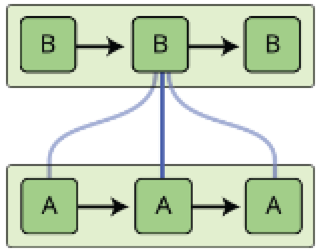
MET CS688
Web Analytics and Mining

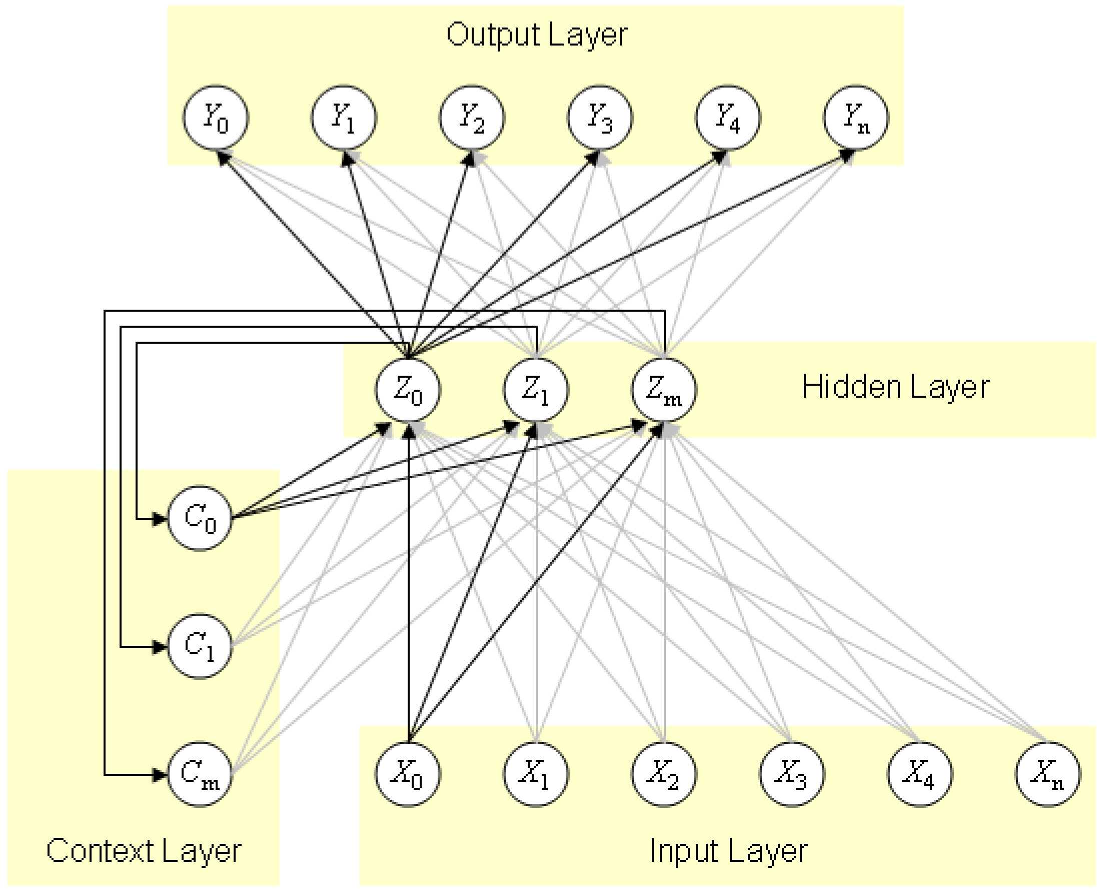
Zlatko Vasilkoski

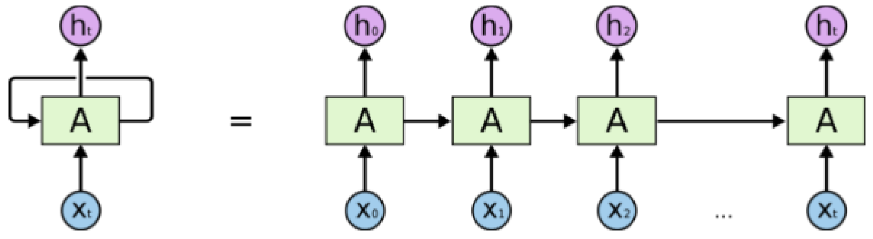
Recurrent Neural Networks



Recurrent Neural Networks

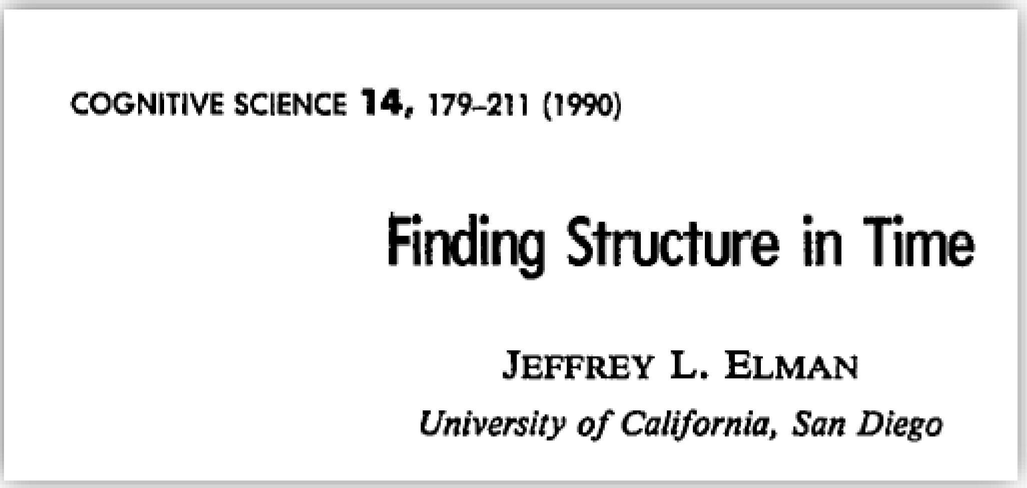
- Attention and Augmented Recurrent Neural Networks work with sequences of data like text, audio and video.
- Contain Context layer
- A special variant – “long short-term memory” LSTM networks
- Very powerful, remarkable results in many tasks including
  - translation,
  - voice recognition, and
  - image captioning.





An unrolled recurrent neural network.

Elman’s RNN



COGNITIVE SCIENCE **14**, 179–211 (1990)

Finding Structure in Time

JEFFREY L. ELMAN
University of California, San Diego

- Introduced by Elman in 1990 with intention to analyze language.

200 ELMAN

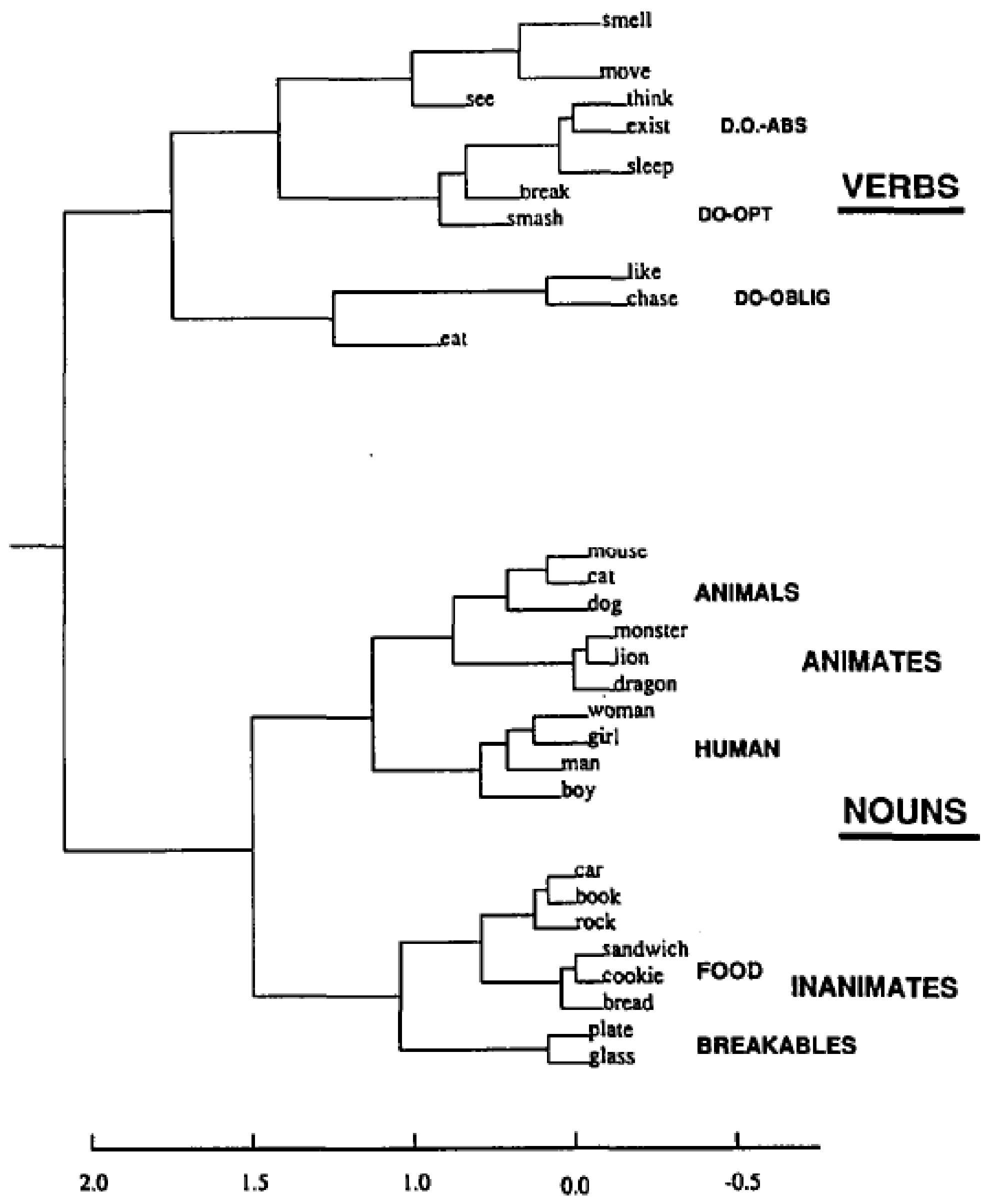


Figure 7. Hierarchical cluster diagram of hidden unit activation vectors in simple sentence prediction task. Labels indicate the inputs which produced the hidden unit vectors; inputs were presented in context, and the hidden unit vectors averaged across multiple contexts.

Recurrent Neural Nets in R – RSNNS package

See “RNN Code” R Project on Blackboard

- RSNNS package enables a use of wide variety of network types and possible applications.
- A convenient feature in R is the inclusion of datasets along with software packages.
- This is important for NNS standardized tests as well as for examples of the usage of the package.
- Use the list *snnsData()* for all of the available datasets in RSNNS.

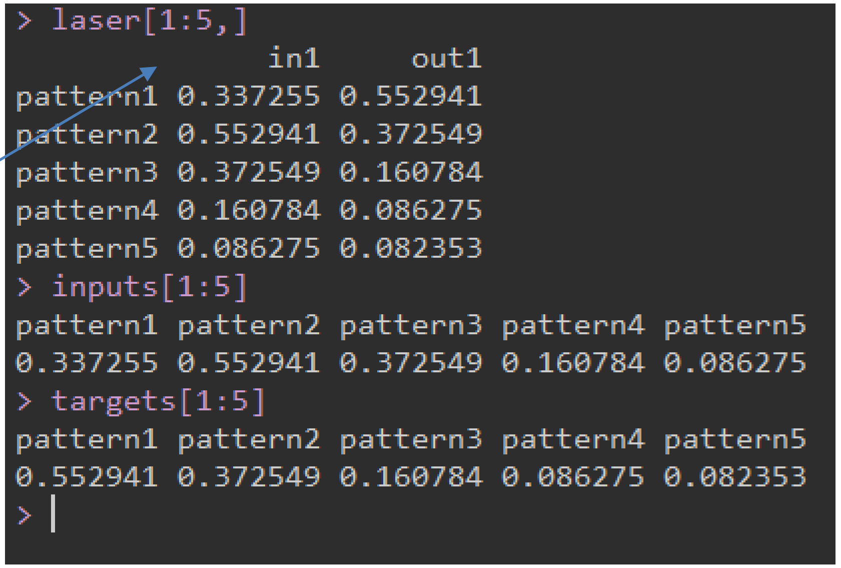
- These functions can be used to pick the right columns according to their names.
  - *inputColumns()* Extracts all columns from a matrix whose column names begin with "**in**"
  - *outputColumns()* Extracts all columns from a matrix whose column names begin with "**out**"
- Functions *splitForTrainingAndTest()* can be used to split the data in a training and a test set.

RSNNS – Access & Splitting the Data

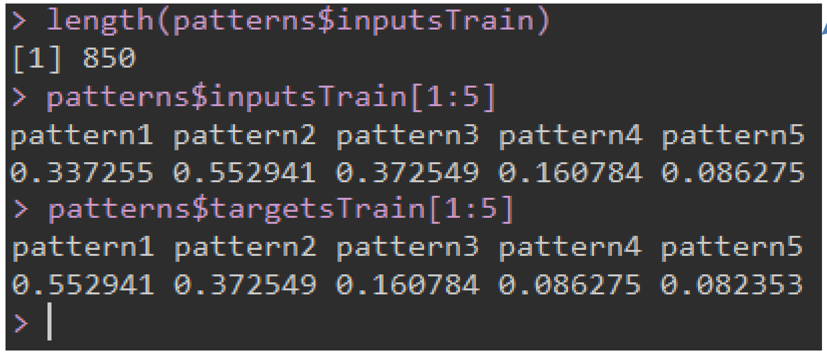
	Dat Table	
Inputs		Outouts
1		1
2	Train	2
-	Data	-
-		-
850		850
851		851
-	Test	-
-	Data	-
1000		1000

Access and split the data into Train and Test data – Note how the data is split.

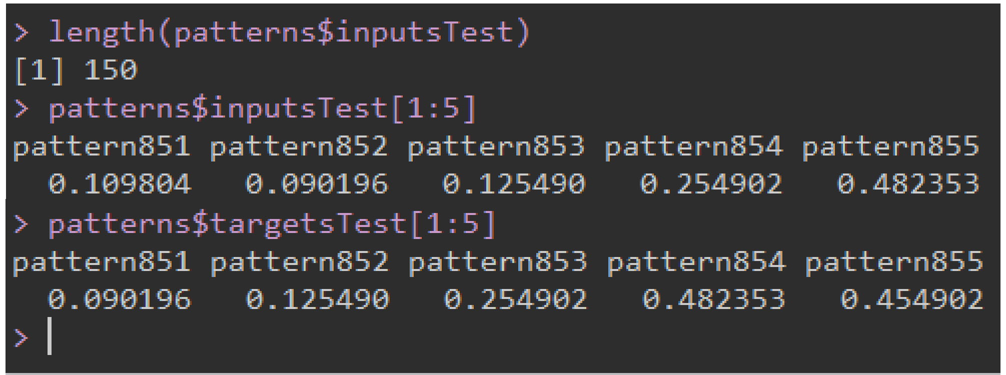
```
# Example: RSNNS Loading data
library("RSNNS")
# Load the Data (more data @  http://sci2s.ugr.es/keel/datasets.php)
data("snnsData")
laser <- snnsData$laser_1000.pat

# Spliting the Data
inputs <- laser[,inputColumns(laser)] # extract all col. names begin with "in"
targets <- laser[,outputColumns(laser)] # extract all col with “out"
# Split the input data (1000) into train (850) and test (150) specified
# by the argument ratio=0.15
patterns <- splitForTrainingAndTest(inputs, targets, ratio = 0.15)
```



```
> laser[1:5,]
              in1      out1
pattern1 0.337255 0.552941
pattern2 0.552941 0.372549
pattern3 0.372549 0.160784
pattern4 0.160784 0.086275
pattern5 0.086275 0.082353
> inputs[1:5]
pattern1 pattern2 pattern3 pattern4 pattern5
0.337255 0.552941 0.372549 0.160784 0.086275
> targets[1:5]
pattern1 pattern2 pattern3 pattern4 pattern5
0.552941 0.372549 0.160784 0.086275 0.082353
> |
```

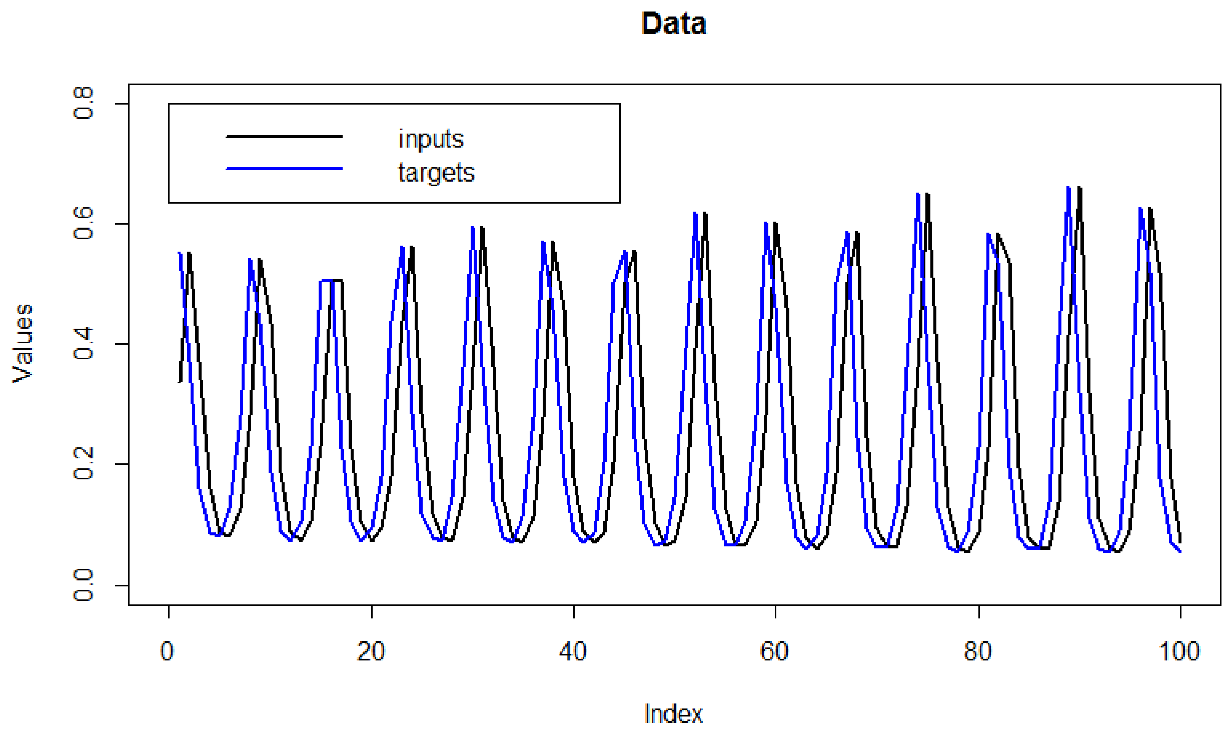


```
> length(patterns$inputsTrain)
[1] 850
> patterns$inputsTrain[1:5]
pattern1 pattern2 pattern3 pattern4 pattern5
0.337255 0.552941 0.372549 0.160784 0.086275
> patterns$targetsTrain[1:5]
pattern1 pattern2 pattern3 pattern4 pattern5
0.552941 0.372549 0.160784 0.086275 0.082353
> |
```



```
> length(patterns$inputsTest)
[1] 150
> patterns$inputsTest[1:5]
pattern851 pattern852 pattern853 pattern854 pattern855
  0.109804   0.090196   0.125490   0.254902   0.482353
> patterns$targetsTest[1:5]
pattern851 pattern852 pattern853 pattern854 pattern855
  0.090196   0.125490   0.254902   0.482353   0.454902
> |
```

RSNNS – Input Data Overview



Always have a look at the data to understand it better.

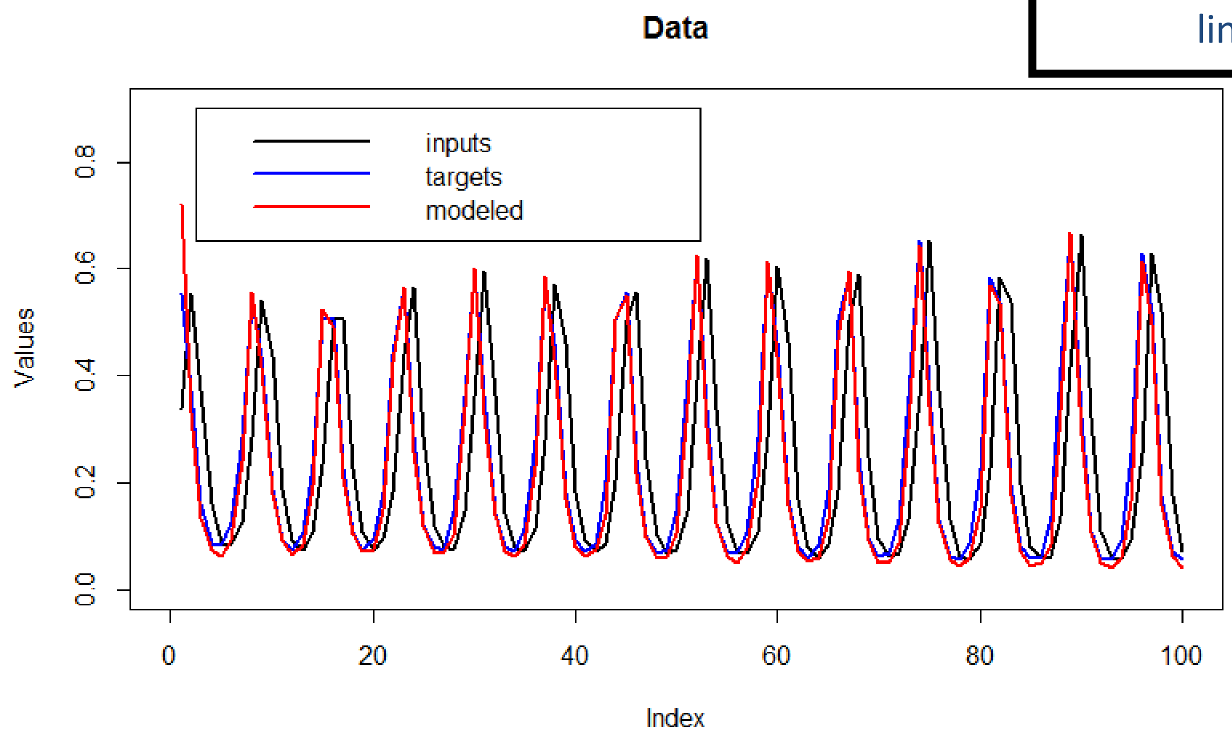
R is a powerful tools for data visualization.
Very useful visualizing the data used in neural net modeling.

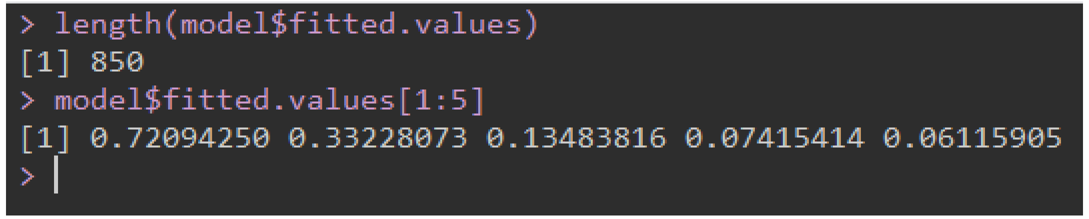
Note how the target data (blue) lags after the input data (black).

RSNNS - 1. Regression Predictions on Train Data

- Illustration of how to use an Elman network (Elman 1990) for time series regression.
- The result (fitted values) is stored in *model$fitted.values* – same size as *inputs* and *targets*.
- Note how the fitted values (red) match the target values (blue)

```
# Example: RSNNS Prediction using Elman network
model <- elman(patterns$inputsTrain, patterns$targetsTrain,
          size = c(8, 8), learnFuncParams = c(0.1), maxit = 500,
          inputsTest = patterns$inputsTest, targetsTest = patterns$targetsTest,
          linOut = FALSE)
```

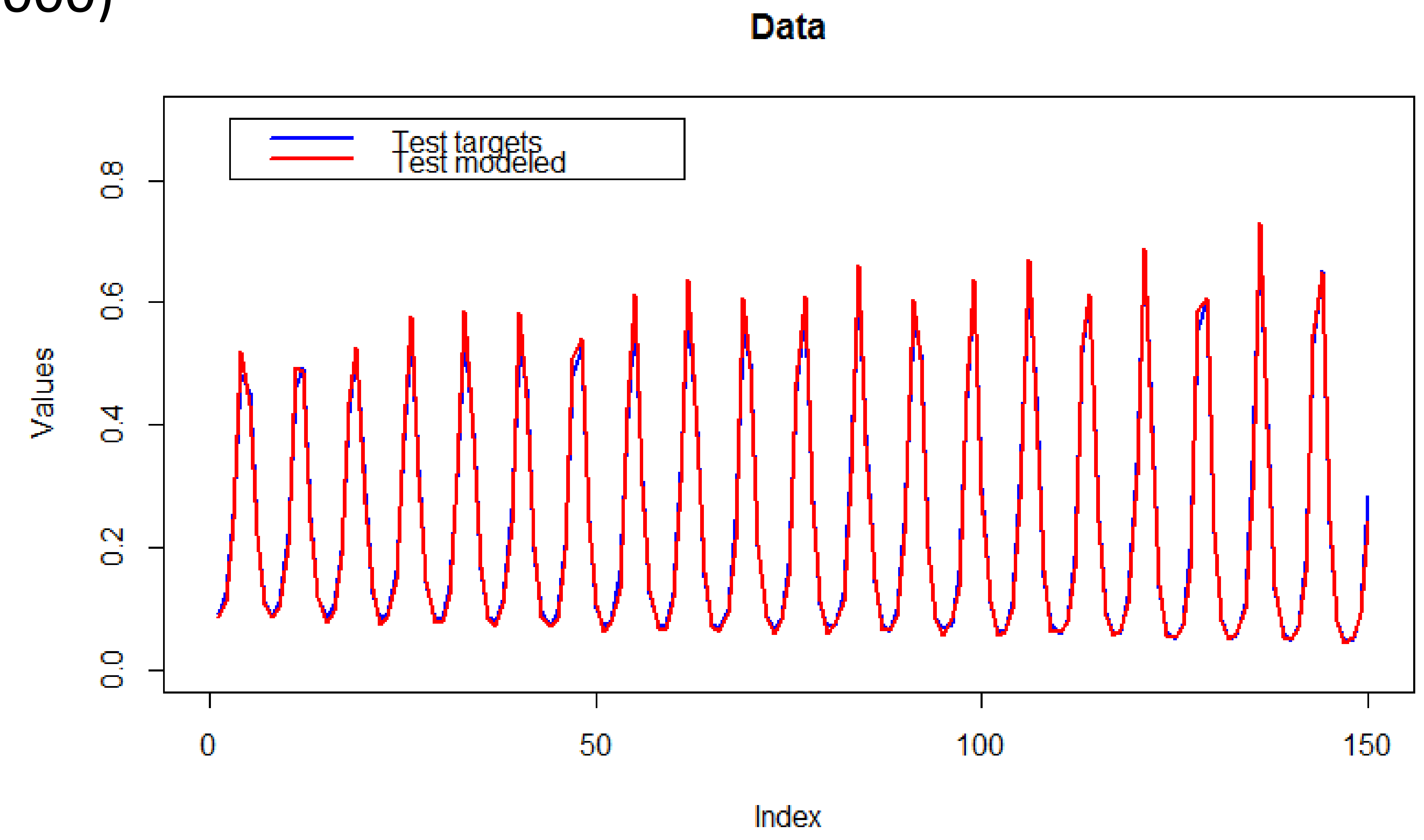




```
> length(model$fitted.values)
[1] 850
> model$fitted.values[1:5]
[1] 0.72094250 0.33228073 0.13483816 0.07415414 0.06115905
> |
```

RSNNS - 1. Regression Predictions on Test Data

Test Targets data in blue vs Test Modeled (predicted) values in red (indices 851 to 1000)



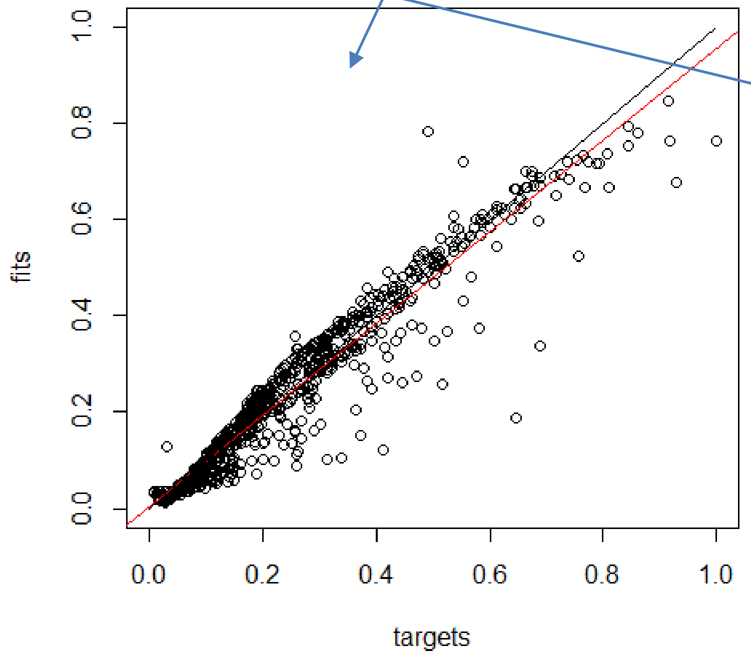
RSNNS - 1. Regression Error Estimates

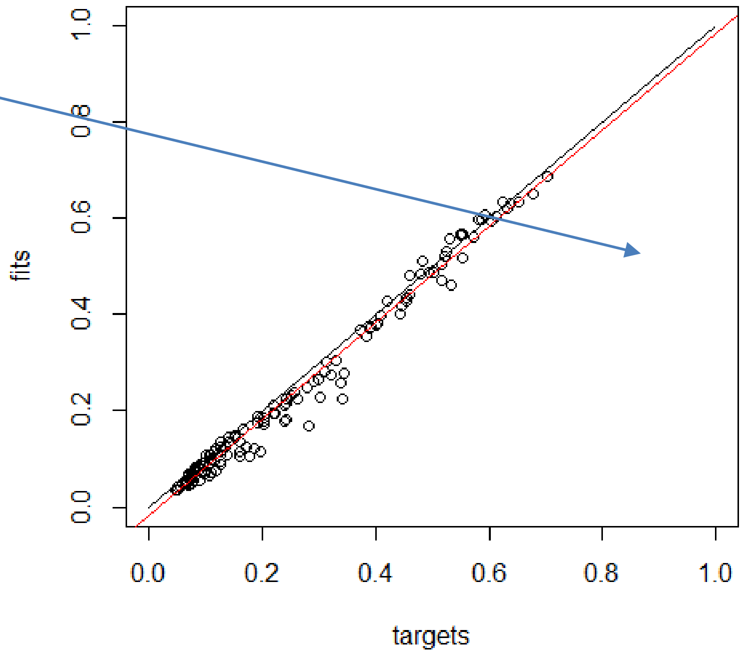
```
# Example: RSNNS Elman network – Plot Errors
plotRegressionError(patterns$targetsTrain, model$fitted.values)
plotRegressionError(patterns$targetsTest, model$fittedTestValues)
```

Error plots

Regression plots

- For the **training data** - showing an optimal linear fit (black line), and linear fit (red line) to the **training data** (dots).
- Same for the **test data**. Note how the error for the test data (indices 851 to 1000) is larger.





IIoT Work Study Example 1 – Forecasting with Neural Networks Model

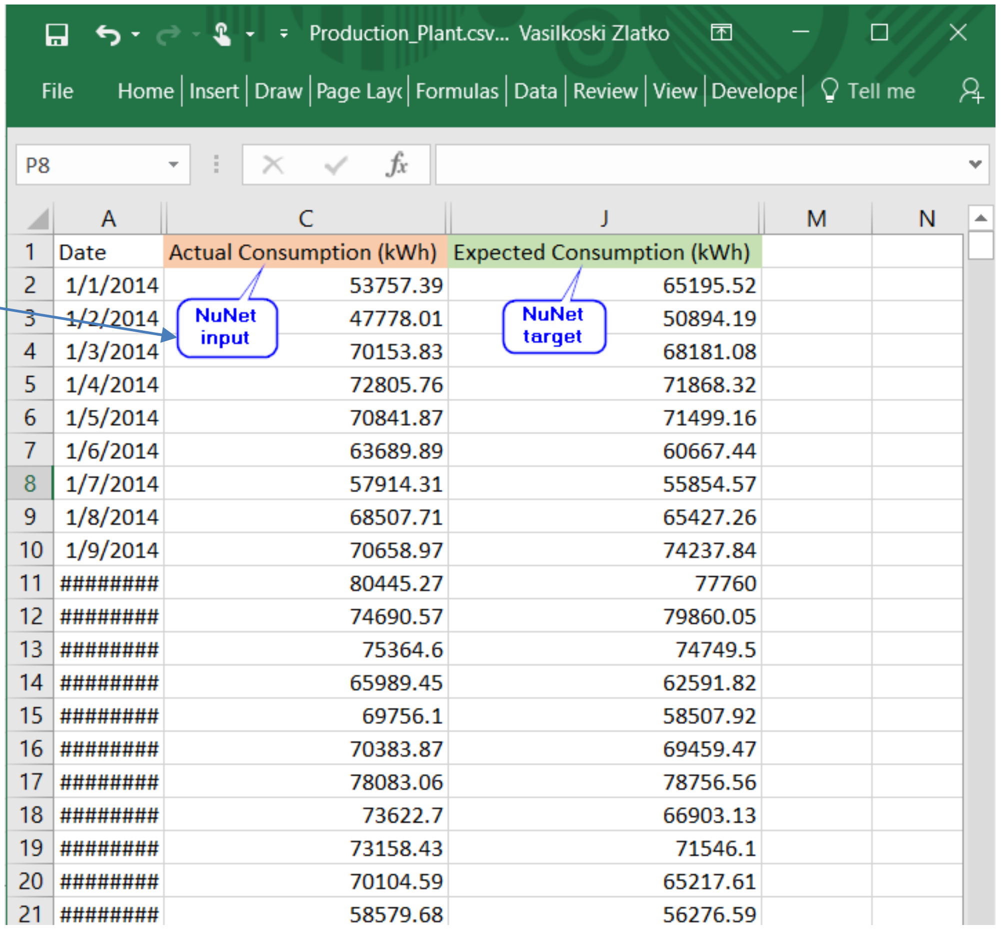
As an exercise replicate the calculated linear regression (Expected Consumption) from slide on using RNN.

Train the RNN on the input tags, for which we need to provide targets (or labels) – in this case the ones we already calculated.

We want to test if the RNN will give comparable predictions to the linear regression model.

- Selects the appropriate tags (columns) from the provided CSV data set.
- Split data into training (85%) and testing (15%)
- Scale & normalize data. NN like values [0 to 1]
- Run elaman RNN
- Plot results and estimate errors

- Refer to RNN slides for more detailed information



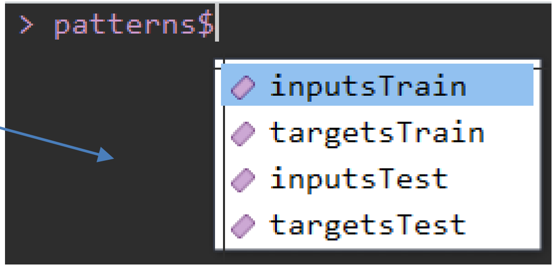
	A	C	J	M	N
1	Date	Actual Consumption (kWh)	Expected Consumption (kWh)		
2	1/1/2014	53757.39	65195.52		
3	1/2/2014	47778.01	50894.19		
4	1/3/2014	70153.83	68181.08		
5	1/4/2014	72805.76	71868.32		
6	1/5/2014	70841.87	71499.16		
7	1/6/2014	63689.89	60667.44		
8	1/7/2014	57914.31	55854.57		
9	1/8/2014	68507.71	65427.26		
10	1/9/2014	70658.97	74237.84		
11	########	80445.27	77760		
12	########	74690.57	79860.05		
13	########	75364.6	74749.5		
14	########	65989.45	62591.82		
15	########	69756.1	58507.92		
16	########	70383.87	69459.47		
17	########	78083.06	78756.56		
18	########	73622.7	66903.13		
19	########	73158.43	71546.1		
20	########	70104.59	65217.61		
21	########	58579.68	56276.59		

RSNNS – Access & Splitting the Data

- Read the raw data into R from a CSV file ("EIS_Data.csv“ on Blackboard)
- Access and split the data into Train and Test data.

```
# Example: RSNNS
# Separate Data into train and test
inputs <- temp.CSV$Actual.Consumption..kWh. # used for Training
targets <- temp.CSV$Expected.Consumption..kWh. # (Regression data) used as Targets
# Split the input data into train and test specified by the argument ratio=0.15
patterns <- splitForTrainingAndTest(inputs, targets, ratio = 0.15)
```

	Dat Table	
Inputs		Outouts
1		1
2	Train	2
-	Data	-
-		-
850		850
851		851
-	Test	-
-	Data	-
1000		1000



```
> patterns$
```

- Take input to be Actual Consumption (plant’s consumption)
- Take targets to be the Expected.Consumption..kWh (from the regression model)
- Train the Nu-Net on the Actual Consumption as an input to forecast (predict)

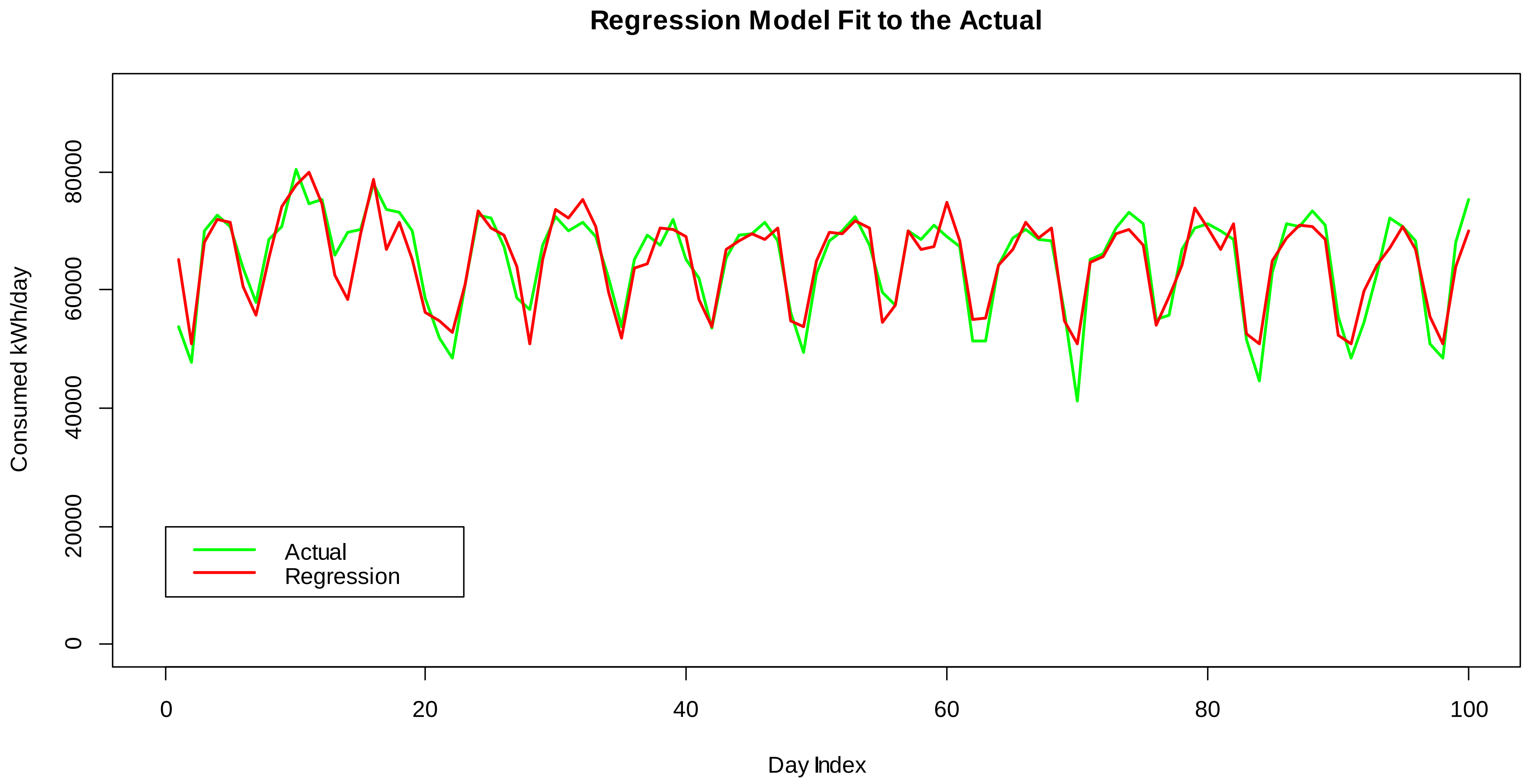
    the Expected Consumption that is also given by the linear regression equation

Expected Consumption (kWh) = A+ B*Avg Temperature + C*Total Production +D*Seasonal Indicator

- **Note** that in the NN model we are not using any of the features used in the equation:

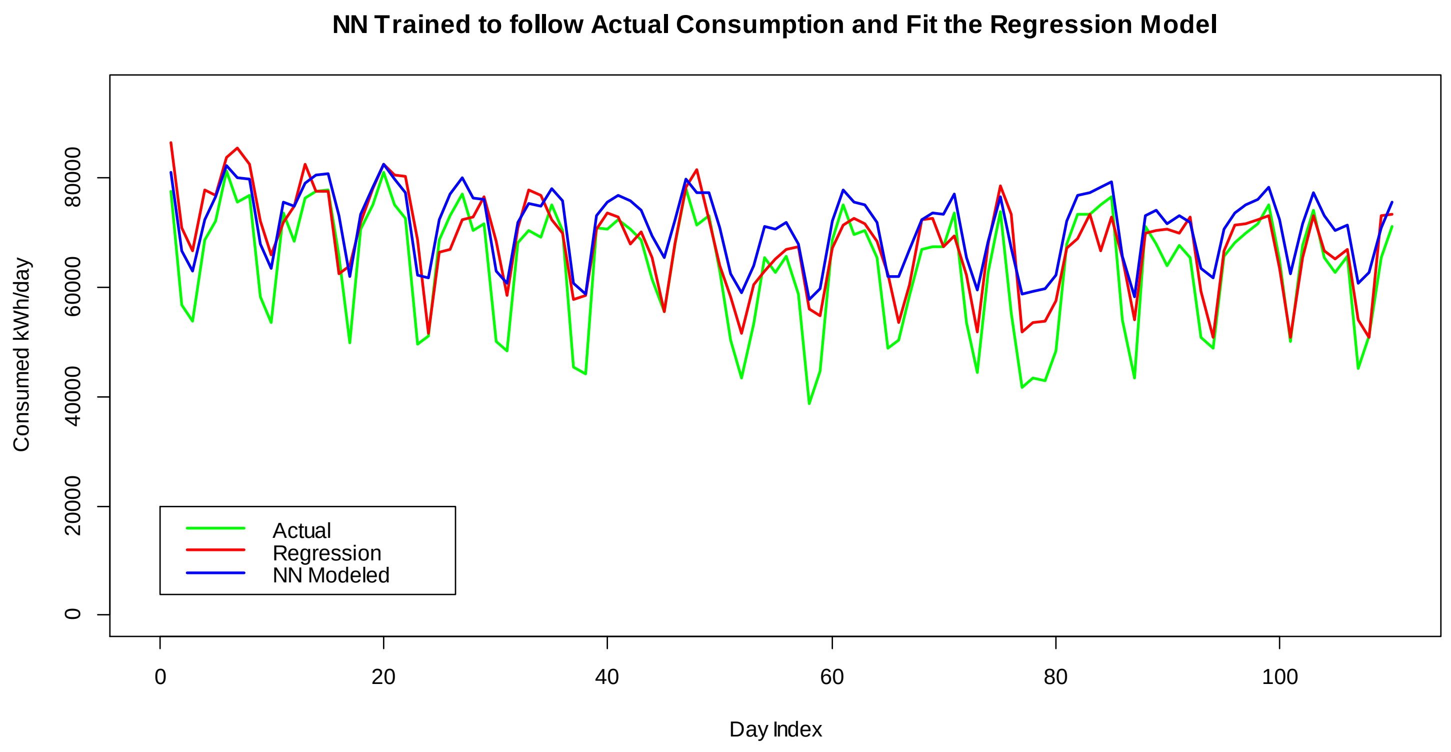
Constant	A=16066.497
Avg Temperature (°F)	B=647.173
Total Production	C=0.02
Seasonal Indicator	D=1946.042

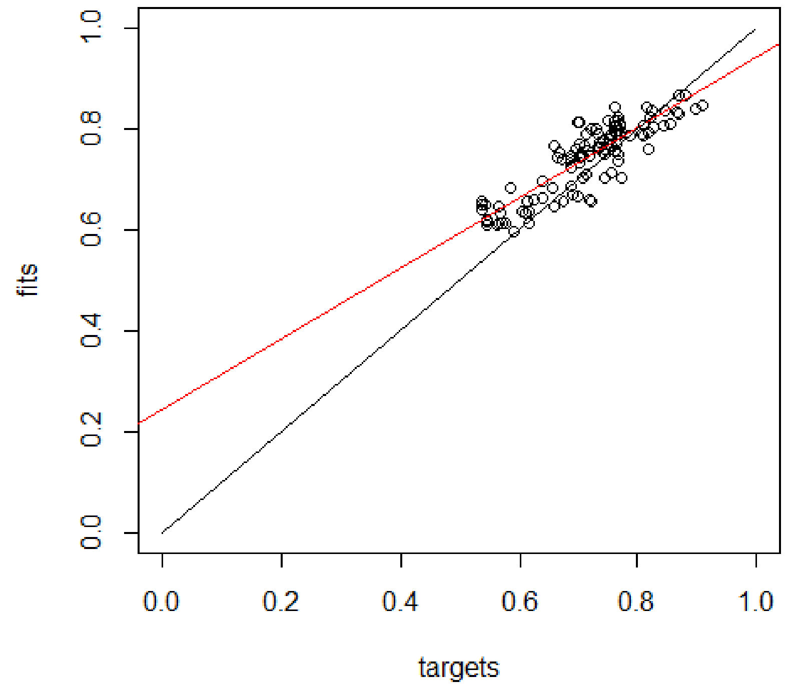
Actual Data – first 100 days



The NN was trained on input - the Actual Consumption (green) to predict (forecast) the target – the Expected Consumption (red) calculated from the linear regression equation

NN Trained to Match the Regression Model

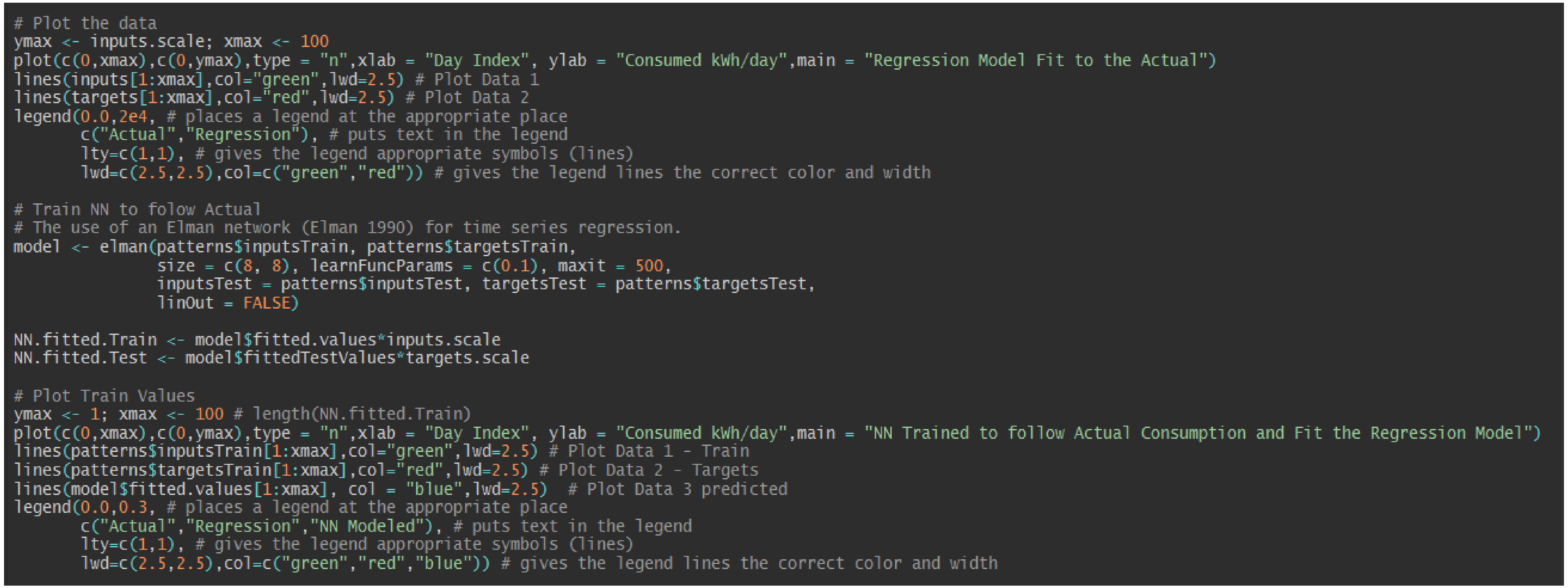




After NN being trained
The NN input is Actual Consumption (green)
The NN output (blue) is very close to calculated values from the linear regression equation (red)

NOTE: <u>The NN match is pretty good considering the temperature, seasonal, and weekend variability of the data for which the NN is not aware of, but it learned to take them into account.</u>

The Code



```r
# Plot the data
ymax <- inputs.scale; xmax <- 100
plot(c(0,xmax),c(0,ymax),type = "n",xlab = "Day Index", ylab = "Consumed kWh/day",main = "Regression Model Fit to the Actual")
lines(inputs[1:xmax],col="green",lwd=2.5) # Plot Data 1
lines(targets[1:xmax],col="red",lwd=2.5) # Plot Data 2
legend(0.0,2e4, # places a legend at the appropriate place
       c("Actual","Regression"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("green","red")) # gives the legend lines the correct color and width

# Train NN to folow Actual
# The use of an Elman network (Elman 1990) for time series regression.
model <- elman(patterns$inputsTrain, patterns$targetsTrain,
               size = c(8, 8), learnFuncParams = c(0.1), maxit = 500,
               inputsTest = patterns$inputsTest, targetsTest = patterns$targetsTest,
               linOut = FALSE)

NN.fitted.Train <- model$fitted.values*inputs.scale
NN.fitted.Test <- model$fittedTestValues*targets.scale

# Plot Train Values
ymax <- 1; xmax <- 100 # length(NN.fitted.Train)
plot(c(0,xmax),c(0,ymax),type = "n",xlab = "Day Index", ylab = "Consumed kWh/day",main = "NN Trained to follow Actual Consumption and Fit the Regression Model")
lines(patterns$inputsTrain[1:xmax],col="green",lwd=2.5) # Plot Data 1 - Train
lines(patterns$targetsTrain[1:xmax],col="red",lwd=2.5) # Plot Data 2 - Targets
lines(model$fitted.values[1:xmax], col = "blue",lwd=2.5)  # Plot Data 3 predicted
legend(0.0,0.3, # places a legend at the appropriate place
       c("Actual","Regression","NN Modeled"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("green","red","blue")) # gives the legend lines the correct color and width
```