Spectral data of fossil fuels: scalar-on-function regression

Sarah Brockhaus[1]

[1] Institut für Statistik
Ludwig-Maximilians-Universität München
Ludwigstraße 33, D-80539 München, Germany
sarah.brockhaus@stat.uni-muenchen.de

The dataset was originally analyzed by Fuchs et al. (2015). The results of this vignette together with more explanations can be found in Brockhaus et al. (2015).

1 Load and plot data

Load FDboost package.

Load data and compute the first derivative.

```
> data(fuelSubset)
> fuel <- fuelSubset
> str(fuel)
```

```
List of 7
 $ heatan      : num [1:129] 26.8 27.5 23.8 18.2 17.5 ...
 $ h2o         : num [1:129] 2.3 3 2 1.85 2.39 ...
 $ nir.lambda  : num [1:231] 800 803 805 808 810 ...
 $ NIR         : num [1:129, 1:231] 0.2818 0.2916 -0.0042 -0.034 -0.1804 ...
 $ uvvis.lambda: num [1:134] 250 256 261 267 273 ...
 $ UVVIS       : num [1:129, 1:134] 0.145 -1.584 -0.814 -1.311 -1.373 ...
 $ h2o.fit     : num [1:129] 2.58 3.43 1.83 2.03 3.07 ...
```

```
> # # normalize the wavelength to 0-1
> # fuel$nir.lambda0 <- (fuel$nir.lambda - min(fuel$nir.lambda)) /
> #   (max(fuel$nir.lambda) - min(fuel$nir.lambda))
> # fuel$uvvis.lambda0 <- (fuel$uvvis.lambda - min(fuel$uvvis.lambda)) /
> #   (max(fuel$uvvis.lambda) - min(fuel$uvvis.lambda))
>
> # compute first derivatives as first order differences
> fuel$dUVVIS <- t(apply(fuel$UVVIS, 1, diff))
> fuel$dNIR <- t(apply(fuel$NIR, 1, diff))
> # get the wavelength for the derivatives
```

```
> fuel$duvvis.lambda <- fuel$uvvis.lambda[-1]
> fuel$dnir.lambda <- fuel$nir.lambda[-1]
> # fuel$duvvis.lambda0 <- fuel$uvvis.lambda0[-1]
> # fuel$dnir.lambda0 <- fuel$nir.lambda0[-1]
```

Compute the model to predict humidity. The predicted humidity is contained already in the dataset *fuel*.

2 Model to predict humidity

We consider the following regression model to predict the humidity.

$$E(Y_i) = \int \text{NIR}_i(s_1)\beta_1(s_1)ds_1 + \int \text{UVVIS}_i(s_2)\beta_2(s_2)ds_2 + \int \text{dNIR}_i(s_3)\beta_1(s_3)ds_3 + \int \text{dUVVIS}_i(s_4)\beta_2(s_4)ds_4,$$

with Y_i being the humidity and NIR, UVVIS are the spectra and dNIR, dUVVIS the respective derivatives, measured over $s_1, \ldots, s_4$ respectively. The optimal stopping iteration is determined by 10-fold bootstrap.

```
> modH2O <- FDboost(h2o ~ bsignal(UVVIS, uvvis.lambda, knots=40, df=4)
+                     + bsignal(NIR, nir.lambda, knots=40, df=4)
+                     + bsignal(dUVVIS, duvvis.lambda, knots=40, df=4)
+                     + bsignal(dNIR, dnir.lambda, knots=40, df=4),
+                     timeformula=~bols(1), data=fuel)
> set.seed(212)
> cvmH2O <- suppressWarnings(cvrisk(modH2O, grid=seq(100, 5000, by=100),
+                                   folds=cv( model.weights(modH2O),
+                                   type = "bootstrap", B = 10), mc.cores=10))
> par(mfrow=c(1,2))
> plot(cvmH2O)
> modH2O[mstop(cvmH2O)]
> #modH2O[2400]
>
> #### create new variable of predicted h2o
> h2o.fit <- modH2O$fitted()
> plot(fuel$h2o, h2o.fit)
> abline(0,1)
```

3 Model to predict heat value

We consider the following regression model to predict the heat values.

$$E(Y_i) = \int \text{NIR}_i(s_1)\beta_1(s_1)ds_1 + \int \text{UVVIS}_i(s_2)\beta_2(s_2)ds_2,$$

with Y_i being the heat value and NIR and UVVIS are the spectra, measured over s_1 and s_2 respectively.

```
> formula <- formula(heatan ~ bsignal(UVVIS, uvvis.lambda, knots=40, df=4.41)
+                    + bsignal(NIR, nir.lambda, knots=40, df=4.41))
> ## do a model fit:
> mod <- FDboost(formula, timeformula=~bols(1), data=fuel)
> mod <- mod[198]
```

The optimal stopping iteration is determined by 50-fold bootstrap. We compute in each bootstrap-sample the coefficient functions to get an idea of the variability of the estimates.

```
> ## get optimal mstop and do bootstrapping for coefficient estimates
> set.seed(2703)
> val <- validateFDboost(mod,
+                        folds=cv(model.weights(mod), type = "bootstrap", B = 50),
+                        grid = 10:500, mc.cores=10)
> mopt <- val$grid[which.min(colMeans(val$oobrisk))]
> print(mopt)
> ## use optimal mstop
> mod <- mod[mopt] # 198
```

Plot the coefficient functions.

References

Brockhaus S, Scheipl, F., Hothor, T., and Greven, S. (2015), The functional linear array model, *Statistical Modelling*, 15(3), 279–300.

Fuchs, K., Scheipl, F., and Greven, S. (2015), Penalized scalar-on-functions regression with interaction term, *Computational Statistics and Data Analysis*, 81, 38–51.

```
> par(mfrow=c(1,2))
> plot(mod, which=1, lwd=2, lty=5, rug=FALSE,
+      ylab="", xlab="wavelength [nm]")
> plot(mod, which=2, lwd=2, lty=5, rug=FALSE,
+      ylab="", xlab="wavelength [nm]")
> # plot with bootstrapped coefficient functions
> if(FALSE){
+   pdf("spec_valCoef.pdf")
+   par(mar=c(5, 3, 1, 1), cex.axis=1.5, cex.lab=1.5)
+   plot(mod, which=1, lwd=2, col="white", main="", lty=5, rug=FALSE,
+      ylab="", xlab="wavelength [nm]", ylim=range(val$coefCV[[1]]$value) )
+   plotPredCoef(val, terms=FALSE, commonRange=TRUE, which=1, add=TRUE)
+   plot(mod, which=1, lwd=2, col=4, main="", lty=5, rug=FALSE, add=TRUE)
+
+   legend("topright", legend=c("data", "BS", "mean BS", "5, 95% BS"),
+        lty=c(5,1,1,2), col=c(4,8,1,2), cex=1.5,
+        lwd=c(2,1,2,2))
+
+   plot(mod, which=2, lwd=2, col="white", main="", lty=5, rug=FALSE,
+      ylab="", xlab="wavelength [nm]", ylim=range(val$coefCV[[2]]$value) )
+   plotPredCoef(val, terms=FALSE, commonRange=TRUE, which=2, add=TRUE)
+   plot(mod, which=2, lwd=2, col=4, main="", lty=5, rug=FALSE, add=TRUE)
+   dev.off()
+ }
```

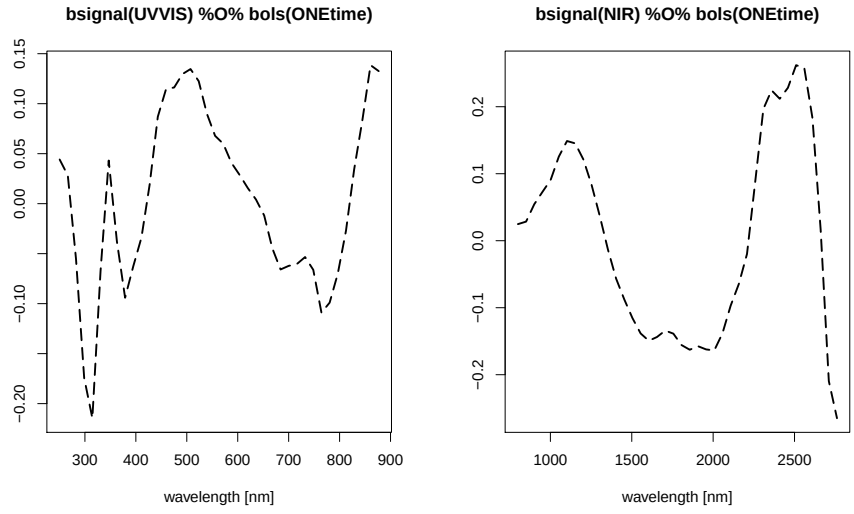

Figure 1: Coefficient estimates for the effects of the two spectra.