MARGE: Tutorial using presence-absence data

Jakub Stoklosa & David Warton

1st June 2017

This tutorial documents fitting multivariate adaptive regression splines using generalized estimating equations (MARGE) via the `marge` R-package. We also fit some addtional models and compare them with MARGE using a predictive squared error (PSE) on test data. Computationanl times for each model fit are also reported,

For more specific details on the MARGE algorithm see:

Stoklosa, J. and Warton, D.I. (2016). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, in review.

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

We use the `leptrine` data set which is provided within the `marge` R-package and annalysed in Stoklosa and Warton (2016). 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 futher details.

Below we fit a MARS model, a generlaized 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.

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

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

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

```
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).
```

Set the required arguments and tuning parameters to fit models.

```
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 tolerence (stopping condition) in the forward pass for MARS.
tols_score<-0.0001      # A set tolerence (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.

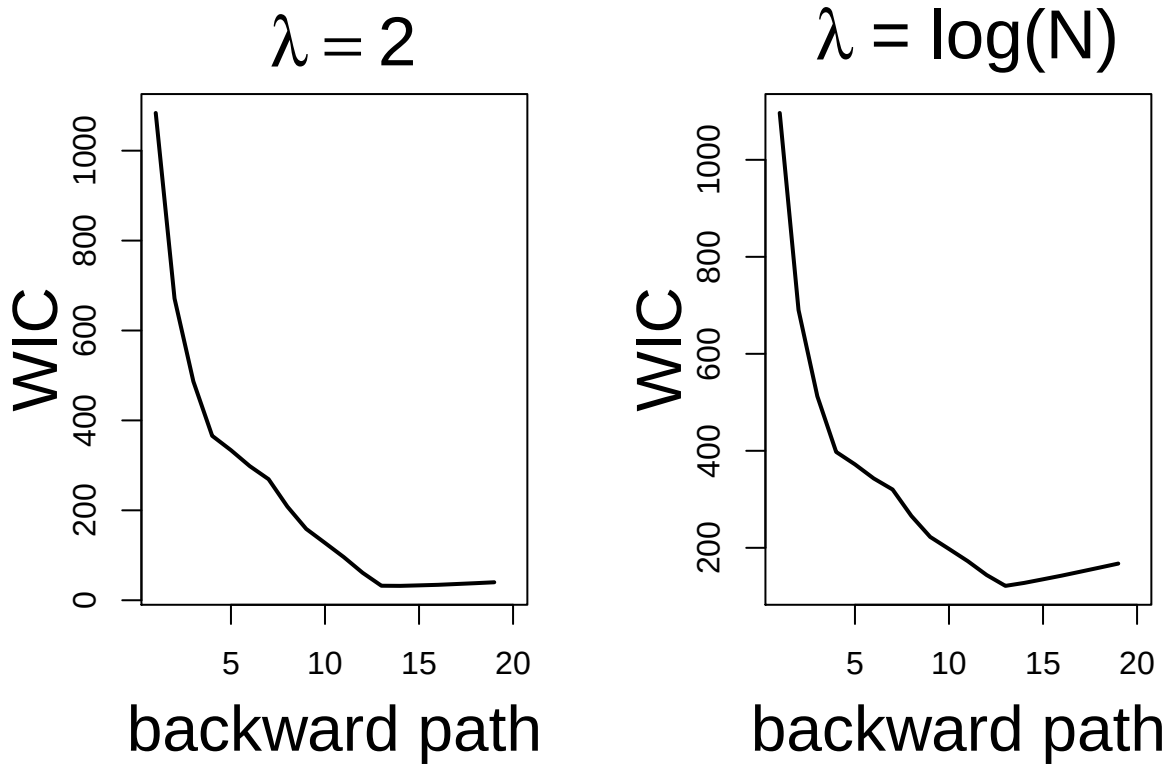
Model fitting can take ~12 minutes to complete (this depends on your computer speed/memory).

```
## 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<-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)
```




```
end<-Sys.time()
t4<-difftime(end,start,units="secs")
```

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

```
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 (2016), except for BART (which should be pretty close).

```
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                153.63 0.849
## GAM                   8.74 0.913
## MARS                162.50 0.941
## MARGE_2_ind         191.79 0.846
## MARGE_logN_ind      191.79 0.839
```