APPENDIX 1: Determining the circadian phases using the package dendrometeR

Jorge Palmero-Barrachina

16/2/2022

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

This R Markdown is an example of how to use the dendrometeR package for determining the circadian phases from band-automatic dendrometer data.

The input data from automatic dendrometer should look like it is shown below

```
library(readxl) #Importing the data from band automatic dendrometers
data <- read_excel("D:/Mlady_vedec/definitive/Dendro_and_sapflow_excels/site 4_2019_incr_from April to A
  col_types = c("date", "numeric", "numeric",
                "numeric", "numeric", "numeric",
                "numeric", "numeric", "numeric",
                "numeric", "numeric", "numeric",
                "numeric", "numeric", "numeric",
                "numeric", "numeric", "numeric",
                "numeric"))

data <- data.frame(Time = as.POSIXct(data$Time),
                   Sensor1 = as.numeric(data$`3.  Increment [mm] 1/3 4C1`)*1000,
                   Sensor2 = as.numeric(data$`7.  Increment [mm] 2/3 4C3`)*1000)
```

```
head(data)
```

```
##                  Time  Sensor1  Sensor2
## 1 2019-04-01 01:00:00 5568.000 6109.000
## 2 2019-04-01 02:00:00 5567.667 6109.334
## 3 2019-04-01 03:00:00 5568.501 6111.167
## 4 2019-04-01 04:00:00 5570.667 6108.334
## 5 2019-04-01 05:00:00 5574.000 6107.834
## 6 2019-04-01 06:00:00 5577.333 6105.334
```

```
summary(data)
```

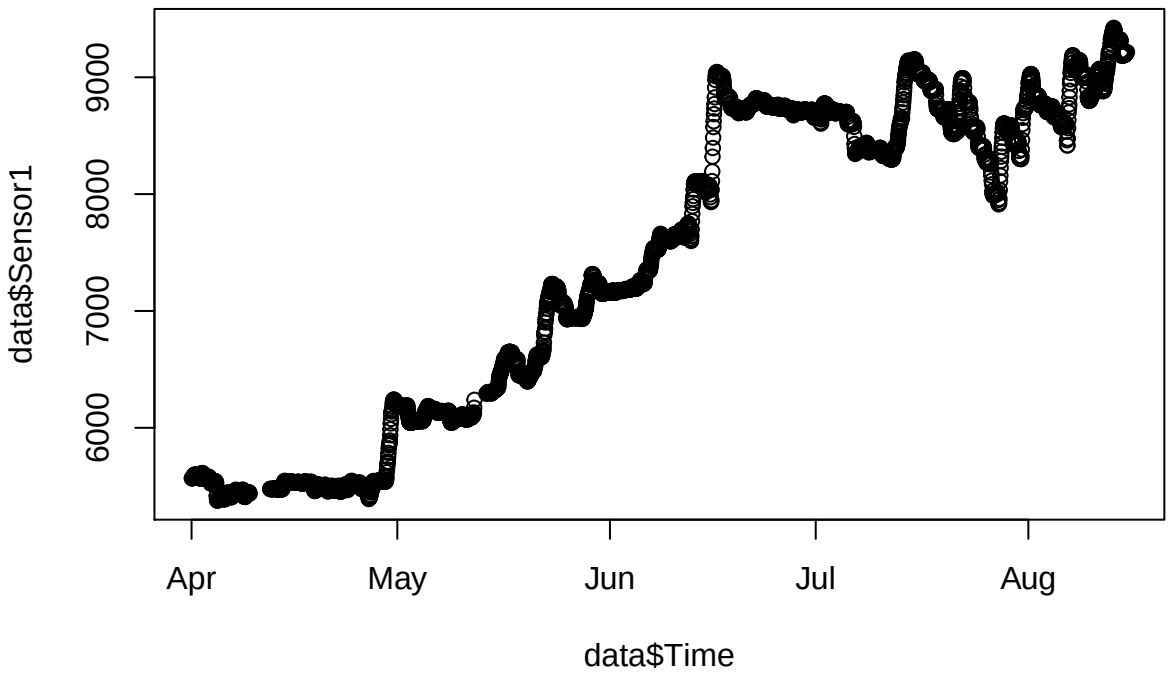
```
##       Time                          Sensor1        Sensor2
##  Min.   :2019-04-01 01:00:00   Min.   :5376   Min.   : 5231
```

```
##  1st Qu.:2019-05-05 03:00:00   1st Qu.:6134   1st Qu.: 6280
##  Median :2019-06-08 05:00:00   Median :7635   Median : 7694
##  Mean   :2019-06-08 05:00:00   Mean   :7449   Mean   : 7672
##  3rd Qu.:2019-07-12 07:00:00   3rd Qu.:8716   3rd Qu.: 8860
##  Max.   :2019-08-15 09:00:00   Max.   :9423   Max.   :10584
##                                NA's   :112    NA's   :112
```

```
length(data[,1])
```

```
## [1] 3273
```

```
plot(data$Sensor1~data$Time)
```



For using the dendrometeR's package, it is necessary to call the names of the data frame as the Time column. Then, we can delete the time column, and let only columns with sensor¡s data.

```
rownames(data) <- data$Time
data$Time <- NULL
knitr::kable(
  data[1:10,],
  caption = "Example of how the data looks like before using dendrometeR's package"
)
```

Table 1: Example of how the data looks like before using dendrometeR's package

	Sensor1	Sensor2
2019-04-01 01:00:00	5568.000	6109.000
2019-04-01 02:00:00	5567.667	6109.334
2019-04-01 03:00:00	5568.501	6111.167
2019-04-01 04:00:00	5570.667	6108.334
2019-04-01 05:00:00	5574.000	6107.834
2019-04-01 06:00:00	5577.333	6105.334
2019-04-01 07:00:00	5583.333	6106.833
2019-04-01 08:00:00	5591.834	6106.833
2019-04-01 09:00:00	5593.834	6105.167
2019-04-01 10:00:00	5597.333	6081.501

To check if the input data is well prepared for dendrometer's package it is necessary to use the function "is.dendro".

```
library(dendrometeR)
```

```
## Warning: package 'dendrometeR' was built under R version 3.6.3
```

```
## Loading required package: forecast
```

```
## Warning: package 'forecast' was built under R version 3.6.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Loading required package: pspline
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
is.dendro(data) #TRUE, the fata frame is prepared to be used with dendrometeR
```

```
## [1] TRUE
```

```
dendro.resolution(data) #3600 (default is in seconds 3600 seg = 1 hour
```

```
## [1] 3600
```

```
dendro.resolution(data, unts = "hours") #1 hour intervals
```

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

It is necessary to check if data contains gaps. If data contains gaps, daily and cycle approaches cannot be executed by dendrometeR's package

```
TRUE %in% is.na(data) #If true is returned, data contains gaps
```

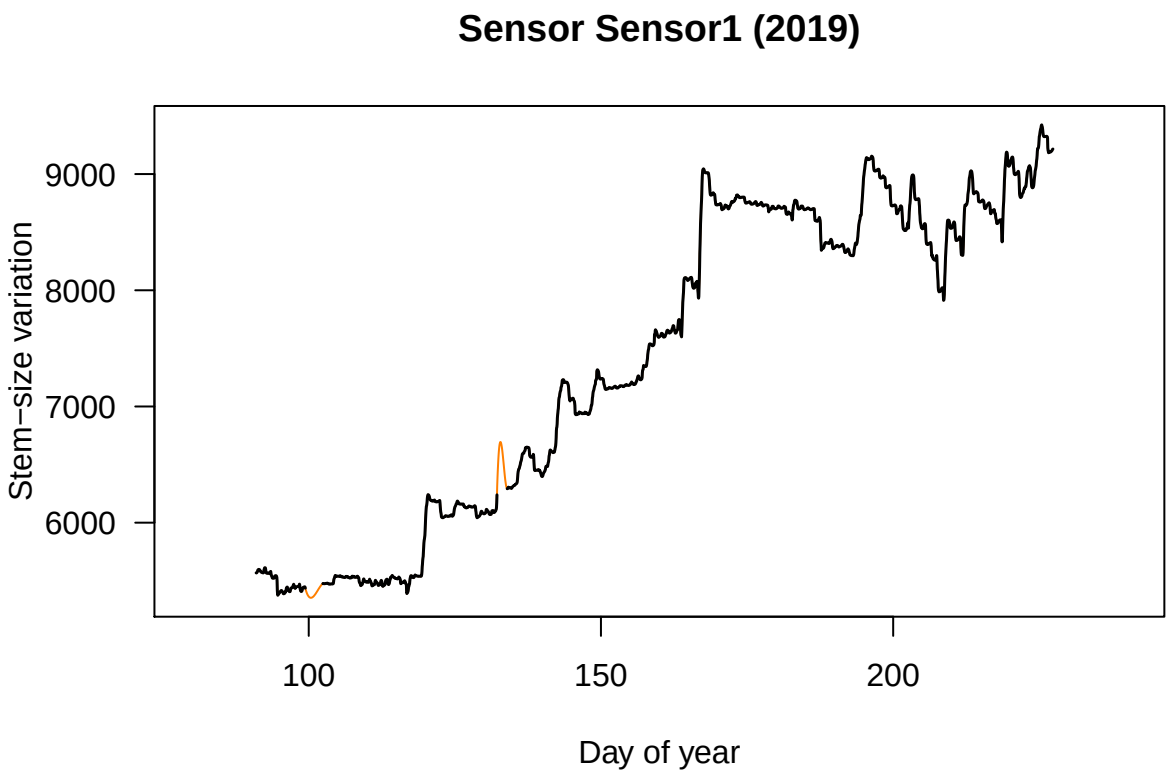
```
## [1] TRUE
```

As our data contains gaps, we will fil the gaps using an ARIMA model (fill_gaps function).

```
data_gf <- fill_gaps(data, Hz = 0.01, season = FALSE)
TRUE %in% is.na(data_gf) #if FALSE is returned data do not contain gaps
```

```
## [1] FALSE
```

```
fill_plot(data, data_gf, sensor = 1, year = NULL) #This plot, shows in orange
```




```
#the gaps which were filled using time series The ARIMA model cannot sensibly
#handle long gaps, i.e. lasting over more than a day (Van der Maaten et al., 2016).
```

Once the data does not contain gaps, a daily approach can worked out using the function daily_stats. This approach calculates the minimum and maximum values for each days, and calculates the difference between them "amplitude". Timing of each aximum and minimum values are showed as well.

```
daily_data_1 <- daily_stats(data_gf, sensor = 1, value = "max") #for only sensor 1
knitr::kable(
  daily_data_1[1:10,],
  caption = "Example of the daily approach for one sensor"
)
```

Table 2: Example of the daily approach for one sensor

dmID	date	DOY	min	mean	max	amplitude	time_min	time_max
Sensor1	2019-04-01	91	5567.667	5583.870	5598.333	30.66635	2019-04-01 02:00:00	2019-04-01 12:00:00
Sensor1	2019-04-02	92	5564.000	5581.396	5612.500	48.50006	2019-04-02 23:00:00	2019-04-02 14:00:00
Sensor1	2019-04-03	93	5522.334	5554.924	5580.501	58.16698	2019-04-03 22:00:00	2019-04-03 13:00:00
Sensor1	2019-04-04	94	5375.500	5484.097	5543.167	167.66691	2019-04-04 18:00:00	2019-04-04 10:00:00
Sensor1	2019-04-05	95	5388.167	5401.479	5417.000	28.83339	2019-04-05 17:00:00	2019-04-05 09:00:00
Sensor1	2019-04-06	96	5397.667	5420.207	5446.167	48.50006	2019-04-06 00:00:00	2019-04-06 09:00:00
Sensor1	2019-04-07	97	5434.501	5446.493	5470.167	35.66647	2019-04-07 17:00:00	2019-04-07 11:00:00
Sensor1	2019-04-08	98	5408.167	5437.389	5471.167	62.99973	2019-04-08 19:00:00	2019-04-08 11:00:00
Sensor1	2019-04-09	99	5370.031	5419.137	5448.167	78.13583	2019-04-09 23:00:00	2019-04-09 09:00:00
Sensor1	2019-04-10	100	5353.692	5359.452	5371.129	17.43648	2019-04-10 10:00:00	2019-04-10 23:00:00

This approach can be applied for all sensors at the same time, however If "ALL" is specified, a single value will be calculated or extracted for all series in the data.frame..

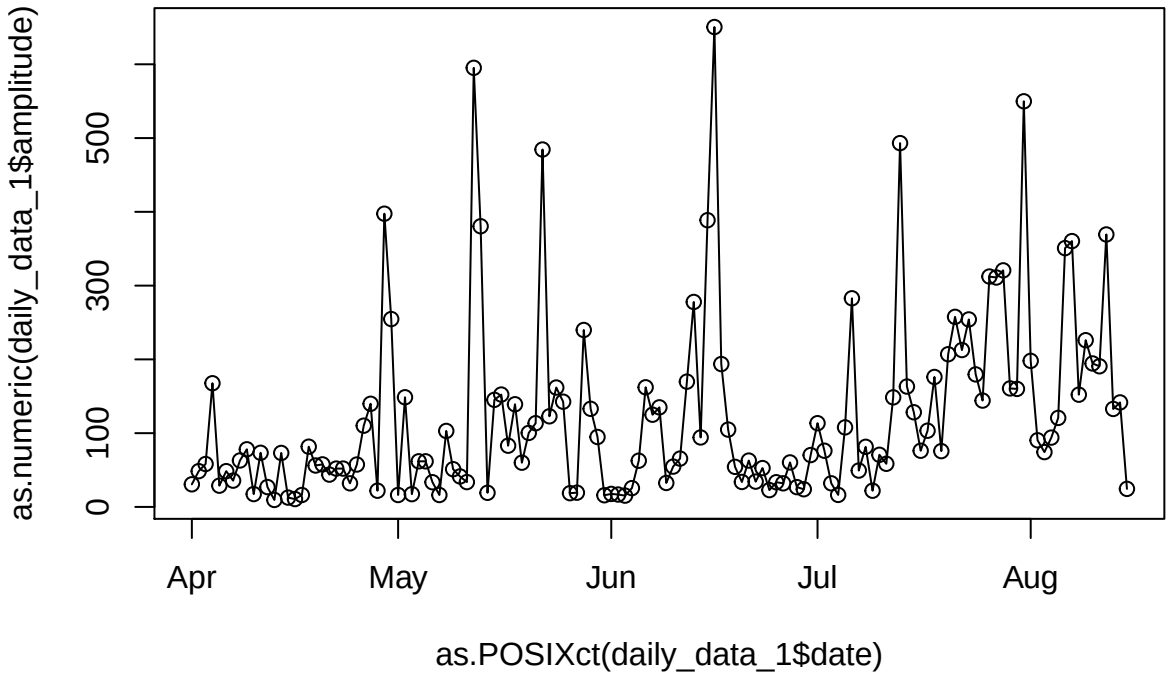
```
daily_data_ALL <- daily_stats(data_gf, sensor = "ALL", value = "max")
knitr::kable(
  daily_data_ALL[1:10,],
  caption = "Example of the daily approach for all sensors"
)
```

Table 3: Example of the daily approach for all sensors

	Sensor1	Sensor2
2019-04-01	5598.333	6111.167
2019-04-02	5612.500	6097.834
2019-04-03	5580.501	6090.167
2019-04-04	5543.167	5827.000

	Sensor1	Sensor2
2019-04-05	5417.000	5667.667
2019-04-06	5446.167	5670.834
2019-04-07	5470.167	5680.500
2019-04-08	5471.167	5674.334
2019-04-09	5448.167	5668.936
2019-04-10	5371.129	5691.901

```
plot(as.numeric(daily_data_1$amplitude)~as.POSIXct(daily_data_1$date),
     type = "o")
```



One interesting thing that can be calculated with this approach is the difference between the day "i" and day "i-1". For that, we used the package dplyr

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
daily_data_1$increment_between_days <-
  daily_data_1 %>% mutate(Diff = max - lag(max, 1))
daily_data_1$increment_between_days <-
  daily_data_1$increment_between_days$Diff
head(daily_data_1)
```

```
##      dmID       date DOY      min     mean      max amplitude
## 1 Sensor1 2019-04-01  91 5567.667 5583.870 5598.333  30.66635
## 2 Sensor1 2019-04-02  92 5564.000 5581.396 5612.500  48.50006
## 3 Sensor1 2019-04-03  93 5522.334 5554.924 5580.501  58.16698
## 4 Sensor1 2019-04-04  94 5375.500 5484.097 5543.167 167.66691
## 5 Sensor1 2019-04-05  95 5388.167 5401.479 5417.000  28.83339
## 6 Sensor1 2019-04-06  96 5397.667 5420.207 5446.167  48.50006
##              time_min            time_max increment_between_days
## 1 2019-04-01 02:00:00 2019-04-01 12:00:00                     NA
## 2 2019-04-02 23:00:00 2019-04-02 14:00:00               14.16683
## 3 2019-04-03 22:00:00 2019-04-03 13:00:00              -31.99959
## 4 2019-04-04 18:00:00 2019-04-04 10:00:00              -37.33349
## 5 2019-04-05 17:00:00 2019-04-05 09:00:00             -126.16682
## 6 2019-04-06 00:00:00 2019-04-06 09:00:00               29.16670
```

```
5580.50-5612.500
```

```
## [1] -32
```

Positive values shows increment between day i and i-1. These values are correlated with heavy rainy days or days when the trees have no good conditions for doing transpiration. Negative values are correlated with heigh temperatures, sunny days and there is available water in the soil. Thus, tree can do tranpiration and and there is a stem diameter shrinkage.

```
par(mar = c(5, 4, 4, 4) + 0.25)

plot(x =as.POSIXct(daily_data_1$date), y = as.numeric(daily_data_1$increment_between_days), type = "o",
     lwd = 1.5,
     main = "Daily approach",
     col="black",
     bty='l',
     ylim = c(-400,1000),
     xlim = as.POSIXct(c("2019-04-07","2019-08-15")),
     xlab="Time", ylab="",
     cex.axis=.75,)

library(greekLetters)
```

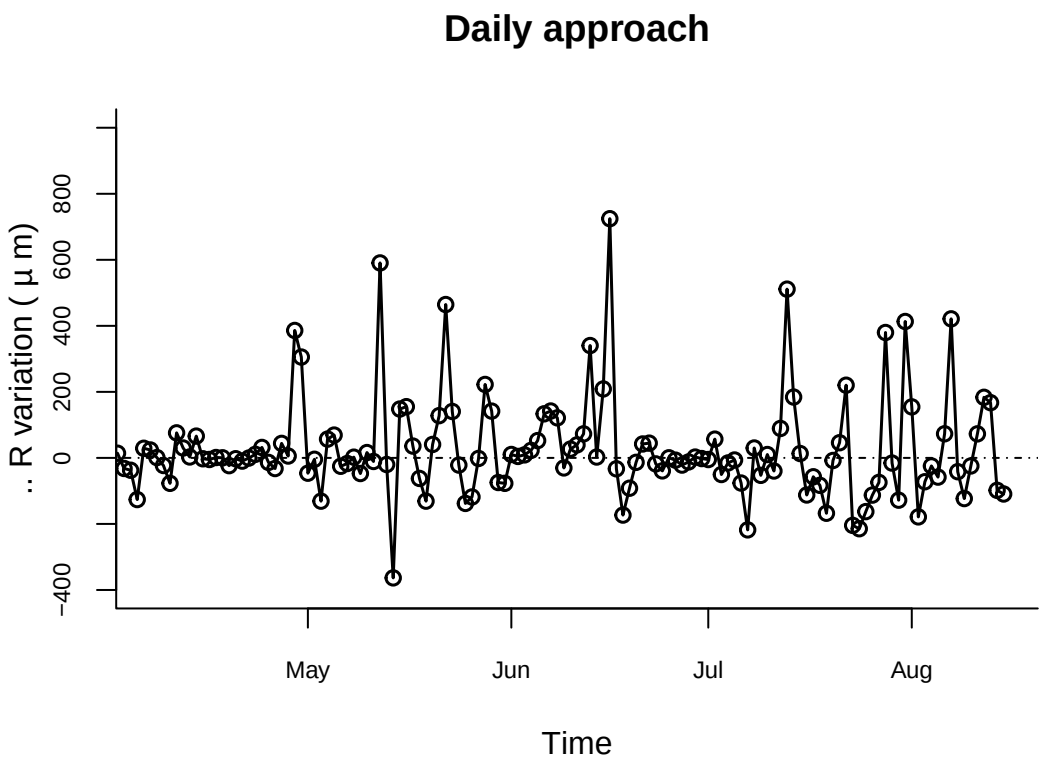
```
## Warning: package 'greekLetters' was built under R version 3.6.3
```

```
mtext(paste(greeks("Delta"),"R variation (",greeks("mu"), "m)"), side = 2,
      line = 2)
```

```
## Warning in mtext(paste(greeks("Delta"), "R variation (", greeks("mu"), "m)"), :
## conversion failure on 'Ī" R variation ( Ī¼ m)' in 'mbcsToSbcs': dot substituted
## for <ce>
```

```
## Warning in mtext(paste(greeks("Delta"), "R variation (", greeks("mu"), "m)"), :
## conversion failure on 'Ī" R variation ( Ī¼ m)' in 'mbcsToSbcs': dot substituted
## for <94>
```

```
abline(h = 0, lty = 4)
```



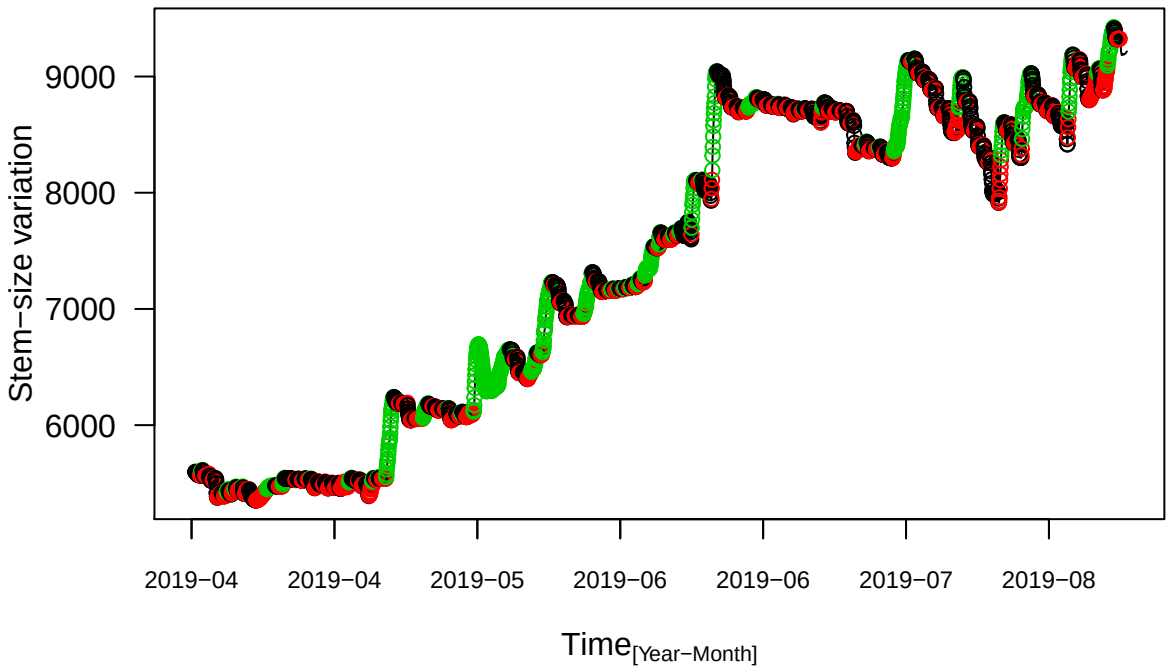
For the cycle approach, the dendrometeR's package calculates the circadian phases for each sensor.Phase 1 = contraction "or shrinkage", phase 2 = expansion, phase 3 = stem radius increment (SRI).

```
dm.phase_data <- phase_def(data_gf, radialIncrease = "max")
knitr::kable(
  dm.phase_data[1:20,],
  caption = "Example of the cycle approach"
)
```

Table 4: Example of the cycle approach

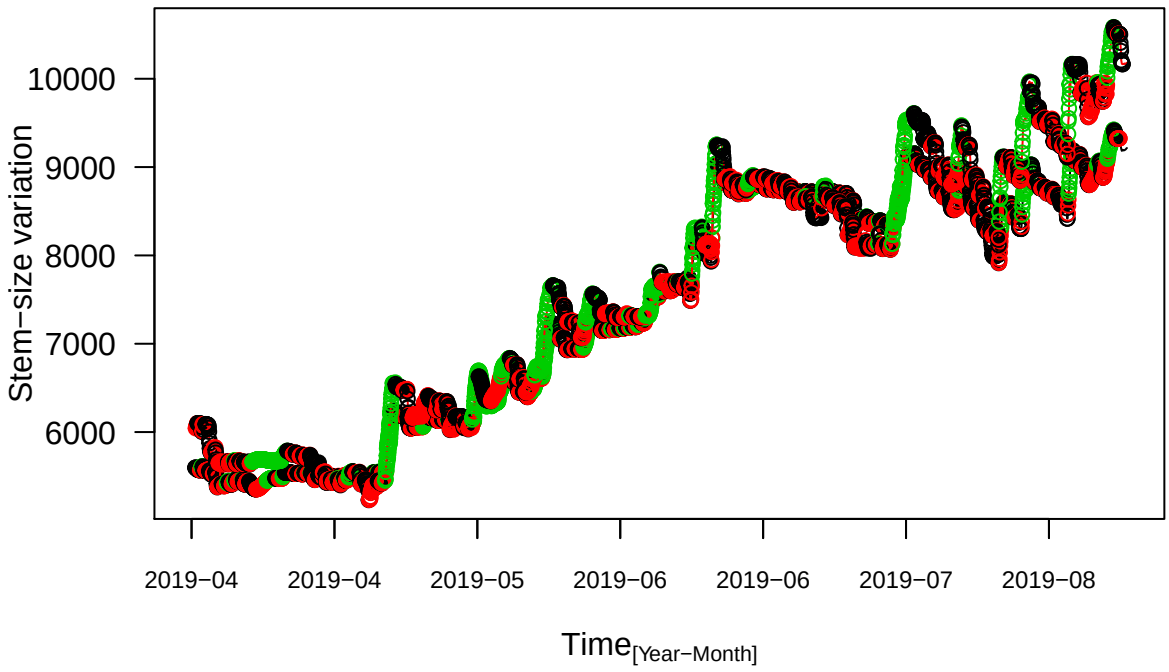
	Sensor1	Sensor2
2019-04-01 01:00:00	NA	NA
2019-04-01 02:00:00	NA	NA
2019-04-01 03:00:00	NA	NA
2019-04-01 04:00:00	NA	NA
2019-04-01 05:00:00	NA	NA
2019-04-01 06:00:00	NA	NA
2019-04-01 07:00:00	NA	NA
2019-04-01 08:00:00	NA	NA
2019-04-01 09:00:00	NA	NA
2019-04-01 10:00:00	NA	NA
2019-04-01 11:00:00	NA	NA
2019-04-01 12:00:00	NA	NA
2019-04-01 13:00:00	1	NA
2019-04-01 14:00:00	1	NA
2019-04-01 15:00:00	1	2
2019-04-01 16:00:00	1	2
2019-04-01 17:00:00	1	2
2019-04-01 18:00:00	1	2
2019-04-01 19:00:00	1	2
2019-04-01 20:00:00	1	2

```
phase_plot(data_gf, dm.phase_data, sensor = 1)
```



Phase 1 “contraction” is marked with black colour, phase 2 “expansion” with red colour and phase 3 “SRI” with green colour.

```
phase_plot(data_gf, dm.phase_data)
```



The function cycle_stats calculates... Unfortunatelly, it must be executed sensor by sensor, there is no way to do all the sensors at once.

```
cycle_stats_data <- cycle_stats(data_gf, dm.phase_data, sensor = 1)
cycle_stats_data$cycleStats[1:20,]
```

```
##       dmID cycle phase               begin                 end duration_h
## 1  Sensor1     1     1 2019-04-01 13:00:00 2019-04-02 05:00:00   16 hours
## 2  Sensor1     1     2 2019-04-02 05:00:00 2019-04-02 10:00:00    5 hours
## 3  Sensor1     1     3 2019-04-02 10:00:00 2019-04-02 14:00:00    4 hours
## 4  Sensor1     1     4 2019-04-01 13:00:00 2019-04-02 14:00:00   25 hours
## 5  Sensor1     2     1 2019-04-02 14:00:00 2019-04-03 00:00:00   10 hours
## 6  Sensor1     2     2 2019-04-03 00:00:00 2019-04-03 13:00:00   13 hours
## 7  Sensor1     2     4 2019-04-02 14:00:00 2019-04-03 13:00:00   23 hours
## 8  Sensor1     3     1 2019-04-03 13:00:00 2019-04-04 02:00:00   13 hours
## 9  Sensor1     3     2 2019-04-04 02:00:00 2019-04-04 10:00:00    8 hours
## 10 Sensor1     3     4 2019-04-03 13:00:00 2019-04-04 10:00:00   21 hours
## 11 Sensor1     4     1 2019-04-04 10:00:00 2019-04-04 18:00:00    8 hours
## 12 Sensor1     4     2 2019-04-04 18:00:00 2019-04-05 09:00:00   15 hours
## 13 Sensor1     4     4 2019-04-04 10:00:00 2019-04-05 09:00:00   23 hours
```

```
## 14 Sensor1     5     1 2019-04-05 09:00:00 2019-04-05 17:00:00    8 hours
## 15 Sensor1     5     2 2019-04-05 17:00:00 2019-04-06 03:00:00   10 hours
## 16 Sensor1     5     3 2019-04-06 03:00:00 2019-04-06 09:00:00    6 hours
## 17 Sensor1     5     4 2019-04-05 09:00:00 2019-04-06 09:00:00   24 hours
## 18 Sensor1     6     1 2019-04-06 09:00:00 2019-04-06 20:00:00   11 hours
## 19 Sensor1     6     2 2019-04-06 20:00:00 2019-04-07 06:00:00   10 hours
## 20 Sensor1     6     3 2019-04-07 06:00:00 2019-04-07 11:00:00    5 hours
##    duration_m  magnitude      min      max
## 1    960 mins  29.000282 5567.500 5596.500
## 2    300 mins  29.333115 5568.667 5598.000
## 3    240 mins   6.999969 5605.500 5612.500
## 4   1500 mins  45.000076 5567.500 5612.500
## 5    600 mins  46.666622 5563.667 5610.333
## 6    780 mins  16.833782 5563.667 5580.501
## 7   1380 mins  46.666622 5563.667 5610.333
## 8    780 mins  42.333603 5521.833 5564.167
## 9    480 mins  17.833710 5525.333 5543.167
## 10  1260 mins  42.333603 5521.833 5564.167
## 11   480 mins 167.333126 5375.500 5542.833
## 12   900 mins  38.166523 5378.834 5417.000
## 13  1380 mins 167.333126 5375.500 5542.833
## 14   480 mins  26.166439 5388.167 5414.333
## 15   600 mins  23.499966 5388.334 5411.834
## 16   360 mins  24.000168 5422.167 5446.167
## 17  1440 mins  58.000088 5388.167 5446.167
## 18   660 mins  31.499863 5406.167 5437.667
## 19   600 mins  34.166813 5410.833 5445.000
## 20   300 mins  22.166729 5448.000 5470.167
```

```
knitr::kable(
  cycle_stats_data$cycle.df[1:20,],
  caption = "Example of the cycle stats"
)
```

Table 5: Example of the cycle stats

	dmID	cycle	phase
2019-04-01 13:00:00	Sensor1	0	1
2019-04-01 14:00:00	Sensor1	1	1
2019-04-01 15:00:00	Sensor1	1	1
2019-04-01 16:00:00	Sensor1	1	1
2019-04-01 17:00:00	Sensor1	1	1
2019-04-01 18:00:00	Sensor1	1	1
2019-04-01 19:00:00	Sensor1	1	1
2019-04-01 20:00:00	Sensor1	1	1
2019-04-01 21:00:00	Sensor1	1	1
2019-04-01 22:00:00	Sensor1	1	1
2019-04-01 23:00:00	Sensor1	1	1
2019-04-02 00:00:00	Sensor1	1	1
2019-04-02 01:00:00	Sensor1	1	1
2019-04-02 02:00:00	Sensor1	1	1
2019-04-02 03:00:00	Sensor1	1	1

	dmID	cycle	phase
2019-04-02 04:00:00	Sensor1	1	1
2019-04-02 05:00:00	Sensor1	1	1
2019-04-02 06:00:00	Sensor1	1	2
2019-04-02 07:00:00	Sensor1	1	2
2019-04-02 08:00:00	Sensor1	1	2

APPENDIX 2: Generalized Linear Model for Binomial Distributions

Jorge Palmero-Barrachina

18/3/2022

R Markdown

This is an R Markdown document. Markdown is a simple extension of R software to print the code and results that the author obtained. More information in http://rmarkdown.rstudio.com.

```
library(readxl)
data <- read_excel("D:/Results_cycle_stats/cycle_stats_attacked_vs_healthy/Nela/Good/mixed1_225_244.xlsx",
                           col_types = c("text", "numeric", "text",
                                         "numeric", "numeric", "numeric",
                                         "numeric", "numeric", "numeric",
                                         "numeric", "text", "numeric"))

attach(data)
```

```
head(data)
```

```
## # A tibble: 6 x 12
##   dmID  cycle phase  begin    end duration_h duration_m magnitude   min   max
##   <chr> <dbl> <chr>  <dbl>  <dbl>      <dbl>      <dbl>     <dbl> <dbl> <dbl>
## 1 T4C1      9 1     43325. 43325.          7        420   0.0155   4.07  4.08
## 2 T4C1      9 2     43325. 43326.         18       1080   0.00933  4.07  4.08
## 3 T4C1      9 4     43325. 43326.         25       1500   0.0155   4.07  4.08
## 4 T4C1     10 1     43326. 43326.          5        300   0.123    3.95  4.07
## 5 T4C1     10 2     43326. 43327.         18       1080   0.0290   3.95  3.98
## 6 T4C1     10 4     43326. 43327.         23       1380   0.123    3.95  4.07
## # ... with 2 more variables: Period <chr>, Health_status <dbl>
```

```
summary(data)
```

```
##      dmID               cycle          phase               begin
##  Length:834         Min.   : 1.00   Length:834         Min.   :43323
##  Class :character   1st Qu.: 8.00   Class :character   1st Qu.:43342
##  Mode  :character   Median :14.00   Mode  :character   Median :43969
##                     Mean   :13.46                      Mean   :43801
##                     3rd Qu.:19.00                      3rd Qu.:43990
##                     Max.   :29.00                      Max.   :44015
##       end          duration_h       duration_m     magnitude
##  Min.   :43325   Min.   :  1.00   Min.   :  60   Min.   :0.00000
```

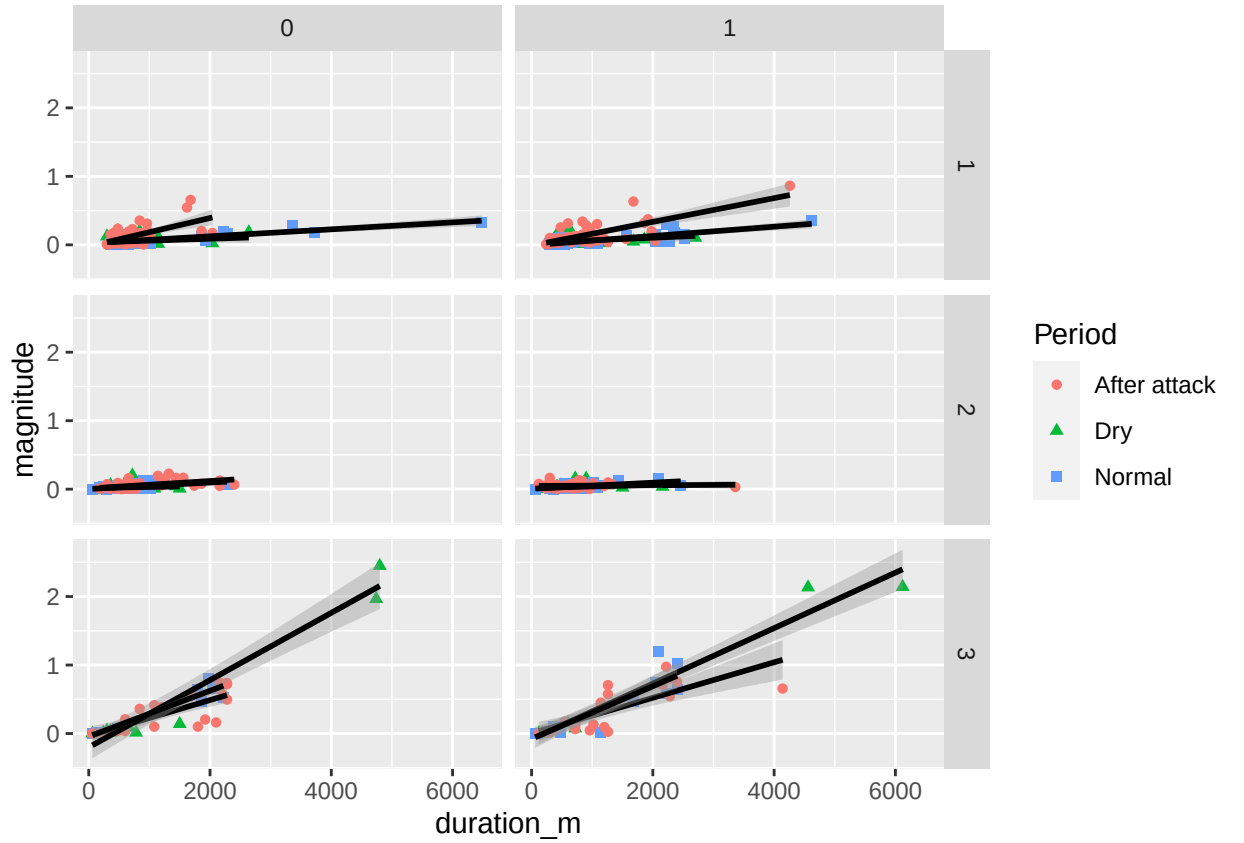
```
##  1st Qu.:43343   1st Qu.:  9.00   1st Qu.: 540   1st Qu.:0.02150
##  Median :43969   Median : 15.00   Median : 900   Median :0.05425
##  Mean   :43802   Mean   : 20.53   Mean   :1232   Mean   :0.14619
##  3rd Qu.:43991   3rd Qu.: 25.00   3rd Qu.:1500   3rd Qu.:0.13133
##  Max.   :44016   Max.   :129.00   Max.   :7740   Max.   :2.47500
##       min              max           Period          Health_status
##  Min.   : 1.622   Min.   : 1.678   Length:834         Min.   :0.0000
##  1st Qu.: 5.329   1st Qu.: 5.490   Class :character   1st Qu.:0.0000
##  Median : 9.942   Median :10.099   Mode  :character   Median :0.0000
##  Mean   : 8.963   Mean   : 9.110                      Mean   :0.4904
##  3rd Qu.:11.959   3rd Qu.:12.052                      3rd Qu.:1.0000
##  Max.   :14.712   Max.   :14.908                      Max.   :1.0000
```

To take a quick look into the linear models for different phases of circadian cycle between healthy and attacked trees we did the next plot. Column named "0" are healthy trees. Column named "1", are attacked trees. Row named "1" (on the right) is phase of contraction. Row named "2" is expansion phase. Row named "3" is SRI.

```
healthy<-subset(data, Health_status=="0")
attack<-subset(data, Health_status=="1")

phases<-subset(data, phase!="4")
phases$phase<-as.character(phases$phase)
phases$Health_status<-as.character(phases$Health_status)

library(ggplot2)
library(MASS)
ggplot(phases,aes(x=duration_m,y=magnitude,color=Period, shape=Period))+
  geom_point()+
  facet_grid(phase~Health_status)+
  geom_smooth(method=lm, color="black")
```



We created the glm for negative distributions by using the next R code. We filtered the dry period Dry (DOY 225-244, Year 2018)

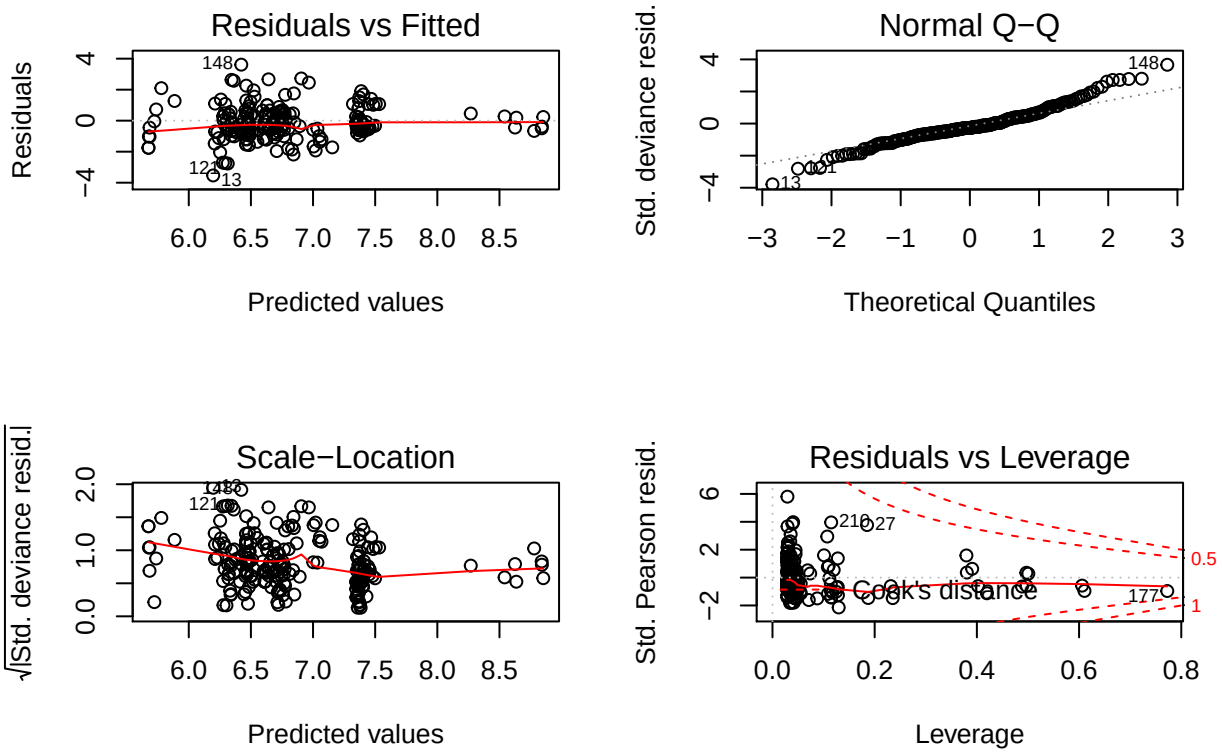
```
summary(m_dry <- glm.nb(duration_m ~ magnitude*Health_status*phase,
                        data = data[which(data$Period == "Dry"),]))
```

```
##
## Call:
## glm.nb(formula = duration_m ~ magnitude * Health_status * phase,
##     data = data[which(data$Period == "Dry"), ], init.theta = 5.234862049,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5328  -0.6572  -0.2644   0.3747   3.6120
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     6.59418    0.10670  61.799  < 2e-16 ***
## magnitude                       2.02368    1.36042   1.488   0.1369
## Health_status                   0.07264    0.14936   0.486   0.6267
## phase2                         -0.14756    0.14173  -1.041   0.2978
## phase3                         -0.39751    0.19081  -2.083   0.0372 *
## phase4                          0.76304    0.13292   5.741 9.42e-09 ***
## magnitude:Health_status         0.26147    1.78948   0.146   0.8838
## magnitude:phase2               -0.25836    2.54275  -0.102   0.9191
```

```
## magnitude:phase3               -0.96876    1.36963  -0.707   0.4794
## magnitude:phase4               -1.42404    1.36820  -1.041   0.2980
## Health_status:phase2           -0.28314    0.20220  -1.400   0.1614
## Health_status:phase3           -0.59377    0.26397  -2.249   0.0245 *
## Health_status:phase4           -0.09682    0.18694  -0.518   0.6045
## magnitude:Health_status:phase2  3.15037    3.53509   0.891   0.3728
## magnitude:Health_status:phase3  0.06593    1.80384   0.037   0.9708
## magnitude:Health_status:phase4 -0.16681    1.80168  -0.093   0.9262
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(5.2349) family taken to be 1)
##
##     Null deviance: 706.06  on 230  degrees of freedom
## Residual deviance: 238.71  on 215  degrees of freedom
## AIC: 3444.2
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  5.235
##           Std. Err.:  0.477
##
##  2 x log-likelihood:  -3410.190
```

Diagnostic plots of this model can be seen in the next graph

```
par(mfrow=c(2,2))
plot(m_dry)
```



The likelihood ratio for this glm can be seen in the results of the thesis.

We calculated the same model but for the period one month before the attack (DOY 117-147, Year 2020). This period was called in our R code as "normal".

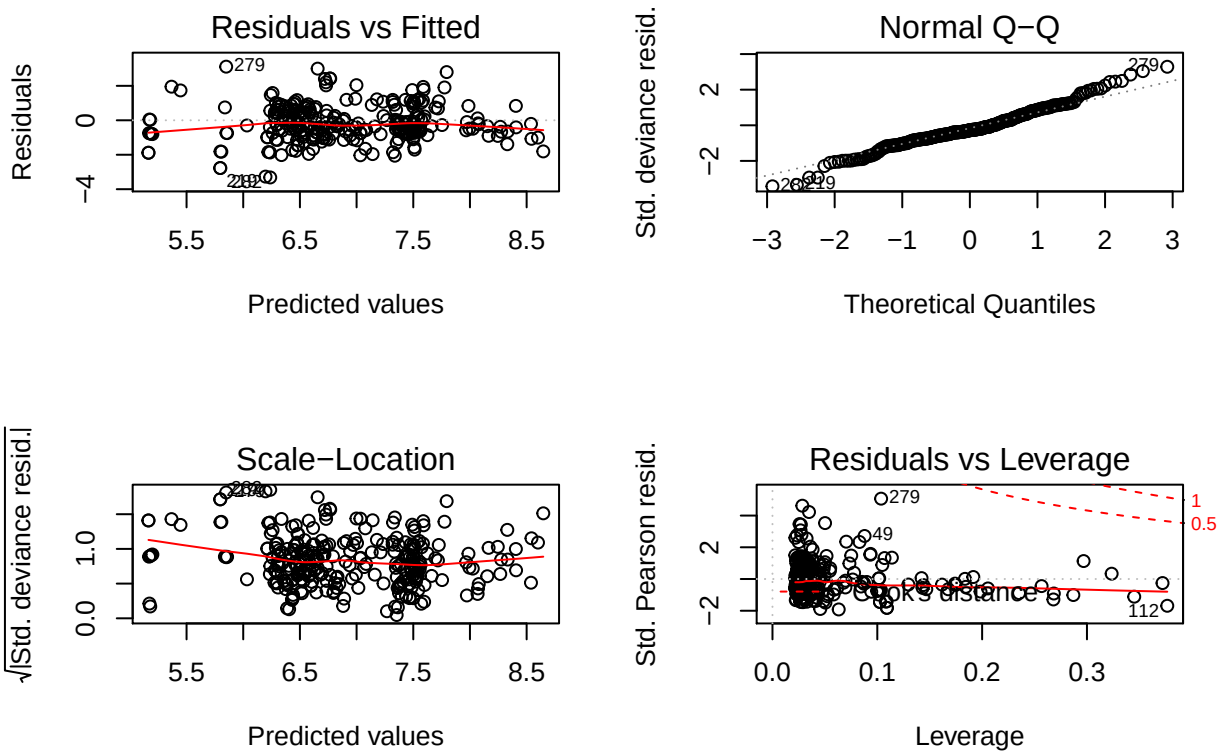
```
summary(m_normal <- glm.nb(duration_m ~ magnitude*Health_status*phase,
                           data = data[which(data$Period == "Normal"),]))
```

```
##
## Call:
## glm.nb(formula = duration_m ~ magnitude * Health_status * phase,
##     data = data[which(data$Period == "Normal"), ], init.theta = 4.463089788,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3174  -0.7388  -0.2756   0.4309   3.1121
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     6.35345    0.10345  61.413  < 2e-16 ***
## magnitude                       6.15522    0.97084   6.340 2.30e-10 ***
## Health_status                   0.11167    0.14293   0.781   0.4346
## phase2                         -0.15684    0.14496  -1.082   0.2793
## phase3                         -1.18940    0.18510  -6.426 1.31e-10 ***
## phase4                          0.97755    0.13638   7.168 7.63e-13 ***
## magnitude:Health_status        -0.37436    1.36264  -0.275   0.7835
```

```
## magnitude:phase2                1.80720    2.34224   0.772   0.4404
## magnitude:phase3               -2.22778    1.04463  -2.133   0.0330 *
## magnitude:phase4               -4.76422    1.02457  -4.650 3.32e-06 ***
## Health_status:phase2           -0.07010    0.21219  -0.330   0.7411
## Health_status:phase3            0.52172    0.26225   1.989   0.0467 *
## Health_status:phase4            0.01024    0.19107   0.054   0.9573
## magnitude:Health_status:phase2  0.86459    3.41454   0.253   0.8001
## magnitude:Health_status:phase3 -1.16161    1.44838  -0.802   0.4226
## magnitude:Health_status:phase4 -0.11698    1.42613  -0.082   0.9346
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(4.4631) family taken to be 1)
##
##     Null deviance: 836.95  on 285  degrees of freedom
## Residual deviance: 297.14  on 270  degrees of freedom
## AIC: 4334.3
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  4.463
##           Std. Err.:  0.363
##
##  2 x log-likelihood:  -4300.251
```

Diagnostic plots of this model can be seen in the next graph

```
par(mfrow=c(2,2))
plot(m_normal)
```



The likelihood ratio for this glm can be seen in the results of the thesis.

We calculated the same model but for the period one month after the attack (DOY 150-185, Year 2020). This period was called in our R code as "After attack".

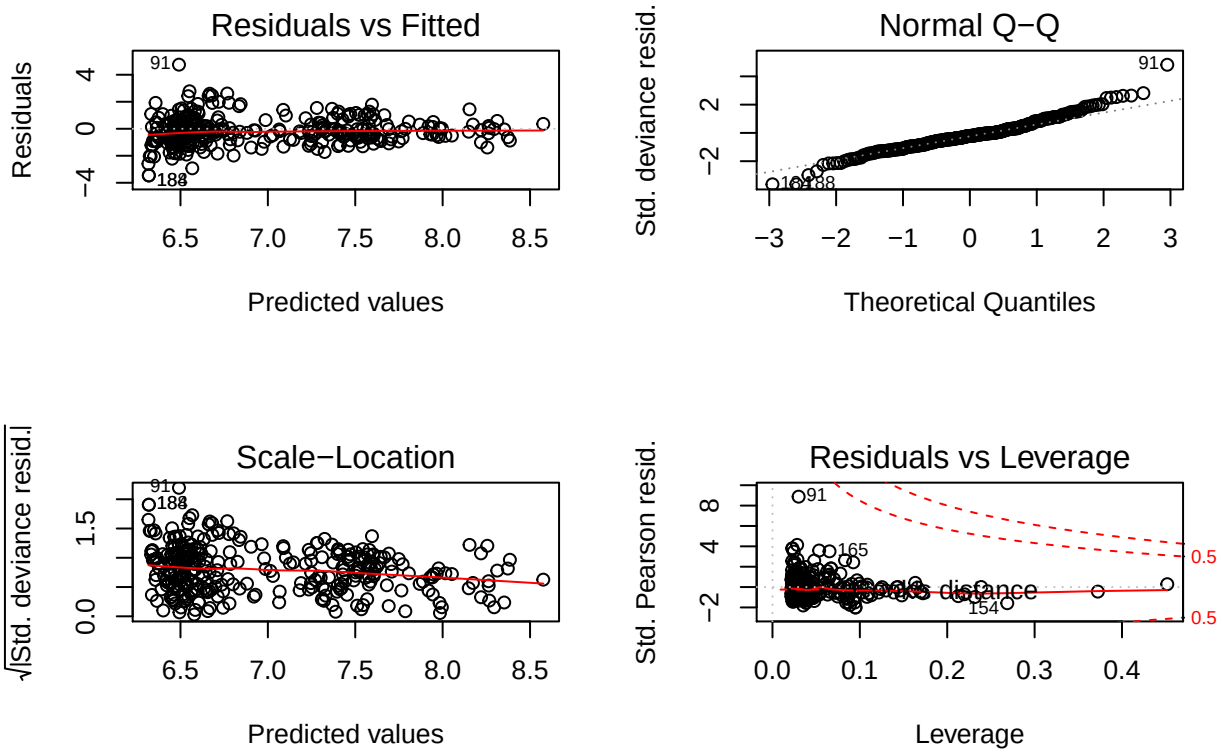
```
summary(m_after <- glm.nb(duration_m ~ magnitude*Health_status*phase,
                          data = data[which(data$Period == "After attack"),]))
```

```
##
## Call:
## glm.nb(formula = duration_m ~ magnitude * Health_status * phase,
##     data = data[which(data$Period == "After attack"), ], init.theta = 4.588469325,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4582  -0.7881  -0.2100   0.3082   4.7468
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    6.321806   0.098873  63.939  < 2e-16 ***
## magnitude                      1.956998   0.527976   3.707  0.00021 ***
## Health_status                  0.097688   0.136376   0.716  0.47380
## phase2                         0.084846   0.139705   0.607  0.54364
## phase3                        -0.002874   0.175079  -0.016  0.98690
## phase4                         1.071683   0.142672   7.512 5.84e-14 ***
## magnitude:Health_status        0.180379   0.679492   0.265  0.79065
```

```
## magnitude:phase2                3.499967   1.315186   2.661  0.00779 **
## magnitude:phase3                0.282326   0.644758   0.438  0.66147
## magnitude:phase4               -0.952344   0.590661  -1.612  0.10689
## Health_status:phase2           -0.063074   0.209571  -0.301  0.76344
## Health_status:phase3           -0.109304   0.243492  -0.449  0.65350
## Health_status:phase4           -0.290591   0.198445  -1.464  0.14310
## magnitude:Health_status:phase2 -4.001182   2.457523  -1.628  0.10350
## magnitude:Health_status:phase3 -0.458353   0.834174  -0.549  0.58268
## magnitude:Health_status:phase4  0.103759   0.765865   0.135  0.89223
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(4.5885) family taken to be 1)
##
##     Null deviance: 832.46  on 316  degrees of freedom
## Residual deviance: 328.82  on 301  degrees of freedom
## AIC: 4825.1
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  4.588
##           Std. Err.:  0.355
##
##  2 x log-likelihood:  -4791.125
```

Diagnostic plots of this model can be seen in the next graph

```
par(mfrow=c(2,2))
plot(m_after)
```



The likelihood ratio for this glm can be seen in the results of the thesis.