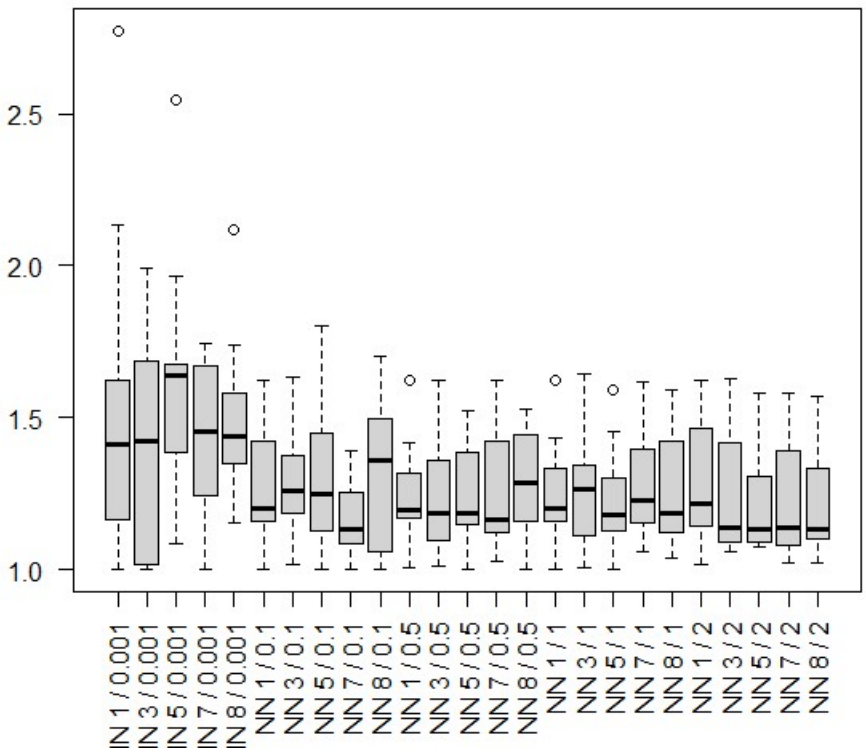
3. Now try tuning the NN *on these two variables* using 5-fold CV. Use a grid of (1, 3, 5, 7, 9) nodes and (.001, .1, .5, 1, and 2) decay. Refit each combination of parameters 10 times to find the best sMSE for that combination.
(a) Compute the overall MSPE for each combination, and add 95% confidence intervals. **Take square roots and report the results.**
(b) **Show relative root-MSPE boxplots of the five splits.**
(c) Use these results to **identify (i) the best combination, and (ii) other combinations that seem to perform just as well.**
(d) Is further tuning necessary? **That is, is the best model (i) at a boundary or (ii) quite different from neighbouring models?**

```
library(caret)

# This function will do min-max scaling internally with preProcess="range"
# Need to use y as a numeric vector and not a matrix.
# Can change numbers used in tune.Grid as needed.

# Default is 25 bootstrap reps
# Can be changed by adding
# trcon = trainControl(method=..., number=..., repeats=...)
#  Can do method= "boot", "cv", "repeatedcv", "LOOCV", and several others
#  number= is number of boot reps or cv folds
#  repeats= number of CV replicates
#  returnResamp="all" retains the error measures from each split.

#Specify 5-fold CV run twice
trcon = trainControl(method="repeatedcv", number=5, repeats=2,
                     returnResamp="all")
parmgrid = expand.grid(size=c(1,3,5,7,8),decay= c(0.001, .1, .5, 1 ,2))

tuned.nnet <- train(x=data[,-1], y=data[,1], method="nnet",
                    trace=FALSE, linout=TRUE,
                    trControl=trcon, preProcess="range",
                    tuneGrid = parmgrid)
tuned.nnet
names(tuned.nnet)
tuned.nnet$bestTune

#If I want to make plots, I need to rearrange the resamples
(resamp.caret = tuned.nnet$resample[,-c(2,3)])

library(reshape)
RMSPE.caret = reshape(resamp.caret, direction="wide", v.names="RMSE",
                      idvar=c("size","decay"), timevar="Resample")


# Plot results.
siz.dec <- paste("NN",RMSPE.caret[,1],"/",RMSPE.caret[,2])
x11(pointsize=10)
boxplot(x=as.matrix(RMSPE.caret[,-c(1,2)]), use.cols=FALSE, names=siz.dec,
        las=2, main="caret Root-MSPE boxplot for various NNs")

```

```
# Plot RELATIVE results.
lowt = apply(RMSPE.caret[,-c(1,2)], 2, min)

x11(pointsize=10)
boxplot(x=t(as.matrix(RMSPE.caret[,-c(1,2)]))/lowt, las=2,
        names=siz.dec)

#Focused
x11(pointsize=10)
boxplot(x=t(as.matrix(RMSPE.caret[,-c(1,2)]))/lowt, las=2,
        names=siz.dec, ylim=c(1,2))

R=2
V=5
relMSPE = t(RMSPE.caret[,-c(1,2)])/lowt
(RRMSPE = apply(X=relMSPE, MARGIN=2, FUN=mean))
(RRMSPE.sd = apply(X=relMSPE, MARGIN=2, FUN=sd))
RRMSPE.CIl = RRMSPE - qt(p=.975, df=R*V-1)*RRMSPE.sd/sqrt(R*V)
RRMSPE.CIu = RRMSPE + qt(p=.975, df=R*V-1)*RRMSPE.sd/sqrt(R*V)
(all.rrcv = cbind(RMSPE.caret[,1:2],round(sqrt(cbind(RRMSPE,RRMSPE.CIl, RRMSPE.CIu)),2)))
all.rrcv[order(RRMSPE),]
```



(c) Use these results to **identify (i) the best combination, and (ii) other combinations that seem to perform just as well.**

It looks like a combination with 7 nodes and 0.1 shrinkage has the best result. However, other combinations for example, 5 nodes and 1 shrinkage, 1 node and 1 shrinkage also do well.

(d) Is further tuning necessary? **That is, is the best model (i) at a boundary or (ii) quite different from neighbouring models?**

It's quite similar with some other with other neighbouring models, so might need further tuning