Survival data analysis - Lab 3

Jeremy Bron

Février 2019

Packages utilisés:

```
library(KMsurv)
library(survival)
library(tidyverse)
library(knitr)
library(data.table)
library(glmnet)
library(randomForestSRC)
library(ggRandomForests)
library(risksetROC)
library(caret)
library(corrplot)
library(ROCR)
```

Exercice 1

But: Construction et interprétation d'un modèle de Cox pour le jeu de données BMT. BMT est un jeu de données médicales contenant 137 patients ayant subi une greffe de moelle osseuse. Le but est d'expliquer le retour à la normale (ou la rechute) de certains paramètres médicaux.

Question 1 - Codage et transformation des variables statiques

Le jeu de données est transformé. Les variables sont renommées, l'âge est translaté (par rapport à la médiane), les variables non utilisées sont retirées.

En détails:

```
#Renommer les variables
names(bmt) = c("group", "t1",    "DFS",    "d1"  ,  "d2" ,   "DFSstatus"  ,  "ta"  ,  "da" ,   "tc"   , "dc" ,   "tp" ,   "d
p" ,   "agep" ,   "aged"  ,  "genderp"  ,  "genderd",    "cmvp"  ,  "cmvd"  ,  "waiting"  ,  "FAB" ,   "hospital" ,   "MTW"
)
#Renommer les valeurs prises par la variable group
bmt$group = recode(bmt$group, "1"="ALL","2"="Low","3"="High")
#Translater les âges agep et aged
bmt$aged = bmt$aged -28
bmt$agep = bmt$agep -28
#Retirer les variables t1,d1,d2,ta,da,tc,dc.
bm2 = bmt %>% select(-c(t1,d1,d2,ta,da,tc,dc))
```

Question 2 - Transformation des données en format start-stop

Valeurs des individus 1 et 14:

	group	t1	DFS	d1	d2	DFSstatus	ta	da	tc	dc	tp	dp	agep	aged	genderp	genderd	cmvp	cmvd	waiting	FAB	hospital	MT
1	ALL	2081	2081	0	0	0	67	1	121	1	13	1	-2	5	1	0	1	1	98	0	1	
14	ALL	1167	1167	0	0	0	39	1	487	1	1167	0	-1	-6	0	1	1	1	191	0	2	

Le jeu de données est transformé au format start-stop à l'aide de la fonction `tmerge` du package `survival` . Une colonne id est rajouté pour identifier les patients . En détail:

Question 3 - Construction d'un modèle de Cox

La variable numérique `hospital` , qui indique l'hôpital ou le patient a été traité, est transformé en facteur. La sélection des variables est effectuée sur le modèle suivant parle critère AIC.

```
bmt2 = bmt2 %>% mutate(hospital = factor(hospital))
mod1 = coxph(Surv(tstart, tstop,endpt) ~ group + agep + aged + genderp + genderd + cmvp + cmvd + waiting + FAB + hospital, d
ata = bmt2_merge)
mod1S = step(mod1, direction = "both", k=2, trace=0)
summary(mod1S)
```

```
## Call:
## coxph(formula = Surv(tstart, tstop, endpt) ~ group + agep + FAB +
##     hospital, data = bmt2_merge)
##
##   n= 256, number of events= 83
##
##               coef exp(coef) se(coef)      z Pr(>|z|)
## groupHigh -0.10378   0.90142  0.34251 -0.303  0.76189
## groupLow  -0.77800   0.45932  0.35006 -2.222  0.02625 *
## agep       0.02239   1.02264  0.01365  1.641  0.10089
## FAB        0.83689   2.30917  0.27478  3.046  0.00232 **
## hospital2  0.54615   1.72658  0.32244  1.694  0.09030 .
## hospital3 -0.19964   0.81903  0.32667 -0.611  0.54112
## hospital4 -0.91524   0.40042  0.42061 -2.176  0.02955 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##           exp(coef) exp(-coef) lower .95 upper .95
## groupHigh    0.9014     1.1094    0.4607    1.7639
## groupLow     0.4593     2.1771    0.2313    0.9122
## agep         1.0226     0.9779    0.9957    1.0504
## FAB          2.3092     0.4331    1.3476    3.9569
## hospital2    1.7266     0.5792    0.9178    3.2482
## hospital3    0.8190     1.2210    0.4318    1.5537
## hospital4    0.4004     2.4974    0.1756    0.9131
##
## Concordance= 0.685  (se = 0.033 )
## Rsquare= 0.121   (max possible= 0.946 )
## Likelihood ratio test= 33.1  on 7 df,   p=3e-05
## Wald test            = 31.63  on 7 df,   p=5e-05
## Score (logrank) test = 33.8  on 7 df,   p=2e-05
```

La sélection pas le critère AIC retient 4 variables: `group` et `hôpital` sont des facteurs à plusieurs niveaux qui produisent plus de covariables dans le modèle. `âge` et `FAB` sont également retenus.

Interprétation du modèle et des coefficients :

- La variable `group` indique le risque associé au développement de Leucémies aiguës myéloïdes (AML)
- Les individus du `groupLow` sont significatifs (p-value = 0.026) et le risque de rechute est réduit de moitié (multiplié par 0.46)
- Les individus du `groupHigh` ne sont pas significatifs (p-value > 0.76), l'intervalle de confiance à 95% est du coefficient est 0.46-1.76: le patient appartenant à ce groupe ont eux des rétablissements très inégaux.

- La variable `hôpital` identifie dans quel hôpital le patient a été traité. On constate que selon l'hôpital, les probabilités de rechute du patient sont très différentes, à savoir:
- `hospital2` (Alferd) : relativement significatifs (p-value = 0.09), les patients ont une probabilité de rechute multipliée par 1.7.
- `hospital4` (Hahnemann) : significatifs (p-value = 0.003), les patients ont une probabilité de rechute multipliée par 0.4.
- `hospital3` (St. Vincent): pas significatif

L'effet du choix de l'hôpital influence donc beaucoup les probabilités de rechute du patient, ce constat pourrait mener à vouloir comprendre plus en détail pourquoi cela est le cas.

- La variable `agep` est linéaire et à une influence similaire sur toute la plage d'âge. On aurait pu créer une variable factorielle avec des classes de tranche d'âge pour mieux analyser l'effet de l'âge du patient. `âge` est légèrement significative (p-value = 0.101) et et le risque de rechute est augmenté de 2.3% par années supplémentaires par rapport à la médiane des âges de 28 ans (en effet cette variole avait été transformée). Pour les patients plus jeunes que 28 ans le risque diminue par symétriquement.

- La variable `FAB` est binaire et décrit (entre autres) la présence ou l'absence de monocytes dans le sang. Cette variable est la plus significative parmi celles retenues (p-value = 0.002). Les patients avec des paramètres associés au facteur 1 ont une chance de rechute multipliée par 2.3.

À noter que la sélection avec le BIC pénalise beaucoup les variables et ne retient que la variable `group` et `FAB` .

En synthèse on constate que le retour à la normale de `platelet_recov` dépends principalement du groupe de risque du patient, de certains paramètres sanguins et l'âge à également une influence. On constate également que le choix de l'hôpital impact significativement les chances de recovery, ce qui ne devrait normalement pas être autant le cas.

Exercice 2

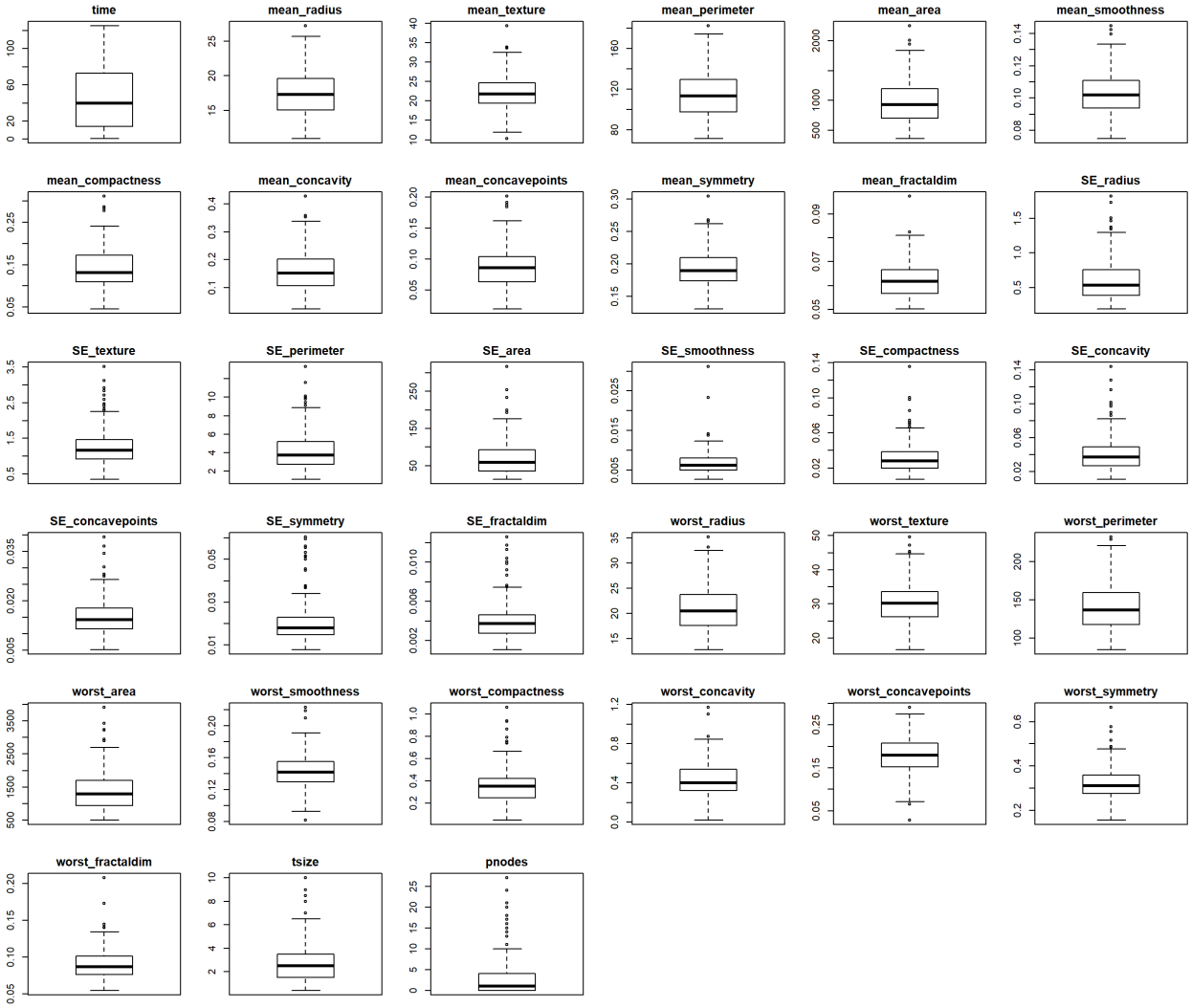
But: Comparer les méthodes de l'analyse de survie aux méthodes de classification sur le jeu de données `wpbc` .

`wpbc` est un jeu de données médicales contenant le suivie de 198 patients traités pour un cancer du sein.

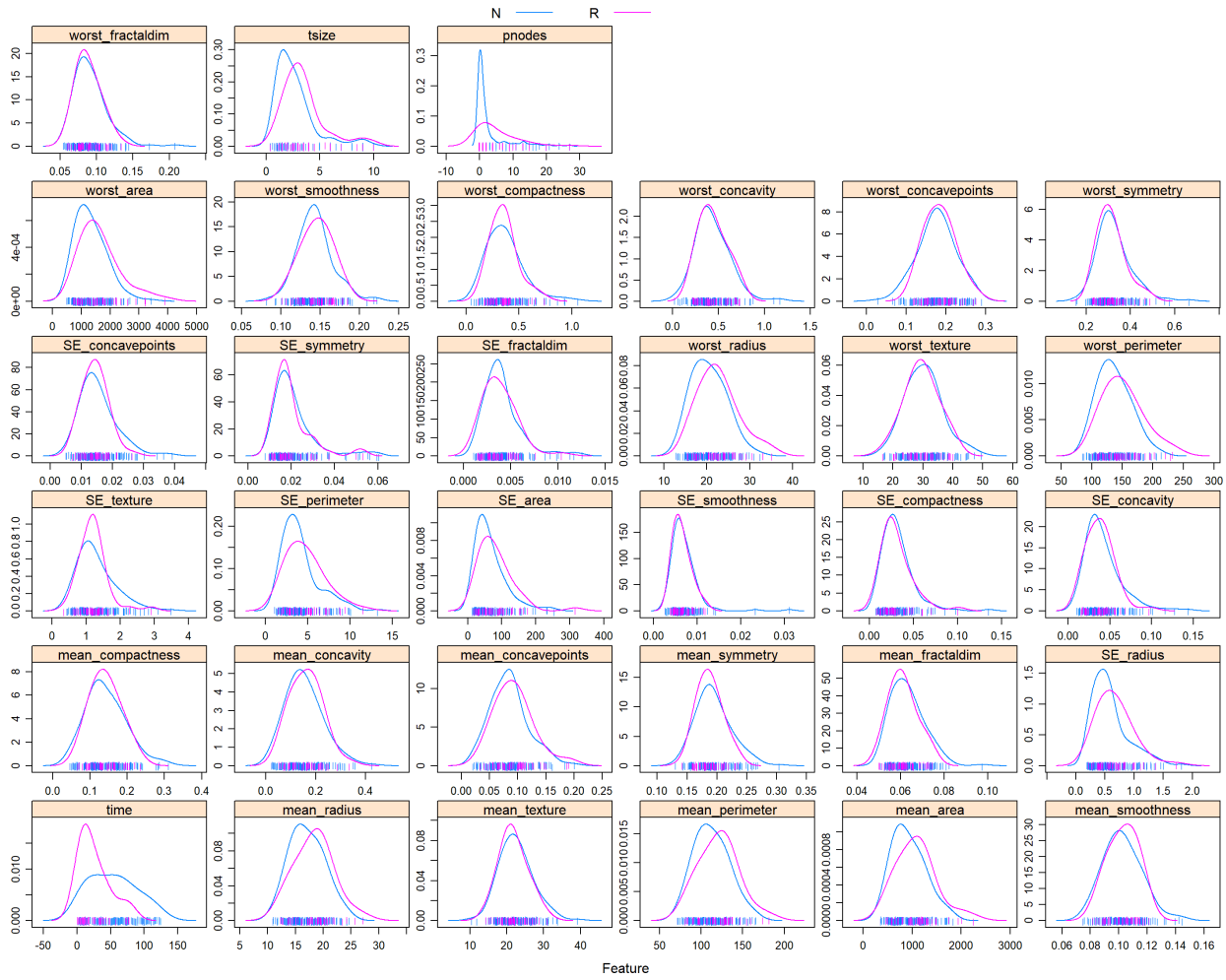
Exploration des données

On s'intéresse à la probabilité de rechute à 24 mois. Celle-ci est codée par la variable `status` qui indique si le patient a rechuté (R: cancer récurent) ou pas (N: Non récurent). La variable `time` contient le temps en mois jusqu'a la récurrence ou le temps sans récurrence (sain). Les autres variables décrivent les attributs des tumeurs (taille, aspect, nombres ,etc.) basés sur de l'imagerie médicale des patients.

La distribution de ces variables est comme suit:



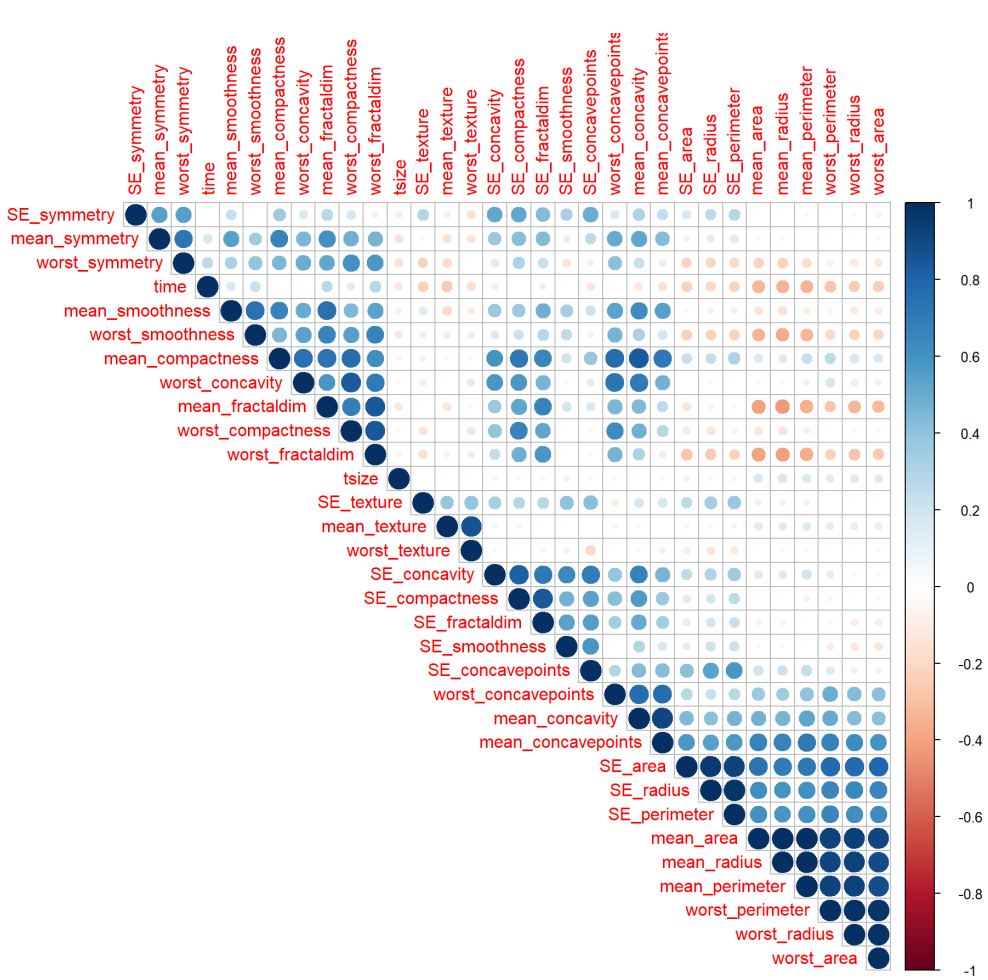
La densité des variables selon le statut de récurrences est comme suit:



On constate que les variables décrivant les mesures des tumeurs ont une densité ressemblant à des lois normales. Les densités pour les patients N ou R sont peu différentiées. La densité de variable `pnode` pour les non récurrents est très différente de celle des récurrents. De plus cette variable cette variable est manquante pour 4 individus.

Ainsi au vu de cette grande différence (et que l'imputation n'est pas le but de l'exercice ici) les individus avec données manquantes sont enlevés

Visualisation de la corrélation des données



Certaines variables sont très corrélées entre elles. En effet il s'agit de différentes mesures des tumeurs (par exemple périmètre et diamètre) qui représentent des données relativement similaires. Cela pourra avoir une influence sur les modèles utilisés par la suite.

Transformation des données

On souhaite prévoir la probabilité de rechute ("récurrent") à 24 mois, pour cela on introduit une nouvelle variable indicatrice de rechute dans l'intervalle. Les données sont également centrées et réduites pour garantir la convergence de l'algorithme, en détail:

```
data("wpbc", package = "TH.data")
wpbc = wpbc %>% drop_na(pnodes)

wpbcProcessCoef=preProcess(wpbc,  method = c("center", "scale"))
wpbcProcessd=predict(wpbcProcessCoef,wpbc)

#la variable de temps ne dois pas etre centrée...
wpbcProcessd$time = wpbc$time

#Indicateur de rechute dans l'interval 0-24 mois (probleme qui nous interesse ici !) et preparation des données
wpbcProcessd$lessthan24 = (wpbcProcessd$time <= 24 & wpbcProcessd$status == "R")
wpbcProcessd$lessthan24 = ifelse(wpbcProcessd$lessthan24, 1,0)

wpbcProcessd$status = if_else(wpbcProcessd$status=="R",1,0)
```

Création du modèle de Cox et validation croisée

- La sélection des variables se fait sur le jeu de données entier grâce à la fonction step et le critère AIC (les autres critères donnent de moins bons résultats).
- La fonction `risk` permet de calculer le risque (`1 - survie`) associé au modèle de cox pour un moment donné, ici 24 mois.
- La validation croiser se fait en stratifiant les échantillons afin d'avoir une proportion de rechutes similaire dans les sous-ensembles.

```
mod1 = coxph(Surv(time,status) ~ . - lessthan24, data = wpbcProcessd , control = coxph.control(iter.max = 150))
#selection des variables grace a la méthode step (AIC)
mod1S = step(mod1, direction = "both", k=2)


#pour le calcul du risque de rechute à x(variable time) mois
risk = function(model, newdata, time) {
  as.numeric(1-summary(survfit(model, newdata = newdata, se.fit = F, conf.int = F), times = time)$surv)
}

#Valisation croisée
PredMatsurv = function(N) {
  result=data.table()

  for (i  in 1:N) {
    i=1
    Train = as.vector(createDataPartition(wpbcProcessd$status, p= .7 , times=1))

    modI = coxph(formula = Surv(time, status) ~ mean_radius + mean_perimeter +
                   mean_area + mean_smoothness + mean_concavity + mean_fractaldim +
                   SE_compactness + SE_concavity + worst_radius + worst_area +
                   worst_compactness + worst_concavity + pnodes, data = wpbcProcessd[Train$Resample1,]
                 )

    result = rbind (result, cbind(risk(modI,wpbcProcessd[-Train$Resample1,],24),wpbcProcessd$lessthan24[-Train$Resample1]))
  }


  return(result)
}

resultAll.cox = PredMatsurv(1000)
colnames(resultAll.cox) = c("prediction", "class")
pred = prediction(resultAll.cox$prediction,resultAll.cox$class)
perf.auc = performance(pred, measure = "auc")
perf.auc@y.values
```

Résultats:

- L'interprétation des résultats se fait grâce au critère de l'AUC, estimé grâce au package `ROCR`
- La validation croisée sur 1000 itérations donne un AUC de 0.731

Avant d'illustrer ce résultat, nous testons une autre méthode pour sélectionner les variables du modèle de cox: le lasso.

Modèle de Cox avec variables sélectionnées avec le Lasso

L'algorithme de validation croisée est le même que le précédent avec les variables sélectionnées par Lasso

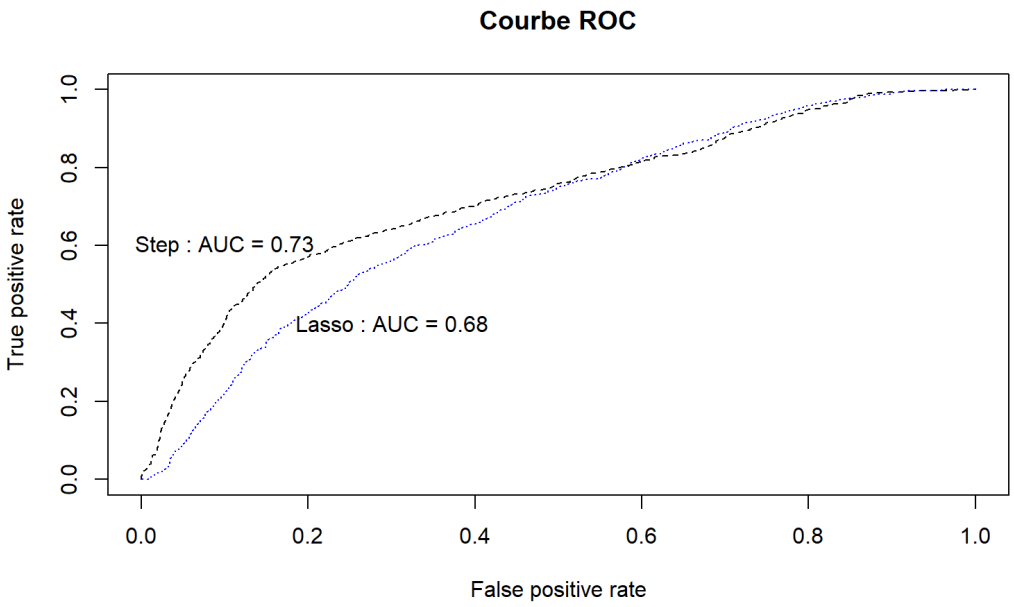
```
mod.lasso = model.matrix(lessthan24~ .  -status, data = wpbcProcessd)

#mod.lasso = model.matrix(status~ .  -lessthan24, data = wpbcProcessd)
lasso.reponse = Surv(wpbcProcessd$time, wpbcProcessd$status)
lasso.mod = glmnet(mod.lasso,lasso.reponse,family="cox",alpha=1 ,maxit = 1000000)
plot(lasso.mod)
lasso.fit = cv.glmnet(mod.lasso,lasso.reponse,family="cox",alpha=1,maxit = 1000000)
plot(lasso.fit)
numero_du_best_model=which(lasso.fit$lambda==lasso.fit$lambda.min)
lasso.fit$glmnet.fit$beta[,numero_du_best_model]
```

La sélection des variables par Lasso donne de moins bons résultats:

- Lorsque l'ont utilise statuts pour la variable réponse du Lasso, l'AUC est de 0,63
- Lorsque l'ont test la variables `lessthan24` qui code les individus avec rechute dans la période 0-24 mois, l'AUC est légèrement meilleur avec 0,68

Dans les deux cas avec le lasso on obtient des performances inférieures qu'avec l'AIC. En effet certaines variables sont très corrélées entre elles et cela influence négativement les performances du Lasso. D'ailleurs, nous avons dû augmenter le nombre d'itérations de l'algorithme lasso pour qu'il converge (sans warnings)

Comparaison des courbes ROC des modèles précédents



Modèle survival random forest

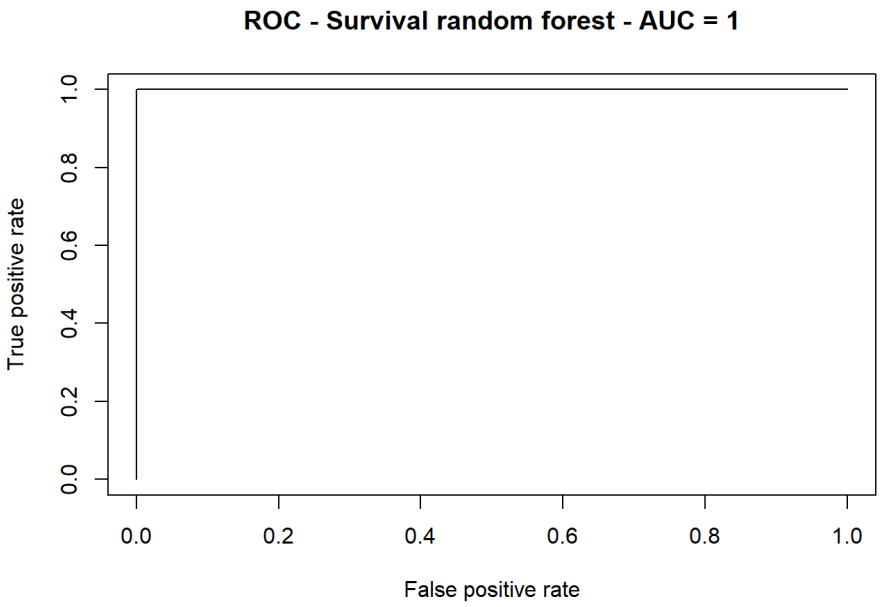
Le code ci-dessous sert à créer l'arbre. Il est implémenté avec la même méthode de validation croisée que précédemment.

```
#création de l'arbre
rforestmod = rfsrc(Surv(time, status) ~ . -lessthan24,
                   data = wpbcProcessd,
                   ntree = 1000,
                   importance = "none",
                   nsplit = 1)

#prédiction
rforestmod.pred = predict( rforestmod,
                        newdata = wpbcProcessd[-Train$Resample1,],
                        importance = "none" )


#identification de la ligne au temps d'interet (24)
which(rforestmod.pred$time.interest==24)

#data a 24 (avec 1- pour la proba qui nous interesse)
1-rforestmod.pred$survival[,which(rforestmod.pred$time.interest==24)]
```

Remarques

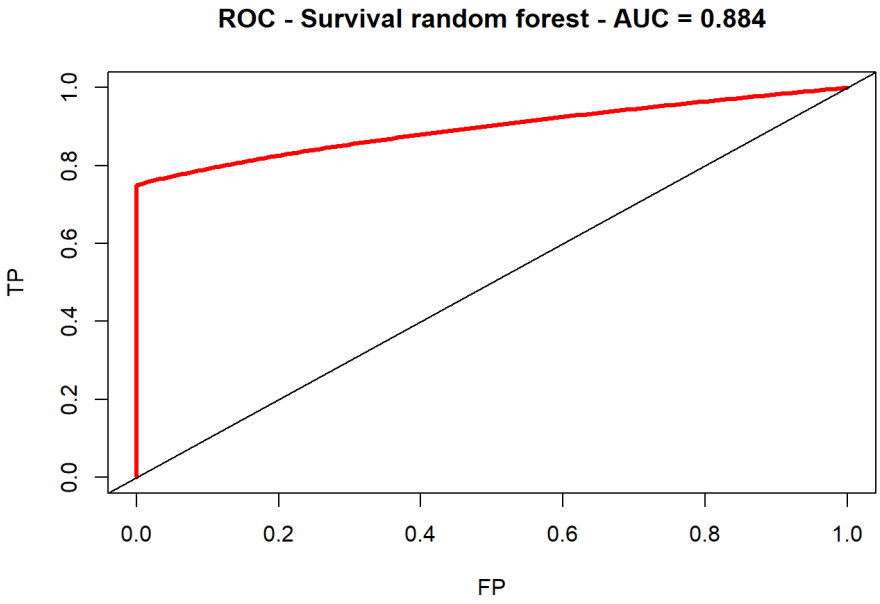
- Lors de la validation croisée, certaines itérations de l'arbre ne "fonctionnent pas" et aucun individu n'est classé au temps d'intérêt. De plus, plus l'échantillon d'entrainement est petit, plus le nombre d'arbres faux est élevé.
- La mesure de l'AUC par validation croisée (50 itérations) avec les arbres qui fonctionnent est de 1, soit un classement parfait des prédictions

```
plot(perf.rforest, main = "ROC - Survival random forest - AUC = 1")
```



Nous pouvons également calculer la courbe ROC à l'aide du package `risksetROC` :

```
w.ROC = risksetROC(Stime = wpbcProcessd$time,
                   status = wpbcProcessd$status,
                   marker = rforestmod$predicted.oob,
                   predict.time = 24,
                   method = "Cox",
                   main ="ROC - Survival random forest - AUC = 0.884",
                   lwd = 3,
                   col = "red" )
```

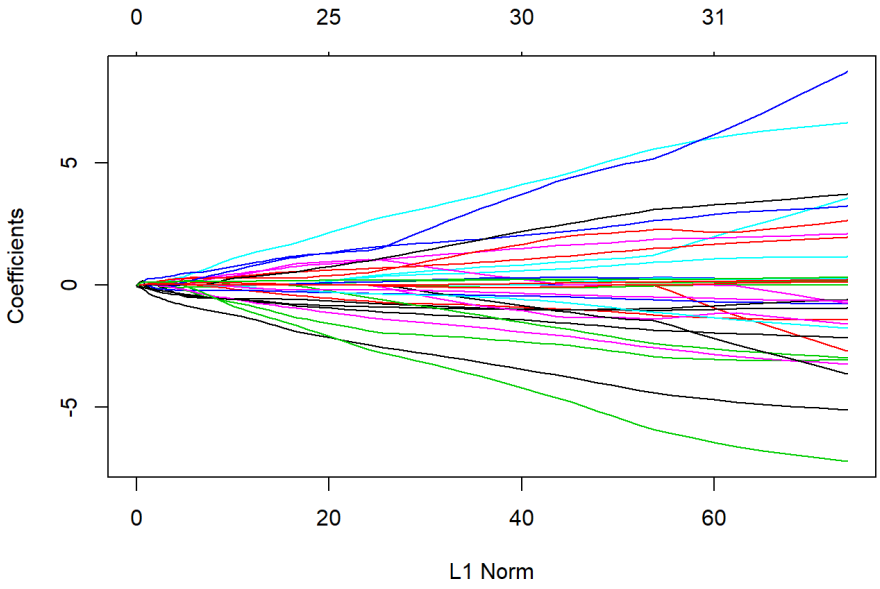


Avec ce package on obverse un AUC plus crédible et inférieur à 1. On serait tenté de croire que la différence d'AUC doit "être prises en compte" dans le phénomène d'arbre qui ne fonctionnaient pas relevé précédemment.

On constate que le taux de True Positif n'est supérieur à 75% pour aucun faux positif, contrairement au modèle de survie où la croissance est linéaire. Le random forest donne de bien meilleurs résultats et peut être préférée aux autres méthodes.

Annexe - Plots du Lasso

```
plot(lasso.mod)
```




```
plot(lasso.fit)
```