Neural Network Imputer Demonstration

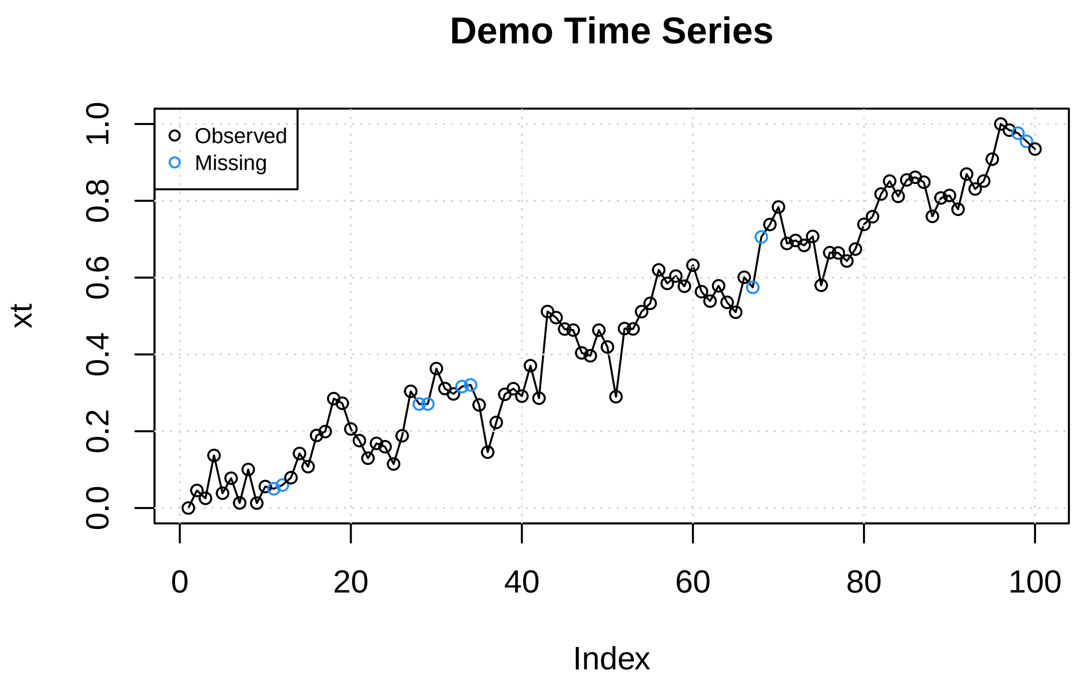
Algorithm:

1. Interpolate missing values in the original time series using linear interpolation.
2.
    a. Estimate the trend component of the input time series
    b. Estimate the periodic component(s) of the input time series
3. Simulate a sufficient number of time series with the same trend and periodic structure by applying small perturbations (targets).
4. Impose a gap structure on the simulated time series (i.e., the same gap structure as observed in the original time series) and produce K unique 'gappy' time series (inputs).
5. Construct, compile, and fit a neural network autoencoder model using the newly created input and target time series. Using the autoencoder, predict on the original time series.
6. Extract the predicted values of the missing data points to complete the original time series.
7. Repeat steps 2-6 using the output of Step 6 as the input for Step 2, each time storing the predictions.
8. Return the final predictions as the average predictions over the total number of iterations.

Setup

```
## Importing libraries
library(tsinterp)
library(interpTools)
library(tensorflow)
library(keras)
```

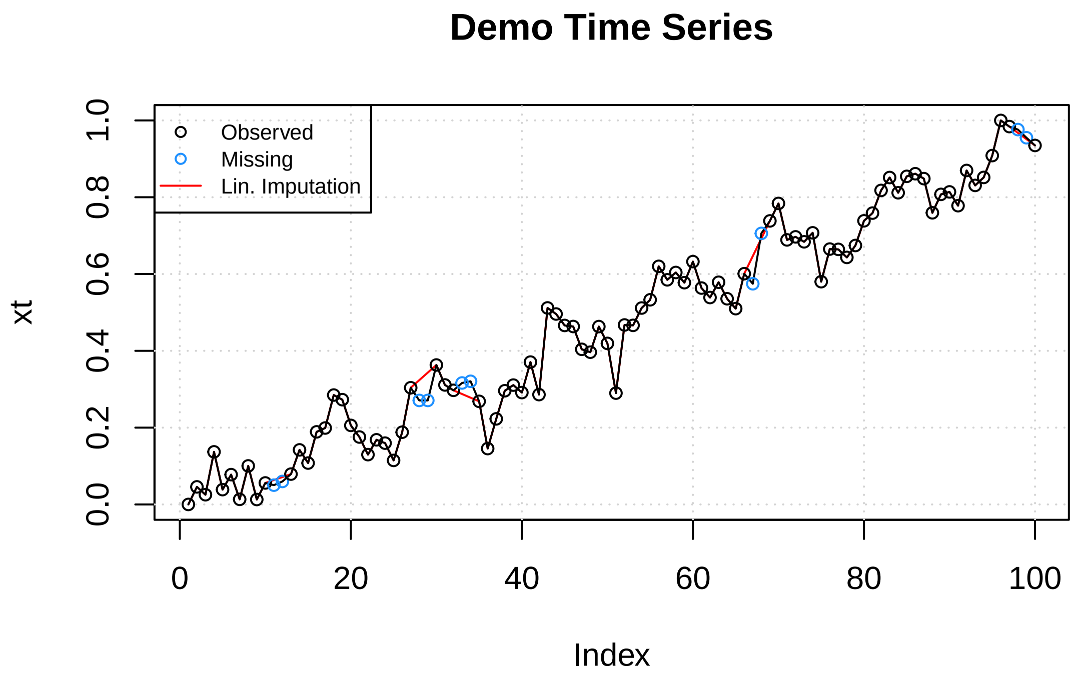
```
## Defining the time series for demo
set.seed(42)
xt = simXt(N = 100, mu = 0, numTrend = 1, numFreq = 2)$Xt
xt = (xt - min(xt)) / (max(xt) - min(xt))
x_gapped = simulateGaps(list(xt), p = 0.1, g = 2, K = 1)
x_gappy = x_gapped[[1]]$p0.1$g2[[1]]
plot(xt, type = 'l', main = 'Demo Time Series'); grid()
lines(xt, type = 'p', col = ifelse(is.na(x_gappy), 'dodgerblue', 'black'), cex = 0.8)
legend('topleft', legend = c('Observed', 'Missing'), col = c('black', 'dodgerblue'),
       pch = 1, cex = 0.7)
```



Demo 1: Initialization Step

The first step of the algorithm is to use linear imputation to fill in the gaps for the original input time series.

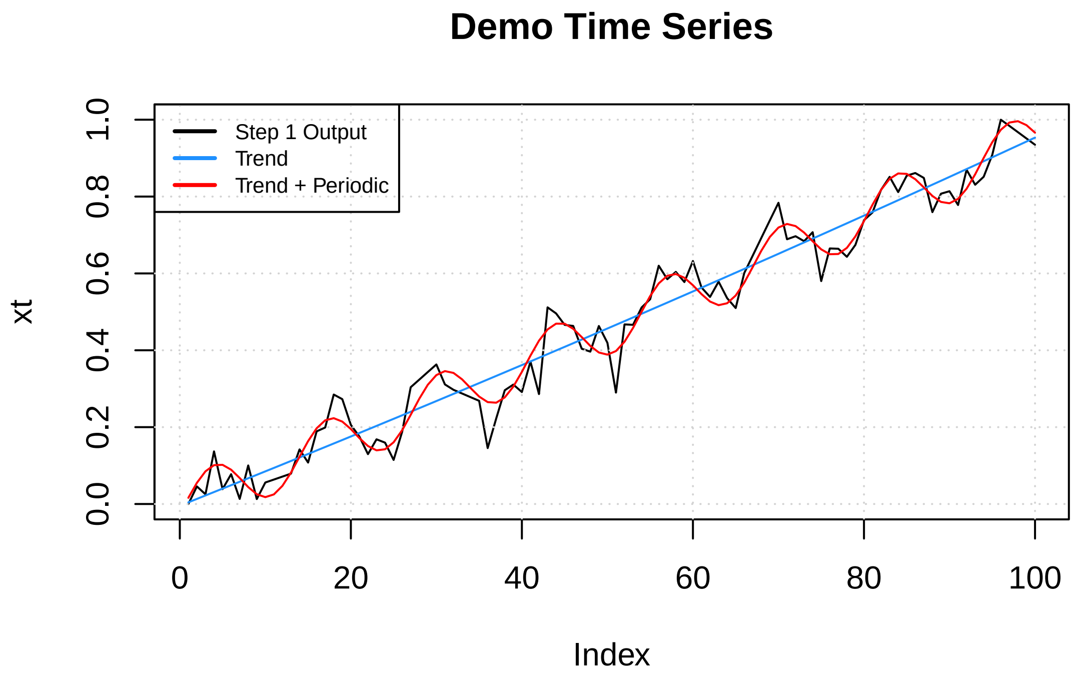
```r
step_1_demo <- function(x0){
  xI = initialize_demo(x0)
  plot(xI, type = 'l', main = 'Demo Time Series', ylab = 'xt', col = 'red'); grid()
  lines(xt, type = 'l')
  lines(xt, type = 'p', col = ifelse(is.na(x_gappy), 'dodgerblue', 'black'), cex = 0.8)
  legend('topleft', legend = c('Observed', 'Missing', 'Lin. Imputation'),  lty = c(0, 0, 1),
         col = c('black', 'dodgerblue', 'red'), pch = c(1, 1, -1), cex = 0.7)
  return(xI)
}
step1 = step_1_demo(x_gappy)
```



Demo 2: Trend and Periodic Estimation

The next step of the algorithm is to use functions from Wesley's TSinterp package to estimate the trend and periodic components of the input time series.

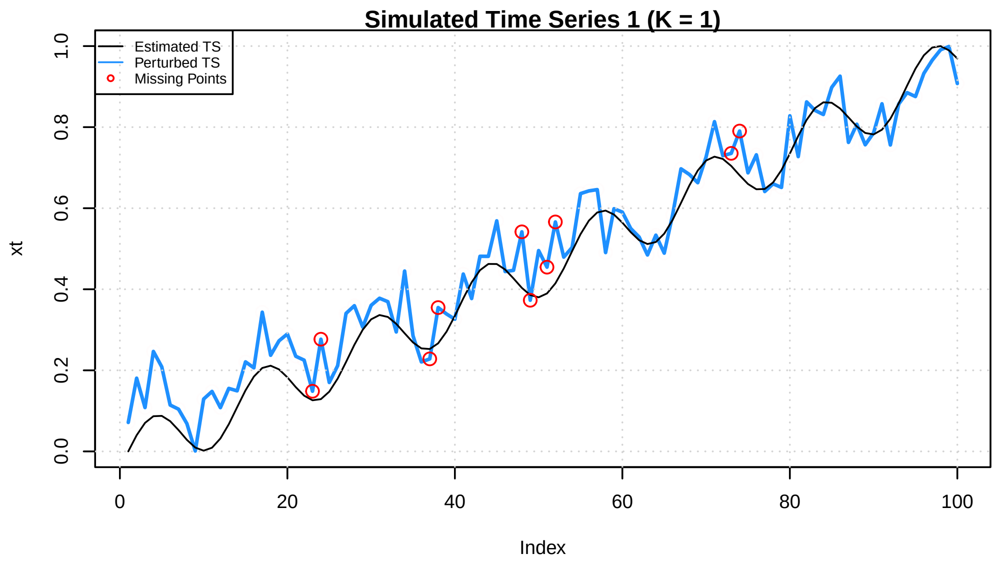
```r
step_2_demo <- function(xI){
  x_est = estimator_demo(xI)
  Mt = x_est[[1]]; Tt = x_est[[2]]; Xt = x_est[[3]]
  plot(xI, type = 'l', main = 'Demo Time Series', ylab = 'xt'); grid()
  lines(Mt, type = 'l', col = 'dodgerblue'); lines(Xt, type = 'l', col = 'red')
  legend('topleft', legend = c('Step 1 Output', 'Trend', 'Trend + Periodic'), lty = 1,
         lwd = 2, col = c('black', 'dodgerblue', 'red'), cex = 0.7)
  return(Xt)
}
step2 = step_2_demo(step1)
```

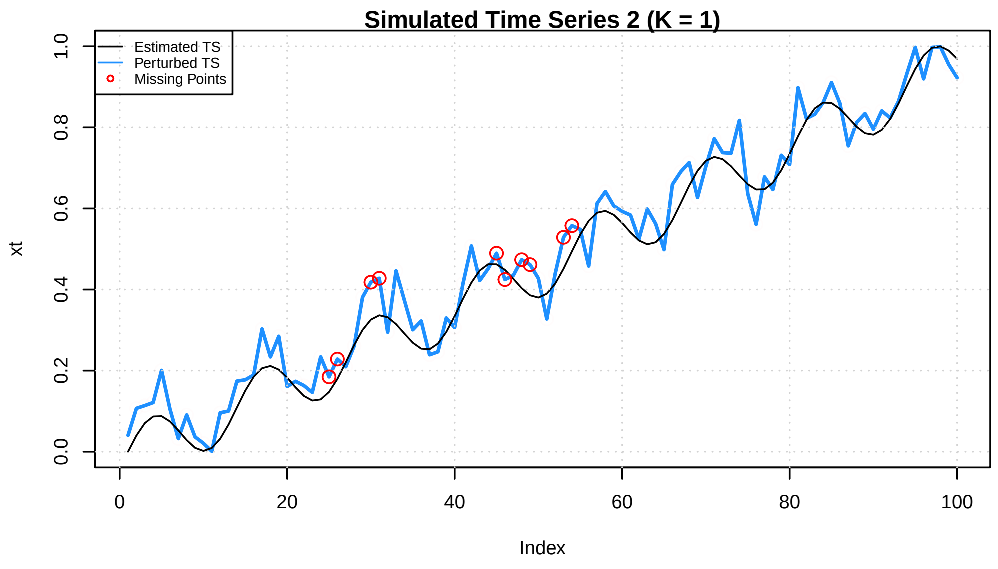


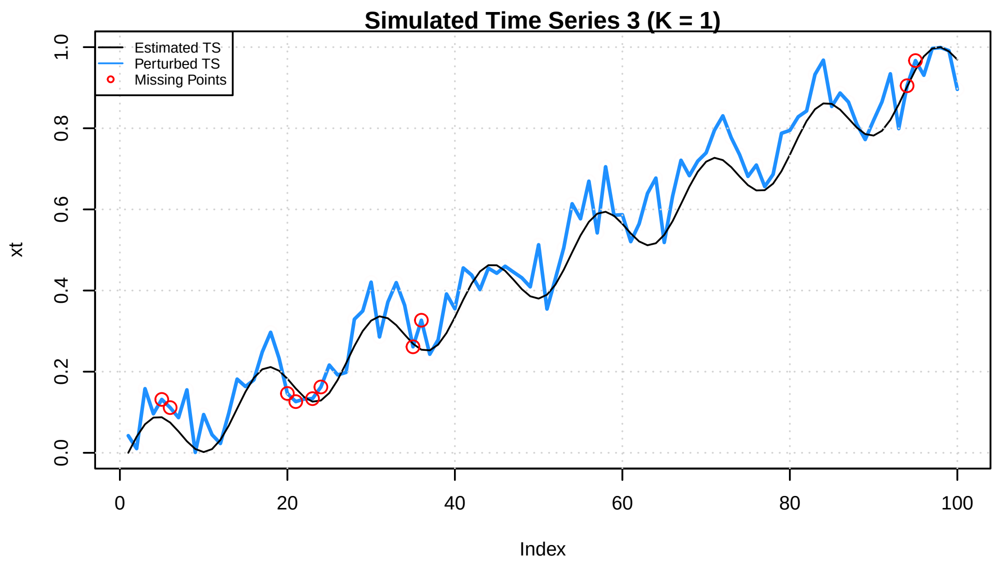
Demo 3/4: Time Series Simulation

The next steps are to generate an appropriate number of time series with the same trend and periodic structure by applying small perturbations to the output of step 2 and then imposing a gap structure similar to that of the original input time series.

```r
step_34_demo <- function(x_est, n_series, var, p, g, K){
  x = simulator_demo(x_est, n_series, var, p, g, K)
  inputs = x[[1]]; targets = x[[2]]
  par(mfcol=c(3,1), mar = c(5.1, 4.1, 1, 2.1))
  x_est_scaled = (x_est - min(x_est)) / (max(x_est) - min(x_est))
  for (i in 1:3){
    plot(targets[i,], type = 'l', col = 'dodgerblue', lwd = 2, ylab = 'xt',
         main = paste0('Simulated Time Series ', i, ' (K = 1)'));
    lines(x_est_scaled, type = 'l', col = 'black'); grid()
    lines(targets[i,], type = 'p', cex = 1.5,
          col = ifelse(inputs[i,] == 0, 'red', adjustcolor( "red", 0.01)))
    legend('topleft', legend = c('Estimated TS', 'Perturbed TS', 'Missing Points'), cex = 0.7,
           pch = c(-1, -1, 1), lty = c(1, 1, 0), col = c('black', 'dodgerblue', 'red'))
  }
  return(x)
}
step34 = step_34_demo(step2, n_series = 3, var = 0.05, p = 0.1, g = 2, K = 1)
```


Simulated Time Series 1 (K = 1)
Estimated TS
Perturbed TS
Missing Points
xt
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
Index


Simulated Time Series 2 (K = 1)
Estimated TS
Perturbed TS
Missing Points
xt
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
Index


Simulated Time Series 3 (K = 1)
Estimated TS
Perturbed TS
Missing Points
xt
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
Index

```r
inputs = step34[[1]]; targets = step34[[2]]
```