pre-assignment

Jakub Skrajny

27 04 2021

Libraries

```
library(openxlsx)
library(tseries)
```

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

```
library(forecast)
library(EnvStats)
```

```
##
## Attaching package: 'EnvStats'
```

```
## The following objects are masked from 'package:stats':
##
##     predict, predict.lm
```

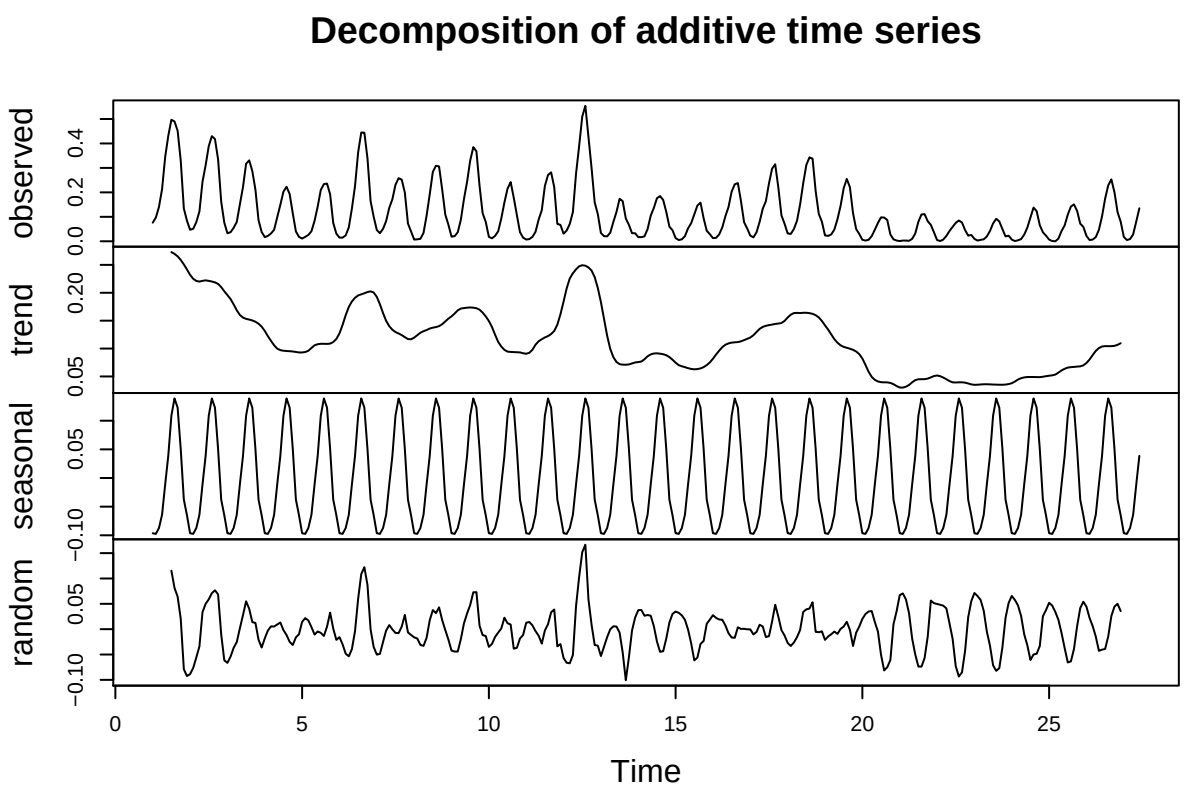
```
## The following object is masked from 'package:base':
##
##     print.default
```

Read data (date = number of days from 1900)

```
data <- read.xlsx("Lions_Den_data.xlsx")
timeseries <- unlist(data[2])
dates <- unlist(data[1])
```

Decomposition

```r
ts <- ts(timeseries, frequency = 12)
decompose <- decompose(ts, "additive")
plot(decompose)
```




```r
decompose$trend
```

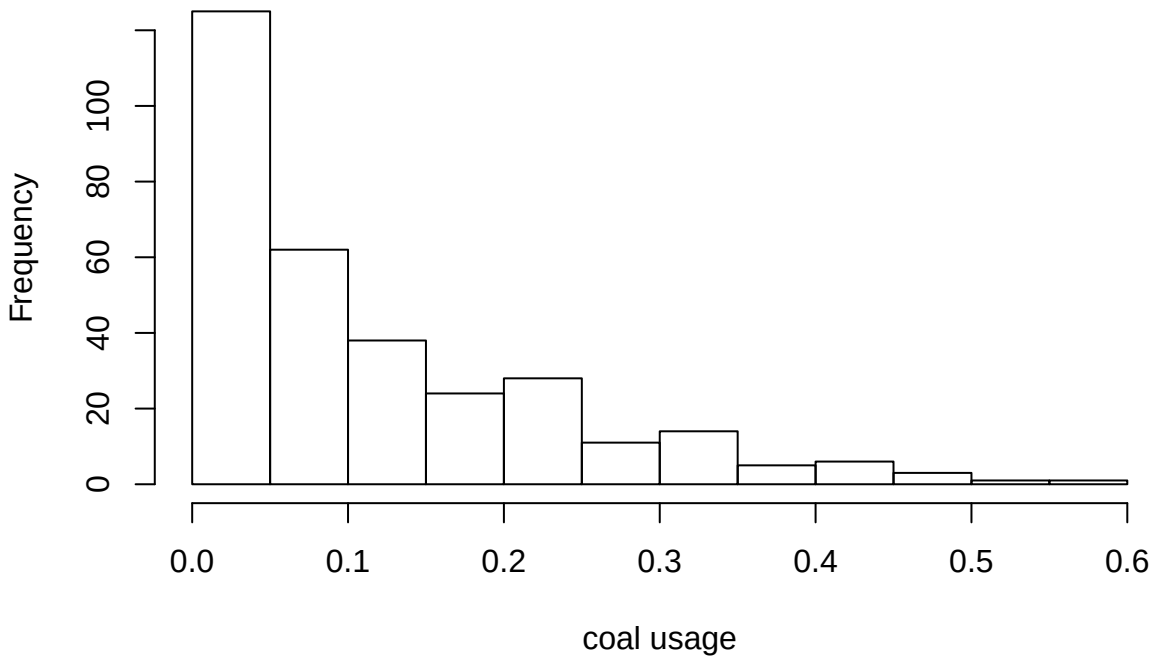
```
##           Jan        Feb        Mar        Apr        May        Jun        Jul
## 1          NA         NA         NA         NA         NA         NA 0.27287500
## 2  0.23216667 0.22516667 0.22125000 0.22004167 0.22141667 0.22225000 0.22133333
## 3  0.19591667 0.18879167 0.17920833 0.16858333 0.16025000 0.15550000 0.15325000
## 4  0.13150000 0.12225000 0.11383333 0.10587500 0.09991667 0.09716667 0.09608333
## 5  0.09304167 0.09379167 0.09604167 0.10108333 0.10579167 0.10779167 0.10850000
## 6  0.12716667 0.14237500 0.15983333 0.17462500 0.18445833 0.19104167 0.19520833
## 7  0.19262500 0.17929167 0.16350000 0.14966667 0.14004167 0.13375000 0.12962500
## 8  0.11962500 0.12395833 0.12833333 0.13112500 0.13316667 0.13583333 0.13750000
## 9  0.15766667 0.16275000 0.16850000 0.17170833 0.17258333 0.17329167 0.17370833
## 10 0.14937500 0.13862500 0.12475000 0.11183333 0.10358333 0.09766667 0.09454167
## 11 0.09070833 0.09237500 0.09791667 0.10700000 0.11316667 0.11675000 0.11991667
## 12 0.18425000 0.20787500 0.22570833 0.23500000 0.24191667 0.24741667 0.24937500
## 13 0.18341667 0.15316667 0.12300000 0.09941667 0.08450000 0.07600000 0.07204167
## 14 0.07583333 0.07679167 0.08083333 0.08620833 0.08954167 0.09116667 0.09150000
## 15 0.07466667 0.07029167 0.06820833 0.06666667 0.06487500 0.06350000 0.06283333
## 16 0.08179167 0.08929167 0.09625000 0.10250000 0.10687500 0.10929167 0.11066667
## 17 0.11983333 0.12391667 0.12979167 0.13533333 0.13883333 0.14083333 0.14204167
## 18 0.15570833 0.16075000 0.16362500 0.16395833 0.16358333 0.16416667 0.16416667
## 19 0.14162500 0.13400000 0.12545833 0.11683333 0.11050000 0.10583333 0.10300000
## 20 0.08220833 0.07066667 0.05850000 0.04929167 0.04391667 0.04104167 0.03970833
```

```
## 21 0.03025000 0.03004167 0.03158333 0.03483333 0.03954167 0.04341667 0.04500000
## 22 0.05170833 0.05045833 0.04800000 0.04508333 0.04175000 0.03945833 0.03920833
## 23 0.03500000 0.03520833 0.03566667 0.03600000 0.03604167 0.03579167 0.03554167
## 24 0.03791667 0.04083333 0.04450000 0.04691667 0.04829167 0.04900000 0.04900000
## 25 0.05133333 0.05200000 0.05333333 0.05691667 0.06079167 0.06375000 0.06587500
## 26 0.07412500 0.08045833 0.08833333 0.09575000 0.10087500 0.10370833 0.10441667
## 27          NA          NA          NA          NA          NA          NA
##             Aug         Sep         Oct         Nov         Dec
## 1   0.26975000 0.26525000 0.25887500 0.25087500 0.24166667
## 2   0.22016667 0.21862500 0.21587500 0.21020833 0.20262500
## 3   0.15195833 0.15037500 0.14808333 0.14454167 0.13929167
## 4   0.09566667 0.09525000 0.09475000 0.09391667 0.09304167
## 5   0.10854167 0.10833333 0.10891667 0.11166667 0.11725000
## 6   0.19725000 0.19912500 0.20133333 0.20254167 0.20041667
## 7   0.12695833 0.12420833 0.12045833 0.11708333 0.11658333
## 8   0.13854167 0.14020833 0.14358333 0.14866667 0.15362500
## 9   0.17329167 0.17229167 0.16970833 0.16445833 0.15766667
## 10 0.09395833 0.09366667 0.09341667 0.09312500 0.09158333
## 11 0.12237500 0.12604167 0.13162500 0.14279167 0.16150000
## 12 0.24854167 0.24545833 0.23979167 0.22804167 0.20862500
## 13 0.07116667 0.07100000 0.07162500 0.07333333 0.07508333
## 14 0.09083333 0.08983333 0.08812500 0.08475000 0.08000000
## 15 0.06320833 0.06441667 0.06675000 0.07033333 0.07516667
## 16 0.11116667 0.11162500 0.11295833 0.11516667 0.11741667
## 17 0.14300000 0.14412500 0.14483333 0.14575000 0.14954167
## 18 0.16350000 0.16225000 0.15983333 0.15579167 0.14941667
## 19 0.10145833 0.09975000 0.09770833 0.09479167 0.09008333
## 20 0.03950000 0.03937500 0.03845833 0.03612500 0.03279167
## 21 0.04508333 0.04516667 0.04616667 0.04833333 0.05079167
## 22 0.03950000 0.03954167 0.03891667 0.03741667 0.03579167
## 23 0.03533333 0.03516667 0.03512500 0.03545833 0.03633333
## 24 0.04900000 0.04883333 0.04875000 0.04945833 0.05045833
## 25 0.06666667 0.06712500 0.06766667 0.06812500 0.06987500
## 26 0.10425000 0.10433333 0.10487500 0.10675000 0.10954167
## 27
```

Outliers

Rosner's test suggest that there is only one outlier.

```
hist(timeseries,
  xlab = "coal usage",
)
```




```
rosnerTest(timeseries, k = 3)
```

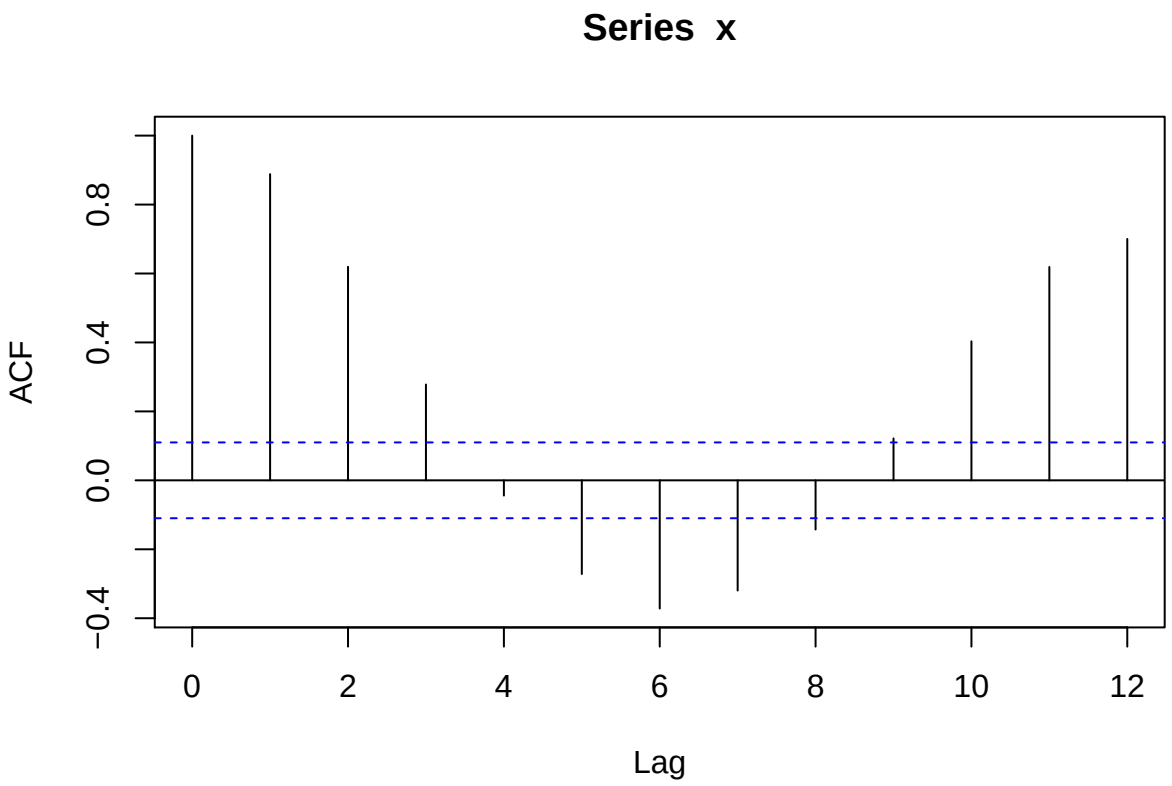
```
##
## Results of Outlier Test
## -------------------------
##
## Test Method:                     Rosner's Test for Outliers
##
## Hypothesized Distribution:       Normal
##
## Data:                            timeseries
##
## Sample Size:                     318
##
## Test Statistics:                 R.1 = 3.782021
##                                  R.2 = 3.487550
##                                  R.3 = 3.445420
##
## Test Statistic Parameter:        k = 3
##
## Alternative Hypothesis:          Up to 3 observations are not
##                                  from the same Distribution.
##
## Type I Error:                    5%
##
## Number of Outliers Detected:     1
##
##   i    Mean.i      SD.i Value Obs.Num    R.i+1 lambda.i+1 Outlier
## 1 0 0.1170692 0.1155284 0.554     140 3.782021   3.739949    TRUE
```

```
## 2 1 0.1156909 0.1130619 0.510     139 3.487550   3.739067   FALSE
## 3 2 0.1144430 0.1110335 0.497       7 3.445420   3.738181   FALSE
```

Autocorrelation and stationarity analysis

We can see that time-series is already stationary.

```
x <- timeseries
#autocorrelation
acf(x, lag.max = 12)
```



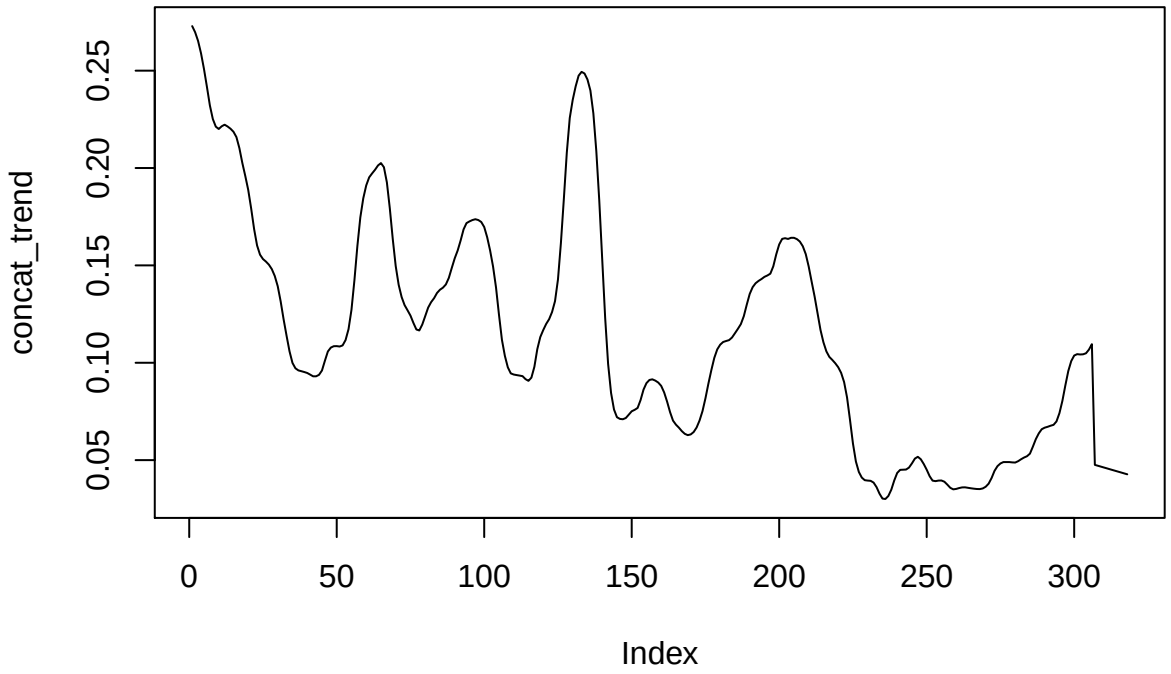

```
#stationary test
adf.test(x)
```

```
## Warning in adf.test(x): p-value smaller than printed p-value
```

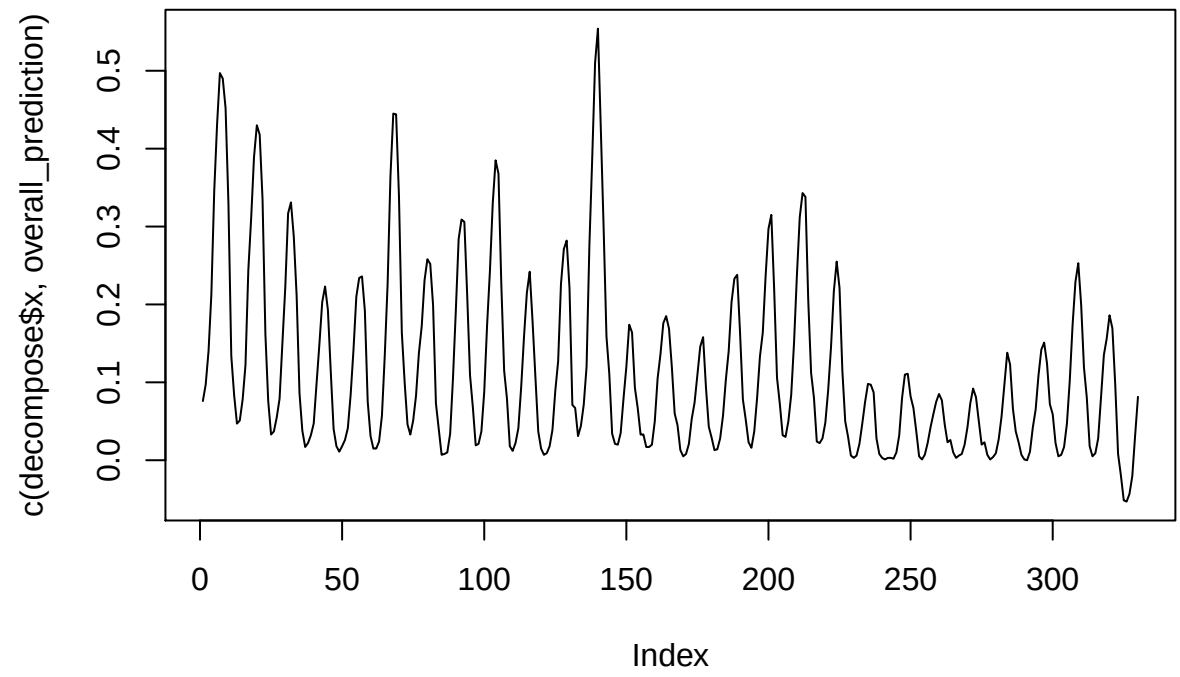
```
##
##  Augmented Dickey-Fuller Test
##
## data:  x
## Dickey-Fuller = -4.9571, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Linear regression

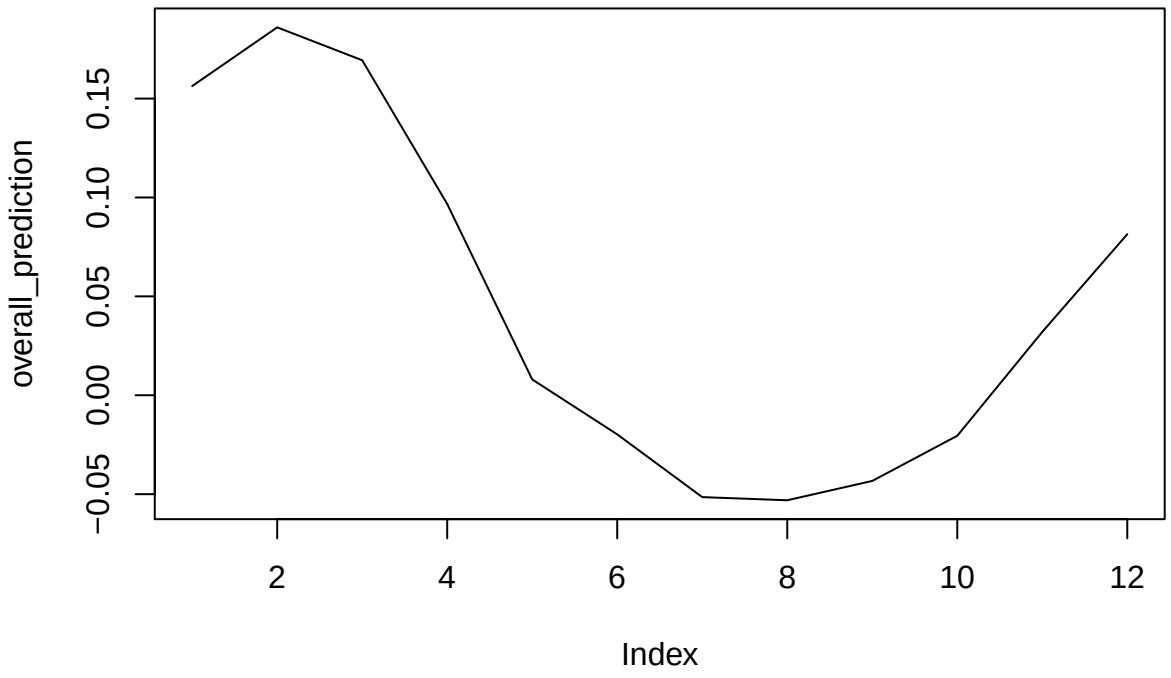
```
#trend
x <- as.numeric(decompose$trend)
x <- x[!is.na(x)]
time <- seq(1, length(x))
df <- data.frame(x, time)
model <- lm(x ~ time, data = df)
p <- as.data.frame(seq(length(x)+1, length(x)+12))
colnames(p) <- "time"
trend_prediction <- as.numeric(predict(model, newdata=p))
concat_trend <- c(x, trend_prediction)
plot(concat_trend, type="l")
```




```
#seasonal
seasonal_prediction <- as.numeric(tail(decompose$seasonal, n=12))
concat_seasonal <- c(
  as.numeric(decompose$seasonal),
  seasonal_prediction
)
# overall predicion
overall_prediction <- trend_prediction + seasonal_prediction
plot(c(decompose$x, overall_prediction), type="l")
```




```
plot(overall_prediction, type="l")
```




```
overall_prediction
```

```
##  [1]  0.156327689  0.186011968  0.169390158  0.096725078  0.008003908
##  [6] -0.019770146 -0.051498174 -0.053085561 -0.043319615 -0.020492003
## [11]  0.032028943  0.081419889
```