Test PMLSeg

Ninh

2024-08-09

Introduction

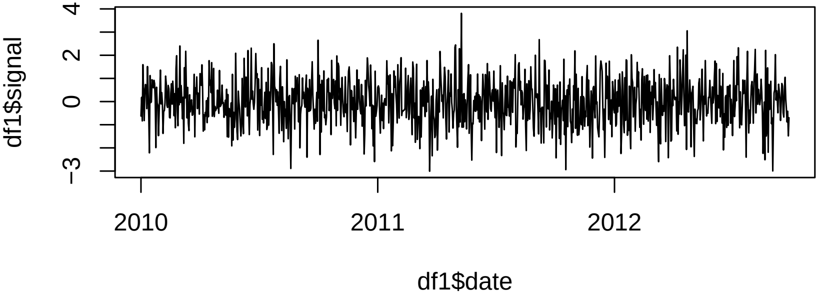
This document presents results from several tests of the `PMLSeg` package, including:

- Test of the `Segmentation` function with and without offsets, considering the following examples:
  - Ex1 : zero mean + IID noise
  - Ex2 : periodic mean + IID noise
  - Ex3 : periodic mean + monthly variance
- Tests of other functions such as:
  - `PlotSeg` to visualize segmentation results
  - `Cluster_screening` to detect groups of close change-points (usually due to outliers) and determine whether to keep or remove the cluster. The cluster in this document is determined by a group of close change-points within 80 days (this threshold is changeable).
  - `Validation` to validate the detected change-points with the help of metadata.

Generate Example Data

Ex1 time series

```
set.seed(1)
length_series = 1000
df1 = data.frame(date = seq.Date(from = as.Date("2010-01-01"),
                                to = as.Date("2010-01-01")+(length_series-1),
                                by = "day"),
                signal = rnorm(n = length_series, mean = 0, sd = 1))

plot(df1$date, df1$signal, type = "l")
```

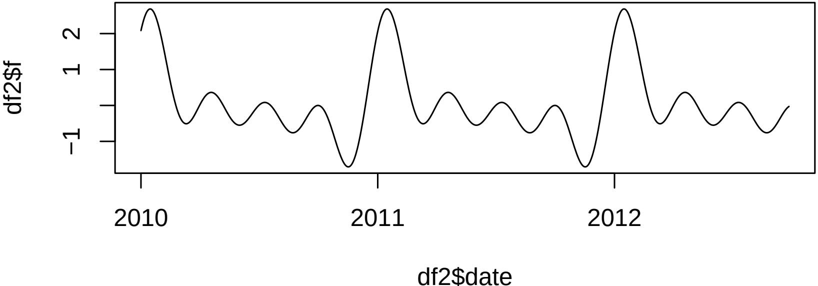


```
head(df1, 3)
#>         date     signal
#> 1 2010-01-01 -0.6264538
#> 2 2010-01-02  0.1836433
#> 3 2010-01-03 -0.8356286
```

Generate Example Data

Ex2 time series : Add the functional with 4 Fourier series components, each with a coefficient of 0.5

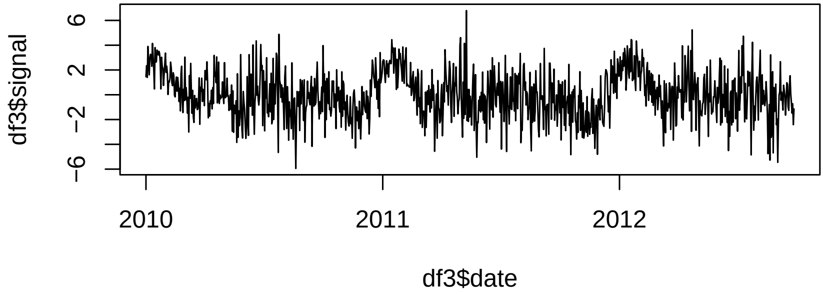
```
library(dplyr)

T <- 365.25
df2 <- df1 %>%
  mutate(t = as.numeric(date - date[1])+1,
  f = rowSums(sapply(1:4, function(i) 0.5*cos(i*t*(2*pi)/T) + 0.5*sin(i*t*(2*pi)/T))),
  signal = signal + f)

head(df2, 3)
#>         date   signal t        f
#> 1 2010-01-01 1.457297 1 2.083751
#> 2 2010-01-02 2.346461 2 2.162818
#> 3 2010-01-03 1.401342 3 2.236970
plot(df2$date, df2$f, type = "l")
```



Generate Example Data

Ex3 time series : Apply the same functional as Ex2, adjusted for monthly variance

```r
std = c(1, 1.2, 1.4, 1.6, 1.8, 2, 2, 1.8, 1.6, 1.5, 1.4, 1.3)

df3 <- df1 %>%
  mutate(sd = std[as.numeric(format(date, "%m"))],
         signal = signal * sd ) %>%
  mutate(signal = signal + df2$f)

head(df3, 3)
#>         date   signal sd
#> 1 2010-01-01 1.457297  1
#> 2 2010-01-02 2.346461  1
#> 3 2010-01-03 1.401342  1
plot(df3$date, df3$signal, type = "l")
```



Generate Example Data

Harmonize Formats of 3 Dataframes for Testing

```r
df2 <- df2 %>% select(date, signal)
df3 <- df3 %>% select(date, signal)

names(df1)
#> [1] "date"   "signal"
names(df2)
#> [1] "date"   "signal"
names(df3)
#> [1] "date"   "signal"
```

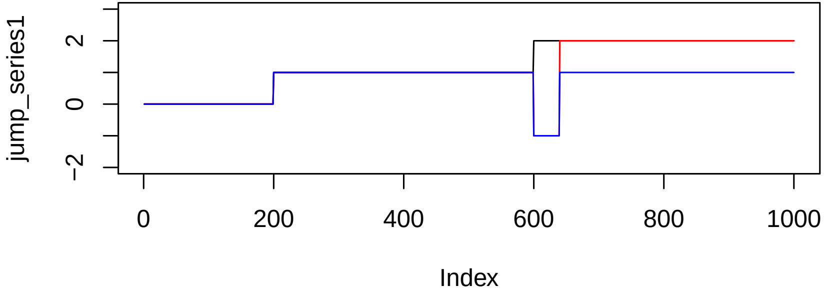
Preliminary Settings

Generate different offset series to add to the original series :

```r
# Function to generate jump series
generate_jump_series <- function(jump_indices, jump_amp, length_series) {
  jump_series <- rep(0, length_series)
  jump_indices <- c(1, jump_indices, length_series + 1)

  changes <- rep(0, length_series)
  changes[jump_indices[-length(jump_indices)]] <- jump_amp

  jump_series <- cumsum(changes)

  return(jump_series)
}
# No cluster
jump_ind1 <- c(200, 600)
jump_amp1 <- c(0, 1, 1)
# One cluster formed by the second and third change-points, which need to be
# replaced by the middle point
jump_ind2 <- c(200, 600, 640)
jump_amp2 <- c(0, 1, -2, 3)
# One cluster formed by the second and third change-points, which need to be removed
jump_amp3 <- c(0, 1, -2, 2)
```

Premilinary setting

```r
# Generate jump series
jump_series1 <- generate_jump_series(jump_ind1, jump_amp1, length_series)
jump_series2 <- generate_jump_series(jump_ind2, jump_amp2, length_series)
jump_series3 <- generate_jump_series(jump_ind2, jump_amp3, length_series)
# Visualize jump series
plot(jump_series1, type = "l", ylim = c(-2,3))
lines(jump_series2, col = "red")
lines(jump_series3, col = "blue")
```



Premilinary setting

```
# Create an example metadata
meta = data.frame(date = c(df1$date[jump_ind2], df1$date[400]),
                  type = c("type1", "type2", "type3", "type3"))
meta
#>         date  type
#> 1 2010-07-19 type1
#> 2 2011-08-23 type2
#> 3 2011-10-02 type3
#> 4 2011-02-04 type3
```

Testing the Segmentation Function

When the series is homogeneous

```r
library(PMLseg)
# Ex1
seg1a = Segmentation(OneSeries = df1, FunctPart = FALSE)
str(seg1a)
#> List of 5
#>  $ Tmu     :'data.frame':    1 obs. of  5 variables:
#>   ..$ begin: int 1
#>   ..$ end  : int 1000
#>   ..$ mean : num -0.00423
#>   ..$ se   : num 29.7
#>   ..$ np   : num 1000
#>  $ FitF    : logi FALSE
#>  $ CoeffF  : logi FALSE
#>  $ MonthVar: num [1:12] 1.089 0.887 1.334 1.092 1.21 ...
#>  $ SSR     : num 933
seg1a$Tmu
#>   begin  end         mean       se   np
#> 1     1 1000 -0.004229094 29.73496 1000
```

No change-points are detected, as shown in the `Tmu` dataframe, which lists all segments of the series. The mean is close to 0. Additionally, the results include not only the `Tmu` dataframe but also other outputs such as the fitted functional part `FitF` (which is not fitted when `FunctPart` is set to FALSE), coefficients of functional elements `CoeefF`, monthly variance `MonthVar`, and the Sum of Squares of Residuals `SSR`.

Test the Segmentation Function

When series is homogeneous

```
# Ex2
seg2a = Segmentation(OneSeries = df2, FunctPart = TRUE)
seg2a$Tmu
#>   begin  end         mean       se   np
#> 1     1 1000 -0.007048795 29.70068 1000
# Ex3
seg3a = Segmentation(OneSeries = df3, FunctPart = TRUE)
seg3a$Tmu
#>   begin  end        mean       se   np
#> 1     1 1000 -0.01249977 20.49008 1000
```

No change-points are detected in these two examples either.

Test the Segmentation Function

Adding Jump Series to the original Series and running the segmentation

```
# Ex1
df1b <- df1 %>% mutate(signal = signal + jump_series1)
seg1b = Segmentation(OneSeries = df1b, FunctPart = FALSE)
seg1b$Tmu
#>   begin  end       mean       se  np
#> 1     1  199 0.04231954 13.32685 199
#> 2   200  598 0.99732352 18.65043 399
#> 3   599 1000 1.96897996 18.96817 402

# Ex2
df2b <- df2 %>% mutate(signal = signal + jump_series1)
seg2b = Segmentation(OneSeries = df2b, FunctPart = TRUE)
seg2b$Tmu
#>   begin  end       mean       se  np
#> 1     1  200 0.02694284 13.37988 200
#> 2   201  598 0.99309421 18.60675 398
#> 3   599 1000 1.97809762 18.93598 402

# Ex3
df3b <- df3 %>% mutate(signal = signal + jump_series1)
seg3b = Segmentation(OneSeries = df3b, FunctPart = TRUE)
seg3b$Tmu
#>   begin  end      mean        se  np
#> 1     1  233 0.1119287  9.998523 233
#> 2   234  601 1.0163802 12.372062 368
#> 3   602 1000 1.9701980 12.920066 399
```

Test the Segmentation Function

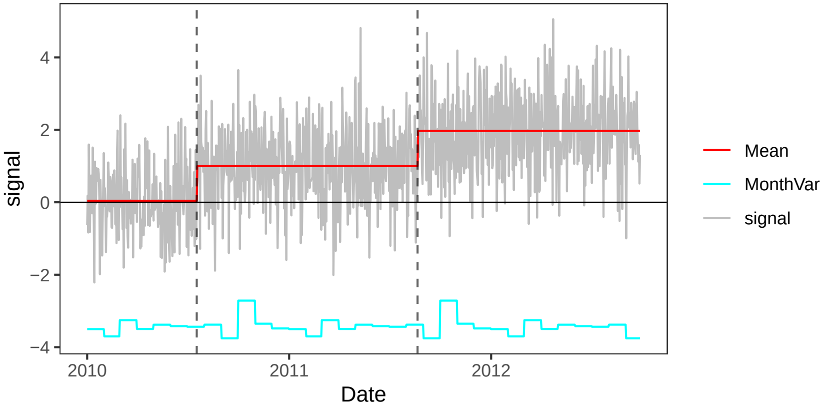
Adding other jump series in the most complicated series (Ex3)

```
df3c <- df3 %>% mutate(signal = signal + jump_series2)
seg3c = Segmentation(OneSeries = df3c, FunctPart = TRUE)
seg3c$Tmu
#>   begin  end       mean        se  np
#> 1     1  233  0.1077994  9.989649 233
#> 2   234  593  1.0273785 12.267514 360
#> 3   594  639 -0.8757300  4.221990  46
#> 4   640 1000  1.9620110 12.264455 361

df3d <- df3 %>% mutate(signal = signal + jump_series3)
seg3d = Segmentation(OneSeries = df3d, FunctPart = TRUE)
seg3d$Tmu
#>   begin  end        mean        se  np
#> 1     1  200  0.04790145  9.596181 200
#> 2   201  599  0.99064613 12.641622 399
#> 3   600  641 -0.92149852  4.101278  42
#> 4   642 1000  0.95430531 12.251166 359
```

Test the `PlotSeg` Function

Visualize the segmentation result of example 1 with jump series 1

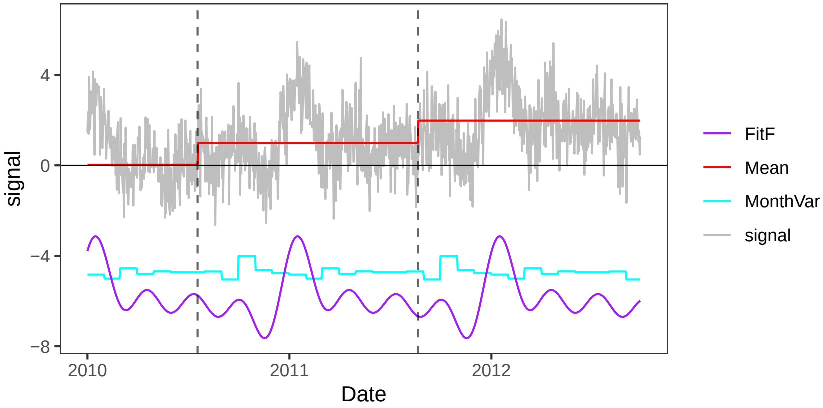
```
PlotSeg(OneSeries = df1b, SegRes = seg1b, FunctPart = FALSE)
```



Test the `PlotSeg` Function

Visualize the segmentation result of example 2 with jump series 1

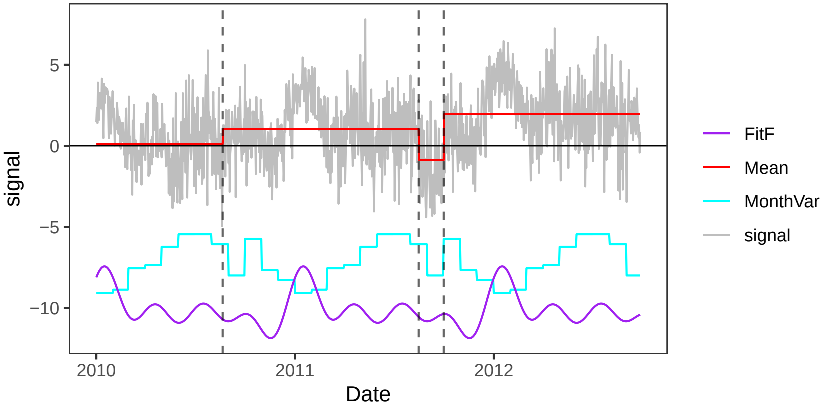
```
PlotSeg(OneSeries = df2b, SegRes = seg2b, FunctPart = TRUE)
```



Test the `PlotSeg` Function

Visualize the segmentation result of example 3 with with jump series 2

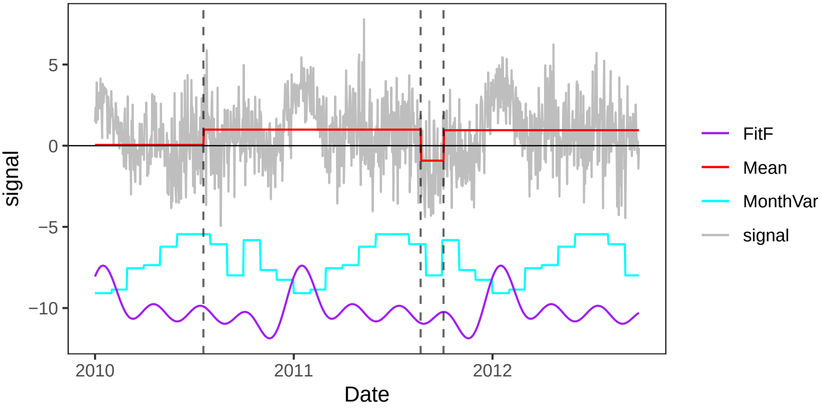
```
PlotSeg(OneSeries = df3c, SegRes = seg3c, FunctPart = TRUE)
```



Test the `PlotSeg` Function

Visualize the segmentation result of example 3 with with jump series 3

```r
PlotSeg(OneSeries = df3d, SegRes = seg3d, FunctPart = TRUE)
```



Test the Cluster_screening Function

Segmentation sometime detects the cluster like in example 3c, 3d. When it is too close and after cluster, the change in mean is not significant, we will remove such kind of cluster (screening2). If it is significant, we will replace a cluster by the middle point (screening1) and recommend to remove the data within the cluster.

```
screening1 = Cluster_screening(Tmu = seg3c$Tmu,
                              MaxDist = 80)
str(screening1)
#> List of 3
#>  $ UpdatedCP : num [1:3] 233 617 1000
#>  $ RemoveData:'data.frame':  1 obs. of  2 variables:
#>   ..$ begin: int 594
#>   ..$ end  : int 639
#>  $ ChangeCP  : chr "Yes"

screening2 = Cluster_screening(Tmu = seg3d$Tmu,
                              MaxDist = 80)
str(screening2)
#> List of 3
#>  $ UpdatedCP : int [1:2] 200 1000
#>  $ RemoveData:'data.frame':  1 obs. of  2 variables:
#>   ..$ begin: int 600
#>   ..$ end  : int 641
#>  $ ChangeCP  : chr "Yes"
```

Test the `UpdatedParametersForFixedCP` function

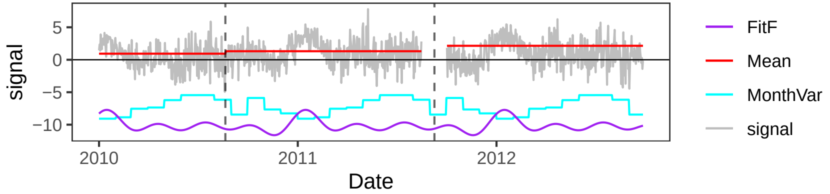
When the screening say it is needed to replace the changepoint, we need to reestimate the mean and other parameters in the `Tmu` dataframe.

```
seg_upd = UpdatedParametersForFixedCP(OneSeries = df3c,
                                      ResScreening = screening1)
str(seg_upd)
#> List of 4
#>  $ MonthVar: num [1:12] 1.06 1.28 2.6 2.79 3.92 ...
#>  $ Tmu     :'data.frame':    3 obs. of  5 variables:
#>   ..$ begin: num [1:3] 1 234 618
#>   ..$ end  : num [1:3] 233 617 1000
#>   ..$ mean : num [1:3] 0.932 1.313 2.149
#>   ..$ se   : num [1:3] 10 12.8 12.9
#>   ..$ np   : num [1:3] 233 384 383
#>  $ FitF    : Named num [1:1000] 1.85 1.92 1.99 2.06 2.12 ...
#>   ..- attr(*, "names")= chr [1:1000] "1" "2" "3" "4" ...
#>  $ CoeffF  : Named num [1:8] 0.36 0.274 0.525 0.488 0.41 ...
#>   ..- attr(*, "names")= chr [1:8] "cos1" "sin1" "cos2" "sin2" ...
seg_upd$Tmu
#>   begin  end      mean       se  np
#> 1     1  233 0.9323601  9.99568 233
#> 2   234  617 1.3132601 12.79474 384
#> 3   618 1000 2.1486382 12.91073 383
```

Test the UpdatedParametersForFixedCP function

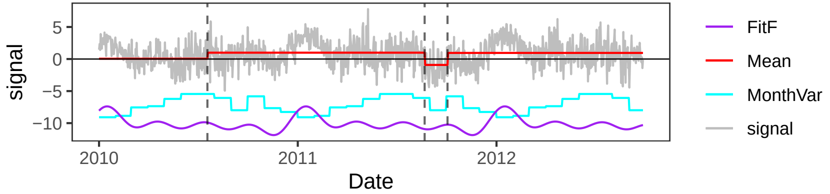
Visualize the time series after the updated segmentation

```
PlotSeg(OneSeries = df3d,
        SegRes = seg_upd,
        RemoveData = screening1$RemoveData)
```



Compare to the result before the cluster screening

```
PlotSeg(OneSeries = df3d,
        SegRes = seg3d)
```



Test the Validation function

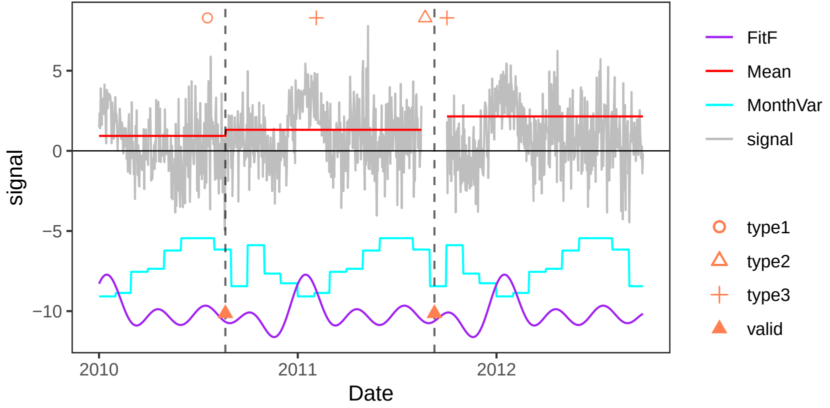
Validate the detected change-points with respect to the metadata

```
valid = Validation(OneSeries = df3d,
           Tmu = seg_upd$Tmu,
           MinDist = 62,
           Metadata = meta)
valid
#> # A tibble: 2 x 5
#>   CP         closestMetadata Distance type  valid
#>   <date>     <date>             <dbl> <chr> <dbl>
#> 1 2010-08-21 2010-07-19            33 type1     1
#> 2 2011-09-09 2011-10-02            17 type3     1
```

Test the Validation function

Visualize results with metadata

```
PlotSeg(OneSeries = df3d,
        SegRes = seg_upd,
        RemoveData = screening1$RemoveData,
        Metadata = meta,
        Validated_CP_Meta = valid)
```


signal
Date
FitF
Mean
MonthVar
signal
type1
type2
type3
valid