## Time Series Encoding and Reconstruction (encode-decode)

This example shows how to transform a time series into windows (p) and train an autoencoder to encode (p -> k) and reconstruct (k -> p) these windows, allowing evaluation of reconstruction quality.

Prerequisites - R packages: daltoolbox, ggplot2 - Python with PyTorch accessible via reticulate (backend called internally)

# Loading required packages  
library(daltoolbox)

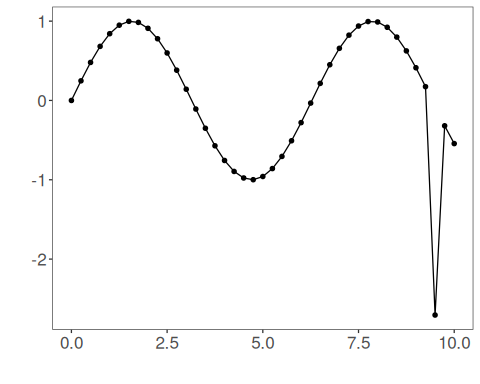
## Series for study

data(tsd)  
tsd$y[39] <- tsd$y[39] \* 6 # inject a synthetic outlier for illustration

sw\_size <- 5 # sliding window size (p)  
ts <- ts\_data(tsd$y, sw\_size) # series -> windows with p columns  
ts\_head(ts, 3)

## t4 t3 t2 t1 t0  
## [1,] 0.0000000 0.2474040 0.4794255 0.6816388 0.8414710  
## [2,] 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846  
## [3,] 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950

library(ggplot2)  
plot\_ts(x = tsd$x, y = tsd$y) +  
 theme(text = element\_text(size = 16))



## Data sampling

samp <- ts\_sample(ts, test\_size = 5)  
train <- as.data.frame(samp$train)  
test <- as.data.frame(samp$test)

## Train the model (encode-decode)

auto <- autoenc\_ed(5, 3) # 5 -> 3 -> 5 dimensions  
auto <- fit(auto, train)

## Reconstruction evaluation (train)

print(head(train)) # original windows (p columns)

## t4 t3 t2 t1 t0  
## 1 0.0000000 0.2474040 0.4794255 0.6816388 0.8414710  
## 2 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846  
## 3 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950  
## 4 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859  
## 5 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974  
## 6 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732

result <- transform(auto, train) # reconstructed windows (p columns)  
print(head(result))

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.009739388 0.2481622 0.4803727 0.6917699 0.8450632  
## [2,] 0.250326037 0.4793235 0.6825855 0.8440691 0.9506348  
## [3,] 0.476539016 0.6843566 0.8409365 0.9465727 0.9952309  
## [4,] 0.675030053 0.8420857 0.9478266 0.9923905 0.9837302  
## [5,] 0.845349133 0.9472832 0.9981513 0.9822196 0.9087911  
## [6,] 0.949238002 0.9943464 0.9849461 0.9116096 0.7818269

## Reconstruction of the test set

print(head(test))

## t4 t3 t2 t1 t0  
## 1 0.9893582 0.9226042 0.7984871 0.6247240 0.4121185  
## 2 0.9226042 0.7984871 0.6247240 0.4121185 0.1738895  
## 3 0.7984871 0.6247240 0.4121185 0.1738895 -2.7054403  
## 4 0.6247240 0.4121185 0.1738895 -2.7054403 -0.3195192  
## 5 0.4121185 0.1738895 -2.7054403 -0.3195192 -0.5440211

result <- transform(auto, test)  
print(head(result))

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.9873567 0.9208630 0.7961031 0.6290081 0.4069995  
## [2,] 0.9285811 0.8025890 0.6257131 0.4080846 0.1668130  
## [3,] 0.8388368 0.2937016 -0.2902761 -0.8001619 -1.2479628  
## [4,] 0.1473438 -0.1190581 -0.4160034 -0.6399105 -0.8431425  
## [5,] -0.2363527 -0.5470047 -0.8223292 -1.0399469 -1.1889921