RenDate

Ph Dufresne

07/02/2020

Contents

Pour faire un pdf avec Knit

install.packages('tinytex') tinytex::install_tinytex()

Chargement des fonctions de RenDate.R

##via GitHub

```
 if (!require(devtools))
 {install.packages("devtools")}

devtools::install_github("chrono35/RenDate", force = TRUE)
```

```
##      checking for file '/private/var/folders/ms/3r6m3pqn4jq1hk94t646qdd00000gn/T/RtmpG2kdgr/remotes2
##   -  preparing 'RenDate':
##      checking DESCRIPTION meta-information ...  v  checking DESCRIPTION meta-information
##   -  checking for LF line-endings in source and make files and shell scripts
##   -  checking for empty or unneeded directories
##   -  looking to see if a 'data/datalist' file should be added
##        NB: this package now depends on R (>= 3.5.0)
##        WARNING: Added dependency on R >= 3.5.0 because serialized objects in  serialize/load version
##   -  building 'RenDate_1.0.3.0007.tar.gz'
##
##
```

```
library("RenDate") #, lib.loc="/Library/Frameworks/R.framework/Versions/3.6/Resources/library")
```

via source dans répertoire /R

```
# source('R/RenDate.R', echo=FALSE)
```

Memo

Quand on modifie le code, il faut penser à faire la documentation avec:

setwd("/Users/dufresne/Documents/R_Project/RenDate") devtools::document()

Chargement des courbes de calibration

Chargement des fichiers de calibration AM au format csv

```
# ne pas oublier d'enlever les points de référence à la fin !!!
GAL2002sph2014_I <- read.table("Calib/AM/GAL2002sph2014_I.csv", dec=',', sep=";",header=FALSE)
GAL2002sph2014_D <- read.table("Calib/AM/GAL2002sph2014_D.csv", dec=',', sep=";",header=FALSE)

gwh2013uni_f <- read.table("Calib/AM/gwh2013uni_f.csv", dec=',', sep=";",header=FALSE)
```

Chargement des fichiers de calibration 14C

```
 intCal13 <- read.csv("Calib/14C/intcal13.14c", header = FALSE, comment.char="#")
 intCal20 <- read.csv("Calib/14C/intcal20.14c", header = FALSE, comment.char="#")
```

Création et sauvegarde des fichiers rda. utilisés dans RenDate dans le répertoire courant

```
#library(Bchron)
# création courbe en AD/BC
createCalCurve("GAL2002sph2014_I", GAL2002sph2014_I$V1, GAL2002sph2014_I$V2, GAL2002sph2014_I$V3 )
createCalCurve("GAL2002sph2014_D", GAL2002sph2014_D$V1, GAL2002sph2014_D$V2, GAL2002sph2014_D$V3)
createCalCurve("gwh2013uni_f", gwh2013uni_f$V1, gwh2013uni_f$V2, gwh2013uni_f$V3)

# création courbe en BP
createCalCurve("GAL2002sph2014_I_BP",GAL2002sph2014_I$V1 - 1950, GAL2002sph2014_I$V2, GAL2002sph2014_I$V
createCalCurve("GAL2002sph2014_D_BP",GAL2002sph2014_D$V1 - 1950, GAL2002sph2014_D$V2, GAL2002sph2014_D$V
createCalCurve("gwh2013uni_f_BP", gwh2013uni_f$V1 - 1950, gwh2013uni_f$V2, gwh2013uni_f$V3)
```

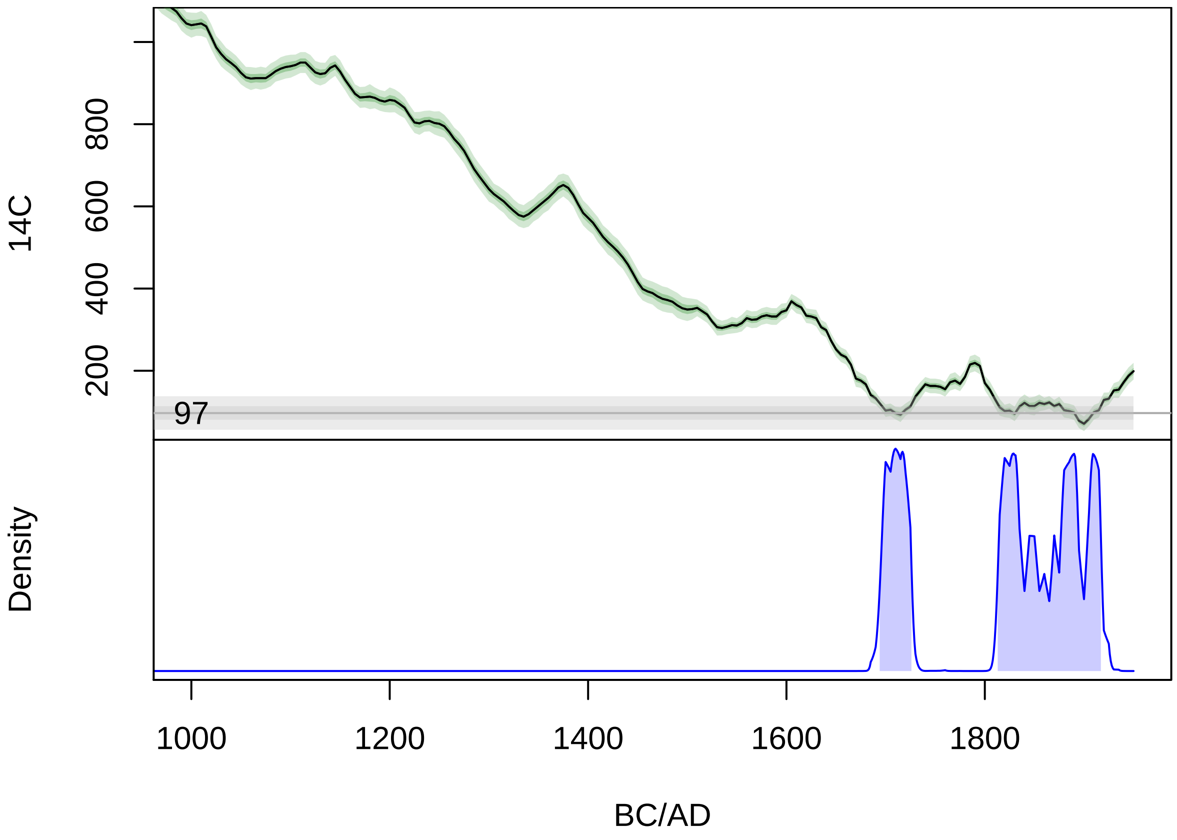
```
## Warning in Ops.factor(gwh2013uni_f$V1, 1950): '-' not meaningful for factors
```

```
intCal13_AD <- intCal13
intCal13_AD$V1 <- 1950 - intCal13_AD$V1
createCalCurve("IntCal13_AD", intCal13_AD$V1, intCal13_AD$V2, intCal13_AD$V3)
createCalCurve("IntCal13_BP", intCal13$V1, intCal13$V2, intCal13$V3)

intCal20_AD <- intCal20
intCal20_AD$V1 <- 1950 - intCal20_AD$V1
createCalCurve("IntCal20_AD", intCal20_AD$V1, intCal20_AD$V2, intCal20_AD$V3)
createCalCurve("IntCal20_BP", intCal20_AD$V1, intCal20_AD$V2, intCal20_AD$V3)
```

Datation

Datation 14C

```
C14 <- 97
errC14 <- 16
date14C <- calibrate(mesures = C14 ,std = errC14, calCurves = 'IntCal13_AD', ids = 'C14', timeScale = 1

# Tracé de la courbe 14C
 # réduction de la période
tmin <- 1000
tmax <- 1950
xlim <- c(tmin, tmax)

# Il faut mettre des valeur qui existent dans le tableau
imin <- which(intCal13_AD$V1 == tmin)
imax <- which(intCal13_AD$V1 == tmax)
ylim <- range(intCal13_AD$V2[imin:imax] )

par( fig=c(0, 1, 0.50, 1), mar=c(0, 5, 0, 1))

courbe_enveloppe(t=intCal13_AD$V1, mean=intCal13_AD$V2, intCal13_AD$V3, ylab = '14C', xlab = NA,  xaxt =
                 xlim = xlim, ylim = ylim)
mesure_enveloppe(intCal13_AD$V1, mesure = C14, std = errC14)
text(intCal13_AD$V1[imin], C14, labels=as.character(C14) )

par(fig=c(0, 1, 0.0, 0.50), new= TRUE, mar=c(5, 5, 0, 1) )

plot(date14C, col = "blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, xlab ='BC/AD',
     xlim = xlim, yaxt="n")
```



```
print('Resultat pour le 14C')
```

```
## [1] "Resultat pour le 14C"
```

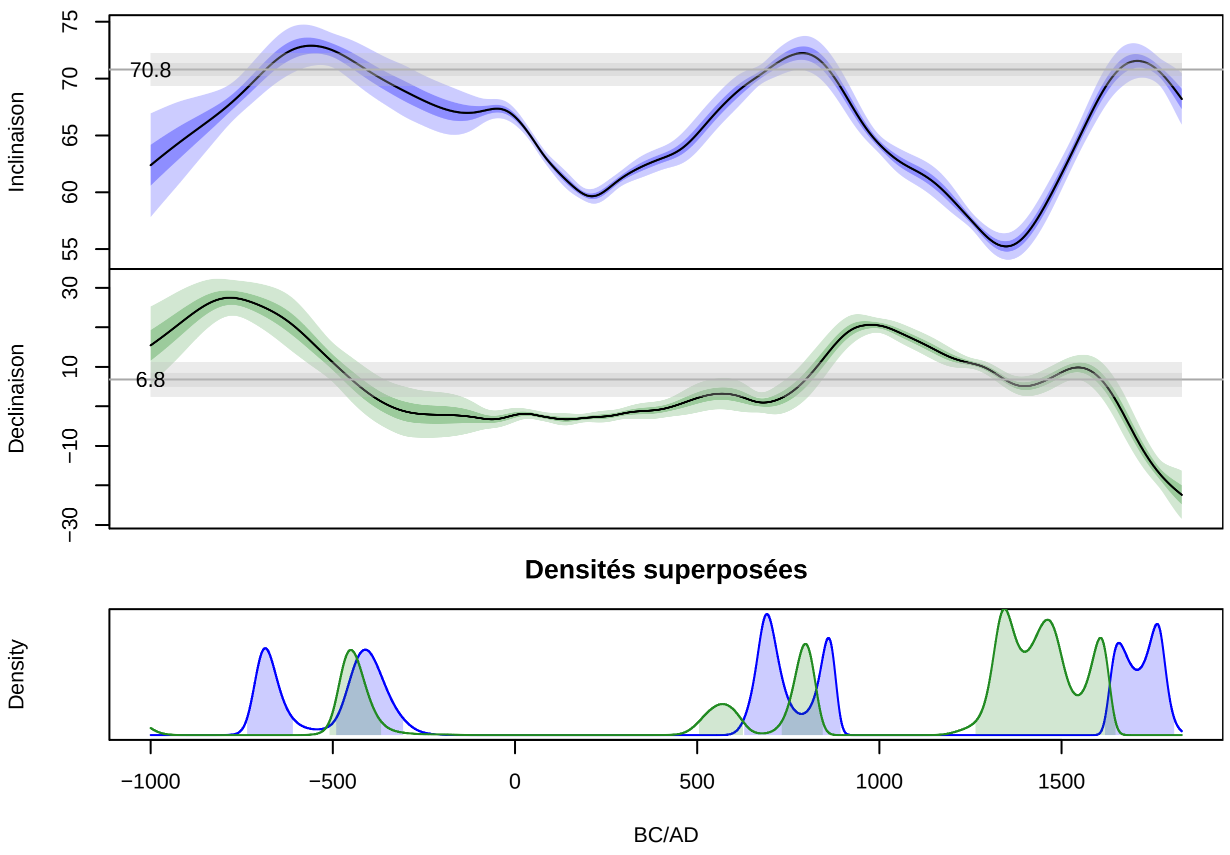
```
hpd(date14C$C14, prob = .955)
```

```
## $`27.3%`
## [1] 1694 1727
##
## $`68.5%`
## [1] 1813 1917
```

Datation AM

```
IParis = 70.8
DParis = 6.8
alpha95 = 1.38
# clcul des erreurs
errInc <- alpha95 /2.448
errDec <- alpha95/(2.448*cos(IParis*pi/180))

dateInc1 <- calibrate(mesures = IParis, std = errInc, calCurves='GAL2002sph2014_I', ids='Inclinaison', 

dateDec1 <- calibrate(mesures = DParis , std = errDec, calCurves='GAL2002sph2014_D', ids='Declinaison', 
# Exemple : plusieurs dates en même temps
#dateIncDec <- calibrate(mesures = c(IParis, DParis), std = c(errInc, errDec), calCurves = c('GAL2002sp
```

```
# Tracé de la superposition des deux densités obtenues mfrow = c(3,1),

# RenDate.plot(dateIncDec, col = "blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2), main="Densités p

 par(cex=0.7)
# Tracé de la courbe Inclinaison
par( fig=c(0, 1, 0.70, 1), new=FALSE, mar=c(0, 5, 0.2, 0))
courbe_enveloppe(GAL2002sph2014_I$V1, GAL2002sph2014_I$V2, GAL2002sph2014_I$V3, ylab='Inclinaison', xla
mesure_enveloppe(GAL2002sph2014_I$V1, mesure = IParis, std = errInc)
text(GAL2002sph2014_I$V1[1], IParis, labels=as.character(IParis) )
# Tracé de la courbe Déclinaison
par(fig=c(0, 1, 0.40, 0.70), new=TRUE, mar=c(0, 5, 0, 0))
courbe_enveloppe(GAL2002sph2014_D$V1, GAL2002sph2014_D$V2, GAL2002sph2014_D$V3, ylab='Declinaison', xla
mesure_enveloppe(GAL2002sph2014_D$V1, mesure = DParis, std = errDec)
text(GAL2002sph2014_D$V1[1], DParis, labels=as.character(DParis) )

# Tracé de la superposition des deux densités obtenues
par(fig=c(0, 1, 0.0, 0.40), new=TRUE, mar=c(5, 5, 3, 0))
plot(dateInc1, col = "blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2), main="Densités superposées",
lines(dateDec1, col="forestgreen", hdrCol = adjustcolor( "forestgreen", alpha.f = 0.2))
```



Statistique sur une datation

Recherche du MAP

```
#MAP(date14C$C14)
```

Recherche des Quantiles

```
quantile(date14C$C14)
```

```
## $`25.5547665801745%`
## [1] 1722
##
## $`50.2335754724071%`
## [1] 1838
##
## $`75.8879714309055%`
## [1] 1886
```

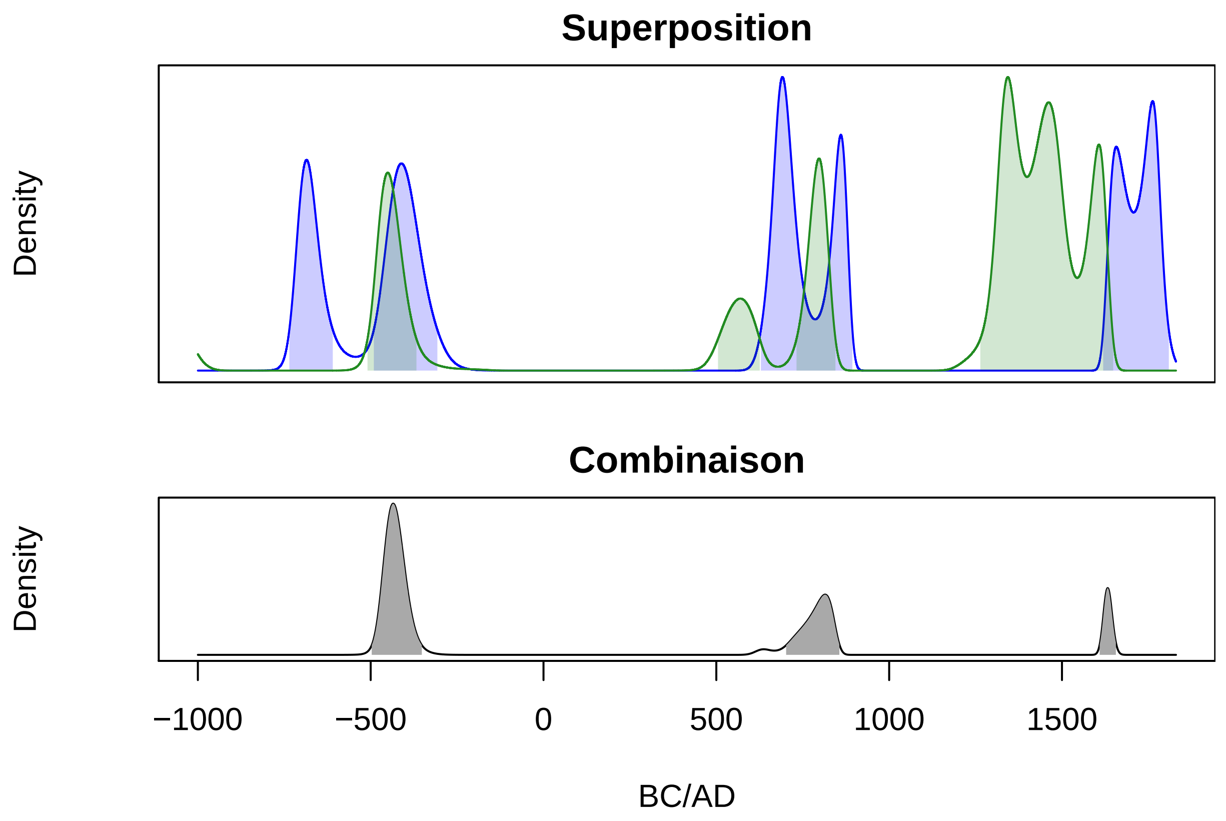
Recherche du HPD

```
hpd(date14C$C14)
```

```
## $`27%`
## [1] 1694 1726
##
## $`68.5%`
## [1] 1813 1917
```

Combinaison AM

```
combiAM<- produit(dateInc1, dateDec1, timeScale = 1)

par(mfrow = c(2,1), mar=c(1, 5, 2, 0))

plot(dateInc1, col = "blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2), xlab=NA, main='Superposition
lines(dateDec1, col="forestgreen", hdrCol = adjustcolor( "forestgreen", alpha.f = 0.2),  normalize = TRU

par( mar=c(5, 5, 2, 0))
val.hdr <- paste(hpd(combiAM$Combinaison, prob = .95), ' à 95% BC/AD' )
plot(combiAM, withHDR = TRUE, main='Combinaison' ,  normalize = TRUE, yaxt="n", xlab='BC/AD')
```



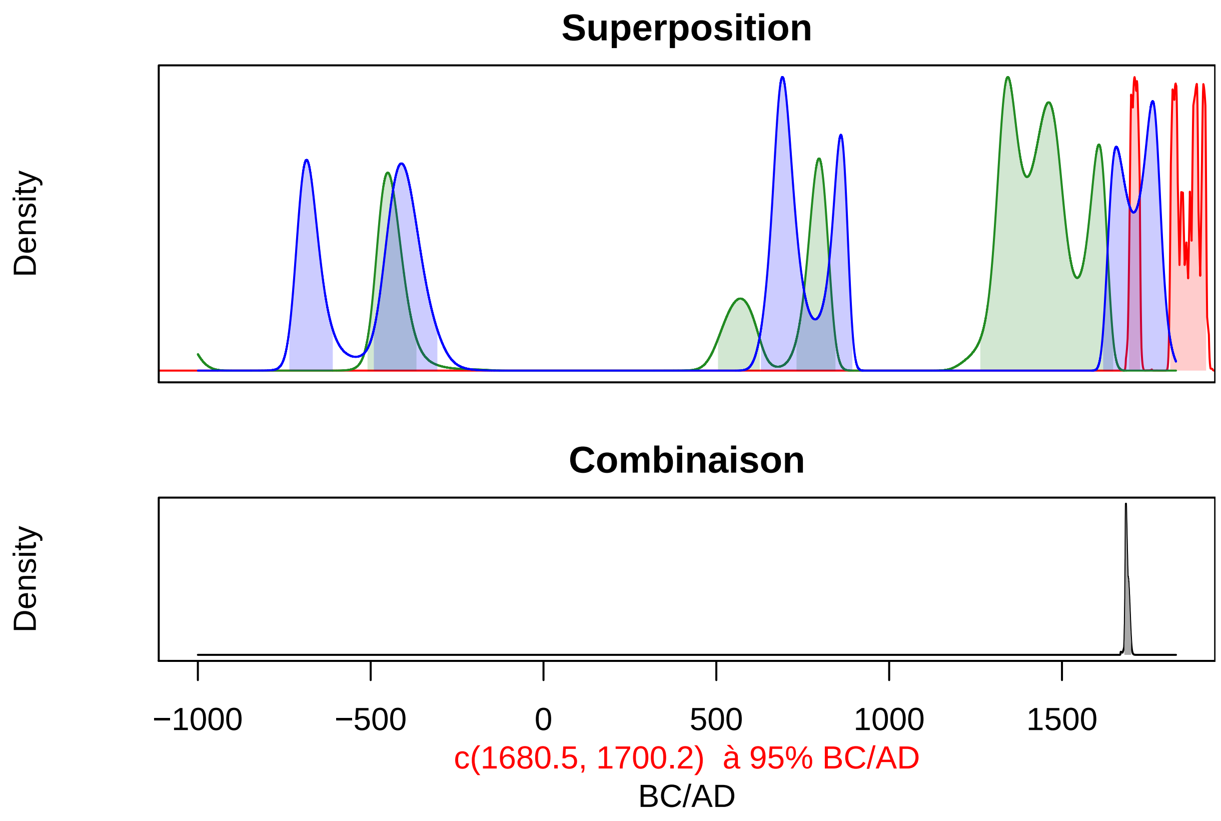
```r
#mtext(val.hdr, side = 1, col = 'red')

hpd(combiAM$Combinaison, prob = .95)
```

```
## $`56.9%`
## [1] -497 -352
##
## $`28%`
## [1] 702 856
##
## $`10.1%`
## [1] 1609 1656
```

Combinaison AM et 14C

```r
combiAMC14<- produit(combiAM, date14C, timeScale = 0.1)

par(mfrow = c(2,1), mar=c(1, 5, 2, 0))

xlim <- range(combiAMC14$Combinaison$timeGrid)
plot(date14C, col = "red", hdrCol = adjustcolor( "red", alpha.f = 0.2), xlab=NA, xlim = xlim, main='Supe
lines(dateDec1, col="forestgreen", hdrCol = adjustcolor( "forestgreen", alpha.f = 0.2),  normalize = TRU
lines(dateInc1, col="blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2),  normalize = TRUE)

par( mar=c(5, 5, 2, 0))
```

```
val.hdr <- paste( hpd(combiAMC14$Combinaison, prob = .95), ' à 95% BC/AD' )
plot(combiAMC14, withHDR = TRUE, main='Combinaison', normalize = TRUE ,  yaxt="n", xlab="BC/AD")
mtext(val.hdr, side = 1, col = 'red', line = 2)
```



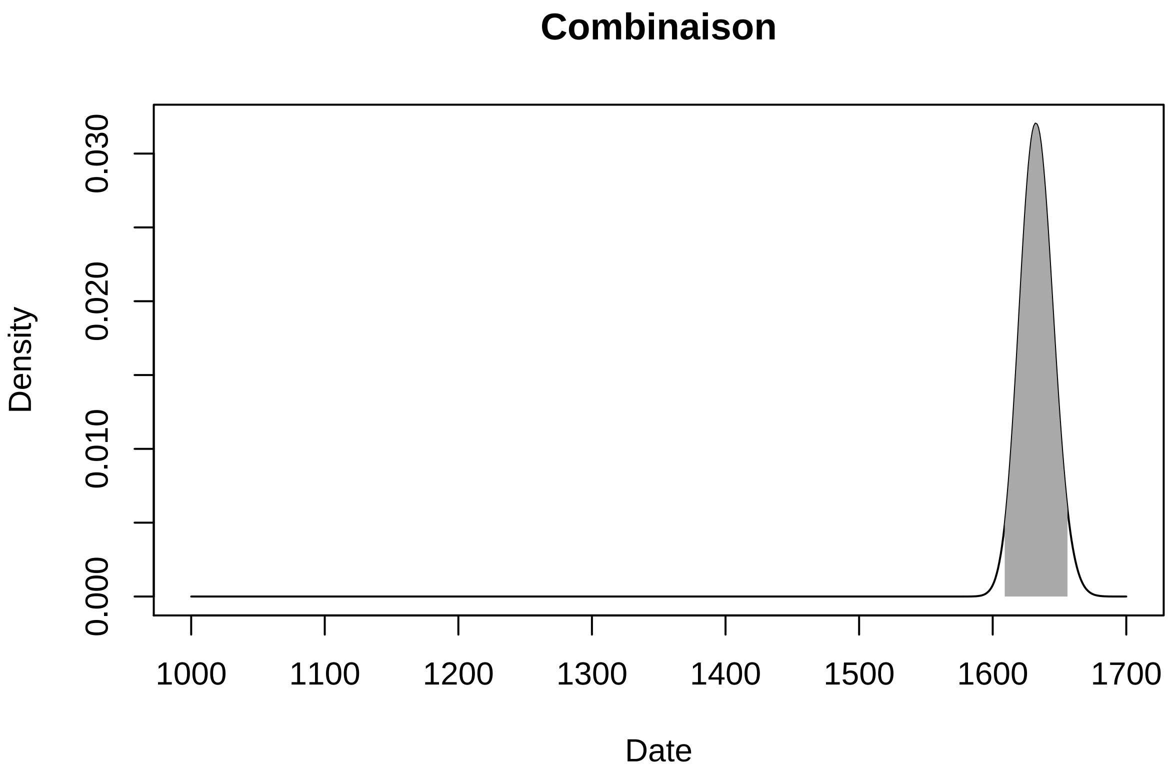
```
hpd(combiAMC14$Combinaison, prob = .95)
```

```
## $`95%`
## [1] 1680.5 1700.2
```

```
 # réduction de la période
tmin <- 1000
tmax <- 1700
# Il faut mettre des valeurs qui existent dans le tableau
imin <- which(combiAM$Combinaison$timeGrid == tmin)
imax <- which(combiAM$Combinaison$timeGrid == tmax)
 tmp <- combiAM
 tmp$Combinaison$timeGrid <- tmp$Combinaison$timeGrid[imin:imax]
 tmp$Combinaison$densities <- tmp$Combinaison$densities[imin:imax]
 tmp$Combinaison$densities <-tmp$Combinaison$densities /sum(tmp$Combinaison$densities)

# Affichage résultat
hpd(tmp$Combinaison, prob = .95)
```

```
## $`94.6%`
## [1] 1609 1656
```

```
plot(tmp)
```



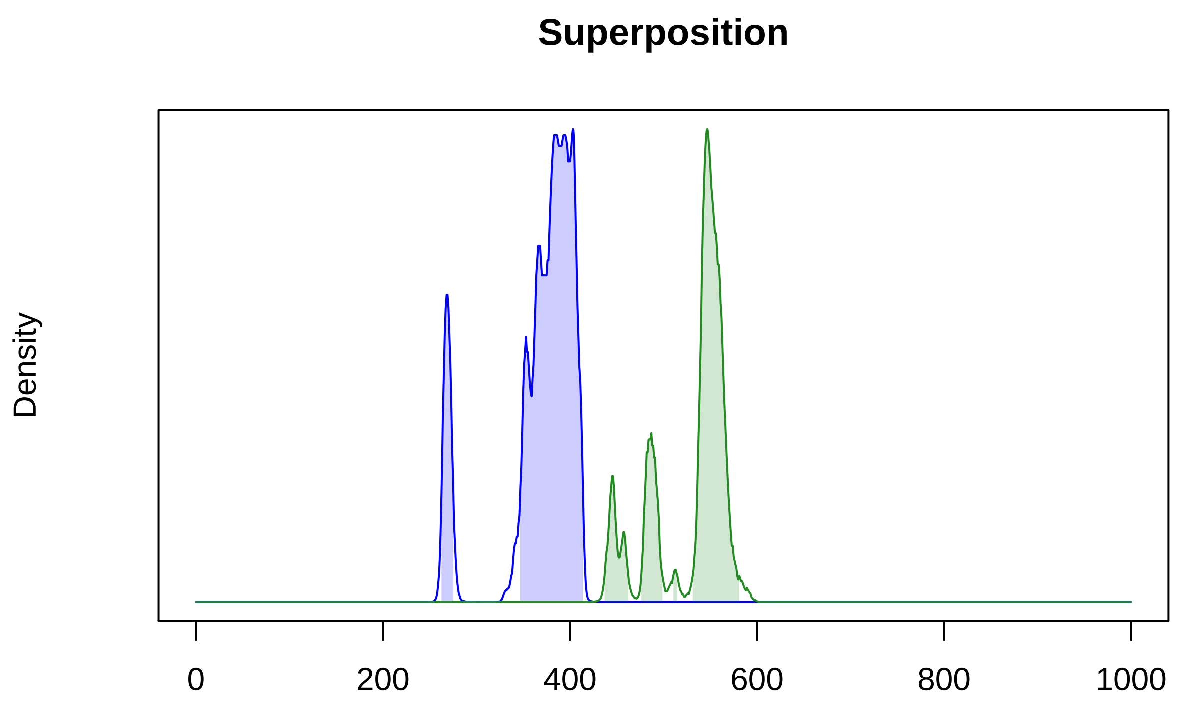
Wiggle Matching type Produit

Wiggle fixe

Code Oxcal: Options() { Curve("IntCal20","intcal20.14c"); Resolution=1; }; Plot() { D_Sequence() { R_Date("ULA 9174", 1690, 15); Gap(130); R_Date("ULA 9175", 1545, 15); }; };

```
C14.1 <- 1690
errC14.1 <- 15
date14C.1 <- calibrate(mesures = C14.1 ,std = errC14.1, calCurves = 'IntCal20_AD', ids = 'C14', timeScal


C14.2 <- 1545
errC14.2 <- 15
date14C.2 <- calibrate(mesures = C14.2 ,std = errC14.2, calCurves = 'IntCal20_AD', ids = 'C14', timeScal

wiggle = 130
```

```
plot(period_reduction( date14C.1, 0, 1000), col = "blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2),
lines(period_reduction(date14C.2, 0, 1000), col="forestgreen", hdrCol = adjustcolor( "forestgreen", alpl
```

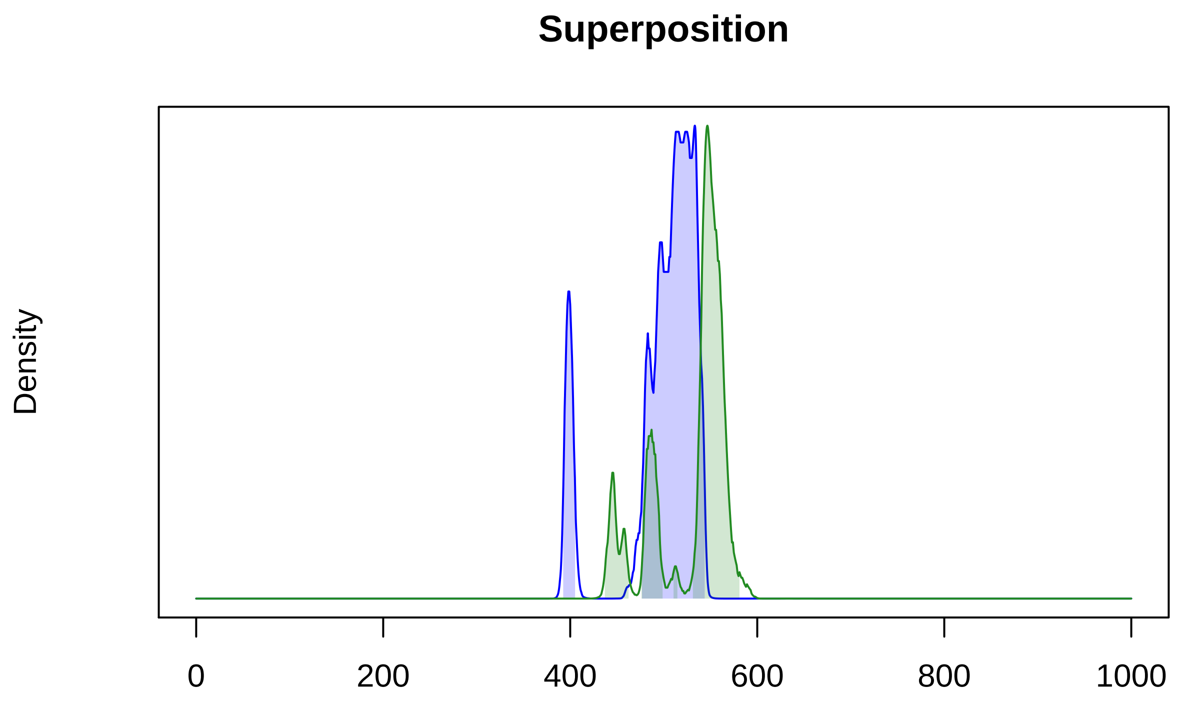


Calcul de la densité shiftée

Dans notre cas, c'est la 14C.1 qui est décalée vers 14C.2

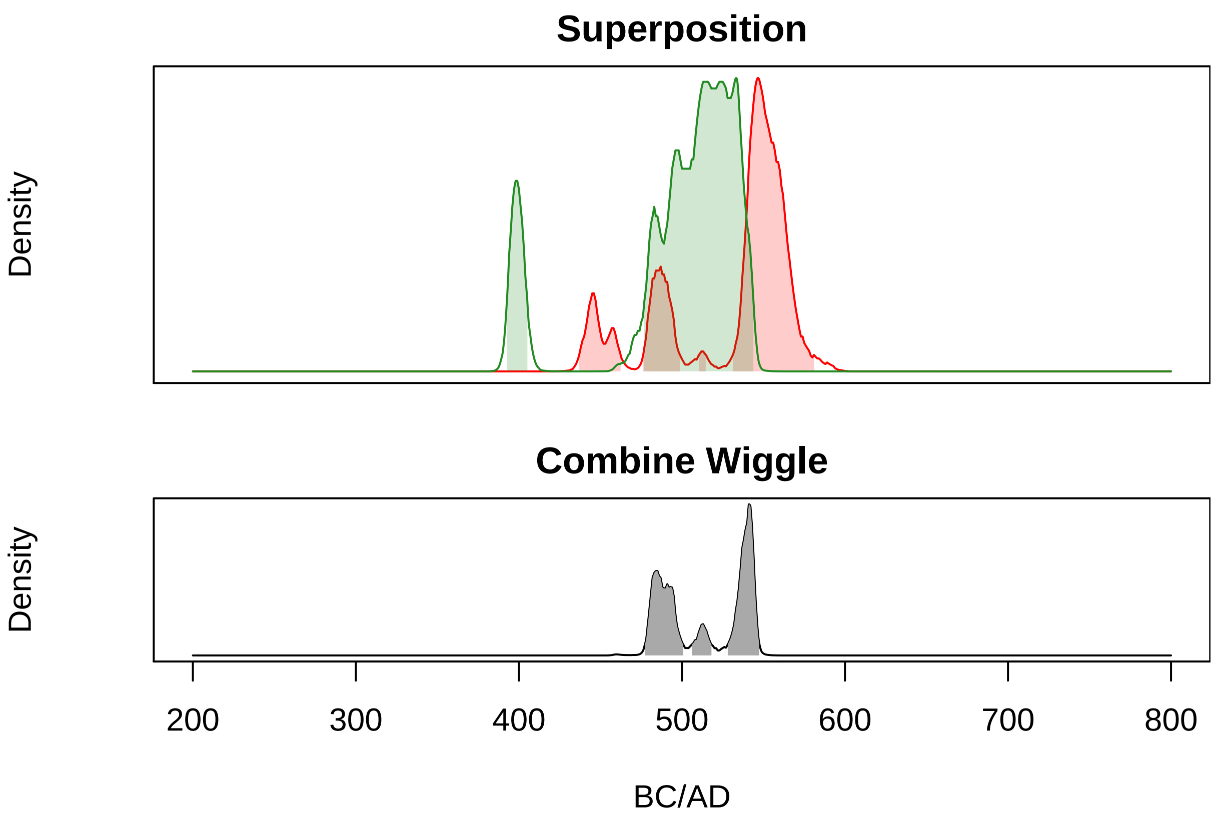
```
date14C.1.shifted <- date14C.1
date14C.1.shifted$C14$timeGrid <- date14C.1$C14$timeGrid + wiggle
```

```
plot(period_reduction( date14C.1.shifted, 0, 1000), col = "blue", hdrCol = adjustcolor( "blue", alpha.f
lines(period_reduction(date14C.2, 0, 1000), col="forestgreen", hdrCol = adjustcolor( "forestgreen", alp
```



Résultat wiggle

```
wiggleC14 <- produit(date14C.1.shifted, date14C.2, timeScale = .1)

par(mfrow = c(2,1), mar=c(1, 5, 2, 0))
tmin = 200
tmax = 800
xlim <- range(period_reduction(wiggleC14, tmin, tmax)$Combinaison$timeGrid)
plot(period_reduction(date14C.2, tmin, tmax), col = "red", hdrCol = adjustcolor( "red", alpha.f = 0.2),
lines(period_reduction(date14C.1.shifted, tmin, tmax), col="forestgreen", hdrCol = adjustcolor( "forestg

par( mar=c(5, 5, 2, 0))
val.hdr <- paste( hpd(wiggleC14$Combinaison, prob = .95), ' à 95% BC/AD' )
plot(period_reduction(wiggleC14, tmin, tmax), withHDR = TRUE, main='Combine Wiggle', normalize = TRUE ,
```



```
#mtext(val.hdr, side = 1, col = 'red', line = 2)

hpd(wiggleC14$Combinaison, prob = .954)
```

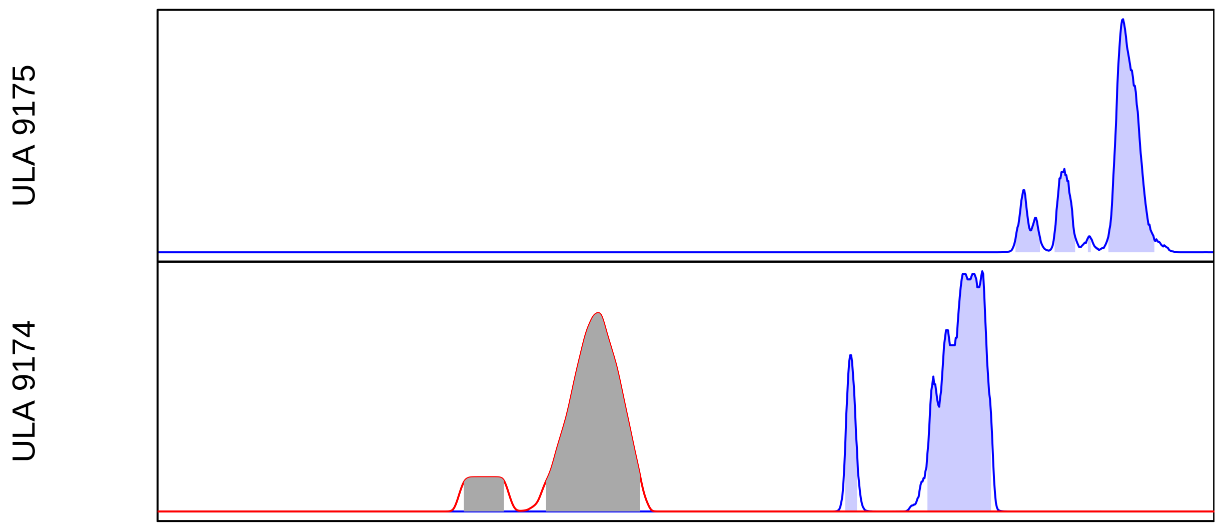
```
## $`41.1%`
## [1] 477.2 501.0
##
## $`8.1%`
## [1] 505.8 518.4
##
## $`46.2%`
## [1] 527.9 547.5
```

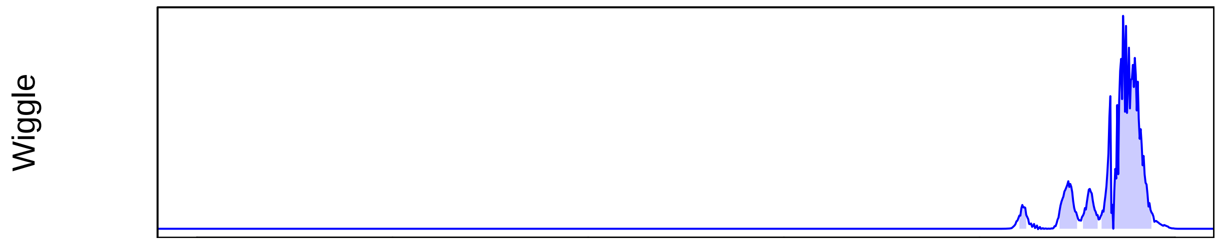
Wiggle intervalle Uniforme

Remarque avec OxCal: Si on utilise D_Sequence, nous ne pouvons pas mettre d'erreur sur le gap(). Il faut utiliser une V_Sequence et alors le gap() est gaussien * Code Oxcal

Options() { Curve("IntCal20","intcal20.14c"); Resolution=.1; }; Plot() { D_Sequence() { R_Date("ULA 9174", 1690, 15); Gap(130); R_Date("ULA 9175", 1545, 15); }; };

```
# 1ere date
date14C.9174 <- calibrate(mesures = 1690 , std = 15, calCurves = 'IntCal20_AD',  ids = '9174', pathToCal

# 2eme date

date14C.9175 <- calibrate(mesures = 1545 , std = 15, calCurves = 'IntCal20_AD',  ids = '9175', pathToCal

tmin <- -400
tmax <- 600
xlim <- c(tmin, tmax)

# date14C <- c(date14C, wiggle.uniform(date14C, 200, 200) )
# class(date14C) = "RenDate"
par( fig=c(0, 1, 0.70, 1), new=FALSE, mar=c(0, 5, 0.2, 0))
plot(date14C.9175, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, 

# Wiggle

date14C.9174.wiggle <-  wiggle_uniform(date14C.9174, -400, -350 )
par(fig=c(0, 1, 0.40, 0.70), new=TRUE, mar=c(0, 5, 0, 0))
plot(date14C.9174, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, 
lines(date14C.9174.wiggle, col ="red", xaxt="n")

# Combinaison

D_Sequence <- produit(date14C.9174.wiggle, date14C.9175, timeScale = date14C.9174.wiggle[[1]]$timeScale
par(fig=c(0, 1, 0.0, 0.40), new=TRUE, mar=c(0, 5, 3, 0))
plot(D_Sequence, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, xl
```





```
hpd(date14C.9174[[1]])
```

```
## $`10.9%`
## [1] 263 275
##
## $`83.6%`
## [1] 347 412
```

```
hpd(date14C.9174.wiggle[[1]])
```

```
## $`10.1%`
## [1] -127  -86
##
## $`84.8%`
## [1] -43  53
```

```
print("Combinaison")
```

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

```
hpd(D_Sequence$Combinaison)
```

```
## $`2.4%`
## [1] 441 448
##
## $`9.7%`
## [1] 482 500
##
## $`6.4%`
## [1] 506 521
##
## $`76.5%`
## [1] 525 576
```

Wiggle intervalle gaussien

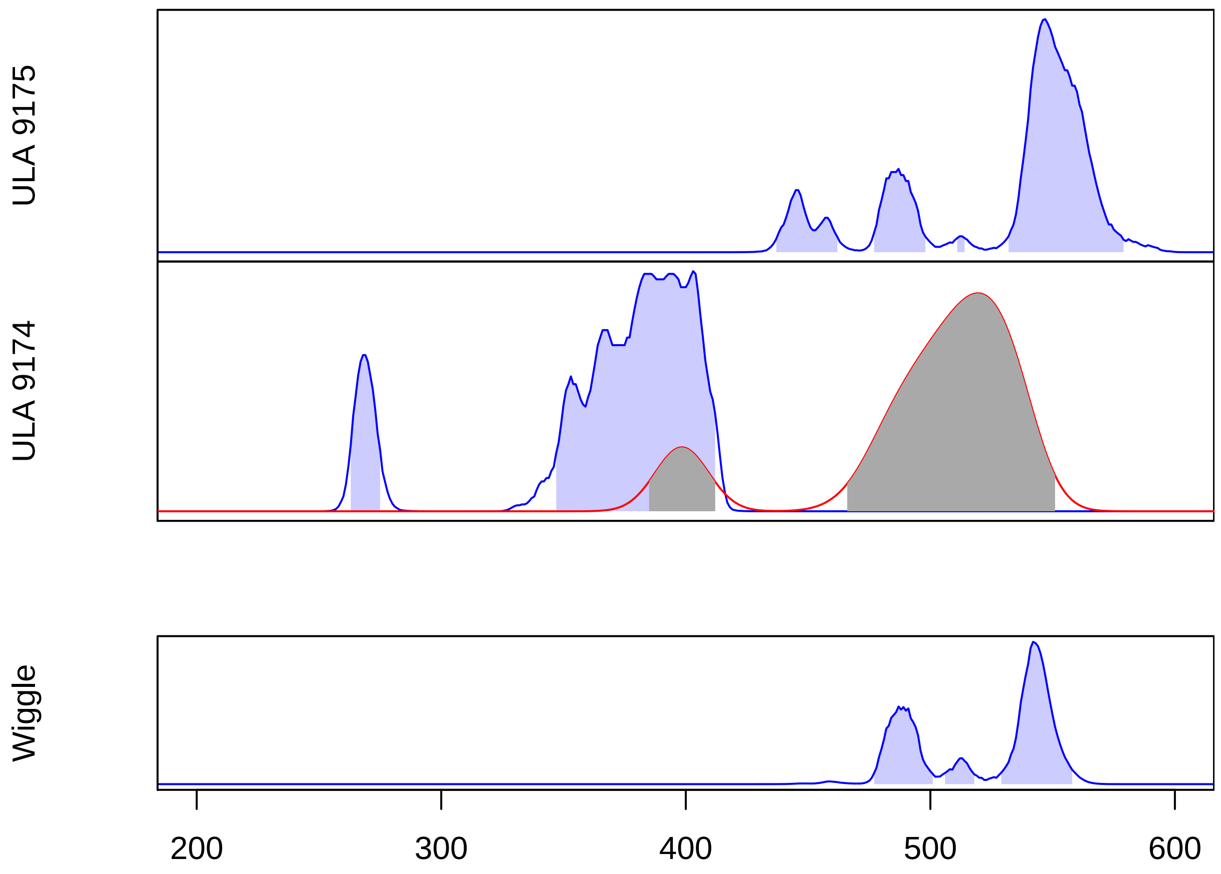
OxCal donne le résultat par calcul MCMC et donne les "a posteriori" des deux datations. Ici nous calculons l'une puis l'autre ensuite. Comme dit précédement, il faut utiliser la fonction V_Sequence pour pouvoir utiliser gap() avec une incertitude * Code OxCal

Options() { Curve("IntCal20","intcal20.14c"); Resolution=.1; }; Plot() { V_Sequence() { R_Date("ULA 9174", 1690, 15); Gap(130, 10); R_Date("ULA 9175", 1545, 15); }; };

On recherche la date la plus récente

On reprend le même exemple que pour le wiggle uniform. Remarque avec Oxcal, le résultat correspond à la date "ULA 9175"

```
# 1ere date
date14C.9174 <- calibrate(mesures = 1690 , std = 15, calCurves = 'IntCal20_AD',  ids = '9174', pathToCal

# 2eme date

date14C.9175 <- calibrate(mesures = 1545 , std = 15, calCurves = 'IntCal20_AD',  ids = '9175', pathToCal

tmin <- 200
tmax <- 600
xlim <- c(tmin, tmax)

par( fig=c(0, 1, 0.70, 1), new=FALSE, mar=c(0, 5, 0.2, 0))
plot(date14C.9175, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, 

# Wiggle

date14C.9174.wiggle <-  wiggle_gauss(date14C.9174, 130, 10)
par(fig=c(0, 1, 0.40, 0.70), new=TRUE, mar=c(0, 5, 0, 0))
plot(date14C.9174, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, 
lines(date14C.9174.wiggle, col ="red", xaxt="n")

# Combinaison

V_Sequence <- produit(date14C.9174.wiggle, date14C.9175, timeScale = date14C.9174.wiggle[[1]]$timeScale
par(fig=c(0, 1, 0.0, 0.40), new=TRUE, mar=c(2, 5, 3, 0))
plot(V_Sequence, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, xla
```



```
hpd(date14C.9174[[1]])
```

```
## $`10.9%`
## [1] 263 275
##
## $`83.6%`
## [1] 347 412
```

```
hpd(date14C.9174.wiggle[[1]])
```

```
## $`10%`
## [1] 385 412
##
## $`84.9%`
## [1] 466 551
```

```
print("Combinaison")
```

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

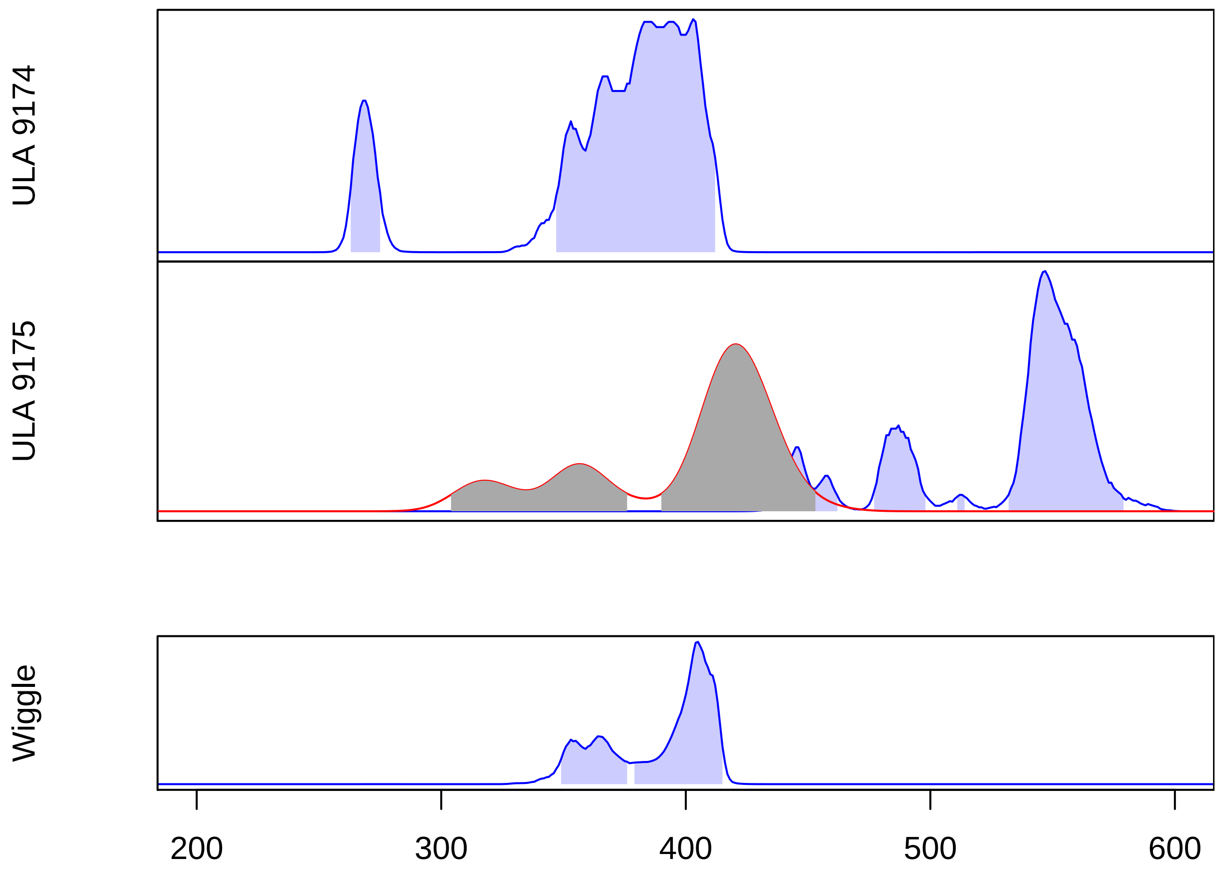
```
hpd(V_Sequence$Combinaison, prob= 0.954)
```

```
## $`32.2%`
## [1] 477 501
##
## $`6.5%`
## [1] 505 518
##
## $`56.4%`
## [1] 529 558
```

On recherche la date la plus vieille

Cette fois, c'est la plus récente qu'on décale vers la date plus vieille 9174. C'est le même code OxCal Résultat OxCal: 349 (95.4%) 416 AD

```
# 1ere date
date14C.9174 <- calibrate(mesures = 1690 , std = 15, calCurves = 'IntCal20_AD',  ids = '9174', pathToCal

# 2eme date

date14C.9175 <- calibrate(mesures = 1545 , std = 15, calCurves = 'IntCal20_AD',  ids = '9175', pathToCal

tmin <- 200
tmax <- 600
xlim <- c(tmin, tmax)

par( fig=c(0, 1, 0.70, 1), new=FALSE, mar=c(0, 5, 0.2, 0))
plot(date14C.9174, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, 

# Wiggle

date14C.9175.wiggle <-  wiggle_gauss(date14C.9175, -130, 10)
par(fig=c(0, 1, 0.40, 0.70), new=TRUE, mar=c(0, 5, 0, 0))
plot(date14C.9175, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, 
lines(date14C.9175.wiggle, col ="red", xaxt="n")

# Combinaison

V_Sequence <- produit(date14C.9175.wiggle, date14C.9174, timeScale = date14C.9174.wiggle[[1]]$timeScale
par(fig=c(0, 1, 0.0, 0.40), new=TRUE, mar=c(2, 5, 3, 0))
plot(V_Sequence, col = "blue", xlim = xlim, hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, xla
```



```
hpd(date14C.9174[[1]])
```

```
## $`10.9%`
## [1] 263 275
##
## $`83.6%`
## [1] 347 412
```

```
hpd(date14C.9174.wiggle[[1]])
```

```
## $`10%`
## [1] 385 412
##
## $`84.9%`
## [1] 466 551
```

```
print("Combinaison")
```

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

```
hpd(V_Sequence$Combinaison, prob= 0.954)
```

```
## $`27.9%`
## [1] 349 376
##
## $`67.1%`
## [1] 379 415
```

Lecture fichier Ref

```
# si on a l'erreur :la chaîne de caractères entrée 1 est incorrecte dans cet environnement linguistique
# cela correspond au mauvais encoding
read.Ref <- function(file.Ref, encoding = "macroman")
{
  # Lecture et extraction des points de la courbe
  file.Ref="Calib/AM/GAL2002sph2014_I.ref"
  lin<- NULL
  fil <- file(file.Ref, "r", encoding = "macroman") #, encoding = encoding)
  lin <- readLines(fil)
  close(fil)
  # Recherche position-ligne des points de référence
  g <- NULL
  # compte le nombre de mesures
  n.measures <-0
  for (i in 1:length(lin)) {
    if (as.logical(length(grep("#", lin[i])))==FALSE  && as.logical(length(grep("/", lin[i])) )==FALSE
      n.measures <- n.measures + 1

    if (length(grep("# reference points", lin[i])) ){
      g<- i
      break
    }
  }

  list <- NULL
  list$measures<- read.table(file.Ref, dec='.', sep=",",header=FALSE, skip = i-n.measures-1, comment.ch
  colnames(list$measures) <- c("t", "value", "sigma")

  if (g<length(lin) ) {
    list$pts.ref<- read.table(file.Ref, dec='.', sep=",",header=FALSE, skip = i+1 , comment.char = "#")
  colnames(list$pts.ref) <- c("No", "tij1", "tij2", "tm", "Yij", "Err_Yijc")

  }

  #-------

  return(list)
}
```

```r
test <- read.Ref("Calib/AM/GAL2002sph2014_I.ref")
```

Creation d'une densité à partir d'une trace (vecteur de points)

```r
Chain_A <- read.csv2("~/Documents/R_Project/RenDate/Chain_A.csv", comment.char="#")
Chain_B <- read.csv2("~/Documents/R_Project/RenDate/Chain_B.csv", comment.char="#")
density_A.end <-trace_to_date(Chain_A$End)
density_B.begin <-trace_to_date(Chain_B$Begin)
```

Simulation d'une date entre 3F2 et 3C3

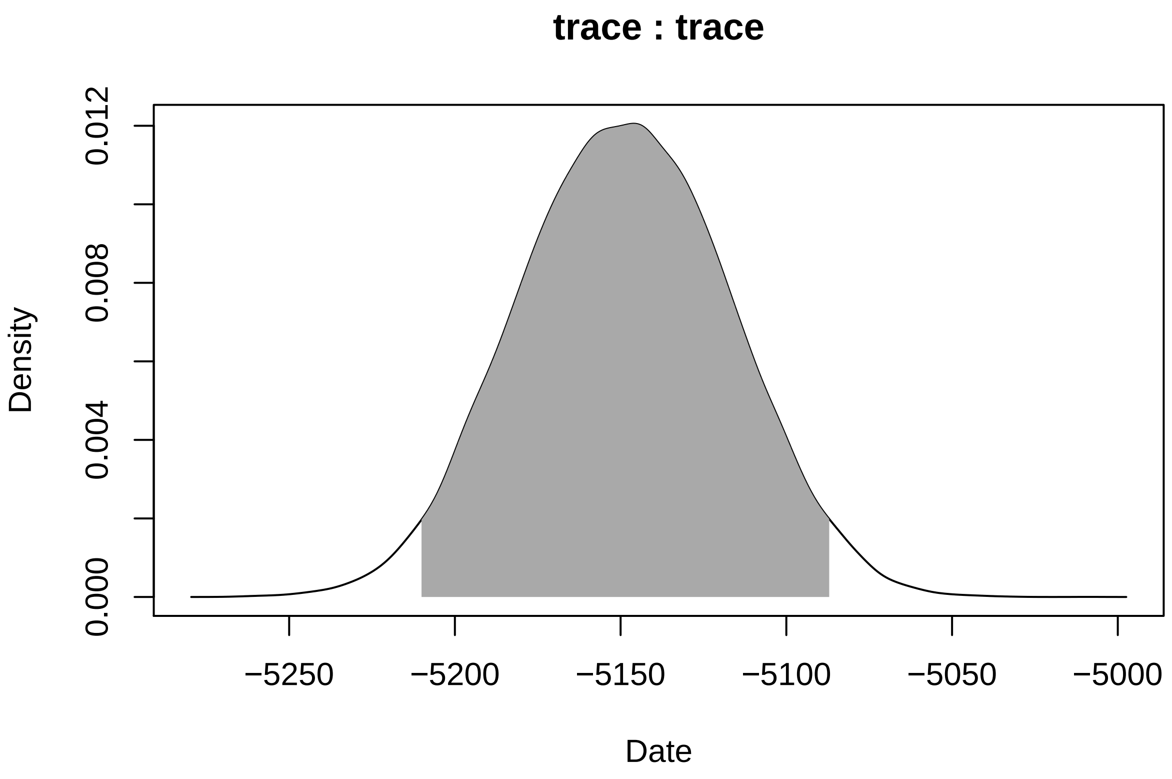
Random value between Unit.B and Unit.D

$$tc = tb + Unif[0;1] * (td - tb)$$

OR

$$tc = Unif[tb; td]$$

```r
Chain_III <- pred_value(Chain_A$End, Chain_B$Begin)
plot(trace_to_date(Chain_III))
```



test trouve valeur et erreur d'un age 14C à partir du HPD

Le but est de faire une routine qui retrouve la valeur (l'age) et l'erreur à partir d'une hpd - 1694 AD (28.1%) 1725 AD -1810 AD (67.4%) 1917 AD

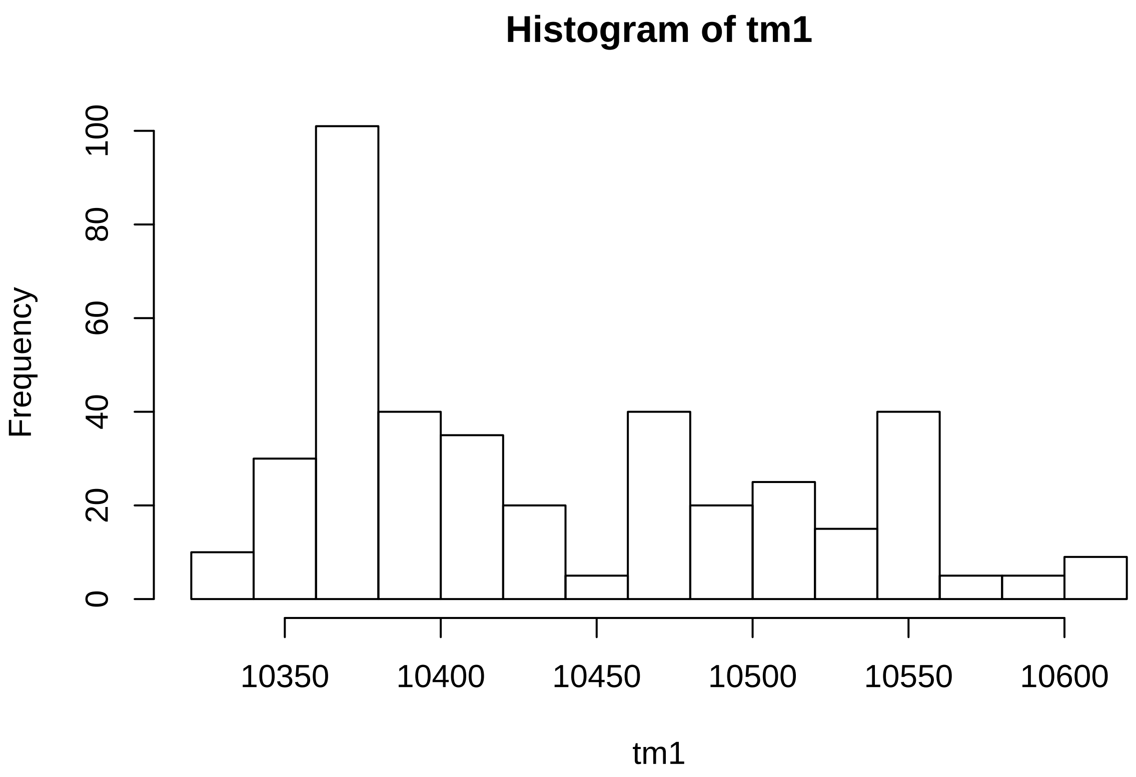
2eme test [1] "Resultat pour le 14C" `$86.5%` [1] -10684 -10426

`$7.6%` [1] -10396 -10285

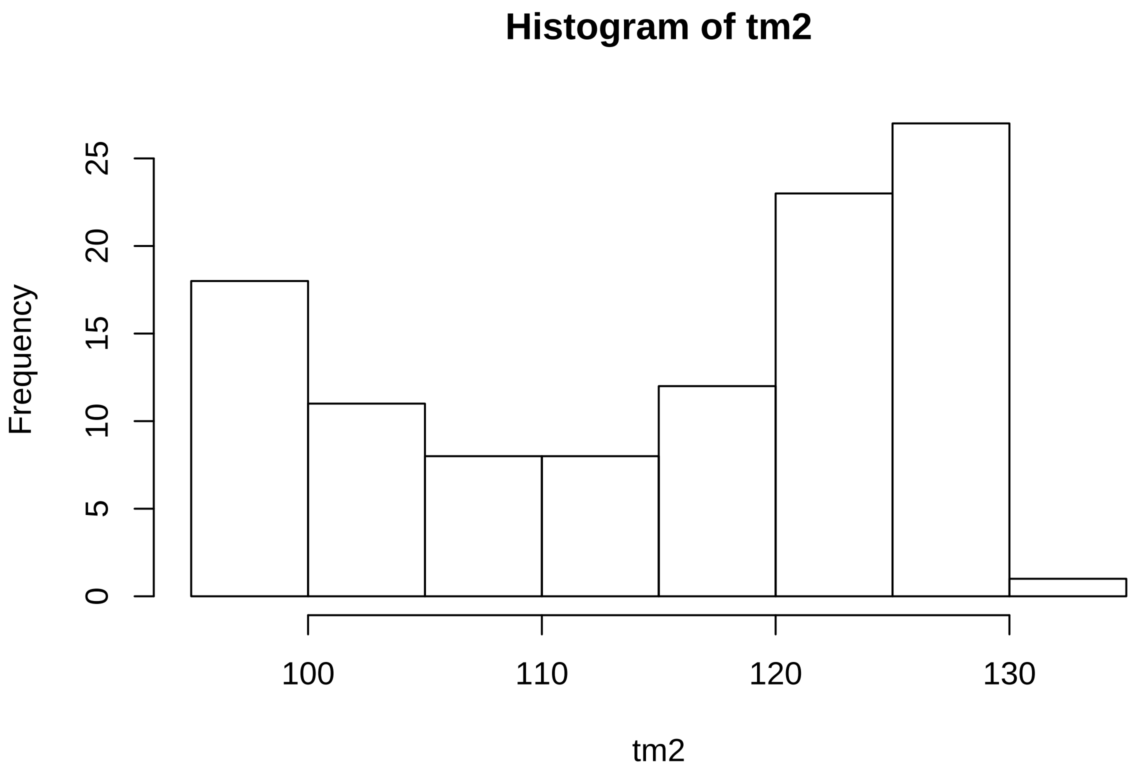
`$0.8%` [1] -10250 -10232

`$0.6%` [1] -10229 -10215

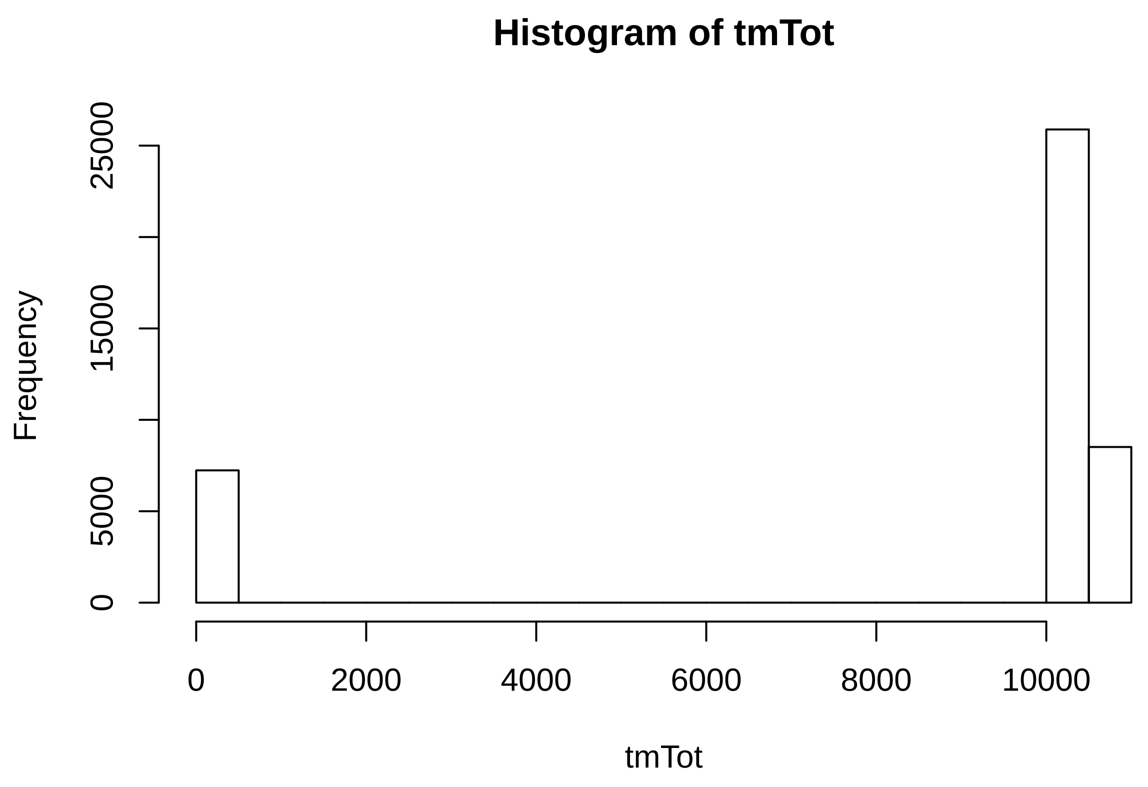
```
time.step = 1

find.tmin = -10684
find.tmax = -10285
confidence1 = 86.5

#find
m1 <- intCal20_AD[max(which(intCal20_AD$V1<=find.tmin) ), ]
m2 <- intCal20_AD[max(which(intCal20_AD$V1<=find.tmax) ), ]

tm1 <- NULL
for (m in seq(find.tmin, find.tmax, by=time.step)) {
  tm1 <- c(tm1, intCal20_AD[max(which(intCal20_AD$V1<=m) ), ]$V2)
}
hist(tm1)
```



```
tmTot <- rep(tm1, trunc(confidence1))
find.tmin = 1810
find.tmax = 1917
confidence2 = 67.4

#find
m1 <- intCal20_AD[max(which(intCal20_AD$V1<=find.tmin) ), ]
m2 <- intCal20_AD[max(which(intCal20_AD$V1<=find.tmax) ), ]

tm2 <- NULL
for (m in seq(find.tmin, find.tmax, by=time.step)) {
  tm2 <- c(tm2, intCal20_AD[max(which(intCal20_AD$V1<=m) ), ]$V2)
}
 hist(tm2)
```



```
 tmTot <- c(tmTot, rep(tm2, trunc(confidence2)))

 hist(tmTot)
```



```
mean(tmTot)
```

```
## [1] 8642.75
```

```
sqrt(var(tmTot) )
```

```
## [1] 3911.449
```

```
find.tmin = -10684
find.tmax = 10285


#find
id.min = intCal20_AD$V1 > find.tmin
id.max = intCal20_AD$V1 < find.tmax


v.find <- intCal20_AD[id.max&id.min, 2]
age.min <- min( v.find)
age.max <- max( v.find)
```

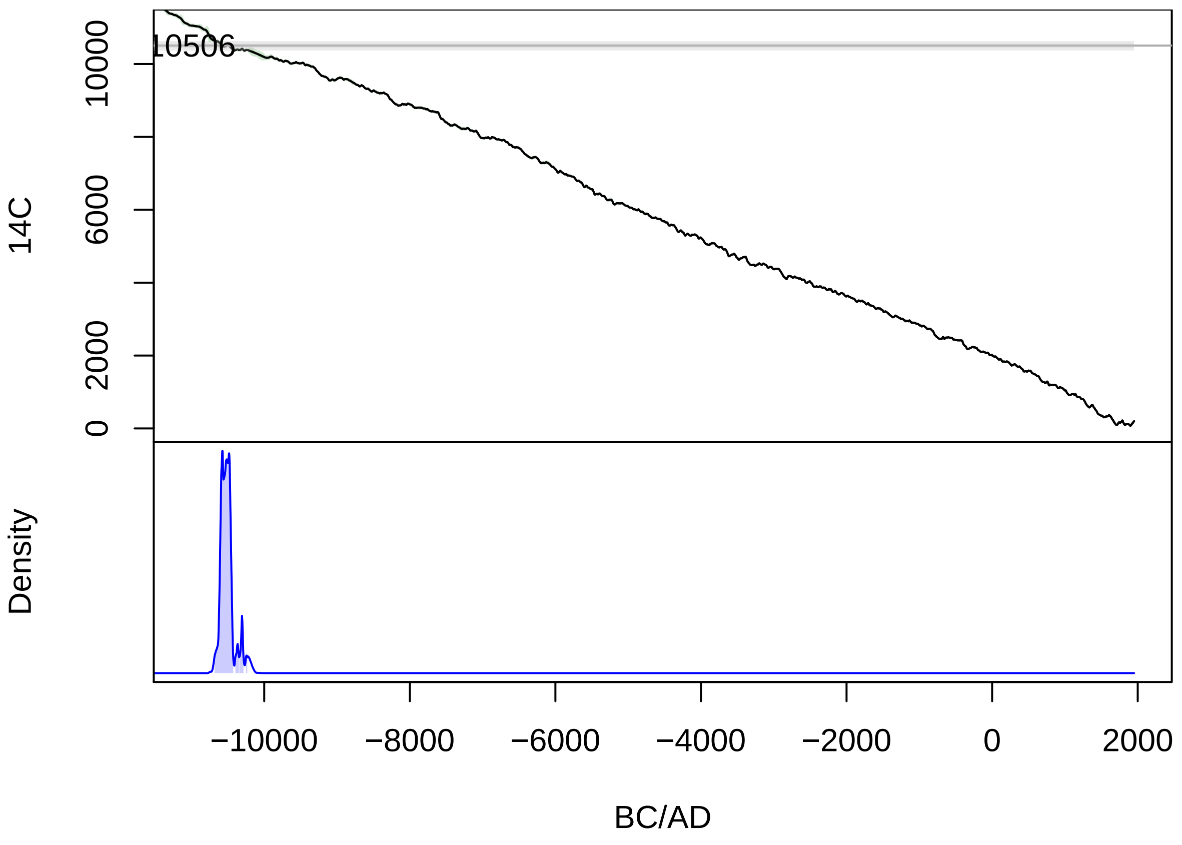
```
# routine très longue
v.min = age.min
v.max = age.max
v.step = 1
```

```r
e.min = 5
e.max = 50
e.step = 1

hpd.res <- NULL
hpd.tmp <- NULL
hpd.tmp.form <- NULL

prob.total = 95.5

hpd.res = vector('list')
j <- 1
for (v in seq(v.min, v.max, by=v.step)) {
  for (e in seq(e.min, e.max, by=e.step)) {
    date14C <- calibrate(mesures = v ,std = e, calCurves = 'IntCal13_AD', ids = 'C14', timeScale = 1)

    hpd.tmp <- hpd(date14C$C14, prob = prob.total/100)
    hpd.tmp.form = vector('list')
    for (i in 1:length(hpd.tmp)) {
      conf <- as.numeric( trimws(substr(names(hpd.tmp[i]), 0, nchar(names(hpd.tmp[i]))-1 ) ) )
      hpd.tmp.form$conf[i] <- conf
      hpd.tmp.form$tmin[i] <- hpd.tmp[[i]] [1]
      hpd.tmp.form$tmax[i] <- hpd.tmp[[i]] [2]
      hpd.tmp.form$v[i] <- v
      hpd.tmp.form$err[i] <- e
    }

    hpd.res[[j]] <- hpd.tmp.form
    j = j+1
  }

}
```

```r
# Comparaison
i.optimum <- NA
C.optimum <- 1

#T1 = 1694
#T2 = 1725

C1 = 86.5
ecart1 <- 3

C2 = 7.6
ecart2 <- 3

str.res = NULL
# comparaison des confiances
for (i in 1:(length(hpd.res)-1)) {
  for (j in 1:length(hpd.res[[i]]$conf) ) {

    if ( abs( hpd.res[[i]]$conf[j] - C1) <= ecart1 ) {
      str.res1 <-  paste0( i," ", j, " v=", hpd.res[[i]]$v[j], " err=",hpd.res[[i]]$err[j] , " conf", j
        if ( (j+1<=length(hpd.res[[i]]$conf)) &&  ( abs( hpd.res[[i]]$conf[j+1] - C2) <= ecart2) ) {
          str.res1 <-  paste0(str.res1, " conf", j+1, "=", hpd.res[[i]]$conf[j+1], "% ", hpd.res[[i]]$tr
          if (abs( hpd.res[[i]]$conf[j] - C1) * abs( hpd.res[[i]]$conf[j+1] - C2) <= C.optimum) { # Mem
              C.optimum <- abs( hpd.res[[i]]$conf[j] - C1) * abs( hpd.res[[i]]$conf[j+1] - C2)
              i.optimum<- i
            }
        }

    }

    else if ( abs( hpd.res[[i]]$conf[j] - C2) <= ecart2 ) {
        str.res1 <-  paste0( i," ", j, " v=", hpd.res[[i]]$v[j], " err=",hpd.res[[i]]$err[j] , " conf",


        if ( (j-1>=1) &&  ( abs( hpd.res[[i]]$conf[j-1] - C1) <= ecart1) ) {
          str.res1 <-  paste0(str.res1, " conf", j-1, "=",  hpd.res[[i]]$conf[j-1], "% ", hpd.res[[i]]$

          if (abs( hpd.res[[i]]$conf[j-1] - C1) * abs( hpd.res[[i]]$conf[j] - C2) < C.optimum) { # Memo
              C.optimum <- abs( hpd.res[[i]]$conf[j-1] - C1) * abs( hpd.res[[i]]$conf[j] - C2)
              i.optimum<- i
            }

        }

    } else {
      str.res1 <- NULL
    }

     str.res <- c(str.res, str.res1)
  }

}
# str.res
#i.optimum
hpd.res[[i.optimum]]
print(paste0(str.res1, " Age =",  hpd.res[[i.optimum]]$v[1], " BC/AD err=", hpd.res[[i.optimum]]$err[1]

for (i in 1:length(hpd.res[[i.optimum]]$conf)) {
  print(paste0(str.res1, " conf", i, " -> ",  hpd.res[[i.optimum]]$conf[i], "% ", hpd.res[[i.optimum]]$
}
```

```r
C14 <- 10506 #hpd.res[[i.optimum]]$v[1]
errC14 <- 50 #hpd.res[[i.optimum]]$err[1]
date14C <- calibrate(mesures = C14 ,std = errC14, calCurves = 'IntCal13_AD', ids = 'C14', timeScale = 1

# Tracé de la courbe 14C
 # réduction de la période
tmin <- -11000
tmax <- 1950
xlim <- c(tmin, tmax)

# Il faut mettre des valeurs qui existent dans le tableau
imin <- which(intCal13_AD$V1 == tmin)
```

```
imax <- which(intCal13_AD$V1 == tmax)
ylim <- range(intCal13_AD$V2[imin:imax] )

par( fig=c(0, 1, 0.50, 1), mar=c(0, 5, 0, 1))

courbe_enveloppe(t=intCal13_AD$V1, mean=intCal13_AD$V2, intCal13_AD$V3, ylab = '14C', xlab = NA,  xaxt =
                 xlim = xlim, ylim = ylim)
mesure_enveloppe(intCal13_AD$V1, mesure = C14, std = errC14)
text(intCal13_AD$V1[imin], C14, labels=as.character(C14) )

par(fig=c(0, 1, 0.0, 0.50), new= TRUE, mar=c(5, 5, 0, 1) )

plot(date14C, col = "blue", hdrCol = adjustcolor( "blue", alpha.f = 0.2), main = NA, xlab ='BC/AD',
     xlim = xlim, yaxt="n")
```



```
print('Resultat pour le 14C')
```

```
## [1] "Resultat pour le 14C"
```

```
hpd(date14C$C14, prob = .955)
```

```
## $`86.5%`
## [1] -10684 -10426
##
## $`7.6%`
## [1] -10396 -10285
##
## $`0.8%`
## [1] -10250 -10232
##
## $`0.6%`
## [1] -10229 -10215
```