Pearson Type IV curve fit to the frequency domain

Keyong

March 27, 2018

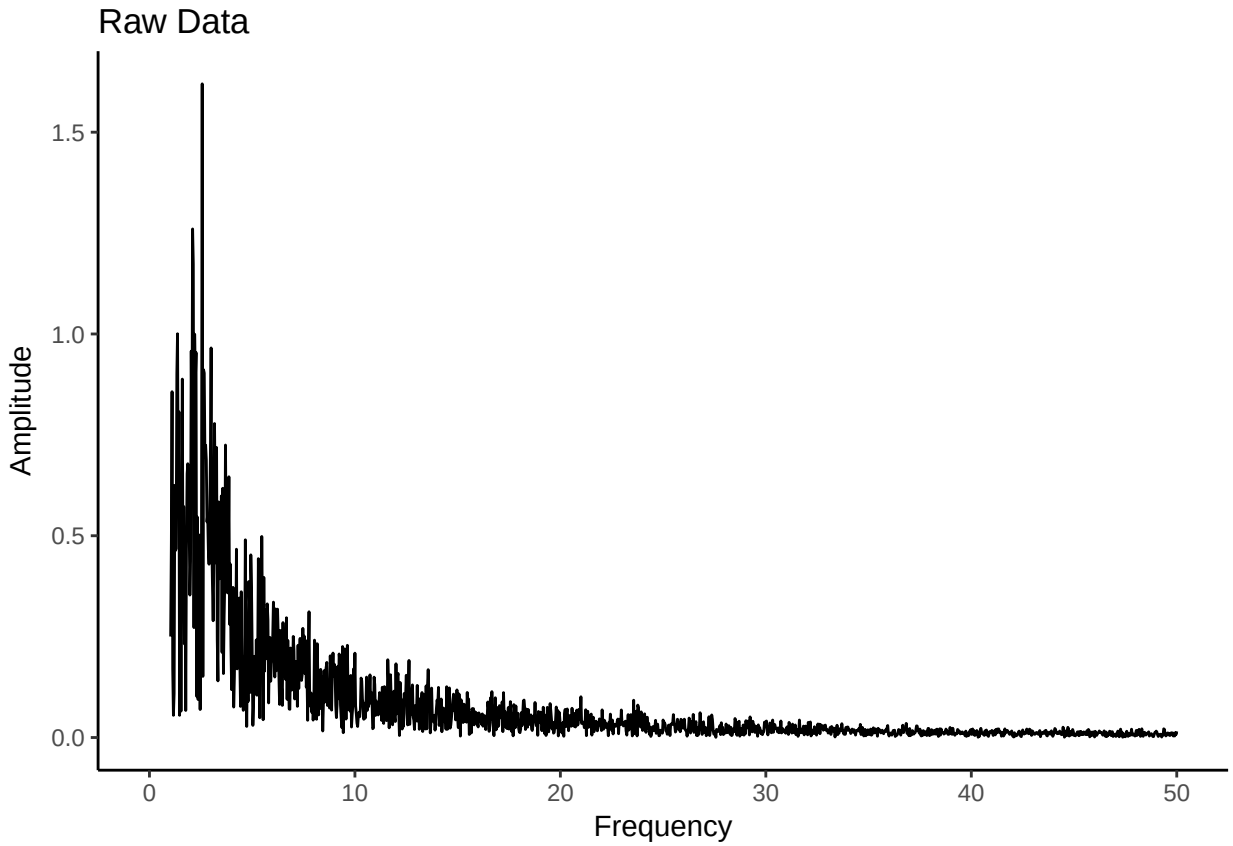
Let's read in the data

```
# Load the readxl library
library(readxl)

# Read in data as a data frame
fsp <- read_xlsx("17217_frequency_spectrum.xlsx")
fsp
```

```
## # A tibble: 149,970 x 2
##        freq      amp
##       <dbl>    <dbl>
##  1 1.033330 0.250259
##  2 1.066663 0.482672
##  3 1.099996 0.856958
##  4 1.133330 0.170223
##  5 1.166663 0.054996
##  6 1.199996 0.109106
##  7 1.233329 0.624864
##  8 1.266662 0.463502
##  9 1.299996 0.468146
## 10 1.333329 0.901121
## # ... with 149,960 more rows
```

Let's plot the data

```
# Load the dplyr and ggplot2 libraries
library(dplyr)
library(ggplot2)

# Plot the data as a line
ggplot(aes(freq, amp), data = fsp) +
    geom_line() + xlim(0, 50) + theme_classic() +
    ggtitle("Raw Data") +
    xlab("Frequency") + ylab("Amplitude")
```



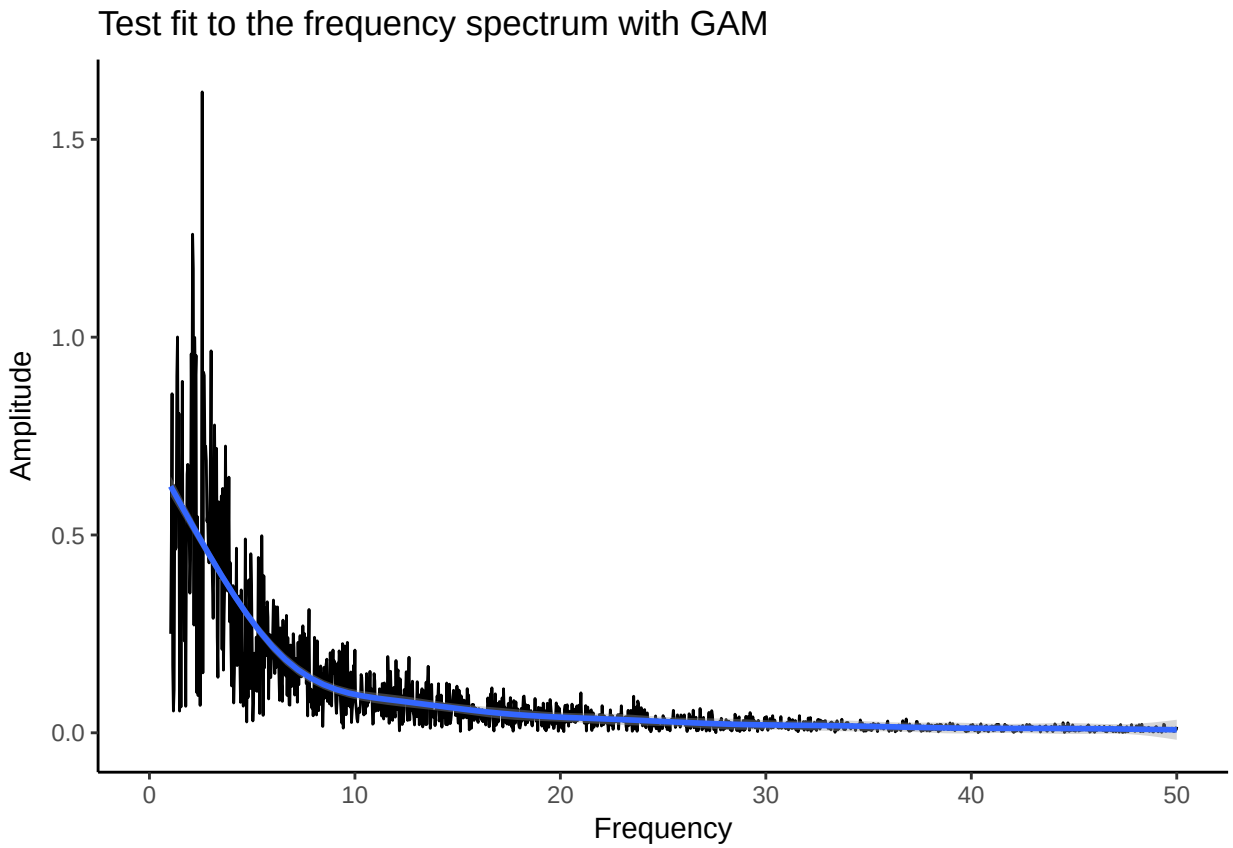
To fit to a density function, we first need to calculate the **area under the curve**

```
# Load the DescTools library
library(DescTools)

# Calculate the area under the curve
area <- AUC(x = fsp$freq, y = fsp$amp)
area
```

```
## [1] 4.915767
```

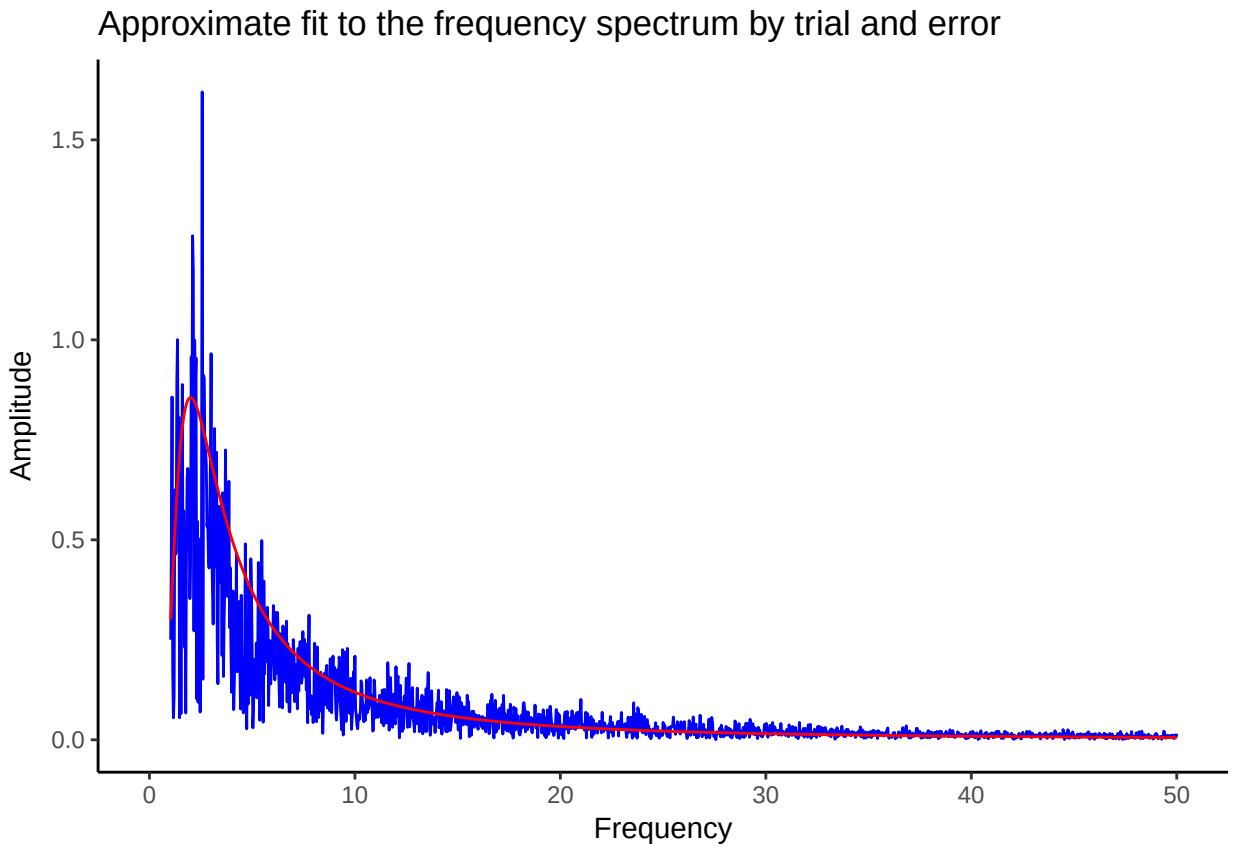
We want to fit to a **Pearson Type IV density function**. Let's first try to use `geom_smooth()` (which uses the generalized additive model (gam) method):

```
# Load required libraries
library(gsl)
library(PearsonDS)

# Define the formula for curve fitting
pearson4Curve <- amp ~ area * dpearsonIV(freq, m, nu, location, scale)

# Plot the data with a fitted curve
ggplot(aes(freq, amp), data = fsp) +
    geom_line() + xlim(0, 50) + theme_classic() +
    geom_smooth(formula = pearson4Curve) +
    ggtitle("Test fit to the frequency spectrum with GAM") +
    xlab("Frequency") + ylab("Amplitude")
```



Since it didn't work very well, let's first use trial and error to find a Pearson Type IV density curve that approximately matches our data:

```
# Define starting parameters
m0 <- 1
nu0 <- -6
location0 <- 0.5
scale0 <- 0.5

# Add a column for the predicted amplitude values
fsp <-
    fsp %>%
    mutate(ampPredicted = area *
            dpearsonIV(freq, m = m0, nu = nu0, location = location0, scale = scale0))

# Plot the data with the Pearson Type IV curve
ggplot(aes(freq, amp), data = fsp) +
    geom_line(color = 'Blue') + xlim(0, 50) + theme_classic() +
    geom_line(aes(freq, ampPredicted), color = 'Red') +
    ggtitle("Approximate fit to the frequency spectrum by trial and error") +
    xlab("Frequency") + ylab("Amplitude")
```



Now we have initial guesses for the 4 parameters, let's use **nonlinear least squares** method to get a better fit to the data:

```
# Use nonlinear least squares to do curve fitting
model <-
    nls(formula = pearson4Curve, data = fsp,
        start = list(m = m0, nu = nu0, location = location0, scale = scale0))
model
```

```
## Nonlinear regression model
##   model: amp ~ area * dpearsonIV(freq, m, nu, location, scale)
##    data: fsp
##        m       nu location    scale
##   0.7343  -0.9472   1.3447   1.1341
##  residual sum-of-squares: 9.375
##
## Number of iterations to convergence: 24
## Achieved convergence tolerance: 9.677e-06
```

Use the **broom** package to tidy the results

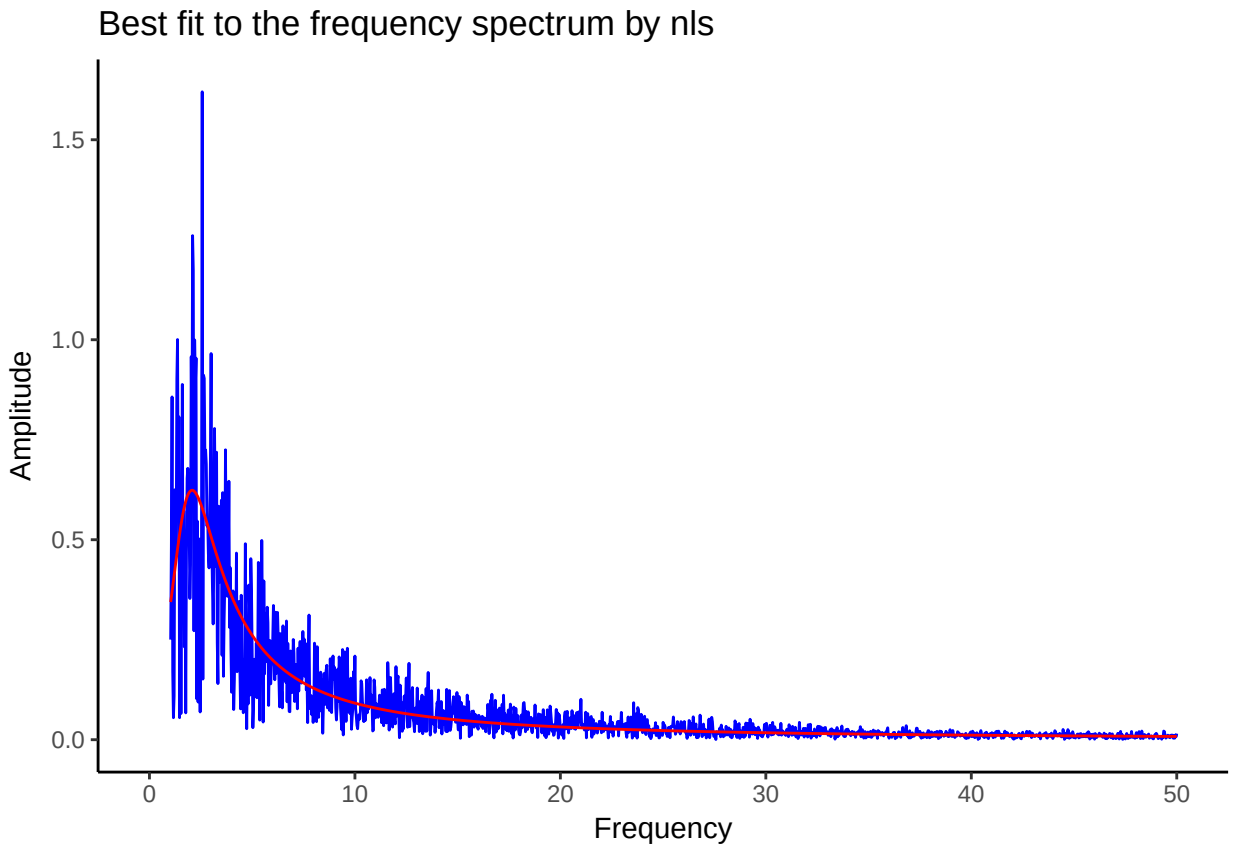
```
# Import the broom library
library(broom)

# Tidy the model
results <- tidy(model)
results
```

```
##       term   estimate   std.error statistic p.value
```

```
## 1        m  0.7343259 0.001353247 542.63977       0
## 2       nu -0.9471546 0.016402752 -57.74364       0
## 3 location  1.3446970 0.011623758 115.68522       0
## 4    scale  1.1340622 0.011964901  94.78242       0
```

Finally, plot the data with the refined Pearson Type IV model

```
# Add a column for the predicted amplitude values
fsp <-
    fsp %>%
    mutate(ampPredicted = area *
            dpearsonIV(freq, m = results$estimate[1], nu = results$estimate[2],
                       location = results$estimate[3], scale = results$estimate[4]))

# Plot the data with the Pearson Type IV curve
ggplot(aes(freq, amp), data = fsp) +
    geom_line(color = 'Blue') + xlim(0, 50) + theme_classic() +
    geom_line(aes(freq, ampPredicted), color = 'Red') +
    ggtitle("Best fit to the frequency spectrum by nls") +
    xlab("Frequency") + ylab("Amplitude")
```




```
# Find the maximum of the fit
maxAmpPredicted <- max(fsp$ampPredicted)
maxAmpPredicted
```

```
## [1] 0.623565
```

```
# Print the row withthe maximum amplitude
fsp %>%
```

```
  filter(ampPredicted == maxAmpPredicted)
```

```
## # A tibble: 1 x 3
##      freq      amp ampPredicted
##     <dbl>    <dbl>        <dbl>
## 1 2.06666 0.700164     0.623565
```