Rapport du projet

AimÃ© Cazeel

3 fÃ©vrier 2020

Résumé

Le CBSM ou Cognitive Behavioral Stress Management est un programme de gestion du stress qui mélange des exercices de relaxation, de restructuration cognitive et de dynamique de groupe.Si ce programme étudié principalement aux Etats-Unis s'avère efficace sur des maladies comme le cancer du sein ou le VIH, il n'existe cependant que peu d'études sur l'application du programme CBSM sur des patients atteints de maladies cardio-vasculaires et de son effet sur le stress, l'anxiété et les pensées intrusives.

Ainsi, l'objectif de ce document est d'étudier l'efficacité de ce programme auprès de patients atteints de maladies cardio-vasculaires. Pour cela, les patients ont été séparés en 2 groupes, un groupe témoin et un groupe participant au programme CBSM. Des mesures physiologiques ont étés relevés avant et après l'application du programme.

Une étude ultérieur ayant mis en valeur l'efficacité du programme concernant l'amélioration du stress perçu et de l'anxiété, nous cherchons ici à étudier l'efficacité du programme sur le stress ressenti.

<— DO TO COMPLETER AVEC CONCLUSION —!>

Introduction

Etat des lieux

Les maladies cardiovasculaires sont un ensemble de troubles affectant le coeur et les vaisseaux sanguins. Première cause de mortalité dans le monde selon l'OMS, elles nécessitent souvent une prise en charge lourde comprenant soutien psychologique et médicaments. Ainsi, il est important de trouver des solutions efficaces pour aider les personnes exposées à ces maladies.

Si l'alimentation et le tabagisme sont des facteurs aggravant connus, de nombreuses études mettent en évidence l'existence d'un lien entre stress et maladies cardio-vasculaires. Ainsi, proposer des solutions agissant sur le stress et ne nécessitant pas une prise en charge lourde ou médicamenteuse peut permettre de soulager les personnes atteintes de problèmes cardiovasculaires.

Or, il existe de nombreuses méthodes pour faciliter la gestion du stress, dont le programme CBSM.

CBSM ou Cognitive Behavioral Stress Management

Le CBSM est un programme de gestion du stress qui mélange des exercices de relaxation, de restructuration cognitive et de dynamique de groupe.L'objectif est de permettre aux patients d'avoir accès à des connaisances sur eux-même, sur le stress et son impact, et sur les réactions psychologiques qu'il peut susciter. Il est constitué de plusieurs séances en groupe ainsi que d'exercices à réaliser chez soi.

Si ce programme étudié principalement aux Etats-Unis s'avére efficace sur des maladies comme le cancer du sein ou le VIH, il n'existe cependant que peu d'études sur l'application du programme CBSM sur des patients atteints de maladies cardio-vasculaires et de son effet sur le stress, l'anxiété et les pensées intrusives.

Expérience

Ainsi en 2016 une expérience a été réalisé sur des patients atteints de pathologies cardiaques. Ces patients ont suivi le programme CBSM et ont répondus à des questionnaires afin d'évaluer leurs états psychologiques. De plus des relevés physiologiques ont aussi été réalisés. Les questionnaires doivent permettre d'évaluer le ressenti des patients par rapport au stress, tandis que les relevés physiologique permettent d'évaluer l'impact physique des interventions.

Objectif

Ce document fait suite au travail rédigé par Franck D'ALESSANDRO mettant en avant l'efficacité du programme CBSM sur la diminution du stress perçu par les patients ainsi que leur anxiété, ainsi que le travail de Aimé CAZEEL confirmant une partie de ces résultats.

Le but de cette étude est donc d'analyser l'influence du programme sur le stress "physique" des patients (que l'on distinguera du stress perçu).

```
## New names:
## * `Mean RR  (ms):` -> `Mean RR  (ms):...5`
## * `STD RR (ms):` -> `STD RR (ms):...6`
## * `Mean HR (1/min):` -> `Mean HR (1/min):...7`
## * `STD HR (1/min):` -> `STD HR (1/min):...8`
## * `RMSSD (ms):` -> `RMSSD (ms):...9`
## * ... and 47 more problems
```

Approche du problème

Participants

Au départ, l'expérience porte sur 150 participants ayant développés une maladie cardiaque. 50 personnes participent au programme CBSM, 50 personnes participent à des séances la relaxation et 50 personnes sont des individus "contrôle" ne suivant pas de programme particulier (hormis les soins). Ces personnes proviennent de différent lieux dans l'aglomération de Grenoble :

- Service de réadaptation cardiaque de l'hôpital Sud (Echirolles)
- Institut cardio-vascualire du groupe hospitalier mutualiste de Grenoble
- Réseau des pathologies vasculaires GRANTED à Saint-Martin-d'Hères
- Service de cardiologie du CHU La Tronche
- Service de diabétologie du CHU La Tronche

Les patients sont recrutés par les équipes soignantes, les personnes acceptant de participés sont placés aléatoirement dans l'un des 3 groupes.

Méthodes d'expérimentation

Le programme CBSM est constitué de plusieurs séances (2h par semaine) avec en plus des excercices à réaliser chez soi. Après l'ensemble des séances, les patients sont invités à répondre à des questionnaires mesurant leur perception du stress puis des mesure physiologiques sont prises. Ces mesures sont prises avec un module BIOPAC MP 150 qui va permettre de relever plusieurs variables.

Les mesures et les réponses aux questionnaires sont pris à différents moments :

- T0 : avant le début des séances CBSM
- T1 : à la fin des 10 semaines d'interventions
- T2 : 6 mois après l'intervention

Mesure physiologique : HRV ou Heart Rate Variability

Les recherches en psychophysiologie intègrent de plus en plus d'étude sur la variabilité du rythme cardiaque (HRV). En effet, il existe un lien entre le système nerveux parasympathique (lié à la régulation cardiaque) et de nombreux phénomènes psychophysiologique. Le HRV est d'ailleurs utilisé pour prédire les risques de mortalité provenant de cause mental ou physique.

Un relevé du HRV est simple à mettre en place et sans douleur, d'où son utilisation répandue. Parmis les nombreuses variables étudiables, celles d'intérêts sont :

- RMSSD (Root Mean Square of Succesive differences) dont les variations sont dépendantes du tonus vagal (activité du nerf vague, composant du sytème parasympathique contrôlant les activités involontaires des organes).
- HF (High Frenquencies) dont les variations proviennent aussi du tonus vagal mais peuvent être influencé par la respiration.
- LF (Low Frequencies) ainsi que le rapport LF/HF, dont les variations dépendent de divers éléments dont le système sympathique (responsable du rythme cardiaque mais aussi de la contraction des muscles lisses) et le tonus vagal.

Bien que facile à relever, le HRV est sujet à des erreurs de mesures ou à des modifications de celui-ci dû à des facteurs externes pouvant le rendre difficile à étudier (caféine etc. . .)

Dans les études statistiques, le HRV est très souvent utlisé comme une variable les régressions ou les corrélations, permettant souvent de distinguer des groupes selons d'autres critères (comme des différences individuelles). Parfois, le HRV peut être considéré comme une variable dépendante en créant 2 groupes séparés par la médiane. A ce moment, on suppose que le HRV illustre des particularité individuelles (on sait par exemple que le controle vagal est partiellement héritable, ce qui peut en faire une information propre à chaque individus et non dépendantes de variables externes).

Concernant la distribution des variables liées au HRV, la question de la normalité des variables est discutée. Mais des études tendent à observer une non normalité de la distribution de ces variables. La transformation logarithmique est alors une procédure courante pour remédier à ce problème.

Limitations

Erreurs

Plusieurs problèmes apparaissent dans notre méthodologie :

- Si au départ, nous devions avoir 3 groupes, au final, seulement 2 groupes existent effectivement : les groupes CBSM et CONTROLE. Ces deux groupes sont de tailles différentes. De plus, le groupe CONTROLE réalise des exercices de relaxation.
- Des erreurs de mesures peuvent fausser nos résultats (erreurs de manipulation).De plus, nous faisons face à des individus sous médication ayant une pathologie cardiaque, les chances d'obtenir des valeurs aberantes sont grandes.

Durées de l'expérimentation

Les individus de l'expérience ont été exposé au programme pendant 10 semaines, sans obligations d'être présent à toute les séances, ni obligation à réaliser les exercices à faire chez soi, cela limite donc l'influence du programme sur nos patients.

Enfin, cette étude est basée sur le bénévolat. Hormis la volonté des patients, il n'y avait que peu d'obligations de poursuivre l'étude. Ainsi, nous observons une très grande absence de réponse pour les temps T2 (plusieurs mois après expérience). Nous avons aussi des patients absents lors des premières mesures, mais présent après etc.

Au final, au vu du faible de nombre de réponse pour le temps T2, nous avons décider de limiter nos analyses à T0 et T1, ne nous permettant pas de constater des résultats sur le long terme.

Objectifs de l'analyse

Ainsi, nos objectifs sont donc les suivants :

- Corriger les valeurs aberantes de certains patients
- Déterminer un modèle de prédiction de l'amélioration de l'état d'un individu.

Méthodes pour l'analyse

Imputation

TODO

SVM

Cette méthode sera utilisé dans le cadre de la recherche d'un modèle de prédiction.

Les séparateurs à vaste marge sont des techniques d'apprentissage supervisé destinées à résoudre des problèmes de discimination et de regression. Le principe est de chercher une frontière entre nos groups qui maximisent la marge, c'est à dire la distance entre la frontière et les éléments les plus proches.

Random Forest

Cette méthode sera utilisé dans le cadre de la recherche d'un modèle de prédiction

Les forêts d'arbres de décisions sont des méthodes d'apprentissage basées sur les concepts de bagging (ou bootstrap) et de sous-espaces aléatoire. Le principe est d'effectuer un apprentissage d'arbre de décision sur des sous-ensembles de données.

L'arbre de décision est une méthode d'apprentissage qui cherche à découper nos données en sous-ensemble par des critères maxisimisant (selon les algorithmes) la différence entre les sous-ensembles. On sépare nos données selon une variable puis l'on réintère ce processus pour chaque sous-groupe.

Ces méthodes sont assez courantes parmis les méthodes d'apprentissage supervisée.

Leave-One-Out

La validation croisée est une méthode d'estimation de la fiabilité basé sur une technique d'échantillonage. Le principe est de découper nos données d'entrainements en plusieurs partition. A chaque itération, on utilisera une partition comme test et les autres comme données pour entrainer le modèle.

Le leave-one-out est un cas particulier de la validation croisée où l'on considère chaque individus comme une partition. Pour des jeux de données restreints, il permet de réaliser une validation croisée sans avoir un problème lors de l'entrainement (où l'on risquerai d'avoir très peu de données en réduisant plus encore la taille de l'échantillon d'apprentissage).

Bootstrap

Avec le peu de données que nous risquons d'avoir au final, nous avons besoin d'une méthode nous assurant de contrôler la "robustesse" de nos méthodes. Le bootstrap est une méthode qui se base sur le reéchantillonage avec remise. Il s'agit de considérer nos données comme la distribution de la population (pour nous, les personnes atteintes de maladie cardiaque) et donc de tirer plusieurs échantillons au sein de cette distribution.

Dans notre cas, le bootstrap est un moyen de contourner notre manque de données. Néanmoins, cette méthode dépend beaucoup de nos données de départ pour être efficace. Il faudra utiliser celle-ci avec parcimonie.

Premières analyses

Données manquantes et premiers constats

Le premier problème dans notre jeu de données est la présence de données manquantes. Certaines personnes n'ont des données que concernant le temps T0 ou T1.

En effet, sur les 105 individus du jeu de données de départ, 23 n'ont pas de valeurs à T0, 48 n'ont pas de données à T1 et 57 n'ont pas de données à T0 et T1.

Si nous pouvons par la suite discuter de l'intérêt de conserver des individus à qui il manque des données à T0 ou T1, il semble relativement logique de ce débarasser des individus sans données à T0 et T1 (il s'agit des personnes présentent seulement au temps T2), qui seront sans intérêt dans notre cas.

Cela nous laisse alors avec 90, dont 0 sans données à T0 et 0 sans données à T1.

Valeurs extrêmes

Le second problème est la présence de valeurs abérantes.

En effet, sur les 81 individus du jeu de données de départ avec des valeurs à T0, 15 ont aun moins une valeur extrème, et sur les 57 individus avec des valeurs à T1, 6 ont au moins une valeur extrême.

Imputation des données

Notre but premier est de remplacer les valeurs extrêmes chez nos individus. Au départ notre population est dans le même état (pas de groupe particulier à l'intérieur <– A VERIFIER SI LE TEMPS –!>), nous pouvons donc utiliser l'ensemble des données disponible pour réaliser une imputation de données sans trop de problème.

Nous allons exclure les données manquantes de l'imputation. Imputer les valeurs manquantes au temps T1 reviendrait à prédire l'amélioration ou non de l'état de santé des patients, ce qui est certe l'objectif, mais nous nous limitons à indiquer si oui ou non il y a amélioration, nous ne souhaitons pas prédire l'ensemble des valeurs physiologique. Nous risquons alors d'influencer de façon non négligeable nos données.

Nous nous concentrerons donc seulement sur les valeurs abérantes

Imputation à T0

En T0, nous avons environ 81 observations et 10 variables avec 35 valeurs manquantes. Vu la taille de nos données, nous ne pouvons pas nous permettre de supprimer ces valeurs manquantes, donc nous allons procéder à une imputation.

###Répartition des NA par variables Regardons le taux de valeurs abérantes pour chaque varialbe à T0:

```
imput_T0=imput_T0[,-1]
sapply(imput_T0, function(x) sum(is.na(x)))
```

```
##             T0_Mean_RR_ms              T0_STD_RR_ms
##                         2                         6
##          T0_Mean_HR_1_min           T0_STD_HR_1_min
##                         4                         4
##               T0_RMSSD_ms               T0_VLF_ms2
##                         6                         1
##                 T0_LF_ms2                 T0_HF_ms2
##                         2                         4
##       T0_Total_power_ms2 T0_Frequence_Respiratoire
##                         4                         2
```

```
#plot the missing values
#nhanes_miss = aggr(imput_T0, col=mdc(1:2), numbers=TRUE, sortVars=TRUE, labels=names(imput_T0), cex.ax
```

On observe que de manière générale, les variables n'ont pas un taux trop important de valeurs abérantes. Nous allons donc imputer l'ensemble des variables. Considérons notre tableau de données sans les NA, puis créons artificiellement et aléatoirement 5% de valeurs manquantes.

```
set.seed(10000)
#dim(imput_T0)
#sum(is.na(imput_T0))
tab=na.omit(imput_T0)
tabb <- prodNA (tab, noNA = 0.05)
```

Avant d'imputer, comparons tout d'abord plusieurs techniques d'imputations pour voir laquelle est la meilleure. Nous allons utiliser le package MICE. MICE (Mutivariate Imputation via Chained Equations) est un package qui permet d'utiliser plusieurs techniques d'imputations pour traiter les données manquantes.

Pour notre imputation de données nous avions choisi de faire un premier test sur de différentes méthodes d'imputation sur le jeu de données. En première étape il s'agit de récupérer une partie du jeu de données assez complète afin de recréer des valeurs manquantes et de comparer plus tard les résultats des différentes méthodes d'imputations que nous allons effectuer sur le jeu de données.

La fonction MICE de la librairie mice qui renferme en elle différentes méthodes d'imputations nous servira pour ce test. De cette liste de fonction, 5 types d'imputations ont été expérimentées. Il s'agit de la Predictive mean matching PMM, la moyenne MEAN, random forest RF, Random sample for observed values SAMPLE et une dernière le Bayesian Linear Regression NORM.

Le critère de comparaison est axé sur la minimisation des erreurs de prédiction le RMSE entre les valeurs prédites par les imputations et les valeurs réelles. Et aussi en observant les statistiques descriptives avant et après les imputations.

Pour le choix du meilleur modèle d'imputation, choisir celle qui minimise au mieux sur l'ensemble des 10 variables et aussi en gardant un oeil sur la variable RM_SSD.

A la fin de ce test nous avons retenu 3 types d'imputations qui marchent le mieux. Le PMM et le Bayesian linear régression et la moyenne. Nous avions donc dégagé pour une première étape les trois méthodes qui seront choisies comme méthode à analyser.

La régression linéaire bayésienne est une approche de la régression linéaire dans laquelle l'analyse statistique est entreprise dans le contexte de l'inférence bayésienne.

La méthode de la moyenne utilise la moyenne obtenue sans les valeurs manquantes et les impute la même valeur à l'ensemble des valeurs manquantes

Par rapport aux méthodes standard basées sur la régression linéaire et la distribution normale, PMM produit des valeurs imputées qui ressemblent beaucoup plus à des valeurs réelles. Si la variable d'origine est asymétrique, les valeurs imputées seront également asymétriques. Si la variable d'origine est limitée par 0 et 100, les valeurs imputées seront également limitées par 0 et 100. Et si les valeurs réelles sont discrètes (comme le nombre d'enfants), les valeurs imputées seront également discrètes. En effet, les valeurs imputées sont des valeurs réelles qui sont «empruntées» à des individus disposant de données réelles. Elle fait donc moyenne des valeurs de ces observations qui le ressemblent.

###Méthode par la Moyenne (Mean) Nous avons utilisé 3 paramètres: - m= le nombre imputation par NA - maxit= nombre d'itérations - method= méthode utilisée

```
#dim(tab)
imputed_Data_mean <- mice(tabb, m=1, maxit = 5, method = c('mean'), seed = 500)
```

```
##
##  iter imp variable
```

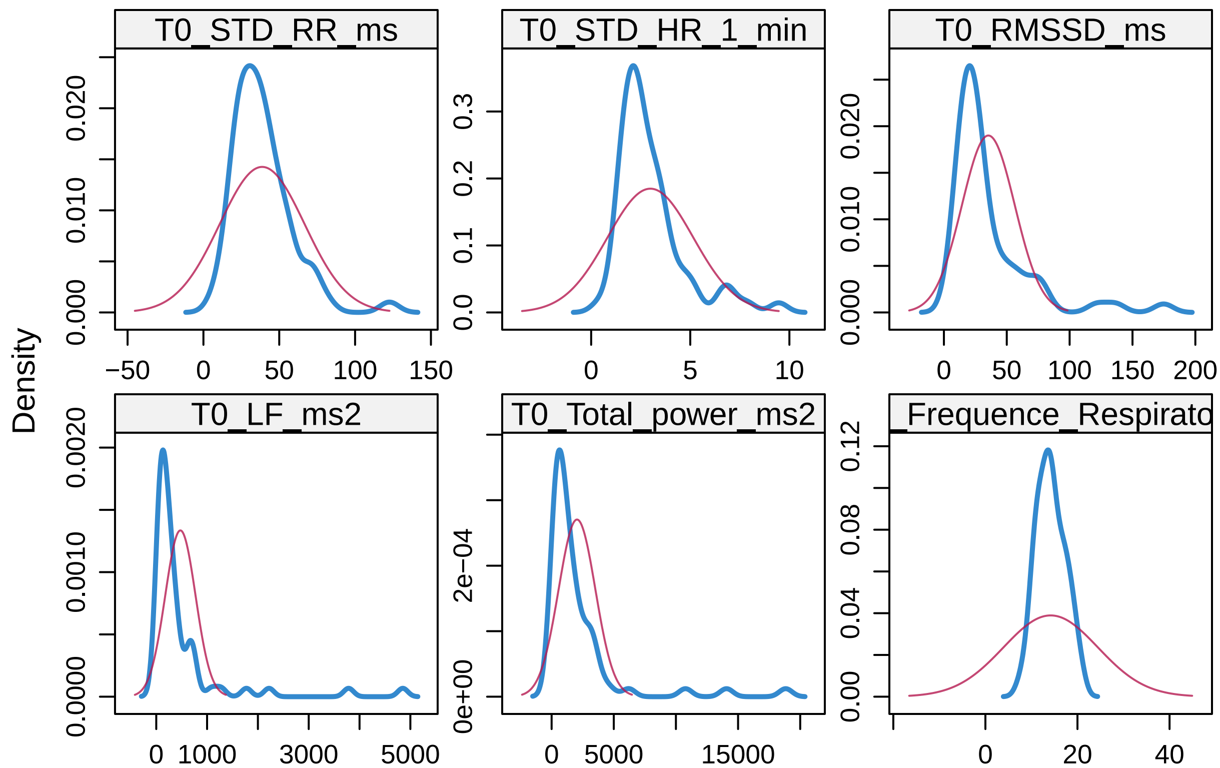
```
##   1   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   2   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   3   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   4   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   5   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
```

Voici un summary de l'imputation:

```
summary(imputed_Data_mean)
```

```
## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
##             T0_Mean_RR_ms              T0_STD_RR_ms
##                    "mean"                    "mean"
##          T0_Mean_HR_1_min           T0_STD_HR_1_min
##                    "mean"                    "mean"
##               T0_RMSSD_ms                T0_VLF_ms2
##                    "mean"                    "mean"
##                 T0_LF_ms2                 T0_HF_ms2
##                    "mean"                    "mean"
##        T0_Total_power_ms2 T0_Frequence_Respiratoire
##                    "mean"                    "mean"
## PredictorMatrix:
##                  T0_Mean_RR_ms T0_STD_RR_ms T0_Mean_HR_1_min
## T0_Mean_RR_ms                0            1                1
## T0_STD_RR_ms                 1            0                1
## T0_Mean_HR_1_min             1            1                0
## T0_STD_HR_1_min              1            1                1
## T0_RMSSD_ms                  1            1                1
## T0_VLF_ms2                   1            1                1
##                  T0_STD_HR_1_min T0_RMSSD_ms T0_VLF_ms2 T0_LF_ms2
## T0_Mean_RR_ms                  1           1          1         1
## T0_STD_RR_ms                   1           1          1         1
## T0_Mean_HR_1_min               1           1          1         1
## T0_STD_HR_1_min                0           1          1         1
## T0_RMSSD_ms                    1           0          1         1
## T0_VLF_ms2                     1           1          0         1
##                  T0_HF_ms2 T0_Total_power_ms2 T0_Frequence_Respiratoire
## T0_Mean_RR_ms            1                  1                         1
## T0_STD_RR_ms             1                  1                         1
## T0_Mean_HR_1_min         1                  1                         1
## T0_STD_HR_1_min          1                  1                         1
## T0_RMSSD_ms              1                  1                         1
## T0_VLF_ms2               1                  1                         1
```

```
densityplot(imputed_Data_mean)
```



En regardant les fonctions de densité, nous remarquons que, pour la plupart des variables, les fonctions de densités des valeurs imputées(en rouge) sont assez différentes des vraies valeurs(en bleu). Après imputation, nous allons calculer le MAE, RMSE, et MAPE afin de voir l' accuracy de chaque méthode d'imputation.

A COMPLETER DEFINITION ET FORMULES

DEFINITION MAE MRMSE ET MAPE!!! RMSE (Root Mean Squared Error) and MAE(Mean Absolute Error), represent two different measures of the model prediction error. The lower the RMSE and the MAE, the better the model. Voici les résultats obtenus après imputation par la moyenne: #########

```
imput_mean <- complete(imputed_Data_mean)

mae_mean<-c()
rmse_mean<-c()
mape_mean<-c()
for (i in c(1:10)){
  #mae_mean[i]<-round(mae(imput_mean[,i],tab[,i]),2)
   mae_mean[i]<-round(mae(tab[,i],imput_mean[,i]),2)

  #rmse_mean[i]<-round(rmse(imput_mean[,i],tab[,i]),2)
      rmse_mean[i]<-round(rmse(tab[,i],imput_mean[,i]),2)

  #mape_mean[i]<-round(mape(imput_mean[,i],tab[,i]),2)
    mape_mean[i]<-round(mape(tab[,i],imput_mean[,i]),2)

}
Accuracy_mean<-cbind (mae=mae_mean, rmse=rmse_mean, mape=mape_mean)
```

```
Accuracy_mean
```

```
##         mae   rmse mape
##  [1,]  2.79  18.26 0.00
##  [2,]  0.43   2.24 0.02
##  [3,]  0.04   0.28 0.00
##  [4,]  0.04   0.19 0.02
##  [5,]  1.25   3.96 0.06
##  [6,] 12.11  75.23 0.03
##  [7,] 24.38  88.97 2.11
##  [8,]  7.64  62.08 0.04
##  [9,] 53.11 257.90 0.08
## [10,]  0.17   0.88 0.02
```

###Méthode par Predictive Mean Matching (pmm) pmm=? à definir

```
imputed_Data_pmm <- mice(tabb, m=1, maxit = 5, method = 'pmm', seed = 500)
```

```
##
##  iter imp variable
##   1   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   2   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   3   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   4   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   5   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
```

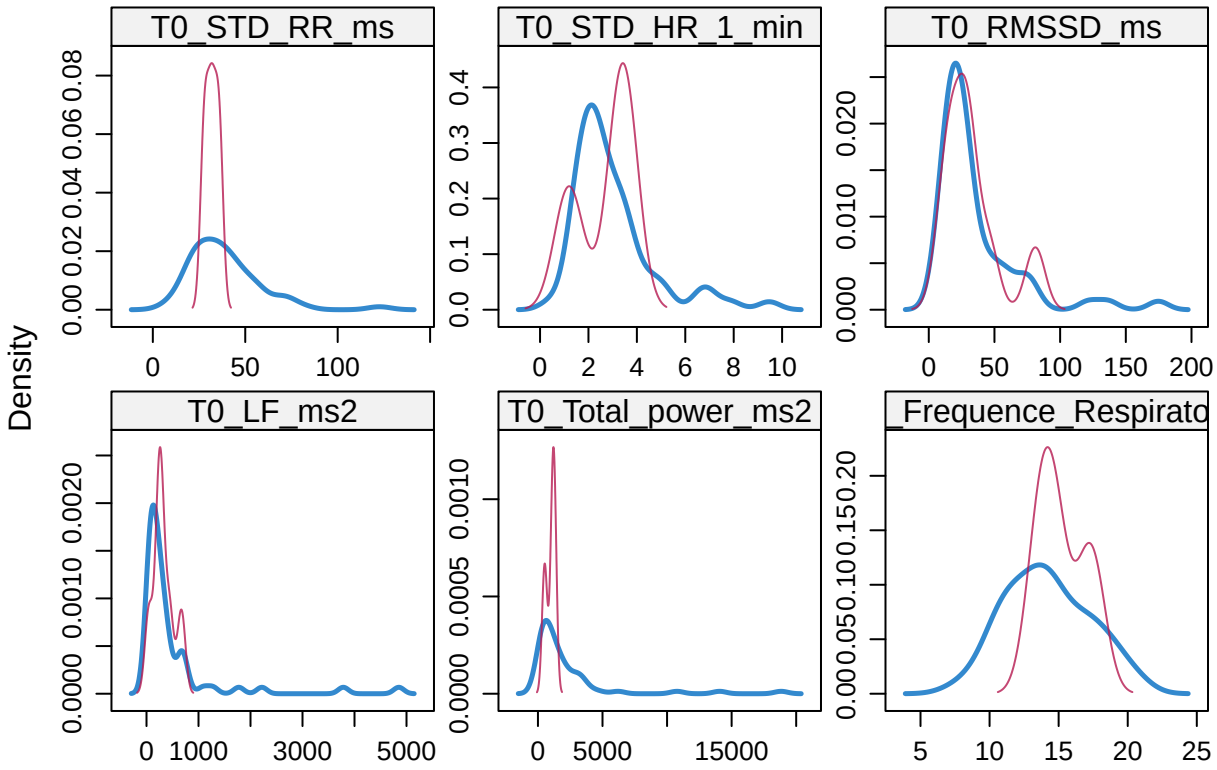
```
## Warning: Number of logged events: 19
```

```
summary(imputed_Data_pmm)
```

```
## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
##             T0_Mean_RR_ms              T0_STD_RR_ms
##                     "pmm"                     "pmm"
##          T0_Mean_HR_1_min           T0_STD_HR_1_min
##                     "pmm"                     "pmm"
##               T0_RMSSD_ms                T0_VLF_ms2
##                     "pmm"                     "pmm"
##                 T0_LF_ms2                 T0_HF_ms2
##                     "pmm"                     "pmm"
##        T0_Total_power_ms2 T0_Frequence_Respiratoire
##                     "pmm"                     "pmm"
## PredictorMatrix:
##                  T0_Mean_RR_ms T0_STD_RR_ms T0_Mean_HR_1_min
## T0_Mean_RR_ms                0            1                1
## T0_STD_RR_ms                 1            0                1
## T0_Mean_HR_1_min             1            1                0
## T0_STD_HR_1_min              1            1                1
## T0_RMSSD_ms                  1            1                1
## T0_VLF_ms2                   1            1                1
##                  T0_STD_HR_1_min T0_RMSSD_ms T0_VLF_ms2 T0_LF_ms2
## T0_Mean_RR_ms                  1           1          1         1
## T0_STD_RR_ms                   1           1          1         1
## T0_Mean_HR_1_min               1           1          1         1
## T0_STD_HR_1_min                0           1          1         1
```

```
## T0_RMSSD_ms                     1           0          1         1
## T0_VLF_ms2                      1           1          0         1
##                   T0_HF_ms2 T0_Total_power_ms2 T0_Frequence_Respiratoire
## T0_Mean_RR_ms             1                  1                         1
## T0_STD_RR_ms              1                  1                         1
## T0_Mean_HR_1_min          1                  1                         1
## T0_STD_HR_1_min           1                  1                         1
## T0_RMSSD_ms               1                  1                         1
## T0_VLF_ms2                1                  1                         1
## Number of logged events:  19
##   it im                        dep meth                out
## 1  2  1 T0_Frequence_Respiratoire  pmm T0_Total_power_ms2
## 2  3  1             T0_Mean_RR_ms  pmm T0_Total_power_ms2
## 3  3  1              T0_STD_RR_ms  pmm T0_Total_power_ms2
## 4  3  1          T0_Mean_HR_1_min  pmm T0_Total_power_ms2
## 5  3  1           T0_STD_HR_1_min  pmm T0_Total_power_ms2
## 6  3  1               T0_RMSSD_ms  pmm T0_Total_power_ms2
```

```
densityplot(imputed_Data_pmm)
```




```
imput_pmm <- complete(imputed_Data_pmm)
mae_pmm<-c()
rmse_pmm<-c()
mape_pmm<-c()
for (i in c(1:10)){
    mae_pmm[i]<-round(mae(tab[,i],imput_pmm[,i]),2)
    rmse_pmm[i]<-round(rmse(tab[,i],imput_pmm[,i]),2)
```

```
    mape_pmm[i]<-round(mape(tab[,i],imput_pmm[,i]),2)

}
Accuracy_pmm<-cbind (mae=mae_pmm, rmse=rmse_pmm, mape=mape_pmm)
Accuracy_pmm
```

```
##        mae  rmse mape
##  [1,] 0.08  0.50 0.00
##  [2,] 0.12  0.63 0.00
##  [3,] 0.02  0.10 0.00
##  [4,] 0.03  0.13 0.01
##  [5,] 1.63  6.08 0.05
##  [6,] 1.12  6.48 0.00
##  [7,] 7.72 44.76 0.24
##  [8,] 0.01  0.07 0.00
##  [9,] 5.50 41.27 0.01
## [10,] 0.24  1.26 0.02
```

Méthode par Random Forest (rf)

```
imputed_Data_rf <- mice(tabb, m=1, maxit = 5, method = 'rf', seed = 500)
```

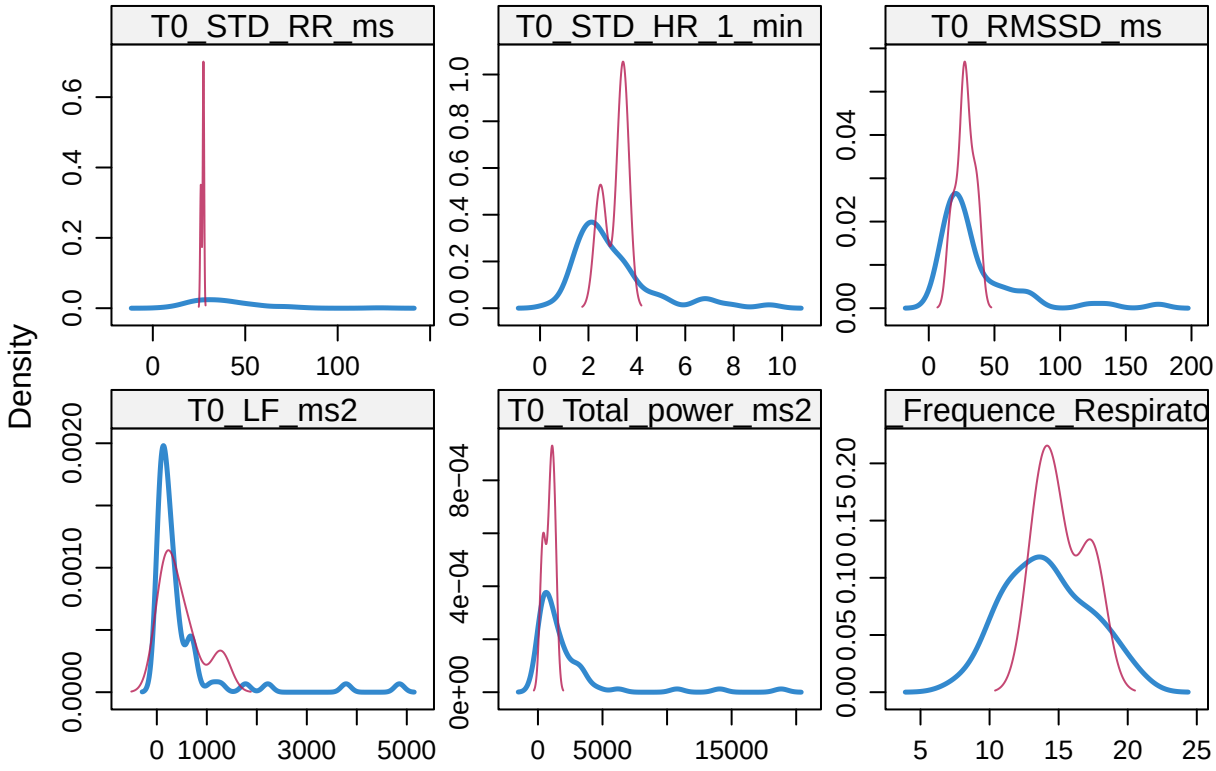
```
##
##  iter imp variable
##   1   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   2   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   3   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   4   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   5   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
```

```
summary(imputed_Data_rf)
```

```
## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
##             T0_Mean_RR_ms              T0_STD_RR_ms
##                      "rf"                      "rf"
##          T0_Mean_HR_1_min           T0_STD_HR_1_min
##                      "rf"                      "rf"
##               T0_RMSSD_ms                T0_VLF_ms2
##                      "rf"                      "rf"
##                 T0_LF_ms2                 T0_HF_ms2
##                      "rf"                      "rf"
##        T0_Total_power_ms2 T0_Frequence_Respiratoire
##                      "rf"                      "rf"
## PredictorMatrix:
##                  T0_Mean_RR_ms T0_STD_RR_ms T0_Mean_HR_1_min
## T0_Mean_RR_ms                0            1                1
## T0_STD_RR_ms                 1            0                1
## T0_Mean_HR_1_min             1            1                0
## T0_STD_HR_1_min              1            1                1
## T0_RMSSD_ms                  1            1                1
## T0_VLF_ms2                   1            1                1
##                  T0_STD_HR_1_min T0_RMSSD_ms T0_VLF_ms2 T0_LF_ms2
```

```
## T0_Mean_RR_ms                  1           1           1           1
## T0_STD_RR_ms                   1           1           1           1
## T0_Mean_HR_1_min               1           1           1           1
## T0_STD_HR_1_min                0           1           1           1
## T0_RMSSD_ms                    1           0           1           1
## T0_VLF_ms2                     1           1           0           1
##                  T0_HF_ms2 T0_Total_power_ms2 T0_Frequence_Respiratoire
## T0_Mean_RR_ms            1                  1                         1
## T0_STD_RR_ms             1                  1                         1
## T0_Mean_HR_1_min         1                  1                         1
## T0_STD_HR_1_min          1                  1                         1
## T0_RMSSD_ms              1                  1                         1
## T0_VLF_ms2               1                  1                         1
```

```
densityplot(imputed_Data_rf)
```




```
imput_rf <- complete(imputed_Data_rf)
mae_rf<-c()
rmse_rf<-c()
mape_rf<-c()
for (i in c(1:10)){
    mae_rf[i]<-round(mae(tab[,i],imput_rf[,i]),2)
    rmse_rf[i]<-round(rmse(tab[,i],imput_rf[,i]),2)
    mape_rf[i]<-round(mape(tab[,i],imput_rf[,i]),2)

}
Accuracy_rf<-cbind (mae=mae_rf, rmse=rmse_rf, mape=mape_rf)
```

```
Accuracy_rf
```

```
##         mae   rmse mape
##  [1,]  2.31  14.70 0.00
##  [2,]  0.17   1.02 0.01
##  [3,]  0.05   0.35 0.00
##  [4,]  0.04   0.21 0.02
##  [5,]  0.86   3.09 0.03
##  [6,] 18.33 126.26 0.03
##  [7,] 34.28 144.96 0.50
##  [8,]  0.82   6.66 0.00
##  [9,] 13.91  70.60 0.02
## [10,]  0.24   1.27 0.02
```

Méthode par Classification and Regression trees (cart)

```
imputed_Data_cart <- mice(tabb, m=1, maxit = 5, method = 'cart', seed = 500)
```

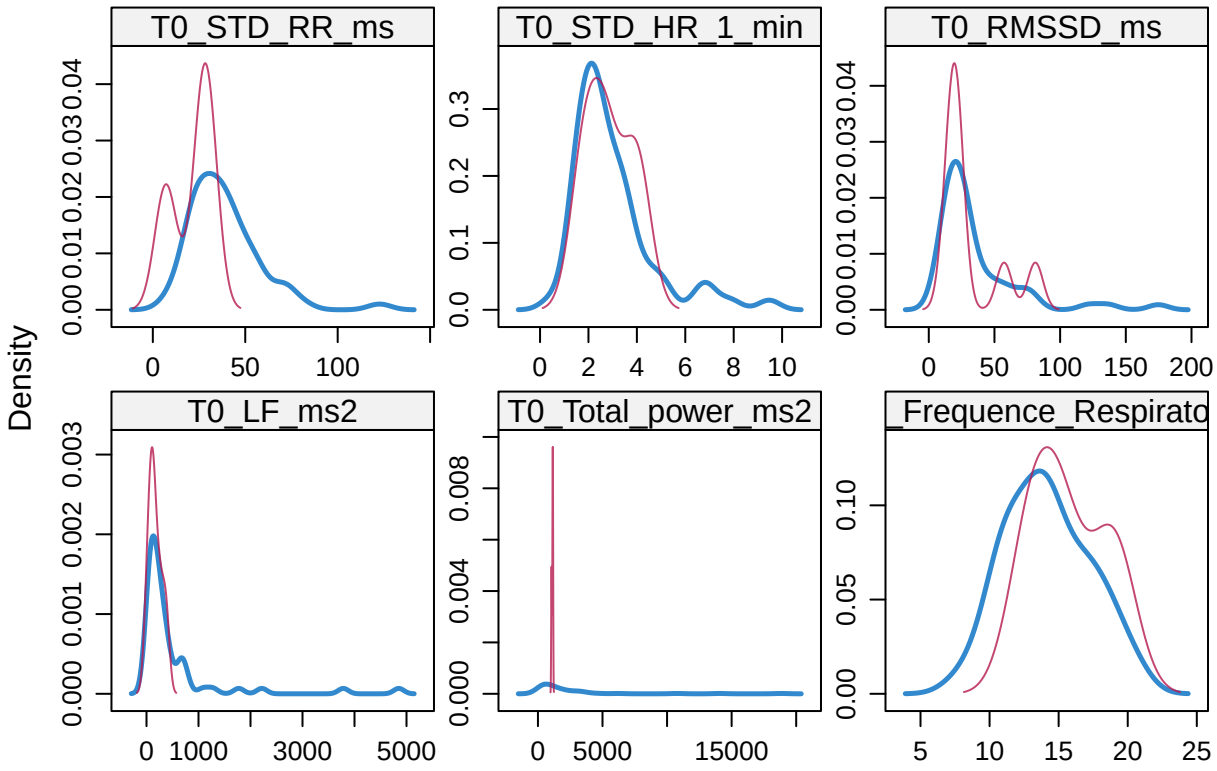
```
##
##  iter imp variable
##   1   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   2   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   3   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   4   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   5   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
```

```
summary(imputed_Data_cart)
```

```
## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
##             T0_Mean_RR_ms              T0_STD_RR_ms
##                    "cart"                    "cart"
##          T0_Mean_HR_1_min           T0_STD_HR_1_min
##                    "cart"                    "cart"
##               T0_RMSSD_ms                T0_VLF_ms2
##                    "cart"                    "cart"
##                 T0_LF_ms2                 T0_HF_ms2
##                    "cart"                    "cart"
##       T0_Total_power_ms2 T0_Frequence_Respiratoire
##                    "cart"                    "cart"
## PredictorMatrix:
##                   T0_Mean_RR_ms T0_STD_RR_ms T0_Mean_HR_1_min
## T0_Mean_RR_ms                 0            1                 1
## T0_STD_RR_ms                  1            0                 1
## T0_Mean_HR_1_min              1            1                 0
## T0_STD_HR_1_min               1            1                 1
## T0_RMSSD_ms                   1            1                 1
## T0_VLF_ms2                    1            1                 1
##                   T0_STD_HR_1_min T0_RMSSD_ms T0_VLF_ms2 T0_LF_ms2
## T0_Mean_RR_ms                   1           1          1         1
## T0_STD_RR_ms                    1           1          1         1
## T0_Mean_HR_1_min                1           1          1         1
## T0_STD_HR_1_min                 0           1          1         1
## T0_RMSSD_ms                     1           0          1         1
```

```
## T0_VLF_ms2                    1            1           0         1
##                   T0_HF_ms2 T0_Total_power_ms2 T0_Frequence_Respiratoire
## T0_Mean_RR_ms             1                 1                          1
## T0_STD_RR_ms              1                 1                          1
## T0_Mean_HR_1_min          1                 1                          1
## T0_STD_HR_1_min           1                 1                          1
## T0_RMSSD_ms               1                 1                          1
## T0_VLF_ms2                1                 1                          1
```

```
densityplot(imputed_Data_cart)
```




```
##          mae  rmse mape
##  [1,]  0.88  5.24 0.00
##  [2,]  0.49  3.51 0.01
##  [3,]  0.05  0.31 0.00
##  [4,]  0.00  0.01 0.00
##  [5,]  1.78  6.48 0.06
##  [6,] 13.27 88.90 0.02
##  [7,] 16.03 69.92 0.09
##  [8,]  0.05  0.37 0.00
##  [9,] 15.03 78.46 0.02
## [10,]  0.13  0.67 0.01
```

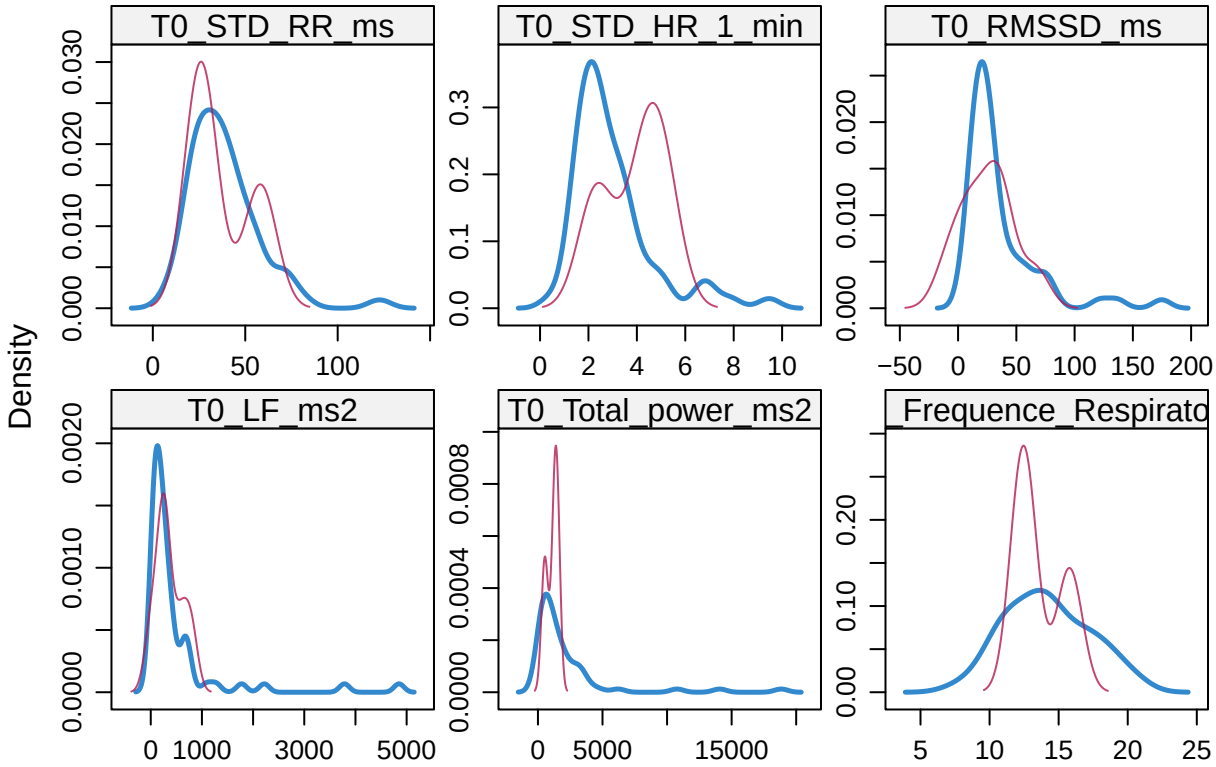
Méthode par Bayesian linear regression

```
##
##  iter imp variable
```

```
##   1   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   2   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   3   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   4   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
##   5   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0_
```

```
## Warning: Number of logged events: 18
```

```
## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
##             T0_Mean_RR_ms              T0_STD_RR_ms
##                    "norm"                    "norm"
##          T0_Mean_HR_1_min           T0_STD_HR_1_min
##                    "norm"                    "norm"
##               T0_RMSSD_ms                T0_VLF_ms2
##                    "norm"                    "norm"
##                 T0_LF_ms2                 T0_HF_ms2
##                    "norm"                    "norm"
##        T0_Total_power_ms2 T0_Frequence_Respiratoire
##                    "norm"                    "norm"
## PredictorMatrix:
##                  T0_Mean_RR_ms T0_STD_RR_ms T0_Mean_HR_1_min
## T0_Mean_RR_ms                0            1                1
## T0_STD_RR_ms                 1            0                1
## T0_Mean_HR_1_min             1            1                0
## T0_STD_HR_1_min              1            1                1
## T0_RMSSD_ms                  1            1                1
## T0_VLF_ms2                   1            1                1
##                  T0_STD_HR_1_min T0_RMSSD_ms T0_VLF_ms2 T0_LF_ms2
## T0_Mean_RR_ms                  1           1          1         1
## T0_STD_RR_ms                   1           1          1         1
## T0_Mean_HR_1_min               1           1          1         1
## T0_STD_HR_1_min                0           1          1         1
## T0_RMSSD_ms                    1           0          1         1
## T0_VLF_ms2                     1           1          0         1
##                  T0_HF_ms2 T0_Total_power_ms2 T0_Frequence_Respiratoire
## T0_Mean_RR_ms            1                  1                         1
## T0_STD_RR_ms             1                  1                         1
## T0_Mean_HR_1_min         1                  1                         1
## T0_STD_HR_1_min          1                  1                         1
## T0_RMSSD_ms              1                  1                         1
## T0_VLF_ms2               1                  1                         1
## Number of logged events:  18
##   it im                       dep meth               out
## 1  2  1 T0_Frequence_Respiratoire norm T0_Total_power_ms2
## 2  3  1             T0_Mean_RR_ms norm T0_Total_power_ms2
## 3  3  1              T0_STD_RR_ms norm T0_Total_power_ms2
## 4  3  1          T0_Mean_HR_1_min norm T0_Total_power_ms2
## 5  3  1               T0_RMSSD_ms norm T0_Total_power_ms2
## 6  3  1 T0_Frequence_Respiratoire norm T0_Total_power_ms2
```

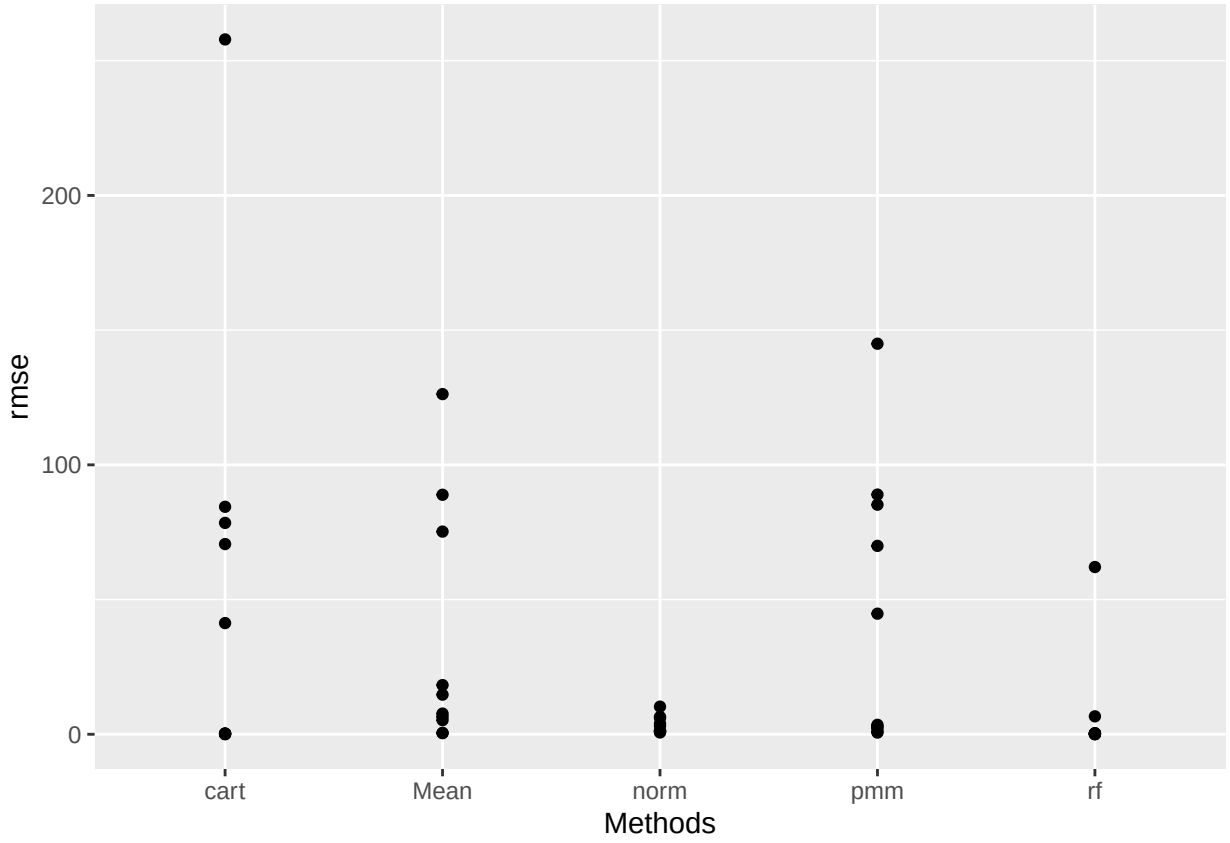



```
##          mae  rmse mape
##  [1,]   1.29  7.68 0.00
##  [2,]   0.39  2.82 0.01
##  [3,]   0.01  0.07 0.00
##  [4,]   0.05  0.30 0.02
##  [5,]   3.04 10.25 0.10
##  [6,]   0.06  0.44 0.00
##  [7,]  10.70 85.22 0.13
##  [8,]   0.01  0.07 0.00
##  [9,]  10.47 84.43 0.01
## [10,]   0.20  1.13 0.02
```

Comparaison des résultats

```
##    Accuracy_mean[, 2] Accuracy_pmm[, 2] Accuracy_rf[, 2]
## 1               18.26               0.50            14.70
## 2                2.24               0.63             1.02
## 3                0.28               0.10             0.35
## 4                0.19               0.13             0.21
## 5                3.96               6.08             3.09
## 6               75.23               6.48           126.26
## 7               88.97              44.76           144.96
## 8               62.08               0.07             6.66
## 9              257.90              41.27            70.60
## 10               0.88               1.26             1.27
##    Accuracy_cart[, 2] Accuracy_norm[, 2]
```

```
## 1                5.24              7.68
## 2                3.51              2.82
## 3                0.31              0.07
## 4                0.01              0.30
## 5                6.48             10.25
## 6               88.90              0.44
## 7               69.92             85.22
## 8                0.37              0.07
## 9               78.46             84.43
## 10               0.67              1.13
```



En faisant une analyse des cinq méthodologies d'imputations nous observons que les deux premières méthodes à savoir la regression linéaire bayesienne(norm) et le pmm minimise au mieux les variables. Mais pour en choisir qu'une à la fin nous analysons aussi les statistiques descriptives que fournissent ces données avant et après imputation et surtout pour la variable RMSSD en phase T1 qui servira plus tard pour la prédiction.

Comparaison des statistiques descriptives des méthodes

```
##  T0_Mean_RR_ms     T0_STD_RR_ms     T0_Mean_HR_1_min T0_STD_HR_1_min
##  Min.   : 654.9   Min.   :  7.107   Min.   :50.01    Min.   :0.4276
##  1st Qu.: 827.8   1st Qu.: 25.279   1st Qu.:60.42    1st Qu.:1.8964
##  Median : 908.1   Median : 35.295   Median :66.84    Median :2.4959
##  Mean   : 914.6   Mean   : 38.304   Mean   :66.93    Mean   :2.9802
##  3rd Qu.: 997.4   3rd Qu.: 45.589   3rd Qu.:72.59    3rd Qu.:3.4310
##  Max.   :1200.8   Max.   :122.626   Max.   :91.79    Max.   :9.4619
##   T0_RMSSD_ms        T0_VLF_ms2         T0_LF_ms2
##  Min.   :  4.853   Min.   :   16.31   Min.   :   3.159
```

```
##  1st Qu.: 17.550   1st Qu.:  220.68   1st Qu.:  94.505
##  Median : 26.012   Median :  457.08   Median : 237.181
##  Mean   : 34.630   Mean   :  831.88   Mean   : 453.914
##  3rd Qu.: 40.421   3rd Qu.:  990.71   3rd Qu.: 414.436
##  Max.   :174.752   Max.   :11123.04   Max.   :4851.623
##    T0_HF_ms2         T0_Total_power_ms2 T0_Frequence_Respiratoire
##  Min.   :    7.728   Min.   :   30.41   Min.   : 7.524
##  1st Qu.:   83.177   1st Qu.:  490.68   1st Qu.:11.772
##  Median :  200.949   Median : 1097.55   Median :13.855
##  Mean   :  697.922   Mean   : 1988.42   Mean   :14.124
##  3rd Qu.:  484.893   3rd Qu.: 2186.91   3rd Qu.:16.519
##  Max.   :10309.582   Max.   :18832.94   Max.   :20.753
```

```
##  T0_Mean_RR_ms     T0_STD_RR_ms     T0_Mean_HR_1_min T0_STD_HR_1_min
##  Min.   : 654.9   Min.   :  7.107   Min.   :50.01    Min.   :0.4276
##  1st Qu.: 827.8   1st Qu.: 25.387   1st Qu.:60.42    1st Qu.:1.9783
##  Median : 912.4   Median : 35.308   Median :66.85    Median :2.4959
##  Mean   : 914.9   Mean   : 38.643   Mean   :66.93    Mean   :3.0289
##  3rd Qu.: 997.4   3rd Qu.: 47.137   3rd Qu.:72.59    3rd Qu.:3.4342
##  Max.   :1200.8   Max.   :122.626   Max.   :91.79    Max.   :9.4619
##   T0_RMSSD_ms        T0_VLF_ms2         T0_LF_ms2
##  Min.   : -9.546   Min.   :   16.31   Min.   :   3.159
##  1st Qu.: 16.861   1st Qu.:  220.68   1st Qu.:  94.839
##  Median : 24.875   Median :  457.08   Median : 240.734
##  Mean   : 34.136   Mean   :  831.94   Mean   : 464.565
##  3rd Qu.: 38.238   3rd Qu.:  990.71   3rd Qu.: 438.148
##  Max.   :174.752   Max.   :11123.04   Max.   :4851.623
##    T0_HF_ms2         T0_Total_power_ms2 T0_Frequence_Respiratoire
##  Min.   :    7.728   Min.   :   30.41   Min.   : 7.524
##  1st Qu.:   83.177   1st Qu.:  490.68   1st Qu.:11.960
##  Median :  201.249   Median : 1147.67   Median :13.855
##  Mean   :  697.931   Mean   : 1998.84   Mean   :14.153
##  3rd Qu.:  484.893   3rd Qu.: 2186.91   3rd Qu.:16.381
##  Max.   :10309.582   Max.   :18832.94   Max.   :20.753
```

```
##  T0_Mean_RR_ms     T0_STD_RR_ms     T0_Mean_HR_1_min T0_STD_HR_1_min
##  Min.   : 654.9   Min.   :  7.107   Min.   :50.01    Min.   :0.4276
##  1st Qu.: 827.8   1st Qu.: 25.279   1st Qu.:60.42    1st Qu.:1.8964
##  Median : 908.1   Median : 35.295   Median :66.84    Median :2.4959
##  Mean   : 914.6   Mean   : 38.304   Mean   :66.93    Mean   :2.9802
##  3rd Qu.: 997.4   3rd Qu.: 45.589   3rd Qu.:72.59    3rd Qu.:3.4310
##  Max.   :1200.8   Max.   :122.626   Max.   :91.79    Max.   :9.4619
##   T0_RMSSD_ms        T0_VLF_ms2         T0_LF_ms2
##  Min.   :  4.853   Min.   :   16.31   Min.   :   3.159
##  1st Qu.: 17.550   1st Qu.:  220.68   1st Qu.:  94.505
##  Median : 26.012   Median :  457.08   Median : 237.181
##  Mean   : 34.630   Mean   :  831.88   Mean   : 453.914
##  3rd Qu.: 40.421   3rd Qu.:  990.71   3rd Qu.: 414.436
##  Max.   :174.752   Max.   :11123.04   Max.   :4851.623
##    T0_HF_ms2         T0_Total_power_ms2 T0_Frequence_Respiratoire
##  Min.   :    7.728   Min.   :   30.41   Min.   : 7.524
##  1st Qu.:   83.177   1st Qu.:  490.68   1st Qu.:11.772
##  Median :  200.949   Median : 1097.55   Median :13.855
##  Mean   :  697.922   Mean   : 1988.42   Mean   :14.124
##  3rd Qu.:  484.893   3rd Qu.: 2186.91   3rd Qu.:16.519
```

```
##  Max.   :10309.582   Max.   :18832.94   Max.   :20.753
```

```
##  T0_Mean_RR_ms     T0_STD_RR_ms     T0_Mean_HR_1_min T0_STD_HR_1_min
##  Min.   : 654.9   Min.   :  7.107   Min.   :50.01    Min.   :0.4276
##  1st Qu.: 827.8   1st Qu.: 25.524   1st Qu.:60.42    1st Qu.:1.8964
##  Median : 908.1   Median : 35.308   Median :66.57    Median :2.4959
##  Mean   : 914.5   Mean   : 38.420   Mean   :66.92    Mean   :2.9739
##  3rd Qu.: 997.4   3rd Qu.: 45.589   3rd Qu.:72.59    3rd Qu.:3.4251
##  Max.   :1200.8   Max.   :122.626   Max.   :91.79    Max.   :9.4619
##   T0_RMSSD_ms        T0_VLF_ms2         T0_LF_ms2
##  Min.   :  4.853   Min.   :   16.31   Min.   :   3.159
##  1st Qu.: 17.084   1st Qu.:  220.68   1st Qu.:  94.839
##  Median : 25.737   Median :  457.08   Median : 240.734
##  Mean   : 35.022   Mean   :  830.77   Mean   : 461.288
##  3rd Qu.: 42.498   3rd Qu.:  990.71   3rd Qu.: 438.148
##  Max.   :174.752   Max.   :11123.04   Max.   :4851.623
##    T0_HF_ms2         T0_Total_power_ms2 T0_Frequence_Respiratoire
##  Min.   :    7.728   Min.   :   30.41   Min.   : 7.524
##  1st Qu.:   83.177   1st Qu.:  490.68   1st Qu.:11.960
##  Median :  200.650   Median : 1141.51   Median :14.054
##  Mean   :  697.912   Mean   : 1993.06   Mean   :14.229
##  3rd Qu.:  484.893   3rd Qu.: 2186.91   3rd Qu.:16.519
##  Max.   :10309.582   Max.   :18832.94   Max.   :20.753
```

Nous allons donc imputer par le pmm!

Imputation à T1

Création des valeurs NA dans le jeu de données complet

Nous avons créer un jeu de données contenant des valeurs manquantes afin de tester plus tard avec les différentes méthodes d'imputations laquelle minimise les erreurs de prédictions

Les 5 différentes méthodes testées: (pmm, rf, sample, mean, norm)

```
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   1   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   1   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   1   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   1   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   2   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   2   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   2   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   2   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   3   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   3   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   3   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   3   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   4   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   4   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
##   4   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1_
```

```
##   4   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
```

```
##   1   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
```

```
##   5   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   1   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   2  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   3  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   4  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   5  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##
##  iter imp variable
##   1   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   2   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   3   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   4   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
##   5   1  T1_Mean_RR_ms  T1_STD_RR_ms  T1_Mean_HR_1_min  T1_STD_HR_1_min  T1_RMSSD_ms  T1_VLF_ms2  T1
## Warning: Number of logged events: 19
```

Comparaison des différentes méthodes d'imputations

```
##         R_pmm  R_mean    R_rf R_sample R_norm
##  [1,]    5.61   34.12   19.05    27.90   7.68   5.61
##  [2,]    0.78    3.18    0.52     3.28   0.16   0.16
##  [3,]    0.59    3.23    2.65     6.28   1.44   0.59
```

```
##  [4,]    0.10    0.31    0.11     0.11   0.13   0.10
##  [5,]    1.52    5.24    1.50     4.26   1.72   1.50
##  [6,]   52.67  112.61   46.92    92.82 150.72  46.92
##  [7,]  351.39  415.07  224.40   390.93  50.51  50.51
##  [8,]  998.34 1043.01 1077.23  1054.89 825.08 825.08
##  [9,] 1118.40 1183.66  396.02  1181.35 839.87 396.02
## [10,]    0.64    0.43    0.40     0.85   0.49   0.40
```

En faisant une analyse des cinq méthodologies d'imputations nous observons que les deux premières méthodes à savoir la regression linéaire bayesienne(norm) et le pmm minimise au mieux les variables. Mais pour en choisir qu'une à la fin nous analysons aussi les statistiques descriptives que fournissent ces données avant et après imputation et surtout pour la variable RMSSD en phase T1 qui servira plus tard pour la prédiction.

Comparaison des statistiques descriptives des méthodes

```
##  T1_Mean_RR_ms     T1_STD_RR_ms     T1_Mean_HR_1_min T1_STD_HR_1_min
##  Min.   : 598.9   Min.   :  7.107   Min.   : 48.47   Min.   : 0.4276
##  1st Qu.: 827.8   1st Qu.: 26.849   1st Qu.: 58.97   1st Qu.: 1.8960
##  Median : 952.0   Median : 35.540   Median : 63.10   Median : 2.2972
##  Mean   : 937.1   Mean   : 41.142   Mean   : 65.84   Mean   : 3.2437
##  3rd Qu.:1019.0   3rd Qu.: 50.265   3rd Qu.: 73.08   3rd Qu.: 3.2837
##  Max.   :1238.2   Max.   :120.334   Max.   :100.24   Max.   :12.6799
##   T1_RMSSD_ms        T1_VLF_ms2        T1_LF_ms2          T1_HF_ms2
##  Min.   :  4.618   Min.   :  31.73   Min.   :   3.561   Min.   :   4.629
##  1st Qu.: 14.343   1st Qu.: 260.51   1st Qu.: 108.754   1st Qu.:  66.013
##  Median : 25.749   Median : 502.61   Median : 180.623   Median : 162.398
##  Mean   : 36.865   Mean   : 692.77   Mean   : 378.233   Mean   : 633.798
##  3rd Qu.: 36.300   3rd Qu.: 778.72   3rd Qu.: 453.271   3rd Qu.: 507.147
##  Max.   :178.357   Max.   :5395.04   Max.   :3282.235   Max.   :7930.139
##  T1_Total_power_ms2 T1_Frequence_Respiratoire
##  Min.   :  43.03    Min.   : 6.694
##  1st Qu.: 448.07    1st Qu.:11.289
##  Median :1038.30    Median :14.107
##  Mean   :1708.81    Mean   :13.835
##  3rd Qu.:1843.76    3rd Qu.:16.792
##  Max.   :9817.45    Max.   :19.062
```

```
##  T1_Mean_RR_ms     T1_STD_RR_ms     T1_Mean_HR_1_min T1_STD_HR_1_min
##  Min.   : 598.9   Min.   :  7.107   Min.   : 48.47   Min.   : 0.4276
##  1st Qu.: 827.8   1st Qu.: 26.849   1st Qu.: 58.59   1st Qu.: 1.8960
##  Median : 952.0   Median : 35.540   Median : 63.10   Median : 2.2972
##  Mean   : 937.2   Mean   : 41.164   Mean   : 65.62   Mean   : 3.2428
##  3rd Qu.:1019.0   3rd Qu.: 50.265   3rd Qu.: 73.08   3rd Qu.: 3.2837
##  Max.   :1238.2   Max.   :120.334   Max.   :100.24   Max.   :12.6799
##   T1_RMSSD_ms        T1_VLF_ms2       T1_LF_ms2          T1_HF_ms2
##  Min.   :  4.853   Min.   :-120.8   Min.   :   3.561   Min.   :   4.629
##  1st Qu.: 14.142   1st Qu.: 260.5   1st Qu.: 108.754   1st Qu.:  66.013
##  Median : 25.749   Median : 523.7   Median : 194.623   Median : 162.398
##  Mean   : 36.751   Mean   : 705.9   Mean   : 385.872   Mean   : 518.435
##  3rd Qu.: 36.300   3rd Qu.: 809.1   3rd Qu.: 457.962   3rd Qu.: 507.147
##  Max.   :178.357   Max.   :5395.0   Max.   :3312.369   Max.   :5442.191
##  T1_Total_power_ms2 T1_Frequence_Respiratoire
##  Min.   :  43.03    Min.   : 6.694
##  1st Qu.: 448.07    1st Qu.:11.408
##  Median :1093.46    Median :14.265
```

```
##  Mean   :1612.74    Mean   :13.903
##  3rd Qu.:1844.84    3rd Qu.:16.792
##  Max.   :9817.45    Max.   :19.062
```

```
##  T1_Mean_RR_ms     T1_STD_RR_ms     T1_Mean_HR_1_min T1_STD_HR_1_min
##  Min.   : 598.9   Min.   :  7.107   Min.   : 48.47   Min.   : 0.4276
##  1st Qu.: 827.8   1st Qu.: 26.849   1st Qu.: 58.97   1st Qu.: 1.8960
##  Median : 952.0   Median : 35.540   Median : 63.10   Median : 2.2972
##  Mean   : 937.1   Mean   : 41.142   Mean   : 65.84   Mean   : 3.2437
##  3rd Qu.:1019.0   3rd Qu.: 50.265   3rd Qu.: 73.08   3rd Qu.: 3.2837
##  Max.   :1238.2   Max.   :120.334   Max.   :100.24   Max.   :12.6799
##   T1_RMSSD_ms        T1_VLF_ms2        T1_LF_ms2          T1_HF_ms2
##  Min.   :  4.618   Min.   :  31.73   Min.   :   3.561   Min.   :   4.629
##  1st Qu.: 14.343   1st Qu.: 260.51   1st Qu.: 108.754   1st Qu.:  66.013
##  Median : 25.749   Median : 502.61   Median : 180.623   Median : 162.398
##  Mean   : 36.865   Mean   : 692.77   Mean   : 378.233   Mean   : 633.798
##  3rd Qu.: 36.300   3rd Qu.: 778.72   3rd Qu.: 453.271   3rd Qu.: 507.147
##  Max.   :178.357   Max.   :5395.04   Max.   :3282.235   Max.   :7930.139
##  T1_Total_power_ms2 T1_Frequence_Respiratoire
##  Min.   :  43.03    Min.   : 6.694
##  1st Qu.: 448.07    1st Qu.:11.289
##  Median :1038.30    Median :14.107
##  Mean   :1708.81    Mean   :13.835
##  3rd Qu.:1843.76    3rd Qu.:16.792
##  Max.   :9817.45    Max.   :19.062
```

```
##  T1_Mean_RR_ms     T1_STD_RR_ms     T1_Mean_HR_1_min T1_STD_HR_1_min
##  Min.   : 598.9   Min.   :  7.107   Min.   : 48.47   Min.   : 0.4276
##  1st Qu.: 827.8   1st Qu.: 26.849   1st Qu.: 58.97   1st Qu.: 1.9889
##  Median : 952.0   Median : 35.540   Median : 63.10   Median : 2.2972
##  Mean   : 937.9   Mean   : 41.033   Mean   : 65.78   Mean   : 3.2556
##  3rd Qu.:1019.0   3rd Qu.: 50.265   3rd Qu.: 73.08   3rd Qu.: 3.2837
##  Max.   :1238.2   Max.   :120.334   Max.   :100.24   Max.   :12.6799
##   T1_RMSSD_ms        T1_VLF_ms2        T1_LF_ms2          T1_HF_ms2
##  Min.   :  4.853   Min.   :  31.73   Min.   :   3.561   Min.   :   4.629
##  1st Qu.: 14.142   1st Qu.: 260.51   1st Qu.: 108.754   1st Qu.:  66.013
##  Median : 25.749   Median : 502.61   Median : 194.623   Median : 162.398
##  Mean   : 36.886   Mean   : 687.32   Mean   : 330.151   Mean   : 494.501
##  3rd Qu.: 36.300   3rd Qu.: 778.72   3rd Qu.: 453.271   3rd Qu.: 507.147
##  Max.   :178.357   Max.   :5395.04   Max.   :1682.161   Max.   :5442.191
##  T1_Total_power_ms2 T1_Frequence_Respiratoire
##  Min.   :  43.03    Min.   : 6.694
##  1st Qu.: 448.07    1st Qu.:11.408
##  Median :1093.46    Median :14.265
##  Mean   :1540.96    Mean   :13.925
##  3rd Qu.:1767.00    3rd Qu.:16.792
##  Max.   :9817.45    Max.   :19.062
```

Notre jeu de données sera imputée par la méthode de la moyenne prévisionnelle.

Fichier final

```
imput_T0<-cbind(data_physio[vec_na_T0==0,"Numéro HRV"],apply(data_physio[vec_na_T0==0,num_var_T0],2, FU
  need_replace<-!(cherche.ext(x))
  xbis<-x
```

```
  xbis[need_replace]<-NA
  return(xbis)})
)

imput_T1<-cbind(data_physio[vec_na_T1==0,"Numéro HRV"],apply(data_physio[vec_na_T1==0,num_var_T1],2, FU
  need_replace<-!(cherche.ext(x))
  xbis<-x
  xbis[need_replace]<-NA
  return(xbis)})
)

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

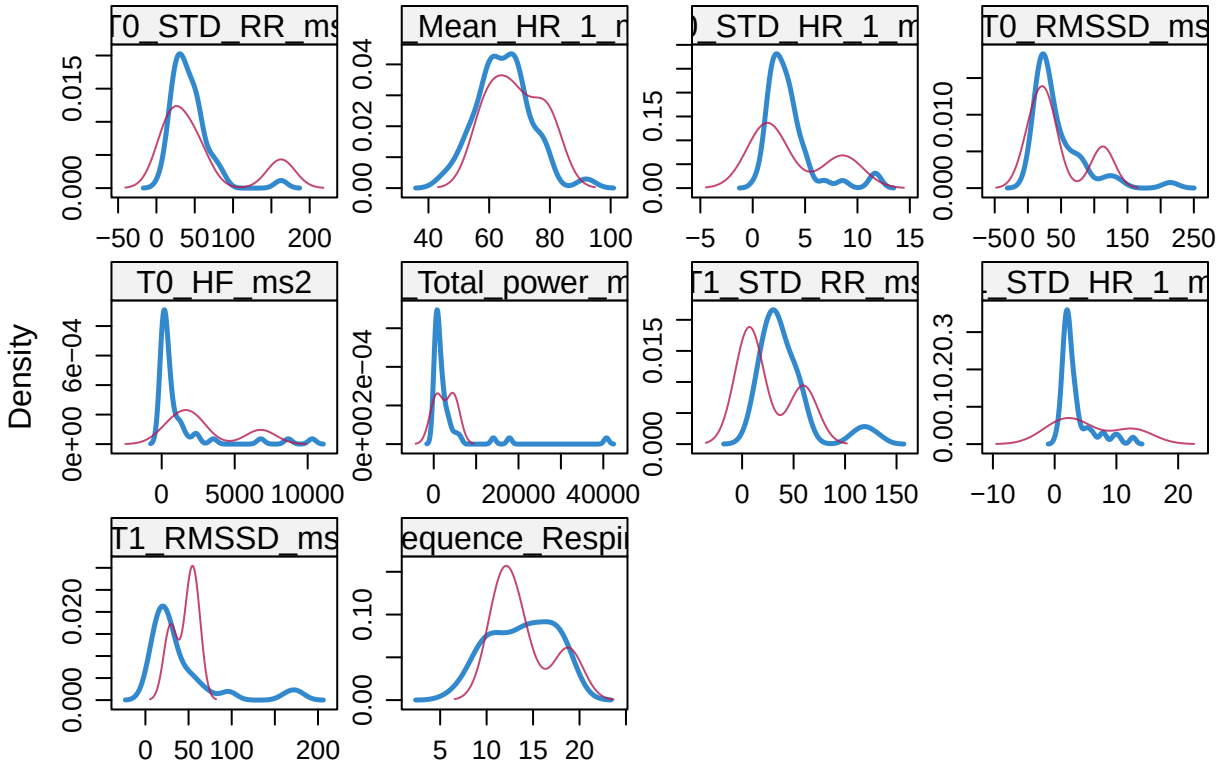
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
pp=imput_T0 %>% inner_join(imput_T1, by = "Numéro HRV")
ppp=pp[,-1]

#View(pp)

imp_data <- mice(ppp, m=1, maxit = 5, method = 'pmm', seed = 500)
```

```
##
##  iter imp variable
##   1   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   2   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   3   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   4   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
##   5   1  T0_Mean_RR_ms  T0_STD_RR_ms  T0_Mean_HR_1_min  T0_STD_HR_1_min  T0_RMSSD_ms  T0_VLF_ms2  T0
```

```
## Warning: Number of logged events: 63
```

```
densityplot(imp_data)
```




```
final <- complete(imp_data)
NumeroHRV=pp[,1]
final=cbind(NumeroHRV,final)
#View(final)
```

Comparaison avant et après imputation

```
## [1] "RMSSD Avant Imputation"
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   4.853  20.021  28.035  44.796  57.952 214.604       4
```

```
## [1] "########################################################################"
```

```
## [1] "########################################################################"
```

```
## [1] "RMSSD Après Imputation"
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.618  17.094  28.166  40.341  51.035 178.357
```

En observant par exemple pour la variable RMSSD avant imputation et après imputation, nous obtenons une moyenne de 41,1 contre 39,9 précédemment, ce qui est acceptable par rapport à la methode utilisée et la troisième quartile qui varie de 42 à 43,82 , la médiane qui était de 26,5 passe à 27,4. Notre imputation n'a pas eu d'impact énorme sur notre jeu de données.

```
##
## yes  no
## 0.5 0.5
```

Nous avons 50% de progrès.

Boostrap: comparaison de moyenne

A COMMENTER!!!!

```
#H0:T0>T1

x=T0$T0_RMSSD_ms
y=T1$T1_RMSSD_ms
n=length(x)
m=length(y)

wilcox=c()
M0=c()
M1=c()
B=200
wilcox=numeric(B)
M0=numeric(B)
M1=numeric(B)

for(b in 1:B){
ind1 = sample(c(1:n),n, replace = T)
ind2 = sample(c(1:m),m, replace = T)

x1=x[ind1]
y1=y[ind2]
wilcox[b] = wilcox.test(x1,y1,alternative = "greater")$p.value
M0[b]=mean(x1)
M1[b]=mean(y1)
}
```

```
## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
```

```
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(x1, y1, alternative = "greater"): cannot
## compute exact p-value with ties
```

```r
help("wilcox.test")
```

```
## starting httpd help server ... done
```

```r
print("pval")
```

```
## [1] "pval"
```

```r
wilcox
```

```
##   [1] 3.001157e-01 2.975611e-01 8.802816e-01 1.865792e-01 7.464413e-01
##   [6] 1.602511e-01 6.599666e-02 4.956140e-01 1.431374e-01 7.124998e-01
##  [11] 2.988512e-01 4.489724e-01 2.883250e-02 5.301220e-02 1.925488e-01
##  [16] 9.915853e-02 9.318337e-03 4.547754e-01 8.024046e-01 3.973334e-01
##  [21] 3.318429e-02 3.461349e-01 2.132442e-01 1.666504e-01 1.532674e-01
##  [26] 4.956143e-01 2.306618e-01 7.878285e-01 4.562305e-01 8.523148e-02
##  [31] 1.915470e-01 3.447796e-01 3.426898e-02 3.393898e-01 2.262311e-01
##  [36] 3.832941e-01 7.625926e-01 7.846300e-01 5.262858e-02 1.226832e-01
```

```
##  [41] 2.606355e-01 2.111262e-01 4.664097e-01 2.079548e-01 1.532642e-01
##  [46] 3.735374e-01 2.111089e-01 8.064615e-01 3.132908e-02 3.353796e-01
##  [51] 2.431206e-01 2.048100e-01 3.611157e-01 1.532814e-01 4.460721e-01
##  [56] 4.888218e-03 4.215618e-01 4.868444e-01 1.111625e-01 3.597989e-02
##  [61] 4.087050e-01 9.598944e-02 3.059522e-03 2.976911e-02 1.532715e-01
##  [66] 1.489301e-05 5.813060e-01 4.431766e-01 3.434332e-01 7.075285e-01
##  [71] 3.583799e-01 5.626058e-01 1.439457e-01 2.824911e-01 6.675968e-03
##  [76] 9.725744e-02 1.447793e-01 7.485128e-02 6.555717e-02 1.057069e-01
##  [81] 2.442745e-01 1.063734e-01 1.754636e-02 1.481125e-01 9.477551e-01
##  [86] 5.597151e-01 2.306793e-01 7.512123e-01 4.693242e-01 2.988413e-01
##  [91] 1.567838e-01 2.348993e-02 8.931504e-02 4.373954e-01 7.236359e-01
##  [96] 1.023978e-01 8.353063e-03 4.664117e-01 5.382443e-02 1.390194e-01
## [101] 2.397069e-01 6.167035e-01 1.017672e-01 6.805174e-05 3.103975e-01
## [106] 1.475351e-02 3.052257e-01 3.529121e-01 7.637139e-01 3.917195e-01
## [111] 1.506930e-01 7.082716e-02 6.144399e-02 5.796161e-02 2.612755e-03
## [116] 7.260869e-01 1.132555e-01 8.688157e-03 5.219194e-01 3.599002e-02
## [121] 1.788252e-01 4.209772e-03 1.749574e-01 7.759502e-01 1.125558e-01
## [126] 8.130254e-02 6.592683e-01 9.232384e-02 9.538179e-02 3.958562e-02
## [131] 8.351411e-02 8.815642e-02 7.993283e-01 3.917199e-01 8.990530e-02
## [136] 4.504199e-01 1.326201e-01 4.941515e-01 1.488643e-02 1.017503e-01
## [141] 6.838032e-02 5.070010e-02 9.168541e-02 6.600585e-02 8.268668e-01
## [146] 6.659667e-01 1.464554e-01 7.260848e-01 7.393362e-01 1.846333e-01
## [151] 1.017211e-01 2.396659e-01 2.849916e-01 5.192323e-04 7.648495e-01
## [156] 3.026750e-01 3.474843e-01 7.184125e-02 1.084163e-01 3.052330e-01
## [161] 9.663478e-02 3.484450e-02 4.663201e-02 4.795413e-01 2.523890e-01
## [166] 1.256844e-01 6.600978e-02 8.610222e-01 4.853829e-01 1.647133e-02
## [171] 7.693352e-01 4.985379e-01 2.630498e-01 1.146604e-01 1.189244e-01
## [176] 1.439392e-01 3.055502e-02 2.121644e-01 1.576644e-01 2.351573e-01
## [181] 3.103815e-01 2.512243e-01 3.638665e-01 5.073096e-01 3.958867e-02
## [186] 2.925180e-01 3.313889e-01 5.219200e-01 8.560502e-01 3.370488e-02
## [191] 3.902940e-01 2.937693e-01 3.860824e-01 4.660924e-02 3.502001e-01
## [196] 4.028889e-03 1.616675e-02 4.287450e-01 4.698722e-02 5.927124e-01
```

```
print("M0")
```

```
## [1] "M0"
```

```
M0
```

```
##   [1] 43.47096 43.55182 37.28752 46.42444 36.03258 37.25825 48.01957
##   [8] 40.43129 48.33585 39.92718 42.61603 45.02586 44.96772 44.11890
##  [15] 42.60401 39.40151 50.29685 49.19212 43.96757 38.88150 52.93303
##  [22] 40.73224 46.58274 46.71164 45.97009 36.57553 42.11233 41.37603
##  [29] 43.86254 49.52600 37.67750 44.75268 45.55448 45.68696 45.07650
##  [36] 42.69370 38.51449 37.43185 49.40477 46.22290 52.71242 46.55494
##  [43] 34.99584 40.11135 43.02577 40.13524 40.50745 36.75712 49.91326
##  [50] 43.66160 49.54435 44.73703 44.20346 42.56098 33.83660 64.55431
##  [57] 50.56152 45.00504 51.71206 56.82199 37.86756 46.88780 56.58269
##  [64] 48.67614 46.94046 56.77822 38.28332 40.63704 40.16817 35.30496
##  [71] 49.14431 39.47176 46.29652 41.67832 44.51238 45.77760 44.08744
##  [78] 43.90635 47.58536 37.85181 45.87912 51.15450 53.85437 40.46916
##  [85] 34.03171 37.61593 52.98074 33.53470 43.70567 40.84895 56.28755
##  [92] 39.29953 40.23114 40.56717 43.04040 44.74160 44.62758 40.90978
##  [99] 49.35065 42.60950 44.32775 45.41287 43.98674 59.11988 42.38833
## [106] 37.81481 42.65674 46.16877 35.09577 36.43210 45.26517 41.48605
## [113] 44.87374 46.38056 50.23667 48.41992 38.30139 57.76705 40.47777
```

```
## [120] 47.96385 47.71421 57.86173 53.61747 33.79910 40.20861 33.87919
## [127] 39.93390 49.15401 42.26006 60.00961 48.71229 47.91082 37.63611
## [134] 44.43077 42.20811 51.22438 46.65704 42.30773 51.96316 33.22692
## [141] 53.45823 36.68835 52.49420 51.50422 41.44901 38.10910 42.27166
## [148] 42.29502 40.22590 45.54193 49.88481 47.56254 41.36096 53.16698
## [155] 35.70680 42.80022 44.22551 41.56190 46.95084 46.68697 37.72305
## [162] 47.11396 47.50284 36.05678 43.01828 50.11246 52.24064 46.95484
## [169] 31.71503 45.07591 48.28494 44.87291 35.08097 42.76314 48.20372
## [176] 45.67613 39.53146 39.07026 50.63966 49.94360 40.35524 47.15003
## [183] 47.24827 38.88729 42.71227 53.97160 39.91090 37.20994 34.65195
## [190] 43.70710 40.81621 46.76895 39.38326 45.51851 47.71793 49.47860
## [197] 37.31281 52.73781 61.31084 41.60398
```

```
print("M1")
```

```
## [1] "M1"
```

```
M1
```

```
##   [1] 40.81484 36.25478 55.39881 44.79900 51.31629 41.68994 32.80522
##   [8] 43.20200 39.61241 40.22988 40.13817 43.88962 33.22993 37.58936
##  [15] 31.66189 36.42252 38.16441 42.54488 41.80729 33.92009 40.01348
##  [22] 38.13050 37.62018 44.60956 37.50367 47.88309 42.41467 52.41163
##  [29] 48.59422 41.89232 31.69619 40.21649 41.02040 44.02749 38.36264
##  [36] 37.26999 37.03440 46.34660 35.06199 49.84504 38.45242 34.83058
##  [43] 41.43703 38.24625 41.98734 44.50215 35.83169 41.74491 39.93306
##  [50] 40.16699 41.53778 40.12485 46.54957 35.70949 43.49851 40.20336
##  [57] 50.35361 45.19665 45.32731 42.36919 33.20484 36.80938 37.29819
##  [64] 39.71170 42.34947 34.97462 36.37804 34.77209 34.28530 41.41767
##  [71] 37.22296 46.01655 42.28039 42.18680 31.30413 42.96441 44.36211
##  [78] 36.85562 39.57174 34.81022 35.00618 47.56056 39.94108 41.22738
##  [85] 45.10492 41.36664 42.47103 34.57541 42.09853 38.12021 47.90722
##  [92] 30.08022 35.82403 42.74660 47.98699 37.06418 31.91383 43.99960
##  [99] 43.85246 33.43551 37.36785 42.58169 44.16849 35.92771 43.29861
## [106] 29.79818 39.74670 38.63025 47.03261 33.01305 40.69359 30.76726
## [113] 39.68863 39.80560 31.86408 39.32563 40.93420 33.87725 36.81360
## [120] 39.18374 36.86855 37.51405 37.61991 43.47227 34.74784 35.65326
## [127] 46.98430 44.74202 41.26018 41.24527 35.30226 34.71577 45.27556
## [134] 38.01931 41.52423 47.74099 42.72731 44.18484 35.93503 34.99164
## [141] 44.25567 35.92493 37.99937 38.07326 51.54089 40.91058 33.32198
## [148] 44.02259 43.20044 37.49397 36.99968 47.78982 39.08784 33.83821
## [155] 46.43977 40.15444 36.93290 39.25994 36.80779 37.17450 31.48378
## [162] 33.84356 40.24312 39.30537 36.10396 41.21788 53.77715 56.58615
## [169] 34.43190 34.00350 47.10090 44.54186 43.51908 40.90198 38.92643
## [176] 34.92324 33.10231 45.94202 36.97263 39.48847 32.92860 37.92151
## [183] 49.09684 40.32371 31.66363 46.90795 38.76371 38.68085 41.75732
## [190] 29.26388 43.91802 47.07697 42.06627 30.12763 42.92694 29.26293
## [197] 31.67398 42.99429 41.17532 43.32040
```

```
pval_boot=mean(wilcox)
M0_boot=mean(M0)
M1_boot=mean(M1)

print("pval"); pval_boot; print("M0"); M0_boot; print("M1"); M1_boot;
```

```
## [1] "pval"
```

```
## [1] 0.2798285
## [1] "M0"
## [1] 44.40285
## [1] "M1"
## [1] 39.95523
```

Construction de modèles

Maintenant que nous avons imputé nos variables, nous pouvons chercher à créer un modèle pour prédire si une personne a vu sa condition physique s'améliorée. Pour cela, nous n'utilisons que les variables du temps T0.

Avec moins d'une cinquantaine d'individus dans nos résultats finaux, nous décidons de simplement chercher un modèle avec l'ensemble de nos données. Il faudra alors espérer avoir plus d'individus dans le futur afin de tester nos modèles.

Optimisation via Leave one Out

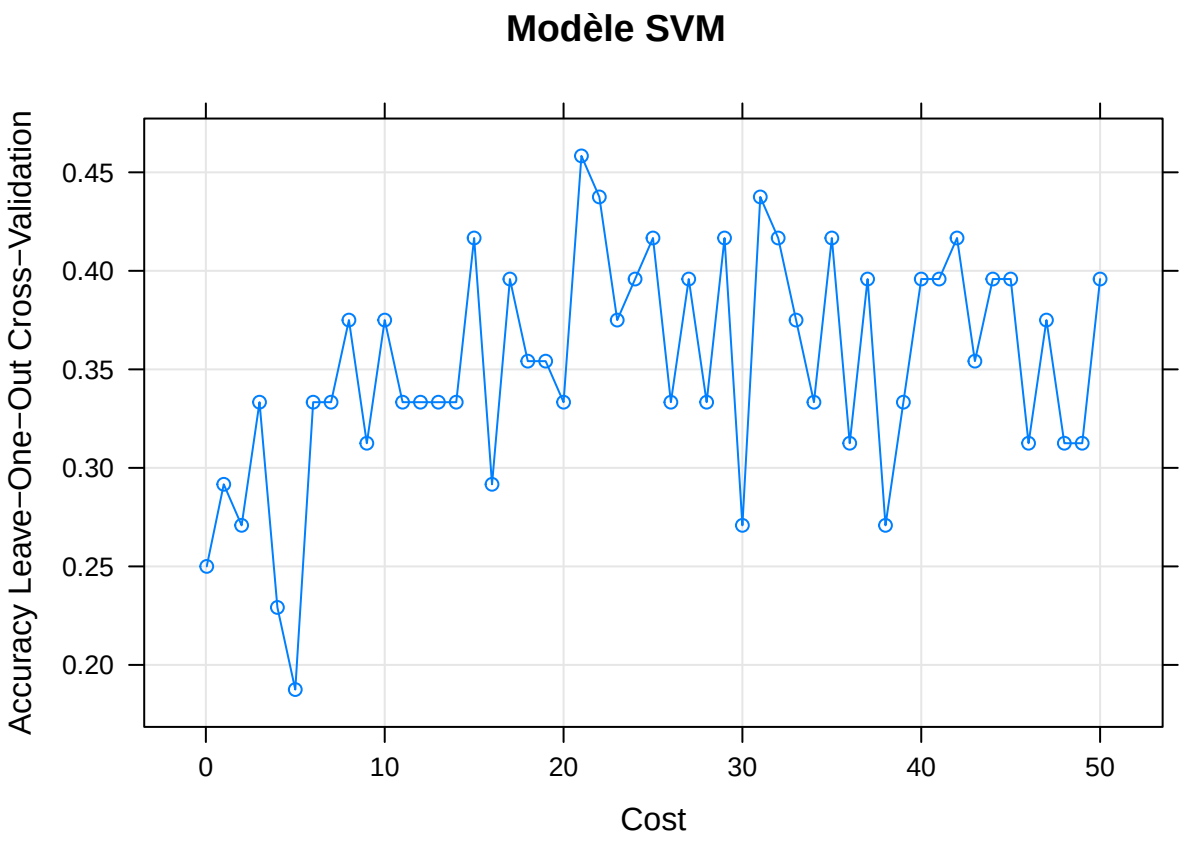
Nous allons chercher ici à optimiser nos modèles via la validation croisée. La méthode de validation croisée Leave one out permet de faire des tests avec peu d'individus.

Regression logistique

La première méthode que nous allons utiliser ici est la regression logistique descendante.

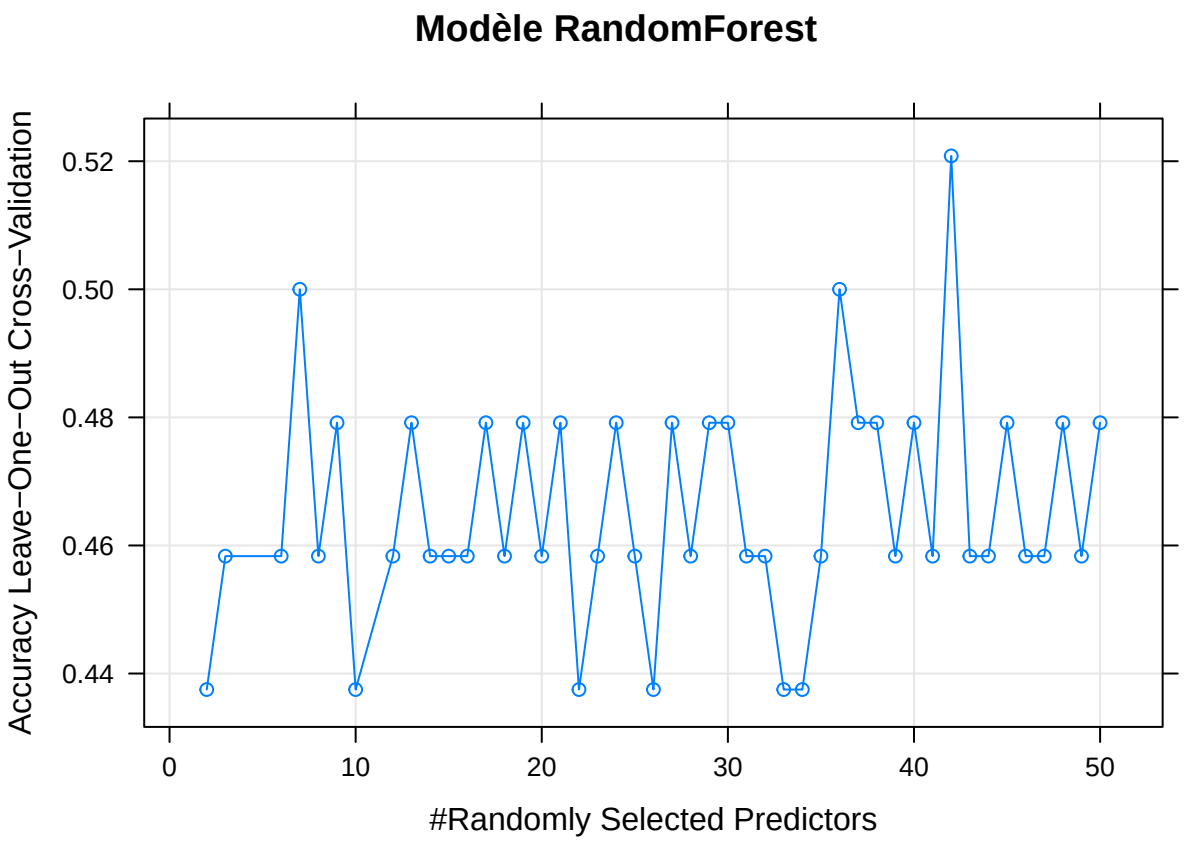
La précision de notre modèle est d'environ 0.5208333, ce qui est légèrement meilleur qu'un modèle renvoyant toujours la même prédiction avec nos données. Les variables conservée sont les variables RMSSD, VLF, HF, Mean HR, STD RR. Les variables conservées sont principalement celles que l'on considère souvent comme assez importante dans la litterature scientifique. Parmis ce type de variable, seul LF a été écartée.

SVM



Le SVM ici a dans l'ensemble une accuracy proche ou inférieur à 50%. Vu nos données, cela signifie qu'il est moins bon qu'un predicteur prédisant toujours la même valeur.

randomForest



Optimiser la méthode de forêt aléatoire avec la validation croisée "Leave One Out" n'est pas évidente. En effet, l'évaluation via un seul individu ne permet pas d'obtenir de résultat robuste dans ce cas là car le résultat dépend trop des variables choisies. De plus, l'accuracy, qu'importe le nombre de variable, semble se stabiliser vers 50% ou moins, ce qui est actuellement moins efficace qu'un predicteur prédisant toujours la même variable dans notre cas.

Optimisation via Bootstrap

Ici, nous allons utiliser une autre méthode, le bootstrap pour essayer de contourner notre problème de manque de donnée.

Regression logistique

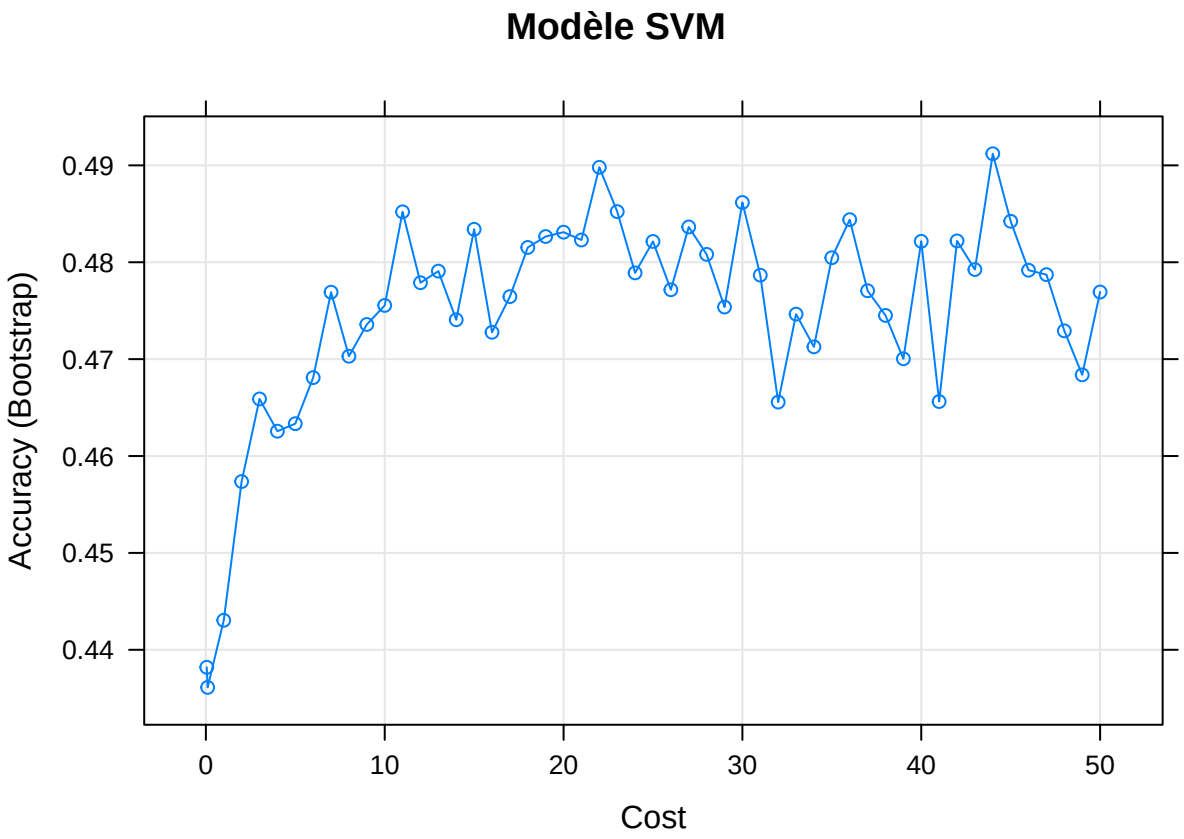
```
## Generalized Linear Model with Stepwise Feature Selection
##
## 48 samples
## 10 predictors
##  2 classes: 'yes', 'no'
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 48, 48, 48, 48, 48, 48, ...
## Resampling results:
##
##   Accuracy   Kappa
```

```
##   0.5218346  0.06559364
##
## Call:  NULL
##
## Coefficients:
##      (Intercept)      T0_STD_RR_ms  T0_Mean_HR_1_min   T0_STD_HR_1_min
##       -1.082e+01         1.549e-01         1.671e-01        -3.139e+00
##      T0_RMSSD_ms        T0_VLF_ms2         T0_HF_ms2
##        7.751e-02         7.322e-04        -8.272e-04
##
## Degrees of Freedom: 47 Total (i.e. Null);  41 Residual
## Null Deviance:       66.54
## Residual Deviance: 47.28     AIC: 61.28
```

La précision de notre modèle est d'environ 0.5218346, ce qui est légèrement meilleur qu'un modèle renvoyant toujours la même prédiction avec nos données. Les variables conservée sont les variables RMSSD, VLF, HF, Mean HR, STD RR. Les variables conservées sont principalement celles que l'on considère souvent comme assez importante dans la litterature scientifique. Parmis ce type de variable, seul LF a été écartée.

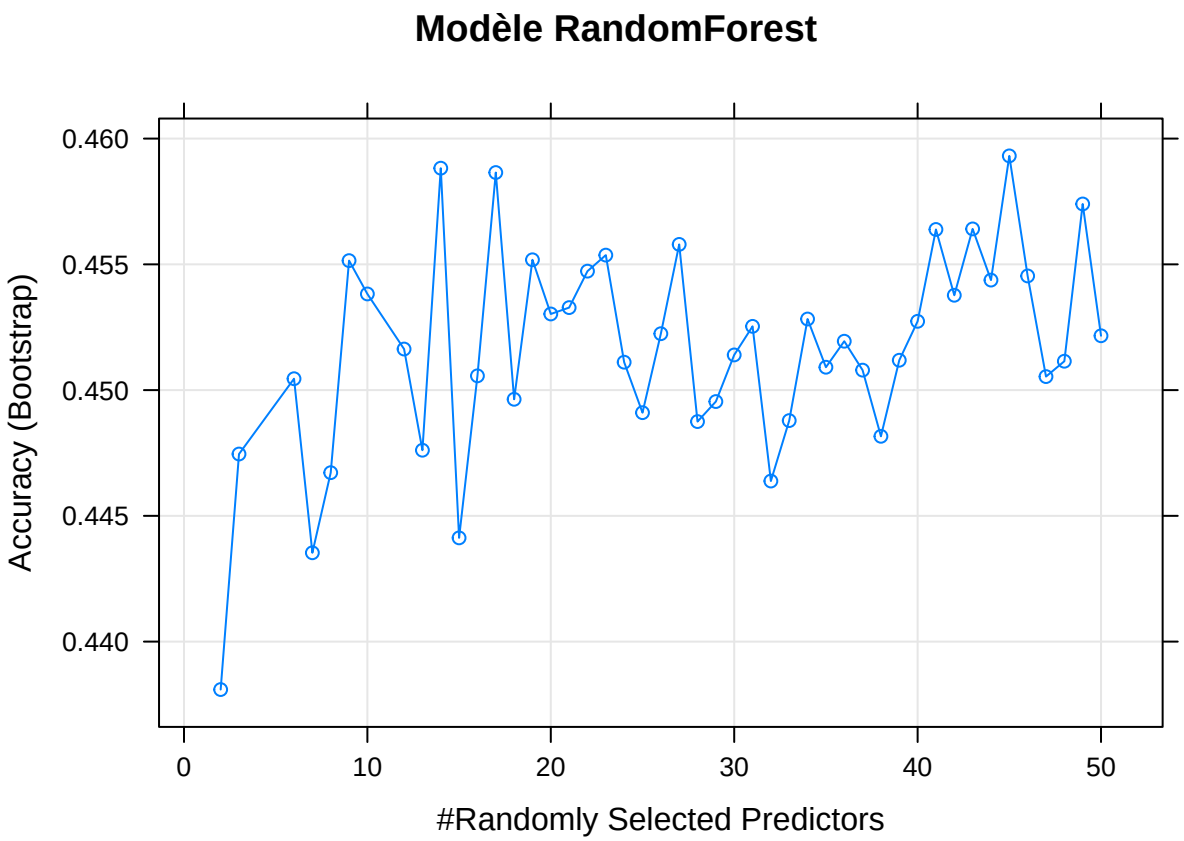
Au final, on retrouve pratiquement le même résultat qu'avec le leave one out.

SVM



Le SVM ici a dans l'ensemble une accuracy proche ou inférieur à 50%. Vu nos données, cela signifie qu'il est moins bon qu'un predicteur prédisant toujours la même valeur.

randomForest

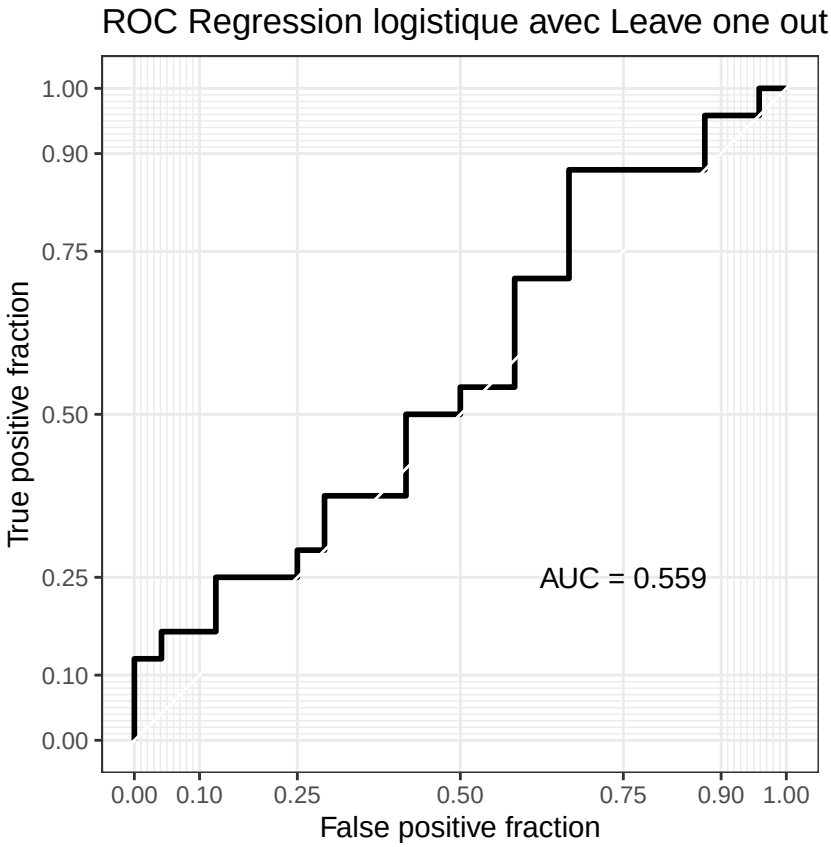


L'accuracy, qu'importe le nombre de variable, semble se stabiliser vers 50% ou moins, ce qui est actuellement moins efficace qu'un predicteur prédisant toujours la même variable dans notre cas.

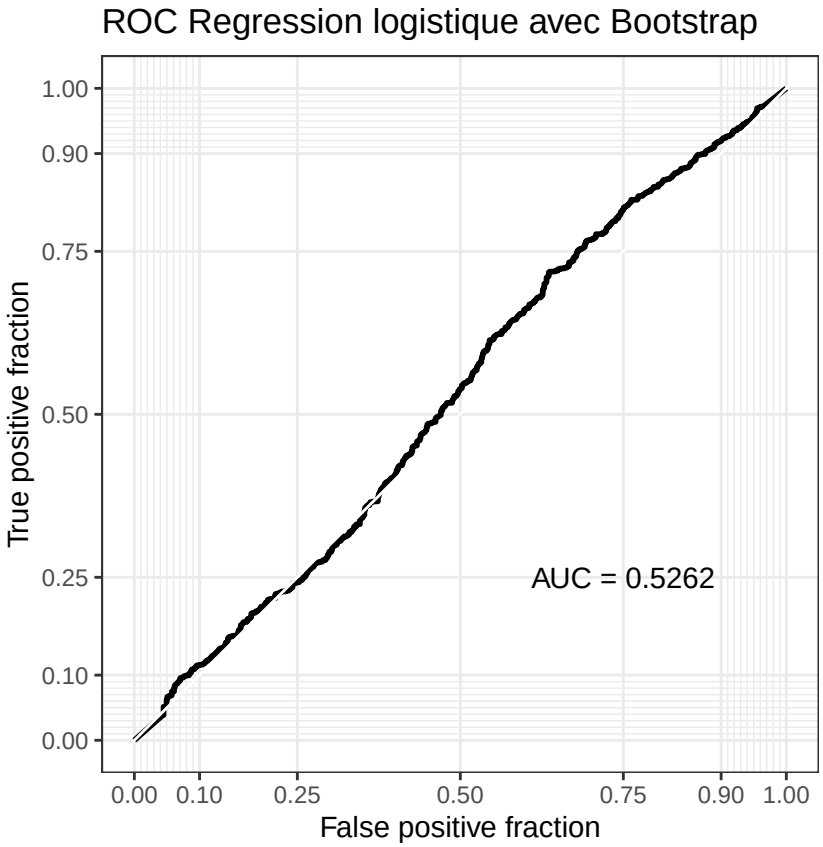
Comparaisons

Nous avons vu que quelques soit les méthodes d'évaluation et de prédiction que nous utilisons, dans l'ensemble, nous obtenons souvent des résultats semblables, avec une précision proche de 50%. Nous allons maintenant terminer en comparant nos méthodes selon d'autres critères afin d'arrêter notre choix.

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!

## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```



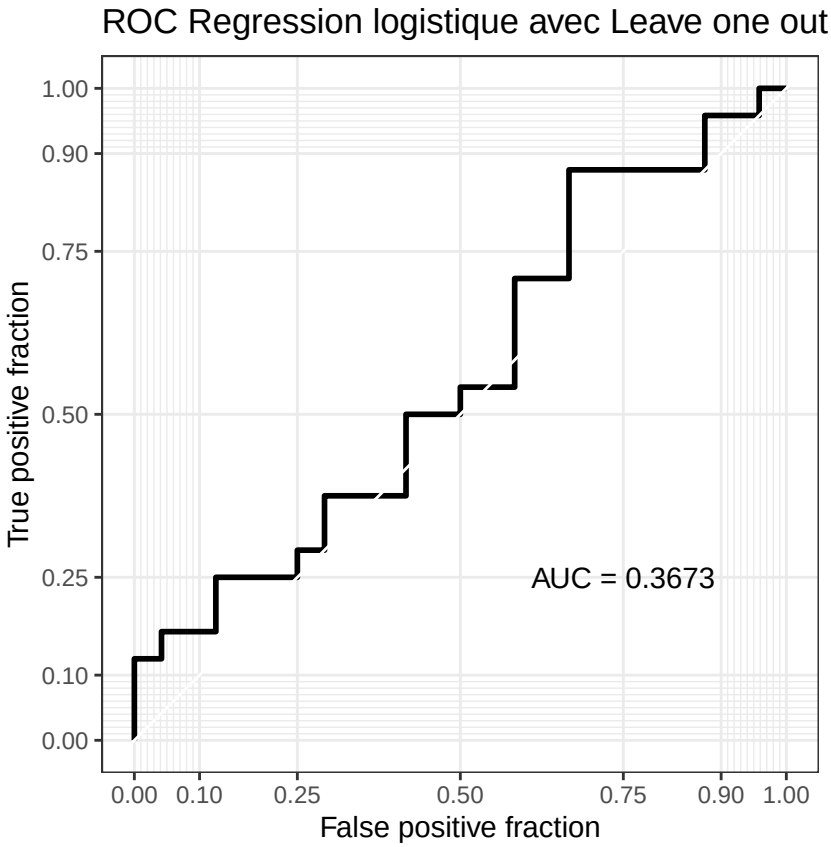

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!

## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```



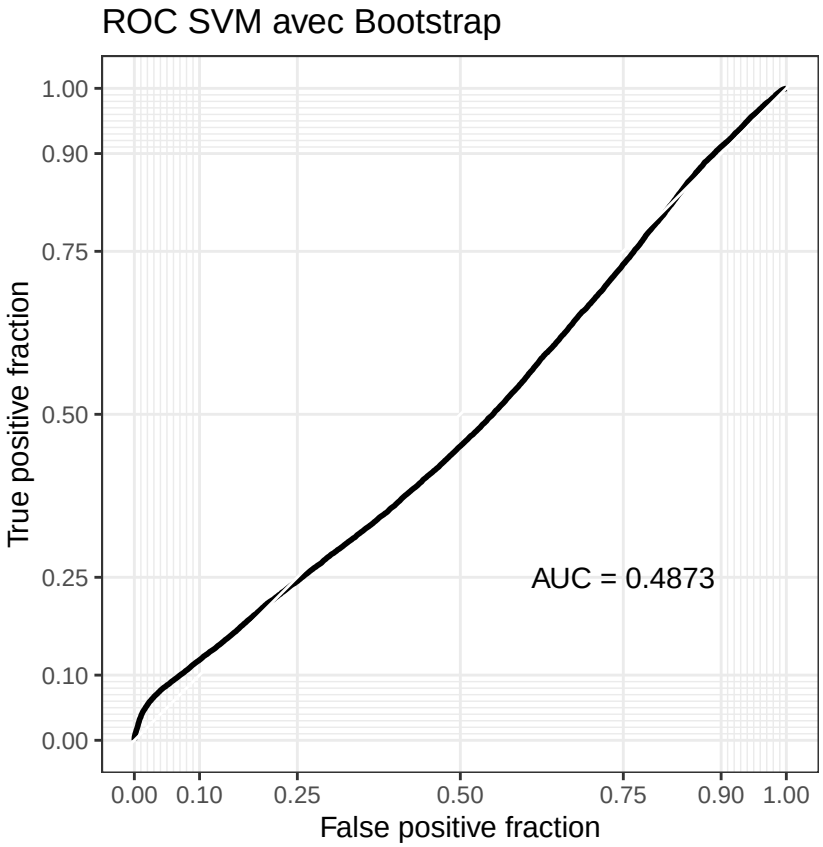
Dans l'ensemble, on peut supposer que l'AUC de la regression logistique est d'environ 0.55 .

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```

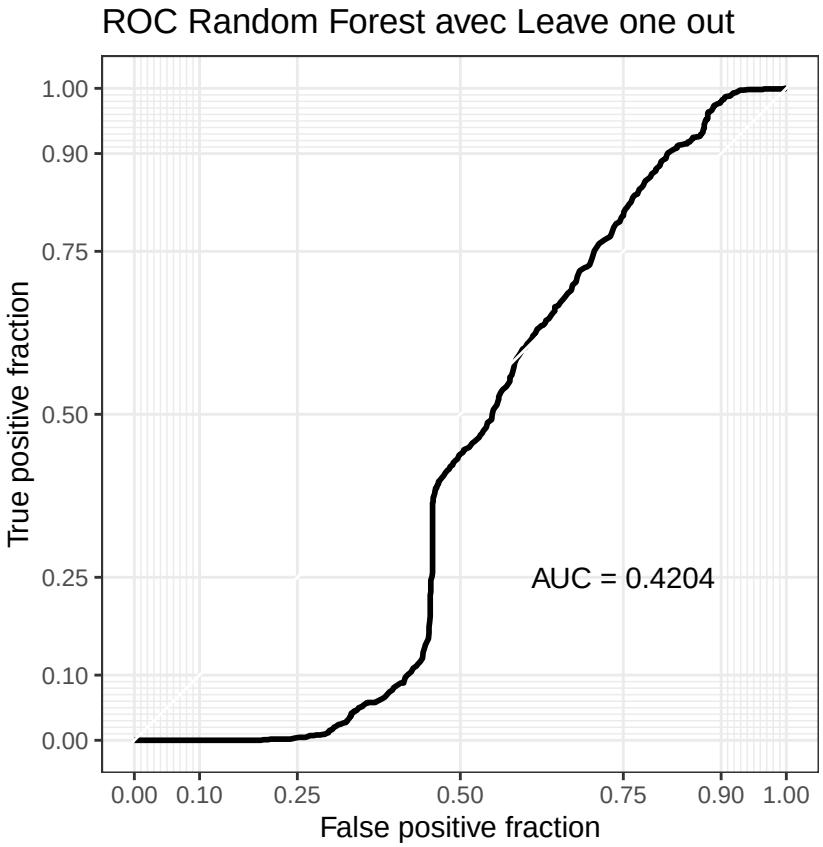



```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!

## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```

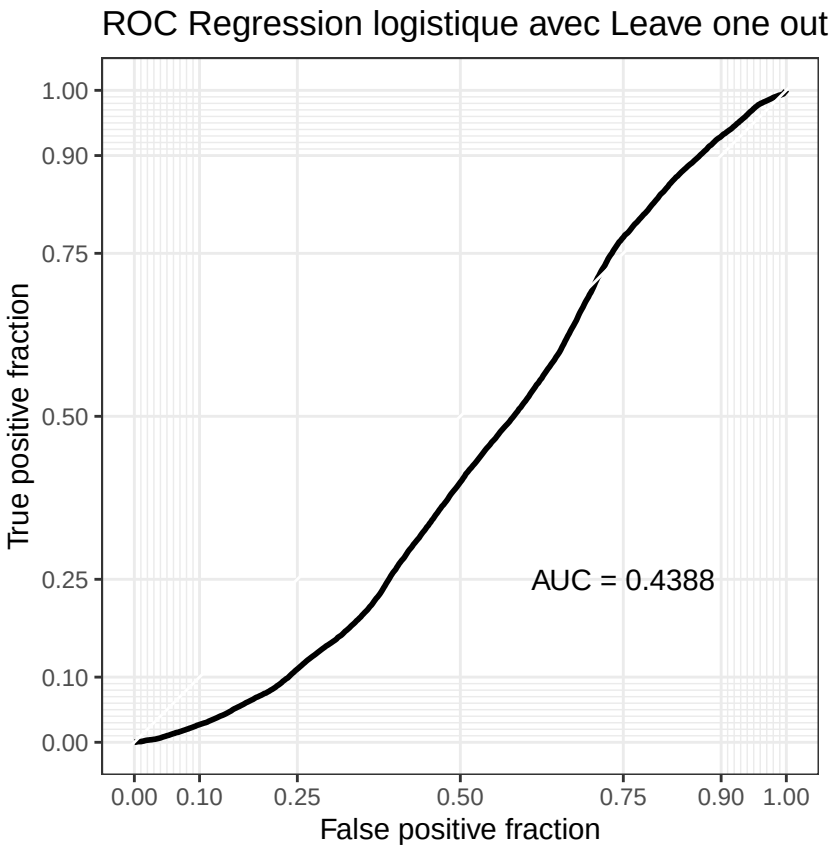


Dans l'ensemble, on peut supposer que l'AUC du SVM est d'environ 0.48 . C'est moins qu'avec la regression logistique.

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!

## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```




```
## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!

## Warning in verify_d(data$d): D not labeled 0/1, assuming no = 0 and yes =
## 1!
```



Dans l'ensemble, on peut supposer que l'AUC du random Forest est d'environ 0.43, c'est encore inférieur au SVM et à la regression logistique.

Conclusion

BLABLA

Concernant les modèles de prédictions, nous pouvons voir qu'avec nos données, il semble difficile d'établir un modèle convenable. Même si nous utilisons différentes méthodes pour la recherche de predicteur efficace, nous avons au final une précision faible dans tout les cas. Il semble alors pour le moment difficile de prédire l'amélioration de l'état de santé d'un individus via de programme en se basant seulement sur des informations physilogique avant expérience.