Functional Neural Network for Scalar-on-Function Regression

In this notebook, we implement and evaluate a Functional Neural Network (FNN) for predicting fat values from spectroscopic data. The FNN is built roughly following the framework outlined in Thind et al. (2023). See FuncNN GitHub Repo for the authors' own library for their experiments. The functional predictors are first preprocessed (centered and scaled) and then smoothed using a basis expansion. The network is then fitted with a chosen set of hidden layers and activation functions. In the following sections, we compare the performance of the FNN for different smoothing (penalty) parameters and also compare it with a functional linear model (i.e. no hidden layers).

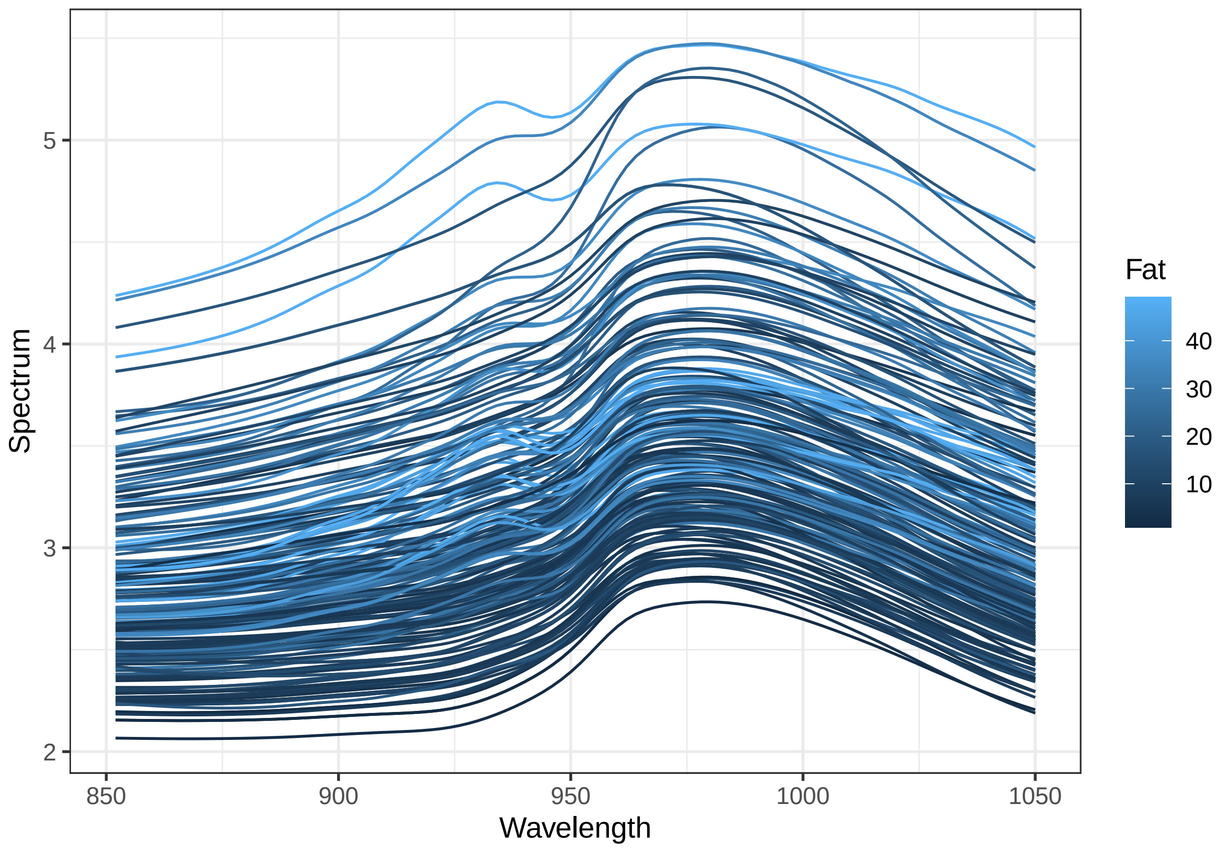
Loading Dependencies and Data

```r
# dependencies
packages <- c(
  "fda", "torch", "dplyr", "tidyr", "ggplot2"
)
installed <- packages %in% rownames(installed.packages())
if (any(!installed)) {
  install.packages(packages[!installed])
}

library(tidyr)
library(dplyr)
library(ggplot2)
library(fda)
library(torch)
source("SoFNN.R")
```

We load the `Fatspectrum` and `Fatvalues` data from the `{fds}` package. The spectroscopic data (X) and corresponding fat values (y) are reshaped into a tidy format for visualization. The first plot shows the spectrums for each subject over the observed wavelengths, colored by their fat value.

```r
data(Fatspectrum, package = "fds")
data(Fatvalues, package = "fds")

tvals = Fatspectrum$x
names(tvals) = paste0("t", seq_along(tvals))
X = t(Fatspectrum$y)
y = Fatvalues

colnames(X) = paste0("t", seq_along(tvals))
tidy_data = cbind(subj=1:nrow(X), X, Fat=y) |>
  as_tibble() |>
  pivot_longer(
    cols=paste0("t", seq_along(tvals)),
    names_to = "tid",
    values_to = "Spectrum"
  ) |>
  mutate(Wavelength = tvals[tid])
```

```r
ggplot(tidy_data, aes(x = Wavelength, y = Spectrum)) +
  geom_line(aes(color=Fat, group=subj)) +
  theme_bw()
```



Fitting the FNN

Case 1: Low Smoothing (lambda = 1e-7)

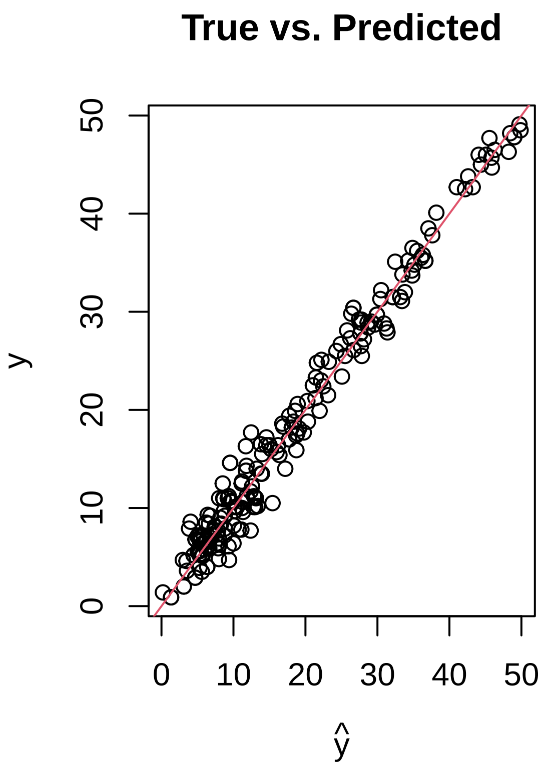
With a small lambda, little penalty is imposed. This configuration may result in a functional weight that fits the training data closely but could have larger variations.

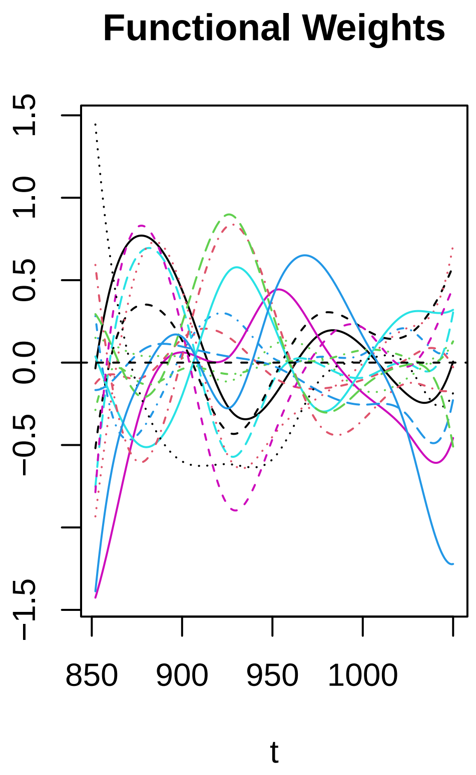
```r
sofnn = fit.sofnn(
  y, X, tgrid = tvals, lambda = 1e-7, nbasis = 11,
  hidden_sizes = c(16), act_func = "relu", max_epoch = 1000,
  verbose = FALSE)

cat(
  "Initial loss:", sofnn$loss_history[1], "\n",
  "Final loss:", tail(sofnn$loss_history, 1), "\n"
)
```

```
## Initial loss: 0.9389007
##  Final loss: 0.02086103
```

```r
par(mfrow=c(1,2))
predict(sofnn, X=X, tgrid=tvals) |>
  plot(y, main="True vs. Predicted", xlab=expression(hat(y)), ylab="y")
abline(coef=c(0,1), col=2)
```

```
plot(sofnn$func_weights[[1]], xlab="t", ylab="", main="Functional Weights")
```





```
## [1] "done"
```

```
par(mfrow=c(1,1))
```

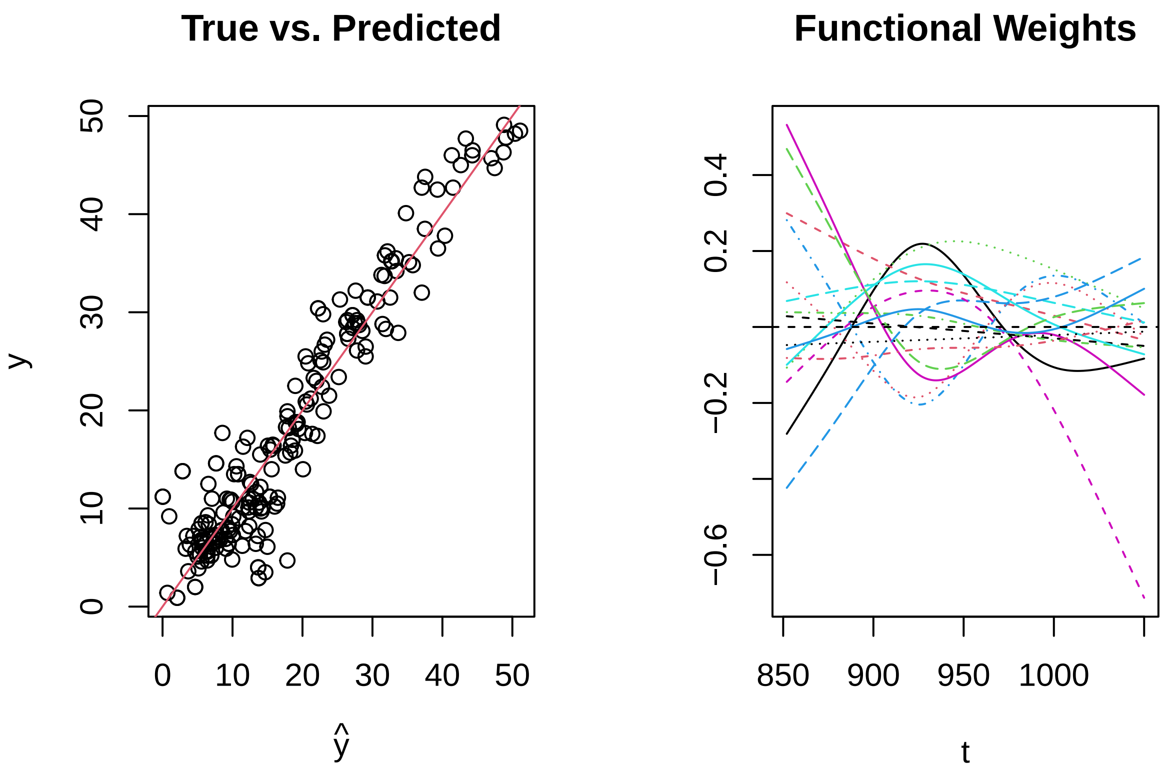
Case 2: High Smoothing (lambda = 1e3)

A very large lambda imposes a strong penalty, forcing the estimated weight function to be very smooth, which might oversmooth the relationship between the predictors and response.

```
sofnn = fit.sofnn(
  y, X, tgrid = tvals, lambda = 1e3, nbasis = 11,
  hidden_sizes = c(16), act_func = "relu", max_epoch = 1000,
  verbose = FALSE)

cat(
  "Initial loss:", sofnn$loss_history[1], "\n",
  "Final loss:", tail(sofnn$loss_history, 1), "\n"
)
```

```
## Initial loss: 4.73372
##  Final loss: 0.1211135
```

```
par(mfrow=c(1,2))
predict(sofnn, X=X, tgrid=tvals) |>
  plot(y, main="True vs. Predicted", xlab=expression(hat(y)), ylab="y")
abline(coef=c(0,1), col=2)
plot(sofnn$func_weights[[1]], xlab="t", ylab="", main="Functional Weights")
```



```
## [1] "done"
```

```
par(mfrow=c(1,1))
```

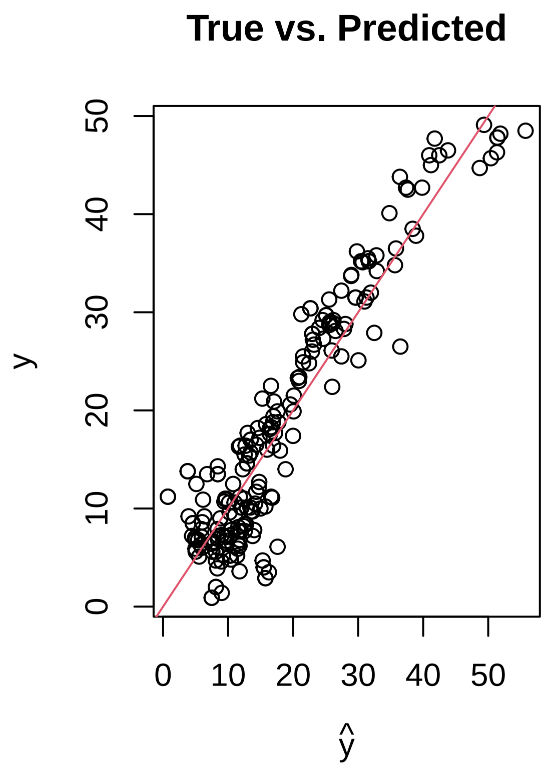
Case 3: Functional Linear Model (no hidden layers)

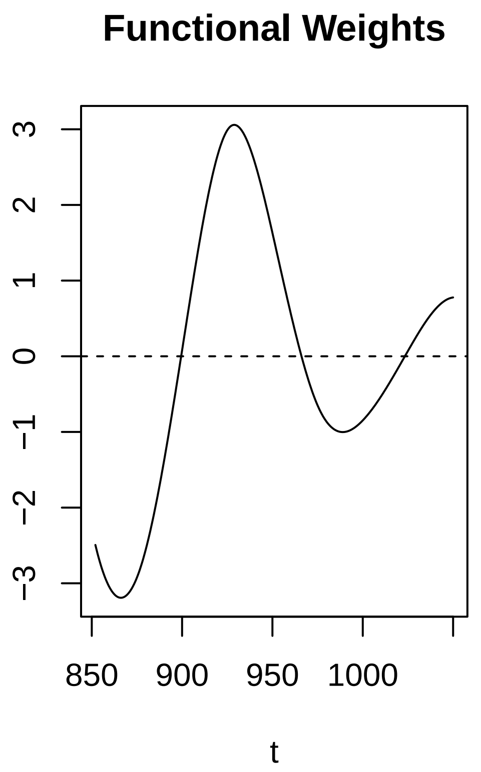
By setting `hidden_sizes` to an empty vector, the model is reduced to a functional linear model. This serves as a baseline for comparison against the nonlinear FNN.

```
sofnn = fit.sofnn(
  y, X, tgrid = tvals, lambda = 1e-3, nbasis = 11,
  hidden_sizes = c(), act_func = "relu", max_epoch = 1000,
  verbose = FALSE)

cat(
  "Initial loss:", sofnn$loss_history[1], "\n",
  "Final loss:", tail(sofnn$loss_history, 1), "\n"
)
```

```
## Initial loss: 0.8086581
##  Final loss: 0.1281526
```

```
par(mfrow=c(1,2))
predict(sofnn, X=X, tgrid=tvals) |>
  plot(y, main="True vs. Predicted", xlab=expression(hat(y)), ylab="y")
abline(coef=c(0,1), col=2)
plot(sofnn$func_weights[[1]], xlab="t", ylab="", main="Functional Weights")
```





```
## [1] "done"
```

```r
par(mfrow=c(1,1))
```