## 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)

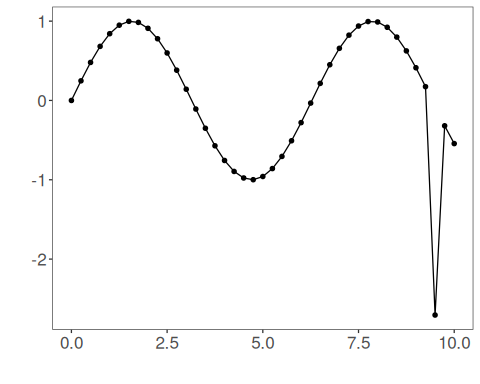
## Series for study

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,] 0.0112375 0.2477744 0.4803282 0.6918381 0.8445019  
## [2,] 0.2503530 0.4800859 0.6806654 0.8469747 0.9504015  
## [3,] 0.4782169 0.6861279 0.8410481 0.9460607 0.9923353  
## [4,] 0.6756940 0.8428720 0.9500131 0.9896486 0.9834681  
## [5,] 0.8455303 0.9484267 0.9974016 0.9801796 0.9092858  
## [6,] 0.9508401 0.9959251 0.9843779 0.9115022 0.7813786

## 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,] 0.9886540 0.9223537 0.7960541 0.6293573 0.4054892  
## [2,] 0.9291114 0.8046473 0.6267461 0.4075139 0.1661665  
## [3,] 0.8273664 0.2886907 -0.2889423 -0.8216463 -1.2503543  
## [4,] 0.1651905 -0.1032671 -0.4032366 -0.6384751 -0.8437970  
## [5,] -0.2315048 -0.5395460 -0.8178805 -1.0430510 -1.1829259