

MKL Cox (MKCox)

We will be mimicking example 1, two factors with linear relationship with the hazard function $h(X) = (X_1 + 2 * X_2)$, from ‘Fenchel duality of Cox partial likelihood and its application in survival kernel learning’ Wilson et. al (2020). Here is quick example of how to run the code. The concordance for Random survival forest, gradient boosting, and MKCox are printed and note that they are compatible. All of the machine learning methods are able to capture the linear relationship between the features and hazard function as shown in the scatterplots.

Loading data and plotting data

```
library(RMKL)
library(ggplot2)
library(survival)
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
library(randomForestSRC)
```

```
##
## randomForestSRC 2.9.3
##
## Type rfsrc.news() to see new features, changes, and bug fixes.
##
```

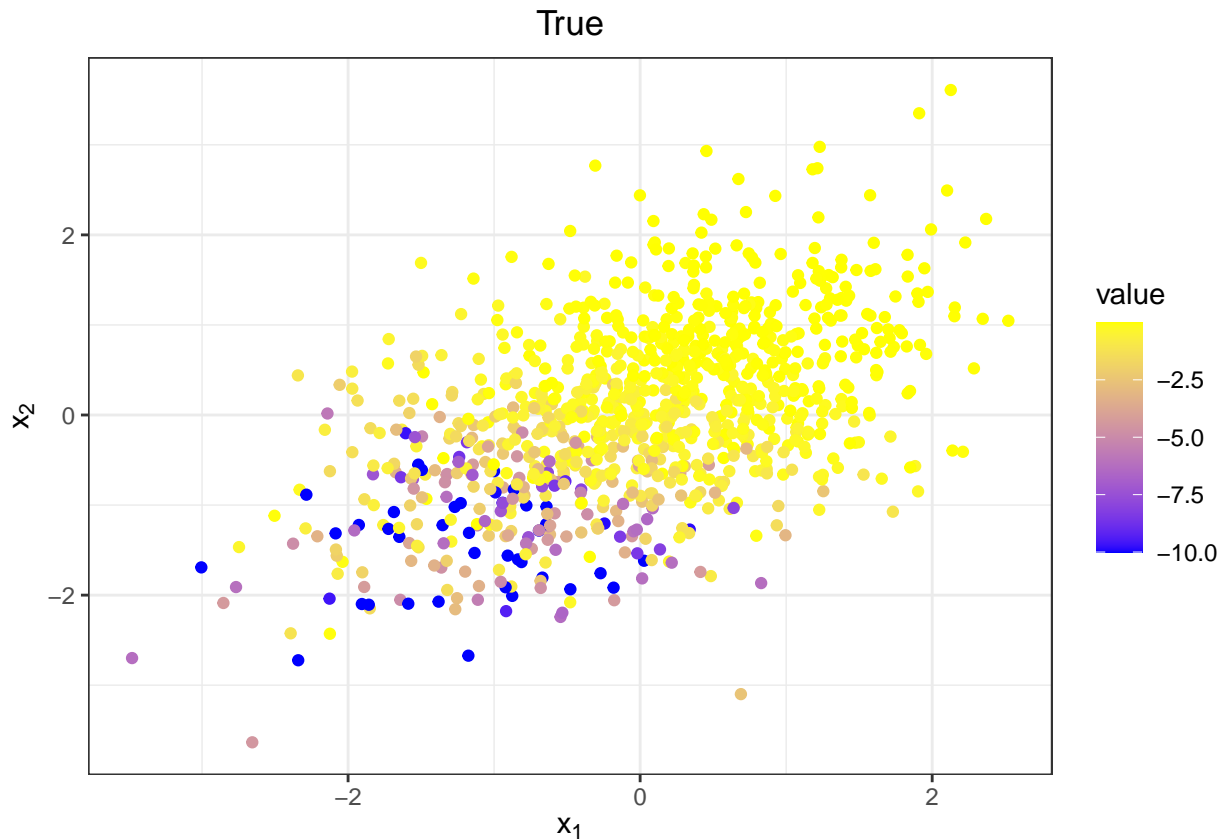
```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:ggplot2':
##
## alpha
```

```
data(Surv_data)
head(Surv_data)
```

```
##           x1           x2           time status
## 1  0.70152066  0.8990872  0.040041376    TRUE
## 2 -0.21223395 -0.6490695  1.230534188    TRUE
## 3 -0.08404824  1.4702301  0.005509966    TRUE
## 4  0.86906459  0.4015776  0.437023163    TRUE
## 5 -1.88954937 -1.9095317  4.220627369    TRUE
## 6 -0.58108709  0.1864732  0.138348879    TRUE
```

```
ggplot(Surv_data, aes(x = x1, y = x2, color = -time)) + geom_point() + scale_color_gradient(low = 'blue', high = 'red') +
labs(color = 'value', title = 'True', x = expression(x[1]), y = expression(x[2])) + theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```



```
print
```

```
## function (x, ...)
## UseMethod("print")
## <bytecode: 0x556f1acd4c58>
## <environment: namespace:base>

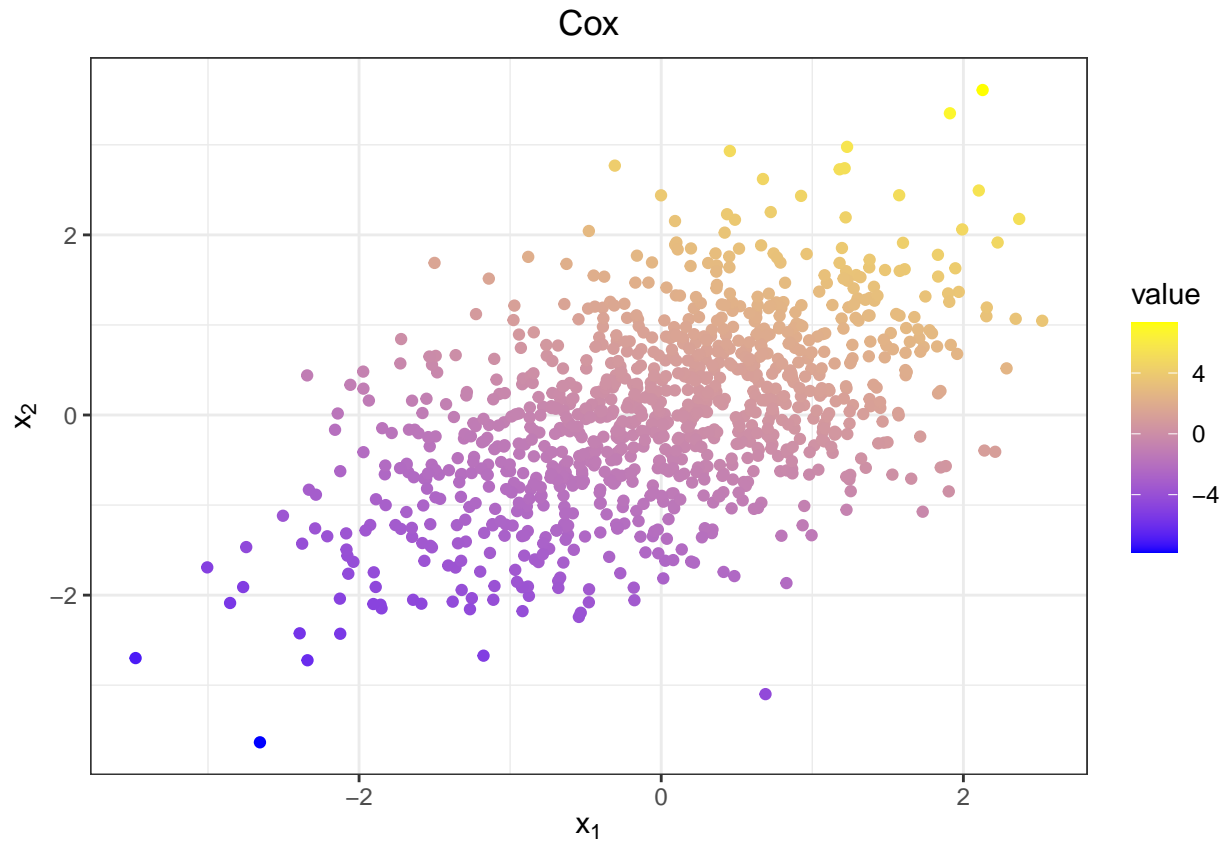
#Cox model
fit=coxph(Surv(Surv_data$time, Surv_data$status) ~ Surv_data$x1 + Surv_data$x2, method = "breslow")
fit2 <- step(fit, direction='both', k = log(dim(Surv_data)[1]))

## Start:  AIC=8815.32
## Surv(Surv_data$time, Surv_data$status) ~ Surv_data$x1 + Surv_data$x2
##
##           Df    AIC
## <none>          8815.3
## - Surv_data$x1  1 9078.6
## - Surv_data$x2  1 9524.6

cox_pred=predict(fit, as.data.frame(Surv_data[,1:2]))
summary(fit)$concordance[1]

##           C
## 0.831493

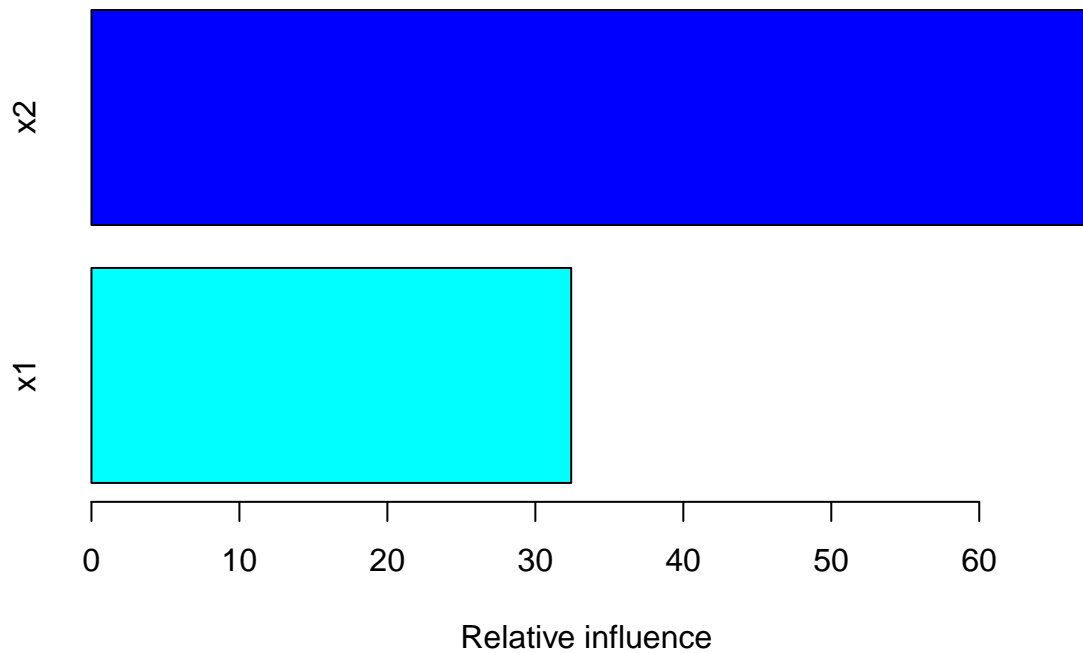
Surv_data$Cox = cox_pred
ggplot(Surv_data, aes(x = x1, y = x2, color = Cox)) + geom_point() + scale_color_gradient(low = 'blue',
```



#Gradient Boosting

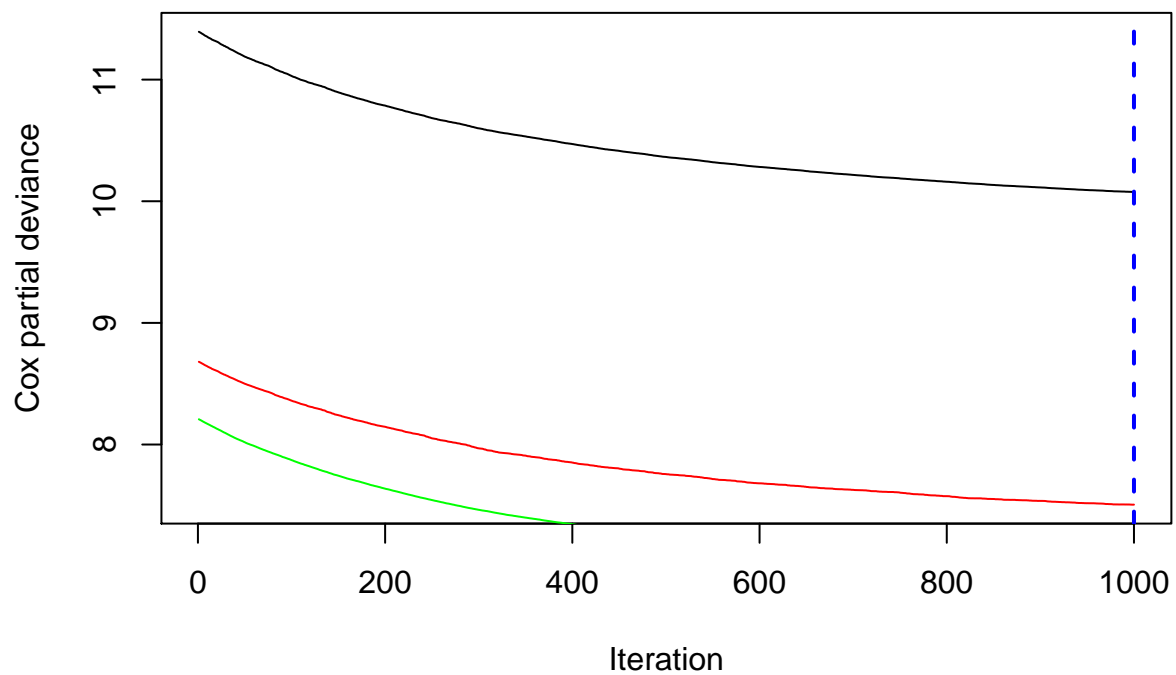
```
gbm1 <- gbm(Surv(time, status) ~ x1 + x2,      # formula
             data=Surv_data,                  # dataset
             distribution="coxph",
             n.trees=1000,                    # number of trees
             shrinkage=0.005,                 # shrinkage or learning rate, 0.001 to 0.1 usually work
             interaction.depth=1,              # 1: additive model, 2: two-way interactions, etc
             bag.fraction = 0.5,               # subsampling fraction, 0.5 is probably best
             train.fraction = 0.8,             # fraction of data for training, first train.fraction*N used f
             cv.folds = 5,                     # do 5-fold cross-validation
             verbose = F)                     # print progress

summary(gbm1)
```



```
##   var  rel.inf
## x2  x2 67.57844
## x1  x1 32.42156
```

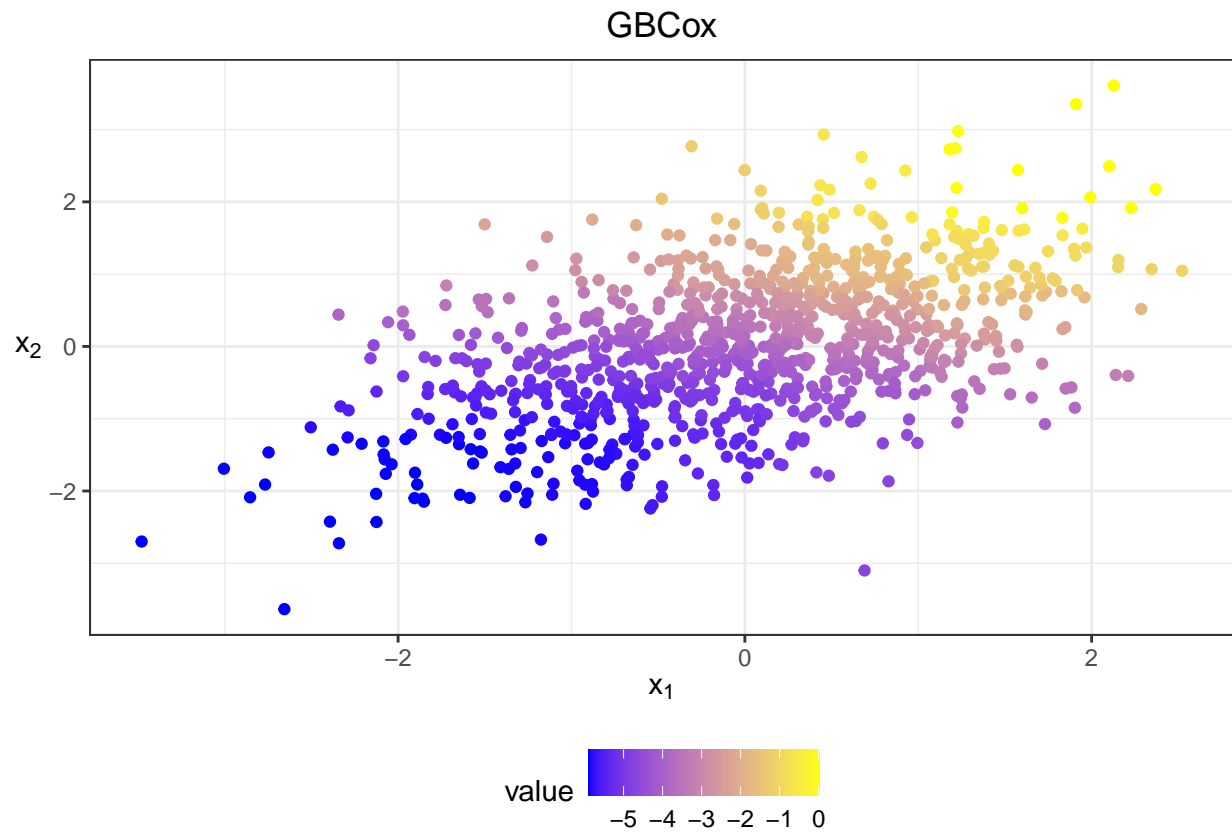
```
best.iter <- gbm.perf(gbm1,method = "cv")
```



```
gpred2=predict(gbm1,Surv_data,best.iter)
```

```
Surv_data$GBM = gpred2
```

```
ggplot(Surv_data, aes(x = x1, y = x2, color = GBM)) + geom_point() + scale_color_gradient(low = 'blue',
```

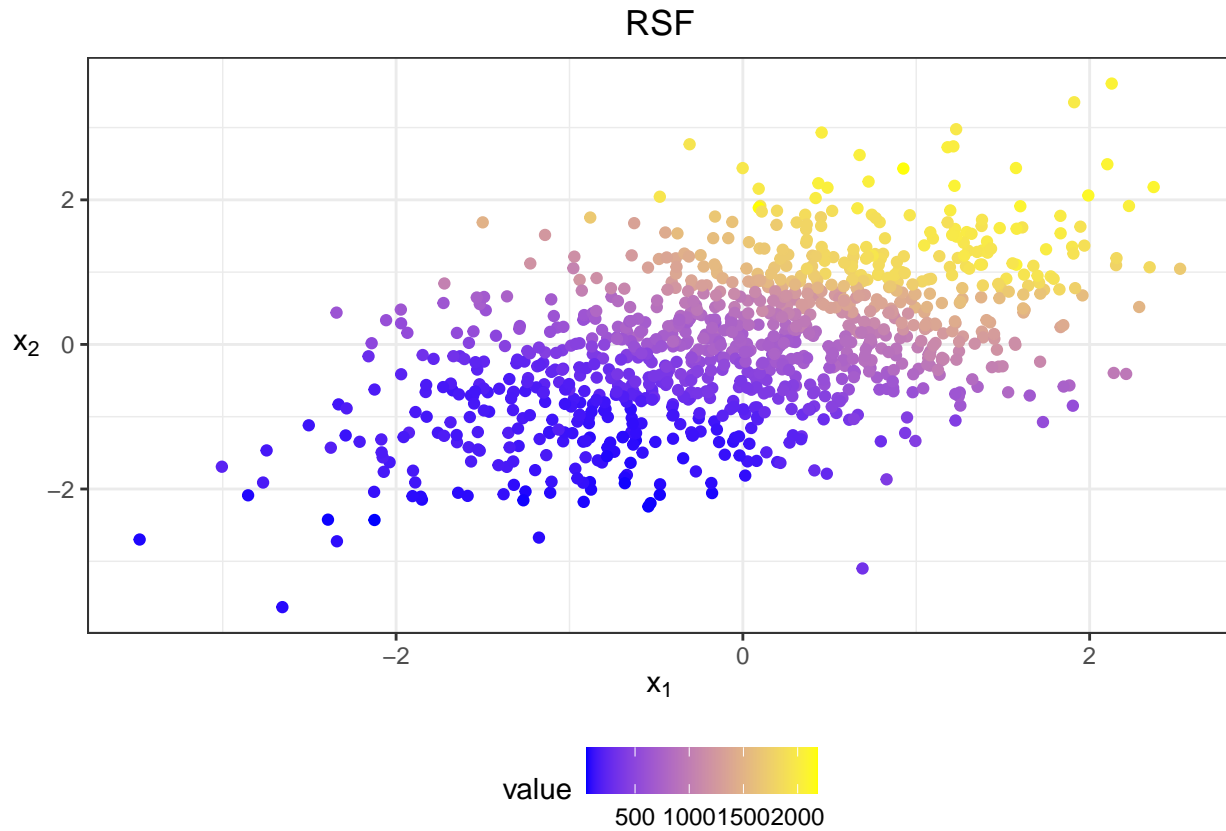


```
gbm.con = survConcordance(Surv(time, status) ~ GBM, Surv_data)$con
gbm.con
```

```
## concordant
## 0.8321379
```

```
#Random Survival Forest
```

```
modrf <- rfsrc(Surv(time, status) ~ x1 + x2, data = Surv_data, nsplit = 10)
prerf <- predict(modrf, Surv_data, outcome = 'test')$predicted.oob
Surv_data$RSF = prerf
ggplot(Surv_data, aes(x = x1, y = x2, color = RSF)) + geom_point() + scale_color_gradient(low = 'blue',
```



```
RSF.con = survConcordance(Surv(time, status) ~ RSF, Surv_data)$con
RSF.con
```

```
## concordant
## 0.8204752
```

```
#MKCox
```

```
#Getting survival times in ascending order
```

```
ordtr <- order(Surv_data$time)
Surv_data_ordered = Surv_data[ordtr,]
```

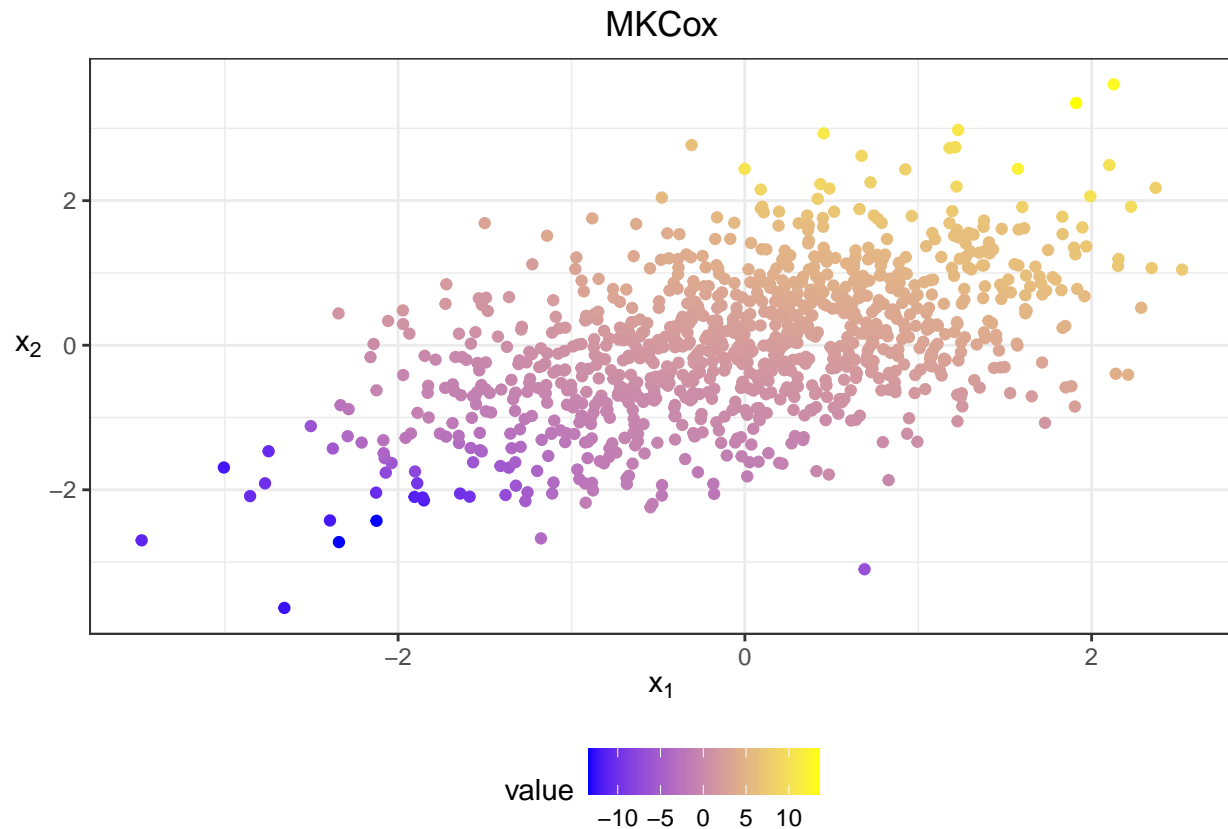
```
xx = Surv_data_ordered[,1:2]
del = Surv_data_ordered$status
yy = Surv_data_ordered$time
if (!del[1]) {
  first1 <- which(del)[1]
  xx <- xx[-(1:(first1 - 1)), ]
  yy <- yy[-(1:(first1 - 1))]
  del <- del[-(1:(first1 - 1))]
  nn <- dim(Surv_data)[1] - first1 + 1
} else {
  nn <- dim(Surv_data)[1]
}
```

```
rho0 <- .001*(Surv_data$status - seq(0, 10, length.out = dim(Surv_data)[1]))
klist <- list(kernelMatrix(rbfdot(1), as.matrix(xx)),
              kernelMatrix(vanilladot(), as.matrix(xx)))
ktlist <- list(kernelMatrix(rbfdot(1), as.matrix(xx), as.matrix(Surv_data[,1:2])),
```

```

      kernelMatrix(vanilladot(), as.matrix(xx), as.matrix(Surv_data[,1:2])))
kk <- simplify2array(klist)
kkk <- simplify2array(ktlist)
modmkl <- SurvMKL(y = Surv_data$time, del = Surv_data$status, K = kk, rho = rho0, C = 0.005, lambda = 0)
mkl = predict_Surv(modmkl, kkk)
Surv_data$MKCox = mkl
ggplot(Surv_data, aes(x = x1, y = x2, color = MKCox)) + geom_point() + scale_color_gradient(low = 'blue', high = 'yellow')

```



```

MKCox.con = survConcordance(Surv(time, status) ~ MKCox, Surv_data)$con
MKCox.con

```

```

## concordant
## 0.8329882

```