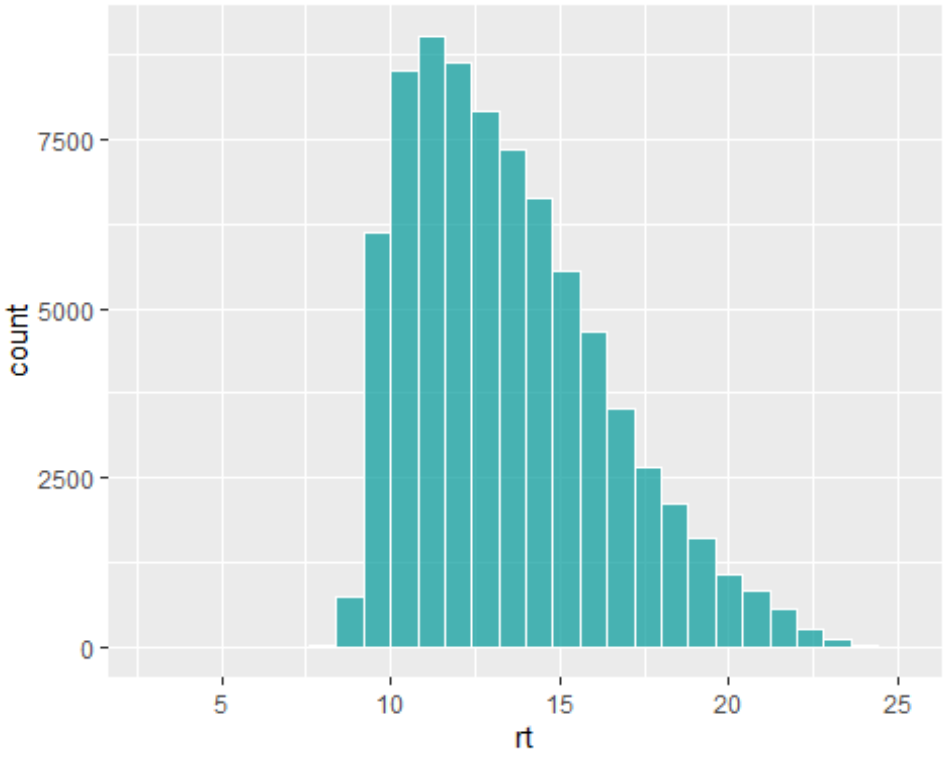
Train 1D CNN on METLIN SMRT data set

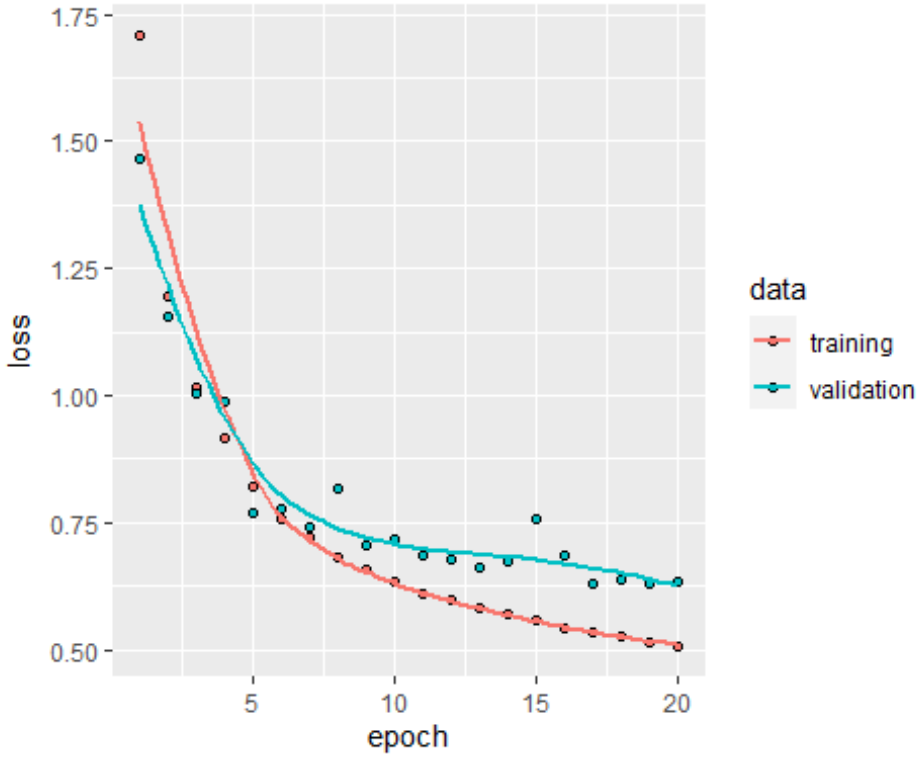
The data used in this study is taken from https://github.com/Elizachem/SMRT_transfer

```
setwd("C:/markdown")
library(caret)

library(Metrics)

##
## Attaching package: 'Metrics'

library(data.table)
library(abind)
library(keras)
library(ggplot2)
### Load data
matrix<- readRDS("matrix_abind.rda")
 dim(matrix)

## [1] 77983    93    35

retention_time <- as.data.frame(fread("SMRT_data.csv")$rt)
retention_time <- apply(retention_time,2,function(x)as.numeric(as.character(x)))
retention_time<- as.data.frame(retention_time)
names(retention_time)[1]<- "rt"
ggplot(dpi = 500, height = 90, width = 90)+
geom_histogram( data =retention_time, aes(x=rt),fill="#009999",  binwidth =
0.8,colour= "white", alpha = 0.7,  show.legend = F)
```




```
trainIndex<-sample(createDataPartition(retention_time$rt, p=0.8, list=FALSE))
ytrain = as.matrix(retention_time[trainIndex,])
ytest <- as.matrix(retention_time[-trainIndex,])
xtrain <- matrix[trainIndex,,]
xtest<- matrix[-trainIndex,,]

in_dim = c(dim(xtrain)[2:3])
print(in_dim)

## [1] 93 35

### Build model
model = keras_model_sequential() %>%
      layer_conv_1d(filters = 200, kernel_size = 11,strides =1, input_shape =
in_dim, activation = "relu")%>%
    layer_conv_1d(filters = 200, kernel_size = 9,strides = 1, activation =
"relu") %>%
  layer_global_average_pooling_1d()%>%
  layer_dense(units = 200, activation = "relu")%>%
  layer_dense(units = 1, activation = "linear")

model %>% compile(
  loss = "mae",
  optimizer <- "adam")
###TRAIN model
#Stop training when validation loss stops decreasing
early_stop <- callback_early_stopping(monitor = "val_loss", patience =7,mode =
"min",min_delta= 0.01,restore_best_weights = T)
```

```
history1 <- model%>% fit(
  xtrain,
  ytrain,
  batch_size =32,
  epochs = 20,
  validation_split = 0.05,
  callbacks = list(early_stop))

history2<- model %>% fit(xtrain, ytrain, epochs = 5, batch_size=128,
validation_split=0.05, callbacks = list(early_stop))

plot(history1, smooth.spline)

## `geom_smooth()` using formula 'y ~ x'
```




```
###Calculate Errors
mdape <- function(x,z) median(abs(x-z)/x)
test_y_pred = model %>% predict(xtest)
RMSE<- RMSE(ytest, test_y_pred)
MAE <- MAE(ytest, test_y_pred)
MdAE<- mdae(ytest, test_y_pred)
MedRE<- mdape(ytest, test_y_pred)
MRE <- mape(ytest, test_y_pred)
test_metrics <- as.data.frame(cbind(RMSE,MAE,MdAE,MRE,MedRE))

print(test_metrics)

##       RMSE       MAE      MdAE         MRE      MedRE
## 1 1.086085 0.5827213 0.3125336 0.04354273 0.02367508
```

```
plot(ytest,test_y_pred)
```