## 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)

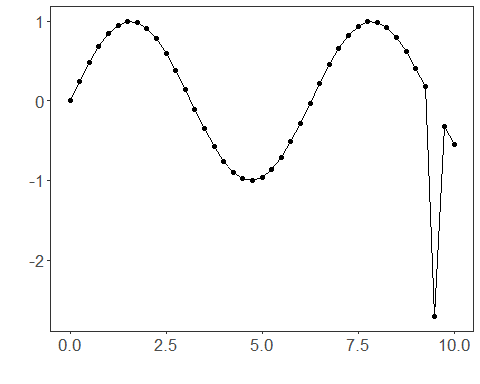
## Series for study

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,] -0.002183719 0.2478515 0.4718369 0.6787611 0.8382691  
## [2,] 0.248861864 0.4806462 0.6827217 0.8357649 0.9545360  
## [3,] 0.478426993 0.6817709 0.8456433 0.9458112 0.9968357  
## [4,] 0.682134032 0.8419870 0.9497386 0.9993858 0.9826288  
## [5,] 0.844489813 0.9510972 0.9964575 0.9866852 0.9111491  
## [6,] 0.948821664 0.9969462 0.9828075 0.9096808 0.7790877

## 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,] 0.9884202 0.9245862 0.7965988 0.6170152 0.4101516  
## [2,] 0.9289421 0.8059270 0.6227967 0.4146117 0.1751617  
## [3,] 0.9521191 0.4856281 0.1133452 -0.3401330 -0.8778204  
## [4,] 0.2402520 0.0523901 -0.4660450 -0.5119648 -0.9863667  
## [5,] -0.1180102 -0.3889379 -0.6125702 -0.8266070 -0.9720942