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

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

```
## Loaded gbm 2.1.5
```

```r
library(randomForestSRC)
```

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

```r
library(kernlab)
```

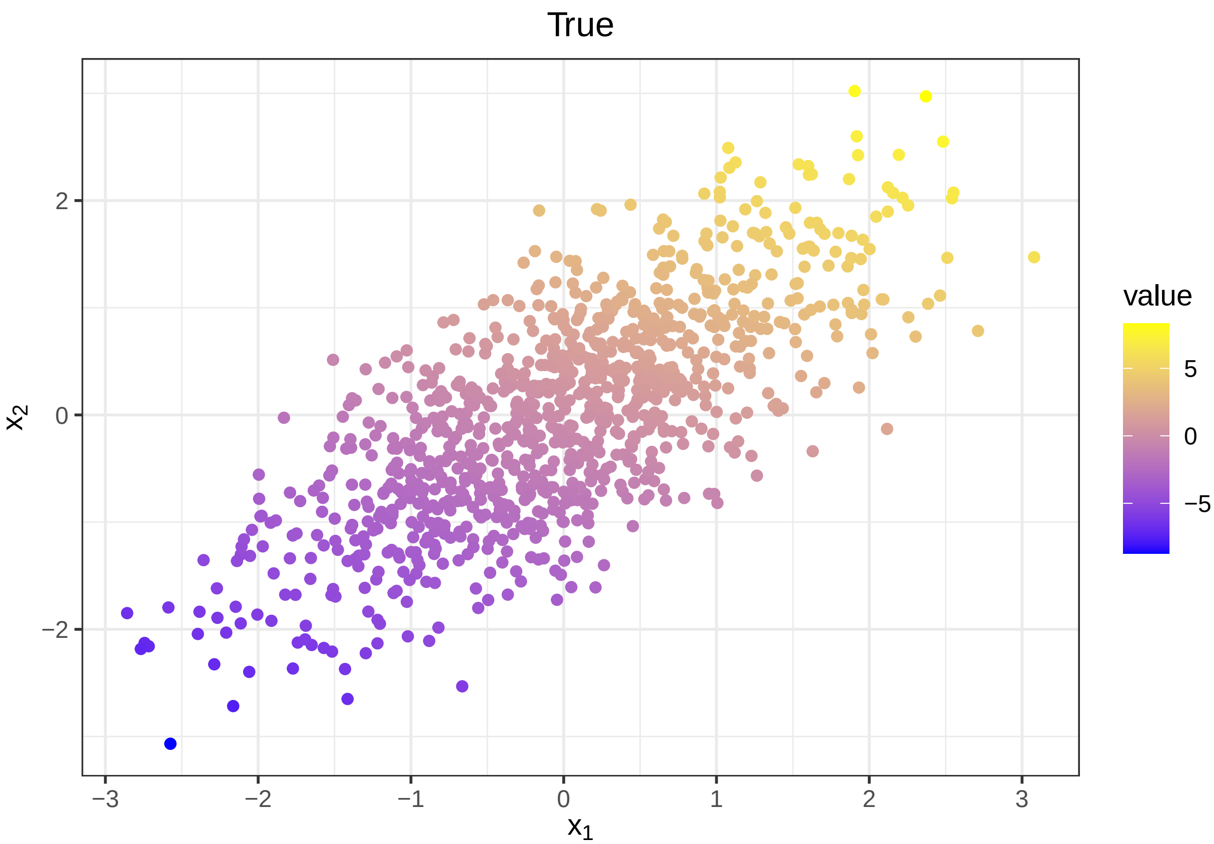
```
##
## Attaching package: 'kernlab'
```

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

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

```
##           x1         x2        time status     f.true
## 1  0.5832340  0.1701828  0.50816428   TRUE  0.9235995
## 2 -0.1635738  1.2076649  0.02034150   TRUE  2.2517561
## 3 -1.1993542 -0.1038692  1.29859953   TRUE -1.4070927
## 4 -0.3287327  0.7051839  0.08373968   TRUE  1.0816350
## 5 -0.5471016 -0.7309604  1.77983095  FALSE -2.0090225
## 6 -0.6874842 -1.3567569 10.00000000  FALSE -3.4009981
```

```r
ggplot(Surv_data, aes(x = x1, y = x2, color = f.true)) + geom_point() + scale_color_gradient(low = 'blu
labs(color = 'value', title = 'True',x = expression(x[1]), y = expression(x[2])) + theme_bw() + theme(pl
```



#Cox model

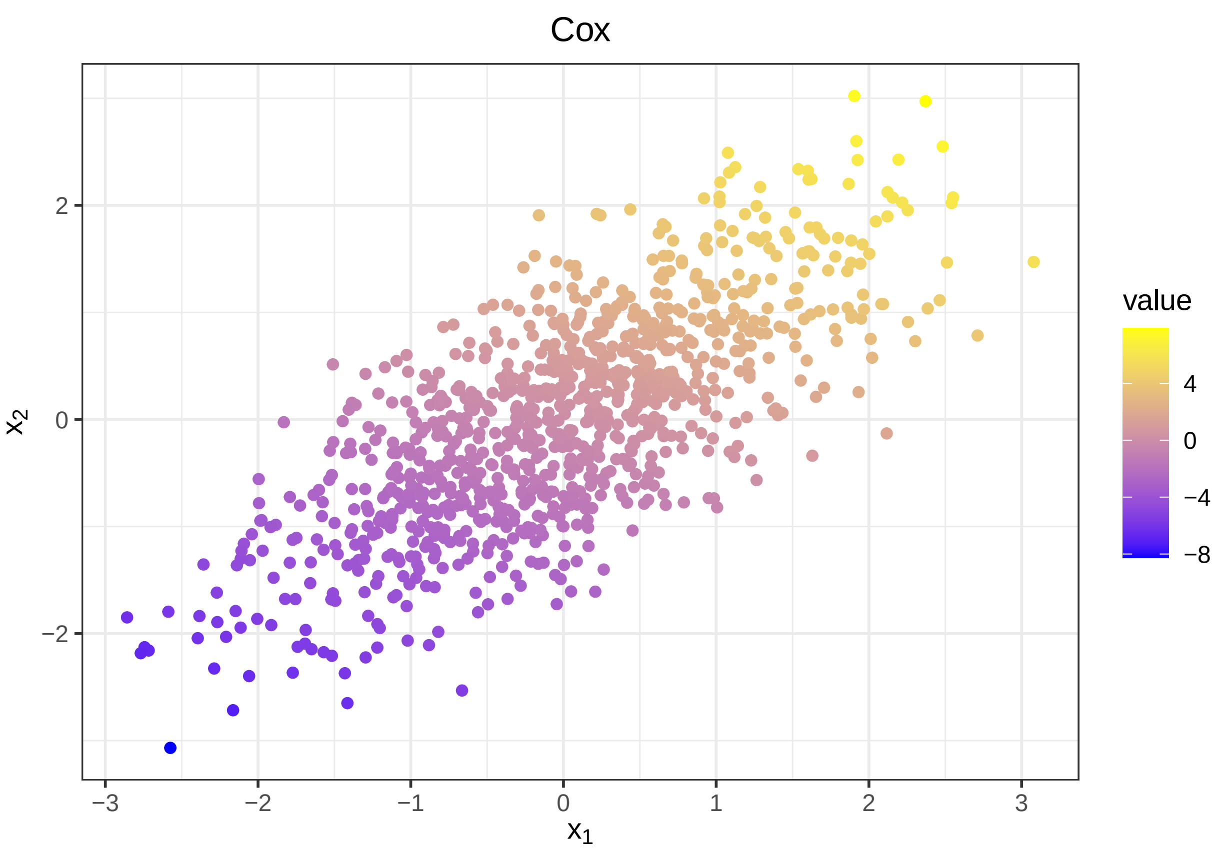
```
fit=coxph(Surv(Surv_data$time, Surv_data$status) ~ Surv_data$x1 + Surv_data$x2, method = "breslow")
fit2 <- step(fit, direction='both', k = log(dim(Surv_data)[1]))
```

```
## Start:  AIC=8439.86
## Surv(Surv_data$time, Surv_data$status) ~ Surv_data$x1 + Surv_data$x2
##
##                Df    AIC
## <none>            8439.9
## - Surv_data$x1  1 8693.7
## - Surv_data$x2  1 9195.9
```

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

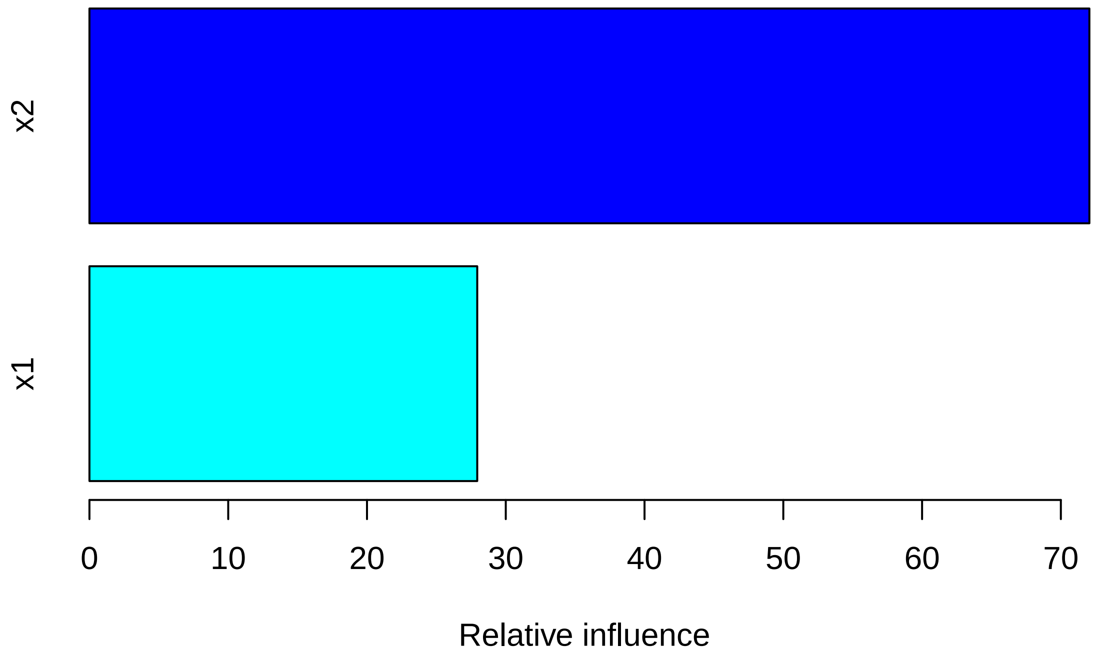
```
##         C
## 0.8707435
```

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



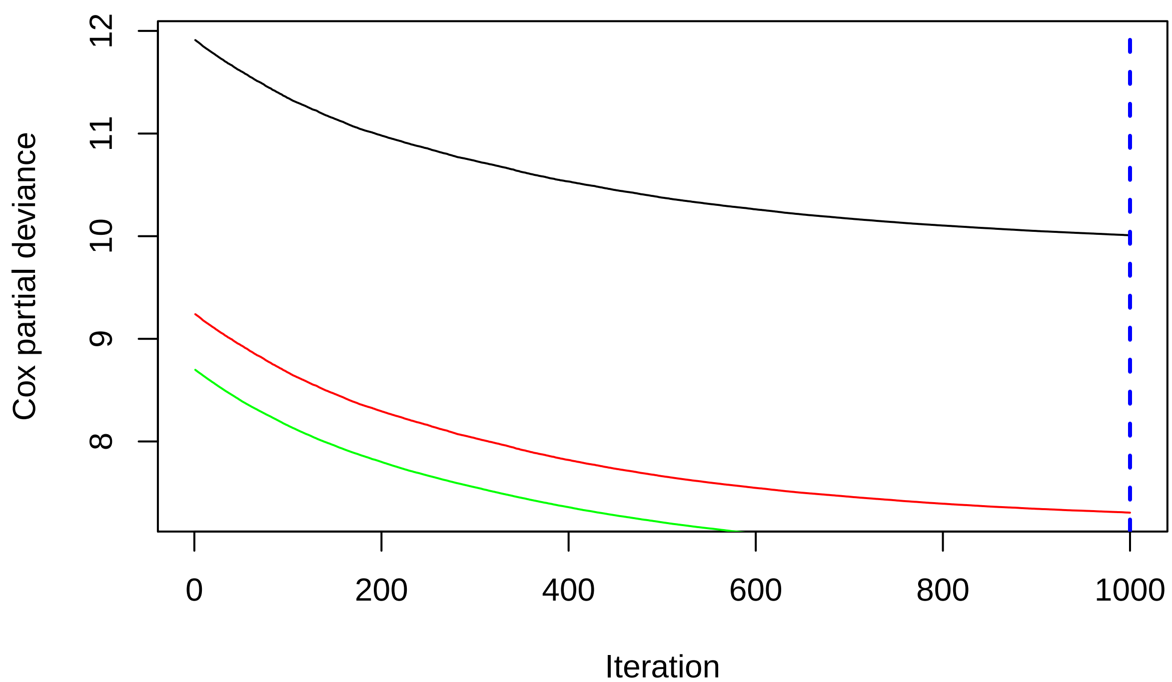
#Gradient Boosting

```
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)
```

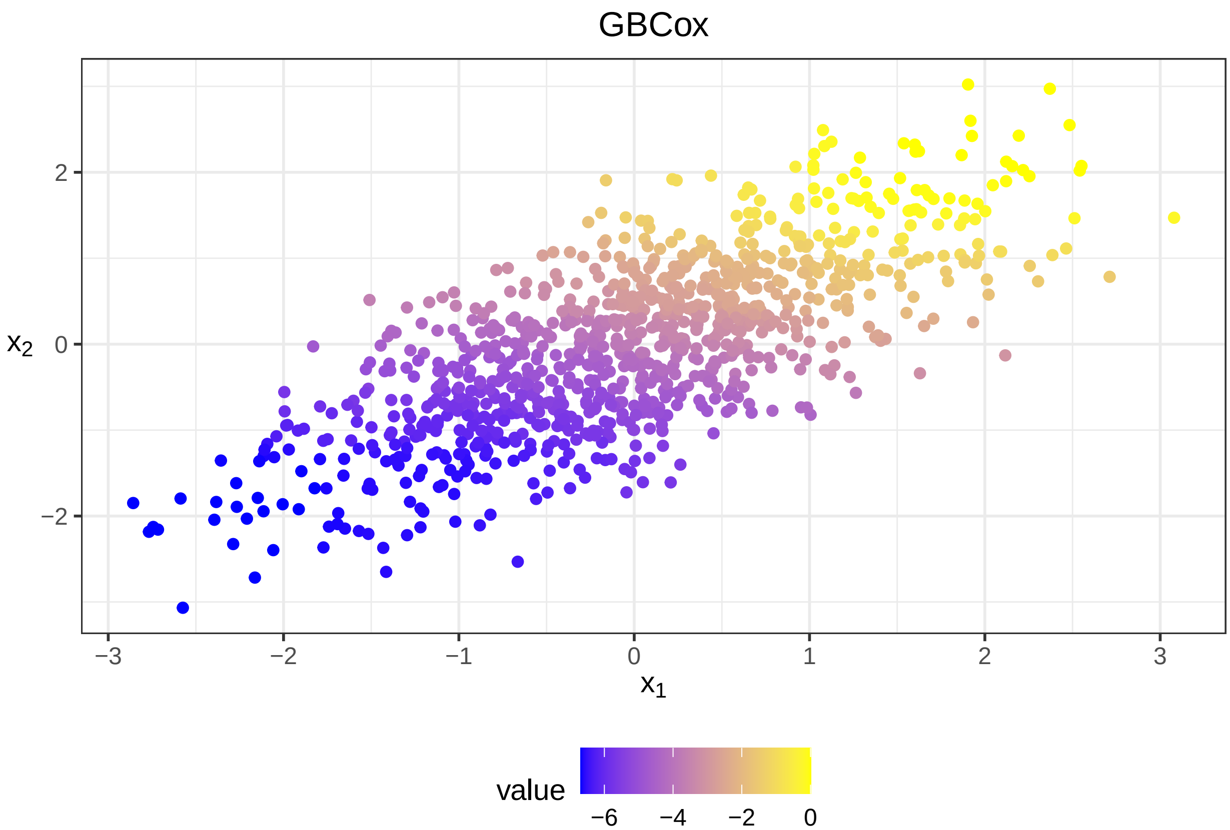



```
##    var  rel.inf
## x2  x2 72.05622
## x1  x1 27.94378
```

```
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',
```

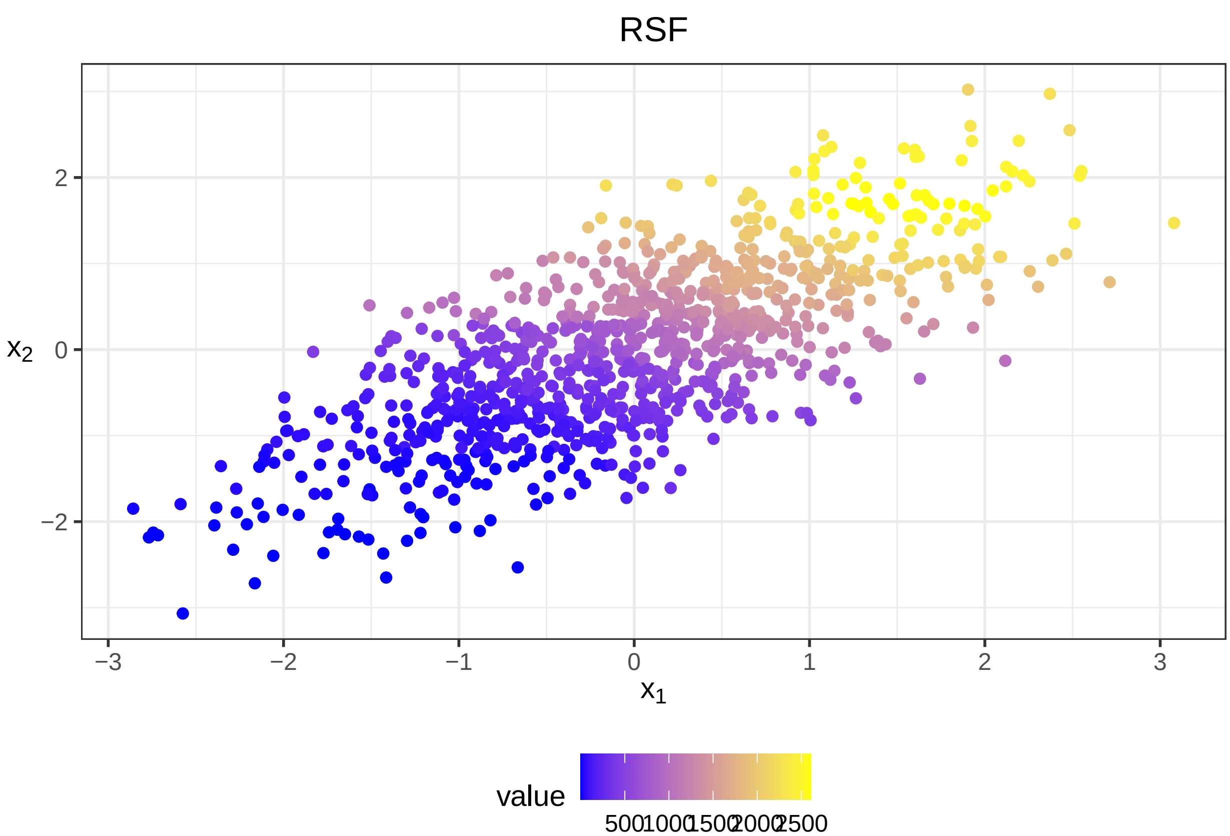



```
gbm.con = survConcordance(Surv(time, status) ~ GBM, Surv_data)$con
gbm.con
```

```
## concordant
##  0.8740848
```

#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',
```




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

```
## concordant
##  0.8653995
```

Run ELM model

‘ “{r} modelm <- ELMCox(Surv_data[,1:2], Surv(Surv_data$time, Surv_{d}ata$status)) ypreelm <- predict(modelm, Surv_data[,1:2]) Surv_data$ELM = ypreelmsurvConcordance(Surv(time, status)\ ELM, data = Surv_{d}ata$)con ggplot(Surv_data, aes(x = x1, y = x2, color = ELM)) + geom_point() + scale_color_gradient(low = ‘blue’, high = ‘yellow’) + labs(x = expression(x[1]), y = expression(x[2]), color = ‘value’, title = ‘ELM’) + theme_bw() + theme(plot.title = element_text(hjust = .5), axis.title.y = element_text(angle = 0, vjust = .5), legend.position = ‘bottom’)

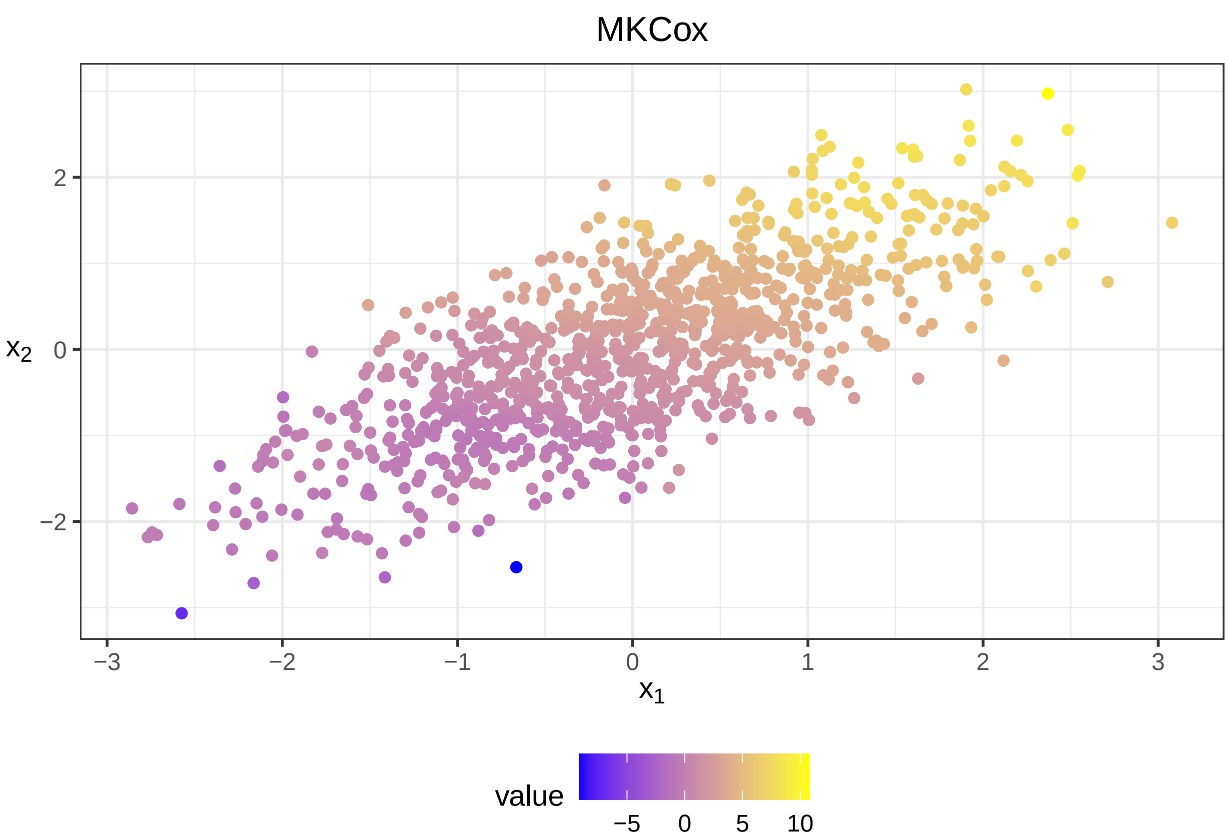
```
#MKCox

```r
#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$status, K = kk, rho =  rho0, C = 0.005, lambda = (
mkl = predict_Surv(modmkl, kkk)
Surv_data$MKCox = mkl
ggplot(Surv_data, aes(x = x1, y = x2, color = MKCox)) + geom_point() + scale_color_gradient(low = 'blue
```




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

```
## concordant
##  0.8716352
```