paleo-spectral-analysis

Thomas Laepple, Andrew Dolman

2/26/24

Table of contents

Preface

This book is a guide to to power-spectrum based analysis of paleo-climate data. It is primarily intended for members of the Earth System Diagnostics group.

1 Introduction

2 Quick intro to PaleoSpec

PaleoSpec is an R package to assist in the spectral analysis of timeseries, in particular timeseries of climate variables from observational, model, and proxy paleoclimate data sources. PaleoSpec contains functions to analyse existing timeseries and to generate timeseries with specific spectral properties.

2.1 Installation

You can install the development version of PaleoSpec from GitHub with:

```
# install.packages("remotes")
remotes::install_github("EarthSystemDiagnostics/paleospec")
```

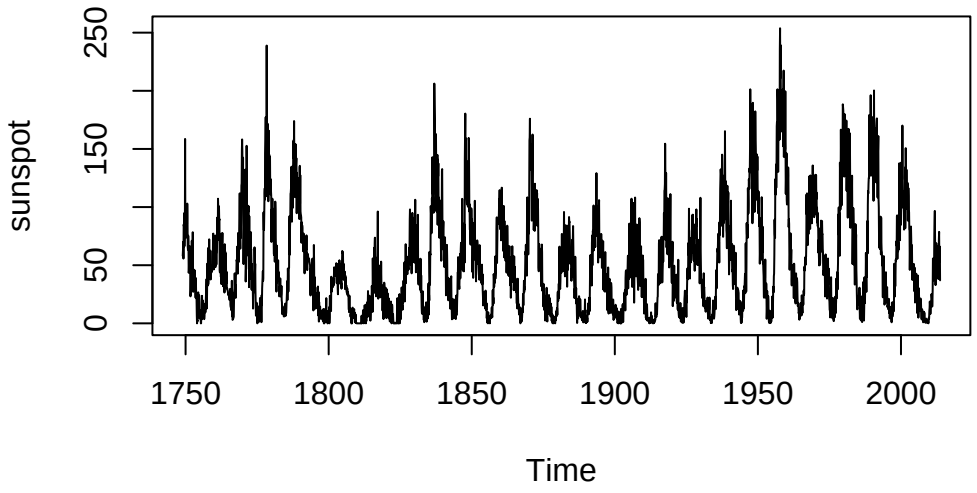
Please refer to function references here: https://earthsystemdiagnostics.github.io/paleospec/reference/index.htm

2.2 Usage

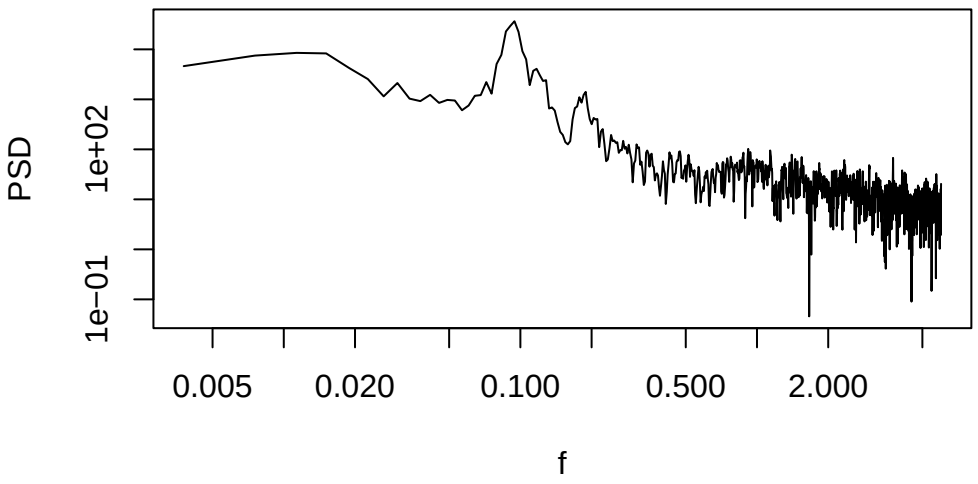
`SpecMTM` can be used to estimate the power spectrum of a timeseries using the multitaper method.

Here we estimate the spectrum of the monthly sunspot data that comes with R. The sunspot data are already a timeseries object so SpecMTM knows the correct frequency of the observations. We can plot the power spectrum with the PaleoSpec function LPlot.

```
sunspot <- datasets::sunspot.month
plot(sunspot)
```



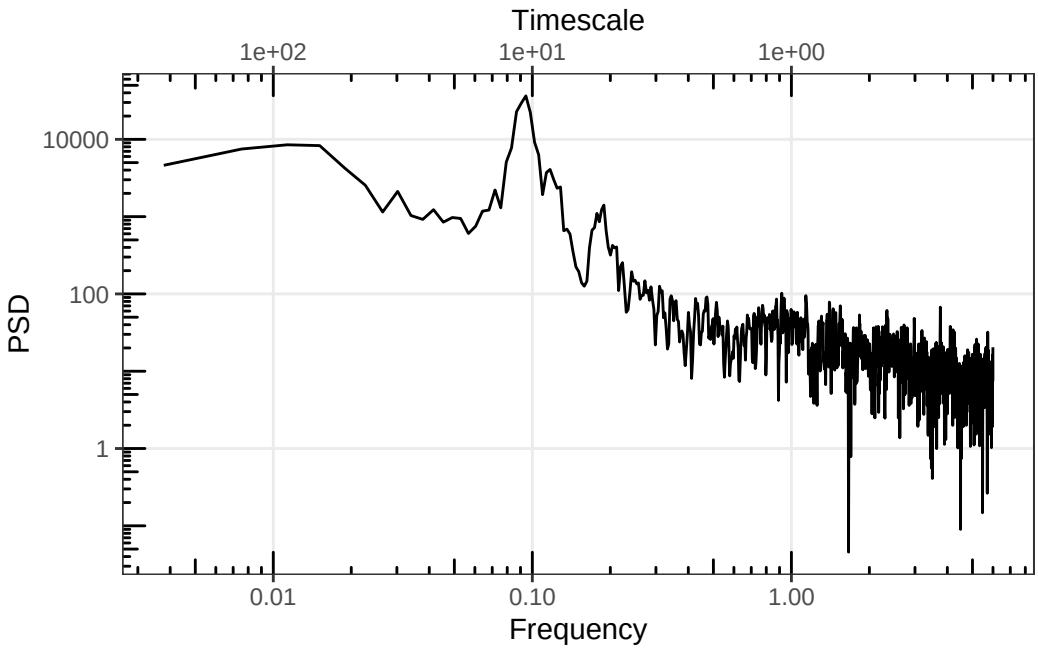

```
library(PaleoSpec)
sp_sun <- SpecMTM(sunspot)
LPlot(sp_sun)
```



Alternatively we can use the gg_spec() function to get a ggplot2

```
gg_spec(sp_sun)
```

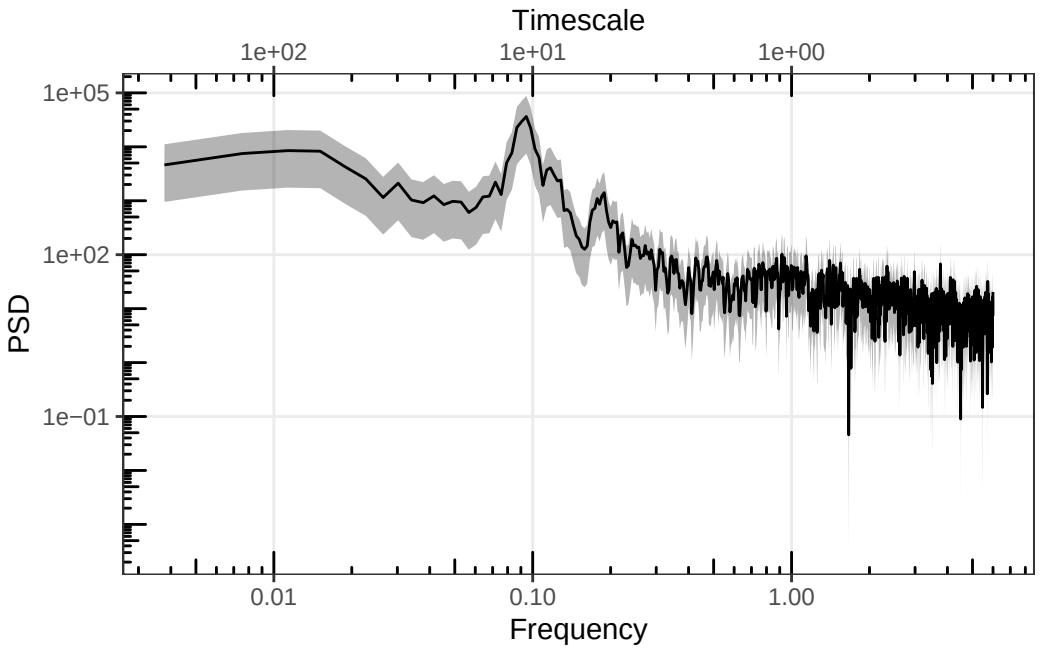
```
Scale for colour is already present.
Adding another scale for colour, which will replace the existing scale.
```



Appproximate confidence intervals can be added with the function AddConfInterval()

```
sp_sun <- AddConfInterval(sp_sun)
gg_spec(sp_sun)
```

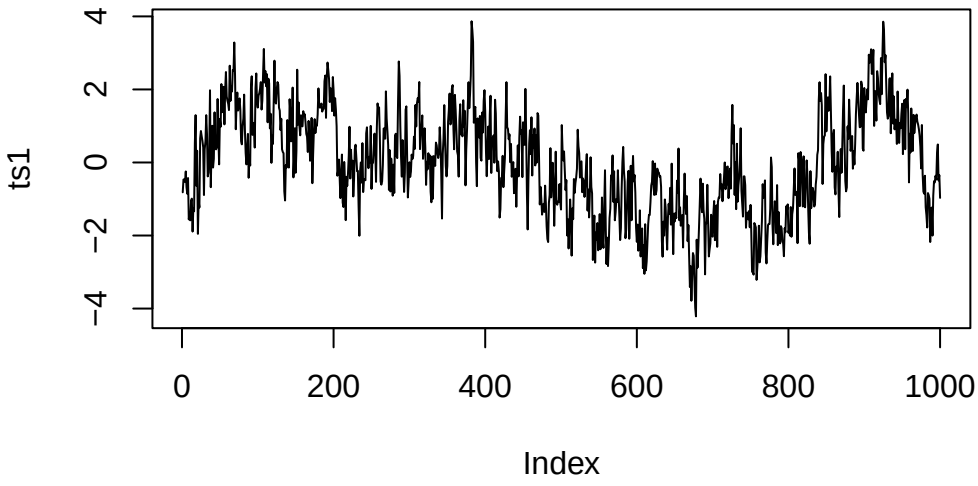
```
Scale for colour is already present.
Adding another scale for colour, which will replace the existing scale.
```



2.2.1 Simulating timeseries with given spectral properties

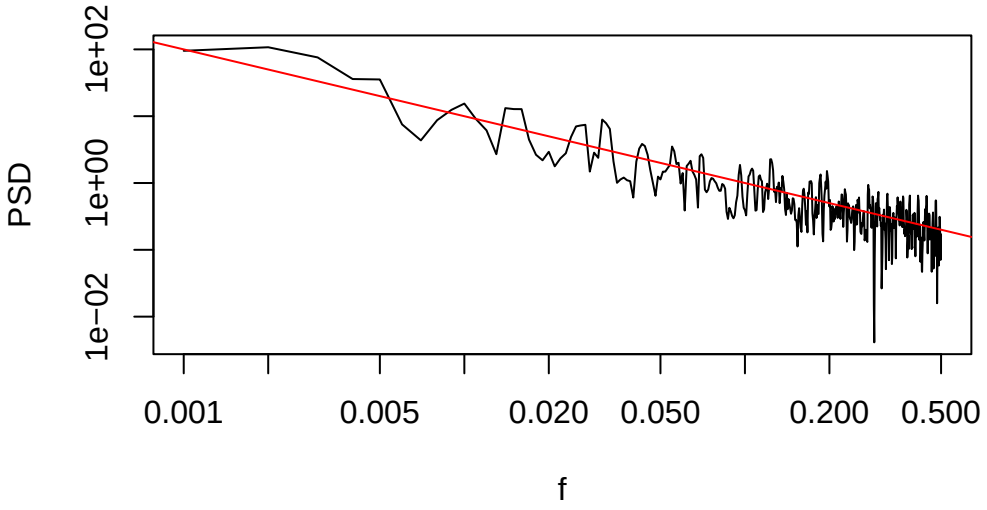
`SimPLS` can be used to create a timeseries whose power spectrum has powerlaw like properties, where: $S(f) = \alpha f^{-\beta}$

```
# setting the seed of the random number generator so that this example will
# always generate the same time series
set.seed(20221109)

# length of the time series
N <- 1e03

# parameters of the powerlaw spectrum
alpha <- 0.1
beta <- 1

ts1 <- SimPLS(N = N, b = beta, a = alpha)
plot(ts1, type = "l")
```



`SpecMTM` can again be used to estimate the power spectrum using the multitaper method. If we convert the vector from SimPLS to a timeseries object, and add information about the sampling frequency of the timeseries then SpecMTM will have the correct frequency axis.

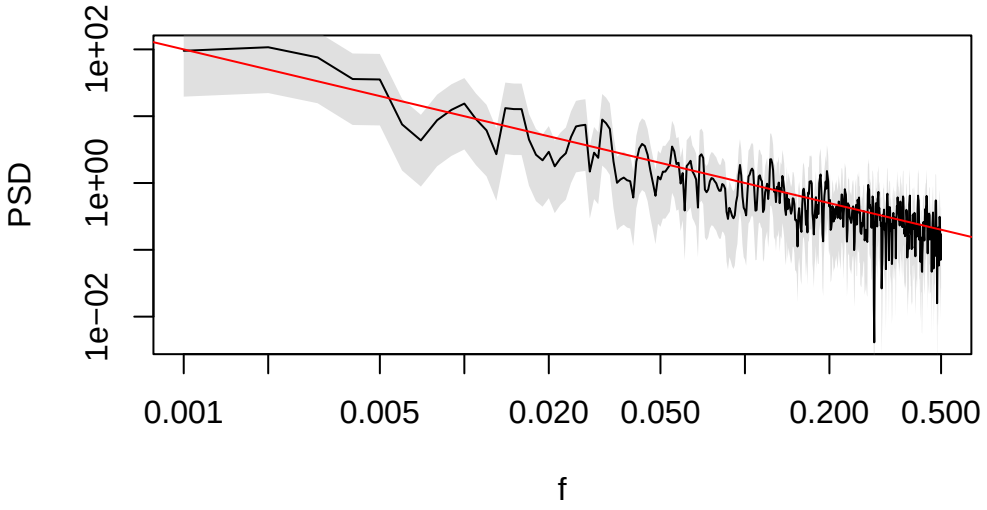
```
sp1 <- SpecMTM(ts(ts1, deltat = 1))

LPlot(sp1)
abline(log10(alpha), -beta, col = "red")
```



2.2.2 Smoothing and adding confidence intervals

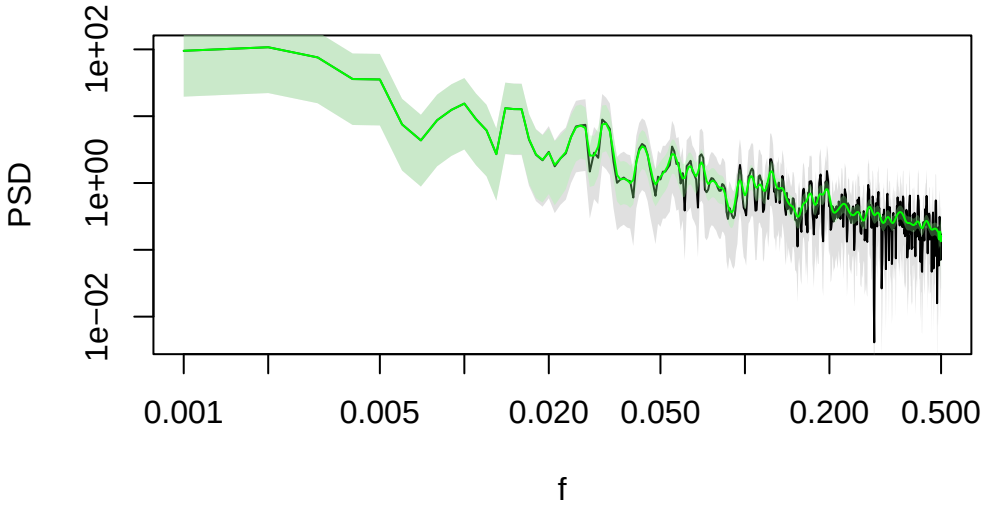
You can add confidence intervals to the spectral estimates with `AddConfInterval`

```
sp1 <- AddConfInterval(sp1)
LPlot(sp1)
abline(log10(alpha), -beta, col = "red")
```



The `LogSmooth` function can be used to smooth power spectra with equally spaced filter in log-space.

```
sp1_f <- LogSmooth(sp1, df.log = 0.01)
LPlot(sp1)
LLines(sp1_f, col = "green")
```


PSD
1e+02
1e+00
1e−02
0.001
0.005
0.020
0.050
0.200
0.500
f

Part I

Estimating powerspectra

3 Regular timeseries

3.1 The Fourier Transform

3.2 The Multitaper Method

3.3 Errors

- Explain Gamma / Chi-Sq nature of the errors
- Degrees of freedom / shape of Gamma
- Effect of tapering and smoothing on error distribution
- Confidence intervals

4 Irregular time-series

4.1 Interpolation

4.2 Binning

Part II

Variance

5 Variance from a powerspectrum

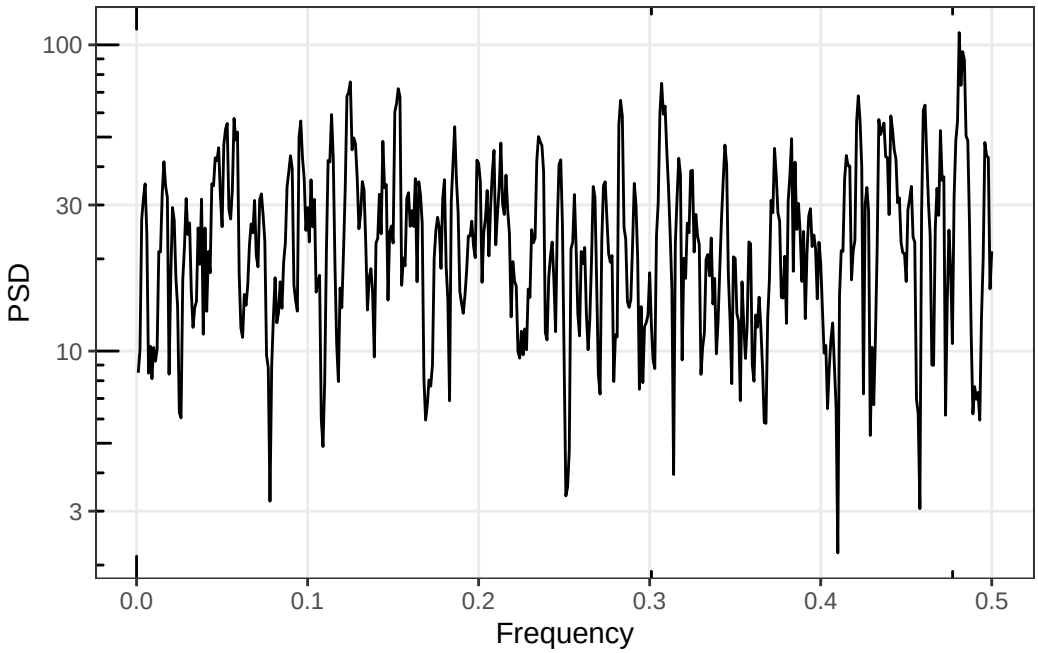
The integral of the full power spectrum is equal to the variance of the whole timeseries. We can calculate this by summing the spectral estimates and multiplying by delta_f.

```
library(PaleoSpec)
library(ggplot2)
```

```
set.seed(20240228)
N <- 1e03
delta_t = 1

ts1 <- ts(rnorm(N), deltat = delta_t) * 5
sp1 <- SpecMTM(ts1, detrend = TRUE)
```

```
gg_spec(sp1) +
  scale_x_continuous()
```

```
Scale for colour is already present.
Adding another scale for colour, which will replace the existing scale.
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```



Note that SpecMTM returns only the positive frequencies, therefore to get the total variance we must multiply by a factor of 2.

```
var(ts1)
```

```
[1] 25.29412
```

```
2 * sum(sp1$spec) * diff(sp1$freq)[1]
```

```
[1] 25.67279
```

Why are these not identical?

5.1 Variance by timescale

As we simulated a "white" timeseries, the total variance of the timeseries is spread equally across all frequencies. Therefore the integrals of the low and upper halves of the power spectrum are expected to be equal and sum to the variance of the whole timeseries.

PaleoSpec has a function `GetVarFromSpectra` which we will use to integrate sections of the power spectrum

```
f_range <- c(1/N, 1/2)
f_range_low <- c(1/N, mean(f_range))
f_range_high <- c(mean(f_range), 1/2)

PaleoSpec::GetVarFromSpectra(sp1, f_range)
```

```
$var
[1] 25.63878

$dof
[1] 959.4585
```

```
PaleoSpec::GetVarFromSpectra(sp1, f_range_low)
```

```
$var
[1] 12.97496

$dof
[1] 481.1547
```

```
PaleoSpec::GetVarFromSpectra(sp1, f_range_high)
```

```
$var
[1] 12.66096

$dof
[1] 478.2898
```

```
PaleoSpec::GetVarFromSpectra(sp1, f_range_low)$var +
PaleoSpec::GetVarFromSpectra(sp1, f_range_high)$var
```

```
[1] 25.63592
```

Part III

Simulating Timeseries

References