Newark_Cycle_Depth_Rank

R Markdown

(I) Synthetic Data -

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────── tidyverse 1.2.1 ──
```

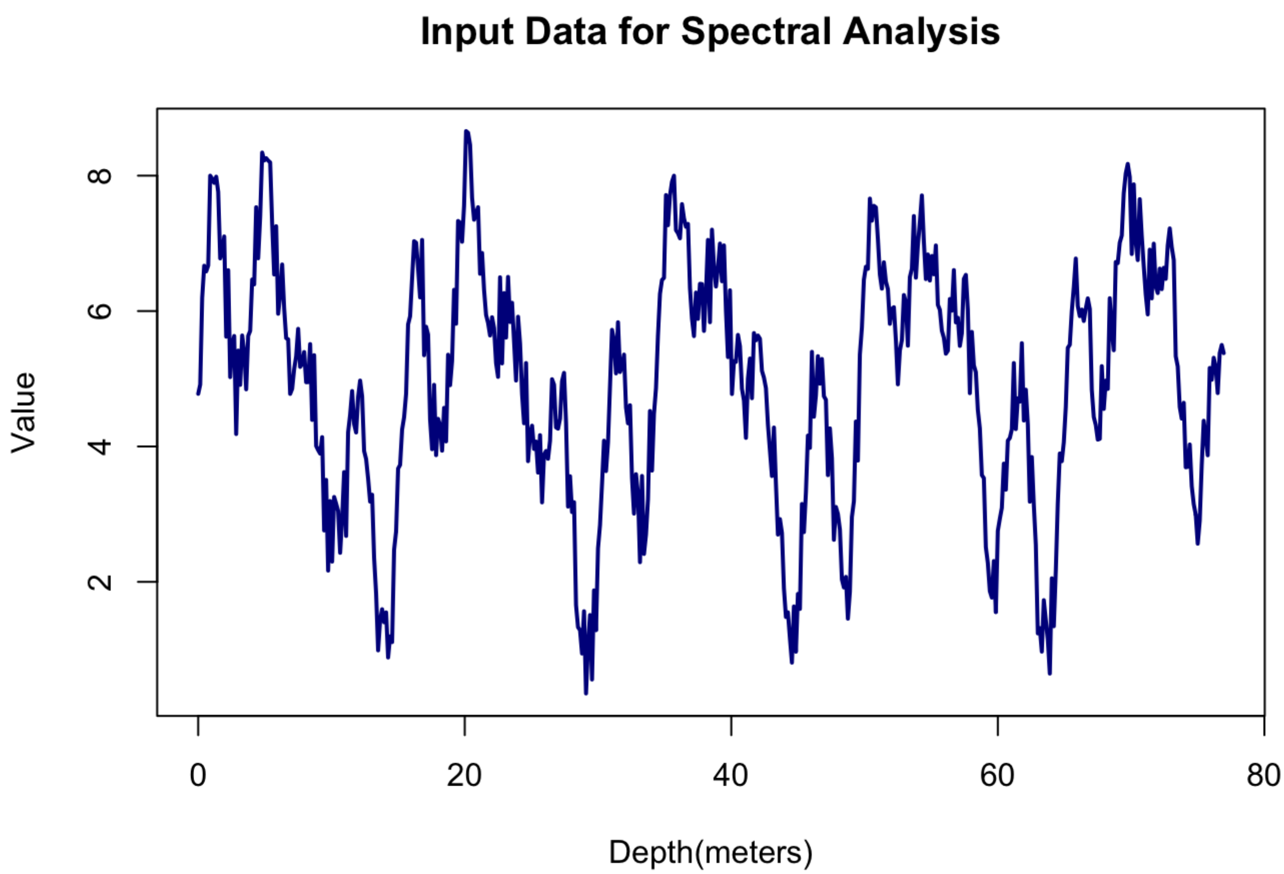
```
## ✔ ggplot2 2.2.1     ✔ purrr   0.2.4
## ✔ tibble  1.4.1     ✔ dplyr   0.7.4
## ✔ tidyr   0.7.2     ✔ stringr 1.2.0
## ✔ readr   1.1.1     ✔ forcats 0.2.0
```

```
## ── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(readxl)
### Step 1: Data Collection
# Read Lake Cycle excel sheet
lake_xls_f <- 'LakeCycleHomework_AnalysisExcel.XLS'
excel_sheets(lake_xls_f)
```

```
## [1] "Plot of Input"     "Input Data"        "Data Worksheet"
## [4] "Spectral Analysis" "Sheet4"
```

```
lake_xls <- read_excel(lake_xls_f, sheet='Input Data')

# Synthetic Data
synth_dat <- data.frame(lake_xls[8:521,c(1,2,3)])
colnames(synth_dat) <- c('Point', 'Depth(m)', 'Data')

plot(synth_dat$`Depth(m)`, synth_dat$Data, xlab='Depth(meters)',
     ylab='Value', type='l', col='darkblue', lwd=2,
     main='Input Data for Spectral Analysis'
     )
```



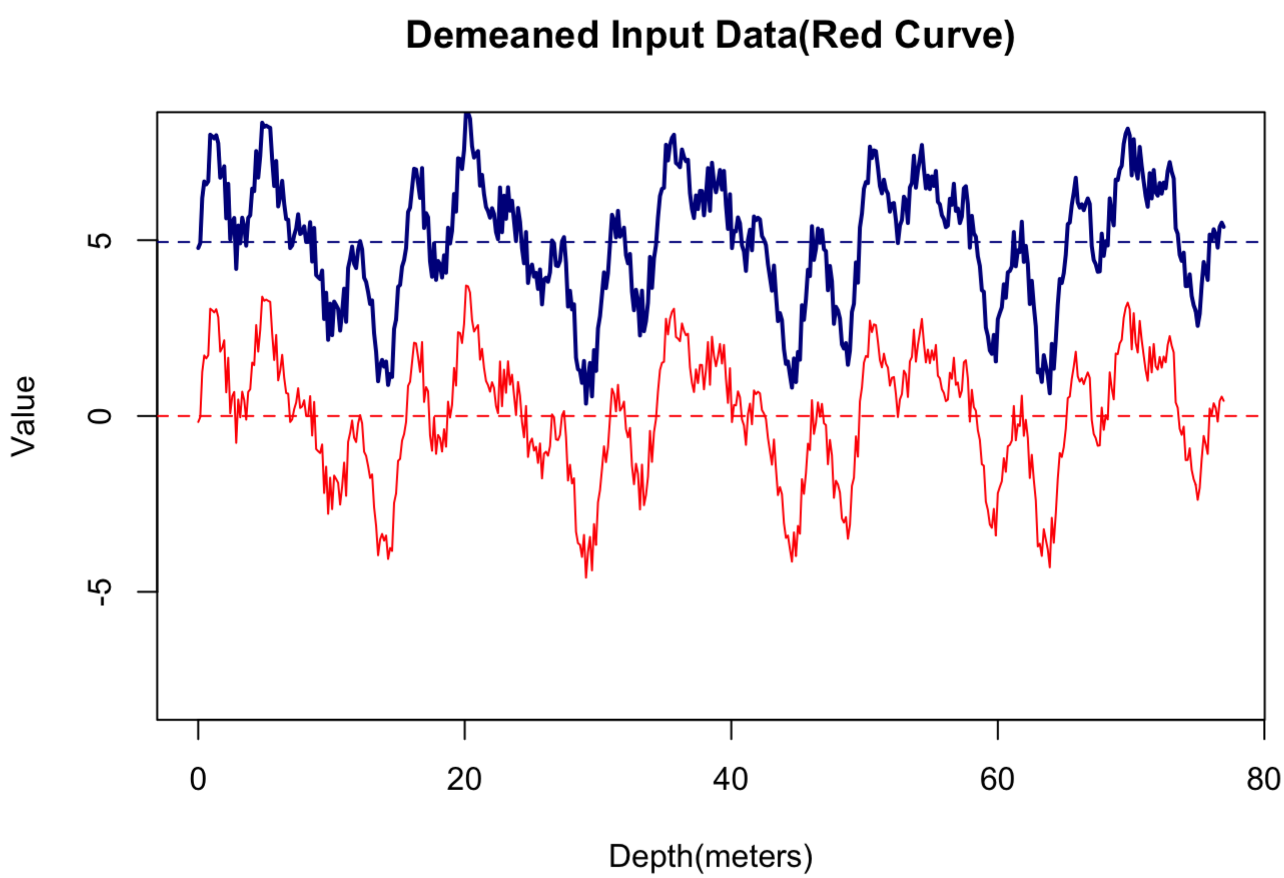

```
# Newark Basin Lake Sediments
# Selection from Interval I (lower lake series)
# 0.85m data spacing; 512  point section (+2 pts.)
nb_sed_d <- data.frame(lake_xls[8:521,c(6,7,8)])
colnames(nb_sed_d) <- c('Point', 'Depth (m)', 'Depth Rank')

# Simulated Hiatus
# Same Newark Interval
# With 100 pt removed (gap)
sim_hts_d <- data.frame(lake_xls[8:521,c(11,12,13)])
colnames(sim_hts_d) <- c('Point', 'Depth (m)', 'Depth Rank')

# Complete Interval I of Newark Basin
# Interval I = 2768-3370 m; 706 pts
nb_d <- data.frame(lake_xls[8:713,c(16,17,18)])
colnames(nb_d) <- c('Point', 'Depth (m)', 'Depth Rank')
```

```
# Depth Vec
dv <- as.numeric(nb_sed_d$`Depth (m)`)
diff(dv)
```

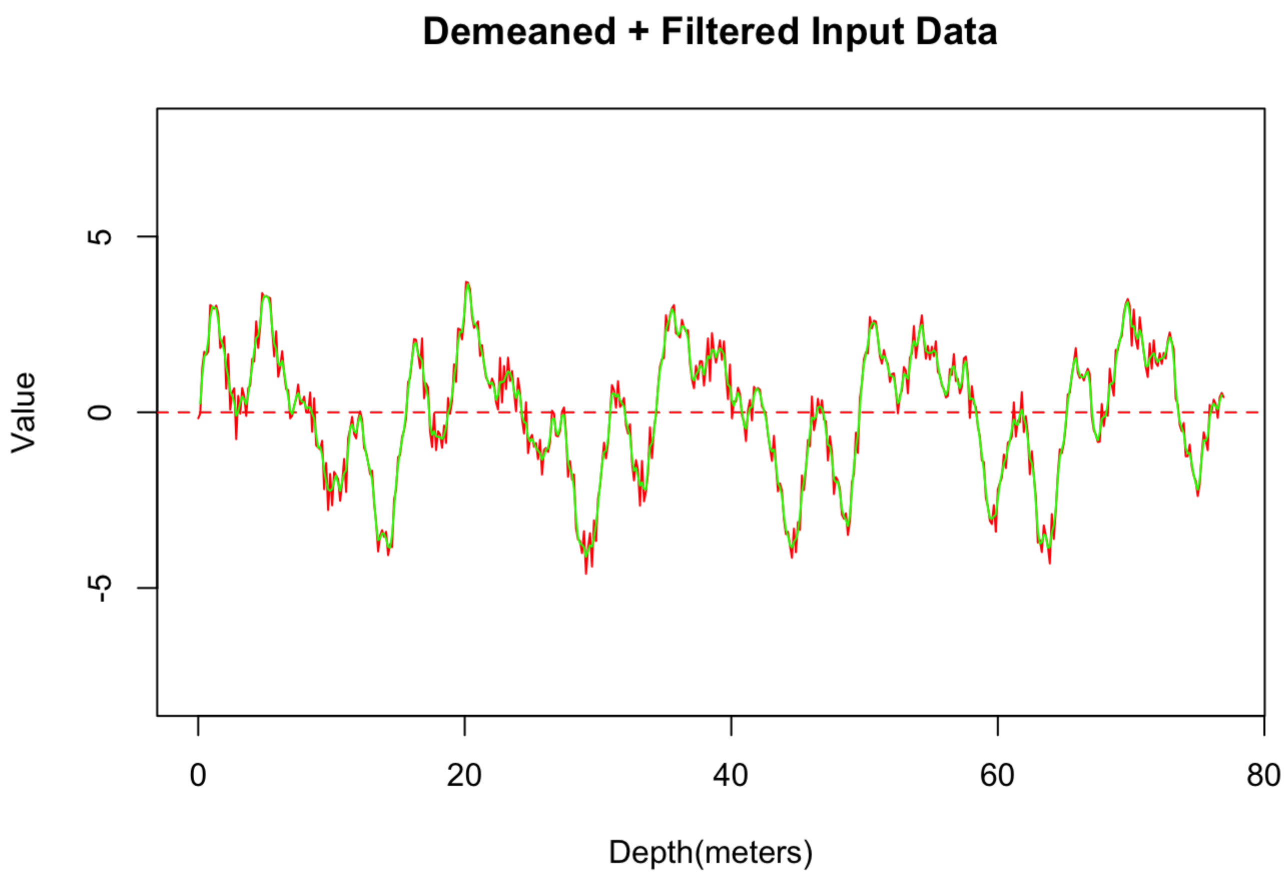
```
##   [1] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85
##  [15] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
##  [29] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
##  [43] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
##  [57] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
##  [71] 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
##  [85] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
##  [99] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85
## [113] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [127] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86
## [141] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [155] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [169] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [183] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [197] 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [211] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [225] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [239] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [253] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85
## [267] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [281] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86
## [295] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [309] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [323] 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [337] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [351] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [365] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [379] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [393] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [407] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85
## [421] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [435] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [449] 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85
## [463] 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [477] 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85
## [491] 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.85 0.86
## [505] 0.85 0.85 0.86 0.85 0.85 0.86 0.85 0.86 0.85
```

```
max_d <- max(dv)
min_d <- min(dv)
# window of depth
window <- max(dv)-min(dv)


#Mean of Points 1-512 =
synth_dat.avg <- mean(as.numeric((synth_dat$`Data`)[2:nrow(synth_dat)-1]))
```

```
#______________________________________________________________________________#
### Step 2. De-mean the data
# Demean synthetic dat
synth_dat.dm <- as.numeric(synth_dat$Data) - synth_dat.avg

# plot
plot(synth_dat$`Depth(m)`, synth_dat$Data, xlab='Depth(meters)',
     ylab='Value', type='l', col='darkblue', lwd=2,
     main='Demeaned Input Data(Red Curve)', ylim=c(-8,8)
)
abline(h=synth_dat.avg, lty=2, col='darkblue')
lines(synth_dat$`Depth(m)`, synth_dat.dm, col='red')
abline(h=0, lty=2, col='red')
```

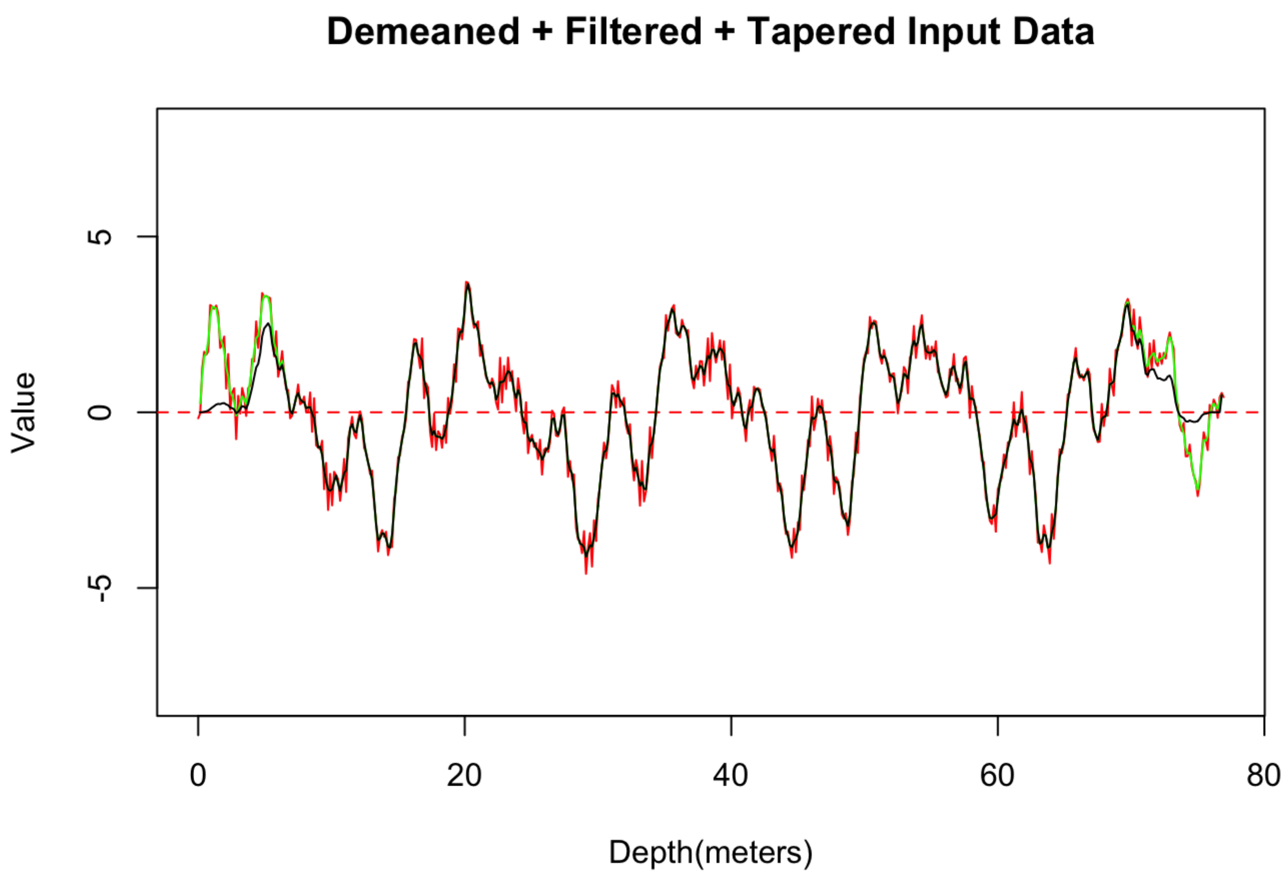


```
#__________________________________________________________________________________#
###

# Filtering - White noise removed
# (3pt moving "Hanning" average)
# 0.5*n + 0.25*(n-1 + n+1)
hanning_ma <- function(df, n) {
   x <- 0.5 * df[n] + 0.25*(df[n-1] + df[n+1])
   return(x)
}

synth_dat.filt <- c()
i <- 1
nr <- length(synth_dat.dm)
for (n in 2:(nr-1)) {
   h <- hanning_ma(synth_dat.dm, n)
   synth_dat.filt <- c(synth_dat.filt, h)
}

# plot
plot(synth_dat$`Depth(m)`, synth_dat.dm, xlab='Depth(meters)',
     ylab='Value', type='l', col='red', lwd=1,
     main='Demeaned + Filtered Input Data', ylim=c(-8,8)
)
abline(h=0, lty=2, col='red')
lines((synth_dat$`Depth(m)`)[2:(nr-1)], synth_dat.filt, col='green')
```



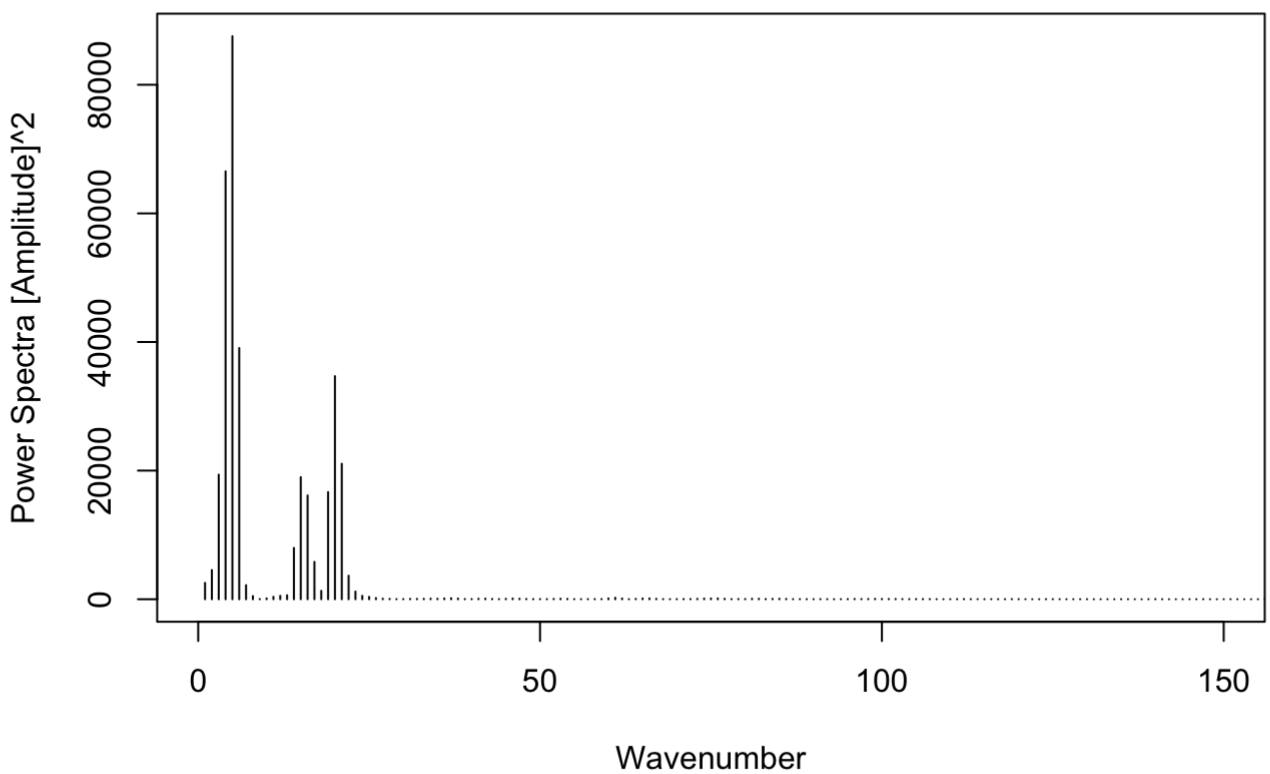
Demeaned + Filtered Input Data
Value
5
0
-5
0
20
40
60
80
Depth(meters)

```
#____________________________________________________________________________________#
###
synth_dat.tap <- synth_dat.filt
i <- 1
nr <- length(synth_dat$`Depth(m)`)
n <- length(synth_dat.filt)

tap_perc_n <- n * 10/100 # 10% tapering to reduce the effect of the terminations of data
 at the ends of the window

for (j in 1:tap_perc_n) {
  w = j * pi / tap_perc_n
  synth_dat.tap[j] = synth_dat.filt[j] * 0.5 * (1-cos(w))
}
for (j in (n-tap_perc_n):n) {
  w = (512-j) * pi / tap_perc_n
  synth_dat.tap[j] = synth_dat.filt[j] * 0.5 * (1-cos(w))
}
# plot
plot(synth_dat$`Depth(m)`, synth_dat.dm, xlab='Depth(meters)',
     ylab='Value', type='l', col='red', lwd=1,
     main='Demeaned + Filtered + Tapered Input Data', ylim=c(-8,8)
)
abline(h=0, lty=2, col='red')
lines((synth_dat$`Depth(m)`)[2:(nr-1)], synth_dat.filt, col='green')
lines((synth_dat$`Depth(m)`)[2:(nr-1)], synth_dat.tap, col='black')
```

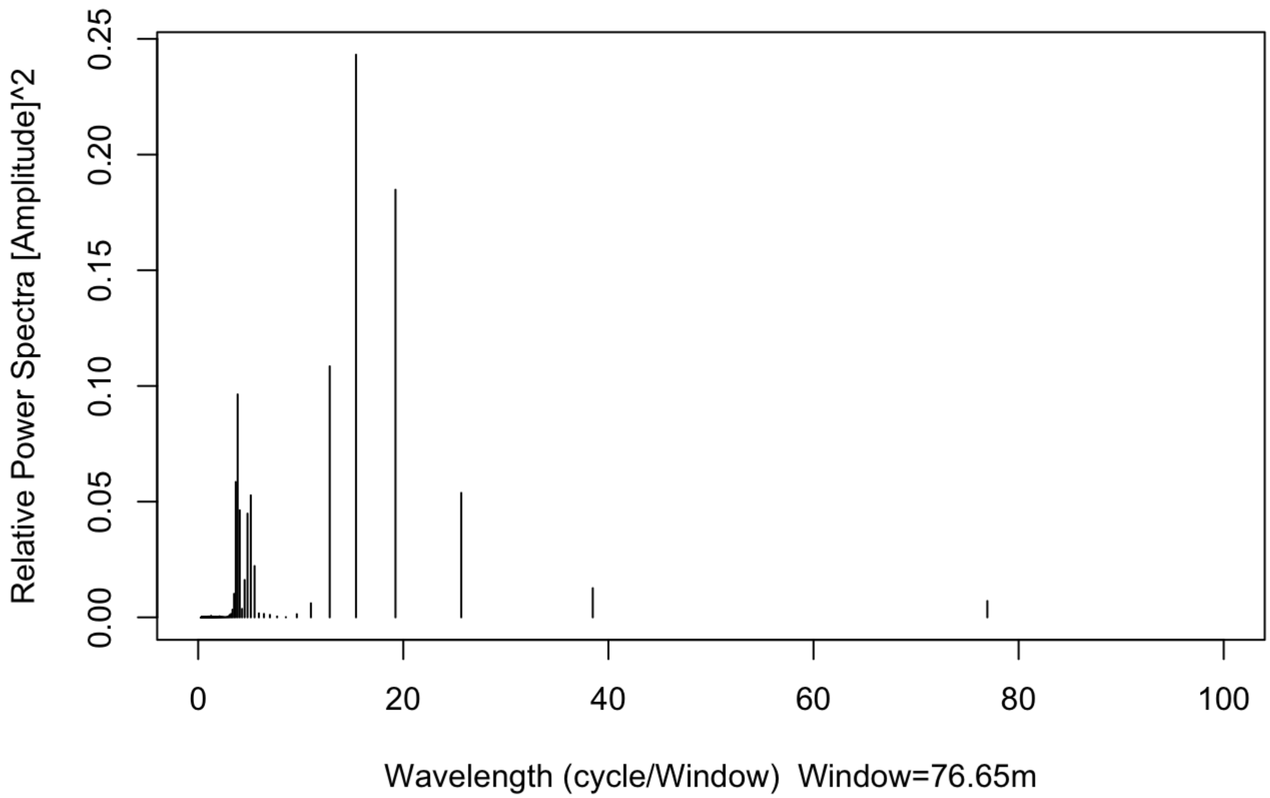



```
#________________________________________________________________________________#
###
synth_dat.fft <- fft(synth_dat.tap)

# Calculate Amplitude
Amp <- Mod(synth_dat.fft)

# Calculate Power Spectrum
PowerSpec <- Amp^2

# Take half of it , coz mirrored
PowerSpec <- PowerSpec[1:(length(Amp)/2)]

PowerSpec.han <- c()
i <- 1
nr <- length(PowerSpec)
for (n in 2:(nr-1)) {
  h <- hanning_ma(PowerSpec, n)
  PowerSpec.han <- c(PowerSpec.han, h)
}
```

```
#
dp <- as.numeric(synth_dat$`Depth(m)`)
window_width <- max(dp) - min(dp)
wavenum <- 1:(length(PowerSpec)-2)
wavelen <- window_width/wavenum

TotPowerSpec <- sum(PowerSpec)
RelPowerSpec <- PowerSpec.han/TotPowerSpec

# Charts of Wavenumber vs. Smoothed Power
plot(wavenum, PowerSpec.han, t='h',
     xlab='Wavenumber',
     ylab='Power Spectra [Amplitude]^2',
     xlim=c(0,150))
```




```
# )Wavelength vs. Relative Power Spectra
plot(wavelen, RelPowerSpec, t='h',
     xlab='Wavelength (cycle/Window)  Window=76.65m',
     ylab='Relative Power Spectra [Amplitude]^2',
     xlim=c(0,100))
```

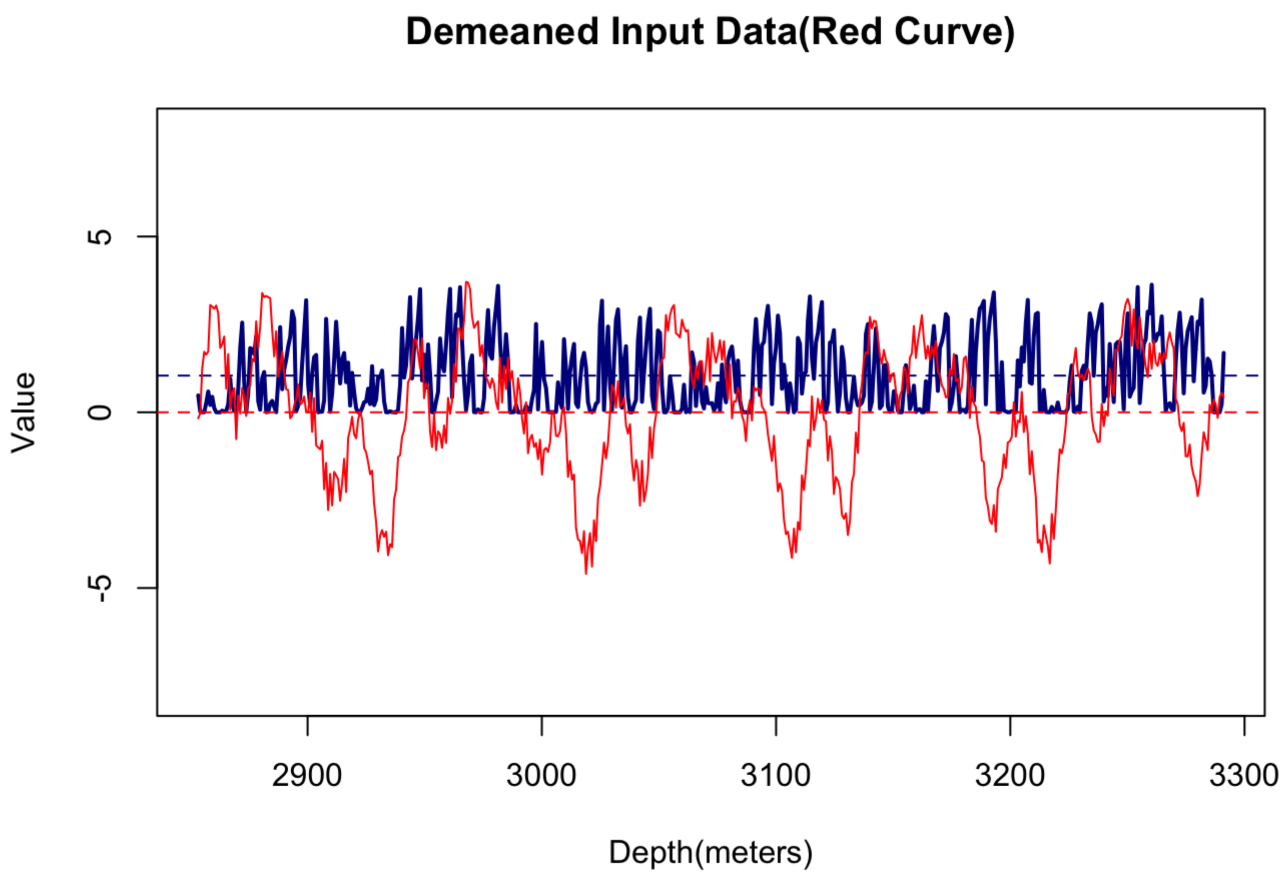



```
d <- data.frame(WaveLength=wavelen, RelativePowerSpectra=RelPowerSpec)
RelPowerSpec.sorted <- d[order(d['RelativePowerSpectra'], decreasing=TRUE),]
```

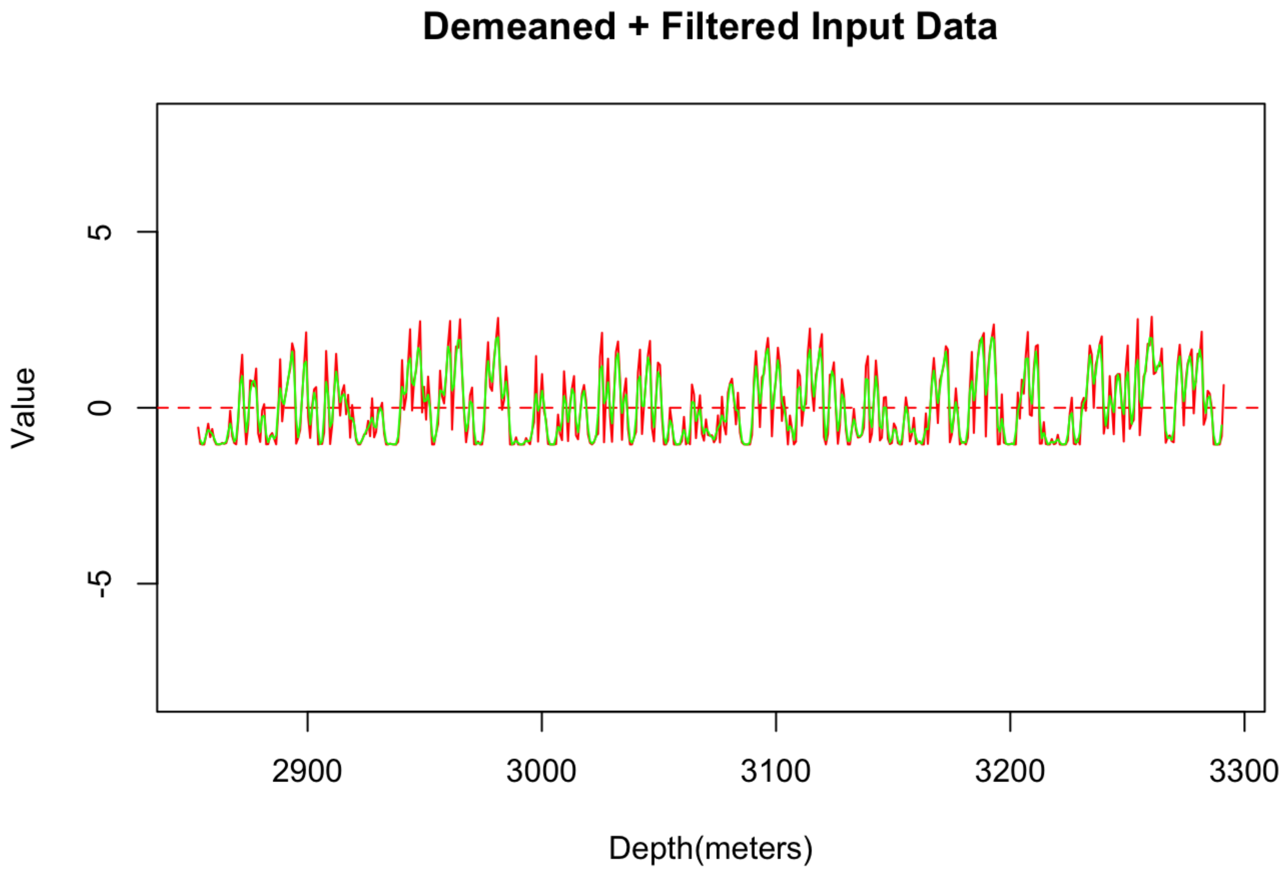
```
#__________________________________________________________________________________#
###
# Method #1:  Assuming a peak is a Milankovitch Cycle;  computing Sedimentation Rate
# We shall assume that the middle major peak, wihich has an apparent wavelength of about
 1/3rd of the large long-wavelength peak, is exactly the 123 k.y. eccentricity cycle.
#Eccentricity   412, 123,  and 95 k.y.
#Obliquity      41 k.y.
#Precession    23, and 19 k.y.


eccentricity <- 123

sed_rate <- (RelPowerSpec.sorted$WaveLength[1]/3) / (eccentricity/1000)
# Therefore, sedimentation rate is 41.70 m/m.y.

#For example, the Newark suite may exhibit peaks with wavelengths of 25.0 m
# and 19.23 m.  The ratio of 25.65/19.23 is  1.33.
# The ratio of the two eccentricity periods of 123 k.y. and 95 k.y. is 1.295,
# which is within 3% of the wavelength ratio.
# Therefore, within the limitations of integer-wavenumbers,
# these two peaks can be assigned to the two Eccentricity periods, and
# a more accurate sedimentation rate can be computed.
```

(II) Newark Drill-Core Data array

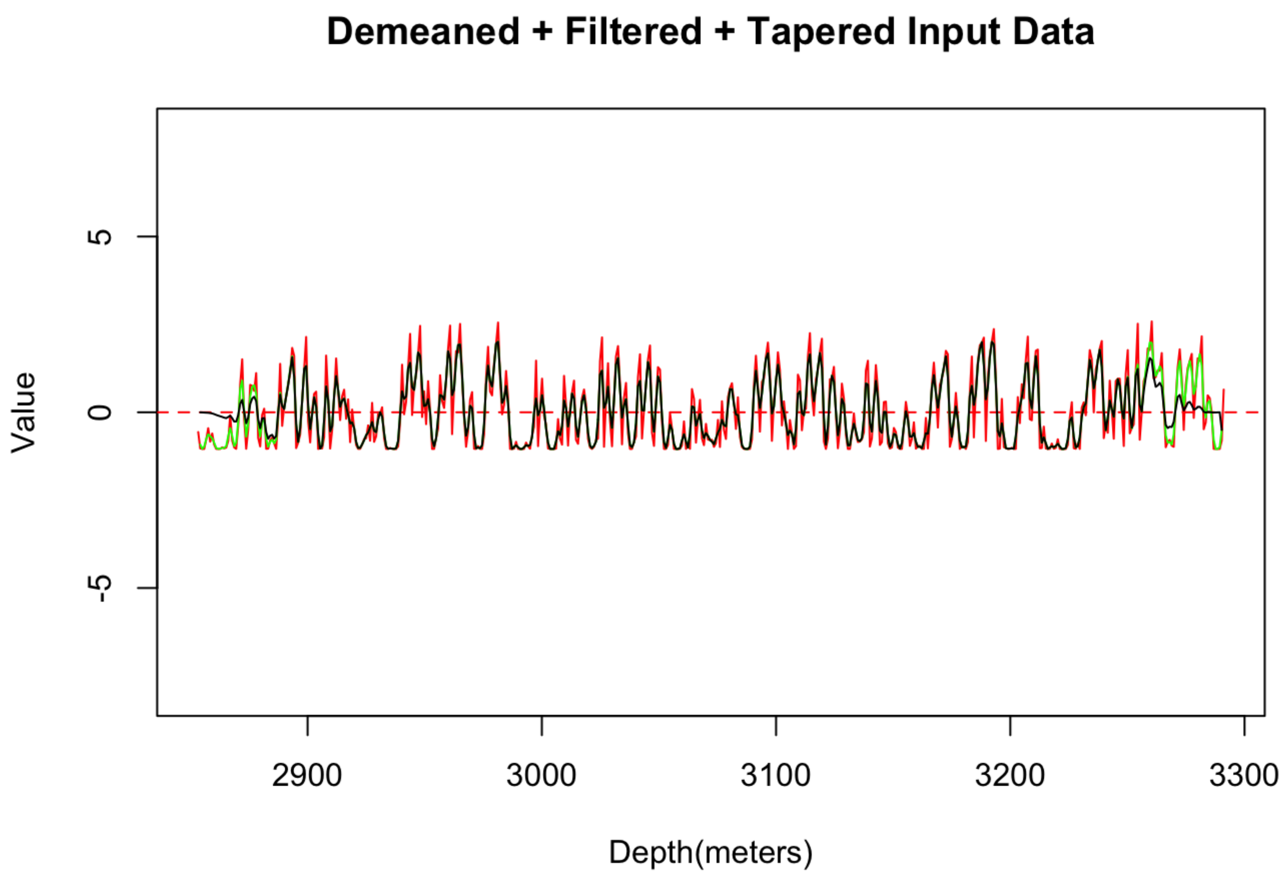
```
#__________________________________________________________________________________#
### Step 2. De-mean the data
# Demean synthetic dat
nb_sed_d.avg <- mean(as.numeric(nb_sed_d$`Depth Rank`))
nb_sed_d.dm <- as.numeric(nb_sed_d$`Depth Rank`) - nb_sed_d.avg

# plot
plot(nb_sed_d$`Depth (m)`, nb_sed_d$`Depth Rank`, xlab='Depth(meters)',
     ylab='Value', type='l', col='darkblue', lwd=2,
     main='Demeaned Input Data(Red Curve)', ylim=c(-8,8)
)
abline(h=nb_sed_d.avg, lty=2, col='darkblue')
lines(nb_sed_d$`Depth (m)`, synth_dat.dm, col='red')
abline(h=0, lty=2, col='red')
```


Demeaned Input Data(Red Curve)
Value
5
0
-5
2900
3000
3100
3200
3300
Depth(meters)

```
#______________________________________________________________________________#
###

# Filtering - White noise removed
# (3pt moving "Hanning" average)
# 0.5*n + 0.25*(n-1 + n+1)
hanning_ma <- function(df, n) {
   x <- 0.5 * df[n] + 0.25*(df[n-1] + df[n+1])
   return(x)
}

nb_sed_d.filt <- c()
i <- 1
nr <- length(nb_sed_d.dm)
for (n in 2:(nr-1)) {
   h <- hanning_ma(nb_sed_d.dm, n)
   nb_sed_d.filt <- c(nb_sed_d.filt, h)
}

# plot
plot(nb_sed_d$`Depth (m)`, nb_sed_d.dm, xlab='Depth(meters)',
     ylab='Value', type='l', col='red', lwd=1,
     main='Demeaned + Filtered Input Data', ylim=c(-8,8)
)
abline(h=0, lty=2, col='red')
lines((nb_sed_d$`Depth (m)`)[2:(nr-1)], nb_sed_d.filt, col='green')
```

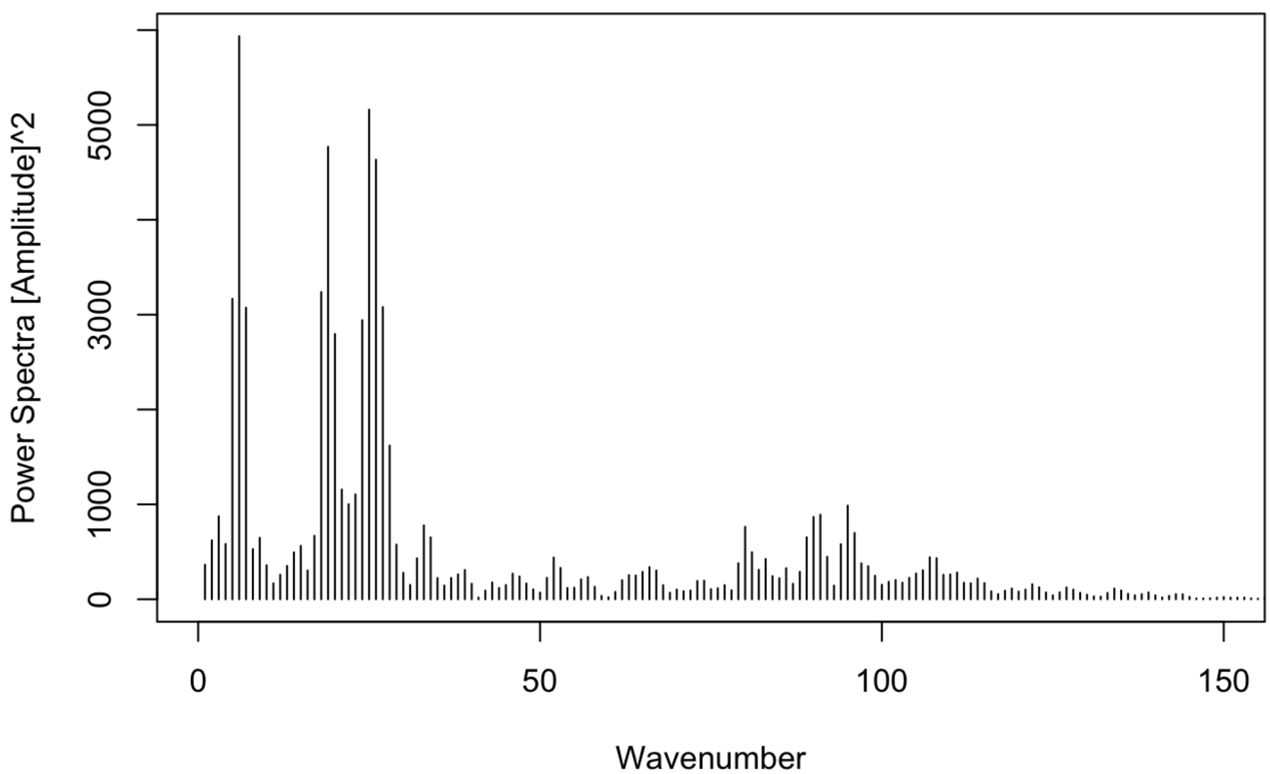

Demeaned + Filtered Input Data
Value
5
0
-5
2900
3000
3100
3200
3300
Depth(meters)

```
#_______________________________________________________________________________#
###
nb_sed_d.tap <- nb_sed_d.filt
i <- 1
nr <- length(as.numeric(nb_sed_d$`Depth (m)`))
n <- length(nb_sed_d.filt)

tap_perc_n <- n * 10/100 # 10% tapering to reduce the effect of the terminations of data
 at the ends of the window

for (j in 1:tap_perc_n) {
  w = j * pi / tap_perc_n
  nb_sed_d.tap[j] = nb_sed_d.filt[j] * 0.5 * (1-cos(w))
}
for (j in (n-tap_perc_n):n) {
  w = (512-j) * pi / tap_perc_n
  nb_sed_d.tap[j] = nb_sed_d.filt[j] * 0.5 * (1-cos(w))
}
# plot
plot(nb_sed_d$`Depth (m)`, nb_sed_d.dm, xlab='Depth(meters)',
     ylab='Value', type='l', col='red', lwd=1,
     main='Demeaned + Filtered + Tapered Input Data', ylim=c(-8,8)
)
abline(h=0, lty=2, col='red')
lines(as.numeric(nb_sed_d$`Depth (m)`)[2:(nr-1)], nb_sed_d.filt, col='green')
lines(as.numeric(nb_sed_d$`Depth (m)`)[2:(nr-1)], nb_sed_d.tap, col='black')
```

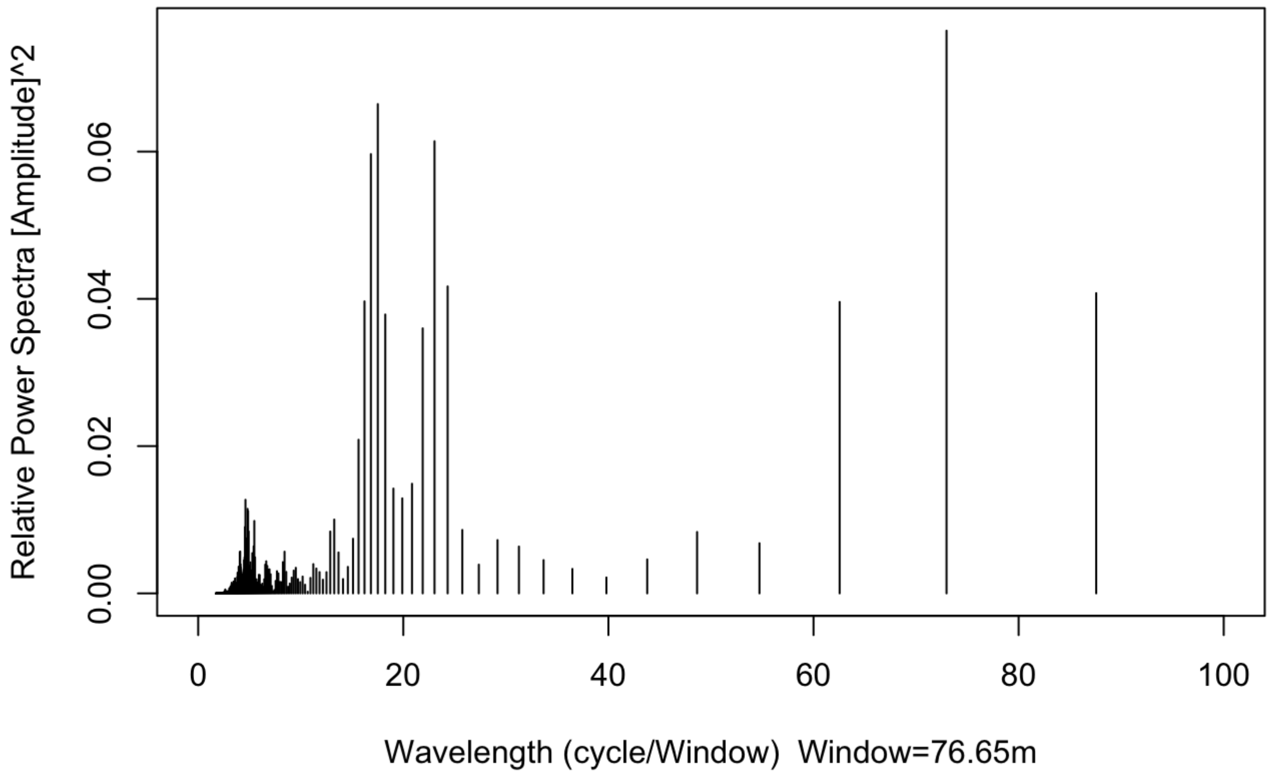



```
#_______________________________________________________________________________#
###
nb_sed_d.fft <- fft(nb_sed_d.tap)

# Calculate Amplitude
Amp <- Mod(nb_sed_d.fft)

# Calculate Power Spectrum
PowerSpec <- Amp^2

# Take half of it , coz mirrored
PowerSpec <- PowerSpec[1:(length(Amp)/2)]

PowerSpec.han <- c()
i <- 1
nr <- length(PowerSpec)
for (n in 2:(nr-1)) {
  h <- hanning_ma(PowerSpec, n)
  PowerSpec.han <- c(PowerSpec.han, h)
}
```

```
#
dp <- as.numeric(nb_sed_d$`Depth (m)`)
window_width <- max(dp) - min(dp)
wavenum <- 1:(length(PowerSpec)-2)
wavelen <- window_width/wavenum

TotPowerSpec <- sum(PowerSpec)
RelPowerSpec <- PowerSpec.han/TotPowerSpec

# Charts of Wavenumber vs. Smoothed Power
plot(wavenum, PowerSpec.han, t='h',
     xlab='Wavenumber',
     ylab='Power Spectra [Amplitude]^2',
     xlim=c(0,150))
```




```
# )Wavelength vs. Relative Power Spectra
plot(wavelen, RelPowerSpec, t='h',
     xlab='Wavelength (cycle/Window)  Window=76.65m',
     ylab='Relative Power Spectra [Amplitude]^2',
     xlim=c(0,100))
```




```
d <- data.frame(WaveLength=wavelen, RelativePowerSpectra=RelPowerSpec)
RelPowerSpec.sorted <- d[order(d['RelativePowerSpectra'], decreasing=TRUE),]
```

```
#________________________________________________________________________________#
###
# Method #1:  Assuming a peak is a Milankovitch Cycle;  computing Sedimentation Rate
# We shall assume that the middle major peak, wihich has an apparent wavelength of about
 1/3rd of the large long-wavelength peak, is exactly the 123 k.y. eccentricity cycle.
#Eccentricity   412, 123,  and 95 k.y.
#Obliquity      41 k.y.
#Precession    23, and 19 k.y.


eccentricity <- 123

sed_rate <- (RelPowerSpec.sorted$WaveLength[1]/3) / (eccentricity/1000)
# Therefore, sedimentation rate is 197.75 m/m.y.

# a more accurate sedimentation rate can be computed.
```