot(data = subset(spec.df)) + geom_path(aes(x = freq, y = spec,
    color = taper)) + scale_x_continuous("Period (years)", breaks = yrs.freqs,
    labels = yrs.labels) + scale_y_log10()

plot2 <- ggplot(data = subset(spec.df)) + geom_path(aes(x = freq, y = spec,
    color = taper)) + scale_x_log10("Period (years)", breaks = yrs.freqs,
labels = yrs.labels) +
    scale_y_log10()

grid.arrange(plot1, plot2)
```

In practice, a 5% (from each side) often works well - and this gives good results with this data set .

```
k <- kernel("daniell", c(2))
spec.df[, "10%"] <- spec.pgram(nino$SSA, taper = 0.05)$spec
```

There are another set of spectral density estimates called "multitaper" estimates. Multitaper estimates can have good localisation in time. Multitaper esimates have two smoothing parameters. In the software, they are called "NW" and "k". Typically, k is set equal to 2NW-1, so NW is the only parameter . For any 'true' frequency signal, it will be resolved to within $\pm NW$ frequency intervals.

Confidence intervals are shown in the R implementation:

```
library(multitaper)
mt.spec <- spec.mtm(nino$SSA, nw = 16, k = 2 * 16 - 1, jackknife = TRUE,
dtUnits = "month")

# multitaper can resolve frequencies to about +/- NW/N Hz. i.e 16/1518 Hz
# k is typically equal to 2NW - 1.  Higher k is smoother
mt.spec <- spec.mtm(nino$SST, nw = 16, k = 2 * 16 - 1, jackknife = TRUE, dT =
1/12,
    dtUnits = "year")
```

Time-Frequency estimation One of the potential shortcomings of spectral analysis is the assumption that the time-series structure is stationary. You might want to evaluate this empirically.

Intuitively, you could cut your time series into different segments and calculate the periodgram separately for each one. Note, that since each interval is now shorter, you will have (1) less resolution between frequencies, and (2) you won't be able to detect low frequency effects as easily.

Now, you could imagine letting those segments overlap. This will allow you to see how periodogram is changing at various times. Finally, rather than just choosing segments, (where every datum in a segment gets a "weight" of 1, and every datum outside gets a weight of 0), you could choose segments by smoothly weighting points, giving more weight to the nearby time points, and less weight to the distant time points. This is precistly what wavelets do.

There are many types of wavelets. Not all of them estimate the periodogram. Some of them estimate, slope, for example. But one that estimates the periodogram is called the *morlet* wavelet. And the resulting plot is called a spectrogram.

```
library(dplR)
wave.out <- morlet(y1 = nino$SST, x1 = nino$Year, p2 = 8, dj = 0.1, siglvl =
0.95)
# p2=6 <=> estimate out to 2^8 = 256 months dj <=> controls the frequency
# resolution hack the period estimate to be in years, not months
wave.out$period <- wave.out$period/12
levs <- quantile(wave.out$Power, c(0, 0.25, 0.5, 0.75, 0.95, 1))
wavelet.plot(wave.out, wavelet.levels = levs, crn.ylim = c(22.5, 30))
```

We see that the annual component is strong at all time periods. There is a strong component at 3-7 years. That would be what we call El Nino. But it is noticeably absent between 1920 and 1960. This seemed to be a period of weakening in the El Nino/La Nina cycle. There also seems to be something going on at 12-16 years.

We can also calculate the 'averaged' wavelet. If we calculate the average across all times, we should get another estimate of the spectral density function.

```
wave.avg <- data.frame(power = apply(wave.out$Power, 2, mean), period =
(wave.out$period))
plot(wave.avg$period, wave.avg$power, type = "l")
```