A Partitioning Deletion/Substitution/Addition Algorithm for Creating Survival Risk Groups

Karen Lostritto Yale University
Annette M. Molinaro University of California, San Francisco
Stephen Weston Yale University

Abstract

As an extension of the **partDSA** package (Molinaro, Lostritto, and Weston 2009), we present the **partDSA** Survival package to accommodate survival outcomes. The algorithmic methodology of **partDSA** is described in Molinaro, Lostritto, and van der Laan (2010), and the extension of partDSA to survival outcomes is described in Lostritto, Strawderman, and Molinaro (2011). PartDSA creates a piecewise constant model for predicting an outcome of interest from a complex interaction of covariates. This model consists of "and/or" boolean statements on covariates such that each statement defines a group of observations which are assigned a constant predicted value. The creation of a partDSA model is guided by a specified loss function, and in the case of a survival outcome this loss function must be modified to accommodate a censored outcome. The functionality of partDSA for categorical and continuous outcomes is described in the partDSA vignette and here we describe the new functionality for survival data.

Keywords: recursive partitioning,Survival.

1. Introduction

The Partitioning Deletion/Substitution/Addition algorithm initiates with all observations in a single group and splits and recombines observations in order to exhaustively search potential models. A list of the best models of each size is created, and a cross-validation procedure is employed to select the best model size.

The loss function plays an important role in the model creation process. The goal of minimizing a given loss function guides the choice of the best next step in the iterative model generation process, the choice of the best predicted value, and ultimately the choice of the best model size. In survival data, the outcome is a time to event but in medical studies, many patients will not have experienced this event by the end of the study or they will have dropped out of the study before experiencing the event. These patients will not have a time to event, but rather a time last seen event-free. This time is recorded and these patients are referred to as "censored". The outcome given to partDSA contains both a continuous time T (either time to event or last time seen event free) and a binary censoring variable δ (where a 1 corresponds to a patient who has had the event and a 0 corresponds to a censored patient who has not had the event). The *Surv* function in the Survival library creates a "survival" outcome from T and δ as shown below:

```
> library(survival)
> T=rexp(25)
> delta=sample(0:1,25,TRUE)
> y=Surv(T,delta)
> y

 [1] 2.89546810+ 0.08160169  0.81705171  0.22745616  0.36369706+
 [6] 0.68819946  1.02563999+ 0.20346649  1.25853001  0.45959990
[11] 1.83135418+ 2.16201519  0.58988178  1.74553147  0.36942431
[16] 1.56421249+ 0.19204091+ 0.13089282+ 1.18420683  0.06958681+
[21] 0.45919638+ 0.28860715+ 1.13934619+ 0.05590528+ 0.70803983+
```

In this guide, we will discuss two different partDSA options which allow for the application of this algorithm to survival data. Both modifications make use of a weighting scheme applied to the L_2 loss function for continuous outcomes:

$$L(X, \psi) = (Y - \psi(W))^2 \tag{1}$$

As described in the **partDSA** vignette, continuous outcome data $W_1, ..., W_n$ represent the n observations where $W = (Y, X)$ such that Y is the outcome and X is a vector of p covariates, $X = (X_1, ..., X_p)$ and ψ is the prediction rule. As described above, for survival data, if Y is the time to event, this value is missing (censored) for some observations, requiring a modification to the L_2 loss function approach. The two modifications implemented in partDSA are the IPCW weighting scheme and the Brier weighting scheme, both of which we demonstrate on simulated data. Finally we present an example on the German Breast Cancer dataset.

2. IPCW Weighting Scheme

In the IPCW Weighting Scheme, only uncensored observations contribute directly to the loss function. The contribution of each uncensored observation to the loss function is given a weight, calculated using all of the observations. The details of this weight calculation are described in Lostritto *et al.* (2011). The concept is that uncensored observations who were likely to have been censored by their event time will be given a higher weight and uncensored observations who were unlikely to have been censored by their event time will be given a lower weight. These weights are calculated either with a Kaplan Meier estimate or a Cox estimate. This choice can be specified in the wt.method parameter in DSA.control, which is either set to "Cox" or "KM", with a default value of "KM". Using the "KM" option does not take into account the covariate values when calculating the weights whereas the "Cox" option calculates the weights based on the covariate values. In small sample sizes, it is safer to use the "KM" option because there is less of a chance of running into errors in the coxph function, but if there is informative censoring, it may be worthwhile to choose the "Cox" option. Both options are shown below on a simulation study.

2.1. Simulation Code

In this code, a training set of size 250 and a test set of size 5000 were created. Five covariate values were simulated where the first two determine the scale parameter and thus the time

to event. The remaining three covariates are noise variables. Censoring times are simulated from a uniform distribution in order to achieve a thirty percent censoring level. Survival times are truncated at the 95th percentile. In Lostritto *et al.* (2011), this simulation corresponds to multivariate simulation 1 with high signal. The low signal version uses values of 1 and 0.5 for the scale parameter instead of 5 and 0.5.

```
> library(VGAM)
> library(partDSA)
> set.seed(1)
> p=5
> tr.n=250
> ts.n=5000
> C_level=0.3
> x=matrix(NA,tr.n,p)
> for (j in 1:p){
+ x[,j]=sample(1:100,tr.n,replace=TRUE)
+ }
> x=data.frame(x)
> names(x)=c(paste("X",1:p,sep=""))
> x.test=matrix(NA,ts.n,p)
> for (j in 1:p){
+ x.test[,j]=sample(1:100,ts.n,replace=TRUE)
+ }
> x.test=data.frame(x.test)
> names(x.test)=c(paste("X",1:p,sep=""))
> scale=ifelse(x[,1]>50|x[,2]>75,5,.5)
> time=data.frame(rgpd(tr.n,0,scale,0))
> scale.test=ifelse(x.test[,1]>50|x.test[,2]>75,5,.5)
> time.test=(rgpd(ts.n,0,scale.test,0))
> group2=which(x[,1]>50|x[,2]>75)
> group1=c(1:tr.n)[-group2]
> level1=0
> level2=0
> cens.time=NULL
> while((abs(C_level-level1)>.02)|(abs(C_level-level2)>.02)){
+  cens1.time=runif(length(group1),0,1.25)
+    cens2.time=runif(length(group2),0,15)
+
+    cens1=apply(cbind(cens1.time,time[group1,]),1,which.min)-1
+    cens2=apply(cbind(cens2.time,time[group2,]),1,which.min)-1
+
+    level1=1-sum(cens1)/length(cens1)
+    level2=1-sum(cens2)/length(cens2)
+
+ }
> cens.time[group1]=cens1.time
> cens.time[group2]=cens2.time
```

```
> y0=apply(cbind(cens.time,time),1,min)
> cens0=apply(cbind(cens.time,time),1,which.min)-1
> L = quantile(y0,.95)
> y = pmin(y0,L)
> cens = cens0
> cens[y0 > y] = 1
> y.new=y
> y=log(y)
> y.new.test <- time.test
> y.new.test <- pmin(y.new.test,L)
> y.test=log(y.new.test)
> cens.test=rep(1,ts.n)
> wt=rep(1,tr.n)
> wt.test=rep(1,ts.n)
> detach(package:VGAM)
> model.KM.IPCW=partDSA(x=x,y=Surv(y,cens),x.test=x.test,y.test=Surv(y.test,cens.test),con

<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>

> model.Cox.IPCW=partDSA(x=x,y=Surv(y,cens),x.test=x.test,y.test=Surv(y.test,cens.test),co

<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>

>
>
>
>
```

2.2. Output

The output shown below is similar to that for the usual partDSA model. In both cases the minimum error plus one standard error occurs for the model with two partitions, and we can see that the structure of the model recovers the underlying structure of the simulated data with cutpoints close to those of 50 and 75. The coefficients represent the predicted log of the survival time. In both models we see that the first partition has a much lower survival time than does the second partition. Therefore those observations in the first partition are predicted to be at a higher risk. This is in accordance with the data set-up.

```
> model.KM.IPCW

partDSA object
# partitions   mean CV error   sd CV error   test risk
1              2.880113        0.558193      2.647484
2              1.753652        0.600463      1.718656
3              1.766709        0.482772      1.759015
4              1.692064        0.249118      1.774854

Outcome:
Best of 1 partitions:
   Part.1
    0.24
Best of 2 partitions:
   Part.1    Part.2
    -1.827    0.938
Best of 3 partitions:
   Part.1    Part.2    Part.3
    -1.604    0.938    -2.388
Best of 4 partitions:
   Part.1    Part.2    Part.3    Part.4
    -1.604    0.866    -2.388    1.329

Best 2 partitions
  Partition 1 [of 2]:
    (X1 <= 48.000000) && (X2 <= 72.000000)
  Partition 2 [of 2]:
    (48.000000 < X1)
    (X1 <= 48.000000) && (72.000000 < X2)
Best 3 partitions
  Partition 1 [of 3]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (X5 <= 76.000000)
  Partition 2 [of 3]:
    (48.000000 < X1)
    (X1 <= 48.000000) && (72.000000 < X2)
  Partition 3 [of 3]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (76.000000 < X5)
Best 4 partitions
  Partition 1 [of 4]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (X5 <= 76.000000)
  Partition 2 [of 4]:
    (48.000000 < X1) && (X2 <= 77.000000)
    (X1 <= 48.000000) && (72.000000 < X2)
  Partition 3 [of 4]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (76.000000 < X5)
  Partition 4 [of 4]:
    (48.000000 < X1) && (77.000000 < X2)
```

```
Variable importance matrix:
   COG=1 COG=2 COG=3 COG=4
X1     0     2     3     4
X2     0     2     3     4
X3     0     0     0     0
X4     0     0     0     0
X5     0     0     2     2
```

```
> model.Cox.IPCW
```

```
partDSA object
# partitions   mean CV error   sd CV error   test risk
1              3.038612        0.956169      2.686837
2              1.697476        0.370865      1.745388
3              1.761277        0.296691      1.784603
4              1.833974        0.388278      1.810989

Outcome:
Best of 1 partitions:
   Part.1
    -0.13
Best of 2 partitions:
   Part.1    Part.2
    -1.867    0.694
Best of 3 partitions:
   Part.1    Part.2    Part.3
    -1.64    0.694    -2.431
Best of 4 partitions:
   Part.1    Part.2    Part.3    Part.4
    -1.64    0.6    -2.431    1.246

Best 2 partitions
  Partition 1 [of 2]:
    (X1 <= 48.000000) && (X2 <= 72.000000)
  Partition 2 [of 2]:
    (48.000000 < X1)
    (X1 <= 48.000000) && (72.000000 < X2)
Best 3 partitions
  Partition 1 [of 3]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (X5 <= 76.000000)
  Partition 2 [of 3]:
    (48.000000 < X1)
    (X1 <= 48.000000) && (72.000000 < X2)
  Partition 3 [of 3]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (76.000000 < X5)
Best 4 partitions
```

```
  Partition 1 [of 4]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (X5 <= 76.000000)
  Partition 2 [of 4]:
    (48.000000 < X1) && (X2 <= 77.000000)
    (X1 <= 48.000000) && (72.000000 < X2)
  Partition 3 [of 4]:
    (X1 <= 48.000000) && (X2 <= 72.000000) && (76.000000 < X5)
  Partition 4 [of 4]:
    (48.000000 < X1) && (77.000000 < X2)

Variable importance matrix:
   COG=1 COG=2 COG=3 COG=4
X1     0     2     3     4
X2     0     2     3     4
X3     0     0     0     0
X4     0     0     0     0
X5     0     0     2     2
```

3. Brier Weighting Scheme

In contrast to the IPCW method shown below, Brier allows for some censored observations to be included directly into the loss function. In order to do this, Brier selects a specific time point t* (or several time points) and evaluates whether an observation has had the event by this time point. For uncensored observations, an observation is assigned a 0 if the observation has experienced the event by t* and a 1 if the observation has not experienced the event by t*. Observations censored past the t* will receive an outcome of a 1 because they have not experienced the event by t*. Observations censored before t* cannot be included into the loss function directly because we are unsure as to whether they have experienced the event by t*. Unlike in IPCW where the predicted outcome is a survival time, in Brier, the predicted outcome is a probability of not having experienced the event by t*. The weights are found using all observations, censored and uncensored, with the Kaplan Meier method. In the partDSA function, we still provide the same outcome (Surv(y,cens)) and we specify the loss function to be "Brier". We can select a single value for t* or a vector of values for t* such that the sum of the loss for each t* value is minimized. Note that selecting a vector of values for t* will increase the running time. If no t* value is selected, the default t* will be the median of the observed times.

3.1. Simulation Code

A similar covariate set-up will be applied for the second multivariate simulation in Lostritto *et al.* (2011). The corresponding low signal simulation replaces the 4 in the theta definition with a 1. The code is shown below

```
> tr.n=250
> ts.n=5000
> p=5
```

```
> C_level=0.3
> set.seed(1)
> library(VGAM)
> x=matrix(NA,tr.n,p)
> for (j in 1:p){
+ x[,j]=runif(tr.n)
+ }
> x=data.frame(x)
> names(x)=c(paste("X",1:p,sep=""))
> x.test=matrix(NA,ts.n,p)
> for (j in 1:p){
+ x.test[,j]=runif(ts.n)
+ }
> x.test=data.frame(x.test)
> names(x.test)=c(paste("X",1:p,sep=""))
> theta=4*as.numeric(x[,1]<=.5 | x[,2]>.5)
> psi=exp(theta)
> shape=0
> scale=1/psi
> location=0
> time=data.frame(rgpd(tr.n,location, scale,shape))
> group2=which(scale==unique(scale)[1])
> group1=which(scale==unique(scale)[2])
> theta=4*as.numeric(x.test[,1]<=.5 | x.test[,2]>.5)
> psi=exp(theta)
> shape=0
> scale.test=1/psi
> location=0
> time.test=rgpd(ts.n,location, scale.test,shape)
> level1=0
> level2=0
> cens.time=NULL
> upper1=3.5
> upper2=.1
> while((abs(C_level-level1)>.02)|(abs(C_level-level2)>.02)){
+
+    cens1.time=runif(length(group1),0,upper1)
+    cens2.time=runif(length(group2),0,upper2)
+
+    cens1=apply(cbind(cens1.time,time[group1,]),1,which.min)-1
+    cens2=apply(cbind(cens2.time,time[group2,]),1,which.min)-1
+
+    level1=1-sum(cens1)/length(cens1)
+    level2=1-sum(cens2)/length(cens2)
+    if(abs(level1-C_level)>.1){ upper1=ifelse(level1>C_level,upper1+.1,upper1-.05)}
+    if(abs(level2-C_level)>.1){upper2=ifelse(level2>C_level,upper2+.01,upper2-.01)}
+
```

```
+ }
> if(C_level==0){
+ y0=time[,1]
+ cens0=rep(1,tr.n)
+ }else{
+ cens.time[group1]=cens1.time
+ cens.time[group2]=cens2.time
+ y0=apply(cbind(cens.time,time),1,min)
+ cens0=apply(cbind(cens.time,time),1,which.min)-1
+ }
> L = quantile(y0,.95)
> y = pmin(y0,L)
> cens = cens0
> cens[y0 > y] = 1
> y.new=y
> y=log(y)
> y.new.test <- time.test
> y.test=log(time.test)
> cens.test=rep(1,ts.n)
> wt=rep(1,tr.n)
> wt.test=rep(1,ts.n)
> detach(package:VGAM)
> model1.Brier=partDSA(x=x,y=Surv(y,cens),x.test=x.test,y.test=Surv(y.test,cens.test),cont

<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>

> model2.Brier=partDSA(x=x,y=Surv(y,cens),x.test=x.test,y.test=Surv(y.test,cens.test),cont

<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>

>
```

3.2. Output

If we examine the Brier models and look at the minimum plus one standard error of the cross-validated error, we would select a model of size two. We see that both models yield a

model similar to the original data structure. However, now note that the predicted outcome is a probability, i.e. between 0 and 1, of not having had the event by t*. Therefore, those observations in partition 2 are less likely to experience the event by t* which is in line with the results found with the IPCW models above where observations in partition 2 had higher survival times.

```
> model1.Brier

partDSA object
# partitions   mean CV error   sd CV error   test risk
1              0.245607        0.008975      0.255346
2              0.120329        0.038934      0.179881
3              0.130070        0.041701      0.190401
4              0.139148        0.049660      0.198227

Outcome:
Best of 1 partitions:
   Part.1
    0.575
Best of 2 partitions:
   Part.1    Part.2
    0.269    1
Best of 3 partitions:
   Part.1    Part.2    Part.3
    0.319    1    0
Best of 4 partitions:
   Part.1    Part.2    Part.3    Part.4
    0.244    1    0    0.53

Best 2 partitions
  Partition 1 [of 2]:
    (X1 <= 0.507642)
    (0.507642 < X1) && (0.498515 < X2)
  Partition 2 [of 2]:
    (0.507642 < X1) && (X2 <= 0.498515)
Best 3 partitions
  Partition 1 [of 3]:
    (X1 <= 0.507642) && (X3 <= 0.770225)
    (0.507642 < X1) && (0.498515 < X2)
  Partition 2 [of 3]:
    (0.507642 < X1) && (X2 <= 0.498515)
  Partition 3 [of 3]:
    (X1 <= 0.507642) && (0.770225 < X3)
Best 4 partitions
  Partition 1 [of 4]:
    (X1 <= 0.507642) && (X3 <= 0.770225) && (X4 <= 0.699729)
    (0.507642 < X1) && (0.498515 < X2)
```

```
  Partition 2 [of 4]:
    (0.507642 < X1) && (X2 <= 0.498515)
  Partition 3 [of 4]:
    (X1 <= 0.507642) && (0.770225 < X3)
  Partition 4 [of 4]:
    (X1 <= 0.507642) && (X3 <= 0.770225) && (0.699729 < X4)

Variable importance matrix:
   COG=1 COG=2 COG=3 COG=4
X1     0     2     3     4
X2     0     2     2     2
X3     0     0     2     3
X4     0     0     0     2
X5     0     0     0     0

> model2.Brier

partDSA object
# partitions   mean CV error   sd CV error   test risk
1              0.632933        0.061008      0.614225
2              0.361231        0.075247      0.381562
3              0.373278        0.070381      0.403945
4              0.373754        0.076717      0.415171

Outcome:
Best of 1 partitions:
   Part.1
    0.376
Best of 2 partitions:
   Part.1    Part.2
    0    0.802
Best of 3 partitions:
   Part.1    Part.2    Part.3
    0    0.802    0
Best of 4 partitions:
   Part.1    Part.2    Part.3    Part.4
    0    0.802    0    0

Best 2 partitions
  Partition 1 [of 2]:
    (X1 <= 0.507642)
    (0.507642 < X1) && (0.502251 < X2)
  Partition 2 [of 2]:
    (0.507642 < X1) && (X2 <= 0.502251)
Best 3 partitions
  Partition 1 [of 3]:
    (X1 <= 0.507642) && (X3 <= 0.770225)
```

```
    (0.507642 < X1) && (0.502251 < X2)
  Partition 2 [of 3]:
    (0.507642 < X1) && (X2 <= 0.502251)
  Partition 3 [of 3]:
    (X1 <= 0.507642) && (0.770225 < X3)
Best 4 partitions
  Partition 1 [of 4]:
    (X1 <= 0.507642) && (X3 <= 0.770225) && (X4 <= 0.527782)
    (0.507642 < X1) && (0.502251 < X2)
  Partition 2 [of 4]:
    (0.507642 < X1) && (X2 <= 0.502251)
  Partition 3 [of 4]:
    (X1 <= 0.507642) && (0.770225 < X3)
  Partition 4 [of 4]:
    (X1 <= 0.507642) && (X3 <= 0.770225) && (0.527782 < X4)

Variable importance matrix:
   COG=1 COG=2 COG=3 COG=4
X1     0     2     3     4
X2     0     2     2     2
X3     0     0     2     3
X4     0     0     0     2
X5     0     0     0     0
```

4. Data Example

We now demonstrate the application of both partDSA with the IPCW weighting scheme and the Brier weighting scheme on the German Breast Cancer Study Data (from TH.data package in R). The goal in this dataset is to predict the survival time from the covariates: hormone therapy, age, menopausal status, tumor size, tumor grade, positive nodes, progesterone receptor, and estrogen receptor.

```
> data("GBSG2", package = "TH.data")
> set.seed(1)
> data(GBSG2)
> dataset=GBSG2
> y.new=dataset$time
> y.new=ifelse(dataset$time>2000,2000,dataset$time)
> y=log(y.new)
> cens=dataset$cens
> cens[which(dataset$time>2000)]=1
> x=dataset[,c(1:8)]
> brier.vec=median(y)
> set.seed(1)
> model.IPCW=partDSA(x=x,y=Surv(y,cens),control=DSA.control(vfold=5, cut.off.growth=5,MPD=
```

```
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
```

```
> set.seed(1)
> model.Brier=partDSA(x=x,y=Surv(y,cens),control=DSA.control(vfold=5, cut.off.growth=5,MPD
```

```
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
<simpleError in x[i, , drop = FALSE]: subscript out of bounds>
```

```
> model.IPCW

partDSA object
# partitions   mean CV error   sd CV error   test risk
1              0.496450        0.072504      0.492555
2              0.429613        0.085979      0.387659
3              0.415749        0.071409      0.377856
4              0.419135        0.067032      0.373843
5              0.419747        0.070552      0.372322

Outcome:
Best of 1 partitions:
   Part.1
    7.059
Best of 2 partitions:
   Part.1    Part.2
    7.21    6.363
Best of 3 partitions:
   Part.1    Part.2    Part.3
    7.21    6.551    6.069
Best of 4 partitions:
   Part.1    Part.2    Part.3    Part.4
    7.21    6.678    6.069    6.259
Best of 5 partitions:
   Part.1    Part.2    Part.3    Part.4    Part.5
    7.21    6.678    5.915    6.259    6.211

Best 2 partitions
  Partition 1 [of 2]:
    (pnodes <= 4.000000)
```

```
    (4.000000 < pnodes) && (28.000000 < progrec)
  Partition 2 [of 2]:
    (4.000000 < pnodes) && (progrec <= 28.000000)
Best 3 partitions
  Partition 1 [of 3]:
    (pnodes <= 4.000000)
    (4.000000 < pnodes) && (28.000000 < progrec)
  Partition 2 [of 3]:
    (tgrade is in {I, II}) && (4.000000 < pnodes) && (progrec <= 28.000000)
  Partition 3 [of 3]:
    (tgrade is III) && (4.000000 < pnodes) && (progrec <= 28.000000)
Best 4 partitions
  Partition 1 [of 4]:
    (pnodes <= 4.000000)
    (4.000000 < pnodes) && (28.000000 < progrec)
  Partition 2 [of 4]:
    (tgrade is in {I, II}) && (4.000000 < pnodes) && (progrec <= 28.000000) && (estrec <=
  Partition 3 [of 4]:
    (tgrade is III) && (4.000000 < pnodes) && (progrec <= 28.000000)
  Partition 4 [of 4]:
    (tgrade is in {I, II}) && (4.000000 < pnodes) && (progrec <= 28.000000) && (22.000000
Best 5 partitions
  Partition 1 [of 5]:
    (pnodes <= 4.000000)
    (4.000000 < pnodes) && (28.000000 < progrec)
  Partition 2 [of 5]:
    (tgrade is in {I, II}) && (4.000000 < pnodes) && (progrec <= 28.000000) && (estrec <=
  Partition 3 [of 5]:
    (age <= 51.000000) && (tgrade is III) && (4.000000 < pnodes) && (progrec <= 28.000000)
  Partition 4 [of 5]:
    (tgrade is in {I, II}) && (4.000000 < pnodes) && (progrec <= 28.000000) && (22.000000
  Partition 5 [of 5]:
    (51.000000 < age) && (tgrade is III) && (4.000000 < pnodes) && (progrec <= 28.000000)

Variable importance matrix:
         COG=1 COG=2 COG=3 COG=4 COG=5
horTh        0     0     0     0     0
age          0     0     0     0     2
menostat     0     0     0     0     0
tsize        0     0     0     0     0
tgrade       0     0     2     3     4
pnodes       0     2     3     4     5
progrec      0     2     3     4     5
estrec       0     0     0     2     2


> model.Brier
```

```
partDSA object
# partitions   mean CV error   sd CV error   test risk
1              0.227160        0.004204      0.227808
2              0.211217        0.019837      0.195658
3              0.206163        0.027426      0.192379
4              0.208590        0.029763      0.191082
5              0.211238        0.029368      0.190160

Outcome:
Best of 1 partitions:
   Part.1
    0.649
Best of 2 partitions:
   Part.1    Part.2
    0.738    0.287
Best of 3 partitions:
   Part.1    Part.2    Part.3
    0.738    0.209    0.5
Best of 4 partitions:
   Part.1    Part.2    Part.3    Part.4
    0.738    0.377    0.5    0.156
Best of 5 partitions:
   Part.1    Part.2    Part.3    Part.4    Part.5
    0.738    0.377    0.5    0.251    0.067

Best 2 partitions
  Partition 1 [of 2]:
    (pnodes <= 5.000000)
    (5.000000 < pnodes) && (53.000000 < progrec)
  Partition 2 [of 2]:
    (5.000000 < pnodes) && (progrec <= 53.000000)
Best 3 partitions
  Partition 1 [of 3]:
    (pnodes <= 5.000000)
    (5.000000 < pnodes) && (53.000000 < progrec)
  Partition 2 [of 3]:
    (tsize <= 45.000000) && (5.000000 < pnodes) && (progrec <= 53.000000)
  Partition 3 [of 3]:
    (45.000000 < tsize) && (5.000000 < pnodes) && (progrec <= 53.000000)
Best 4 partitions
  Partition 1 [of 4]:
    (pnodes <= 5.000000)
    (5.000000 < pnodes) && (53.000000 < progrec)
  Partition 2 [of 4]:
    (tsize <= 21.000000) && (5.000000 < pnodes) && (progrec <= 53.000000)
  Partition 3 [of 4]:
    (45.000000 < tsize) && (5.000000 < pnodes) && (progrec <= 53.000000)
```

```
  Partition 4 [of 4]:
    (21.000000 < tsize <= 45.000000) && (5.000000 < pnodes) && (progrec <= 53.000000)
Best 5 partitions
  Partition 1 [of 5]:
    (pnodes <= 5.000000)
    (5.000000 < pnodes) && (53.000000 < progrec)
  Partition 2 [of 5]:
    (tsize <= 21.000000) && (5.000000 < pnodes) && (progrec <= 53.000000)
  Partition 3 [of 5]:
    (45.000000 < tsize) && (5.000000 < pnodes) && (progrec <= 53.000000)
  Partition 4 [of 5]:
    (age <= 53.000000) && (21.000000 < tsize <= 45.000000) && (5.000000 < pnodes) && (prog
  Partition 5 [of 5]:
    (53.000000 < age) && (21.000000 < tsize <= 45.000000) && (5.000000 < pnodes) && (progr

Variable importance matrix:
         COG=1 COG=2 COG=3 COG=4 COG=5
horTh        0     0     0     0     0
age          0     0     0     0     2
menostat     0     0     0     0     0
tsize        0     0     2     3     4
tgrade       0     0     0     0     0
pnodes       0     2     3     4     5
progrec      0     2     3     4     5
estrec       0     0     0     0     0
```

The models created using the IPCW and Brier methods both have two partitions involving the variables progesterone receptor and number of positive nodes. These models show the ability of partDSA to elucidate complex relationships among covariates and survival time.

References

Lostritto K, Strawderman RL, Molinaro AM (2011). "*partDSA*: A Partitioning Deletion/Substitution/Addition Algorithm for Creating Survival Risk Groups." *Biometrics*.

Molinaro AM, Lostritto K, van der Laan MJ (2010). "*partDSA*: Deletion/Substitution/Addition Algorithm for Partitioning the Covariate Space in Prediction." *Bioinformatics*. Doi: 10.1093/bioinformatics/btq142.

Molinaro AM, Lostritto K, Weston S (2009). "partDSA: Partitioning using deletion, substitution, and addition moves." http://cran.r-project.org/web/packages/partDSA.

Affiliation:

Firstname Lastname
Affiliation

Address, Country
E-mail: name@address
URL: http://link/to/webpage/