Model-based boosting in R

Case study 1: birth weight prediction for small fetuses

Andreas Mayr

andreas.mayr@imbe.med.uni-erlangen.de
Institut für Medizininformatik, Biometrie und Epidemiologie (IMBE)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Statistical Computing 2011

Aims and scope

- The task is to find a prediction formula for birth weight at delivery for small fetuses ($\leq 1600g$), based on 3D (left) and 2D (right) ultrasound measurements about one week before delivery.

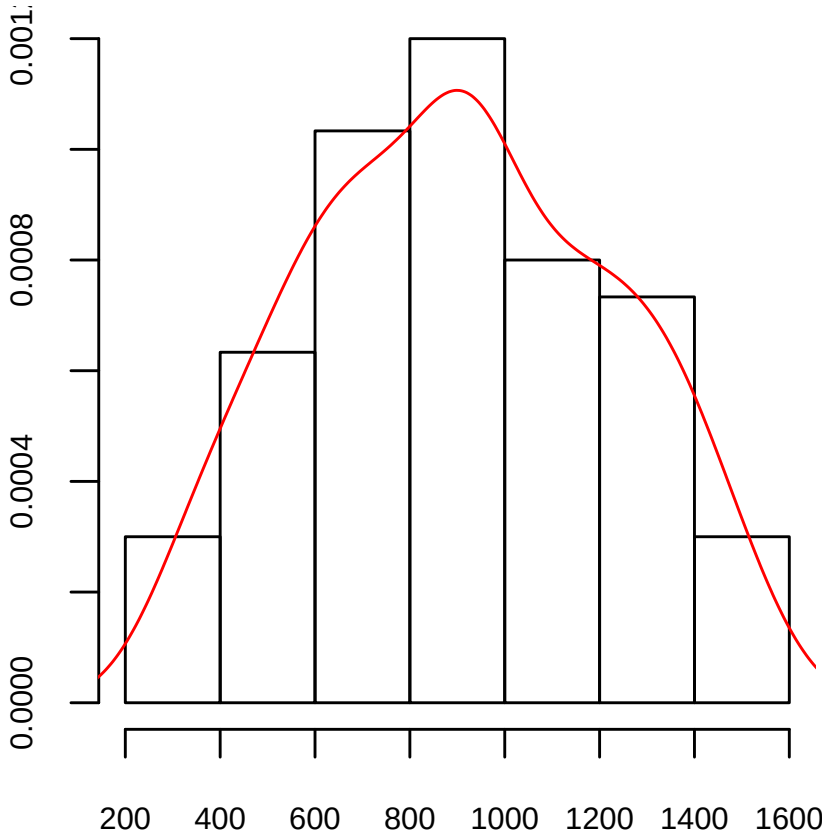
Sources: www.yourultrasound.com, www.fetalultrasoundutah.com

- As many decision-rules concerning the appropriate steps to be taken in case of extremely small newborns depend on their actual birth weight, an accurate estimation **before** delivery is of high clinical relevance.

Data

- Outcome: Birth weight (`birthw`) from $n = 150$ fetuses.

```
R> dat <- read.table("birthweight.txt", header=TRUE, sep="\t")
R> hist(dat$birthw, prob=TRUE,  ylab="", main="")
R> lines(density(dat$birthw), col=2)
```



Data

- Predictors: 8 biometric measurements that are highly correlated.

```
R> round(cor(dat[,-1]), 3)
```

	volARM	volFEM	volABDO	bpd	hc	atd	fl	ac
volARM	1.000	0.916	0.867	0.775	0.800	0.786	0.859	0.830
volFEM	0.916	1.000	0.895	0.784	0.823	0.803	0.879	0.863
volABDO	0.867	0.895	1.000	0.785	0.844	0.907	0.873	0.942
bpd	0.775	0.784	0.785	1.000	0.942	0.723	0.827	0.743
hc	0.800	0.823	0.844	0.942	1.000	0.774	0.883	0.791
atd	0.786	0.803	0.907	0.723	0.774	1.000	0.833	0.955
fl	0.859	0.879	0.873	0.827	0.883	0.833	1.000	0.869
ac	0.830	0.863	0.942	0.743	0.791	0.955	0.869	1.000

Data

- To evaluate predictive performance, we split the data frame in training and test data.

```
R> set.seed(0804)
R> random.split <- sample(1:nrow(dat), 100)
R> dat.train <- dat[random.split,]
R> dat.test <- dat[-random.split,]
R> nrow(dat.train)
```

```
[1] 100
```

```
R> nrow(dat.test)
```

```
[1] 50
```

Boosting formula

- We let the algorithm decide if a variable enters with linear or non-linear effect, so every predictor enters with `bols(...,intercept=FALSE)` and `bbs(...,center=TRUE,df=1)` – the latter by `center=TRUE` fits only the deviation from the linear effect.

```
R> formula1 <- birthw ~ bols(INT, intercept=FALSE)  +
+    bols(volARM, intercept=FALSE) + bbs(volARM, center=TRUE, df=1) +
+    bols(volFEM, intercept=FALSE) + bbs(volFEM, center=TRUE, df=1) +
+    bols(volABDO, intercept=FALSE)+ bbs(volABDO, center=TRUE, df=1)+
+    bols(bpd, intercept=FALSE)    + bbs(bpd, center=TRUE, df=1)    +
+    bols(hc, intercept=FALSE)     + bbs(hc, center=TRUE, df=1)     +
+    bols(atd, intercept=FALSE)    + bbs(atd, center=TRUE, df=1)    +
+    bols(fl, intercept=FALSE)     + bbs(fl, center=TRUE, df=1)     +
+    bols(ac, intercept=FALSE)     + bbs(ac, center=TRUE, df=1)
```

- **Warning:** With `intercept=FALSE` the covariates have to be mean centered beforehand, and a separate intercept should be included.

Boosting

- First fit the model with 500 iterations.

```
R> library(mboost)
R> m1 <- gamboost(formula1, control=boost_control(mstop=500),
+                 data=dat.train)
```

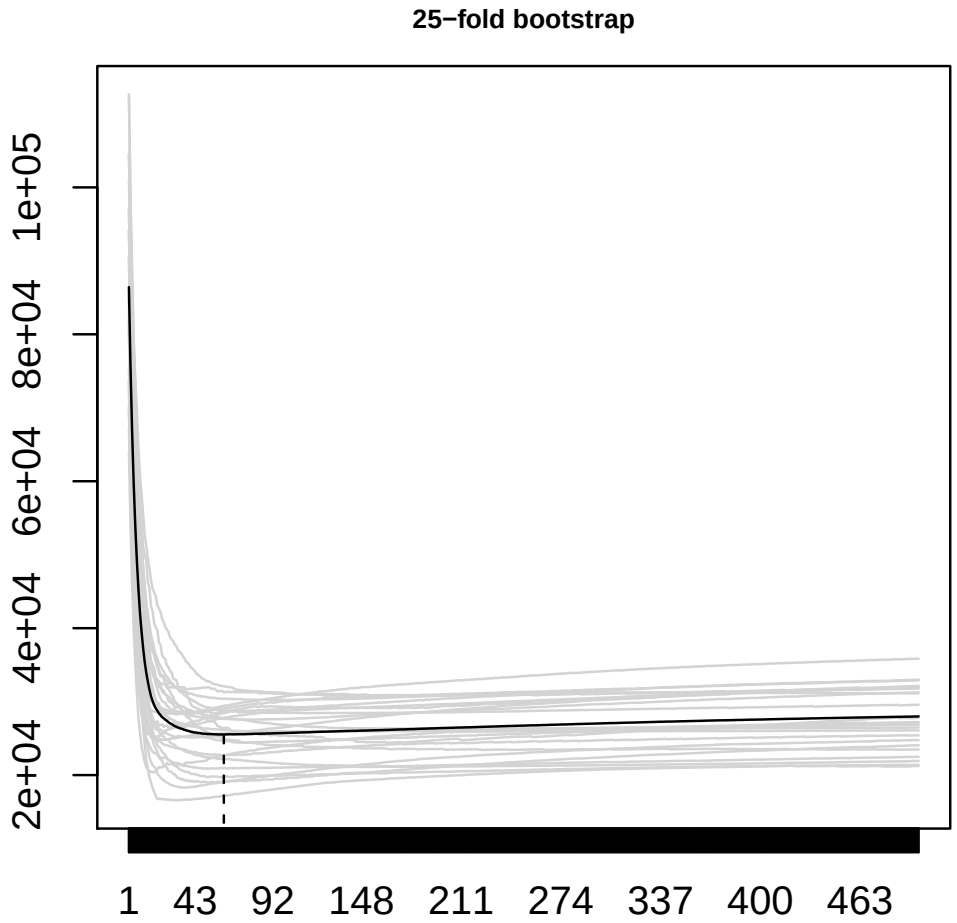
- Then optimize `mstop` by 25-fold bootstrap.

```
R> set.seed(0804)
R> cvr <- cvrisk(m1, method="bootstrap", B=25)
R> plot(cvr)
```

- If package `multicore` is not available, `cvrisk()` delivers a warning message, but automatically uses `lapply()` instead of `mclapply()`.

Boosting: 25-fold bootstrap

```
R> mstop(cvr)
[1] 61
R> plot(cvr)
```



Boosting: early stopping

- We can now go over to the prediction-optimized model (iteration 61):

```
R> m1[61]
```

- **Warning:** This subsets the model-object directly. However, the later iterations are not lost.
- Compute the MSE on the test data:

```
R> mean((dat.test$birthw - predict(m1, newdata=dat.test))^2)

[1] 22577.94
```

- Compare to the model with 500 iterations (without new fitting):

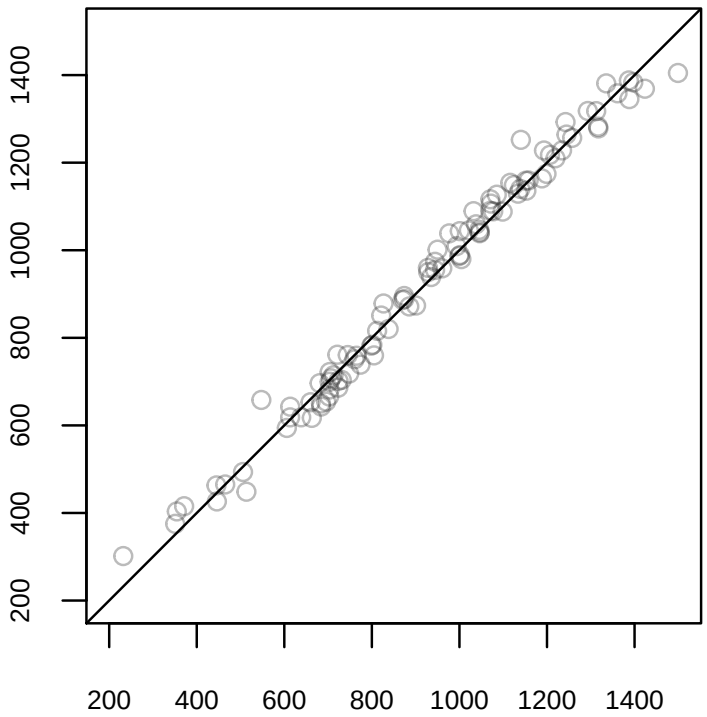
```
R> mean((dat.test$birthw - predict(m1[500], newdata=dat.test))^2)

[1] 25390.07
```

▷ Early stopping incorporates shrinkage and tends to produce a more stable solution leading to an improved prediction accuracy.

Classical GAM

- Compare to classical gam() of package mgcv:

```
R> library(mgcv)
R> gam1 <- gam(birthw ~ s(volARM) + s(volFEM) + s(volABDO) + s(bpd) +
+                s(hc) + s(atd) + s(fl) + s(ac) , data=dat.train)
R> mean((dat.test$birthw - predict(gam1, newdata=dat.test))^2)
[1] 25104.72
R> plot(fitted(gam1), fitted(m1[500]))
R> abline(a=0,b=1)
```



New prediction formula

- In order to derive a new prediction formula for small fetuses, we use boosting again to fit the model on the whole data set.

```
R> m2 <-gamboost(formula1, control=boost_control(mstop=500), data=dat)
```

- 25-fold bootstrap to find the optimal `mstop`

```
R> cvr <- cvrisk(m2, method="bootstrap", B=25)
R> mstop(cvr)
```

```
[1] 74
```

- Subset the model to 74 iterations

```
R> m2 <- m2[74]
```

Coefficients

- coef() returns a list with the effect estimates for every selected base-learner:

```
R> summary(coef(m2))
```

```
                                        Length Class  Mode
bols(volFEM, intercept = FALSE)          1     -none- numeric
bbs(volFEM, df = 1, center = TRUE)      22     -none- numeric
bols(volABDO, intercept = FALSE)         1     -none- numeric
bbs(bpd, df = 1, center = TRUE)         22     -none- numeric
bols(hc, intercept = FALSE)              1     -none- numeric
bols(atd, intercept = FALSE)             1     -none- numeric
bols(fl, intercept = FALSE)              1     -none- numeric
```

- ▷ 7 out of 16 base-learners were selected

Coefficients

- By the argument `which` one can specify the covariates for which the corresponding base-learners should be displayed.

```
R> coef(m2, which = "volFEM")
```

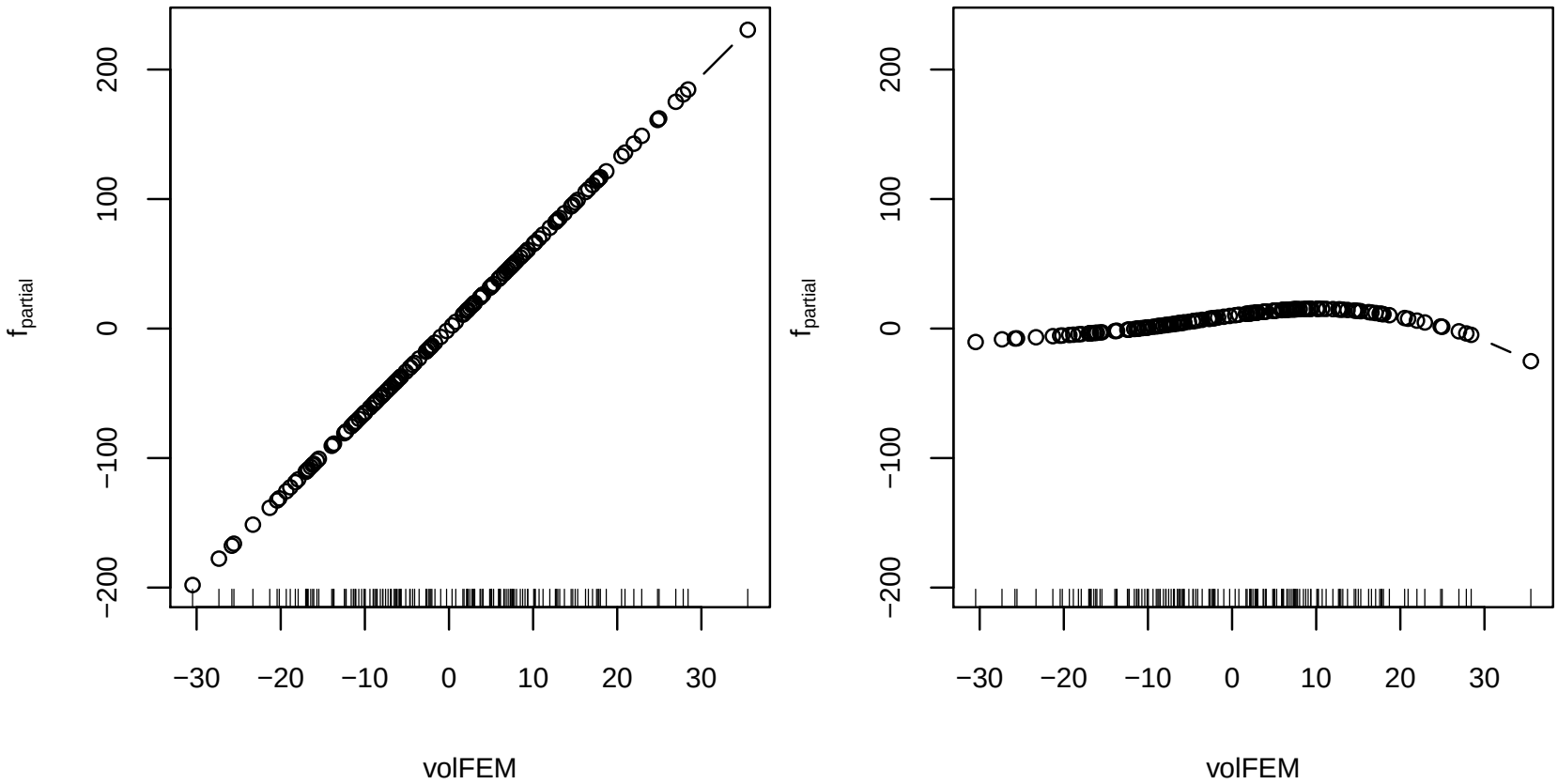
```
$`bols(volFEM, intercept = FALSE)`
volFEM
6.4951

$`bbs(volFEM, df = 1, center = TRUE)`
 [1]  0.01403404  0.05759585  0.14094561  0.22551067
 [5]  0.27797653  0.29629063  0.42675381  0.43261550
 [9]  0.09541524 -0.24586145 -0.32536456 -0.76574777
[13] -1.21455005 -1.30795046 -1.30901870 -1.19698471
[17] -1.07228261 -0.87196582 -0.57729940 -0.31176622
[21] -0.15888143 -0.05401937

attr(,"offset")
[1] 905.03
```

- To display all base-learners use `coef(m2, which = " ")`

Partial plots

- By default, plots of `mboost` objects display the partial effects of single base-learners

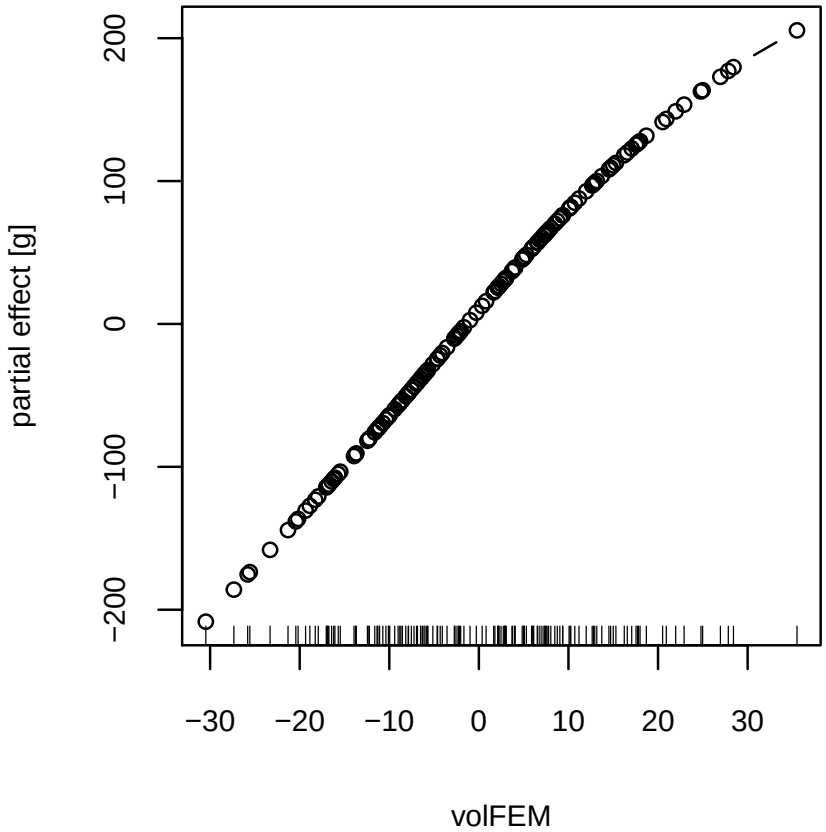
```
R> par(mfrow=c(1,2))
R> plot(m2, which="volFEM")
```



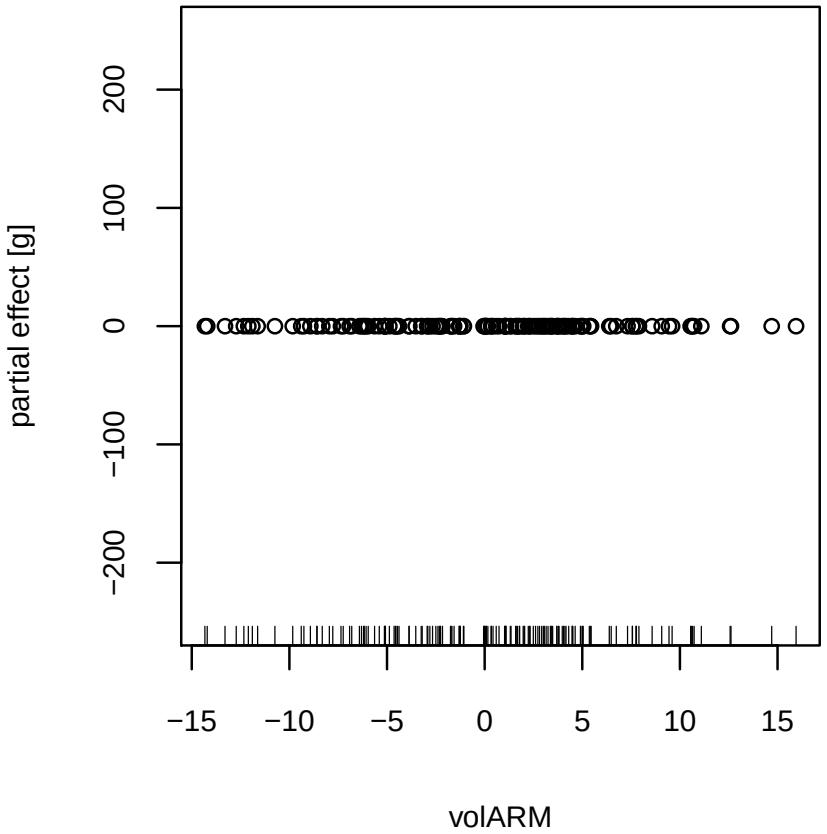
Partial plots

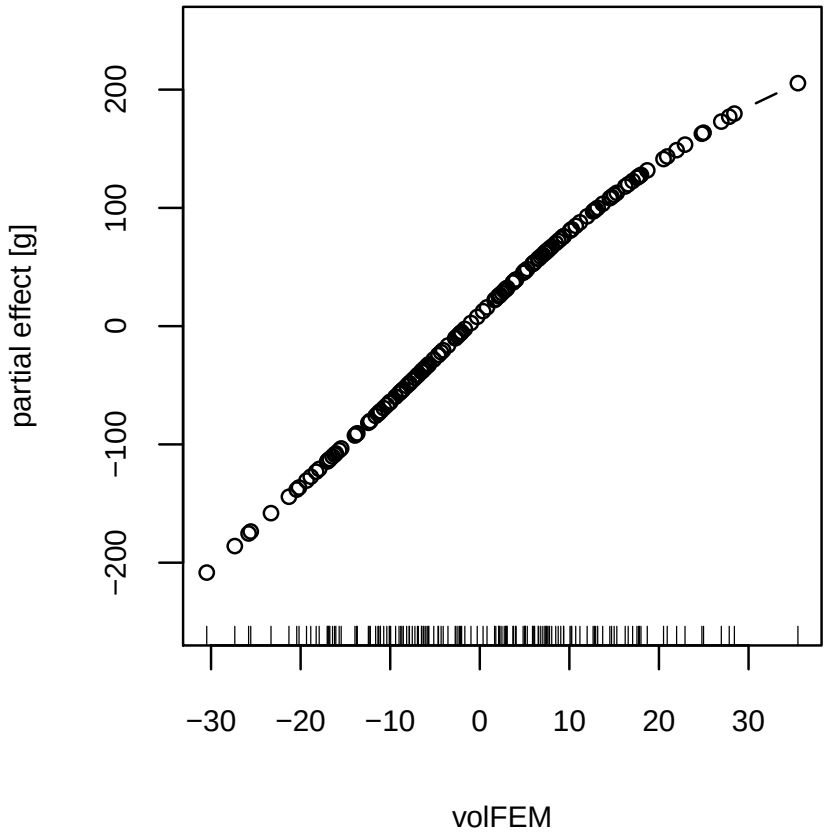
- To combine the effect of two base-learners depending on the same covariate, one can use `predict()` with `rowSums()`:

```
R> plot(dat$volFEM, rowSums(predict(m2, which="volFEM")),
+       ylab = "partial effect [g]", xlab="volFEM", type="b")
```



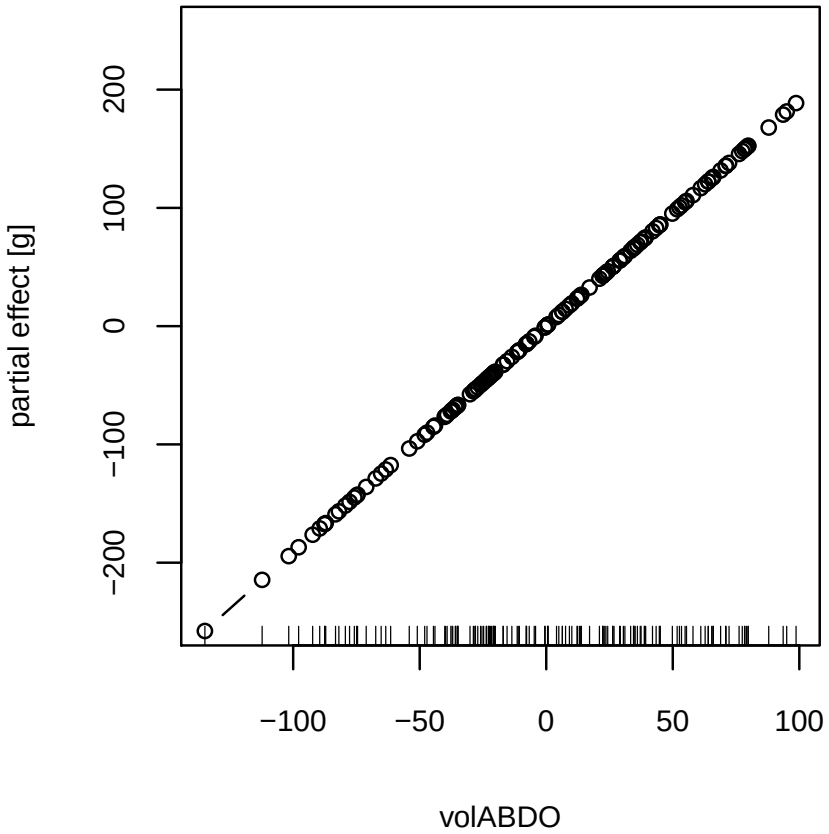
Partial plots for all predictors

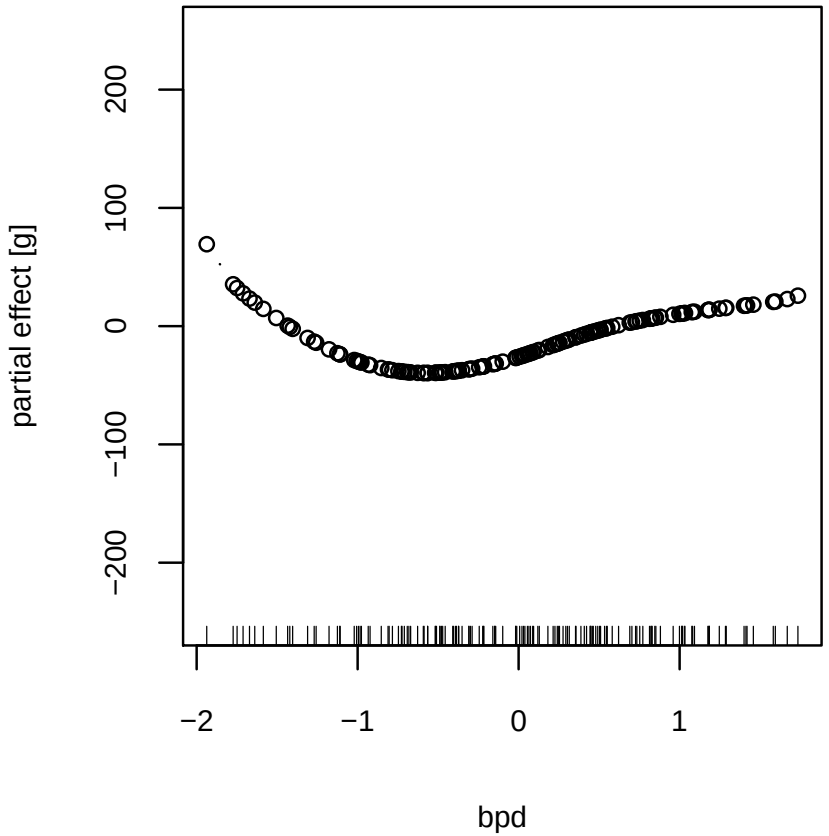




- No effect
- (Quasi) linear effect

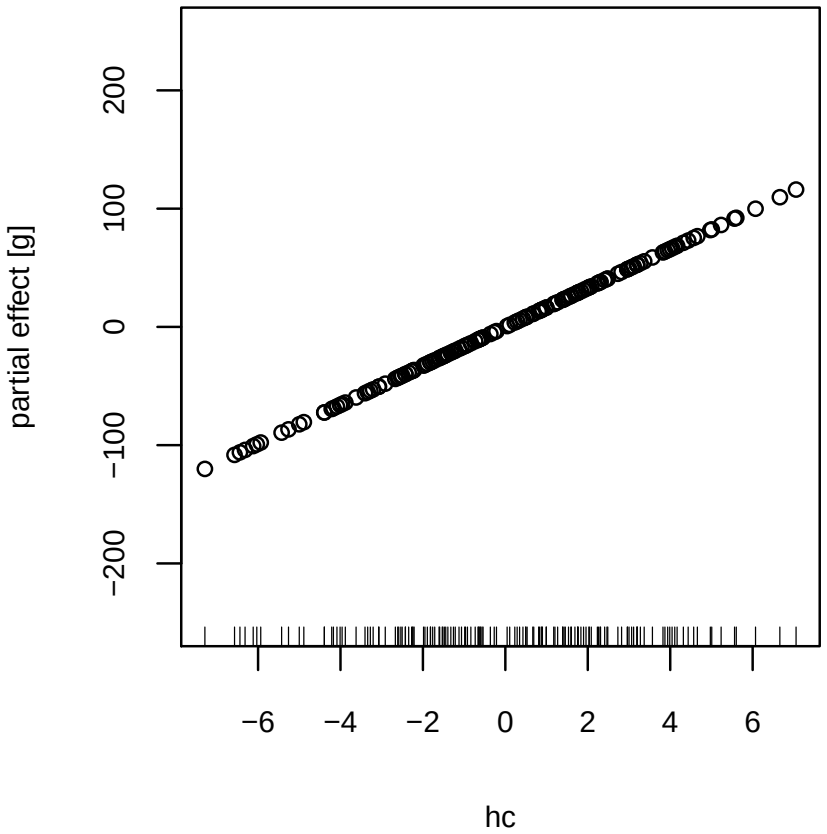
Partial plots for all predictors

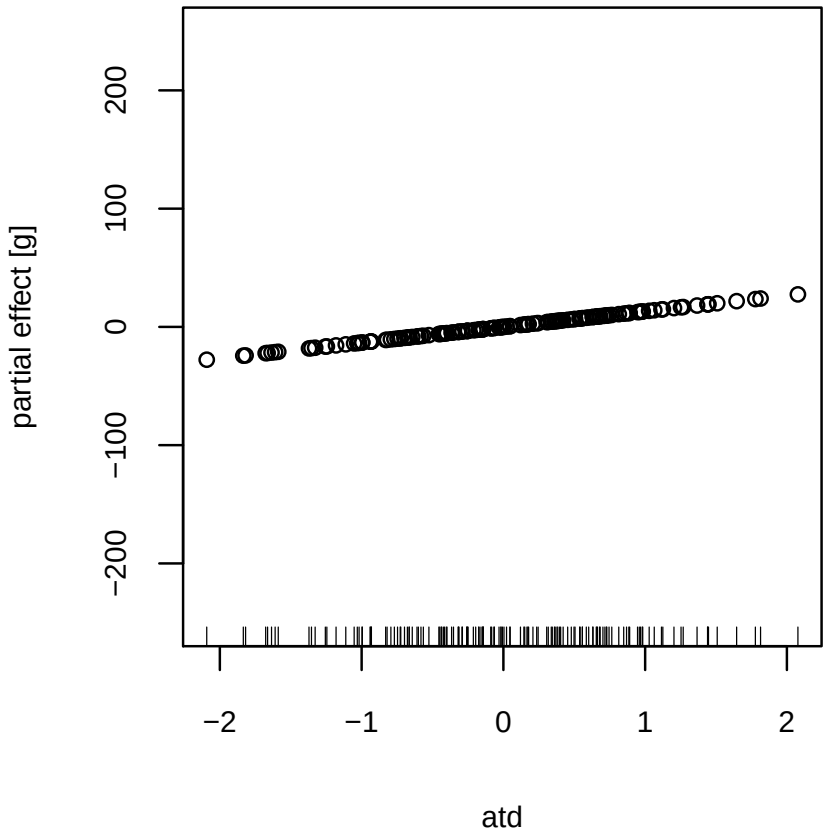




- Linear effect
- (Quasi) quadratic

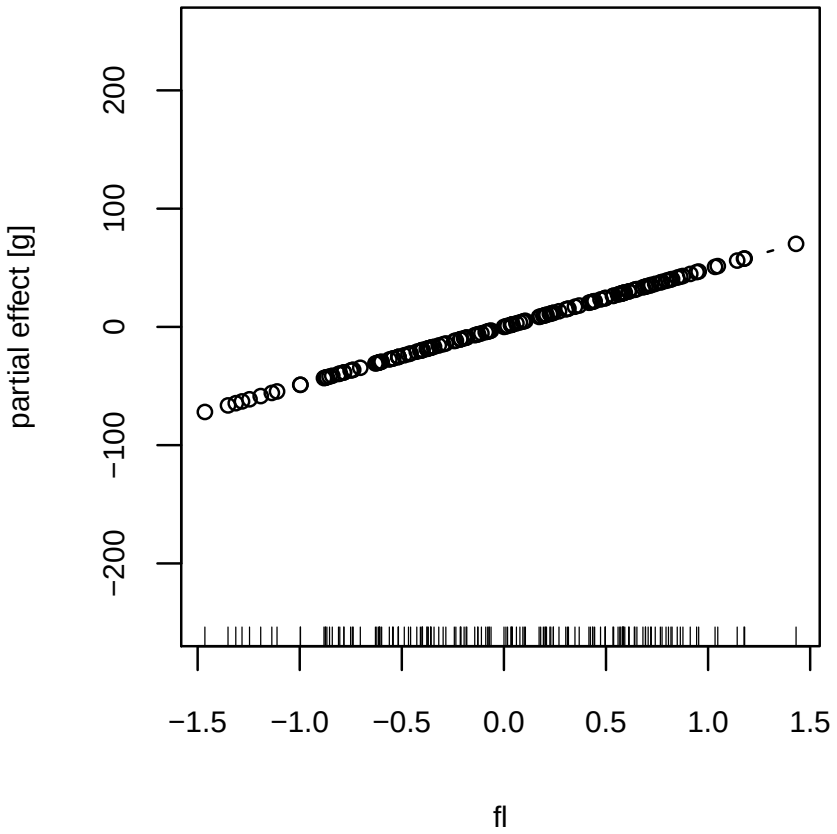
Partial plots for all predictors

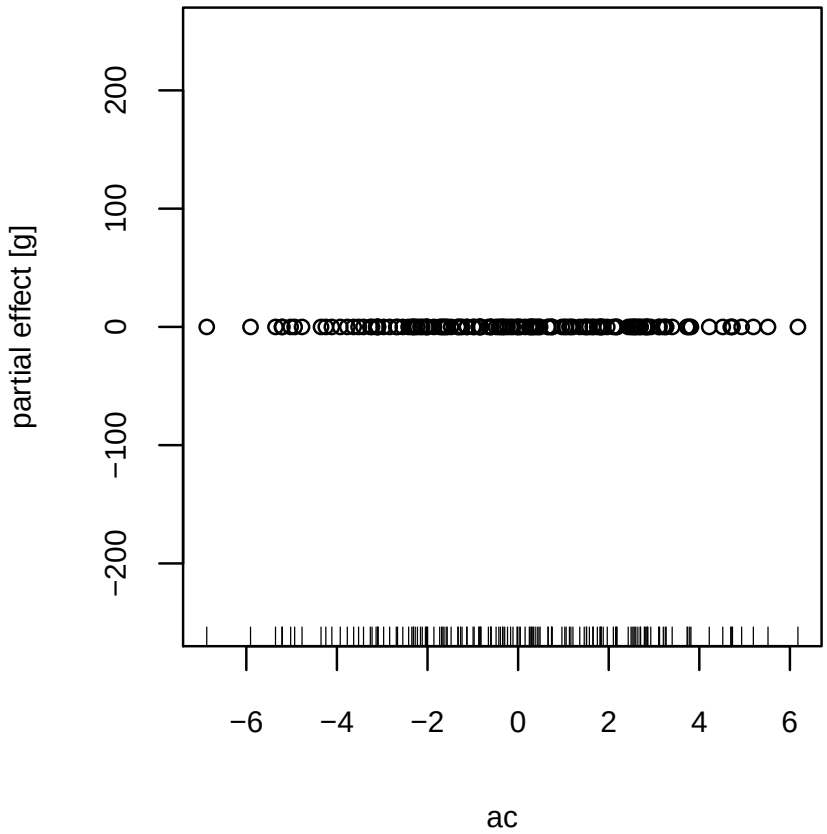




- Linear effect
- Small linear effect

Partial plots for all predictors





- Linear effect
- No effect

Results[1]

- Five linear effects
- One quadratic effect

Final model (Schild *et al.* 2008):

$$\begin{aligned}\widehat{\text{fetal weight}} = \; & 656.41 + 1.8321 \cdot \texttt{volABDO} + 31.1981 \cdot \texttt{hc} \\ & + 5.7787 \cdot \texttt{volFEM} + 73.5214 \cdot \texttt{fl} + 8.3009 \cdot \texttt{ac} \\ & + 444.8863 \cdot \texttt{bpd} + 32.5340 \cdot \texttt{bpd}^2\end{aligned}$$

Schild, R. L., M. Maringa, J. Siemer, B. Meurer, N. Hart, T. W. Goecke, M. Schmid, T. Hothorn, and M. E. Hansmann (2008). *Weight estimation by three-dimensional ultrasound imaging in the small fetus*. Ultrasound in Obstetrics and Gynecology 32, 168-175.

[1] Minor differences to original publication do occur because a bootstrapped data subset is used.