MARGE: Vignette using the `marge` R-package

Jakub Stoklosa & David I. Warton

05 September, 2021

This vignette documents fitting multivariate adaptive regression splines using generalized estimating equations (MARGE) via the `marge` R-package. We also fit some additional models and compare them with MARGE. In the second example we compare the predictive squared error (PSE) on test data, and report the computational times for each model. For specific details on the MARGE algorithm see:

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, pp. 245–253.

Also, see `?marge` for further details on the fitting function, input arguments and output.

Load the `marge`, `mgcv` and `BayesTree` R-packages.

```
suppressMessages(library("marge"))
suppressMessages(library("mgcv"))
suppressMessages(library("BayesTree"))
```

Example 1: Tree growth data.

Our first example uses the longitudinal tree growth multivariate data from Wood (2006). Fourteen trees ($N = 14$) had their heights measured across six ages ($n = 6$). The response variable is the tree height and we use age as a predictor. We also take logarithms of the response variable and assume the distribution is Gaussian. Below, we fit a GLMM and MARGE models with several different working correlation structures.

Load the data.

```
N <- 14                           # Sample size (number of clusters).
n <- 6                            # Cluster size.
id <- factor(rep(1:N, each = n))  # The ID of each cluster.

Y0 <- matrix(log(Loblolly$height), ncol = n, byrow = TRUE)   # Response variable.
Y <- c(t(Y0))
X_pred0 <- matrix(log(Loblolly$age), ncol = n, byrow = TRUE) # Predictor variable.
X_pred <- scale(c(t(X_pred0)))    # Standardize the predictor.
colnames(X_pred) <- c("age")

dat1 <- as.data.frame(cbind(X_pred, c(t(Y)), id))
colnames(dat1) <- c("age", "Y", "id")
dat1$age1 <- c(t(X_pred0))
dat1$id <- factor(dat1$id)
```

Set the required arguments and tuning parameters to fit models.

```
family <- "gaussian"    # The selected "exponential" family for the GLM/GEE.

is.gee <- TRUE          # Is the model a GEE?
nb <- FALSE             # Is this a negative binomial model?

tols_score <- 0.00001   # Tolerance for stopping condition in forward pass for GEE.
M <- 21                 # Max. no. of terms.
pen <- 2                # Penalty to be used in GCV.
minspan <- NULL         # A set minimum span value.
print.disp <- FALSE     # Print ALL the output?
```

Start model fitting functions here.

```
# GAMM/cubic mixed effect model.

dat2 <- groupedData(Y ~ age|id, data = dat1)

model_gamm <- lme(Y ~ age + I(age^2) + I(age^3), data = dat2,
                  correlation = corAR1(form =  ~ age|id), random =  ~ 1)

# MARGE.

corstr <- "independence"   # Independent working correlation structure.
model_marge_ind <- marge(X_pred, Y, N, n, id, family, corstr, pen, tols_score,
                         M, minspan, print.disp, nb, is.gee)
```

```
corstr <- "ar1"            # AR1 working correlation structure.
model_marge_ar1 <- marge(X_pred, Y, N, n, id, family, corstr, pen, tols_score,
                         M, minspan, print.disp, nb, is.gee)
```

```
corstr <- "exchangeable"   # Exchangeable working correlation structure.
model_marge_exch <- marge(X_pred, Y, N, n, id, family, corstr, pen, tols_score,
                          M, minspan, print.disp, nb, is.gee)
```

Example 2: Presence-absence data

Our second example uses the `leptrine` data set which is provided within the `marge` R-package and analyzed in Stoklosa and Warton (2018). These data are binary (presence-absence) data collected on plant species for 8,678 sites located in the Blue Mountains region. These data contain nine environmental predictor variables collected at each site. Our objective is to predict the presence of the species *Leptospermum trinervium* using the environmental predictor variables. There are are 1,751 absences and 6,927 presences. The `leptrine` data consists of training and test sets, such that $N_{train} = 4{,}339$. The nine environmental predictor variables have been standardized. See `?leptrine` for further details.

Load the data, label the training and test data as objects and set data dimensions.

```r
data("leptrine")

dat1 <- leptrine[[1]]   # Training data.
dat1_t <- leptrine[[2]] # Test data.

Y <- dat1$Y             # Response variable.
N <- length(Y)          # Sample size (number of clusters).
n <- 1                  # Cluster size.
id <- rep(1:N, each = n)   # The ID of each cluster.

X_pred <- dat1[, -10]    # A matrix of the nine environmental predictor variables (note
                         # that we remove the response variable).
X_predt <- dat1_t[, -11] # The provided test data already contains a column of ones (for
                         # the intercept term).
```

Below, we fit a MARS model, a generalized additive model (GAM), a Bayesian additive regression tree (BART) model and MARGE (with two different penalties, $\lambda = 2$ and $\lambda = \log(N)$). We record the computational times (in seconds) and calculate the predictive squared errors (PSE) on test data.

Set the required arguments and tuning parameters to fit models.

```r
family <- "binomial"     # The selected "exponential" family for the GLM/GEE.
corstr <- "independence" # The selected "working correlation" structure (used for GEEs).

is.gee <- FALSE          # Is the model a GEE?
nb <- FALSE              # Is this a negative binomial model?

tols <- 0.0001           # A set tolerance (stopping condition) in the forward pass for MARS.
tols_score <- 0.0001     # A set tolerance (stopping condition) in forward pass for MARGE.
M <- 21                  # A set threshold for the maximum no. of basis functions to be used.
pen <- 2                 # Penalty to be used in GCV (required for MARS).
minspan <- NULL          # A set minimum span value.

print.disp <- FALSE  # Print ALL the output?
```

Fit models and record the computational times. Model fitting can take ~12 minutes to complete (this depends on your computer speed/memory).

```r
# BART.
```

```
X_pred_bart <- as.matrix(cbind(rep(1, nrow(X_pred)), X_pred))
colnames(X_pred_bart)[1] <- "Intercept"
start <- Sys.time()
model_bart <- bart(x.train = X_pred_bart, y.train = Y, x.test = as.matrix(X_predt),
                   verbose = FALSE)
end <- Sys.time()
t1 <- difftime(end, start, units = "secs")

# GAM.

start <- Sys.time()
model_gam <- gam(Y ~ s(FC, k = 8) + s(RAIN_WET_QTR) + s(TMP_MIN) + s(TMP_SEAS) +
                   s(TMP_MN) + s(TMP_MAX) + s(RAIN_DRY_QTR) + s(TMP_MN_COLD_QTR) +
                   s(TMP_MN_WARM_QTR), data = dat1, family = "binomial")
end <- Sys.time()
t2 <- difftime(end, start, units = "secs")

# MARS.

start <- Sys.time()
model_own <- mars_ls(X_pred, Y, pen, tols, M, minspan,
                     print.disp, family, nb)
end <- Sys.time()
t3 <- difftime(end, start, units = "secs")

# MARGE.

start <- Sys.time()
model_marge <- marge(X_pred, Y, N, n, id, family, corstr, pen, tols_score, M, minspan,
                     print.disp, nb, is.gee)
```
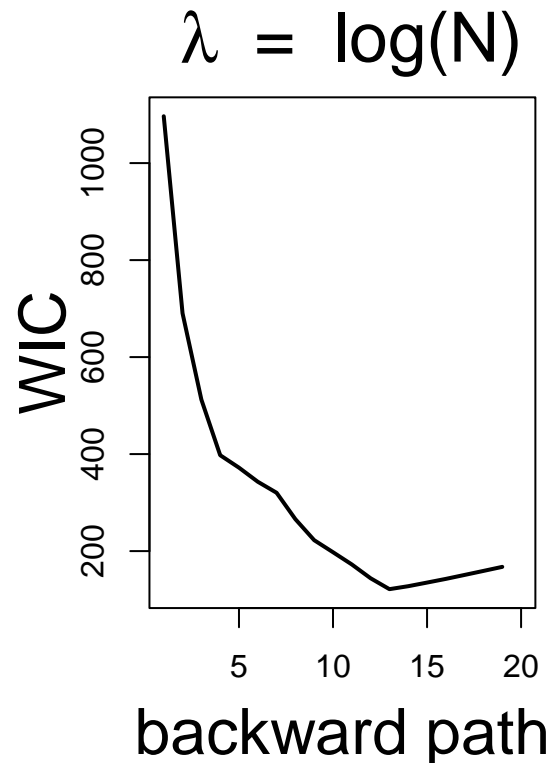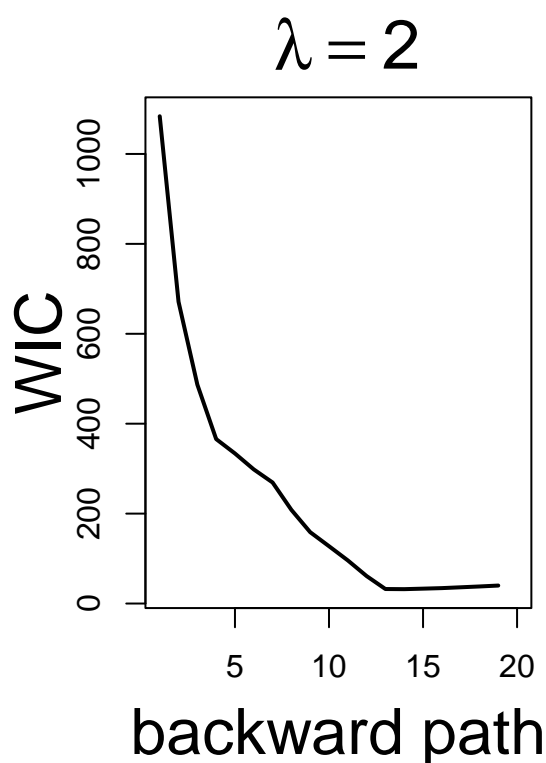
```r
end <- Sys.time()
t4 <- difftime(end, start, units = "secs")
```

Calculate the predictive squared error (PSE) on test data using each fitted model.

```r
H <- function(a) {1 / (1 + exp(-a))} # An inverse logit function (required for PSE below).

pred_own_y <- predict(model_own, X_predt, X_pred)
pred_marge_2_y <- predict(model_marge, X_predt, X_pred, TRUE, "2")
pred_marge_log_y <- predict(model_marge, X_predt, X_pred, TRUE, "logN")

mu_bart <- apply(pnorm(model_bart$yhat.test), 2, mean)
var_bart <- mu_bart*(1 - mu_bart)
PSE_pred_bart<-mean((dat1_t$Y - mu_bart)^2/var_bart, na.rm = TRUE)

mu_gam <- H(predict(model_gam, newdata = dat1_t))
var_gam <- mu_gam*(1 - mu_gam)
PSE_pred_gam <- mean((dat1_t$Y - mu_gam)^2/var_gam, na.rm = TRUE)

mu_own <- H(pred_own_y$eta.p)
var_own <- mu_own*(1 - mu_own)
PSE_pred_own <- mean((dat1_t$Y - mu_own)^2/var_own,na.rm = TRUE)

mu_marge_2 <- H(pred_marge_2_y$eta.p)
mu_marge_log <- H(pred_marge_log_y$eta.p)
var_marge_2 <- mu_marge_2*(1 - mu_marge_2)
var_marge_log <- mu_marge_log*(1 - mu_marge_log)
PSE_pred_marge_2 <- mean((dat1_t$Y - mu_marge_2)^2/var_marge_2, na.rm = TRUE)
PSE_pred_marge_log <- mean((dat1_t$Y - mu_marge_log)^2/var_marge_log, na.rm = TRUE)

PSE_pred_list <- c(PSE_pred_bart, PSE_pred_gam, PSE_pred_own,
                   PSE_pred_marge_2, PSE_pred_marge_log)
```

Display results - these PSE values should match the results given in Table 3 of Stoklosa and Warton (2018), except for BART (which should be similar).

```r
res1 <- PSE_pred_list
res2 <- cbind(t1, t2, t3, t4)

res <- cbind(c(round(res2, digits = 2), round(res2, digits = 2)[4]), round(res1, digits = 3))

rownames(res) <- c("BART", "GAM", "MARS", "MARGE_2_ind", "MARGE_logN_ind")
colnames(res) <- c("time (sec.)", " PSE")

print(res)
```

```
##                time (sec.)   PSE
## BART                180.80 0.835
## GAM                  10.99 0.913
## MARS                204.63 0.941
## MARGE_2_ind         238.04 0.846
## MARGE_logN_ind      238.04 0.839
```